INOVANCE

Advancing industrial technology,
for a better world



iFA Information System

Industrial Automation | New Energy Vehicle | Intelligent Elevator | Intelligent Robot | Digital Energy | Rail Transit

Preface

Introduction

This guide describes basic functions, product configuration in programming applications, instruction examples, and fault codes of the iFA software.

Target audience

This guide is intended for:

- Electrical engineers
- Software engineers
- System engineers

Revision history

Date	Document Version	Software Version	Description of Changes
2026-07	A15	V1.5.2.5	For detailed changes, see New Features in V1.5.2.5.
2026-06	A14	V1.4.4.5.Patch.20260610	**Modified content** • Removed view restrictions for online variable tables, supporting viewing all variables by mouse scrolling. For details, see visible element range setting. • CFC language can be used for FBs/FCs. For details, see CFC Language Programming. • SFC language can be used for the full series products. For details, see SFC Language Programming. • The Modbus TCP port number range has been expanded to 1-65535. For details, see Configuring a Modbus TCP Slave/Modbus TCP Slave Configuration and Configuring a Modbus TCP Slave Under the Master/Modbus TCP Slave Configuration. • Added the instructions of accumulation timer and sequencer output. For details, see AccumulationTimer2 and SQO. • The general instruction of delay timer supports modification of the ET pin. For details, see TON2. • Added the applicable EVO510 series PLCs. • Corrected minor errors and website related issues.

2026-05	A13	V1.4.4.5	• Added Fins communication instructions. For details, see FINS Communication. • Added Sftp communication instructions. For details, see SFTP Communication. • Added Melsec communication instructions. For details, see Melsec Communication. • Added program trace function. For details, see Loading Fault/Data Trace Files, SysHC_FaultRecord and SysHC_OperationRecord. • Added SFTP server. For details, see SFTP Server Configuration. • Added EtherCAT master system variables. For details, see EtherCAT master system variables. • Added BASE64 encoding instruction. For details, see AnyBASE64. • Added duplicate name checking for new variables against library variables and POUs. For details, see program check. • Added system information instruction. For details, see SysHC_AppDownInfo. • Added advanced process control instructions. For details, see PFCs_v2, PFCm_v2, PFCs_dv_v2, PFCm_dv, PIDR, PFCm22, and PRBS. • Added the instruction of enabling or disabling axes in an axis group. For details, see MC_GroupPower. • Added "HIC" configuration to the mechanical types for axis groups. For details, see Axis Configuration and HIC Configuration Example. • Added default gateway settings to EVO400 series. For details, see Ethernet Address Settings. • Added the feature of EIP quick connection to EVO700C series. For details, see Network Port Configuration. • Added HMI security authentication policies. For details, see HMI Security Authentication Policies.

Date	Document Version	Software Version	Description of Changes
			Modified content • Updated the number of OPC UA exposed variables for EVO500 series to 100–5000. For details, see Configuring OPC UA Server. • Updated the number of key points in the cam table to 20,000. • In the rotary axis mode, the servo drive can operate in either multi-turn or single-turn mode. For details, see Position Retention upon Power Failure.
2026-04	A12	V1.4.4.5	**New content** • Added the SFC language to programming languages. For details, see CFC Language Programming. • Added Ethernet expansion module property configuration. For details, see Ethernet Expansion Module Property Configuration. • Added PRP monitoring. For details, see PRP Monitoring. • Added EVO400 series related functions. **Modified content** Corrected minor errors and website related issues.
2026-03	A11	V1.4.4.5	**Modified content** Corrected minor errors and website related issues.
2026-02	A10	V1.4.4.5	**Modified content** Corrected minor errors and website related issues.
2026-01	A09	V1.4.4.5	For detailed changes, see New Features in V1.4.4.5.
2026-01	A08	V1.4.3.5	**Modified content** Corrected minor errors and website related issues.
2025-12	A07	V1.4.3.5	**Modified content** Corrected minor errors and website related issues.

Date	Document Version	Software Version	Description of Changes
2025-11	A06	V1.4.3.5	**New content** • Added password complexity setting policies. For details, see Setting Password Policies. • Added screen zoom level setting. For details, see Zooming a Screen. **Modified content** Corrected minor errors and website related issues.
2025-10	A05	V1.4.3.5	For detailed changes, see New Features in V1.4.3.5.
2025-10	A04	V1.4.2.5	**Modified content** Corrected minor errors and website related issues.
2025-09	A03	V1.4.2.5	**Modified content** Corrected minor errors and website related issues.
2025-08	A02	V1.4.2.5	**New content** • Added information related to safety PLC functions. For details, see Safety PLC Programming. • Added functions related to EVO800 and EVO700 series. For details, see the content with model description.
2025-07	A01	V1.4.2.5	For detailed changes, see New Features in V1.4.2.5.
2025-02	A00	V1.3.4.5	First release.

Access to the guide

This guide is not delivered with the product. You can obtain the PDF version in the following ways:

- Visit https://www.inovance.com/global to download the PDF documents.
- Get it directly from the help system of the software.

Contents

1 Legal Information 78

2 Version Information 79

2.1 New Features in V1.5.2.5 79

2.1.1 Introduction 79

2.1.2 Newly Supported Hardware Models 79

2.1.3 New Software Features 80

2.2 New Features in V1.4.4.5 86

2.2.1 Introduction 86

2.2.2 Newly Supported Hardware Models 86

2.2.3 New Software Features 87

2.3 New Features in V1.4.3.5 92

2.3.1 Introduction 92

2.3.2 Newly Supported Hardware Models 92

2.3.3 New Software Features 92

2.4 New Features in V1.4.2.5 96

2.4.1 Introduction 96

2.4.2 Newly Supported Hardware Models 96

2.4.3 New Software Features 97

2.5 Compatibility 101

2.6 Feature Upgrade 101

3 Introduction to iFA Evolution 104

3.1 Software Overview 104

3.2 Features 104

3.3 Help and Support 104

3.3.1 Help System 104

3.3.2 Online Help 106

3.3.3 Issue Feedback 108

3.3.4 Technical Forum 109

3.3.5 About iFA 110

3.4 AI Assistant 111

4 Installation and Uninstallation 115

4.1 Operating Environment Requirements 115

4.2 Installing the Software 115

4.3 Uninstalling the Software 117

4.4 Licensing the Software 118

4.5 Upgrading the Software 119

5 Software Interface Layout 123
5.1 Opening and Exiting the Software 123
5.2 Project View 123
5.3 Toolbar 127
5.4 Project Tree 129
5.5 Workspace 133
5.5.1 Workspace Feature and Layout 133
5.5.2 Splitting the Workspace 134
5.5.3 Moving to Previous/Next Group 136
5.5.4 Floating an Element 136
5.5.5 Maximizing an Element 137
5.5.6 Switching Between Elements 138
5.5.7 Closing an Element 138
5.5.8 Inverting Object Selection 139
5.6 Monitoring Area 139
5.6.1 Monitoring Area Feature and Layout 139
5.6.2 Floating a Tab 140
5.6.3 Maximizing a Tab 141
5.7 Toolbox 142
5.7.1 Toolbox Features and Layout 142
5.7.2 Floating a Tab 144
5.7.3 Maximizing a Tab 145
6 System Settings 146
6.1 General Settings 146
6.2 HMI Visualization 148
6.3 PLC Programming 149
7 Project Deployment 156
7.1 Creating a Project 156
7.2 Configuring a Device 157
7.2.1 Adding a Device 157
7.2.2 Copying a Device 164
7.2.3 Renaming a Device 165
7.2.4 Deleting a Device 166
7.2.5 Replacing a Device 167
7.3 Configuring a Network 169
7.3.1 HMI Ethernet Settings 170
7.3.2 Online Network Settings 171

8 Project Management 174
8.1 Opening a Project 174
8.2 Saving a Project 174
8.3 Saving a Project As 175
8.4 Backing up a Project 175
8.5 Closing a Project 175
8.6 Removing a Project 176
8.7 Finding and Replacing Within a Project 178
8.8 Reuse by Copy and Paste 180
8.9 Packaging a Project 182
8.10 Importing a Project 183
8.11 Upgrading a Project 183
8.12 Importing/Exporting a PLCopenXML File 185
8.13 Migrating a Project 189
8.14 Collapsing Project Child Nodes with One Click 190
8.15 Using the Device Library 191
8.15.1 Installing a Device 191
8.15.2 Exporting a Device 191
8.15.3 Uninstalling a Device 192
8.15.4 Device Details 193
9 Project Security Management 195
9.1 Overview 195
9.2 Feature Permissions in User Management 197
9.3 Enabling Project Protection 200
9.4 Disabling Project Protection 202
9.5 Setting Password Policies 202
9.6 HMI Security Authentication Policies 203
9.7 Managing Users 208
9.8 Managing Roles 209
9.9 Assigning Roles 211
9.10 Opening a Protected Project 213
9.11 Changing the Password 214
9.12 Switching the User 215
9.13 Exiting a Project 215
9.14 PLC Permissions 216
9.15 Library Project Security Management 217

10 PLC Programming 220
10.1 Programming Basics 220
10.1.1 Variables 220
10.1.1.1 Global Variables 220
10.1.1.2 Local Variables 222
10.1.1.3 Temporary Variables 223
10.1.1.4 Instance Variables 224
10.1.1.4.1 Input Variables 225
10.1.1.5 Output Variables 225
10.1.1.6 Static Variables 226
10.1.1.7 Input/Output Variable 228
10.1.1.8 Data Types (EVO400, EVO500, EVO700, EVO800) 229
10.1.1.8.1 Standard Data Types 229
10.1.1.8.2 User-Defined Types 231
10.1.1.8.2.1 STRUCT 231
10.1.1.8.2.2 UNION 238
10.1.1.8.2.3 ENUM 240
10.1.1.8.2.4 ALIAS 242
10.1.1.8.3 Complex Types 244
10.1.1.8.3.1 ARRAY 244
10.1.1.8.3.2 POINTER 246
10.1.1.8.4 BIT 247
10.1.1.8.5 ANY 247
10.1.1.8.6 REFERENCE 250
10.1.1.8.7 Data Type Conversion 251
10.1.1.9 Variable Declaration 252
10.1.1.9.1 Adding a Global Variable Table 252
10.1.1.9.2 Adding a Param Table 253
10.1.1.9.3 Variable Declaration Properties 254
10.1.1.9.4 Variable Address Range 258
10.1.1.9.5 Default Variable Table 258
10.1.1.9.6 System Variable Table 258
10.1.1.9.7 Power Failure Retention Table 264
10.1.1.9.8 Resource Usage Table 267
10.1.1.10 Variable Operations 269
10.1.1.10.1 Adding a Variable 269
10.1.1.10.2 Copying and Pasting a Variable Table 270
10.1.1.10.3 Setting Variable Properties 271

10.1.1.10.4 Changing Variable Category 272
10.1.1.10.5 Cross-Reference 272
10.1.1.10.6 Variable Import 274
10.1.1.10.7 Variable Export 275
10.1.1.10.8 Smart Input 276
10.1.1.10.9 Parameter Update 278
10.1.1.10.10 Input Assistant 280
10.1.1.10.11 Public Variable Export 282
10.1.1.10.12 Going to Definition 283
10.1.1.10.13 Going to Next Level 284
10.1.2 Program Block 285
10.1.2.1 Program 285
10.1.2.1.1 Overview 285
10.1.2.1.2 Adding a User Program 286
10.1.2.1.3 Adding a Method to a Program 288
10.1.2.1.4 Adding an Action to a Program 292
10.1.2.1.5 Calling a User Program 293
10.1.2.2 Functions 294
10.1.2.2.1 Adding a User Function 294
10.1.2.2.2 Modifying a User Function 295
10.1.2.3 Function Block 297
10.1.2.3.1 Adding a User Function Block 297
10.1.2.3.2 Adding a Method to a Function Block 298
10.1.2.3.3 Adding an Action to a Function Block 298
10.1.2.3.4 Adding a Property to a Function Block 298
10.1.2.3.5 Modifying a User Function Block 301
10.1.2.3.6 Using a User Function Block 302
10.1.2.4 Program Block Callee Update 309
10.1.2.5 Program Block Access Permissions 311
10.1.2.6 Program Block Cross-Reference 314
10.1.2.7 Viewing Call Relationships Between Program Blocks 315
10.1.3 LD Language Programming 316
10.1.3.1 LD Language Overview 316
10.1.3.2 LD Creation and Management 317
10.1.3.2.1 Creating an LD Program Block 317
10.1.3.2.2 Managing an LD Program Block 318
10.1.3.2.2.1 Copying an LD Program Block 318
10.1.3.2.2.2 Renaming an LD Program Block 318

10.1.3.2.2.3 Deleting an LD Program Block 319
10.1.3.3 LD Editor 320
10.1.3.3.1 LD Editor Interface 320
10.1.3.3.2 LD Editor Settings 323
10.1.3.3.3 Lines in LD Editor 326
10.1.3.3.4 LD Elements 327
10.1.3.3.5 LD Instructions 330
10.1.3.3.5.1 Up 330
10.1.3.3.5.2 Down 331
10.1.3.3.5.3 = ... 333
10.1.3.3.5.4 >= 334
10.1.3.3.5.5 > 335
10.1.3.3.5.6 <= 337
10.1.3.3.5.7 < 338
10.1.3.3.5.8 <> 339
10.1.3.4 Network Usage 341
10.1.3.4.1 Network Overview 341
10.1.3.4.2 Adding a Network 341
10.1.3.4.3 Inserting a Network 342
10.1.3.4.4 Moving a Network 343
10.1.3.4.5 Inserting a Label 344
10.1.3.4.6 Modifying Network Comment 345
10.1.3.4.7 Switching Network Comment Status 345
10.1.3.4.8 Collapsing/Expanding a Network 346
10.1.3.4.9 Deleting a Network 346
10.1.3.5 Branch Usage 347
10.1.4 ST Language Programming 348
10.1.4.1 Overview 348
10.1.4.2 ST Creation and Management 349
10.1.4.2.1 Creating an ST Program Block 349
10.1.4.2.2 Managing an ST Program Block 349
10.1.4.2.2.1 Copying an ST Program Block 349
10.1.4.2.2.2 Renaming an ST Program Block 350
10.1.4.2.2.3 Deleting an ST Program Block 350
10.1.4.3 ST Editor 351
10.1.4.3.1 ST Editor Interface 351
10.1.4.3.2 ST Editor Settings 353
10.1.4.3.3 Basic Operations of ST Editor 354

10.1.4.4 ST Operators 357
10.1.4.5 ST Keywords 359
10.1.4.6 ST Statements 360
10.1.5 CFC Language Programming 372
10.1.5.1 CFC Language Overview 372
10.1.5.2 CFC Creation and Management 372
10.1.5.2.1 Creating a CFC Program Block 372
10.1.5.2.2 Creating a CFC Partition 373
10.1.5.2.3 Managing a CFC Program Block and Group 375
10.1.5.2.3.1 Adding a Group 375
10.1.5.2.3.2 Copying a CFC Block/Group 375
10.1.5.2.3.3 Renaming a CFC Block/Group 376
10.1.5.2.3.4 Deleting a CFC Program Block 377
10.1.5.2.3.5 Deleting a Group 378
10.1.5.3 CFC Editor 378
10.1.5.3.1 CFC Editor Interface 378
10.1.5.3.2 CFC Editor Settings 380
10.1.5.3.3 CFC Program Configuration Options 380
10.1.5.4 CFC Object Creation and Management 381
10.1.5.4.1 Creating an Operation Block 381
10.1.5.4.2 Creating a Comment 384
10.1.5.4.3 Managing an Operation Block 385
10.1.5.4.3.1 Configuring Operation Block Properties 385
10.1.5.4.3.2 Reordering an Operation Block 387
10.1.5.4.3.3 Resizing an Operation Block 389
10.1.5.4.3.4 Repositioning an Operation Block 389
10.1.5.4.3.5 Deleting an Operation Block 390
10.1.5.4.3.6 Updating an Operation Block 390
10.1.5.4.3.7 Operation Block Operations 392
10.1.5.5 Input/Output Parameter Connection 393
10.1.5.5.1 Setting Input/Output Parameters 393
10.1.5.5.2 Adding an Input Parameter 394
10.1.5.5.3 Connection Routing 394
10.1.5.5.4 Connecting Input and Output Parameters 397
10.1.5.5.5 Modifying a Connection 398
10.1.5.5.6 Changing Connection Line Color 398
10.1.5.5.7 Deleting a Connection 399
10.1.5.5.8 Text Connection 400

10.1.5.6 CFC Configuration and Commissioning 400
10.1.5.6.1 Execution Order Between CFC Blocks 400
10.1.5.6.2 Execution Order of Operation Blocks within a CFC Block 401
10.1.5.6.3 CFC Instance Call 401
10.1.5.6.4 CFC Commissioning 401
10.1.6 SFC Language Programming 402
10.1.6.1 SFC Language Overview 402
10.1.6.2 SFC Creation and Management 402
10.1.6.2.1 Creating an SFC Program Block 402
10.1.6.2.2 Managing an SFC Program Block 403
10.1.6.2.2.1 Copying an SFC Program Block 403
10.1.6.2.2.2 Renaming an SFC Program Block 404
10.1.6.2.2.3 Deleting an SFC Program Block 405
10.1.6.3 SFC Editor 405
10.1.6.3.1 SFC Editor Interface 405
10.1.6.3.2 SFC Editor Settings 407
10.1.6.3.2.1 Setting the SFC Editor in iFA Options 407
10.1.6.3.2.2 Setting SFC Program Block Properties 408
10.1.6.3.2.3 Setting SFC Program Block Power Failure Retention Property 410
10.1.6.3.2.4 Setting SFC Program Block Interface Mode 410
10.1.6.3.3 SFC Programming Window 411
10.1.6.3.4 SFC Elements 412
10.1.6.3.5 Opening/Closing All Program Segments 418
10.1.6.3.6 SFC Program Block Interface Parameters 420
10.1.6.3.6.1 Overview 420
10.1.6.3.6.2 Interface Parameter Mode 420
10.1.6.3.6.3 Input Parameters 421
10.1.6.3.6.4 Output Parameters 422
10.1.6.3.6.5 Static Parameters 423
10.1.6.4 Step and Transition Condition 425
10.1.6.4.1 Step and Transition 425
10.1.6.4.1.1 Step Overview 425
10.1.6.4.1.2 Transition Overview 425
10.1.6.4.2 Step 425
10.1.6.4.2.1 Step Basic Properties 425
10.1.6.4.2.2 Step Status 427
10.1.6.4.2.3 Condition 427
10.1.6.4.2.4 Action 428
10.1.6.4.2.5 Monitoring 431

10.1.6.4.3 Transition 431
10.1.6.4.3.1 Transition Basic Properties 431
10.1.6.4.3.2 Transition Status 432
10.1.6.5 Step and Transition Usage 433
10.1.6.5.1 Adding a Step and Transition 433
10.1.6.5.2 Adding a Step 433
10.1.6.5.3 Adding a Transition 434
10.1.6.5.4 Deleting a Step and Transition 435
10.1.6.6 Branch Usage 435
10.1.6.6.1 Adding a Selective Branch 435
10.1.6.6.2 Adding a Parallel Branch 438
10.1.6.6.3 Closing a Branch 440
10.1.6.6.4 Deleting a Branch 444
10.1.6.7 Jump Usage 445
10.1.6.7.1 Adding a Jump 445
10.1.6.7.2 Changing Jump Target 446
10.1.6.7.3 Deleting a Jump 446
10.1.6.8 SFC Configuration and Commissioning 447
10.1.6.8.1 SFC Instantiation 447
10.1.6.8.2 SFC Commissioning 448
10.2 Oscilloscope 450
10.2.1 Basic Oscilloscope 450
10.2.1.1 Adding a Basic Oscilloscope 450
10.2.1.2 Variable Area 452
10.2.1.3 Plot Area 460
10.2.1.4 Loading Fault/Data Trace Files 464
10.2.2 Task Oscilloscope 465
10.3 Recipe 468
10.3.1 Configuring a Recipe 468
10.3.2 Adding a Recipe 473
10.3.3 Online Recipe 475
10.4 Watch Table 477
10.4.1 Monitoring Variable Table 477
10.4.2 Monitoring Specified Variables 480
10.4.3 Drive Watch Table 481
10.4.4 Derivative Watch Table 484
10.5 Task Configuration 486
10.5.1 Adding a Task 486
10.5.2 Configuring Task Parameters 488

10.5.3 System Event Management 492
10.6 PLC Configuration 494
10.6.1 Hardware Configuration and Device Management 494
10.6.1.1 Device Scanning 494
10.6.1.2 Viewing Device ID 496
10.6.1.3 Exporting I/O Mapping 496
10.6.1.4 Importing I/O Mapping 497
10.6.2 Device and Network Editor 498
10.6.2.1 Overview 498
10.6.2.2 Network View 499
10.6.2.3 Device View 501
10.6.2.4 Device List 503
10.6.3 EtherCAT Communication Configuration (EVO500, EVO700, APC500, EVO700C) 505
10.6.3.1 Adding an EtherCAT Device 505
10.6.3.1.1 Adding a Master 505
10.6.3.1.2 Adding a Slave 506
10.6.3.1.3 Adding a Wireless Management Station 507
10.6.3.1.4 Adding an Online EtherCAT Slave to Project 509
10.6.3.2 Configuring Master Parameters 511
10.6.3.2.1 General Settings 511
10.6.3.2.2 Parameters 517
10.6.3.2.3 Upgrade 517
10.6.3.2.4 Unit Synchronization Assignment 518
10.6.3.2.5 Online 518
10.6.3.2.6 Overview 519
10.6.3.2.7 Machine Parameter Management 520
10.6.3.2.7.1 Machine Recipe Management 520
10.6.3.2.7.2 Machine Parameter Management 523
10.6.3.2.7.3 Batch Firmware Upgrade 527
10.6.3.2.8 I/O Mapping 528
10.6.3.2.9 System Variables 529
10.6.3.2.10 Information 530
10.6.3.3 Configuring Slave Parameters 530
10.6.3.3.1 General Settings 530
10.6.3.3.2 Process Data (PDO Configuration) 533
10.6.3.3.3 Startup Parameters (SDO Configuration) 534
10.6.3.3.4 Online 536
10.6.3.3.5 Online CoE 538

10.6.3.3.6 EoE Settings 538
10.6.3.3.7 Servo Parameters 540
10.6.3.3.8 ESC Register 541
10.6.3.3.9 I/O Mapping 542
10.6.3.3.10 System Variables 545
10.6.3.3.11 Information 546
10.6.3.4 Disabling/Enabling an EtherCAT Device 546
10.6.4 EtherNet Communication Configuration 548
10.6.4.1 Modbus TCP Device Configuration 548
10.6.4.1.1 Adding a Modbus TCP Device 548
10.6.4.1.2 Configuring a Modbus TCP Master 550
10.6.4.1.2.1 Modbus TCP Master Configuration 550
10.6.4.1.2.2 System Variables 551
10.6.4.1.2.3 Information 551
10.6.4.1.3 Configuring a Modbus TCP Slave 551
10.6.4.1.3.1 Modbus TCP Slave Settings 551
10.6.4.1.3.2 System Variables 552
10.6.4.1.3.3 Information 552
10.6.4.1.4 Configuring a Modbus TCP Slave Under the Master 553
10.6.4.1.4.1 Modbus TCP Slave Settings 553
10.6.4.1.4.2 Modbus TCP Slave Communication Settings 553
10.6.4.1.4.3 I/O Mapping 555
10.6.4.1.4.4 System Variables 555
10.6.4.1.4.5 Information 556
10.6.4.1.5 Modbus TCP Variable Addressing 556
10.6.4.1.6 Modbus TCP Communication Frame Format 558
10.6.4.1.7 Common Modbus TCP Faults 564
10.6.4.2 EtherNet/IP Device Configuration 566
10.6.4.2.1 Adding an EtherNet/IP Device 566
10.6.4.2.2 EtherNet_IP_Scanner Configuration 569
10.6.4.2.2.1 Scanner Settings 569
10.6.4.2.2.2 General Settings of Remote Adapters 570
10.6.4.2.2.3 I/O Mapping 572
10.6.4.2.3 EtherNet_IP_Adapter Configuration 572
10.6.4.2.3.1 General Settings of Local Adapters 572
10.6.4.2.3.2 I/O Mapping 573
10.6.4.2.3.3 Connection Status 573
10.6.4.2.3.4 Information 574

10.6.4.2.4 Connection Configuration (Class1) 574
10.6.4.2.5 Communication Between Producer and Consumer Labels 577
10.6.4.2.6 Communication Between PLCs 580
10.6.4.2.7 Communication with HMI 584
10.6.4.2.8 Quick Connect Communication (EVO700, EVO700C, EVO800, APC760C) 590
10.6.4.3 Free Network Port Communication 593
10.6.4.4 Disabling/Enabling an EtherNet Device 595
10.6.5 PROFINET Communication Configuration (EVO400, EVO500, EVO700, APC500, EVO800)\ 596
10.6.5.1 Overview 596
10.6.5.2 PN Slave Configuration(EVO400,EVO500,EVO700,APC500) 597
10.6.5.2.1 PN Slave Configuration 597
10.6.5.2.2 PROFINET I/O Module 601
10.6.5.2.3 TIA Portal and PN Master Configuration 602
10.6.5.2.4 PN Slave Troubleshooting 606
10.6.5.3 PN Master Configuration (EVO800) 607
10.6.5.3.1 PN Master Configuration (SYCON.net) 607
10.6.5.3.2 PN Master Configuration (iFA) 613
10.6.5.3.3 PN Master Configuration 616
10.6.5.3.3.1 General Configuration 616
10.6.5.3.3.2 I/O Mapping 617
10.6.5.3.3.3 Information 617
10.6.6 CANopen Communication Configuration (EVO400, EVO500, EVO700, APC500) 618
10.6.6.1 Usage Workflow 618
10.6.6.2 CANopen Master Configuration 621
10.6.6.2.1 General Settings 621
10.6.6.2.2 I/O Mapping 624
10.6.6.2.3 System Variables 624
10.6.6.2.4 Information 624
10.6.6.3 CANopen Slave Configuration 624
10.6.6.3.1 General Settings 624
10.6.6.3.2 Process Data (PDO) 627
10.6.6.3.3 Startup Parameter (SDO) 630
10.6.6.3.4 Online 631
10.6.6.3.5 I/O Mapping 633
10.6.6.3.6 System Variables 634
10.6.6.3.7 Information 634
10.6.6.4 CANopen Module 634
10.6.6.5 CANopen Parameter Configuration 634

10.6.6.6 Programming Interface 635
10.6.7 CAN Free Communication Configuration (EVO400, EVO500, EVO700, APC500) 635
10.6.7.1 Network Configuration (EVO400, EVO500, APC500) 635
10.6.7.2 Network Configuration (EVO700) 637
10.6.7.3 CAN Free Protocol Configuration 638
10.6.7.4 Usage Example 638
10.6.8 COM Communication Configuration (EVO400, EVO500, EVO700, APC500) 640
10.6.8.1 Modbus RTU Device Configuration 640
10.6.8.1.1 Adding a Modbus RTU Device 640
10.6.8.1.2 Configuring a Modbus RTU Master 642
10.6.8.1.2.1 Modbus RTU Master Configuration 642
10.6.8.1.2.2 Information 643
10.6.8.1.3 Configuring a Modbus RTU Slave 643
10.6.8.1.3.1 Modbus RTU Slave Configuration 643
10.6.8.1.3.2 Modbus RTU Slave Communication Configuration 644
10.6.8.1.3.3 System Variables 645
10.6.8.1.3.4 Information 645
10.6.8.1.4 Modbus RTU Variable Addressing 645
10.6.8.1.5 Modbus RTU Communication Frame Format 648
10.6.8.1.6 Common Modbus RTU Faults 652
10.6.8.2 Free Serial Communication 652
10.6.8.2.1 Adding Serial_Port_Free 652
10.6.8.2.2 Configuring Serial_Port_Free 653
10.6.8.2.3 Transmitting/Receiving Data Through Free Serial Port 654
10.6.8.3 Disabling/Enabling a COM Device 655
10.6.9 FINS TCP&UDP Communication Configuration 656
10.6.9.1 Overview 656
10.6.9.2 FINS Parameters and Addresses 656
10.6.9.3 FINS Protocol Configuration 657
10.6.10 DP Communication Configuration (EVO800) 657
10.6.10.1 DP Master Configuration (SYCON.net) 657
10.6.10.2 DP Master Configuration (iFA) 666
10.6.10.3 DP Master Configuration 668
10.6.10.3.1 General Configuration 668
10.6.10.3.2 I/O Mapping 669
10.6.10.3.3 Information 669
10.6.11 I/O Module Configuration 670
10.6.11.1 Rack Overview 670
10.6.11.2 Adding a Local I/O Module 673

10.6.11.3 Adding a Remote I/O Module 675
10.6.11.4 Deleting a Module 679
10.6.11.5 Inserting a Module 679
10.6.11.6 Replacing a Module 679
10.6.11.7 Moving a Module 680
10.6.11.8 Disabling/Enabling an I/O Module 680
10.6.12 High Speed I/O Module Configuration (EVO400, EVO500, EVO700) 681
10.6.12.1 Opening High-Speed I/O Module Interface 681
10.6.12.2 Configuring a High-Speed I/O Module 682
10.6.12.3 Disabling/Enabling a High-Speed I/O Module 684
10.6.13 EXPA/EXPB (Local Expansion Module) Configuration (EVO400, EVO500, APC500) 685
10.6.13.1 Adding/Deleting a Local Expansion Module 685
10.6.13.2 Disabling/Enabling a Local Expansion Module 686
10.6.14 PLC Property Configuration 687
10.6.14.1 I/O Settings (EVO400, EVO500, EVO700) 687
10.6.14.2 Network Settings (EVO500) 688
10.6.14.3 Network Port Switching 689
10.6.14.4 Ethernet Address Settings 692
10.6.14.5 OPC UA Server Configuration 695
10.6.14.5.1 Overview 695
10.6.14.5.2 Configuring OPC UA Server 696
10.6.14.5.3 OPC UA Diagnostics 702
10.6.14.5.4 Fault Codes 708
10.6.14.6 SFTP Server Configuration 710
10.6.14.7 Web Server Configuration 712
10.6.15 Ethernet Expansion Module Property Configuration 715
10.6.15.1 Setting Ethernet Address 715
10.6.15.2 Setting PRP 717
10.6.15.3 Configuring OPC UA Server 719
10.6.15.4 Configuring Web Server 720
10.7 PLC Compilation 720
10.7.1 Overview 720
10.7.2 Pre-Check 720
10.7.3 Compilation 721
10.7.4 Recompilation 723
10.7.5 Clearing 725
10.7.6 Clearing All 727

10.7.7 Option Configuration 727
10.7.7.1 Compilation Options 727
10.7.7.2 Public Variables 728
10.8 PLC Download 729
10.8.1 PLC Connection 729
10.8.2 Download Permission Check 732
10.8.3 Full Download 732
10.8.4 Online Modification/Monitoring without Modification 736
10.8.5 Redundant System Download (EVO800) 739
10.9 PLC Upload 741
10.9.1 Uploading 741
10.9.2 Uploading with a New Device 743
10.10 PLC Commissioning 745
10.10.1 PLC Start/Stop 745
10.10.2 PLC Online/Offline 747
10.10.3 Value Display and Calculation 749
10.10.3.1 Writing a Value 749
10.10.3.2 Forcing a Value 750
10.10.3.3 Releasing a Value 751
10.10.3.4 Display Modes 752
10.10.4 Breakpoint Commissioning 754
10.10.4.1 Enabling a Breakpoint 754
10.10.4.2 Modifying a Breakpoint 757
10.10.4.3 Deleting a Breakpoint 760
10.10.4.4 Disabling a Breakpoint 762
10.10.5 ST Program Commissioning 764
10.10.5.1 Step Execution 764
10.10.5.2 Running to Cursor 765
10.10.5.3 Displaying Current Statement 766
10.10.5.4 Stepping Out 766
10.10.5.5 Stepping Into 767
10.10.5.6 Executing One Scan Cycle 768
10.10.6 PID Function Block Commissioning Wizard 769
10.10.6.1 Overview 769
10.10.6.2 Adding a PID Function Block 769
10.10.6.3 Configuration and Commissioning 771
10.11 PLC Online Operations 778
10.11.1 Hot Reset 778

10.11.2 Cold Reset 779
10.11.3 Clearing User Program 780
10.11.4 Synchronizing Files to PLC 781
10.11.5 Network Settings 783
10.11.6 PLC Permission Configuration 785
10.11.7 WIFI Settings (EVO820) 789
10.11.8 Backing up Program to External Storage 792
10.11.9 Exporting Device Data 793
10.11.10 Power Failure Data Management 794
10.11.11 Restoring Factory Settings (EVO800, EVO1000) 798
10.11.12 Web Diagnostics and Monitoring 799
10.11.13 Time Settings 803
10.11.14 Task Monitoring 804
10.11.15 Time Synchronization Settings 805
10.11.16 DLR Ring Network Management (EVO800) 807
10.11.17 Ethernet Loop Monitoring 811
10.11.18 Monitoring Main/Backup Device Status (EVO800) 812
10.11.19 Calling a Stack 813
10.11.20 Diagnostic Information 814
10.11.21 Operation Log 815
10.11.22 Network Port Configuration 817
10.11.23 PRP Monitoring 820
10.11.24 Uploading/Downloading Permission Settings (EVO500) 822
10.12 PLC Project Comparison 823
10.12.1 Overview 823
10.12.2 Offline/Offline Comparison 825
10.12.3 Offline/Online Comparison 826
10.12.4 Viewing and Processing Comparison Results 827
10.13 PLC Simulation Mode 830
10.13.1 Simulation Panel Overview 830
10.13.2 Establishing a Simulation Environment 832
10.13.3 Simulation Function Description 835
10.13.4 Simulation Commissioning 838
10.13.4.1 OPCUA Communication Protocol Simulation 838
10.13.4.2 EtherNet/IP Communication Protocol Simulation 839
10.13.4.3 Modbus TCP Communication Protocol Simulation 841
10.13.4.4 EtherCAT Communication Protocol Simulation 841

10.14 Multi-User Collaboration 843
10.14.1 Overview 843
10.14.2 Multi-User Online Editing 844
10.14.3 Multi-User Download Synchronization 846
10.14.4 Multi-User Commissioning 848
10.14.5 Uploading Only Differences 849
11 Safety PLC Programming 851
11.1 Safety PLC Access Protection 851
11.1.1 PLC Configuration Write Protection 851
11.1.2 PLC Permission Configuration 851
11.2 Programming Basics 852
11.2.1 Variables 852
11.2.1.1 Data Types 852
11.2.1.1.1 Standard Data Types 852
11.2.1.1.2 Safety Data Types 852
11.2.1.1.3 User-Defined Types 853
11.2.1.1.4 Data Type Conversion 854
11.2.1.2 Variable Declaration 854
11.2.1.2.1 Variable Declaration Properties 854
11.2.1.2.2 Variable Address Range 856
11.2.1.2.3 Default Variable Table 856
11.2.1.2.4 System Variable Table 856
11.2.1.2.5 Resource Usage Table 857
11.2.1.3 Variable Operations 859
11.2.2 Library Reference 859
11.2.3 Program Block 860
11.2.3.1 Program 860
11.2.3.1.1 Adding a User Program 860
11.2.3.1.2 Calling a User Program 861
11.2.3.2 Function Block 862
11.2.3.2.1 Adding a User Function Block 862
11.2.3.2.2 Using a User Function Block 863
11.2.4 Safety LD Language Programming 864
11.2.4.1 Overview 864
11.2.4.2 Opening the Safety LD Editor 864
11.2.4.3 Setting the Safety LD Editor 865
11.2.4.4 Safety LD Elements 865
11.2.4.5 Safety LD Programming 869

11.2.4.6 Graphic Configuration Modification 873
11.2.4.6.1 Modifying Text Data 873
11.2.4.6.2 Modifying Block Type 874
11.2.4.6.3 Moving an Element 874
11.2.5 Safety FBD Language Programming 875
11.2.5.1 Overview 875
11.2.5.2 Opening the Safety FBD Editor 875
11.2.5.3 Setting the Safety FBD Editor 875
11.2.5.4 Safety FBD Elements 878
11.2.5.5 Safety FBD Programming 881
11.2.5.6 Graphic Configuration Modification 884
11.2.5.6.1 Modifying Text Data 884
11.2.5.6.2 Modifying Block Type 885
11.2.5.6.3 Moving an Element 886
11.3 Data Record Configuration 886
11.4 PLC Configuration 889
11.4.1 Hardware Configuration and Device Management 889
11.4.1.1 Adding a Safety PLC and Communication Unit 889
11.4.1.2 Device Scanning 891
11.4.2 Failsafe I/O Module Configuration 891
11.4.2.1 Rack Overview 891
11.4.2.2 Adding a Module 892
11.4.2.3 Deleting a Module 895
11.4.2.4 Inserting a Module 896
11.4.2.5 Replacing a Module 897
11.4.2.6 Moving a Module 898
11.4.2.7 Failsafe I/O Module Configuration 898
11.4.3 EtherNet/IP Communication Configuration 903
11.4.3.1 Standard EtherNet/IP Communication Configuration 903
11.4.3.1.1 EtherNet/IP Communication Between PLCs 903
11.4.3.1.2 EtherNet/IP Communication with HMI 906
11.4.3.2 Safety CIP Safety Communication Configuration 907
11.4.3.2.1 Overview 907
11.4.3.2.2 CIP Safety Communication Between PLCs 907
11.4.3.2.3 Adding a Remote CIP Safety Device 910
11.4.3.2.4 Configuring a Remote CIP Safety Device 913
11.4.3.3 Modbus TCP Communication Configuration 915

11.4.4 EtherCAT Communication Configuration 917
11.4.4.1 Configuring a Safety PLC as an EtherCAT Slave of a Standard PLC 917
11.4.4.1.1 Configuring an Independent Safety PLC as an EtherCAT Slave 918
11.4.4.1.2 Directly Creating a Safety PLC as an EtherCAT Slave 920
11.4.4.1.3 Configuring Communication Parameters of Safety PLC 920
11.4.4.2 Configuring I/O Modules 922
11.4.4.2.1 Adding a Local Failsafe I/O Module 922
11.4.4.2.2 Adding a Remote Failsafe I/O Module 923
11.4.4.2.3 Assigning FSoE Master to Failsafe I/O Modules 924
11.4.4.2.4 Reading FSoE DIP Switch Addresses of Failsafe I/O Modules 925
11.4.4.2.5 Configuring Failsafe I/O Module Parameters 927
11.4.4.3 Configuring Safety Servo Modules 928
11.4.4.4 FSoE Communication Between Safety PLCs 929
11.4.4.5 EtherCAT Communication Between Safety PLC and Standard PLC 932
11.4.5 PLC Property Settings 933
11.4.5.1 PLC Configuration 933
11.4.5.2 Ethernet Address and Safety Network Settings 934
11.4.5.3 Power-On Confirmation 935
11.4.5.4 Safety Information 936
11.4.6 Task Configuration 938
11.5 PLC Compilation 939
11.6 PLC Download 939
11.6.1 Full Download 939
11.7 PLC Commissioning 943
11.7.1 PLC Start/Stop 943
11.7.2 PLC Online/Offline 943
11.7.3 Value Display and Calculation 943
11.8 PLC Online Operations 944
11.8.1 Operation Modes 944
11.8.2 Safety Confirmation 948
11.8.3 Power-On Manual Confirmation 949
11.8.4 Operation Log 950
11.9 Safety PLC Simulation Mode 950
11.10 Common Safety Instructions 951
11.10.1 Basic Instructions 951
11.10.1.1 Comparison Operations 951
11.10.1.1.1 Instruction List 951
11.10.1.1.2 EQ 951
11.10.1.1.3 GE 952

11.10.1.1.4 GT 954
11.10.1.1.5 LE 956
11.10.1.1.6 LT 958
11.10.1.1.7 NE 960
11.10.1.2 Logic Operations 962
11.10.1.2.1 Instruction List 962
11.10.1.2.2 AND 962
11.10.1.2.3 NOT 964
11.10.1.2.4 OR 966
11.10.1.2.5 XOR 968
11.10.1.3 Mathematical Operations 970
11.10.1.3.1 Instruction List 970
11.10.1.3.2 ADD 971
11.10.1.3.3 SUB 973
11.10.1.3.4 MUL 976
11.10.1.3.5 DIV 979
11.10.1.4 Selection Operations 983
11.10.1.4.1 Instruction List 983
11.10.1.4.2 MUX 983
11.10.1.4.3 SEL 985
11.10.1.5 Shift Operations 989
11.10.1.5.1 Instruction List 989
11.10.1.5.2 ROL 989
11.10.1.5.3 ROR 991
11.10.1.5.4 SHL 993
11.10.1.5.5 SHR 995
11.10.1.6 Conversion Operators 997
11.10.1.6.1 Instruction List 997
11.10.1.6.2 TIME_TO_WORD 999
11.10.1.6.3 INT_TO_TIME 1001
11.10.1.6.4 DINT_TO_WORD 1003
11.10.1.6.5 TIME_TO_DINT 1005
11.10.1.6.6 TIME_TO_BYTE 1007
11.10.1.6.7 BOOL_TO_INT 1009
11.10.1.6.8 WORD_TO_INT 1011
11.10.1.6.9 INT_TO_BYTE 1013
11.10.1.6.10 INT_TO_BOOL 1015

11.10.1.6.11 DINT_TO_TIME 1017
11.10.1.6.12 DINT_TO_BYTE 1019
11.10.1.6.13 BOOL_TO_TIME 1021
11.10.1.6.14 BYTE_TO_DINT 1023
11.10.1.6.15 BOOL_TO_WORD 1025
11.10.1.6.16 BYTE_TO_WORD 1027
11.10.1.6.17 BYTE_TO_INT 1029
11.10.1.6.18 BOOL_TO_DINT 1031
11.10.1.6.19 BYTE_TO_TIME 1033
11.10.1.6.20 INT_TO_WORD 1035
11.10.1.6.21 BOOL_TO_BYTE 1037
11.10.1.6.22 DINT_TO_BOOL 1039
11.10.1.6.23 WORD_TO_BOOL 1041
11.10.1.6.24 BYTE_TO_BOOL 1043
11.10.1.6.25 INT_TO_DINT 1045
11.10.1.6.26 WORD_TO_BYTE 1047
11.10.1.6.27 WORD_TO_TIME 1049
11.10.1.6.28 WORD_TO_DINT 1051
11.10.1.6.29 DINT_TO_INT 1053
11.10.1.6.30 TIME_TO_INT 1055
11.10.1.6.31 TIME_TO_BOOL 1057
11.10.1.6.32 BOOL_TO_DWORD 1059
11.10.1.6.33 BYTE_TO_DWORD 1061
11.10.1.6.34 DINT_TO_DWORD 1063
11.10.1.6.35 DWORD_TO_BOOL 1065
11.10.1.6.36 DWORD_TO_BYTE 1067
11.10.1.6.37 DWORD_TO_DINT 1069
11.10.1.6.38 DWORD_TO_INT 1071
11.10.1.6.39 DWORD_TO_TIME 1073
11.10.1.6.40 DWORD_TO_WORD 1075
11.10.1.6.41 INT_TO_DWORD 1077
11.10.1.6.42 TIME_TO_DWORD 1079
11.10.1.6.43 WORD_TO_DWORD 1080
11.10.2 Safety Instructions 1082
11.10.2.1 Basic Safety Instruction 1082
11.10.2.1.1 Instruction List 1082
11.10.2.1.2 SF_Compare_DINT 1083

11.10.2.1.3 SF_Compare_INT 1086
11.10.2.1.4 SF_CTD 1090
11.10.2.1.5 SF_CTU 1090
11.10.2.1.6 SF_CTUD 1091
11.10.2.1.7 SF_F_TRIG 1093
11.10.2.1.8 SF_Limit_DINT 1093
11.10.2.1.9 SF_Limit_INT 1096
11.10.2.1.10 SF_R_TRIG 1098
11.10.2.1.11 SF_RS 1099
11.10.2.1.12 SF_Scaling_DINT 1099
11.10.2.1.13 SF_Scaling_INT 1102
11.10.2.1.14 SF_SR 1105
11.10.2.1.15 SF_TON 1106
11.10.2.1.16 SF_TOF 1107
11.10.2.1.17 SF_TP 1108
11.10.2.2 Safety Function Blocks 1109
11.10.2.2.1 Instruction List 1109
11.10.2.2.2 General Description of Safety Function Blocks 1111
11.10.2.2.2.1 General Input Variables of Safety Function Blocks 1111
11.10.2.2.2.2 General Output Variables of Safety Function Blocks 1112
11.10.2.2.2.3 General Representation of Safety Function Blocks 1113
11.10.2.2.2.4 Diagnostic Codes of Safety Function Blocks 1115
11.10.2.2.2.5 General Status Transition Diagram of Safety Function Blocks 1116
11.10.2.2.2.6 Precautions of Safety Function Blocks 1117
11.10.2.2.3 SF_Antivalent 1118
11.10.2.2.4 SF_CamMonitoring 1124
11.10.2.2.5 SF_CamshaftMonitor 1139
11.10.2.2.6 SF_CycleControl 1147
11.10.2.2.7 SF_EDM 1162
11.10.2.2.8 SF_EmergencyStop 1171
11.10.2.2.9 SF_EnableSwitch 1179
11.10.2.2.10 SF_Equivalent 1188
11.10.2.2.11 SF_ESPE 1194
11.10.2.2.12 SF_FootSwitch 1202
11.10.2.2.13 SF_Guard 1209
11.10.2.2.14 SF_GuardLocking 1217
11.10.2.2.15 SF_ModeSelector 1227

11.10.2.2.16 SF_MutingPar 1237
11.10.2.2.17 SF_MutingPar_2Sensor 1253
11.10.2.2.18 SF_MutingSeq 1263
11.10.2.2.19 SF_OutControl 1275
11.10.2.2.20 SF_Override 1282
11.10.2.2.21 SF_PressControl 1292
11.10.2.2.22 SF_PSE 1308
11.10.2.2.23 SF_ResetButton 1316
11.10.2.2.24 SF_SafetyRequest 1321
11.10.2.2.25 SF_ManualAck 1328
11.10.2.2.26 SF_TempAlarm 1329
11.10.2.2.27 SF_DataLog 1329
11.10.2.2.28 SF_TestableSafetySensor 1330
11.10.2.2.29 SF_TwoHandControlTypeII 1344
11.10.2.2.30 SF_TwoHandControlTypeIII 1351
11.10.2.2.31 SF_SafetyConnStatus 1358
11.10.2.2.32 SF_SafetyTaskInfo 1358
11.10.2.2.33 SF_GetSafetySignature 1359
11.10.2.2.34 SF_GetSystemMode 1360
11.10.2.2.35 SF_GetSafetyConfirmInfo 1361
11.10.2.2.36 SF_GetCouplerState 1361
11.10.2.2.37 SF_GetGlBusState 1363
11.10.2.2.38 SF_GetEtherNetIpStateCode 1363
11.10.2.2.39 SF_GetFSoeStateCode 1365
11.10.2.2.40 SF_GetCIPSafetyStateCode 1367
11.10.2.3 Extended Function Blocks 1368
11.10.2.3.1 Instruction List 1368
11.10.2.3.2 BOOL_AS_BYTE 1368
11.10.2.3.3 BOOL_AS_WORD 1369
11.10.2.3.4 BOOL_AS_DWORD 1370
11.10.2.3.5 SAFEBOOL_AS_SAFEBYTE 1372
11.10.2.3.6 SAFEBOOL_AS_SAFEWORD 1372
11.10.2.3.7 SAFEBOOL_AS_SAFEDWORD 1373
11.10.2.3.8 BYTE_AS_BOOL 1374
11.10.2.3.9 WORD_AS_BOOL 1375
11.10.2.3.10 DWORD_AS_BOOL 1376

11.10.2.3.11 SAFEBYTE_AS_SAFEBOOL 1378
11.10.2.3.12 SAFEWORD_AS_SAFEBOOL 1378
11.10.2.3.13 SAFEDWORD_AS_SAFEBOOL 1379
12 Library Management 1381
12.1 Library Editing 1381
12.1.1 Overview 1381
12.1.2 Library Update 1381
12.1.3 Compiled Library 1381
12.1.4 Source Code Library 1383
12.2 Library Manager 1386
12.3 Library Reference 1389
12.4 Library Properties 1394
12.5 Library Permissions 1394
13 Visual Design 1395
13.1 Establishing a Connection 1395
13.1.1 Connecting Devices 1395
13.1.1.1 Establishing a Device Connection 1395
13.1.1.2 Setting Device Connection Property 1398
13.1.1.3 Deleting a Device Connection 1400
13.1.2 Connecting External Devices 1400
13.1.2.1 Establishing an External Device Connection 1400
13.1.2.2 Deleting an External Device Connection 1402
13.1.3 Adding a System Variable 1402
13.1.4 Modifying a Connection 1403
13.1.5 Copying a Connection 1403
13.1.6 Supported Device Drivers 1403
13.1.6.1 Communicating with Inovance PLCs 1405
13.1.6.1.1 Easy Series PLCs 1405
13.1.6.1.2 H5U Series PLCs 1405
13.1.6.1.3 EVO Series PLCs 1406
13.1.6.1.4 AM300 and AM500 Series PLCs 1407
13.1.6.1.5 AM400 and AM600 Series PLCs 1411
13.1.6.1.6 AC700 and AC800 Series PLCs 1412
13.1.6.1.7 EtherNet/IP Tags Protocol 1414
13.1.6.1.8 AC, AI, AM, and AP Series Products 1415
13.1.6.2 Communicating with Siemens S7 Series PLCs 1416
13.1.6.2.1 S7-300 PLCs 1416
13.1.6.2.2 S7-1200 and S7-15001500 PLCs 1418

13.1.6.3 Communicating with Rockwell (AB) Devices 1419
13.1.6.3.1 Configuring Connection Properties 1419
13.1.6.3.2 Importing Tag Variables 1420
13.1.6.4 Communicating with Omron Devices 1423
13.1.6.5 OPC UA Client 1424
13.1.6.6 Modbus IDA 1425
13.1.6.6.1 Modbus TCP Protocol 1425
13.1.6.6.2 Modbus RTU Protocol 1428
13.2 Using Variables 1430
13.2.1 System Variable Table 1430
13.2.2 Default Variable Table 1433
13.2.3 Tag Variables 1433
13.2.4 Using a Variable Table 1436
13.2.4.1 Variable Table Groups 1437
13.2.4.2 Adding a Variable Table 1437
13.2.4.3 Importing and Exporting a Variable Table 1438
13.2.4.4 Copying and Pasting a Variable Table 1441
13.2.4.5 Deleting a Variable Table 1441
13.2.5 Configuring Variables 1441
13.2.5.1 Adding an Internal Variable 1441
13.2.5.2 Adding an External Variable 1443
13.2.5.2.1 Adding an External Variable 1443
13.2.5.2.2 Naming an External Variable 1445
13.2.5.2.3 Synchronizing Variable Names 1446
13.2.5.3 Setting Variable Properties 1447
13.2.5.4 Setting Properties in Batch 1452
13.2.5.5 Multiplex Variables 1452
13.2.5.6 Storing Variable Data 1454
13.2.5.6.1 Setting Recording Rules for Data 1454
13.2.5.6.2 Variable-Associated Recording Rules 1456
13.2.5.7 Configuring Variable Events 1458
13.2.6 Operating Variables 1459
13.2.6.1 Copying and Pasting Variables 1459
13.2.6.2 Importing and Exporting Variables 1459
13.2.6.3 Sorting Variables 1461
13.3 Creating a Screen 1461
13.3.1 Screen Types 1462
13.3.2 Adding a Screen 1462

13.3.3 Screen Properties 1464
13.3.4 Renaming a Screen 1465
13.3.5 Object Selection 1466
13.3.5.1 Basic Elements 1466
13.3.5.1.1 Line 1466
13.3.5.1.2 Polyline 1468
13.3.5.1.3 Arc 1470
13.3.5.1.4 Rectangle 1471
13.3.5.1.5 Circle 1474
13.3.5.1.6 Ellipse 1476
13.3.5.1.7 Sector 1476
13.3.5.1.8 Circular Segment 1478
13.3.5.1.9 Polygon 1481
13.3.5.1.10 Text 1482
13.3.5.1.11 Image 1483
13.3.5.1.12 Pipe 1484
13.3.5.2 Control and Display 1486
13.3.5.2.1 Numeric I/O Field 1486
13.3.5.2.2 Character I/O Field 1494
13.3.5.2.3 Date-Time Field 1495
13.3.5.2.4 Button 1497
13.3.5.2.5 Bit Button 1499
13.3.5.2.6 Text Button 1503
13.3.5.2.7 Bit Indicator 1506
13.3.5.2.8 Text Indicator 1507
13.3.5.2.9 Text List 1507
13.3.5.2.10 Combobox 1509
13.3.5.2.11 List Box 1511
13.3.5.2.12 Text Switch 1513
13.3.5.2.13 Radio Button 1514
13.3.5.2.14 Checkbox 1518
13.3.5.2.15 Table 1519
13.3.5.2.16 Picture-in-Picture View 1522
13.3.5.3 Data Visualization 1523
13.3.5.3.1 Bar Graph 1524
13.3.5.3.2 Slider 1526
13.3.5.3.3 Progress Bar 1528

13.3.5.3.4 Circular Progress Bar 1530
13.3.5.3.5 Pie Chart 1532
13.3.5.3.6 F(t) Trend Curve 1533
13.3.5.3.7 F(x) Trend Curve 1537
13.3.5.3.8 Scale 1538
13.3.5.4 Production and Management 1541
13.3.5.4.1 Alarm View 1542
13.3.5.4.2 Alarm Bar 1544
13.3.5.4.3 Recipe View 1545
13.3.5.4.4 User Login 1548
13.3.5.4.5 User View 1549
13.3.5.4.6 Variable Log View 1551
13.3.5.4.7 System Diagnostics 1555
13.3.5.4.8 Operation Records 1556
13.3.5.4.9 SFC View 1558
13.3.5.4.10 Code View 1560
13.3.5.5 Others 1562
13.3.5.5.1 QR Code 1562
13.3.5.5.2 Barcode 1564
13.3.5.5.3 PDF 1565
13.3.5.5.4 Clock 1567
13.3.5.5.5 Browser 1568
13.3.5.5.6 Media Player 1570
13.3.6 Gallery Usage 1572
13.3.6.1 Overview 1573
13.3.6.2 System Gallery Usage 1573
13.3.6.3 User Gallery Usage 1576
13.3.6.4 Project Gallery Usage 1579
13.3.6.5 Local Gallery Usage 1580
13.3.7 Screen Configuration 1581
13.3.7.1 Creating an Object 1582
13.3.7.2 Renaming an Object 1583
13.3.7.3 Copying/Pasting an Object 1584
13.3.7.4 Cutting/Pasting an Object 1585
13.3.7.5 Deleting an Object 1585
13.3.7.6 Rotating an Object 1585
13.3.7.7 Flipping an Object 1586

13.3.7.8 Aligning Objects 1586
13.3.7.9 Grouping Objects 1587
13.3.7.10 Ungrouping Objects 1589
13.3.7.11 Locking an Object 1590
13.3.7.12 Unlocking an Object 1591
13.3.7.13 Adding an Animation 1591
13.3.7.14 Adding an Event 1592
13.3.7.15 Configuring a Picture-in-Picture Screen 1595
13.3.7.16 Resizing an Object 1595
13.3.7.17 Moving an Object 1596
13.3.7.18 Adjusting Object Spacing 1597
13.3.7.19 Configuring a Pop-up Screen 1598
13.3.7.20 Setting Properties in Batch 1600
13.3.8 Animation Configuration 1601
13.3.8.1 Basic Animation 1601
13.3.8.2 Movement Animation 1602
13.3.8.2.1 Movement Animation 1602
13.3.8.2.2 Pipe Flow 1604
13.3.8.3 Rotation Animation 1605
13.3.8.4 Scaling Animation 1607
13.3.8.5 Flashing Animation 1608
13.3.8.6 Appearance Animation 1610
13.3.9 Layer Usage 1612
13.3.9.1 Overview 1612
13.3.9.2 Renaming a Layer 1613
13.3.9.3 Setting an Active Layer 1615
13.3.9.4 Displaying or Hiding a Layer 1618
13.3.9.5 Common Operations for Layer Tree 1620
13.3.10 Screen Management 1621
13.3.10.1 Grouping Screens 1621
13.3.10.2 Zooming a Screen 1622
13.3.10.3 Copying a Screen 1622
13.3.10.4 Changing Screen Resolution 1623
13.4 Template Library Usage 1624
13.4.1 Introduction 1625
13.4.2 Folder Usage 1625
13.4.2.1 Adding a Folder 1625
13.4.2.2 Deleting a Folder 1625

13.4.2.3 Renaming a Folder 1626
13.4.3 Symbol Template Usage 1626
13.4.3.1 Overview of Symbol Template 1626
13.4.3.2 Adding a Symbol Template 1626
13.4.3.3 Modifying a Symbol Template 1627
13.4.3.3.1 Modifying a Symbol Template 1628
13.4.3.3.2 Adding a Variable Interface 1630
13.4.3.3.3 Adding a Property Interface 1632
13.4.3.3.4 Adding an Event Interface 1633
13.4.3.3.5 Adding a Local Variable 1635
13.4.3.3.6 Adding an Animation to Object 1636
13.4.3.3.7 Adding an Event to Object 1636
13.4.3.4 Applying a Symbol Template 1636
13.4.4 User-Defined Types 1639
13.4.4.1 Adding a User-Defined Type 1639
13.4.4.2 Modifying a User-Defined Type 1641
13.4.4.3 Impact of User-Defined Type Version Upgrade 1642
13.4.4.4 Applying a User-defined Type 1644
13.4.5 Template Status Management 1645
13.4.6 Common Operations on Template 1646
13.4.6.1 Opening a Template 1646
13.4.6.2 Modifying a Template 1647
13.4.6.3 Publishing a Template 1648
13.4.6.4 Updating a Template 1649
13.4.6.5 Deleting a Template 1651
13.4.6.6 Renaming a Template 1652
13.4.6.7 Copying a Template 1652
13.4.6.8 Exporting a Template 1653
13.4.6.9 Importing a Template 1654
13.4.6.10 Setting as Default 1654
13.4.6.11 Synchronizing PLC Types 1655
13.4.7 Copying and Pasting a Template Across Projects 1656
13.4.8 Copying and Pasting a Template Instance Across Projects 1658
13.5 Alarm Usage 1659
13.5.1 Discrete Alarm Configuration 1659
13.5.2 Analog Alarm Configuration 1660
13.5.3 System Event Configuration 1662

13.5.4 Alarm Category Configuration 1670
13.5.5 Alarm Group Configuration 1672
13.5.6 Historical Alarm Records 1673
13.5.7 Alarm Import/Export 1675
13.5.8 Alarm Runtime Configuration 1677
13.5.9 Alarm Application Example 1677
13.6 Recipe Usage 1680
13.6.1 Function Overview 1680
13.6.2 Adding a Recipe 1681
13.6.3 Modifying a Recipe 1684
13.6.4 Deleting a Recipe 1685
13.6.5 Copying and Pasting a Recipe 1685
13.6.6 Importing and Exporting a Recipe 1685
13.7 Scheduled Task 1687
13.7.1 Adding a Task 1687
13.7.2 Editing a Task 1688
13.7.3 Task Properties 1689
13.7.4 Event Configuration for Tasks 1690
13.7.5 Copying and Pasting a Task 1691
13.7.6 Deleting a Task 1691
13.8 Operation Record Usage 1691
13.8.1 Operation Record Configuration Process 1692
13.8.2 Operation Records for Stored Projects 1693
13.8.3 Operation Record Runtime Configuration 1694
13.9 Text List Usage 1694
13.9.1 Adding a Text List 1694
13.9.2 Modifying a Text List 1695
13.9.3 Deleting a Text List 1697
13.9.4 Copying/Pasting a Text List 1697
13.9.5 Exporting a Text List 1698
13.9.6 Importing a Text List 1698
13.10 System Function and Script Usage 1699
13.10.1 System Functions 1699
13.10.2 User Script 1700
13.10.2.1 Adding a Script 1700
13.10.2.2 Grouping Scripts 1700
13.10.2.3 Modifying a Script 1701
13.10.2.4 Script Properties 1704

13.10.2.5 Script Toolbox 1706
13.10.2.6 JavaScript Syntax Basics 1707
13.10.2.6.1 Language Overview 1707
13.10.2.6.2 Common Operators 1707
13.10.2.6.3 Operator Precedence 1713
13.11 Installment Payment 1715
13.11.1 Function Overview 1715
13.11.2 Enabling Installment Payment 1716
13.11.3 Selecting an Installment Strategy 1717
13.11.4 Installment Unlocking Modes 1720
13.11.5 Configuring Pre-Due Reminder 1721
13.11.6 Configuring Lock Screen 1723
13.11.7 Impact of Installment Payment on Other Functions 1727
13.12 Monitoring Runtime Configuration 1727
13.12.1 Screen Display Configuration 1728
13.12.2 Preload Screen Configuration 1730
13.12.3 OPC UA Server Configuration 1730
13.12.4 VNC Server Configuration 1731
13.12.5 VNC Client Configuration 1733
13.13 Compilation Configuration 1733
13.13.1 Compiling/Clearing 1733
13.13.2 Saving a Compiled File As 1734
13.14 Project Download/Upload 1737
13.14.1 Monitoring Device Connection 1737
13.14.2 Project Download 1738
13.14.3 Project Upload 1742
13.15 Screen Operation 1743
13.15.1 Real-time Variable Value Monitoring 1743
13.15.2 Online Recipe View Operations 1744
13.15.3 Online Historical Data Viewing 1747
13.15.4 Online Alarm View Operations 1752
13.15.5 Online User View Operations 1754
13.15.6 Online Operation Record Operations 1757
13.15.7 Online System Diagnostics 1758
13.15.8 Online PLC Program Viewing 1761
13.16 Monitoring Simulation Commissioning 1765
13.16.1 Simulation Data Storage 1765
13.16.2 Simulating an HMI 1765

13.16.3 Simulating a Variable 1766
13.16.4 Co-simulation Between PLC and HMI 1768
13.17 Multilingual Settings 1771
13.17.1 Multilingual Overview 1771
13.17.2 Multilingual Configuration and Modification 1773
13.17.3 Multilingual Translation Scope 1774
13.18 HMI Program Reuse Across Projects 1778
13.19 Migrating InoTouchPad Project to iFA Evolution 1779
13.19.1 InoTouchPad Project Migration 1779
13.19.2 Communication Protocol Conversion During Project Migration 1783
14 Maintenance Upgrade 1784
14.1 PLC Firmware and Project Upgrade 1784
14.1.1 Firmware Upgrade 1784
14.1.1.1 Upgrading Firmware via SD Card 1784
14.1.1.2 Upgrading Firmware via Ethernet 1785
14.1.1.3 Upgrading Firmware via Type C 1787
14.1.1.4 Upgrading Firmware via USB Flash Disk (EVO700) 1789
14.1.1.5 Upgrading Redundant System Firmware (EVO800) 1790
14.1.1.5.1 Upgrading via SD Card 1790
14.1.1.5.2 Upgrading via Ethernet 1791
14.1.1.6 Upgrading Safety PLC Firmware (EVO1000) 1793
14.1.2 PLC User Program Upgrade 1797
14.1.2.1 Saving a PLC User Program As 1797
14.1.2.2 Upgrading a User Program via SD Card/USB Flash Disk 1798
14.1.2.3 Upgrading a User Program via Type-C/Ethernet 1799
14.1.2.4 Upgrading a User Program via SFTP 1802
14.2 HMI Firmware and Project Upgrade 1803
14.2.1 Upgrading HMI Firmware 1803
14.3 I/O Firmware Upgrade 1807
14.3.1 Upgrading Hi50 Series Safety I/O Module Firmware 1807
14.3.2 Upgrading Hi80 Series I/O Firmware 1811
15 Diagnostics 1817
15.1 Overview 1817
15.2 Quick Guide 1817
15.2.1 Diagnostics via Segment Code Screen 1817
15.2.2 Diagnostics via iFA Software 1817
15.3 Display of System Diagnostics 1819
15.3.1 Viewing Diagnostic Information via Hardware Interface 1819
15.3.1.1 Device Indicators 1819
15.3.1.2 Segment Code Screen 1821

15.3.2 Viewing Diagnostic Information via iFA Software 1821
15.3.2.1 Diagnostic Symbol Description 1821
15.3.2.2 Device View and Network View 1822
15.3.2.3 Online and Diagnostics 1824
15.3.3 Viewing Diagnostic Information via Web Server 1825
15.3.4 Viewing Diagnostic Information via HMI Device 1828
15.4 System Variable Diagnostics 1830
15.5 Instruction Diagnostics 1835
15.6 Axis Control Diagnostics 1837
16 Motion Control 1840
16.1 Axis Configuration 1840
16.1.1 Overview 1840
16.1.2 Adding an Axis 1840
16.1.3 Basic Axis Settings 1840
16.1.4 Batch Servo Assignment and Axis Parameter Settings 1842
16.1.5 Bus-Controlled Axis 1845
16.1.5.1 Basic Axis Settings 1845
16.1.5.2 Unit Conversion 1845
16.1.5.3 General Settings 1847
16.1.5.4 Commissioning 1849
16.1.5.5 Homing Parameter Settings 1851
16.1.5.6 Axis Diagnostics 1855
16.1.5.7 I/O Mapping 1856
16.1.5.8 System Variables 1857
16.1.5.9 Information 1857
16.1.6 Bus Encoder Axis 1857
16.1.6.1 Overview 1857
16.1.6.2 Basic Axis Settings 1857
16.1.6.3 Unit Conversion 1858
16.1.6.4 Counter Settings 1860
16.1.6.5 Axis Diagnostics 1866
16.1.6.6 I/O Mapping 1866
16.1.6.7 System Variables 1866
16.1.6.8 Information 1867
16.1.7 Local Pulse Axis 1867
16.1.7.1 Overview 1867
16.1.7.2 Basic Axis Settings 1867
16.1.7.3 Unit Conversion 1867

16.1.7.4 Mode/Parameter Settings 1869
16.1.7.5 Homing Settings 1870
16.1.7.6 Commissioning 1871
16.1.7.7 Axis Diagnostics 1874
16.1.7.8 I/O Mapping 1874
16.1.7.9 System Variables 1874
16.1.7.10 Information 1874
16.1.8 Local Encoder Axis 1875
16.1.8.1 Overview 1875
16.1.8.2 Basic Axis Settings 1875
16.1.8.3 Unit Conversion 1875
16.1.8.4 Mode/Parameter Settings 1876
16.1.8.5 Axis Diagnostics 1880
16.1.8.6 I/O Mapping 1880
16.1.8.7 System Variables 1880
16.1.8.8 Information 1881
16.2 Axis Group Configuration 1881
16.2.1 Adding an Axis Group 1881
16.2.2 Axis Group Parameter Settings 1881
16.2.2.1 Mechanical Types 1881
16.2.2.2 Structural Parameters 1885
16.2.2.3 Axis Configuration 1889
16.2.2.4 Motion Parameters 1894
16.2.2.5 Coordinate System Parameters 1896
16.2.2.6 HIC Configuration Example 1898
16.3 Cam Configuration 1906
16.3.1 Overview 1906
16.3.2 Introduction to Cam Table 1907
16.3.3 Cam Table Data Structure 1908
16.3.4 Creating a Cam Table 1909
16.3.5 Cam Table Properties 1912
16.3.6 Cam Chart 1912
16.3.6.1 Cam Table Configuration Interface 1912
16.3.6.2 Cam Diagram 1913
16.3.6.3 Cam Table 1918
17 Drive Settings 1925
17.1 EtherCAT Configuration 1925
17.1.1 EOE Connection 1925

17.1.2 Network Settings 1925
17.1.3 Master Creation and Configuration 1927
17.1.4 Adding a Device 1927
17.1.4.1 Adding a Device 1928
17.1.4.2 Scanning for Devices 1930
17.1.5 Disabling/Enabling a Device 1931
17.1.6 Deleting a Device 1932
17.1.7 Device Configuration 1932
17.1.7.1 General Settings 1932
17.1.7.2 Process Data (PDO Settings) 1936
17.1.7.3 Startup Parameter (SDO Settings) 1937
17.1.7.4 Online 1939
17.1.7.5 Online CoE 1940
17.1.7.6 EoE Settings 1941
17.1.7.7 Servo Parameters 1943
17.1.7.8 ESC Register 1943
17.1.7.9 I/O Mapping 1944
17.1.7.10 System Variables 1947
17.1.7.11 Information 1948
17.1.8 Drive Connection 1948
17.1.9 Version Switching 1950
17.2 General Functions 1951
17.2.1 Oscilloscope 1951
17.2.1.1 Overview 1951
17.2.1.2 Functions 1953
17.2.1.3 Configuration 1957
17.2.1.4 Continuous Oscilloscope 1958
17.2.1.5 Trigger Oscilloscope 1959
17.2.1.6 Channel Selection 1961
17.2.1.7 Waveform File Import/Export 1962
17.2.1.8 Analysis Auxiliary Function 1962
17.2.2 Parameter List 1965
17.2.2.1 Parameter Viewing 1966
17.2.2.2 Custom Group 1967
17.2.2.3 Toolbar 1969
17.2.2.4 Parameter Modification 1973
17.2.2.5 Parameter Search 1975

17.2.2.6 Parameter List (MD6X0 Series) 1975
17.2.3 Parameter Comparison 1977
17.2.4 Machine Parameter Management 1979
17.2.4.1 Machine Recipe Management 1979
17.2.4.2 Machine Parameter Management 1979
17.2.4.3 Firmware Batch Update 1980
17.3 Common Servo Function 1980
17.3.1 Overview 1980
17.3.2 Fault Management 1980
17.3.2.1 Servo Fault Management 1980
17.3.2.2 Fault Overview 1981
17.3.2.3 Troubleshooting 1981
17.3.2.4 Fault Dictionary 1981
17.3.2.5 Fault Diagnostics 1982
17.3.2.6 Fault History 1983
17.3.3 Multi-Axis Device 1984
17.3.4 JOG Motion Control 1985
17.3.4.1 Overview 1985
17.3.4.2 Mode Selection 1986
17.3.4.2.1 Position JOG and Parameter Settings 1987
17.3.4.2.2 Speed JOG and Parameter Settings 1991
17.3.5 Axis Copy 1992
17.3.6 Device Information 1996
17.3.7 Drive Parameter Monitoring 1997
17.3.8 Mechanical Characteristic Analysis 1999
17.3.8.1 Mechanical Characteristics 1999
17.3.8.2 High-Performance Tuning 2005
17.3.9 Inertia Auto-Tuning 2010
17.3.10 STune Adjustment 2012
17.3.11 ETune Guided Auto-Tuning 2015
17.3.11.1 Overview 2015
17.3.11.2 Position Settings 2015
17.3.11.3 Parameter Settings 2018
17.3.11.4 Tuning 2019
17.3.11.5 Recipe Storage 2021
17.3.12 Homing 2022
17.3.13 Digital I/O Settings 2024
17.3.13.1 I/O Settings and Monitoring 2024

17.3.13.2 Virtual I/O Settings and Monitoring 2025
17.3.13.3 Analog Input Settings 2027
17.3.13.4 Analog Output Settings 2028
17.3.14 Dynamic Braking Calculation 2029
17.3.15 Safety Parameters 2032
17.3.16 Comparison Output 2041
17.3.17 Black Box 2046
17.3.17.1 Overview 2046
17.3.17.2 Configuration 2046
17.3.17.3 Waveform Display 2048
17.3.18 Current Loop Adjustment 2049
17.3.18.1 Function Interface 2049
17.3.18.2 Current Loop Tuning 2050
17.3.18.2.1 Current Loop Auto-Tuning 2050
17.3.18.2.2 Static Step Test of Current Loop 2050
17.3.18.2.3 Step Test of Current Loop 2051
17.3.18.3 Current Loop Parameters 2051
17.3.18.4 Current Loop Waveform Display 2051
17.3.19 Fully Closed-Loop Functions 2052
17.3.19.1 External Encoder Feedback Type Settings 2053
17.3.19.2 External Encoder Resolution Settings 2054
17.3.19.3 External Encoder Direction Settings 2054
17.3.19.4 Electronic Gear Ratio Settings 2056
17.3.19.5 Alarm Detection Settings 2056
17.3.20 Bus Motor Parameters 2057
17.3.21 Process Segment 2059
17.3.21.1 Overview 2059
17.3.21.2 General Parameter Configuration 2059
17.3.21.3 Process Segment Parameter Configuration 2059
17.3.21.4 Homing Configuration 2060
17.3.21.5 Process Segment Configuration 2060
17.3.21.6 Event Trigger Configuration 2061
17.3.21.7 Commissioning 2061
17.3.22 Overload Protection 2061
17.3.23 Motor Management Library 2063
17.3.24 Accuracy Compensation 2071
17.3.25 Cogging Compensation 2074

17.3.26 Gantry Synchronization 2076
17.3.27 Motion Unit Configuration 2079
17.3.28 Motion Trajectory and Simulation 2081
17.3.28.1 Motion Trajectory 2081
17.3.28.2 Simulation 2091
17.4 Common AC Drive Functions 2094
17.4.1 Fault Management 2094
17.4.1.1 AC Drive Fault Management 2094
17.4.1.2 Fault Overview 2095
17.4.1.3 Troubleshooting 2095
17.4.1.4 Fault Diagnostics 2096
17.4.2 Control Panel 2096
17.4.3 Function Diagram 2100
17.4.4 Motor Auto-Tuning 2103
17.4.5 Black Box 2104
17.4.5.1 Overview 2104
17.4.5.2 Configuration 2105
17.4.5.3 Read 2109
17.4.5.4 Waveform Display 2110
17.4.6 I/O monitoring 2110
17.5 Servo Parameter Description 2114
17.6 AC Drive Parameter Description 2115
18 Common Instructions 2116
18.1 Basic Instructions 2117
18.1.1 Bit Logic 2117
18.1.1.1 Instruction List 2117
18.1.1.2 R_TRIG 2118
18.1.1.3 F_TRIG 2119
18.1.1.4 PLS 2121
18.1.1.5 PLF 2123
18.1.1.6 SR 2125
18.1.1.7 RS 2127
18.1.1.8 ALT 2130
18.1.1.9 BOUT 2131
18.1.1.10 BSET 2133
18.1.1.11 BRST 2135
18.1.1.12 D_TRIG 2137

18.1.2 Data Processing 2140
18.1.2.1 Instruction List 2140
18.1.2.2 BON 2143
18.1.2.3 SUM 2145
18.1.2.4 SWAP 2146
18.1.2.5 SWAP16 2148
18.1.2.6 SWAP32 2150
18.1.2.7 XCH 2151
18.1.2.8 BitCnt 2153
18.1.2.9 Clear 2155
18.1.2.10 Byte_To_HexString 2156
18.1.2.11 Word_To_HexString 2158
18.1.2.12 Dword_To_HexString 2159
18.1.2.13 ColmToLine_BYTE 2160
18.1.2.14 ColmToLine_WORD 2162
18.1.2.15 ColmToLine_DWORD 2165
18.1.2.16 ColmToLine_LWORD 2167
18.1.2.17 CopyRealToNum 2170
18.1.2.18 CopyLrealToNum 2172
18.1.2.19 CopyByteToNum 2173
18.1.2.20 CopyWordToNum 2174
18.1.2.21 CopyDwordToNum 2176
18.1.2.22 CopyLwordToNum 2177
18.1.2.23 CopyDwordToReal 2179
18.1.2.24 CopyLwordToLreal 2180
18.1.2.25 CopyNumToReal 2182
18.1.2.26 CopyNumToLreal 2183
18.1.2.27 CopyNumToByte 2185
18.1.2.28 CopyNumToWord 2186
18.1.2.29 CopyNumToDword 2188
18.1.2.30 CopyNumToLword 2189
18.1.2.31 CopyRealToDword 2191
18.1.2.32 CopyLrealToLword 2192
18.1.2.33 Dispart8Bit 2194
18.1.2.34 DispartDigit 2196
18.1.2.35 DispartReal 2198

18.1.2.36 Exchange 2199
18.1.2.37 FixNumToString 2201
18.1.2.38 GRAY 2203
18.1.2.39 GrayToBin_BYTE 2207
18.1.2.40 GrayToBin_WORD 2209
18.1.2.41 GrayToBin_DWORD 2210
18.1.2.42 GrayToBin_LWORD 2212
18.1.2.43 LineToColm 2213
18.1.2.44 MoveDigit 2215
18.1.2.45 ReadNbit_BYTE 2218
18.1.2.46 ReadNbit_WORD 2220
18.1.2.47 ReadNbit_DWORD 2222
18.1.2.48 ReadNbit_LWORD 2224
18.1.2.49 SetBlock 2226
18.1.2.50 StringToFixNum 2228
18.1.2.51 TransBits 2230
18.1.2.52 Unite8Bit_BYTE 2233
18.1.2.53 Unite8Bit_WORD 2235
18.1.2.54 Unite8Bit_DWORD 2237
18.1.2.55 Unite8Bit_LWORD 2239
18.1.2.56 UniteDigit_BYTE 2241
18.1.2.57 UniteDigit_WORD 2243
18.1.2.58 UniteDigit_DWORD 2245
18.1.2.59 UniteDigit_LWORD 2247
18.1.2.60 UniteReal 2249
18.1.2.61 WriteNbit_BYTE 2251
18.1.2.62 WriteNbit_WORD 2253
18.1.2.63 WriteNbit_DWORD 2255
18.1.2.64 WriteNbit_LWORD 2257
18.1.2.65 StringSum 2259
18.1.3 Comparison Operations 2261
18.1.3.1 Instruction List 2261
18.1.3.2 EQ 2262
18.1.3.3 GE 2263
18.1.3.4 GT 2264
18.1.3.5 LE 2266

18.1.3.6 LT 2267
18.1.3.7 NE 2269
18.1.4 Logic Operations 2270
18.1.4.1 Instruction List 2270
18.1.4.2 AND 2271
18.1.4.3 NOT 2272
18.1.4.4 OR 2274
18.1.4.5 XOR 2275
18.1.4.6 TestABit 2277
18.1.4.7 TestABitN 2278
18.1.4.8 SetBits 2280
18.1.4.9 ResetBits 2282
18.1.4.10 RESET_BF 2283
18.1.4.11 SET_BF 2285
18.1.5 Mathematical Operations 2287
18.1.5.1 Instruction List 2287
18.1.5.2 CeilReal 2288
18.1.5.3 CeilLReal 2290
18.1.5.4 FloorReal 2292
18.1.5.5 FloorLReal 2293
18.1.5.6 TruncateF 2295
18.1.5.7 TruncateL 2296
18.1.5.8 fmodLR 2298
18.1.5.9 Rand 2299
18.1.5.10 CheckReal 2301
18.1.5.11 Inc 2303
18.1.5.12 Dec 2304
18.1.5.13 ModReal 2305
18.1.5.14 ModReal_LR 2308
18.1.5.15 ABS 2310
18.1.5.16 ACOS 2311
18.1.5.17 ADD 2313
18.1.5.18 ASIN 2314
18.1.5.19 ATAN 2315
18.1.5.20 COS 2316
18.1.5.21 DIV 2318
18.1.5.22 EXP 2319

18.1.5.23 EXPT 2320
18.1.5.24 LN 2322
18.1.5.25 LOG 2323
18.1.5.26 MOD 2324
18.1.5.27 MOVE 2326
18.1.5.28 MUL 2327
18.1.5.29 SIN 2329
18.1.5.30 SIZEOF 2330
18.1.5.31 SQRT 2332
18.1.5.32 SUB 2333
18.1.5.33 TAN 2334
18.1.5.34 RAD 2336
18.1.5.35 DEG 2337
18.1.5.36 SCALE_X 2338
18.1.5.37 NORM_X 2340
18.1.6 Selection Operations 2343
18.1.6.1 Instruction List 2343
18.1.6.2 LIMIT 2344
18.1.6.3 MAX 2345
18.1.6.4 MIN 2347
18.1.6.5 MUX 2348
18.1.6.6 SEL 2351
18.1.7 Shift Operations 2353
18.1.7.1 Instruction List 2353
18.1.7.2 RCL 2353
18.1.7.3 RCR 2355
18.1.7.4 ROL 2357
18.1.7.5 ROR 2359
18.1.7.6 SHL 2360
18.1.7.7 SHR 2362
18.1.8 Counter 2364
18.1.8.1 Instruction List 2364
18.1.8.2 CTU 2365
18.1.8.3 CTD 2367
18.1.8.4 CTUD 2369
18.1.8.5 CTU_DINT 2374
18.1.8.6 CTU_UDINT 2377

18.1.8.7 CTU_LINT 2379
18.1.8.8 CTU_ULINT 2382
18.1.8.9 CTD_DINT 2384
18.1.8.10 CTD_UDINT 2387
18.1.8.11 CTD_LINT 2389
18.1.8.12 CTD_ULINT 2392
18.1.8.13 CTUD_DINT 2394
18.1.8.14 CTUD_UDINT 2399
18.1.8.15 CTUD_LINT 2405
18.1.8.16 CTUD_ULINT 2411
18.1.9 Timer 2417
18.1.9.1 Instruction List 2417
18.1.9.2 RTC 2418
18.1.9.3 TOF 2421
18.1.9.4 TON 2424
18.1.9.5 TON2 2427
18.1.9.6 TP 2433
18.1.9.7 LTOF 2436
18.1.9.8 LTON 2438
18.1.9.9 LTP 2439
18.1.9.10 MTP 2441
18.1.9.11 TimeSwitch 2445
18.1.9.12 AccumulationTimer 2456
18.1.9.13 AccumulationTimer2 2460
18.1.9.14 Timer 2467
18.1.9.15 Timer100ms 2469
18.1.10 String 2472
18.1.10.1 Instruction List 2472
18.1.10.2 CONCAT 2474
18.1.10.3 DELETE 2475
18.1.10.4 FIND 2477
18.1.10.5 INSERT 2479
18.1.10.6 LEFT 2481
18.1.10.7 LEN 2482
18.1.10.8 MID 2484
18.1.10.9 REPLACE 2486
18.1.10.10 RIGHT 2487

18.1.10.11 WConcat ... 2489
18.1.10.12 WDelete ... 2490
18.1.10.13 WFind ... 2491
18.1.10.14 WInsert ... 2493
18.1.10.15 WLeft ... 2494
18.1.10.16 WLen ... 2495
18.1.10.17 WMid ... 2496
18.1.10.18 WReplace ... 2498
18.1.10.19 WRight ... 2499
18.1.10.20 AddDelimiter ... 2500
18.1.10.21 AddDelimiter_LR ... 2503
18.1.10.22 ClearString ... 2506
18.1.10.23 GetByteLen ... 2507
18.1.10.24 SubDelimiter ... 2509
18.1.10.25 SubDelimiter_LR ... 2513
18.1.10.26 TodToString ... 2517
18.1.10.27 ToLCase ... 2519
18.1.10.28 ToUCase ... 2520
18.1.10.29 TrimL ... 2522
18.1.10.30 TrimR ... 2523
18.1.11 Pointer ... 2525
18.1.11.1 Instruction List ... 2525
18.1.11.2 ADR ... 2525
18.1.12 Conversion Operators ... 2527
18.1.12.1 Instruction List ... 2527
18.1.12.2 BOOL_TO_<TYPE> ... 2528
18.1.12.3 BYTE_TO_<TYPE> ... 2530
18.1.12.4 DATE_TO_<TYPE> ... 2531
18.1.12.5 DINT_TO_<TYPE> ... 2533
18.1.12.6 DT_TO_<TYPE> ... 2534
18.1.12.7 DWORD_TO_<TYPE> ... 2536
18.1.12.8 INT_TO_<TYPE> ... 2537
18.1.12.9 LINT_TO_<TYPE> ... 2539
18.1.12.10 LREAL_TO_<TYPE> ... 2540
18.1.12.11 LTIME_TO_<TYPE> ... 2542
18.1.12.12 LWORD_TO_<TYPE> ... 2543
18.1.12.13 REAL_TO_<TYPE> ... 2544

18.1.12.14 SINT_TO_<TYPE> 2546
18.1.12.15 STRING_TO_<TYPE> 2547
18.1.12.16 TIME_TO_<TYPE> 2549
18.1.12.17 TOD_TO_<TYPE> 2551
18.1.12.18 UDINT_TO_<TYPE> 2552
18.1.12.19 UINT_TO_<TYPE> 2553
18.1.12.20 ULINT_TO_<TYPE> 2555
18.1.12.21 USINT_TO_<TYPE> 2556
18.1.12.22 WORD_TO_<TYPE> 2558
18.1.12.23 WSTRING_TO_<TYPE> 2559
18.1.12.24 UNICODE_TO_UTF8 2561
18.2 Extended Instructions 2563
18.2.1 System Time 2563
18.2.1.1 Instruction List 2563
18.2.1.2 GetLocalTime 2566
18.2.1.3 GetSysTimeMs 2567
18.2.1.4 GetSysTimeUs 2568
18.2.1.5 GetSysTimeNs 2569
18.2.1.6 GetSysTimeRTC 2571
18.2.1.7 SetSysTimeRTC 2572
18.2.1.8 GetSystemTime 2575
18.2.1.9 ADD_DT_TIME 2576
18.2.1.10 ADD_TOD_TIME 2578
18.2.1.11 ChkLeapYear 2580
18.2.1.12 DateStructToDt 2581
18.2.1.13 DateToSec 2583
18.2.1.14 DaysToMonth 2584
18.2.1.15 DIVTIME 2586
18.2.1.16 DtToDateStruct 2588
18.2.1.17 DtToSec 2590
18.2.1.18 GetDayOfWeek 2591
18.2.1.19 GetDaysOfMonth 2593
18.2.1.20 GetSystemDate_sDt 2595
18.2.1.21 GetWeekOfYear 2596
18.2.1.22 MULTIME 2598
18.2.1.23 NanoSecToTime 2600
18.2.1.24 SecToTime 2601

18.2.1.25 SecToDate 2603
18.2.1.26 SecToTod 2604
18.2.1.27 SecToDt 2606
18.2.1.28 SUB_DATE_DATE 2607
18.2.1.29 SUB_DT_DT 2609
18.2.1.30 SUB_DT_TIME 2611
18.2.1.31 SUB_TOD_TIME 2613
18.2.1.32 SUB_TOD_TOD 2614
18.2.1.33 TimeToNanoSec 2616
18.2.1.34 TimeToSec 2617
18.2.1.35 TodToSec 2618
18.2.1.36 TruncDt 2620
18.2.1.37 TruncTime 2622
18.2.1.38 TruncTod 2624
18.2.2 Data Calculation 2625
18.2.2.1 Instruction List 2625
18.2.2.2 HYSTERESIS 2626
18.2.2.3 LIMITALARM 2628
18.2.2.4 DERIVATIVE 2631
18.2.2.5 INTEGRAL 2633
18.2.2.6 LIN_TRAFO 2635
18.2.2.7 STATISTICS_INT 2638
18.2.2.8 STATISTICS_REAL 2640
18.2.2.9 VARIANCE 2642
18.2.3 BCD Code Conversion 2644
18.2.3.1 Instruction List 2644
18.2.3.2 BCD_TO_BYTE 2644
18.2.3.3 BCD_TO_DWORD 2645
18.2.3.4 BCD_TO_INT 2647
18.2.3.5 BCD_TO_WORD 2648
18.2.3.6 BYTE_TO_BCD 2650
18.2.3.7 DWORD_TO_BCD 2651
18.2.3.8 INT_TO_BCD 2653
18.2.3.9 WORD_TO_BCD 2654
18.2.4 Byte/Bit Operations 2656
18.2.4.1 Instruction List 2656
18.2.4.2 BIT_AS_BYTE 2657

18.2.4.3 BIT_AS_WORD 2659
18.2.4.4 BIT_AS_DWORD 2662
18.2.4.5 BYTE_AS_BIT 2666
18.2.4.6 WORD_AS_BIT 2669
18.2.4.7 DWORD_AS_BIT 2671
18.2.4.8 EXTRACT 2675
18.2.4.9 PACK 2677
18.2.4.10 PUTBIT 2679
18.2.4.11 SWITCHBIT 2681
18.2.4.12 UNPACK 2682
18.2.4.13 UnpackWord 2684
18.2.4.14 NSHLC 2688
18.2.4.15 NSHRC 2690
18.2.5 String Conversion 2692
18.2.5.1 Instruction List 2692
18.2.5.2 DateToString 2693
18.2.5.3 DtToString 2695
18.2.5.4 Decoder 2696
18.2.5.5 Encoder 2698
18.2.5.6 HexStringToNum_DINT 2700
18.2.5.7 HexStringToNum_INT 2702
18.2.5.8 HexStringToNum_LINT 2704
18.2.5.9 HexStringToNum_SINT 2705
18.2.5.10 HexStringToNum_UDINT 2707
18.2.5.11 HexStringToNum_UINT 2708
18.2.5.12 HexStringToNum_ULINT 2710
18.2.5.13 HexStringToNum_USINT 2711
18.2.5.14 MovingAverage 2713
18.2.5.15 NumToDecString 2717
18.2.5.16 NumToHexString 2719
18.2.5.17 EnumToNum 2722
18.2.5.18 NumToEnum 2723
18.2.5.19 RoundUp 2725
18.2.5.20 AnyBASE64 2727
18.2.5.21 AnyNumFmt 2734
18.2.6 Gray Code Conversion 2736
18.2.6.1 Instruction List 2736

18.2.6.2 BYTE_TO_GRAY 2737
18.2.6.3 WORD_TO_GRAY 2739
18.2.6.4 DWORD_TO_GRAY 2740
18.2.6.5 GRAY_TO_BYTE 2742
18.2.6.6 GRAY_TO_WORD 2743
18.2.6.7 GRAY_TO_DWORD 2745
18.2.6.8 BinToGray_BYTE 2746
18.2.6.9 BinToGray_WORD 2748
18.2.6.10 BinToGray_DWORD 2750
18.2.6.11 BinToGray_LWORD 2751
18.2.7 Signal Processing 2753
18.2.7.1 Instruction List 2753
18.2.7.2 BLINK 2754
18.2.7.3 GEN 2756
18.2.7.4 FREQ_MEASURE 2761
18.2.7.5 GetClk_1ms 2764
18.2.7.6 GetClk_10ms 2765
18.2.7.7 GetClk_20ms 2766
18.2.7.8 GetClk_100ms 2767
18.2.7.9 GetClk_1s 2769
18.2.7.10 GetClk_1min 2770
18.2.8 Filtering Algorithm 2772
18.2.8.1 Instruction List 2772
18.2.8.2 ArithmeticAverageFilter 2772
18.2.8.3 DebounceFilter 2778
18.2.8.4 FirstOrderLagFilter 2784
18.2.8.5 LimitingAverageFilter 2792
18.2.8.6 LimitingDebounceFilter 2799
18.2.8.7 LimitingFilter 2806
18.2.8.8 MedianAverageFilter 2813
18.2.8.9 MedianFilter 2819
18.2.8.10 RecursiveAverageFilter 2826
18.2.8.11 Fault Codes 2833
18.2.9 ASCII Code Conversion 2834
18.2.9.1 Instruction List 2834
18.2.9.2 HEXinASCII_TO_BYTE 2834
18.2.9.3 BYTE_TO_HEXinASCII 2835

18.2.9.4 WORD_AS_STRING 2837
18.2.10 Table Range 2839
18.2.10.1 Instruction List 2839
18.2.10.2 RecMax 2840
18.2.10.3 RecMin 2844
18.2.10.4 RecNum 2847
18.2.10.5 RecRangeSearch 2850
18.2.10.6 RecSearch 2855
18.2.10.7 RecSort 2860
18.2.10.8 StackDel 2865
18.2.10.9 StackFIFO 2868
18.2.10.10 StackIns 2870
18.2.10.11 StackLIFO 2874
18.2.10.12 StackPush 2876
18.2.10.13 TableCmp 2879
18.2.10.14 ZoneCmp 2882
18.2.10.15 RAMP_INT 2884
18.2.10.16 RAMP_REAL 2888
18.2.10.17 CHARCURVE 2893
18.2.10.18 BZAND_TAB 2896
18.2.10.19 RAMP_TAB 2898
18.2.10.20 ZONE_TAB 2902
18.2.11 File Operations 2904
18.2.11.1 Instruction List 2904
18.2.11.2 Overview 2906
18.2.11.3 FileOpen 2906
18.2.11.4 FileClose 2910
18.2.11.5 FileSeek 2913
18.2.11.6 FileTell 2917
18.2.11.7 FileFindList 2921
18.2.11.8 FileCopy 2927
18.2.11.9 FileRemove 2931
18.2.11.10 FileRename 2933
18.2.11.11 DirCreate 2937
18.2.11.12 DirRemove 2939
18.2.11.13 File Read/Write (String Supported) 2942
18.2.11.13.1 FileGets 2942

18.2.11.13.2 FilePuts 2946
18.2.11.13.3 FileRead 2949
18.2.11.13.4 FileWrite 2953
18.2.11.13.5 FileReadVar 2956
18.2.11.13.6 FileWriteVar 2959
18.2.11.14 File Read/Write (Wstring Supported) 2963
18.2.11.14.1 FileGetsWstr 2963
18.2.11.14.2 FilePutsWstr 2967
18.2.11.14.3 FileReadWstr 2970
18.2.11.14.4 FileWriteWstr 2973
18.2.11.14.5 FileReadVarWstr 2977
18.2.11.14.6 FileWriteVarWstr 2979
18.2.12 Special File Processing 2983
18.2.12.1 XML File Processing 2983
18.2.12.1.1 Instruction List 2983
18.2.12.1.2 XmlProc_FileOpen 2983
18.2.12.1.3 XmlProc_FileClose 2985
18.2.12.1.4 XmlProc_FileDelete 2987
18.2.12.1.5 XmlProc_GetNameSpaceInfo 2988
18.2.12.1.6 XmlProc_NodeEdit 2990
18.2.12.1.7 XmlProc_AddJmfInfo 2995
18.2.12.1.8 Fault Codes 2997
18.2.13 Analog Signal 2998
18.2.13.1 Instruction List 2998
18.2.13.2 AEXTIME 2999
18.2.13.3 CHDETECT 3002
18.2.13.4 CHDETECT_T 3004
18.2.13.5 FILTER 3007
18.2.13.6 FLOWACC 3013
18.2.13.7 INCRSUM 3019
18.2.13.8 SWND 3022
18.2.13.9 THRSEL 3024
18.2.13.10 TWOSEL 3031
18.2.14 Digital Signal 3036
18.2.14.1 Instruction List 3036
18.2.14.2 DIG4SEL2 3036
18.2.14.3 DIG4SEL3 3038

18.2.14.4 DIGCNUM 3040
18.2.14.5 DIGONUM 3042
18.2.14.6 DIGSTA 3048
18.2.14.7 DIGXOr04 3051
18.2.14.8 FSTDEF 3053
18.2.14.9 Inlock4 3060
18.2.15 Array Comparison 3063
18.2.15.1 Instruction List 3063
18.2.15.2 AryCmpEQ 3063
18.2.15.3 AryCmpNE 3066
18.2.15.4 AryCmpEQV 3069
18.2.15.5 AryCmpNEV 3071
18.2.15.6 AryCmpLT 3073
18.2.15.7 AryCmpLE 3076
18.2.15.8 AryCmpGT 3079
18.2.15.9 AryCmpGE 3081
18.2.15.10 AryCmpLTV 3084
18.2.15.11 AryCmpLEV 3086
18.2.15.12 AryCmpGTV 3088
18.2.15.13 AryCmpGEV 3091
18.2.16 Any Array 3094
18.2.16.1 Instruction List 3094
18.2.16.2 AryByteTo 3095
18.2.16.3 AryExchange 3101
18.2.16.4 AryMove 3103
18.2.16.5 AryToBCD 3105
18.2.16.6 AryToBin 3107
18.2.16.7 AryToString 3110
18.2.16.8 SizeOfAry 3111
18.2.16.9 StringToAry 3113
18.2.16.10 ToAryByte 3115
18.2.16.11 AryAnd 3120
18.2.16.12 AryOr 3122
18.2.16.13 AryXor 3125
18.2.16.14 AryXorN 3127
18.2.16.15 AryAdd 3130
18.2.16.16 ArySub 3132

18.2.16.17 AryMul 3135
18.2.16.18 AryDiv 3137
18.2.16.19 AryAddV 3140
18.2.16.20 ArySubV 3142
18.2.16.21 AryMulV 3144
18.2.16.22 AryDivV 3146
18.2.16.23 AryMean 3149
18.2.16.24 ArySD 3150
18.2.16.25 AryMax 3153
18.2.16.26 AryMin 3156
18.2.16.27 ArySearch 3159
18.2.16.28 ArySHL 3162
18.2.16.29 ArySHR 3165
18.2.16.30 AryShiftReg 3167
18.2.16.31 AryShiftRegLR 3170
18.2.17 Memory Copy 3173
18.2.17.1 Instruction List 3173
18.2.17.2 AnyMemSet 3174
18.2.17.3 AnyMemCpy 3176
18.2.17.4 AnyMemMove 3178
18.2.17.5 AnyMemCmp 3180
18.2.18 Recipe Library 3183
18.2.18.1 RecipeManager 3183
18.2.18.1.1 Instruction List 3183
18.2.18.1.2 Overview 3185
18.2.18.1.3 CreateRecipe 3185
18.2.18.1.4 CreateRecipeNoSave 3188
18.2.18.1.5 DeleteAllRecipe 3190
18.2.18.1.6 DeleteRecipe 3195
18.2.18.1.7 DeleteRecipeFile 3197
18.2.18.1.8 LoadAllRecipes 3199
18.2.18.1.9 LoadAndWriteRecipe 3200
18.2.18.1.10 LoadFromAndWriteRecipe 3202
18.2.18.1.11 LoadRecipe 3204
18.2.18.1.12 ReadAndSaveAs 3206
18.2.18.1.13 ReadAndSaveRecipe 3208
18.2.18.1.14 ReadAndSaveRecipeAs 3210

18.2.18.1.15 SaveRecipe 3213
18.2.18.1.16 ReadRecipe 3214
18.2.18.1.17 WriteRecipe 3216
18.2.18.1.18 ReloadRecipes 3218
18.2.18.1.19 GetRecipeCount 3219
18.2.18.1.20 GetRecipeNames 3221
18.2.18.1.21 GetRecipeValues 3224
18.2.18.1.22 GetRecipeVariableNames 3228
18.2.18.1.23 SetRecipeValues 3231
18.2.18.1.24 GetRecipeNamesV2 3235
18.2.18.1.25 GetRecipeValuesV2 3238
18.2.18.1.26 GetRecipeVariableNamesV2 3242
18.2.18.1.27 SetRecipeValuesV2 3245
18.2.18.1.28 GetLastError 3249
18.2.18.1.29 ResetLastError 3251
18.2.18.1.30 SetStoragePath 3252
18.2.18.1.31 CreateRecipeV3 3253
18.2.18.1.32 DeleteALLRecipeV3 3255
18.2.18.1.33 DeleteRecipeV3 3257
18.2.18.1.34 GetRecipeDefinitionInfoV3 3260
18.2.18.1.35 GetRecipeInfoV3 3262
18.2.18.1.36 LoadRecipeV3 3265
18.2.18.1.37 ReadRecipeV3 3268
18.2.18.1.38 RenameRecipeV3 3270
18.2.18.1.39 SaveRecipeV3 3272
18.2.18.1.40 SetRecipeValuesV3 3274
18.2.18.1.41 WriteRecipeV3 3277
18.2.18.1.42 SetStoragePathV3 3278
18.2.18.1.43 Fault Codes 3280
18.2.19 C Function Support 3282
18.2.19.1 Instruction List 3282
18.2.19.2 IOBlockCall 3282
18.2.19.3 IOBlockProcess 3285
18.2.19.4 IOBlockLibCall 3288
18.2.19.5 IOBlockLibProcess 3291
18.2.20 Sequencer 3295
18.2.20.1 Instruction List 3295

18.2.20.2 SQO 3295
18.3 Process Instructions 3301
18.3.1 PID Control 3301
18.3.1.1 Instruction List 3301
18.3.1.2 PD 3302
18.3.1.3 PID 3307
18.3.1.4 PID_FIXCYCLE 3311
18.3.1.5 PID_AT 3316
18.3.1.6 PID_ATC 3352
18.3.1.7 Help Function 3391
18.3.1.7.1 FPTLGSim 3391
18.3.1.7.2 FSMITHSim 3397
18.3.1.7.3 SPTLGSim 3403
18.3.1.7.4 SSMITHSim 3410
18.3.1.7.5 LEAD_LAG 3416
18.3.1.7.6 DELIM 3422
18.3.1.7.7 DZLIM 3426
18.3.1.7.8 RALIM 3429
18.3.1.7.9 MANCON 3433
18.3.1.7.10 ORSEL 3438
18.3.1.7.11 PWMC 3441
18.3.1.7.12 RATIO 3446
18.3.1.7.13 SPLRANGE 3449
18.3.1.7.14 THPOSC 3457
18.3.1.7.15 TWPOSC 3466
18.3.2 Advanced Process Control 3474
18.3.2.1 Instruction List 3474
18.3.2.2 PFCs 3475
18.3.2.3 PFCm 3488
18.3.2.4 PFCs_dv 3502
18.3.2.5 PFCs_v2 3508
18.3.2.6 PFCm_v2 3523
18.3.2.7 PFCs_dv_v2 3538
18.3.2.8 PFCm_dv 3544
18.3.2.9 PIDR 3550
18.3.2.10 PFCm22 3560
18.3.2.11 PRBS 3578

18.3.2.12 PFC11 3580
18.3.2.13 PFC22 3594
18.3.2.14 PFC_DV 3614
18.3.2.15 Fault Codes 3621
18.3.3 Motion Control 3626
18.3.3.1 Overview 3626
18.3.3.1.1 PLCopen Specifications 3626
18.3.3.1.2 Single-Axis Instructions 3630
18.3.3.1.3 Synchronous Axis Instructions 3634
18.3.3.1.4 Axis Group Instructions 3638
18.3.3.1.5 Position Retention upon Power Failure 3643
18.3.3.2 Object Definitions 3650
18.3.3.2.1 Data Object Description 3650
18.3.3.2.2 Single-Axis Data Object Definitions 3651
18.3.3.2.2.1 AXIS_REF: Axis Structure 3651
18.3.3.2.2.2 AXIS Communication State Machines 3654
18.3.3.2.2.3 MC_BUS_SERVO_AXIS_REF: Bus Servo Axis Structure 3655
18.3.3.2.2.4 MC_FBERROR_REF: Fault Record Structure 3656
18.3.3.2.2.5 MC_AXIS_STATE: Axis State Machine Enumeration 3657
18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration 3657
18.3.3.2.2.7 MC_CONTROLLER_MODE: Controller Mode Enumeration 3658
18.3.3.2.2.8 MC_MOVE_MODE: Move Mode Enumeration 3658
18.3.3.2.2.9 MC_DIRECTION: Motion Direction Enumeration 3658
18.3.3.2.2.10 MC_STOP_MODE: Stop Mode Enumeration 3658
18.3.3.2.2.11 MC_REFERENCE_TYPE: Data Type Enumeration 3659
18.3.3.2.2.12 MC_COMBINE_MODE: Combine Mode Enumeration 3659
18.3.3.2.2.13 MC_HOMING_MODE: Homing Mode Enumeration 3659
18.3.3.2.2.14 MC_LIMIT_REVERSE_MODE: Limit Reverse Mode Enumeration 3660
18.3.3.2.2.15 MC_RAMPTYPE: Ramp Type Enumeration 3660
18.3.3.2.2.16 MC_ZERO_SIGNAL: Zero Signal Enumeration 3660
18.3.3.2.2.17 MC_CONTROLLER_MODE_SWITCH: Controller Mode Switch Mode Enumeration 3661
18.3.3.2.3 Synchronous Axis Data Object Definitions 3661
18.3.3.2.3.1 MC_CAM_TABLE_REF: Cam Table Structure 3661
18.3.3.2.3.2 MC_CAM_NODE_TYPE_REF: Cam Node Type Structure 3661
18.3.3.2.3.3 MC_CAM_KEY_NODE_REF: Key Node Structure 3661
18.3.3.2.3.4 MC_CAM_FITTING_NODE_REF: Dense Node Structure 3662
18.3.3.2.3.5 MC_CAM_IN_STATE_REF: Camming State Structure 3662

18.3.3.2.3.6 MC_SWITCH_REF: Switch Structure 3663
18.3.3.2.3.7 MC_OUTMODE: Desynchronization Mode Enumeration 3663
18.3.3.2.3.8 MC_MASTER_DIRECTION: Master Axis Motion Direction Enumeration 3663
18.3.3.2.4 Axis Group Data Object Definitions 3664
18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure 3664
18.3.3.2.4.2 MC_AXIS_SPACE_REF: Logical Axis Configuration Structure 3665
18.3.3.2.4.3 MC_WORK_SPACE_REF: Work Space Configuration Structure 3666
18.3.3.2.4.4 MC_GROUP_CFG_REF: Axis Group State Structure 3667
18.3.3.2.4.5 MC_AXES_GROUP_STATE: Axis Group State Enumeration 3667
18.3.3.2.4.6 MC_CIRCE_TYPE: Circular Path Type Enumeration 3668
18.3.3.2.4.7 MC_COORD_TYPE: Coordinate System Type Enumeration 3668
18.3.3.2.4.8 MC_CURVE_TYPE: Curve Type Enumeration 3668
18.3.3.2.4.9 MC_ERROR_STOP: Stop Mode Enumeration 3669
18.3.3.2.4.10 MC_TRANSITION_MODE: Transition Mode Enumeration 3669
18.3.3.2.4.11 MC_UNIT_SPACE: Main Unit Enumeration 3669
18.3.3.2.4.12 MC_GROUP_STOP_MODE: Stop Mode Enumeration 3669
18.3.3.2.4.13 MC_CORNER_TYPE: Corner Type Enumeration 3670
18.3.3.2.4.14 MC_HOME_MODE: Home Mode Enumeration 3670
18.3.3.2.4.15 MC_LOAD_PATH_STATUS: Path Status Enumeration 3670
18.3.3.2.4.16 MC_PATH_CHOICE: Path Choice Enumeration 3670
18.3.3.2.4.17 MC_CIRC_PATHCHOICE: Circular Path Choice Enumeration 3670
18.3.3.2.4.18 MC_CIRCMODE: Circular Interpolation Mode Enumeration 3671
18.3.3.2.4.19 MC_COORDSYSTEM: Coordinate System Enumeration 3671
18.3.3.2.4.20 MC_PART_STATUS: Coordinate System Switch Status Enumeration 3671
18.3.3.3 Single Axis 3672
18.3.3.3.1 Instruction List 3673
18.3.3.3.2 MC_Power 3681
18.3.3.3.3 MC_ReadActualPosition 3685
18.3.3.3.4 MC_ReadAxisError 3688
18.3.3.3.5 MC_ReadStatus 3691
18.3.3.3.6 MC_Reset 3695
18.3.3.3.7 MC_ReadActualTorque 3700
18.3.3.3.8 MC_ReadActualVelocity 3702
18.3.3.3.9 MC_SetPosition 3704
18.3.3.3.10 MC_TouchProbe 3708
18.3.3.3.11 MC_ReadBoolParameter 3718
18.3.3.3.12 MC_ReadParameter 3721

18.3.3.3.13 MC_VibrationShapping 3724
18.3.3.3.14 MC_WriteBoolParameter 3727
18.3.3.3.15 MC_WriteParameter 3729
18.3.3.3.16 MC_AbortTrigger 3732
18.3.3.3.17 MC_ReadSetPosition 3735
18.3.3.3.18 MC_InPosition 3738
18.3.3.3.19 MC_ChangeGearingRatio 3742
18.3.3.3.20 MC_SetSoftwareLimits 3747
18.3.3.3.21 MC_ReadDigitalInput 3751
18.3.3.3.22 MC_SetControllerMode 3754
18.3.3.3.23 MC_CheckLimits 3759
18.3.3.3.24 MC_SetTorqueLimit 3762
18.3.3.3.25 MC_GetMaxSetAccDec 3766
18.3.3.3.26 MC_GetMaxSetVelocity 3770
18.3.3.3.27 MC_GetTrackingError 3773
18.3.3.3.28 MC_Halt 3777
18.3.3.3.29 MC_Stop 3782
18.3.3.3.30 MC_Home 3787
18.3.3.3.31 MC_MoveAbsolute 3791
18.3.3.3.32 MC_MoveRelative 3809
18.3.3.3.33 MC_MoveModulo 3819
18.3.3.3.34 MC_FollowPosition 3826
18.3.3.3.35 MC_FollowVelocity 3831
18.3.3.3.36 MC_FollowPositionVelocity 3834
18.3.3.3.37 MC_MoveVelocity 3839
18.3.3.3.38 MC_Jog 3845
18.3.3.3.39 MC_TorqueControl 3852
18.3.3.3.40 MC_ImmediateStop 3859
18.3.3.3.41 MC_MoveFeed 3862
18.3.3.3.42 MC_SDORead 3877
18.3.3.3.43 MC_SDOWrite 3881
18.3.3.3.44 MC_PositionFilter 3884
18.3.3.3.45 MC_SetOverride 3890
18.3.3.3.46 MC_ResetFollowingError 3896
18.3.3.3.47 MC_HaltSuperImposed 3901
18.3.3.3.48 MC_MoveSuperImposed 3904

18.3.3.3.49 MC_MoveContinuousAbsolute 3912
18.3.3.3.50 MC_MoveContinuousRelative 3921
18.3.3.3.51 MC_CheckAxisCommunication 3927
18.3.3.3.52 MC_Inch 3930
18.3.3.3.53 MC_Homing 3935
18.3.3.3.54 MC_HomingWithParameter 3951
18.3.3.3.55 MC_SyncMoveVelocityCSV 3958
18.3.3.3.56 MC_BacklashCompensate 3963
18.3.3.3.57 MC_LeadScrewCompensate 3970
18.3.3.3.58 MC_MoveAdditive 3977
18.3.3.3.59 MC_GetGearingRatio 3985
18.3.3.4 Cam 3988
18.3.3.4.1 Instruction List 3988
18.3.3.4.2 MC_GearIn 3989
18.3.3.4.3 MC_GearOut 3995
18.3.3.4.4 MC_Phasing 4000
18.3.3.4.5 MC_StopPhasing 4009
18.3.3.4.6 MC_GearInPos 4012
18.3.3.4.7 MC_GetCamTablePhase 4021
18.3.3.4.8 MC_GetCamTableDistance 4025
18.3.3.4.9 MC_DigitalCamSwitch 4030
18.3.3.4.10 MC_GenerateCamTable 4037
18.3.3.4.11 MC_WriteCAM 4043
18.3.3.4.12 MC_ReadCAM 4046
18.3.3.4.13 MC_CAMBounds 4049
18.3.3.4.14 MC_CamIn 4054
18.3.3.4.15 MC_CamInV2 4083
18.3.3.4.16 MC_CamOut 4126
18.3.3.4.17 MC_CombineAxes 4131
18.3.3.4.18 MC_AxesObserve 4140
18.3.3.4.19 MC_CheckCamTable 4144
18.3.3.4.20 MC_CamNodeBinding 4147
18.3.3.4.21 MC_MoveLink 4149
18.3.3.4.22 MC_PhasingEx 4158
18.3.3.5 Axis Group 4164
18.3.3.5.1 Instruction List 4164
18.3.3.5.2 MC_GroupContinue 4166

18.3.3.5.3 MC_GroupDisable 4170
18.3.3.5.4 MC_GroupEnable 4172
18.3.3.5.5 MC_GroupHalt 4175
18.3.3.5.6 MC_GroupHome 4180
18.3.3.5.7 MC_GroupInterrupt 4183
18.3.3.5.8 MC_GroupReadActualAcceleration 4188
18.3.3.5.9 MC_GroupReadActualPosition 4191
18.3.3.5.10 MC_GroupReadActualVelocity 4193
18.3.3.5.11 MC_GroupReadError 4196
18.3.3.5.12 MC_GroupReadStatus 4199
18.3.3.5.13 MC_GroupReset 4202
18.3.3.5.14 MC_GroupSetOverride 4205
18.3.3.5.15 MC_GroupSetPosition 4209
18.3.3.5.16 MC_GroupStop 4213
18.3.3.5.17 MC_MoveCircularAbsolute 4218
18.3.3.5.18 MC_MoveCircularRelative 4234
18.3.3.5.19 MC_MoveDirectAbsolute 4249
18.3.3.5.20 MC_MoveDirectRelative 4261
18.3.3.5.21 MC_MoveLinearAbsolute 4280
18.3.3.5.22 MC_MoveLinearRelative 4291
18.3.3.5.23 MC_SetCartesianTransform 4304
18.3.3.5.24 MC_TrackConveyorBelt 4308
18.3.3.5.25 MC_GroupPower 4314
18.3.3.6 High-Speed Axis 4317
18.3.3.6.1 Instruction List 4317
18.3.3.6.2 ENC_Counter 4317
18.3.3.6.3 ENC_Reset 4327
18.3.3.6.4 ENC_Preset 4329
18.3.3.6.5 ENC_TouchProbe 4337
18.3.3.6.6 ENC_ArrayCompare 4346
18.3.3.6.7 ENC_StepCompare 4354
18.3.3.6.8 ENC_ReadStatus 4361
18.3.3.6.9 ENC_DigitalOutput 4364
18.3.3.6.10 ENC_Compare 4367
18.3.3.6.11 ENC_SetUnit 4372
18.3.3.6.12 ENC_SetLineRotationMode 4375

18.3.3.7 Drive 4379
18.3.3.7.1 Instruction List 4379
18.3.3.7.2 MC_BackupServoParameter 4379
18.3.3.7.3 MC_RestoreServoParameter 4382
18.3.3.7.4 MC_BackupServoParameter2 4386
18.3.3.7.5 MC_RestoreServoParameter2 4389
18.3.3.8 Fault Codes 4392
18.3.3.9 ParameterNumber 4506
18.3.3.10 Pin Differences 4511
18.3.3.11 Function Differences 4518
18.3.3.12 Appendix A: Examples of Different Buffer Modes 4536
18.3.4 Motor Control 4549
18.3.4.1 Instruction List 4549
18.3.4.2 MOT 4550
18.3.4.3 MOTREV 4559
18.3.4.4 SOVLv 4569
18.3.4.5 VAL 4572
18.3.5 High-Speed I/O Instructions 4581
18.3.5.1 Instruction List 4581
18.3.5.2 HC_PWM 4581
18.3.5.3 HC_EnableInterrupt 4583
18.3.5.4 Fault Codes 4587
18.4 Communication and Diagnostics 4590
18.4.1 System Information 4590
18.4.1.1 Instruction List 4590
18.4.1.2 SysHC_HWInfo 4592
18.4.1.3 SysHC_SWInfo 4595
18.4.1.4 SysHC_NetworkInfo 4597
18.4.1.5 SysHC_NetworkConfig 4600
18.4.1.6 GetTaskInterval 4603
18.4.1.7 GetDeviceDiag 4604
18.4.1.8 GetDeviceStatus 4608
18.4.1.9 GetTaskInfo 4611
18.4.1.10 SysHC_GetPersistentDatainfo 4613
18.4.1.11 SysHC_WritePersistentData 4615
18.4.1.12 SysHC_ReadPersistentData 4618
18.4.1.13 SysHC_Umount 4621

18.4.1.14 SysHC_GetSystemDate 4624
18.4.1.15 SysHC_SetSystemDate 4626
18.4.1.16 SysHC_GetSystemTimeZone 4629
18.4.1.17 SysHC_SetSystemTimeZone 4632
18.4.1.18 SysHC_CPUInfo 4635
18.4.1.19 SysHC_UDiskPath 4638
18.4.1.20 GetBootVersion 4640
18.4.1.21 GetPLCVersion 4641
18.4.1.22 GetProductName 4642
18.4.1.23 GetSerialNumber 4643
18.4.1.24 GetFPGALogicVersion 4644
18.4.1.25 SysHC_Appbackup 4645
18.4.1.26 SysHC_FaultRecord 4648
18.4.1.27 SysHC_OperationRecord 4656
18.4.1.28 SysHC_AppDownInfo 4660
18.4.1.29 UTF16ToGB2312 4662
18.4.1.30 UTF8ToGB2312 4664
18.4.1.31 GB2312ToUTF16 4666
18.4.1.32 GB2312ToUTF8 4668
18.4.1.33 SDCard1StatusGet 4670
18.4.1.34 SysHC_NtpClient 4674
18.4.1.35 SysHC_NetworkInfoV2 4677
18.4.1.36 SysHC_NetworkConfigV2 4679
18.4.1.37 Fault Codes 4682
18.4.2 Redundancy Information 4683
18.4.2.1 Instruction List 4683
18.4.2.2 GetModeChassisID 4684
18.4.2.3 GetRunStopKeyMatchState 4686
18.4.2.4 GetSystemRedundancyMode 4688
18.4.2.5 GetSystemRedundancyState 4690
18.4.2.6 GetControllerModuleDiagnosisInformation 4691
18.4.2.7 GetRedundancyModuleDiagnosisInformation 4695
18.4.2.8 GetTemperature 4698
18.4.3 EtherNetIP 4700
18.4.3.1 Instruction List 4700
18.4.3.2 CIPOpen 4701
18.4.3.3 CIPClose 4705

18.4.3.4 CIPOpenWithDataSize 4708
18.4.3.5 CIPRead 4712
18.4.3.6 CIPWrite 4717
18.4.3.7 CIPSend 4721
18.4.3.8 generic_Service 4726
18.4.3.9 Get_Attributes_All 4732
18.4.3.10 Get_Attribute_Single 4735
18.4.3.11 Set_Attributes_All 4738
18.4.3.12 Set_Attribute_Single 4741
18.4.3.13 GetEIPSystemstats 4744
18.4.3.14 GetEIPConnectstats 4748
18.4.3.15 EIP_SetRemAdapterConfIP 4752
18.4.3.16 SysHC_EthernetIPDeviceDiagnose 4756
18.4.3.17 SysHC_EthernetIPSlaveDiagnose 4757
18.4.3.18 EIP Standard Service Code 4759
18.4.3.19 EIP Standard Fault Codes 4760
18.4.3.20 Fault Codes 4774
18.4.4 Local Bus 4775
18.4.4.1 Instruction List 4775
18.4.4.2 SysHC_CPU_IOMODULE_Diagnose 4775
18.4.4.3 LBus_GetDiagMailBoxData 4777
18.4.4.4 LBus_GetModuleInfo 4780
18.4.4.5 SysHC_GetAllGL20ModuleInfo 4786
18.4.4.6 SysHC_GetGL20ModuleInfo 4789
18.4.4.7 Fault Codes 4792
18.4.5 ModbusRTU 4794
18.4.5.1 Instruction List 4794
18.4.5.2 SysHC_ModbusRtuSlaveDiagnose 4794
18.4.5.3 MB_Connect 4796
18.4.5.4 MB_Master 4801
18.4.5.5 MB_Slave 4806
18.4.5.6 Fault Codes 4811
18.4.5.7 Example of ModbusRTU Communication Instruction Usage 4812
18.4.6 ModbusTCP 4816
18.4.6.1 Instruction List 4816
18.4.6.2 SysHC_ModbusTcpSlaveDiagnose 4816
18.4.6.3 SysHC_ModbusTcpDeviceDiagnose 4818

18.4.6.4 MB_TCP_Connect 4819
18.4.6.5 MB_Server 4823
18.4.6.6 MB_Client 4828
18.4.6.7 Example of ModbusTCP Communication Instruction Usage 4833
18.4.7 Free Serial Port 4839
18.4.7.1 Instruction List 4839
18.4.7.2 SerialRcv 4840
18.4.7.3 SerialSend 4844
18.4.7.4 SerialRcvV2 4848
18.4.7.5 SerialSendV2 4853
18.4.7.6 Serial_Connect 4856
18.4.7.7 Serial_Recv 4860
18.4.7.8 Serial_Recv_By_Idle_Time 4864
18.4.7.9 Serial_Recv_Ch 4868
18.4.7.10 Serial_Send 4872
18.4.8 Free Network Port 4875
18.4.8.1 Instruction List 4875
18.4.8.2 TRCV_C 4875
18.4.8.3 TSEND_C 4881
18.4.8.4 TRCV_CV2 4887
18.4.8.5 TSEND_CV2 4893
18.4.8.6 Example of Free Port Communication Instruction Usage 4898
18.4.9 EtherCAT 4902
18.4.9.1 Instruction List 4902
18.4.9.2 ETC_SDORead 4902
18.4.9.3 ETC_SDOWrite 4906
18.4.9.4 ETC_COE_SdoRead 4910
18.4.9.5 ETC_COE_SdoWrite 4917
18.4.9.6 ETC_ClearSlavesErrorCounter 4924
18.4.9.7 ETC_DisableDevice 4926
18.4.9.8 ETC_GetCouplerInfo 4933
18.4.9.9 ETC_GetMasterinfo 4938
18.4.9.10 ETC_GetSlaveInfo 4943
18.4.9.11 ETC_SlaveReset 4948
18.4.9.12 ETC_MasterRestart 4951
18.4.9.13 ETC_MasterDiagnose 4953
18.4.9.14 ETC_SDO_ERROR: ETC Fault Codes 4956

18.4.9.15 ETC_STD_ERROR: ETC Fault Codes 4956
18.4.9.16 ETC_CO_ERROR: ETC Fault Codes 4957
18.4.9.17 SDO Fault Codes 4958
18.4.9.18 IO Fault Codes 4958
18.4.10 CANBus 4963
18.4.10.1 Instruction List 4963
18.4.10.2 CANBus_CmpDrvVer_t: Version of the Drive Component Used by CANBus 4964
18.4.10.3 CANBus_DiagInfo_t: CANBus Diagnostic Information Structure 4964
18.4.10.4 CANBus_FrameMsg_t: CANBus Frame Message Structure 4965
18.4.10.5 CANBus_RxFilter_t: CANBus Receive Frame Filter Structure 4965
18.4.10.6 CANBus_BaudrateType_Enum: CANBus Baud Rate Enumeration 4966
18.4.10.7 CANBus_CtrlCmdType_Enum: CANBus Control Command Enumeration 4967
18.4.10.8 CANBus_ErrorType_Enum: CANBus Error Enumeration 4967
18.4.10.9 CANBus_FBErrorType_Enum: CANBus Function Block Error Enumeration 4967
18.4.10.10 CANBus_FBStateType_Enum: CANBus Function Block State Enumeration 4968
18.4.10.11 CANBus_StateType_Enum : CANBus State Enumeration 4968
18.4.10.12 CANBus_CtrlDevState 4968
18.4.10.13 CANBus_DevRxMsg 4973
18.4.10.14 CANBus_DevRxMultiMsg 4977
18.4.10.15 CANBus_DevTxMsg 4983
18.4.10.16 CANBus_GetDevDiagInfo 4987
18.4.10.17 CANBus_GetIoDrvVer 4990
18.4.10.18 CANBus_SetDevBaud 4994
18.4.11 CIA405 4997
18.4.11.1 Instruction List 4997
18.4.11.2 CIA405_GET_LOCAL_NODE_ID 4998
18.4.11.3 CIA405_GET_STATE 5001
18.4.11.4 CIA405_NMT 5005
18.4.11.5 CIA405_RECV_EMCY_DEV 5010
18.4.11.6 CIA405_SDO_READ4 5014
18.4.11.7 CIA405_SDO_WRITE4 5018
18.4.11.8 COP_GET_MST_STATISTICS 5024
18.4.11.9 Fault Codes 5026
18.4.12 Free TCP 5027
18.4.12.1 Instruction List 5027
18.4.12.2 SktTCPServer 5028
18.4.12.3 SktTCPClient 5030

18.4.12.4 SktTCPSend 5032
18.4.12.5 SktTCPRecv 5033
18.4.12.6 SktTCPGetStatus 5036
18.4.12.7 SktTCPClearBuffer 5038
18.4.12.8 SktTCPRecv_C 5039
18.4.12.9 SktTCPSend_C 5042
18.4.12.10 SktClose 5045
18.4.12.11 Fault Codes 5046
18.4.12.12 Example of Free TCP Communication Instruction Usage 5047
18.4.13 Free UDP 5062
18.4.13.1 Instruction List 5062
18.4.13.2 SktUDPCreate 5062
18.4.13.3 SktUDPRecv 5063
18.4.13.4 SktUDPSend 5066
18.4.13.5 SktUDPJoinGroup 5067
18.4.13.6 SktUDPClearBuffer 5069
18.4.13.7 SktClose 5070
18.4.13.8 Example of Free UDP Communication Instruction Usage 5071
18.4.14 S7 Communication 5076
18.4.14.1 Instruction List 5076
18.4.14.2 S7_Client 5076
18.4.14.3 S7_Server 5079
18.4.14.4 S7_Get 5081
18.4.14.5 S7_Put 5086
18.4.14.6 S7_Data_Area_Create 5089
18.4.14.7 Fault Codes 5092
18.4.15 FINS Communication 5093
18.4.15.1 Instruction List 5093
18.4.15.2 FINS_Server 5093
18.4.15.3 FINS_ServerDiag 5095
18.4.15.4 Example of FINS Communication Instruction Usage 5098
18.4.16 Melsec Communication 5100
18.4.16.1 Instruction List 5100
18.4.16.2 MELSEC_OpenServer 5100
18.4.16.3 MELSEC_CloseServer 5102
18.4.16.4 MELSEC_ServerDiag 5103
18.4.16.5 MELSEC_SetServerSoftAddr 5104

18.4.16.6 Fault Codes 5107
18.4.16.7 Example of Melsec Communication Instruction Usage 5108
18.4.17 SFTP Communication 5110
18.4.17.1 Instruction List 5110
18.4.17.2 SFTP_GetFileList 5110
18.4.17.3 SFTP_GetFile 5113
18.4.17.4 SFTP_PutFile 5117
18.4.17.5 SFTP_DeleteFile 5120
18.4.17.6 SFTP_MoveFile 5122
18.4.17.7 SFTP_RenameFile 5125
18.4.17.8 SFTP_CreateDir 5128
18.4.17.9 Fault Codes 5131
18.4.18 SysPub 5133
18.4.18.1 Instruction List 5133
18.4.18.2 SysHC_PNDPExtendDeviceDiagnose 5133
18.4.18.3 Diagnostic Codes 5137
18.4.19 VarForce 5138
18.4.19.1 Instruction List 5138
18.4.19.2 CheckPrgForceValueState 5138
18.5 Special Instructions 5139
18.5.1 Instruction Description 5139
18.5.2 Util 5139
18.5.2.1 Instruction List 5139
18.5.2.2 BASE64 5139
18.5.3 StringUtils 5146
18.5.3.1 Ansi 5146
18.5.3.1.1 Instruction List 5146
18.5.3.1.2 StrCaseCmpA 5147
18.5.3.1.3 StrCaseCmpEndA 5149
18.5.3.1.4 StrCaseCmpStartA 5151
18.5.3.1.5 StrCaseFindA 5153
18.5.3.1.6 StrCmpA 5155
18.5.3.1.7 StrCmpEndA 5157
18.5.3.1.8 StrCmpStartA 5159
18.5.3.1.9 StrConcatA 5161
18.5.3.1.10 StrCpyA 5164
18.5.3.1.11 StrDeleteA 5167
18.5.3.1.12 StrFindA 5169

18.5.3.1.13 StrIsNullOrEmptyA 5171
18.5.3.1.14 StrLenA 5173
18.5.3.1.15 StrMidA 5175
18.5.3.1.16 StrReplaceA 5179
18.5.3.1.17 StrToLowerA 5184
18.5.3.1.18 StrToUpperA 5186
18.5.3.1.19 StrTrimA 5188
18.5.3.1.20 StrTrimEndA 5190
18.5.3.1.21 StrTrimStartA 5192
18.5.3.1.22 CharToUpper 5194
18.5.3.2 Unicode 5195
18.5.3.2.1 Instruction List 5195
18.5.3.2.2 StrCaseFindW 5196
18.5.3.2.3 StrCmpW 5198
18.5.3.2.4 StrConcatW 5200
18.5.3.2.5 StrCpyW 5203
18.5.3.2.6 StrDeleteW 5206
18.5.3.2.7 StrFindW 5209
18.5.3.2.8 StrIsNullOrEmptyW 5211
18.5.3.2.9 StrLenW 5213
18.5.3.2.10 StrMidW 5214
18.5.3.2.11 StrReplaceW 5219
18.5.3.2.12 WCharToUpper 5224
18.5.3.3 UTF8 5226
18.5.3.3.1 Instruction List 5226
18.5.3.3.2 ConvertUTF16toUTF8 5226
18.5.3.3.3 ConvertUTF8toUTF16 5227
18.5.3.3.4 IsLegalUTF8 5229
19 HMI System Functions 5232
19.1 Variables 5232
19.2 Bit Operations 5233
19.3 Calculation 5234
19.3.1 Increasing the Variable Value 5234
19.3.2 Decreasing the Variable Value 5235
19.3.3 Setting the Variable Value 5236
19.3.4 Getting the Variable Value 5236
19.3.5 Linear Scaling 5237

19.3.6 Inverse Linear Scaling 5237
19.3.7 Generating a Random Value 5238
19.4 Screen 5238
19.4.1 Activating a Screen 5238
19.4.2 Activating a Screen by Number 5239
19.4.3 Activating the Previous Screen 5239
19.4.4 Showing the Popup Screen 5240
19.4.5 Hiding a Popup Screen 5241
19.4.6 Hiding All Popup Screens 5241
19.4.7 Showing the Message Box 5241
19.4.8 Getting Parameters from a Popup Screen 5242
19.5 Alarm 5243
19.5.1 Acknowledging an Alarm in the Alarm View 5243
19.6 Record 5244
19.6.1 Clearing a Record 5244
19.6.2 Starting Recording 5244
19.6.3 Stopping Recording 5245
19.6.4 Starting All Records 5245
19.6.5 Stopping All Records 5245
19.6.6 Exporting a Data Record 5246
19.6.7 Exporting an Alarm Record 5247
19.6.8 Exporting an Operation Record 5247
19.6.9 Exporting Data Records by Time 5248
19.7 Recipe 5250
19.7.1 Creating a Recipe Data Record 5250
19.7.2 Deleting a Recipe Data Record 5251
19.7.3 Setting a Recipe Data Record to PLC 5252
19.7.4 Getting a Recipe Data Record from PLC 5252
19.7.5 Getting the Recipe Data Record Name 5253
19.7.6 Exporting Recipe Data Records 5254
19.7.7 Importing Recipe Data Records 5255
19.8 User Management 5256
19.8.1 Displaying the Login Dialog 5256
19.8.2 Login 5257
19.8.3 Logout 5257
19.8.4 Getting the Username 5258
19.8.5 Switching the User 5259

19.8.6 Changing the Password 5260
19.8.7 Exporting User Data 5261
19.8.8 Importing User Data 5262
19.9 Configuration 5263
19.9.1 Opening the Control Panel 5263
19.9.2 Setting the IP Address 5263
19.9.3 Setting the Screensaver Time 5264
19.9.4 Setting the Backlight Brightness 5265
19.9.5 Setting the Backlight Mode 5266
19.9.6 Setting the Backlight Time 5266
19.9.7 Setting the LED Mode 5267
19.9.8 Playing a Beep 5267
19.9.9 Setting the Sound Mode 5268
19.9.10 Restart 5268
19.9.11 Safely Removing the Hardware 5269
19.9.12 Setting the Language 5269
19.9.13 Unlocking Installments 5270
19.9.14 Triggering Installments in Advance 5271
19.9.15 Minimizing the Software 5271
19.9.16 Exiting the Software 5271
19.10 Date and Time 5272
19.10.1 Setting the Date and Time 5272
19.10.2 Setting the Time Zone 5273
19.10.3 Setting the Date 5273
19.10.4 Setting the Time 5274
19.10.5 Getting the Date and Time 5275
19.10.6 Getting the Timestamp 5275
19.11 Print 5276
19.11.1 Screenshot 5276
19.12 VNC Operations 5277
19.12.1 Setting the VNC Enable 5277
19.12.2 Setting the VNC Username 5278
19.12.3 Setting the VNC View-Only Password 5278
19.12.4 Setting the VNC Control Password 5279
19.12.5 Setting the VNC Control Permission 5280
19.13 Timer 5282

20 Hardware Device 5284
20.1 Introduction to Hardware Guides 5284
20.2 PLC 5284
20.2.1 EVO400 5284
20.2.2 EVO500 5285
20.2.3 EVO700 5285
20.2.4 EVO700C 5285
20.2.5 EVO800 5286
20.2.6 EVO1000 5286
20.2.7 APC 5286
20.3 HMI 5287
20.3.1 IT9000 5287
20.4 I/O Modules 5287
20.4.1 GL10 5287
20.4.1.1 I/O Coupler 5287
20.4.1.2 DI 5287
20.4.1.3 DO 5287
20.4.1.4 AI 5288
20.4.1.5 AO 5288
20.4.1.6 Temperature 5288
20.4.2 GL20 5288
20.4.2.1 I/O Coupler 5288
20.4.2.2 DI 5289
20.4.2.3 DO 5290
20.4.2.4 DI/DO 5291
20.4.2.5 AI 5292
20.4.2.6 AO 5293
20.4.2.7 Temperature 5293
20.4.2.8 Communication 5294
20.4.2.9 Power Supply 5295
20.4.2.10 Encoder 5295
20.4.2.11 High-Speed Counting 5296
20.4.3 GR10 5296
20.4.3.1 DI/DO 5296
20.4.3.2 AI 5296
20.4.3.3 AO 5296
20.4.3.4 Temperature 5297
20.4.3.5 Pulse Input 5297

20.4.3.6 Pulse Output 5297
20.4.3.7 EtherCAT Splitter 5297
20.4.4 GR20 5298
20.4.4.1 Digital Slave 5298
20.4.4.2 Optical Fiber Converter 5298
20.4.5 GR20T 5299
20.4.5.1 DI 5299
20.4.5.2 DO 5299
20.4.5.3 DI/DO 5299
20.4.6 GS20 5300
20.4.6.1 IO-Link Master 5300
20.4.6.2 Digital Slave 5300
20.4.6.3 Analog Slave 5303
20.4.7 GE20 5303
20.4.7.1 Local Expansion Module 5303
20.4.8 Hi50 5304
20.4.8.1 I/O Coupler 5304
20.4.8.2 DI 5305
20.4.8.3 DO 5305
20.4.8.4 AI 5305
20.4.8.5 Temperature 5306
20.4.8.6 Stopper Plate 5306
20.4.9 Hi80 5306
20.4.9.1 I/O Coupler 5306
20.4.9.2 DI 5307
20.4.9.3 DO 5307
20.4.9.4 AI 5307
20.4.9.5 AO 5308
20.4.9.6 Temperature 5308
20.5 Wireless Module 5308
20.6 Valve Terminal 5308
20.7 Servo Drive 5309
20.7.1 Single-Axis Servo Drive 5309
20.7.2 Multi-Axis Servo Drive 5311
20.7.3 Direct Drive Servo Drive 5311
20.7.4 Low-Voltage Servo Drive 5311
20.7.5 Stepper Servo Drive 5312

20.8 Low-Voltage AC Drive 5312
20.8.1 General-Purpose Low-Voltage AC Drive 5312
20.8.2 Industrial AC Drive 5312
21 Appendix 5313
21.1 Appendix 1: List of Fault Codes 5313
21.2 Appendix 2: Version Compatibility Table 5578
21.3 Appendix 3: PN/DP Master Diagnostic Codes 5594

1 Legal Information

Copyright

This documentation is the exclusive property of Shenzhen Inovance Technology Co., Ltd. No individual or entity may excerpt, reproduce, modify, translate, or distribute any content herein without written consent from Inovance.

Legal action will be taken against infringement.

Trademarks

INOVANCE is a registered trademark of Shenzhen Inovance Technology Co., Ltd. and its affiliates. All other trademarks or registered trademarks mentioned in this documentation are the property of their respective owners. Unauthorized use of these trademarks by third parties for any purpose without written authorization could violate the rights of their owners.

Disclaimer of Liability

Due to continuous updates and improvements of products and technologies, the content of this documentation may not fully match the actual products. In the event of any discrepancies, the actual products shall prevail.

The contents are subject to change without notice due to product upgrade.

Waste Disposal

The storage, use, and disposal of this product (including optional accessories) must comply with local laws and regulations.

Qualifed Personnel

The product/system described in this documentation may be operated only by personnel qualified for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel can identify the risks of the product/system and prevent possible dangers.

Proper Use of the Product

Proper transportation, storage, assembly, installation, commissioning, operation, and maintenance are required to ensure the safe operation of the product without any problems. The required ambient conditions must be met. All operations must follow the guidelines provided in this documentation.

2 Version Information

2.1 New Features in V1.5.2.5

2.1.1 Introduction

This section summarizes all the important new features compared to software V1.4.4.5. For more details on each topic, see corresponding sections in the product documentation.

2.1.2 Newly Supported Hardware Models

This section summarizes the newly added hardware product series. For more details on each topic, see the product documentation.

PLC

Product Series	Firmware Version
EVO522-0808TN	V1.7
EVO410-A12N16	V1.5
EVO410-A14N16	V1.5
EVO410-A15N16	V1.5
EVO410-A17N16	V1.5
EVO510-A12P16	V1.5
EVO510-A14P16	V1.5
EVO510-A15P16	V1.5
EVO800-CM-DPM	V1.2
EVO800-CM-PNM	V1.2

Safety PLC

Product Series	Firmware Version
EVO1000EF-F50	V1.1
EVO1000EF-F51	V1.1
EVO1000EF-F52	V1.1

Monitor

Product Series	Firmware Version
IT9100E	V1.0
IT9070E	V1.0

Drive

Product Series	Firmware Version
SV630NS	H01-00=1042.0/H00-02=000.00

Product Series	Firmware Version
MD520	F7-10=U60.12/F7-11=U61.12/F7-15=000.00/F7-16=000.00

2.1.3 New Software Features

This section summarizes the important new features of the software. For more details on each topic, see corresponding sections in the documentation.

Feature	Description	Applicable Products
Library project security management (new)	Supports security settings for library projects. For details, see 9.15 Library Project Security Management.	General
HMI security authentication policies (new)	Added HMI security authentication policies. For details, see 9.6 HMI Security Authentication Policies.	General
Collapsing project child nodes with one click (new)	All child nodes under the PLC can be collapsed. For details, see 8.14 Collapsing Project Child Nodes with One Click.	General
Uninstalling the software while keeping user configuration data	Reinstallation does not require reconfiguring user information.	General
Inverting object selection in the view area (new)	The parent node objects in the view area can be inverted. For details, see 5.5.8 Inverting Object Selection.	General

PLC

Feature	Description	Applicable Products
Default gateway configuration (new)	Supports enabling the default gateway. For details, see 10.6.14.4 Ethernet Address Settings.	EVO400, EVO500, EVO700, APC, EVO800
Importing/Exporting a PLCopenXML file (updated)	PLCopenXML files can be imported and exported using LD and CFC languages. For details, see 8.12 Importing/Exporting a PLCopenXML File.	Full product series
Overwriting existing data during the PLC upload	Project upload supports overwriting existing data. For details, see 10.9.1 Uploading.	Full product series

Feature	Description	Applicable Products
Watch table and power failure retention table (optimized)	Removed view restrictions for watch tables, and power failure retention tables, supporting viewing all variables by mouse scrolling. For details, see visible element range setting and power failure retention table.	Full product series
Offline variable table (optimized)	Expansion of up to 100,000 child elements are supported in offline variable tables.	Full product series
Library reference (updated)	System library references are bound to a placeholder library by default. For details, see library referencing and dereferencing.	Full product series
Finding and replacing within a project (updated)	Search in both source code libraries and projects can be performed simultaneously. For details, see 8.7 Finding and Replacing Within a Project.	Full product series
CFC programming language (optimized)	• Pins can be hidden and pin comments can be edited through a right-click pop-up dialog. For details, see 10.1.5.4.3.1 Configuring Operation Block Properties. • Added the feature of quick operation block alignment. For details, see 10.1.5.4.3.4 Repositioning an Operation Block. • Supports quick one-to-many connection. For details, see 10.1.5.5.4 Connecting Input and Output Parameters. • Added the feature of distinguishing connection line colors by data type. For details, see 10.1.5.5.6 Changing Connection Line Color. • Added the feature of text connection. For details, see 10.1.5.5.8 Text Connection.	Full product series

Feature	Description	Applicable Products
SFC programming language (optimized)	• Supports dual-line display. For details, see 10.1.6.3.2.1 Setting the SFC Editor in iFA Options. • Supports enabling or disabling all networks with a single click. For details, see 10.1.6.3.5 Opening/Closing All Program Segments. • Supports the power failure retention property setting. For details, see 10.1.6.3.2.3 Setting SFC Program Block Power Failure Retention Property. • Supports viewing the SFC view by right-clicking and dragging. For details, see 10.1.6.3.3 SFC Programming Window. • Updated some UIs. For details, see the actual display in the software.	Full product series
Program block access permission settings (updated)	For details, see 10.1.2.5 Program Block Access Permissions.	Full product series
Variable comment display method (new).	Instance comments can be edited. For details, see 10.1.3.3.2 LD Editor Settings.	Full product series
Cumulative timer and sequencer output instructions (new)	For details, see 18.1.9.13 AccumulationTimer2and 18.2.20.2 SQO.	Full product series
General instruction of delay timer (updated)	The general instruction of delay timer supports modification of the ET pin. For details, see 18.1.9.5 TON2.	Full product series
Added the feature of task startup initialization.	For details, see 10.5.3 System Event Management.	Full product series
Copying and pasting between the variable table and Excel.	For details, see 10.1.1.10.2 Copying and Pasting a Variable Table.	Full product series
Batch reset/set instructions (new)	For details, see 18.1.4.10 RESET_BFand 18.1.4.11 SET_BF.	Full product series
Instruction of conversion between string and numeric formats (new)	For details, see 18.2.5.21 AnyNumFmt.	Full product series

Feature	Description	Applicable Products
Instruction for calling C function (new)	For details, see 18.2.19.4 IOBlockLibCalland 18.2.19.5 IOBlockLibProcess.	Full product series
Instructions of conversion between GB2312 encoding and STRING/WSTRING (new)	For details, see 18.4.1.29 UTF16ToGB2312, 18.4.1.30 UTF8ToGB2312, 18.4.1.31 GB2312ToUTF16, and 18.4.1.32 GB2312ToUTF8.	Full product series
Instruction of reading axis parameters (new)	For details, see 18.3.3.3.59 MC_GetGearingRatio.	EVO400, EVO500, EVO700, EVO700C, APC520, and APC760C
Instruction of getting all the status flags of SD memory card (new)	For details, see 18.4.1.33 SDCard1StatusGet.	Full product series
In-position check for motion control instructions	For details, see each instruction section.	EVO400, EVO500, EVO700, EVO700C, APC520, and APC760C
I/O mapping variable binding (updated)	I/O mapping supports one-click binding of structures. For details, see 10.6.3.3.9 I/O Mapping.	Full product series
Using one-dimensional Bit array in unions	For details, see 10.1.1.8.2.2 UNION.	Full product series
Renaming EVO760C	Modified the device name for EVO760C series PLCs. For details, see 20.2.4 EVO700C.	EVO760C
Power failure data management (new).	Data import and export can be performed during power failure. For details, see 10.11.10 Power Failure Data Management.	Full product series
Instruction Of Rotary Axis Modulo Positioning (new)	For details, see 18.3.3.3.33 MC_MoveModulo.	EVO400, EVO500, EVO700, EVO700C, APC520, and APC760C
Instruction of clearing UDP stack buffer (new)	For details, see 18.4.13.6 SktUDPClearBuffer.	Full product series
NTP time synchronization instruction (new)	For details, see 18.4.1.34 SysHC_NtpClient.	Full product series
Instructions of getting network information and setting network properties (updated)	The instructions of getting network information and setting network properties support checking the enable status of the default gateway. For details, see 18.4.1.35 SysHC_NetworkInfoV2and 18.4.1.36 SysHC_NetworkConfigV2.	Full product series
Instruction of processing special files (new)	XML file processing is supported. For details, see 18.2.12 Special File Processing.	EVO400, EVO500, EVO700, EVO700C, APC520, and APC760C

Feature	Description	Applicable Products
EtherCAT master configuration parameter "Overview" interface (updated)	The "Overview" interface includes "Associated array" and "Array index" items. For details, see 10.6.3.2.6 Overview.	EVO400, EVO500, EVO700, EVO700C, APC520, and APC760C
Advanced process control instructions (optimized)	For details, see 18.3.2.12 PFC11, 18.3.2.13 PFC22, and 18.3.2.14 PFC_DV.	APC520
Oscilloscope (optimized)	Updated the oscilloscope UI and basic operations on the interface. For details, see 10.2 Oscilloscope.	Full product series
Phase offsetting for task configuration	For details, see 10.5.2 Configuring Task Parameters.	EVO800
PN/DP master expansion module (new)	For details, see 10.6.5.3 PN Master Configuration (EVO800)and 10.6.10 DP Communication Configuration (EVO800).	EVO800
SysPub instruction (new)	For details, see 18.4.18 SysPub.	EVO800
PN/DP master diagnostic codes (new)	For details, see 21.3 Appendix 3: PN/DP Master Diagnostic Codes.	EVO800
VarForce instruction (new)	For details, see 18.4.19.2 CheckPrgForceValueState.	EVO800

Safety PLC

Feature	Description	Applicable Products
Data types (new)	• Added the data types of DWORD and DATE_AND_TIME. For details, see 11.2.1.1.1 Standard Data Types. • Added the data types of SAFEDWORD and SAFE_DATE_AND_TIME. For details, see 11.2.1.1.2 Safety Data Types.	Full product series of safety PLCs
Safety PLC models and communication unit models (new)	For details, see 11.4.1 Hardware Configuration and Device Management.	Full product series of safety PLCs
EtherNet/IP communication with HMI (new)	For details, see 11.4.3.1.2 EtherNet/IP Communication with HMI.	Full product series of safety PLCs
Modbus TCP communication (new)	For details, see 11.4.3.3 Modbus TCP Communication Configuration.	Full product series of safety PLCs
Communication between safety PLCs and standard PLCs via EtherCAT, communication between safety PLCs via FSoE, and compatibility of FSoE-capable servo devices with safety PLCs (new)	Added the EtherCAT communication. For details, see 11.4.4 EtherCAT Communication Configuration.	Full product series of safety PLCs

Feature	Description	Applicable Products
Operation log (new)	For details, see 11.8.4 Operation Log.	Full product series of safety PLCs
Safety PLC simulation (new)	For details, see 11.9 Safety PLC Simulation Mode.	Full product series of safety PLCs
Safety PLC firmware upgrade (EVO1000) introduction (new)	For details, see 14.1.1.6 Upgrading Safety PLC Firmware(EVO1000).	Full product series of safety PLCs
DWORD data conversion operator instructions (new) and data types of all conversion operator instructions (updated)	For details, see 11.10.1.6 Conversion Operators.	Full product series of safety PLCs
Safety function block instructions (new)	For details, see 11.10.2.2.35 SF_GetSafetyConfirmInfo, 11.10.2.2.36 SF_GetCouplerState, 11.10.2.2.37 SF_GetGlBusState, 11.10.2.2.38 SF_GetEtherNetIpStateCode, 11.10.2.2.39 SF_GetFSoeStateCode, and 11.10.2.2.40 SF_GetCIPSafetyStateCode.	Full product series of safety PLCs
Extended function block instructions (new)	For details, see 11.10.2.3 Extended Function Blocks.	Full product series of safety PLCs
Overall resource usage of FSoE (new)	For details, see the "Equipped with an ECT communication unit" section under 11.2.1.2.5 Resource Usage Table.	Full product series of safety PLCs
Safety library project (updated) and risk warnings (new)	For details, see 11.2.2 Library Reference.	Full product series of safety PLCs
FSoE master functionality, enabling local or remote communication with failsafe I/O modules (new)	Added the FSoE configuration. For details, see 11.4.2.7 Configuring a Failsafe I/O Module.	Full product series of safety PLCs
DLR networking on the EIP2 port (new)	For details, see 11.8 PLC Online Operations.	Full product series of safety PLCs
SF_SafetyConnStatus instruction (updated)	For details, see 11.10.2.2.31 SF_SafetyConnStatus.	Full product series of safety PLCs
"Instruction," "Library," and "Applicable Library Version" in instruction lists of all common safety instructions (updated)	For details, see 11.10 Common Safety Instructions.	Full product series of safety PLCs

Monitor

Feature	Description	Applicable Products
Installment payment (new)	For details, see 13.11 Installment Payment.	HMI products
Reusing HMI program across projects	For details, see 13.18 HMI Program Reuse Across Projects.	HMI products

Feature	Description	Applicable Products
Image resource management (optimized)	For details, see 13.3.6 Gallery Usage.	HMI products
Automatic screen scaling across HMI devices	For details, see 13.3.10.4 Changing Screen Resolution.	HMI products

2.2 New Features in V1.4.4.5

2.2.1 Introduction

This section summarizes all the important new features compared with software V1.4.3.5. For more details on each topic, see corresponding sections in the product documentation.

2.2.2 Newly Supported Hardware Models

This section summarizes the newly added hardware product series. For more details on each topic, see the product documentation.

PLC

Product Series	Firmware Version
EVO760C	V1.5
EVO780C	V1.5
APC760C	V1.5
EVO800-CM-EN	V1.1
EVO810-A10	V1.1
EVO810-A30	V1.1
EVO820-A30	V1.1
EVO820-A50	V1.1
EVO410-A12P16	V1.5
EVO410-A14P16	V1.5
EVO410-A15P16	V1.5
EVO410-A17P16	V1.5

Drive

Product Series	Firmware Version
SV660ND	H01-00=2401.9/H00-02=000.00
SV630ND	H01-00=2501.6/H00-02=000.00
IS810N multi-axis	H01-00=5702.5/H00-02=000.00

2.2.3 New Software Features

This section summarizes the important new features of the software. For more details on each topic, see corresponding sections in the documentation.

Feature	Description	Applicable Products
Logging in to technical forum using an account (new)	Supports logging in to the iFA software, Inovance Planet, and course platform with the same account. For details, see 3.3.4 Technical Forum.	General
Separate installation and uninstallation of AI Assistant	Supports separate installation and uninstallation of the AI Assistant. For details, see 3.4 AI Assistant3.4 installation and uninstallation.	General
Diagnostics (new)	Updated the diagnostic methods and improved the related content. For details, see 15 Diagnostics.	General

PLC

Feature	Description	Applicable Products
Export method for source code library (new)	Supports saving library files as source code libraries. For details, see 12.1.4 Source Code Library.	Full product series
Library version upgrade or downgrade (new)	Supports one-click upgrade or downgrade of the library version when a device is changed. For details, see One-click upgrade/downgrade.	Full product series
Library version switching (new)	Supports changing the dependent library version in a project. For details, see Library version switching.	Full product series
Basic LD instructions (new)	For details, see 10.1.3.3.4 LD Elements and 10.1.3.3.5 LD Instructions.	Full product series
Network title navigation in LD editor area	For details, see Network title navigation area.	Full product series
Column operations in ST editor	For details, see Column operations.	Full product series
Array Intellisense in ST editor	For details, see Direct input.	Full product series
Region navigation in ST editor	For details, see Region navigation area.	Full product series
Breakpoint (optimized)	Added the Ctrl+F9 shortcut to disable breakpoints. The "Enable breakpoint" menu item is now renamed "Enable/Delete breakpoint". For details, see 10.10.4 Breakpoint Debugging.	Full product series
Variable category change (new)	Supports changing the categories of POU input/output variables. For details, see 10.1.1.10.4 Changing Variable Category.	Full product series

Feature	Description	Applicable Products
Configuration of external event task (new)	Supports selecting external event task as the task type. For details, see Task types.	EVO500 and APC500
EIP remote adapter status variable (new)	Supports selecting remote adapter status variable as the system variable. For details, see Remote adapter status variable.	Full product series
I/O mapping import/ export (new)	Supports I/O mapping import/export of the hardware configuration. For details, see 10.6.1.4 Importing10.6.1.4 I/O Mappingand 10.6.1.3 Exporting10.6.1.3 I/O Mapping.	Full product series
EIP quick connection (new)	Supports EtherNet/IP quick connection. For details, see 10.6.4.2.8 Quick Connect Communication.	EVO700 and EVO800
Wireless master station (new)	Supports adding the wireless management station under the EtherCAT node. For details, see 10.6.3.1.3 Adding a Wireless Management Station.	EVO500, EVO700, and APC500
Instruction cheat sheet (new)	For details, see each instruction section.	Full product series
Mathematical operation instructions (new)	For details, see 18.1.5.34 RAD, 18.1.5.35 DEG, 18.1.5.36 SCALE_X, and 18.1.5.37 NORM_X.	Full product series
Logic operation instructions (new)	For details, see 18.1.4.6 TestABit, 18.1.4.7 TestABitN, 18.1.4.8 SetBits, and 18.1.4.9 ResetBits.	Full product series
System time instruction (new)	For details, see 18.2.1.8 GetSystemTime.	Full product series
Data processing instructions (new)	For details, see 18.1.2.10 Byte_To_HexString, 18.1.2.11 Word_To_HexString, and 18.1.2.12 Dword_To_HexString.	Full product series
EtherCAT instructions (new)	For details, see 18.4.9.4 ETC_COE_SdoReadand ETC_COE_SdoWrite .	EVO500, EVO700, and APC500
S7 communication instruction (new)	For details, see 18.4.14 S7 Communication.	Full product series
Free serial port instructions (new)	For details, see 18.4.7.6 Serial_Connect, 18.4.7.7 Serial_Recv, 18.4.7.8 Serial_Recv_By_Idle_Time, 18.4.7.9 Serial_Recv_Ch, and18.4.7.10 Serial_Send.	Full product series
Power failure retention instructions (new)	For details, see 18.4.1.10 SysHC_GetPersistentDatainfo, 18.4.1.11 SysHC_WritePersistentData, and 18.4.1.12 SysHC_ReadPersistentData.	Full product series

Feature	Description	Applicable Products
System information instructions (new)	For details, see 18.4.1.13 SysHC_Umount, 18.4.1.14 SysHC_GetSystemDate, 18.4.1.15 SysHC_SetSystemDate, 18.4.1.16 SysHC_GetSystemTimeZone, 18.4.1.17 SysHC_SetSystemTimeZone, 18.4.1.18 SysHC_CPUInfo, 18.4.1.19 SysHC_UDiskPath, 18.4.1.20 GetBootVersion, GetPLCVersion, 18.4.1.22 GetProductName, 18.4.1.23 GetSerialNumber, 18.4.1.24 GetFPGALogicVersion, and 18.4.1.25 SysHC_Appbackup.	Full product series
Instruction for setting remote adapter IP via EIP (new)	For details, see 18.4.3.15 EIP_SetRemAdapterConfIP.	Full product series
Instructions for getting EIP system status and connection status (new)	For details, see 18.4.3.13 GetEIPSystemstatsand 18.4.3.14 GetEIPConnectstats.	Full product series
EIP instructions (new)	For details, see 18.4.3.8 generic_Service, 18.4.3.9 Get_Attributes_All, 18.4.3.10 Get_Attribute_Single, 18.4.3.11 Set_Attributes_All, and 18.4.3.12 Set_Attribute_Single.	Full product series
High-speed I/O instruction (new)	For details, see 18.3.5 High-Speed I/O Instructions.	EVO500, EVO700, and APC500
Recipe library instructions (new)	For details, see 18.2.18.1.31 CreateRecipeV3, 18.2.18.1.32 DeleteALLRecipeV3, 18.2.18.1.33 DeleteRecipeV3, 18.2.18.1.34 GetRecipeDefinitionInfoV3, 18.2.18.1.35 GetRecipeInfoV3, 18.2.18.1.36 LoadRecipeV3, 18.2.18.1.37 ReadRecipeV3, 18.2.18.1.38 RenameRecipeV3, 18.2.18.1.39 SaveRecipeV3, 18.2.18.1.40 SetRecipeValuesV3, 18.2.18.1.41 WriteRecipeV3, and 18.2.18.1.42 SetStoragePathV3.	Full product series
EtherCAT slave reset instruction (new)	For details, see 18.4.9.11 ETC_SlaveReset.	EVO500, EVO700, and APC500
Memory comparison instruction (new)	For details, see 18.2.17.5 AnyMemCmp.	Full product series
Local bus instructions (new)	For details, see 18.4.4.4 LBus_GetModuleInfo, 18.4.4.5 SysHC_GetAllGL20ModuleInfo, and 18.4.4.6 SysHC_GetGL20ModuleInfo.	EVO500, EVO700, and APC500
Axis array variables (new)	The system variables include axis array variables. For details, see Axis array variables.	EVO500, EVO700, and APC500

Feature	Description	Applicable Products
Upper limit of string length (adjusted)	The STRING length is increased from 256 to 4096 bytes, and the WSTRING length is increased from 512 to 16382 bytes. For details, see 10.1.1.8.1 Standard Data Types. The instruction string length is increased from 256 to 1985 bytes.	Full product series
Axis commissioning interface (updated)	Updated the configuration UI for axis commissioning. For details, see 16.1 Axis16.1 Configuration.	EVO500, EVO700, and APC500
PLC simulation mode (updated)	Updated the UI for PLC simulation. For details, see 10.13 PLC Simulation Mode.	Full product series
File operation instruction (updated)	Supports file operations for the user directory on the PLC. For details, see 18.2.11 File Operations.	Full product series
Instruction for disabling ECAT device (updated)	Updated the instruction for disabling ECAT device - slave disable function. For details, see 18.4.9.7 ETC_DisableDevice.	EVO500, EVO700, and APC500
Removing variable download restrictions for oscilloscope	For details, see 10.2 Oscilloscope.	Full product series
ECAT instruction renaming	ETCMaster_Restart is renamed to ETC_MasterRestart. For details, see 18.4.9.12 ETC_MasterRestart. SysHC_ETCMasterDiagnose is renamed to ETC_MasterDiagnose. For details, see 18.4.9.13 ETC_MasterDiagnose.	EVO500, EVO700, and APC500
DHCP configuration method (new)	Added the DHCP configuration method for Ethernet. For details, see 10.6.14.4 Ethernet Address Settings.	EVO500, EVO700, and APC500
EtherCAT scanning (optimized)	For details, see 10.6.3.1.4 Adding an Online EtherCAT Slave to Project.	EVO500, EVO700, and APC500
EtherCAT slave system variables (updated)	Added ConfigProductID, ProductID, and Name to the EtherCAT slave system variable structure members. For details, see EtherCAT slave system variables.	EVO500, EVO700, and APC500
EtherCAT master configuration parameters (updated)	Optimized the Overview interface. For details, see 10.6.3.2.6 Overview.	EVO500, EVO700, and APC500
EtherCAT slave configuration parameters (updated)	Optimized the EoE interface. For details, see 10.6 .3.3.6 EoE Settings.	EVO500, EVO700, and APC500

Feature	Description	Applicable Products
AXIS_REF: New axis structure variables	Added wAxNo to the axis structure variables. For details, see 18.3.3.2.2.1 AXIS_REF: Axis Structure.	EVO500, EVO700, and APC500
New trigger mode for MC_TouchProbe	Adjusted the valid range of the input parameter TriggerMode for MC_TouchProbe to 0-2, and updated the continuous trigger mode. For details, see 18.3.3.3.10 MC_TouchProbe.	EVO500, EVO700, and APC500
ALT instruction format (updated)	Adjusted the variable xOut type of ALT from output variable to in-out variable. The corresponding library version is Util 1.2.0.0 or later. For details, see 18.1.1.8 ALT.	Full product series
Function block (FB) declaration restriction	The FB is not allowed to be declared as a stack variable. For details, see function block instantiation.	Full product series
Ethernet expansion module property configuration (new)	For details, see 10.6.15 Ethernet Expansion Module Property Configuration.	EVO800-CM-EN
PRP monitoring (new)	For details, see 10.11.23 PRP monitoring.	EVO800-CM-EN

Monitor

Feature	Description	Applicable Products
Multiple languages (new)	HMI interface supports multi-language switching. For details, see 13.17 Multilingual Settings.	IT9000 series HMI
Multimedia control (new)	See 13.3.5.5.6 Multimedia.	IT9000 series HMI
Batch editing of alarms	Supports importing or exporting the alarm variables. For details, see 13.5.7 Alarm13.5.7 Import/Export.	IT9000 series HMI
Delayed triggering of button events	Supports delayed triggering for events associated with 13.3.5.2.5 Bit Button and 13.3.5.2.6 Word Button.	IT9000 series HMI
Mitsubishi Q series drive (new)	For parameter configuration, see 13.1.6 Supported Device Drives.	IT9000 series HMI

Drive

Feature	Description	Applicable Products
Servo common function - multi-axis device (new)	Added a servo common function of multi-axis device. For details, see 17.3.3 Multi-Axis Devices.	SV660ND, SV630ND, and IS810N

2.3 New Features in V1.4.3.5

2.3.1 Introduction

This section summarizes all the important new features compared with software V1.4.2.5. For more details on each topic, see corresponding sections in the product documentation.

2.3.2 Newly Supported Hardware Models

This section summarizes the newly added hardware product series. For more details on each topic, see the product documentation.

PLC

Product Series	Firmware Version
APC520-A10P(N)16	V1.4

Drive

Product Series	Firmware Version
SV680N	H01-00=4002.5/H00-02=000.00
SV680N-INT	H01-00=2000.1/H00-02=000.00
SV680L	H01-00=4701.6/H00-02=000.00
SV640N	H01-00=3604.8/H00-02=000.00
SV670N-LK	H01-00=5001.6/H00-02=180.02
SVD60N	H01-00=1600.4/H00-02=000.00
IS810N single axis	H01-00=5701.9/H00-02=000.00
MD630N	A2-22=U63.02/A2-23=U63.04/A2-24=000.00/A2-25=000.00
MD810	F7-10=L02.03/F7-11=L03.04/F7-15=000.01/F7-16=000.00
MD800 rectifier	F0-02=U05.07/F0-03=000.00
MD800 inverter	F7-10=U80.05/F7-11=U84.04/F7-15=000.00/F7-16=000.00

2.3.3 New Software Features

This section summarizes the important new features of the software. For more details on each topic, see corresponding sections in the documentation.

Feature	Description	Applicable Products
Floating window (new)	Added the floating window. For details, see 5.2 Project View.	General
PLCopenXML import and export (new)	Added the feature of PLCopenXML import and export. For details, see 8.12 Importing and Exporting a8.12 PLCopenXML File.	General

Feature	Description	Applicable Products
Batch installation and uninstallation of device files (new)	Supports batch installation and uninstallation of device files. For details, see 8.15.1 Installing a Deviceand 8.15.3 Uninstalling a Device.	General

PLC

Feature	Description	Applicable Products
CFC programming language (new)	Added the CFC language to programming languages. For details, see 10.1.5 CFC Language Programming.	Full product series
Template creation for user-defined types in PLC (new)	Supports template creation for user-defined data types in the PLC. Fore details, see Structure templating.	Full product series
Function block property (new)	Supports adding properties for the function block. For details, see 10.1.2 .3.4 Adding Properties to a Function Block.	Full product series
Whole-structure access to tag structures (new)	Supports whole-structure access to EIP tag communication variable types. For details, see 10.6.4.2.7 Communication with HMI.	Full product series
Web server for EVO500 (new)	EVO500 supports Web server. For details, see 10.6.14.7 Web Serverand 10.11.12 Web Diagnostics and Monitoring.	EVO500
Memory copy instruction (new)	For details, see 18.2.17 Memory Copy.	Full product series
Instruction for converting UNICODE to UTF8 (new)	For details, see 18.1.12.24 UNICODE_TO_UTF8.	Full product series
Instruction for deleting all recipes (new)	For details, see 18.2.18.1.5 DeleteAllRecipe.	Full product series
Instruction for binding cam node (new)	For details, see 18.3.3.4.20 MC_CamNodeBinding.	EVO500, EVO700, and APC500
Instructions for backing up servo parameters and restoring servo parameters (new)	For details, see 18.3.3.7.4 MC_BackupServoParameter2 and 18.3.3.7.5 MC_RestoreServoParameter2.	EVO500, EVO700, and APC500
Instruction for advanced process control (new)	For details, see 18.3.2 Advanced Process Control.	APC500
EtherNet/IP online scan to add device (new)	Supports EtherNet/IP online scan to add a device. For details, see 10.6.4.2.1 Adding an EtherNet/IP Device.	Full product series

Feature	Description	Applicable Products
100 ms timer instruction (updated)	For details, see 18.1.9.15 Timer100ms.	Full product series
Instructions for getting device diagnostic information (new)	For details, see 18.4.1.7 GetDeviceDiag.	Full product series
Instructions for getting device status (new)	For details, see 18.4.1.8 GetDeviceStatus.	Full product series
Timed pulse output instructions (new)	For details, see 18.2.7.5 GetClk_1ms, 18.2.7.6 GetClk_10ms, 18.2.7.7 GetClk_20ms, 18.2.7.8 GetClk_100ms, 18.2.7.9 GetClk_1s, and 18.2.7.10 GetClk_1min.	Full product series
Instruction for calling C function (new)	For details, see 18.2.19 C Function Support.	EVO500, EVO700, and APC500
Free TCP instruction (new)	For details, see 18.4.12 Free TCP.	EVO500, EVO700, and APC500
Free UDP instruction (new)	For details, see 18.4.13 Free UDP.	EVO500, EVO700, and APC500
Homing method filter option in axis parameter configuration (new)	For details, see 16.1.5.5 Homing Parameter Settings.	EVO500, EVO700, and APC500
Adding remote I/O module	For details, see 10.6.11.3 Adding a Remote I/O Module.	Full product series
Delta configuration in axis group (new)	For details, see 16.2.2.1 Mechanical Types, 16.2.2.2 Structural Parameters, and 16.2.2.3 Axis Configuration.	EVO500, EVO700, and APC500
AXIS_REF: New axis structure variables	The new axis structure variables include fSetJerk, fFollowingError, bHomed, uiPhysSlaveAddr, and uiAliasSlaveAddr. For details, see 18.3.3.2.2.1 AXIS_REF: Axis Structure.	EVO500, EVO700, and APC500
Program check (new)	Supports program check. For details, see program check.	Full product series
Parameter update (new)	Supports parameter update. For details, see 10.1.1.10.9 Parameter Update.	Full product series
Program block callee update (new)	Supports program block update callees. For details, see 10.1.2.4 Program Block10.1.2.4 Callee Update.	Full product series
EtherCAT online and overview interfaces (optimized)	For details, see 10.6.3.2.5 Online and 10.6.3.2.6 Overview.	EVO500, EVO700, and APC500
Oscilloscope (optimized)	For details, see 10.2 Oscilloscope.	Full product series
ST language programming (optimized)	For details, see 10.1.4 ST Language Programming.	Full product series

Feature	Description	Applicable Products
LD language programming (optimized)	For details, see 10.1.3 LD Language Programming.	Full product series
Upper limit of public variables (adjusted)	Adjusted the upper limit of public variables. For details, see network public property.	Full product series

Monitor

Feature	Description	Applicable Products
Module channel fault monitoring for system diagnostics control	For details, see 13.3.5.4.7 System Diagnostics.	IT9000 series HMI
New device driver–Rockwell (AB) EtherNet/IP	HMI devices support communication with ControlLogix series PLCs. For configuration information, see 13.1.6.3 Communication with Rockwell (AB) Devices.	IT9000 series HMI
Expansion of user-defined types in module library	User-defined structures in the PLC can be dragged into the template library to achieve templating. For related operations, see 13.4.4 User-defined Types and 13.4.6 Common Operations on Templates.	IT9000 series HMI
Cross-project template copy/paste	When a template instance is copied/pasted, the associated templates are also pasted. For related requirements and operations, see 13.4.7 Copying/ Pasting a Template Across Projects and 13.4.8 Copying/Pasting a Template Instance Across Projects.	IT9000 series HMI
Customizing pop-up window location	The display position of a pop-up screen can be set as a constant. For position settings, see 19.4.4 Showing the Popup Screen.	IT9000 series HMI
Group management of user scripts	Supports group management of user scripts. For related operations, see 13.10.2.2 Grouping13.10.2.2 Scripts.	IT9000 series HMI
Batch editing of the same type of control properties	Supports batch editing of the same type of control properties on the screen. For the applicable scope, see 13.3.7.20 Setting Properties in Batch.	IT9000 series HMI

Feature	Description	Applicable Products
Extended time format for "Date-time field" control	Supports display of system time that consists of date and time. For details, see 13.3.5.2.3 Date-Time Domain.	IT9000 series HMI
Displaying linkage name of external variables	Set whether to automatically display the linkage name through configuration options. For configuration details, see 13.2.5.2.2 Naming an External Variable.	IT9000 series HMI

Drive

Feature	Description	Applicable Products
AC drive common function - parameter list	Added an AC drive common function of parameter list. For details, see 17.2.2.6 Parameter List.	MD800, MD810, and MD630N
AC drive common function - fault management	Added an AC drive common function of fault management. For details, see 17.4.1 Fault Management.	MD800, MD810, and MD630N
AC drive common function - control panel	Added an AC drive common function of control panel. For details, see 17.4.2 Control Panel.	MD800 inverter, MD810 inverter, and MD630N
AC drive common function - function diagram	Added an AC drive common function of function diagram. For details, see Function Diagram.	MD630N
AC drive common function - motor auto-tuning	Added an AC drive common function of motor auto-tuning. For details, see 17.4.4 Motor Auto-Tuning.	MD630N
AC drive common function - black box	Added an AC drive common function of black box. For details, see 17.4.5 Black Box.	MD630N
AC drive common function - I/O monitoring	Added an AC drive common function of I/O monitoring. For details, see 17.4.6 I/O Monitoring.	MD800 rectifier

2.4 New Features in V1.4.2.5

2.4.1 Introduction

This section summarizes all the important new features compared with software V1.3. For more details on each topic, see corresponding sections in the product documentation.

2.4.2 Newly Supported Hardware Models

This section summarizes the newly added hardware product series. For more details on each topic, see the product documentation.

PLC

Product Series	Firmware Version
EVO521-0808TN(TP)	V1.3
EVO522-0808TN(TP)	V1.3
EVO523-0808TN(TP)	V1.3
EVO763-0804TN	V1.1
EVO764-0804TN	V1.1
EVO765-0804TN	V1.1
EVO766-0804TN	V1.1
EVO810	V1.0
EVO820	V1.0
EVO1000EF	V1.0

Drive

Product Series	Firmware Version
SV630N	H01-00=3704.3/H00-02=000.00
SV660N	H01-00=0904.3/H00-02=000.00
SV670N	H01-00=5002.5/H00-02=000.00
SV635N	H01-00=3404.6/H00-02=000.00
STD60N	H01-00=6000.4/H00-02=000.00
IS650N-PLUS	H01-00=1110.2/H00-02=000.00
SV660NC	H01-00=5201.9/H00-02=000.00
SV660NS	H01-00=1022.0/H00-02=000.00

2.4.3 New Software Features

This section summarizes the important new features of the software. For more details on each topic, see corresponding sections in the documentation.

Feature	Description	Applicable Products
Project comparison (new)	Project comparison can be performed before downloading. For more details, see 10.8.4 Online Modification/Monitoring without Modification.	General
Update check (optimized)	Supports software version upgrade. For details, see 4.5 Upgrading the4.5 Software.	General
Enabling project protection (new)	Supports enabling project protection during project permission setting. For details, see 9.3 Enabling Project Protection.	General

Feature	Description	Applicable Products
AI Assistant (new)	Added the feature of AI assistant. For details, see 3.4 AI Assistant.	General
Project upgrade (new)	Project upgrade is required to open a project created in the historical software version in a new software version. For more details, see 8.11 Upgrading a8.11 Project.	General
Batch setting of axis parameters (new)	Supports batch setting of axis parameters when a device has multiple axes. For details, see 16.1.4 Batch Servo Assignment and Axis Parameter Settings.	General
CFC programming language (new)	Added the CFC language to programming languages. For details, see Programming with CFC Language.	General
Template library (optimized)	Supports expanding the template configuration and version management. For details, see 13.4 Template Library Usage.	General

PLC

Feature	Description	Applicable Products
CANopen communication protocol (new)	Added the CANopen communication protocol in hardware configuration. For details, see 10.6.6 CANopen Communication Configuration.	EVO500 and EVO700
CAN free protocol (new)	Added the CAN free protocol in hardware configuration. For details, see 10.6.7 CAN Free Communication Configuration.	EVO500 and EVO700
PROFINET communication protocol (new)	Added the PROFINET communication protocol in hardware configuration. For details, see 10.6.5 PROFINET Communication Configuration.	EVO500 and EVO700
Network settings (new), supporting independent IP settings for dual network ports	Added the network settings in the property configuration of the network configuration view. For details, see 10.6.14.2 Network settings.	EVO500

Feature	Description	Applicable Products
Ethernet settings (new)	Added the Ethernet settings in the property configuration of the network configuration view. For details, see 10.6.14.4 Ethernet Address Settings.	EVO500 and EVO700
PLC permission configuration	Added the PLC permission configuration in online operations. For details, see 10.11.6 PLC Permission Configuration.	Full product series
Firmware upgrade (new)	Added the firmware upgrade method. For details, see 14.1.1.3 Upgrading14.1.1.3 Firmware via TypeC.	EVO500 and EVO700
User program upgrade method (new)	Added the user program upgrade via TypeC and Ethernet. For details, see 14.1.2.3 Upgrading a User Program via Type-C/Ethernet.	EVO500 and EVO700
Ethernet loop monitoring (new)	Supports monitoring the Ethernet loop status when the device is online. For details, see 10.11.17 Ethernet Loop10.11.17 Monitoring.	EVO500, EVO700, and EVO800
OPC UA diagnostics (new)	Supports displaying the status through OPC UA diagnostics when the device is online. For details, see 10.6.14.5.3 OPC UA Diagnostics.	EVO500, EVO700, and EVO800
PID instruction auto-tuning (new)	Added the auto-tuning algorithms for PID_AT and PID_ATC instructions. For details, see 18.3.1.5 PID_ATand 18.3.1.6 PID_ATC.	EVO500, EVO700, and EVO800
EtherNetIP communication instruction (new)	For details, see 18.4.3 EtherNetIP.	EVO500 and EVO700
ModbusRTU communication instruction (new)	For details, see 18.4.5 ModbusRTU.	EVO500 and EVO700
ModbusTCP communication instruction (new)	For details, see 18.4.6 ModbusTCP.	EVO500 and EVO700
CAN free communication instruction (new)	For details, see 18.4.10 CANBus.	EVO500 and EVO700
CANopen communication instruction (new)	For details, see 18.4.11 CIA405.	EVO500 and EVO700

Monitor

Feature	Description	Applicable Products
Operation record control (new)	For details, see 13.3.5.4.8 Operation Logs.	IT9000 series HMI
HMI project upload	Supports uploading the HMI device projects to the project. For details, see 13.14.3 Project13.14.3 Upload.	IT9000 series HMI
HMI project screen capture	Supports screen capture and storage of the monitoring interface during the HMI project operation. For details, see 19.11.1 Screenshot.	IT9000 series HMI
Communication with S7-300 PLC	Supports the communication between the HMI device and S7-300 PLC. For configuration information, see 13.1.6.2.1 S7-300 PLC.	IT9000 series HMI

Drive

Feature	Description	Applicable Products
Fully closed-loop function debugging (new)	Added the feature of fully closed-loop function debugging. For details, see 17.3.19 Fully Closed-Loop Function.	IS650N-PLUS
Current loop tuning (new)	Added the feature of current loop tuning. For details, see 17.3.18 Current Loop Tuning.	IS650N-PLUS
Single triggering of oscilloscope	Supports single triggering of the oscilloscope. For details, see 17.2.1.5 Triggering the Oscilloscope.	SV630N, SV630NS, SV660N, SV670N, SV680N, SV635N, IS650N-PLUS
Process segment (new)	Added the feature of process segment. For details, see 17.3.21 Process Segment.	STD60N
Overload protection (new)	Added the feature of overload protection. For details, see 17.3.22 Overload Protection.	SV680N-INT, SV670N-INT, and SV680LN
Motor management library (new)	Added a servo commissioning function of motor management library. For details, see 17.3.23 Motor Management Library.	SV680N-INT, SV670N-INT, and SV680LN
Accuracy compensation (new)	Added a servo commissioning function of accuracy compensation. For details, see 17.3.24 Accuracy Compensation.	SV680N-INT, SV670N-INT, and SV680LN

Feature	Description	Applicable Products
Cogging compensation (new)	Added a servo commissioning function of cogging compensation. For details, see 17.3.25 Cogging Compensation.	SV680N-INT, SV670N-INT, and SV680LN
Gantry synchronization (new)	Added a servo commissioning function of gantry synchronization. For details, see 17.3.26 Gantry Synchronization.	SV680N-INT and SV680LN
Motion unit configuration (new)	Add a servo commissioning function of motion unit configuration. For details, see 17.3.27 Motion Unit Configuration.	SV680N-INT, SV670N-INT, and SV680LN
Motion trajectory and simulation (new)	Add a servo commissioning function of motion trajectory and simulation. For details, see 17.3.28 Motion Trajectory and Simulation.	SV660NS, SV660NC, SV660N, SV635N, SV630N, SV630NS, IS650N-PLUS, SV680N, SV680N-INT, SV670N-INT, and SV680LN

2.5 Compatibility

Software V1.4.2.5 and later support firmware updates for different PLCs. For details, see 14.1.1 Firmware Update.

When the updated PLC firmware is incompatible with the PLC program, the user program and related configurations can be cleared.

The firmware version of the actual PLC can be greater than or equal to that of the configured PLC in the iFA.

Projects are backward compatible, which means that later version software can open projects configured in earlier version software, but earlier version software cannot open projects configured in later version software.

When some feature permissions are incompatible, upgrading or downgrading the firmware automatically deletes the old permissions in the PLC.

2.6 Feature Upgrade

Provided that the device firmware version remains unchanged, the feature upgrade supports online modification of the device or a full download to the device.

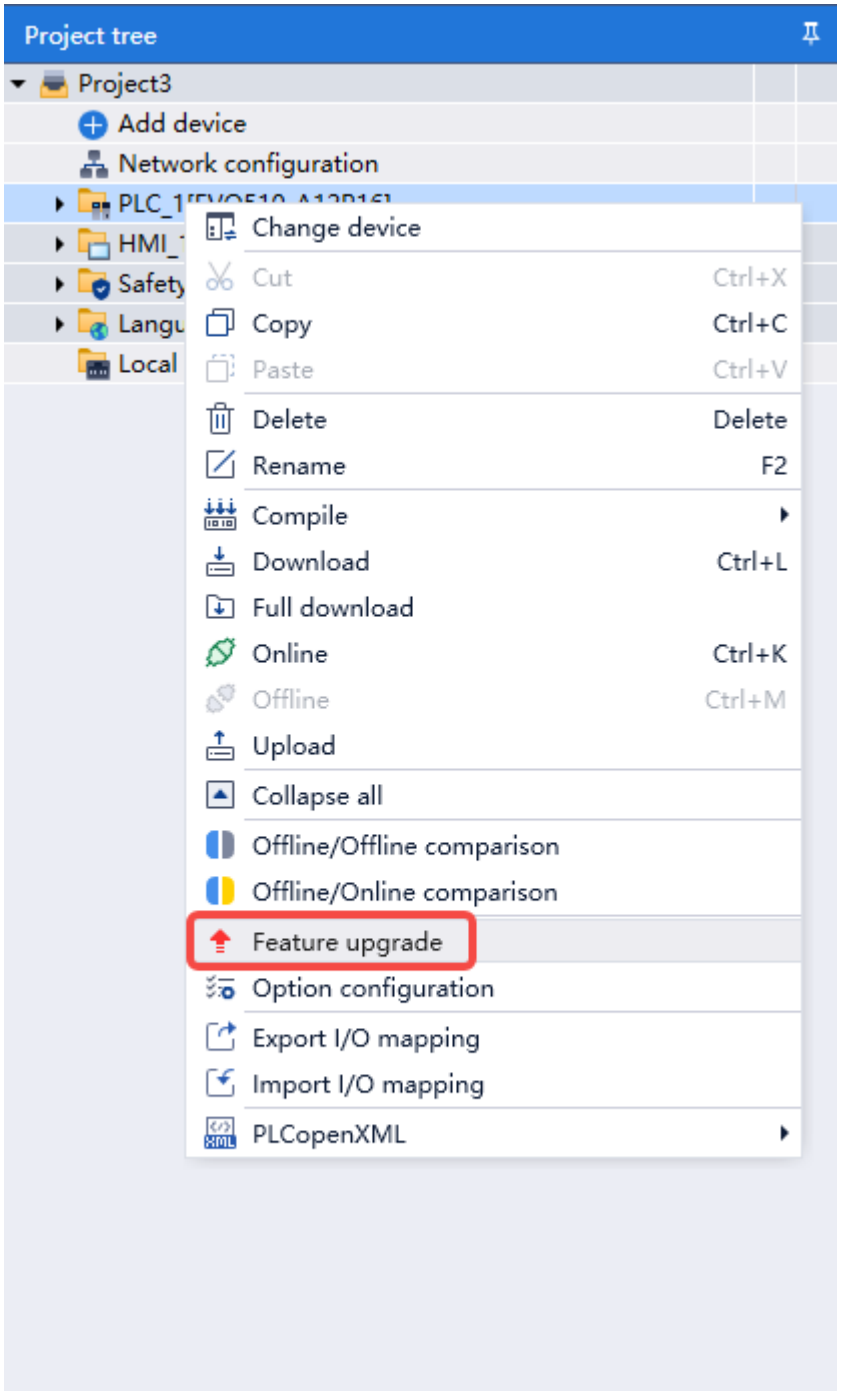
Device feature upgrade

Prerequisites

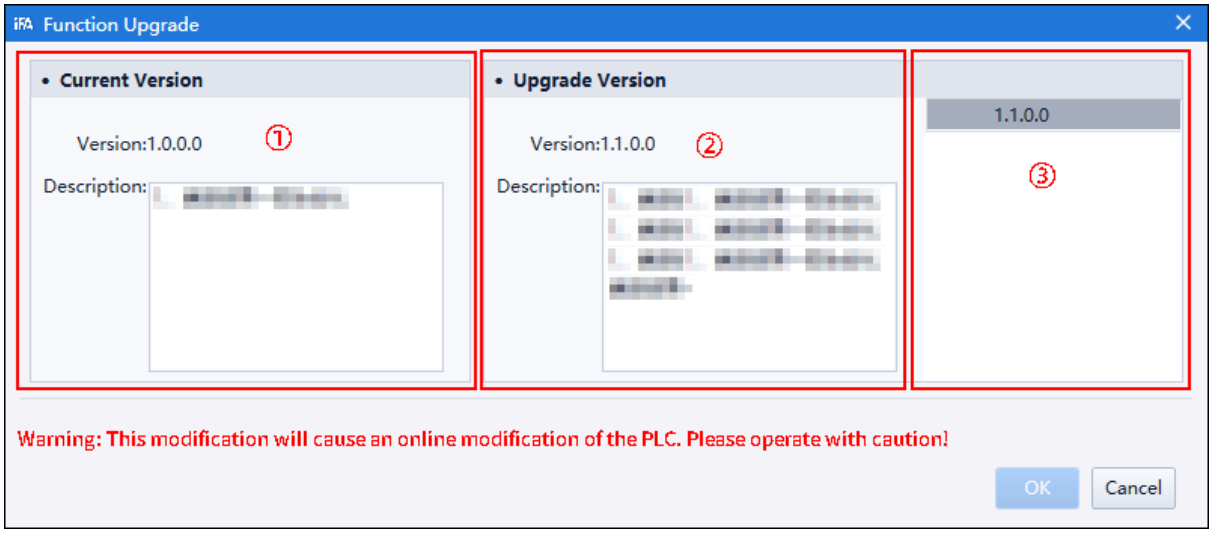
- A project has been created.
- A device has been added.

Procedure

1. Right-click the device node in the left project tree, and select "Feature upgrade" to access the "Feature upgrade" tab.



2. Select the upgrade version number and confirm the relevant feature information.



①: The feature version at the time of creating the current PLC node or the version after the last feature upgrade and the description of the current feature version.

②: The upgrade version number and related description.

③: Supported upgrade versions of the feature.

3. Click "OK" to perform feature upgrade.

> **NOTE**
>
> Before feature upgrade, the original project is backed up by default to the backup folder, with the file named in the format "Projectname_BackUp_YYYYMMDDHHMMSS". You can also choose a directory for backup.

Library project feature upgrade

The feature upgrade procedure for a library project is similar to the that for a device.

3 Introduction to iFA Evolution

3.1 Software Overview

iFA Evolution is an intelligent industrial control software platform developed by Inovance to support diverse industrial scenarios. It enables the development and maintenance of fully integrated automation systems for the entire factory, production lines, and individual machines of modern digital factories. Currently, it can be used for PLC configuration and development, HMI configuration and development, servo commissioning, as well as PLC and HMI simulation. As its functions continue to improve, it will support more scenarios and products in the future.

3.2 Features

The iFA Evolution software development environment includes the following software components:

Logic: PLC configuration and programming commissioning software, including single PLC and redundant PLC systems;

View: HMI system configuration and programming commissioning software;

Drive: Drive commissioning software.

You can selectively install the above components during installation based on actual usage.

All of the above software supports online updates, ensuring that users can promptly receive the latest software to experience new products and features. Moreover, the software provides unified permission management, allowing centralized control and flexible allocation of PLC and HMI development and operation permissions.

With a fully integrated development environment, iFA Evolution improves project development efficiency by functions such as variable referencing and variable dragging.

3.3 Help and Support

3.3.1 Help System

The iFA help system integrates the software help guide and can provide detailed guidance on basic functions and complex operation steps for users who are new to the iFA software. It helps users quickly find resolutions to problems encountered while using the software, such as function details and error message meanings. The system covers both basic operation instructions and advanced features and application cases, helping users have a deeper understanding of the software applications and expand their knowledge.



The help guide can only be accessed with network connectivity; otherwise, issues such as lagging or errors may occur.

Procedure

1. Choose “Help -> Help System” from the menu bar, or press the shortcut <F1>.
2. Access the help system interface that consists of search area, navigation area, and content area, as shown in the following figure.
   ①: Search area

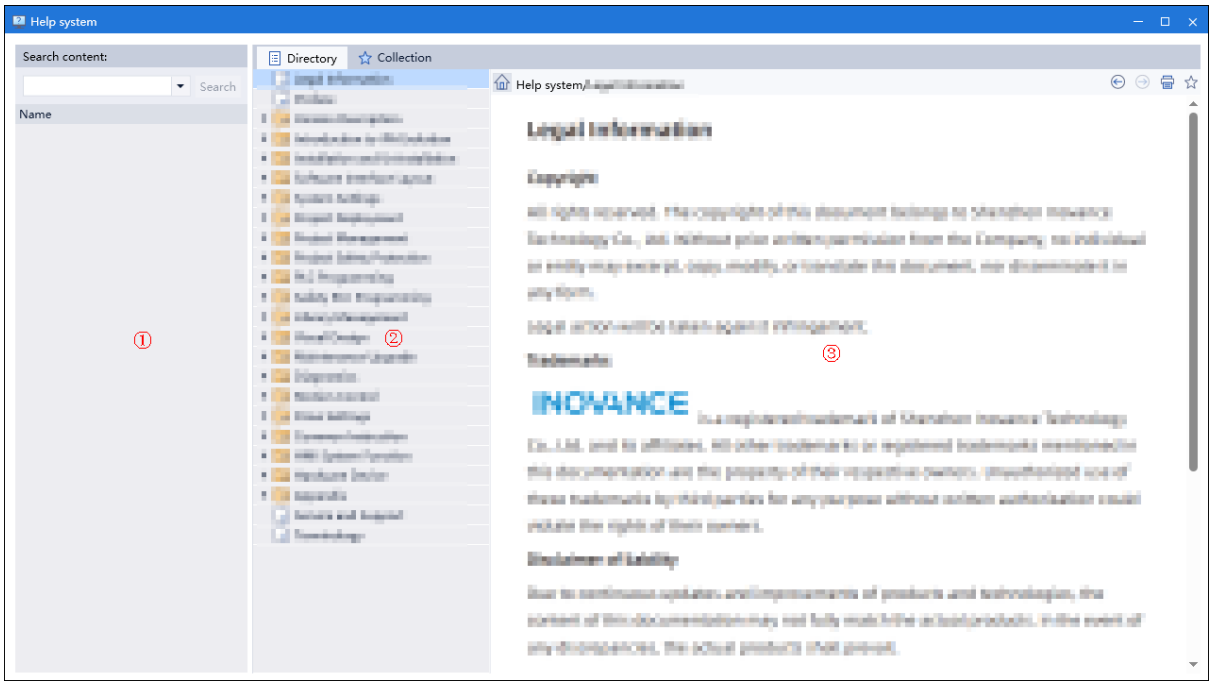
   Enter a keyword in the search box, click the "Search" button or press Enter to perform a full-text search in the help guide. The search results are displayed below the search box. The search history can be found in the drop-down list of the search box.

   ②: Navigation area

   Includes the Contents and Favorites tabs, which display the guide's contents and the favorites, respectively.

   ③: Content area

   Displays the specific page of the guide.



3. Click the page you want to view in the Contents of the navigation area, or double-click a page in the search results list. The page is displayed in the content area.
4. The functions of the icons are described in the following table.

Icon	Function	Note
	Indent	Adjust the width of the search area.
	Previous/Next	Go to previous/next page among the opened pages.
	Print	Print the page of the help guide.

Icon	Function	Note
☆	Favorite	Add the current page to the Favorites. Favorited pages are displayed in the Favorites of the navigation area. Even if the help system is closed, the content of the Favorites will not be lost, saving time for subsequent searches of such pages.

3.3.2 Online Help

The iFA online help integrates the software help guide and can provide detailed guidance on basic functions and complex operation steps for users who are new to the iFA software. It helps users quickly find resolutions to problems encountered while using the software, such as function details and error message meanings. The online help covers both basic operation instructions and advanced features and application cases, helping users have a deeper understanding of the software applications and expand their knowledge.

Users can access the online help via a browser on any devices connected to the Internet. There are two ways to access the online help.

- Choose "Help > Online help" from the menu bar in the iFA software.
- Use the shared web page link.

Homepage

The online help homepage includes the following contents.

No.	Name	Description
①	Management tab	Includes the Homepage and Contents. • Homepage: Displays the home interface of the online help. • Contents: Switches to the contents interface of the online help.
②	-	ⓘ Click the icon to access the online help license agreement on the web page. ⊙ Click the icon to enter access the contents interface of the online help. Click the icon to collapse the left management tabs; click to unfold the management tabs.

No.	Name	Description
③	Filter and search	• Filter: Filter products by type. When a type is selected, the search is performed within the help information related to the selected product type. The product types available for filtering include PLC (all, EVO500, EVO700, EVO800, EVO1000EF), HMI, Drive, I/O (Hi50, Hi80). • Search: Enter a keyword in the search box, and then click or press Enter to perform a full-text search in the help guide. Click the title of a search result to redirect to the target page, or click in the search results area to open the guide in a new browser tab.
④	Function navigation	Provides quick access to four sections: Getting Started, PLC Programming, Motion Control, and Visual Configuration. Clicking any section navigates to the default page of that section.
⑤	Additional features	Not available yet.

Contents

The online help contents include the following:

No.	Name	Description
①	Contents bar	Contents navigation for the help guide. Enter a keyword in the search box and press Enter to search the contents of the help guide. The matched entries will be highlighted.
②	My favorites	Displays the favorited pages.
③	Advanced features area	Provides access to favorite, copy link, and print current page functions.
④	Content	Displays the content of the current page.

My favorites

1. Select the target section, and then click in the "Advanced features area".
2. In the opened "Add to my favorites" dialog box, edit the name as needed, and then click "OK".
3. View the favorites in the "My Favorites" area. Proceed as follows to unfavorite a page:
   - In the "My favorites" area, click to remove all favorites.
   - Open a page you have favorited, and then click the Favorite icon in the "Advanced features area" to unfavorite the current page.

Sharing

1. Select the target section, and then click in the "Advanced features area" to copy the current page link.
2. Share the link by social media.

Printing current page

1. Select the target section, and then click in the "Advanced features area" to go to the print page.
2. Configure print parameters on the print page, and then click "Print".

3.3.3 Issue Feedback

In the iFA software, the Issue Feedback feature provides you with a direct channel to communicate with the technical support team, facilitating the continuous optimization and upgrade of the software. This feature includes submitting issue reports and feature suggestions, and collecting error information.

> NOTE
>
> Network connectivity is required to open and use the feedback related functions.

Issue report/Feature suggestion

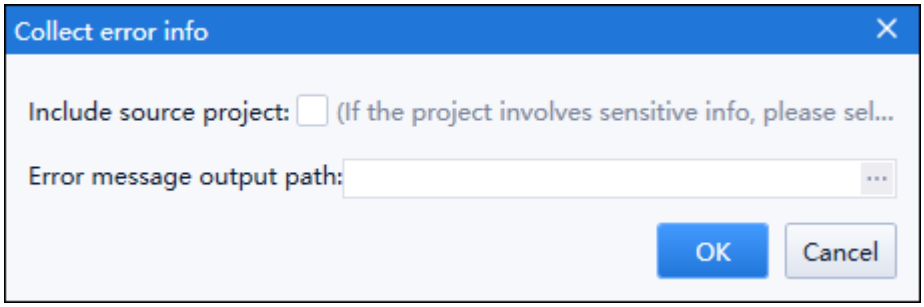
1. Choose “Help -> Issue Feedback -> Issue Report/Feature Suggestion” from the menu bar.
2. The system opens the relevant feedback page in the browser. On the opened page, fill in related information about issue report or feature suggestion, and upload attachments as needed.
   - Issue report: Provide feedback on the issue report page in case of any abnormalities while using the iFA software, such as software crashes, performance degradation, or feature malfunction. Describe the issue in detail, including the scenario where the issue occurred, the steps taken, the frequency/time of occurrence, any possible triggering conditions, and the software version number and operating system information at the time. Upload proper attachments. This information helps the technical support team quickly identify, analyze, and resolve the issue.
   - Feature suggestion: Provide feedback on the feature suggestion page if you wish to add a new feature or optimize an existing feature based on your experience and business needs. Describe in detail the feature you wish to improve or add, and upload proper attachments. This information helps the technical support team understand the value and priority of the requirement implementation, and provides reference and direction for future feature development of the software.

3. Click "Submit". The technical support personnel analyzes the feedback upon receipt, and then takes appropriate actions.

Collecting error information

With the "Collect error info" function, you can package the current project files, including user project, operation logs, software version number, etc. Upload the packaged file when providing feedback on issues or suggestions. This feature helps technical support personnel efficiently analyze and resolve issues, and avoids missing details during manual information sorting, making the issue or suggestion feedback simpler.

1. Choose "Help -> Issue feedback -> Collect error info" from the menu bar.
2. In the opened "Collect error info" window, select whether to include the source project and error message output path.
   - Include source project: Check the box to include the project files in the package. If the project involves sensitive information, check this box with caution.
   - Error message output path: Select the path where the error message is saved.



3. Click "OK", and then the collected error information is packaged into a zip file and saved to the selected path, as shown in the following picture.



3.3.4 Technical Forum

"Inovance Planet" is a technical sharing and exchange platform provided by Inovance, aiming to offer users a space for learning and interactions. Here you can find diverse tutorials, case analyses, and technologies about Inovance's products, covering everything from beginner basics to advanced topics. You can also join discussions or create new posts to communicate with other users and experts, improving your technical skills.

The main functions of the platform are as follows:

- Knowledge sharing and learning

The platform offers a comprehensive resource library with diverse tutorials, case analyses, and technologies about Inovance's products, covering everything from beginner basics to advanced topics, all available for users to explore and download. It also offers comprehensive online training courses to help users better understand and use Inovance's products.

- Technical exchange and experience sharing

The platform has a discussion section where users can post questions or answers, engage in real-time communication and discussion on technical challenges, and share successful cases and technical research findings. For example, you can help other users find solutions more quickly by sharing practical experience in solving real-world problems; you can propose innovative ideas and perspectives during in-depth discussion on underlying product technology principles, cutting-edge technology applications, and other aspects, enhancing technical understanding and promoting innovation.

NOTE

Inovance Planet page can only be accessed with network connectivity.

Procedure

1. Choose "Help > Inovance Planet iFA Zone" from the menu bar, or click  in the upper right corner of the interface.
2. The browser opens the iFA page of "Inovance Planet", where you can view related introductions, guides, and courses for the iFA software. You can also access other sections of the forum as needed. See the actual interfaces for details.

You can log in to Inovance Planet in the following ways:

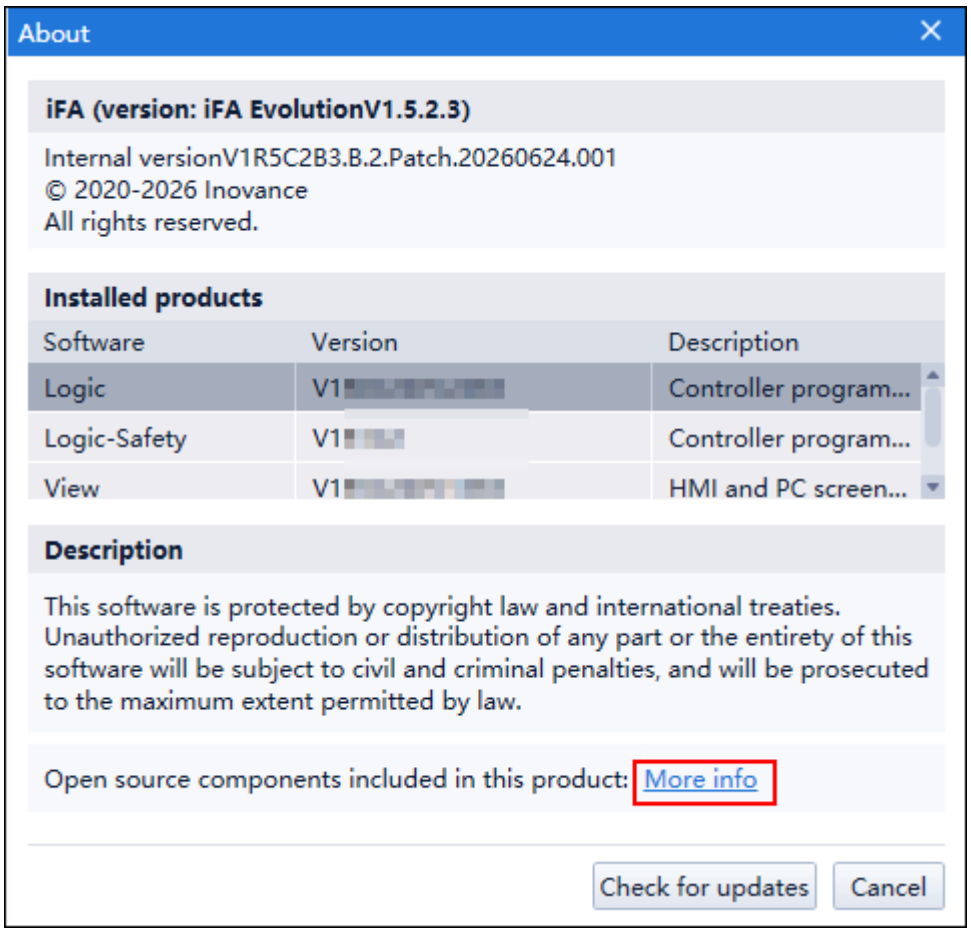
- Log in with the mobile number registered through the iFA software, which enables registration-free access to the Inovance Planet and course platform;
- Register and log in through the Inovance Planet website, which enables registration-free access to the course platform.

3.3.5 About iFA

On the "About iFA" page, you can view information such as the current software version, installed products, and open-source components. This feature makes it easy to compare with the latest official version, and supports online version updates, ensuring that users can continuously obtain the latest version.

Procedure

1. Choose "Help -> About iFA" from the menu bar.
2. In the opened interface, you can view the software version, installed products and their versions and descriptions, as well as open-source components.
3. Click the "More info" button in the opened interface, and the browser pops up the window displaying the open-source components included in the product, including component name and open-source licenses.



4. Click the "Check for updates" button to perform online update of the software version. For details, see 4.5 Upgrading the4.5 Software.

3.4 AI Assistant

iFA software is developed based on mature AI technology and introduces the AI Assistant to improve development efficiency and answer user questions. AI Assistant plays a significant role in multiple aspects, mainly reflected in helping improve programming efficiency and code quality, accelerating knowledge acquisition, lowering technical barriers, and optimizing production process.

AI Assistant installation and uninstallation

AI Assistant exists as a component and can be installed or uninstalled separately.

Install the AI Assistant

- Check the AI Assistant component during software installation and follow the installation procedure.
- If the AI Assistant is not checked during software installation, you can install it later as follows:
  1. Double-click the software installer (.exe file).
  2. Select "Add or remove components" and click "Next". The component selection tab pops up.
  3. Check the AI Assistant component and click "Install" to complete the installation.

Uninstall the AI Assistant

Proceed as follows to uninstall the AI Assistant:

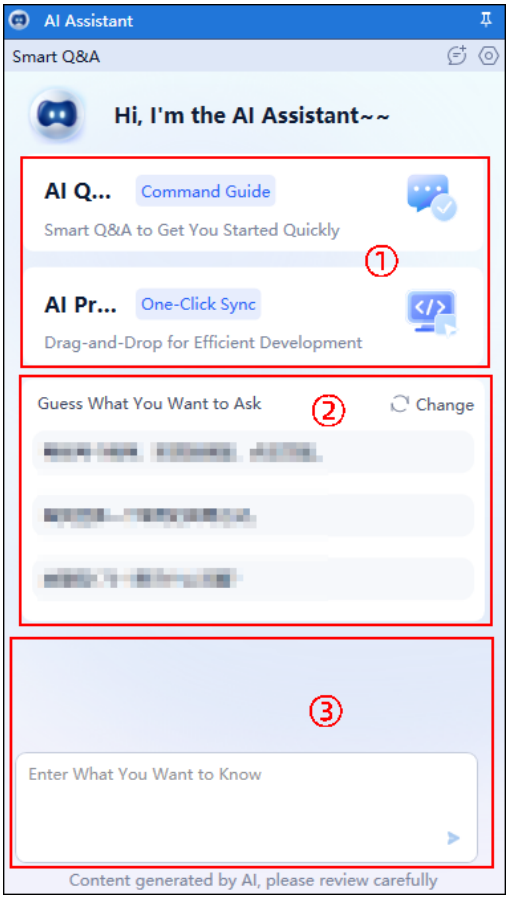
1. Double-click the software installer (.exe file).
2. Select "Add or remove components" and click "Next". The component selection tab pops up.
3. Uncheck the AI Assistant component and click "Install" to complete the uninstallation.

Use the AI Assistant

Procedure

1. Launch the iFA Evolution and open a project, and then click the "AI assistant" from the toolbar on the right side of the interface.
2. The AI Assistant interface mainly consists of three parts, as detailed below.
   - ① Function introduction area: Introduces the main functions of the AI Assistant, including AI Q&A and AI programming.
   - ② Quick input area: A series of commonly used input entries are provided. The AI Assistant quickly generates the proper answer when you click an entry, providing you with relevant information quickly.

     By default, three entries are displayed. You can switch entries by clicking Change.
   - ③ Q&A area: You can interact with the AI Assistant in natural language by asking questions in the input box of this area. The AI Assistant quickly and accurately parses your needs and output detailed and comprehensive answers. The AI Assistant provides Q&A interaction with extensive coverage, including but not limited to answering questions about Inovance's products and solutions; providing support for PLC programming, including code generation, code commenting, code explanation, code error detection, and instruction search; and offering technical consultations about IEC 61131-3 standards, Structured Text programming, and functional interface design.



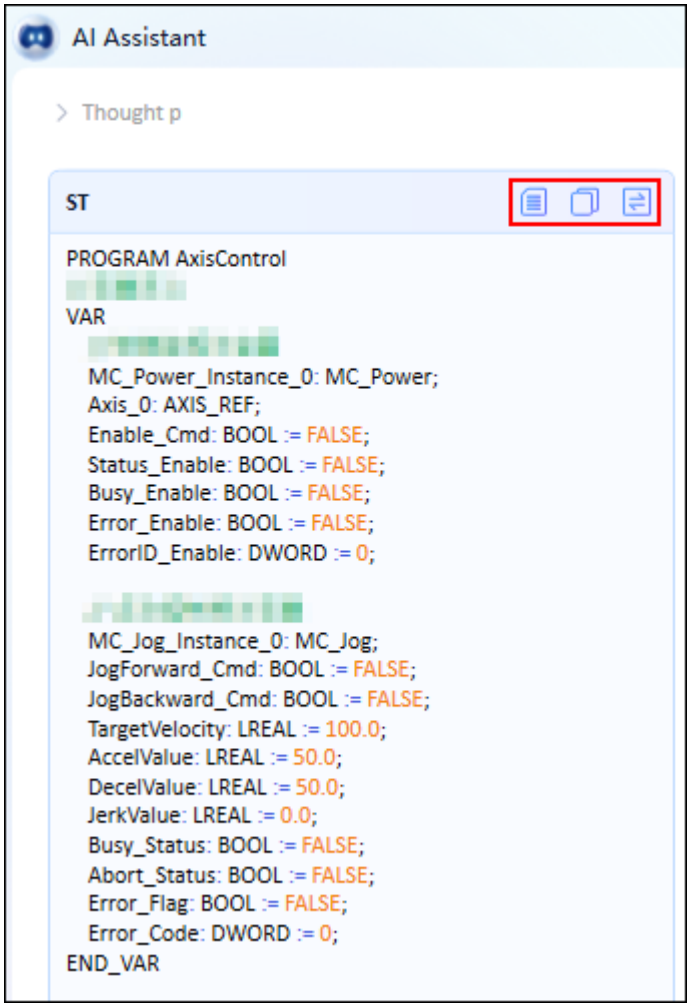
3. Multi-round interactions are supported. When you are dissatisfied with an answer, you can continue to ask follow-up questions.

> NOTE
>
> The input box supports up to 500 characters.

4. During the process of generating answers, you can click the Stop icon in the input box at any time to terminate the response.



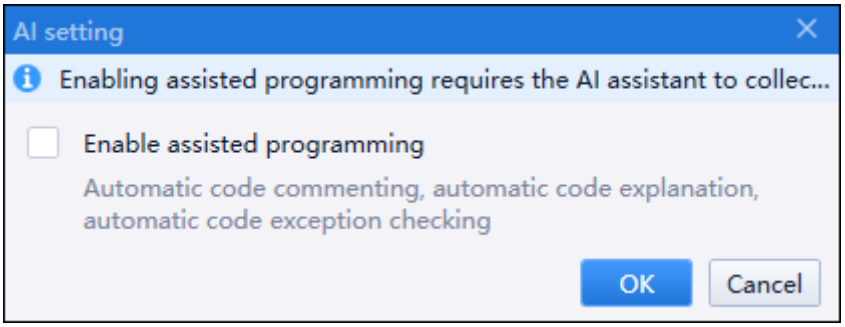
5. The generated answers can be selected as a whole or in part and then copied by right-clicking Copy icon or using the shortcut "Ctrl+C." The generated codes can be dragged, copied, and synchronized to relevant programming modules.



6. Click  in the top right corner of the interface to clear the conversation history and start a new conversation.

7. Click  in the top right corner of the interface. In the pop-up window, set whether to enable assistive programming, which is enabled by default.

   The AI-based assistive programming helps users write codes. Its main functions include automatic code commenting, automatic code explanation, and automatic code error detection, greatly improving coding efficiency and quality, making the coding process smoother and more efficient, and enhancing code stability and reliability.



Disclaimer of liability

The contents generated by the AI Assistant, including but not limited to generated codes, comments, and answers, are provided for reference only and do not constitute professional advice in any field. Information is generated based on training data and algorithms and may be inaccurate, incomplete, or outdated. For:

- High-risk business activities

- Scenarios that may affect personal safety

You should seek guidance from professionals. The platform neither guarantees the accuracy of the generated content nor assumes any responsibility for any consequences arising from use of or reliance on such information.

4 Installation and Uninstallation

4.1 Operating Environment Requirements

Basic hardware requirements

Item	Requirements
CPU	Intel Core i5 standard version 4xxx or later
RAM	More than 8 GB; 16 GB is recommended.
Hard disk	Available space > 100 GB
Resolution	Laptop: 1920*1080 or 22-inch monitor
Language	English

NOTE

When the screen resolution exceeds 1920*1080, it may result in text and interface elements appearing too small. You would better follow the system recommended default zoom ratio to ensure the best display effect. To adjust the display ratio, you can modify your computer system resolution through the "Scale and Layout". The change takes effect upon modification.

Basic software requirements

Operating System	Version
Windows 10	Enterprise edition (64-bit)
Windows 10	Professional edition (64-bit)
Windows 11	Enterprise edition (64-bit)
Windows 11	Professional edition (64-bit)

4.2 Installing the Software

The software package is automatically installed through the installer, which is started immediately after the installation medium is inserted into the drive.

Requirements

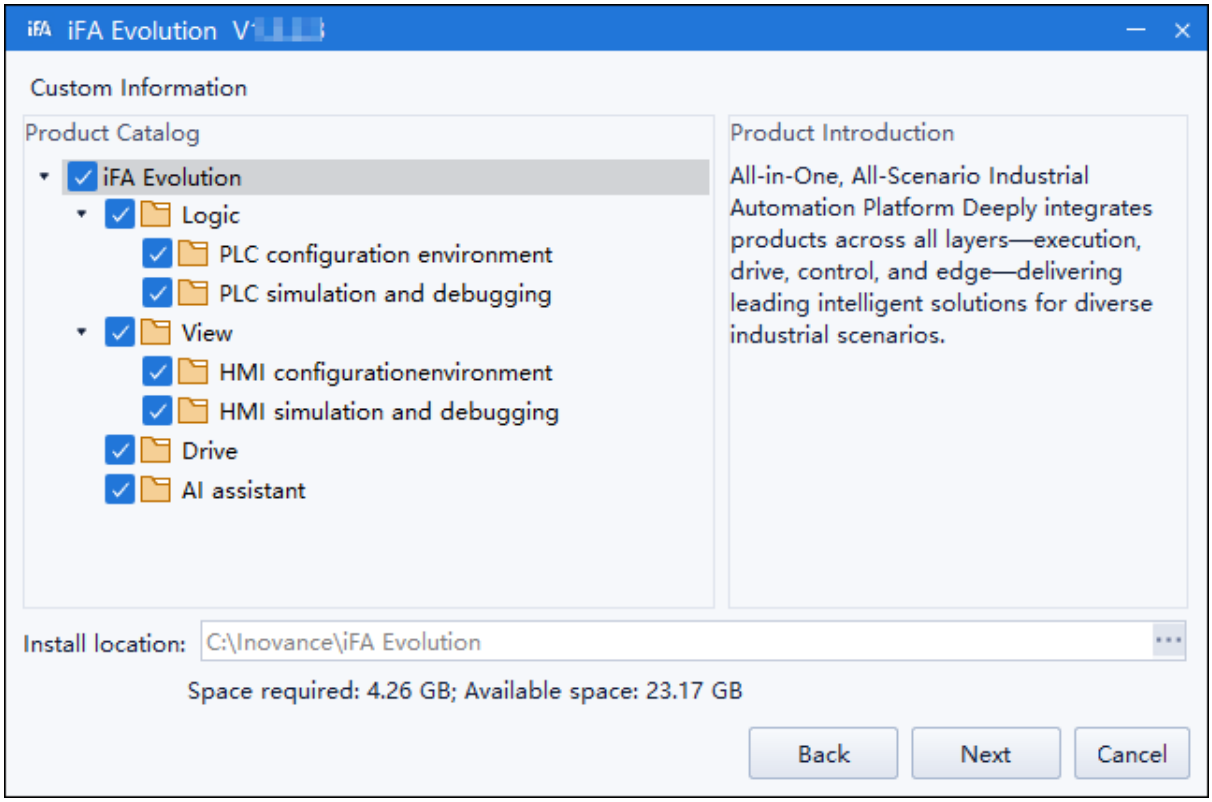
- The hardware and software meet the system requirements.
- The administrator privileges on the computer are obtained.
- All running programs have been closed.

Procedure

> **NOTE**
>
> When installing the software package for the first time, you will see a prompt on the interface indicating that licensing is required. For specific steps and instructions, see 4.4 Licensing the Software.

Proceed as follows to install the software package:

1. Double-click the installer to access the installation interface.
   You can select the language version for installation as appropriate, or modify the language in "Options -> General". For details, see language setting in 6.1 General Settings.
2. Click "Install" after selecting the language.
3. (Optional) If the iFA software already exists in your computer, running the installer will display the following options. Select the appropriate option and click "Next".
   - Add or remove components: Add or remove components based on the currently installed software, which means incremental installation.
   - Repair: Reinstall the software by directly overwriting it.
   - Uninstall: Uninstall all installed products.
4. (Optional) This step is required for a fresh installation. Carefully read the Software License Agreement, and then select "I accept the licenses" and click "Next".
5. Select the installation options and installation location.



   - Select proper components based on the project planning.
     - Logic: Integrated development software for controllers, including development and simulation components.
       - PLC configuration environment: Controller programming software, used for controller hardware configuration and software development, supporting languages such as LD, ST, CFC, and SFC.

        - PLC simulation commissioning: Supports code simulation for controllers (excluding safety controllers and redundant controllers).
    - View: Integrated development software for monitoring, supporting the development and simulation of operation screens and PC monitoring. Currently, the following two components are supported and cannot be installed separately.
        - HMI configuration environment: Monitoring development software, currently supporting HMI devices.
        - HMI simulation commissioning: Monitoring simulation software, which can be used to simulate HMI projects.
    - Drive: Drive commissioning software, supporting online commissioning of drive devices.
    - AI Assistant: The AI Assistant component can be installed separately. For details, see 3.4 AI Assistant.
  - Specify the installation location.

    The installation path cannot be changed in subsequent steps. Therefore, it is recommended to choose a hard disk with sufficient space.

6. Click "Install" to start the program installation.

7. After the program installation is complete, click "Run now" to open iFA. The iFA shortcut icon 

will appear on the desktop.

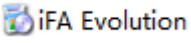
4.3 Uninstalling the Software

The software can be installed by:

- Using the control panel.
- Using the uninstall.exe.
- Using the software package.

Prerequisite: The software has been closed and is not running in the background.

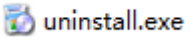
- **Procedure (using the control panel)**

1. Open "Control panel" and select "Programs and Features".
2. Right-click 

and select "Uninstall".
3. Click "Start", and then click "Yes" in the pop-up dialog box.
4. The uninstallation is complete.

   Confirm that the uninstallation is complete in the pop-up window. At this point, the Inovance folder in the installation path is cleared.

- **Procedure (using the uninstall.exe)**

Double-click 

in the installation path to start the uninstallation; the default path is C:\Inovance\iFA Evolution.

For subsequent steps, see the uninstallation procedure using the control panel.

- **Procedure (using the software package)**

  1. Double-click the installer (.exe file).
  2. Click "Run" to access the "Installation type" tab.
  3. Select "Uninstall" and click "Next". A confirmation tab for uninstallation appears.
  4. Click "OK" to complete the uninstallation.

Keep local settings

During software uninstallation, the system enables the "Keep local settings" option by default. When the current version is installed, the user data such as third-party reference libraries and device files will be retained. When the software is reinstalled, these retained data can be directly loaded and used without reconfiguration.

4.4 Licensing the Software

When installing the software package for the first time, you will see a prompt on the interface indicating that licensing is required. You can proceed with the installation only after licensing is complete.

Procedure

1. Double-click the installer to access the "iFA Software Licensing" interface.
2. "Apply for licensing" is selected by default. Take appropriate steps for the licensing application based on the actual conditions.
   - If your PC can access the Internet, directly click the "Apply for licensing" button in the lower right corner.
   - If your PC cannot access the Internet, click the "Copy link" button, copy the link to a PC that can access the Internet, and paste the link in the browser.
3. On the left side of the licensing application interface, carefully read the Software License Agreement and check "I accept the licenses"; on the right side, fill in the "Licensing Application Information" and then click "Submit."

> **NOTE**
>
> In the Licensing Application Information section, items marked with an asterisk are mandatory. The specific items are described as follows:
>
> - PC ID: Automatically identifies and populates the ID of the PC on which the licensing application is sent. It is not editable. When the link is sent to another PC for licensing application, the PC ID remains unchanged.
> - iFA Evolution version: Automatically identifies and populates the version of the software being installed. It is not editable.
> - Recipient email: Enter an email address to receive the license.
> - User information: Enter relevant information of the user, so that the approver can identify the applicant's identity based on this information.
> - Company information: Enter the user's company information. It is optional.

4. Inovance will process the application within 2 working days upon receipt. Once the application is approved, the license code will be sent to the provided email address.
5. After receiving the license code email, select "Install license" in the "iFA Software Licensing" interface, copy and paste the license code from the email into the input box, and click the "Install license" button in the lower right corner.
6. At this point, the interface redirects to the installation interface. See 4.2 Installing the Software to complete the installation.

> **NOTE**
>
> - The license code is valid for 14 days. You must complete the licensing within this period; otherwise, reapply for a new license code.
> - If the software is uninstalled and reinstalled within 30 days after the licensing and installation are complete, no additional licensing is required; if uninstalled and reinstalled after 30 days, a new license code must be requested; if not uninstalled or reinstalled, this restriction does not apply.

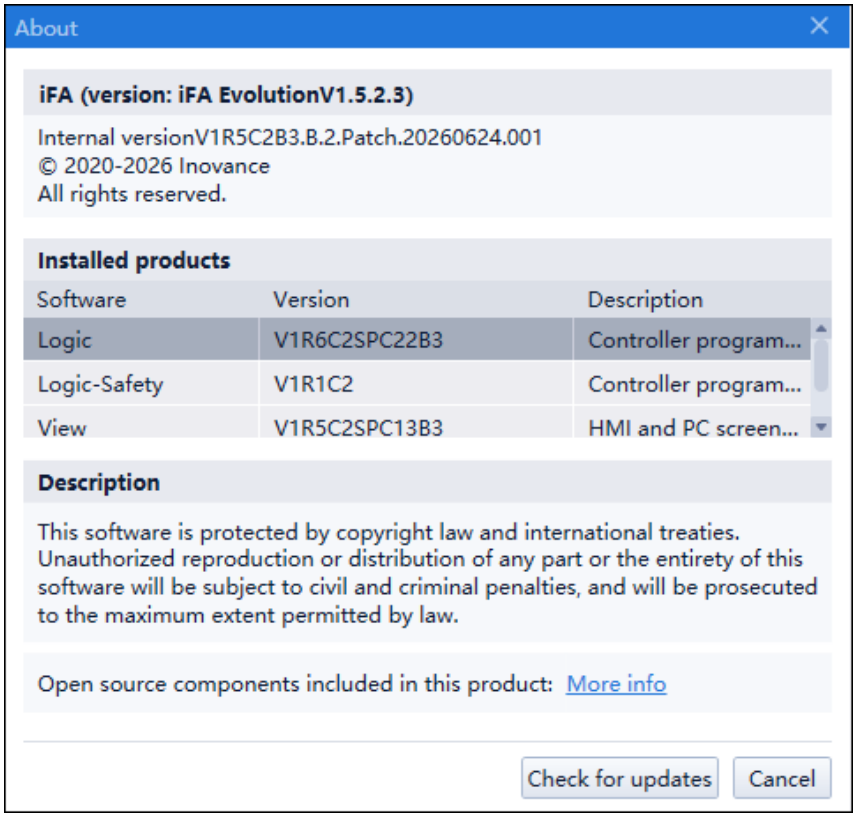
4.5 Upgrading the Software

iFA software supports both online and offline version updates.

Online update

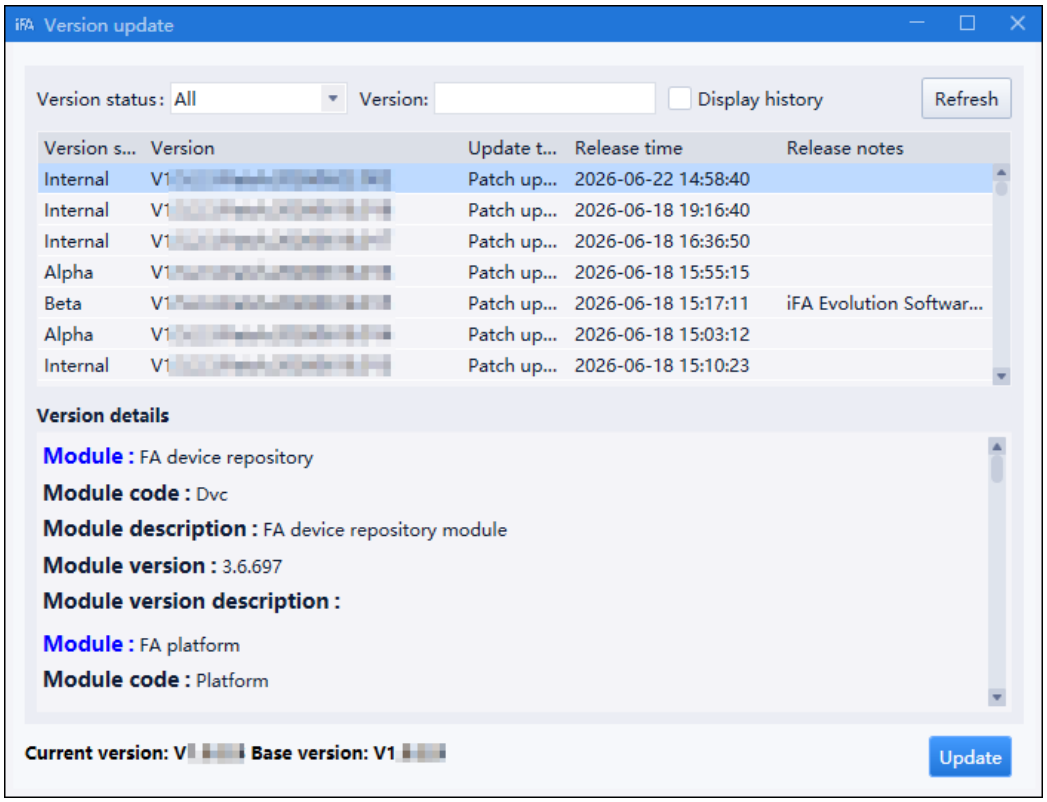
iFA software supports online version detection and upgrade. Follow the steps below:

1. Check that the device on which iFA software is installed has a stable Internet connection.
2. Open a project.
3. Open the "About iFA" window.
   Choose "Help -> About iFA" from the menu bar to open the window.



4. Select the upgrade package or the version to be updated.

   Click "Check for updates" to access the version update window.



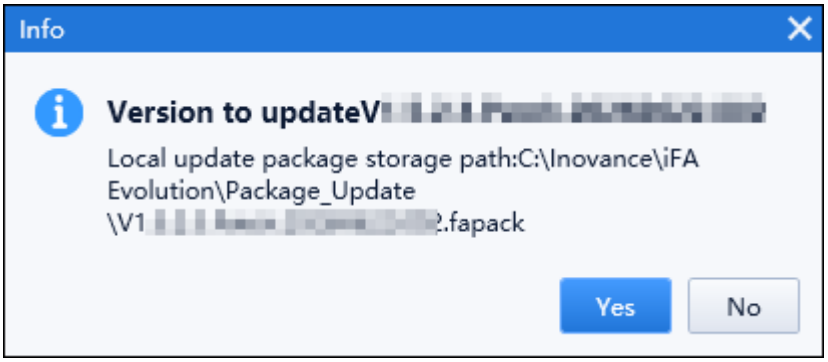
   When retrieving update packages from the server, you will see the latest software version or patches based on the current software version. You can select the target version from the result list for upgrade installation.

   Check Historical Versions to view details. This feature is supported from V1.4.3.5 and later.

   Version details display the patch package or version description.

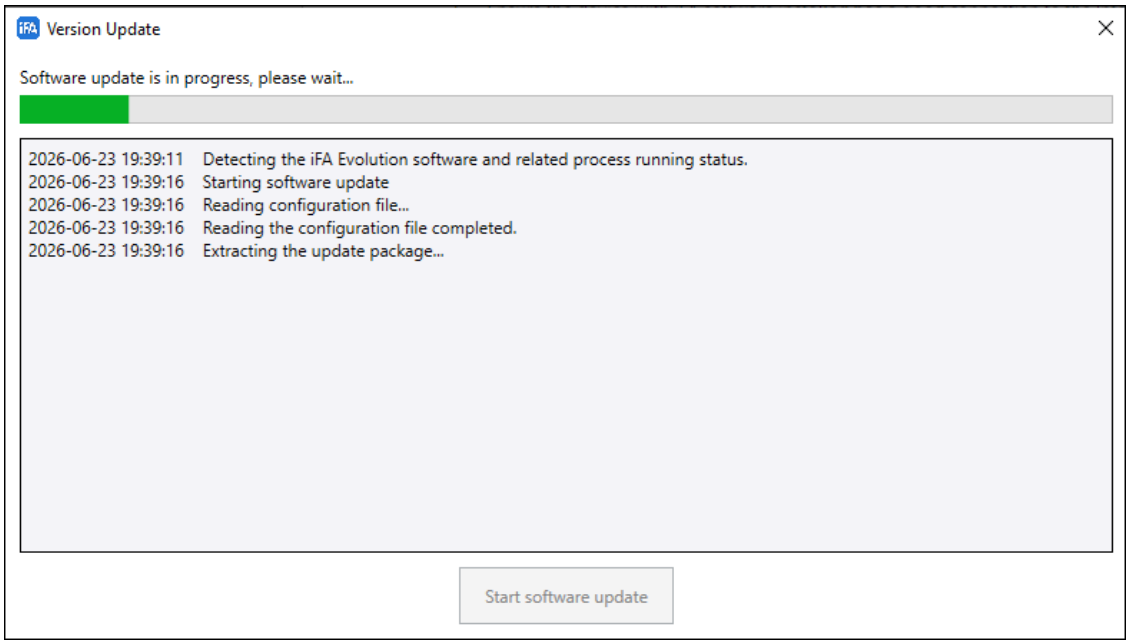
> **NOTE**
>
> If the current version is low, you must complete the update in the following order: Upgrade the software to the latest version, and then install the latest patches. To ensure system compatibility, it is essential to strictly follow the order of upgrading the software version and then installing the patches; otherwise, it may result in component dependency error.

5. Download and install the patch package.
   a. Click the "Update" button, and then click "OK" in the pop-up window to download the patch package.
   b. After the patch package is downloaded, click "Yes" in the pop-up window.



   c. Click "Start software update" in the pop-up dialog box.

      After the update is complete, the iFA software is started automatically.



The procedure for software version upgrade is similar to that for patch installation.

Offline update

iFA software also supports offline version update.

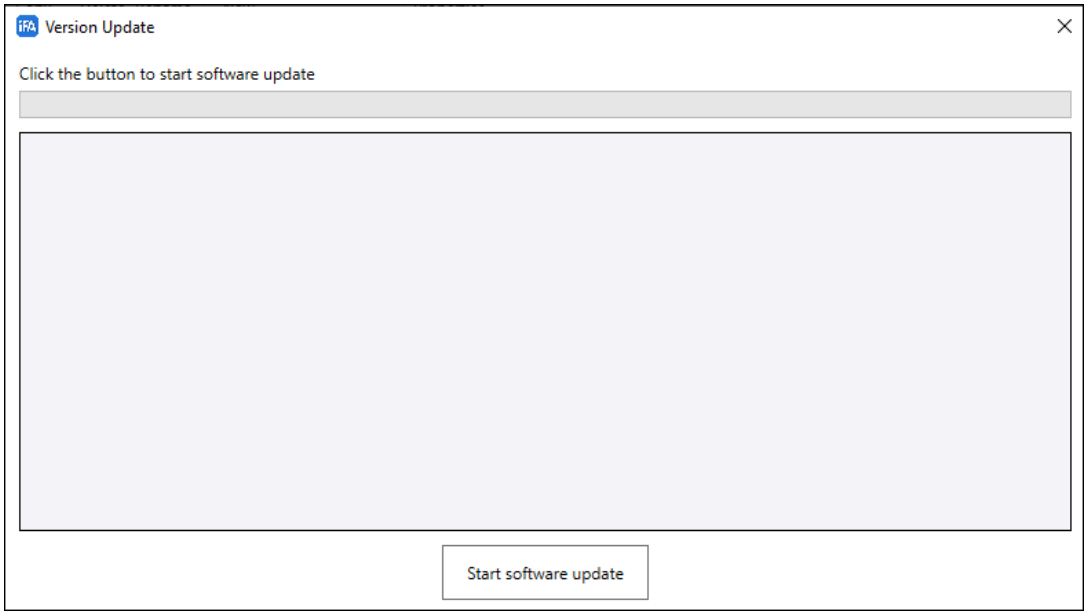
Prerequisites

- A PLC with Internet access is available.
- A USB flash disk is available for file copying.
- Check that the iFA software version matches the patch package version for offline update.

- The iFA software has been started or opened once.

Procedure

1. Perform an "Online Patch Upgrade" on a device with Internet access. After the upgrade is complete, a "Package_Update" folder is generated in the installation directory xx\iFA Evolution.
2. Compress the generated folder using a compression tool to create a .zip file as the offline patch package.
3. Copy the .zip file with USB flash disk, place it in the iFA software installation directory xx\iFA Evolution on the device without Internet access, and then decompress it.
4. Open the decompressed "Package_Update" folder and verify the file information (with the .fapack extension) and version information.
5. Double-click to open the xx.fapack file or right-click to open it using the xx\Bin\FA_Update.exe in the installation directory.
6. Click "Start software update" to perform the patch upgrade.



CAUTION

To apply offline software patches, ensure that the base versions are identical; otherwise, the upgrade will fail. For example, software with a base version of V1.4.4.5 cannot be upgraded using a patch package intended for V1.5.2.1.

5 Software Interface Layout

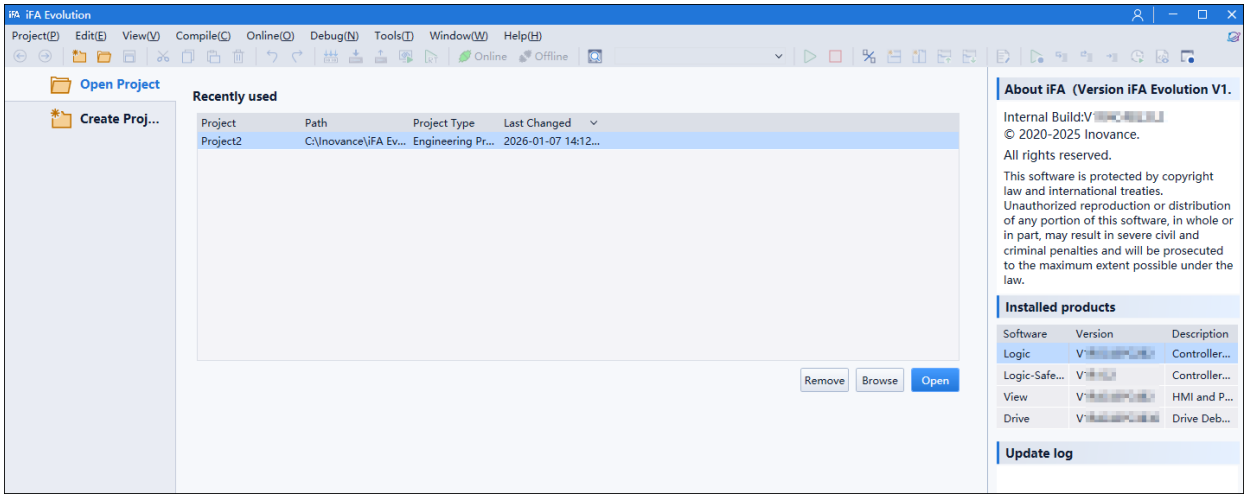
5.1 Opening and Exiting the Software

Open the iFA

The iFA can be opened by:

- Choosing "Inovance > iFA Evolution" from the Windows Start menu;
- Double-clicking the "iFA Evolution" icon on the desktop;
- Double-clicking the FASLN file in the project storage path (Default path: C:\Users\Administrator\Documents\Inovance\iFA\Projects\Project Name).

After opening the software, you can access the project interface, which provides an entrance for creating or opening projects and displays recent projects in a list.



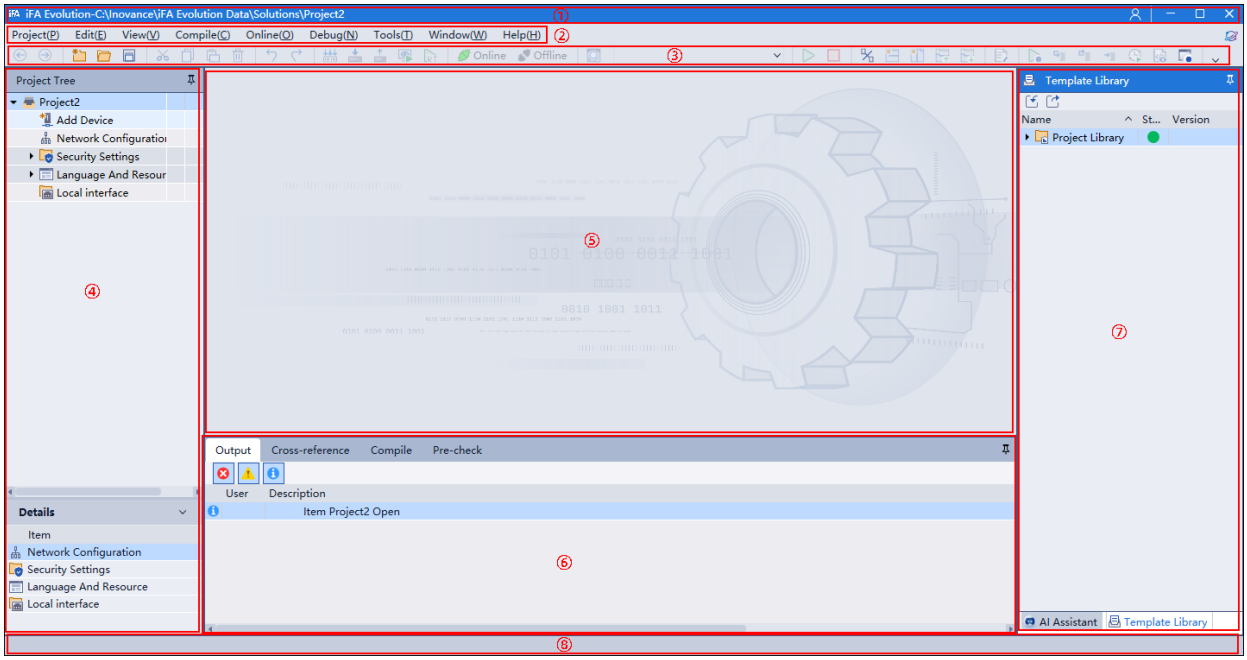
Exit the iFA

Click "Exit" in the "Project" menu. If the project contains unsaved changes, the system will prompt you to save them:

- Select "Yes" to save the changes in the current project and then close the iFA.
- Select "No" to close the iFA without saving the recent changes in the project.
- Select "Cancel" to cancel the closing process (the software will remain open).

5.2 Project View

The project view of the iFA Evolution consists of title bar, menu bar, toolbar, project tree, workspace, monitoring area, toolbox, and status bar, as shown in the following figure.



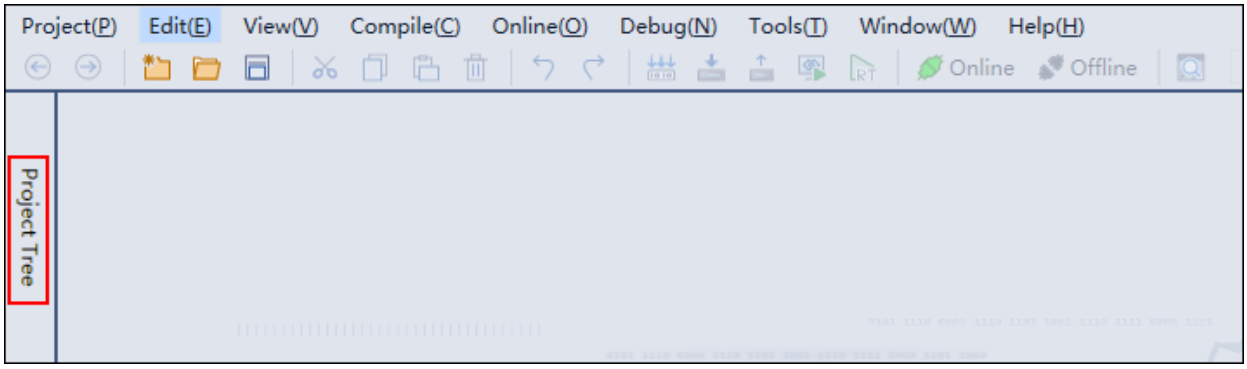
No.	Name	Description
①	Title bar	Located at the top of the main interface, and includes the following functions: • Displays the main project path. • Controls the main window maximizing/minimizing and closing, as well as account login/logout.
②	Menu bar	Collection of buttons grouped by program function; located in a horizontal bar just below the title bar. Provides an entrance to commonly used functions, including primary menus such as Project, Edit, View, Compile, Online, Debug, Tools, Window, and Help, each containing multiple submenus.
③	5.3 Toolbar	When the main interface opens, the toolbar is pinned below the menu bar, offering commonly used editing tools.
④	5.4 Project Tree	Provides an entrance for displaying and operating all project data, and displays projects in tree and table formats. The following tasks can be performed in the project tree. • Add a new node. • Edit an existing node. • Scan and modify the properties of an existing node.
⑤	5.5 Workspace	Displays the opened objects, such as tables, editors, and views.
⑥	5.6 Monitoring Area	Located at the bottom of the interface, and displays information about Output, Cross-reference, Compile, and Pre-check.
⑦	5.7 Toolbox	Provides related tool tabs for performing additional operations based on the edited or selected object. The tool tabs can be collapsed or reopened at any time.

No.	Name	Description
⑧	Status bar	Located at the bottom of the main interface, and includes the following functions: • Left: Displays the path. The path consists of an icon and path text and is displayed when the edit window is activated. • Right: Displays the prompt. The prompt consists of status prompt text and an indicator. For example: When a PLC is online, it displays the current number of logged-in users, CPU utilization, and memory utilization.

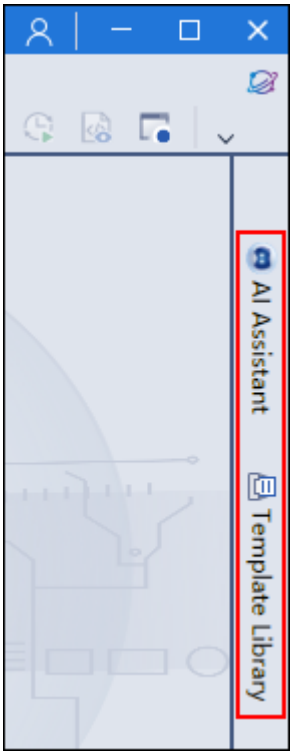
Hide an area

You can hide the project tree, toolbox, and monitoring area.

- Hide: Click in the title bar of an area.
- Unhide
  - Project tree: When the project tree is hidden, "Project Tree" is displayed on the sidebar, as shown in the following figure. Click "Project Tree" to display it, and click again to pin the area.



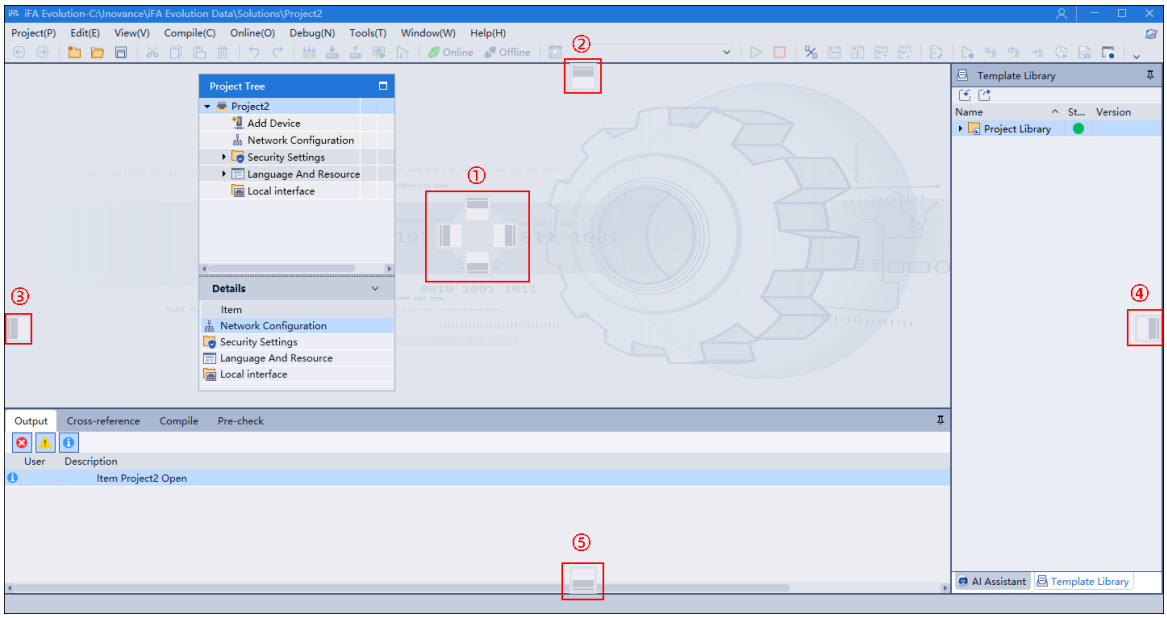
  - Monitoring area: When the monitoring area is hidden, the pushpin icon in the title bar changes to . Click the icon to unhide the area.
  - Toolbox: When the toolbox is hidden, the tool tabs are displayed on the sidebar, as shown in the following figure. Click any tab to display the content, and click again to pin the area.



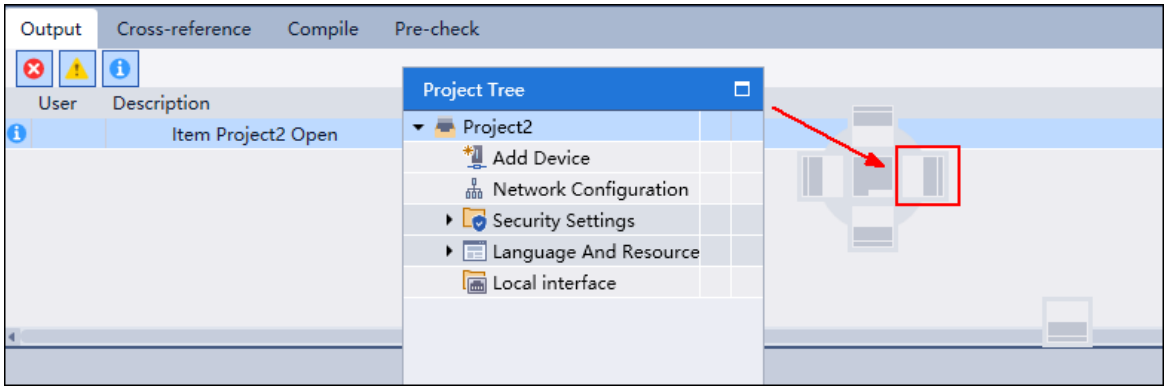
Float an area

You can float the project tree or toolbox window and dock it back.

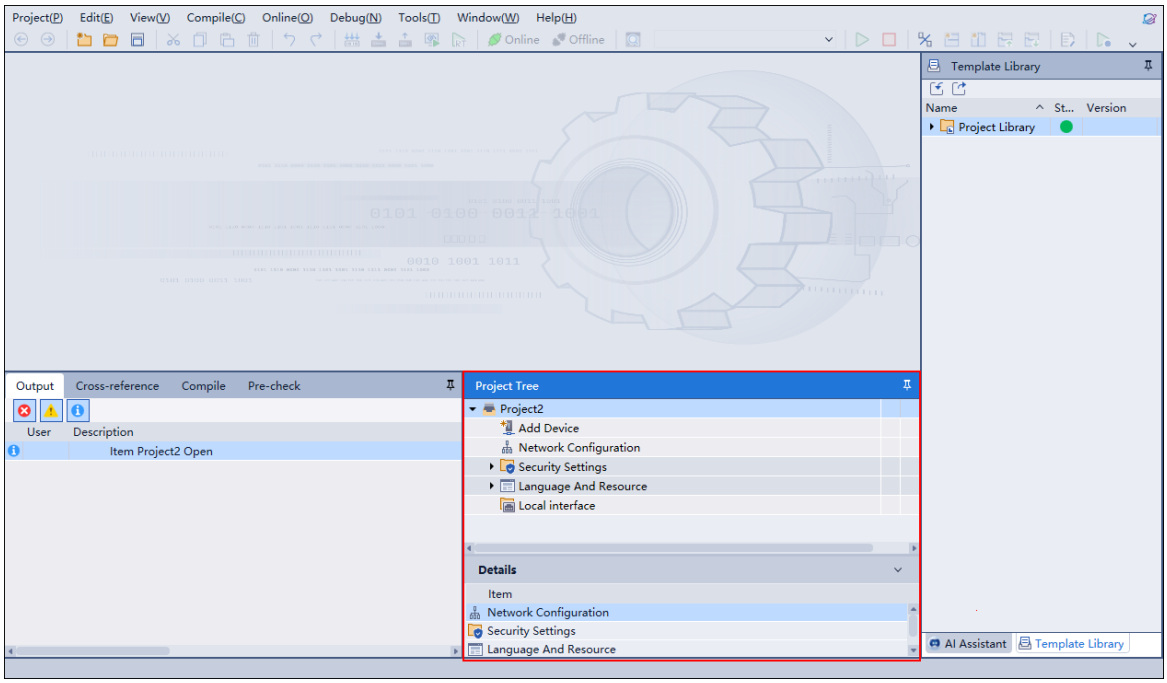
- Float
  Hover over the title bar of the project tree or toolbox, and then drag the window to any desired position.
- Dock
  - Method 1: Hover over the title bar of the floating window, and then drag the window until the docking buttons appear on the interface, as shown in the following figure. Drag the floating box to the target position indicated by the docking button.
    ① indicates docking into a specific direction within the area.
    ②③④⑤ indicate docking into the top, left, right, and bottom sides of the project view, respectively.



Taking the floating project tree window as an example, drag the window to the monitoring area; after the docking buttons appear, drag the window to the right side of the monitoring area, as shown in the following figure.



The docking effect is as follows:



- Method 2: In the "Window" menu, select "Reset window layout" to dock the floating window back to its previous position.

> **NOTE**
>
> The Reset Window Layout function does not apply to docked windows, which means once a floating window is docked, Clicking "Reset window layout" cannot restore the window to its previous position.

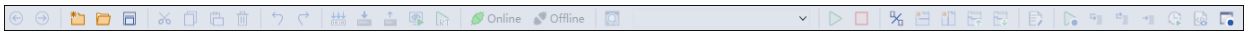
Maximize an area

You can maximize a floating window.

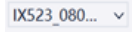
- Maximize: Click in the upper-right corner of the floating window to maximize the window.
- Restore: Click in the upper right corner of the maximized window to restore the window.

5.3 Toolbar

When the main interface opens, the toolbar is pinned below the menu bar, offering commonly used editing tools.



The toolbar categories and icons are described in the following table.

Category	Icon	Name	Note
Navigate		Navigate backward	Move the mouse focus back to the previous position in the workspace.
		Navigate forward	Move the mouse focus to the next position in the workspace.
Project		New	Create and open a new project by entering the required information.
		Open	Select and open the target project from existing local projects.
		Save	Save the active project data to its original path.
Edit		Cut	Cut the selected object in the project.
		Copy	Copy the selected object in the project.
		Paste	Paste the copied object to the selected position.
		Delete	Delete the selected object in the project.
Undo/ Redo		Undo	Revert the project configuration data to the previous step.
		Redo	Advance the project configuration data to the next step.
Compile		Compile	Compile the project to generate compiled data or executable code.
Upload/ Download		Download to device	Download the project, hardware configuration, or program to a device in the network.
		Upload from device	Upload the entire project, including control-level devices, slaves, and network topology relationships.
		Start/Stop simulation	See 10.13 PLC Simulation Mode.
		Start runtime system on PLC	See 10.13 PLC Simulation Mode.
Online/ Offline		Online	Log in to a control-level device to directly control and debug the device.
		Offline	Log out to end the device control and debugging.
Device		Device connection	Display the device connection history.
		Device scanning	Scan for all compatible devices in the same network. You can select the target device for connection as desired.
Start/Stop		Start	See 10.10.1 PLC Start/Stop.
		Stop	See 10.10.1 PLC Start/Stop.
Window		New horizontal group	Create a horizontal group for the active tab and other tabs.

Category	Icon	Name	Note
		New vertical group	Create a vertical group for the active tab and other tabs.
		Move to previous group	Move the active tab to the previous group.
		Move to next group	Move the active tab to the next group.
		Cross-reference	Open the cross-reference view in the monitoring area.
Debug		Write value	Write a variable value while running.
		Step over	Step over the next statement.
		Step into	Step into the next statement.
		Step out of	Step out of a stored procedure/function. If there is a breakpoint between the stop statement and the end of the procedure/function, execution will stop at the breakpoint; if there is no breakpoint, execution will step out of the call hierarchy.
		Run to cursor	The program runs to the location of the cursor. If there is a breakpoint between the stop statement and the cursor, the program will stop at the breakpoint.
		Execute one scan cycle	The program runs for one cycle. If the pointer does not stop at the beginning of the program, the program will run from the current position to the end. If a breakpoint exists between the stop pointer and the end position, it will be triggered.
		Set next statement	Move the cursor to the next statement.
		Show current statement	Move to the statement where the pointer stops.
		Breakpoint window	See 10.10.4 Breakpoint Debugging.

5.4 Project Tree

Feature

The project tree serves as the entrance for displaying and operating all project data, and displays projects in tree and table formats. The following tasks can be performed in the project tree.

- Add a new node.
- Edit an existing node.
- Scan and modify the properties of an existing node.

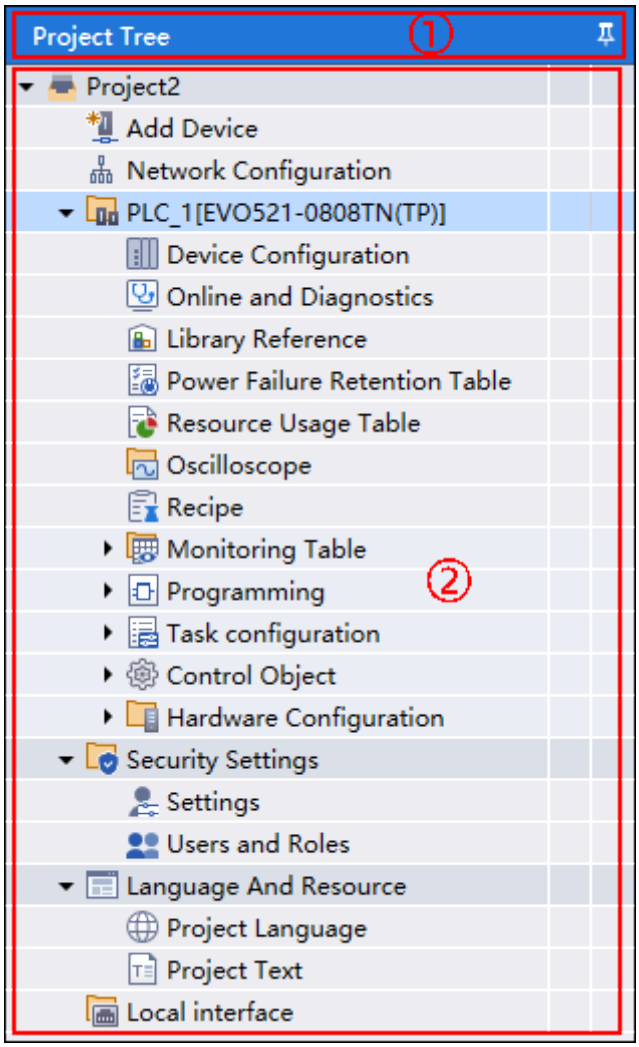
Layout

The project tree consists of a main project and a details view, which are arranged sequentially from top to bottom.

- Main project: Entrance for adding, editing, and previewing a node.
- Details view: Displays detailed information of the currently selected node in the main project and reference project in a list.

Main project

The main project consists of the window title bar and project elements (including Add Device, Device and Network Configuration, Devices/Folders, Safety Setting nodes), as shown in the figure below.



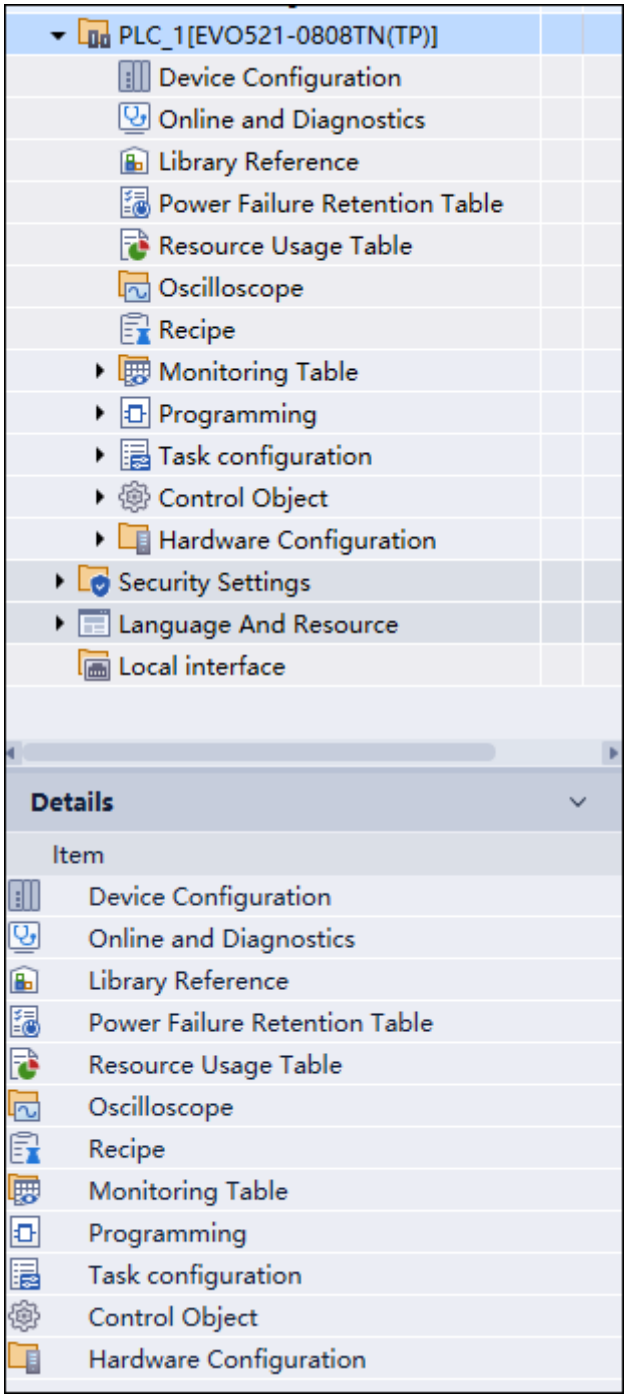
No.	Name	Description
①	Title bar	Title bar of the project tree, which can be hidden using the pushpin button.

No.	Name	Description
②	Project	A project consists of a three-column tree table. • First column: ◦ Project root node: expanded by default, and displays project nodes in a tree structure, which can be manually expanded or collapsed. ◦ Devices/Folders: child nodes of the project; displays devices/folders and their child nodes in a tree structure. ◦ Safety setting: Allows viewing and setting project permissions and user roles. • Second column: Node (online) status indicator ◦  Appears next to the project tree root node when any node in the project tree has an abnormal status. ◦  Applies to child nodes for programming; indicates that a program is not configured in the task. ◦  Applies to hardware device nodes; indicates that a device is operating properly. ◦  Applies to hardware device nodes; indicates that a device is disabled. ◦  Applies to hardware device sub-nodes; indicates that a device malfunctions. Double-click the icon to access the "Online and diagnostics" page for fault information. ◦  Applies to hardware device parent nodes; appears next to the parent node when any of devices on its child node malfunctions. • Column 3: Project comparison status indicator Applies to the project comparison results in the online/offline state. See 10.12 PLC Project Comparison.

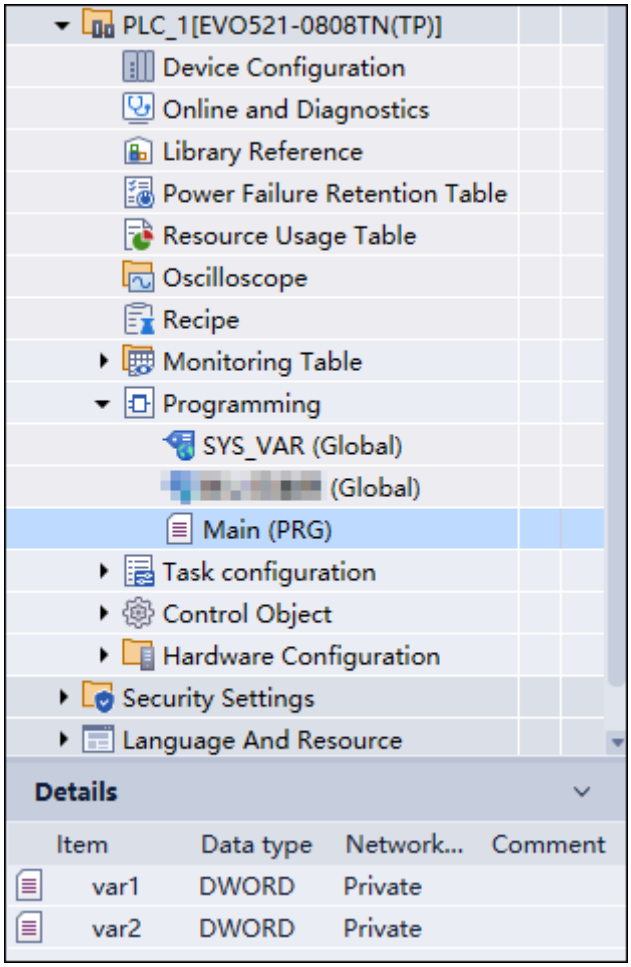
Details view

- Displays detailed information of the currently selected node in the main project and reference project in a list;
- Supports the same double-click behavior for both main and reference projects;
- Only supports the copy operation for dynamically created type nodes.

When a parent node is selected, all child nodes at the next level are displayed. For example, the details view is shown as follows when a PLC device node is selected:



When a child node is selected, the detailed information is displayed based on the added product. For example, the following details view displays the variable table for the selected Main node:



When you edit the details of a node in the workspace, the details view will be refreshed simultaneously.

5.5 Workspace

5.5.1 Workspace Feature and Layout

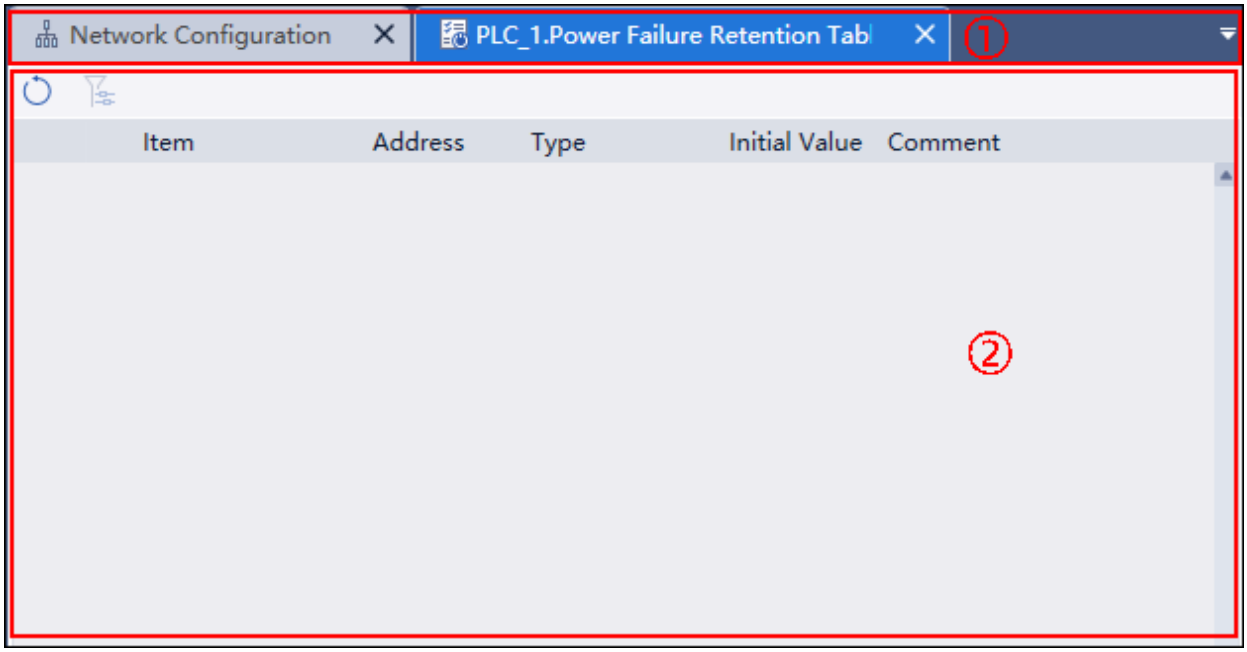
Feature

The workspace is the primary area for users to perform core editing operations. All opened editable objects (such as editors and tables) are loaded into the workspace as separate windows for visual editing.

> NOTE
>
> Up to 64 tabs can be created.

Layout

The workspace is located in the center of the main interface and its size can be adjusted by dragging the splitter to the left, right, or bottom. The default background image is displayed when no editing window is added. The workspace consists of a tab bar at the top and an edit view below. The edit view is filled by the operator. See the following interface screenshot for reference.



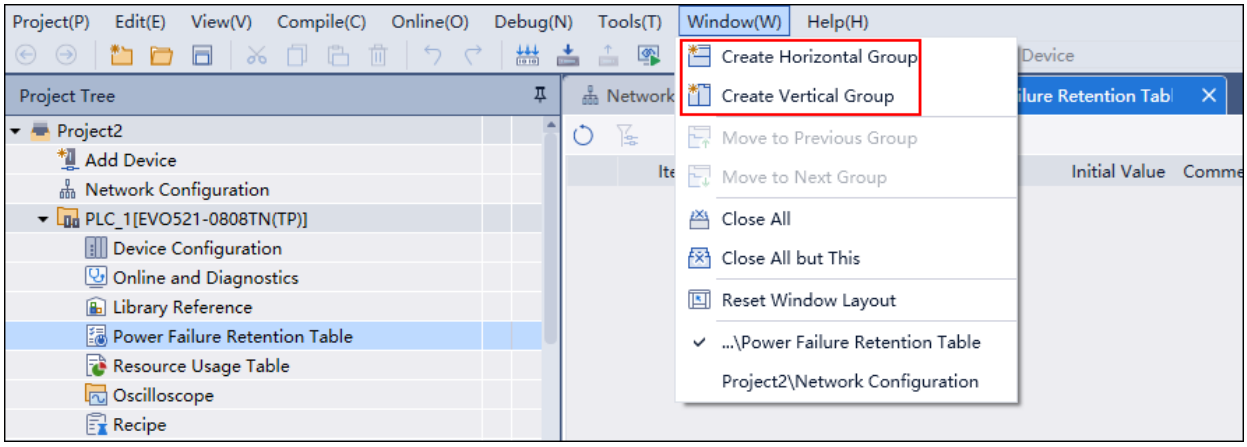
Area	Name	Description
①	Tab bar	Each tab bar contains a tab name and a close icon. Tabs can be moved by dragging with the mouse. The shortcut menu for a single tab includes Create Horizontal Group, Create Vertical Group, Move to Previous Group, Move to Next Group, Close, Close All Except This, Close All, Invert Object Selection, and Invert Parent Object Selection.
②	Edit view	The edit view is filled by the operator.

5.5.2 Splitting the Workspace

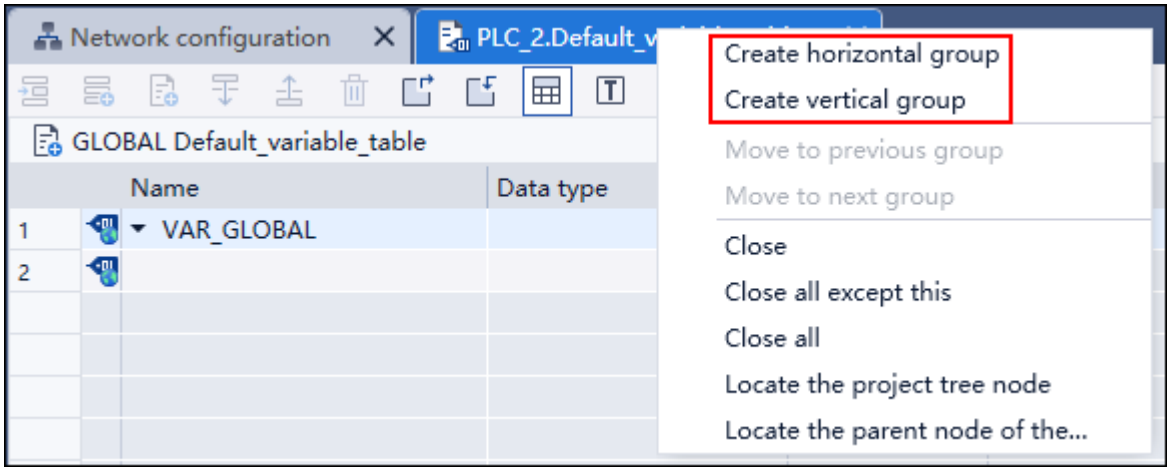
You can split the workspace vertically or horizontally to display the selected elements side-by-side or stacked with other elements in the editor bar.

Perform one of the following steps to split the workspace vertically or horizontally:

- In the Window menu, select "Create Horizontal Group" or "Create Vertical Group".



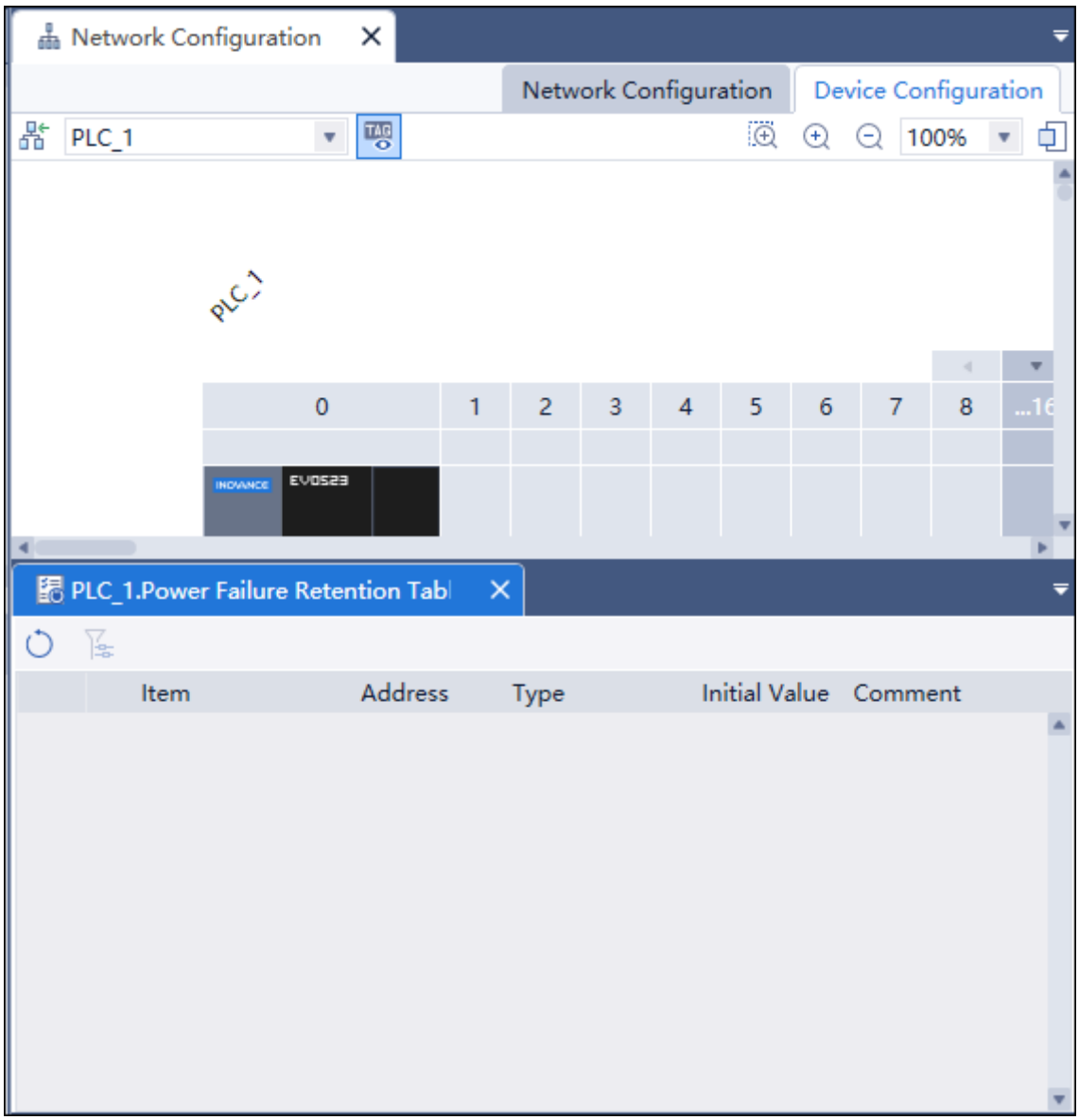
- Select the title bar of the workspace, and then select "Create Horizontal Group" or "Create Vertical Group" from the shortcut menu.



NOTE

When less than two elements are included in the workspace, the "Create Horizontal Group" and "Create Vertical Group" options cannot be selected.

Example: Splitting the workspace horizontally.



5.5.3 Moving to Previous/Next Group

You can flexibly move elements in the workspace to the previous or next split window by using Move to Previous/Next Group, so as to display similar or related elements as a group in the same split window.

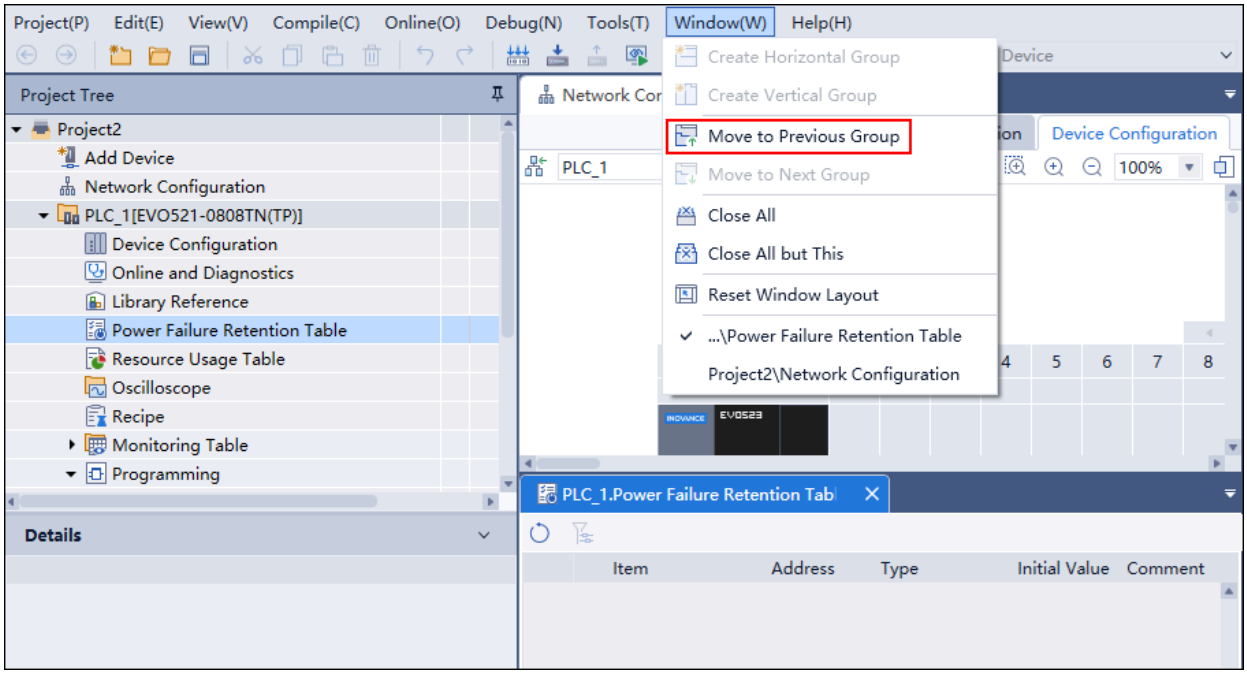
Prerequisites

- The workspace has been split vertically or horizontally.

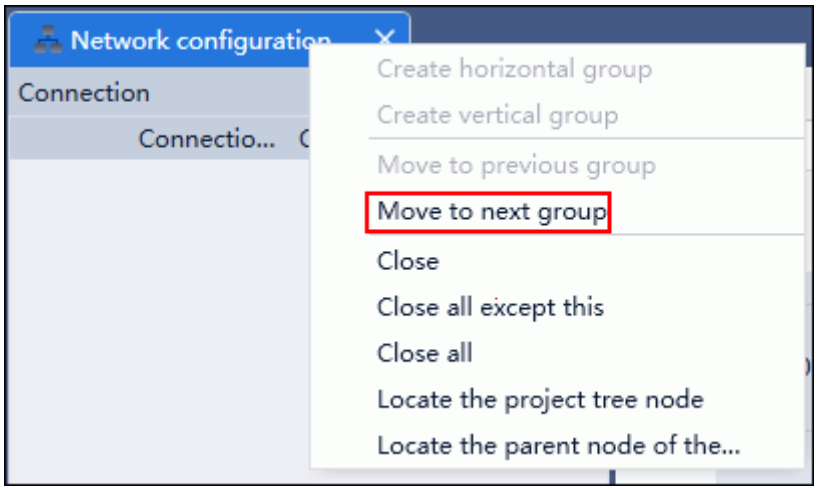
Procedure

Follow one of the following steps to move to the previous or next group:

- In the "Window" menu, select "Move to Previous Group" or "Move to Next Group".



- Select the title bar of the workspace, and then select "Move to Previous Group" or "Move to Next Group" from the shortcut menu.



5.5.4 Floating an Element

You can float elements in the workspace into separate windows and then dock them back at any time.

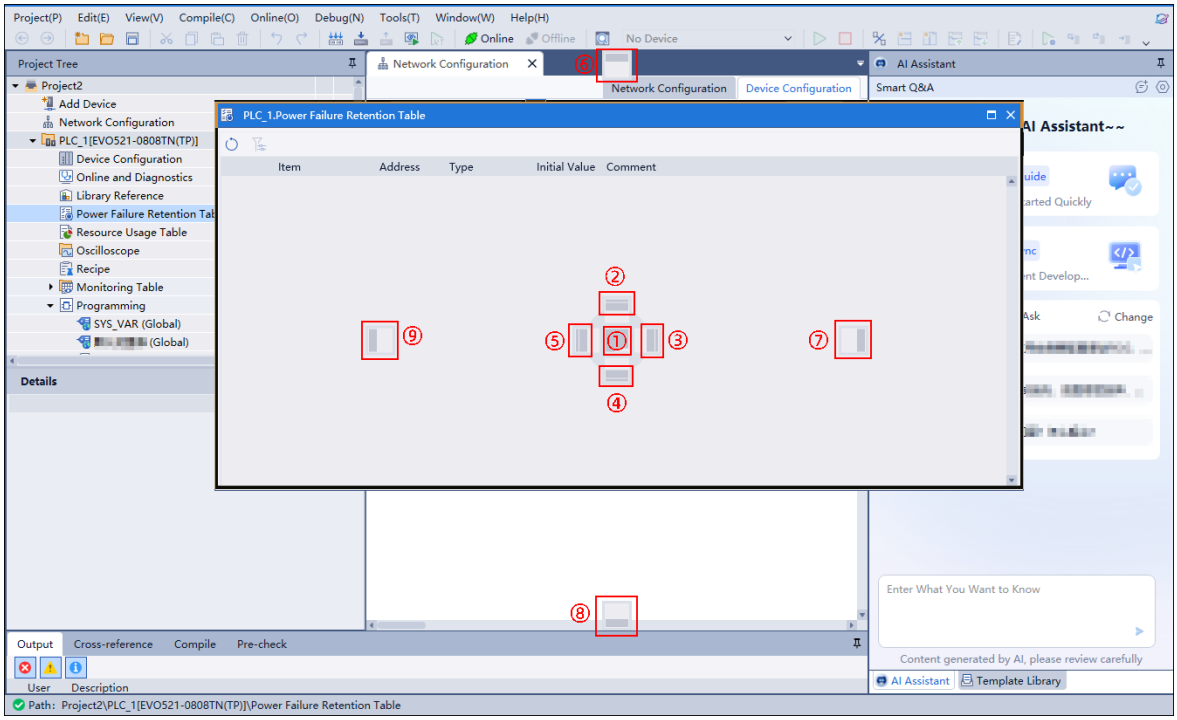
Float an element

Hover over the title of a workspace element and drag the element.

The element will move out of the workspace and appear in a separate window. At this point, you can put the window at any desired position.

Dock a floating element

- Method 1:
  1. Hover over the title bar of a floating element and drag the element until the docking buttons appear on the interface, as shown in the following figure.
     ① indicates docking into the current window in the workspace.
     ②③④⑤ indicate docking into the top, right, bottom, and left sides of the current window in the workspace, respectively.
     Floating elements in the workspace can be docked back to the workspace only. Therefore, the outer docking buttons ⑥⑦⑧⑨ have the same functions as ②③④⑤, indicating docking into the top, right, bottom, and left sides of the current window in the workspace, respectively.



  2. Drag the floating box to the target position indicated by the docking button.

- Method 2:

  In the "Window" menu, select "Reset window layout" to dock the floating window back to its previous position.

> **NOTE**
>
> The Reset Window Layout function does not apply to docked windows, which means once a floating window is docked, Clicking "Reset window layout" cannot restore the window to its previous position.

5.5.5 Maximizing an Element

You can maximize an element in the workspace.

Maximize an element

1. Hover over the title of a workspace element, and then drag the element to float it.
2. Click in the upper right corner of the floating window.

Restore the maximized element

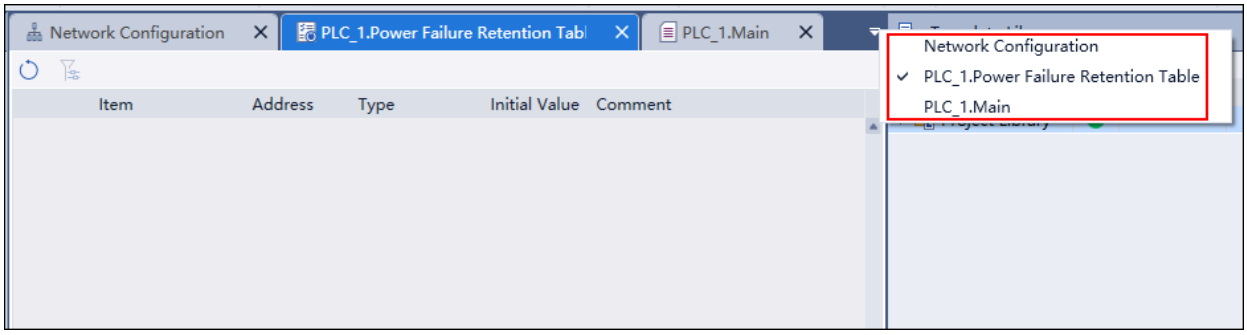
Click in the upper right corner of the maximized window to restore the window.

5.5.6 Switching Between Elements

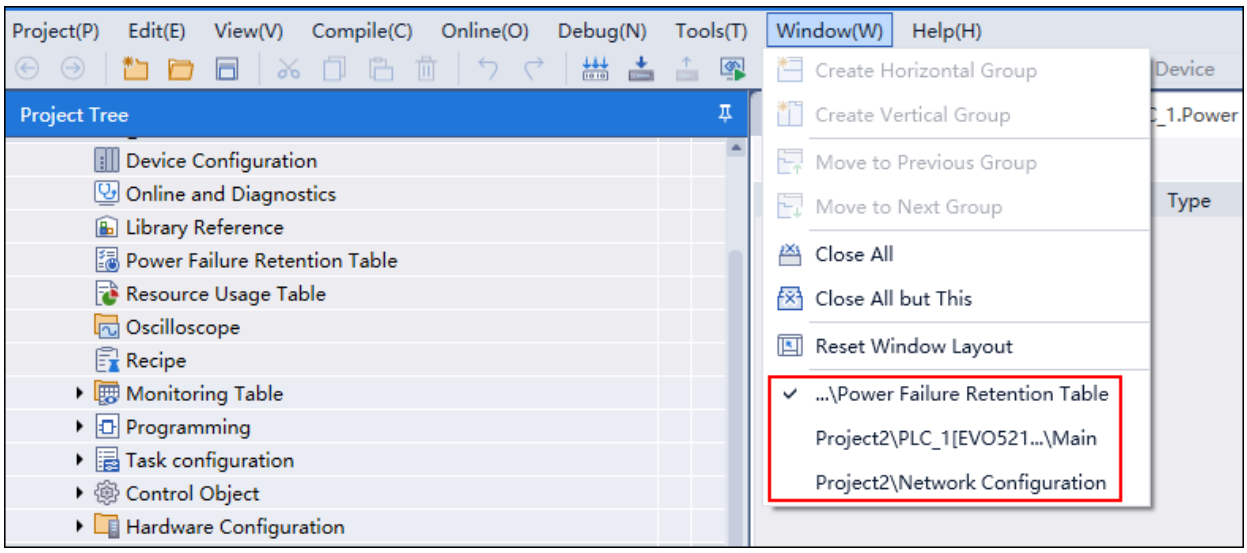
You can switch between elements in the workspace at any time.

Perform either of the following steps to switch between elements in the workspace:

- Click on the right side of the title bar in the workspace, and then select the element name from the drop-down list.



- In the "Window" menu, select the element name.



5.5.7 Closing an Element

Close the current element

Perform either of the following steps to close the current element in the workspace:

- Click in the title bar of the target element in the workspace.
- Select the title bar and then select "Close" from the shortcut menu.

Close all except this

Perform either of the following steps to close all elements in the workspace except the selected one:

- In the "Window" menu, select "Close All Except This".
- Select the title bar of the workspace, and then select "Close all except this" from the shortcut menu.

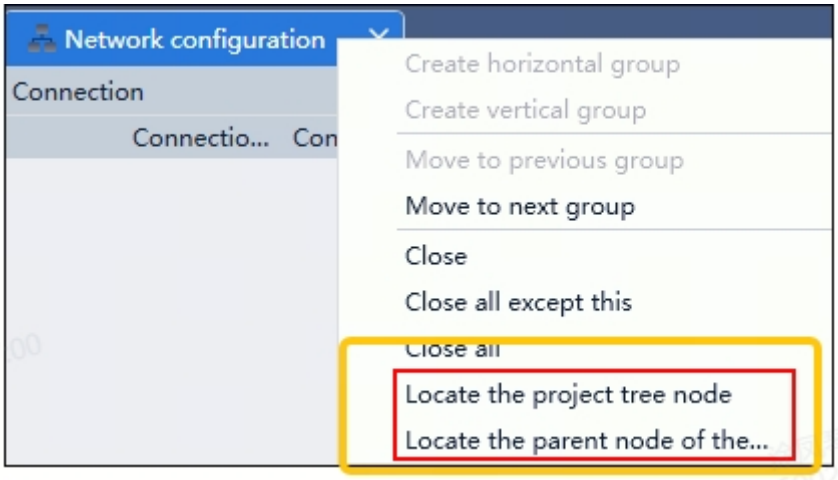
Close all

Perform either of the following steps to close all elements in the workspace:

- In the "Window" menu, select "Close all".
- Select the title bar of the workspace, and then select "Close all" from the shortcut menu.

5.5.8 Inverting Object Selection

Select the target window in the workspace, right-click the title bar to open the context menu, and click "Locate the project tree node" or "Locate the parent node of the project tree" to locate the node or parent node to which the current window belongs in the project tree.



5.6 Monitoring Area

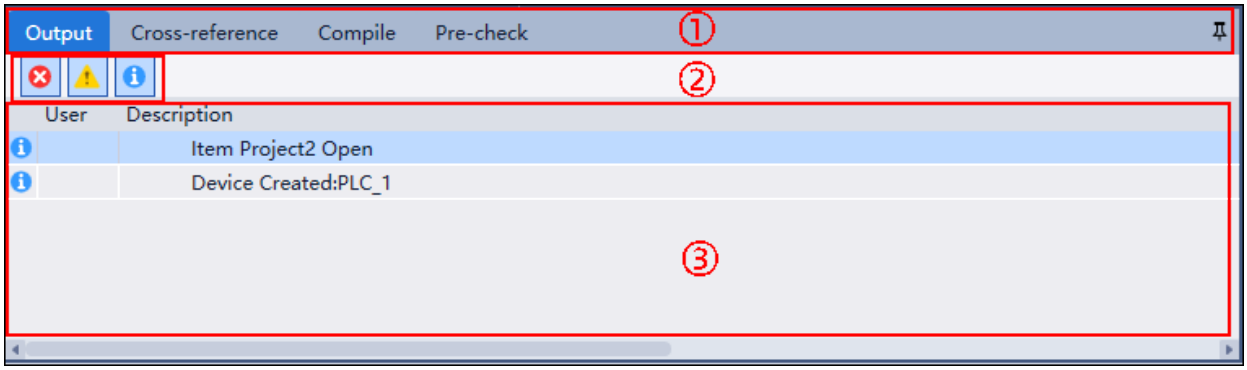
5.6.1 Monitoring Area Feature and Layout

Feature

Additional information about the selected object or the executed operation is displayed in the monitoring area.

Layout

The monitoring area is located at the bottom of the interface and displays information including Output, Cross-reference, Compile, and Pre-Check, as shown in the following figure.



No.	Name	Description
①	Tab bar	Displays information about Output, Cross-reference, Compile, and Pre-Check. Click the pushpin button to hide the monitoring area. • Output: Used to display and edit properties of the project tree, workspace, or selected object in the workspace. When different objects in the workspace are selected, the Output area displays different information. • Cross-reference: Displays cross-reference relationships for the selected project tree node on the left or the selected object in the workspace, in a tree list format. • Compile: Displays information related to device compilation. • Pre-check: Checks for syntax errors in the configuration project in a real-time manner and displays corresponding prompts. Pre-check performs only basic syntax checks and does not include analysis or checks for user-defined types.
②	Toolbar	Used in the "Output," "Compile," and "Pre-check" tabs to filter the prompt types to be displayed. The following prompt types can be displayed or hidden: • Error • Alarm • Information
③	Display content	Displays additional information about the selected object or the executed operation.

5.6.2 Floating a Tab

You can float tabs in the monitoring area into separate windows and then dock them back at any time.

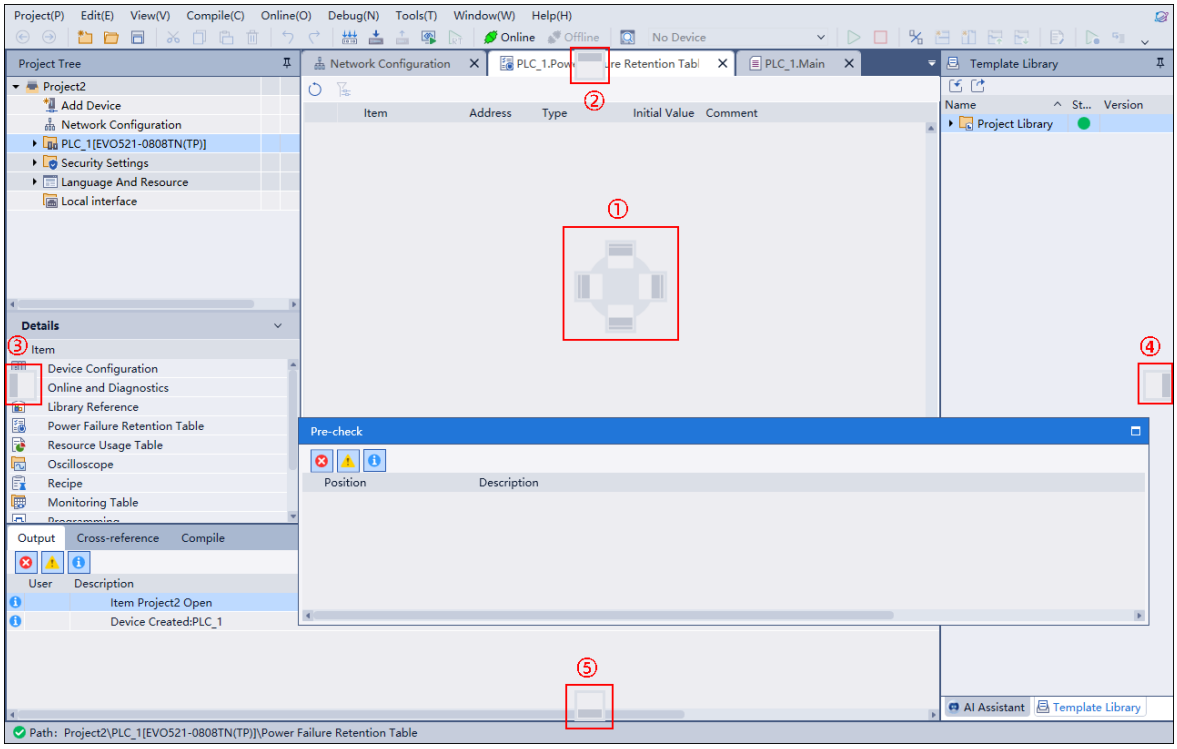
Float a tab

Hover over a tab in the tab bar of the monitoring area, and then drag the tab.

The tab will move out of the monitoring area and appear in a separate window. At this point, you can put the window at any desired position.

Dock a floating tab

- Method 1:
  1. Hover over the title bar of a floating tab and drag the window until the docking buttons appear on the interface, as shown in the following figure.
     ① indicates docking into a specific direction within the area.
     ②③④⑤ indicate docking into the top, left, right, and bottom sides of the project view, respectively.



  2. Drag the floating box to the target position indicated by the docking button.

- Method 2: In the "Window" menu, select "Reset window layout" to dock the floating window back to its previous position.

> **NOTE**
>
> The Reset Window Layout function does not apply to docked windows, which means once a floating window is docked, Clicking "Reset window layout" cannot restore the window to its previous position.

5.6.3 Maximizing a Tab

You can maximize a tab in the monitoring area.

Maximize a tab

1. Hover over a tab in the tab bar of the monitoring area, and then drag the tab to float it.
2. Click in the upper right corner of the floating window.

Restore the maximized tab

Click  in the upper right corner of the maximized window to restore the window.

5.7 Toolbox

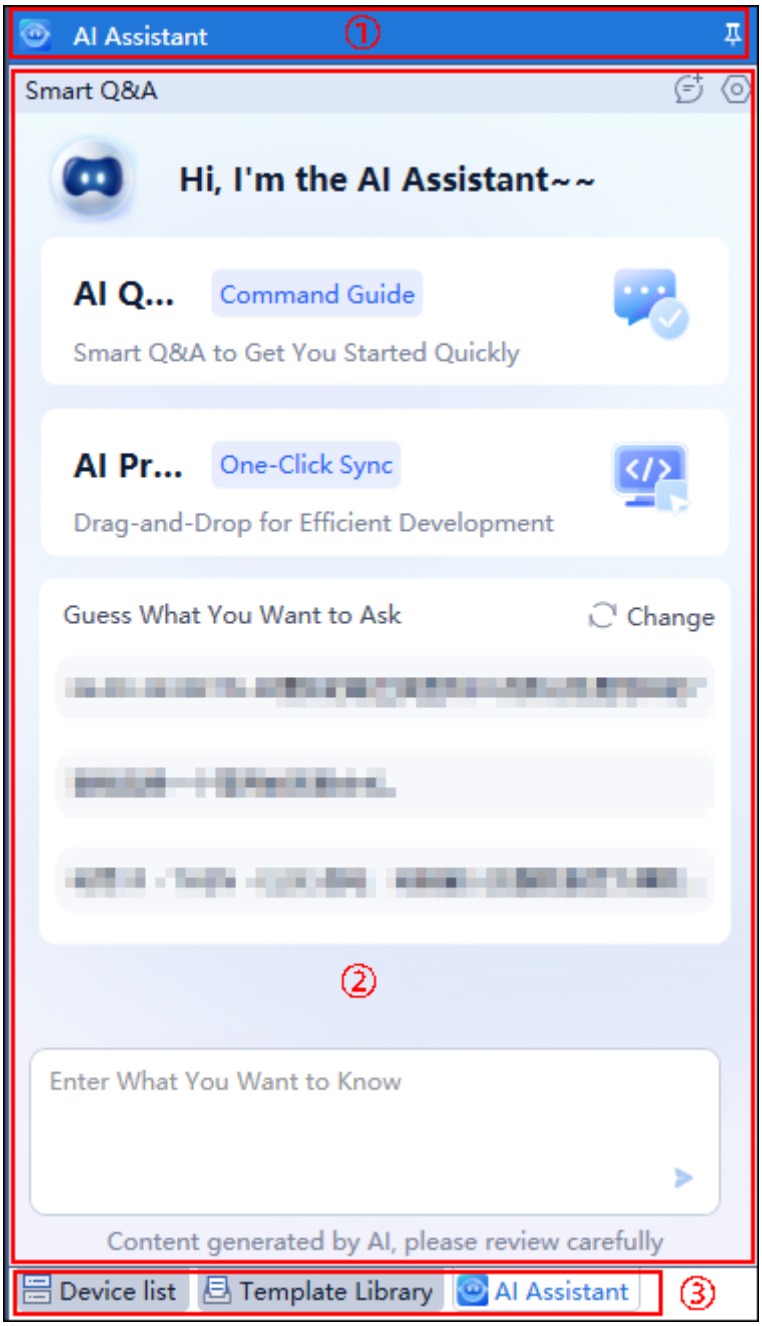
5.7.1 Toolbox Features and Layout

Features

The toolbox provides the following features:

- Appropriate tool tabs for performing additional operations based on the active object or view type in the workspace. For example, when the device configuration and network configuration views are present in the workspace, the "Device list" tab is displayed.
- AI Assistant: a smart Q&A tool that provides answers to questions regarding industrial automation, PLC, servo, ST programming, or other related fields.

Layout



No.	Name	Description
①	Title bar	Title bar of the Toolbox, which can be hidden using the pushpin button.
②	Function area	• Provides related functions based on the tool tabs. • AI Assistant Q&A workspace.

No.	Name	Description
③	Tab bar	Provides tool tabs, including Device List, Template Library, AI Assistant, etc. The Template Library and AI Assistant tabs are pinned, while other tabs automatically appear based on the active object or view type in the workspace. • Library: Displays elements from the project library and opened global library. • Device List: Displays hardware components by type to facilitate component selection. • Toolbox: When the workspace is in POU view, the toolbox includes operation methods, library files, and operators for configuration programs. • General Properties: When the workspace is in the monitoring variable table view, the toolbox adds a General Properties tab, allowing reconfiguration of variable-related parameters. • Monitoring Toolbox: When the workspace is in the monitoring view, the toolbox adds a Toolbox tab, displaying specific graphics and controls for configuring a monitoring screen. • AI Assistant: a smart Q&A tool that provides answers to questions regarding industrial automation, PLC, servo, ST programming, or other related fields.

5.7.2 Floating a Tab

You can float tabs in the tab bar of the toolbox into separate windows and then dock them back as desired.

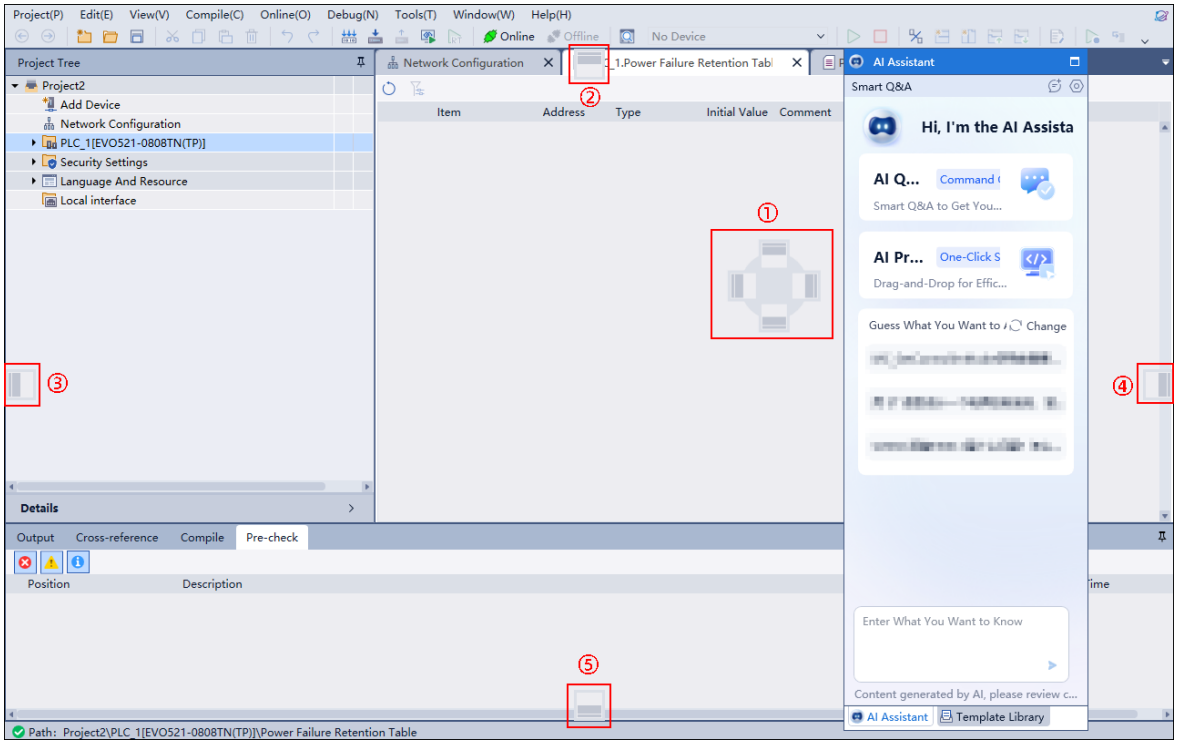
Float a tab

Hover over a tab in the tab bar of the toolbox and drag the tab.

The tab will move out of the toolbox and appear in a separate window. At this point, you can put the window at any desired position.

Dock a floating tab

- Method 1:
  1. Hover over the title bar of a floating tab and drag the window until the docking buttons appear on the interface, as shown in the following figure.
     ① indicates docking into a specific direction within the area.
     ②③④⑤ indicate docking into the top, left, right, and bottom sides of the project view, respectively.



2. Drag the floating box to the target position indicated by the docking button.

- Method 2: In the "Window" menu, select "Reset window layout" to dock the floating window back to its previous position.

> **NOTE**
>
> The Reset Window Layout function does not apply to docked windows, which means once a floating window is docked, Clicking "Reset window layout" cannot restore the window to its previous position.

5.7.3 Maximizing a Tab

You can maximize a tab in the toolbox.

Maximize a tab

1. Hover over a tab in the tab bar of the toolbox, and then drag the tab to float it.
2. Click in the upper right corner of the floating window.

Restore the maximized tab

Click in the upper right corner of the maximized window to restore the window.

6 System Settings

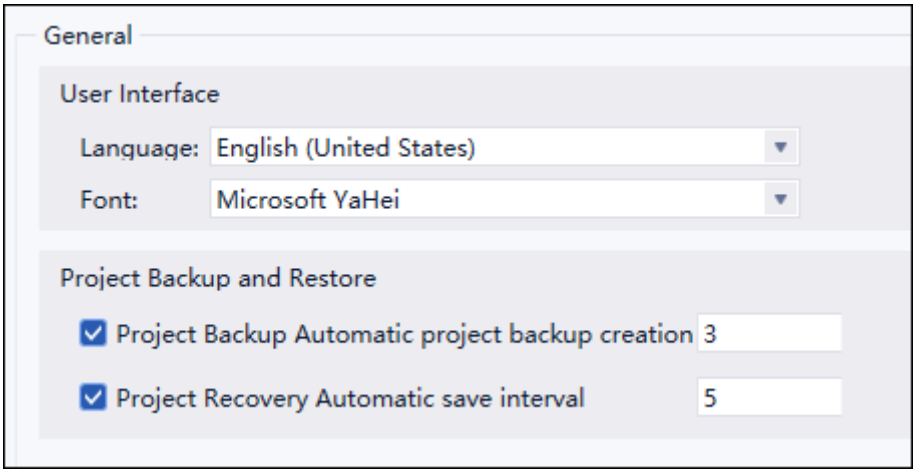
6.1 General Settings

Choose "Tools -> Options" from the menu bar, and open the "General" tab, where the following settings can be made.

General settings

- Language: Switch between Chinese and English.
- Font: Select the system font. Changing the font may affect all comments on the interface, including but not limited to menu names, tab names, and node names.
- Project backup: For details about automatic project backup, see 8.4 Backing up a Project.
- Project restore: Set the automatic save interval (in minutes). You can use the default 5 min or set another value [1-1440].
  When a project is not manually saved, it will be automatically backed up at the set interval for project recovery. The default file path is the same as the project creation path, with the default name being projectname.recovery.

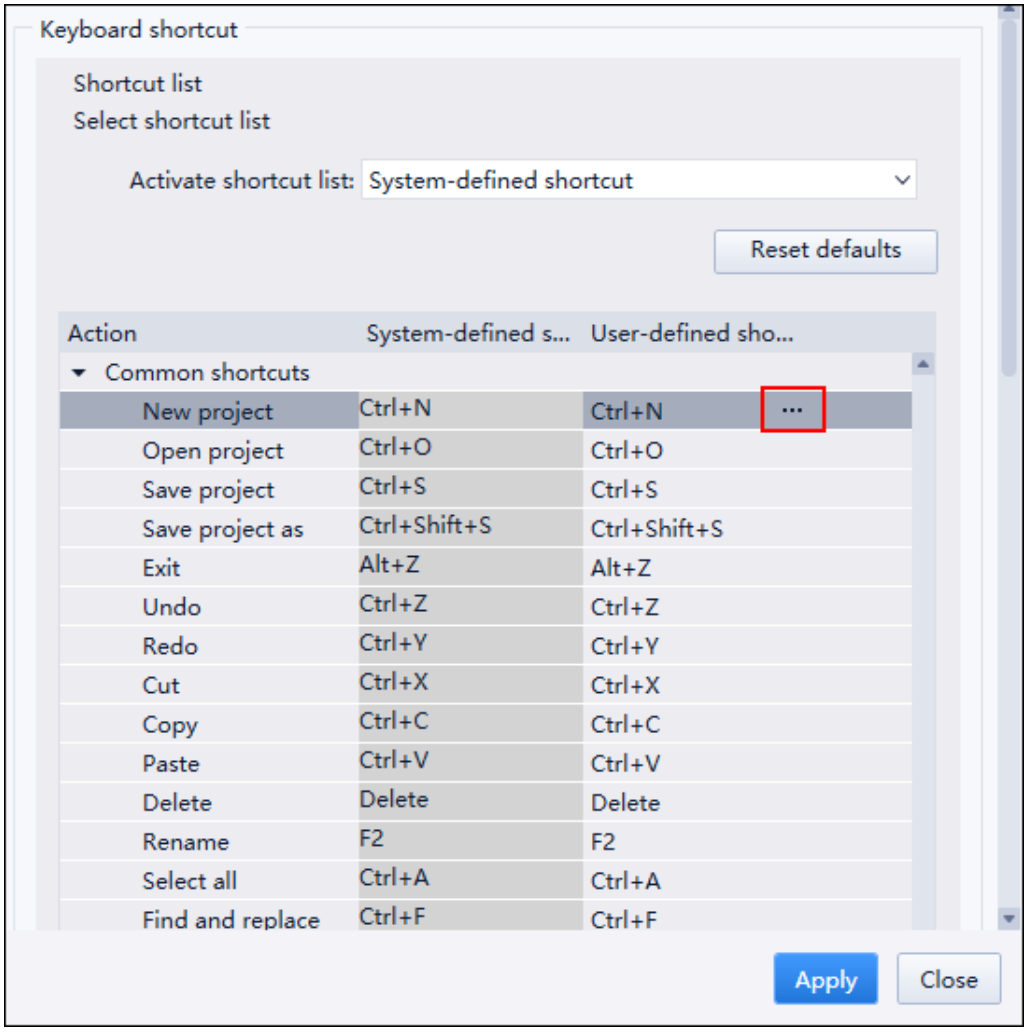
  Any of the above settings will take effect only after "OK" is clicked.
  When reopening the software, you will see a prompt asking whether to open the original project or the automatically backed-up project. Choose the appropriate option as needed.



Keyboard shortcuts

This tab is used to view, modify, and delete shortcuts for editor-related operations.

- System-defined Shortcuts: default shortcuts for editor-related operations.
- User-defined Shortcuts: shortcuts defined by the user. Select a row, and click "..." to open the editing window to modify or delete the system-defined shortcut.
- Reset to Default Values: After modifying any default shortcuts, you can use this button to reset all shortcuts to their default values. This operation will take effect after "Apply" is clicked.



Script/text editor

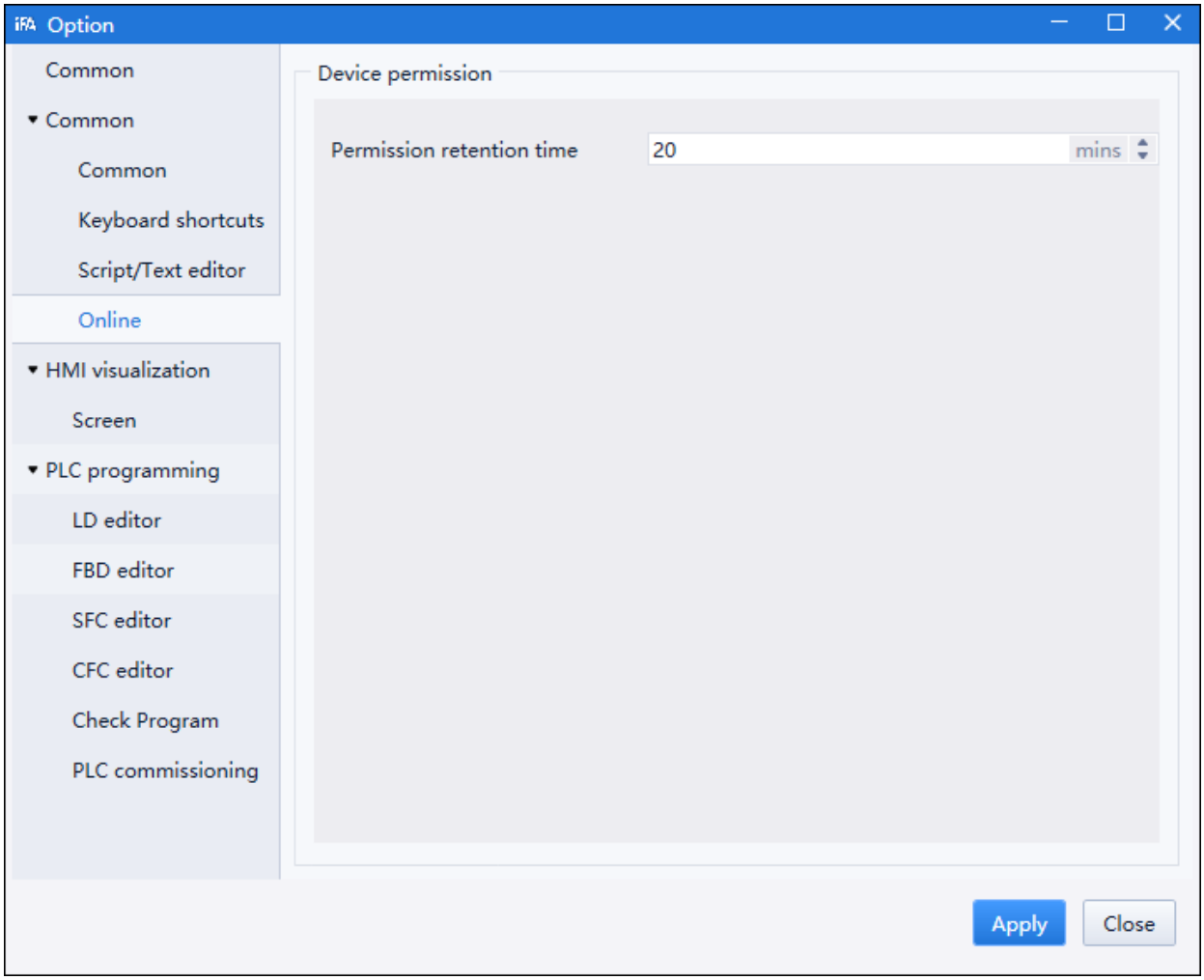
You can use the script/text editor to set the editor font and color.

For details, see 10.1.4.3.2 ST Editor10.1.4.3.2 Settings.

Online

Set the online duration (in minutes) for the PLC. The default value is 20, and the value range is [1-1440]. This setting will take effect after "Apply" is clicked.

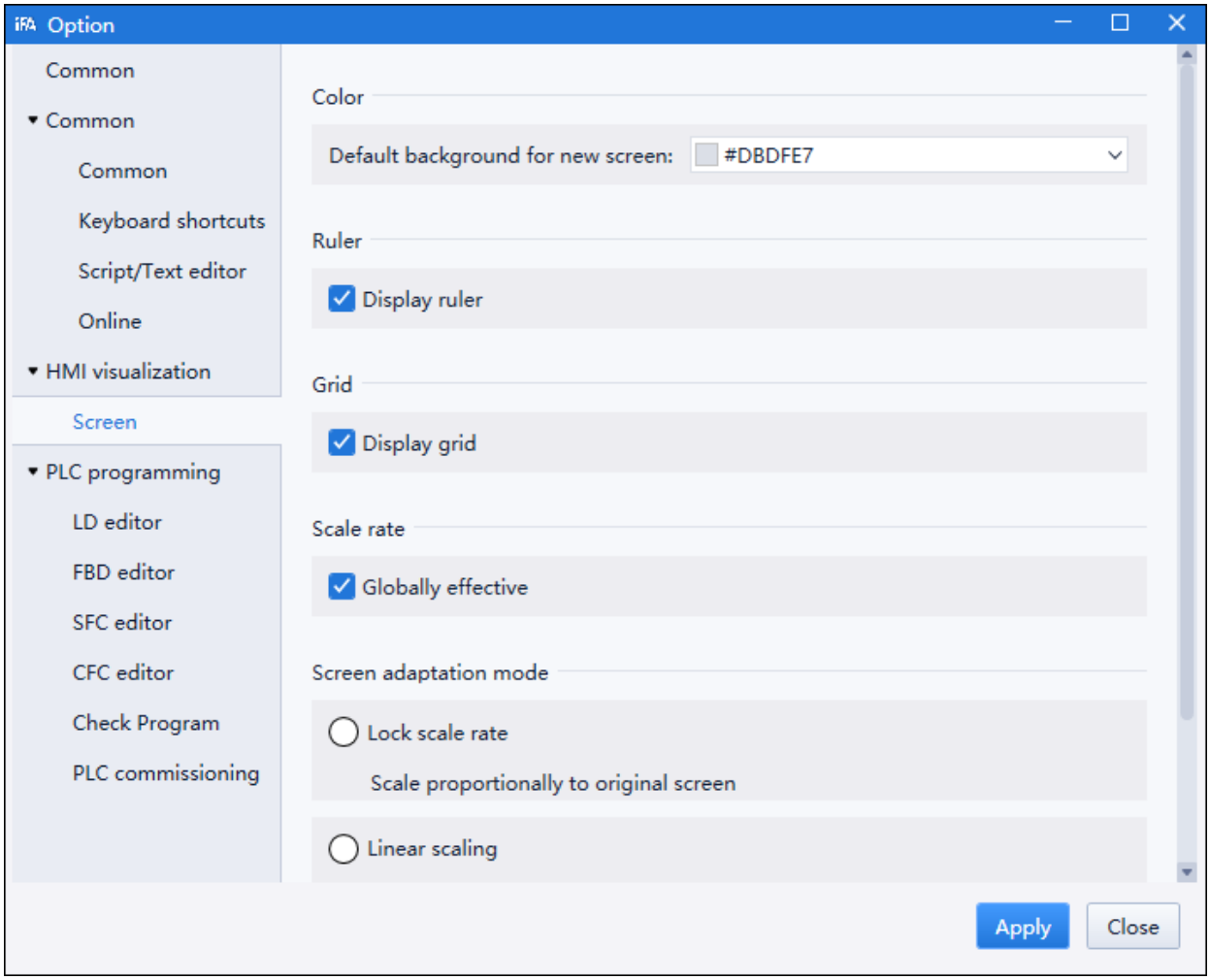
When a device comes online, you can bypass account verification within the valid time range. The permission validity is reset when the current project or software is closed.



6.2 HMI Visualization

Choose "Tools -> Options" from the menu bar, and open the "Screen" tab under Visualization to perform the following settings:

- Default Background for New Screen: Set the background color of the screen. After modifying the color, you can choose to automatically restore the default background color.
- Ruler: Set whether to display the ruler in the screen.
- Grid: Set whether to display the grid in the screen background.
- Zoom level: Set whether the screen zoom factor applies globally. This option is checked by default. Adjusting the zoom level of any screen will synchronously update the zoom level of all screens in the project. For details, see 13.3.10.2 Zooming a Screen.



Any of the above settings will take effect immediately after "Apply" is clicked.

6.3 PLC Programming

LD editor

For details, see 10.1.3.3.2 LD Editor Settings.

FBD editor

For details, see 11.2.5.3 Setting the Safety FBD Editor.

SFC editor

For details, see 10.1.6.3.2 SFC Editor Settings.

CFC editor

For details, see 10.1.5.3.2 CFC Editor Settings.

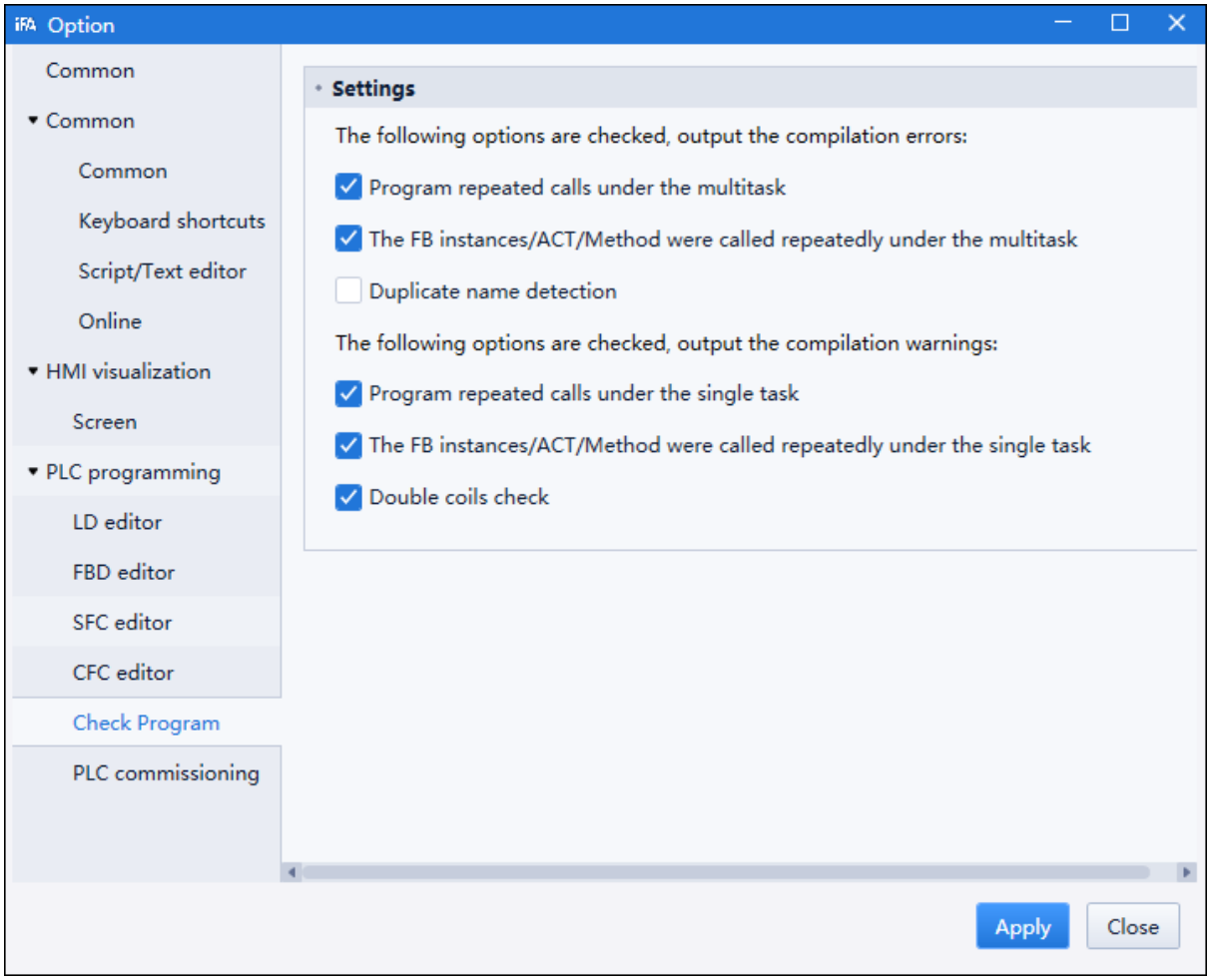
Program check

Program check is used for comparing or verifying whether all programs under the task scheduling list are repeatedly called within the programming environment to ensure logical consistency and accuracy.

Configuring program check

1. Choose "Tools -> Options" from the menu bar.

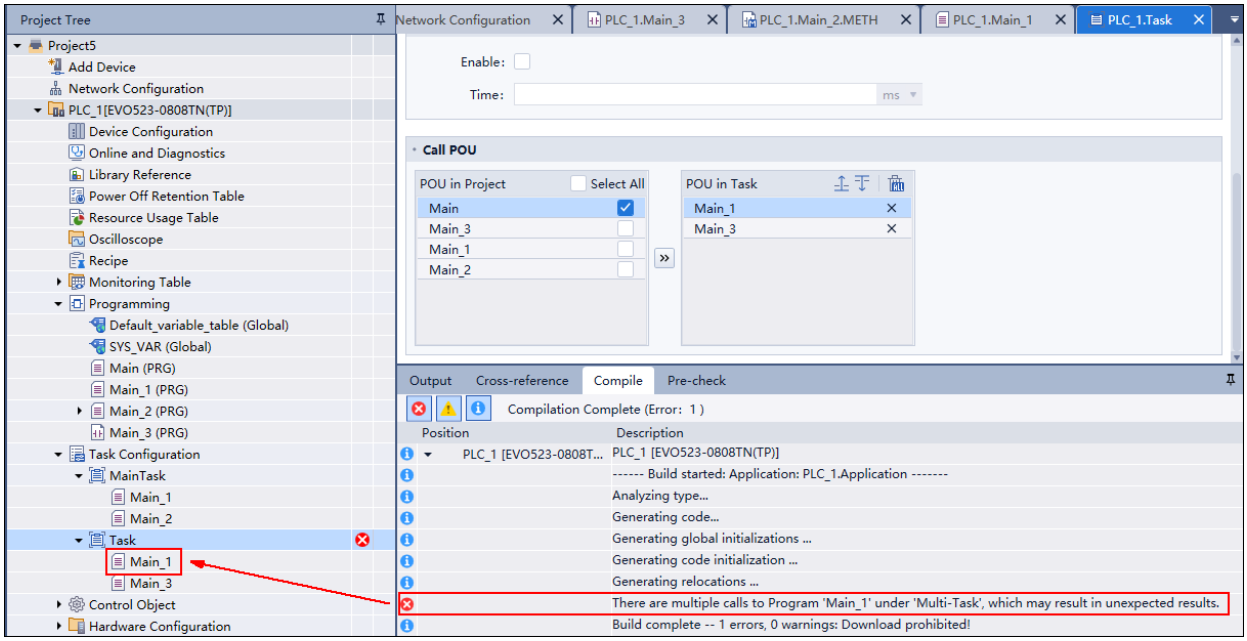
2. Choose "PLC programming -> Check program" in the displayed "iFA Options" window.
3. Select the appropriate check items. During compilation or pre-check, the system checks the selected items and outputs any errors or warnings. For details on the check items, see description of check items.



4. Click "Apply" to apply the configuration.

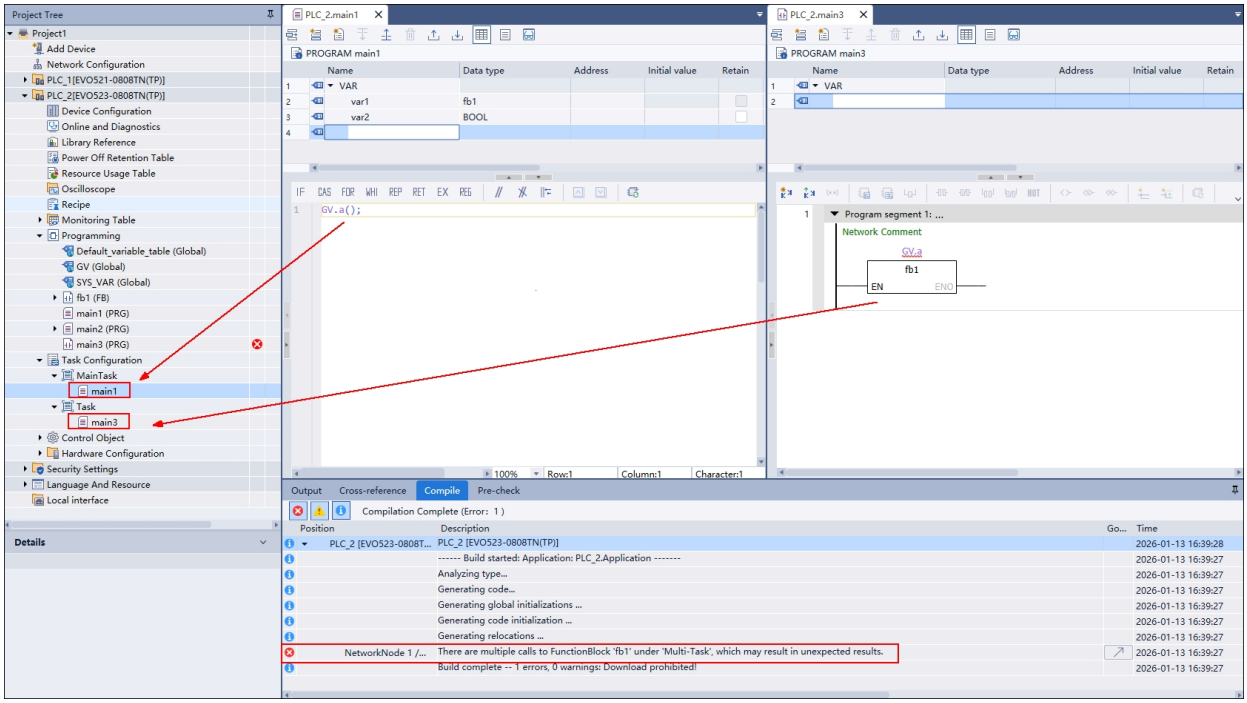
Description of check items

- Once the following options are applied, the system will output any compilation error:
  - Program called repeatedly in multiple tasks
    In this case, an error will be displayed, indicating that the program XXX is called in multiple tasks, which may lead to unexpected results.



- FB instance/ACT/Method called repeatedly in multiple tasks

  In this case, an error will be displayed, indicating that the function block/action/method XXX is called repeatedly in multiple tasks, which may lead to unexpected results.



Duplicate name check

Check whether any symbols in the project have duplicate names. The scope of the check includes referenced libraries, POUs, global variable tables, default variable tables, system variable tables, custom types, and device system variables. If there is a duplicate name among the aforementioned objects (all involved variables are global variables), an error "duplicate names for XXX and XXX" will be displayed.

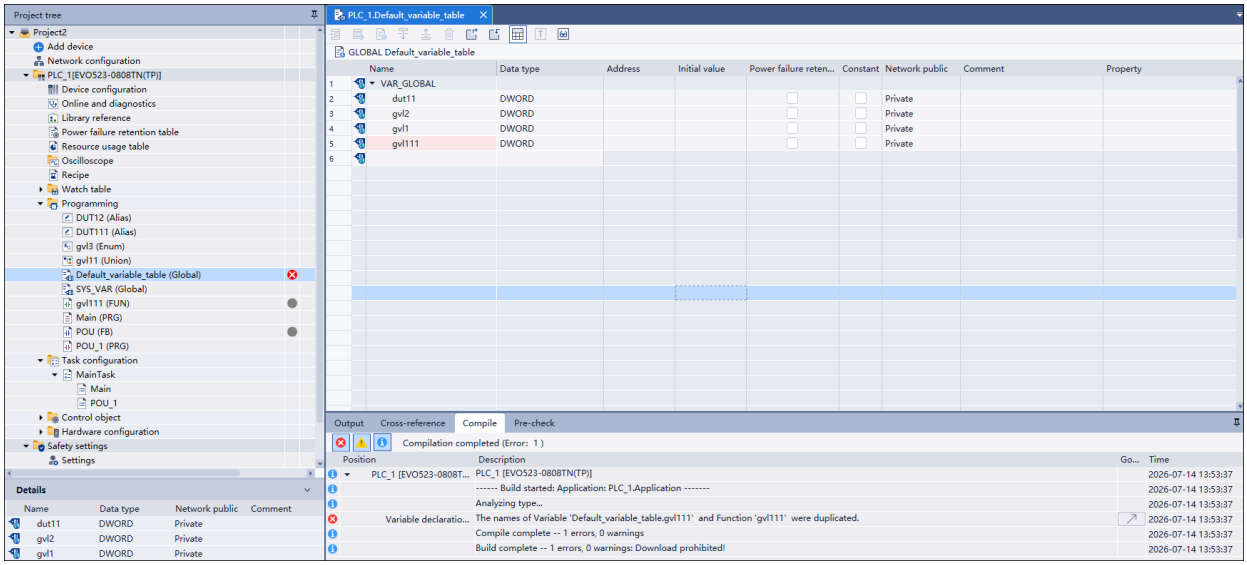


Figure 6-3-1 Compilation error

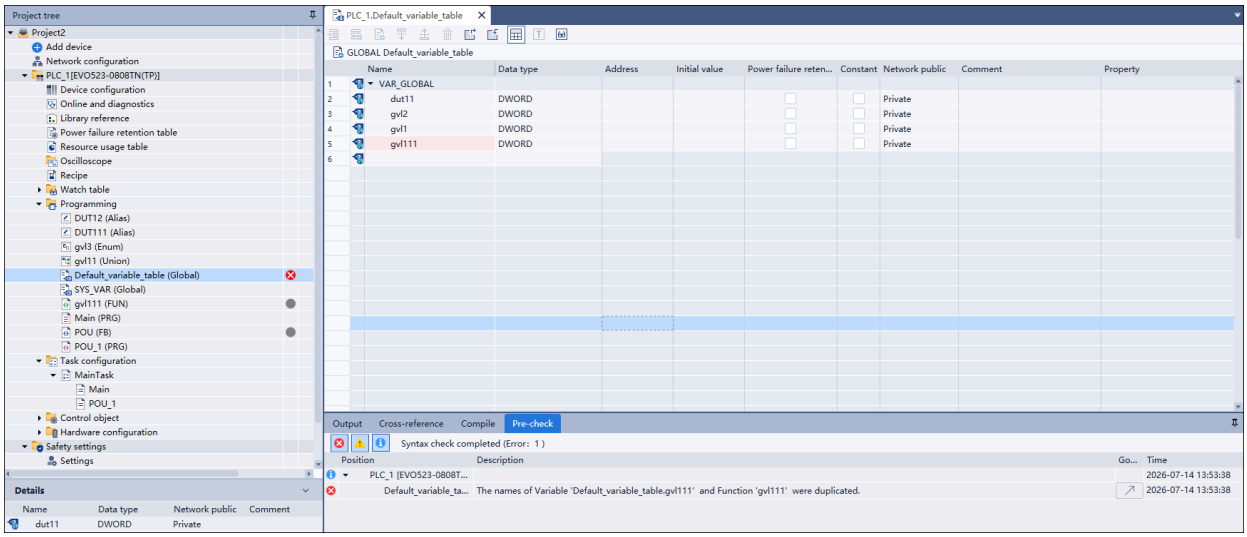
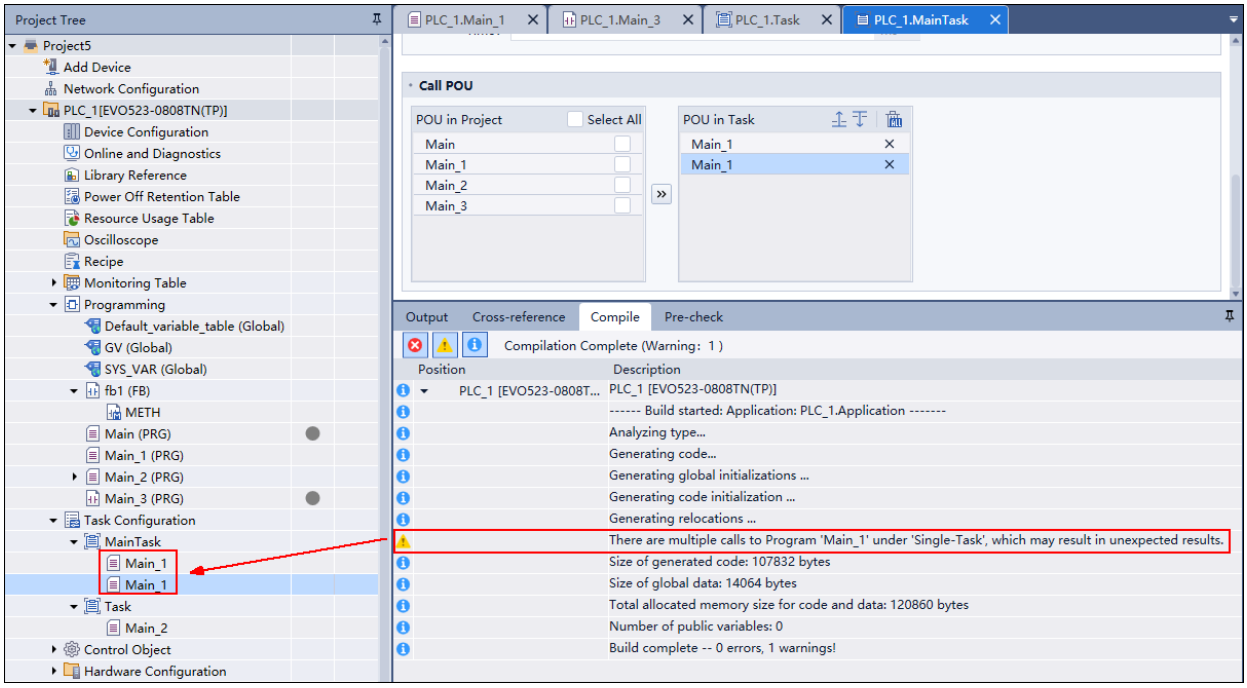


Figure 6-3-2 Pre-check error

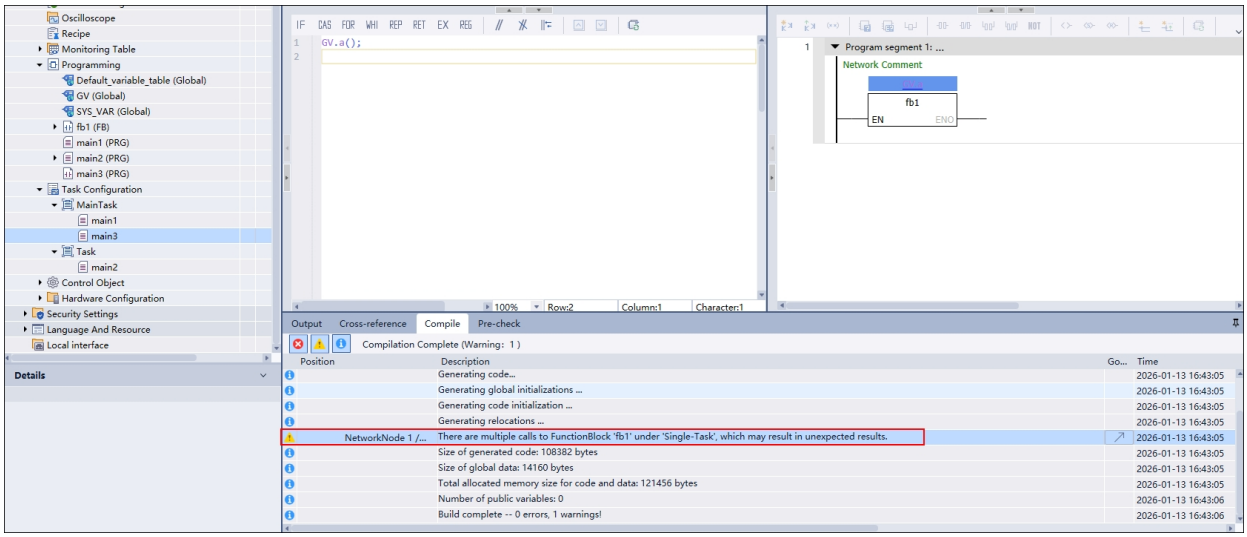
- Once the following options are applied, the system will output any compilation alarm:
  - Program called repeatedly in the same task

    In this case, an alarm will be displayed, indicating that the program XXX is called repeatedly in the same task, which may lead to unexpected results.



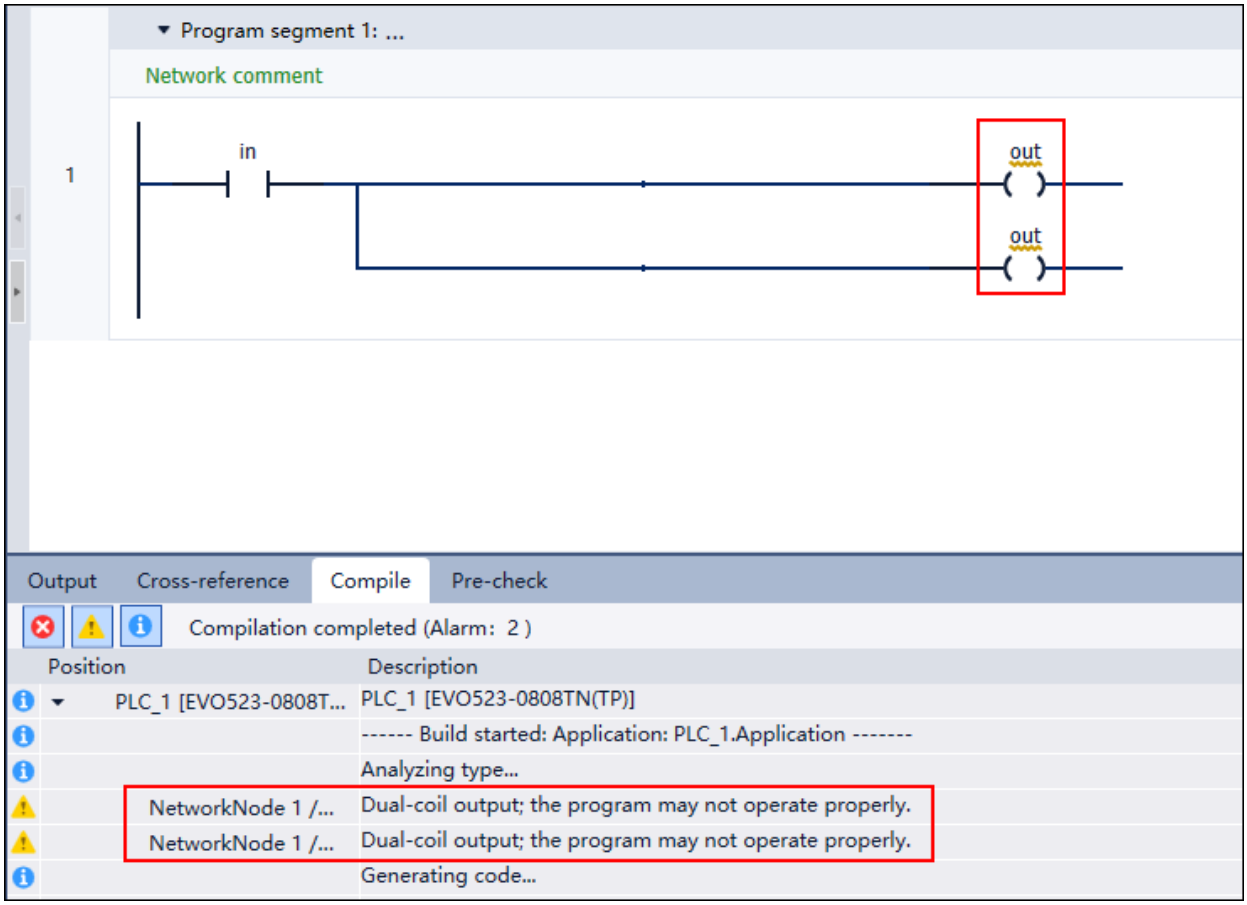
- FB instance/ACT/Method called repeatedly in the same task

  An alarm will be reported, indicating that the function block/action/method XXX is called in the same task, which may produce unexpected results



- Double coil check

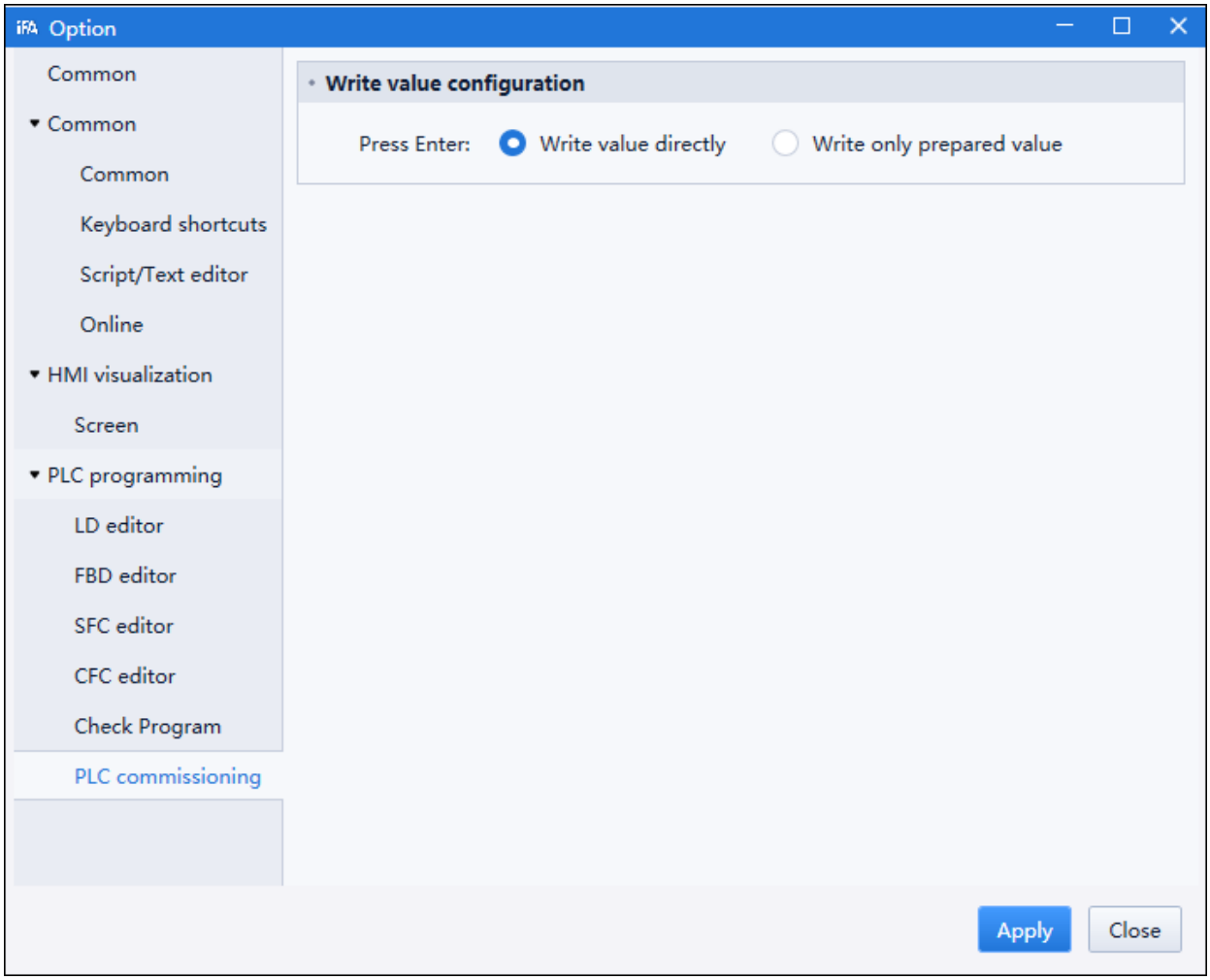
  Check whether the same variable is assigned to multiple coils. Once this option is checked, the system will scan all ladder diagram (LD) POUs called by the task during the program check or compilation, and will issue an alarm when any variable assigned to multiple double coils is found (reset/ set coils are excluded from the check).



PLC commissioning

PLC commissioning is used for configuring the method of writing values to the program editing area in online mode. You can configure whether pressing Enter applies the value immediately or stores it as a pending value.

Choose "Tools > Options > PLC programming > PLC commissioning" from the menu bar.

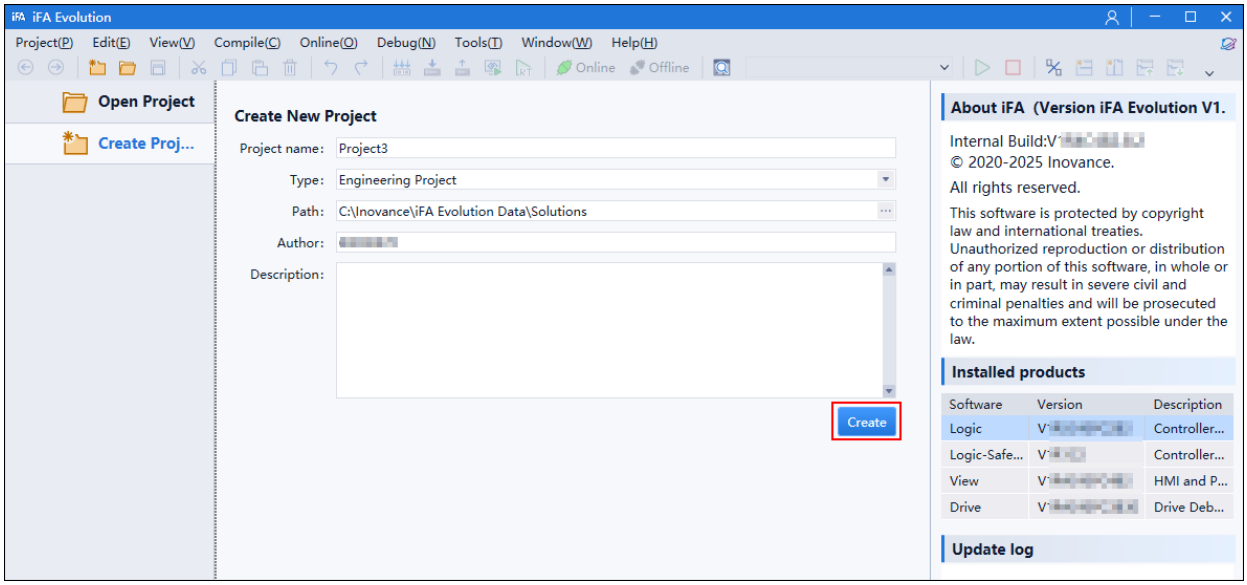

Option
Common
Common
Common
Keyboard shortcuts
Script/Text editor
Online
HMI visualization
Screen
PLC programming
LD editor
FBD editor
SFC editor
CFC editor
Check Program
PLC commissioning
Write value configuration
Press Enter:
Write value directly
Write only prepared value
Apply
Close

7 Project Deployment

7.1 Creating a Project

Proceed as follows to create a new project:

1. Open the "Create new project" window in one of the following ways:
   - Method 1: On the startup interface, select "Create project".
   - Method 2: On the system interface, choose "Project -> New" from the menu bar.
   - Method 3: On the system interface, click the New icon on the toolbar.
   - Method 4: On the system interface, use the shortcut key Ctrl+N.
2. Taking Method 1 as an example: After the "Create new project" window appears, configure the project information, and then click "Create". At this time, a new project is created and opened automatically.



Item	Description
Name	Enter the project name. The name can contain Chinese characters, English letters, numbers, and underlines, but cannot start with a number.
Type	Select a project type: • Engineering project • Library project
Path	Select the path where the project will be saved.
Author	Enter the author's name.
Description	(Optional) Enter the project description.

7.2 Configuring a Device

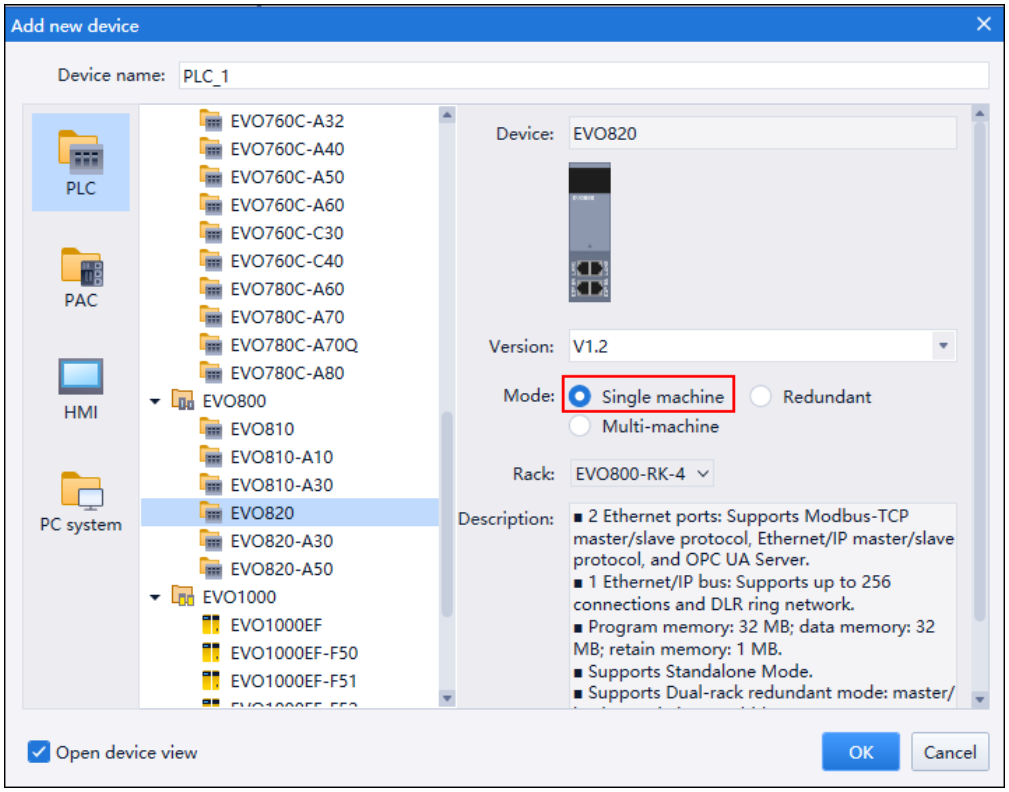
7.2.1 Adding a Device

You can add a device in the project tree after creating a project.

Taking the EVO800 series as an example, the procedure for adding the EVO800 series single PLC is similar to that for adding other series PLC.

Add a single PLC

1. Double-click the "Add device" node in the project tree to access the "Add new device" dialog box, in which the PLC is selected by default.
2. Click the required PLC, and the product information will be displayed on the right side, as shown in the following figure:



The product information related parameters are described in the table below.

Parameter	Description
Device	Displays the device model.
Version	Displays the current available versions of the device. When multiple versions are available, select the desired one from the drop-down list.
Mode (optional for EVO800 series)	Displays different modes. Select "Single machine" here.
Description	Displays related product information.

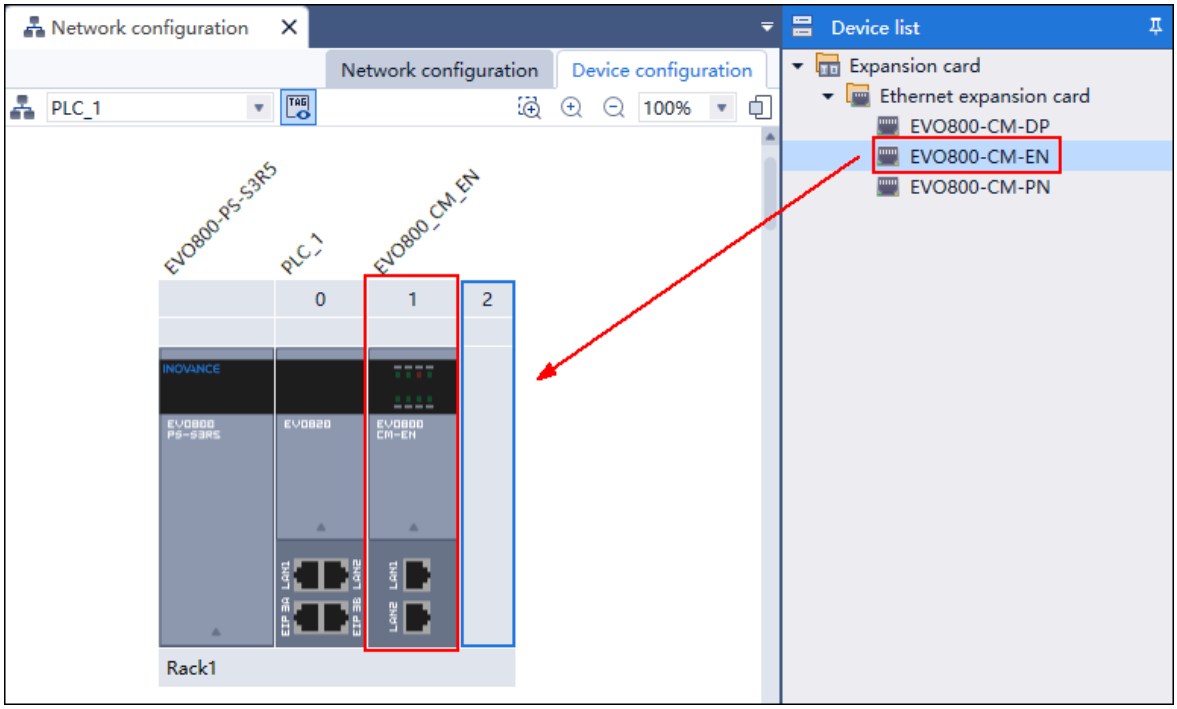
3. (optional for EVO800 series) Select an appropriate rack from the "Rack" drop-down list.
   The default rack is EVO800-RK-4; options include EVO800-RK-4, EVO800-RK-6, EVO800-RK-5R, and

EVO800-RK-7R. Among them, EVO800-RK-4 and EVO800-RK-6 are single-power racks, while EVO800-RK-5R and EVO800-RK-7R are dual-power racks. In the Device Configuration view, one power supply module can be seen in the single-power racks, and two power supply modules can be seen in the dual-power racks.

4. Click "OK" to add the PLC.

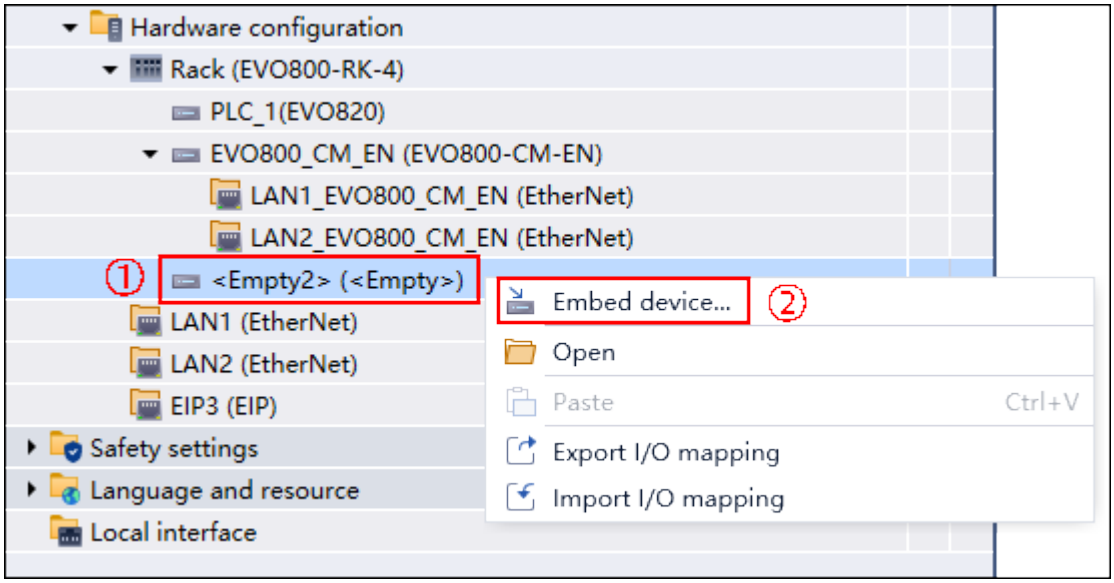
5. (Optional for EVO800 series) Add an Ethernet/DP master/PN master expansion module. The following uses the Ethernet expansion module as an example.
   In the project engineering, the slot positions of the configured modules must match the corresponding slot positions in the actual physical topology.

➢Method 1: In the "Device configuration" interface, select an empty slot, drag an Ethernet module from the "Device list" to this empty slot.



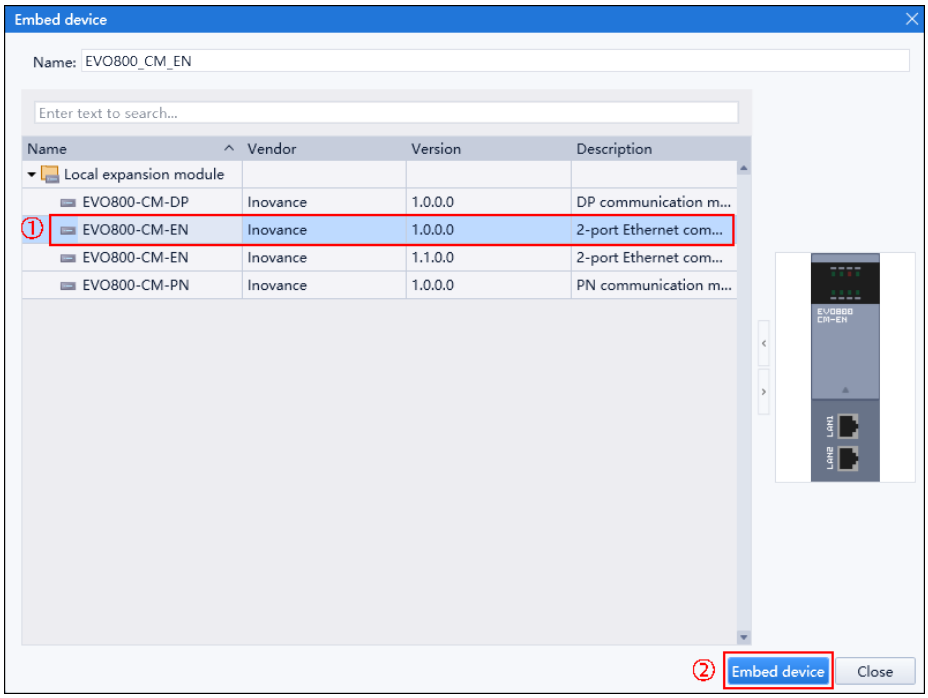
➢Method 2:

a. Right-click an empty slot in the "Project Tree" and select "Embed device" from the shortcut menu.



b. In the opened dialog box, select "EVO800-CM-EN", and click "Embed device" to add the Ethernet

expansion module.

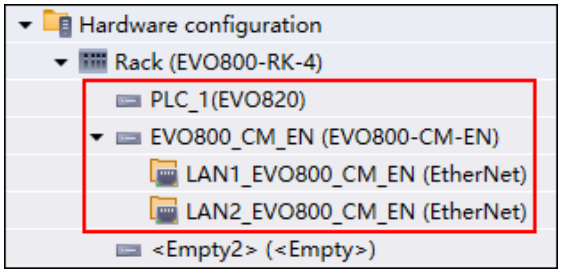


NOTE

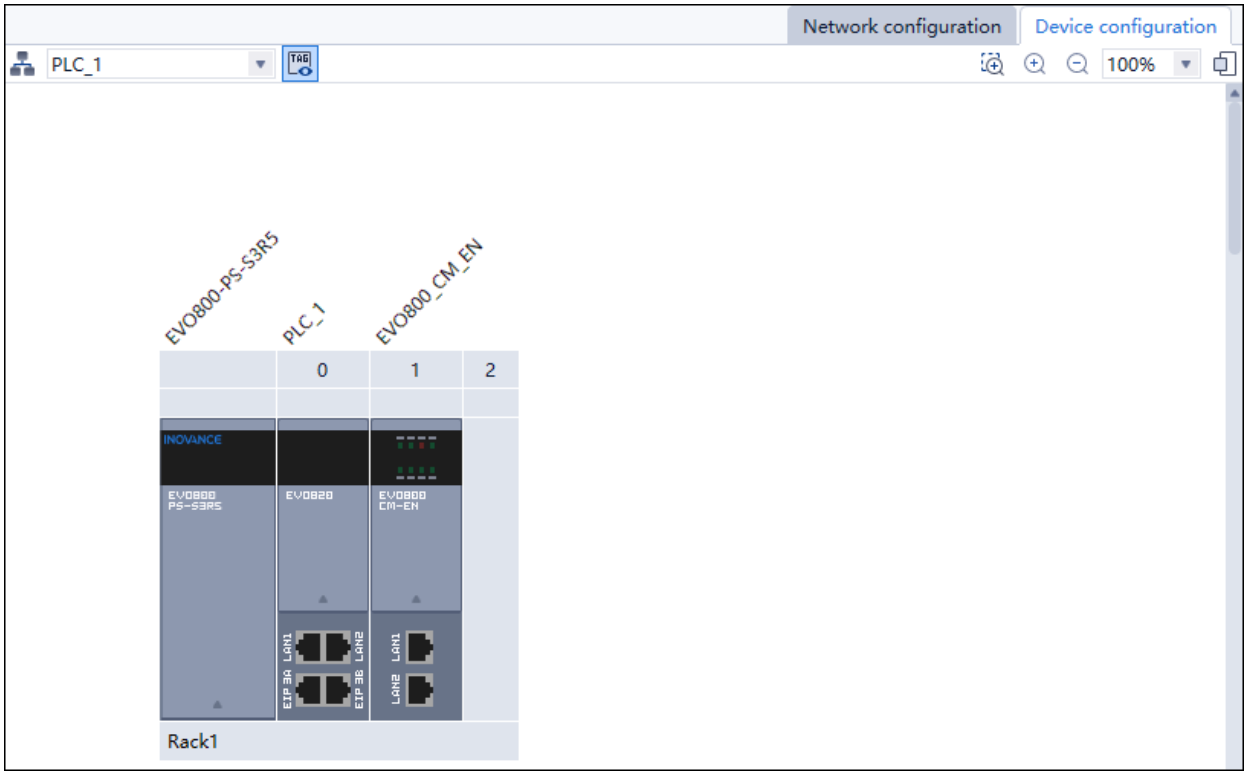
The added Ethernet expansion module is part of the EVO800 PLC.

Result

After the device is added, the device node appears in the project tree.

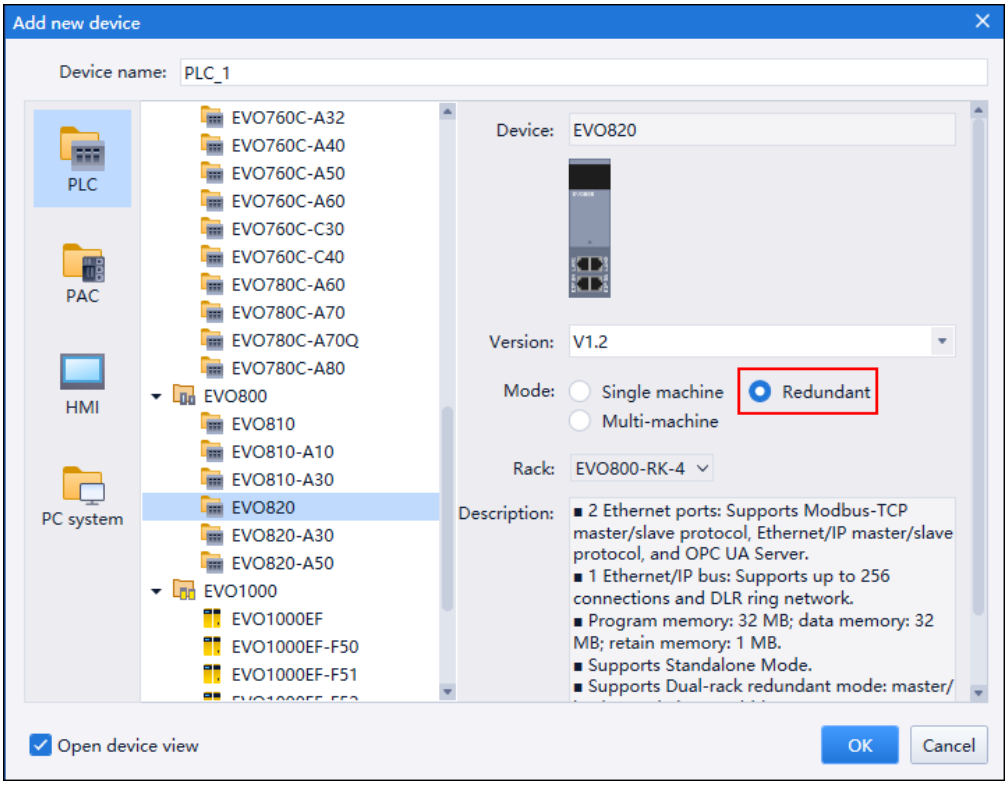


The Device Configuration view is shown as follows:



Add a redundant PLC

The procedure for adding a redundant PLC is basically similar to that for adding a single PLC, with the only difference being that the mode on the right side must be changed to "Redundant" after selecting the EVO820 product. Click "OK" to complete the addition of the redundant PLC.

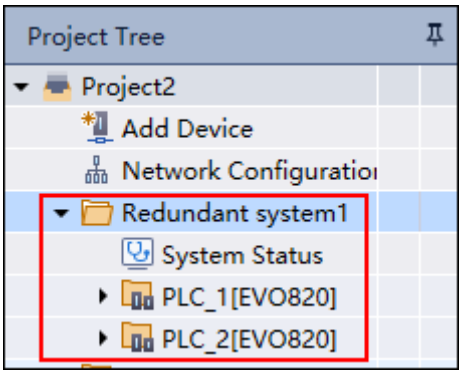


NOTE

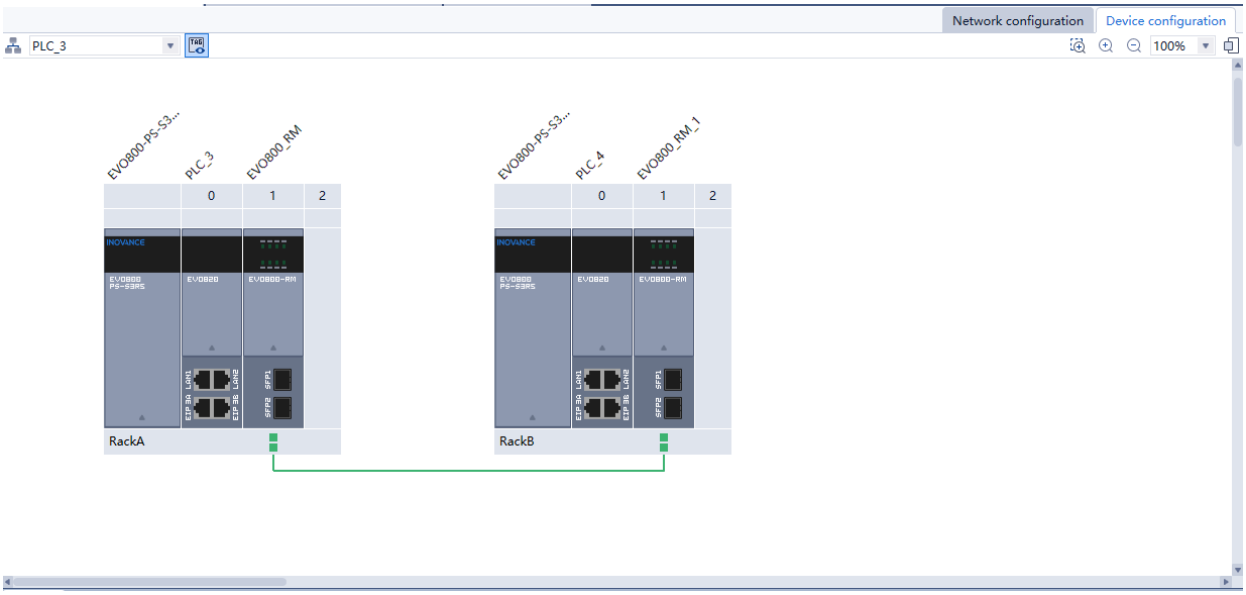
- Any of the EVO800-RK-4, EVO800-RK-6, EVO800-RK-5R, and EVO800-RK-7R racks also applies to the redundant PLC.
- The redundant mode is not applicable to EVO810 series devices.

Result

After adding the redundant PLC, the device node appears in the project tree.



The Device Configuration view for the redundant PLC is shown as follows:

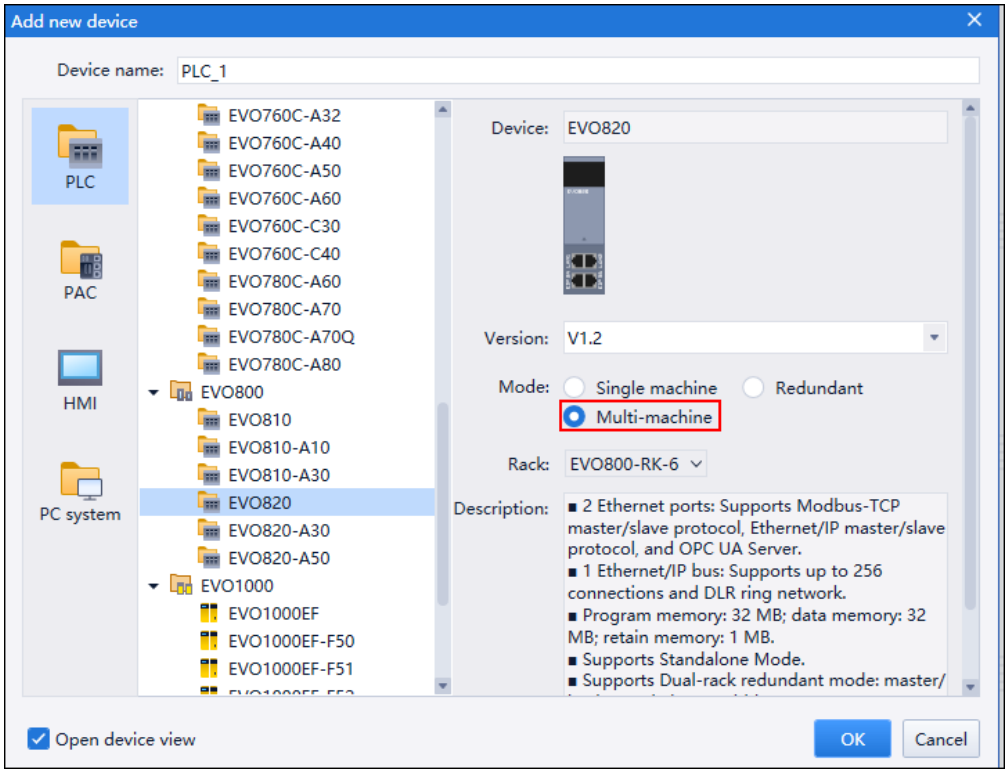


Add multiple PLCs

CAUTION

The firmware versions of all CPUs must be consistent. For specific firmware upgrade operations, see "14.1.1 Firmware Update."

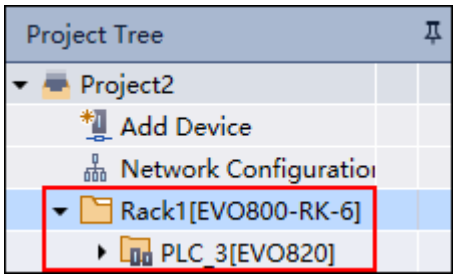
1. The procedure for adding multiple PLCs is basically similar to that for adding a single PLC, with the only difference being that the mode on the right side must be changed to "Multi-machine" after selecting the EVO820 product. Click "OK" to complete the addition of multiple PLCs.



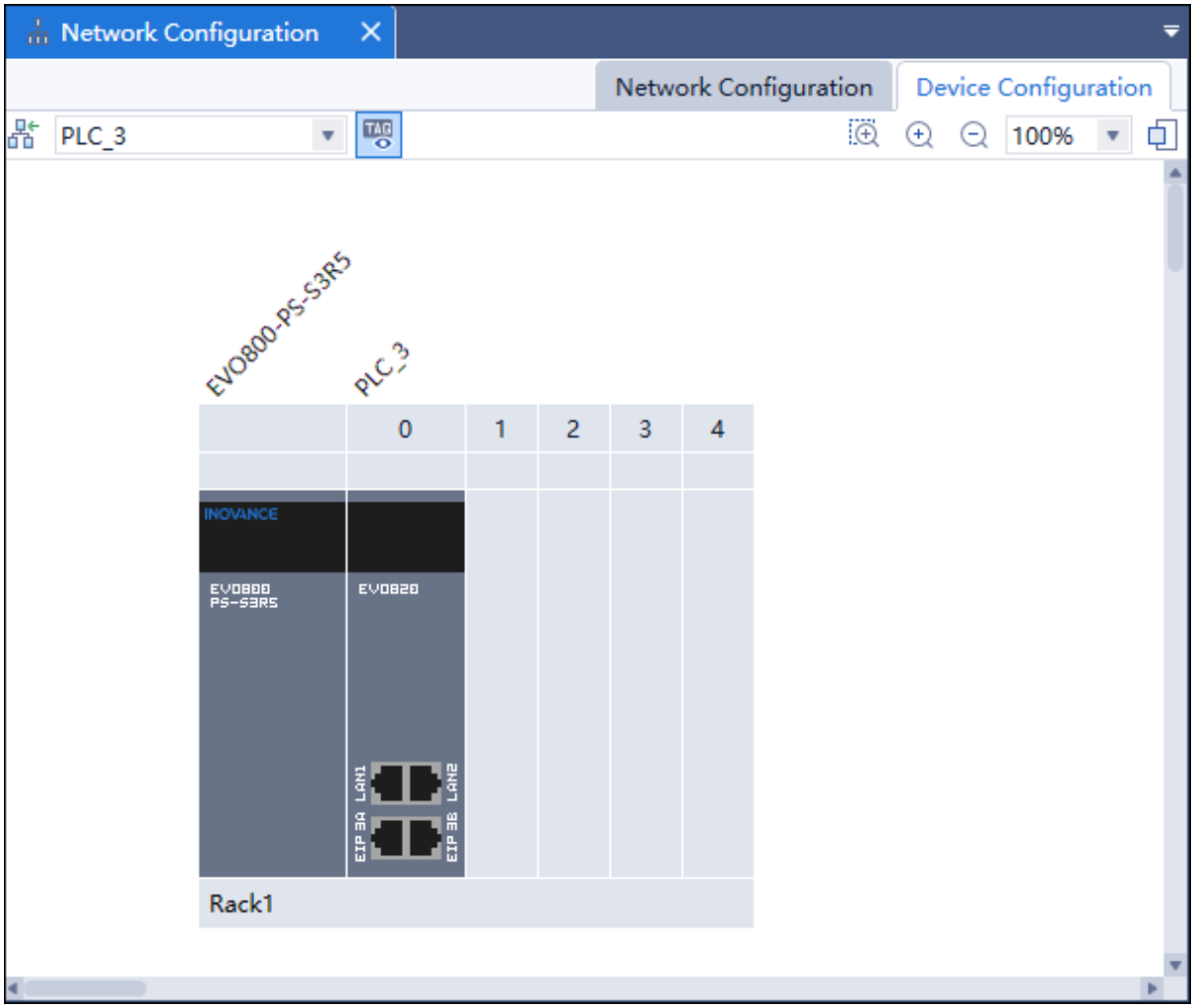
NOTE

- The default rack for multiple PLCs is EVO800-RK-6, but any of EVO800-RK-4, EVO800-RK-6, EVO800-RK-5R, and EVO800-RK-7R racks can be freely selected.
- Ethernet expansion modules can be also added for multiple PLCs.

After adding multiple PLCs, the device node appears in the project tree.



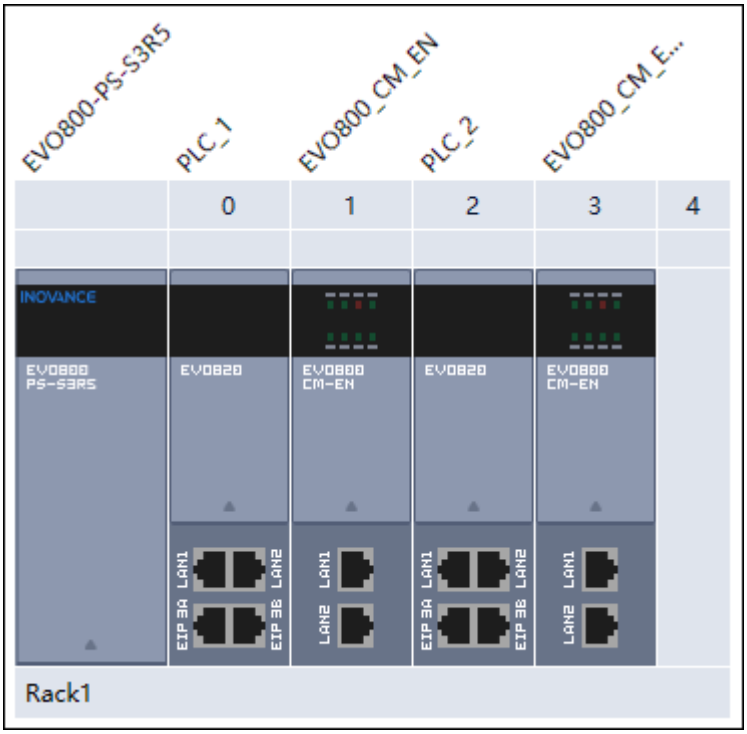
Generate a single PLC node. The Device Configuration view is shown as follows:



2. Add a PLC module or an Ethernet expansion module. For the addition procedure, see "Step 5" in the "Add a single PLC" section.

NOTE

- PLC modules can only be added via drag-and-drop from the "Device list" in the toolbox. A six-slot rack supports adding up to four PLC devices.
- The added Ethernet modules belong to the nearest PLC on their left. For example, in the figure below, the Ethernet expansion modules in slots 1 and 2 belong to PLC_1 in slot 0, and the Ethernet expansion module in slot 4 belongs to PLC_2 in slot 3.



7.2.2 Copying a Device

You can copy the relevant hardware devices in the Device Configuration or Network Configuration view. The copied hardware devices are stored in the clipboard and can be pasted to a valid location from the clipboard.

Usage

- Either a single device or multiple devices can be copied at a time.
- Copied devices and modules can be pasted to the rack in the Network Configuration or Device Configuration view.
- Modules installed in a rack can be copied.
- Copied devices and modules can be pasted to available slots provided that the slot usage is followed.

Procedure

1. Select a device you want copy.
   - Device Configuration: Select a module on the rack.
   - Network Configuration: Select a device.
   - Project tree: Select a device.
2. Select "Copy" from the shortcut menu or press <Ctrl+C>.

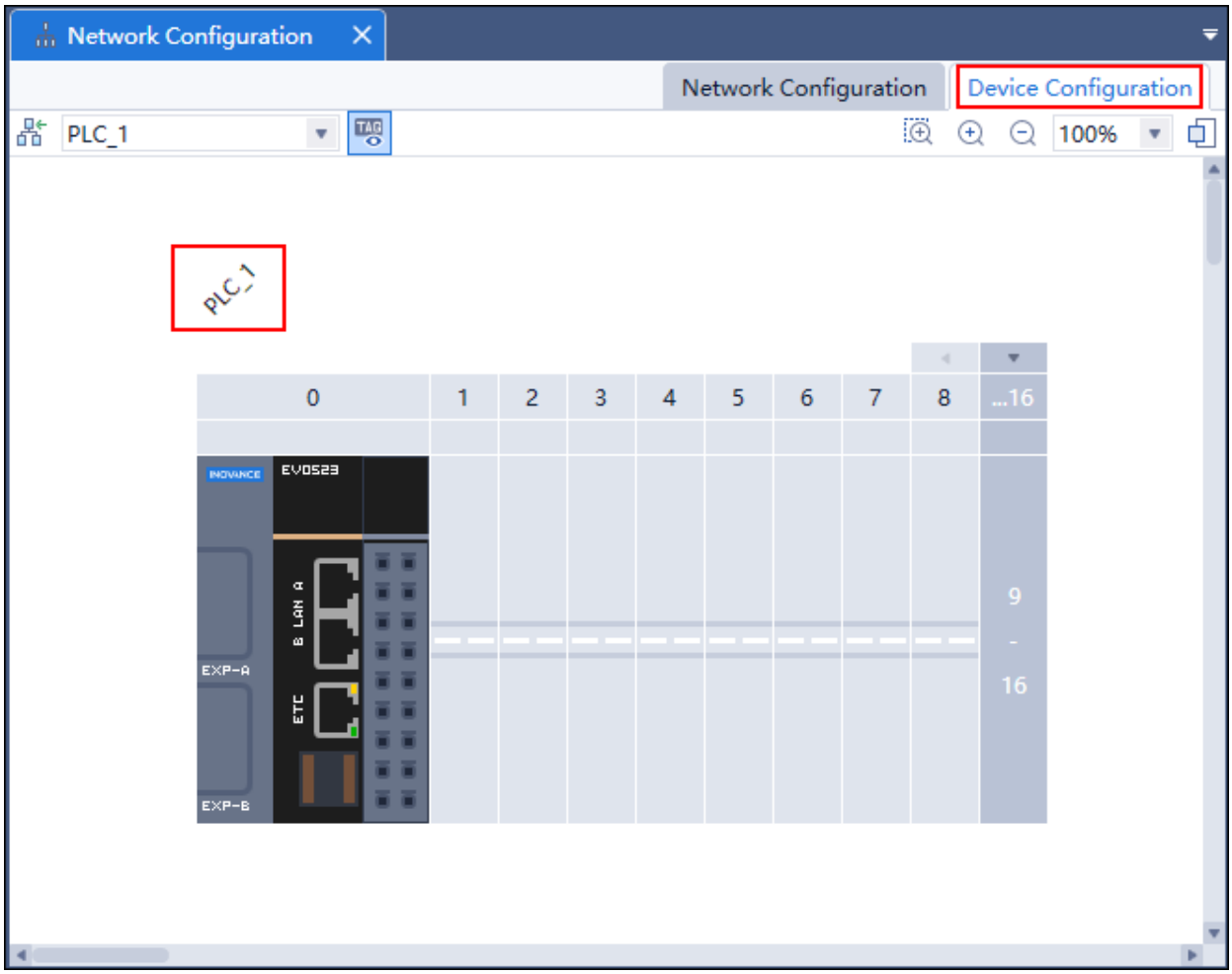
When the "Copy" item cannot be selected, it indicates that the selected content includes at least one device that cannot be copied.

3. Select the position where you want to paste the clipboard content.
   - Device Configuration: Select a free slot in the rack or an area without a module installed.
   - Network Configuration: Select a device to insert.
4. Select "Paste" from the shortcut menu or press <Ctrl+V>.
   - If you paste the copied object using the shortcut menu without making a selection in the Network Configuration view, the object will be pasted at the pointer position.
   - If you paste the copied object using <Ctrl+V> without making a selection in the Network Configuration view, the copied object will be pasted to the right of the original object.

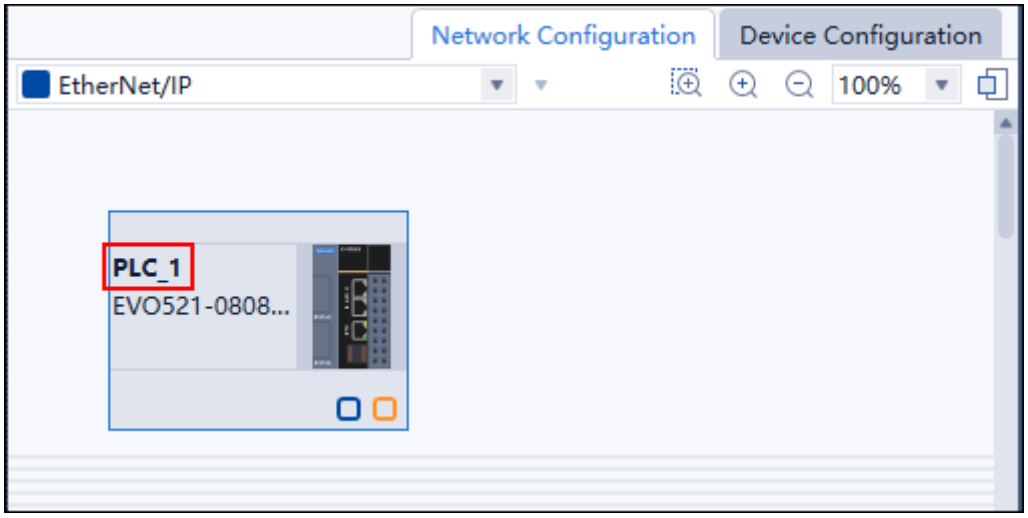
7.2.3 Renaming a Device

You can edit the device name in the "Network Configuration" and "Device Configuration" tabs for a device.

- Proceed as follows to rename a device in the graphics field of the "Device Configuration" tab:
  1. Selects a device.
  2. Double-click the name above the device to modify it.



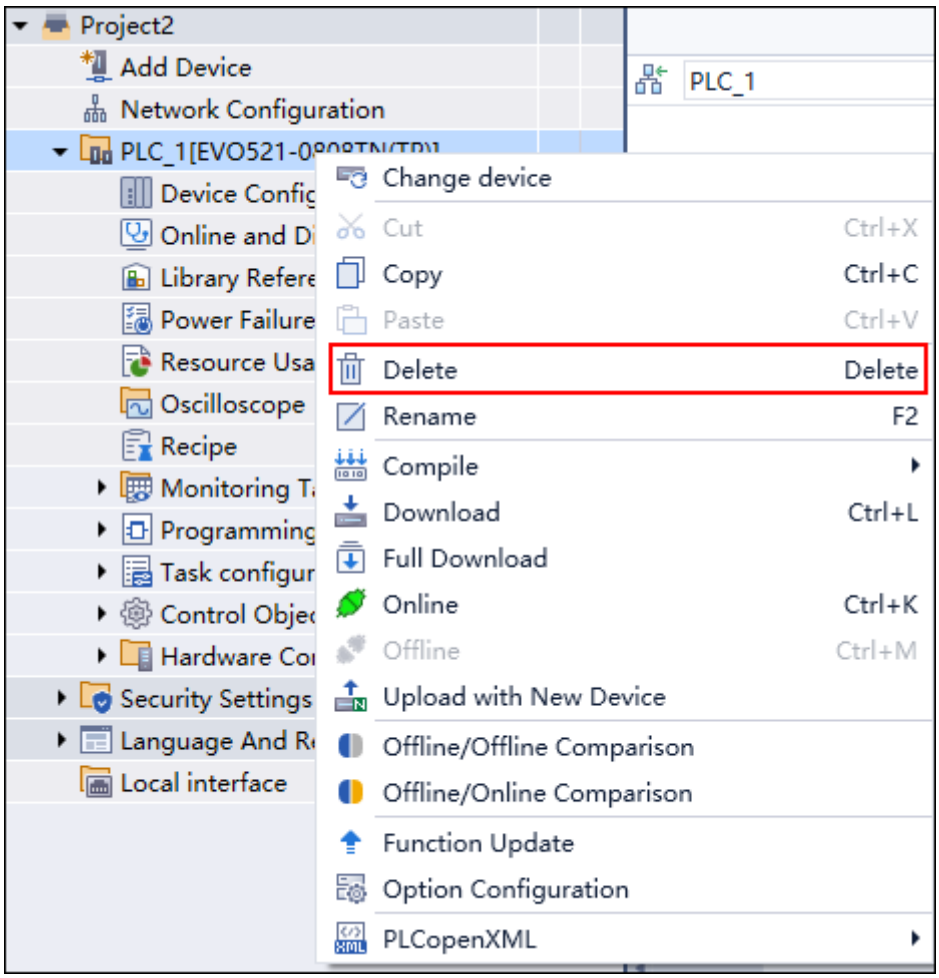
- Proceed as follows to rename a device in the "Network Configuration" tab:
  1. Selects a device.
  2. Double-click the name above the device to modify it.



7.2.4 Deleting a Device

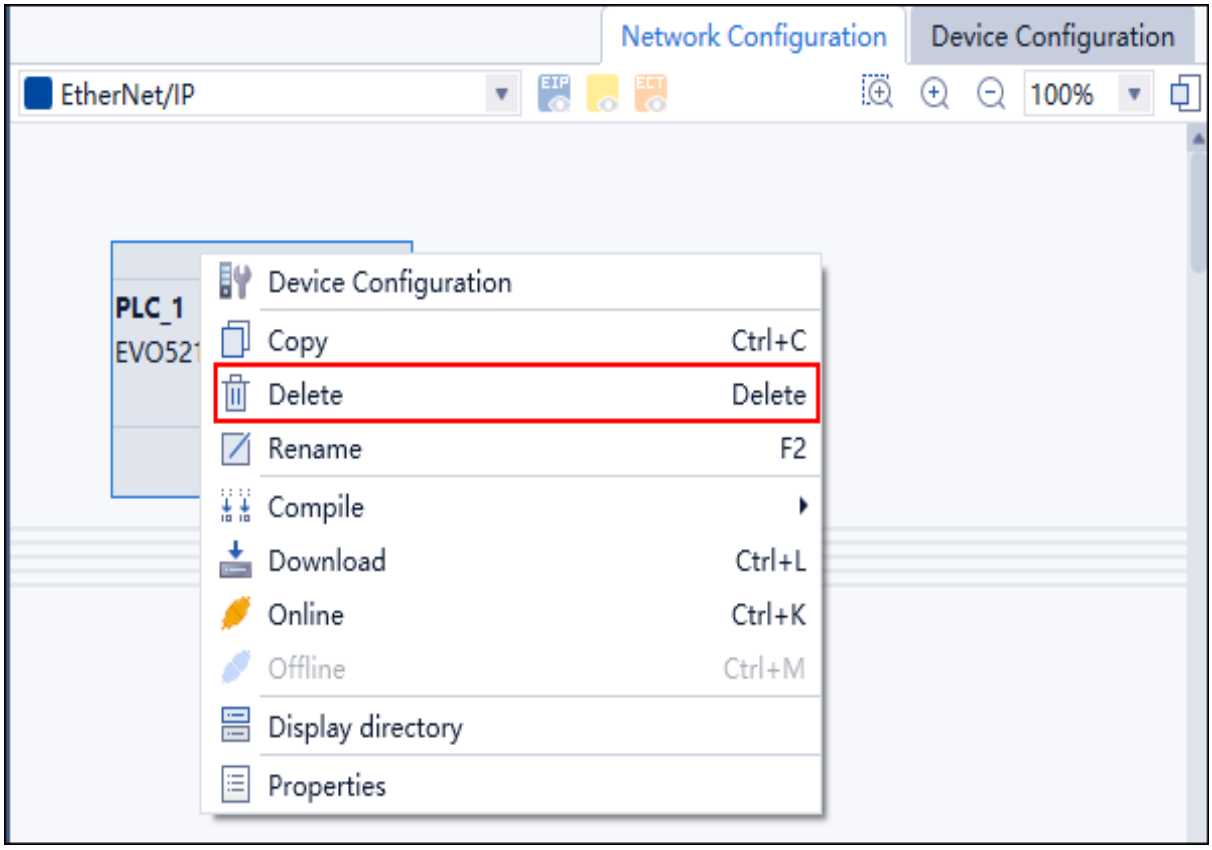
Perform one of the following steps to delete an added device.

- Right-click the device node in the left project tree and select "Delete".



- In the Network Configuration or Device Configuration view, select the device, right-click, and choose "Delete".

  Note: The PLC cannot be deleted in the Device Configuration view.



CAUTION

Deleting a PLC device takes effect immediately and cannot be undone!

7.2.5 Replacing a Device

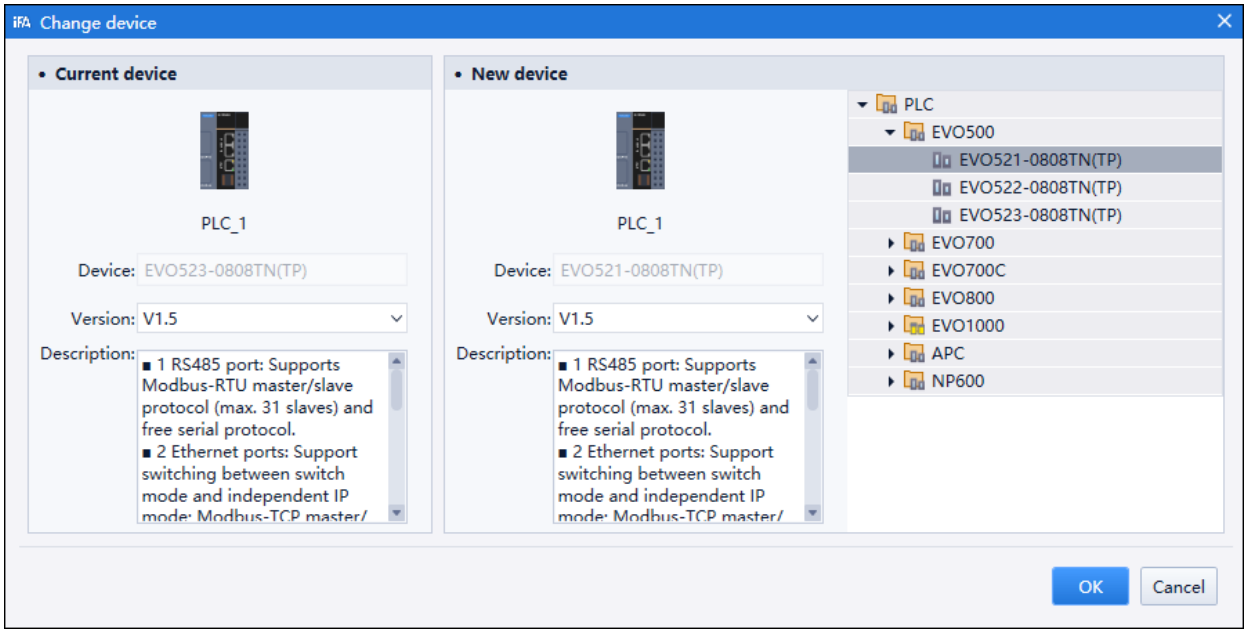
Replacing a device applies to replacing a created device to another one. This operation is allowed only when the device node is in offline mode.

Prerequisites

- A project has been created and opened.
- A device has been added to the project.

Procedure

1. Open the Change Device tab.
   Right-click the device in the project tree and select "Change device".
2. Select the target device from the list on the right.
3. Select the firmware version of the new device following the "Version" field.
4. Click "OK" to replace the device.



Compatibility rules for replacing different devices

- Rule 1: When the new device does not support certain configuration data from the old device, it will discard such data after the replacement.

  Example 1: The EtherCAT communication and slave have been configured for PLC 1. If PLC 2 does not support EtherCAT communication, it will discard all EtherCAT configuration data from PLC 1 after the replacement.

  Example 2: Some basic elements and special controls have been configured on the HMI 1 screen. If HMI 2 does not support certain special controls, it will discard such controls from HMI 1.

- Rule 2: When the new device supports only part of the configuration data from the old device, it will retain such configuration data after the replacement.

  Example 1: All parameters and I/O variable mappings for 8 channels have been configured for GL20-0800END. When GL20-0800END is replaced by GL20-1600END, GL20-1600END will retain all 8 channels of data from GL20-0800END and map them channel by channel. The additional 8 channels will be set to default values.

- Rule 3: When the new device fully supports the configuration data from the old device, it will retain all configuration data after the replacement.

  Example 1: When PLC 1 is replaced by PLC 2 within the same series, PLC 2 will retain all variables and programs from PLC 1.

  Note: This requires that PLC 2 and PLC 1 support the same variable data types, POU types, and POU programming languages.

- Rule 4: When the firmware version is switched from a lower version to a higher version for the same device, Rule 3 applies. When the firmware version is switched from a higher version to a lower version, Rule 1 or Rule 2 applies.

The following table shows the PLC model replacement relationships:

Product Series	EVO52X	EVO76X	EVO8XX	EVO1000EF
EVO52X	√	√	√	×
EVO76X	√	√	√	×

Product Series	EVO52X	EVO76X	EVO8XX	EVO1000EF
EVO8XX	√	√	√	×
EVO1000EF	×	×	×	√

Note: √ indicates that product replacement is supported; x indicates that product replacement is not supported.

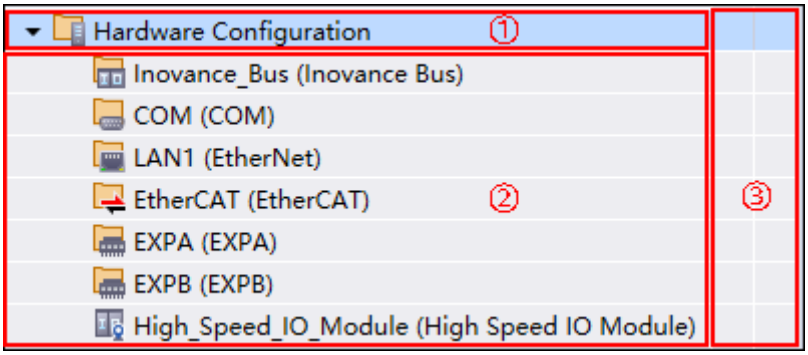
The following table shows the HMI/PC product replacement relationships:

Product Series	IT9000	PC Station (HMI)
IT9000	√	×
PC Station (HMI)	×	√

7.3 Configuring a Network

The communication protocols supported by the PLC model are displayed under "Hardware Configuration". These protocols can be used to meet different functional requirements by expanding the PLC. To communicate with or use different external devices, you can use the hardware configuration function in the project tree to perform hardware configuration and communication protocol configuration.

Taking the EVO500 series as an example, commonly used protocols include Inovance_Bus, Modbus protocol via COM port, EtherCAT protocol, EtherNet protocol based on TCP/IP, Modbus TCP protocol based on TCP/IP, EXPA and EXPB peripheral expansion ports, and high-speed counter modules. The configurable protocols depend on the actual PLC model. The hardware configuration interface and supported protocols are shown in the following figure:



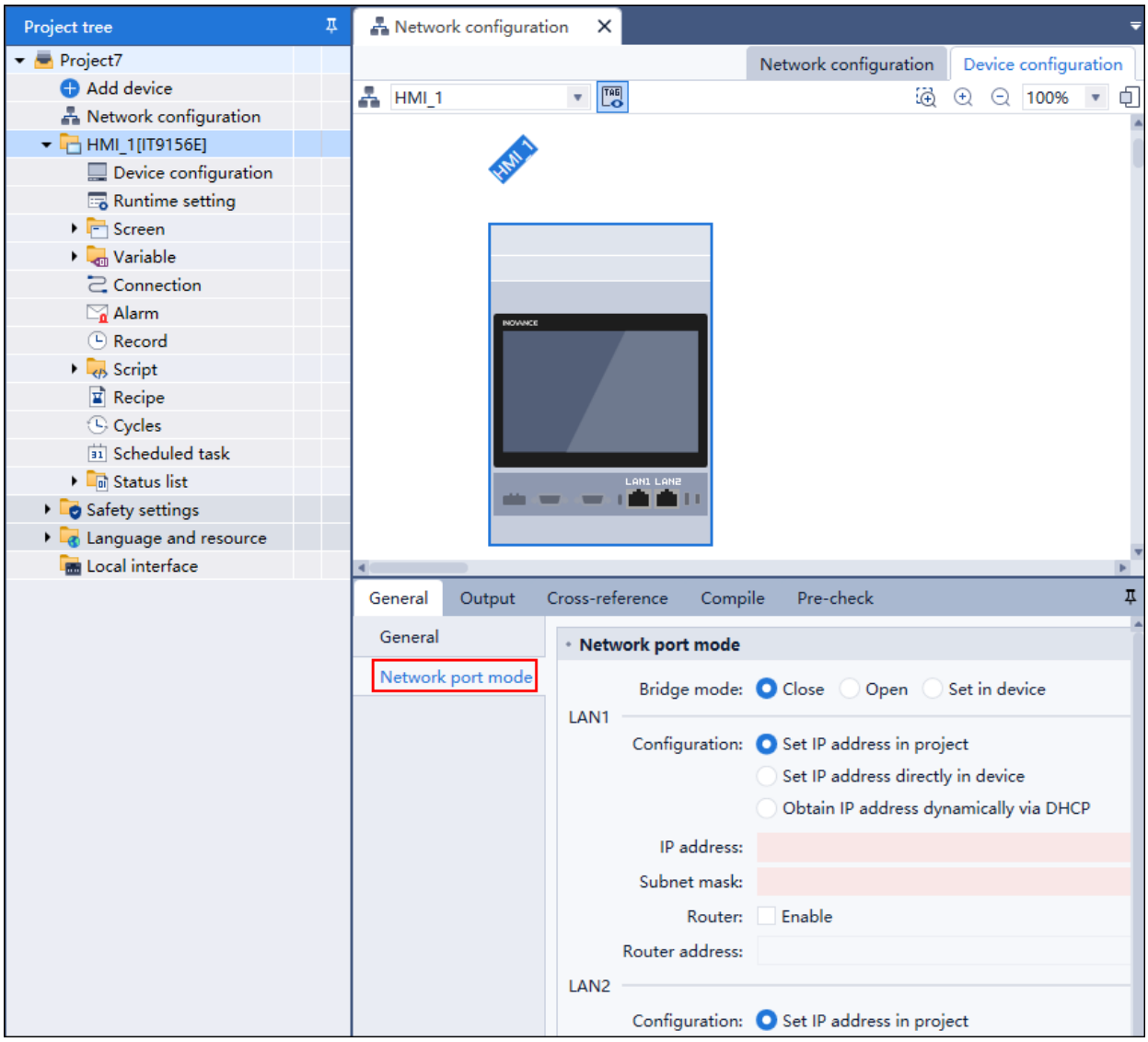
No.	Function	Description
①	Title bar	Using the button on the left side of the hardware configuration title bar, you can manually collapse or expand the hardware configuration project tree.
②	Supported protocols	Under the hardware configuration list of the selected device, the protocols supported by the device are displayed by default.

No.	Function	Description
③	Status bar	Hardware status display bar.

7.3.1 HMI Ethernet Settings

Before downloading a project, set the network port mode of the HMI device according to the network deployment.

Double-click the "Device configuration" node under the HMI device, and in the opened window, click the device image:



Category	Property	Description
Bridge mode	Close	In non-bridge mode, each network port operates independently and is connected to a separate network. This mode is suitable for scenarios requiring simultaneous connection to multiple networks.

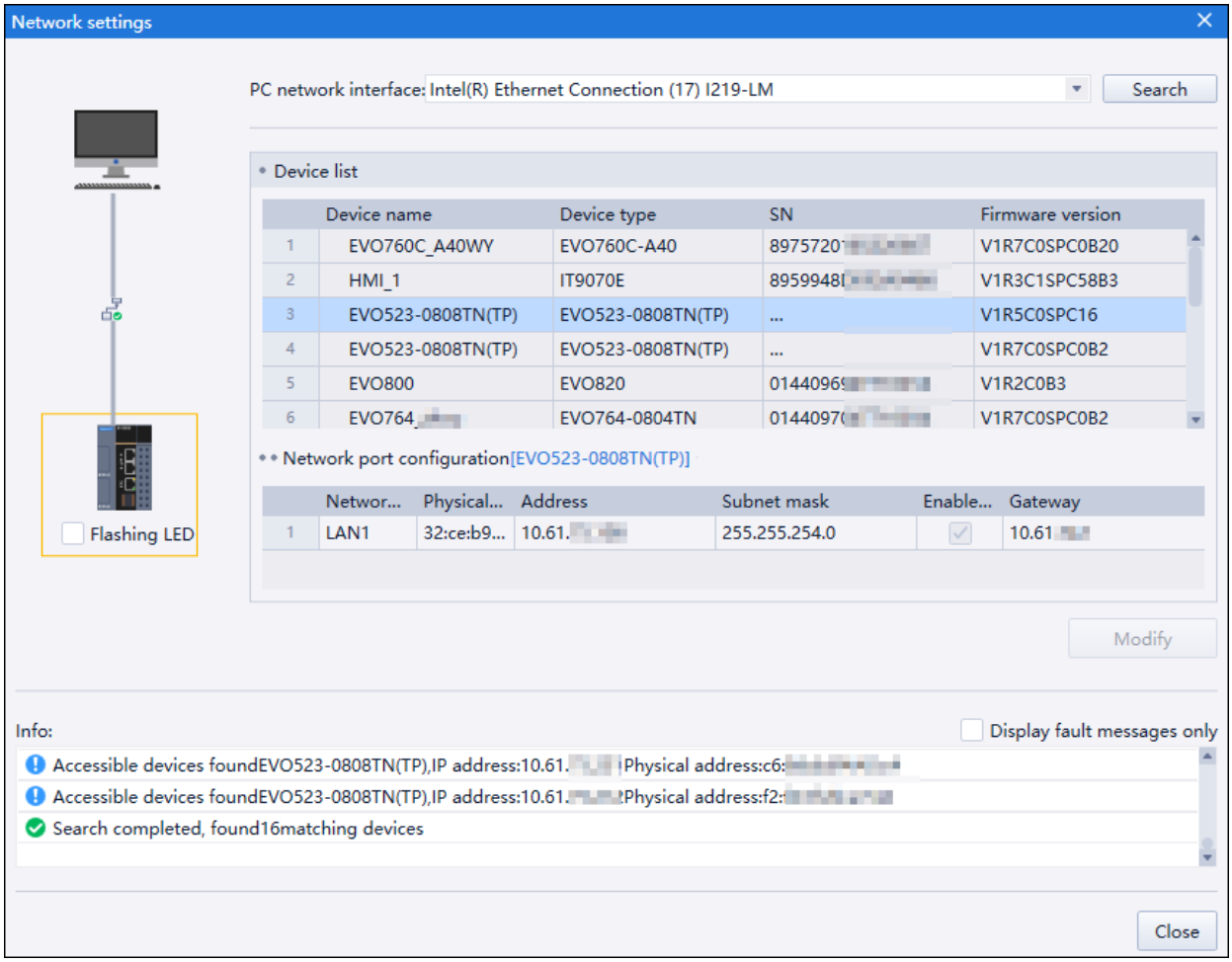
Category	Property	Description
	Open	In bridge mode, the HMI device combines two network ports into a single logical interface, and the IP address can be set by configuring LAN1. In this way, the devices connected to LAN1 and LAN2 can communicate within the same local area network. For example, when two network ports of the HMI device are connected to Device A and Device B respectively, Device A and Device B can exchange data.
	Set in device	Set whether to enable the bridge mode in the control panel of the HMI device.
LAN1/LAN2	Configuration	Methods for configuring the HMI device IP address: • Set IP address in project: Configure the IP address in the current window, and then download the project to the device to make the address effective. • Set IP address directly on device: Configure the IP address on the control panel of the HMI device and apply it. • Obtain IP address dynamically via DHCP: The device acquires an IP address dynamically from a DHCP server. Ensure that a properly configured DHCP server is running in the network to assign IP addresses.
	IP address	Access information for LAN1/LAN2 network card. For example, the LAN1 IP address is 192.168.0.100, and the subnet mask is 255.255.255.0.
	Subnet mask	
	Router	Enter gateway access information, and select or fill in the fields as required based on the network deployment.
	Router address	

7.3.2 Online Network Settings

This section use the EVO500 series PLC as an example.

1. Choose "Online -> Network Settings" from the menu bar to open the Network Settings page.

2. Click "Search" to view all devices on the same or different network segments.



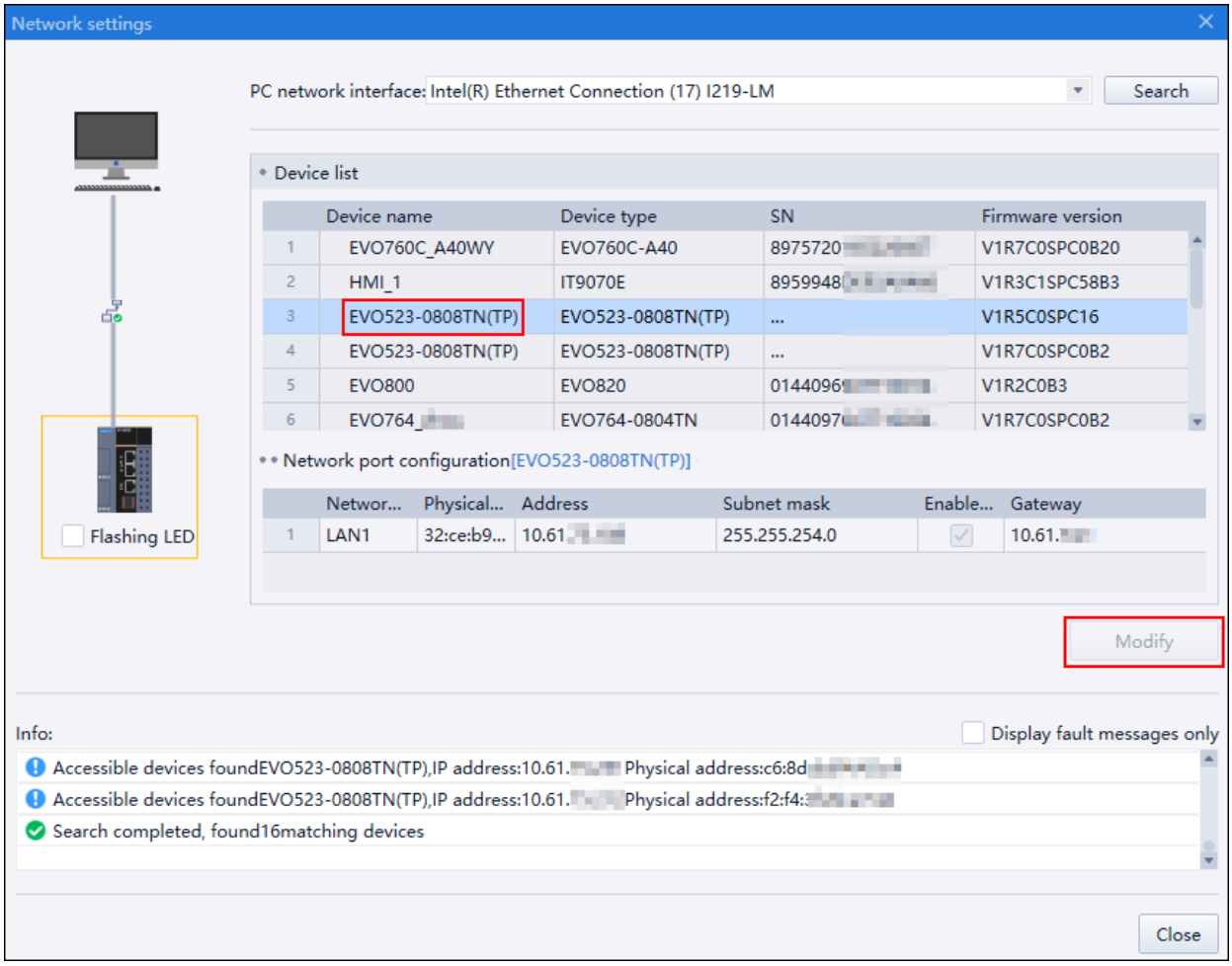
3. Confirm the devices

   Confirm that the device selected online matches the target device in the physical environment to avoid connection errors. Two confirmation methods are provided below:

   - SN consistency check: Check for consistency between the SN in the window and the SN on the physical device.
   - LED flashing check: While the "Flashing LED" option is checked in the window, the **RUN** indication on the physical device will flash in response.

4. Edit the device information

   In the device list of the current window, double-click a row to modify the device name, address, and gateway, and then click the "Modify" button to apply the changes.



The information bar displays the changes and operation result.

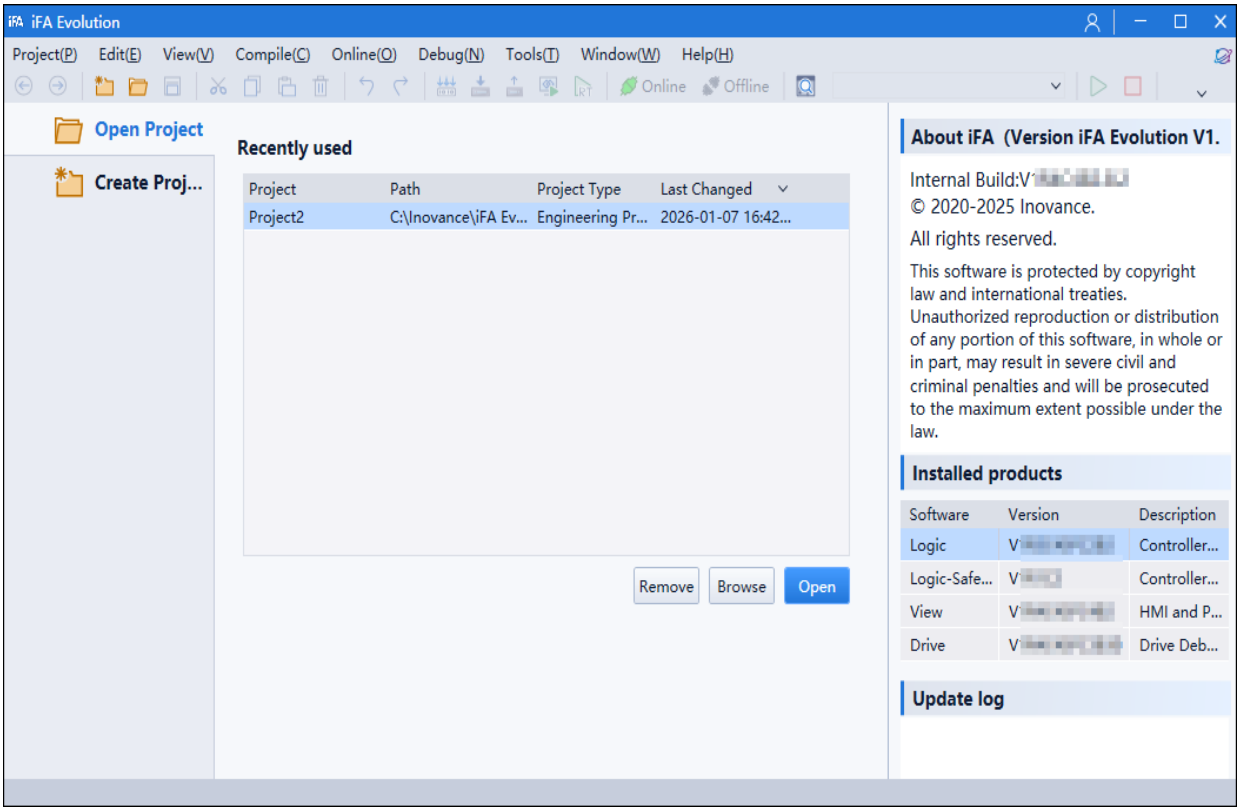
8 Project Management

8.1 Opening a Project

With this feature, you can directly open a target project from existing local projects.

You can open a project in the following ways:

- On the startup interface, select the "Open project" tab, select a project, and then click "Open". The effect is shown below:



- On the system interface, choose "Project -> Open" from the menu bar.
- On the system interface, click the Open icon on the toolbar.
- On the system interface, use the shortcut Ctrl+O.
- Double-click or right-click the project file.

CAUTION

- The same project cannot be opened repeatedly.
- Projects that do not exist or are corrupted cannot be opened.
- Opening a project from the system interface closes the current project and opens the new one.

8.2 Saving a Project

With this feature, you can save the active project data in the original project path and update the last saved time of the project. The current workspace keeps unchanged after the saving is complete.

A project can be saved in the following ways:

- Choose "Project -> Save" from the menu bar.
- Click the Save Project icon on the toolbar.
- Use the shortcut Ctrl+S.

> CAUTION
>
> A project cannot be saved without any changes.

8.3 Saving a Project As

With this feature, you can save the active project as a new project by specifying a new name and save path. After the saving is complete, the active project is switched to the newly saved project.

1. You can open the "Save as" dialog in the following two ways:
   - On the system interface, choose "Project -> Save as" from the menu bar.
   - Use the shortcut Ctrl+Shift+S.
2. In the "Save as" dialog, select the save path, enter the project name, and then click "Save".
3. After successful saving, the active project is switched to the newly saved project.

> CAUTION
>
> If any functions in the project cannot be closed, attempting "Save as" will trigger an error message and prevent the save operation.

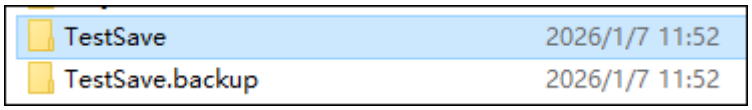
8.4 Backing up a Project

iFA supports automatic backup of project data. You can restore a project to the "historical" version by opening the automatically backed-up project file.

Project backup is triggered automatically when the "Save" option is selected.

The backed-up project is named as xxx.backup and stored in a separate project folder (xxx is the original project name).

For example: If the original project is TestSave, the backed-up project is TestSave.backup. The original and backed-up project are stored in the same directory.

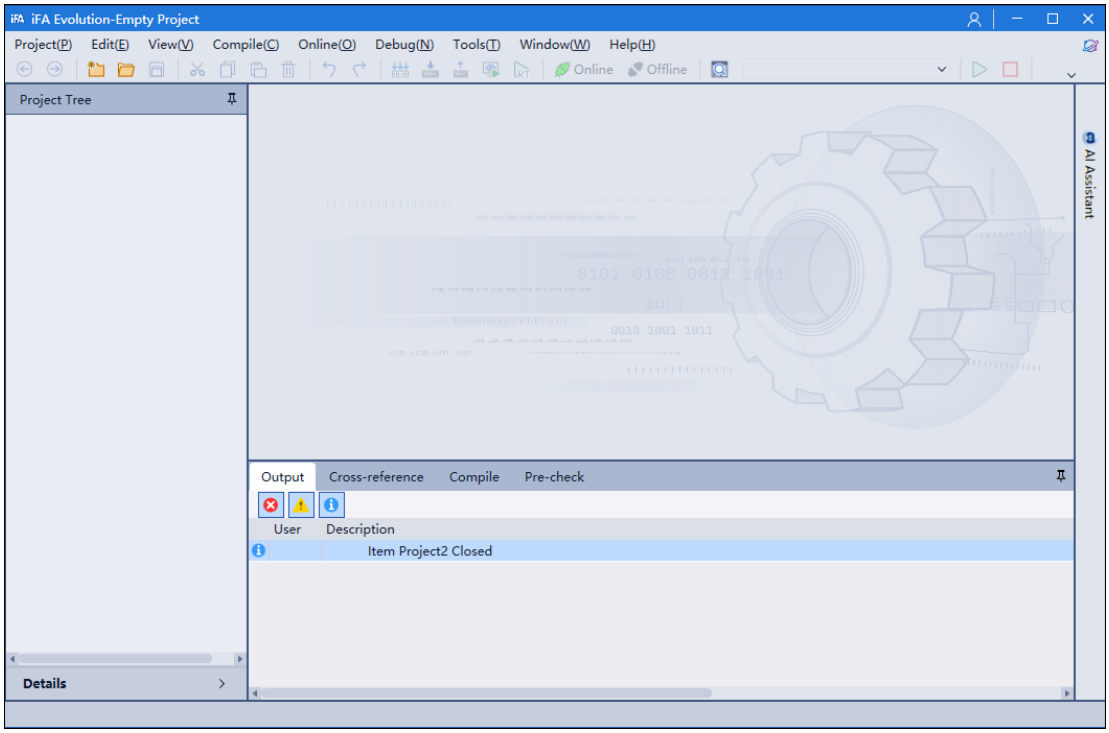


8.5 Closing a Project

With this feature, you can save and close the active project, and restore the system to an empty project state.

Choose "Project -> Close" from the system menu bar to close a project.

After the project is closed, the iFA interface appears as follows:



CAUTION

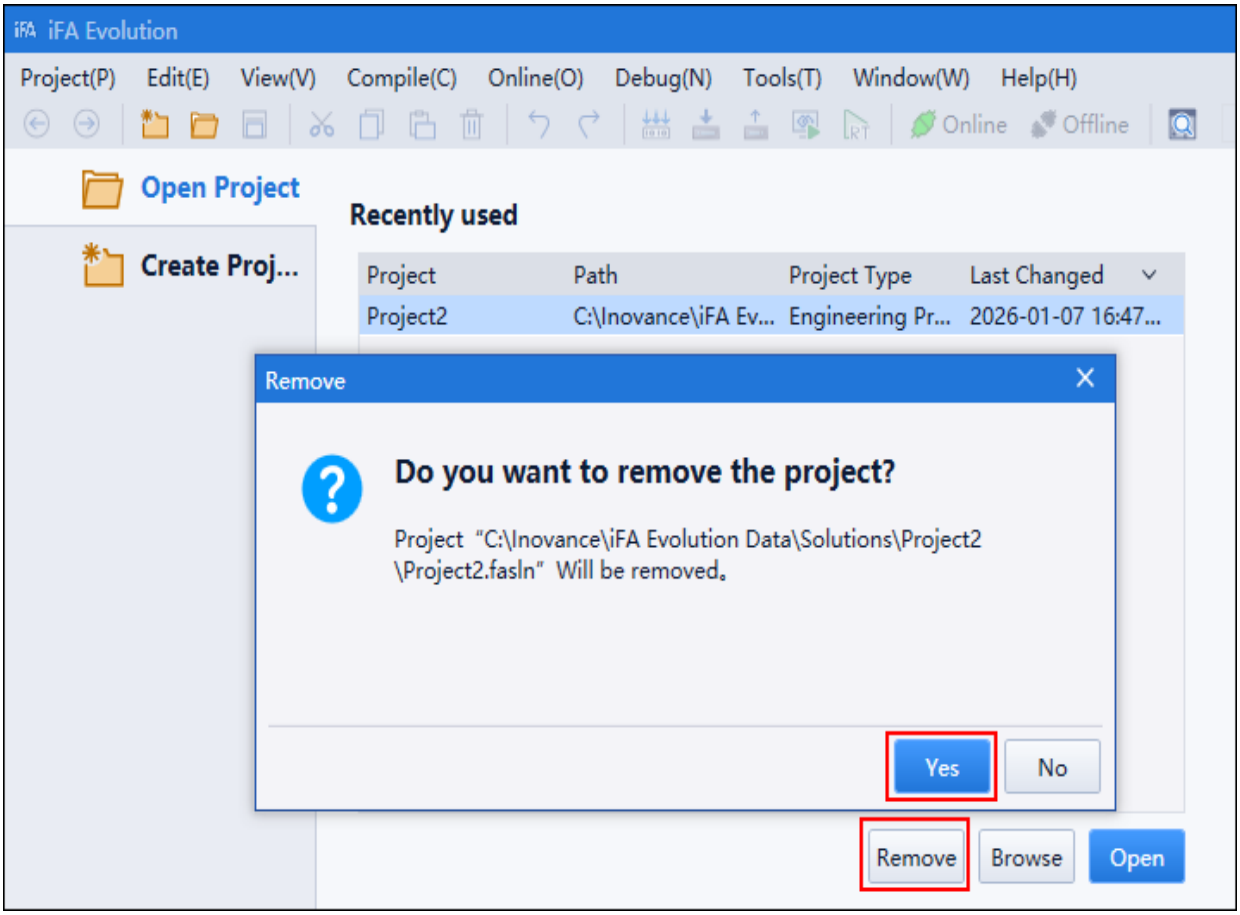
- If a project has not been saved, closing the project will prompt whether to save the current project.
- If any functions in the project cannot be closed, closing the project will trigger an error message and prevent the close operation.

8.6 Removing a Project

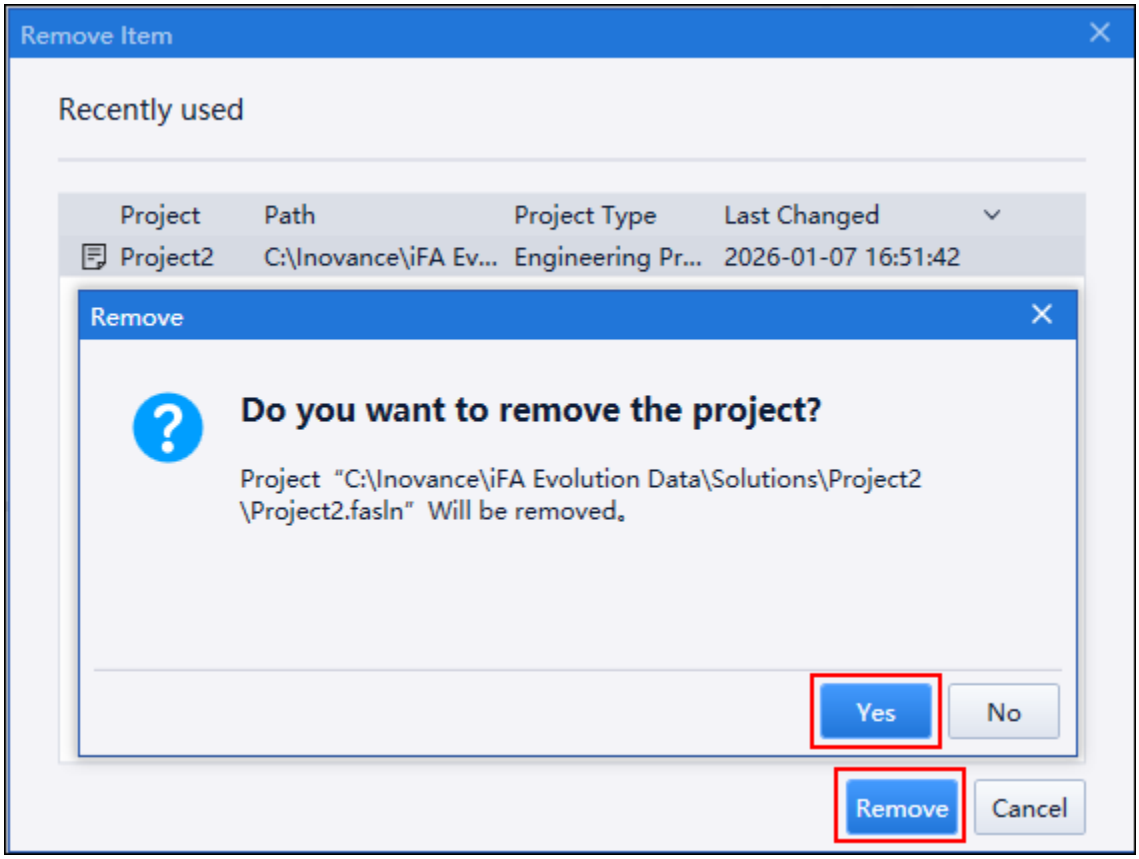
With this feature, you can remove a project from the list of recently used projects, which will not delete the project data file.

Use one of the following methods to remove a project:

- On the startup interface, select the "Open project" tab, choose the desired project, click "Remove," and then click "Yes" in the pop-up window. The effect is shown below:



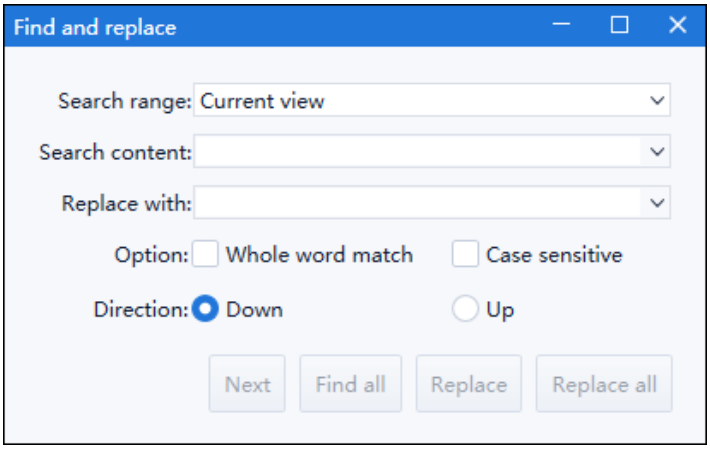
- Click "Remove" from the "Project" menu bar, access the Remove Project interface, select the desired project, click "Remove," and then click "Yes" in the pop-up window. The effect is shown below:



8.7 Finding and Replacing Within a Project

With this feature, you can search for and replace editable information in a project. Proceed as follows to meet specific requirements:

- Optimize the search by adding options.
- Specify the search area.
- Specify the search direction.



Function		Description
Scope	Current view	The page currently displayed in the workspace.
	Current device	Applies to PLCs only. • If a node in the project tree is selected, it refers to the device associated with the selected node. • If a view in the workspace is opened, it refers to the device associated with the opened view.
	Current device and source code library	If the source code libraries are already installed, search can be performed in the source code libraries and the current device simultaneously.
	Opened views	All views opened in the workspace.
	Selection	Applies to HMI devices only. This option is valid when the HMI screen is the current page. The Find and Replace operations are limited to the selected area.
Find		Enter up to 32 keyword characters for search.
Replace with		Enter up to 32 keyword characters for replacement.
Options	Case sensitive	• When this option is selected, the search mode becomes case-sensitive; • When not selected, the search mode ignores case differences, and this is the default state.

Function		Description
	Match whole word	• When this option is selected, the search mode is whole word matching; • When not selected, the search mode is fuzzy matching, and this is the default state.
	Forward	When this option is selected, the search starts from the current cursor position or selected location and proceeds forward.
	Backward	When this option is selected, the search starts from the current cursor position or selected location and proceeds backward. By default, the search starts backward.
Next		Click the "Next" button to perform the search and highlight the result. Click this button again to continue searching according to the specified conditions.
Find all		Click the "Find all" button to display all matching results in the "Find and replace" panel in the monitoring area; double-click a result to directly locate the specific object.
Replace		Click the "Replace" button to perform a single target replacement.
Replace all		Click the "Replace all" button to replace all matching items within the selected scope.

Start search

Follow the following steps to start the "Find and replace" function:

1. Open the "Find and replace" tab in one of the following ways:
   Menu bar: Choose "Edit -> Find and replace" from the menu bar.

   Shortcut: Press Ctrl+F on the keyboard.
2. Enter the search word in the "Find" input box.
3. Select the required option.
4. Select the search direction.
5. Click "Next". At this point, the first result is highlighted in the editor.
6. Click "Next" again to display the next result. Repeat this step as needed until the last result is displayed.
7. Click "Find all" to immediately locate all results.

Replace keywords

You can replace the search results one by one or all found results at once. Proceed as follows to perform replacement:

1. Open the "Find and replace" tab in one of the following ways:
   Menu bar: Choose "Edit -> Find and replace" from the menu bar.

   Shortcut: Press Ctrl+F on the keyboard.
2. Enter the search word in the "Find" input box.

3. Select the required option.
4. Click "Next" to start searching for a specified keyword. At this point, the first result is highlighted in the editor.
5. Enter the text you want to replace with in the "Replace with" input box.
6. Click the "Replace" button to replace a selected result with the specified text. Repeat this step until all desired results have been replaced.
7. Click "Replace all" to automatically replace all results immediately.

8.8 Reuse by Copy and Paste

Device scope

- Master devices: PLC, HMI, PC.
- Slave devices: servo, distributed I/O, I/O coupler.

Naming rules for objects created by copy and paste

- The new object name is the original name with "_x" appended, where x is a number. For details, see the following:

If the original object name does not end with "_x", the pasted new node name will be formed by appending "_x". The value of "x" is determined by searching from 1 until a unique value is found.

Example 1: If only one node is named "nodename", the first new object created by copy and paste will be named "nodename_1".

Example 2: Based on Example 1, the second new object created by copy and paste will be named "nodename_2"; other new objects follow the same pattern.

- If the original object name ends with "_x", the pasted new node name will be also formed by appending "_x". The value of "x" is determined by searching from 1 until a unique value is found.

Example 3: If three nodes of nodename, nodename_1, nodename_2 are present, and copy and paste operations are performed on nodename_1, the pasted new object name will be nodename_3.

Example 4: If two nodes of nodename_2 and nodename_3 are present, and copy andpaste operations are performed on nodename_3, the pasted new object name will be nodename_1; the second new object name will be nodename_4.

Copyable content

- Operation nodes in the project tree cannot be copied.

Example 1: iFA projects: "Add device" and "Device and network".

Example 2: PLC: "Device configuration" and "Online and diagnostics".

- All user-added resource nodes in the project tree can be copied.

Example 3: User-added devices in the project tree, such as PLC and HMI.

Example 4: Resources within devices in the project tree, such as POU and variable tables in the PLC; screens and recipes in the HMI.

Within-project copy and paste

Copy a device

- Copying and pasting a device is supported within a project. The pasted new device has the same configuration data as the original one.
- A device can be copied once and then pasted multiple times.
- Multiple devices can be copied and pasted at a time.
- Paste position.
  - Project tree root node (project name node). New devices will be added at the end of the device list.
  - Any device in the project tree. New devices will be added at the end of the device list.
- Operation areas and methods.
  - Operation areas: project tree, network view.
  - Operation methods: top menu, shortcut menu, shortcuts, toolbar buttons.

Copy resources under a device

- Copy and paste within a device

In the project tree, the sub-resources of a device can be copied and pasted into the same resource area of the current device.

- Copy and paste across devices

In the project tree, the sub-resources of a device can be copied and pasted into the same type of resource area of another device.

  - In case of no name conflict, the same name will be used for the new resource in the target area.
  - In case of a name conflict, a new name will be created for the new resource according to the above naming rules.

- Any copyable object can be copied once and consecutively pasted multiple times.
- Multiple child nodes under the same node can be selected using the Ctrl key for copy and paste operations.
- Exception handling: If multiple objects are copied and then some resources are deleted, a prompt box will appear stating that "Some objects have been deleted. Do you want to continue pasting the remaining objects?"

Options: Yes or No. Yes: Paste the remaining objects; No: Cancel the paste. The prompt appears only once, irrespective of the number of deleted objects.

- Folders under a device can be copied and pasted. All resources under the folders can be also copied and pasted. The position restriction rules for resources under folders also apply to the pastable positions.
- If the read-only access is granted for a resource in the project, the resource cannot be copied.
- Paste positions.
  - Select a parent node of the target device for pasting. The new resource can be pasted under the parent node or under a folder within the parent node.
  - If a resource with the same name already exists under the parent node of the target device, the newly pasted object will be renamed.
  - After the pasting is complete, the resource under the parent node is expanded automatically.

- Operation area and methods
  - Operation area: project tree.
  - Operation methods: top menu, shortcut menu, shortcuts, toolbar buttons.

Copy and paste permissions

If the project-specific permissions have been configured for the current project, only users with write permissions can perform paste operations.

Across-project copy and paste

Open different projects in two iFA software in the same PC, and then perform copy and paste operations between the projects.

Copy an entire device

A device within a project can be copied and pasted into another project.

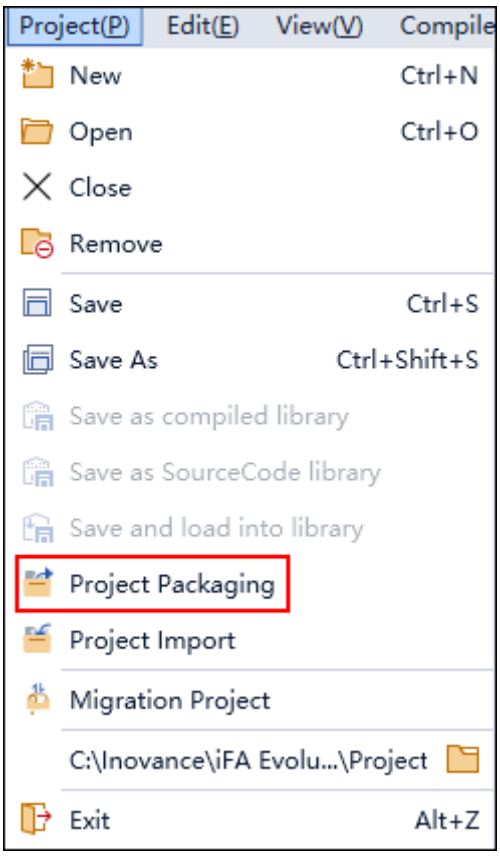
Copy resources within a project

Child node resources under a device of a project can be copied and pasted into a resource node of the same type under a device in another project.

8.9 Packaging a Project

With this feature, you can save the active project in the current software to the local disk in a file (.slnpkg format) by specifying a new save path. The file name is the project name plus a timestamp.

When selecting "Project packaging" for a modified but not saved project, a prompt will appear asking whether to save the current project changes. You can select an option as needed.



Select the save path for a packaged project, and handle it follows when saving the project:

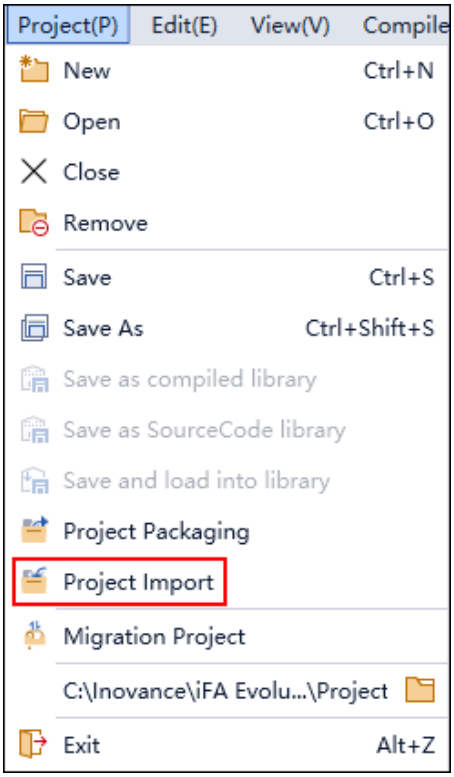
- If no file with the same name exists in the project path, the corresponding directory will be automatically created;
- If a file with the same name exists in the project path, you can choose to overwrite it or create a new file name.

> NOTE
>
> Project packaging does not affect the currently opened projects.

8.10 Importing a Project

With this feature, you can import the packaged project file into a project in the software.

Select "Project import" from the "Project" menu bar, double-click to select the .slnpkg file, and then import the file into the software.



8.11 Upgrading a Project

Projects created in the old versions (V1.3 and earlier) of the iFA Evolution cannot be opened before the software is upgraded to a later version.

You can see the system prompt for project upgrade when opening a project containing HMI products. You can directly open a project not containing HMI products.

Projects requiring upgrade will be upgraded under a new path, without any impact on the original project.

Upgrading a protected project

If you need to upgrade a project created in an earlier product version and protected by administrator permissions, the upgrade can be performed directly without permission restrictions, and the original project permissions are retained after the upgrade.
Operations on an upgraded protected project are still subject to permissions configured for the original project.

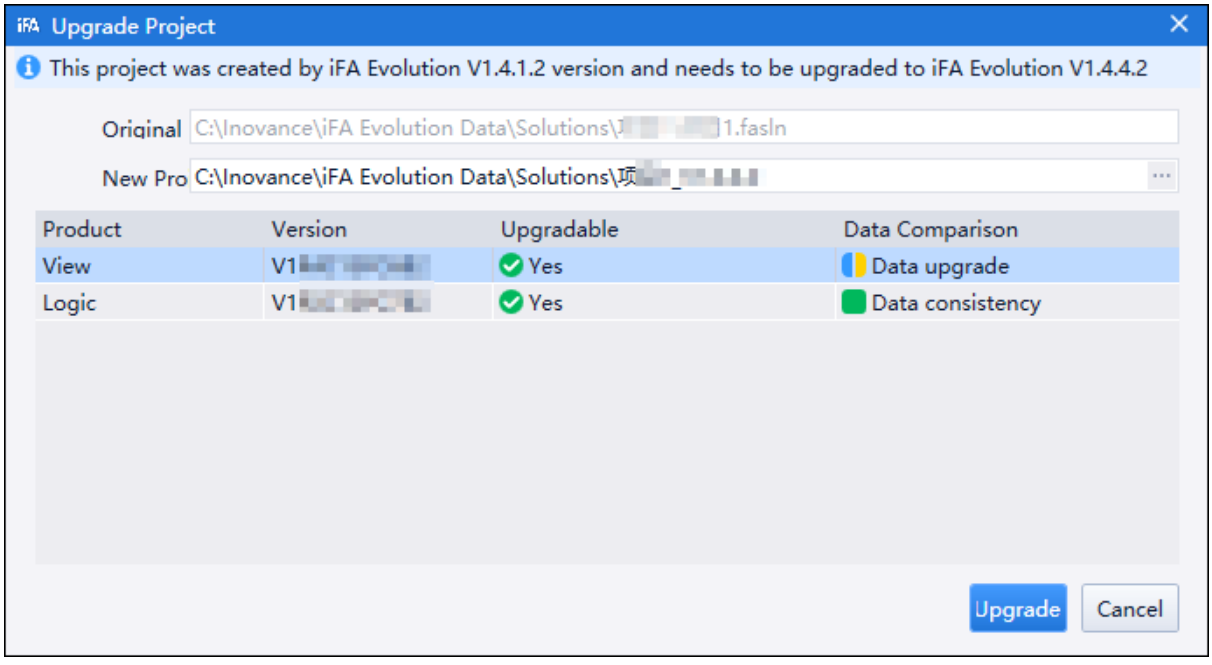
Upgrade method

Prerequisites

- All components in the original project have been installed;
- The new version software can be opened normally.

Procedure

1. Click the "Open" option from the "Project" menu.
   The "Open project" dialog box appears immediately, displaying a list of recently used projects.
2. Select a project from the list, and then click "Open".
3. If the target project is not in the list, click "Browse". Navigate to the specified project folder, and open the project file.
   The "Open project" dialog box appears immediately.
4. In the project folder, double-click to open the project file.
5. Click "Upgrade" to access the "Upgrade project" dialog box.
   Original Project: Displays the storage path of the original project, which is not editable.
   New Project: Displays the default storage path and new name of the new project, which are editable. The name must comply with the project naming rules.
   Upgrade Information: Displays the information about Product, Version, Upgradable, and Data Comparison.



6. Click "OK".
   The project is upgraded and then the upgraded new project is opened.

8.12 Importing/Exporting a PLCopenXML File

PLCopenXML is used for programming standardization. It supports exporting XML files for editing with third-party tools and re-importing them into the software.

NOTE

- This feature requires software V1.4.2.5 or later and is not currently supported by the Safety PLCs.
- Currently, import or export is not supported in the following scenarios:
  - If there are duplicate names among the GV (global variable table), DUT (user-defined data types), and POU (program organization units), import with overwrite will fail. For example, if the global variable table has the same name as the PRG, the import will fail and the monitoring area will display a warning.
  - The POUs are encrypted.
  - If the PLC type or version is inconsistent, the import of hardware configuration nodes will fail.
  - Only ST (structured text), LD (ladder diagram), and CFC (continuous function chart) languages can be imported and exported, while other languages are not supported yet.

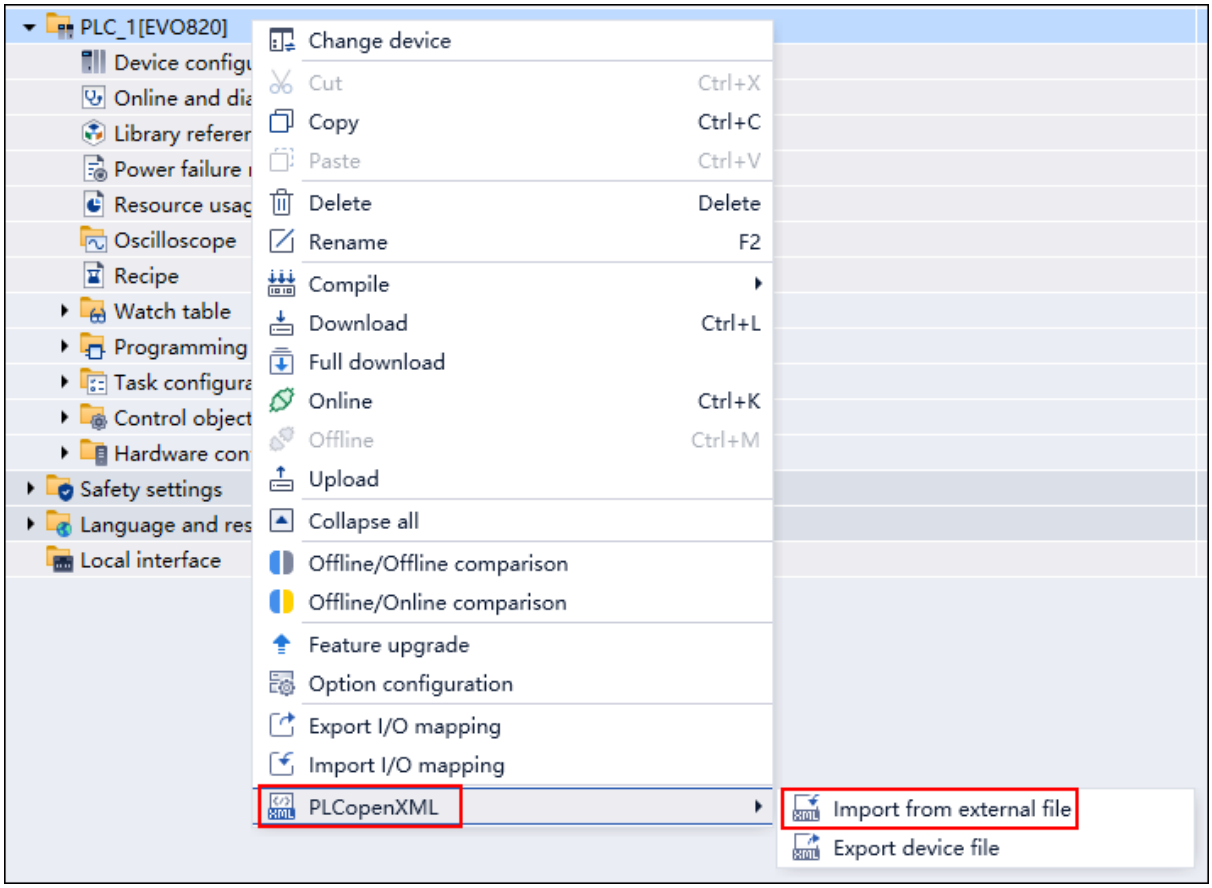
Import a PLCopenXML file

Prerequisites

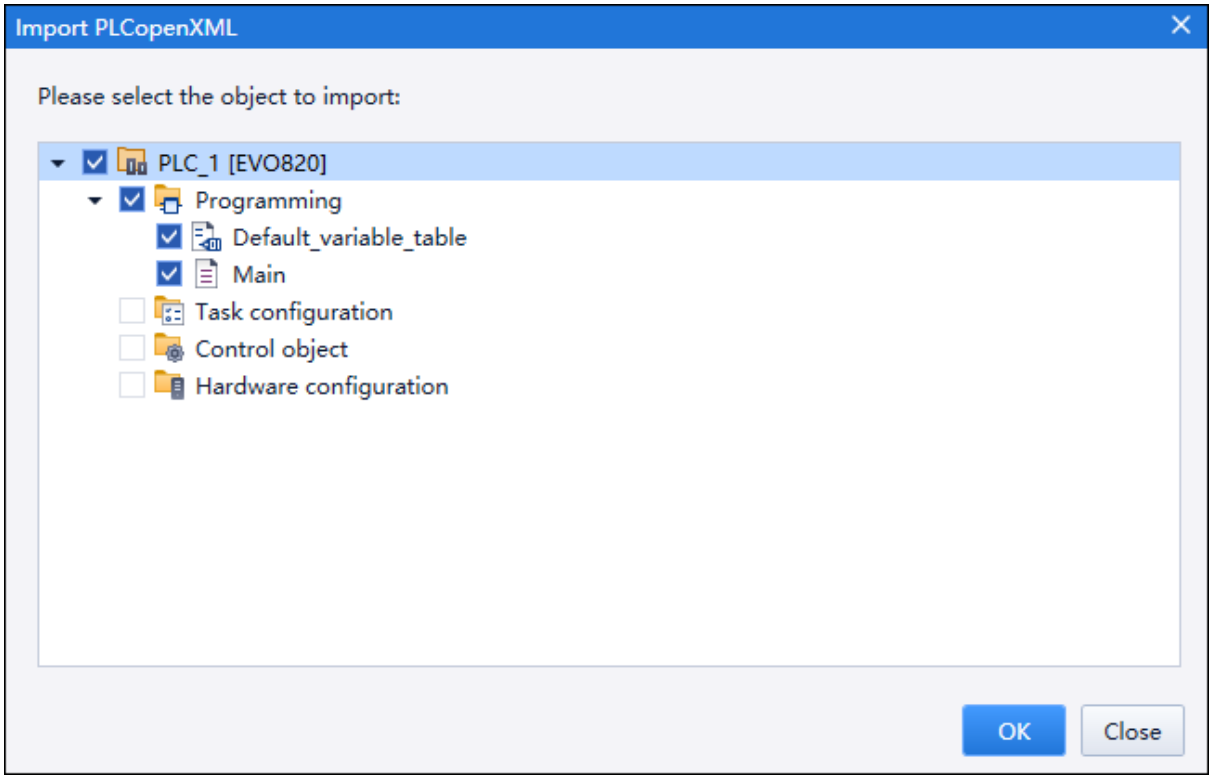
- A PLC has been created.
- The XML file conforms to the PLCopenXML standards.

Procedure

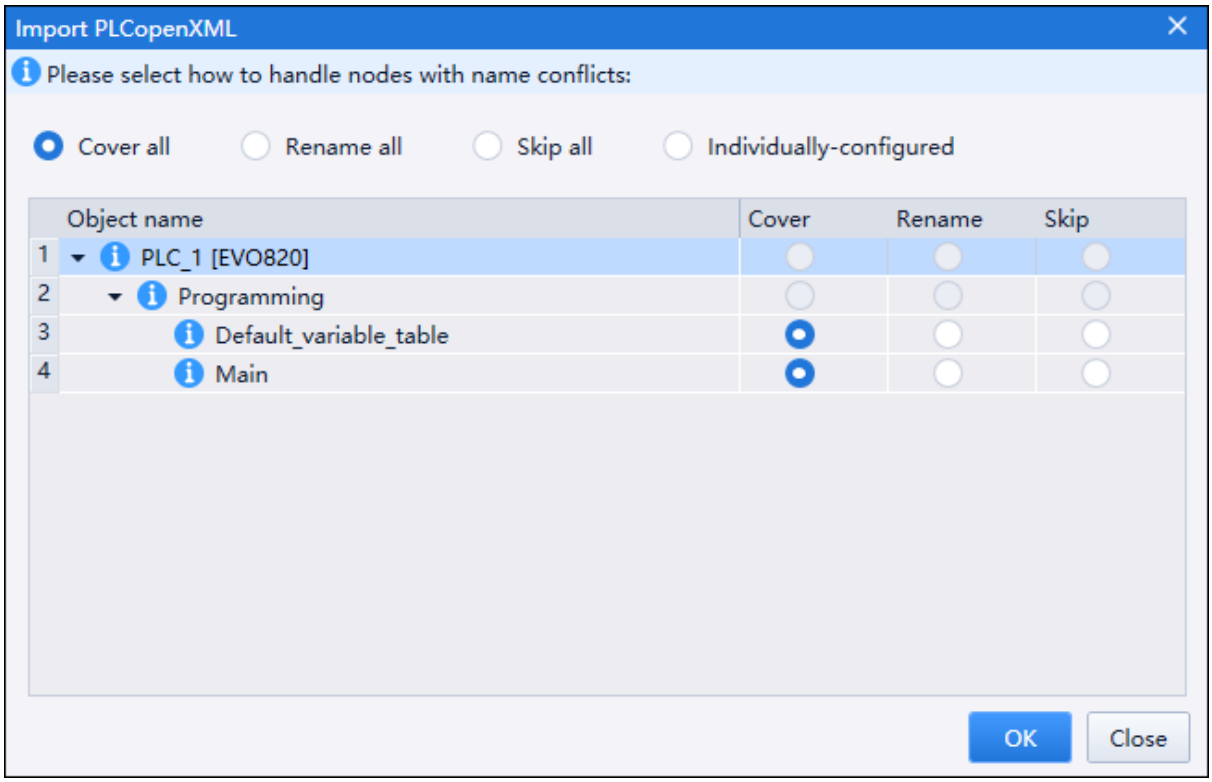
1. Right-click the added PLC in the left project tree and select "PLCopenXML -> Import from external file".



2. In the opened dialog box, select an appropriate XML file, click "Open", and the "Import PLCopenXML" dialog box will appear. Select the target object and click "OK".

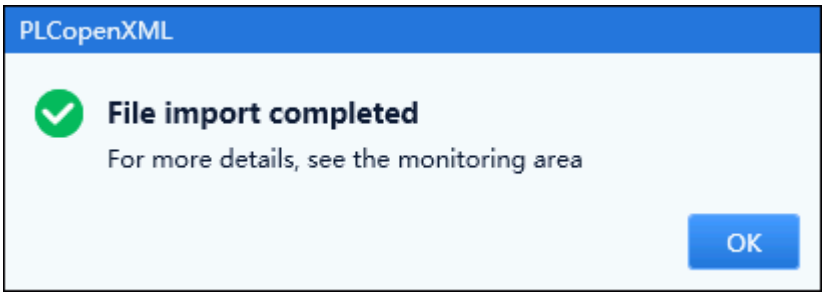


3. After clicking "OK", select how to handle node name conflicts.



- Overwrite all: Overwrite nodes in the current project.
- Rename all: Rename all according to the current renaming rules.
- Skip all: Do not perform import.
- Individually configure: Process the imported list individually.

4. After clicking "OK", the file import is complete, and relevant information is displayed in the monitoring area.



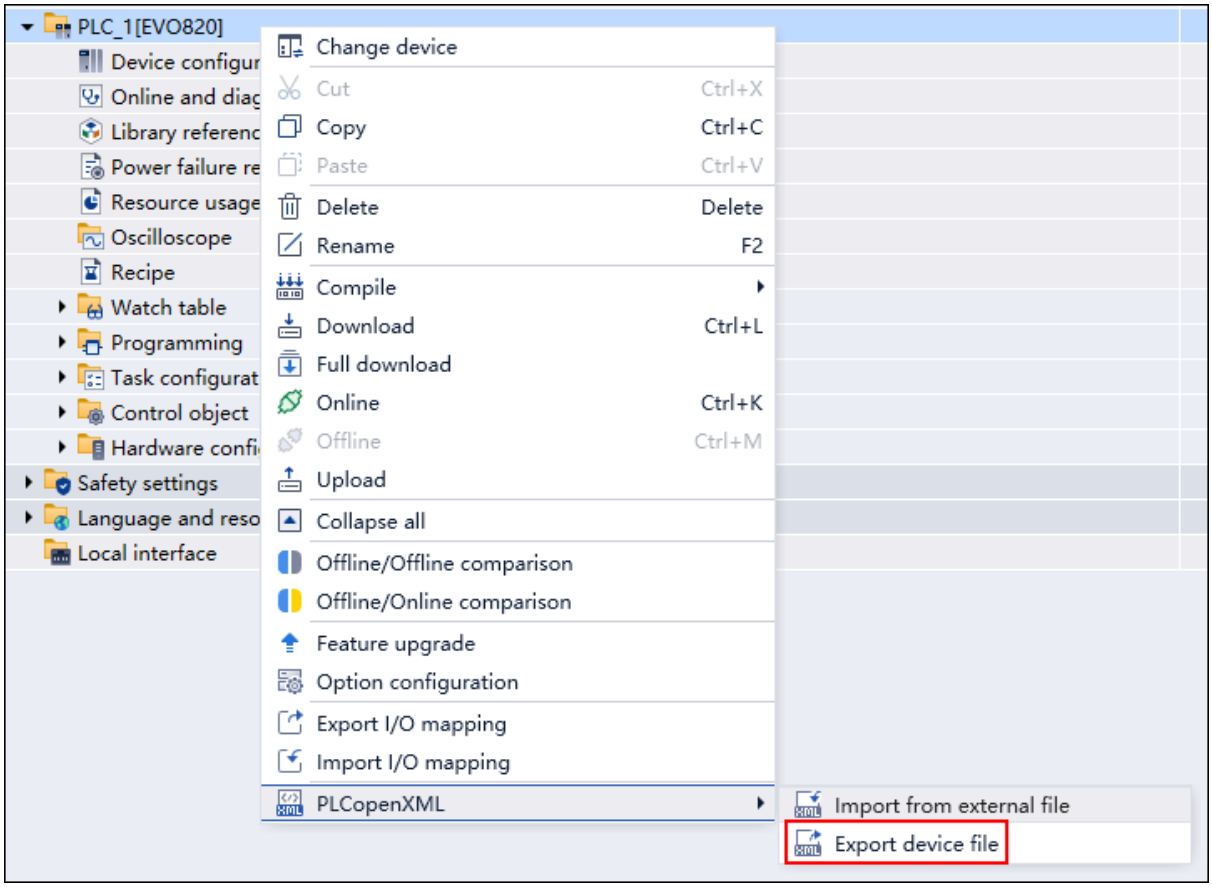
Export a PLCopenXML file

Prerequisites

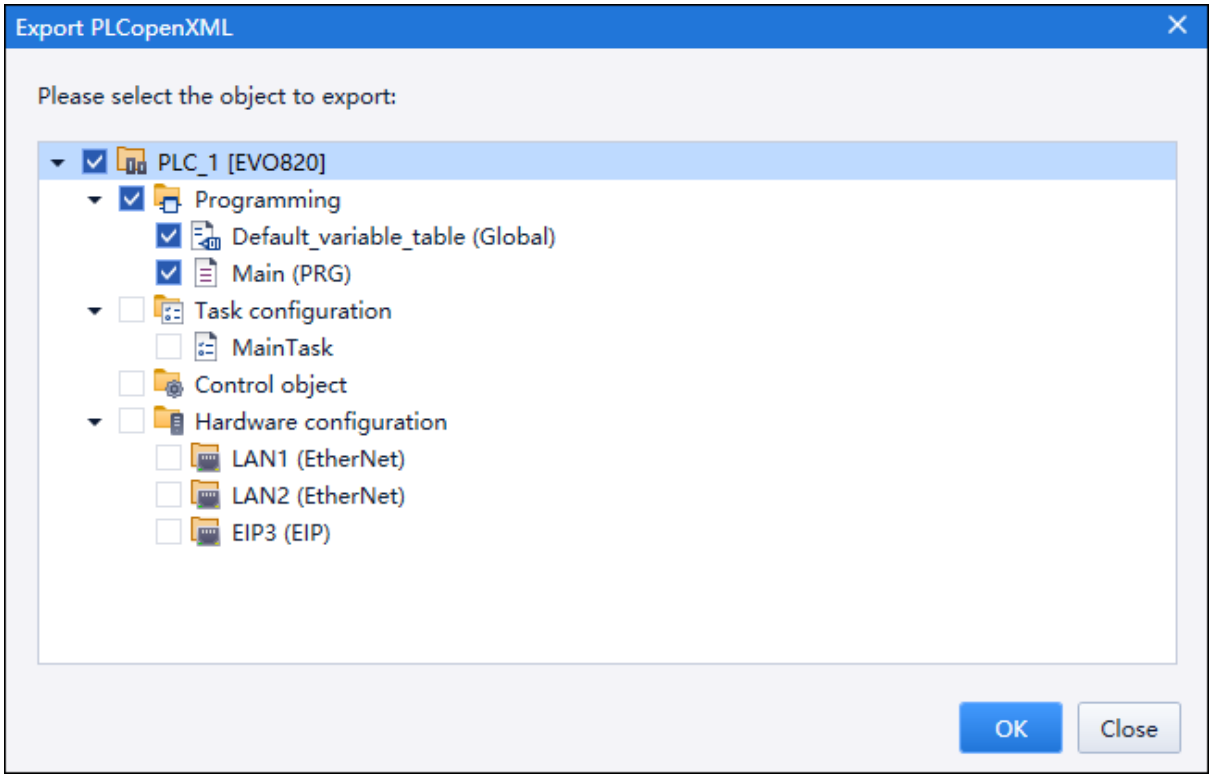
A PLC has been created.

Procedure

1. Right-click the added PLC device in the left project tree and select "PLCopenXML -> Export device file".

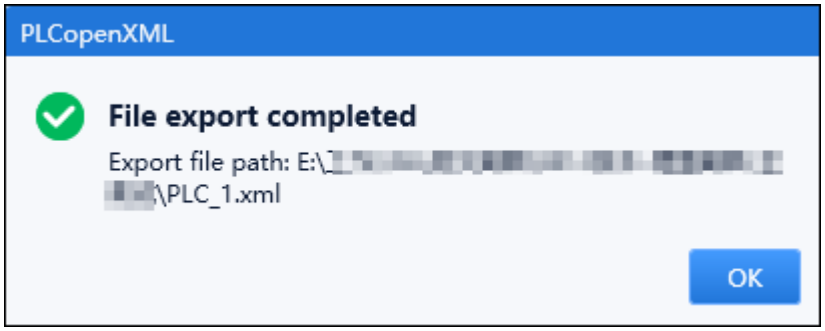


2. In the opened "Export PLCopenXML" dialog box, select the target object and click "OK".



3. In the opened dialog box, select the export path. You can define the file name according to naming conventions. The default file name is "Device Name.xml".

4. Click "OK" to complete the file export. After successful export, the system will display a dialog box showing the path of the exported file.



8.13 Migrating a Project

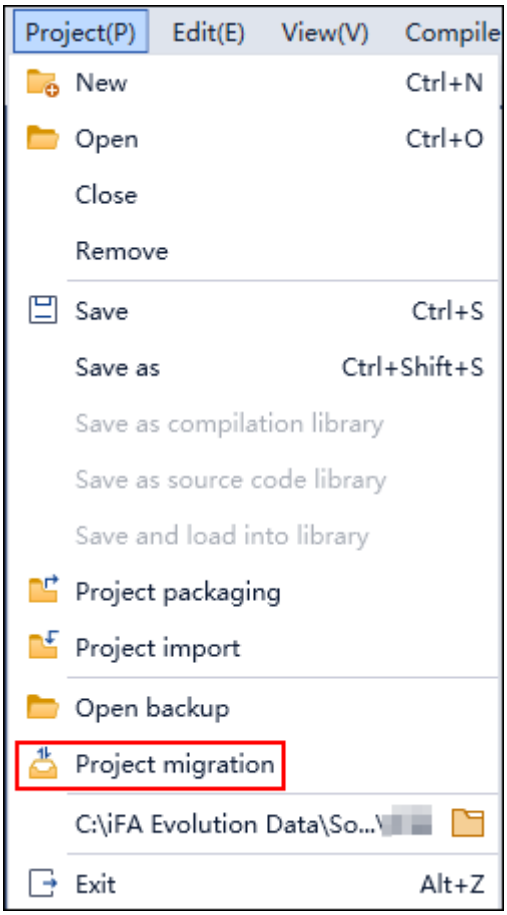
With this feature, you can migrate projects created in the original InoProShop or InoTouchPad software to the iFA.

Prerequisites

- A project has been created in the iFA.
- The project file with an extension of .ino/.itp/.transform is ready.

Procedure

1. Choose "Project > Project migration" without adding any devices.



2. Select the target project file and then click "Open".

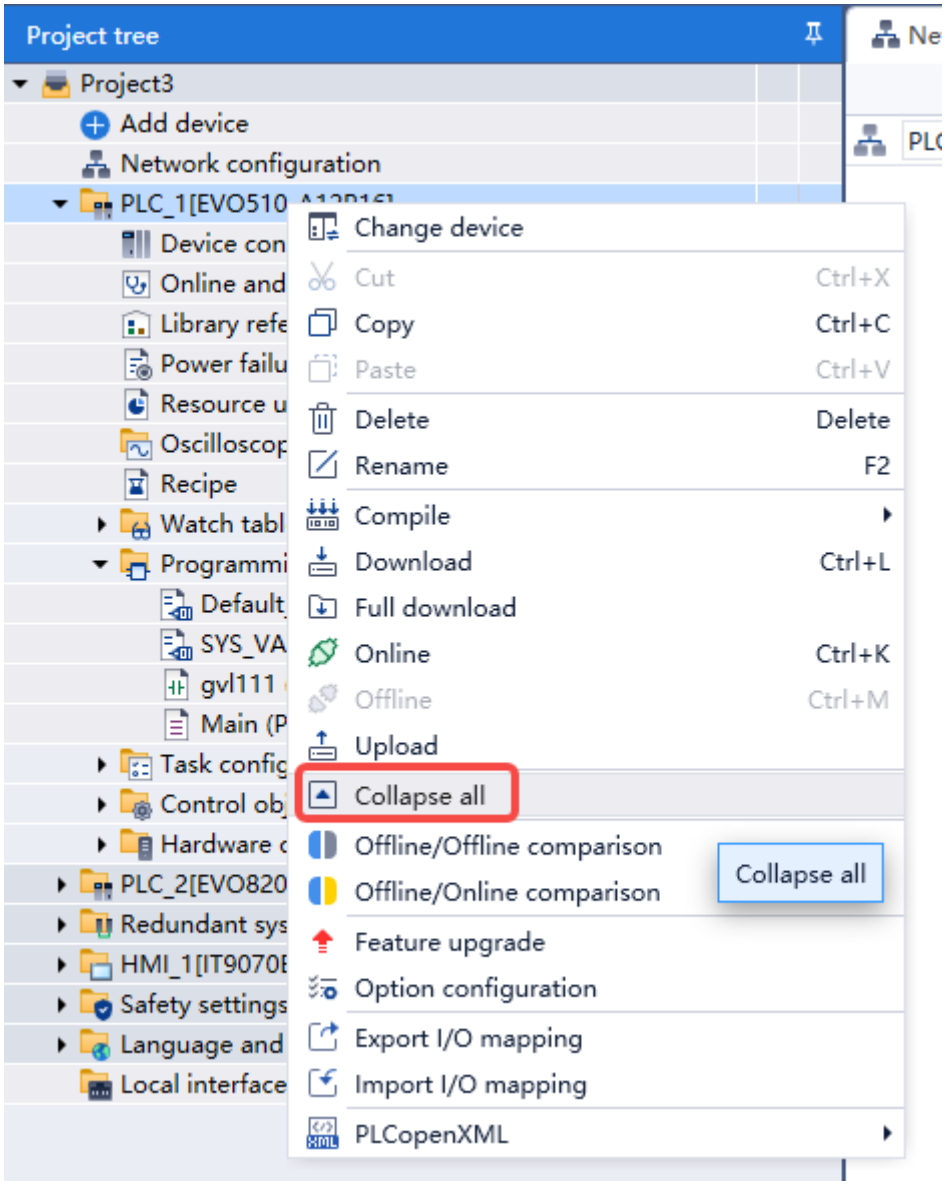
   After successful import, the import result is displayed in the monitoring area.

NOTE

- A project can be imported when multiple PLCs or HMIs are used.
- Repeated import is supported.
- The imported projects are named according to the device names. If the new name duplicates an existing name, it will be sequentially named as Device_1, Device_2, and so on.

8.14 Collapsing Project Child Nodes with One Click

When multiple levels of expanded child nodes exist under a device or folder node, making the list too long to locate items easily, right-click the device/folder node and select "Collapse all" to instantly collapse that node and all its child nodes, restoring the initial state.



8.15 Using the Device Library

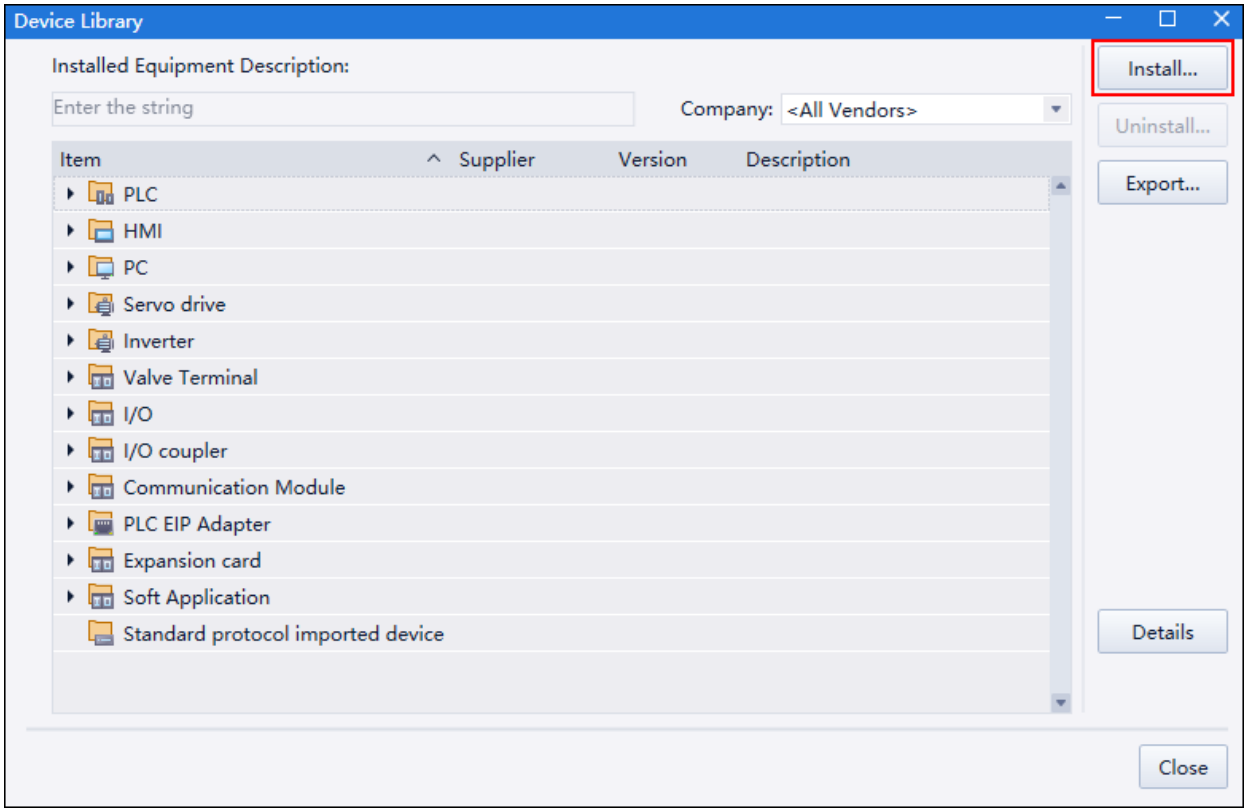
8.15.1 Installing a Device

You can select ZIP, XML, EDS, and DVC files for offline device installation to add devices to the device library.

Procedure

1. Click the "Tools" option in the menu bar, and then select "Device Library" to access the Device Library dialog box.

2. Click "Install Device" in the right-side menu bar.



3. Select a supported file type in the appropriate path, and then click "Open" to import the file. You can also select multiple device description files at a time.

4. After successful import, click "OK."

After the device installation is complete, the newly installed device appears under the "Add Device" node in the project tree and in the "Device List" on the right-side menu bar.

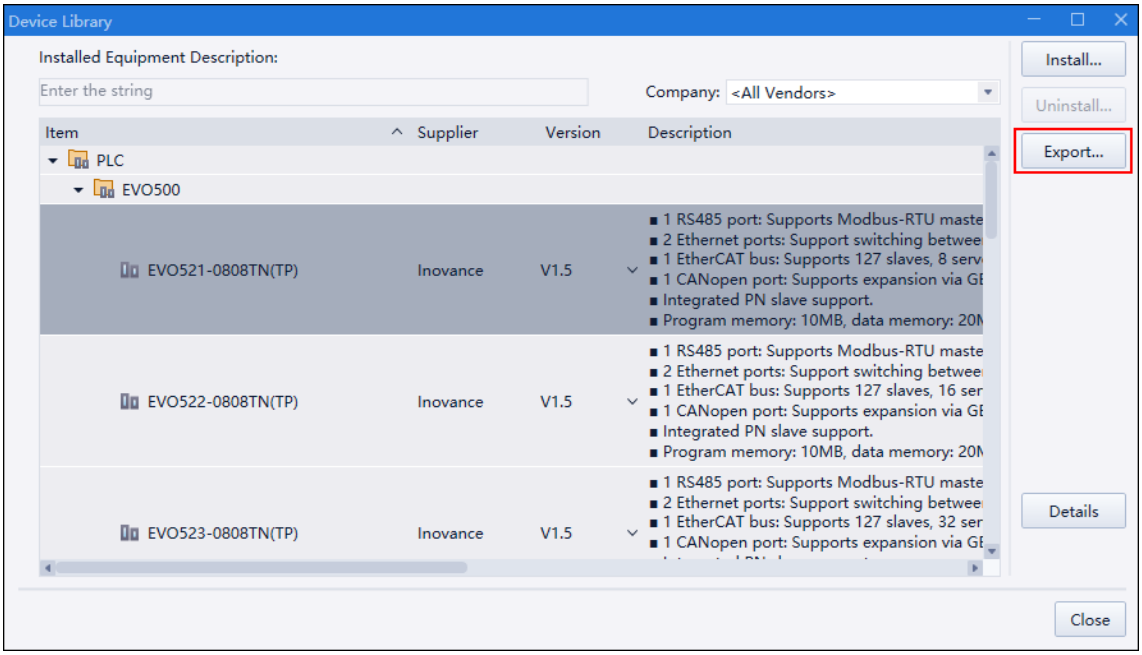
8.15.2 Exporting a Device

You can export the selected devices in the device library to generate a DVC file.

Procedure

1. Click the "Tools" option in the menu bar, and then select "Device Library" to access the Device Library dialog box.

2. Search for and select devices using the device description, company, or device information fields.



3. Click "Export Device" on the right-side menu bar.

4. Select a path, and then click "OK" to complete the device export.

After the device is exported, a DVC file is generated in the specified path.

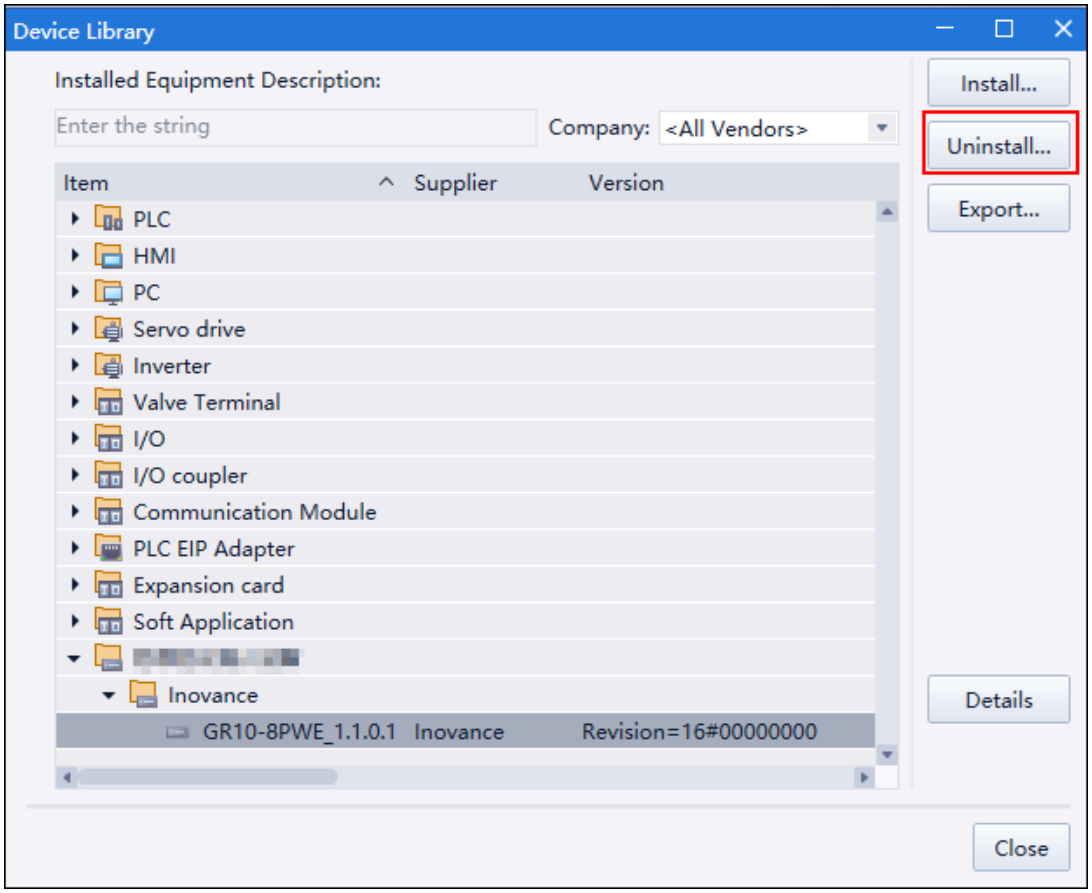
8.15.3 Uninstalling a Device

With this feature, you can uninstall third-party devices imported into the device library to prevent their use in projects. Devices pre-installed in the software cannot be uninstalled.

Procedure

1. Click the "Tools" option in the menu bar, and then select "Device Library" to access the Device Library dialog box.

2. Search for and select third-party devices using the device description, company, or device information fields.



3. After selecting one or more third-party devices, click "Uninstall Device" in the right-side menu bar, and then click "OK".

The uninstalled devices cannot be found under the "Add Device" node or in the "Device List".

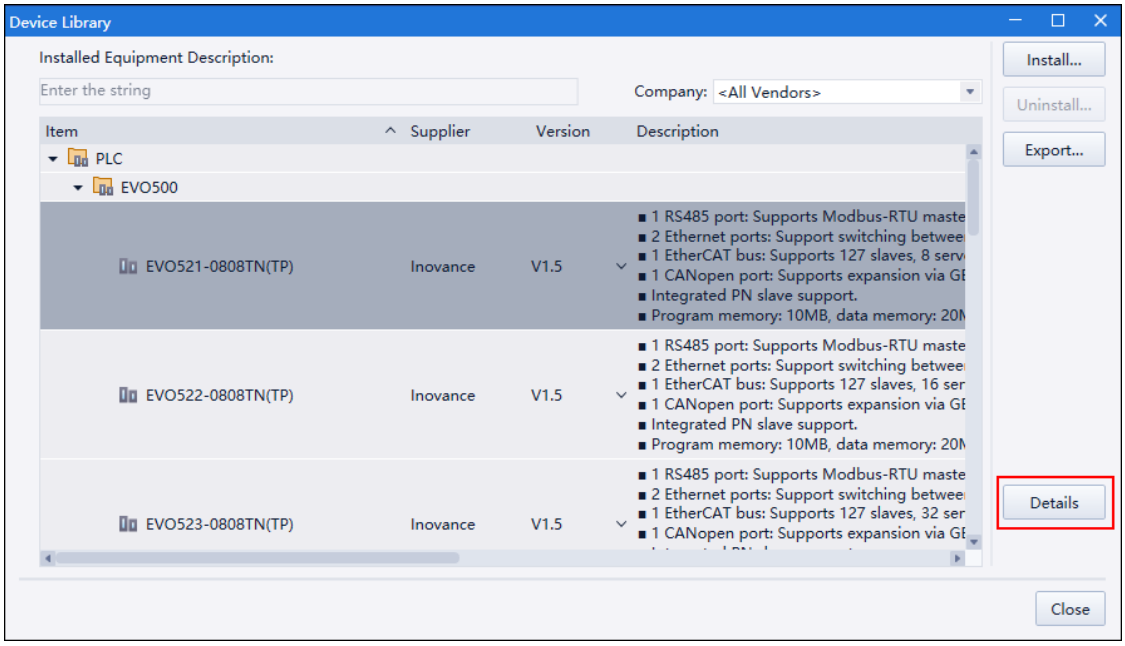
8.15.4 Device Details

You can view the detailed information about the selected device.

Procedure

1. Click the "Tools" option in the menu bar, and then select "Device Library" to access the Device Library dialog box.

2. Search for and select devices using the device description, company, or device information fields.



3. Click "Details" in the right-side menu bar to view the device details.

9 Project Security Management

9.1 Overview

In the fully integrated iFA Evolution, permissions are divided into project permissions, feature permissions, and device permissions.

- Project permissions: permissions for encrypting projects, managing members and roles, etc.
- Feature permissions: permissions for editing, using, and viewing functions.
- Device permissions: built into the devices; controlling internal parameters or operating permissions on the devices; unrelated to the iFA Evolution.

Relationships among permissions

- Relationships between project permissions/roles and user-defined permissions/roles
  - The relationships between project permissions and roles are fixed and cannot be changed;
  - The user-defined permissions are limited to be used during the operation of the monitoring system;
  - User-defined roles can be linked to both built-in system permissions and user-defined permissions, enabling more flexible permission management.
- Relationships between project permissions and device permissions
  Project permissions and device permissions are independent of each other. If a user has permissions for editing and downloading projects in the development environment, it does not guarantee that the user can download a project successfully. If device permissions are activated, the device will perform permission verification in the download process.
- Relationships between PLC upload password and other permissions
  The PLC project upload password (for project encryption) is independent of project permissions. Activating PLC device permissions represents activating the PLC upload password; projects can be uploaded only when the administrator password is entered. Each PLC is activated and configured separately, and the settings are saved within the PLC. If the device permissions are not activated on the PLC, projects can be uploaded without password input.
- Relationships between POU permissions and other permissions
  POU permissions are configured separately and are independent of project permissions. The POU can be opened with the POU password, irrespective of offline or uploaded projects.
- Relationships between PLC hardware parameter settings and other permissions
  PLC hardware parameters, including IP address, Wi-Fi function enable (only for large-scale PLCs), and Wi-Fi parameters, can be configured for both offline and online projects. These parameters can be also configured in the iFA software without any project.

  NTP settings are synchronized with project settings and can also be performed online on the device diagnosis page.

  If the PLC permission protection is enabled, the user logging into the PLC will be subject to permission verification before setting parameters. Only after successful permission verification, the user can modify the parameters.
- Relationships between PLC Web user permissions and other permissions
  PLC Web is activated on the CPU Properties page of the iFA Evolution, and its login username and password must be configured. These settings are saved with the project and downloaded to the PLC. The

Web username and password must be configured and downloaded separately for each PLC, independent of the iFA Evolution.

- Relationships between CPU properties and feature entry points
  √ indicates the feature entry point supports setting the CPU properties; × indicates it does not support.

Feature	iFA Evolution			Web Client
	Project CPU Properties	Project CPU Online Diagnostics Page	Interface Independent of Projects	Web Page
Activating PLC Web	√	×	×	×
PLC Web user	√	×	×	×
Adding variables to be monitored via Web	√	×	×	×
Setting PLC Wi-Fi parameters	×	×	√	√
IP	√	×	√	×
NTP	×	√	×	×
Activating OPC UA	√	×	×	×
Setting OPC UA parameters	√	×	×	×
Updating firmware	×	×	√	×
Exporting device data	×	×	√	×
Restoring factory settings	×	×	√	×

- Relationships between development permissions and operation permissions
  Due to the special nature, the monitoring system is generally developed and operated by different roles. Development engineers are responsible for project development, while on-site operation engineers are responsible for operating devices in running mode. Development and operation permissions are both configured in the iFA Evolution. Development permissions take effect immediately upon configuration, while operation permissions take effect after the project is downloaded to the runtime environment and started.
- Relationships between drive permissions and other permissions
  The built-in drive permissions are independent of project permissions. Each drive has its own password set and downloaded separately. If the user using the iFA Evolution has the permission for debugging a drive (or the PLC code debugging permission for drives connected to the PLC), the user can perform the drive debugging. For independent drive debugging software (IDE) without PLC, the debugging software permissions are controlled by the iFA Evolution, but the drive's internal permissions undergo a secondary verification.

Permission categories

Category		Permissions
Project permissions	Project permissions	Encrypting projects, managing members, managing roles
Feature permissions	PLC feature permissions	Editing PLC hardware, editing PLC programs, downloading PLC projects, editing PLC libraries
	Drive feature permissions	Editing drive hardware, editing drive programs, downloading drive projects, editing drive libraries
	Monitoring system feature permissions	Editing the monitoring system, downloading the monitoring system, editing monitoring system libraries, managing the monitoring system, operating the monitoring system, viewing the monitoring system
	User-defined permissions	A supplement to built-in system permissions. The permission name can be self-defined, with a default name of "Right_x". Up to 32 user-defined permissions can be created, mainly including detailed operation permissions related to monitoring system, such as assigning permissions for buttons A and button B separately to a role (enabling precise control over operation permissions).

9.2 Feature Permissions in User Management

Permission management in the iFA Evolution include two categories:

- Permissions bound to user accounts (username and password must be set and are managed in the iFA Evolution).
  The permission categories and descriptions are as follows:

Category			Description
Role category	Project configuration	Project configuration administrator	Has all permissions in the project, including global development, device development, and device operation permissions.
		Project configuration member	Assigning roles
	PLC development	PLC engineer	Editing PLC hardware, editing PLC programs, editing PLC libraries, downloading PLC projects (For PLC runtime password authentication, see 10.11.6 PLC Permission Configuration) Note: For the relationships between the PLC-related operation scopes and built-in system permissions, see 9.14 PLC Permissions.

Category				Description
	Drive	Drive engineer		Editing drive hardware, editing drive parameters and programs (password authentication performed by the drive when parameters are modified), editing drive libraries, downloading drive projects
	Monitoring	Development permissions	Monitoring engineer	Editing monitoring parameters and screens, downloading monitoring projects to target devices, editing monitoring libraries and templates, managing users in running mode, operating and viewing the monitoring system in running mode
		Operation permissions	Monitoring administrator	Managing users in running mode, operating and viewing the monitoring system in running mode
			Monitoring operator	Operating and viewing the monitoring system in running mode
			Monitoring observer	Viewing the monitoring system in running mode
	User-defined roles			A supplement to built-in system roles; allows binding permissions (all permissions except project encryption /decryption and user creation/deletion) to meet on-site needs
Category	Development	PLC		Configuration, modifying PLC settings
		Monitoring		Configuration, modifying monitoring settings, editing monitoring templates
		Drive		Independent drive commissioning software, drive commissioning, parameter modification
		User-defined permissions		A supplement to built-in system permissions. The permission name can be self-defined, with a default name of "Right_x". Up to 32 user-defined permissions can be created, mainly including detailed operation permissions related to monitoring system, such as assigning permissions for buttons A and button B separately to a role (enabling precise control over operation permissions).

- Permissions not bound to user accounts (permission granted upon password match). The permission categories and descriptions are as follows:

Category			Description
Permissions set independently but managed by username and password	Web server access	Web user	Username and password are set independently. For details, see 10.6.14.7 Web Server Configuration.

Category			Description
	OPC UA access	Username and password	Set independently for each device. For details, see 10.6.14.5.2 Configuring10.6.14.5.2 OPC UA server.
		Credential	Credentials are managed independently for each device.
Permissions managed solely by password	PLC program protection		Read-only permission
			Comparison permission (write protection)
	PLC built-in access permission	Administrator	Full access to all device functions
		Operator	Modifying parameters unrelated to program operation
		Observer	View-only permission without any operation permission
		Anonymous	Viewing diagnostic information without password
	Online setting of PLC hardware parameters		IP address
			NTP server
			Wi-Fi
	Drive access permission		Modifying parameters with the drive password
	Drive	Modifying parameters	Password verification by the drive
		Viewing parameters	Password verification by the drive
	Uploading	Uploading PLC projects	Projects can be uploaded with the upload password.
		Uploading monitoring projects	Projects can be uploaded with the upload password.

Permission setting

In the iFA Evolution, the permissions include built-in system permissions and user-defined permissions. Among the built-in system permissions, all permissions are project configuration permissions except "Monitoring operation" and "Monitoring view," which are operation permissions.

In the "User and role" configuration item, select the "Permissions" tab. You can add permissions by double-clicking "Add permission" in the permission list. Comments can be added for newly added permissions, but the comments for other system permissions cannot be modified.

NOTE

If the logged-in user does not have the required permission, the following will occur:

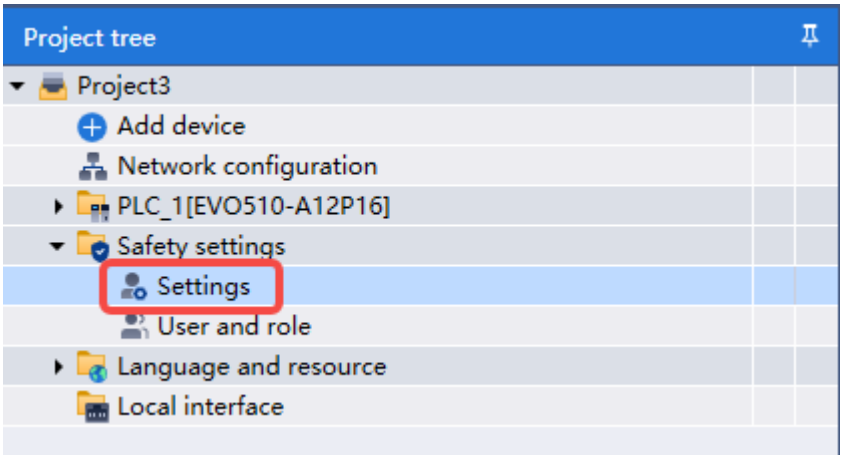
- In editing mode, the user will be unable to perform modification. Any modification attempts will trigger a pop-up window indicating that the current user lacks modification permission and is allowed to view information only.
- Clicking the Download button will trigger a pop-up window indicating that the current user lacks download permission; this also applies for operation permissions of the monitoring system.
- If a module is not installed in the user's software, the roles and permissions for that module will not be displayed on the page.

9.3 Enabling Project Protection

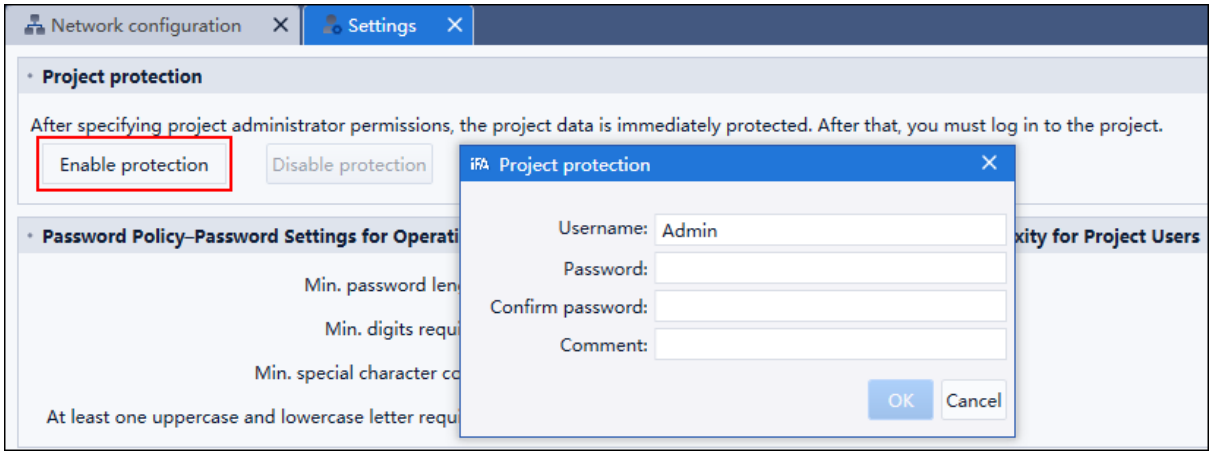
You can protect the project data in a project by managing users to prevent unauthorized access. To enable project protection, create a project configuration administrator, who can subsequently create other users and assign them appropriate roles and permissions. Only users with sufficient permissions can open and modify the project.

Procedure

1. In the project tree, select "Safety settings" and then double-click the "Settings" tab.



2. Click "Enable protection", and then "Project protection" dialog box appears.



3. Enter a username.
4. Enter the password.
   When activating user management for the first time, follow these password policies:

   “Min. password length” : The minimum number of characters in the user password must be between 3 and 32, with a default value of 8.

   “Min. digit count” : The minimum number of digits in the user password must be between 0 and the minimum password length, with a default value of 1.

   “Min. special character count” : The minimum number of special characters in the user password must be between 0 to the minimum password length, with a default value of 0.

   “At least one uppercase and lowercase letter required,” which is optional. When this option is selected, the password must meet this requirement. This option is selected by default.

   The project administrator can adjust these password policies as needed. For details, see 9.5 Setting Password Policies.
5. Re-enter the password for confirmation.
6. When necessary, enter the comments.
7. Click "OK" to enable project protection and encrypt the project data.

> NOTE
>
> The first user created in the system is the project administrator, who can create other roles.

Subsequent operations

When project protection is enabled, a pop-up window will appear during project login, which states that "This project is protected. Log in with valid credentials." Enter the appropriate username and password in the login field, and then click "OK" to access the project. You can only perform operations allowed by your permissions after logging in.

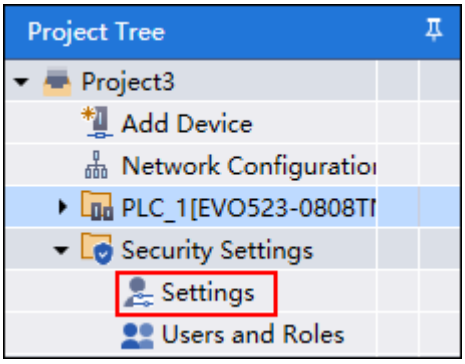
After logging in as the project configuration administrator, you can create new users, assign them appropriate roles, and add user-defined roles and permissions. For details, see 9.7 Managing Users, 9.8 Managing Roles, and 9.9 Assigning Roles.

9.4 Disabling Project Protection

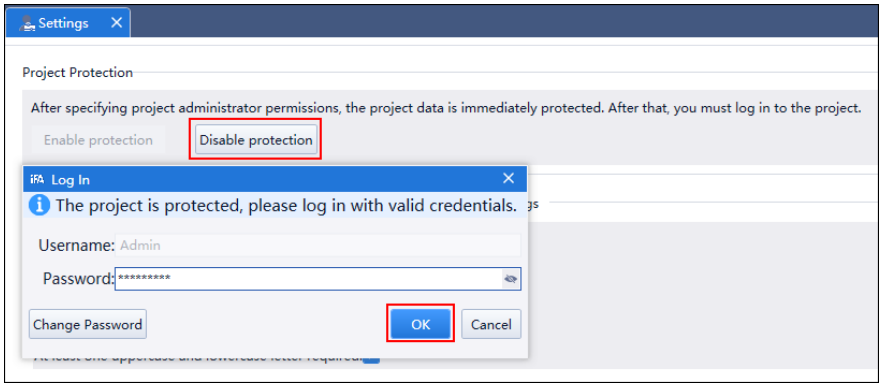
When project protection is enabled, the project configuration administrator can cancel the project protection at any time, and then the project data will no longer be encrypted.

Procedure

1. In the project tree, select "Safety settings" and then double-click the "Settings" tab.



2. Click "Disable protection," and enter the project configuration administrator's password in the pop-up window.



3. Click "OK" to disable the project protection; no login is required for subsequent access to the project.

9.5 Setting Password Policies

The following are the restrictions regarding the project configuration administrator and users, including usernames, passwords, and comments.

Username

The rules for entering a username are as follows:

- The number of input characters cannot exceed 128.
- The username can consist of letters, numbers, and special characters. Supported special characters include (~`!@#$%^&*()-_=+[]{}|\;:,.<>/? “”).

Password

You can flexibly set your own password rules to meet different requirements on password complexity.

“Min. password length” : The minimum number of characters in the user password must be between 3 and 32, with a default value of 8.

“Min. digit count” : The minimum number of numeric characters in the user password must be between 0 and the minimum password length, with a default value of 1.

"Min. special character count" : The minimum number of special characters in the user password must be between 0 to the minimum password length, with a default value of 0.

"At least one uppercase letter and lowercase letter required," which is optional. When this option is selected, the password must meet this requirement. This option is selected by default.

Supported characters for usernames and passwords:

- ~`!@#$%^&*()-_=+[]{}|\;:,.<>/?
- 0123456789
- A...Z a...z

Comment

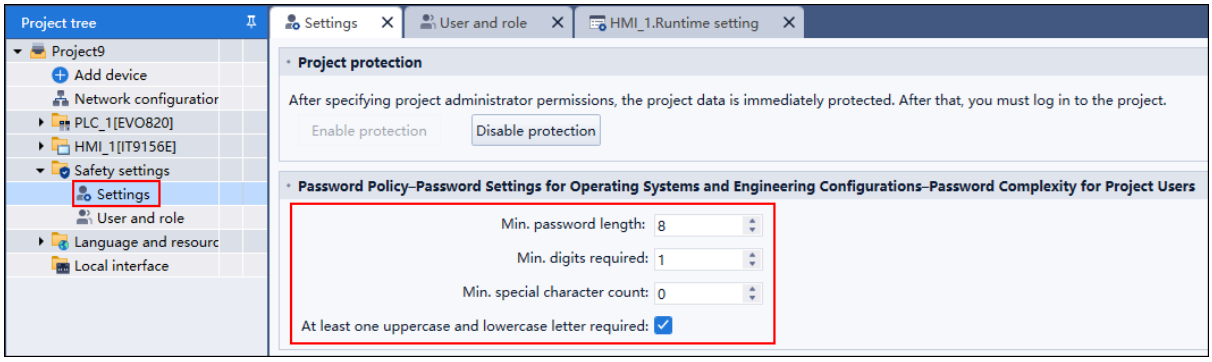
The rules for entering comments are as follows:

- The number of input characters cannot exceed 128.
- The input content is unrestricted and may be left empty.

Procedure

Proceed as follows to set the password policies:

1. Open a project for which you want to set the password policies.
2. In the project tree, select "Safety settings" and then double-click the "Settings" tab.
3. Perform the required settings.



4. Save the project, and then the password policies take effect.

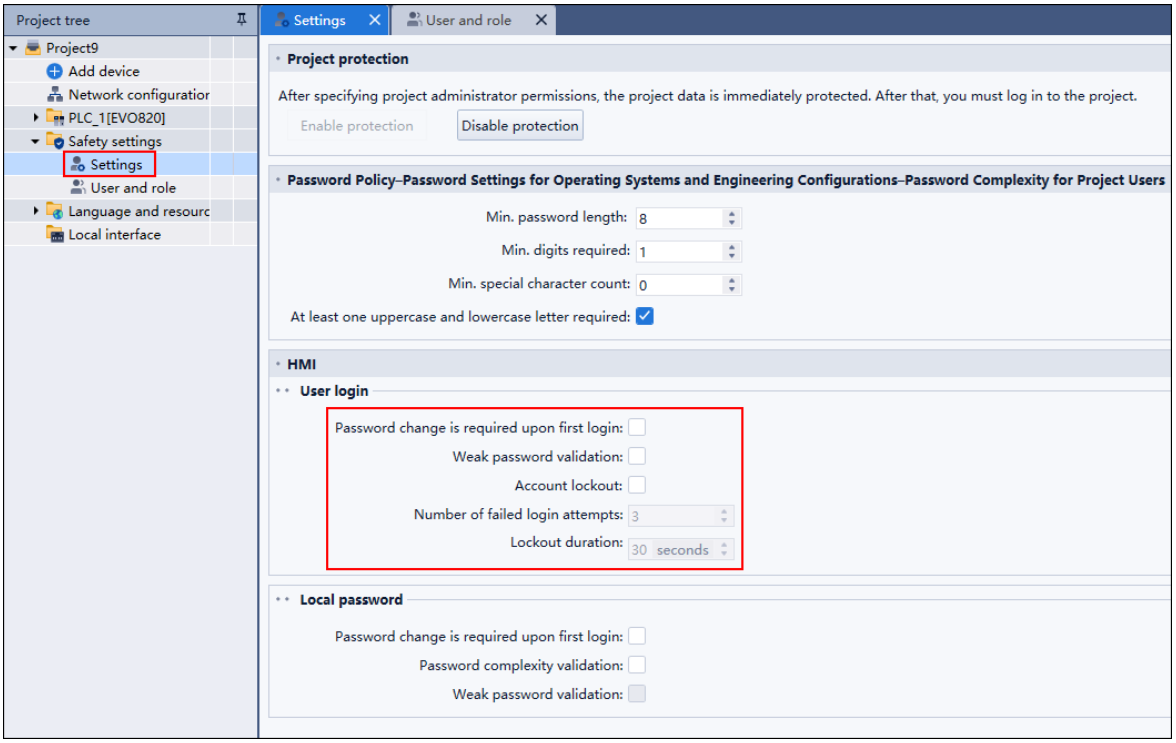
> **NOTE**
>
> Password storage uses AES-128 encryption, ensuring secure, reliable, and efficient protection of stored passwords.

9.6 HMI Security Authentication Policies

Set the security authentication policies for HMI user login and local password.

User login

1. In the project tree, select "Safety settings" and then double-click the "Settings" tab.
2. Set the security authentication policies for HMI user login.



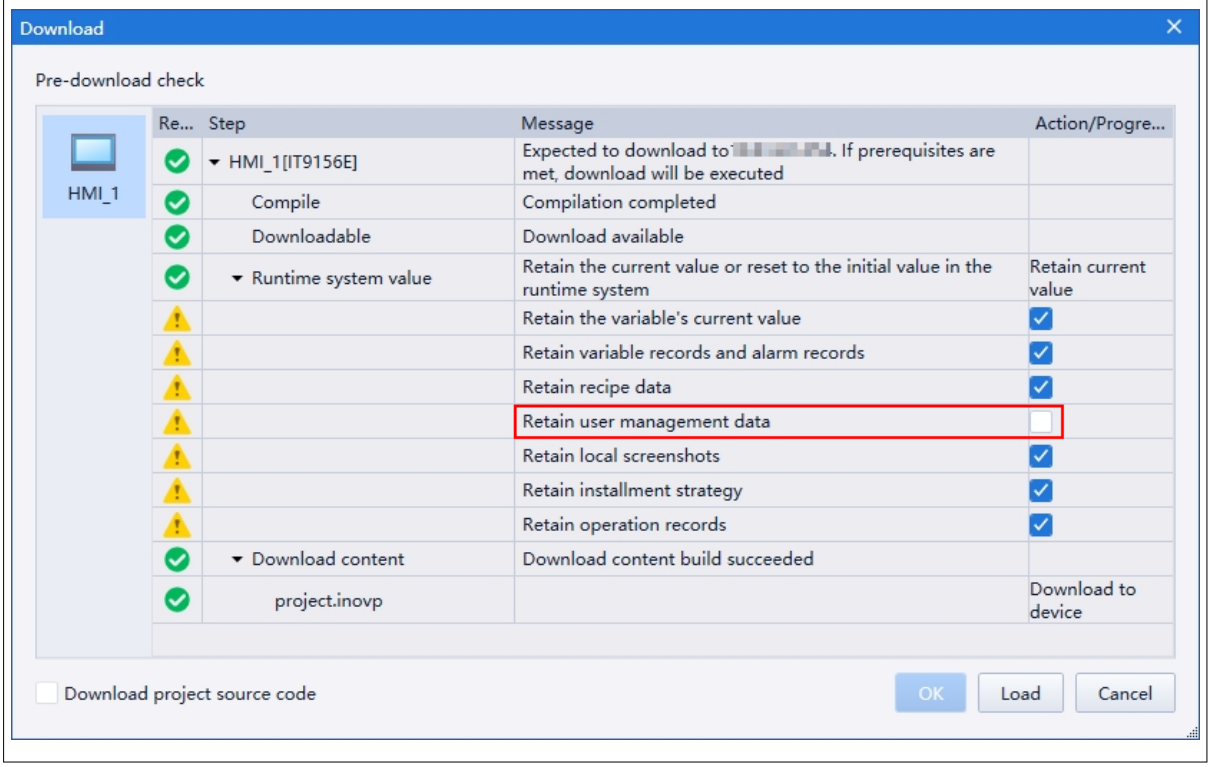
Item	Description
Password change is required upon first login	Unchecked by default. Once this option is checked, users must change their passwords upon first login. The new password must comply with the password complexity requirements. For details, see 9.5 Setting Password Policies.
Weak password validation	Unchecked by default. Once this option is checked, the password strength will be validated; if a weak password is identified, set a new password. Weak passwords: 1QAZ@wsx, Aa1!aaaa, Passw0rd!, Qwerty1@, Hello123#, Abc123!@
Account lockout	Unchecked by default. Once this option is checked, the system will keep track of the number of failed login attempts. When the number of failed login attempts reaches the setpoint, the account will be locked and cannot be used to log in during the configured lockout duration. NOTE ◦ During account lockout, the lockout duration remains active even after the device is restarted. ◦ After the lockout duration expires, you can attempt to log in again. If the password is still incorrect, the failed login attempts will not be counted, and the account will be immediately locked until the lockout duration expires again.

Item	Description
Number of failed login attempts	When the "Account lockout" option is checked, you can define the number of failed login attempts. Once exceeded, the account will be locked. The default value is 3 attempts, with a configurable range of 1 to 9. Note: Login attempts without a password are considered invalid and not counted toward the failed login attempt limit.
Lockout duration	When the "Account lockout" option is checked, you can define the lockout duration, during which the account will be prevented from logging in. The default value is 30s, with a configurable range of 1s to 3600s.

3. After the previous step is done, all subsequent HMI login attempts will be authenticated according to the configured security policies.

NOTE

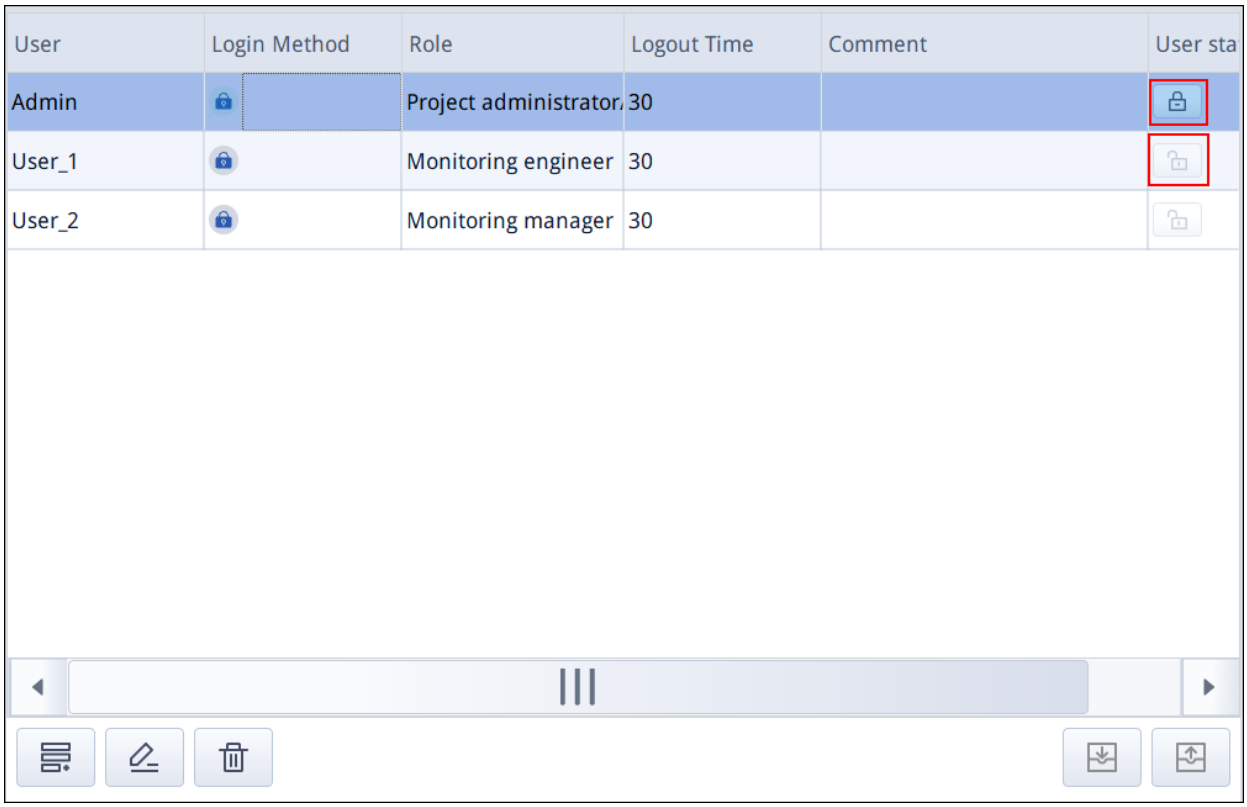
If the "Retain user management data" option is checked before the project download, the account status, lockout duration, and current number of failed login attempts remain active; if unchecked, all data will be cleared.



Manual account lock/unlock

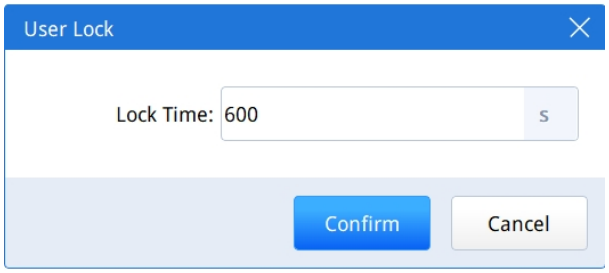
The 13.3.5.4.5 User View control is used to display and manage user information for the current device. You

can view the user status and manually lock/unlock accounts through this control.



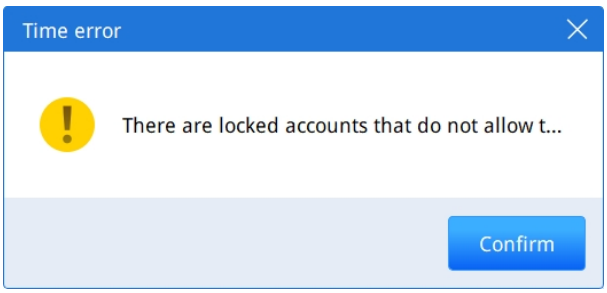
1. When the "User view" control is operating, clicking in the User Status column will immediately unlock the account.

2. Click in the User Status column, set the lockout duration in the pop-up window, and then click "Confirm" to lock the account.
   The default lockout duration is 0s, with a configurable range of 0 to 3600s. Setting it to 0s means permanently locking the account.



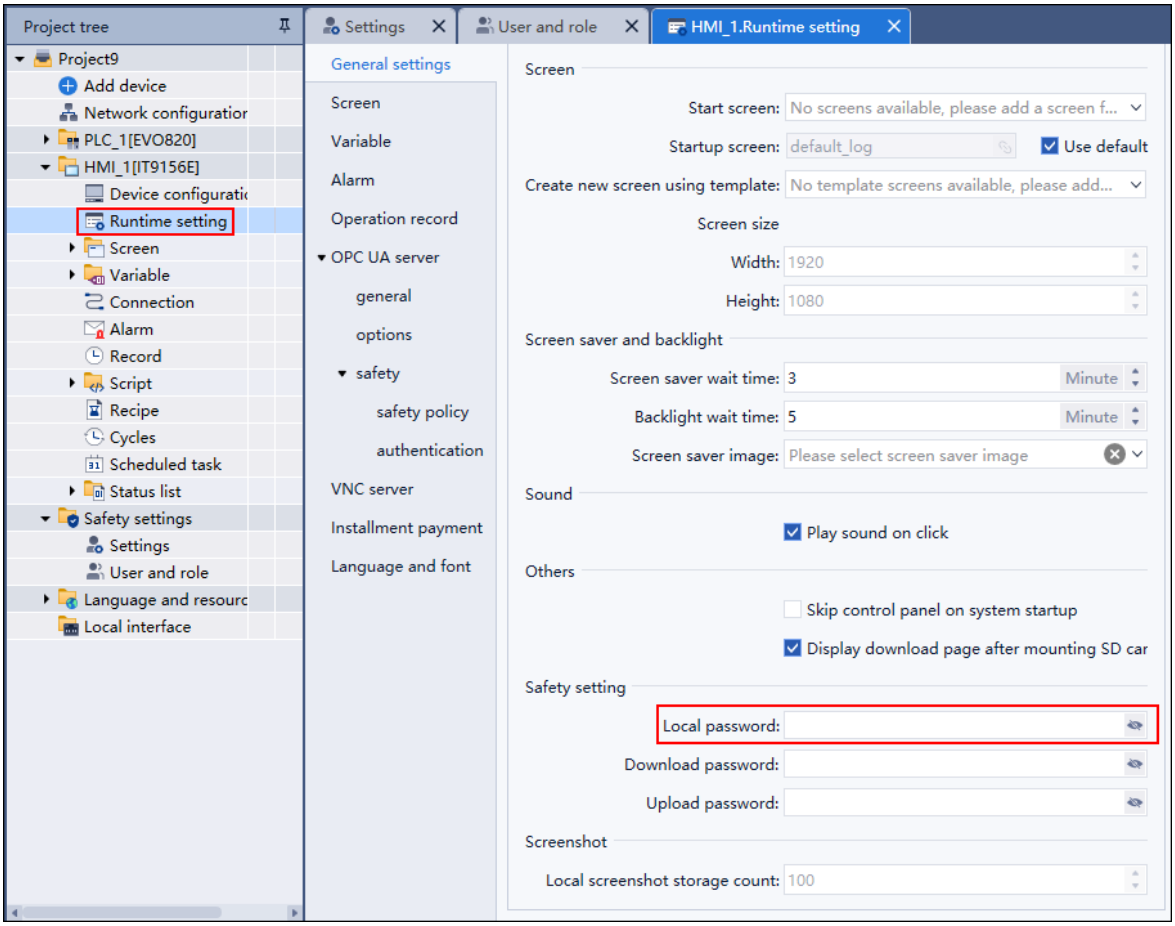
NOTE

- Users with the monitoring administrator permissions are authorized to unlock and lock accounts.
- Users with the monitoring administrator permissions can unlock one another, but cannot lock one another.
- Users with the monitoring administrator permissions cannot lock themselves. The field under the User Status column for the currently logged-in user with the monitoring administrator permissions is disabled.
- When an account is temporarily locked and its lockout timer is counting down, the following operations will not take effect: setting the time zone, date, time, or date and time through system functions; modifying the time zone, time, or date through the touchscreen. If an attempt is made to perform any of the preceding operations, a prompt box will pop up on the interface, as shown in the following figure.



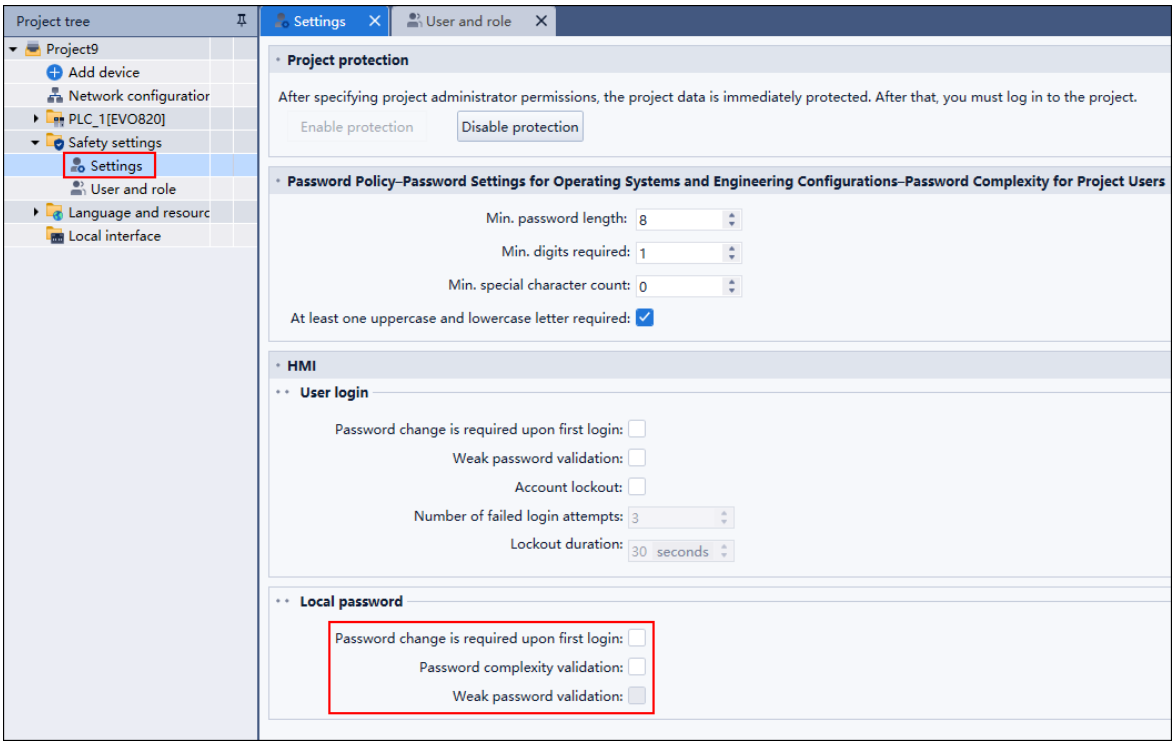
Local password

1. The local password is the password for accessing the HMI control panel. It can be set by clicking the "General settings" tab of the "Runtime setting" node on an HMI device.



2. In the project tree, select "Safety settings" and then double-click the "Settings" tab.

3. Set the security authentication policies for HMI local password.



Item	Description
Password change is required upon first login	Unchecked by default. Once this option is checked, change the local password when accessing the HMI control panel for the first time with this password.
Password complexity validation	Unchecked by default. Once this option is checked, the local password complexity will be validated. If any password does not meet the requirements, set a new password. For password complexity settings, see 9.5 Setting Password Policies.
Weak password validation	Unchecked by default. Once this option is checked, the local password strength will be validated; if a weak password is identified, set a new password. Weak passwords: 1QAZ@wsx, Aa1!aaaa, Passw0rd!, Qwerty1@, Hello123#, Abc123!@

4. After the previous step is done, subsequent HMI control panel login attempts with a local password will be authenticated according to the configured security policies.

9.7 Managing Users

After enabling project protection, the project configuration administrator can add users to a project and assign them appropriate roles. Users can be assigned one or more roles. For example, if role A and role B each include five different permissions, a user being assigned both roles A and B can perform operations defined by ten permissions.

Add a user

1. In the project tree, select "Safety settings" and then double-click "User and role" tab.
2. Double-click "<Add user>" in the user list.
3. Enter the user password twice in the pop-up window, and then click "OK".
4. In the user list, select the desired user, and then check the appropriate roles on the "Developer role" or "Monitoring runtime role" tab below.

NOTE

- The default name for a new user is User_x; the initial password is null; the default logout time is 30 min.
- If a new user does not set a password, role association and related operations are not supported.
- Up to 100 users can be created, including the project configuration administrator.
- In the password field of the user interface, the project configuration administrator can set passwords for any user.
- Each user can modify their own information; the users without user management permission cannot modify other users' information.
- The logout timeout is enabled by default and can be configured from 1 to 600 min. You can also choose to disable it.
- When the logout timeout is enabled and a user logs in during runtime, the user will be automatically logged out after the specified period of inactivity and must log in again to perform further operations.
- The "HMI enable" option is checked by default for new users, indicating that the users are allowed to log in to the HMI runtime and perform related operations. Users that are not enabled will not be displayed in the HMI runtime login interface and User View control, and a pop-up prompt "Your account is not enabled" will appear when the login, password change, or user switching function is used.

Copy/Paste/Delete a user

Right-click an existing user in the user list to copy, paste, delete, or rename the user.

- Pasted roles are automatically renamed and arranged at the end of the list.
- The current user's password cannot be copied.
- The project configuration administrator cannot be deleted.
- Multiple rows can be selected for batch copying, pasting, and deletion.

9.8 Managing Roles

A role represents a group of permissions. One or more permissions can be assigned to a role, and then the role can be assigned to a specific device. Once a user is bound to a role, the user can perform the relevant operations defined by the permissions contained in that role on the corresponding device.

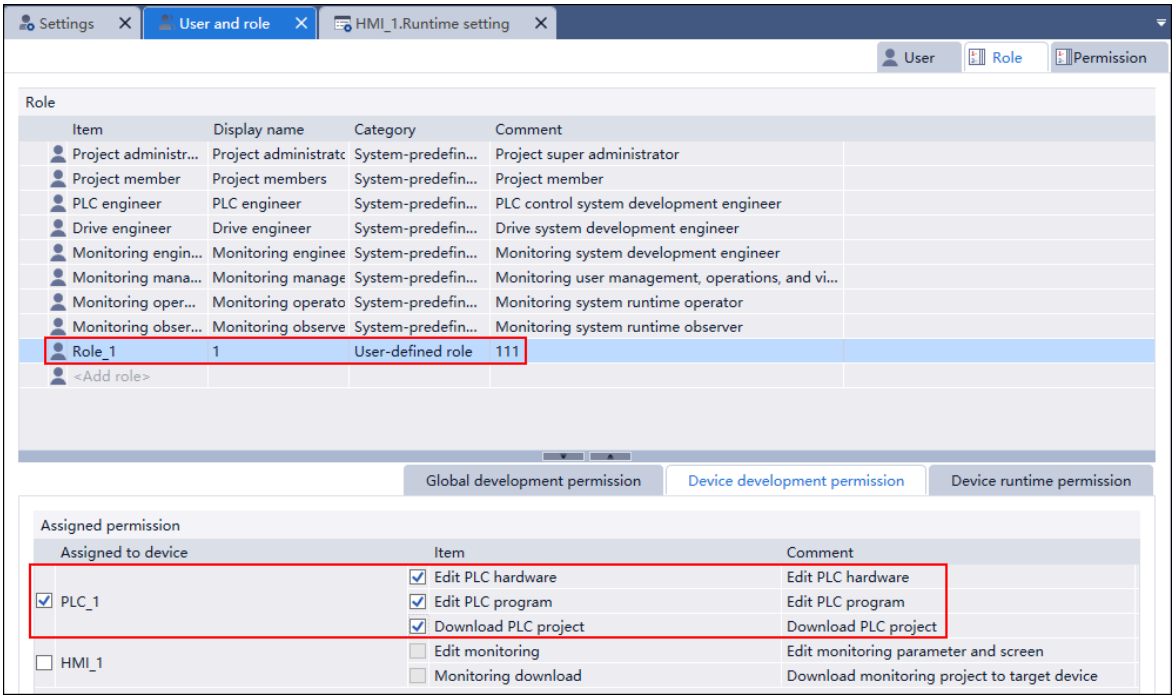
Roles include built-in system roles and user-defined roles. Built-in system roles are associated with appropriate permissions by default, while user-defined roles require manual permission assignment to allow more granular and flexible permission control.

Role categories

Role	Description
Project configuration administrator	By default, the first administrator who enables project protection is the project configuration administrator, possessing all permissions in the project.
Project configuration member	More than one member can be designated by the project configuration administrator, and each of them can be granted additional permissions beyond the default ones.
PLC engineer	Primarily responsible for development and commissioning of PLCs.
Drive engineer	Primarily responsible for development and commissioning of drives.
Monitoring engineer	Primarily responsible for development and commissioning of monitoring systems.
Monitoring administrator	Monitoring runtime administrator, mainly responsible for managing users in the monitoring runtime mode, and operating and viewing the runtime monitoring system.
Monitoring operator	Monitoring runtime operator, having monitoring system operation permissions matched with the control permissions configured in the configuration environment.
Monitoring observer	Monitoring runtime observer, having the permissions to switch screens and view equipment operation status but no control permissions over the monitoring system.
User-defined role	A supplement to built-in system roles, for which appropriate permissions can be defined by the user to meet on-site practical needs; more than one role can be added.

Add a role

1. In the project tree, choose "Safety settings -> Settings", enable project protection, and set the username and password.
2. In the project tree, select "Safety settings" and then double-click "User and role" tab.
3. Select the "Role" tab, and then double-click "Add role" in the role list.
4. After selecting a role, check the boxes in the "Global development permission," "Device development permission," or "Device runtime permission" tab below to assign permissions.
   Among these, "Device development permission" and "Device runtime permission" only take effect after being assigned to specific devices. For details, see 9.9 Assigning Roles.



NOTE

- Only when project protection is enabled, the device development and runtime permissions can be configured.
- Up to 100 roles can be created, including the built-in system roles, and only one project configuration administrator is allowed.
- For user-defined roles, the name, display name, and description can be modified; for built-in system roles, only the display name can be modified.
- The default name for a new role is Role_x.

Copy/Paste/Delete a role

Right-click an existing role in the role list to copy, paste, delete, or rename the role. The built-in system roles cannot be deleted or renamed.

- Pasted roles are automatically renamed and arranged at the end of the list.
- Multiple roles can be selected for batch copying, pasting, and deletion.
- For pasted "built-in system roles", their role category is automatically changed to "user-defined role."

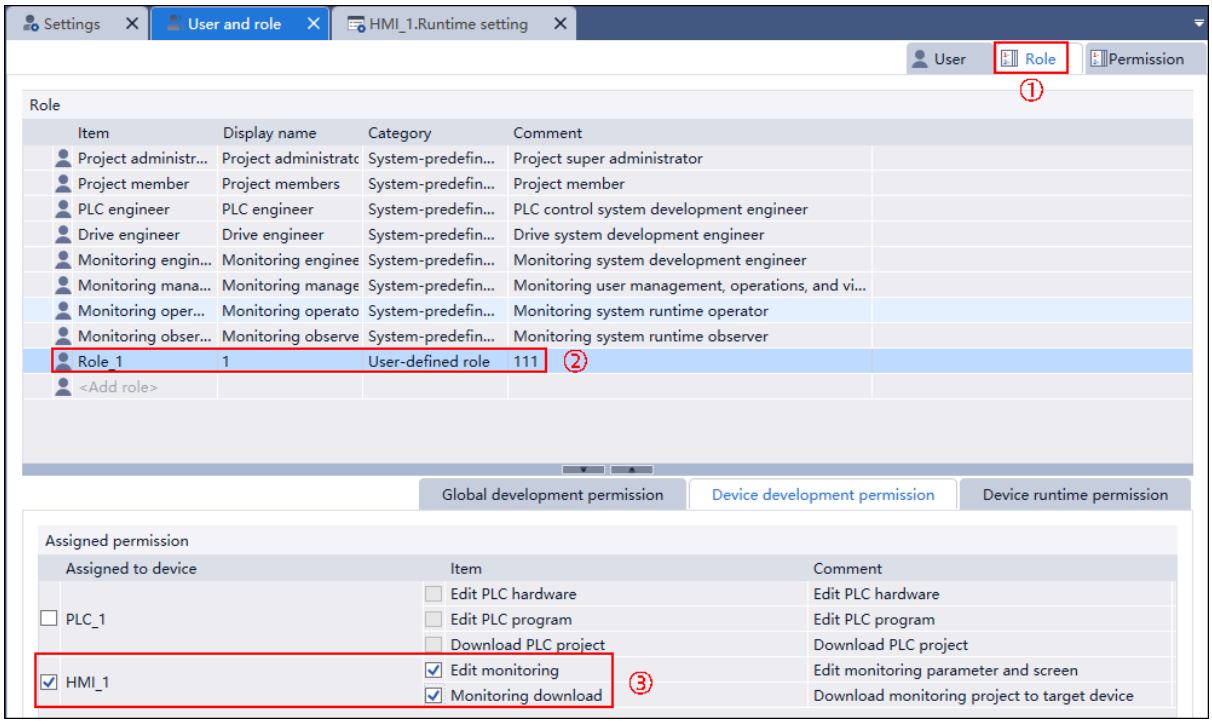
9.9 Assigning Roles

When project protection is enabled, the project configuration administrator can assign different permissions and devices to appropriate roles, and then assign these roles to different users, enabling role-based and domain-based access control.

> **NOTE**
>
> When a project contains online devices, operations such as assigning roles to users and assigning permissions and devices to roles are not supported; these operations can be performed only when the devices are offline.

Procedure

1. In the project tree, select "Safety settings" and then double-click "User and role" tab.
2. Select the "Role" tab and select the desired role from the role list.
3. Assign appropriate permissions and devices in the "Global development permission," "Device development permission," and "Device runtime permission" tabs below.

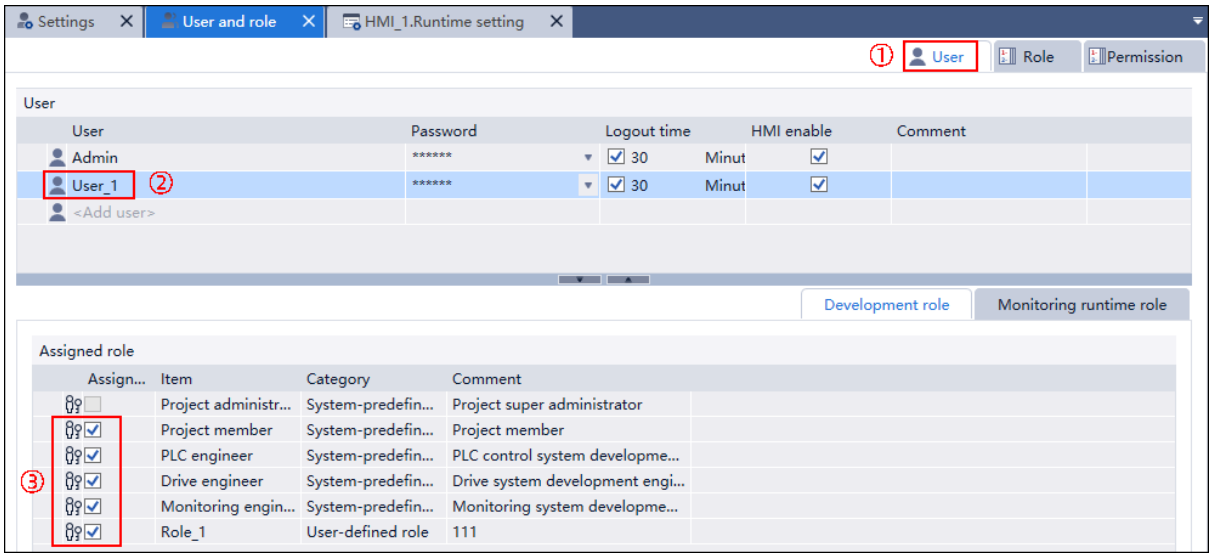


Category	Note
Global development permission	Project configuration permissions, including project encryption, member management, and role permission management; not bound to devices. The permissions for built-in system roles cannot be modified; user-defined roles can be assigned "management" permissions.
Device development permission	Includes development permissions for PLCs, drives, and monitoring devices. Permissions only take effect after being bound to a device. The permissions for built-in system roles cannot be modified, and they can be assigned to devices; user-defined roles allow flexible assignment of permissions and devices.

Category	Note
Device runtime permission	Includes monitoring device operation permission and user-defined permissions. Permissions only take effect after being bound to a device. The permissions for built-in system roles cannot be modified, and users can define permissions and link them to specific devices; user-defined roles allow flexible assignment of permissions and devices.

4. Select the "User" tab and select the desired user from the user list.

5. Assign appropriate roles in the "Development role" and "Monitoring runtime role" tabs below.



Role category	Note
Development roles	Includes built-in system roles such as project configuration administrator, project configuration member, PLC engineer, drive engineer, and monitoring engineer.
Monitoring runtime roles	Includes built-in system roles such as monitoring administrator, monitoring operator, and monitoring observer, as well as user-defined roles.

NOTE

User-defined roles are widely used in monitoring devices and can be used to accurately control the operation permissions of a specific widget on the HMI runtime screen. For example, if a permission is configured for a specific button and only user-defined role A is allowed to operate it, only users assigned user-defined role A can operate the button after logging in.

9.10 Opening a Protected Project

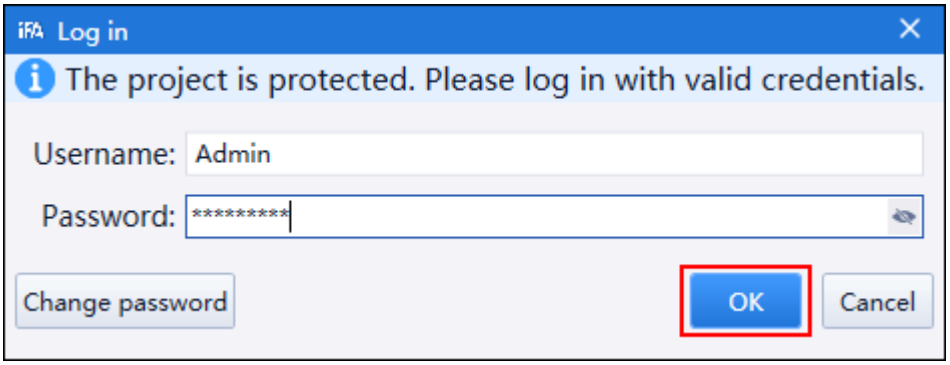
When project protection is enabled, enter the appropriate username and password to open the project. Logged-in users can perform operations defined by their permissions.

Procedure

1. Choose "Project -> Open" from the menu bar to access the "Open project" page.

2. Select the required protected project from the project list and click "Open".
   For other ways to open a project, see 8.1 Opening a Project.

3. In the "Log in" dialog box, enter the username and password, and then click "OK".



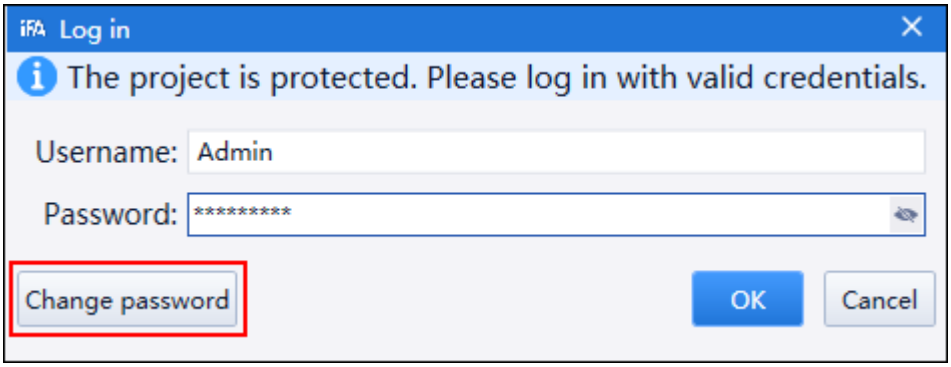
4. In the "Log in" dialog box, click the "Change password" button to change the password. For details, see 9.11 Changing the Password.

9.11 Changing the Password

After enabling project protection, you can change the user password in either of the following ways. If you log into a protected project as the project configuration administrator, you can change the password for any user; if you log in as any other user, you can change your password only.

Change the password when opening a protected project

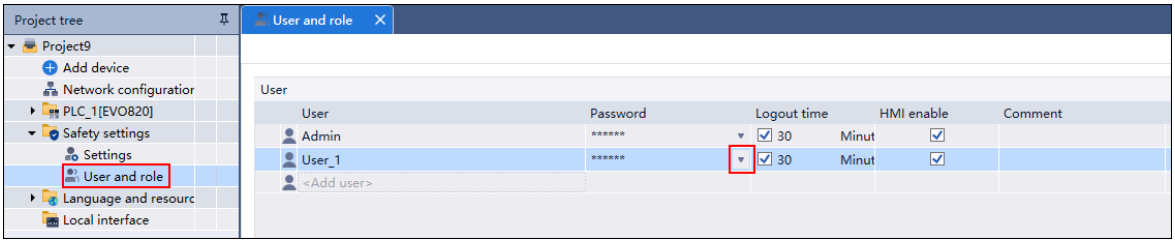
1. Choose "Project -> Open" from the menu bar to access the "Open project" page.
2. Select the required protected project from the project list and click "Open".
   For other ways to open a project, see 8.1 Opening a Project.
3. Click the "Change password" button in the "Log in" dialog box.



4. In the "Change password" dialog box, enter the username, old password, new password, and then confirm the new password.
5. Click "OK". Afterward, use the new password to log into the protected project.

Change the password when operating a protected project

1. In the project tree, select "Safety settings" and then double-click "User and role" tab.
2. In the user list, click the dropdown button in the field under the Password column corresponding to the target user.



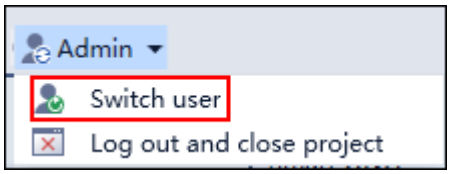
3. In the "Change password" dialog box, enter a new password and re-enter it to confirm.
4. Click "OK". Afterward, use the new password to log into the protected project.

9.12 Switching the User

After logging into a protected project, you can directly switch the logged-in user without closing the project.

Procedure

1. Log into a protected project.
2. Choose "Admin -> Switch user" from the toolbar.

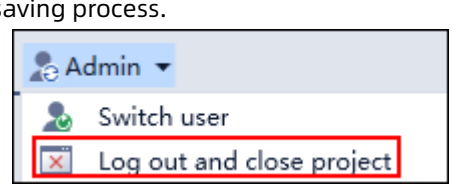


3. If you have made changes to the project but not saved them, a dialog box will appear asking "Save project?". At this point, click "OK".
4. In the "Log in" dialog box, enter the username and password of the new user.
5. Click "OK".

9.13 Exiting a Project

After logging into a protected project, you can log the user out and close the project in one of the following ways:

- Directly close the iFA Evolution. The system logs out and closes the project according to the project saving process.
- Directly close the project. The system logs out and closes the project according to the project saving process.
- Choose"Admin -> Log out and close project" from the menu bar. The system logs out and closes the project according to the project saving process.



9.14 PLC Permissions

Permission matrix

For the most complex PLC permissions in access control scenarios, the following table lists the relationships between PLC-related operation scopes and built-in system permissions.

Operation Scope\System Permission	Edit PLC hardware	Edit PLC Program	Download PLC Project
Adding deleting, or modifying a device in the network view	√	×	×
Adding or deleting all nodes under hardware configuration	√	×	×
Configuring I/O mapping	×	×	×
Establishing communication between PLCs	√	×	×
Modifying properties of the monitoring area in the network view	√	×	×
Adding deleting, or modifying a sub-module in Device Configuration view	√	×	×
Adding a new PLC, module, or I/O slave	√	×	×
Library reference (installation and reference)	×	√	×
Configuring the oscilloscope (viewing and downloading are not restricted)	×	√	×
Recipe manager operations	×	√	×
Operations under programming nodes	×	√	×
Task configuration operations	×	√	×
POU encryption	×	√	×
Adding, deleting, or renaming under motion control nodes	×	√	×
PID\PFC	×	√	×
Project comparison (online)	×	×	√
Project comparison (Copying to the left and right - software)	×	√	×

Operation Scope\System Permission	Edit PLC hardware	Edit PLC Program	Download PLC Project
Project comparison (Copying to the left and right - hardware)	√	×	×
Downloading to device	×	×	√
Uploading a project	×	×	√
Switching to online/Switching to offline	×	×	√
Replace	√	√	×
Controlling objects (including cams/axes/ axis groups)	×	√	×
Controlling objects (PID/PFC)	×	√	×
Adding and deleting watch tables and variables	×	√	×
Copying and pasting the entire PLC nodes	√	√	×
Replacing a device	√	√	×
Overwrite upload	√	√	√

NOTE

- √ indicates that the permission allows the operation. All conditions required for the operations covered by this permission must be satisfied; otherwise, the permission will not take effect.
- × indicates that no permissions are required to perform the operation.

Restrictions

- Without PLC hardware editing permission, hardware cannot be added or deleted, and hardware parameters cannot be modified.
- Without PLC program editing permission, POUs cannot be opened for viewing, and software parameters cannot be modified.
- Without PLC project downloading permission, the Download button will be grayed out, and all online permissions associated with the project will also be grayed out.

9.15 Library Project Security Management

Overview

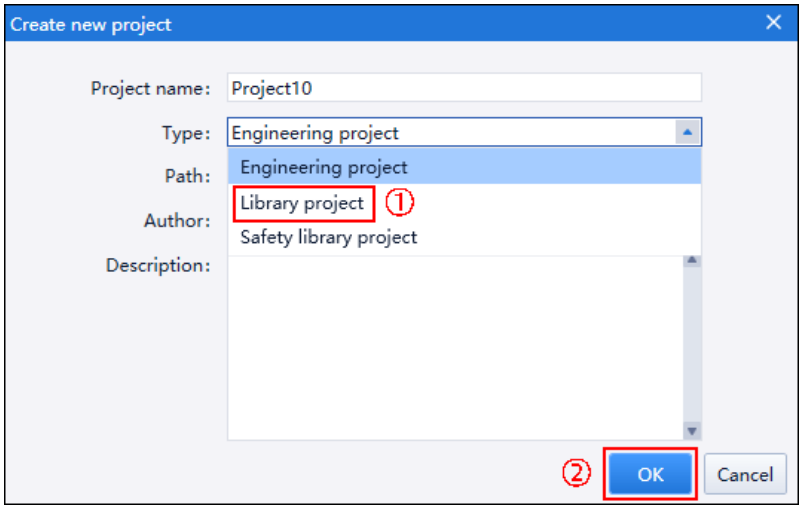
You can protect the project data in a library project by managing users to prevent unauthorized access. To enable project protection, create a project administrator, who can subsequently create other users and

assign them appropriate roles and permissions. Only users with sufficient permissions can open, edit, and view the library project.

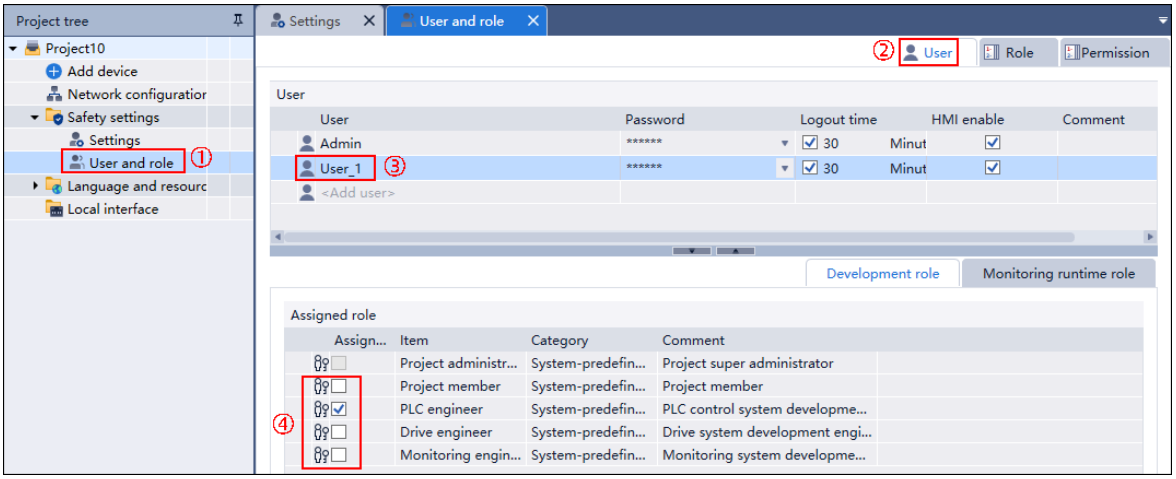
After creating or opening a library project, the operations for 9.3 Enabling Project Protection, 9.4 Disabling Project Protection9.4 ,9.5 Setting Password Policies, 9.10 Opening a Protected Project, 9.11 Changing the Password, 9.12 Switching the User, and 9.13 Exiting a Project are the same as those for regular projects.

Manage users

1. Create or open a library project.



2. After 9.3 enabling project protection, the project administrator can add users to a project and assign them appropriate roles.
3. In the project tree, select "Safety settings" and then double-click "User and role" tab.
4. Select the "User" tab, and double-click "<Add user>" in the user list.
5. Enter the user password twice in the pop-up window, and then click "OK".
6. In the user list, select the desired user, and then check the appropriate roles on the "Developer role" or "Monitoring runtime role" tab below.



NOTE

- The default name for a new user is User_x; the initial password is null; the default logout time is 30 min.
- If a new user does not set a password, role association and related operations are not supported.
- Up to 100 users can be created, including the project configuration administrator.
- In the password field of the user interface, the project configuration administrator can set passwords for any user.
- Each user can modify their own information; the users without user management permission cannot modify other users' information.
- The logout timeout is enabled by default and can be configured from 1 to 600 min. You can also choose to disable it.
- When the logout timeout is enabled and a user logs in during runtime, the user will be automatically logged out after the specified period of inactivity and must log in again to perform further operations.

7. Select the "Role" tab. In the role list, select the desired role. In the "Global development permissions" list below, check the permissions of the selected role.
   The built-in system roles and permissions are shown in the table below:

Role	Permissions	Description
Project administrator	◦ Project encryption ◦ Member management ◦ Role management ◦ PLC library editing ◦ PLC library viewing	By default, the first user who enables project protection is the project administrator, possessing all permissions, including project encryption/decryption, user management, role assignment, PLC library editing, and PLC library viewing.
Library development engineer	◦ PLC library editing ◦ PLC library viewing	This role can view and edit PLC library codes.
Library observer	PLC library viewing	This role can view PLC library code logic, but cannot perform operations such as editing, copying, pasting, cutting, and deleting.

NOTE

- Only built-in system roles and permissions can be used in library projects, while user-defined roles and permissions cannot be used. The built-in system roles are fixed to their permissions and cannot be reassigned.
- It is possible to view the built-in system roles and permissions and assign permissions to users only after the project protection is enabled.

10 PLC Programming

10.1 Programming Basics

10.1.1 Variables

10.1.1.1 Global Variables

Global variables refer to variables between VAR_GLOBAL and END_VAR that are used within the corresponding configuration and resource.

This includes system-generated SYS_VAR variables and user-defined variables in the program, as described below:

SYS_VAR

The system variable table is automatically deployed under the programming node upon project creation, named "SYS_VAR". It cannot be modified or deleted.

- When users add slave devices (devices configured under the hardware configuration node, such as I/O modules and branch modules), corresponding variables are generated. Each slave device corresponds to a unique system variable (including the disabled variable bDisableEnable); when a slave device is modified or deleted, the system variable will be updated synchronously.
- System variables support export, but do not support paste, copy, cut, and delete operations.
- The names and data types of system variables cannot be modified by users.
- When the disabled variable bDisableEnable is called in the program, the variable value of True/False is used to enable or disable the slave.

User-defined variable

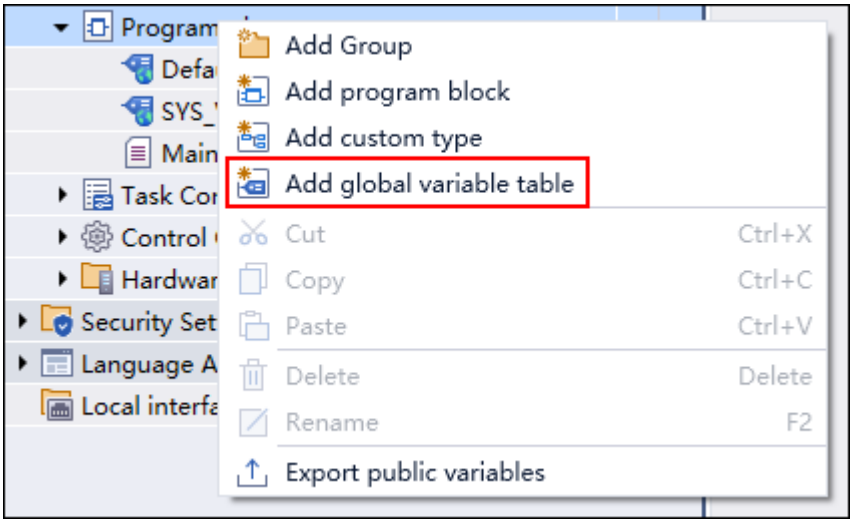
All POUs under "Programming" can use the global variables declared in the global variable table.

Prerequisites

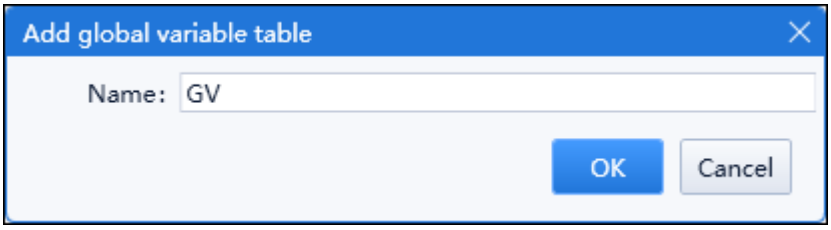
A global variable table has been created.

Procedure

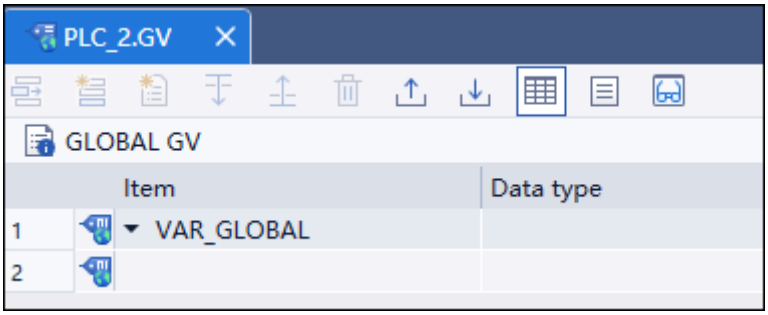
1. Right-click "Programming > Add global variable table" and the "Add global variable table" dialog box pops up.



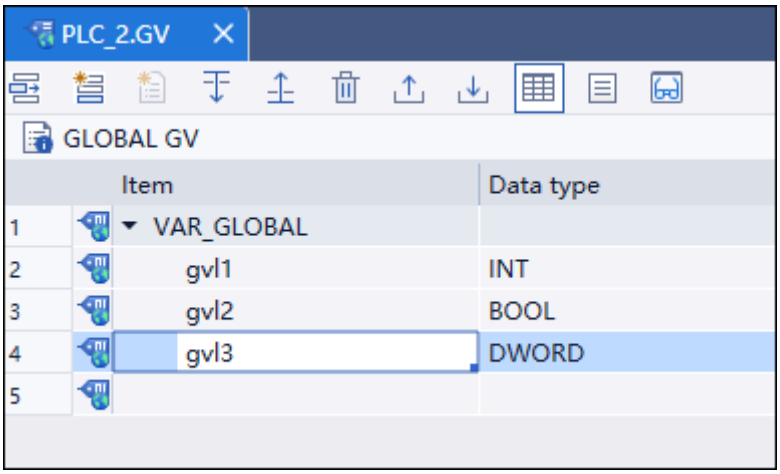
2. Set the name of the global variable table and click "OK".



3. Set the variable category to "VAR_GLOBAL" in the global variable table and add variables.



Operation result



NOTE

- When operating through variable names, if the name of a local variable and a global variable are the same, the local variable is operated on. Global variables can be accessed by prefixing the variable name with the "global variable list name" followed by a ".", such as "GV.gvl1".
- The global variable list is initialized before POU local variables are defined.

10.1.1.2 Local Variables

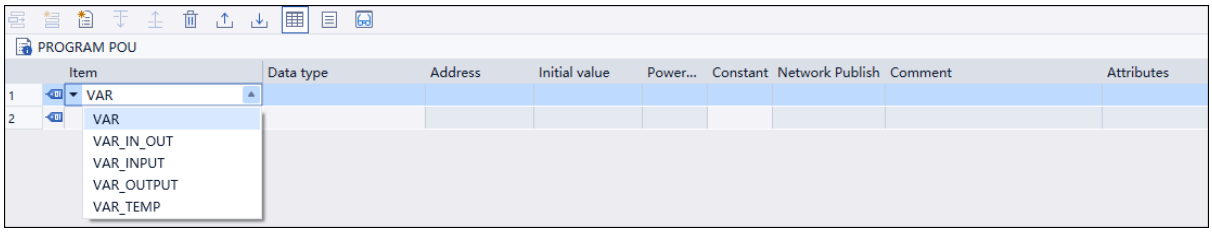
Local variables refer to variables between VAR and END_VAR in a program organization unit (POU). They can be used within this POU and other POUs.

Prerequisites

A POU has been created.

Procedure

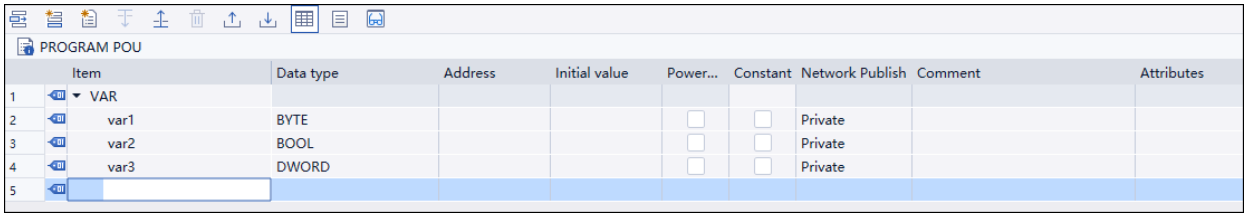
1. Open a POU (PRG, FB, or FUN) under "Programming". In the variable declaration editor in the table view, set the variable category to "VAR".



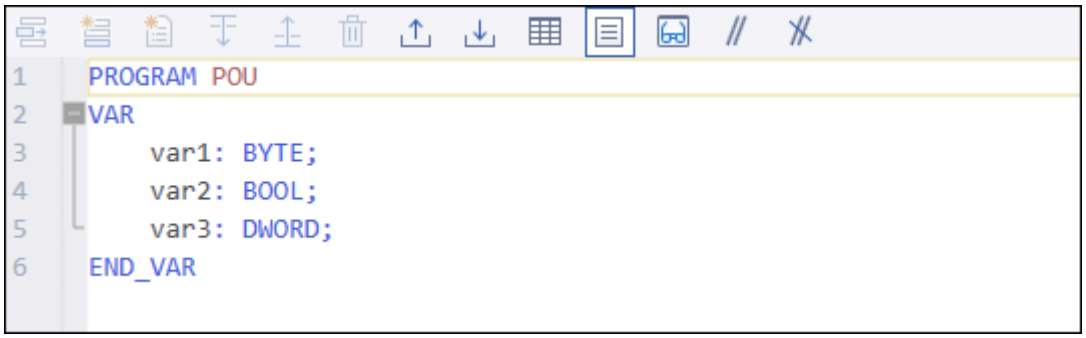
2. Add variables, and set their properties such as names and data types.

Operation result

The variable declaration editor in table view is shown in the figure below:



The variable declaration editor in text view is shown in the figure below:



10.1.1.3 Temporary Variables

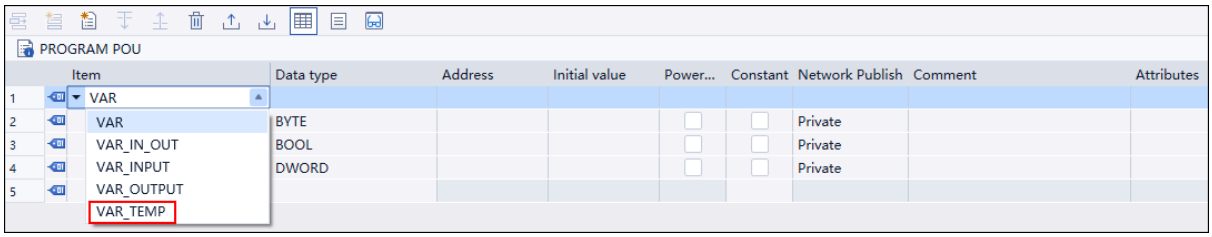
Temporary variables refer to variables defined between VAR_TEMP and END_VAR. They are stored temporarily in a program (PRG) or function block (FB). Temporary variables are initialized each time they are called.

Prerequisites

A POU (only PRG and FB) has been created.

Procedure

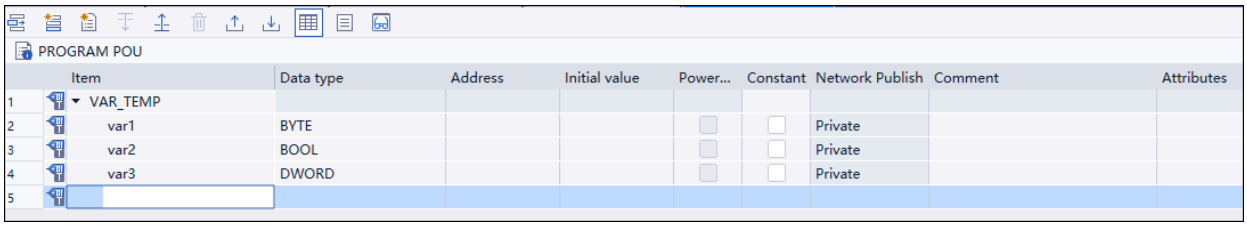
1. Open a POU (PRG or FB) under "Programming". In the declaration editor in table view, set the variable category to "VAR_TEMP".



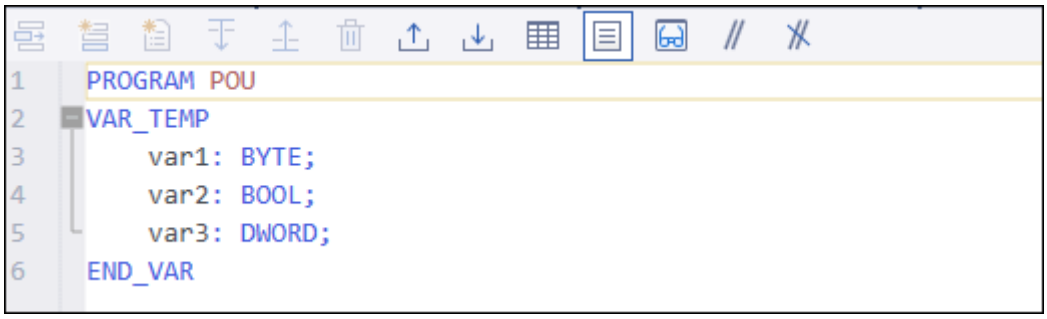
2. Add variables, and set their properties such as names and data types.

Operation result

The variable declaration editor in table view is shown in the figure below:



The variable declaration editor in text view is shown in the figure below:



NOTE

- Temporary variables can only be declared in programs (PRG) and function blocks (FB), and not in functions (FUN).
- Temporary variables are non-public variables.

10.1.1.4 Instance Variables

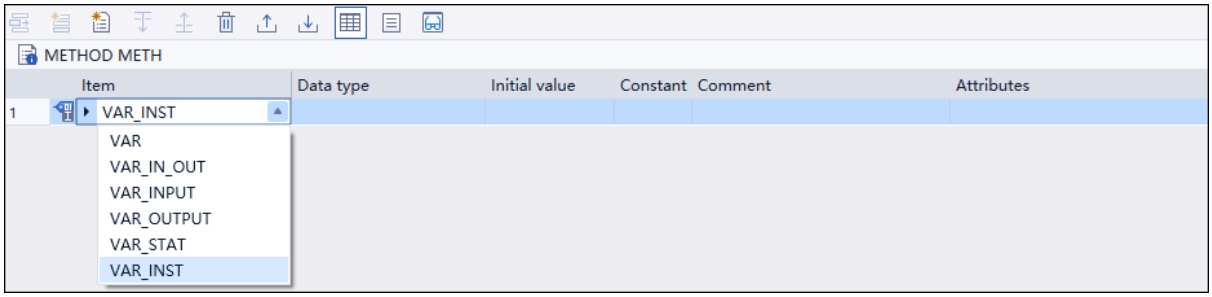
Instance variables refer to variables declared between VAR_INST and END_VAR. They are stored on the function block instance stack.

Prerequisites

A POU (only for methods under FB) has been created.

Procedure

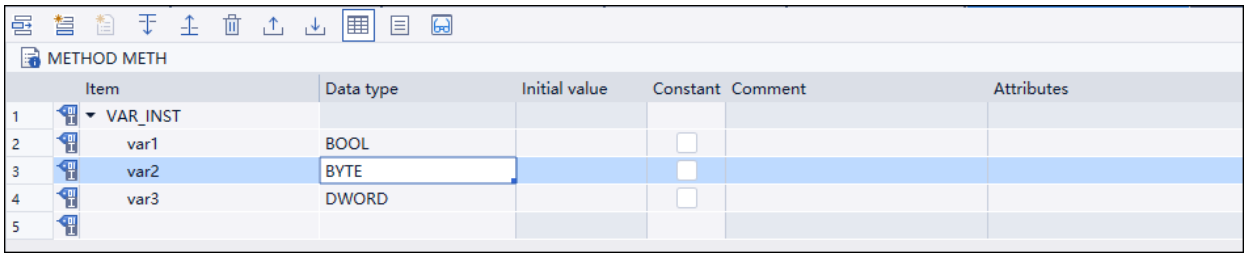
1. Open a METH under an FB under "Programming". In the declaration editor in table view, set the variable category to "VAR_INST".



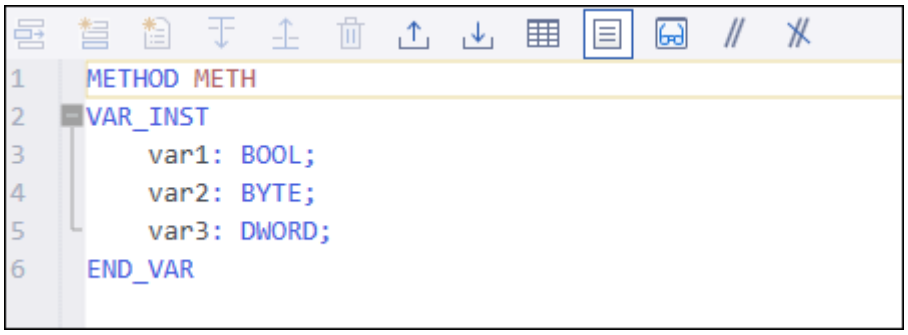
2. Add variables, and set their properties such as names and data types.

Operation result

The variable declaration editor in table view is shown in the figure below:



The variable declaration editor in text view is shown in the figure below:



NOTE

- Instance variables can only be added and used within methods under a custom function block.
- When a function block calls a method, the values of instance variables are not initialized.

10.1.1.4.1 Input Variables

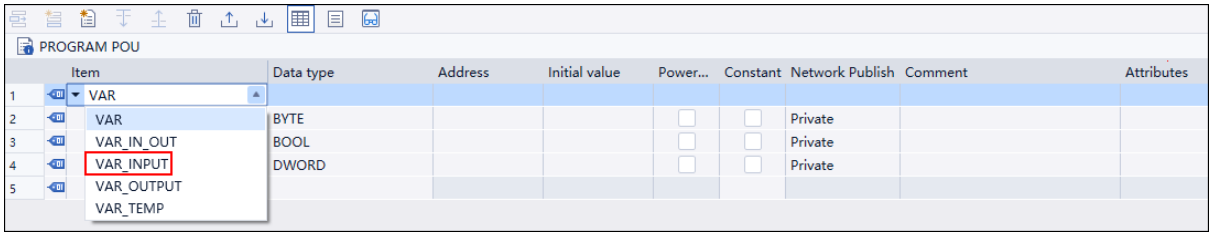
Input variables refer to variables provided externally and defined between VAR_INPUT and END_VAR. They are assigned values at the call location.

Prerequisites

A POU has been created.

Procedure

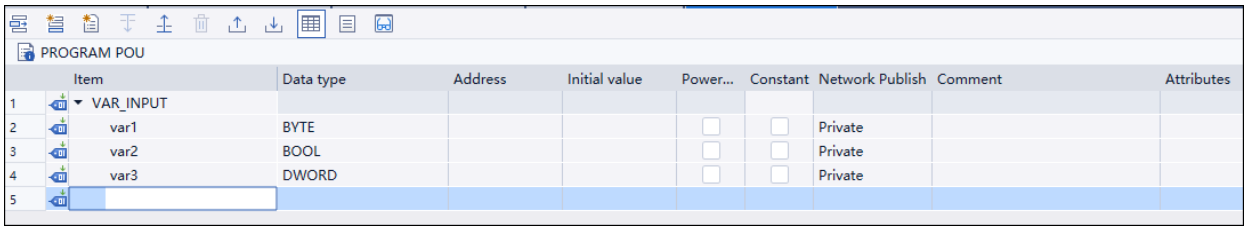
1. Open a POU (PRG, FB, or FUN) under "Programming". In the variable declaration editor in the table view, set the variable category to "VAR_INPUT".



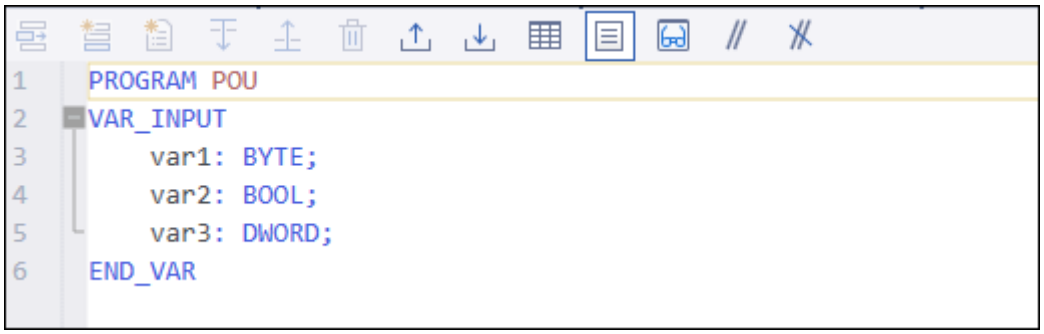
2. Add variables, and set their properties such as names and data types.

Operation result

The variable declaration editor in table view is shown in the figure below:



The variable declaration editor in text view is shown in the figure below:



NOTE

- Input variables can also be modified within the POU, even if the CONSTANT property has been added.
- Supports data types ANY or ANY_<Type>.

10.1.1.5 Output Variables

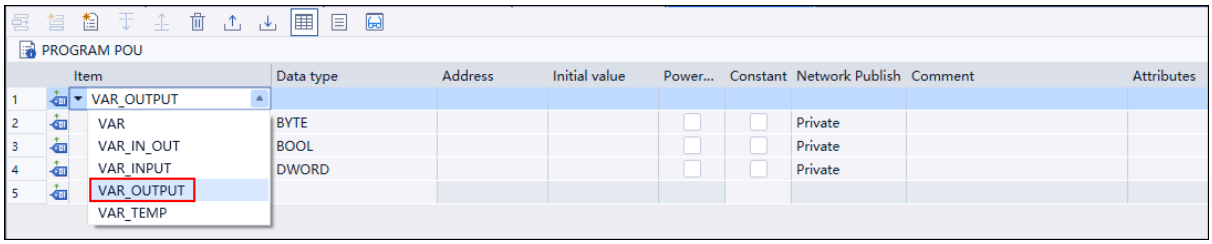
Output variables refer to variables provided internally and defined between VAR_OUTPUT and END_VAR. They are only accessible by external entities. They return values to the caller upon call for further processing.

Prerequisites

A POU has been created.

Procedure

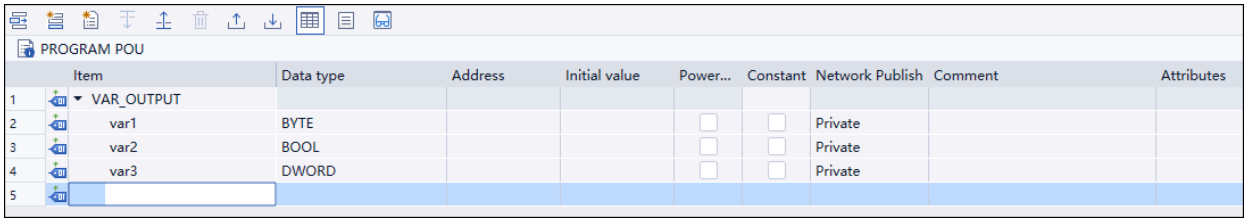
1. Open a POU (PRG, FB, or FUN) under "Programming". In the variable declaration editor in the table view, set the variable category to "VAR_OUTPUT".



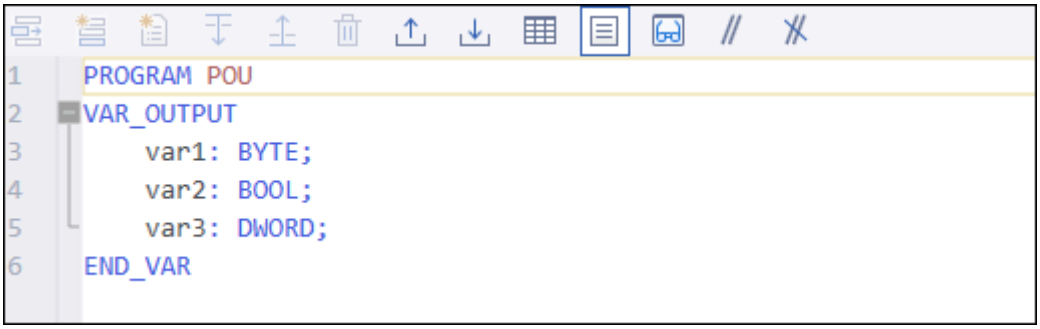
2. Add variables, and set their properties such as names and data types.

Operation result

The variable declaration editor in table view is shown in the figure below:



The variable declaration editor in text view is shown in the figure below:



10.1.1.6 Static Variables

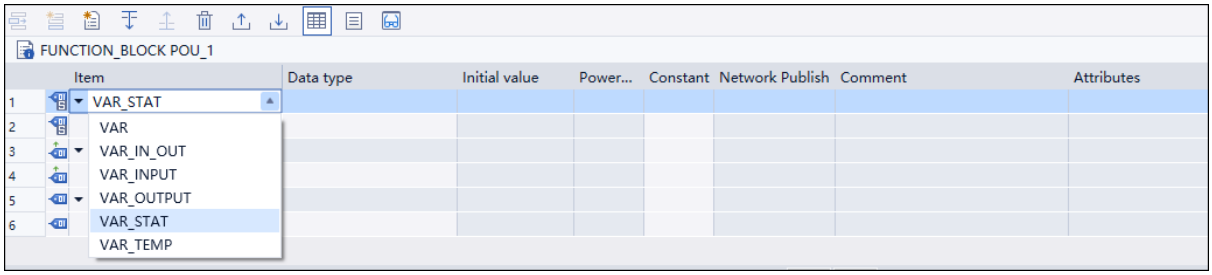
Static variables refer to variables defined between VAR_STAT and END_VAR. Stored in the global data area, they are initialized only upon the first call and retain their values after every POU call.

Prerequisites

A POU has been created.

Procedure

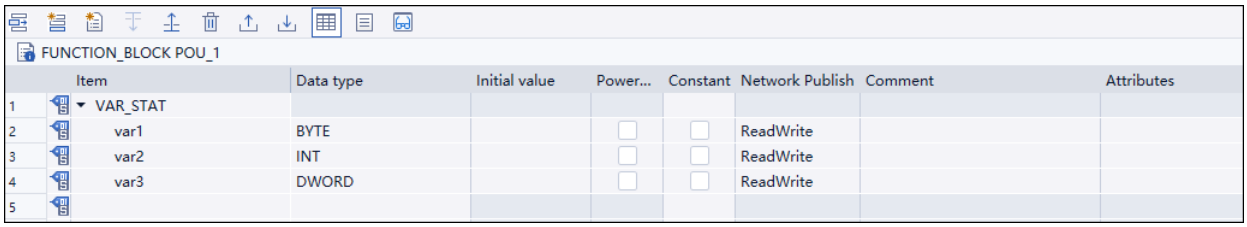
1. Open a POU (FB, FUN, or METH) under "Programming". In the variable declaration editor in the table view, set the variable category to "VAR_STAT".



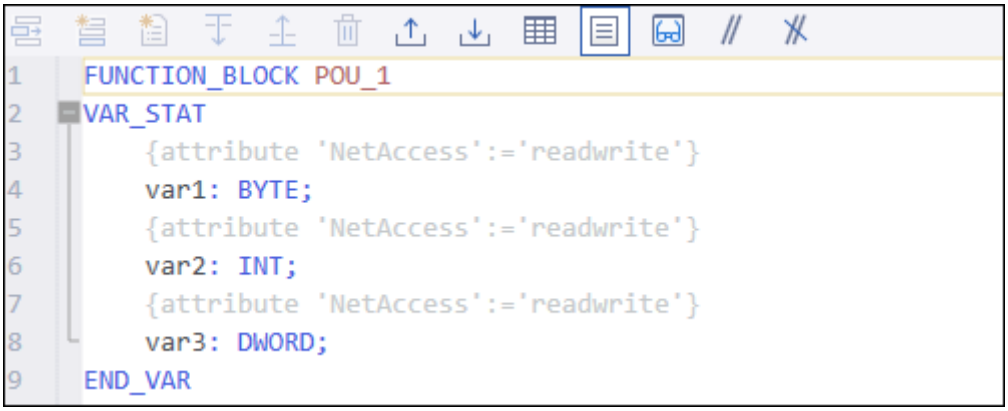
2. Add variables, and set their properties such as names and data types.

Operation result

The variable declaration editor in table view is shown in the figure below:



The variable declaration editor in text view is shown in the figure below:



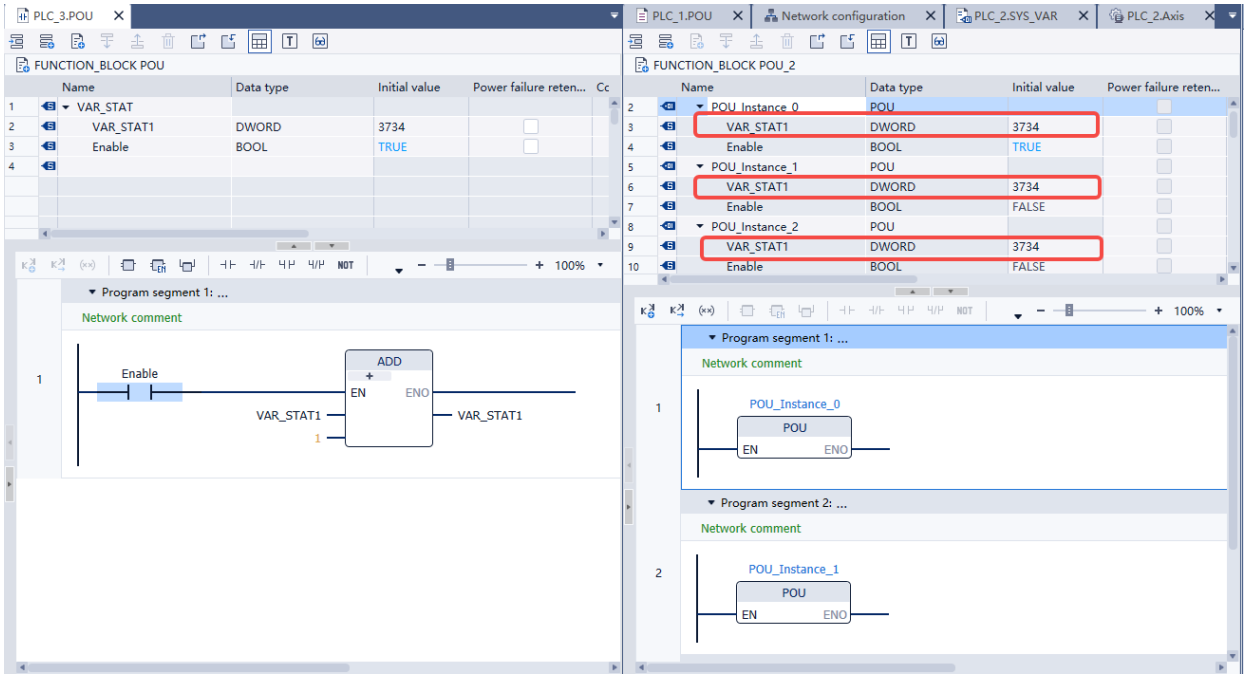
NOTE

- Static variables can only be declared in function blocks, functions, and methods, but cannot be declared in programs.
- Static variables can only be used in the declared POU.

Application Scenarios

- Static variables in FB
  Static variables declared in FB are shared across all instances of the FB.
- Static variables in FUN
  Static variables defined in FUN are unrelated to the return value of FUN. As they are in the global data area, their values can be viewed online even without setting breakpoints.

  Take self-increment operation using a static variable defined in FB as an example. Although each FB instance is independent, the static variable is stored in the global data area and is shared across all instances. Updating it in one instance affects all others.



10.1.1.7 Input/Output Variable

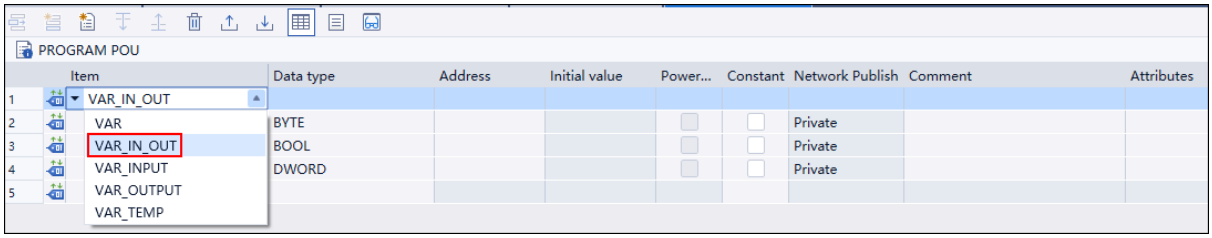
Input/output variables refer to variables defined between VAR_IN_OUT and END_VAR. They can be transferred to the called POU and modified within the called POU. The variables transferred to the called POU are referenced by the caller.

Prerequisites

A POU has been created.

Procedure

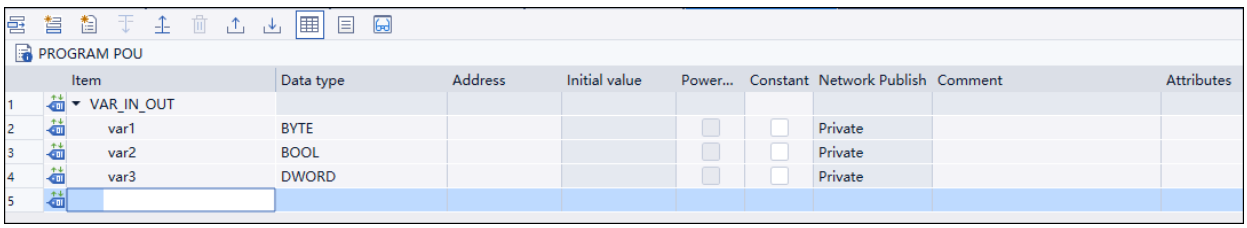
1. Open a POU (PRG, FB, or FUN) under "Programming". In the variable declaration editor in the table view, set the variable category to "VAR_IN_OUT".



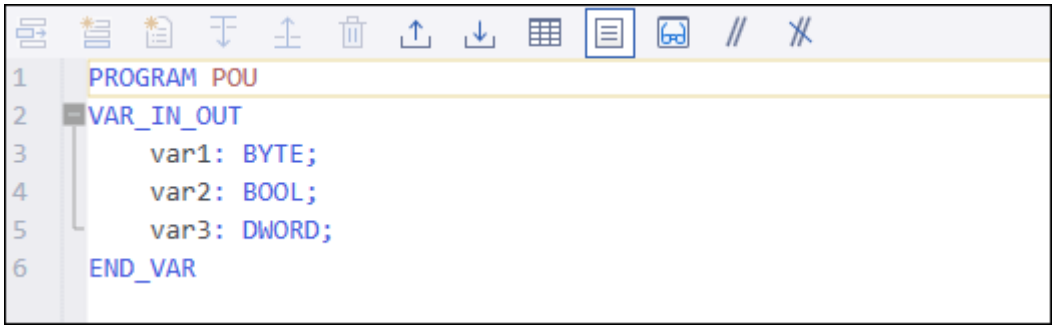
2. Add variables, and set their properties such as names and data types.

Operation result

The variable declaration editor in table view is shown in the figure below:



The variable declaration editor in text view is shown in the figure below:



NOTE

The input and output variables are non-public variables.

10.1.1.8 Data Types (EVO400, EVO500, EVO700, EVO800)

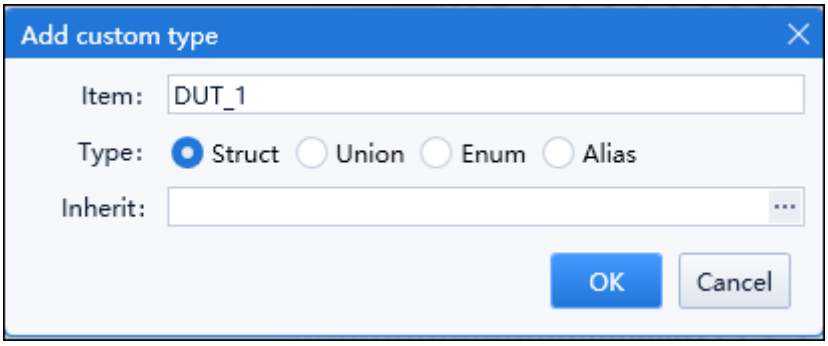
10.1.1.8.1 Standard Data Types

Data Type	Value Range	Memory Usage (Byte)	Input Example
BOOL	TRUE, FALSE	1	TRUE FALSE
BYTE	0 to 255	1	BYTE#2#0000_1111
WORD	0~65535	2	WORD#61_680
DWORD	0 to 4294967295	4	DWORD#2#0000_0000_1111_0000_1111_1111_0000_1111
LWORD	0 to 18446744073709551615	8	8#13_724_557_213
SINT	-128 to 127	1	100 INT#2#00000000_01100100 INT#8#144 INT#10#100 INT#16#64 -100
USINT	0 to 255	1	
INT	-32768~32767	2	
UINT	0~65535	2	
DINT	-2147483648 to 2147483647	4	
UDINT	0 to 4294967295	4	
LINT	-9223372036854775808 to 9223372036854775807	8	
ULINT	0 to 18446744073709551615	8	
REAL	-3.402823e+38 to -1.175495e-38 ±0.0 +1.175495e-38 to +3.402823e+38 +Inf/-Inf	4	-3.14 1.0e-5 LREAL#1.0e-5

Data Type	Value Range	Memory Usage (Byte)	Input Example
LREAL	-1.7976931348623157e+308 to -2.2250738585072014e-308 ±0.0 +2.2250738585072014e-308 to +1.7976931348623157e+308 +Inf/-Inf	8	
STRING	UTF-8 encoding 0 to 4096 bytes (default is 80 bytes)	1	'This is a String' Use single quotes with the English input method.
WSTRING	UNICODE encoding 0 to 16382 bytes (default is 160 bytes)	2	"This is a WString" Use double quotes with the English input method.
TIME	T#0d0h0m0s0ms to T#49d17h2m47s295ms	4	T#10d_20h_30m_20s_630ms T#61m5s T#3s20ms TIME#10d_20h_30m_20s_630ms TIME#6d_10m
TIME_OF_DAY (TOD)	TOD#00:00:00.000 to TOD#23:59:59.999	4	Start with TIME_OF_DAY#, time_of_day#, TOD#, or tod# and followed by a time in hh:mm:ss format. tod#12:50:30.14
DATE	D#1970-01-01 to D#2106-02-07	4	Start with DATE#, date#, D#, or d#, and followed by a date in yyyy-mm-dd format. D#2009-12-31, DATE#2009-12-31
DATE_AND_TIME (DT)	DT#1970-01-01-00:00:00 to DT#2106-02-07-06:28:15	4	Start with DT# or dt#, and followed by a date and time in yyyy-mm-dd-hh:mm:ss format. dt#2007-07-01-05:40:25
LTIME	LTIME#0NS to LTIME#213503D23H34M33S709MS551US615NS	8	LTIME#0NS LT#0NS

10.1.1.8.2 User-Defined Types

Right-click "Programming" in the project tree and select "Add user-defined type" to add four types of custom data types: structure, enumeration, union, and alias.



10.1.1.8.2.1 STRUCT

Concepts

A structure is a collection composed of data items of the same type or different types.

It is similar to basic data types, except that it allows custom encapsulation of data types.

By operating on the internal variables of a structure, a large amount of data can be stored in memory.

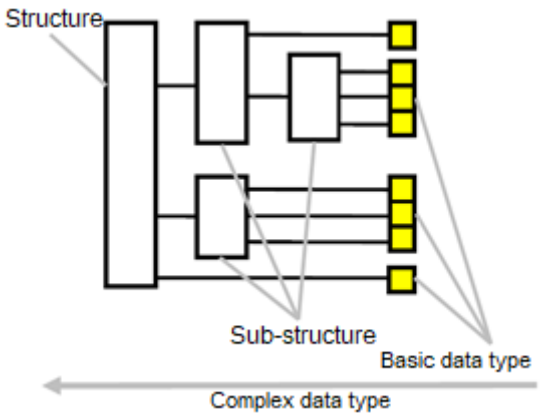
For example, a motor has information such as "Product_ID", "Vendor", "Nominal Voltage", "Nominal Current", and "Poles". Combine them through a structure to reflect the relationship among the variables.

The syntax for the struct declaration is as follows:

```
TYPE <structurename> | EXTENDS DUTTYPE:
STRUCT
   <declaration of variables 1>
   ...
   <declaration of variables n>
END_STRUCTEND_TYPE
```

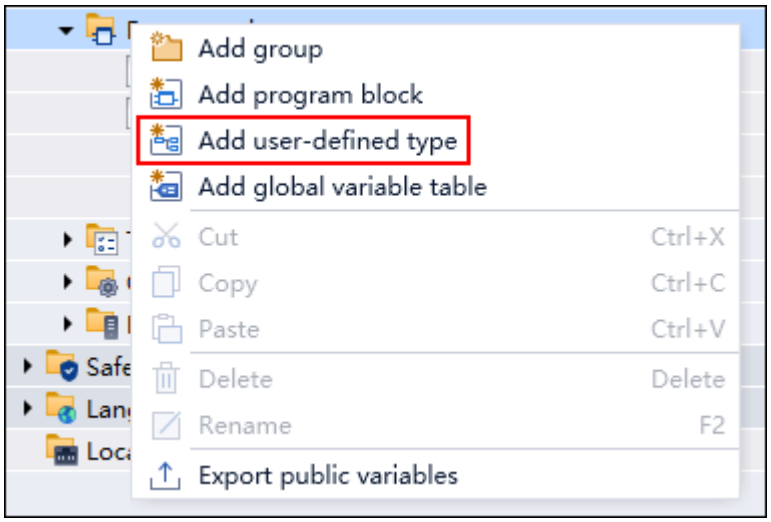
<structurename> is a type and can be used as a data type. EXTENDS DUTTYPE is optional and indicates inheritance from the members of DUTTYPE. Variables of the structure name type can be used to access the members of DUTTYPE. DUTTYPE is of the structure, union, or alias type.

A structure is extensible and supports inheritance. It can inherit from another structure, where the inherited one is the parent and the inheriting one is the child. This process is also called structure nesting. Once nested, the structure is considered a complex data type.

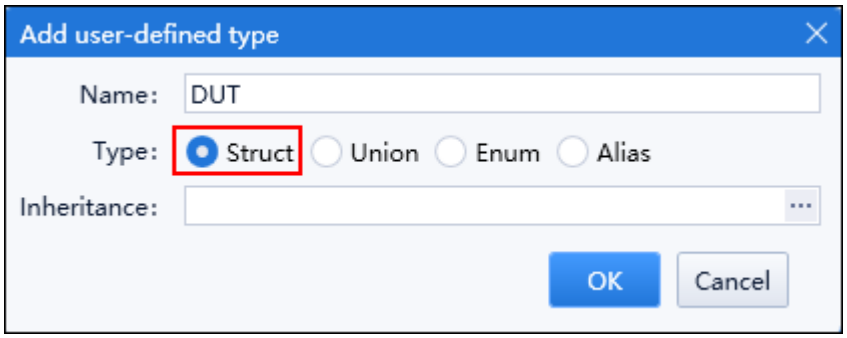


Structure definition

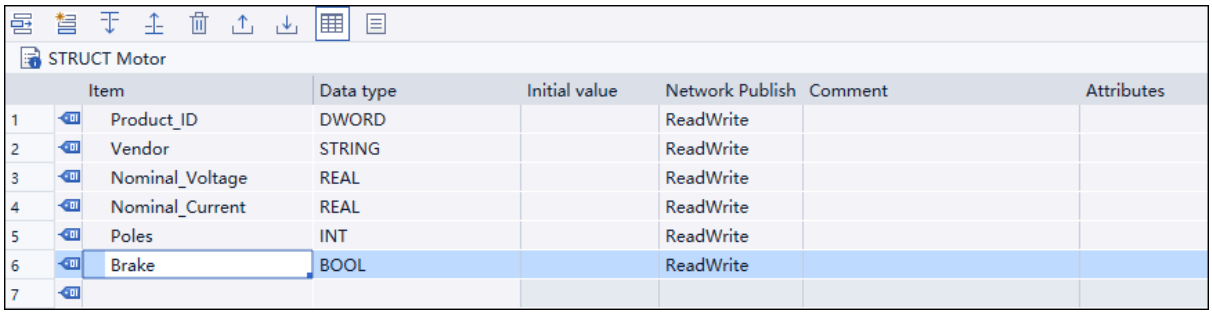
1. Right-click "Programming" in the project tree and select "Add user-defined type".



2. In the "Add user-defined type" window, set the name of the structure, select the type as "Struct", and select Inheritance as needed. Click "OK", and the system automatically enters the DUT editor.



3. Create a structure named "Motor" in the DUT editor, as shown below.



	Item	Data type	Initial value	Network Publish	Comment	Attributes
1	Product_ID	DWORD		ReadWrite		
2	Vendor	STRING		ReadWrite		
3	Nominal_Voltage	REAL		ReadWrite		
4	Nominal_Current	REAL		ReadWrite		
5	Poles	INT		ReadWrite		
6	Brake	BOOL		ReadWrite		
7						

> **NOTE**
>
> The EVO500 series PLC V1.2 version does not support setting the "Network public" property of a structure.

Struct inheritance

Structures can inherit structure types or alias structures. There are two methods to add "Inheritance" to a structure:

- Method 1: Set in the "Add user-defined type" window.
- Method 2: Set in the variable declaration editor.

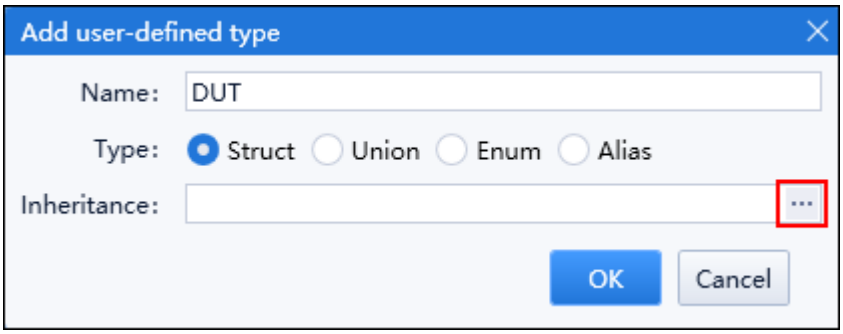
Method 2 adds inheritance to a structure that already exists.

Prerequisites

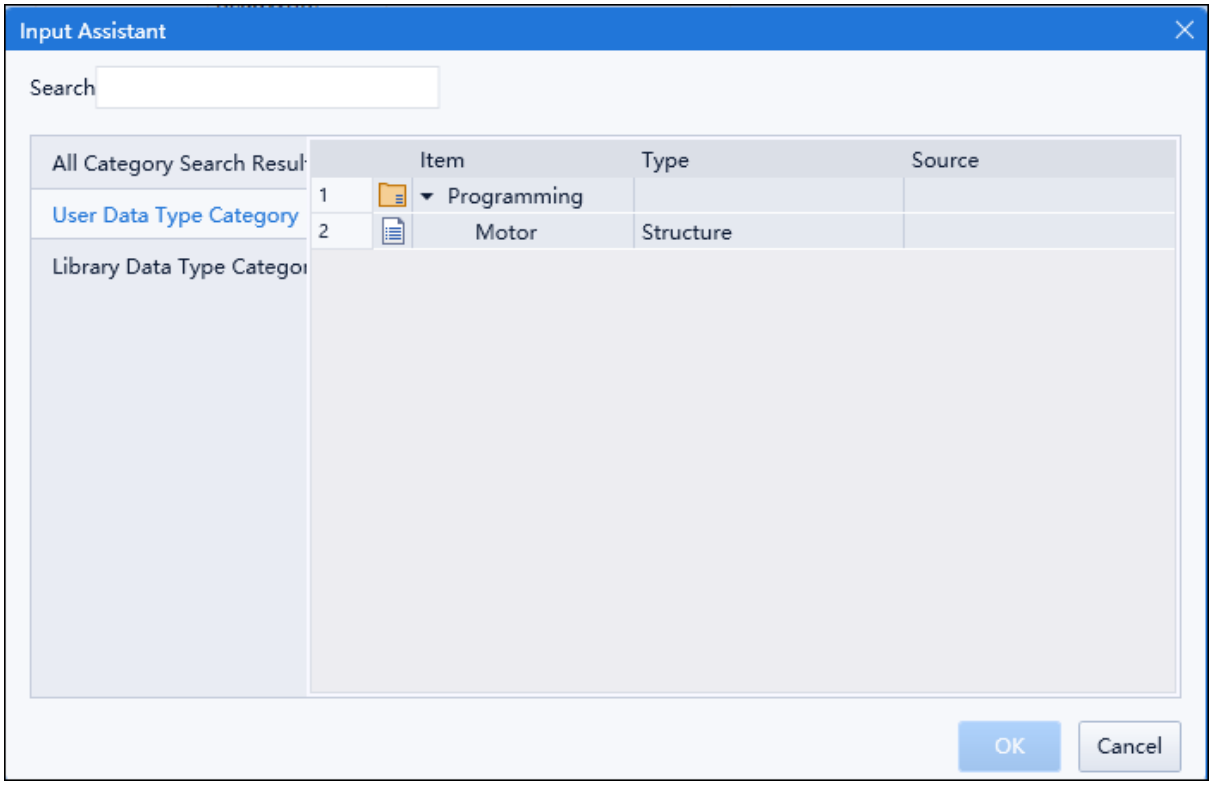
- Method 1: A structure to serve as the base must already exist.
- Method 2: Both a base structure and a child structure must already exist.

Procedure to add inheritance in the "Add user-defined type" window

1. Right-click "Programming" in the project tree and select "Add user-defined type".
2. In the pop-up window, select "Struct" as the type.
3. Click the "..." button after "Inheritance" to open the "Input assistant" window.

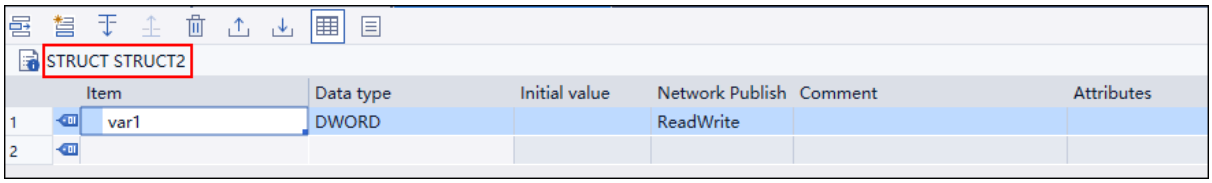


4. Select the inherited structure in the "Input assistant", as shown in the following figure.

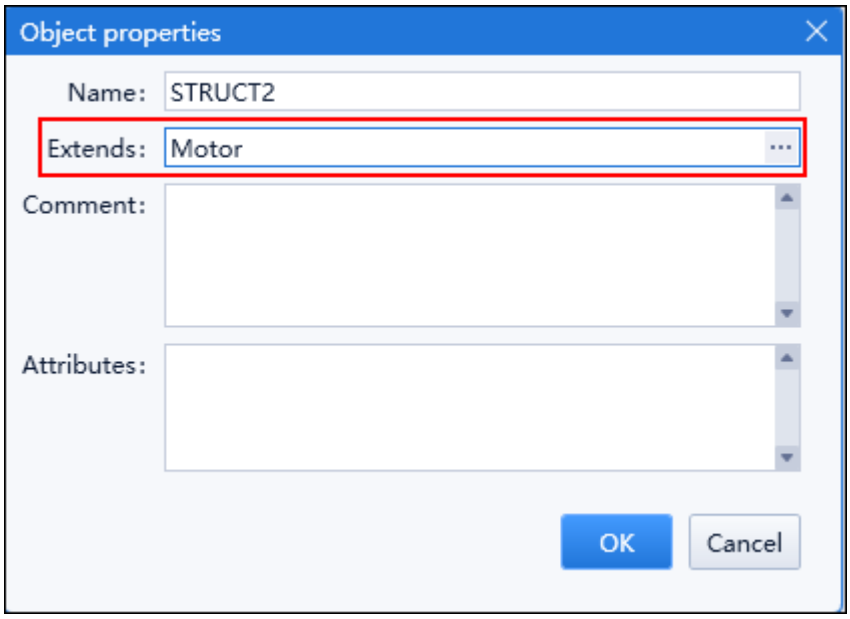


Procedure to add inheritance in the variable declaration editor

1. In the table view of the variable declaration editor, click the name of the structure and the "Object property" window pops up.



2. Add the structure being inherited, as shown below.



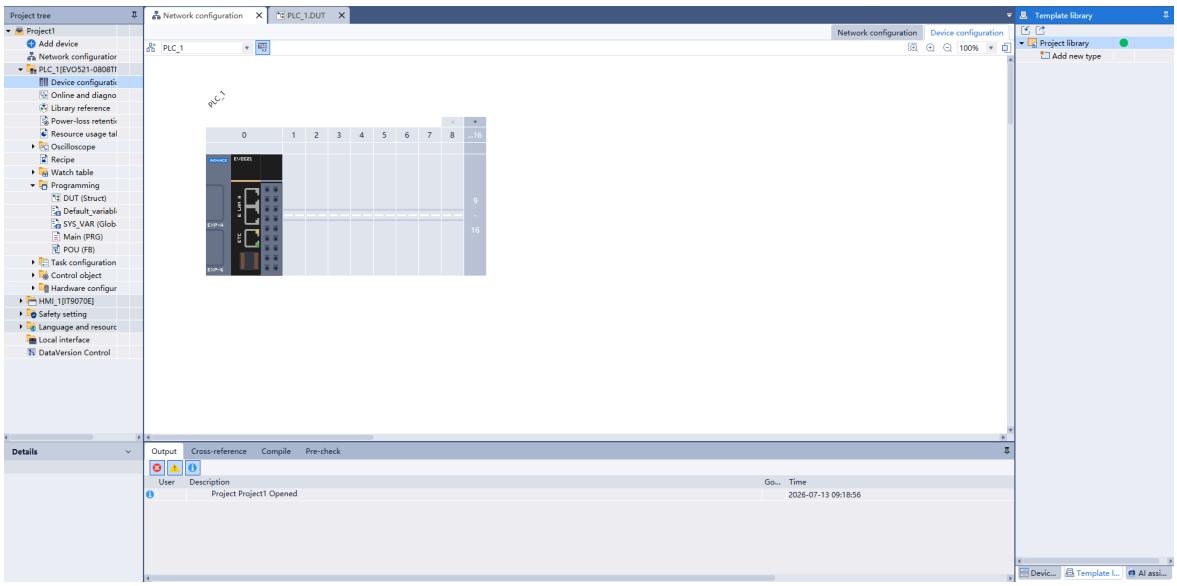
Structural templating

User-defined data types in PLC support templating, allowing common resources to be reused across multiple projects during configuration.

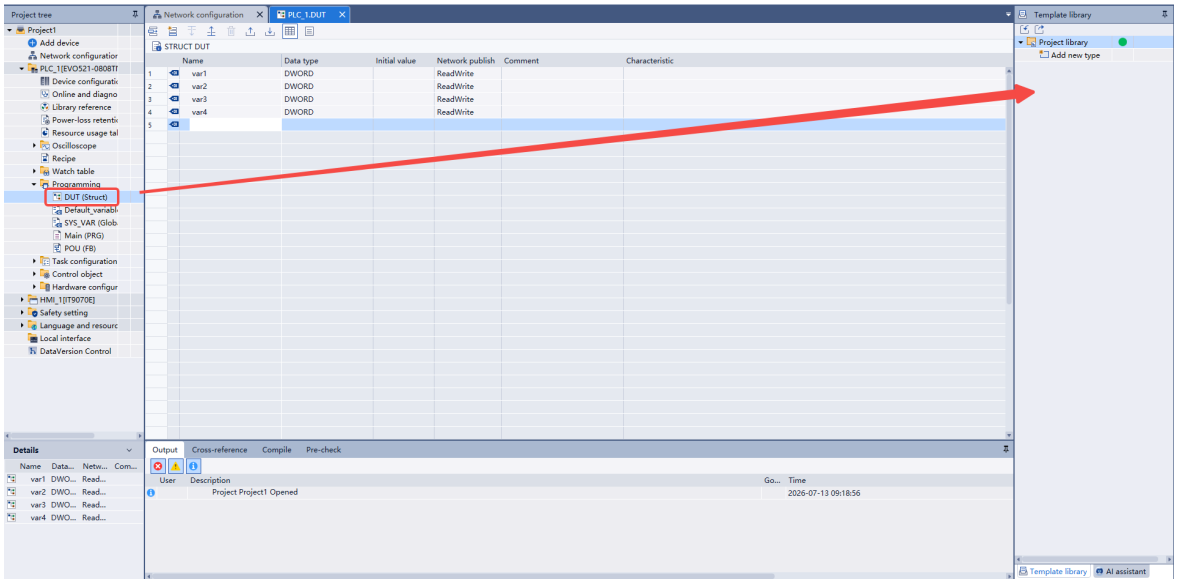
Creating a template

- **Prerequisites**
  - User-defined data types have been created or imported in the current project.
  - Only user-defined 10.1.1.8.1 standard data types and array-type structures in PLC devices can be templated. Arrays support multi-dimensional and nested arrays, with a maximum of 2000 elements. Standard data types with specified upper/lower limits and arrays with non-numeric boundaries are not supported.
  - Enumerations, unions, aliases, FBs, FCs, PRGs, and library types are not supported.
  - Structure inheritance is not supported.
- **Procedure**

1. Open the project view and select the "Template library" tab on the right.

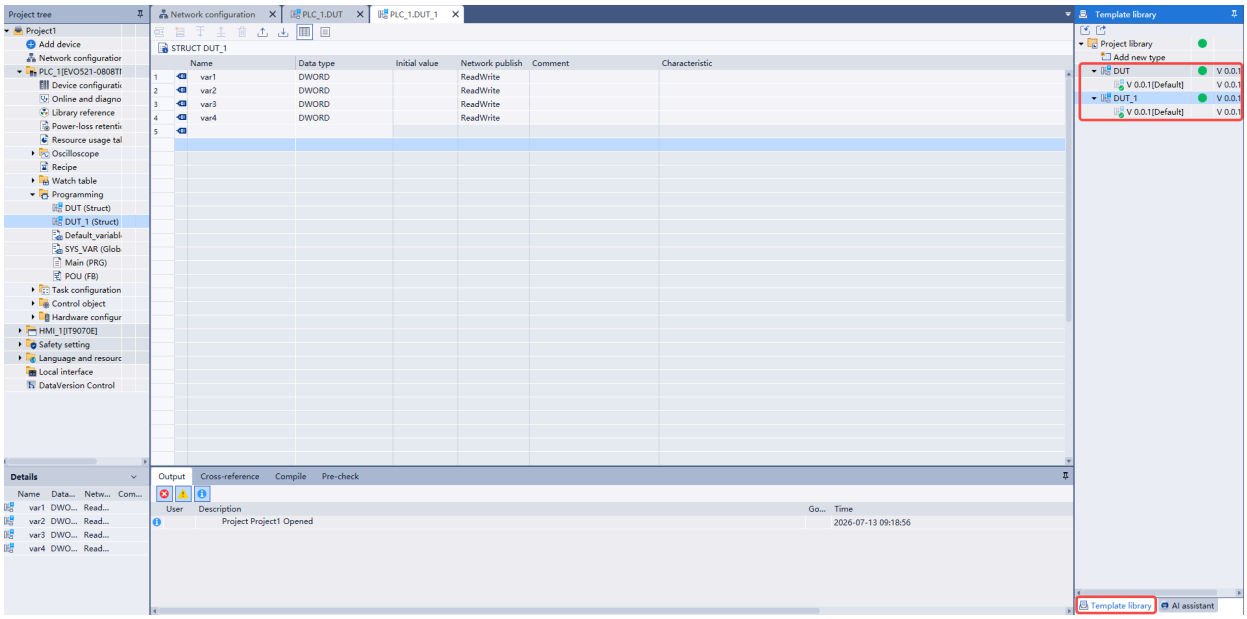


2. Select the user-defined data type for templating under the "Programming" node in the left project tree, and drag it to the "Template library".



- **Result**
  - Create the corresponding user-defined type under "Template library > Project library", with version V0.0.1. The user-defined type nodes in both the project tree and the template library are marked with .

- When editing a templated PLC type, a prompt appears indicating that the content is associated with a type in the template library, and that any changes made must be manually synchronized.



NOTE

- During templating, if the referenced object contains nested user-defined sub-structures, all nested sub-structures will be automatically templated.
- A templated user-defined data type cannot be re-templated, ensuring that each structure type in the project tree maps to a unique type in the template library.

Positioning a template

When multiple user-defined types exist in the project tree, you can quickly view the corresponding type in the template library. Right-click a type and select "Go to type" to jump to the corresponding node in the template library.

Disconnecting from a template

Right-click the user-defined type to disconnect in the project tree and select "Disconnect from the type" to disconnect this type and the corresponding type in the template library. After disconnection, this type can no longer be used to update instances, and the node identifier of this type becomes .

Deleting a template

Right-click the user-defined type in the template library and select "Delete" to remove the template.

NOTE

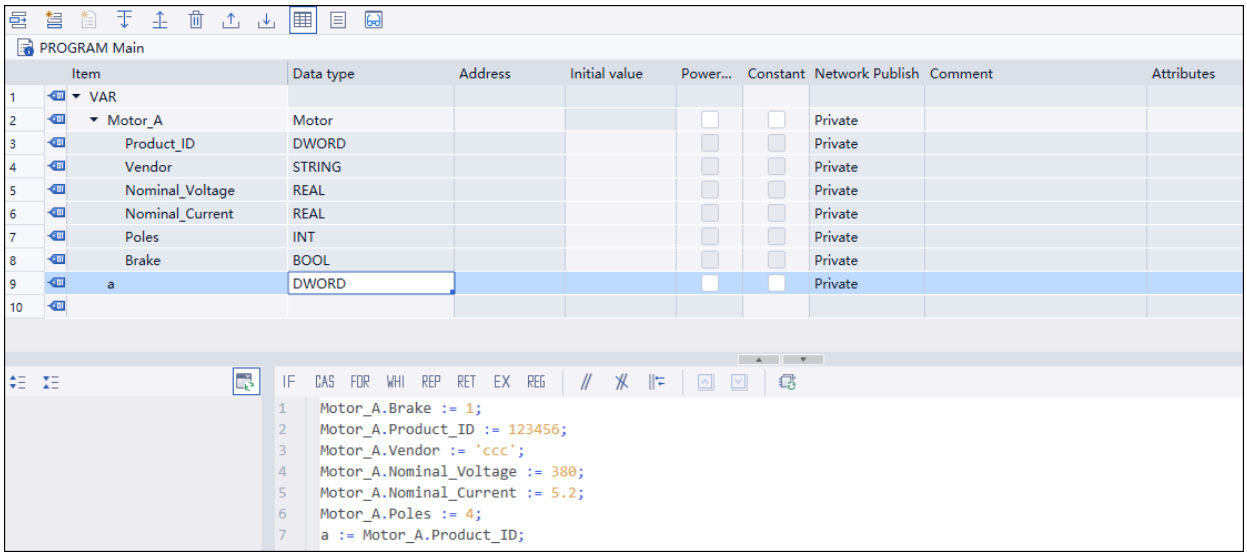
- Deleting a user-defined type from the template library will not delete the corresponding type or its instances from the project tree.
- Deleting a user-defined type from the project tree will not delete the corresponding type from the template library.

Using a template

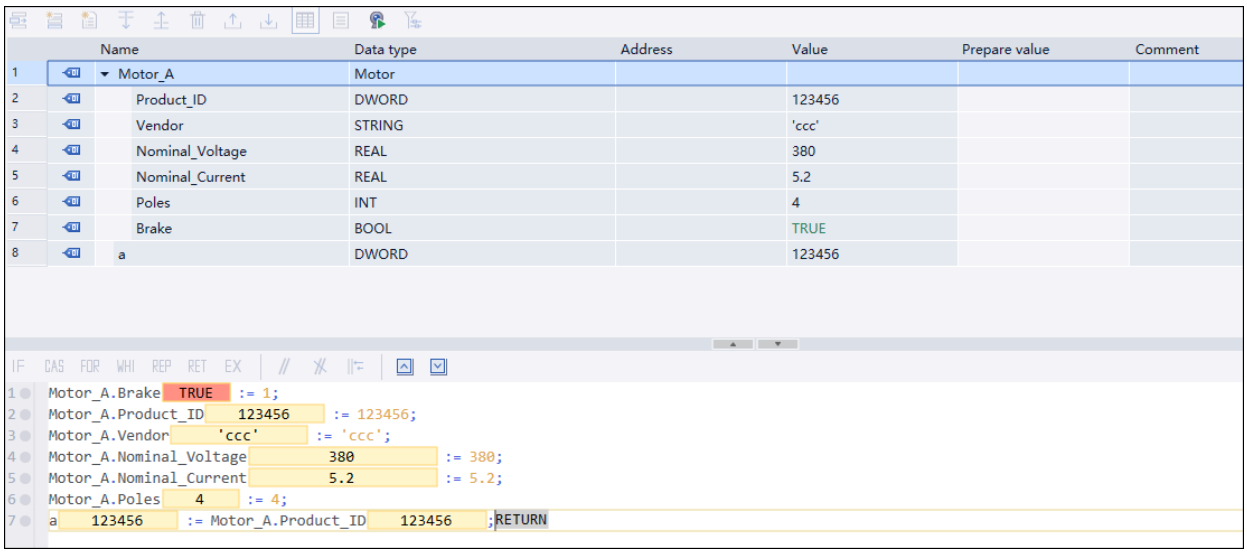
After the template is created, it can be instantiated and used in other PLC devices. For specific usage, see 13.4.4.4 Applying a User-defined Type.

Structure instantiation

After a structure is defined, create a variable in the program with the type set to the newly created structure name, that is "Motor". In the "PROGRAM Main" window, enter "VariableName." (the variable name followed by a dot). The system will automatically display a dropdown list of the structure's members. Select the target variable and use assignment statements to perform read or write operations on the structure, as shown below.



Go online and run the program to monitor the operation data. As shown in the figure below, the member assignment of the variable Motor_A of type Motor succeeded.



NOTE

- During structure instantiation, the "Network public" property (options include "Public" and "Private") is set to "Private" by default. If the parent node is set to "Private", all child nodes will also be "Private"; if the parent node is set to "Public", the child nodes will inherit the public property configured for the structure itself.
- During structure instantiation for the EVO500 series PLC V1.2 version, the "Network public" property (options include "Private", "Read/Write", "Read-Only", and "Write-Only") is set to "Private" by default. The child node properties inherit from the parent node and cannot be set separately. If the device version is downgraded from V1.3 to V1.2, a fault of public variable property compilation will be reported.

10.1.1.8.2.2 UNION

Concepts

Unions are used to store different types of variables in the same memory unit. The following table shows an example where a BYTE variable, an INT variable, and a DWORD variable are stored in the memory unit starting at address 16#100. The merged part in the table represents the coverage domain region. Such a structure where different variables share the same memory region is called a union.

	16#100	16#101	16#102	16#103
BYTE (8 bits)				
INT (16 bits)				
DWORD (32 bits)				

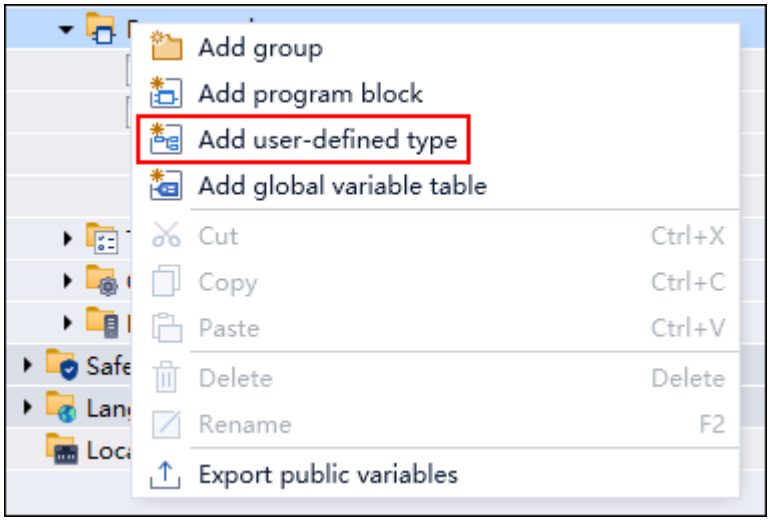
The syntax for the union declaration is as follows:

```
TYPE <unionname>:UNION
   <declaration of variables 1>
   ...
   <declaration of variables n>
END_UNIONEND_TYPE
```

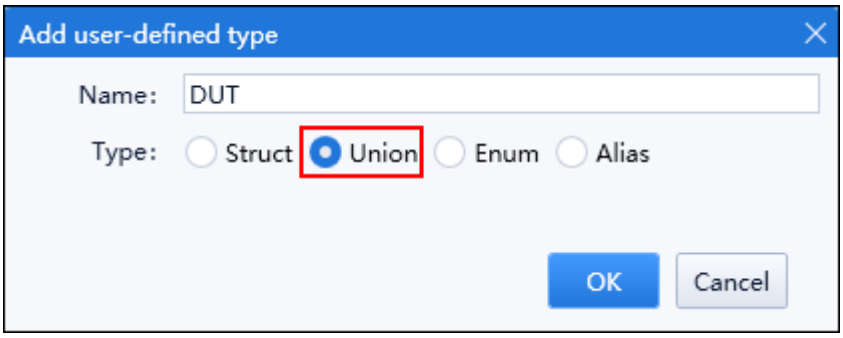
<unionname> is a type and can be used as a data type. All variables of the union type share the same storage location and are allocated with space same as that of the variable that occupies the largest space. The size of the union in the above table is based on DWORD, which is 32 bits.

Union definition

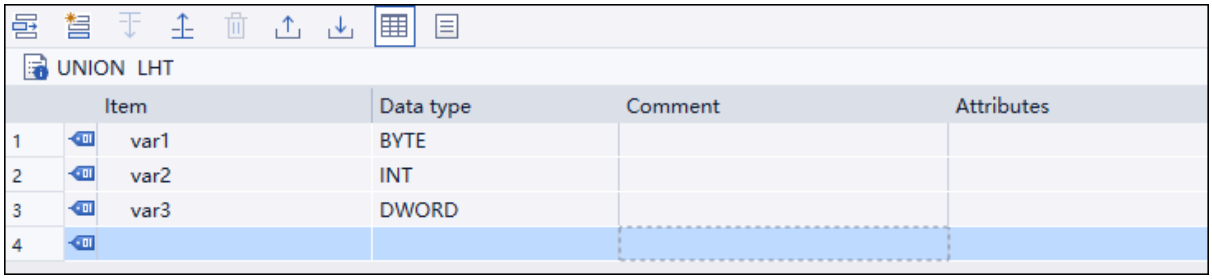
1. Right-click "Programming" in the project tree and select "Add user-defined type".



2. In the "Add user-defined type" window, set the name of the union and select "Union" as the type. Click "OK", and the system automatically enters the DUT editor.



3. Create a union named LHT in the DUT editor, as shown below.

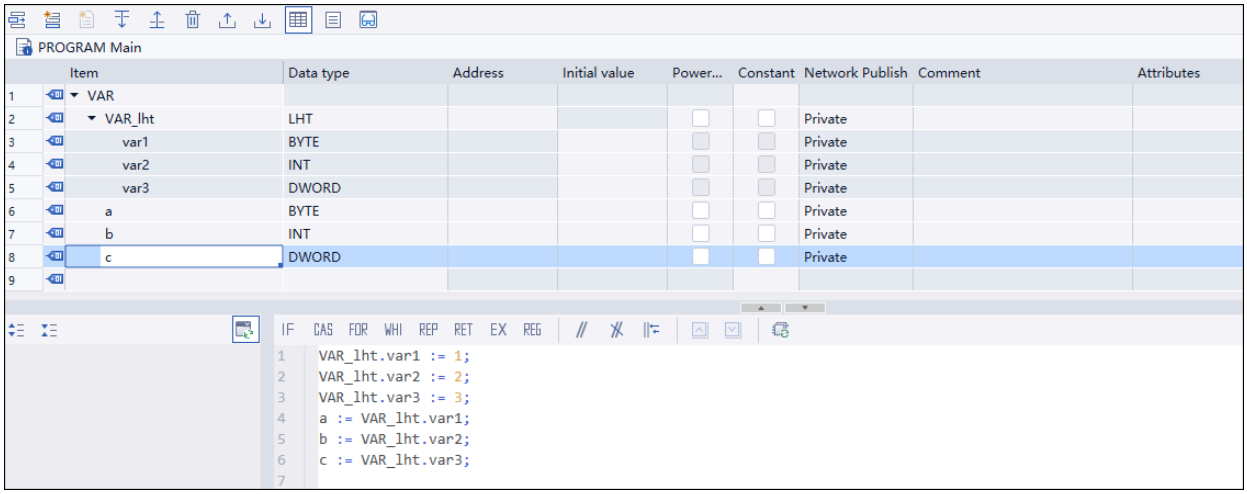


NOTE

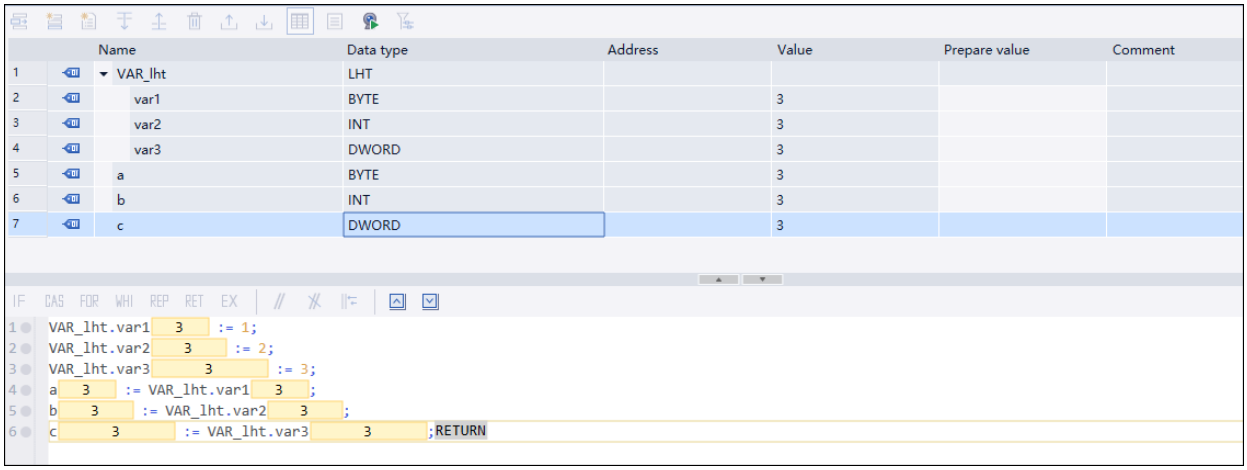
Unions support defining one-dimensional BIT arrays, which can be used for bit-level data access.

Union usage

After a union is created, create a variable in the program with the type set to the newly created union name, that is LHT. In the "PROGRAM Main" window, enter "VariableName." (the variable name followed by a dot). The system will automatically display a dropdown list of the union's members. Select the target variable and use assignment statements to perform read or write operations on the union, as shown below.



Go online, click the "Run" icon, and monitor the running data, as shown in the following figure.



10.1.1.8.2.3 ENUM

Concepts

If a variable has several possible values, it can be defined as an enumeration type. Enumeration lists the possible values of a variable one by one. The variable's value is limited to the listed range. For example, define a variable whose value represents the four seasons of a year. This variable can only store four meaningful values. To define these values, you can use an enumeration type. The set of possible values for the four seasons is: {Spring, Summer, Autumn, Winter}

This set can be defined as an enumeration type describing the seasons, which contains four elements. A variable declared with this type is restricted to a single value from the set.

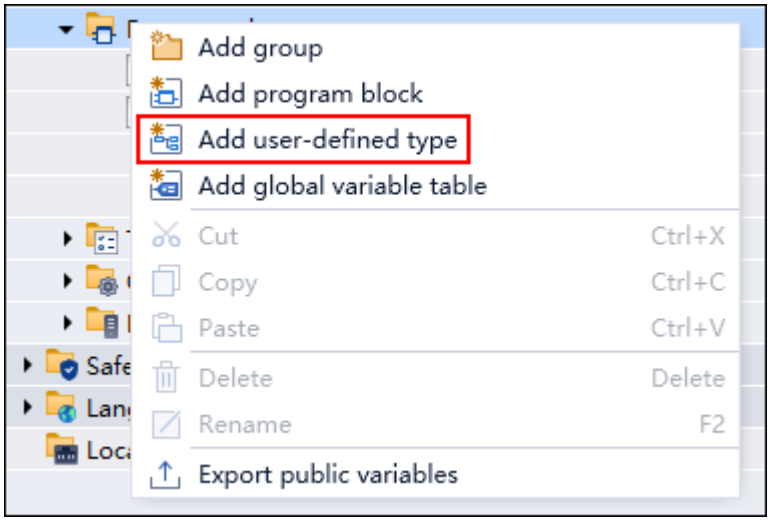
The syntax for an enumeration declaration is as follows:

```
TYPE <identifier>:(<enum_0> ,<enum_1>, ...,<enum_n>) |<base
data type>;END_TYPE
```

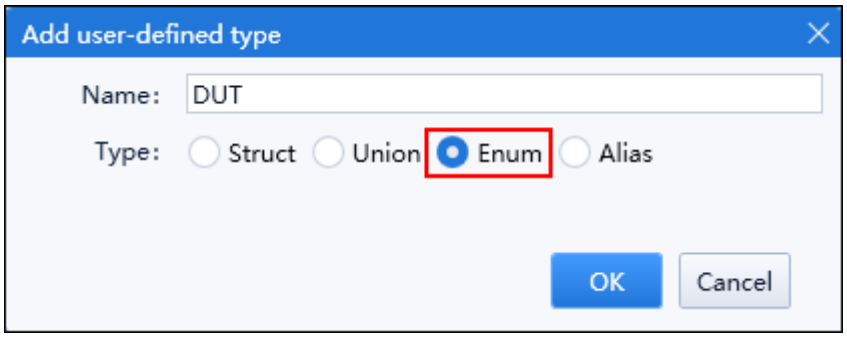
identifier: user-defined enumeration type; enum_n: constant value of the enumeration type. Each constant can declare its value. If no value is declared, the default value is used. The data type of enumerated constants is base data type. The value may not be declared and is an integer by default.

Enumeration definition

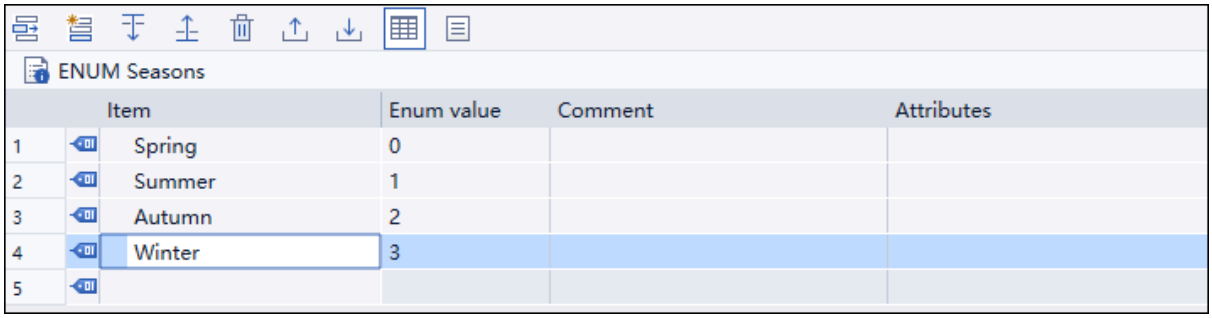
1. Right-click "Programming" in the project tree and select "Add user-defined type".



2. In the "Add user-defined type" window, set the name of the enumeration and select "Enum" as the type. Click "OK", and the system automatically enters the DUT editor.



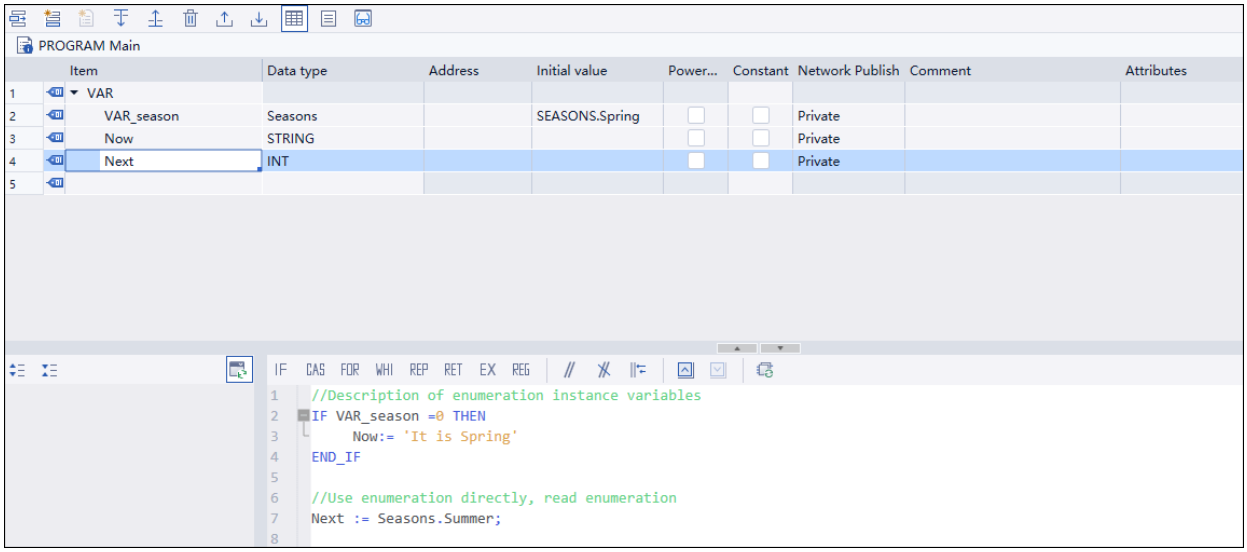
3. Create an enumeration named Seasons in the DUT editor, as shown below.



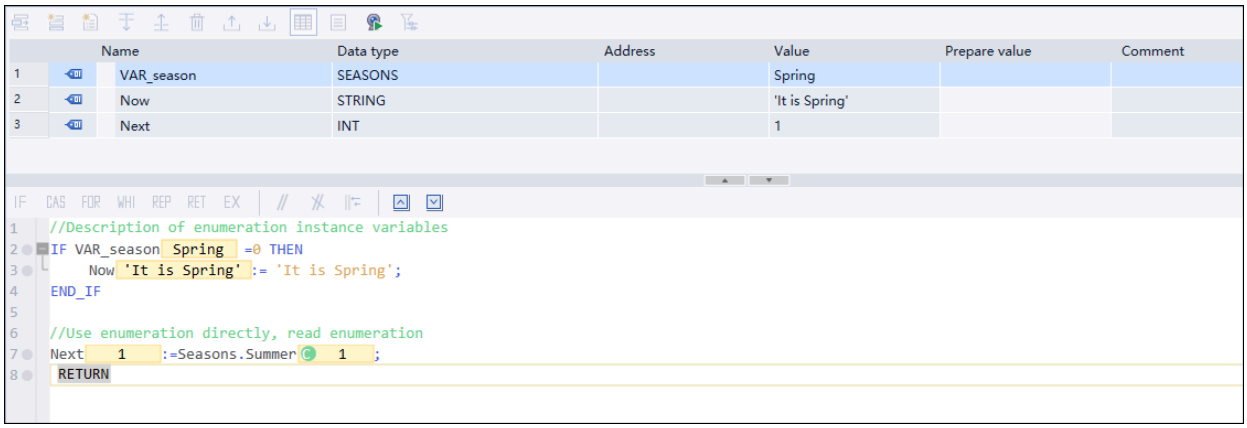
Enumeration usage

After an enumeration is created, create a variable in the program with the type set to the newly created enumeration name, that is "Seasons". Select the "Initial value" in the variable table. In the editor, a judgment statement uses an instance of the enumeration variable "Seasons".

You can also directly use the enumeration for read operations, as shown on line 7 in the figure.



Go online, click the "Run" icon, and monitor the running data, as shown in the following figure.



10.1.1.8.2.4 ALIAS

Concepts

An alias represents a data type by another name.

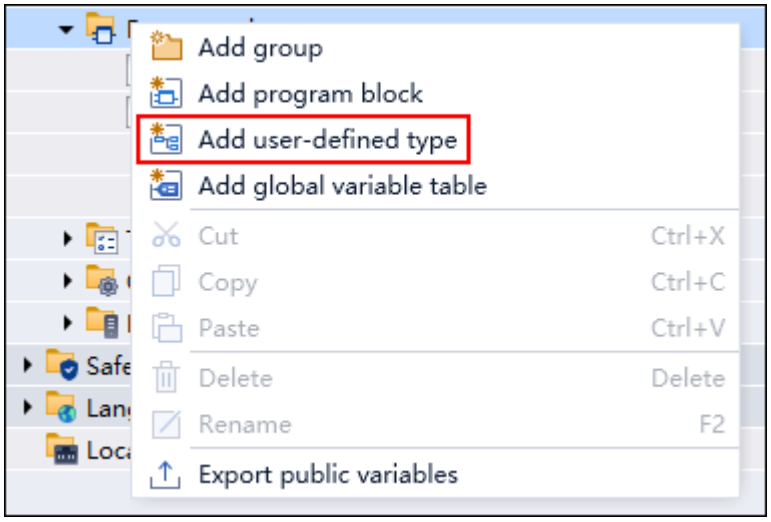
The syntax for the alias declaration is as follows:

```
TYPE <aliasname>:basetype END_TYPE
```

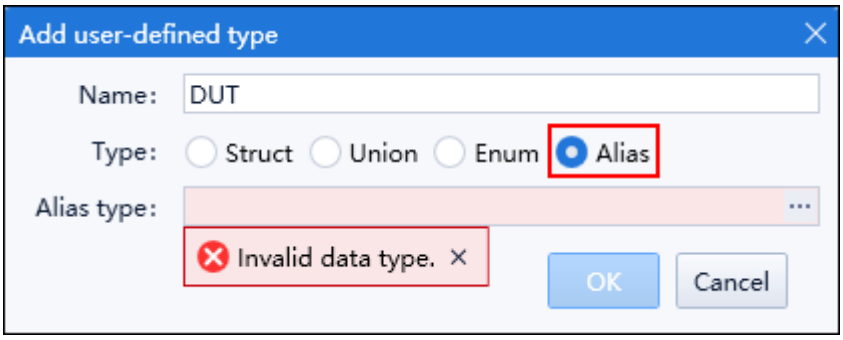
"aliasname" indicates the alias type and is used as a data type. "basetype" is a standard, user-defined data type, or FB.

Alias definition

1. Right-click "Programming" in the project tree and select "Add user-defined type".



2. In the "Add user-defined type" window, set the name of the alias, select the type as "Alias", and select "Alias type" as needed. Click "OK", and the system automatically enters the DUT editor.

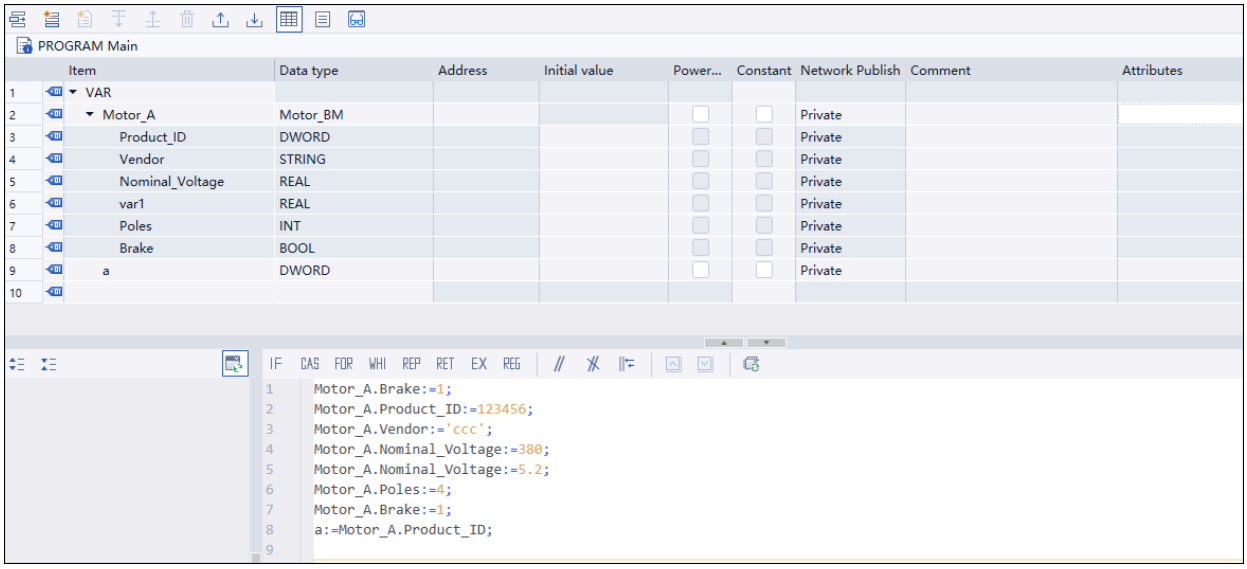


3. In the DUT editor, configure an alias for the structure Motor as follows.

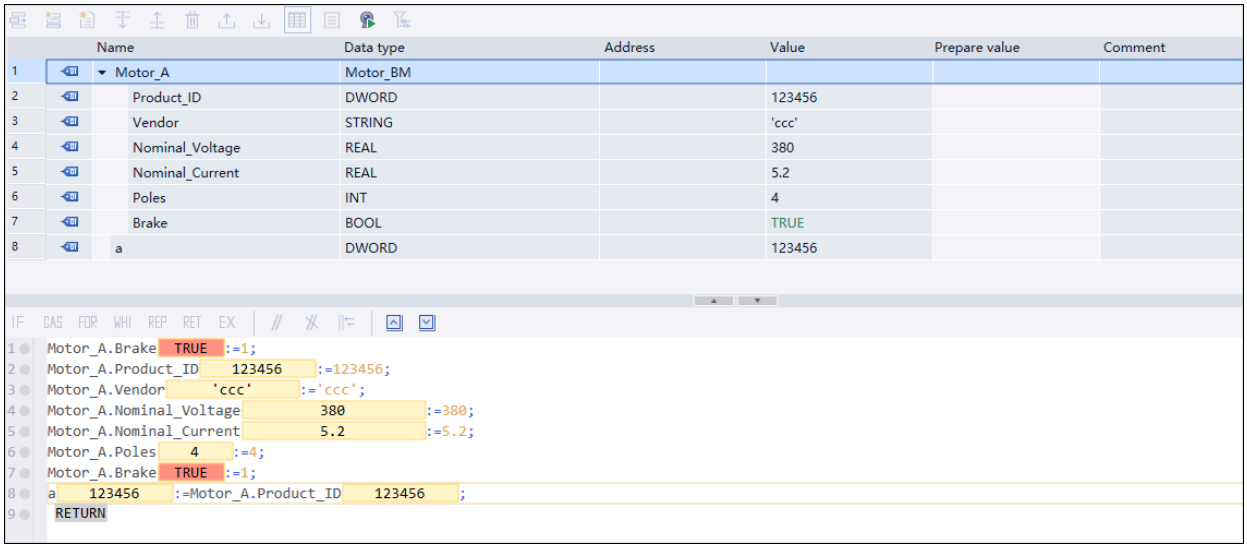


Alias usage

After an alias is created, create a variable in the program with the type set to the newly created alias name, that is "Motor_BM". In the "PROGRAM Main" window, enter "VariableName." (the variable name followed by a dot). The system will automatically display a dropdown list of the union's members. Select the target variable and use assignment statements to perform read or write operations on the union, as shown below. The interface is shown in the following figure.



Go online, click the "Run" icon, and monitor the running data, as shown in the following figure.



10.1.1.8.3 Complex Types

10.1.1.8.3.1 ARRAY

ARRAY refers to the data structure consisting of a fixed number of elements with the same data type. These elements can be of all data types except ARRAY.

Array variable definition

The syntax for declaring an array-type variable is as follows:

<Array_Name>:ARRAY [<ll1>..<ul1>,<ll2>..<ul2>,<ll3>..<ul3>] OF <elem. Type>

ll1, ll2, and ll3 define the lower limits of the array dimensions, while ul1, ul2, and ul3 define the upper limits. They must be integers. Type defines the data type of each array element.

Array variable usage

- Array initialization example
  Card_game: ARRAY [1..13, 1..4] OF INT;
  arr1 : ARRAY [1..5] OF INT := [1,2,3,4,5];
  arr2 : ARRAY [1..2,3..4] OF INT := [1,3(7)]; (* Array value 1,7,7,7*)

arr3 : ARRAY [1..2,2..3,3..4] OF INT := [2(0),4(4),2,3]; (* Array value 0,0,4,4,4,4,2,3*)

arr4 : ARRAY [1..10] OF INT := [1,2]; (* Partially initialized; uninitialized elements default to 0 *)

- Array structure initialization example

  Structure definition:

PROGRAM Main

	Item	Data type	Address	Initial value	Power...	Constant	Network Publish	Comment	Attributes
1	VAR								
2	arr1	ARRAY[1..3] OF STRUCT1					Private		
3	arr1[1]	STRUCT1					Private		
4	p1	INT		1			Private		
5	p2	INT		10			Private		
6	p3	DWORD		4723			Private		
7	arr1[2]	STRUCT1					Private		
8	p1	INT		2			Private		
9	p2	INT		0			Private		
10	p3	DWORD		299			Private		
11	arr1[3]	STRUCT1					Private		
12	p1	INT		14			Private		
13	p2	INT		5			Private		
14	p3	DWORD		112			Private		

  Syntax of array structure initialization:

  arr1:ARRAY[1..3] OF STRUCT1:= [(p1:=1,p2:=10,p3:=4723),(p1:=2,p2:=0,p3:=299),(p1:=14,p2:=5,p3:=112)];

- Syntax of accessing union elements

  <Array-Name>[Index1,Index2].

  Example: Card_game [9,2].

Variable-length fixed-type array definition

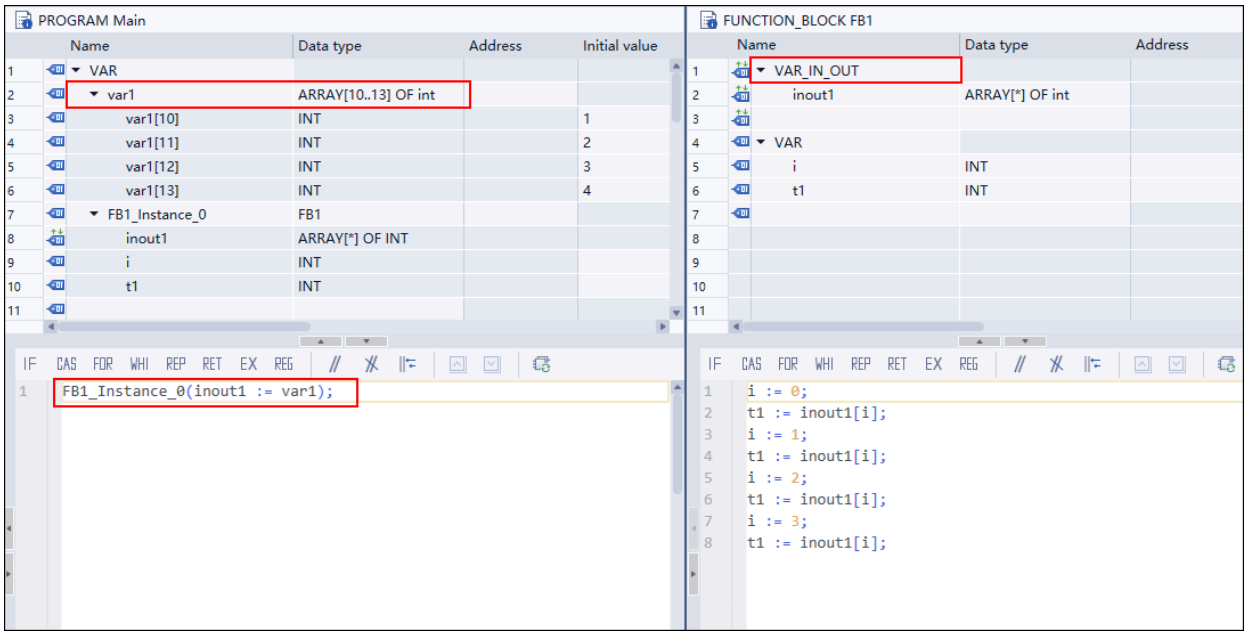
A variable-length fixed-type array is an array with variable length, whose elements are of a fixed type. It can only be used for input/output variables (VAR_IN_OUT) in FB or FUN.

The syntax for declaring a variable-length fixed-type array is as follows:

```
VAR_IN_OUT
var1: ARRAY[*] OF specific type;
END_VAR
```

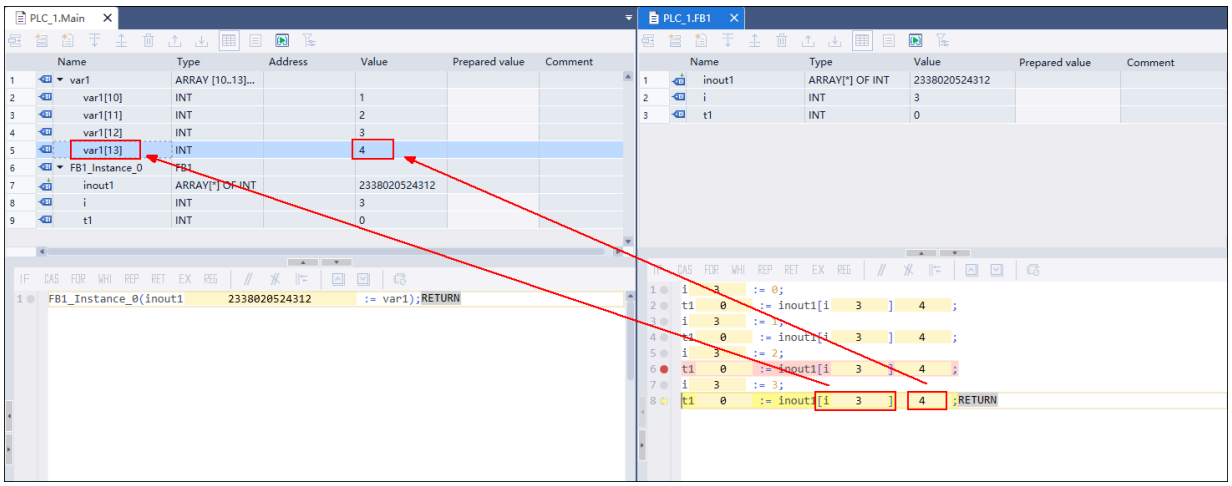
Variable-length fixed-type array usage

As shown in the following figure, define an INT-type variable-length array for the VAR_IN_OUT pin in FB, then instantiate it in "PROGRAM Main". Note that the ARRAY[*] OF INT type is used as a formal parameter in the called POU, and the caller can pass an INT array of any length as an actual parameter.



NOTE

During the call, the online value of the formal parameter expression uses a relative index, while the actual parameter at the call location displays an absolute index. The conversion is: Absolute index = Lower bound of the actual array + Relative index of the formal parameter. As shown in the figure below, the absolute index 13 is calculated as lower bound 10 + relative index 3.



10.1.1.8.3.2 POINTER

Concepts

A pointer represents an address. If a variable is defined in the software, the system allocates corresponding memory space based on the variable's data type after compilation. For example, the system allocates 1 byte of memory space for a BYTE-type variable and 4 bytes for a REAL-type variable.

Generally, programs perform storage operations through variable names. This is because after the program is compiled, variable names are converted into the memory addresses of the variables, and the storage operations on variables are carried out via memory addresses.

For example, define three WORD-type variables var1, var2, and var3 in the program. After compilation, the system allocates 2 bytes of memory to var1 at addresses 1000 and 1001, and var2 at 1002 and 1003. var3 is 1004 and 1005. The specific data stored in addresses 1000 and 1001 corresponds to the data of var1, and the same logic applies to var2 and var3, as shown below. This method of storing data by variable address is also known as "direct access" in high-level languages.

Variable	Data Content
var1	Byte_Low
	Byte_High
var2	Byte_Low
	Byte_High
var3	Byte_Low
	Byte_High

Address	Data Content
1000	Byte_Low
1001	Byte_High
1002	Byte_Low
1003	Byte_High
1004	Byte_Low
1005	Byte_High

Pointer variable

The address of a variable is called the "pointer" of that variable. For example, variables with addresses 1000 and 1001 are pointers to var1. If a variable is specifically used to store the address (pointer) of another

variable, it is called a "pointer variable". A pointer variable is declared in the software using the keyword "POINTER TO+ <type>". The <type> can refer to the memory address of variables, programs, function blocks, methods, and functions. A pointer variable can point to any of the aforementioned objects and any data type, including user-defined data types.

Pointer variable definition

The syntax for declaring a pointer is as follows:

```
<identifier>: POINTER TO <data type>;
```

identifier: The pointer identifier.

data type: The data type that the pointer points to. It can be any data type (except BIT type), function block, program, method, and function.

Pointer operations are performed using address operators. Address operators include ADR (to access variable address) and ^ (value corresponding to variable address).

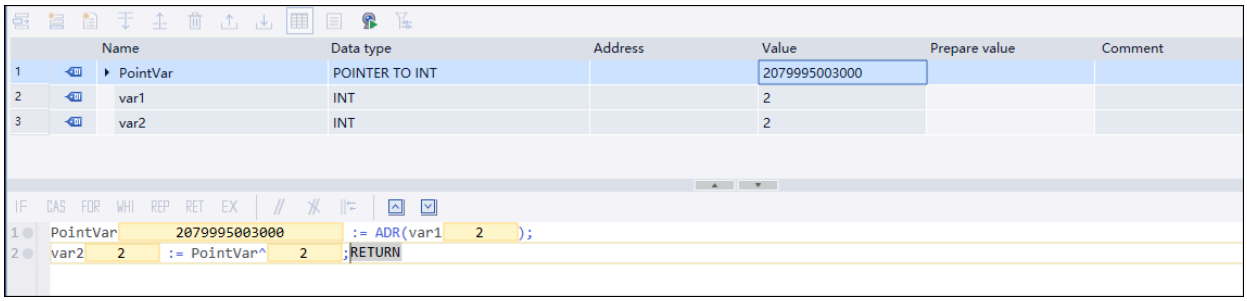
Pointer variable usage

Reading the content of a pointer address means reading the data stored in the address that the pointer currently refers to. The content can be obtained by adding the content operator "^" after the pointer identifier. The following example illustrates the usage of pointer variables.

Define PointVar as a pointer variable, which will be used to store address data.

In the program, use the ADR instruction in line 1, which is the operator used to obtain the memory address of a variable. After executing line 1, PointVar will have obtained the memory address of var1 (2079995003000 in decimal).

PointVar^ refers to the specific data (data at the address 2079995003000 in decimal) corresponding to this memory address, which is the value 2 stored in var1. After line 2 is executed, the value of PointVar^ is assigned to var2, so var2 will equal 2.



10.1.1.8.4 BIT

The BIT data type is used only for a single variable in a structure or function block, with values of TRUE (1) and FALSE (0).

The keyword "BITADR" can be used to obtain the bit address of a variable. A BIT occupies 1 bit of memory and bit access is relatively time-consuming. Therefore, the BIT data type should only be used when data needs to be defined in a predefined format.

10.1.1.8.5 ANY

ANY refers to runtime assignment of values of any data type.

When implementing function blocks, functions, or methods, it is supported to declare the input (<VAR_INPUT) as a variable of data type ANY or ANY_ <type>. ANY is a type used for variables with an undetermined type during programming.

The correspondence between general data types and basic data types is shown in the following table.

Hierarchy of General Data Types			Basic Data Types
ANY	ANY_BIT		BYTE, WORD, DWORD, LWORD
	ANY_DATE		DATE_AND_TIME, DT, DATE, TIME_OF_DAY, TOD
	ANY_NUM	ANY_REAL	REAL, LREAL
		ANY_INT	USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT
	ANY_STRING		STRING, WSTRING

ANY-type variable definition

The syntax for declaring an ANY-type variable is as follows:

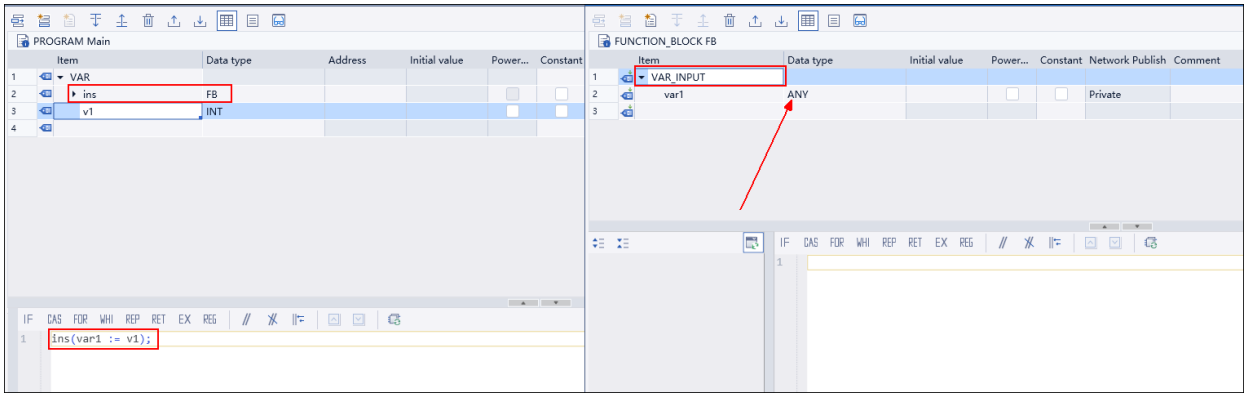
```
VAR_INPUT
   var1: ANY;
END_VAR
```

During program compilation, the compiler automatically converts the ANY type into the following structure:

```
TYPE __AnyType:
STRUCT
pValue: POINTER TO BYTE;    // A pointer to array elements
diSize: DINT;                    // Size of array elements
typeClass : __SYSTEM.TYPE_CLASS; // Type of the formal parameter (expressed by semantics)
typeId: INT; // When typeClass is of USEDEF type, typeId points to a unique random number, ensuring that each type has a distinct ID. typeId is used to determine if USERDEF types are identical.
elementTypeClass : _SYSTEM. ELEMENT _TYPE_CLASS; // When typeClass is of USERDEF type, elementTypeClass specifies the type (Struct, Union, FB)
arrayInfo: pointer of __SYSTEM.__ARRAY__DIM__INFO; // A pointer to the array dimension structure on the stack
dimCount: DINT; // Number of array dimensions
END_STRUCT
END_TYPE
```

ANY-type variable usage

As shown in the following figure, define an ANY-type variable for the VAR_INPUT pin in FB, then instantiate it in "PROGRAM Main".



Variable-length ANY-type array definition

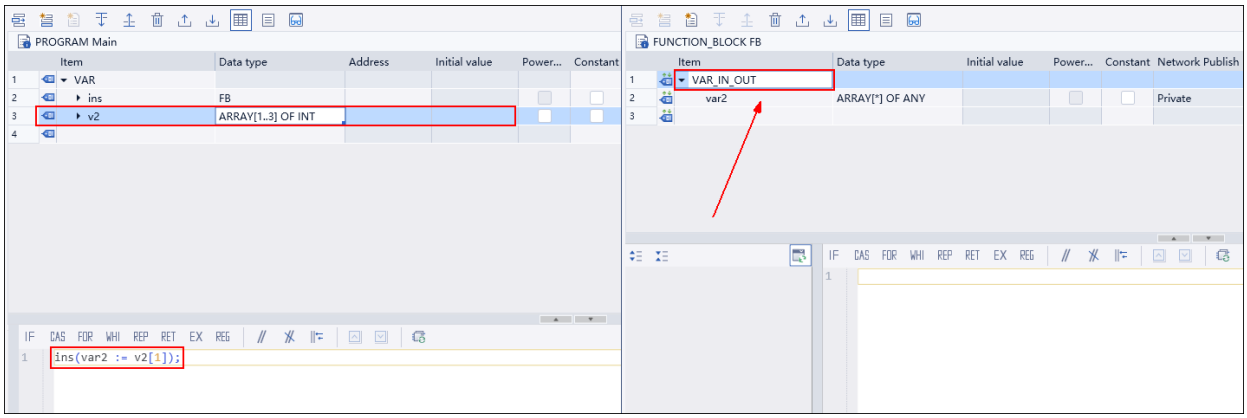
A variable-length ANY-type array is an array with variable length, whose elements are of ANY type. It can only be used for input/output variables (VAR_IN_OUT) in FB or FUN.

The syntax for declaring a variable-length ANY-type array is as follows:

```
VAR_IN_OUT
   var2: ARRAY[*] OF ANY;
END_VAR
```

Variable-length ANY-type array usage

As shown in the following figure, define a variable-length ANY-type array for the VAR_IN_OUT pin in FB, then instantiate it in "PROGRAM Main" and pass parameters (subscripts required).



ARRAY_xxx keyword usage

After defining a variable-length array for VAR_IN_OUT in FB or FUN, the ARRAY_LOWER/UPPER/INDEX keywords can be used to obtain the lower limit/upper limit/subscript of the variable-length array.

ARRAY_LOWER: Obtains the lower limit of the variable-length array.
ARRAY_UPPER: Obtains the upper limit of the variable-length array.
ARRAY_INDEX: Obtains the subscript for the current element of the variable-length array.

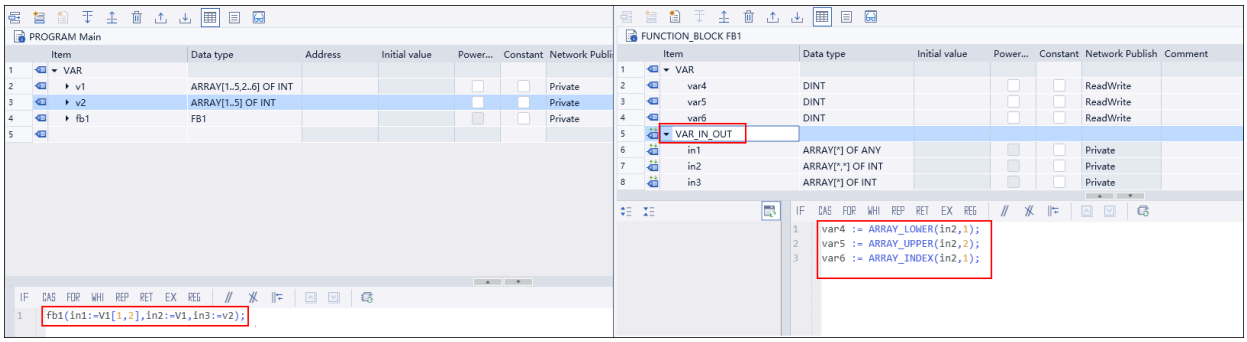
- Related variables

Input/Output	Variable	Name	Data type	Initial Value	Description
Input	In1	Variable-length array	ARRAY[*]	-	Determined by the array passed in by the formal parameter.

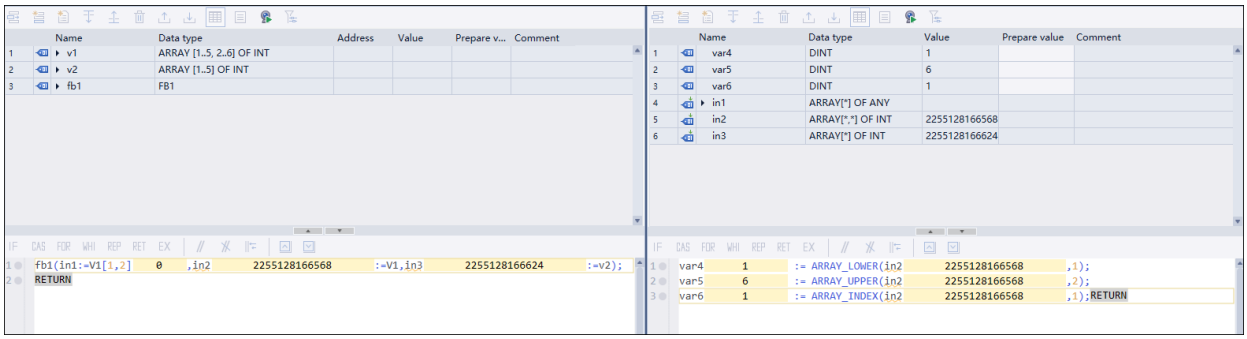
Input/Output	Variable	Name	Data type	Initial Value	Description
	In2	Dimension	BIT, BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT	-	Indicates the dimension of the array, which cannot exceed the maximum dimension limit.
Output	Out	Return value	DINT	-	Indicates the result.

- Program example

Take the ST language as an example. Define a variable-length array for the VAR_IN_OUT pin in FB and use ARRAY_LOWER/UPPER/INDEX keywords to obtain the corresponding values. Then, instantiate the array in "PROGRAM Main" and pass parameters (subscripts required).



The operation result is shown in the figure below.



10.1.1.8.6 REFERENCE

A reference is an alias of an object. This alias can be read and written by the variable name. Unlike a pointer, the data that the reference refers to is changed directly, so the assignment of the reference is identical to the data it refers to. A specific assignment is required to set the address of the reference. A dedicated operator can be used to check whether the reference refers to valid data (not equal to 0).

Reference-type variable definition

The syntax for declaring a reference is as follows:

```
<identifier> : REFERENCE TO <data type>
```

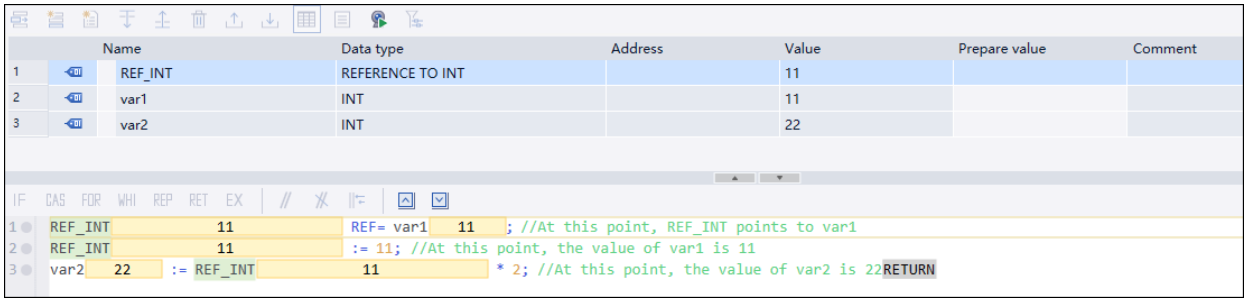
identifier: Reference identifier.

data type: The data type of the referenced object.

> **CAUTION**
>
> The BIT type cannot be referenced. Specifically, the declaration of ref1: REFERENCE TO BIT is invalid.

Reference-type variable usage

The following example illustrates how to use a reference-type variable.



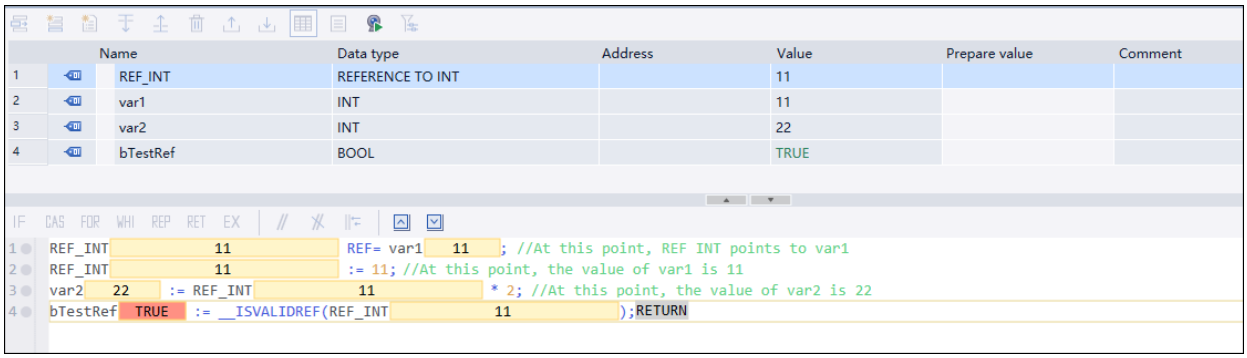
The dedicated instruction "__ISVALIDREF" can be used to check whether a variable has been properly referenced.

Its specific syntax is as follows:

```
<Boolean variable> := __ISVALIDREF(<data type> declared as reference type);
```

The return value <Boolean variable> is TRUE if the reference points to a valid value, otherwise it is FALSE.

<data type> must be declared as reference type, otherwise the instruction is invalid.



10.1.1.8.7 Data Type Conversion

Data type conversion refers to the process of converting data from one type (e.g., integer, floating-point number, string) to another. This is very common in programming and data processing to make different types of data compatible for operation, storage, or transmission.

There are two main methods of data type conversion:

- Explicit type conversion
  Explicit conversion is performed by specifying the target data type. It is usually used to convert a data type with a large range to one with a small range, or to convert between different types. The iFA software provides explicit conversion instructions for users. For details, see 18.1.12 "Conversion Operators".
- Implicit type conversion

Implicit conversion is performed automatic by the compiler. It ensures security and will not result in significant data or symbol loss during calculation. Implicit type conversion follows the following rules:

- The BOOL type and BIT type can be implicitly converted to each other, but neither can be implicitly converted to other data types.
- Only the following data types support implicit conversion: BYTE, WORD, DWORD, LWORD, INT, SINT, DINT, LINT, UINT, USINT, UDINT, ULINT, REAL, LREAL, STRING, WSTRING, CHAR, and WCHAR.
- The bit length of the source data type shall not exceed that of the target data type. For example, an operand of the WORD type cannot be assigned to a parameter of the BYTE type.

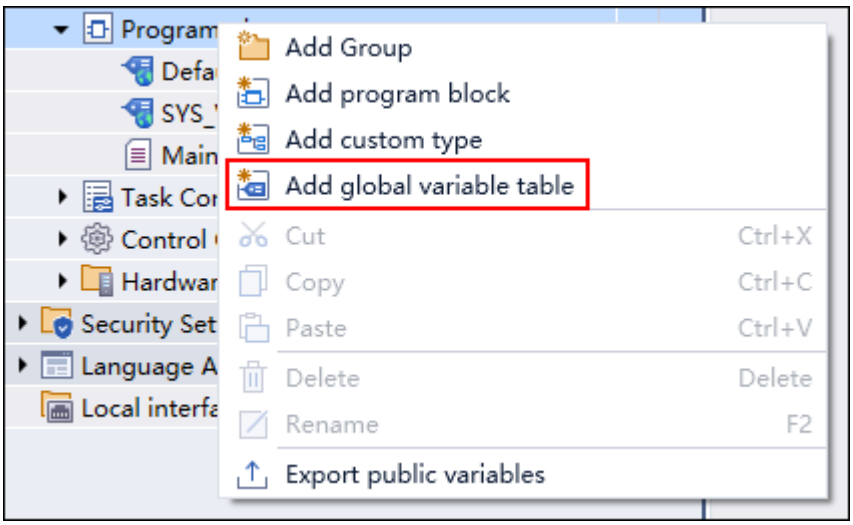
> NOTE
>
> Integer data of any length can be implicitly converted to a floating-point type, while the floating-point type cannot be implicitly converted to an integer. If the word length of the integer data is less than or equal to that of the floating-point data, the conversion will be performed normally without warnings; otherwise, a warning will be reported.

10.1.1.9 Variable Declaration

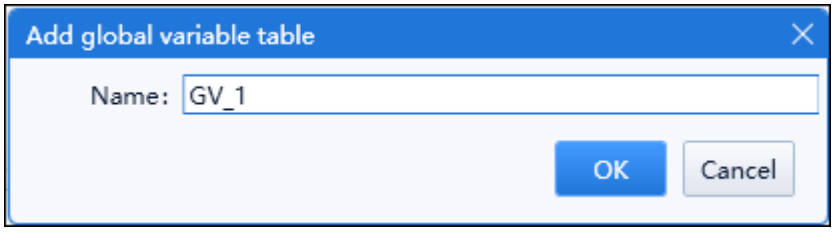
10.1.1.9.1 Adding a Global Variable Table

Procedure

1. Right-click "Programming" and select "Add global variable table".



2. Set the name of the global variable table and click "OK" .



3. The added global variable table is displayed in the project tree under "Programming".

NOTE

The name of the global variable table shall conform to IEC standards and must not exceed 128 in length.

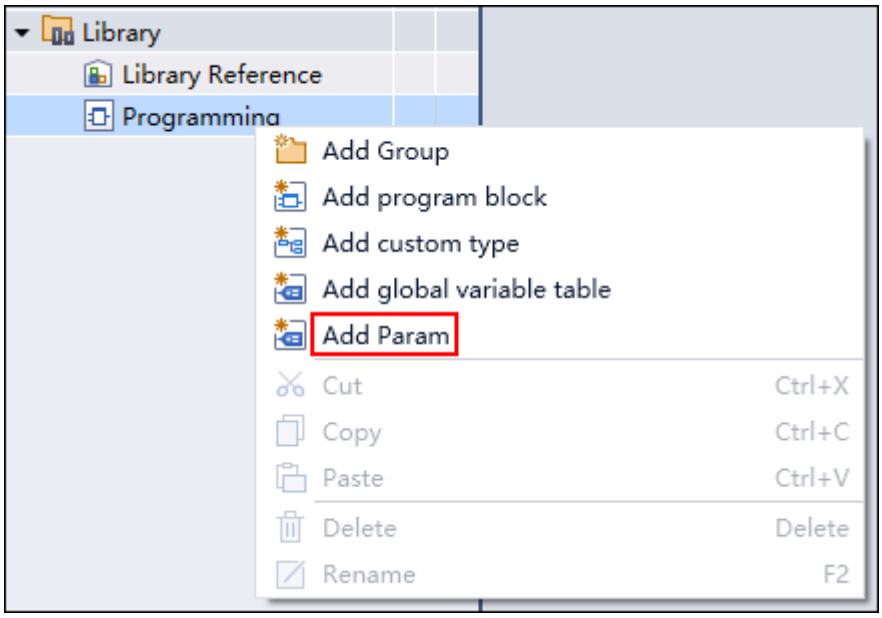
10.1.1.9.2 Adding a Param Table

Prerequisites

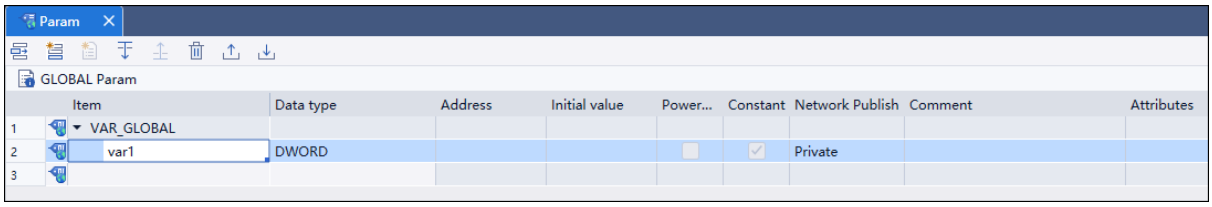
A library project has been created or opened.

Procedure

1. Right-click "Programming" and select "Add Param".



2. Add a constant to the "Param" table.



NOTE

The "Param" table can only be used in library projects and only one table can be added.

10.1.1.9.3 Variable Declaration Properties

A variable has nine properties: ID (name), data type, address, initial value, power-failure retention, constant, network public property, comment, and property.

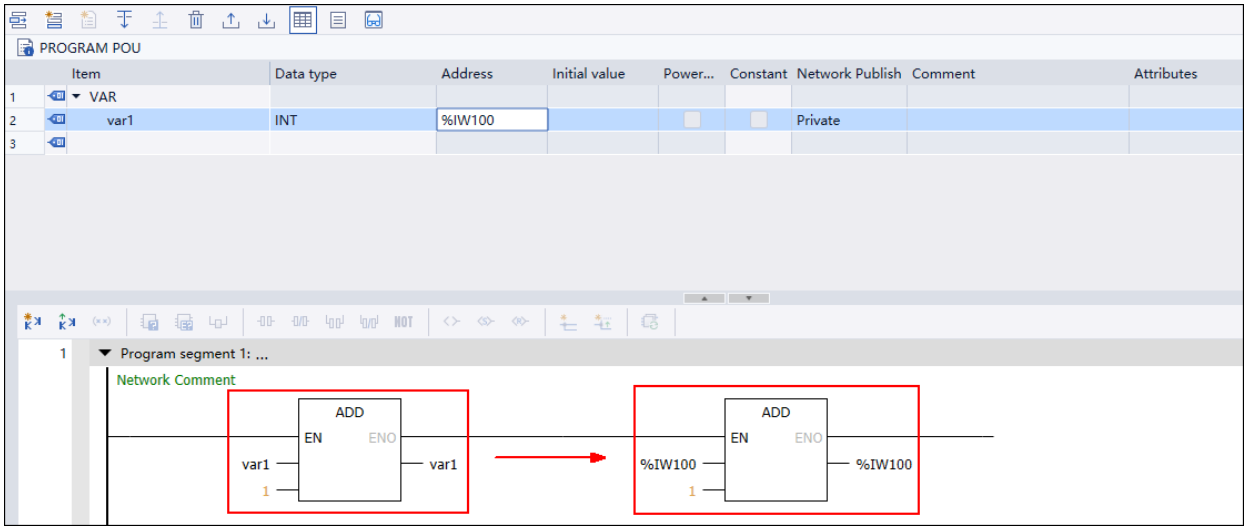
- **ID**
  ID is the variable's name. When naming variables, keep the following in mind:
  - It cannot include spaces or special characters.
  - It cannot be a predefined keyword.
  - Names are not case-sensitive.
  - The maximum length is 128 characters.
  - Names cannot be defined repeatedly.
- **Data type**
  For the supported data types in the iFA platform, see 10.1.1.8 Data Types.
- **Address**
  Address can be used to define the memory location where the variable is stored, known as the direct address (AT). The syntax for defining a direct address is as follows:

```
Plain Text
%<memory area prefix><size prefix><number>|.<number>
```

  The programming system supports the following three storage area prefixes:
  - I: Input, physical input, "sensor".
  - Q: Output, physical output, "activator".
  - M: Storage position.

  The programming system supports the following size prefixes:
  - X: Bit, one bit.
  - B: Byte, one byte.
  - W: Word, two bytes.
  - D: Double Word, four bytes.

  Program editing supports using addresses to manipulate objects.



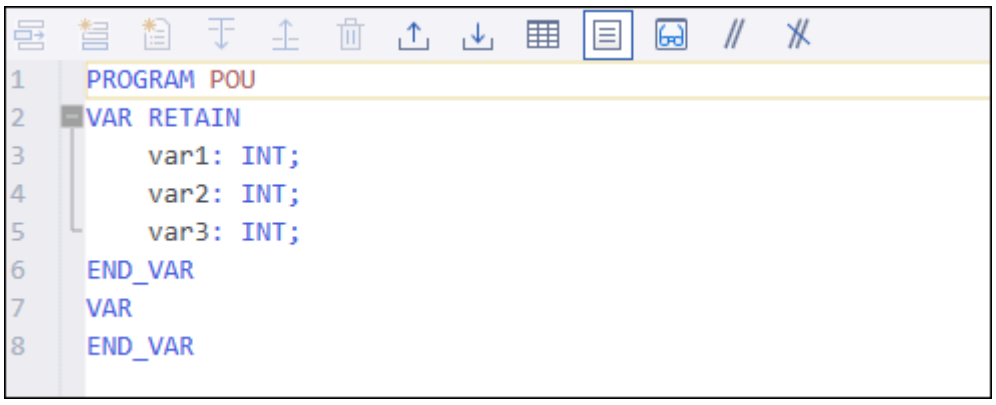
- **Initial value**

  The initial value of a variable defaults to 0 but can be customized during variable declaration.

- **Power failure retention**

  Power failure retention ensures that variables retain their original values after PLC power failures or program downloads. This is commonly used to define critical parameters in projects, preventing the data loss due to unexpected PLC power failures or program downloads.

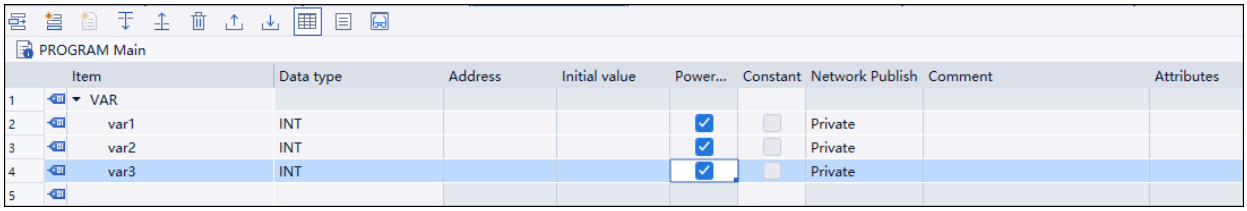
  The power failure retention characteristic is primarily declared using the property key RETAIN.

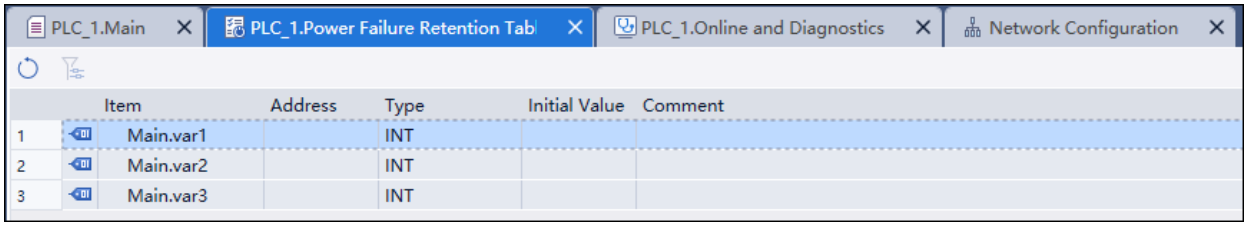


Action	VAR	VAR RETAIN
Power failure	Initialization	Retain the original value.
Hot Reset	Initialization	Retain the original value.
Cold Reset	Initialization	Retain the original value.
Program download	Initialization	Retain the original value.
Online modification	Retain the original value.	Retain the original value.

Power failure retention table

Check the "Power failure retention" options for variables in "PROGRAM Main". After compilation, these variables will be displayed collectively in the power failure retention table.



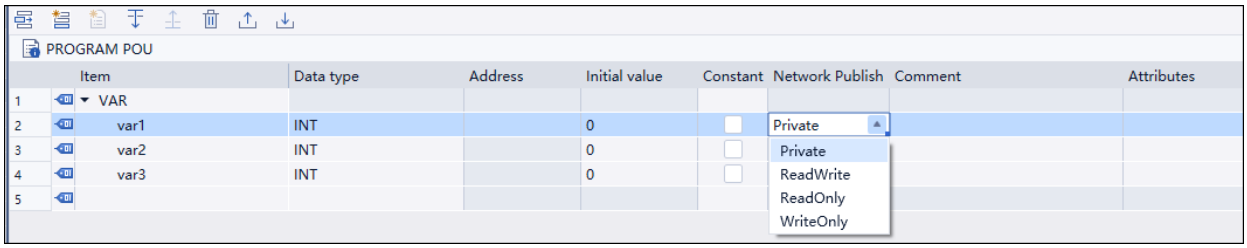


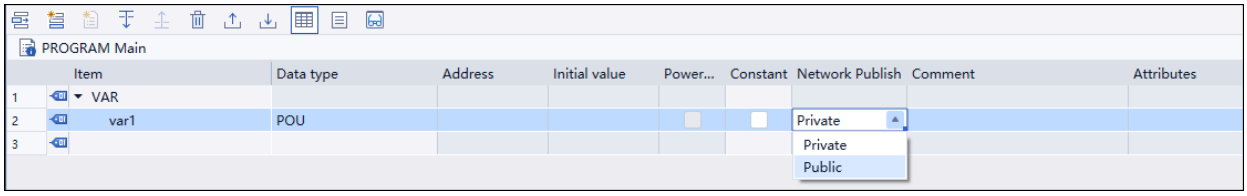
- **Constants**

  Constants are numerical values that do not change. Constants do not support the power-failure retention property.

- **Network public property**

  The network public property can be used to display the network public status of a variable.





  - When the "Private" option is selected, the variable does not support access by third-party equipment or software.
  - When the "Read/Write", "Read-Only", or "Write-Only" option is selected, the variable supports access by third-party equipment or software.
    - Read/Write: The data of the variable can be modified or viewed.
    - WriteOnly: The data of the variable can only be modified, but cannot be viewed or read.
    - Read-Only: The data of the variable can only be viewed or accessed, but cannot be modified.
  - When the "Public" option is selected, the public property of the child node is applied.

    Network public property restrictions:

  - Restriction rules
    - The number of variables is determined by the number of nodes in the variable or variable group. For complex data types, the number of variables is determined by the number of child elements. For example, the array ARRAY[1...5000] OF INT represents 5000 public variables.
    - Only variables within the compilation call chain are counted, encompassing all global variables and those in referenced PRGs (including various instantiated complex types). When calculating the number of child elements in complex types, elements containing pointers, references, ANY-type variables, input/output variables, temporary variables, constants, and variables within methods are excluded. All private child elements are excluded from the public variable calculation results.
  - Restriction limit

    The upper limits of public variables differs for various products, as follows.

Product Series	Number of network public variables
EVO400/EVO500 series	≤ 50,000
EVO700	≤ 200,000

Product Series	Number of network public variables
EVO800 series	≤ 500,000

◦ Principles

After the upper limit is exceeded, the compilation reports a fault.

Different network public properties:

Type	Public Attribute
Standard type instances	Read-only / Write-only / Read-write / Private
Structure instances	Public / Private
Union instances	Read-only / Write-only / Read-write / Private
Enumeration instances	Read-only / Write-only / Read-write / Private
FB instances	Public / Private
Alias instances	Depends on the original type (following the rules in this table)
Structure / FB array instances	Public / Private
Other array instances	Read-only / Write-only / Read-write / Private

- **Comments**

  Comments are used to add annotations to variables, making them easier to understand.

- **Property**

  Properties are used for differentiated handling of individual variables. For example, when a user needs to permanently display a certain variable in decimal format, this can be achieved by setting the variable's attribute 'displaymode' := 'dec'. The following are commonly used properties:

No.	Keyword	Value	Note
1	displaymode	'bin', 'dec', 'hex'	Specifies the base for displaying the online value of a variable, supporting binary, decimal, and hexadecimal formats.
2	visibility	none	Visibility check for variables/notations; if it includes "none," it is hidden.
3	qualified_only	/	When this marking is added to a global variable, it must be accessed using the global variable group name followed by a dot.
4	strict	/	Specifies a strict enumeration type. After setting this property, the value of an enumeration variable must be a valid enumeration value, otherwise a compilation fault will be reported.
5	to_string	/	Converts an INT-type enumeration value to the corresponding character string using the tostring function.

10.1.1.9.4 Variable Address Range

The following table shows the address ranges of PLC variables. The variable address ranges differ among different products, as shown below.

Storage Area	Address Range				
	EVO400 (Byte)	EVO500 (Byte)	EVO700 (Byte)	EVO800 (Byte)	APC500 (Byte)
I area	0 to 131071	0 to 131071	0 to 131071	0 to 1048575	0 to 131071
Q area	0 to 131071	0 to 131071	0 to 131071	0 to 1048575	0 to 131071
M area	0 to 524287	0 to 524287	0 to 5242879	0 to 2097151	0 to 524287

10.1.1.9.5 Default Variable Table

When adding a PLC device to the project, a "Default variable table (Global)" node is automatically created under the device. Double-click this node to open the corresponding window.

The default variable table does not support copying or deletion, and the node name cannot be modified.

The default variable table supports import and export operations. For details, see sections 10.1.1.10.6 Variable Input and 10.1.1.10.7 Variable Output.

When creating new variables in I/O mapping, they are automatically added to this table.

10.1.1.9.6 System Variable Table

When adding a PLC device to the project, a "SYS_VAR(Global)" node is automatically created under the device. Double-click this node to open the corresponding window.

The system variable table is the variable table created by default, which does not support copying or deletion, and the node name cannot be modified.

The system variable table supports export. For details, see section 10.1.1.10.7 Variable Export. Variable import is not supported.

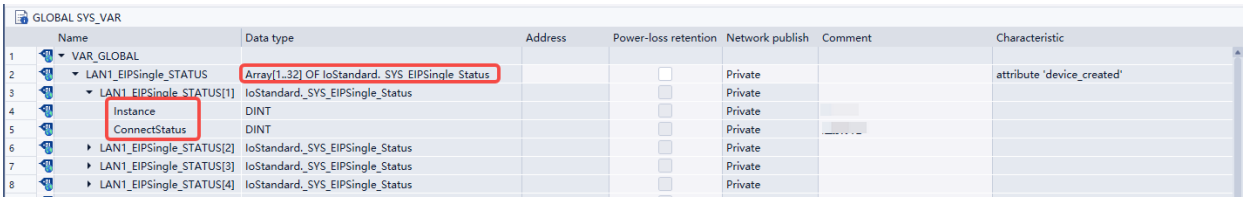
System variables include the following parts:

- **Disabled variables**
  Disabled variables are related to PLC hardware configuration. Each time a new device is added, corresponding variables are automatically created in the system variable table, with the naming rule: J_device name. Variables can only be added or deleted by adding or deleting devices, and the names and data types of variables cannot be modified. Users can modify the address, power failure retention property, and network public property of variables according to requirements. For details, see 10.1.1.9.3 Variable Declaration Property. The structure variable bDisableEnable can be operated to disable or enable the device.

GLOBAL SYS_VAR

	Name	Data type	Address	Power-loss retention	Network publish	Comment	Characteristic
1	VAR_GLOBAL						
2	LAN1_EIPSingle_STATUS	Array[1..32] OF IoStand...			Private		attribute 'device_created'
3	LAN1_EIPSlave_STATUS	Array[1..16] OF RemsA...			Private		attribute 'device_created'
4	J_GL20_4AD	IoStandard._SYS_SlaveS...		☑	Private		attribute 'device_created'
5	bDisableEnable	BOOL			Private		
6	J_GL20_8ADV	IoStandard._SYS_SlaveS...		☑	Private		attribute 'device_created'
7	bDisableEnable	BOOL			Private		
8	J_GL20_8ADI	IoStandard._SYS_SlaveS...		☑	Private		attribute 'device_created'
9	bDisableEnable	BOOL			Private		

- **Device diagnostic status variables**
  Device diagnostic status variables are related to EtherNet/IP device configuration. When adding a PLC device, corresponding variables are automatically created in the system variable table. Variable name: Device network port number_EIPSingle_STATUS. Variables are automatically generated and do not support addition or deletion. The names and data types of variables cannot be modified. Users can modify the address, power failure retention property, and network public property of variables according to requirements. For details, see 10.1.1.9.3 Variable Declaration Property. These variables are used to display connection information of the master. The specific structure details are shown in the table below.

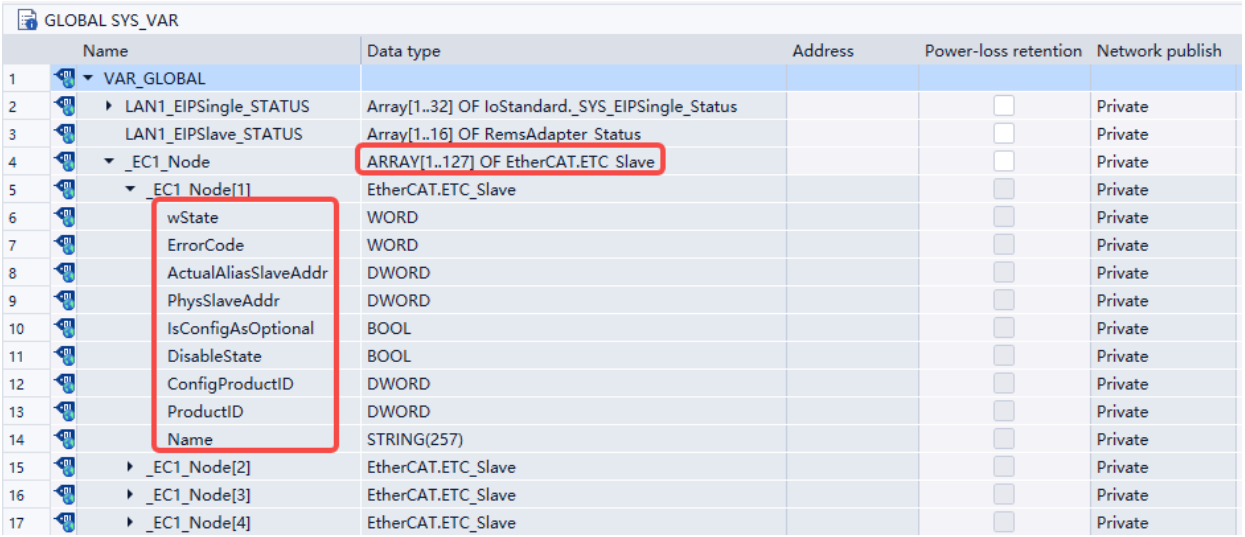


Variable	Definition	Function description
Instance	Instance number	-
ConnectStatus	Connection Status	0: Connection closed 1: Connection is abnormal 3: Connection is abnormal

NOTE

The number of Ethernet ports and the number of connections supported by each port vary by product. For details, see the relevant product manuals.

- **EtherCAT slave system variables**
  EtherCAT slave system variables are related to EtherCAT device configuration and are automatically created in the system variable table when adding an EtherCAT device, with the variable name: _EC1_Node. Variables are automatically generated and do not support addition or deletion. The names and data types of variables cannot be modified. Users can modify the address, power failure retention property, and network public property of variables according to requirements. For details, see 10.1.1.9.3 Variable Declaration Property. These variables are used to display EtherCAT slave information. Take the EVO500 series PLC V1.5 version as an example, the specific structure details are shown in the table below.


GLOBAL SYS_VAR

	Name	Data type	Address	Power-loss retention	Network publish
1	VAR_GLOBAL				
2	LAN1_EIPSingle_STATUS	Array[1..32] OF IoStandard._SYS_EIPSingle_Status			Private
3	LAN1_EIPSlave_STATUS	Array[1..16] OF RemsAdapter_Status			Private
4	_EC1_Node	ARRAY[1..127] OF EtherCAT.ETC_Slave			Private
5	_EC1_Node[1]	EtherCAT.ETC_Slave			Private
6	wState	WORD			Private
7	ErrorCode	WORD			Private
8	ActualAliasSlaveAddr	DWORD			Private
9	PhysSlaveAddr	DWORD			Private
10	IsConfigAsOptional	BOOL			Private
11	DisableState	BOOL			Private
12	ConfigProductID	DWORD			Private
13	ProductID	DWORD			Private
14	Name	STRING(257)			Private
15	_EC1_Node[2]	EtherCAT.ETC_Slave			Private
16	_EC1_Node[3]	EtherCAT.ETC_Slave			Private
17	_EC1_Node[4]	EtherCAT.ETC_Slave			Private

Variable	Definition	Function description
wState	Slave connection status	1: Initialization 2: Pre-operational 3: Boot-operational 4: Safe-operational 8: Operational 17: Initialization fault 18: Pre-operation fault 19: Boot fault 20: Safe-operational fault 24: Operational fault
ErrorCode	Slave fault code	0x0001: Slave device offline 0x0002: Slave device vendor ID mismatch 0x0004: Slave device product code mismatch 0x0008: Slave device version number mismatch 0x0010: Slave device fault flag 0x0020: Slave device reverse connection 0x0040: Slave device port closed 0x0080: Slave device status fault 0x0100: Slave device SDO configuration failed 0x0200: Slave device status transition timeout 0x0400: Slave device SDO download timeout 0x0800: Slave device EEPROM fault Note: When multiple faults exist simultaneously, the slave fault codes will be displayed in an accumulated manner.
ActualAliasSlaveAddr	Slave alias address	-

Variable	Definition	Function description
PhysSlaveAddr	Slave physical address	Slave address 1001
IsConfigAsOptional	Whether the slave is configured with an alias	TRUE: The slave is configured with an alias. FALSE: The slave is not configured with an alias.
DisableState	Whether the slave is disabled	TRUE: The slave is disabled. FALSE: The slave is enabled.
ConfigProductID	Configured product ID	-
ProductID	Actual slave product ID	-
Name	Slave name	-

NOTE

The EVO500 series PLC V1.2 version does not support automatically creating EtherCAT slave system variables when adding EtherCAT devices.

- **Remote adapter status variables**
  Remote adapter status variables are related to EtherNet/IP device configuration. When adding an EtherNet/IP device, corresponding variables are automatically created in the system variable table. Variable name: Device network port number_EIPSlave_STATUS. Variables are automatically generated and do not support addition or deletion. The names and data types of variables cannot be modified. Users can modify the address, power failure retention property, and network public property of variables according to requirements. For details, see 10.1.1.9.3 Variable Declaration Property. These variables are used to display the operation status of the remote adapter. The specific structure details are shown in the table below.

GLOBAL SYS_VAR

	Name	Data type	Address	Power f...	Network ...	Comment	Property
1	VAR_GLOBAL						
2	LAN1_EIPSingle_STATUS	Array[1..32] OF IoStandard._SYS_EIPSingle_Status			不公开		attribute 'device_created'
3	LAN1_EIPSlave_STATUS	Array[1..16] OF EtherNetIP.RemAdapter_Status			不公开		attribute 'device_created'
4	LAN1_EIPSlave_STATUS[1]	EtherNetIP.RemAdapter_Status			不公开		
5	DevName	STRING			不公开		
6	wState	USINT			不公开		
7	ErrorCode	DWORD			不公开		
8	CfgIP	STRING			不公开		
9	ActualIP	STRING			不公开		
10	DisableStatus	BOOL			不公开		
11	DevAliasID	WORD			不公开		
12	LAN1_EIPSlave_STATUS[2]	EtherNetIP.RemAdapter_Status			不公开		
13	LAN1_EIPSlave_STATUS[3]	EtherNetIP.RemAdapter_Status			不公开		
14	LAN1_EIPSlave_STATUS[4]	EtherNetIP.RemAdapter_Status			不公开		
15	LAN1_EIPSlave_STATUS[5]	EtherNetIP.RemAdapter_Status			不公开		

Variable	Definition	Function description
DevName	Remote adapter name	-
wState	Remote adapter connection status	0: Connection closed 1: Remote adapter is disabled 2: Connection is abnormal 3: Connection is abnormal

Variable	Definition	Function description
ErrorCode	Remote adapter fault code	Refer to 21.1 Appendix 1: Fault Code
CfgIP	Remote adapter configured IP	-
ActualIP	Remote adapter actual IP	-
DisableStatus	Remote adapter disabled status	TRUE: The slave is disabled. FALSE: The slave is enabled.
DevAliasID	Remote adapter alias ID	-

NOTE

- The number of Ethernet ports and the number of connections supported by each port vary by product. For details, see the relevant product manuals.
- EVO series PLCs with version below V1.5 do not support automatically creating EtherNet/IP remote adapter status system variables when EtherNet/IP devices are added.

- **Axis array variables**

  Axis array variables are related to axis device configuration. When an axis device is added, the corresponding variables are automatically created in the system variable table. Variable name: _MC_Axis. Variables are automatically generated and do not support addition or deletion. The names and data types of variables cannot be modified. Users can modify the address, power failure retention property, and network public property of variables according to requirements. For details, see 10.1.1.9.3 Variable Declaration Property. These variables are used to manage all real axes and virtual axes under all EtherCAT masters in a centralized way, and store system variable information for these axes. For detailed structure description, refer to 18.3.3.2.2.1 AXIS_REF: Axis Structure.

	Name	Data type	Address	Power f...	Network ...	Comment	Property
1	VAR_GLOBAL						
2	LAN1_EIPSingle_STATUS	Array[1..32] OF IoStandard._SYS_EIPSingle_Status			Private		attribute 'device_created'
3	LAN1_EIPSlave_STATUS	Array[1..16] OF RemsAdapter_Status			Private		attribute 'device_created'
4	MC_Axis	ARRAY[1..64] OF MotionControl.AXIS_REF			Private		attribute 'device_created'
5	_MC_Axis[1]	MotionControl.AXIS_REF			Private		
6	_MC_Axis[2]	MotionControl.AXIS_REF			Private		
7	_MC_Axis[3]	MotionControl.AXIS_REF			Private		
8	_MC_Axis[4]	MotionControl.AXIS_REF			Private		
9	_MC_Axis[5]	MotionControl.AXIS_REF			Private		
10	_MC_Axis[6]	MotionControl.AXIS_REF			Private		

Axis array variables	Description
Array name	_MC_Axis
Array length	Total number of EtherCAT axes supported by this PLC
Array mapping	Axis array subscript represents the axis number. For example, the array element corresponding to "Axis" is _MC_Axis[1].
Variables accessible in the array	Refer to 18.3.3.2.2.1 AXIS_REF: Axis Structure.

Axis array variables	Description
Axis array subscript corresponding to the axis	The subscript of the axis in the axis array is displayed under the project tree "PLC_X node > Control object > Axis". The subscript can be manually adjusted in the 16.1.3 Axis basic settings interface. Note: Axis numbers must not be duplicated and must not exceed the maximum number of axes supported by the PLC.

NOTE

EVO series PLCs with version below V1.5 do not support automatically creating axis array variables when axis devices are added.

- **EtherCAT master system variables**
  EtherCAT master system variables are related to EtherCAT device configuration and are automatically created in the system variable table when adding an EtherCAT device, with the variable name: _EC_Master. Variables are automatically generated and do not support addition or deletion. The names and data types of variables cannot be modified. Users can modify the address, power failure retention property, and network public property of variables according to requirements. For details, see 10.1.1.9.3 Variable Declaration Property. EtherCAT master system variables are used to display information about the EtherCAT master. The specific structure is described in the table below.

GLOBAL SYS_VAR

	Name	Data type	Address	Power-loss retention	Network publish
1	VAR_GLOBAL				
2	LAN1_EIPSingle_STATUS	Array[1..32] OF IoStandard._SYS_EIPSingle_Status			Private
3	LAN1_EIPSlave_STATUS	Array[1..16] OF RemsAdapter_Status			Private
4	_EC1_Node	ARRAY[1..127] OF EtherCAT.ETC_Slave			Private
5	_EC_Master	ARRAY[1..1] OF EtherCAT.ETC_Master			Private
6	_EC_Master[1]	EtherCAT.ETC_Master			Private
7	bSyncWindowOver	BOOL			Private
8	ErrorCode	WORD			Private
9	DCClockReferenceTime	ULINT			Private
10	Instance	UINT			Private
11	SyncFinished	BOOL			Private
12	LostFrameCount	UDINT			Private

Variable	Definition	Function description
bSyncWindowOver	Synchronization window deviation	The default value is FALSE. bSyncWindowOver is set to TRUE when the "Sync window" value (μs) is greater than 0 and is no less than the current bus's "Max. sync window deviation".
ErrorCode	Master fault code	0x0001: Configured slave count < scanned slave count 0x0002: Slave alias conflict 0x0004: Slave fault detected 0x0008: Continuous frame loss 0x0010: Slave CRC fault 0x0020: Main network card not connected 0x0040: Redundant network card not connected 0x0080: Slave alias is 0 0x0100: Network card not connected

Variable	Definition	Function description
DCClockReferenceTime	DC reference clock	DC reference clock real-time value (read-only).
Instance	Master instance address	-
SyncFinished	Master synchronization status	SyncFinished is set to TRUE after master startup completes, regardless of whether slaves are in OP (Operational) status.
LostFrameCount	Lost frame count	Consistent with the "Lost frames" displayed on the "Online" configuration page of the EtherCAT master.

NOTE

The EVO series PLC V1.6 version and below does not support automatically creating EtherCAT master system variables when adding EtherCAT devices.

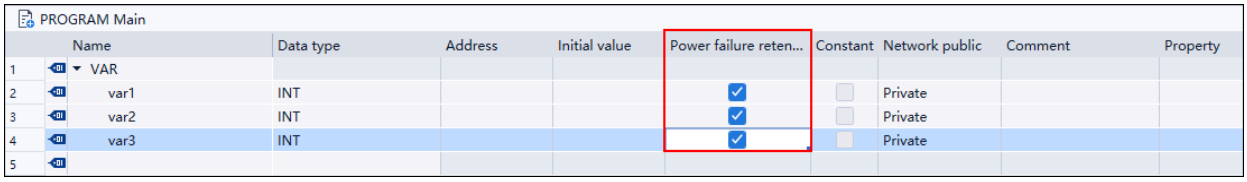
10.1.1.9.7 Power Failure Retention Table

Power failure retention ensures that variables retain their original values after PLC power failures or program downloads. This is commonly used to define critical parameters in projects, preventing the data loss due to unexpected PLC power failures or program downloads.

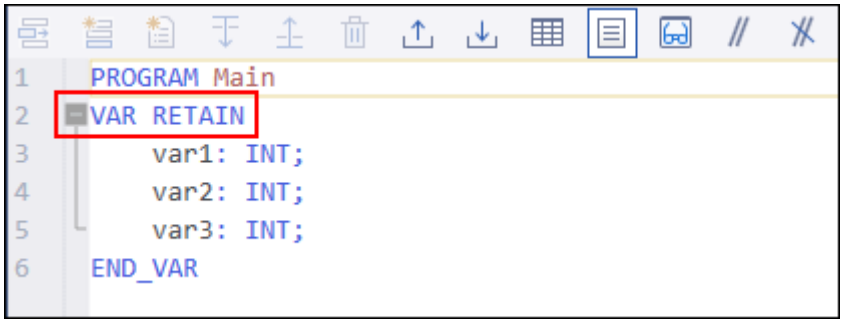
Variable definition

Power failure retention variables can be defined in the global variable tables, programs, function blocks, and functions (limited to static variables). They cannot be defined in methods, properties, structures, unions, enumerations, or aliases. The power failure retention property is mainly declared using the keyword RETAIN and can be defined through a table view and a text view. Take definition in a program (PRG) as an example.

- In the table view of the variable declaration editor, check "Power failure retention" options to define the corresponding variables as power failure retention variables, as shown in the figure below.



- In the text view of the variable declaration editor, add the "RETAIN" keyword to the variable to define it as a power failure retention variable, as shown in the figure below.



Variable category

Power failure retention variables can be set as local variables, input variables, output variables, static variables, and so on. They cannot be set as input/output variables, temporary variables, constants, etc.

Variable types

- The power failure retention property of variables in an FB can only be configured during the FB's declaration phase, but cannot be configured after the FB is instantiated. If an FB member variable is of FB type, this variable cannot be selected for power failure retention.
- The power failure retention property of variables in structures, unions, and aliases can only be configured through the corresponding instances, but cannot be configured during the declaration phase.
- For sub-members or nested sub-members of a type that contain variables of the FB type, the power failure retention is not supported.
- The power failure retention is not supported for variables of pointer and reference type.
- The power failure retention is not supported for ANY-type variables and ANY arrays, which contain pointers.
- "Constant" and "Power failure retention" can not be checked simultaneously.

Variable address

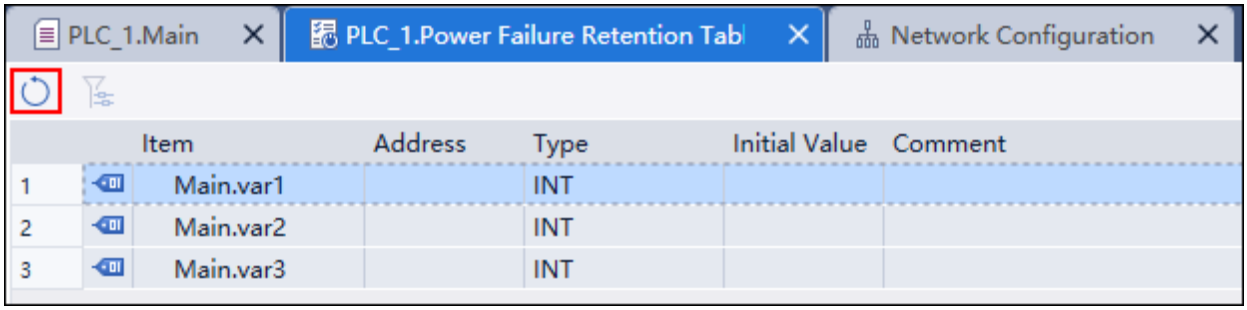
Only variables that are unbound or bound to M area addresses can be marked as power failure retention variables.

Variable response action

Action	VAR	VAR RETAIN
Power failure	Initialization	Retain the original value.
Hot Reset	Initialization	Retain the original value.
Cold Reset	Initialization	Retain the original value.
Program download	Initialization	Retain the original value.
Online modification	Retain the original value.	Retain the original value.

Power Failure Retention Table

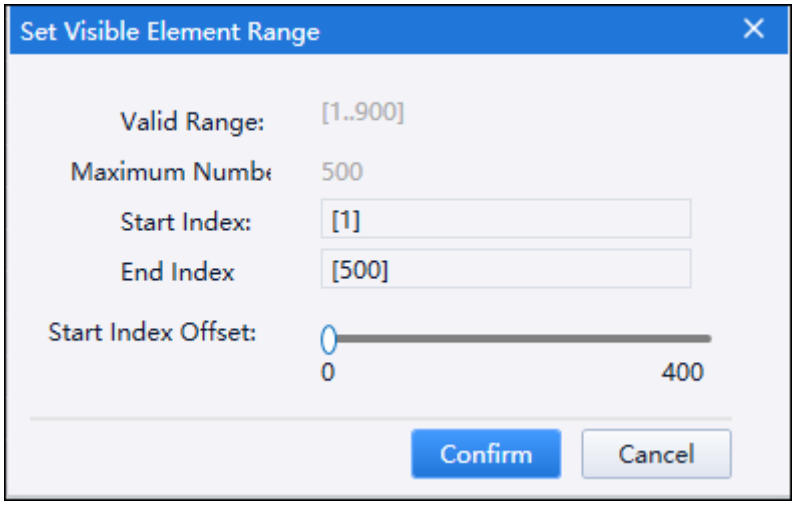
- All variables with power failure retention enabled in the project can be displayed in the power failure retention table after compiling and refreshing with the " " button, as shown in the following figure.



NOTE

Right-click items in the power failure retention table. For offline mode, "Go to definition" and "Cross-reference" are supported; for online mode, "Go to definition", "Cross-reference", and base switching (binary, decimal, and hexadecimal) are supported.

- The maximum number of variables displayed in the power failure retention table is 500. When this number is exceeded, the " " button will be enabled to set the visible element range. Users can set the variable display range through this button, as shown in the following figure.



NOTE

Since iFA Evolution V1.5.2.5, the viewing function of the power failure retention table has changed:

- For versions prior to V1.5.2.5: The maximum number of viewable variables is limited to 500.
- For versions V1.5.2.5 and later: Users can view all variables with no limit.

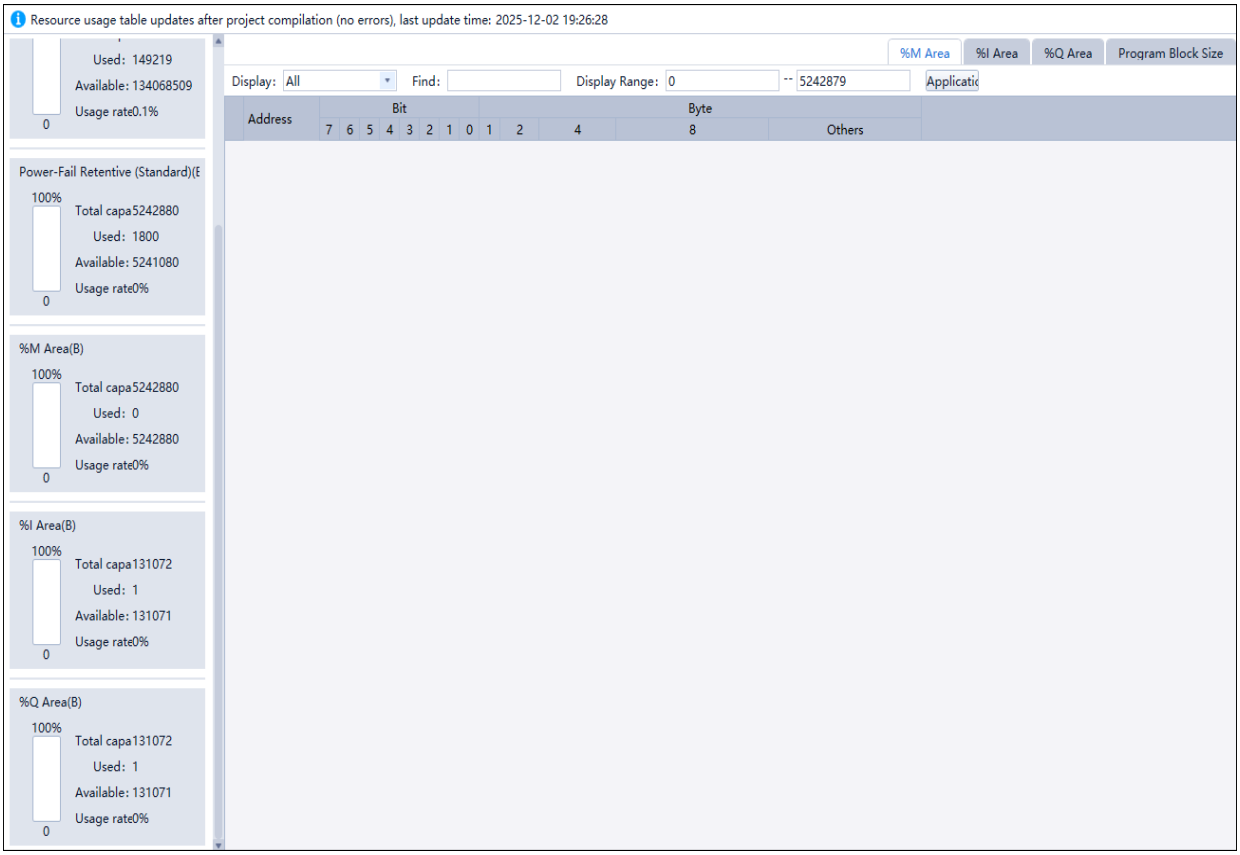
Power failure retention property

1. Power failure retention variables are uniquely identified by their full path variable names. Variables with the same name under different POUs are considered different variables, such as Main.Var1 and PRG_1.Var1. Variables with the same path in different projects are considered the same variable. When the variable path changes, the power failure retention variable is reinitialized.
2. When the binding address, initial value, comment, declaration category, or property of a variable change, the power loss retention variable retains its original value.
3. For non-FB types: When the declaration name changes, all instance variables of this type are reinitialized.
4. For FB types: When the declaration name changes, the direct instance variables of the FB are retained, while the FB instance variables in other higher-level structures that include this FB instance are reinitialized.

5. For complex types, when their sub-members change:
    - Structures: Members that do not change retain their original values, while members that change are initialized.
    - Function blocks: Members that do not change retain their original values, while members that change are initialized.
    - Unions: Unions are initialized when their total size changes.
    - Enumeration: Enumerations are initialized when their total size changes.
6. For array types:
    - When the element type changes, all elements of the array will be reinitialized.
    - When the number of array dimensions changes, all elements of the array will be reinitialized.
    - When the number of array dimensions does not change, but the upper or lower bounds of some dimensions change, the elements in the overlapping positions retain their original values, while the other elements are initialized.
7. Performance constraints
    - The memory size of power failure retention variables is determined by the configuration files of the device. If it exceeds the limit, a compilation fault will be reported and the generation fails.
    - The memory size of power failure backup code is determined by the configuration files of the device. If it exceeds the limit, all power failure retention variables will be initialized.

10.1.1.9.8 Resource Usage Table

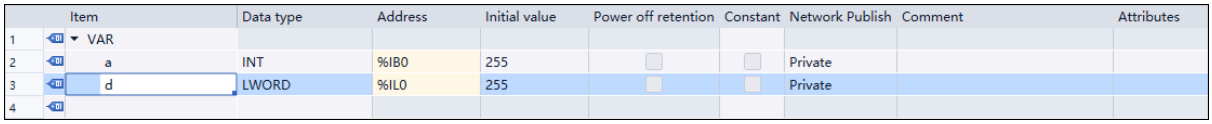
The resource usage table is mainly used to display the status of the program, including the overall resource usage of the program, data, power failure retention, M area, I area, Q area, and program block size, as well as specific displays of the M area, I area, Q area, and program block size.



1. The resource usage table is not generated in real-time but is updated after the project is compiled (without faults). The updated results after compilation need to be saved so that they can be viewed the next time the project is opened. Variables that have not been assigned absolute addresses will not be counted or displayed in the resource usage table. The resource usage table only shows the usage of defined variables in the program. Undefined and unused variables are not displayed in the address conflict table on the right side, but only in the statistical information on the left side. Variables declared with address information AT are also included in the display and statistics.

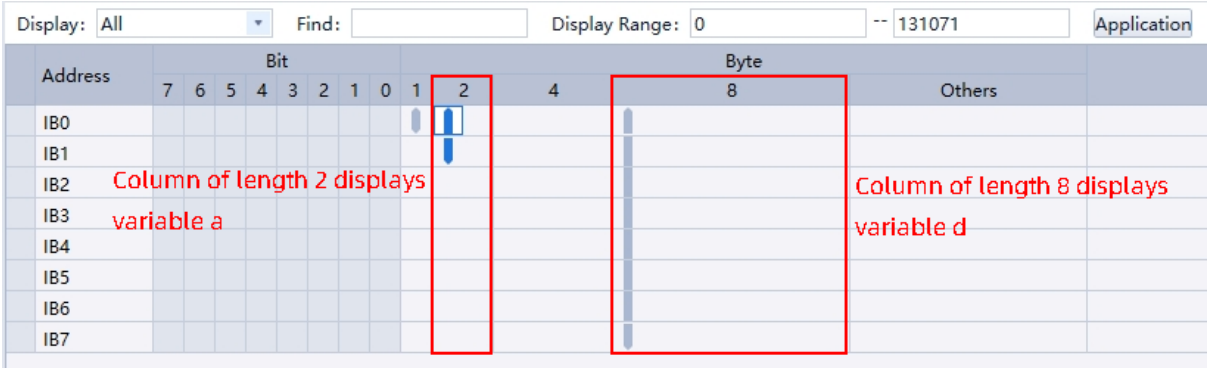
2. The overall resource usage is displayed using six cards on the left side, including program, data, power failure retention, M area, I area, and Q area. The "Display" filter in the tabs of the M area, I area, and Q area can be divided into four categories:
   - All: Lists all defined (configured, used, or coupled) variables.
   - Used: Lists all used variables and addresses (including direct addresses).
   - Idle: Lists all defined but unused variables.
   - Conflict: Lists all variables detected to have address conflicts or overlaps.

3. The width corresponding to the byte length in theh table indicates scenarios where there is byte address overlap among variables. For example, an 8-byte variable of LWORD type can be overlapped with other variables in eight possible ways, so the width is 8. For complex data types such as arrays or structures, the display is based on the unit length of the variable type of their smallest sub-member.
   - As shown in the figure below, two overlapping variables in the I area are defined in the main function. Variable a is of INT type, with a length of two bytes; variable d is of LWORD type, with a length of 8 bytes.

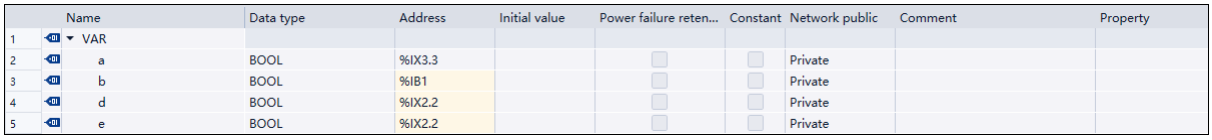


	Item	Data type	Address	Initial value	Power off retention	Constant	Network Publish	Comment	Attributes
1	VAR								
2	a	INT	%IB0	255			Private		
3	d	LWORD	%IL0	255			Private		
4									

   - View the usage of the I area in the resource usage table.

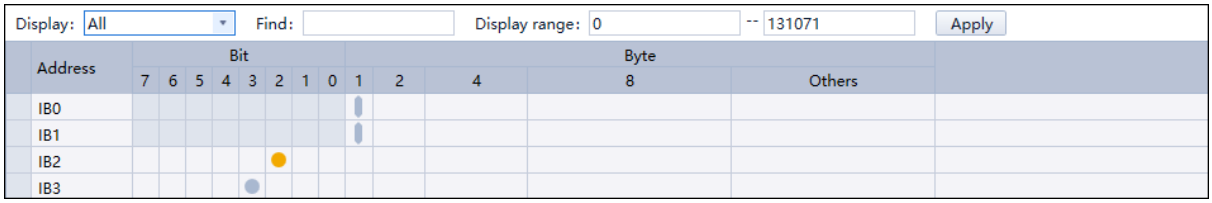


4. The background color is used to distinguish different usage scenarios of addresses. A light background indicates that only a single bit of the address is accessed in the program, not the entire byte. For example, variable a uses only the IX3.3 address. A dark background indicates that the address is accessed as an entire byte. For example, variable b uses the IB1 address. Additionally, if an address is assigned to multiple (≥ 2) variables, the address uses an orange-yellow icon to indicate that this address is shared by multiple variables. For example, variables d and e both use the %IX2.2 address.
   - As shown in the figure below, four variables: a, b, d, and e, are defined in the main function.



	Name	Data type	Address	Initial value	Power failure reten...	Constant	Network public	Comment	Property
1	VAR								
2	a	BOOL	%IX3.3				Private		
3	b	BOOL	%IB1				Private		
4	d	BOOL	%IX2.2				Private		
5	e	BOOL	%IX2.2				Private		

   - The colors of the addresses corresponding to a, b, d, and e can be viewed in the resource usage table.



5. Hover the mouse over a variable to view the variable name. Right-click a variable and select "Go to definition" to jump to the declaration location of this variable; select "Cross-reference table" to view the usage information of this variable.

10.1.1.10 Variable Operations

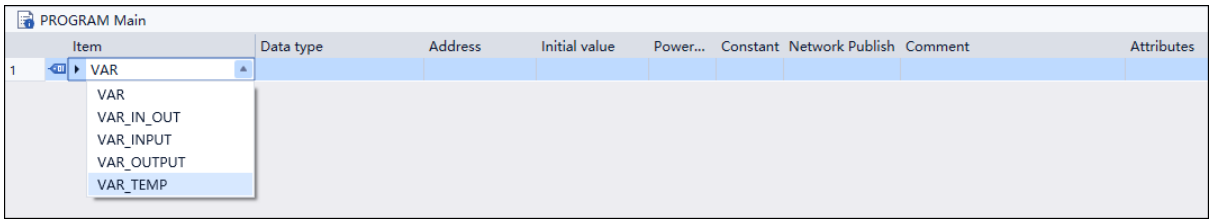
10.1.1.10.1 Adding a Variable

Prerequisites

A POU has been created.

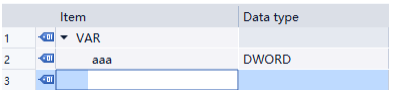
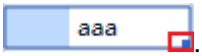
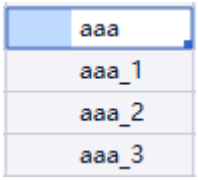
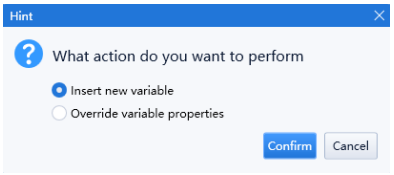
Procedure

1. Open a POU under "Programming" and set the variable category in the variable declaration editor in the table view.



2. Add tags.
   Variables in the table view are managed in a list format, with one row representing one variable

(identifier).

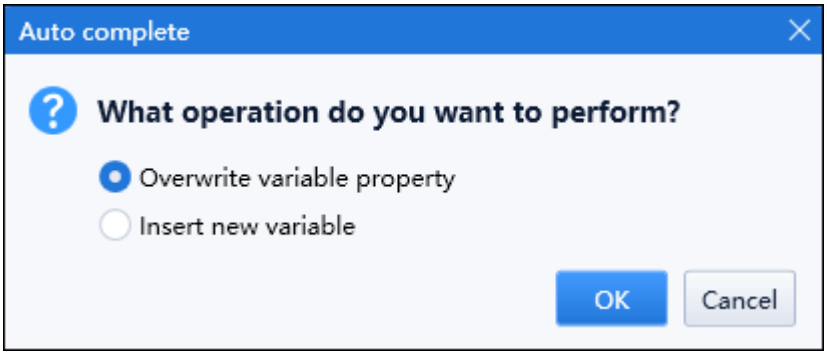
Number	Procedure	Example
One	Double-click the text box , enter a valid variable name, and press Enter to add the variable. The software will automatically fill in the data type with the default value.	
Multiple	Click the name of an existing variable, and hold and pull down the square in the lower right corner of the cell of each column . In the pop-up window, select "Insert new variable" and the new variable will copy the target variable's name and data type by default. The other properties of the new variable depend on whether the corresponding cells are held and pulled down. When adding, the software will automatically check whether the new variable name duplicates with the existing names.  Add or insert new variables in the drop-down area. Existing variables will shift down automatically.	
Multiple	Variables can also be added by batch import. For details, see 10.1.1.10.6 Variable Import.	-

10.1.1.10.2 Copying and Pasting a Variable Table

Users can select partial cells in the variable table for copy-and-paste operations with Excel, improving variable editing efficiency.

Copying Excel content to variable table

1. Copy Excel content.
2. In the variable table, right-click the target location and select "Paste".
3. In the pop-up window, select the operation to execute, and click "OK".
   - Overwrite variable property: Based on the top-left cell of the selected area, paste and overwrite the existing content.
   - Add variable: Insert the data to be pasted below the selected area without overwriting the existing content.



NOTE

- The property column does not support paste operations.
- Members of array variables, structure variables, and FB instances do not support paste operations. During batch pasting, the content in these cells is automatically ignored.
- When the data type column has no data, the system defaults to "DWORD". When the network public column has no data, the system defaults to "Private".
- If the pasted data in the data type column does not match the system type, an error prompt is displayed in that cell.
- If the pasted data in the address column does not match the address format, an error prompt is displayed in that cell.
- If the pasted data in the initial value column does not match the data type, an error prompt is displayed in that cell.
- When pasting data to the power failure retention column and constant column, if the data is TRUE or FALSE, the system converts it to a checkbox; non-BOOL values are unchecked by default.
- When pasting data to the network public column, if the pasted content matches an item in the cell's dropdown list, it is pasted; otherwise, the original state is retained.

Copying variable table content to Excel

1. Use the Shift key to select data in the variable table.
2. Paste in Excel.

NOTE

When pasting data from the power failure retention column and constant column to Excel, it is automatically converted to TRUE or FALSE.

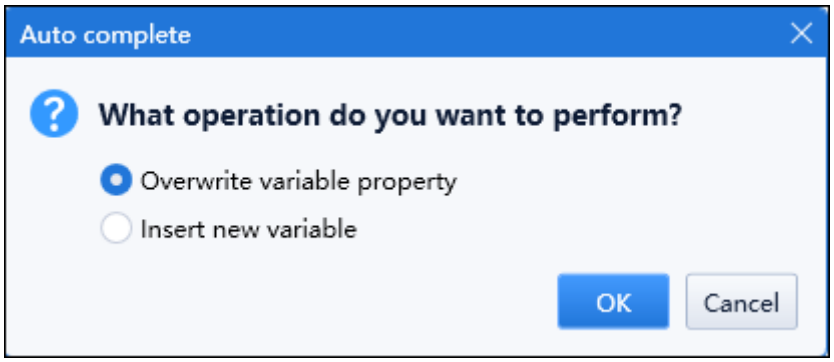
10.1.1.10.3 Setting Variable Properties

After adding variables in the corresponding POU, users can set variable properties as needed. For detailed property descriptions, see 10.1.1.9.3 Variable Declaration Property.

Variable properties support batch processing. Users can hold and pull down the square in the lower right corner of the cell (

) to select the range for batch processing.

In the pop-up "Reminder" window, select "Overwrite variable property" to complete batch modifications of multiple rows of properties.

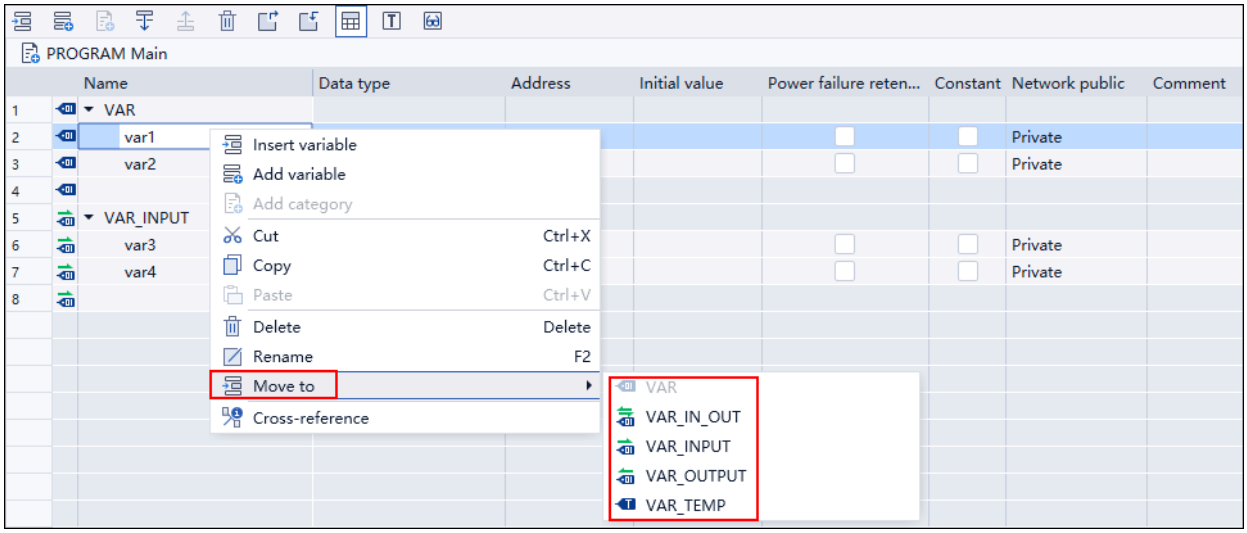


10.1.1.10.4 Changing Variable Category

After adding variables in the corresponding POU, users can change the category of the added variables as needed.

Procedure

Right-click a variable and select "Move to" to move this variable to another category.



NOTE

After changing the category, the variable will be removed from the original category; this operation does not affect any properties of the variable.

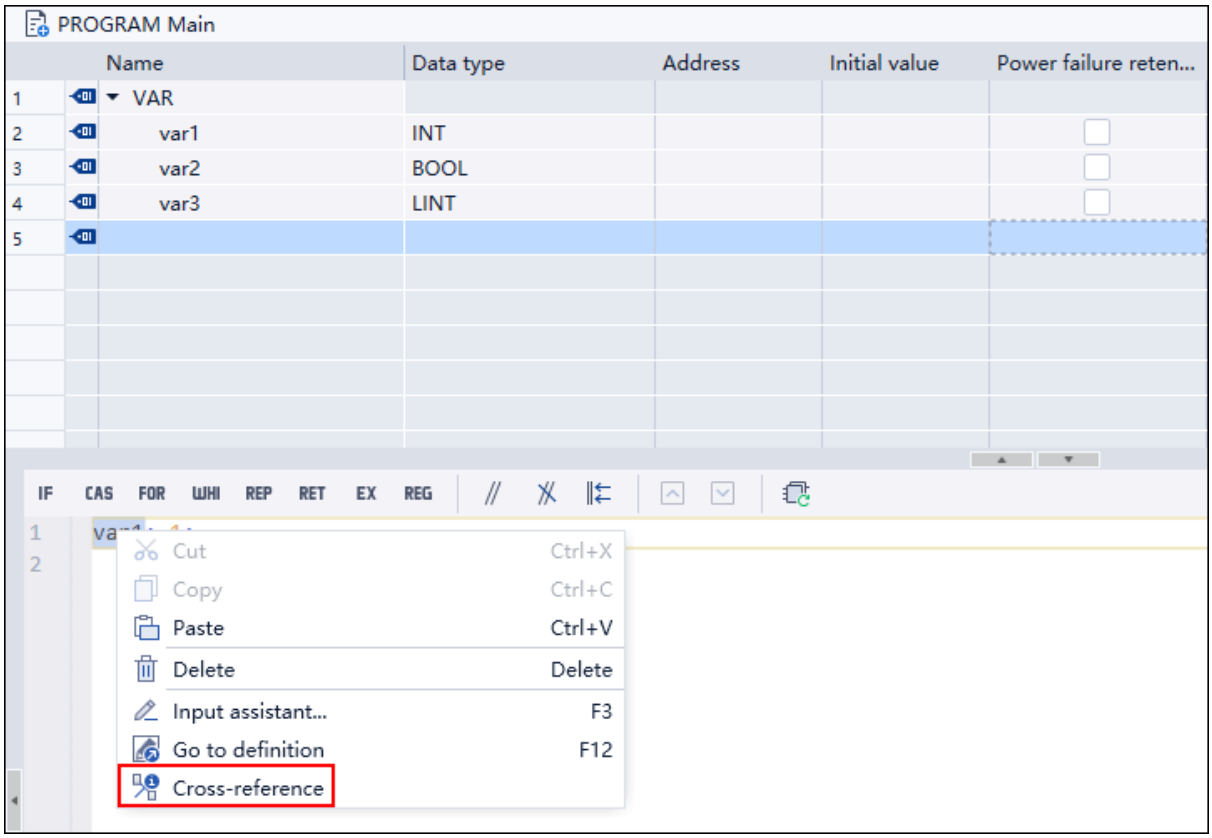
10.1.1.10.5 Cross-Reference

Prerequisites

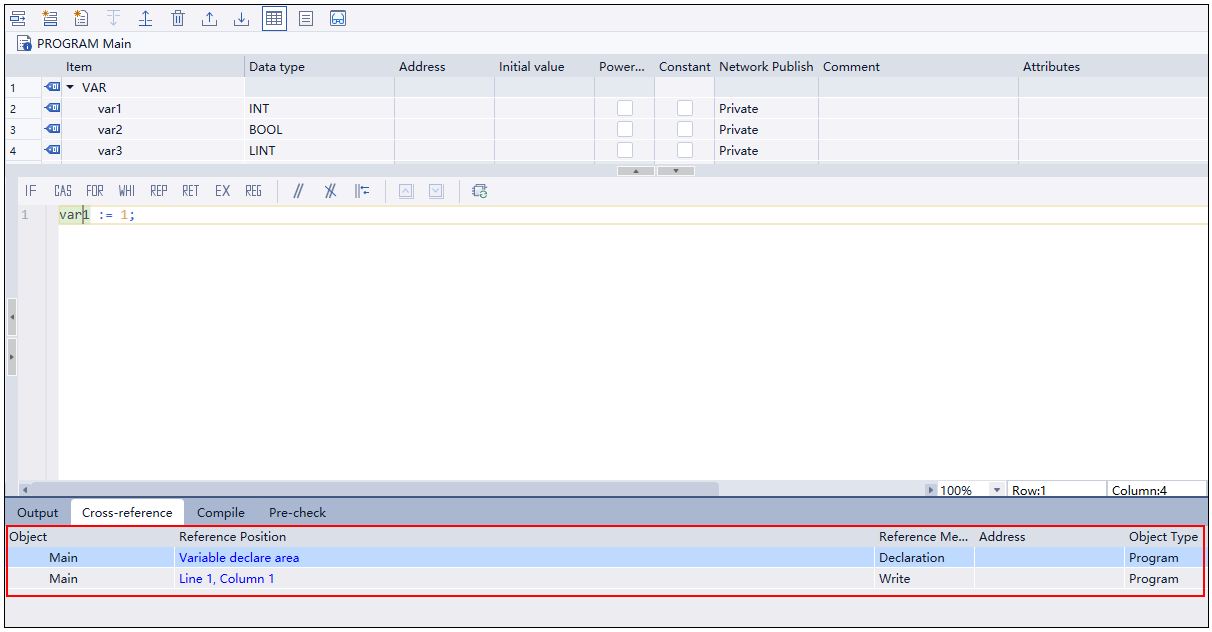
- The editing area contains variables.
- Variables are used by the POU.

Procedure

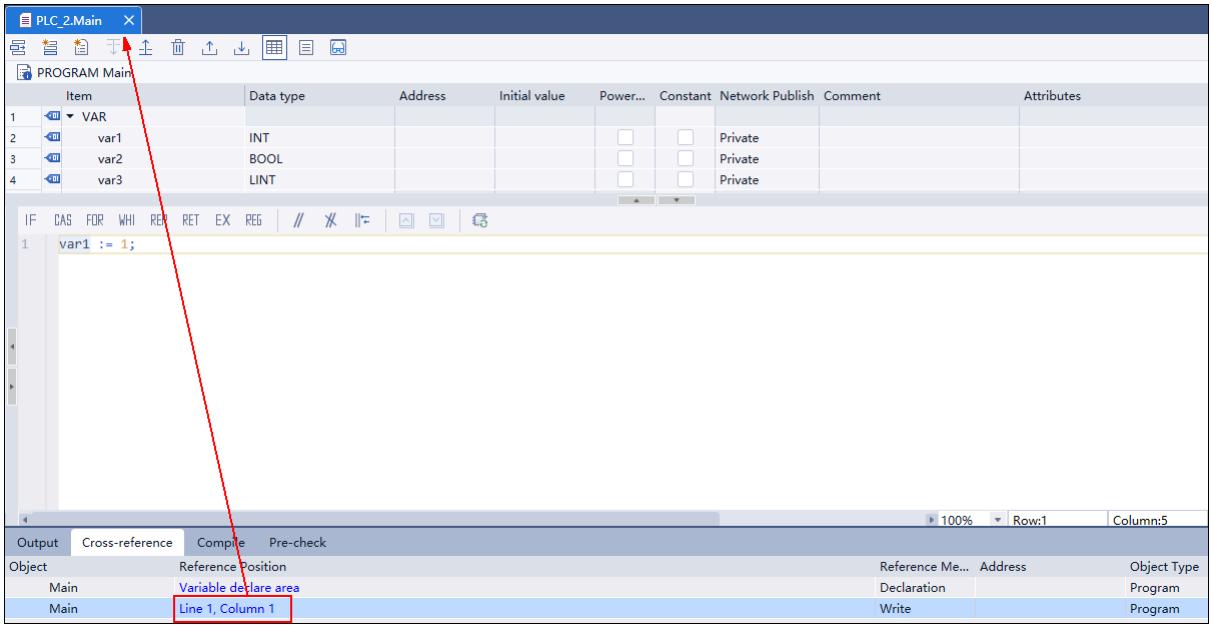
1. Right-click the variable in the editing area and select "Cross-reference", as shown in the following figure.



2. The "Cross-reference" tab will be displayed at the bottom of the interface, showing the usage information of the current variable, as shown in the following figure.



3. Click the link in the "Reference position" column to navigate to the corresponding interface.



10.1.1.10.6 Variable Import

Prerequisites

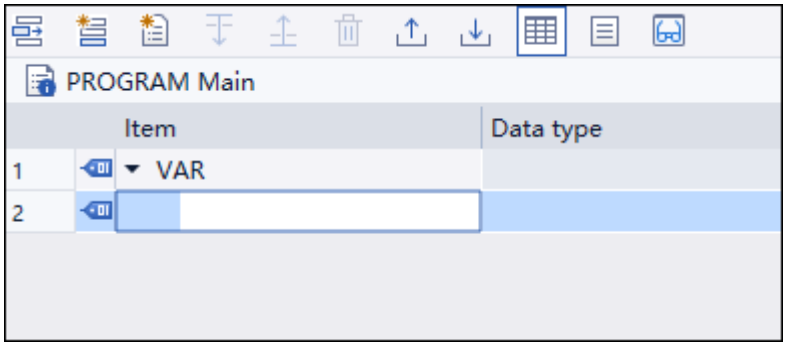
A file that meets the import conditions already exists.

Procedure

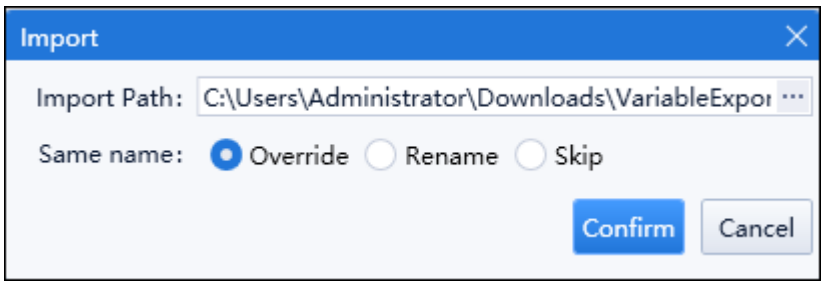
1. Open the variable export file "VariableExport.xlsx". Add variable var9, and delete variable var4, as shown in the figure below.

Data type	Address	Initial value	Power off retentior	Constant	Network Publish	Comment	Attributes
INT			False	False	Private		
BOOL			False	False	Private		
LINT			False	False	Private		
BYTE			False	False	Private		
BOOL			False	False	Private		
DINT			False	False	Private		
DWORD			False	False	Private		
DWORD			False	False	Private		

2. In the variable declaration editor, click the "Import" button in the toolbar, and the "Import" dialog box will pop up.



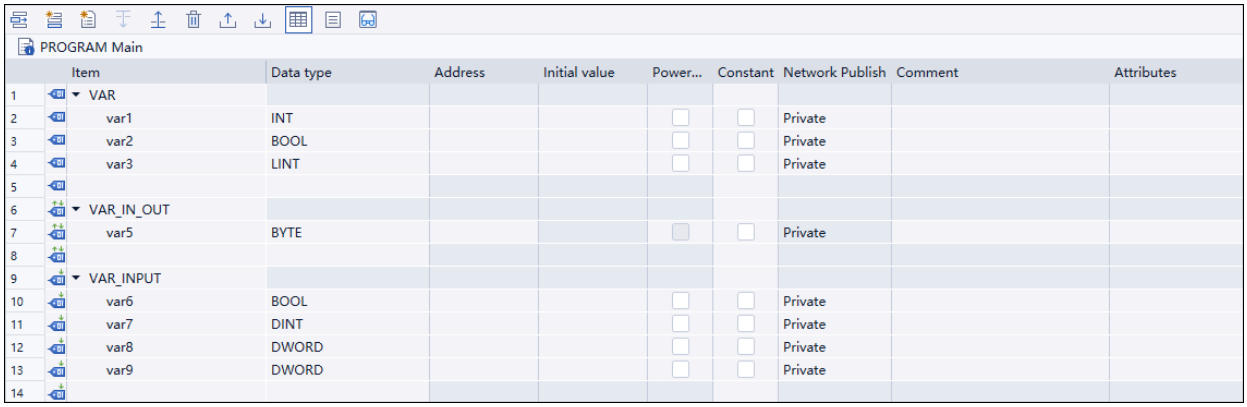
3. Select the file to import. For variables with the same name, the system supports covering, renaming, and skipping them. Click "OK" to import the variables.



4. If the import succeeds, an "Import successful" prompt will pop up. If the import fails, an "Import failed" prompt will pop up. Check the failure reason and resolve the fault before attempting to import again.

Result

The variables imported into the variable declaration editor are shown in the figure below.


PROGRAM Main

	Item	Data type	Address	Initial value	Power...	Constant	Network Publish	Comment	Attributes
1	VAR								
2	var1	INT					Private		
3	var2	BOOL					Private		
4	var3	LINT					Private		
5									
6	VAR_IN_OUT								
7	var5	BYTE					Private		
8									
9	VAR_INPUT								
10	var6	BOOL					Private		
11	var7	DINT					Private		
12	var8	DWORD					Private		
13	var9	DWORD					Private		
14									

10.1.1.10.7 Variable Export

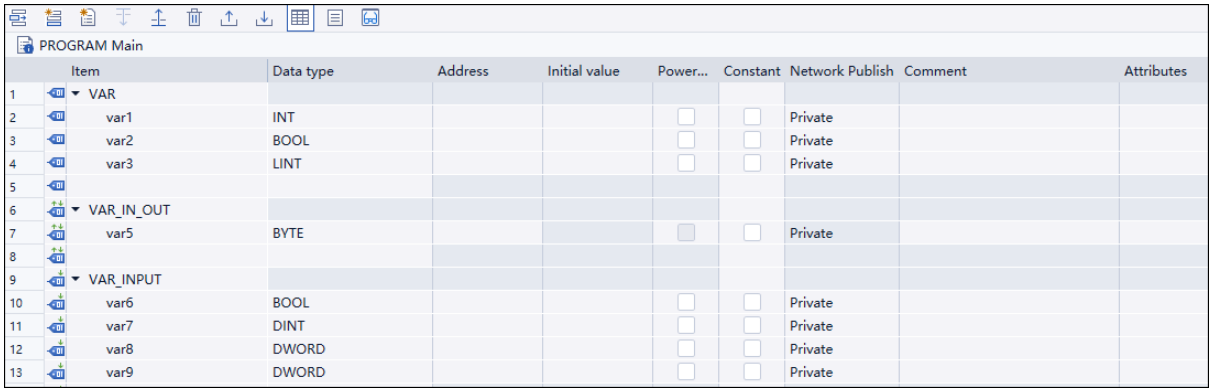
The variable tables in POU support importing and exporting variables (in XLSX format). Users can also add, delete variables, and modify variable properties in Excel according to the format. The Excel can then be imported back into the iFA variable table.

Prerequisites

A POU has been created.

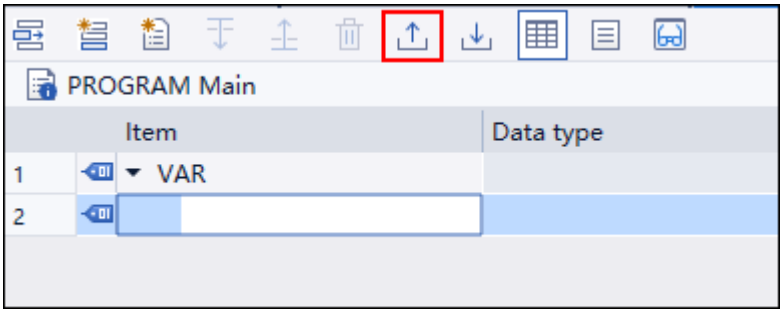
Procedure

1. Add some variables in the variable table.

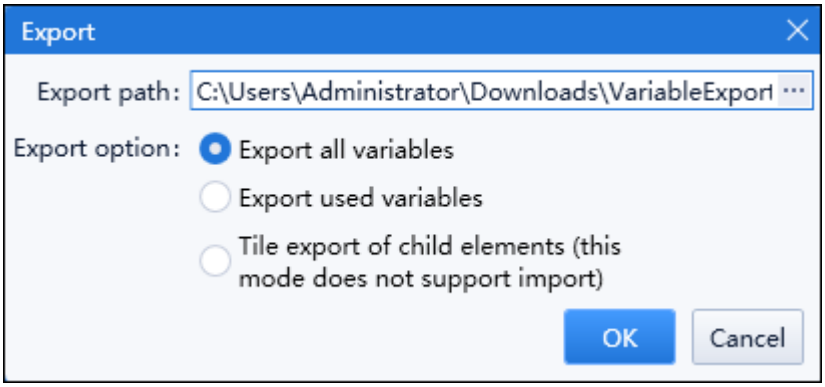

PROGRAM Main

	Item	Data type	Address	Initial value	Power...	Constant	Network Publish	Comment	Attributes
1	VAR								
2	var1	INT					Private		
3	var2	BOOL					Private		
4	var3	LINT					Private		
5									
6	VAR_IN_OUT								
7	var5	BYTE					Private		
8									
9	VAR_INPUT								
10	var6	BOOL					Private		
11	var7	DINT					Private		
12	var8	DWORD					Private		
13	var9	DWORD					Private		

2. In the variable declaration editor, click the "Export" button in the toolbar, and the "Export" dialog box will pop up.



3. Select the path for export and select the export range in "Export option". Click "OK" to export the variables.



4. If the export succeeds, a prompt window will pop up, asking users whether to open the exported file. If the export fails, an "Export failed" prompt will pop up. Check the failure reason and resolve the fault before attempting to export again.

> NOTE
>
> When "Tile export of child elements" is checked, the variables will be exported in full name mode as "namespace.variable name", displaying all child elements.

Result

Take the configuration of "Tile export of child elements" as an example. The variable export file "VariableExport.xlsx" is shown below.

VariableType	Item	Data type	Address	Initial value	Power off retentior	Constant	Network Publish	Comment	Attributes
VAR	AAA.var1	DWORD			False	False	Private		
VAR	AAA.var2	DWORD			False	False	Private		
VAR	AAA.var3	DWORD			False	False	Private		
VAR_OUTPUT	BBB.var1	DWORD			False	False	ReadWrite		
VAR_OUTPUT	BBB.var2	DWORD			False	False	ReadWrite		
VAR_OUTPUT	BBB.var3	DWORD			False	False	ReadWrite		

10.1.1.10.8 Smart Input

When entering or deleting any character in the offline editing area, the system matches data based on the word currently selected by the cursor. If there is matching data, a smart input page will pop up for user selection, providing convenience for operations.

Applicable position:

- Global variable table
- Variable declaration area of program blocks
- Program editing area
- I/O mapping

Editing data types in the variable list

- Letter search
  Double-click the data type cell of the target variable, and the software will search for relevant data types based on the entered letters.

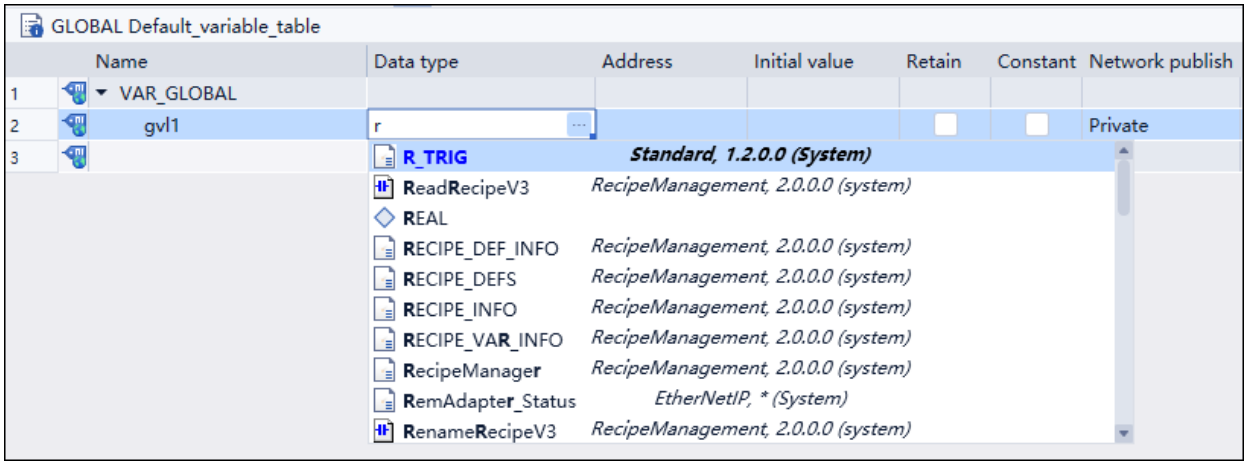


Figure 10-1-1-10-8-1 Operations in the global variable table

Search results are displayed in alphabetical order based on the position of the input content within the data type. Data types are also categorized as follows.

: Standard data types, such as BOOL and REAL.

: Function block types, such as RTC and TOF. Built-in function blocks will display information about the library they belong to.

: Function block types suitable for application libraries, such as MotionControl library and RecipeManagement library.

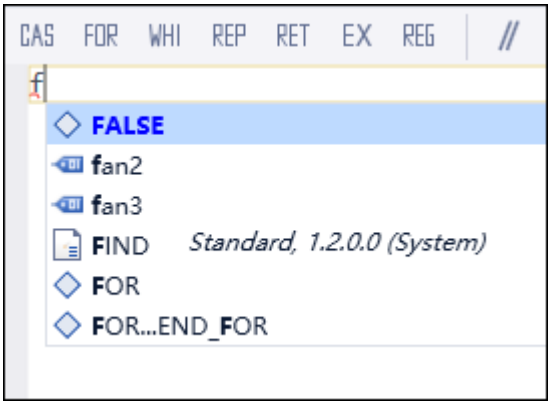
: Complex data types, such as structures and enums.

Users can complete input by double-clicking the target object.

Editing programs in the ST editor

- Program input
  1. Text input

     When entering letters in the ST program editing area, the software will automatically search for relevant keywords (in ST language), data types, variable names, and function block instance names.



     Results are displayed in alphabetical order based on the position of the input content within each item, and the results are categorized as follows.

     - : Standard data types and keywords (in ST language), such as BOOL and IF.
     - : Function block types, such as RTC and TOF.
     - : Function block types suitable for application libraries, such as MotionControl library and RecipeManagement library.
     - : Algorithm library names.

- : Global variables.
- : Local variables.

2. Target object selection

   Double-click the target object to complete the input.

10.1.1.10.9 Parameter Update

When an FB instance, FC, PRG, or Meth is called by the program, if the parameters of the called object change, the parameter update function can be used to update the parameters in the program editing area with one click.

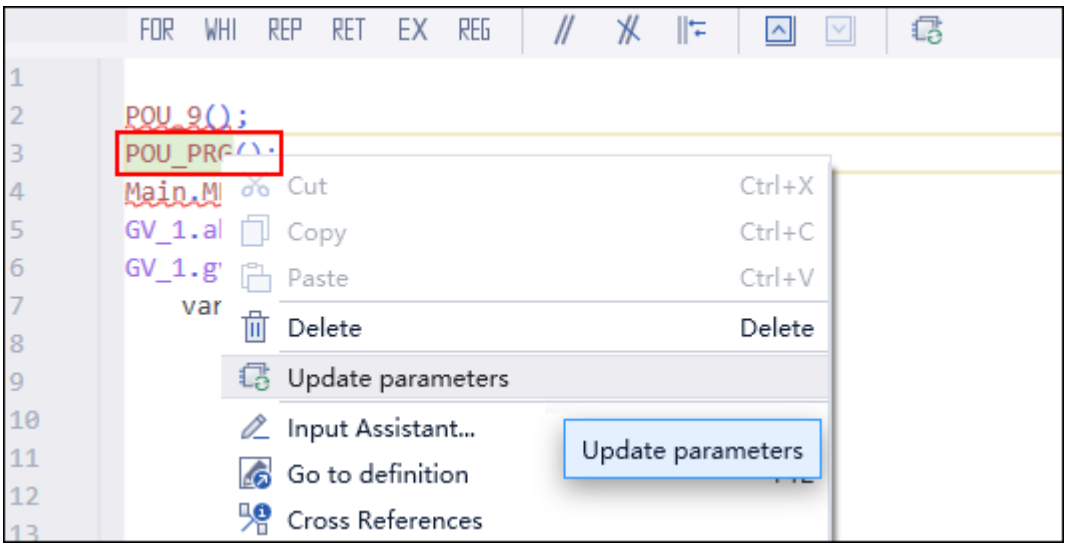
Prerequisites

- The FB instance is called in the editing area.
- The FC, PRG, or Meth is called within the editing area.

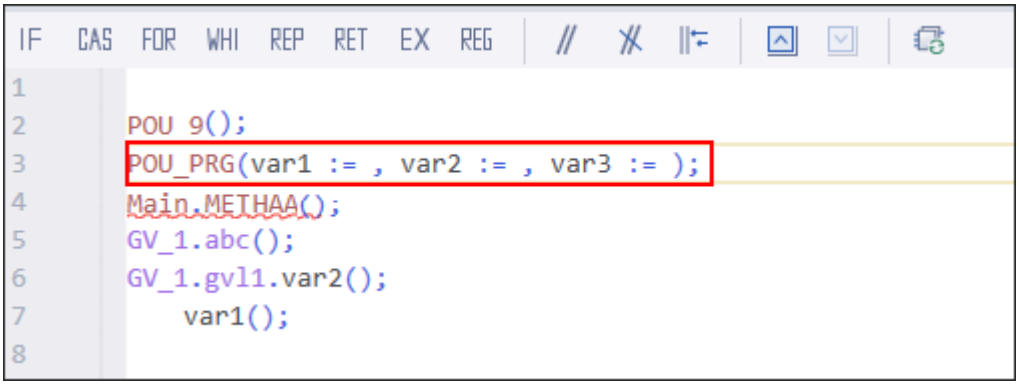
Updating parameters in the ST editor

For the ST editor, the parameter update function is mainly used to complete missing parameters. If the parameters of the called object are deleted, the parameter update operation will not synchronously delete the parameters at the call site; if the parameter name of the called object is refactored, the parameters at the call site will automatically synchronize without manual updates.

1. Right-click the FB instance, FC, PRG, or Meth called in the editing area and a menu will pop up. Click "Update parameter", as shown in the following figure.



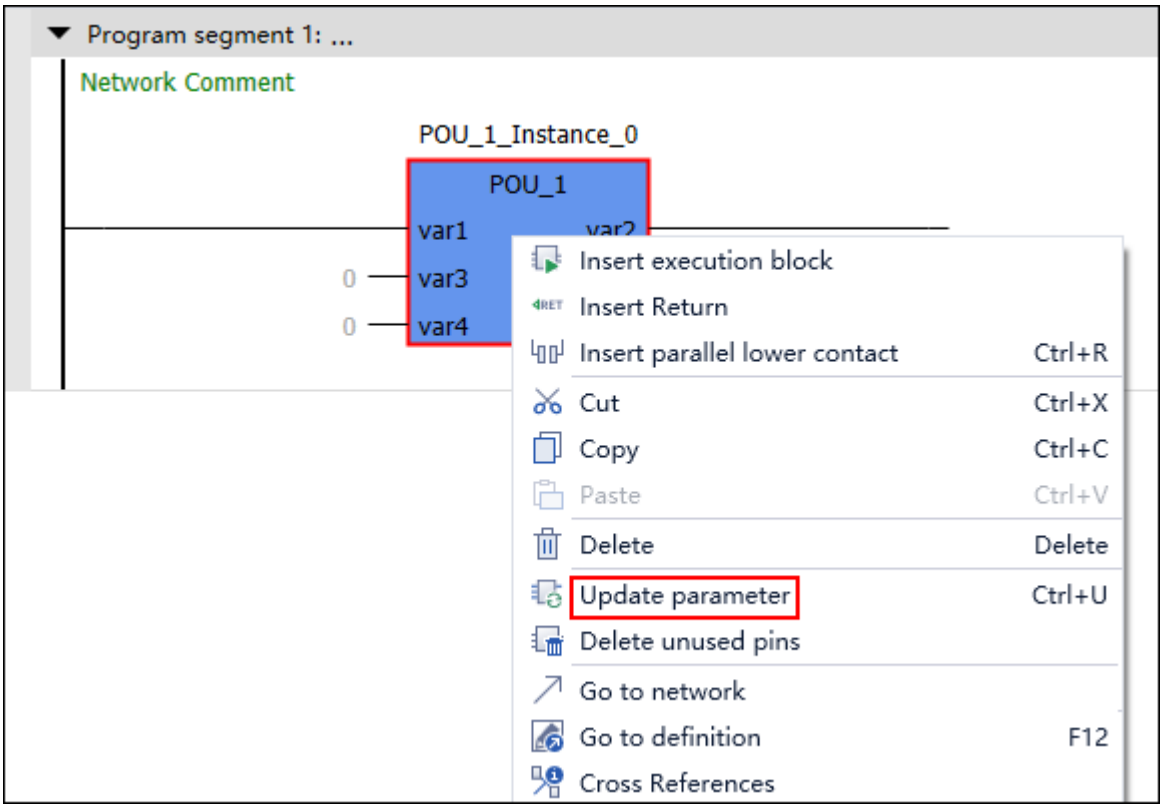
2. Update the parameter list of the target instruction, as shown in the following figure.



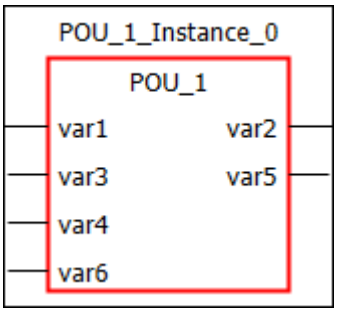
Updating parameters in the LD editor

When the parameters of the called object change (including addition, deletion, and parameter name refactoring), a red border appears around the block in the program editing area, indicating that you can update parameters of an individual block or update all parameters.

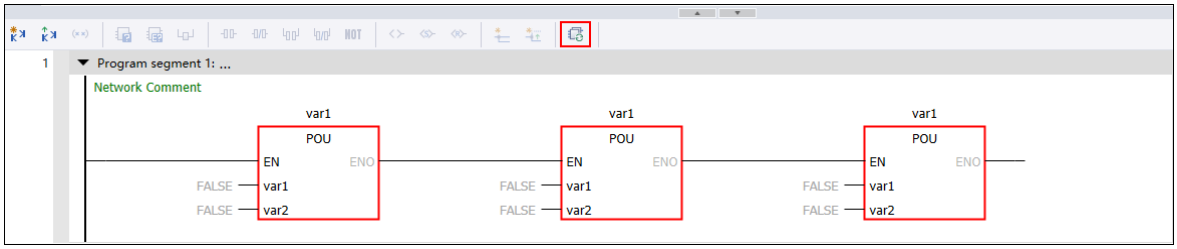
- Updating parameters of an individual block
  1. Right-click an individual block in the editing area. In the pop-up menu, click "Update parameter", as shown in the following figure.



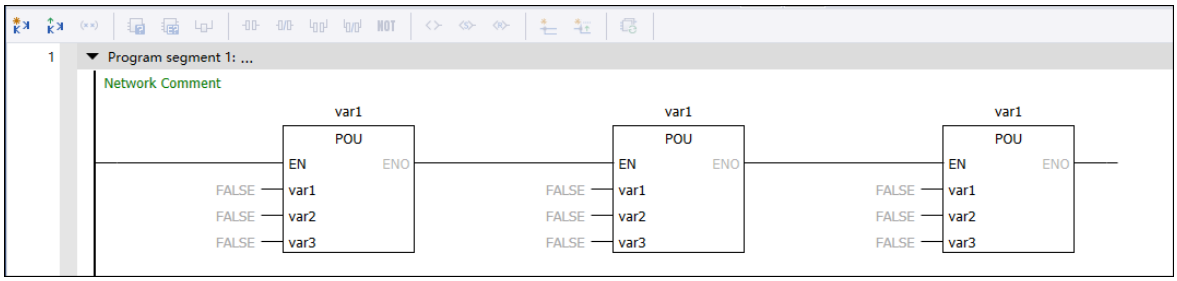
  2. Update the target instruction parameters to align the parameters in the current block with the parameters of the called object.



- Updating all parameters
  1. Click the "Update all parameter" button in the menu bar of the editing area, as shown in the following figure.



2. Perform a batch update of all blocks with parameter changes in the editing area.



10.1.1.10.10 Input Assistant

The input assistant helps users quickly search for and input target data types or variables while editing variables, reducing programming time.

Applicable locations:

- Global variable table
- Variable declaration area of program blocks
- Program editing area
- I/O mapping

Editing data types in the variable list

- In the input assistant window, select the target variable.

  Double-click the data type cell of the target variable and click ... to select "Input assistant".

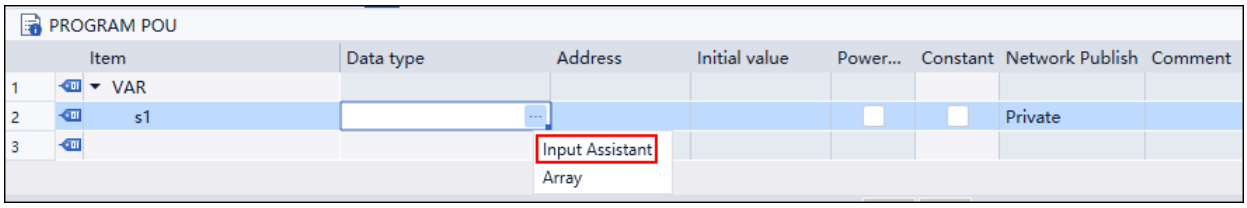


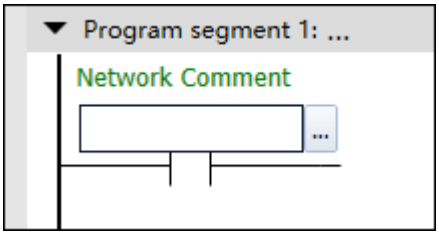
Figure 10-1-1-10-10-1 Operations in the variable declaration area of program blocks

Open the input assistant window, double-click the target object to complete the input of the data type.

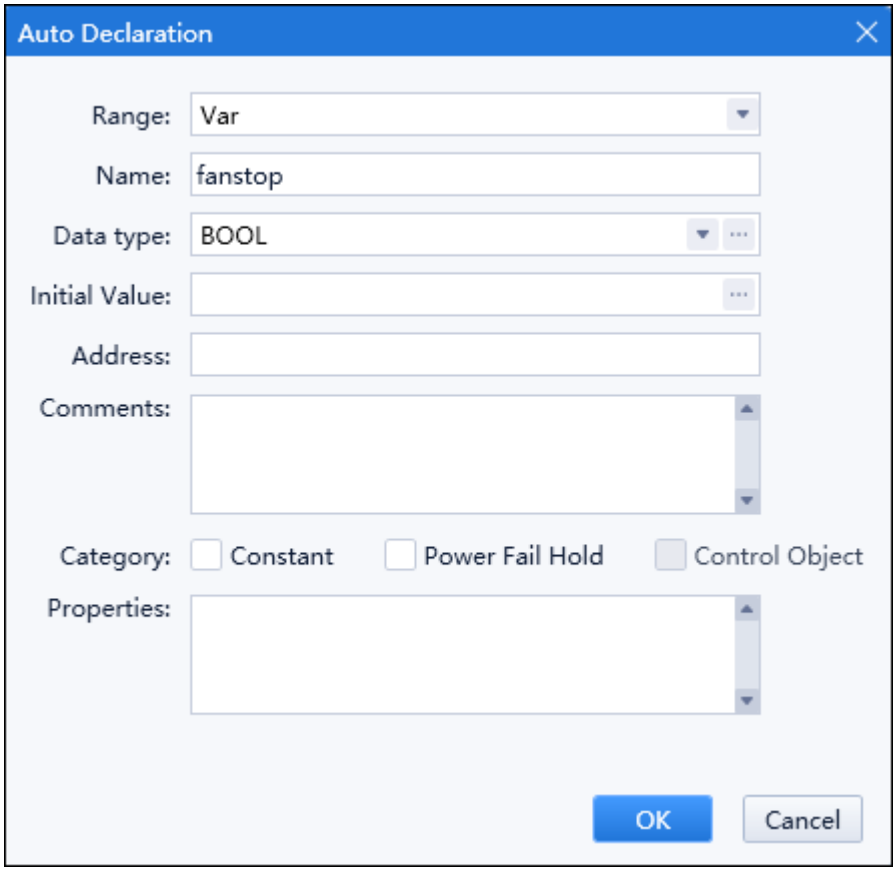
Editing programs in the LD editor

- Declaring variables

  1. Double-click the variable name of the component. The default is ???.



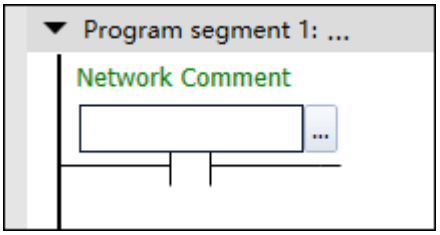
  2. Enter the desired variable name in the text box and press the Enter key.



3. Select > and open the input assistant window.
4. Double-click the target data type to complete the data type setting.

- Associating variable name or instance name with components

1. Double-click the variable name of the component. The default is ???.

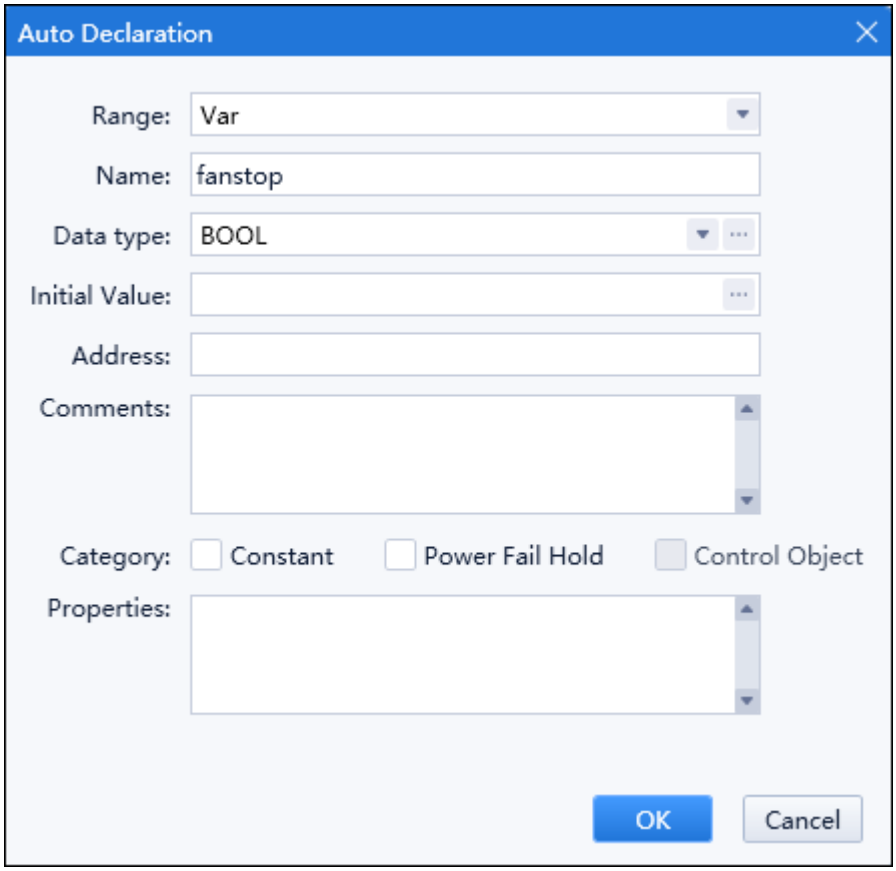


2. Select ... and open the input assistant window.
3. Select the target variable name or instance name to complete the association operation.

Editing programs in the ST editor

- Declaring variables

1. Enter the desired variable name.
   In the ST program editing area, enter the desired variable name, and press the Enter key.



2. Select > and open the input assistant window.
3. Double-click the target data type to complete the data type setting.

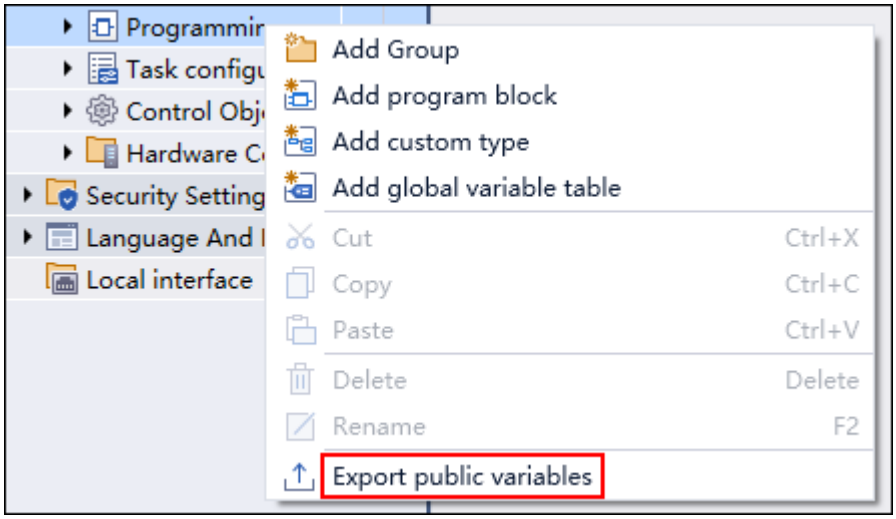
10.1.1.10.11 Public Variable Export

Prerequisites

The network public property of the variable is set to public, which can be "Read/Write", "Read-Only", or "Write-Only".

Procedure

1. After the project has been compiled, right-click "Programming" > "Export public variables", and an "Inquiry" dialog box will pop up.



2. Click "Confirm" and select the export path to export public variables in XML format. After successful export, a prompt will appear, saying "Public variables exported successfully".

Result

The exported XML file for public variables is as shown in the following figure.

```
<?xml version="1.0" encoding="utf-8"?>
<Symbolconfiguration>
  <Header>
    <ProjectInfo name="plc" devicename="PLC_1" appname="Application" codeId="341b70a9-0000-0000-0000-0000000(
  </Header>
  <TypeList>
    <TypeSimple name="T_BOOL" size="1" iecname="BOOL" typeclass="Bool" />
    <TypeSimple name="T_INT" size="2" iecname="INT" typeclass="Int" />
    <TypeSimple name="T_LINT" size="8" iecname="LINT" typeclass="LInt" />
    <TypeSimple name="T_DWORD" size="4" iecname="DWORD" typeclass="DWord" />
  </TypeList>
  <NodeList>
    <Node name="Application">
      <Node name="Main">
        <Node name="var1" type="T_BOOL" access="readwrite" offset="292" />
        <Node name="var2" type="T_BOOL" access="readwrite" offset="293" />
        <Node name="var3" type="T_INT" access="readonly" offset="294" />
        <Node name="var4" type="T_LINT" access="writeonly" offset="296" />
        <Node name="var5" type="T_DWORD" access="readonly" offset="304" />
        <Node name="var6" type="T_DWORD" access="writeonly" offset="308" />
        <Node name="var7" type="T_DWORD" access="readwrite" offset="312" />
        <Node name="var8" type="T_DWORD" access="readwrite" offset="316" />
        <Node name="var9" type="T_DWORD" access="readonly" offset="320" />
      </Node>
    </Node>
  </NodeList>
</Symbolconfiguration>
```

NOTE

The structure of the exported XML file depends on the configuration method of the public variables. For details, see 10.7.7.2 Public Variable.

10.1.1.10.12 Going to Definition

The "Go to definition" feature is used to help users quickly jump to the declaration location of a variable during program development, supporting both online and offline operations.

Applicable locations:

- Program editing area
- Recipe manager
- Resource Usage Table
- I/O mapping

Procedure

1. In the program editing area or recipe variable area, right-click the selected variable name, and select "Go to definition" in the pop-up menu or use the shortcut key "F12".

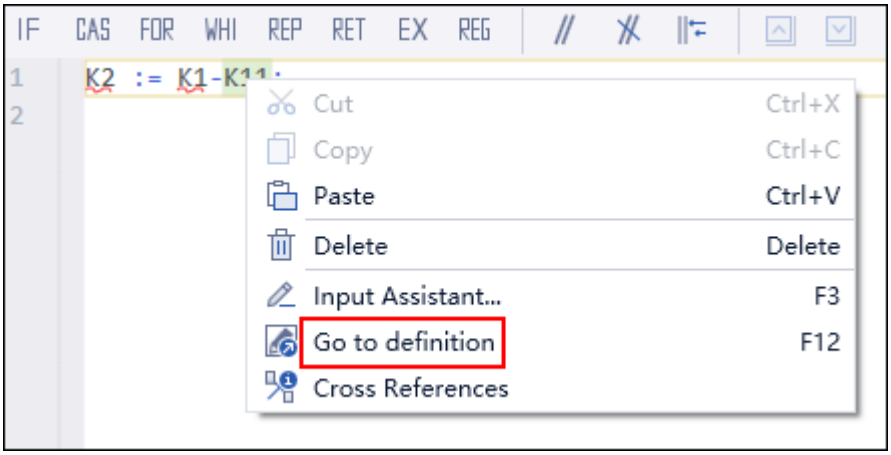


Figure 10-1-1-10-12-1 Operation in the program editing area

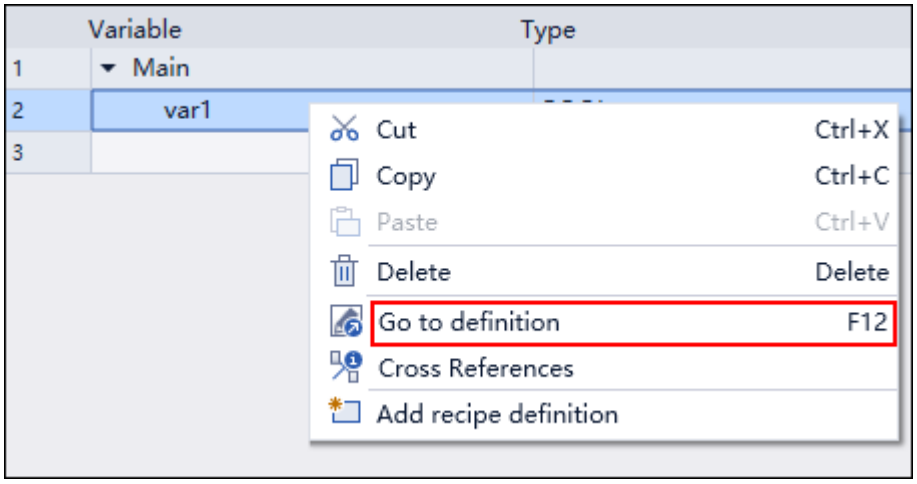


Figure 10-1-1-10-12-2 Operation in the recipe variable area

Result

The software will jump to the target variable's declaration location. For user-defined function blocks or functions, it will also open their internal programs simultaneously.

NOTE

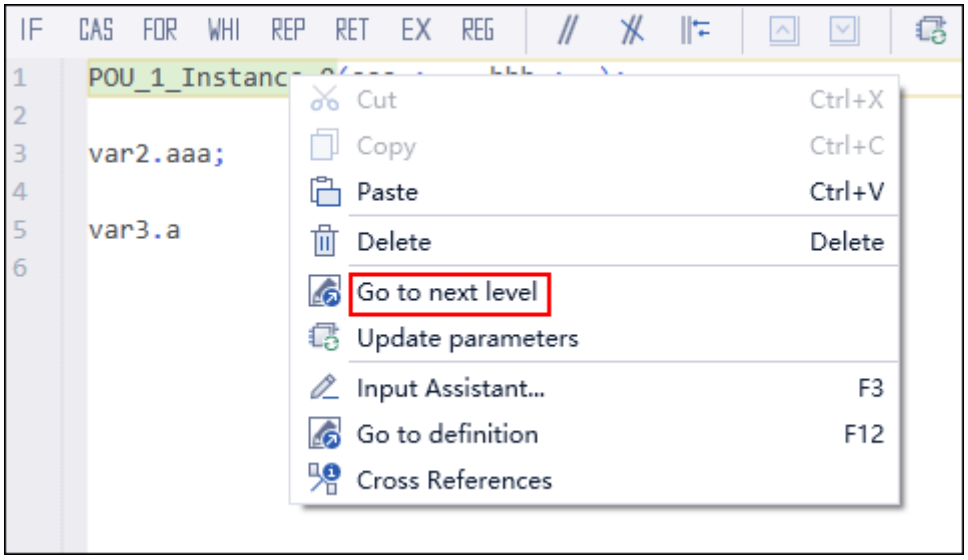
- When a variable in the program editing area is in a comment state, the "Go to definition" operation is not supported.
- For variables and types in the compilation library, the "Go to definition" operation is not supported.

10.1.1.10.13 Going to Next Level

The "Go to next level" feature is used to help users quickly jump to the declaration location of a statement during program development, supporting both online and offline operations.

Procedure

1. Right-click the selected instance or function in the program editing area, and select "Go to next level" from the pop-up menu.



Result

The software will jump to the target statement's declaration location. For user-defined function blocks or functions, it will also open their internal programs simultaneously.

> NOTE
>
> When statements in the program editing area are in comment status, the "Go to next level" operation is not supported.

10.1.2 Program Block

10.1.2.1 Program

10.1.2.1.1 Overview

A program is the core of task planning and has the highest calling authority, allowing it to call function blocks and functions.

Generally, three types of variables are defined in the program: Regular global variables, global variables mapped to hardware addresses, and local variables. The application logic is implemented through inter-program calls.

- **Custom program representation**
  Custom programs are represented using the following syntax expression.

```
Plain Text
PROGRAM <program name>
VAR_INPUT
… (*input interface variable declaration of the program*)
END_VAR
VAR_OUTPUT
… (*export interface variable declaration of the program*)
END_VAR
VAR
… (*local variable declaration of the program*)
END_VAR
```

```
Plain Text
...; // program logic
```

- **Variable declaration in a program**
  The syntax for declaring variables in PRG is as shown above. The types of variables that can be declared in PRG include local variables (VAR), input variables (VAR_INPUT), output variables (VAR_OUTPUT), input/output variables (VAR_IN_OUT), and temporary variables (VAR_TEMP). All properties of the variables are configurable.

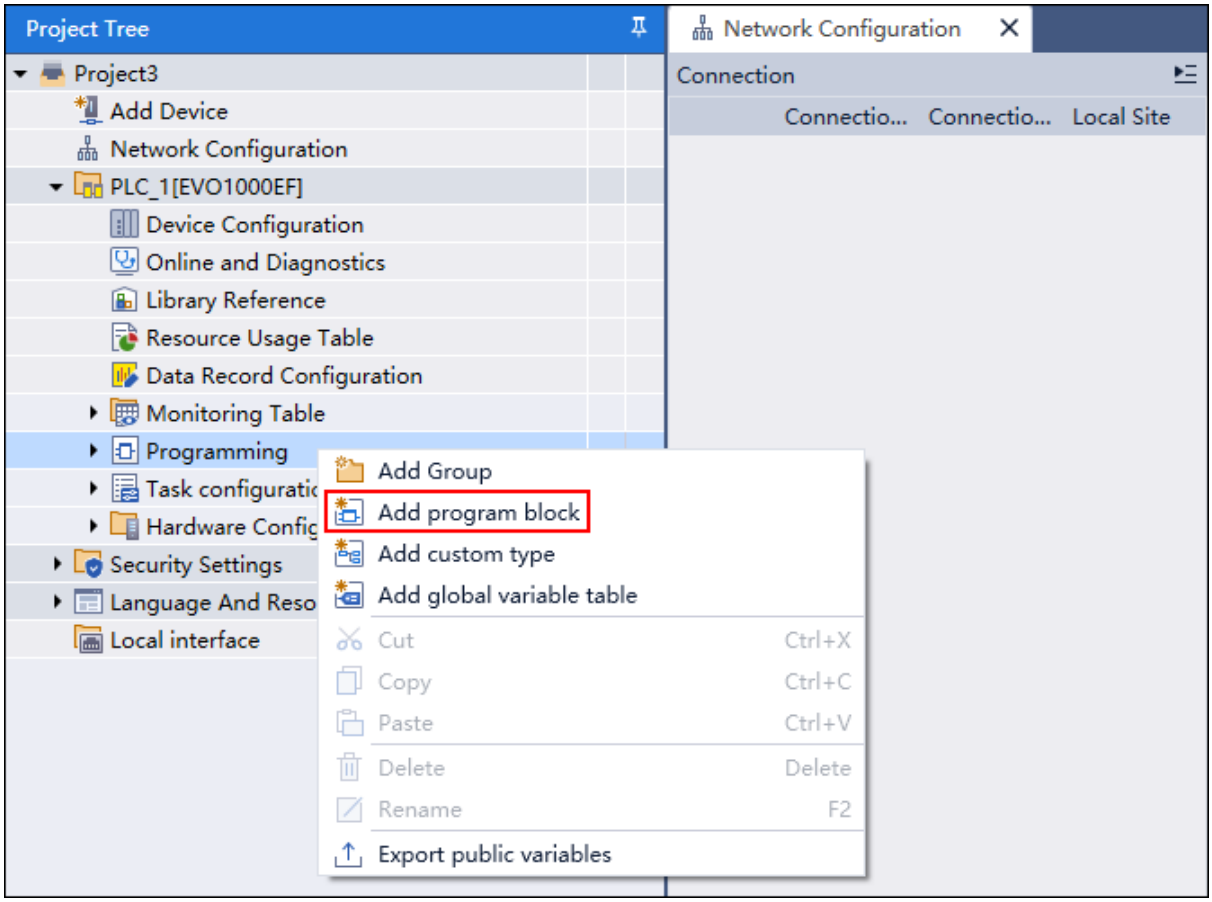
10.1.2.1.2 Adding a User Program

Prerequisites

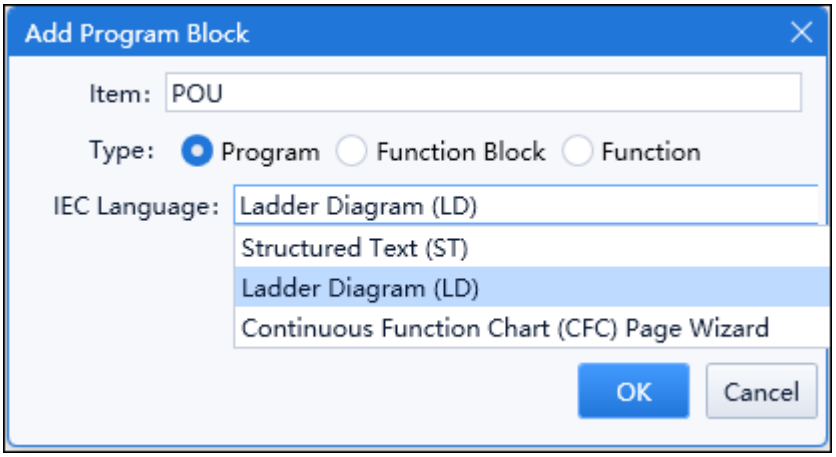
- A project has been created.
- A device has been added.

Procedure

1. Right-click "Programming > Add program block".



2. In the "Add program block" window, set the program name, select "Program" as the type, and select the corresponding IEC language, as shown in the figure below.



No.	Item	Description
①	Name	Indicates the program name. It supports letters, numbers, and underscores (_), and must start with a letter or underscore. Underscores cannot be used consecutively. The program name is case-insensitive, with a maximum length of 128 bytes. For example, __POU and PO__U are invalid names, while P_O_U is considered valid.
②	Type	Indicates the program block type. To add a user program, select "Program" here.
③	IEC Language	Supports LD, ST, and CFC programming languages.

3. Click "OK" to add the user program POU. The user program has a "(PRG)" suffix. When a user program is added, iFA will automatically open its editor. The editor is divided into the variable declaration area and the program editing area, which can be collapsed and expanded using and .

> NOTE
>
> When a POU is being compiled, the system will check the upper limit based on device requirements. For specific limitations, see the device specifications.

10.1.2.1.3 Adding a Method to a Program

Similar to function (FUN), a method (METH) has at least one input variable, and only returns one value. The difference is that methods must be created under a program (PRG) or function block (FB), and have access modifiers (PUBLIC, PROTECTED, PRIVATE, and INTERNAL). Methods can be used by functions, function blocks, and programs. When a method is called with the same input parameters, it always generates the same result as the return value.

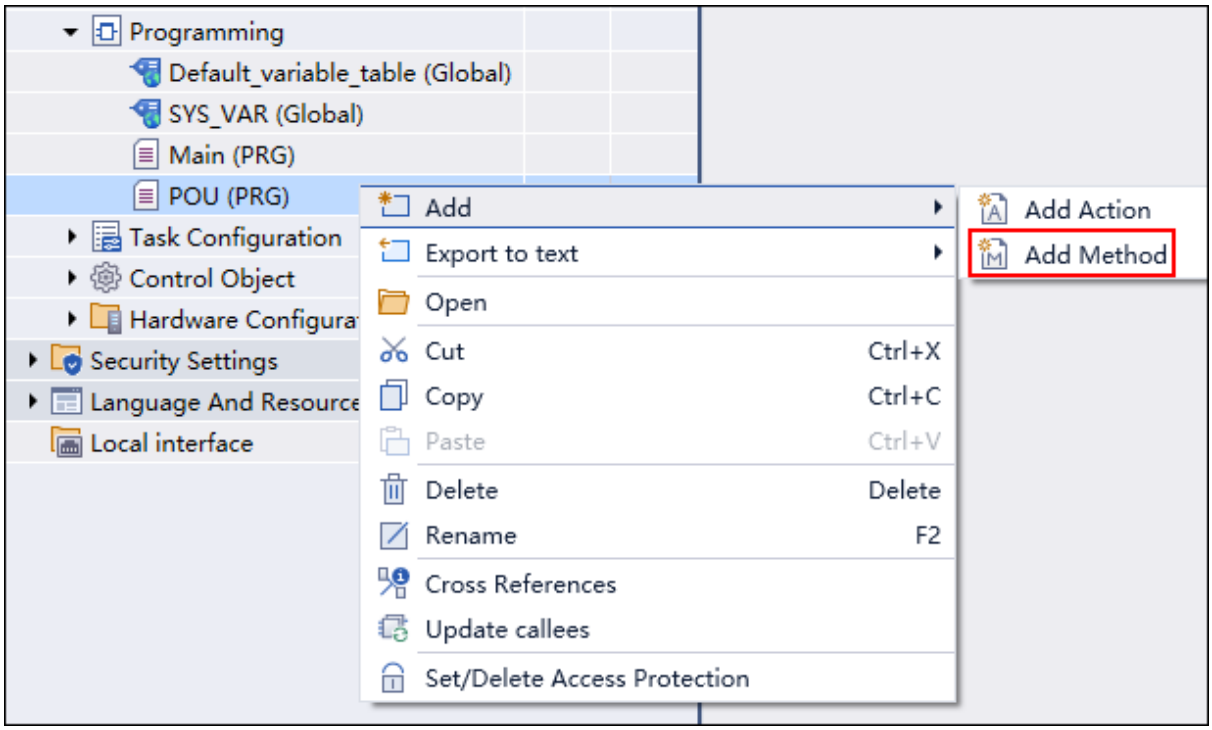
Method Creation

Prerequisites

A program (PRG) or function block (FB) has been created.

Procedure

1. Right-click an FB or PRG under "Programming" in the project tree, and select "Add > Add method", as shown in the figure below.



2. In the "Create method" window, set the name, return value type, IEC language, and access modifier of the method.

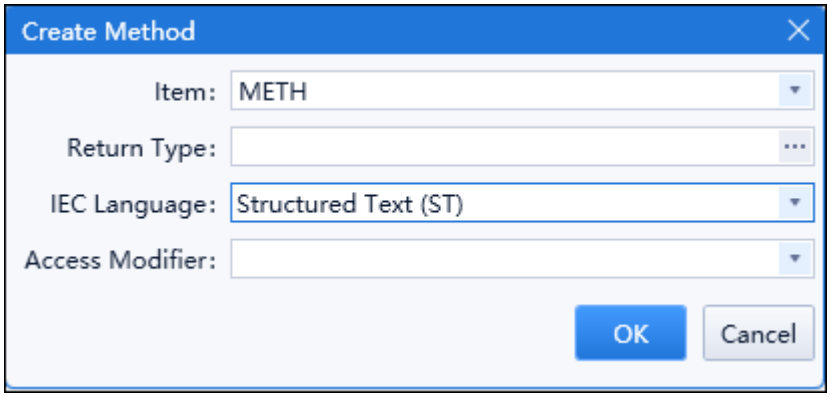


Table10-1-2-1-3-1 Method creation properties

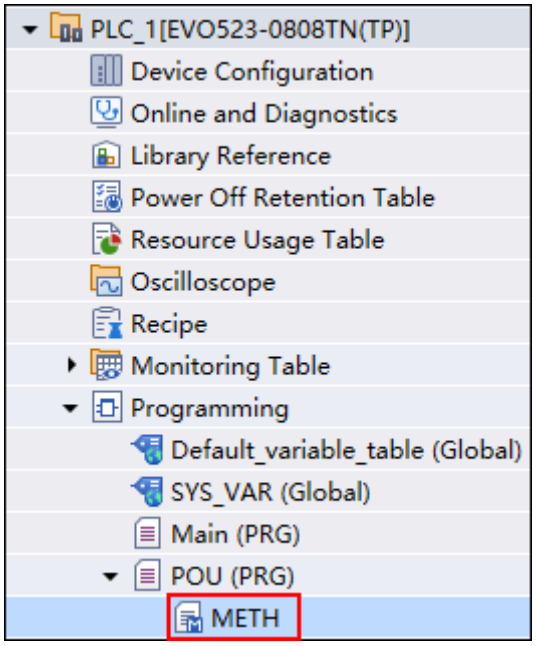
No.	Item	Description
①	Name	Indicates the method name. It must comply with the IEC61131-3 standard and is case-insensitive, with a length ≤ 128 bytes.
②	Return type	• The return value type must be specified and the length of the return value must be less than or equal to 128 bytes. • The return value type can be filled from the input assistant, which provides basic data types, custom types for the current project, and types from custom libraries. • The return value type is case-insensitive.
③	IEC Language	Supports LD and ST programming languages.
④	Access Qualifier	PUBLIC, PROTECTED, PRIVATE, and INTERNAL

Table10-1-2-1-3-2 Descriptions of access modifiers

Access Qualifier	Description
PUBLIC	Public access: Methods added under FB can be used across POU and across libraries.
PROTECTED	Protected access: Used for sub-function block to access the methods of base function block.
PRIVATE	Private access: Methods are only used within the PRG or FB itself and cannot be overridden by sub-function blocks.
INTERNAL	Access permissions within the namespace: Methods added under FB can be used in this project or this library project.

Result

A method (METH) is added under the program (PRG).



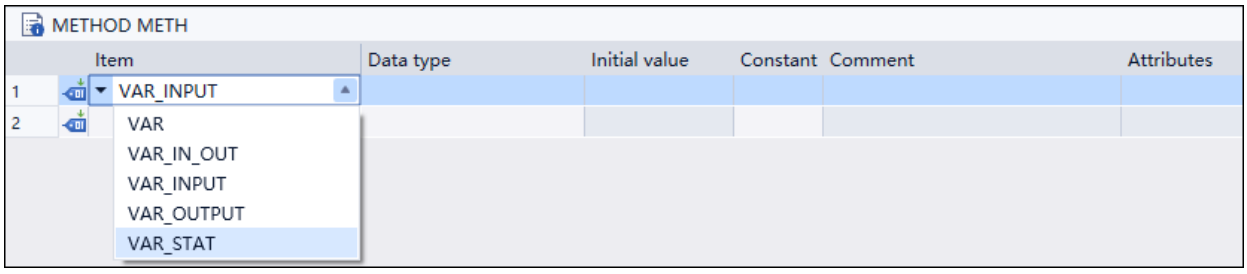
Method usage

Syntax for method representation and declaration

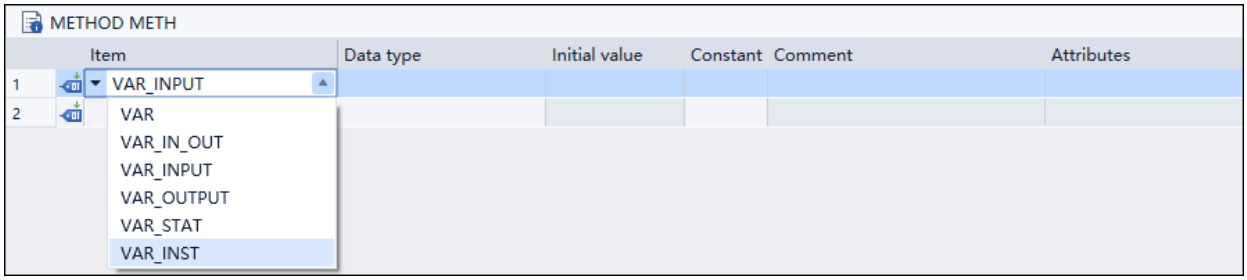
```
Go
METHOD  <access modifier>  <method name> : <return value type>
VAR_IN_OUT
// Input/output variable declaration
END_VAR
VAR
// Local variable declaration
END_VAR
VAR_OUTPUT
// Output variable declaration
END_VAR
```

Variable declaration in methods

- The variable types supported by methods under a program include local variables (VAR), input variables (VAR_INPUT), output variables (VAR_OUTPUT), input/output variables (VAR_IN_OUT), and static variables (VAR_STAT). All properties of the variables are configurable.



- The variable types supported by methods under a function block include local variables (VAR), input/output variables (VAR_IN_OUT), input variables (VAR_INPUT), output variables (VAR_OUTPUT), static variables (VAR_STAT), and instance variables (VAR_INST). All properties of the variables are configurable.



Precautions for variable declaration in methods:

- Supports multiple input variables and output variables. The caller must allocate variables to receive the output variables of the method.
- Supports only one return value (of output variable type).
- Storing values in internal variables is not supported, which is distinctly different from function blocks.
- Functions do not have specified memory allocation and do not need to be instantiated like function blocks.
- The actual arguments configured to VAR_INPUT by the caller can be empty, constants, variables, or function calls. In the case of a function call, The function is executed first, and its return value is passed as the actual argument.

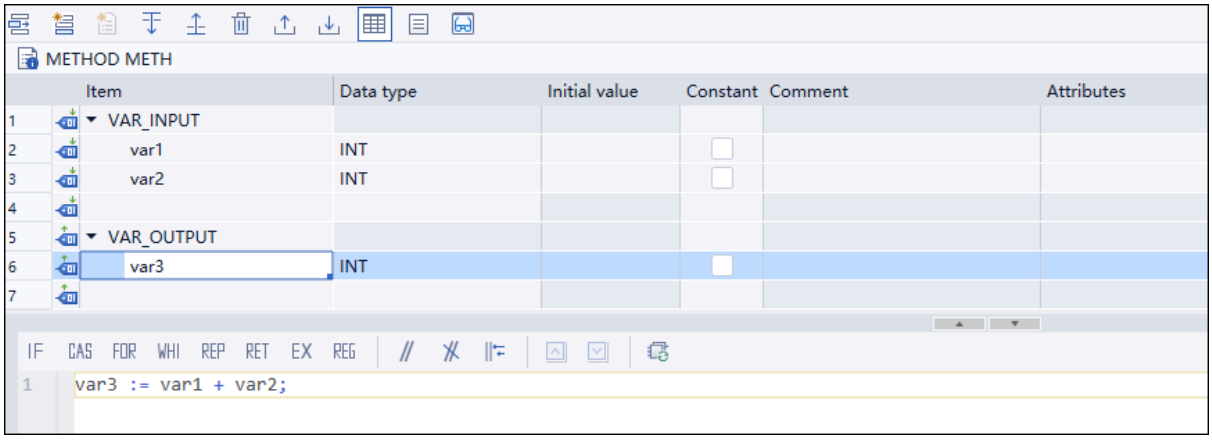
Method usage

- **Prerequisites**

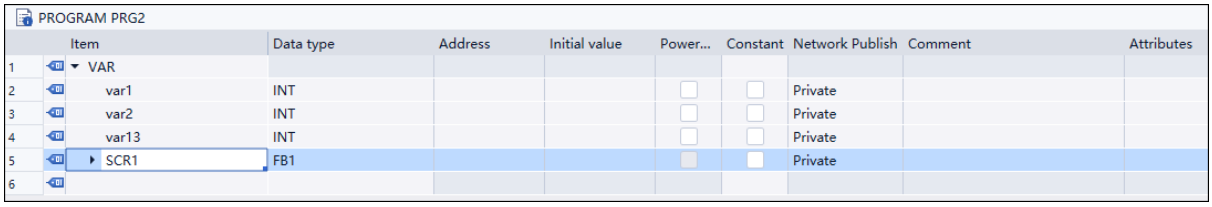
Methods have been added under the program (PRG) or function block (FB).

- **Procedure**

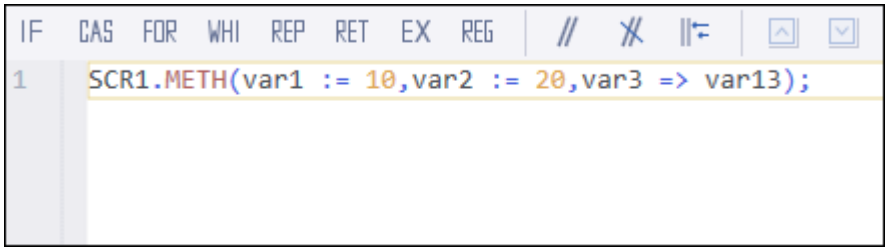
1. Add the method METH under function block FB1, with the variable table and programming logic as shown in the figure below.



2. Instantiate an object SCR1 of FB1 in PRG2 (PRG), and create variable var3.



3. Call METH under FB1, as shown in the figure below.



10.1.2.1.4 Adding an Action to a Program

Actions are added and used under programs and function blocks. They are generally used for operations such as assigning values to variables or performing calculations in PRG or FB, and are called where needed in the program. Actions require no inputs or outputs, do not have their own variable table, and cannot create variables that belong to themselves. They must use variables from PRG or FB.

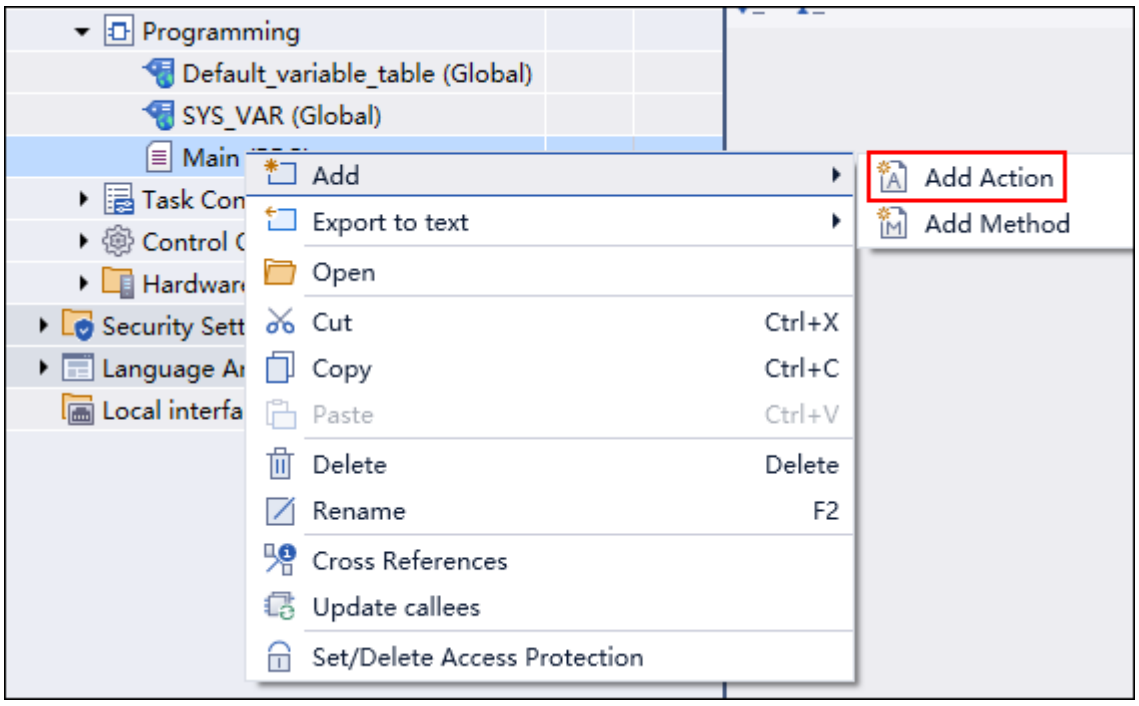
Action Creation

Prerequisites

A program (PRG) or function block (FB) has been created.

Procedure

1. Right-click an FB or PRG under "Programming" in the project tree, and select "Add > Add action", as shown in the figure below.



2. In the "Create action" window, set the name and IEC language of the action.

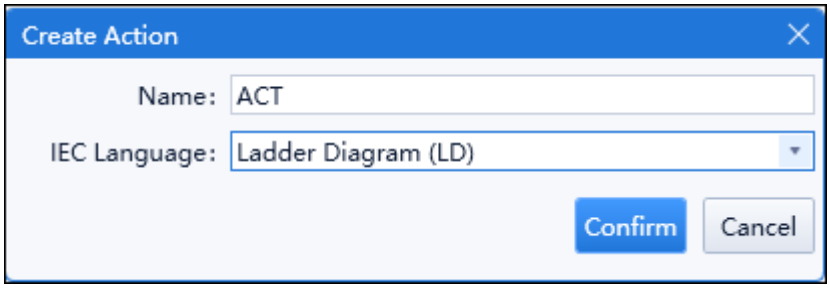
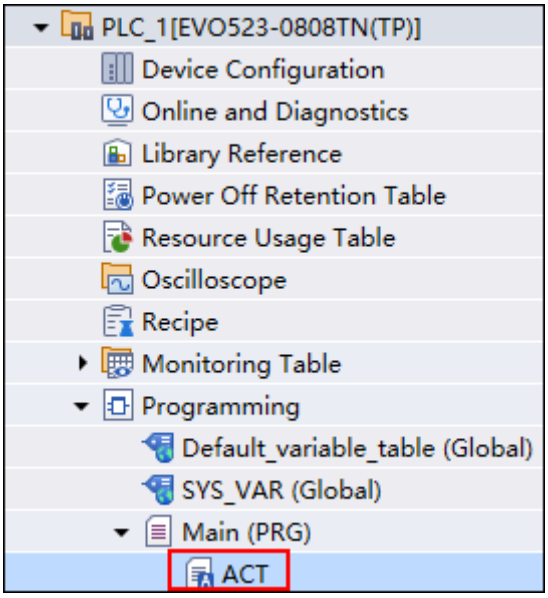


Table10-1-2-1-4-1 Action creation properties

No.	Item	Description
①	Name	Indicates the action name. It must comply with the IEC61131-3 standard and is case-insensitive, with a length ≤ 128 bytes.
②	IEC Language	Supports LD and ST programming languages.

Result

An action is added under a program.



Action Usage

- When the parent object is a program, the call format is "PRG name.ACT name()".
- When the parent object is a function block, the call format is "FB instance name.ACT name()".

```
// Call the action ACT in PRG

Main.ACT( );

// Call the action ACT in FB1

// fb1_1 is an instance of FB1

fb1_1.ACT( );
```

10.1.2.1.5 Calling a User Program

Before a user program is downloaded to the controller, it needs to be called first; otherwise, the program will not be executed.

User program calls are mainly managed in "Task configuration". For details, see 10.5 Task Configuration.

NOTE

If a user program is added without being called, the program will not be checked during compilation. If the program has not changed, it will not trigger a download.

10.1.2.2 Functions

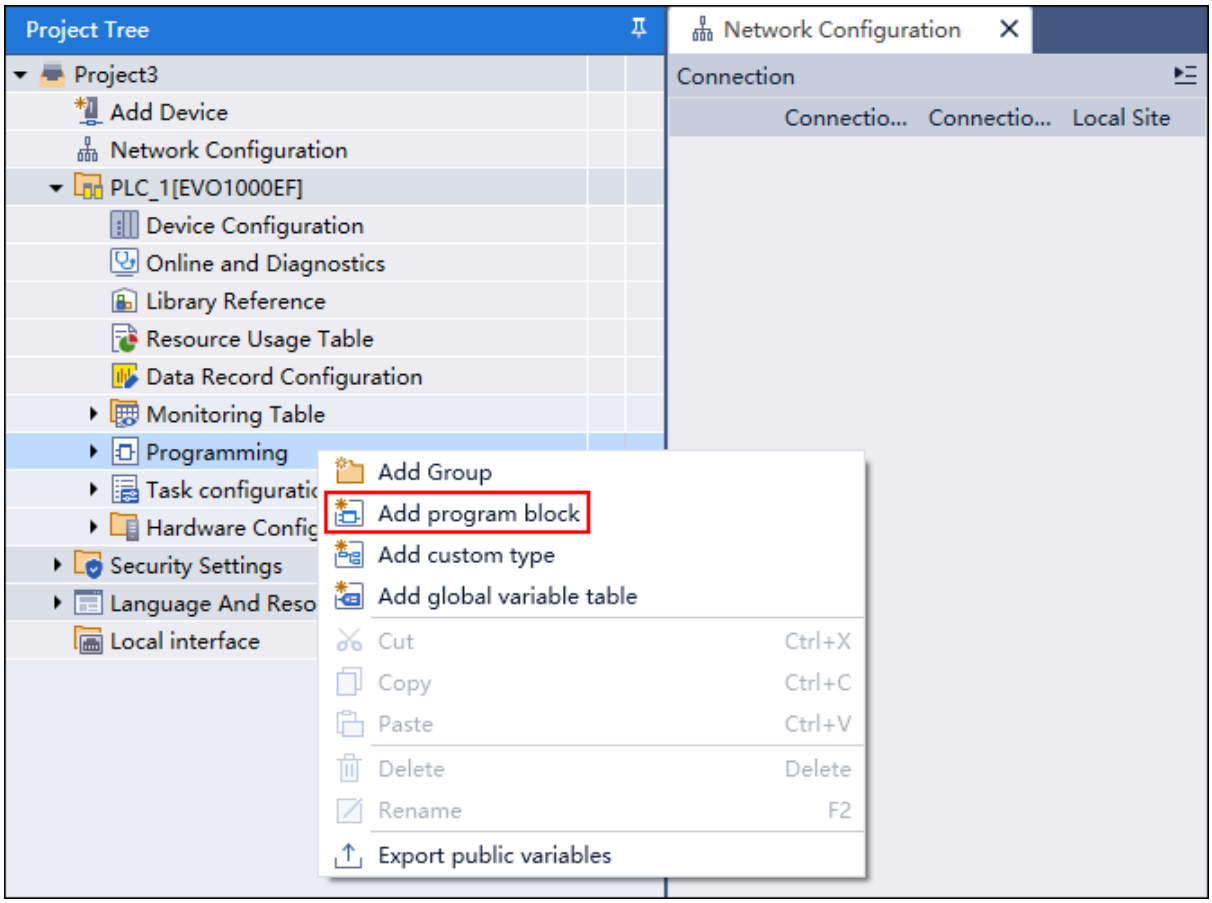
10.1.2.2.1 Adding a User Function

Prerequisites

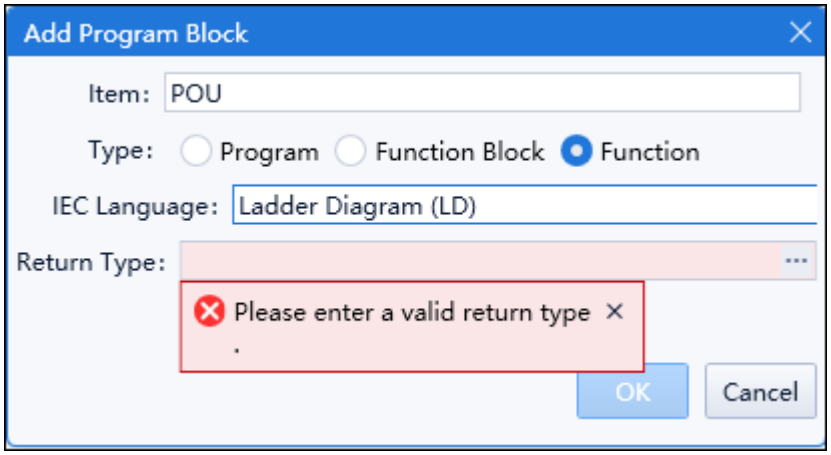
- A project has been created.
- A device has been added.

Procedure

1. Right-click "Programming > Add program block".



2. In the "Add program block" window, set the name of the function, select "Function" as the type, select the corresponding IEC language, and select the return value type as needed, as shown in the figure below.



No.	Item	Description
①	Name	Indicates the function name. It supports letters, numbers, and underscores (_), and must start with a letter or underscore. Underscores cannot be used consecutively. The program name is case-insensitive, with a maximum length of 128 bytes. For example, __POU and PO__U are invalid names, while P_O_U is considered valid.
②	Type	Indicates the program block type. To add a user function, select "Function" here.
③	IEC Language	Supports LD and ST programming languages.
④	Return type	◦ The return value type must be specified and the length of the return value must be less than or equal to 128 bytes. ◦ The return value type can be filled from the input assistant, which provides basic data types, custom types for the current project, and types from custom libraries. ◦ The return value type is case-insensitive.

3. Click "OK" to add the user function POU. The user function has a "(FUN)" suffix. When a user function is added, iFA will automatically open its editor. The editor is divided into the variable declaration area and the program editing area, which can be collapsed and expanded using and .

NOTE

When a POU is being compiled, the system will check the upper limit based on device requirements. For specific limitations, see the device specifications.

10.1.2.2.2 Modifying a User Function

A function (FUN) is defined as a program organization unit (POU). It can be assigned parameters but has no static variables. That is, when a function is called with the same input parameters, it always generates the same result as the return value. An important characteristic of functions is that they cannot use internal variables to store values, which is completely different from function blocks.

A function is a basic algorithm unit without an internal state (no runtime memory allocation). That is, given the same input parameters, calling the function will definitely produce the same result with no ambiguity. Mathematical operation functions commonly used, such as sin(x) and sqrt(x), are typical examples of functions.

A function has at least one input variable, no private data, and only one return value. Functions can be used by functions, function blocks, and programs.

Representation and declaration of functions

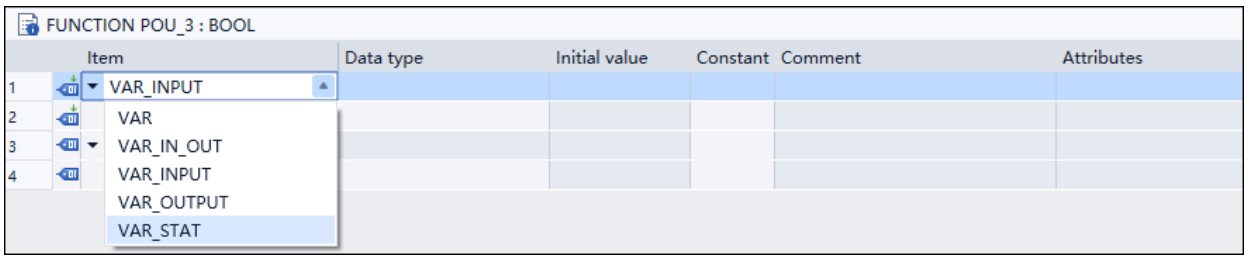
Functions are represented using the following syntax expression. The logic part can be written in either ST or LD programming languages.

```
SQL
FUNCTION <function name/return value>: <return type>
VAR_INPUT
… // Input interface variable declaration of the function*
END_VAR
VAR
… // Local variable declaration of the function
END_VAR
```

```
Plain Text
…; // Internal logic of the function
```

Variable declaration in functions

The syntax for declaring variables in FUN is shown above. The variable types that can be declared in FUN include local variables (VAR), input/output variables (VAR_IN_OUT), input variables (VAR_INPUT), output variables (VAR_OUTPUT), and static variables (VAR_STAT). All properties of the variables are configurable.



When defining a custom function, the following should be noted:

- A function can have many input variables, but it can only have one return value (output variable). There is no restriction on the data type of the return value, which can be a structure.
- An important feature of functions is that they cannot store values in internal variables, which is distinctly different from function blocks.
- Functions do not have specified memory allocation and do not need to be instantiated like function blocks.
- The actual arguments configured to VAR_INPUT by the caller can be empty, constants, variables, or function calls. In the case of a function call, The function is executed first, and its return value is passed as the actual argument.

10.1.2.3 Function Block

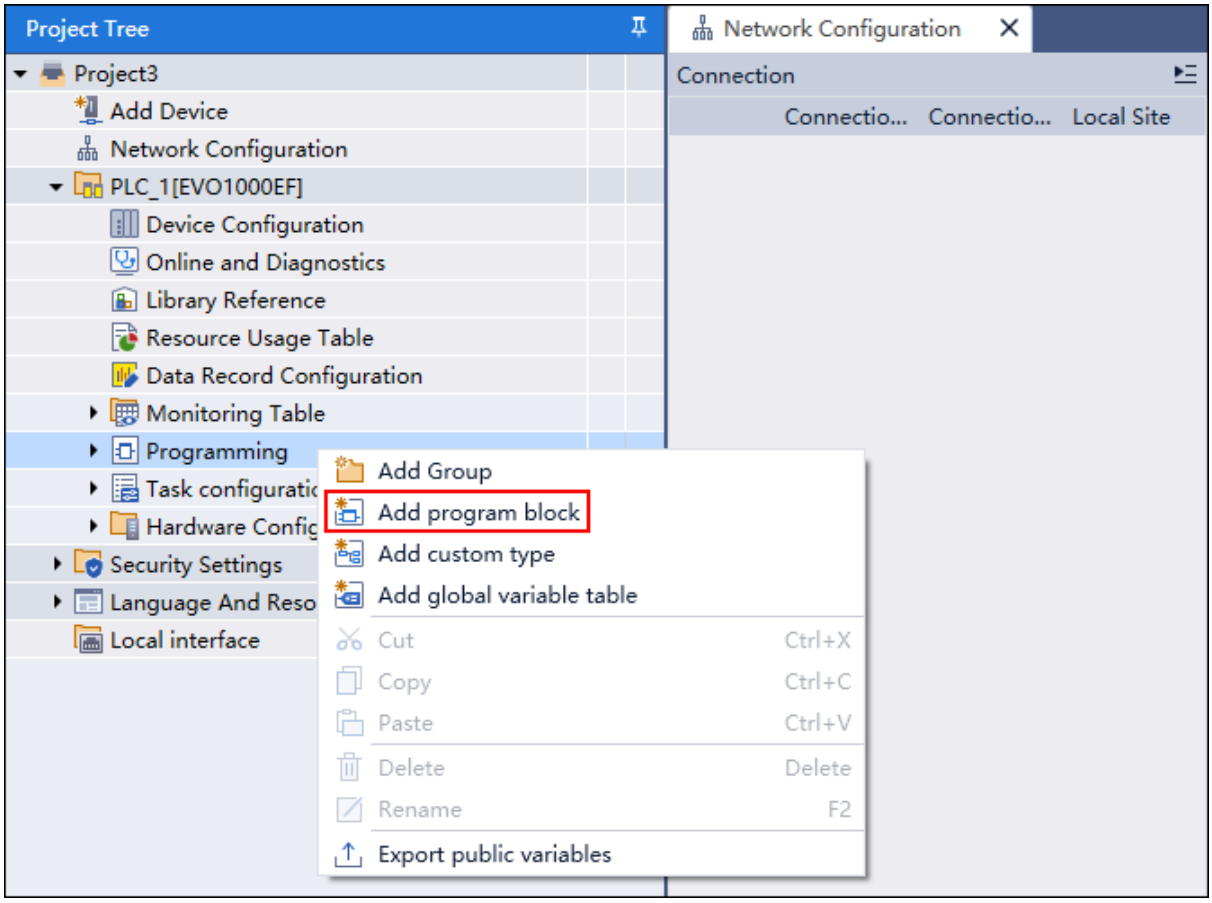
10.1.2.3.1 Adding a User Function Block

Prerequisites

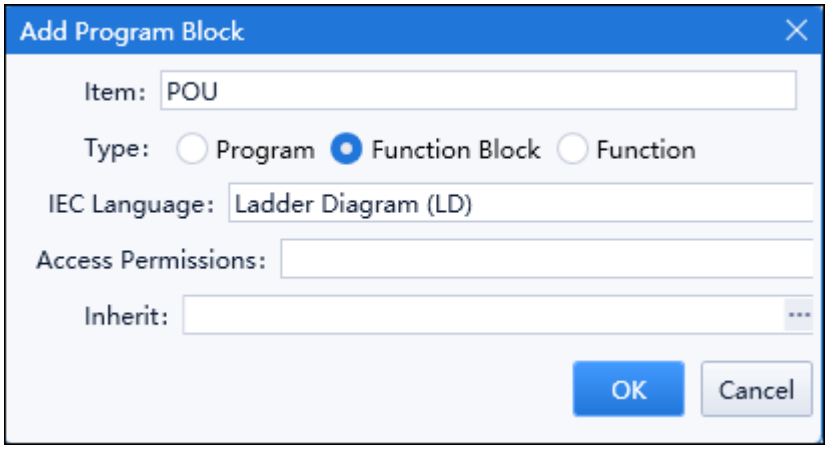
- A project has been created.
- A device has been added.

Procedure

1. Right-click "Programming > Add program block".



2. In the "Add program block" window, set the name of the function block and select "Function block" as the type. Select the corresponding IEC language, access permissions, and inheritance relationship, as shown in the figure below.



No.	Item	Description
①	Name	Indicates the name of the function block. It supports letters, numbers, and underscores (_), and must start with a letter or underscore. Underscores cannot be used consecutively. The program name is case-insensitive, with a maximum length of 128 bytes. For example, __POU and PO__U are invalid names, while P_O_U is considered valid.
②	Type	Indicates the program block type. To add a user function, select "Function block" here.
③	IEC Language	Supports LD and ST programming languages.
④	Access Permissions	Options include PUBLIC, INTERNAL, and FINAL.
⑤	Inheritance	You can select the function block being inherited using the input assistant.

3. Click "OK" to add the user function block POU. The user function block has a "(FB)" suffix. When a user function block is added, iFA will automatically open its editor. The editor is divided into the variable declaration area and the program editing area, which can be collapsed and expanded using and .

NOTE

When a POU is being compiled, the system will check the upper limit based on device requirements. For specific limitations, see the device specifications.

10.1.2.3.2 Adding a Method to a Function Block

For details about how to add a method to a function block, see "10.1.2.1.3 Adding a Method to a User Program".

10.1.2.3.3 Adding an Action to a Function Block

For details about how to add an action to a function block, see 10.1.2.1.4 Adding an Action to a User Program.

10.1.2.3.4 Adding a Property to a Function Block

Properties are added and used under function blocks. They provide two accessor methods: Get and Set, and allow read/write access to the instance data associated with the property.

Get: Read access, used to read properties in FB.

Set: Write access, used to write properties in FB.

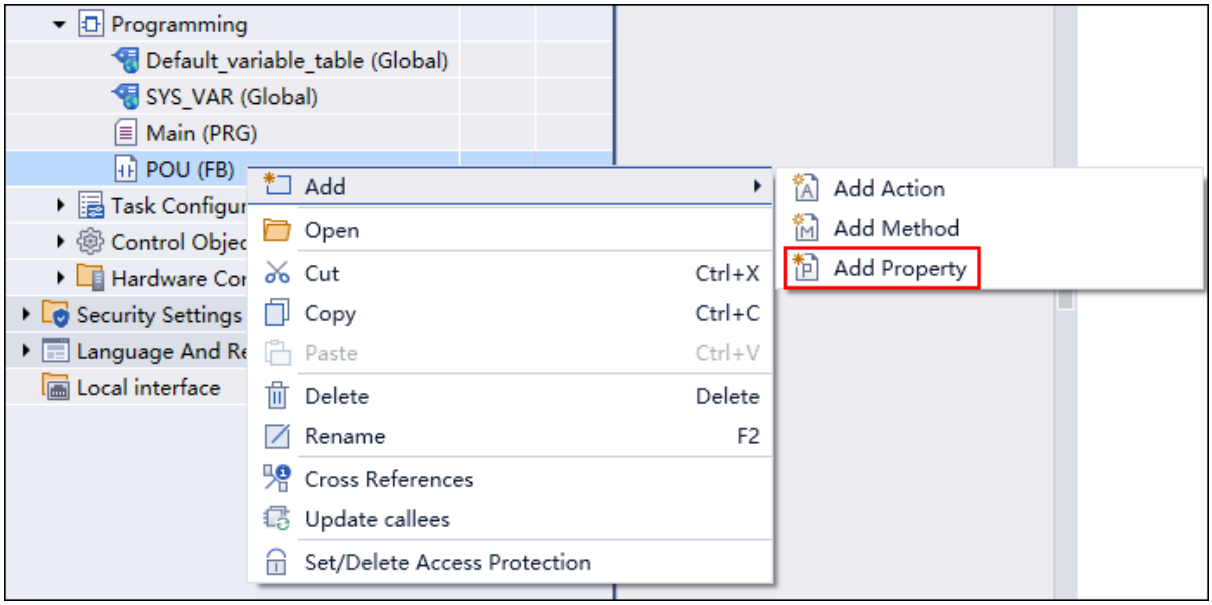
Property creation

Prerequisites

A function block has been created.

Procedure

1. Right-click an FB under "Programming" in the project tree, and select "Add > Add property", as shown in the figure below.



2. In the "Create property" window, set the property name, return value type, IEC language, and access qualifier.

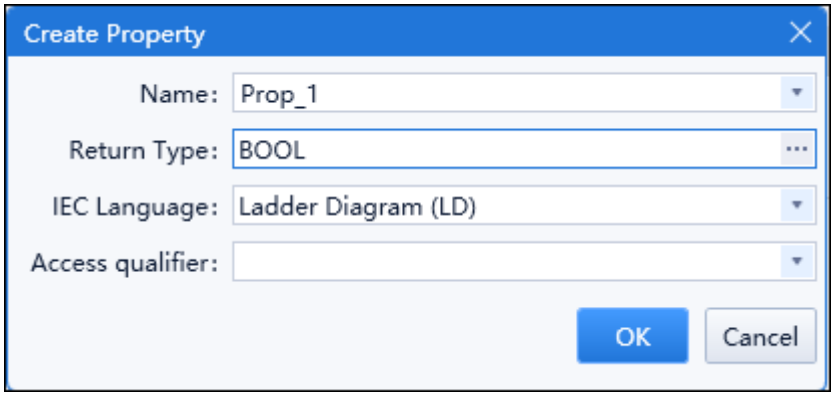


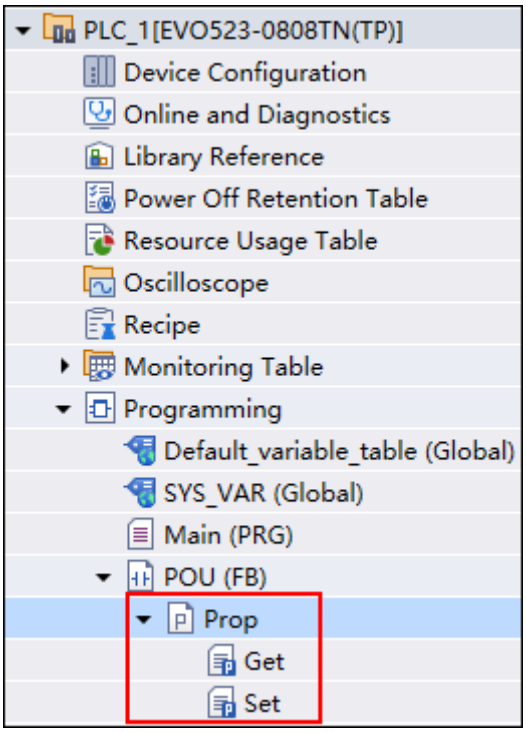
Table10-1-2-3-4-1 Property creation items

No.	Item	Description
①	Name	Indicates the property name. It must comply with the IEC61131-3 standard and is case-insensitive, with a length ≤ 128 bytes.
②	Return type	• The return value type must be specified and the length of the return value must be less than or equal to 128 bytes. • The return value type can be filled from the input assistant, which provides basic data types, custom types for the current project, and types from custom libraries. • The return value type is case-insensitive.
③	IEC Language	Supports LD and ST programming languages.

No.	Item	Description
④	Access Qualifier	Options include PUBLIC, PROTECTED, PRIVATE, and INTERNAL.

Result

A property is added under a function block. Accessors Get and Set are automatically generated.



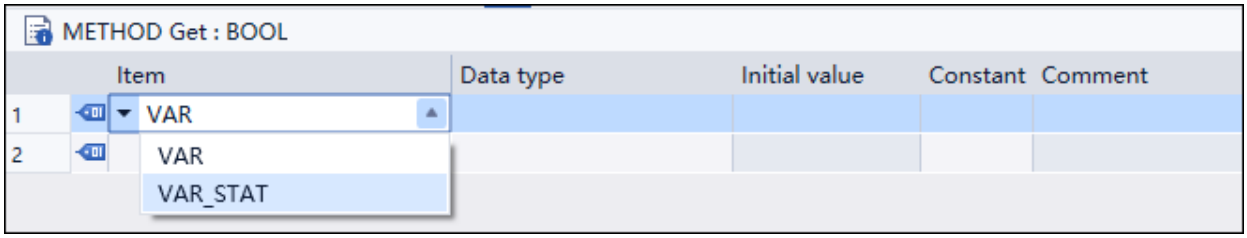
NOTE

- Accessors Get and Set can be manually deleted by users.
- After deleting an existing Get or Set accessor, it can be recreated under the "Prop" node, as follows:
  1. Select the "Prop" node.
  2. Select "Add object" > "Get accessor"/"Set accessor".
  3. In the pop-up "Create property" window, set the IEC language and access qualifier for the property.
- Each property supports only one Get accessor and one Set accessor.
- Modifying the return value type of a Get or Set accessor in the variable declaration editor (in text view) is invalid; the system still uses the initial declaration type of the property.

Property usage

Variable declaration in properties

The variable types supported by properties under a function block include local variables (VAR) and static variables (VAR_STAT). The data type, address, initial value, constant, and comment of the variables can be configured.



Property call

When the parent object is a function block (FB), the call format is "FB instance name.Prop name".

```
// Call the property Prop in FB
// FB_1 is an instance of FB
FB_1.Prop;
```

NOTE

The Get accessor is used to read properties in FB, and the Set accessor is used to write properties in FB.

10.1.2.3.5 Modifying a User Function Block

A function block converts frequently used program segments into a reusable component. It can be called by any programming language in the program and reused repeatedly, improving development efficiency and reducing errors, thereby enhancing program quality.

A function block is a program organization unit that can produce one or more values during execution. It retains its own special internal variables, and the controller's target execution system must allocate memory for the internal state variables of the function block. These internal variables form its own state characteristics.

The execution logic of a function block forms its own object behavior characteristics. Therefore, for the same input variable values, calculation results may vary due to different internal state variables.

Representation and declaration of function blocks

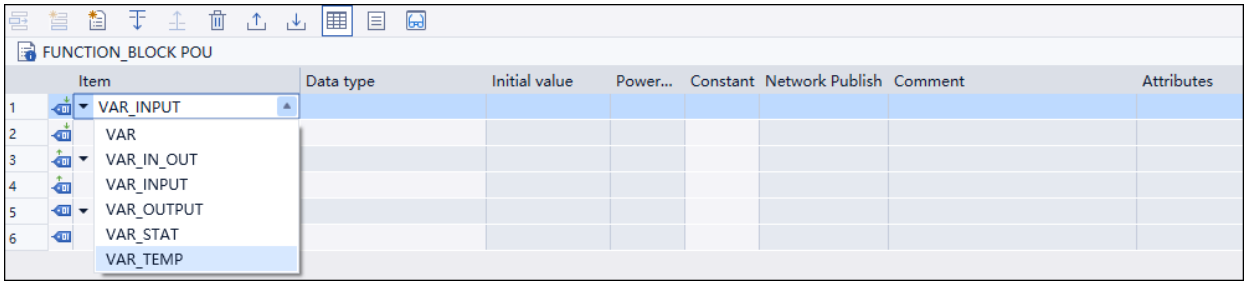
Function blocks are represented using the following syntax expression. The logic part can be written in either ST or LD programming languages.

```
FUNCTION_BLOCK <function block name>
VAR_INPUT
… // Input interface variable declaration of the function block
END_VAR
VAR_OUTPUT
… // Output interface variable declaration of the function block
END_VAR
VAR
… // Local variable declaration of the function block
END_VAR
```

```
...; // Internal logic of the function block
```

Variable declaration in function blocks

The syntax for declaring variables in FB is as shown above. The variable types that can be declared in FB include local variables (VAR), input/output variables (VAR_IN_OUT), input variables (VAR_INPUT), output variables (VAR_OUTPUT), static variables (VAR_STAT), and temporary variables (VAR_TEMP). All properties of the variables are configurable.



NOTE

The EVO500 series PLC V1.2 version does not support setting the "Network public" property of a function block.

10.1.2.3.6 Using a User Function Block

Instantiation of function blocks

According to the IEC61131-3 standard, the type of function block is an abstract definition of a structure, not a real data entity. If it is not defined and instantiated, it cannot be called and executed by the program. Therefore, function blocks need to be instantiated before they can be used. After instantiation, a function block is a completely encapsulated and independent structured variable that owns private data and can perform specific functions according to predefined logic.

The instantiation of a function block is like defining a variable. When the program needs to call this function block, it only needs to declare a variable of the FB type. The program reads and writes to the function block through this variable during execution. Instance types can have the same name, but instance names must not be the same within the same POU.

NOTE

- FBs cannot be declared as temporary variables, or local variables of FUN/Meth; otherwise, a compilation fault will be reported.
- FB pointers and FB references can be declared as temporary variables, or local variables of FUN/Meth.
- Regarding FB nesting, a compilation fault will only be reported if the nested FB object is declared as a temporary variable, or a local variable of FUN/Meth. The parent object will not trigger a fault.
- During function block instantiation, the "Network public" property (options include "Public" and "Private") is set to "Private" by default. If the parent node is set to "Private", all child nodes will also be "Private"; if the parent node is set to "Public", the child nodes will inherit the public property configured for the function block itself.
- During function block instantiation for the EVO500 series PLC V1.2 version, the "Network public" property (options include "Private", "Read/Write", "Read-Only", and "Write-Only") is set to "Private" by default. The child node properties inherit from the parent node and cannot be set separately. If the device version is downgraded from V1.3 to V1.2, a fault of public variable property compilation will be reported.

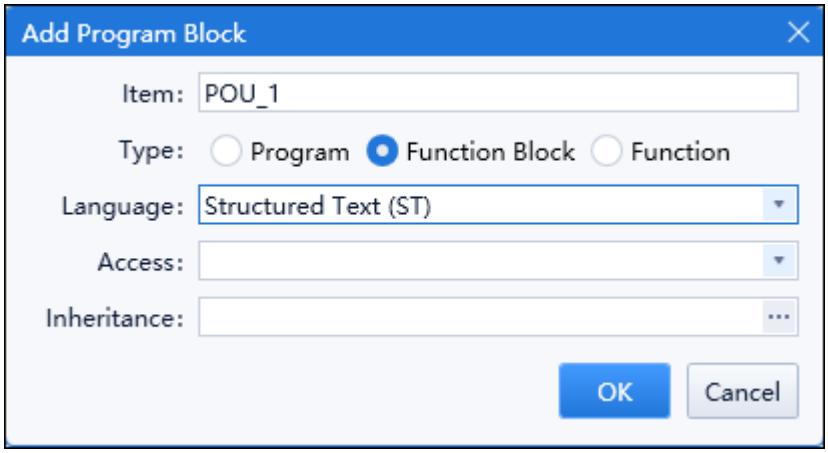
Extension of function blocks

iFA supports object-oriented programming. Function blocks can derive child function blocks. Child function blocks inherit the properties of parent function blocks and can have their own additional characteristics. They can be considered an extension of the parent function block.

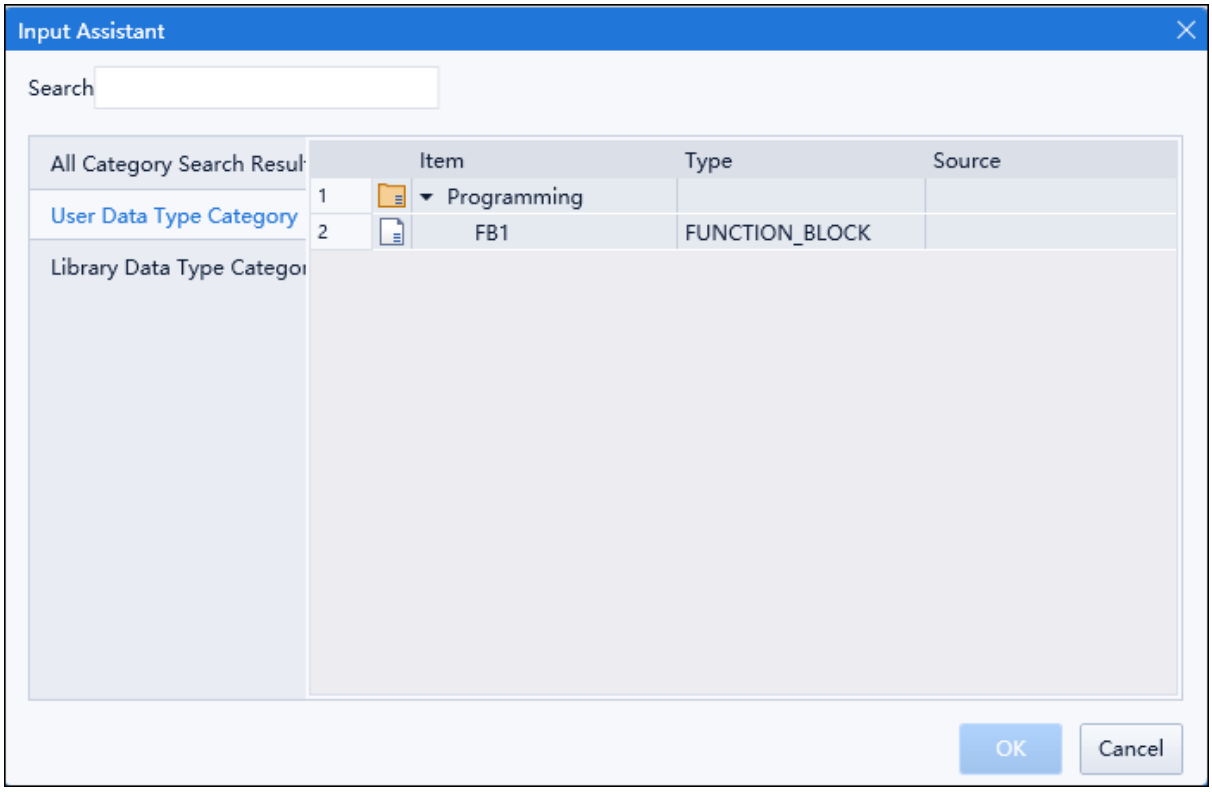
When creating a function block, add the keyword "EXTENDS" to enable the extension feature. This feature can also be enabled through the "Inheritance" configuration when adding a function block.

Enabling extension through inheritance

1. Right-click "Programming > Add program block".
2. In the "Add program block" window, set the name of the function block and check the type as "Function block", as shown in the figure below.

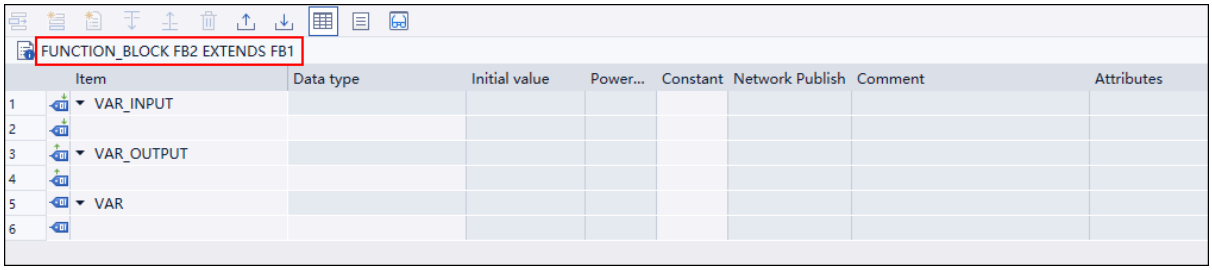


3. Click the "..." button after "Inheritance" to enter the "Input assistant" window and select the function block to be inherited, as shown in the figure below.

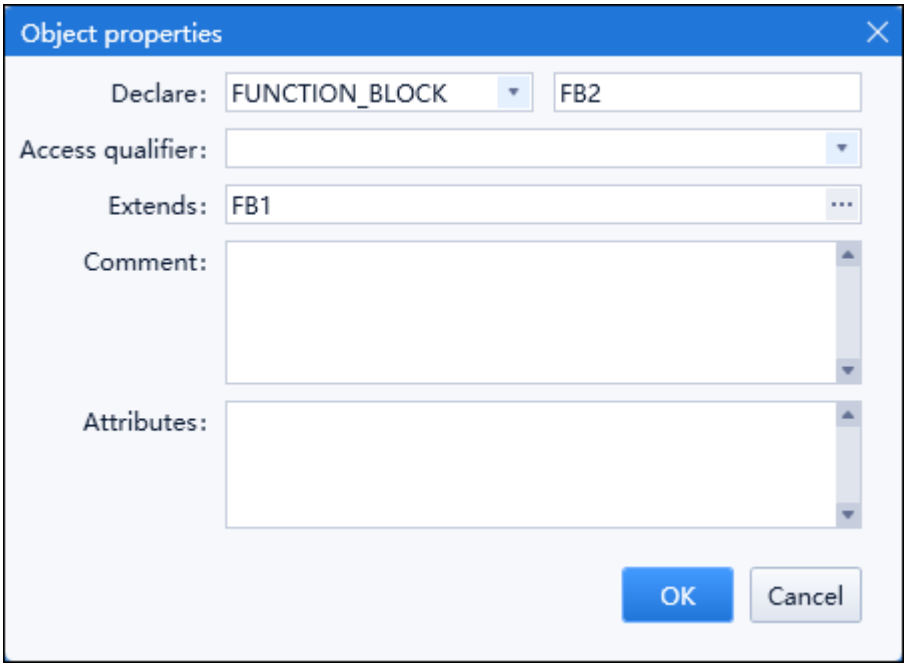


Enabling extension through variable declaration editor (in table view)

1. In the table view of the variable declaration editor, click the name of the function block and the "Object property" window pops up.



2. Add the function block being inherited, as shown below.

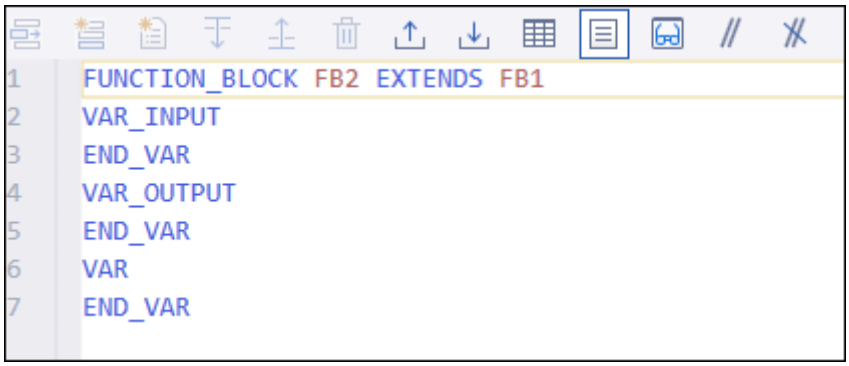


Enabling extension through variable declaration editor (in text view)

Switch the variable declaration editor to text view, and add the inheritance property for the function block. The format for declaring an extended function block is as follows.

```
FUNCTION_BLOCK <child function block name> EXTENDS <parent function block name>
```

Result



In the above example, function block FB2 is extended from FB1. FB2 includes all variables and methods of FB1 and can replace FB1 wherever FB1 is used.

Methods in FB1 can be overwritten in FB2. That is, users can redeclare a method within FB2 that already exists in FB1, with the same name, inputs, and outputs.

Variables in FB2 and FB1 must not have the same name to avoid compilation faults.

The internal algorithms of FB2 support using variables and methods from FB1 with the keyword "SUPER" ("SUPER^." + "variable name or method name in the parent function block"). The "SUPER" keyword is case-insensitive.

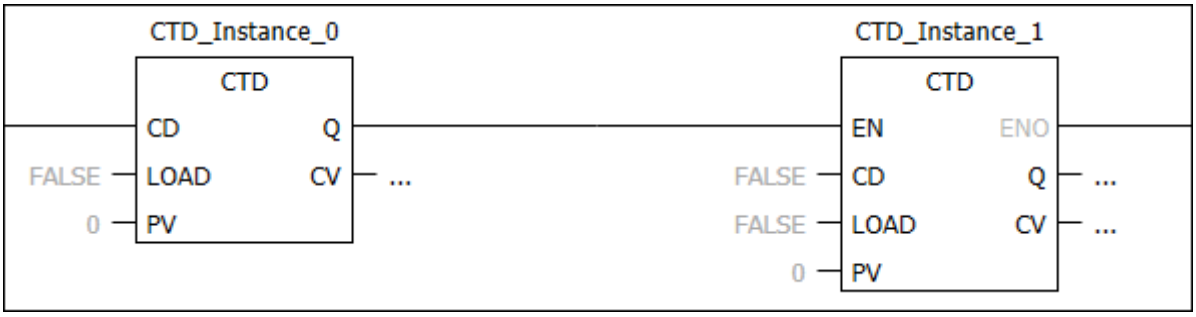
EN and ENO

EN and ENO are the auxiliary properties of function blocks. They are only valid in ladder diagram and function block diagram programming languages. EN and ENO respectively control whether to enable the input and output of function blocks. They can be enabled or disabled by all functions.

The application principles for enabling inputs and outputs are as follows:

- When a function is called and the EN value is set to False, the operations defined by the function block will not be executed by the program, and the ENO value will also be False.
- When a function is called and the EN value is set to True, the operations defined by the function block will be executed and the ENO value will also be True.
- EN and ENO can be enabled or disabled as needed.

The following figure shows a regular function block and a function block with EN/ENO.



> **NOTE**
>
> EN/ENO is specific to the LD programming language. User variables cannot be declared with EN/ENO, otherwise a fault will occur.

Access partitioning

Function blocks achieve their access partitioning through access qualifiers, which include PUBLIC, INTERNAL, and FINAL. If no access qualifier is set, it defaults to PUBLIC.

There are two methods to set the access qualifier for function blocks:

- Method 1: Set in the "Add program block" window.
- Method 2: Set in the variable declaration editor (in table view).

Descriptions of access modifiers

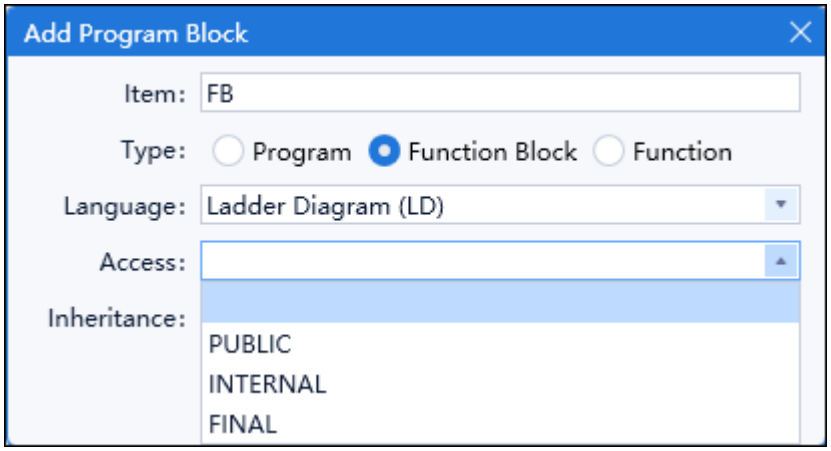
Access Qualifier	Description
PUBLIC	Public access: Variables and methods added under FB can be used across POU and across libraries.
INTERNAL	Access permissions within the namespace: Variables and methods added under FB can be used in this project or this library project.
FINAL	The FB cannot be inherited.

Prerequisites

- Prerequisite for Method 1: Set access modifiers when creating the function block.
- Prerequisite for Method 2: The function block has already been created.

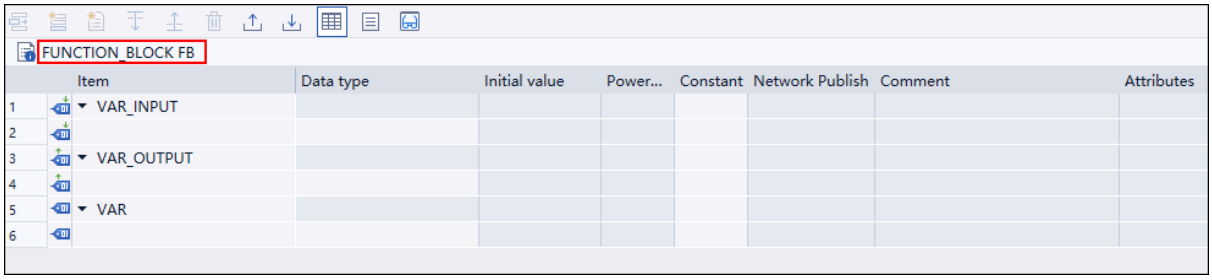
Setting access qualifiers in the "Add program block" window

1. Right-click "Programming" in the project tree and select "Add program block".
2. Select "Function block" as the type.
3. In the "Access" drop-down menu, set the access qualifier as shown in the figure below.

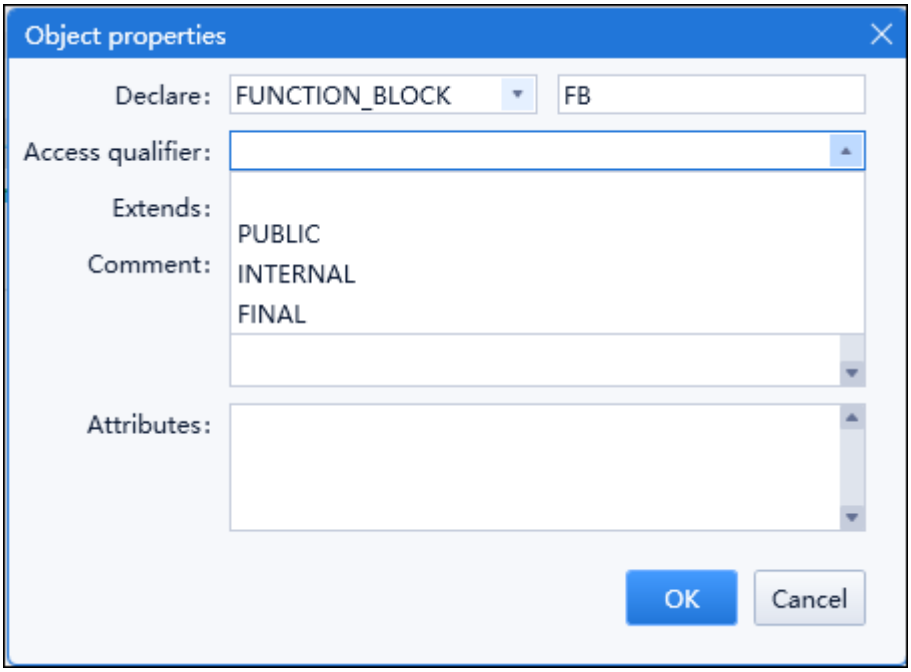


Setting access qualifiers in the variable declaration editor (in table view)

1. Click the name of the function block.

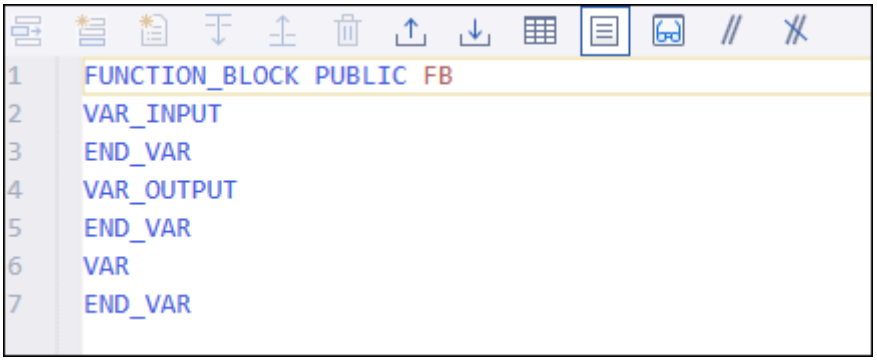


2. Set or modify the access qualifier in the "Object property" window, as shown in the figure below.



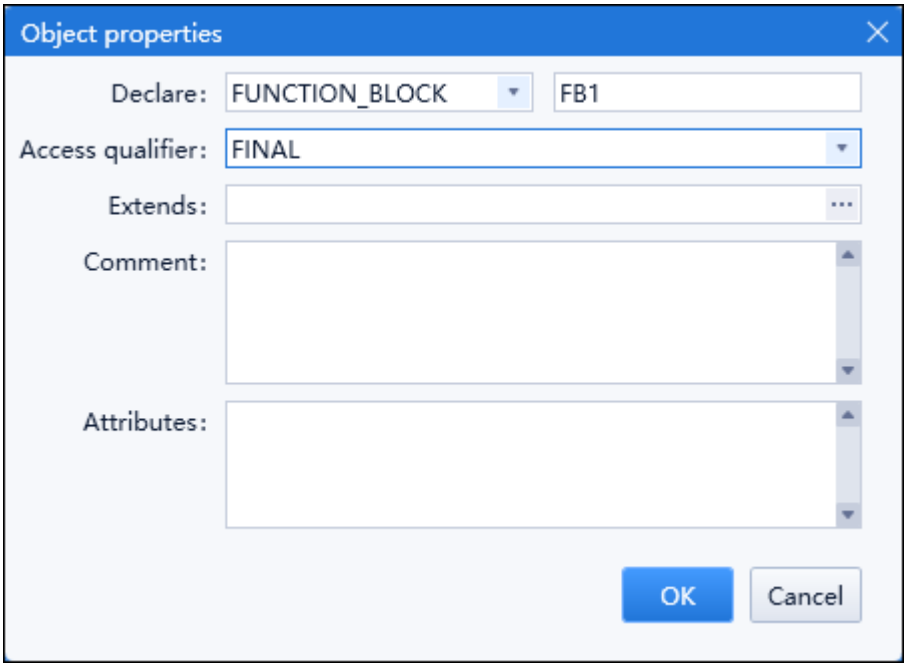
Setting access qualifiers in the variable declaration editor (in text view)

Directly add the access qualifier "PUBLIC", "INTERNAL", or "FINAL" between "FUNCTION_BLOCK" and the name "FB", as shown in the figure below.

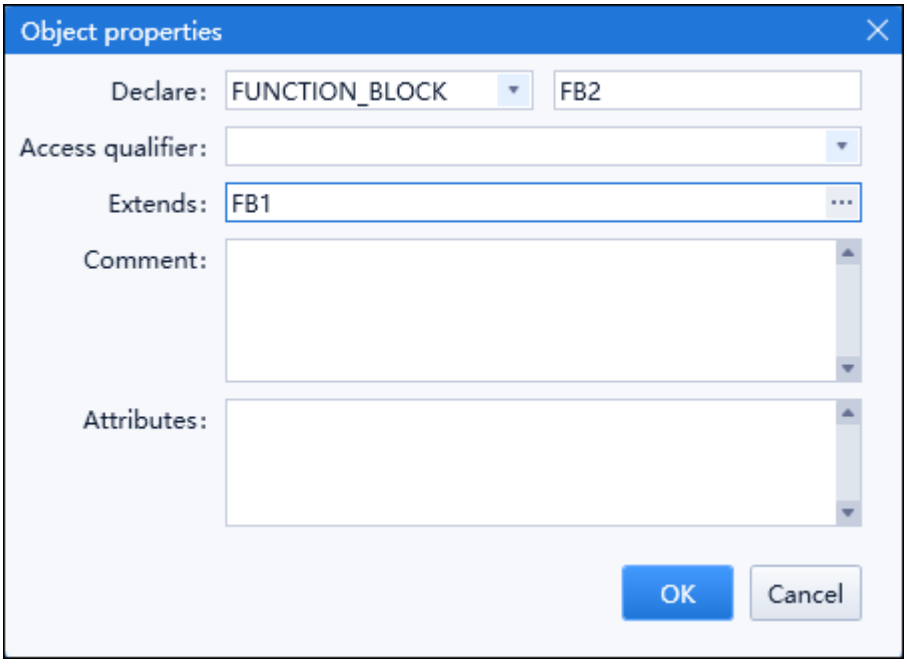


Example of adding the "FINAL" access qualifier

1. Add a function block named FB1 and set the access qualifier to "FINAL".

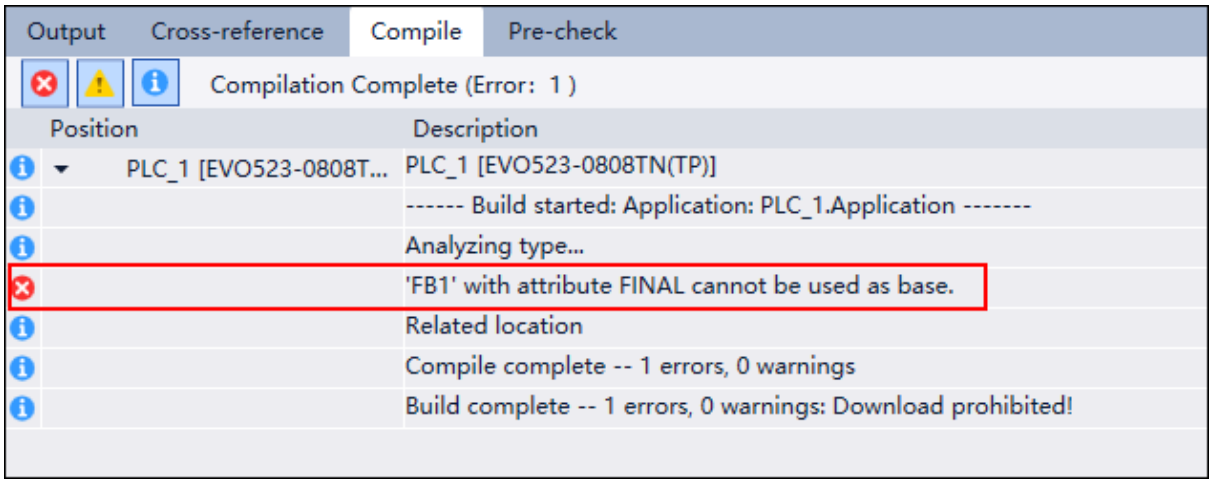


2. Add a function block named FB2 and set it to inherit from FB1.



3. Instantiate FB2.

4. Click the "Compile" tab at the bottom of the interface. A compilation fault is reported, indicating that FB1 with property FINAL cannot be used as a parent function block, as shown in the figure below.



Referencing methods and variables of the current function block

Keyword THIS is used to reference methods and variables of the current function block. It can be used inside functions and function blocks.

Syntax for using keyword THIS

"THIS^." + "variable name or method name" (Keyword THIS is case-insensitive)

In most cases, "THIS^." can be omitted. However, when a local variable has the same name as a member variable of an instance variable, the local variable must be prefixed with **"THIS^."**.

Prerequisites

Functions (FUN) and function blocks (FB) have been created.

Example of using keyword THIS

1. An INT-type variable "age" is added to the function block FB2.
2. An INT-type variable "age" and an instance of FB2 type (named "fb2_Instance_0") are added to the function block FB1.
3. To assign the value of the "age" variable of instance fb2_Instance_0 to the "age" variable of FB1, use the following code: "THIS^.age:=fb2_Instance_0.age;".

Browsing instance objects

When a user-defined function block contains at least two instances within the current project, double-clicking the function block node during online operation will pop up a window containing all instance objects.

The function block instance box lists the instantiation information of POU_fb2 in the current project. POUld_1.fb2_Instance0 indicates that the instance name of function block POU_fb2 in program POUld_1 is fb2_Instance0. Select any instance, and the real-time internal calculation results of the corresponding instance will be displayed.

10.1.2.4 Program Block Callee Update

Overview

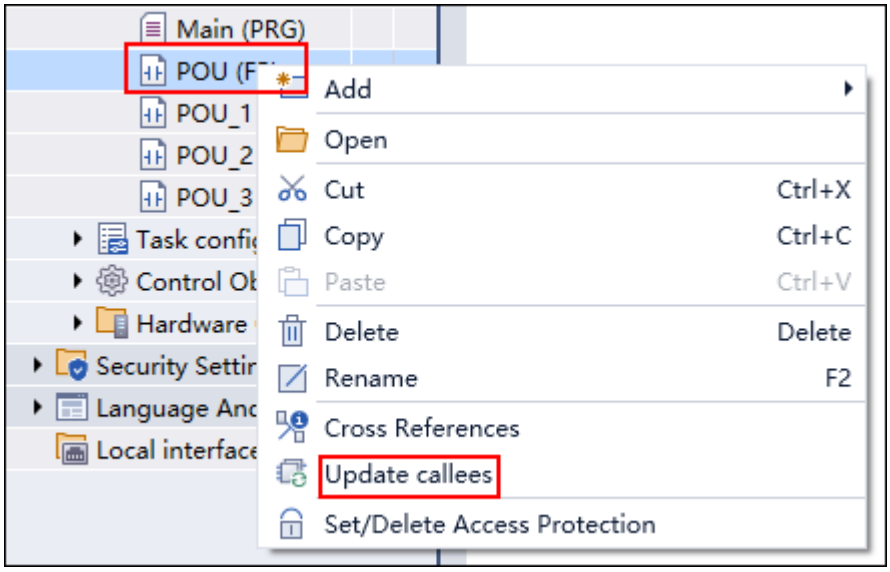
The "Update callee" function is used to update the parameter list at the call positions of specified FB instances, FCs, PRGs, and Meths in the current project.

Prerequisites

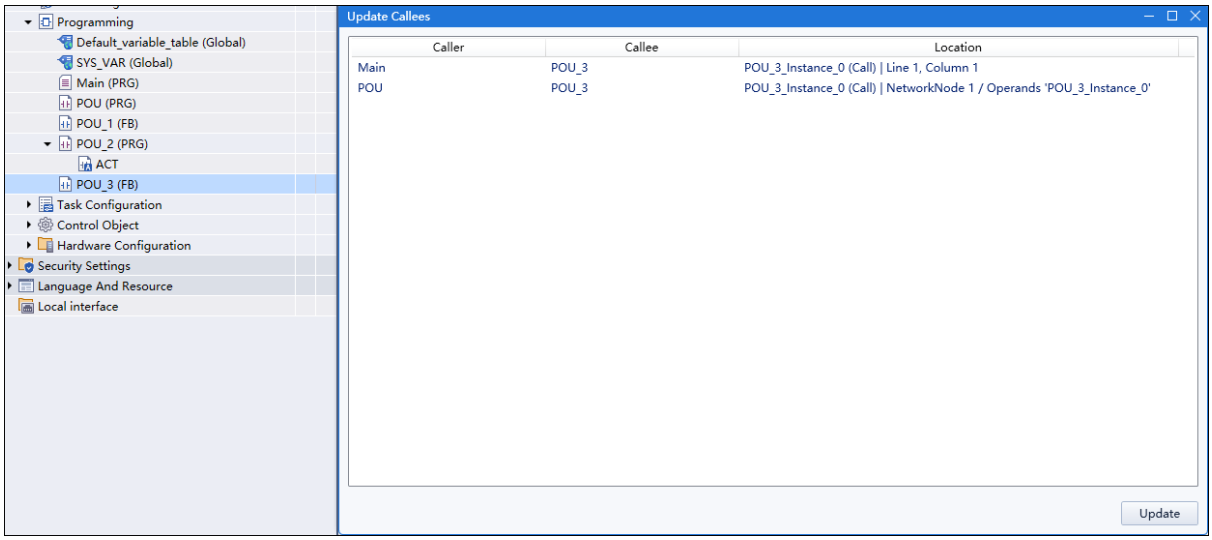
- An FB has been created, and its instance has been called.
- FB instances, FCs, PRGs, and Meths are called in other POUs using the ST and LD programming languages.

Procedure

1. Right-click the FB, FC, PRG, or Meth object in the device tree.
2. In the pop-up menu, select "Update callee"



3. In the pop-up window, the information about the caller, callee, and calling location is displayed.



Parameter	Description
Caller	Indicates which objects have called the specified FB, PRG, FC, or Meth.
Callee	Indicates the name of the called object, including the target object right-clicked, as well as its child objects and grandchild objects (the parent and child objects are separated by a colon).
Position	Indicates where the object is called.

4. Click "Update" to update the parameter list at the call positions of the instruction.

Result

The pins of the POU (FB) call positions are updated, as shown in the following figures.

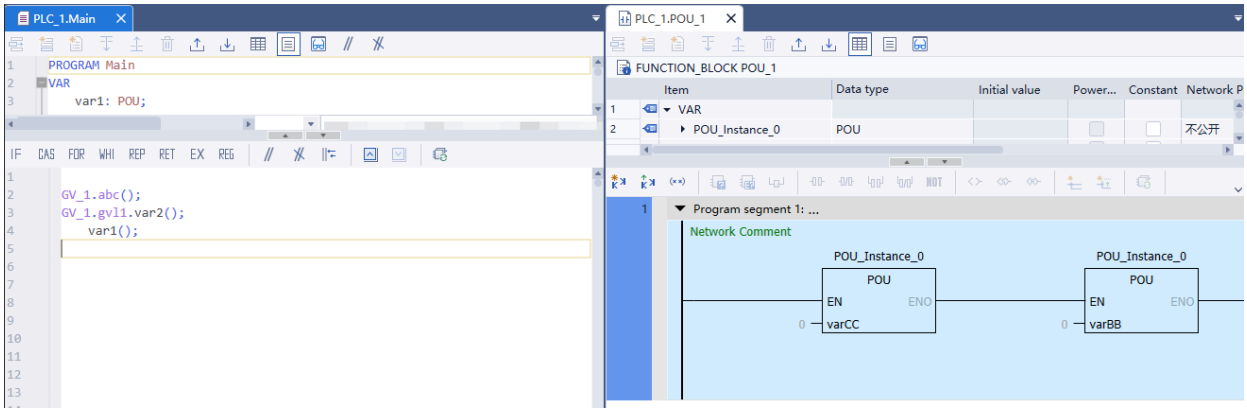


Figure 10-1-2-4-1 Before update

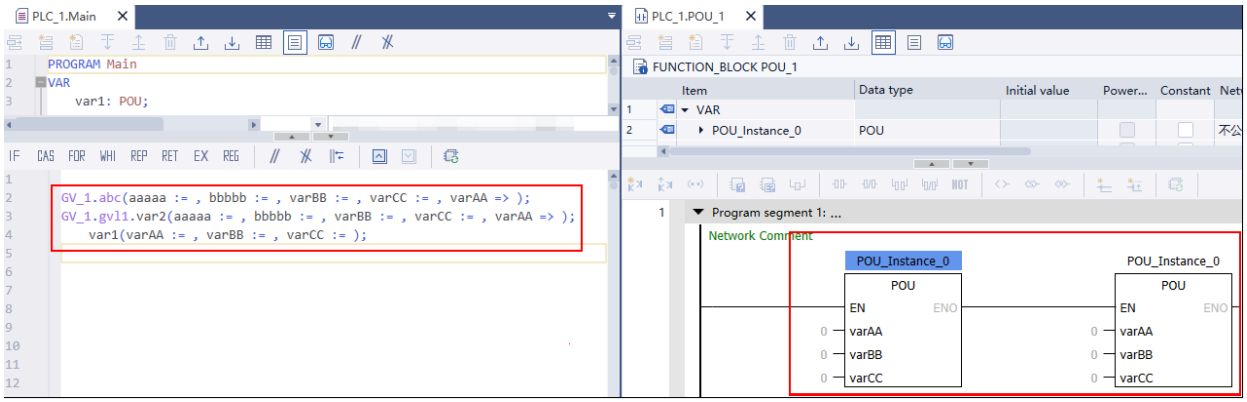


Figure 10-1-2-4-2 After update

10.1.2.5 Program Block Access Permissions

Overview

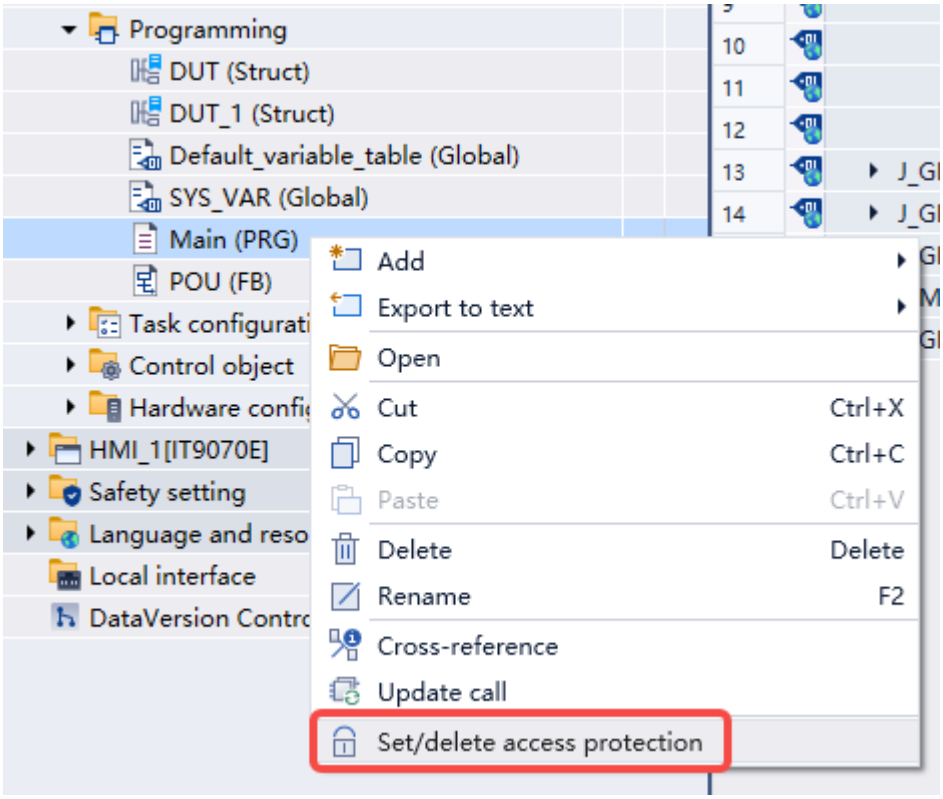
The iFA platform defaults to no operation restrictions on program blocks (PRG/FB/FC), allowing direct viewing and modification of content.

Configurable permissions for program blocks include: write protection and read/write protection. Only one of the write protection and read/write protection can be enabled at the same time.

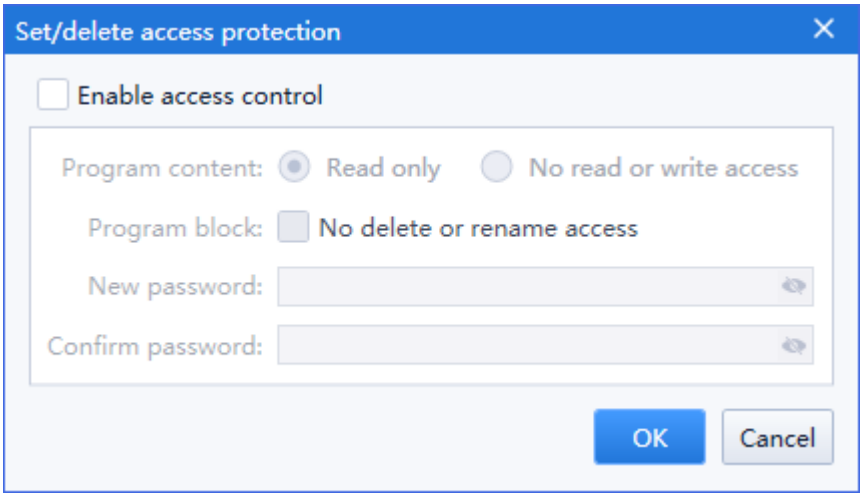
- Write protection: Without a password, the block is view-only. Editing is allowed only after entering the correct password.
- Read/write protection: Without a password, the block cannot be accessed at all. Editing is allowed only after entering the correct password.

Permission configuration

1. Right-click the PRG/FB/FC node under "Programming" in the project tree and select "Set/delete access protection".



2. In the opened window, set the access permissions for the program block and click "OK" to apply.

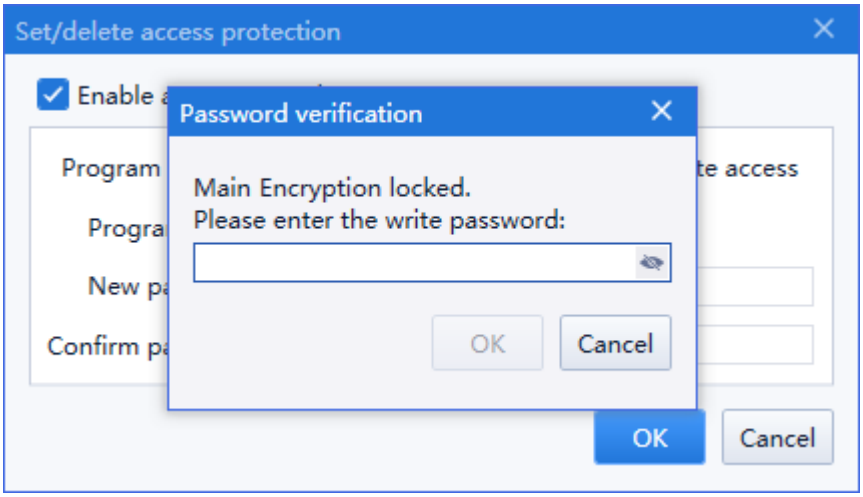


Parameter	Description
Enable access control	By default, it is unchecked, and all configuration items are grayed out. When checked, users can configure program content, block access permissions, and password.
Program content	Controls read/write operations for the program in the program editor: ◦ Read-only: Viewable but not editable. ◦ No read or write access: Cannot be viewed or edited.
Program block	Controls renaming and deletion for program blocks (PRG/FB/FC) under the "Programming" node in the project tree. Unchecked by default. When checked, deleting or renaming a program block requires a password.
Password field	Password length should be 6-8 characters, containing at least one digit, one uppercase letter, and one lowercase letter.

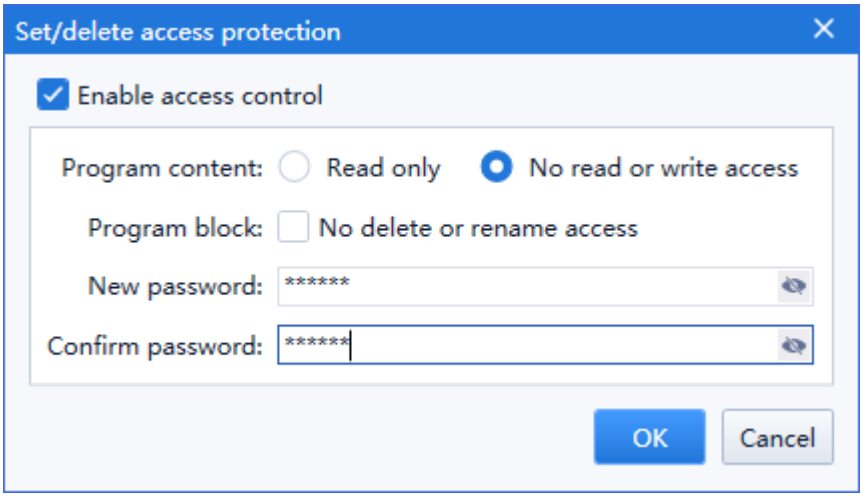
Switching permissions

- Switching from write protection to read/write protection

1. Right-click the PRG/FB/FC node under "Programming" in the project tree and select "Set/delete access protection". In the opened window, check "No read or write access". A dialog box will prompt for the write password.



2. After entering the correct password, check "No read or write access". Enter a valid "New password" and "Confirm password", and click "OK" to switch the permission.



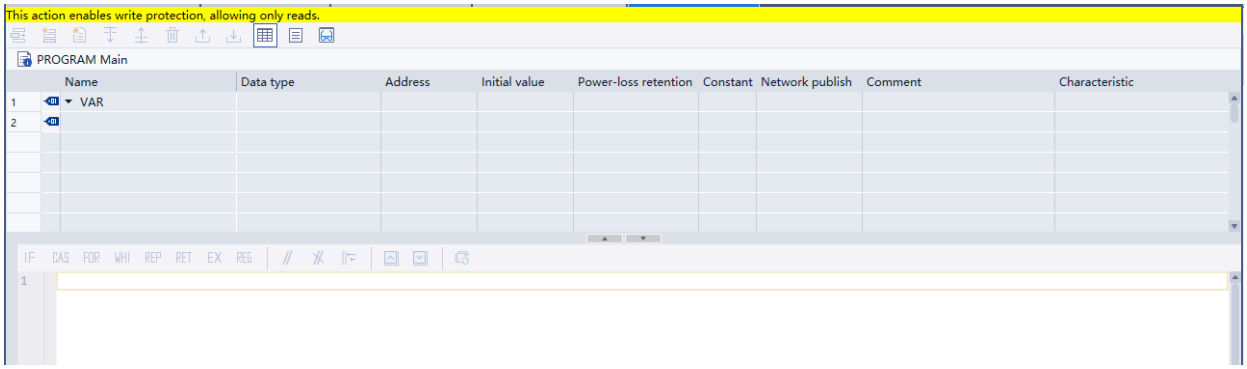
- Switching from read/write protection to write protection
  The switching method is the same as the above operation.

Canceling permissions

- Method 1: Right-click the PRG/FB/FC node under "Programming" in the project tree and select "Set/delete access protection". Uncheck "Enable", the password verification interface will pop up. Enter the correct password and click "OK" to apply.
- Method 2: Right-click the PRG/FB/FC node under "Programming" in the project tree and select "Disable/enable access protection". Enter the correct password on the password verification dialog box, and click "OK" to apply.

Opening a POU with write protection

For a POU with write protection, if no password is entered, it can only be opened in read-only mode, and the following prompt message appears.



After entering the correct password, the POU is opened in edit mode without any prompt messages.

Opening a POU with read/write protection

A POU with read/write protection must be successfully verified with a password before it can be opened. After successful verification, the user can normally edit the POU.

10.1.2.6 Program Block Cross-Reference

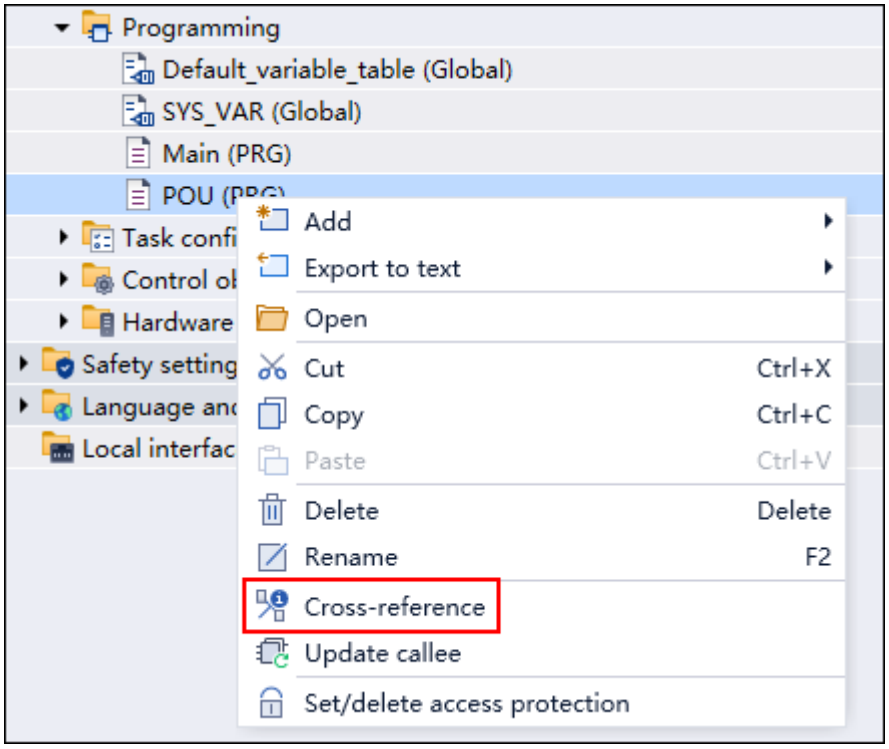
The cross-reference function of program blocks is used to view where a program block is referenced in the program. This helps users quickly locate the usage and position of the specified program block, improving commissioning and maintenance efficiency.

Prerequisites

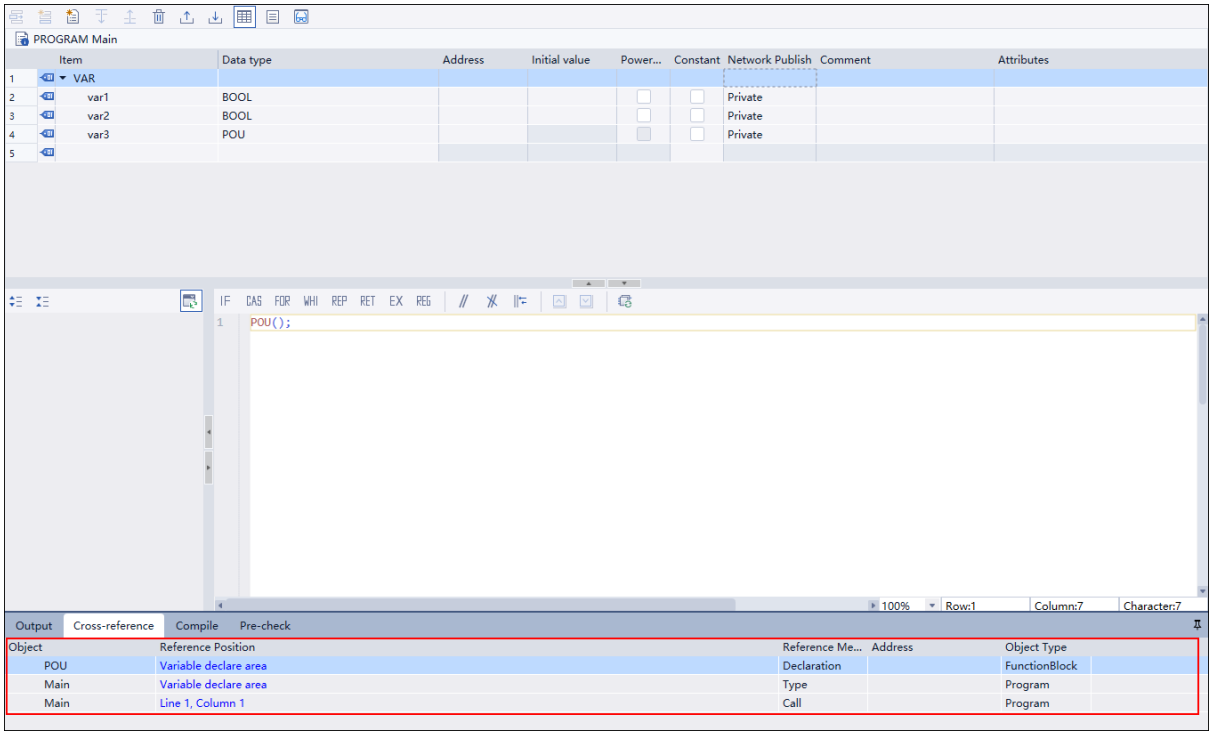
POUs have been created with cross-reference relationships in between.

Procedure

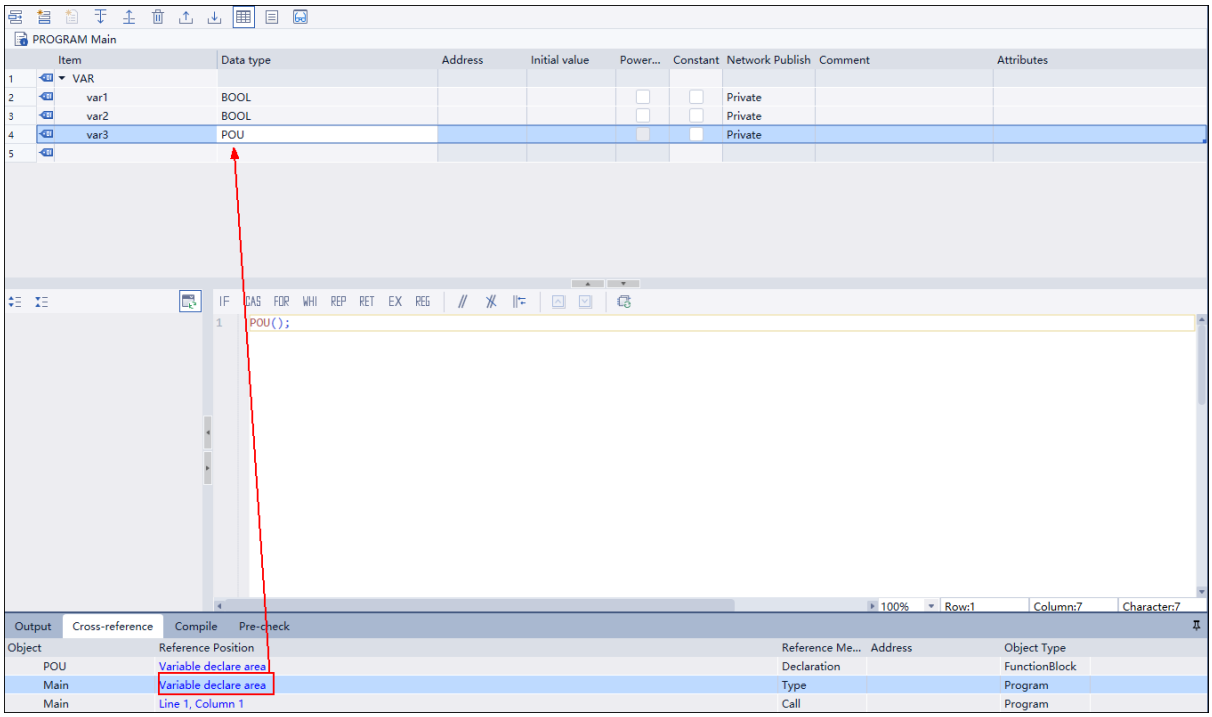
1. Right-click a POU under "Programming" in the project tree, and select "Cross-reference", as shown in the figure below.



2. The "Cross-reference" tab will be displayed at the bottom of the interface, showing the usage information of the current POU, as shown in the following figure.



3. Click "Reference position" to automatically jump to the corresponding position.



10.1.2.7 Viewing Call Relationships Between Program Blocks

The call tree is used to view the call information of a POU, expanding with two root nodes: POU caller and POU callee. The call tree includes the symbol, call position, and device information of the POU.

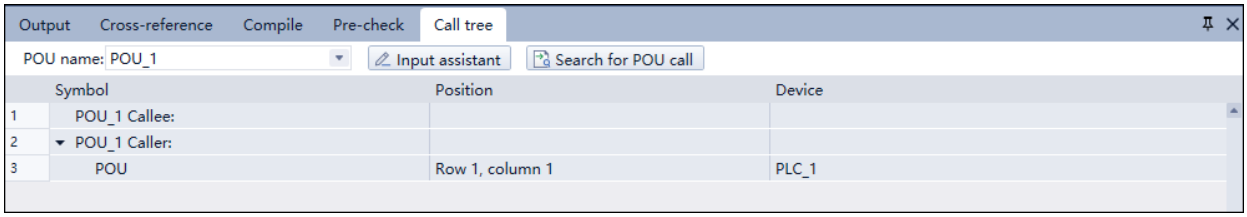
Prerequisites

POUs have been created with call relationships in between.

Procedure

1. In the software menu, select "View > Call tree" to display the call tree tab at the bottom of the interface.
2. In the "POU name" edit box, enter the name of the POU.
3. Click the "Search for POU call" button.

Result



Introduction to the call tree

Description	Description
POU name	Enter the name of the POU to view the call relationships of it.
Input assistant	Used to select a specified POU.
Search for POU call	Search for the call relationships of the specified POU.
Operator	Displays two root nodes, POU callers and POU callees. Each root node displays the contained POUs.
Position	• For the root node, the "Position" value indicates the line number where the POU is declared. • For POU callers and callees under the root node, the "Position" value indicates the line number, column number, and network number, depending on the programming language.
Device	Displays the device to which the POU belongs.
Double-click to locate	Double-click the selected POU to jump to the declaration position of the POU.

Input methods for POU call querying

- Method 1: Directly enter the name of the POU to be queried in the "POU name" edit box.
- Method 2: Select the POU to be queried from the "POU name" drop-down menu. The drop-down menu contains all records of POUs that have been selected.
- Method 3: Select the POU to be queried in the "Input assistant".

NOTE

The call tree only displays the call relationship information of the POU, and does not include variables.

10.1.3 LD Language Programming

10.1.3.1 LD Language Overview

A ladder diagram is a graphical programming language.

It features two vertical power rails on its left and right sides. The left power rail provides a power flow that passes through various contacts, functions, function blocks, coils, and other components along horizontal rungs from left to right. The right power rail serves as the endpoint of the power flow.

Each contact represents the status of a Boolean variable, each coil represents the status of an actual device, and functions or function blocks correspond to the standard libraries in IEC 61131-3 or user-created functions or function blocks.

In ladder diagram programming, although relay components such as coils and contacts are used, they are not actual physical relays but rather storage units (soft relays) within the PLC. Each soft relay corresponds to a storage unit in the PLC's memory mapped register, and its status (TRUE/FALSE) determines the logical behavior of the soft relay:

- TRUE (ON) status: When the storage unit is in TRUE status, the corresponding soft relay coil is "energized", with its normally open contacts closed and its normally closed contacts opened.
- FALSE (OFF) status: When the storage unit is in FALSE status, the soft relay coil is "de-energized", with its normally open contacts remain opened, and its normally closed contacts remain closed.

10.1.3.2 LD Creation and Management

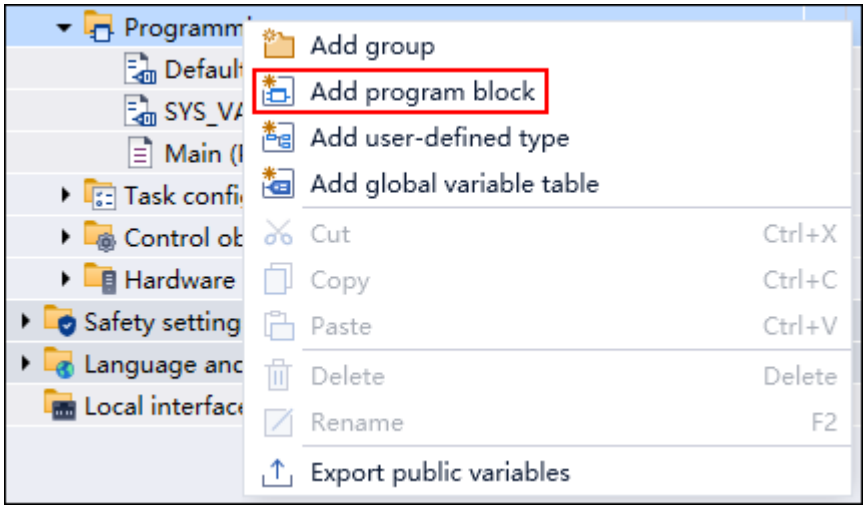
10.1.3.2.1 Creating an LD Program Block

Prerequisites

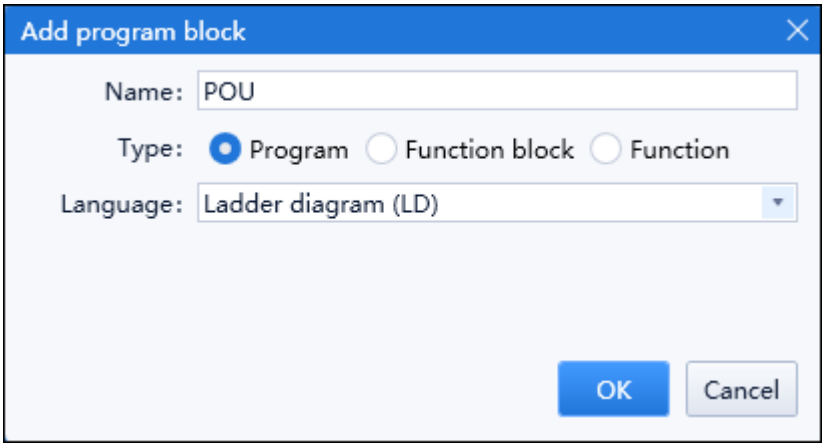
- A project has been created.
- A device has been added.

Procedure

1. Right-click "Programming" in the project tree and select "Add program block".



2. In the pop-up window, set the name of the program block, select "Program", "Function block", or "Function" for the type, and select "Ladder diagram (LD)" for the IEC language, as shown in the figure below.



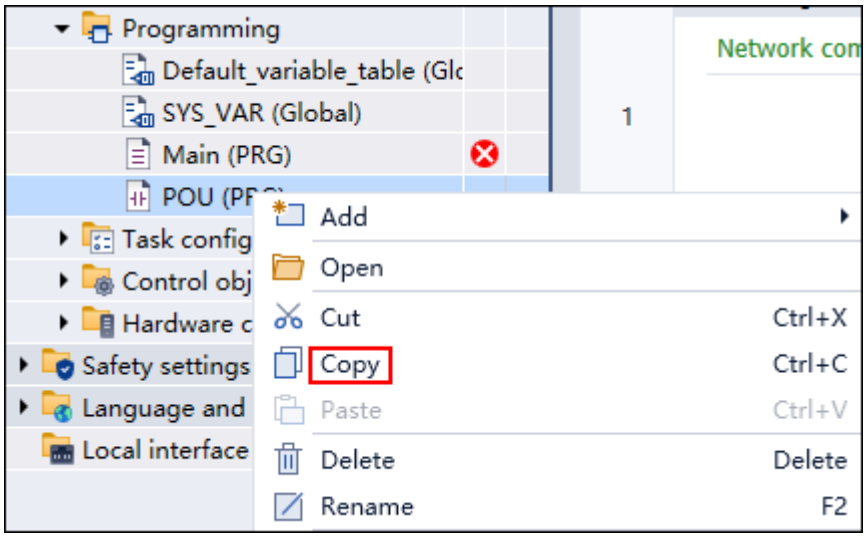
3. Click "OK" and the LD editor will be automatically opened.

10.1.3.2.2 Managing an LD Program Block

10.1.3.2.2.1 Copying an LD Program Block

LD program blocks can be copied and pasted within a PLC device, between PLC devices in the same project, or between different projects. The internal program logic of the LD program block remains unchanged after pasting.

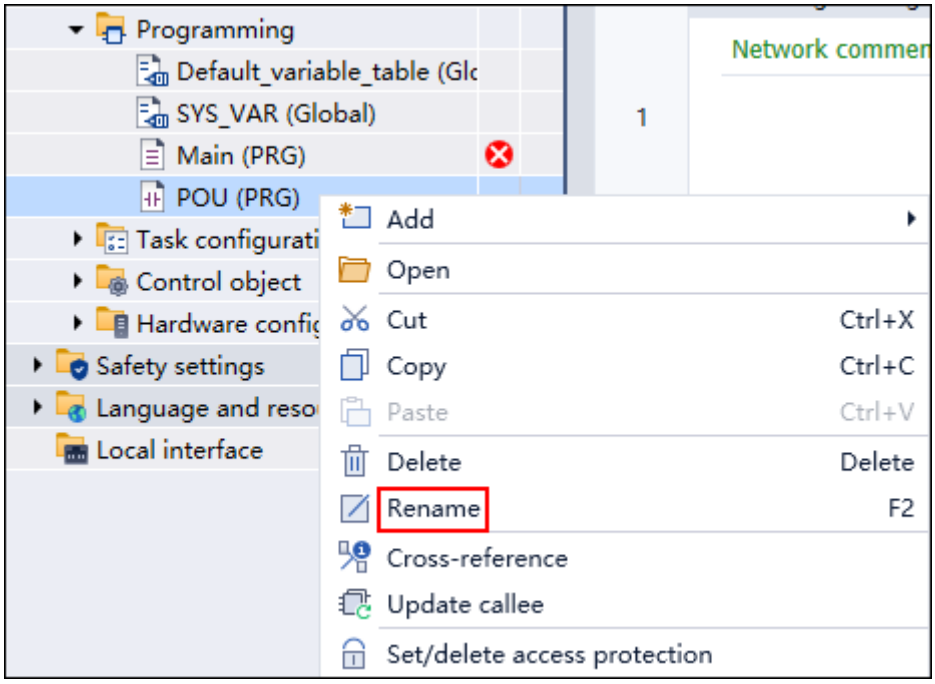
1. Right-click an LD program block name under the "Programming" node, and select "Copy".



2. Right-click the "Programming" node or its subordinate node, and select "Paste" to paste the copied LD program block at the corresponding location.

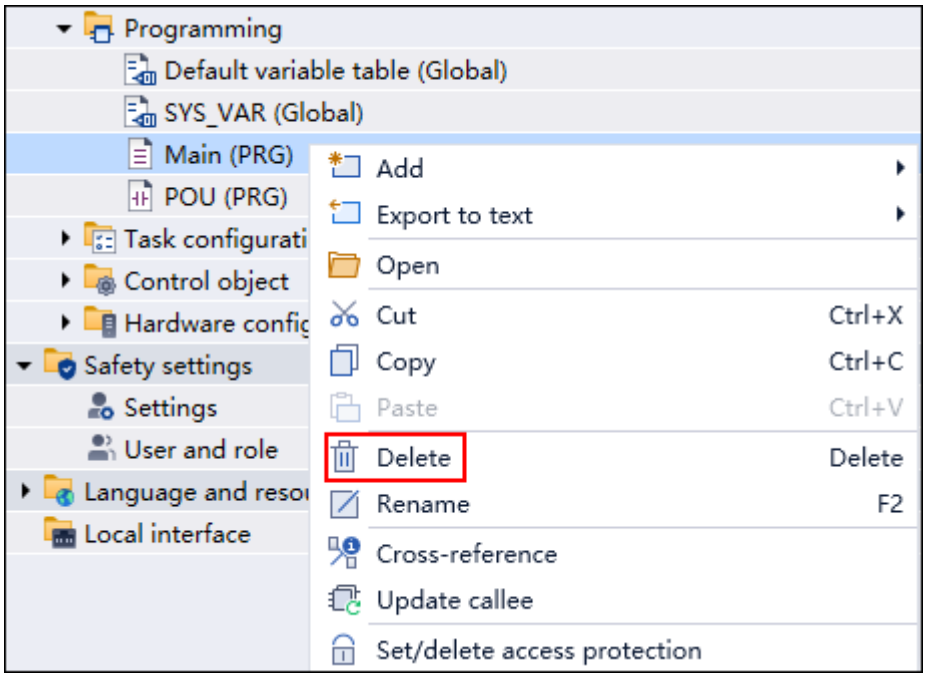
10.1.3.2.2.2 Renaming an LD Program Block

Right-click an LD program block name under the "Programming" node, select "Rename", and modify the group name.



10.1.3.2.2.3 Deleting an LD Program Block

1. Right-click an LD program block name under the "Programming" node, and select "Delete".

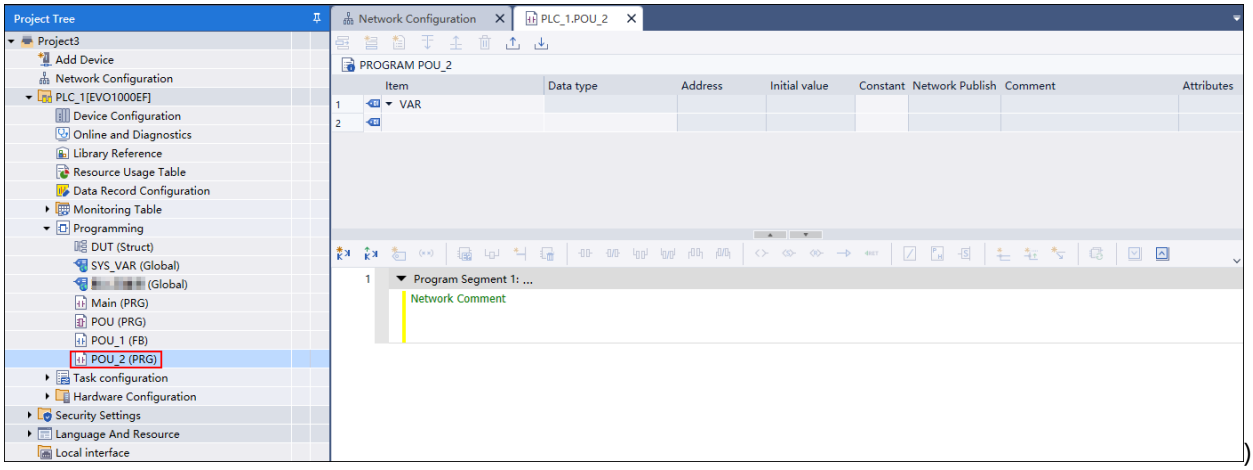


2. Click "OK" in the pop-up window.

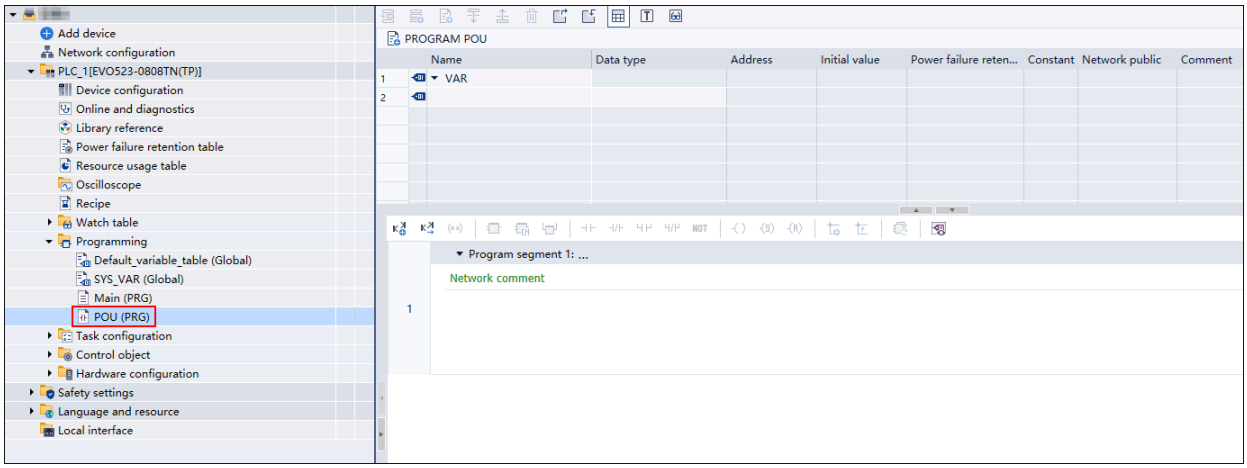
10.1.3.3 LD Editor

10.1.3.3.1 LD Editor Interface

Double-click the LD language program block under the "Programming" node (node identifier:

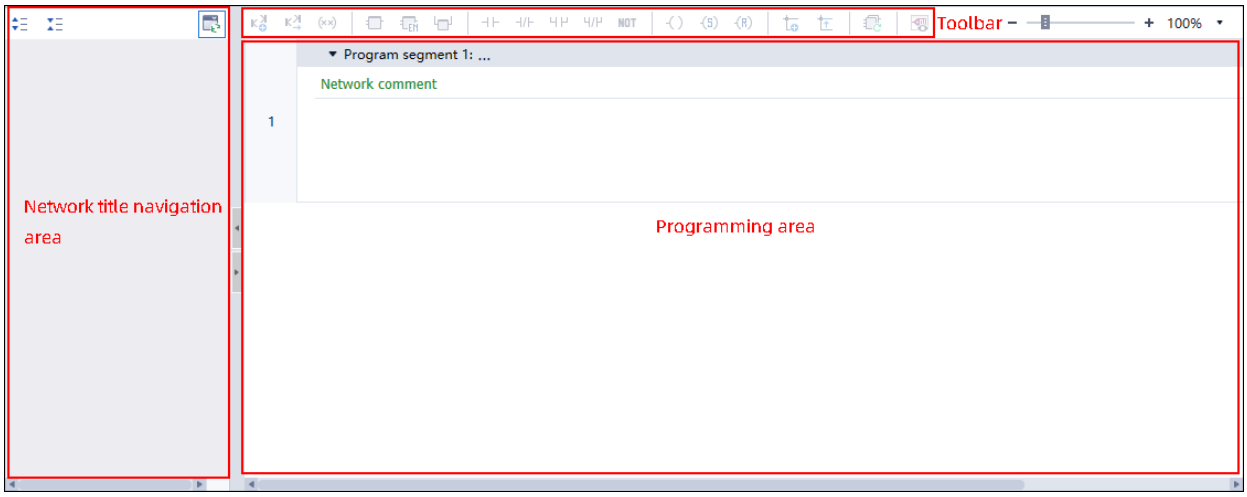


) to open the LD editor.



The LD editor is divided into the variable declaration area and the program editing area, which can be collapsed and expanded using and .

The program editing area consists of three parts: the toolbar, the programming area, and the network title navigation area, as shown in the figure below:



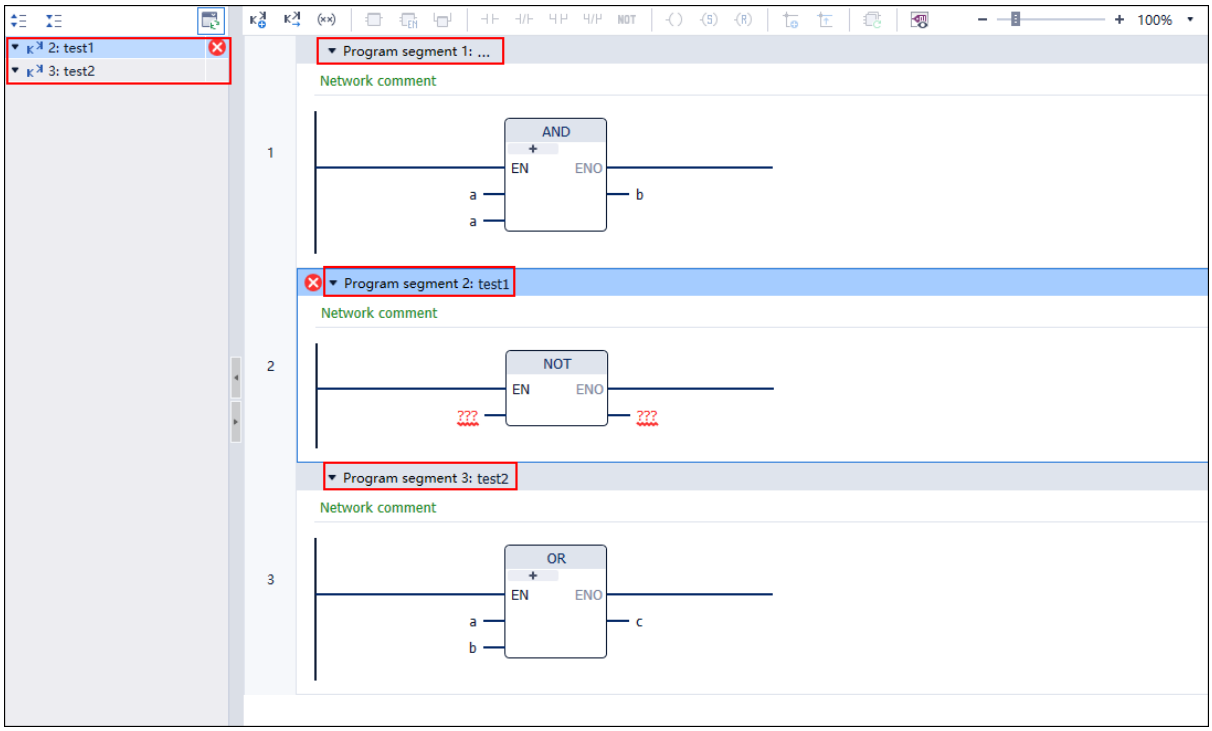
- Toolbar

Description	Icon	Description
Add network		Add a network in the program editing area.
Inserting a Network		Insert a network in the program editing area.
Switching Network Comment Status		During program commissioning, after the network is switched to annotation status, it no longer participates in compilation and computation.
Insert an empty operation block		Used to insert a rectangular block. Enter the name of the function or function block at "???". It is suitable for users who are familiar with functions and function blocks.
Insert an empty operation block with EN/ENO		Directly insert an empty operation block with EN/ENO, suitable for users who are not familiar with functions and function blocks.
Insert parallel operation block below		Insert a parallel operation block below the current operation block.
Insert an NO contact		Insert an NO contact.
Insert an NC contact		Insert an NC contact.
Insert a parallel NO contact		Multiple contacts can be connected in parallel; insert an NO contact in parallel below the current contact.
Insert a parallel NC contact		Multiple contacts can be connected in parallel; insert an NC contact in parallel below the current contact.
Energy flow inversion		Inverts the status of the input signal or the result of a logical operation.
Insert a coil		Insert a coil to pass the status of the left-connected element to the relevant Boolean variable and the right-connected element.
Insert a set coil		Insert a set coil; when the status of the left-connected element is TRUE, the Boolean variable of this coil is set and remains set until it is reset by a reset coil.
Insert a reset coil		Insert a reset coil; when the status of the left-connected element is TRUE, the Boolean variable of this coil is reset and remains reset until it is set by a set coil.
Insert a branch		Insert a branch.
Insert a branch above		Insert a branch above the current position.
Update all parameters		Update all parameters in the program editing area if the parameters of the called object are changed.

Description	Icon	Description
Hide/Show variable comment		Set the visibility of variable comments, which only applies to the current POU.

- Programming area
  Used for LD programming.
- Network title navigation area
  This area visually displays network title information in the LD program, enabling quick location and jump to target positions within the network. The network title navigation area contains two columns: the first column includes the collapse/expand icons and network titles; if the network title is empty, it will not be displayed here; the second column displays the network status, where an error flag is shown if there are pre-check or compilation faults in the network.

  Use  and  to collapse or expand the network title navigation area.



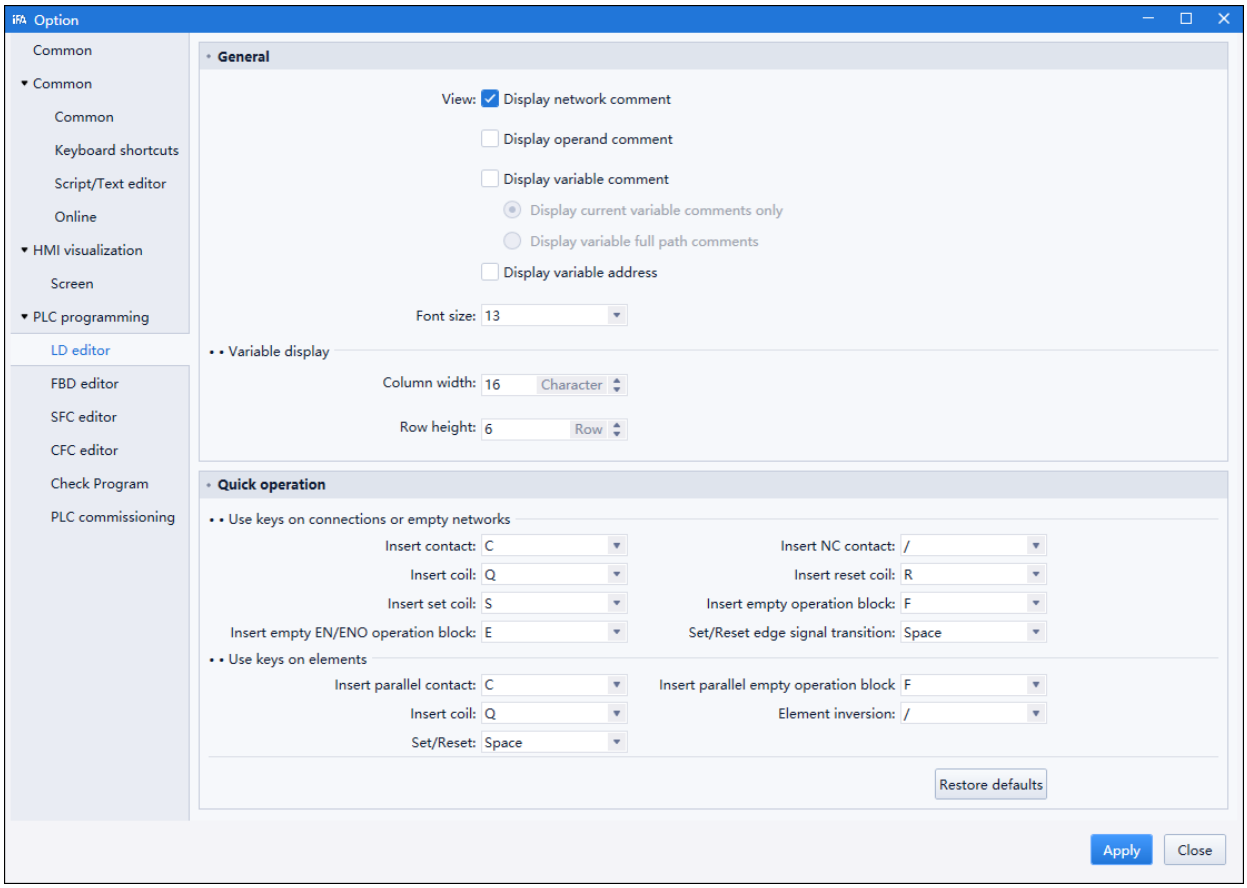
In this area, network titles can be copied, renamed, expanded/collapsed, and synchronized through the right-click menu or toolbar.

Function	Description
Copy	Copy the network corresponding to the network title node within the LD program editing area. The copied Regions can be pasted in the program editing area.
Rename	Rename the network title node and the name of the corresponding network in the LD program editing area will be synchronized, and vice versa. The network title in the navigation area cannot be empty.
Expand/Collapse	Expand or collapse the selected network title node.
Expand all/ Collapse all	Expand or collapse all network title nodes.

Function	Description
Sync on/off	◦ When synchronization is enabled: click a network title node on the left and the corresponding network will be displayed in the LD program editing area; the collapse, select, and highlight operations performed on the navigation area will be mirrored in the LD program editing area, and vice versa. ◦ When synchronization is disabled: the collapse, select, and highlight operations will not be synchronized between the network title navigation area and the LD program editing area.

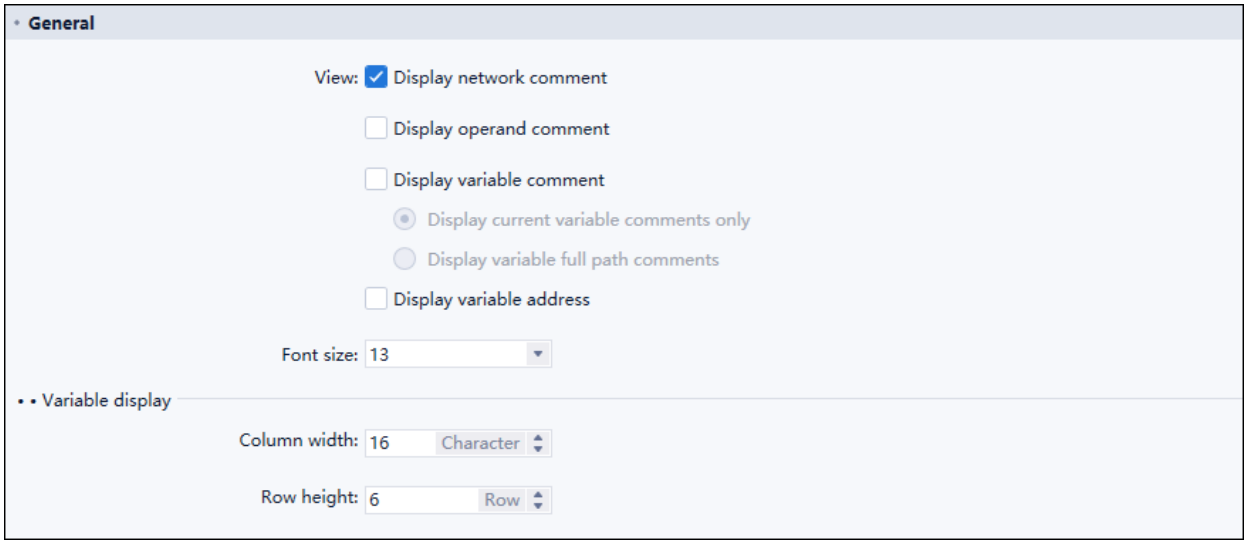
10.1.3.3.2 LD Editor Settings

In the software menu, select "Tools > Options > LD editor".



General items

The general items set the information visibility, font size, and variable display range in the LD editing area.

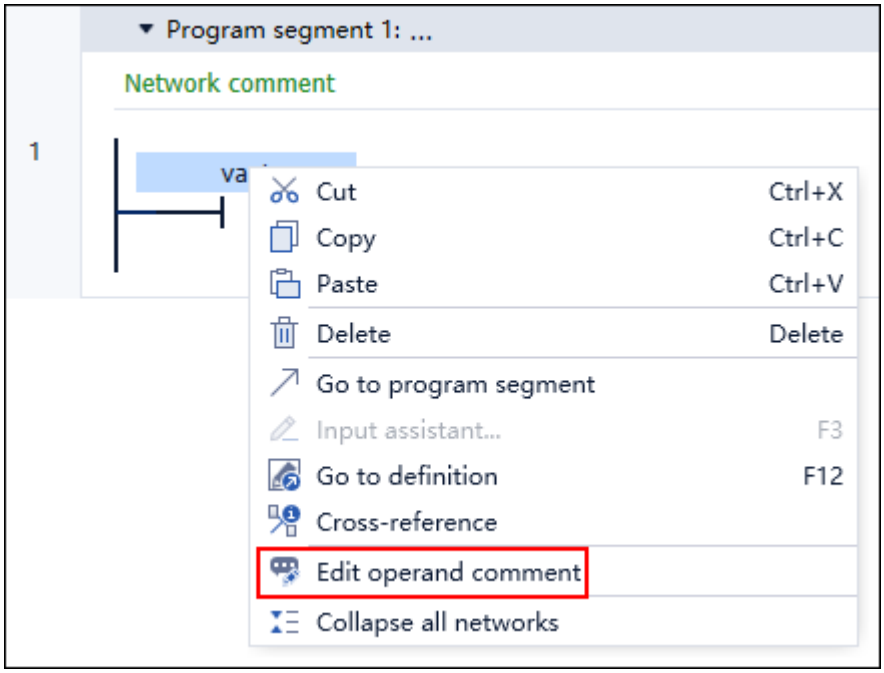


- Show network comment: Sets the visibility of network comment.

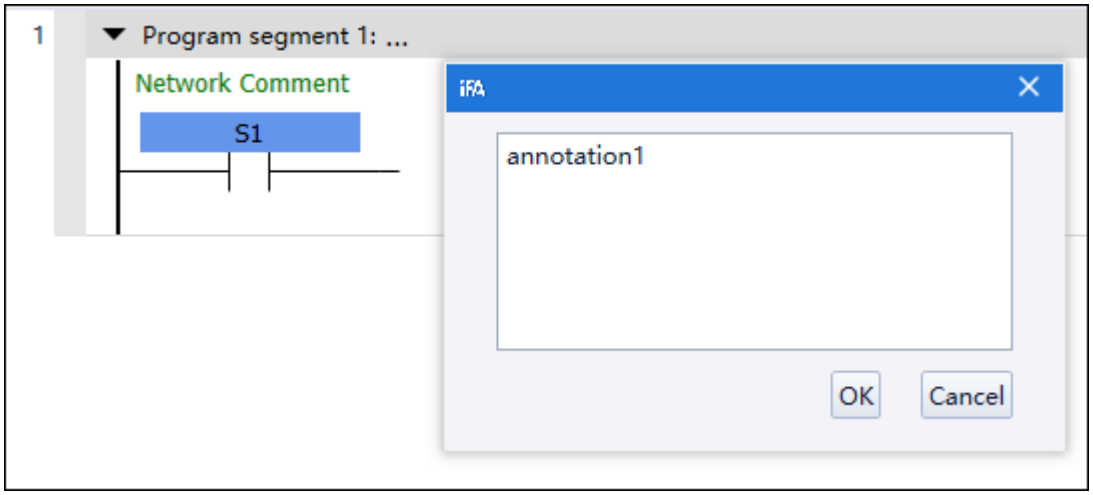


- Show operand comment: Sets the visibility of operand comment.

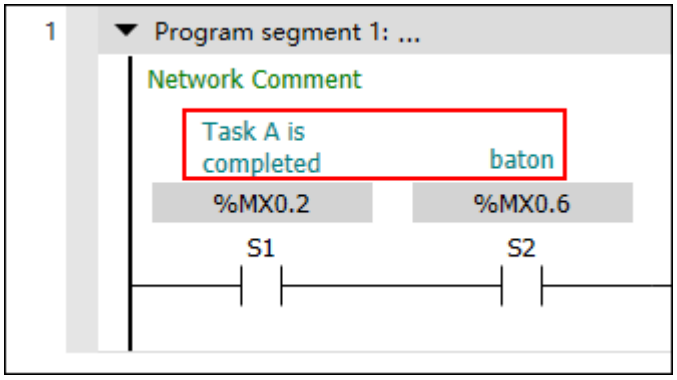
1. After checking "Display operand comment", right-click an operand in the program editing area and select "Edit operand comment".



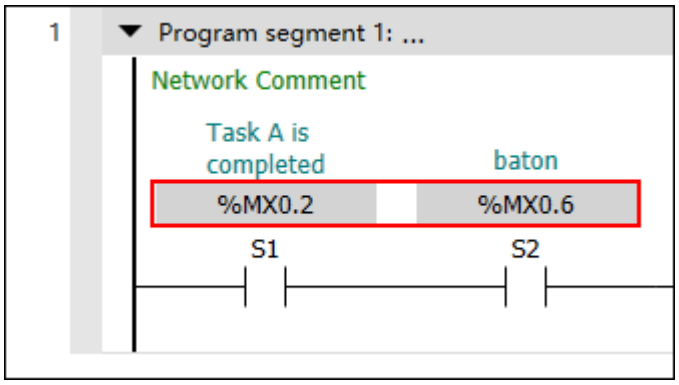
2. The entered comment will be displayed above the operand. If "Display operand comment" is unchecked, the comment will not be displayed.



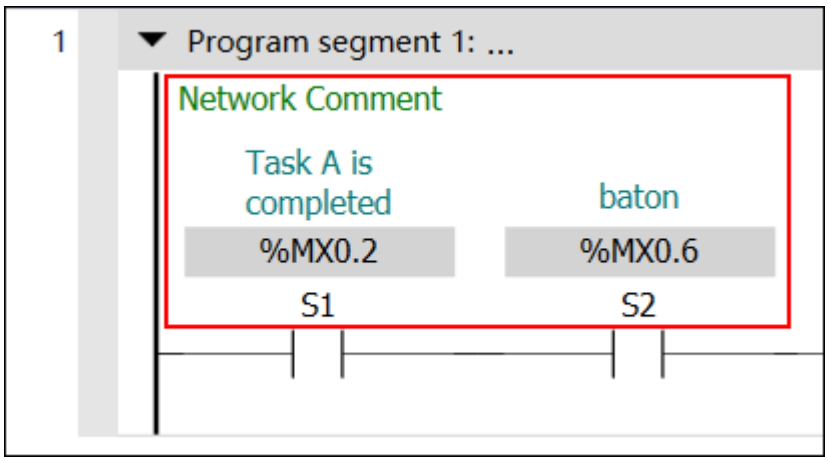
- Show variable comment: Sets the visibility of variable comments. Supports two modes:
  - Display current variable comments only: Displays the comment for only the currently selected variable.
  - Display variable full path comments: Displays the comment for the variable along with its full path.



- Display variable addresses: Sets the visibility of variable addresses.



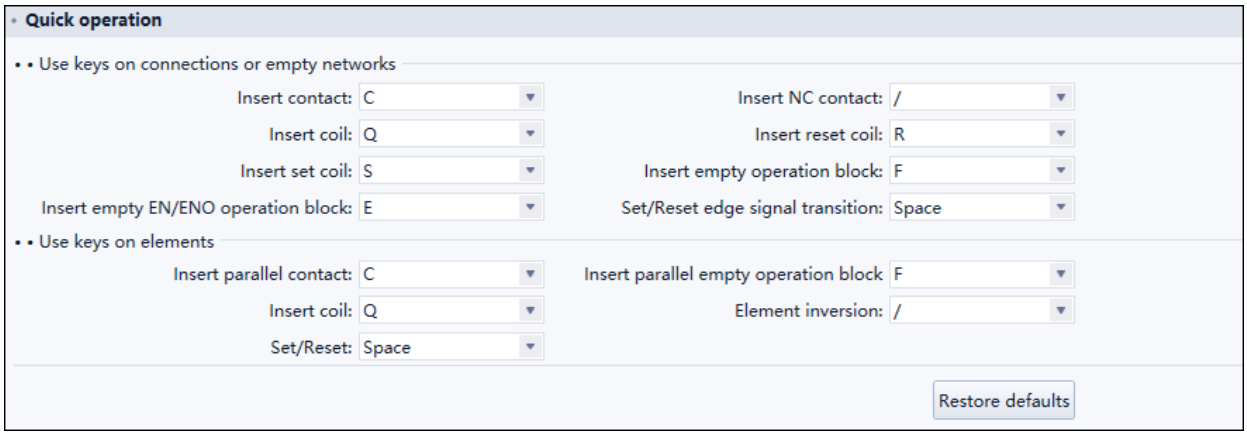
- Font size: Sets the font size of the text in the editing area.
  Applicable objects include network annotations, operand annotations, label names, variable names, variable annotations, variable addresses, jump positions.



- Variable display: Sets the length and width of variables displayed in the LD editing area. By default, the column width is 16 characters, with a range of 8 to 32 characters, and the row height is 6 lines, with a range of 3 to 99 lines.

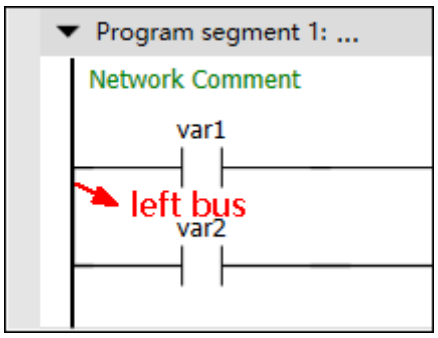
LD tab

The items in the "LD" tab sets commonly used shortcut keys for LD program editing, selected from drop-down lists. Click "Restore defaults" to reset all shortcut keys to their default values.



10.1.3.3.3 Lines in LD Editor

- **Power rail**
  The power rails in a ladder diagram are referred to as buses. It is on the left side of the ladder diagram and can also be called the left power bus. The figure below shows the graphical representation of the left bus.



- **Connection**
  A connection line refers to a line connecting one position to another. Connection lines mainly include contact connections, input/output connections of operation blocks, connections between coils, terminal connections, branch terminal connections, and vertical connections.

  Connection line modification include three cases: forming horizontal parallel connections, splitting parallel connections into left and right parts, and closing disconnected connections.

Figure 10-1-3-3-3-1 Horizontal connection elements



Figure 10-1-3-3-3-2 Vertical connection elements

10.1.3.3.4 LD Elements

- **Add network**

  Used to add a network in the program editing area.

  Table10-1-3-3-4-1 Icon and description for adding a network

Function	Icon	Description
Add network		Add a network in the program editing area.

- **Label**

  Label is an optional identifier and its address can be determined when it is jump-referenced.

  Table10-1-3-3-4-2 Icon and description for labels

Function	Icon	Description
Insert label		Enabled when the entire network is selected. A label is at the head of the network and marks the jump position. Users can jump to the label position through a jump element. A jump label is a string in comply with the identifier naming rules.

- **Function block**

  Used to insert an operation block (containing contacts and coils). In the network, a function/function block must be equipped with an input and an output with Boolean values. They can be used like contacts at the same position, that is, on the left side of the LD network.

  Table10-1-3-3-4-3 Icon and description for operation blocks

Function	Icon	Description
Empty operation block		Used to insert a rectangular block. Enter the name of the function or function block at "???". It is suitable for users who are familiar with functions and function blocks.
Empty operation block with EN/ENO		The function or function block is executed only when EN is TRUE. The status can be passed downstream. It is suitable for users who are less familiar with functions and function blocks.

  Ladder diagram programming language supports calling functions and function blocks. When calling, note the following:

  - In ladder diagrams, a function/function block is represented by a rectangular box. Functions can have multiple input parameters but only one return parameter. Function blocks can have multiple input parameters and multiple output parameters.
  - To ensure that energy flow can pass through the functions or function blocks, each called function/function block must have at least one input and one output parameter. To execute the connected function blocks, at least one Boolean input should be connected horizontally to the vertical left power rail.
  - When calling a function block, the actual parameter values can be directly entered at the external connection line of the formal parameter variables within the function block.

- **Execution block**

  Execution blocks can be inserted with embedded ST, where ST statements can be modified within the block. They can be zoomed in and out.

Table10-1-3-3-4-4 Icons and descriptions for execution blocks

Type	Icon	Description
Execution block		Directly inserts an execution block.

- **Contact**

  Contacts are graphical elements of the ladder diagram. They are the same as the contacts in electrical logic diagrams and are used to represent changes in the status of Boolean variables. They pass a status to the horizontally connected elements on their right side.

  Contacts can be divided into normally open contacts (NO) and normally closed contacts (NC). A normally open contact is disconnected under normal operating conditions, with its status being FALSE. A normally closed contact is connected under normal operating conditions, with its status being TRUE. The following table summarizes the commonly used contact icons and descriptions.

Table10-1-3-3-4-5 Icons and descriptions of contacts

Contact	Icon	Description
Normally open contact		If the current Boolean variable value corresponding to the contact is TRUE, the contact closes. If the status of the element connected to the left side of the contact is TRUE, the TRUE status is passed to the right side of the contact, making the status of the element connected to the right side TRUE. Conversely, when the Boolean variable value is FALSE, the status of the element connected to the right side is FALSE.
Normally closed contact		Multiple contacts can be connected in series by inserting contacts on the right side. When all the contacts connected in series are closed, the last contact can transmit TRUE.
Parallel NO contact		Multiple contacts can be connected in parallel by inserting NO contacts below the contact. If one of the two parallel contacts is TRUE, then the parallel line transmits TRUE.
Parallel NC contact		Multiple contacts can be connected in parallel by inserting NC contacts below the contact. NC contacts are closed by default. If the current Boolean variable value corresponding to the contact is FALSE, and the status of the element connected to the left side is TRUE, then the element connected to the right side of the parallel contact transmits TRUE.

- **Energy flow inversion**

  Energy flow inversion is used to invert signal flow or control logic for reverse control.

Table10-1-3-3-4-6 Icons and descriptions for energy flow inversion

Type	Icon	Description
Energy flow inversion		Inverts the status of the input signal or the result of a logical operation.

- **Coil**

  Coils are graphical elements of the ladder diagram. They are the same as the coils in electrical logic diagrams and are used to represent changes in the status of Boolean variables.

Coils can be divided into instantaneous coils and latching coils. The latter can be further divided into set coils and reset coils. The table below lists the commonly used icons and descriptions of coils in ladder diagrams.

Table10-1-3-3-4-7 Icons and descriptions of coils

Type	Icon	Description
Coil		Used to pass the status of the left-connected element to the relevant Boolean variable and the right-connected element. If the status of the left-connected element is TRUE, the Boolean variable of the coil is TRUE; otherwise, the variable is FALSE.
Set coil		There is an S in the coil. When the status of the left-connected element is TRUE, the Boolean variable of this coil is set and remains set until it is reset by a reset coil.
Reset coil		There is an R in the coil. When the status of the left-connected element is TRUE, the Boolean variable of this coil is reset and remains reset until it is set by a set coil.

- **Return**

Table10-1-3-3-4-8 Icon and description for return elements

Function	Icon	Description
Insert return		Insert a return element. When the input on the left side of the return element is TRUE, the program will immediately exits. The return element is located at the far right of the network.

- **Jump**

Table10-1-3-3-4-9 Icon and description for jump elements

Function	Icon	Description
Insert jump		Insert a jump element and specify a label. When the input on the left side of the jump element is TRUE, the program will jump to the specified label position. The jump element is located at the far right of the network.

- **FOR**

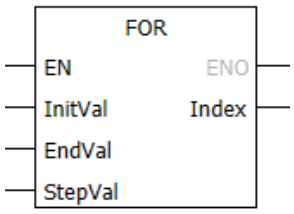
Table10-1-3-3-4-10 Icon and description for operation block FOR

Function	Icon	Description
FOR		The FOR function block is used to mark the starting position of a loop, and set the loop condition and loop count. It occupies an entire network segment.

NOTE

FOR and END_FOR function blocks must appear in pairs, and no other elements are allowed to exist to the right of the EXIT function block, otherwise a compilation fault will occur.

Table10-1-3-3-4-11 FOR function block parameters

Operation block	Parameter		Description
	Input Parameters	Initial value	Loop index start value (INT type)
		EndVal	Loop index end value (INT type)
		StepVal	Loop index increment (INT type)
	Output Parameters	Index	Current loop index value (INT type)

- **END_FOR**

Table10-1-3-3-4-12 Icon and description for operation block ENO_FOR

Function	Icon	Description
ENO_FOR		The END_FOR function block is used to mark the end position of a loop and occupies an entire network segment.

- **EXIT**

Table10-1-3-3-4-13 Icon and description for operation block EXIT

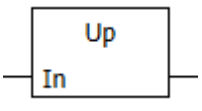
Function	Icon	Description
EXIT		The EXIT function block is used in the network between FOR and END_FOR blocks and occupies an entire network segment. It is used to immediately terminate and exit the current innermost FOR loop iteration when a specific condition is met. When the EXIT statement is executed, it directly jumps out of the FOR loop structure, no longer executing subsequent instructions within the loop, and proceeds to execute the next instruction following the corresponding END_FOR.

10.1.3.3.5 LD Instructions

10.1.3.3.5.1 Up

This instruction outputs a TRUE signal for only one task period when an input signal transitions from FALSE to TRUE (rising edge).

- **Instruction format**

Instruction	Name	FB/FC	LD Representation
Up	Up trigger	FB	

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Input signal	Follows data type.	FALSE	Valid when FALSE changes to TRUE.

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Output signal	Follows data type.	FALSE	Rising edge detected.

Variable type

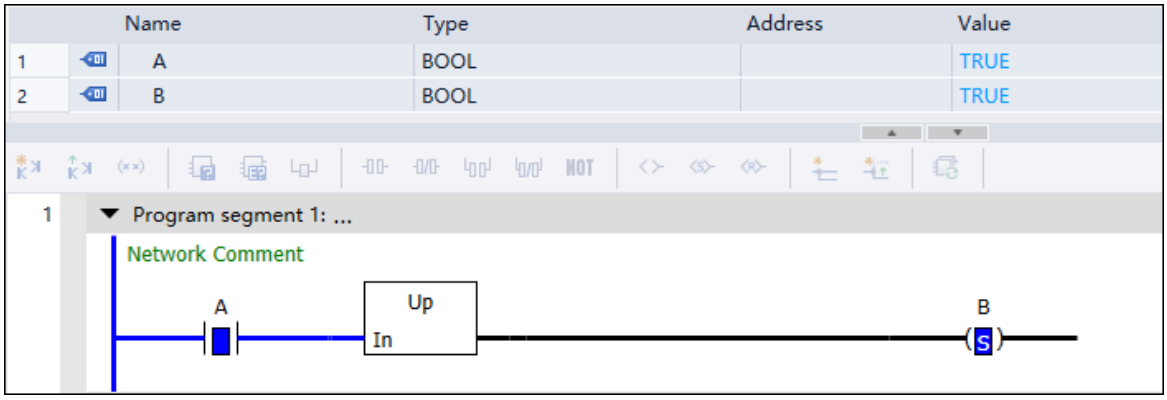
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**

  When the value of the input signal "In" is changed from FALSE to TRUE, the output pin signal is set to TRUE in only one task period and set to FALSE in other conditions.

- **Program example**

  LD:



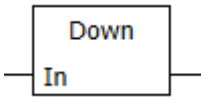
- **Precautions**

  The Up function block for rising edge detection does not support being called within a function; if called, a compilation fault will occur.

10.1.3.3.5.2 Down

This instruction outputs a TRUE signal for only one task cycle when an input signal transitions from TRUE to FALSE (falling edge).

- **Instruction format**

Instruction	Name	FB/FC	LD Representation
Down	Down trigger	FB	

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Input signal	Follows data type.	FALSE	Valid when TRUE changes to FALSE.
Output	Out	Output signal	Follows data type.	FALSE	Falling edge detected.

  Variable type

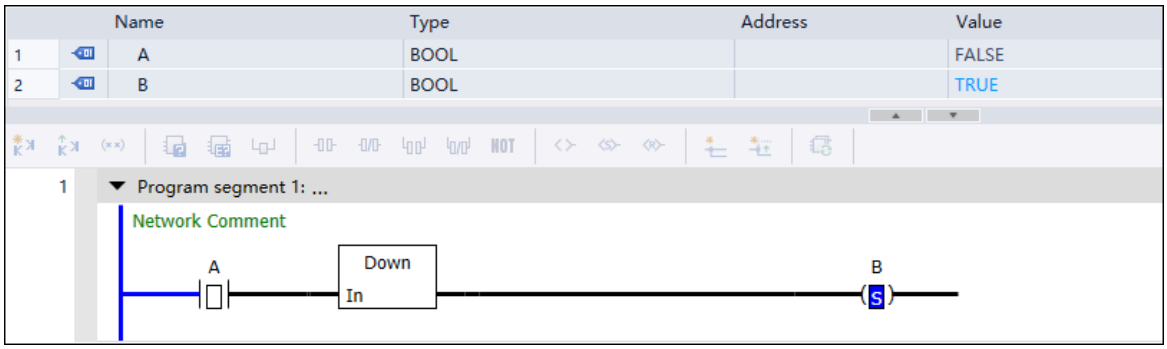
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**

  When the value of the input signal "In" is changed from TRUE to FALSE, the output pin signal is set to TRUE in only one task period and set to FALSE in other conditions.

- **Program example**

  LD:



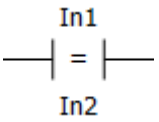
- **Precautions**

  The Down function block for falling edge detection does not support being called within a function; if called, a compilation fault will occur.

10.1.3.3.5.3 =

EQ determines whether the two input values are equal. If the first input value equals the second, the output is TRUE; otherwise, the output is FALSE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation
=	Equal instruction	FC	

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Input 1	Follows data type	0	Data 1
	In2	Input 2	Follows data type	0	Data 2
Output	Out	Return value	Follows data type	FALSE	Comparison result

Data type

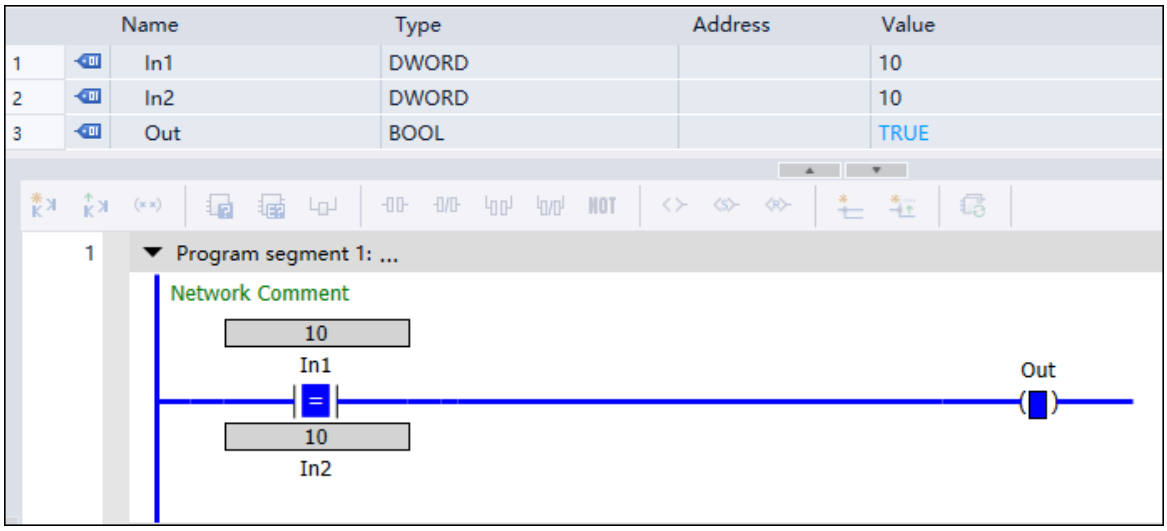
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**

Outputs TRUE when In1 equals In2, otherwise outputs FALSE.

- **Program example**

LD:



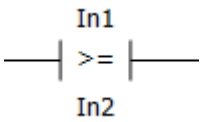
- **Precautions**

  When the data types of two input variables are inconsistent, a compilation fault occurs.

10.1.3.3.5.4 >=

GE compares the two input values. If the first input value is greater than or equal to the second, the output is TRUE; otherwise, the output is FALSE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation
>=	Greater than or equal instruction	FC	

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Input 1	Follows data type	0	Data 1
	In2	Input 2	Follows data type	0	Data 2
Output	Out	Return value	Follows data type	FALSE	Comparison result

  Data type

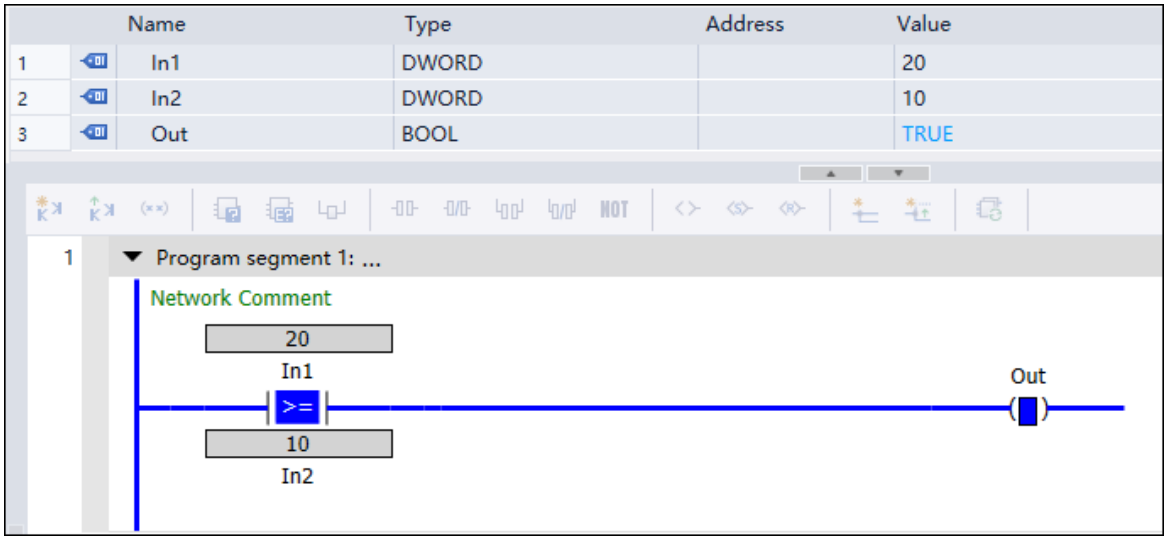
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**

  Outputs TRUE when In1 is greater than or equal to In2, otherwise outputs FALSE.

- **Program example**

  LD:



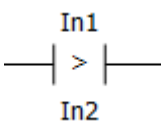
- **Precautions**

  When the data types of two input variables are inconsistent, a compilation fault occurs.

10.1.3.3.5.5 >

GE compares the two input values. If the first input value is greater than the second, the output is TRUE; otherwise, the output is FALSE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation
>	Greater than instruction	FC	

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Input 1	Follows data type	0	Data 1
	In2	Input 2	Follows data type	0	Data 2
Output	Out	Return value	Follows data type	FALSE	Comparison result

Data type

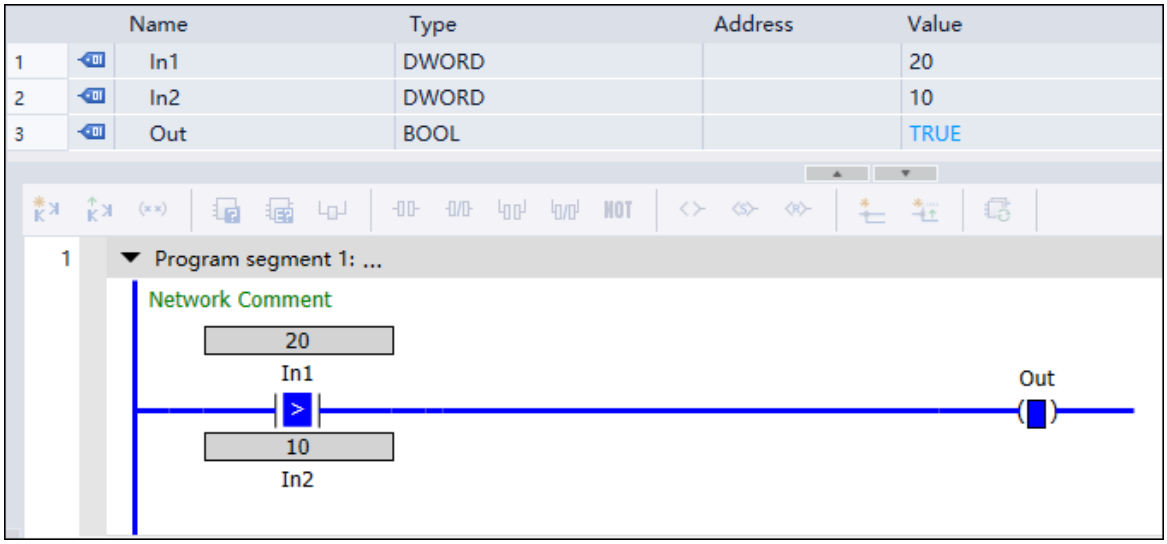
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**

  Outputs TRUE when In1 is greater than In2, otherwise outputs FALSE.

- **Program example**

  LD:



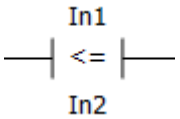
- **Precautions**

  When the data types of two input variables are inconsistent, a compilation fault occurs.

10.1.3.3.5.6 <=

GE compares the two input values. If the first input value is smaller than or equal to the second, the output is TRUE; otherwise, the output is FALSE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation
<=	Less than or equal instruction	FC	

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Input 1	Follows data type	0	Data 1
	In2	Input 2	Follows data type	0	Data 2
Output	Out	Return value	Follows data type	FALSE	Comparison result

Data type

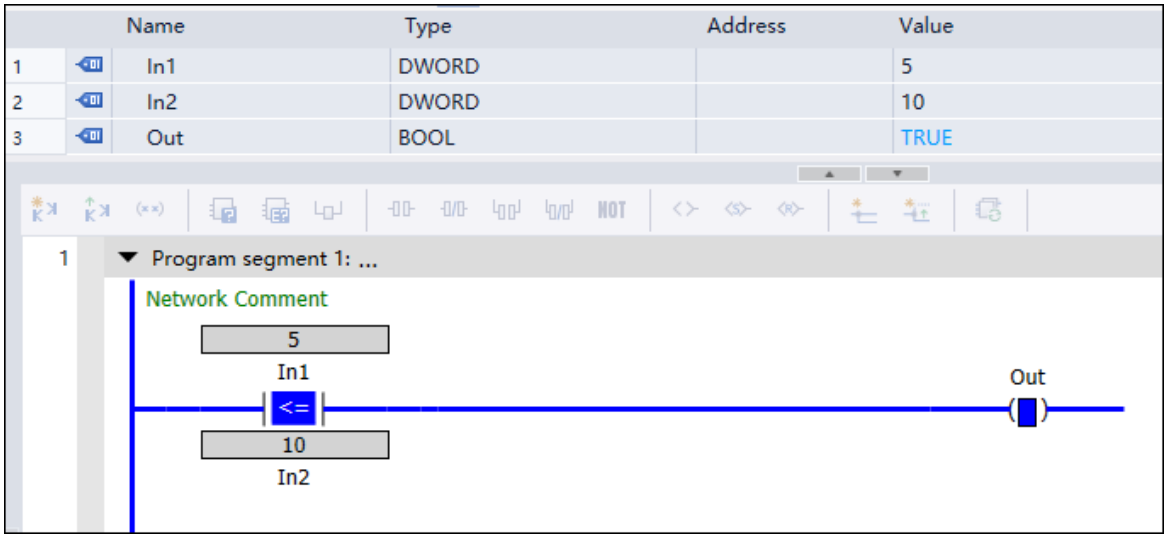
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**

Outputs TRUE when In1 is less than or equal to In2, otherwise outputs FALSE.

- **Program example**

LD:



- **Precautions**

  When the data types of two input variables are inconsistent, a compilation fault occurs.

10.1.3.3.5.7 <

GE compares the two input values. If the first input value is smaller than the second, the output is TRUE; otherwise, the output is FALSE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation
<	Less than instruction	FC	In1 < In2

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Input 1	Follows data type	0	Data 1
	In2	Input 2	Follows data type	0	Data 2
Output	Out	Return value	Follows data type	FALSE	Comparison result

  Data type

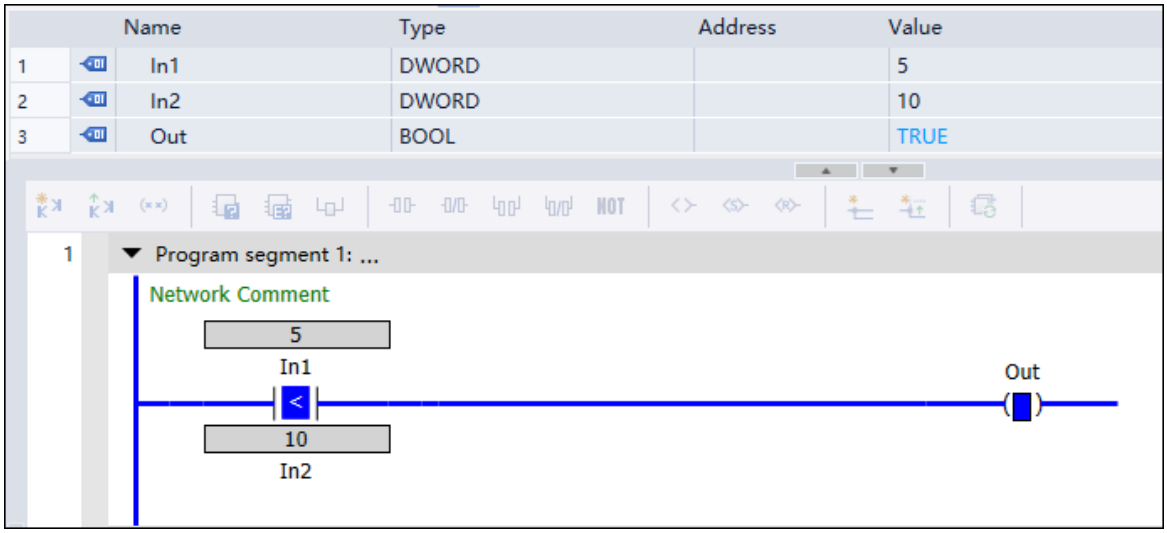
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**

  Outputs TRUE when In1 is less than In2, otherwise outputs FALSE.

- **Program example**

  LD:



- **Precautions**

  When the data types of two input variables are inconsistent, a compilation fault occurs.

10.1.3.3.5.8 <>

GE compares the two input values. If the first input value does not equal the second, the output is TRUE; otherwise, the output is FALSE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation
<>	Not equal instruction	FC	In1 <> In2

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Input 1	Follows data type	0	Data 1
	In2	Input 2	Follows data type	0	Data 2
Output	Out	Return value	Follows data type	FALSE	Comparison result

Data type

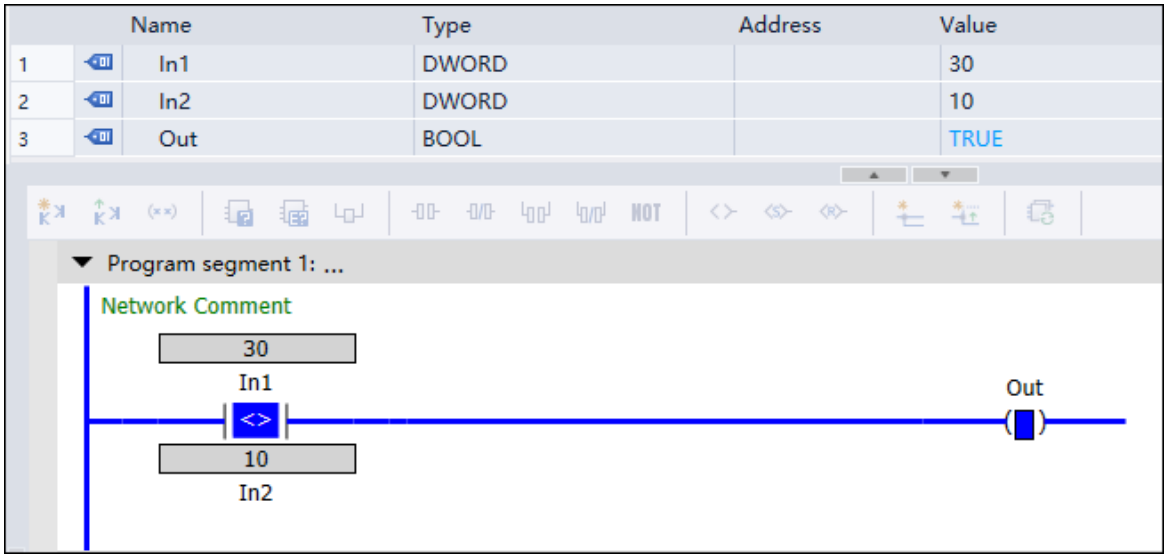
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**

  Outputs TRUE when In1 does not equal In2, otherwise outputs FALSE.

- **Program example**

  LD:



- **Precautions**

  When the data types of two input variables are inconsistent, a compilation fault occurs.

10.1.3.4 Network Usage

10.1.3.4.1 Network Overview

Network is the basic unit of the LD program.

In the LD editor, networks are arranged in a list. Each network has an incremental number on the left, and can include the following: logical and arithmetic expressions, program/function/function block calls, jump and return statements.

In 10.1.3.3.2 LD editor settings, you can define whether to display network comments, operand comments, variable comments, variable addresses, and so on in the editor.

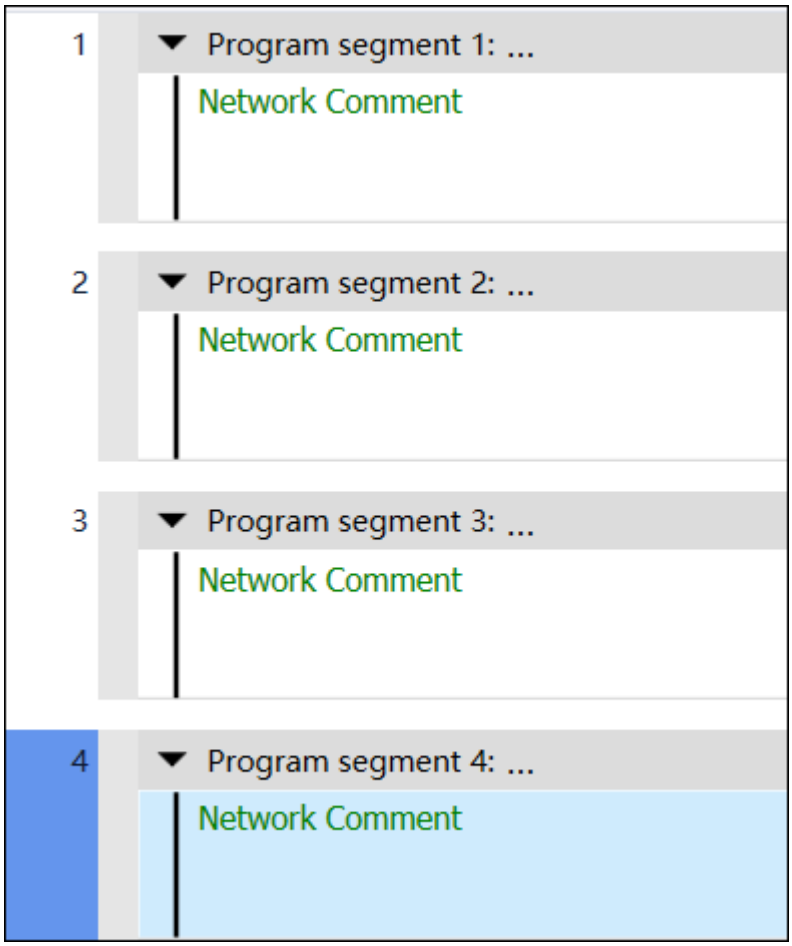
10.1.3.4.2 Adding a Network

Add a network in the program editing area.

Use the icon  or shortcut key Ctrl + T to add a network.

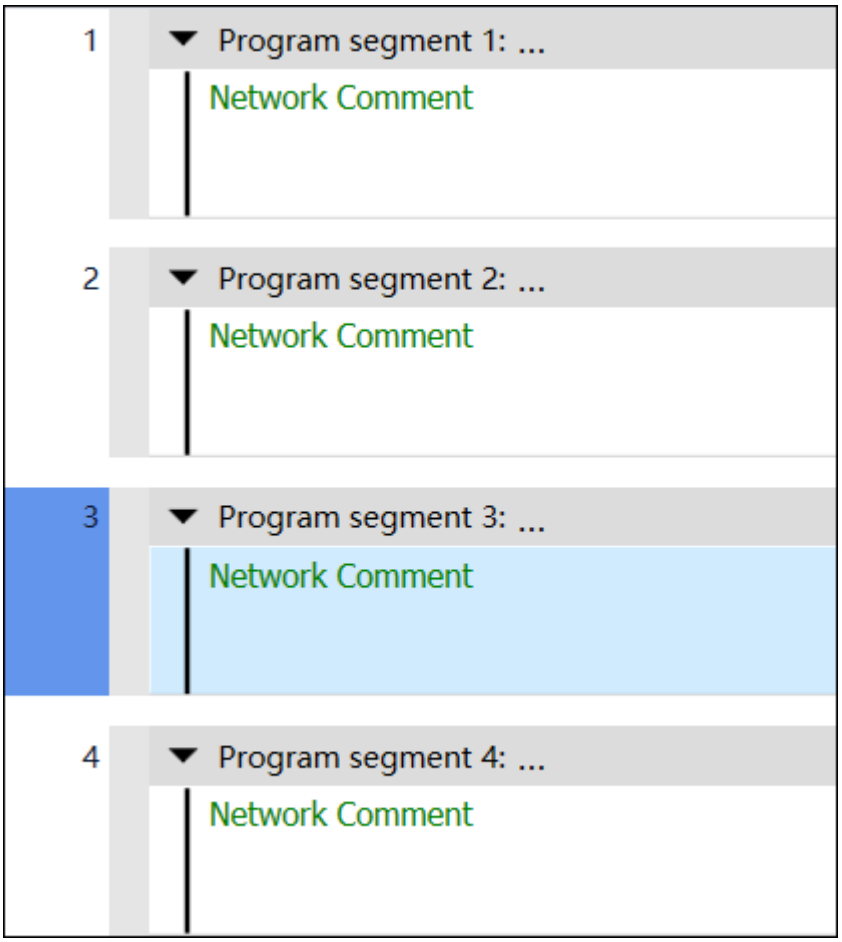
The mouse position affects where the network will be added.

- Selecting an empty space in the editing area
  The new network will be added to the end of the existing networks.



- Selecting a network
  The new network will be added below the selected network. The network number and network title number (program segment n:) of the new network will be automatically re-sorted.
  As shown in the figure below, selecting network 2 will result in the addition of network 3, whose network

number and title number will automatically increment by 1.



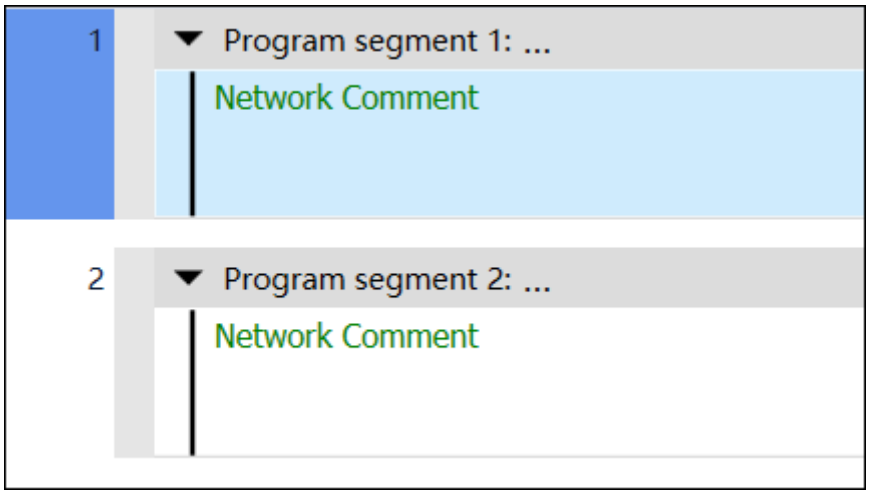
10.1.3.4.3 Inserting a Network

Insert a network in the program editing area.

Use the icon  or shortcut key Ctrl + J to insert a network.

Select a network in the editing area, and the new network will be inserted above the selected network.

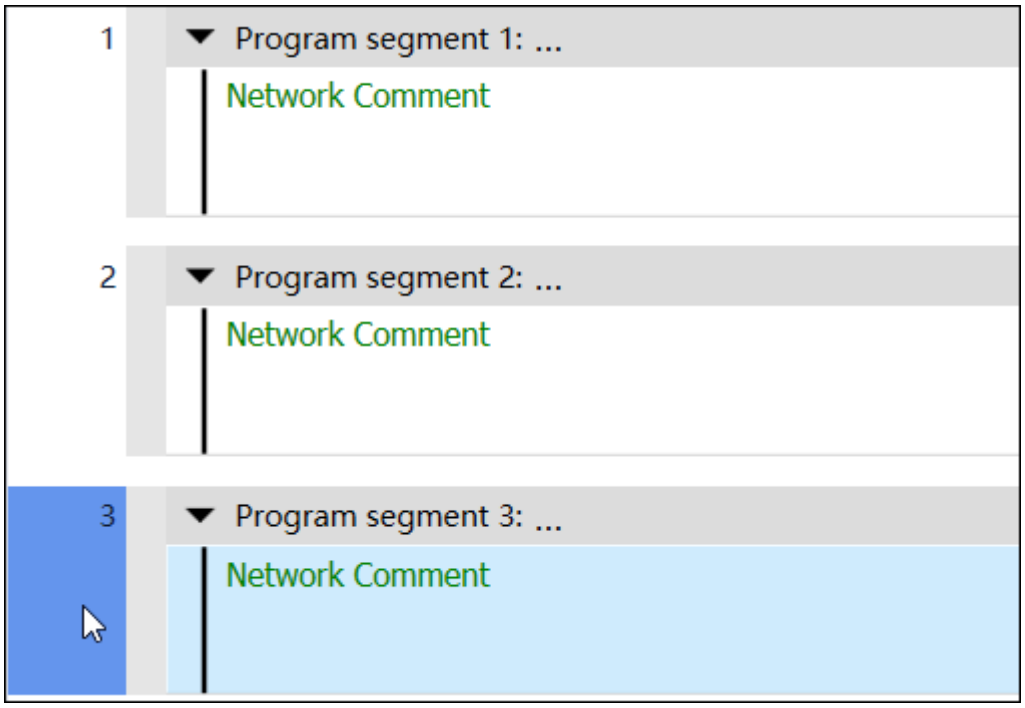
Network numbers and network title numbers (program segment n:) will be automatically re-sorted. In the following figure, network 1 before insertion becomes network 2.



10.1.3.4.4 Moving a Network

The program executes the networks in the editing area from top to bottom. To change the execution order, you must change the order of the network.

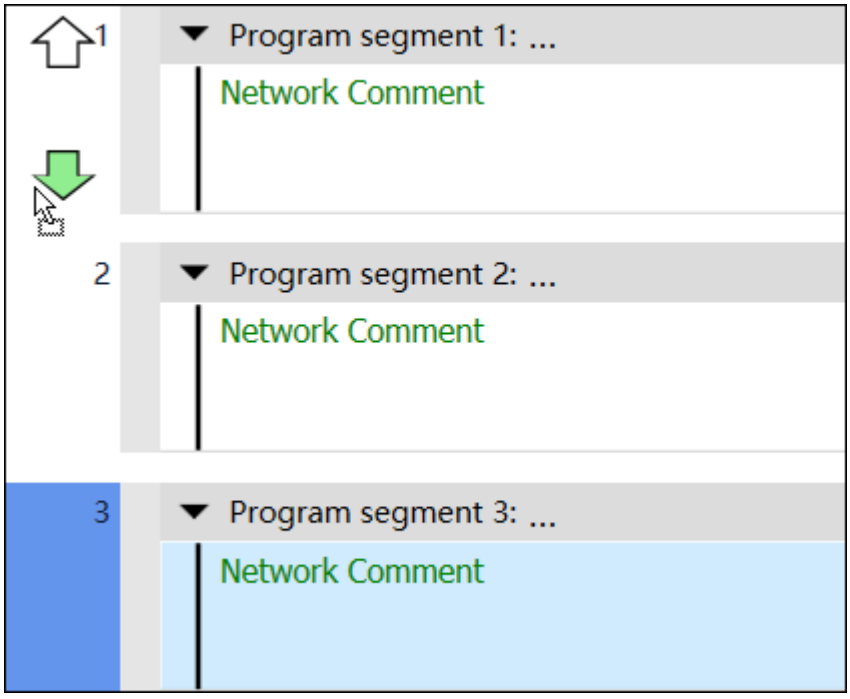
1. Select the network whose order you want to change.



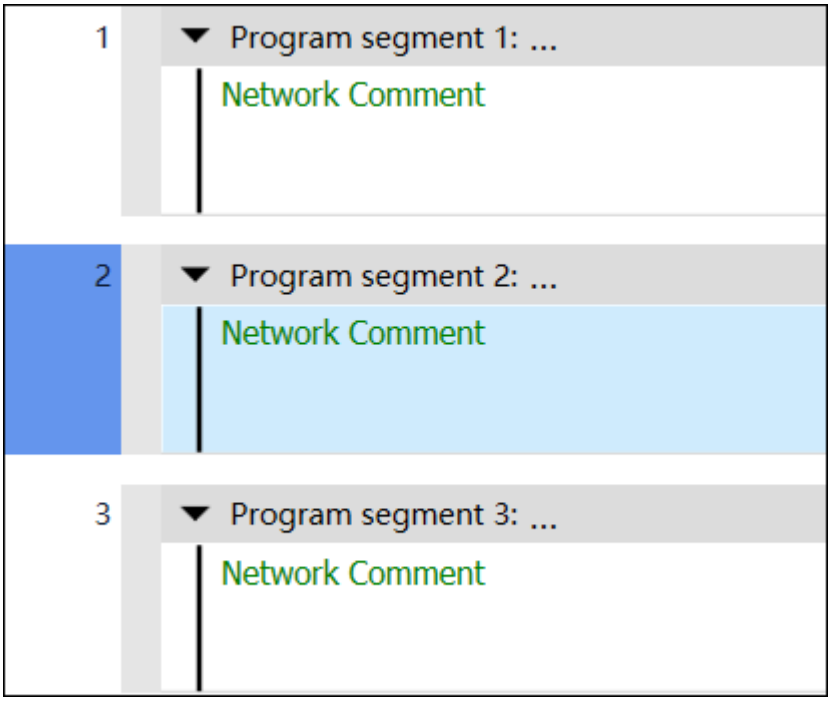
2. Drag it to the target position.

   When dragging a network, up/down arrows will appear on the left side of the network at the cursor position, indicating moving the selected network above/below that network. The arrow in green indicates the position where it will be inserted.

   The following figure indicates that Network 3 will be inserted below Network 1.



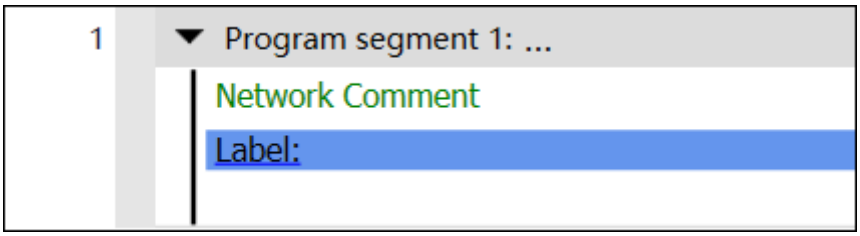
   Release the mouse to complete the moving, and the network number and network title number (program segment n:) will automatically refresh.



10.1.3.4.5 Inserting a Label

Jump labels can be added to a network to facilitate control logic jumps.

Right-click a network and select "Insert tag".



Select a network in the editing area, and the label will be displayed below the network annotation. The label ends a colon (:) and the default label is **Label:**.

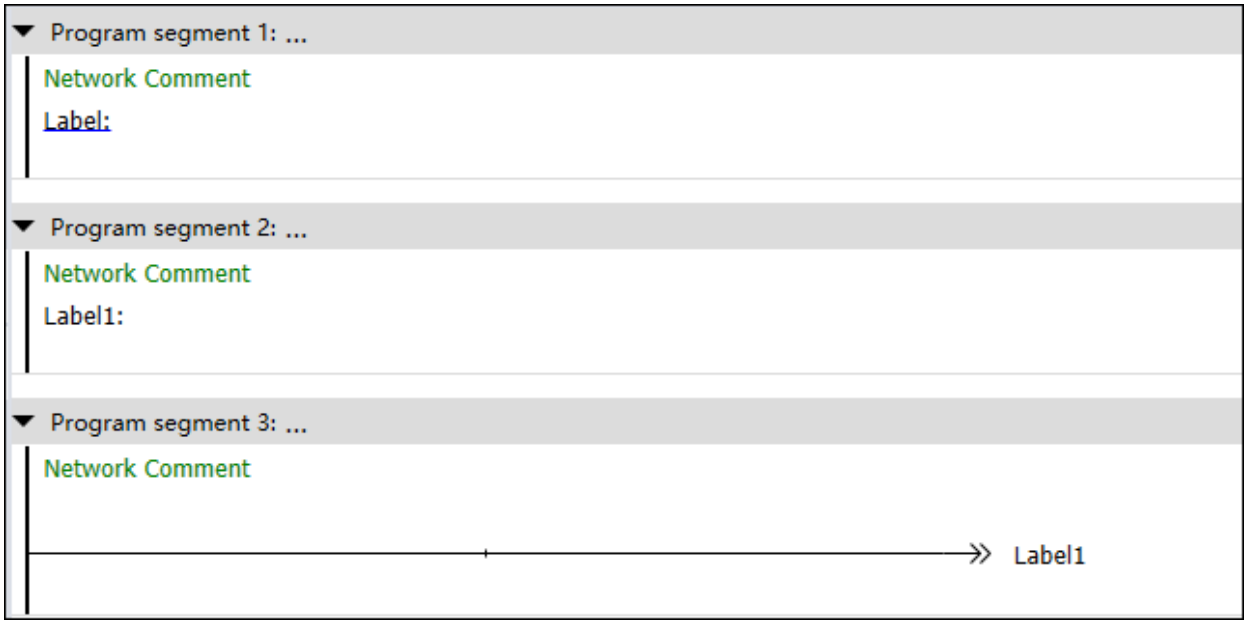
Double-click the label to modify its content. Duplicate names are not allowed within an editing area.

The font size of the label can be modified. For details, see 10.1.3.3.2 LD Editor Settings for configuration.

The status of the label is indicated by different symbols:

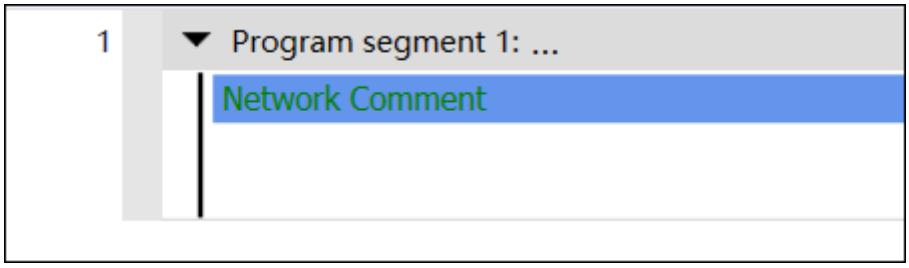
- Labels referenced: No symbol, such as the label in Network 1.
- Labels not referenced: Blue underline, such as the label in Network 2.
- Duplicate label: Red wavy line, such as the label in Network 3.

For the last two statuses, a prompt will appear when the mouse hovers over the label.



10.1.3.4.6 Modifying Network Comment

Annotations can be added to the program of a network. For example, users can add annotations to describe the control significance of the current network implementation.



Procedure of modifying annotations:

- Double-click the annotation line and type in the new description.
- Right-click a network, select "Edit network comment" and type in the new description.

If the comment line is invisible, see 10.1.3.3.2 LD Editor Settings to make it visible.

The font size of comments is adjustable; see 10.1.3.3.2 LD Editor Settings to complete the operation.

10.1.3.4.7 Switching Network Comment Status

During program commissioning, after the network is switched to annotation status, it no longer participates in compilation and computation.

When the network is in annotation status, operations such as "Go to definition" and "Cross-reference" are not supported.

Use the icon  to switch the network annotation status.

Select the target network. After executing the command, all elements within the network lose their control functions, and labels are converted to plain text, losing their jump positioning functions. Take the following two figures as an example. In the first figure, Label 1 is valid because the network is in normal status. In the second figure, Network 1 is in annotation status, it causes the Label 1 in Network 2 to become invalid.

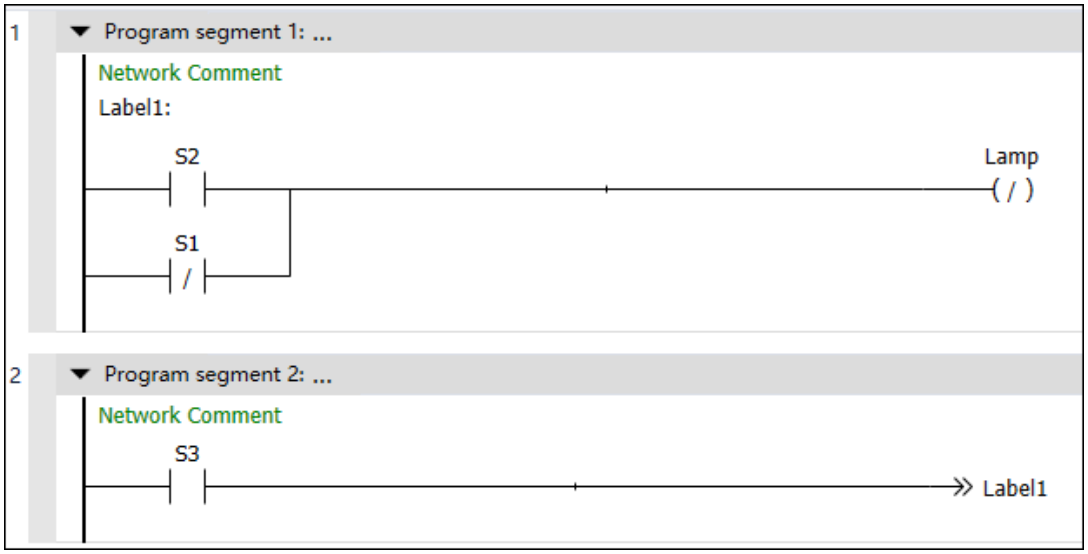


Figure 10-1-3-4-7-1 Network in normal status

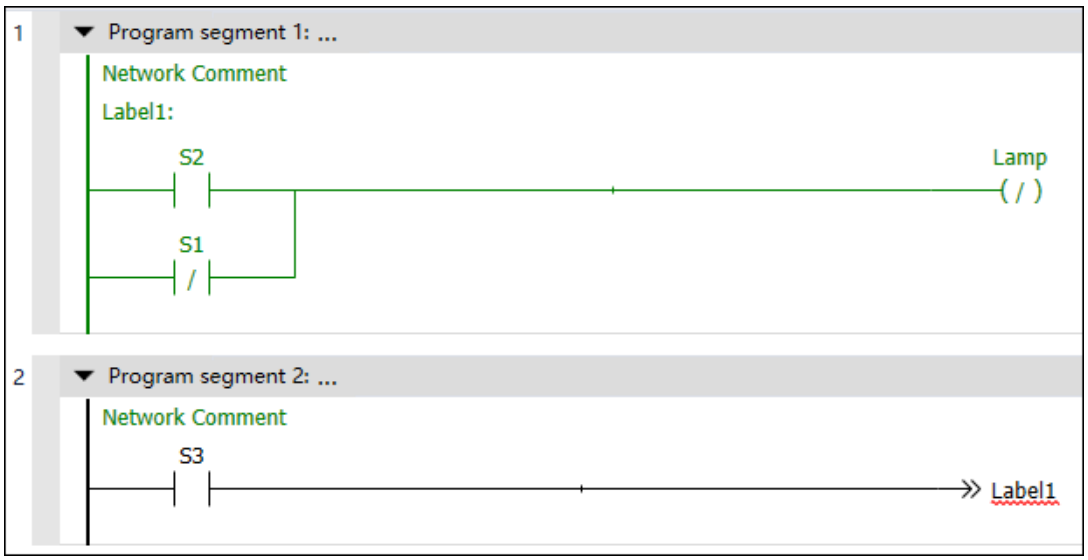
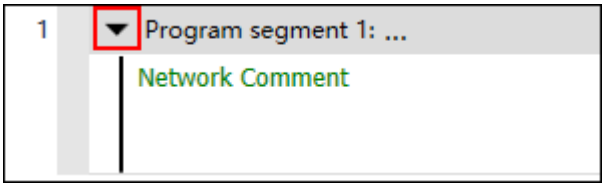


Figure 10-1-3-4-7-2 Network in annotation status

10.1.3.4.8 Collapsing/Expanding a Network

Networks can be collapsed or expanded in the following two ways:

1. Click the triangle icon on the left of the network name to collapse or expand the network.



2. Right-click a network and select "Collapse"/"Expand".

10.1.3.4.9 Deleting a Network

Users can delete a single or multiple networks.

1. Select the network(s) to be deleted.
   For batch operations, use the Ctrl key or Shift key.
2. Right-click the network(s) to be deleted and select "Delete", or use the Delete key.

10.1.3.5 Branch Usage

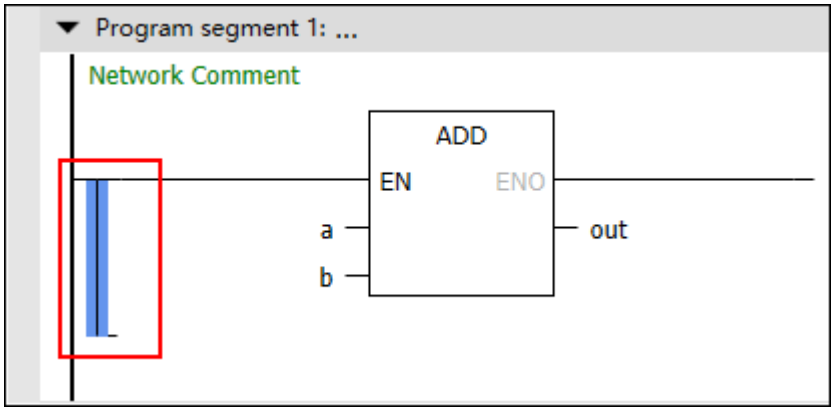
The branch forms a non-closed parallel logic.

Table10-1-3-5-1 Icon and description for branches

Function	Icon	Description
Insert a branch		Insert a branch.
Insert a branch above		Insert a branch above the current position.
Setting branch start/end point		Set the start/end point of the selected branch.

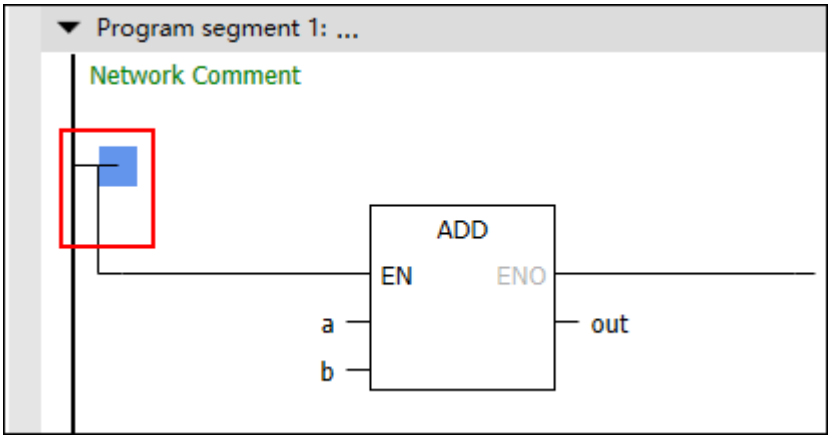
- **Inserting a branch**

  Select a connection line and click the icon to insert a branch below the selected connection line. The inserted branch is an open line.



- **Inserting a branch above**

  Select a connection line and click the icon to insert a branch above the selected connection line.



- **Setting branch start/end point**

  This operation is used to connect the start and end points of a branch. To connect the two points, right-click the branch start point and select "Set branch start point". will be displayed at the start point position. Right-click the branch end point and select "Set branch end point". will be displayed at the end point position and the two points are connected.

10.1.4 ST Language Programming

10.1.4.1 Overview

Structured text (ST) is an advanced text programming language that can be used to describe the behavior of functions, function blocks, and programs. It can also describe the behavior of steps, actions, and transitions in sequential function charts. For those familiar with high-level computer language development, ST is easier to learn and use, as it can implement features such as selection, iteration, and jump statements. Moreover, ST is highly readable, helping users quickly understand program logic through semantic identifiers and detailed comments.

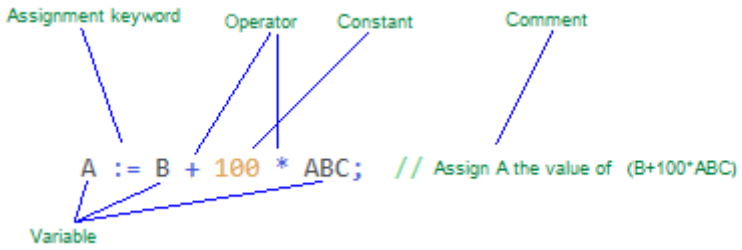
ST language structure

ST language allows for various operations, such as calling function blocks, assigning values, and conditionally executing instructions and repeating tasks. The scope of ST language can be extended by calling conditional statements and loop statements with corresponding keywords. The execution order can be determined by the sequence of statements arranged in the ST editor, from top to bottom and from left to right. This order can only be changed by inserting loop statements.

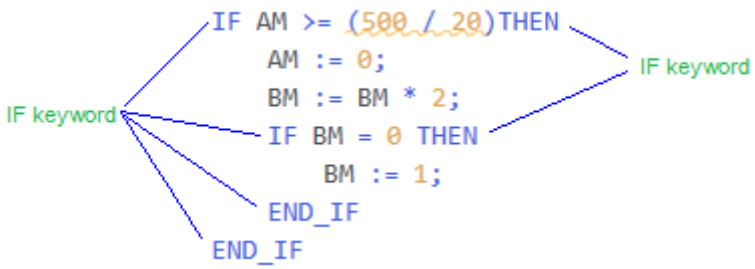
ST language consists of various elements, including the following:

- Expression: A structure composed of operands and operators, which returns a value when executed.
- Operand: Represents variables, numbers, addresses, function blocks, and so on.
- Operator: Symbols used in the execution of operations.
- Statement: Used to assign the value returned by an expression to an actual parameter, and construct and control expressions.

Example of an assignment statement:



Example of an IF statement:



10.1.4.2 ST Creation and Management

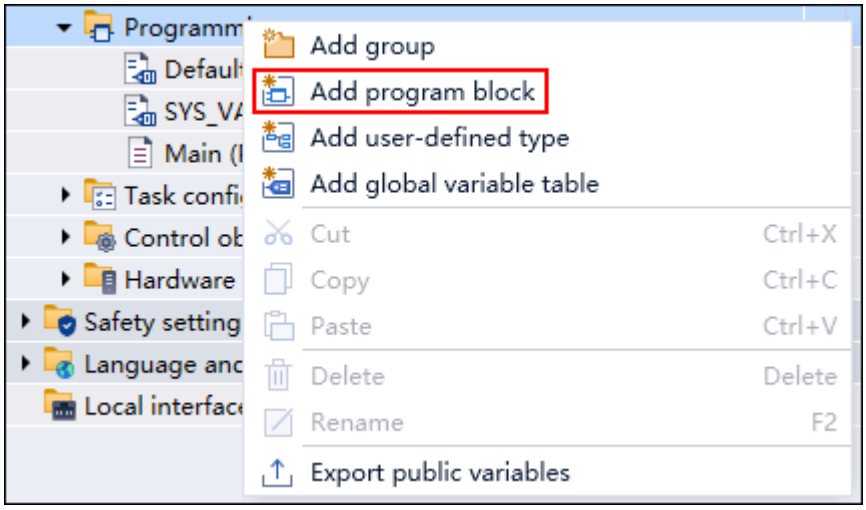
10.1.4.2.1 Creating an ST Program Block

Prerequisites

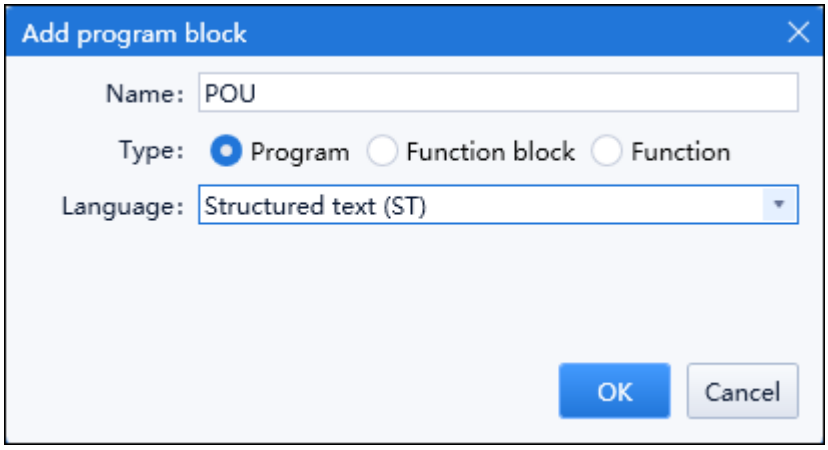
- A project has been created.
- A device has been added.

Procedure

1. Right-click "Programming" in the project tree and select "Add program block".



2. In the pop-up window, set the name of the program block, select "Program", "Function block", or "Function" for the type, and select "Structured text (ST)" for the language, as shown in the figure below.



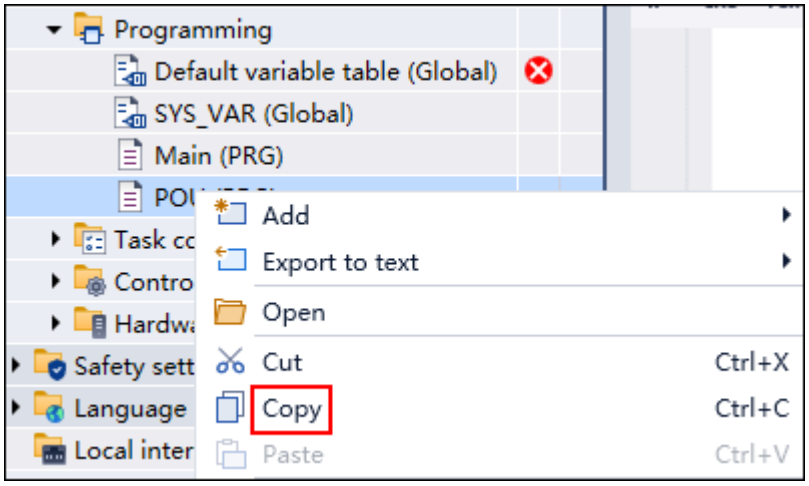
3. Click "OK" and the ST editor will be automatically opened.

10.1.4.2.2 Managing an ST Program Block

10.1.4.2.2.1 Copying an ST Program Block

ST program blocks can be copied and pasted within a PLC device, between PLC devices in the same project, or between different projects. The internal program logic of the ST program block remains unchanged after pasting.

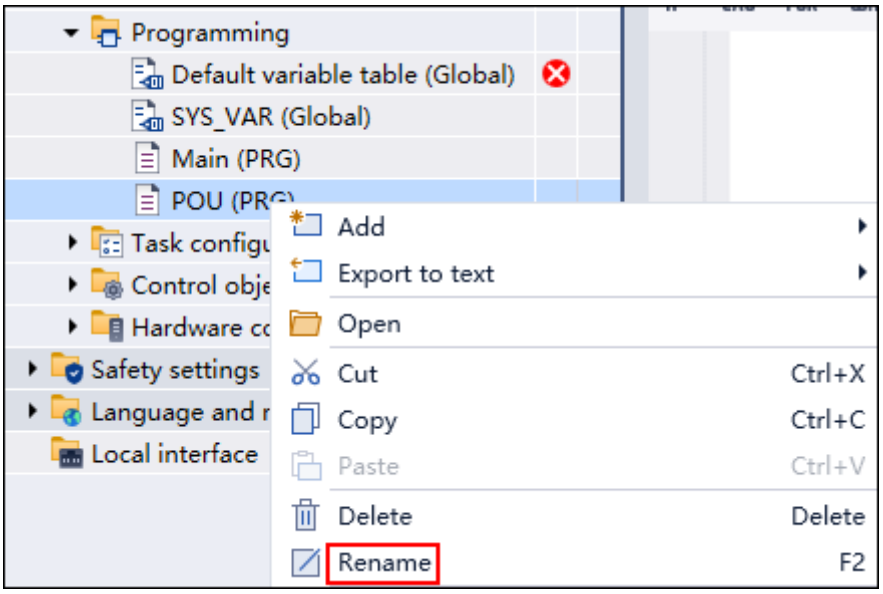
1. Right-click an ST program block name under the "Programming" node, and select "Copy".



2. Right-click the "Programming" node or its subordinate node, and select "Paste" to paste the copied LD program block at the corresponding location.

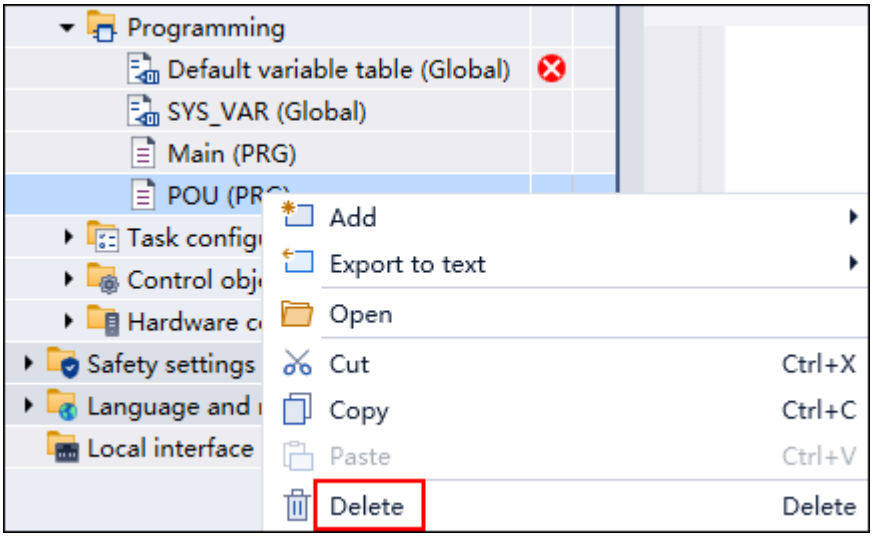
10.1.4.2.2.2 Renaming an ST Program Block

Right-click an ST program block name under the "Programming" node, select "Rename", and modify the name.



10.1.4.2.2.3 Deleting an ST Program Block

1. Right-click an ST program block name under the "Programming" node, and select "Delete".

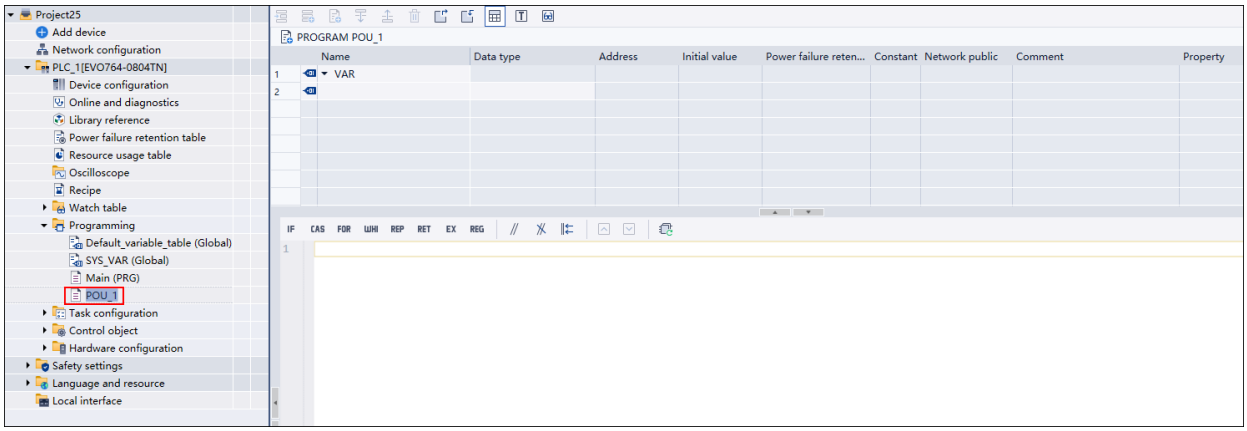


2. Click "OK" in the pop-up window.

10.1.4.3 ST Editor

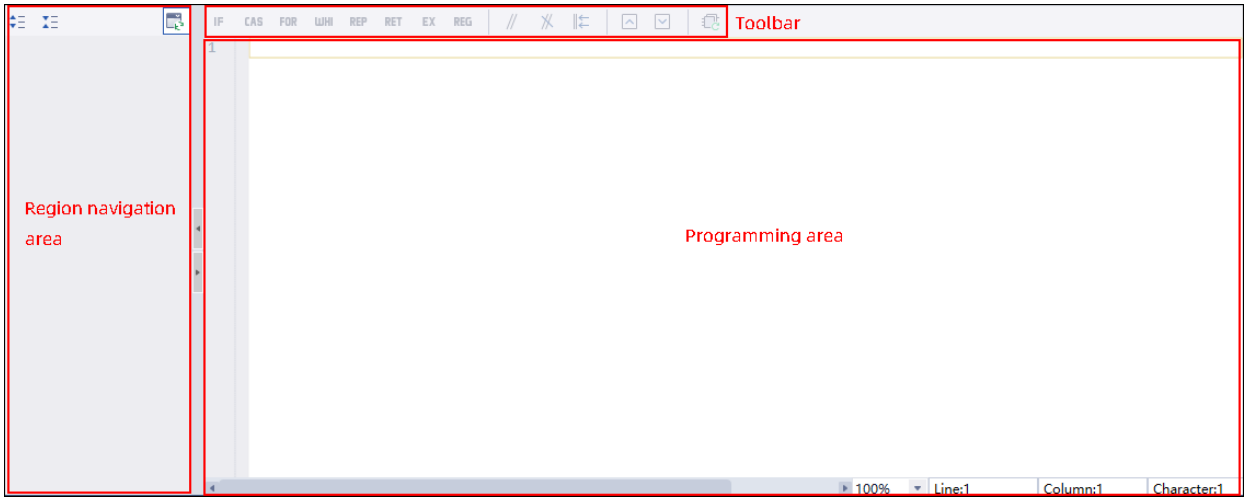
10.1.4.3.1 ST Editor Interface

Double-click the ST language program block under the "Programming" node (node identifier:) to open the ST editor.



The ST editor is divided into the variable declaration area and the program editing area, which can be collapsed and expanded using and .

The program editing area consists of three parts: the toolbar, the programming area, and the Region navigation area, as shown in the figure below:



- Toolbar

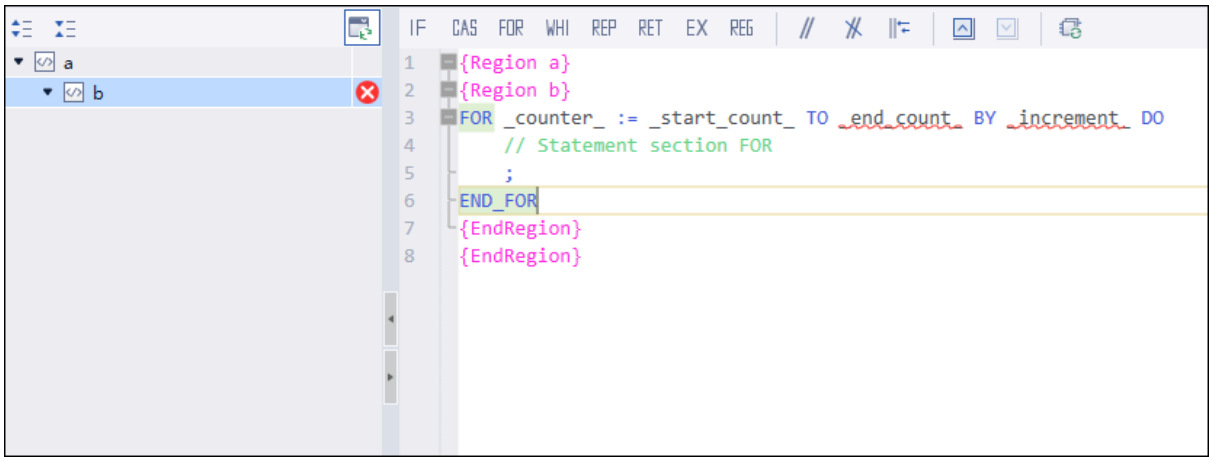
Description	Icon	Description
If	IF	Insert an IF statement.
Case	CAS	Insert a CASE statement.
For	FOR	Insert a FOR loop statement.
While	WHI	Insert a WHILE loop statement.
Repeat	REP	Insert a REPEAT loop statement.
Return	RET	Insert a RETURN instruction.
Exit	EX	Insert an EXIT statement.
Region	REG	Insert a Region instruction.
Comment	//	Insert a single-line comment.
Uncomment	//	Remove a single-line comment.
One-click formatting		Quickly reformat and optimize code layout.
Collapse all		Collapse logical structures that can be folded (such as functions, loops, conditional statements, etc.).
Expand all		Expand all collapsed content.
Update all parameters		Update all parameters in the program editing area if the parameters of the called object are changed.

- Programming area
  Used for ST programming. The bottom of the programming area displays the exact position of the mouse cursor, allowing users to quickly jump to any line in the program.

- Region navigation area
  Used to visually display structurally complete Region blocks in the ST program, helping users efficiently manage program structure and achieve rapid program positioning and navigation to target locations. Region navigation area consists of two columns: the first column contains the collapse/expand buttons and node names; the second column indicates whether the text associated with the node contains pre-compilation faults. Double-click the error flag to navigate to the corresponding code highlighted in the ST

program editing area.

Use and to collapse or expand the Region navigation area.



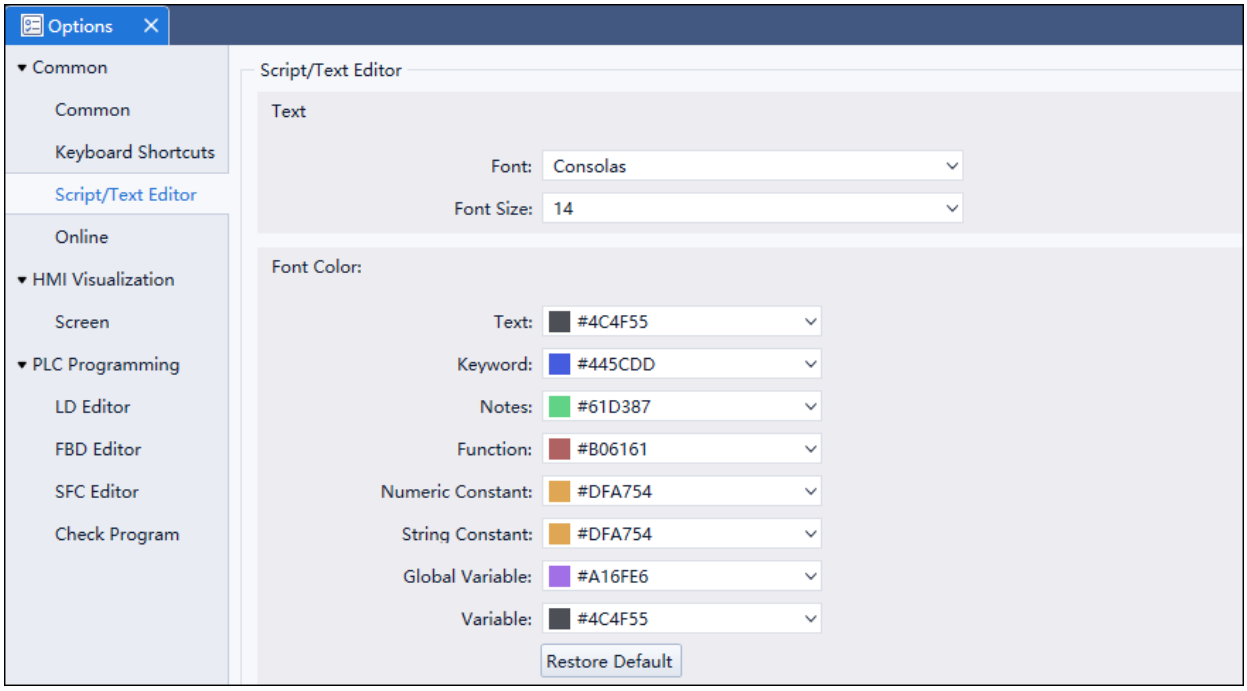
In this area, Regions can be copied, renamed, expanded/collapsed, and synchronized through the right-click menu or toolbar.

Function	Description
Copy	Copy all code within the selected Region block in the ST program editing area. The copied Regions can be pasted in the program editing area.
Rename	Rename the Region node and synchronously update the name of the corresponding Region node in the ST program editing area, and vice versa.
Expand/Collapse	Expand or collapse the selected Region node and its child nodes.
Expand all/ Collapse all	Expand or collapse all Region nodes and their child nodes, including all ST code block keywords.
Sync	◦ When synchronization is enabled, click a Region node on the left to navigate to the corresponding location in the ST program editing area; simultaneously, any changes to the Region structure, name, collapse state, or highlighted display in the editing area are synchronized to the Region nodes on the left, and vice versa. ◦ When synchronization is disabled, the collapse and select operations will not be synchronized between the ST program editing area and the Region navigation area. However, any changes to Region structure or name still sync between the two.

10.1.4.3.2 ST Editor Settings

In the software menu, select "Tools > Options". In the "Script/Text editor" tab, users can set the font, size, and color of the text.

- The settings in the text area will affect the character font and size in the ST program.
- The settings of font color will affect the color display of characters in the ST program. Click "Restore defaults" to reset all font colors to their default values.

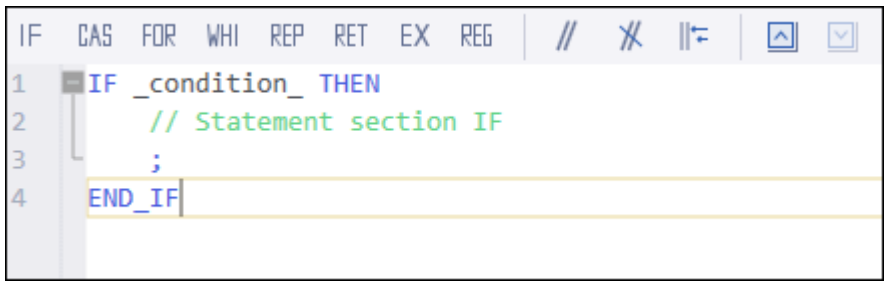


10.1.4.3.3 Basic Operations of ST Editor

Variables, operators, and statements in the program can be input in the following ways:

1. Input assistant
   At the insertion point in the editing area, press F3 to open the "Input assistant" dialog box, select the target object to insert it.
2. Direct input
   At the insertion point in the editing area, directly type the object. When characters are typed, the software automatically searches and matches them in the current project. Users can select the target variable from the result list. When operators are typed, the input method must be switched to English.
   When typing an array variable, enter "0" to trigger intelligent prompts for all sub-items of the current array dimension (cursor position). When array indices contain non-numeric values, non-static variables, or complex constant expressions (e.g., abc.d, ab[8]), the intelligent prompts for array sub-items cannot be accurately listed.
3. Drag-and-drop
   At the insertion point in the editing area, drag and drop the target object from the variable table or toolbox to insert it.
   When dragging a target variable from the variable table to the insertion point in the editor:
   - After dragging the target variable, if the mouse hovers over the insertion point for less than 1.5 seconds, the target variable will be directly inserted at the current position.
   - If the mouse hovers over the insertion point for 1.5 to 3 seconds, the system will draw a border around the variable at the insertion point. Releasing the mouse will perform an overwrite operation, replacing the hovered variable.
   - If the mouse hovers over the insertion point for more than 3 seconds, the system will draw a border around the full path of the hovered variable. Releasing the mouse will also perform an overwrite operation of the full path of the hovered variable.
4. Adding a statement

At the insertion point in the editing area, select the statement icon from the toolbar to insert the corresponding keyword.



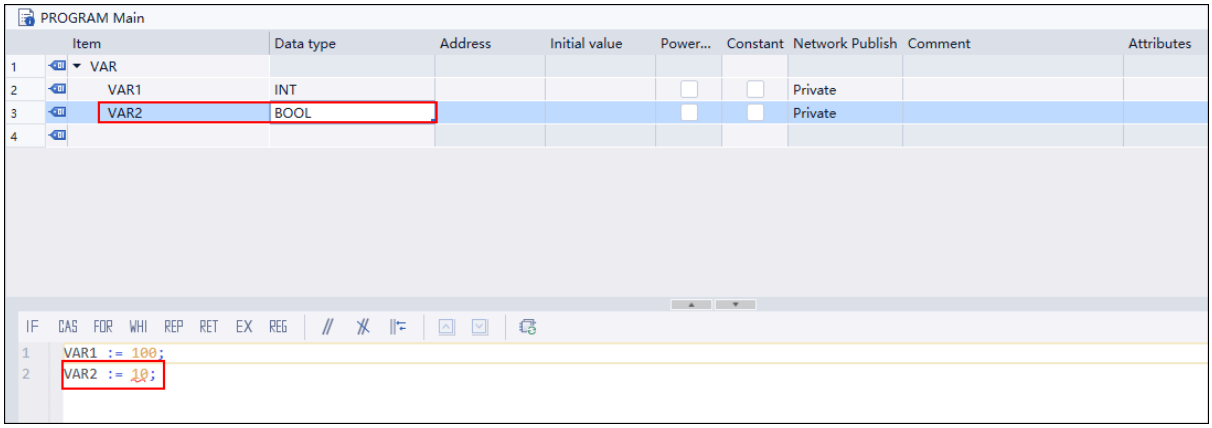
The ST Editor supports the following functions:

① Copy, paste, cut, delete, undo, and redo

These functions are similar to ordinary text. The undo function in the ST editor undoes one action at a time based on user input.

② Indentation with Shift+Tab

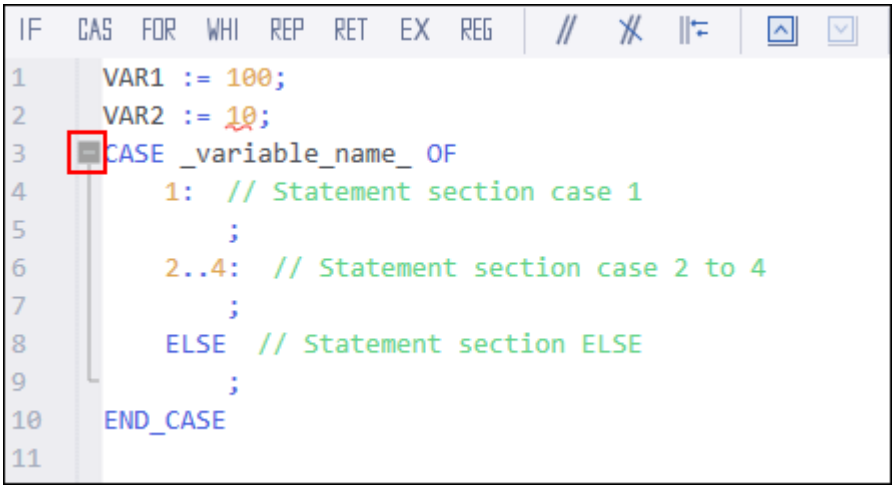
③ The editing area supports syntax highlighting, with red wavy lines for wrong syntax and yellow highlighting for warning syntax.



④ Supports automatic completion of block statements. Parameters for FB instances, FC, PRG, or Meth called within the editing area can be auto-completed using the Tab key.

5. Collapsing/Expanding

ST supports code collapsing and expanding code segments. The icon is shown in the following figure.

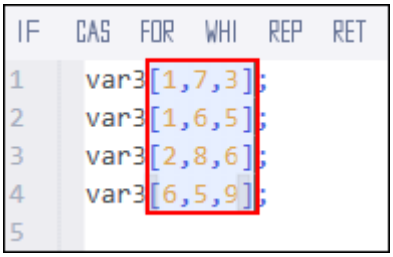


- Collapsing and expanding are supported in both online and offline statuses.
- Click the icon to collapse or expand the block.

- Click the "Collapse all" icon in the ST toolbar or use the shortcut key "Alt + F" to collapse all code segments. Click the "Expand all" icon in the ST toolbar or use the shortcut key "Alt + U" to expand all code segments.
- The collapsed area include block statements, annotations, and Regions.
- There are collapsing symbols and range lines in the collapsed area.

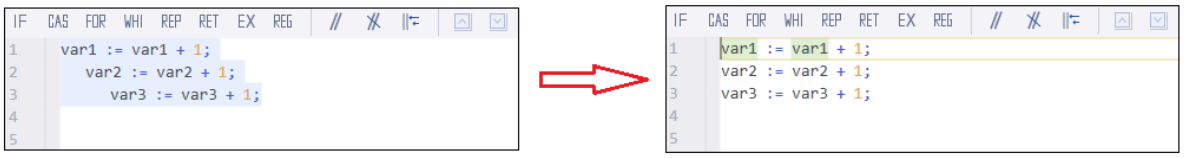
6. Column operations

   In the ST editor, hold down the Alt key and drag the left mouse button to select specific columns across multiple lines. Users can edit, batch delete, copy, and paste the selected columns.



7. One-click formatting

   Users can select the area to be formatted and click the "One-click formatting" icon in the ST toolbar or use the shortcut key "Ctrl + Shift + W" to quickly optimize the code format.



> **NOTE**
>
> The ST editor also supports automatic text formatting. When users enter a semicolon (;), the editor will automatically format the code according to predefined rules, ensuring orderly and standardized code.

8. Updating all parameters

   The ST editor supports one-click updating of called instruction parameters. Users can click the "Update all parameters" icon in the ST toolbar to quickly update all parameters of the instructions within the current editor program.

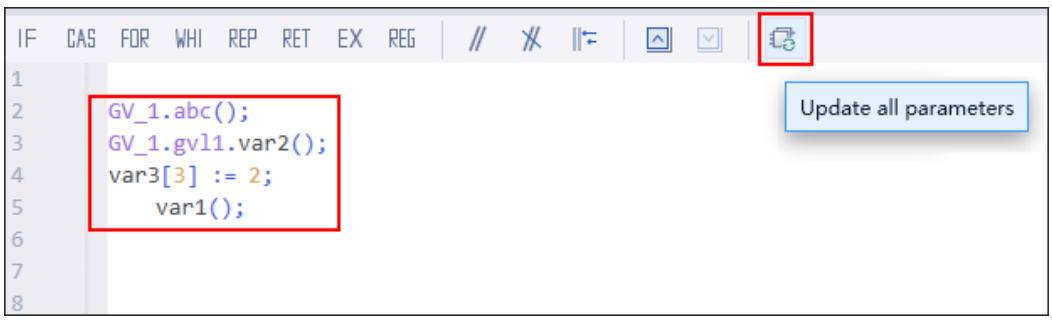


Figure 10-1-4-3-3-1 Updating all parameters

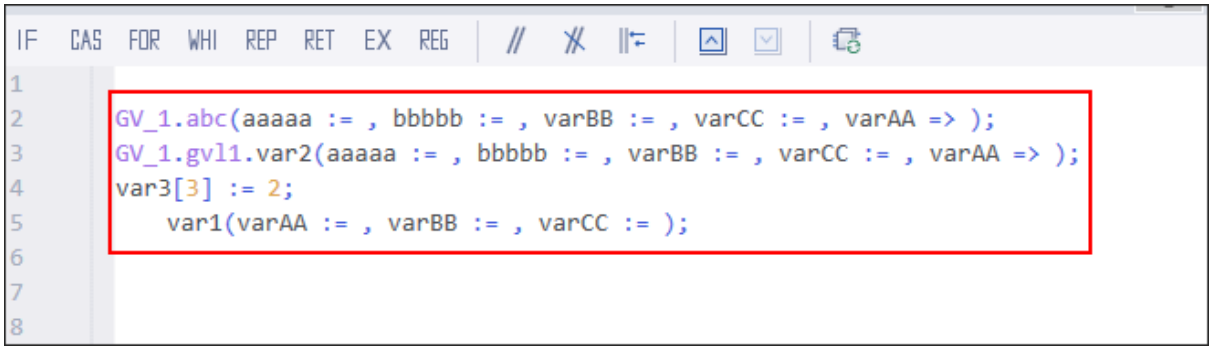
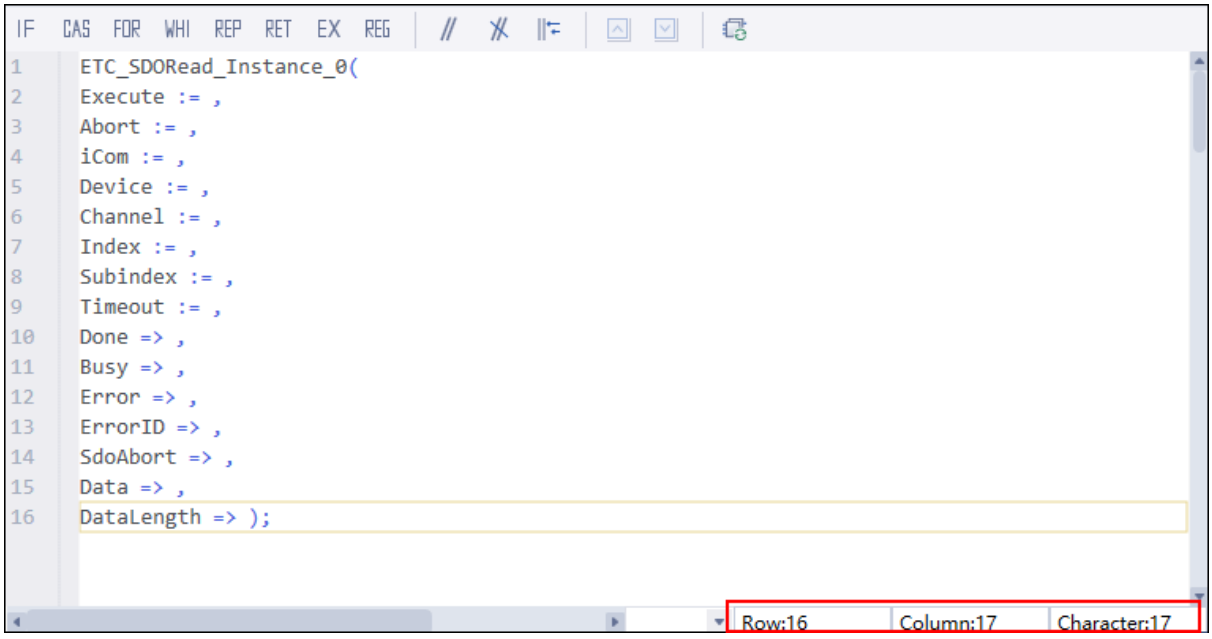


Figure 10-1-4-3-3-2 Update results

9. Line jumping

Line jumping allows users to quickly jump to any line in the program.

The bottom of the editing area displays the exact position of the current mouse. Users can double-click the line number or use the shortcut key Ctrl+G to open a pop-up window. Enter the target line number and click "OK" to jump to the specified line.



10.1.4.4 ST Operators

If there are several operators in an expression, the operators will be executed according to the predefined precedence: operators with higher precedence are executed first, followed by those with lower precedence. If there are operators with the same precedence in an expression, they are executed from left to right in the order they are written. The precedence of operators is shown in the table:

Operation	Operator	Example	Priority
Parentheses	()	(2 + 3) * (5 - 4) The result is 5.	1
Function call	Function name (Parameter list)	FUN2 (FUN1 (Var1A, Var1B) , Var2B) FUN1 is executed first, then FUN2.	2

Operation	Operator	Example	Priority

Truncation	TRUNC
	TRUNC_INT
		OUT := TRUNC (1.234); // Returns the result 1 of the data type DINT.
		OUT := TRUNC_INT (2.456); // Returns the result 2 of the data type INT.
			2

Inversion	NOT	NOT TRUE The result is FALSE.	3
Exponentiation	EXPT	OUT:= EXPT (R1 , R2); If R1 = 5 and R2 = 4, OUT is 625.	4
Multiplication	*	OUT:= R1 * R2; If R1 = 5 and R2 = 4, OUT is 20.	5
Division	/	OUT := R1 / R2; If R1 = 5 and R2 = 4, OUT is 1. // When the data type of OUT is DWORD, the result is 1.	5
Modulus	MOD	OUT := R1 MOD R2; If R1 = 9 and R2 = 4, OUT is 1.	5
Addition	+	OUT := R1 + R2; If R1 = 5 and R2 = 4, OUT is 9.	6
Subtraction	-	OUT := R1 - R2; If R1 = 5 and R2 = 4, OUT is 1.	6
Compare	<, >, <=, >=	OUT := R1 < 9; If R1 = 8, OUT is TRUE.	7
Equal to	=	OUT := R1 := 8; OUT is 8.	8
Not equal	<>	OUT := R1 <> 7; If R1 = 8, OUT is TRUE.	8
Logic AND	AND, &	OUT := B1 & B2; If B1 = TRUE and B2=TRUE, OUT is TRUE.	9
Logical XOR	XOR	OUT := B1 XOR B2; If B1 = FALSE and B2=TRUE, OUT is FALSE.	10
Logic OR	OR	OUT := B1 OR B2; If B1 = TRUE and B2=TRUE, OUT is TRUE.	11

10.1.4.5 ST Keywords

Keyword	Meaning	Example
:=	Assignment	d:=10;
S=	Set a bit	Var1: BOOL; Var2: BOOL; Var1 S= Var2;
R=	Reset a bit	Var1: BOOL; Var2: BOOL; Var1 R= Var2;
RETURN	Return	-

IF	Select	D:=B*B;
		IF D<0.0 THEN
		C:=A;
		ELSIF D=0.0 THEN
		C:=B;
		ELSE
		C:=D;
		END_IF;

CASE	Multiple selection	CASE INT1 OF
		1: g:= 11;
		2: g:= 22;
		ELSE
		g:= 5;
		END_CASE;

FOR	FOR loop	FOR I:=1 TO 100 BY 2 DO J:=J*2; END_FOR;
WHILE	WHILE loop	WHILE J<= 100 DO J:=J+2; END_WHILE;

Keyword	Meaning	Example
AND_THEN	AND operation	Var1: BOOL; Var2: BOOL; Var3: BOOL; Var3 := Var1 AND_THEN Var2;
OR_ELSE	OR operation	Var1: BOOL; Var2: BOOL; Var3: BOOL; Var3 := Var1 OR_ELSE Var2;
REPEAT	REPEAT loop	REPEAT Var1:=Var1*2; Counter:=Counter-1; UNTIL Counter=0; END_REPEAT;

EXIT	Exit loop	FOR I:=1 TO 100 DO
		FOR J:=1 TO 10 DO
		IF Val[I,J]>100 THEN EXIT;
		END_IF;
		END_FOR;
		END_FOR;

;	Empty statement	WHILE Var<0 DO ;(*Empty statement*) END_WHILE;
(* Text *)	Annotation	Motor_A.Nominal_Voltage:=380; (*Voltage 380V*)
// Text	Annotation	Execute:=RUNF1, // Triggers execution flag

10.1.4.6 ST Statements

- Assignment statement
  Assignment statement (represented by ":=") is one of the most commonly used statements in structured text. It is used to assign the value produced by the expression on its right side to the operand (variable or address) on its left side. The specific format is as follows.

```
<variable>:=<expression>;
```

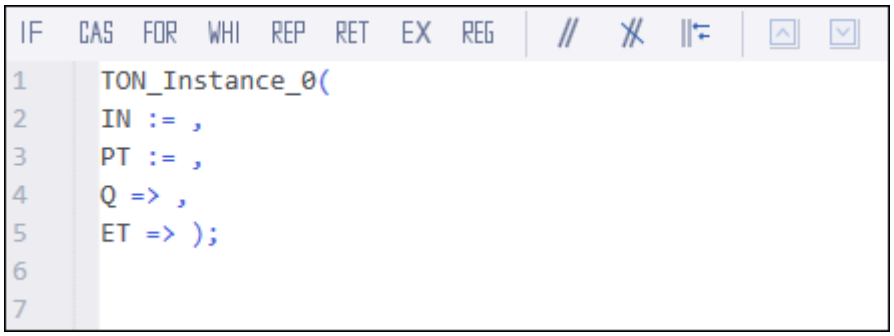
The following example assigns a value to an INT type variable var1.

```
var1:=100;
```

- Function and function block calls
  Function blocks can be called by instantiating their names, with the specific format as follows.

```
Function block instance name: (function block parameters);
```

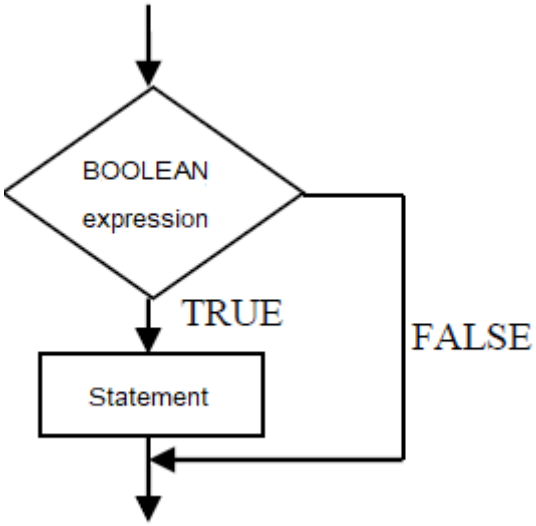
To call a function block in ST, directly enter the instance name of the function block and assign values or variables to each parameter of the function block in the subsequent parentheses, separated by commas. The function block call ends with a semicolon. The following example calls the TON timer function block (instance name: TON_Instance_0) in ST.



- Selection statements
  - IF statement
    The IF statement is used to implement a single-branch selection structure, with the basic format as follows.

```
IF <Boolean expression> THEN
<statement content>;
END_IF
```

If the above format is used, the statement content is only executed when the <Boolean expression> is TRUE; otherwise, the statement content is not executed. The statement content can be a single statement, an empty statement, or multiple parallel statements. The execution flowchart of the statement expression is shown in the figure below.



The following example shows how to use a PLC to determine whether the current temperature exceeds 65°C. If it does, the program will turn on the fan for cooling. The specific code is as follows.

```
VAR
nowTemp:BYTE; (*current temperature status signal*)
bFan:BOOL; (*fan control signal*)
END_VAR
```

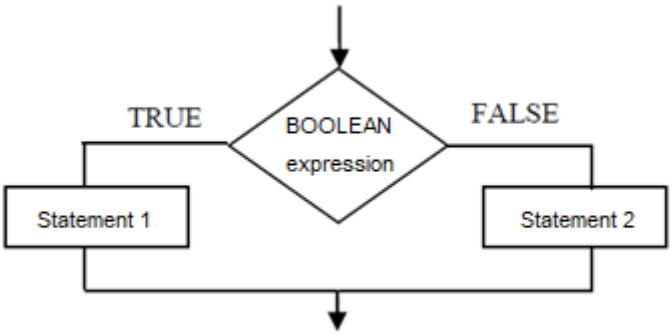
```
nowTemp:=75;
IF nowTemp>60 THEN
bFan:=TRUE;
END_IF
```

- IF...ELSE statement

  The IF...ELSE statement is used to implement a dual-branch selection structure, with the basic format as follows.

```
IF <Boolean expression> THEN
<statement content 1>;
ELSE
<statement content 2>;
END_IF
```

The above expression first evaluates the value within the <Boolean expression>. If it is TRUE, <statement content 1> is executed; if it is FALSE, <statement content 2> is executed. The program execution flowchart is shown in the figure below.



The following example shows how to use a PLC to determine whether the current temperature exceeds 65°C. If it does, the program will turn on the fan for cooling; otherwise, the fan is turned off.

```
VAR
nowTemp:BYTE; (*current temperature status signal*)
bFan:BOOL; (*fan control signal*)
END_VAR
```

```
IF nowTemp>65 THEN
bFan:=TRUE;
ELSE
bFan:=FALSE;
END_IF
```

When there is more than one conditional statement, a nested IF...ELSE statement is needed. It is a multi-branch selection structure and the basic format is as follows.

```
IF <Boolean expression 1> THEN
  IF <Boolean expression 2> THEN
    <statement content 1>;
  ELSE
    <statement content 2>;
  END_IF
ELSE
 <statement content 3>;
END_IF
```

With the following code, For equipment in automatic mode, the fan is turned on and the heater is turned off when the actual temperature is greater than 50°C. If it is less than or equal to 50°Cs, the fan is turned off and the heater is turned on. For equipment in manual mode, neither the heater nor the fan is turned on.

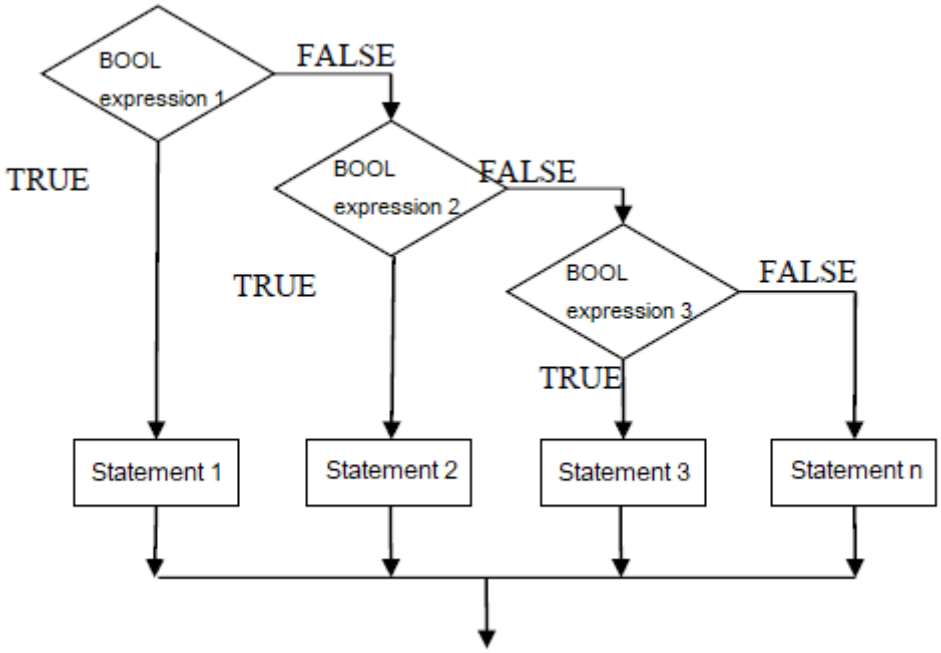
```
VAR
bAutoMode: BOOL; (*manual/auto mode signal*)
nTemp:BYTE; (*current temperature status signal*)
bFan:BOOL; (*fan control signal*)
bHeating:BOOL; (*heater control signal*)
END_VAR
```

```
IF bAutoMode=TRUE THEN
  IF nTemp>50 THEN
    bFan:=TRUE;
    bHeating:=FALSE;
  ELSE
    bFan:= FALSE;
    bHeating:= TRUE;
  END_IF
ELSE
 bFan:= FALSE;
 bHeating:=FALSE;
END_IF
```

- The multi-branch selection structure can also be implemented wit the IF..ELSIF..ELSE statement. The specific format is as follows.

```
IF <Boolean expression 1> THEN
  <statement content 1>;
ELSIF <Boolean expression 2> THEN
  <statement content 2>;
ELSIF <Boolean expression 3> THEN
  <statement content 3>;
. . .
. . .
ELSE
  <statement content n>;
END_IF
```

If the <Boolean expression1> is TRUE, only <statement content1> is executed. Otherwise, judgments are made from the <Boolean expression2> until one of the Boolean expressions is TRUE, and then the corresponding statement content for that Boolean expression is executed. If none of the Boolean expressions are TRUE, only the <statement content n> is executed. The program execution flowchart is shown in the figure below.



◦ CASE statement

The CASE statement is a multi-branch selection statement that selects one branch to execute from multiple branches based on the value of an expression, and the basic format is as follows.

```
CASE <condition variable> OF
  <value 1>: <statement content 1>;
  <value 2>: <statement content 2>;
  <value 3, value 4, value 5>: <statement content 3>;
  <value 6 .. value 10>: <statement content 4>;
  ...
  <value n>: <statement content n>;
ELSE
  <ELSE statement>;
END_CASE;
```

Execution method of CASE statement:

- If the value of <condition variable> is <value i>, the <statement content i> is executed.
- If <condition variable> has no specified value, the < ELSE statement> is executed.
- If the same instruction needs to be executed for several values of <condition variable>, these values can be written consecutively and separated by commas. This ensures the common instruction is executed, as shown in line 4 of the program above.
- If the same instruction needs to be executed for the values of <condition variable> within a certain range, the initial and final values of the range can be entered and separated by two dots. This ensures the common instruction is executed, as shown in line 5 of the program above.

- Iteration statement

  The iteration statement is mainly used for programs that need to be repeated. Common iteration statements include FOR, REPEAT, and WHILE.

  ◦ FOR loop

The FOR loop statement first evaluates an initialization sequence. If the loop condition is TRUE, it repeatedly executes the nested statements and calculates the iteration expression sequence. If the loop condition is FALSE, the loop terminates. The specific format is as follows.

```
FOR <variable> := <initial value> TO <target value> {BY <step>} DO
  <statement content>
END_FOR;
```

The execution order of the FOR loop is as follows:

- Calculate whether <variable> is within the range of <initial value> and <target value>.
- When <variable> is less than <target value>, execute <statement content>.
- When <variable> is greater than <target value>, <statement content> will not be executed.
- Each time <statement content> is executed, <variable> is incremented by the specified step size. The step size can be any integer value. If the step size is not specified, its default value is 1. When <variable> is greater than <target value>, exit the loop.

The following example uses the FOR loop to implement increment-by-1 calculation.

```
VAR
Counter: BYTE; (*loop counter*)
Var1:WORD; (*output result*)
END_VAR
```

```
FOR Counter:=1 TO 5 BY 1 DO
  Var1:=Var1+1;
END_FOR;
```



If <target value> equals the limit of <variable>, it will enter an infinite loop. Assuming the variable Counter in the above example is of type BYTE (0 to 255), setting <target value> to 255 will cause the controller to enter an infinite loop. Therefore, the <target value> should not be set to the limit of <variable>.

◦ WHILE loop

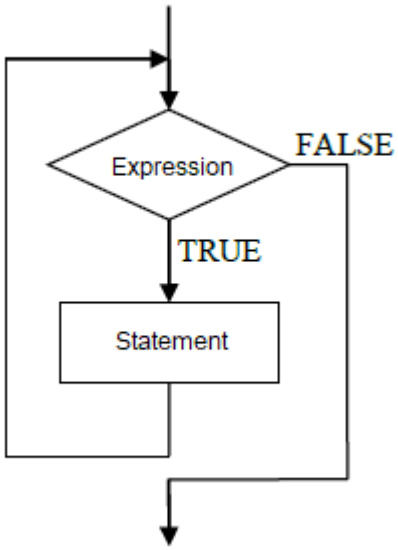
The WHILE loop is similar to the FOR loop in usage. The difference is that the termination condition of the WHILE loop can be any logical expression. It can be a condition. When the condition is met, the loop is executed. The specific format is as follows.

```
WHILE <Boolean expression>
  <statement content> ;
END_WHILE;
```

The execution order of the WHILE loop is as follows:

- Calculate the return value of <Boolean expression>.
- When the value of <Boolean expression> is TRUE, repeatedly execute <statement content>.

- If the initial value of <Boolean expression> is FALSE, the <statement content> will not be executed, and the program will jump to the end of the WHILE statement. The flowchart is shown in the following figure.



CAUTION

If the value of <Boolean expression> is always TRUE, an infinite loop will occur. It can be avoided by changing the condition of the loop statement. For example, an incrementing or decrementing counter can be used to avoid an infinite loop.

The following example shows that as long as Counter is not zero, the program within the loop body will be executed continuously.

```
VAR
Counter: BYTE; (*counter*)
Var1:WORD;
END_VAR
```

```
WHILE Counter<>0 DO
  Var1 := Var1+2;
  Counter := Counter-1;
END_WHILE
```

- The difference between a REPEAT loop and a WHILE loop is that the REPEAT loop checks the termination condition only after the statement content has been executed. This means that the loop will be executed at least once regardless of the termination condition.

```
REPEAT
  <statement content>
UNTIL
  <Boolean expression>
END_REPEAT;
```

The execution order of the REPEAT loop is as follows:

- When the value of <Boolean expression> is FALSE, execute <statement content>.
- When the value of <Boolean expression> is TRUE, stop executing <statement content>.
- After the first execution of <statement content>, if the value of <Boolean expression> is TRUE, then <statement content> is only executed once.



If the value of <Boolean expression> is always TRUE, an infinite loop will occur. It can be avoided by changing the condition of the loop statement. For example, an incrementing or decrementing counter can be used to avoid an infinite loop.

When the counter value is 0, the loop stops.

```
VAR
Counter: BYTE;
END_VAR
```

```
REPEAT
  Counter := Counter+1;
UNTIL
  Counter=0
END_REPEAT;
```

The result is that the REPEAT loop is entered in each program cycle. With Counter being a BYTE (0 to 255), 256 calculations are performed in each cycle.

Each time the REPEAT statement is entered, Counter is initially set to 1, and the statement content Counter := Counter + 1 is executed 256 times in each cycle until the Counter variable overflows to 0, at which point the loop exits. The variable then overflows again after being incremented repeatedly, repeating this process.

- Jump statement
  - EXIT statement

    The EXIT statement can be used in FOR, WHILE, and REPEAT loops. The inner loop will immediately stop regardless of the termination condition. The specific format is as follows.

    ```
    EXIT;
    ```

    The following example uses the EXIT statement to avoid division by zero in iterative statements.

    ```
    FOR Counter:=1 TO 5 BY 1 DO
    INT1:= INT1/2;
    IF INT1=0 THEN
    EXIT; (*To avoid division by zero*)
    END_IF
    Var1:=Var1/INT1;
    END_FOR
    ```

When INT1 equals 0, the FOR loop ends.

- CONTINUE statement

  This instruction is an extension of the IEC 61131-3 standard. The CONTINUE statement can be used in FOR, WHILE, and REPEAT loops.

  It interrupts the current loop, ignoring the code after it and starting a new iteration immediately. The specific format is as follows.

```
CONTINUE;
```

  The following example uses the CONTINUE statement to avoid division by zero in iterative statements.

```
VAR
Counter: BYTE; (*loop counter*)
INT1,Var1: INT; (*intermediate variable*)
Erg: INT; (*output result*)
END_VAR
```

```
FOR Counter:=1 TO 5 BY 1 DO
   INT1:= INT1/2;
 IF INT1=0 THEN
      CONTINUE; (*To avoid division by zero*)
  END_IF
   Var1:=Var1/INT1; (*Executed only when INT1 is not 0*)
END_FOR;
Erg:=Var1;
```

- JMP statement

  The JMP statement can be used for unconditional jumps to lines of code marked with a label. The specific format is as follows.

```
<label>:
.
JMP <label>;
```

  <label> can be any identifier, placed at the beginning of a line of code. The JMP statement is followed by a pre-defined label, indicating the target jump position. When the JMP statement is executed, the program jumps to the line corresponding to the label.



Infinite loops can be avoided by using the JMP statement with IF conditions to control jumps.

The following example uses the JMP statement to implement a counter looping within the range of 0 to 10.

```
VAR
nCounter: BYTE;
END_VAR
```

```
Label1:nCounter:=0;
Label2:nCounter:=nCounter+1;
IF nCounter<10 THEN
  JMP Label2;
ELSE
  JMP Label1;
END_IF
```

In the above example, Label1 and Label2 are labels, not variables, so they do not need to be declared as variables in the program.

The IF statement is used to determine if the counter is within the range of 0 to 10. If it is, the JMP Label2 statement is executed, and the program jumps to the Label2 position in the next cycle to execute the program nCounter:=nCounter+1, incrementing the counter by 1. Otherwise, the program jumps to the Label1 position and execute nCounter:=0, resetting the counter to zero.

- RETURN statement
  The RETURN statement is used to exit a program organization unit (POU). The specific format is as follows.

```
RETURN;
```

The following example uses an IF statement for condition check. When the condition is met, the program is immediately terminated.

```
VAR
nCounter: BYTE;
bSwitch: BOOL; (*switch signal*)
END_VAR
```

```
IF bSwitch=TRUE THEN
  RETURN;
END_IF;
nCounter:= nCounter +1;
```

If bSwitch is FALSE, nCounter always increments by 1. If bSwitch is TRUE, nCounter retains its value from the previous cycle and immediately exits the POU.

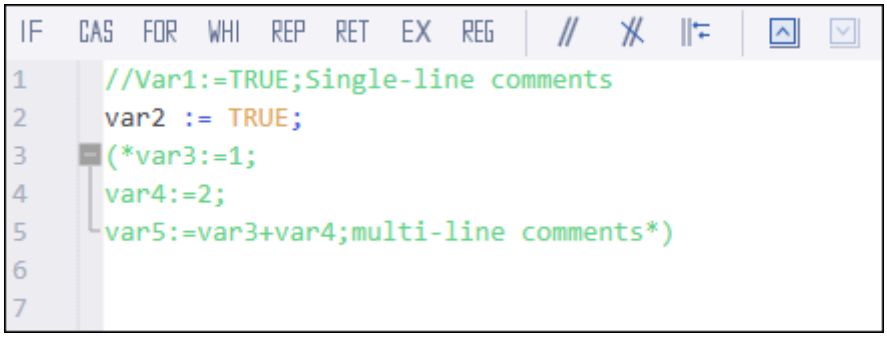
- Empty statement
  An empty statement does not execute any content. The specific format is as follows.

```
;
```

- Comment
  Annotations are an important part of the program, making it more readable without affecting its execution. They can be added anywhere in the declaration or execution sections of the ST editor.

  Annotations can be added with the following two methods in ST programming:

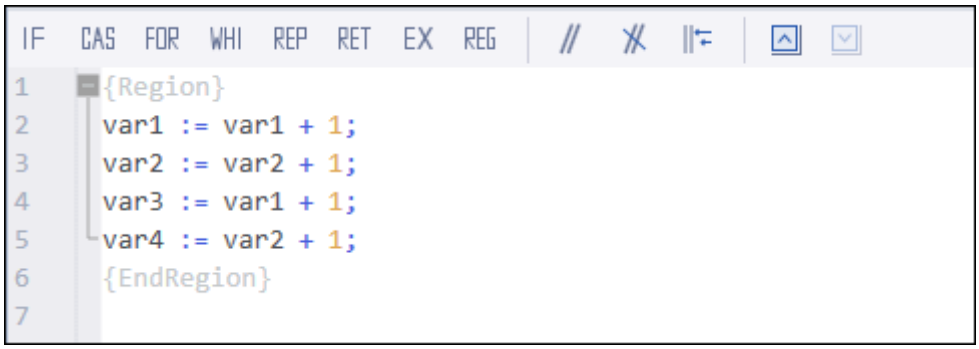
  - Single-line annotations start with "//" and continue until the end of the current line, as shown in the following figure. The ST editor also supports the shortcut "Ctrl + /" to add a single-line annotation, and "Ctrl + ." to remove a single-line annotation.
  - Multi-line annotations start with (* and end with *), as shown in the figure.

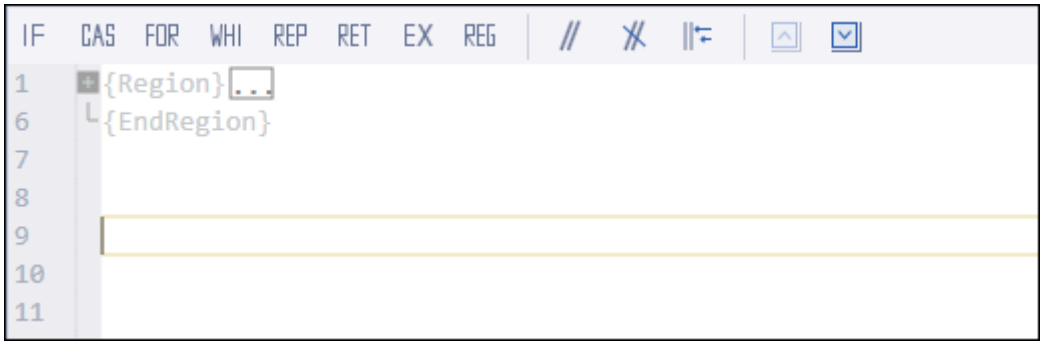


- Region statement

  The Region compilation statement is used to group multiple lines into a block in the text editor. The block can be named. The Region statement supports nesting, and can be used in the ST editor and all declaration editors.

  The expanded and collapsed views of code with the Region statement are shown in the figure below.





The syntax format of the Region statement is as follows:

```
{Region} or {Region "name"}
//Code
//Code
{EndRegion}
```

> **NOTE**
>
> When naming a Region block, the name must be separated from the Region start marker by at least one space, no consecutive characters are allowed. The name is case-insensitive and can be "Region".

10.1.5 CFC Language Programming

10.1.5.1 CFC Language Overview

Continuous function chart (CFC) is a graphical programming language, an extension of the IEC 61131-3 standard. It describes control logic graphically. By customizing the names and execution order of each block, it facilitates the implementation of process calculations as well as the assignment of tasks to programs and function blocks, enabling intuitive design and implementation of complex control systems.

The design and implementation of the CFC language involve the design of signal acquisition circuits, signal conversion, and data processing software workflows, playing a crucial role in industrial automation systems.

10.1.5.2 CFC Creation and Management

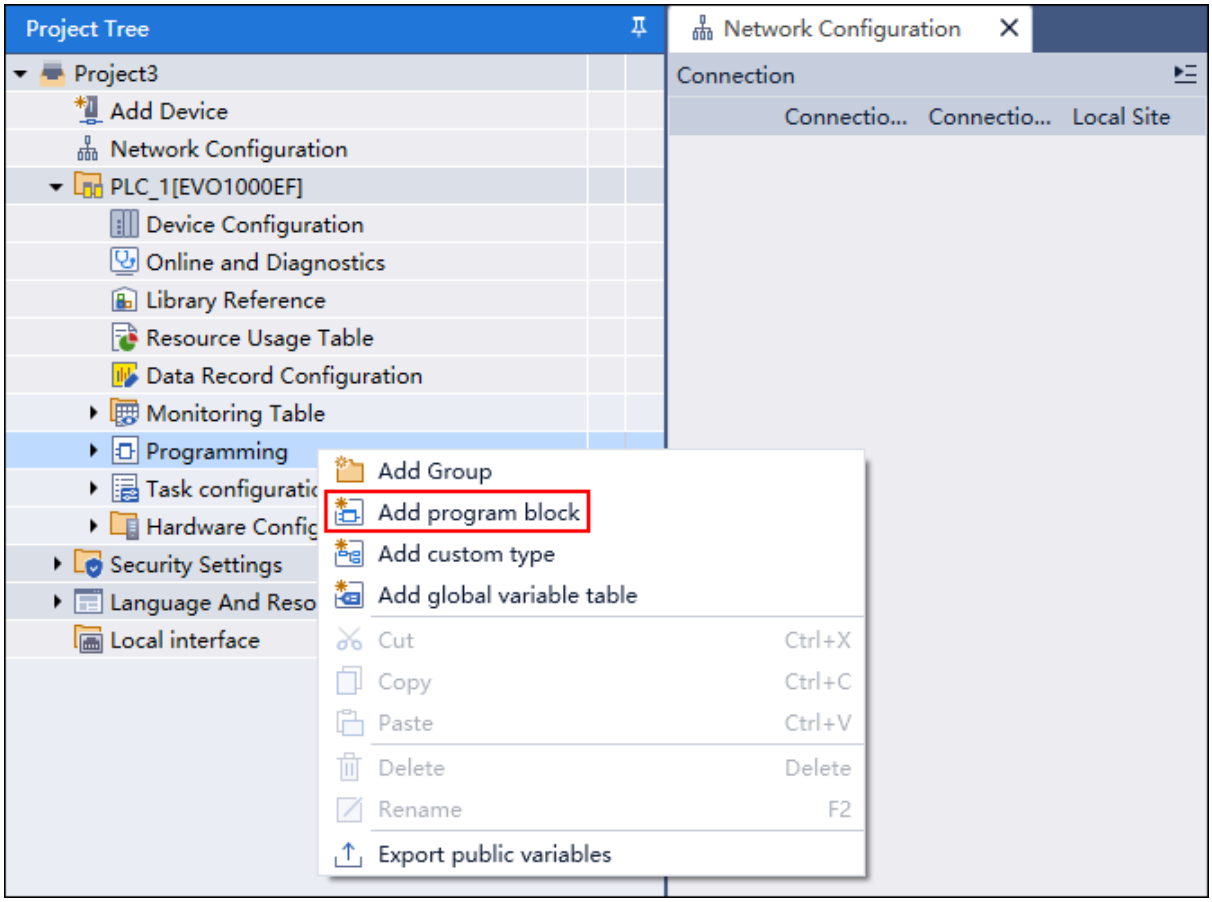
10.1.5.2.1 Creating a CFC Program Block

Prerequisites

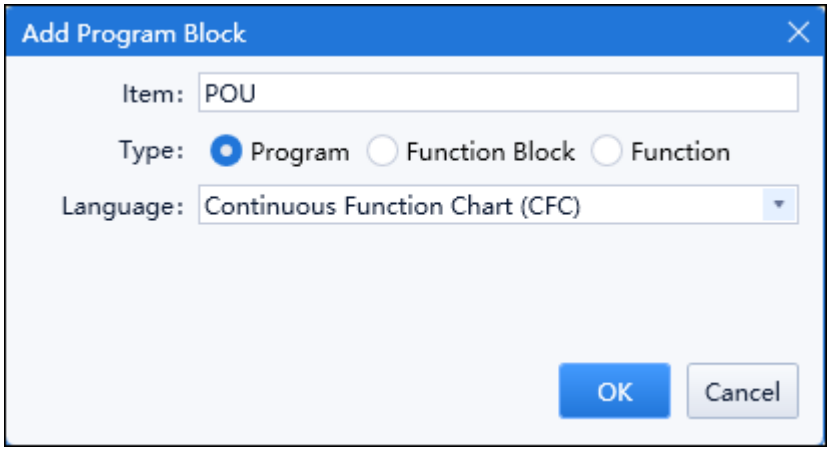
- A project has been created.
- A device has been added.

Procedure

1. Right-click "Programming" in the project tree and select "Add program block".



2. In the pop-up window, set the name of the program block, select "Program" for the type, and select "Continuous function chart (CFC)" for the IEC language, as shown in the figure below.



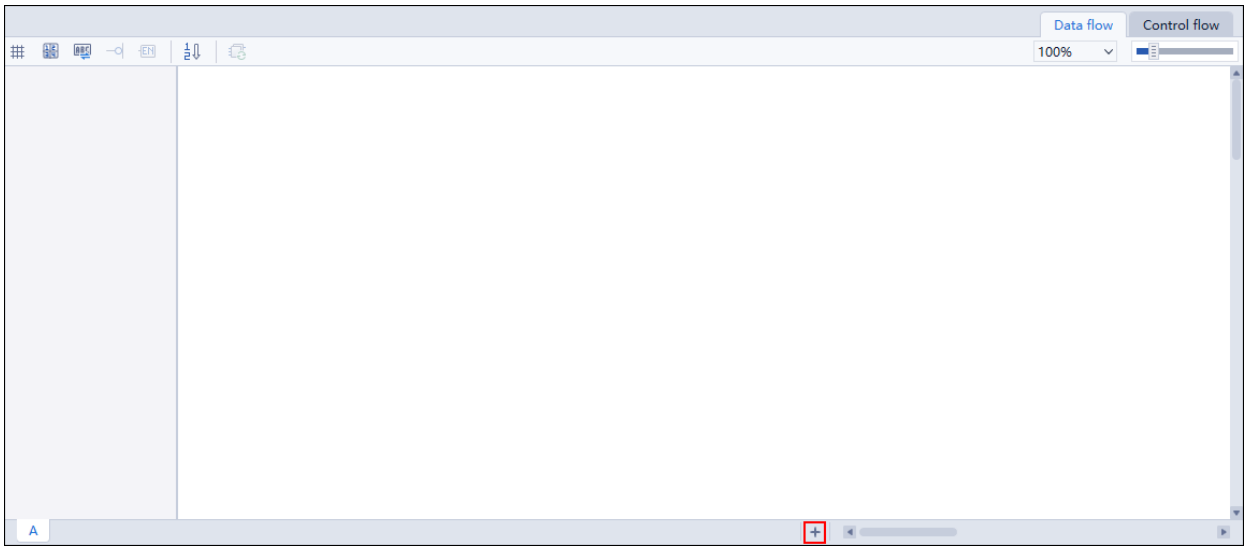
3. Click "OK" and the CFC editor will be automatically opened.

10.1.5.2.2 Creating a CFC Partition

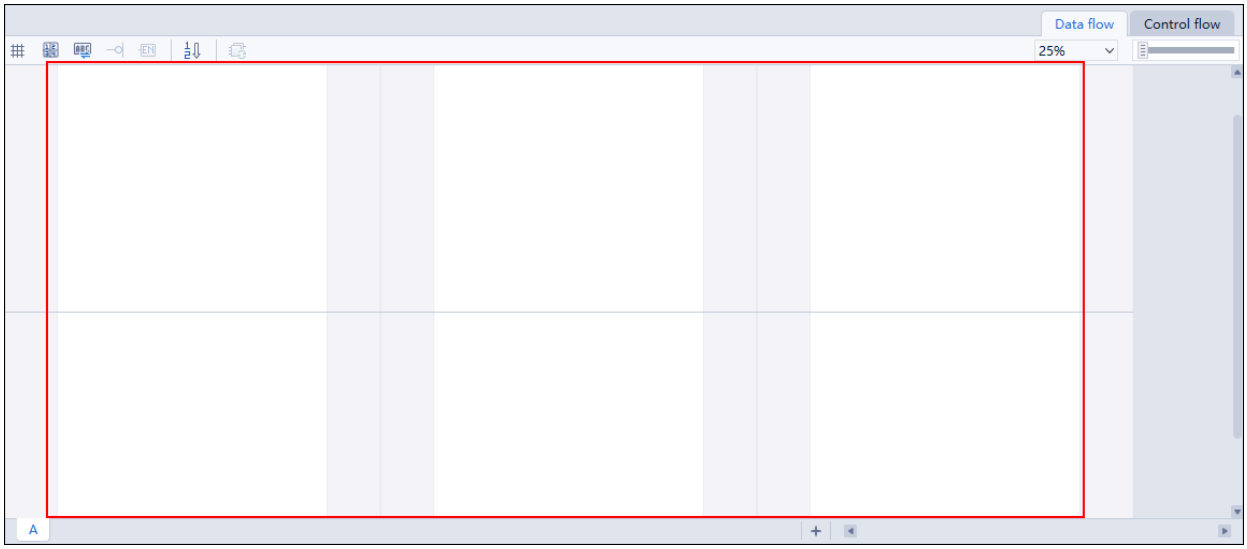
A CFC program block has a partition (with the default name A) by default. Partitions can be added as needed, with a maximum of 26 partitions.

Adding a partition

In the CFC editor, click + in the partition operation bar to add a partition. New partitions are named sequentially as A, B, C, D, etc.



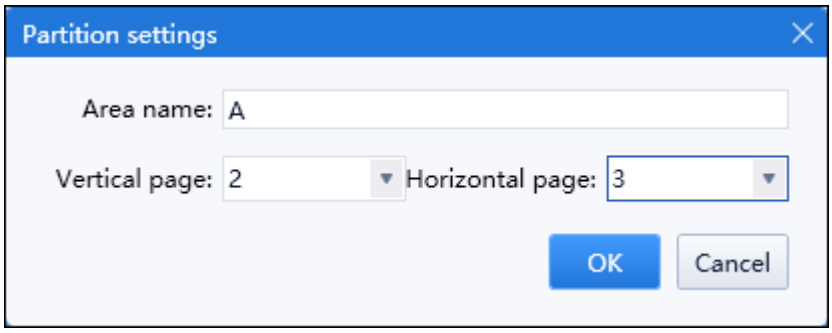
A partition has 6 sub-pages by default, as shown in the figure below. Each sub-page is an independent editing area and does not support adding or deleting.



Partition operations

- Partition settings

  1. Right-click the partition name and select "Partition setting".
  2. In the pop-up window, set the partition name and vertical/horizontal page, as shown in the figure below.



Parameter	Description
Partition name	Customizes the partition name.
Vertical/Horizontal page	Adjusts the sub-page layout. The default sub-page layout is 2*3 (2 rows and 3 columns), meaning the vertical page count is 2 and the horizontal page count is 3. Supported configurations include: 2*3, 3*2, 1*6, 6*1.

- Deleting a partition

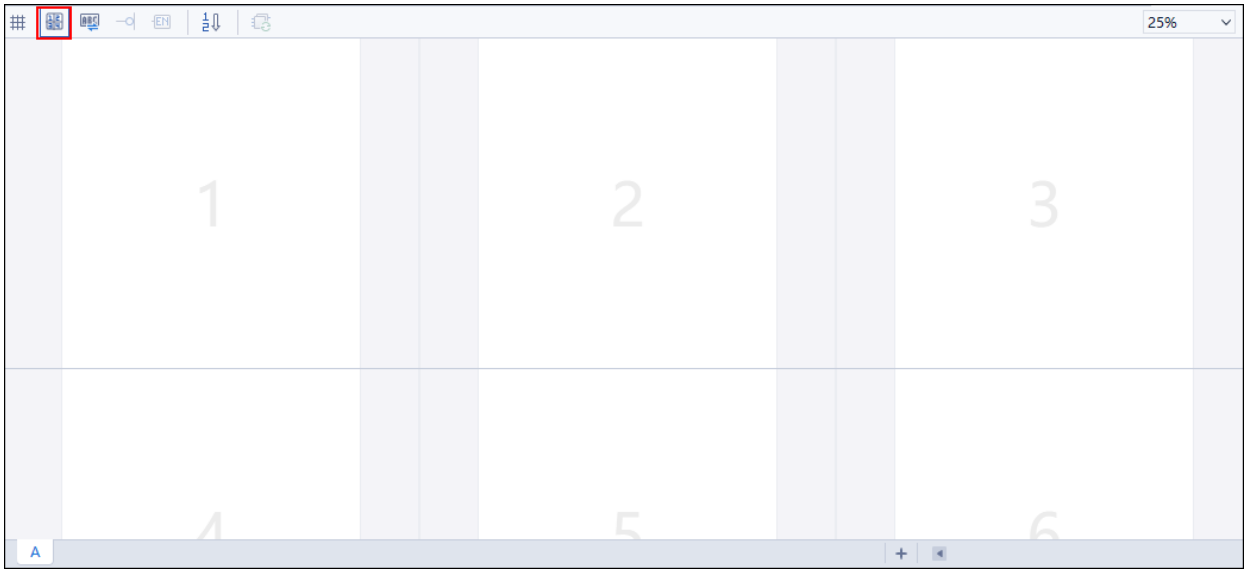
  1. Right-click the partition name and select "Delete partition."
  2. In the pop-up window, click "Yes" to delete the partition.

- Switching view

  Double-click the blank space to the left or right of each sub-page to switch between the sub-page and the overview page.

- Displaying sub-page number

  Click  in the toolbar of the CFC editing area to display the page numbers sub-pages.

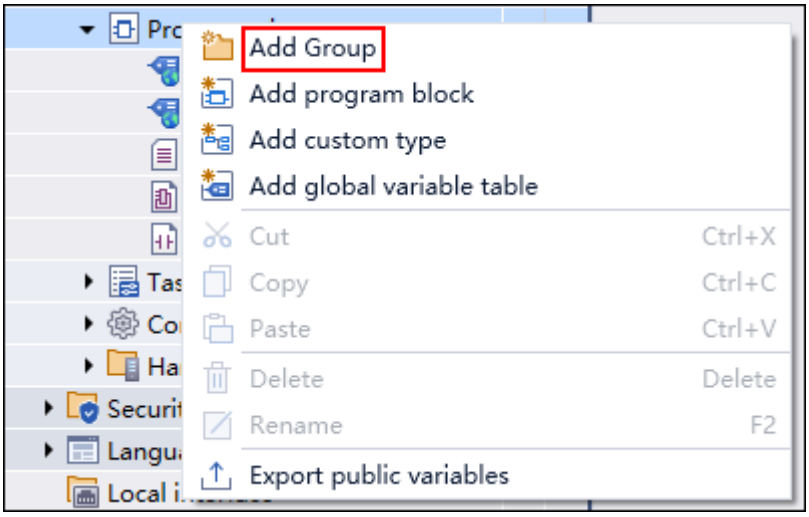


10.1.5.2.3 Managing a CFC Program Block and Group

10.1.5.2.3.1 Adding a Group

A first-level group can be created under the "Programming" node, which supports expanding multiple levels of subgroups.

1. Right-click "Programming" in the project tree and select "Add group".



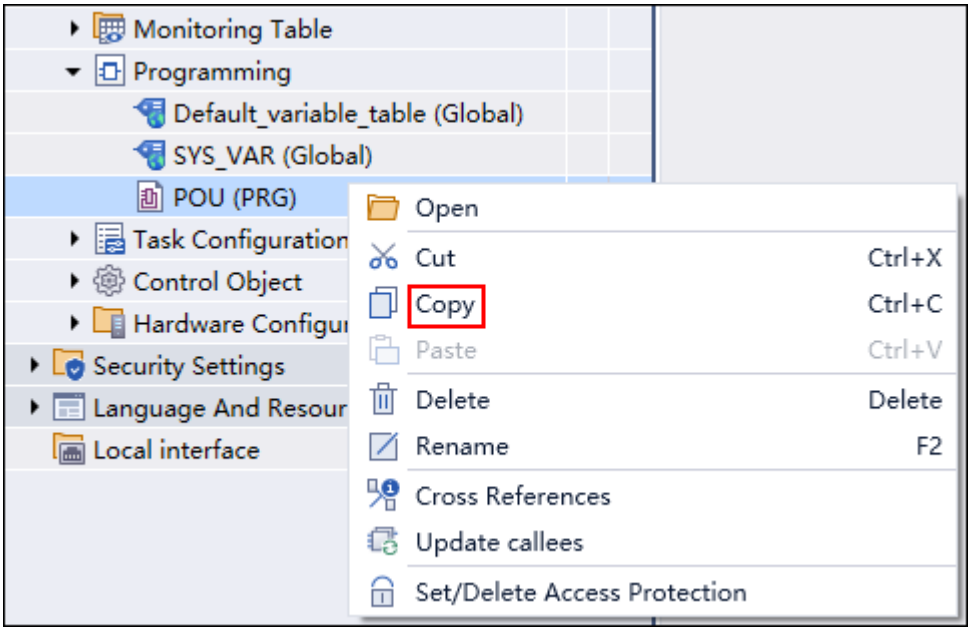
2. In the pop-up window, customize the group name and click "OK".

10.1.5.2.3.2 Copying a CFC Block/Group

Copying/Pasting CFC program blocks

CFC program blocks can be copied and pasted within a PLC device, between PLC devices in the same project, or between different projects. The internal program logic of the CFC program block remains unchanged after pasting.

1. Right-click a CFC program block name under the "Programming" node, and select "Copy".

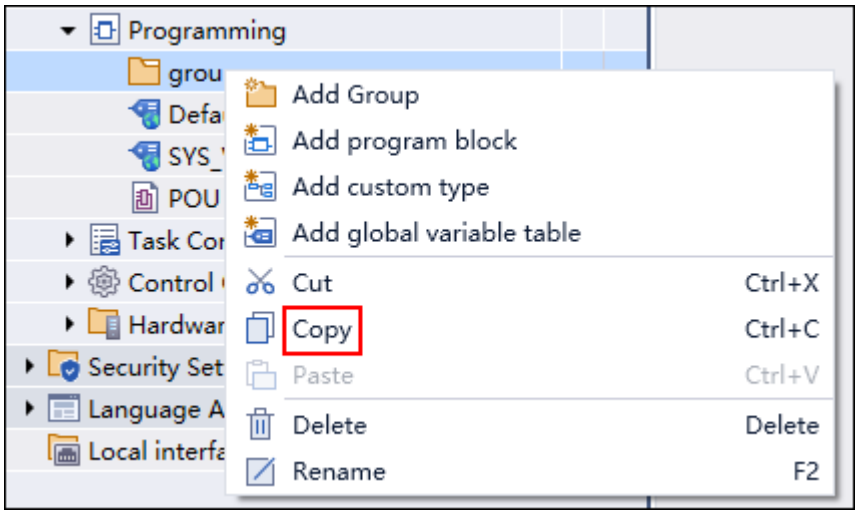


2. Right-click the "Programming" node or its subordinate node, and select "Paste" to paste the copied LD program block at the corresponding location.

Copying/Pasting groups

Groups can be copied and pasted within a PLC device, between PLC devices in the same project, or between different projects. When copying a group, the system also copies the resources added to the group, including subgroups, program blocks, global variable tables, and so on.

1. Right-click a group name under the "Programming" node, and select "Copy".

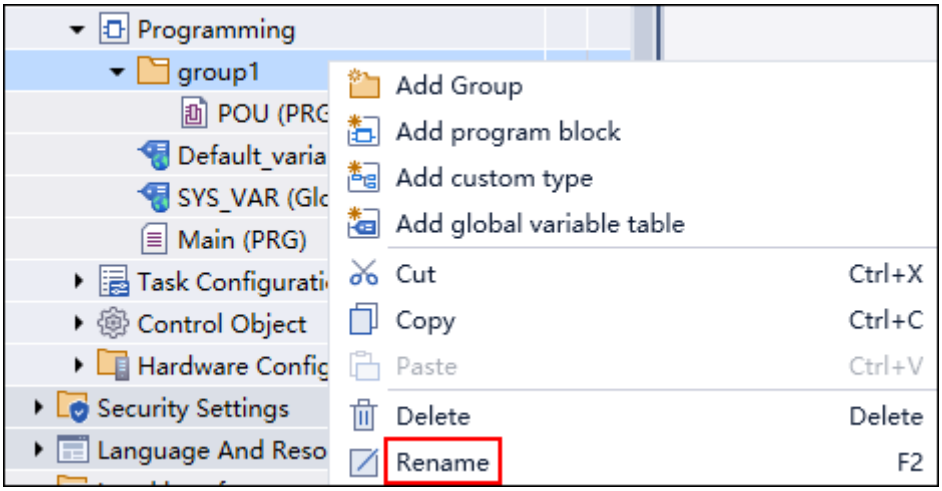


2. Right-click the target location under the "Programming" node, and select "Paste".

10.1.5.2.3.3 Renaming a CFC Block/Group

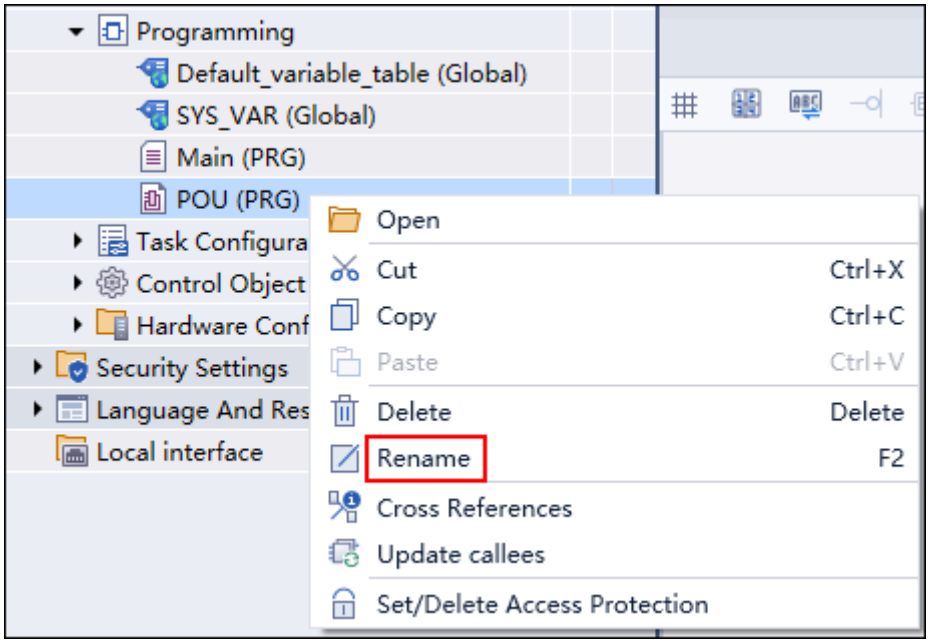
Renaming groups

Right-click a group name under the "Programming" node, select "Rename", and modify the group name.



Renaming CFC program block

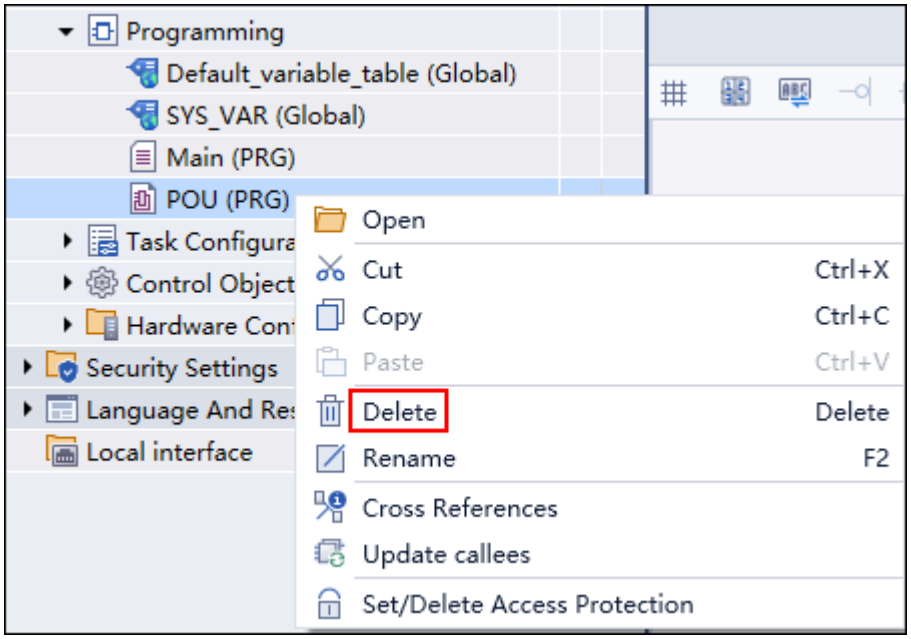
Right-click a CFC program block name under the "Programming" node, select "Rename", and modify the group name.



10.1.5.2.3.4 Deleting a CFC Program Block

After the CFC program block is deleted, its record in the task is also deleted.

1. Right-click a CFC program block name under the "Programming" node, and select "Delete".

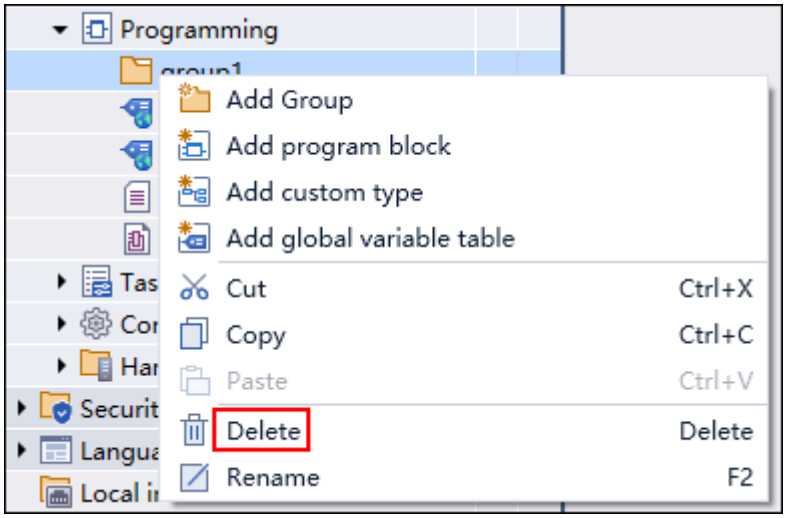


2. Click "OK" in the pop-up window.

10.1.5.2.3.5 Deleting a Group

When deleting a group, the system will also delete the resources added within the group, including subgroups, program blocks, global variable tables, and so on.

1. Right-click a group name under the "Programming" node, and select "Delete".

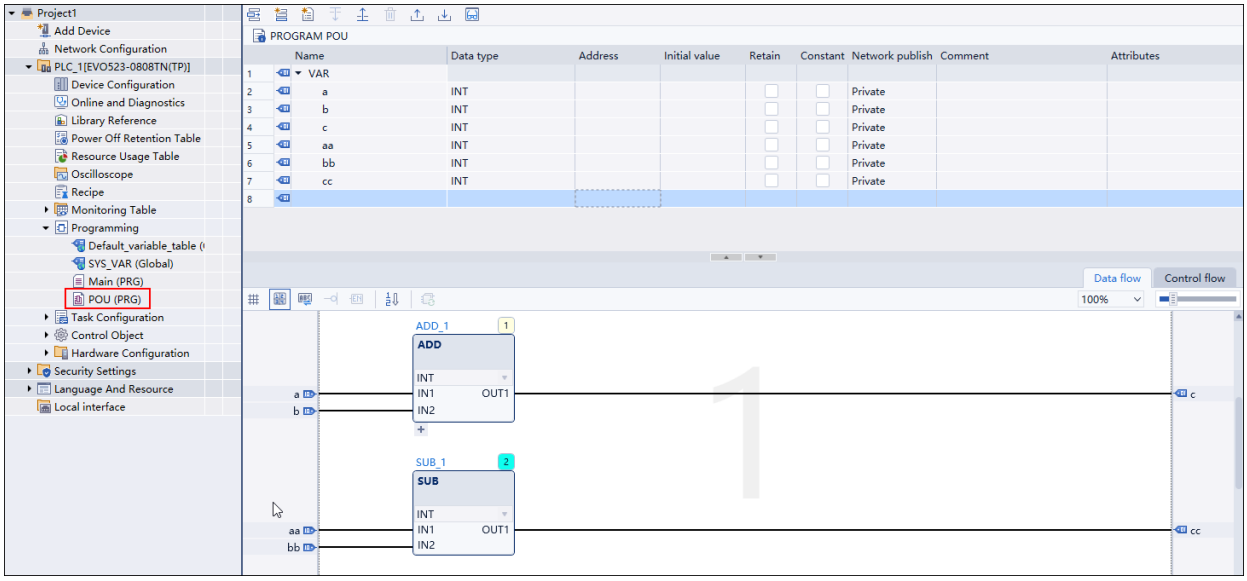


2. Click "OK" in the pop-up window.

10.1.5.3 CFC Editor

10.1.5.3.1 CFC Editor Interface

Double-click the CFC language program block (identifier: ) under the "Programming" node.



The CFC editor is divided into the variable declaration area and the program editing area, which can be collapsed and expanded using and .

The program editing area is divided into data flow and control flow display modes as follows.

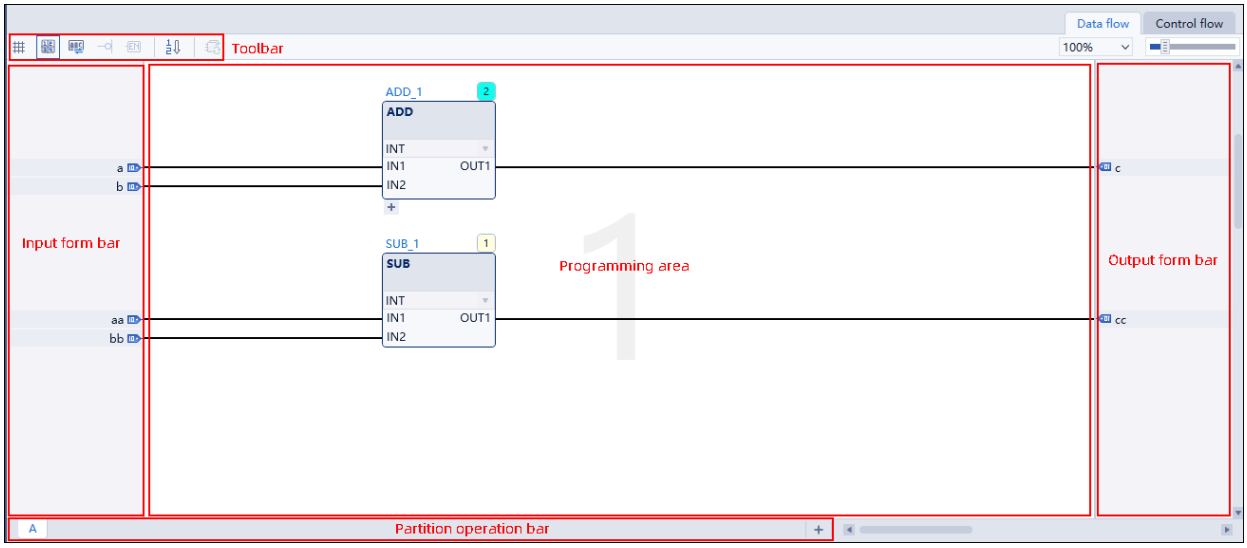


Figure 10-1-5-3-1-1 Data flow mode

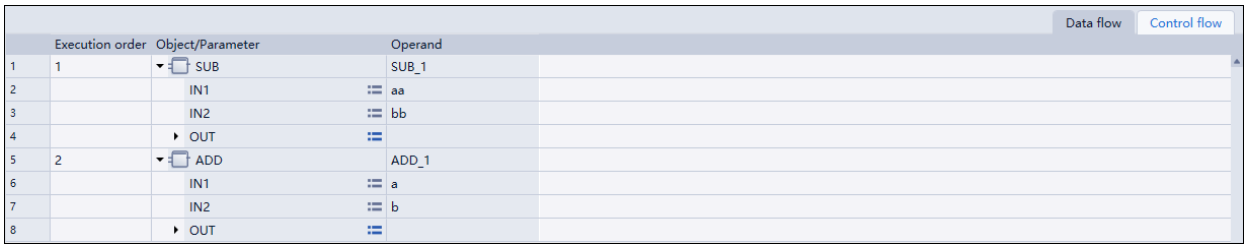


Figure 10-1-5-3-1-2 Control flow mode

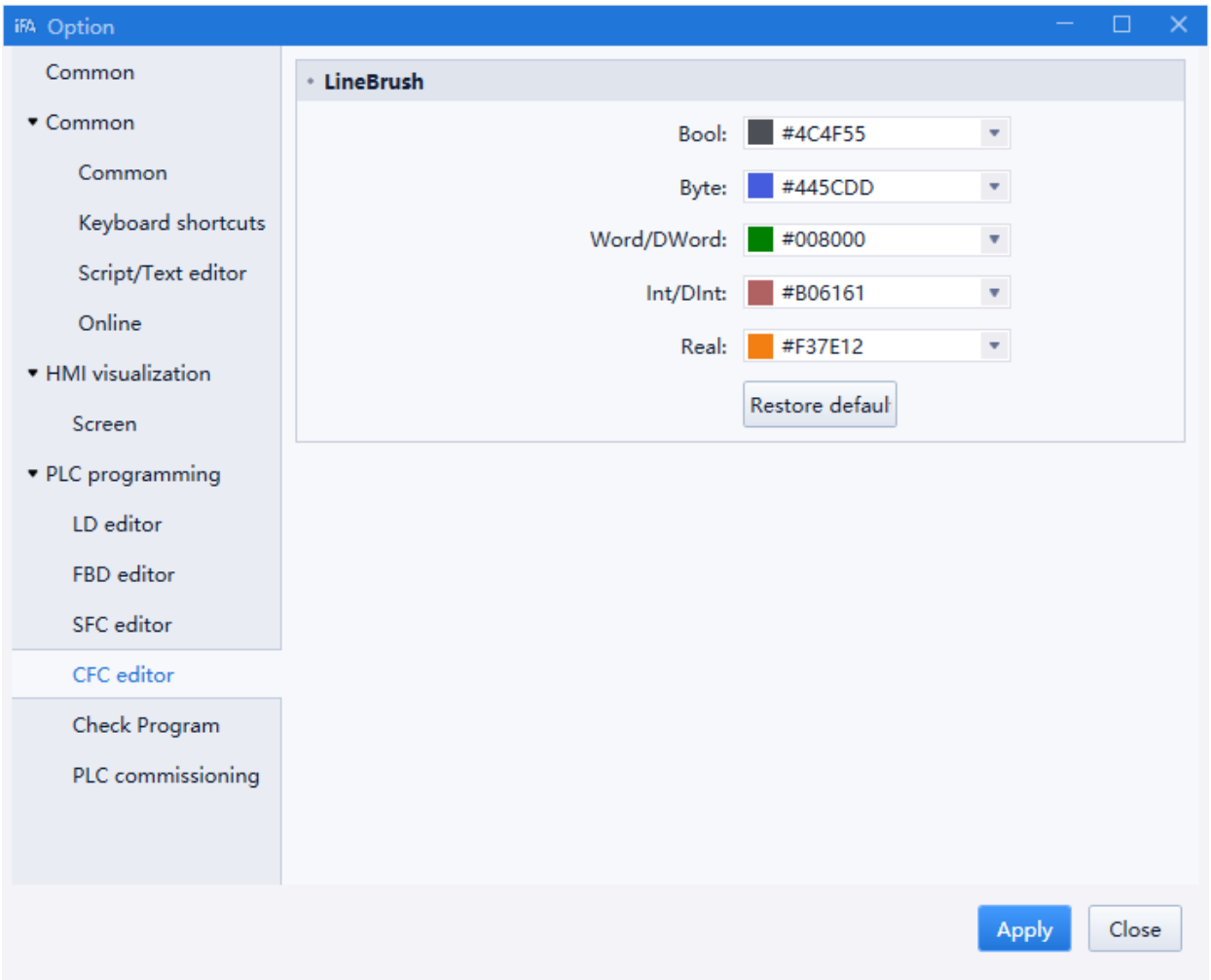
Table10-1-5-3-1-1 Introduction to the toolbar in data flow mode

Description	Icon	Description
Grid		Grid reference lines are displayed in the programming area.
Page number		Sub-page numbers are displayed.
Switch between parameter names and annotations		When input/output pins are set with aliases, the display mode of the pins can be switched between pin names and aliases.

Description	Icon	Description
Inversion		Inverts BOOL pins.
Add/Delete EN/ENO		Add or delete EN/ENO for function blocks.
Sort by data flow		Reorders operation blocks via algorithms according to the logic of "input first, then output". If there are multiple networks, sort them from top-left to bottom-right.
Update all parameters		Updates the parameters of the FC, FB instances called by the CFC program in the program editing area with one click.

10.1.5.3.2 CFC Editor Settings

Select "Tools > Option" from the menu bar. On the "PLC programming > CFC editor" tab, set the color of variable connections in the CFC editor. Click "Restore defaults" to reset all connection colors to their default values.



10.1.5.3.3 CFC Program Configuration Options

Configuring CFC programs

In the CFC editor, it is easy to use graphical tools to configure CFC programs graphically in data flow mode through simple drag-and-drop operations. Users can also switch to control flow mode and view and

manage the relevant textual information in the data flow through a list. Users can switch between the two configuration methods at any time without any restrictions.

Configuring CFC programs in data flow mode

Data flow is a graphical representation of configured CFC programs, where users can insert, connect, rename, and resort statements and blocks with drag-and-drop operations and shortcuts.

Configuring CFC programs in control flow mode

Control flow displays the relevant information of elements with execution order in the data flow in a tabular form. Users can view the associations between statements and blocks, their execution order, and pin connections.

Currently, control flow only supports reordering statements and blocks. After reordering, execution order of the data flow will also be adjusted accordingly.

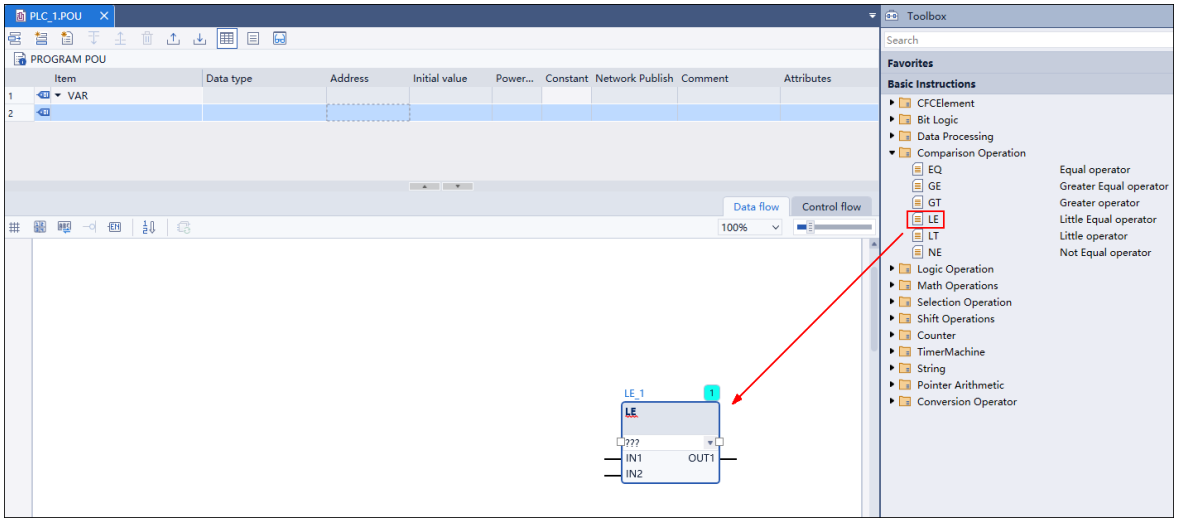
10.1.5.4 CFC Object Creation and Management

10.1.5.4.1 Creating an Operation Block

Adding an operation block

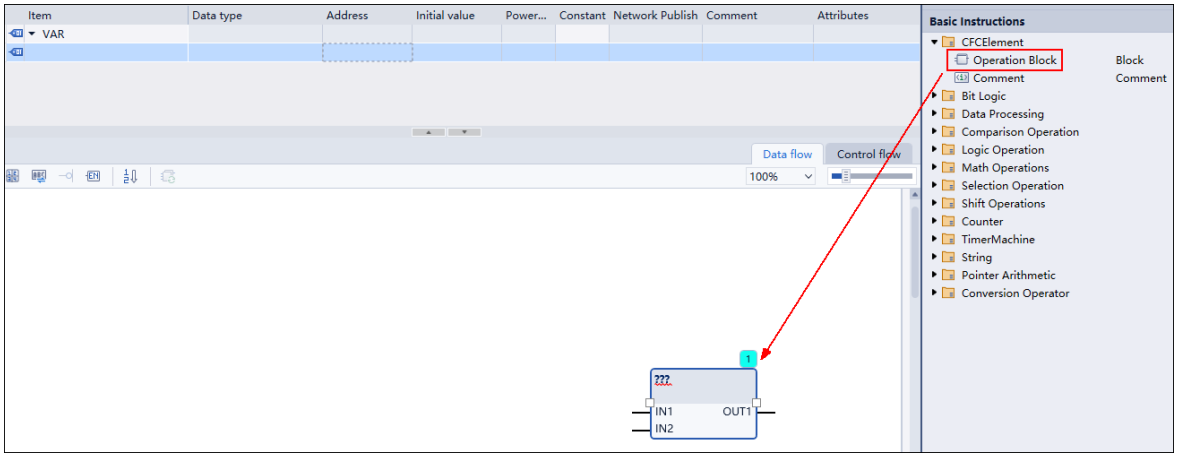
- Dragging elements from the toolbox
  - Adding an instruction

    Drag and drop instructions from the toolbox on the right to the programming area.



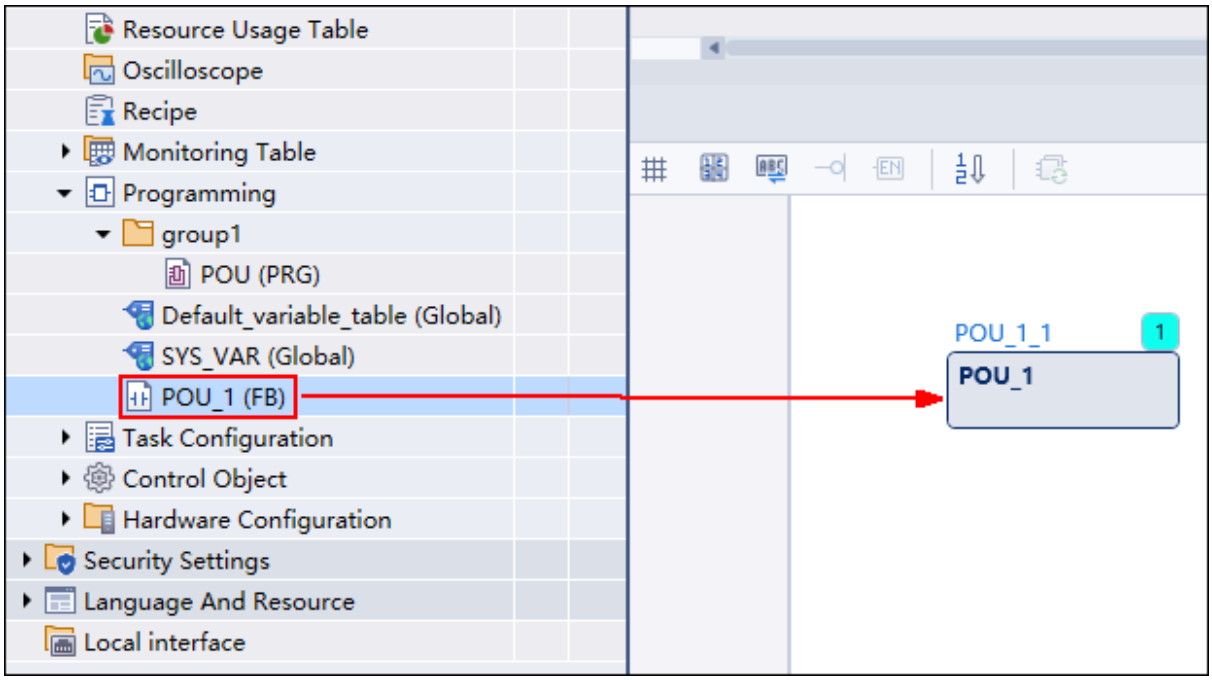
  - Adding an empty operation block

    In the toolbox, drag an "Operation block" element under "Basic instructions > CFC elements" to the programming area. It is suitable for users who are familiar with functions and function blocks.



- Dragging from the project tree

  Drag and drop an FB or FC instance from the project tree to the programming area.



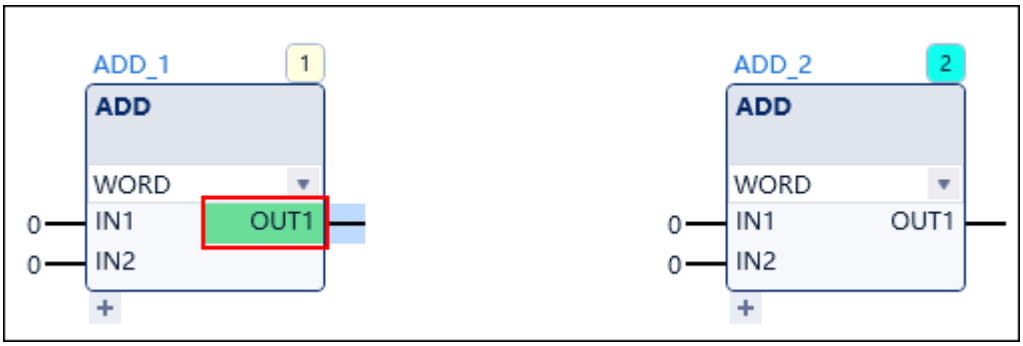
Connecting operation block pins

Operation block pins can be connected within the same subpage, across subpages, across partitions, and across program blocks.

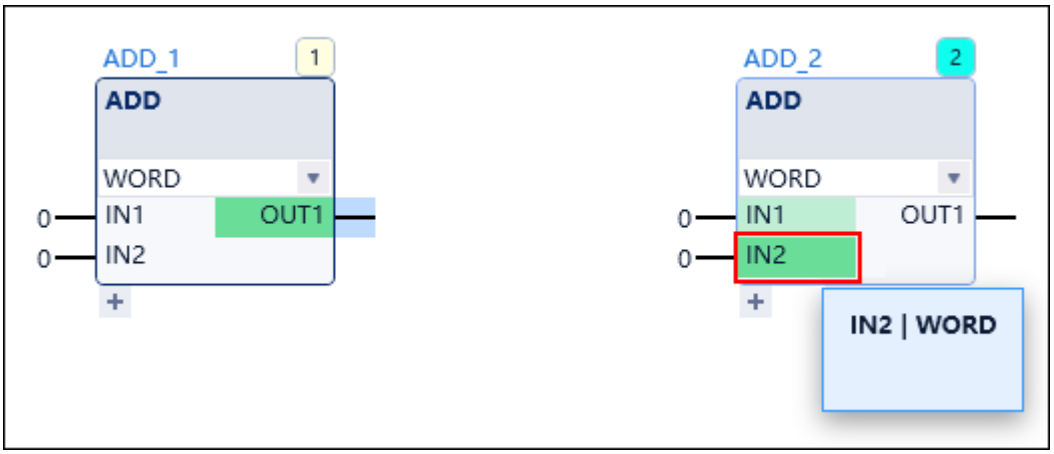
> **NOTE**
>
> All the following methods for connecting two pins support connecting from an Out pin to an In pin, as well as from an In pin to an Out pin, meaning the order of operation is not restricted.

- Connection by dragging
  - Click and hold the pin of the source operation block. All connectable pins are highlighted. Move the mouse to the target pin to complete the connection.
  - If the target pin is already in the form bar (in the blank space to the left/right of each sub-page), you can directly drag the source pin to the form bar object.

- Connection by clicking
  1. Double-click the pin of the operation block to activate the starting pin, which will be highlighted, as shown in the figure below.



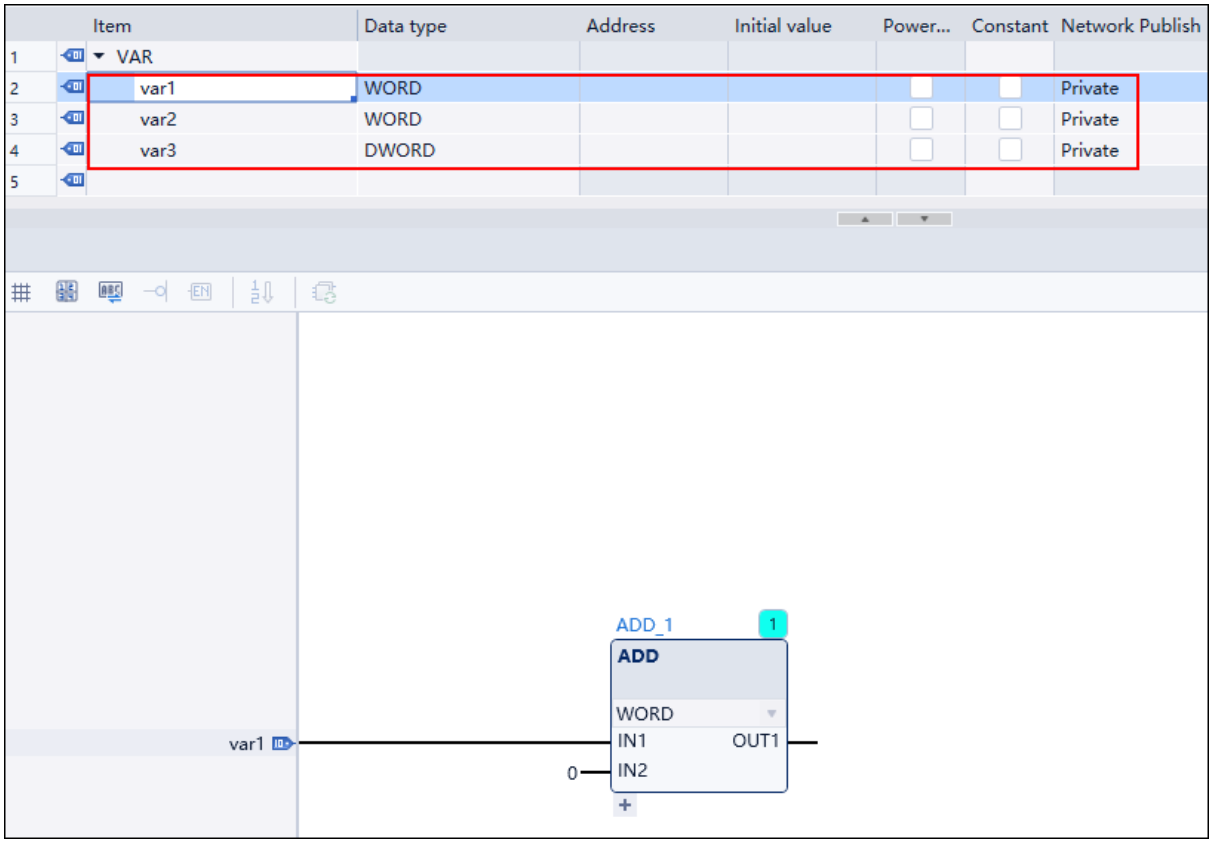
  2. When hovering the mouse over the highlighted starting pin, connectable pins will be highlighted (as shown in the IN1 and IN2 pins of ADD_2 in the figure below). Click the target pin to complete the connection.



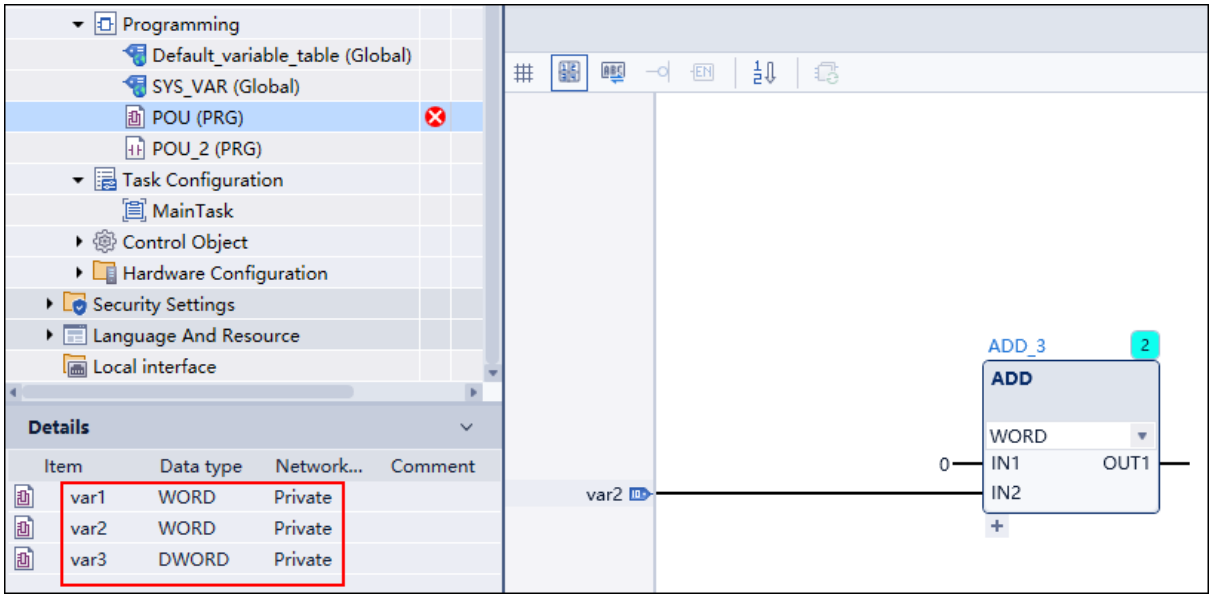
Connecting operation block pins and variables

The pins of the operation block can be connected to variables in the variable table. After connection, a connection line is automatically generated in the form bar. There are two methods:

- Drag and drop variables from the variable table to the In or Out pin of the CFC operation block.



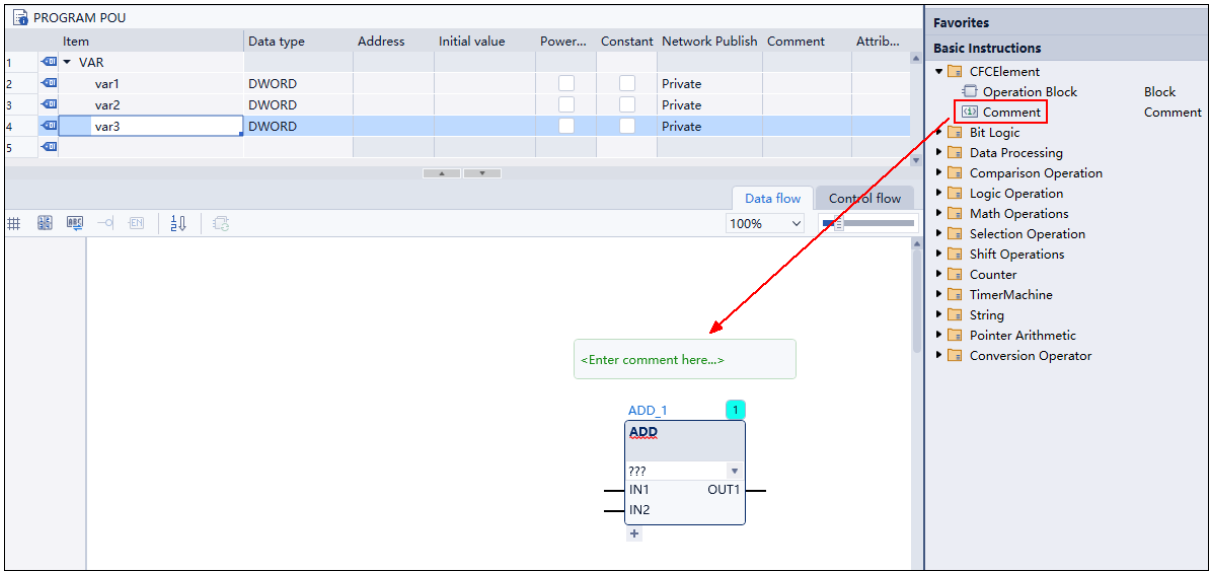
- Drag and drop variables from the "Details" window (in the lower left corner of the interface) to the In or Out pin of the CFC operation block.



10.1.5.4.2 Creating a Comment

Annotate the program content.

1. In the toolbox, drag a "Comment" element under "Basic instructions > CFC elements" to the programming area.

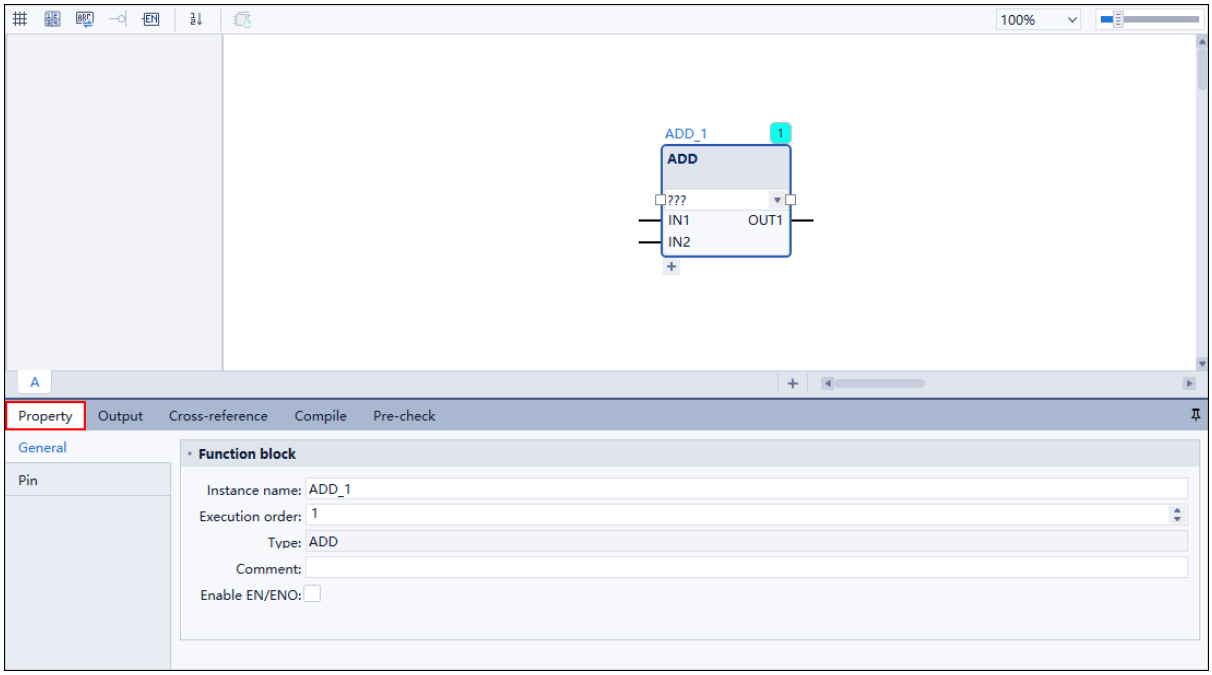


2. Double-click the annotation block and enter the content directly.

10.1.5.4.3 Managing an Operation Block

10.1.5.4.3.1 Configuring Operation Block Properties

1. When selecting an operation block in the programming area, the "Property" tab in the monitoring area displays the properties of this operation block.



2. Configure the operation block and pin properties in the "Property" tab. The parameter descriptions are as follows:
   - General Settings

Name	Editable	Note
Instance name	Yes	Indicates the operation block name, which must be unique within the current program block. The default is type_number. It is user-defined, and supports Chinese, letters, numbers, and underscores (_), with a maximum length of 128 characters.

Name	Editable	Note
Execution order	Yes	Indicates the execution order of the selected operation block within the program block, starting from 1 and incrementing according to creation order. It must be unique and continuous within the current CFC program block. It can be modified in both data flow mode and control flow mode.
Type	No	Indicates the function of the selected operation block, such as ADD, MIN, and DIV.
Comment	Yes	Used to explain the function of the operation block in the program or process. It is user-defined and is empty by default.
Enable EN/ENO	Yes	Indicates whether to enable EN/ENO pins for the selected operation block. It is user-defined and is disabled by default.
Network Public Property	No	Indicates whether the pins of the selected operation block are public, which is consistent with the instance definition of variables.

◦ Pin

Name	Editable	Description
Expression	No	Indicates the function of the pin.
Alias	Yes	Indicates the alias of the pin. The maximum length is eight characters.
Value	Yes	Indicates the value corresponding to the pin. When there is no connected object, the default value is displayed and users can manually enter a value.
Unit	Yes	Indicates the process unit represented by the pin data, such as °C, kg/L, and m/s. It is user-defined and is empty by default.
Data Type	No	Indicates the data type of the pin.
Pin Type	No	Indicates whether the pin is input or output.
Monitorable	Yes	Indicates whether the pin displays online values during online commissioning. It is user-defined and is non-onitorable by default.
Invisible	Yes	Indicates whether the operation block displays the pin. It is user-defined and is unchecked by default, indicating that the operation block displays the pin.

Name	Editable	Description
Network public property	No	Indicates whether HMI and other devices can access the value of the pin. It is consistent with the configuration in the variable type definition.
Comment	Yes	Indicates the additional explanation of the function and meaning of the pin. It is user-defined and is empty by default, with a maximum length of 64 characters.

NOTE

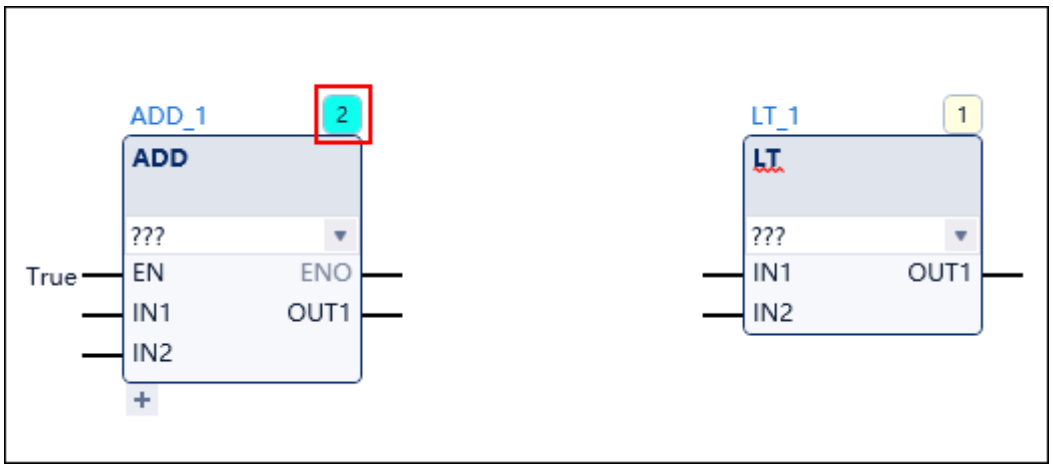
In addition to configuring pin comments in the "Property" tab, you can also directly right-click the target pin and select "Pin comment" for quick editing.

10.1.5.4.3.2 Reordering an Operation Block

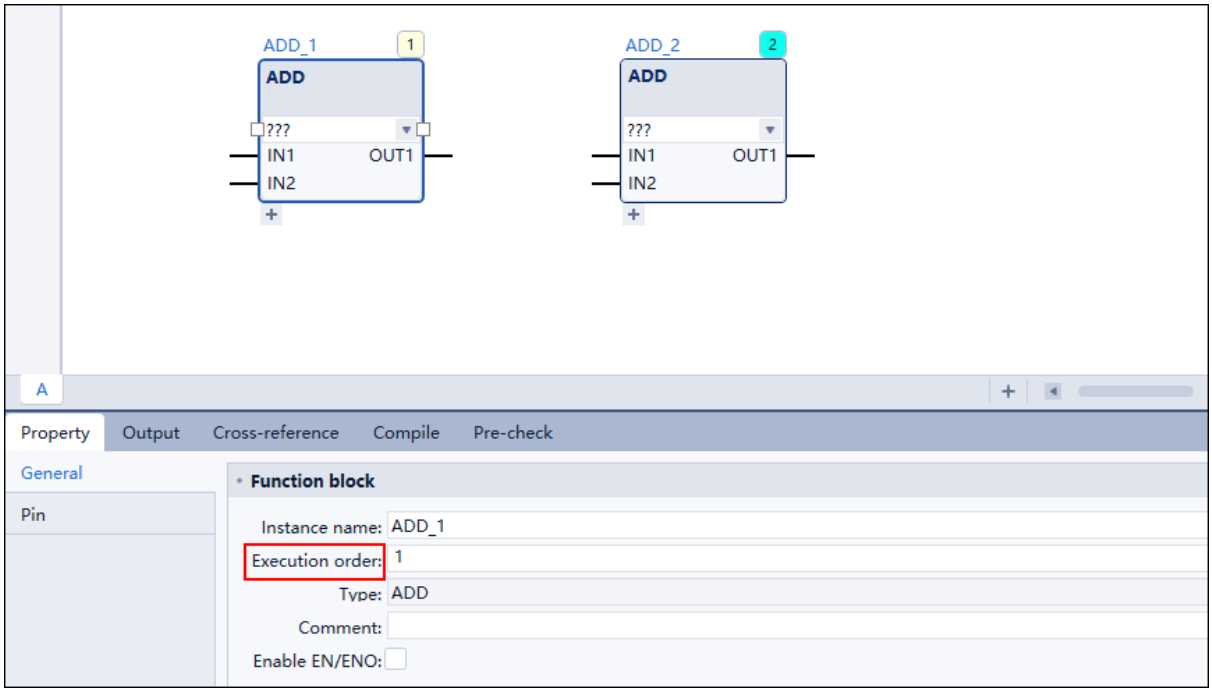
The numbers on the operation blocks represent their execution order, which can be adjusted in several ways:

Adjusting execution order in data flow mode

- Modifying numbers
  Double-click the number of an operation block in the programming area, and input a new value. For example, there are three operation blocks, A1, B2, and C3. If the sequence number of C is changed to 1, the execution order will automatically be resorted to C1, A2, and B3.

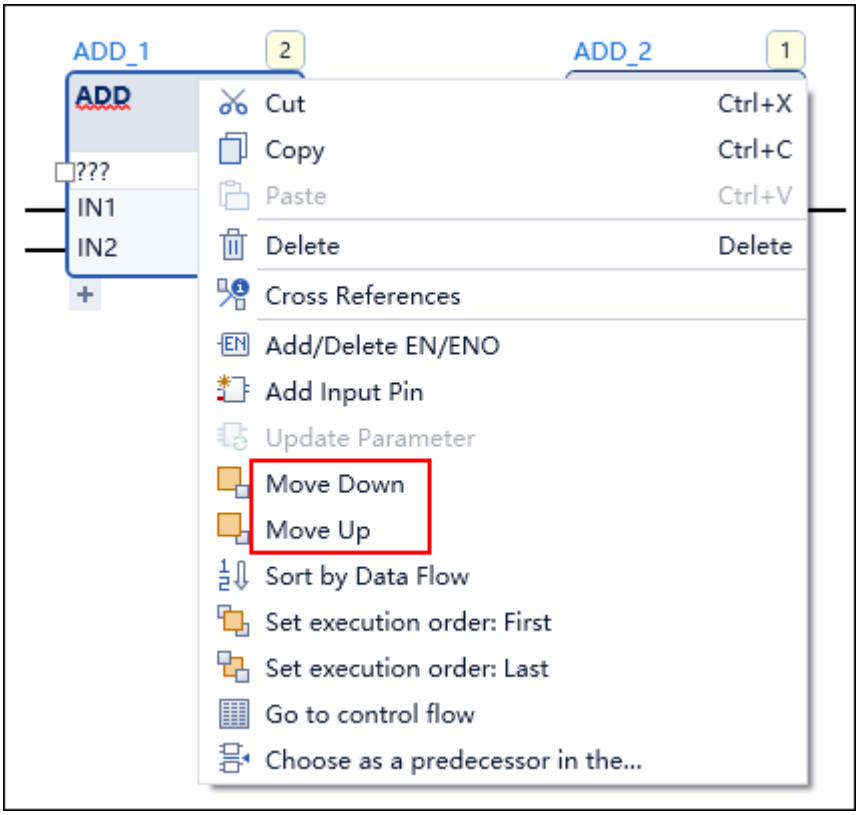


- Modifying order parameters
  Select an operation block in the programming area, and modify the "Execution order" parameter in the "Property" tab of the monitoring area.



- Moving down/up

  Right-click an operation block in the programming area and select "Move down" or "Move up" to move the block backward or forward by one sequence number.



- Set head/tail

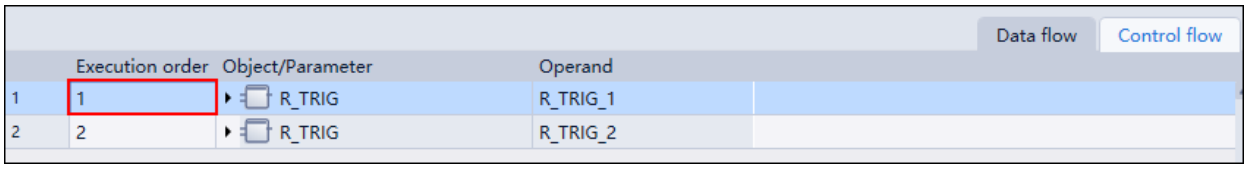
  Right-click an operation block in the programming area and select "Set head" or "Set tail" to set the block as the first or last to execute.

- Sorting by data flow

  Click in the programming area toolbar or right-click an operation block and select "Sort by data flow" to automatically adjust the execution order.

Adjusting execution order in control flow mode

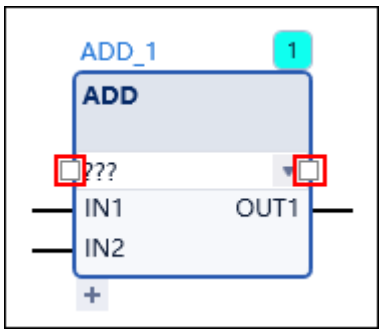
Click the "Control flow" tab in the upper right corner of the programming area to switch to the control flow mode. Double-click the cells in the "Execution order" column to modify the order.



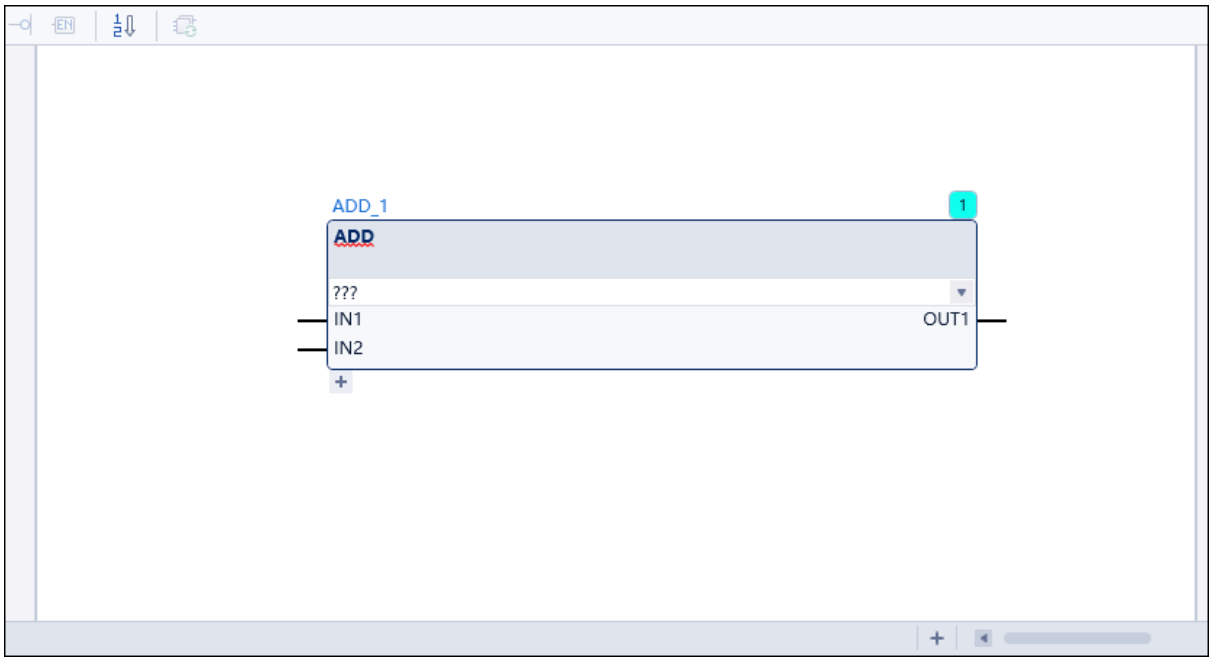
10.1.5.4.3.3 Resizing an Operation Block

The width of an operation block can be adjusted by dragging with the mouse.

1. Click the left/right border of an operation block, and a drag box as shown in the figure below will appear.

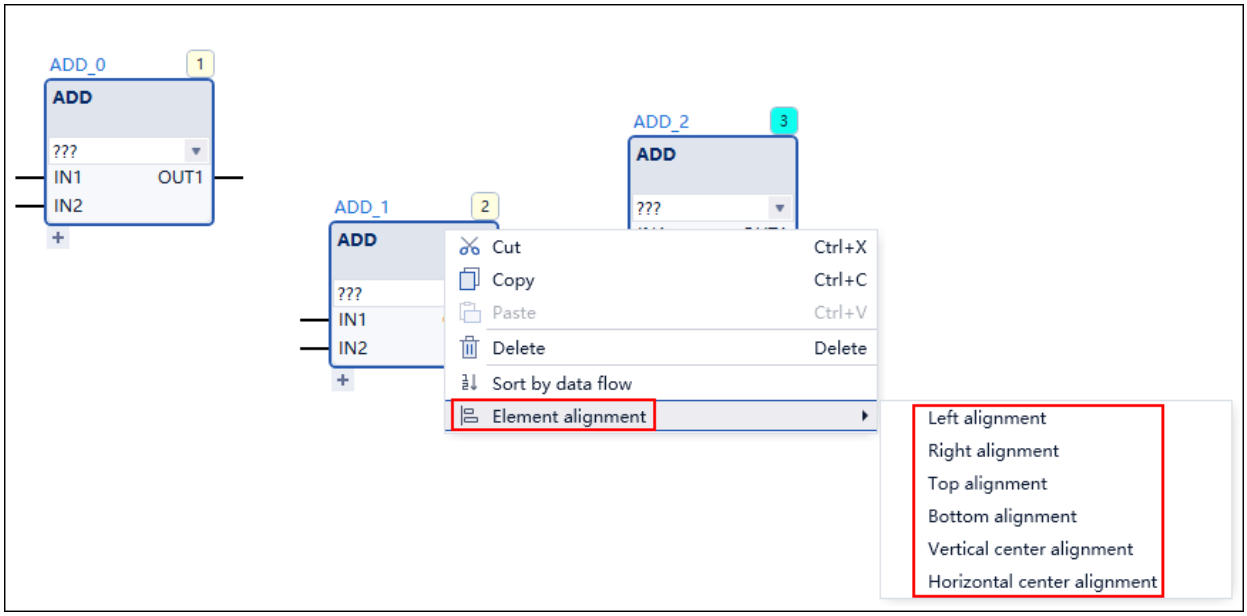


2. Move the mouse to the drag box, and drag left/right. The effect is shown in the figure.
   During the dragging process, the system automatically constrains the left and right boundaries. The maximum width of the operation block is 80% of the subpage width.



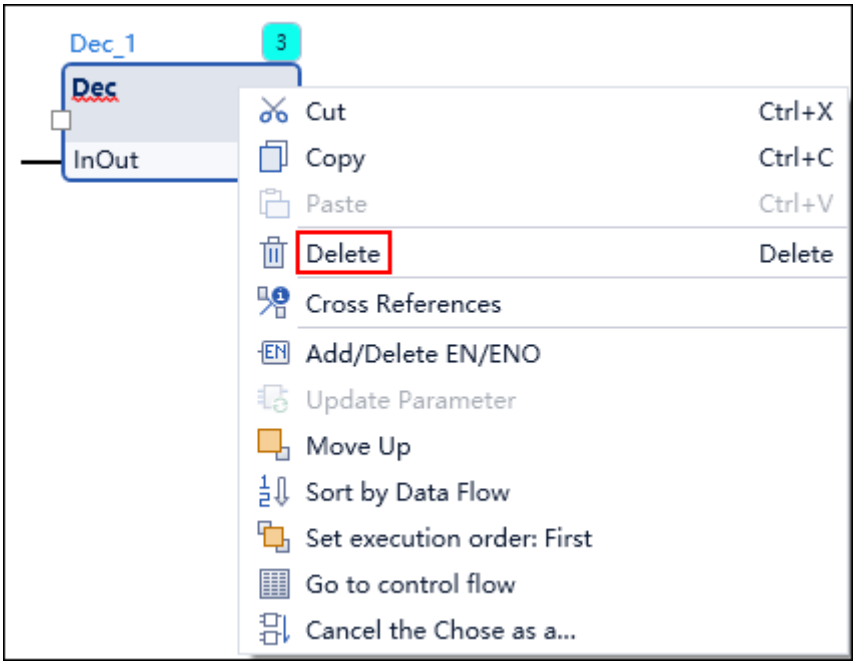
10.1.5.4.3.4 Repositioning an Operation Block

Right-click the operation block to move in the programming area and select "Element alignment" to adjust the alignment position of the operation block.



10.1.5.4.3.5 Deleting an Operation Block

Right-click an operation block in the programming area and select "Delete" to delete the block. After deleting a CFC operation block, all connections to its pins are also deleted.



10.1.5.4.3.6 Updating an Operation Block

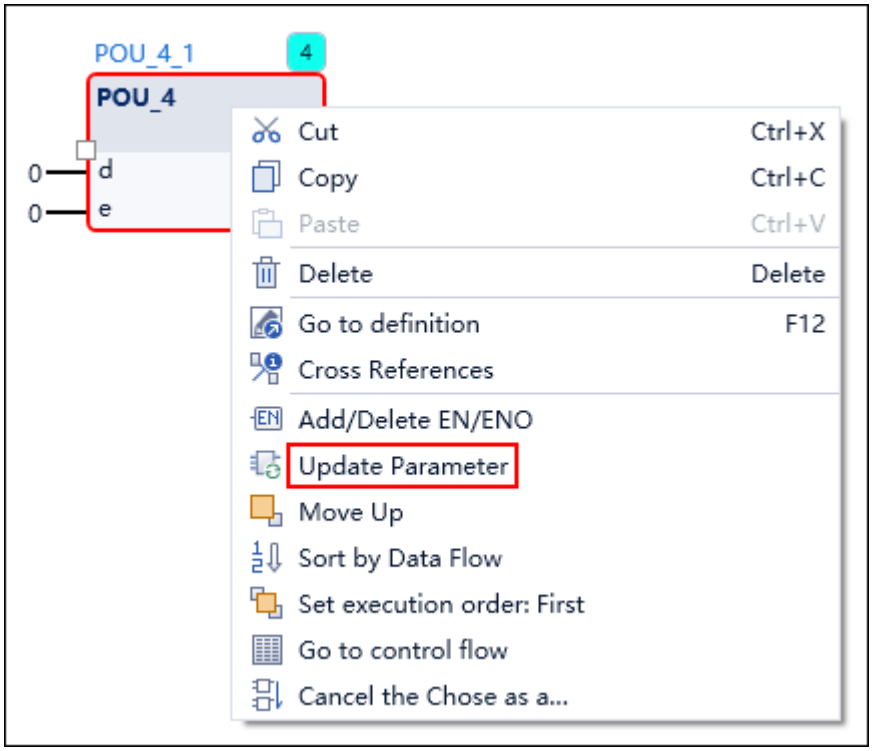
When an FB or FC instance is called by a CFC program, the parameters of the called object can be updated with one click in the programming area. Parameter updates are mainly used when parameters are added, deleted, or when their data types are modified. If the parameter names of the called object change, they will be automatically renamed, without the need for manual updates.

Prerequisites

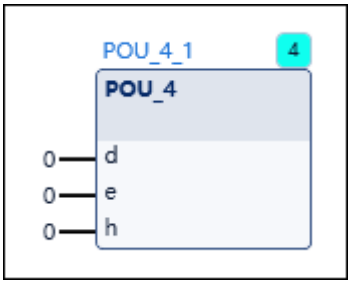
- An FB or FC instance has been called by a CFC program.
- Parameters in the called FB or FC instance are added, deleted, or their data types are modified.

Updating parameters of an individual block

1. In the editing area, right-click a CFC program block and select "Update parameters", as shown in the following figure.

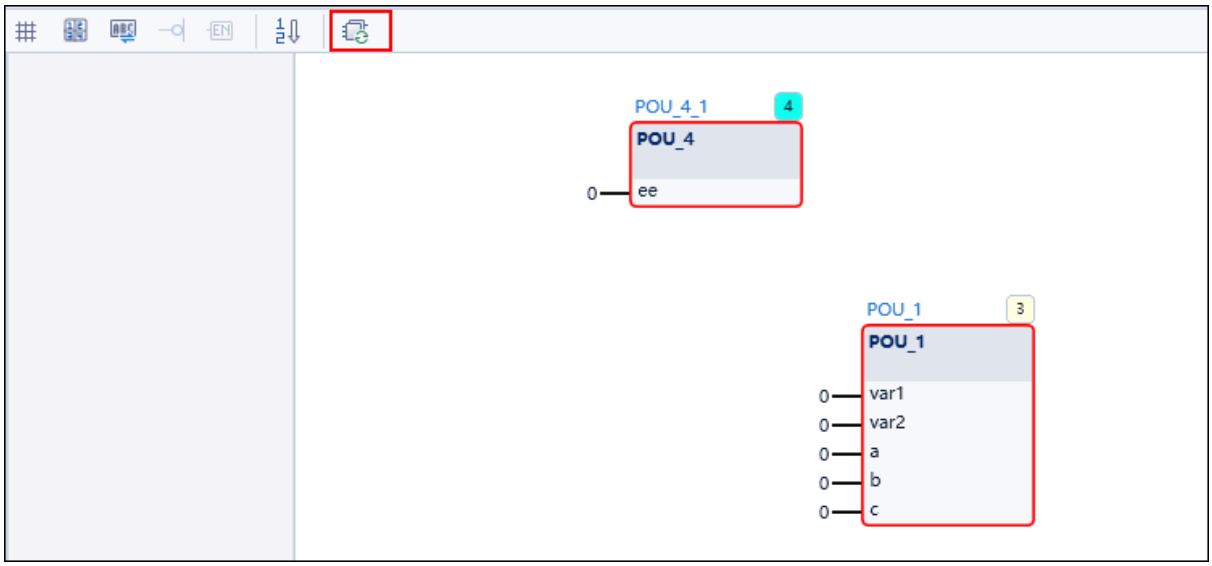


2. Update the parameters of the target instruction to align the parameters in the current program block with the latest parameter configuration of the called object.

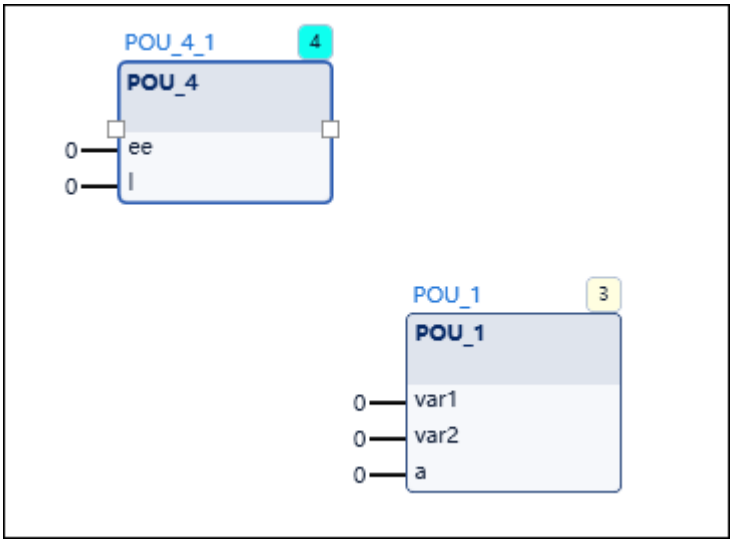


Updating all parameters

1. In the toolbar of the program editing area, click the "Update all parameters" icon, as shown in the following figure.



2. All program blocks in the editing area that have parameter changes will be updated in batches.



10.1.5.4.3.7 Operation Block Operations

Introduction to the operation block menu

Function	Description
Cut	Cuts the operation block.
Copy	Copies the operation block.
Paste	Pastes the operation block.
Delete	Deletes the operation block.
Cross-reference	Used to view where an operation block is referenced in the program. This helps users quickly locate the usage and position of the specified operation block, improving commissioning and maintenance efficiency.
Add/Delete EN/ENO	Adds or deletes EN/ENO for operation blocks.
Update parameters	Updates the parameters of the FC/FB instances called by the CFC program with one click.
Move down	Moves the execution order of the operation block back by one position.
Move up	Moves the execution order of the operation block forward by one position.
Sort by data flow	Reorders operation blocks via algorithms according to the logic of "input first, then output". If there are multiple networks, sort them from top-left to bottom-right.
Set head	Sets the operation block as the first to execute.
Set tail	Sets the operation block as the last to execute.
Go to control flow	Locates the position of the selected operation block in the control flow.
Switch pin monitoring status	Switches the pin monitoring status between monitorable and non-monitorable modes.
Leading block of execution order	Newly inserted operation blocks will follow the leading block. For example, there are three operation blocks, A1, B2, and C3. If B is set as the leading block, the newly inserted operation block D will follow B, resulting in A1, B2, D3, and C4.

10.1.5.5 Input/Output Parameter Connection

10.1.5.5.1 Setting Input/Output Parameters

In CFC programming, operation blocks contain two types of predefined pins: input pins and output pins. For pins of basic data types, the system automatically assigns initial default values. When pins are connected to external objects, the connection signals will override the default values of the pins.

Users can define variables of various types in the variable declaration area and connect them to the operation block pins with matched data types, as follows.

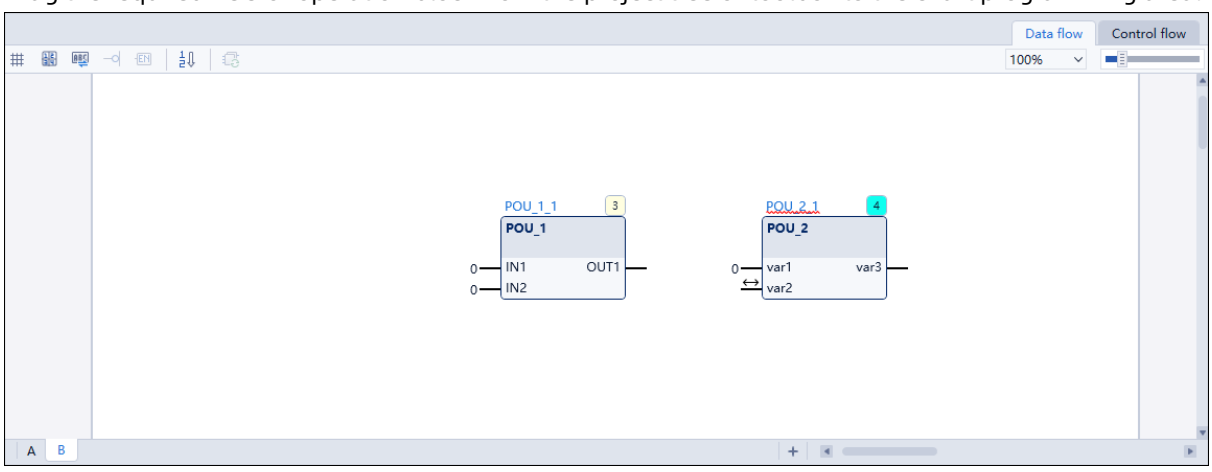
Variable Source	Operation Block Pin	Variable type
Current POU	Input pins	VAR
		VAR_IN_OUT
		VAR_INPUT
		VAR_OUTPUT
	Output pins	VAR
		VAR_IN_OUT
		VAR_OUTPUT
External POU	Input pins	VAR
		VAR_INPUT
		VAR_OUTPUT
	Output pins	VAR_INPUT
Default Variable Table	Input pins and output pins	VAR_GLOBAL
Global variable table	Input pins and output pins	VAR_GLOBAL

Prerequisites

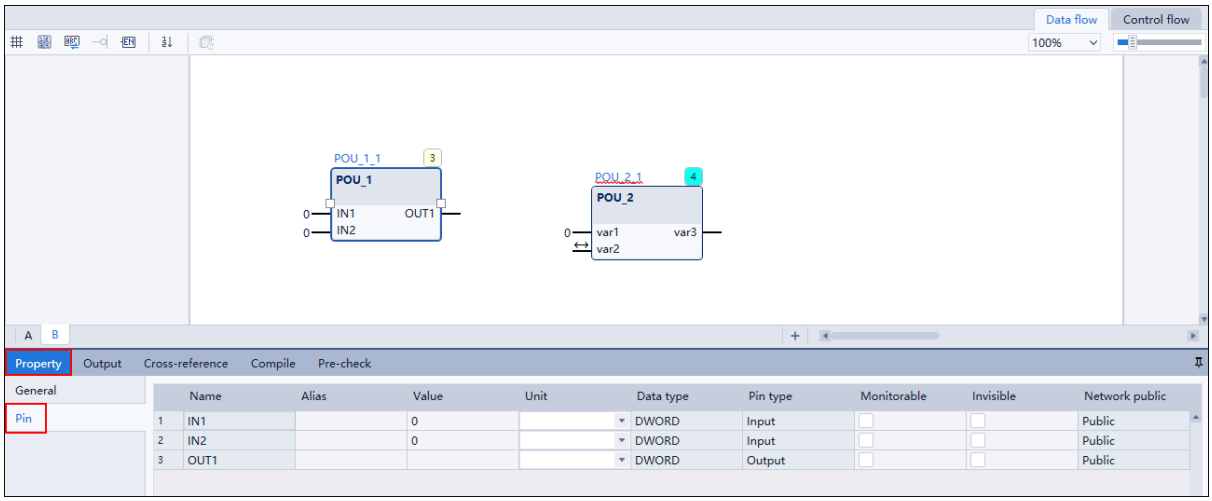
- The CFC editor has been opened.
- The CFC editor is in data flow mode.
- The required operation blocks or instructions have been added to the chart.

Procedure

1. Drag the required POU or operation block from the project tree or toolbox to the chart programming area.



2. Select an operation block. Its pin parameters can be viewed and configured in the "Property > Pin" tab in the monitoring area. For detailed parameter descriptions, see Configuring Operation Block Properties > Pin.



Result

The input and output parameters are configured for the operation block. When the CFC chart is downloaded to the device, these parameters will default to all the attributes that have been set.

10.1.5.5.2 Adding an Input Parameter

When inserting an instruction into the CFC chart programming area, the instruction will default to include relevant input and output parameters. For expandable instructions, the system allows the addition of extra input parameters individually. These instructions will be marked with + at the bottom.

Prerequisites

- The CFC editor has been opened.
- The CFC editor is in data flow mode.
- An expandable instruction has been added to the chart.

Procedure

1. Select the required instruction in the CFC chart programming area.
2. Perform one of the following actions:
   Click + to add an input pin.
   Right-click the instruction and select "Add input pin".

Result

The selected instruction is added with one or more input parameters. Select the instruction. Its pin parameters can be viewed and configured in the "Property > Pin" tab in the monitoring area.

- Click + to add an input pin.
- Right-click the instruction and select "Add input pin".

10.1.5.5.3 Connection Routing

In CFC programming, connection refers to connecting the input and output parameters of operation blocks. The connection routing refers to the path through which data flows from a source object (such as an output

pin of an operation block or a variable) to the input pin of a target operation block. This path is automatically generated by the editor and cannot be manually adjusted by users.

Connection requirements

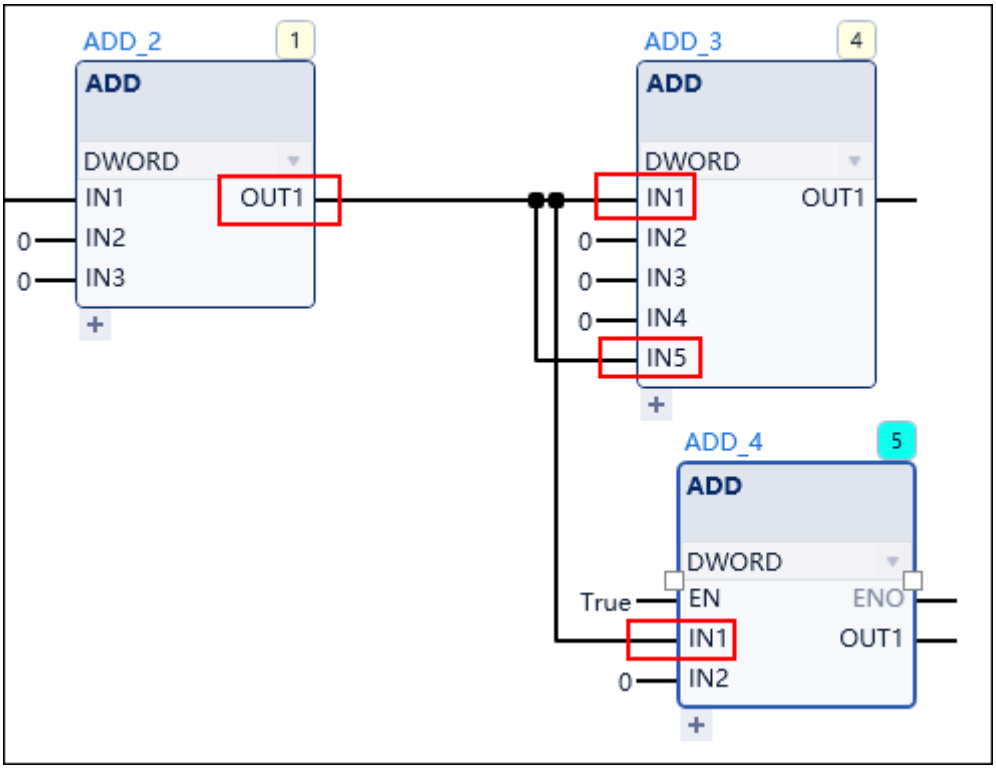
- Pins of operation blocks within the same chart can be directly connected.
- The same output pin can be connected to multiple different input pins.
- The output pin of an operation block can be connected to its own input pin.
- Pins of operation blocks support connections across partitions, sub-pages, and program blocks.
- Pins of operation blocks can be connected to the variables in the variable declaration area, "Details" window, or the global variable table.
- It is prohibited to connect multiple different output pins to the same input pin.
- It is prohibited to connect the input pins of the same or different operation blocks to each other.
- It is prohibited to connect the output pins of the same or different operation blocks to each other.

NOTE

- Connections are only allowed between pins with matching data types.
- If the data type of either end of a connection is modified after the connection is established:
  - The compiler issues an alarm but the connection is allowed when there is an available implicit conversion.
  - A compilation fault is reported when there is no valid conversion.

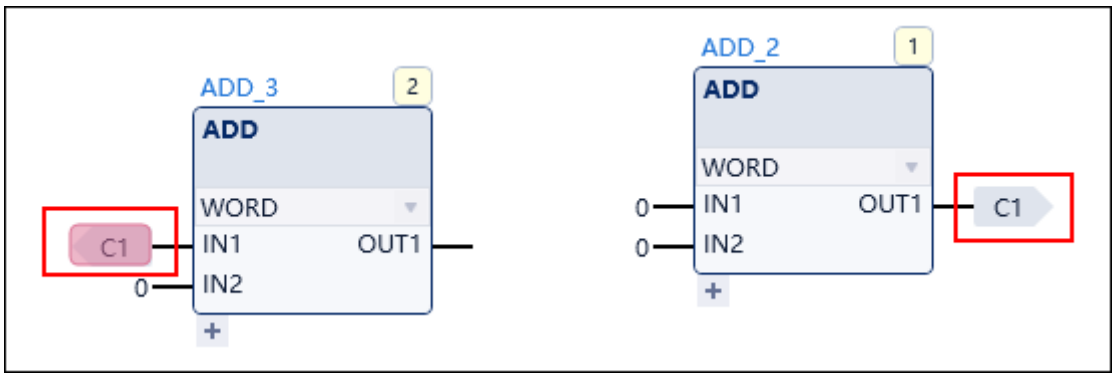
One-output-to-many-inputs connection

One-output-to-many-inputs connection refers to connecting one output parameter to several input parameters. The following figure shows a case where the same output pin is connected to three different input pins.



Incompletely displayed connection

Due to layout limitations, connection lines that cannot be fully displayed will be replaced by connection markers automatically inserted by the editor.

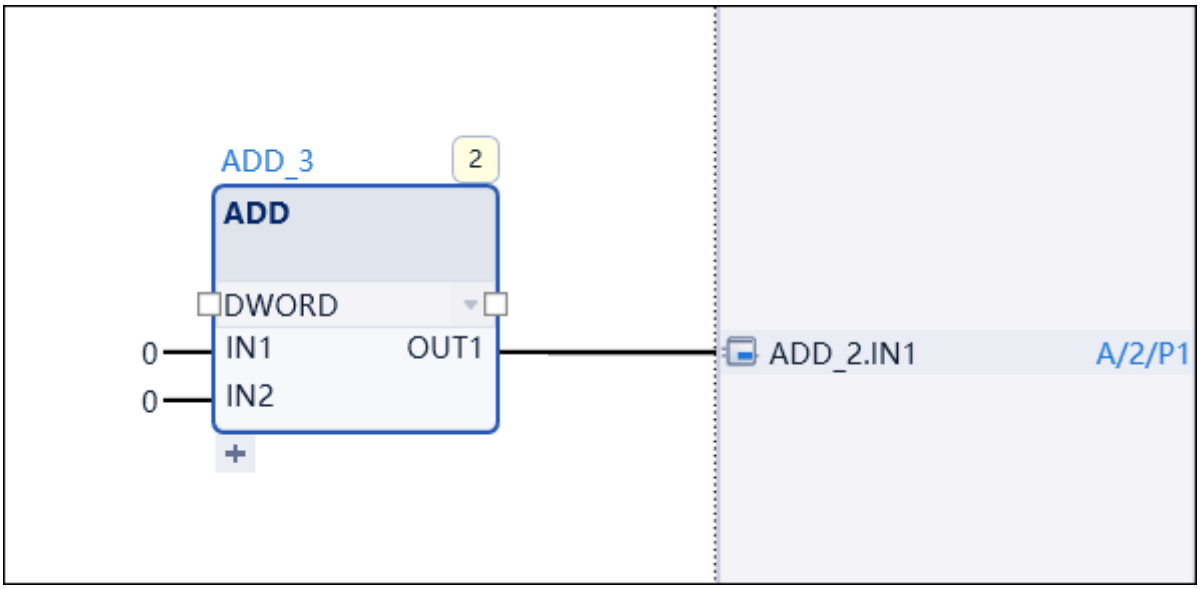


NOTE

- Markers with the same number indicate the same connection.
- If there are multiple connection markers in the chart, their numbers increase sequentially.
- Double-click any marker to quickly locate the matching marker.

Connection across pages and partitions

If the connection line extends beyond the current form boundary, the connection information will be displayed in the form bar (the blank space to the left or right of each sub-page).



NOTE

- A connector will be automatically inserted in the form bar, displaying the name of the target object and parameters.
- If the connection target is located in another CFC chart, the name of the target chart will also be displayed.
- If the CFC chart contains multiple sub-pages, the sub-sub-page name and form number will also be displayed.

10.1.5.5.4 Connecting Input and Output Parameters

There are three ways to connect input parameters and output parameters.

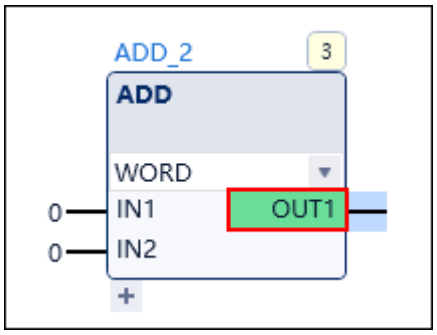
- Double-click > click
- Copy > click
- Drag and drop

Prerequisites

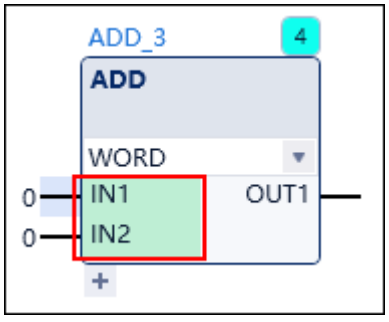
- The CFC editor has been opened.
- The CFC editor is in data flow mode.
- The required operation blocks or instructions have been added to the chart.

Procedure of double-clicking > clicking

1. Double-click the source pin; the pin will be highlighted and the connection function is activated.



2. Move the mouse to the target pin. If connection is allowed, the target pin will be displayed in a light color. Click any target pin to create a connection.



3. With the source pin being active, move the mouse and click other target pins to quickly establish connections from a single source pin to multiple target pins.

Procedure of copying > clicking

1. Copy the source pin; the pin will be highlighted and the connection function is activated.
2. Move the mouse to the target pin. If connection is allowed, the target pin will be displayed in a light color. Click any target pin to create a connection.
3. With the source pin being active, move the mouse and click other target pins to quickly establish connections from a single source pin to multiple target pins.

Procedure of dragging and dropping

1. Click and hold the source pin. All connectable pins will be displayed in a light color.
2. Drag the source pin to the target pin and drop it to complete connection.

10.1.5.5.5 Modifying a Connection

During the development of CFC charts, users can adjust the connections between parameters by dragging the input and output pins of operation blocks to dynamically update control logic.

Prerequisites

- The CFC editor has been opened.
- The CFC editor is in data flow mode.
- Connections have been established between operation blocks.

Procedure

1. Select the connection line to be modified.
2. Perform one of the following actions:
   Connect the input pin to another output pin.
   Connect the output pin to another input pin.
3. Release the mouse to complete the reconnection.

Result

The connection is successfully updated. The system will automatically adjust the layout of the connection lines to ensure a clear chart. Data flow will be transmitted according to the new connection path.

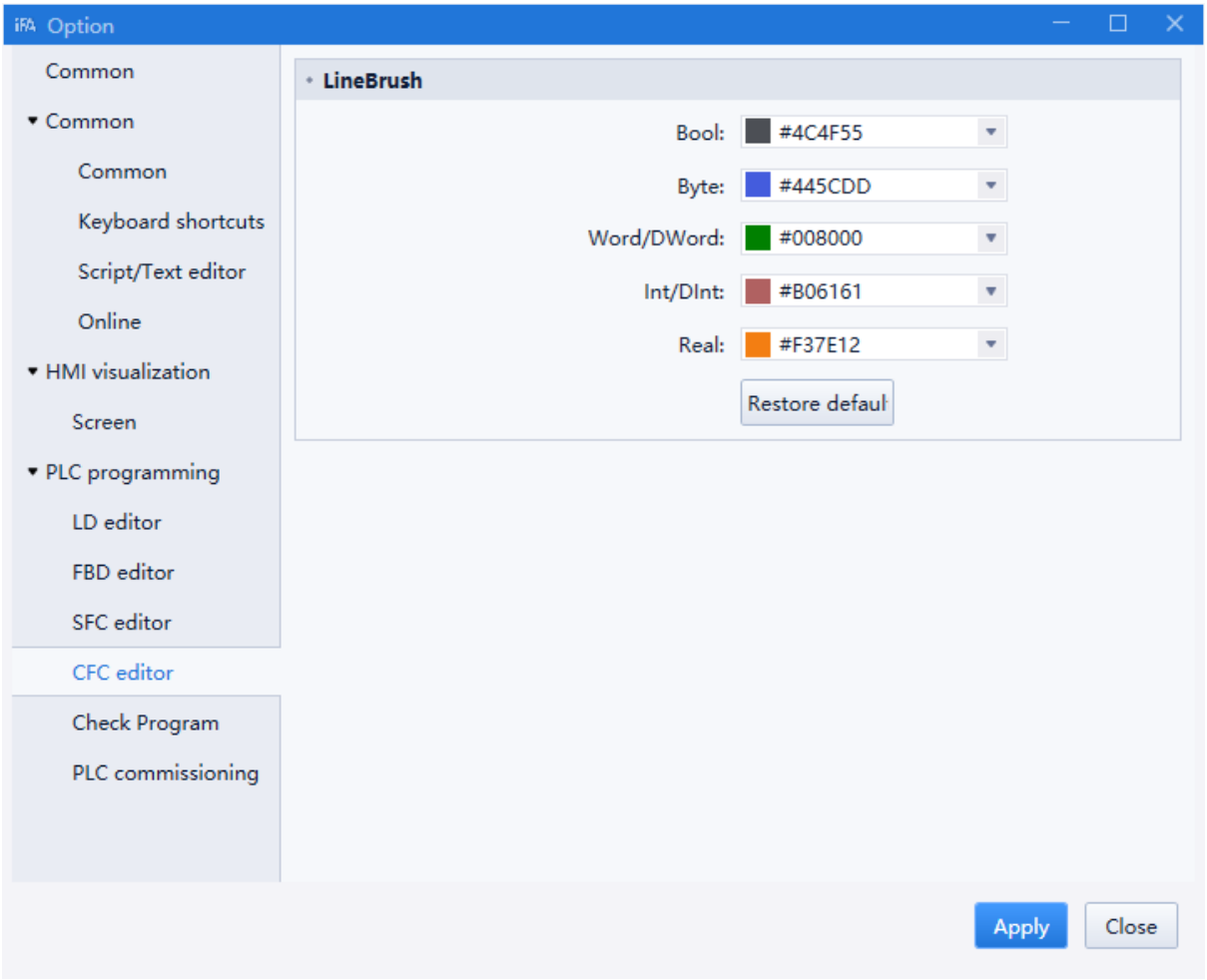
10.1.5.5.6 Changing Connection Line Color

In the CFC "Data flow" view, connection lines are displayed in different colors based on data type. The data type of the data source automatically determines the color of its connection lines; users can customize the mapping between data types and colors.

Procedure

1. In the software menu, select "Tools > Options > CFC editor".

2. Set the connection line colors for different data types. Click "Restore defaults" to reset all connection line colors to their default values.



Result

Connection line colors are changed successfully.

NOTE

- For operands with undefined data types, their connection lines are black by default.
- The new color setting will apply to connection lines in all open CFC views.

10.1.5.5.7 Deleting a Connection

To interrupt the data flow connection between operation blocks, users can delete the established connections.

Prerequisites

- The CFC editor has been opened.
- The CFC editor is in data flow mode.
- Connections have been established between operation blocks.

Procedure

1. Right-click the connection line to be deleted.

2. Select "Delete" or directly press the Delete key.

> NOTE
>
> If an operation block is deleted, the associated connections will also be deleted.

Result

The selected connection is successfully deleted, and the data flow connection between operation blocks is interrupted. The associated pins revert to an unconnected status, allowing users to reconfigure parameters or establish new connections.

10.1.5.5.8 Text Connection

In CFC, text connection is used to preserve the connection relationships of the original calculation blocks when copying and pasting them within the same POU, between different POUs, or between different PLCs. Text connection marks are automatically generated upon pasting and cannot be manually modified by the user.

CFC calculation block operations and text connection

- **Copy and paste**
  When copying and pasting CFC calculation blocks, the connection relationships of their input pins are also pasted accordingly, as detailed below:
  1. During copy, only the established connections on the input pins are copied.
  2. After pasting, the system searches the current scope for the pins or variables corresponding to the copied connections.
  3. If the target pin or variable is found, a connection is automatically established; if not found, a text connection mark is generated and a fault is reported. Once the target pin or variable becomes available, right-click the text connection mark and select "Text connection" to complete the connection.
- **Move**
  When moving a calculation block, its existing text connection marks move with it. Currently, connection relationships are preserved only when moving within the same POU.
- **Delete**
  When deleting a calculation block, its text connection marks are also deleted.

10.1.5.6 CFC Configuration and Commissioning

10.1.5.6.1 Execution Order Between CFC Blocks

The CFC program blocks are executed in the order of the POU calls as configured in the task configuration window. Users can adjust the execution order of the program blocks. For details, see Call POU.

10.1.5.6.2 Execution Order of Operation Blocks within a CFC Block

The operation blocks in a CFC program block are executed in numerical order. Users can adjust the execution order of the operation blocks. For details, see 10.1.5.4.3.2 Adjusting Operation Block Execution Order.

10.1.5.6.3 CFC Instance Call

A CFC instance call allows dragging one program block into the programming area of another program block. The call relationships are as follows:

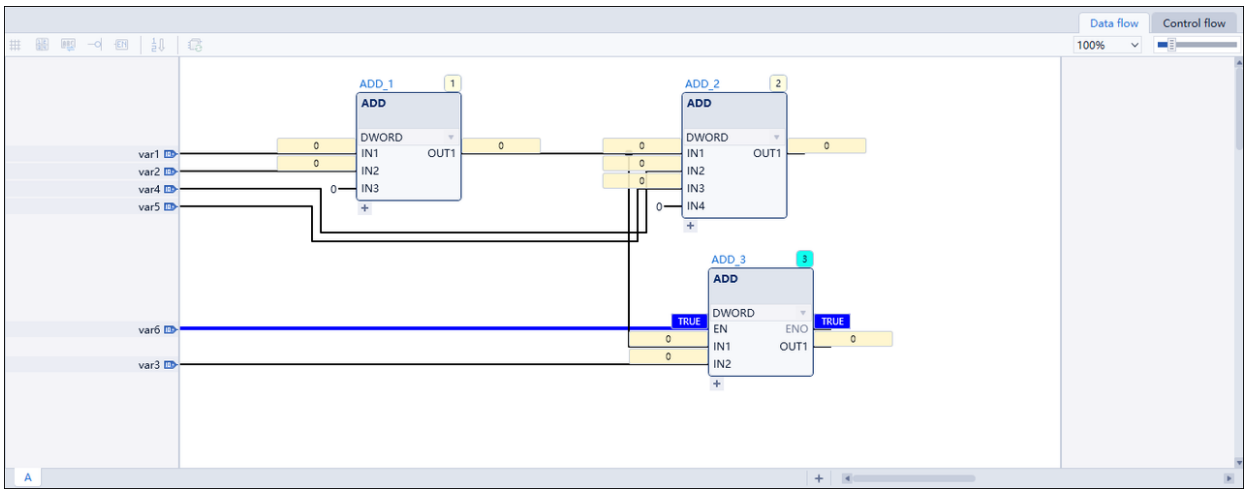
- A CFC program block can call FB and FC program blocks written in LD or ST language under the "Programming" node, but cannot call PRG program blocks.
- A CFC program block cannot call methods, actions, or properties under FB.
- A CFC program block cannot call other PRG program blocks written in non-CFC languages.
- A CFC program block cannot call SFC programs.
- Program blocks written in other languages under the "Programming" node cannot call CFC program blocks.

10.1.5.6.4 CFC Commissioning

Continuous function charts support monitoring, operand writing, and forced value writing in online mode.

Monitor

When monitoring programs in online mode, the online values of operation block pins can only be displayed in real-time with the "Monitorable" property set to "Yes".



NOTE

For the configuration of operation block properties, see 10.1.5.4.3.1 Configuring Operation Block Properties.

Writing and forcing values

In online mode, writing and forcing values to variables are supported. The procedure is as follows:

1. Before writing or forcing values, modify the prepared values. There are two methods:
   - Method 1: Modifying in the program editing area
     For BOOL type variables, double-click the variable value position to switch between TRUE and FALSE.
     For non-BOOL type variables, double-click the variable value position to open the "Prepared value" window and enter the prepared value.
   - Method 2: Modifying in the variable declaration area
     For BOOL type variables, double-click the variable prepared value position to switch between TRUE and FALSE.
     For non-BOOL type variables, double-click the variable prepared value position and enter the prepared value.
2. After modifying the prepared value, you can perform write or force operations. For details, see 10.10.3 Numerical Display and Calculation. After forcing a value, you can cancel the forced state by releasing the value.

10.1.6 SFC Language Programming

10.1.6.1 SFC Language Overview

Sequential function chart (SFC) is a graphical programming language that follows the IEC 61131-3 standard. It uses graphical elements (steps and transitions) to intuitively describe control logic, reducing programming complexity and making it easier for engineers to locate issues. It is primarily used in industrial automation and process control.

In SFC, the control flow is divided into multiple independent steps, each representing a specific operational state. Steps are connected by transition conditions, which define when the flow moves from one step to the next. This modular and structured design simplifies the management of complex systems, helping developers build stable, flexible, and easily maintainable automated control systems.

10.1.6.2 SFC Creation and Management

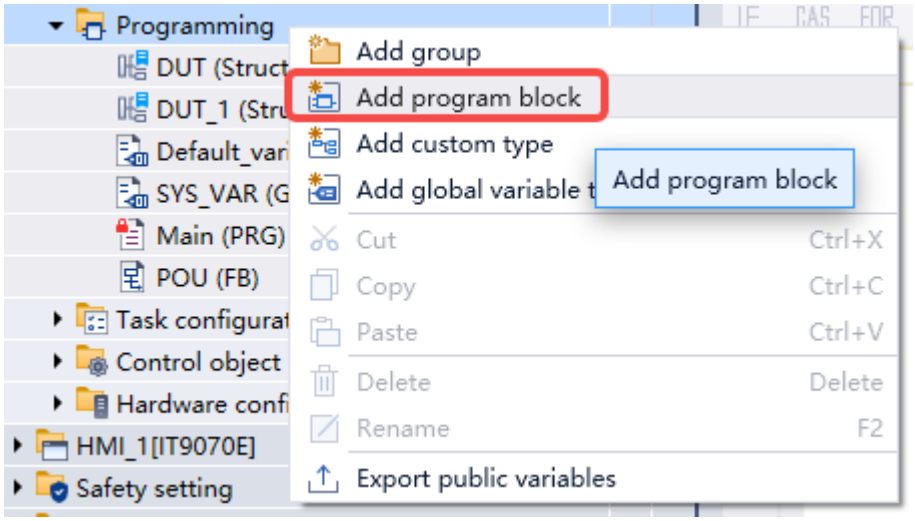
10.1.6.2.1 Creating an SFC Program Block

Prerequisites

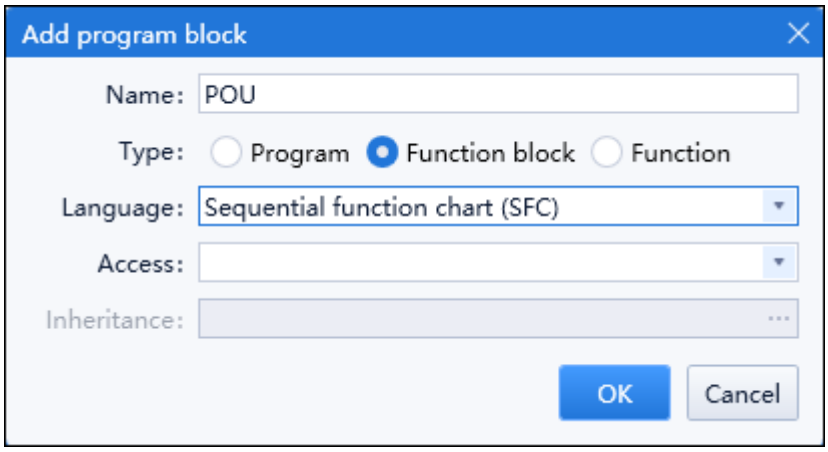
- A project has been created.
- A device has been added.

Procedure

1. Right-click "Programming" in the project tree and select "Add program block".



2. In the pop-up window, set the name of the program block, select "Function block" for the type, and select "Sequential function chart (SFC)" for the IEC language, as shown in the figure below.



3. Click "OK" and the SFC editor will be automatically opened.

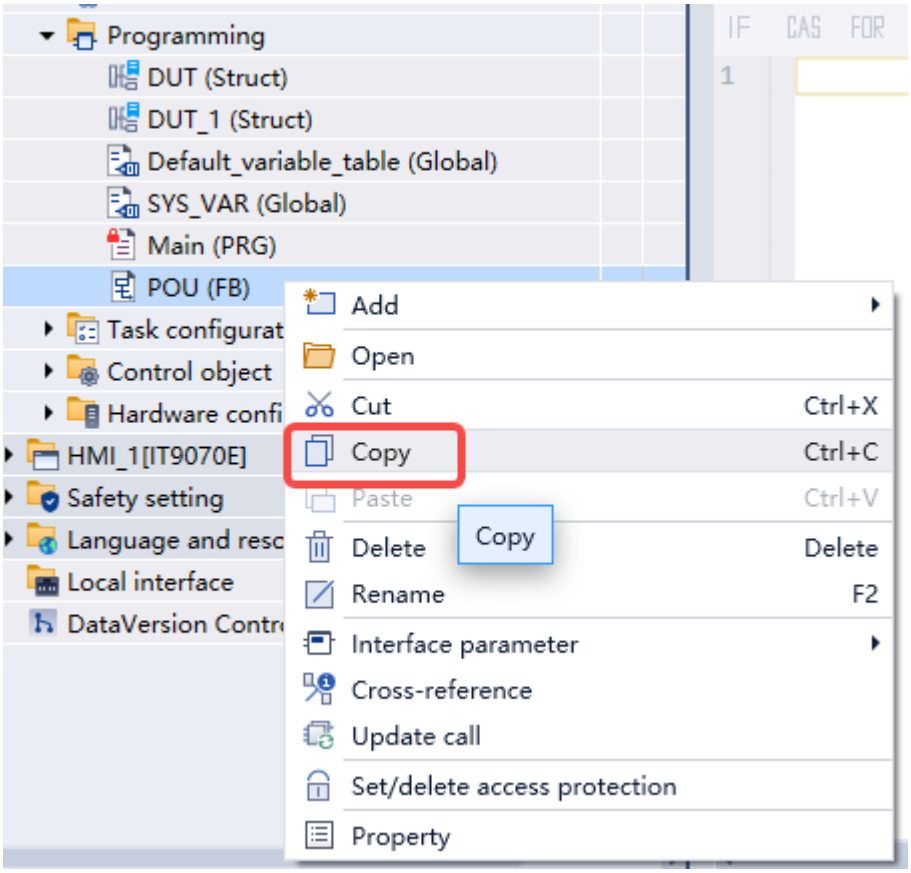
> NOTE
>
> When creating a program block, the SFC programming language is only supported when the program block type is set to "Function block".

10.1.6.2.2 Managing an SFC Program Block

10.1.6.2.2.1 Copying an SFC Program Block

SFC program blocks can be copied and pasted within a PLC device, between PLC devices in the same project, or between different projects. The internal program logic of the SFC program block remains unchanged after pasting.

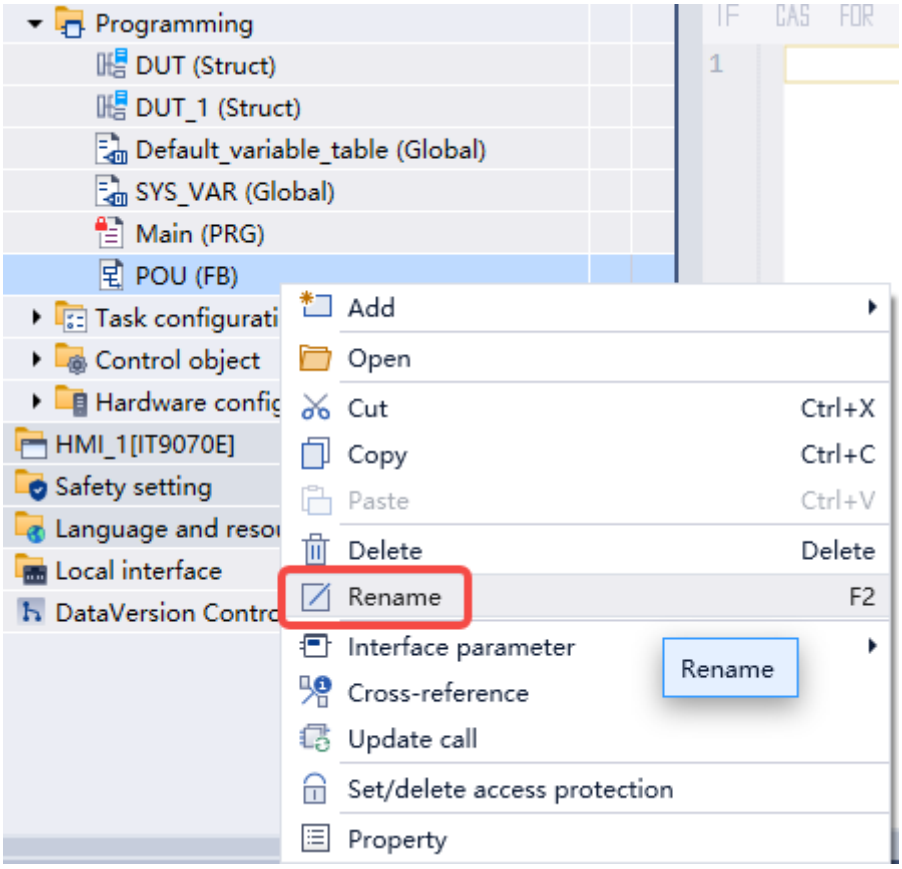
1. Right-click an SFC program block name under the "Programming" node, and select "Copy".



2. Right-click the "Programming" node or its subordinate node, and select "Paste" to paste the copied LD program block at the corresponding location.

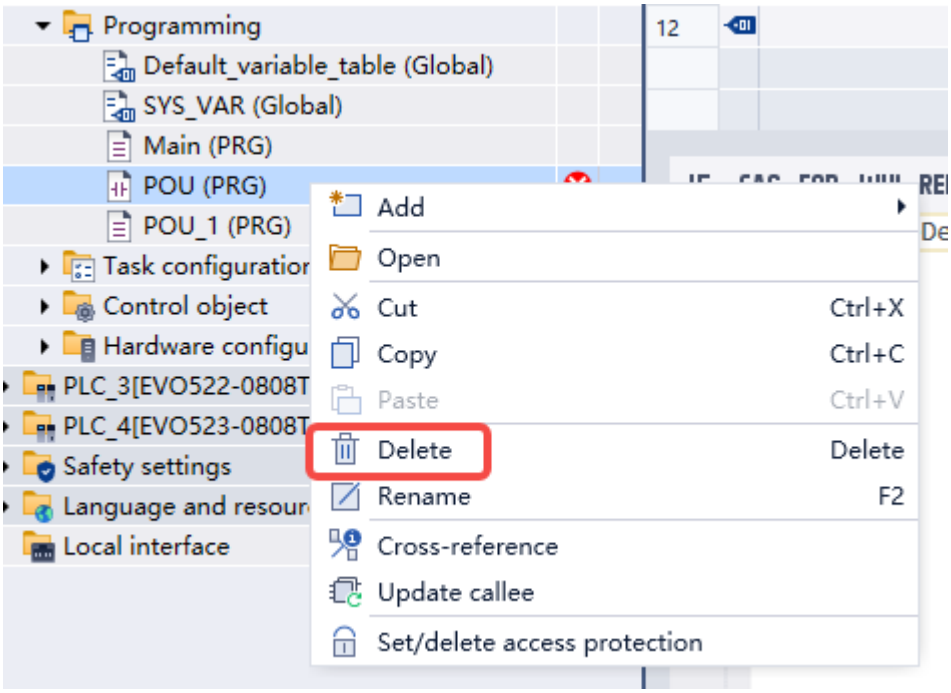
10.1.6.2.2.2 Renaming an SFC Program Block

Right-click an SFC program block name under the "Programming" node, select "Rename", and modify the group name.



10.1.6.2.2.3 Deleting an SFC Program Block

1. Right-click an SFC program block name under the "Programming" node, and select "Delete".

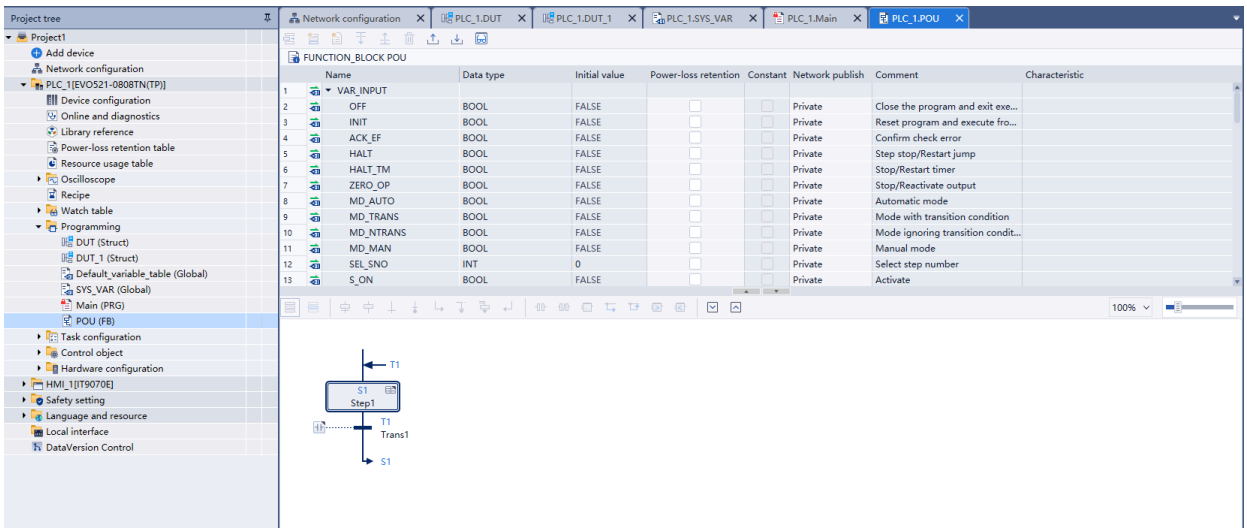


2. Click "OK" in the pop-up window.

10.1.6.3 SFC Editor

10.1.6.3.1 SFC Editor Interface

Double-click the SFC language program block (identifier:) under the "Programming" node.



The SFC editor is divided into the variable declaration area and the program editing area, which can be collapsed and expanded using and .

The program editing area consists of the toolbar and the programming area, as shown in the following figure:

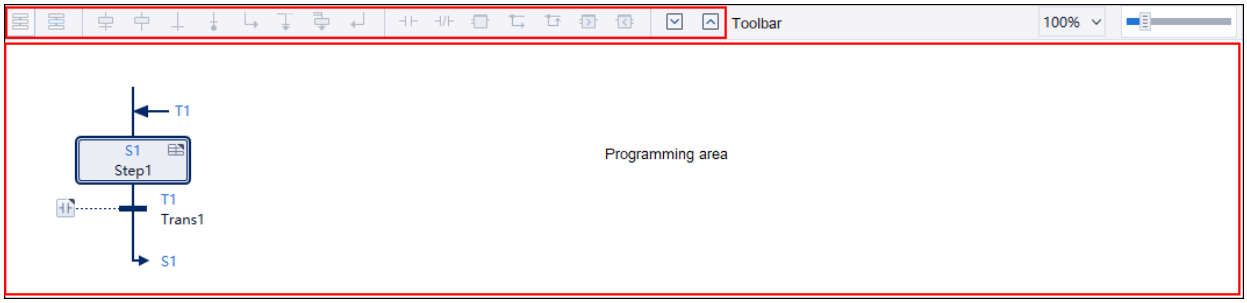


Table10-1-6-3-1-1 Introduction to toolbar

Description	Icon	Description	Application Scope
Program view		Switch to program view	Program view and step view
Step view		Switch to step view	
Step and transition		Add a step and transition below	SFC program view
Step		Add a step below	
Transition		Add a transition below	
Insert an end		Add an end	
Insert a jump		Add a jump	
Selective branch		Add a selective branch	
Parallel branch		Adding a parallel branch	
Branch closing		Add a closed branch	
Insert NO contact		Add a normally open contact	Switch edit mode, inter-lock edit mode, monitoring edit mode
Insert an NC contact		Add a normally closed contact	
Insert an empty operation block		Add an empty operation block	
Branch disconnected		Branch disconnected	
Branch closing		Branch closing	
Insert a greater-than block		Insert a greater-than block	
Insert a less-than block		Insert a less-than block	
Opening all program segments		Expand all program segments in the "Program view"; expand all editing areas in the "Step view"	Program view and step view

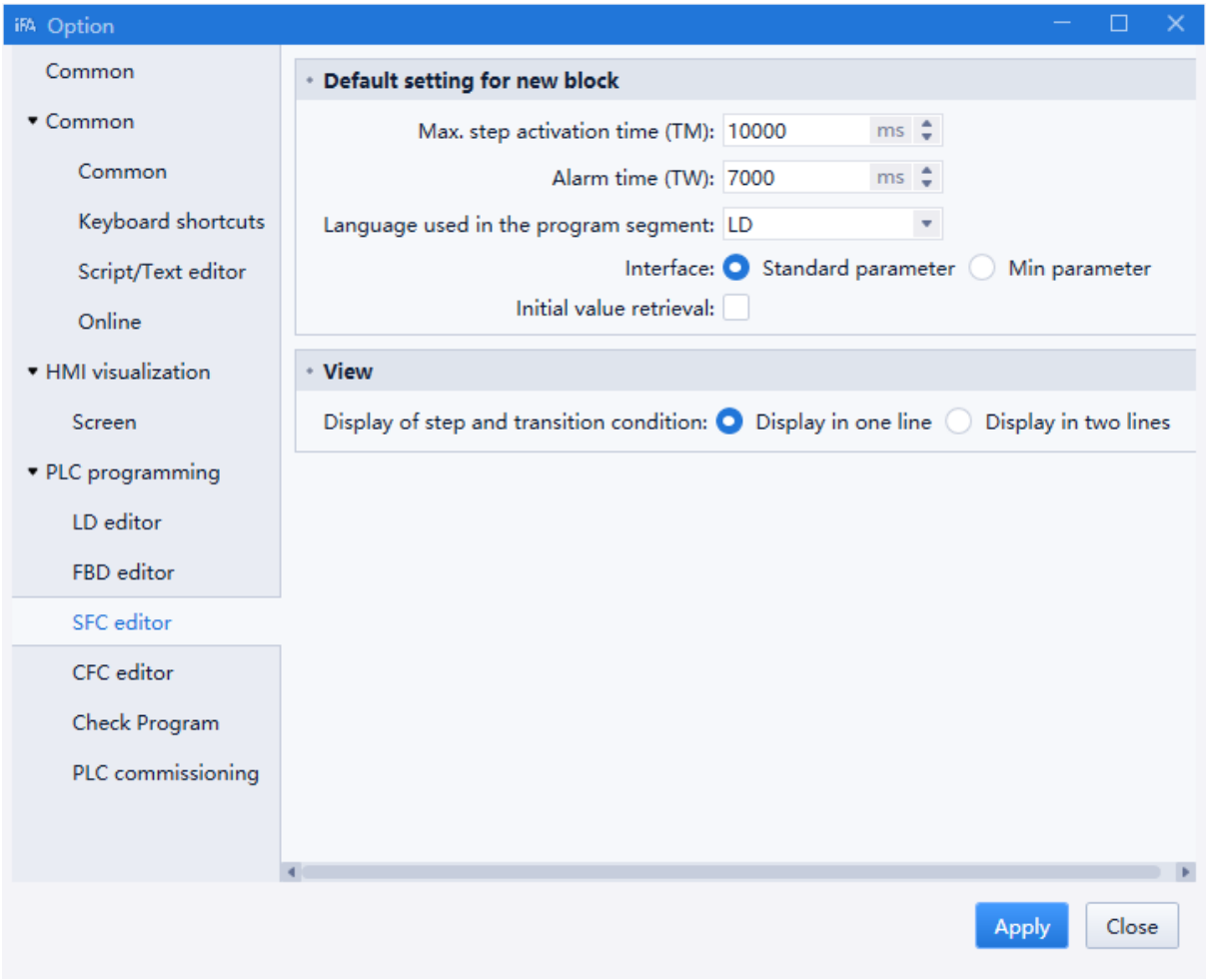
Description	Icon	Description	Application Scope
Close all program segments		Collapse all program segments in the "Program view"; collapse all editing areas in the "Step view"	

10.1.6.3.2 SFC Editor Settings

10.1.6.3.2.1 Setting the SFC Editor in iFA Options

The SFC editor parameters in iFA options are global. They affect all SFC blocks in every project.

1. Select "Tools > Option" from the iFA menu bar.
2. In the "Option" window, configure the related parameters for the SFC editor in the "SFC editor" tab under "PLC programming".



- **Default setting for new block**

Parameter	Description
Maximum step activation time	The maximum time a step is allowed to remain active. It ranges from 0 to 3600000 ms with a default of 10000 ms, and is editable by users.

Parameter	Description
Alarm time	When a step is activated, if its execution time exceeds the set alarm time but does not reach the maximum step activation time, the system issues an alarm but does not interrupt program execution. It ranges from 0 to 3600000 ms with a default of 7000 ms, and is editable by users.
Language used in the program segment	Supports only LD language.
Default interface parameters	• Standard parameters: The user program calling the SFC program block needs to control or externally influence the execution mode and flow of the SFC program. • Minimum parameters: The user program calling the SFC function block does not need to control or externally influence the execution mode and flow of the SFC program.
Initial value retrieval	Records the 32 boolean values of the step's condition, stored in the step's static parameter CD_STAT. Records the 32 boolean values of a transition (e.g., Trans1), stored in the transition's static parameter TRANS.

NOTE

In the default settings for new blocks, parameter modifications apply only to newly created SFC programs, while existing programs keep their original configuration.

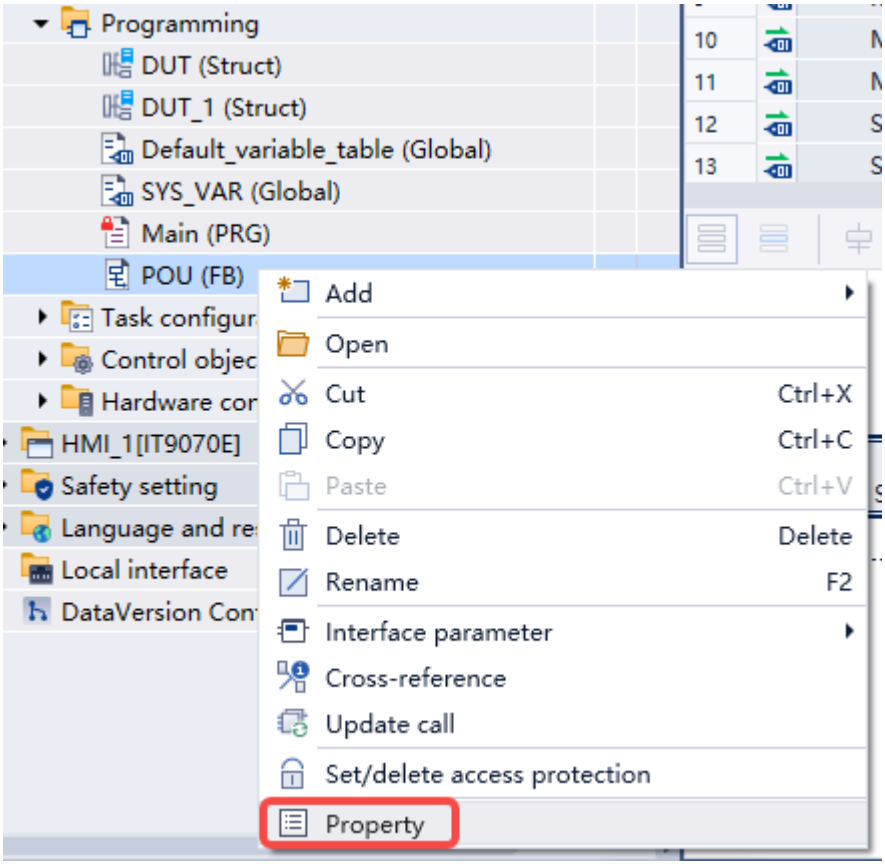
- **View**

Parameter	Description
Step and transition name length	Sets whether step and transition names are displayed in one line or two lines. Select "Display in two lines" to view longer names.

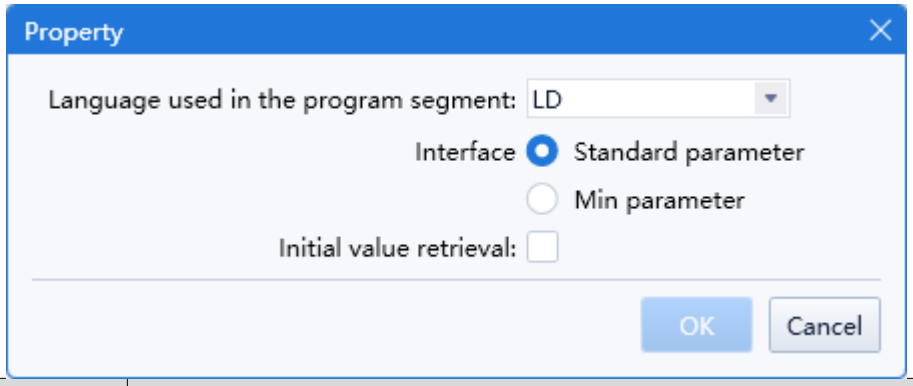
10.1.6.3.2.2 Setting SFC Program Block Properties

After setting the properties of the SFC program block, the settings only apply to the current SFC program block.

1. Right-click the SFC language program block under the "Programming" node in the project tree, and select "Property".



2. In the "Property" dialog box, configure the parameters related to the current SFC program block, and click "OK".

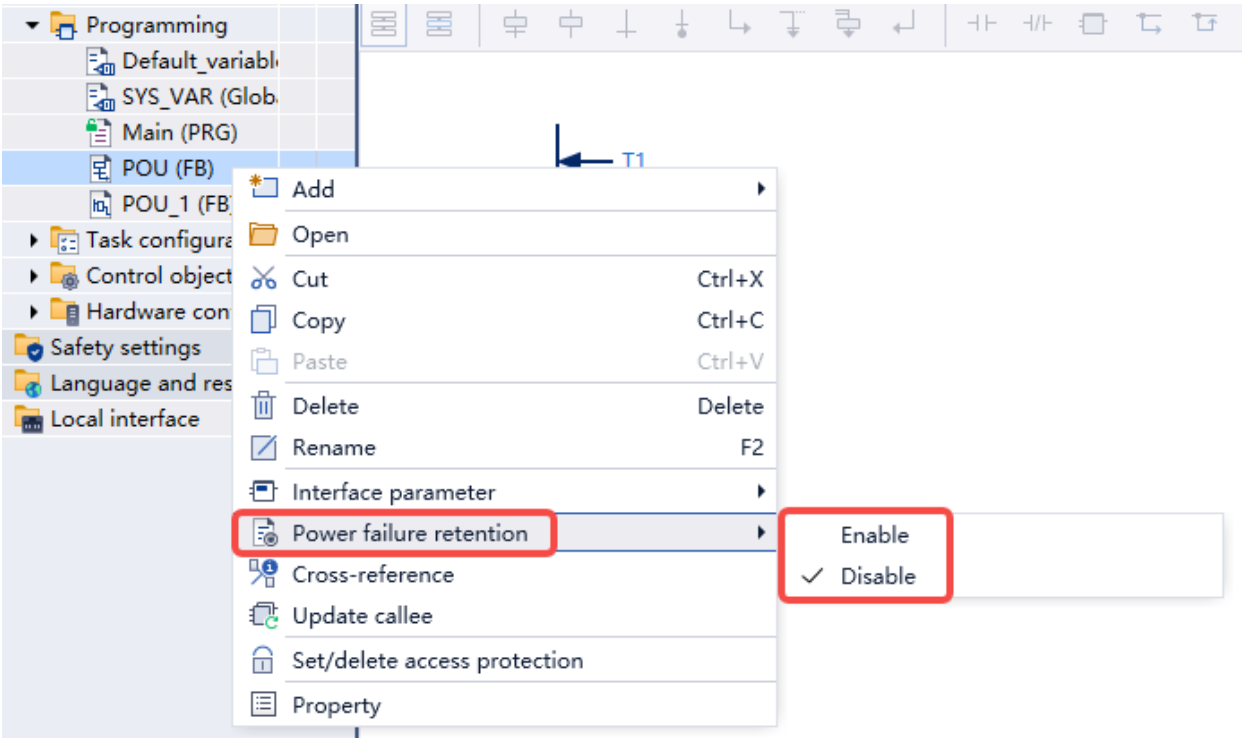


Parameter	Description
Language used in the program segment	Supports only LD language.
Interface	◦ Standard parameter: The user program calling the SFC program block needs to control or externally influence the execution mode and flow of the SFC program. ◦ Minimum parameters: The user program calling the SFC function block does not need to control or externally influence the execution mode and flow of the SFC program.
Initial value retrieval	Records the 32 boolean values of the step's condition, stored in the step's static parameter CD_STAT. Records the 32 boolean values of a transition (e.g., Trans1), stored in the transition's static parameter TRANS.

10.1.6.3.2.3 Setting SFC Program Block Power Failure Retention Property

In an SFC program block, the power failure retention property of automatically generated system parameters is not checked by default and cannot be directly modified in the variable table. To modify this property, follow the steps below:

Right-click the SFC language program block under the "Programming" node in the project tree, select "Power failure retention", and select "Enable" or "Disable" as needed.



NOTE

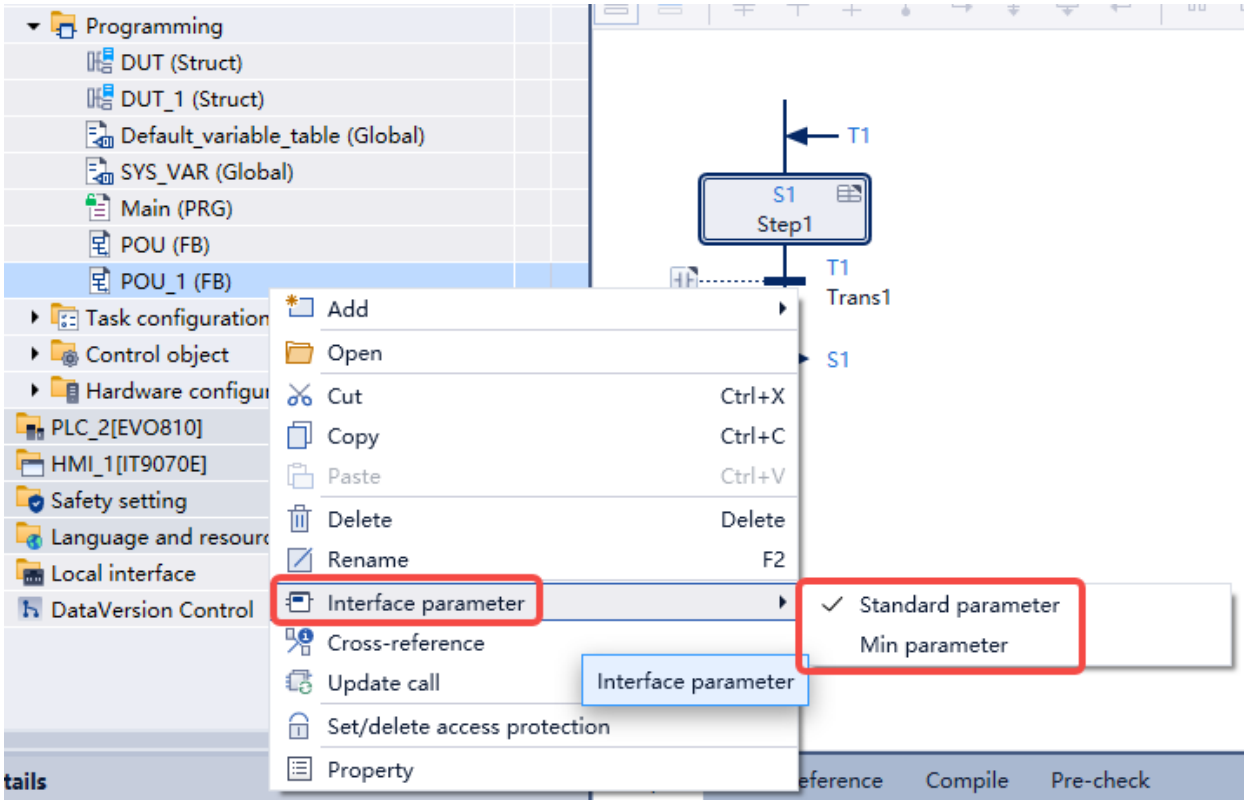
This operation applies only to the system-generated default parameters. For user-defined parameters, this property can be modified directly in the variable table.

10.1.6.3.2.4 Setting SFC Program Block Interface Mode

The SFC program block supports two interface modes: "Standard parameter" and "Min. parameter". The mode can be set as follows:

Right-click the SFC language program block under the "Programming" node in the project tree, select "Interface parameter", and then select "Standard parameter" or "Min. parameter" as needed.

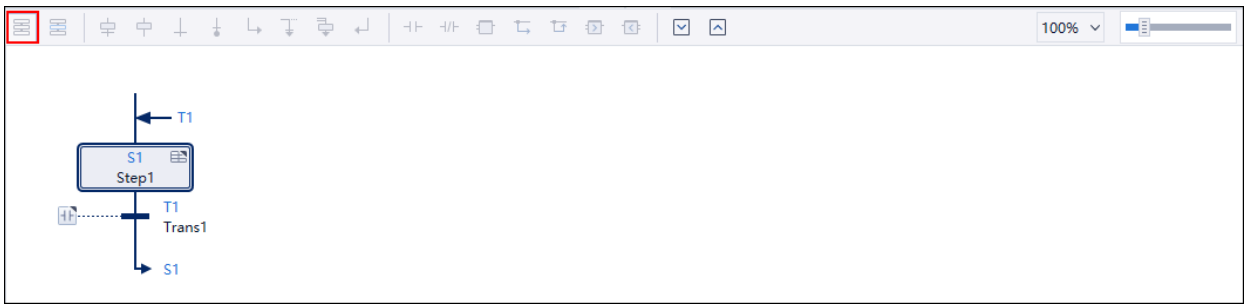
- Standard parameter: The user program calling the SFC program block needs to control or externally influence the execution mode and flow of the SFC program.
- Min. parameter: The user program calling the SFC function block does not need to control or externally influence the execution mode and flow of the SFC program.



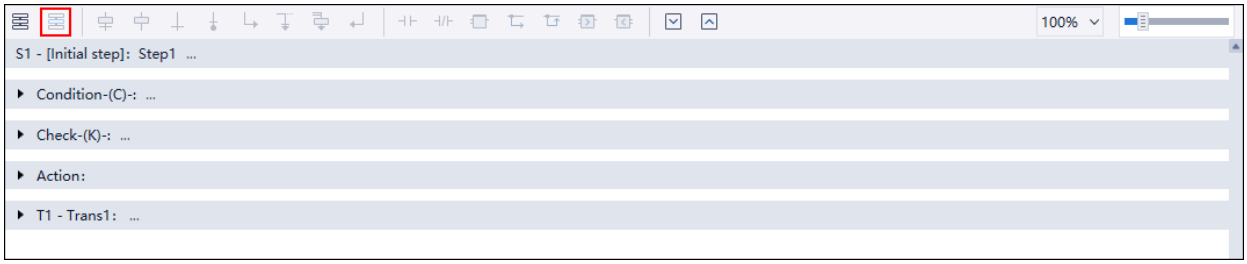
10.1.6.3.3 SFC Programming Window

Window introduction

- Program view
  After an SFC program is created, iFA automatically opens the program view of the SFC editor. Core elements such as steps, transitions, branches, jumps, and ends can be added in the program view to complete the SFC main program. In step view, you can switch to program view by clicking the "Program view" icon on the toolbar.

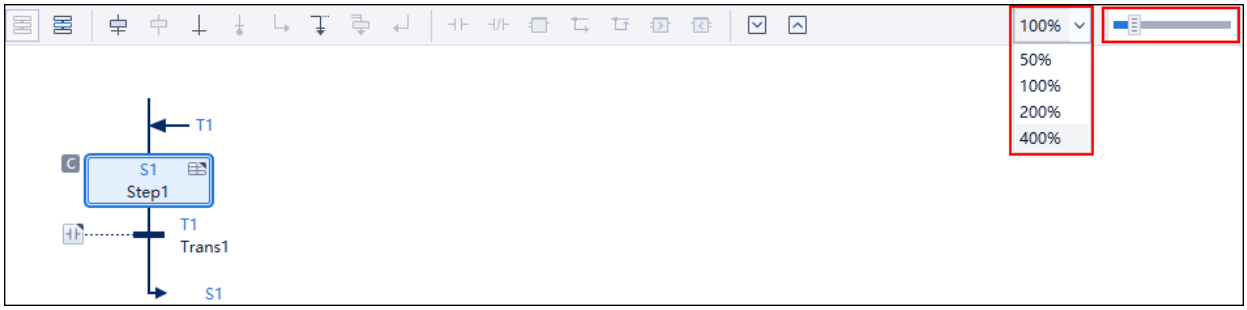


- Step view
  In the program view, select the target step in the SFC program, click the "Step view" icon on the toolbar to enter the editor for the current step. The interface is divided into: condition editing area, monitoring editing area, action editing area, and transition editing area.
  - Condition/Monitoring/Transition editing area: Users can write logic conditions using LD language elements and comparison instructions.
  - Action editing area: Users can define the operations to be completed in the current step.



Window operations

- Scaling
  In the upper-right corner of the program editing area, adjust the display size of the content using the drop-down menu or the slider.



- Offset
  - When the content exceeds the programming area, scrollbars appear in the window; drag the scrollbar to pan the view.
  - When the content exceeds the programming area, hold down the right mouse button and drag within the programming area to pan the view.

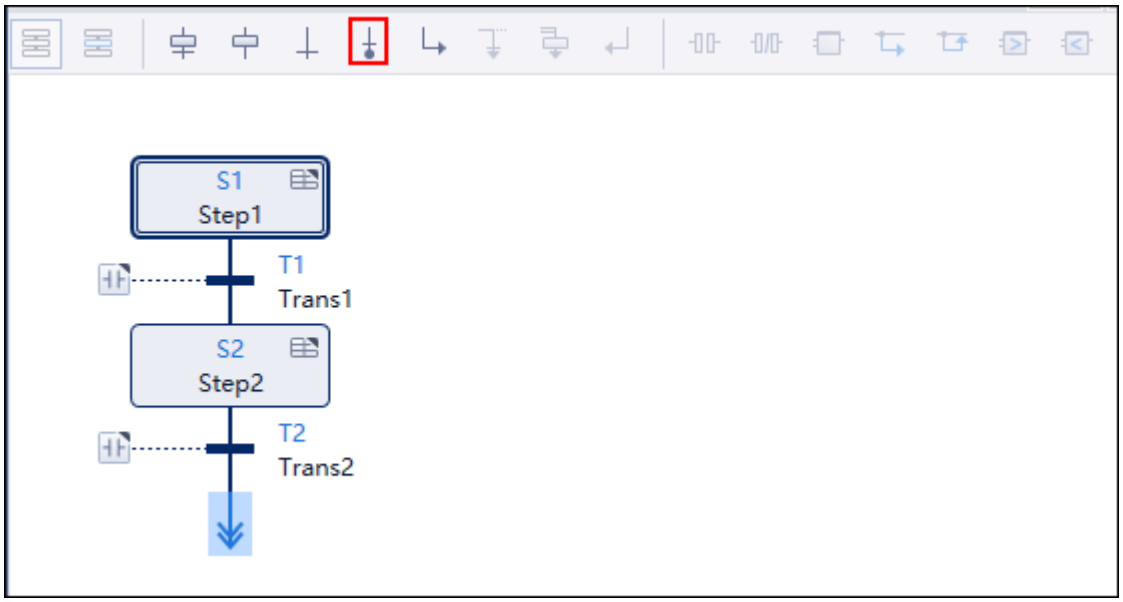
10.1.6.3.4 SFC Elements

Users can select SFC and LD elements through various methods such as the right-click menu, SFC editor toolbar, or the toolbox on the right. For illustration, this section uses toolbar operations as an example. In actual programming, users can flexibly choose the addition method. Other methods will not be elaborated.

SFC elements

- **Step and transition**
  For details, see 10.1.6.5.1 Adding a Step and Transition.
- **Step**
  For details, see 10.1.6.5.2 Adding a Step.
- **Transition**
  For details, see 10.1.6.5.3 Adding a Transition.
- **Insert an end**
  Indicates the end of the SFC POU, where execution stops.
  Select one of the  in the program, click  from the toolbar to insert an end, as shown in the figure

below.



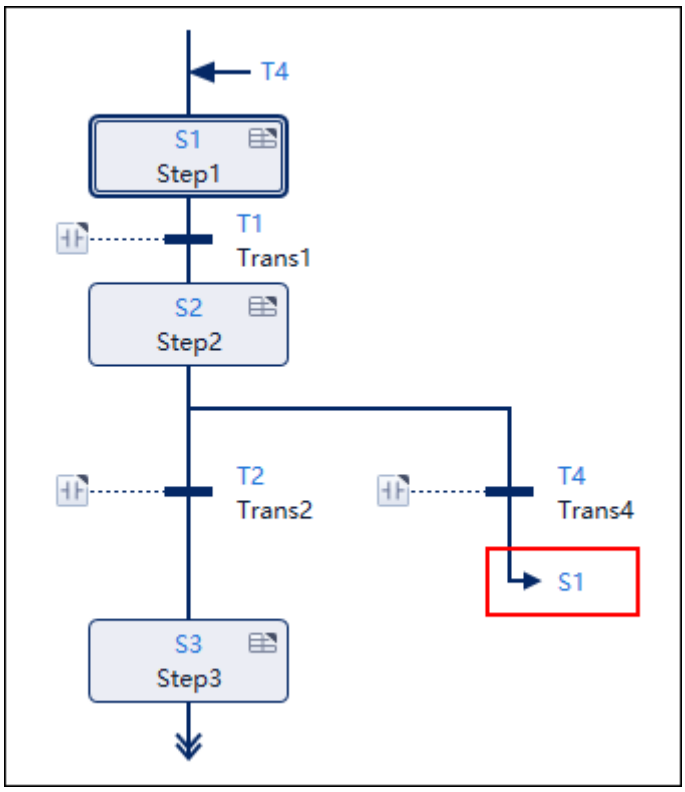
- **Insert a jump**

  By using the jump function, you can jump from any step to a specified step for program loop processing.

> NOTE
>
> A transition must exist before a jump; if missing, please insert a transition first.

1. Select one of the in the program, and click on the toolbar.
2. Select the target step. The target step is a step within this program block (POU).

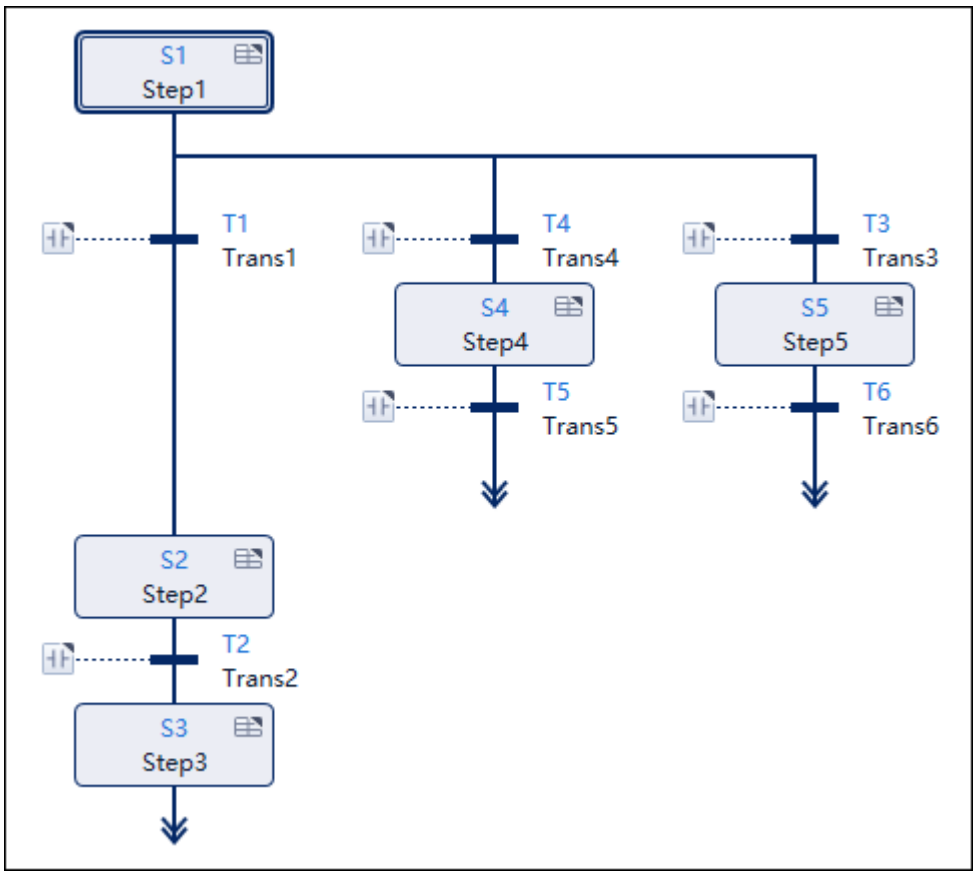
As shown in the figure below, if the Trans4 condition is met and Step2 is completed, the program jumps to Step1.



- **Selective branch**

  The logical relationship between multiple selective branches is OR, meaning the program executes only one of them. If multiple conditions are met simultaneously, execute the leftmost branch.

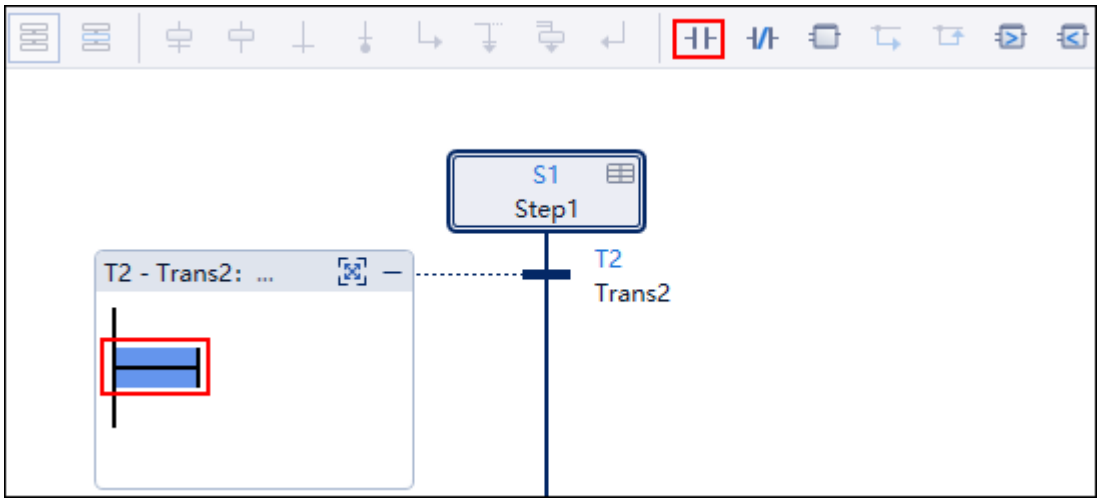
  As shown in the figure below, if both Trans1 and Trans4 conditions are met, Step1 is executed, followed by Step2. See 10.1.6.6.1 Adding a Selective Branch for details.



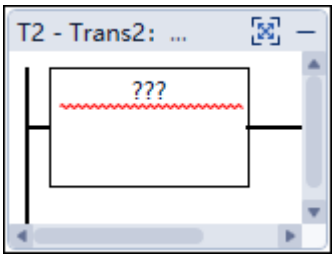
- **Parallel branch**
  The logical relationship between multiple parallel branches is AND, meaning the program executes all branches simultaneously. See 10.1.6.6.2 Adding a Parallel Branch for details.
- **Closed branch**
  Mainly used to end branch tasks and return to the main process. See Returning to Parent Branch for details.

LD elements

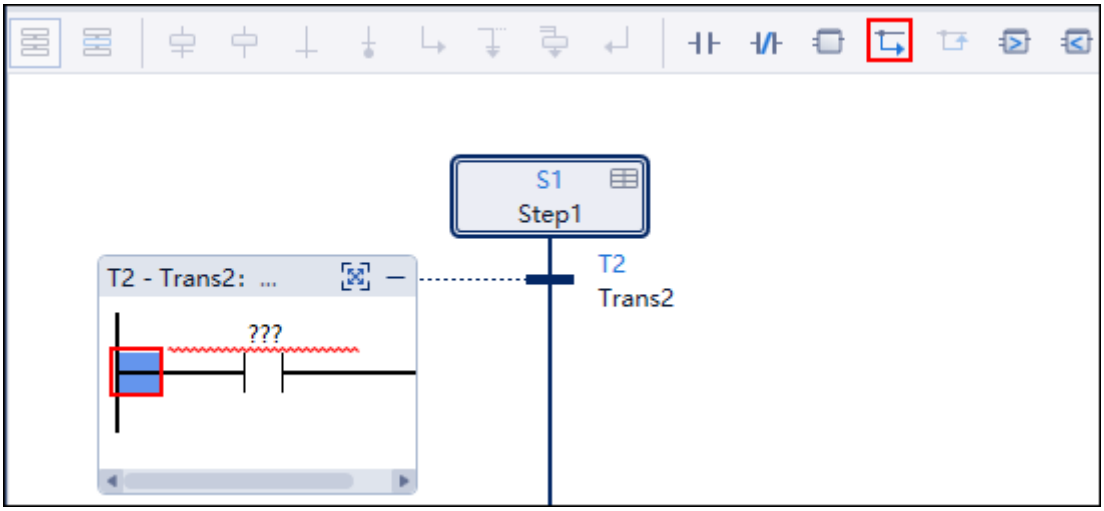
- **Insert NO contact**
  A normally open contact is open under normal conditions, and its state is FALSE.
  In the LD programming area, select the connection line, and click from the toolbar, as shown in the following figure.



- **Insert NC contact**
  A normally closed contact is closed under normal conditions, and its state is TRUE.
  In the LD programming area, select the connection line, and click from the toolbar, as shown in the following figure.
- **Insert empty operation block**
  Used to insert an empty operation block. It is suitable for users who are familiar with functions and function blocks.
  1. In the LD programming area, select the connection line, and click from the toolbar.
  2. In the operation block, directly enter the function or function block name in the "???" field.

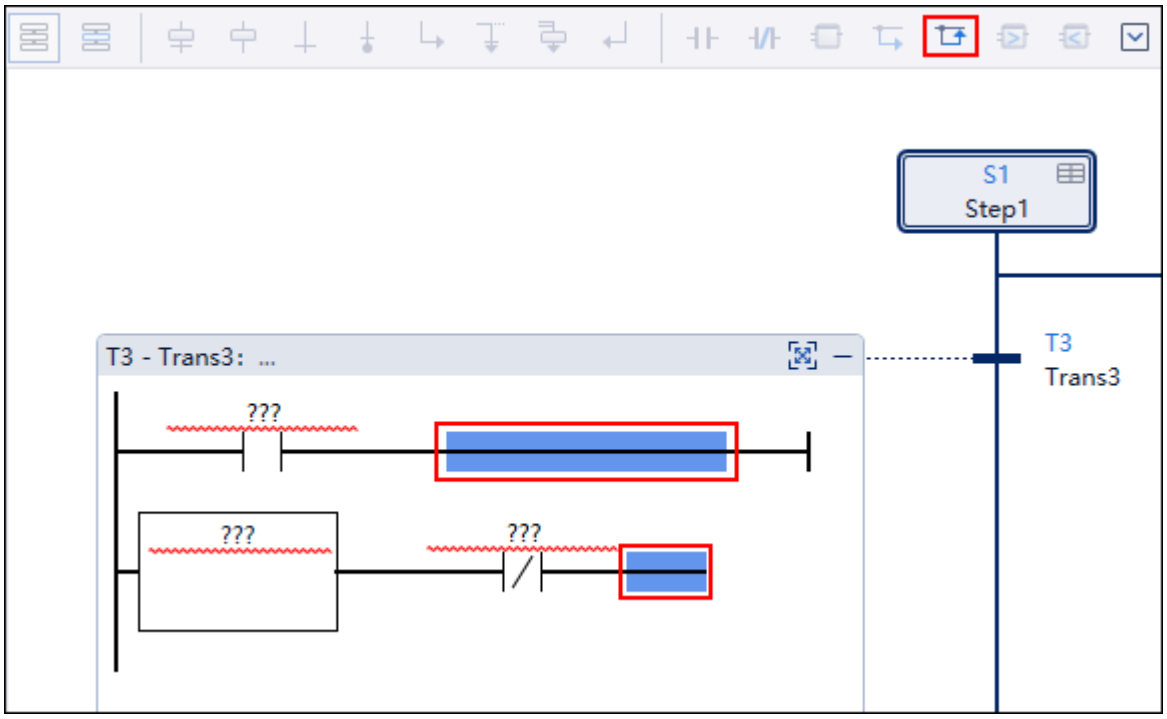


- **Open branch**
  Select a connection line in the LD programming area and click from the toolbar to insert a branch below the selected line. The inserted branch is an open line.



- **Close branch**

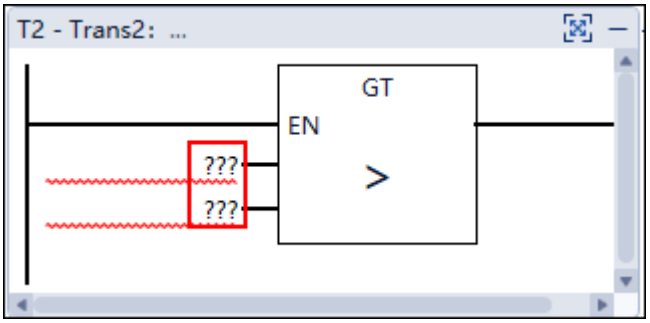
  Select a branch start point in the LD programming area, then hold down the Ctrl key to select the branch end point, and click from the toolbar to connect the branch start and end points.



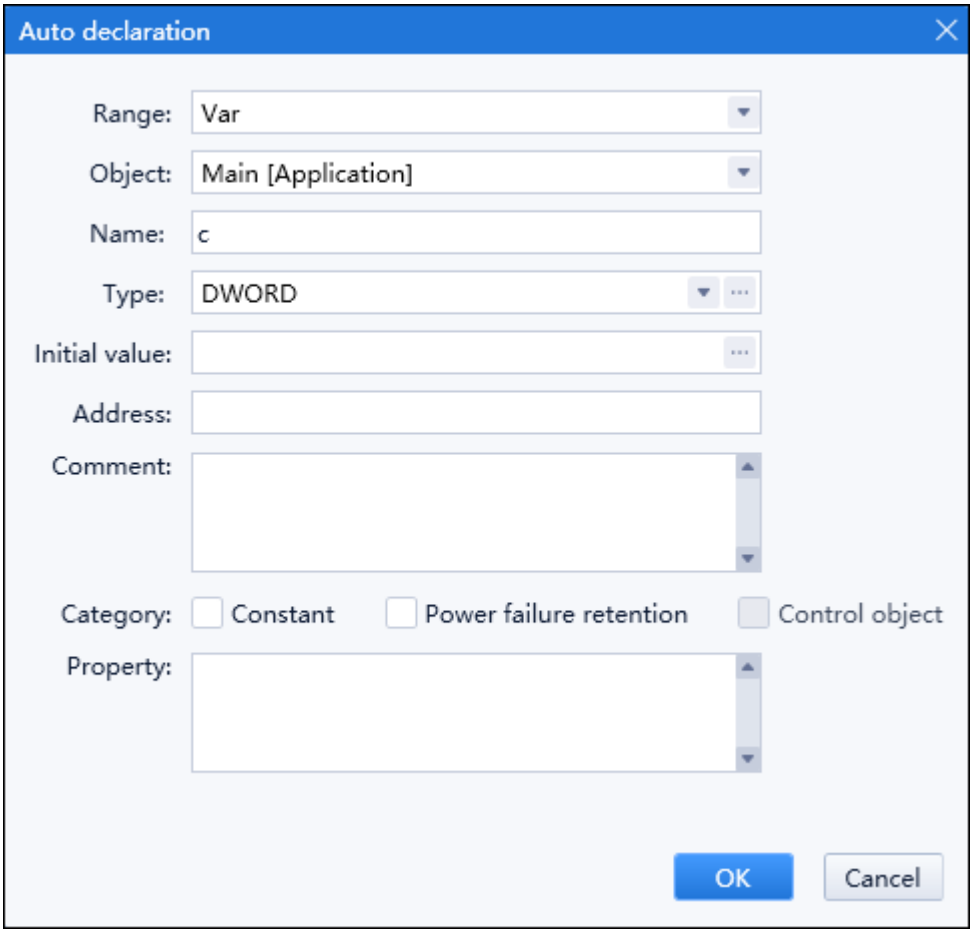
- **Insert a greater-than block**

  1. In the LD programming area, select the connection line, and click from the toolbar.

  2. Enter or select an expression at the input pin of the greater-than operation block.

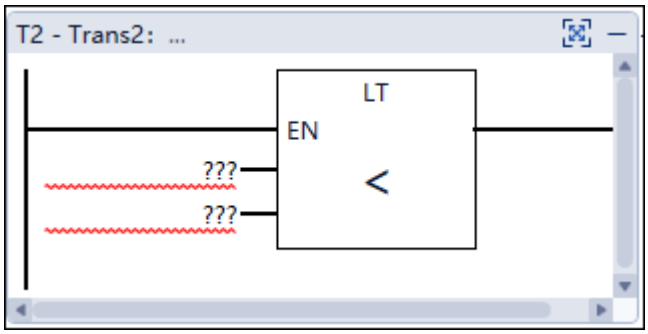


  3. (Optional) User-defined variables require configuration of scope, type, initial value, category, and so on.

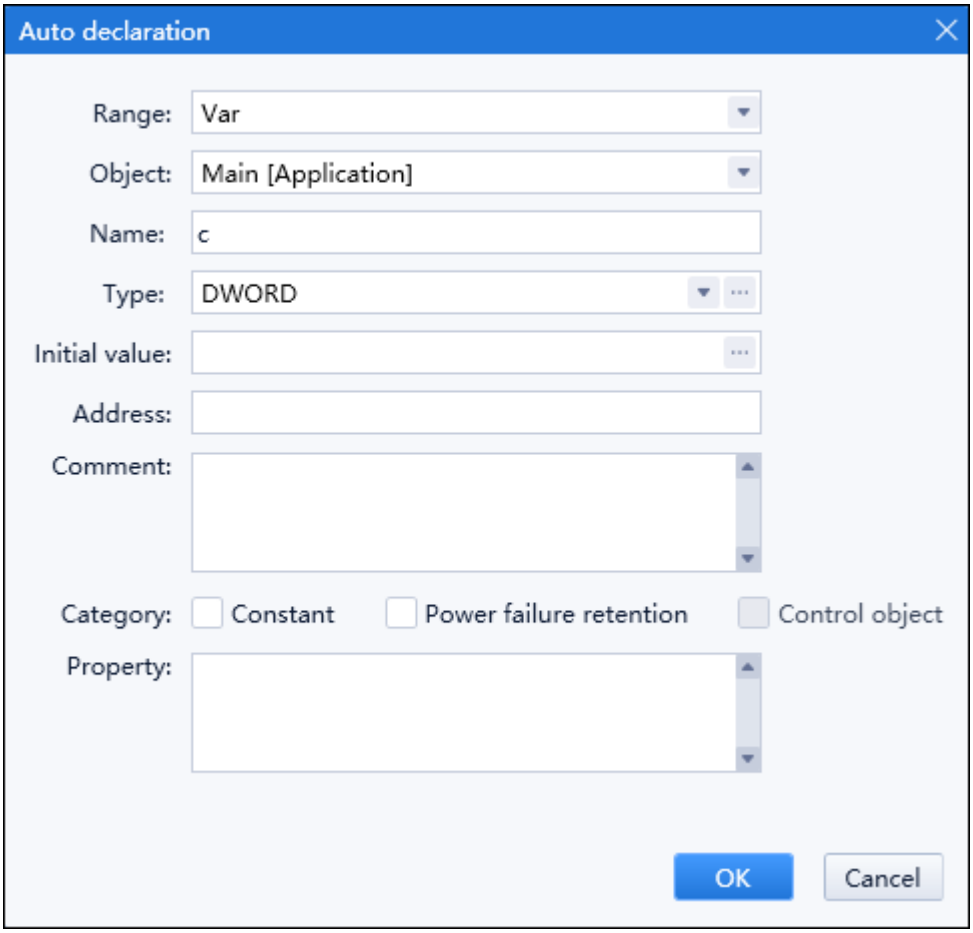


- **Insert a less-than block**

  1. In the LD programming area, select the connection line, and click from the toolbar.
  2. Enter or select an expression at the input pin of the less-than operation block.



  3. (Optional) User-defined variables require configuration of scope, type, initial value, category, and so on.

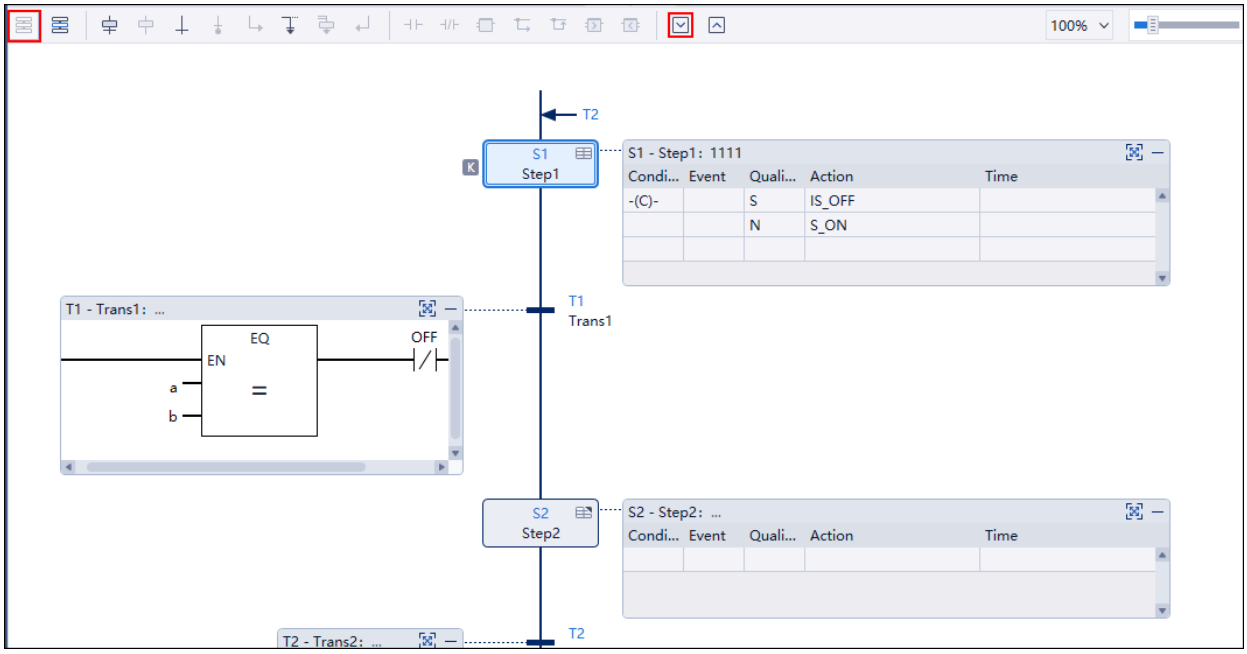


10.1.6.3.5 Opening/Closing All Program Segments

Opening all program segments

- Program view

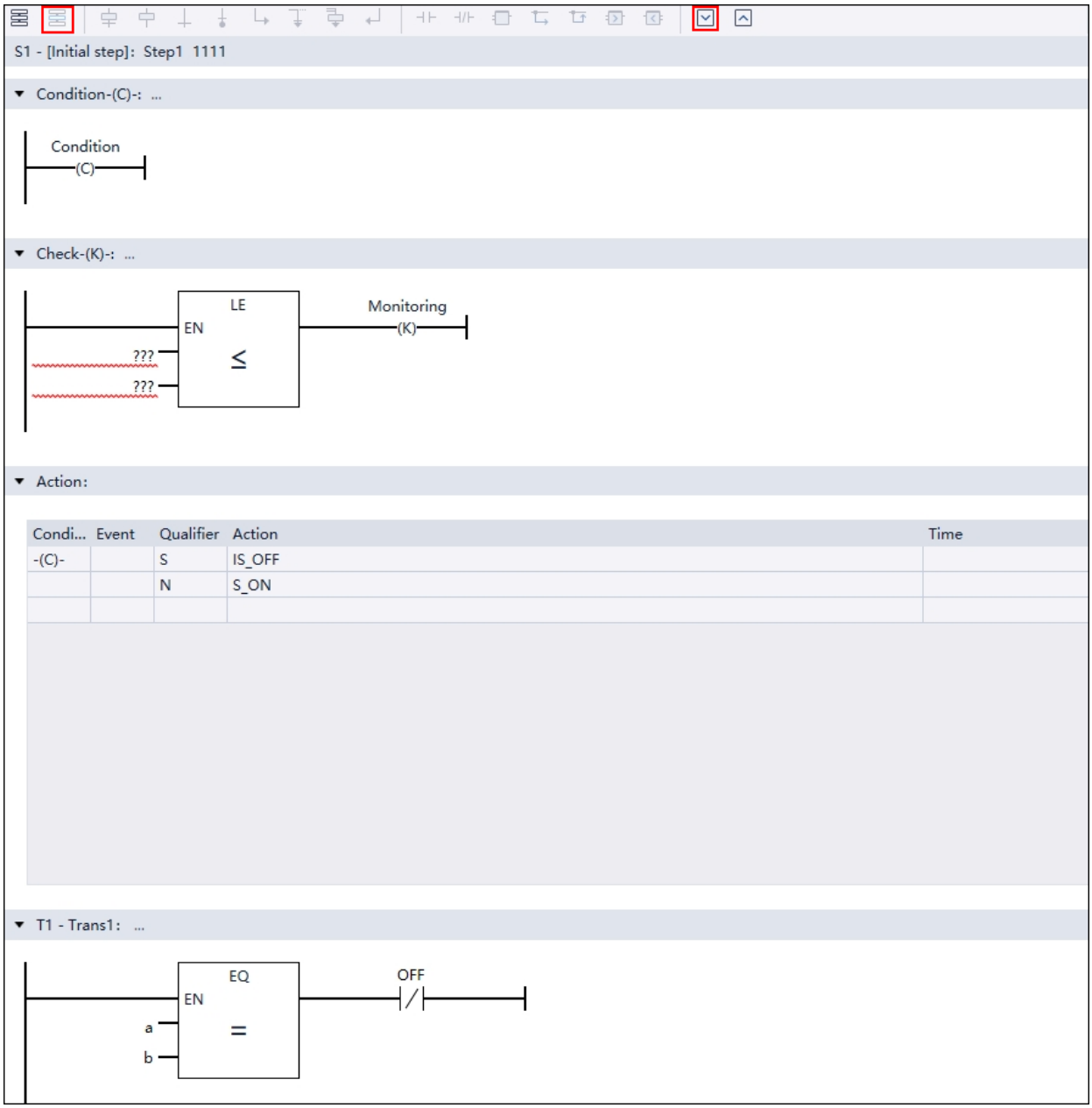
  In the program view, click the "Open all program segments" icon on the toolbar to expand the actions and transition logic for all steps in the program editing area.



- Step view

  In the step view, click the "Open all program segments" icon on the toolbar to expand all content in

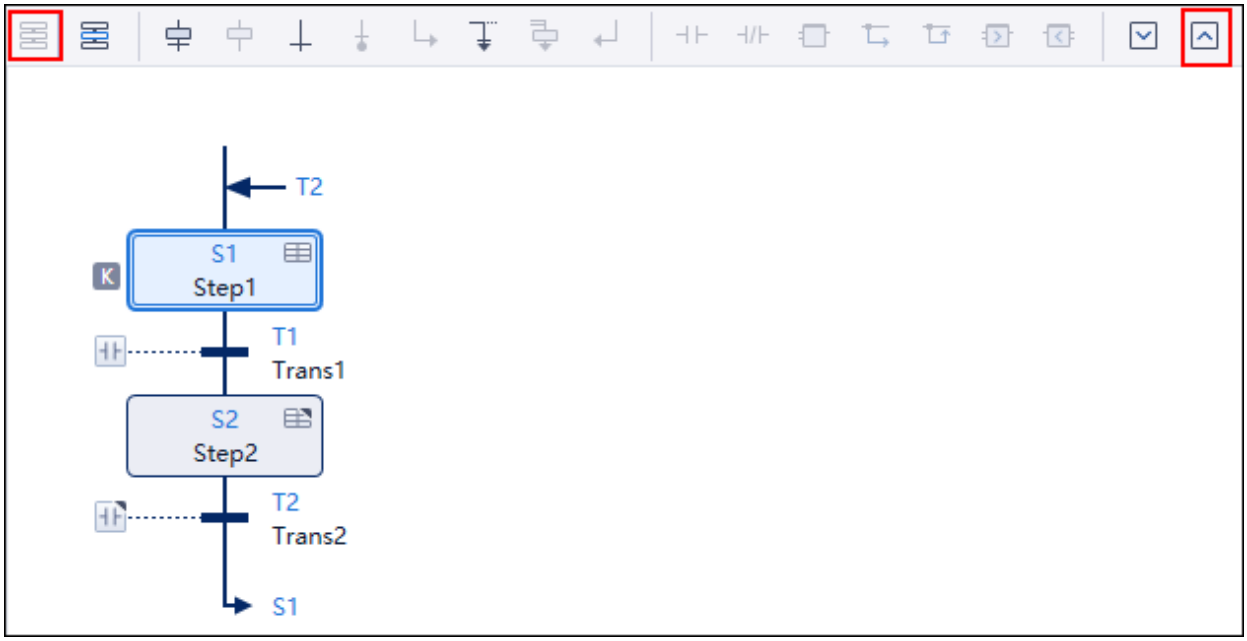
the condition editing area, monitoring editing area, action editing area, and transition editing area for the current step.



Close all program segments

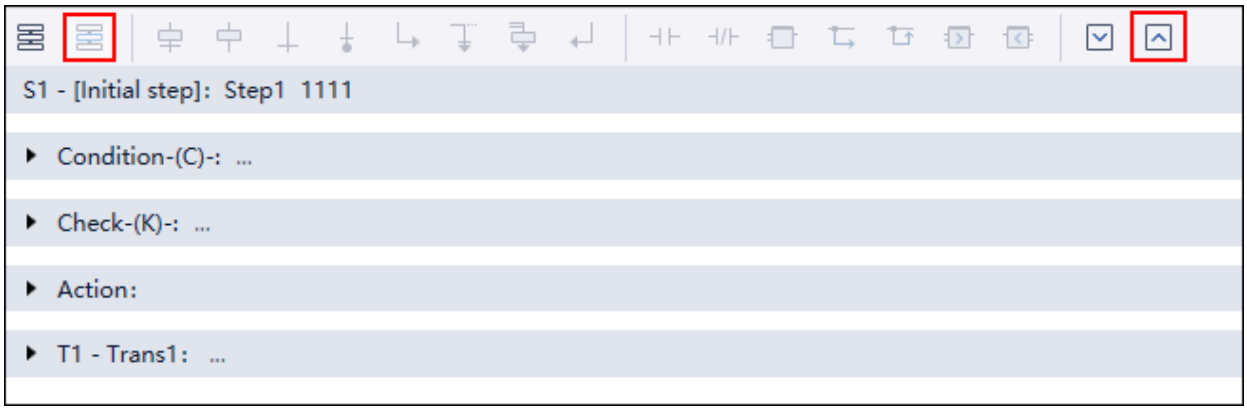
- Program view

  In the program view, click the "Close all program segments" icon on the toolbar to collapse the actions and transition logic for all steps in the program editing area, restoring it to the initial state.



- Step view

  In the step view, click the "Close all program segments" icon on the toolbar to collapse the condition editing area, monitoring editing area, action editing area, and transition editing area for the current step.



10.1.6.3.6 SFC Program Block Interface Parameters

10.1.6.3.6.1 Overview

External programs can dynamically adjust the execution mode and logic of the SFC program through the program block interface parameters.

10.1.6.3.6.2 Interface Parameter Mode

The SFC program interface supports two modes (minimum mode and standard mode), corresponding to different scenarios. System-generated Input-type interface parameters take effect through a rising edge (FALSE > TRUE).

You can set the interface mode of the SFC program in various ways. For details, see 10.1.6.3.2 SFC Editor Settings.

- Minimum mode
  The user program calling the SFC function block does not need to control or externally influence the execution mode and flow of the SFC program.
- Standard mode
  The user program calling the SFC program block needs to control or externally influence the execution mode and flow of the SFC program.

10.1.6.3.6.3 Input Parameters

Input parameters are parameters generated by default when creating an SFC program block. Users cannot manually modify or delete them.

Para. Type	Parameter	Data Type	Initial Value	Note	Minimum Mode	Standard Mode
VAR_INPUT	OFF	BOOL	FALSE	Close the program and exit execution Rising edge triggered.	×	√
	INIT	BOOL	FALSE	Reset program and execute from the initial step Rising edge triggered.	√	√
	ACK_EF	BOOL	FALSE	Confirm monitor error Rising edge triggered.	×	√
	HALT	BOOL	FALSE	Stop/resume step switching Rising edge triggered.	×	√
	HALT_TM	BOOL	FALSE	Stop/resume timer Rising edge triggered.	×	√
	ZERO_OP	BOOL	FALSE	Stop/resume action output Rising edge triggered.	×	√
	MD_AUTO	BOOL	FALSE	Automatic mode Rising edge triggered.	×	√
	MD_TRANS	BOOL	FALSE	Mode with transition condition Rising edge triggered.	×	√
	MD_NTRANS	BOOL	FALSE	Mode ignoring transition condition Rising edge triggered.	×	√

Para. Type	Parameter	Data Type	Initial Value	Note	Minimum Mode	Standard Mode
	MD_MAN	BOOL	FALSE	Manual mode Rising edge triggered.	×	√
	SEL_SNO	INT	0	Select step number Effective immediately.	×	√
	S_ON	BOOL	FALSE	Activate Effective immediately.	×	√
	S_OFF	BOOL	FALSE	Deactivate Effective immediately.	×	√
	T_PUSH	BOOL	FALSE	Execute once Rising edge triggered.	×	√

10.1.6.3.6.4 Output Parameters

Output parameters are parameters generated by default when creating an SFC program block. Users cannot manually modify or delete them.

Para. Type	Parameter	Data Type	Initial Value	Note	Minimum Mode	Standard Mode
VAR_OUTPUT	SNO	INT	0	Step number of the currently executing step	×	√
	ACTIVE	BOOL	FALSE	Step indicated by SNO activated or not	×	√
	ERROR	BOOL	FALSE	Condition error	×	√
	FAULT	BOOL	FALSE	Monitor error	×	√
	ERR_FLT	BOOL	FALSE	Condition or monitor error	×	√
	HALTED	BOOL	FALSE	Stop step switching	×	√
	TM_HALTED	BOOL	FALSE	Stop timer	×	√
	OP_ZEROED	BOOL	FALSE	Stop action output	×	√
	AUTO_ON	BOOL	FALSE	Automatic mode	×	√
	TRANS_ON	BOOL	FALSE	Mode with transition condition	×	√
	NTRANS_ON	BOOL	FALSE	Mode ignoring transition condition	×	√
	MAN_ON	BOOL	FALSE	Manual mode	×	√
	IS_OFF	BOOL	FALSE	Program closed	×	√

10.1.6.3.6.5 Static Parameters

Static parameters of a step

The static parameters of a step are intrinsic read-only properties of each step in the SFC program. They are used to monitor the real-time status of the step, and are independent of the interface parameter mode.

Parameter	Data Type	Initial Value	Description	Function description
SNO	INT	1	Step number	Step number
S1	BOOL	FALSE	Step activation event	Rising edge: TRUE at the moment the step transitions from not inactive to active, then becomes FALSE (remains FALSE even when the step remains active).
S0	BOOL	FALSE	Step deactivation event	Falling edge: TRUE at the moment the step transitions from active to inactive, then becomes FALSE.
SX	BOOL	FALSE	Step activation status	TRUE when the step is active (displayed in green).
C1	BOOL	FALSE	Condition unsatisfied event	Condition (C) leaves: Condition result changes from TRUE to FALSE.
C0	BOOL	FALSE	Condition satisfied event	Condition (C) enters: Condition result changes from FALSE to TRUE.
CA	BOOL	FALSE	Condition unsatisfied status	TRUE when the condition is not satisfied; FALSE when satisfied.
CS	BOOL	FALSE	Condition result	The logical execution result of condition (C).
K1	BOOL	FALSE	Monitor error occurrence event	Transitions from FALSE to TRUE when a monitor (K) error occurs.
K0	BOOL	FALSE	Monitor error confirmed event	Monitoring (K) error is confirmed after occurrence.
KA	BOOL	FALSE	Monitoring active status	TRUE when monitor is active, which is the same as SX.
KS	BOOL	FALSE	Direct monitoring result	The logical execution result of monitor (K).
A1	BOOL	FALSE	Monitor error confirmed	After an error occurs, the step stays at current position (displayed in red). After the error is confirmed, the step continues.
TT	TIME	TIME#0ms	Total active time of step	• Accumulates while the step is active. • When the step becomes inactive, accumulation stops but does not reset to 0.

Parameter	Data Type	Initial Value	Description	Function description
UT	TIME	TIME#0ms	Active time without condition /monitor error	• Accumulates while the step is active and no condition/monitor error is present. If either a condition/monitor error occurs, accumulation stops. • For a condition error, accumulation resumes when the error clears. • For a monitor error, UT resets to 0 when the error is confirmed. • When the step becomes inactive, UT does not reset to 0.
TM	TIME	TIME#10s0ms	Maximum step active time	User configurable. Default value can be set.
TW	TIME	TIME#7s0ms	Step alarm active time	User configurable. Default value can be set.
CD_STAT	DWORD	0	Status of 32 LD elements in the condition	Stores the status of the 32 elements in condition (C).
CD_STAT_ERR	DWORD	0	Copies CD_STAT if the condition leaves	Stores the previous status of the 32 elements in condition (C).

Static parameters of a transition

The static parameters of a transition are intrinsic read-only properties of each step in the SFC program. They are used to determine whether the transition condition is met, and are independent of the interface parameter mode.

Parameter	Data Type	Initial Value	Description	Function description
TNO	INT	1	Transition number	Transition number
TA	BOOL	FALSE	Transition active status	Indicates whether the transition is active.
TT	BOOL	FALSE	Transition result	The logic result of the transition.
TS	BOOL	FALSE	Transition occurrence	Indicates the transition has occurred.
TRANS	DWORD	0	Status of 32 LD elements in the current transition	Stores the status of the 32 elements in the transition.
TRANS_OLD	DWORD	0	Status of 32 LD elements in the previous transition	Stores the status of the 32 elements in the previous transition (copies the value of TRANS).

Parameter	Data Type	Initial Value	Description	Function description
TRANS_ERR	DWORD	0	Copies TRANS if an error occurs	Stores the value of TRANS when an error occurs.

10.1.6.4 Step and Transition Condition

10.1.6.4.1 Step and Transition

10.1.6.4.1.1 Step Overview

In SFC, the control process is divided into clear, sequential stages called steps. Steps are connected by transitions. When the transition condition between two adjacent steps is met, the transition occurs: the previous step is deactivated and the next step is activated, ensuring no overlap between steps. Each step represents the start, continuation, or end of an action. The more steps a process cycle is broken into, the more precisely the entire process is described.

10.1.6.4.1.2 Transition Overview

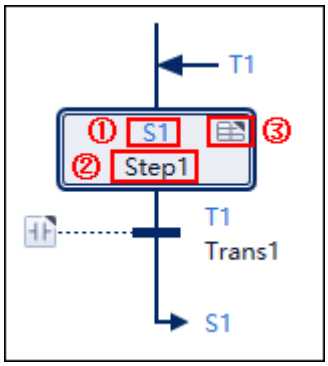
In SFC, transition is the core logical condition that controls the flow of process from one step to the next step. Each transition corresponds to a unique condition. The transition can only be realized when the condition is met, allowing the process to proceed to the next step.

10.1.6.4.2 Step

10.1.6.4.2.1 Step Basic Properties

Each step includes step number, name, action, maximum step activation time, alarm time, and comment.

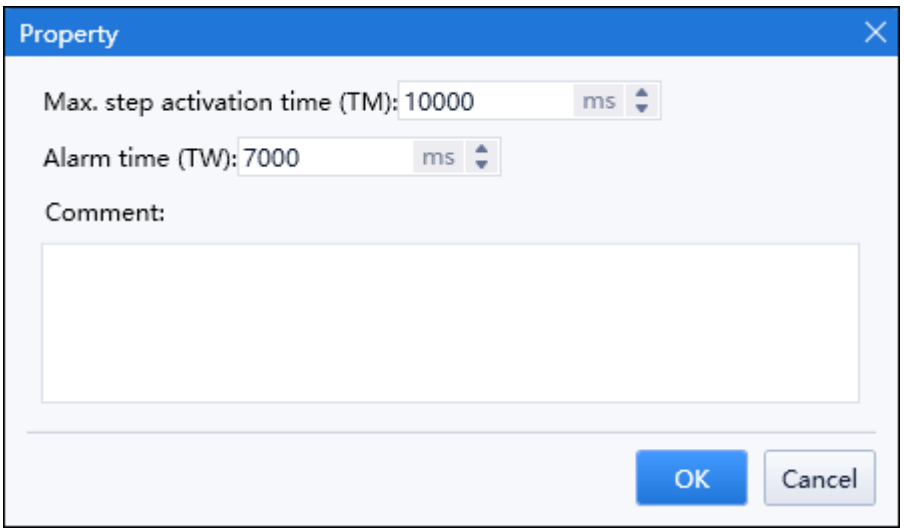
- Basic properties



No.	Item	Description
①	Step number	Steps are numbered numerically. Each step number is unique within the same program block. The step number is automatically generated when a step is created. Double-click to edit. Valid range: 1 to 32767.

No.	Item	Description
②	Step name	The name of the step, which must be unique within the same program block. The step name is automatically generated when a step is created. Double-click to edit. Allowed characters: letters, Chinese characters, numbers, and underscores. Max length: 64 characters.
③	Action	A step can contain zero or multiple actions. An action includes condition, event, qualifier, and action body. • Condition: If a condition is set, the action is executed only if the step is activated and the condition is met. By default, no condition logic is created; users can program the logic. • Event: If an event is set, the action is executed only when the event occurs and the step is activated. By default, no event is created; users can program the event. • Qualifier: Defines how the action is executed. For example: S is set (sets the operand to 1), and R is reset (sets the operand to 0). • Action: Combined with a qualifier to determine the operand or program call to be executed. It can be a variable, an expression, or a program block call.

- More properties
  1. Right-click the target step and select "Property".
  2. Set the maximum step activation time, alarm time, and comment in the "Property" window, then click "OK".



Item	Description
Max. step activation time (TM)	The maximum time a step is allowed to remain active. It ranges from 0 to 3600000 ms with a default of 10000 ms, and is editable by users.
Alarm time (TW)	When a step is activated, if its execution time exceeds the set alarm time but does not reach the maximum step activation time, the system issues an alarm but does not interrupt program execution. It ranges from 0 to 3600000 ms with a default of 7000 ms, and is editable by users.

Item	Description
Comment	Each step can have a comment, which is an empty string by default and can be modified by users. It can contain letters, Chinese characters, numbers, and underscores, with a length limit of 128 characters.

10.1.6.4.2.2 Step Status

Steps have active and inactive states.

- Active state
  When a step is active, the program logic within the step (e.g., conditions, monitors, actions) is executed, and the internal state of the step is updated.
  - Step condition error state
    When the step is active but its condition is not met, the step is in a condition error state.
  - Step monitor error state
    When the step is active and a monitor condition is met, the step is in a monitor error state.
- Inactive state
  When a step is inactive, no logic inside the step is executed.

10.1.6.4.2.3 Condition

Conditions are optional in steps. If a condition is set, the action is executed only if the step is activated and the condition is met.

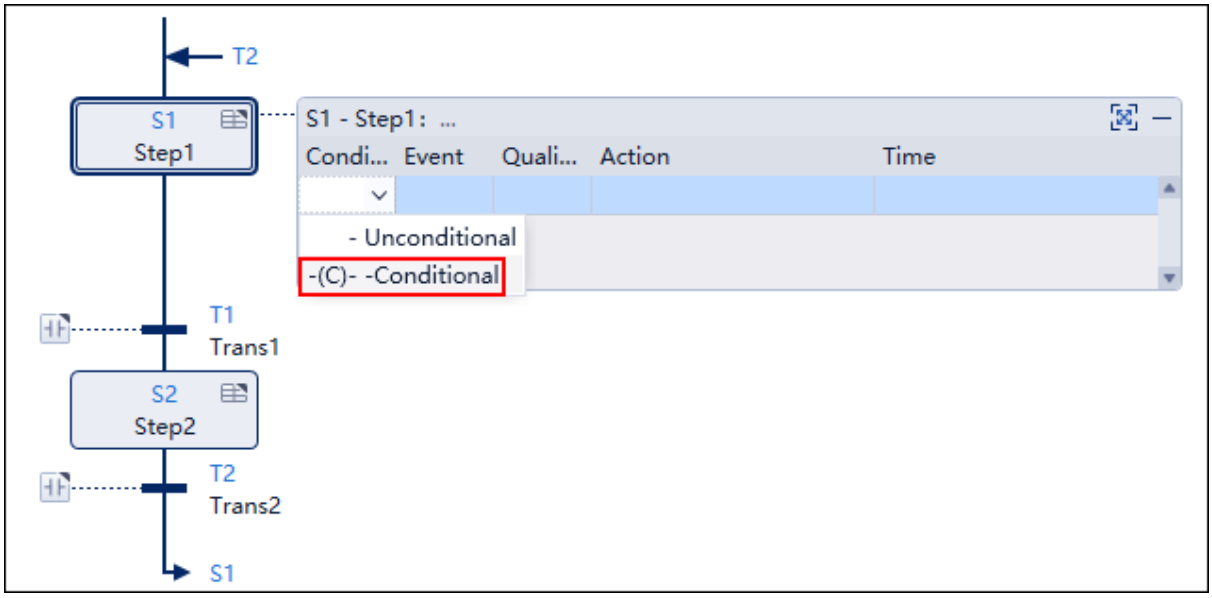
By default, no condition logic is created. Users can program the logic through LD language, but it only supports one LD network segment and only supports comparison operations.



In an SFC program, a maximum of 32 conditions can be added.

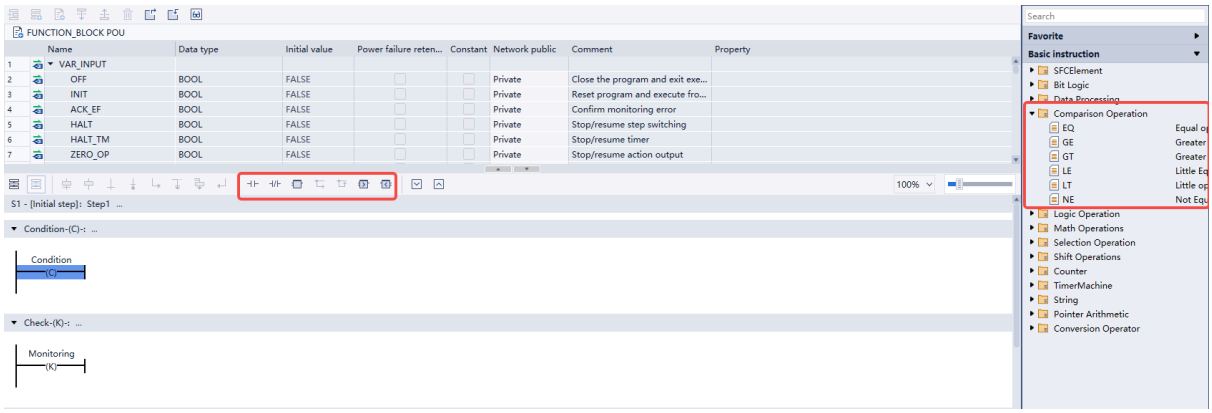
Procedure

1. In the SFC program, click  in the upper right corner.
2. In the "Condition" column of the action list, select "-(C)- -Condition", to add the condition.



3. Select the current step and click in the toolbar to enter the step view.

4. Write the LD program for a condition in the step view. You can use LD elements from the toolbar or the right-side toolbox, as well as comparison operation instructions from the toolbox.

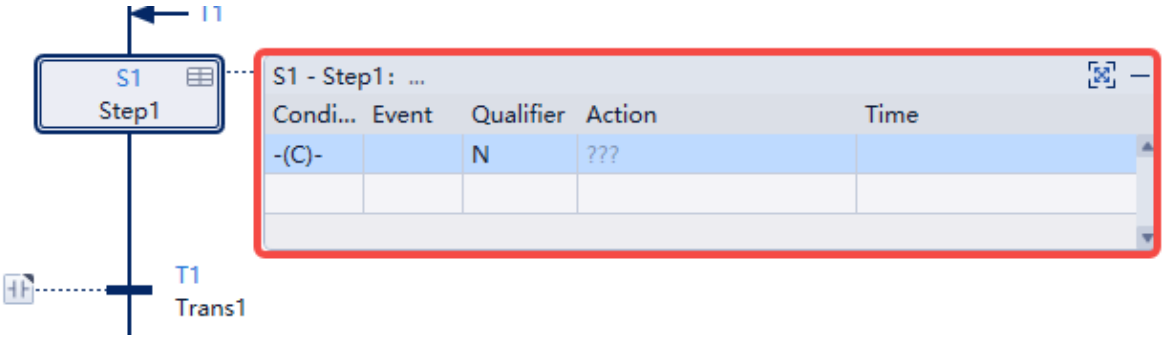


10.1.6.4.2.4 Action

Actions define the operations to be completed in a specific step. An action includes conditions, events, qualifiers, action bodies, and time. Multiple actions are executed in the same step based on their order in the action list, from top to bottom.

> **NOTE**
>
> In an SFC program, a maximum of 100 action blocks can be added.

Click at the upper right corner of the step to open the action list. If there are many actions in the current step, the list may exceed the display area. Click the "Fit size" icon to fully display all action content within the visible area; click the "Minimize" icon to collapse the action list and simplify the interface.



Property	Optional/ Required	Note
Condition	Optional	If a condition is set, the action is executed only if the step is activated and the condition is met. By default, no condition logic is created; users can program the logic.
Event	Optional	If an event is set, the action is executed only when the event occurs and the step is activated. For details, see Event description.
Qualifier	Required	Defines how the action is executed. For example: S is set (sets the operand to 1), and R is reset (sets the operand to 0). For details, see Qualifier description.
Action	Required	Combined with a qualifier to determine the operand or program call to be executed. It can be a variable, an expression, or a program block call.
Time	Optional	When using the qualifiers D and L, a variable of time type must be passed as an input parameter.

Table10-1-6-4-2-4-1 Qualifier description

Qualifier	Description	Supported Action Type	Remarks
N	Set whenever the step is active	BOOL variable, expression	When the step is active, the action is executed each cycle.
R	Set to 0	BOOL variable, expression	• Performs a reset operation for a BOOL variable. • Stops executing the action for an ACT call.
S	Set to 1	BOOL variable, expression	• Performs a set operation for a BOOL variable. • Starts executing the action for an ACT call. The action continues even after the step becomes inactive.

Qualifier	Description	Supported Action Type	Remarks
L	Set within the specified time	BOOL variable, expression	When the step becomes active, the action is executed repeatedly for a user-specified duration, until the step deactivates or the time expires. For example, when the time parameter is 3s: • If the step is active for 2s, the action is executed for 2s. • If the step is active for 5s, the action stops after 3s.
D	Switch-on delay	BOOL variable, expression	After the step is activated, the action starts executing only if the step remains active for the user-specified delay. It continues until the step deactivates. For example, when the time parameter is 3s: • If the step becomes inactive within 3s, the action is not executed. • If the step is still active after 3s, the action is executed. • If the step becomes inactive at 5s, the action stops.

Table10-1-6-4-2-4-2 Event description

Event	Description	Triggered Signal	Note
None	No event	-	/
A1	Alarm confirmation	Commissioning panel button clicked or a rising edge triggered on ACK_EF	The monitoring condition has been triggered, i.e., Step.KS = FALSE and Step.KA = TRUE.
C0	Conditions for arrival	Rising edge	The condition is met, i.e., the error is cleared (signal state is 1).
C1	Conditions for departure	Falling edge	The condition is met, i.e., an error occurs (signal state is 0).
S0	Step deactivated	Falling edge	Step has become inactive (signal state is 0).
S1	Step activated	Rising edge	Step has become active (signal state is 1).
K0	Monitor error cleared	Falling edge	The monitoring condition is no longer met, i.e., the error is cleared (signal state is 0).
K1	Monitor error triggered	Rising edge	The monitoring condition is met, i.e., an error occurs (signal state is 1).

10.1.6.4.2.5 Monitoring

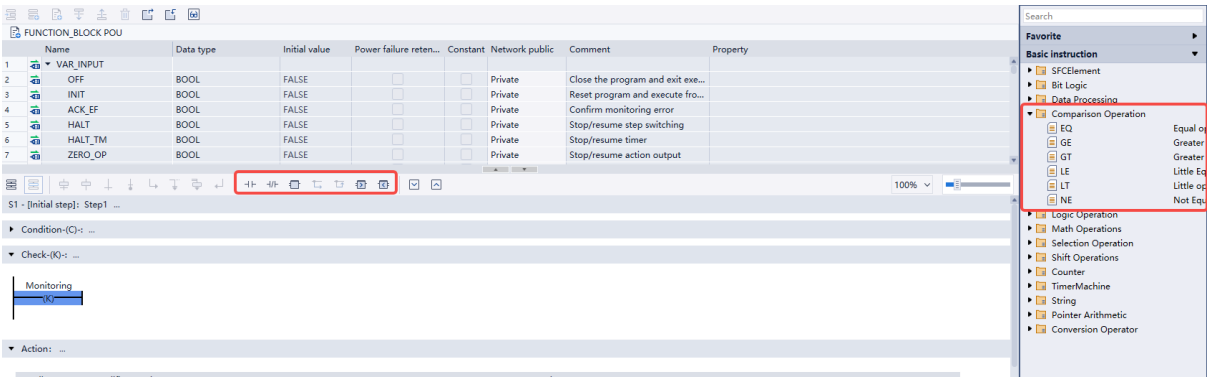
Monitoring is optional element in a step. If the user writes monitor logic, when the step is active and the monitor condition is met, the step reports an error and remains in the current step (even if the following transition condition is met, the step will not switch automatically).

By default, no monitor is created when a step is created. Users can program the logic through LD language, but it only supports one LD network segment and only supports comparison operations.

> NOTE
>
> In an LD program, up to 32 monitors can be added.

Procedure

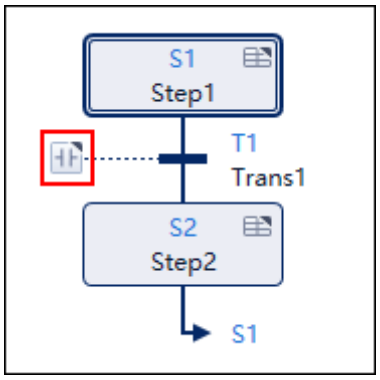
1. Select the target step and click in the toolbar to enter the step view.
2. Write the LD program for monitor in the step view. You can use LD elements from the toolbar or the right-side toolbox, as well as comparison operation instructions from the toolbox.

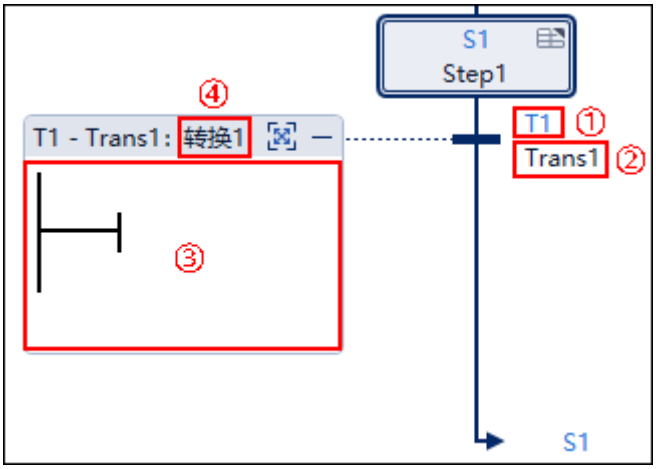


10.1.6.4.3 Transition

10.1.6.4.3.1 Transition Basic Properties

Click on the side of the transition to expand the transition condition logic editing area. If the current program content is large, it may exceed the display area. Click the "Fit size" icon to fully display all program content within the visible area; click the "Minimize" icon to collapse the condition logic editing area and simplify the interface.





No.	Item	Description
①	Transition number	Transitions are numbered numerically. Each transition number is unique within the same program block. A transition number is automatically generated when a transition is created, which can be modified by users. Value range: 1 to 999.
②	Transition name	The name of the transition, which must be unique within the same program block. A transition name is automatically generated when a transition is created, which can be modified by users. It can include letters, Chinese characters, numbers, and underscores, with a length limit of 64 characters.
③	Transition condition	A transition is essentially a logic result (TRUE or FALSE). Transition logic can be programmed using LD language, but only one LD network segment is allowed, and only comparison operations are supported. Newly created transitions have empty logic by default, but the result is TRUE.
④	Comment	Each transition can have a comment, which is an empty string by default and can be modified by users. It can contain letters, Chinese characters, numbers, and underscores, with a length limit of 128 characters.

10.1.6.4.3.2 Transition Status

A transition condition can be active or inactive.

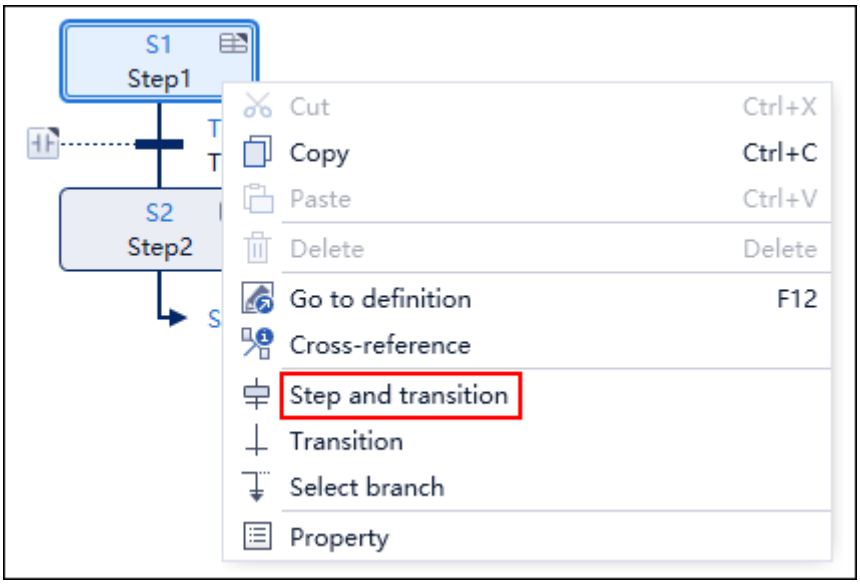
- Active state
  When the step before the transition is active, the transition condition becomes active.
  - Transition condition satisfied
    The transition condition is active, and the transition result is TRUE.
  - Transition condition unsatisfied
    The transition condition is active, and the transition result is FALSE.
- Inactive state
  When the step before the transition is inactive, the transition condition is inactive.

10.1.6.5 Step and Transition Usage

10.1.6.5.1 Adding a Step and Transition

There are multiple ways to add a step and transition, including the right-click menu, SFC editor toolbar, and the right-side toolbox. This section only introduces how to add through the SFC editor toolbar and the right-click menu.

- Method 1: Select a step or transition in the program, click on the toolbar to add a step and transition after the selected step.
- Method 2: Right-click a step or transition in the program, select "Step and transition" to add a step and transition after the selected step.

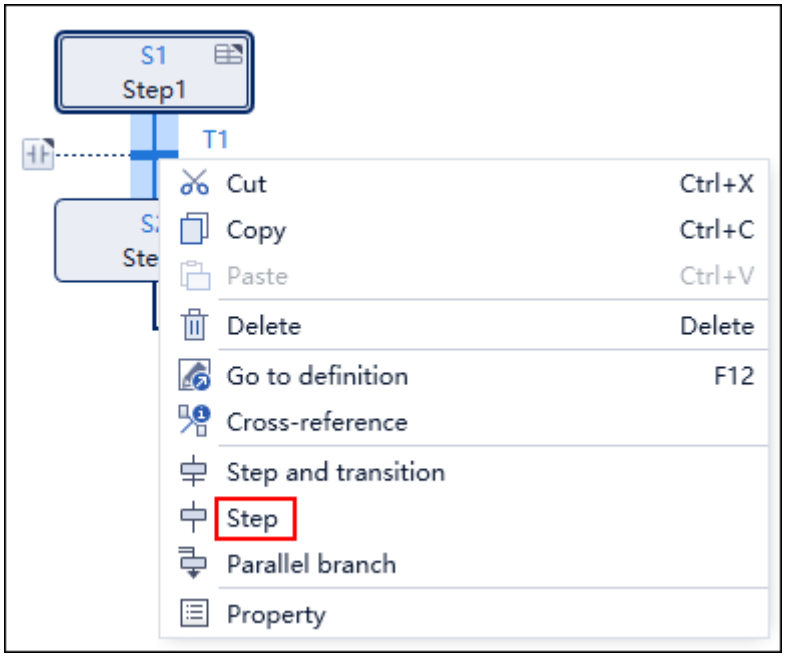


10.1.6.5.2 Adding a Step

There are multiple ways to add a step, including the right-click menu, SFC editor toolbar, and the right-side toolbox.

This section only introduces how to add through the SFC editor toolbar and the right-click menu.

- Method 1: Select a step in the program, click on the toolbar to add a step after the selected step.
- Method 2: Right-click a step in the program, select "Step" to add a step after the selected step.



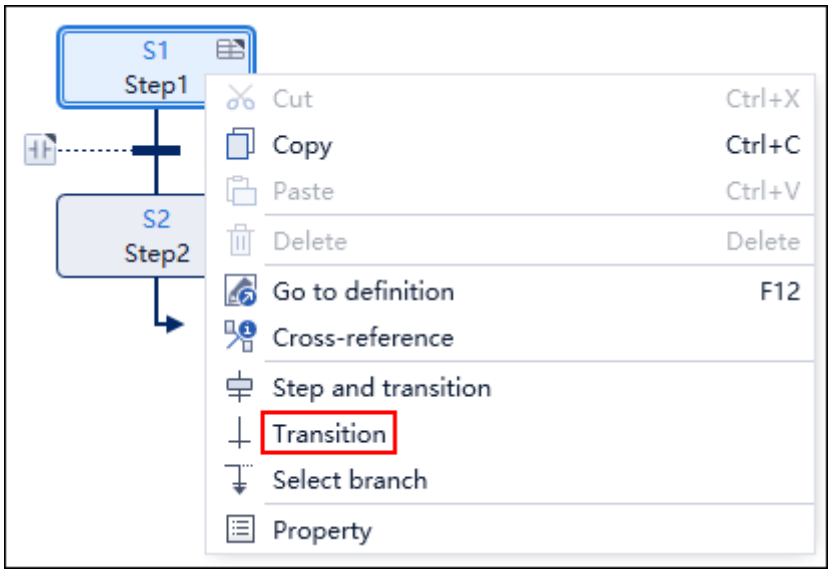
NOTE

In an SFC program, a maximum of 250 steps can be added.

10.1.6.5.3 Adding a Transition

There are multiple ways to add a transition, including the right-click menu, SFC editor toolbar, and the right-side toolbox. This section only introduces how to add through the SFC editor toolbar and the right-click menu.

- Method 1: Select a transition in the program, click ⊥ on the toolbar to add a transition after the selected transition.
- Method 2: Right-click a transition in the program, select "Transition" to add a transition after the selected transition.



> **NOTE**
>
> In an SFC program, a maximum of 250 transitions can be added.

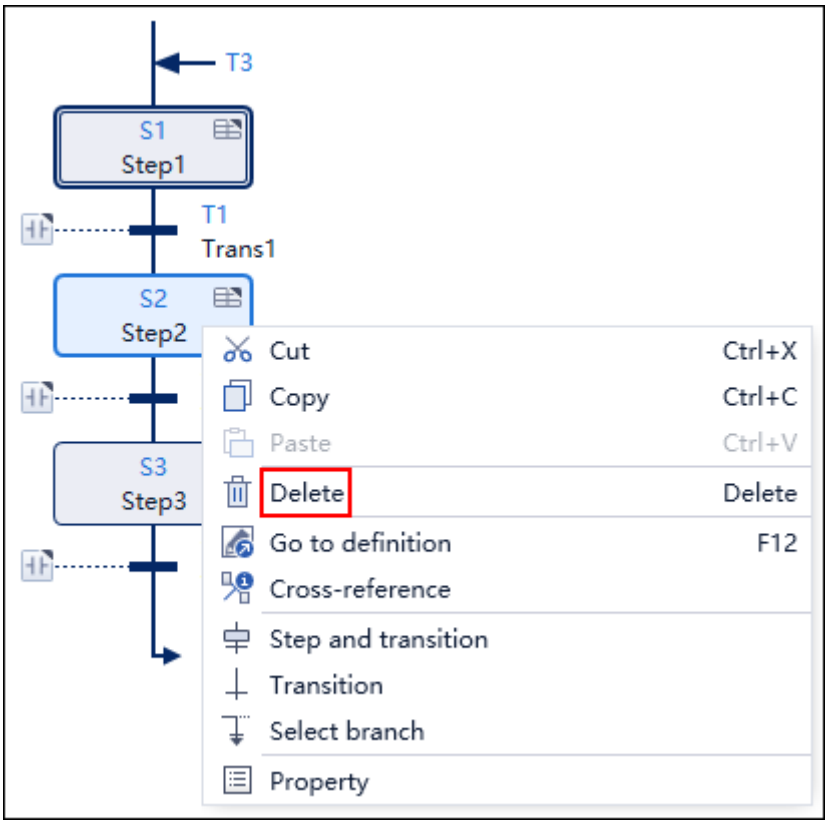
10.1.6.5.4 Deleting a Step and Transition

Restrictions

- The initial step cannot be deleted.
- When creating an SFC program, the first step generated by the system by default cannot be deleted.

Procedure

Right-click a step or transition in the program and select "Delete". Use the Shift key or Ctrl key for multiple selection to enable batch deletion.



10.1.6.6 Branch Usage

10.1.6.6.1 Adding a Selective Branch

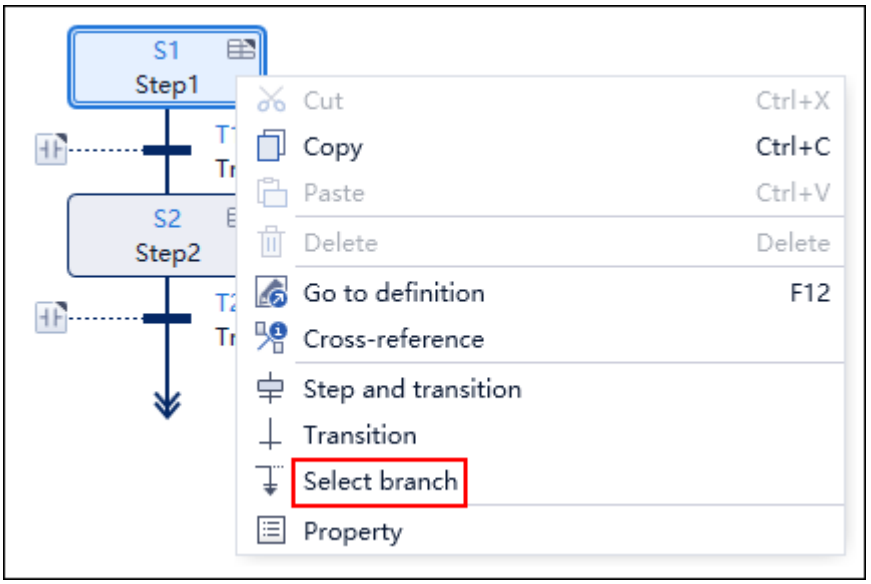
The selective branch can be added only below the step. Each selective branch must start with a transition. The logical relationship between multiple selective branches is OR, meaning the program executes only one of them. If multiple conditions are met simultaneously, execute the leftmost branch.

NOTE

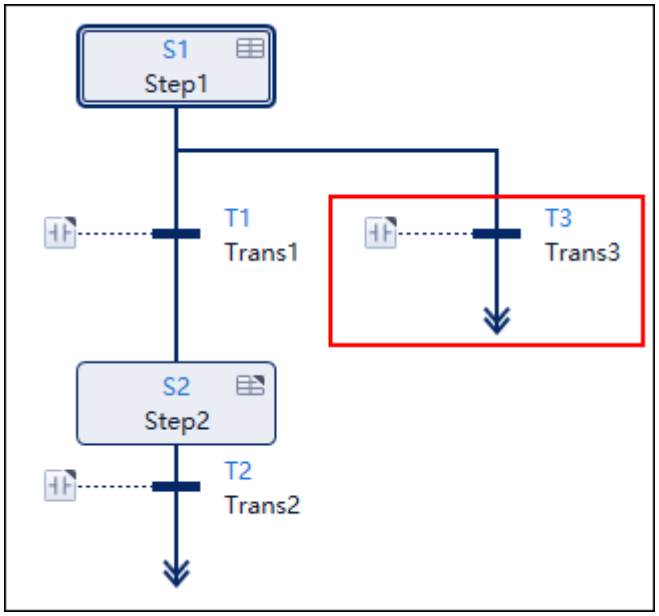
- In an SFC program, a maximum of 125 selective branches can be added.
- There are multiple ways to add an SFC element, including the right-click menu, SFC editor toolbar, and the right-side toolbox. This section only introduces how to add through the SFC editor toolbar and the right-click menu.

Procedure

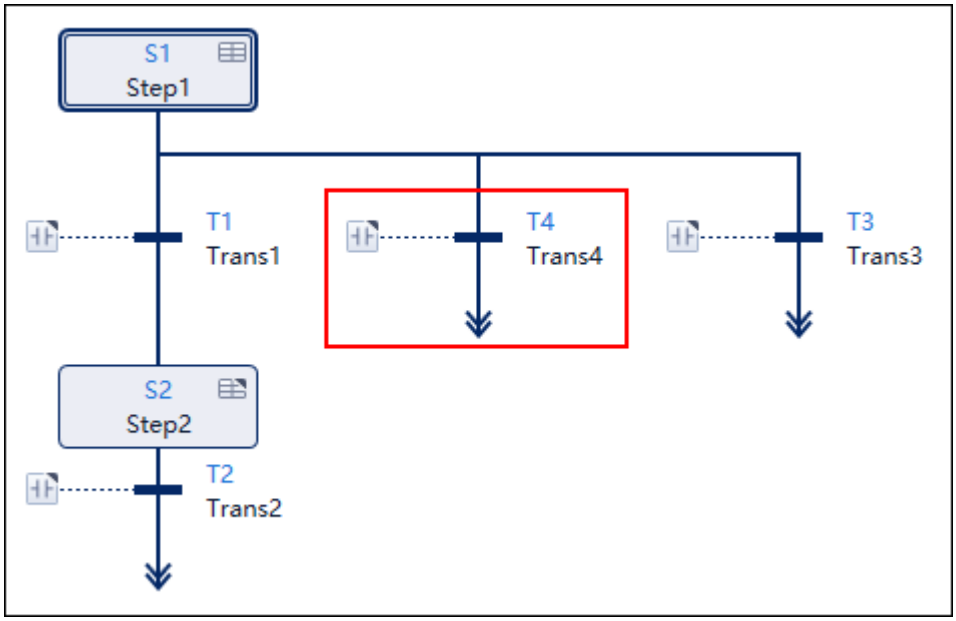
1. Select a step in the program and click on the toolbar. You can also right-click a step and select "Selective branch".



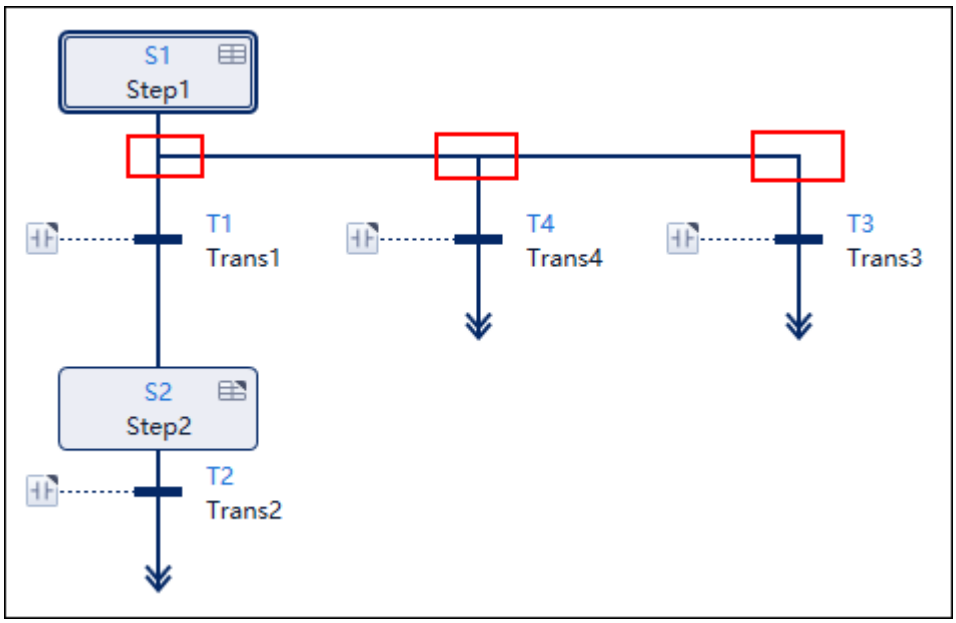
2. A new selective branch is added on the right, which includes a transition, as shown in the following figure.



3. Continue adding selective branches in the following two ways:
   - Keep the current step selected, click on the toolbar or select "Selective branch" from the right-click menu to continue adding new selective branches on the far left, like Trans4 below.

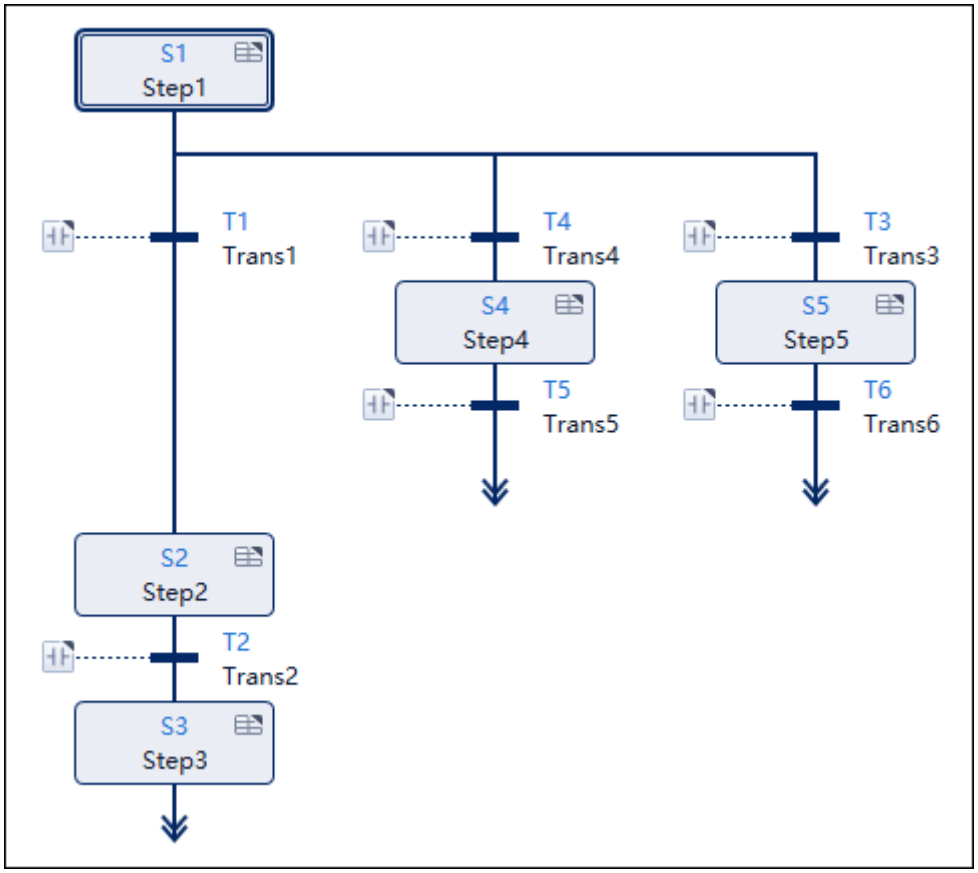


- Select the branch location as shown in the figure, click on the toolbar or select "Selective branch" from the right-click menu to add a new selective branch to the right of the selected branch.



Example Explanation

As shown in the figure below, if both Trans1 and Trans4 conditions are met, Step1 is executed, followed by Step2.



10.1.6.6.2 Adding a Parallel Branch

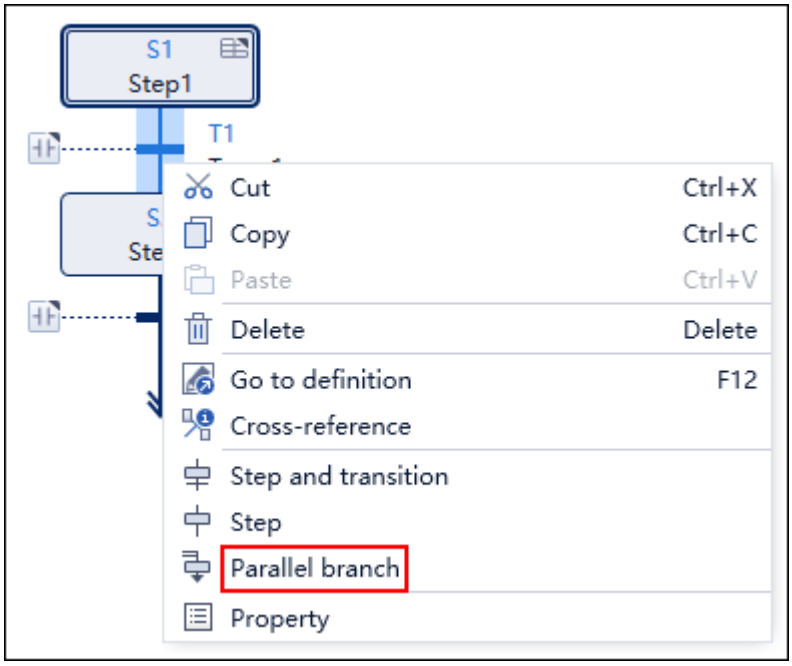
The parallel branch can be added only below the step. Each branch must start with a step. The logical relationship between multiple parallel branches is AND, meaning the program executes all branches simultaneously.

NOTE

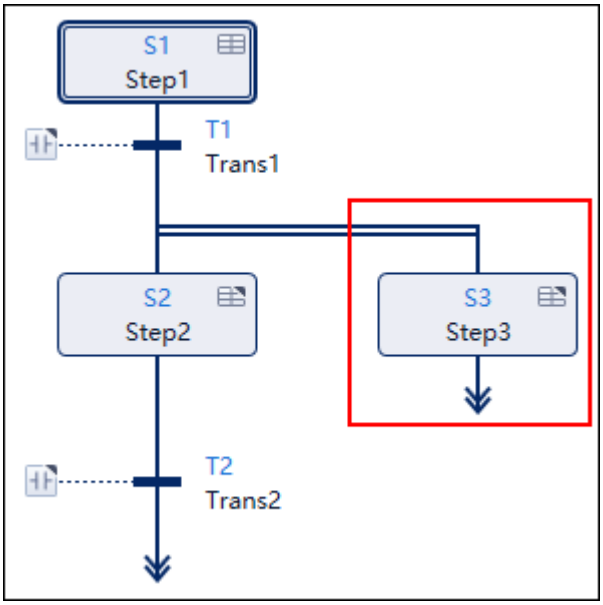
- In an SFC program, a maximum of 249 parallel branches can be added.
- There are multiple ways to add a parallel branch, including the right-click menu, SFC editor toolbar, and the right-side toolbox. This section only introduces how to add through the SFC editor toolbar and the right-click menu.

Procedure

1. Select a transition in the program and click on the toolbar. You can also right-click a transition and select "Parallel branch".

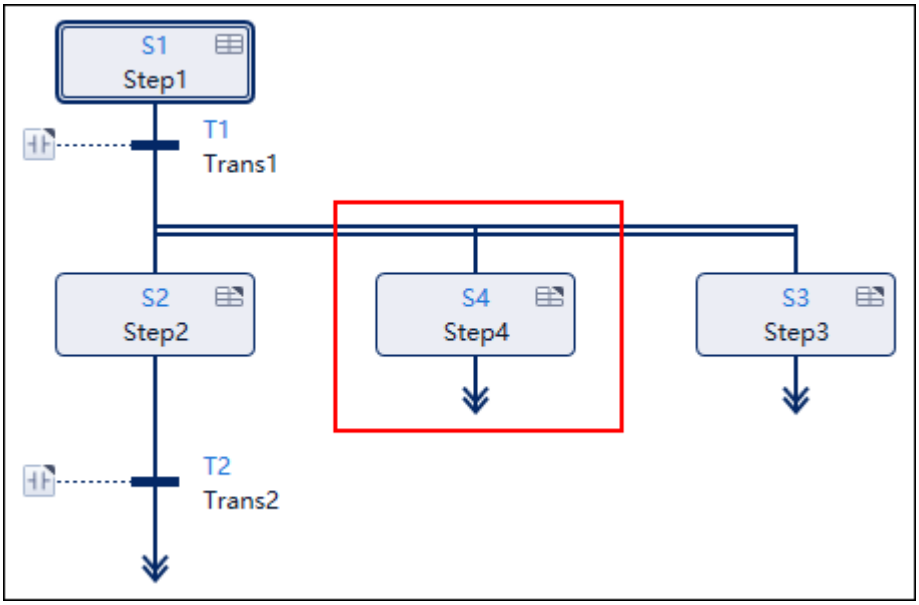


2. A new parallel branch is added on the right, which includes a step, as shown in the following figure.

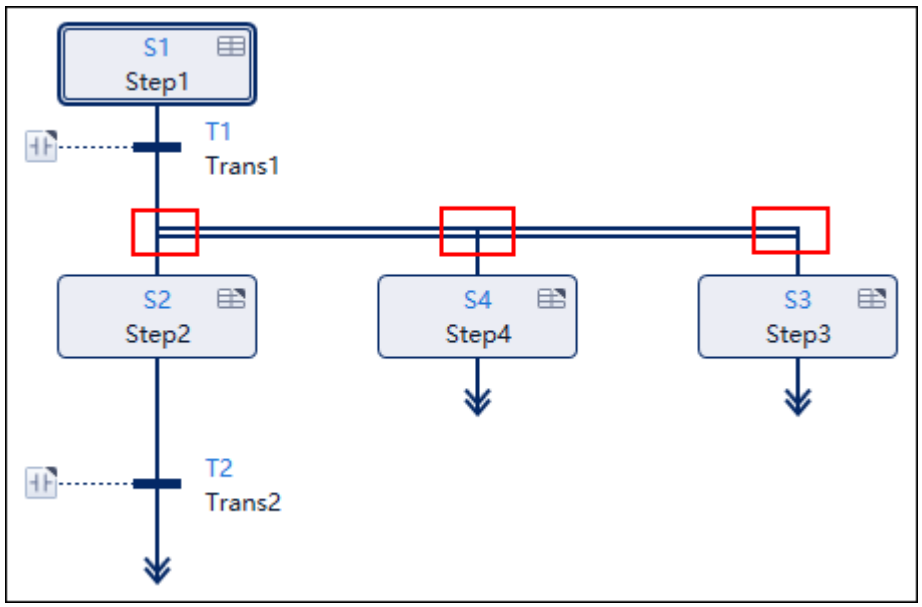


3. Continue adding parallel branches in the following two ways:

   ◦ Keep the current transition selected, click on the toolbar or select "Parallel branch" from the right-click menu to continue adding new parallel branches on the far left, like S4 below.



- Select the branch location as shown in the figure, click on the toolbar or select "Parallel branch" from the right-click menu to add a new parallel branch to the right of the selected branch.

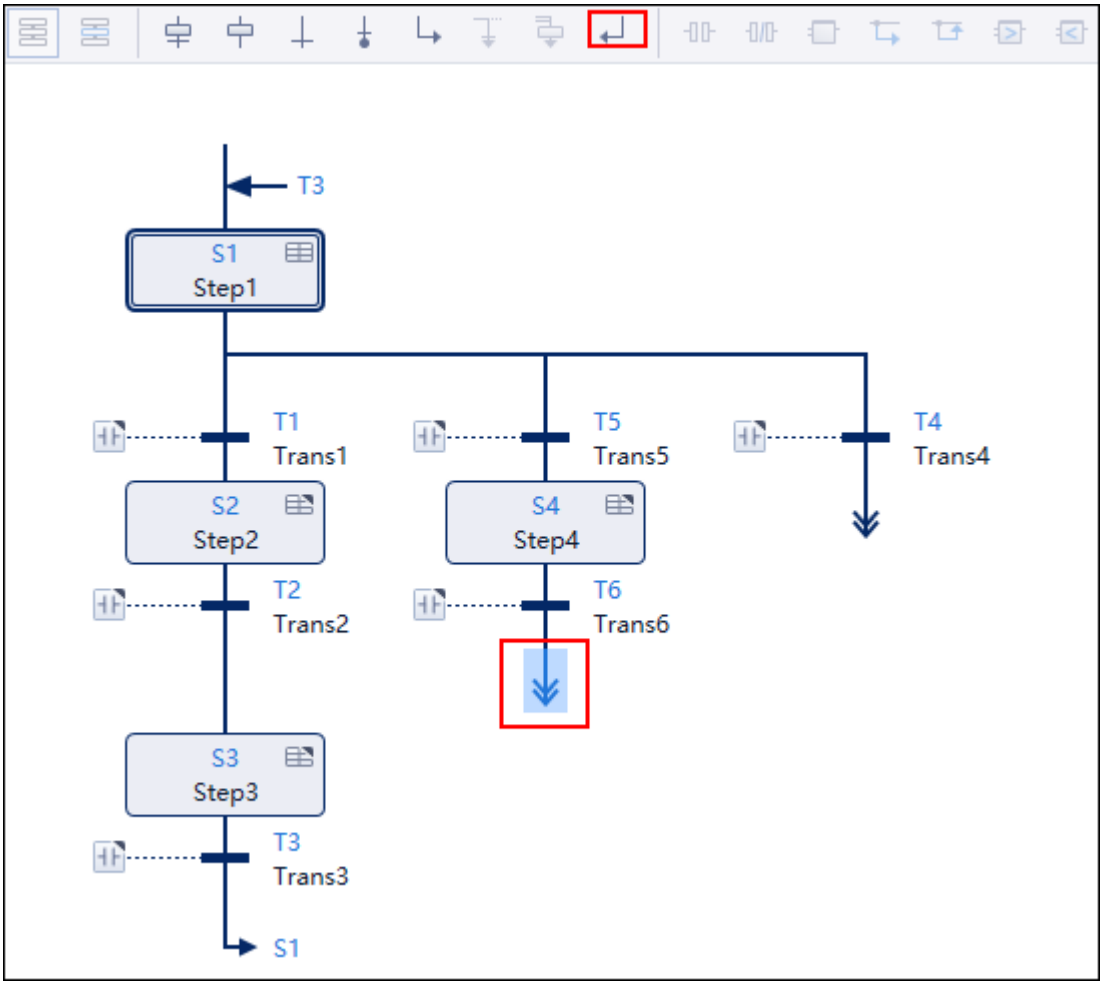


10.1.6.6.3 Closing a Branch

Closing a branch defines its end point and terminates the branch task. There are three methods: return to parent, insert a jump, or insert an end.

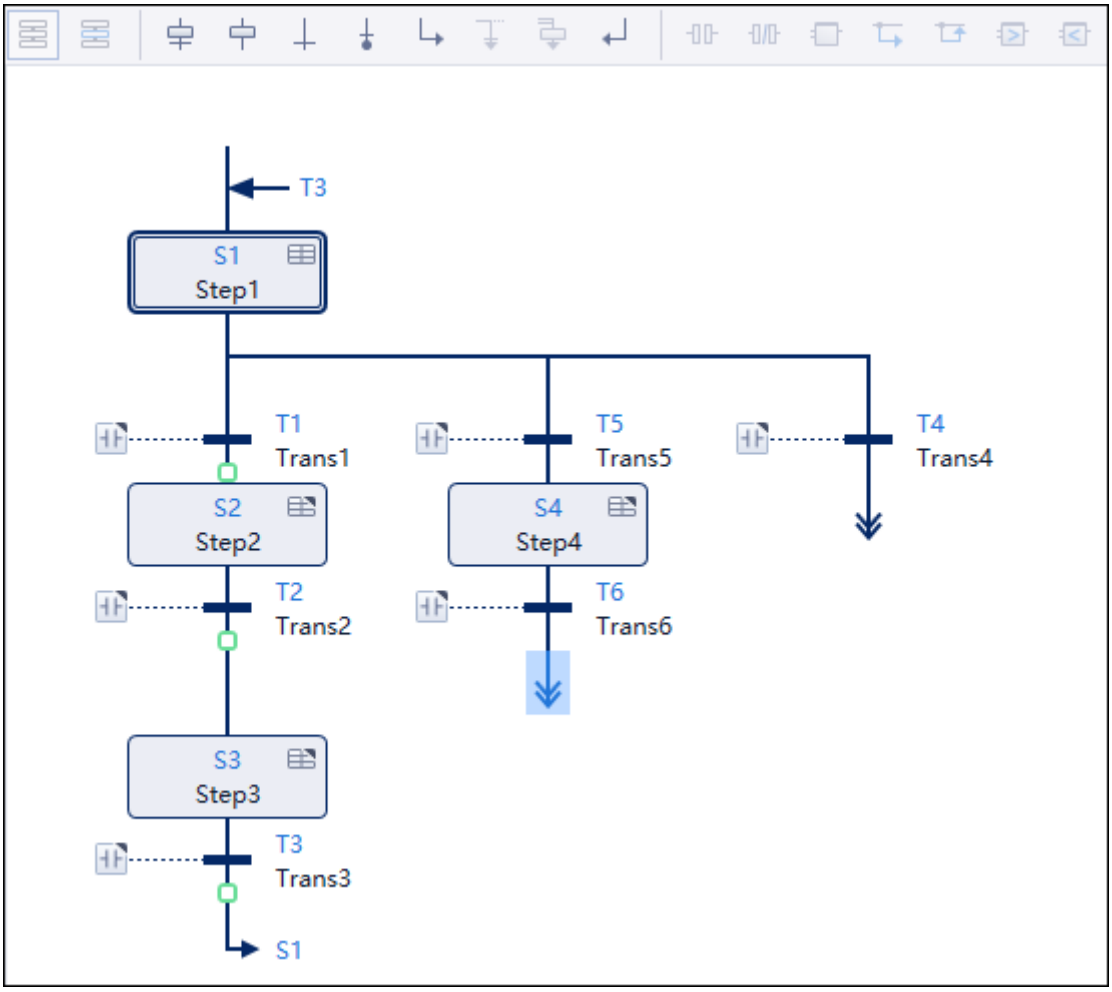
> NOTE
>
> There are multiple ways to add an SFC element, including the right-click menu, SFC editor toolbar, and the right-side toolbox. This section uses only one method as an example. The logic for other methods is similar, and users can choose flexibly.

- **Returning to parent branch**
  - Method 1: Select in a branch, and click on the toolbar, as shown in the following figure.

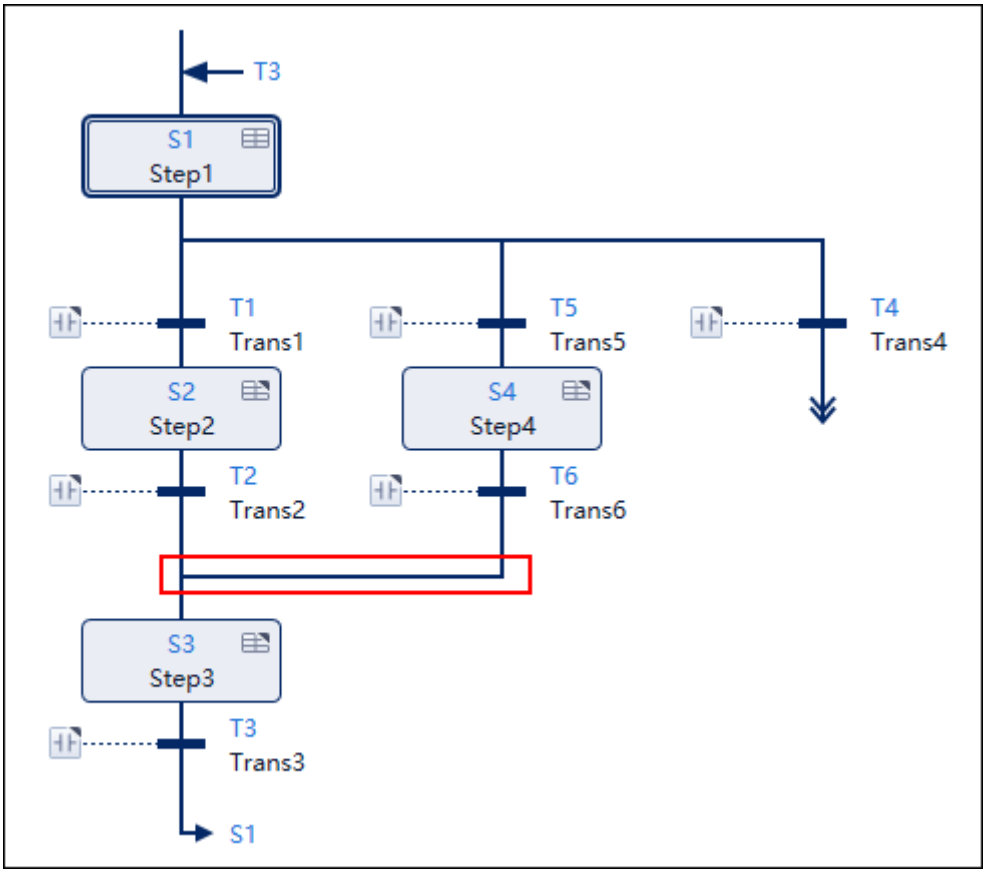


◦ Method 2: Drag and drop

1. Press and hold in a branch. Connectable positions will be highlighted in green, as shown in the figure below.

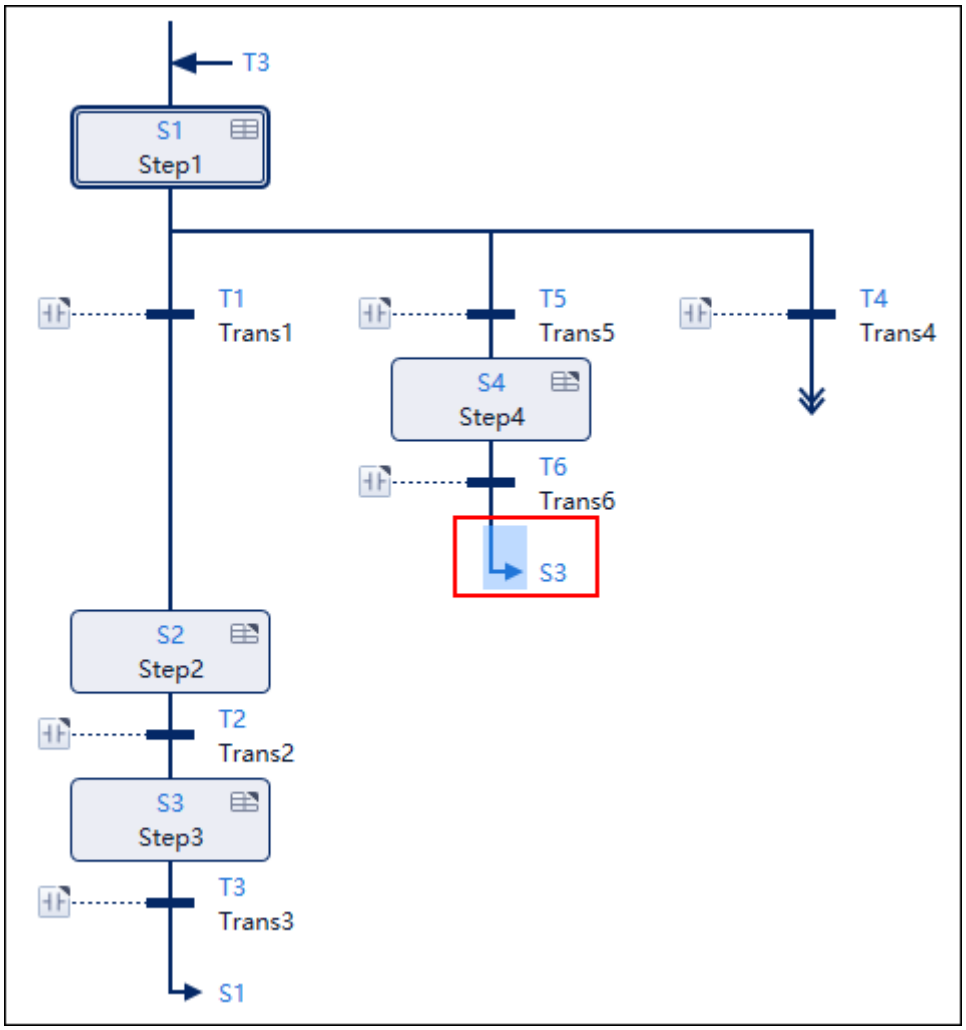


2. Move the mouse to the target position to connect.



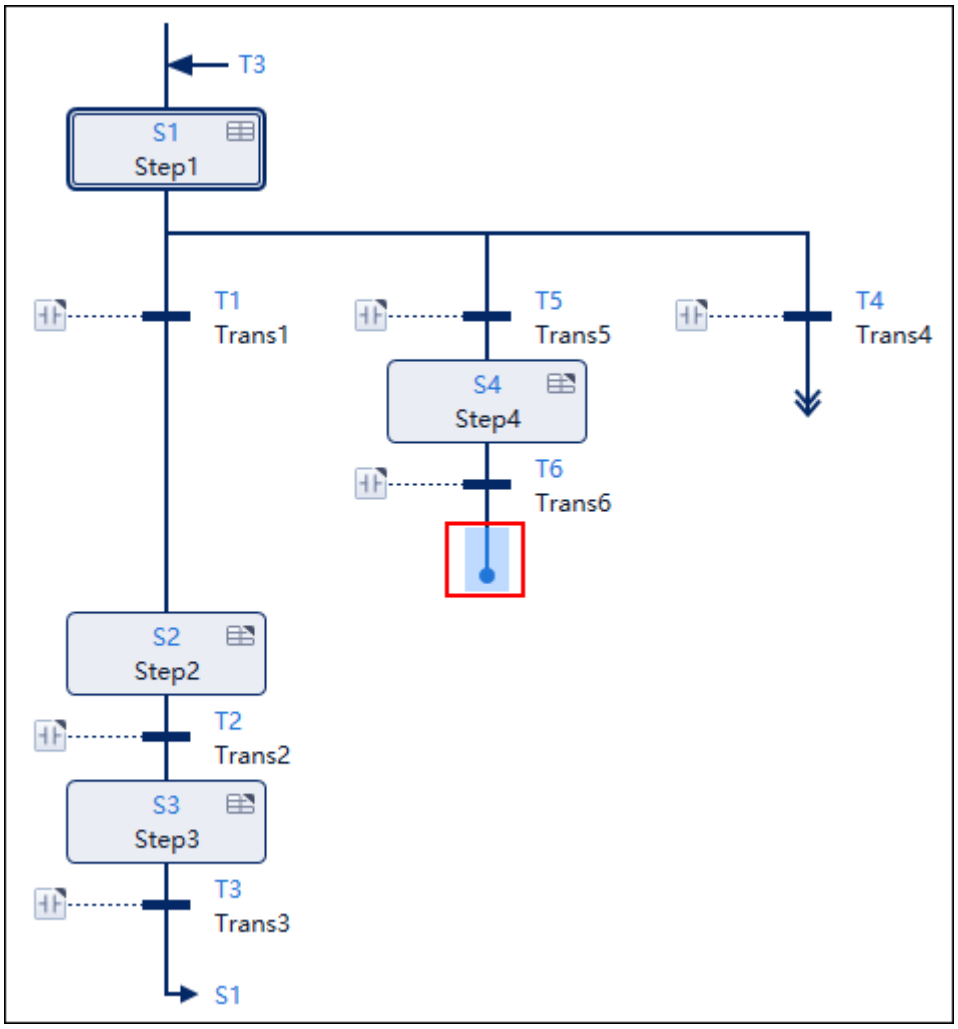
- **Inserting a jump**

Insert a jump to transfer execution to another step.



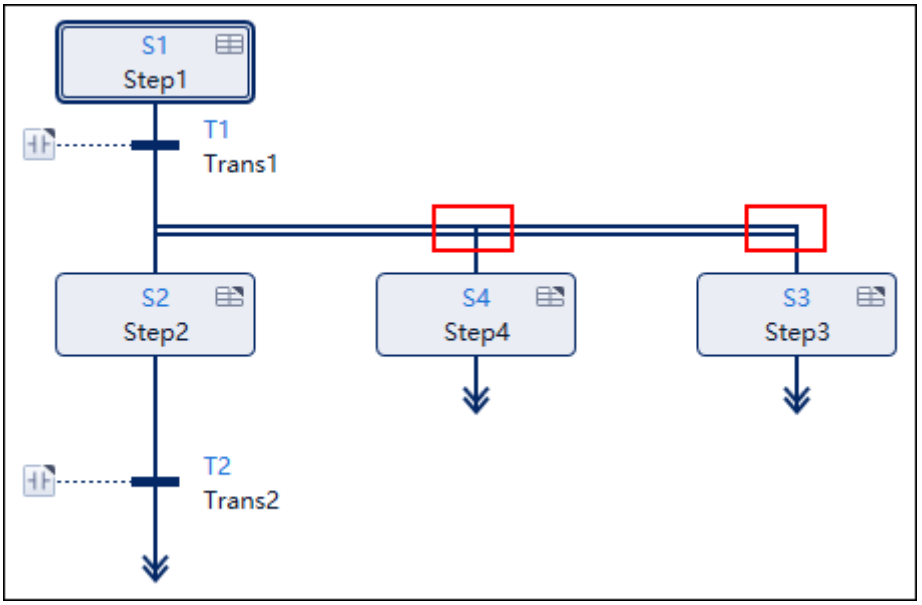
- **Insert an end**

End the program by inserting an end.

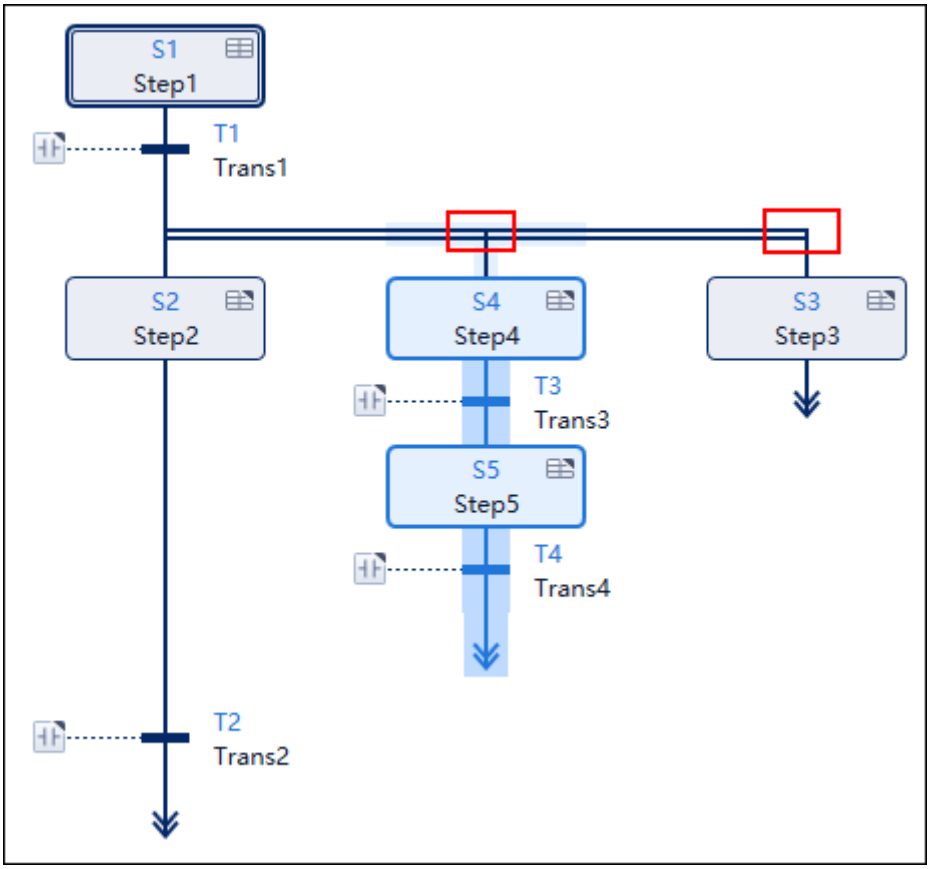


10.1.6.6.4 Deleting a Branch

- Method 1: Right-click the top of the branch and select "Delete".



- Method 2: Double-click the top of the branch to select all branch objects. Right-click the branch and select "Delete".



10.1.6.7 Jump Usage

10.1.6.7.1 Adding a Jump

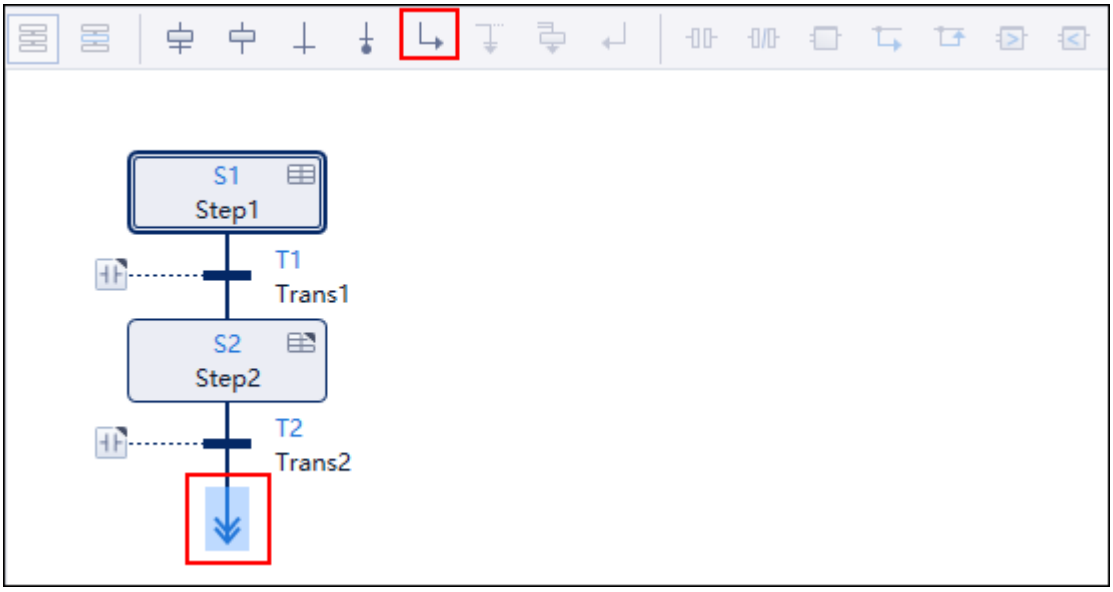
By using the jump function, you can jump from any step directly to a specified step for program loop processing.

> NOTE
>
> There are multiple ways to add an SFC element, including the right-click menu, SFC editor toolbar, and the right-side toolbox. This section uses only one method as an example. The logic for other methods is similar, and users can choose flexibly.

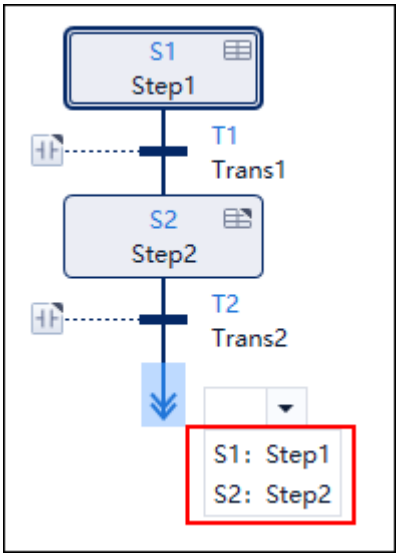
Procedure

1. Select one of the in the program, and click on the toolbar.

> NOTE
>
> A transition must exist before a jump; if missing, please insert a transition first.



Select the target step. The target step is a step within this program block (POU).



10.1.6.7.2 Changing Jump Target

The jump function allows direct navigation to a specified step from any step, and supports changing the target as needed.

Prerequisites

A jump has been added.

Procedure

1. Double-click the current target step number.
2. Select a step as the new jump target.

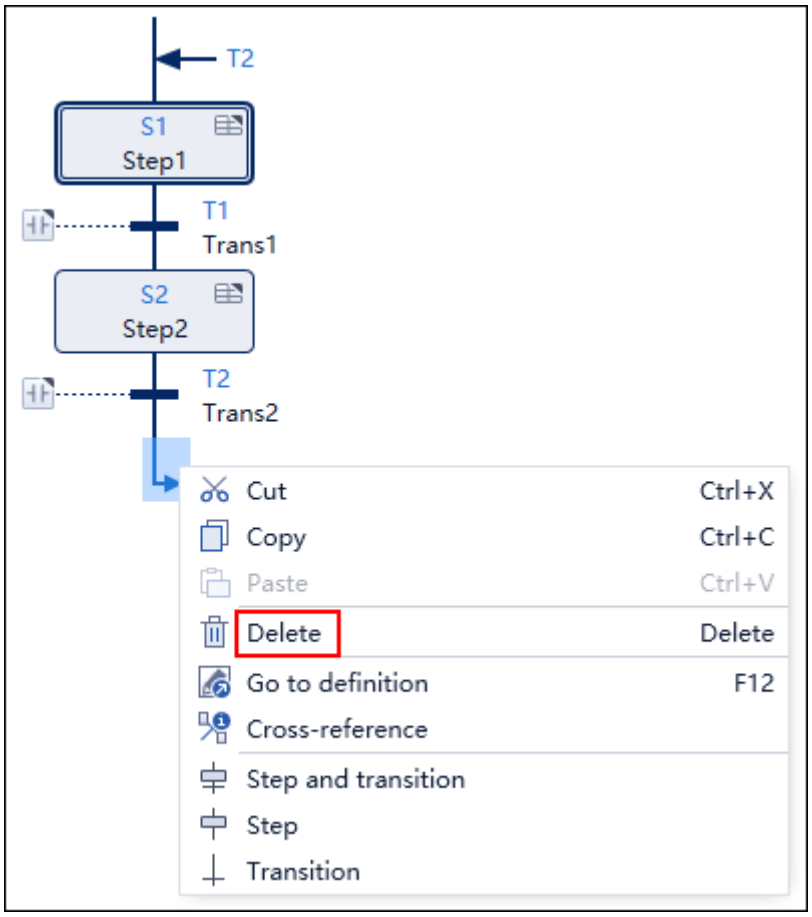
10.1.6.7.3 Deleting a Jump

Prerequisites

A jump has been added.

Procedure

Right-click the added step and select "Delete".



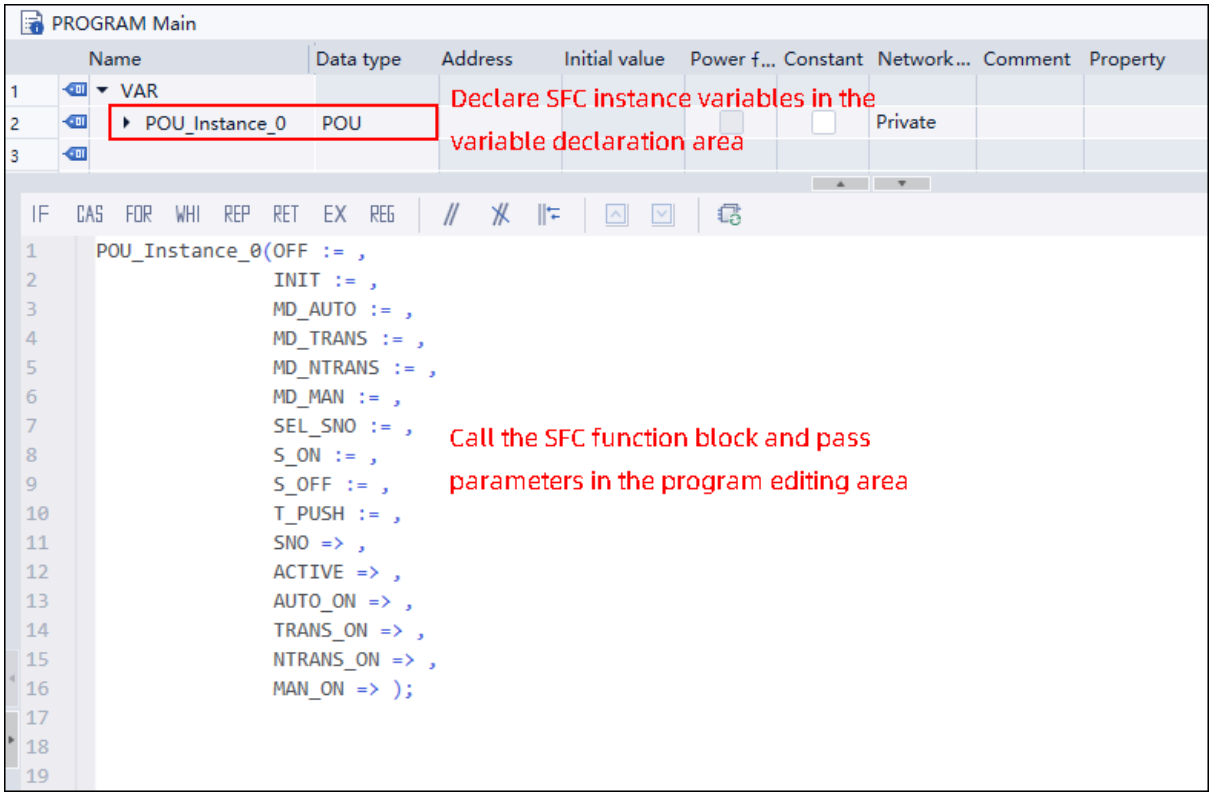
10.1.6.8 SFC Configuration and Commissioning

10.1.6.8.1 SFC Instantiation

SFC instantiation refers to the need for function blocks to be instantiated before use. Function blocks are abstract structure type definitions and not actual data entities; therefore, they cannot be called or executed by a program without instantiation.

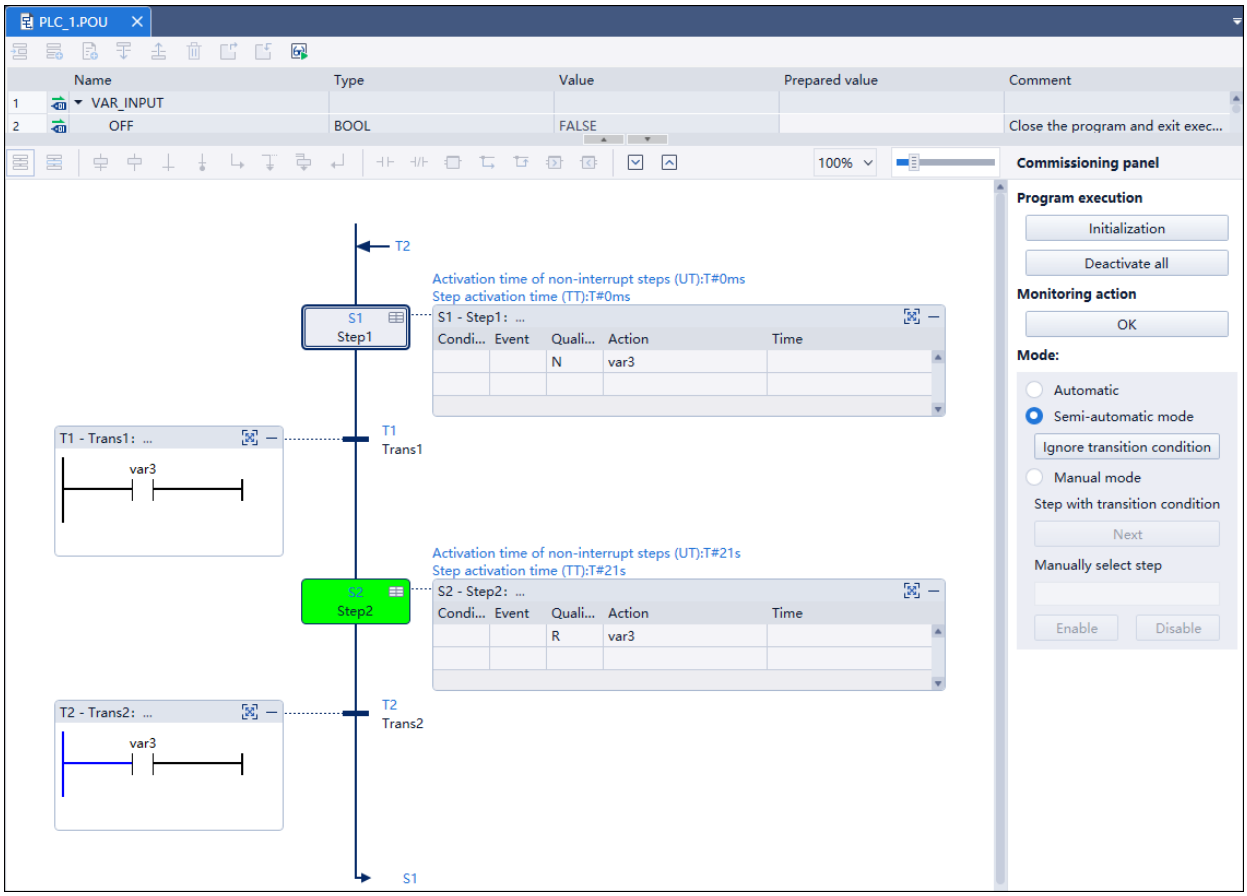
Procedure

1. Create an SFC function block. For details, see 10.1.6.2.1 Creating an SFC Program Block.
2. Instantiate the function block and pass the parameters. For parameter details, see 10.1.6.3.6 SFC Program Block Interface Parameter.



10.1.6.8.2 SFC Commissioning

SFC supports online commissioning.



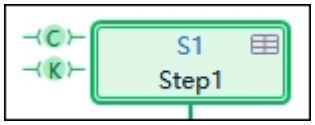
- **Status Display**

SFC displays step status, transition condition status, condition and monitor result. The steps that define conditions and monitors will display the marks -(C)- and -(K)- next to them.

- Step inactive: Gray background.



- Step active: Green background.



- Step active with monitor result FALSE: Orange background.



- Step active with monitor result TRUE: Red background.



- Transition result TRUE: Green background.



- **Commissioning mode**

  The SFC program supports four operating modes: automatic mode (default), mode with transition conditions, mode ignoring transition conditions, and manual mode. Users can switch operating modes by configuring the input parameters of the SFC program.

  - Auto mode (AUTO)
    Automatically moves to the next step when the transition condition is met.
  - Mode with transition conditions (TAP)
    Automatically moves to the next step only when the transition condition is met and a rising edge is triggered on the input parameter "T_PUSH".
  - Mode ignoring transition conditions (TOP)
    Automatically moves to the next step when the transition condition is met or a rising edge is triggered on the input parameter "T_PUSH".
  - Manual mode (MAN)
    Does not automatically move to the next step when the transition condition is met; it requires manual selection of a step or switching to the next step.

- **Commissioning panel**

  The commissioning panel includes two parts: program initialization settings and operation mode selection.

  - Initial value setting
    - Click the "Initialize" button under "Sequential controller operation" to reset the program status.
    - Click the "Deactivate all" button under "Sequential controller operation" to disable all steps within the program.
    - When an error occurs during monitoring, click the "Confirm" button under "Monitoring action" to continue executing the program.

- Operation mode selection
  - Automatic mode

    Automatically moves to the next step when the transition condition is met.
  - Semi-automatic mode

    Automatically moves to the next step when one of the following conditions is met.
    - The transition condition is met.
    - The input parameter "T_PUSH" triggers a rising edge.
    - Manually switch to the next step using the "Ignore transition condition" button.
  - Manual mode

    Users can manually switch to the next step or choose to enable or disable a specific step.

10.2 Oscilloscope

10.2.1 Basic Oscilloscope

10.2.1.1 Adding a Basic Oscilloscope

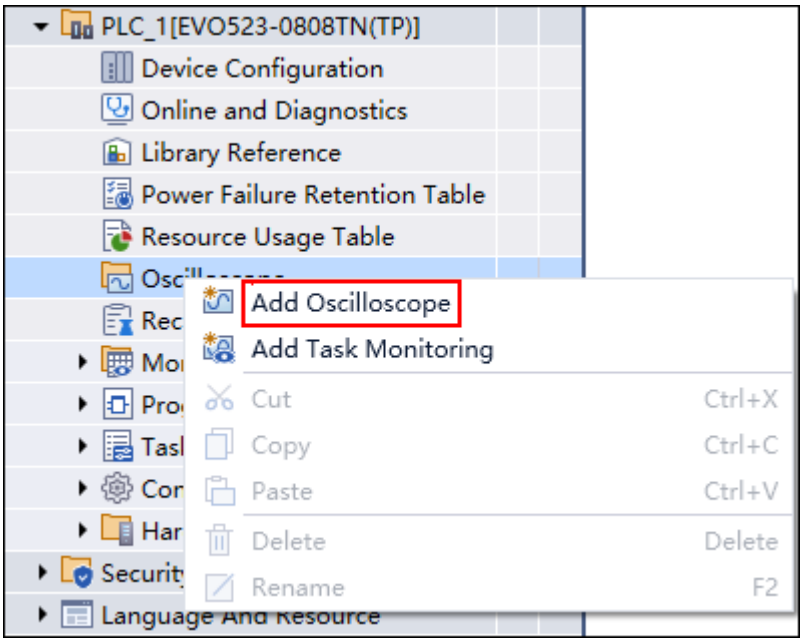
The basic oscilloscope provides real-time visualized monitoring of variables, simulating the functions of a hardware oscilloscope.

Prerequisites

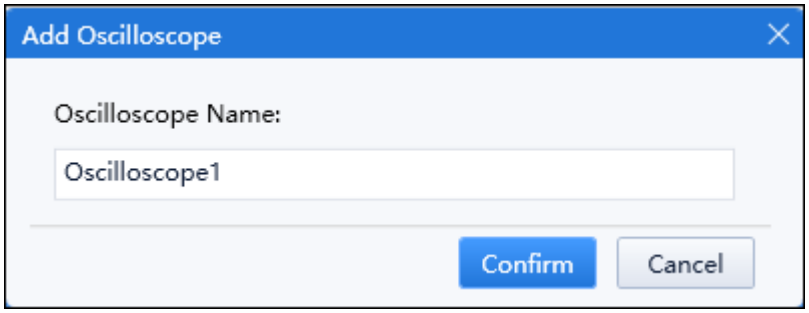
- A PLC device has been added.
- A program has been added, and variables have been created.

Procedure

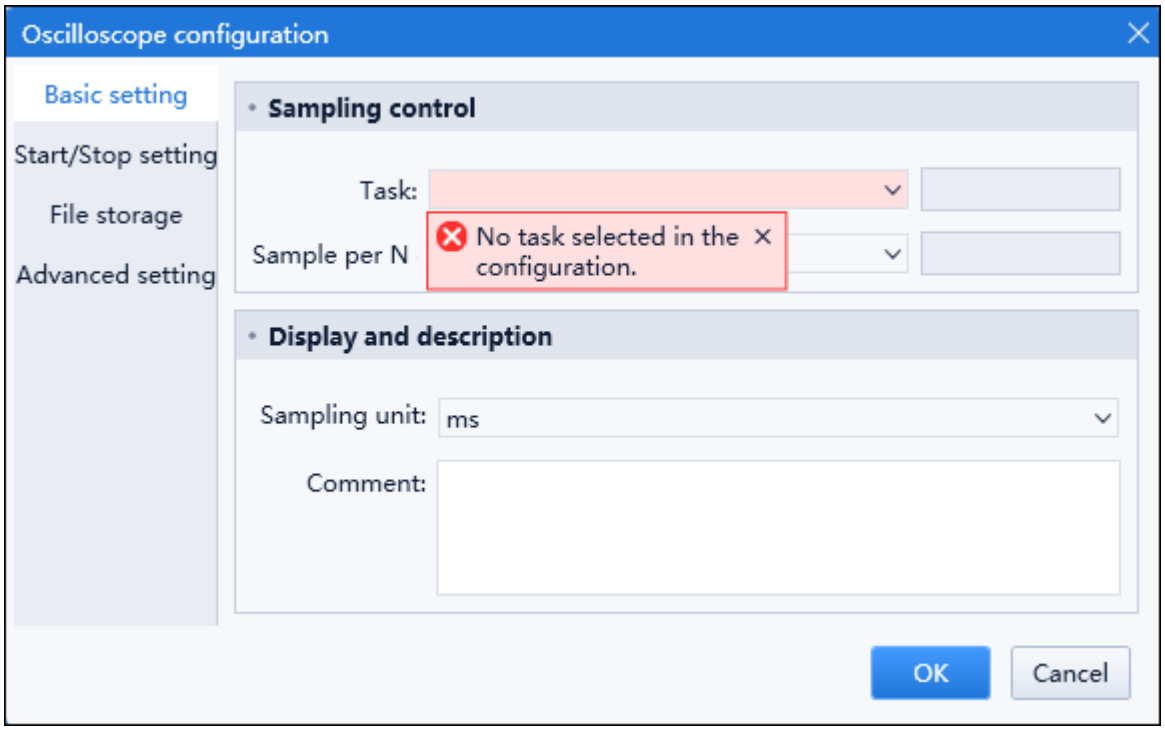
1. Right-click "Oscilloscope" in the project tree and select "Add oscilloscope".



2. In the pop-up window, set the name of the oscilloscope and click "Confirm".



3. Click "Configuration" button in the upper right corner of the interface to open the "Oscilloscope configuration" window, set the related parameters (for details, see the oscilloscope configuration introduction in 10.2.1.2 Variable Area).

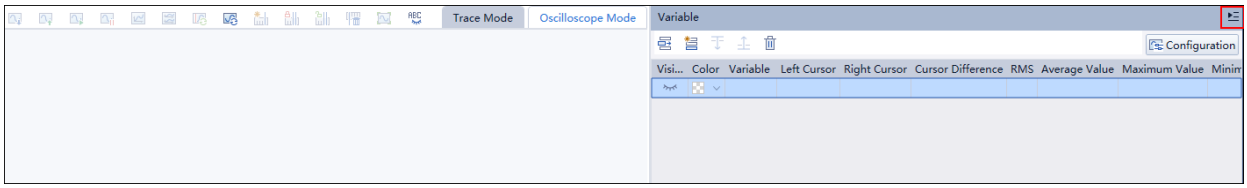


4. Click "OK" to enter the oscilloscope interface.

Result

The oscilloscope interface is divided into two parts: the plot area and the variable area.

The two areas can be resized by dragging the middle divider line. The variable area can be hidden or expanded using the "Collapse" icon, as shown in the figure below.



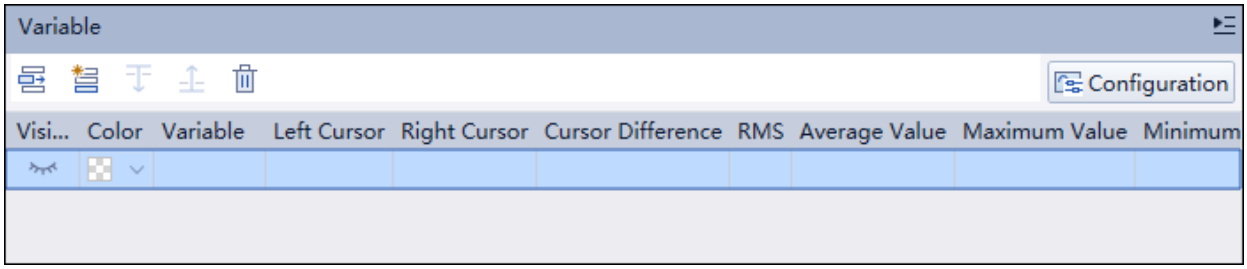
NOTE

The number of oscilloscopes supported varies by product, as shown in the table below.

	EVO400/EVO500	EVO700	EVO800
Max. number of oscilloscopes that can be added	32	64	64
Max. number of oscilloscopes that can be downloaded	32	64	64
Max. number of oscilloscopes that can run simultaneously	5	5	5

10.2.1.2 Variable Area

The variable area includes functions for adding variables and configuring the oscilloscope. The main interface is shown in the figure below.

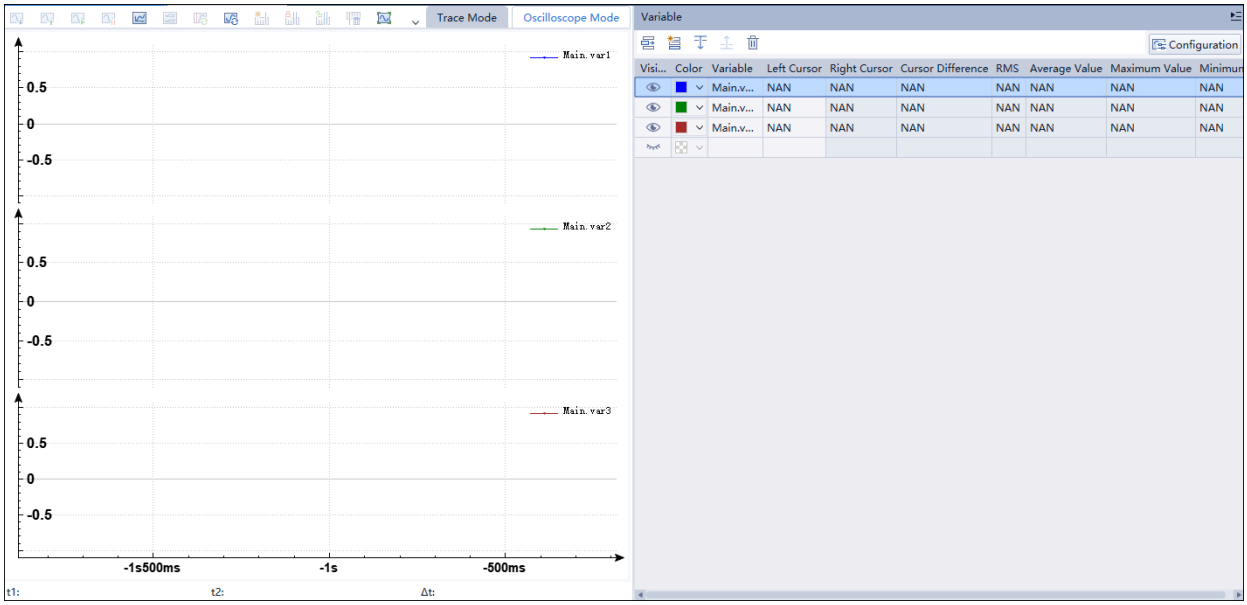


Adding a variable

There are three ways to add a variable:

- Click the "Add row" or "Insert row" icon on the toolbar, then double-click a cell in the "Variable" column to enter the variable name. This method only supports adding one variable at a time.
- Drag the variable directly from the "Details" window to the variable area of the oscilloscope to complete the addition. This method supports batch addition.
- Drag the variable directly from PRG or GV to the variable area of the oscilloscope to complete the addition. This method supports batch addition.

After the variable is added, the corresponding coordinate system will also be generated in the plot area.



NOTE

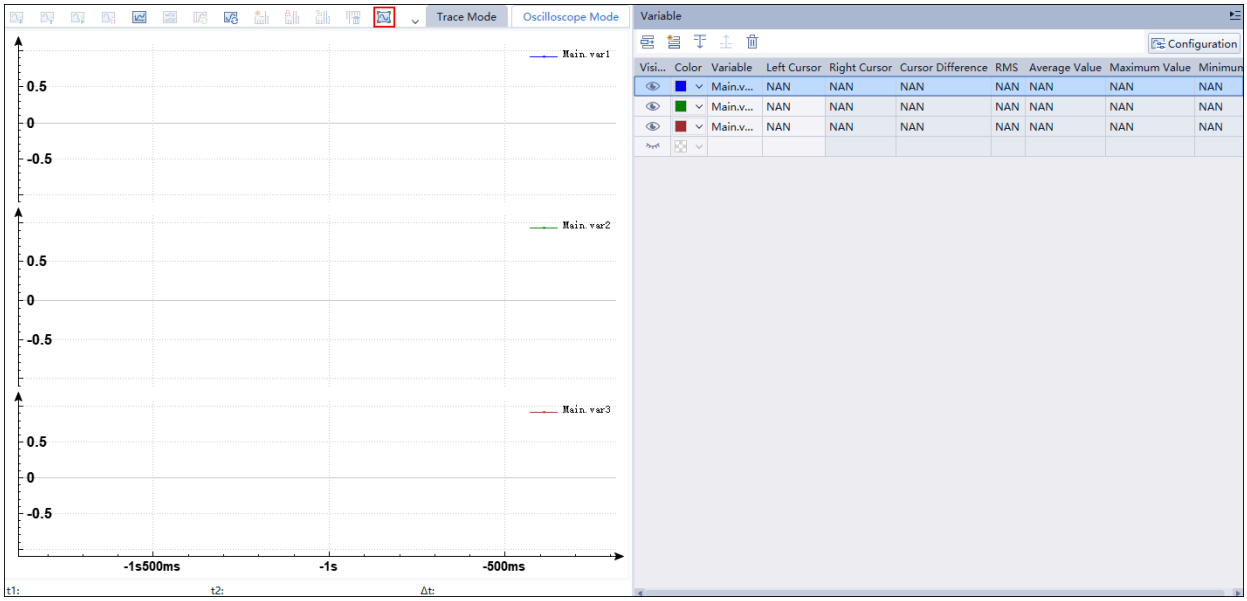
The number of oscilloscope variables supported varies by product, as shown in the table below.

	EVO400/EVO500	EVO700	EVO800
Max. number of oscilloscope variables that can be added or displayed simultaneously on a single oscilloscope	32	64	64
Max. number of oscilloscope variables that can be added to a single device	128	128	128
Max. number of oscilloscope variables that can be displayed simultaneously on a single device	64	64	64

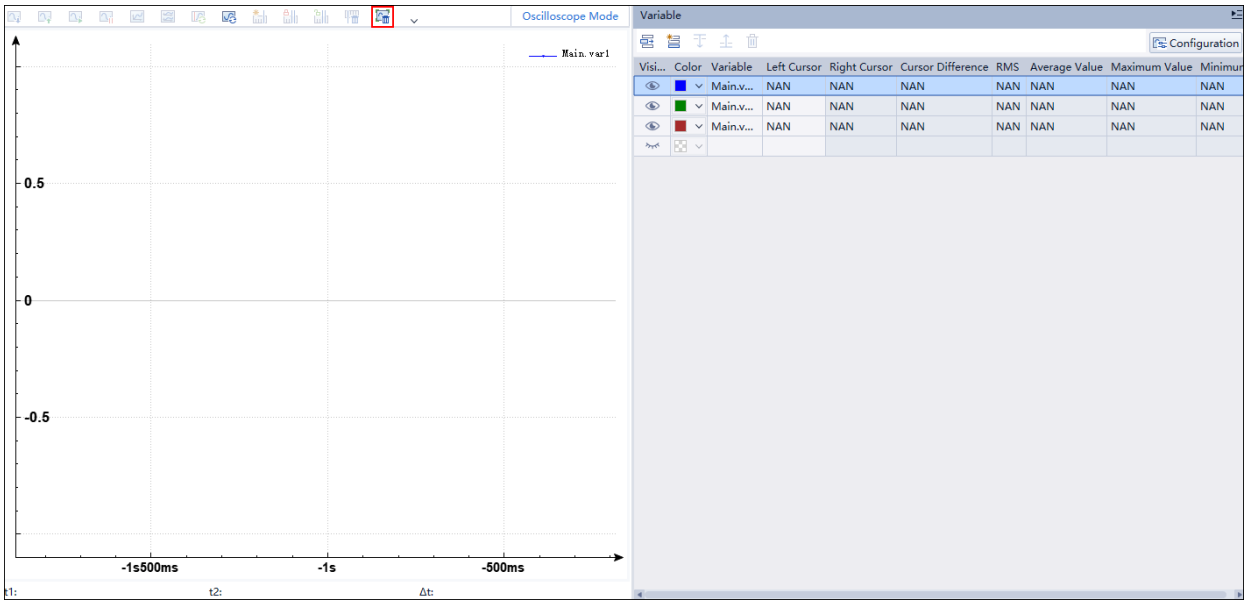
Viewing variables in multi-channel mode

In multi-channel mode, when there are too many variables, select a variable and click the "Exclusive view" icon on the toolbar. Click the same icon to cancel the exclusive view.

- Before switching to the exclusive view, the plot area displays the views of all variables.



- After switching to the exclusive view, the plot area only displays the view of the selected variable.



NOTE

This operation is supported in both online and offline modes, but note the following:

- In offline mode, the variable exclusive view only supports resetting view, saving oscilloscope data, and cursor-related functions.
- In online mode, the variable exclusive view only supports resetting trigger, resetting view, saving oscilloscope data, and cursor-related functions.

Introduction to the variable area toolbar

Description	Icon	Description
Insert a row		Inserts an empty row above.

Description	Icon	Description
Add a row		Add an empty row. Oscilloscope variables can be added to the row in the following two ways: • Input with input assistant. • Manual input with intelligent association.
Move down		Moves the selected row down.
Move up		Moves the selected row up.
Delete		Deletes the selected row. It supports batch selection deletion.

Column properties of the variable table

The variable table displays variables added to the oscilloscope, including the following column properties.

Column Properties	Description
Visible	Sets the visibility of the variable in the oscilloscope. When there is only one variable in the variable table, this function is enabled by default.
Color	Sets the color of the variable in the oscilloscope from the drop-down menu.
Variable	Sets the instance path of the variable.
Left Cursor	Displays the value of the left cursor in the oscilloscope after a left cursor is added in the plot area.
Right Cursor	Displays the value of the right cursor in the oscilloscope after a right cursor is added in the plot area.
Cursor Difference	Displays the difference between the left cursor and the right cursor.
RMS	Displays the root mean square of the variable value during the acquisition time.
Average Value	Displays the average value of the variable during the acquisition time.
Maximum Value	Displays the maximum value of the variable during the acquisition time.
Minimum Value	Displays the minimum value of the variable during the acquisition time.

Right-click menu

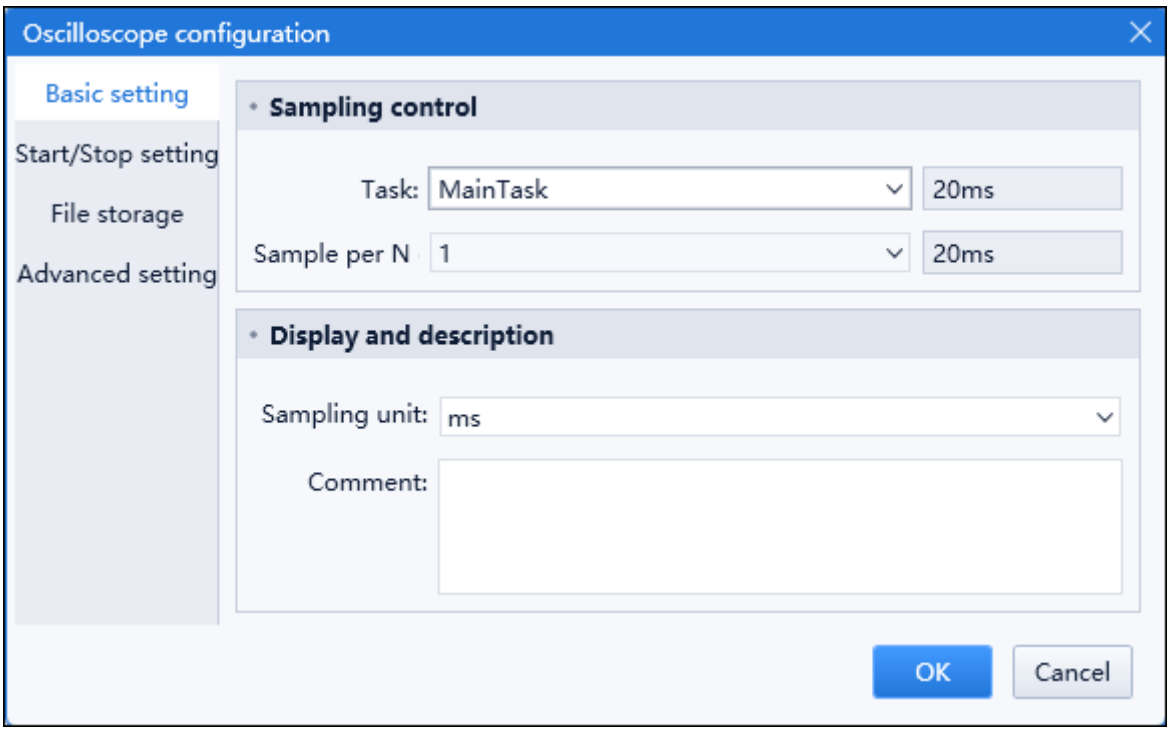
The right-click menu includes cut, copy, paste, delete, input assistant, and configuration functions.

Function	Description
Cut	Cuts the selected variable.
Copy	Copies the selected variable.
Paste	Pastes the variable.
Delete	Deletes the selected variable.
Input assistant	Opens the "Input assistant" page.
Configuration	Opens the "Oscilloscope configuration" page.

Oscilloscope configuration

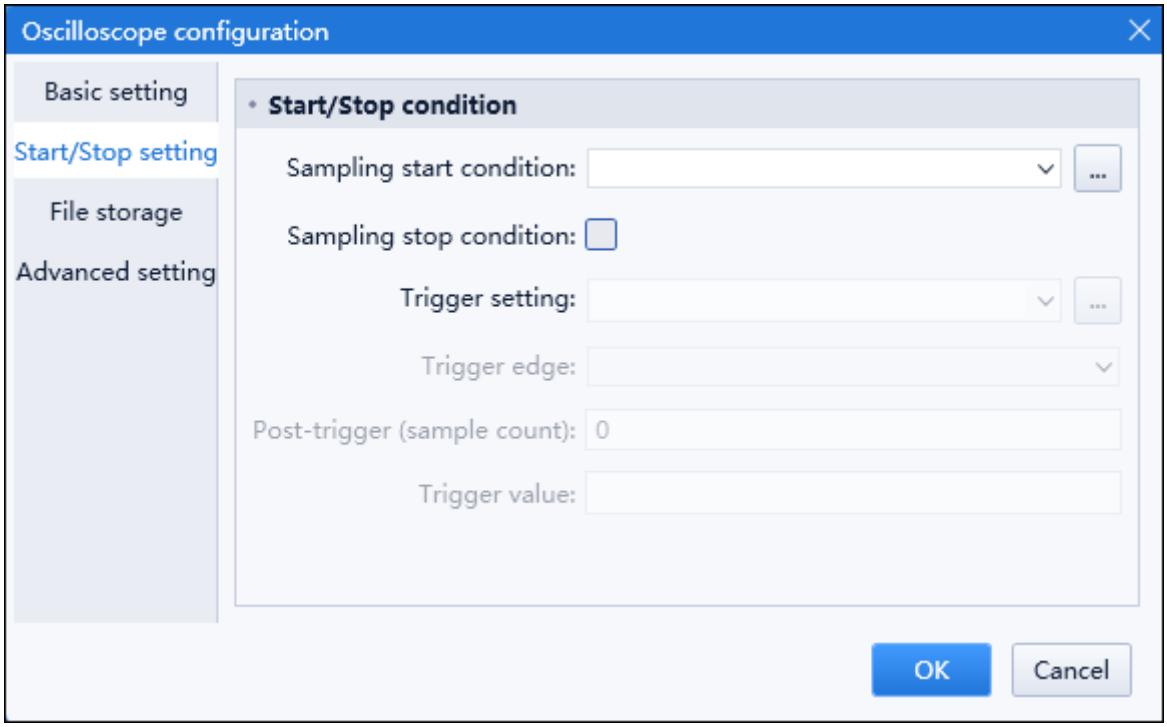
This function is used to configure oscilloscope parameters, including basic settings, start/stop settings, file storage, and advanced settings. The configuration interface is shown as follows.

- **Basic settings**



Item	Parameter	Description
Sampling control	Task	Sets the cycle period for the oscilloscope to collect data, configured by selecting tasks from the drop-down menu.
	Measure per N cycles	Sets the sampling multiplier. Select different sampling multiples from the drop-down menu. The actual sampling cycle will be displayed after setting.
Display and description	Sampling unit	Sets the minimum time interval that can be displayed on the X-axis of the channel. Units include ms and μs (default: ms), which can be selected via the drop-down menu.
	Comment	Supports entering oscilloscope comments. After selecting the oscilloscope in the project tree, comments can be displayed via TIPS.

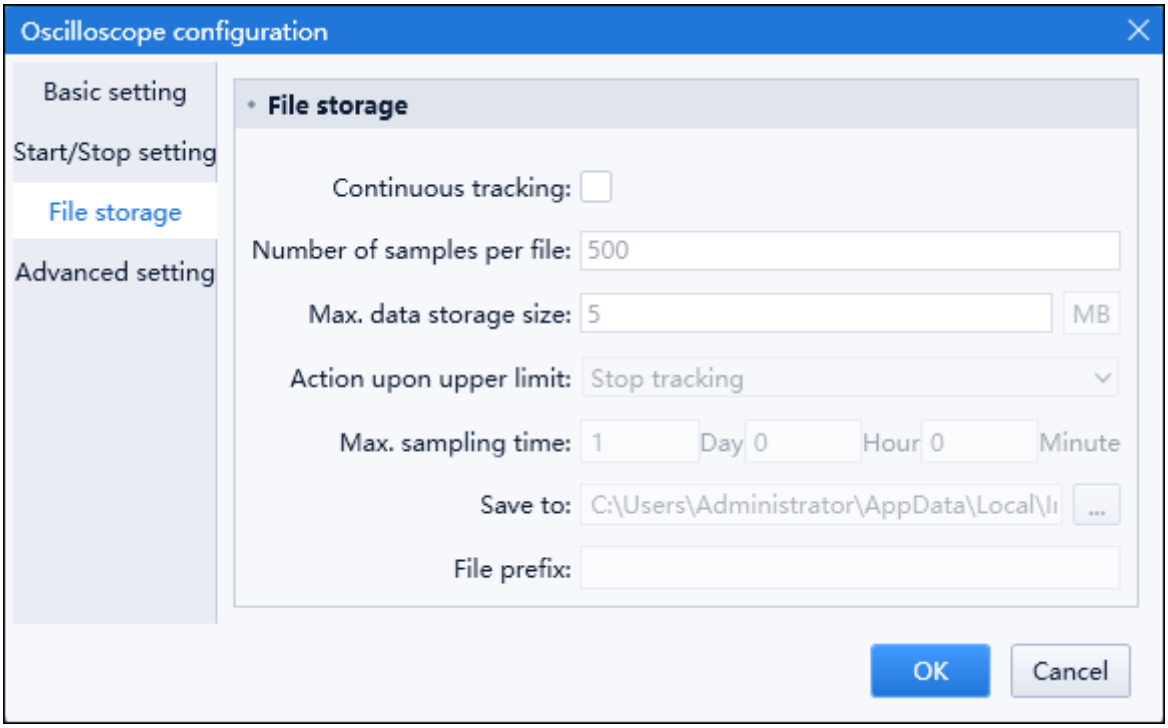
- **Start/Stop settings**



Item	Parameter	Description
Start/Stop conditions	Sampling start condition	Variable or input expression can be selected. When the expression result is TRUE, sampling starts; otherwise, sampling stops.
	Sampling stop condition	After "Sampling stop condition" is checked, the oscilloscope stops collecting and updating real-time data for the variable when the set stop condition (specific variable reaching a specified value) is met.
	Trigger settings	Trigger settings are divided into trigger variables and trigger parameters, which can be set through drop-down menu. • Trigger variable: Determines by specifying a variable in the POU. When the variable meets the configured condition, the oscilloscope stops sampling. • Trigger parameter: Determines by system parameters to stop sampling.

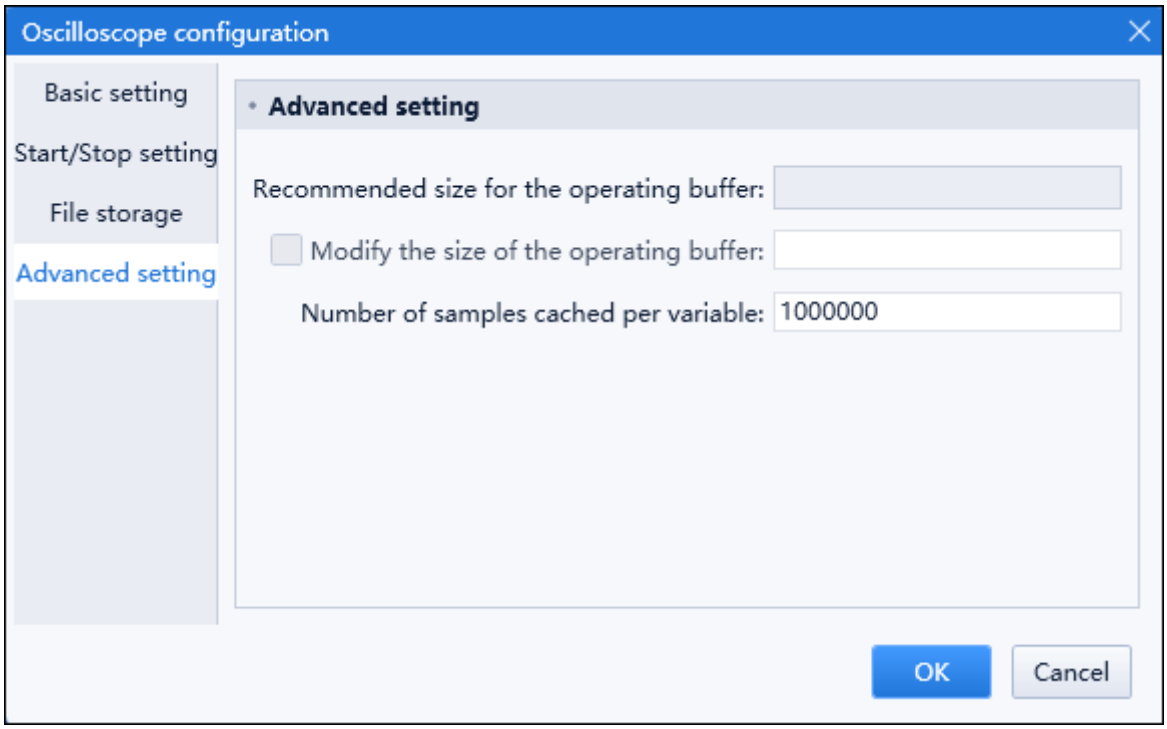
Item	Parameter	Description
	Trigger edge	Trigger edge is divided into rising edge, falling edge, and both rising and falling edges. • Rising edge ◦ For BOOL type trigger variables, triggering occurs when the BOOL value changes from FALSE to TRUE. ◦ For analog trigger variables, triggering occurs when the set value reaches the trigger value from a lower level. • Falling edge ◦ For BOOL type trigger variables, triggering occurs when the BOOL value changes from TRUE to FALSE. ◦ For analog trigger variables, triggering occurs when the set value reaches the trigger value from a higher level. • Rising and falling edges ◦ For BOOL type trigger variables, triggering occurs when the BOOL value changes. ◦ For analog trigger variables, triggering occurs when the set value reaches the trigger value.
	Post-trigger (sample count)	Sets the number of samples to continue collecting after the trigger is activated. The oscilloscope stops once the sample count is reached.
	Trigger value	Sets the trigger value of the specified trigger variable. When the variable reaches the trigger value, the oscilloscope stops sampling. The exact stop time depends on the selected trigger edge type.

- **File storage**



Item	Parameter	Description
File storage	Continuous tracking	• When "Continuous tracking" is checked, data can be collected according to the file storage settings in a custom directory. • When "Continuous tracking" is not checked, there is no need to store oscilloscope data locally and only cached data is displayed.
	Number of samples per file	Sets the number of data items that can be generated per file. After this number is exceeded, a new file needs to be generated to continue collection. The value range is 100 to 30000, with a default value of 500.
	Maximum data storage size	The value range is 5 to 1000000, with a default value of 5 MB.
	When limit is reached	When the maximum data storage is reached, you can choose to stop collection or delete old files through the drop-down menu.
	Maximum sampling time	The oscilloscope stops collection after this sampling time is exceeded. The time range is from 1 minute to 1000 days 23 hours 59 minutes.
	Save to	Sets the save path for the collected files.
	File prefix	Sets the prefix of the generated files. For example, when the prefix is set to XX, files are generated as XX, XX_1, and XX_2.

- **Advanced settings**



Item	Parameter	Description
Advanced settings	Recommended size for the operating buffer	Default is 101.
	Modify the size of the operating buffer	Used to adjust the real-time Runtime buffer; not selected by default. The buffer size range is 10 to 500,000.
	Number of samples cached per variable	The number of samples for each variable cache must be no less than twice the size of the operating buffer and no more than 1,000,000 (default).

10.2.1.3 Plot Area

Trace mode / Oscilloscope mode

Oscilloscope mode displays single-variable data in the oscilloscope.

Trace mode is generated based on the variable list in oscilloscope mode, following the index order of variables (consistent with the display order in the variable table). Each pair of adjacent variables is combined (for example, variables at index 0 and 1 are combined, and variables at index 2 and 3 are combined). If there is an odd number of variables, the last variable will not be combined with any other variable and will be displayed in oscilloscope mode.

Based on the variable list in oscilloscope mode, channels in trace mode are generated by pairing adjacent variables. The channels are divided into combined channels, unpaired channels, and non-combined channels.

Channel	Description
Combined channel	The y-value of the first variable in the pair is used as the x-axis value in trace mode, and the y-value of the second variable is used as the y-axis value, generating the channel in trace mode. The curve color in the combined channel inherits the color of the first variable in the pair from the variable table in oscilloscope mode, so is the color of the second variable in the pair.
Unpaired channel	The unpaired channel contains variables with no adjacent variables for pairing.
Non-combined channel	Variable values are displayed with separate channels. The curve color inherits the color of the variable in oscilloscope mode.

The functions in oscilloscope mode and trace mode is shown in the table below.

Mode	Description	Enabled Feature
Oscilloscope mode	Right-click menu	Cut, delete, input assistant, and configuration are enabled.
	Column properties of the variable table	Left and right cursor value, cursor difference value, and color are enabled.
Trace mode	Right-click menu	Configuration is enabled.
	Column properties of the variable table	Left and right cursor and cursor difference are enabled.

NOTE

- In trace mode, combined channels do not support cursors, upper/lower limit, or Y-axis scaling functions. Non-combined channels support cursors, upper/lower limit, and Y-axis scaling functions.
- Single channel and multi-channel functions are not supported in trace mode. The number of channels is determined by the number of adjacent variable pairs and unpaired variables.

Introduction to the plot area toolbar

Description	Icon	Description
Download		Downloads IEC oscilloscope variables and configuration information to the currently connected target device.
Upload		Uploads the oscilloscope variables and configuration information from the currently connected target device to IEC.
Start		Starts the current oscilloscope operation and the real-time data will be echoed to form the waveform of the oscilloscope variables.
Stop		Stops the current oscilloscope operation, real-time data echoing, and the plot of the oscilloscope variable waveforms.
Single channel		Single channel and multi-channel functions determine the layout of oscilloscope variables. In single channel mode, there is only one display view, and all oscilloscope variable tracing results are displayed in this view.
Multi-channel		In multi-channel mode, each oscilloscope variable has an independent display view, and the tracing results of each variable are displayed separately in their own views.
Reset trigger		After the trigger stops sampling, resetting the trigger can start sampling again.
Reset view		The view can be reset by dragging and zooming.
Add cursor		Cursors are used to measure the waveform values between two points of an oscilloscope variable. • Cursors can be added when the oscilloscope is online and stopped. • Select "Add cursor" and a draggable red line indicating the cursor will be displayed on the corresponding view. • A maximum of two cursors (left cursor and right cursor) are supported per view. • The waveform values of the left and right cursors will be displayed in the variable table.
Lock cursor		• Cursors can be locked when there are two cursors in the current view. • Select "Lock cursor" icon and the relative positions of the two cursors in the current view will be locked. Moving either cursor will cause both cursors to move synchronously.
Unlock cursor		• Locked cursors can be unlocked. • Click the "Unlock cursor" icon and the two locked cursors in the current view will be unlocked.
Delete cursor		• Cursors in the current view can be deleted. • Click the "Delete cursor" icon to delete the cursors from the current view.

Description	Icon	Description
Exclusive view		In multi-channel mode, when there are too many variables, you can select a specific variable for an exclusive view. After switching to the exclusive view, the plot area will only display the view of the selected variable.
Cancel exclusivity		After canceling the exclusive view, the plot area will display the views of all variables.
Hide legend		Hides the legends in the plot area.
Show legend		Show the legends in the plot area.

Right-click menu

Description	Icon	Description
Cut		Cuts variables from the variable table.
Copy		Copies variables from the variable table.
Paste		Pastes variables into the variable table.
Delete		Deletes variables from the variable table.
Input assistant		Opens the "Input assistant" page.
Download		Downloads IEC oscilloscope variables and configuration information to the currently connected target device.
Upload		Uploads the oscilloscope variables and configuration information from the currently connected target device to IEC.
Start oscilloscope		Starts the current oscilloscope operation and the real-time data will be echoed to form the waveform of the oscilloscope variables.
Stop oscilloscope		Stops the current oscilloscope operation, real-time data echoing, and the plot of the oscilloscope variable waveforms.
Single channel		Single channel and multi-channel functions determine the layout of oscilloscope variables. In single channel mode, there is only one display view, and all oscilloscope variable tracing results are displayed in this view.
Multi-channel		In multi-channel mode, each oscilloscope variable has an independent display view, and the tracing results of each variable are displayed separately in their own views.
Reset trigger		After the trigger stops sampling, resetting the trigger can start sampling again.
Reset view		The view can be reset by dragging and zooming.

Description	Icon	Description
Add cursor		Cursors are used to measure the waveform values between two points of an oscilloscope variable. • Cursors can be added when the oscilloscope is online and stopped. • Select "Add cursor" and a draggable red line indicating the cursor will be displayed on the corresponding view. • A maximum of two cursors (left cursor and right cursor) are supported per view. • The waveform values of the left and right cursors will be displayed in the variable table.
Lock cursor		• Cursors can be locked when there are two cursors in the current view. • Select "Lock cursor" icon and the relative positions of the two cursors in the current view will be locked. Moving either cursor will cause both cursors to move synchronously.
Unlock cursor		• Locked cursors can be unlocked. • Select "Unlock cursor" and the two locked cursors in the current view will be unlocked.
Clear cursors		• Cursors in the current view can be deleted. • Select "Delete cursor" to delete the cursors from the current view.
Add upper limit cursor		The upper/lower limits are used to determine a maximum/minimum baseline to help users observe the parts of the oscilloscope variable waveform that exceed the limits. Select "Add upper limit cursor" and a red horizontal line will be drawn on the view.
Add lower limit cursor		Select "Add lower limit cursor" and a green horizontal line will be drawn on the view.
Clear all limit cursors		• Limit cursors can be cleared when there are limits in the current view. • Select "Clear all limit cursors" to clear the limits in the current view.
Load oscilloscope data		Loads local oscilloscope files into IEC and displays them in the interface. The oscilloscope files include the following content: • Configuration information of the oscilloscope object are loaded the configuration interface. • Variable information of the oscilloscope is loaded to the variable table. • Data of the oscilloscope variables is loaded to the corresponding channels.

Description	Icon	Description
Save oscilloscope data		Saves the oscilloscope data in IEC to the local storage. The data can be saved in two formats: ".oscilloscope" and ".csv". • The ".oscilloscope" file is stored in XML form, containing information of the configuration and data of the oscilloscope. • The ".csv" file is stored in TXT form, containing basic configuration data and oscilloscope variable data.
Configuration		Opens the "Oscilloscope configuration" page.

Operations in plot area

Function	Method
Horizontal movement of X-axis	Hold the left mouse button and drag left/right.
Vertical movement of Y-axis	Hold Ctrl and left mouse button, and drag up/down.
X-axis zoom	Scroll the mouse wheel.
Y-axis zoom	Hold Ctrl and scroll the mouse wheel.

10.2.1.4 Loading Fault/Data Trace Files

The fault/data trace file loading function reads saved faults or program data files for visualized replay of the operation process, enabling analysis and diagnosis of historical operating conditions.

Restrictions

This function is applicable to the following PLC models and firmware versions:

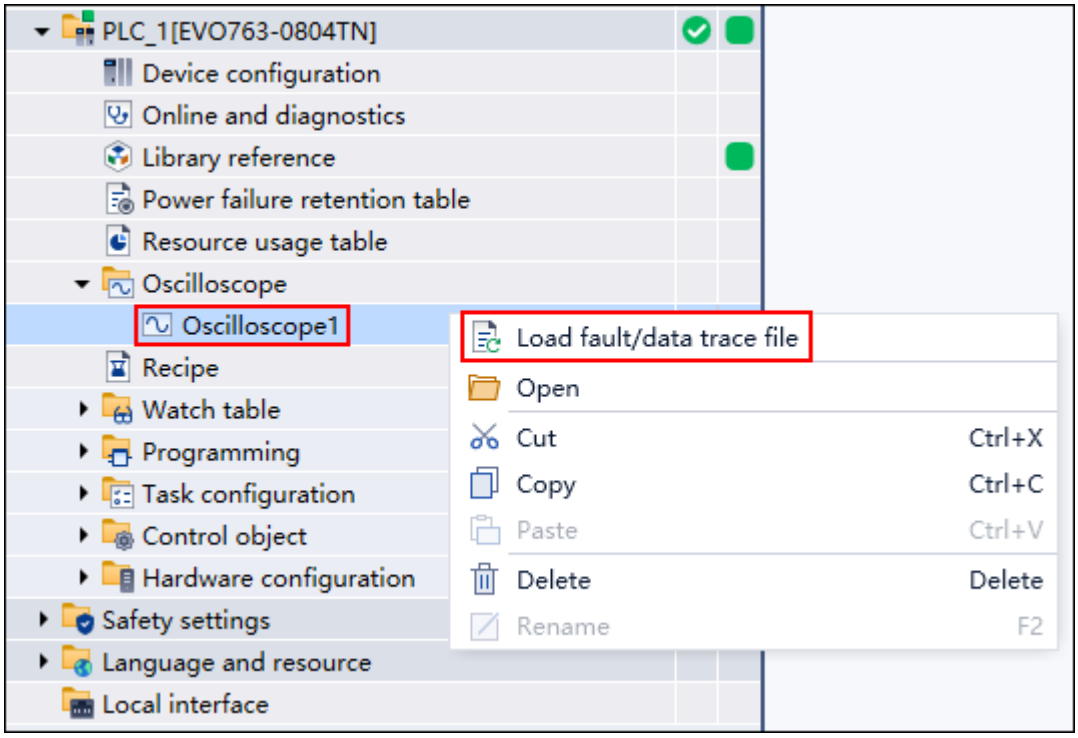
- EVO500, EVO700, and APC series: Firmware version V1.6 and above.
- EVO800 series: Firmware version V1.1 and above.

Prerequisites

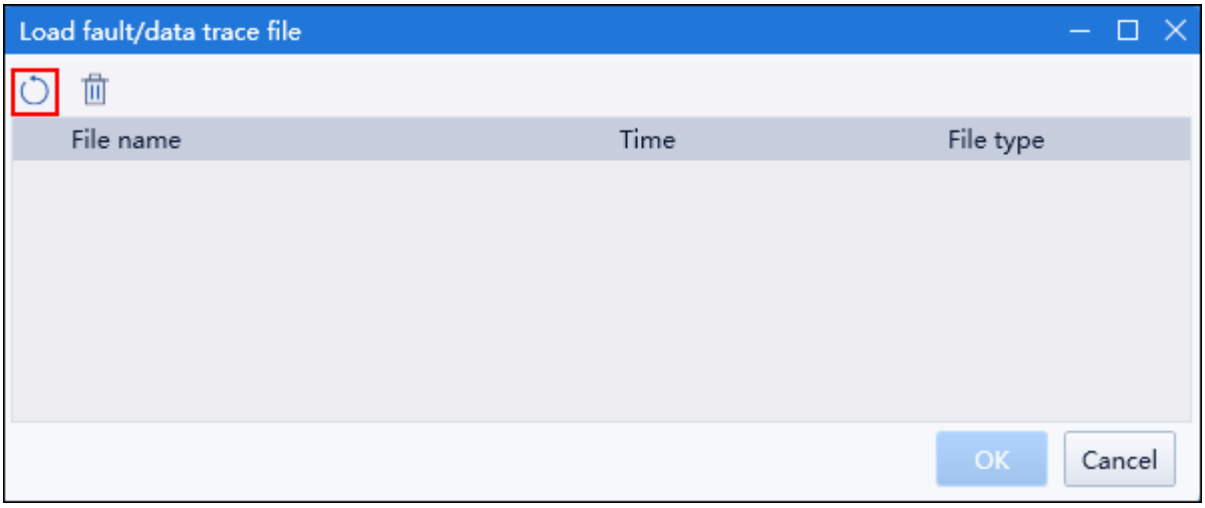
- The controller is equipped with an external storage device (SD card or USB drive).
- A basic oscilloscope has been added. For details, see 10.2.1.1 Adding a Basic Oscilloscope.
- The device is in an online status.

Procedure

1. Right-click an existing basic oscilloscope in the project tree and select "Load fault/data trace file".



2. Click in the pop-up window. The system will automatically load the saved faults or program data files from the external storage device of the controller.



3. Select the target file from the list and click "OK". A prompt window will appear, indicating loading data will overwrite the current oscilloscope. Click "Yes", and the data from the selected file will overwrite the current content on the oscilloscope. Users can replay historical operating states and analyze oscilloscope data changes for troubleshooting.

4. If the loaded oscilloscope is closed, a dialog box will appear asking whether to save the trace file. Select based on actual needs and follow the prompts in the dialog box.

10.2.2 Task Oscilloscope

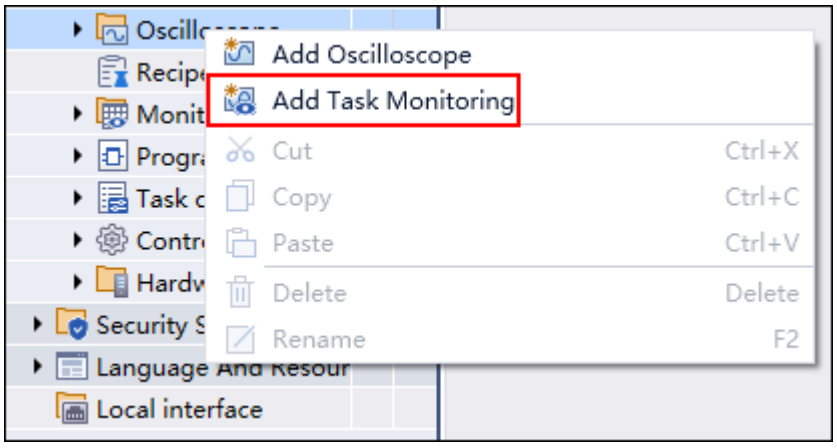
In the task oscilloscope, users can add a target task and monitor the last loop time of the task during online operation.

Prerequisites

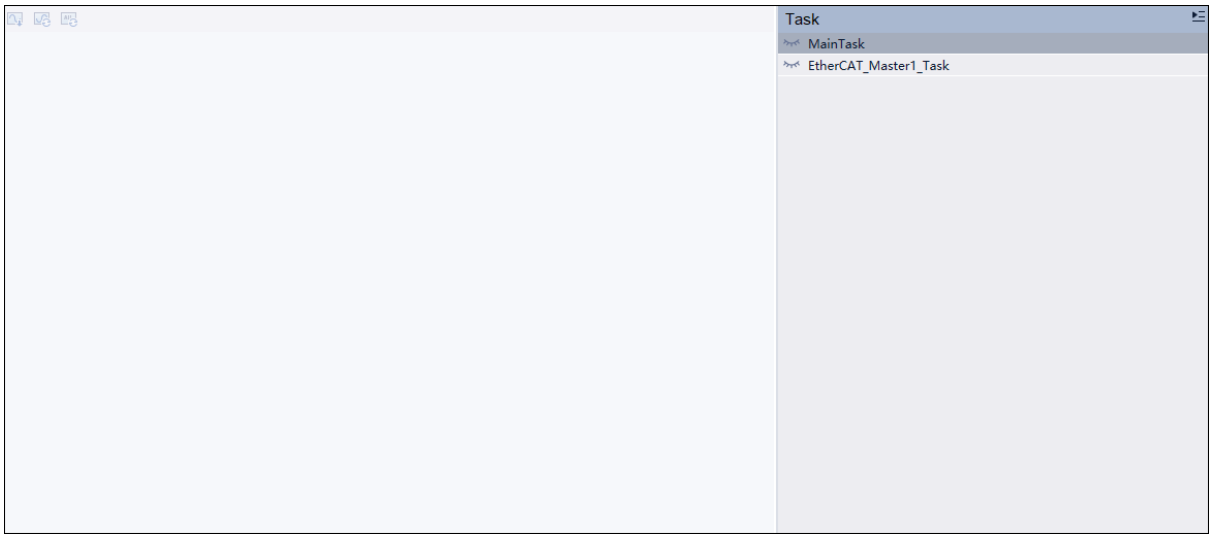
- A program has been created and added to the task.
- The task has been switched to online operation.

Procedure

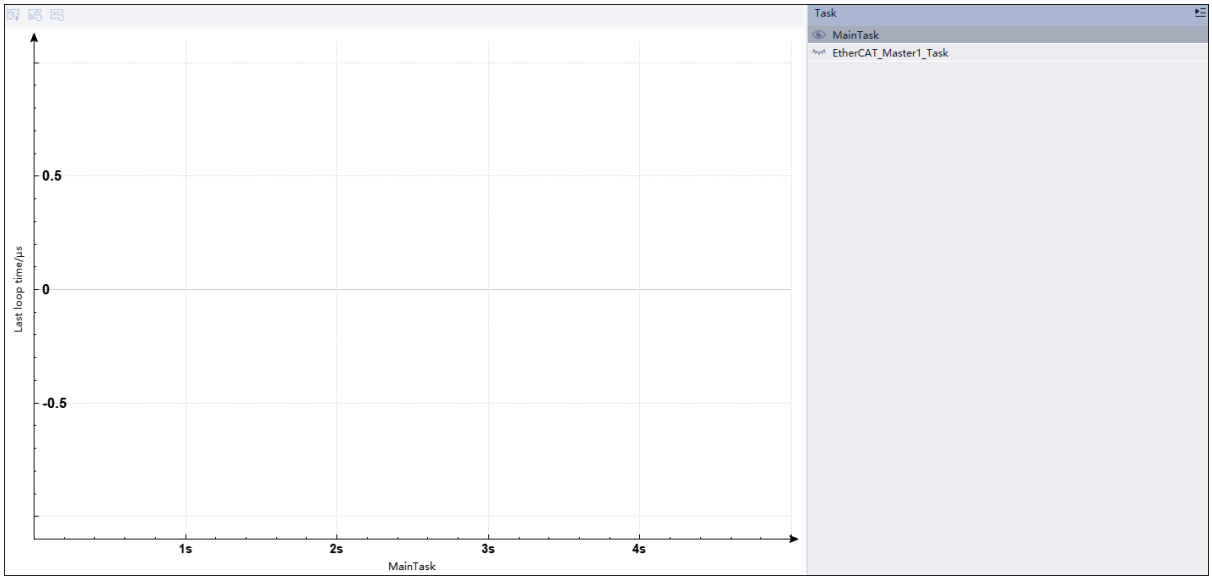
1. Right-click the "Oscilloscope" node in the project tree and select "Add task monitoring".



2. The task oscilloscope interface is opened. The interface is divided into the task list and the task oscilloscope area, as shown in the figure below.



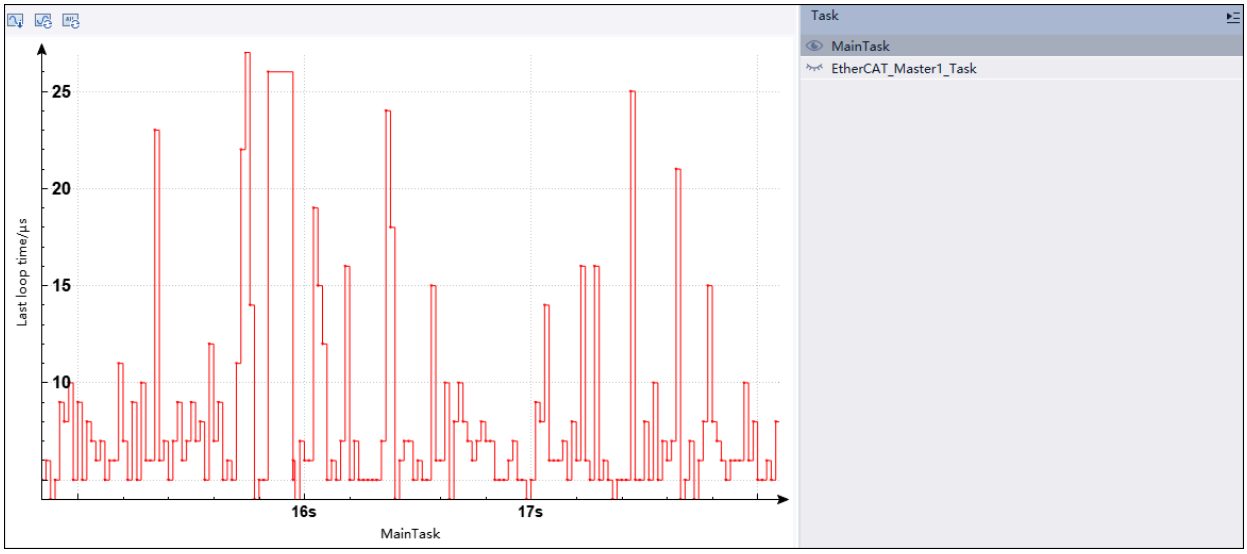
3. Select the task to be monitored in the task list.



4. Switch to online operation, click the "Download" icon on the toolbar to monitor the jitter information of the "Last loop time" parameter of the task.

Result

The task jitter information (last loop time) collected by the task oscilloscope is shown in the figure below.



Introduction to task oscilloscope functions

Description	Icon	Description	Conditions for Use
Download		Download the task monitoring information to the currently connected target device.	The task is in online status.
Reset view		The view can be reset by dragging and zooming.	The selected channel can be scrolled/dragged.
Reset all views		All channel views can be reset by dragging and zooming.	Channels can all be scrolled/ dragged.

NOTE

- The number of channels in the monitoring page is the same as the number of selected tasks. Each channel generates an X-axis label to indicate the corresponding task.
- Data can only be collected in online status, not in offline status.
- When a monitored task is deleted in the "Task configuration" window, the corresponding monitoring object of the task is also deleted.

Operations in the task oscilloscope interface

Function	Method
Horizontal movement of X-axis	Hold the left mouse button and drag left/right.
Vertical movement of Y-axis	Hold Ctrl and left mouse button, and drag up/down.
X-axis zoom	Scroll the mouse wheel.
Y-axis zoom	Hold Ctrl and scroll the mouse wheel.

10.3 Recipe

10.3.1 Configuring a Recipe

Recipes can be added in the "Recipe" section of the project tree and can be used for managing initial values of variables.

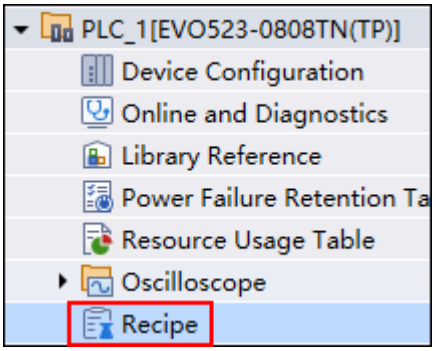
This section introduces how to set recipe-related parameters.

Prerequisites

- A PLC device has been added.
- Variables have been created.

Procedure

1. Double-click the "Recipe" node in the project tree to open the "Recipe" page.

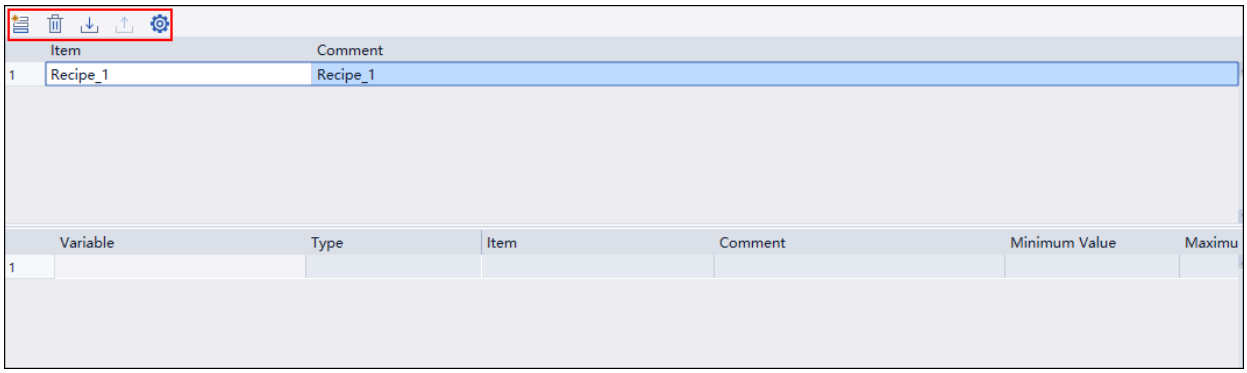


2. Click the "Add" icon in the upper left corner of the page to open the main recipe interface.



Main recipe interface

The main recipe interface includes a toolbar, a recipe definition list, and a recipe parameter list, as shown in the figure below.



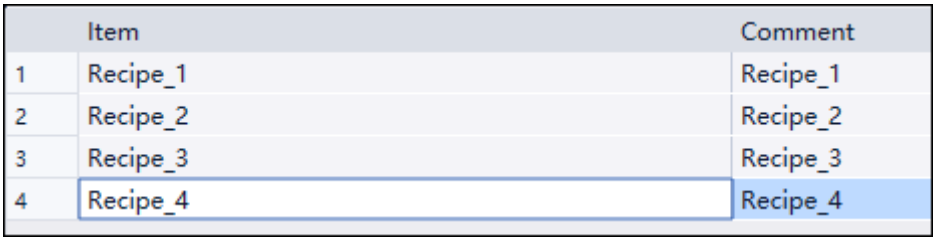
Toolbar

The functions of each icon in the toolbar are described in the table below.

No.	Name	Icon	Description
①	Add		Creates a recipe.
②	Delete		Deletes a recipe.
③	Import		Imports recipes.
④	Export		Exports the recipes in the list.
⑤	Configuration		Opens the recipe configuration page.

Detailed usage instructions for the toolbar functions are as follows.

- **Adding**
  Click the "Add" icon to automatically add a recipe definition to the list, with the default name "Recipe_n" (n is a number), as shown in the figure below.



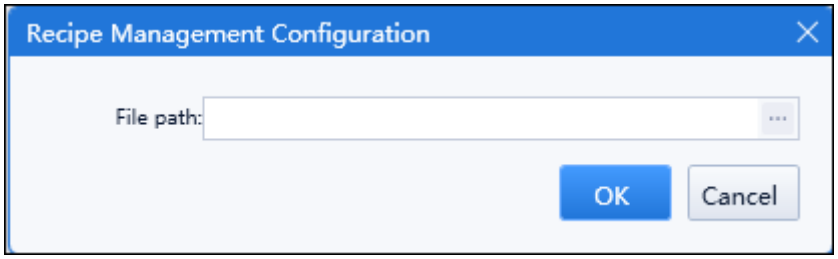
- **Delete**
  Select a row in the recipe list and click the "Delete" icon. Click "OK" in the pop-up confirmation window to delete the selected recipe.

  Recipes can also be deleted in batches by batch deletion using Ctrl and Shift.

- **Importing**
  Click the "Import" icon to open the "Import recipe" window and select the recipe to import.

- **Exporting**
  Right-click a recipe and click the "Export" icon. In the pop-up "Export recipe" window, select the save format as "*.txtrecipe". Batch export is not supported.

- **Configuring**
  Click the "Configure" icon to open the "Recipe management configuration" dialog box. Click the "..." button to set the default recipe import and export paths, as shown in the figure below.



Recipe definition list

The recipe definition list interface is shown in the following figure:

	Item	Comment
1	Recipe_1	Recipe_1
2	Recipe_2	Recipe_2
3	Recipe_3	Recipe_3
4	Recipe_4	Recipe_4

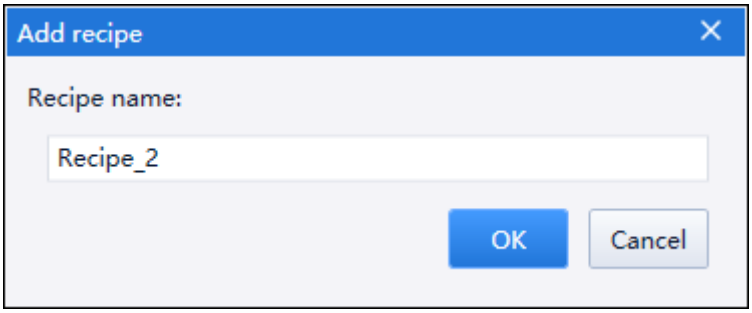
The specific functions of the recipe definition list are as follows:

Function	Description
Right-click menu	Supports copying, pasting, deleting, cutting, and adding recipes.
Data property column	Includes sequence number, name, and comment columns, where the name and comment columns can be modified.
Online status	For online recipes, functions including adding, pasting, deleting, and cutting are grayed out and disabled, while copying is enabled.
Offline status	• For offline recipes, when a recipe is selected, the copying, deleting, and cutting functions are enabled; when no recipe is selected, the copying, deleting, and cutting functions are disabled. Batch operation is supported. • When copying a recipe, the pasting function is enabled; otherwise, it is disabled. • Adding a recipe is enabled. • When deleting a recipe, a confirmation dialog box will pop up. Click "Confirm" to delete the recipe and click "Cancel" to cancel the deletion.

The right-click menu in the recipe definition list is shown in the following figure.

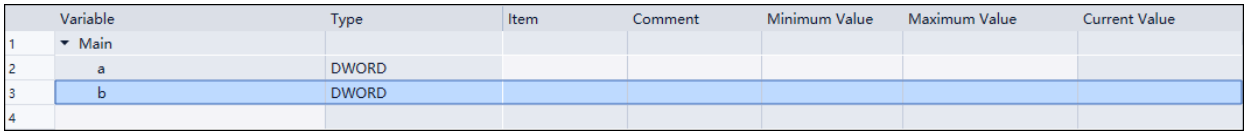


Click "Add recipe definition" and enter the recipe name in the pop-up "Add recipe" window. The default name is "Recipe_n", where n is a number, and automatically increments by 1.



Recipe parameter list

The recipe parameter list (including variable, type, name, comment, minimum value, maximum value, current value) is shown in the following figure.



	Variable	Type	Item	Comment	Minimum Value	Maximum Value	Current Value
1	Main						
2	a	DWORD					
3	b	DWORD					
4							

Descriptions of each parameter:

Column	Description	Precautions
Variable	Variables added in the parameter list	Does not support adding duplicate variables. When there is an empty row, only the "Variable" column can be modified (to add a variable). • Supports adding all global variables (including system variables) under the "Programming" node. • Supports adding all variables in PRGs under the "Programming" node. There are four ways to add variables: • Method 1: Add variables and types through the intelligent input, where you can directly enter the variable name in the edit box and there will be intelligent prompts. • Method 2: Add variables and types through the input assistant. Click the "..." button in the variable edit box to enter the input assistant window and select the variable. • Method 3: Copy and paste data from other recipe parameter tables. • Method 4: Drag and drop data from other recipe parameter tables. Each recipe can add up to 100 variables.
Type	Data types of variables	The following data types are supported: • Basic data types • Custom Type • FB • Array The type column cannot be modified. It is not supported to add constants, I-zone variables, temporary variables, input/output variables, pointer type variables, reference type variables, and Any type variables.
Item	Variable names	Default is empty. Only the last child node (i.e., the basic type variable column) can be modified.
Comment	Variable comments	The comment length should be less than 128 characters. Variable comments are different from the comments in the variable table.

Column	Description	Precautions
Minimum value	Minimum values set for variables	After using the recipe, the value set in the background cannot be less than this value.
Maximum value	Maximum values set for variables	After using the recipe, the value set in the background cannot be greater than this value.
Current value	Current values of the variables in the device	• The current value cannot be modified. • In online mode, the variable value is automatically loaded, or it can be written via the write operation. If maximum/minimum values are set, and the current recipe value is greater/less than the maximum/minimum value, the value written to the PLC is the maximum/minimum value; otherwise, the value written is the current recipe value. • When the current value is forced, the write operation is disabled and the forced value cannot be modified. • When the current value is of BOOL type, the value has a display color.
Recipe column	Variable recipes	For detailed configuration, see 10.3.2 Adding a Recipe.

Descriptions of the right-click menu functions for the recipe parameter list:

Function	Description	Enabled Feature	
		Offline Status	Online Status
Cut	Cuts variables from the parameter table.	Enabled when variables are selected; otherwise, it is grayed out and disabled.	Grayed out and disabled.
Copy	Copies variables from the parameter table.	Enabled when variables are selected; otherwise, it is grayed out and disabled.	Enabled when variables are selected; otherwise, it is grayed out and disabled.
Paste	Pastes variables into the parameter table.	Enabled after cutting or copying; otherwise, it is grayed out and disabled.	Grayed out and disabled.
Delete	Deletes variables from the parameter table.	Enabled when variables are selected; otherwise, it is grayed out and disabled.	Grayed out and disabled.

Function	Description	Enabled Feature	
		Offline Status	Online Status
Go to Definition	Go to the corresponding definition of the selected column property.	• Enabled when the "Variable" column property is selected. • Enabled when the "Type" column property is selected and the type is custom (DUT, FB); other types are not enabled. • Not enabled when any column property other than "Variable" or "Type" is selected.	Enabled when the "Variable" column property is selected.
Cross-reference	Opens the cross-reference table for the selected variables in the monitoring area.	Enabled when the variable name is selected for cross-reference.	Enabled when the variable name is selected for cross-reference.
Adding a Recipe	Adds a recipe for the selected variable.	All enabled.	All enabled.

10.3.2 Adding a Recipe

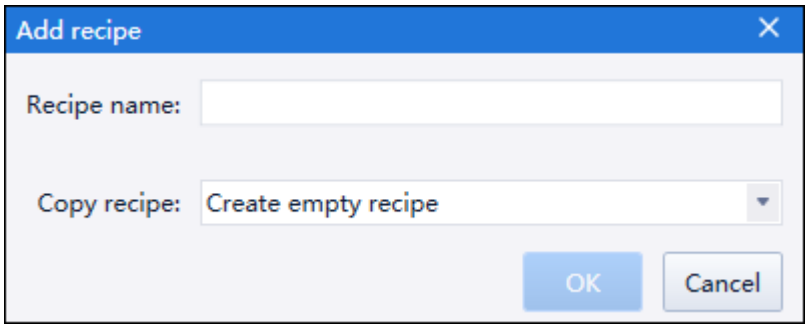
This section introduces how to add a recipe in the recipe manager.

Prerequisites

- The project has been created.
- Recipes have been defined in the recipe manager, and parameters have been added.

Procedure

1. Right-click the recipe parameter list and select "Add recipe" to enter the "Add recipe" page, as shown in the following figure.



2. Set the "Recipe name", select the recipe to be copied in "Copy recipe", and click "OK" to add the recipe.

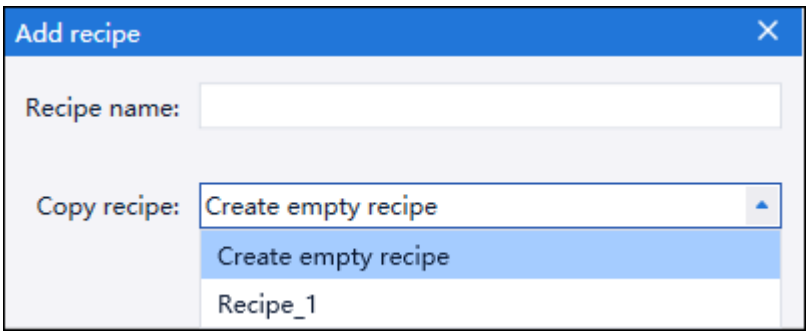
Result

The added recipe is shown in the following figure.

	Variable	Type	Name	Comment	Min. value	Max. value	Current value	Recipe_1
1	Main							
2	a	INT						
3	b	INT						
4								

Descriptions of the "Copy recipe" function

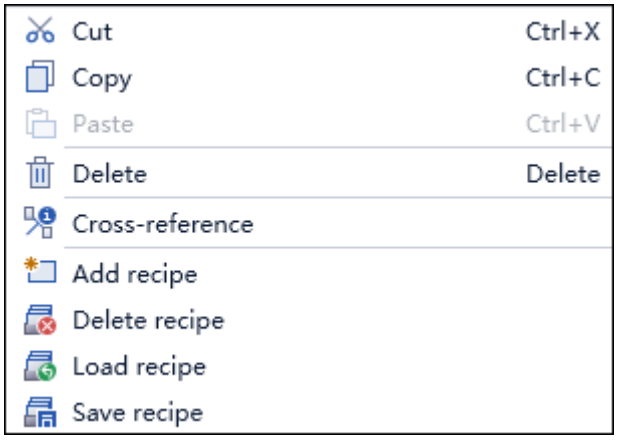
1. The default option in the "Copy recipe" drop-down menu is "Create empty recipe".
2. When there are no recipes in the recipe parameter list, only "Create empty recipe" is displayed in the drop-down menu. When there are recipes in the list, existing recipes are also displayed in the drop-down menu. Custom input is not allowed, as shown in the following figure.



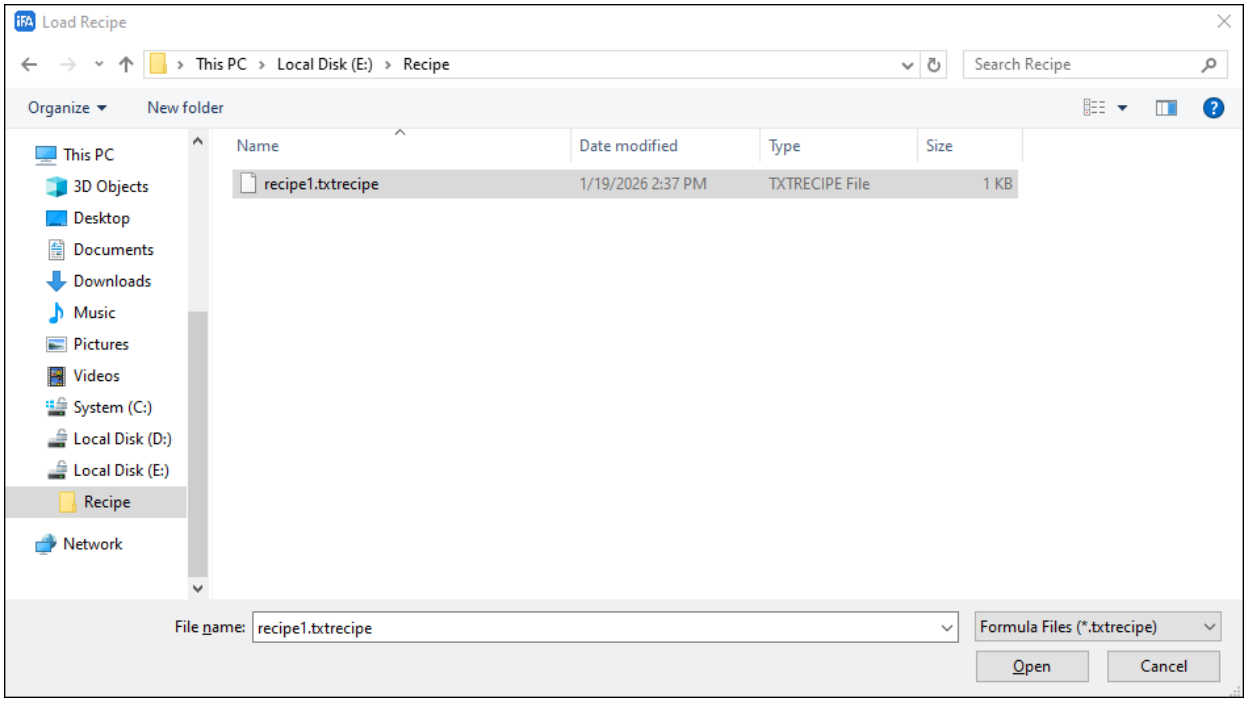
3. When adding a new recipe by selecting an existing recipe, the values of the selected recipe will be copied.

Right-click menu

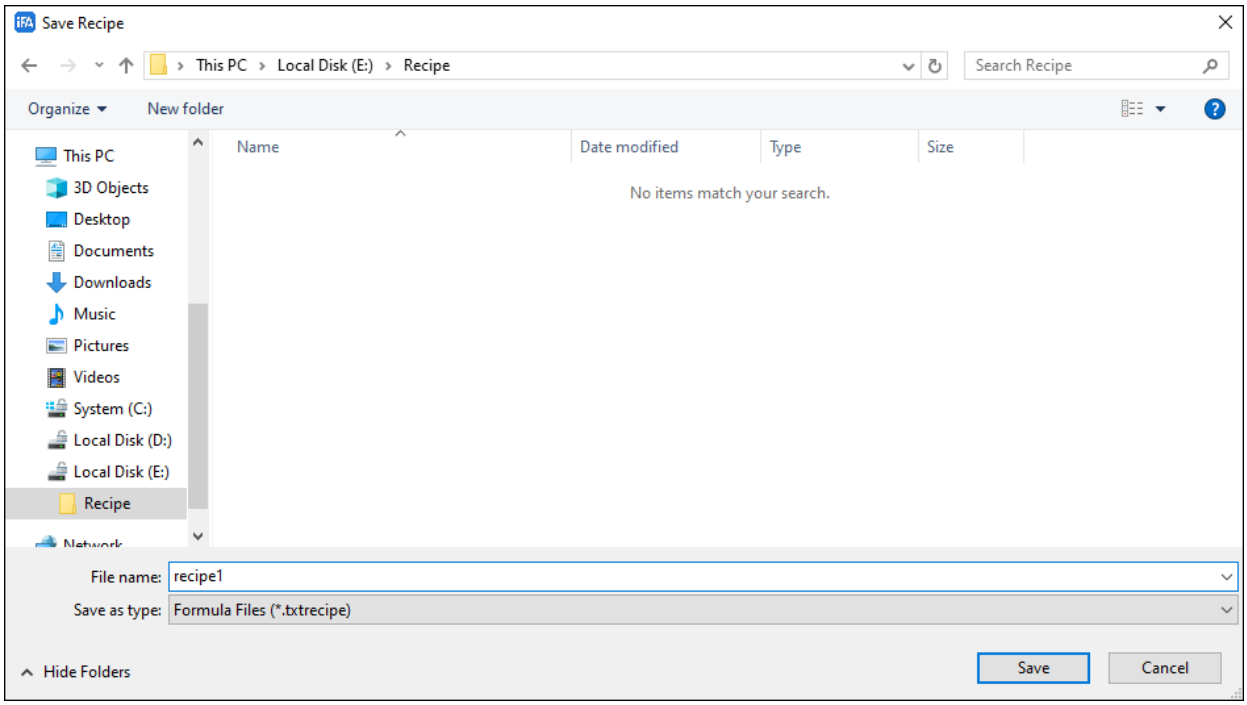
Right-click the "Recipe" column and the menu is shown in the following figure.



- You can perform cut, copy, paste, delete, and cross-reference operations for the selected variable.
- Add recipe: Adds a new recipe.
- Delete recipe: Deletes the selected recipe. Batch deletion is not supported.
- Load recipe: Loads from a local file, read the parameters of the saved recipe, and write them into the selected recipe, as shown in the following figure.



- Save Recipe: Saves the parameters of the selected recipe to a local file with a save file format of ".txtrecipe", as shown in the following figure.



10.3.3 Online Recipe

This section introduces the right-click menu functions of an online recipe.

Prerequisites

- The recipe manager has been created and recipes have been added.
- The recipe has been switched to online status.

Right-click menu functions

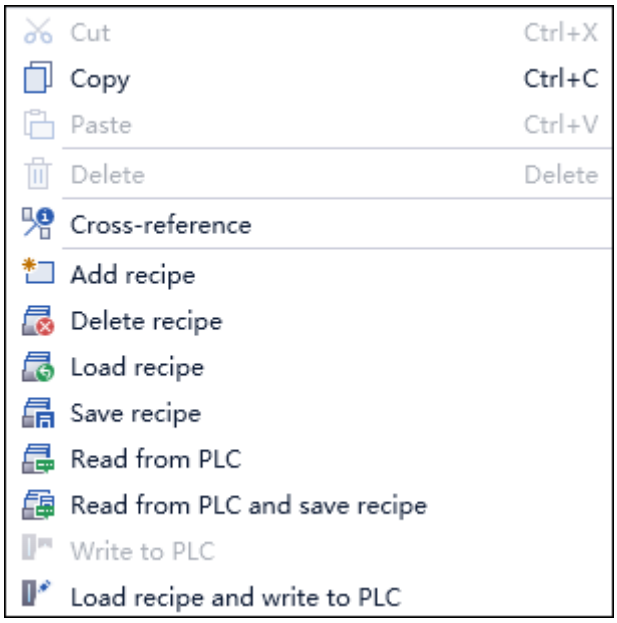
After the recipe is switched to online status, the enabling conditions of the right-click menu functions for the recipe definition list and recipe parameters change.

- Recipe definition list
  The functions to add a recipe, paste, cut, and delete in the recipe list are grayed out and disabled, while copy is enabled.
- Recipe parameter table



Description	Enabled Feature	Description
Cut	Grayed out and disabled	/
Copy	Enabled	Copies the selected variable.
Paste	Grayed out and disabled	/
Delete	Grayed out and disabled	/
Cross-reference	Enabled	Opens the cross-reference table for the selected variable.
Adding a Recipe	Enabled	Adds a recipe list.

- Recipe list
  The right-click menu functions for the recipe list are shown in the figure below.



The specific functions of the right-click menu are described in the table below.

Description	Enabled Feature	Description
Cut	Grayed out and disabled	/
Copy	Enabled	Copies the selected variable.
Paste	Grayed out and disabled	/
Delete	Grayed out and disabled	/
Cross-reference	Enabled	Opens the cross-reference table for the selected variable.
Adding a Recipe	Enabled	Adds a recipe.
Delete recipe	Enabled	Deletes an existing recipe. Only one recipe can be deleted at a time. Batch selection is not supported.
Load recipe	Enabled	Loads from a local file, reads the parameters of the saved recipe, and writes them into the selected recipe.
Save recipe	Enabled	Saves the parameters of the selected recipe to a local file (which can only be imported after successful compilation).
Read from PLC	Enabled	Reads the current values and writes them into the currently selected recipe.
Read from PLC and save recipe	Enabled	Reads the current values and writes them into the currently selected recipe, then saves the parameters of the selected recipe to a local file.
Write to PLC	Enabled	Writes the values of the currently selected recipe to the current values.
Load recipe and write to PLC	Enabled	Loads from a local file, reads the parameters of the saved recipe, writes them into the selected recipe, and writes the values of the currently selected recipe to the current values.

10.4 Watch Table

10.4.1 Monitoring Variable Table

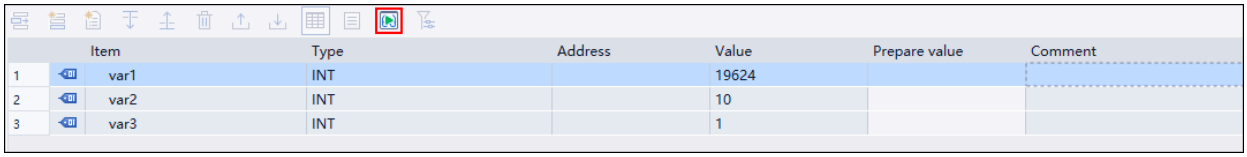
When the device is in online mode, the POU variable table and the global variable table GVL support real-time monitoring.

Prerequisites

- The variable has been added to the variable table.
- The current POU has been called.
- The device is in an online status.

Procedure

1. Switch the device to online mode.
2. Click the monitoring icon in the toolbar of the variable table to enable the independent monitoring function of the variable table, putting the variable table into online status.

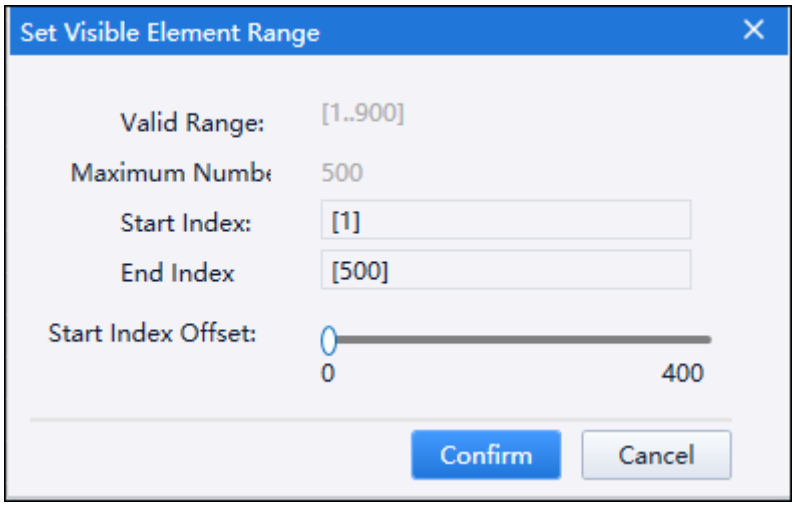


Columns in the online variable table

Column	Description
Name	The variable name
Type	The data type of the variable
Address	The address of the variable
Value	The current real-time value of the variable
Prepared value	The value prepared for writing to the variable • Basic type prepared value: Enter the prepared value by editing the cell. • BOOL/enumeration type prepared value: Select the prepared value from a drop-down list.
Comment	The comment of the variable

Visible element range setting

After the variable table goes online, the maximum number of variables displayed is 500. When this number is exceeded, the "  " icon will be enabled to set the visible element range. Users can set the variable display range through this icon, as shown in the following figure.



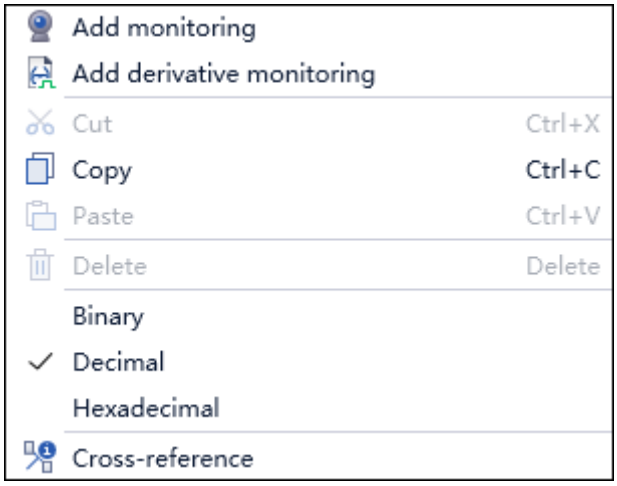
NOTE

Since iFA Evolution V1.4.4.5, the viewing function of the variable table has changed:

- For versions prior to V1.4.4.5.Patch.20260610: The maximum number of viewable variables is limited to 500.
- For versions V1.4.4.5.Patch.20260610 and later: Users can view all variables with no limit.

Variable table right-click menu

Right-click a variable to open the right-click menu, which is shown in the figure below.



Function	Description
Add watch	Adds the variable to the watch table.
Add derivative monitoring	Adds the variable to the derivative watch table. Note: This operation is available when a derivative watch table already exists and the number of monitored variables is less than 16.
Cut/Paste/Delete	Grayed out and disabled in online status.
Copy	Copies variable to clipboard.
Binary	Displays the value and prepared value of the variable in binary format.
Decimal	Displays the value and prepared value of the variable in decimal format.
Hexadecimal	Displays the value and prepared value of the variable in hexadecimal format.
Cross-reference	Opens the cross-reference table for the variable.

NOTE

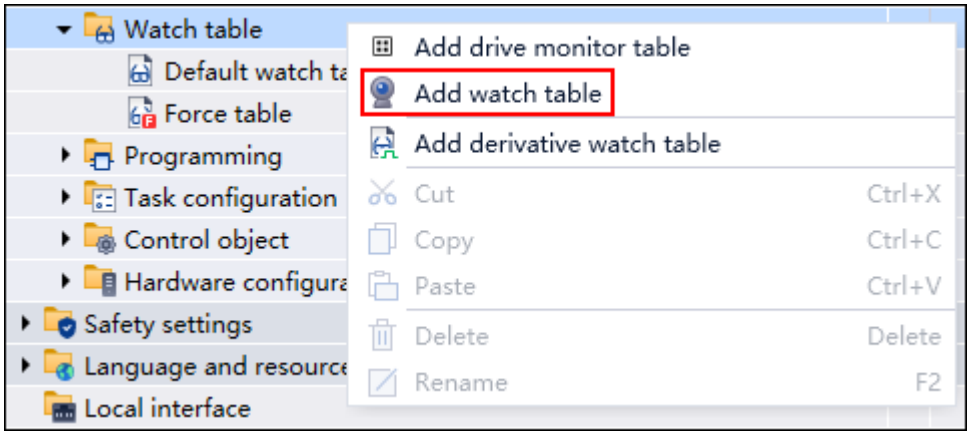
In the same POU/GVL, variables can be displayed in binary, decimal, and binary formats.

10.4.2 Monitoring Specified Variables

When the device is in online mode, the watch table allows real-time monitoring of specified variables and addresses. Users can view the data type, current value, and assign a value to the monitored variable through the watch table.

Procedure

1. Right-click "Watch table" in the project tree and select "Add watch table". A maximum of 20 watch tables can be created.



2. Add variables to the watch table.
   a. Click  or  to add or insert a variable row.
   b. Variables can be added to the watch table in both online and offline states with the following methods:
      - Add from variable list: Double-click the text box to enter the variable name, or click "..." to select a variable using the input assistant.
      - Add from variable declaration area (online only): Right-click the variable and select "Add monitoring", or drag and drop the variable directly onto the watch table.
      - Add from program editing area (online only): Right-click the variable and select "Add monitoring".
3. Switch the device to online mode and variable monitoring will be enabled automatically. Click  to switch the variable's disabled or enabled state, with the default being enabled.

Result

- The watch table displays the variable name/address, type, value, prepared value, and comment.

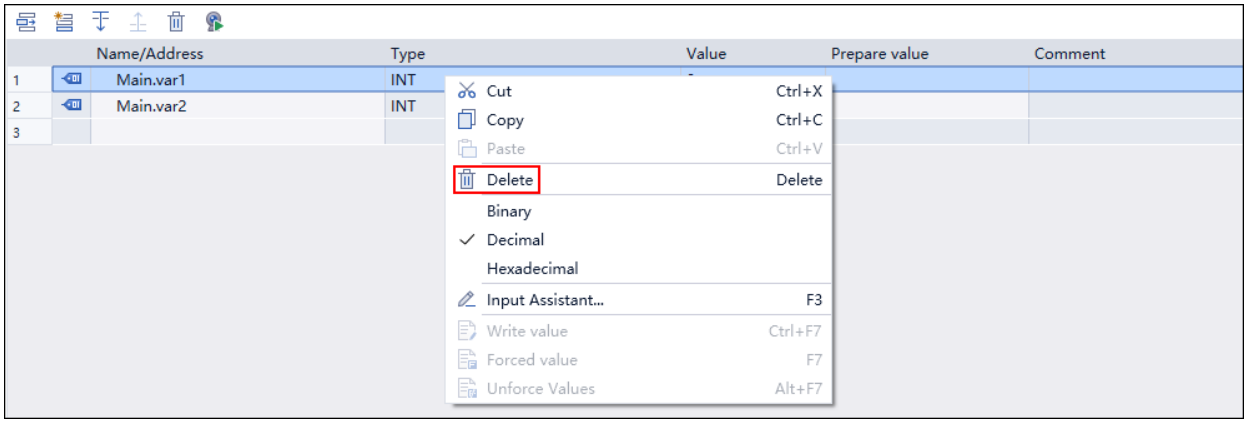
	Name/Address	Type	Value	Prepared value	Comment
1	Main.var1	INT	11180		
2	Main.var2	INT	11180		
3					

- When the device is offline or monitoring is disabled, variables cannot be monitored, and their values are displayed as <No monitoring>.

	Name/Address	Type	Value	Prepared value	Comment
1	Main.var1	INT	<No monitoring>		
2	Main.var2	INT	<No monitoring>		
3					

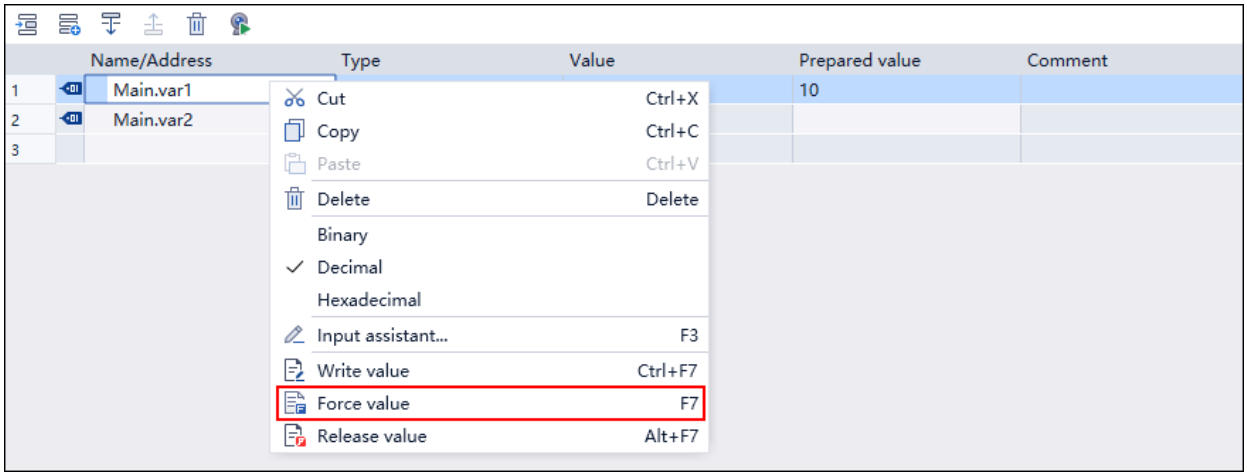
Deleting a variable

Right-click a variable and select "Delete" to delete it from the watch table.

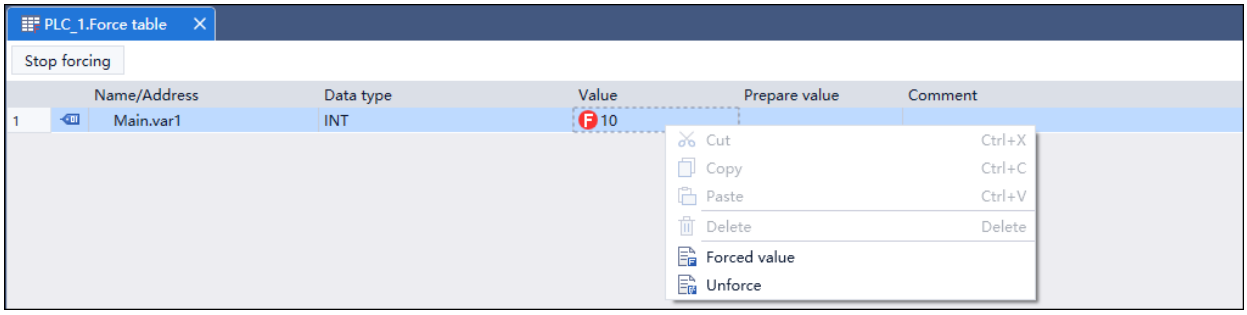


Variable display

Right-click a monitored variable and select "Force value" to add a forced value. The variable will then be displayed by default in the force table under the "Watch table" node in the project tree.



Cancel forcing: Cancel the previously written forced value.



10.4.3 Drive Watch Table

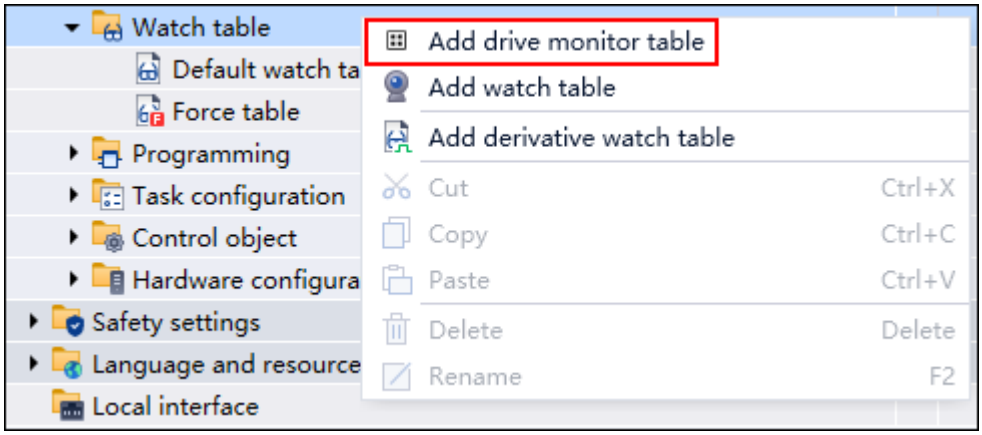
The drive watch table is used to monitor the real-time values of axis parameters during operation of servo slaves, with the monitored axis parameter objects specified by users during configuration.

Prerequisites

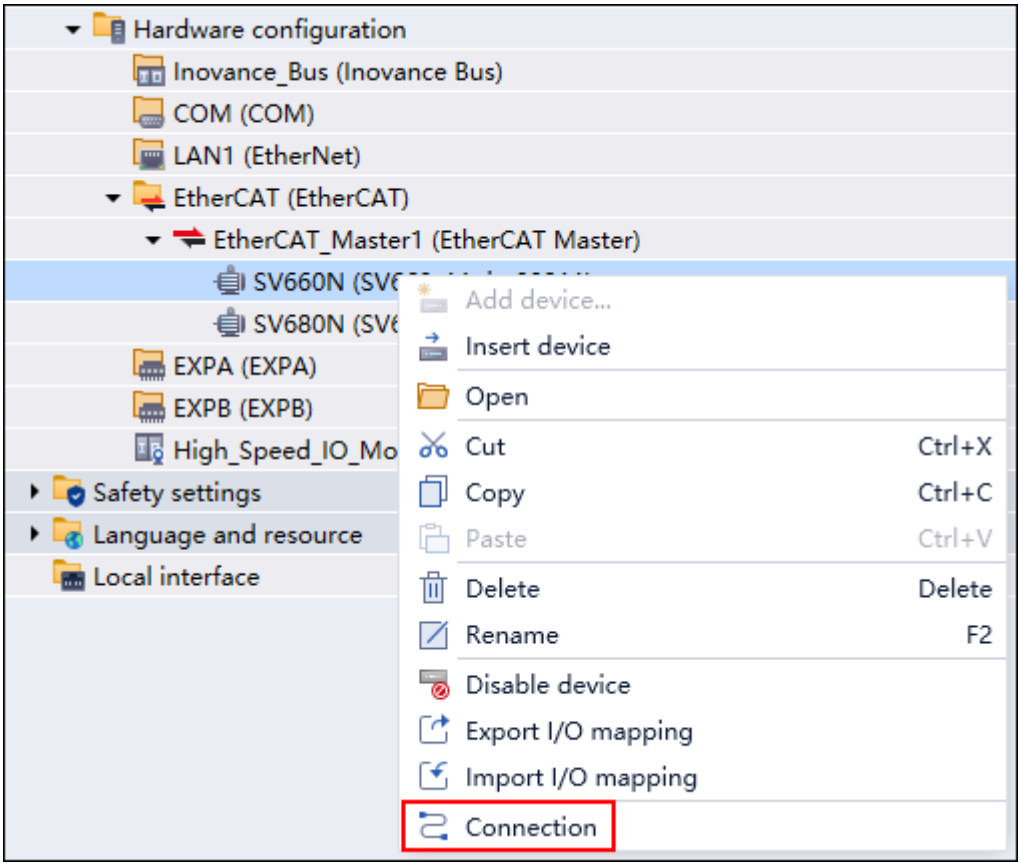
- Supported servo device models are SV660 and SV680.
- The drive watch table supports up to 200 parameters, with no more than 16 servo slaves distributed across them.

Procedure

1. Right-click "Watch table" in the project tree and select "Add drive watch table".

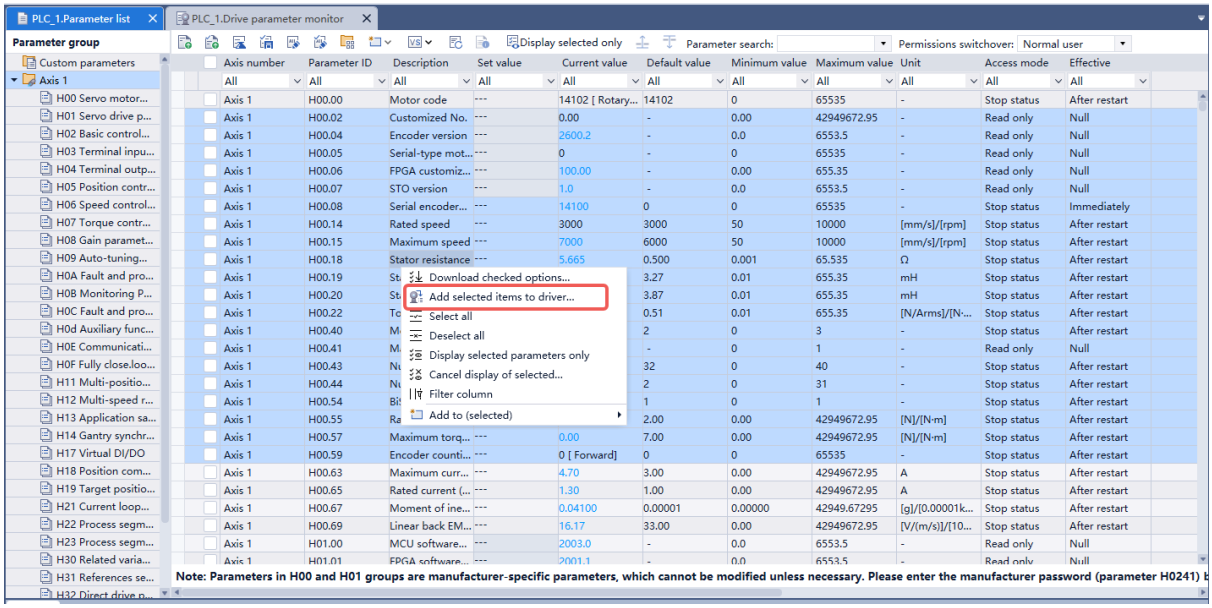


2. Right-click a servo slave and select "Connect" to complete the communication connection between the slave and the PLC. For detailed connection methods, see 17.1.1 EOE Connection.

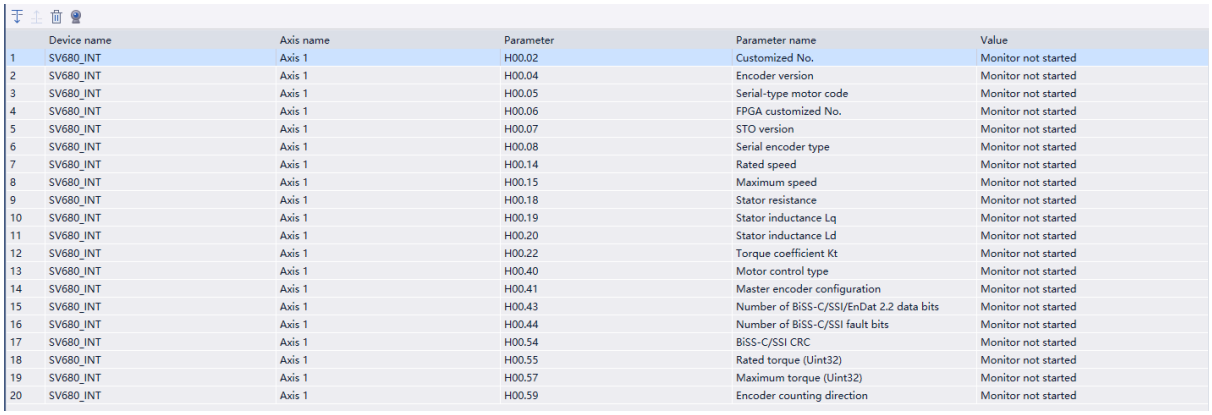


3. Double-click the "Parameter list" of the slave device to open the configuration interface. Right-click the target axis parameters to monitor (supporting multiple selections) and select "Add the selected to the

drive monitoring table (current page)" to complete the parameter configuration in the drive watch table.



4. Double-click the added drive watch table in the project tree to open the corresponding interface.



	Device name	Axis name	Parameter	Parameter name	Value
1	SV680_INT	Axis 1	H00.02	Customized No.	Monitor not started
2	SV680_INT	Axis 1	H00.04	Encoder version	Monitor not started
3	SV680_INT	Axis 1	H00.05	Serial-type motor code	Monitor not started
4	SV680_INT	Axis 1	H00.06	FPGA customized No.	Monitor not started
5	SV680_INT	Axis 1	H00.07	STO version	Monitor not started
6	SV680_INT	Axis 1	H00.08	Serial encoder type	Monitor not started
7	SV680_INT	Axis 1	H00.14	Rated speed	Monitor not started
8	SV680_INT	Axis 1	H00.15	Maximum speed	Monitor not started
9	SV680_INT	Axis 1	H00.18	Stator resistance	Monitor not started
10	SV680_INT	Axis 1	H00.19	Stator inductance Lq	Monitor not started
11	SV680_INT	Axis 1	H00.20	Stator inductance Ld	Monitor not started
12	SV680_INT	Axis 1	H00.22	Torque coefficient Kt	Monitor not started
13	SV680_INT	Axis 1	H00.40	Motor control type	Monitor not started
14	SV680_INT	Axis 1	H00.41	Master encoder configuration	Monitor not started
15	SV680_INT	Axis 1	H00.43	Number of BiSS-C/SSI/EnDat 2.2 data bits	Monitor not started
16	SV680_INT	Axis 1	H00.44	Number of BiSS-C/SSI fault bits	Monitor not started
17	SV680_INT	Axis 1	H00.54	BiSS-C/SSI CRC	Monitor not started
18	SV680_INT	Axis 1	H00.55	Rated torque (Uint32)	Monitor not started
19	SV680_INT	Axis 1	H00.57	Maximum torque (Uint32)	Monitor not started
20	SV680_INT	Axis 1	H00.59	Encoder counting direction	Monitor not started

The icons and parameters in the interface are described in the following table.

Name	Description
	Moves the selected row(s) down, supporting simultaneous movement of multiple rows.
	Moves the selected row(s) up, supporting simultaneous movement of multiple rows.
	Deletes the selected row.
	Monitoring switch: When enabled, the PLC reads and displays the real-time values of the servo slave's function codes; when disabled, the PLC does not read real-time values, and displays "-" instead.
Device name	The name of the servo slave in the project; if the name changes, it will automatically synchronize here.
Axis name	The axis number configured internally in the servo, independent of the motion control axis name.
Parameter	Parameter format: H00.00.
Parameter name	Description of the parameter.

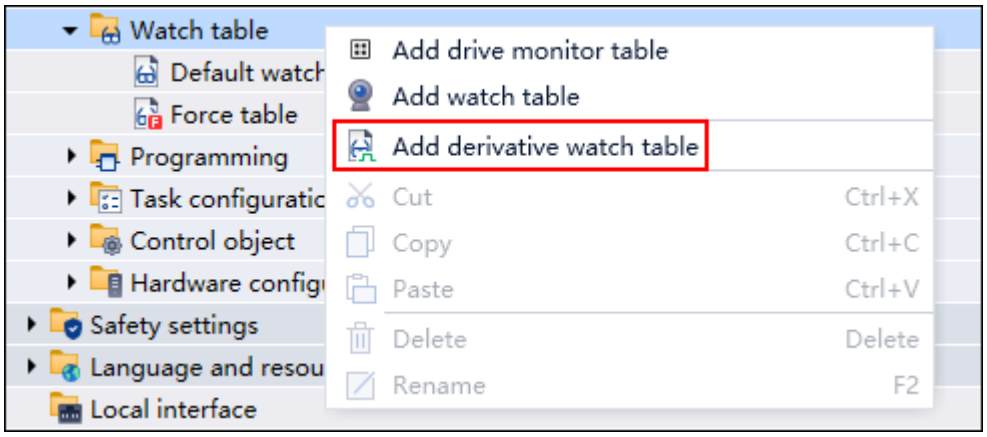
Name	Description
Value	The real-time value of the axis parameter can be monitored here, but not written. The value is refreshed every 1 second. Displays the real-time value when the slave communicates normally with the PLC; displays "-" if communication is abnormal.

10.4.4 Derivative Watch Table

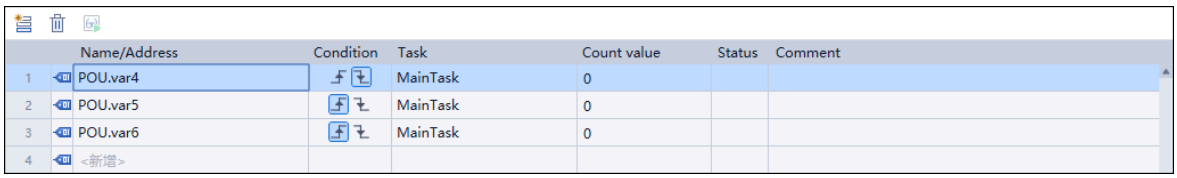
Derivative watch tables are used to monitor the number of changes in up to 16 BOOL/BIT variables within a task cycle.

Procedure

1. Right-click "Watch table" in the project tree and select "Add derivative watch table".



2. Add derivative watch variables.
   a. Click or double-click the derivative variable list text box <新增> to add a derivative watch variable row.
   b. Variables can be added to the derivative watch table in both online and offline states with the following methods:
      - Add from variable list: Double-click the text box to enter the variable name, or click "..." to select a variable using the input assistant.
      - Add from variable declaration area (online only): Right-click the variable and select "Add derivative monitoring", or drag and drop the variable directly onto the derivative watch table.
      - Add from program editing area (online only): Right-click the variable and select "Add derivative monitoring".
   c. Click or in the "Condition" column of the variable row to set the monitoring condition.
   d. Click the "Task" cell of the variable row to select the task to be mounted.



The icons and parameters in the interface are described in the following table.

Name	Description
	Adds a watch variable row.
	Deletes the selected row.

Name	Description
/	Enables or disables the derivative monitor.
Name/Address	Name or address of the monitored variable.
Condition	Conditions for monitoring variables. Click the following icons to select the conditions. Default is rising edge triggered. : Rising edge : Falling edge
Task	Task mounted to the monitored variable. Click to select the task. Default is MainTask.
Count value	Counts how many times the set condition is triggered. Within the task cycle of the monitored variable, the current value is compared with the previous value. A count is incremented when the value changes to match the specified condition. ▪ Only one count is recorded per task cycle even if the value matches the condition multiple times. ▪ No count is recorded if the value at the previous and current evaluation are the same (even if it changed in between).
Status	Monitors the status of the variable. ▪ Condition not triggered: Blank. ▪ Condition triggered: Green/gray icon flashes alternatively for 8s per trigger.
Comment	Comments on the monitored variables.

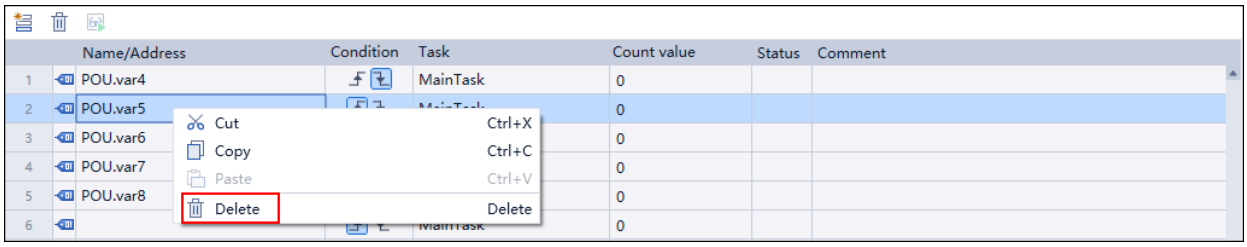
3. Switch the device to online mode and click to enable the derivative monitor.

NOTE

Derivative monitoring variables can only be added when a derivative monitoring table already exists and the number of monitored variables is less than 16.

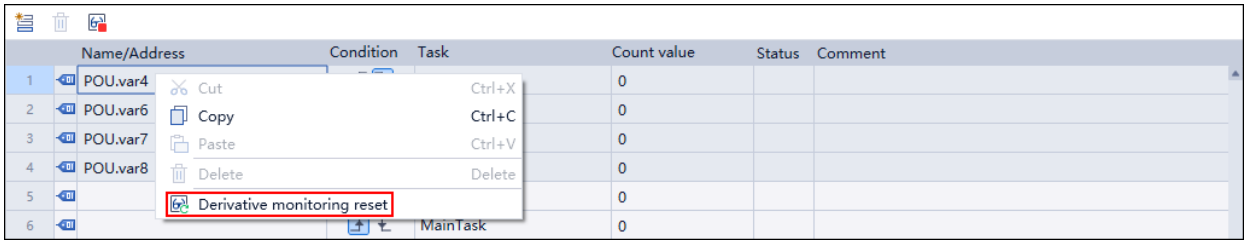
Deleting a variable

Right-click a variable and select "Delete" to delete it from the derivative watch table. The monitored variable cannot be deleted when the derivative monitor is enabled.



Derivative watch reset

When the device is in online state and the derivative monitor is enabled, right-click the variable and select "Derivative monitoring reset" to resume monitoring (restart counting) for that variable.



10.5 Task Configuration

10.5.1 Adding a Task

A task is the fundamental execution units of the scheduler, and all programs must be assigned to a task to run. Tasks can be managed through the "Task configuration" node in the project tree. The system automatically generates a default task when a PLC device is added; users can also create new tasks as needed. By setting task properties (such as execution cycle, priority, and bound CPU core), users can control the allocation of system resources and program execution.

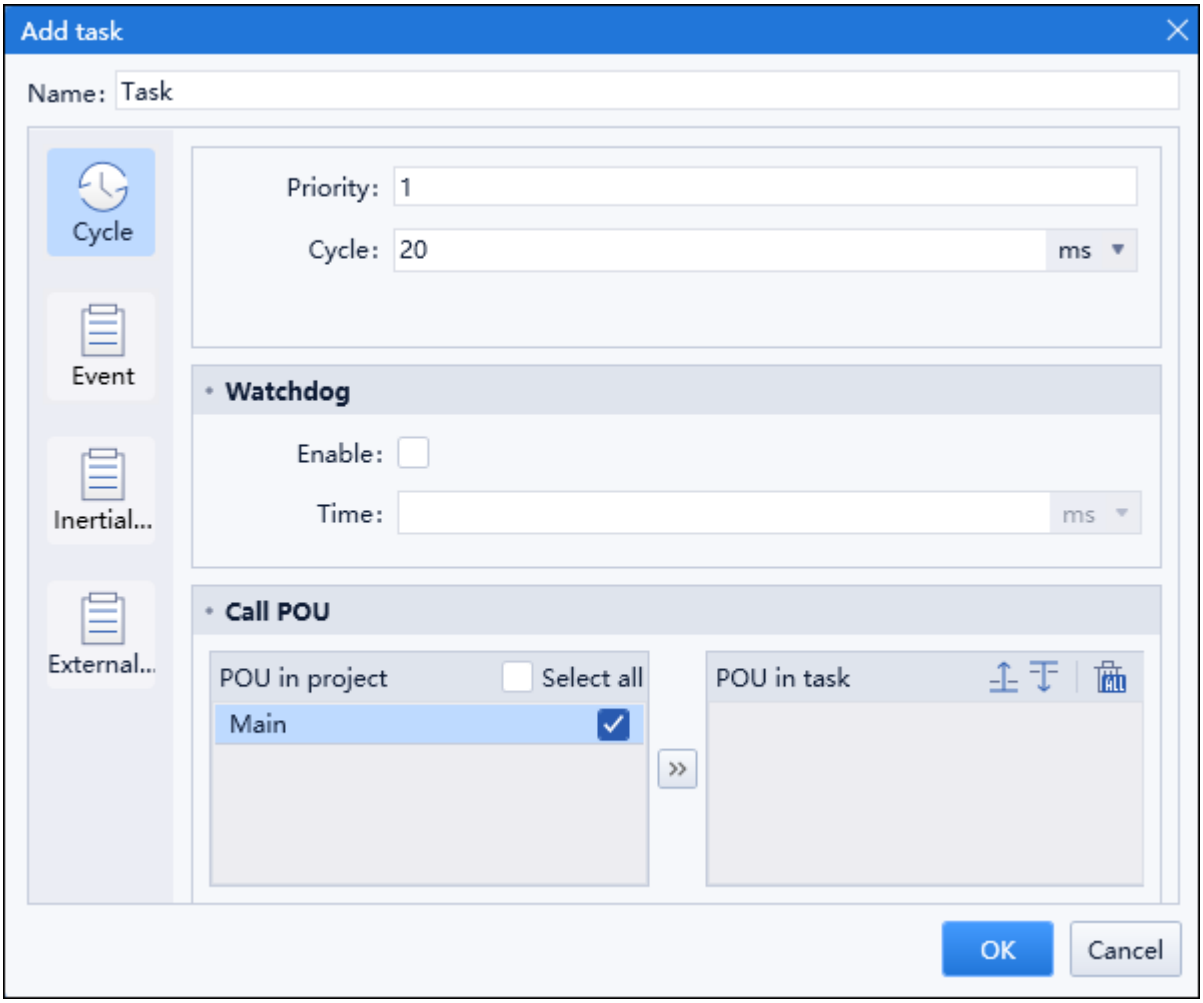
Prerequisites

- A PLC device has been added.
- A program has been created.
- Before creating a task of the "Event" type, first create a BOOL variable for event trigger.

Procedure

A task supports only one execution type, interval, and priority. Add multiple tasks for different execution characteristics.

1. Right-click the "Task configuration" node in the project tree and select "Add task".
2. In the pop-up window, configure the task name, task type, and related parameters. For details, see 10.5.2 Task Parameter Configuration.



3. Click "OK" to add the task.
4. (Optional) Open the added task in the project tree to edit its parameters. There are two methods to open a task:
   - Method 1: Double-click the task node in the project tree.
   - Method 2: Right-click the task node in the project tree and select "Open".

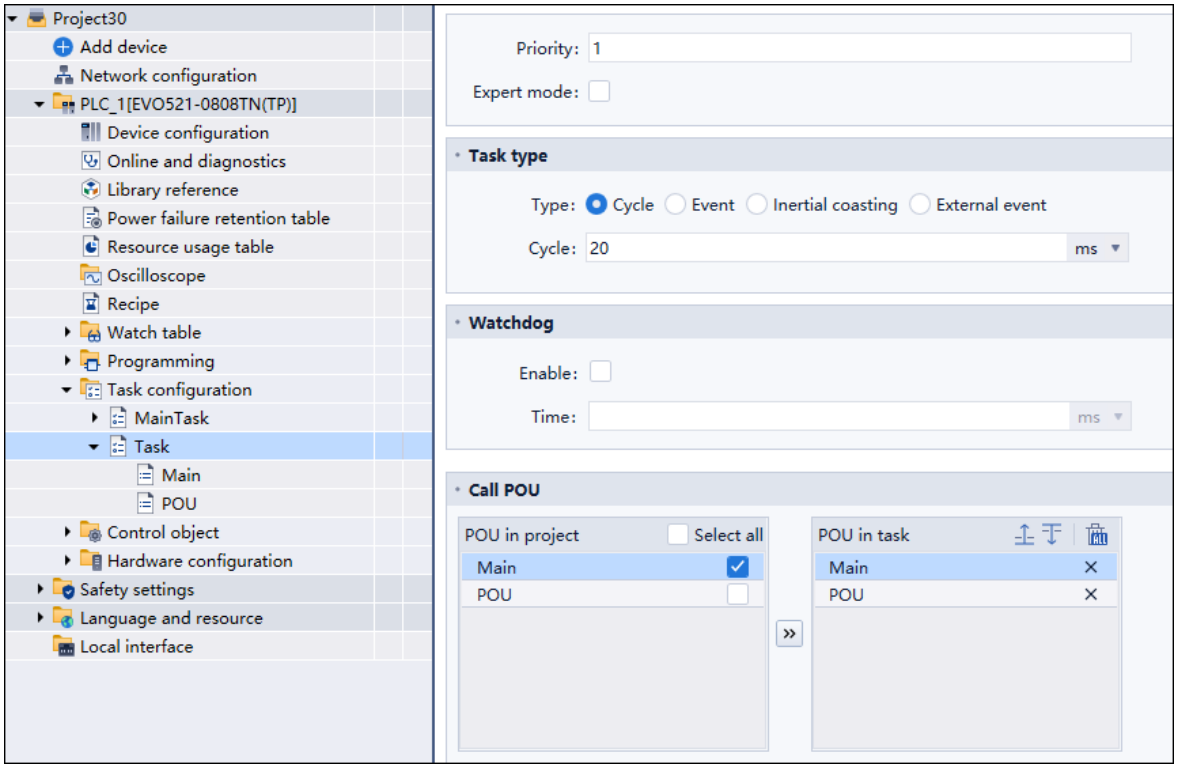


Figure 10-5-1-1 Task configuration interface

10.5.2 Configuring Task Parameters

- **Precedence**

  Priority is the basis for PLC task scheduling, determining the order in which tasks are executed. The execution of a low-priority task may be interrupted by a high-priority task. After the high-priority task has been executed, the low-priority task interrupted previously will be resumed.

  Task priority can be set in the "Add task" or task configuration window. It ranges from 0 to 31 (range for EVO800 series redundant mode is 0 to 15). The smaller the number, the higher the priority. Levels 0 to 15 adopt a priority-based preemptive scheduling strategy, while levels 16 to 31 adopt a time-slice-based round-robin scheduling strategy.

  The system pre-sets different priorities for automatically added tasks (e.g., EtherCAT bus task, CANOPEN interrupt task) to ensure that critical tasks such as motion control are executed first. In applications requiring high-performance motion control (MC), this enables efficient utilization of controllers. The task priority order is as follows (modifying this order is not recommended):

Default Priority	Task name	Description
0	EtherCAT_Master_Task	Only one EtherCAT task is allowed.
2	CANopen	Only one CANopen task is allowed.

- **Expert mode**

  In expert mode, multi-core PLCs can process multiple tasks simultaneously. For multi-core PLC devices, the core for task execution can be configured to assign different tasks to different cores for processing.

  1. Double-click a task node in the project tree to open the task.
  2. In the task configuration interface, select "Expert mode", and select the core for task execution (0 or 1) in the "Assigned core" drop-down menu.

NOTE

- Dynamically allocate cores for task execution based on core utilization rates. To view core utilization, double-click "Online and diagnostics" in the project tree and select "Task monitoring". The core utilization can be viewed in the "CPU and memory monitoring" section.
- Multiple tasks can be assigned to the same core, and the processing order follows the task priority.
- A single task cannot be assigned to multiple cores.
- The EVO800 series supports activating expert mode in single mode, but not in redundant mode.

- **Task type**

1. The task type can be selected in the "Add task" or task configuration interface.

Task type	Note	Execution Characteristics
Periodic	Cycle tasks repeat execution periodically according to the set cycle time.	The POU is executed once at each configured interval.
Event	Event tasks start executing when a specific event is triggered. In iFA software task configuration, set a BOOL variable as the event trigger. When the BOOL type variable is TRUE, the event is triggered, and the task starts executing.	Task execution is triggered once when the preset BOOL variable transitions from 0 to 1, but will not be triggered under other transition combinations such as 0->0, 1->1, and 1->0.
Inertial slide	An inertial coasting task refers to automatically restarting the task in a continuous loop at the beginning and end of task execution.	The task is executed circularly and continuously once the execution is started.
External event	External event tasks are suitable for situations where immediate program response is required when an external signal is input or when the counter axis reaches a set value.	The corresponding POU is executed once when the HSIO status changes or the high-speed counter readings match. External event tasks are not affected by the scan cycle.

NOTE

- The EVO800 series only supports two task types in redundant mode: cycle type and inertial coasting type.
- Only EVO400, EVO500, and APC500 series PLCs support configuring external event tasks.

2. Depending on the selected task type, configure the corresponding parameters. The parameter descriptions for each task type are shown in the table below.

Parameter	Description
Periodic	This parameter is displayed when the task type is "Cycle". Indicates the task execution interval (unit: ms/μs). For slow-changing processes with low real-time requirements (such as conventional logic control), the cycle can be appropriately extended (e.g., to 20 ms); for processes with high real-time requirements (such as interrupt service), a shorter cycle must be configured to ensure system real-time performance. NOTE The minimum task cycle varies for different products. For example, the minimum task cycle for the EVO500 series is 1 ms. For details, see the relevant product manuals.
Offset	This parameter is displayed when the task type is "Cycle". The system uses the highest-priority task as the reference task, and its execution time serves as the time reference point. Other tasks (subordinate tasks) are executed with a set offset delay relative to the execution time of the reference task. Default is 0. The maximum offset can be set to 2147483647, in ms or μs. For example: ▪ If the highest-priority reference task has a cycle of 100 ms, it is executed at 0 ms, 100 ms, 200 ms, etc. ▪ If the subordinate task A has a cycle of 100 ms and has no offset set, it is triggered at 0 ms, 100 ms, 200 ms, etc., simultaneously with the reference task. By setting an offset of 50 ms for the subordinate task A, it is triggered at 50 ms, 150 ms, 250 ms, etc. This avoids the CPU pressure caused by multiple concurrent tasks and ensures real-time response of high-priority tasks. NOTE ▪ Only EVO800 series PLCs (including standalone, redundant, and multi-machine modes) support offset configuration. ▪ To ensure the performance of the highest-priority task, it is recommended to set offsets only when the subordinate task cycle is an integer multiple of the reference task cycle. ▪ Due to various factors such as fluctuations in program logic complexity, system resource competition, or external interference, the execution cycle of tasks may fluctuate within a certain range (task jitter time ±75 μs). Therefore, users need to dynamically adjust the task offset based on the actual runtime results.

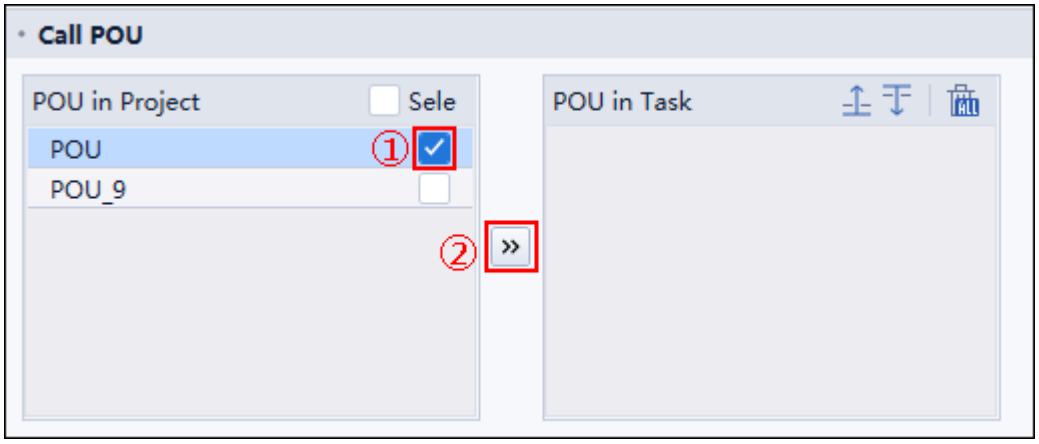
Parameter	Description
Event	This parameter is displayed when the task type is "Event". Click the "..." in the edit box, and select a BOOL-type variable in the "Input assistant" pop-up window.
External event	This parameter is displayed when the task type is "External event". Select the external event for task triggering from the drop-down menu. ▪ X0InterruptEvent to X3InterruptEvent are used to receive interrupt events triggered by input ports X0 to X3. ▪ Cmp1InterruptEvent to Cmp16InterruptEvent are used to receive comparison interrupt events of high-speed counter 0 to 3. High-speed counters 0 to 3 can be arbitrarily bound to external events 1 to 16. NOTE When adding multiple external event tasks, external events must not be duplicated; otherwise, a compilation fault will occur.

- **Watchdog**
  The watchdog monitors the actual runtime of the task. If the actual runtime exceeds the limit set by the watchdog, the watchdog mechanism will be triggered, terminating the task execution.
  1. In the "Add task" or task configuration interface page, check "Enable" in the "Watchdog" section.
  2. Set the watchdog time and unit (ms or us).

NOTE

The EVO800 series (single/redundant mode) has watchdog enabled by default, with a default time of twice the task execution cycle. Users can disable the watchdog or reconfigure the time as needed; the configured value should not be less than twice the task cycle.

- **Call POU**
  A task can include multiple POUs. These POUs follow the same execution cycle and are executed in the order they were added to the task.
  1. In the "Add task" or task configuration interface, select the POUs from the "POU in project" list.
  2. Click » to add the selected POU to the current task.



The POUs in the current task can be moved up/down or deleted, as shown in the table below.

Description	Icon	Description
Move up		Move the selected POU upward; this operation synchronously updates the POU order under the corresponding task node in the project tree.
Move down		Move the selected POU downward; this operation synchronously updates the POU order under the corresponding task node in the project tree.
Delete all		Delete all POUs from the "POU in project" list; this operation synchronously removes the corresponding POUs from the task node in the project tree.

10.5.3 System Event Management

To perform initialization before the PLC starts running the main program, you can add a "Pre-start preparation" event on the "System event management" page and add the initialization program to this event. After the event is activated, the initialization operation will be executed before the formal task begins. The execution rules for the pre-start preparation event are as follows:

1. When the PLC is run for the first time, it is automatically called before all tasks are executed.
2. When the PLC is restarted after stopping, it is regarded as a new operation cycle and is executed with priority again.
3. If users log out and in again after modifying the program (triggering online modification), it will not be executed.
4. If users log out and in again without modifying the program (not triggering online modification), it will not be executed.
5. If online modification is triggered but a full download is executed, it will be executed once.
6. After the device is powered off and on again, it will be executed once.

NOTE

In iFA simulation mode, the rules are consistent with the above 1 to 5.

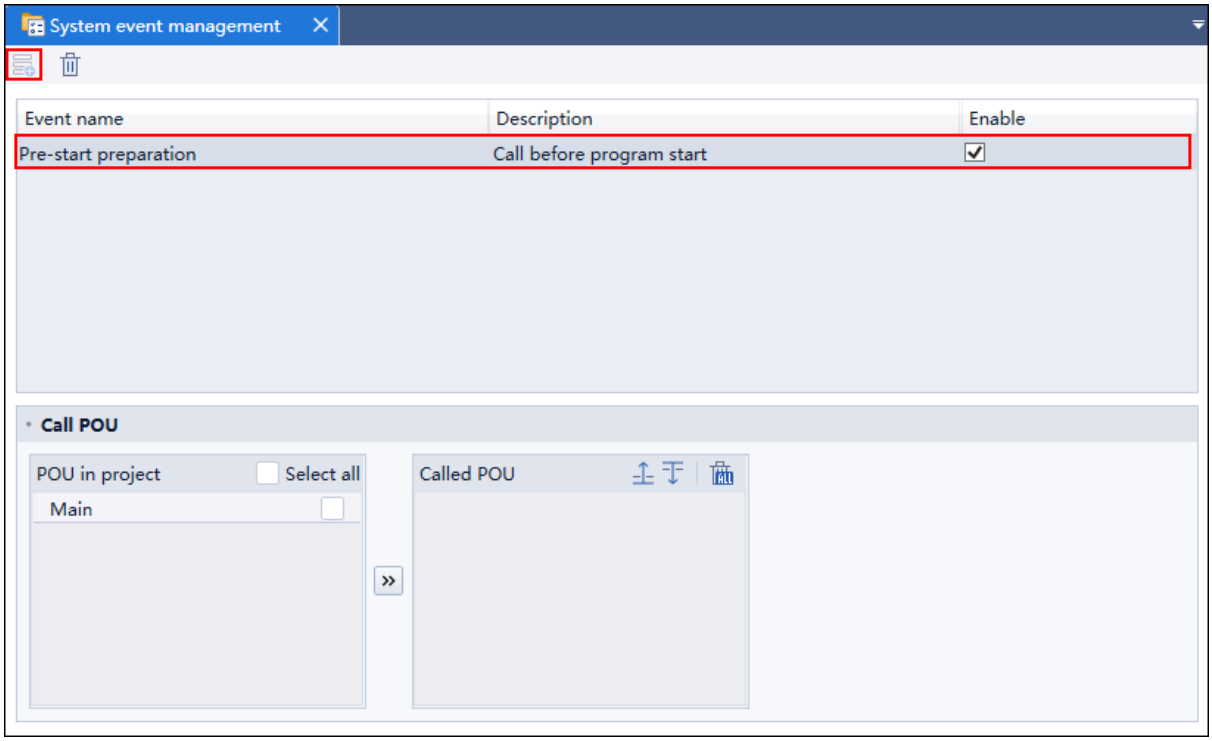
Prerequisites

- A PLC device has been added.
- The initialization program has been created.

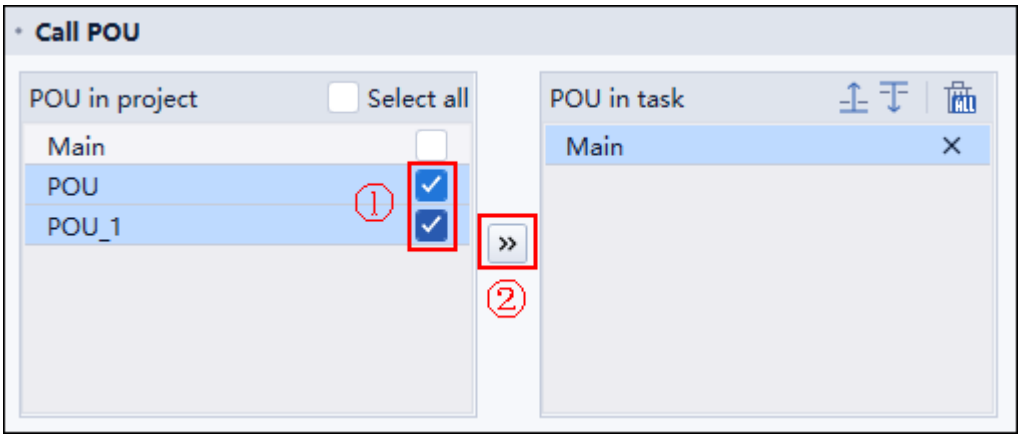
Procedure

1. Right-click the "Task configuration" node in the project tree and select "System event management".
2. On the "System event management" page, click to add the "Pre-start preparation" event. This function is activated by default and can be deactivated as needed.

   Only one "Pre-start preparation" event is supported. Click to delete the event.



3. In the "Call POU" area, check the POU(s) in the "POU in project" list (select all is supported) and click to add the selected POU(s) to the "Pre-start preparation" event.



The "Called POU" can be moved up, moved down, or deleted. The specific instructions are shown in the table below.

Description	Icon	Description
Move up		Moves up the selected POU.
Move down		Moves down the selected POU.
Delete all		Deletes all POUs in the list.

10.6 PLC Configuration

10.6.1 Hardware Configuration and Device Management

10.6.1.1 Device Scanning

When the current PLC needs to connect to devices such as PLCs, HMIs, PACs, PCs, and drives, this function can scan all compatible devices within the same subnet, allowing users to select the target device for connection.

Background

The EVO800 series PLC supports downloading and commissioning through the Ethernet expansion module port. When scanning devices through this port, the current system only supports scanning the PLC to which the connected Ethernet expansion module belongs; other PLCs are not supported for scanning.

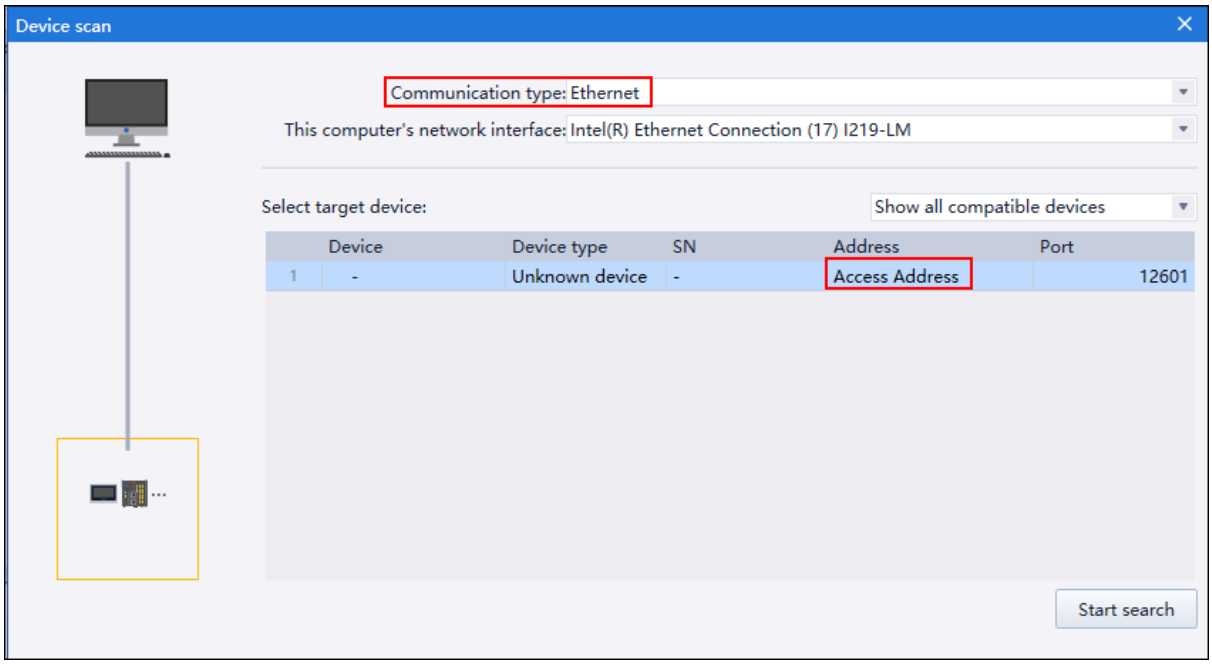
Prerequisites

Devices have been configured in the project.

Steps

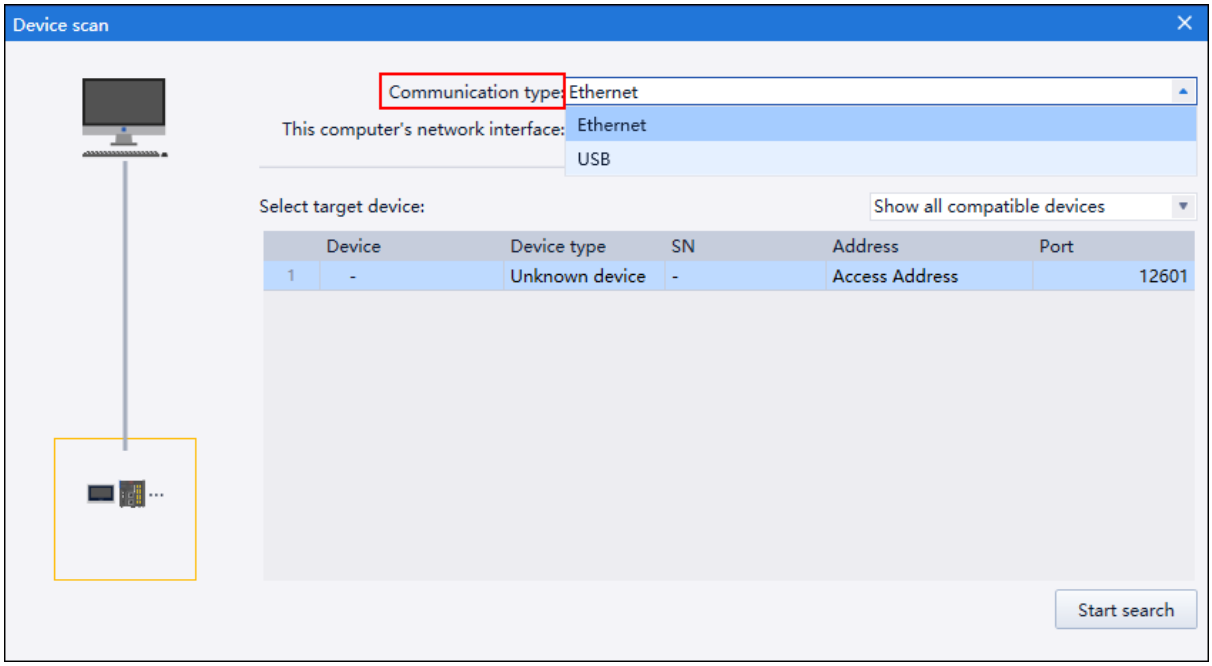
1. Select a device from the left project tree.
2. Scan the selected device.
   - Method 1: Right-click the selected device and select "Online" to open the "Device scanning" window.
   - Method 2: Click the "Device scanning" icon in the toolbar.
3. Set search conditions.
   - Searching by address

     Select communication type as "Ethernet". Double-click "Access address" and enter the IP address.



   - Searching by communication type

     Select communication type as "Ethernet" or "USB."



USB interface communication constraints:

> CAUTION
>
> For non-administrator users, the USB scanning function is not supported. Switch to administrator or scan the device via Ethernet.

- Supports standard USB 2.0 interface.
- Supports direct one-to-one connection between USB interface and PLC.
- During routine function (such as download and commissioning) communication, multiple USB interfaces on the PC can be used simultaneously (to connect to PLCs). During PLC firmware upgrade, only one of the PC's USB interface is supported, with the rest remaining unused.

4. Execute search.
   Click "Start search" to perform the scan, and select the target device from the results.

   To avoid incorrect connections, ensure that the selected device in the list matches the physical target device. There are two methods to confirm:

   - SN number consistency check: Compare the SN number in the window with that on the physical device, ensuring they match.
   - LED indicator consistency check: When "Flashing LED" is checked in the window, the RUN indicator on the corresponding physical device will flash in response.

5. Connect the device.
   After confirming the information, click "OK" to connect to the physical device.

Scanning operations

Cross-subnet device search and renaming

To configure network settings for devices within a region, use the 7.3.2 network configuration functions to search for all devices in the region and modify device names, IP addresses, and so on.

Filtering devices by network card

For multiple network interfaces on the user's PC, devices can be filtered based on different network interfaces.

- Prerequisites

There are multiple network interfaces on the user's PC.

- Procedure

1. There are two ways to scan devices:
   - Method 1: Right-click the selected device and select "Online" to open the "Device scanning" window.
   - Method 2: Click the "Device scanning" icon in the toolbar.
2. In the "Device scanning" window, click "Start search" and filter the scanned devices based on different network card types from the "PC network interface" drop-down list.

Device memory connection

To quickly connect to a previously connected device, the software supports device memory connection.

- Prerequisites
  - Devices have been configured in the project.
  - The device has been successfully connected in the current project before.
- Procedure

Click the drop-down box next to the "Device scanning" icon in the toolbar to view and select previously connected devices.

10.6.1.2 Viewing Device ID

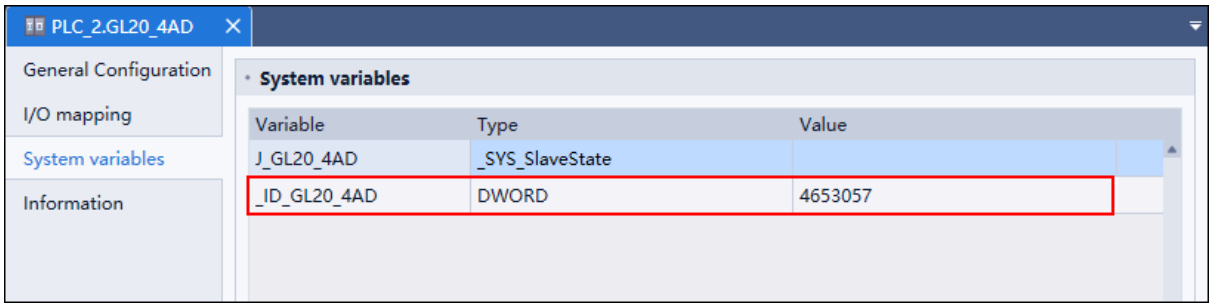
Device ID is a number used to uniquely identify different devices. It is an essential component for communication, configuration, and maintenance among devices.

Prerequisites

Devices have been configured in the project.

Procedure

1. In the left project tree, double-click a device node to open the device configuration interface (using the I/O module GL20-4AD as an example).
2. Select the "System variable" tab to view the device ID (_ID_Device Name).



10.6.1.3 Exporting I/O Mapping

The I/O mapping table can be exported in CSV format. The export scope is divided into:

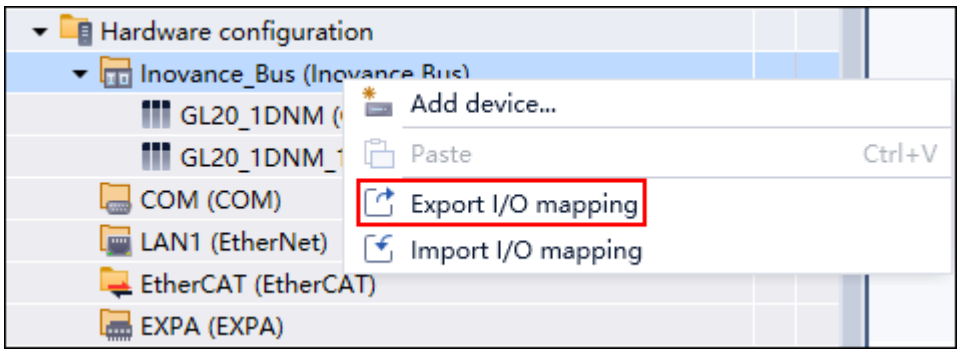
- Single device: Export all I/O mappings of a single PLC device.

- Hardware configuration node: Export all I/O mappings under the PLC hardware configuration node.
- Single port node: Export I/O mappings of Inovance_Bus, COM, LAN, EtherCAT, expansion cards, and so on.

Users can edit the exported file according to the format and import the updated configuration. This section introduces the export of I/O mapping for Inovance_Bus (I/O module) as an example.

Procedure

1. Right-click "Inovance_Bus (I/O module)" under the "Hardware configuration" node in the project tree, and select "Export I/O mapping".



2. Select the file storage path, edit the filename as needed, and click "Save".

Result

The exported file "XXX.csv" is shown in the figure below.

Variable	Variable table	Channel	Device name	Channel address following variable address	Description
		MasterStat	GL20_1DNM	No	
		Bit0	GL20_1DNM	No	
		Bit1	GL20_1DNM	No	
		Bit2	GL20_1DNM	No	
		Bit3	GL20_1DNM	No	
		Bit4	GL20_1DNM	No	
		Bit5	GL20_1DNM	No	
		Bit6	GL20_1DNM	No	
		Bit7	GL20_1DNM	No	
		SlavesState	GL20_1DNM	No	
		SlavesState	GL20_1DNM	No	
		Bit0	GL20_1DNM	No	
		Bit1	GL20_1DNM	No	
		Bit2	GL20_1DNM	No	
		Bit3	GL20_1DNM	No	
		Bit4	GL20_1DNM	No	
		Bit5	GL20_1DNM	No	
		Bit6	GL20_1DNM	No	
		Bit7	GL20_1DNM	No	
		SlavesState	GL20_1DNM	No	
		Bit0	GL20_1DNM	No	
		Bit1	GL20_1DNM	No	
		Bit2	GL20_1DNM	No	
		Bit3	GL20_1DNM	No	

10.6.1.4 Importing I/O Mapping

Importing I/O mapping allows for quick updating of I/O configuration. The import scope is divided into:

- Single device: Import all I/O mappings of a single PLC device.
- Hardware configuration node: Import all I/O mappings under the PLC hardware configuration node.

- Single port node: Import I/O mappings of Inovance_Bus, COM, LAN, EtherCAT, expansion cards, and so on.

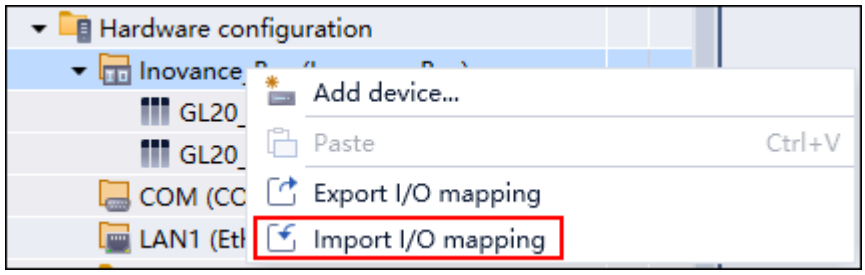
This section introduces the import of I/O mapping for Inovance_Bus (I/O module) as an example.

Prerequisites

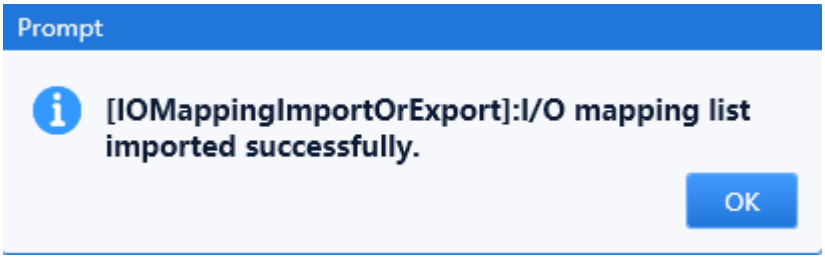
A file that meets the import conditions already exists.

Procedure

1. Right-click "Inovance_Bus (I/O module)" under the "Hardware configuration" node in the project tree, and select "Import I/O mapping".

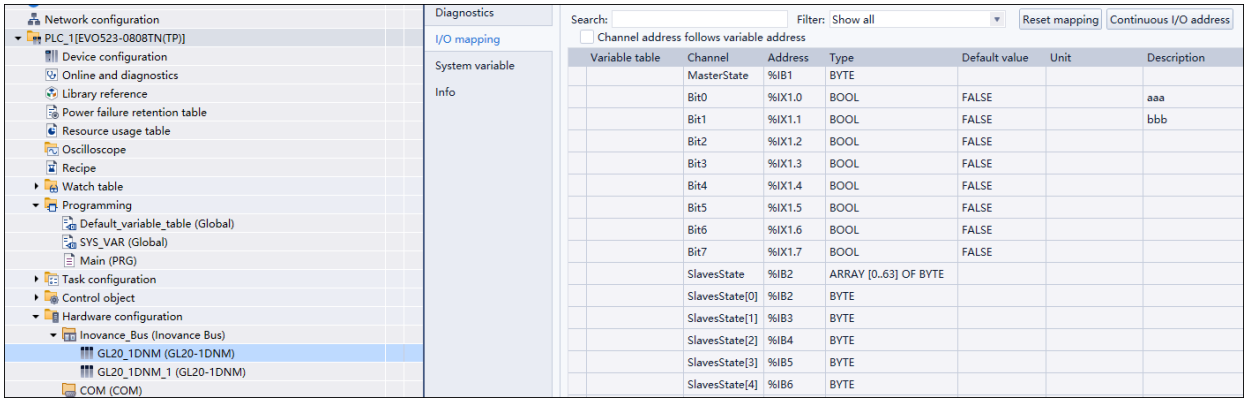


2. Select the file to import, and click "Open".
3. After successful import, click "OK" in the pop-up prompt box.



Result

Open the imported I/O mapping table under Inovance_Bus to view the updated configuration, as shown in the figure below.



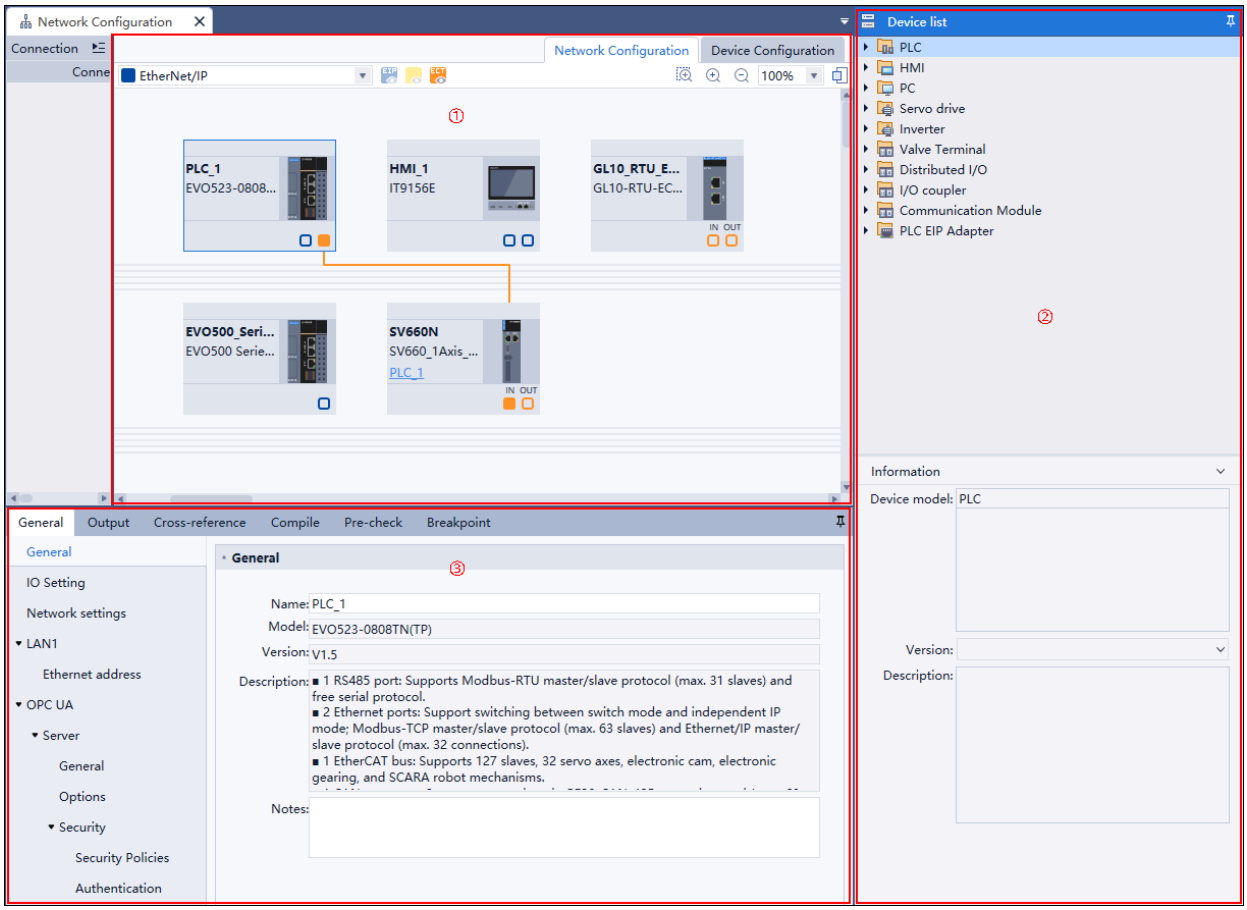
10.6.2 Device and Network Editor

10.6.2.1 Overview

- Functions of the device and network editor
  - The device and network editor is an integrated development environment used for configuring devices and modules, networking, parameter assignment, and other operations.

- Double-click "Device configuration" in the project tree to open the device and network editor.

- Structure of the device and network editor
  - The device and network editor consists of the following components:



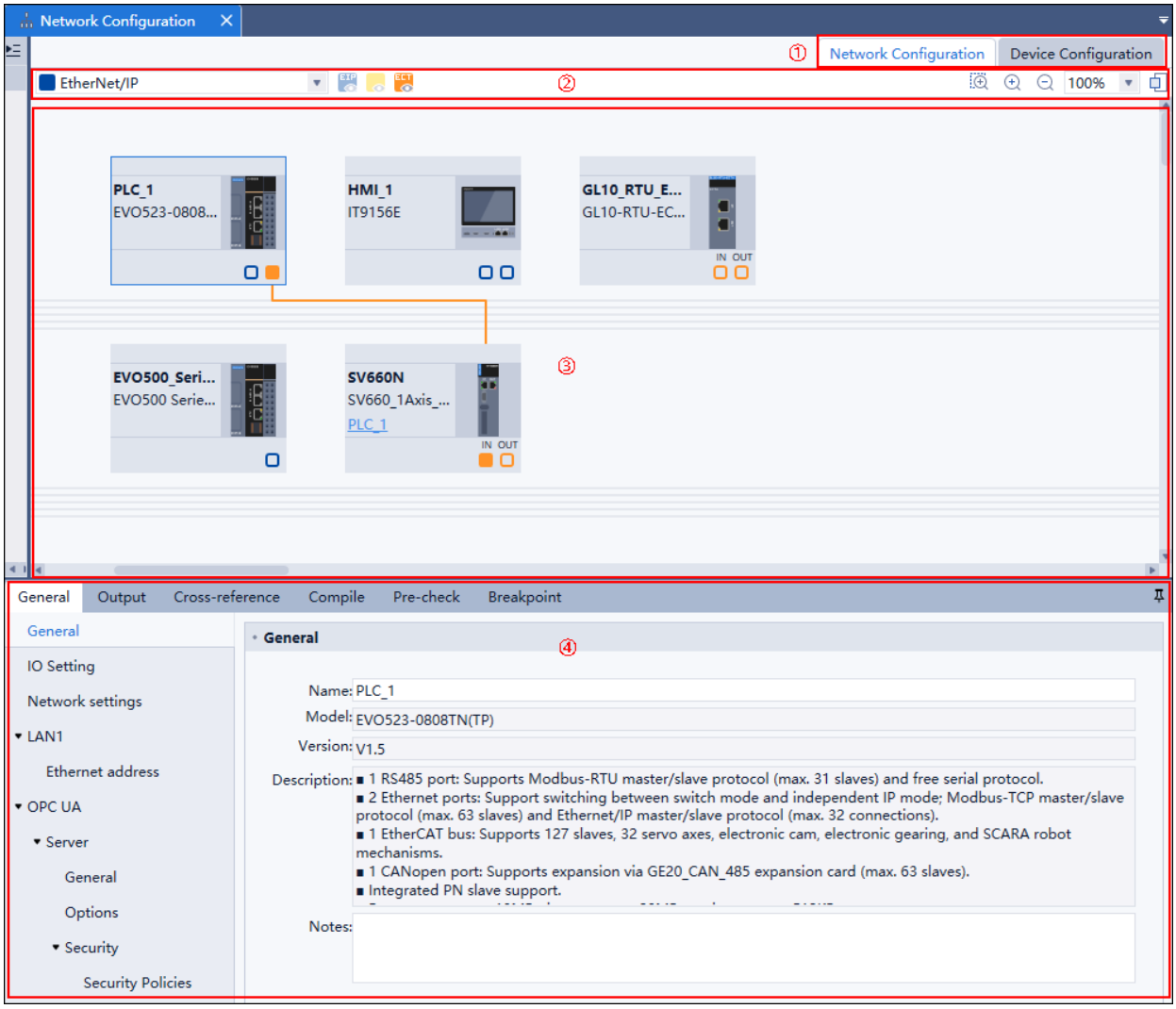
No.	Function	Description
①	Device configuration and network configuration	A graphical area including the device view and network view. You can switch views according to your usage requirements.
②	Device List	Drag devices and modules from the device list to the network configuration or device configuration tab.
③	Monitoring area	Contains relevant information about the currently marked object, which can be modified here.

10.6.2.2 Network View

The network configuration view is one of the two workspaces in the device and network editor, used to plan the network structure between devices, and can perform the following tasks:

- Configure and assign device parameters.
- Network devices.
- Modify device names.

As shown in the figure below, it displays the components of the network configuration view:



No.	Function	Description
①	View switch	Switches between device configuration and network configuration views.
②	Toolbar	Performs operations such as zooming in/out.
③	Graphical area	Contains basic configuration operations.
④	Information display area	Displays relevant information about the device.

- **Toolbar**

  The toolbar provides the following functions:

Icon	Description
	Switches protocol. You can switch the supported protocols of the device through the drop-down menu.
	Displays the EtherNet/IP connection selection status. The connection line will be highlighted when selected.

Icon	Description
	Displays the EtherCAT connection selection status. The connection line will be highlighted when selected.
	Zooms in the overall view. Click the "Zoom in" icon to zoom in the effect gradually by 25%, 75%, 100%, 150%, 200%, 300%, and 400%, with the center of the canvas as the zoom center and the current ratio as the starting ratio.
	Zooms in the selected area Click the "Area zoom in" icon. Press and hold the mouse at the starting position of the selected area to be zoomed in, move the mouse to draw a selection box, and release it at the appropriate position. Only the selected area will be zoomed in.
	Zooms out the overall view. Click the "Zoom out" icon to zoom out the effect gradually by 400%, 300%, 200%, 150%, 100%, 75%, and 25%, with the center of the canvas as the zoom center and the current ratio as the starting ratio.
	Micro view Micro view display: Shows an overview of the canvas in map form, with the current workspace highlighted by a gray semi-transparent frame on the map. Micro view operation: You can drag the gray semi-transparent frame, and the content within the frame will be displayed in the workspace simultaneously. Zoom ratio: Zooms the view based on the selected zoom ratio, with the center of the canvas as the zoom center.

- **Graphical area**
  The graphical area of the network configuration displays all devices, networks, connections, and relationships related to the network. In this area, you can add devices from the device list, connect devices through interfaces, and configure communication settings.

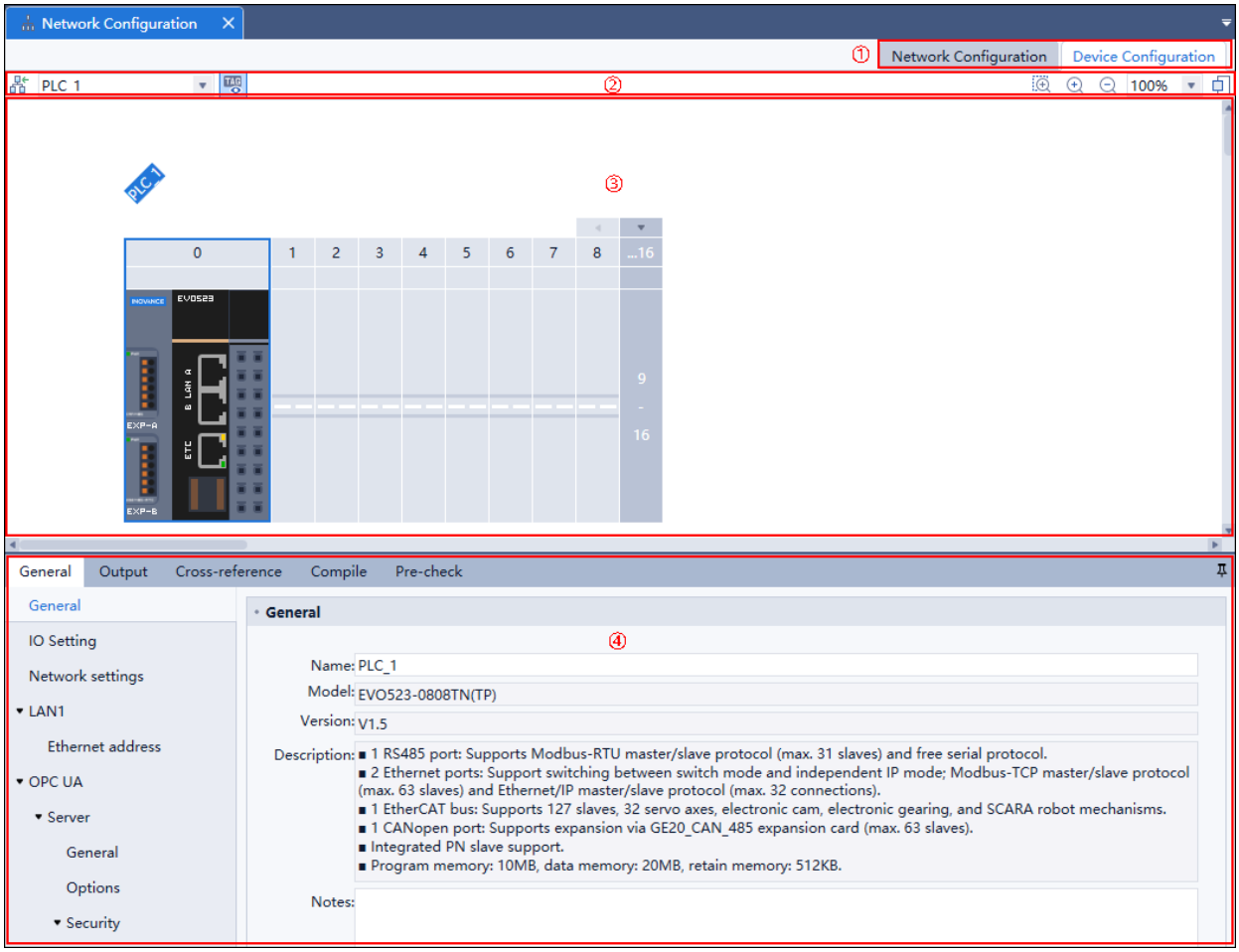
  You can also use scroll bars to view the network configuration, or use Ctrl + mouse wheel to set the zoom level of the graphics.
- **Information display area**
  The information display area shows relevant information about the selected device.
- **View switch**
  The view switch supports switching between the device configuration view and the network configuration view.

10.6.2.3 Device View

Device configuration is one of the two workspaces in the device and network editor, used to configure parameters for devices and submodules. The following operations can be performed:

- Configure and assign device parameters.
- Configure and assign module parameters.
- Modify the names of devices and modules.

As shown in the figure below, it displays the components of device configuration:



No.	Function	Description
①	View switch	Switches between device configuration and network configuration views.
②	Toolbar	Performs operations such as zooming in/out.
③	Graphical area	Contains basic configuration operations.
④	Information display area	Displays relevant information about the device.

- **Toolbar**

  The toolbar provides the following functions:

Icon	Description
	Switches to the network configuration view. Click the icon to switch to the network configuration view.
	Selects the device. Use the drop-down list to switch between devices in the project within the device configuration. After selecting a device, the device configuration view displays the device graphics, and the device list is updated to show the modules that match the selected device.

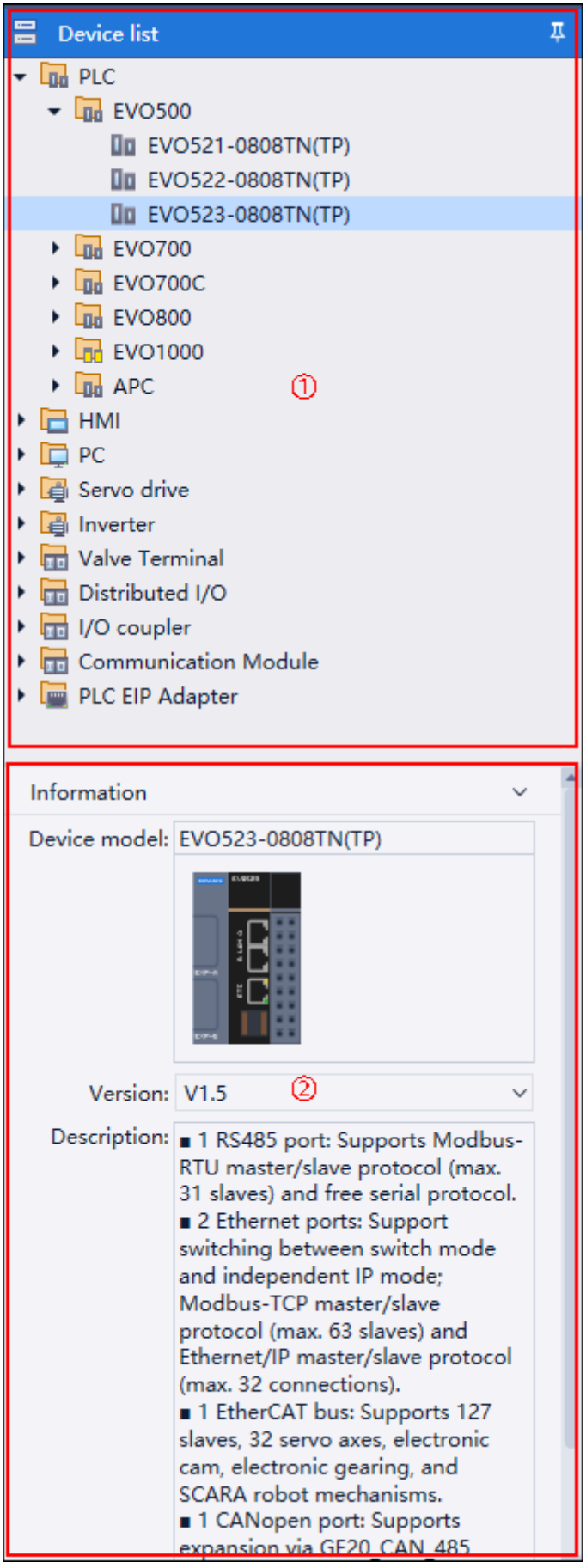
Icon	Description
	Shows/hides module labels. Device names can be modified here.
	Zooms in the overall view. See 10.6.2.2 Network Configuration View.
	Zooms in the selected area See 10.6.2.2 Network Configuration View.
	Zooms out the overall view. See 10.6.2.2 Network Configuration View.
	Micro view Micro view display: Shows an overview of the canvas in map form, with the current workspace highlighted by a gray semi-transparent frame on the map. Micro view operation: You can drag the gray semi-transparent frame, and the content within the frame will be displayed in the workspace simultaneously. Zoom ratio: Zooms the view based on the selected zoom ratio, with the center of the canvas as the zoom center.

- **Graphical area**
  The graphical area of the device configuration view displays hardware components, which are assigned to each other via one or more racks if necessary. For devices with racks, users can install other hardware components from the device list into the slots of the rack.
- **Information display area**
  The information display area shows relevant information about the selected device.
- **View switch**
  The view switch supports switching between the device configuration view and the network configuration view.

10.6.2.4 Device List

The device list is located in the toolbox on the right side of the platform, consisting of two parts: the device list and information. The two parts are placed in separate panes that support expansion, collapse, and can be dragged up and down between panes.

- **Structure**



- **Device list**
  The device list categorizes and displays the directory of components, supporting dragging and moving devices or modules from the device list to the graphical area of the device configuration/network configuration view.
- **Information**

Select a project tree node and the detailed information about the selected device will be displayed in the information pane below. The information pane includes the following content:

Content	Note
Model	The model of the device selected from the project tree node.
Diagram	The diagram of the device selected from the project tree node.
Version	The firmware version of the device selected from the project tree node.
Description	A detailed description of the device selected from the project tree node.

10.6.3 EtherCAT Communication Configuration (EVO500, EVO700, APC500, EVO700C)

10.6.3.1 Adding an EtherCAT Device

10.6.3.1.1 Adding a Master

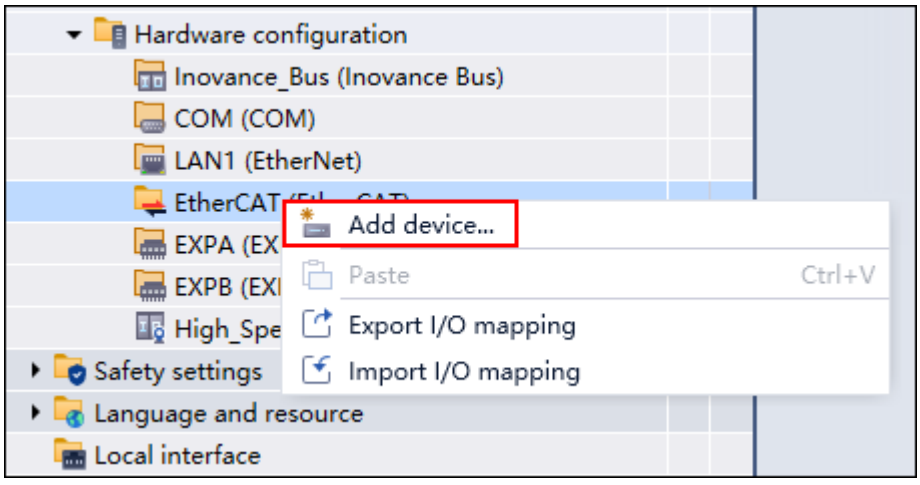
The controller supports EtherCAT communication and EtherCAT master devices can be added.

Prerequisites

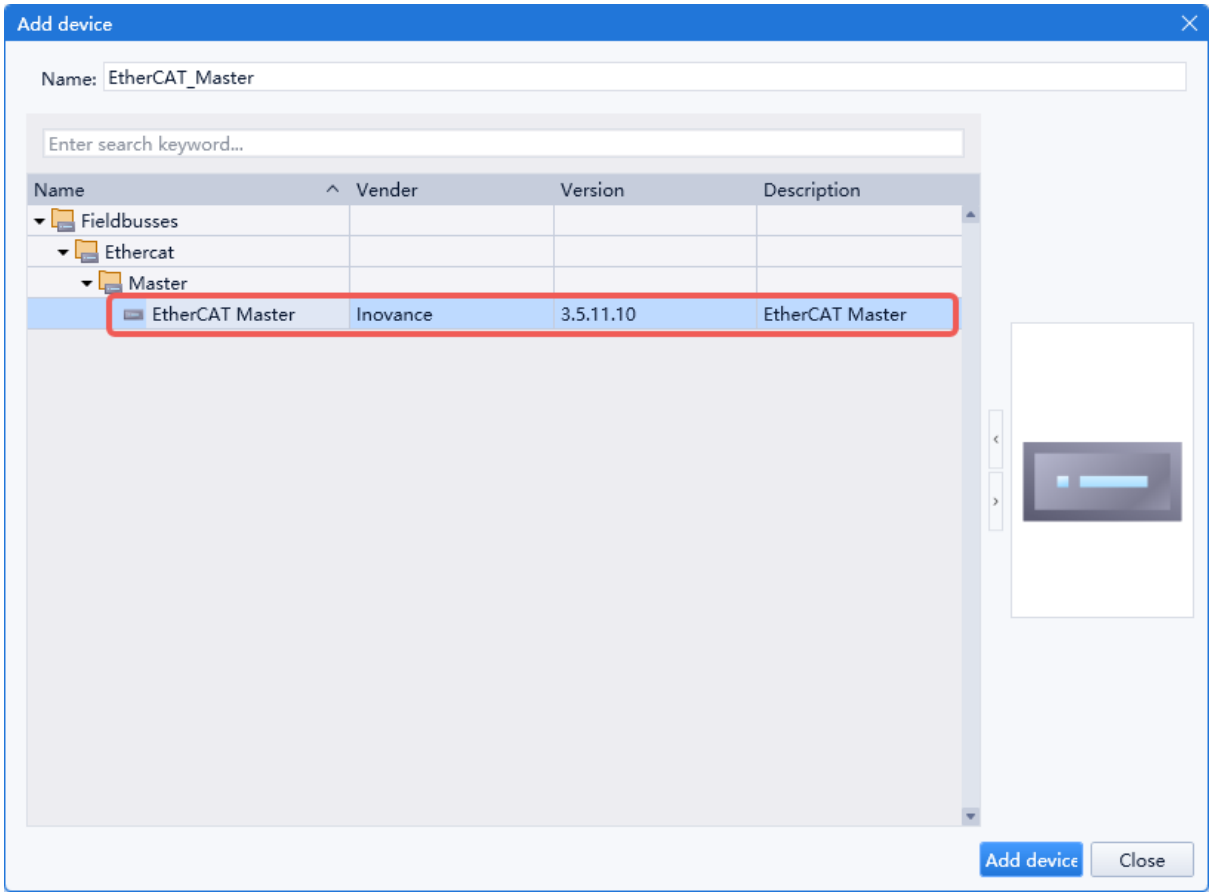
- A project has been created.
- A device has been added.

Procedure

1. Right-click "EtherCAT" under "Hardware configuration" in the project tree and select "Add device".

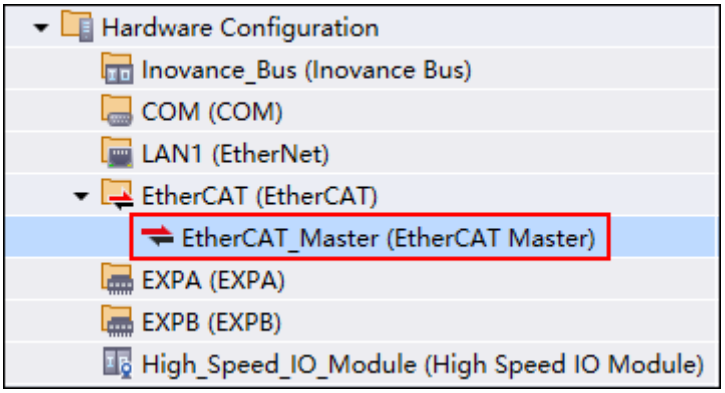


2. In the pop-up window, select "EtherCAT Master" and click "Add device" to add the EtherCAT master device.



Result

Under the EtherCAT node, you can see the added master device node.



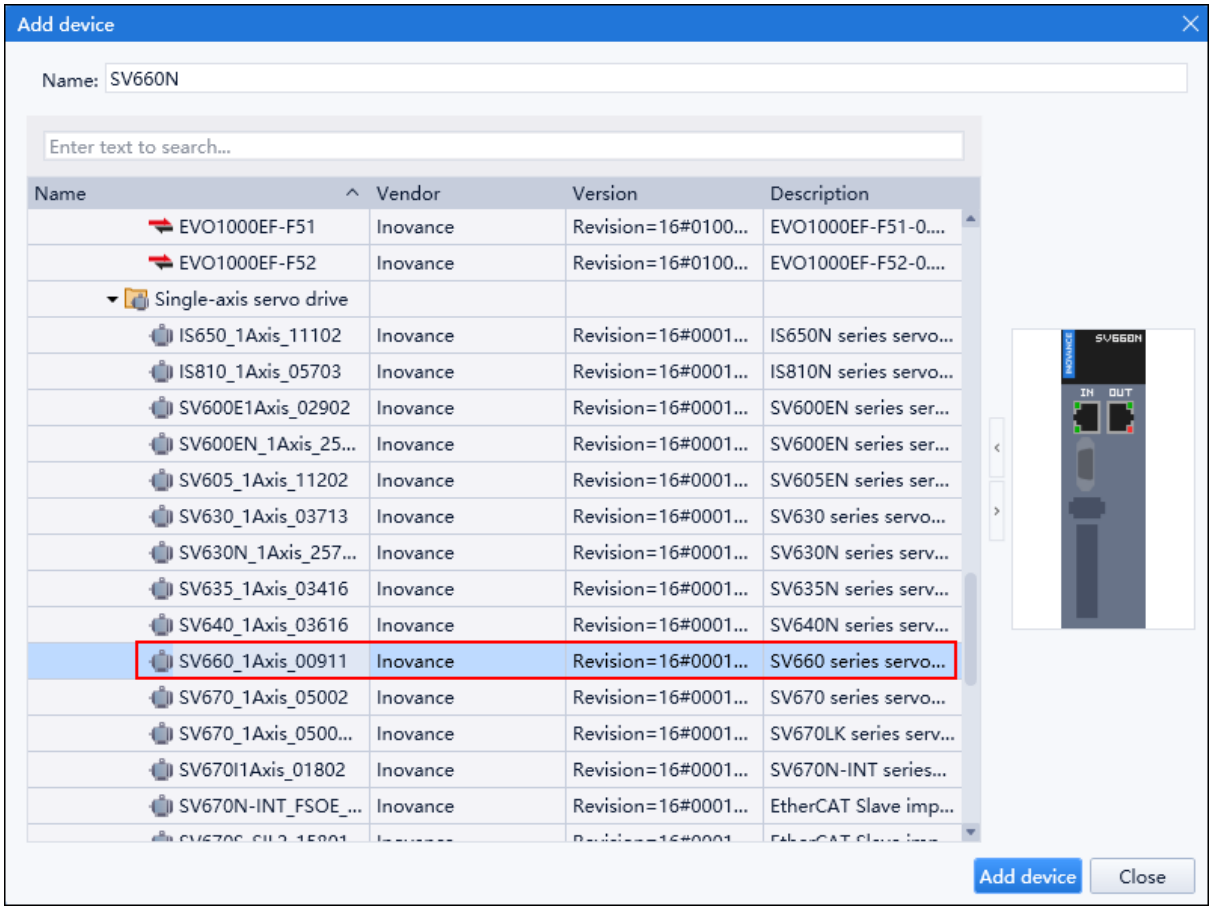
10.6.3.1.2 Adding a Slave

Prerequisites

The EtherCAT master has been added.

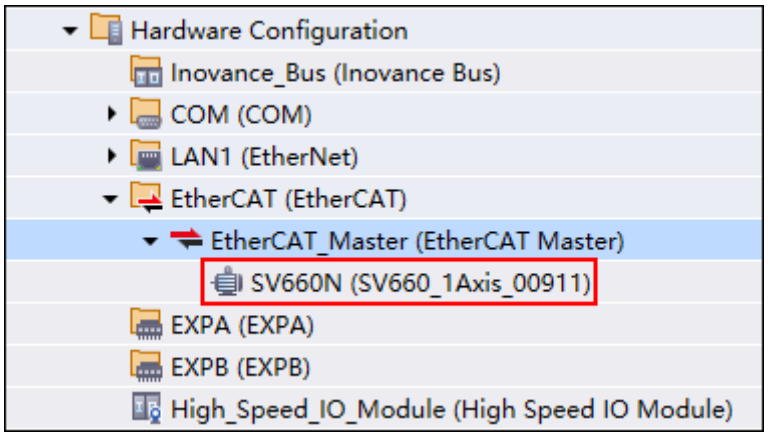
Procedure

1. Right-click the "EtherCAT_Master" node and select "Add device".
2. In the pop-up window, select the slave device and click "Add device".



Result

Under the "Hardware configuration > EtherCAT > EtherCAT_Master" node, you can see the added slave devices.



10.6.3.1.3 Adding a Wireless Management Station

Restrictions

- The wireless management station must be the first station of the PLC.
- Ring network is not supported by the wireless management station.
- Only one wireless management station can be added to a single PLC.

Prerequisites

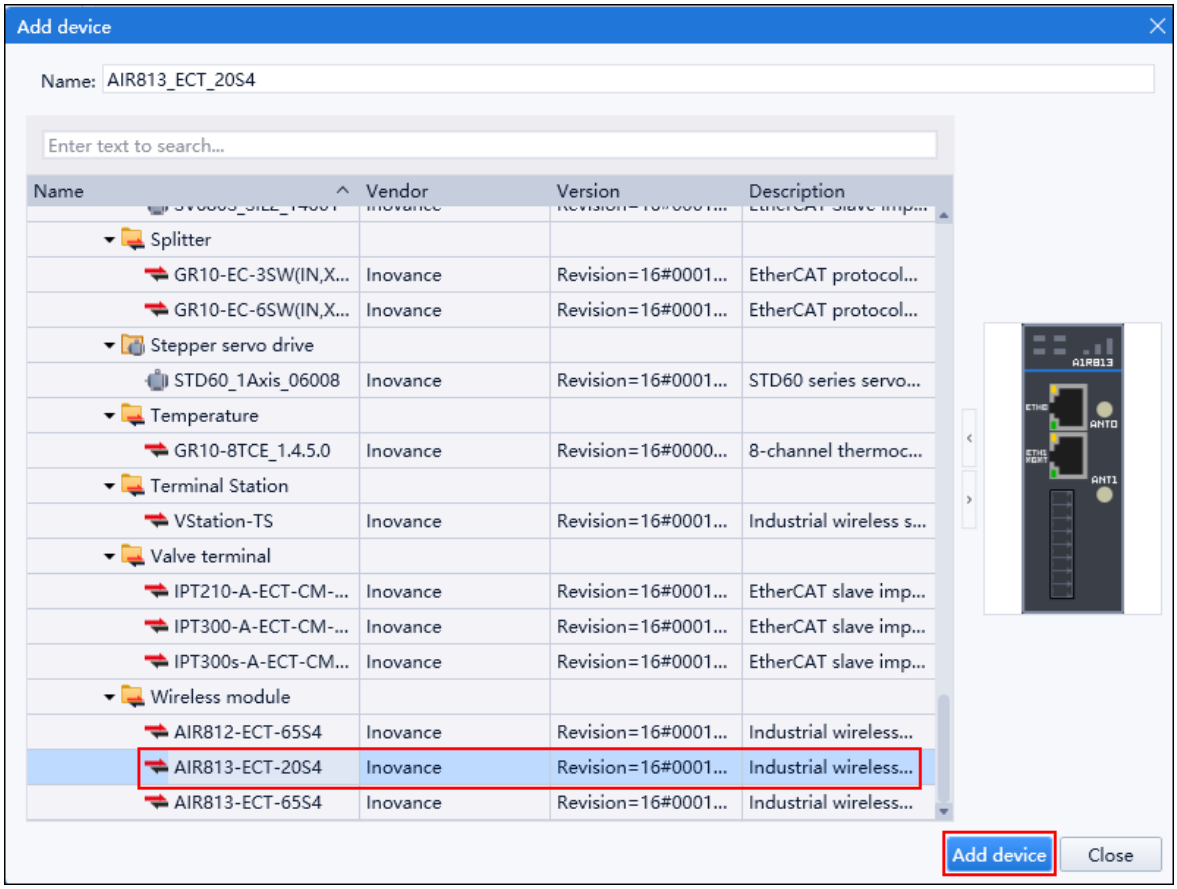
The EtherCAT master has been added.

Procedure

1. Add a wireless management station.

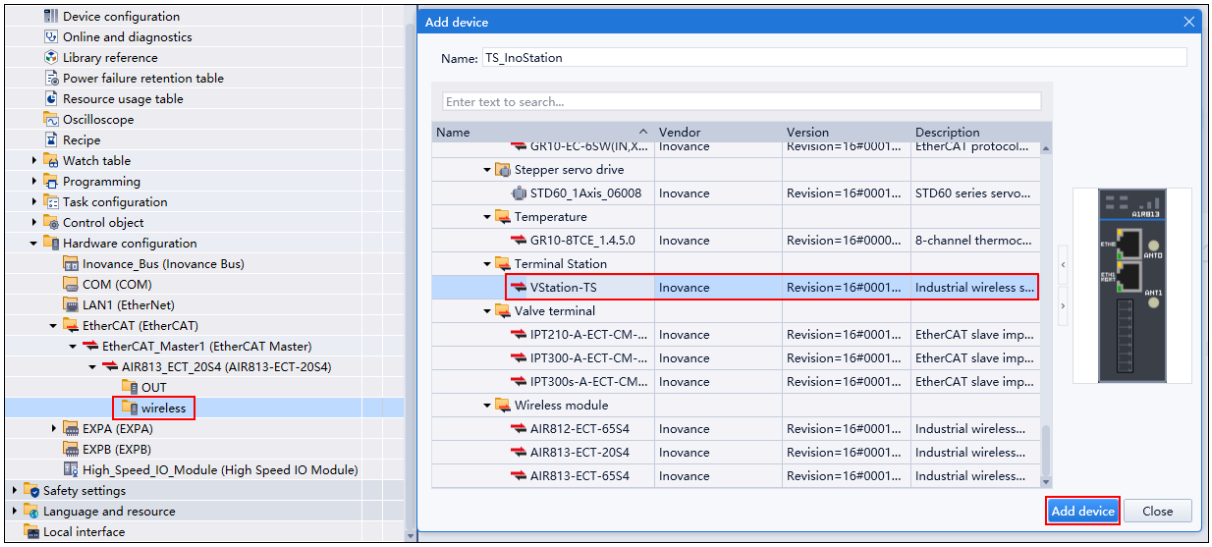
   a. Right-click the "EtherCAT_Master" node, and select "Add device".

   b. In the pop-up window, select a wireless management station, and click "Add device".



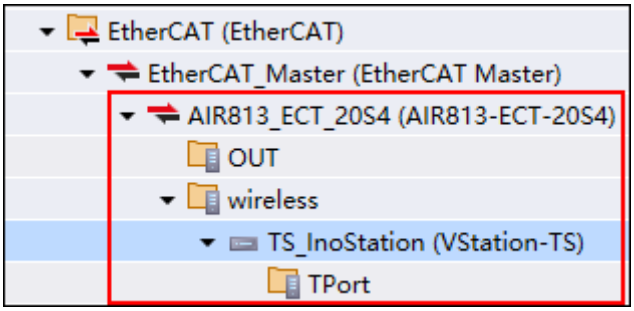
2. Add a terminal station to the wireless port of the management station.

   In the "Add device" window, select a terminal station, and click "Add device".



Result

The added wireless management station and terminal station can be viewed under "Hardware configuration > EtherCAT > EtherCAT_Master".



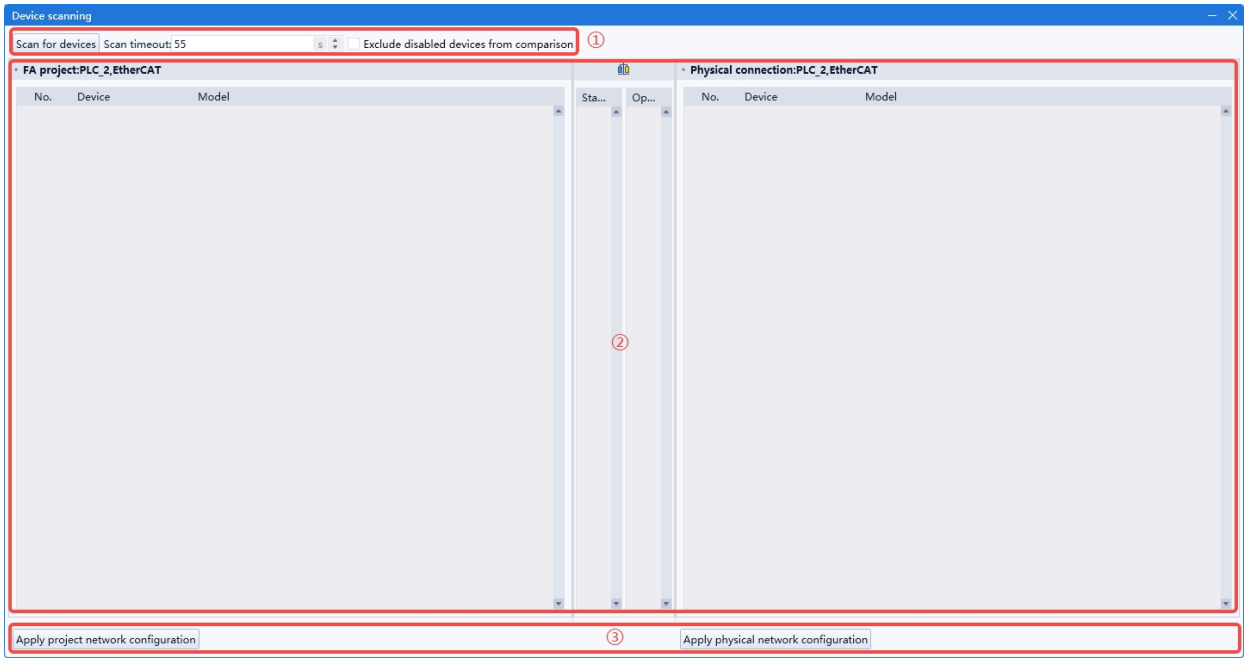
10.6.3.1.4 Adding an Online EtherCAT Slave to Project

Prerequisites

- The PC and PLC have been correctly connected through the gateway.
- The EtherCAT_Master has been configured.

Procedure

1. Download the project to the PLC.
2. Right-click "EtherCAT_Master", select "Scan for devices", and the device scanning window will pop up.



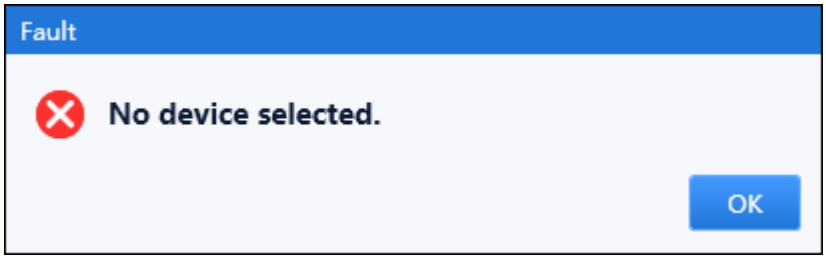
The items in the above figure are defined as follows:

No.	Function	Description
①	Scan for devices	Scans the devices.
	Scan timeout	Scan duration, with a range from 5s to 300s.

No.	Function	Description
	Exclude disabled devices from comparison	When selected, disabled devices are not involved in device comparison; During device copy, if device comparison succeeds, device configuration is retained; if it fails, the device is deleted. The disabled status of devices is adjusted based on their actual status in the configuration list. For example, if a device in the physical device list is successfully compared, its disabled status is enabled. However, if the originally disabled device exists in the project configuration device list, the disabled status remains upon successful comparison.
	Set alias address	After editing the alias address in the "Write alias address" field, click "Set alias address" to make the address effective.
	Batch assign aliases	Alias addresses are automatically assigned starting from the first slave, incrementing by 1 sequentially. If no alias address is set for the first slave row, the original default alias address is used as the starting address. After editing the alias address in the "Write alias address" field, click "Batch assign aliases" to apply the changes.
②	Copy to the left	Copies scanned devices from the network to the project network.
	Status	Compares the data between the project and physical devices.
③	Apply project network configuration	Applies the device configuration in the project network to the project.
	Apply physical network configuration	Applies the configuration of all scanned devices to the project

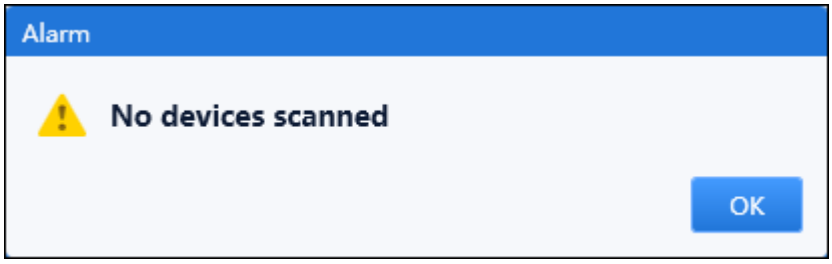
Scanning faults

1. If the PC is not connected to the PLC and you directly scan the slave devices, the following prompt will appear:



Solution: Check the communication connection status between the PC and the PLC.

2. If the PC is connected to the PLC but no slaves are connected under the PLC, the following prompt will appear:



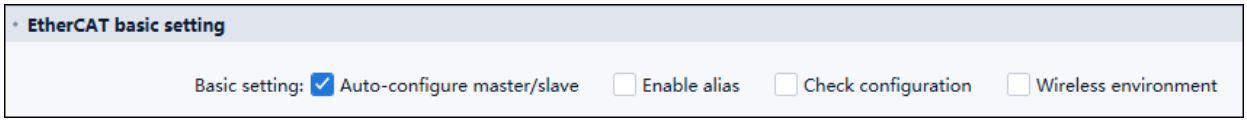
CAUTION

After the EtherCAT bus starts normally, if the project is switched to offline mode, the scan results may fail or be inaccurate when scanning devices online. The reason is that during the scanning process, reading slave information via SDO communication (such as object dictionary 0xF050) may time out, affecting the scan results (SDO communication is asynchronous, and CPU load, bus cycle time, and user program execution time can all affect SDO communication).

10.6.3.2 Configuring Master Parameters

10.6.3.2.1 General Settings

EtherCAT basic settings



- Auto-configure master/slave
  After selecting this option, the system will automatically complete the main configuration for master and slave. Meanwhile, the "FMMU/Sync" tab in the editor of all slaves will be hidden.

NOTE

- "Auto-configure master/slave" is enabled by default; it is strongly recommended to keep it enabled in standard applications. If disabled, all masters and slaves must be configured manually, requiring the operator to have the necessary expertise.
- When configuring slave-to-slave communication, you must disable "Auto-configure master/slave".

- Enable alias
  Used to manage alias settings for slaves. When selected, the "Alias address" in all slave configurations will be enabled; for details, see 10.6.3.2.6 Overview.
- Check configuration
  Configuration checking typically refers to using an EtherCAT slave configuration tool to check or set various slave parameters. This tool provides various functions, such as reading and modifying slave parameters, displaying slave information, commissioning slave status, and so on.

  The current configuration parameters of the slave can be viewed using the configuration check command, such as the IP address, port number, device name, and so on. By checking the configuration,

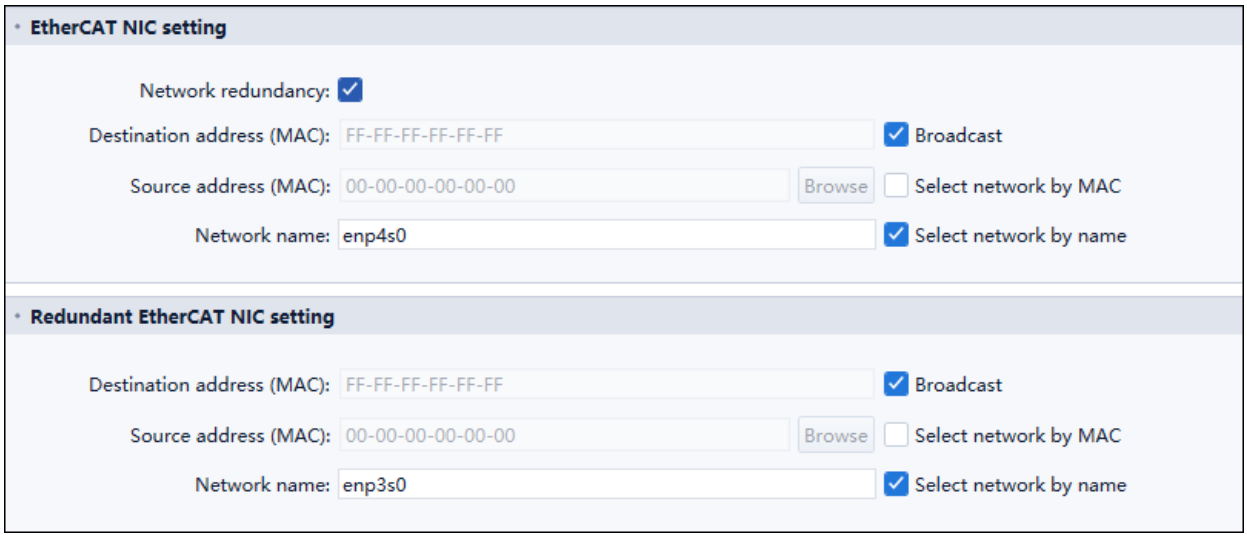
you can ensure that the configuration of the slave devices is correct, thus ensuring the normal operation of EtherCAT communication.

- Wireless environment
  After selecting this option, you can discover and add wireless management terminals through scanning; if not selected, wireless management terminals cannot be scanned.

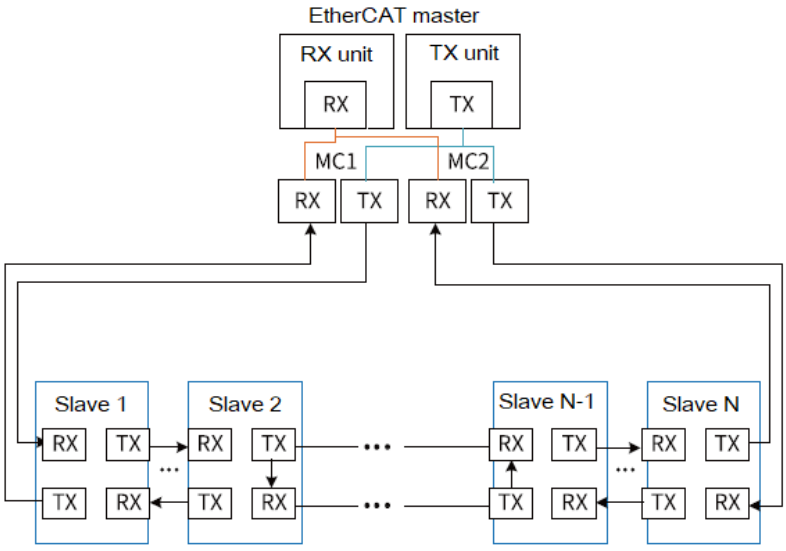
EtherCAT NIC setting

The EVO760 series supports EtherCAT network redundancy function.

After adding an EtherCAT master, double-click it to enter the configuration page. Check the "Network redundancy" option to enable the redundancy function.



In a ring topology, if redundancy is required, enable this option. After enabling, if a single-point fault occurs in the network connection, the EtherCAT network function can still operate normally.



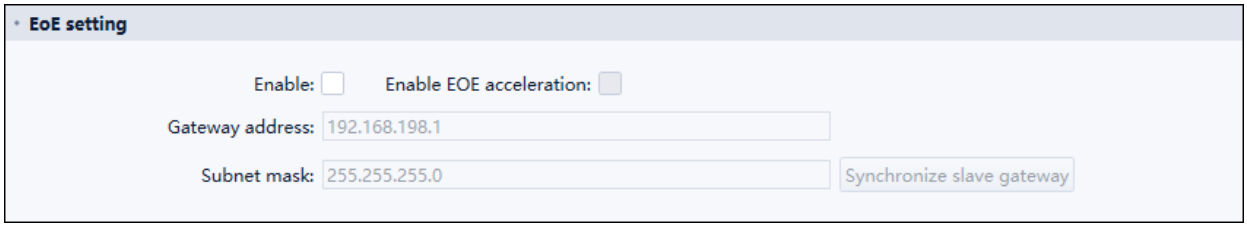
Configuration Item	Description
Destination address (MAC)	The MAC address of the EtherCAT network member receiving the message. If the "Broadcast" option is checked, the broadcast address (FFFFFFFFFFFF) is used.

Configuration Item	Description
Source address (MAC)	The MAC address of the PLC. Select the MAC address of the currently available target device through the "Browse..." button. Each EtherCAT NIC has a unique MAC address. If "Select network by MAC" is checked, this project cannot be used on other devices.
Network name	Network card name. To make the project independent of the device, it is recommended to check "Select network by name".

Redundant EtherCAT NIC setting

If the "Network redundancy" option is enabled, the "Redundant EtherCAT NIC settings" will be displayed. Users can define the relevant settings.

EOE setting



Ethernet over EtherCAT (EoE) allows any Ethernet device to be connected to EtherCAT via a conversion terminal without affecting the real-time performance of EtherCAT. Similar to well-known Internet protocols, such as TCP/IP, VPN, PPPoE (DSL), Ethernet frames are transmitted via the EtherCAT protocol. This setting allows standard network devices to be connected to terminal devices, such as printers or PCs, via a switch.

- Enable: Default is disabled. Check "Enable" to modify the gateway address and subnet mask address for EoE.
- Enable EOE acceleration: Default is disabled. Enable it to improve the transmission rate of EOE.

> CAUTION
>
> Enabling this function will consume network resources and increase the system's overall load. If there is no special need, keep it disabled by default.

- Synchronize slave gateway: Click to synchronize the gateway configuration to the slaves.

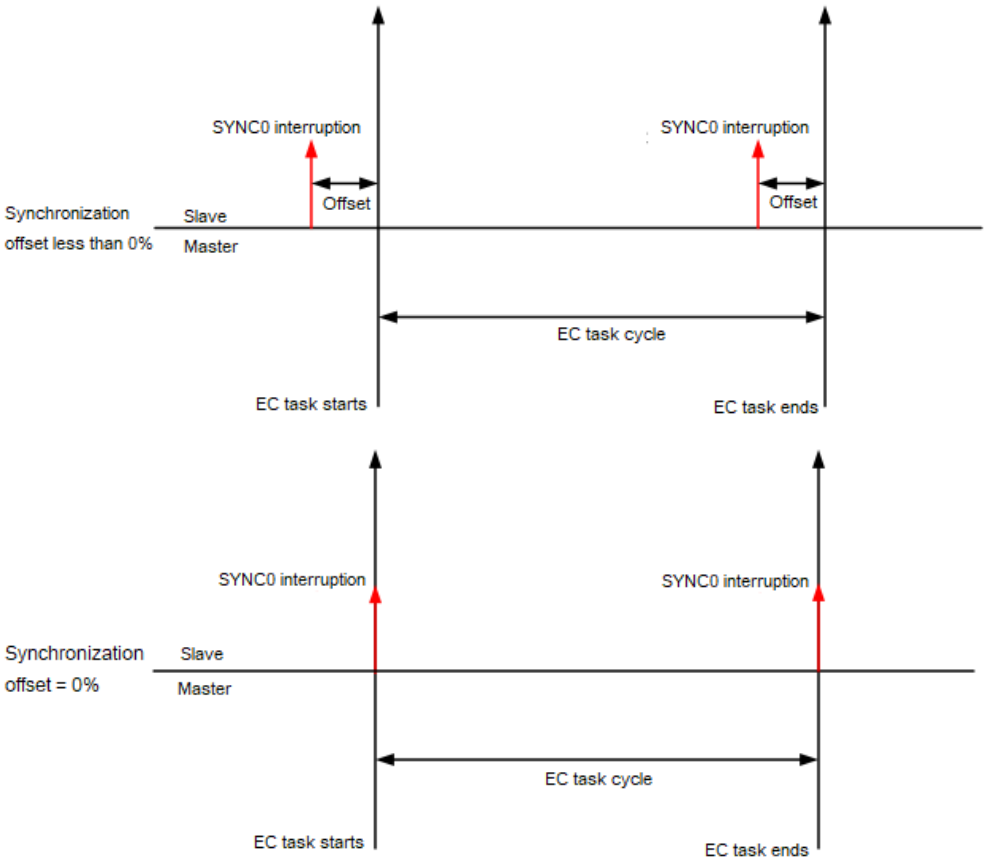
Distributed clock

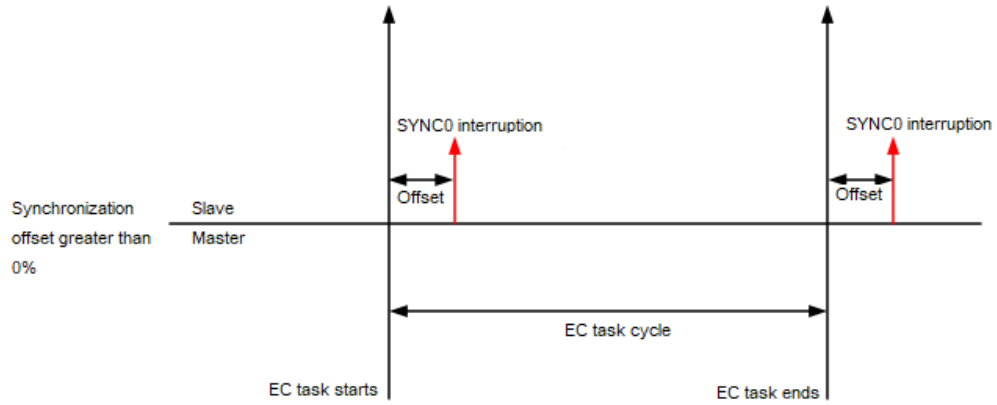
Distributed clock (DC) enables all EtherCAT devices to share the same EtherCAT system time, thus synchronizing task operations across devices. EtherCAT data exchange is based on a pure hardware mechanism. The master can calculate and measure time offset and transmission delay based on the time difference between the sent data frame and the received data frame, while compensating for the drift of the local clock.



- Sync window monitoring:
- Cycle time
  Indicates the cycle time for executing EtherCAT master functions. It must be the same as the cycle time of the IEC task bound to the EtherCAT master. If the distributed clock function of the slave is enabled, the "Cycle time" will synchronize with the "Distributed clock" value set in the slave device editor.
- Synchronization offset
  Indicates the cycle time offset ratio of the EtherCAT master's IEC task (PLC task) relative to the reference distributed clock (normally the SYNC0 synchronization interrupt). This parameter ranges from -50% to +50%, with a default value of 30%.

  Synchronization offset (%) = (SYNC0 interrupt time - PLC task cycle start time) / PLC task cycle time.



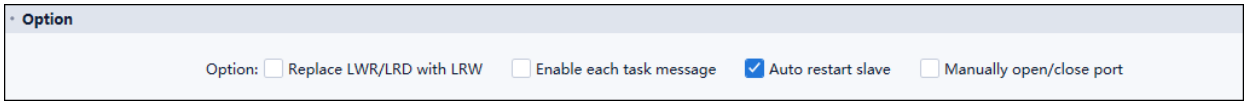


CAUTION

- By default, the PLC task cycle time is equal to the cycle time of the distributed clock of the slave.
- In actual configuration, the following factors must be considered: master clock jitter of the controller (system real-time performance), PLC task execution time, PLC task cycle time, and number of slaves.

- Synchronization window
  Indicates the time monitored by the synchronization window. If all slave synchronization information falls within this time window, the variable xSyncInWindow (IoDrvEtherCAT) is set to TRUE; otherwise, it is set to FALSE.

Options



- Replace LWR/LRD with LRW
  Disabled by default. Check it to enable the slave-slave communication. The logical addressing of the EtherCAT master will use logical memory read/write (LRW) command instead of logical memory read (LRD) and logical memory write (LWR) commands. Enabling LRW mode can save network bandwidth and optimize network load.
- Enable each task message

  CAUTION

  Enabling it will increase system load, so it is only intended for problem diagnosis and troubleshooting.

  Disabled by default. This option is used to enable or disable each task message. By default, each task message is disabled to avoid unnecessary interference with the system. If a fault or error occurs during operation, this option can be enabled for troubleshooting.
- Auto restart slave
  Enabled by default. After checking it, when a slave encounters disconnection, communication faults, or other faults, the system will automatically restart the slave to restore communication and system operation. This only applies to scenarios when all slaves encounter disconnection or communication

faults. After restart, if there are servo axes, the axis status will still report communication faults, and the MC_Reset instruction must be used to reinitialize them.

- Manual port control

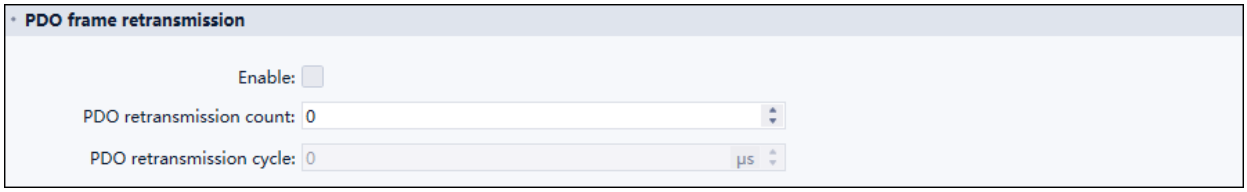
> CAUTION
>
> Enabling it will affect communication, so it is only intended for problem diagnosis and troubleshooting.

An EtherCAT slave supports up to four ports, which will automatically open or close based on whether a connection exists. If a connection exists, the port opens; otherwise, it closes. If the port is open, data frames entering the port are sent to other slaves and return to the port after being processed by other slaves. If the port is closed, data frames flow directly to the next port of the slave.
This option is disabled by default. After enabling, the port can be manually opened or closed, with the main scenarios as follows:

  - Fault diagnosis and troubleshooting: When a network fault occurs, you can manually open or close ports for fault diagnosis and troubleshooting. For example, when a port fails, you can manually close it to determine if it affects the entire network.
  - Security control: By manually opening or closing ports, you can enhance network security. For example, when a port is not needed, you can manually close it to reduce potential security risks.

PDO frame retransmission

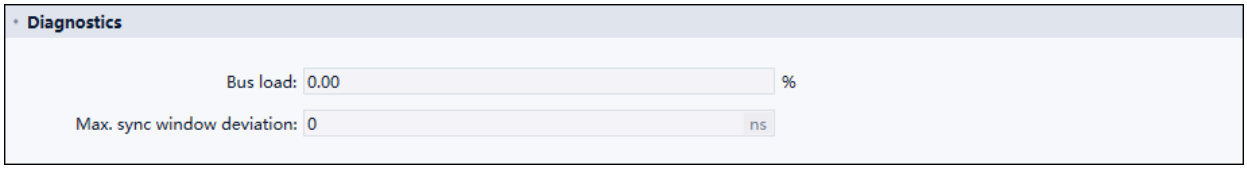
Disabled by default. Enabling it to re-transmit PDO frames. This feature is mainly used in high-reliability scenarios, but it increases CPU and bus load, so it is disabled by default.



- Enable: This option is automatically checked when the "PDO retransmission count" is set to a value greater than 0, with no need for manual configuration.
- PDO retransmission count: Sets the number of times a PDO frame is retransmitted.
- PDO retransmission cycle: Sets the cycle time for retransmitting PDO frames. Configuration is not supported.

Diagnostics

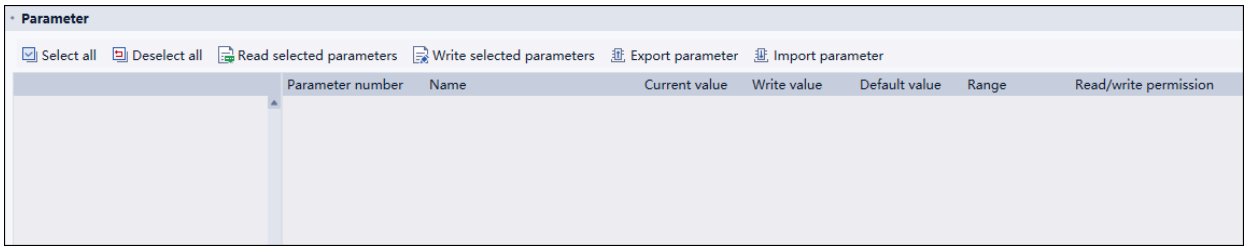
Visible only in online mode.



- Bus load: The bandwidth utilization of data transmission in the EtherCAT network.
- Max. sync window deviation: The maximum deviation (in ns) between the slave system clock and the reference clock.

10.6.3.2.2 Parameters

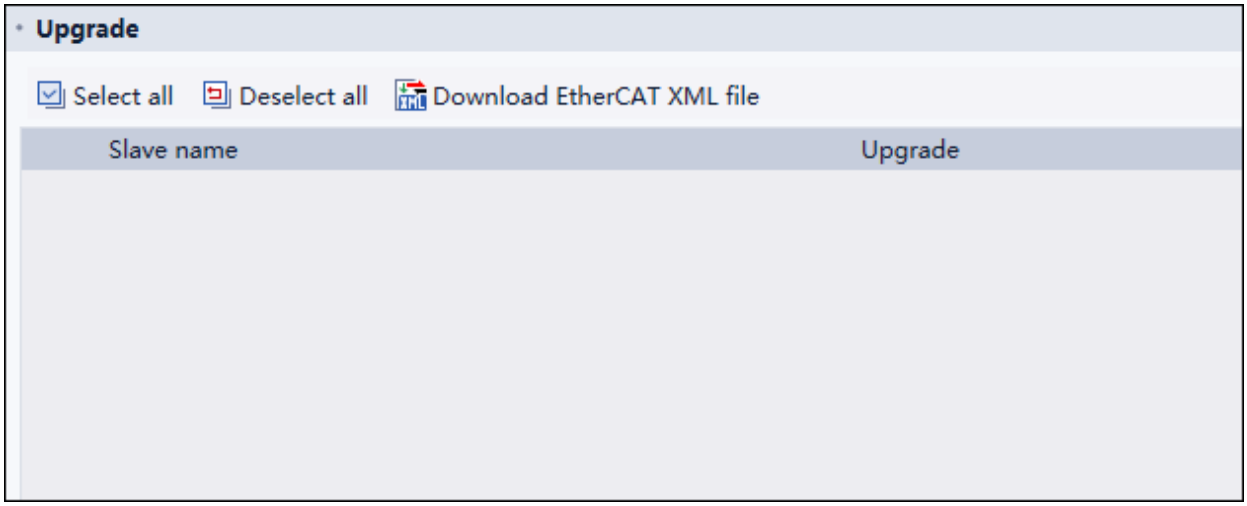
Function codes refer to the manufacturer parameters defined for Inovance servo products. In the master configuration, the manufacturer parameters of multiple product can be read and written in batches, imported, and exported, facilitating commissioning and maintenance.



- Select all: Selects all slaves and axes, as well as the servo function codes under the axes.
- Deselect all: Cancels all selected items.
- Read selected parameters: In the servo operation status, reads the servo function codes with RO, RW, or R properties.
- Write selected parameters: In the servo operation status, writes the servo function codes with W or RW properties.
- Export function code: Exports the servo function codes of all slaves in an Excel file.
- Import parameters: Imports the parameters of all slaves from an external Excel file. If the configuration does not match the file, a fault will be reported.

Note: The commissioning function of the master does not support SV660 series and SV680 series products. For details, see the commissioning function in 17.2.2 Parameter List of the drive settings. The drive devices of other series can perform function code debugging after binding the servo axes.

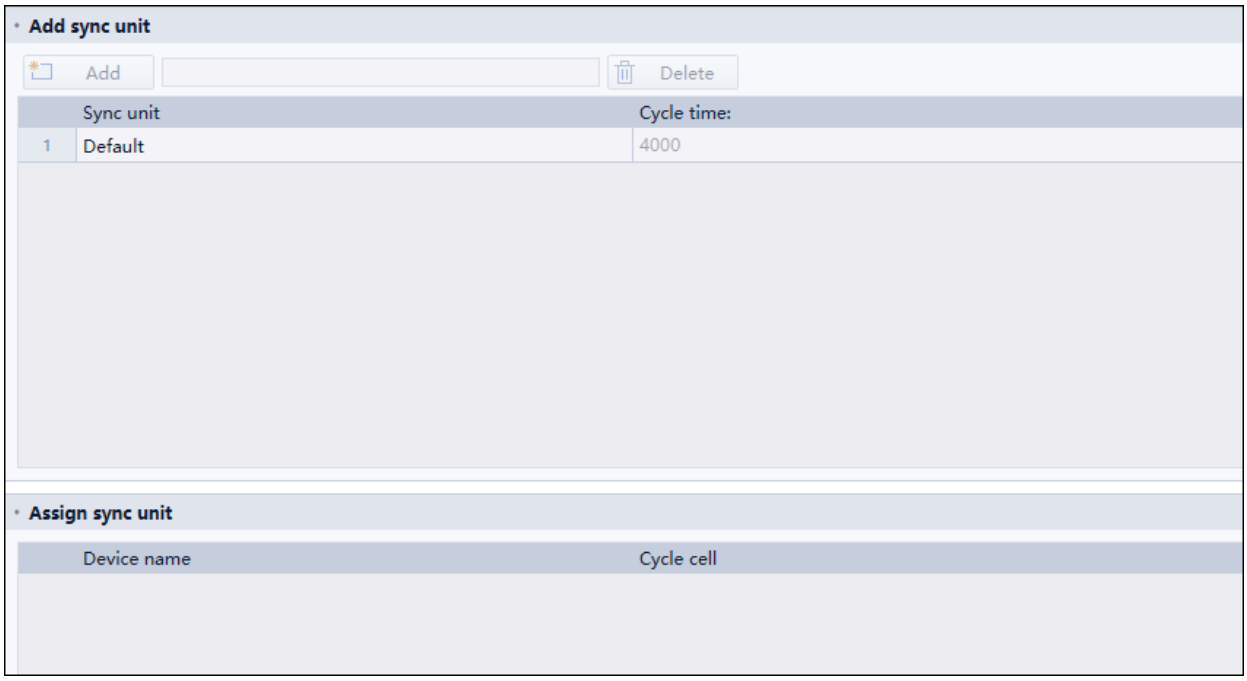
10.6.3.2.3 Upgrade



- Select all: Selects all slave devices.
- Deselect all: Deselects all selected slave devices.
- Download EtherCAT XML file: Downloads the EtherCAT slave XML file from the iFA fully integrated platform to the E2PROM of the slave device. Batch download can be achieved by checking multiple slave devices.

10.6.3.2.4 Unit Synchronization Assignment

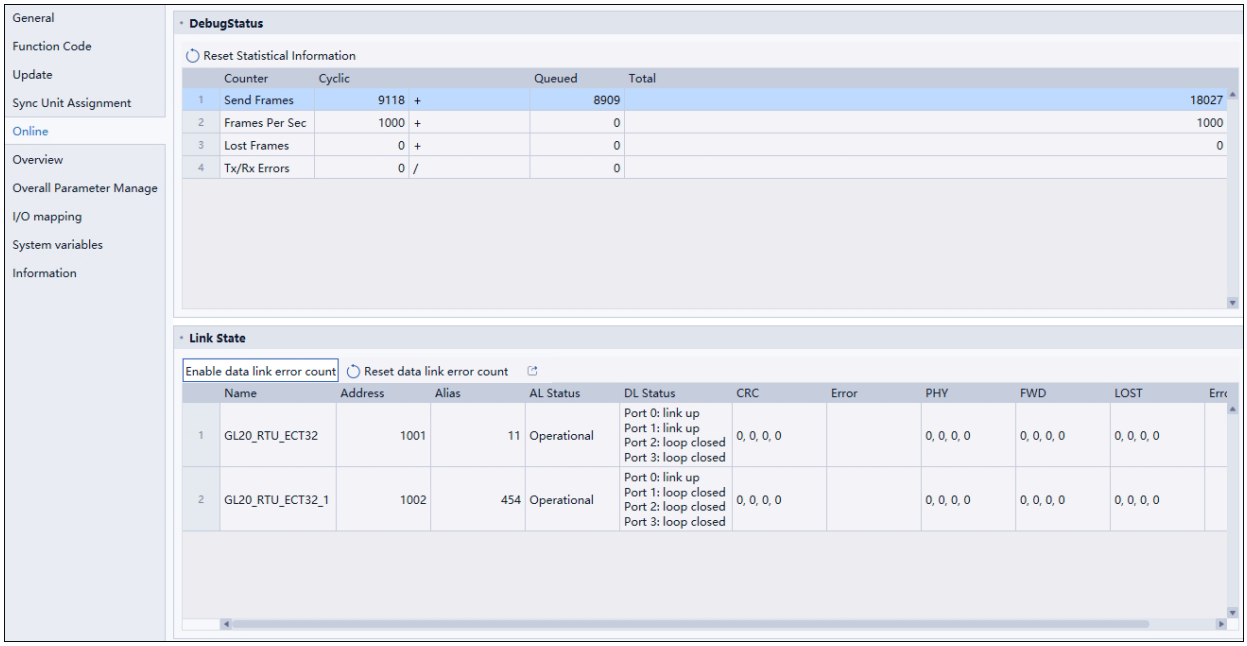
Unit synchronization allocation is mainly used for setting servos to different synchronization units in batches and categorizing them. After allocation, the synchronization unit cycle of the corresponding servos will be adjusted accordingly.



- Add synchronization unit: Enter the name of a synchronization unit and set the cycle time to add it. The cycle time is an integer multiple of the synchronization unit cycle.
- Allocate synchronization unit: Set the cycle of the added synchronization unit through a drop-down box. After setting, this parameter will be synchronized to the DC synchronization unit cycle of the allocated devices.

10.6.3.2.5 Online

After the device switches to online operation, this interface will display the operation status and statistics to better monitor the device's operation.

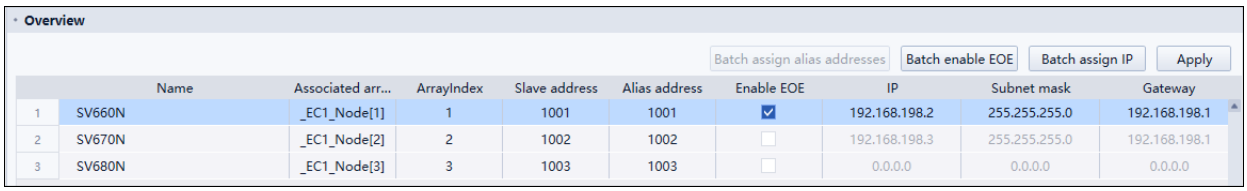


- Debug status
  Includes "Send frames (total number of frames sent)", "Frames per sec (frames sent per second)", "Lost frames (total number of lost frames)", and "Tx/Rx errors".
  Click "Reset statistics" to clear all current cumulative counts.
- Link status
  - Enable data link error count: After enabling this option, invalid frame count, receive error count, transfer error count, and lost link count are displayed in the "Link status" list.
  - Reset data link error count: After enabling this option, all accumulated error statistics in the "Link status" list are cleared and the counting restarts.
  - Export: Click to save the current link status information in CSV format to the backend.

10.6.3.2.6 Overview

The overview interface displays the address information of all slave devices, supports EoE parameter settings, and allows users to directly complete EoE commissioning.

> NOTE
>
> Before configuring EoE parameters, enable EoE settings in the "General" interface. Once enabled, "IP", "Subnet mask", and "Gateway" become editable.



Parameter	Description
Name	Displays the names of all slaves under the master, which cannot be modified.

Parameter	Description
Associated array	Also known as the slave array. Used to centrally manage all EtherCAT slave devices in the bus network. Each EtherCAT master corresponds to a slave array, which stores the corresponding EtherCAT slave system variable information.
Data index	The index of the associated array (typically the slave number). Supports manual editing by the user.
Slave address	Displays the address information of the slave station, which cannot be modified.
Alias address	• When the "Enable alias" option in the "General" tab is not checked, the alias address cannot be modified. • When the "Enable alias" option is checked, batch modification of aliases is supported, and the device aliases must be unique (no duplicates allowed). After the alias settings are completed, they are automatically synchronized to the alias configuration parameters of the sub devices.
Enable EOE	When EoE is enabled, if any of the "IP", "Subnet mask", or "Gateway" is empty, they will be automatically assigned based on EoE setting in the "General" interface.
IP	IP address: Supports user input or modification.
Subnet mask	Subnet mask address: Supports user input or modification.
Gateway	Gateway address: Supports user input or modification.

- Batch assign alias addresses: Automatically sets alias addresses from 1 to 30000 according to the configuration order.
- Batch enable EoE: Enables all devices supporting EoE, with gateway not assigned by default.
- Batch assign IP: Assigns gateways to all EoE-enabled devices based on the EoE setting.
- Apply: Saves all modified settings.



When devices are online, the above buttons will not be displayed.

10.6.3.2.7 Machine Parameter Management

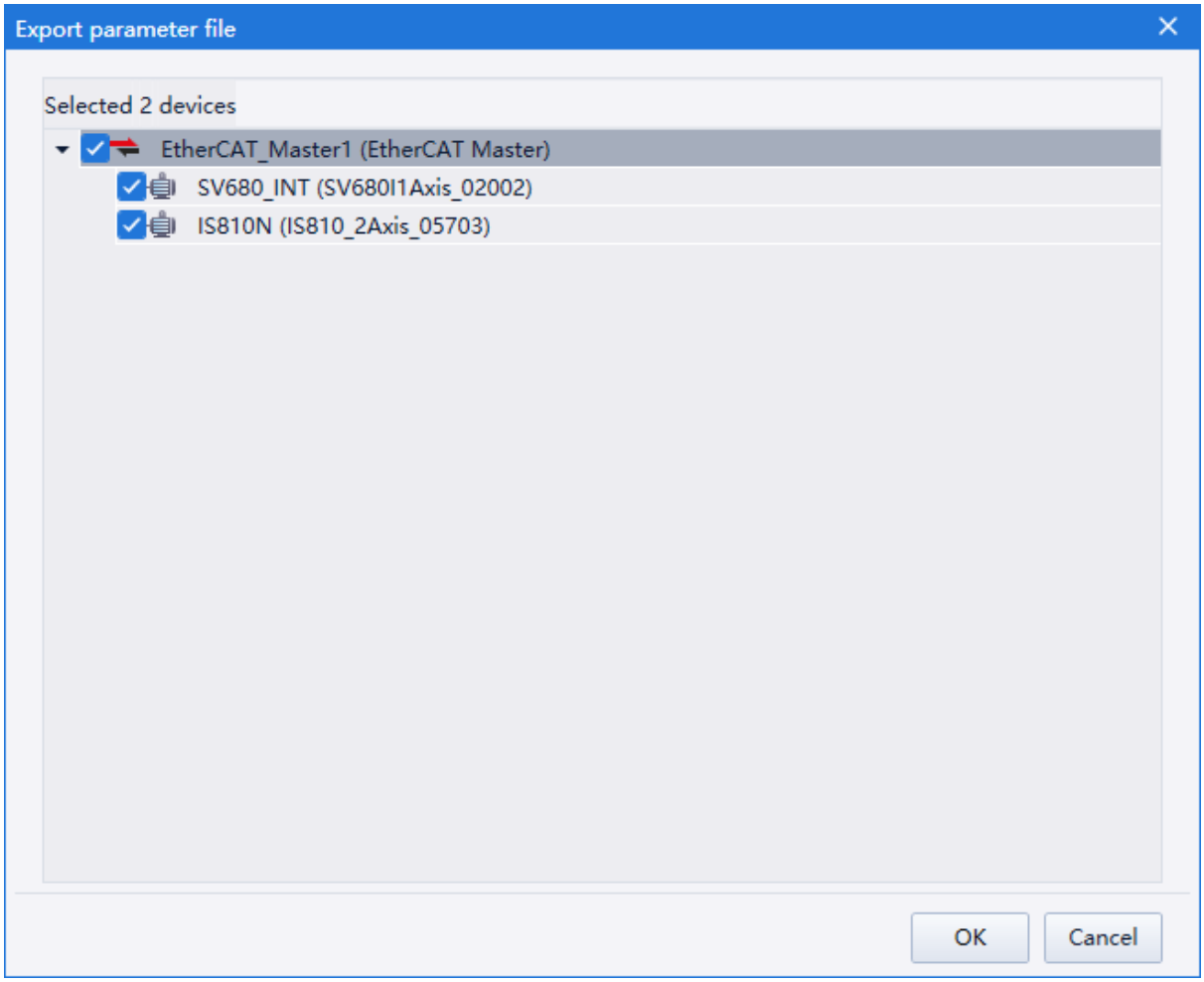
10.6.3.2.7.1 Machine Recipe Management

Machine recipe import

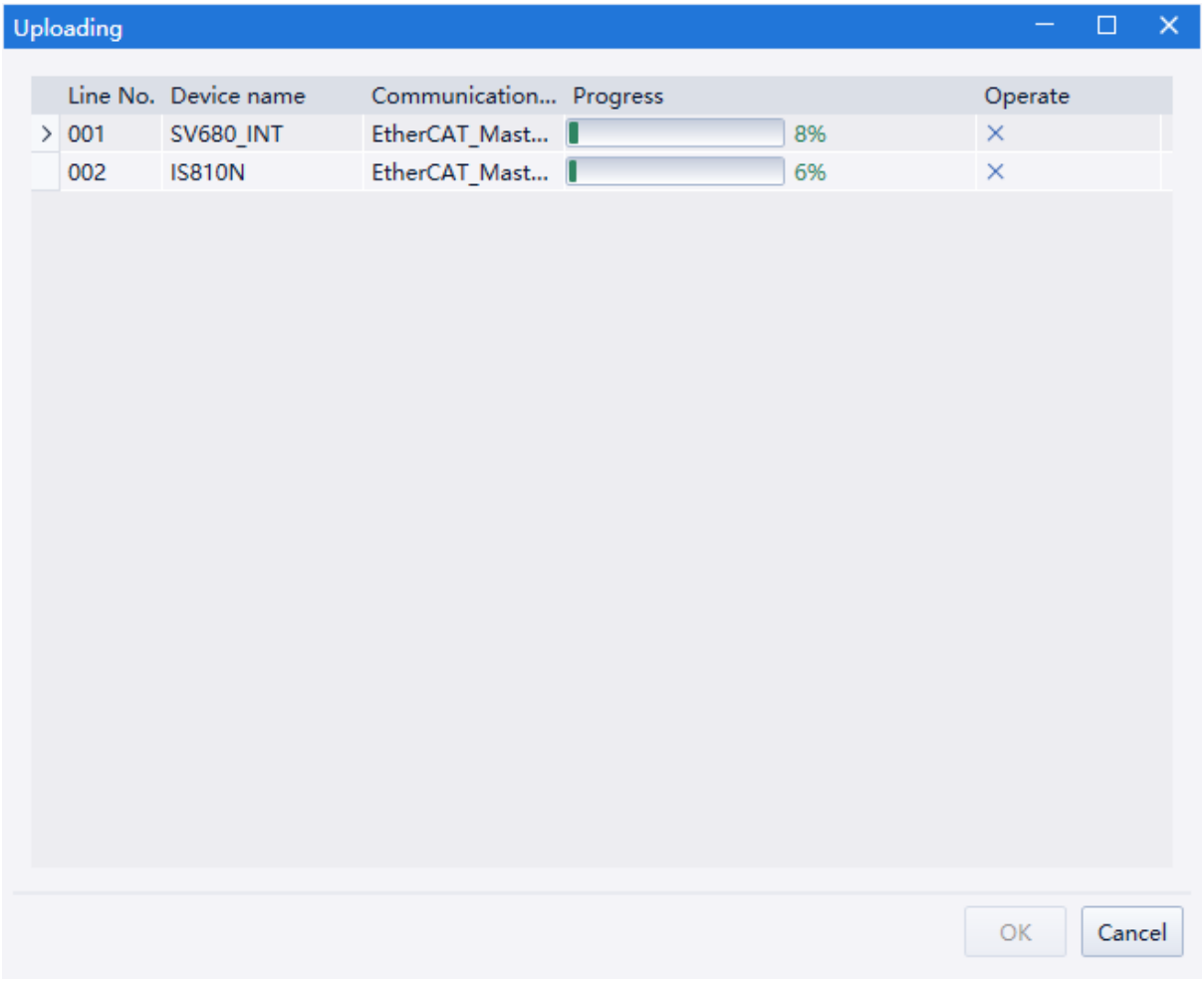
Click "Import", select a file with the .inoDat extension, and select the recipe to be imported into the software.

Machine recipe export

1. Click "Export" and the "Export recipe" window pops up.



2. Select the device to export the recipe, and click "OK" to start exporting.



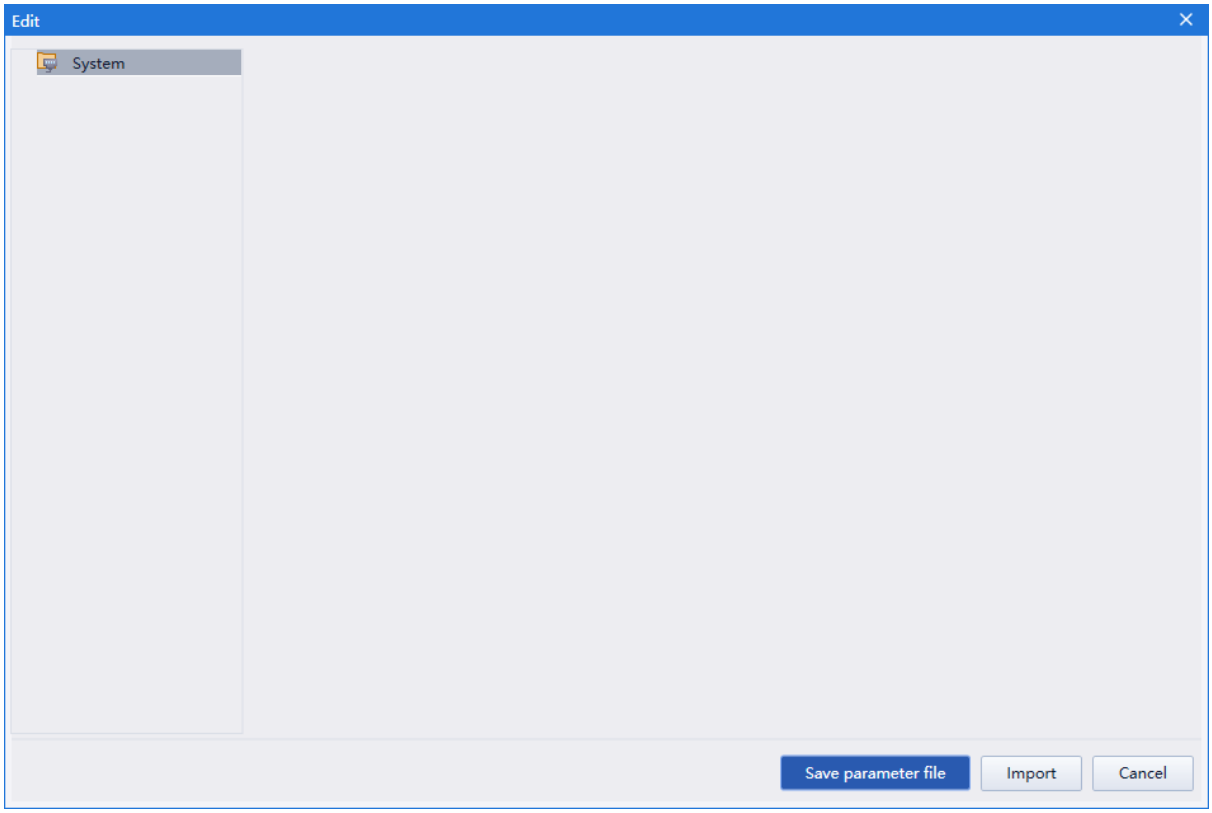
3. After all export progress bars are complete, select the storage location for the recipe.

Offline recipe modification

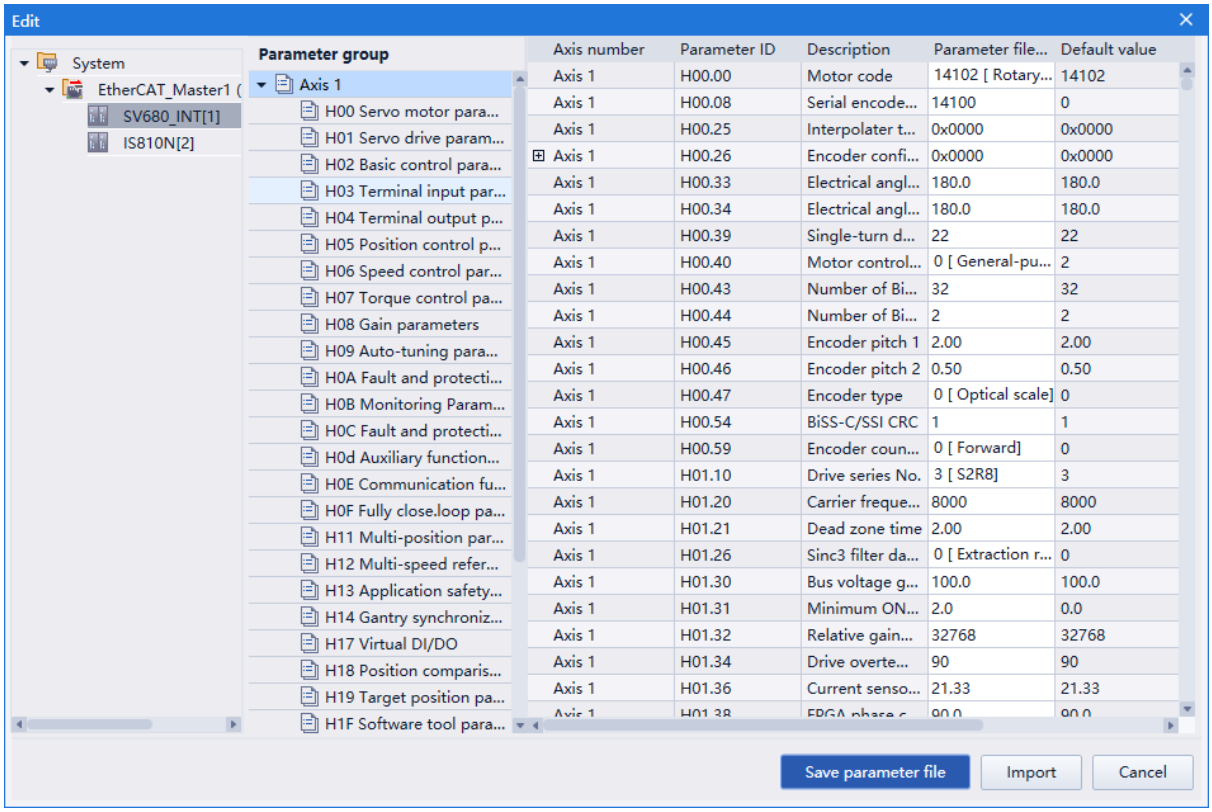
This function is used to modify the *.inoDat type recipe files that have been saved locally.

1. Double-click "EtherCAT_Master (EtherCAT_Master)", select the "Machine parameter management" interface, and click "Offline recipe modification".

2. Click "Import" to import the local recipe, as shown in the figure below.



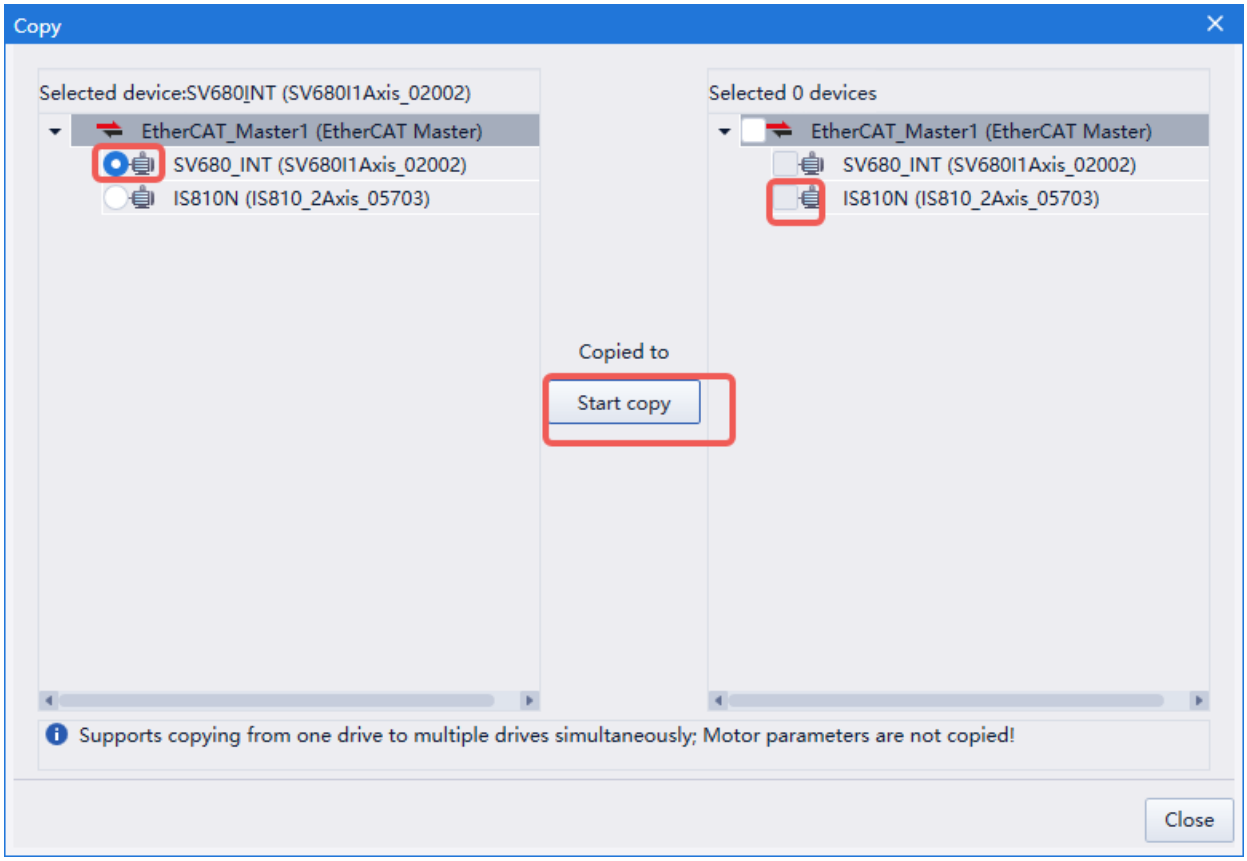
3. After opening the recipe, double-click the corresponding device on the left in the pop-up "Edit recipe" window to open and edit the recipe for the corresponding device.



10.6.3.2.7.2 Machine Parameter Management

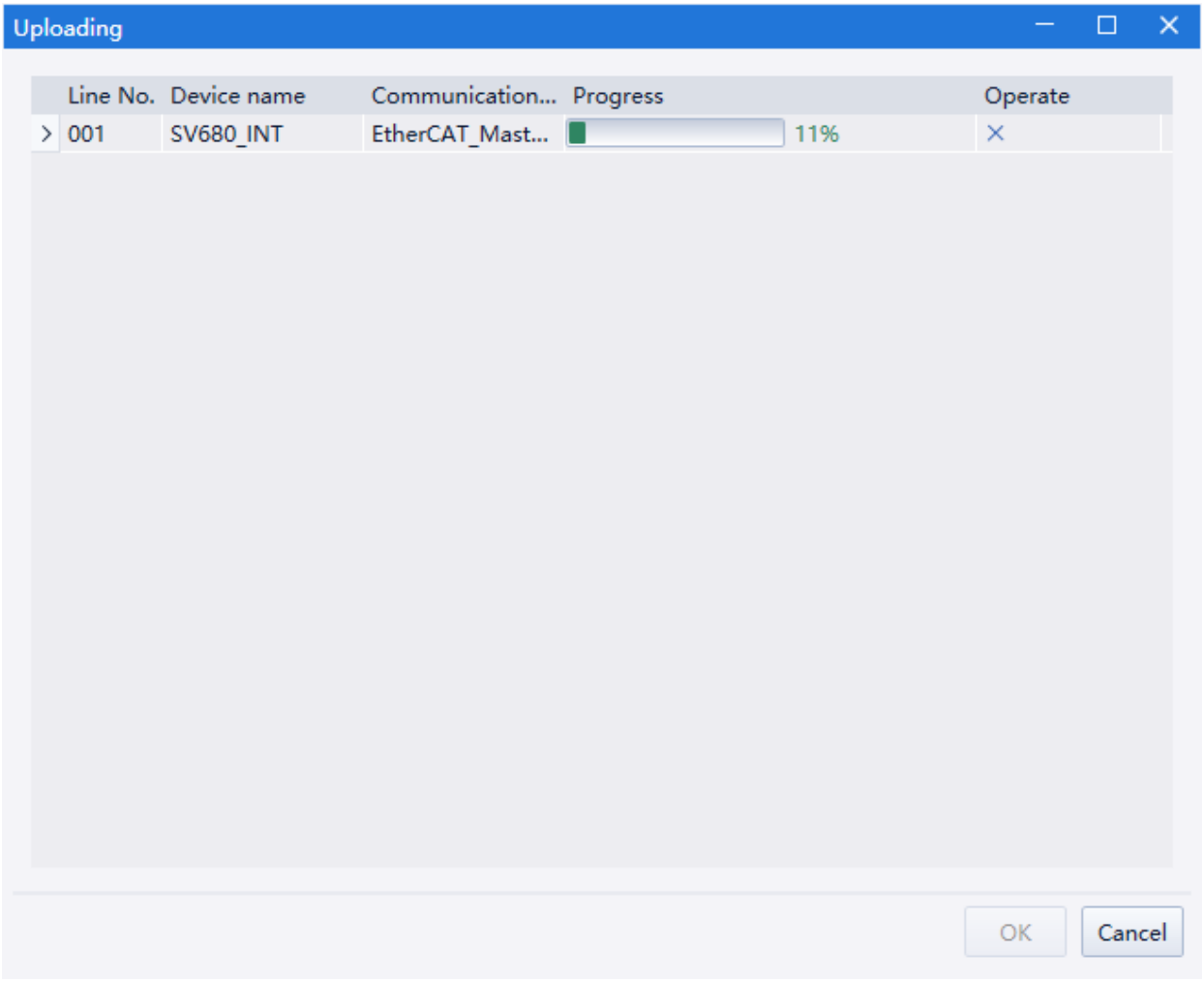
Online parameter copy

This function can copy parameters from one device (excluding motor parameters and non-modifiable parameters) to multiple devices of the same series type simultaneously. As shown in the figure below, the left side is the source device, and the right side is the target device. Click "Start copy" to copy the parameters from the source device to target device.



The copy process is divided into two steps, namely the upload of source device parameters and the copy of target device parameters, as shown in the figure below.

- **Source device parameter upload**

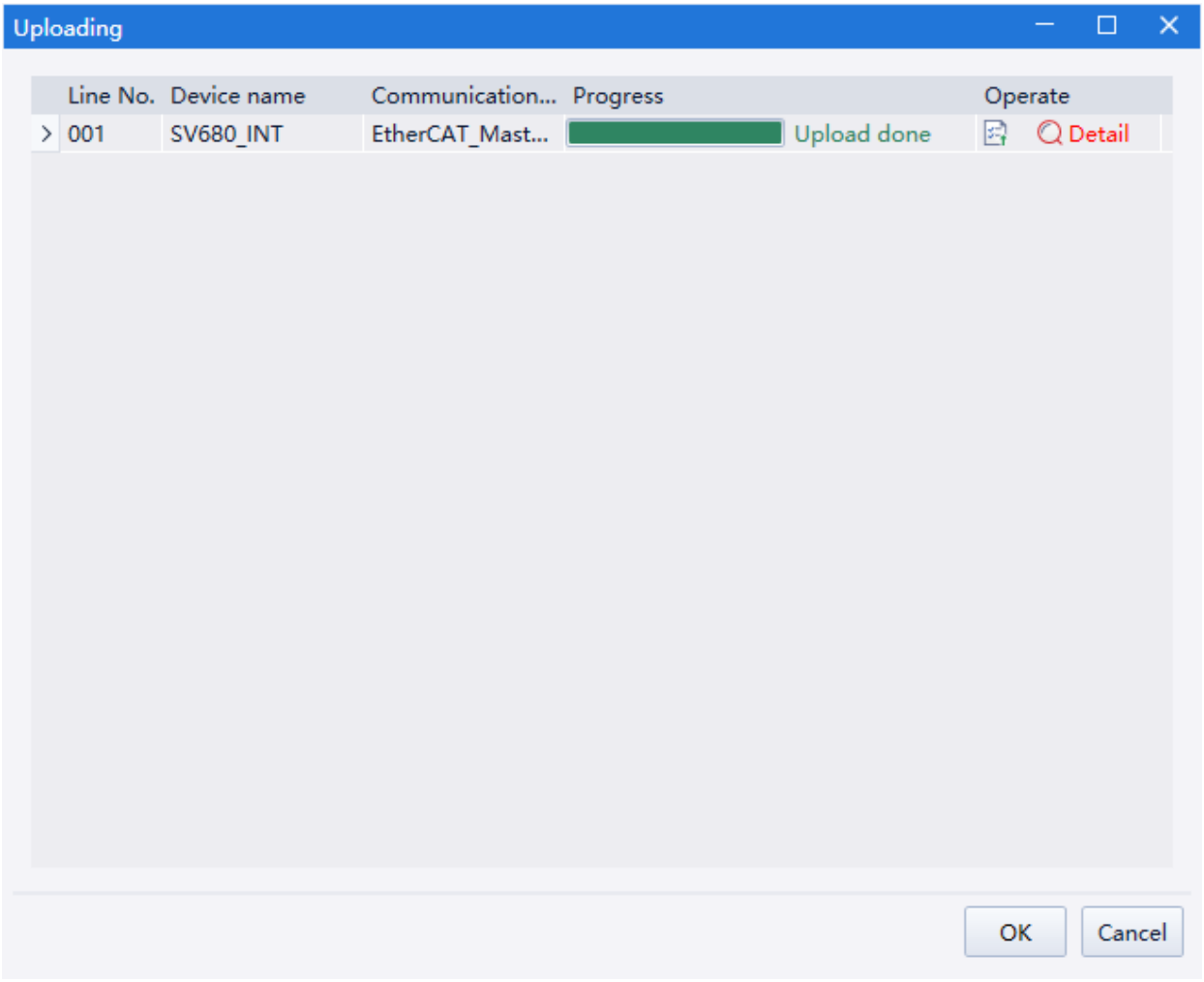


NOTE

Source device parameter upload failures will be indicated in the right "Operation" column. Click to view detailed information.

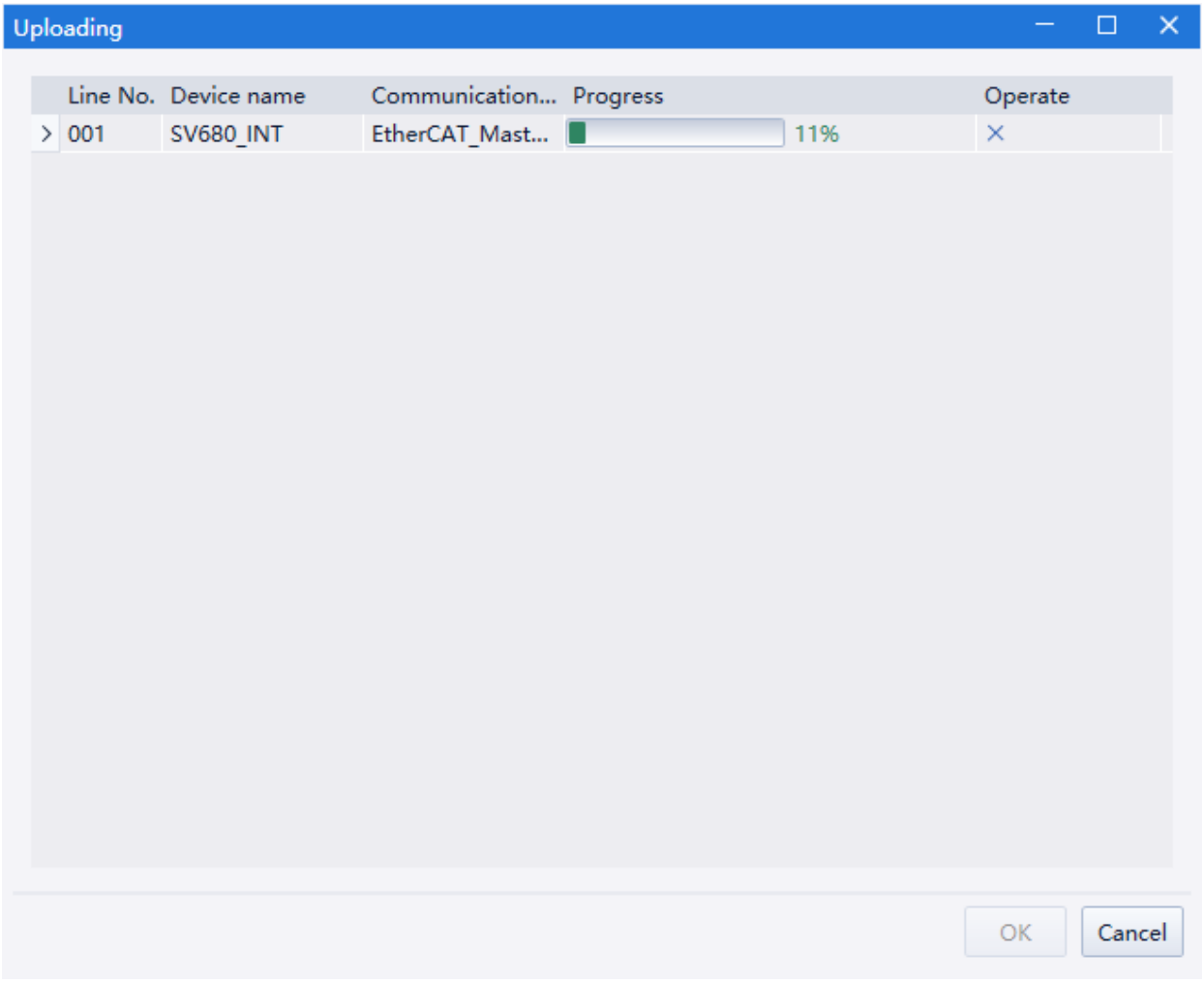
- **Target parameter copy**

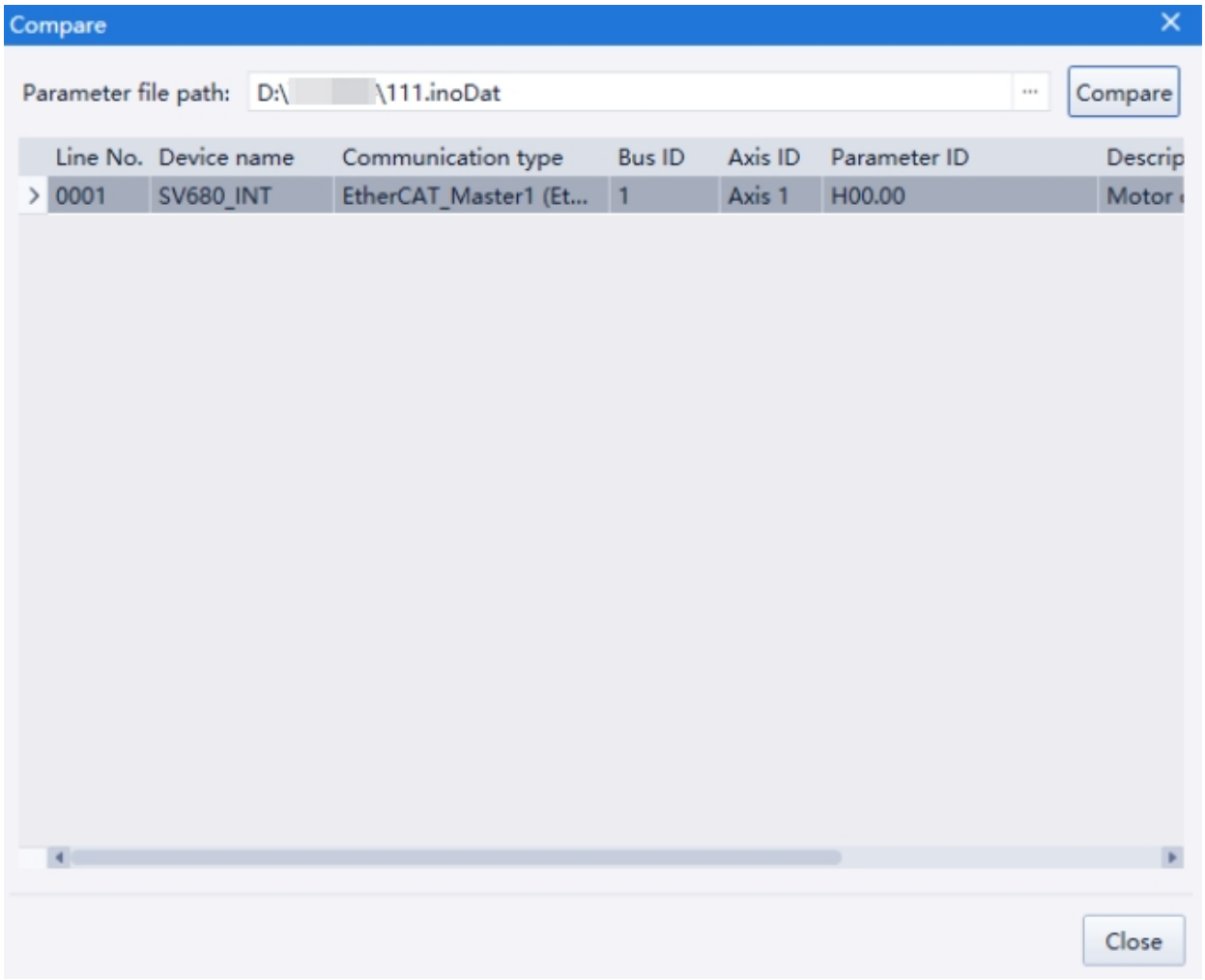
This step copies the parameters uploaded to the source device to the target device.



Online machine comparison

By loading saved recipes, this function compares the data saved in the recipe with the current values of the device, displaying functional codes with differences to help users determine whether to import the recipe into the drive.


Uploading
Line No.
Device name
Communication...
Progress
Operate
001
SV680_INT
EtherCAT_Mast...
11%
OK
Cancel


Compare
Parameter file path:
D:\ \111.inoDat
Compare
Line No.
Device name
Communication type
Bus ID
Axis ID
Parameter ID
Descrip
0001
SV680_INT
EtherCAT_Master1 (Et...
1
Axis 1
H00.00
Motor
Close

10.6.3.2.7.3 Batch Firmware Upgrade

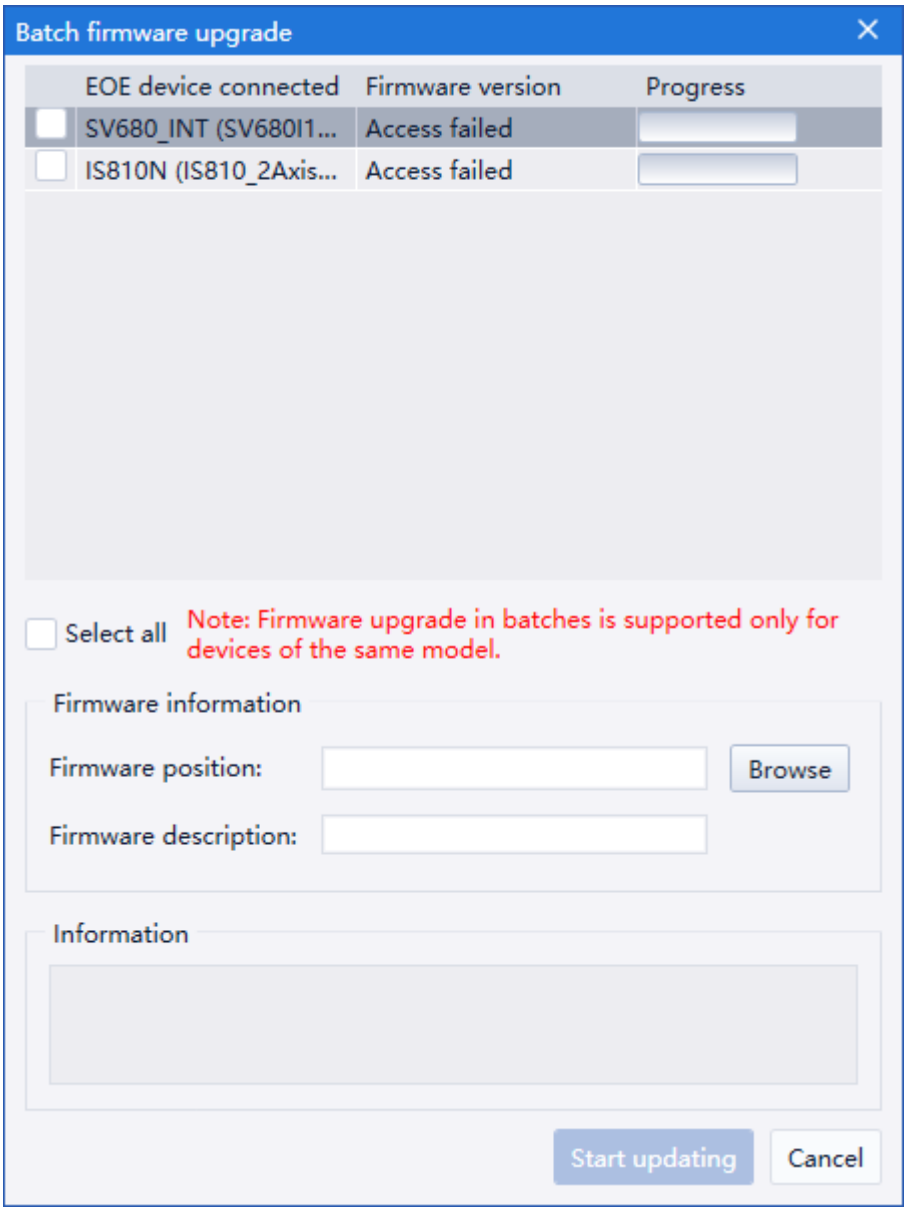
Function description

This function can be used to perform batch firmware burning for multiple devices of the same model.

It is currently not supported by MD800 and MD810 series models.

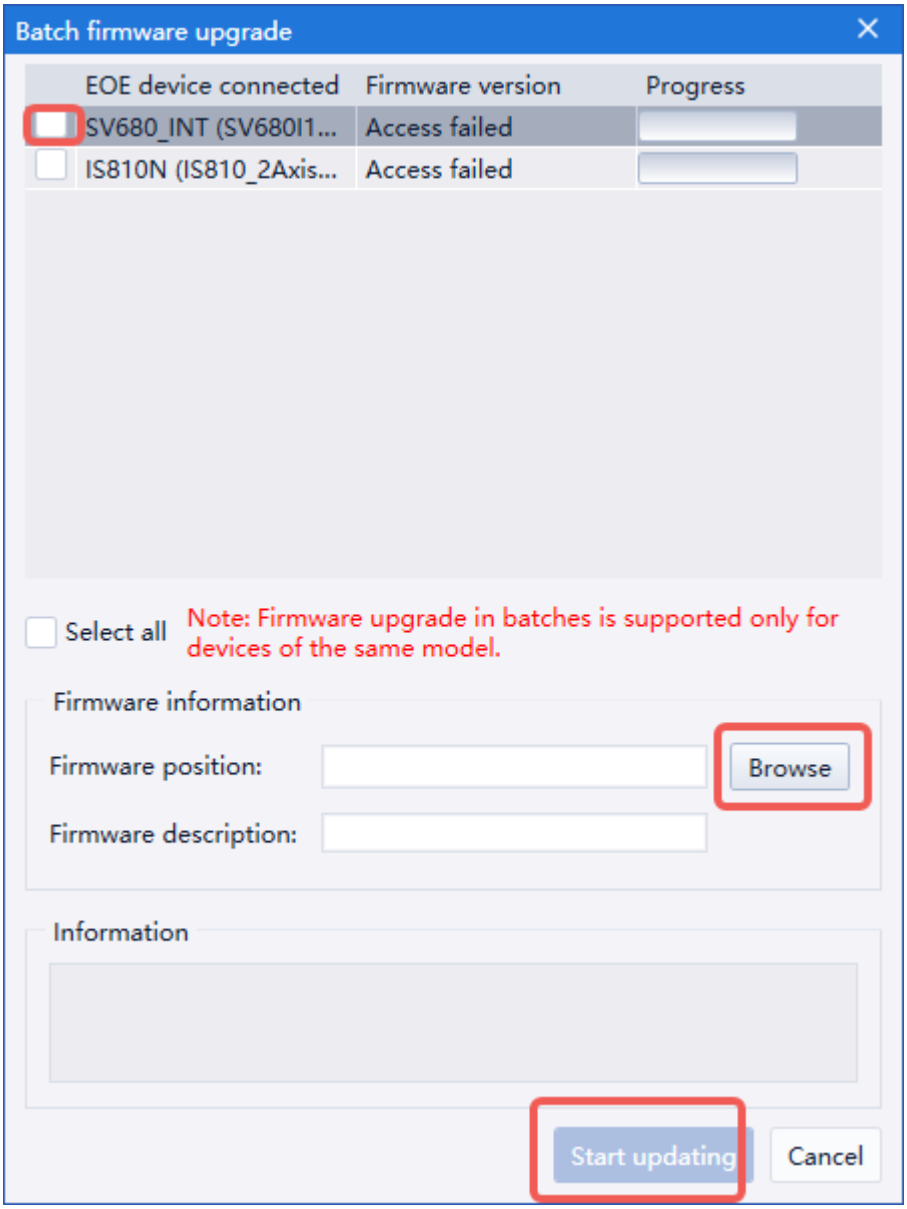
Procedure

1. Double-click "EtherCAT_Master (EtherCAT_Master)", select the "Machine parameter management" interface, and click "Batch firmware burning".
2. The system will automatically read the current firmware versions of each device.



3. Click "Browse" to add the appropriate firmware bin file, check the relevant devices, and click "Start update". The system will automatically read the firmware description information.

> NOTE
>
> Only devices of the same model can be upgraded in batch.



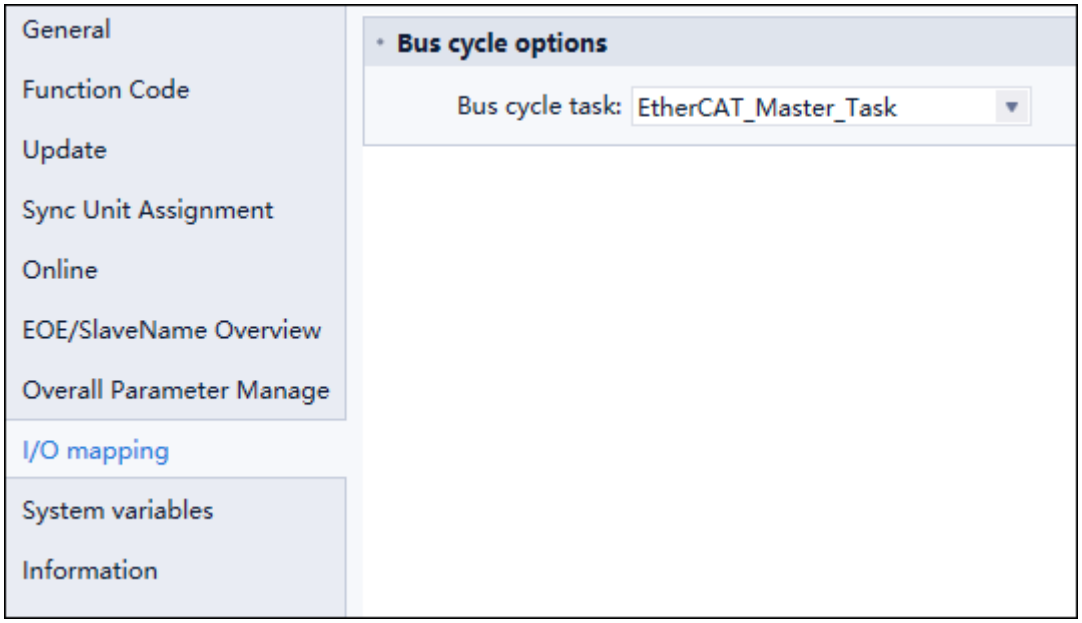
4. After the burn is complete, a success message box will pop up.

NOTE

- After the burn starts, the FA servo integration will automatically disconnect the device.
- After a successful burn, the device needs to be powered on again to connect devices for use.
- If the burn fails, try again. If it fails twice consecutively, contact Inovance technical support.

10.6.3.2.8 I/O Mapping

After adding the corresponding devices under EtherCAT, the I/O mapping is set to a fixed value and cannot be modified.



- Bus cycle task: The system executes bus tasks at a fixed cycle, and the bus cycle task supports modification. The default is EtherCAT_Master Task, and the refresh cycle of I/O mapping for different tasks varies.

10.6.3.2.9 System Variables

In the "System variable" tab of the EtherCAT master, an instance (variable) of type ETC_Master is specified for EtherCAT I/O, supporting control of devices connected to the EtherCAT master via user programs, and displaying the device ID value of this master.

System variable

Variable	Type	Value
EtherCAT_Master1	ETC_Master	
_EC1_Node	ARRAY[1..127] OF ETC_Slave	
_ID_EtherCAT_Master1	DWORD	1376257

The following table describes the members of the ETC_Master type structure.

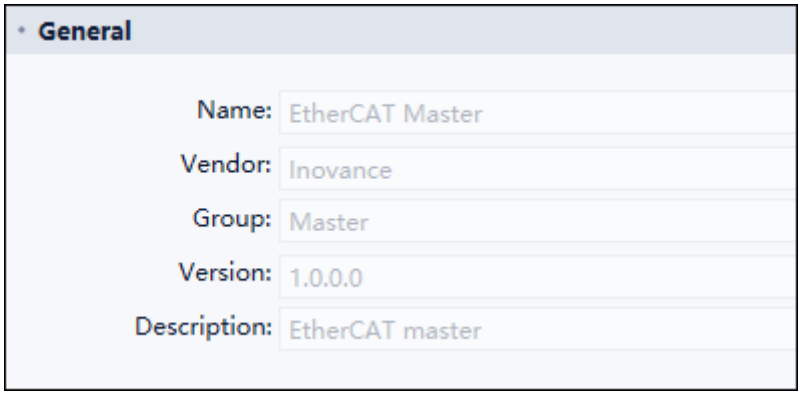
Variable	Data Type	Description
SyncFinished	BOOL	Master synchronization status, indicating whether ECAT master synchronization is completed.
LostFrameCount	UDINT	Number of lost frames.

NOTE

When master synchronization is completed, the variable EtherCAT_Master.SyncFinished is set to TRUE, indicating that the ECAT master has started successfully. This can be used as a condition for executing subsequent ECAT communication-dependent programs.

10.6.3.2.10 Information

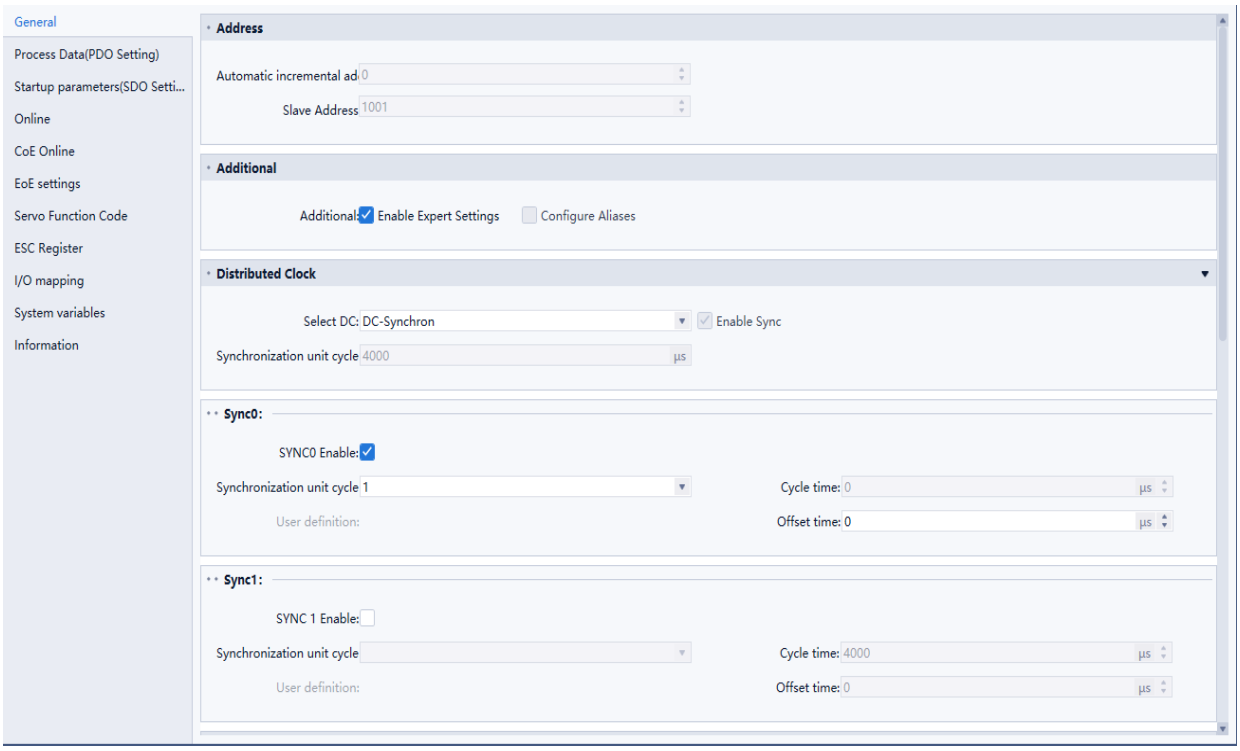
This interface displays basic information about the master, including name, vendor, group, version, and description.



10.6.3.3 Configuring Slave Parameters

10.6.3.3.1 General Settings

Double-click a slave node (servo device) to open the slave configuration window. The "General" interface of the EtherCAT slave provides basic configuration information for the slave, as shown in the figure below.



Address

If the "Auto-configure master/slave" option in the master device editor is not checked, the following options can be configured:

- Automatic incremental address: The automatic incremental address (16-bit) is determined by the physical topology position of the slave in the network. This address is only used when the EtherCAT master starts up. It assigns EtherCAT slave addresses to slaves at corresponding physical topology positions through sequential addressing.

During sequential addressing, the auto-increment address of the slave is determined by its connection position in the physical topology network, represented by a negative number. Sequential addressing sends data frames containing sub-messages. As each data frame passes through a slave device, the auto-increment address in the sub-message automatically increments by 1. When a physical slave receives the data frame, it addresses its own message by checking whether the auto-increment address in the sub-message is 0. This mechanism of updating internal address variables in the slave through data frames is called "sequential addressing" or "auto-increment addressing".

- Slave address: The configuration address (nominal address) of the slave, assigned by the master during startup. This address is independent of the actual position in the network, and the address of the slave is unrelated to its connection order within the segment. After disabling a slave device in hardware configuration, the address of the slave device is set to 0, and it is reassigned according to the current specifications when the slave is re-enabled.

Additional settings

- Enable expert settings: If this option is checked, you can configure distributed clocks, startup checks, timeouts, DC cycle unit control, watchdogs, and other functions.
- Configure aliases: This reflects the status of the "Enable alias" option in the general settings of the EtherCAT master, and cannot be modified here.

Distributed clock

- Enable sync: If this option is checked, it indicates that the "Distributed clock" function is used.
- Synchronization unit cycle (μs): If the "Distributed clock" function is enabled, the synchronization unit cycle value will default to the same as the EtherCAT master's cycle time.
- Select DC: This drop-down menu provides all settings related to distributed clocks in the device description file, generally including free operation, SM event synchronization, DC synchronization, and so on. The setting items are described in the table below:

ID	Item	Description
1	Free operation mode	In this mode, the local control cycle is generated by a local timer interrupt. The cycle time can be set by the master. This is an optional function for the slave.
2	SM event synchronization	A local cycle is triggered when a data input or output event occurs. The master can write the transmission cycle of the process data frame to the slave. The slave can then check whether the cycle time is supported or perform local optimization of the cycle time. The slave can choose to support this function. Synchronization typically occurs with the data output event. If a slave has only input data, synchronization occurs with the data input event.
3	DC synchronization	A local cycle is triggered by the SYNC event. The master must complete the transmission of the data frame before the SYNC event. Therefore, the master clock must also be synchronized to the reference clock.

ID	Item	Description
4	Sync0 and Sync1	Slave synchronization signals 0 and 1. The distributed clock control unit of the slave (an internal function of the EtherCAT-specific communication chip) can generate two synchronization signals, Sync0 and Sync1. These signals provide interrupt flags to the application layer of the slave or directly trigger the update of output data.
5	Sync0 enablement and Sync1 enablement	Check these options to enable the Sync0 and Sync1 synchronization signals. • Synchronization unit cycle: If this option is checked, slave synchronization cycle time = master cycle time × selected coefficient. • The "Cycle time (μs)" item indicates the current set cycle time of the slave.
6	User-defined DC mode	If this option is selected in the "Select DC" drop-down menu, user-defined synchronization cycle time in microseconds can be entered in the "Offset time" fields for Sync0 and Sync1.

Diagnostics

Visible only in online mode.

Current status: Displays the current status of the slave's communication state machine. Possible values include: Initialization, Pre-operational, Safe-operational, Operational, Boot (not supported by Inovance servo). If the current status is operational, it indicates that the slave configuration has been correctly completed.

Option

Option: If a slave device is defined as "optional", no fault messages will be generated to prevent the device from not being present in the bus system. To enable this option, a station address must be stored in the slave device, so the "station alias" address must be defined and written into the EEPROM. This option is only valid when the "Auto-configure master/slave" option is checked in the EtherCAT master general settings and the EtherCAT slave supports this function.

Start check

By default, the system automatically checks the identity of the supplier and the product ID at startup based on the current configuration. You can also check the "Check version number" option to check the specified version. If a mismatch is detected, the bus will stop running and no further actions will be performed. This setting is used to prevent the download of incorrect configurations.

Timeout

By default, the following operations are not defined as timeouts. To observe whether these operations exceed a specific time, you can set them here.

- SDO access: Sets the timeout time for SDO (Service Data Object) access from the EtherCAT master protocol to the slave.
- I -> P: Sets the timeout time for the transition of the slave communication state machine from "Initialization" to "Pre-operational".

- P -> S / S-> O: Sets the timeout time for the transition of the slave communication state machine from "Pre-operational" to "Safe-operational", or from "Safe-operational" to "Operational".
- Detect port -> Open port: After the slave is connected, sets the timeout time for detecting and opening the port.

DC cycle unit control

This option is assigned to the local microprocessor to select the options defined by the distributed clock function. The control function is implemented in register 0x980 of the EtherCAT slave, with optional settings being: Periodic unit, Latch 0, and Latch 1.

Watchdog

- Watchdog multiplier: Sets the multiplier of the watchdog timer to determine its minimum timing increment. The default value is 2498. The minimum timing increment of the watchdog timer is 100 μs.
- PDI watchdog: After enabling the PDI watchdog, if the PDI (Physical Device Interface) communication time of the EtherCAT slave exceeds the set value, the watchdog will be triggered.
- PD watchdog: Sets the PD watchdog.

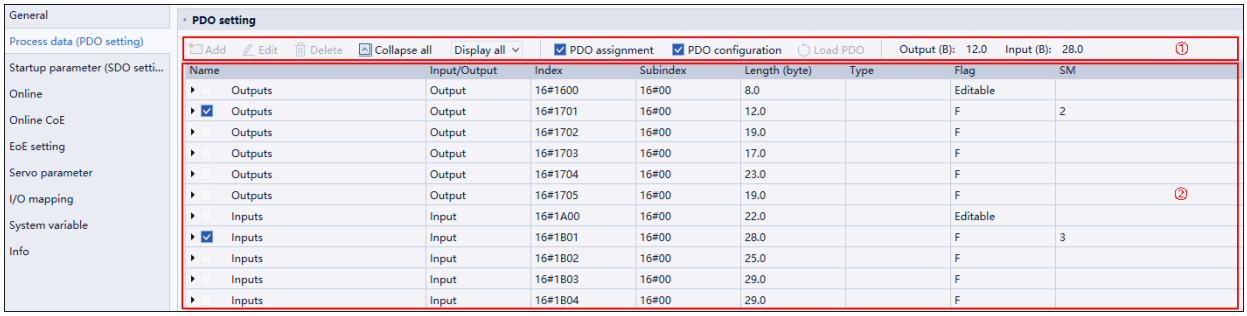
Slave alias

The settings of slave alias are only valid when "Option" is checked and the slave device supports alias addresses (defined by the device description file). If the corresponding alias address of the slave is configured, there is no need to change the configuration in the user program when adjusting its position in the physical topology network, and the slave can still operate normally. Note: After modifying the alias address of the slave, the user program must be re-downloaded for the alias to take effect. Additionally, some slaves must be powered off and restarted for the alias to take effect. For details, see the user guides of relevant slave devices.

- Disable: If this option is checked, the slave will not detect the alias address.
- Configure site alias (ADO 0x0012): When the slave device supports alias addresses and "Option" is checked, writing the alias address is supported only in online operation mode.
- Write EEPROM: This command is visible only in online mode. It allows the defined address to be written to the EEPROM of the slave. If the slave does not support the written address, this command is invalid and the slave cannot operate in site alias form.
- Actual address: This field is visible only in online mode. The actual address of the slave is displayed, which can be used to check whether the "Write EEPROM" command is successful.
- Display device identification (ADO 0x0134): Reserved.
- Data word (2 bytes): Reserved.

10.6.3.3.2 Process Data (PDO Configuration)

In actual automated control systems, there are normally two forms of data exchange between application programs: time-critical and non-time-critical. Time-critical indicates that specific actions must be completed within a designated time window; if not, it may lead to control failure. The process of periodically sending time-critical data is called periodic process data communication. Non-time-critical data can be sent non-periodically using MailBox data communication in EtherCAT.

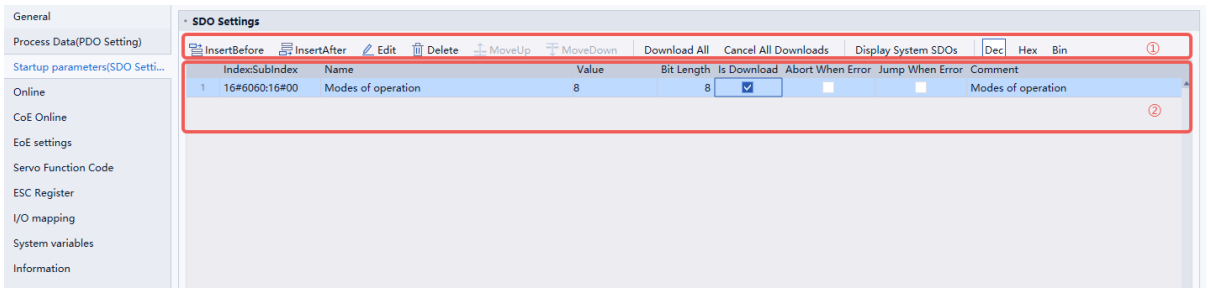


No.	Function	Description
①	Toolbar	Supports adding, editing, deleting, and filtering PDO entries.
②	PDO display area	Displays detailed information about PDO.

- Adding: Adds PDO items based on the properties of the current PDO group (editable properties only). Users can choose to add one set or multiple sets of data.
- Edit: Edits the selected PDO item based on the properties of the current PDO group (editable properties only).
- Delete: Deletes the selected PDO item based on the properties of the current PDO group (editable properties only).
- Collapse all: Collapses all entries in the PDO display area with one click.
- Filter box: Filters the PDO display area to display all, display output PDO entries, or display input PDO entries.
- PDO allocation/PDO configuration: In the configuration of the GR10_4ADE module, PDO allocation and configuration refer to assigning PDO objects to the corresponding input/output channels and setting PDO parameters to ensure correct communication between the module and the CANopen network.
- Load PDO: When the slave is in operation, the current PDO group data of the slave can be uploaded to the backend.

10.6.3.3.3 Startup Parameters (SDO Configuration)

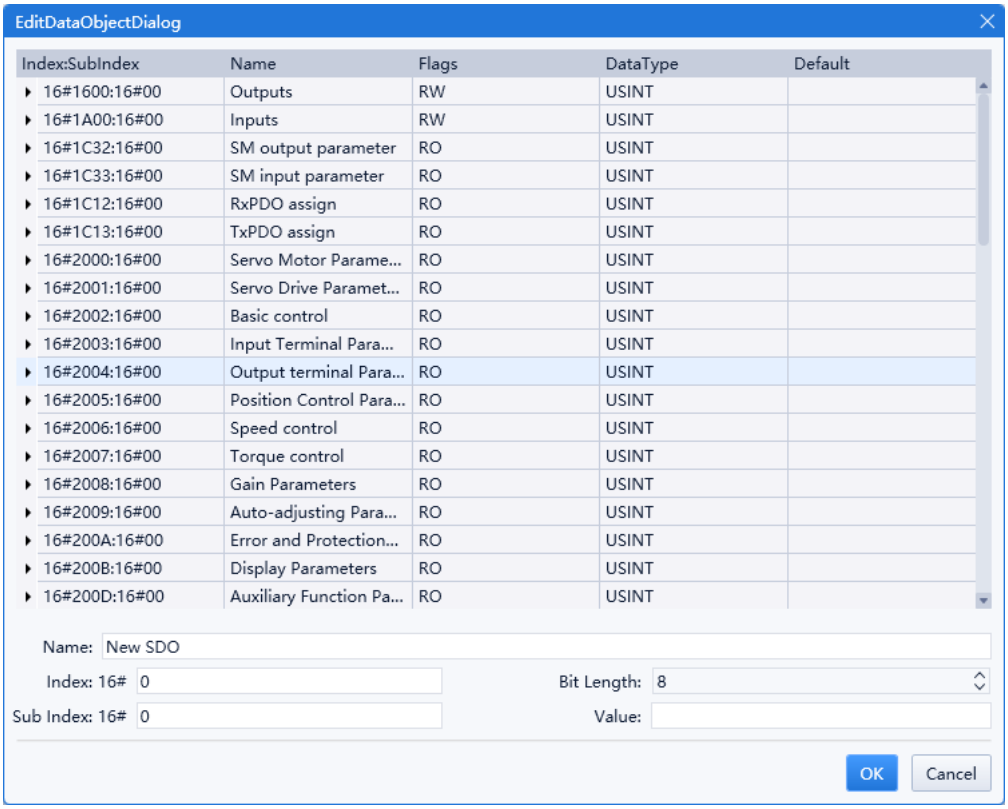
Startup parameters can be transmitted to the slave through SDO (Service Data Object) when the system starts. They include basic configuration parameters required for the slave startup, as shown in the following figure.



No.	Function	Description
①	Toolbar	Supports inserting, editing, deleting, downloading, canceling download, and setting display mode for SDO entries.

No.	Function	Description
②	SDO entry display area	Displays detailed information of SDO entries.

- Insert before: Adds an SDO entry to the startup parameter list before the currently selected entry. The object dictionary selection box that pops up is shown below. After selecting the SDO to add, you can modify its name, index, sub-index, bit length, and value in the edit bars below, then click "OK" to create a new startup parameter.

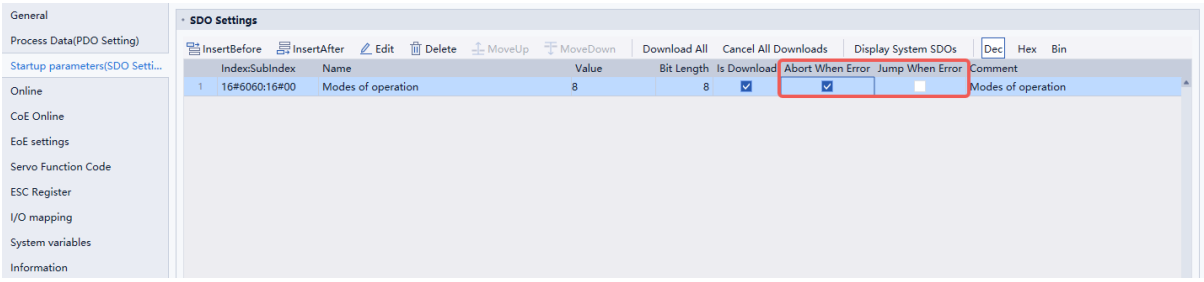


- Insert after: Adds an SDO entry to the startup parameter list after the currently selected entry.
- Edit: Edits the parameters of the currently selected entry. Read-only parameters cannot be edited, such as system parameters.
- Delete: Deletes the currently selected entry.
- Move up/Move down: The order of the SDO list (from top to bottom) represents the order in which startup parameters are transmitted to the module. The "Move up" and "Move down" buttons can change the transmission order.
- Download all (SDO)/Cancel all downloads (SDO): Click "Cancel all downloads" to deselect the download property of all entries. Click "Download all" to select the download property of all entries. You can also select the download property of some entries for partial download.
- Display system parameters: Displays the parameters added to the SDO after PDO allocation and PDO configuration are checked, for comparison only.
- Value display mode: The values in the "Value" column of the SDO entry display area can be displayed in decimal, hexadecimal, or binary.

Note: When a fault occurs during the SDO transmission process, it can be resolved through the following options:

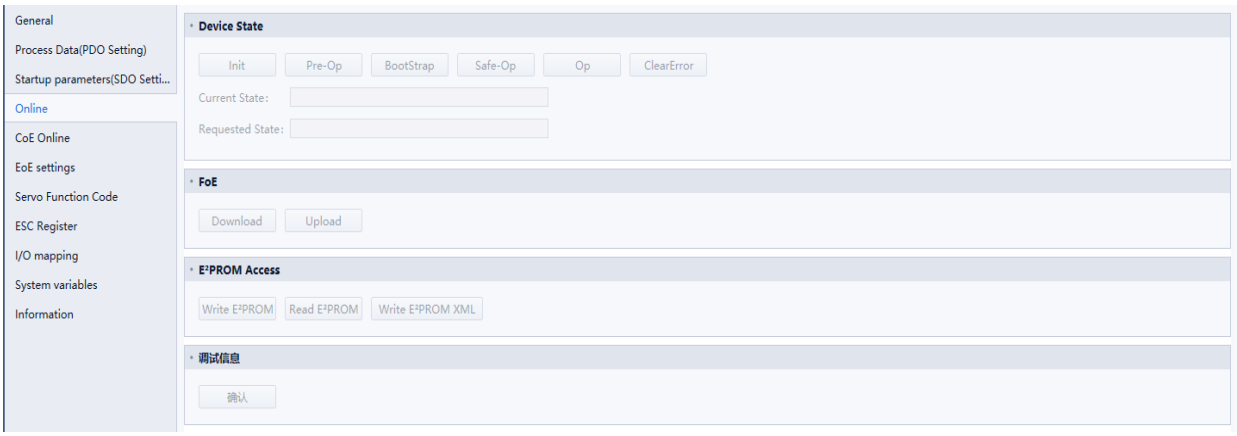
- Exit on error: If this option is selected, the SDO transmission will exit if an error is detected.

- Skip on error: If this option is selected, the SDO transmission will skip to the next line and continue if an error is detected.



10.6.3.3.4 Online

The slave online configuration editor can only be used after logging into the device. The related functions and interface are as follows.



- **Device state**

The EtherCAT state machine coordinates the status relationships between the master and slave applications during initialization and operation, as shown in the following figure. The EtherCAT operation transition sequence is: Initialization -> Pre-operational -> Safe-operational -> Operational. The following table introduces each status transition relationship.

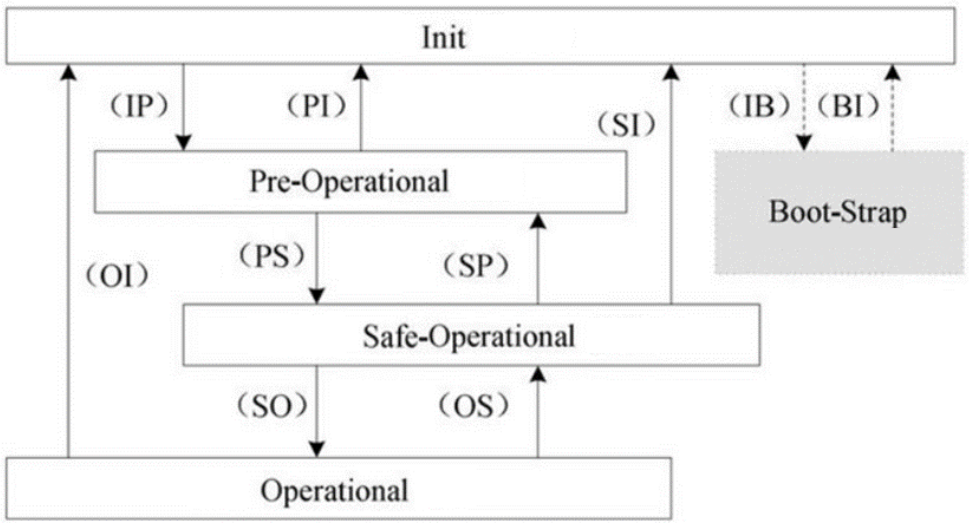


Figure 10-6-3-3-4-1 EtherCAT status transition relationship

No.	Status Transition	Local Management Service
①	IP	Starts mailbox communication.

No.	Status Transition	Local Management Service
②	PI	Stops mailbox communication.
③	PS	Starts input updating.
④	SP	Stops input updating.
⑤	SO	Starts output updating.
⑥	OS	Stops output updating.
⑦	OP	Stops output/input updating.
⑧	SI	Stops output updating and mailbox communication.
⑨	OI	Stops output/input updating and mailbox communication.
⑩	IB	Starts boot mode.
⑪	BI	Restart the device

- Clear error: Click this button to clear the error in the current status.
- Current status: Displays the current operating status of the device.
- Request status: Displays the currently requested status. When selecting another status and the transition has not been completed, this bar will display the requested but uncompleted status.

- **FoE**

  Used to transmit firmware files to the slave for firmware upgrades.

  Click the "Boot status" button to transit the slave to boot mode. Click "Download"/"Upload" to download or upload firmware files and a dialog box pops up. Enter a string and password to execute the file transfer. This information is provided by the slave device and recorded in the slave's data table. Avoid power outages or status switches during firmware upgrades.

- **E2PROM interface**

  The configuration of the slave can be read from or written to the E2PROM.

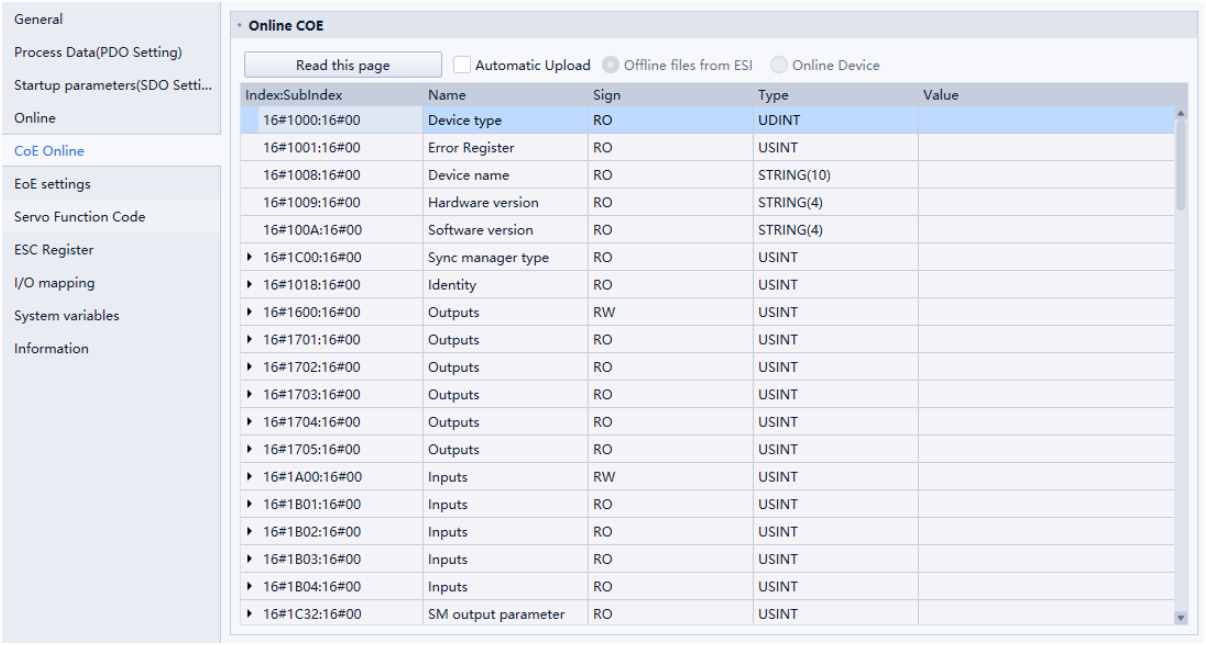
  Similar to firmware file transfer, this operation will also pop up a dialog box to open or save the file. Users can use the "Write E2PROMXML" command to directly write the slave configuration from an XML file to the device. This command is only valid if there is configuration data in the XML file (in the <configuration data> section).

- **Commissioning**

  Click "Clear" to clear all fault codes from the slave.

10.6.3.3.5 Online CoE

The values of online CoE can only be read when the bus is operating normally and the PLC is logged in, as shown in the following figure.



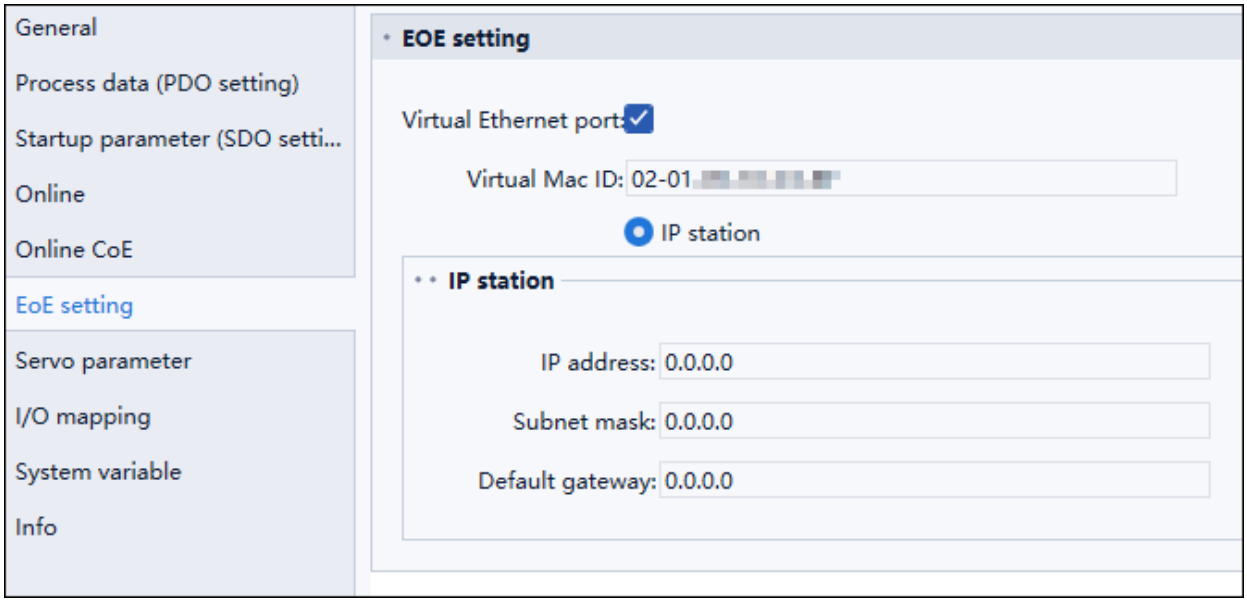
- The interface displays the index, item, flag, type, and value information of the data.
- The flags include RO (read-only) and RW (read-write).
- Click "Read this page" to read the values of the parameters on the current page.
- Click "Auto upload" to automatically read the values of the parameters from the device.
- Select "Offline files from ESI" or "Online device" to switch the source of the data in the list.

10.6.3.3.6 EoE Settings

This function is currently not supported for the MD800 and MD810 series products.

EtherNET over EtherCAT (EoE) allows any EtherNET device to be connected to EtherCAT via a conversion terminal without affecting the real-time performance of EtherCAT. Similar to well-known Internet protocols, such as TCP/IP, VPN, PPPoE (DSL), Ethernet frames are transmitted via the EtherCAT protocol.

- This setting allows standard network devices to be connected to end devices (such as printers, PCs) via a switch by enabling "Virtual Ethernet port" and set the "IP station".



For special slave devices that support EoE, communication can be set. The following dialog box will only pop up when the device supports EoE.

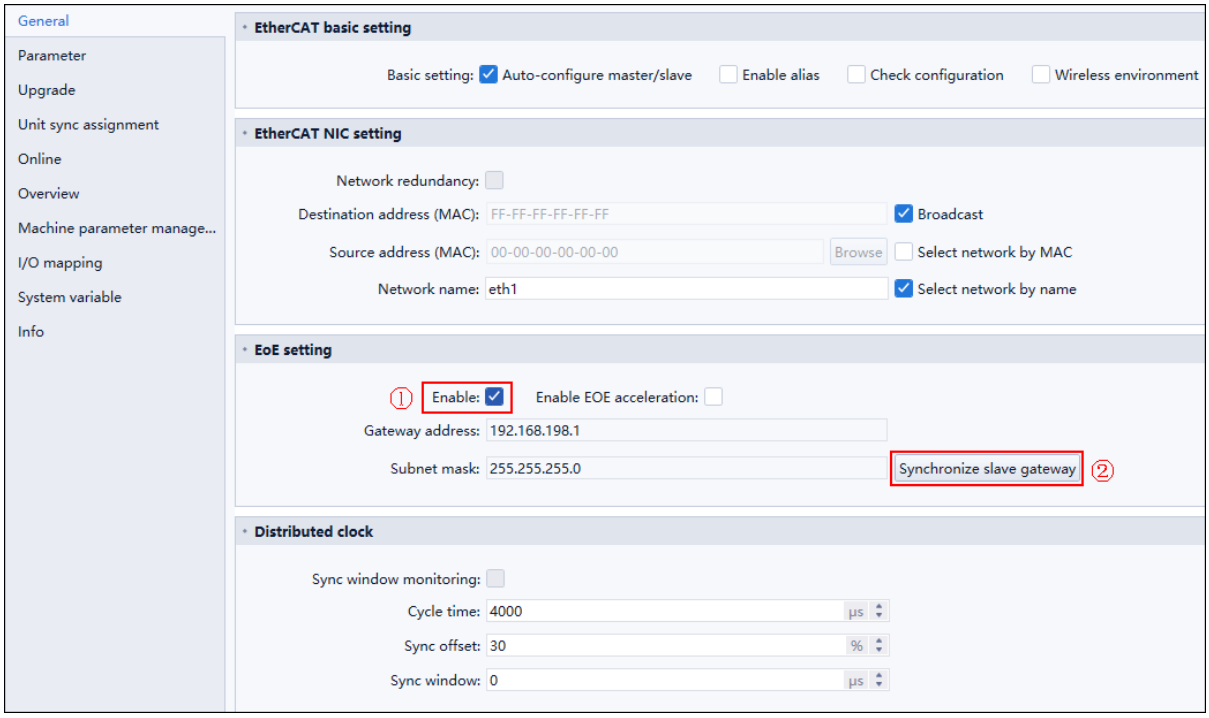
- Virtual Ethernet port: Enables the EoE function of the slave device. If this option is checked, a special "Virtual Mac ID" must be defined.
- Virtual MacID: Automatically assigned by the system, which can be manually modified by users. If a MacID duplication occurs, a compilation fault will occur.
- IP station: Configures the device as an IP interface. When this option is selected, the Ethernet communication parameters must be set.

  Ethernet communication parameters must be set based on the parameters of the virtual Ethernet adapter. The entries for IP address, subnet mask, and default gateway are each allocated four bytes to identify the slave in the network and support modifications to the default settings.

NOTE

The IP address must be in the same subnet as the virtual Ethernet adapter. For example, if the Ethernet adapter's address is 192.168.1.1 and the subnet mask is 255.255.255.0, then the IP address must be within the range of 192.168.1.2 to 192.168.1.254.

- After configuring the EoE settings of the slave, double-click the master node "EtherCAT_Master (EtherCAT Master)", check "Enable" in the "EoE settings" section of the general settings, and then click "Synchronize slave gateway" to complete the EtherCAT configuration, as shown in the figure below:



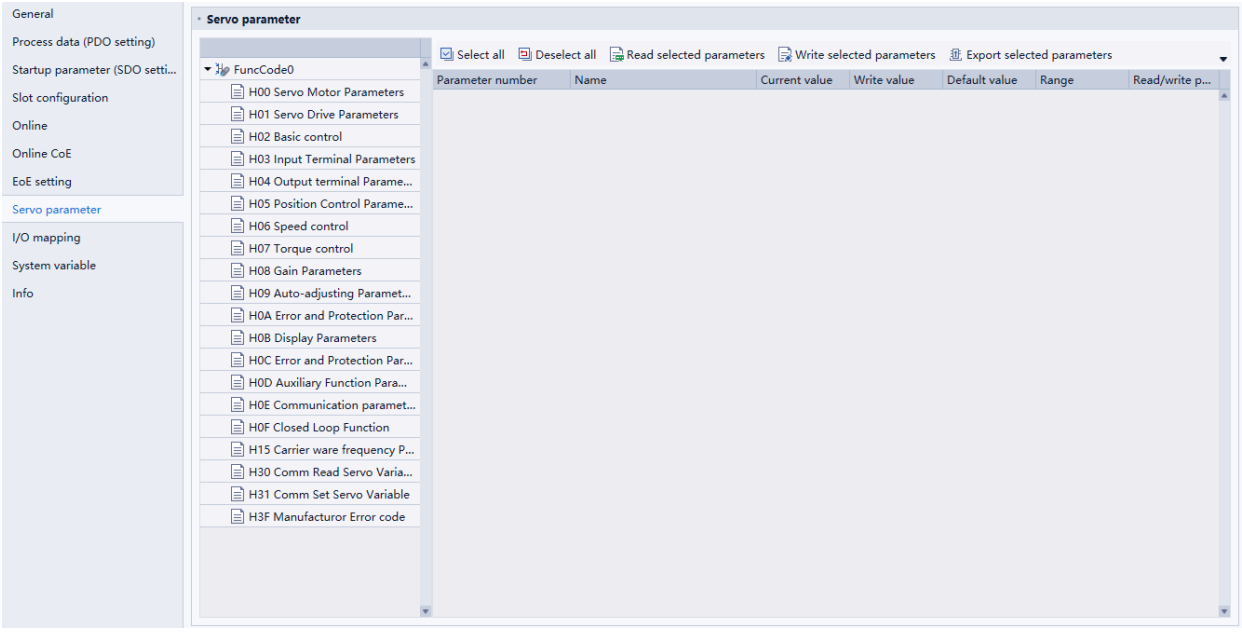
NOTE

The gateway address cannot be in the same subnet as the PLC's IP address. If they are in the same subnet, it is recommended to modify it manually.

Only devices that support the EoE function can be configured.

10.6.3.3.7 Servo Parameters

Servo function codes refer to the manufacturer parameters defined for Inovance servo products. Under the "Servo parameter" tab of slave configuration, the manufacturer parameters of multiple product can be read and written in batches, imported, and exported, facilitating commissioning and maintenance.



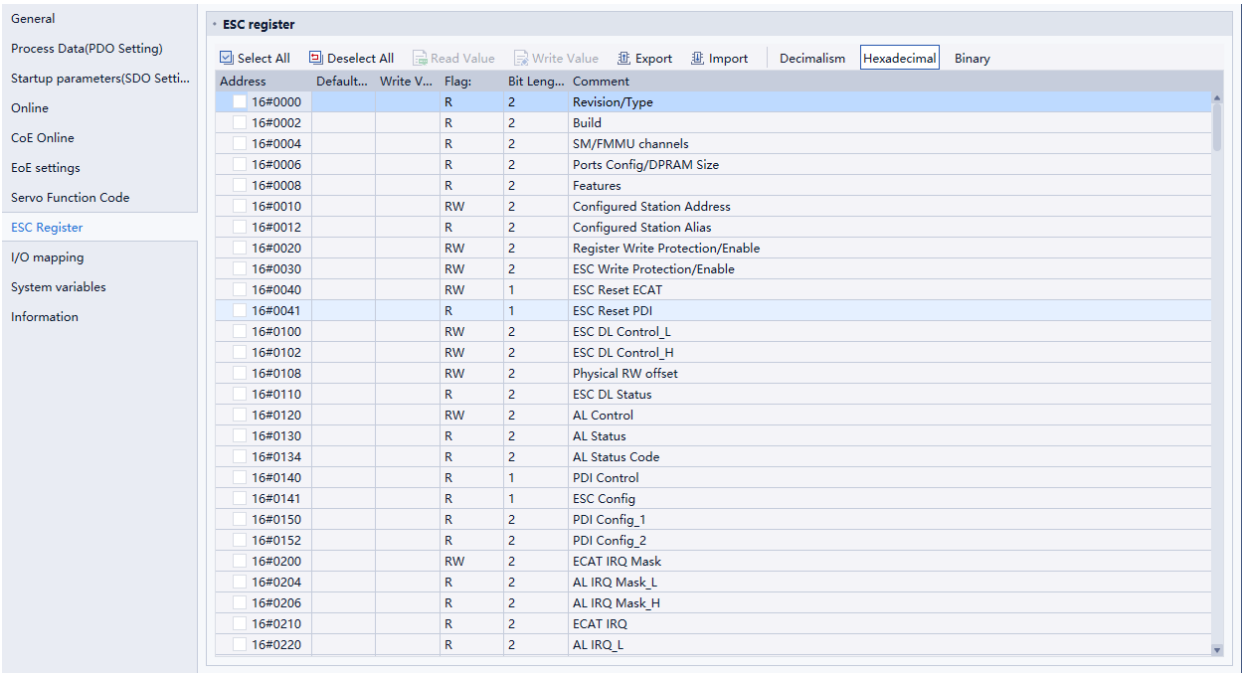
- Select all: Selects all slaves and axes, as well as the servo function codes under the axes.
- Deselect all: Cancels all selected items.

- Read selected parameters: In the servo operation status, reads the servo function codes with RO, RW, or R properties.
- Write selected parameters: In the servo operation status, writes the servo function codes with W or RW properties.
- Export selected parameters: Exports the selected servo parameters in an Excel file.
- Import parameters: Imports the parameters of all slaves from an external Excel file. If the configuration does not match the file, a fault will be reported.

10.6.3.3.8 ESC Register

The ESC register is used to read the values of the ESC chip register addresses in advanced commissioning, as shown in the figure below. For descriptions of each function and parameter, see the table below.

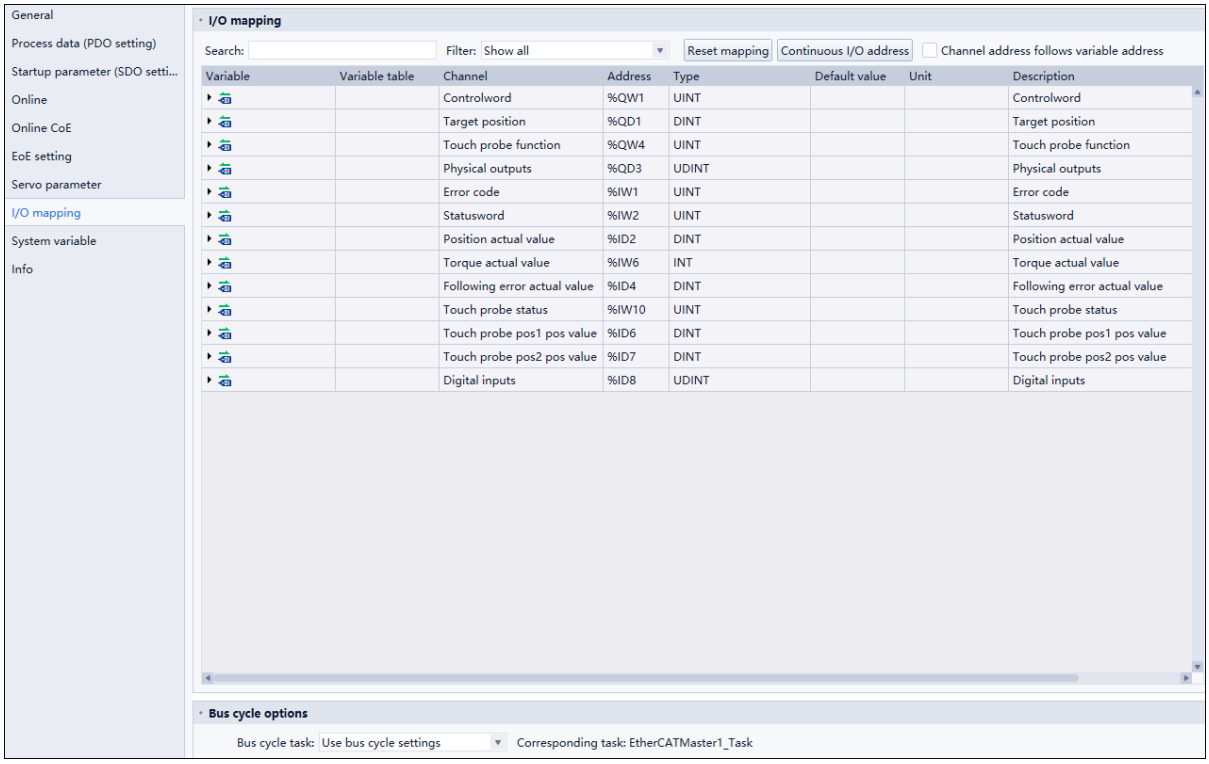
Currently, only the SV680N series device supports this feature.



Parameter	Description
Select all	Selects all parameters in the list.
Deselect all	Deselects all parameters in the list.
Read value	Reads the values of the selected items in operation status.
Writing a Value	Writes the values with write property in operation status.
Export	Exports the selected items as an .xml file and save it to the local disk.
Import	Imports an .xml file from the local disk that meets the format requirements.
Display format	Switches the display format to decimal, hexadecimal, or binary.

10.6.3.3.9 I/O Mapping

The "I/O mapping" interface is shown in the figure below, and detailed explanations of the parameters are provided in the table below.



Parameter	Description
Search	Searches for the corresponding I/O mapping channel based on keywords.
Filter	Displays the corresponding variables based on the filtering option.
Reset mapping	Clears the mapped variable information.
Continuous I/O address	Clears the device's I/O address and reassign it to a continuous address.

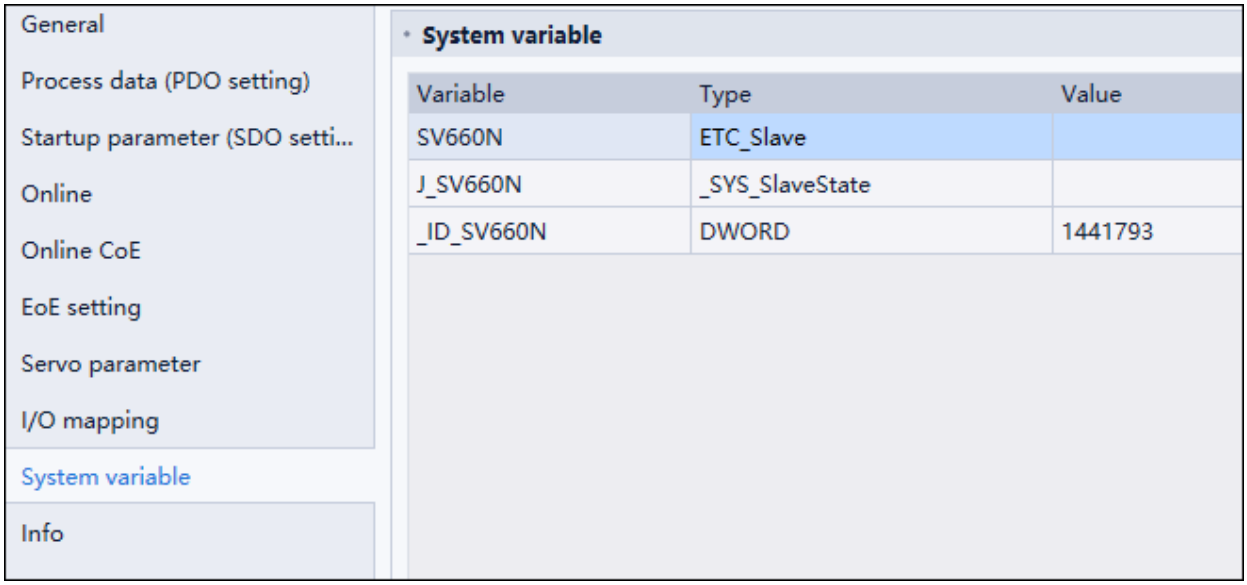
Parameter	Description
Channel address follows variable address	This option is not checked by default. If "Channel address follows variable address" is not selected: The I/O mapping channel address is fixed, and the variable type must strictly match the channel type. This mode only supports binding standard data types. After the variable is bound to the channel, the channel address will be automatically assigned to the variable. If "Channel address follows variable address" is selected: Binding can be completed when the channel byte size matches the variable byte size. This mode supports binding standard data types, structures, unions, and enumerations as custom types. After the variable is bound to the channel, the I/O mapping channel address will be removed, and the channel address will automatically follow the variable address.

Parameter	Description
Variable	Configures the variables for I/O mapping, which can be performed in the following two ways: • Mapping the I/O mapping channel to an existing variable (standard data type) Double-click the cell in the "Variable" column, and click "...". Select the existing variable to be mapped in the pop-up "Input assistant" interface and click "OK". • Mapping the I/O mapping channel to an existing variable (struct/union/array) 1. Select "Channel address follows variable address". 2. Double-click the cell in the "Variable" column, and click "...". Select the user-defined instance to be mapped in the pop-up "Input assistant" interface and click "OK". 3. When binding a user-defined type to the first channel, select the binding method in the dialog box. ▪ Bind the structure directly to this channel: After selected, the variable is bound only to this channel. ▪ Bind the elements in the structure sequentially to all channels of this device: After selected, the variables in the struct are automatically bound to all channels of the device in order; when the struct variable has multiple levels of nesting, only the first-level variables are bound sequentially. • Mapping the I/O mapping channel to a new variable Double-click the cell in the "Variable" column, enter the name of the new variable, and press the Enter key. The new variable will be synchronized to the default variable table. The name of the new variable must comply with IEC programming standards, otherwise it will result in a compilation fault.
Variable table	Identifies the type of variable table for I/O mapping variables.
Channel	Indicates the bit number of the current I/O mapping channel.

Parameter	Description
Address	Indicates the address assigned to the current I/O mapping variable. To customize the I/O address, double-click the cell in the "Address" column to modify it.
Type	Indicates the data type of the current I/O mapping variable.
Default value	Indicates the default value of the current I/O mapping variable.
Unit	Indicates the unit of the current I/O mapping variable.
Description	Indicates the description information of the current I/O mapping variable.
Prepared value	Indicates the value pre-written or forced for the current I/O mapping variable. Double-clicking the cell in the "Prepared value" column for modification. After modification, use "Ctrl+F7" or "F7" to write the value or force it to be uploaded to the I/O mapping variable.
Bus cycle task	This function is applicable to virtual axes. The "Bus cycle task" must be bound to the called EtherCAT task (default is "Use bus cycle settings", which is applicable to a single master). It is prohibited to call the same virtual axis under multiple EtherCAT tasks.

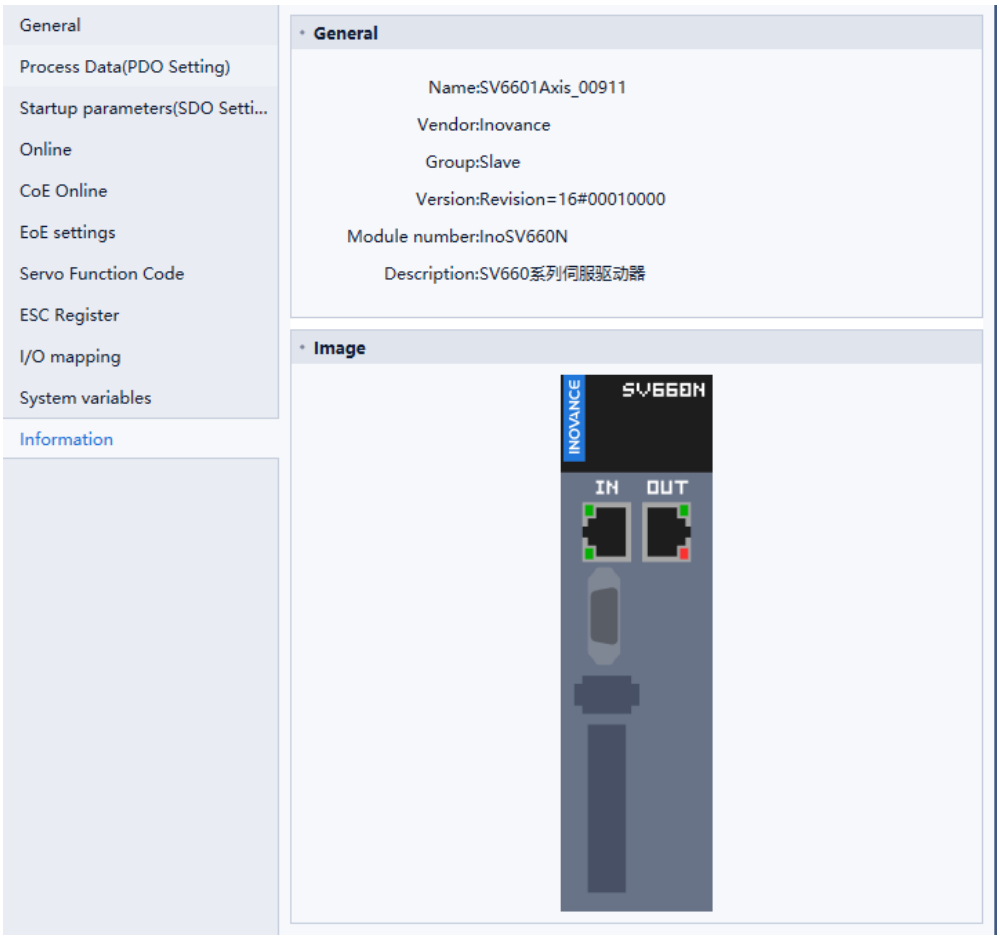
10.6.3.3.10 System Variables

In the "System variable" tab of the EtherCAT slave, instances (variables) of type ETC_Slave and _SYS_SlaveState are specified for servo slaves I/O, supporting control of devices connected to the EtherCAT slave via user programs, and displaying the device ID value of this slave.



10.6.3.3.11 Information

This interface displays the general information and image of the device, as shown in the following figure:



10.6.3.4 Disabling/Enabling an EtherCAT Device

Disabling EtherCAT device

In iFA software, a certain EtherCAT device can be disabled through commands or variable setting, thus "detaching" from the control system while retaining its configuration parameters and I/O mapping

information. If a device fails and needs to be replaced, this function can reduce the time required for replacement.

- Disabling operation and effect
  A device can be disabled through commands or variable setting, each with different scopes. Command disabling applies to all devices except PLCs Variable setting requires that there be corresponding variables in SYS_VAR.
  - Command disabling
    Right-click an EtherCAT master/slave node in the project tree and select "Disable device".
    You can also open the network configuration interface, right-click a slave device and select "Disable device".
    After disabling, it is necessary to perform a 10.8.3 full download of the PLC to make the disabling operation effective.
  - Variable setting
    When adding each EtherCAT slave, a corresponding variable is created in **SYS_VAR**, following the naming rule: J_device model.
    Setting member The member **bDisableEnable** disables the slave, and the setting method supports writing values in **SYS_VAR** or assigning values in user programs.

GLOBAL SYS_VAR

	Name	Data type	Address	Retain	Network publish	Comment	Attributes
1	VAR_GLOBAL						
2	LAN1_EIPSingle_STATUS	Array[1..32] OF IoStand...			Private		attribute 'device_created'
3	LAN1_EIPSlave_STATUS	Array[1..16] OF EtherNe...			Private		attribute 'device_created'
4	_EC1_Node	ARRAY[1..127] OF Ether...			Private		attribute 'device_created'
5	J_SV660N	IoStandard._SYS_SlaveS...		☑	Private		attribute 'device_created'
6	bDisableEnable	BOOL			Private		

After disabling, it is necessary to sequentially download, power off, and power on the PLC to make the disabling operation effective.

> **CAUTION**
>
> Powering off the PLC will result in the loss of values for temporary variables and variables with power-off retention set to FALSE.

- Disabling result
  The disabled devices are marked with the icon for differentiation.

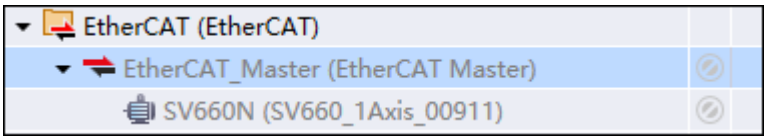
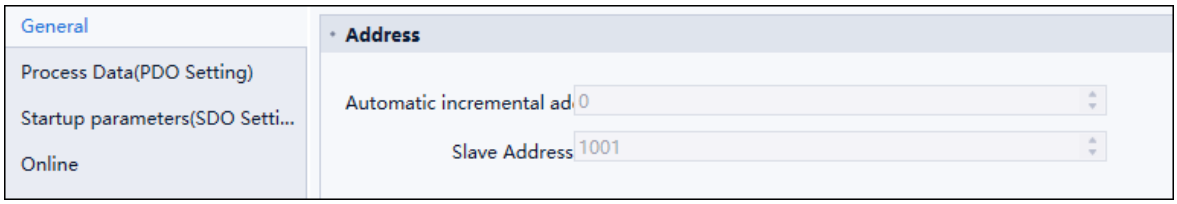


Figure 10-6-3-4-1 Icon in project tree

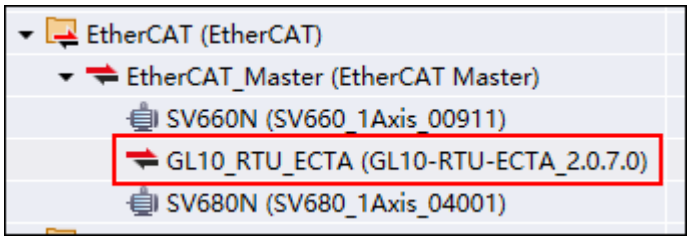


Figure 10-6-3-4-2 Icon in network configuration view

- Disabling Impact
  - If an EtherCAT slave device is addressed by address, after being disabled, iFA software will reassign addresses to subsequent devices. If it is addressed by alias, disabling has no impact on other slave devices.



  - When EtherCAT slave devices have inconsistent models, the physical device that is disabled must be removed from the control system to ensure the normal operation of other slaves.
    As shown in the figure below, three slave devices consist of two models. After the middle device is disabled, it needs to be physically removed to ensure the normal operation of the other two slaves.



Enabling EtherCAT device

The disabling and enabling of EtherCAT devices are a set of mutually corresponding operations. Only when the device is disabled is the enabling command visible.

The enabling operation and its effect are exactly the same as the disabling operation.

A device needs to be disabled and enabled in the same way. That is, if the device is disabled by setting a variable, it must be enabled by resetting the variable. If it is disabled using a command, it must also be enabled using a command.

10.6.4 EtherNet Communication Configuration

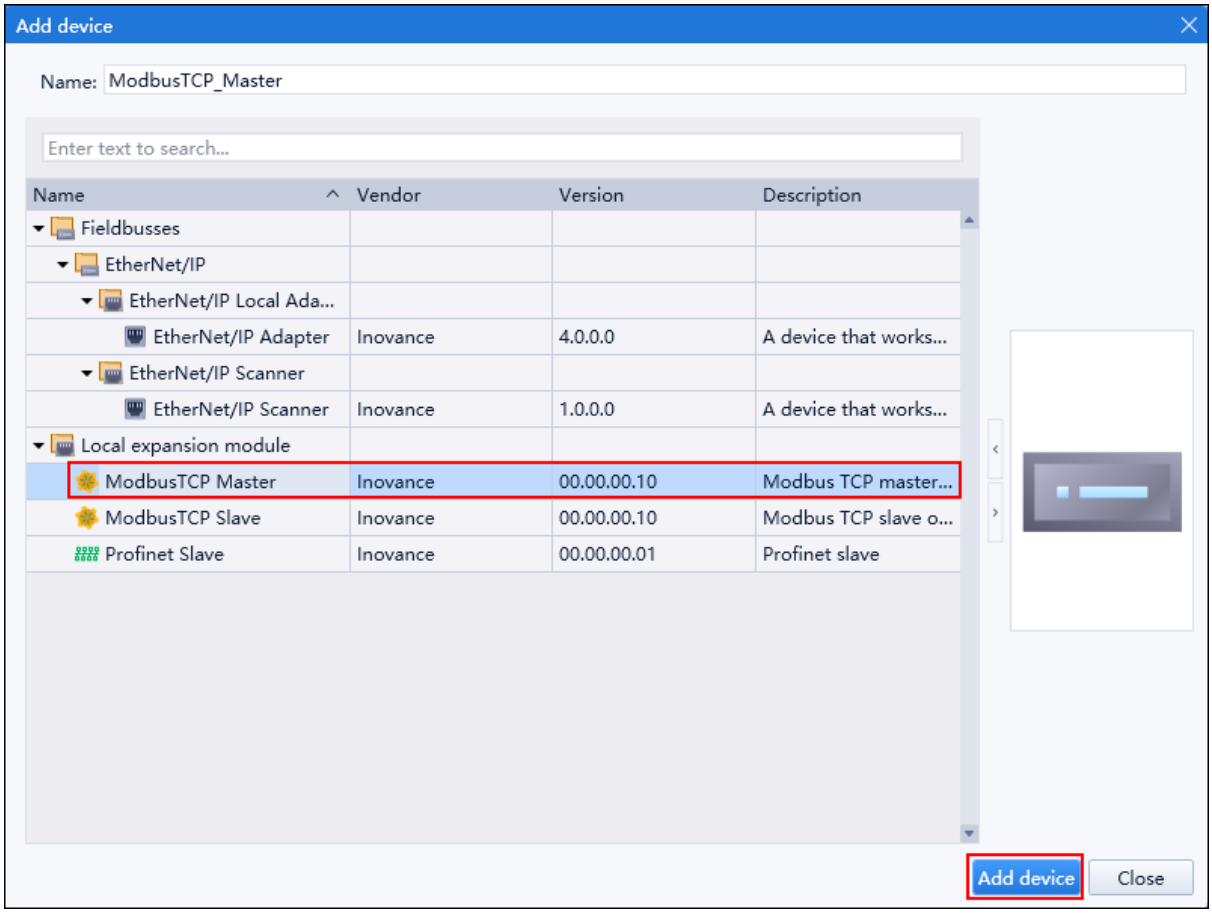
10.6.4.1 Modbus TCP Device Configuration

10.6.4.1.1 Adding a Modbus TCP Device

Adding a Modbus master device

Procedure

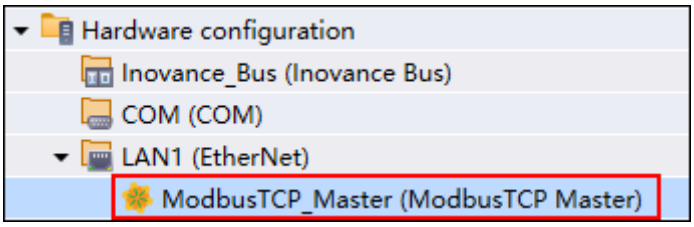
1. Click "Hardware configuration" in the project tree on the left.
2. Right-click "LAN1(EtherNet)" and select "Add device".
3. The pop-up "Add device" window is shown in the figure below.



4. Double-click "ModbusTCP master" or select it and click "Add device" to add the Modbus master.

Result

The added Modbus master is displayed under the "Hardware configuration" node.



Adding a Modbus slave device

Steps

1. Click "Hardware configuration" in the project tree on the left.
2. Right-click "LAN1(EtherNet)" and select "Add device".
3. The "Add device" window pops up.
4. Double-click "ModbusTCP slave" or select it and click "Add device" to add the Modbus slave.

Result

The added Modbus slave is displayed under the "Hardware configuration" node.

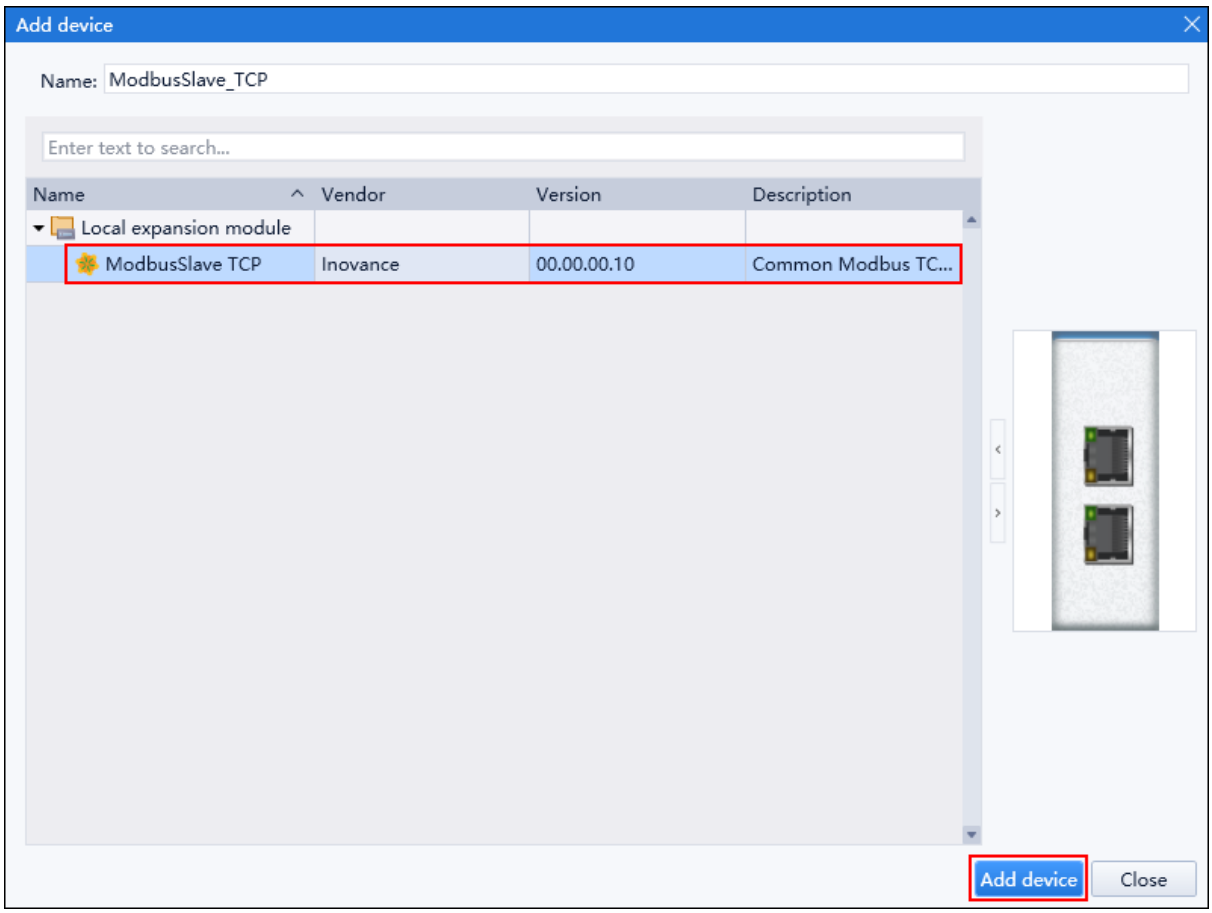
Mounting a Modbus slave under the Modbus master

Prerequisites

A Modbus master has already been added to the current device.

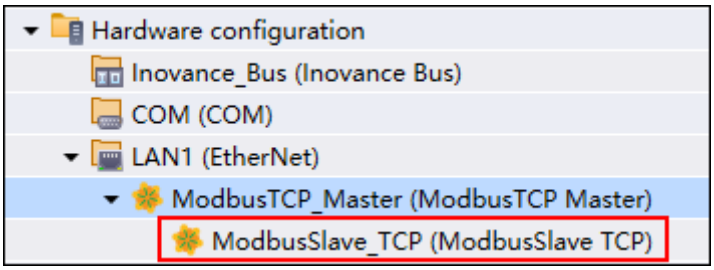
Procedure

1. Right-click the Modbus master device and select "Add device".
2. In the pop-up "Add device" window, select "ModbusSlave TCP".



Result

The corresponding master and slave devices are displayed under the "Hardware configuration" node.



10.6.4.1.2 Configuring a Modbus TCP Master

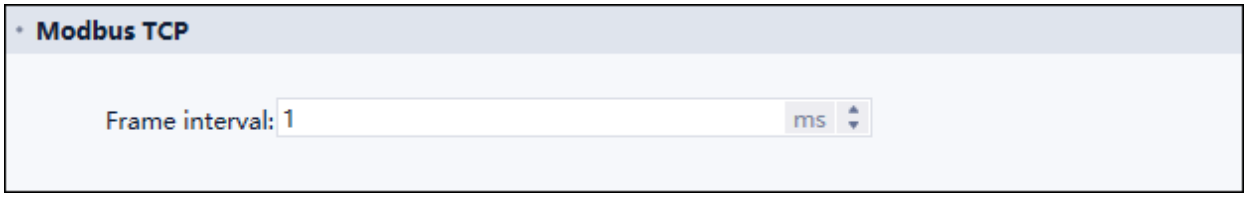
10.6.4.1.2.1 Modbus TCP Master Configuration

Prerequisites

The master device has been added.

Procedure

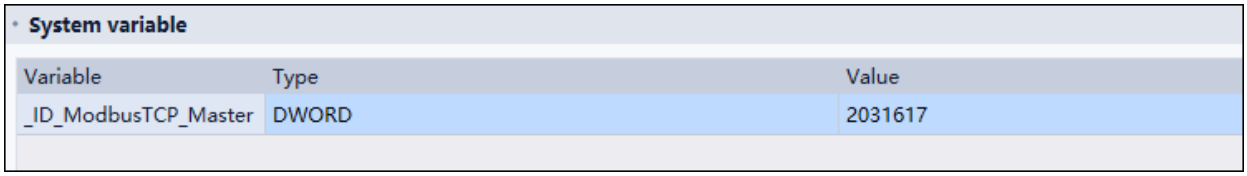
Double-click the Modbus TCP master device node to open the master configuration interface.



Modbus TCP master configuration is relatively simple, only requiring the frame interval to be set. In general, there is no need to change this setting.

10.6.4.1.2.2 System Variables

The device ID value of the Modbus TCP master is displayed in the "System variable" interface.



10.6.4.1.2.3 Information

The "Information" tab displays the following information for the current module: name, vendor, version, module number, and description.



10.6.4.1.3 Configuring a Modbus TCP Slave

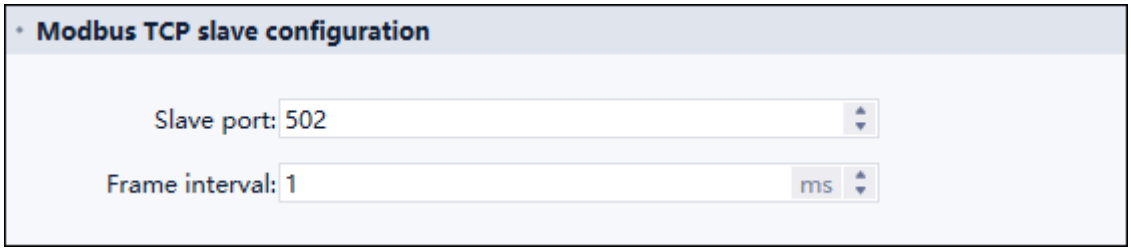
10.6.4.1.3.1 Modbus TCP Slave Settings

Prerequisites

A slave device has been added.

Procedure

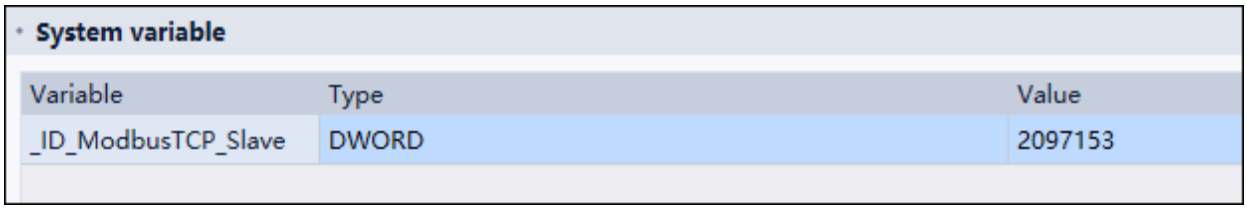
Double-click the Modbus TCP slave device node to open the slave configuration interface.



Item	Description
Slave port	The port number on which the slave listens Range: 1 to 65535. **Note:** System ports are prohibited from being used. System ports include 21, 22, 23, 44818, 2222, 34980, 48000, 12601 to 12610, 1260 to 1263, 8889, 4840, and 60000 to 62000.
Frame interval	The interval between the last received data frame and the next transmitted frame.

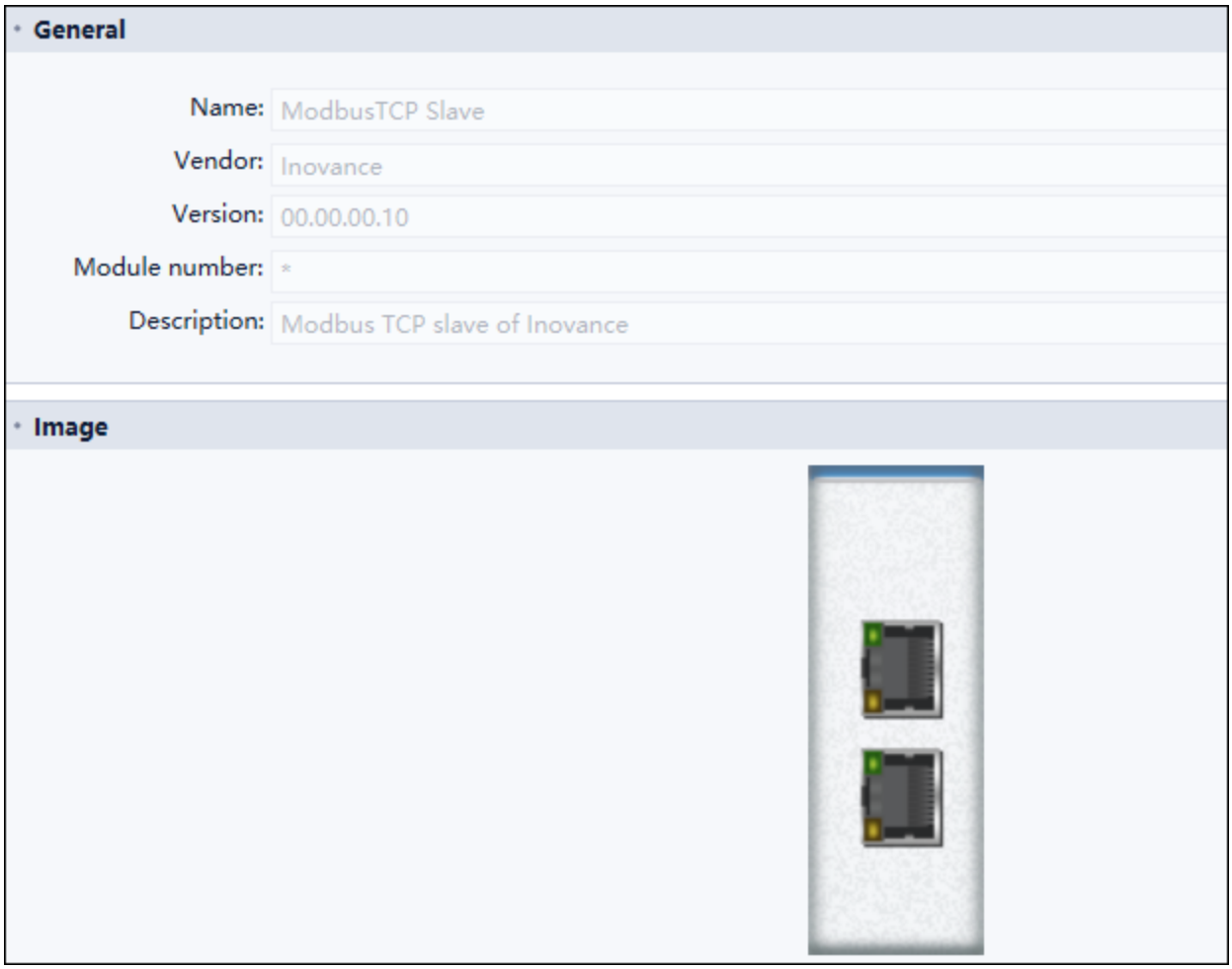
10.6.4.1.3.2 System Variables

The device ID value of the Modbus TCP slave is displayed in the "System variable" interface.



10.6.4.1.3.3 Information

The "Information" tab displays the following information for the current module: name, vendor, version, module number, description, and image.



10.6.4.1.4 Configuring a Modbus TCP Slave Under the Master

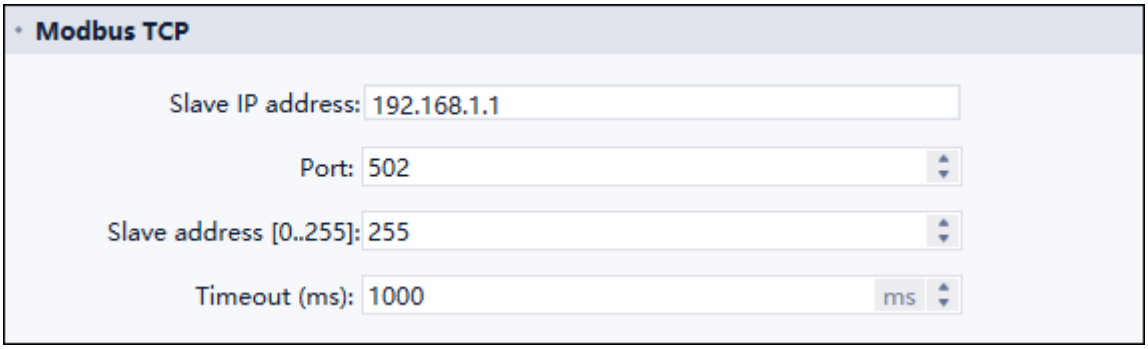
10.6.4.1.4.1 Modbus TCP Slave Settings

Prerequisites

A slave device has been added and mounted on the master.

Procedure

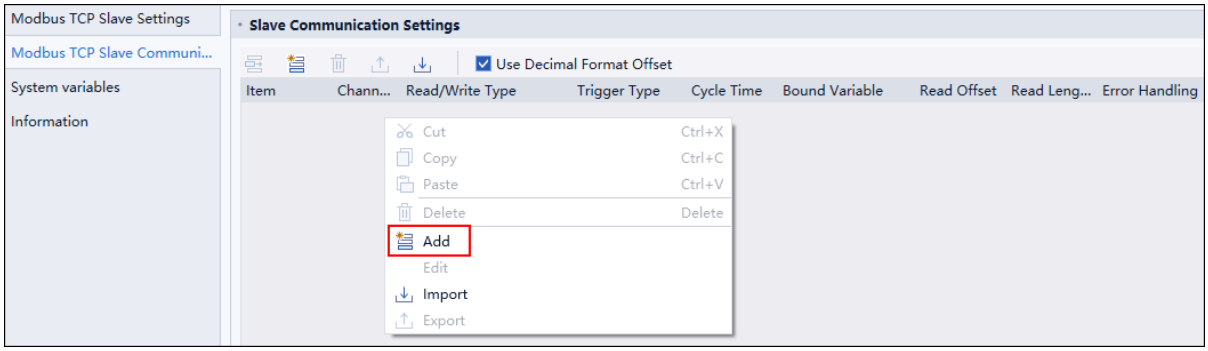
Double-click the added slave device node to open the slave configuration interface.



Item	Description
Slave IP address	The IP address of the Modbus slave in the actual hardware configuration.
Port	The port to access the peer slave. Range: 1 to 65535. **Note:** System ports are prohibited from being used. System ports include 21, 22, 23, 44818, 2222, 34980, 48000, 12601 to 12610, 1260 to 1263, 8889, 4840, and 60000 to 62000.
Slave address [0..255]	The Modbus application layer address corresponding to the slave.
Timeout period	Indicates the response time for a single connection to the slave, with a range from 2 ms to 1000 ms and a default value of 1000 ms. If the slave does not respond within this period, the master will terminate the request after sending the specified number of retry requests and display a fault message. **Note**: The timeout time must be greater than the longest CPU scan cycle, otherwise a fault will occur.

10.6.4.1.4.2 Modbus TCP Slave Communication Settings

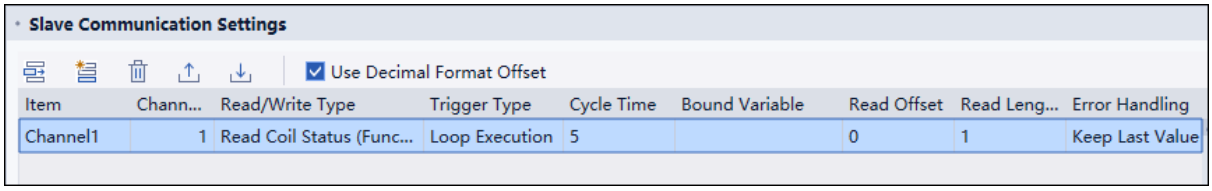
1. In the "Modbus TCP slave communication settings" tab interface, right-click in the blank area on the right and select "Add" to add a communication entry.



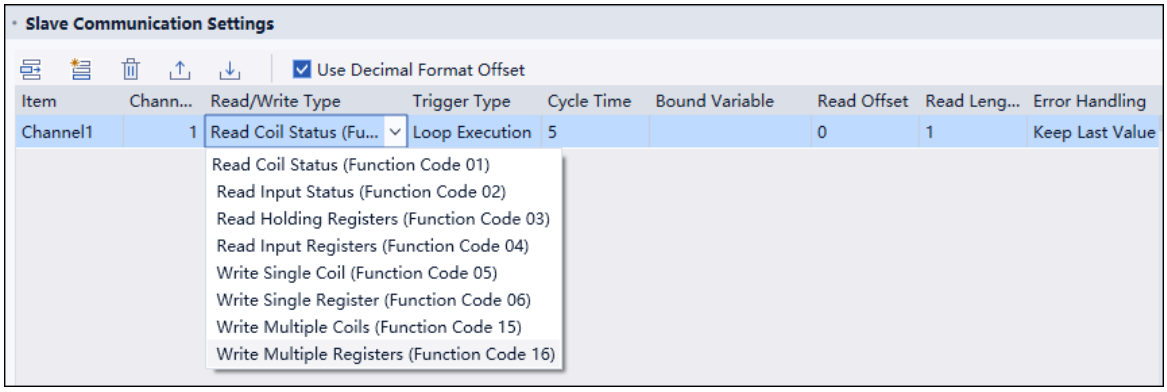
Different devices support different numbers of added channels, as shown in the table below.

Product Series	Firmware Version	Number of channels
EVO400, EVO500, EVO700, EVO700C, and APC series	V1.5 and above	A single slave can support up to 200 channels, and the total number of channels added by all slaves cannot exceed 512.
	V1.4 and below	A single slave can support up to 60 channels.
EVO800 series	V1.1 and above	A single slave can support up to 200 channels, and the total number of channels added by all slaves cannot exceed 512.
	V1.0	A single slave can support up to 60 channels.

2. The added entry is shown in the figure below.



- Item: The name of the entry
- Channel number: The sequence number corresponding to this entry
- Read/Write type: Configures the function code, which can be set to: read coil status (function code 01), read input status (function code 02), read holding register (function code 03), read input register (function code 04), write single coil (function code 05), write single register (function code 06), write multiple coils (function code 15), write multiple registers (function code 16).



- Trigger type: The trigger type corresponding to this entry, supporting cyclic execution and level triggering (for EVO500 series PLCs, only version V1.3 and above support level triggering).
- Cycle time: The cycle time cannot be edited in level triggering mode.

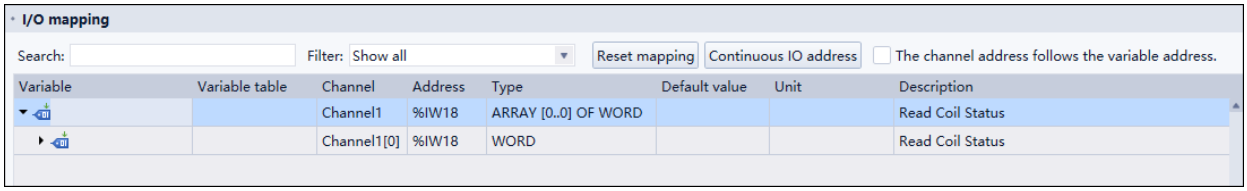
- Bound variable: You can manually input or bind global variables in the project using the input assistant. After binding, the channel data access status can be controlled through the program. The variable type must be BOOL. Non-BOOL types are not supported. One channel can only bind one variable. After executing a trigger command, the BOOL variable value is set to FALSE. The command is effective only once and only when the variable status is TRUE.
- Read offset: The offset position based on the read start address
- Read length: The length of the read operation
- Fault handling: The handling method when a fault occurs
- Write offset: The write offset position
- Write length: The length of the write operation
- Retry count: The number of times to retransmit when encountering network issues
- Comment: Comment information



For details about the entries in this interface, see the Modbus protocol.

10.6.4.1.4.3 I/O Mapping

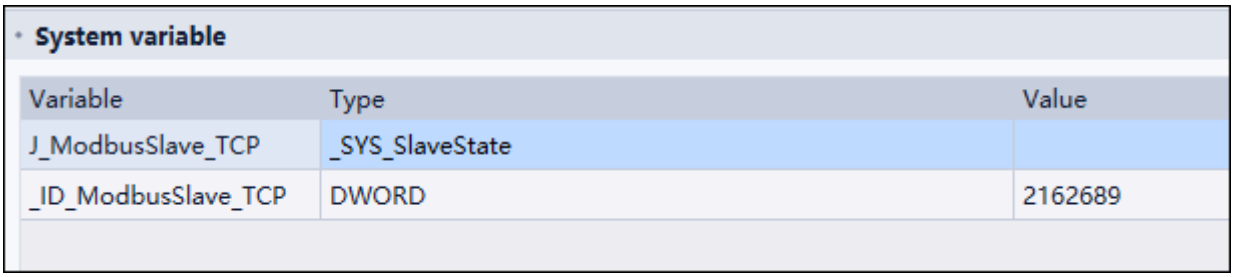
After configuring the Modbus TCP slave communication settings, you can configure the Modbus-related I/O settings. The number of I/O configuration entries in the "I/O mapping" tab is consistent with the number of entries in the "Modbus TCP slave communication settings" tab.



For details about the configuration parameters, see 10.6.3.3.9 I/O Mapping.

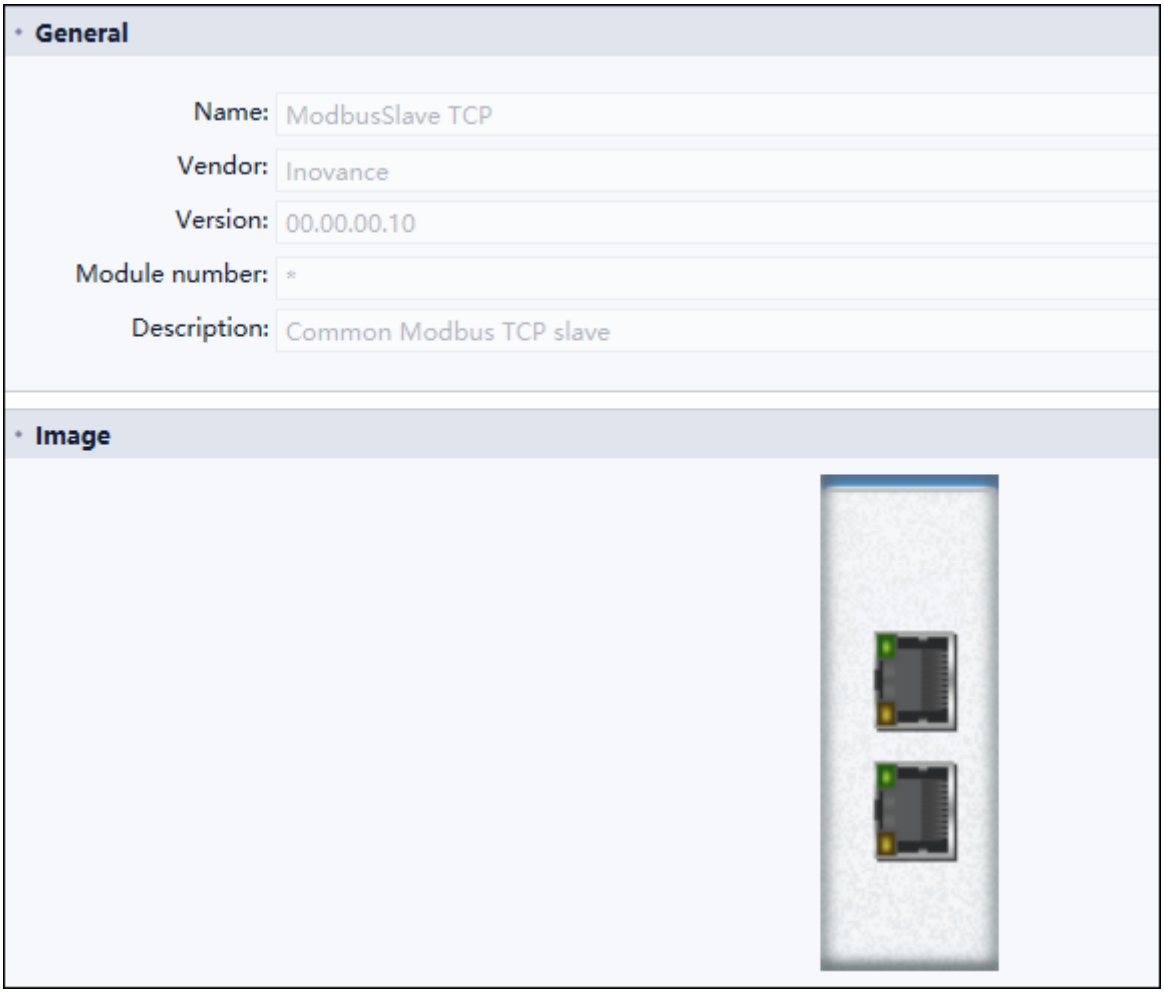
10.6.4.1.4.4 System Variables

In the "System variable" tab, an instance (variable) of type _SYS_SlaveState is specified for the slave I/O and the device ID values of the mounted Modbus TCP slaves are displayed.



10.6.4.1.4.5 Information

The "Information" tab displays the following information for the current module: name, vendor, version, module number, description, and image.



10.6.4.1.5 Modbus TCP Variable Addressing

Coil

Coil is a bit variable with only two statuses, 0 and 1. This PLC (taking the EVO500 series as an example) includes variables in the Q area and I area.

Variable Name	Command Code	Starting Address	Number of coils	Note
QX0.0-QX8191.7	0x01, 0x05, 0x0f	0	65536	Accessible through the universal standard Modbus protocol
IX0.0-IX8191.7	0x2	0	65536	Accessible through the universal standard Modbus protocol

Register

Register is a 16-bit (word) variable. Take the EVO500 series as an example, the PLC includes only variables in M area.

Variable Name	Command Code	Starting Address	Number of Registers	Note
MW0-MW65535	0x03, 0X04, 0x06, 0x10	0	65536	Accessible through the universal standard Modbus protocol

NOTE

The soft elements of the EVO400/EVO500 series contain three types: Q area, I area, and M area. They can be accessed by bit, byte, word, and double word, such as: %QX, %QB, %QW, %QD. The conversion rules are as follows:

QB0 = (QX0.0 to QX0.7)
QW0 = (QB0 to QB1) =((QX0.0 to QX0.7) + (QX1.0 to QX1.7))
QD0 = (QW0 to QW1) = (QB0 to QB3) = ((QX0.0 to QX0.7) + (QX1.0 to X1.7) + (QX2.0 to QX2.7) + (QX3.0 to QX3.7))

Register address indexing rule

Bit Addressing	Byte Addressing	Word Addressing	Dword Addressing		Bit Addressing	Byte Addressing	Word Addressing	Dword Addressing
QX0.0	QB0	QW0	QD0		MX0.0	MB0	MW0	MD0
QX0.1					MX0.1			
QX0.2					MX0.2			
QX0.3					MX0.3			
QX0.4					MX0.4			
QX0.5					MX0.5			
QX0.6					MX0.6			
QX0.7					MX0.7			
QX1.0	QB1				MX1.0	MB1		
QX1.1					MX1.1			
QX1.2					MX1.2			
QX1.3					MX1.3			
QX1.4					MX1.4			
QX1.5					MX1.5			
QX1.6					MX1.6			

Bit Addressing	Byte Addressing	Word Addressing	Dword Addressing		Bit Addressing	Byte Addressing	Word Addressing	Dword Addressing
QX1.7					MX1.7			
QX2.0	QB2	QW1			MX2.0	MB2	MW1	
QX2.1					MX2.1			
QX2.2					MX2.2			
QX2.3					MX2.3			
QX2.4					MX2.4			
QX2.5					MX2.5			
QX2.6					MX2.6			
QX2.7					MX2.7			
QX3.0	QB3				MX3.0	MB3		
QX3.1					MX3.1			
QX3.2					MX3.2			
QX3.3					MX3.3			
QX3.4					MX3.4			
QX3.5					MX3.5			
QX3.6					MX3.6			
QX3.7					MX3.7			
QX4.0	QB4	QW2	QD1		MX4.0	MB4	MW2	MD1
QX4.1					MX4.1			

NOTE

The starting address of EVO400/EVO500's word-type register is an even byte address. The starting address of DWord-type register aligns with even word addresses, and its index number is in a 2x relationship, making address calculation convenient.

10.6.4.1.6 Modbus TCP Communication Frame Format

- Reading coils

Use the 0x01 command code to read Q variables.

Request frame format: transaction meta identifier + protocol identifier + length + slave address + 0x01 + starting address of coil + number of coils

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Transaction meta identifier	2 bytes	Identifier of the Modbus request/response transaction
2	Protocol identifier	2 bytes	0 = Modbus protocol
3	Length	2 bytes	Number of following bytes
4	Slave address	1	Range: 1 to 247
5	0x01 (command code)	1	Read coils.
6	Head address of coils	2 bytes	Upper bits are followed by lower bits.
7	Number of coils (N)	2 bytes	Upper bits are followed by lower bits.

Response frame format: transaction meta identifier + protocol identifier + length + slave address + 0x01 + number of bytes + status of coils

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Transaction meta identifier	2 bytes	Identifier of the Modbus request/response transaction
2	Protocol identifier	2 bytes	0 = Modbus protocol
3	Length	2 bytes	Number of following bytes
4	Slave address	1	Range: 1 to 247
5	0x01 (command code)	1	Read coils.
6	Number of bytes	1	Value: [(N + 7) / 8]
7	Coil status	[(N + 7) / 8] bytes	Eight coils are indicated by one byte. If the last byte has less than eight bits, enter 0 for undefined bits. The first eight coils are indicated by the first byte, and the coil with the smallest address is indicated by the least significant bit, and so on.

- Reading registers

Use the 0x03 command code to read M variables.

Request frame format: transaction meta identifier + protocol identifier + length + slave address + 0x03 + starting address of register + number of registers

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Transaction meta identifier	2 bytes	Identifier of the Modbus request/response transaction
2	Protocol identifier	2 bytes	0 = Modbus protocol
3	Length	2 bytes	Number of following bytes
4	Slave address	1	Range: 1 to 247
5	0x03 (command code)	1	Read registers.
6	Head address of registers	2 bytes	Upper bits are followed by lower bits.
7	Number of registers (N)	2 bytes	Upper bits are followed by lower bits.

Response frame format: transaction meta identifier + protocol identifier + length + slave address + 0x03 + number of bytes + register value

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Transaction meta identifier	2 bytes	Identifier of the Modbus request/response transaction
2	Protocol identifier	2 bytes	0 = Modbus protocol
3	Length	2 bytes	Number of following bytes
4	Slave address	1	Range: 1 to 247
5	0x03 (command code)	1	Read registers.
6	Number of bytes	1	Value: N*2
7	Register value	N x 2	One register value is indicated by two bytes. Upper bits are followed by lower bits. The register with a smaller address is indicated by the first byte.

- Writing a single coil

Use the 0x05 command code to write Q variables.

Request frame format: transaction meta identifier + protocol identifier + length + slave address + 0x05 + coil address + coil state

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Transaction meta identifier	2 bytes	Identifier of the Modbus request/response transaction
2	Protocol identifier	2 bytes	0 = Modbus protocol
3	Length	2 bytes	Number of following bytes

No.	Definition of Data (Byte)	Number of Bytes	Note
4	Slave address	1	Range: 1 to 247
5	0x05 (command code)	1	Writes a single coil.
6	Address of the coil	2 bytes	Upper bits are followed by lower bits.
7	Coil status	2 bytes	Lower bits are followed by upper bits. A non-zero value is valid.

Response frame format: transaction meta identifier + protocol identifier + length + slave address + 0x05 + coil address + coil state

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Transaction meta identifier	2 bytes	Identifier of the Modbus request/response transaction
2	Protocol identifier	2 bytes	0 = Modbus protocol
3	Length	2 bytes	Number of following bytes
4	Slave address	1	Range: 1 to 247
5	0x05 (command code)	1	Writes a single coil.
6	Address of the coil	2 bytes	Upper bits are followed by lower bits.
7	Coil status	2 bytes	Lower bits are followed by upper bits. A non-zero value is valid.

- Writes one register.

Use the 0x06 command code to write M variables.

Request frame format: transaction meta identifier + protocol identifier + length + slave address + 0x06 + register address + register value

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Transaction meta identifier	2 bytes	Identifier of the Modbus request/response transaction
2	Protocol identifier	2 bytes	0 = Modbus protocol
3	Length	2 bytes	Number of following bytes
4	Slave address	1	Range: 1 to 247
5	0x06 (command code)	1	Writes a single register.
6	Register address	2 bytes	Upper bits are followed by lower bits.
7	Register value	2 bytes	Upper bits are followed by lower bits. A non-zero value is valid.

Response frame format: transaction meta identifier + protocol identifier + length + slave address + 0x06 + register address + register value

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Transaction meta identifier	2 bytes	Identifier of the Modbus request/response transaction
2	Protocol identifier	2 bytes	0 = Modbus protocol
3	Length	2 bytes	Number of following bytes
4	Slave address	1	Range: 1 to 247
5	0x06 (command code)	1	Writes a single register.
6	Register address	2 bytes	Upper bits are followed by lower bits.
7	Register value	2 bytes	Upper bits are followed by lower bits. A non-zero value is valid.

- Writing multiple coils

Use the 0x0f command code to write multiple consecutive Q variables.

Request frame format: transaction meta identifier + protocol identifier + length + slave address + 0x0f + starting address of coil + number of coils + number of bytes + coil state

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Transaction meta identifier	2 bytes	Identifier of the Modbus request/response transaction
2	Protocol identifier	2 bytes	0 = Modbus protocol
3	Length	2 bytes	Number of following bytes
4	Slave address	1	Range: 1 to 247
5	0x0f (command code)	1	Writes multiple single coils.
6	Head address of coils	2 bytes	Upper bits are followed by lower bits.
7	Number of coils (N)	2 bytes	Upper bits are followed by lower bits. The maximum is 1968.
8	Number of bytes	1	Value: [(N + 7) / 8]
9	Coil status	[(N + 7) / 8] bytes	Eight coils are indicated by one byte. If the last byte has less than eight bits, enter 0 for undefined bits. The first eight coils are indicated by the first byte, and the coil with the smallest address is indicated by the least significant bit.

Response frame format: transaction meta identifier + protocol identifier + length + slave address + 0x0f + head address of coils + number of coils

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Transaction meta identifier	2 bytes	Identifier of the Modbus request/response transaction
2	Protocol identifier	2 bytes	0 = Modbus protocol
3	Length	2 bytes	Number of following bytes
4	Slave address	1	Range: 1 to 247
5	0x0f (command code)	1	Writes multiple single coils.
6	Head address of coils	2 bytes	Upper bits are followed by lower bits.
7	Number of coils	2 bytes	Upper bits are followed by lower bits.

- Writing multiple registers

Use the 0x10 command code to write multiple consecutive M variables.

Request frame format: transaction meta identifier + protocol identifier + length + slave address + 0x10 + starting address of register + number of registers + number of bytes + register value

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Transaction meta identifier	2 bytes	Identifier of the Modbus request/response transaction
2	Protocol identifier	2 bytes	0 = Modbus protocol
3	Length	2 bytes	Number of following bytes
4	Slave address	1	Range: 1 to 247
5	0x10 (command code)	1	Writes multiple registers.
6	Head address of registers	2 bytes	Upper bits are followed by lower bits.
7	Number of Registers	2 bytes	Upper bits are followed by lower bits.
8	Number of bytes	1	Value: N*2
9	Register value	N*2 (N*4) bytes	-

Response frame format: transaction meta identifier + protocol identifier + length + slave address + 0x10 + register address + register value

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Transaction meta identifier	2 bytes	Identifier of the Modbus request/response transaction
2	Protocol identifier	2 bytes	0 = Modbus protocol
3	Length	2 bytes	Number of following bytes

No.	Definition of Data (Byte)	Number of Bytes	Note
4	Slave address	1	Range: 1 to 247
5	0x10 (command code)	1	Writes multiple registers.
6	Head address of registers	2 bytes	Upper bits are followed by lower bits.
7	Register value	2 bytes	Upper bits are followed by lower bits.

10.6.4.1.7 Common Modbus TCP Faults

The main faults that occur when a Modbus TCP master connects to a Modbus TCP slave are as follows:

- The Modbus TCP master fails to establish communication with the Modbus TCP slave due to incorrect IP configuration.
- The Modbus TCP master accesses an illegal address on the Modbus TCP slave, resulting in a faulty response.
- The Modbus TCP master attempts to write to a register on the Modbus TCP slave, but the register only supports read operations, causing the master to receive a fault response from the slave.

Fault response frame format

Fault response frame format: transaction meta identifier + protocol identifier + length + slave address + (command code+0x80) + fault code + CRC check

This fault frame applies to all operation command frames.

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Transaction meta identifier	2 bytes	Identifier of the Modbus request/response transaction
2	Protocol identifier	2 bytes	0 = Modbus protocol
3	Length	2 bytes	Number of following bytes
4	Slave address	1	Value range: 1 to 247
5	Command code + 0x80	1	Fault command code
6	Fault code	1	1 to 4
7	CRC code	2 bytes	Lower bits are followed by higher bits.

Fault code list

Fault Code	Description	Cause	Solution
0x9E360000	Fault recovery	-	Channel fault has been recovered.

Fault Code	Description	Cause	Solution
0x9E360072	Illegal data address	The accessed peer data address does not exist or is out of bounds.	Check if the data access address or data length exceeds limits.
0x9E360074	Unsupported command code	The accessed peer command code is not supported.	Check if the slave supports the command code requested by the master.
0x9E360075	Receiving timeout	• The communication link is faulty. • The peer device is not operational.	1. Check if the master IP address or port is configured correctly. 2. Check if the Modbus TCP network of the peer device is working properly. 3. Check if the Modbus TCP of the peer device has started normally. 4. Optimize the wiring by eliminating potential interference sources.
0x9E360076	Illegal data value	The data value is illegal.	Check if the response data from the peer slave is faulty.
0x9E360077	Buffer overflow	Buffer overflow	Check if the number of requested coils or registers exceeds the maximum supported by Modbus TCP.
0x9E360078	Frame fault	The data frame is faulty.	Check if the length of the response data frame of the peer slave or the data frame itself is faulty.

Fault Code	Description	Cause	Solution
0x9E36009A	Modbus TCP cannot connect to the device.	Modbus TCP link is faulty.	1. Check if the master IP address or port is configured correctly. 2. Check if the network of the peer device can work normally. 3. Check whether the wiring is disconnected. 4. Optimize the wiring by eliminating potential interference sources.
0x9E35009B	Modbus TCP master IP address declaration failed	EtherNet port IP address does not match the device network port IP address.	Check if the EtherNet port IP address is set correctly.
0x9E3E009B	Modbus TCP slave IP address declaration failed	EtherNet port IP address does not match the device network port IP address.	Check if the EtherNet port IP address is set correctly.
0x9E36009C	Modbus TCP slave port occupied	The slave started on the PC occupies the Modbus TCP port configured by iFA Evolution.	Close the slave started on the PC to release the occupied port.

10.6.4.2 EtherNet/IP Device Configuration

10.6.4.2.1 Adding an EtherNet/IP Device

Adding an adapter

The adapter acts as the target end device.

Prerequisites

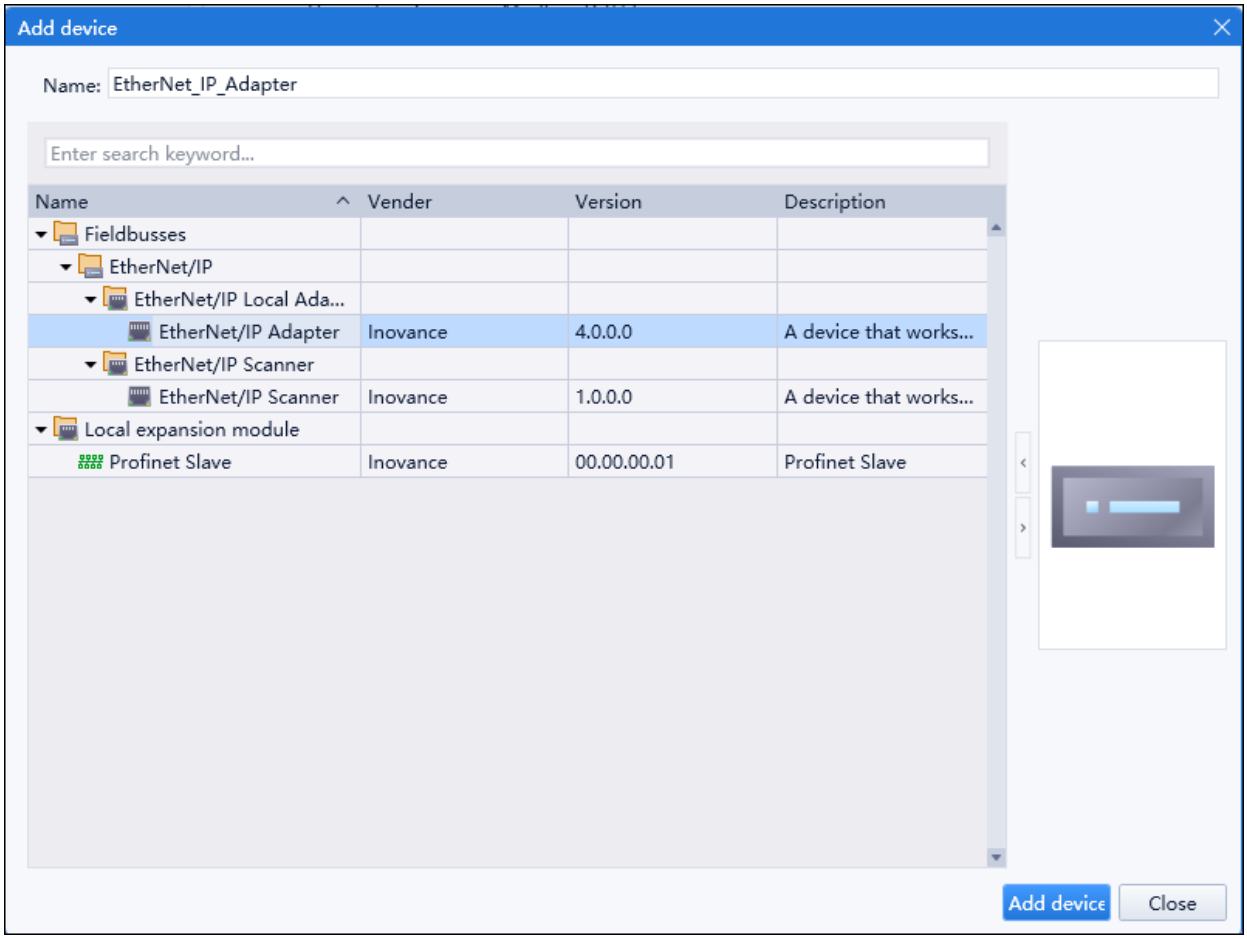
The selected device supports EIP communication.

Procedure

1. Click "Hardware configuration" in the project tree on the left.

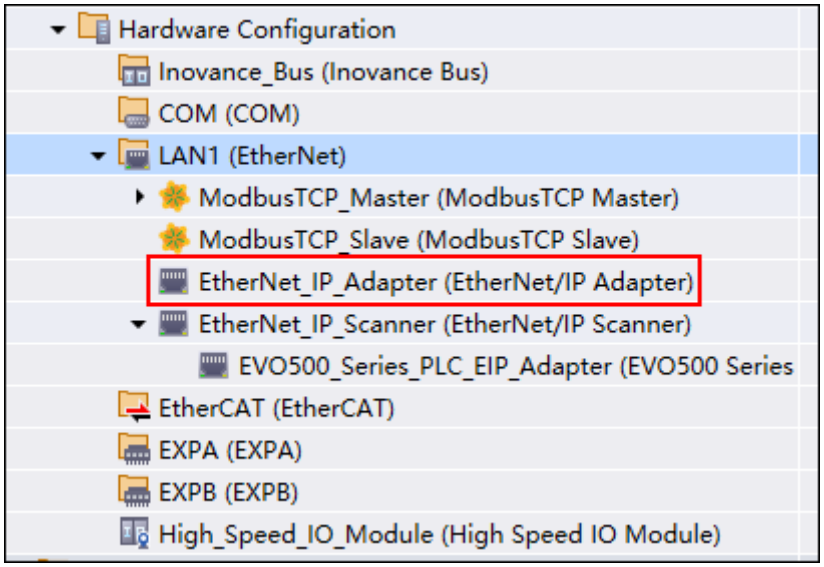
2. Right-click "LAN1(EtherNet)" or "LAN1_EVO800_CM_EN(EtherNet)", and select "Add device".

3. Double-click "EtherNet/IP Adapter", or select it and click "Add device" to add the adapter.



Result

The added adapter is displayed under "Hardware configuration".



Adding a scanner

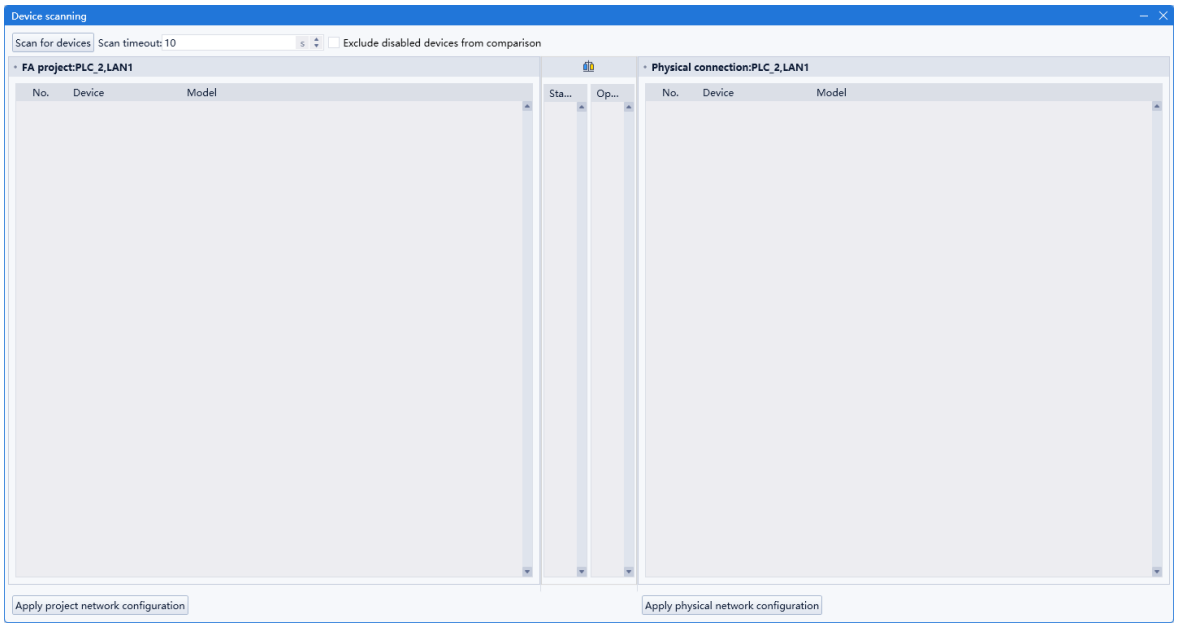
The scanner acts as the originator device.

Prerequisites

The selected device supports EIP communication.

Procedure

1. Click "Hardware configuration" in the project tree on the left.
2. Right-click "LAN1(EtherNet)" or "LAN1_EVO800_CM_EN(EtherNet)", and select "Add device".
3. Double-click "EtherNet/IP Scanner", or select it and click "Add device" to add the scanner.
4. Add a remote adapter.
   - Adding offline
     Right-click the "EtherNet/IP_Scanner" node and select "Add device", then double-click the target device (such as EVO500) in the pop-up window.
   - Adding online
     Right-click the "EtherNet/IP_Scanner" node and select "Scan for devices".



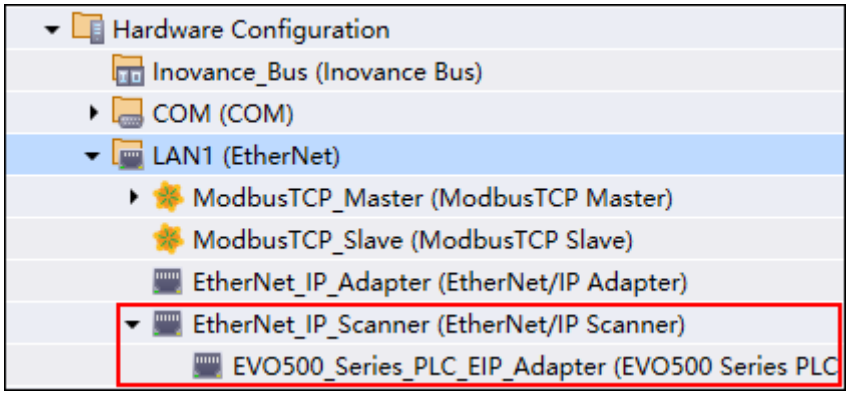
   - Scan timeout: The default scan timeout time is 10 seconds, with a range of 5 to 300. Within this time, scanned devices will be displayed in the right list.
   - Ignore disabled device comparison: If checked, disabled devices do not participate in the comparison. However, they are included in the comparison operation during device copying. If device comparison is successful, the device configuration is retained; otherwise, the device is deleted. The disabled status of devices is adjusted based on their actual status in the configuration list. For example, if a device in the physical device list is successfully compared, its disabled status is enabled. However, if the originally disabled device exists in the project configuration device list, the disabled status remains upon successful comparison.
   - Status: Compares the data between the project and physical devices.
   - Action: Copies scanned devices from the network to the project network.
   - Apply project network configuration: Applies the configuration of the devices in the project network to the project.
   - Apply physical network configuration: Applies the configuration of all scanned devices to the project.

   Select target devices as needed and apply them to the project.

   A remote adapter refers to a remote target end device, which can only be added after it is added to iFA by installing the EDS file. For details, see 8.15 Device Library Usage.

Result

The added scanner and remote adapter are displayed under "Hardware configuration".



10.6.4.2.2 EtherNet_IP_Scanner Configuration

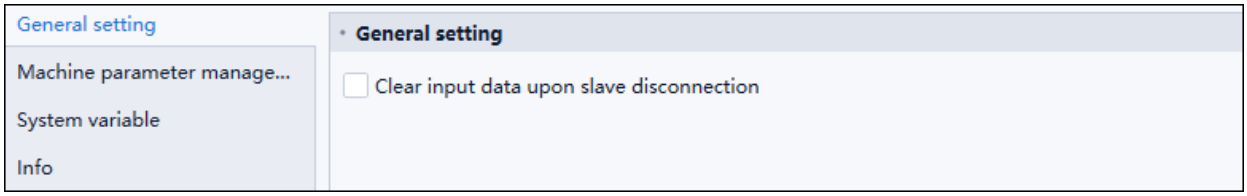
10.6.4.2.2.1 Scanner Settings

Prerequisites

A scanner device has been added.

Procedure

Double-click the added scanner node in the project tree to open the configuration interface.



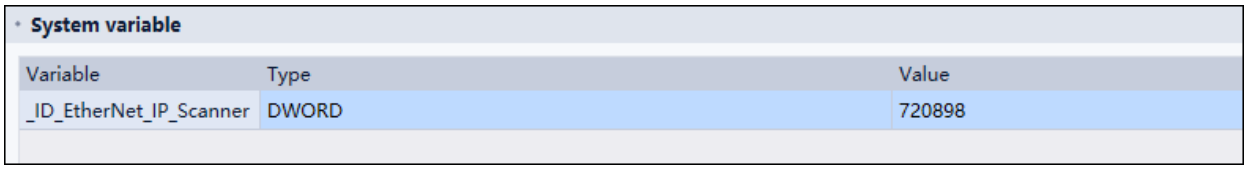
General settings

Clear input data when slave line is disconnected: By default not selected. When selected, if any slave device loses communication, its corresponding input data area will be cleared.



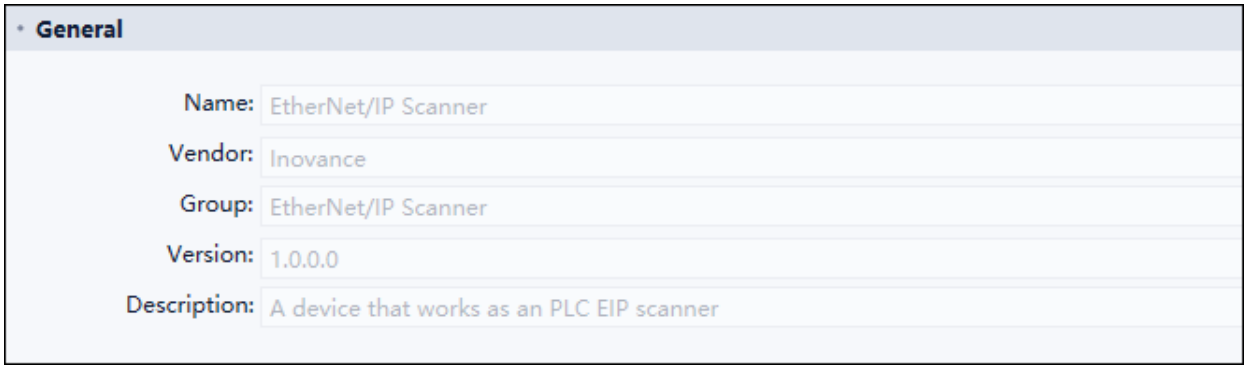
System Variables

Displays the device ID value of the scanner.



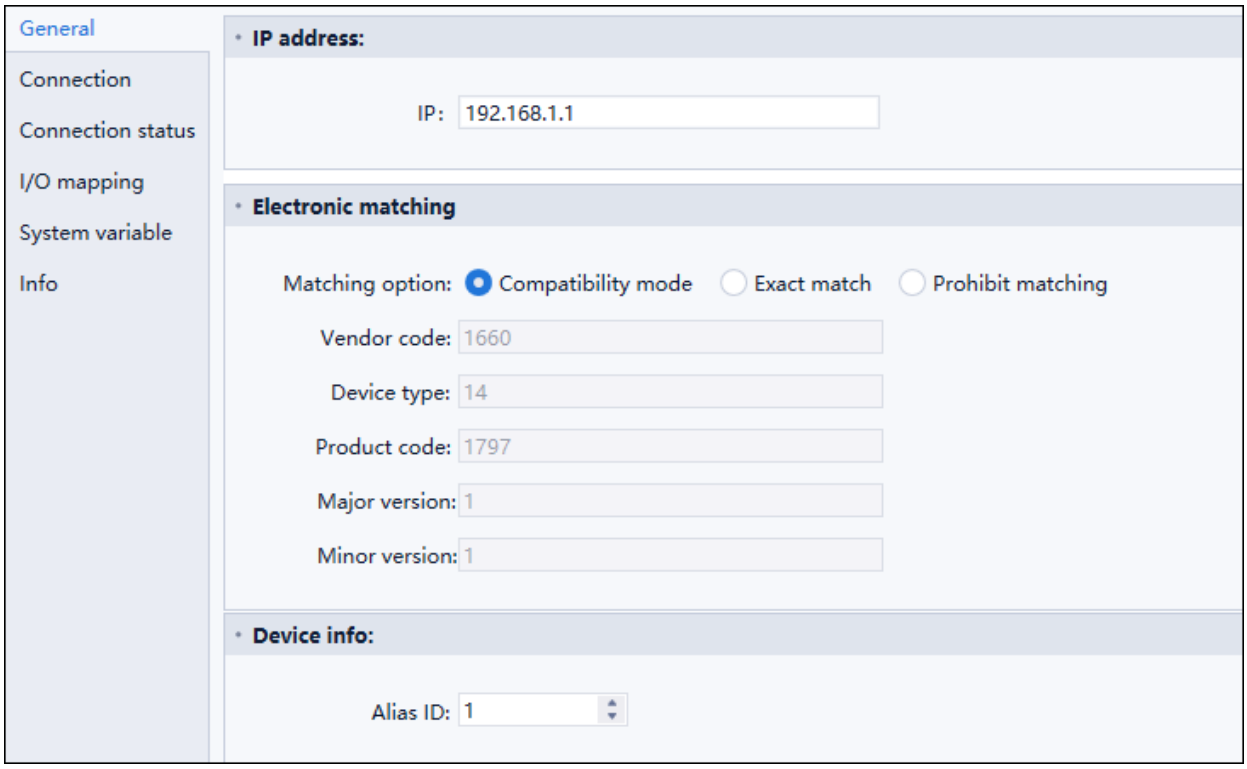
Information

Displays basic information of the scanner, including name, vendor, group, version, and description.



10.6.4.2.2.2 General Settings of Remote Adapters

Double-click the remote adapter node added under **EtherNet_IP_Scanner** to open the general settings interface.



- IP address
  Enter the IP address of the local adapter, which is the IP address of the PLC it belongs to.
- Electronic matching
  This function is used to reduce the risks of misusing devices in control systems. It can compare the devices defined in the project with the installed devices. If the match fails, a fault will occur.
  The following properties are compared.
  Vendor code: The code of the device manufacturer
  Device type: A numeric code
  Product code: A specific product type
  Major version: A number representing the device's major functions
  Minor version: A number representing device's updates
  - Compatibility mode: Conditional matching. Communication can only be established if the version number of the installed device is the same as or higher than that of the defined device, with the following prerequisites:
    The "Vendor code", "Device type", and "Product code" of the installed device are exactly the same as

those of the defined device.
If the major versions are the same, the minor version of the installed device must be the same as or higher than that of the defined device.

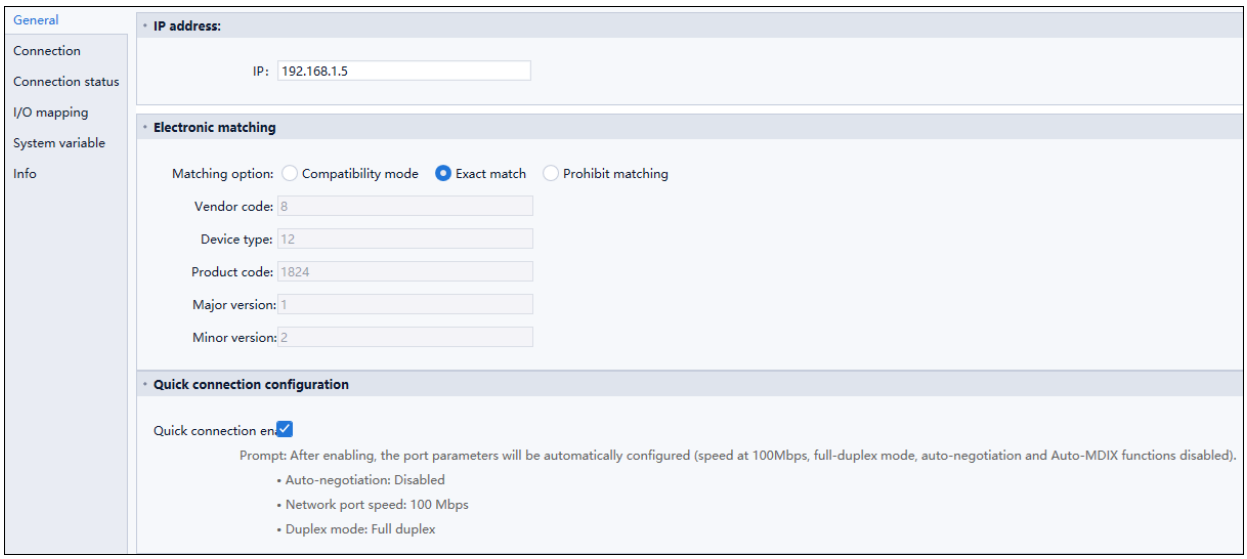
- Exact match: Complete matching. Communication can only be established if all electronic matching information is completely consistent.
- Prohibit matching: Ignores electronic matching information and establishes communication directly.

> **WARNING**
>
> The "Prohibit matching" option should be used with particular caution. Improper use of this option can lead to personal injury, property damage, or economic loss.
> It is strongly recommended not to check the use "Prohibit matching" option.

- Device info
  Set the alias ID of the remote adapter, used to configure the IP address of the remote adapter via instructions.
- Quick connection configuration (EVO700, EVO700C, EVO800, APC760C)
  When the remote adapter device supports the quick connection feature compliant with the ODVA specification, the adapter device can power on, connect to the EtherNet/IP Scanner, and start I/O data exchange within less than 500 ms.

  iFA software can automatically detect and display whether the adapter device supports quick connection. Enabling this feature activates the related configuration options of the PLC, allowing the PLC to quickly connect to adapter slaves and exchange I/O data at the software level.

> **CAUTION**
>
> Enabling quick connection will disable address conflict detection on the network port by default. Potential address conflict detection issues may not be prompted or handled. When no quick connection device is enabled on this port, the address conflict detection function will be automatically reactivated.



NOTE

- The quick connection function is currently not supported by third-party remote adapter devices. For such devices, refer to their user guides to enable quick connection.
- The quick connection function only modifies the PLC at the software level to meet the requirements for quick connection. Additionally, the network port configuration connected to the quick connection device must be adjusted to reduce negotiation time. If the quick connection device is directly connected to the PLC, refer to 10.11.22 Network Port Configuration for modifications; if the quick connection device is connected to the PLC through a switch, modify the corresponding switch port configuration. The following network port settings must be applied:
  - Speed: 100 Mbps.
  - Duplex mode: Full duplex.
  - Auto-negotiation: Disabled.
  - Auto wiring sequence: Disabled.

10.6.4.2.2.3 I/O Mapping

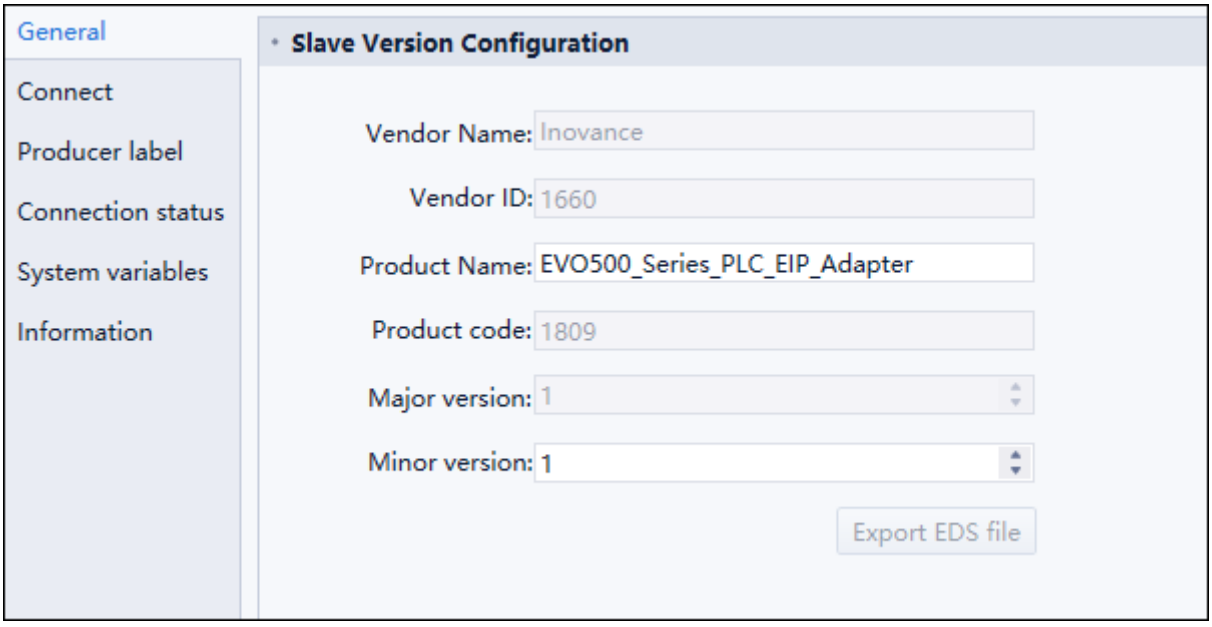
Refer to the 10.6.4.2.3.2 I/O Mapping configuration of the local adapter.

10.6.4.2.3 EtherNet_IP_Adapter Configuration

Configure this machine as the target device.

10.6.4.2.3.1 General Settings of Local Adapters

Double-click the added adapter in the project tree to open the adapter configuration interface.



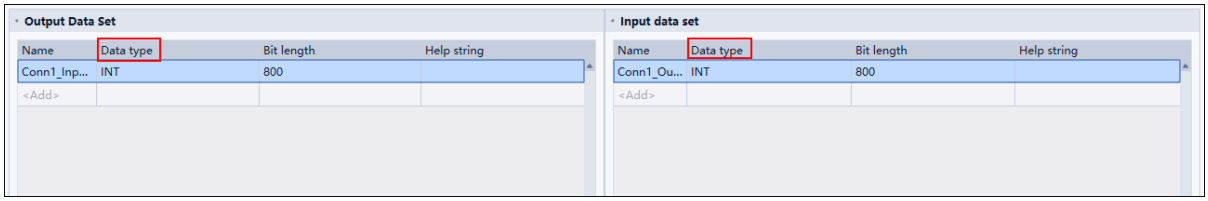
Product name: Supports renaming. When exporting the EDS file, the new name is recorded.

Minor version: When the physical device version is upgraded, this is used to manually adjust the records in the EDS file to align the physical device with the file content.

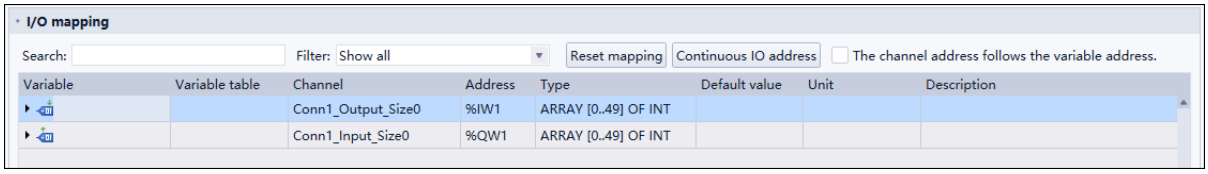
Export EDS file: Exports the current device configuration as an EDS file, which ensures consistency of information when used on the scanner end.

10.6.4.2.3.2 I/O Mapping

1. After configuring the connection information, especially the T->O size and O->T size, configure the data type to be transmitted in the "Connect" tab.



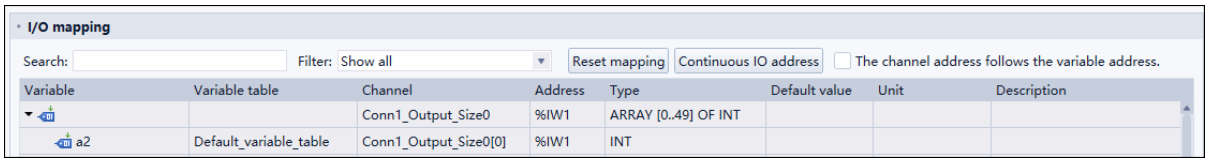
2. The configured variable information is displayed in the "I/O mapping" tab. The channel information corresponds to the established variable information, and a default address has been assigned, which can be modified.



3. The search function can be used to conveniently and quickly find variables. Filtering can be applied for partial display.

4. Click "Continuous IO address" to reassign the addresses of the variables to consecutive I/O addresses.

5. Create a mapping relationship. When creating a mapping, pay attention to the read/write permissions of the variables and ensure they match the variable addresses. I area addresses are used for input, and Q area addresses are used for output. If the relationship is incorrect, a compilation fault will be reported.

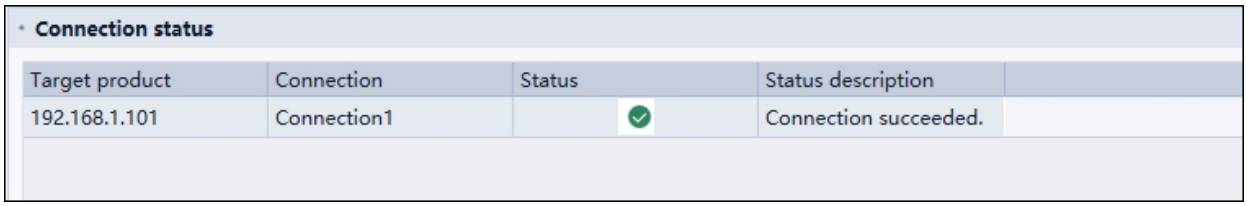


6. The "Reset mapping" button can quickly reset the entire variable mapping table.

7. The "Channel address follows variable address" option can be used to configure the allocation of I/O mapping addresses.
   - If it is not checked, the I/O mapping channel address is fixed. After binding a variable to the channel, the channel address is assigned to the variable.
   - If it is checked, binding is allowed only if the channel byte size matches the bound variable's byte size. After binding, the I/O mapping channel address is deleted, and the channel address follows the variable address.

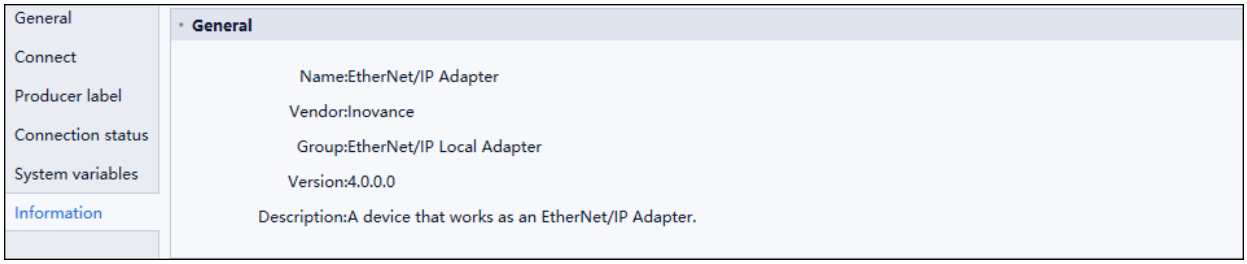
10.6.4.2.3.3 Connection Status

After switching the device to online, it starts listening for client connections. Once a client is connected, the relevant information can be viewed in the "Connection status" tab.



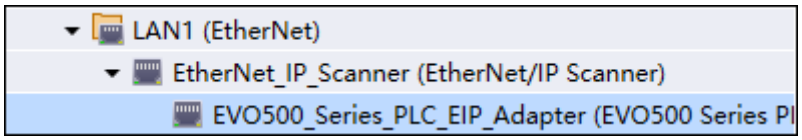
10.6.4.2.3.4 Information

This tab displays the following information for the current module: name, vendor, group, type, ID, version, description, and so on.

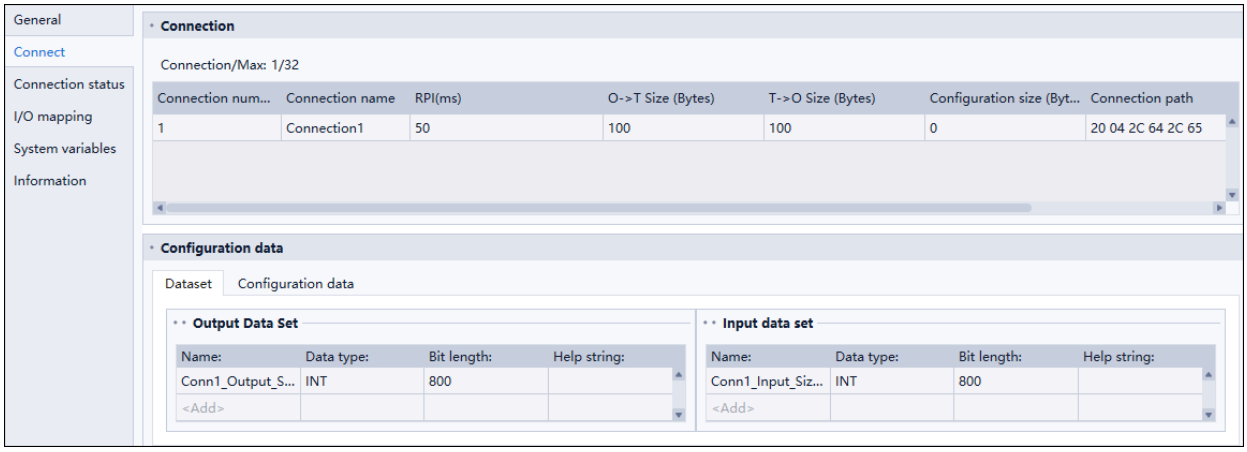


10.6.4.2.4 Connection Configuration (Class1)

Configuring the remote adapter under scanner



When adding an EtherNet/IP remote adapter, a connection path is generated by default, as shown in the figure below.



- Connection

  The "Connection" tab is used to configure connection parameters and count the total number of connections.

  - Connection parameter configuration

    Double-click a connection row to open the parameter configuration window.

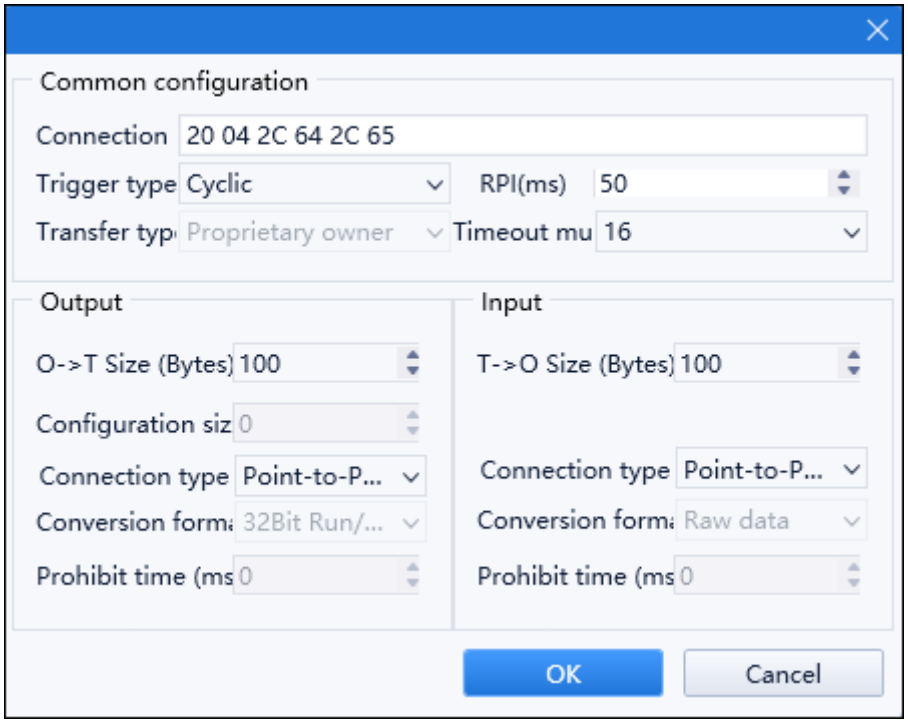


Table10-6-4-2-4-1 Connection configuration table

Category	Parameter	Note
Connection path	Composite configuration - Instance ID	Supports integers, with a range of 0 to 9999. Applicable to Generic_EtherNetIP_device devices.
	Composite consumption (O-->T) - Connection point	Consistent with the ID in the target settings, applicable to Generic_EtherNetIP_device devices.
	Composite production (T-->O) - Connection point	Consistent with the ID in the target settings, applicable to Generic_EtherNetIP_device devices.
General configuration	Connection path	Usually automatically calculated by the software, typically no need to change.
	Trigger type	Cyclic: Sends data according to the RPI cycle. State change: Triggers an immediate sending when data values change during each PLC scan cycle.
	RPI (ms)	Indicates the requested packet interval. The RPI of each node can be set independently without affecting each other.
	Transfer type	Proprietary Owner: Sets both the data transmission from the originator (O) to the target (T) and the data reception from the target (T) to the originator (O). Input only: Sets only the data reception from the target (T) to the originator (O). Listen only: Listens to multicast data without providing configuration or scheduling information for EtherNet/IP devices. Note: "O" refers to the originator and "T" refers to the target.

Category	Parameter	Note
	Timeout multiplier	If data is not received for multiple consecutive RPI cycles, communication is considered to have timed out. The timeout time is equal to the RPI multiplied by the timeout multiplier.
Output/Input	O->T size/T->O size	Displays byte size, which is automatically calculated based on data type and can be adjusted manually. The total number of bytes is constrained by the device's EDS file.
	Configuration size	Indicates the byte size of configuration data. The total number of bytes is constrained by the device's EDS file.
	Connection type	Multicast: Data is produced simultaneously to multiple consumers. Point-to-point: Data is produced individually to one consumer.
	Conversion format	Reads the content of the device's EDS file.
	Prohibit time (ms)	Configurable when "Trigger type" is set to "Upon status change". Indicates the minimum cycle for data refresh, must not be greater than the RPI value. The value ranges from 0 ms (the prohibit time is not effective) to 255 ms.

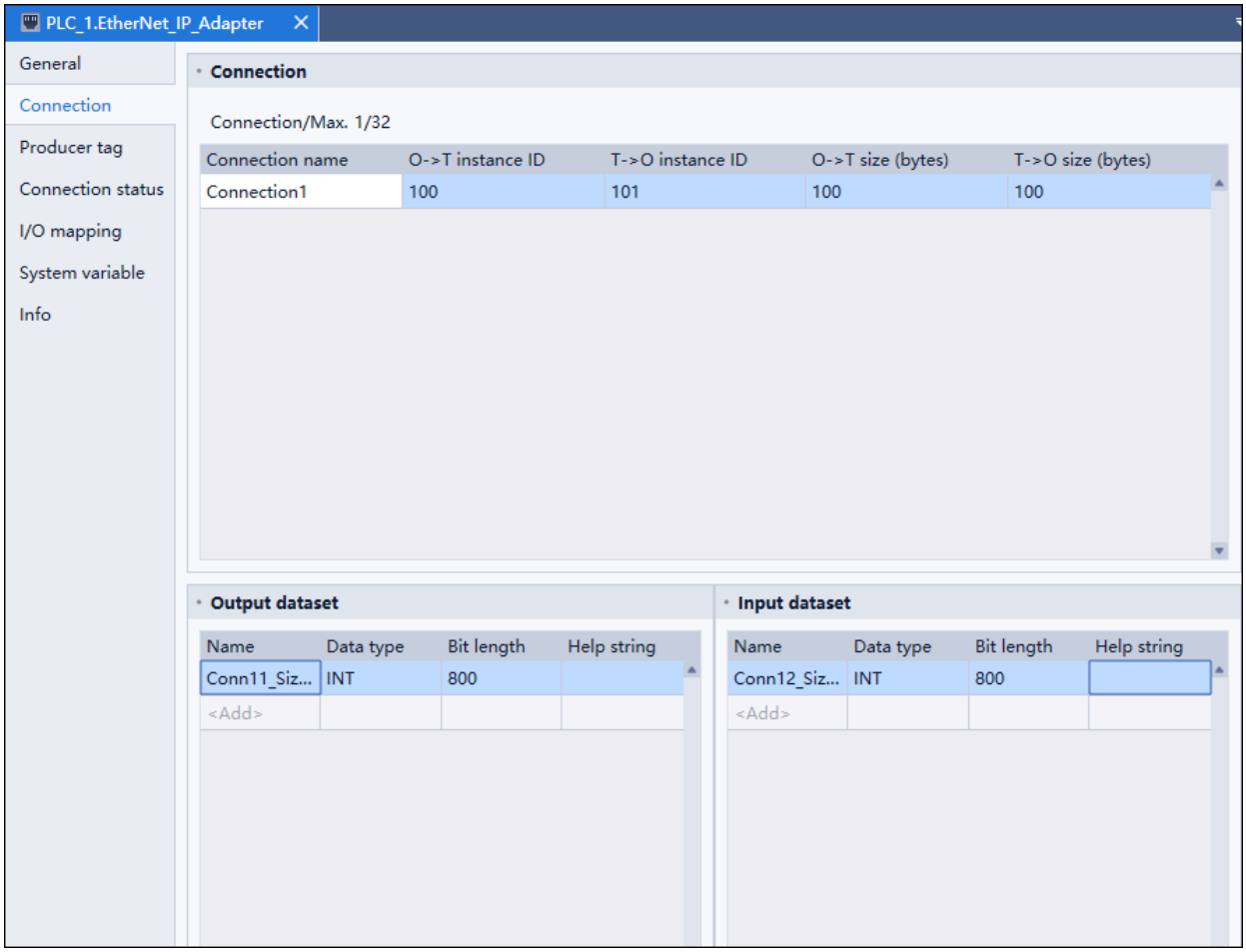
  - Connection count
    The connections consist of the connection rows (Class1 type) added under the corresponding network port, such as LAN1 (Ethernet), and the service message labels (Class3 type).
    The total number of connections is not fixed and depends on the PLC model. The EVO523 series PLC supports 32 connections.
    Right-click the blank space and select "Create connection" to add a new connection.

- Configuration data
  - Data set
    A connection will automatically generate corresponding input/output data sets. Users can manually configure the data sets, including changing names, data types, and adding or deleting data sets. These changes will be synchronized in real-time to the "I/O mapping" tab.
    The maximum total byte size of input and output data sets is 1444 bytes.
  - Configuration data
    The configuration data depends on the EDS file of the device. For example, the Generic_EtherNetIP_device supports adding configuration data, with a maximum total byte size of 1444 bytes.

Configuring local adapter



In the "Connection" tab, create a connection.



You can modify the size and name of the input and output data sets, and these changes will be synchronized in real-time to the "I/O mapping" tab.

10.6.4.2.5 Communication Between Producer and Consumer Labels

The definitions of producer and consumer labels are as follows:

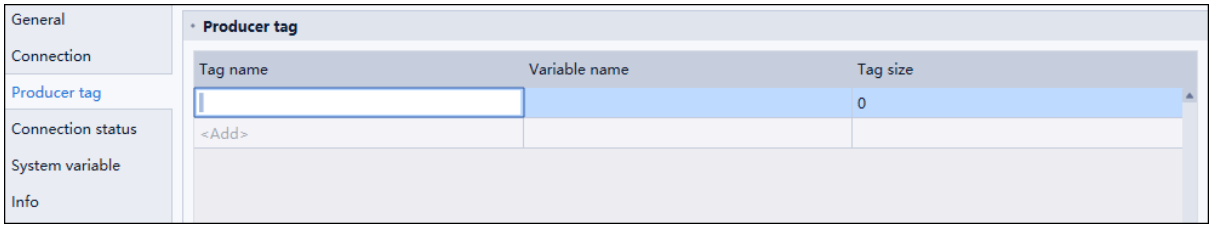
Label Type	Description
Producer label	A label provided by one controller for use by other controllers. Multiple controllers can consume (receive) data simultaneously. Without using logic, a producer label can send its data to one or more consumer labels (consumers). The producer label sends data according to the RPI of the consumer label.
Consumer label	A label that receives data from a producer label. The data type of the consumer label must match that of the producer label, including any array dimensions. The RPI of the consumer label determines the data update cycle.

Communication between producer and consumer labels requires configuration on both the local adapter and the remote adapter, and the content and order of the configurations must be exactly the same.

Configuring producer labels

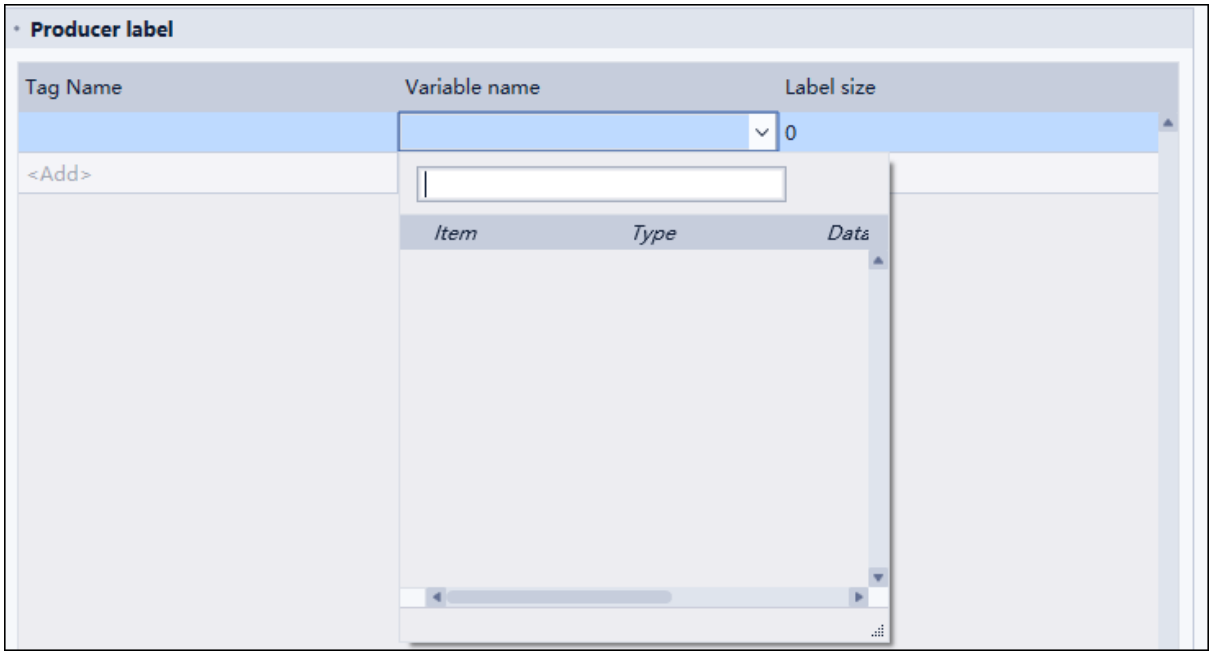
To configure producer labels in the local adapter, follow these steps:

1. Create a label.
   Click "<Add>" in the "Producer tag" tab to create a communication tag.



2. Associate variables with the label.

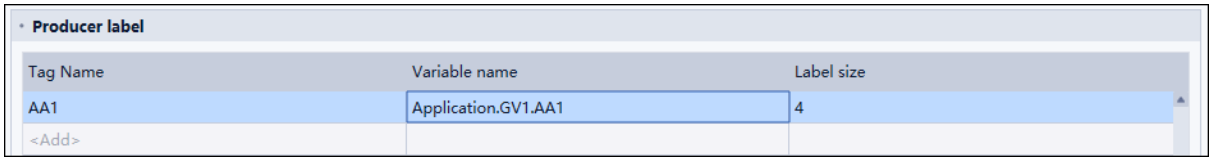
   Double-click a cell in the "Variable name" column to make it selectable.



   Select the target variable from the drop-down menu to complete the association.

   The variables added here must meet the following requirements:

   - 10.1.1.8.1 Standard data types, structures, and unions are supported.
   - The network public property supports **read-only** or **read-write**.
   - Constants, function blocks, and their members are not supported.
   - Variables in SYS_VAR(Global) are not supported.
   - Temporary variables are not supported.



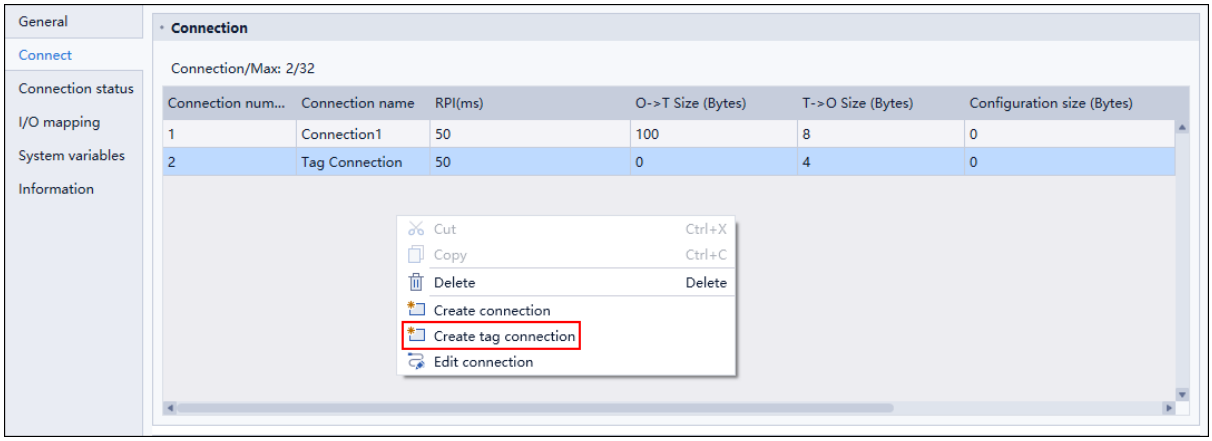
   Label name: Supports modification. It requires a combination of letters, numbers, and underscore characters, with a maximum length of 128 characters. When the new name duplicates an existing label name, a "_1" suffix will be automatically added to avoid renaming.

Configuring consumer labels

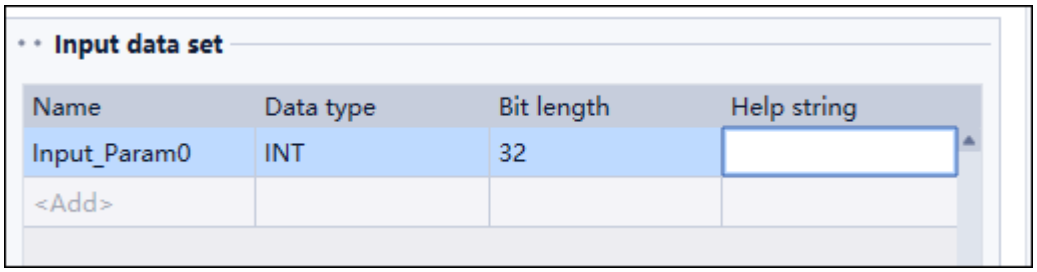
To configure consumer labels in the remote adapter, follow these steps:

1. Create a label.

   In the "Connection" tab, right-click the blank area and select "Create tag connection".

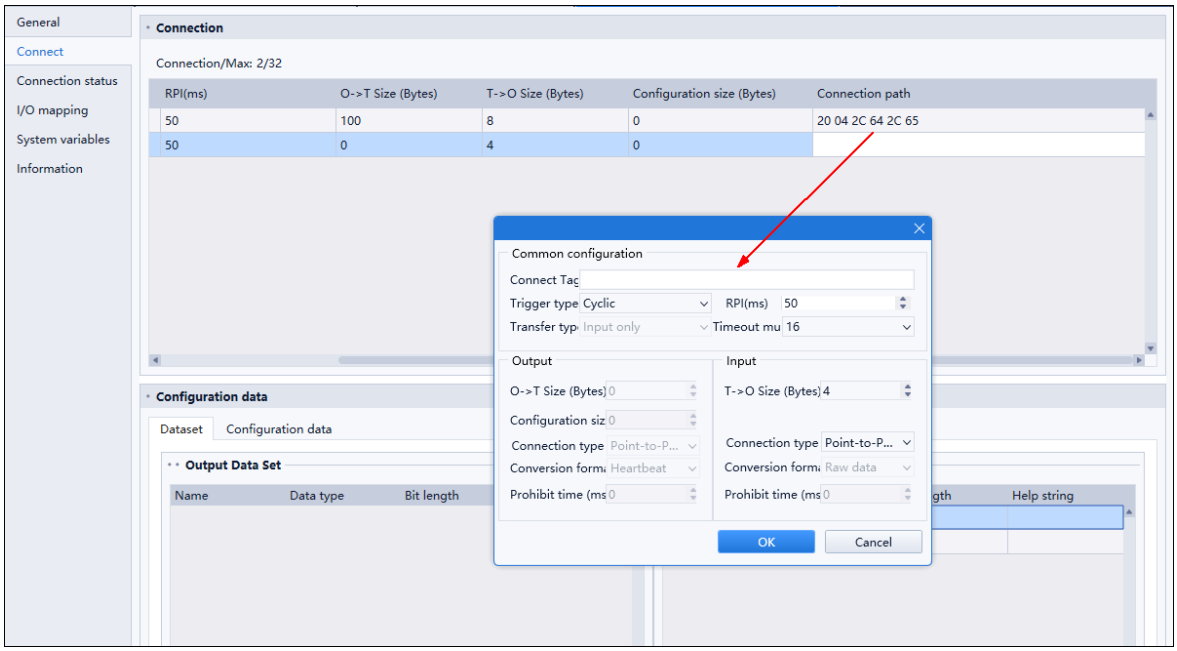


This will simultaneously generate a corresponding data row in the **input dataset area** of the current tab and a corresponding mapping row in the "I/O mapping" tab.



2. Set label information.
   - Connection

     Double-click the connection row to open the modification window.



     The "Connection path" in the connection row and the "Connection label" in the modification window are the same. Enter the label name of the producer label here.

     For other parameters, see the connection configuration table on the Connection Configuration (Class1) page.

   - Input dataset area

     Configure the name and data type of the label in the input dataset area, ensuring they match the content in the producer label exactly.

     Help string is used for remarks and does not participate in communication.

     These changes will be synchronized in real-time to the "I/O mapping" tab.

3. Associate variables in I/O mapping.

Double-click the cell containing the icon , click to open the input assistant, and select the variable to complete the association.

The network public property of the associated variables is not restricted.

With the above steps, the configuration for communication between producer and consumer labels is complete.

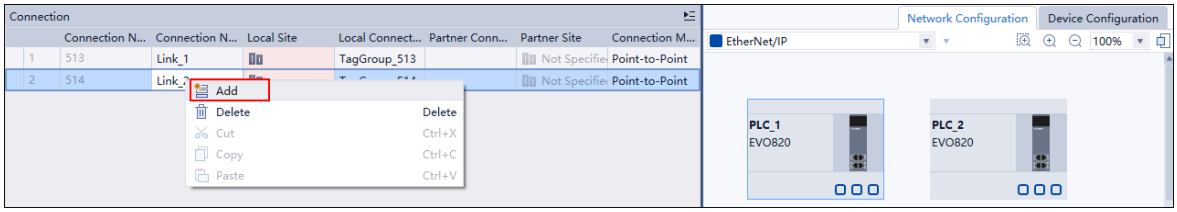
10.6.4.2.6 Communication Between PLCs

Multiple sets of PLC devices configured in the iFA Evolution software can communicate with each other via Ethernet, backplate, and other methods. The following uses the EVO800 series PLC as an example.

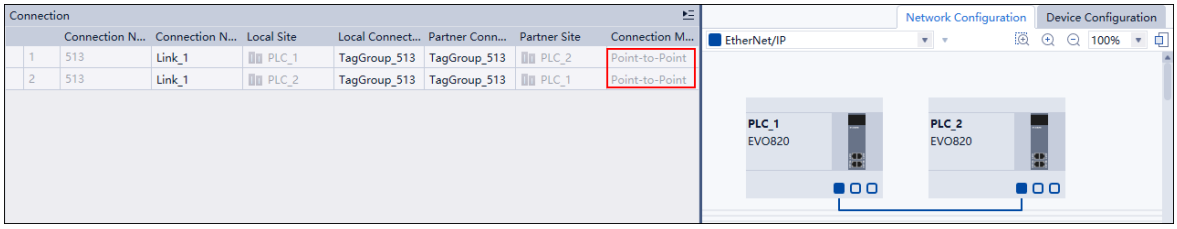
Configuring Ethernet communication

When communicating via Ethernet, the selected devices must support EIP communication.

1. Establish a connection.
   - Right-clicking creation
     Right-click in the connection table in the network configuration view, and select "Add" to create an unspecified connection channel and set the data communication method.

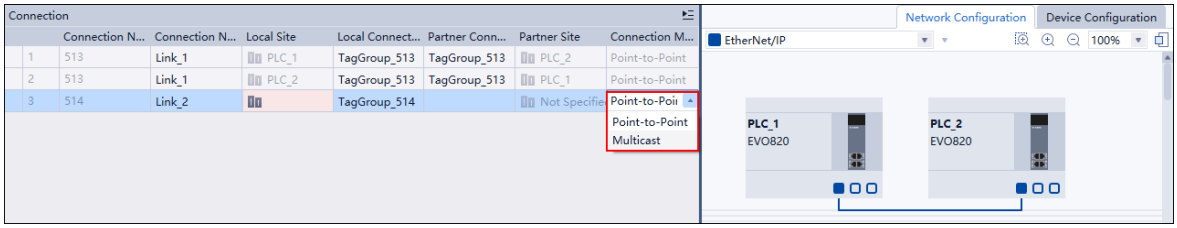


   - Drag-and-drop creation
     In the network configuration view, drag and drop to connect the CPU ports/Ethernet expansion module ports between devices. Since multicast is unidirectional one-to-many communication with one data sender and multiple data receivers, it does not support drag-and-drop creation. The connection type automatically generated by dragging is point-to-point. After a connection is created, it cannot be modified. If created incorrectly, it needs to be deleted and recreated.

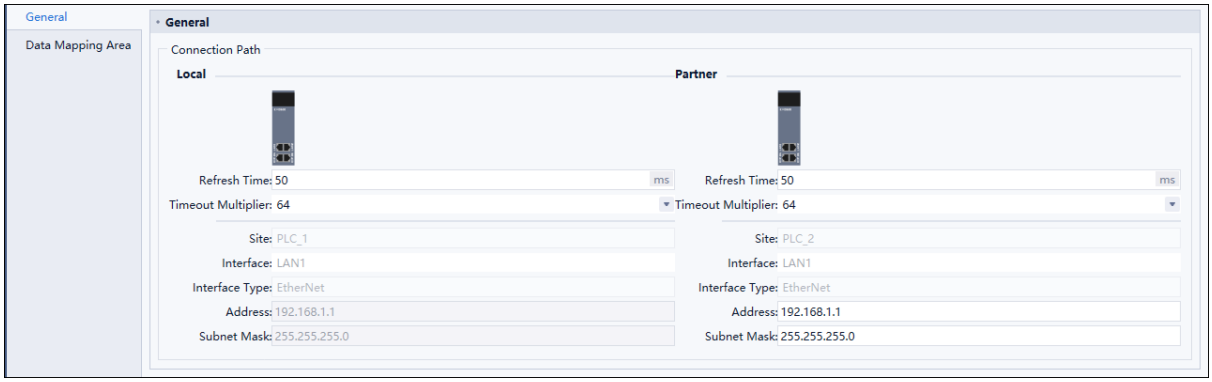


     After a connection is established through right-clicking or dragging, all generated connections are displayed in the same table. If both communication parties are included in iFA, the table displays two connections, each pointing from one end to the other, with the connection method defaulting to "point-to-point" and not supporting modification. If the communication partner is not included in iFA, the table displays only one connection, and the connection method supports selecting "point-to-point" or "multicast".

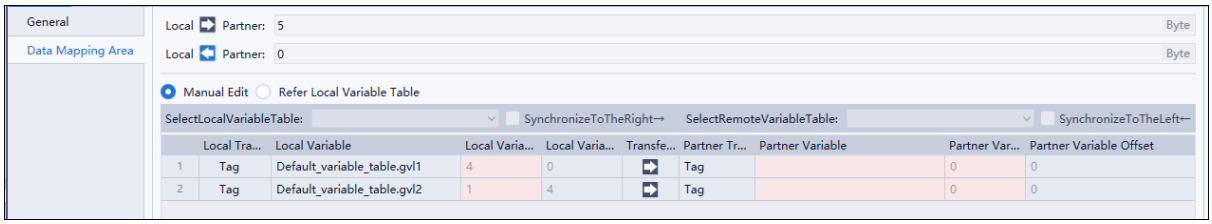
     For scenarios where both communication parties are included in iFA, the connection ID is automatically generated. If the partner is "unspecified", the ID is automatically generated but allows manual modification. This ID is used to identify the connection, including identifying point-to-point and multicast. The data sender generates the ID, and the data receiver initiates a connection request based on this ID to establish the ID and confirm the identity of the connection partner. The columns in the connection table is shown below.



2. After a connection is successfully created, click the connection to view or modify the relevant information in the "General" tab at the bottom of the interface.
   The refresh time only applies to the data receiver, and each data provider associated with a connection ID does not need to set a refresh time. The connection is automatically activated when the CPU/Ethernet expansion module has multiple interfaces, such as multiple Ethernet interfaces. The "Interface" entry supports drop-down selection, with the default interface being the one used to create the connection. When the CPU/Ethernet expansion module has only one interface for communication, the "Interface" entry is grayed out.



3. In the "Data mapping area" tab, set the total size of data interaction required by both communication parties (number of bytes sent/received). For Ethernet communication, the maximum communication data volume per direction for a single connection is 1444 bytes. Configure the data mapping relationship between the parties through binding variables. For point-to-point communication where both parties are within iFA, the schematic diagram is as follows:

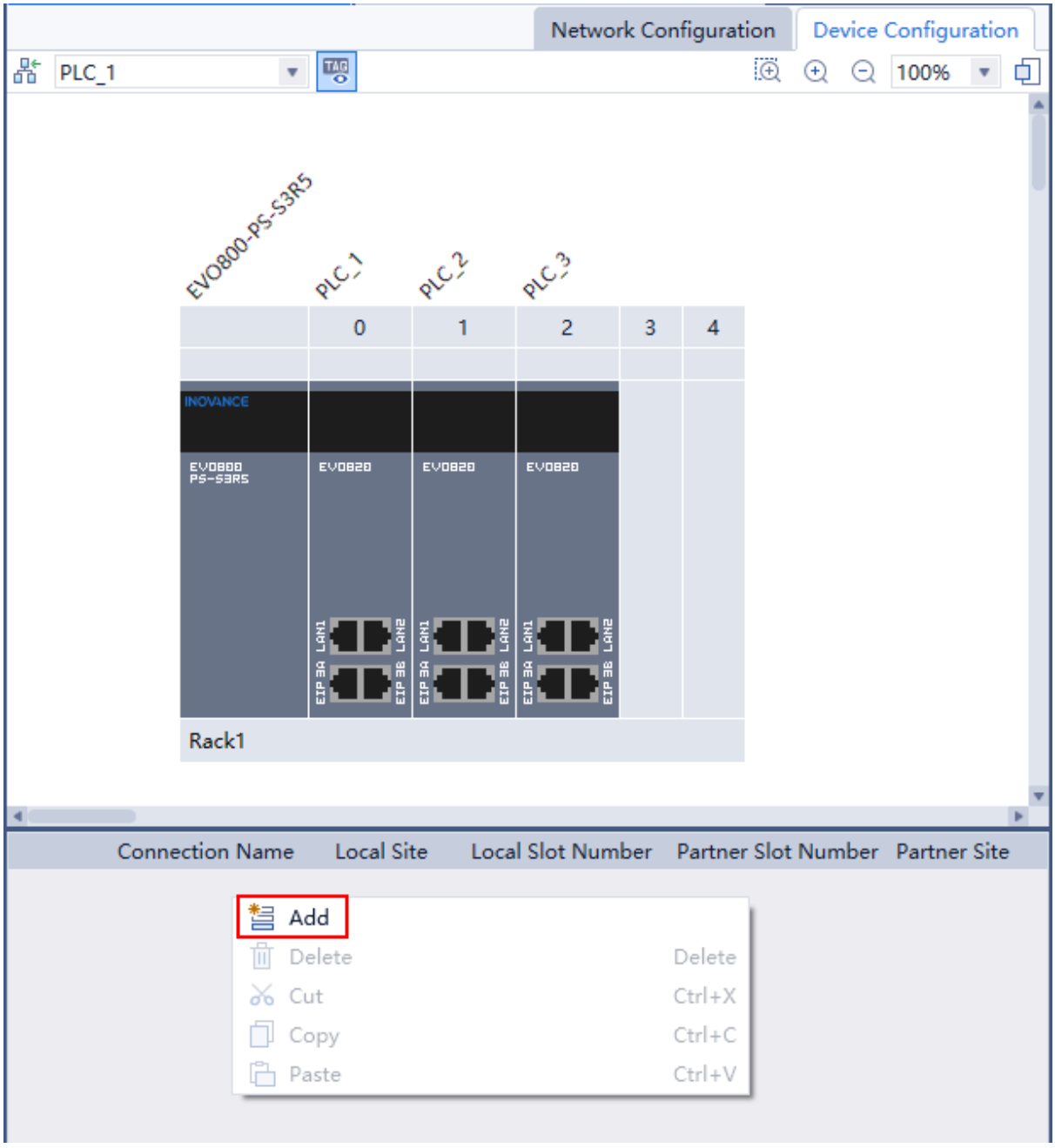


   - Each transmission area can be assigned a name, which has no direct association with PLC variables and serves as an explanation for the transmission area on the local PLC side.
   - Variables: Users can select a PLC variable by dragging or mouse selection. The variable must be network-public with a fixed length, or at least readable for output variables and writable for input variables, otherwise it cannot be dragged or selected.
     - In the detailed view, select one or more variables and add them to the data mapping area by dragging and dropping.
     - In the variable table, select one or more variables to copy and paste into the data mapping area.
   - Offset: Automatically calculates the offset of each variable's data area relative to the starting position in the entire transmission area (byte stream).
   - Length: iFA automatically calculates and displays the total data length of the selected variables for user reference.
   - Communication direction: Click to switch the direction, which indicates the direction of data transmission.

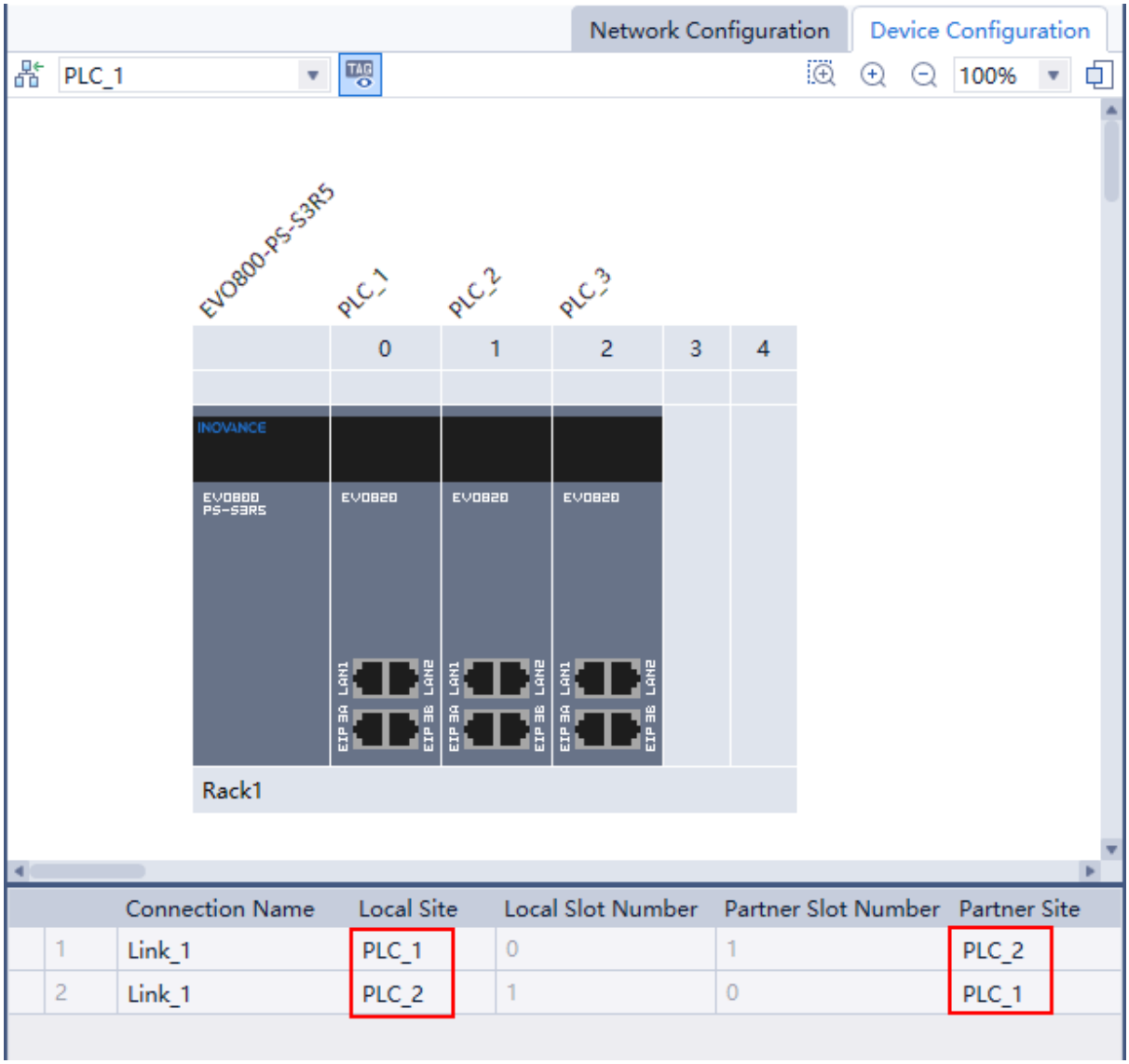
Configuring backplate communication

After configuring the EVO800 series multi-machine (single rack multi-CPU) mode, multiple CPUs within a single rack support communication via the backplate.

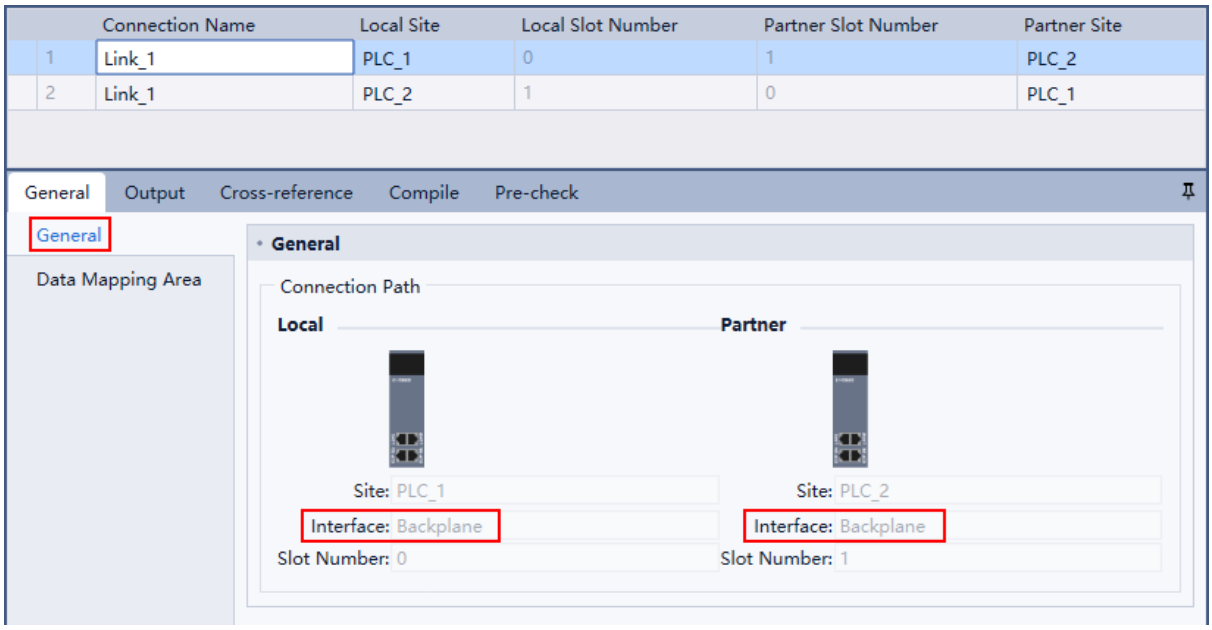
1. In the connection table area below the device view, right-click and select "Add" to create a new connection.



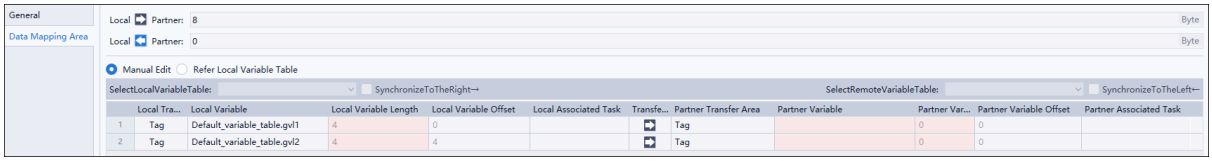
2. In the created connections, select the communication partner through the drop-down lists of the local site and partner site columns. Since the communication partner in this scenario is the same rack CPU, the local sites and partner sites in both connections correspond to each other.



3. After the connection settings, click the connection to view the detailed information about the connection in the "General" tab below, where the interface type is backplate.



4. In the "Data mapping area" tab, configure the data mapping relationship between the communication parties by binding variables. Click the cell in the "Transfer direction" column to switch the data transfer direction.



Associated tasks: Select variables from the drop-down list to bind the variables with the tasks created in the PLC. No variable is bound by default. When a task is bound locally, the partner must also bind a task, otherwise a fault will occur.

10.6.4.2.7 Communication with HMI

The PLC devices configured in the iFA Evolution software support communication with other devices (such as HMIs) via the EtherNet/IP protocol.

The software supports communication configuration for the following two scenarios:

- PLC and HMI are in the same project.
- PLC and HMI are in different projects.

This section uses the communication connection between the LAN1 port of the EVO820 redundant system and the LAN1 port of the HMI as an example.

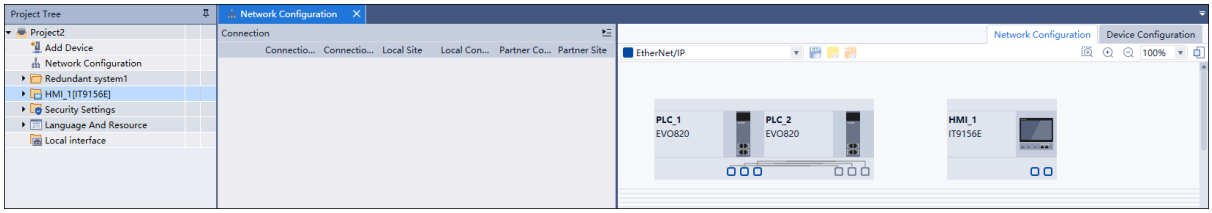
Communication configuration for PLC and HMI in the same project

Prerequisites

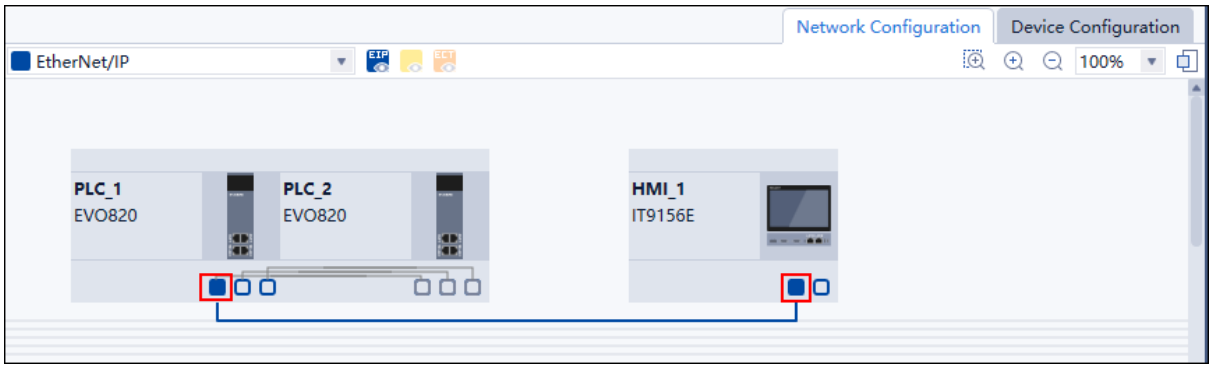
- PLC and HMI have been created. For specific operations, see 7 Project Deployment.
- The IP addresses of the LAN1 port of the EV820 redundant system and the HMI have been set in the same subnet. For specific operations, see 10.11.5 Network Settings.

Procedure

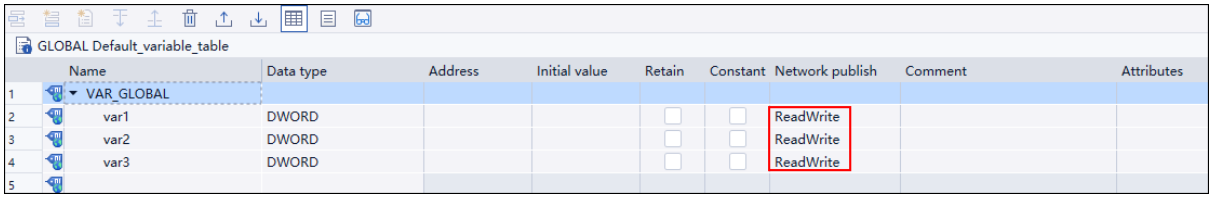
1. Double-click "Network configuration" in the left project tree to open the "Network configuration" view.



2. Drag the LAN1 port of IT9156E HMI_1 to the LAN1 port of EVO820 PLC_1 to establish an EtherNet/IP connection.

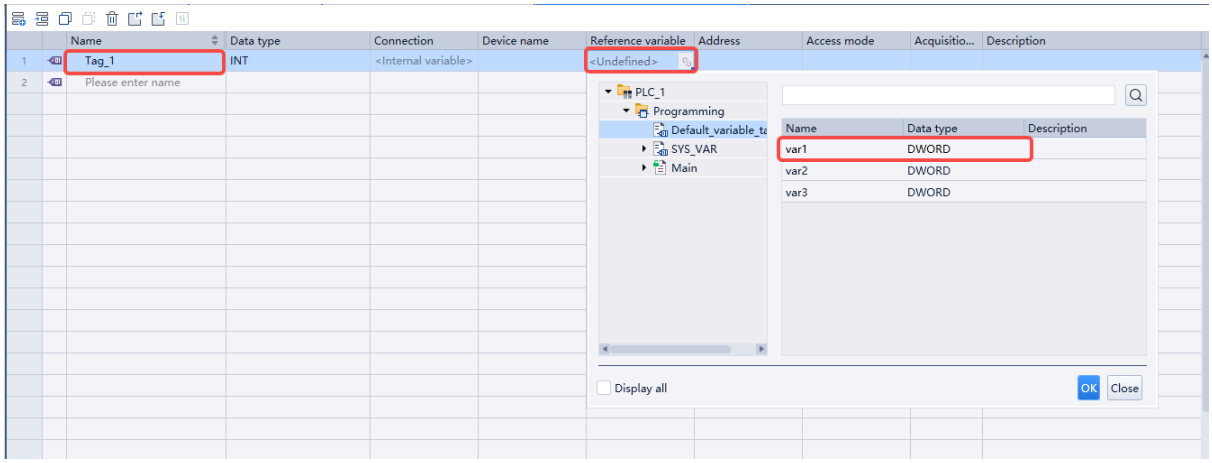


3. In the project tree, under the redundant system node "PLC_1 > Programming > Default variable table", double-click the variable to be read or written and change the network public property to "Read/Write".

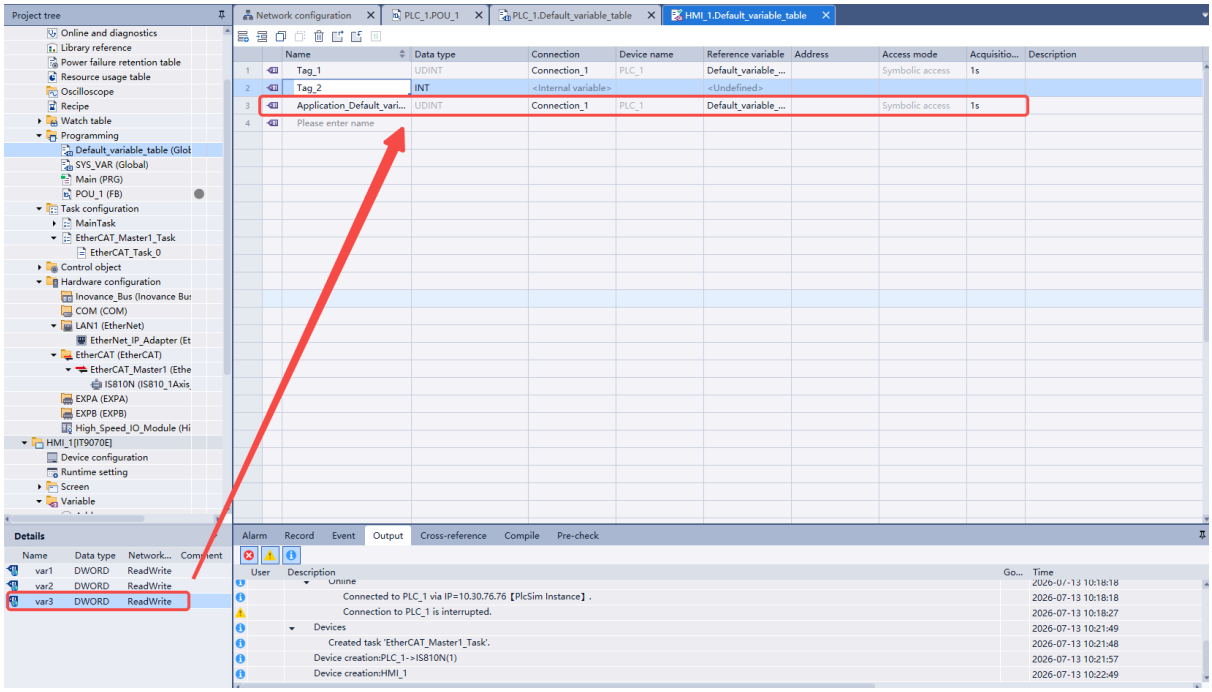


4. In the variable table of HMI_1, reference the variables from the PLC_1 variable table whose network public property has been changed to "Read/Write".

5. In the project tree, under "HMI_1 > Variables > Default variable table", click the cell in the "Reference variable" column, select the variable from the PLC_1 variable table, and click "OK" to associate the referenced variable.



6. In the project tree, under the redundant system node "PLC_1 > Programming > Default variable table", drag the variable in the "Details" window to the HMI_1 variable table to associate the referenced variable.



7. Compile and download the projects for PLC_1 and HMI_1.

8. Observe the variable values on the HMI screen.

Communication configuration for PLC and HMI in different projects

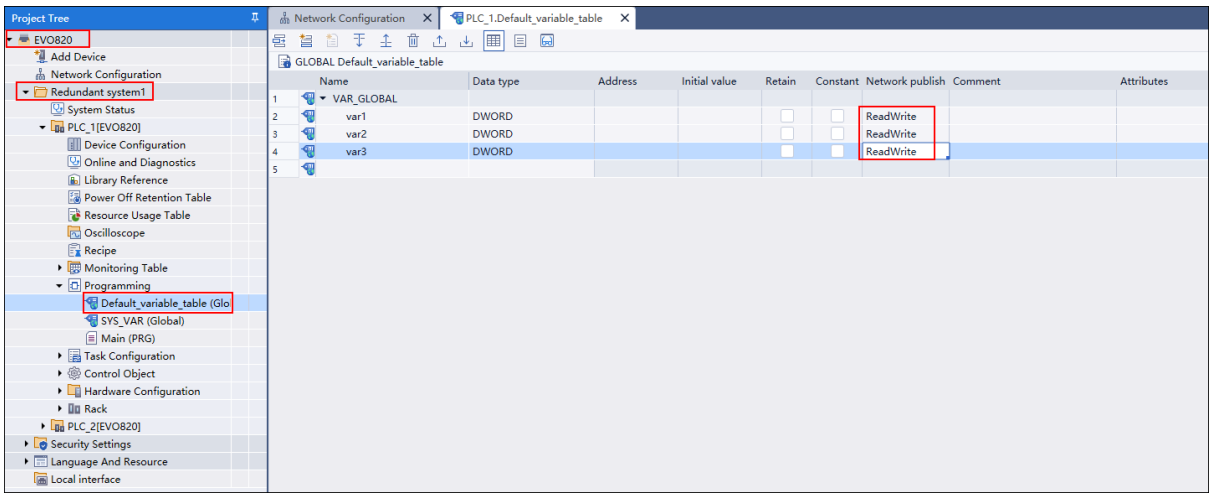
Prerequisites

- PLC and HMI have been created. For specific operations, see 7 Project Deployment.

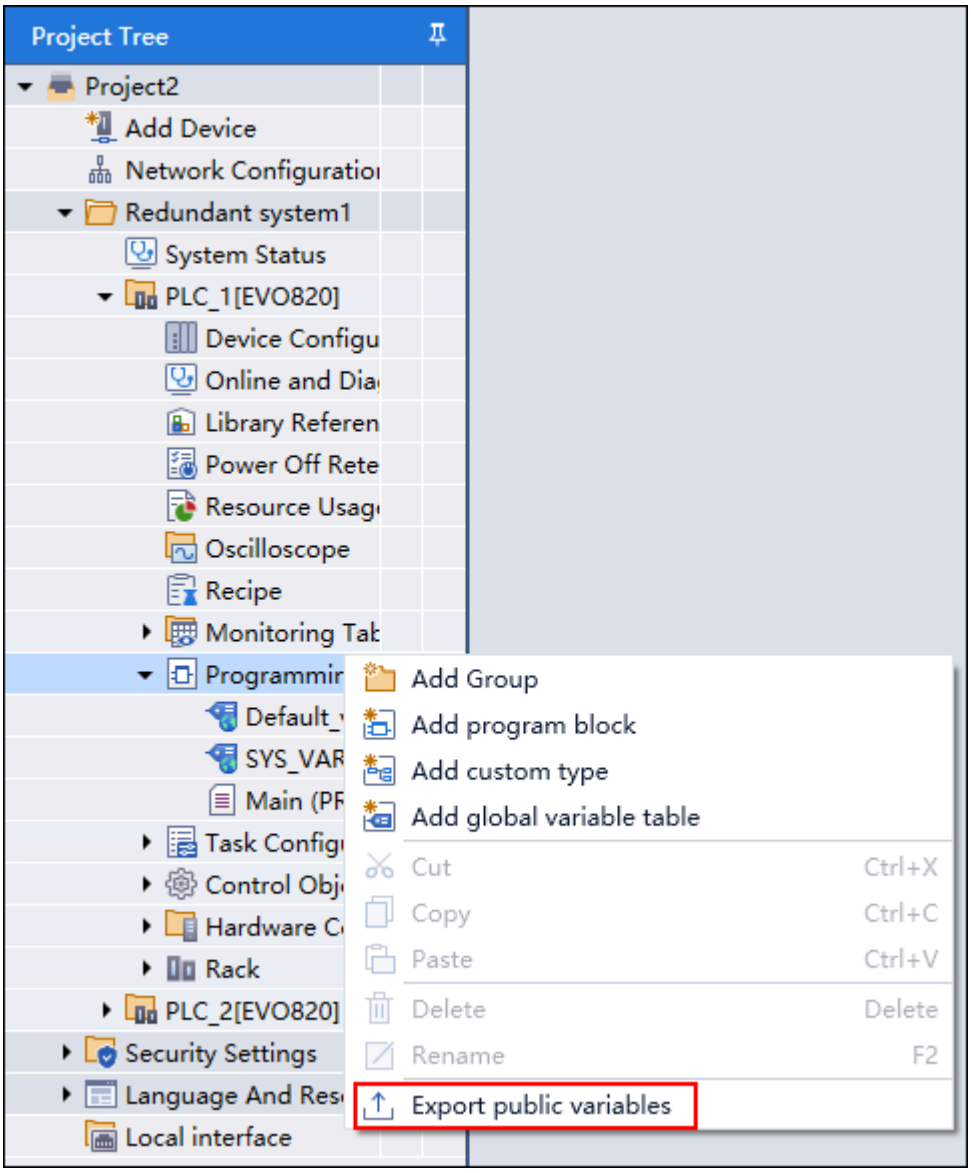
- The IP addresses of the LAN1 port of the EV820 redundant system and the IT9156E have been set in the same subnet. For specific operations, see 10.11.5 Network Settings.

Procedure

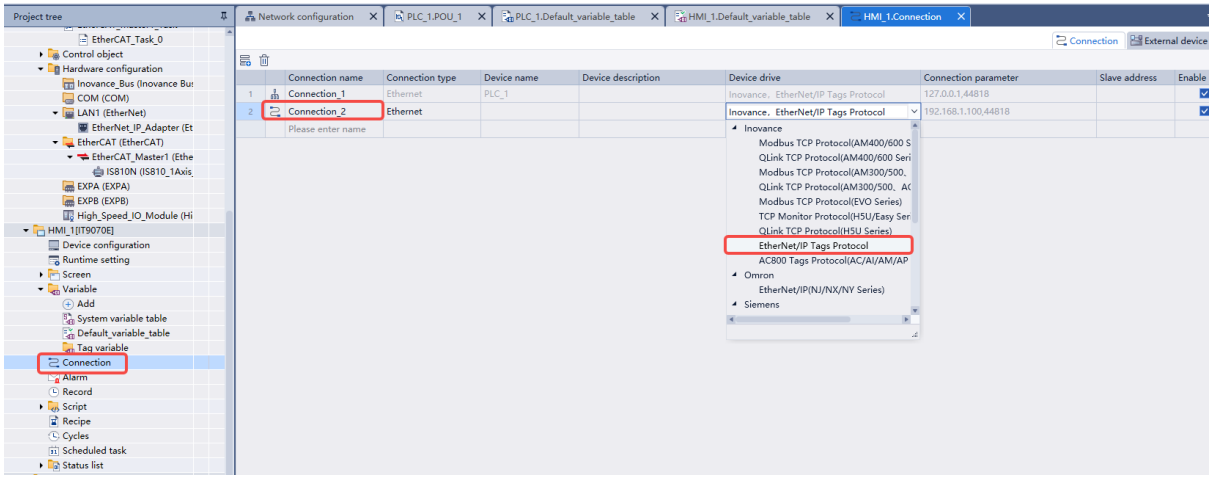
1. In the project tree, under the redundant system node "PLC_1 > Programming > Default variable table", double-click the variable to be read or written and change the network public property to "Read/Write".



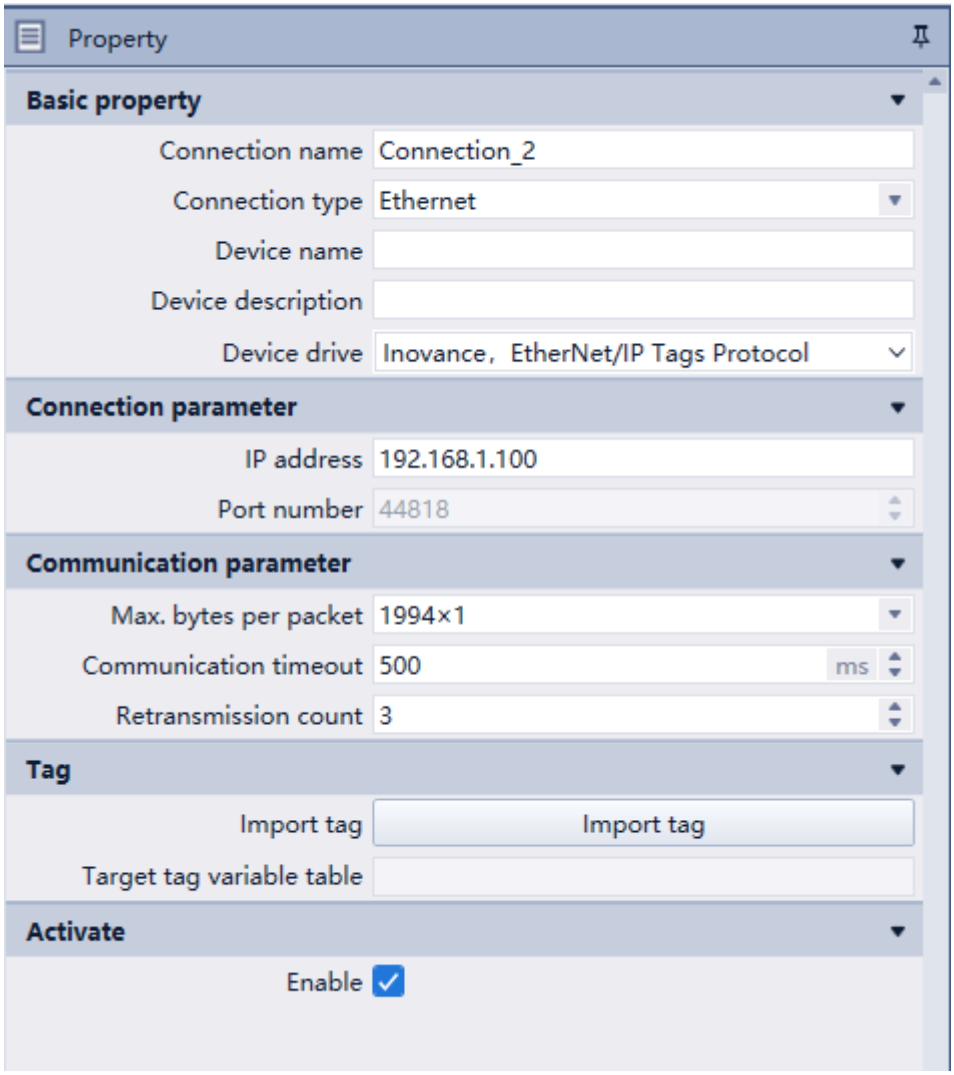
2. Right-click "Programming" under the redundant system node "PLC_1" in the project tree, and select "Export public variables" to export the public variables. The default filename for the public variables is "plc.PLC_1.Application.xml".



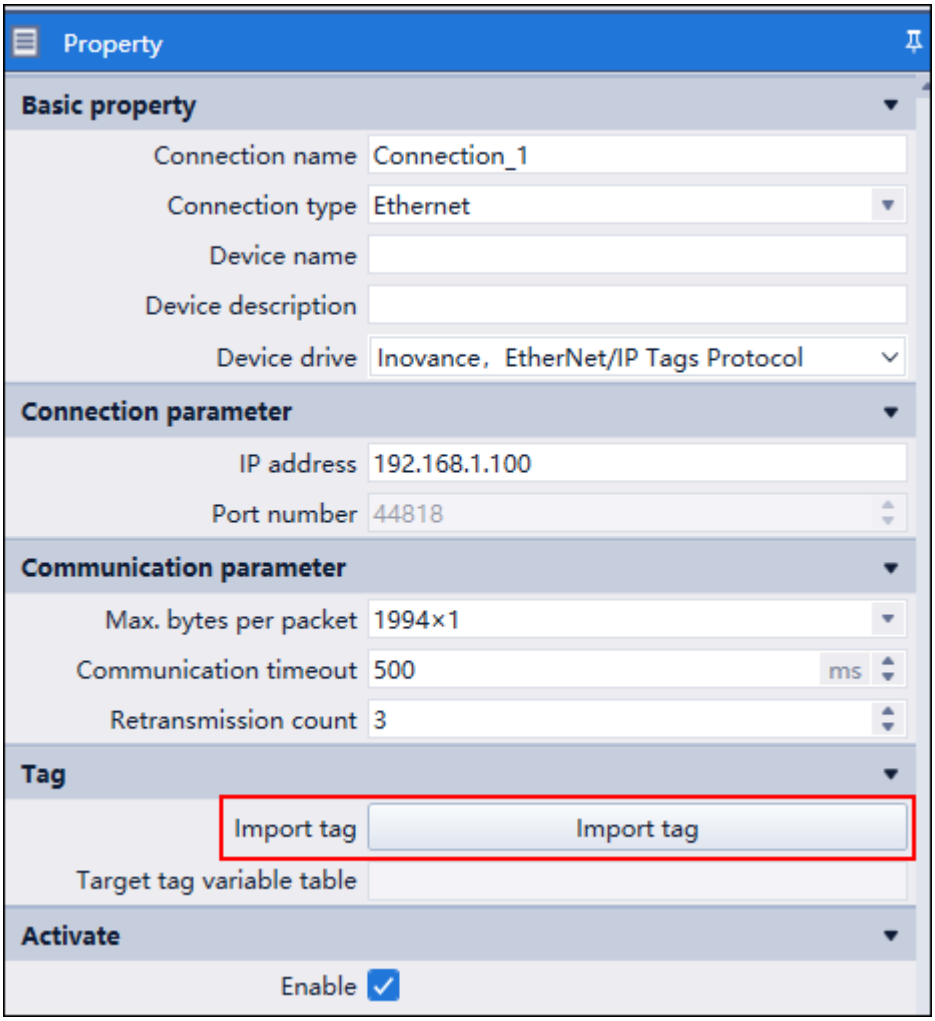
3. In the HMI project tree, double-click "Connections" and set the connection name to create the connection. Select "Inovance, EtherNet/IP Tags Protocol" for the device driver.



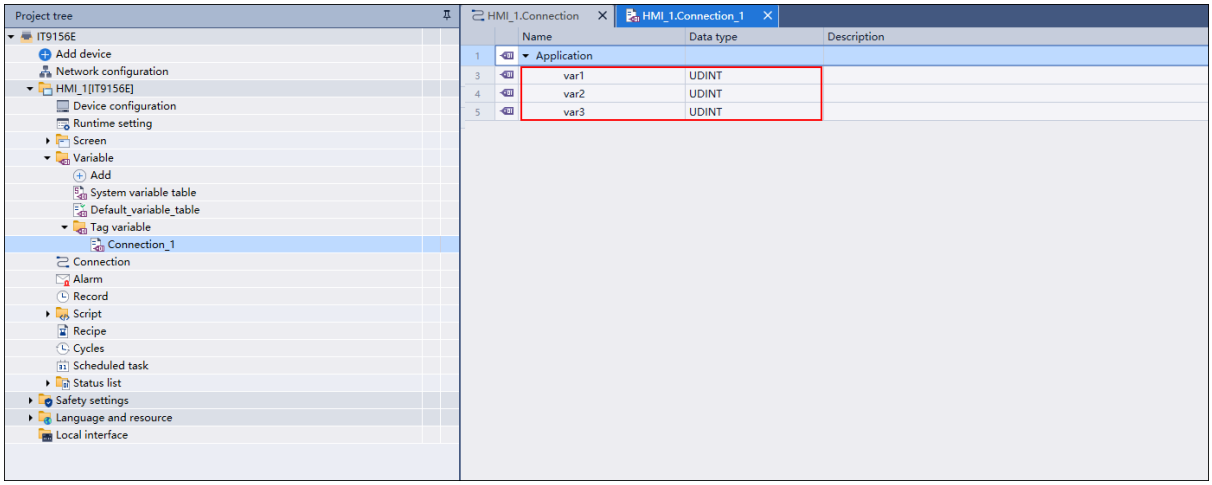
4. In the "Property" area of the HMI project on the right side of the interface, set the IP address of the LAN1 port of the EVO820 device to be connected.



5. Click "Import tag" to import the public variables file plc.PLC_1.Application.xml exported from the EVO820 project to the IT9156E project.



After successful import, the public variables defined in the EVO820 project are displayed in the HMI project.



6. Compile and download the projects for PLC_1 and HMI_1.
7. Observe the variable values on the HMI screen.

Data encoding and transmission methods

The variable types used in EIP label communication mainly include basic data types, BOOL types, STRING types, and structure types. The following explains the encoding and transmission methods for these data types respectively.

Basic data types

For basic data types, the encoding and data size for each type are as follows:

Type	Encoding (Hex)	Description
SINT	C2	8-bit integer
INT	C3	16-bit integer
DINT	C4	32-bit integer
LINT	C5	64-bit integer
USINT	C6	8-bit unsigned
UINT	C7	16-bit unsigned
UDINT	C8	32-bit unsigned
ULINT	C9	64-bit unsigned
REAL	CA	32-bit floating-point
LREAL	CB	64-bit floating-point
BYTE	D1	8-bit bit string
WORD	D2	16-bit bit string
DWORD	D3	32-bit bit string
LWORD	D4	64-bit bit string

BOOL types

For BOOL types, 0xC1 encoding is used for transmission. A BOOL variable occupies one byte, with the least significant bit being valid. For BOOL[n], each BOOL occupies one byte. Take BOOL[17] as an example, if n elements are requested, n bytes will be returned; if the entire array is requested, 17 bytes will be returned.

STRING types

For STRING types, 0xD0 encoding is used for. During data transmission, variable-length transmission is performed according to the actual length of the string. For example, a variable of type STRING[128] containing "ABCDEFG" will transmit 7 bytes of data.

NOTE

- The null terminator "\0" is not included in the transmission.
- STRING type does not support reading or writing the entire array at once; only one element can be accessed at a time.

Structure types

When transmitting structures, byte alignment issues are involved. The following explains the alignment rules for different structure members.

- **Members of basic data types**

LINT: Starts and ends on a 64-bit boundary.
DINT: Starts and ends on a 32-bit boundary.
REAL: Starts and ends on a 32-bit boundary.
INT: Starts and ends on a 16-bit boundary.
SINT: Starts and ends on an 8-bit boundary.

Array: Elements in an array are closely arranged and aligned according to the data type of the basic elements in the array.
Structure: A structure is aligned to an integer multiple of the length of its largest member.

- **Members of BOOL type**

BOOL type: When the member is of BOOL type, it is aligned to 2 bytes, occupying 2 bytes, with the least significant bit being valid.

BOOL[n] type: When the member is a BOOL array, each BOOL is tightly arranged by bits, with a total length of 16 bits (i.e., aligned to 2 bytes). For example, BOOL[16] occupies 2 bytes, BOOL[10] also occupies 2 bytes, while BOOL[17] occupies 4 bytes. Bits are valid sequentially from the least significant to the most significant.

- **Members of STRING type**

A STRING type can specify a length, such as STRING<25> and STRING<100>. Strings in structures can be considered as arrays of SINT, aligned to 1 byte, and transmitted according to its maximum length. For example, STRING<25> occupies 25 bytes and STRING<100> occupies 100 bytes.

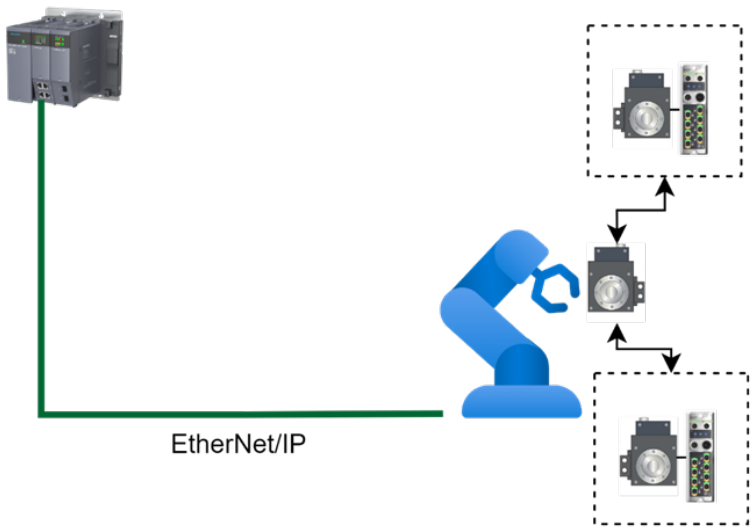
> NOTE
>
> EIP label communication supports up to 8 levels of structure nesting.

10.6.4.2.8 Quick Connect Communication (EVO700, EVO700C, EVO800, APC760C)

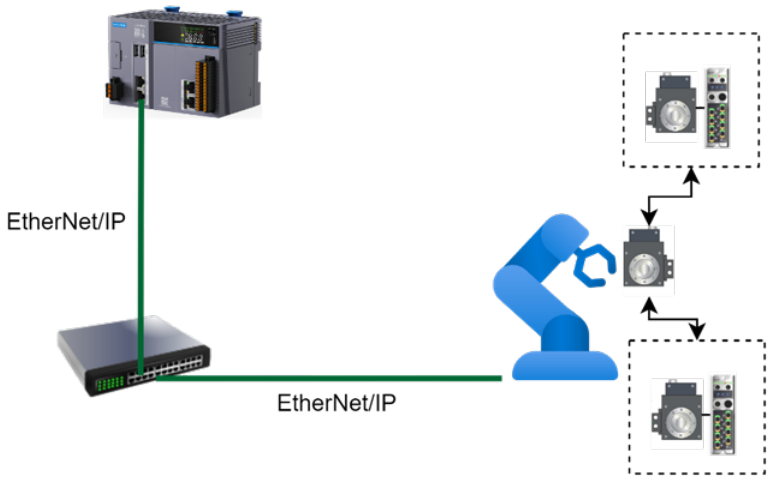
The PLC devices configured in iFA Evolution software support fast power-on and integration into EtherNet/IP networks when connected to ODVA-compliant EtherNet/IP target devices with the quick connection function.

Typical quick connection topologies can be simplified into the following two forms:

- Method 1: The PLC connects directly to the quick connection adapter device via its local Ethernet port on the EtherNet/IP network (requires network port configuration on the PLC).



- Method 2: The PLC device communicates with the quick connection adapter through a managed switch. The switch must support adjustment of its port configuration parameters, and the link establishment time must be less than 300 ms when using the quick connection port configuration.



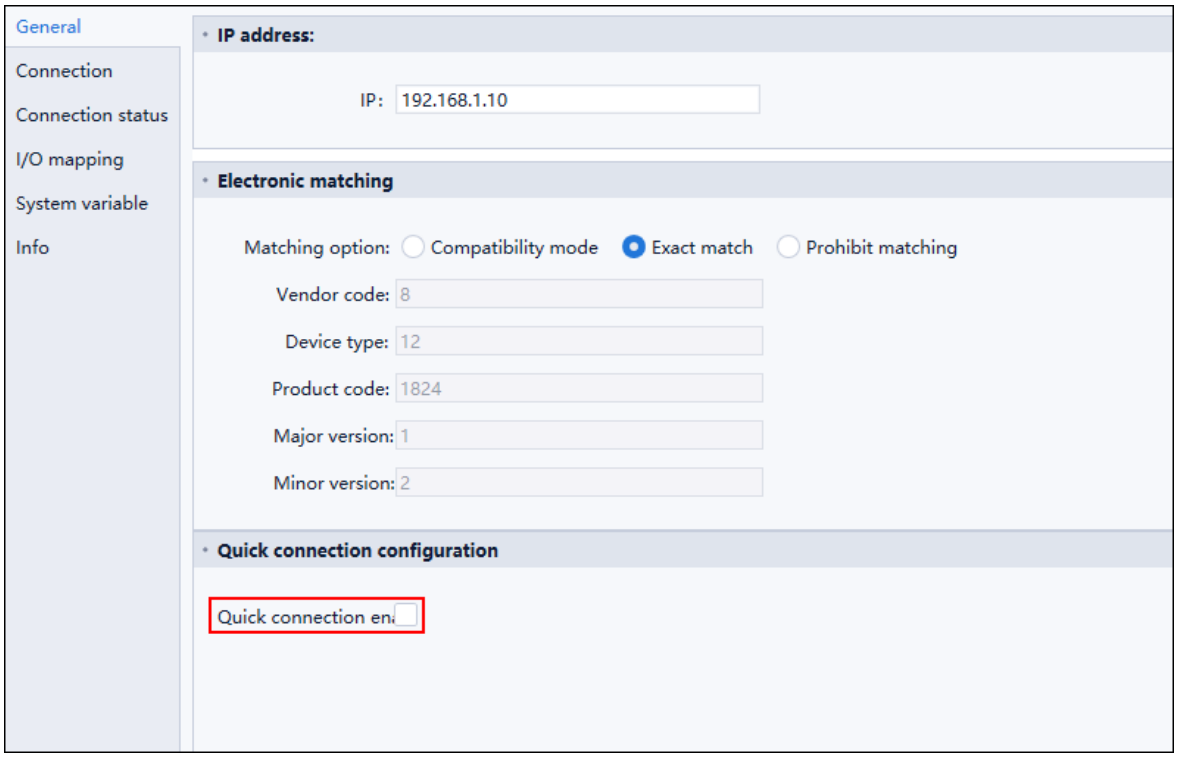
Quick connection communication configuration

Prerequisites

To implement quick connection via EtherNet/IP, both the selected PLC and the remote adapter device must support the EIP quick connection function. If a managed switch is used, it must support modifying port parameters.

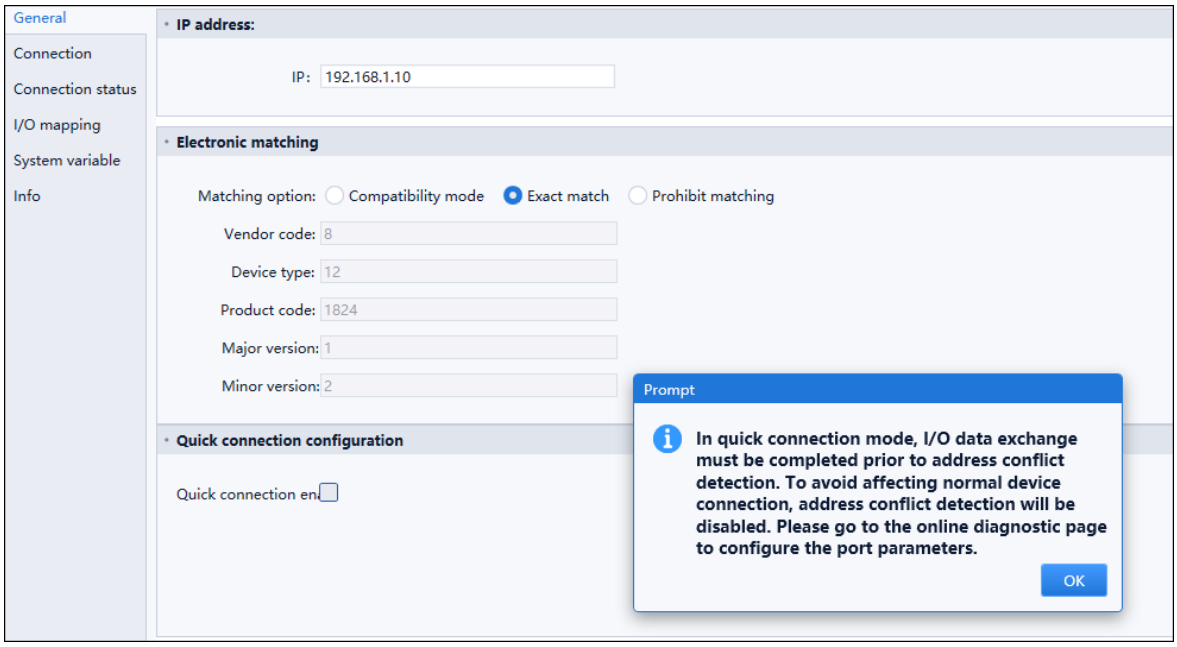
Procedure

1. Enable the quick connection function.
   a. Add an ODVA-compliant remote adapter device that supports quick connection under the PLC project. The quick connection switch is displayed in the "General" settings interface.



> **NOTE**
>
> The configuration steps for the quick connection slave device are the same as those for a common remote adapter device. Refer to 10.6.4.2.2.2 General Settings of Remote Adapters and 10.6.4.2.4 Connection Configuration (Class1) for general configuration and Class1 connection configuration.

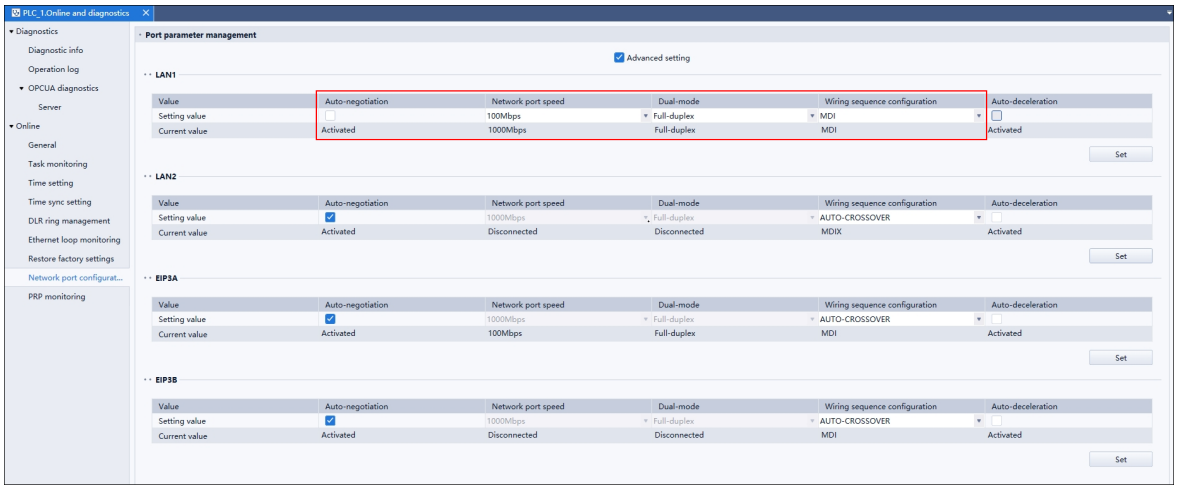
b. Enable quick connection and click "OK" in the pop-up prompt box.



2. Modify port parameters.

To reduce the negotiation time of the quick connection adapter device on the network, the port parameters of the connected peer device must be modified.

- If the peer device is a PLC, refer to 10.11.22 Network Port Configuration to modify the corresponding port parameters. Disable auto-negotiation, set the port speed to 100 Mbps, duplex mode to full-duplex, and wiring sequence to fixed MDI or MDIX.



- If the peer device is a managed switch, manually modify the corresponding port parameters through the Web interface or command-line control.

NOTE

- Modifying the port parameters is mandatory. If a switch is connected to the quick connection device and the switch does not support modifying the above configuration items, the port negotiation time may be delayed, potentially failing to meet the performance requirements of quick connection.
- When configuring the wiring sequence, attention must be paid to the matching relationship with the network cable (twisted-pair cable); otherwise, connection failure may occur.
  There are two types of network cable wiring sequences, used to match MDI and MDI-X interfaces:
  - Straight-through cable
    Pin assignments are identical at both ends (T568A<--> T568A or T568B<--> T568B).
  - Crossover cable
    One end uses T568A pin assignment, the other uses T568B (T568A<--> T568B).

The following table shows the wiring sequence type of device interfaces and the supported cable types:

No.	Device Interface Type	Supported Cable Type
1	MDI<-->MDI-X	Straight-through cable
2	MDI-X<-->MDI-X	Crossover cable
3	MDI<-->MDI	Crossover cable

- If a switch is used to connect quick connection devices, ensure that other switch functions (such as VLAN, QoS queues, and STP) do not interfere with data communication between the port connected to the PLC and the port connected to the quick connection adapter.

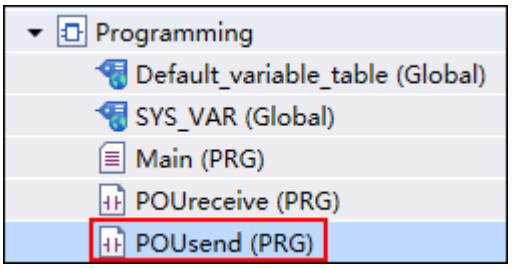
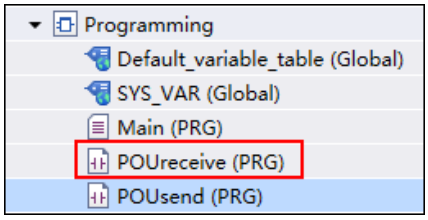
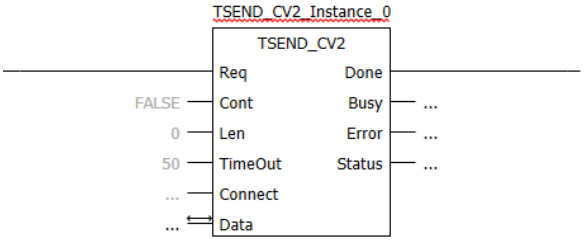
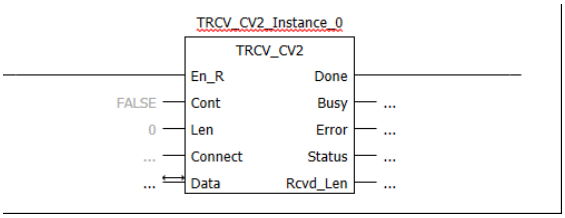
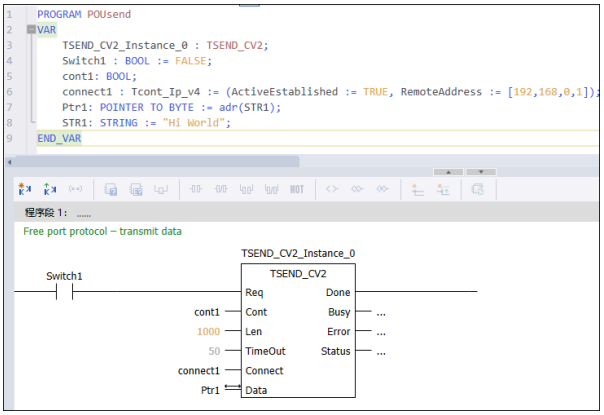
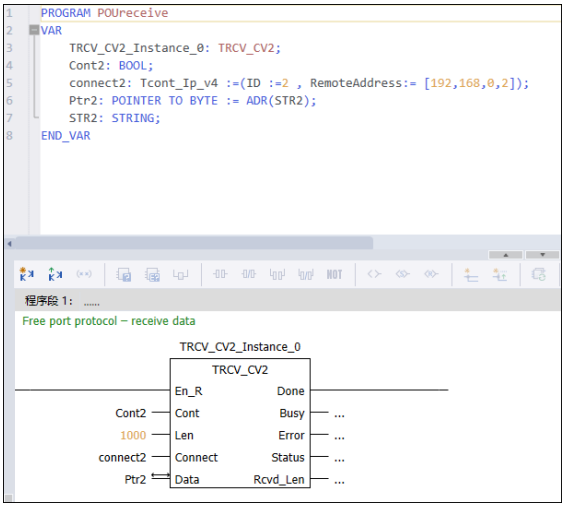
3. Quick connection communication
   After enabling quick connection and configuring port parameters, power on and off the quick connection adapter device connected to the EtherNet/IP network for quick power on, port negotiation, and integration into the EtherNet/IP network for Class1 I/O data exchange.

10.6.4.3 Free Network Port Communication

The free port protocol is used to achieve character data communication between PLC devices and third-party devices, with the physical interface being a network port. The transmission and reception communication of this protocol is completed by the dedicated library NetFreePort. For detailed instructions on the function blocks, see the instruction manual.

This section takes transmitting "Hi, World" at irregular intervals as an example to explain the configuration process.

Step	Transmitter	Receiver
1	Create a PLC project.	
2	Reference the NetFreePort library. For reference operations, see 12.3 Library Reference.	

Step	Transmitter	Receiver
3	Create a PRG type POU in LD language, named POUsend. 	Create a PRG type POU in LD language, named POUreceive. 
4	Add a normally open contact and a TSEND_CV2 function block instance in POUsend. 	Add a TRCV_CV2 function block instance in POUreceive. 
5	Name the normally open contact Switch1. Declare these variables between VAR and END_VAR and connect them to the function blocks. For parameter settings, see 18.4.8.5 TSEND_CV2. 	Declare these variables between VAR and END_VAR and connect them to the function blocks. For parameter settings, see 18.4.8.4 TRCV_CV2. 
6	Check that the port numbers (RemotePort, LocalPort) are the same for both the receiver and transmitter. Ensure that the IDs of the transmitting or receiving blocks are not duplicate.	
7	Call POUsend in MainTask.	Call POUreceive in MainTask. Set the task cycle of POUreceive to be at least twice that of POUsend to avoid receiving delays.
8	Download the controller and start running.	

Step	Transmitter	Receiver
9	Write TRUE to Switch1.	Observe the value changes of STR2. If the value is 'Hi, World', the configuration is successful.

During commissioning, pay attention to the status code of **Status**, and refer to the relevant explanations in instructions 18.4.8.5 TSEND_CV2 or 18.4.8.4 TRCV_CV2.

10.6.4.4 Disabling/Enabling an EtherNet Device

Disabling devices under the LAN1 (EtherNet) node

In iFA software, a certain EtherNet device can be disabled through commands or variable setting, thus "detaching" from the control system while retaining its configuration parameters and I/O mapping information. If a device fails and needs to be replaced, this function can reduce the time required for replacement.

- Disabling operation and effect
  A device can be disabled through commands or variable setting, each with different scopes. Command disabling applies to all devices except PLCs Variable setting requires that there be corresponding variables in SYS_VAR.
  - Command disabling
    Right-click a device under the "LAN1 (EtherNet)" node in the project tree and select "Disable device".
    In the "Network configuration" window, right-click an EtherNet/IP remote adapter and select "Disable device".
  - Variable setting
    When adding a device under the "LAN1 (EtherNet)" node, a corresponding variable is created in **SYS_VAR**, following the naming rule: J_device model.
    Setting member The member **bDisableEnable** disables the slave, and the setting method supports writing values in **SYS_VAR** or assigning values in user programs.

    GLOBAL SYS_VAR

	Name	Data type	Address	Retain	Network publish	Comment
1	VAR_GLOBAL					
2	LAN1_EIPSingle_STATUS	Array[1..32] OF IoStand...		☐	Private	
3	LAN1_EIPSlave_STATUS	Array[1..16] OF EtherNe...		☐	Private	
4	J_ModbusSlave_TCP	IoStandard._SYS_SlaveS...		☑	Private	
5	bDisableEnable	BOOL		☑	Private	从站禁用变量

    Devices that support automatic variable creation include:
    - Remote slave devices added when the current PLC acts as a Modbus TCP Master.
    - Remote adapter devices added when the current PLC acts as an EtherNet/IP scanner.

After disabling, it is necessary to perform a 10.8.3 full download of the PLC to make the disabling operation effective.

- Disabling result
  The disabled devices are marked with the icon for differentiation.

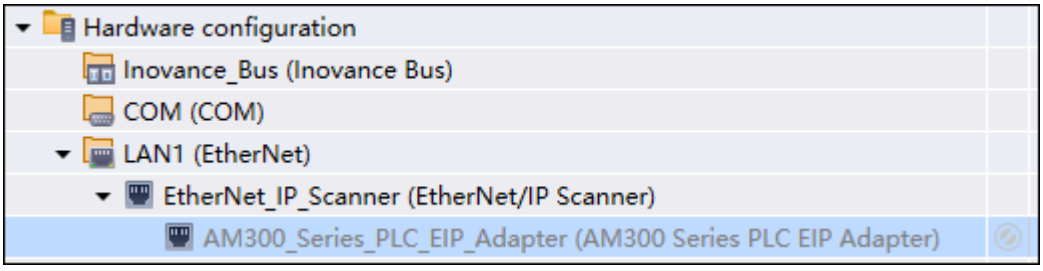


Figure 10-6-4-4-1 Icon in project tree

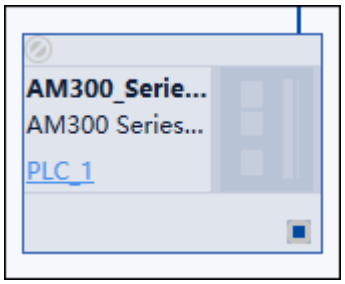


Figure 10-6-4-4-2 Icon in network configuration view

Enabling devices under the LAN1 (EtherNet) node

The disabling and enabling of LAN1 (EtherNet) devices are a set of mutually corresponding operations. Only when the device is disabled is the enabling command visible.

The enabling operation and its effect are exactly the same as the disabling operation.

A device needs to be disabled and enabled in the same way. That is, if the device is disabled by setting a variable, it must be enabled by resetting the variable. If it is disabled using a command, it must also be enabled using a command.

10.6.5 PROFINET Communication Configuration (EVO400, EVO500, EVO700, APC500, EVO800)\

10.6.5.1 Overview

PROFINET is introduced by PROFIBUS International (PI) and is a new generation of automation bus standards based on industrial Ethernet technology. PROFINET provides a complete network solution for the field of automation communication, covering hot topics such as real-time Ethernet, motion control, distributed automation, functional safety, and network security. As a cross-vendor technology, it is fully compatible with industrial Ethernet and existing fieldbus technologies (such as PROFIBUS), protecting existing investments.

PROFINET communication is abbreviated as PN communication. The following section will explain it in the following two modes:

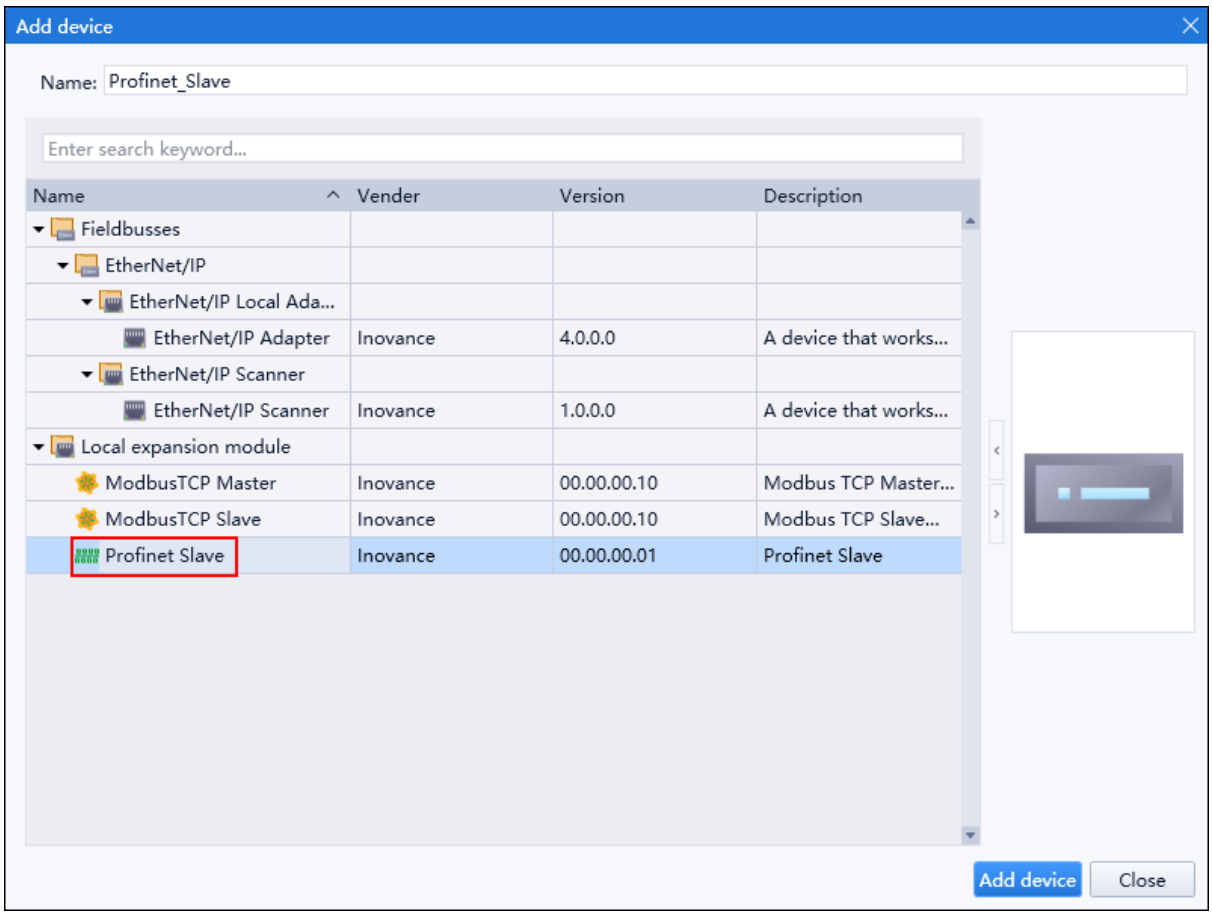
- PN communication between EVO500 series PLCs (as PROFINET slaves) and Siemens PLC masters
- PN communication between PN expansion modules of EVO800 series PLCs (as master) and GL20 series I/O devices

10.6.5.2 PN Slave Configuration(EVO400,EVO500,EVO700,APC500)

10.6.5.2.1 PN Slave Configuration

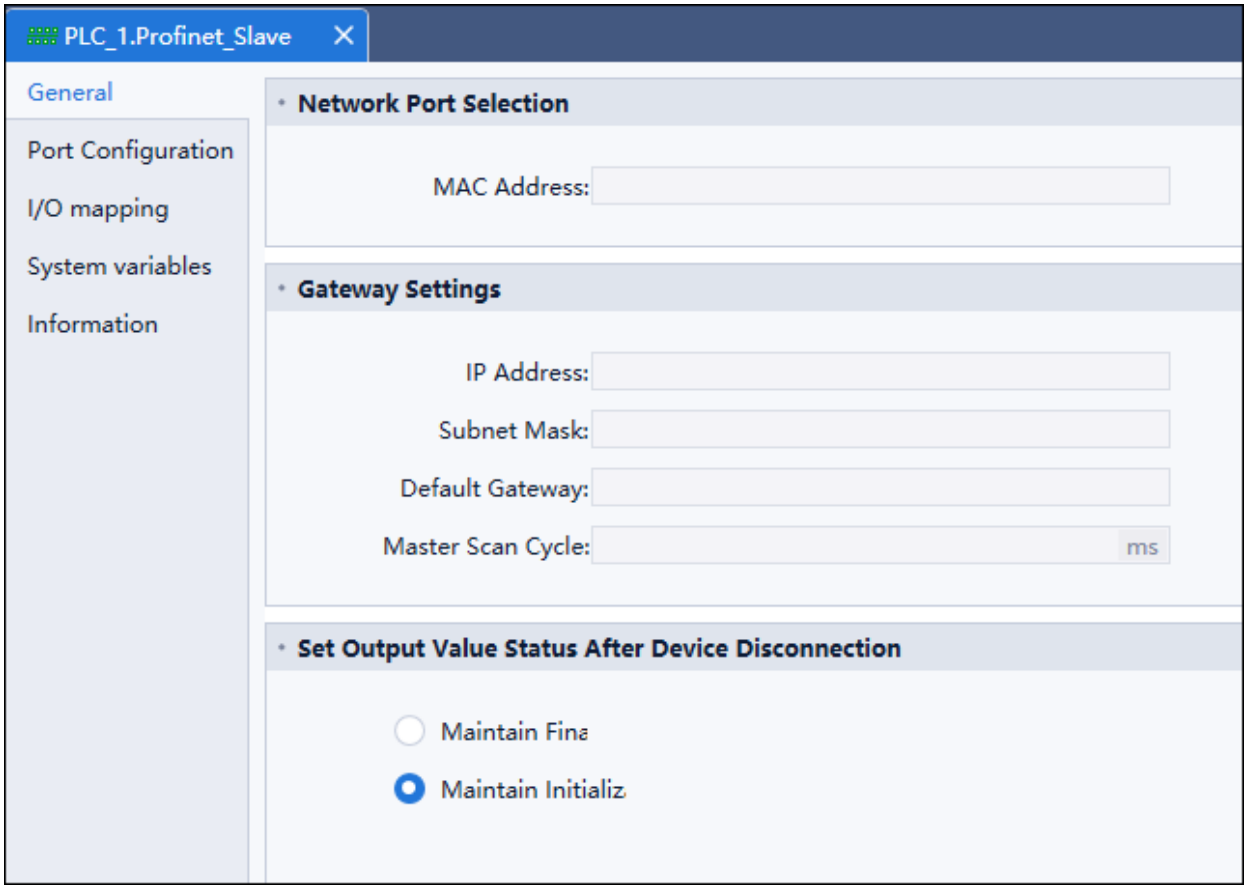
Procedure

1. Select the device for connection.
   Add a device supporting the PN protocol in the project tree. The EVO500 series (firmware version V1.3) is used as an example. PN communication protocol is not supported when an independent IP address is configured for each EtherNet port. The default mode is "Switch". For details, see 10.6.14.2 Network Settings (EVO400, EVO500).
2. Configure and add the PN slave device.



3. Configure parameters.
   Double-click the PN slave device under the "Hardware configuration" node in the left project tree to enter the parameter settings interface.

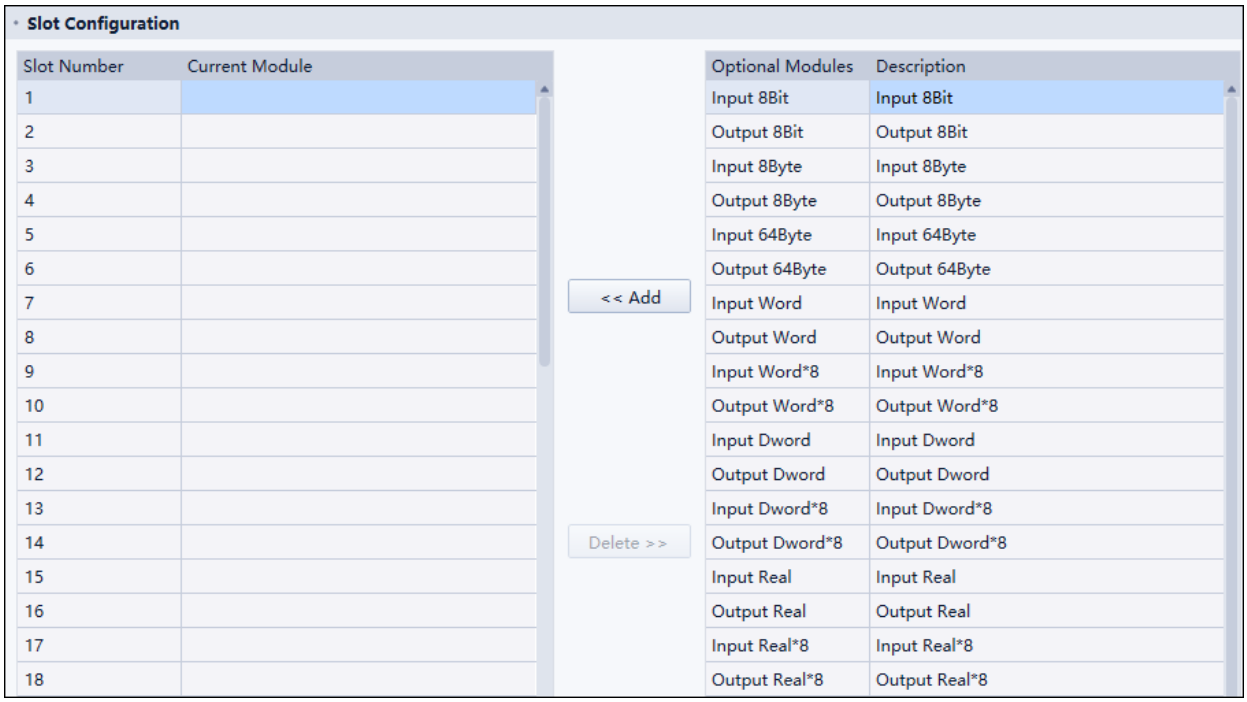
General configuration



Option		Function	Note
Network port selection		Displays information about the LAN port.	"MAC address" displays the actual MAC address of the local network port and cannot be modified. For the EVO500 series PLC, only the MAC address of the LAN1 port is displayed.
Gateway setting	IP address	Displays the IP information assigned to the slave by the master.	Automatically scans the gateway information of the device. This cannot be modified.
	Subnet mask		
	Default gateway		
	Master scan cycle	Displays the master scan cycle for communication.	• Displays 0 when no communication is taking place. • Displays real-time data when communication is active. The recommended master scan cycle for PN task is 4 ms. Values below 4 ms may cause inconsistent data refresh.

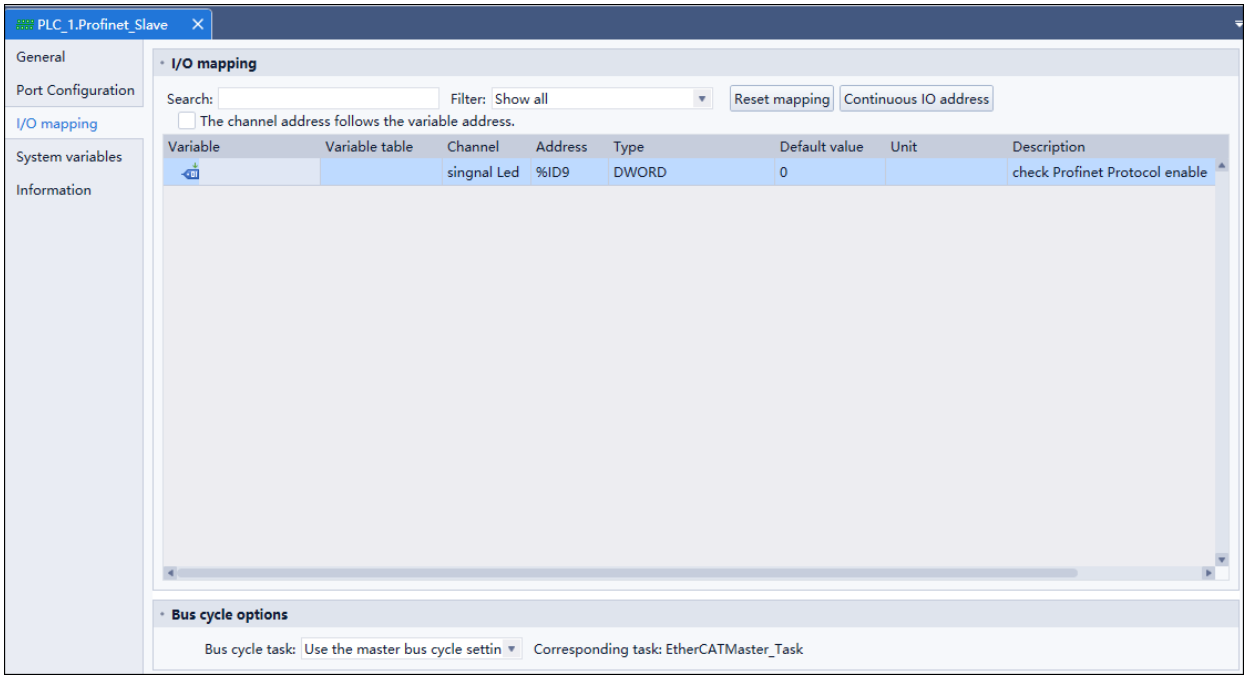
Option	Function	Note
Set the output value status upon device disconnection	Used to set the output value status when the device is disconnected.	• Maintain final state: When disconnected, output data remains at the last output value. • Maintain initialization state: When disconnected, all output data is cleared to zero. Note: When the master is in the STOP state, it is not considered a communication failure, and I/O data exchange proceeds normally.

Port configuration



- Select a blank slot on the left, select the I/O module type on the right, and click "Add" or double-click the selected module to add it to the corresponding slot. Once a module is added, it does not support overwriting addition.
- Select a device on the slot and click "Delete" or use the shortcut key "Delete" to remove the module.

I/O mapping



- I/O mapping
  Includes slave I/O signals and module I/O mapping. When an I/O module is added in port configuration, the corresponding module appears in the I/O mapping table. If no module is added, only slave I/O mapping is displayed by default. For detailed interface parameters, see 10.6.3.3.9 I/O Mapping.
- Bus cycle options
  Bus cycle task: Displays all task names. Default is "Inherit from bus cycle setting", corresponding to MainTask.

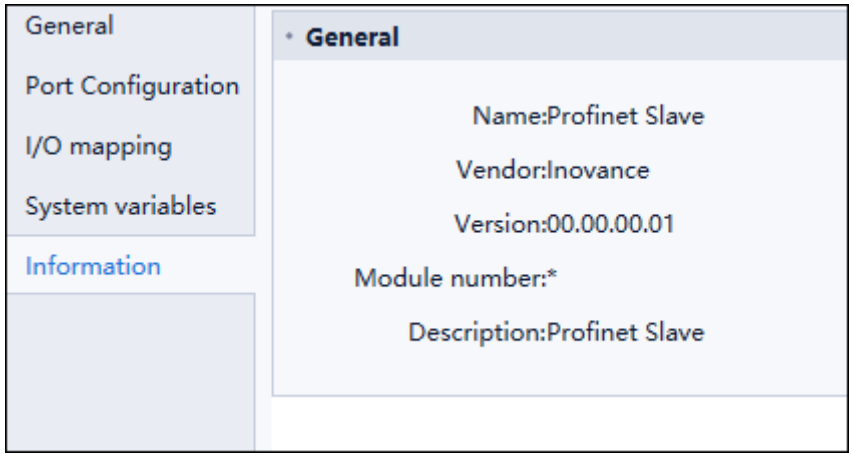
NOTE

Since CPU utilization affects the real-time performance of PN communication, it is recommended to set the watchdog time of the PN master as follows:

- When CPU utilization is below 50%, it is recommended to set the watchdog time to 20 ms.
- When CPU utilization is between 50% and 80%, it is recommended to set the watchdog time to 28 ms.
- When CPU utilization is above 80%, it is recommended to set the watchdog time to 36 ms.

Information

Displays basic device information.



10.6.5.2.2 PROFINET I/O Module

The PN slave communicates with the PN master through I/O modules, which can be divided into three major categories based on the direction of data transmission within the module: Input, Output, and InOut.

- The Input type is a module that transmits data from the slave to the PN master, and the transmitted data is PdIndata.
- The Output type is a module that transmits data from the master to the slave, and the transmitted data is PdOutdata.
- The InOut type is a module where data can be transmitted between the master and slaves, and the transmitted data includes both PdIndata and PdOutdata.

In PN communication, I/O modules must be configured in the master and slaves with corresponding slot numbers and data types for communication. Unmatching configuration of the I/O modules can result in data transmission failure, but this does not affect the functionality of other I/O modules. Up to 64 I/O modules can be configured for a PN slave, and the maximum PdIndata and PdOutdata data length that can be transmitted in a single frame is 1024 bytes. The specific types of I/O modules and their corresponding relationships are as follows:

Table10-6-5-2-2-1 Input type I/O modules

I/O Module for PN Slave	I/O Module for PN Master	Description
Input 64Byte	Input 64Byte	Inputs 64 bytes of data to the master.
Input 8bit	Input 8bit	Inputs 1 byte of data to the master.
Input 8Byte	Input 8Byte	Inputs 8 bytes of data to the master.
Input Dword	Input Dword	Inputs 4 bytes of data to the master.
Input Dword *8	Input Dword *8	Inputs 32 bytes of data to the master.
Input Real	Input Real	Inputs 4 bytes of data to the master.
Input Real *8	Input Real *8	Inputs 32 bytes of data to the master.
Input Word	Input Word	Inputs 2 bytes of data to the master.
Input Word *8	Input Word *8	Inputs 16 bytes of data to the master.

Table10-6-5-2-2-2 Output type I/O modules

I/O Module for PN Slave	I/O Module for PN Master	Description
Output 64Byte	Output 64Byte	Outputs 64 bytes of data from the master.
Output 8bit	Output 8bit	Outputs 1 byte of data from the master.
Output 8Byte	Output 8Byte	Outputs 8 bytes of data from the master.
Output Dword	Output Dword	Outputs 4 bytes of data from the master.
Output Dword *8	Output Dword *8	Outputs 32 bytes of data from the master.

I/O Module for PN Slave	I/O Module for PN Master	Description
Output Real	Output Real	Outputs 4 bytes of data from the master.
Output Real *8	Output Real *8	Outputs 32 bytes of data from the master.
Output Word	Output Word	Outputs 2 bytes of data from the master.
Output Word *8	Output Word *8	Outputs 16 bytes of data from the master.

Table10-6-5-2-2-3 InOut type I/O modules

I/O Module for PN Slave	I/O Module for PN Master	Description
InOut 64Byte	InOut 64Byte	Exchanges 64 bytes of data between master and slave.
InOut 8bit	InOut 8bit	Exchanges 1 byte of data between master and slave.
InOut 8Byte	InOut 8Byte	Exchanges 8 bytes of data between master and slave.
InOut Dword	InOut Dword	Exchanges 4 bytes of data between master and slave.
InOut Dword *8	InOut Dword *8	Exchanges 32 bytes of data between master and slave.
InOut Real	InOut Real	Exchanges 4 bytes of data between master and slave.
InOut Real *8	InOut Real *8	Exchanges 32 bytes of data between master and slave.
InOut Word	InOut Word	Exchanges 2 bytes of data between master and slave.
InOut Word *8	InOut Word *8	Exchanges 16 bytes of data between master and slave.

10.6.5.2.3 TIA Portal and PN Master Configuration

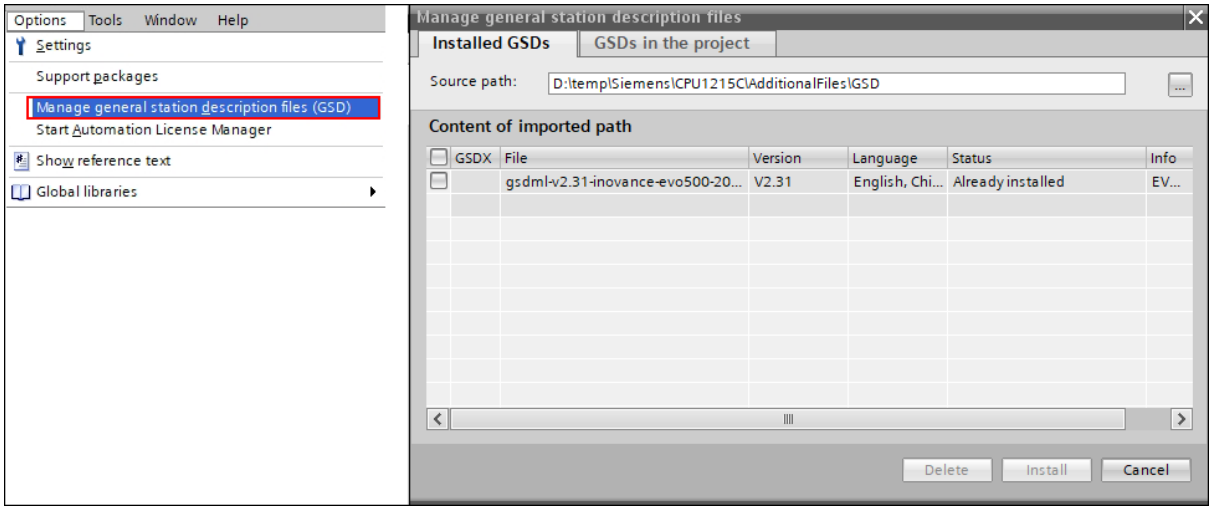
This section introduces the PN slave configuration using a Siemens device as the master and the EVO500 series PLC as an example.

Prerequisites

The GSD file for the EVO500 series device has been obtained.

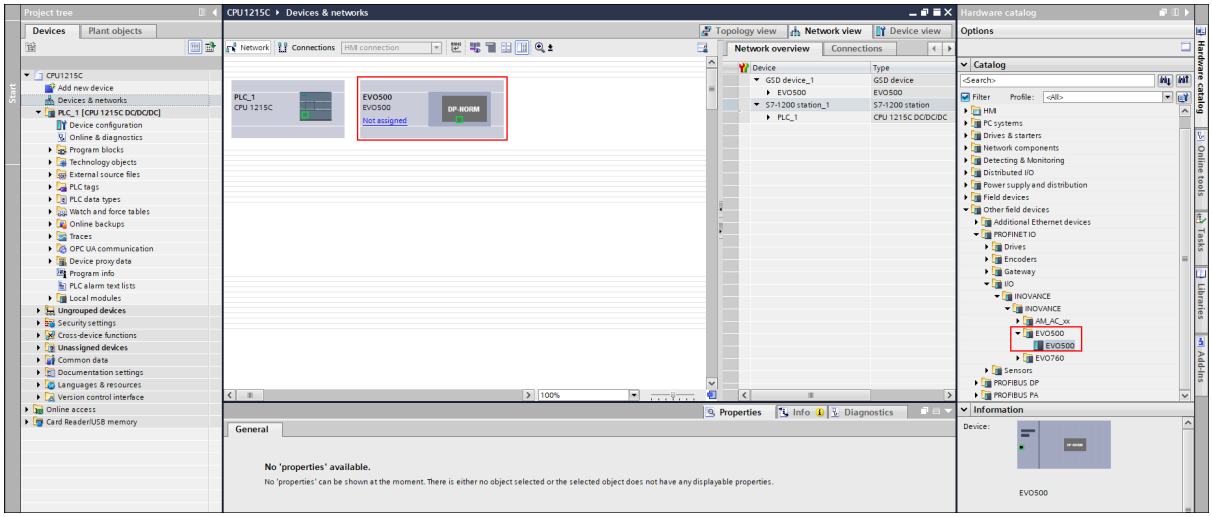
Procedure

1. Import and install the GSD file for the PN device.
   In TIA Portal, select "Options > Manage general station description files (GSD) (D)", select the path where the GSD file is located, and install the required PN GSD file from the automatically retrieved list.

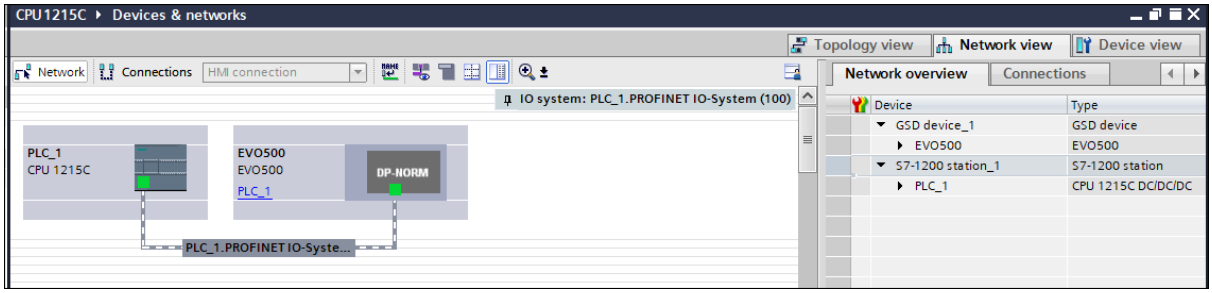


2. Configure and add the PN slave device.

   Click "Devices and networks", then double-click or drag the "EVO500" from the "Hardware catalog" into the network view.



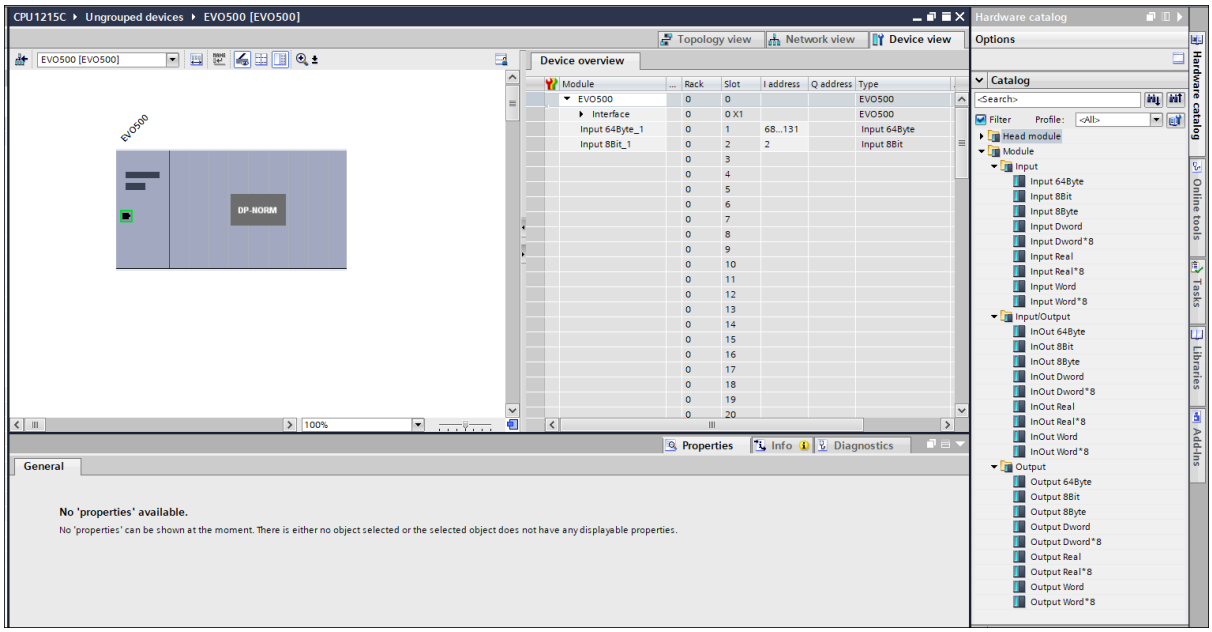
   In the network view, click the green box of the PLC device and drag the connection to the green box of the EVO500 device.



> **NOTE**
>
> When the EVO500 device is configured as a PROFINET slave, the TIA Portal software automatically assigns an IP address to the EVO500 device via the PROFINET bus. The assigned IP address must not be the same as that of the EVO500 device itself, otherwise the actual IP address of the EVO500 device will become invalid.

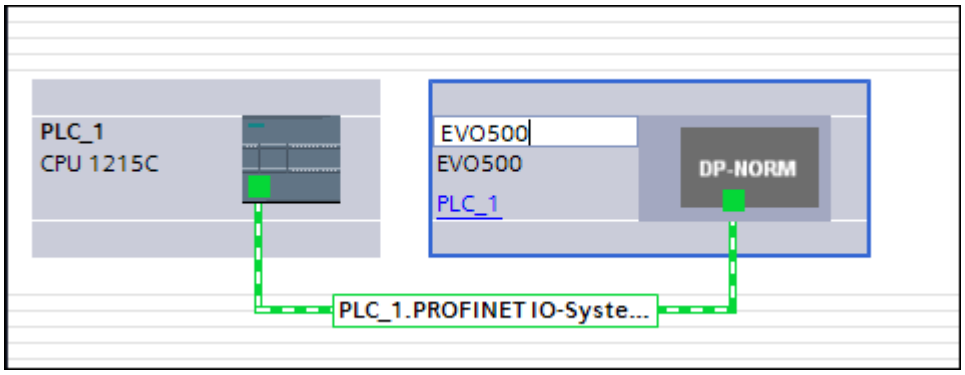
3. Configure the I/O modules.

Click "Device view", then double-click or drag the required I/O modules from the "Hardware catalog" into the "Device overview".



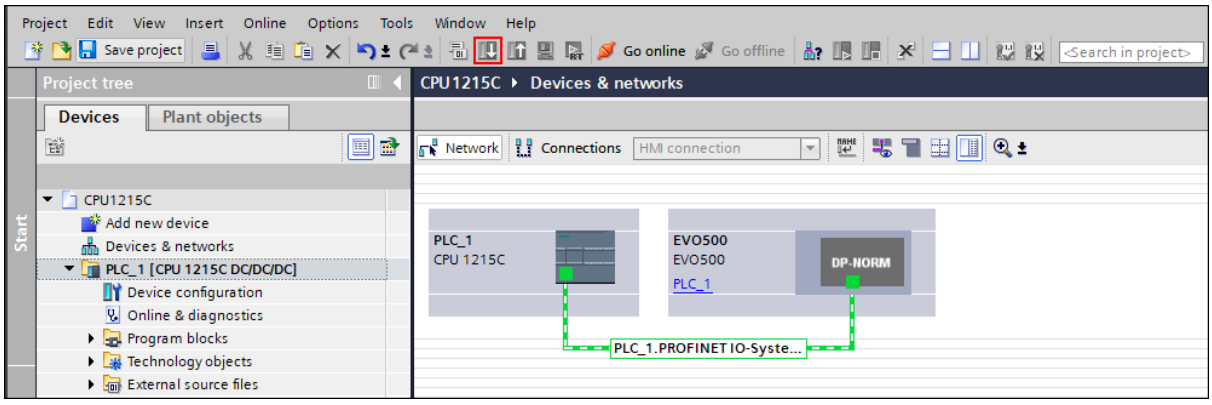
4. Rename the device.

   Double-click the text name to rename the device.



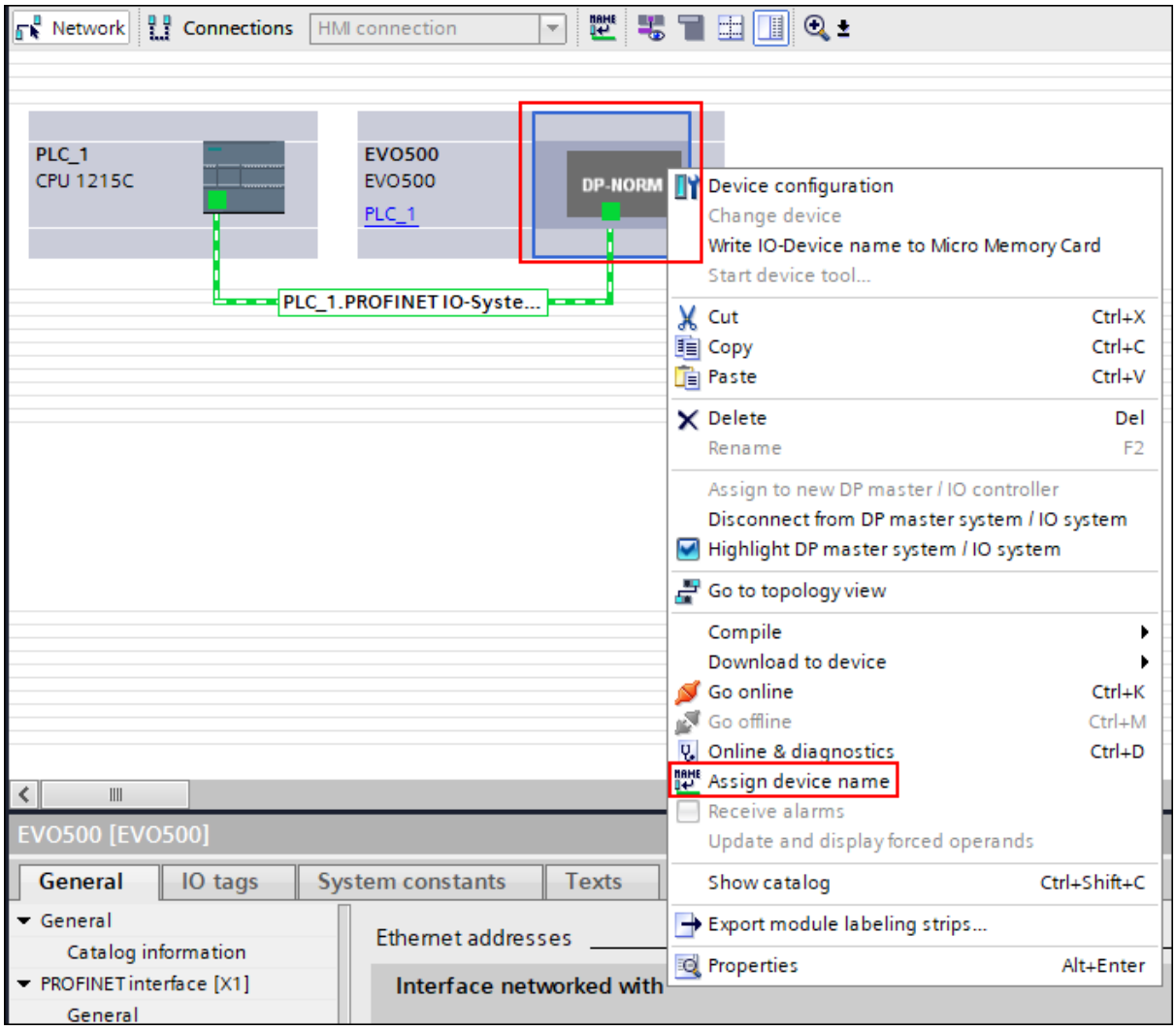
5. Download the configured project to the master PLC.

   Compile and download the configured project to the master PLC.

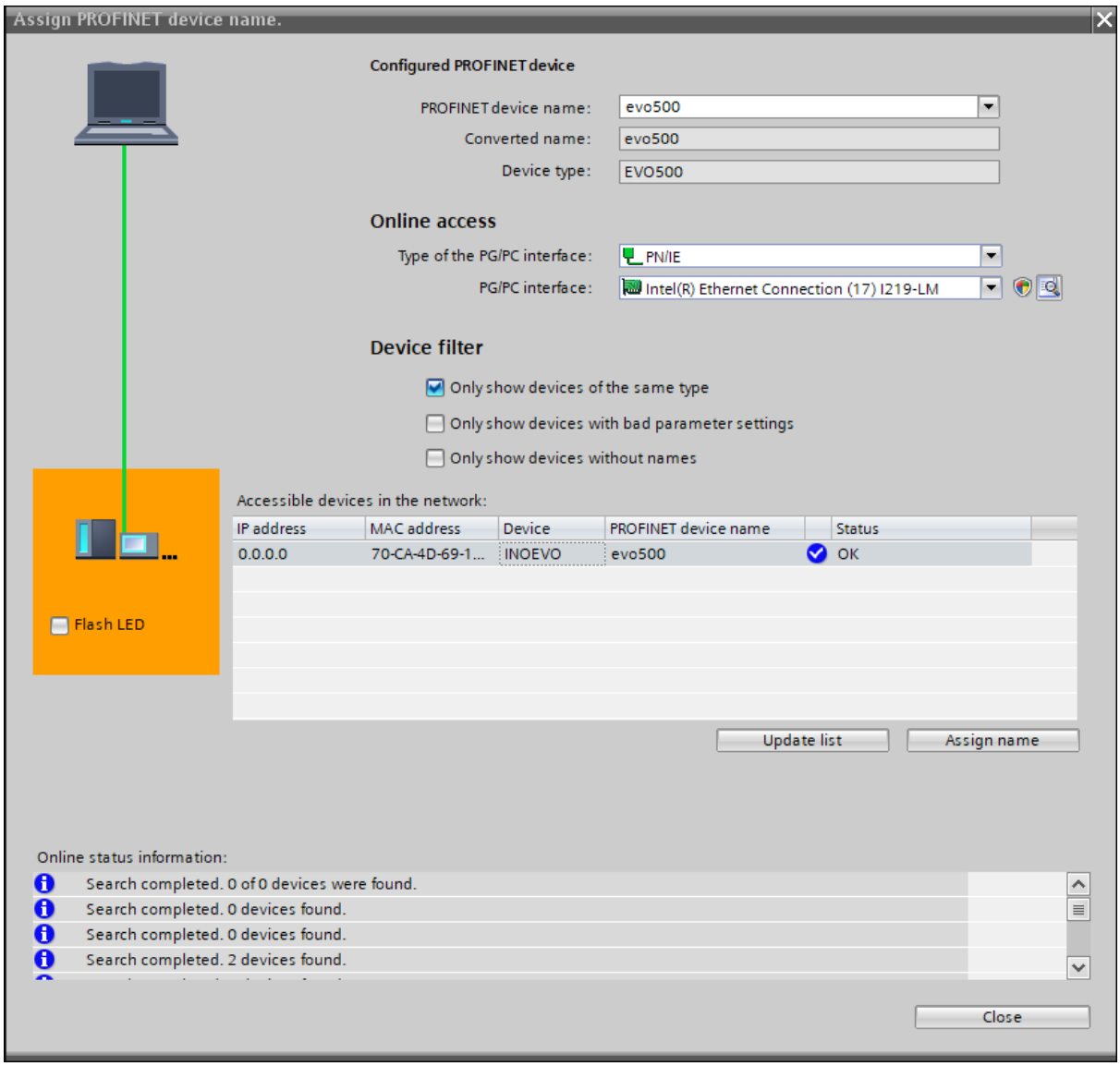


6. Assign a PROFINET device name to the actual PN slave device.

Right-click the EVO500 device in the configuration, and select "Assign device name" to assign the name.



In the "Assign PROFINET device name" dialog box, click "Update list", select the corresponding EVO500 device from the list, then click "Assign name" to complete the configuration on the PN master (TIA Portal).



NOTE

The IP address of the PROFINET slave device must not be the same as that of the device itself.

10.6.5.2.4 PN Slave Troubleshooting

The PN slave troubleshooting include fault information for the PN slave and its I/O submodules.

Fault Code	Diagnostic Information	Definition	Solution
0x00	-	No fault	-

Fault Code	Diagnostic Information	Definition	Solution
0x70	MODULE_ID_ERROR	The I/O module types between the master and slaves do not match.	1. Check whether the I/O module types between the master and slaves are compatible. 2. Check whether the corresponding I/O modules between the master and slaves exist.
0x71	MODULE_INPUT_LEN_MISMATCH	The input data slot length of the corresponding modules between the master and slaves does not match.	Overridden by fault 0x70, the solution is as follows: 1. Check whether the I/O module types between the master and slaves are compatible. 2. Check whether the corresponding I/O modules between the master and slaves exist.
0x72	MODULE_OUTPUT_LEN_MISMATCH	The output data slot length of the corresponding modules between the master and slaves does not match.	Overridden by fault 0x70, the solution is as follows: 1. Check whether the I/O module types between the master and slaves are compatible. 2. Check whether the corresponding I/O modules between the master and slaves exist.
0x80	MASTER_PLC_STOP	The PN master PLC is in STOP status.	Communication requires the PN master PLC to be in RUN status.
0x81	NETWORK_ERROR	The network between the PN master and slaves is not connected.	Check whether the network ports between the PN master and slaves are properly connected. 1. Ensure the slave is working properly. 2. Check whether the wiring is disconnected. 3. Optimize the wiring by eliminating potential interference sources.

10.6.5.3 PN Master Configuration (EVO800)

10.6.5.3.1 PN Master Configuration (SYCON.net)

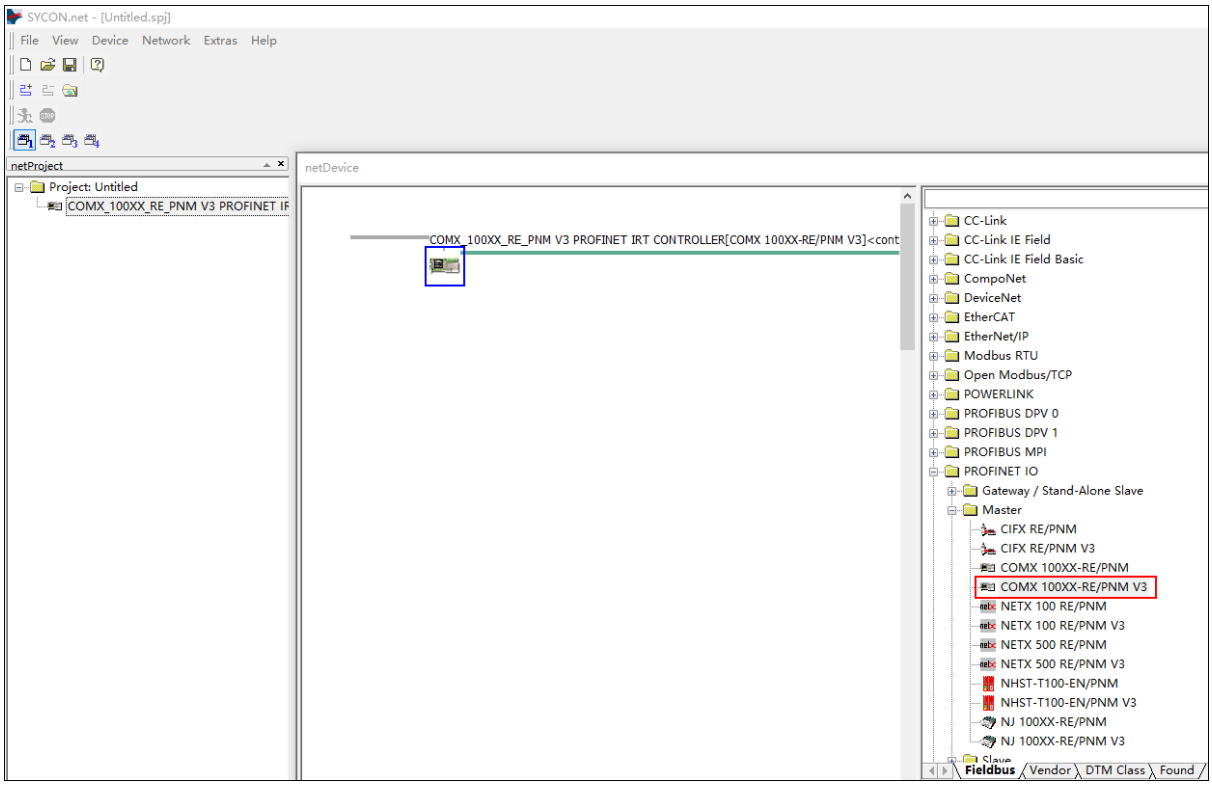
This section introduces how to configure the PN master and its slaves using the SYCON.net software.

Prerequisites

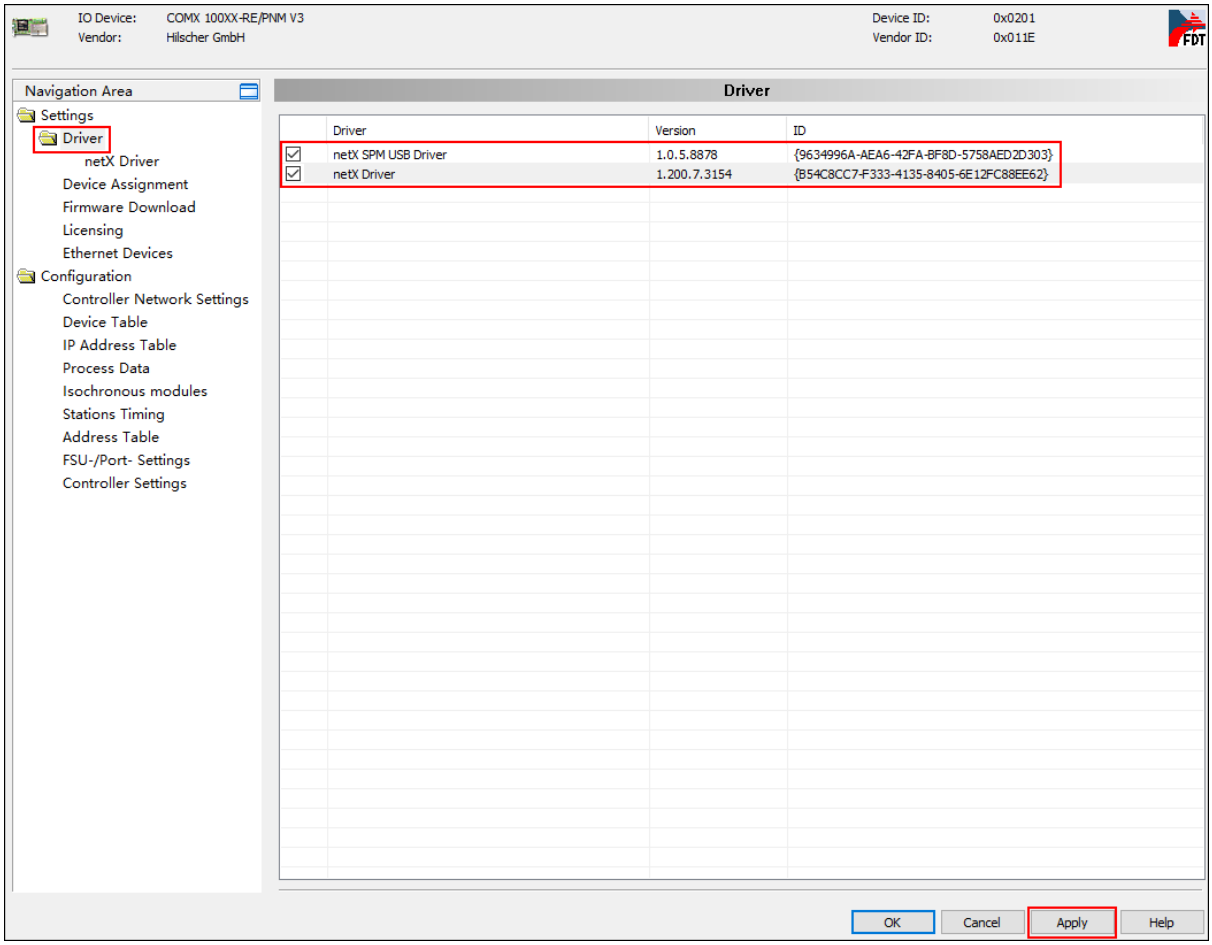
- SYCON.net software has been installed.
- The slave device description file has been obtained.
- Hardware connection has been completed.
  PN can connect to the PC via the commissioning interface. The default IP of the commissioning interface is 192.168.1.200 (ensure that the local network is configured in the 192.168.1.XX subnet).

PN master configuration

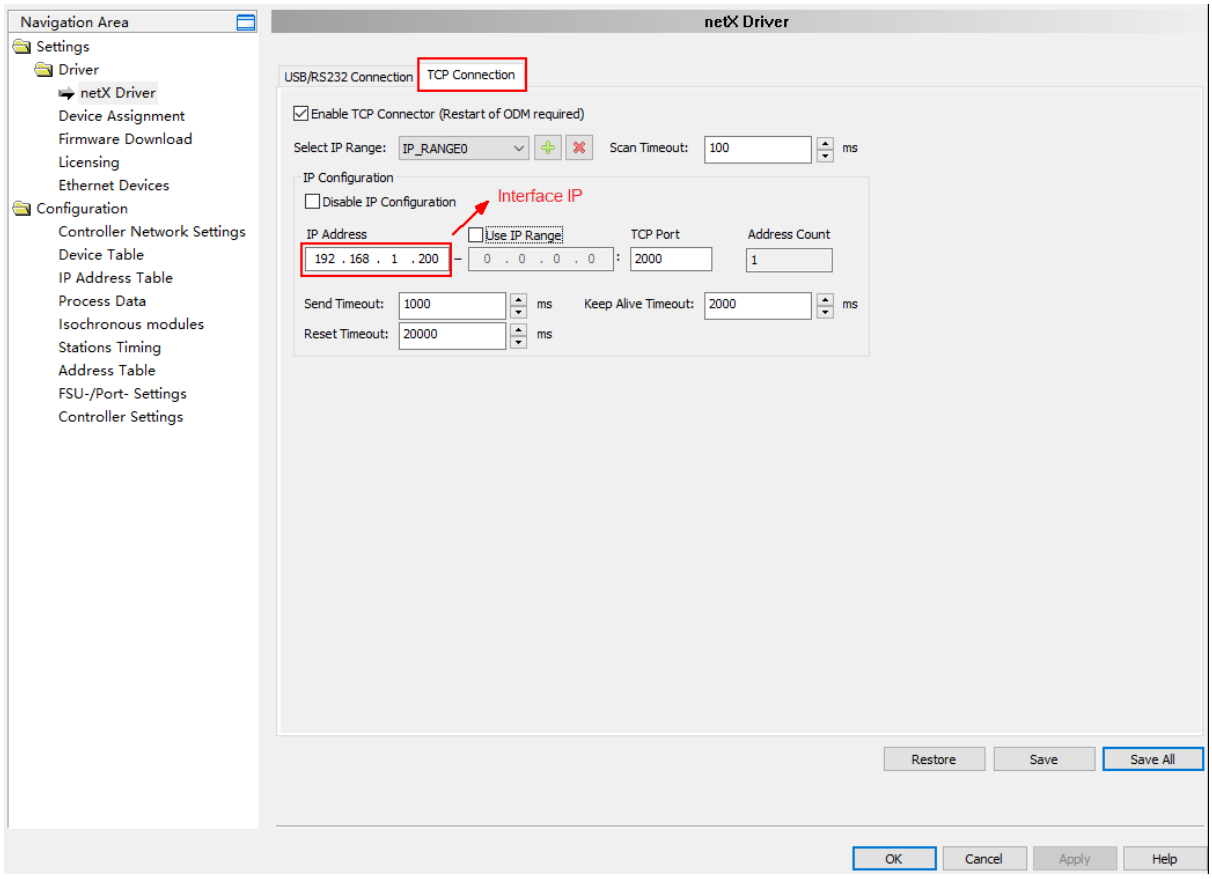
1. Launch the SYCON.net software as the admin.
2. Add a PN master. On the "Fieldbus" tab on the right of the interface, select "PROFINET IO > Master > COMX 100XX-RE/PNM V3" and drag it into the "netDevice" window, connecting it to the gray data line.



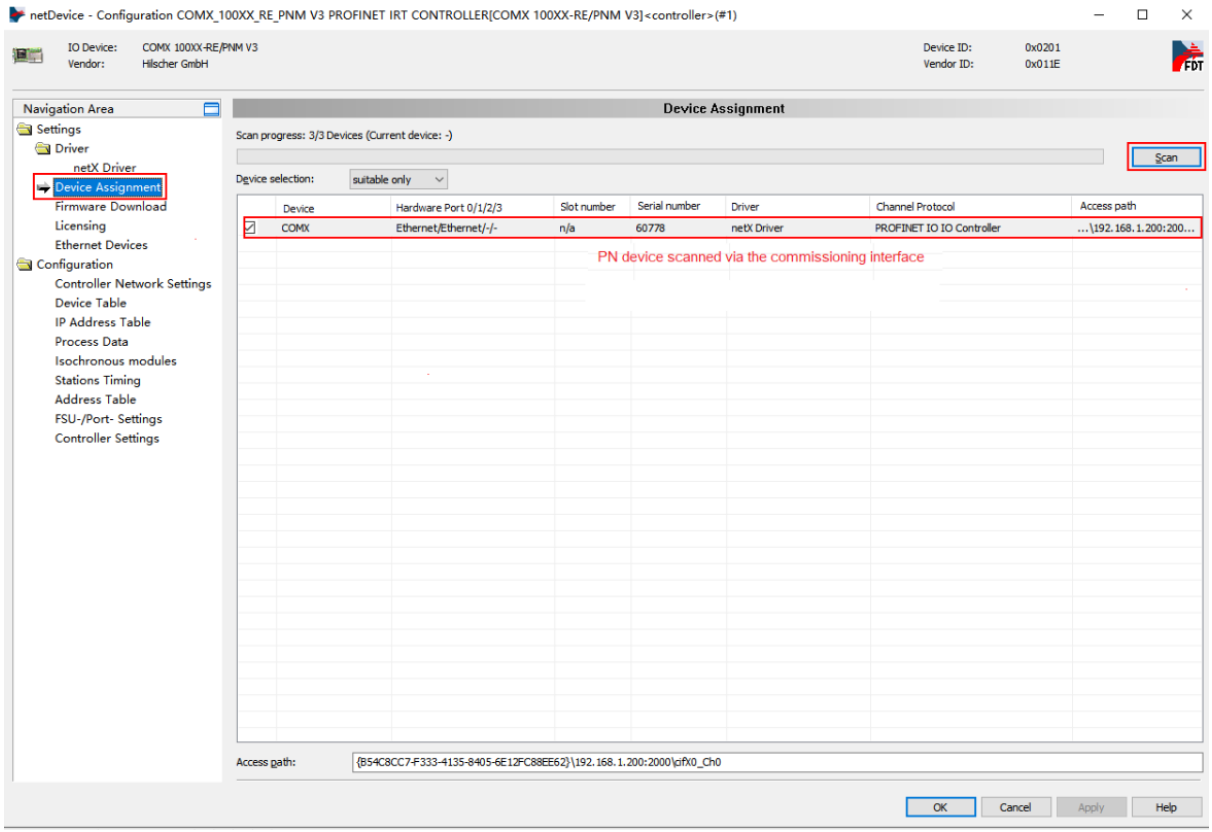
3. After adding the master, double-click the master icon to open the configuration page. On the "Driver" tab, check the configuration items as shown below, and click "Apply".



4. When connecting to the master via Ethernet, select "Driver > netXDriver" on the configuration page. On the "TCP Connection" tab, configure the following parameters: Click  in the "Select IP Range" area, set the IP range to "IP_RANGE0"; set "IP Address" to "192.168.1.200"; set "TCP Port" to "2000".



5. Select the "Device Assignment" tab, click "Scan". Once the PN module is found, click "Apply", then click "OK".

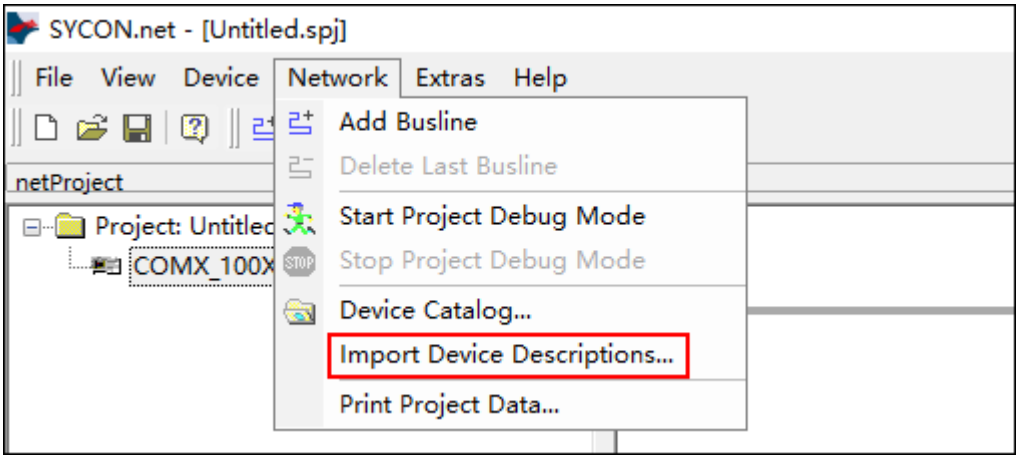


PN slave configuration

Taking the addition of a GL20-RTU-PN slave as an example.

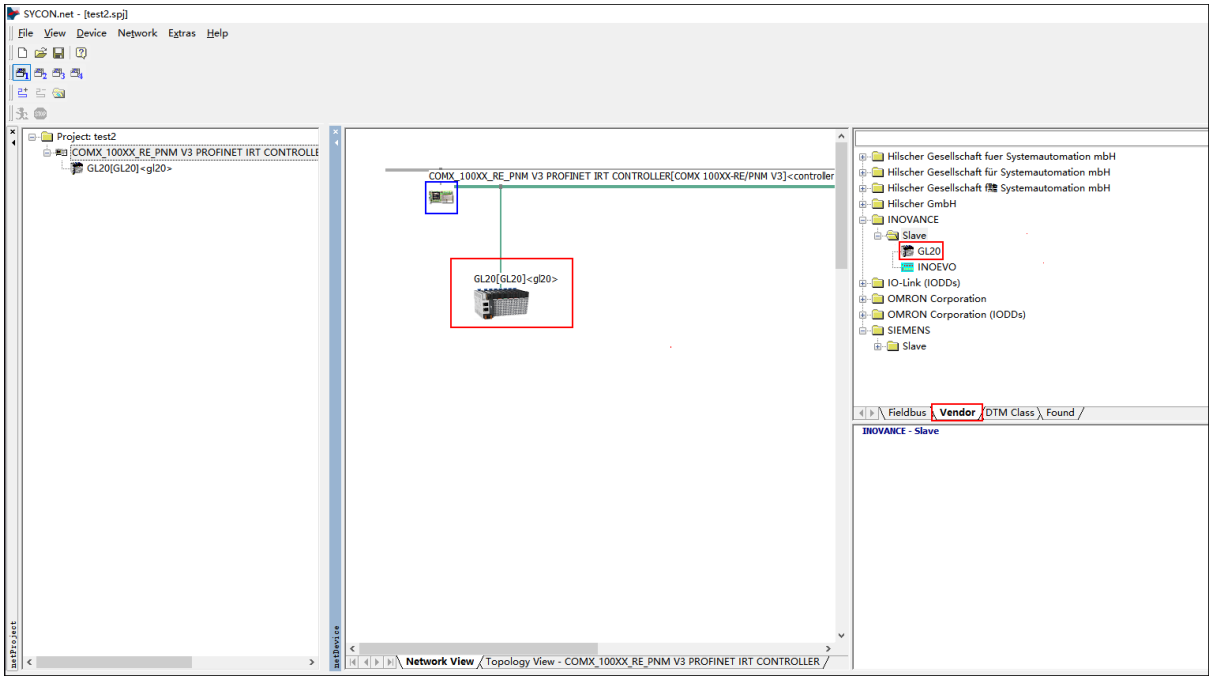
1. Import the slave device description file.

   In the menu bar, select "NetWork > Import Device Description" to import the slave device description file. You can view the added device on the "Vendor" tab on the right of the interface.



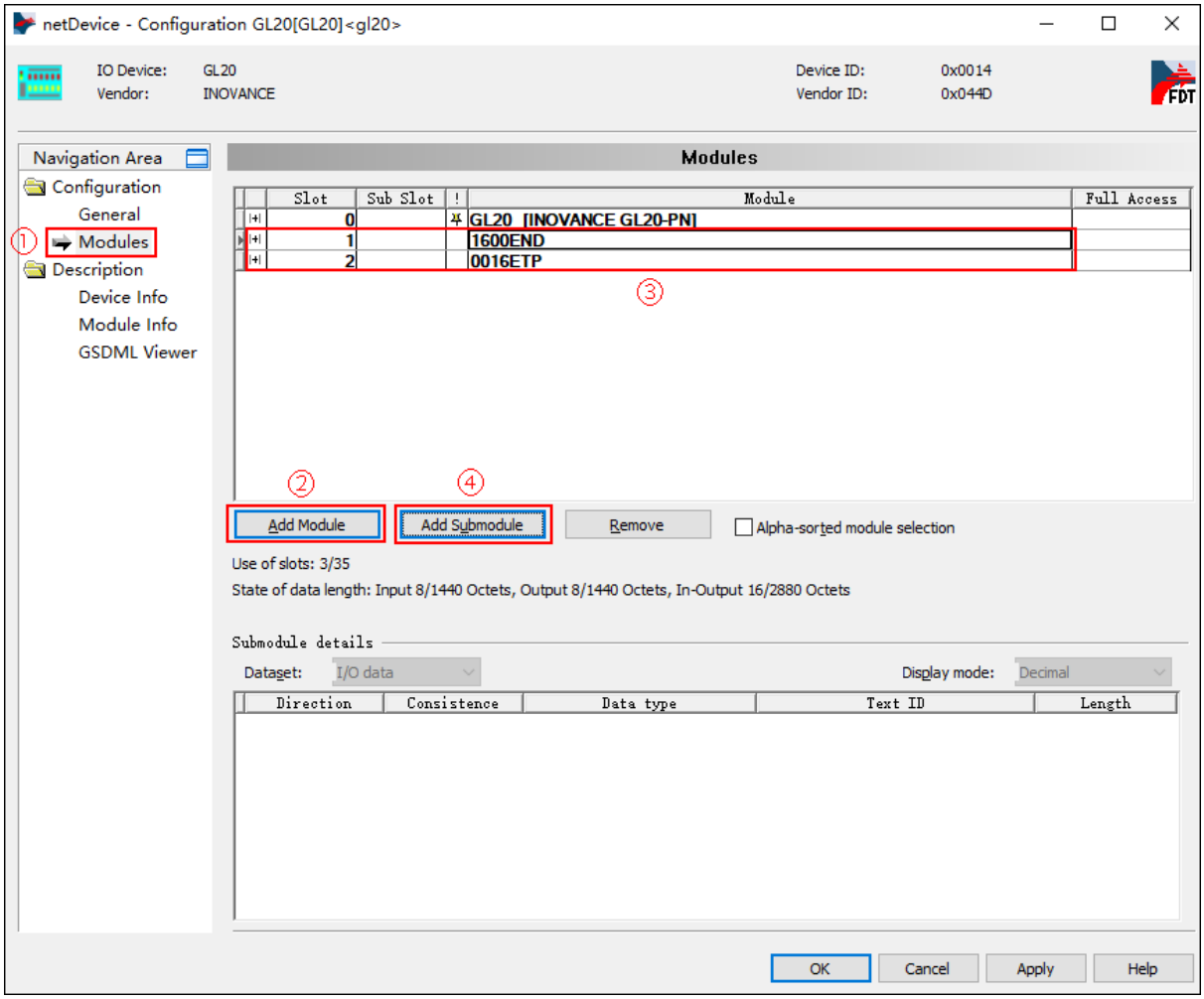
2. Add the GL20-RTU-PN slave.

   On the "Vendor" tab, select "INOVANCE > Slave > GL20", and drag it under the master.



3. Add submodules.

   Double-click the added GL20-RTU-PN and select the "Modules" tab. Click "Add Module", select the submodule type, and click "Add Submodule" to add the submodule.

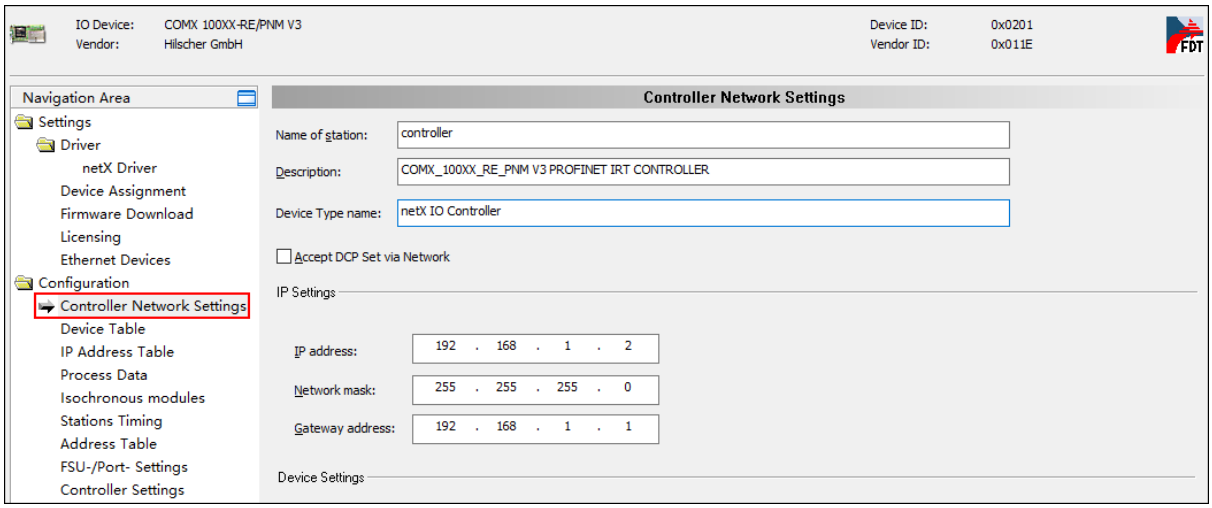


Master/Slave address configuration

1. Set the IP address for the master.

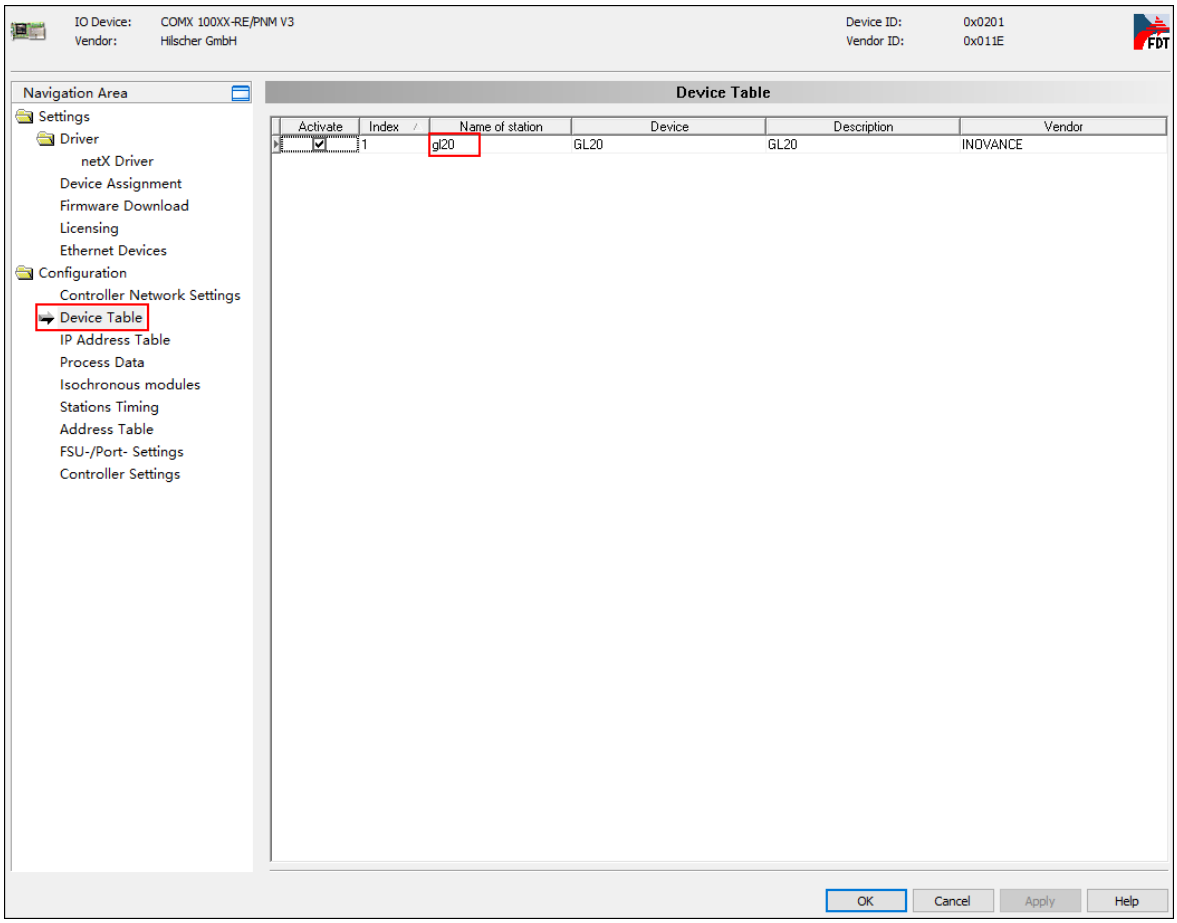
   Double-click the master icon to open the master configuration page. Select "Controller Network Settings"

to configure the parameters related to master IP address.

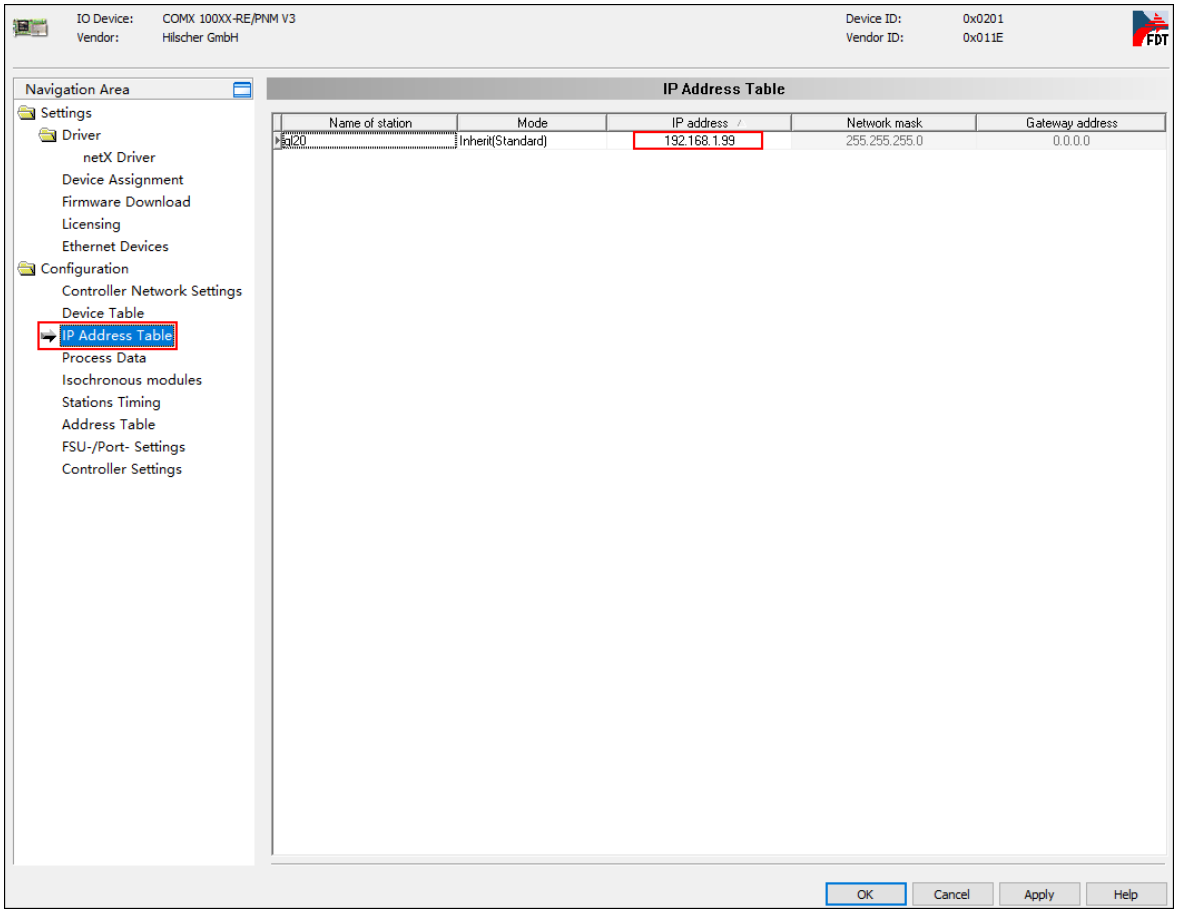


2. Scan and set the slave devices.

a. Select the "Device Table" tab, set the slave device names, and click "OK".



b. Select the "IP Address Table" tab, set the slave IP addresses, and click "OK".



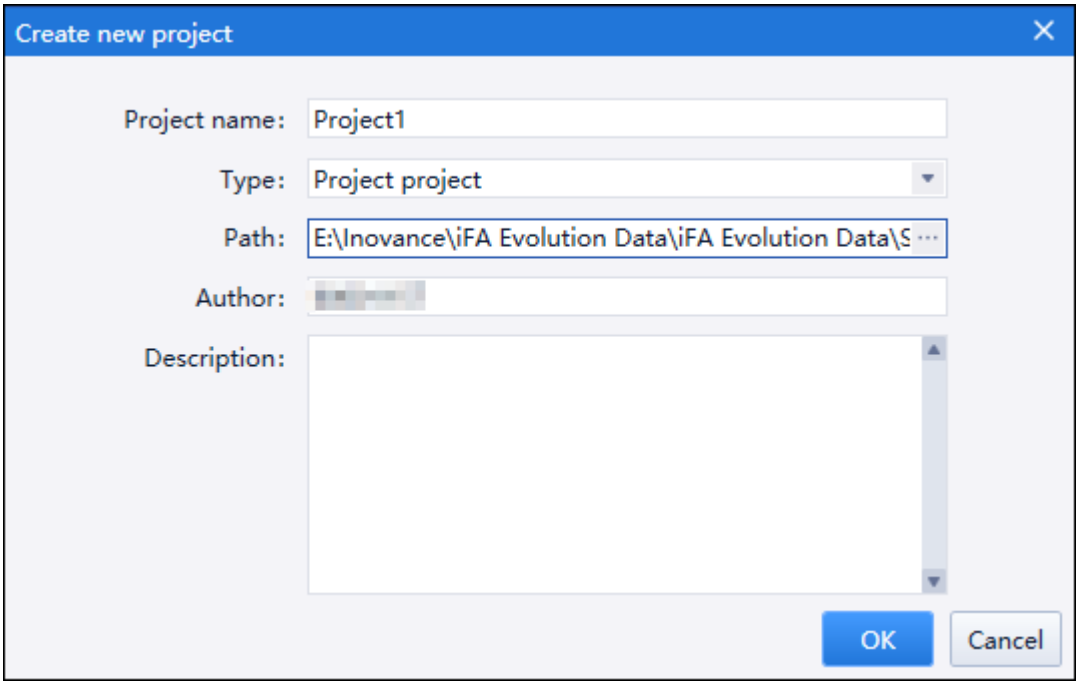
10.6.5.3.2 PN Master Configuration (iFA)

Prerequisites

- All products in the configuration example have been installed, wired, and powered on.
- The PC installed with iFA Evolution is connected to the LAN1 port of the EVO820 control module via a network cable.

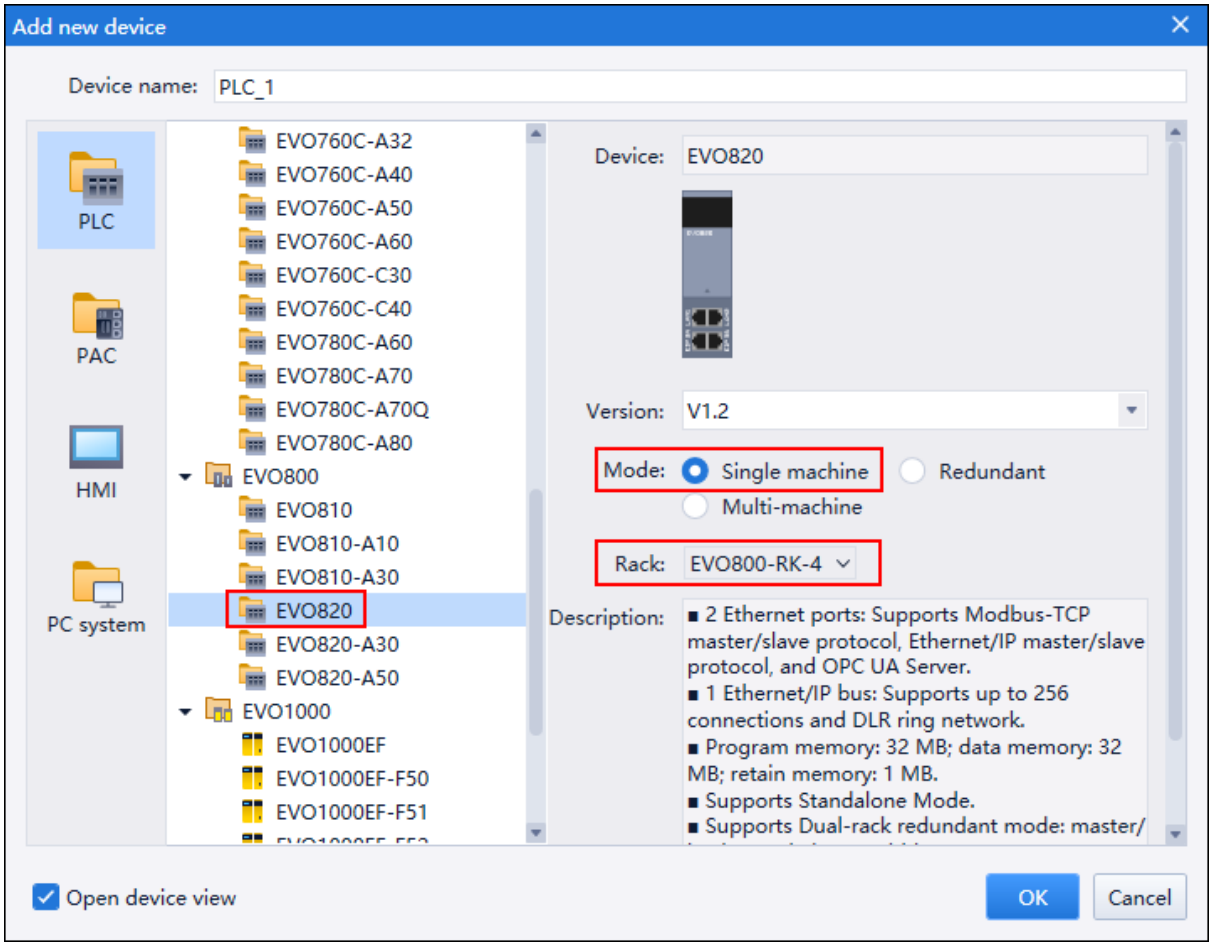
Procedure

1. Create a new project.
   In the menu bar, select "Project > New". Enter the project information (select "Engineering project" for "Type"), and click "OK".

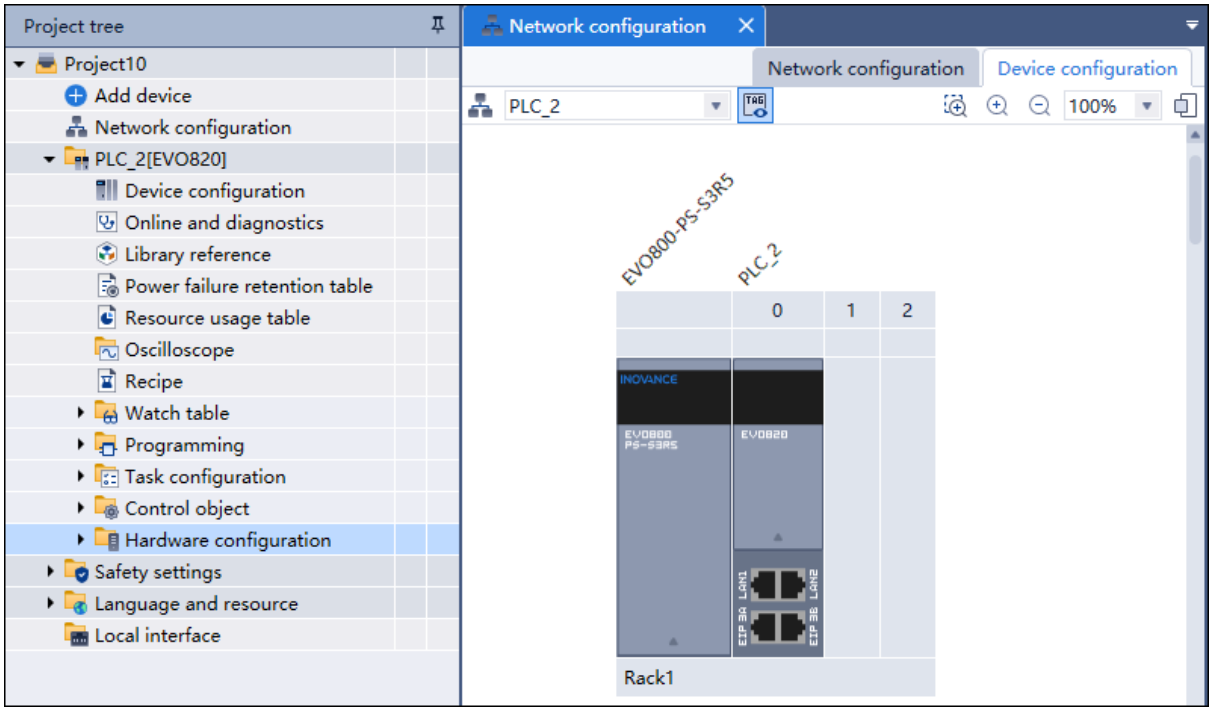


2. Add a new device.

   Double-click "Add device" in the project tree, click "EVO820" in the opened dialog box, enter the device name, "Mode" is set to "Single machine" by default, select the proper "Rack", and click "OK".

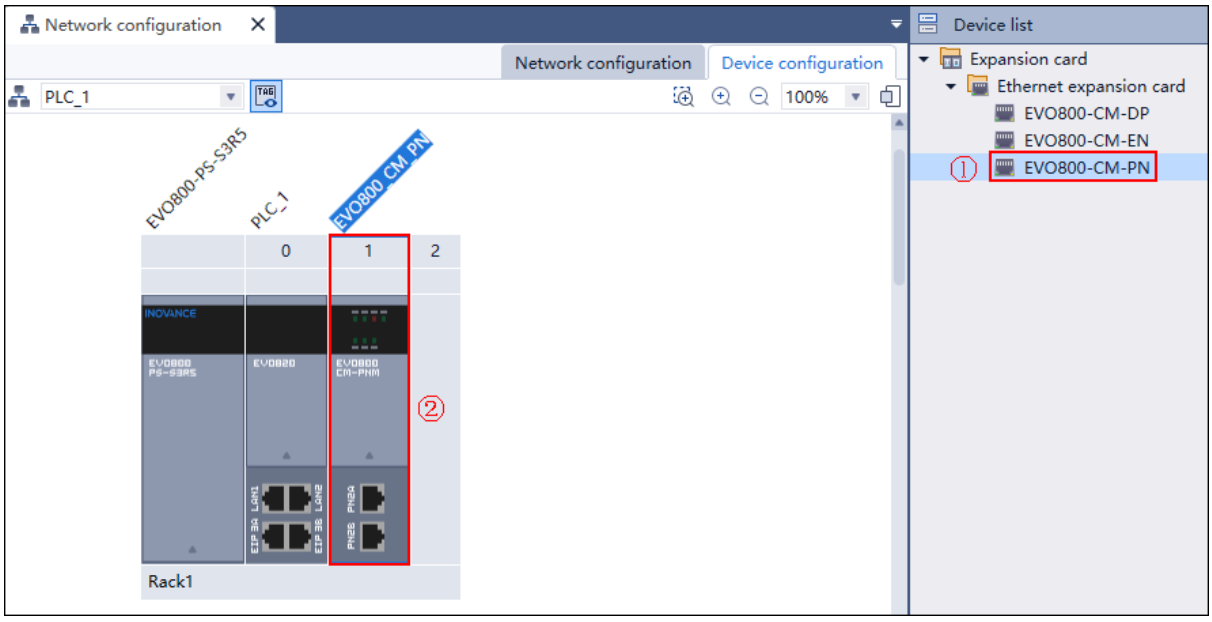


The newly added EVO820 device appears in the project tree.

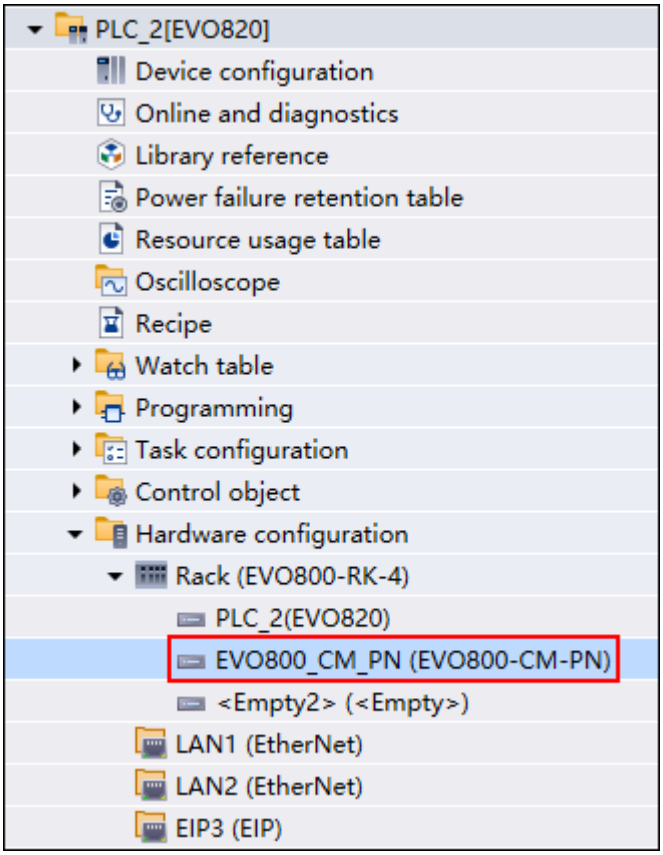


3. Add the EVO800-CM-PN module.

   In the project tree, double-click "Device configuration", then double-click "EVO800-CM-PN" in the "Device list" to add the PN master expansion module.



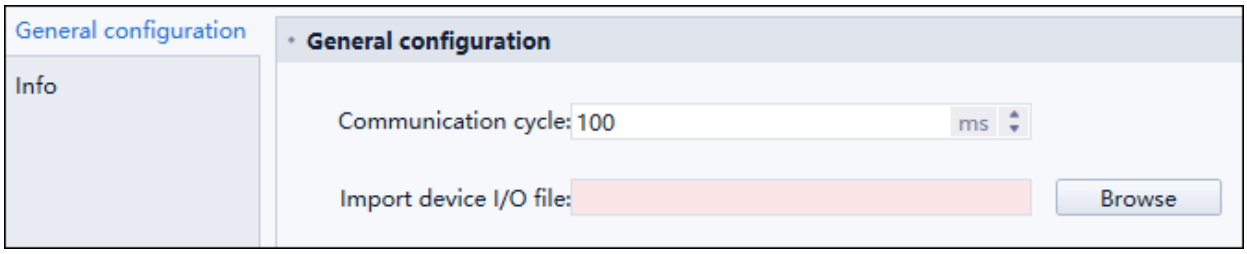
The newly added PN master expansion module appears in the project tree.



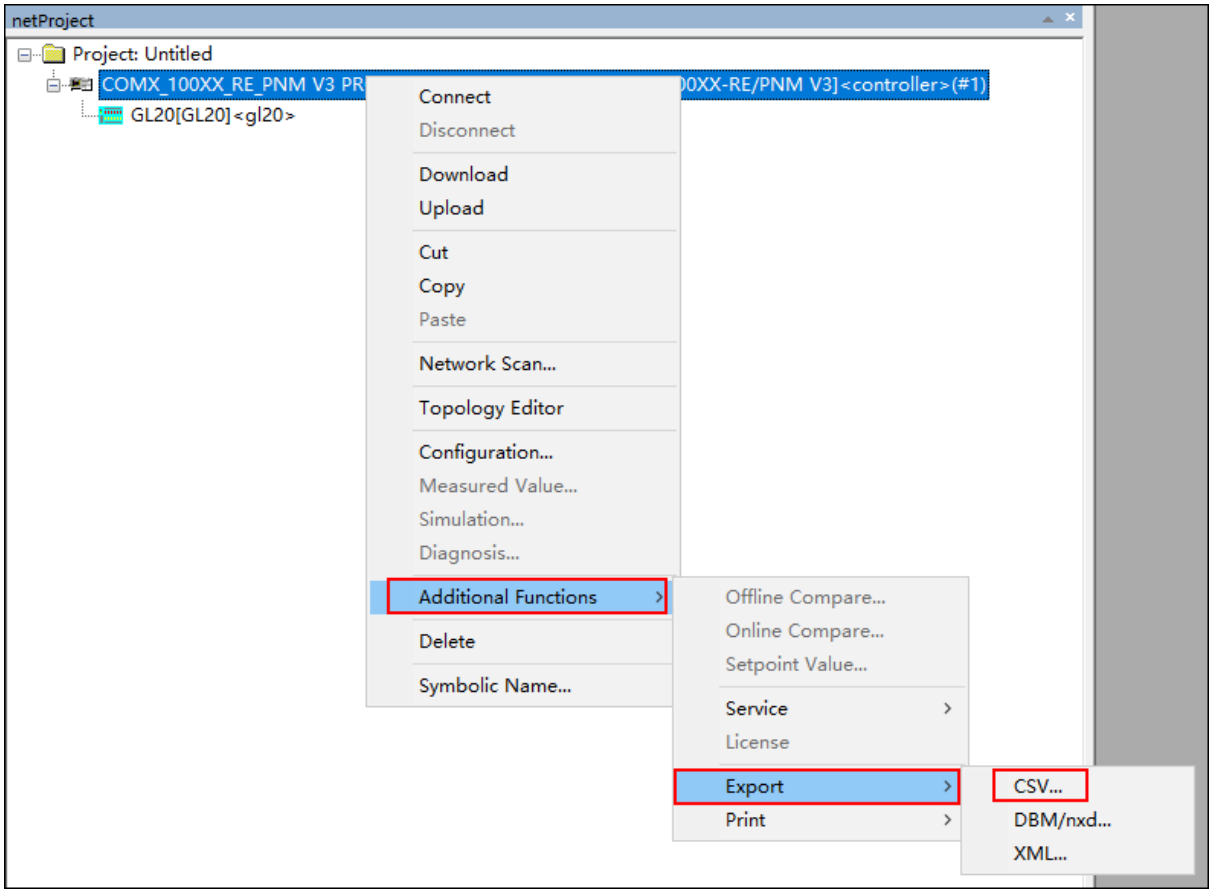
10.6.5.3.3 PN Master Configuration

10.6.5.3.3.1 General Configuration

Double-click the PN master node to enter the master parameter configuration interface.

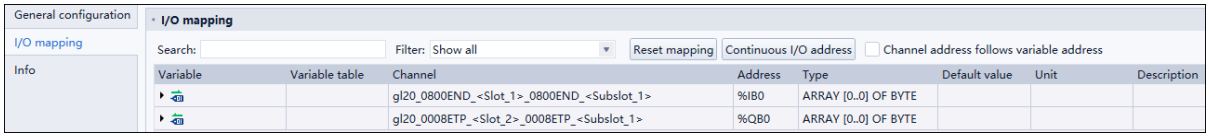


- Communication cycle: The backplate communication cycle between the PLC and the PN module. Default: 100 ms.
- Import device I/O file: Click "Browse" to select a CSV file (exported from SYCON.net software) for import.
  To export the I/O configuration file of the PN module from SYCON.net:
  After configuring the master, right-click the master, and select "Additional functions > Export > CSV" to save the file in CSV format.



10.6.5.3.3.2 I/O Mapping

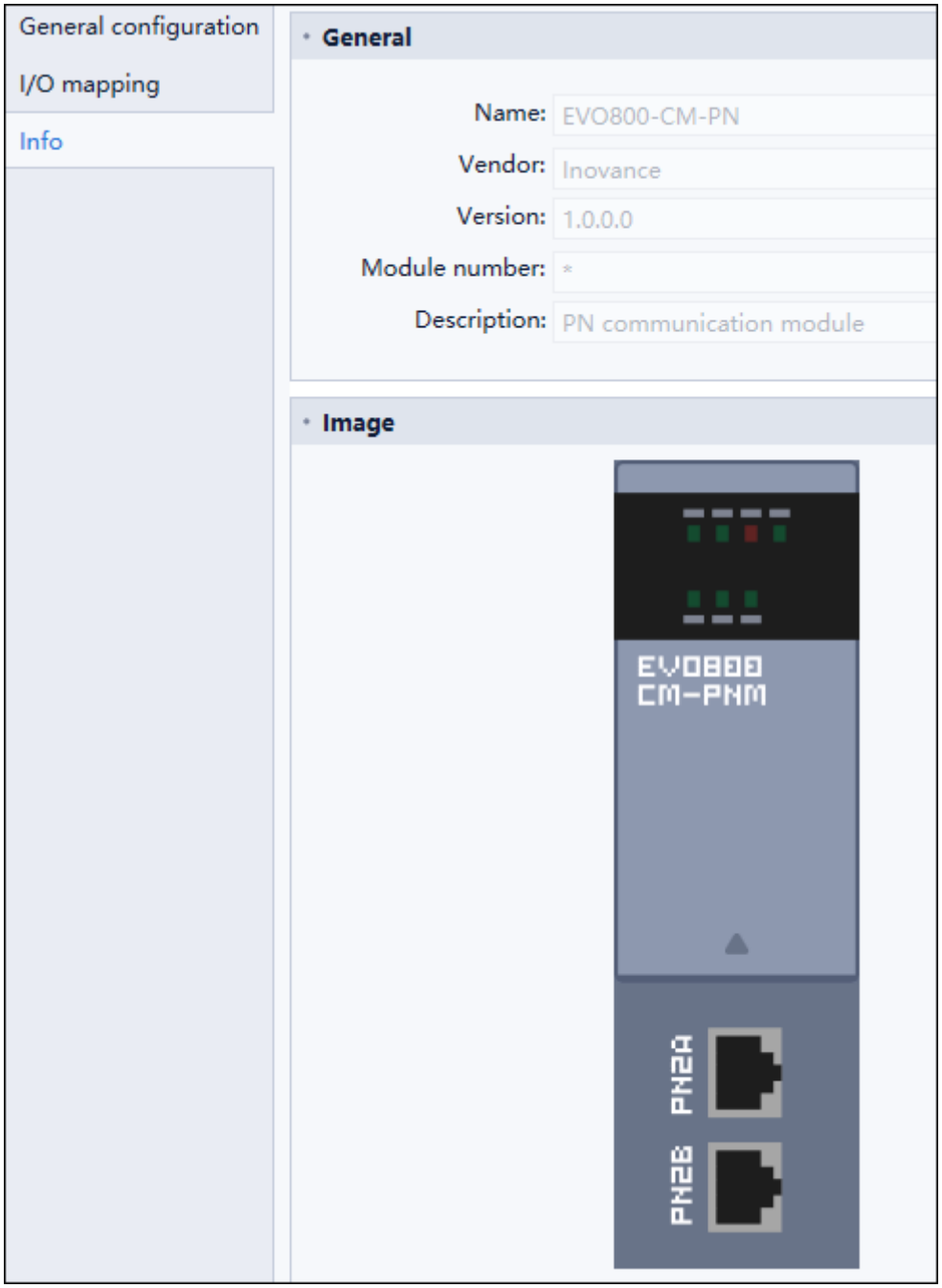
After importing the device I/O configuration file on the general configuration page, you can view the I/O settings related to PN communication, as shown in the figure below:



For details about the configuration parameters, see 10.6.3.3.9 I/O Mapping.

10.6.5.3.3.3 Information

This interface displays the general information and image of the device, as shown in the following figure.

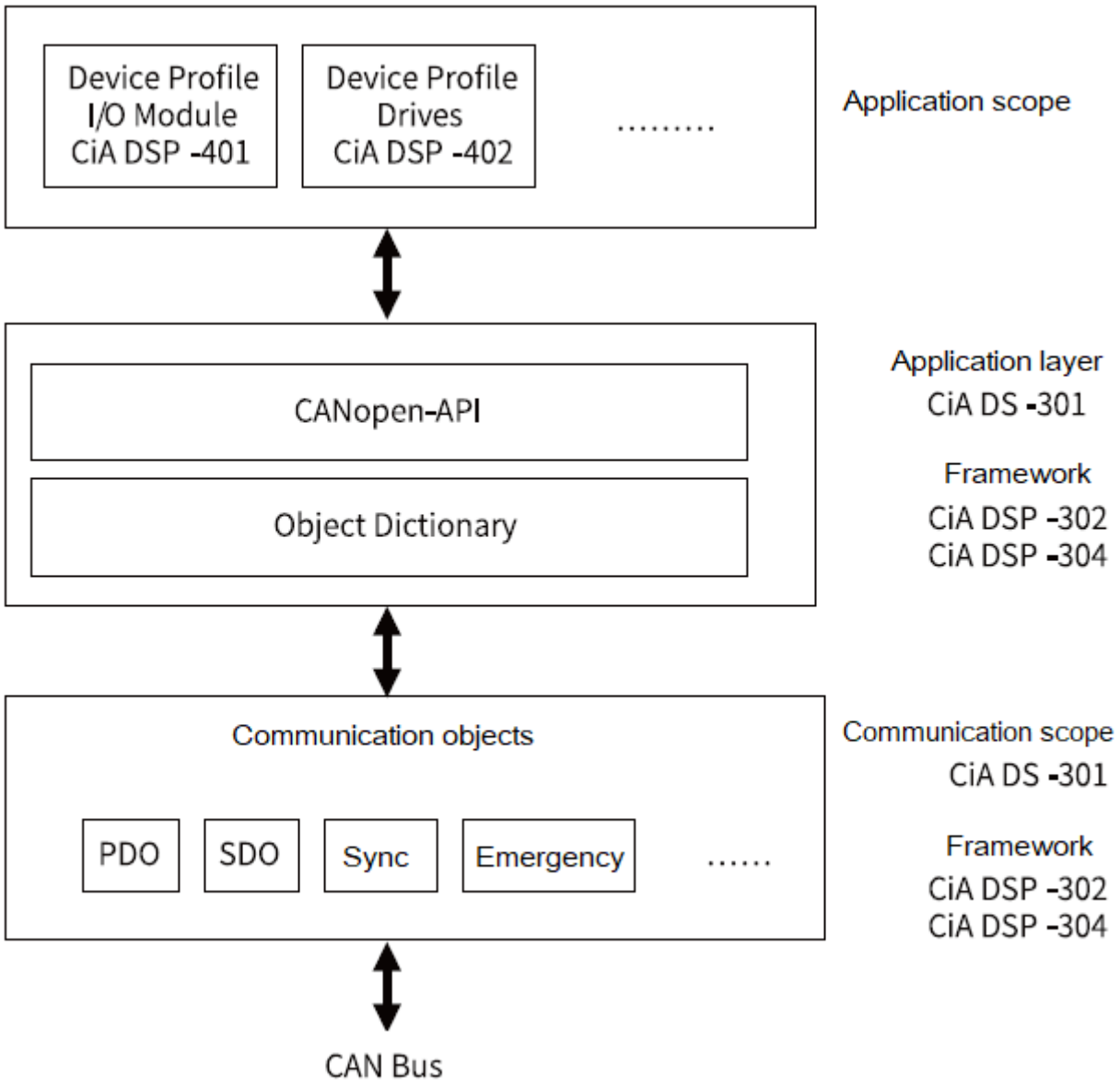


10.6.6 CANopen Communication Configuration (EVO400, EVO500, EVO700, APC500)

10.6.6.1 Usage Workflow

Introduction

CANopen is a family of industrial communication protocols for distributed automation control devices based on the CAN (Controller Area Network) bus. It was promoted and developed by the manufacturers and user association CiA (CAN in Automation), and standardized by the end of 2002 under the standard number CENELEC EN 50325-4. CANopen defines application layer protocols, communication layer protocols, and various application protocols, featuring openness, reliability, and flexibility. It is widely used in fields such as mechanical engineering, robotics, building automation, and automotive electronics. The following figure shows the overall architecture of CANopen.



The application layer provides applications with acknowledged and unacknowledged services and defines communication objects. Those services can be used to request data from a server. Communication objects are used during data exchange. They are used to exchange process data and service data, process management, or system clock synchronization, fault status management and node state control/ monitoring. A communication object is defined by a structure, transmission type, and CAN identifier. The typical parameters of communication objects include: CAN identifiers used for data transmission, transmission type of messages, inhibit time or event time, which are all defined by the communication protocol.

Each CANopen device uses an object dictionary as its primary data structure. The object dictionary serves as the primary medium for data exchange between the application layer and CAN bus communication. The object dictionary can be accessed from both the application layer and the CAN bus side via specific messages. These access entries can be regarded as variables or programmer-defined areas.

Each entry in the object dictionary has an index and a sub-index. Using the index structure, entries in the object dictionary can be accurately located.

The CANopen protocol stack provides standard API functions to define entries in the object dictionary, including their read and write attributes. The object dictionary can also be accessed through communication objects via the CAN bus.

In object dictionaries, attributes of each portal must be defined, including the data type, access permission, data transmission for process data objects (PDOs), and variable range.

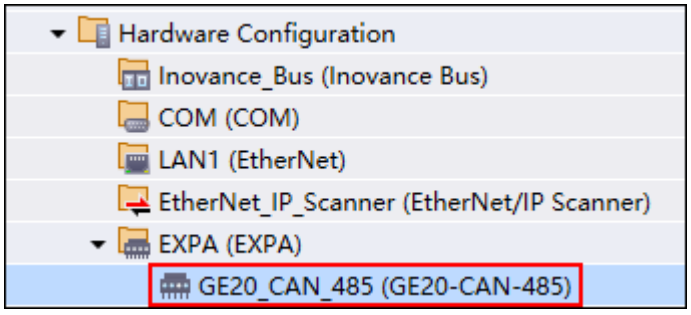
- PDO (Process data object) is used to transmit process data in real time.
- Service data objects (SDOs) are used to access the object dictionary of a device, set parameters, or transmit non-real-time data.
- The SYNC message, which does not contain any data, is sent from the producer to CAN network cyclically to trigger transmission of the PDO data.
- The Emergency message is triggered by internal fatal faults of the device, which consists of emergency fault code, fault register, and manufacturer-specific faults.

The CiA DS-301 specification defines the application layer and communication protocol. The CiA DSP-302 specification defines the framework for programmable CANopen devices. The CiA DSP-304 specification defines the framework for secure redundant data transmission. Specific device data descriptions are defined in application protocols composed of individual device protocols, such as the CiA DSP-401 specification defining the data format for I/O modules, and the CiA DSP-402 specification defining the data format for drive control.

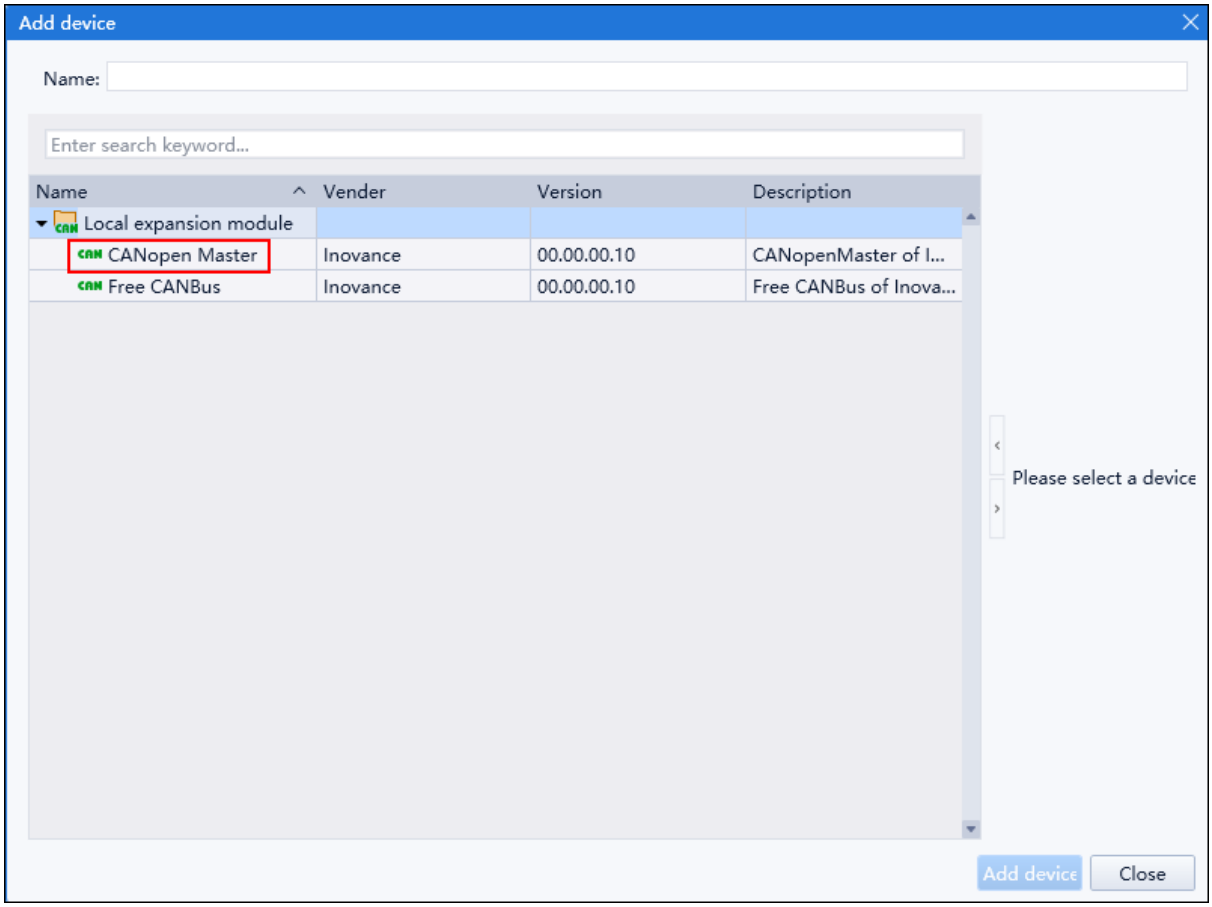
CANopen usage process

The general CANopen usage process is as follows, illustrated using the EVO500 series PLC:

1. Add a PLC device supporting the CANopen protocol to the project tree.
2. In the left project tree, add a CANopen master under the "Hardware configuration" node.
   For controllers with a CAN port, select the CAN bus node and add a CANopen master.
   For controllers without a CAN port, such as the EVO500 series, add a CAN module via an expansion card.



   Right-click the CAN module and select "Add device" to add a CANopen master.



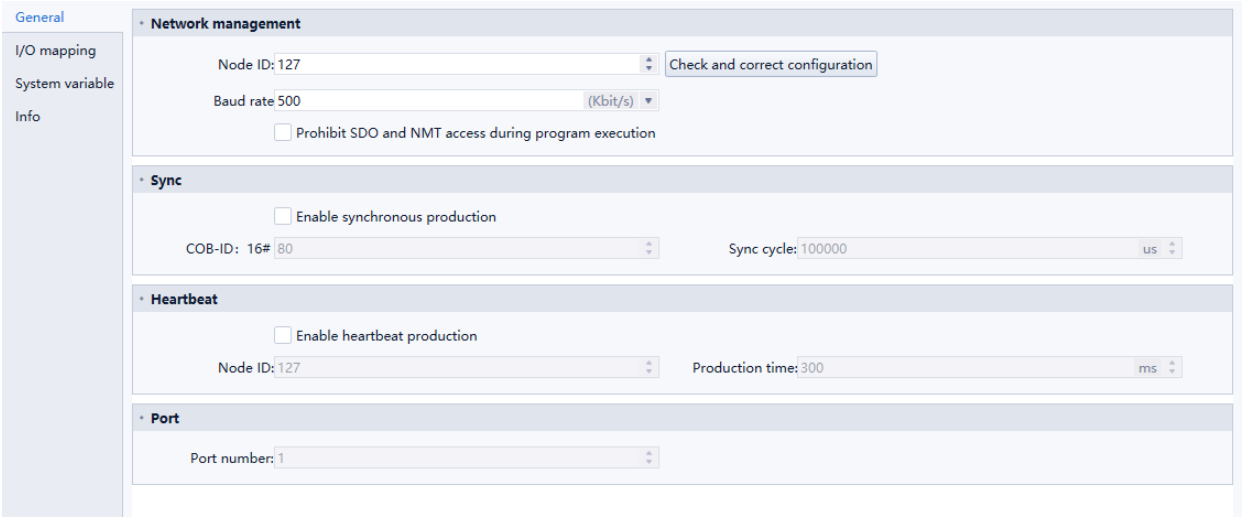
After adding the CANopen master, the CANopen bus task is automatically added. By default, the CANopen bus uses this task for I/O refresh.

3. Add CANopen slave devices and modules to the master according to the hardware structure.
   A CANopen slave refers to a remote CANopen device. For third-party slaves, import them by installing an EDS file in "Menu bar > Tools > Device library", then add the third-party slave.
4. Set configuration parameters of the master, slave, and module properly. Generally, the slave node ID, PDO, and mapping are generated automatically based on the EDS file. Special configurations may involve manual modification.

10.6.6.2 CANopen Master Configuration

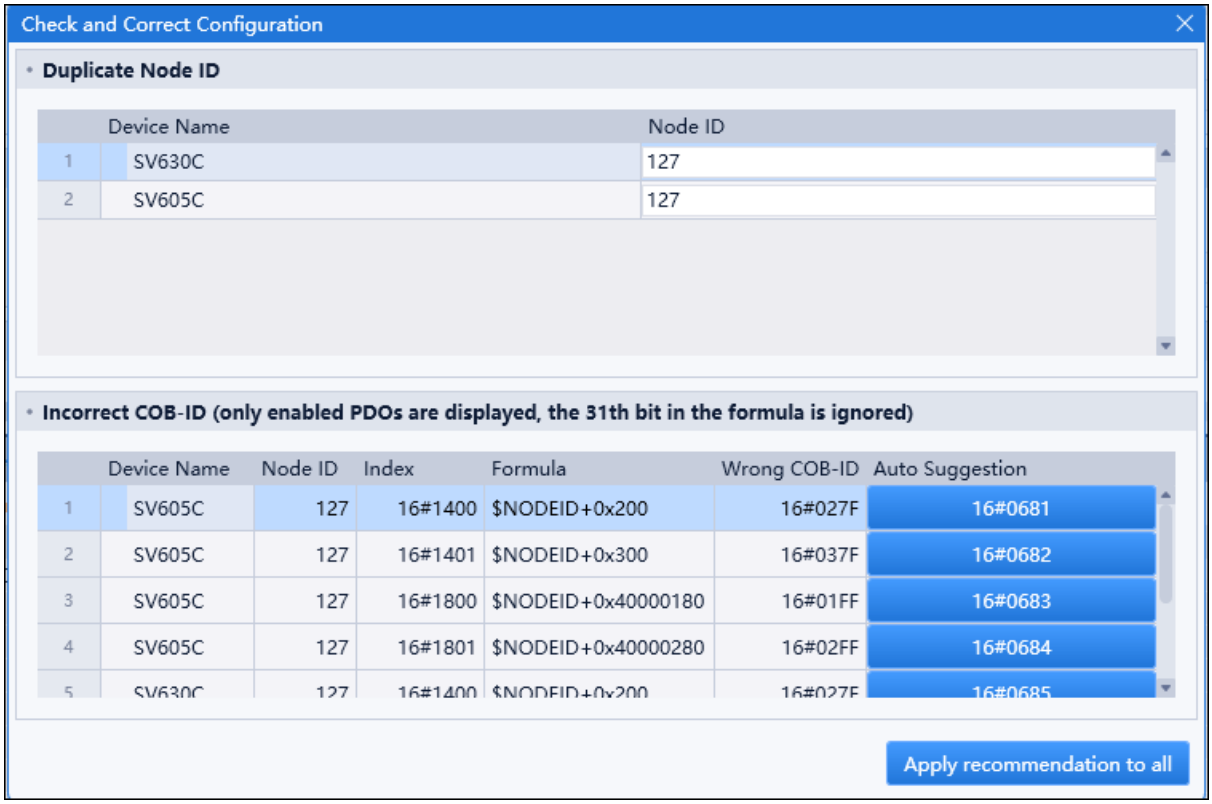
10.6.6.2.1 General Settings

Double-click the master node to open the CANopen master configuration interface.



Network management

- Node ID
  A decimal number indicating the unique identifier of the master in the CANopen network. Default is 127, with a range from 1 to 127.
- Check and correct configuration
  When multiple slaves are added to a CANopen system, the EDS files of different slaves may have default COB-ID configurations or manually modified node IDs, causing duplicate node IDs or COB-ID conflicts on the slave or master side. Click "Check and correct configuration", then click "Apply to all" in the pop-up window to resolve duplicate node IDs or conflicting COB-IDs.



  - Duplicate node ID: This box contains a list of all slave devices with the same node ID. Node ID conflicts between master and slave are also displayed in this list. Users can reassign new node IDs by editing the "Node ID" column. After modification, duplicate node IDs are automatically cleared.

- Incorrect COB-ID: This box displays a list of all slave devices with conflicting or invalid COB-IDs. COB-IDs in PDO can be modified in two ways:
  - Click the button in the "Auto suggest" column to modify the COB-ID for the current slave index according to the value displayed on the button.
  - Click "Apply to all" to modify all incorrect COB-IDs according to the values displayed in the "Auto suggest" column.

- Baud rate
  The baud rate used for transmission on the bus. Unit is kbit/s. The following baud rates can be set: 10, 20, 50, 100, 125, 250, 500 (default), 800, and 1000.

> NOTE
>
> If the CPU module is at the beginning or end of the network, the matching resistor on the CANopen port must be set to ON. Communication distance is related to baud rate, so a reasonable baud rate must be set.

- Prohibit SDO/NMT access during program execution
  If this item is checked, the slave cannot be accessed via SDO or NMT during program execution. For example, SDO and NMT access to the slave cannot be performed in the user program or on the slave commissioning page.

Sync

- Enable sync production
  If enabled (default: disabled), the master will send synchronous messages. Only one slave in a CANopen bus system can enable synchronous production. Synchronous type PDO sends messages according to the set type after the synchronous message is sent.
- COB-ID
  Communication object identifier, used to identify the synchronous message ID. The value is 16#80 and cannot be modified. If the slave enables synchronous production, the same COB-ID is used for identification.
- Sync period
  Synchronous messages are sent at time intervals defined by the synchronous period. The unit is microseconds, with a range from 2000 μs to 4294967000 μs. It must be an integer multiple of the bus task time.

Heartbeat

Heartbeat is another node protection mechanism: Unlike the node watchdog function, heartbeat can be triggered by either the master or the slave. Typically, the master sends heartbeats to the slave devices. The slave sets the node ID of the master to be monitored, enabling the slave to monitor the master.

- Enable heartbeat production
  If enabled (default: disabled), the master will send heartbeat messages.
- Node ID
  The unique identifier for sending heartbeat messages. Default is the master node ID, with a range from 1 to 127.

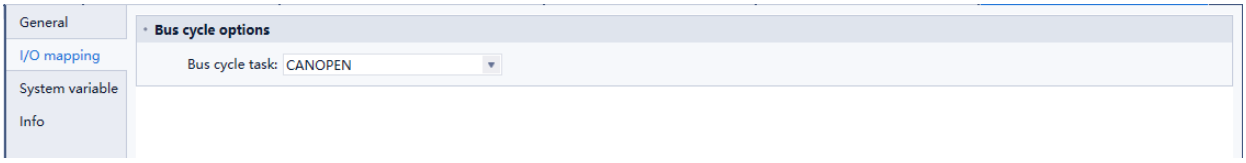
- Production time
  The time interval for sending heartbeat messages (unit: milliseconds), with a range from 2 ms to 32767 ms. It must be an integer multiple of the bus task time. Default is 300 ms.

Port

- Port number
  Used to distinguish between different CAN ports when multiple CAN ports exist.

10.6.6.2.2 I/O Mapping

All task names and bus cycle settings will be displayed in the bus cycle options. For CANopen communication, it is recommended to prioritize the default CANOPEN task.



10.6.6.2.3 System Variables

In the "System variable" tab, an instance (variable) of type _SYS_SlaveState is specified for the master and the device ID value of the master is displayed.



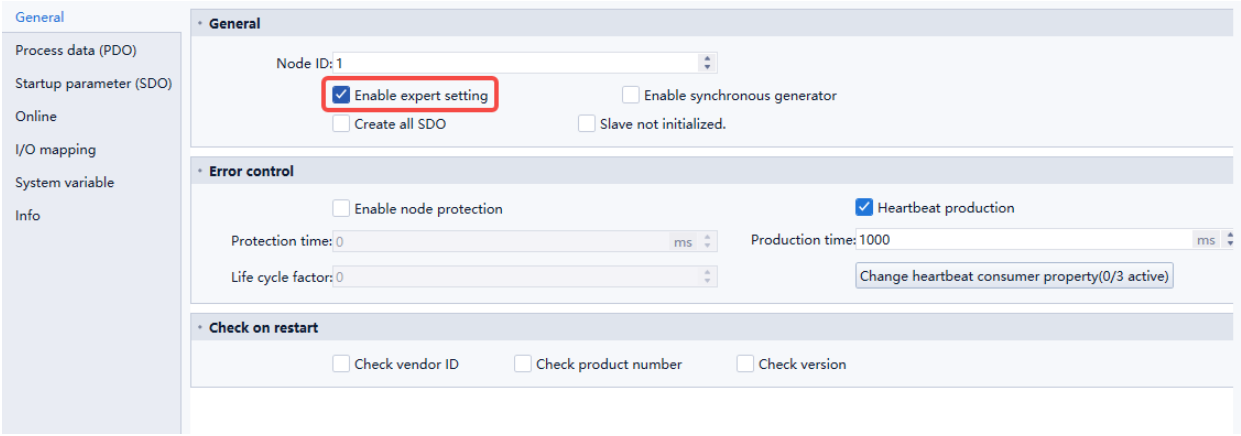
10.6.6.2.4 Information

The "Information" tab displays the following information for the current module: name, vendor, group, version, module number, and description.

10.6.6.3 CANopen Slave Configuration

10.6.6.3.1 General Settings

Double-click the slave node to open the CANopen slave configuration interface.



General Settings

- Node ID
  The unique identifier of the slave in the CANopen network ranges from 1 to 127 (decimal), and must match the slave's own identifier (such as a DIP switch).
- Enable expert settings
  Enable this option to configure expert parameters, such as node protection, heartbeat production, emergency messages, restart checks, PDO mapping operations, system SDO display, and SDO exception jumps.
- Enable sync generator
  If this option is enabled (default: disabled), the slave will send synchronization information. A CANopen bus system can only have one enabled sync producer. Synchronization transmission parameters use the master's synchronization configuration parameters.
- Create all SDOs
  Enable to create SDO objects in the object dictionary with writable attributes, such as object access attributes being RW, WO, RWR, RWW. The created SDOs are displayed in the service data object interface.
- Slave uninitialized
  If enabled, SDO parameters are not sent at startup.

Error control

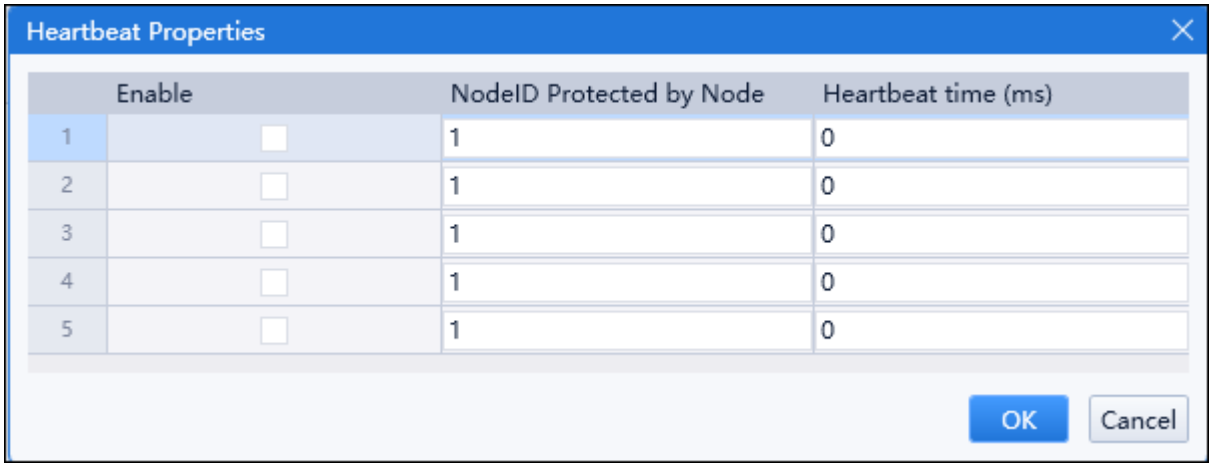
Used to detect the online status of nodes, including node protection and heartbeats.

Node protection is used by the master to detect the online status of the slave. The master periodically sends node protection information, and the slave responds to the information. If the slave does not respond within the node protection time (protection time × lifecycle factor), the master considers the slave lost.

Heartbeats can be produced by the slave or the master. The producer broadcasts heartbeat messages on the CAN bus, and the consumer consumes the heartbeat. If a node is set to consume heartbeats and does not detect the heartbeat production corresponding to the node ID within the set heartbeat consumption time, it considers the node lost. Typically, slaves consume the master's heartbeat to detect the online status of the master.

- Enable node protection
  Node protection and heartbeat production are mutually exclusive. The master periodically sends node protection frames during the protection time. If the slave does not provide a response containing a specific protection COB-ID (communication object ID) within the node protection time (protection time × lifecycle factor), the slave is considered to be offline.

- Protection time
  The interval at which the master sends node protection frames, ranging from 10 ms to 65535 ms. It must be an integer multiple of the bus task cycle.
- Lifecycle factor
  Used in conjunction with protection time. If the slave does not respond within the node protection time (protection time × lifecycle factor), the master considers the slave lost. The value range is from 1 to 255.
- Heartbeat production
  Check this option to enable heartbeat production by the slave, which sends heartbeat production frames at the production time interval. This setting is mutually exclusive with node protection.
- Production time
  The interval at which the slave sends heartbeat production frames, ranging from 10 ms to 32767 ms. It must be an integer multiple of the bus task cycle.
- Change heartbeat consumer properties
  Click this button to set the heartbeat producers consumed by the slave. By heartbeat consumption configuration, the slave can check the online status of the corresponding heartbeat-producing slave. Typically, the slave consumes the heartbeat produced by the master.



The heartbeat consumption configuration list is used to configure the heartbeat producers being consumed. Configuration is only possible after enabling "Heartbeat production". After enabling, the "NodeID protected by node" (range: 1 to 127) defaults to the master's heartbeat production node ID. If heartbeat production is disabled for the master, this NodeID is 0. Heartbeat time defaults to 1.5 times the master's heartbeat production time, ranging from 1 to 65535.

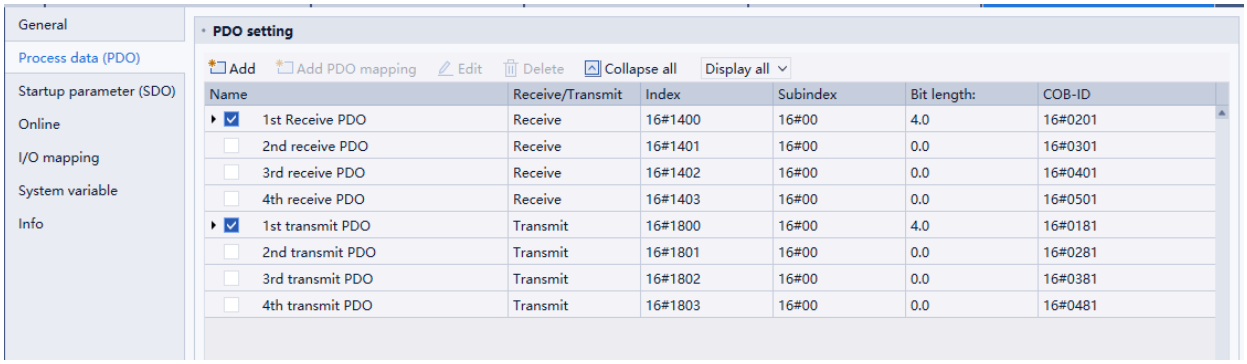
Check on restart

- Check supplier ID
  If this option is checked, the slave will check whether the vendor ID (index 1018, sub-index 01) in the object dictionary matches the slave's own vendor ID. If they do not match, the slave cannot operate normally.
- Check product number
  If this option is checked, the slave will check whether the product number (index 1018, sub-index 02) in the object dictionary matches the slave's own product number. If they do not match, the slave cannot operate normally.
- Version check
  If this option is checked, the slave will check whether the version (index 1018, sub-index 03) in the object dictionary matches the slave's own version. If they do not match, the slave cannot operate normally.

10.6.6.3.2 Process Data (PDO)

PDO (Process Data Object) is used for real-time data transmission between the master and slaves, where the received PDO is the real-time data sent from the master to the slave.

PDO includes communication parameters and mapping parameters. The communication parameters include COB-ID, transmission type, transmission control, and so on. The PDO mapping parameter indicates that the PDO transmission data comes from the index and sub-index of the object dictionary.

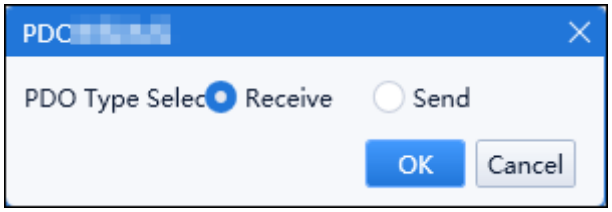


Name	Receive/Transmit	Index	Subindex	Bit length:	COB-ID
1st Receive PDO	Receive	16#1400	16#00	4.0	16#0201
2nd receive PDO	Receive	16#1401	16#00	0.0	16#0301
3rd receive PDO	Receive	16#1402	16#00	0.0	16#0401
4th receive PDO	Receive	16#1403	16#00	0.0	16#0501
1st transmit PDO	Transmit	16#1800	16#00	4.0	16#0181
2nd transmit PDO	Transmit	16#1801	16#00	0.0	16#0281
3rd transmit PDO	Transmit	16#1802	16#00	0.0	16#0381
4th transmit PDO	Transmit	16#1803	16#00	0.0	16#0481

Add

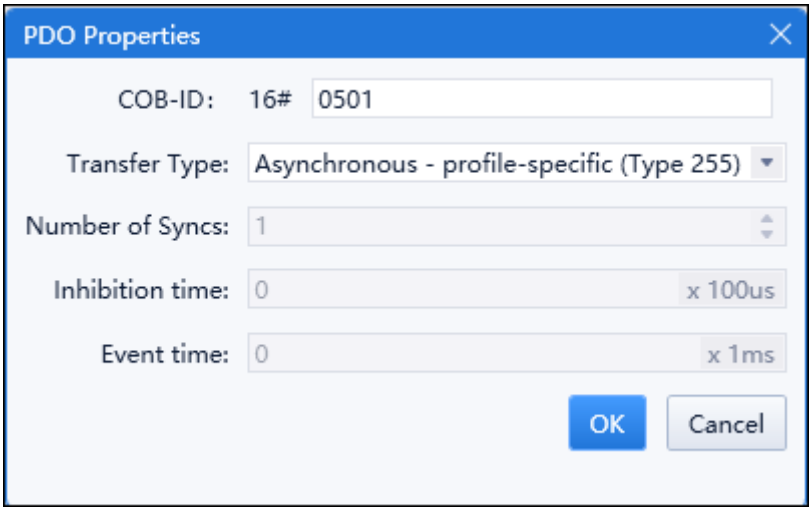
Click "Add" to select the PDO type in the pop-up window.

The added PDO is appended to the end of the list by default. The maximum number of received PDOs in the slave is determined by the number of object dictionary entries from index 1400 to 1600. PDOs exceeding this limit cannot be added. The name and index of the added PDO are automatically obtained from the object dictionary based on usage order and cannot be modified. Select the PDO type in the pop-up "PDO type selection" window. For details, see "PDO properties".



- Receive PDO: The receive PDO comes from the objects in the object dictionary starting from index 1400 and ending at index 1600. The default communication parameters for each PDO come from its corresponding subindexes. The objects that can be mapped by the receive PDO come from objects in the object dictionary that allow write access, such as RW, RWW, and WO.
- Transmit PDO: Indicates the real time data sent by the slave to the master. The transmit PDO comes from the objects in the object dictionary starting from index 1800 and ending at index 1A00. The default communication parameters for each PDO come from its corresponding subindexes. The objects that can be mapped by the transmit PDO come from objects in the object dictionary that allow write access, such as RW, RWR, RO, and CONST.

PDO properties



- COB-ID: Synchronous message ID with a range of 180-57F; 681-6DF.
- Transfer type:
  - Synchronous - Non-cyclic (Type 0): The transfer is triggered by a remote frame or by specific events defined in the device sub-protocol for certain objects.
  - Synchronous - Cyclic (Type 1-240): The transfer is triggered after every 1 to 240 SYNC messages.
  - Asynchronous - Manufacturer specified (Type 254): The transfer is triggered by specific events defined by the manufacturer for certain objects.
  - Asynchronous - Device file specified (Type 255): The transfer is triggered by specific events defined in the device sub-protocol for certain objects.

> NOTE
>
> In non-expert mode, only PDO communication parameters can be modified. PDOs and PDO mappings cannot be added or deleted.

The following table introduces different PDO transmission modes defined by the transmission type, which is part of the PDO communication parameter object and is defined by an 8-bit unsigned integer.

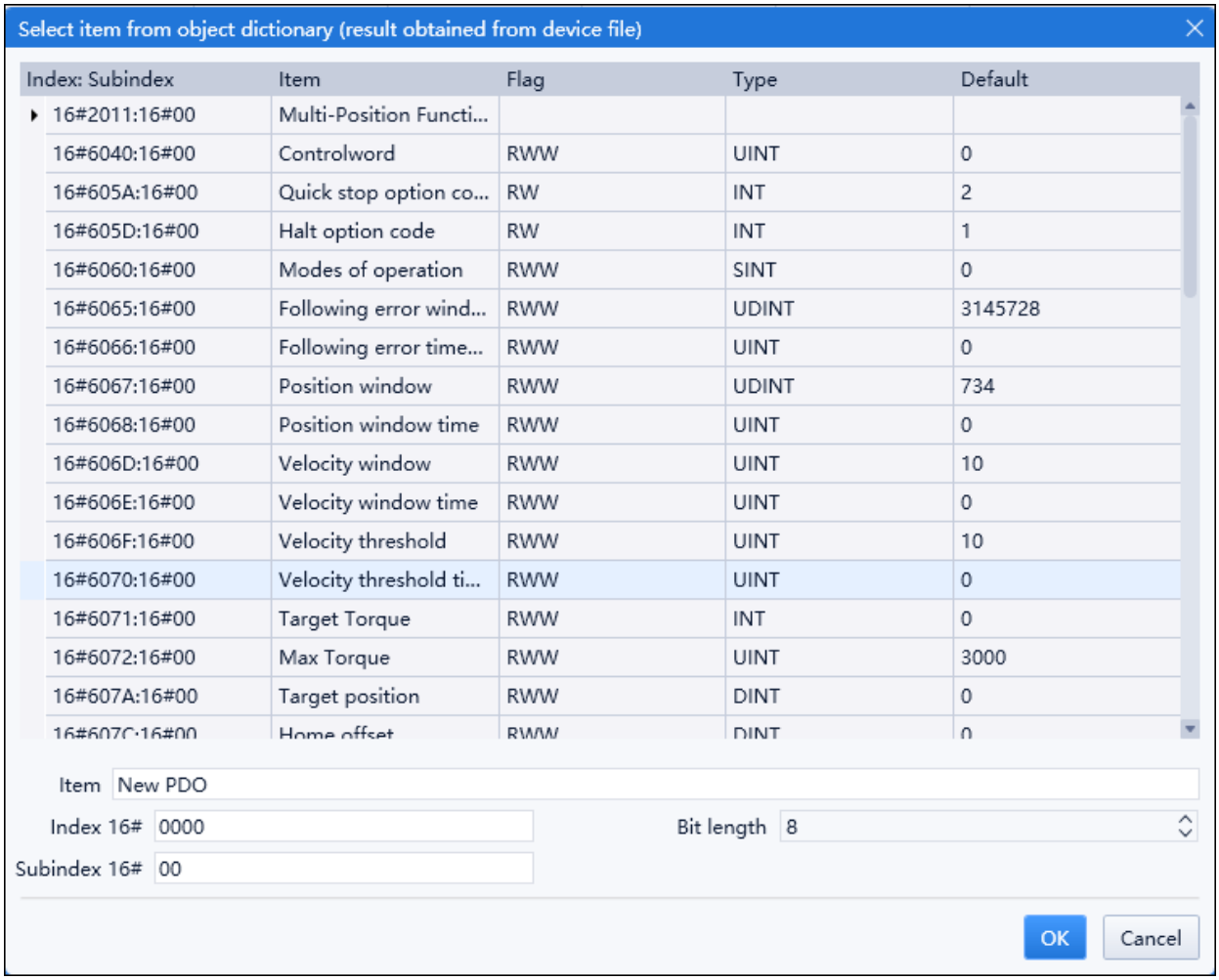
Transmission Type	Trigger Conditions for PDO Communication (B = both needed; O = one or both)			PDO Transmission
	Sync	RTR	Event	
0	B	-	B	Synchronous, non-cyclic
1 to 240	O	-	-	Synchronous, cyclic
241 to 251	-	-	-	Reserved
252	B	B	-	Synchronous, after RTR
253	-	O	-	Asynchronous, after RTR

Transmission Type	Trigger Conditions for PDO Communication (B = both needed; O = one or both)			PDO Transmission
	Sync	RTR	Event	
254	-	O	O	Asynchronous, vendor-specific event
255	-	O	O	Asynchronous, sub-protocol specific event
Note: SYNC - upon receiving a SYNC-object RTR - upon receiving a remote frame Event - such as a value change or timer interrupt When the transmission type is from 1 to 240, this number represents the number of SYNC objects between two PDOs.				

The default transmission type of each PDO comes from the object corresponding to PDO subindex 02 in the object dictionary. If the write permission of the corresponding object dictionary is not granted, the transmission type cannot be changed.

- Number of syncs: Indicates the number of sync frames the slave receives before it can start processing PDO data transmission. It depends on the transmission type and can be modified only when the transmission type is within the range of 1 to 240.
- Inhibition time: Indicates the product of the minimum interval between the transmission of two messages by the same PDO and 100 μs. This parameter prevents PDOs from being sent too frequently when a value is changed.
- Event time: Indicates the interval time (in ms) between two messages transmitted via the same PDO. The default event time for each PDO is derived from the object at sub-index 05 of the corresponding PDO in the object dictionary. If the write permission of the corresponding object dictionary is not granted, the event time cannot be changed.

Adding a PDO mapping



Add a PDO mapping to the currently selected PDO, which is added after the current PDO. The maximum length of PDO mapping is 64 bits. The PDO mapping comes from the object dictionary. The objects that can be mapped by the receive PDO come from objects in the object dictionary that allow write access, such as RW, RWW, and WO.

Editing

Edit the currently selected PDO communication parameters or PDO mapping parameters. If the PDO is selected, the PDO is edited. If the PDO mapping is selected, the PDO mapping is edited.

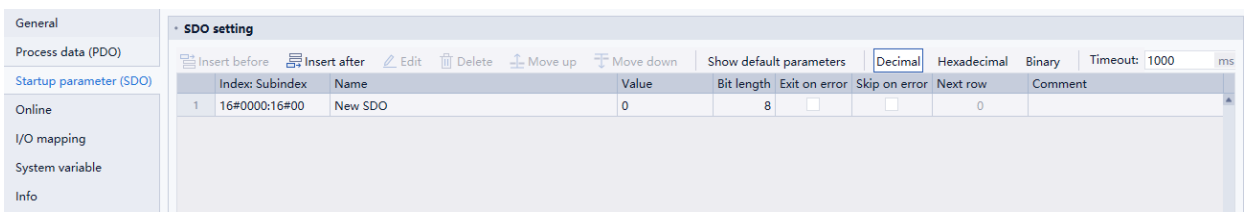
Delete

Delete the currently selected PDO or PDO mapping. If the PDO is selected, the PDO is deleted. If the PDO mapping is selected, the PDO mapping is deleted.

10.6.6.3.3 Startup Parameter (SDO)

SDOs are used to transmit data during slave initialization and operation. Settings on the "Service Data Object" page are written to the slave during slave initialization.

The SDO configuration interface allows you to configure selected SDOs, modify the transmission order of SDOs, and define the handling method when solutions occur during SDO transmission.



In the SDO configuration interface, click "Insert before" or "Insert after" to add an SDO before or after the selected entry. The added SDOs can be modified and moved up or down. Double-click the "Value" cell to modify the SDO value.

When SDO configuration faults occur, you can set the fault handling method. Select "Exit upon error" to directly exit configuration when a fault occurs, and no further SDOs will be configured. Select "Skip upon error" to directly jump to the selected line and continue execution downward when a fault occurs. If neither of these options is selected, the default handling method is "Skip upon error".

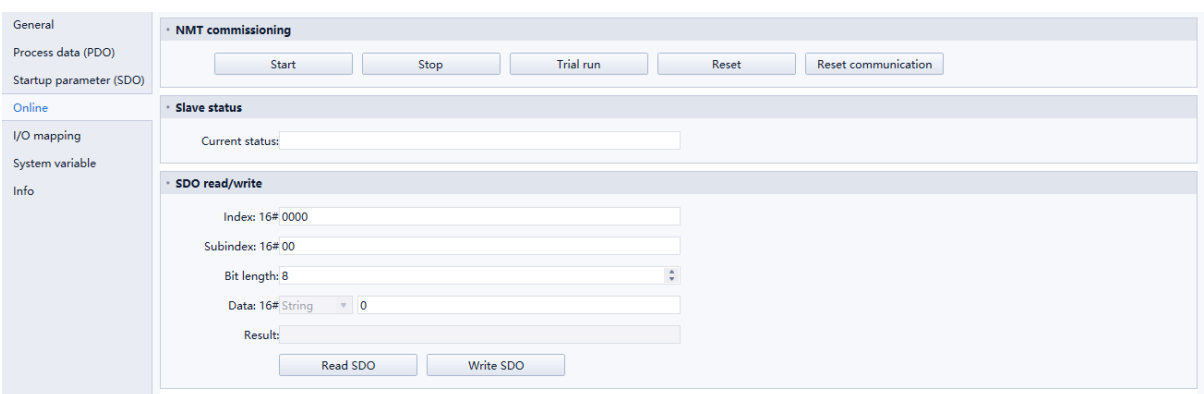
Sub-Function	Function description
Insert before	Insert an SDO parameter before the selected SDO entry. Note that all system SDOs are by default arranged at the top. It is not allowed to insert SDOs in the middle of system SDOs.
Insert after	Insert an SDO parameter after the selected SDO entry.
Edit	Edit the selected editable SDO.
Delete	Delete the selected editable SDO.
Move up	Move up the selected editable SDO. It cannot be moved above system SDOs.
Move down	Move down the selected editable SDO.
Timeout	The timeout time for SDOs. The default is 1000 ms, with a range from 0 ms to 4294967 ms.

NOTE

- Only in expert mode will the system SDOs and fault handling be displayed.
- When configuring SDO fault handling, be cautious with line jump processing. An upward jump may result in an infinite loop in SDO configuration.

10.6.6.3.4 Online

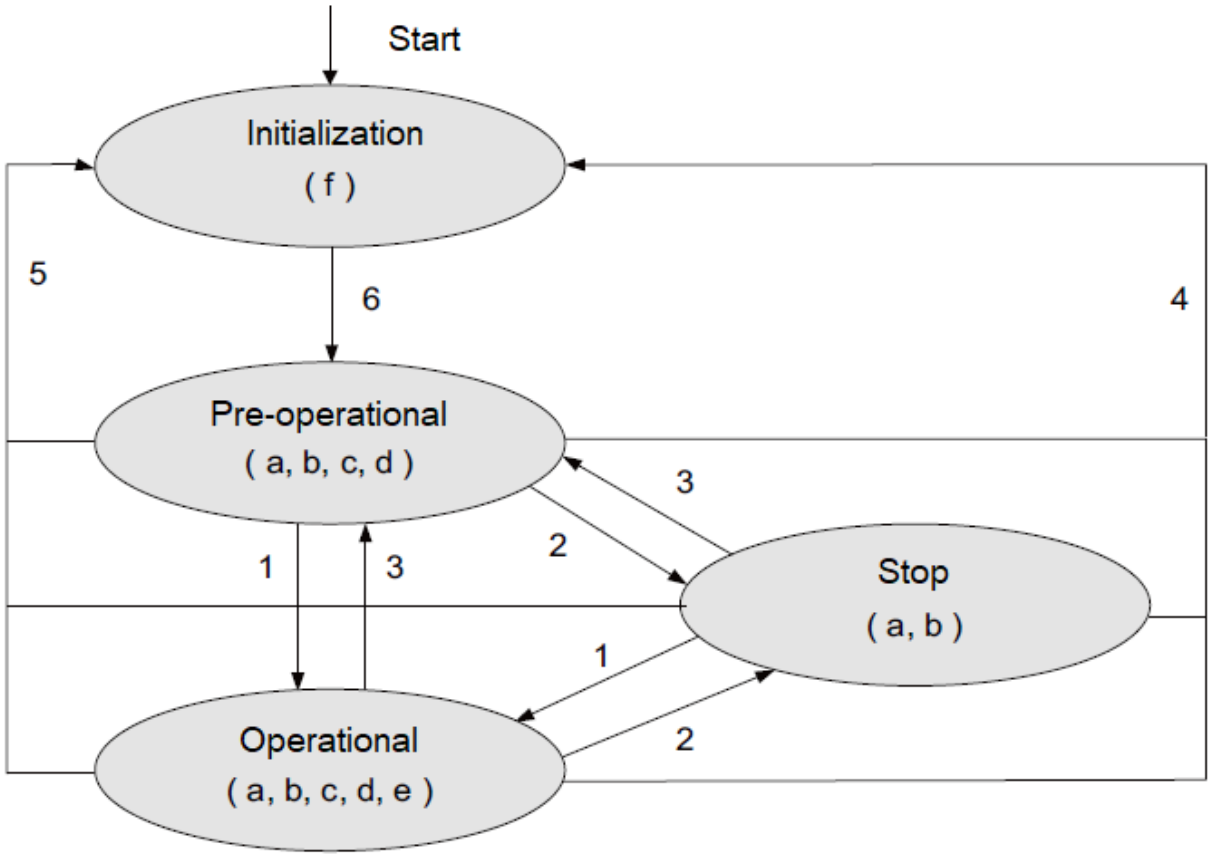
The online interface is used for master NMT control, SDO read/write, and diagnostic information acquisition.



NMT commissioning

NMT provides network management services (such as node initialization, start-up, stop, and failure detection). The services are implemented using a master-slave communication mode (with only one NMT master node).

The status of a slave node during startup can be represented by the following status transition diagram.



Note:

- a: NMT; b: Node guard; c: SDO; d: Emergency; e: PDO; f: Boot-up status transition (commands 1 to 5 are initiated by NMT service)
- NMT command words:

  1: Start_Remote_node (0x01, start node)

  2: Stop_Remote_Node (0x02, stop node)

  3: Enter_Pre-Operational_State (0x80, enter pre-operational)

  4: Reset_Node (0x81, reset node)

  5: Reset_Communication (0x82, reset communication)

  6: After device initialization, the system automatically enters Pre-Operational status and sends Boot-up messages.

Initialization includes application data initialization and communication initialization. Reset_Node resets all data of the slave node, while Reset_Communication only resets communication-related data.

The device completes initialization, enters the Pre_Operational state, and sends a Boot-up message. The NMT service can switch all or part of the nodes to different operating statuses at any time.

Description	Sub-Function	Function description	Scope of Features
NMT	Start	Run the slave node; PDO communication can only occur in the operational status.	When the slave node is in pre-operational or stopped status, click "Start" to switch it to operational status.
	Stop	Stop the slave node operation; in this status, all communication is stopped except node protection and heartbeat.	When the slave node is in pre-operational or operational status, click "Stop" to switch it to stopped status.
	Test run	The slave node enters pre-operational status; in this status, SDO communication is possible but PDO communication is not. After the slave node is initialized, it automatically enters pre-operational status.	When the slave node is in operational or stopped status, click "Test run" to switch it to pre-operational status.
	Reset a bit	First reset application configuration, then reset communication configuration data; after reset, the system automatically enters pre-operational status.	-
	Reset communication	Reset the slave communication configuration data. After reset, the system automatically enters pre-operational status.	-

SDO read/write

SDO is used to transfer large-volume, low-priority data between devices. Typical application scenarios include configuring devices on a CANopen network. It is mainly used during slave node operation to read or write SDO object values. To read or write an SDO object, the object index, sub-index, and bit length must be determined. When writing, the value must also be provided.

- Index: SDO read/write index, with a range from 16#0 to 16#FFFF.
- Sub-index: SDO read/write sub-index, with a range from 16#0 to 16#FF.
- Bit length: SDO read/write bit length, with options 8, 16, 24, and 32.
- Data: SDO read/write value; the first edit box contains hexadecimal value, the value after "=" is decimal. For SDO write operations, the value range is related to the bit length, with minimum value 0 and maximum value being the maximum representable by the bit length.
- Result: SDO read/write result; if read/write is faulty, a fault message is displayed.

10.6.6.3.5 I/O Mapping

For detailed information about I/O mapping, see 10.6.3.3.9 I/O Mapping.

10.6.6.3.6 System Variables

System variables specify variables for slaves, such as an instance (variable) of type J_SV605C_V1_0, supporting control of devices connected to the CANopen master via user programs. The device ID of the corresponding slave is also displayed.



10.6.6.3.7 Information

The "Information" tab displays the following information for the current module: name, vendor, group, version, description, and image.

10.6.6.4 CANopen Module

In CANopen slave configuration, a CANopen slave node can connect to two types of devices:

Modular devices: When connected to a CANopen slave node, the module comes with its own I/O mapping list. There is no need to use the CANopen slave I/O mapping dialog box; the slave node's PDO mapping is automatically added when the module is added.

Non-modular devices: The slave node dialog box includes an I/O mapping dialog box, and PDO mapping cannot be automatically configured.

10.6.6.5 CANopen Parameter Configuration

Configuration is an important part of the Inovance iFA programming software. If drives from other manufacturers are to be used, their EDS files must first be imported. Confirm that the drive program from other manufacturers strictly follows the CANopen communication specifications and the CIA402 standard. Correct parameter settings must be completed before calling the CANopen402 function block. Details are as follows:

Master configuration

Master configuration mainly includes communication baud rate, synchronization mode, synchronization period, heartbeat, and heartbeat interval time.

- Communication baud rate: The higher the baud rate, the higher the communication efficiency, but the shorter the communication distance. The default baud rate is 500 kbps.
- Synchronization mode: The periodic synchronization function must be selected; otherwise, the function block will not operate.
- Synchronization period: For systems with only CANopen axes and three slaves (receive/transmit PDO data < 8 bytes), it is recommended to configure a 4 ms period. It is recommended that the synchronization period match the CANopen task scan cycle time in the program.

- Heartbeat: The master sends a heartbeat frame at each heartbeat interval to allow the slave to monitor whether the master has gone offline. This function must be used in conjunction with the slave. Some slaves do not support heartbeat monitoring, so this function is disabled by default.
- Heartbeat time: The master sends a heartbeat frame to the slave at the set interval, with a default value of 300 ms. It takes effect only when the heartbeat function is enabled.

Slave configuration

Slave configuration mainly includes node ID, heartbeat, heartbeat time, PDO synchronization method, and synchronization object dictionary configuration.

- Node ID (station number): A basic communication parameter for the slave. The station number must match the actual physical station number.
- Heartbeat: The slave sends heartbeat frames to the specified station number, primarily used to allow other stations to monitor the communication status of this station. By default, this function is enabled.
- Heartbeat time: The slave sends heartbeat frames to the specified station number at the set interval. The default is 1000 ms and takes effect only when the heartbeat function is enabled.
- PDO synchronization method: The default is asynchronous mode, which must be changed to synchronous periodic mode.

PDO object dictionary configuration: The PDO object dictionary ensures that data is exchanged once per bus cycle between the master and slaves. The more entries, the more efficient the status exchange between master and slaves. However, an excessive number of PDOs will increase bus load, potentially causing bus data transmission delays, and in severe cases, communication failure. For axis control devices, based on application experience, the mandatory configuration items for transmit and receive PDOs are 6040, 6041, 6060, 6061, 607A, 6064 (6063), 60FF, 606C, 6081. Optional items are 6083, 6084, 607C, 6098.

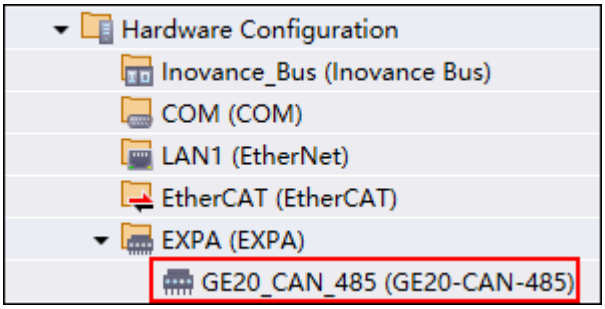
10.6.6.6 Programming Interface

For details, see the 18.4.11 CANopen Communication Commands.

10.6.7 CAN Free Communication Configuration (EVO400, EVO500, EVO700, APC500)

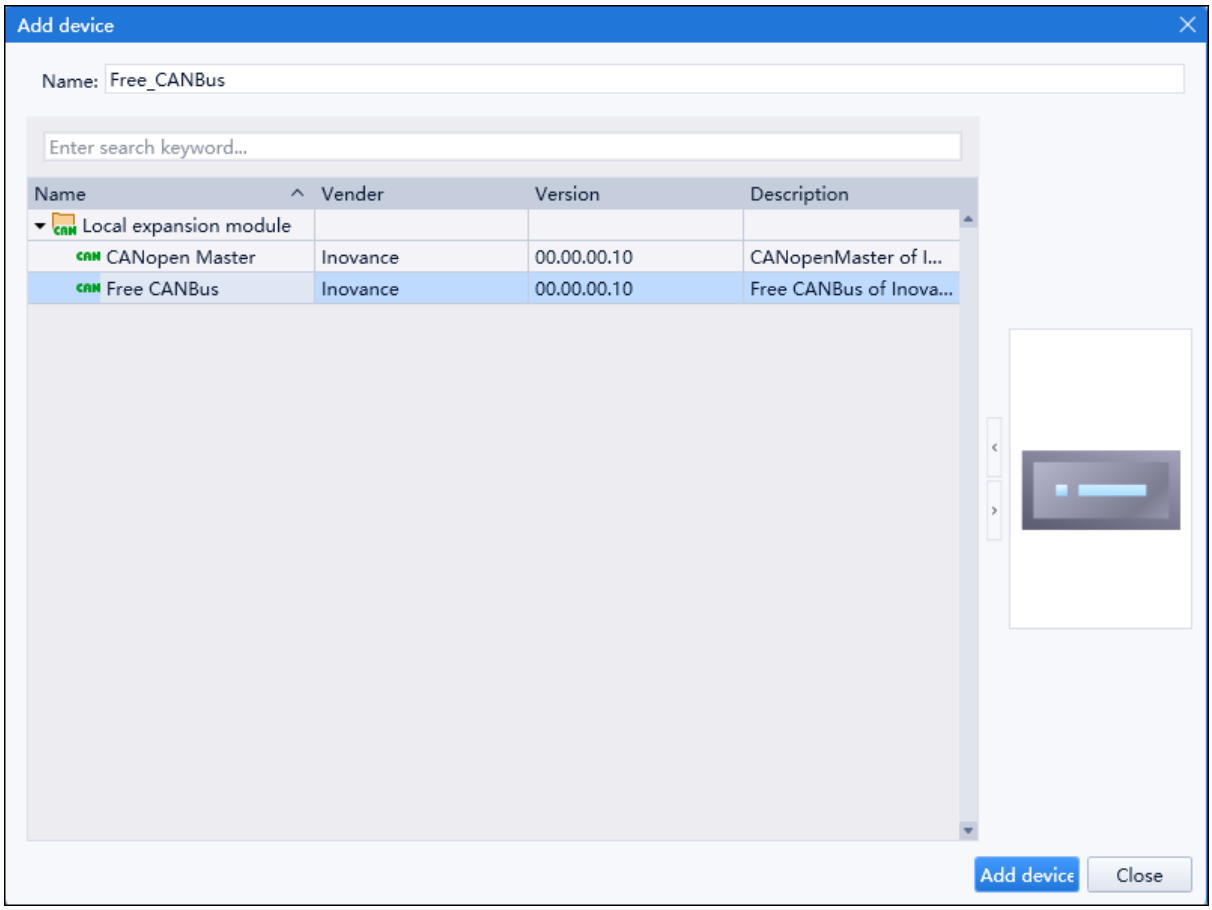
10.6.7.1 Network Configuration (EVO400, EVO500, APC500)

1. Right-click "EXPA (EXPA)" under the "Hardware configuration" node in the left project tree and select "Add device".
2. Select "GE20_CAN_485" in the pop-up window and click "Add device".
   After adding the expansion card, the corresponding CAN port "GE20_CAN_485(GE20-CAN-485)" is automatically generated.



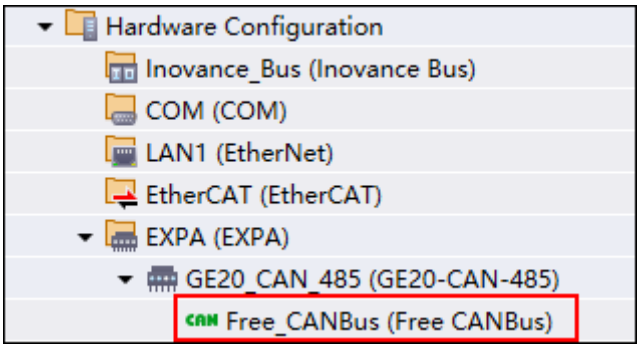
3. Right-click "GE20_CAN_485(GE20-CAN-485)" and select "Add device".

4. Select "Free CANBus" and click "Add device" to configure the CAN free protocol network.

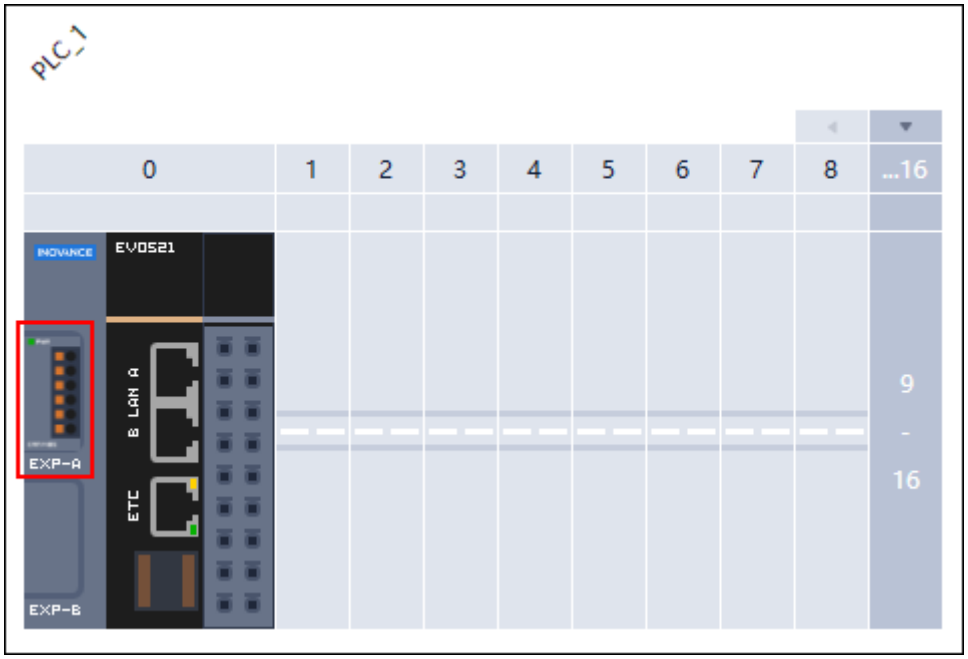


Result

The added CAN free protocol node is displayed in the left project tree.

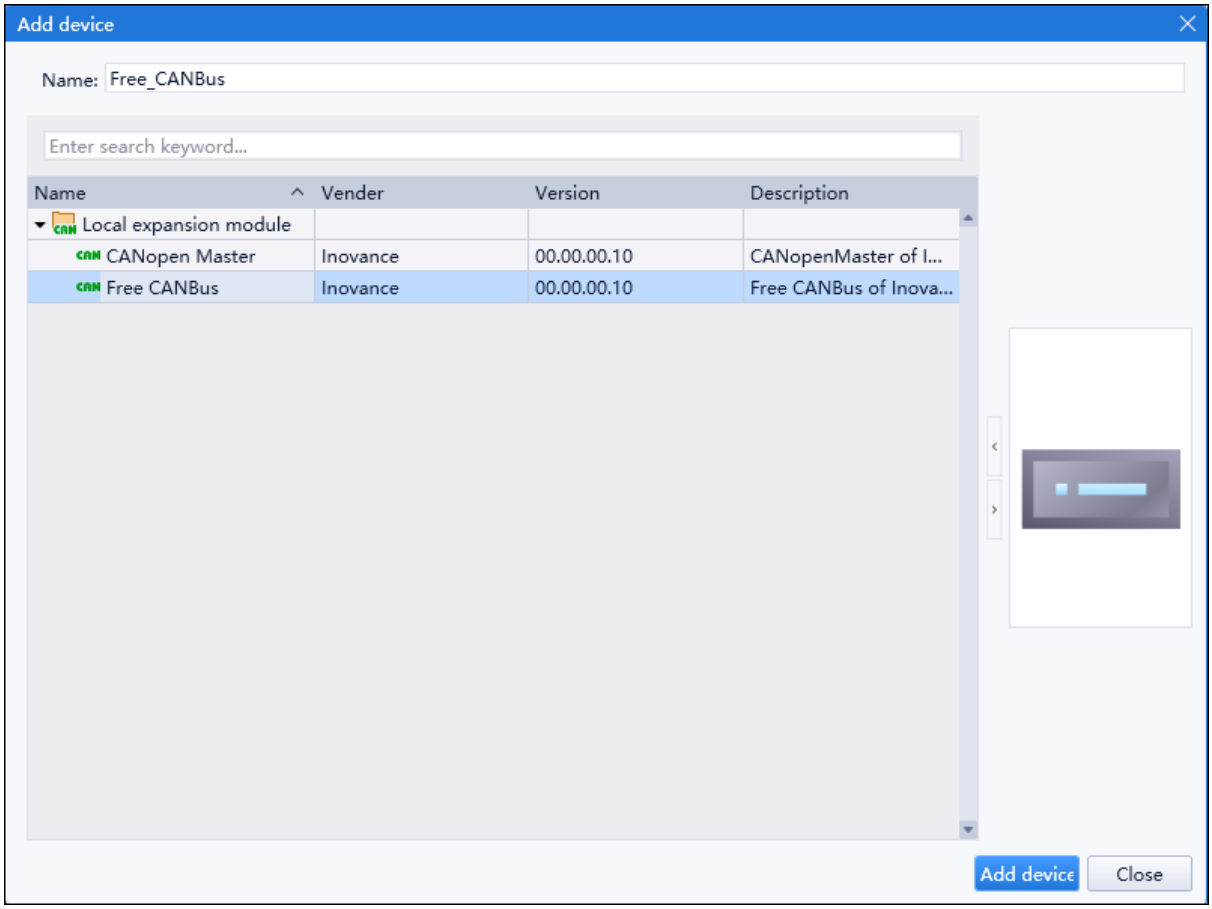


The device configuration view is updated synchronously, and the EXPA port of the expansion card shows that the CAN free protocol has been added.



10.6.7.2 Network Configuration (EVO700)

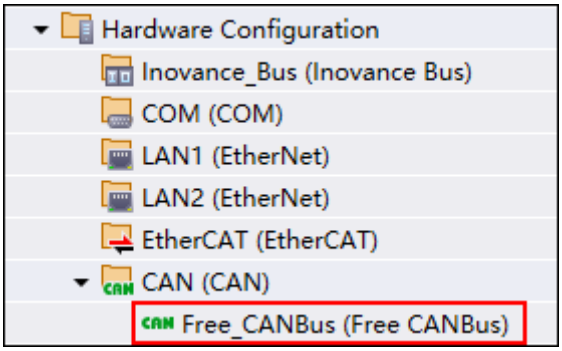
1. Right-click "CAN (CAN)" under the "Hardware configuration" node in the left project tree and select "Add device".



2. Select "Free CANBus" and click "Add device" to configure the CAN free protocol network.

Result

The added CAN free protocol node is displayed in the left project tree.



10.6.7.3 CAN Free Protocol Configuration

Network Settings

Port number

Used to distinguish between different CAN ports when multiple CAN ports exist.

Baud rate

Baud Rate: The baud rate used for transmission on the bus. Unit is kbps. The following baud rates can be set: 10, 20, 50, 100, 125, 250, 500 (default), 800, and 1000.

Information

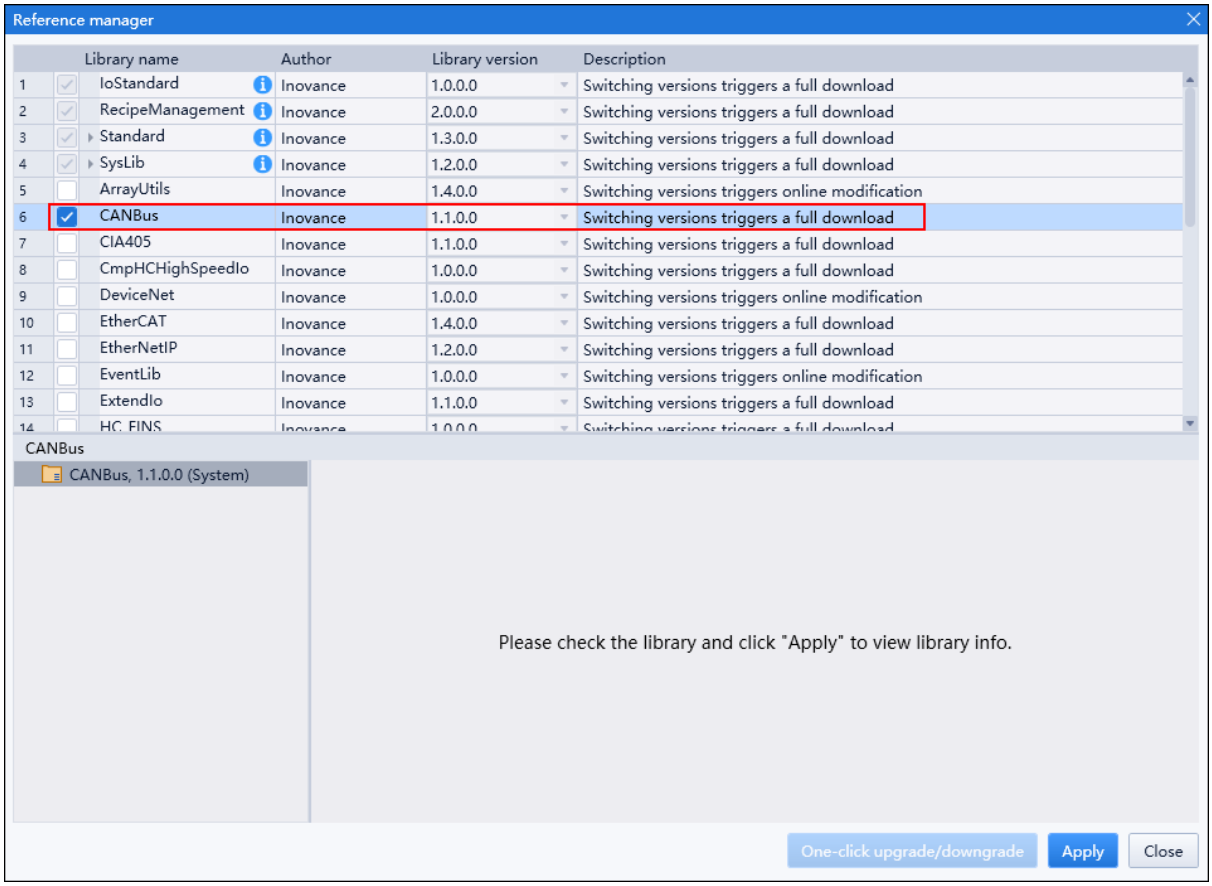
If the current device is available, the following basic information will be displayed: name, vendor, type, version number, module number, and description.

10.6.7.4 Usage Example

CAN free protocol only supports configuration through the CAN Bus function block.

The following example illustrates how to add a CAN Bus function block.

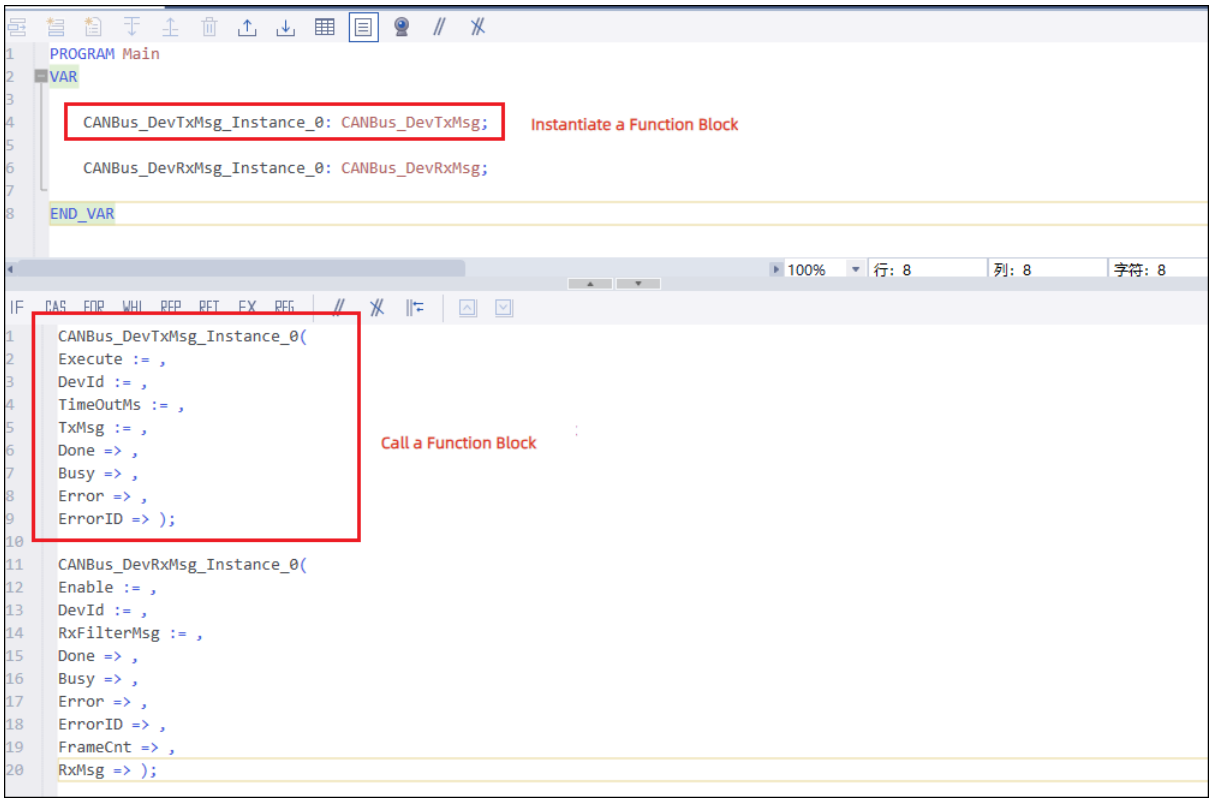
1. Double-click "Library reference" in the left project tree to open the "Reference manager" window.
2. Select "CANBus" and click "Apply" to reference the CANBus library into the project.

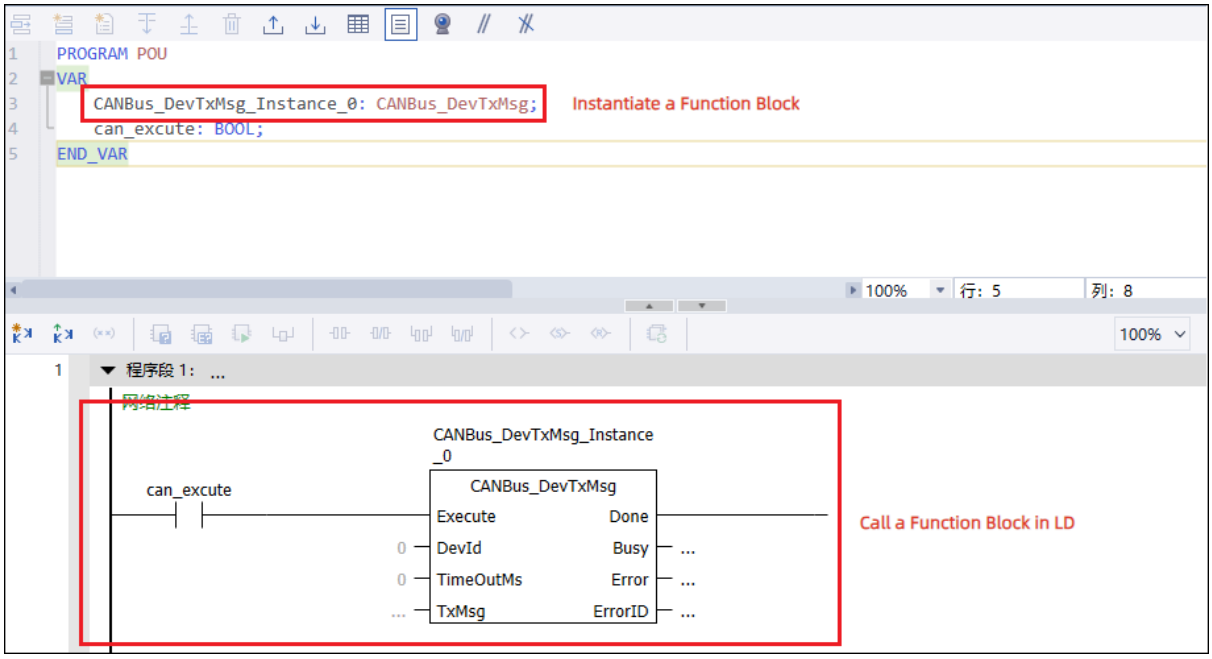


3.

Open the program editing page, double-click the function block instruction in the right toolbox under "Communication and diagnostics > CANBus instructions" to use and instantiate the instruction.

The CAN Bus function block instruction supports ST language and LD language, as shown in the figures below.





10.6.8 COM Communication Configuration (EVO400, EVO500, EVO700, APC500)

10.6.8.1 Modbus RTU Device Configuration

10.6.8.1.1 Adding a Modbus RTU Device

Specifying the current device as a Modbus master device

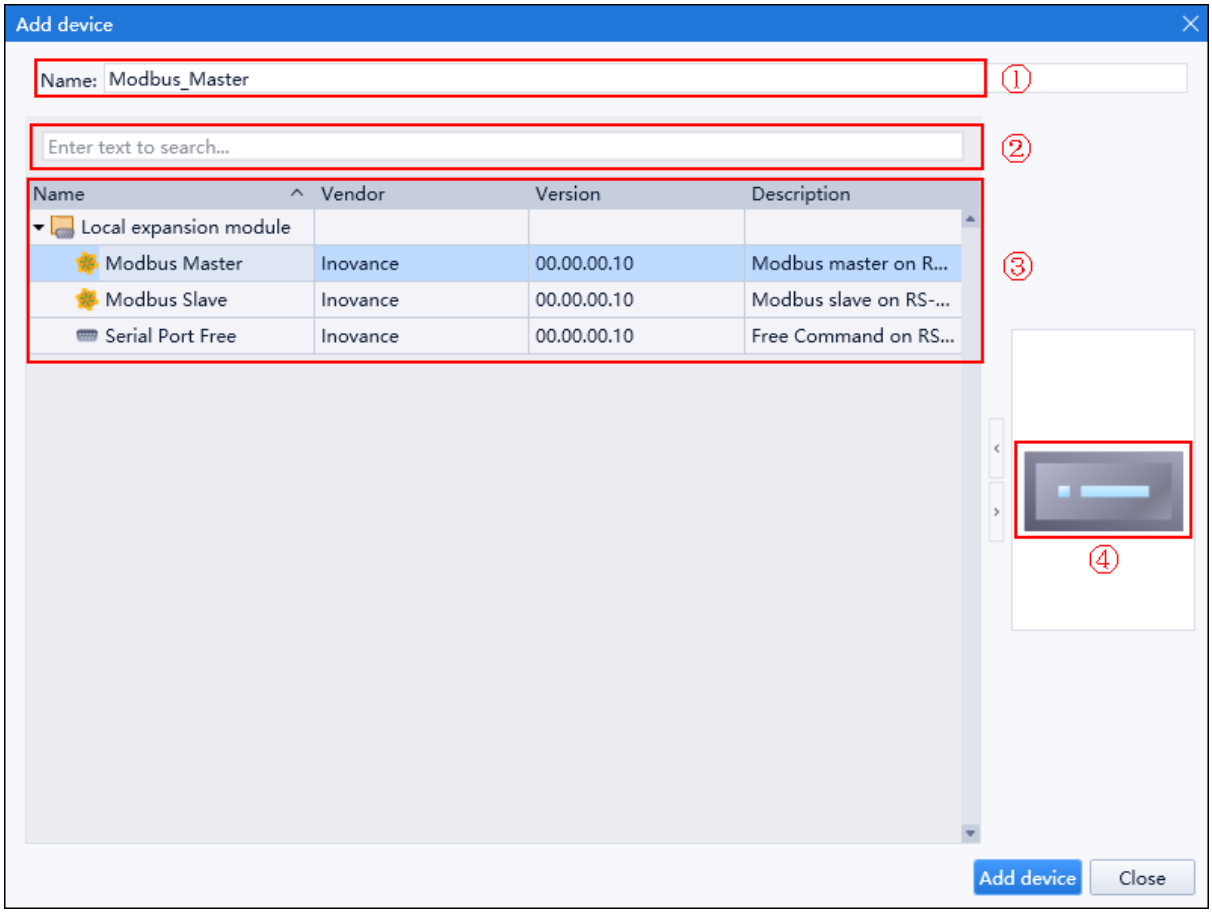
Specify the current controller as a Modbus master device to add a Modbus master. Other types of devices can be added under the master as Modbus slaves; the added slaves must match the actual hardware configuration. When the PLC goes online, the configuration information will be sent to the corresponding PLC. If the actual hardware connection sequence does not match the hardware configuration in the project, faults may occur.

Prerequisites

The master device supports serial communication.

Procedure

1. Click "Hardware configuration" in the left project tree to expand the list.
2. Right-click "COM" and select "Add device".
3. In the pop-up "Add device" window, select the master device and click "Add device" to add the Modbus master device. The "Add device" window is shown below.



No.	Description
①	Displays the name of the selected device; if no device is selected, no name is displayed.
②	A quick search box used for fast searching when there are many devices.
③	Devices supported by the current serial port.
④	Device preview image.

Result

The added Modbus master device can be viewed under "Hardware configuration > COM" in the left project tree.

Mounting a Modbus slave under the Modbus master

Mount a slave under Modbus.

Prerequisites

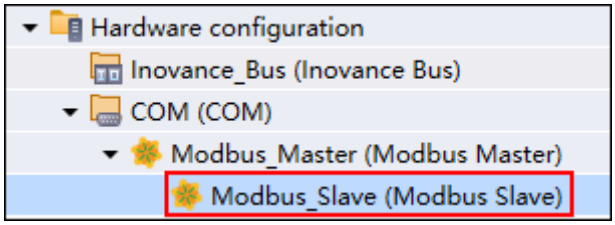
The Modbus master device has been added.

Procedure

1. Right-click "Modbus_Master" under "Hardware configuration > COM", and select "Add device".
2. In the pop-up "Add device" window, select "Modbus slave" and click "Add device", or double-click "Modbus_Slave" to add the slave device.

Result

The added Modbus slave device can be viewed under "Hardware configuration > COM > Modbus_Master".



Specifying the current device as a Modbus slave device

Prerequisites

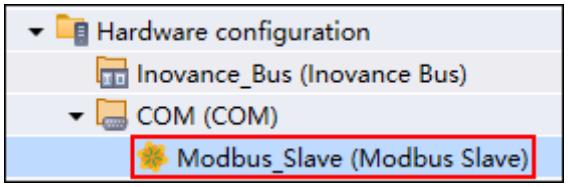
The current device supports serial communication.

Procedure

1. Right-click "COM" under the "Hardware configuration" node in the left project tree, and select "Add device".
2. In the pop-up "Add device" window, select "Modbus slave" and click "Add device", or double-click "Modbus_Slave" to add the slave device.

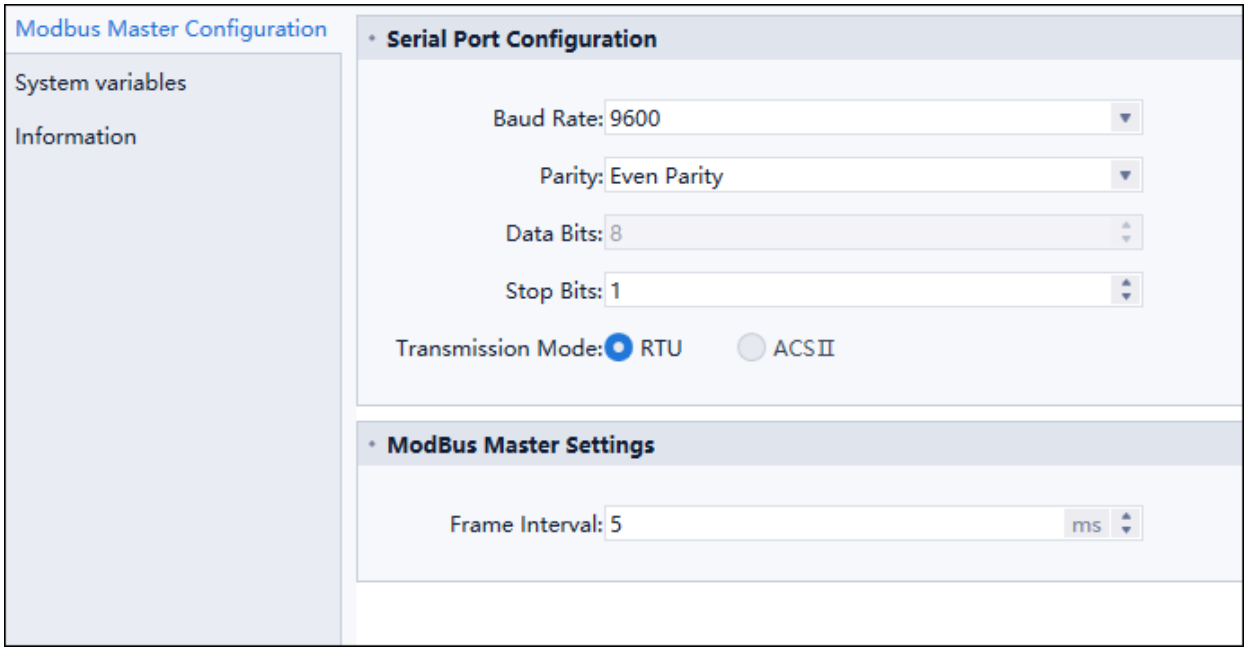
Result

The added Modbus slave device can be viewed under "Hardware configuration > COM" in the left project tree.



10.6.8.1.2 Configuring a Modbus RTU Master

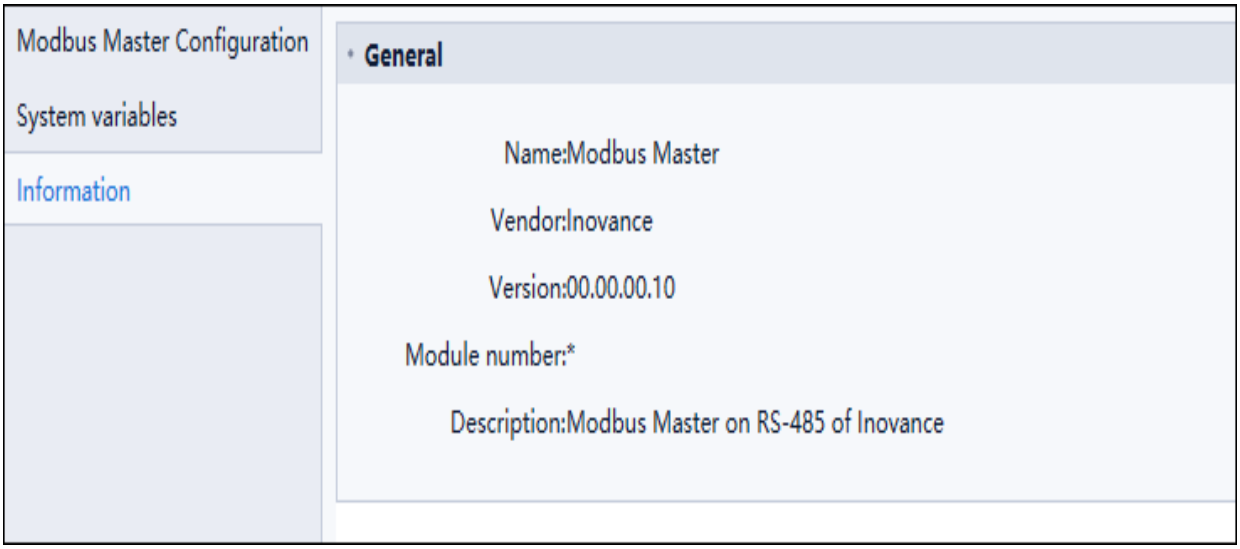
10.6.8.1.2.1 Modbus RTU Master Configuration



Item	Description
Baud rate	Communication speed
Parity check	Check method for communication frames
Data bit	Actual number of data bits contained in a communication frame
Stop bit	Last bit that identifies a single data packet during communication
Transmission mode	Binary encoding or ASCII encoding.
Frame interval	Time interval the master waits between receiving the previous response data frame and sending the next request data frame.

10.6.8.1.2.2 Information

The "Information" tab of the Modbus master configuration interface displays the following information for the current module: name, supplier, type, ID, version, module number, description, and so on.



10.6.8.1.3 Configuring a Modbus RTU Slave

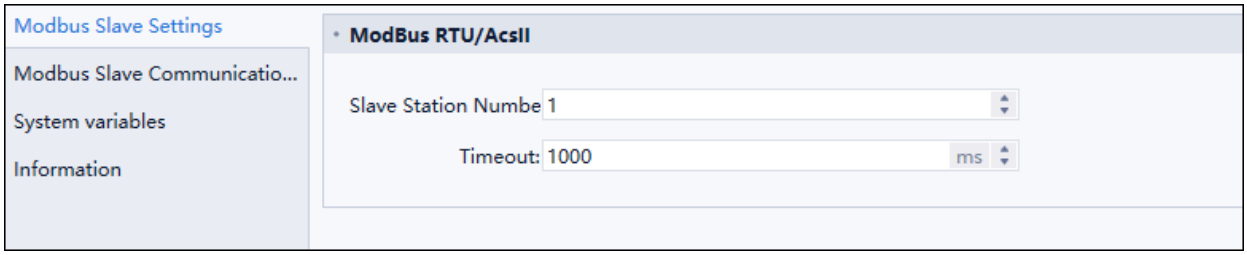
10.6.8.1.3.1 Modbus RTU Slave Configuration

Prerequisites

The master device has been added.

Procedure

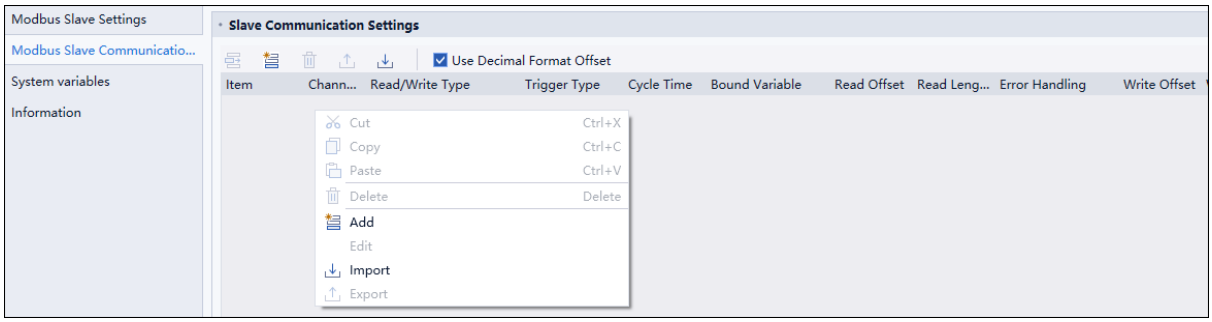
Double-click "Modbus_Slave" under "Hardware configuration > COM > Modbus_Master" in the left project tree to open the slave configuration page.



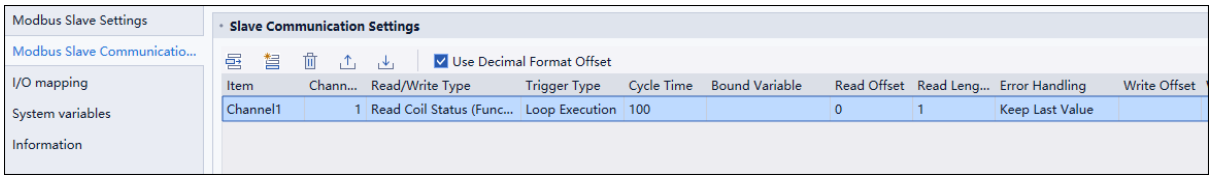
Item	Description
Slave number	Indicates the station number of the current device.
Timeout period	Indicates the response time for a single connection to the slave, with a range from 2 ms to 1000 ms and a default value of 1000 ms. If the slave does not respond within this period, the master will terminate the request after sending the specified number of retry requests and display a fault message. **Note**: The timeout time must be greater than the longest CPU scan cycle, otherwise a fault will occur.

10.6.8.1.3.2 Modbus RTU Slave Communication Configuration

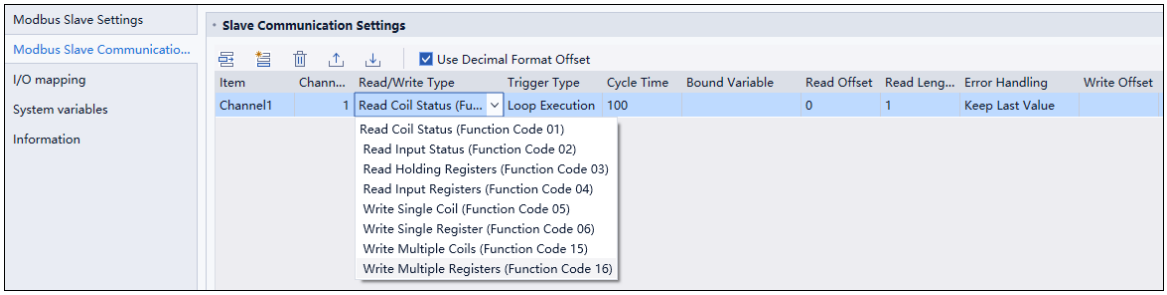
1. In the "Modbus slave communication settings" tab, right-click in the blank area on the right and select "Add" to add a communication entry. Up to 60 channels can be added.



2. The added entry is shown in the figure below.



   - Item: The name of the entry
   - Channel number: The sequence number corresponding to this entry
   - Read/Write type: Configures the function code, which can be set to read coil status (function code 01), read input status (function code 02), read holding register (function code 03), read input register (function code 04), write single coil (function code 05), write single register (function code 06), write multiple coils (function code 15), write multiple registers (function code 16).



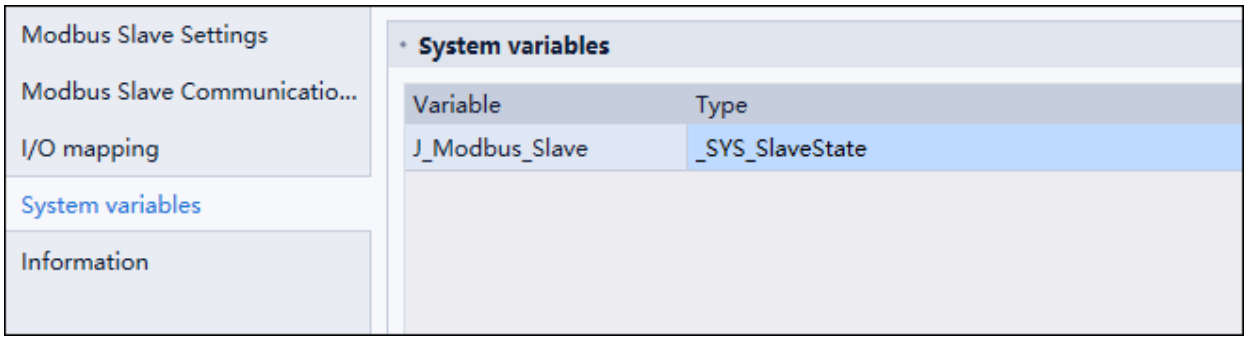
   - Trigger type: The trigger type corresponding to this entry, supporting cyclic execution and level triggering (for EVO500 series PLCs, only version V1.3 and above support level triggering).
   - Cycle time: The cycle time cannot be edited in level triggering mode.

- Bound variable: You can manually input or bind global variables in the project using the input assistant. After binding, the channel data access status can be controlled through the program. The variable type must be BOOL. Non-BOOL types are not supported. One channel can only bind one variable. After executing a trigger command, the BOOL variable value is set to FALSE. The command is effective only once and only when the variable status is TRUE.
- Read offset: The offset position based on the read start address
- Read length: The length of the read operation
- Fault handling: The handling method when a fault occurs
- Write offset: The write offset position
- Write length: The length of the write operation
- Retry count: The number of times to retransmit when encountering network issues
- Comment: Comment information

> CAUTION
>
> For details about the entries in this interface, see the Modbus protocol.

10.6.8.1.3.3 System Variables

The system variables can be configured to specify an instance (variable) of type _SYS_SlaveState for the slave I/O.



10.6.8.1.3.4 Information

The "Information" tab of the Modbus master configuration interface displays the following information for the current module: name, supplier, type, ID, version, module number, description, and so on.

10.6.8.1.4 Modbus RTU Variable Addressing

Coil

Coil is a bit variable with only two statuses, 0 and 1. This PLC (taking the EVO500 series as an example) includes variables in the Q area and I area.

Variable Name	Command Code	Starting Address	Number of coils	Note
QX0.0-QX8191.7	0x01, 0x05, 0x0f	0	65536	Accessible through the universal standard Modbus protocol
IX0.0-IX8191.7	0x2	0	65536	Accessible through the universal standard Modbus protocol

Register

Register is a 16-bit (word) variable. Take the EVO500 series as an example, the PLC includes only variables in M area.

Variable Name	Command Code	Starting Address	Number of Registers	Note
MW0-MW65535	0x03, 0x04, 0x06, 0x10	0	65536	Accessible through the universal standard Modbus protocol

NOTE

The soft elements of the EVO400/EVO500 series contain three types: Q area, I area, and M area. They can be accessed by bit, byte, word, and double word, such as: %QX, %QB, %QW, %QD. The conversion rules are as follows:

QB0 = (QX0.0 to QX0.7)
QW0 = (QB0 to QB1) =((QX0.0 to QX0.7) + (QX1.0 to QX1.7))
QD0 = (QW0 to QW1) = (QB0 to QB3) = ((QX00 to QX0.7) + (QX1.0 to X1.7) + (QX2.0 to QX2.7) + (QX3.0 to QX3.7))

Register address indexing rule

Bit Addressing	Byte Addressing	Word Addressing	Dword Addressing		Bit Addressing	Byte Addressing	Word Addressing	Dword Addressing
QX0.0	QB0	QW0	QD0		MX0.0	MB0	MW0	MD0
QX0.1					MX0.1			
QX0.2					MX0.2			
QX0.3					MX0.3			
QX0.4					MX0.4			
QX0.5					MX0.5			
QX0.6					MX0.6			
QX0.7					MX0.7			
QX1.0	QB1				MX1.0	MB1		

Bit Addressing	Byte Addressing	Word Addressing	Dword Addressing		Bit Addressing	Byte Addressing	Word Addressing	Dword Addressing
QX1.1					MX1.1			
QX1.2					MX1.2			
QX1.3					MX1.3			
QX1.4					MX1.4			
QX1.5					MX1.5			
QX1.6					MX1.6			
QX1.7					MX1.7			
QX2.0	QB2	QW1			MX2.0	MB2	MW1	
QX2.1					MX2.1			
QX2.2					MX2.2			
QX2.3					MX2.3			
QX2.4					MX2.4			
QX2.5					MX2.5			
QX2.6					MX2.6			
QX2.7					MX2.7			
QX3.0	QB3				MX3.0	MB3		
QX3.1					MX3.1			
QX3.2					MX3.2			
QX3.3					MX3.3			
QX3.4					MX3.4			
QX3.5					MX3.5			
QX3.6					MX3.6			
QX3.7					MX3.7			
QX4.0	QB4	QW2	QD1		MX4.0	MB4	MW2	MD1
QX4.1					MX4.1			

NOTE

The starting address of EVO400/EVO500's word-type register is an even byte address. The starting address of DWord-type register aligns with even word addresses, and its index number is in a 2x relationship, making address calculation convenient.

10.6.8.1.5 Modbus RTU Communication Frame Format

- Reading coils
  Use the 0x01 command code to read Q variables.

  Request frame format: slave address + 0x01 + head address of coils + number of coils + CRC code

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Slave address	1	Range: 1 to 247
2	0x01 (command code)	1	Read coils.
3	Head address of coils	2 bytes	Upper bits are followed by lower bits.
4	Number of coils (N)	2 bytes	Upper bits are followed by lower bits.
5	CRC code	2 bytes	Lower bits are followed by higher bits.

Response frame format: slave address + 0x01 + number of bytes + state of coils + CRC code

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Slave address	1	Range: 1 to 247
2	0x01 (command code)	1	Read coils.
3	Number of bytes	1	Value: [(N + 7) / 8]
4	Coil status	[(N + 7) / 8] bytes	Eight coils are indicated by one byte. If the last byte has less than eight bits, enter 0 for undefined bits. The first eight coils are indicated by the first byte, and the coil with the smallest address is indicated by the least significant bit, and so on.
5	CRC code	2 bytes	Lower bits are followed by higher bits.

- Reading registers
  Use the 0x03 command code to read M variables.

  Request frame format: slave address + 0x03 + head address of registers + number of registers + CRC code

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Slave address	1	Range: 1 to 247

No.	Definition of Data (Byte)	Number of Bytes	Note
2	0x03 (command code)	1	Read registers.
3	Head address of registers	2 bytes	Upper bits are followed by lower bits.
4	Number of registers (N)	2 bytes	Upper bits are followed by lower bits.
5	CRC code	2 bytes	Lower bits are followed by higher bits.

Response frame format: slave address + 0x03 + number of bytes + register value + CRC code

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Slave address	1	Range: 1 to 247
2	0x03 (command code)	1	Read registers.
3	Number of bytes	1	Value: N*2
4	Register value	N x 2	One register value is indicated by two bytes. Upper bits are followed by lower bits. The register with a smaller address is indicated by the first byte.
5	CRC code	2 bytes	Lower bits are followed by higher bits.

- Writing a single coil
  Use the 0x05 command code to write Q variables.

  Request frame format: slave address + 0x05 + head address of the coil + state of the coil + CRC code

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Slave address	1	Range: 1 to 247
2	0x05 (command code)	1	Writes a single coil.
3	Address of the coil	2 bytes	Upper bits are followed by lower bits.
4	Coil status	2 bytes	Lower bits are followed by higher bits. A non-zero value is valid.
5	CRC code	2 bytes	Lower bits are followed by higher bits.

Response frame format: slave address + 0x05 + head address of the coil + state of the coil + CRC code

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Slave address	1	Range: 1 to 247
2	0x05 (command code)	1	Writes a single coil.
3	Address of the coil	2 bytes	Upper bits are followed by lower bits.
4	Coil status	2 bytes	Lower bits are followed by higher bits. A non-zero value is valid.

No.	Definition of Data (Byte)	Number of Bytes	Note
5	CRC code	2 bytes	Lower bits are followed by higher bits.

- Writes one register.
  Use the 0x06 command code to write M variables.

  Request frame format: slave address + 0x06 + head address of the register + register value + CRC code

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Slave address	1	Range: 1 to 247
2	0x06 (command code)	1	Writes a single register.
3	Register address	2 bytes	Upper bits are followed by lower bits.
4	Register value	2 bytes	Upper bits are followed by lower bits. A non-zero value is valid.
5	CRC code	2 bytes	Lower bits are followed by higher bits.

  Response frame format: slave address + 0x06 + head address of the register + register value + CRC code

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Slave address	1	Range: 1 to 247
2	0x06 (command code)	1	Writes a single register.
3	Register address	2 bytes	Upper bits are followed by lower bits.
4	Register value	2 bytes	Upper bits are followed by lower bits. A non-zero value is valid.
5	CRC code	2 bytes	Lower bits are followed by higher bits.

- Writing multiple coils
  Use the 0x0f command code to write multiple consecutive Q variables.

  Request frame format: slave address + 0x0f + head address of coils + number of coils + number of bytes + state of coils + CRC code

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Slave address	1	Range: 1 to 247
2	0x0f (command code)	1	Writes multiple single coils.
3	Head address of coils	2 bytes	Upper bits are followed by lower bits.
4	Number of coils (N)	2 bytes	Upper bits are followed by lower bits, with a maximum value of 1968.
5	Number of bytes	1	Value: [(N + 7) / 8]

No.	Definition of Data (Byte)	Number of Bytes	Note
6	Coil status	[(N + 7) / 8] bytes	Eight coils are indicated by one byte. If the last byte has less than eight bits, enter 0 for undefined bits. The first eight coils are indicated by the first byte, and the coil with the smallest address is indicated by the least significant bit, and so on.
7	CRC code	2 bytes	Lower bits are followed by higher bits.

Response frame format: slave address + 0x0f + head address of coils + number of coils + CRC code

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Slave address	1	Range: 1 to 247
2	0x0f (command code)	1	Writes multiple single coils.
3	Head address of coils	2 bytes	Upper bits are followed by lower bits.
4	Number of coils	2 bytes	Upper bits are followed by lower bits.
5	CRC code	2 bytes	Lower bits are followed by higher bits.

- Writing multiple registers

  Use the 0x10 command code to write multiple consecutive M variables.

  Request frame format: slave address + 0x10 + head address of registers + number of registers + number of bytes + register value + CRC code

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Slave address	1	Range: 1 to 247
2	0x10 (command code)	1	Writes multiple registers.
3	Head address of registers	2 bytes	Upper bits are followed by lower bits.
4	Number of registers (N)	2 bytes	Upper bits are followed by lower bits.
5	Number of bytes	1	Value: N*2
6	Register value	N*2 (N*4)	-
7	CRC code	2 bytes	Lower bits are followed by higher bits.

Response frame format: slave address + 0x10 + head address of coils + number of coils + CRC code

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Slave address	1	Range: 1 to 247
2	0x10 (command code)	1	Writes multiple registers.

No.	Definition of Data (Byte)	Number of Bytes	Note
3	Head address of registers	2 bytes	Upper bits are followed by lower bits.
4	Number of registers (N)	2 bytes	Upper bits are followed by lower bits.
5	CRC code	2 bytes	Lower bits are followed by higher bits.

10.6.8.1.6 Common Modbus RTU Faults

The main faults that occur when a Modbus master connects to a slave are as follows:

- The Modbus master configuration is inconsistent with the Modbus slave configuration, preventing the establishment of communication between the master and slave.
- The Modbus master accesses an illegal address on the Modbus slave, resulting in a faulty response.
- The Modbus master attempts to write to a register on the Modbus slave, but the register on the slave only supports reading and not writing, causing the master to receive a faulty response from the slave.

Fault response frame format

Fault response frame format: slave address + (command code + 0x80) + fault code + CRC code

This fault frame applies to all operation command frames.

No.	Definition of Data (Byte)	Number of Bytes	Note
1	Slave address	1	Value range: 1 to 247
2	Command code + 0x80	1	Fault command code
3	Fault code	1	1 to 4
4	CRC code	2 bytes	Lower bits are followed by higher bits.

10.6.8.2 Free Serial Communication

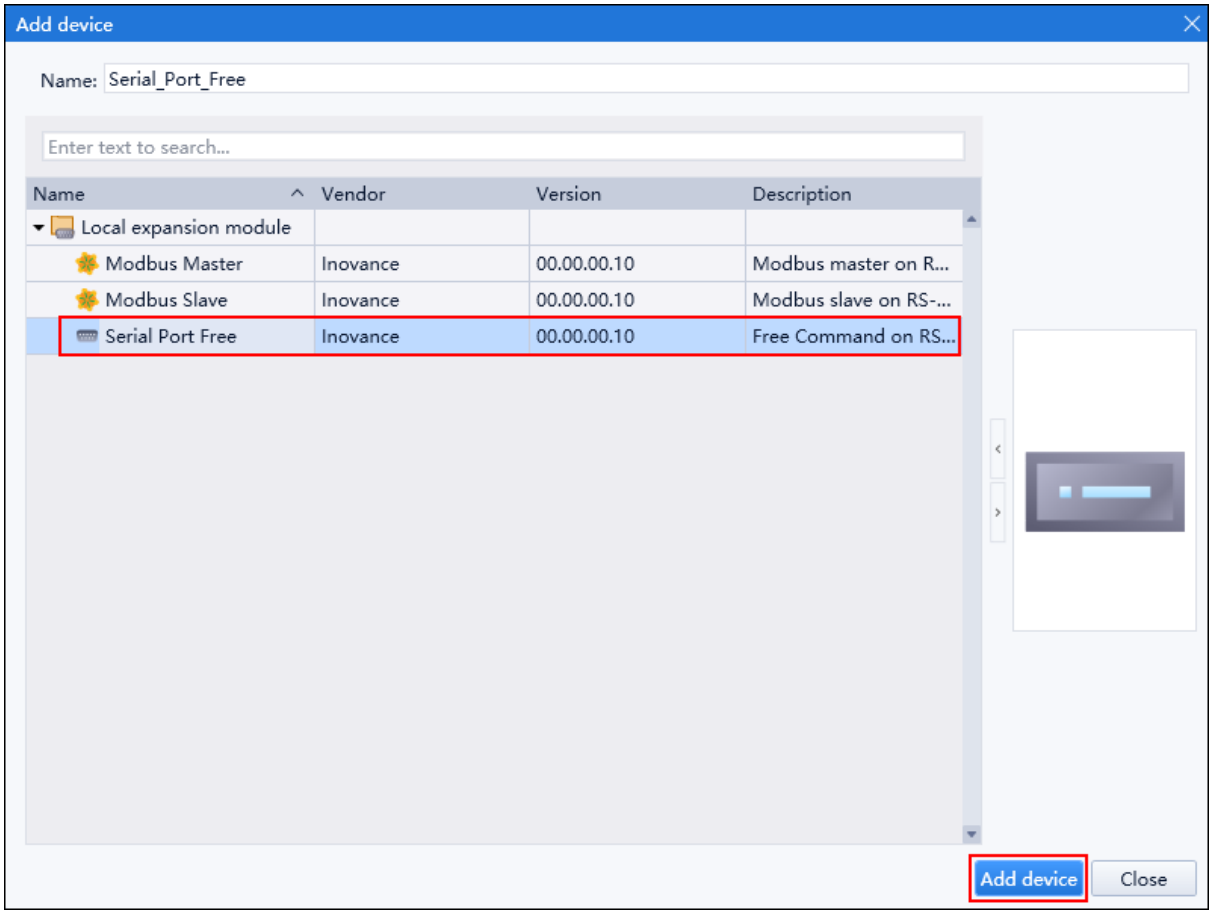
10.6.8.2.1 Adding Serial_Port_Free

Prerequisites

The selected device supports serial communication.

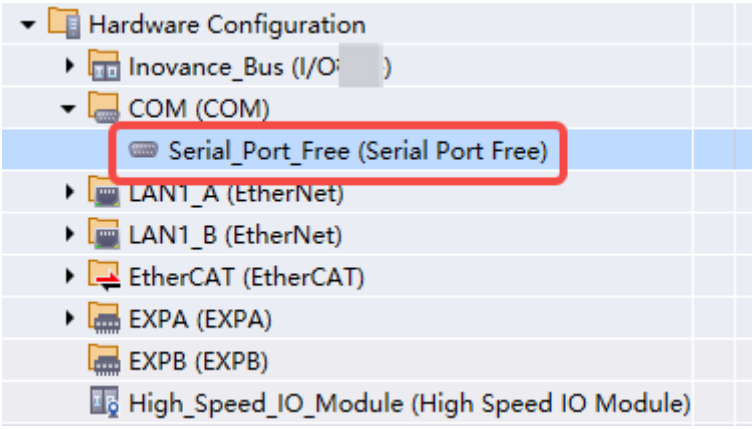
Procedure

1. Click "Hardware configuration" in the left project tree to expand the list.
2. Right-click "COM" and select "Add device".
3. The "Add device" window pops up. Double-click the "Serial port free" entry, or select it and click "Add device".



Result

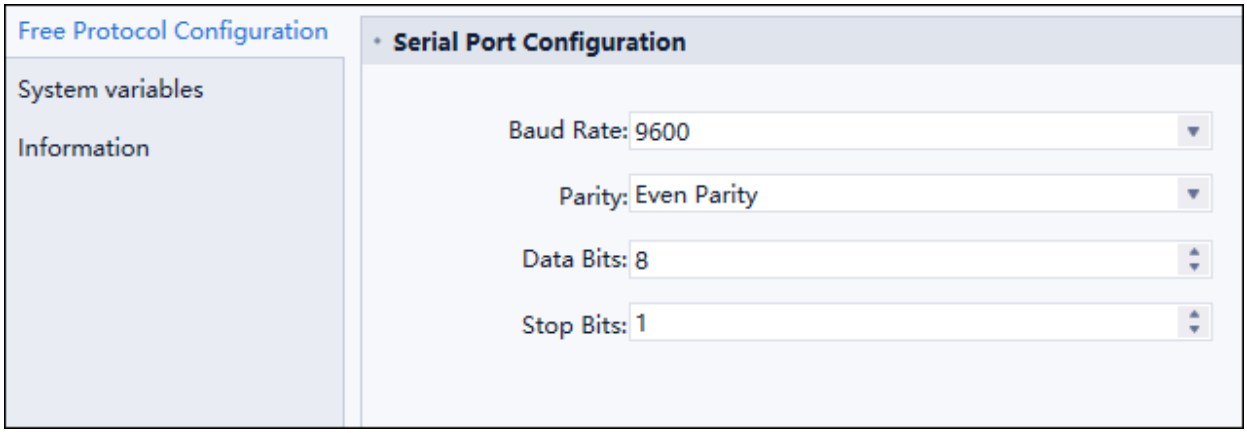
The added free serial port can be viewed in the project tree.



10.6.8.2.2 Configuring Serial_Port_Free

The corresponding free serial port has been added in the hardware configuration.

Double-click the added free serial port to open the serial port configuration interface. Commonly used configuration options are displayed in the configuration interface.

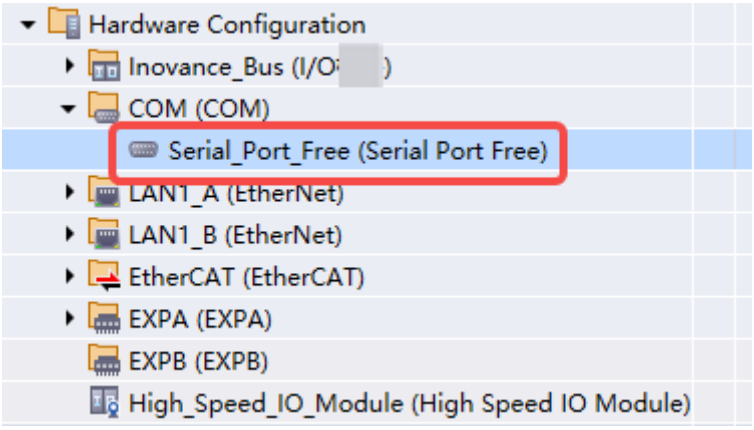


Item	Description
Baud rate	Communication speed
Parity check	Check method for communication frames
Data bit	Actual number of data bits contained in a communication frame
Stop bit	Last bit that identifies a single data packet during communication

10.6.8.2.3 Transmitting/Receiving Data Through Free Serial Port

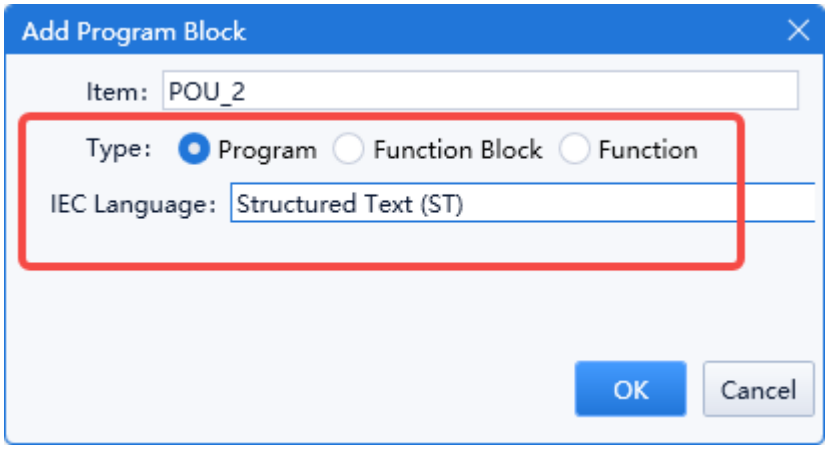
Prerequisites

The corresponding free serial port has been added and configured.

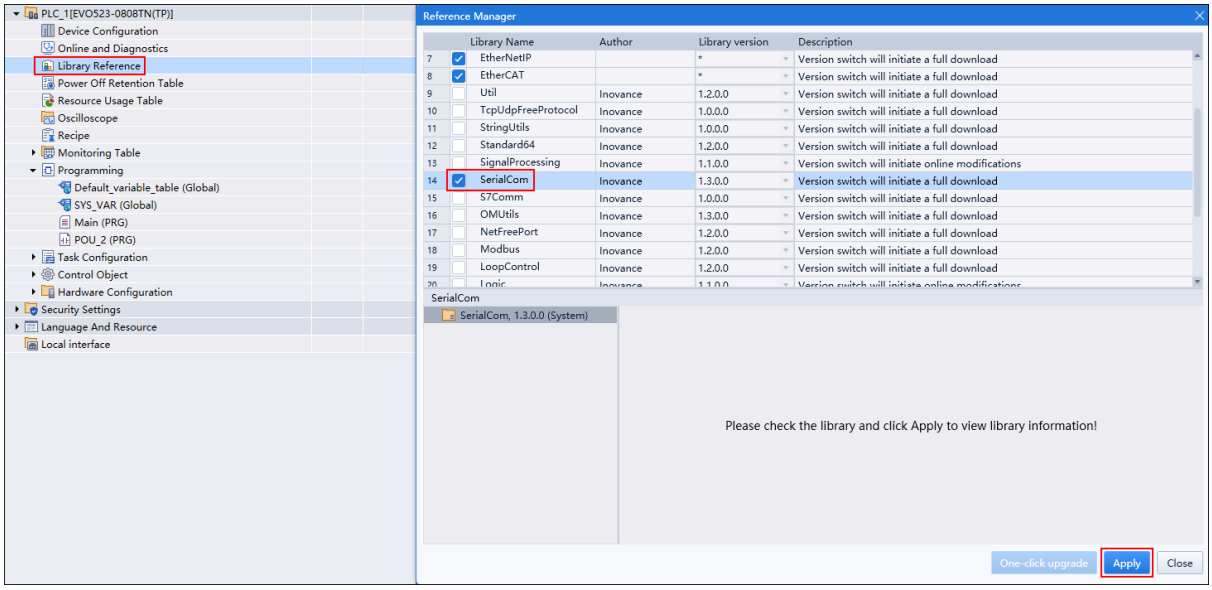


Procedure

1. Create a PLC programming task. Under the "Programming" node in the project tree, add a program block, select "Program" as the programming type, and set the IEC language to "Structured text (ST)".



2. Add the corresponding library reference. Double-click the "Library reference" node in the project tree. In the pop-up "Reference manager" interface, and select the "SerialCom" library.



3. The added library can be viewed in the "Toolbox" on the right side of the interface.

4. Call these functions to implement data communication. For detailed function descriptions, see 18.4.7 Free Serial Port.

10.6.8.3 Disabling/Enabling a COM Device

Disabling the device under the COM node

In iFA software, a certain COM device can be disabled through commands or variable setting, thus "detaching" from the control system while retaining its configuration parameters and I/O mapping information. If a device fails and needs to be replaced, this function can reduce the time required for replacement.

- Disabling operation and effect
  A device can be disabled through commands or variable setting, each with different scopes. Command disabling applies to all devices except PLCs Variable setting requires that there be corresponding variables in SYS_VAR.
  - Command disabling
    Right-click a device under the "COM" node in the project tree and select "Disable device".
  - Variable setting
    When the current PLC is set as a Modbus RTU Master, the added remote slave devices will have corresponding variables created in **SYS_VAR**, with variable naming rule: J_device model.
    Setting member The member **bDisableEnable** disables the slave, and the setting method supports writing values in **SYS_VAR** or assigning values in user programs.

	Name	Data type	Address	Retain	Network publish	Comment
1	VAR_GLOBAL					
2	LAN1_EIPSingle_STATUS	Array[1..32] OF IoStand...			Private	
3	LAN1_EIPSlave_STATUS	Array[1..16] OF EtherNe...			Private	
4	J_Modbus_Slave	IoStandard._SYS_SlaveS...			Private	
5	bDisableEnable	BOOL			Private	

    After disabling, it is necessary to perform 10.8.3 a full download of the PLC to make the disabling operation effective.

- Disabling result
  The disabled devices are marked with the icon for differentiation.



Enabling the device under the COM node

The disabling and enabling of COM devices are a set of mutually corresponding operations. Only when the device is disabled is the enabling command visible.

The enabling operation and its effect are exactly the same as the disabling operation.

A device needs to be disabled and enabled in the same way. That is, if the device is disabled by setting a variable, it must be enabled by resetting the variable. If it is disabled using a command, it must also be enabled using a command.

10.6.9 FINS TCP&UDP Communication Configuration

10.6.9.1 Overview

FINS (Factory Interface Network Service) is an industrial automation control network protocol used for communication between clients and servers. FINS is an application-layer protocol. When using Ethernet communication, data is sent and received as UDP/IP or TCP/IP packets over the default port 9600.

The EVO500, EVO700, EVO760C, and EVO780C series PLCs integrate FINS protocol communication and support read/write access to the M memory area. Currently, these PLCs only act as FINS TCP/UDP servers and can exchange data with Omron, Keyence, and other PLC brands.

The FINS server function is disabled by default. The user must manually call the FINS_Server function block to enable or disable FINS communication. After the PLC stops running, FINS communication will no longer be accessible.

10.6.9.2 FINS Parameters and Addresses

Parameters

When accessing the I/O memory area (M area) of EVO500 and EVO700 series PLCs via FINS protocol commands, the communication command codes are shown in the table below.

Table10-6-9-2-1 FINS communication command code

Type	Primary Command	Secondary Command	Function	Description
I/O memory area (M area) read/write	01	01	Memory reading	Reads data from consecutive I/O memory area.
	01	02	Memory writing	Writes data to consecutive I/O memory area.

Address

The PLC soft components are mapped to the M area (fixed code: 0x82), with an addressing range of 0 to 32767. Access is performed in word components. The table below shows the soft component address mapping supported by the FINS protocol.

Variable	Address Range	Number of Soft Elements	Note
%MW0 to %MW32767	0x0000 to 0x7FFF (0 to 32767)	32768	16-bit register

NOTE

When reading/writing a large number of devices, it is recommended that the FINS TCP/UDP client split the data read/write into multiple packets.

10.6.9.3 FINS Protocol Configuration

Prerequisites

- The default communication port 9600 cannot be modified.
- The maximum number of FINS TCP/UDP client connections is 32.
- The maximum number of network nodes for the FINS protocol is 254.
- The maximum number of consecutive components that can be read via the FINS protocol is 2000 words.
- The last byte of the IP address must be within the range of 1 to 254 and cannot be set to 0 or 255.

Command	RD-CMD (Read Command)	RD-ACK (Read Response)	WR-CMD (Write Command)	WR-ACK (Write Response)
FINS TCP	34 bytes	30 bytes + 2*N	34 bytes + 2*N	30 bytes
FINS UDP	18 bytes	14 bytes + 2*N	18 bytes + 2*N	14 bytes

NOTE

N does not exceed 2000.

10.6.10 DP Communication Configuration (EVO800)

10.6.10.1 DP Master Configuration (SYCON.net)

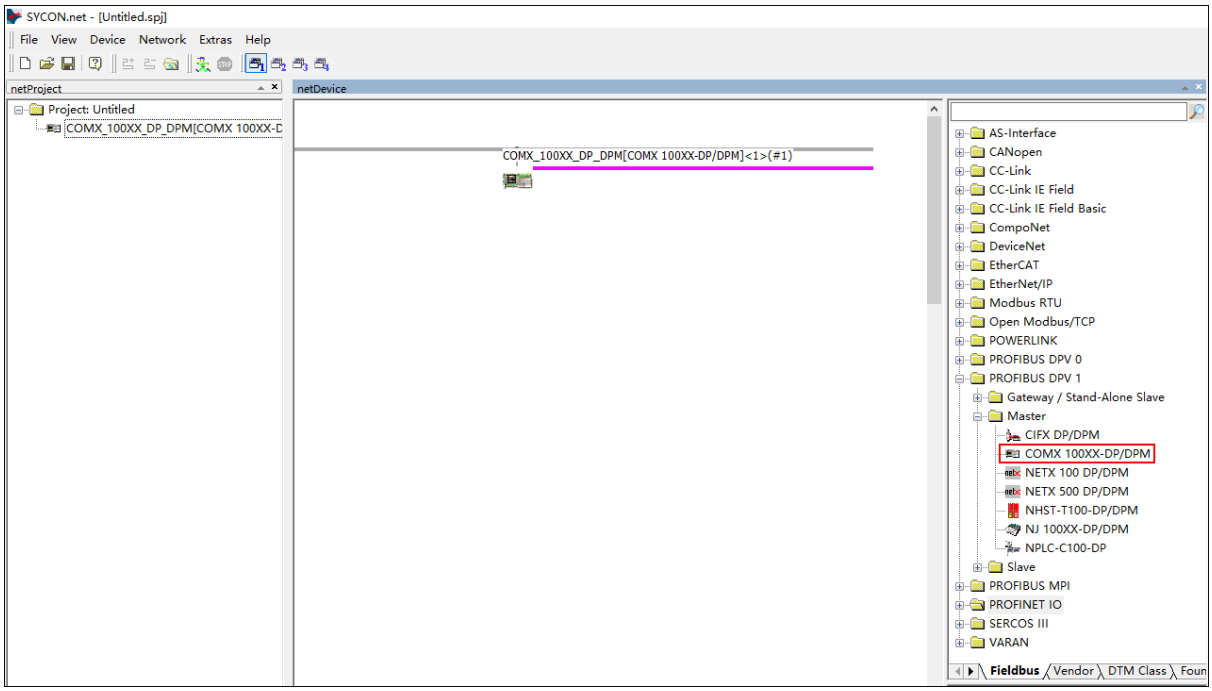
This section introduces how to configure the DP master and its slaves using the SYCON.net software.

Prerequisites

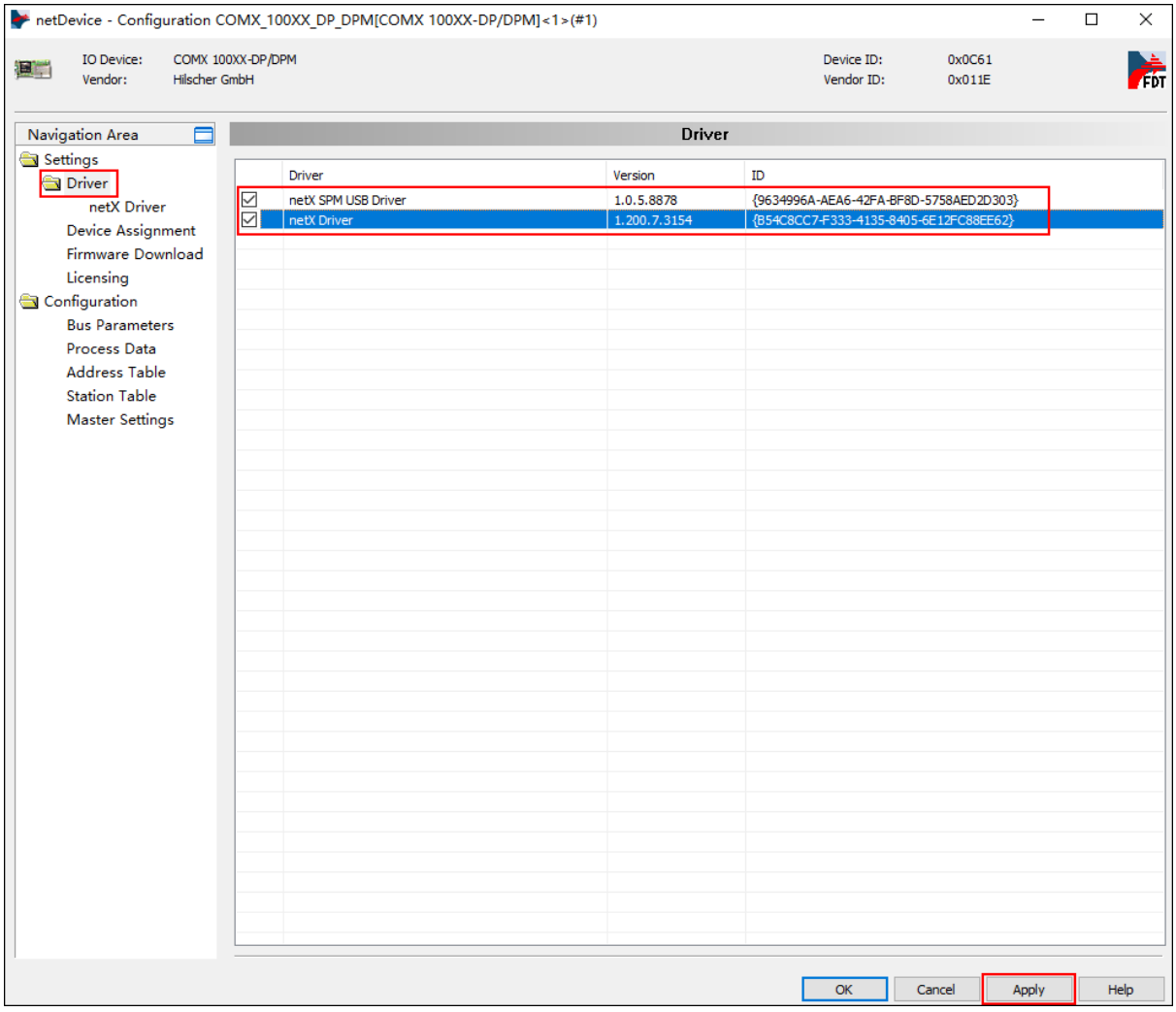
- SYCON.net software has been installed.
- The slave device description file has been obtained.
- Hardware connection has been completed.
  The DP can connect to the PC via the commissioning interface. The default IP of the commissioning interface is 192.168.1.201 (ensure that the local network is configured in the 192.168.1.XX subnet).

DP master configuration

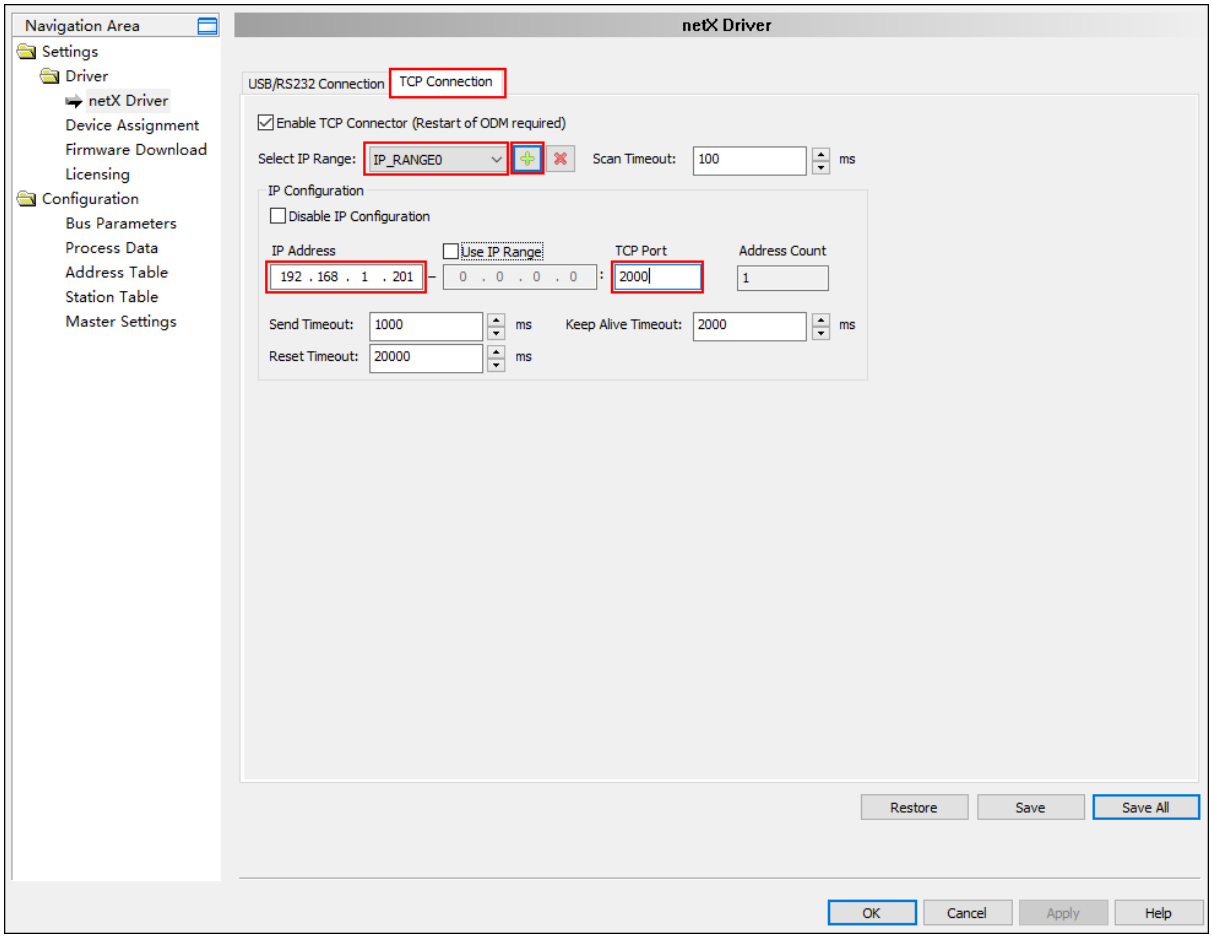
1. Launch the SYCON.net software as the admin.
2. Add a DP master. On the "Fieldbus" tab on the right of the interface, select "PROFIBUS DPV 1 > Master > COMX 100XX-DP/DPM" and drag it into the "netDevice" window, connecting it to the gray data line.



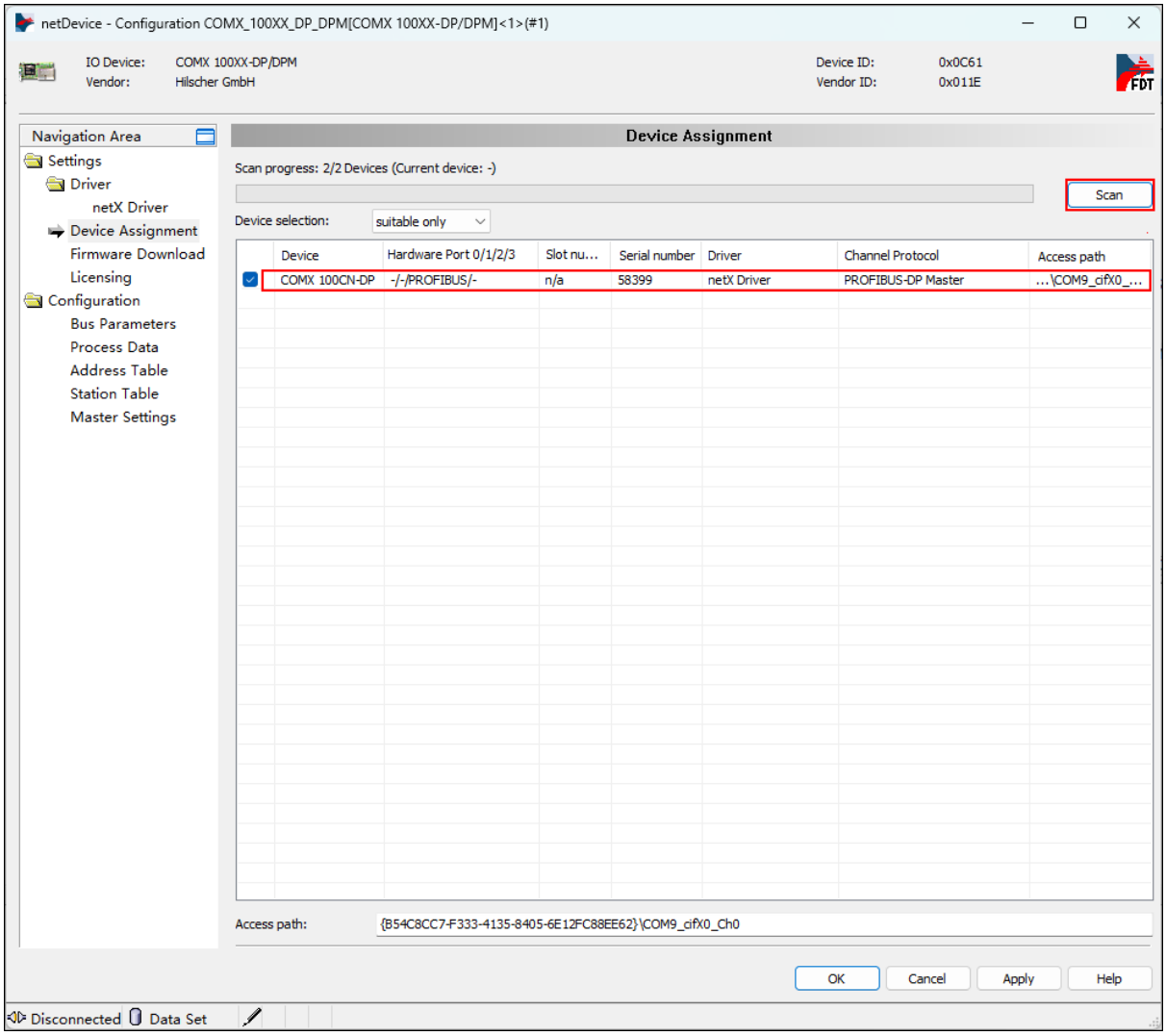
3. After adding the master, double-click the master icon to open the configuration page. On the "Driver" tab, check the configuration items as shown below, and click "Apply".



4. When connecting to the master via Ethernet, select "Driver > netXDriver" on the configuration page. On the "TCP Connection" tab, configure the following parameters: Click in the "Select IP Range" area,  set the IP range to "IP_RANGE0"; set "IP Address" to "192.168.1.201"; set "TCP Port" to "2000".



5. Select the "Device Assignment" tab, click "Scan". Once the DP module is found, click "Apply", then click "OK".



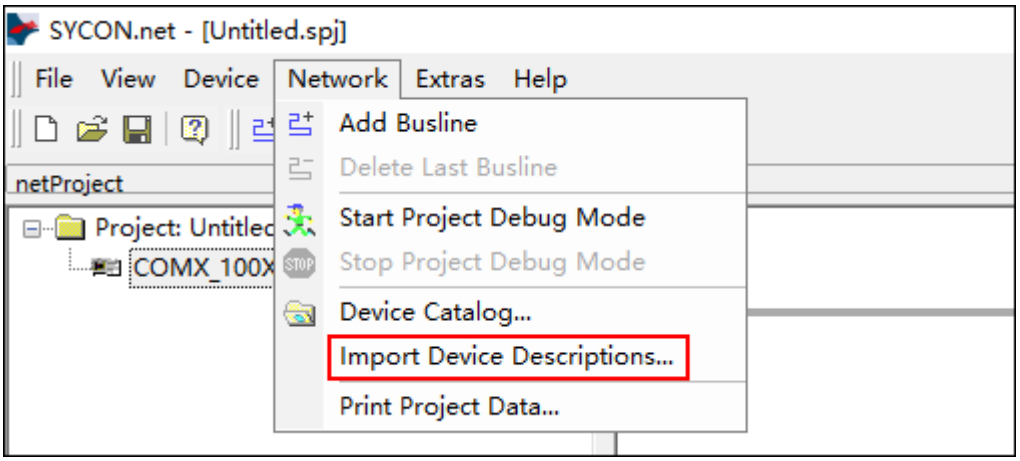
DP slave configuration

Using the addition of the IM155-5 DP ST V3.0 slave as an example.

1. Import the slave device description file.

   In the menu bar, select "NetWork > Import Device Description" to import the slave device description file.

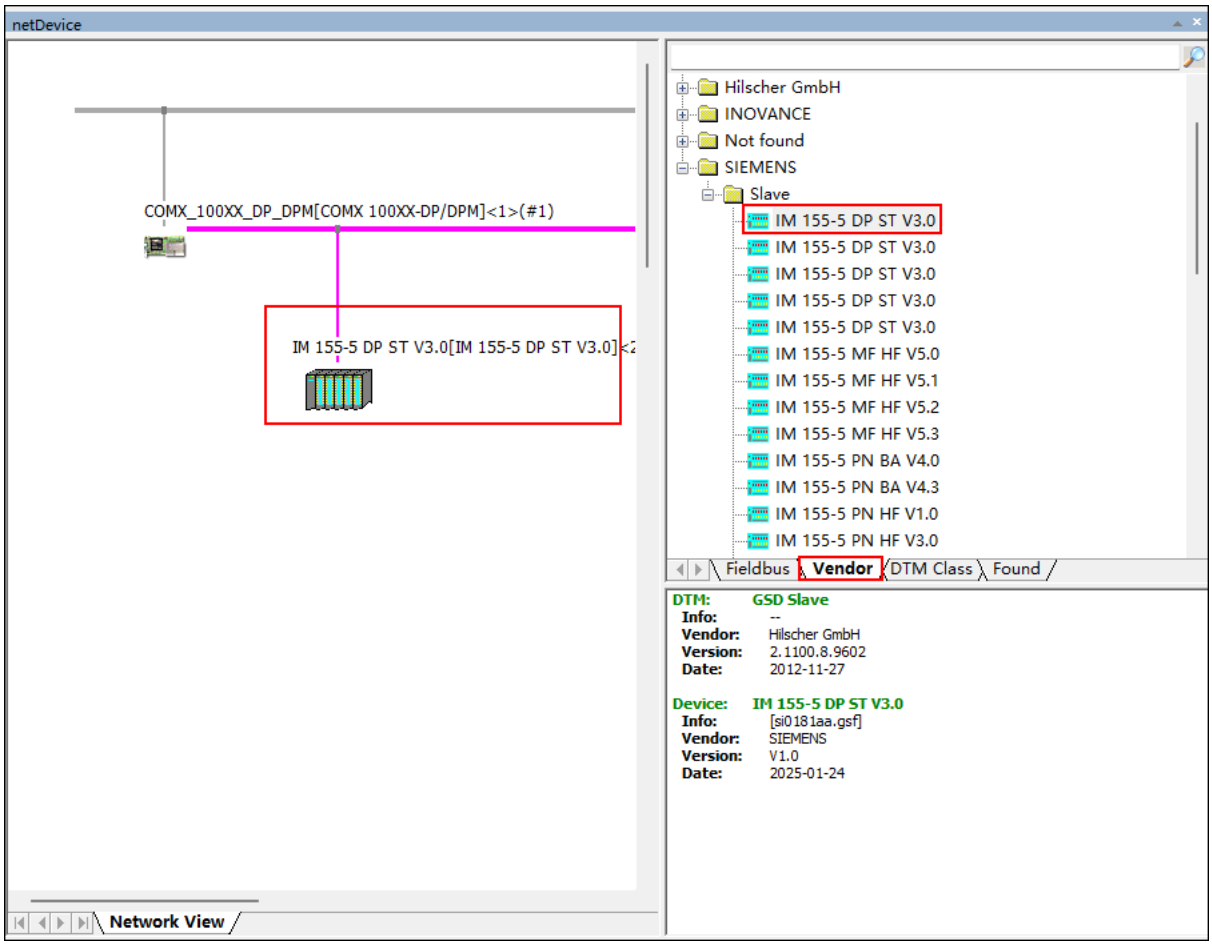
   You can view the added device on the "Vendor" tab on the right of the interface.



2. Add a slave.

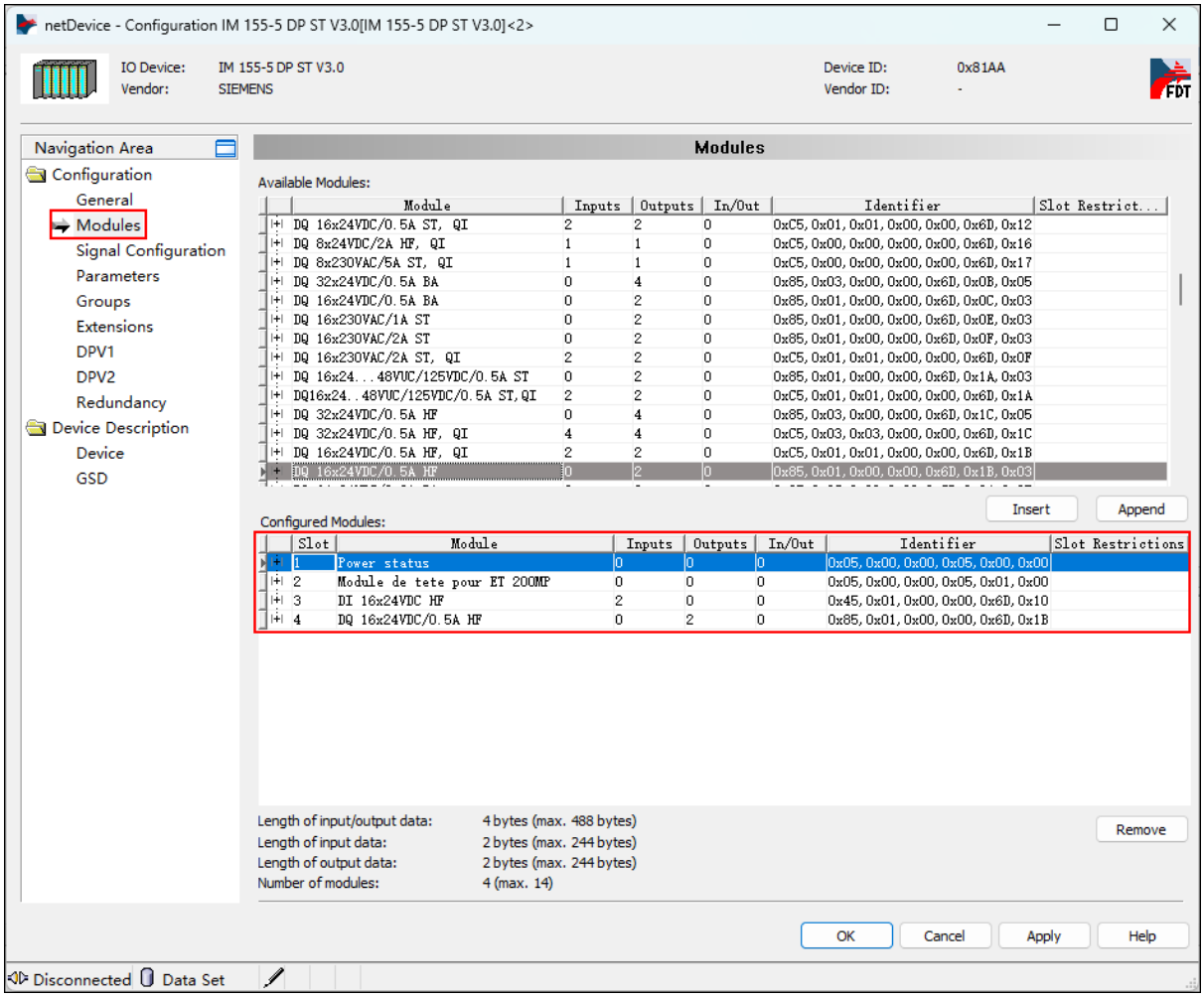
   On the "Vendor" tab on the right of the interface, select "SIEMENS > Slave > IM155-5 DP ST V3.0", and drag

it under the master.



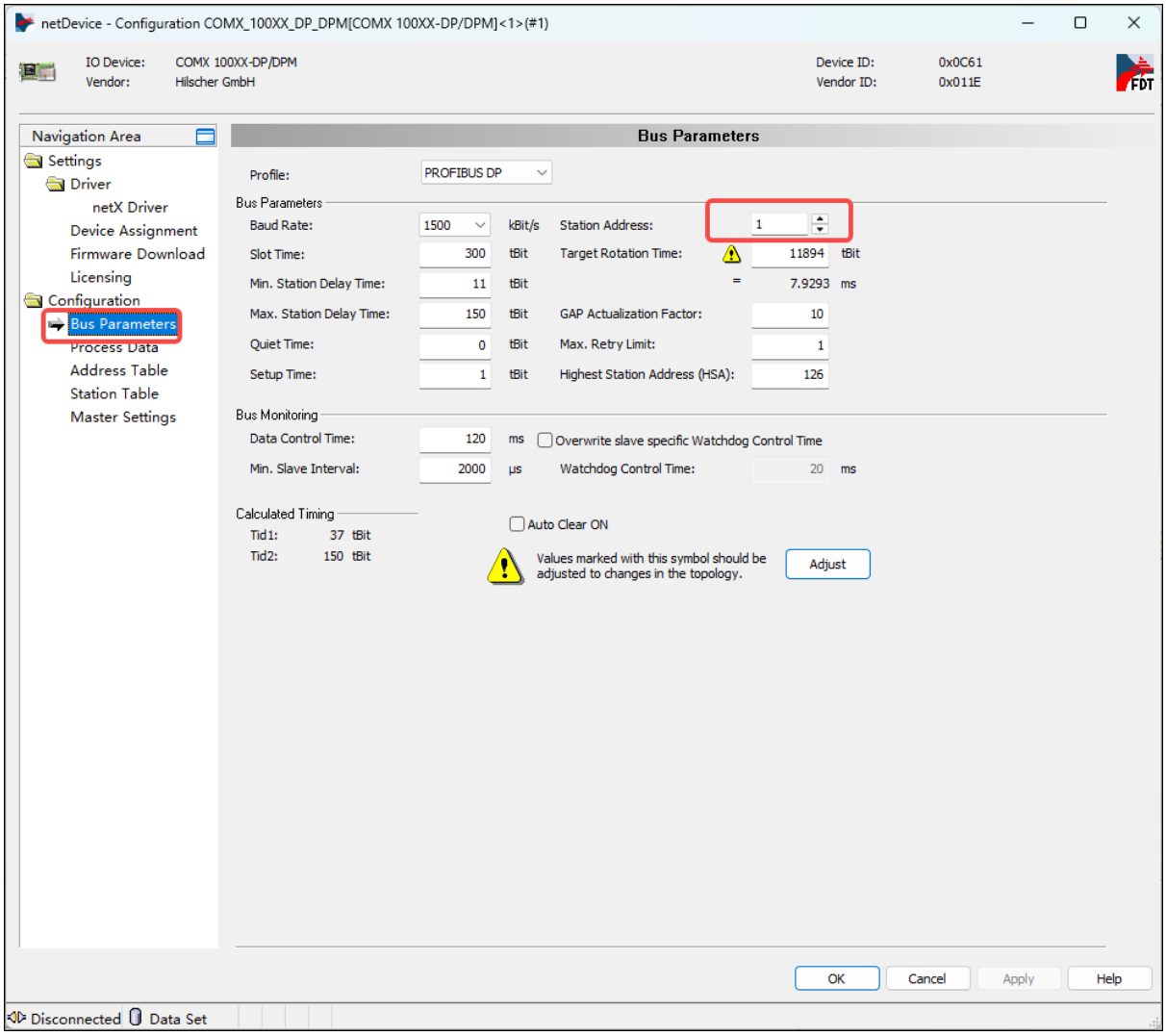
3. Configure the slave.

Double-click the slave icon to open the configuration page. On the "Modules" tab, double-click the corresponding slave model to add it, and click "Apply".



Master/Slave address configuration

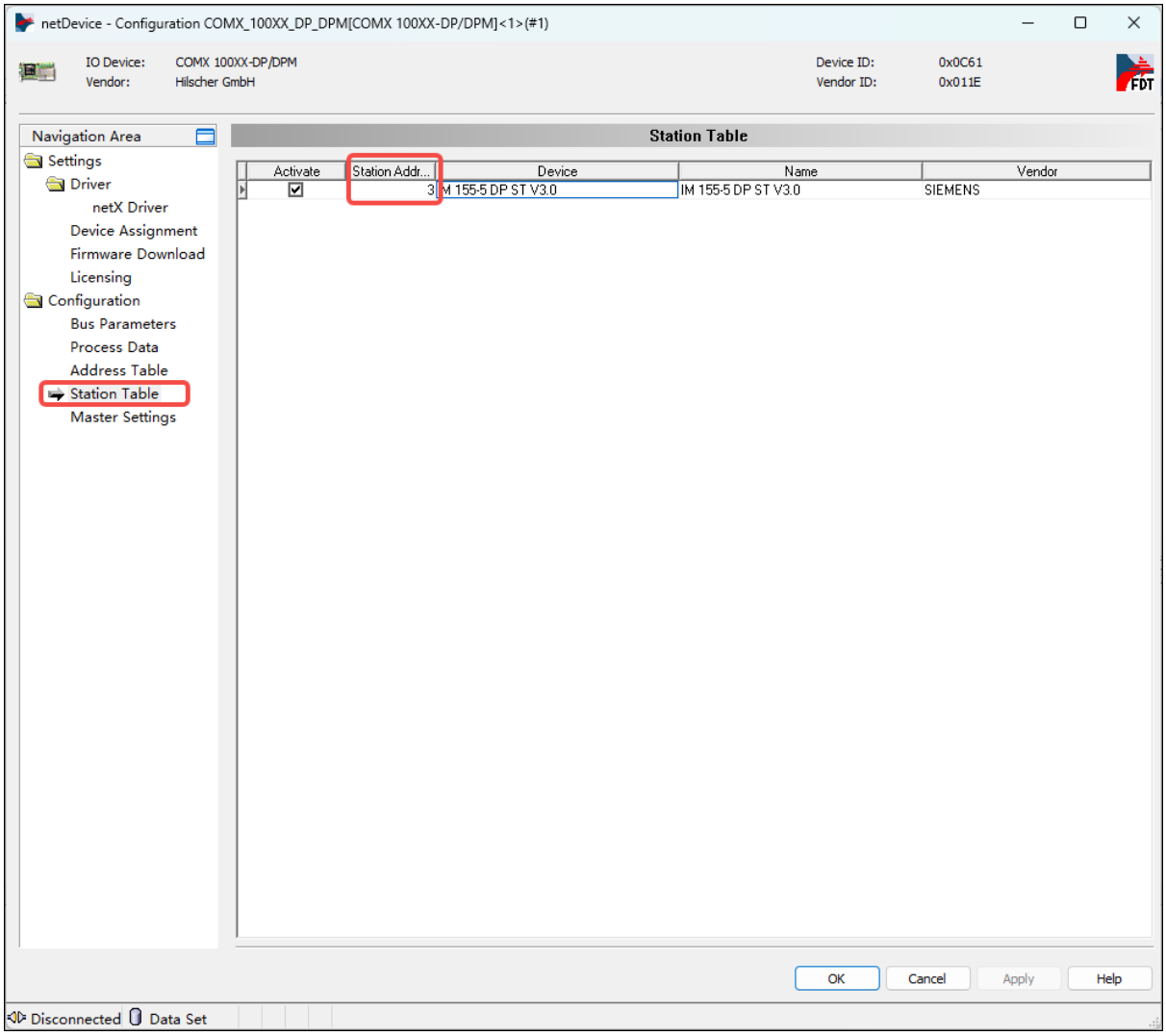
1. Double-click the master icon to open the master configuration page. Select "Bus Parameters"; the default station address of the master is 1.



2. Select "Station Table", change the slave address to 3, click "Apply", then click "OK".

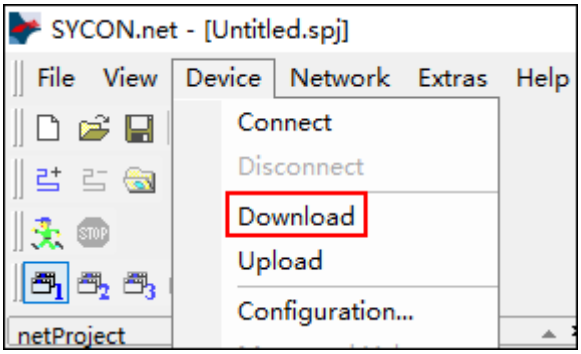
NOTE

The slave address setting shall match the DIP switch setting of the actual connected slave.

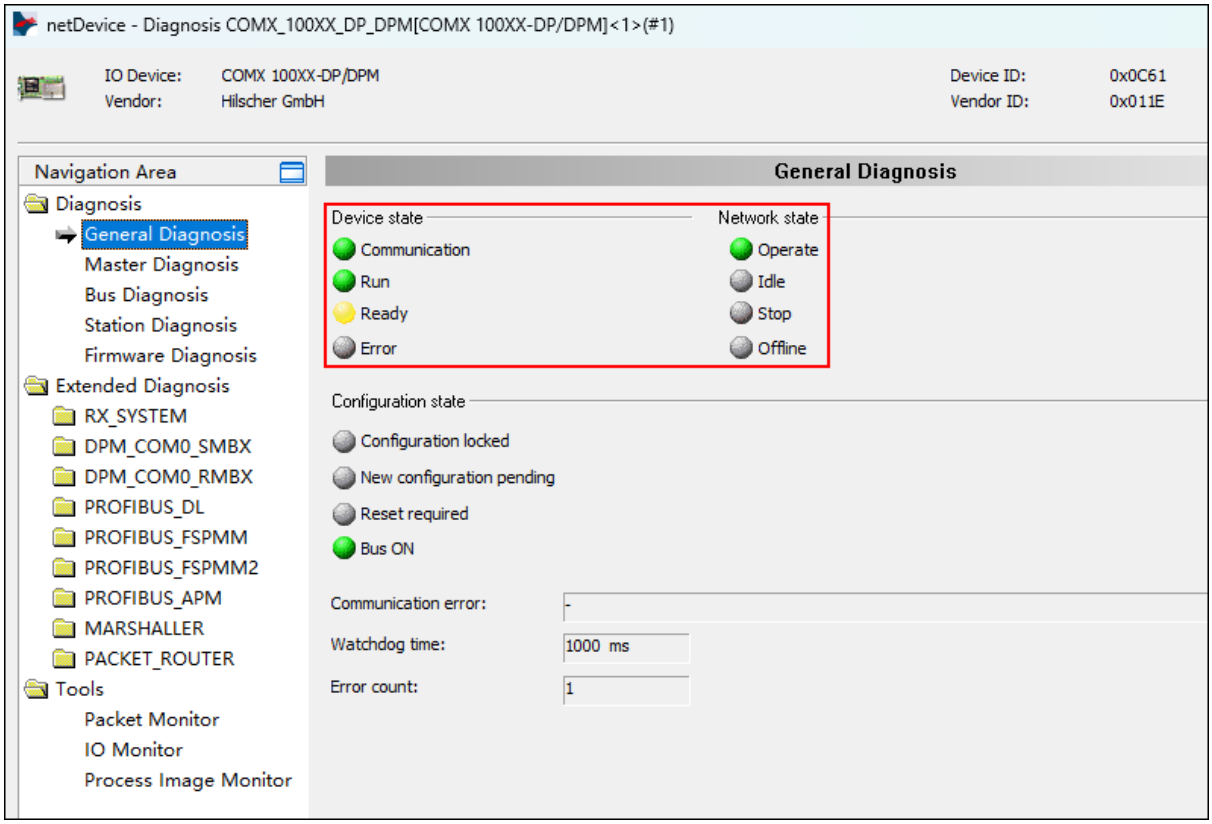


Configuration download

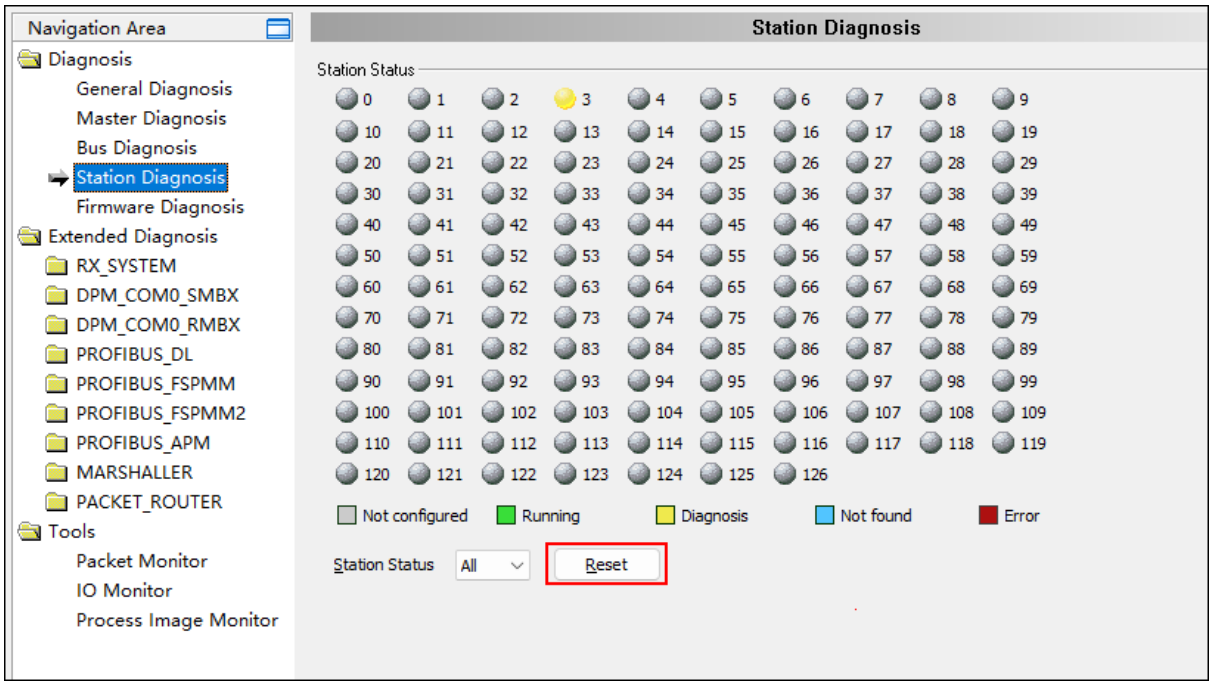
1. Select "Device > Download" from the menu bar to download the configuration to the device.

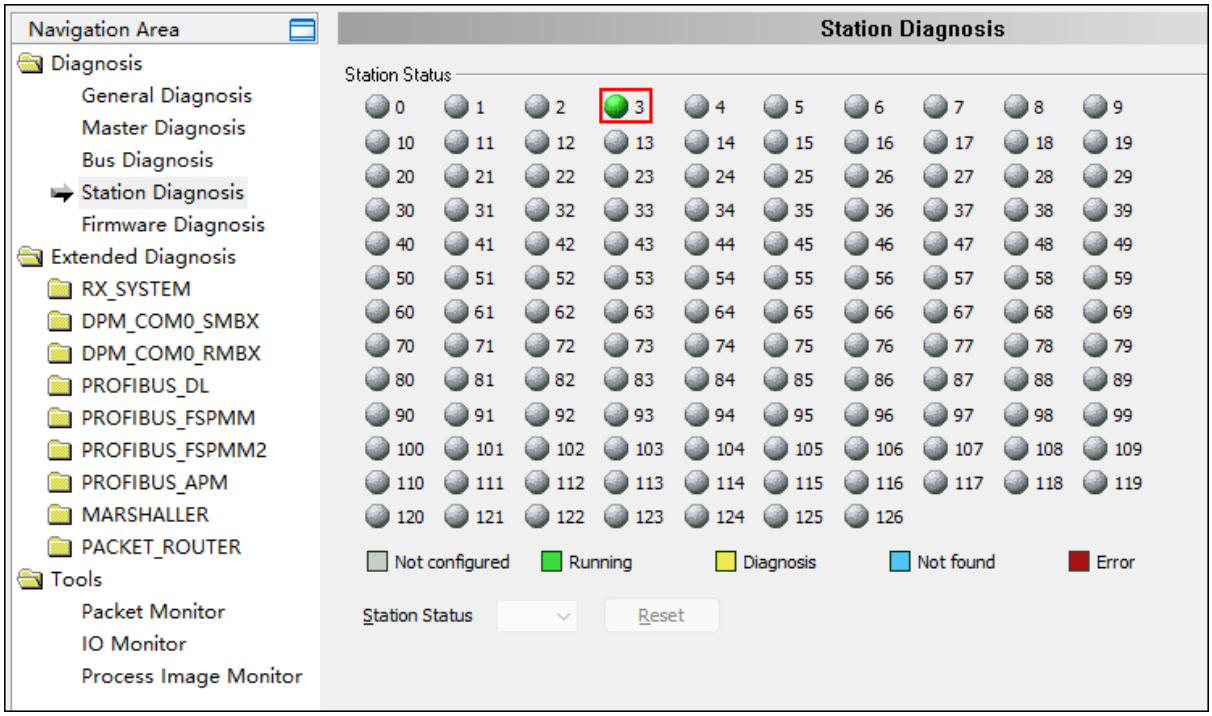


2. Check whether the configuration and communication are successful.
   Double-click the master icon to view diagnostic information on the "General Diagnosis" tab. As shown below, the master is operating normally.



3. Select "Station Diagnosis" to check the slave operation. A yellow light indicates that the slave has reported diagnostic information to the master. Click "Reset" and the light turns green, indicating normal communication and successful configuration.





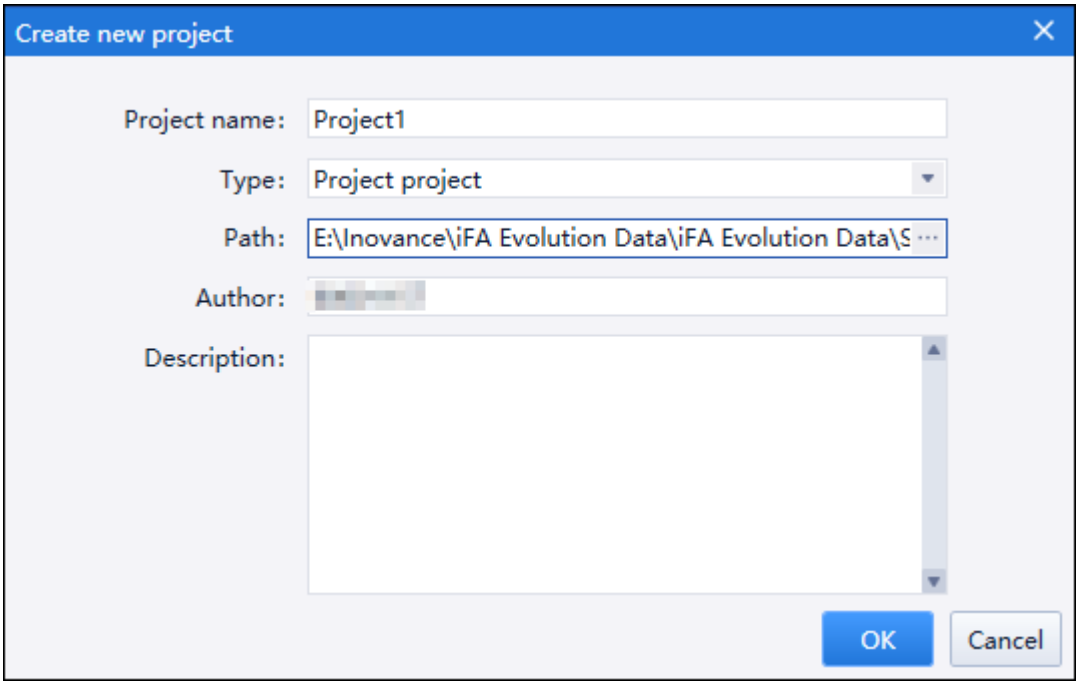
10.6.10.2 DP Master Configuration (iFA)

Prerequisites

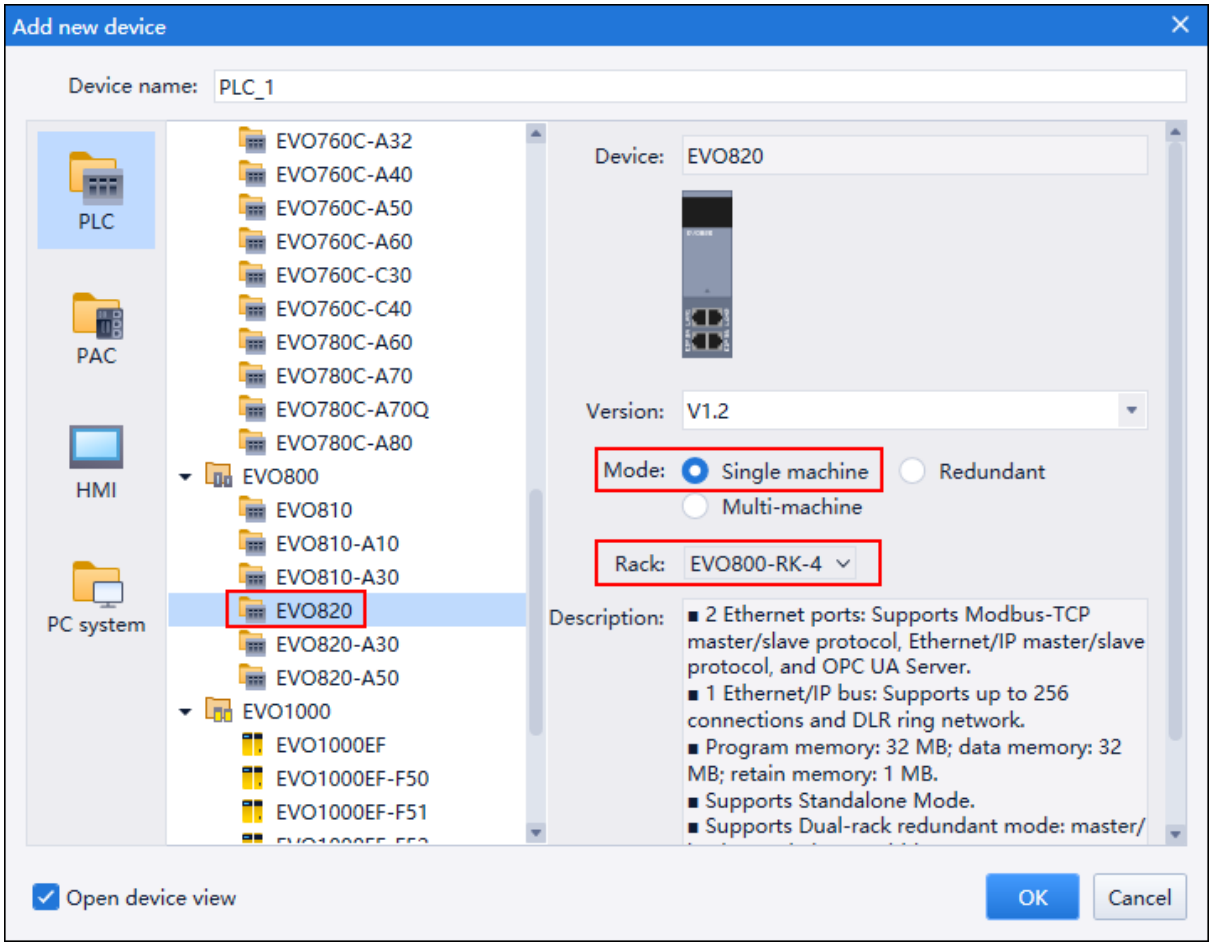
- All products in the configuration example have been installed, wired, and powered on.
- The PC installed with iFA Evolution is connected to the LAN1 port of the EVO820 control module via a network cable.

Procedure

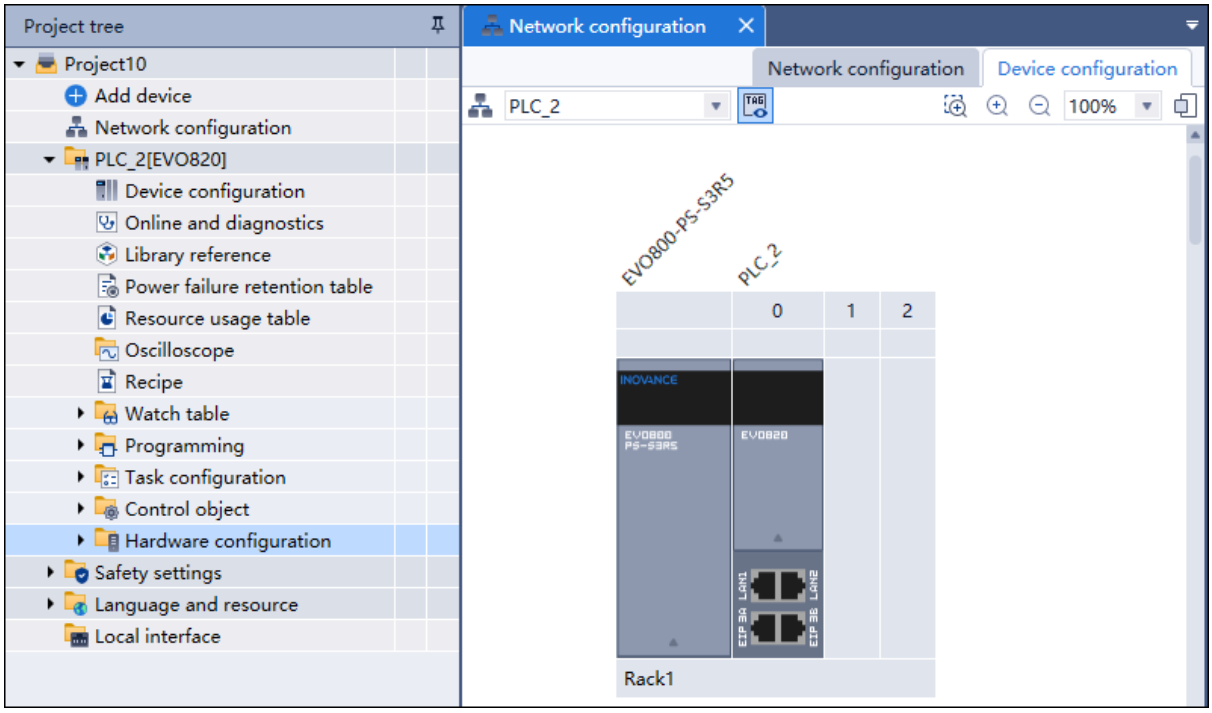
1. Create a new project.
   In the menu bar, select "Project > New". Enter the project information (select "Engineering project" for "Type"), and click "OK".



2. Add a new device.
   Double-click "Add device" in the project tree, click "EVO820" in the opened dialog box, enter the device name, "Mode" is set to "Single machine" by default, select the proper "Rack", and click "OK".

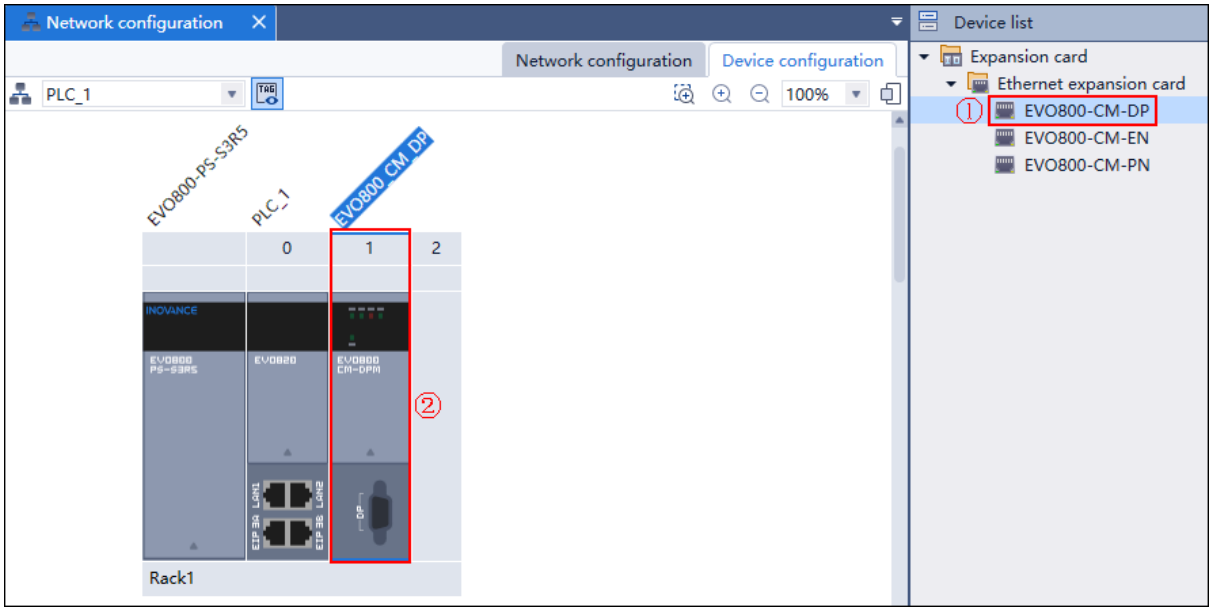


The newly added EVO820 device appears in the project tree.

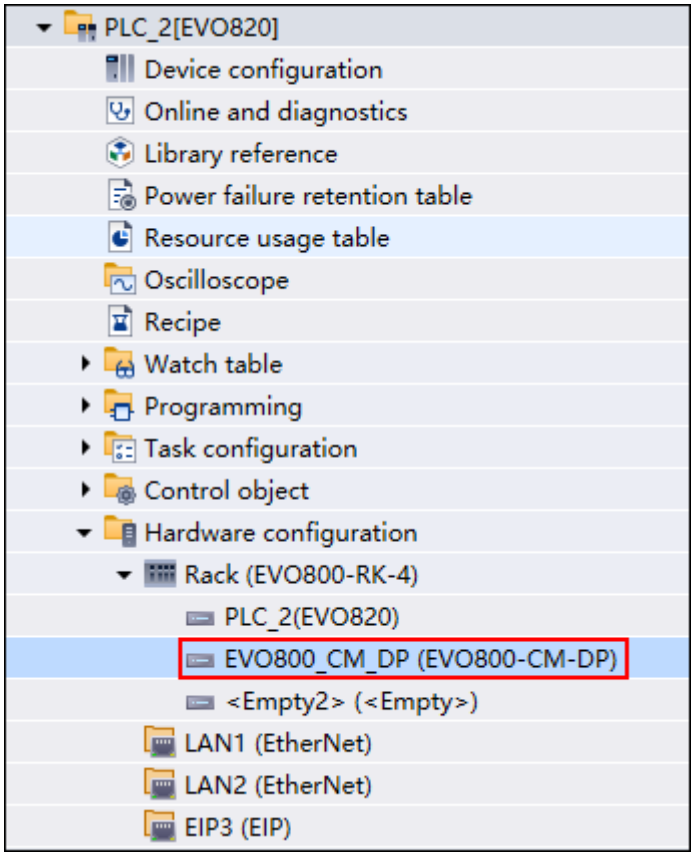


3. Add the EVO800-CM-DP module.

   In the project tree, double-click "Device configuration", select the target slot, then double-click "EVO800-CM-DP" in the "Device list" to add the DP master expansion module.



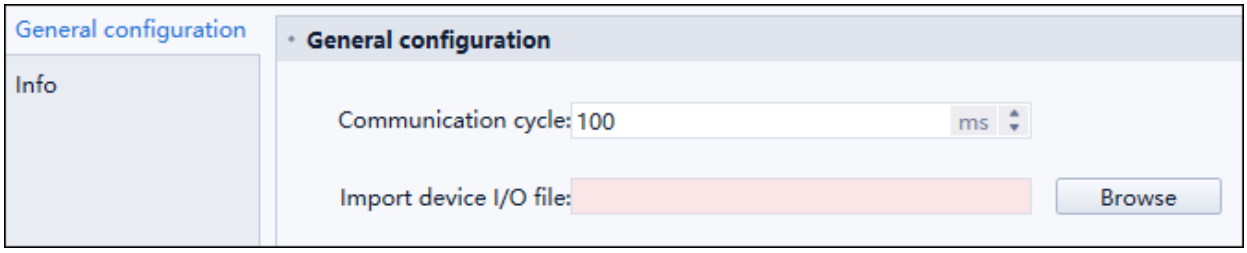
The newly added DP master expansion module appears in the project tree.



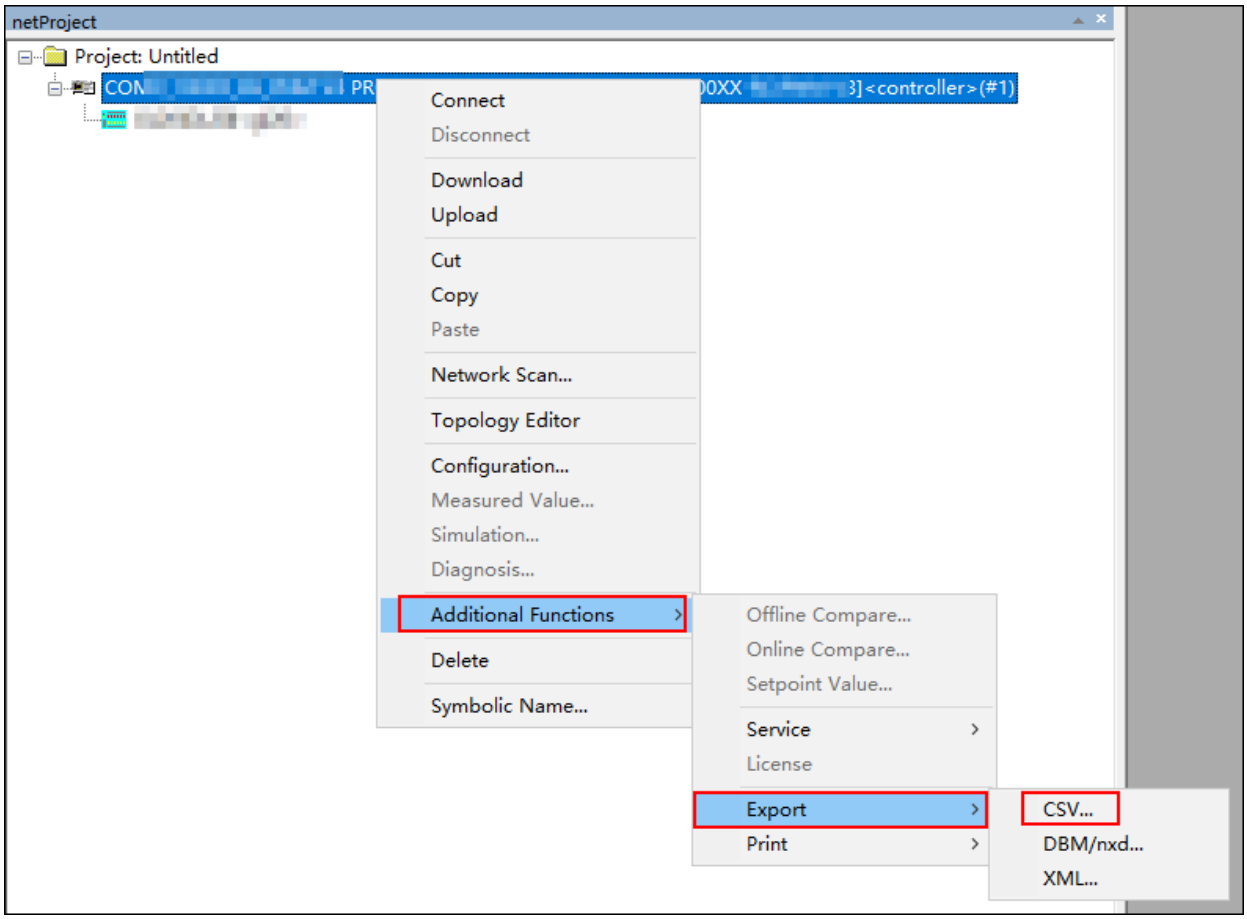
10.6.10.3 DP Master Configuration

10.6.10.3.1 General Configuration

Double-click the DP master node to enter the master parameter configuration interface.

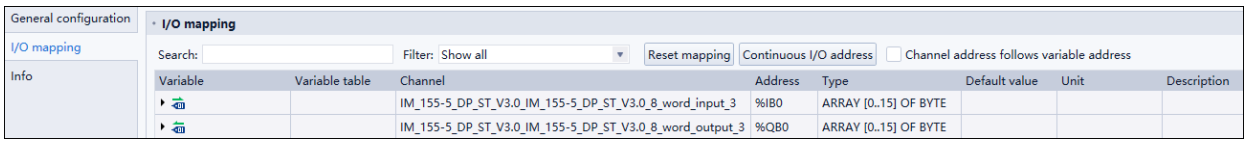


- Communication cycle: The backplate communication cycle between the PLC and the DP module. Default: 100 ms.
- Import device I/O file: Click "Browse" to select a .csv file (exported from SYCON.net software) for import.
  To export the I/O configuration file of the DP module from SYCON.net:
  After configuring the master, right-click the master, and select "Additional functions > Export > CSV" to save the file in CSV format.



10.6.10.3.2 I/O Mapping

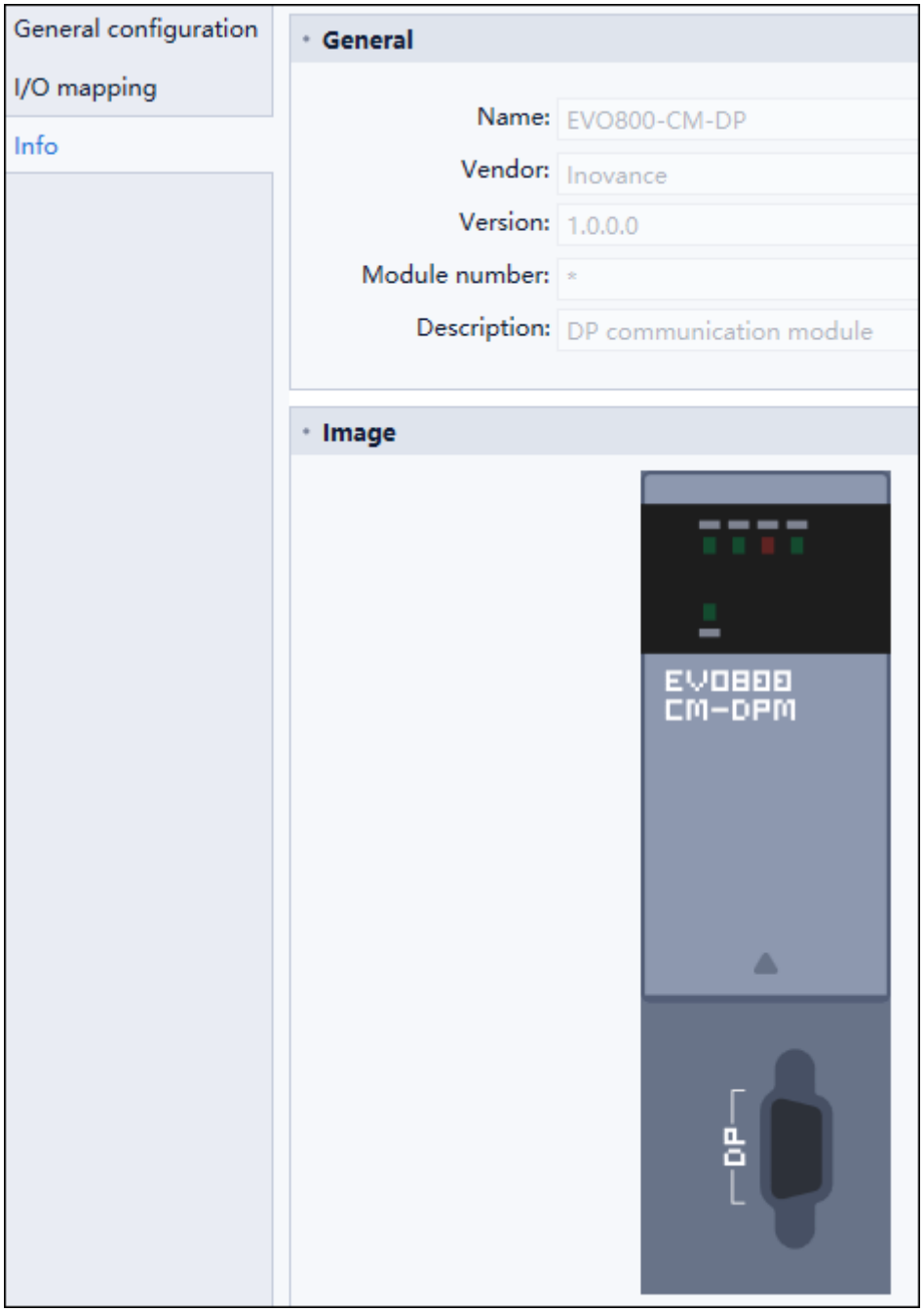
After importing the device I/O configuration file on the general configuration page, you can view the I/O settings related to DP communication, as shown in the figure below:



For details about the configuration parameters, see 10.6.3.3.9 I/O Mapping.

10.6.10.3.3 Information

This interface displays the general information and image of the device, as shown in the following figure:



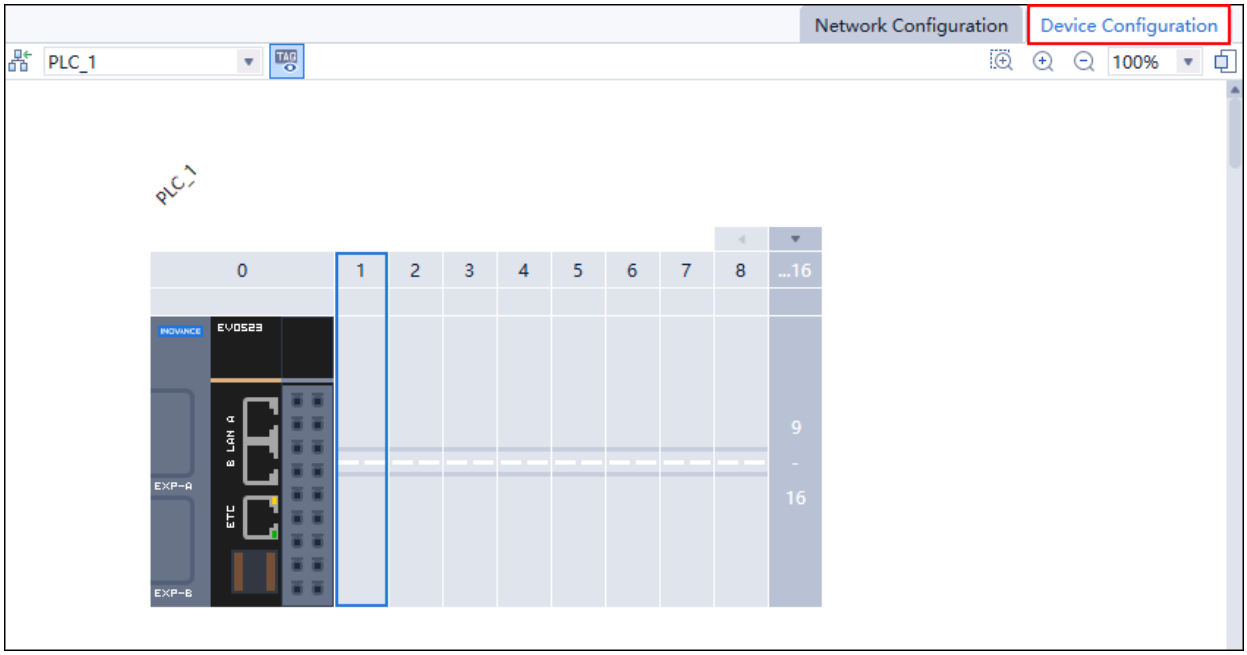
10.6.11 I/O Module Configuration

10.6.11.1 Rack Overview

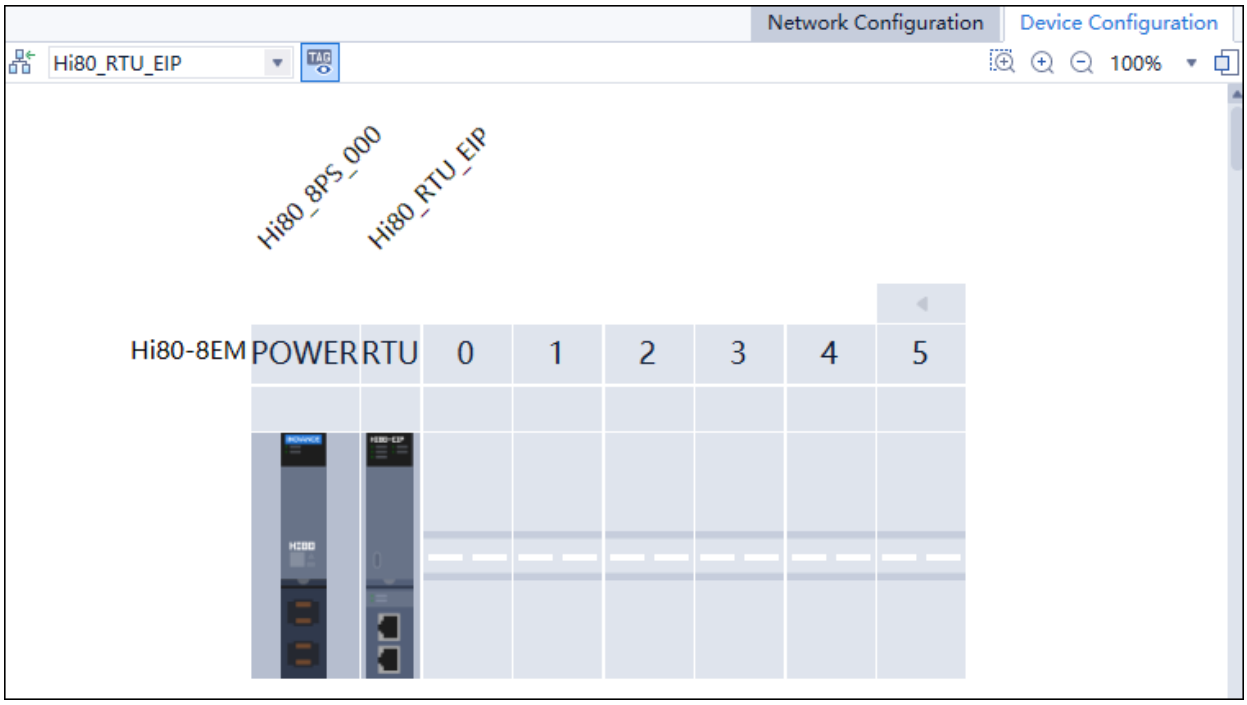
To mount modules after the main device, a rack (i.e., a mounting rail) must be used to secure the modules to the rack and connect them to the CPU, couplers, power supply, or other modules via the backplate bus.

Creating a rack

After adding a device, a rack suitable for the selected device is automatically created. The available slots in the rack are displayed in the device configuration view, and the number of available slots depends on the type of device used.

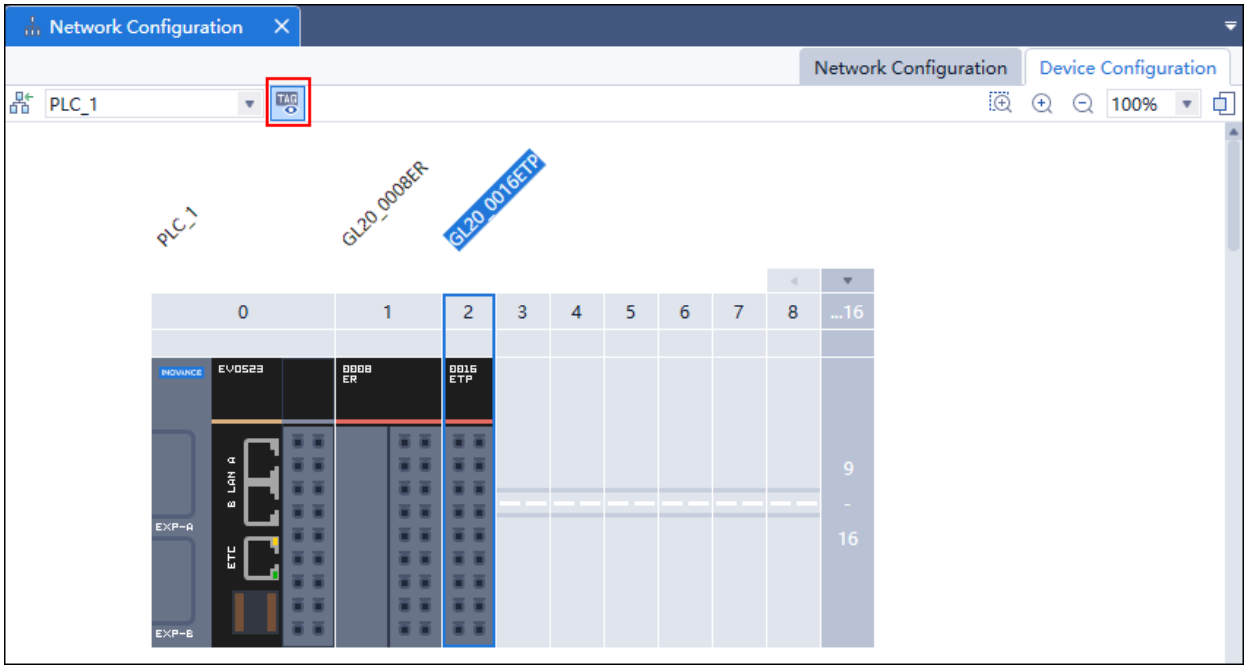


When adding a rack with a backplate bus (such as the Hi80/Hi50 series), the system automatically creates a power supply and an RTU communication module before slot 0. Depending on the selected module, the rack supports different backplates and numbers of slots. If a module is inserted into a slot, its properties can be viewed in the monitoring window.

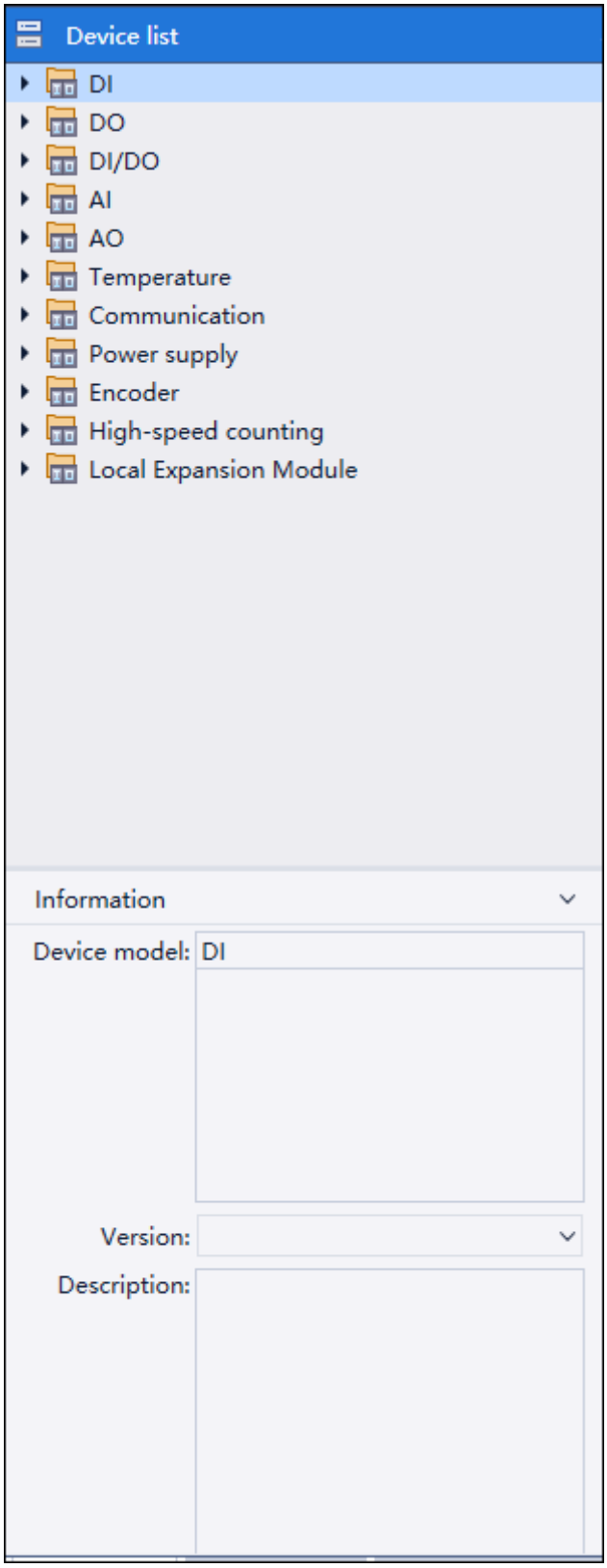


Rack structure

There are empty slots to the right of the device. The slot numbers are located above the slots and a brief description is displayed above the inserted device and modules. Use the "Display label" icon on the toolbar to display or hide the brief description of the device.



In the right device list, the slots that allow insertion of these modules are displayed.



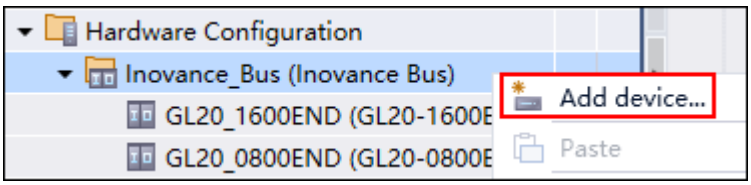
PLC local rack I/O modules is also known as local I/O modules. For the EVO500 series products, the CPU rack supports up to 16 local I/O modules by default for expansion. These modules include digital input modules (DI), digital output modules (DO), digital input/output modules (DIO), analog input modules (AI), analog output modules (AO), temperature modules (PT), power supply modules, and local expansion modules.

10.6.11.2 Adding a Local I/O Module

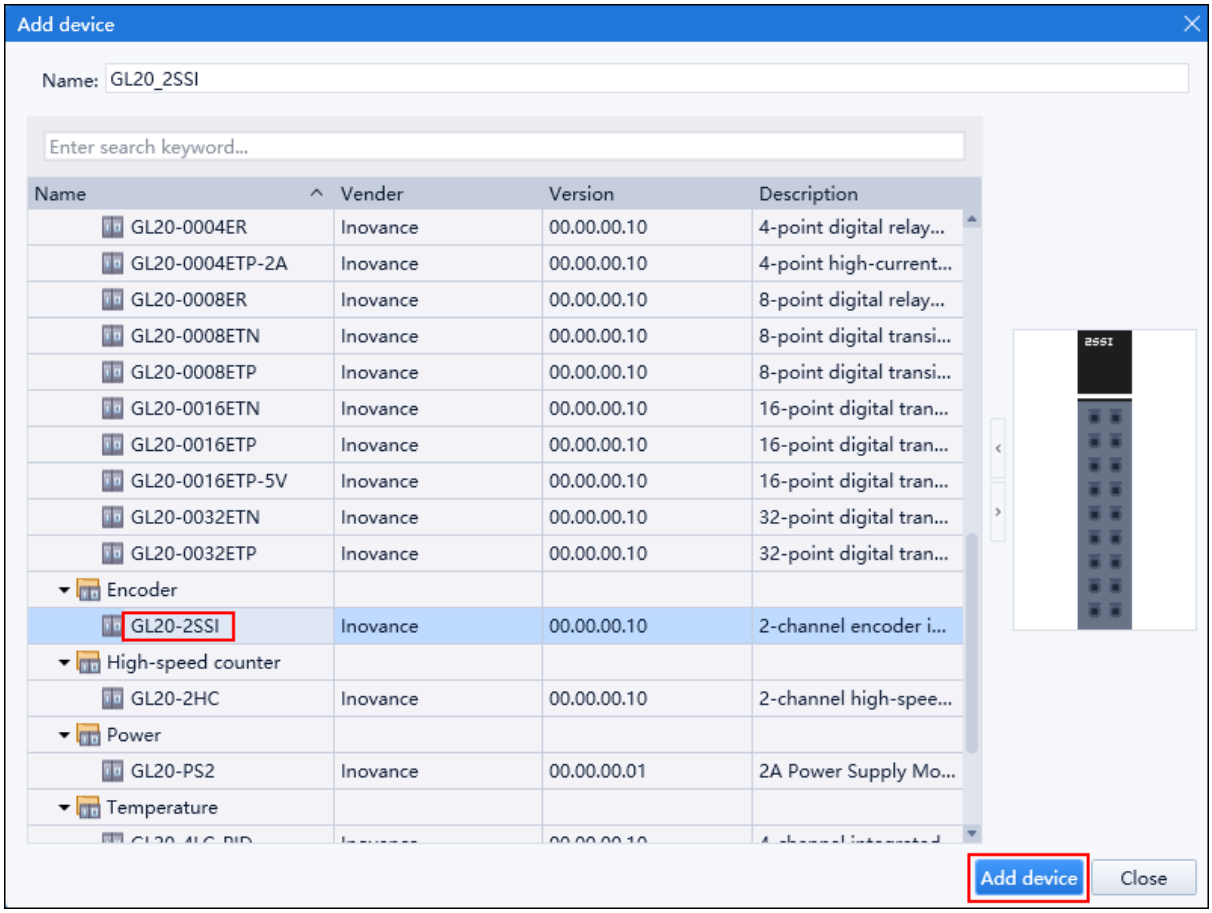
Local I/O modules can be added through the project tree or the device configuration view. The specific steps are as follows:

Method 1: Adding a module through the project tree

1. Expand the "Hardware configuration" list in the left project tree.
2. Right-click "Inovance_Bus (I/O Modules)" and select "Add device".

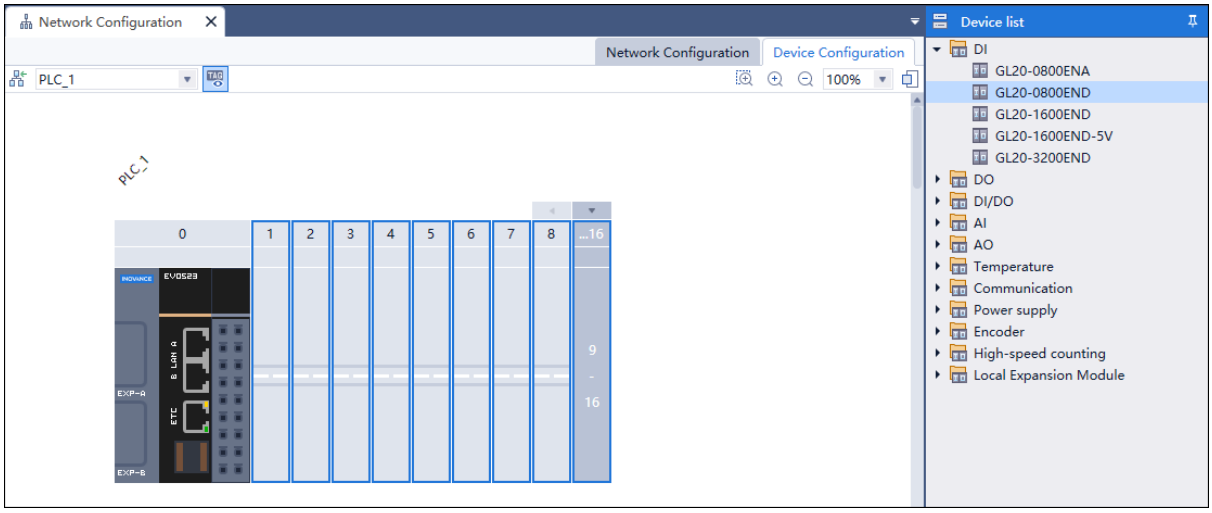


3. In the pop-up window, select the module and click "Add device", or double-click the module to complete the addition.

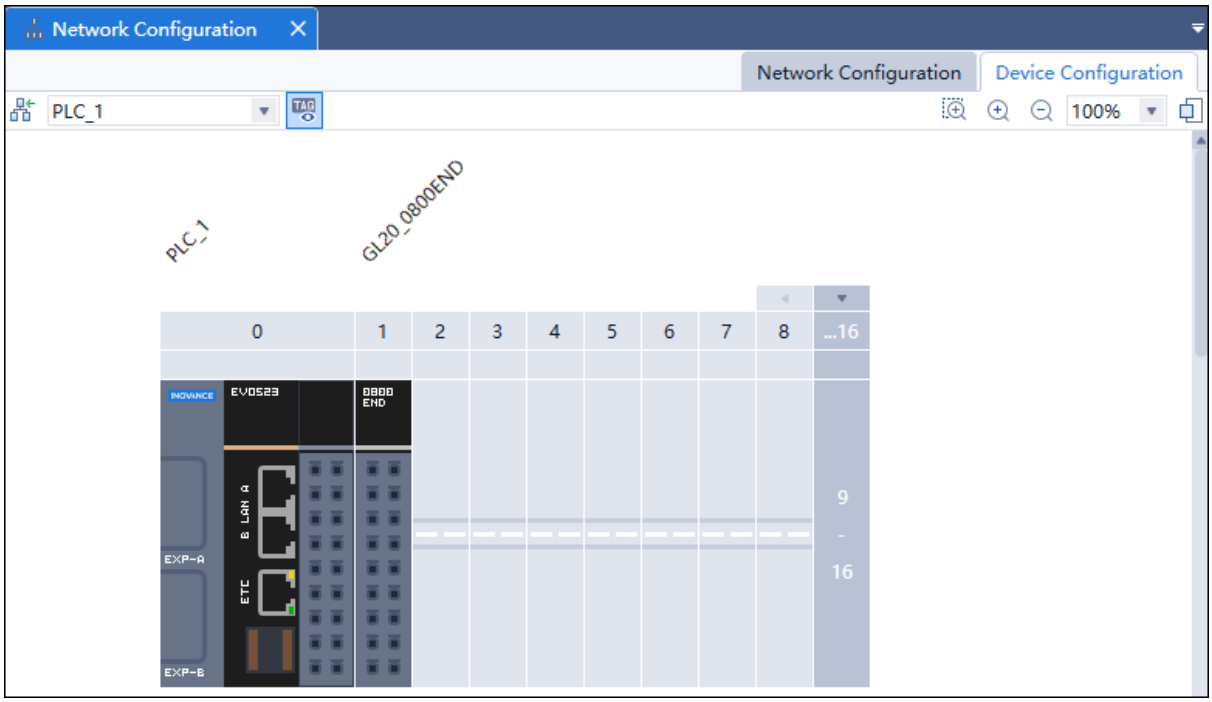


Method 2: Adding a module through the device configuration view

1. In the project tree, double-click the "Device configuration" node.
2. Switch to the device configuration view.
3. In the right device list, select the module. When a module is selected, the insertable slots for the module are highlighted in the configuration diagram.



4. In the device list, double-click the module to add it to the first available slot. You can also drag the module from the device list to an available slot in the device configuration view.



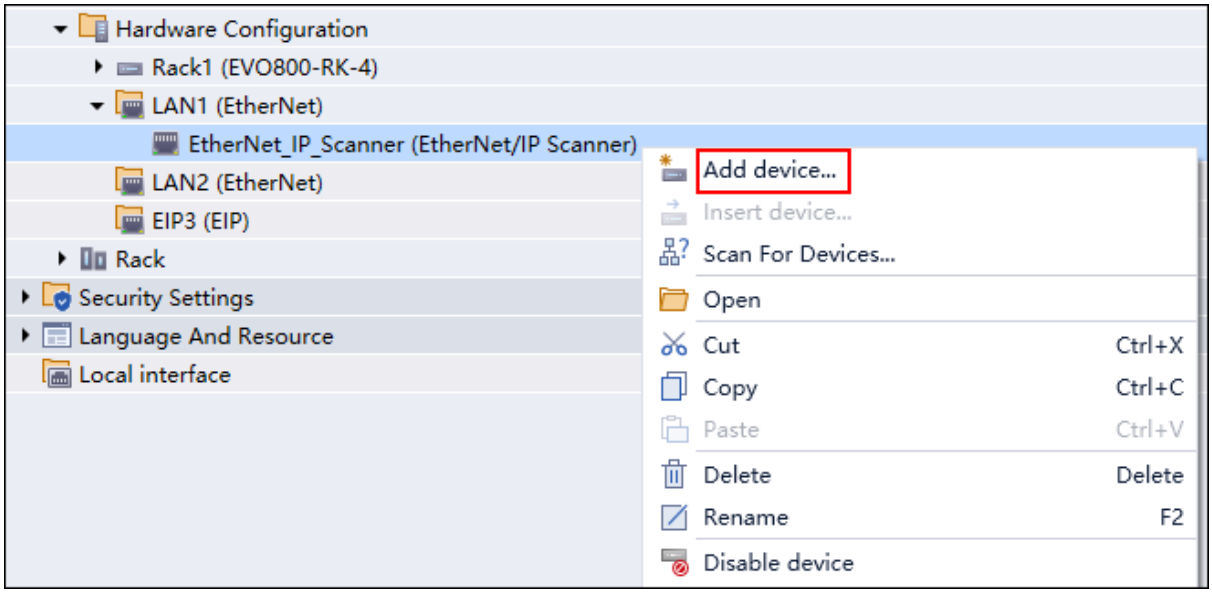
5. To change the module slot, drag the module in the device configuration view to the target slot.
6. The added module can be copied and pasted using the right-click menu or keyboard shortcuts.
   Only one module can be copied and pasted in the device list. Multiple modules can be copied and pasted in the device configuration view.

10.6.11.3 Adding a Remote I/O Module

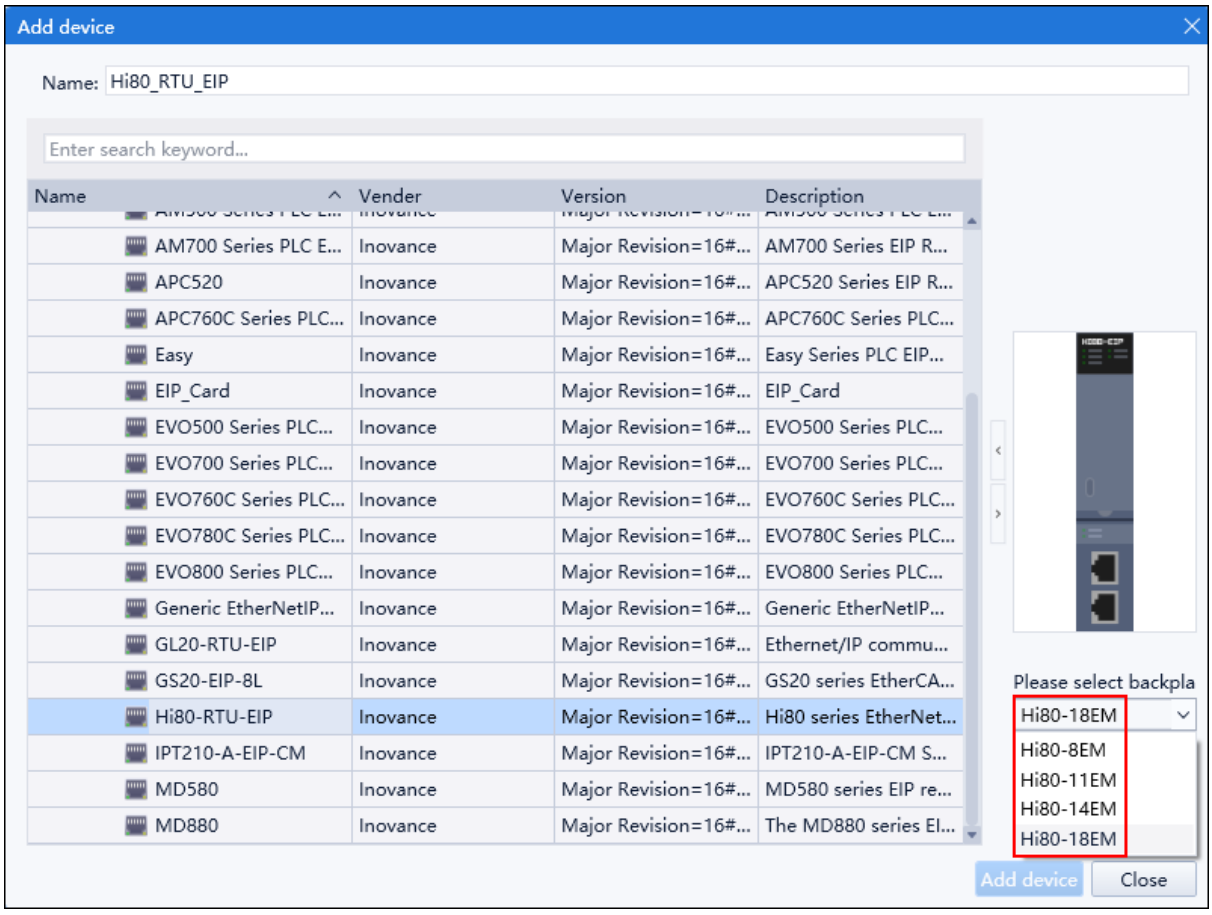
A remote IO module can be added through the project tree, device scanning, or network configuration view. The following example illustrates adding the Hi80-RTU-EIP module. The method for adding modules of other fieldbus types is similar.

Method 1: Adding a device through the project tree

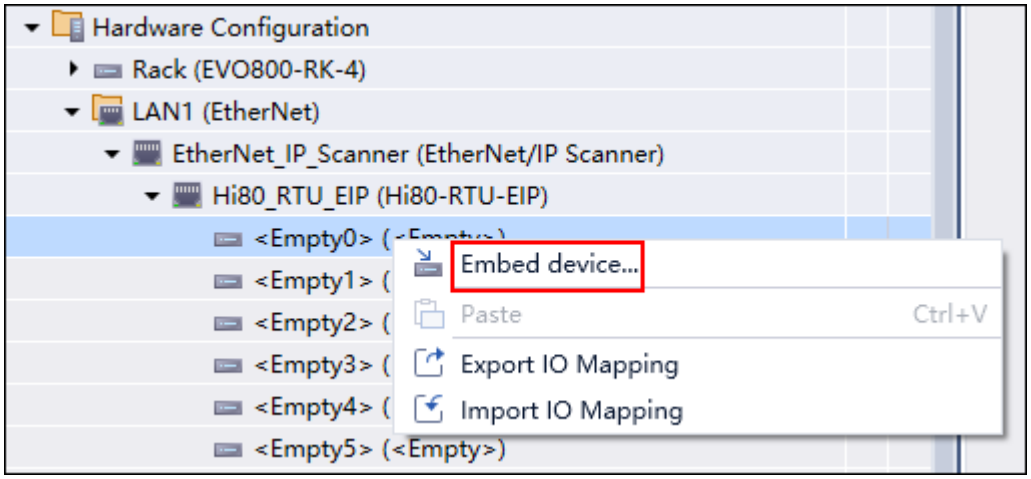
1. Expand the "Hardware configuration" list in the left project tree.
2. Right-click "LAN1(EtherNet)" and select "Add device".
3. Double-click "EtherNet/IP Scanner", or select it and click "Add device" to add the scanner.
4. Right-click the added scanner in the project tree and select "Add device".



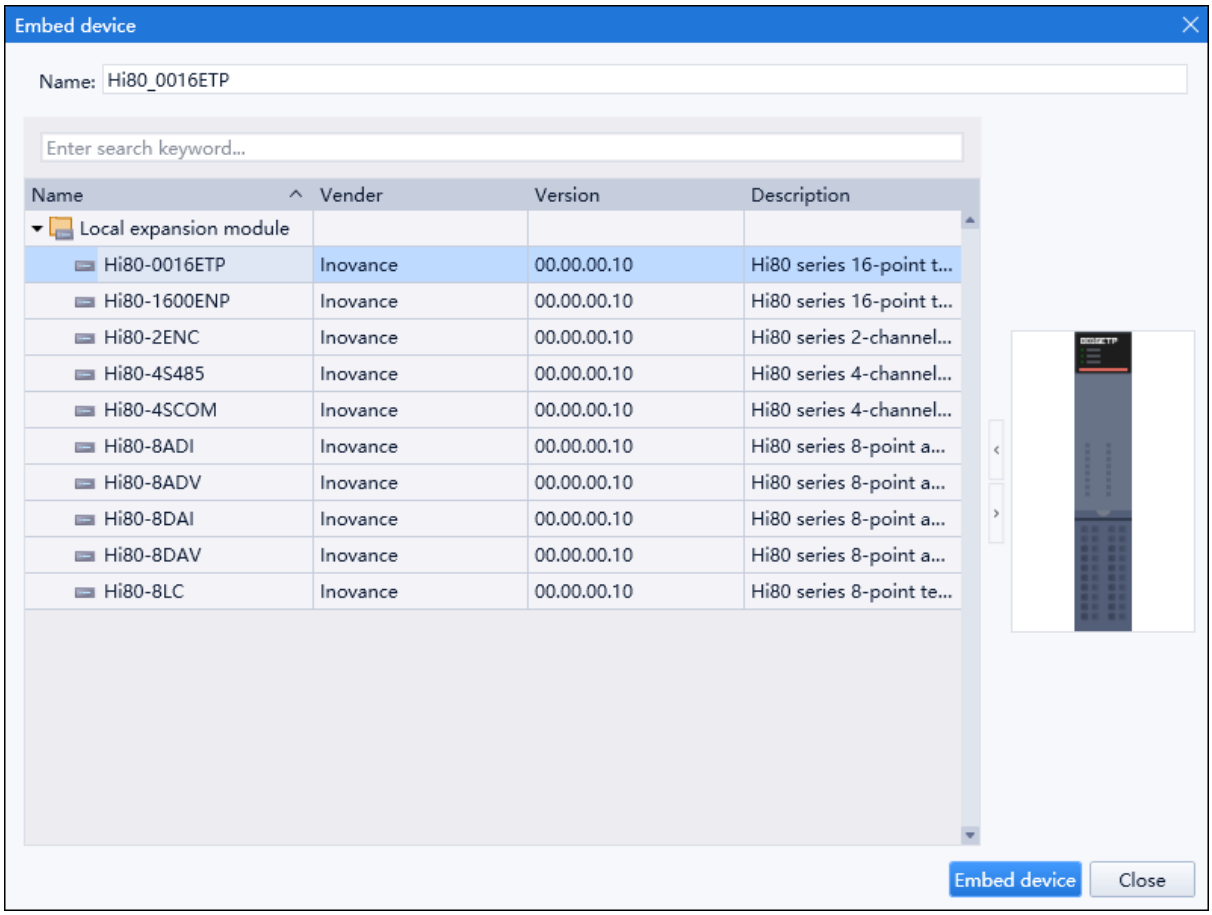
5. In the pop-up window, select the Hi80-RTU-EIP module, select a backplate in the drop-down menu on the right, and click "Add device", or double-click the module to complete the addition.



6. Right-click a slot of the Hi80-RTU-EIP module in the project tree and select "Embed device".

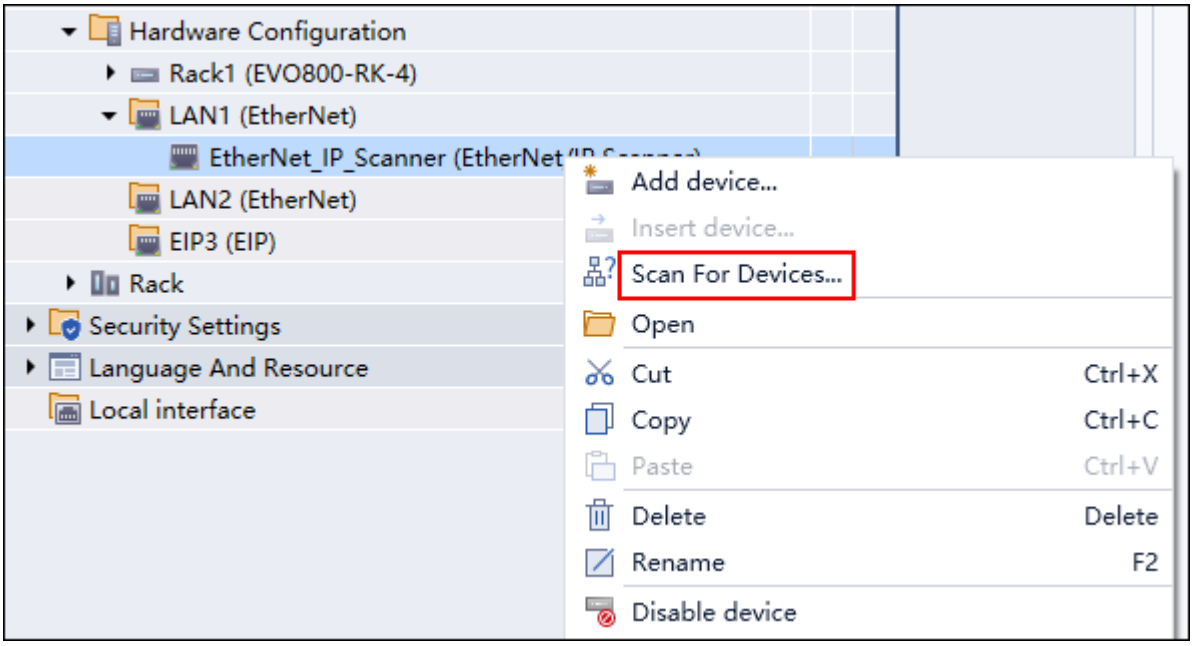


7. In the pop-up window, select the module and click "Embed device", or double-click the module.



Method 2: Adding a device through device scanning

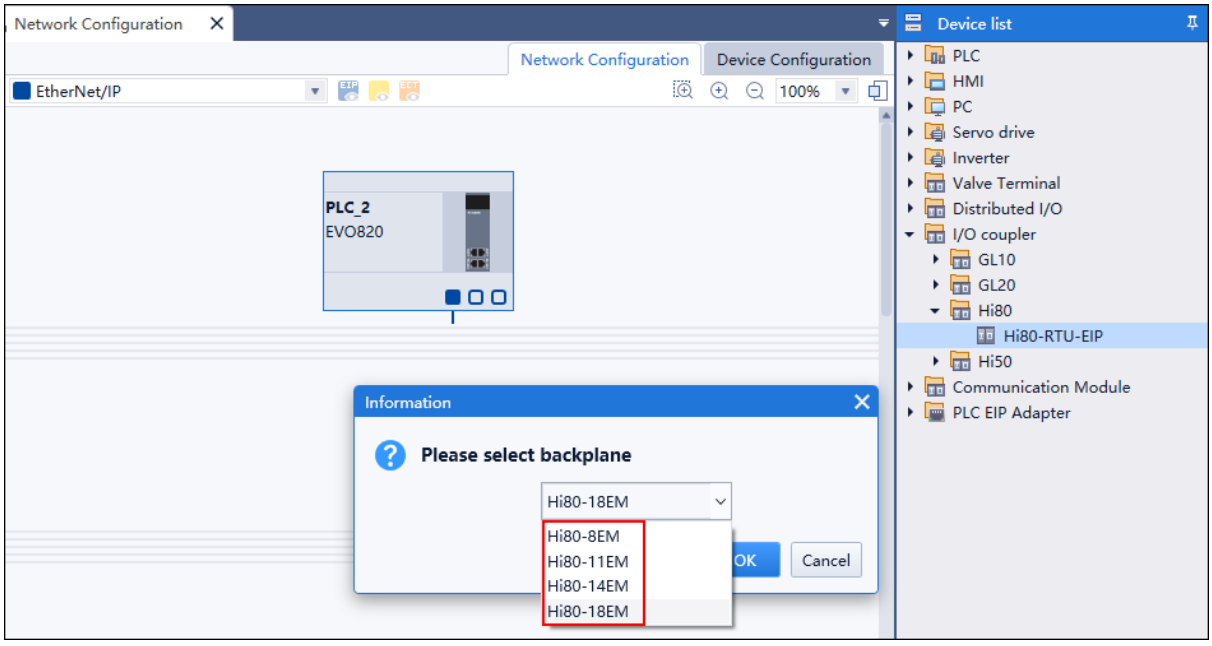
1. Expand the "Hardware configuration" list in the left project tree.
2. Right-click "LAN1(EtherNet)" and select "Add device".
3. Double-click "EtherNet/IP Scanner", or select it and click "Add device" to add the scanner.
4. Right-click the added scanner in the project tree and select "Scan for devices".



5. In the pop-up window, click "Scan for devices" and wait for the scan results.
6. Click the icon in the operation column of the Hi80-RTU-EIP module to add the target module to the left FA project area.
7. Click "OK" in the pop-up window.
8. Click "Apply project network configuration" to add the remote I/O module to the project.

Method 3: Adding via network configuration view

1. In the project tree, double-click the "Device configuration" node.
2. Switch to the network configuration view.
3. Select a module from the right "Device list" and add it by double-clicking or dragging-and-dropping.
4. In the pop-up window, select the backplate and click "OK".



10.6.11.4 Deleting a Module

The deletion operation of the module is the same as that of a regular slave device. The recommended method for deleting a module is as follows:

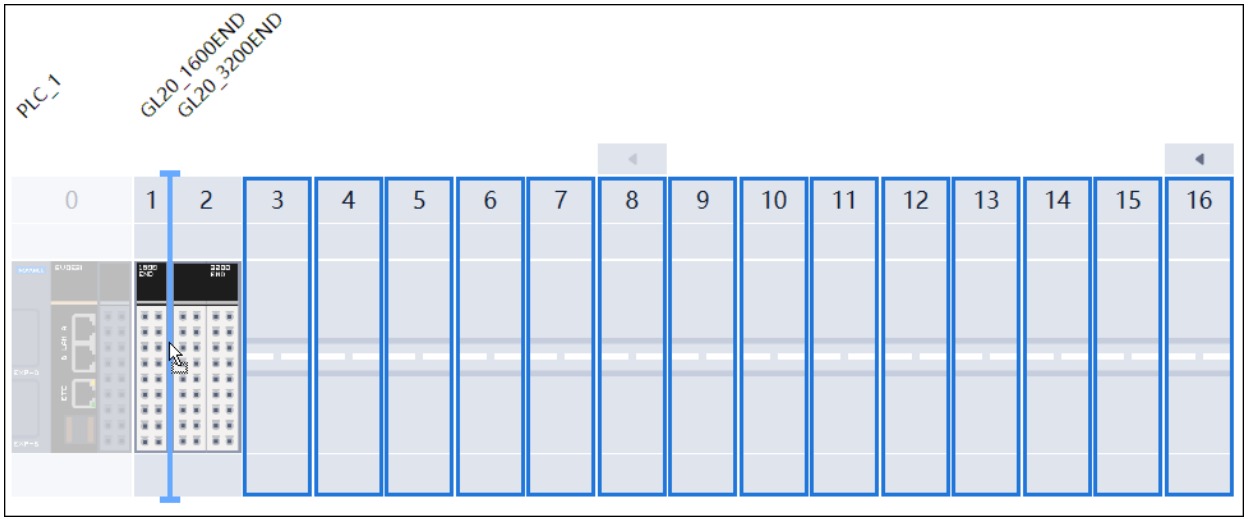
1. Double-click "Device configuration" in the project tree.
2. In the device configuration view, select a module on the PLC rack.
3. Press the Delete key to delete it.

When a module is deleted, the variables mapped to its channels will be released. At the same time, the module will be automatically deleted from "Hardware configuration > Inovance_Bus".

> NOTE
>
> After deleting an I/O module, the addresses of the associated I/O variables will be rearranged.

10.6.11.5 Inserting a Module

- Dragging and dropping from the device list
  Drag a sub-module from the device list to insert it into existing slots.



After successful insertion, the module that has been moved will be highlighted.

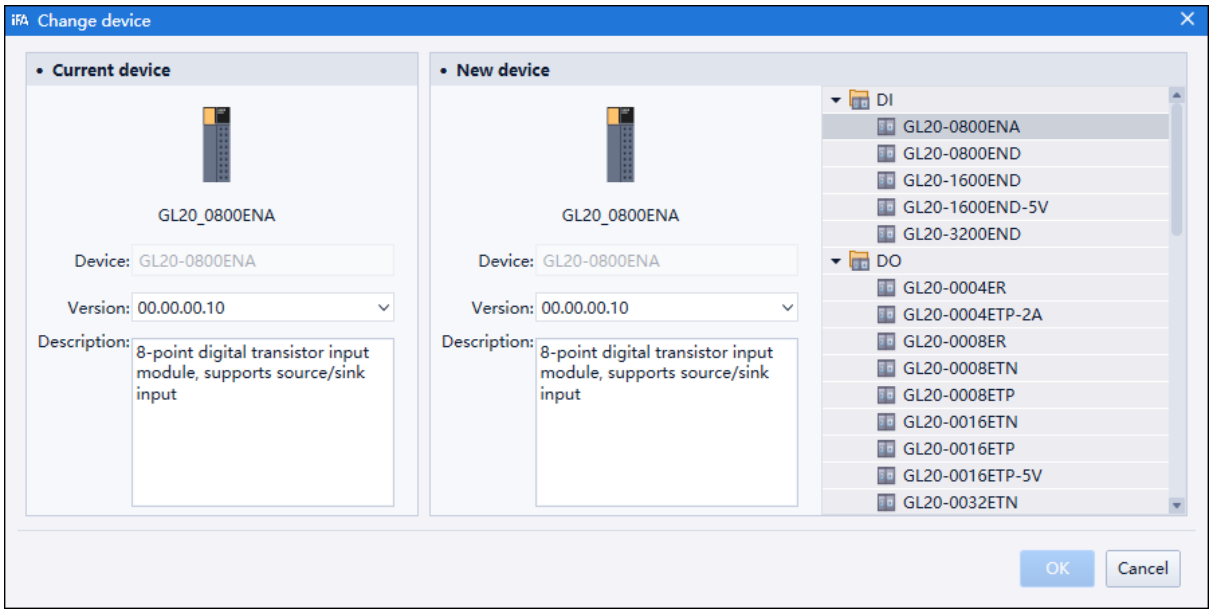
- Dragging and dropping from the device view
  Drag a sub-module from the device configuration view and insert it into another slot.

> NOTE
>
> After inserting an I/O module, the addresses of the associated I/O variables will be rearranged.

10.6.11.6 Replacing a Module

- Modifying by the right-click menu
  In the device configuration view, right-click a sub-device and select "Change device".

In the pop-up "Change device" window, the selected sub-module can be replaced with the new device.



- Modifying by dragging and dropping
  Drag a device from the "Device list" to the slot in device configuration diagram. If the slot is occupied, the "Change device" window pops up, where you can replace the original sub-module with the newly selected one.

10.6.11.7 Moving a Module

- Moving to an empty slot
  : In the device configuration view, drag a sub-module to move it to another empty slot.
- Moving to an occupied slot
  : In the device configuration view, drag a sub-module to move it to an occupied slot to swap the positions of the two modules. The sub-modules whose positions have changed are highlighted.

10.6.11.8 Disabling/Enabling an I/O Module

Disabling local or remote I/O modules

In iFA software, a certain I/O module can be disabled through commands or variable setting, thus "detaching" from the control system while retaining its configuration parameters and I/O mapping information. If a device fails and needs to be replaced, this function can reduce the time required for replacement.

- Disabling operation and effect
  A device can be disabled through commands or variable setting.
  - Command disabling
    Right-click an I/O module in the project tree and select "Disable device".
    Right-click an I/O module in the device configuration view and select "Disable device".
    After disabling, it is necessary to perform a 10.8.3 full download of the PLC to make the disabling operation effective.
  - Variable setting
    When adding each I/O module, a corresponding variable is created in **SYS_VAR**, following the naming rule: J_device model.
    Setting member The member **bDisableEnable** disables the slave, and the setting method supports

writing values in **SYS_VAR** or assigning values in user programs.

	Name	Data type	Address	Retain	Network publish	Comment
1	VAR_GLOBAL					
2	LAN1_EIPSingle_STATUS	Array[1..32] OF IoStand...			Private	
3	LAN1_EIPSlave_STATUS	Array[1..16] OF EtherNe...			Private	
4	J_GL20_3200END	IoStandard._SYS_SlaveS...			Private	
5	bDisableEnable	BOOL			Private	

After disabling, it is necessary to sequentially download, power off, and power on the PLC to make the disabling operation effective.

> CAUTION
>
> Powering off the PLC will result in the loss of values for temporary variables and variables with power-off retention set to FALSE.

- Disabling result

  The disabled devices are marked with the icon for differentiation.

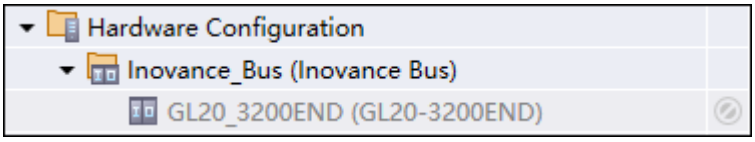


Figure 10-6-11-8-1 Icon in project tree

Enabling local or remote I/O modules

The disabling and enabling of I/O modules are a set of mutually corresponding operations. Only when the module is disabled is the enabling command visible.

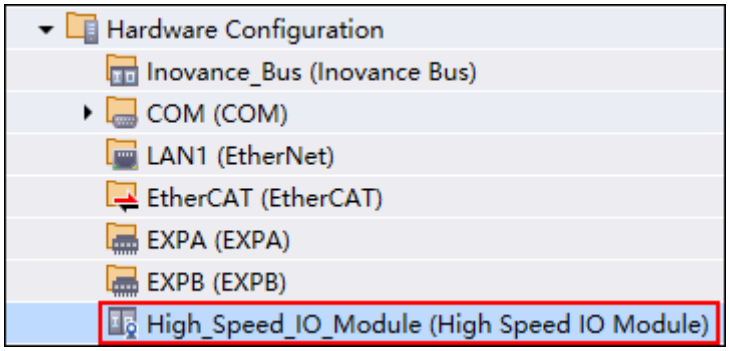
The enabling operation and its effect are exactly the same as the disabling operation.

A module needs to be disabled and enabled in the same way. That is, if the module is disabled by setting a variable, it must be enabled by resetting the variable. If it is disabled using a command, it must also be enabled using a command.

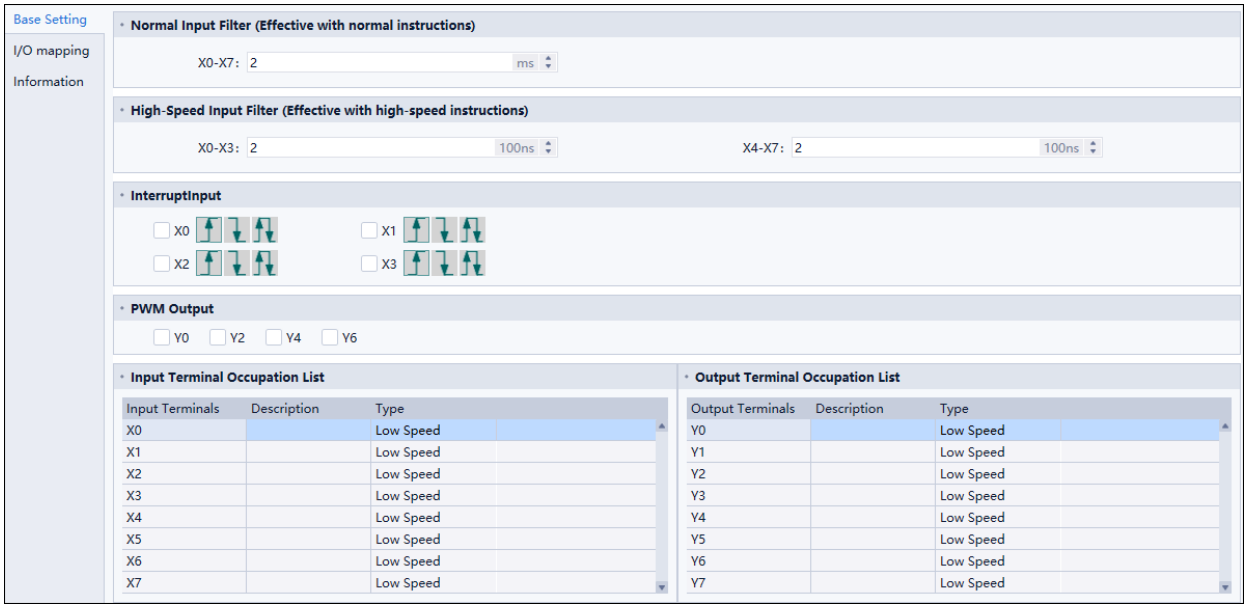
10.6.12 High Speed I/O Module Configuration (EVO400, EVO500, EVO700)

10.6.12.1 Opening High-Speed I/O Module Interface

In the project tree, double-click "High_speed_I/O_module" under the "Hardware configuration" node.



Open the corresponding configuration interface.



Filter parameters: Configure filter values for input signals, and display terminal usage status.

I/O mapping: Assign variables to terminals to acquire field signals, and support address changes.

Information: Display basic background information of the device for user reference.

10.6.12.2 Configuring a High-Speed I/O Module

Filter parameter

The filter setting for the input signals is used to remove waveform signals above the reference frequency, thereby achieving signal stability and accuracy.



Set the reference waveform frequency and the system will filter out waveform signals above the reference frequency. Then, the reference waveform frequency is converted into a waveform period. For example, to filter out waveform signals above 500 Hz, the waveform period corresponding to 500 Hz is 2 ms.

Terminal occupation list

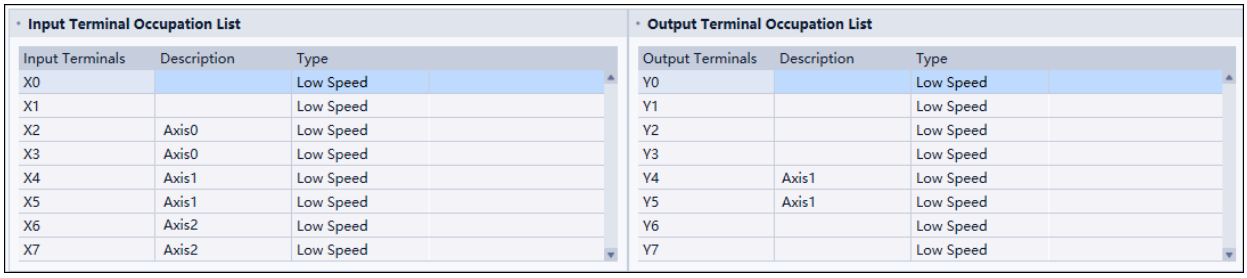


Figure 10-6-12-2-1 Configuration interface for EVO523 PLC

The list displays the high-speed and low-speed signal occupancy status of input/output terminals. High-speed I/O terminals connected to axis signals will be displayed as high-speed, while other signals will be displayed as low-speed.

High-speed I/O terminals support connection of signals from **local encoder axes** and **local pulse axes**. The signal configuration interface is concentrated on the **Axis basic settings** and **Mode/Parameter settings** tabs.



I/O Mapping

The "I/O mapping" tab is used to assign variables to channels to enable data transfer.

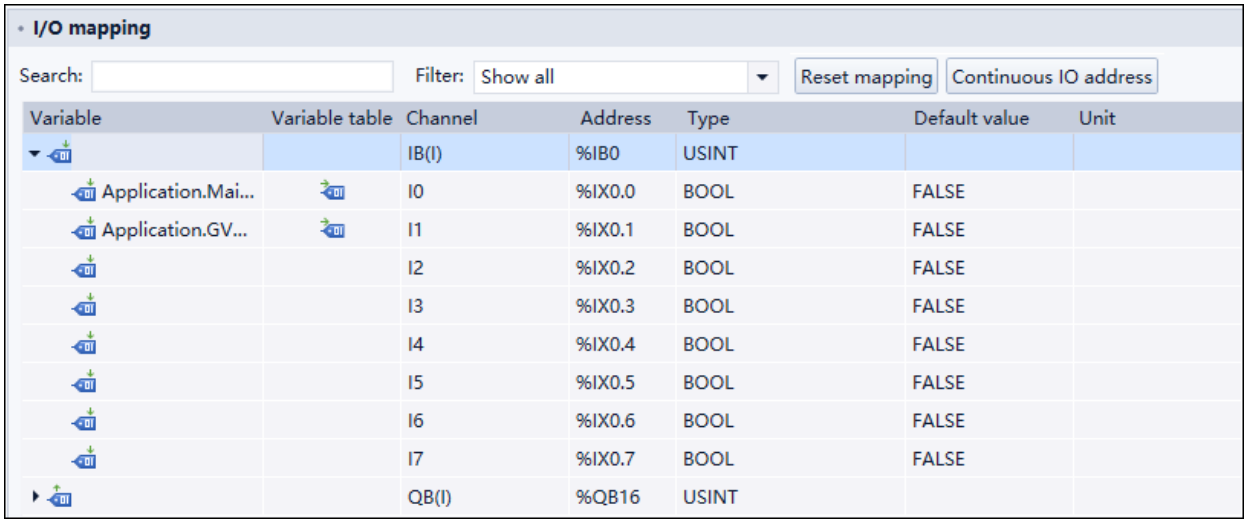
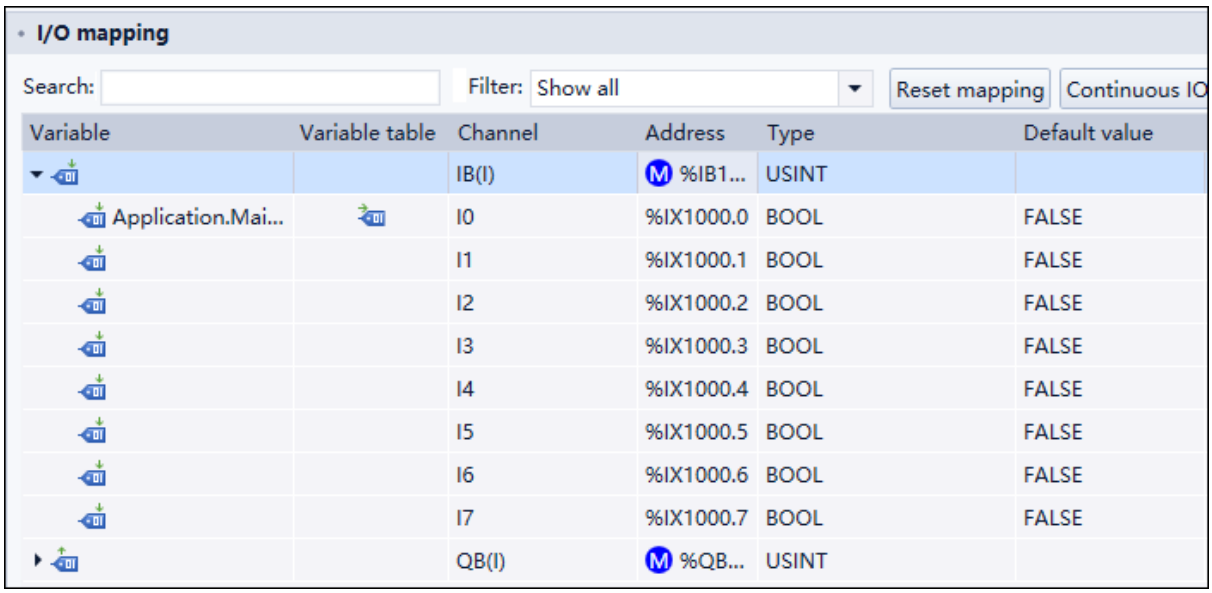


Figure 10-6-12-2-1 Configuration interface for EVO523 PLC

- Channel variables
  Variables support searching and filtering for viewing.
  Click "Reset mapping" to clear mapped variable information.

- Channel address
  Channel addresses support manual modification and system reassignment.
  Manual modification is effective for the parent node. After modification, it will be marked with **M**.
  System reassignment is completed using the "Continuous I/O address" button. If the initial default address is not occupied, the address remains unchanged; otherwise, a new continuous address will be assigned.



- Bus cycle options
  Used to assign tasks for high-speed I/O signals, which should be consistent with the tasks configured for the axis and axis control program to avoid compilation faults.
  The following diagrams illustrate the configuration of the three tasks, all of which are EtherCATMaster_Task.

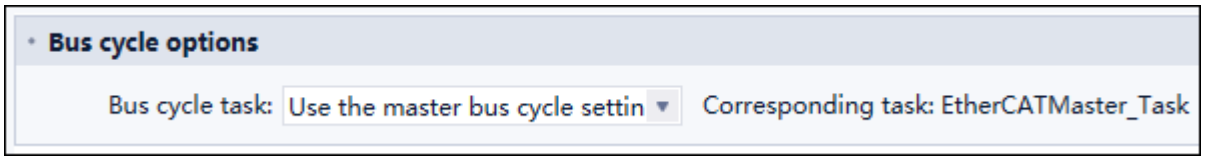


Figure 10-6-12-2-3 Task configuration for high-speed I/O signals

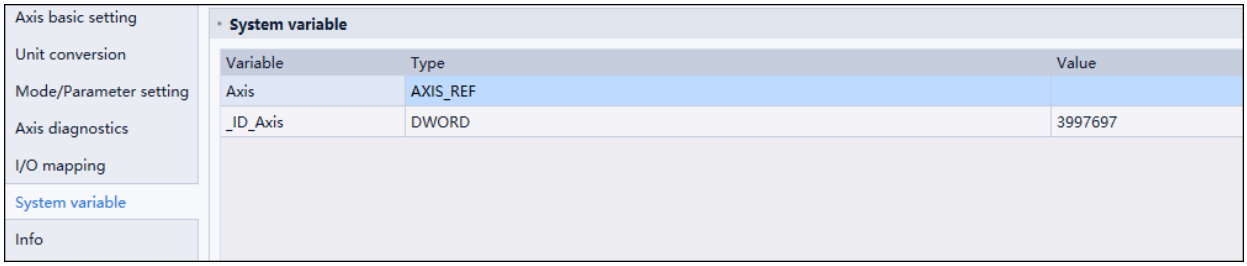


Figure 10-6-12-2-4 Task configuration for axis

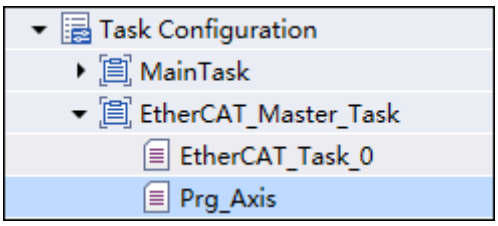


Figure 10-6-12-2-5 Task configuration for axis control program

NOTE

The three task configurations shall be consistent; otherwise, a device compilation fault will occur.

10.6.12.3 Disabling/Enabling a High-Speed I/O Module

Disabling a high-speed I/O module

Disabling a high-speed I/O module: With the physical module remaining unprocessed, disconnect the module from the system at the software level to stop its operation, preventing it from receiving or transmitting signals.

Right-click the "High_speed_I/O_module" node and select "Disable device". The node turns gray and is marked with .

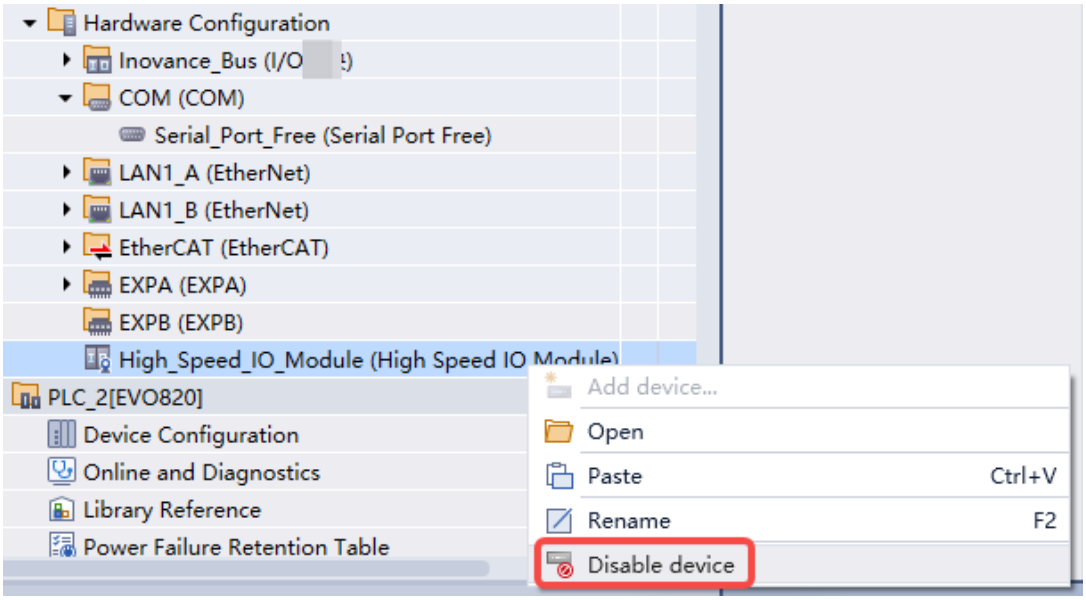


Figure 10-6-12-3-1 Disable command

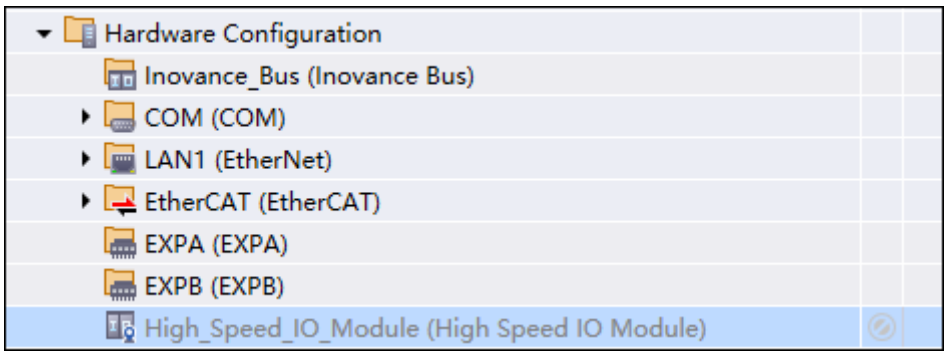


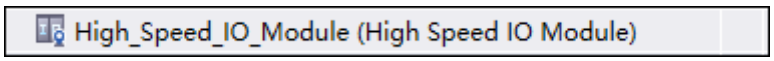
Figure 10-6-12-3-2 Disabled status

After disabling, it is necessary to perform a 10.8.3 full download of the PLC to make the disabling operation effective.

Enabling a high-speed I/O module

Enabling a high-speed I/O module: Restore the module within the system so that it can receive and transmit signals.

Right-click the "High_speed_I/O_module" node and select "Enable device". The node turns black and the mark disappears.



After enabling, it is necessary to perform a 10.8.3 full download of the PLC to make the disabling operation effective.

10.6.13 EXPA/EXPB (Local Expansion Module) Configuration (EVO400, EVO500, APC500)

10.6.13.1 Adding/Deleting a Local Expansion Module

Adding a local expansion module

A PLC supports adding two expansion modules.

The procedure of adding a local expansion module is the same as adding a slave device. The recommended procedure is as follows:

1. In the project tree, double-click the "Device configuration" node.
2. Switch to the device view.
3. Double-click a local expansion module in the device list.

The selected module will automatically be inserted into the left position of the PLC. At the same time, the added module is automatically displayed under "Hardware configuration > EXPA/EXPB" in the project tree.

To add a local expansion module through the device list, follow these steps:

1. Double-click "Device configuration" in the project tree.
2. Open the device view and select the left expansion slot on the PLC.
3. In the device list, select a local expansion module and double-click or drag and drop to add it.

To add a local expansion module in the project tree, follow these steps:

1. Right-click the EXPA/EXPB node under "Hardware configuration" in the project tree.
2. In the right-click menu, select "Add device" and select a local expansion module.

Deleting a local expansion module

When deleting a local expansion module, the variables mapped to the channels will be released.

The procedure of deleting a local expansion module is the same as deleting a slave device. The recommended procedure is as follows

1. Double-click "Device configuration" in the project tree.
2. In the device configuration view, select a module on the PLC rack.
3. Press the Delete key to delete it.

At the same time, the module is automatically deleted from "Hardware configuration > EXPA/EXPB" in the project tree.

10.6.13.2 Disabling/Enabling a Local Expansion Module

Disabling a local expansion module

Disabling a local expansion module: With the physical module remaining unprocessed, disconnect the module from the system at the software level to stop its operation, preventing it from receiving or transmitting signals.

Right-click the child node of EXPA/EXPB and select "Disable device". The child node will turn gray and be marked with .

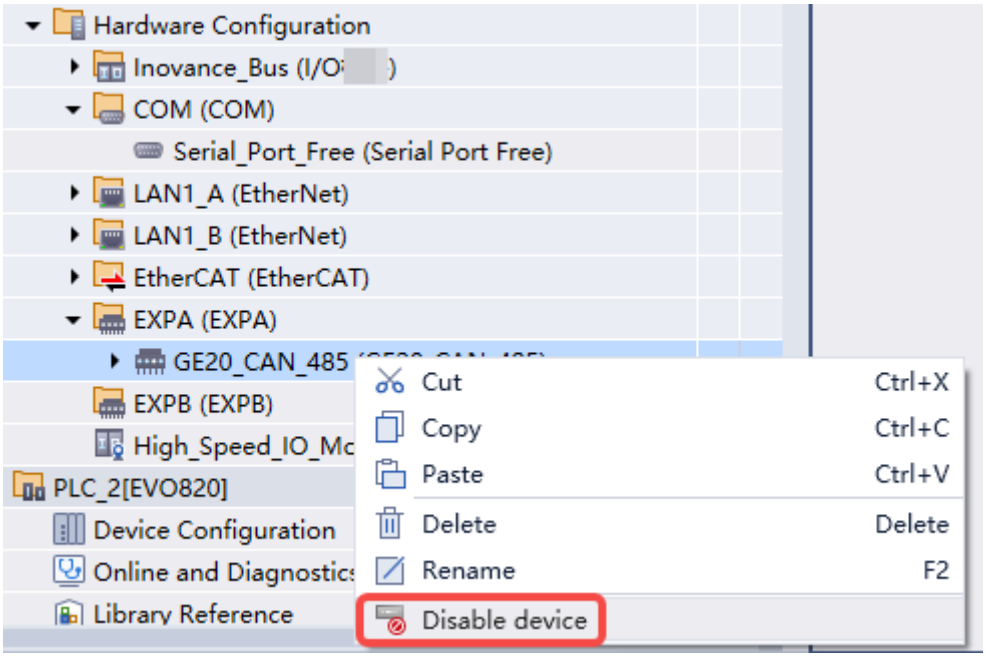


Figure 10-6-13-2-1 Disable command



Figure 10-6-13-2-2 Disabled status

After disabling, it is necessary to perform a 10.8.3 full download of the PLC to make the disabling operation effective.

Enabling a local expansion module

Enabling a local expansion module: Restore the module within the system so that it can receive and transmit signals.

Right-click the child node of EXPA/EXPB and select "Enable device". The child node will turn black and the mark disappears.

After enabling, it is necessary to perform a 10.8.3 full download of the PLC to make the disabling operation effective.

10.6.14 PLC Property Configuration

10.6.14.1 I/O Settings (EVO400, EVO500, EVO700)

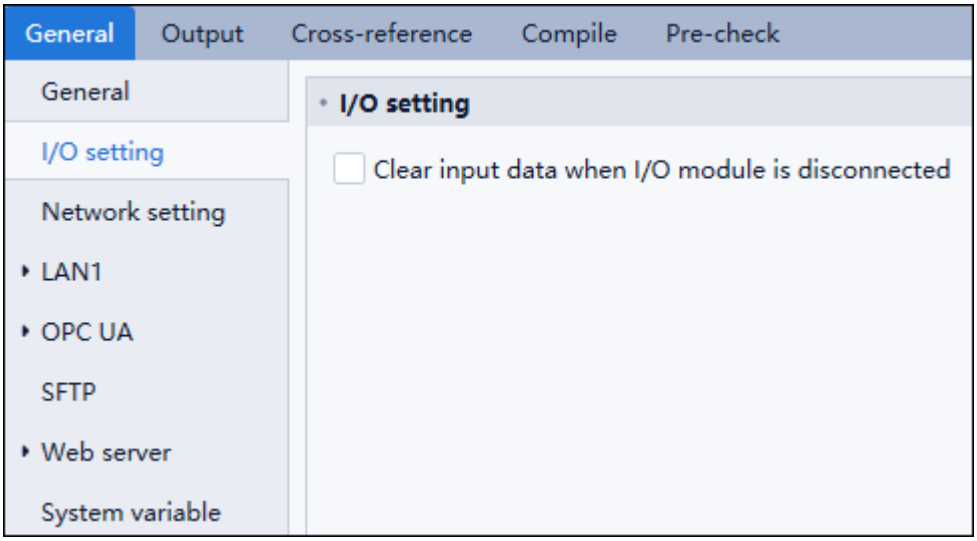
I/O settings are used to configure how input data is handled when a module disconnects during local I/O module configuration.

Prerequisites

PLC devices support configuration of local I/O modules.

Procedure

1. Double-click "Device configuration" in the left project tree and go to the "Network configuration" tab.
2. Select the PLC device diagram. In the "General" tab of the monitoring area below, select "I/O setting" to open the configuration interface.



Clear input data when I/O module is disconnected: Disabled by default. When enabled, if any I/O module loses communication, its corresponding input data area will be cleared.

> NOTE
>
> The I/O settings of the PLC device apply to all modules connected under the local I/O module.

10.6.14.2 Network Settings (EVO500)

Enable the corresponding product features by configuring different network modes for EVO500 series PLCs.

Restrictions

The following conditions must be met simultaneously:

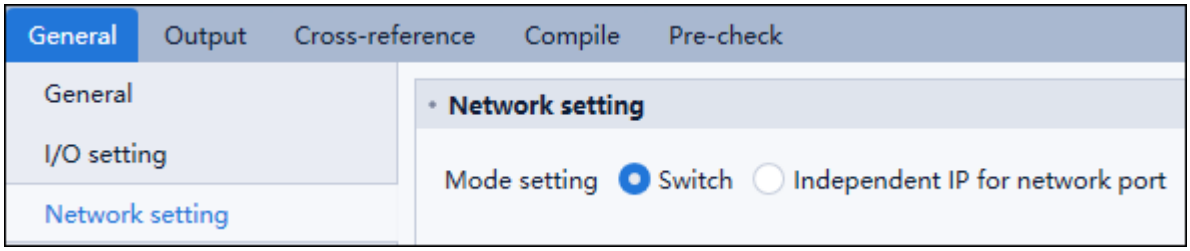
- EVO500 series PLC with firmware version V1.3 or above.
- Device purchased in or after July 2025.

Description

- Neither switch mode nor dual independent network port mode should be connected to a switching network to form a ring network.
- After switching from switch mode to dual independent network port mode, Port A inherits the IP address from switch mode, and Port B defaults to 192.168.2.88.
- After switching modes, the configuration environment EIP needs to re-select the network interface.
- After switching modes, the network will disconnect. Ensure the operation is performed in a safe environment.

Procedure

1. Double-click "Device configuration" in the left project tree and go to the "Network configuration" tab.
2. Select the PLC device diagram, and select mode through the "Network setting" tab in the monitoring area.



Mode	Note
Switch	The default is switch mode, where only LAN1 port can be configured.
Independent IP for network port	◦ LAN1_A The IP address remains the original settings. For example, after switching from switch mode to "Independent IP for network port", the IP 10.44.122.1 remains unchanged. ◦ LAN1_B When switching from independent IP mode to switch mode, all data and configurations on the LAN1_B port are deleted. ▪ If "Set IP address in the project" is checked, the IP follows the project settings and does not support setting a gateway. For detail, see 10.6.14.4 Ethernet Address Settings. ▪ If "Use the IP address directly from the device" is checked, the default IP address 192.168.2.88 is used for the first setup in independent IP mode, and subsequent setups retain the original settings.

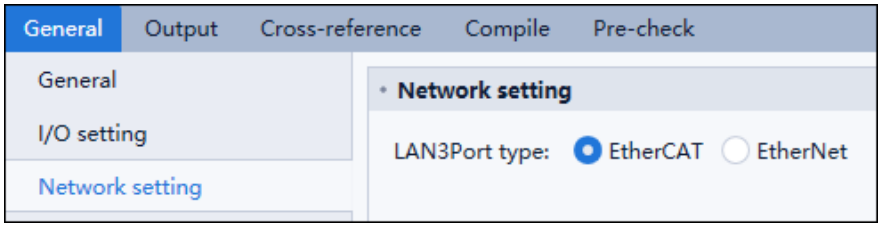
Feature comparison

The impact of functions under the two modes is shown in the table below:

EVO500 Network Port Configuration	Switch Mode	Independent IP for Network Port	
Port number	LAN1	LAN1_A (100 Mbps)	LAN1_B (100 Mbps)
EthernetIP	Supports master and slaves	Supports master and slaves	Supports master and slaves
ModbusTCP master	Support master	Support master	Support master
ModbusTCP slave	Support slave	Only one port can be configured	
OPC UA	Supported	Only one port can be configured	
CANopen	Not supported	Not supported	Not supported
EtherCAT	Not supported	Not supported	Not supported
PROFINET	Support slave	Not supported	Not supported

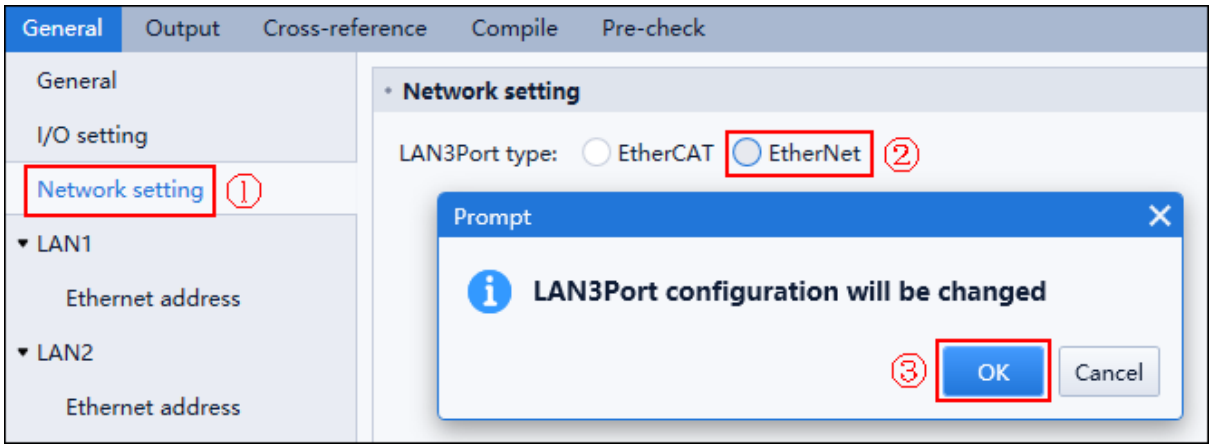
10.6.14.3 Network Port Switching

The LAN3 port of the EVO760 series PLC defaults to EtherCAT type. When the firmware version is V1.5 or above, the LAN3 interface supports free switching between EtherCAT and Ethernet types. By switching different types of network ports, different product features can be enabled.

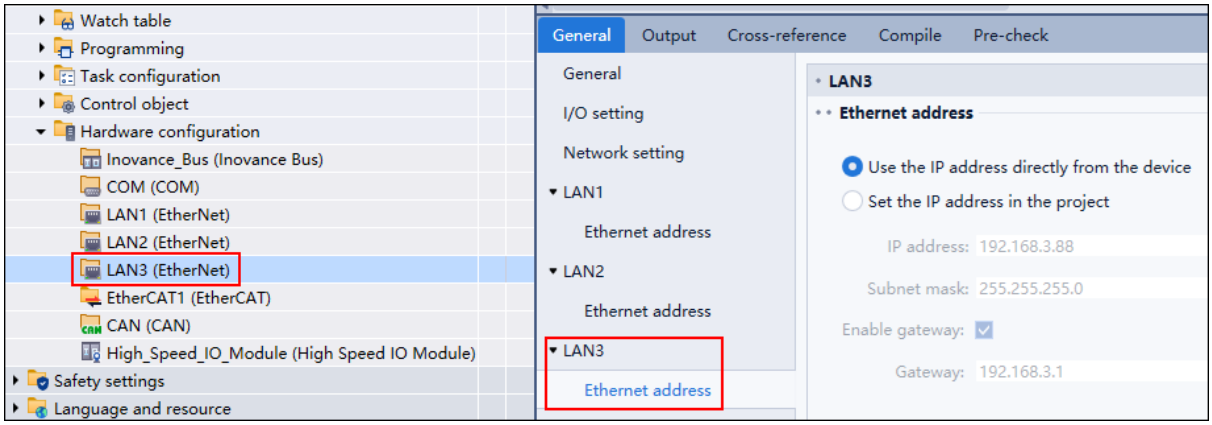


LAN3 network port configuration

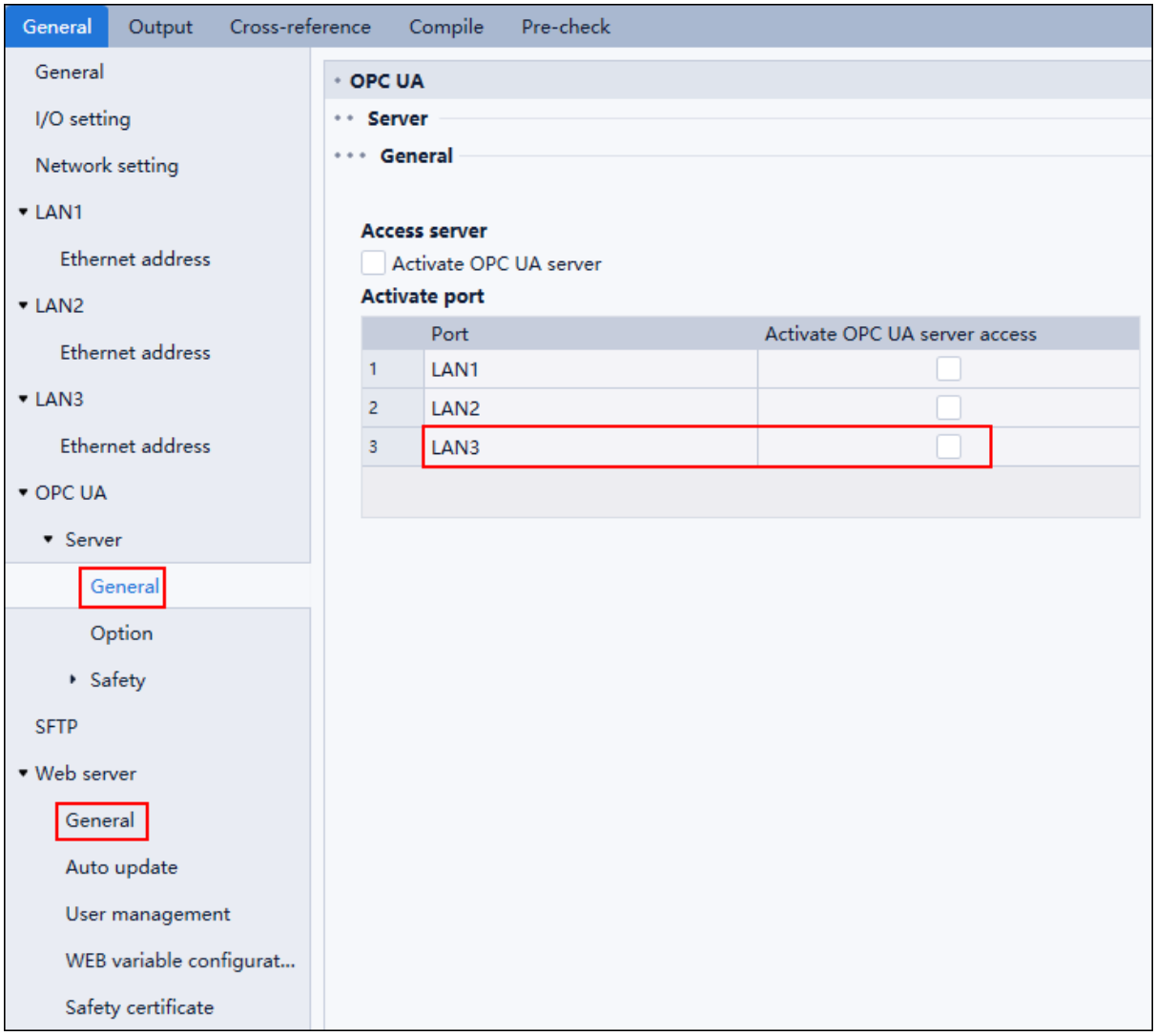
1. On the "General > Network setting" tab, select "Ethernet" as the type for the LAN3 network port.
2. A window pops up, indicating LAN3 configuration will be added. Click "OK" to switch the LAN3 interface to Ethernet type.



3. A "LAN3 (EtherNet)" node is added in the project tree. A LAN3 Ethernet address configuration tab is added on the "General" configuration interface, enabling three independent IPs.

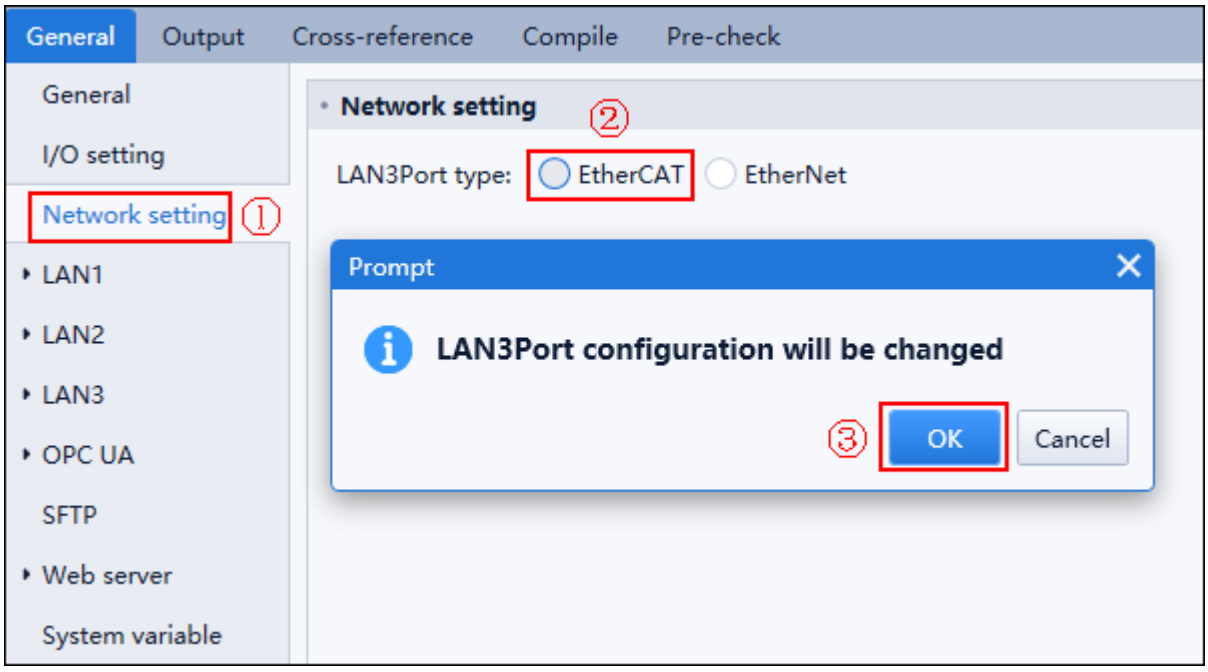


4. A LAN3 port option is added to the "OPC UA > General" and "Web server > General" tabs, which can be activated as access ports for OPC UA and Web server.



5. On the "General > Network setting" tab, select the LAN3 port configuration as the "Ethercat" type.

6. A window pops up, indicating LAN3 configuration will be deleted. Click "OK".



7. When the LAN3 interface is switched to the EtherCAT type, the LAN3 configuration in the project tree and general settings will be deleted.

Device update

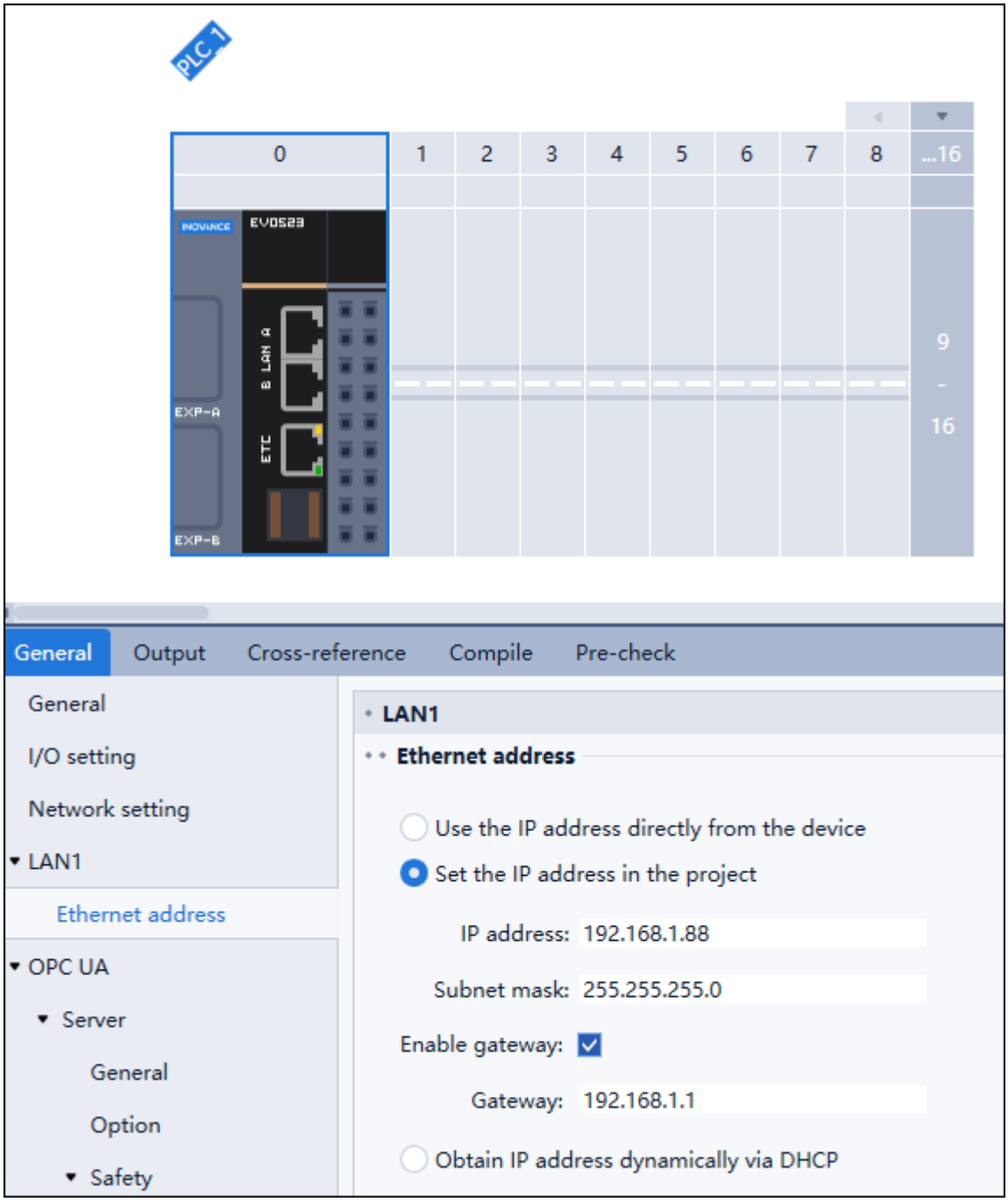
- When switching between devices of the same series (e.g., EVO762 to EVO763), the network port type remains unchanged, and the conversion is performed directly.
- When switching between devices of different series (e.g., EVO760 to EVO760C), the conversion is performed based on the actual port types supported. If a device does not support a certain port type, the corresponding port configuration is deleted.

10.6.14.4 Ethernet Address Settings

The Ethernet address of the PLC is a unique identifier for communication. Duplicate IP addresses are not allowed within a system.

Ethernet address setting

1. Double-click "Device configuration" in the left project tree and go to the "Network configuration" tab.
2. Select the PLC device diagram. On the "General" tab in the monitoring area below, select "Ethernet address" of the target network port.



3. Configure the Ethernet address, with parameters described in the table below.

Parameter		Description
Use the IP address directly from the device		When checked, the pre-set static IP on the device is used. The device IP address, subnet mask, and gateway can be modified in the 17.1.2 Network setting interface. The changes will automatically synchronize here after the device goes online.
Set the IP Address in the Project	IP address	Configures the IP address. When the project is downloaded, this address is automatically written to the PLC. If the PLC has multiple network ports, their IP addresses must not be on the same subnet.
	Subnet mask	Auto-adapts based on IP address and supports manual modification.

<table>
<tr><th colspan="2">Parameter</th><th>Description</th></tr>
<tr><td rowspan="2"></td><td>Enable gateway</td><td>After checked, the router function is enabled, and the gateway address can be set. Checked by default.

NOTE
◦ For multi-port devices, only one default gateway can be enabled at a time.
◦ Devices and firmware versions that support enabling the gateway function are as follows:
▪ EVO500/EVO700: V1.7 and above.
▪ EVO400: V1.5 and above.
▪ APC series: V1.7 and above.
▪ EVO800: V1.2 and above.</td></tr>
<tr><td>Gateway</td><td>The gateway and device IP address must be in the same network segment.</td></tr>
<tr><td colspan="2">Obtain IP Address Dynamically via DHCP</td><td>The controller can dynamically obtain an IP address through a DHCP server. If there is no DHCP server in the network, the IP address is empty (0.0.0.0).
When DHCP acquisition fails, the solution is as follows:
◦ For single-port devices: Connect the device via USB, select "Set IP address in the project", and download the configuration.
◦ For multi-port devices: Select "Set IP address in the project" and download the configuration through other available network ports of the device.

NOTE
◦ After enabling this option, it takes approximately half a minute to take effect; please wait during the first project download.
◦ After setting the DHCP mode, if the device's IP address changes due to network changes, a different DHCP server, or network cable unplugging/replugging, the user must either re-power on the device or re-download the project with the DHCP mode enabled to apply the new IP address.</td></tr>
</table>

The device-level firmware versions that support the DHCP dynamic acquisition function are listed in the table below.

Product Series	Firmware Version		
	V1.0 to V1.4	V1.5	V1.6
EVO400 series	Not supported	Supported	Supported
EVO500 series	Not supported	Supported	Supported
EVO700	Not supported	Supported	Supported
APC520 series	Not supported	Supported	Supported
EVO760C series	Not supported	Not supported	Supported
EVO780C series	Not supported	Not supported	Supported
APC760C series	Not supported	Not supported	Supported



In PROFINET communication mode, the IP address of the PROFINET slave device must not be the same as that of the main device; otherwise, the actual IP address of the main device will become invalid.

Network features of different address setting method

The relationship between the three Ethernet address setting methods and network function is shown in the table below:

Function	Use the IP Address Directly from the Device	Set the IP Address in the Project	Obtain IP Address Dynamically via DHCP
Network setting	Network-based IP setting supported	Network-based IP setting not supported	Network-based IP setting not supported
Instruction setting	Network-based IP setting supported	Network-based IP setting not supported	Network-based IP setting not supported
HMI tag connection	HMI requires manual communication IP configuration	IP is automatically updated on the HMI after modification	HMI requires manual communication IP configuration
C2C			
Redundant PLC	Network-based IP setting supported	Network-based IP setting not supported	Network-based IP setting not supported
Mode Switching	A full download is triggered after mode switching.		

10.6.14.5 OPC UA Server Configuration

10.6.14.5.1 Overview

The OPC UA protocol is divided into two parts: OPC UA server and client.

The OPC UA server provides standardized data access services across platforms and systems, ensuring secure and reliable data exchange from sensors to the cloud.

The core functions of the OPC UA server include the following:

- It provides a unified data model that supports the description of complex information structures, enabling the transmission of real-time data, historical data, and rich metadata.
- It implements secure communication mechanisms, supporting various authentication and encryption modes, including anonymous access, username and password authentication, and certificate authentication.
- It has a wide range of applications, suitable for industrial automation, the Internet of Things (IoT), smart manufacturing, and other fields, facilitating interoperability between different levels of equipment, such as PLCs, SCADA systems, MES systems, and ERP system integration.
- It allows any device to act as a data provider, enabling seamless data sharing among products from different manufacturers, enhancing the transparency and efficiency of the entire production process.
- By being built-in or attached to devices and PLCs, or running on separate computers, the OPC UA server acts as the central hub for industrial data, promoting the informatization and intelligence of enterprises.

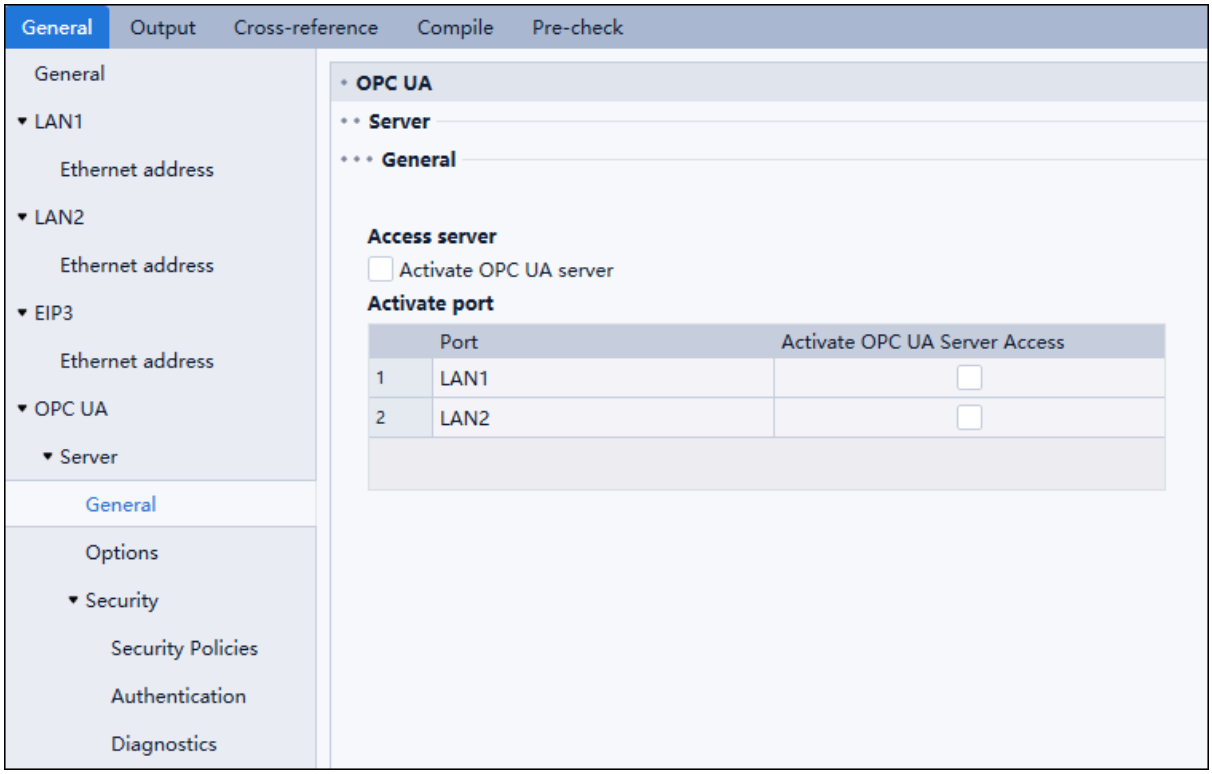
10.6.14.5.2 Configuring OPC UA Server

Prerequisites

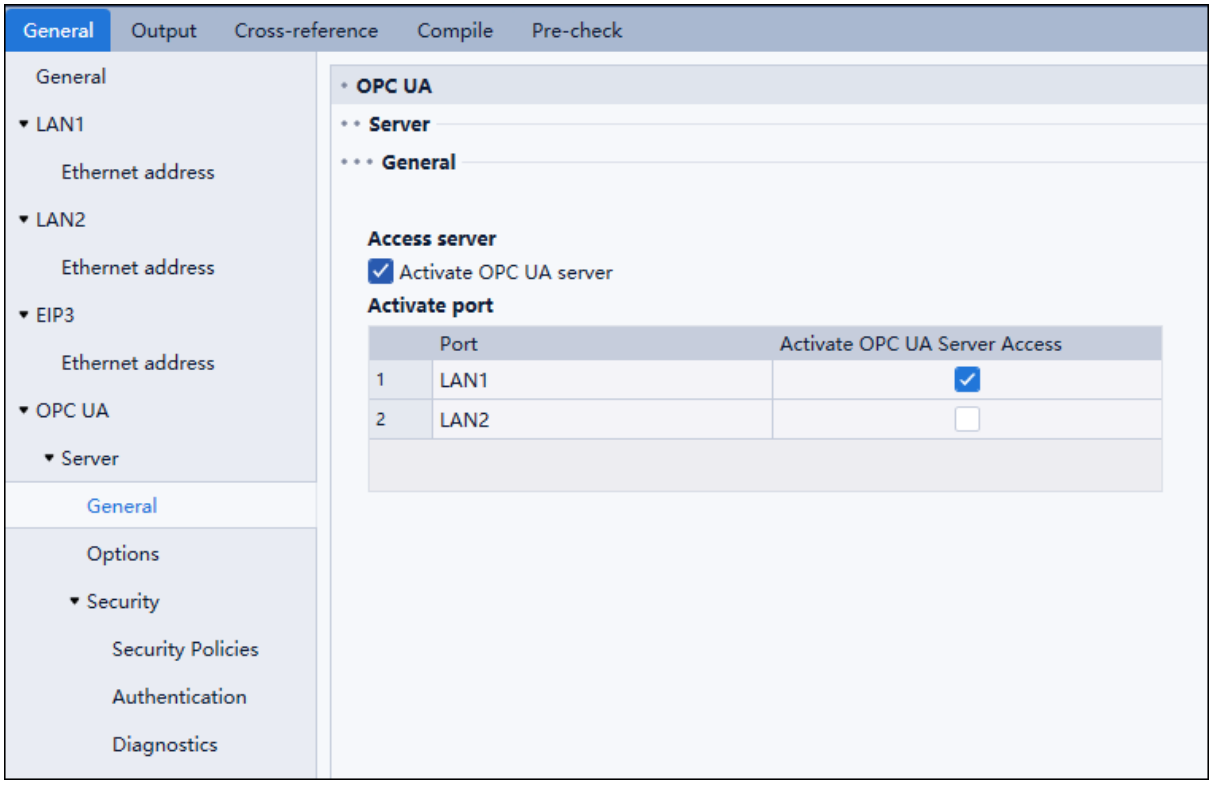
A PLC device has been added.

Procedure

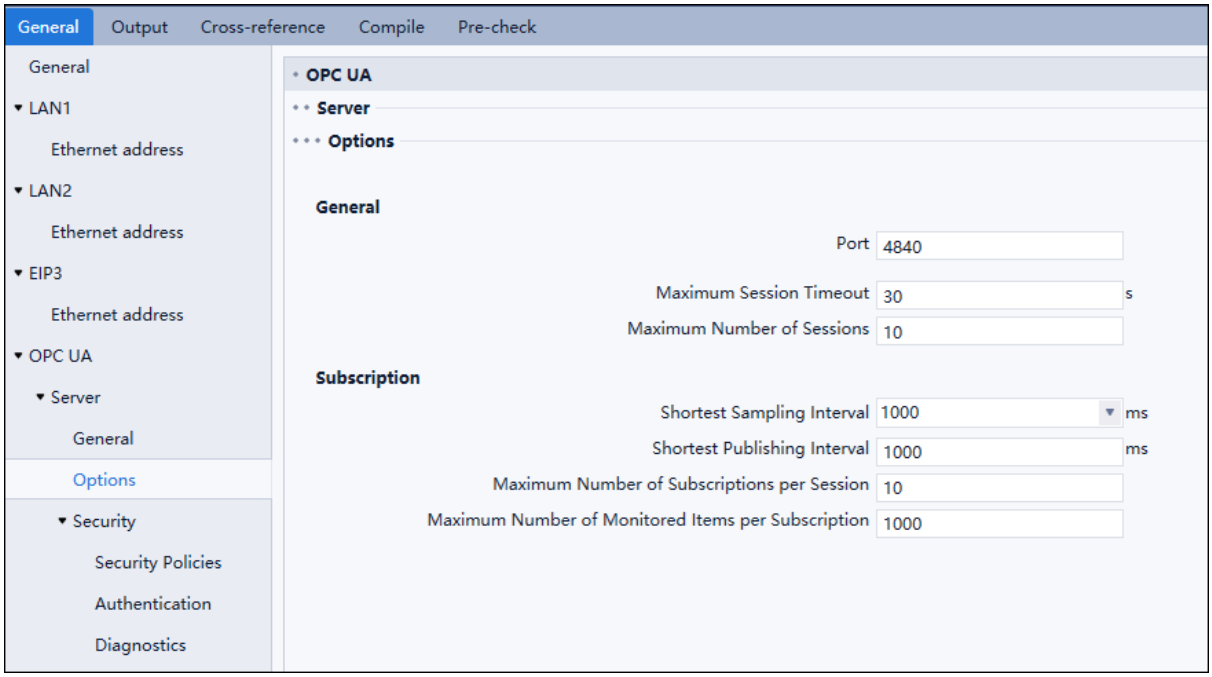
1. Double-click "Network configuration" in the project tree to open the network configuration view. Select the corresponding PLC device, and the "General" tab appears in the monitoring area, where you can configure the OPC UA server.



2. Check "Activate OPC UA server" to configure and enable the OPC UA server, and activate the access port in the port list.



3. The OPC UA server parameters can be configured under "General > OPC UA > Server > Options".



The parameter configuration is described in the table below.

Parameter	Note	Setting Method
Port	Sets the port number for the OPC UA server.	Default value: 4840; range: 1024 to 49151. The following special ports cannot be set: ◦ Modbus TCP: 502 ◦ FINS: 9600 ◦ EtherNet/IP: 44818, 2222 ◦ EtherCAT: 34980 ◦ Label communication: 12939, 12940

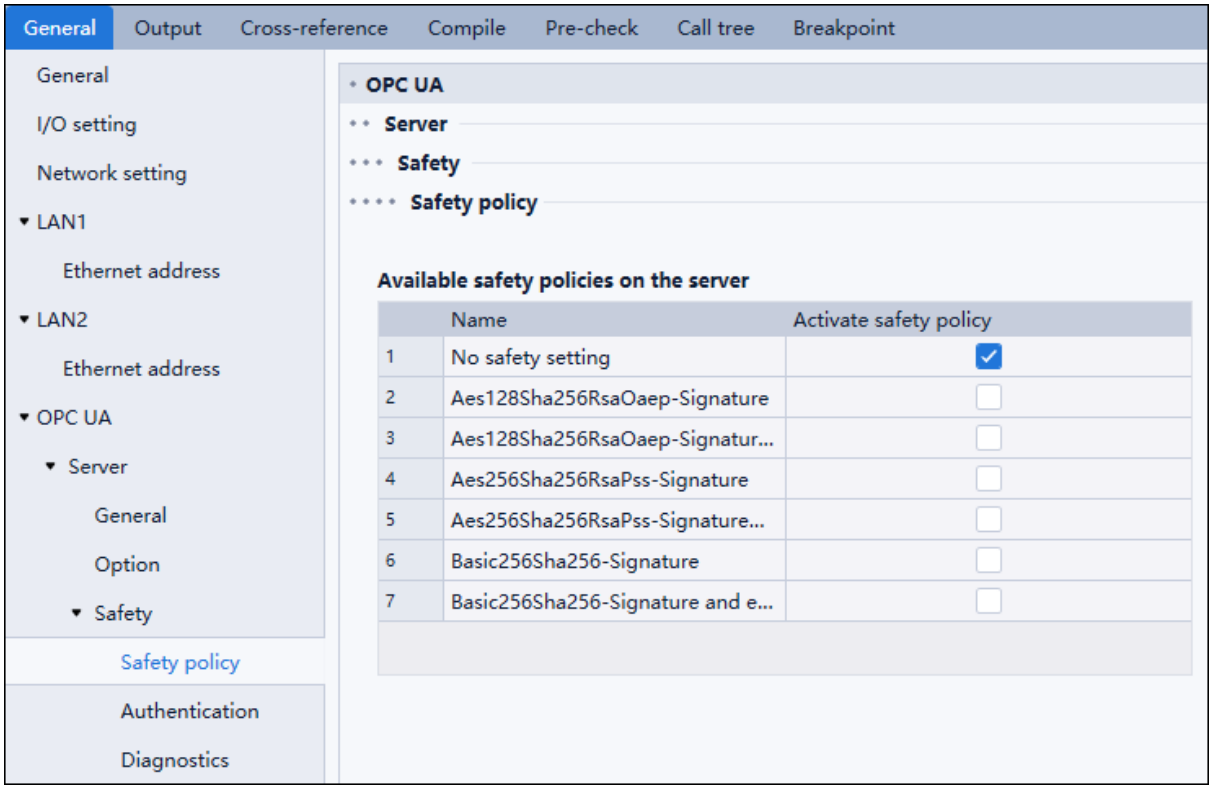
Parameter	Note	Setting Method
Maximum session timeout	Sets the maximum duration before the OPC UA server closes a session if no data exchange occurs.	Default value: 30s; range: 5s to 600000s.
Maximum number of sessions	Sets the maximum number of sessions that the OPC UA server can start and operate simultaneously, which depends on the performance of the PLC.	Default value: 10; range: 1 to 20.
Shortest sampling interval	Sets the time interval at which the OPC UA server records PLC variable values and compares them with previous values to check for changes.	Default value: 1000 ms. Range for EVO400/EVO500 series: 100 ms, 250 ms, 500 ms, 1000 ms, 5000 ms, 10000 ms. EVO700 series value range: 100 ms, 250 ms, 500 ms, 1000 ms, 2000 ms, 5000 ms, 10000 ms. EVO800 series value range: 100 ms, 250 ms, 500 ms, 1000 ms, 2000 ms, 5000 ms, 10000 ms.
Shortest publishing interval	Sets the interval loop rate at which the OPC UA server subscription channel publishes notifications to the client.	Default value: 1000 ms; range: 500 ms to 20000000 ms.
Maximum number of subscriptions per session	Sets the maximum number of subscriptions allowed in each session.	Default value: 10; range: 1 to 20.
Maximum number of monitored items per subscription	Sets the maximum number of monitored elements that the server assembles into a notification package for each subscription. As monitoring consumes resources, this item depends on the PLC performance.	Default value: 1000. EVO400/EVO500 series value range: 100 to 5000. EVO700 series value range: 100 to 10000. EVO800 series value range: 100 to 10000.

The total number of all monitored items for EVO800 series products does not exceed 20000. At the same time, the number of monitorable items is affected by the sampling interval; under the same number of monitored items, the smaller the sampling interval, the greater the system load. To ensure system functionality and execution efficiency, it is recommended to control the number of monitored items based on the sampling interval within the following quantities.

Sampling Interval	Maximum Number of Monitored Items
100 ms	1000

Sampling Interval	Maximum Number of Monitored Items
250 ms	2500
500 ms	5000
1000 ms	10000

4. The encryption policy of the OPC UA server can be configured under "General > OPC UA > Server > Safety > Safety policy" in the monitoring area.



The security policy is the message security mechanism allowed during communication between the OPC UA server and the client, implemented by setting the message signature, message encryption, and the algorithms for signing and encryption. You can select no security policy, or select one or more security policies. The specific security policy information is shown in the table below.

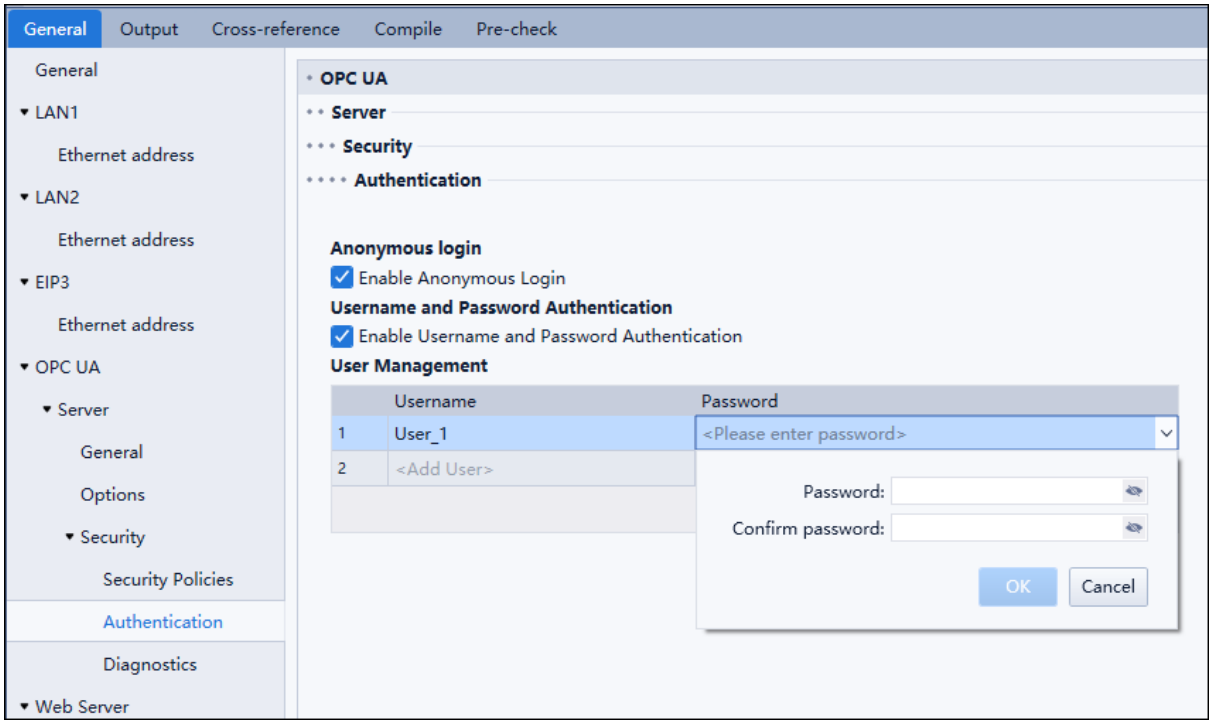
Parameter	Description
No security settings	No security policy connection and no security settings
Aes128Sha256RsaOaep - Sign	Use signing to ensure data integrity (prevent tampering, etc.) Use Aes128Sha256RsaOaep as the signing algorithm
Aes128Sha256RsaOaep - Sign and encryption	Use signing and encryption to ensure data integrity (prevent tampering, etc.) and reliability (prevent eavesdropping, etc.) Use Aes128Sha256RsaOaep as the signing and encryption algorithm

Parameter	Description
Aes256Sha256RsaPss - Sign	Use signing to ensure data integrity (prevent tampering, etc.) Use Aes256Sha256RsaPss as the signing algorithm
Aes256Sha256RsaPss - Sign and encryption	Use signing and encryption to ensure data integrity (prevent tampering, etc.) and reliability (prevent eavesdropping, etc.) Use Aes256Sha256RsaPss as the signing and encryption algorithm
Basic256Sha256 - Sign	Use signing to ensure data integrity (prevent tampering, etc.) Use Basic256Sha256 as the signing algorithm
Basic256Sha256 - Sign and encryption	Use signing and encryption to ensure data integrity (prevent tampering, etc.) and reliability (prevent eavesdropping, etc.) Use Basic256Sha256 as the signing and encryption algorithm

NOTE

- Depending on the hardware device, supported security policies vary, with different policies providing different security encryption methods. For detailed security policies of different products, refer to the relevant product manuals.
- It is recommended to use no security policy during initial commissioning. After commissioning, select a security policy compatible with the device or factory's security concept, and disable other security policies.

5. The user authentication of the OPC UA client can be configured under "General > OPC UA > Server > Safety > Authentication" in the monitoring area.



The parameters for authentication are described in the table below.

Parameter	Description	Setting Method
Enable anonymous login	With anonymous login enabled, users do not need to prove their identity (anonymous access), and the OPC UA server will not check the authorization of the client user.	By default, anonymous login is enabled.
Enable username and password authentication	By enabling this item, users must prove their identity (non-anonymous access), the OPC UA server will check whether the client user has the permission to access the server, and authenticate through a username and correct password.	This item is disabled by default. After checking "Enable username and password authentication", you can edit user information in the "User management" list below.
Add user	Adds a new user.	Up to 100 users can be added. You can set the username and password.

NOTE

It is recommended to enable anonymous login only during the initial commissioning phase. After commissioning, enable "Username and password authentication" to ensure communication security.

Result

After configuring the OPC UA server parameters, download and start the device (for details, see 10.8 PLC Download). Once the PLC device is online, start the OPC UA server.

Precautions

- When the PLC runs the OPC UA server for the first time, or for the first time after IP address modification, the OPC UA server certificate file needs to be regenerated, which requires a waiting time of 1s to 2s.
- When the OPC UA server is running, if the IP address of the PLC is modified via the background or system variables, it is necessary to restart the server, or power off and on the device for the modification to take effect.

10.6.14.5.3 OPC UA Diagnostics

iFA software supports OPC UA self-diagnosis functions, which can be used to view fault information in OPC UA connections and diagnose devices and networks, making it easier to locate issues.

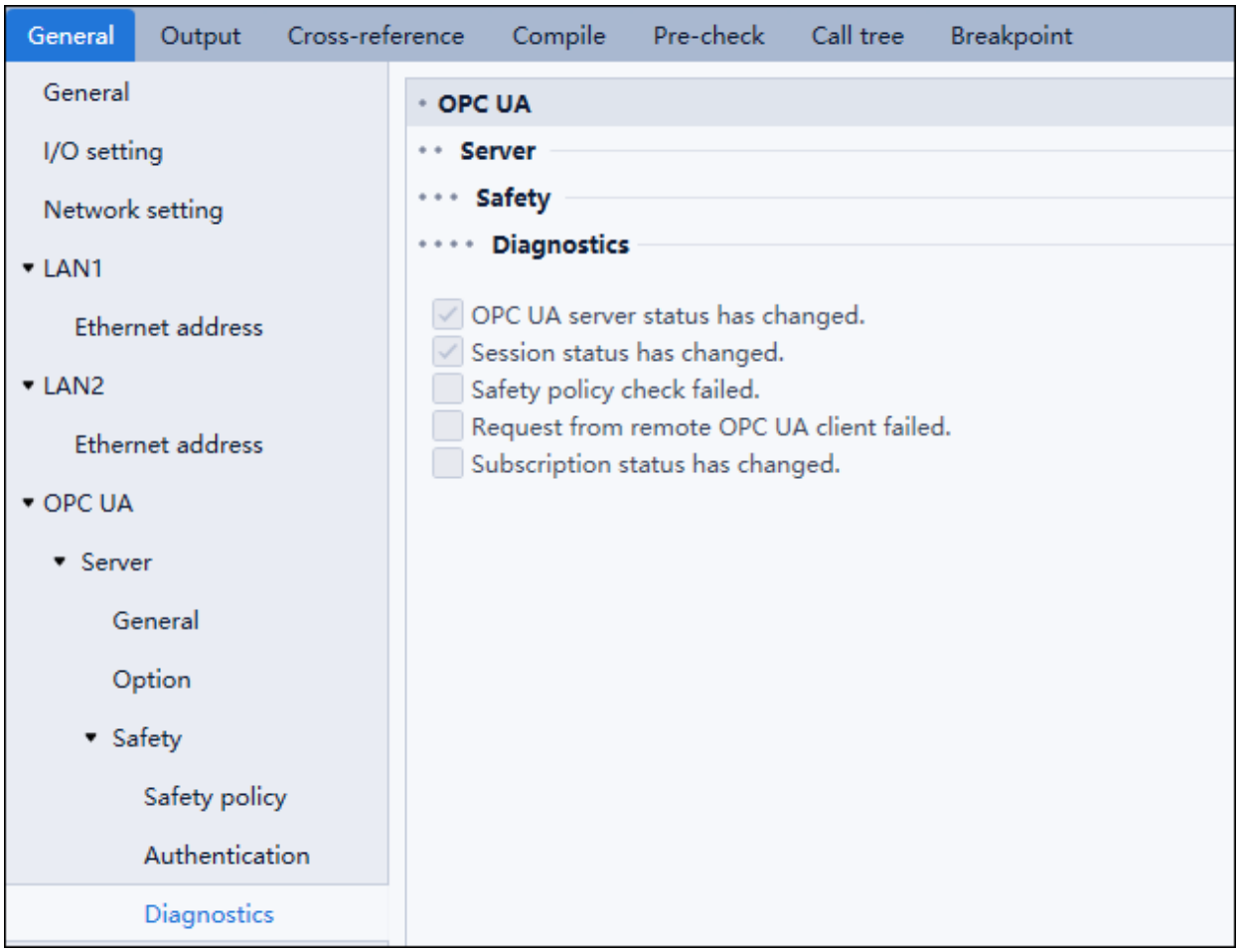
Diagnostic function configuration

Prerequisites

OPC UA server has been configured.

Function entry

The diagnostic function can be configured under "General > OPC UA > Server > Safety > Diagnostics" in the monitoring area.



The diagnostic scenarios include the following:

- OPC UA server status has changed: Supports diagnosing changes in server operation, stop, and timeout statuses, and reporting with diagnostic code.

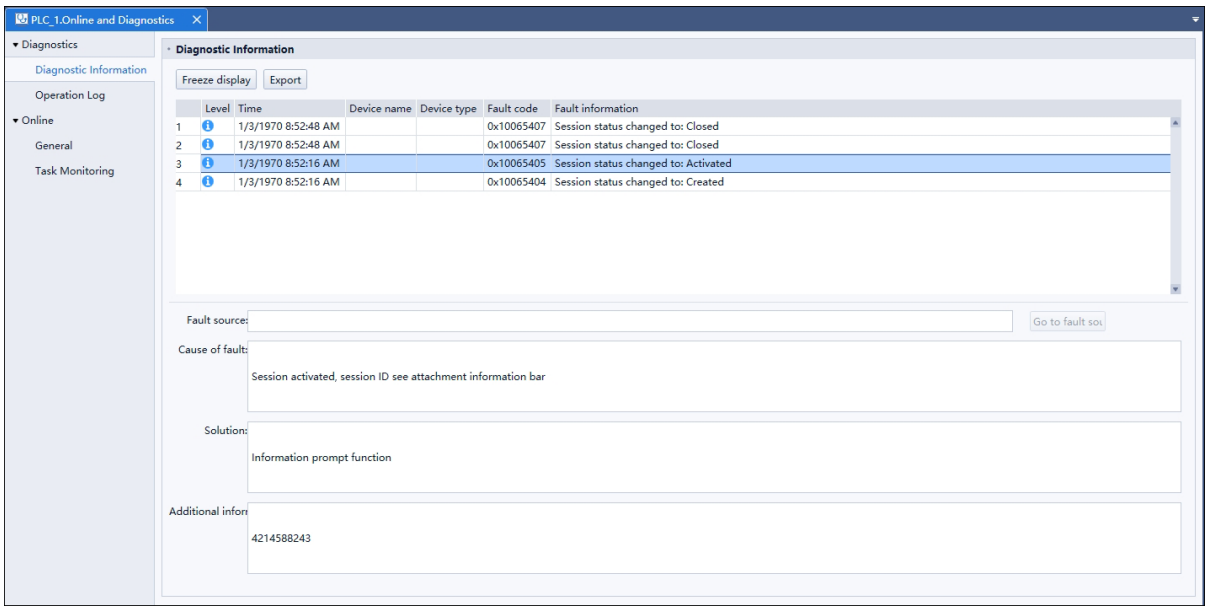
- Session status has changed: Supports diagnosing session status changes for connected sessions, including creation, activation, timeout, and closure, and reporting with diagnostic code.
- Security policy check failed: This diagnostic function is not currently supported.
- Remote OPC UA client request failed: Supports diagnosing failures of OPC UA client requests. Upon a failure, the count of rejected requests is increased in the session tab and the diagnostic code is reported.
- Subscription status has changed: Clients and servers establish subscriptions during sessions. This function supports diagnosing subscription status changes, and reporting with diagnostic code.

Viewing diagnostic code

Function entry

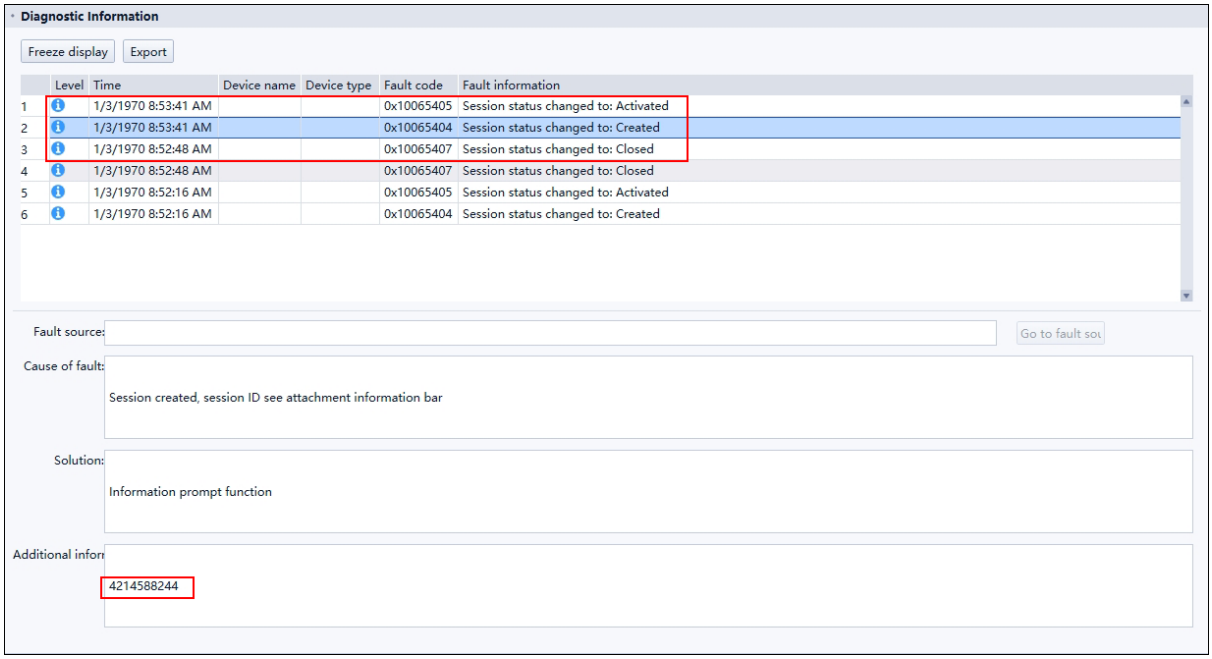
Double-click "Online and diagnostics" in the project tree to open the "Online and diagnostics" interface.

- Viewing diagnostic information
  The diagnostic information of the OPC UA server can be viewed under "Diagnostic > Diagnostic information", including information prompts and fault prompts. Information prompts include session information and subscription information prompts. Click "Freeze display" in the upper left corner to view specific session and subscription information, which allows you to see session and subscription status transitions. Additional information below include the session ID and subscription ID. Click the specific fault information to view the cause and solution of the fault.



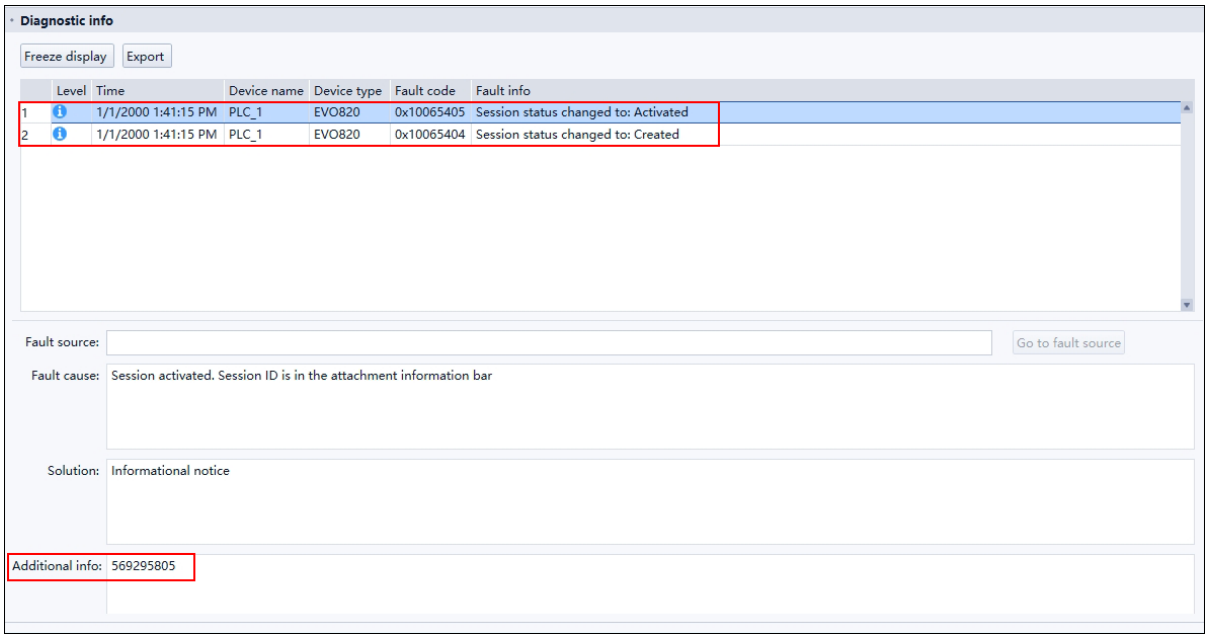
Session status:

Session status displays transitions between session creation, activation, and closure statuses, with specific session IDs attached in the additional information below.



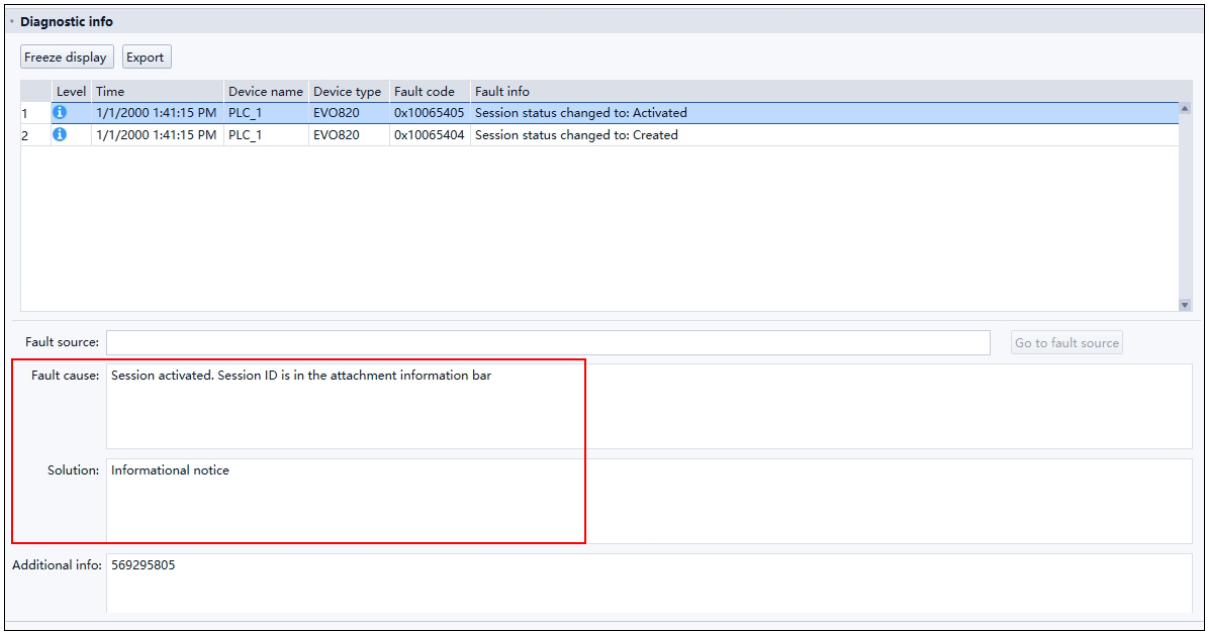
Subscription status:

The subscription status displays the creation and normal status of the subscription, and the corresponding subscription ID will be shown in the information prompt below.



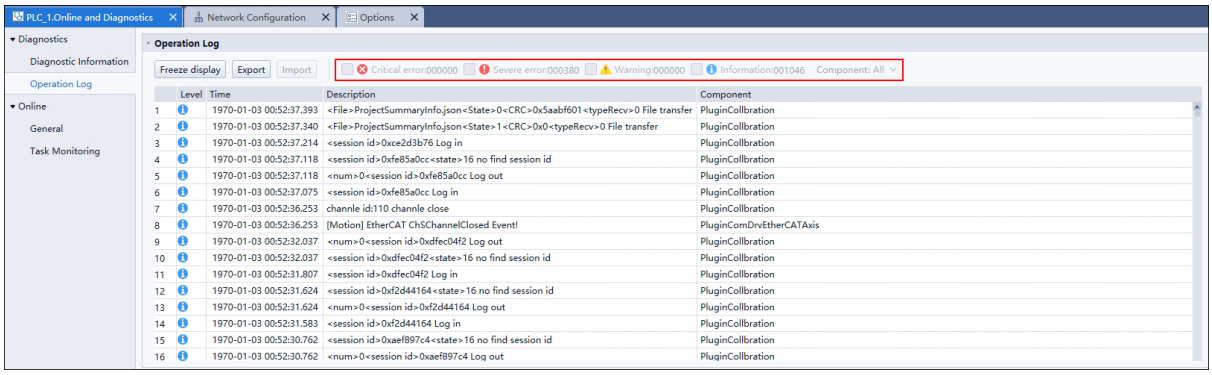
Fault status:

Displays specific information, causes, and corresponding solutions of faults.



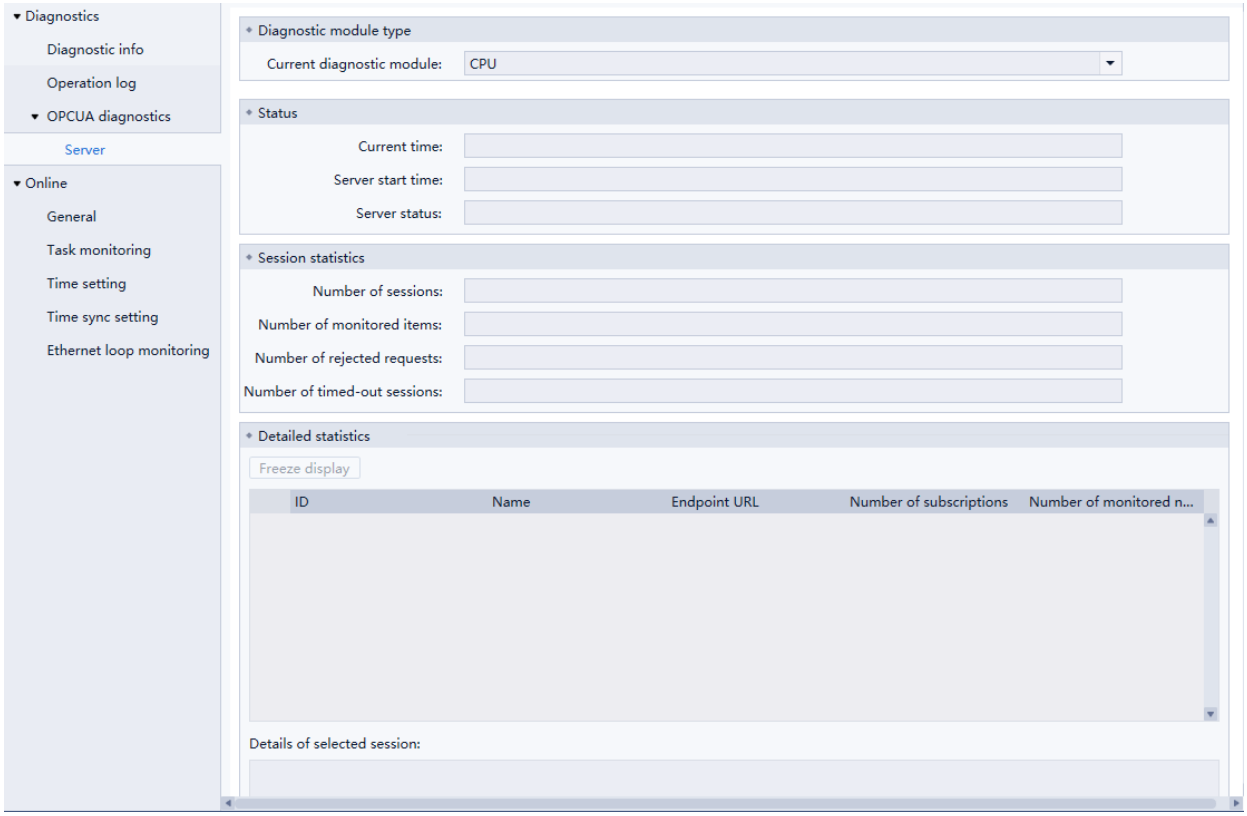
- Viewing operation logs

  Switch the component to PluginOPCUAServer to filter and view the operation logs created by the OPC UA server. The statistics above show the number of critical faults, severe faults, warnings, and information messages. Select a specific fault type to view the corresponding fault logs.



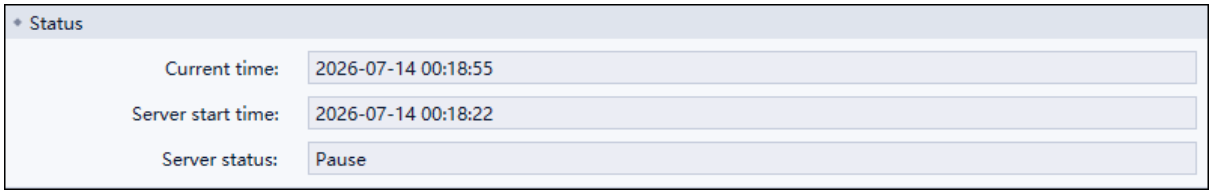
Viewing server status monitoring and self-diagnostic information

Select the module to be diagnosed from the drop-down list of "Current diagnostic module".



- Status

  Supports monitoring the status of the OPC UA Server. Includes the current enabled status of the server, current time, and server start time.



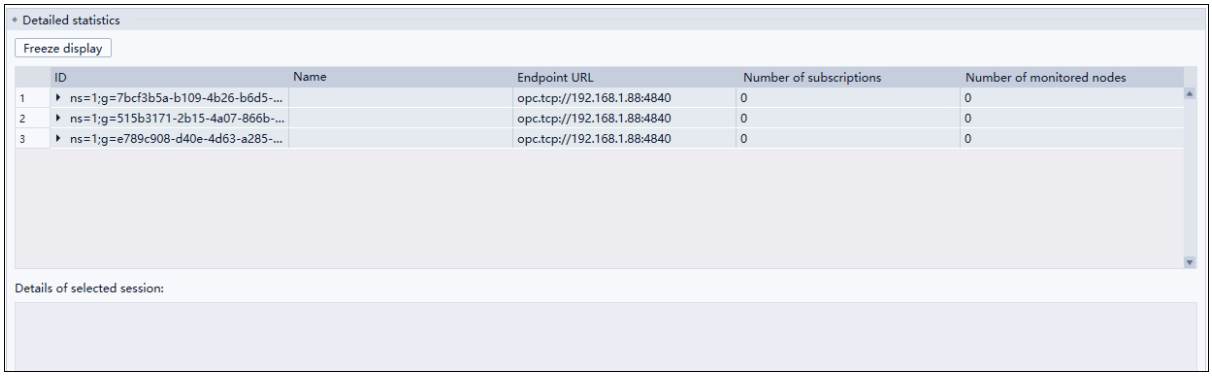
- Session statistics

  Supports viewing the number of sessions, monitored items, rejected requests, and timed-out sessions on the interface.



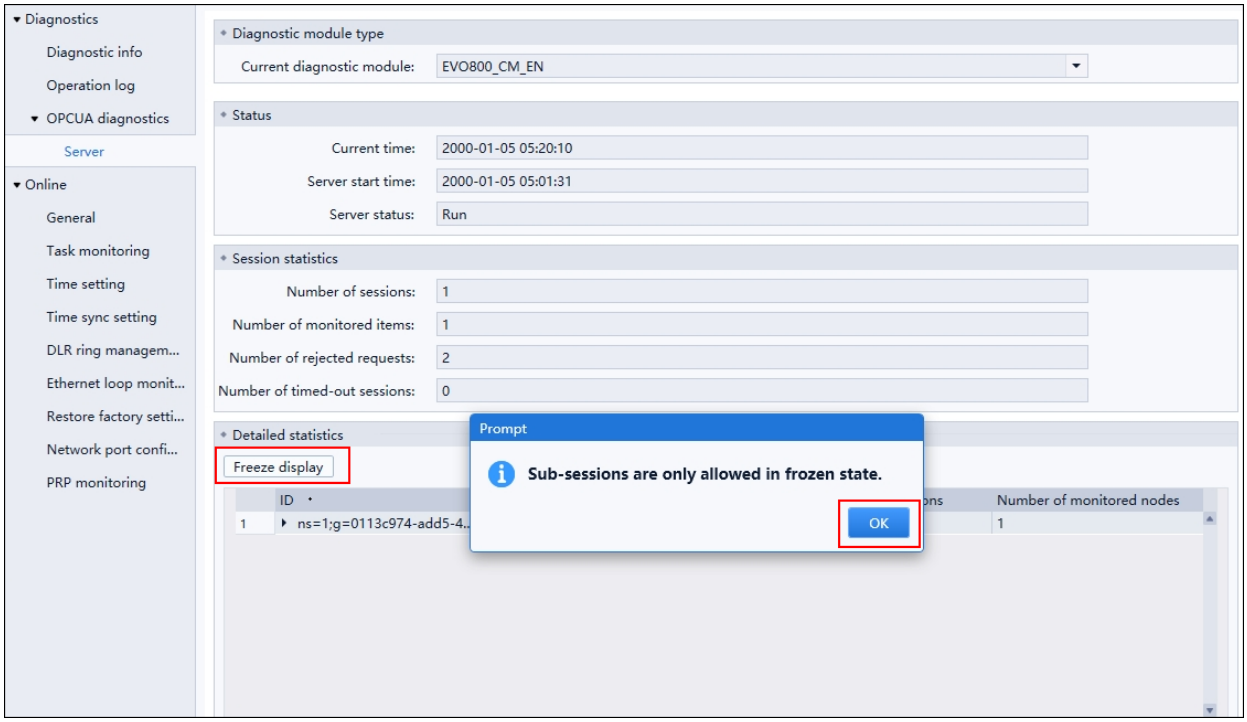
- Details

  Displays specific session/subscription diagnostic information, including the IDs, names, and the number of subscriptions and monitored items in the current established sessions/subscriptions.

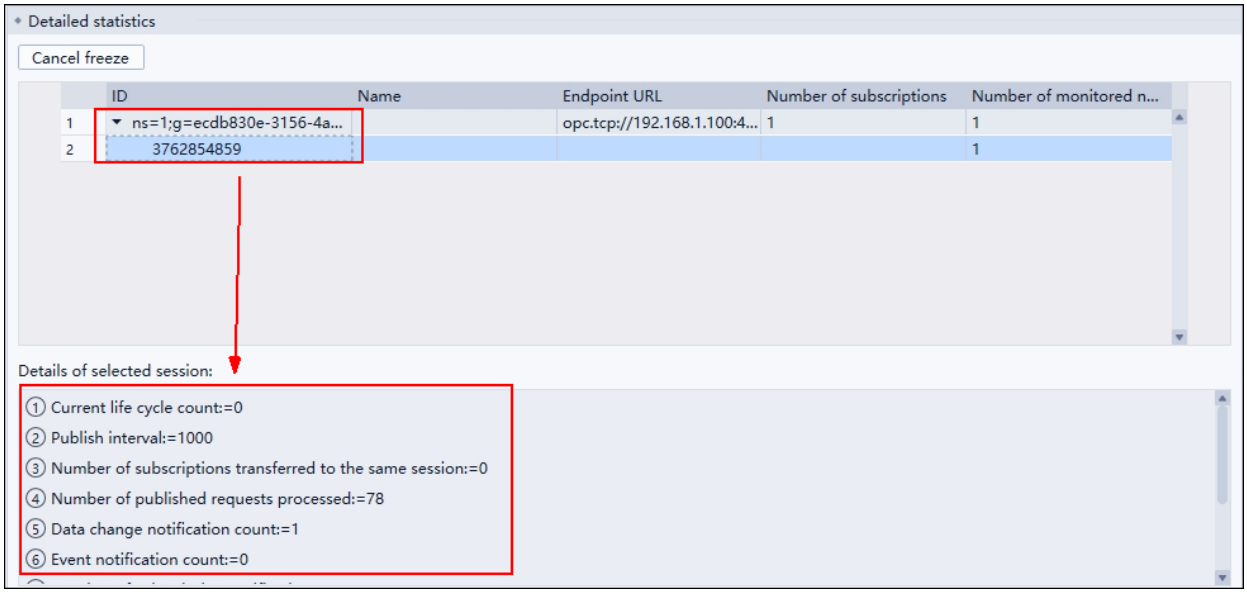


Precautions

To view detailed session information, freeze the current window to expand the sub-session information card for details.



After freezing, select a detailed session to view its description in the selected session details below.



10.6.14.5.4 Fault Codes

Fault Code	Level	Description	Cause
0x10065000	Information	Normal connection with no fault	/
0x10065001	Fatal	Out of memory	Insufficient memory
0x10065002	Critical	Socket failed to open	Insufficient permissions, insufficient system resources, occupied port number, and so on
0x10065003	Critical	Unspecified IP	Socket bound without specifying an IP
0x10065004	Critical	Socket failed to be bound	Insufficient permissions, insufficient system resources, occupied IP address or port number, and so on
0x10065006	Critical	hostname too long	hostname length exceeds 512 bytes.
0x10065008	Critical	Invalid EndpointUrl	The configured server endpoint Url is invalid.
0x10065009	Warning	Empty security policy	Security policy is not correctly configured.
0x1006500A	Warning	Failed to set security policy	Security policy is not correctly configured.
0x1006500B	Warning	Mismatch between security mode and security protocol	When the security mode is not None, the security protocol is None (cannot be None).
0x1006500D	Alarm	OPC UA network anomaly	OPC UA network connection is disconnected, or the remote network is unreachable.
0x1006500E	Warning	Port settings out of limit	In the issued configuration parameters, the port setting exceeds the limit range and has been set to the default value.
0x1006500F	Warning	Maximum session timeout out of limit	In the issued configuration parameters, the maximum session timeout exceeds the limit range and has been set to the default value.
0x10065010	Warning	Maximum number of sessions out of limit	In the issued configuration parameters, the maximum number of sessions exceeds the limit range and has been set to the default value.
0x10065011	Critical	Sampling interval out of limit	In the issued configuration parameters, the sampling interval exceeds the limit range.
0x10065012	Warning	Publishing interval out of limit	In the issued configuration parameters, the publishing interval exceeds the limit range and has been set to the default value.

Fault Code	Level	Description	Cause
0x10065013	Warning	Maximum number of subscriptions per session out of limit	In the issued configuration parameters, the maximum number of subscriptions per session exceed the limit range and have been set to the default value.
0x10065014	Warning	Maximum number of monitored items per subscription out of limit	In the issued configuration parameters, the maximum number of monitored items per subscription exceed the limit range and have been set to the default value.
0x10065015	Fatal	Configuration file not exist	The ini configuration file is missing from the device.
0x10065016	Warning	Server certificate about to expire	The server certificate is about to expire.
0x10065201	Fatal	Out of memory	Insufficient memory
0x10065401	Fatal	Out of memory	Insufficient memory
0x10065402	Warning	Excessive sessions	The number of session connections exceeds the limit.
0x10065404	Information	Session status changed to: Created	The session is created.
0x10065405	Information	Session status changed to: Activated	The session is activated.
0x10065406	Information	Session status changed to: Timeout	The session is timed out.
0x10065407	Information	Session status changed to: Closed	The session is closed.
0x10065601	Fatal	Out of memory	Insufficient memory
0x10065602	Warning	Excessive read/write operations	When reading or writing properties and historical data, the number of read/write requests in a single frame exceeds the upper limit.
0x10065603	Warning	Node value unreadable	When reading the value property of a variable, the read permission of the node is disabled.
0x10065605	Warning	No data in indexRange	When reading node properties, the specified indexRange has no data.
0x10065606	Warning	Node value unwritable	When writing the value property of a variable, the write permission of the node is disabled.

Fault Code	Level	Description	Cause
0x10065801	Fatal	Out of memory	Insufficient memory
0x10065802	Warning	Excessive subscriptions	When creating or transferring subscriptions, the number of subscriptions in a single session exceeds the limit.
0x10065803	Information	Subscription status changed to: Creating	The subscription is normal, whose ID is displayed in the attachment information section.
0x10065804	Information	Subscription status changed to: Normal	The subscription is normal, whose ID is displayed in the attachment information section.
0x10065805	Information	Subscription status changed to: Late	The subscription is overdue, whose ID is displayed in the attachment information section.
0x10065806	Information	Subscription status changed to: Closed	The subscription is closed, whose ID is displayed in the attachment information section.
0x10065807	Warning	Excessive monitored nodes	The number of monitored nodes exceeds the limit.
0x10065C01	Fatal	Out of memory	Insufficient memory

10.6.14.6 SFTP Server Configuration

By configuring an SFTP server, users can transfer files with the PLC using standard SFTP clients or commands. This function is primarily used for remote user program upgrades.

Restrictions

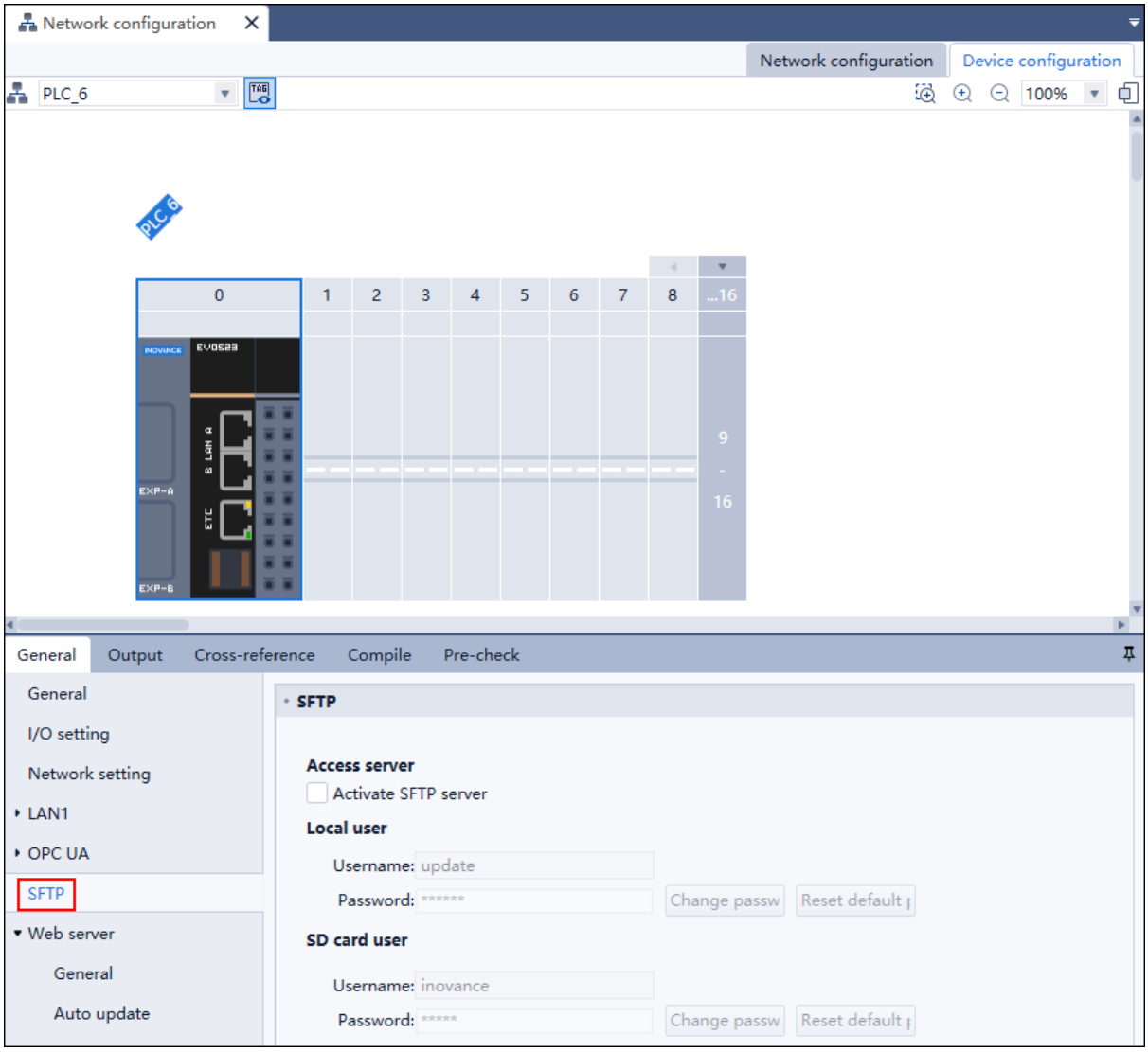
SFTP server configuration is only applicable to EVO500, EVO700, EVO760C, EVO780C, and APC series PLCs, and requires firmware version V1.6 or above.

Prerequisites

A PLC device has been added.

Procedure

1. Double-click "Device configuration" in the left project tree and go to the "Network configuration" tab.
2. Select the PLC device diagram. In the "General" tab of the monitoring area below, select "SFTP" to open the SFTP server configuration interface.



3. Check "Activate SFTP server" to enable the SFTP server function of the PLC. Two built-in accounts are provided:

Parameter	Description
Local account	Has read/write access to the PLC storage path and can be used for user program upgrades. Default username and password are as follows: ◦ Username: update (non-modifiable) ◦ Password: 123456 (modifiable)
SD card account	Has read-only access to the SD card path, mainly for file transfer. Default username and password are as follows: ◦ Username: inovance (non-modifiable) ◦ Password: admin (modifiable)

4. (Optional) The password for either account can be changed or reset as follows:
   - Click "Change password". In the pop-up dialog, enter the current password and the new password, then click "OK".
     The new password must contain at least one digit, one uppercase letter, and one lowercase letter.



- Click "Reset default password", enter the current password in the dialog box, and click "OK" to restore the account password to default.

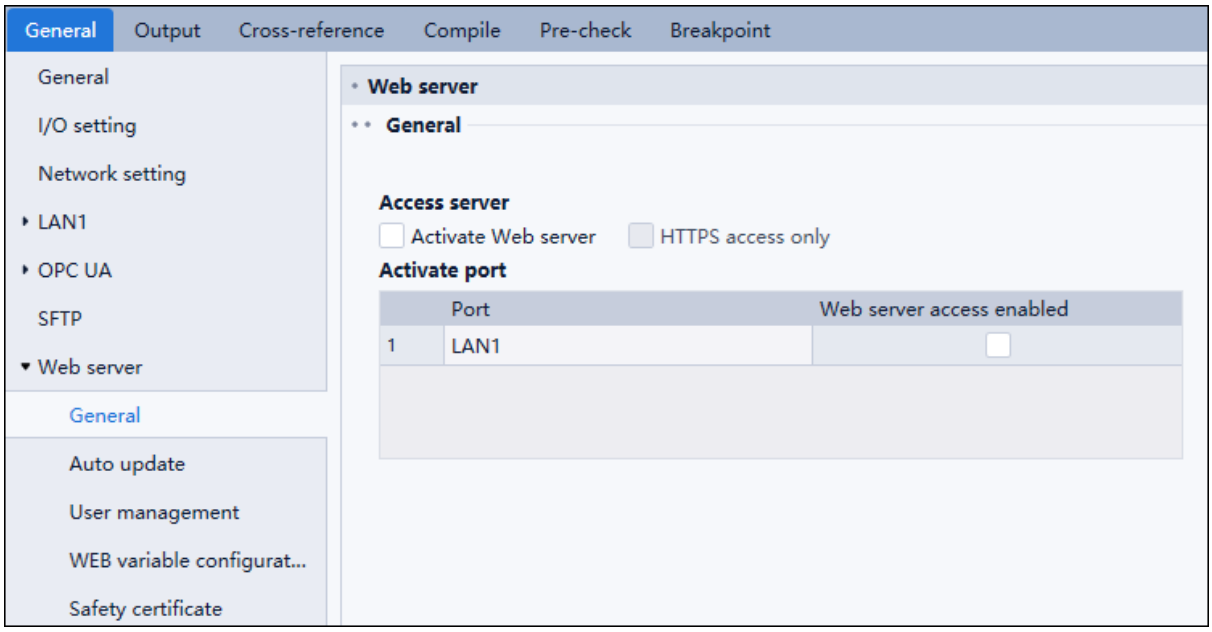
10.6.14.7 Web Server Configuration

Prerequisites

A PLC device has been added.

Procedure

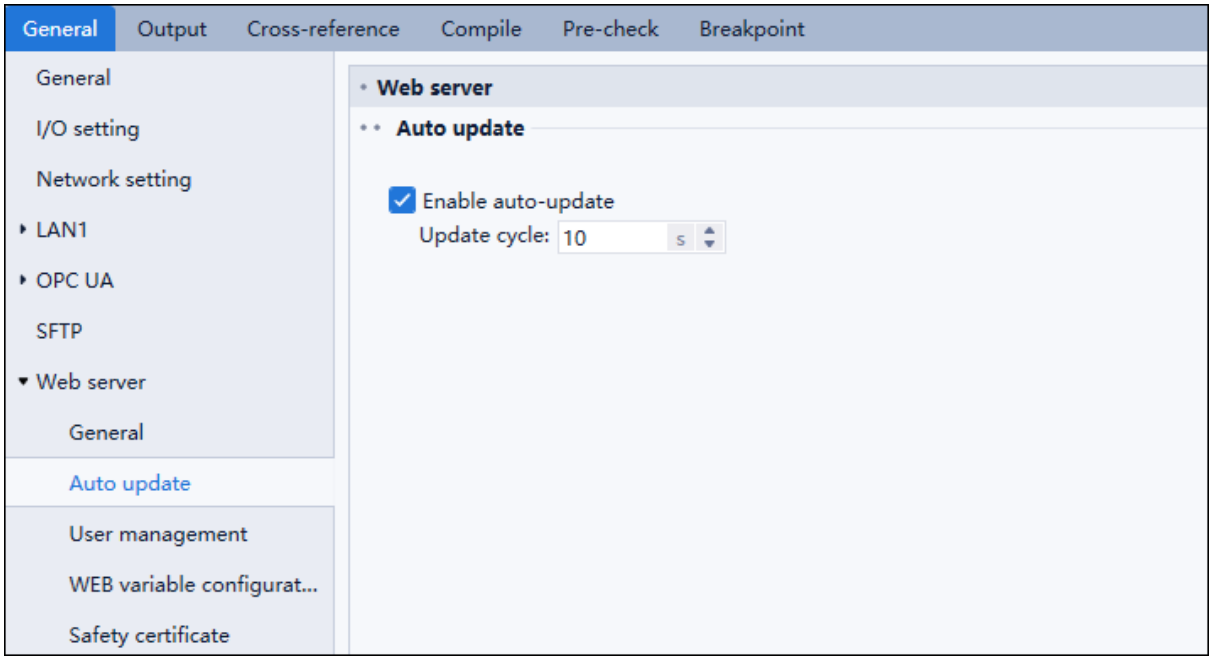
1. Open the Web configuration interface.
   Double-click "Network configuration" or "Device configuration" in the left project tree to open the network configuration interface. Click the PLC device and the Web server can be configured under "General > Web server" in the monitoring area below.



   When the Web is activated, users can log in to the Web server via "http://<address>". For details, see 10.1 1.12 Web Diagnostics and Monitoring.

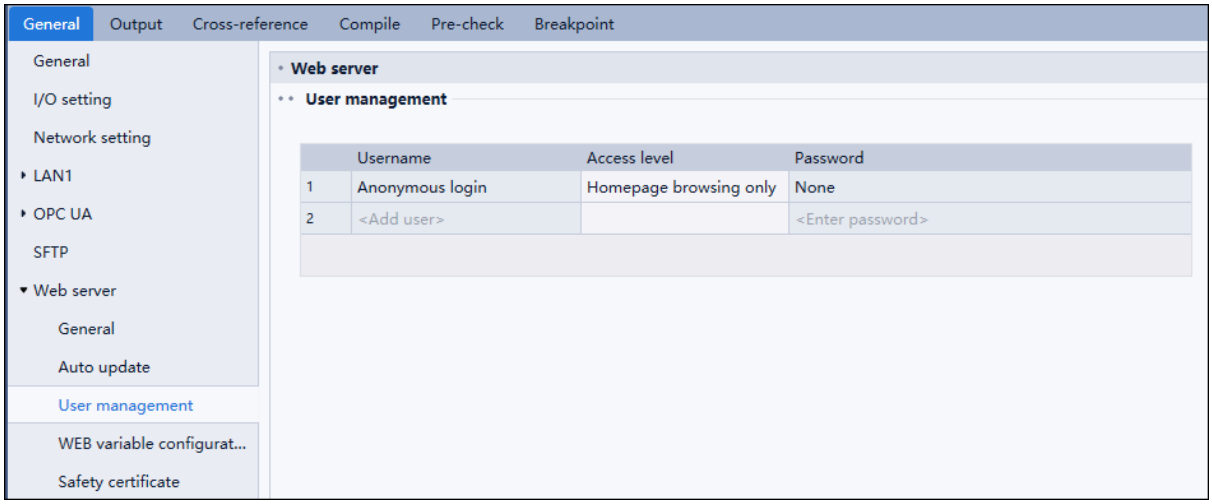
2. Activate the Web server.
   Check "Activate Web server" and activate the access port in the port list. If "HTTPS access only" is checked, only HTTPS access is supported.

3. Configure automatic updates.
   In the "Auto update" tab, check "Enable auto-update" and the Web server will update data in real-time according to the set "Update cycle".



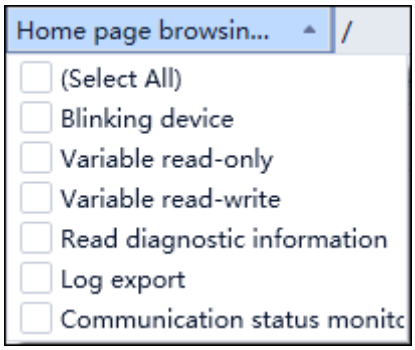
4. Configure user management.

   By default, all users can log in anonymously with permissions limited to homepage browsing. The anonymous permissions can be customized by users.



In the "User management" tab, user accounts can be added to increase access permissions; the number of users that can be created is 1 to 20.

Parameter	Note
Username	Required field, default name is "User_1". Composed of letters (case-insensitive), numbers, Chinese characters, and special characters. Supported special characters include ~`!@#$ %^&*()-_=+[]{}|\;:,.<>/?!　（）、。：；“” The maximum character length is 128.

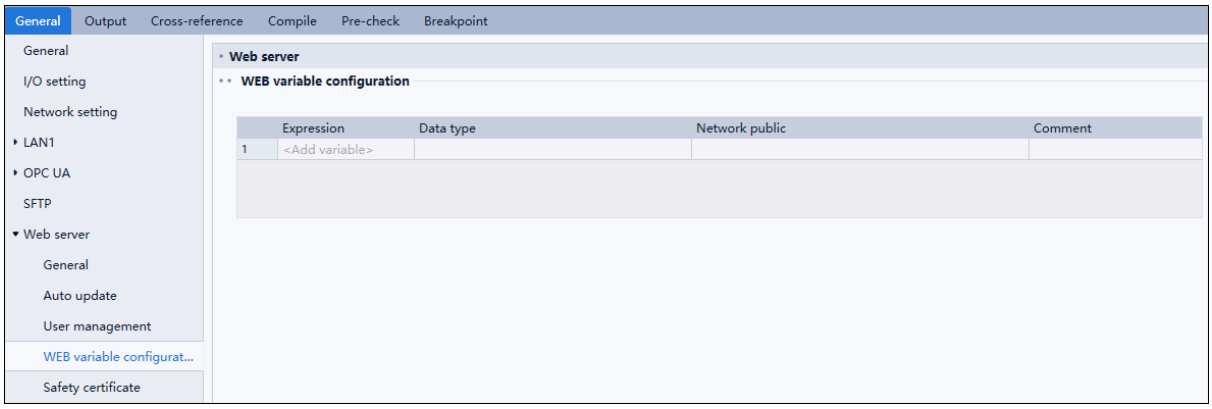
Parameter	Note
Access level	The Web interface only supports the corresponding access levels. The options in the drop-down list of "Access level" are as follows: 
Password	Composed of letters (case-sensitive), numbers, Chinese characters, and special characters. Supported special characters include ~`!@#$%^&*()-_=+[]{}|\;:,.<>?/ The password must contain numbers, uppercase letters, and lowercase letters. Character length: 8 to 32.

5. Configure Web variables.

   When configuring variables in the "Web variable configuration", full name addition of variables is supported. Click "..." to add variables in the pop-up input assistant.

   The software interface supports displaying properties of variables, such as expressions, data types, network public property, and annotations.

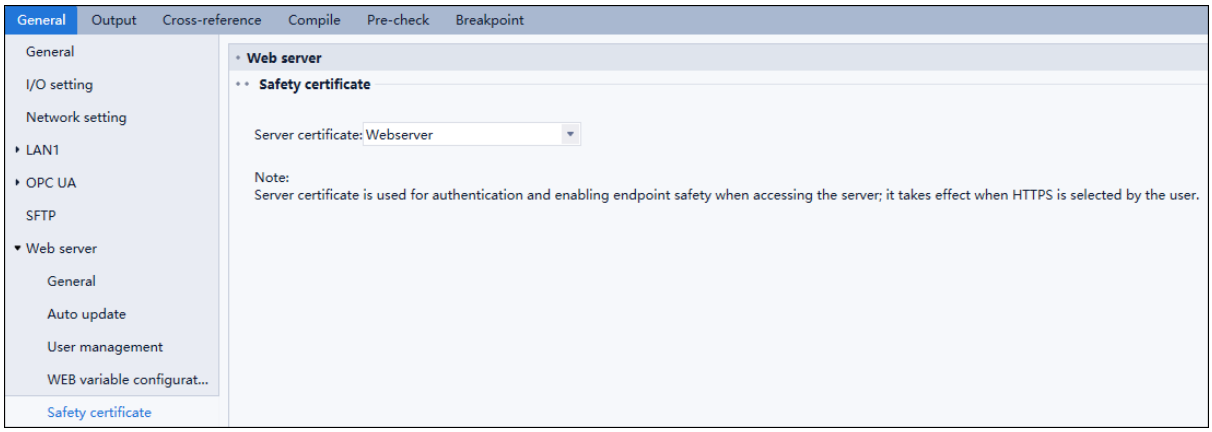
   Only read-write and read-only variables are supported, with a limit of 100 variables. For variables containing sub-elements such as arrays and structures, their sub-elements are also counted.



6. Configure the security certificate.

   The server certificate is used for identity verification when users access the server.

The default name for the certificate is Webserver. Users can select the certificate from a drop-down box.



NOTE

The EVO500 series PLC only supports Web server functionality from version V1.4.2.5 and above.

10.6.15 Ethernet Expansion Module Property Configuration

Ethernet expansion module property configuration is downloaded to a PLC module via a PLC project. The PLC module then synchronizes this configuration to the Ethernet expansion module via rack communication.

CAUTION

iFA Evolution does not support downloading directly to the Ethernet expansion module itself.

10.6.15.1 Setting Ethernet Address

This section describes how to set IP addresses for the LAN1 and LAN2 ports of the Ethernet expansion module via its properties. IP addresses can also be configured through the toolbar: "Online > Network setting" (see 10.11.5 Network Setting for details). After the device goes online, the changes are automatically synced back to this interface.

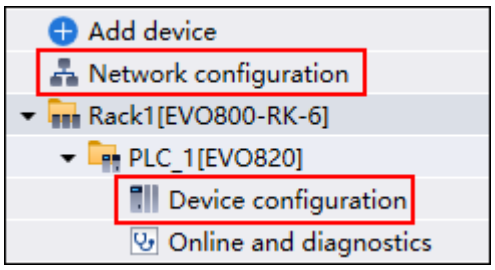
Background

- The two ports of the expansion module cannot be on the same network segment. For example, when the IP address of LAN1 is 162.168.0.1, the IP address of the LAN2 cannot be set to 192.168.0.2.
- The expansion module ports can be on the same network segment as the PLC's built-in port. For example, when the IP address of LAN1 on the Ethernet expansion module is 192.168.0.1, the IP address of the LAN1 on the PLC can be set to 192.168.0.10.

Procedure

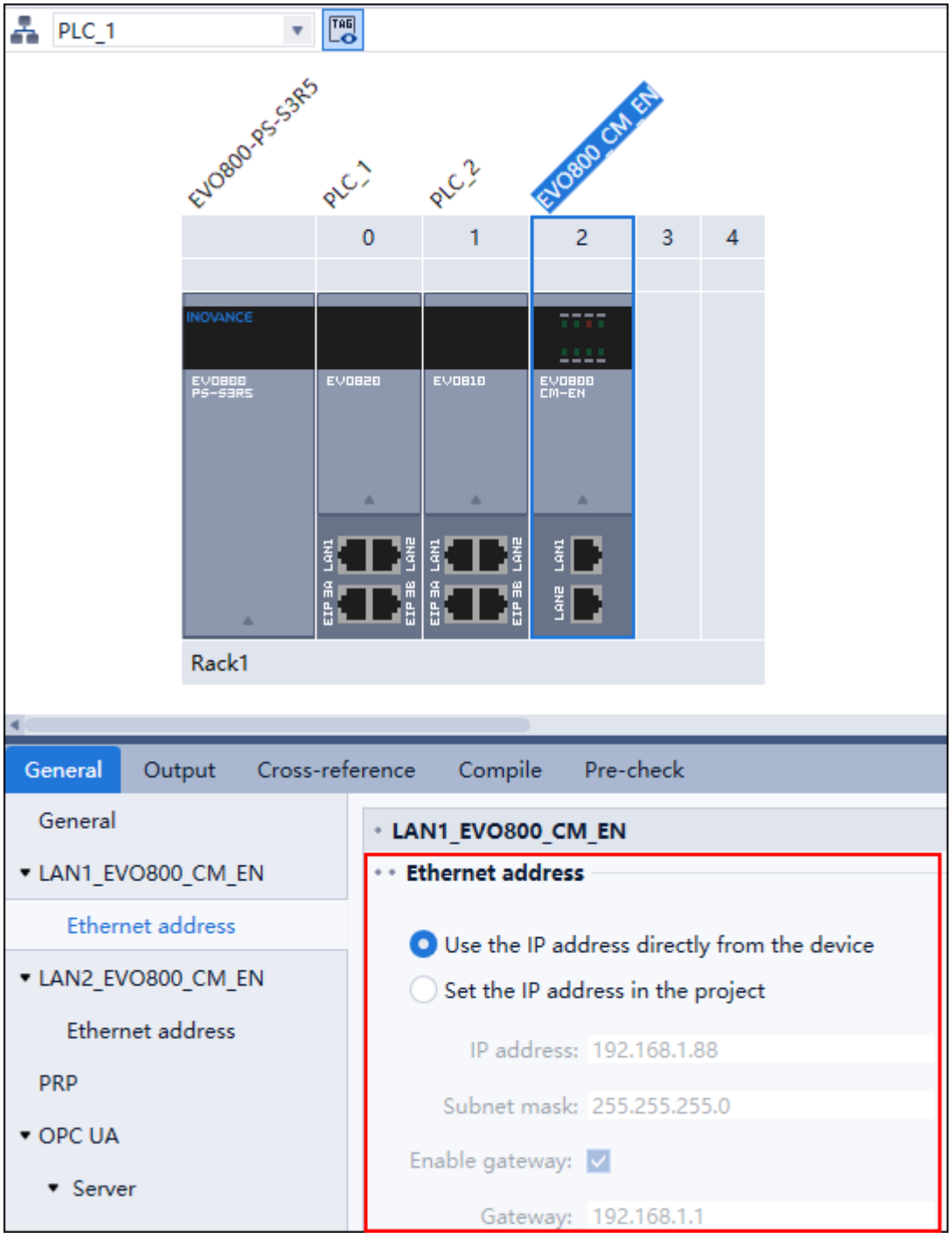
1. Open the configuration interface.

Double-click "Network configuration" or "Device configuration" in the project tree to open the configuration window.



2. Open the IP address setting interface.

   Switch to the network view or device view. Click the Ethernet expansion module diagram, and click the "Ethernet address" under LAN1 or LAN2 port on the "General" tab.

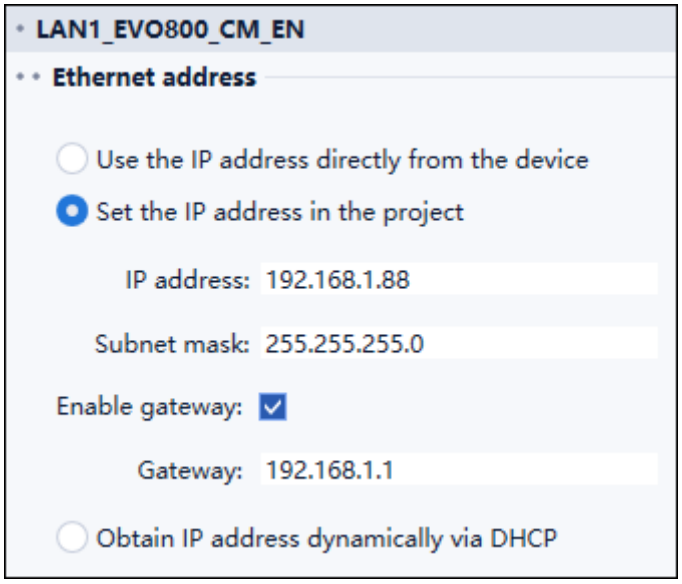


3. Set the IP address.

   - Use the IP address directly from the device.

     The IP address follows that set by the device and cannot be modified.

◦ Set the IP address in the project.



Parameter	Description
IP address	The two ports of the expansion module cannot be in the same network segment, but they can be in the same network segment as the PLC's built-in Ethernet port. Set them according to the actual network planning.
Subnet mask	Auto-adapts based on IP address and supports manual modification.
Gateway	The gateway's IP address must be in the same subnet as the "IP address".

10.6.15.2 Setting PRP

This section describes how to activate PRP and set PRP IP addresses via Ethernet expansion module properties. After PRP is enabled, the two network ports present only one PRP IP address (virtual IP) externally. The PRP IP address can also be set via "Online > Network setting" (see 10.11.5 Network Settings for details). After the device goes online, the changes are automatically synced back to this interface.

Background

PRP (Parallel Redundancy Protocol) is a redundancy protocol based on dual-network parallel transmission (compliant with IEC 62439-3). Identical data frames are sent via two independent Ethernet ports, with valid frames selected by the receiver per preset rules for zero-switch-delay high-availability communication. Upon link failure of one port, communication switches seamlessly to the other port automatically to ensure uninterrupted system communication.

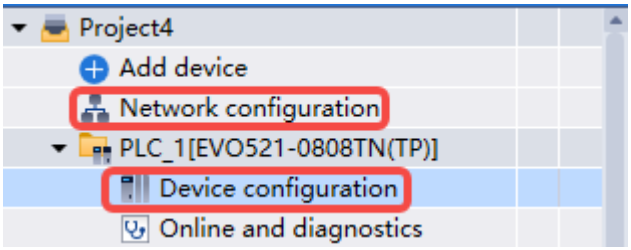
PRP core features:

- Dual-network redundancy: Two physically isolated networks (LAN A/LAN B) are required.
- Frame identification: A six-byte RCT is appended to the end of Ethernet frames, containing serial numbers and path identifiers.
- Seamless switchover: The optimal path is automatically selected upon faults, with recovery time less than 1 ms.

Procedure

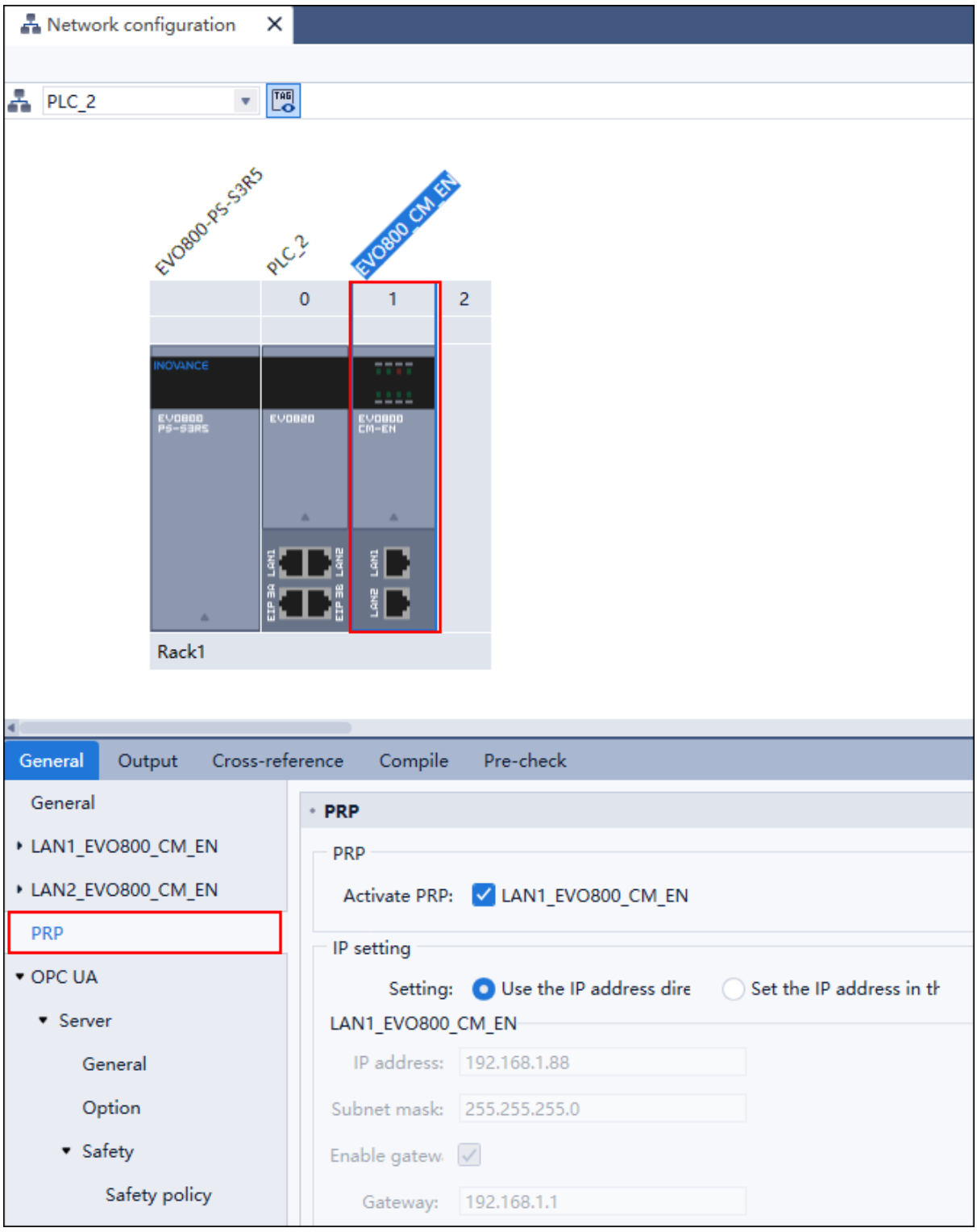
1. Open the configuration interface.

Double-click "Network configuration" or "Device configuration" in the project tree to open the configuration window.



2. Open the PRP setting interface.

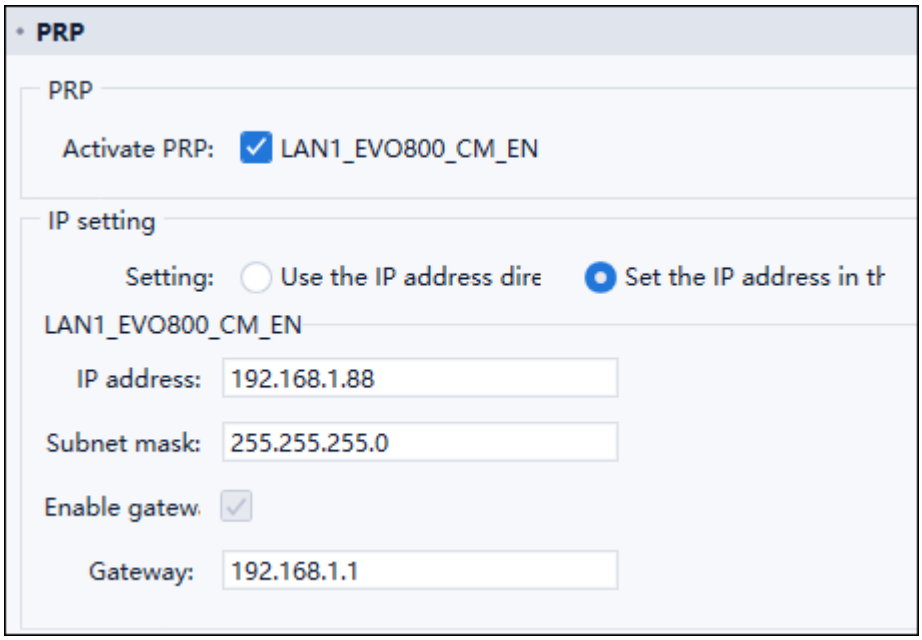
Switch to the network view or device view. Click the Ethernet expansion module diagram, and click "PRP" on the "General" tab.



3. Check "Activate PRP".

> **NOTE**
>
> After activating PRP, communication with external devices is only possible via the LAN1 port. That is:
>
> - Slaves can only be connected under the LAN1 port.
> - External communication services (Web Server, OPC UA Server, Modbus TCP Server, etc.) are only available through LAN1.

4. Set the PRP IP address.
   - Use the IP address directly from the device.
     The IP address follows that set by the device (through 10.11.5 Network setting) and cannot be modified.
   - Set the IP address in the project.



Parameter	Description
IP address	The PRP IP address of the Ethernet expansion module: the two network ports present only one virtual IP address externally. The PRP IP address of the Ethernet expansion module's network port and the IP address of the PLC module's network port can be set to the same subnet or to different subnets. Configure them according to actual network planning.
Subnet mask	Auto-adapts based on IP address and supports manual modification.
Enable gateway	Takes effect when checked.
Gateway	The gateway's IP address must be in the same subnet as the "IP address".

5. (Optional) To monitor PRP, see 10.11.23 PRP Monitoring for details.
6. (Optional) To view diagnostic information, see 10.11.20 Diagnostic Information for details.

10.6.15.3 Configuring OPC UA Server

The configuration related to the Ethernet expansion module's OPC UA server is the same as that of the PLC's OPC UA server. For specific operations, see 10.6.14.5 OPC UA Server Configuration.

NOTE

PLC public variables can be accessed externally via the OPC UA service on the Ethernet expansion module.

10.6.15.4 Configuring Web Server

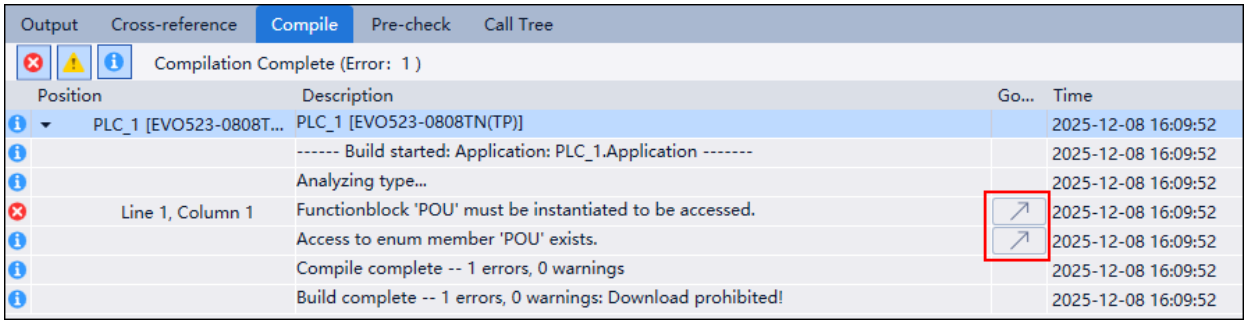
The Web server configuration related to the Ethernet expansion module is the same as that of the PLC's Web server. See 10.6.14.7 Web Server for specific operations.

10.7 PLC Compilation

10.7.1 Overview

After the program is written, it needs to be compiled before downloading. The compile command checks the syntax of the written program and only compiles the programs added to the task. If the created POU is not added to the task, the compile command does not perform a syntax check on that POU. If the program is directly switched to online, the system will execute the compile command by default (equivalent to manually executing the compile command first), and execute the connection login command if no syntax error is detected. Similarly, during compilation, no syntax check is performed on POUs that are not added to the task. Code will be generated when executing the login command.

After the compile command is executed, the generated information is displayed in the "Compile" tab in the monitoring area, where you can view the number and location of faults or alarms. Through the monitoring area, you can view and search for faults or alarms, and modify the program according to the prompt information.



- Double-click the faults or alarms in the monitoring area to locate.
- Right-click a fault or alarm in the monitoring area and select "Go to" to locate the specific position.
- Click the  icon after a fault or alarm to locate the specific position.

10.7.2 Pre-Check

The pre-check function is used to check for syntax errors in the user-configured project in real time and provide prompts. When the user configuration content changes, the compiler will automatically trigger a pre-check to alert the user to syntax errors in the code in advance.

The pre-check aims to provide a quick preliminary check mainly based on syntax, to verify whether the project configuration conforms to the syntax rules. It can also check for basic types associated with the configuration (excluding user-defined types).

The pre-check mainly covers two parts: the variable area and the program area.

- The variable area can be displayed in table view or text view. In the table view, the system checks the validity of each property field of the variables. In the text view, the system checks the validity of each property field of the variables and the syntax of the variable declaration statements. The specific check points are as follows:
  - Whether the variable declaration statement conforms to the syntax rules for declaring variables.
  - Whether the variable name is a valid identifier and whether it is duplicated.
  - Whether the data type is valid.
  - Whether the address is valid.
  - Whether the initial value is within the valid range of the type.
  - Whether the variable supports power failure retention.
- The program area includes two languages: graphical language and text language. In the graphical language mode, the system checks the validity of the content associated with each component of the graphical language In the text language mode, the system checks the syntax and validity of the text content. The specific check points are as follows:
  - Whether statements and expressions conform to syntax rules.
  - Whether the variables referenced by statements and expressions exist.
  - Whether the variable types associated with expressions conform to syntax rules.
  - Whether the expression types match.
  - Whether the number and types of formal parameters and actual parameters in function calls match.
  - Whether the constant subscripts in array access are within the upper and lower bounds.
  - Whether type conversions conform to the rules.
  - Whether the label of the jump target exists.

10.7.3 Compilation

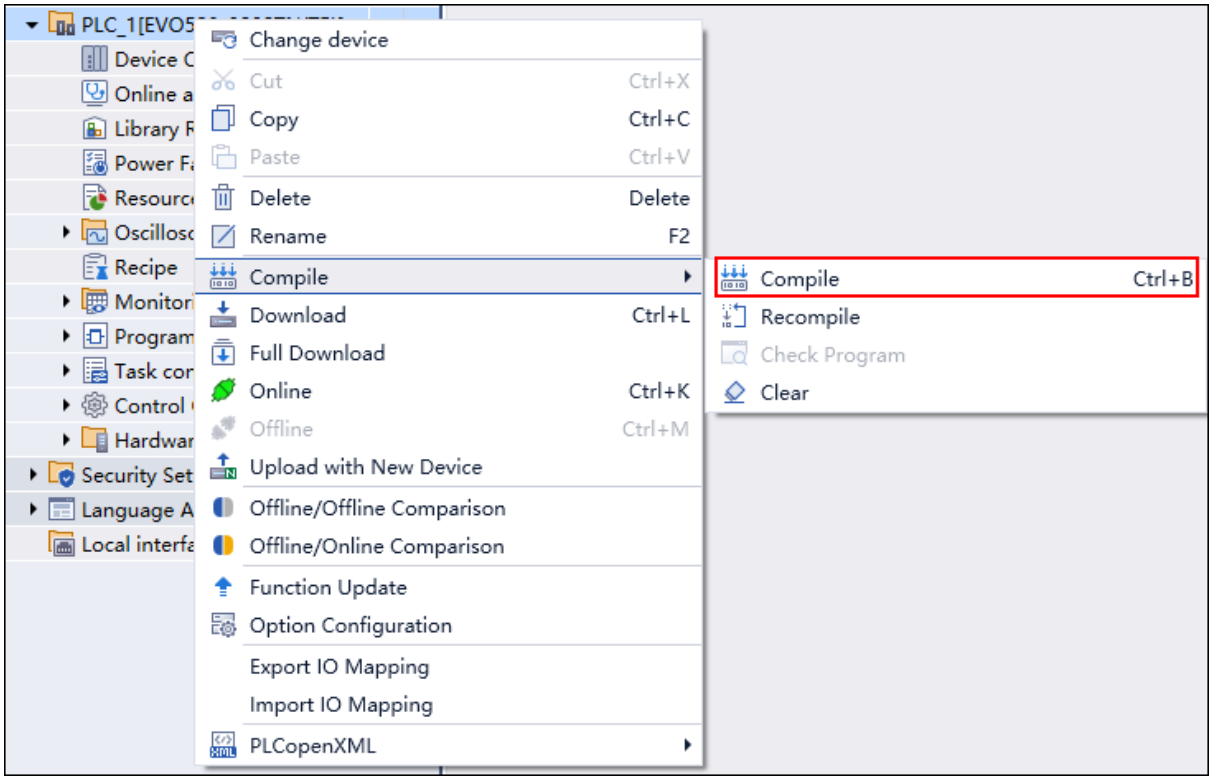
Compile all code into code that the PLC can execute.

Prerequisites

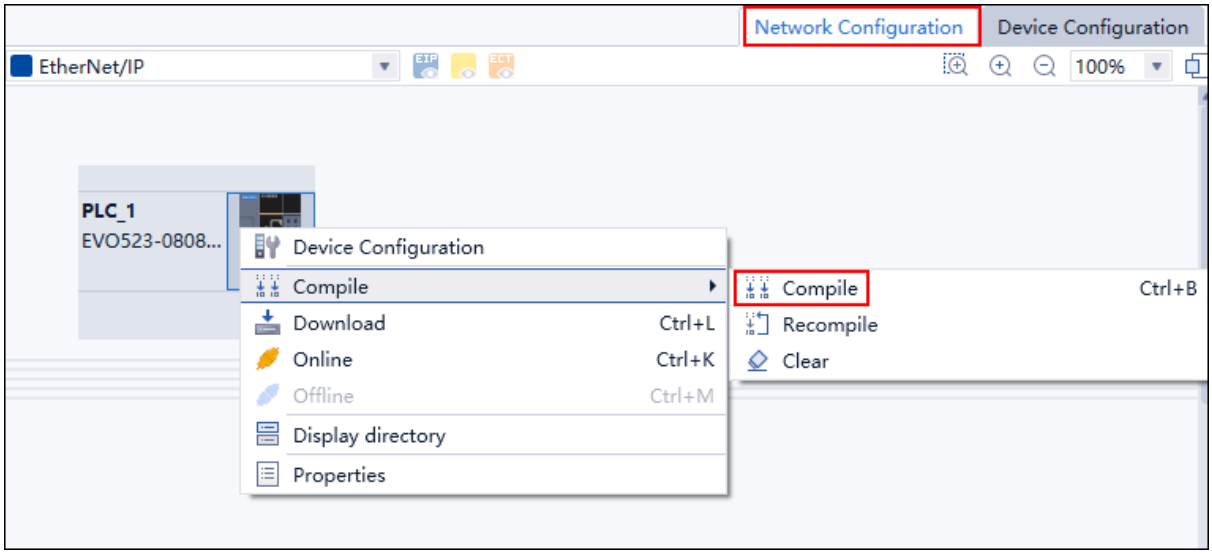
There are compilable programs in the project.

Procedure

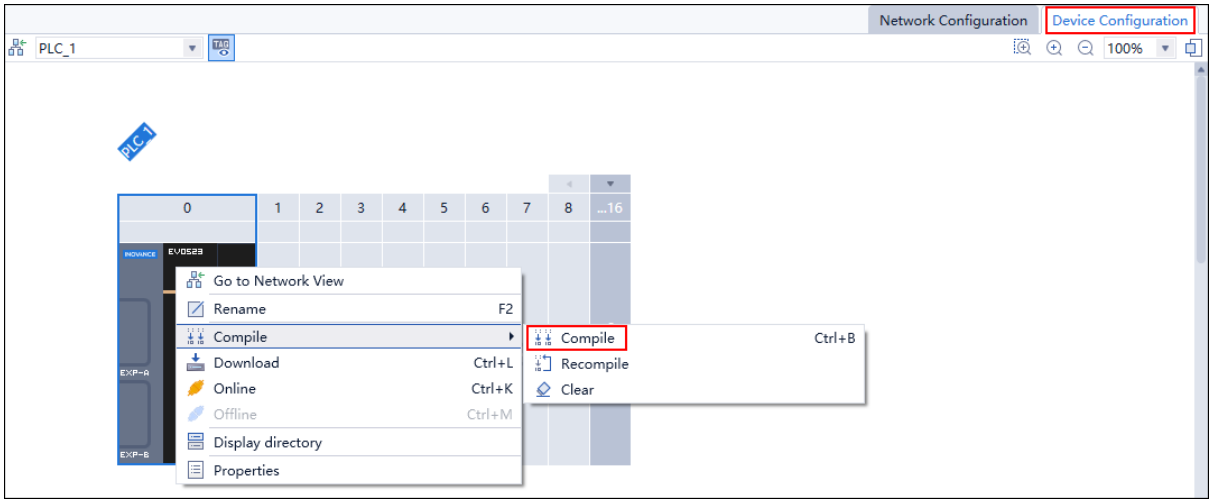
1. There are several ways to compile the project.
   - Toolbar: In the project tree, select the corresponding PLC project and click the "Compile" icon on the toolbar.
   - Menu bar: In the project tree, select the corresponding PLC project and select "Compile > Compile" in the menu bar.
   - Right-click menu: Right-click the corresponding PLC project in the project tree and select "Compile > Compile" to download the project to the device, as shown in the following figure:



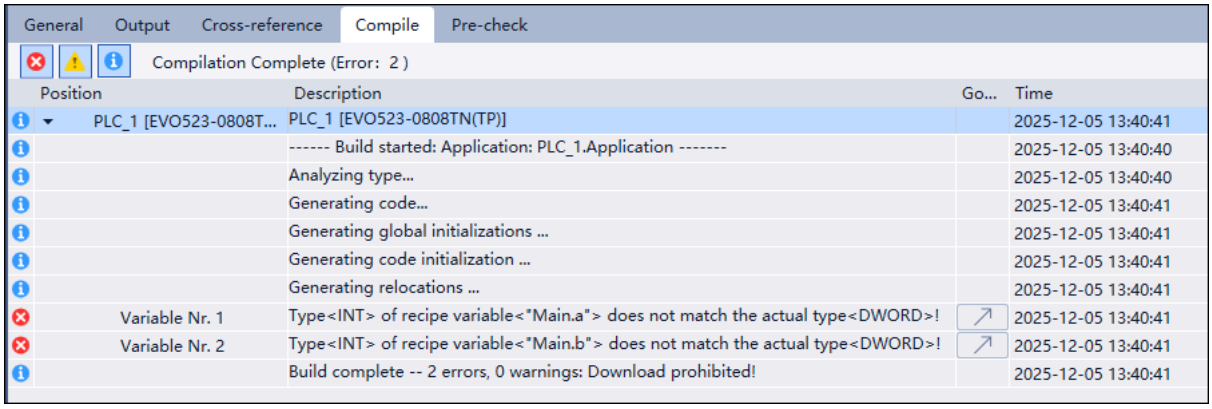
- Shortcut key Ctrl+B: After selecting the device, when the compile icon is enabled, press Ctrl+B on the keyboard to compile.
- Network view: In the "Network configuration" interface, right-click the corresponding PLC and select "Compile > Compile".



- Device view: In the "Device configuration" interface, right-click the corresponding PLC and select "Compile > Compile".



2. View the compilation information.
   After compilation, the corresponding compilation information is displayed in the "Compile" tab in the monitoring area, which includes faults, alarms, and information. Click the icons to hide/show the corresponding information. When a compilation is abnormal, there will be a corresponding prompt. Double-click the prompt to locate the specific position.



10.7.4 Recompilation

Clear the previous compilation information and re-run the compilation command.

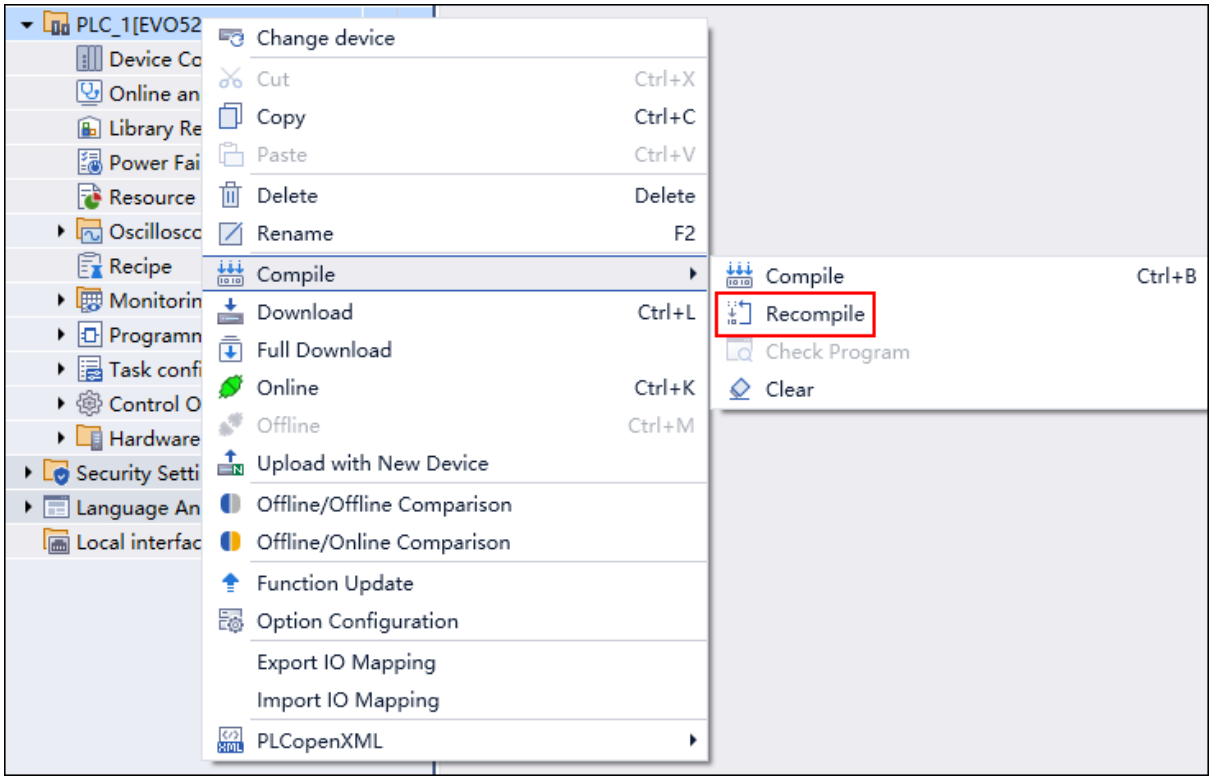
> **NOTE**
>
> Under the premise that the source code remains unchanged, recompiling, clearing, and full clearing operations will not affect the consistency of the compilation, and the project remains online.

Prerequisites

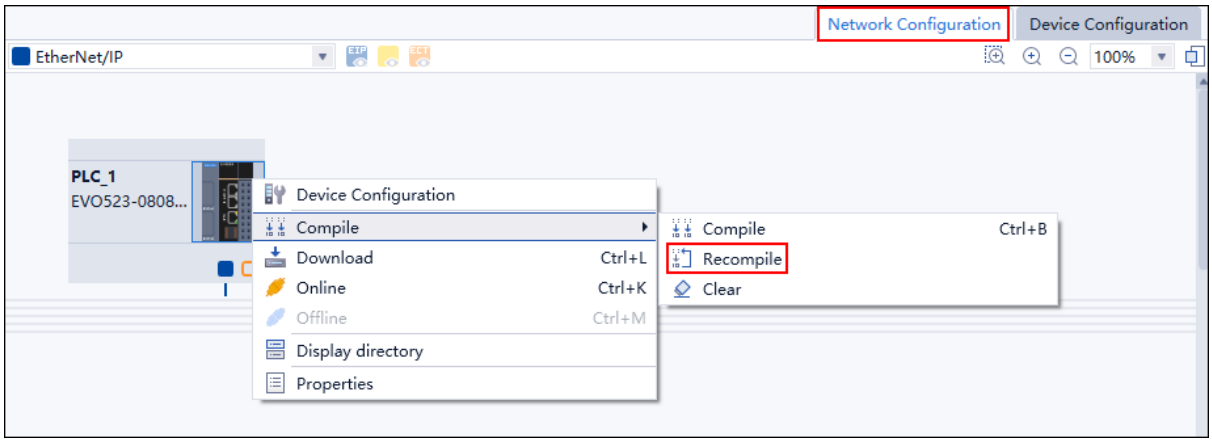
The device is in an offline status.

Procedure

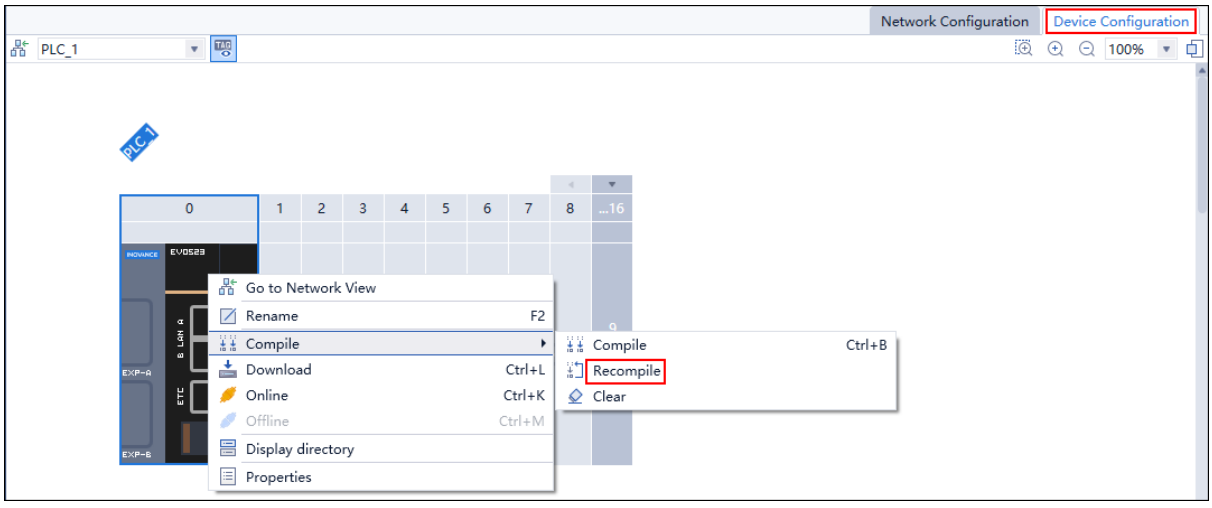
1. Execute the recompilation command.
   There are several ways to recompile.
   - Menu bar: In the project tree, select the corresponding PLC project and select "Compile > Recompile" in the menu bar.
   - Right-click menu: Right-click the corresponding PLC project in the project tree and select "Compile > Recompile" to download the project to the device, as shown in the following figure:



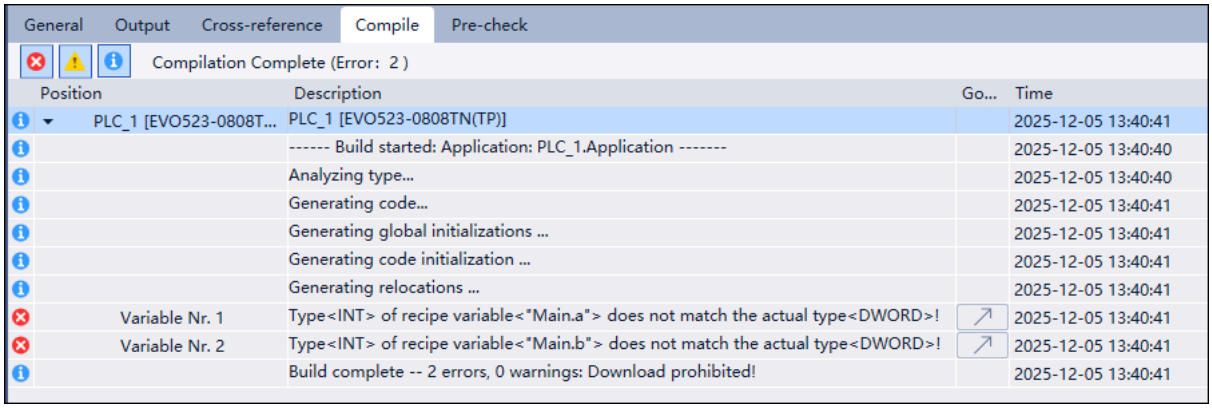
- Network view: In the "Network configuration" interface, right-click the corresponding PLC and select "Compile > Recompile".



- Device view: In the "Device configuration" interface, right-click the corresponding PLC and select "Compile > Recompile".



2. View the compilation information. After recompilation, the corresponding compilation information will be displayed in the "Compile" tab in the monitoring area.



10.7.5 Clearing

Clear the compilation information of the current device (excluding library information). If you log in to the device again, the compilation information needs to be regenerated.

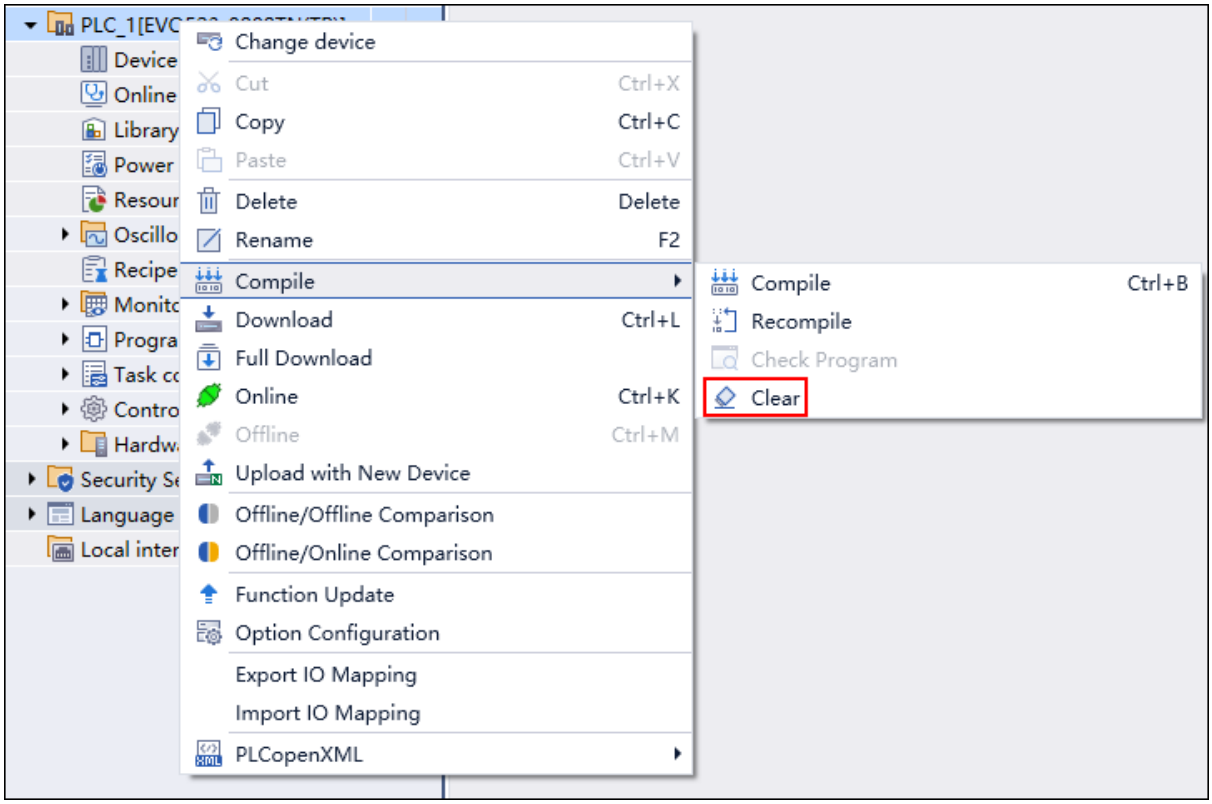
> **NOTE**
>
> Under the premise that the source code remains unchanged, recompiling, clearing, and full clearing operations will not affect the consistency of the compilation, and the project remains online.

Prerequisites

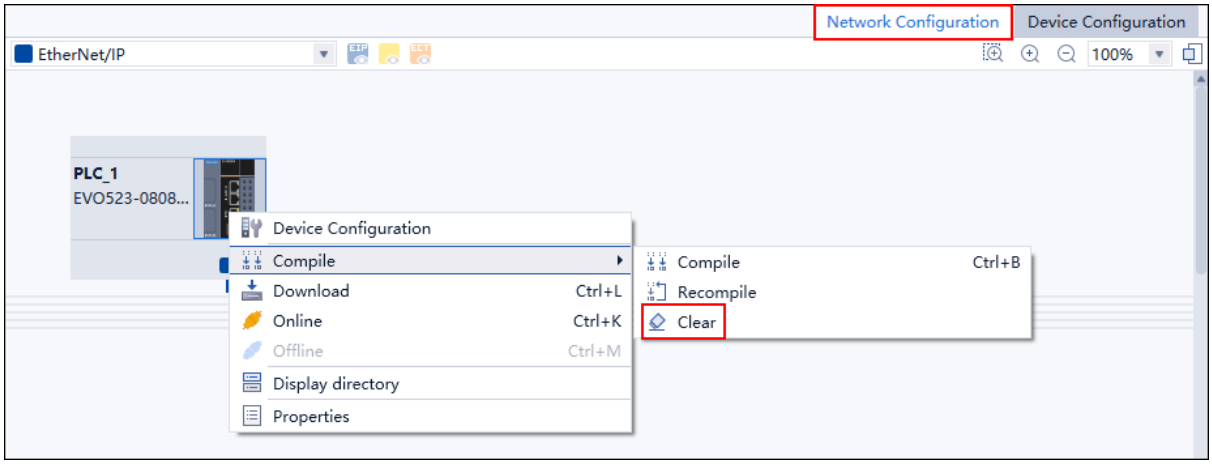
The current device is in an offline status.

Procedure

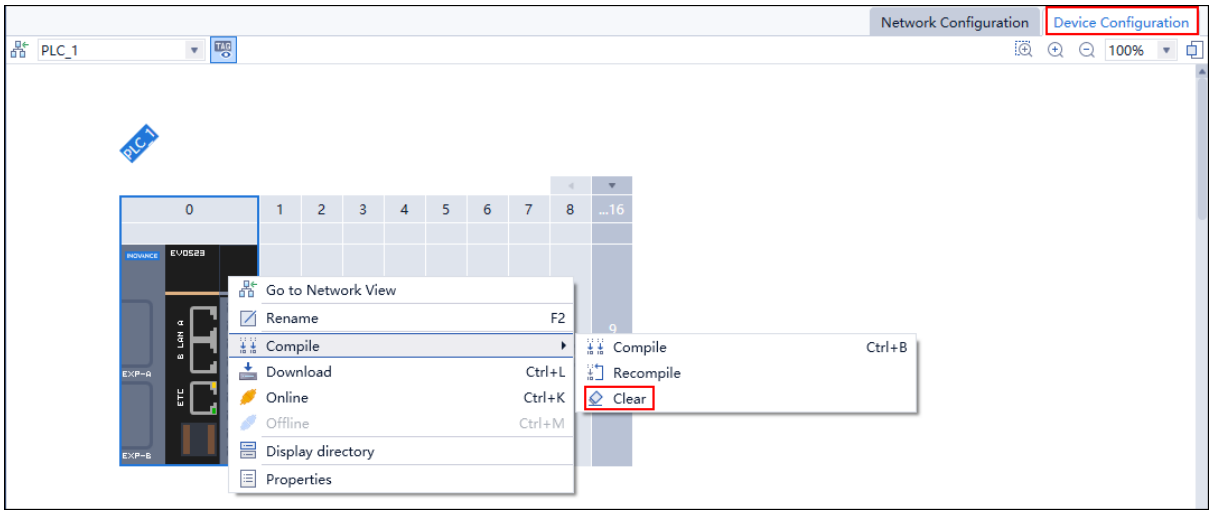
1. There are several ways to clear the compilation information of the project.
   - Menu bar: In the project tree, select the corresponding PLC project and select "Compile > Clear" in the menu bar.
   - Right-click menu: Right-click the corresponding PLC project in the project tree and select "Compile > Clear", as shown in the following figure:



◦ Network view: In the "Network configuration" interface, right-click the corresponding PLC and select "Compile > Clear".



◦ Device view: In the "Device configuration" interface, right-click the corresponding PLC and select "Compile > Clear".



2. In the pop-up "Confirm to clear" window, click "OK" to perform the clear operation.

10.7.6 Clearing All

Clear all compilation information, download information, and reference information from all devices, and refresh the compilation information for all libraries and project data.

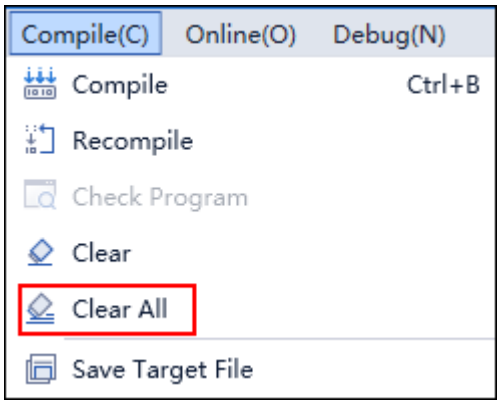
> NOTE
>
> Under the premise that the source code remains unchanged, recompiling, clearing, and full clearing operations will not affect the consistency of the compilation, and the project remains online.

Prerequisites

All current devices are in an offline status.

Procedure

1. Select "Compile > Clear All" in the menu bar to clear all related information.



2. In the pop-up "Confirm to clear" window, click "OK" to perform the clearing all operation.

10.7.7 Option Configuration

10.7.7.1 Compilation Options

The software supports configuring compilation options before compilation.

Function entry

Right-click a PLC project node in the project tree and select "Option configuration" to open the configuration interface.

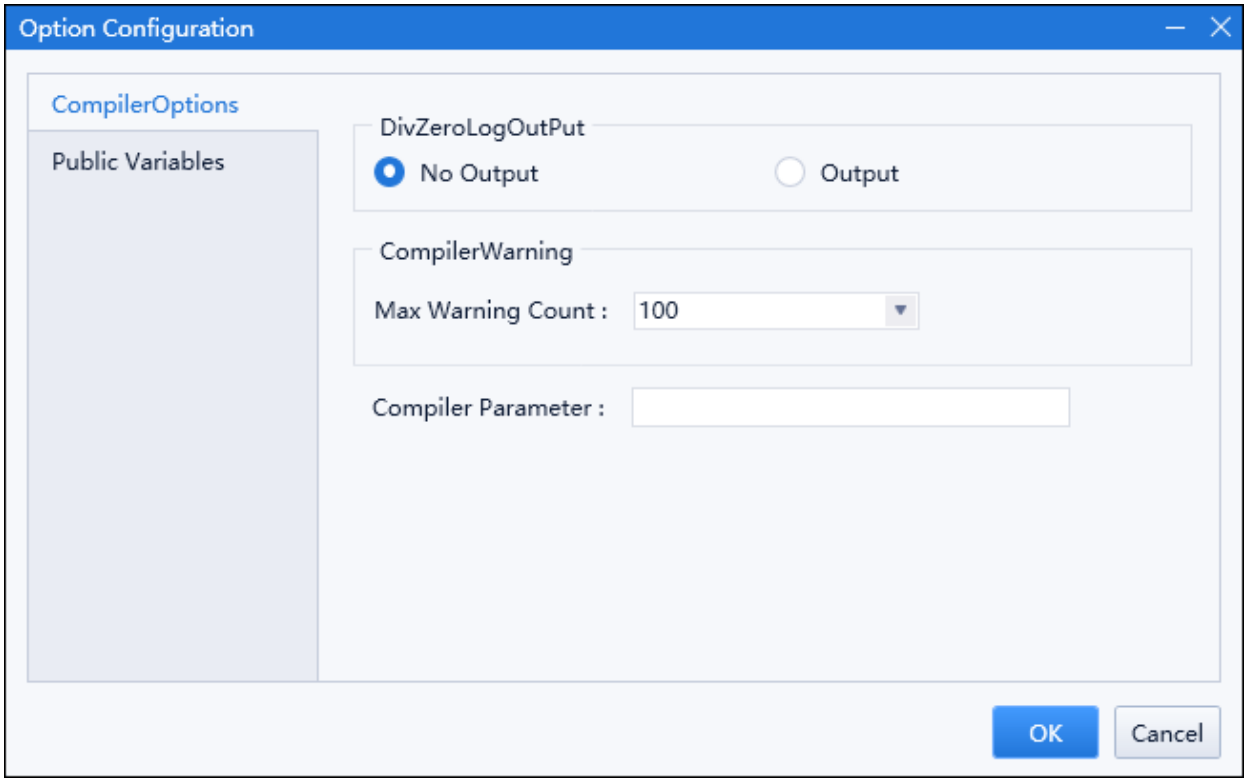


Figure 10-7-7-1-1 Compilation option configuration interface of EVO800 PLC

- Division by zero diagnostic information
  - No output: Do not output diagnostic information.
  - Output: Output diagnostic information.
- Configuration
  - Support pointer: If checked, using pointers will trigger an alarm; if unchecked, the use of pointers is prohibited, and any usage will result in a fault.
- Compilation alarm
  - Maximum number of alarms: Configurable to 100/200; after configuration, compilation can only report the corresponding number of pre-compilation messages.
- Compiler parameter
  Indicates developer parameters Users do not need to pay attention.

NOTE

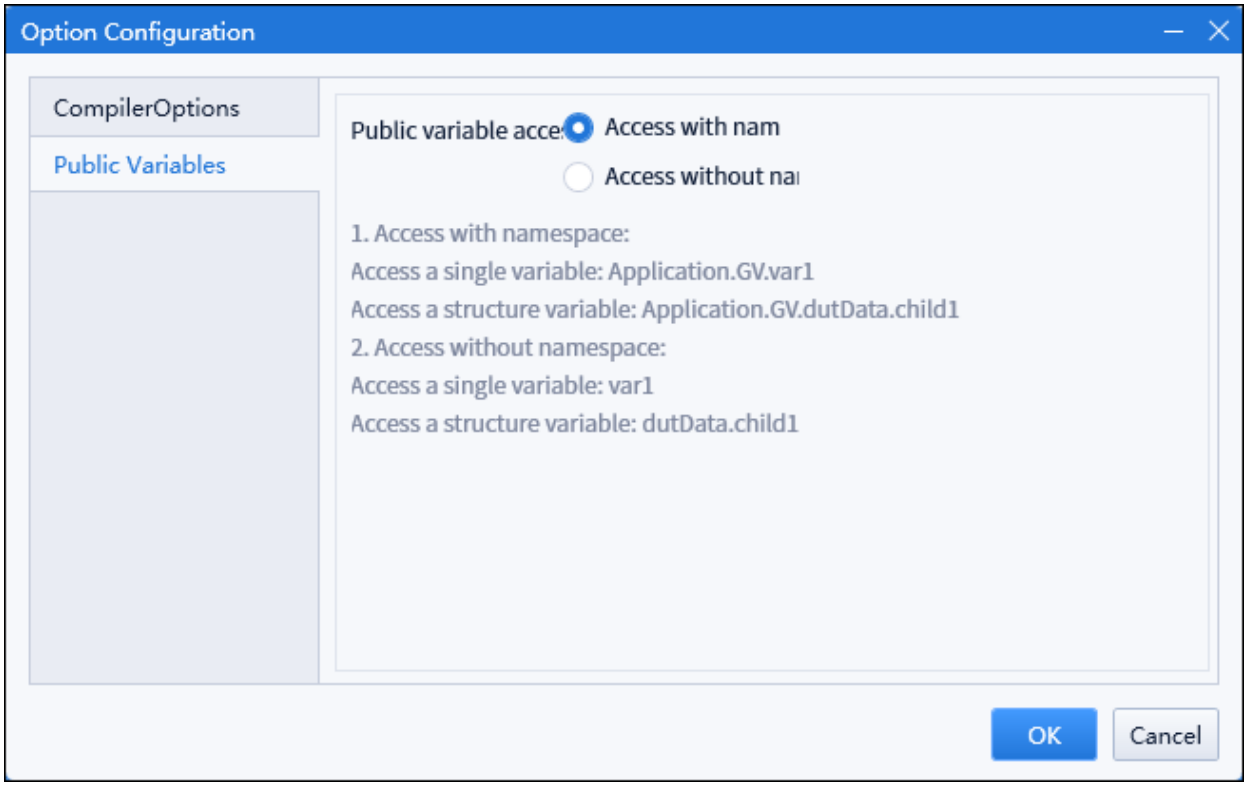
The EVO400/EVO500 series PLC does not support pointer settings.

10.7.7.2 Public Variables

Software supports configuring public variables before compilation.

Function entry

Right-click a PLC project node in the project tree and select "Option configuration > Public variables" to open the configuration interface.



- Public variable access configuration
  - Access with namespace: If checked, users can access network public variables in this device using "Application.Namespace.VariableName", which supports duplicate variable names.
  - Access without namespace: If checked, users can directly access network public variables in this device using "VariableName", and duplicate variable names are not supported.

NOTE

The EVO500 series PLC V1.2 version currently does not support the public variable configuration feature.

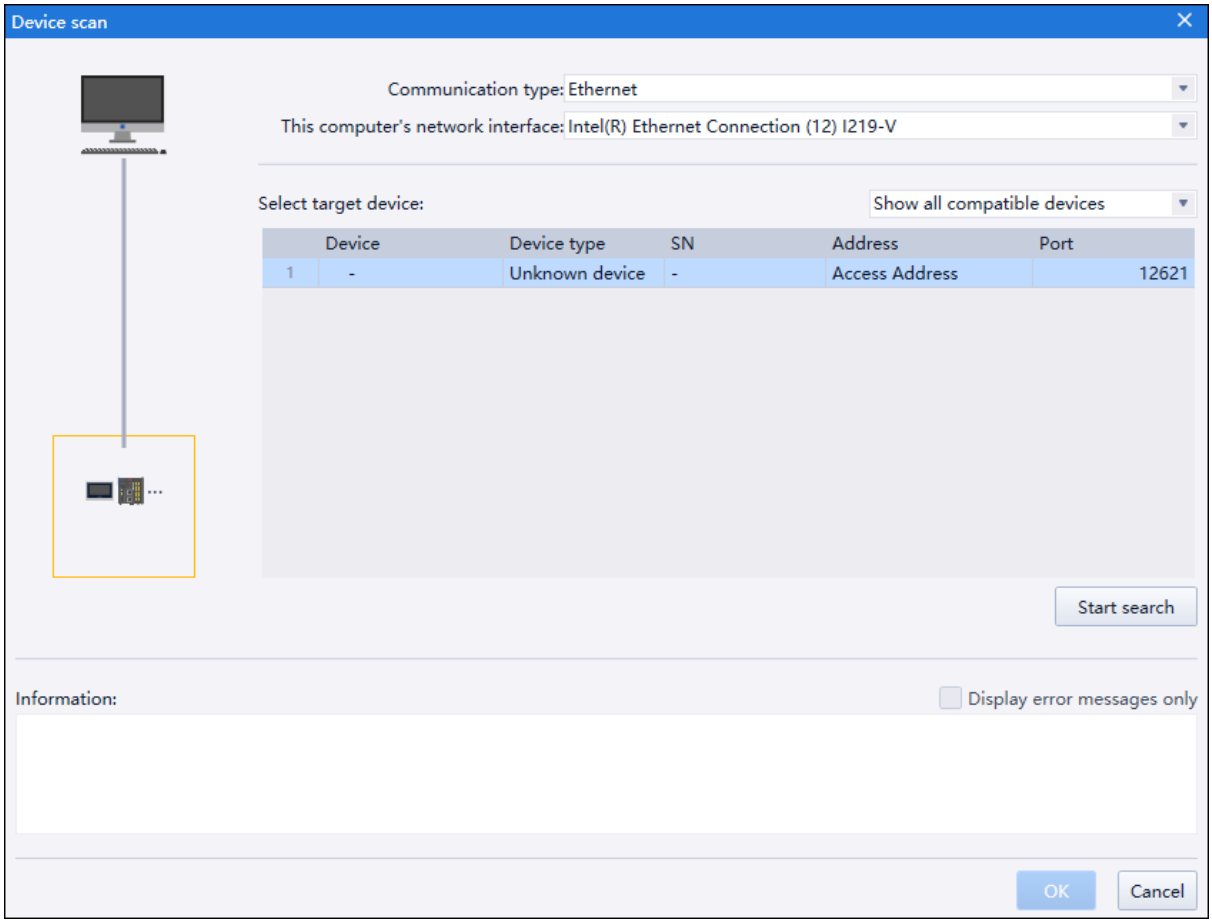
10.8 PLC Download

10.8.1 PLC Connection

Before downloading the PLC, establish a connection between the computer and the PLC, and set the connection method, IP address, and other communication parameters.

Device scanning

1. Select "Online > Download" in the menu bar to open the "Device scanning" window.



2. Set communication parameters.
   Select the communication type and the network interface of this computer, then click "Start search".
   The virtual PLC only supports Ethernet communication type. For details, see 10.13.2 Establishing a Simulation Environment.

3. Check the target controller information.
   Verify the name, SN code, IP address, port information of the target PLC, or check whether the RUN indicator of the target object flashes to ensure proper connection.
   The virtual PLC does not support connection check via RUN indicator or SN code comparison.

NOTE

The EVO800 series PLC supports downloading and debugging via the Ethernet port of the Ethernet expansion module. When performing device scanning through the Ethernet expansion module port, only the PLC to which the currently connected Ethernet expansion module belongs can be scanned; other PLCs cannot be scanned.

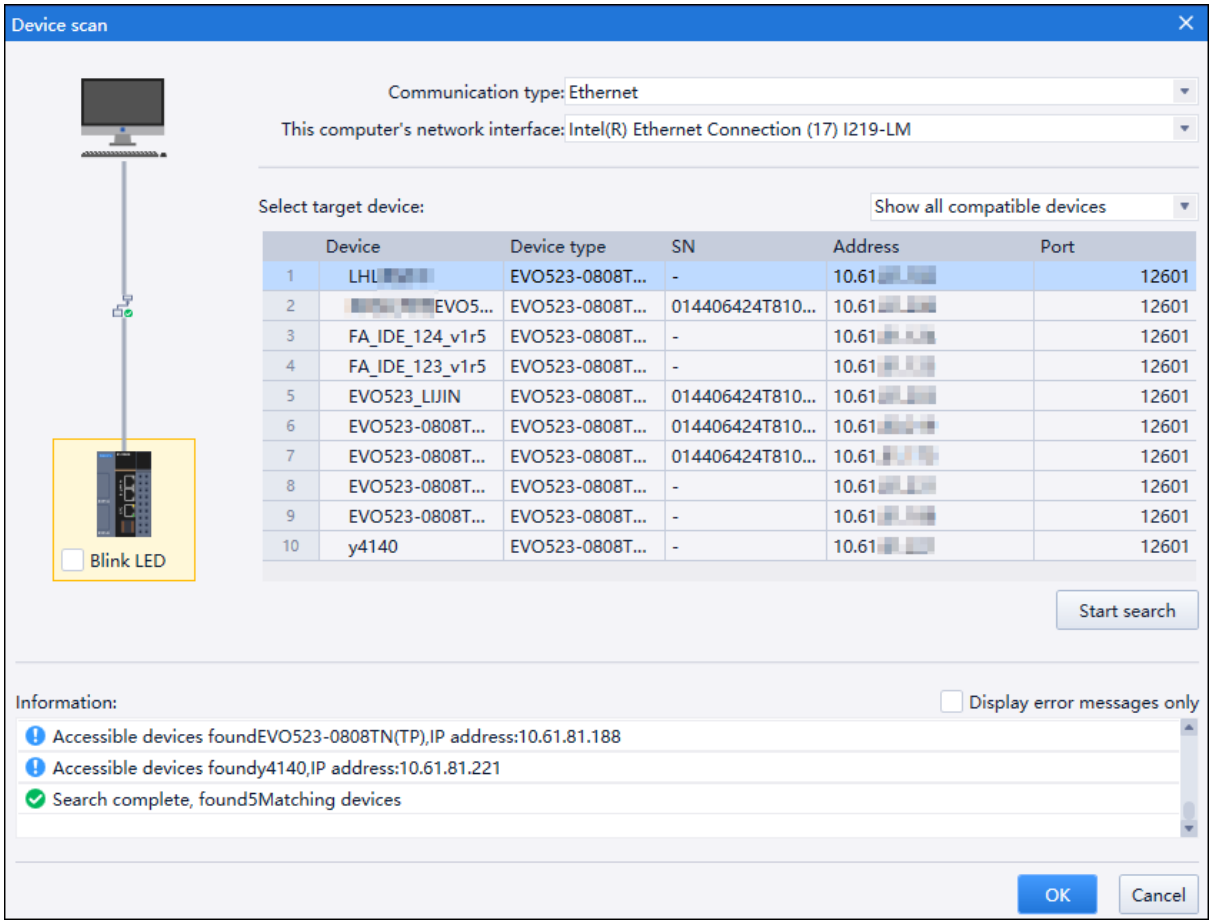


Figure 10-8-1-1 Physical PLC search results

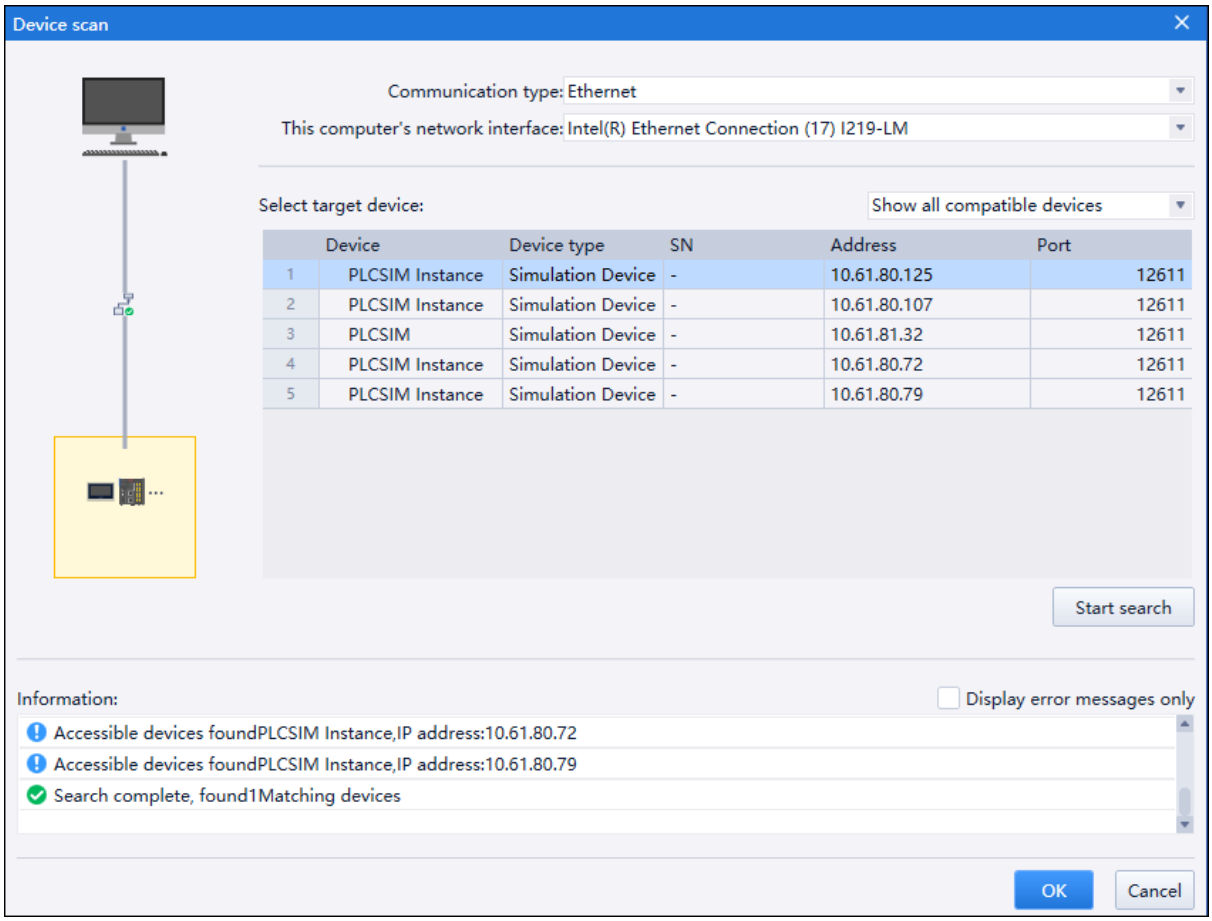


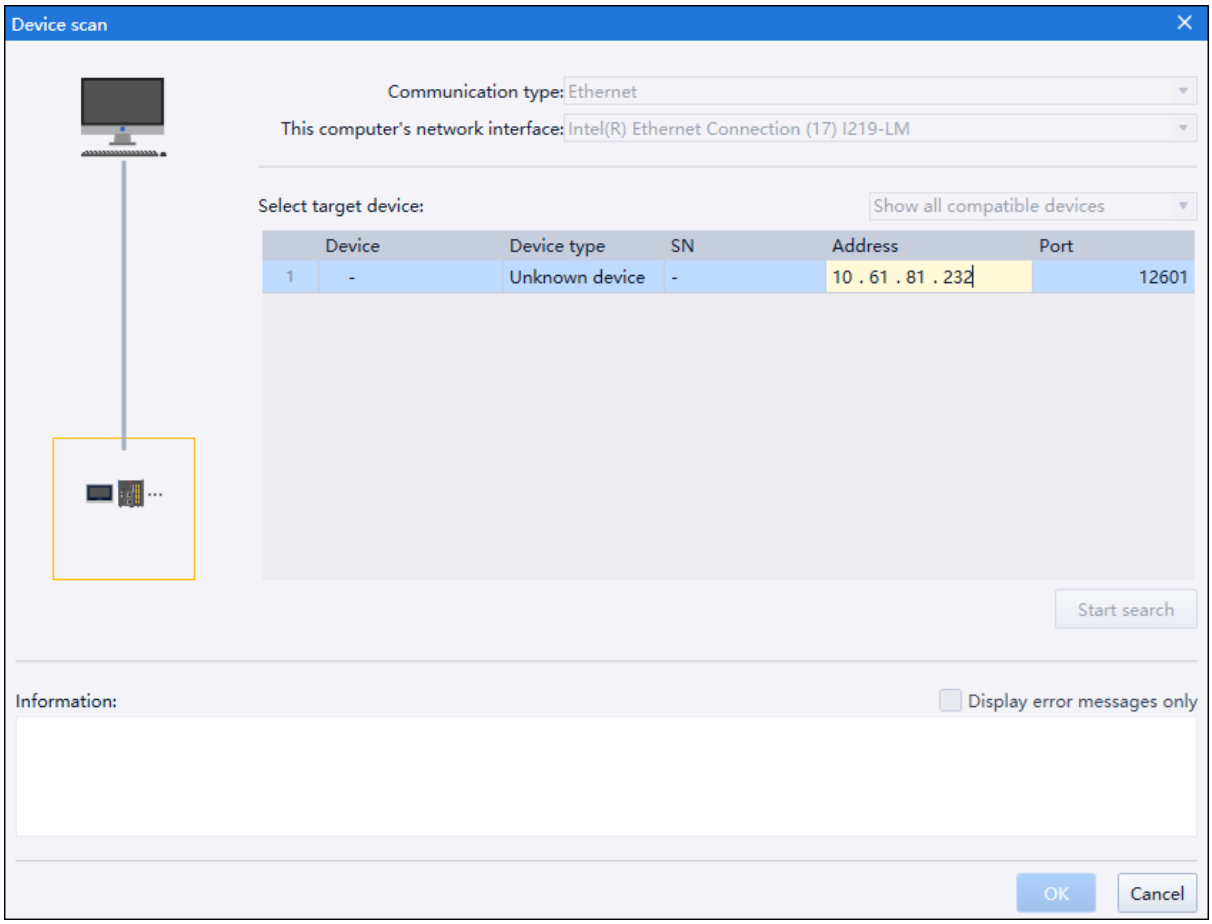
Figure 10-8-1-2 Virtual PLC search results

4. Confirm the connection object.

   Select the target PLC and click "OK" to establish the connection between the computer and the target PLC.

Direct connection

1. Select "Online > Download" in the menu bar to open the "Device scanning" window.
2. Double-click the address area, enter the IP address of the target device, and click any other area to directly connect to the target device.



10.8.2 Download Permission Check

The upload/download operations of the PLC are restricted by both software user permissions and PLC access permissions. The upload/download operations are supported only when both permissions are satisfied.

- User login permission of the iFA software
  Users logged in to the software must have the permission to download PLC projects. For details, see 9.1 Permission Management.
- PLC access permission
  If 10.11.24 upload/download permissions are set for the PLC, password verification will be performed during the execution of upload/download commands.
- PLC operation permission

Users need to log in to the software to execute the PLC operations with the corresponding permissions. For details, see 10.11.6 PLC Permission Settings.

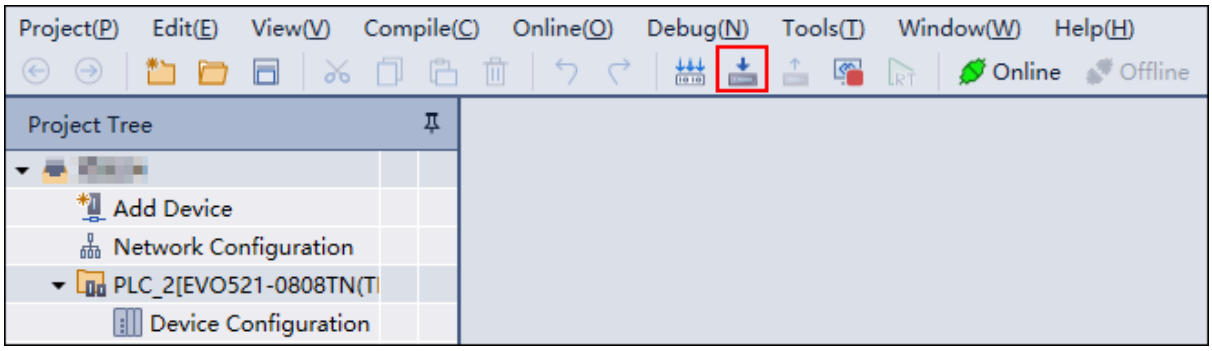
10.8.3 Full Download

Prerequisites

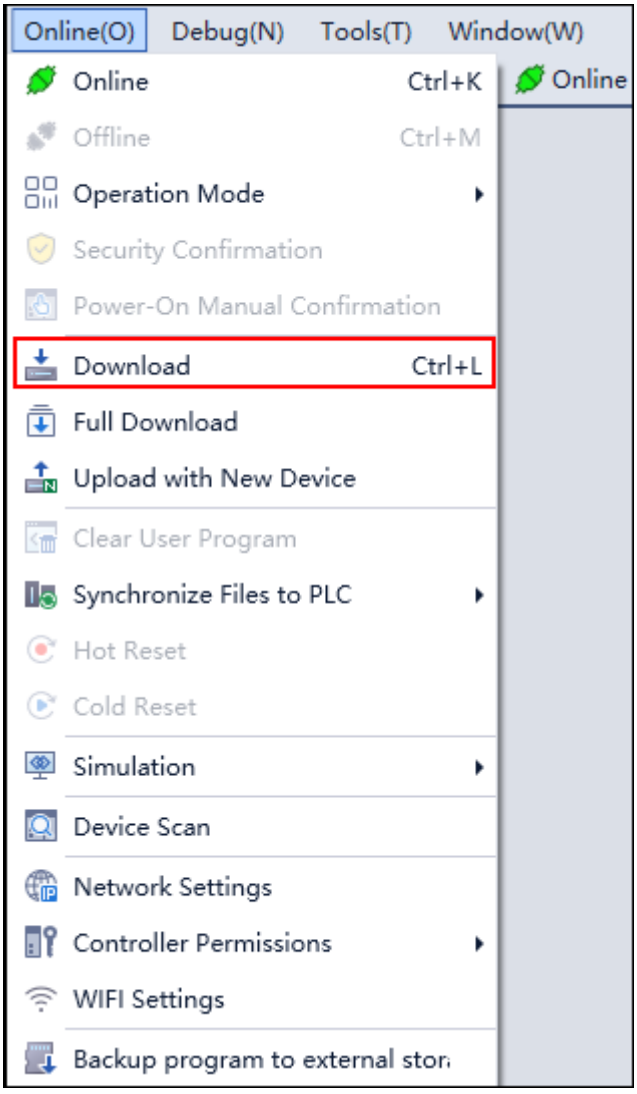
- A program or hardware configuration has changed.
- The project is in offline mode.

Procedure

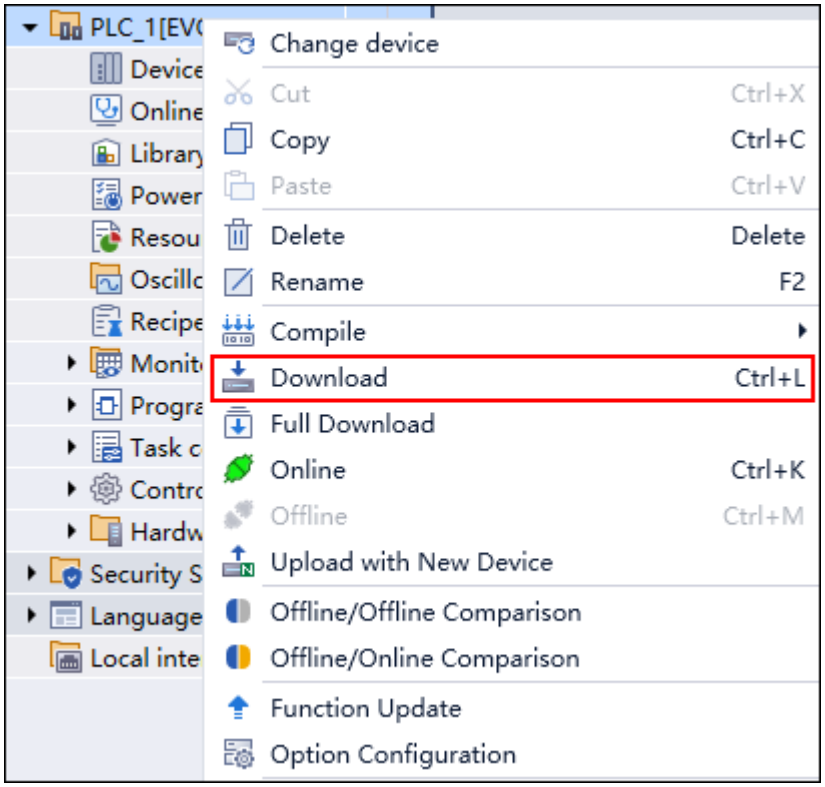
1. Click "Download".
   There are three download entries: the toolbar, the menu bar, and the right-click menu of the PLC node in the project tree.
   - Toolbar: Select the corresponding PLC project in the project tree, then click the "Download" icon in the Toolbar, as shown in the figure below.



   - Menu bar: Select the corresponding PLC project in the project tree, then click "Online > Download" in the menu bar, as shown in the figure below.

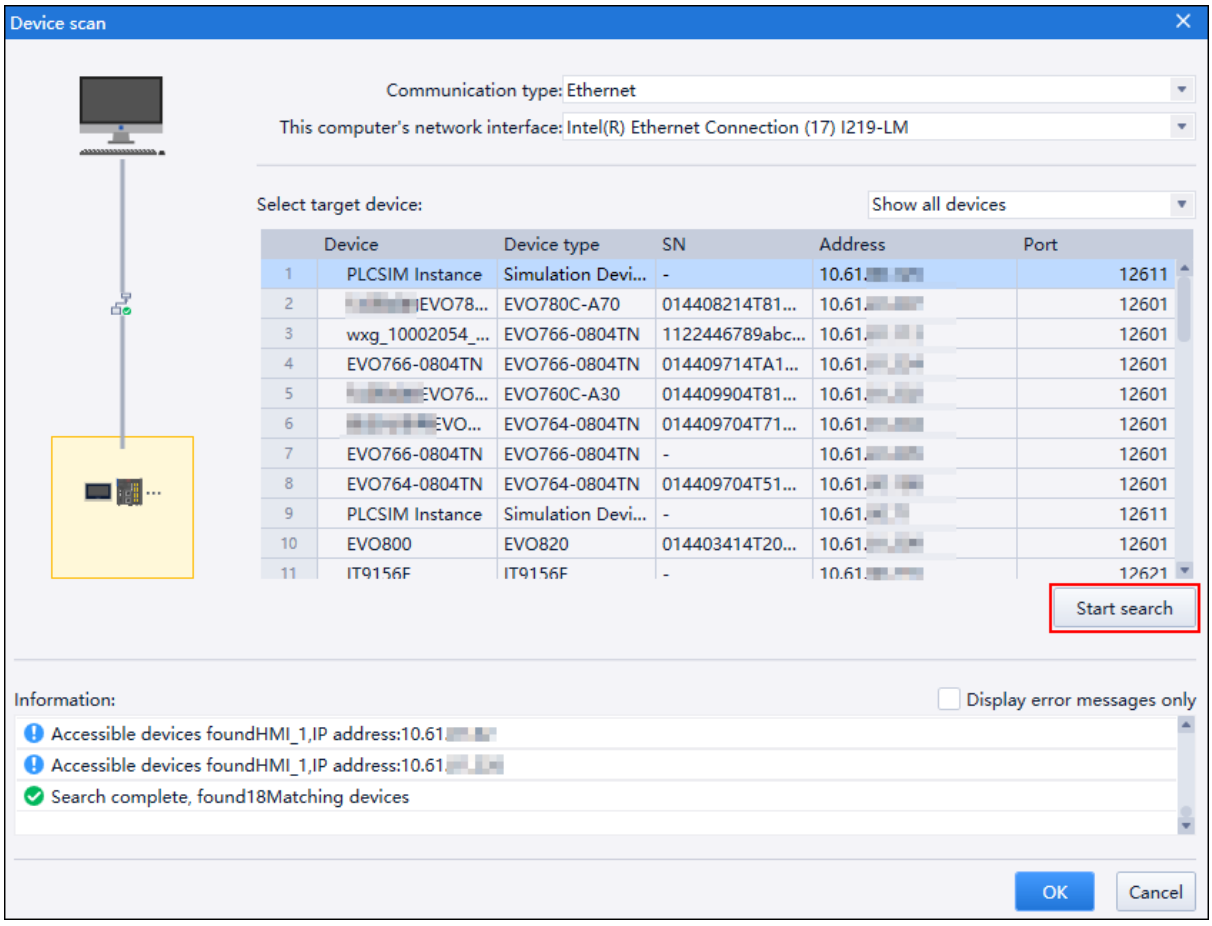


   - Right-click menu: Right-click the corresponding PLC project in the project tree, then select "Download", as shown in the figure below.



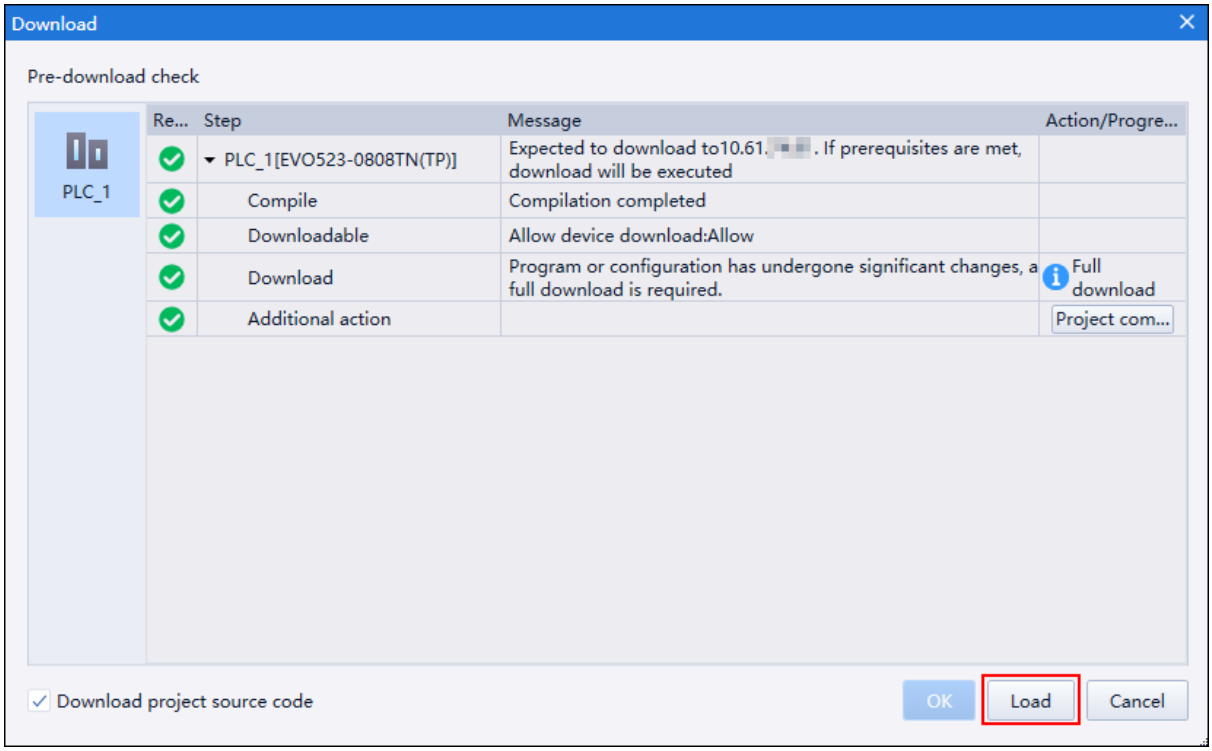
2. Configure download options.

   In the pop-up device scanning window, click "Start search". Select the target device to download, then click "OK", as shown in the figure below.



3. Check before downloading.

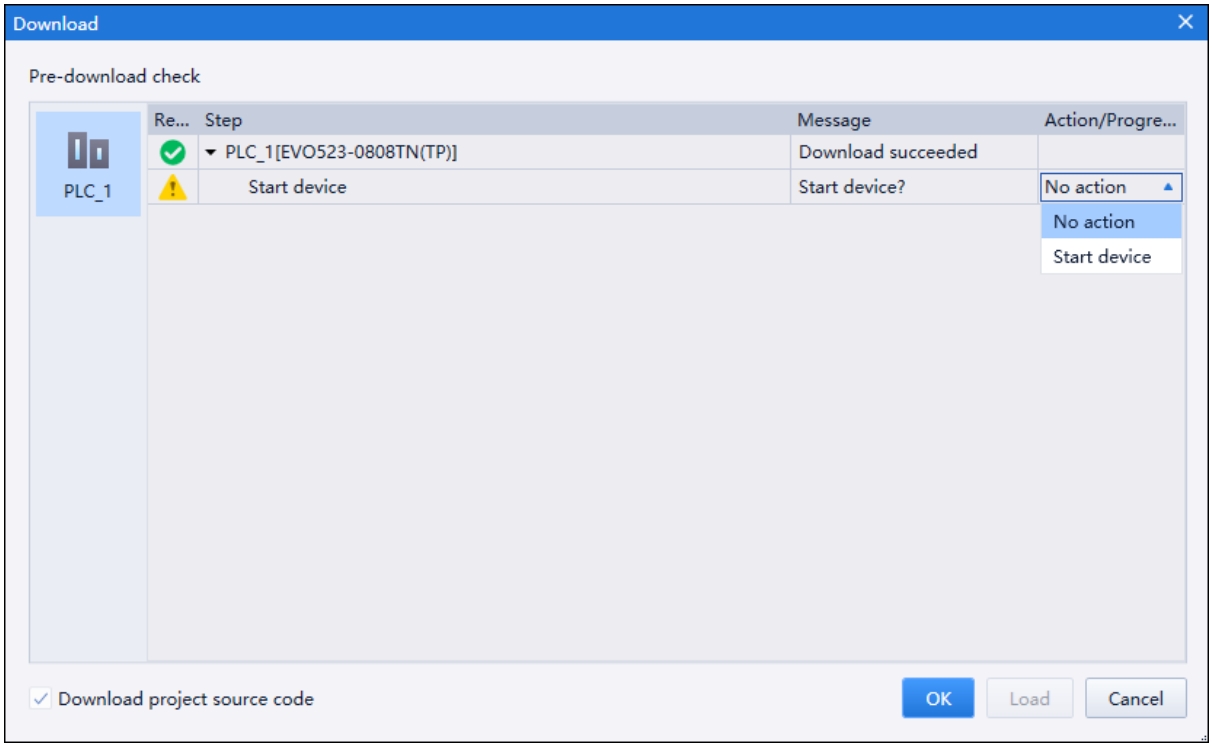
   Before downloading, a series of checks will be performed, and information such as the IP address of the target device will be prompted. After configuration, click "Load", as shown in the figure below:



The main check items are as follows:

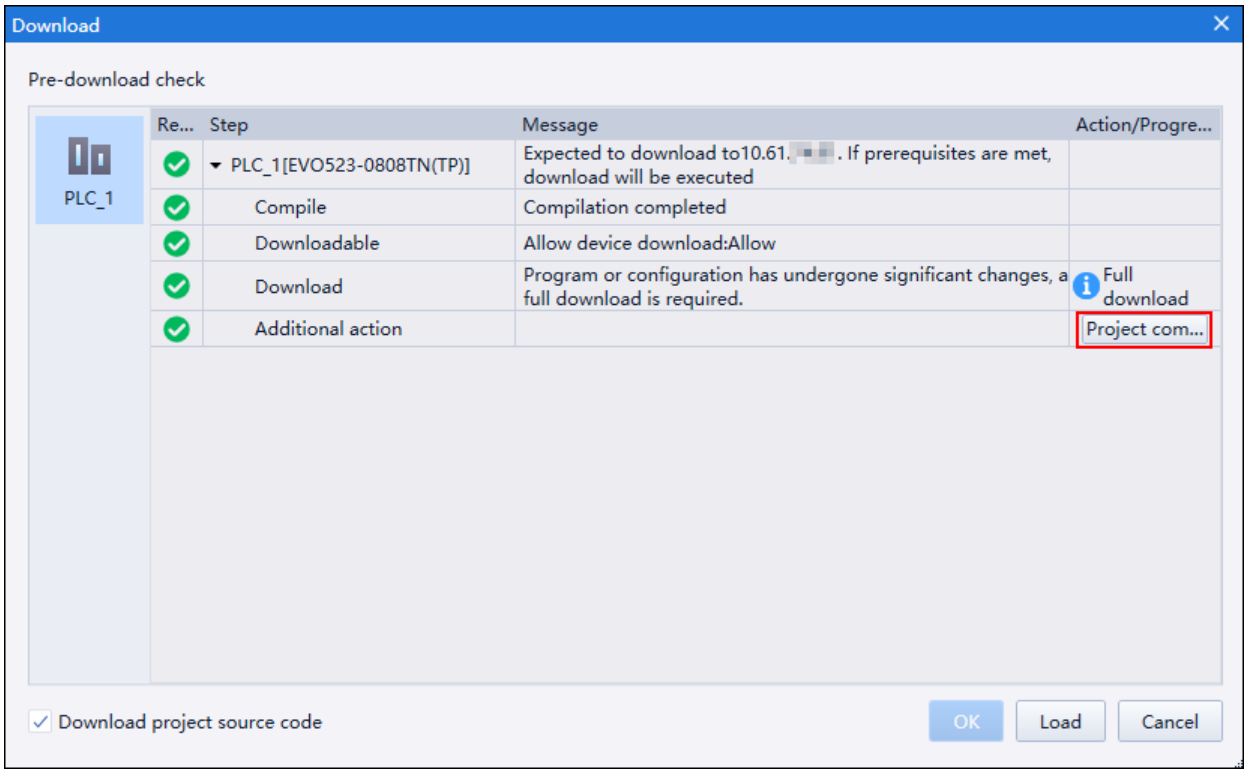
- Check whether shutdown is required
  For programs that must stop the PLC before downloading, users must select the "Stop device" action before downloading.
- Automatic compilation before downloading
  If compilation has not been performed before downloading, compilation will be automatically executed.
- Confirm whether to download project source code
  By default, "Download project source code" is checked, which will download the project source code to the PLC. If unchecked, the project source code will not be downloaded to the PLC.
- Check for encryption permissions downgrade
  A security prompt will be issued if the encryption permissions are downgraded before downloading.
- Prompt about modified content
  A notification will be displayed before downloading, indicating the modified content, allowing users to review the changes.
- Check for password
  If a password has been set for the PLC, users must enter the password before downloading; otherwise, downloading is not allowed.

4. Actions after downloading
   After download completes, an option to start the device will appear. Select "Start device" to run the downloaded project; select "No action" to only download the project to the device without starting it, as shown in the figure below.



Project comparison

When downloading a project, click "Project comparison" to view differences in project contents. For details, see 10.12 PLC Project Comparison.



After starting the project comparison, the download operation is canceled and must be restarted to download the project to the device.

10.8.4 Online Modification/Monitoring without Modification

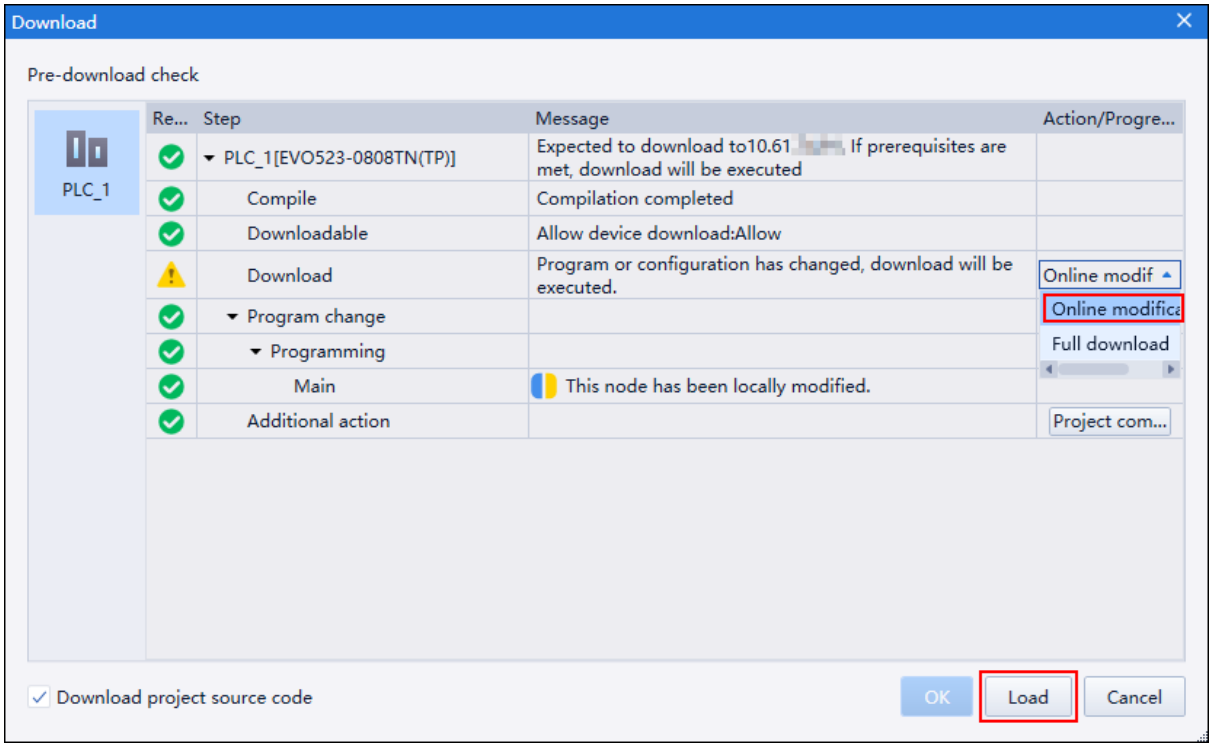
When the project is offline and needs to connect to an online controller, there are two modes:

- Online modification mode: Allows online modification and download of project objects that are not under monitoring in the controller, without stopping the PLC.

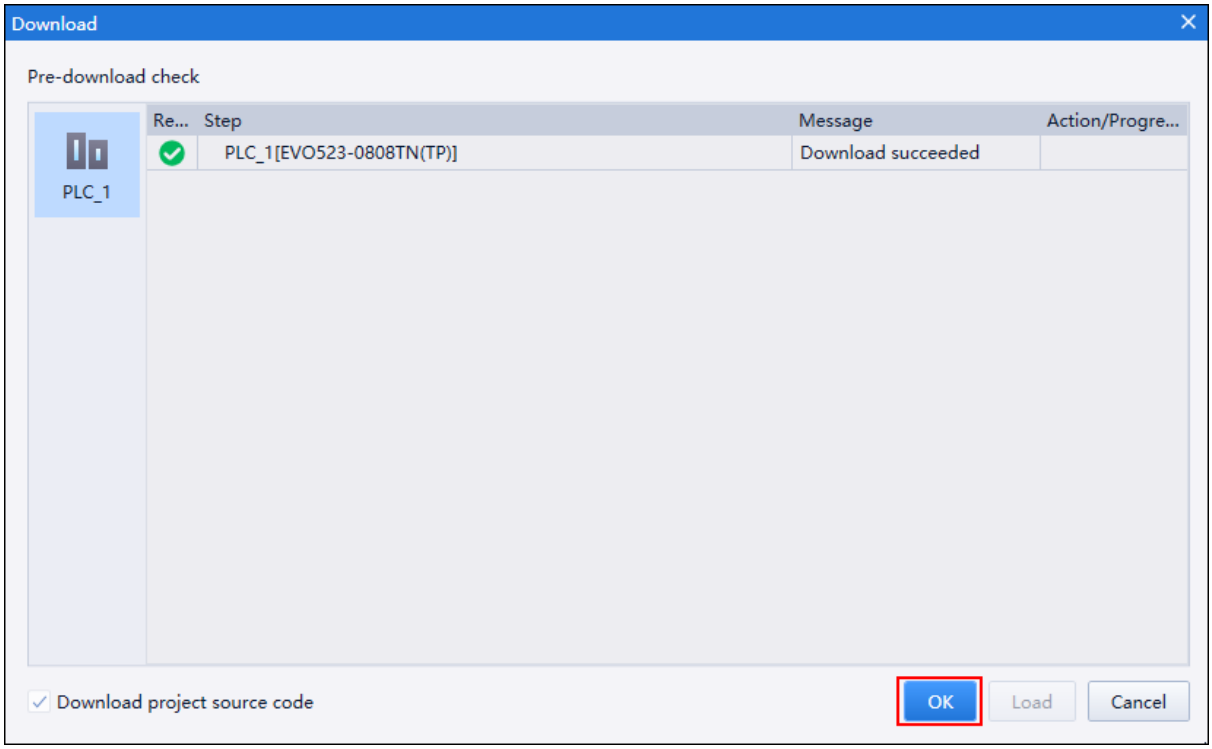
- Monitor without modification mode: Allows online monitoring of data changes from the last project downloaded to the controller.

Online change

1. Compile the project and click "Download" after compilation. If online modification is feasible, the dialog box for download pre-checks will appear, confirming related check items and providing a prompt for the modification content.
2. Select "Online modification" and click "Load" to download the current modification to the device.



3. After download, a success message appears.



Monitoring without modifying

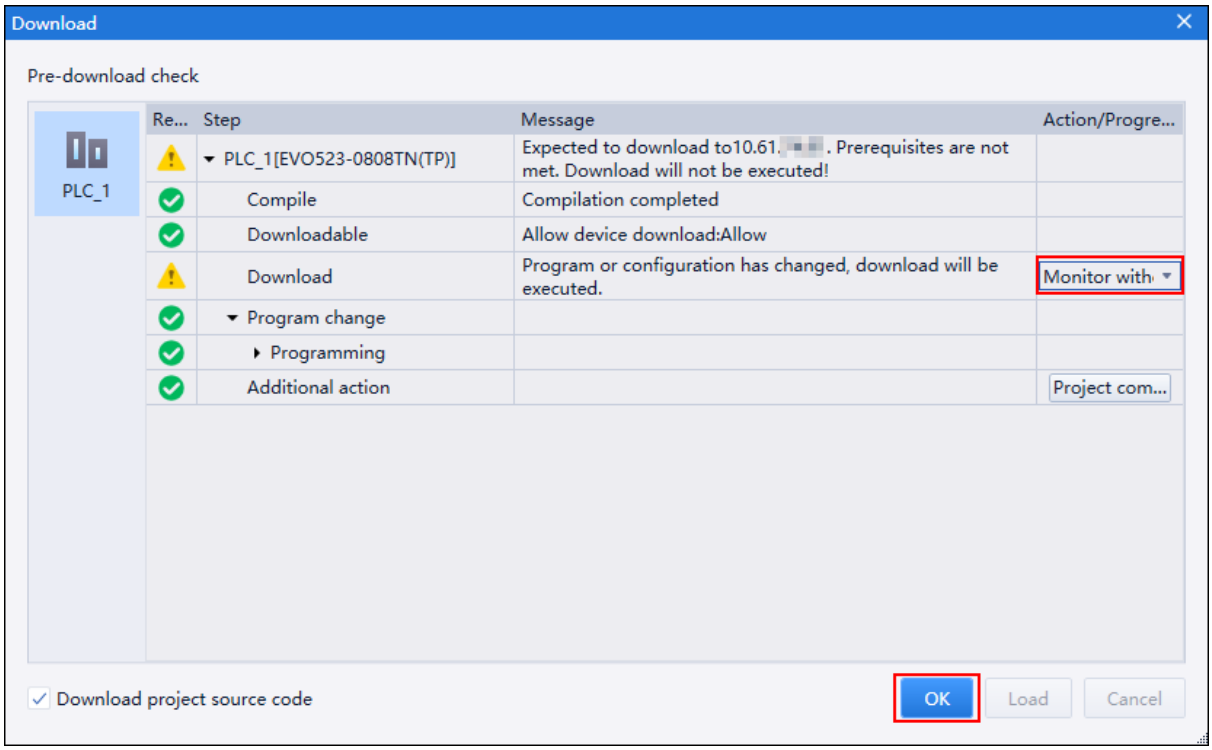
> **NOTE**
>
> This feature is only available in software versions V1.4.4.5 and earlier.

Prerequisites

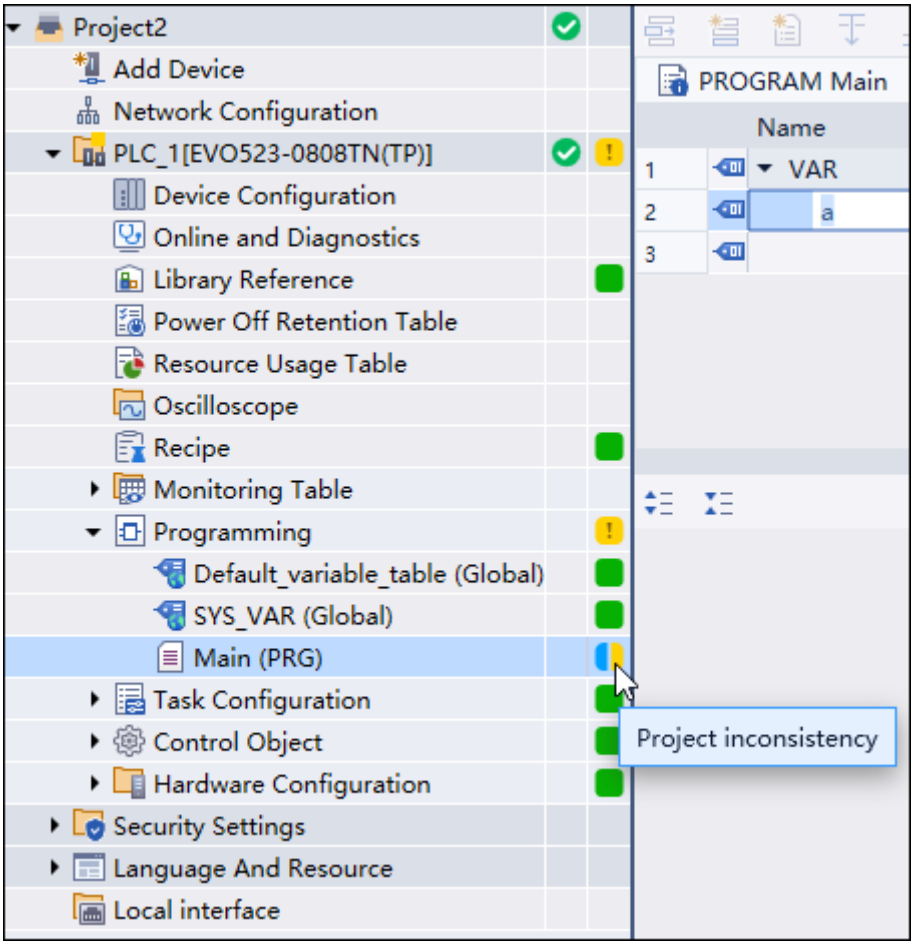
- A program or hardware configuration has changed.
- The project is in offline mode.

Procedure

1. Edit the project and click the "Online" icon in the menu bar. The dialog box for download pre-checks will pop up.
2. Select "Monitor without modification".



3. The device enters online status, and the project tree displays the data changes in the project, as shown in the figure below.

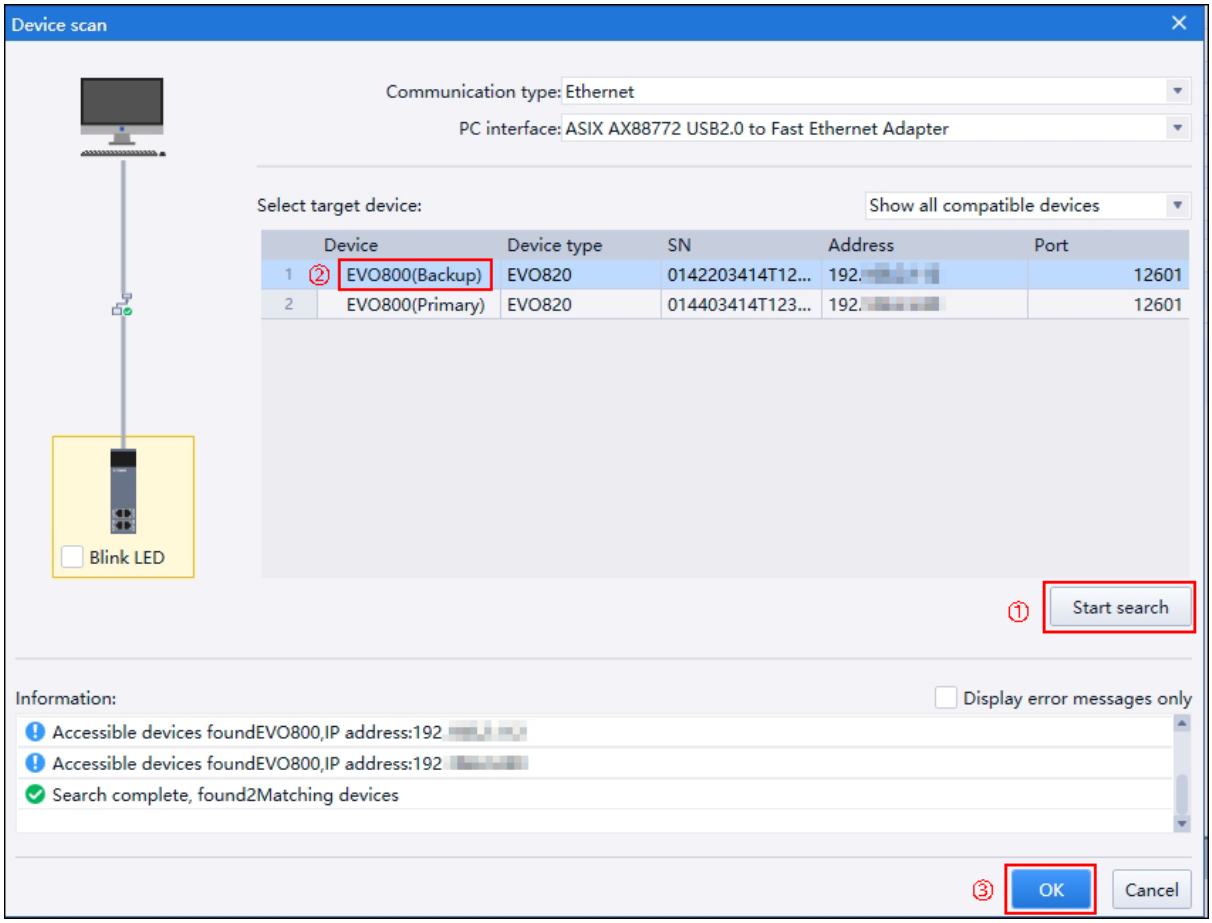


10.8.5 Redundant System Download (EVO800)

The EVO820 series PLC supports dual-rack redundancy mode. When downloading programs to a redundant system, only the main device can be selected for download. After download, the PLC host synchronizes the new program to the PLC backup unit.

Procedure

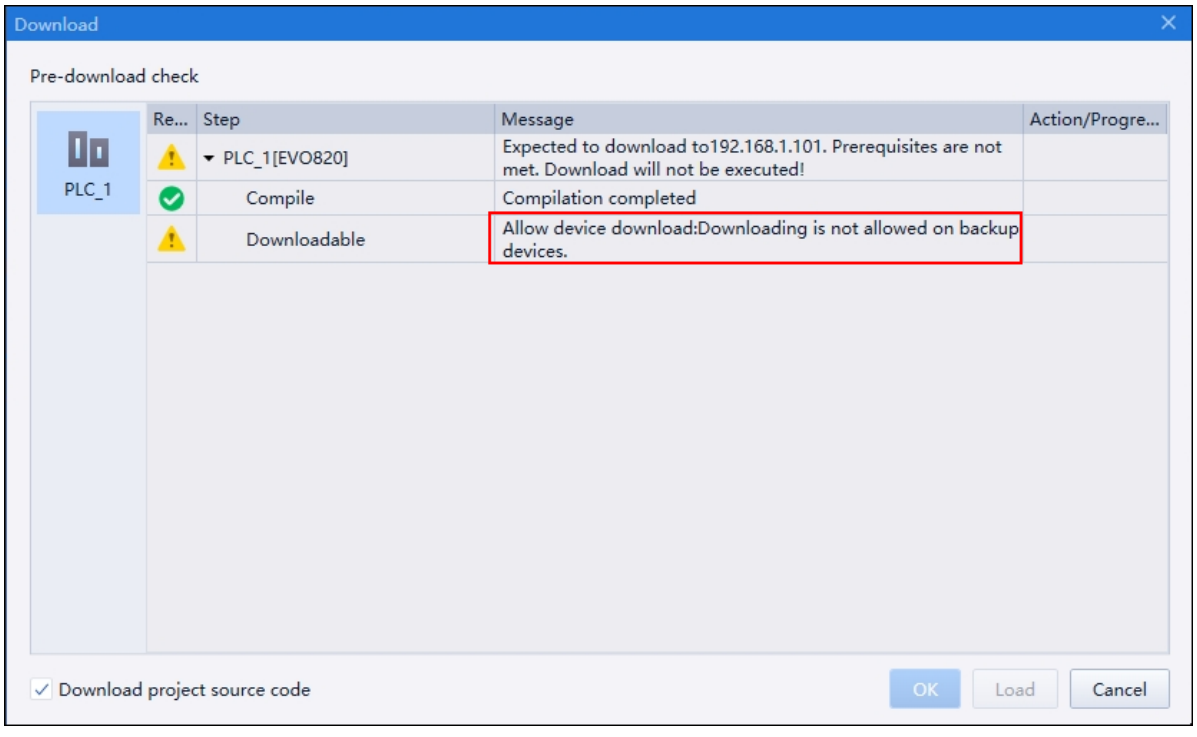
1. Right-click the first PLC device under the redundant system in the left project tree, then select "Download".
2. In the pop-up device scanning window, click "Start search", select the target PLC main device, then click "OK" to begin the download.



3. The subsequent download procedure is the same as for a single device. For details, see 10.8.3 Full Download and 10.8.4 Online Modification/Monitoring without Modifying.

NOTE

A redundant system only allows downloading to the main device. When attempting to connect to the backup device, the download interface will display a message indicating that downloading is not allowed, as shown in the figure below.



10.9 PLC Upload

10.9.1 Uploading

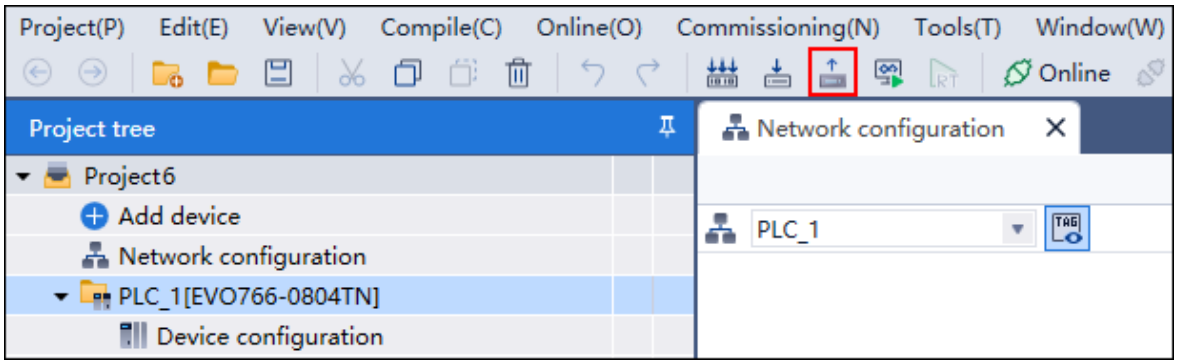
This function overwrites the existing devices in the software with the project source code (including source code, compilation information files, device configuration files, etc.) from the device.

Prerequisites

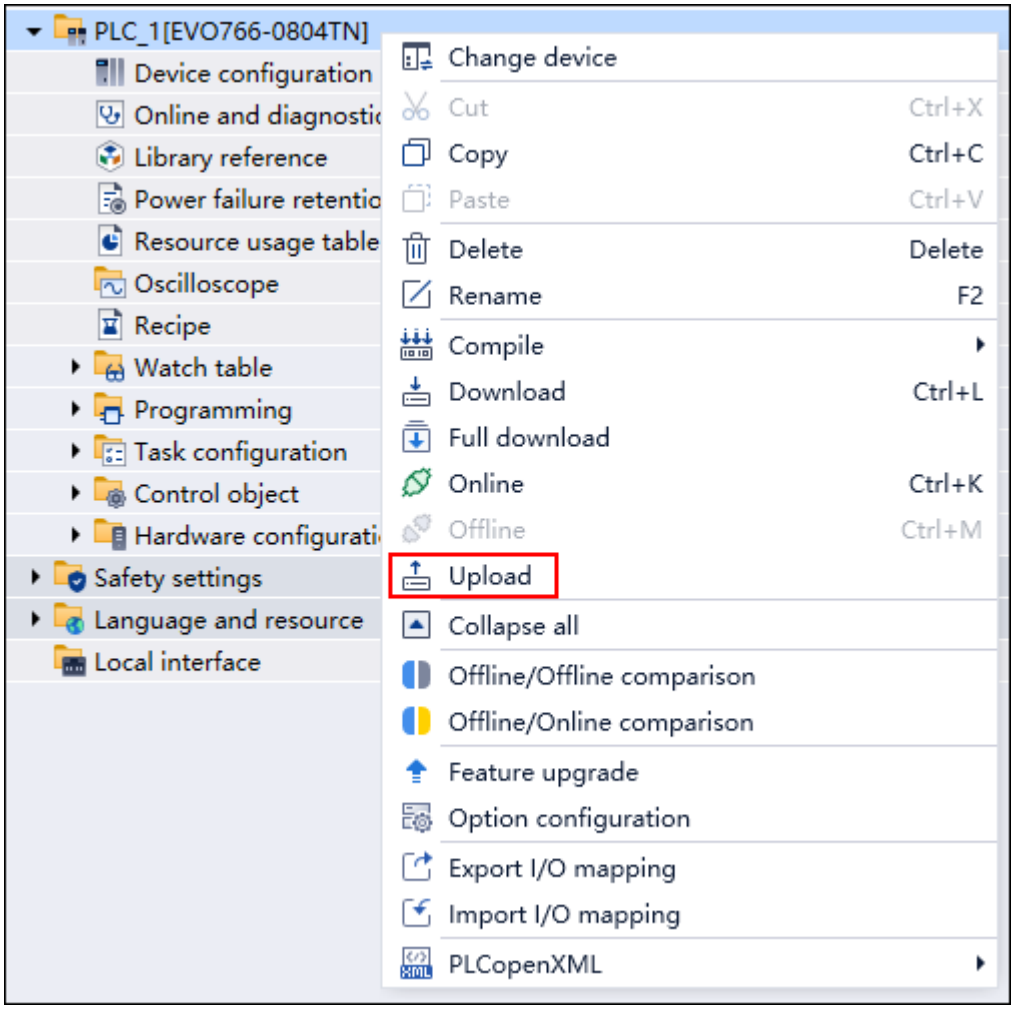
The project is in offline mode.

Procedure

1. Click "Upload".
   There are two upload entries: the toolbar and the right-click menu of the PLC node in the project tree.
   - Toolbar: Select the corresponding PLC project in the project tree, then click the "Upload" icon in the Toolbar, as shown in the figure below:

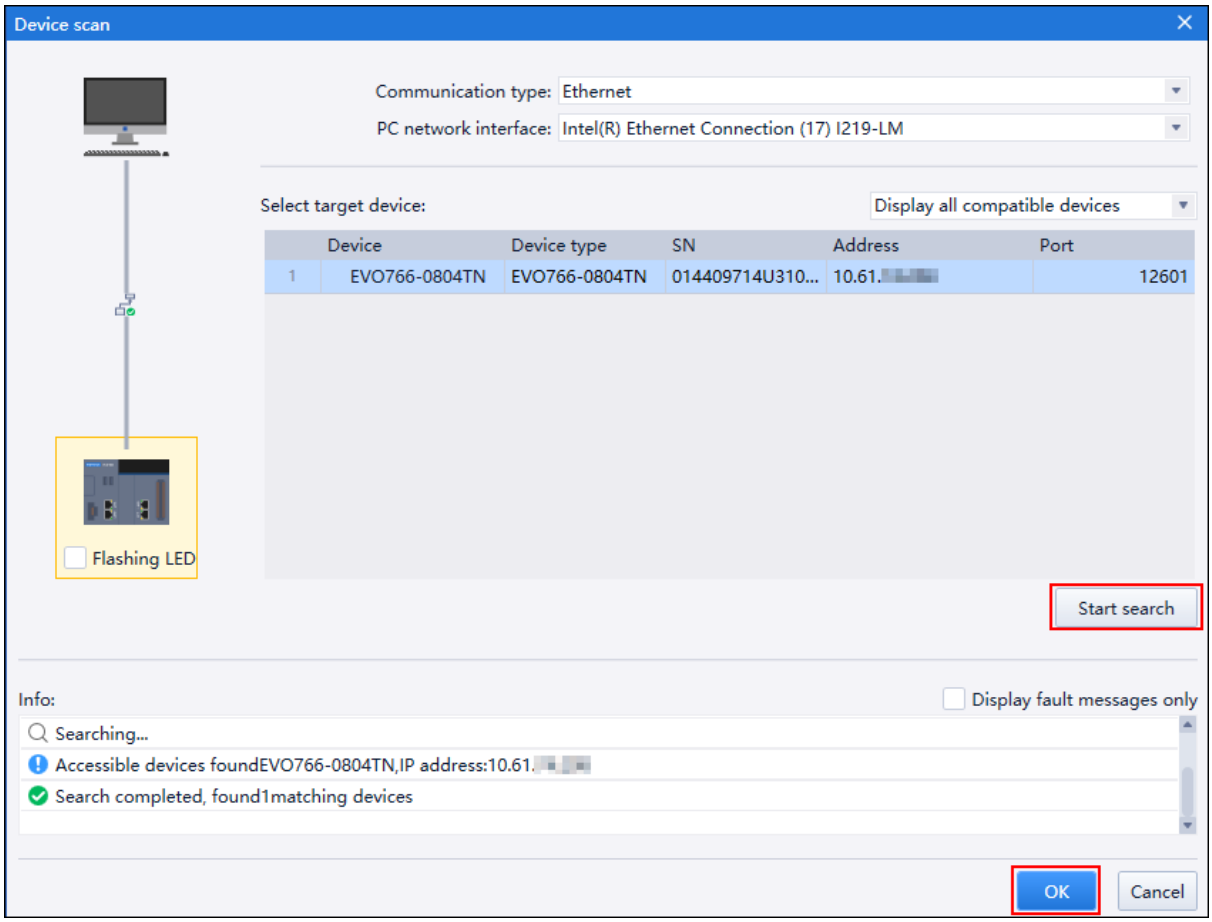


   - Right-click menu: Right-click the corresponding PLC project in the project tree, then select "Upload", as shown in the figure below:

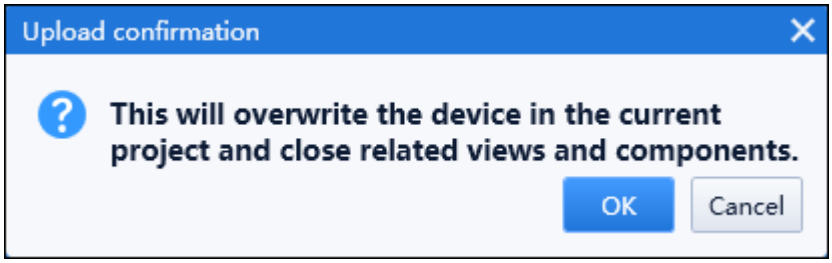


2. Upload configuration.

   In the pop-up device scanning window, click "Start search". Select the target device to upload, then click "OK", as shown in the figure below.



3. Checks before uploading.
   The system will automatically perform the following checks before uploading:
   - Whether the controller model and firmware version of the project in the PLC match the controller model and firmware version selected in the project.
   - Whether the device ID of the project in the PLC matches the device ID selected in the project.
   - Whether the project in the PLC contains source code.
4. Actions after uploading.
   - If all the above checks pass, the "Upload confirmation" window will pop up, prompting that the operation will overwrite the device in the current project and close related views and components. Click "OK" to complete the upload.



   - If any check fails, an "Error" window will pop up, indicating the reason for the error. Please troubleshoot and handle the issue based on the prompt message in the pop-up window.

10.9.2 Uploading with a New Device

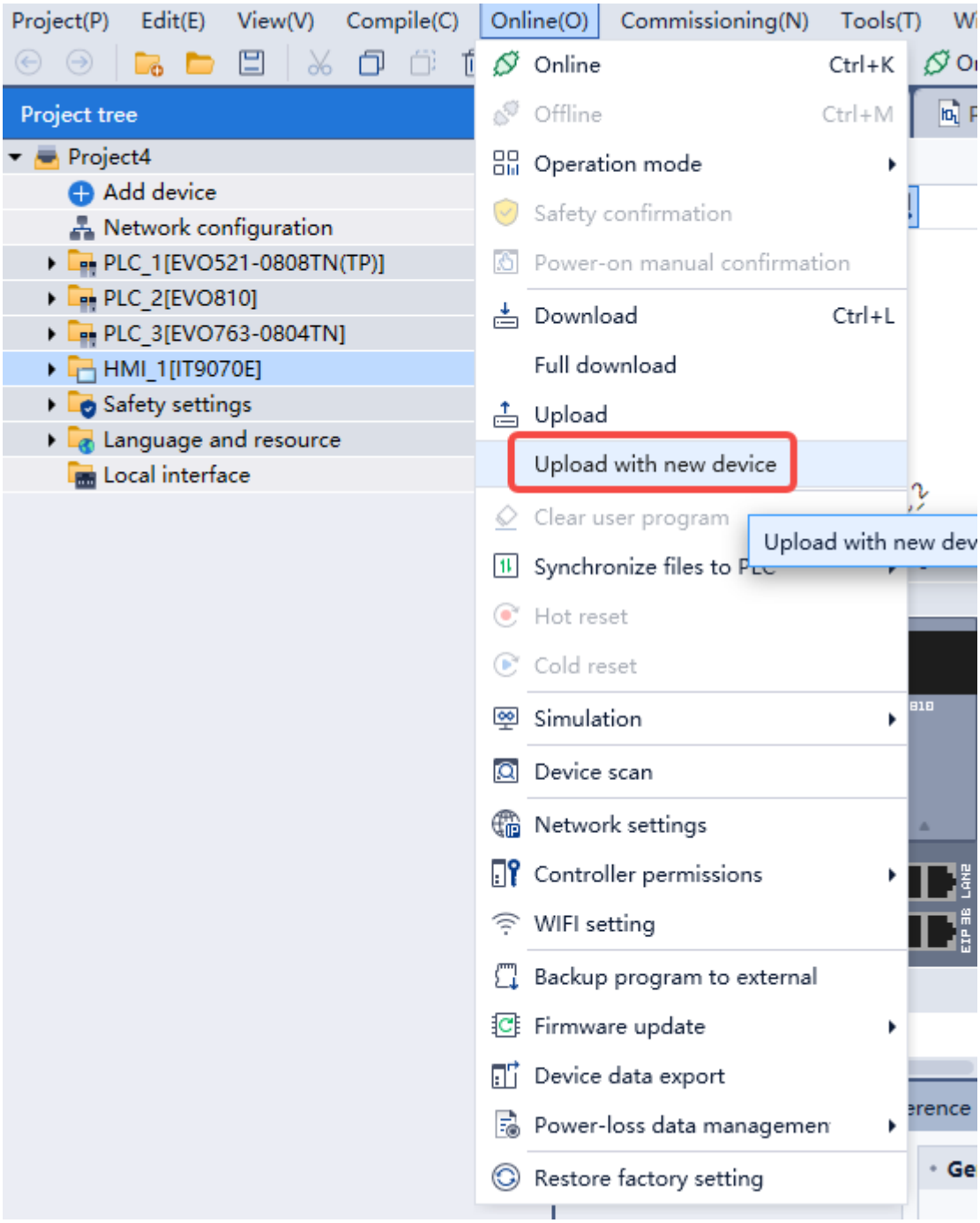
This function uploads the project source code (including source code, compilation information files, device configuration files, etc.) from the device to the software in the form of a new device.

Prerequisites

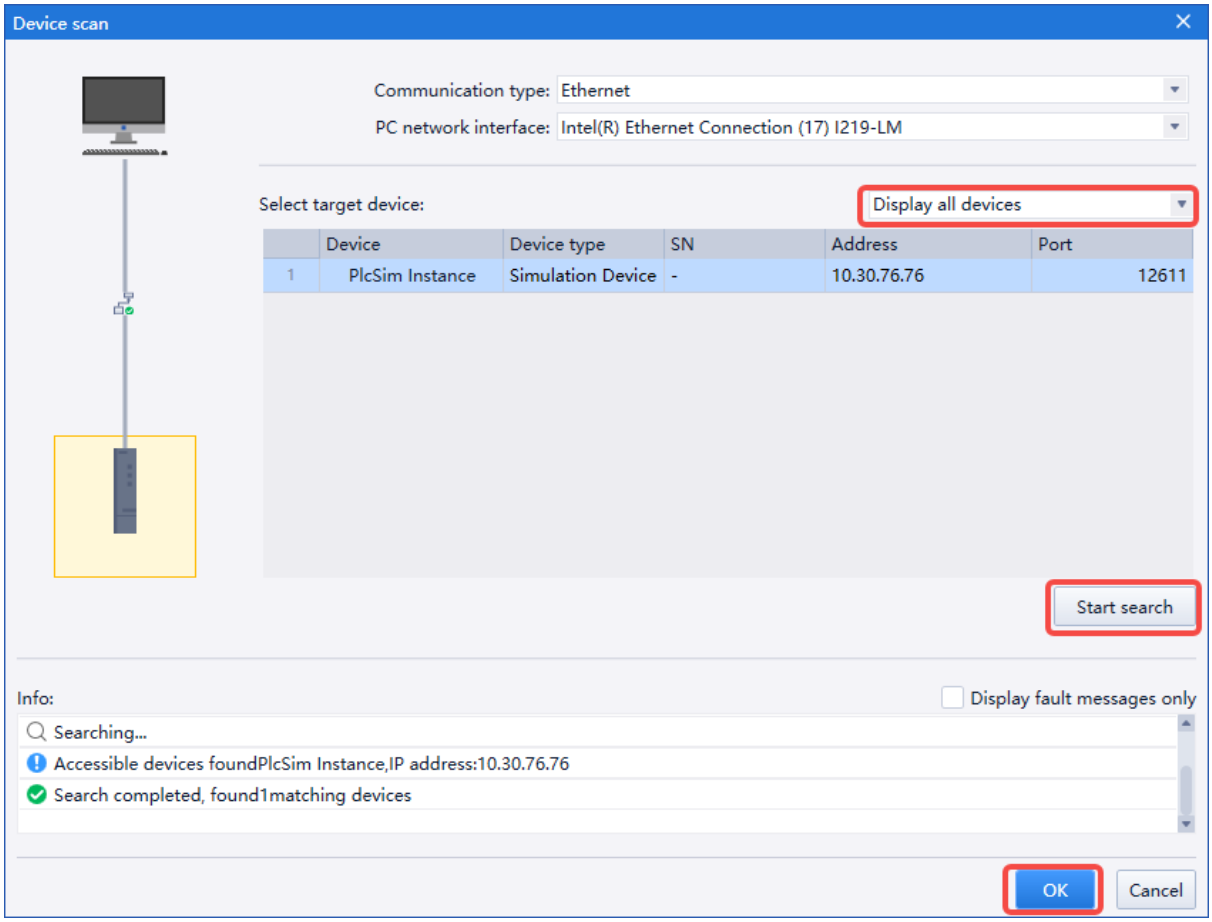
The software has established communication with the device.

Procedure

1. Select "Online > Upload with new device" in the menu bar.



2. In the pop-up device scanning window, click "Start search". The system will automatically scan for all devices and support filtering by device type. Select the target device to be uploaded and click "OK" to upload, as shown in the figure below:



3. Before uploading, the system will automatically check whether the current project contains a device with the same name or identical configuration.
4. Actions after uploading.
   - If the corresponding device does not exist in the iFA software, the "Upload confirmation" window will pop up, prompting a new device will be created in the current project. Click "OK" to complete the upload.
   - If the project contains a PLC with the same name as the current PLC, a fault will be reported, prompting that the device already exists.
   - If the project content in the PLC is the same as the current project content, a fault will be reported, prompting that the device already exists.

10.10 PLC Commissioning

10.10.1 PLC Start/Stop

Device startup

Create and deploy the application, and start the service to initiate the application.

Prerequisites

The project has been downloaded to the device.

Procedure

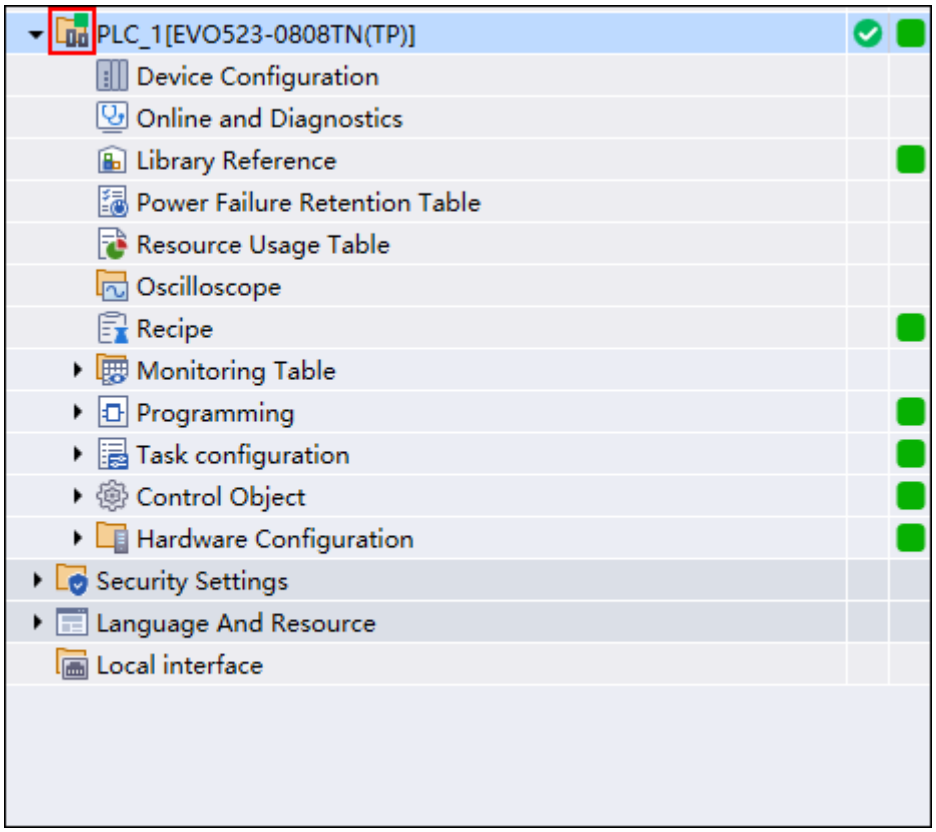
Click the "Start" icon in the toolbar; or select "Commissioning > Start" in the menu bar.

NOTE

If not in simulation mode or no breakpoints are set, a dialog box will appear asking whether to switch the PLC to RUN mode. Click "OK" to start the device.

Result

After the device starts successfully, the device identifier in the project tree changes from yellow to green.



Device stop

Create and deploy the application stop service to stop the application. After the device is started, the stop icon becomes enabled.

Prerequisites

The device has started.

Procedure

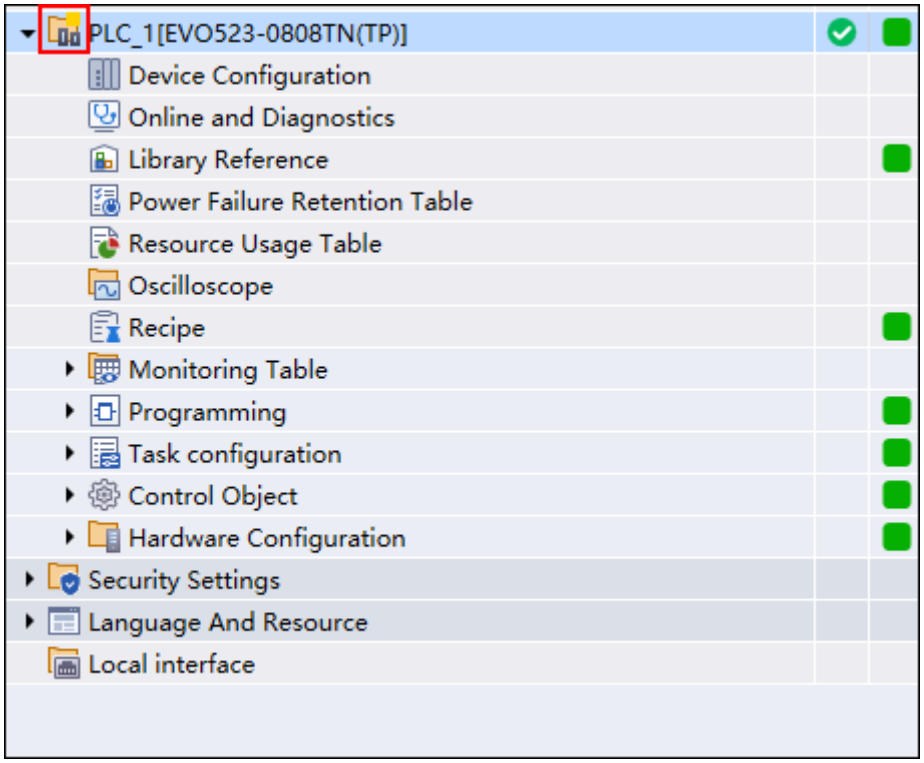
Click the "Stop" icon in the toolbar; or select "Commissioning > Stop" in the menu bar.

NOTE

Click the "Stop" icon, and a dialog box will appear asking whether to switch the PLC to STOP mode. Click "OK" to stop the device.

Result

After the device stops successfully, the device identifier in the project tree changes from green to yellow.



NOTE

When the device is manually switched from RUN status to STOP status, the power failure retention variables maintain their previous values, while other variables are reinitialized.

10.10.2 PLC Online/Offline

The online function is used to establish a connection to the device.

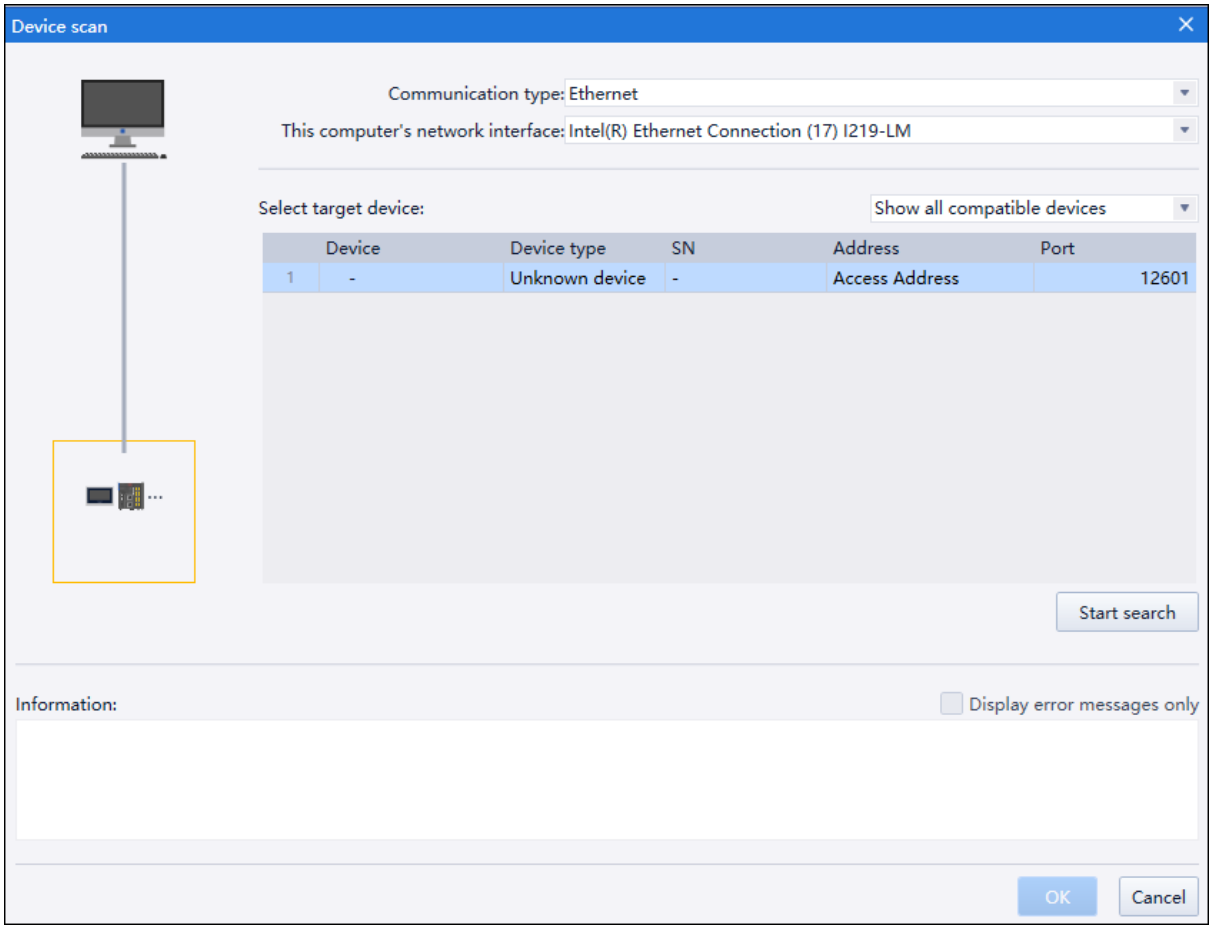
Prerequisites

- The project has been downloaded to the device.
- The device is in an offline status.

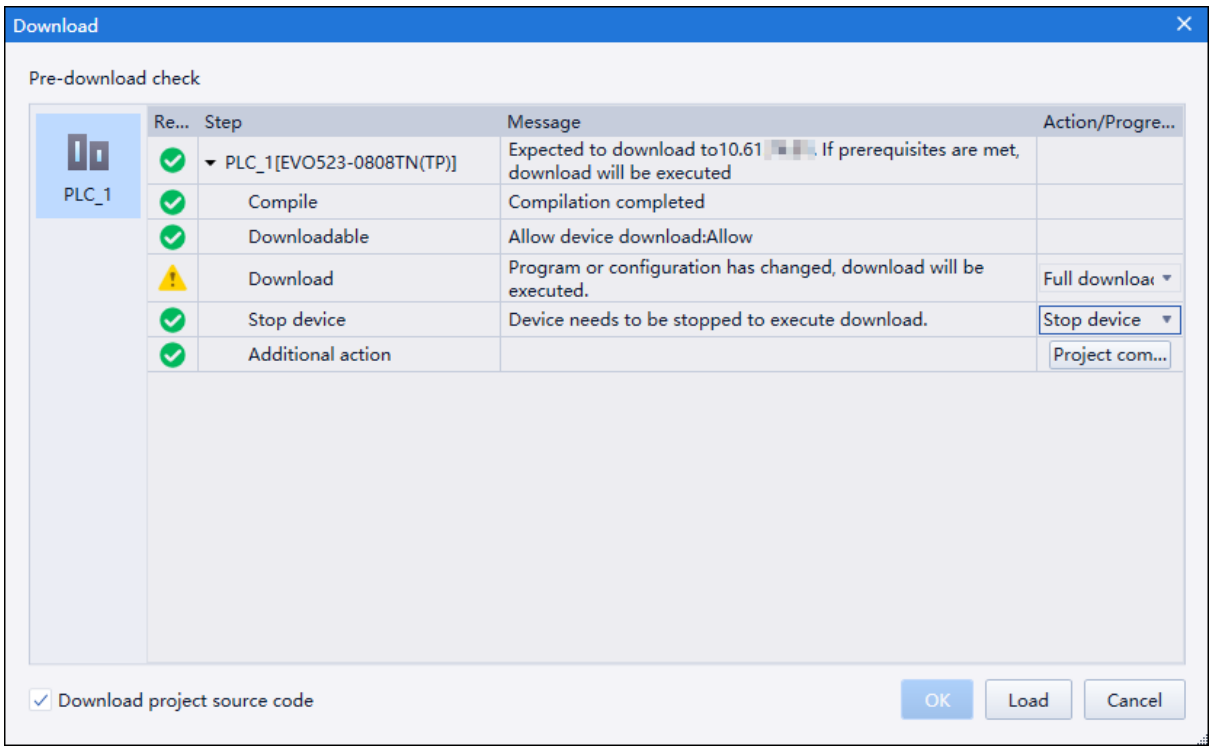
Procedure

1. Online function entry
   - After selecting the device node in the project tree, click the "Online" icon in the menu bar.
   - Right-click the device node in the project tree and select "Online".
   - After selecting the device, use the Ctrl+K shortcut key to go online; use the Ctrl+M shortcut key to go offline.
   - In the network view/device view, right-click the device and select "Online".

   The online function does not support batch operations for multiple projects. It can only be enabled for a PLC device node or its child node in the project tree. Click "Online" to establish a connection to the device.
2. Go online
   - If the device has not been configured for connection, click "Online" and the following connection configuration window will pop up.

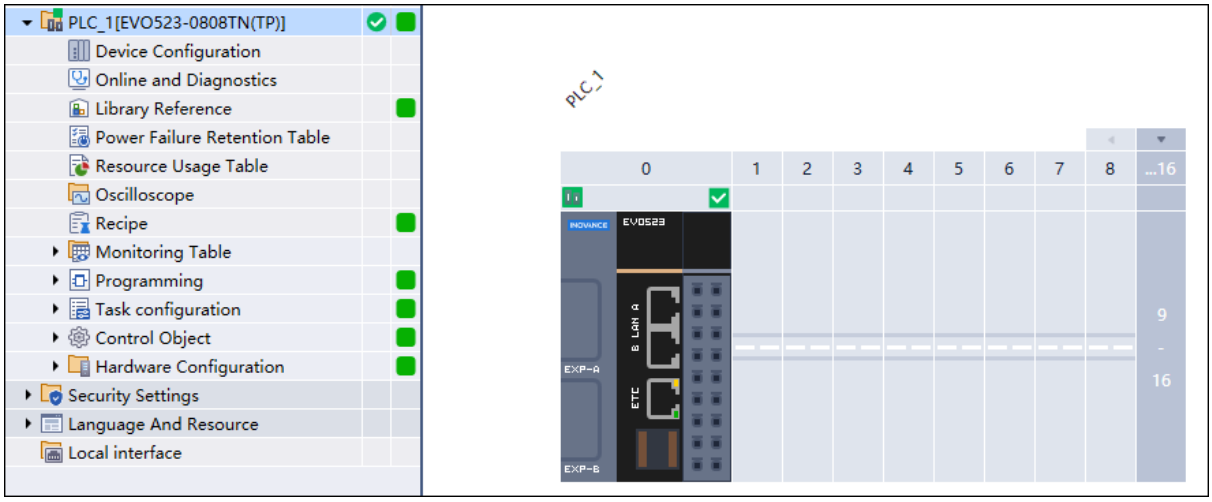


◦ After configuring the connection path, the dialog box for download ore-checks will appear. After downloading, a connection to the device is established.



3. In online mode

After going online, the device identifiers in the project tree and device configuration view will change, as shown in the figure below.



NOTE

- For the EVO800 series redundant system, only the first PLC device under the redundant system in the project tree can be selected to perform the online operation.
- During the online process, in the "Device scanning" window, you can select either the main device or backup device to establish a connection and go online. After going online, the diagnostic information and operation logs will display relevant information about the connected (main or backup) devices.
- For online projects in versions V1.4.3.5 and earlier, after upgrading the software version and going online directly, the project comparison column in the left project tree will display icon differences, which can be ignored. This column can be synchronized by performing re-download or online modification.

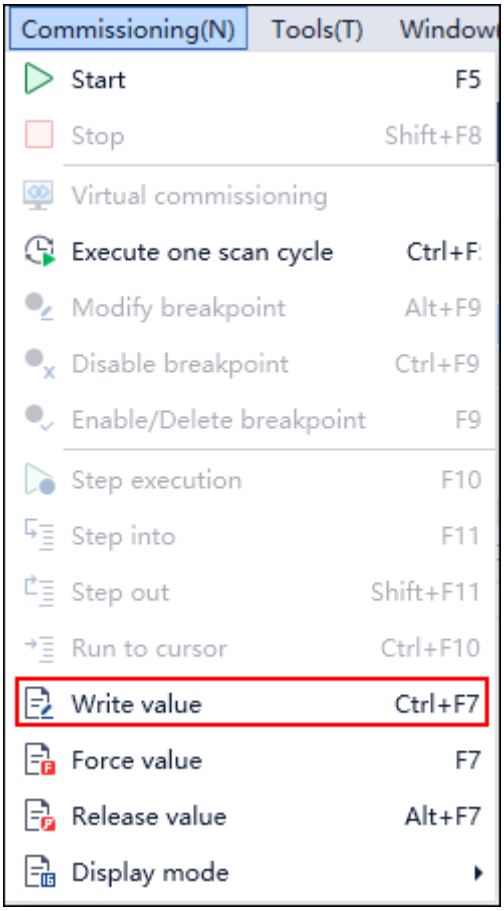
10.10.3 Value Display and Calculation

10.10.3.1 Writing a Value

When the PLC is in online running status, values can be written to variables.

Writing method

- After editing the prepared value, execute the write operation by selecting "Commissioning > Write value" in the menu bar.



- After editing the prepared value, use the shortcut key Ctrl+F7 to execute the write value operation.
- After editing the prepared value, click the "Write value" icon in the toolbar to execute the write value operation.



Result

After writing the value, the value of the variable in the variable declaration area and the program editing area will change to the edited prepared value.

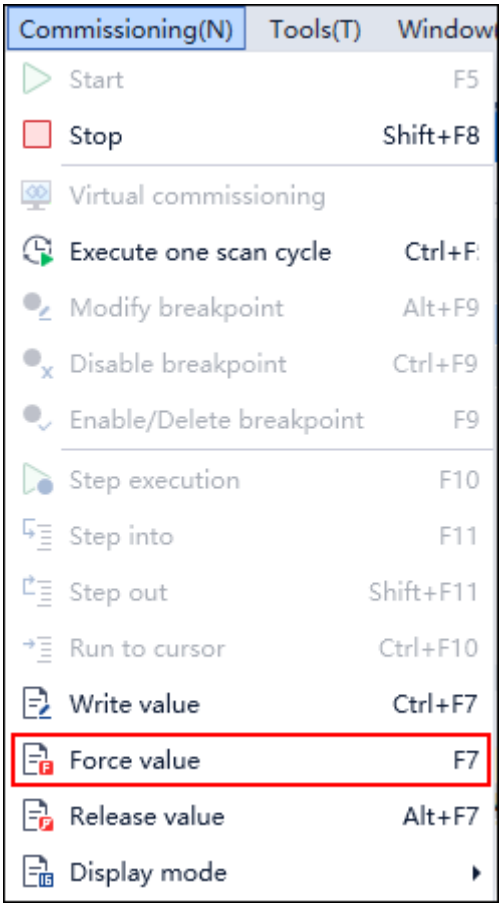
> NOTE
>
> For expressions containing POU calls, their online values cannot be viewed directly. Enter specific index numbers in the watch table to view online data.

10.10.3.2 Forcing a Value

When the PLC is in online running status, a forced value can be written to a variable.

Forcing method

- After editing the prepared value, execute the force value operation by selecting "Commissioning > Force value" in the menu bar to change the variable to the forced status.

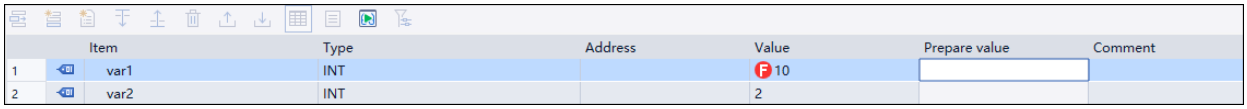


- After editing the prepared value, use the shortcut key F7 to execute the force value operation and change the variable to the forced status.

Result

After writing the forced value, the identifier is added at the front of the value. This identifier indicates that the value is a forced value, and the identifier appears in both the variable declaration area and the program editing area.

- Variable declaration area



	Item	Type	Address	Value	Prepare value	Comment
1	var1	INT		10		
2	var2	INT		2		

- Program editing area

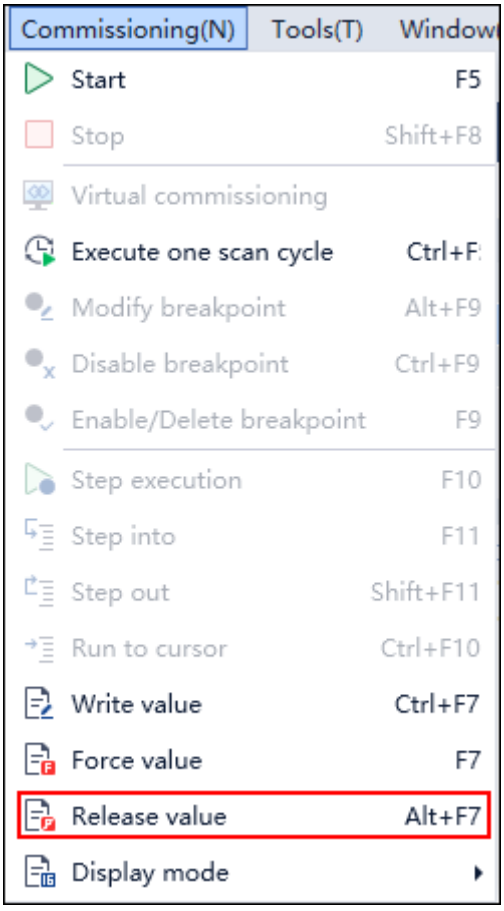


10.10.3.3 Releasing a Value

When the PLC is in online running status, executing the value release operation will release the existing prepared values and forced values, and cancel the forced statuses.

Releasing method

- Select "Commissioning > Release value" in the menu bar to release the forced values.

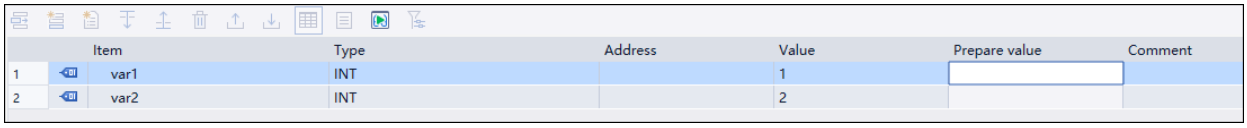


- The forced values can also be released using the shortcut key Alt+F7.

Result

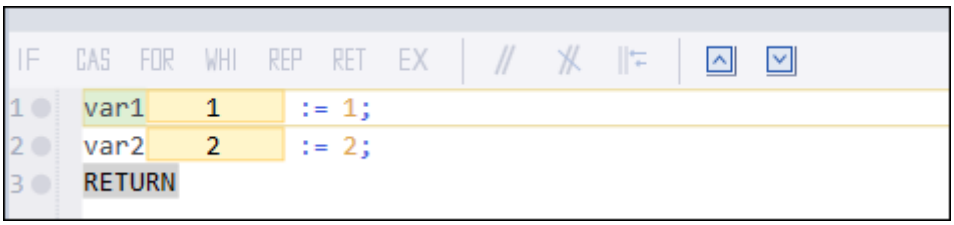
After the forced values are released, the corresponding identifiers disappear.

- Variable declaration area



	Item	Type	Address	Value	Prepare value	Comment
1	var1	INT		1		
2	var2	INT		2		

- Program editing area

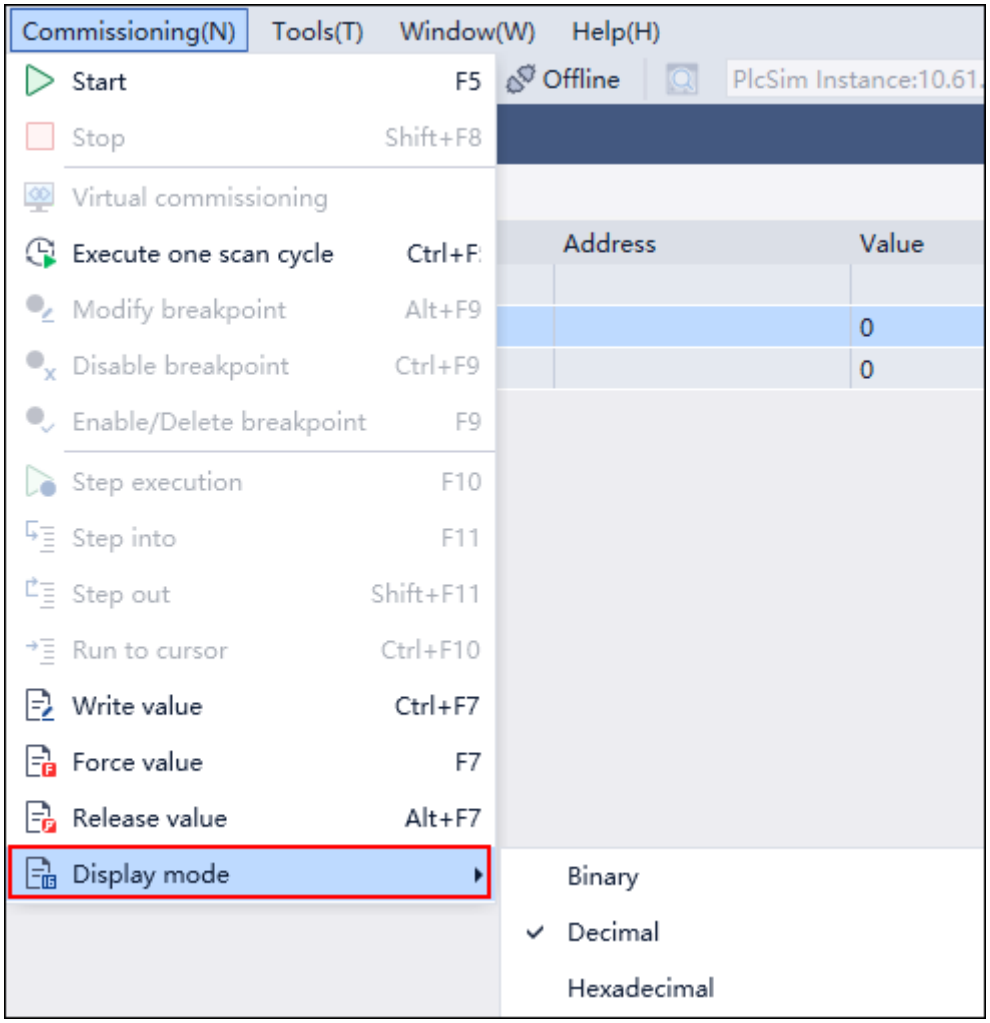


10.10.3.4 Display Modes

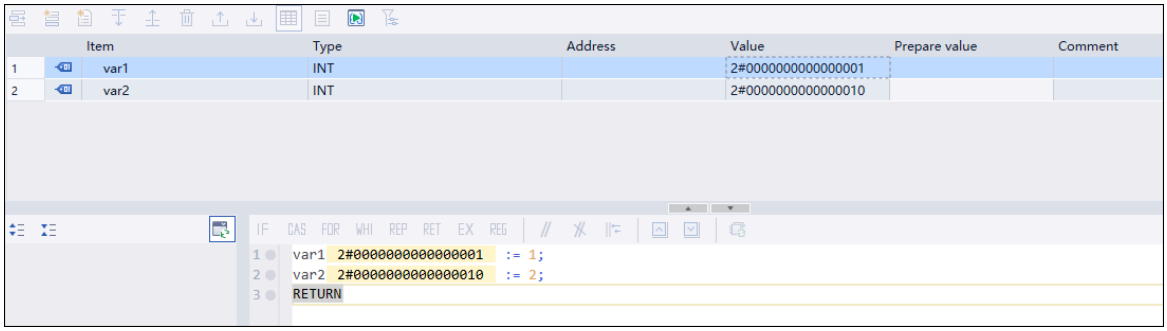
The display mode function allows switching variable values to different number bases for display. For example, this function can be used to switch a variable to binary format for observing the status of each bit.

Switching method

- Select "Commissioning > Display mode" in the menu bar to switch the display mode.



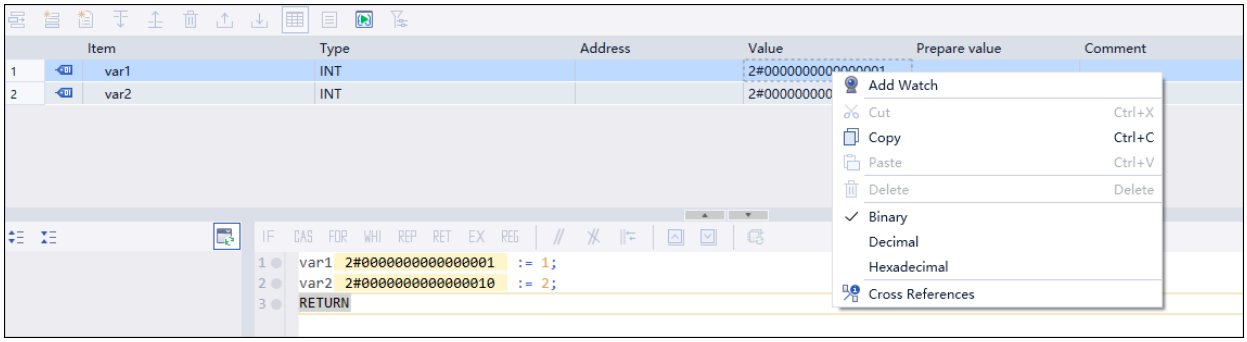
After switching, all variables are displayed in the selected format, as shown in the figure below for binary display.



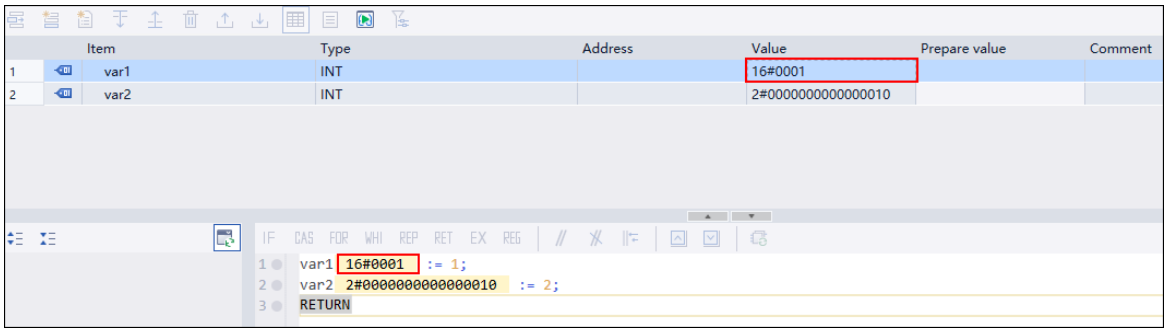
NOTE

Using this method will change the display mode for all variables.

- In online mode, right-click a variable and select a different display mode; the variable's value will then be displayed in the selected mode.



After switching, the variable is displayed in the selected mode, as shown in the figure below.



NOTE

The display mode selected from the menu bar overrides the current variable's display mode.

10.10.4 Breakpoint Commissioning

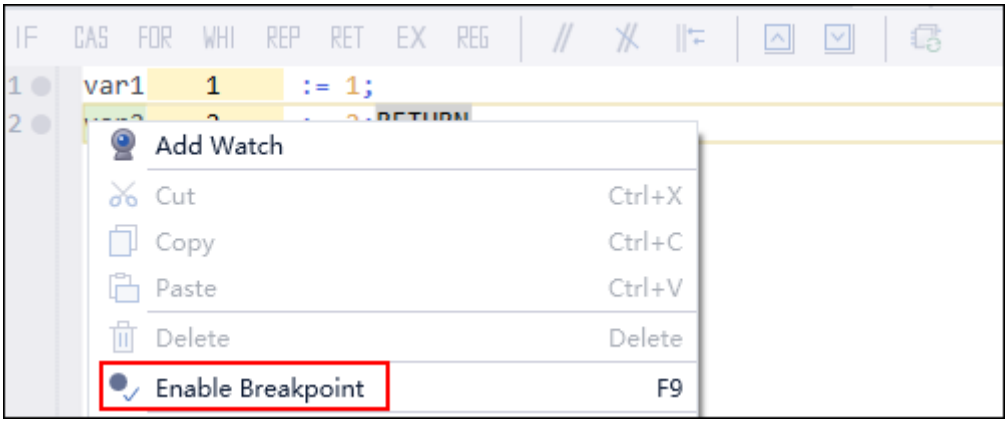
10.10.4.1 Enabling a Breakpoint

Prerequisites

The device is in an online status.

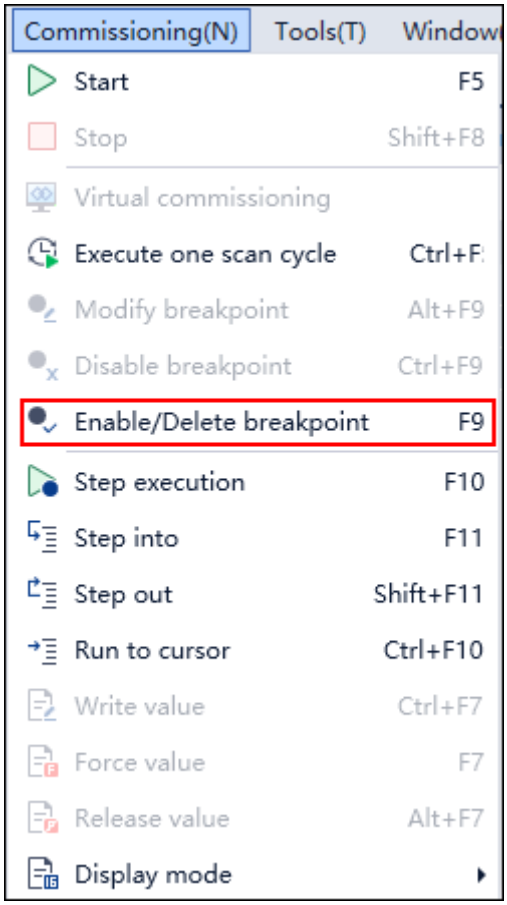
Enabling method

- Move the mouse to a statement with no enabled breakpoint (the compiler determines whether this location is valid for setting a breakpoint). Right-click and select "Enable breakpoint" to enable a breakpoint.



- Click to configure the breakpoint area (the beginning of an ST line) and add an enabled breakpoint.

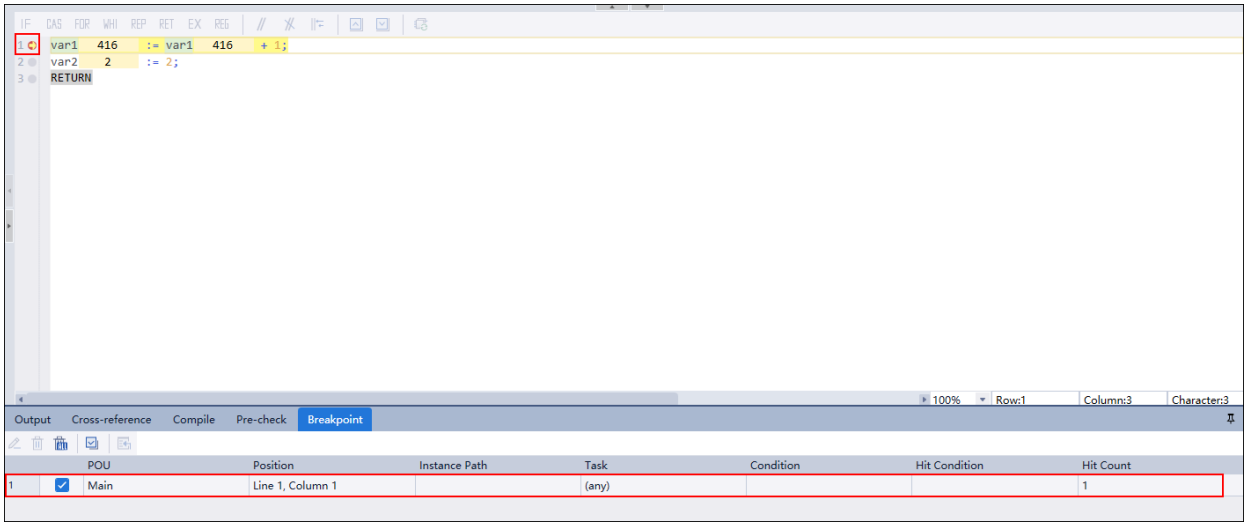
- Click a statement with no enabled breakpoint (the compiler determines whether this location is valid for setting a breakpoint), and select "Commission > Enable/Disable breakpoint" in the menu bar to enable a breakpoint.



- Click a statement with no enabled breakpoint (the compiler determines whether this location is valid for setting a breakpoint), and press the keyboard shortcut F9 to enable a breakpoint.

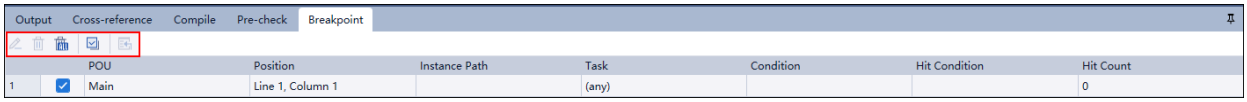
Result after enabling a breakpoint

- A breakpoint identifier appears before the corresponding statement, and a new variable (checked) is added to the breakpoint monitoring view, as shown in the figure.



The breakpoint monitoring window displays the following columns:

Name	Description
POU name	The name of the POU where the breakpoint is set, which corresponds to the POU name in the offline view
Breakpoint location	The location where the breakpoint is set, such as a specific line and column in the ST text
Instance path	The instance name of the FB type POU where the breakpoint is set
Task	The tasks in which the breakpoint is hit, defaulting to any task
Condition	Breakpoint hit condition, defaulting to always break upon hit; this condition can be a BOOL variable or an expression that evaluates to a BOOL value
Hit condition	Conditions related to the number of hits



As shown in the above figure, the breakpoint monitoring window toolbar supports modifying a breakpoint, deleting a breakpoint, deleting all breakpoints, enabling/disabling all breakpoints, and navigating to breakpoint location.

Icon	Name	Description
	Modify a breakpoint	In the breakpoint monitoring window, select the corresponding POU and click this icon to modify the breakpoint. For details, see 10.10.4.2 Modifying a Breakpoint.
	Deleting a Breakpoint	In the breakpoint monitoring window, select the corresponding POU and click this icon to delete the breakpoint. For details, see 10.10.4.3 Deleting a Breakpoint.
	Delete all breakpoints	Click this icon to delete all breakpoints at once.
	Enable/Disable all breakpoints	Click this icon to enable/disable all breakpoints at once. For details, see 10.10.4.1 Enabling a Breakpoint and 10.10.4.4 Disabling a Breakpoint.
	Navigate to breakpoint location	In the breakpoint monitoring window, select the corresponding POU and click this icon to jump directly to the breakpoint location.

CAUTION

- When the breakpoint view switches from online to offline, the list is cleared; when switching back to online, the breakpoints from the current PLC are reloaded, and the reloaded breakpoints are disabled by default.
- When multiple PLCs are online simultaneously, the breakpoint list switches when switching between PLC1 and PLC2.

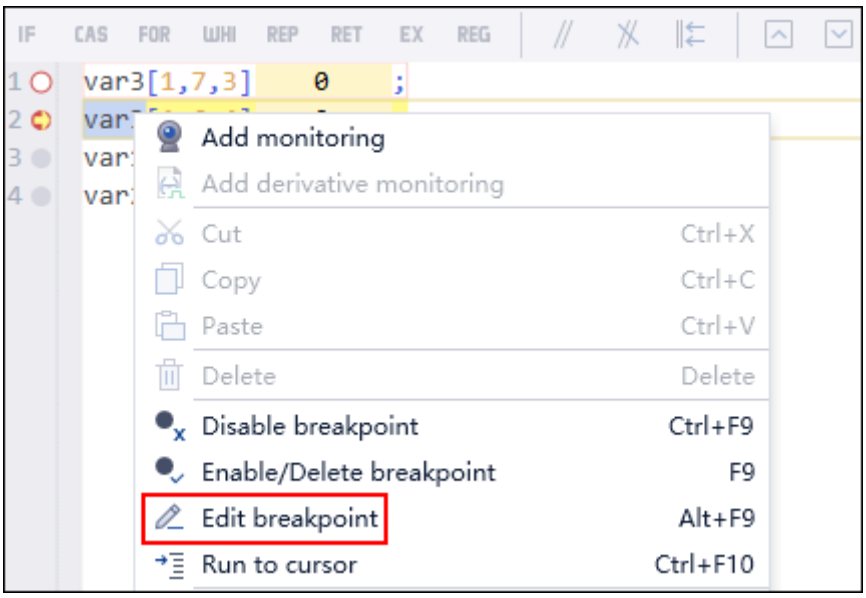
10.10.4.2 Modifying a Breakpoint

Prerequisites

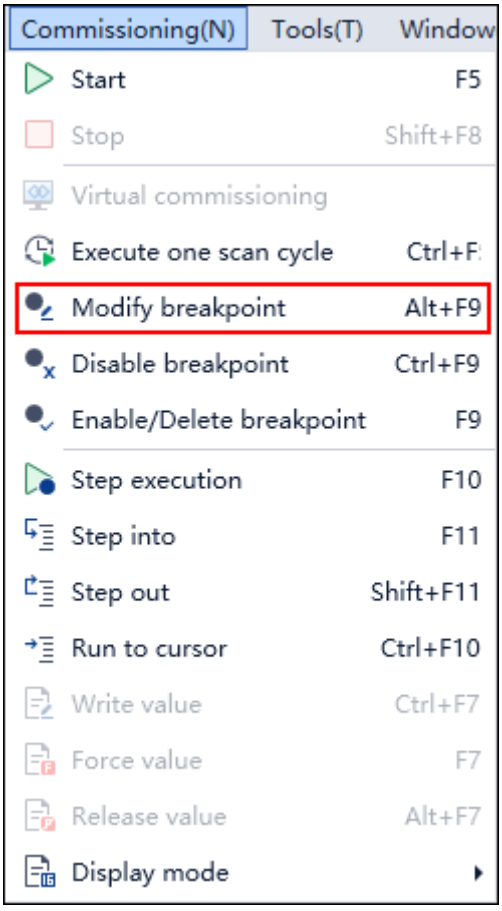
The breakpoint is enabled.

Modifying method

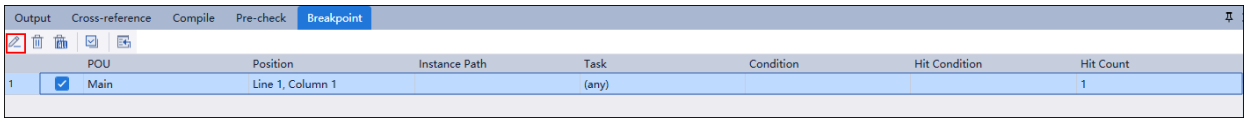
- Move the mouse to a statement location with an enabled/disabled breakpoint, right-click and select "Modify breakpoint" to open the breakpoint modification window.



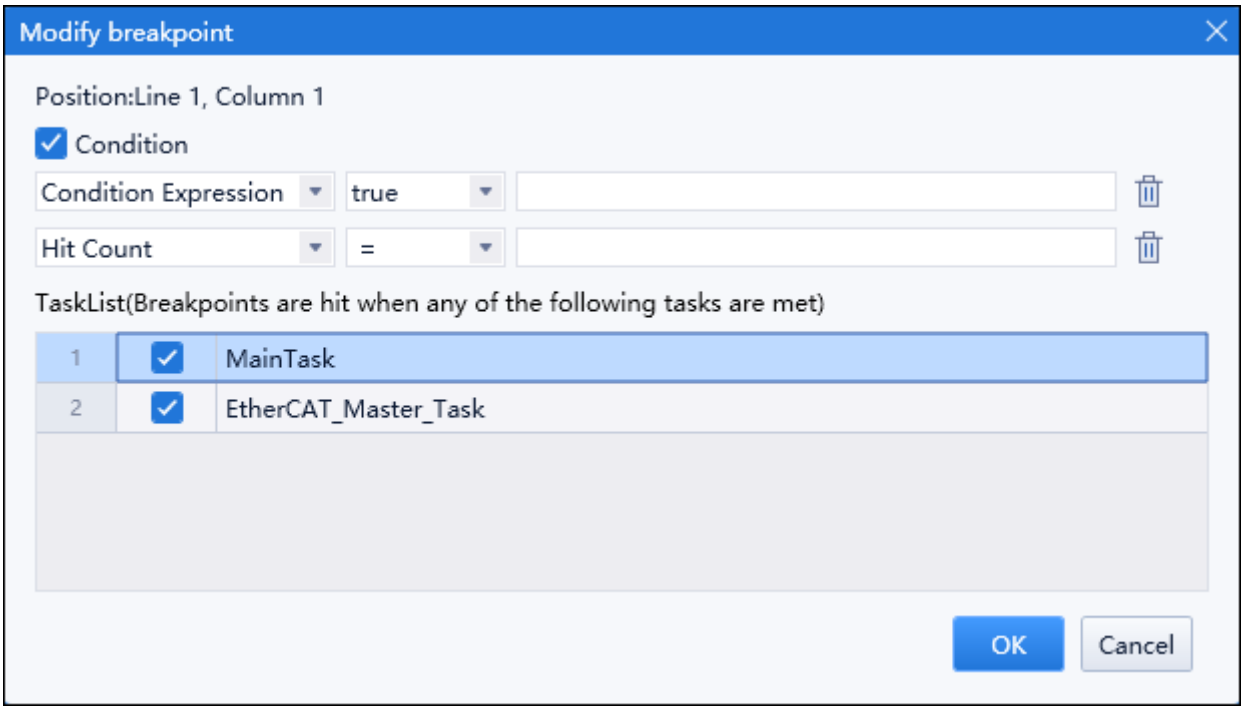
- Click a statement location with an enabled/disabled breakpoint and select "Commissioning > Modify breakpoint" from the menu bar to open the breakpoint modification window.



- Click a statement location with an enabled/disabled breakpoint and use the shortcut key Alt+F9 to open the breakpoint modification window.
- After adding a breakpoint, its information is displayed in the breakpoint monitoring window, where you can modify the breakpoint. In the breakpoint monitoring window, select the corresponding POU and click the "Modify breakpoint" icon in the toolbar.

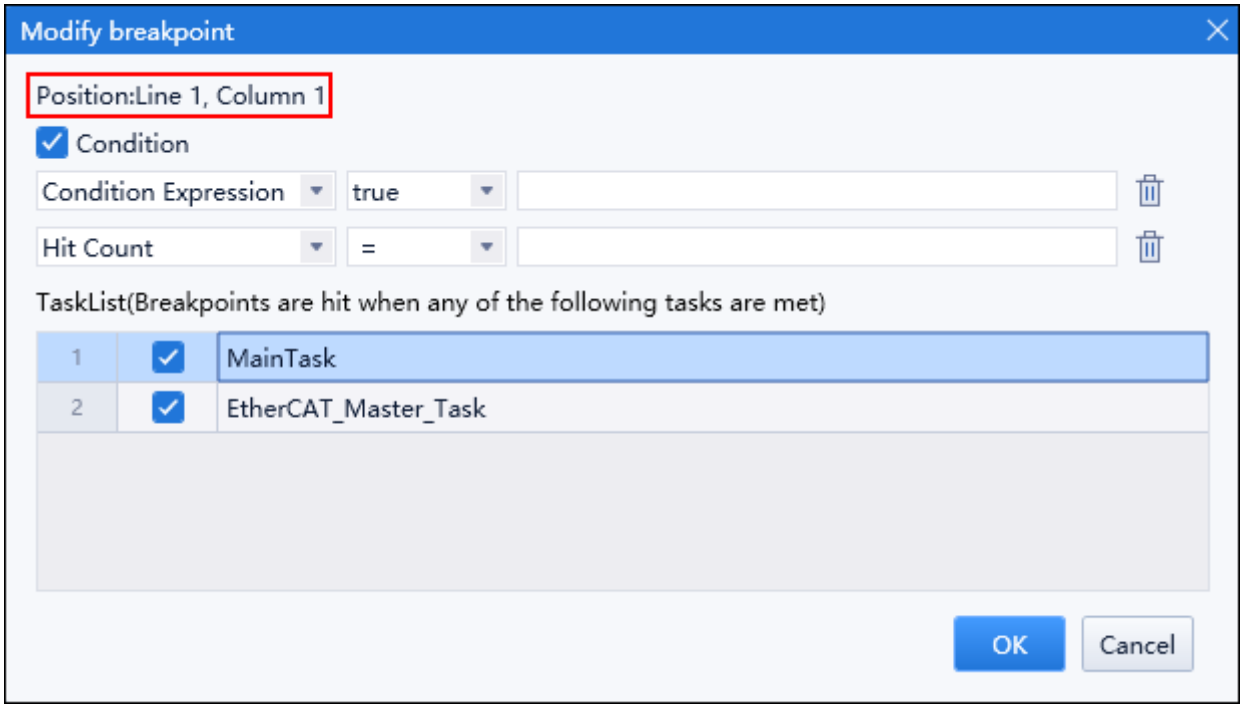


The following figure is the breakpoint modification window.

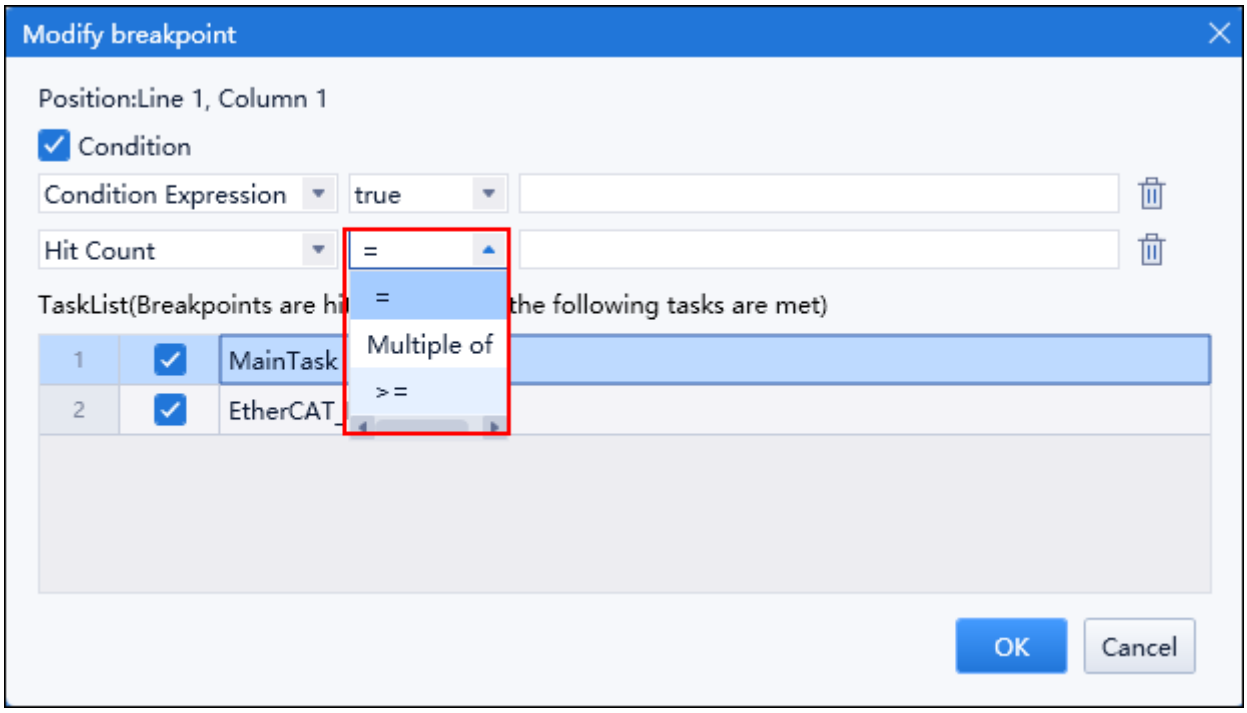


The breakpoint modification window contains the following items:

- Location information

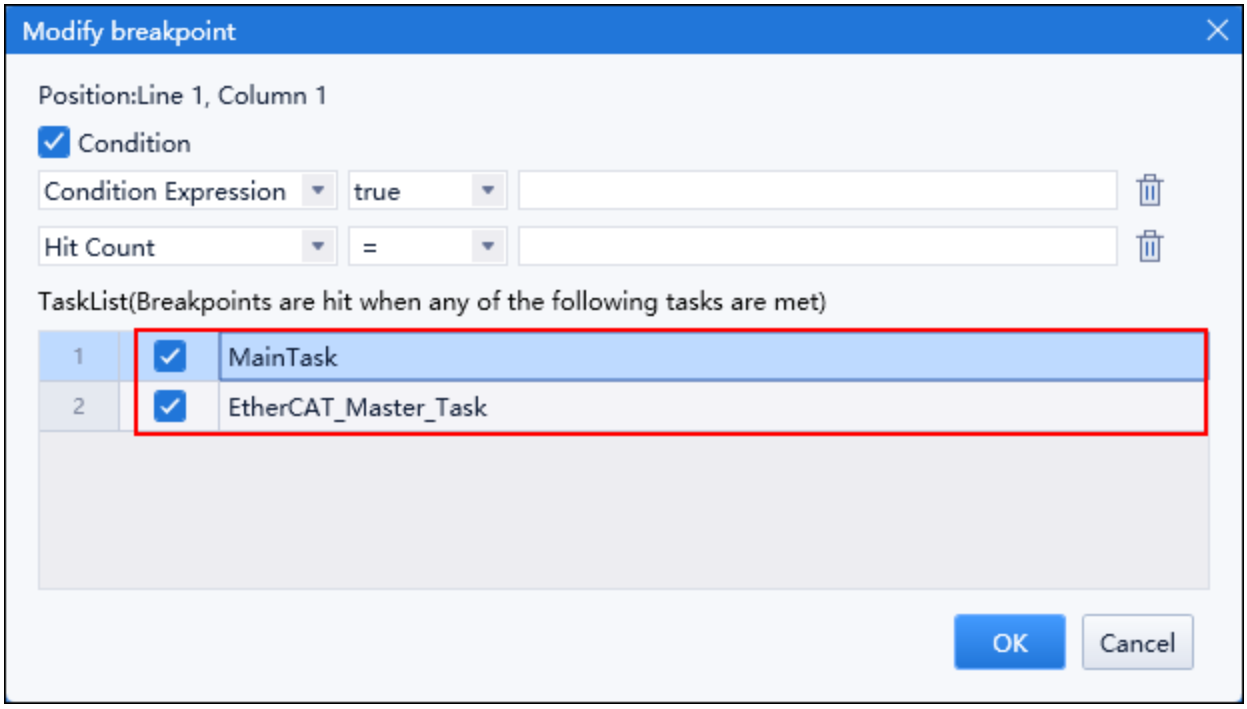


- Condition: Condition expression and hit count



Hit count	Determines when to stop the program based on how many times the breakpoint location has been executed.
Not configured	Always stop the program upon reaching the breakpoint location
Equal to n	Stop when the hit count reaches n
Multiple of n	Stop when the hit count is a multiple of n
Greater than or equal to n	Stop when the hit count is greater than or equal to n

- Task



CAUTION

- Expression condition: The breakpoint is hit when the condition expression evaluates to true. The condition expression can be a complex expression but must be of BOOL type after validation.
- You can select either the expression condition or the hit count condition, or set both. (Note: When both the expression condition and the hit count condition are set, the breakpoint is hit when the number of hits satisfying the expression condition reaches the hit count condition.)
- Task: All tasks are displayed in the list. Users can select one or more tasks at which the breakpoint will stop. If not set, the breakpoint will be hit in any task. (Note: When multiple tasks are selected, if one of the tasks is deleted, the corresponding data in the breakpoint view will be refreshed; if the breakpoint is set for only one task, when that task is deleted, the breakpoint will be deleted synchronously. When all tasks are selected, the task option in the view is not selected by defaulted, and the list of selected tasks is not expanded.)

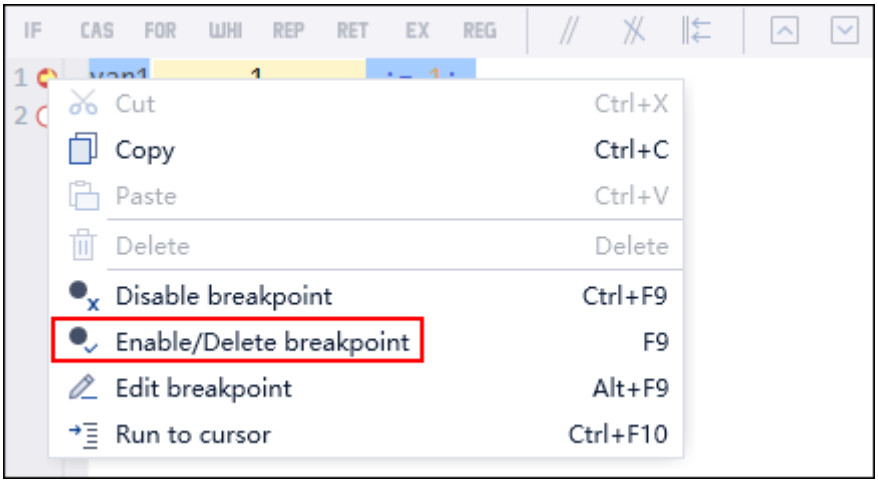
10.10.4.3 Deleting a Breakpoint

Prerequisites

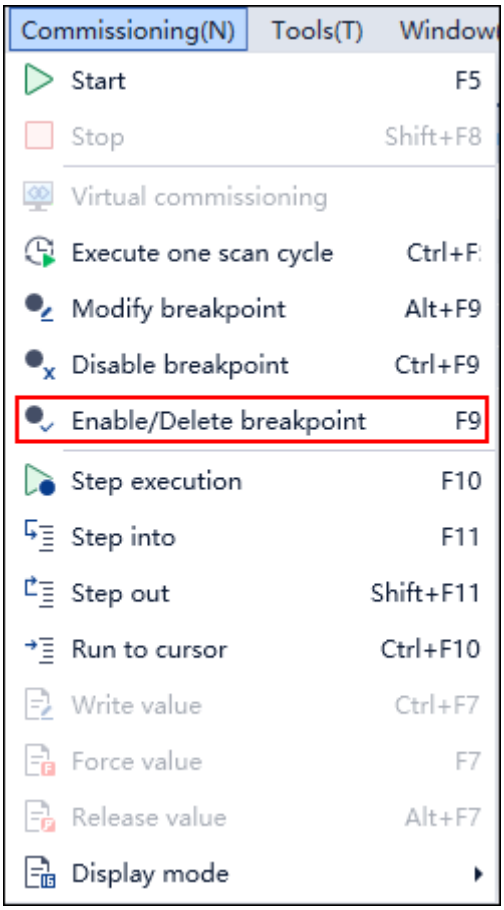
The breakpoint is enabled.

Deleting method

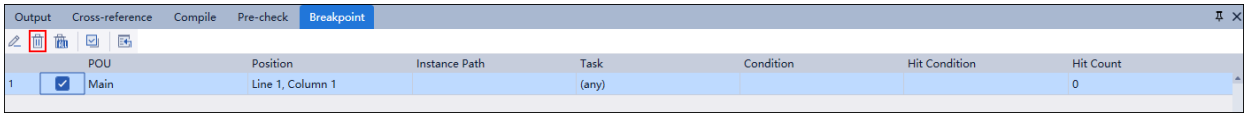
- Move the mouse to a statement location with an enabled breakpoint, right-click and select "Enable/Delete breakpoint" to delete the breakpoint.



- Click a statement location with an enabled breakpoint and select "Commissioning > Enable/Delete breakpoint" to delete the breakpoint.

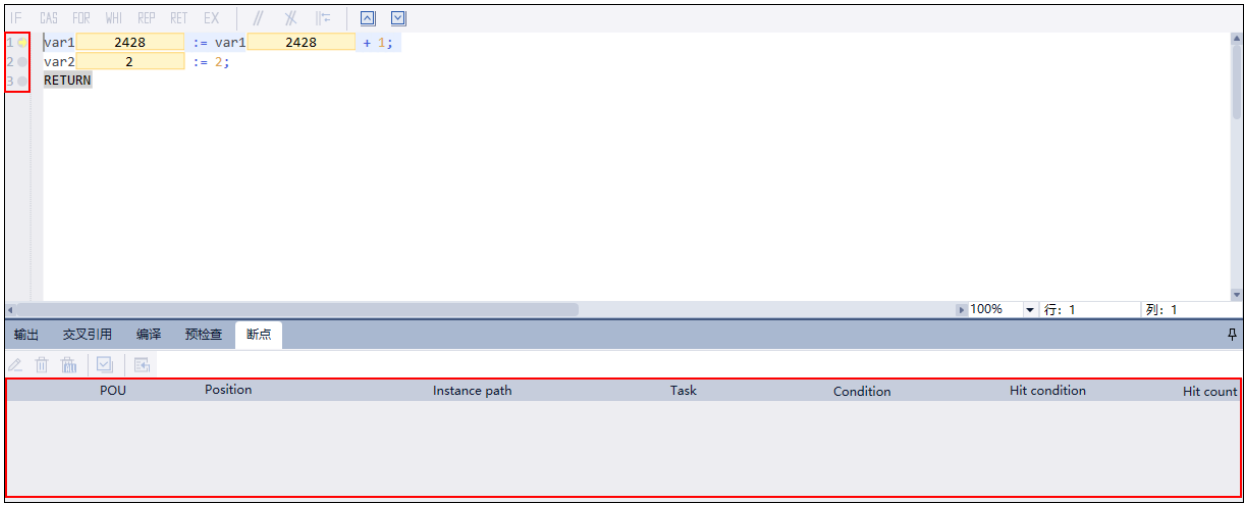


- Click a statement location with an enabled breakpoint and press the shortcut key F9 to delete the breakpoint.
- After adding a breakpoint, its information is displayed in the breakpoint monitoring window, where you can delete the breakpoint. In the breakpoint monitoring window, select the corresponding POU and click the "Delete breakpoint" icon in the toolbar.



Result after deleting a breakpoint

The identifier in front of the statement disappears, and the corresponding breakpoint disappears from the breakpoint monitoring window.



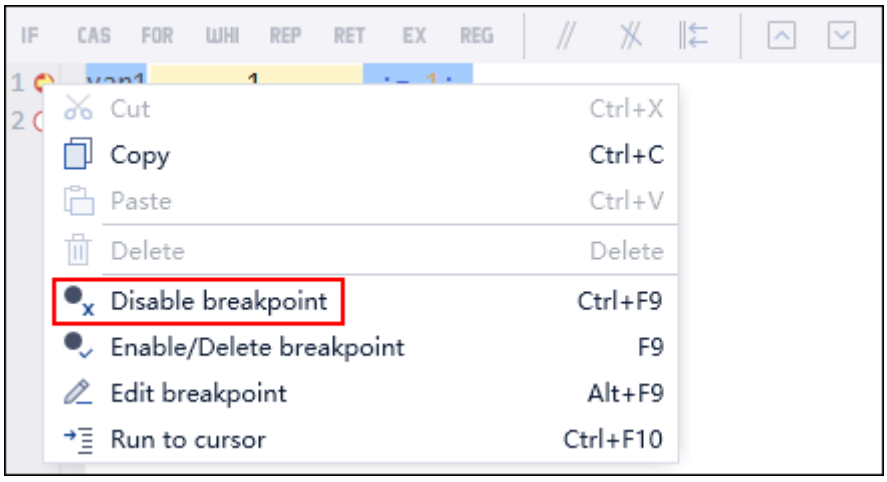
10.10.4.4 Disabling a Breakpoint

Prerequisites

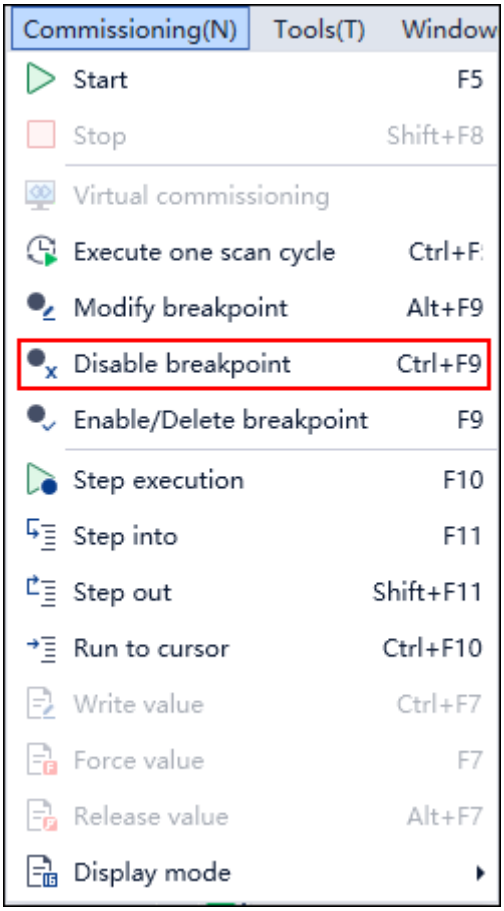
A breakpoint has been set and is enabled.

Disabling method

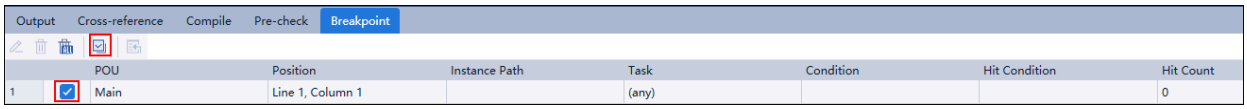
- Move the mouse to a statement location with an enabled breakpoint, right-click and select "Disable breakpoint" to disable the breakpoint.



- Move the mouse to a statement location with an enabled breakpoint, select "Commissioning > Disable breakpoint" from the menu bar, and the breakpoint disabling window will pop up.

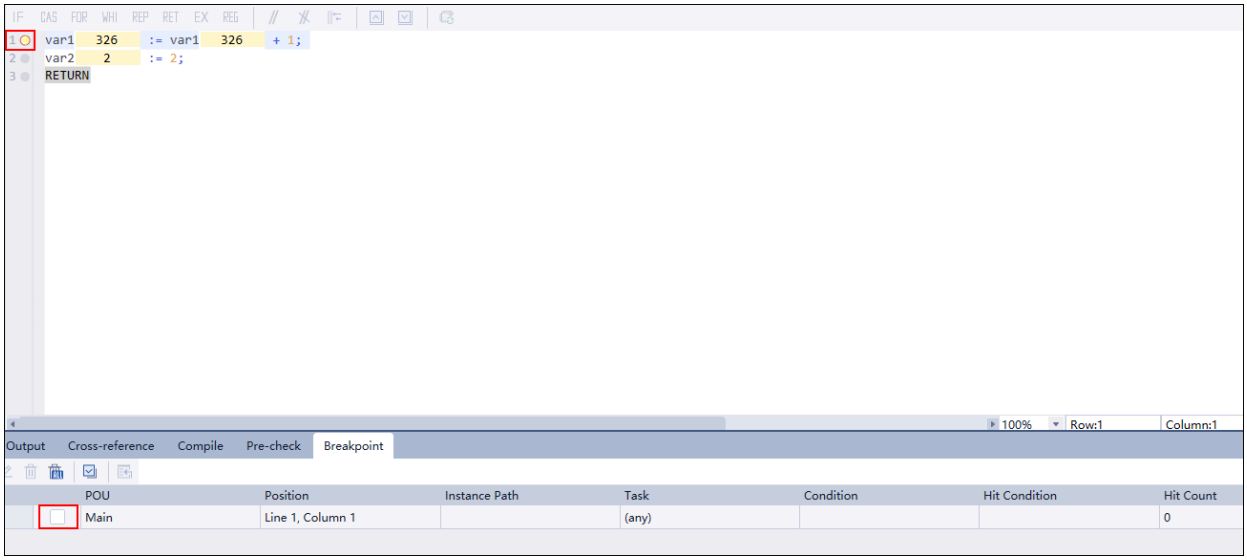


- Select an enabled breakpoint and click the breakpoint area (the beginning of an ST line) to disable the breakpoint.
- Move the mouse to a statement location with an enabled breakpoint and use the shortcut Ctrl+F9 to disable the breakpoint.
- After adding a breakpoint, its information is displayed in the breakpoint monitoring window, where you can disable the breakpoint in the following two ways:
  - Click the "Enable/Disable breakpoint" icon in the toolbar.
  - Uncheck the checkbox to the left of the breakpoint.



Result after disabling a breakpoint

A red hollow circle indicating a disabled breakpoint appears before the corresponding statement, and the corresponding POU in the breakpoint monitoring window is no longer checked.



10.10.5 ST Program Commissioning

10.10.5.1 Step Execution

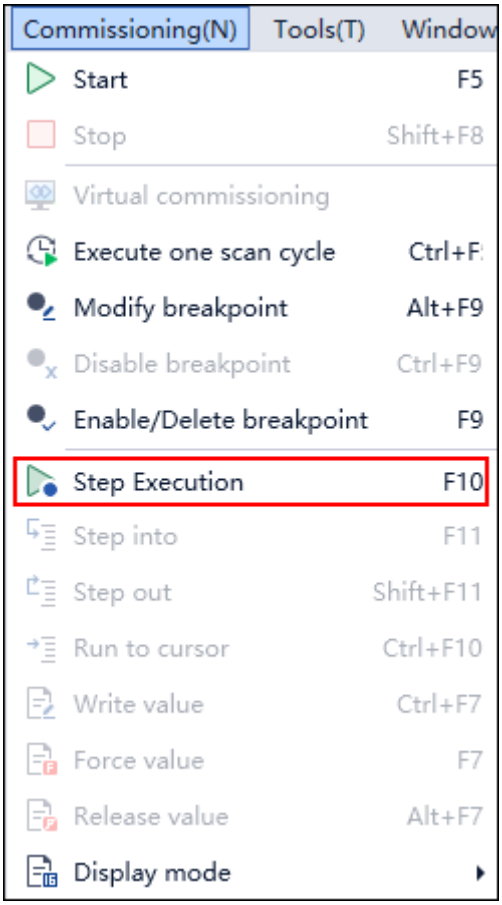
Execute one statement in the order of program statements, without entering methods/functions.

Prerequisites

The program is in breakpoint commissioning status.

Process

- Move the mouse to the statement location and select "Commissioning > Step execution" in the menu bar to execute the statement.



- Move the mouse to the statement location and use the shortcut key F10 to execute the statement.
- Move the mouse to the statement location and click the "Step execution" icon in the toolbar to execute the statement.



10.10.5.2 Running to Cursor

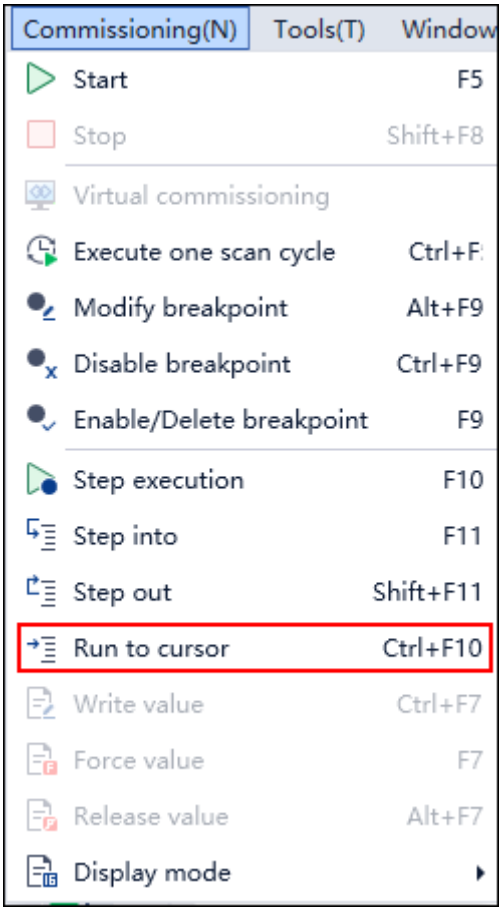
The program runs to the location of the cursor. If there is a breakpoint between the stop statement and the cursor, it runs first to the breakpoint in between.

Prerequisites

The program is in breakpoint commissioning status.

Process

- Move the mouse to the statement location, select "Commissioning > Run to cursor" in the menu bar, and the program will run to the location of the cursor.



- Move the mouse to the statement location and use the shortcut key Ctrl+F10.
- Move the mouse to the statement location and click the "Run to cursor" icon in the toolbar.



10.10.5.3 Displaying Current Statement

Use this function to jump to the statement where the pointer stops during breakpoint commissioning.

Prerequisites

The program is in breakpoint commissioning status.

Process

Click the "Display current statement" icon in the toolbar to jump to the statement where the pointer stops.



10.10.5.4 Stepping Out

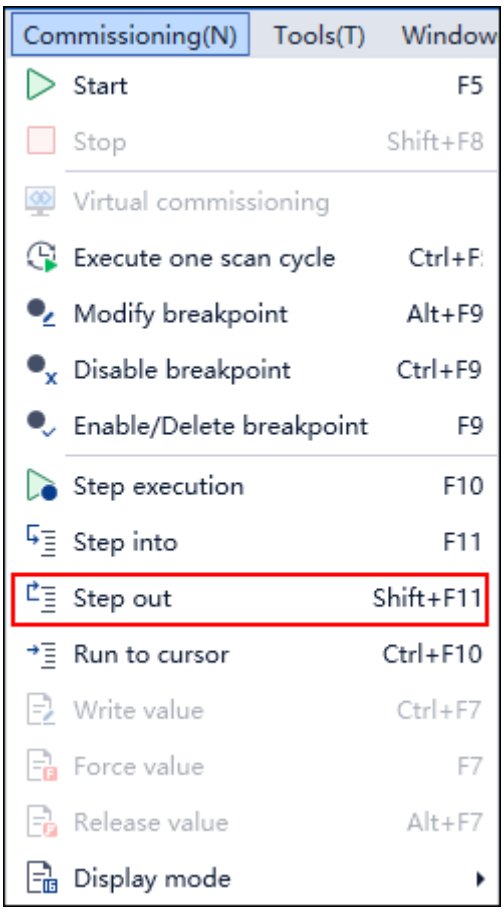
Statement exits the method/function directly. If there are breakpoints between the stop statement of the method/function and the end, the program first runs to the intermediate breakpoints. If there are no intermediate breakpoints, it exits the call hierarchy level by level.

Prerequisites

The program is in breakpoint commissioning status.

Process

- Move the mouse to the statement location, select "Commissioning > Step out" in the menu bar, and program exits the call hierarchy level by level.



- Move the mouse to the statement location and use the shortcut key Shift+F11.
- Move the mouse to the statement location and click the "Step out" icon in the toolbar.



10.10.5.5 Stepping Into

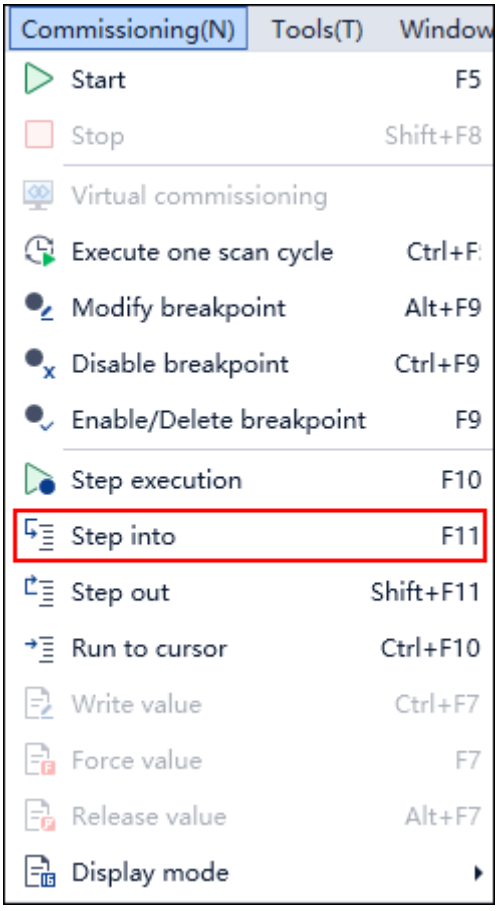
Execute one statement in the order of program statements and enter methods/functions encountered during execution.

Prerequisites

The program is in breakpoint commissioning status.

Process

- Move the mouse to the statement location, select "Commissioning > Step into" in the menu bar, and the program executes one statement in the order of program statements.



- Move the mouse to the statement location and use the shortcut key F11.
- Move the mouse to the statement location and click the "Step into" icon in the toolbar.

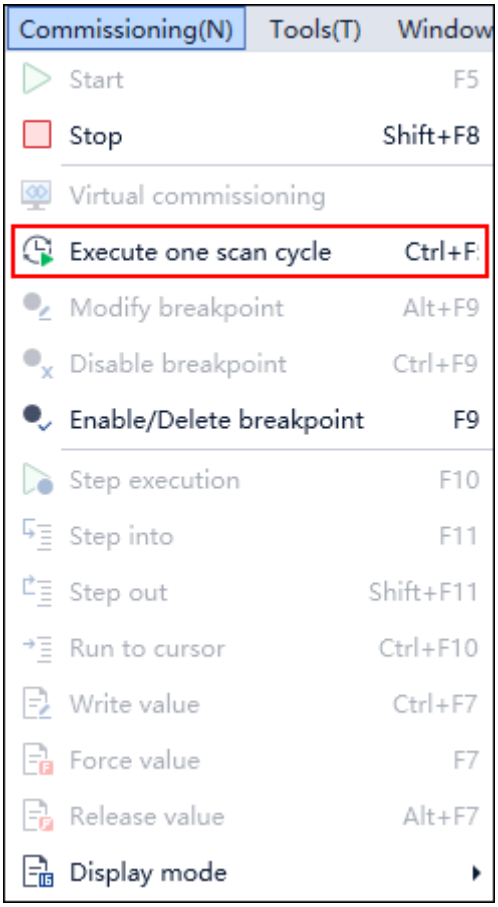


10.10.5.6 Executing One Scan Cycle

The program runs for one cycle. If the pointer stop position is not at the start of the program, it runs from the current position to the end. If there are breakpoints between the stop pointer and the end position, the breakpoints will be triggered.

Process

- Select "Commissioning > Execute a scan cycle" in the menu bar and the program runs for one cycle.



- Use the shortcut key Ctrl+F5 to run the program for one cycle.
- Click the "Execute a scan cycle" icon in the toolbar to run the program for one cycle.



10.10.6 PID Function Block Commissioning Wizard

10.10.6.1 Overview

The PID_AT and PID_ATC commands support configuration and commissioning functions.

- Configuration: Provides parameter names, upper and lower limits, grayed-out properties, and personalized displays, making complex command configurations simpler.
- Commissioning: For online command commissioning, this interface displays commonly used parameters for commissioning. Users can operate these parameters and see the commissioning results intuitively.

10.10.6.2 Adding a PID Function Block

Prerequisites

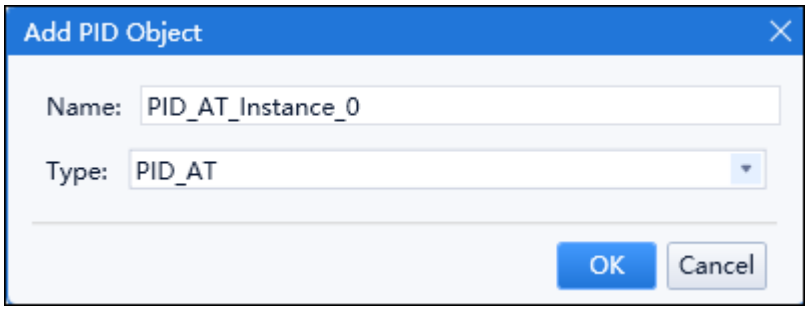
- A project has been created.
- The device has been added.

Procedure

- Adding through the project tree
  1. Select "Control object > PID" in the left project tree.

2. Right-click the "PID" node, select "Add PID object", and a dialog box will pop up.

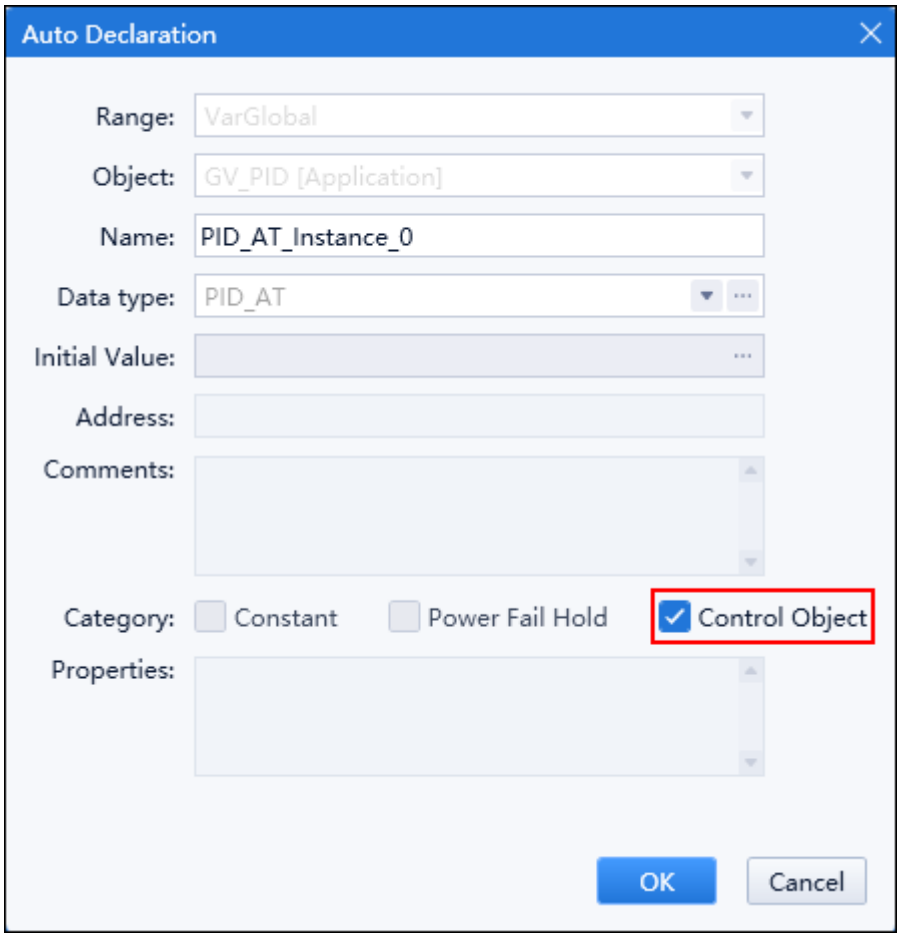
3. Enter the name and type of the PID object, and click "OK" to complete the addition.



- Automatic declaration in program editing area

1. Create a POU (only PRG and FB are supported) under "Programming" in the left project tree.

2. Open a POU (PRG or FB) under "Programming". Double-click a PID object from the toolbox or drag it to the program editing area, and the "Auto declaration" dialog box for the PID object will pop up. Check "Control object" and click "OK" to add it under the "Control object > PID" node without creating a variable in the POU.

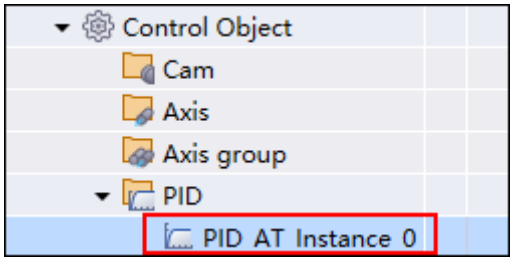


NOTE

When adding a PID object through automatic declaration in the program editing area, if "Control object" is not checked, it will be added as a regular variable, which will be added to the corresponding POU or global variable table.

Result

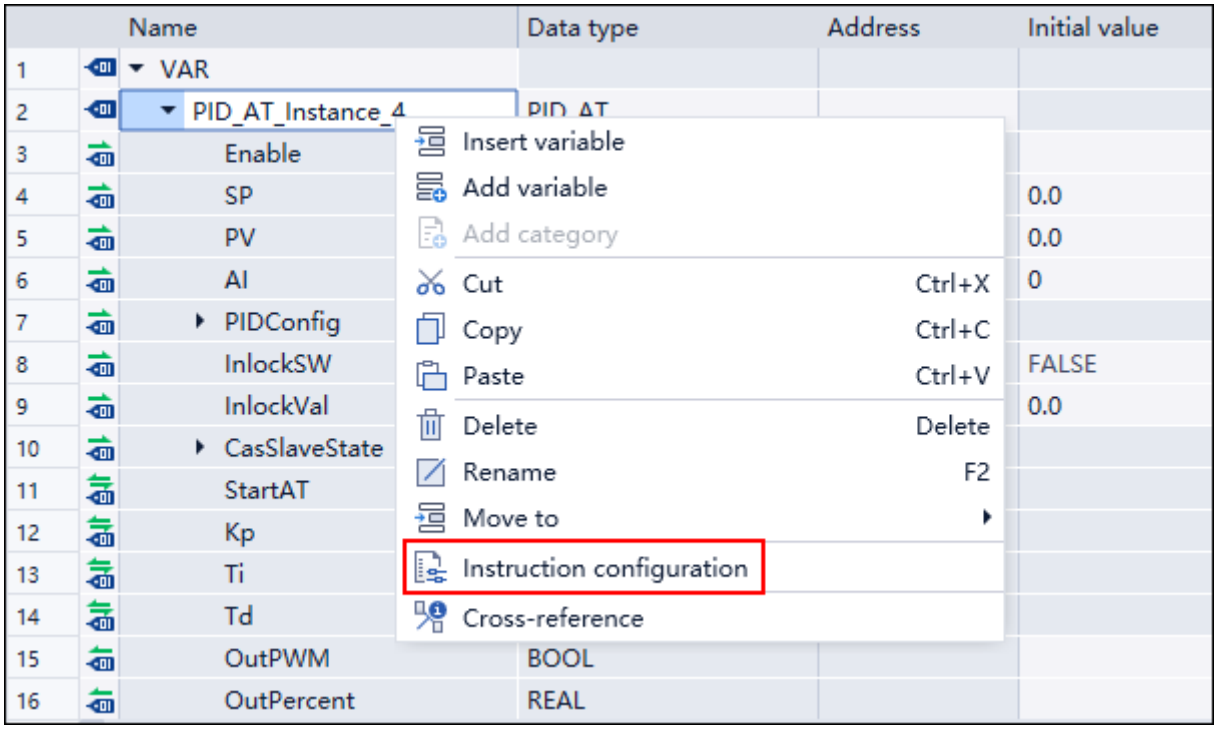
The added PID object can be viewed under the PID node.



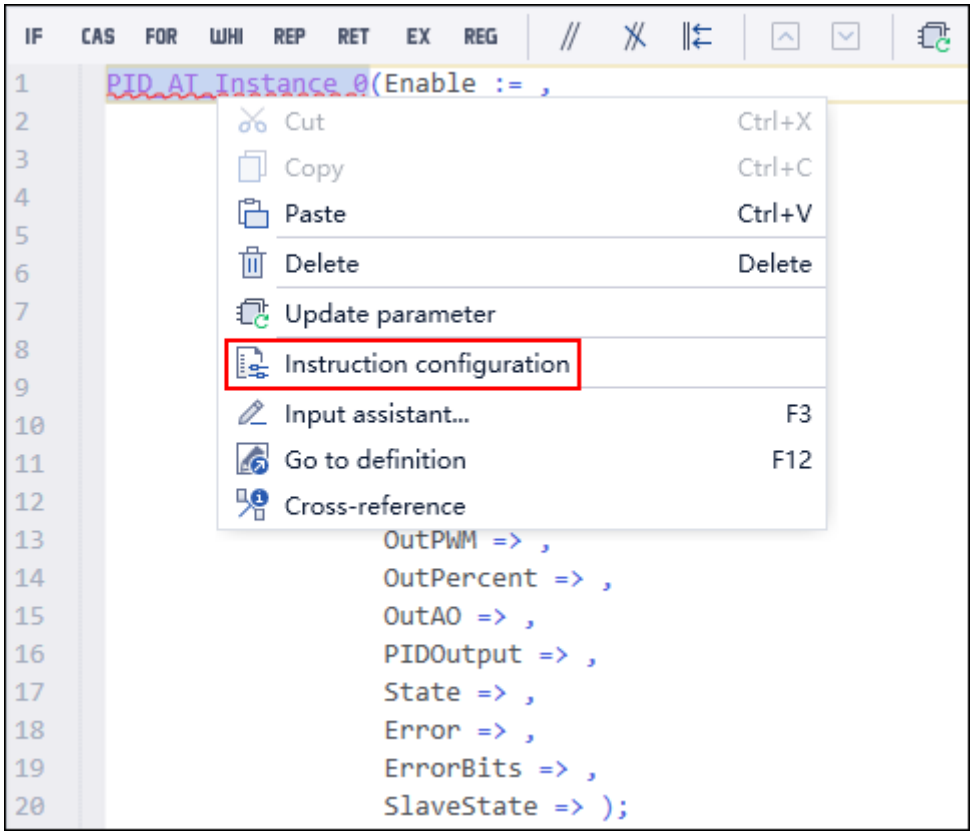
10.10.6.3 Configuration and Commissioning

Function entry

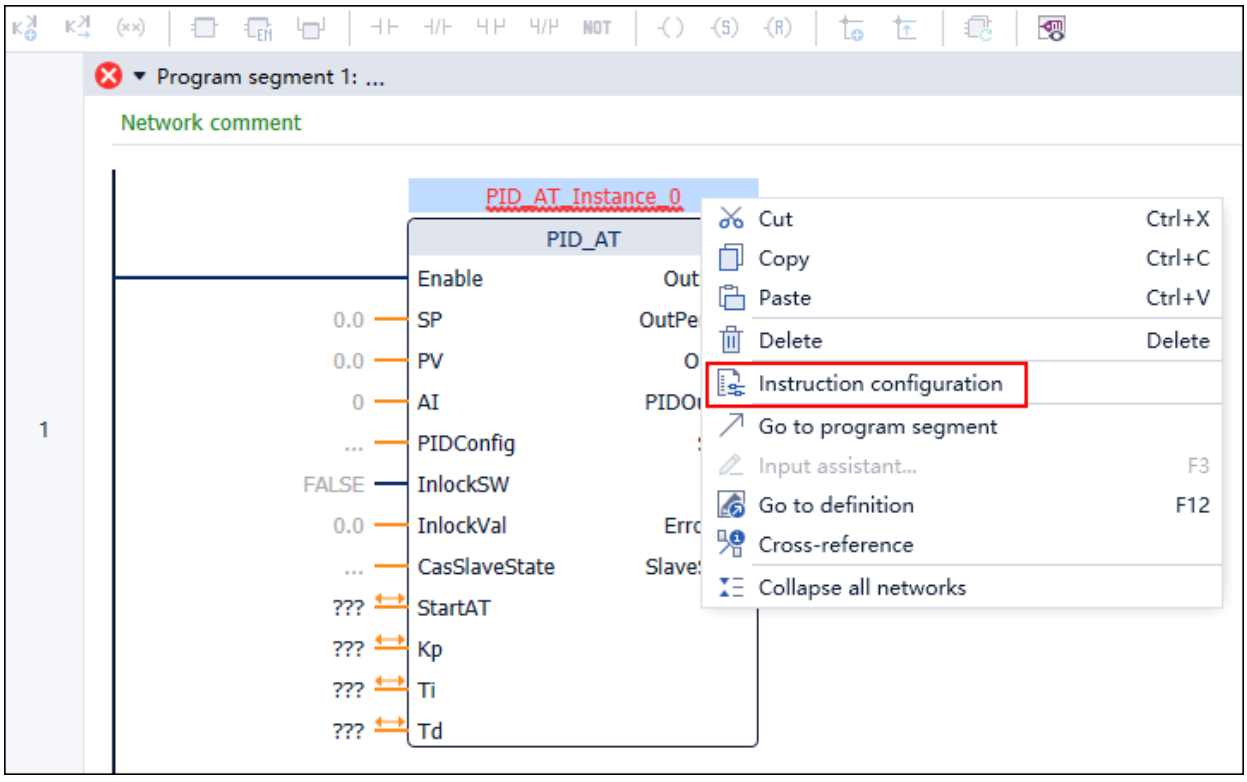
- Double-click the PID object node added under the PID node to open the configuration and commissioning interface.
- After creating a PID object instance, right-click the instance and select "Instruction configuration" to open the configuration and commissioning interface.
  - Variable table: Right-click the PID object instance row in the variable table and select "Instruction configuration".



  - ST: Right-click the PID object instance name in the ST programming area and select "Instruction configuration".



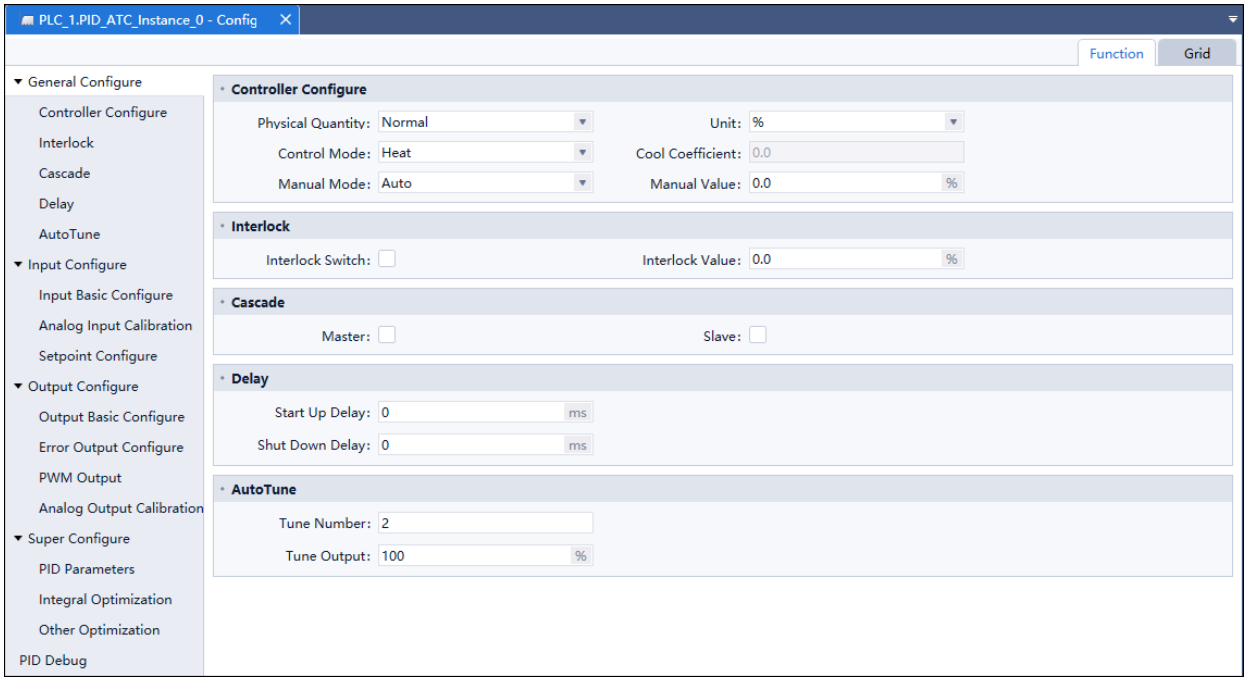
- LD: Right-click the PID object instance name in the LT programming area and select "Instruction configuration".



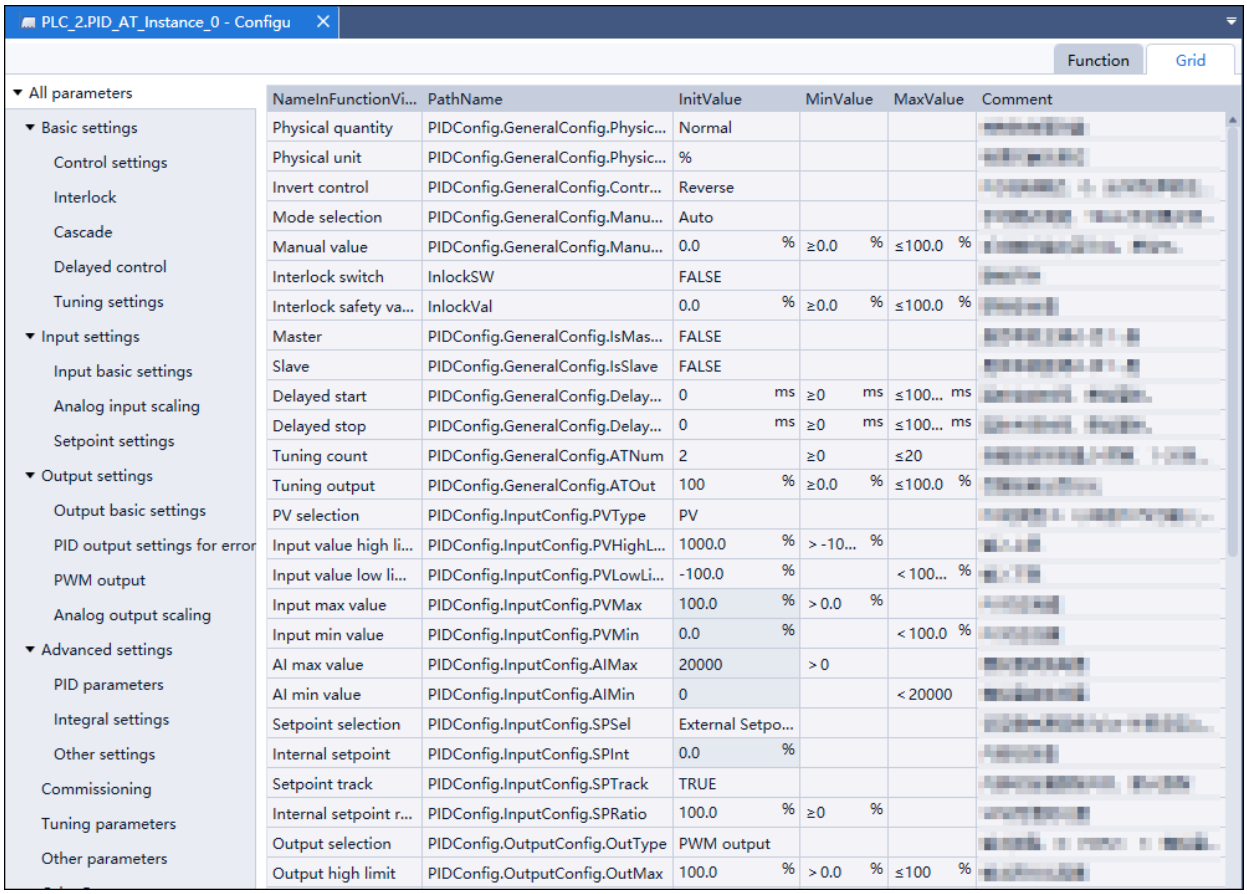
Configuration

The configuration and commissioning interface can be displayed in function view or parameter view. You can switch the display mode using the button in the upper right corner.

- Function view
  - Click each parameter category on the left, and the corresponding configuration parameters will be displayed on the right.



- Parameter view
  - Click each parameter category on the left, and the corresponding configuration parameters will be displayed on the right, showing more detailed information, including parameter names in the function view, names in the FB, initial values, minimum values, maximum values, and comments.

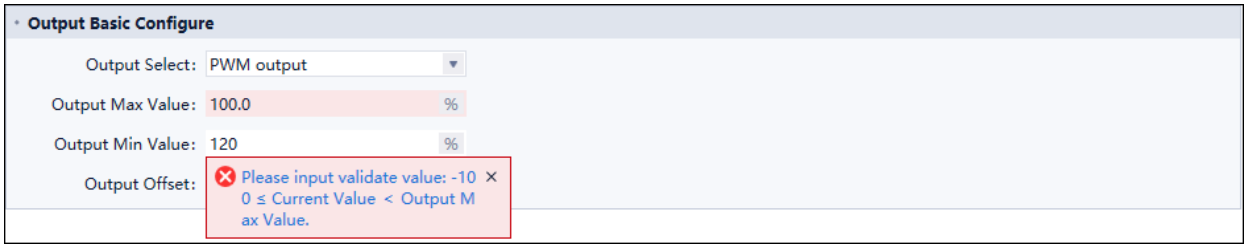


NOTE

For details and configurations of PID_AT and PID_ATC parameters, see 18.3.1 PID Control.

Offline editing

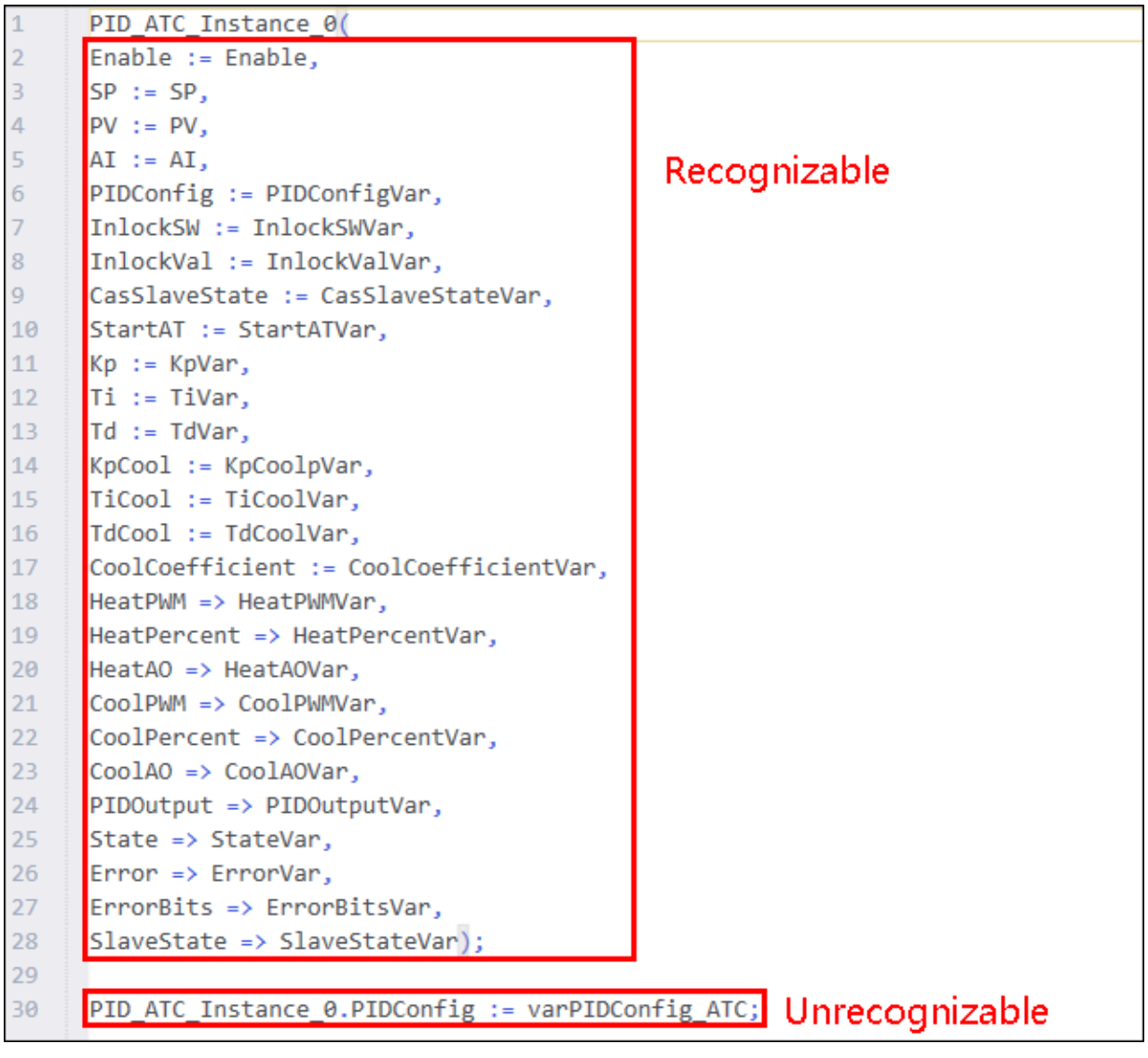
- Parameters can be edited in both function view and parameter view.
- Parameter editing supports undo and redo.
- If the input parameter exceeds the limit, a fault message will be displayed (which is only shown in the configuration interface and will not cause a compilation fault).



- If a parameter is bound to another variable in the programming area, editing the parameter value will change the bound variable.

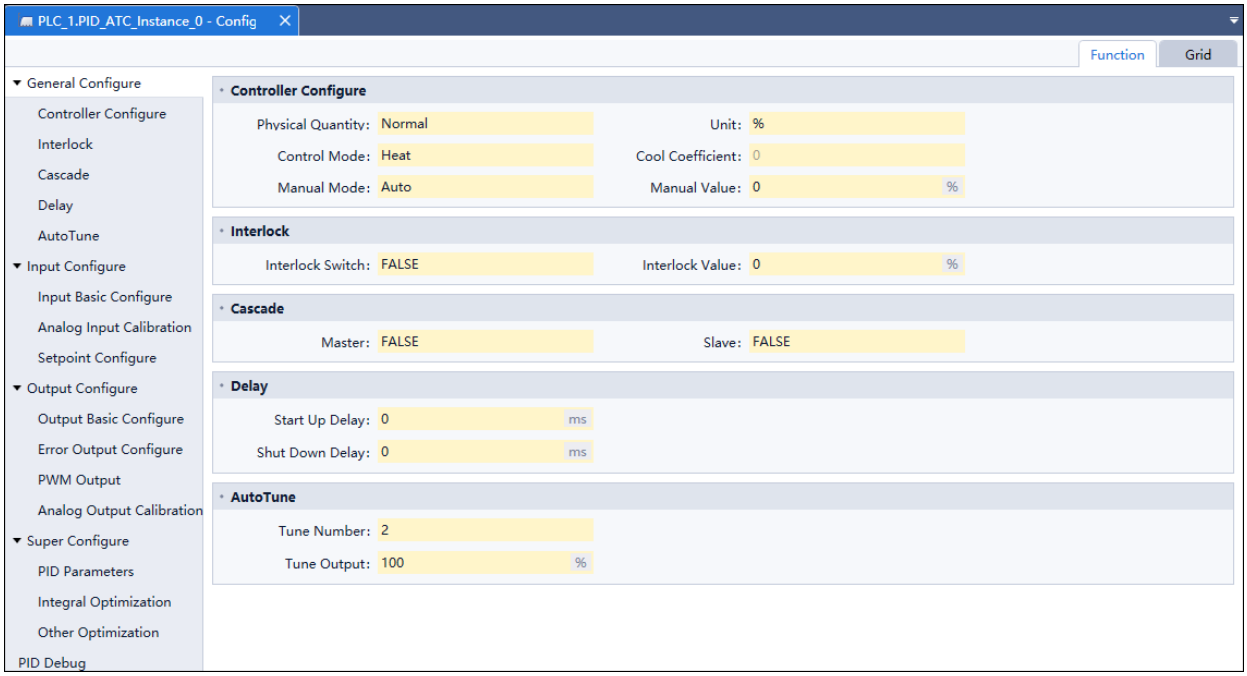
CAUTION

Variables bound to internal parameter pins of function blocks are supported for recognition, while those bound to external pins are not supported.

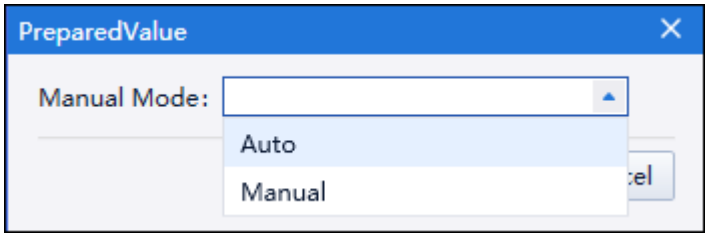


Online monitoring

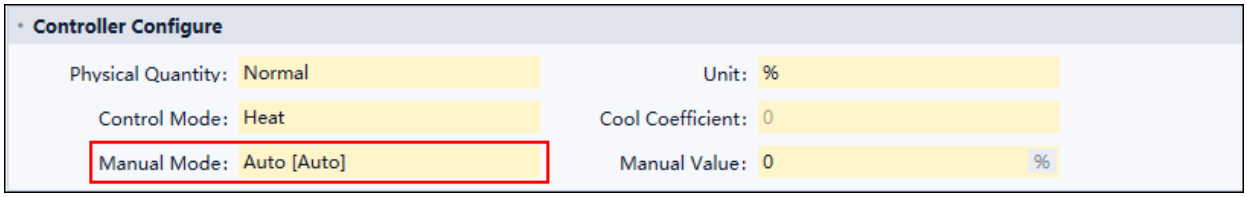
- Function view



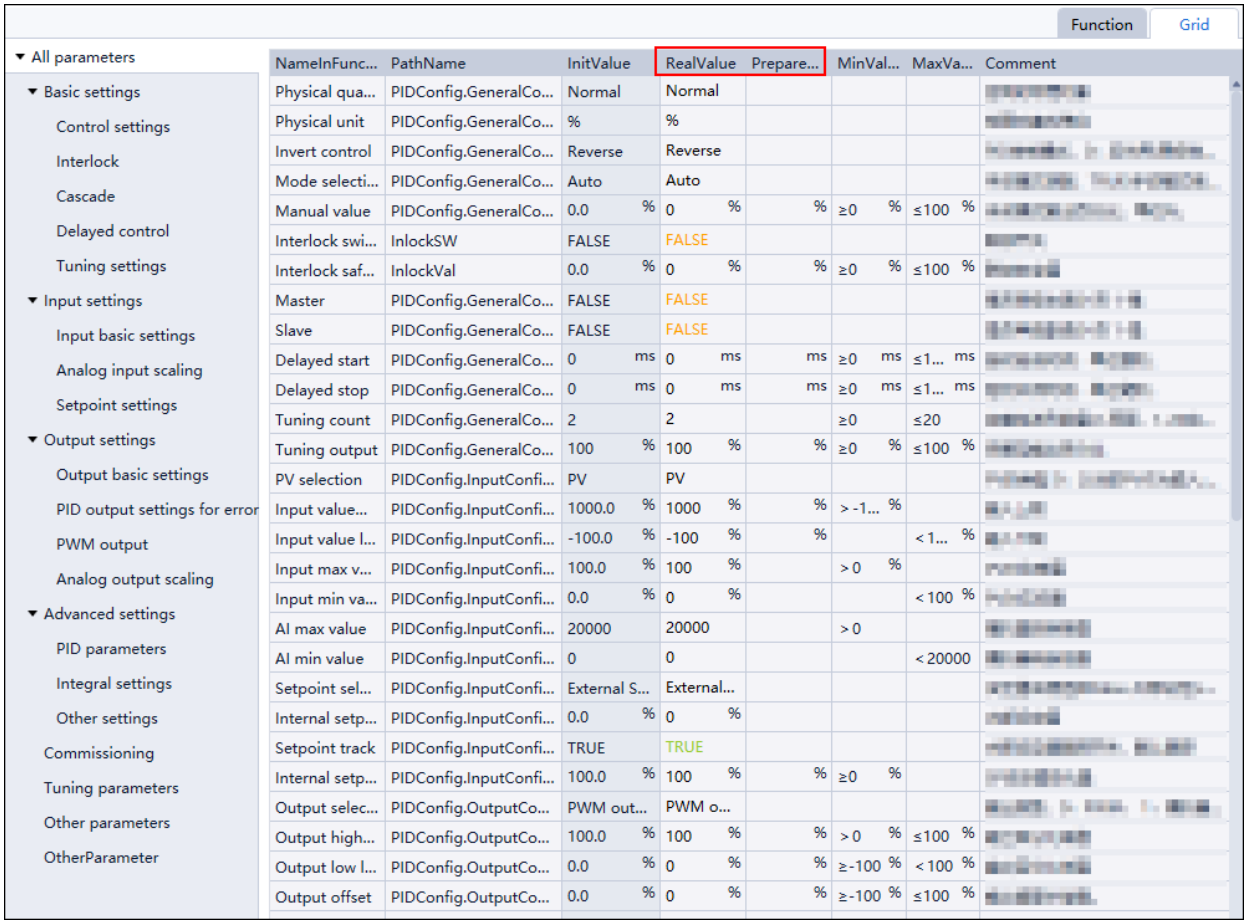
  - The input box displays real-time monitored values (decimal).
  - Double-click an ungrayed input box and the prepared value window will pop up.



  - Enter the prepared value, click "OK," and the input box will simultaneously display the monitored value and the prepared value.

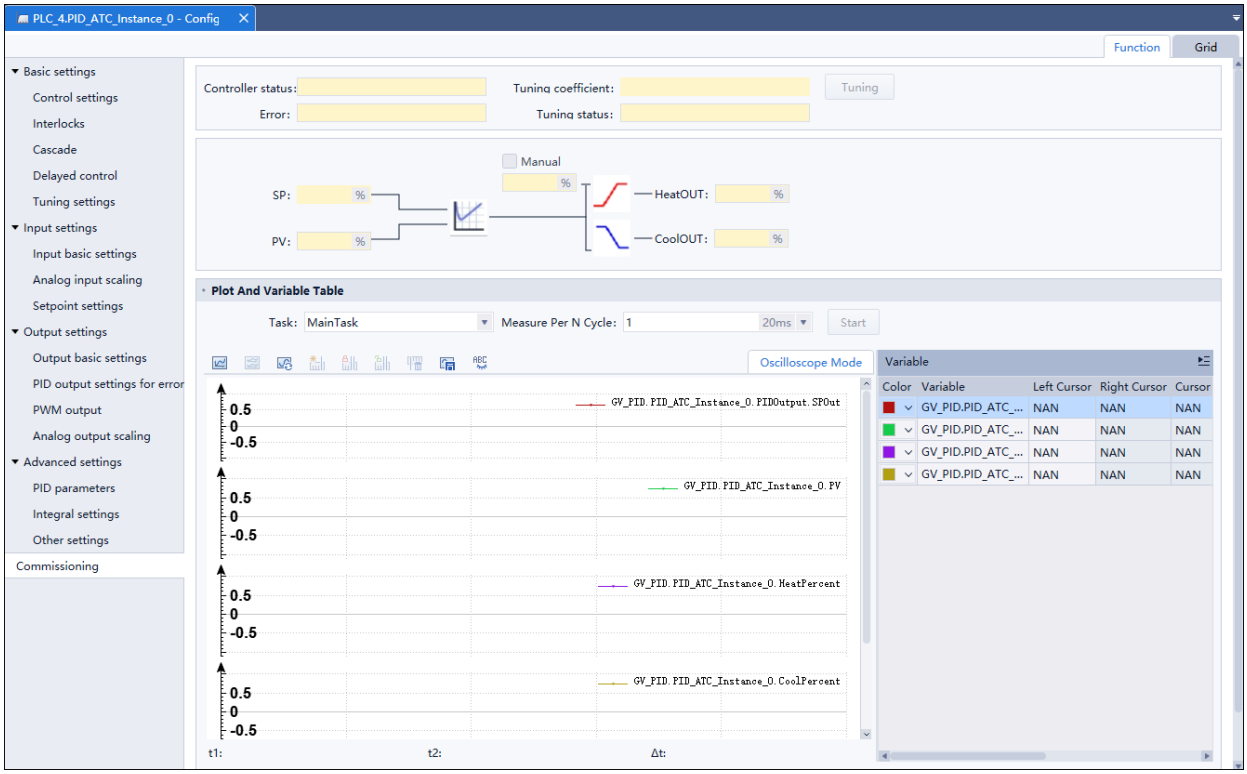


- Parameter view
  - In online mode, there are "Monitored value" and "Prepared value" columns in the parameter view, displaying real-time monitored values (in decimal) and input prepared values.



Commissioning

- The commissioning interface displays commonly used parameters during commissioning. On this interface, users can operate these parameters and see the commissioning results intuitively.
- The commissioning parameters for each command vary, so are the corresponding commissioning interfaces.
- The following figure shows the "PID_ATC" command interface. On this interface, you can operate the parameters related to the "PID_ATC" command, start the oscilloscope with one click, and observe the commissioning results.



Parameter	Description
PLC state	Corresponding to variable State; for display only.
Error bits	Corresponding to variable ErrorBits; for display only. If there is a parameter fault, an icon will appear after the monitored value box. Hovering the mouse over it will show detailed fault code and descriptions.
Auto-tuning coefficient	Corresponding to editable variable PIDConfig.GeneralConfig.ATCoefficient; supports value writing and value forcing in online status.
Auto-tuning state	Corresponding to variable PIDOutput.ATState; for display only.
StartAT	Corresponding to variable StartAT; click to write the prepared value. Call the "Write value" command to trigger tuning (set StartAT=TRUE), and the StartAT button will be grayed out. After successful tuning, it will be rewritten as FALSE, and the StartAT button can be triggered again. To stop tuning during the process, set "StartAT" to "Tuning complete" in the "Other parameters" tab in the parameter view of the command configuration interface, and then call the "Write value" command.
SP	Current setpoint; for display only.
PV	Current actual PV value; for display only.
OUT	Corresponding to variable OutPercent; for display only.
Manual	Corresponding to variable PIDConfig.GeneralConfig.ManualEnable; check to write the prepared value. Call the "Write value" command to write the value TRUE, triggering manual mode. In manual mode, double-click the manual value text box to write the prepared value, and the operation variable is PIDConfig.GeneralConfig.ManualValue. ManualValue. In automatic mode, the prepared value cannot be written to the manual value box.
Task	The task can be any user-created task or EtherCAT task.

Parameter	Description
Measure per N cycles	The sampling period (task cycle × N)
Start/Stop	Click "Start" to begin sampling with the oscilloscope. Each time you click "Start", the previous oscilloscope data will be cleared, and a new session will begin. When the oscilloscope is running, switching to offline mode or closing the commissioning interface will automatically stop the oscilloscope first. Click "Stop" to halt oscilloscope sampling.

10.11 PLC Online Operations

10.11.1 Hot Reset

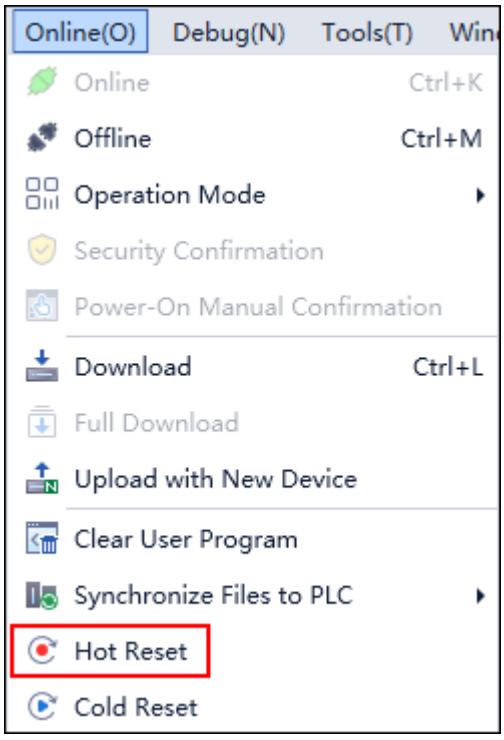
After a hot reset, all variables of the currently active application, except for power failure retention variables, will be reinitialized. If a variable has an initial value, its value will revert to the initial value after a hot reset. Other variables will be set to their standard initial values (such as 0).

Prerequisites

- The program has been downloaded to the PLC device.
- The current project is in online mode.

Procedure

1. Select "Online > Hot reset" in the menu bar.

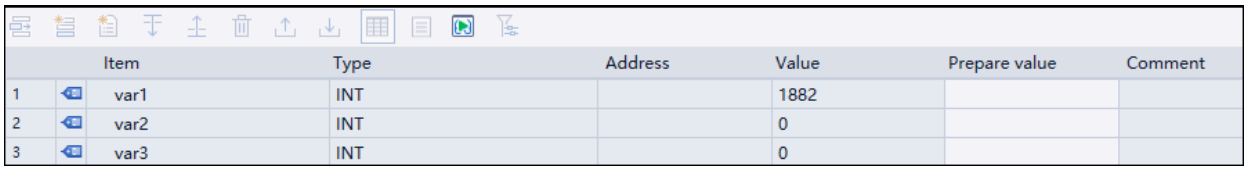


2. In the pop-up window, click "OK" to perform the online hot reset.



Operation result

After a hot reset, all variables of the currently active application, except for power failure retention variables, will be reinitialized.



		Item	Type	Address	Value	Prepare value	Comment
1		var1	INT		1882		
2		var2	INT		0		
3		var3	INT		0		

10.11.2 Cold Reset

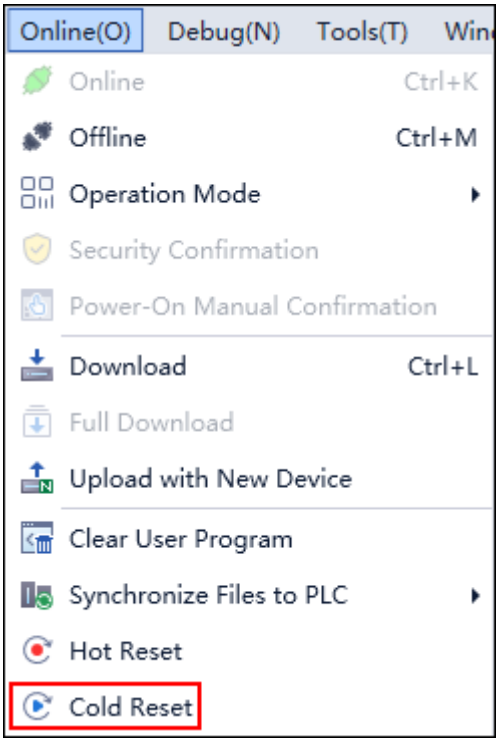
After a cold reset, all variables of the currently active application, except for power failure retention variables, will be reinitialized. If a variable has an initial value, its value will revert to the initial value after a cold reset. Other variables will be set to their standard initial values (such as 0).

Prerequisites

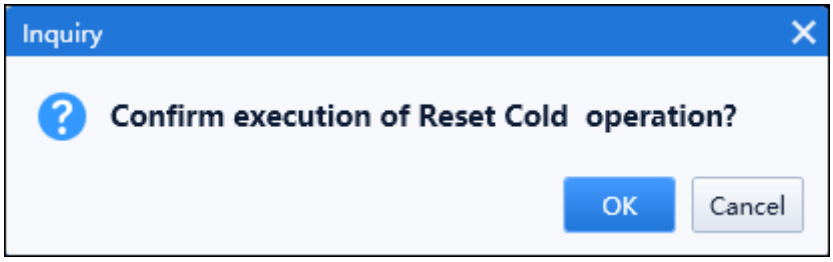
- The program has been downloaded to the PLC device.
- The current project is in online mode.

Procedure

1. Select "Online > Cold reset" in the menu bar.

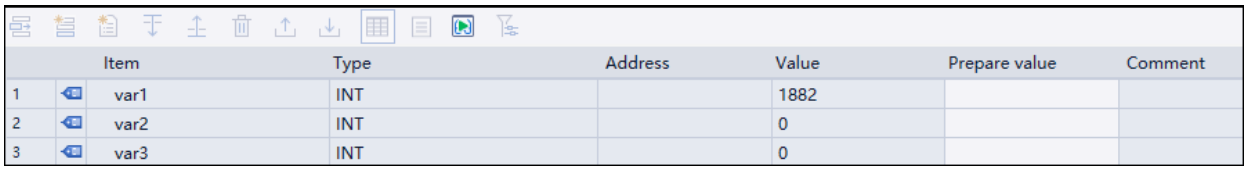


2. In the pop-up window, click "OK" to perform the online cold reset.



Operation result

After a cold reset, all variables of the currently active application, except for power failure retention variables, will be reinitialized.



		Item	Type	Address	Value	Prepare value	Comment
1		var1	INT		1882		
2		var2	INT		0		
3		var3	INT		0		

10.11.3 Clearing User Program

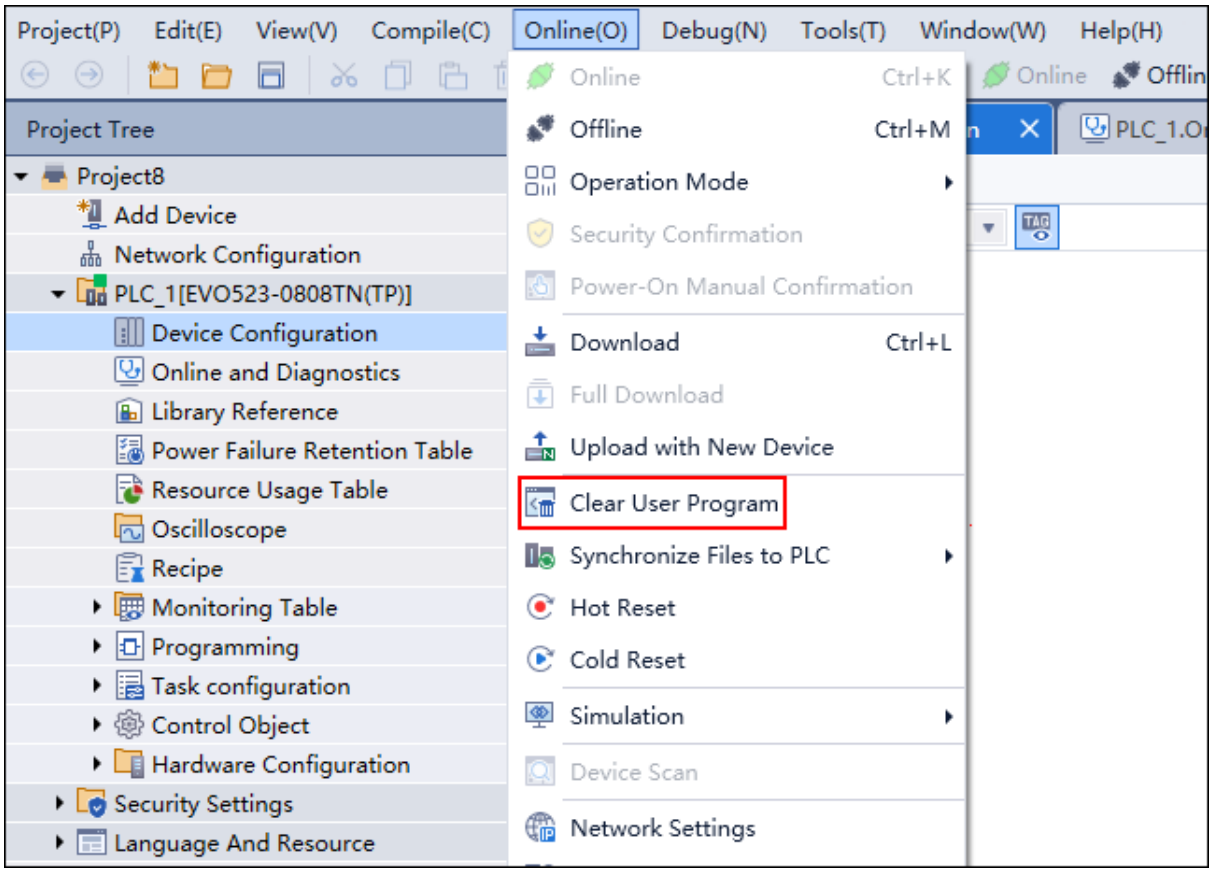
After clearing user programs, the programs in the PLC will be cleared, and the device will be switched to the offline status.

Prerequisites

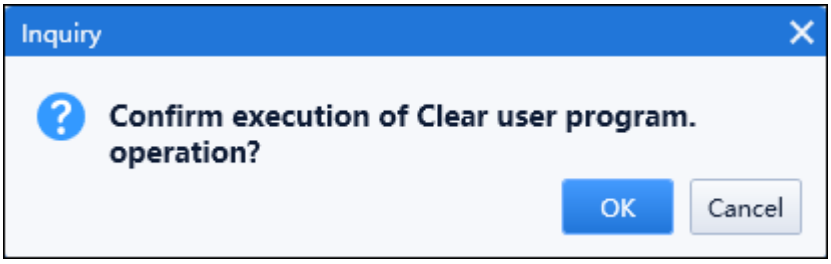
The device is in an online status.

Procedure

1. Select "Online > Clear user program" in the menu bar.

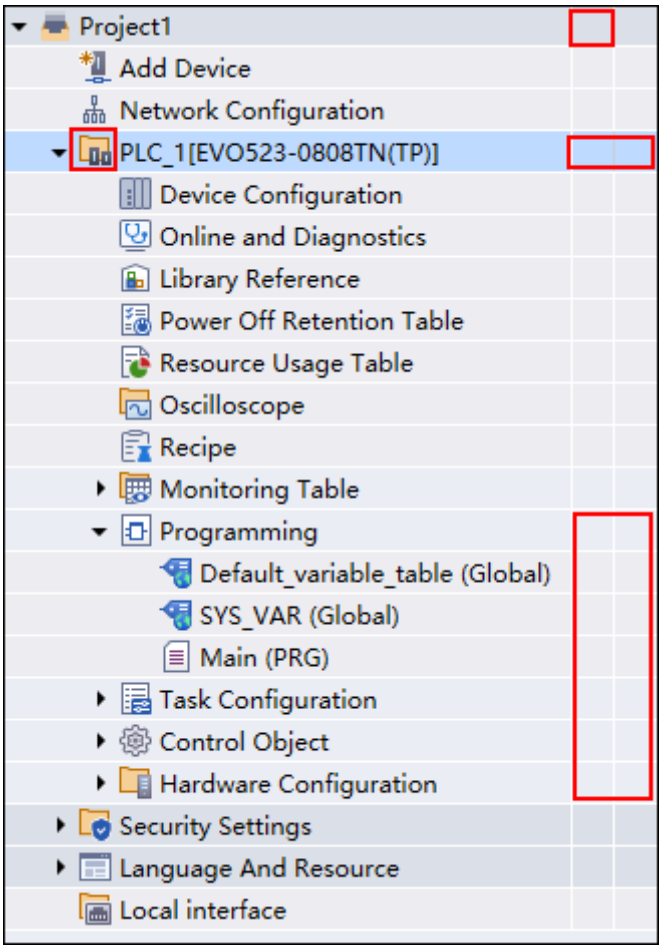


2. In the pop-up window, click "OK" to clear user programs.



Operation result

After clearing user programs, the PLC project will switch to an offline status, and the green identifier for the corresponding device in the project tree will disappear.



10.11.4 Synchronizing Files to PLC

This function is used to synchronize files to the PLC, including two source code and compilation files.

Synchronizing source code

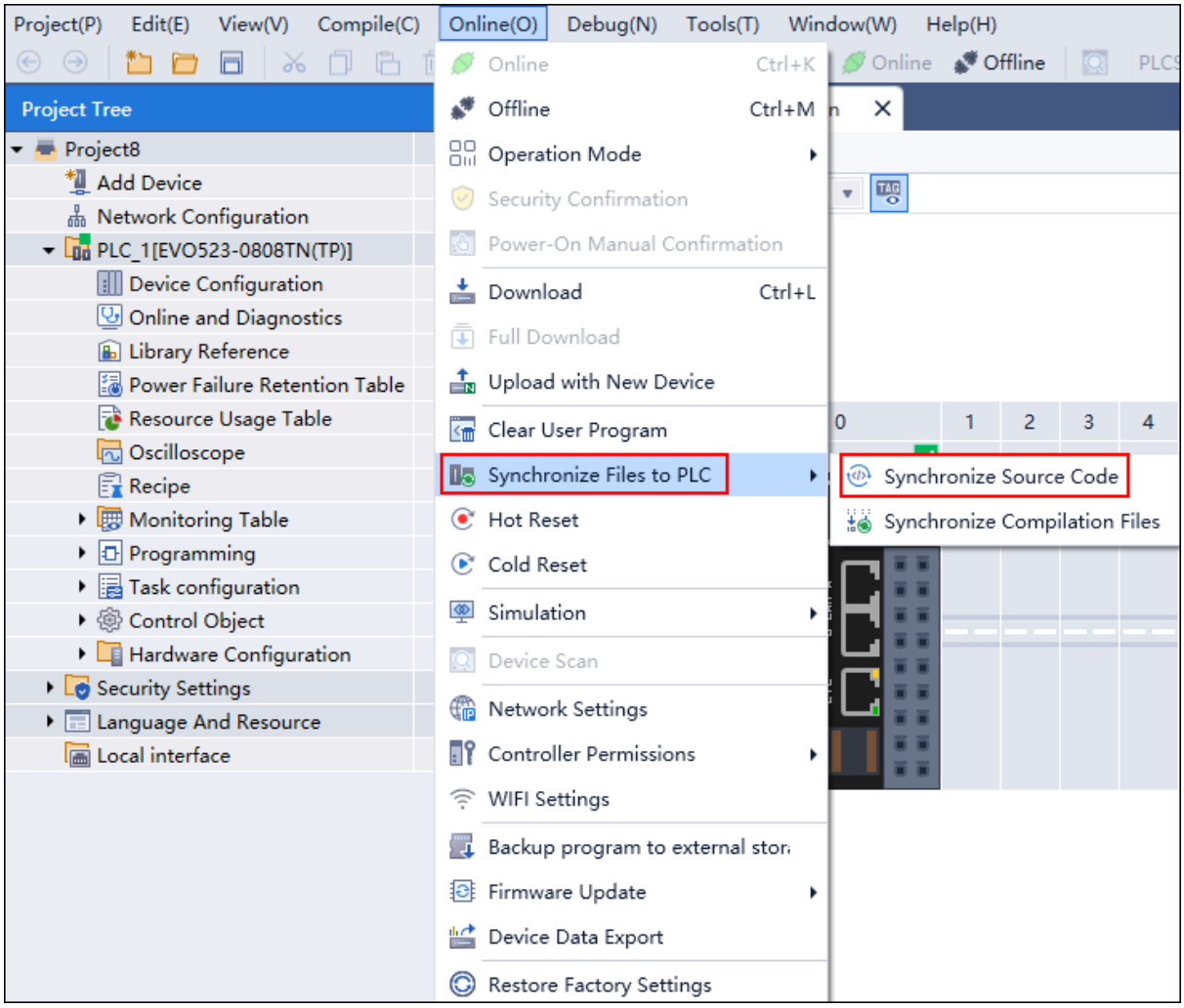
Synchronize project source code to the PLC.

Prerequisites

- The current project is in online mode.
- The user program has not changed.

Procedure

1. Select "Online > Synchronize files to PLC > Synchronize source code" in the menu bar to download the project source code to the PLC.



2. The download result will be displayed in the monitoring area.



Synchronizing compilation files

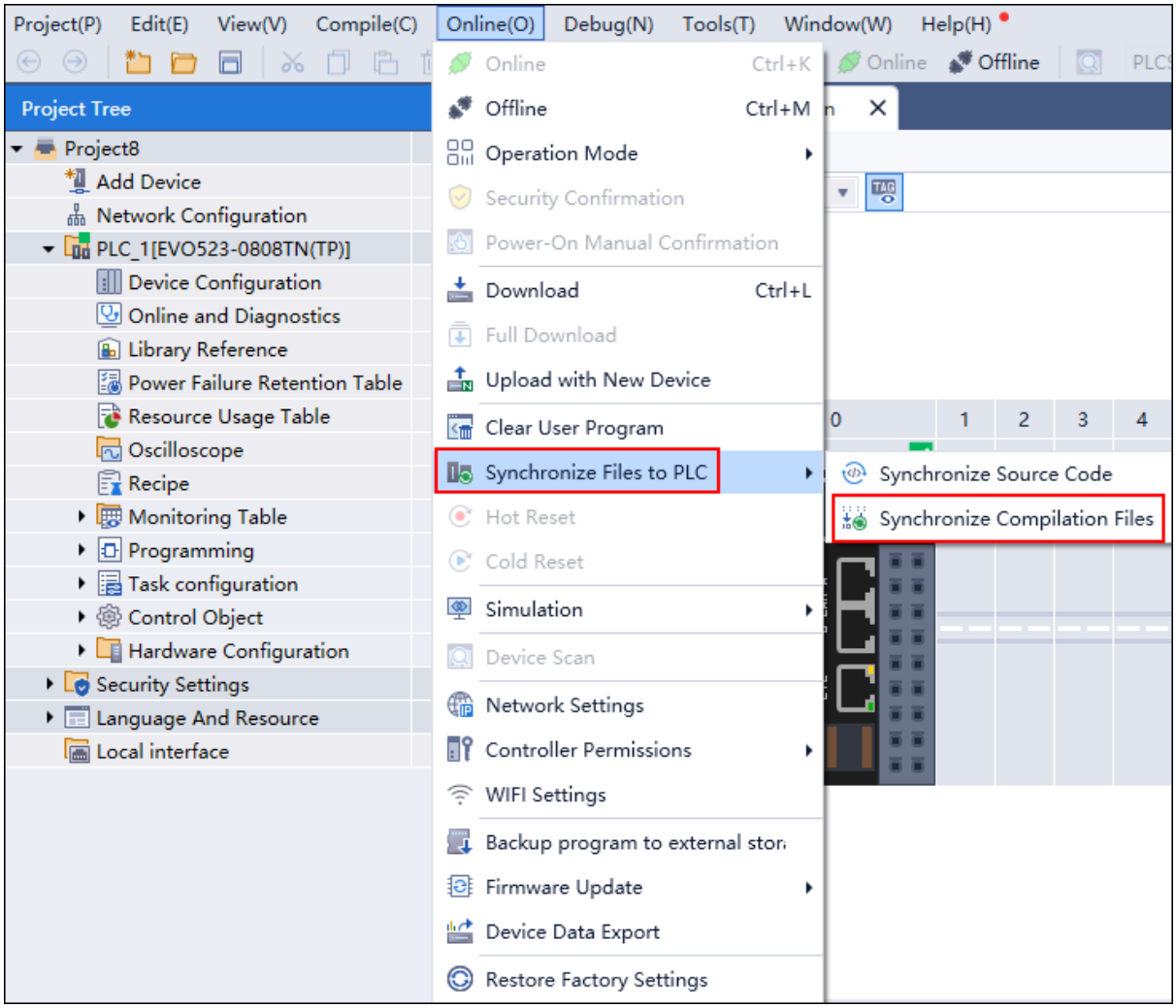
Synchronize compilation files to the PLC.

Prerequisites

- The current project is in online mode.
- The user program has not changed.

Procedure

1. Select "Online > Synchronize files to PLC > Synchronize compilation files" in the menu bar to download the compilation files to the PLC.



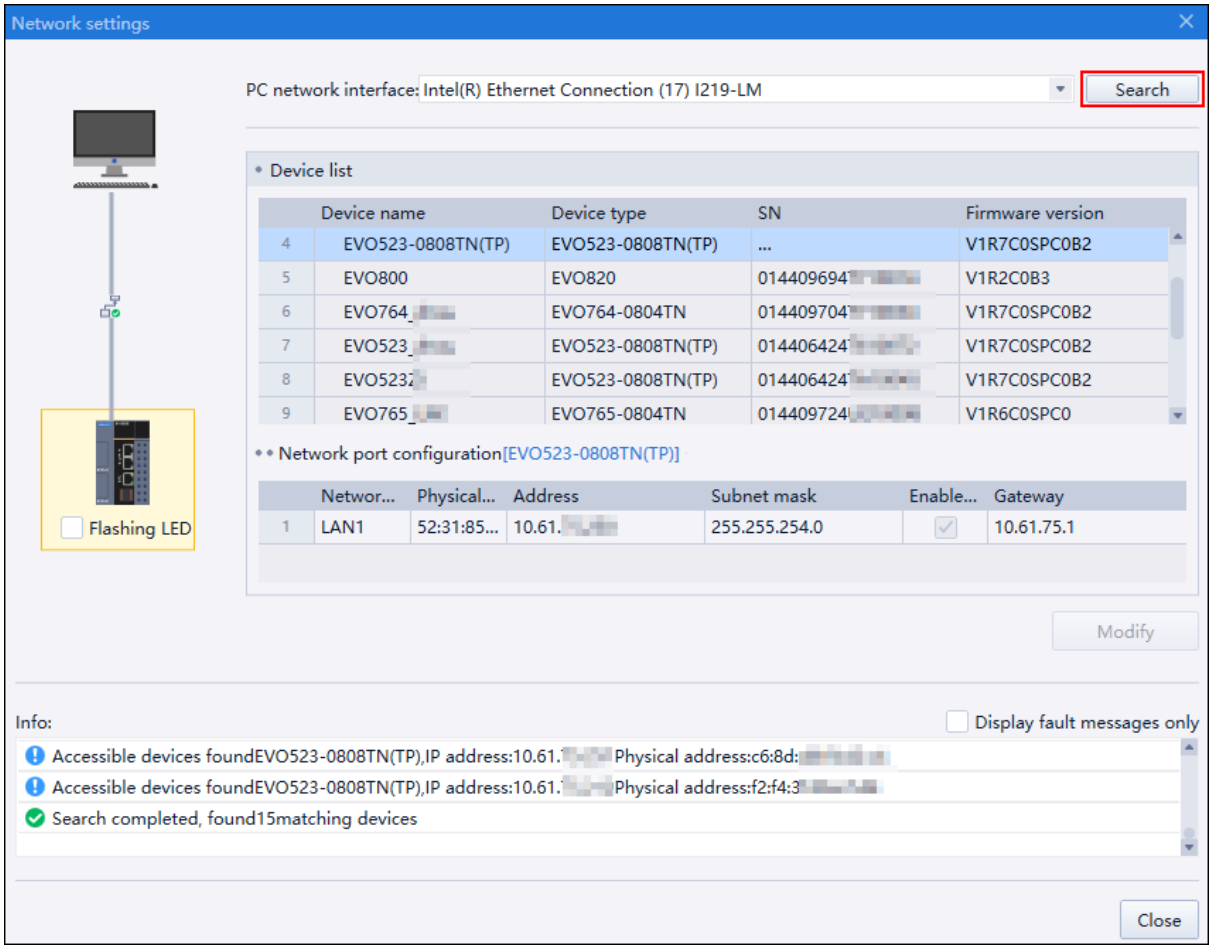
2. The download result will be displayed in the monitoring area.



10.11.5 Network Settings

1. Select "Online > Network setting" in the menu bar to open the network settings interface.
2. Click "Search for devices" to view all devices in the same subnet and across subnets.

> NOTE
>
> The EVO800 series PLC supports downloading and commissioning through the Ethernet expansion module port. When scanning devices through this port, the current system only supports scanning the PLC to which the connected Ethernet expansion module belongs; other PLCs are not supported for scanning. Regardless of whether iFA Evolution is connected to the PLC or to the expansion module: In single or redundant mode, the IP addresses of both the PLC and the expansion module can be set. In single-rack multi-PLC mode, only the IP addresses of the current group (the expansion module and its associated PLC) can be set.



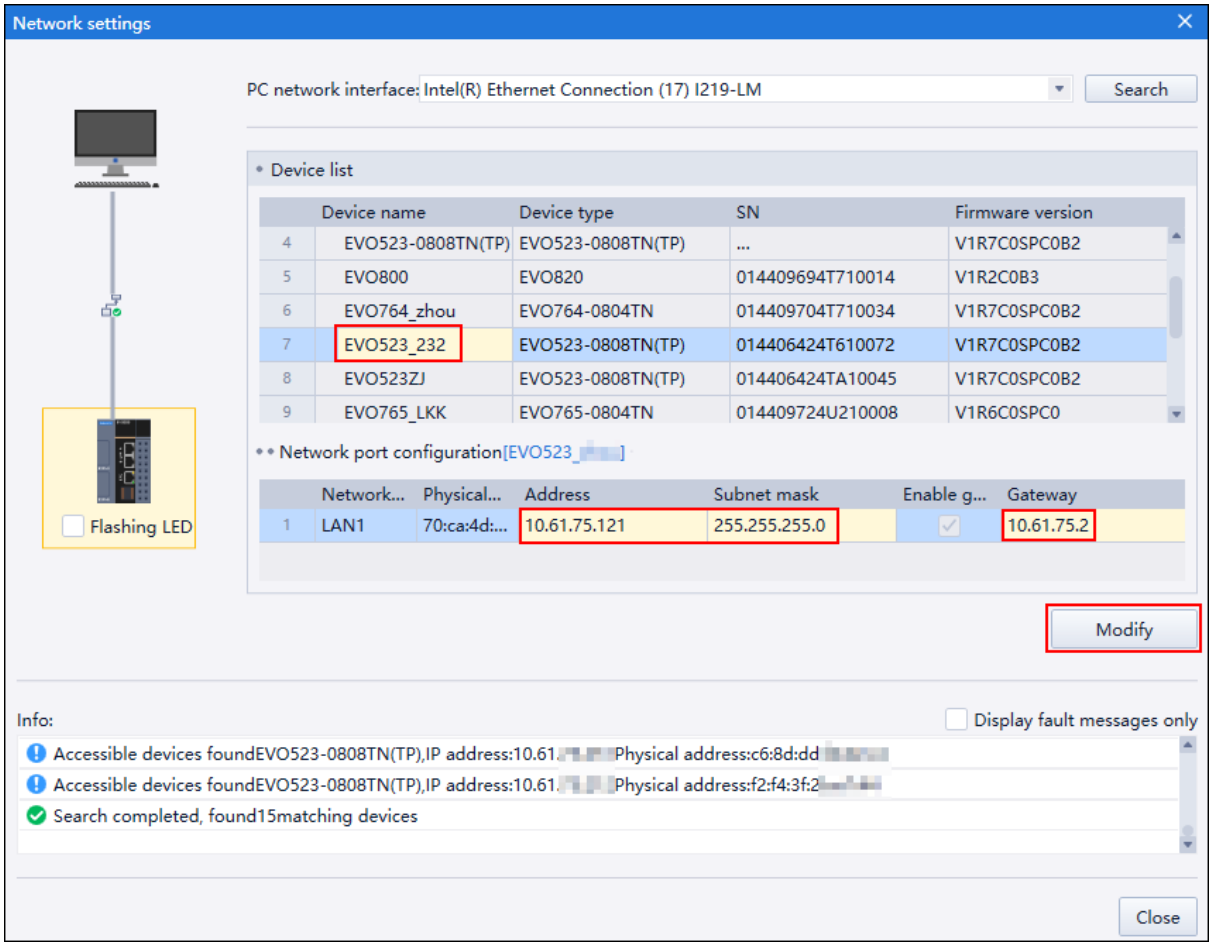
3. Confirm the device.

   Ensure the device selected online matches the target device in the physical environment to avoid connection faults. There are two methods to confirm:

   - SN number consistency check: Compare the SN number in the window with that on the physical device, ensuring they match.
   - LED indicator consistency check: When "Flashing LED" is checked in the window, the **RUN** indicator on the corresponding physical device will flash in response.

4. Edit device information.

   Double-click to edit the device name, address, subnet mask, and gateway. Then, click "Modify device info" to apply the changes.



The information bar will display the modified content and the operation result.

NOTE

- The gateway address must be in the same network segment as the device's IP address.
- For multi-port devices, only one default gateway can be enabled at a time.
- Devices and firmware versions that support enabling the gateway function are as follows:
  - EVO500/EVO700: V1.7 and above.
  - EVO400: V1.5 and above.
  - APC series: V1.7 and above.
  - EVO800: V1.2 and above.

10.11.6 PLC Permission Configuration

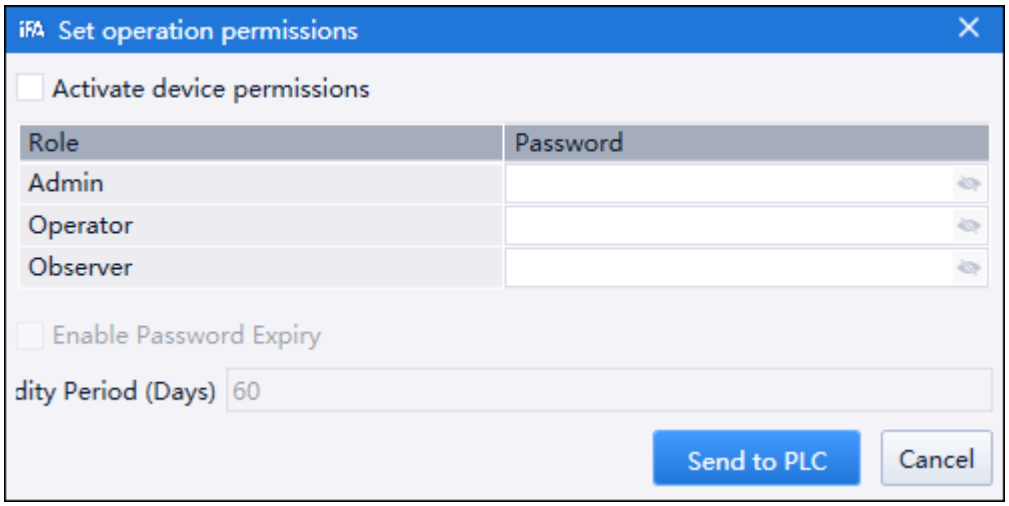
The controller permission configuration function is used to set operation permissions for PLCs. After permissions are set, users must select the appropriate role and pass authentication when connecting to the device (for downloading, going online, etc.). Once authenticated, they can perform the relevant operations.

Prerequisites

Only the administrator can configure controller permissions. Ensure the current user is an administrator.

Operation permission settings

1. In the iFA menu bar, select "Online > Controller permission > Set operation permissions".
2. On the "Device scanning" page, click "Start search", select the target PLC, and click "OK".
3. Set permissions for the PLC. The parameters are described in the table below.



Parameter	Description
Activate device permissions	When checked, device permission configuration is enabled.
Password	After "Activate device permissions" is checked, you can enter passwords for each role. The password length is 8 characters and must include at least one number, uppercase letter, and lowercase letter.
Enable password validity	When checked, you can configure the password validity period. Each time you log in to the device, the remaining days are displayed. When the password is expired, users can only log in as the administrator. After successful login, they must change the passwords for administrators, operators, and observers. No other operations can be performed until the password change is successful.
Password validity period (days)	Default is 60 days. The configurable range is 1 to 3650 days.

4. After settings, click "Send to PLC" and the permission settings take effect.

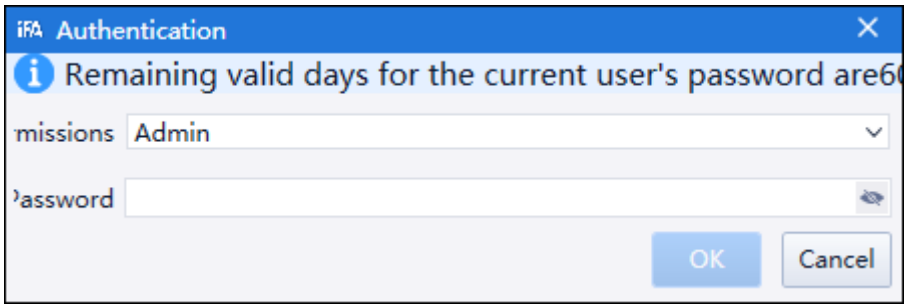
NOTE

- To cancel permission configuration, uncheck "Activate device permissions" in the "Operation permission settings" window, then click "Send to PLC".
- When the software version used by the user is lower than V1.5.2.5 and the connected device is a standard PLC (including EVO400, EVO500, EVO500C, EVO700, EVO700C, and APC series with firmware version V1.7 or later, and EVO800 with firmware version V1.2 or later), if the device has been set with password permission, the software will not be able to connect to this device. To connect, upgrade the software to V1.5.2.5 or later.

Connecting to a controller

Before connecting to a device that has permissions set (e.g., downloading, going online), authentication is required. The download operation is used as an example:

1. Click the "Download" icon on the iFA toolbar.
2. On the "Device scanning" page, click "Start search", select the target PLC, and click "OK".
3. In the pop-up "Authentication" window, select the operation permission, enter the password, and click "OK".

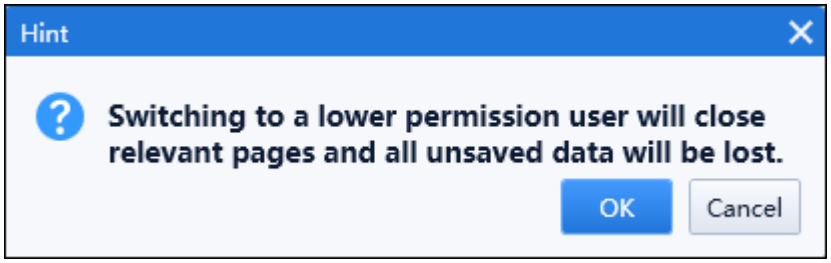


> **NOTE**
>
> During commissioning, to avoid entering passwords when connecting to the device, you can set the PLC device's online retention duration. Within this duration, connecting to the device does not require re-entering the password. For details, see 6.1 General Settings.

Switching roles

For a device with permissions set, use the "Switch user" function to change the current user's operation permissions.

1. In the iFA menu bar, select "Online > Controller permission > Switch user".
2. On the "Device scanning" page, click "Start search", select the target PLC, and click "OK".
3. In the prompt box that appears, click "OK".



4. In the pop-up "Authentication" window, select the operation permission, enter the password, and click "OK" to switch permissions.

Description

The table below describes the permission scope for controllers of different roles.

Permission Scope	Role		
	Administrator	Operator	Observer
Upload/Download	√	×	×
Modify administrator password	√	×	×
Modify role password	√	×	×
WIFI settings[1]	√	√	×
Main/Backup switching[1]	√	√	×
DLR ring network	√	√	×
Time data settings	√	×	×
Online comparison	√	×	×
CPU reset	√	×	×
CPU start/stop	√	×	×
Firmware upgrade	√	×	×
SD card upgrade	×	×	×
Clearing user programs	√	×	×
Trace monitoring	√	√	√
Breakpoint commissioning	√	×	×
Online program modification	√	×	×
Online program monitoring	√	√	√
Variable Monitoring	√	√	√
Variable modification	√	√	×
Variable forcing	√	√	×
Exporting device data	√	×	×
Log clearing	√	×	×
Program and configuration backup[2]	√	×	×

√ indicates that the permission is supported, × indicates that the permission is not supported.

NOTE

All PLCs support the permissions unless otherwise specified.

[1]: This permission is only supported by EVO800 series PLCs.

[2]: This permission is only supported by EVO800 series and EVO1000 series PLCs.

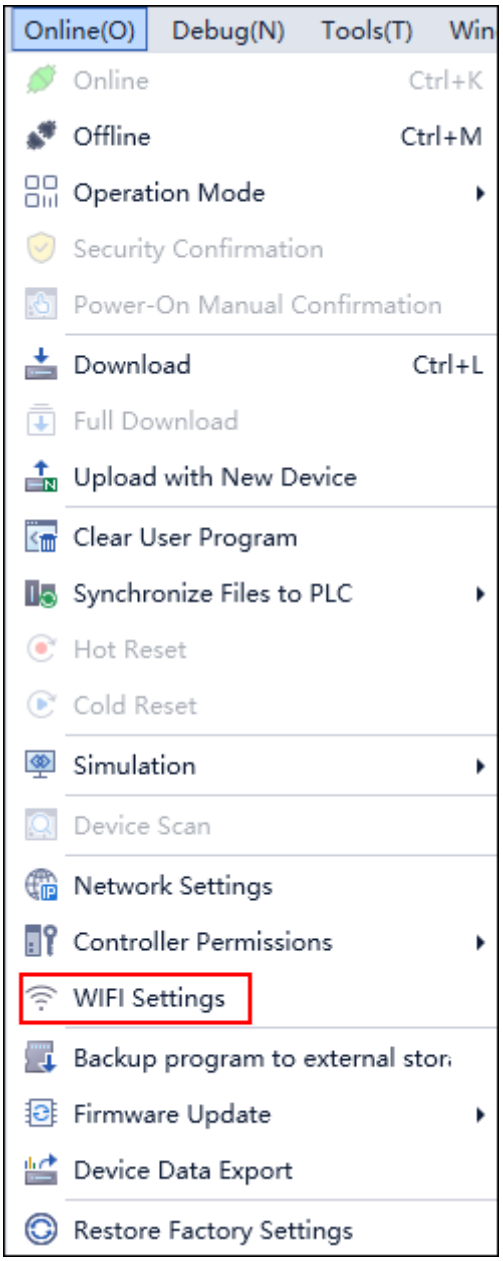
10.11.7 WIFI Settings (EVO820)

iFA supports setting the WIFI parameters of a PLC and activating or deactivating it without relying on a project. Currently, only the EVO820 series PLC supports this feature.

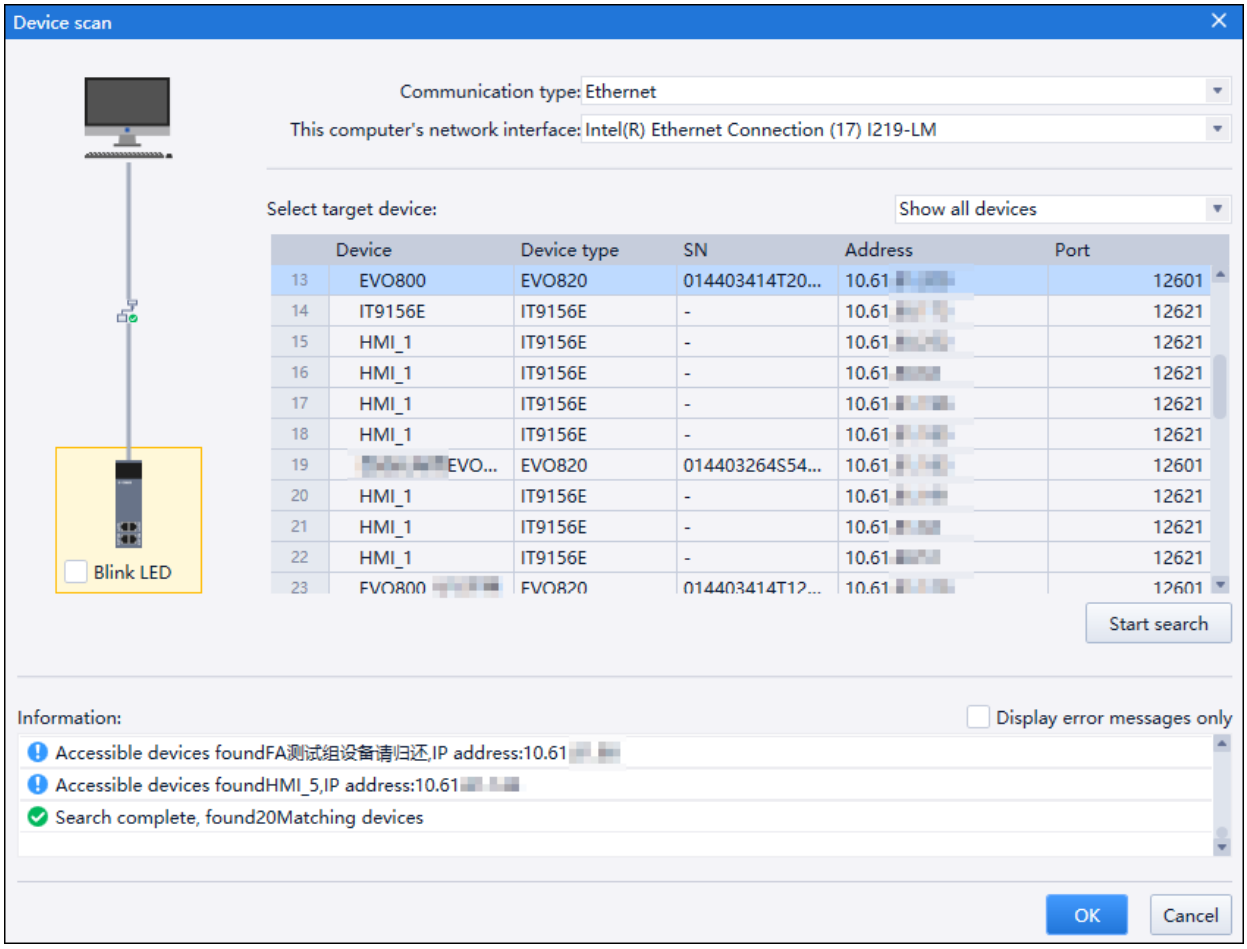
There are two ways to set this up.

- Select "Web diagnostics and monitoring > WIFI settings" in iFA Web. For details, see 10.11.12 Web Diagnostics and Monitoring.
- Select "Online > WIFI settings" in the menu bar.

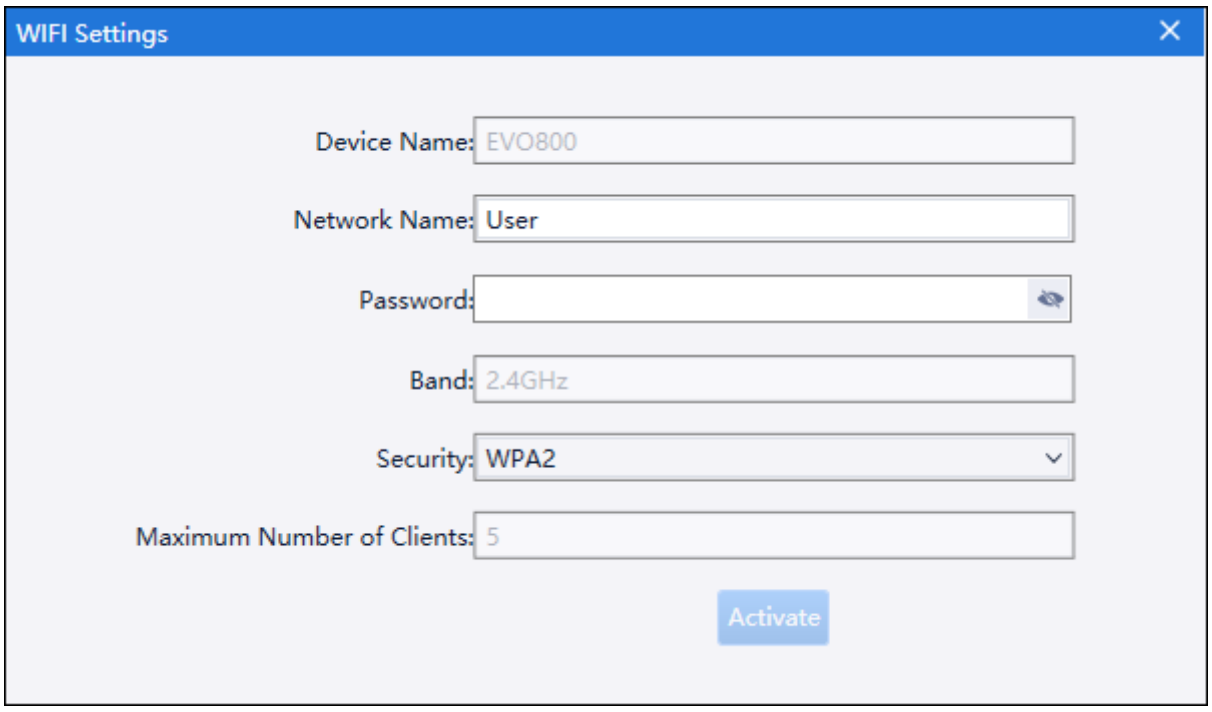
Select "Online > WIFI settings" in the menu bar, as shown in the figure below.



In the pop-up device scan interface, select the EVO800 series PLC.

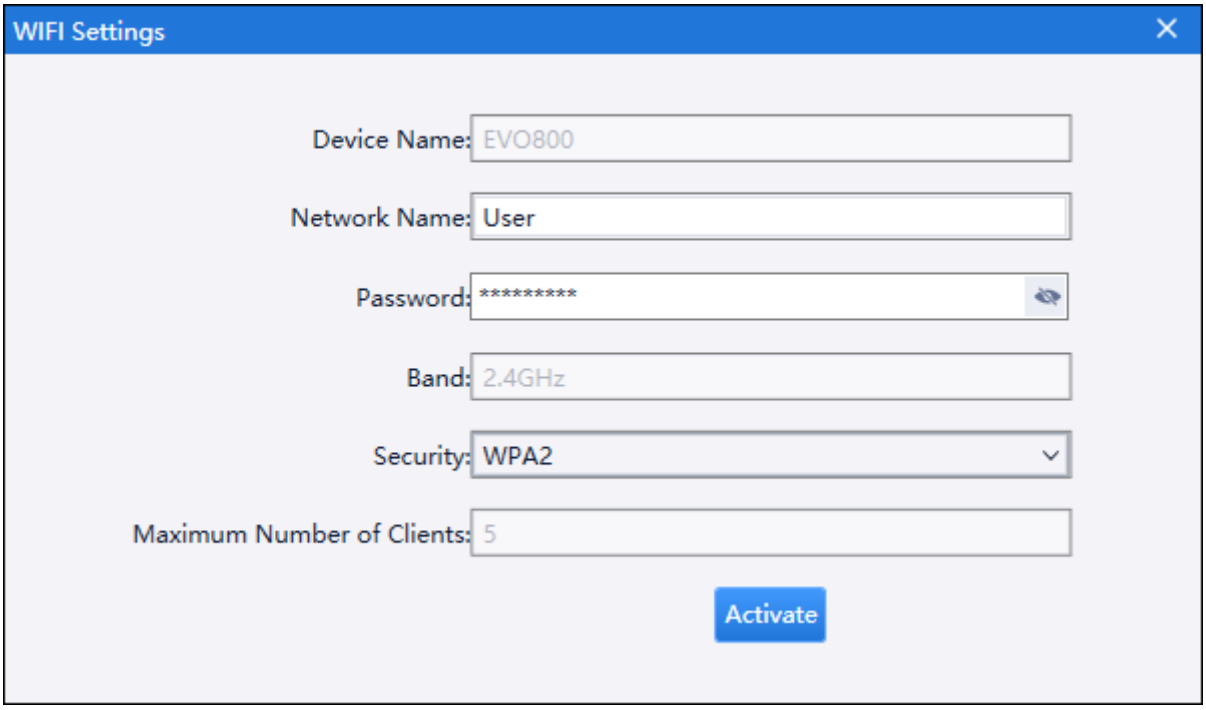


Click "OK" to enter the WIFI settings interface.

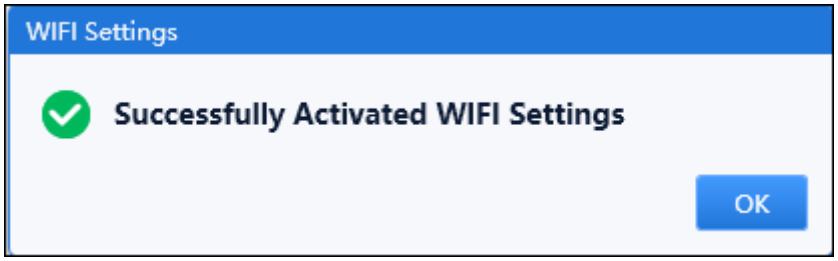


- Device name: The name of the selected device
- Network name: The WIFI network name, which supports customization. The default is InovancePLC.
- Password: The initial WIFI password is empty. Set it according to the prompts.
- Frequency band: The default is 2.4 GHz and cannot be modified.
- Security: Supports WPA2 (default) and WPA3.
- Maximum number of clients: The default is 5 and cannot be modified.

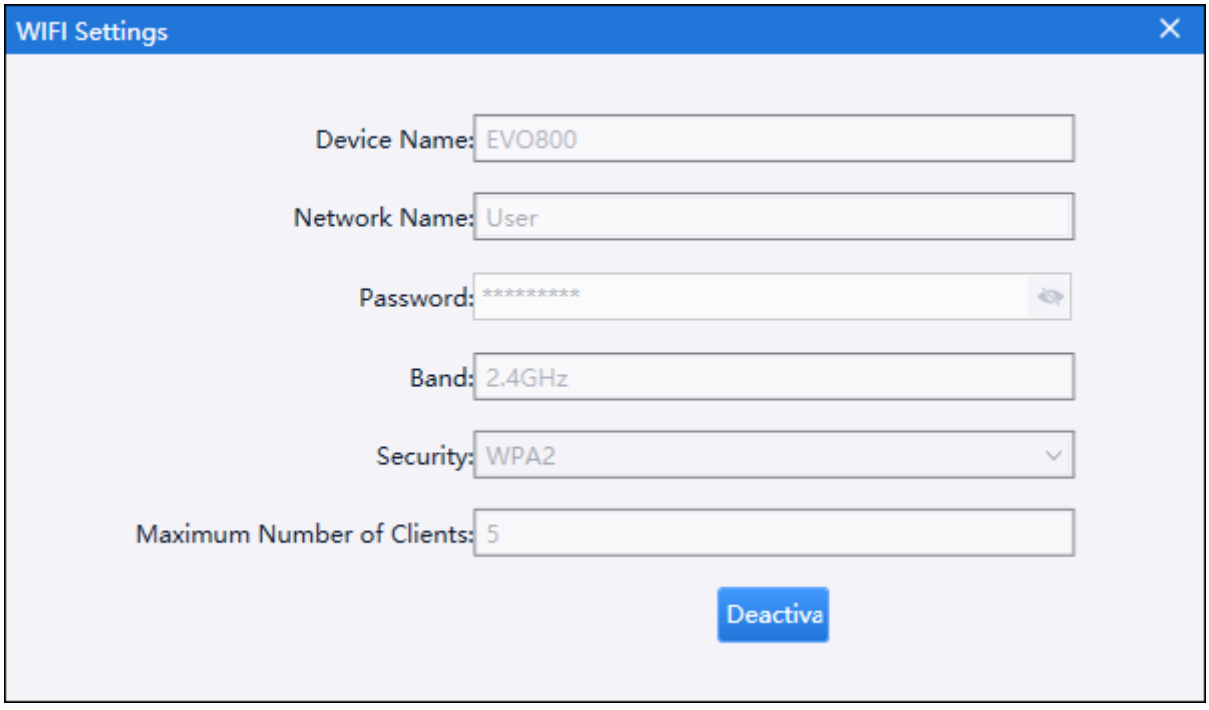
After setting the parameters in the WIFI settings interface, the activation button becomes enabled.



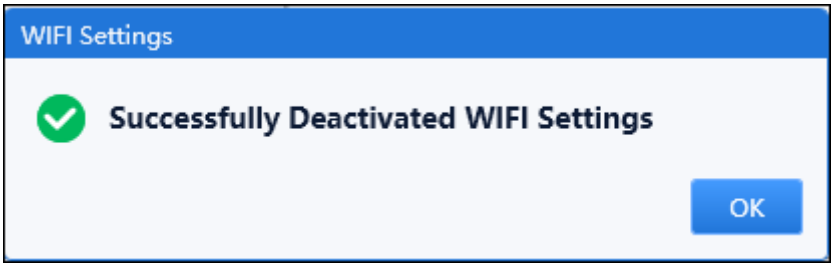
Click the activation button, and a dialog box indicating successful activation will pop up.



Select the device that has successfully activated WIFI, and a dialog box for turning off WIFI will pop up.



Click the close button, and a dialog box indicating successful WIFI closure will pop up.



WIFI settings apply to standalone large-scale PLCs, and main/backup devices of a redundant system. If the PLC is configured with permissions, enter the device password first to obtain the setting permission.

10.11.8 Backing up Program to External Storage

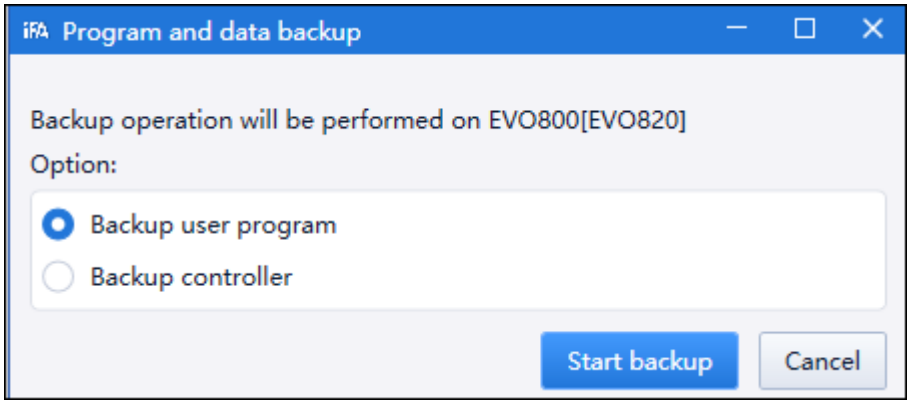
Back up the running program and data from the PLC device to an external storage device.

Prerequisites

- The current device supports program and data backup functions.
- An external storage device (SD card or USB drive) is installed on the current PLC device.
- The current PLC device is in a STOP status.

Backing up program to external storage

1. Select "Online > Backup program to external storage" in the menu bar.
2. In the pop-up "Device scanning" window, click "Start search".
3. Select the target device, click "OK", and the "Program and data backup" window appears.
4. Users can choose either "Backup user program" or "Backup controller".



   - Backup user program: Backs up the user program files.
   - Backup controller: Performs a complete backup of the controller, including system configuration (time, time zone, date, device gateway, etc.) in addition to user program files.
5. Click "Start backup". After completion, a window pops up, indicating successful backup.

Restoring from external storage

1. Insert the SD card containing the backup files into the card slot of the target PLC.
2. Power off and on the PLC.
   During power-on, the PLC will automatically read the backup data from the SD card and perform the restore operation.
3. If the PLC's segment code screen displays "100", it indicates that the restore operation is completed.

NOTE

- Before backing up, it is recommended to format the SD card or USB drive to avoid issues during the backup process.
- The EVO500 series PLC firmware V1.2 version does not currently support backup functions.
- The EVO760 series PLC only supports backup using a USB drive, and only one USB drive can be inserted. If two USB drives are inserted, the backup fails.
- The EVO800 series PLC supports backup in both single and multi-machine modes, but in redundant mode, only the main device supports backup, while the backup device does not support backup.
- The EVO1000 series PLC currently only supports backing up user programs by default and does not allow user selection.
- During the backup process, do not remove the USB drive, as this may cause backup abnormalities.
- When restoring the PLC, it supports inserting two USB drives simultaneously:
  - If both USB drives contain backup files, the file from the first inserted USB drive will be restored by default.
  - If only one USB drive contains backup files, the backup file will be selected for restoration.
- If device permissions have been set and activated, this operation is subject to permission control. For details, see 9 Project Security Protection.

10.11.9 Exporting Device Data

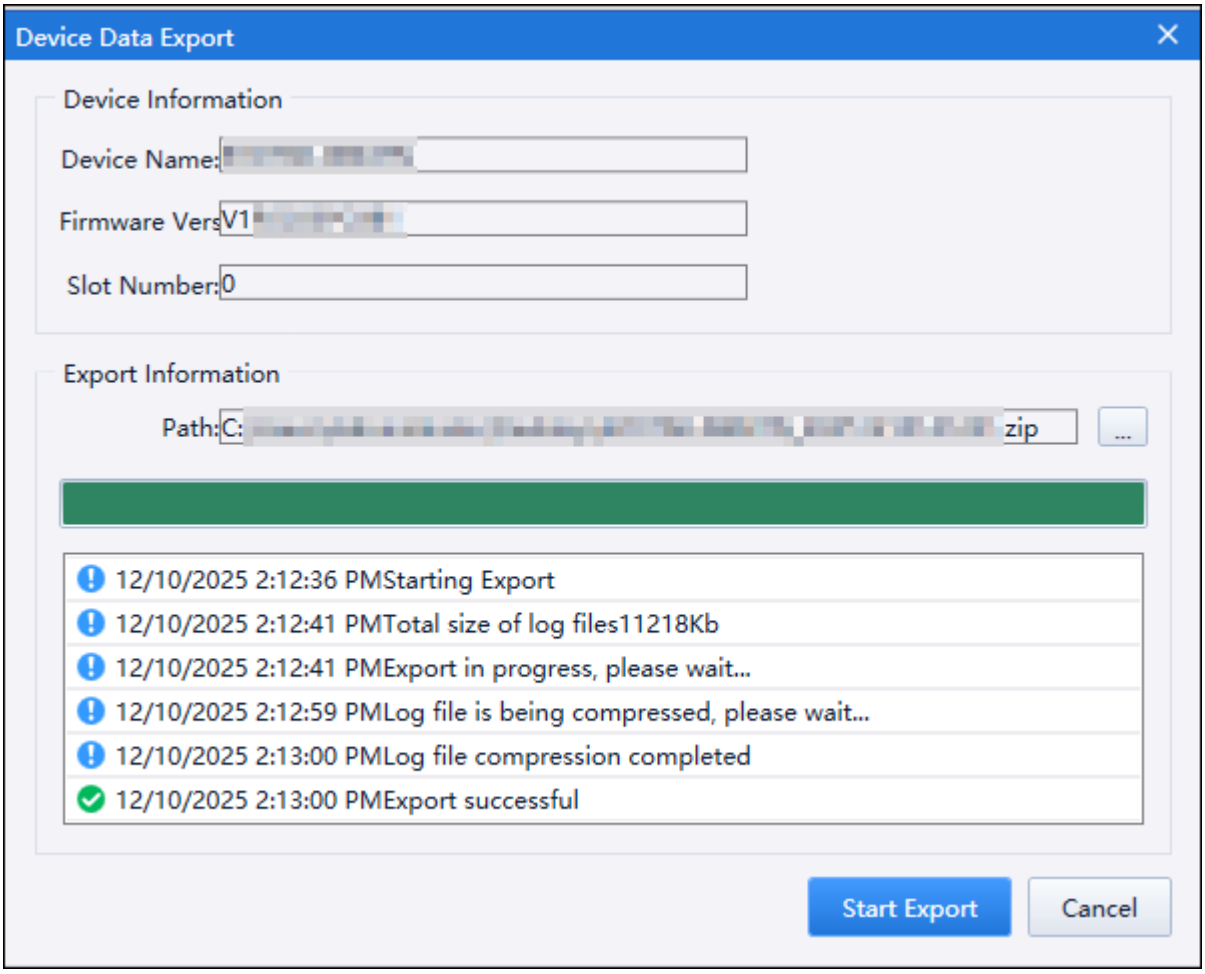
The data recorded in the device, such as device operation logs, can be exported through the device data export function for analysis.

Prerequisites

The IP addresses of the PLC device and the PC is in the same local network segment.

Procedure

1. Select "Online > Device data export" to enter the device scan page.
2. Select the PLC device to be exported, and enter the export page.



Device name: The name of the current device

Firmware version: The version number of the device firmware

Slot number: The slot number where the device is located

3. Select the path for the export file and click "Export" to start the device file upload process, which will generate a .zip file in the selected path.
   Path: Select the storage location of the export file on the PC's disk. The default file name that supports user modification is "current device name + date and time", such as "PLC1_20240118120012".

NOTE

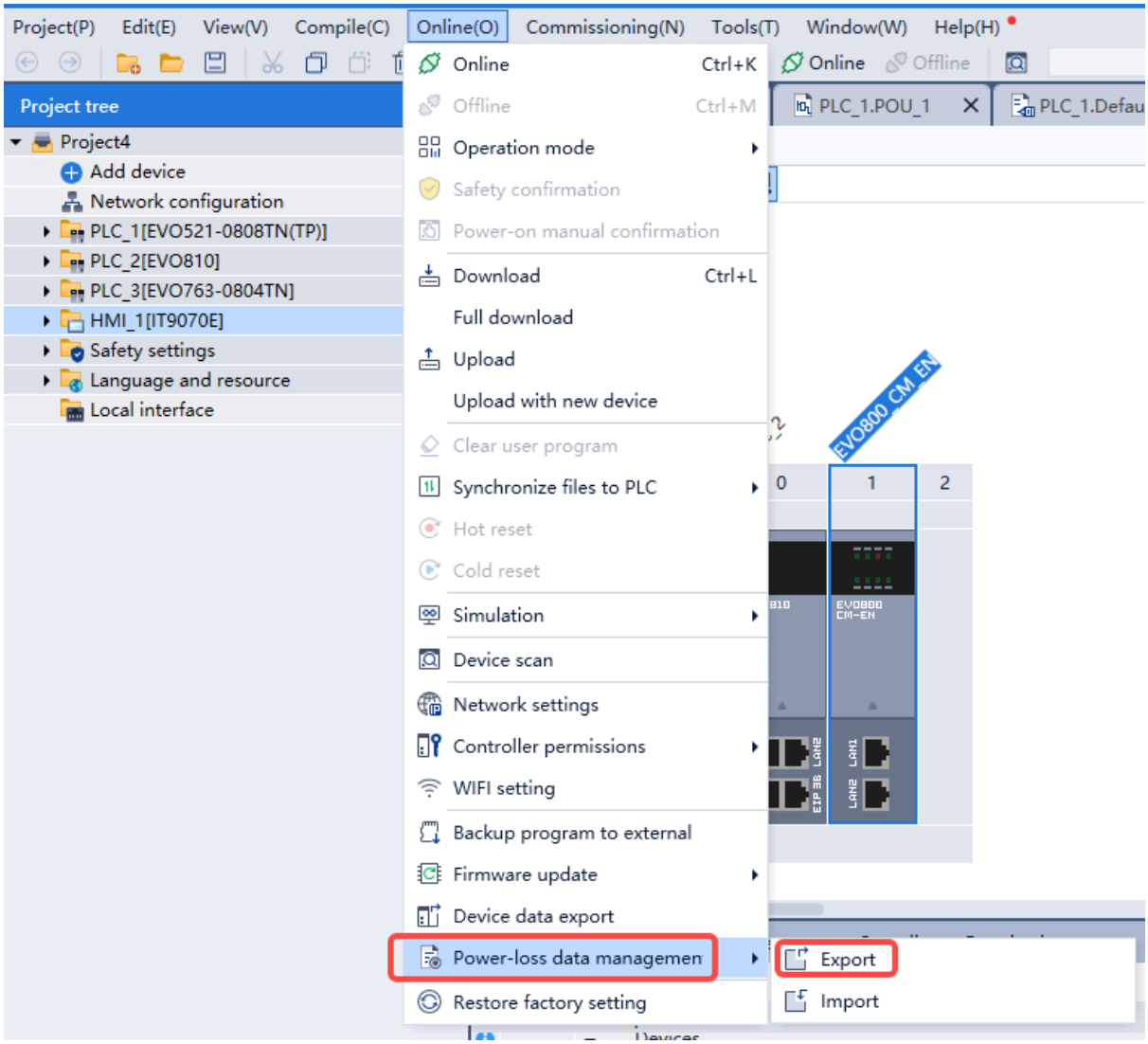
Only one PLC device can be exported at a time.

10.11.10 Power Failure Data Management

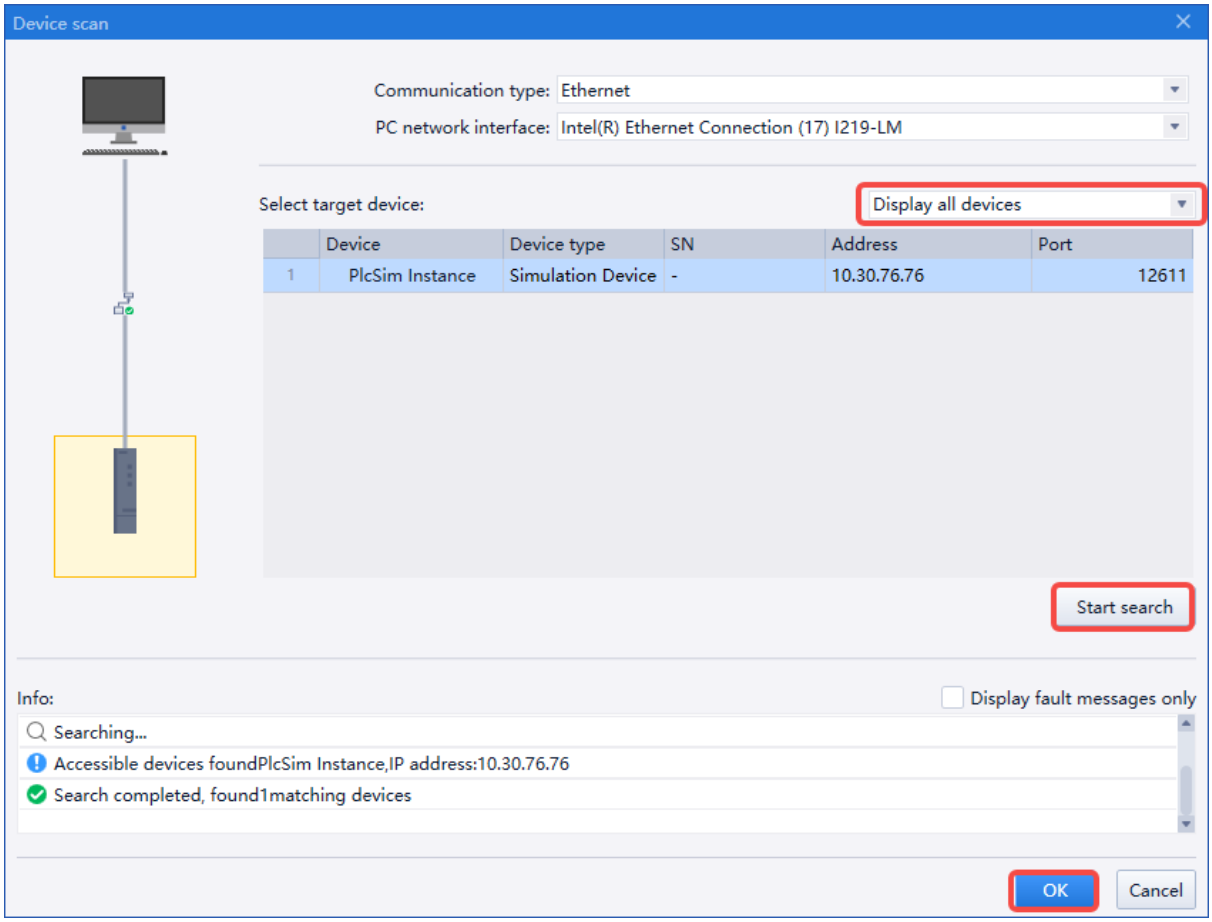
Power failure data management is used to back up and restore the device's data upon power failure to prevent information loss due to unexpected power outages. You can back up the device's power failure data using the export operation, and restore the backed-up data to the device using the import operation.

Export

1. Select "Online > Power failure data management > Export" in the menu bar.



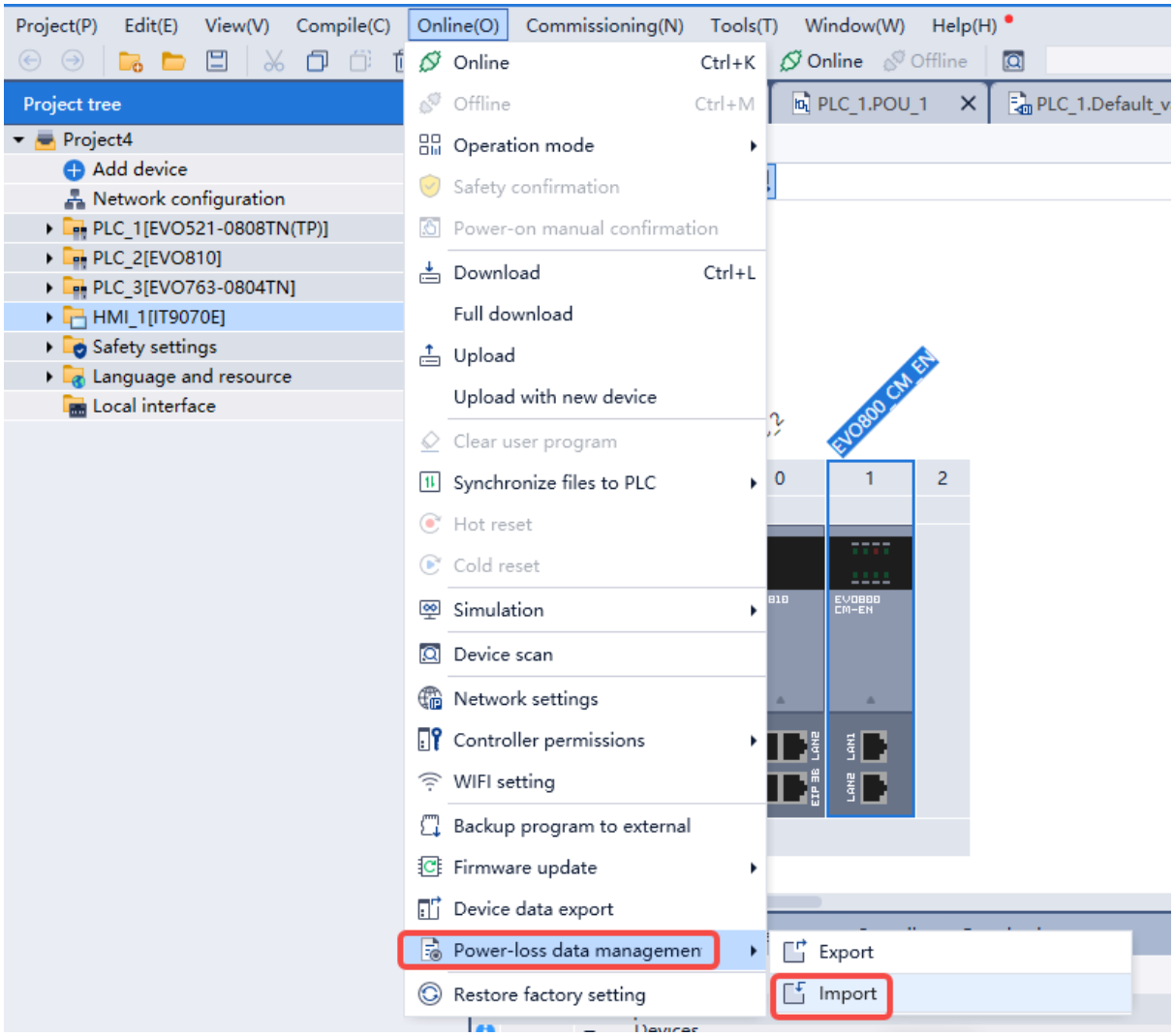
2. In the pop-up device scanning window, click "Start search". The system will automatically scan for all devices and support filtering by device type. Select the target device for data export and click "OK", as shown in the figure below:



3. In the pop-up dialog box, select the folder for data export. The system will display a pop-up message: "Executing, please wait...". If the execution succeeds, the system will prompt that the power failure data has been successfully uploaded. If the execution fails, a fault will be reported. Please troubleshoot and handle the issue according to the prompt information.
4. The exported power failure data will be saved in the specified folder, with the default file name "PersistentBackup.rgfile".

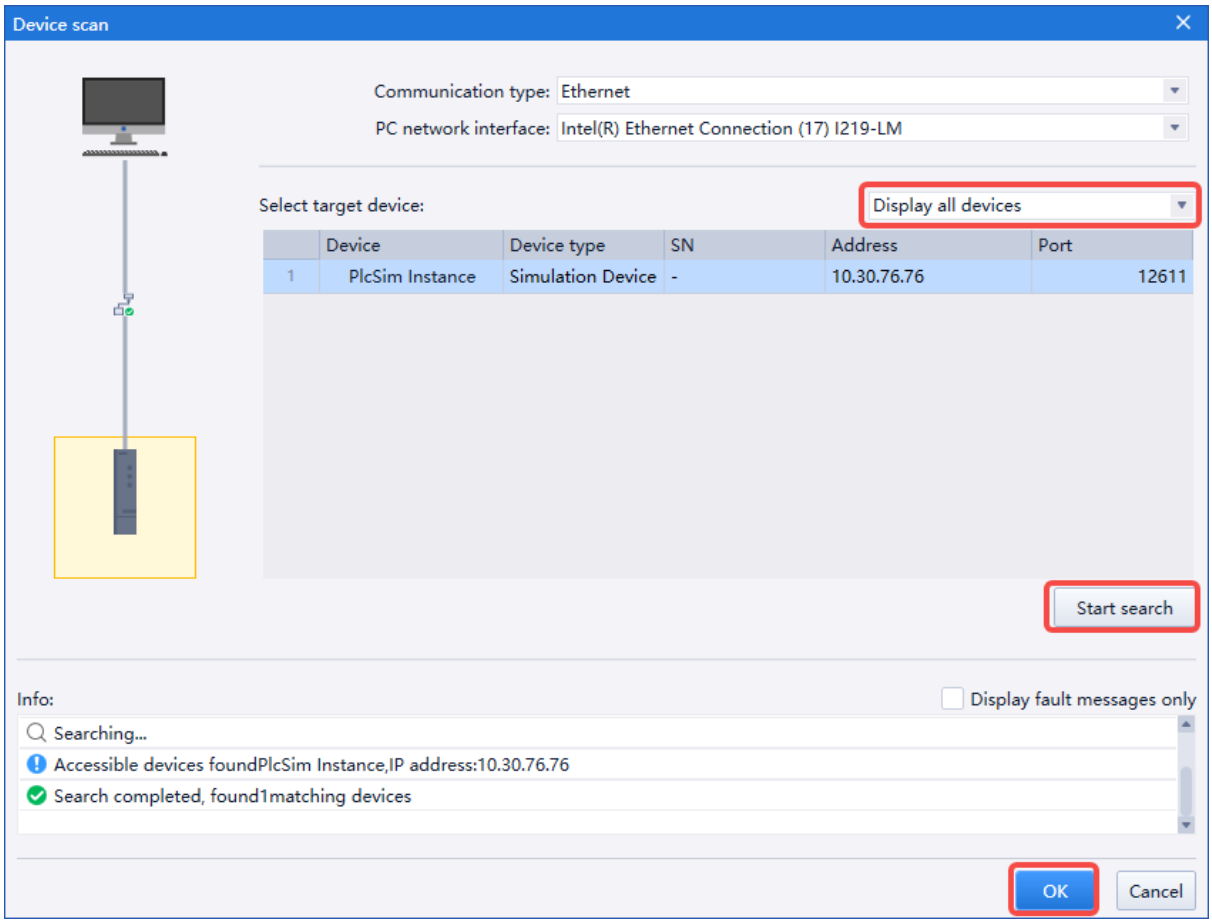
Import

1. Select "Online > Power failure data management > Import" in the menu bar.



2. In the pop-up device scanning window, click "Start search". The system will automatically scan for all devices and support filtering by device type. Select the target device for data import and click "OK", as

shown in the figure below:



3. In the pop-up dialog box, select the power failure data file in .rgfile format to be restored and click "Open".
4. After successful import, the system prompts that power failure data has been transferred to the PLC. Power off and restart the device to restore the data. If the execution fails, a fault will be reported. Please troubleshoot and handle the issue according to the prompt information.

10.11.11 Restoring Factory Settings (EVO800, EVO1000)

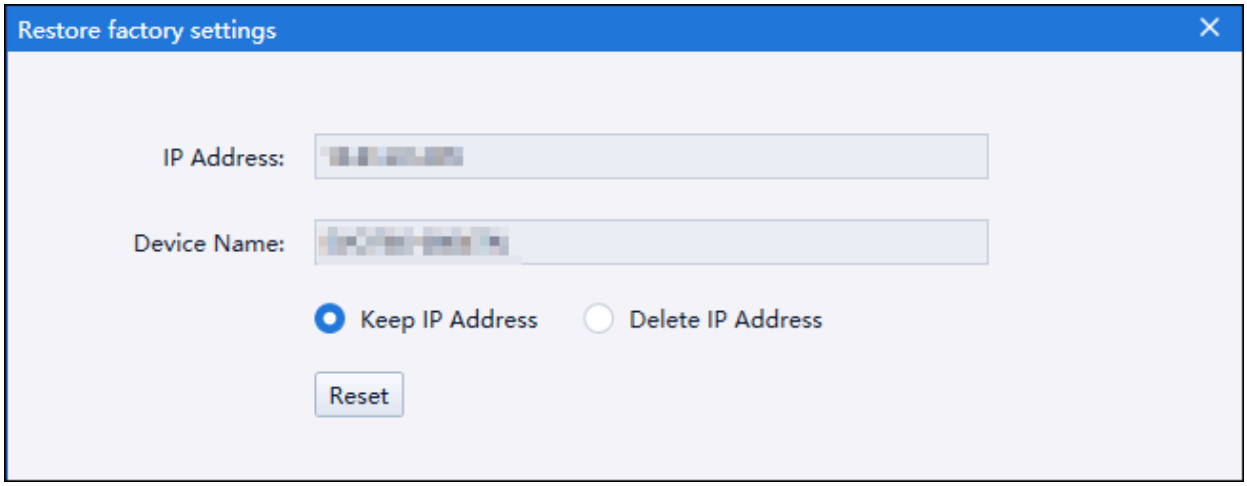
Restore the factory settings for the currently connected PLC device, which can be in single mode or redundant mode.

This function can be performed in the following two ways.

Restoring factory settings through the menu bar

Procedure

1. Select "Online > Restore factory setting" to enter the device scan page.
2. Select the target PLC device to enter the factory restoration page.



Restoring factory settings through "Online and diagnostics"

Prerequisites

- A project has been created and a PLC device has been added.
- The device has been switched to online.

Procedure

1. Double-click "Online and diagnostics" in the left project tree and select the "Restore factory setting" tab page.
2. Select the target PLC device to enter the factory restoration page.

Function description

Keep IP address: Maintain the IP address set by users without making any changes.

Delete IP address: Clear users program and restore the IP to the initial IP (192.168.1.88).

Restoring factory settings will reset the default time to 8:00 A.M., January 1, 2000.

NOTE

- The factory restoration operation can only be performed on the currently connected PLC device.
- Both the main device and backup device support deleting the IP for factory restoration.
- When two PLCs are in redundant operation mode, restoring the factory settings of the master device does not affect the slave device.

10.11.12 Web Diagnostics and Monitoring

The web server supports the following browsers:

PC:

- Google Chrome 63.0 or later
- Mozilla Firefox 67.0 or later
- Microsoft Edge 18.0 or later
- MacOS Safari 12.0 or later

Mobile:

- iOS Safari 13.0 or later
- Android Chrome 80.0 or later
- Android Firefox 68.0 or later

> NOTE
>
> Web diagnostics and monitoring functions are supported only on EVO500 series PLCs from V1.4.2.5 and later.

User Login

Enter the device's IP address in the browser, and log in to the 10.6.14.7 web interface using the username and password set on the user management tab of the web server.

If anonymous login has been set in the permission settings of the user management, select "Anonymous access" and no username or password is required.

Home Page

After logging in, go to the main page of web diagnostics and monitoring.

Item		Description
Product information		Specific model information of the product, and the current RTC time /time zone of the controller
System status		Device operating status
CPU operation		The device flashes based on online PLC. Click the "Device flashing" button once and the device will flash for 5s.
Device information (in EVO800 series redundant mode, the main/ backup device status will be displayed; in single mode of EVO700 series and other devices, device information will not be displayed)		Displays the optical fiber status information of the main/backup devices.
Device image		Displays the device image.
Basic information	Project name	Name of the project created by the user
	Controller version	Controller firmware version
	Product model	Specific product model, such as EVO800
	SN code	PLC hardware serial number
	Main control module (Displayed in EVO800 redundant mode)	For redundant control modules, the model of the main control module is displayed, such as EVO820.

Item		Description
	Redundant synchronization module (Displayed in EVO800 redundant mode)	For redundant control modules, the model of redundant module is displayed, such as EVO800-RM.
	Last compilation time	The last project compilation time in the controller (including incremental compilation)
	Last download time	The last project download time in the controller (including online modifications)

Online diagnosis

Displays diagnostic information of the current device.



When the online diagnostic information section shows blank content, manually clear the browser cache and refresh the web server to retrieve the latest online diagnostic information.

Function	Description
Search	By entering a fault code in the search input box, the corresponding fault information is displayed.
Advanced filtering	Use the drop-down box to select fault level and the filtered fault information is displayed accordingly. Fault levels include critical fault, severe fault, alarm, information, and unknown.
Freeze display	Diagnostic information can be frozen for display. After freezing, diagnostic information will no longer be refreshed in real time.
Export controller log	Click to export current diagnostic information, which is saved to a user-defined local path.
Previous page/Next page	Click next or previous page to navigate through diagnostic information.
View fault details	Click the fault level to display detailed fault information, including the cause of the fault code and recommended solutions.
Sort	Sorting can be performed by level, time, or in ascending/descending order.

Variable monitoring

Monitor changes in variable values.

Function	Description
Write	After clicking "Write", the input prepared value will be written, and the value in the variable table will be synchronized and refreshed. Variable write rules: • INT/LINT/SINT/SINT/DINT/WORD/DWORD/LWORD: Enter decimal data. • BOOL: Select TRUE or FALSE. • TIME: TIME type supports input format "XX (unit)", such as 1 h, 1 m, 1 s, 1 ms, and combinations of units h/m/s/ms. For numeric input without a unit, the default unit is ms. • Ltime: Supports input format "XX (unit)", such as 1 d, 1 h, 1 m, 1 s, 1 ms, 1 us, and combinations of units d/h/m/s/ms/us. For numeric input without a unit, the default unit is ms. • Time of DAY: Click the calendar to select time. • DateAndTime/Date: Click the calendar to select date and time. • Enumeration: Select enumeration values from the drop-down list.
Read	Refreshes variable values.
Previous page/Next page	Click next or previous page to navigate through variable information.
Search	Enter an expression to search; supports fuzzy search.

Network port communication information

Displays the network port communication information of the device. The information is read-only, uneditable, and refreshed every 1 second.

Parameter	Description
Port	The port refers to the actual physical port of the device.
Port IP	The IP address of the port.
Subnet mask	The subnet mask of the port.
Connected device information	The IP address of the connected device. If connected via a personal PC, the PC's IP address is displayed.
Communication type	The protocol type used by the device. Currently supports communication protocols such as EIP/OPCUA/WEB/Modbus TCP.
Connection status	Displays whether the device's connection status is normal or faulty. • For active connection, the slave status matches the project tree status, and the PLC acts as the master (such as EIP master and Modbus TCP/RTU master). • For passive connection, the connection status displays "Connected", and the PLC acts as a slave (such as EIP slave, Modbus TCP/RTU slave, WEB, OPCUA server, or HMI-CLASS3 connection).

Wi-Fi configuration

Currently, only EVO800 series PLCs support Wi-Fi settings, and the function is the same as "Wi-Fi settings" under "Online" in the menu bar.

Parameter	Description
Network name	The prefix is fixed as the PLC device name, and the suffix can be customized by the user. It can include English letters, Chinese characters, numbers, and underscores, but cannot start with a number or end with a number or underscore.
Password	The Wi-Fi password is initially empty, which can be set according to the password naming requirements. The password can consist of letters, numbers, and special characters, and must include at least one digit, one uppercase letter, and one lowercase letter, with a length between 8 and 32 characters.
Frequency band	The default is 2.4 GHz and cannot be modified.
Security	Supports WPA2 (default) and WPA3. • WPA2: Uses AES encryption technology to provide reliable network protection. • WPA3: Enhances encryption algorithms to resist brute password attacks and offline attacks, significantly improving privacy and security.
Max. number of clients	The default is 5 and cannot be modified; new connections will be rejected if more than 5 clients are connected, while existing connections remain unaffected.
Wi-Fi activation status	Activation is only allowed when network name, password, security settings, and client count are correctly entered. After activation, the parameter page becomes grayed out and cannot be edited.

10.11.13 Time Settings

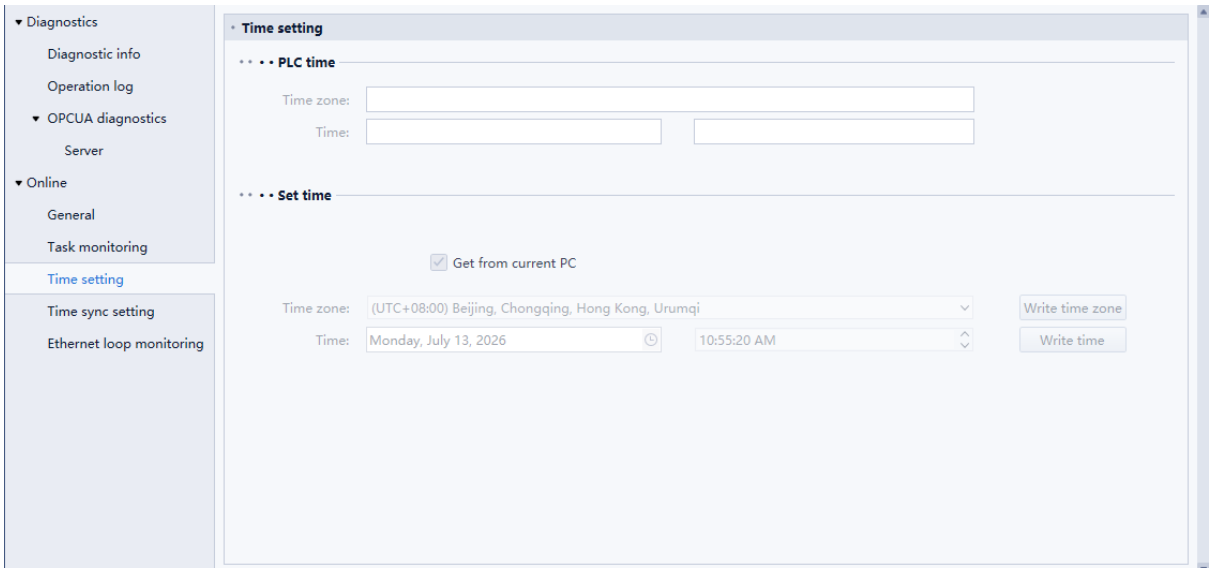
Time settings are used to set the time reference for the PLC, which is supported when the PLC is online.

Prerequisites

- The device is in an online status.
- The PLC has a built-in battery or an RTC clock card.

Procedure

1. Double-click "Online and diagnostics" in the project tree.



2. Set the time.

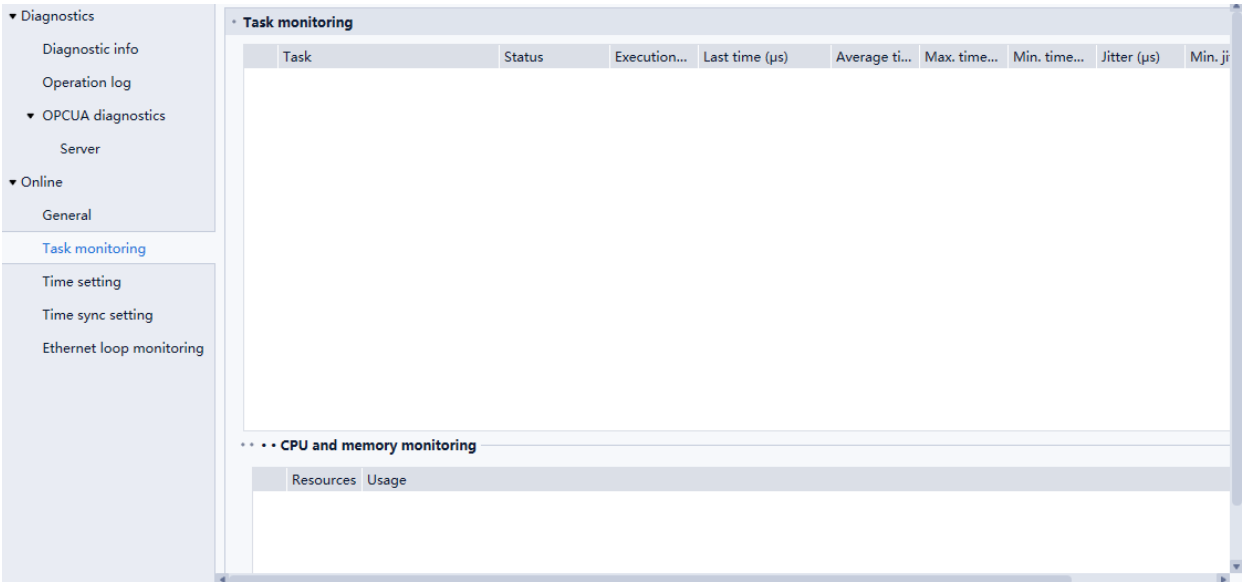
- PLC time: The current time and time zone of the PLC, with a refresh cycle of 1 second.
- Set time: The desired time reference for the PLC, which can be obtained from the current computer or by entering the corresponding time zone and time.

NOTE

To use the RTC (Real-Time Clock) function, ensure that the PLC has a built-in battery. If the PLC does not have this feature (such as EVO500), a dedicated RTC clock card (such as GE20-RTC) must be selected to achieve the time retention function after power failure.

10.11.14 Task Monitoring

Double-click "Online and diagnostics" in the project tree.



- Task monitoring
  The "Task monitoring" tab displays the information of tasks currently contained in the PLC and the running status of each task.
- CPU and memory monitoring

  The explanation for CPU usage is as follows:

  - 0% to 80%: CPU usage is normal. The PLC runs relatively stably, and functions such as logic execution, bus synchronization, I/O refresh, data synchronization, and data saving can be used normally.
  - 80% to 100%: CPU usage is high or has reached its limit, reducing the stability of the PLC's operation.

    Main impacts of high CPU usage

    - The stability of EtherCAT operation cannot be guaranteed, and there may be issues with EtherCAT slaves dropping out or losing synchronization.
    - In severe cases, it may cause the PLC to enter a false alarm status, making it impossible to log in to the PLC.
    - Power failure storage data cannot be saved.
    - EIP/OPC UA/Modbus and other related communication services may experience slow data refresh or disconnection risks.
    - Online modification or PLC program download may become slower or fail.
    - Monitored PLC variable values may experience slow refresh or inability to refresh.

10.11.15 Time Synchronization Settings

The time synchronization function allows all devices in the network to obtain time from the same source, ensuring time consistency across all devices.

Prerequisites

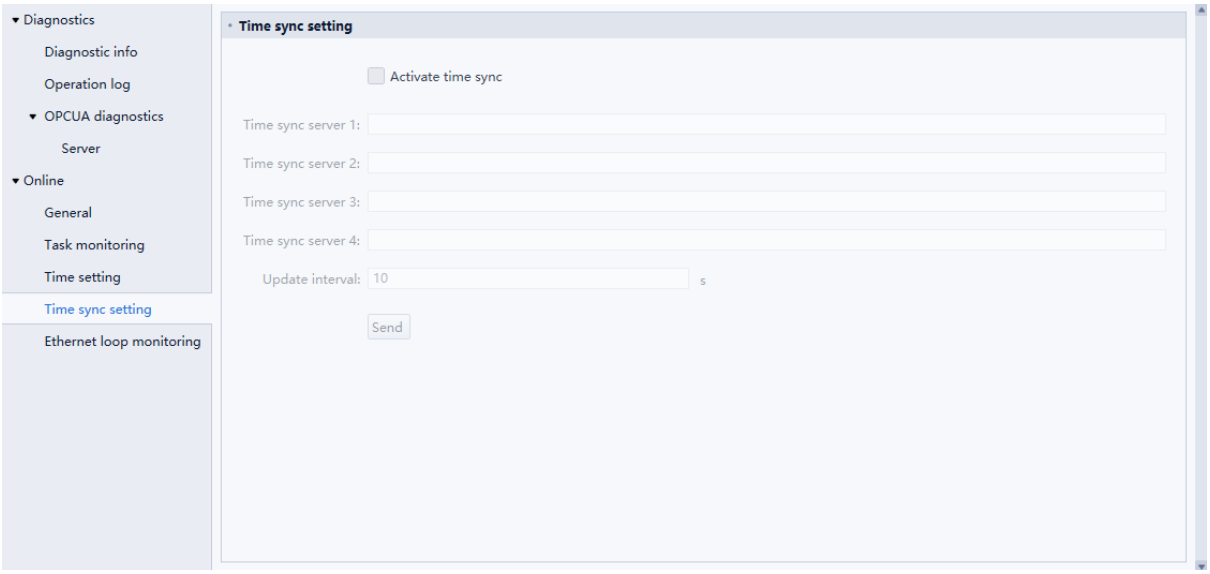
The device is in an online status.

Procedure

1. Double-click "Online and diagnostics" in the project tree on the left.

2. Go to the "Online > Time sync setting" tab.

3. Check "Activate time sync", configure the valid IPs for time sync servers and the update interval, and click "Send".



Parameter	Description
Activate time sync	Unchecked by default. When checked, it activates the time synchronization function.
Time sync server 1/2/3/4	Enter valid IP addresses for time sync servers. At least one IP address must be provided. If all four are configured, the priority order is 1 > 2 > 3 > 4.
Update interval	The interval for time synchronization updates. Range: 10s to 86400s (1 day).

4. After time synchronization, a pop-up displays "Execution succeeded".

NOTE

- If the current program contains the 18.4.1.34 SysHC_NtpClient(NTP calibration) instruction and it is enabled, this configuration will fail. Please disable the NTP instruction before configuring on this page.
- The EVO800 series PLC only supports setting the time for the main device (PLC), which then synchronizes the time to the backup device.
- If both time synchronization settings and time settings are active, the time setting information is only effective within the update interval of the time synchronization settings. For example, if the time written is 20:00:00 and the NTP synchronization update interval is 10 minutes, the written time will be maintained for a maximum of 10 minutes, after which the new round of NTP synchronization will overwrite 20:00:00 with the server time.
- NTP synchronization is specific to the device and is independent of the software system.

10.11.16 DLR Ring Network Management (EVO800)

Set the communication parameters of the DLR ring network during the online operation of the PLC, and monitor its real-time operating status. Currently, only the EVO800 series PLC supports this function.

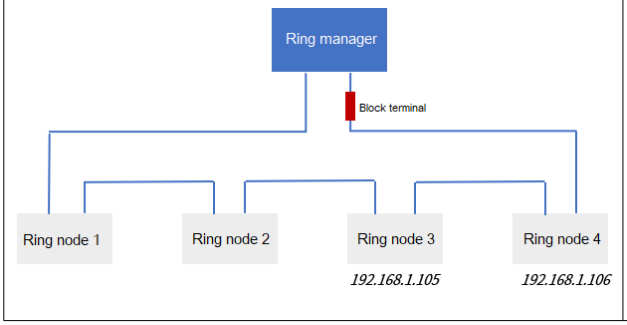
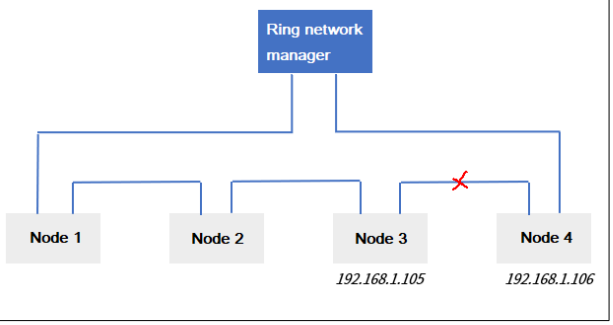
Prerequisites

- The ring network has been established.
- The current PLC is the ring network manager.
- The EVO800 series PLC is in online operation mode.

Function entry

Double-click "Online and diagnostics" in the project tree.

Real-time monitoring of the ring network:

The ring network is running well	A break has occurred in the ring network
	

When opening this interface, the system reads the current status of the ring network; it then refreshes every 1 second until the interface is closed.

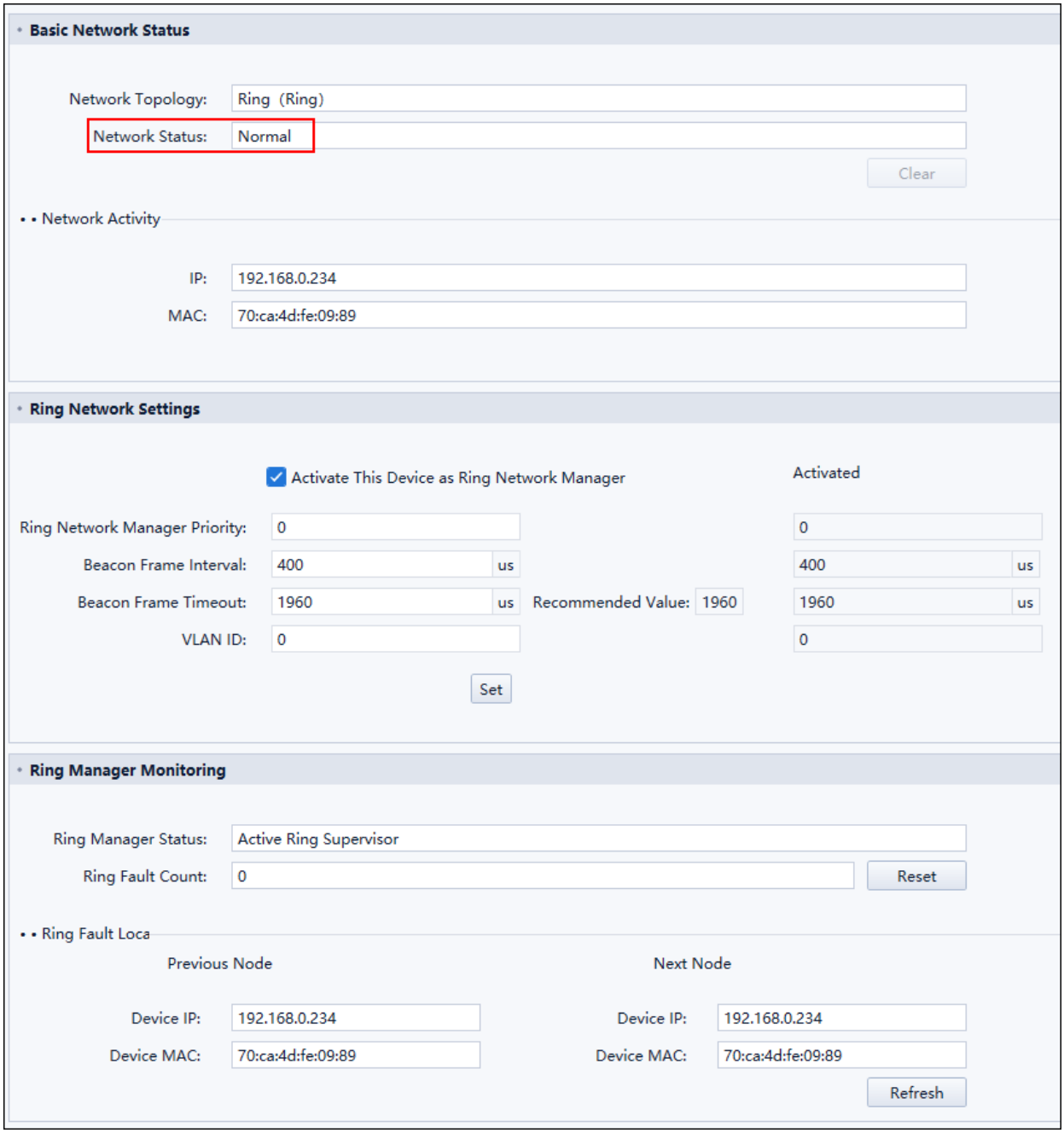


Figure 10-11-16-1 Monitoring results when the ring network is running well

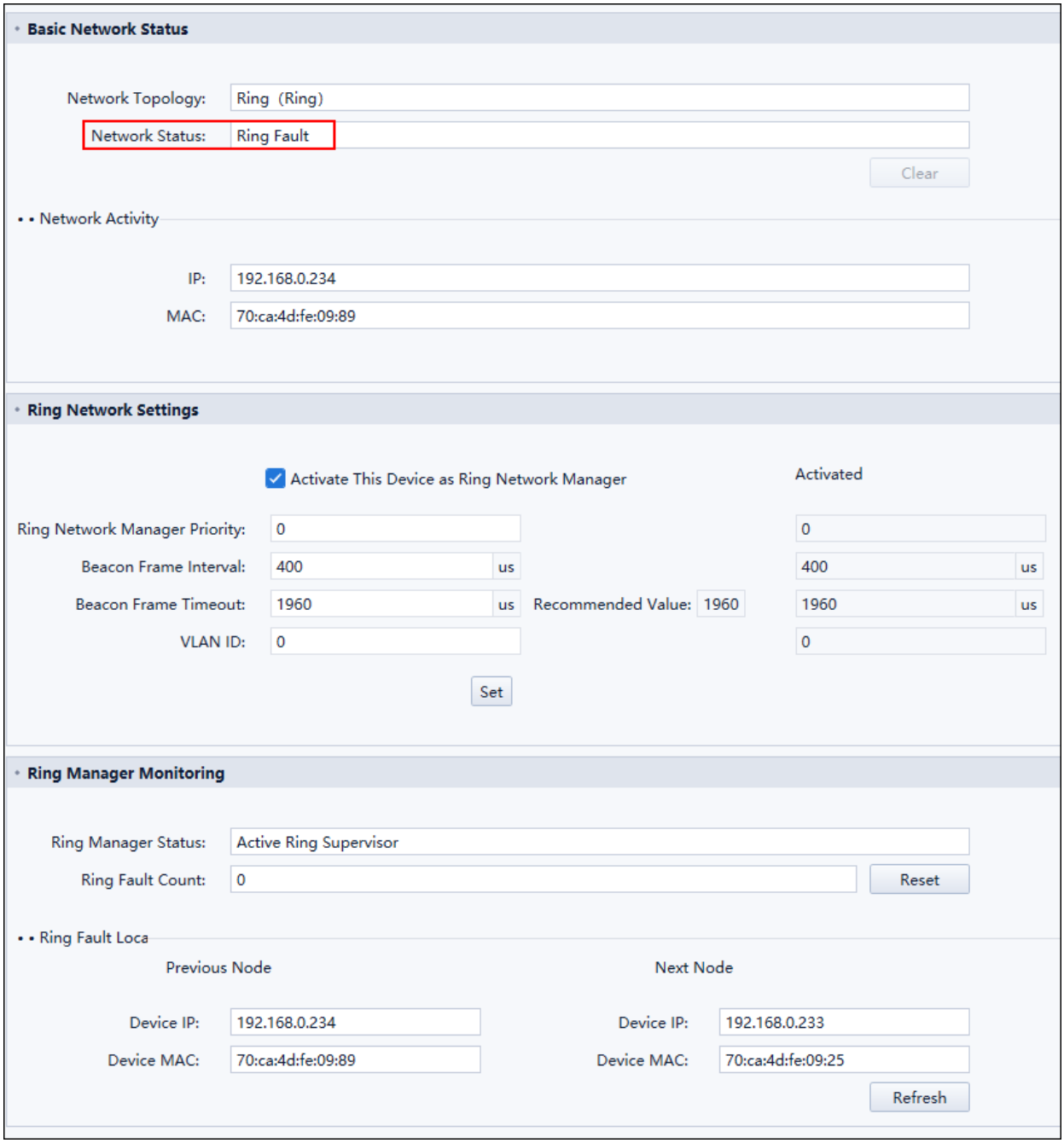


Figure 10-11-16-2 Monitoring results when a break occurs in the ring network

	Item	Description	Remarks
Basic network status	Network topology	Supports ring topology and linear topology. • Ring topology: Related to the configuration of devices in the network. As long as one device in the network is enabled as a ring network manager, it will become a ring network; • Linear topology: If no device in the network is enabled as a ring network manager, it will become a linear network.	-

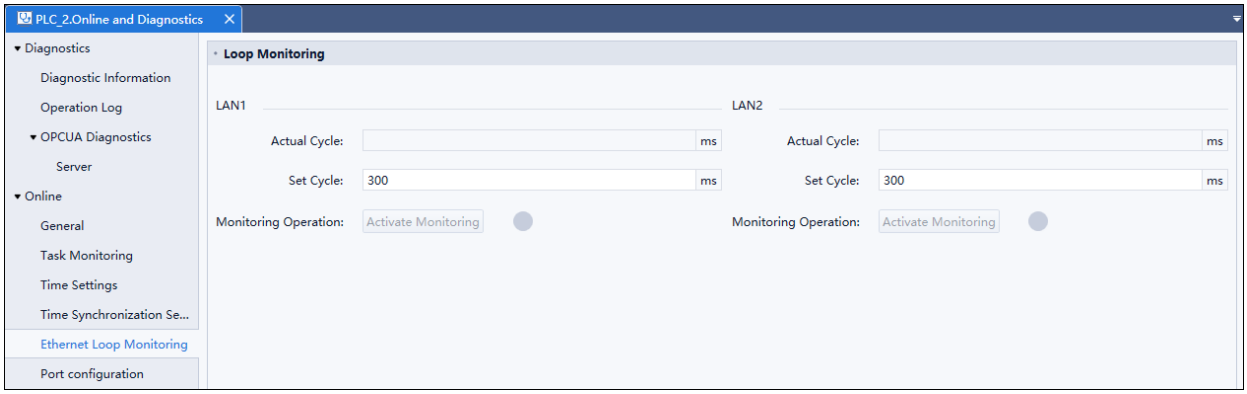
	Item	Description	Remarks
	Network status	Current network operating status, with five statuses: • Normal: There is an active ring network manager, and the network is running well without any faults. • Ring fault: (Case 1) There is an active ring network manager, and a fault has occurred in the ring network; (Case 2) No ring network manager is active. • Unexpected loop detected: A network storm has occurred in the ring network. • Partial network fault: A partial network fault occurs when transmission is lost in only one direction on the network. • Rapid fault/restore cycle: Link faults occur more than five times within 30 seconds.	-
	Network activity	Displays the IP and MAC addresses of the current ring network manager.	-
Ring network settings	Activate this device as ring network manager	For a ring network, at least one ring network manager must be configured; otherwise, it becomes a linear network.	The area is divided into two parts: Left side: New value input, click to apply settings. Right side: Current value monitoring.
	Ring network manager priority	When multiple nodes are activated, the node with the highest priority number, ranging from 0 (low) to 255 (high), becomes the active ring network manager, and other ring network management devices automatically become backup devices.	
	Beacon frame interval	The time interval at which the ring network manager sends beacon frames, with a range from 100 μs to 100 ms.	

	Item	Description	Remarks
	Beacon frame timeout	The total waiting time for the manager node or ring network node after the ring network manager sends a beacon frame. Range: 200 μs to 500000 μs, with a default of 1960 μs. The timeout time must be at least twice the interval. If the entered value is less than twice the interval, the system automatically sets it to twice the interval. Note: The system supports up to 50 node configurations by default. When the total number of nodes exceeds 50 by adding switches to expand the topology structure or when the configuration data frame length exceeds 1000, the DLR (Device Level Ring) timeout time should be extended accordingly to ensure network stability.	
	VLAN ID	In most cases, there is no need to change the VLAN ID. When using DLR dedicated switches in the ring network, VLAN ID needs to be set to a unique value from 0 to 4094.	
Ring manager monitoring	Ring manager status	Displays the role and status of the current device in the ring network: Backup Active ring supervisor Normal ring node Non-DLR topology Bad current ring parameters	-
	Ring fault count	Cumulative number of the detected ring faults. Click "Reset" clear the fault count.	-
	Ring fault location	Displays ring network fault information, including status and the breakpoint location between two devices.	-

10.11.17 Ethernet Loop Monitoring

During online operation of the PLC, the iFA software periodically monitors whether a loop fault has occurred in its Ethernet network. Information about loop fault occurrence or elimination is recorded in the diagnostic information and operation logs.

Double-click "Online and diagnostics" in the project tree.



This interface displays all Ethernet ports on the current PLC device, with port names matching the actual PLC device for easy identification.

- Actual cycle: Displays the currently executing cycle.
- Set cycle: The setting range is 100 ms to 30000 ms.
- Monitoring operation: Includes the "Activate monitoring" button and indicators showing real-time monitoring status. Click "Activate monitoring" to monitor the loop status, and the indicator turns on; otherwise, monitoring is disabled and the indicator turns off.

10.11.18 Monitoring Main/Backup Device Status (EVO800)

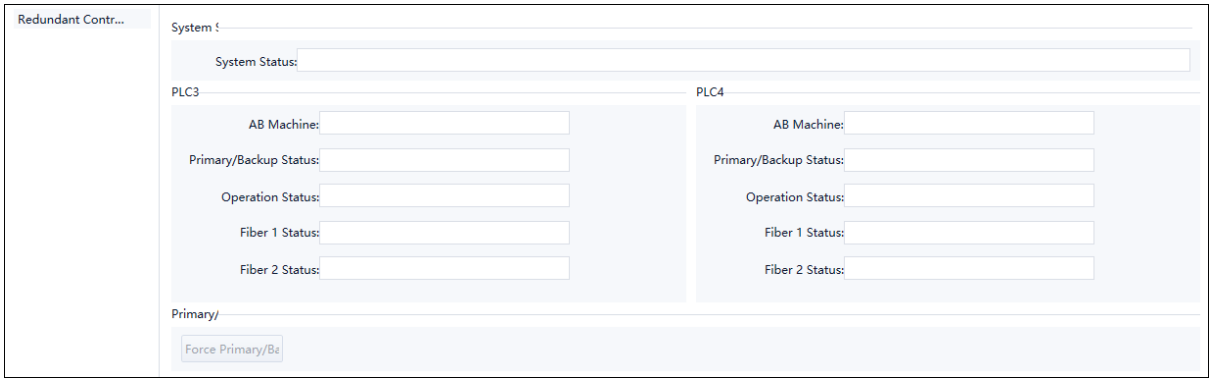
Redundant system supports online switching between the main and backup devices, and monitoring the status of the devices.

Prerequisites

- A redundant device has been added.
- The main or backup device has been switched to online status.

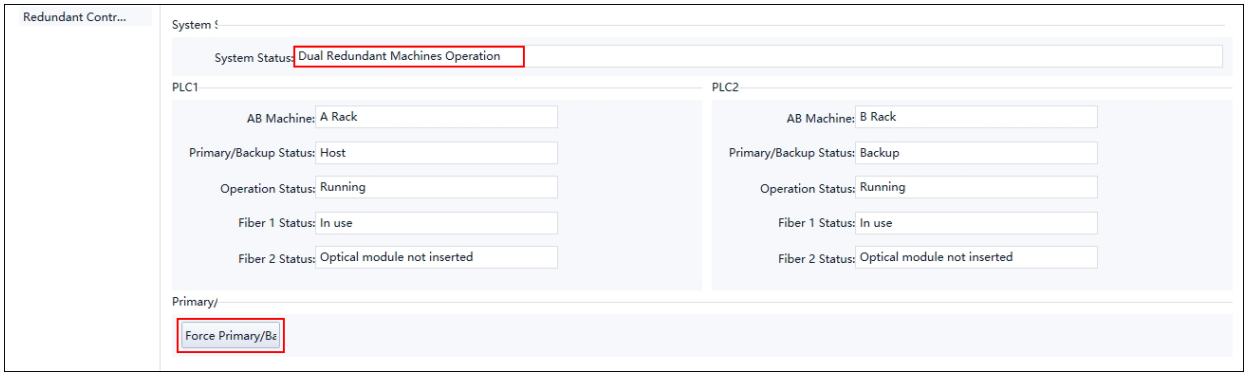
Procedure

1. Double-click the redundant system node in the left project tree to enter the system status tab.
2. View device status.



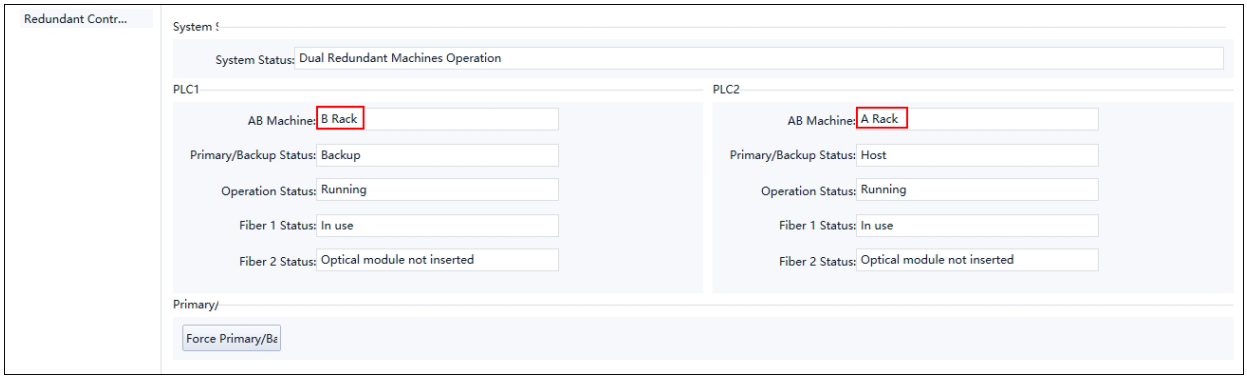
   - A/B device: Unknown, A device, B device
   - Main/backup status: Unknown, main device, backup device
   - Operating status: Unknown, running, stopped, commissioning
   - Fiber 1 status: Not inserted, faulty, in use, on backup
   - Fiber 2 status: Not inserted, faulty, in use, on backup

When the redundant system configuration is complete and the redundant PLC is online, the system status is redundant dual-machine operation. At this point, users can manually switch the main/backup status of the current PLC device.



Click the force switch button and confirm the switch in the pop-up window to switch the main/backup status of the current PLC.

After successful forced switching, click the "Online" icon and the main device rack has changed from the original A rack to the B rack, and the backup device rack has changed from the B rack to the A rack.



NOTE

By default, as long as the redundant system is running normally, any changes to the main device's program or hardware configuration due to downloading will be automatically synchronized to the backup device.

10.11.19 Calling a Stack

The "Call stack" interface displays the current PLC call status in real time. When debugging the program step by step in online mode, the "Call stack" interface provides real-time information during single-step program execution, showing the complete call path of the current position reached.

The "Call stack" interface supports the function of double-clicking a stack entry to locate the corresponding position, and identifies the stack top and the viewing stack. The stack list displays the POU name, corresponding position, and instance path of each layer (the instance path is the same as the POU name for non-FB instances).

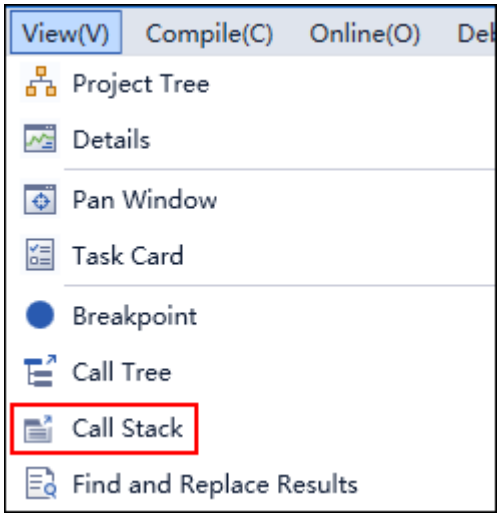
The call stack is displayed as a list of positions, with each position described by POU name, location, and instance path. In the text editor, the location is indicated by line and column numbers; in the graphical editor, the location is indicated by network or element numbers.

Prerequisites

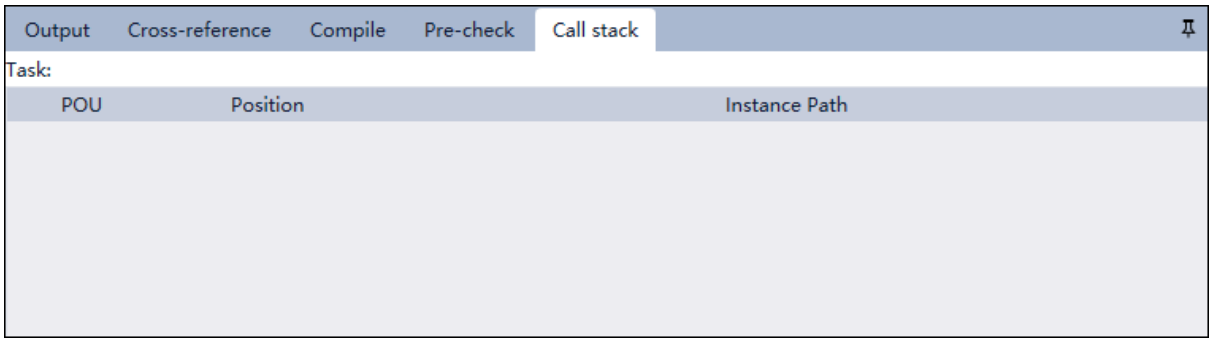
The device is in online mode and is in breakpoint commissioning status.

Procedure

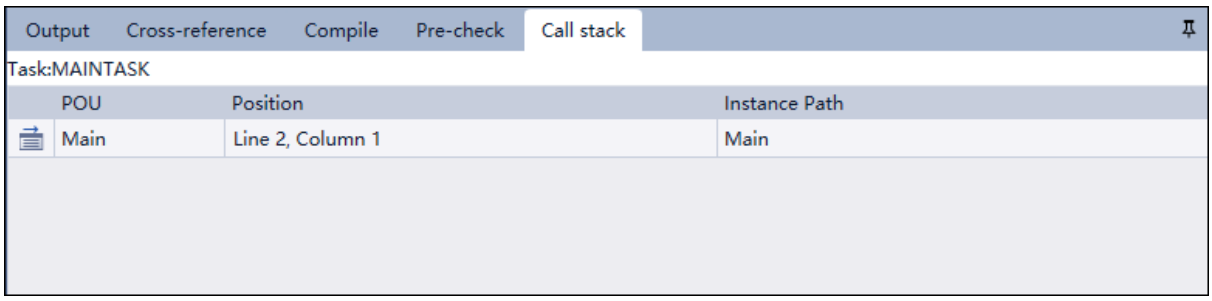
1. Select "View > Call stack" in the menu bar to display the "Call stack" interface.



2. In the monitoring area, click "Call stack" to open the corresponding interface, as shown in the figure below.



3. When debugging the program step by step in online mode, the "Call stack" interface displays the current step's POU name, location, and instance path, as shown in the figure below.



NOTE

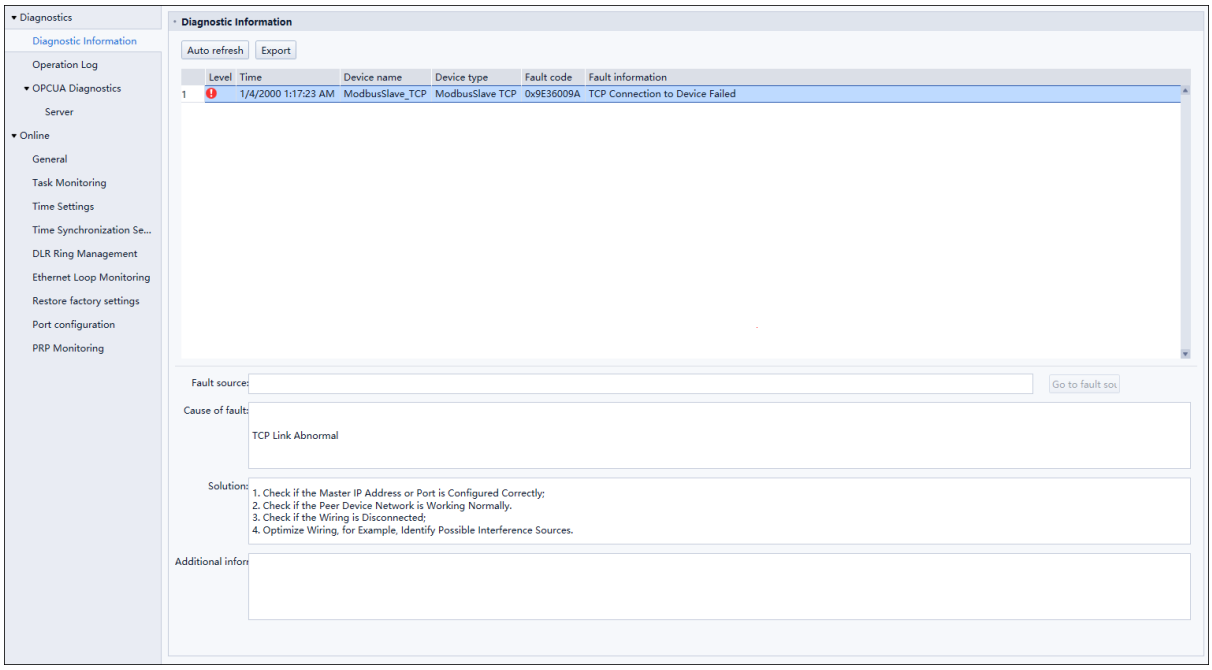
The call stack displays up to a maximum of 10 levels.

10.11.20 Diagnostic Information

Diagnostic information is fault data automatically recorded and generated by the device during operation, including basic fault information, fault causes, and recommended solutions. Through diagnostic information, users can quickly understand the device's abnormal status, accurately locate the root cause of the problem, and complete repairs, thereby ensuring stable device operation.

Viewing diagnostic information

1. Double-click "Online and diagnostics" in the project tree.
2. On "Diagnostic information" tab, view device diagnostic information, including fault level, time, device name, device type, fault code, and fault information.



3. Select a fault entry to display specific details such as the fault source, cause, and solution. Click "Go to fault source" to accurately locate the fault position.
4. (Optional) Above the list, click "Freeze display" to show only the existing diagnostic information without updating newly generated faults. Click "Auto refresh" to resume real-time updates.

Exporting diagnostic information

Above the list, click "Export", select the save path and set the file name, then click "Save" to export the current diagnostic information in .csv format.

10.11.21 Operation Log

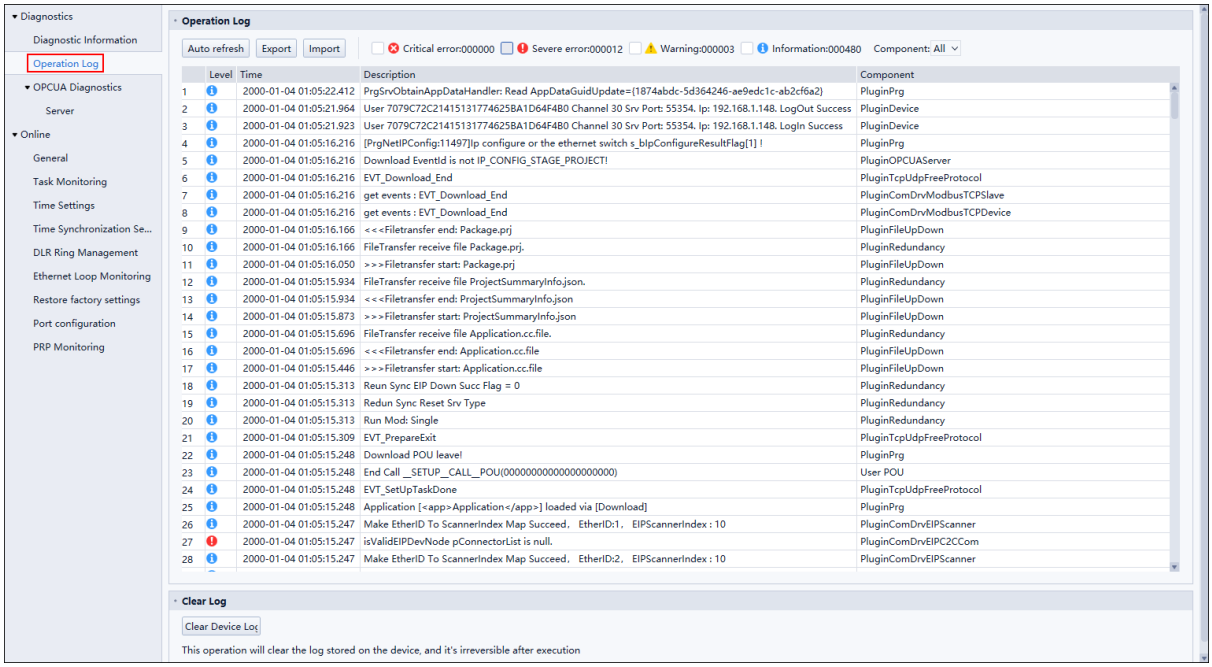
The operation log records log information during the device's operation, including device status, faults, alarms, program execution data, communication status, and user operation records. When a device is faulty, the log helps quickly locate the fault and reproduce the scenario. In addition, the real-time recorded operation data can be used for preventive maintenance, performance evaluation, and optimization.

Prerequisites

The device has been switched to online.

Viewing Operation Logs

1. Double-click the "Online and diagnostics" node in the left project tree to open the "Online and diagnostics" page.
2. On the "Operation log" tab, view the device's current log information, including level, time, description, and associated component.



3. (Optional) Above the list, click "Freeze display" to show only the existing operation logs without updating newly generated logs. Click "Auto refresh" to resume real-time updates.
4. (Optional) In the frozen display state, you can filter by log level and associated component: Check the corresponding fault level to display only logs of that level; select the corresponding component from the "Component" dropdown list to display only logs for that component in the list.

Importing/Exporting logs

- **Export**
  Click "Export", select the save path and set the file name, then click "Save" to export the current operation log in .csv format.
- **Import**
  In the frozen display state, click "Import", select a .csv format log file in the pop-up window to import and display the locally saved log file.

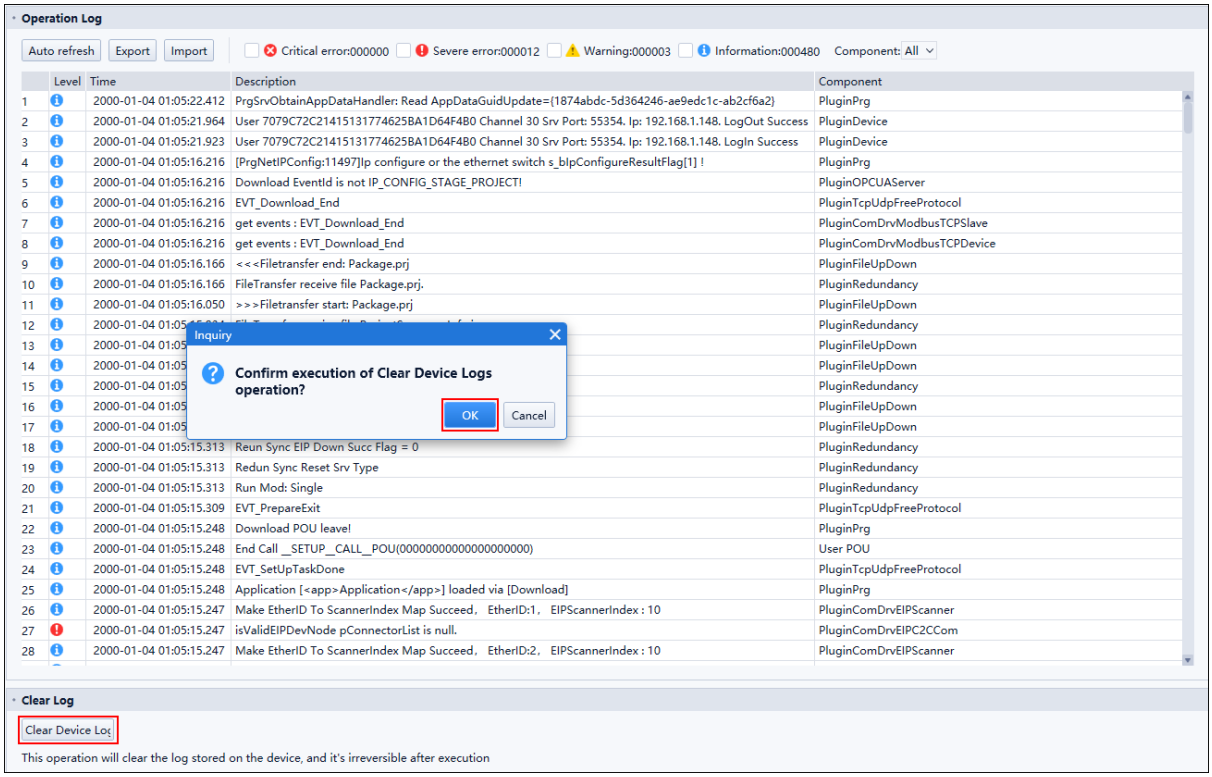
Clearing device logs

When the device is connected to a physical PLC device via iFA and switched to online mode, you can clear device logs. This operation is not supported when the device is connected to a virtual PLC.

CAUTION

This operation cannot be undone. Please proceed with caution.

1. On the "Operation log" page, click "Clear device logs".
2. Click "OK" in the pop-up window to clear the operation log information stored in the PLC device.



10.11.22 Network Port Configuration

This tab is used to configure the network port parameters while the PLC is online, and monitor the current network port operation status.

Restrictions

Network port configuration applies to the following PLC models and firmware versions:

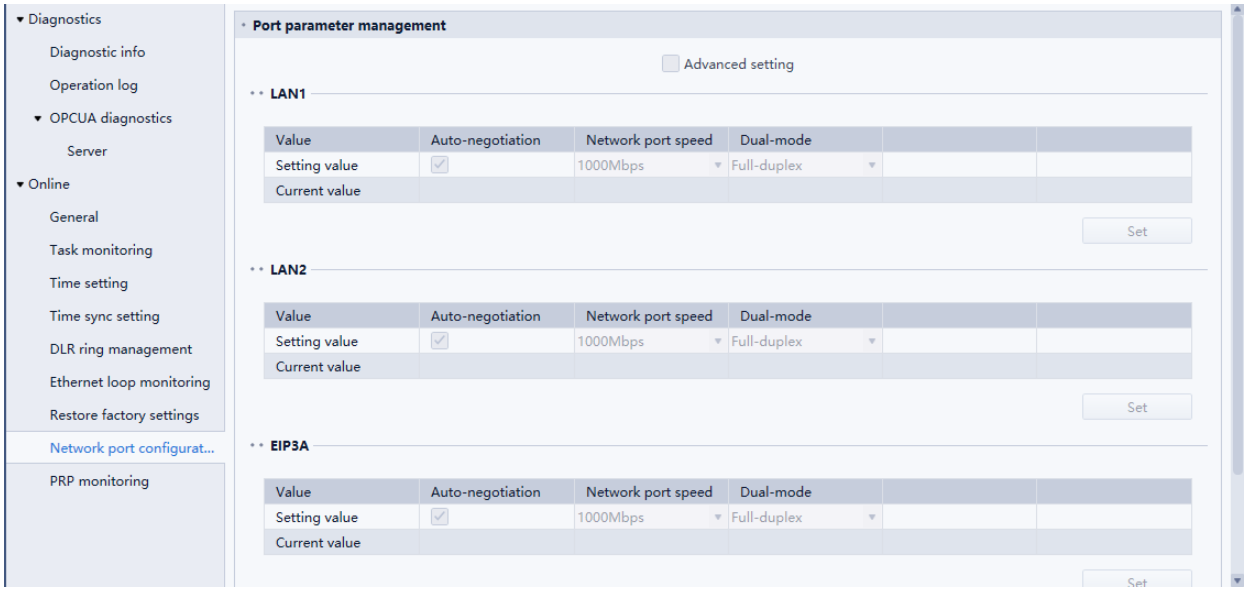
- EVO800 series: Firmware version V1.1 or above.
- EVO700C series: Firmware version V1.6 or above.
- APC760C series: Firmware version V1.6 or above.

Prerequisites

The device is in an online status.

Function entry

Double-click the "Online and diagnostics" node in the left project tree, and select the "Online > Network port configuration" tab.



This interface displays the configured values and current values of all Ethernet port parameters of the current PLC. The Ethernet port names are consistent with the actual PLC, making it easier for engineers to identify the Ethernet ports.

Select "Advanced settings" to display the configured values and current values of the "Wiring sequence configuration" and "Auto-deceleration" parameters.

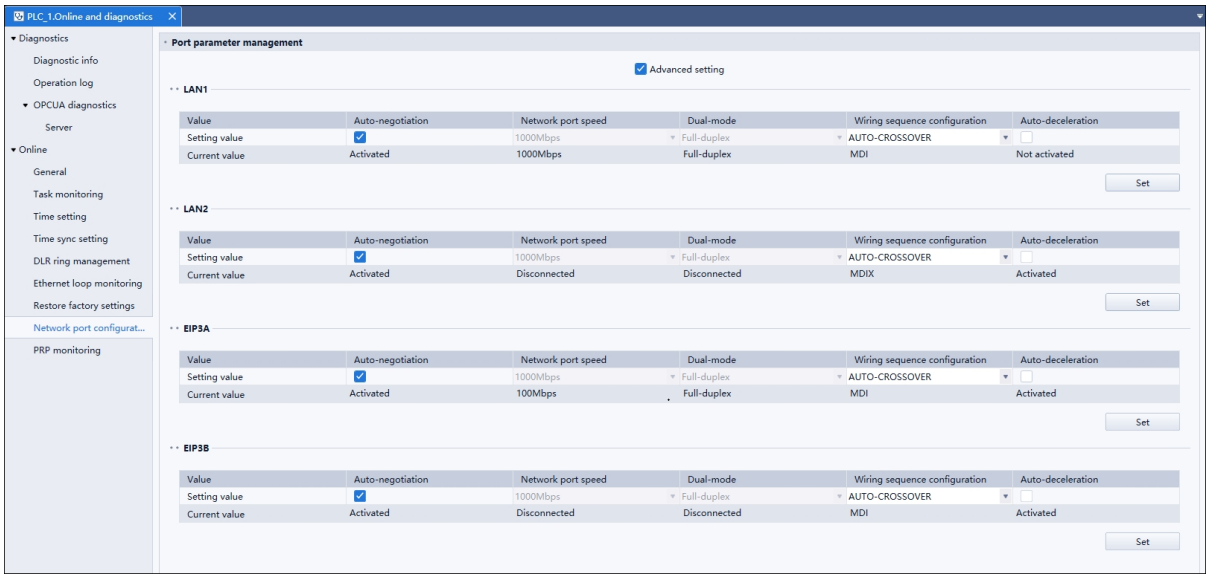


Figure 10-11-22-1 Network port configuration interface of EVO800 PLC

Table10-11-22-1 Network port parameter description

Parameter	Description
Auto-negotiation	Based on the IEEE 802.3u standard, this feature automatically negotiates the optimal speed and duplex mode with the peer device. This is enabled by default.
Port speed	Defines the physical transmission rate of the Ethernet port (bits per second). Supported rates include 10 Mbps, 100 Mbps, and 1000 Mbps (default).

Parameter	Description
Duplex mode	Controls the data transmission direction. Default is full duplex. • Full duplex: Simultaneous transmission and reception. • Half duplex: Alternative transmission and reception.
Wiring sequence configuration (MDI/MDI-X)	Defines the electrical connection method between the port and the transmission medium (such as twisted pair and fiber). Default is AUTO-CROSSOVER. • AUTO-CROSSOVER: The two connected devices negotiate to determine pin assignments (such as whether pins 1 and 2/3 and 6 are used for transmitting or receiving signals). • MDIX: Uses pins 1 and 2 for receiving signals, and pins 3 and 6 for transmitting signals. • MDI: Uses pins 1 and 2 for transmitting signals, and pins 3 and 6 for receiving signals.
Auto-deceleration	When the port repeatedly fails to connect after a certain number of attempts, the PHY (Physical layer) chip can automatically reduce the speed to complete the connection. This feature is enabled by default.

Procedure

1. Open the "Network port configuration" interface and modify setting values of basic network port parameters. After selecting "Advanced settings", wiring sequence and auto-deceleration can be configured.

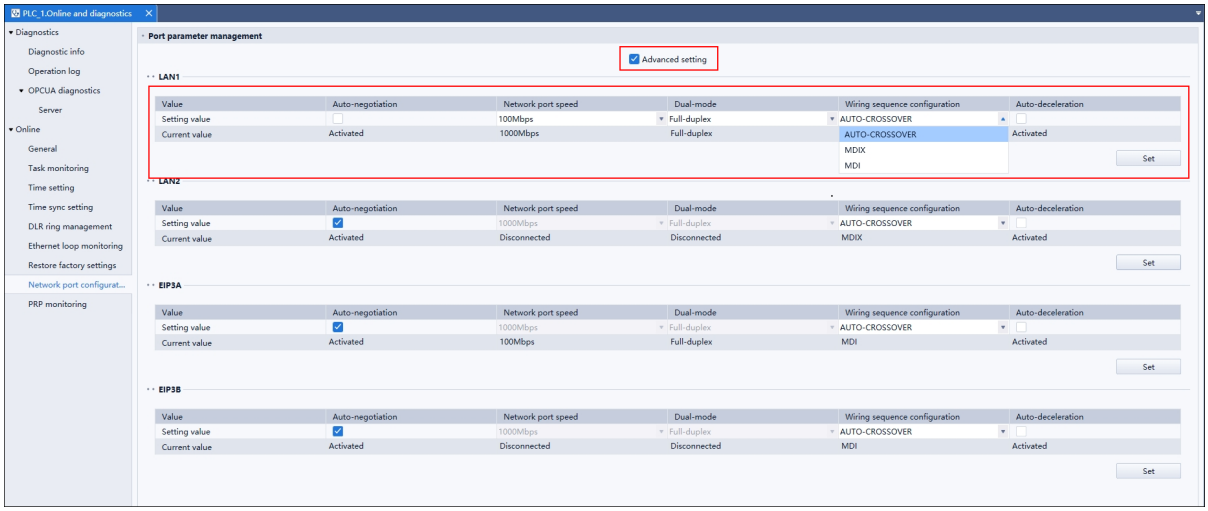


Figure 10-11-22-1 Network port configuration interface of EVO800 PLC

2. After completing the parameter settings, click the "Set" button. In the pop-up prompt window, click "OK" to send the configuration instruction to the PLC and complete the parameter modification of the corresponding port.
3. After modification, the corresponding PLC port will re-negotiate according to the set parameters and display the current set parameters in real time.

NOTE

- The current values in the "Network port configuration" tab display the negotiation results between the PLC port and the connected device, which are related to the peer device configuration, cable material, PHY characteristics, and other factors.
- If a specific network port on the current device does not support network port configuration, the current value will not be displayed for that port. Clicking the "Set" button will display an "Execution failed" prompt.

CAUTION

- Modifying Ethernet port parameters will cause the port to undergo a link down and link up process. This will inevitably impact current communication services, manifesting as temporary interruption and subsequent recovery of all communication connections on this port.
- If the configured port is currently being used for monitoring, modifying its parameters may cause the device to go offline.

10.11.23 PRP Monitoring

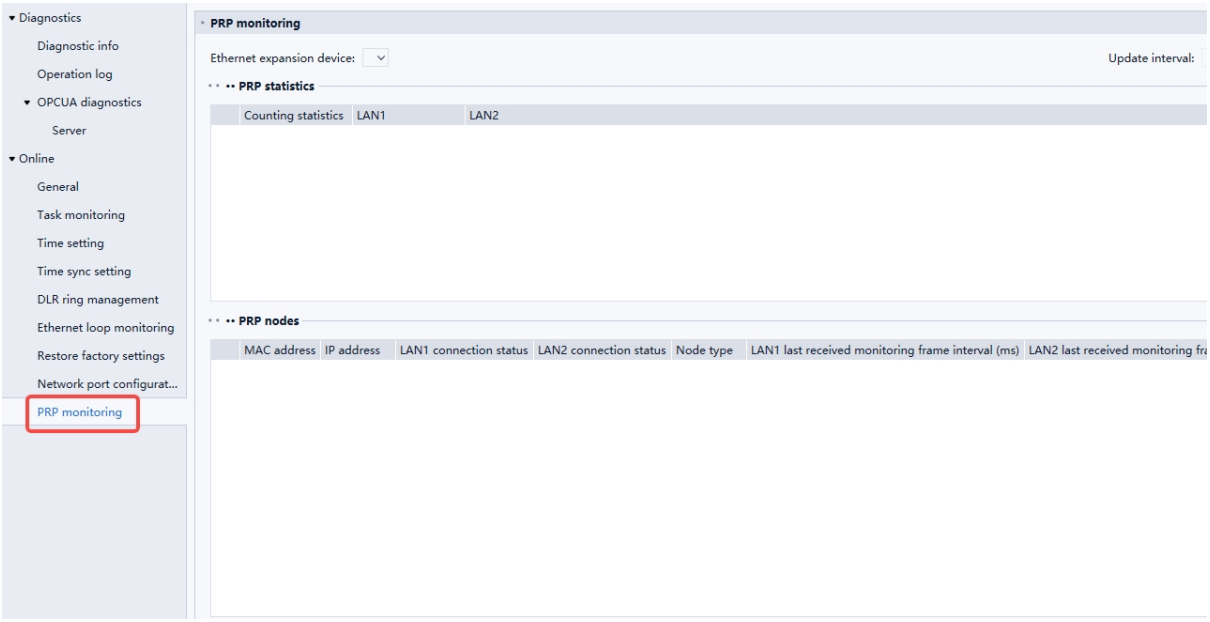
After the EVO800 series Ethernet expansion module activates the PRP function and the PLC goes online, PRP information can be monitored online.

Prerequisites

The device is in an online status.

Procedure

1. Double-click "Online and diagnostics" in the project tree, select "Online > PRP" in the opened interface to enter the PRP monitoring interface.
   When PRP is activated, the default data update interval between iFA Evolution and the PLC is 3s. The status information of all nodes in the network is automatically refreshed. The update interval can be set from 1s to 10s.



2. Select the device to monitor from the "Ethernet expansion device" drop-down list and set the update interval.
3. View PRP statistical information. The related parameters are shown in the table below.

Parameter	Description
Transmitted package count	Number of packets transmitted via LAN1/2.
Received package count	Number of packets received via LAN1/2.
Unique received frame count	Number of unique PRP frames received by the port.
Duplicated received frame count	Number of duplicate PRP frames received by the port.
Incorrect port count	Number of packets received by the port that do not belong to that port. For example, LAN1 receives a packet with tag lan_id = LANB (which corresponds to LAN2).
Alarm count	Number of PRP faults/alarms.
Warning occurs?	Indicates whether a PRP fault/alarm has occurred, including events such as no PRP frame received or an incorrect PRP suffix, etc. ◦ 0: No fault/alarm has occurred. ◦ 1: A fault/alarm has occurred.

4. View PRP node information. The related parameters are shown in the table below.

Parameter	Description
MAC address	The MAC address of the peer PRP node (a PRP node has only one MAC address externally).
IP address	The IP address of the peer PRP node (a PRP node has only one IP address externally).

Parameter	Description
LAN1 connection status	The status indicating whether the LAN1 or LAN2 port has received PRP packets. ◦ True: The port has received PRP packets. ◦ False: The port has not received PRP packets.
LAN2 connection status	
Node type	PRP node type. ◦ Dual-node connection: The node device supports PRP mode. ◦ Virtual dual-node connection: The node device does not support PRP mode and connects via a PRP management switch.
LAN1 last received monitoring frame interval (ms)	The interval between the received monitoring frame and the previous frame.
LAN2 last received monitoring frame interval (ms)	

10.11.24 Uploading/Downloading Permission Settings (EVO500)

The permission settings function is used to manage the password for external access (upload/download) to the physical controller.

NOTE

This function is only applicable to the EVO500 series with firmware version V1.2.

Prerequisites

- The physical controller is online.
- The PLC project is online.

Adding a password

1. Right-click a PLC project node in the project tree and select "Permission setting" to open the "Upload/Download password setting" window.
2. Set the permission password.
   Upload/Download password: When checked, the password management permission is activated; when unchecked, the corresponding permission is revoked.
   Enter/Confirm password: The password for accessing the controller. Supports letters, numbers, and special characters, with a length between 6 and 8 characters.
3. Click "OK".
   The permission and password settings for managing controller access are configured and active. Password verification is required during the execution of download or upload commands.

Canceling password verification

1. Right-click a PLC project node in the project tree and select "Permission setting" to open the "Upload/Download password setting" window.
2. Cancel permission settings
   Uncheck "Upload/Download password", and no password will be required when accessing the controller. This setting takes effect immediately.

Changing password

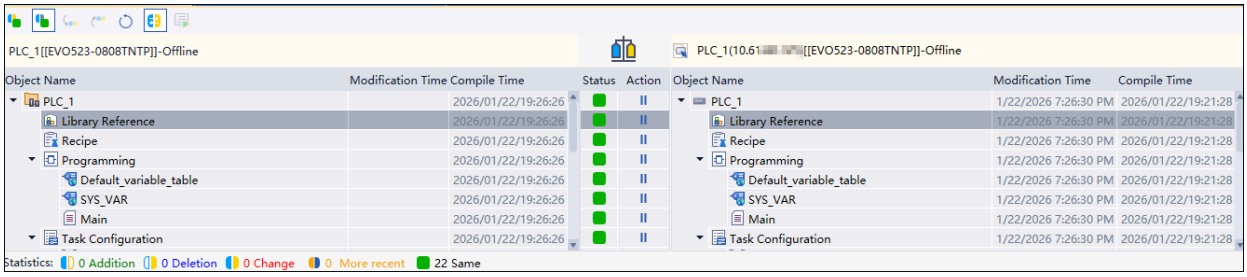
1. Right-click a PLC project node in the project tree and select "Permission setting" to open the "Upload/Download password setting" window.
2. Change password
   Click the "Change password" button and set the new password in the pop-up window. The new password takes effect immediately.

10.12 PLC Project Comparison

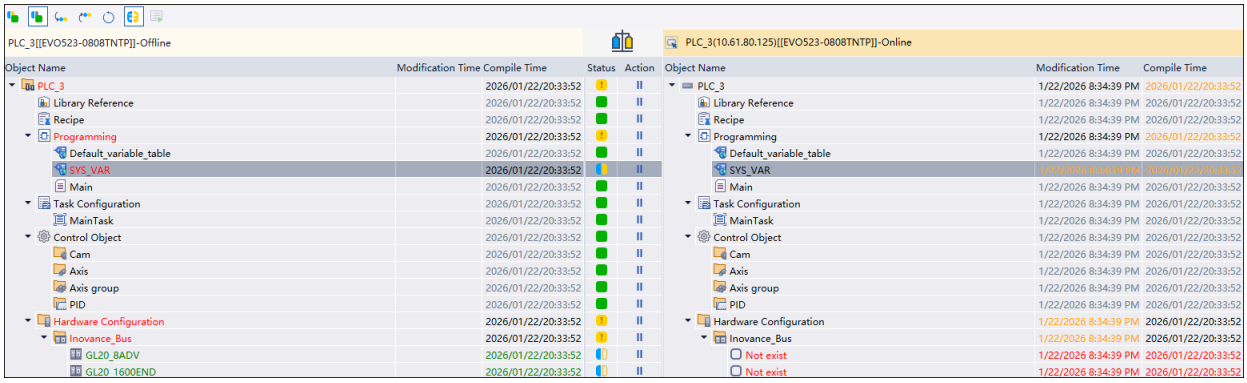
10.12.1 Overview

PLC project comparison is a comparison of equipment content and coverage, helping users quickly locate differences between two project contents. This feature supports comparison between offline projects, and between online and offline projects.

Offline/offline comparison



Offline/online comparison



The comparison editor includes toolbar, project selection panel, left comparison table, right comparison table, status area, action area, and comparison result area.

Toolbar

Icon	Description	Description
	Shows differences in objects.	Switches to detailed comparison view.
	Shows only different comparison results.	Shows only different comparison results.
	Shows all comparison results.	Shows all comparison results, including identical and different items.
	Next difference	Selects the next different object.
	Previous difference	Selects the previous different object.
	Refresh	Refreshes the view.
	Synchronizes scrolling	Enables/disables synchronized scrolling between left and right comparison tables.
	Executes action	Executes the selected action item in the action area of the comparison view.

Status area

Icon	Description
	Contains differing sub-items.
	The project nodes on both sides are identical.
	The project nodes on both sides are different.
	The project node on the left side does not exist on the right side.
	The project node on the right side does not exist on the left side.

Action area

Icon	Action description
	No action.
	Copies the right project information to the left.
	Copies the left project information to the right. This action is supported only in offline/offline comparison.
	Downloads all left project information to the PLC. This action is supported only in offline/online comparison.

Comparison result area

Icon	Description
	Additional items in the left project information
	Missing items in the left project information
	Modified items in both sides
	More recent items
	Identical items in both sides

10.12.2 Offline/Offline Comparison

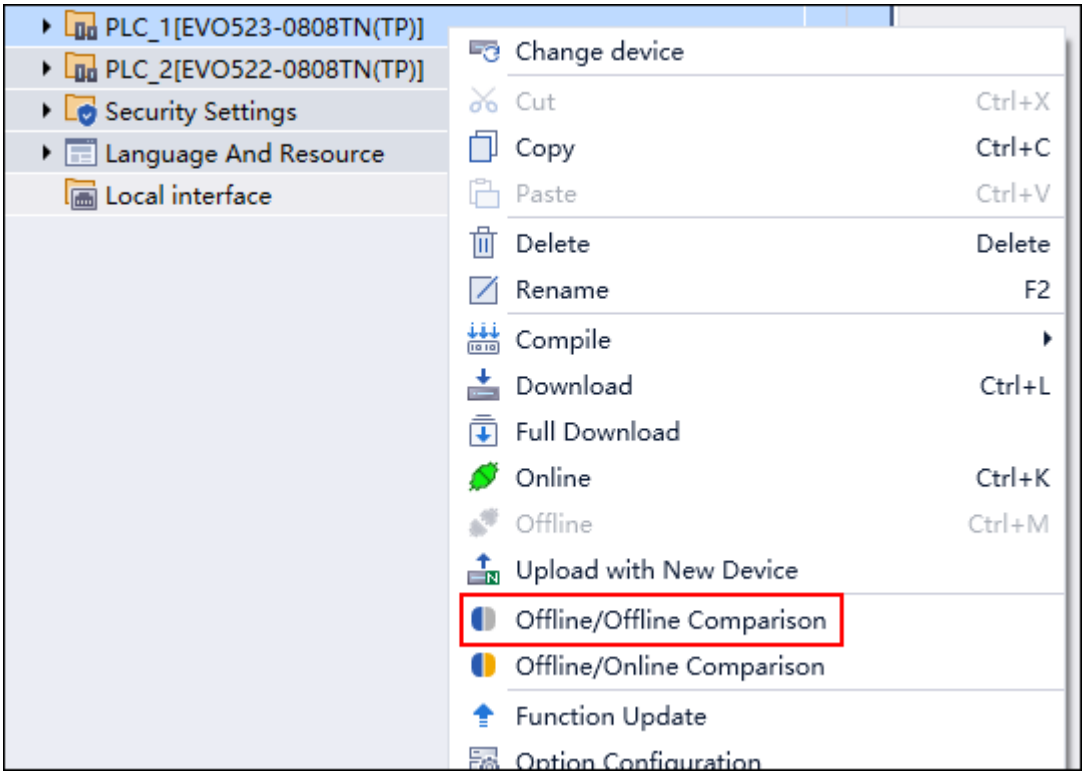
Offline/offline comparison is suitable for comparing offline PLC projects with the same product series.

Prerequisites

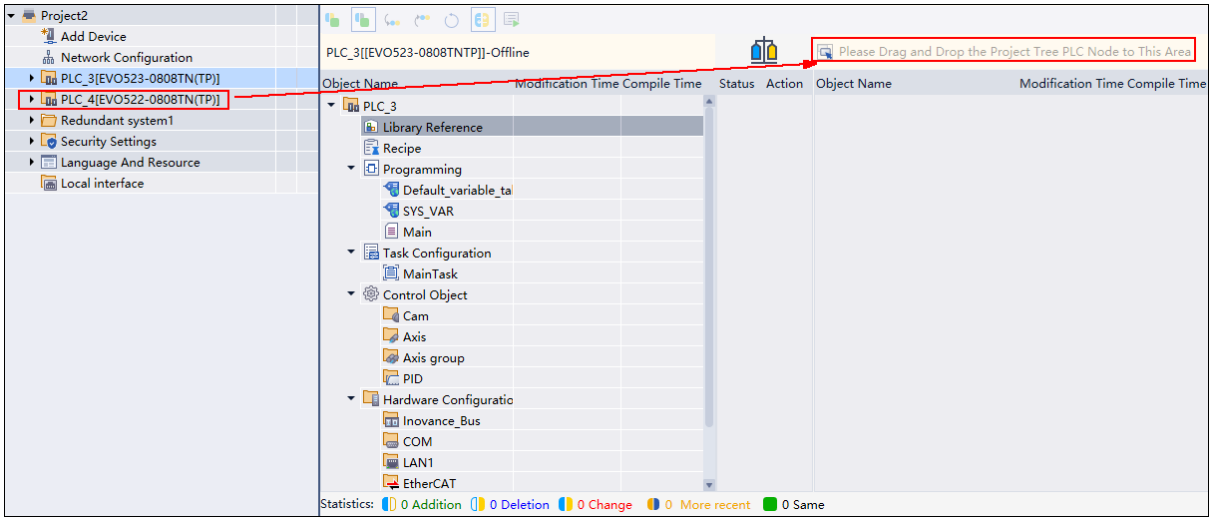
- The project is in offline mode.
- At least two PLC project nodes exist in the project tree.
- Comparison is supported between PLC projects of the same product series or different products.

Procedure

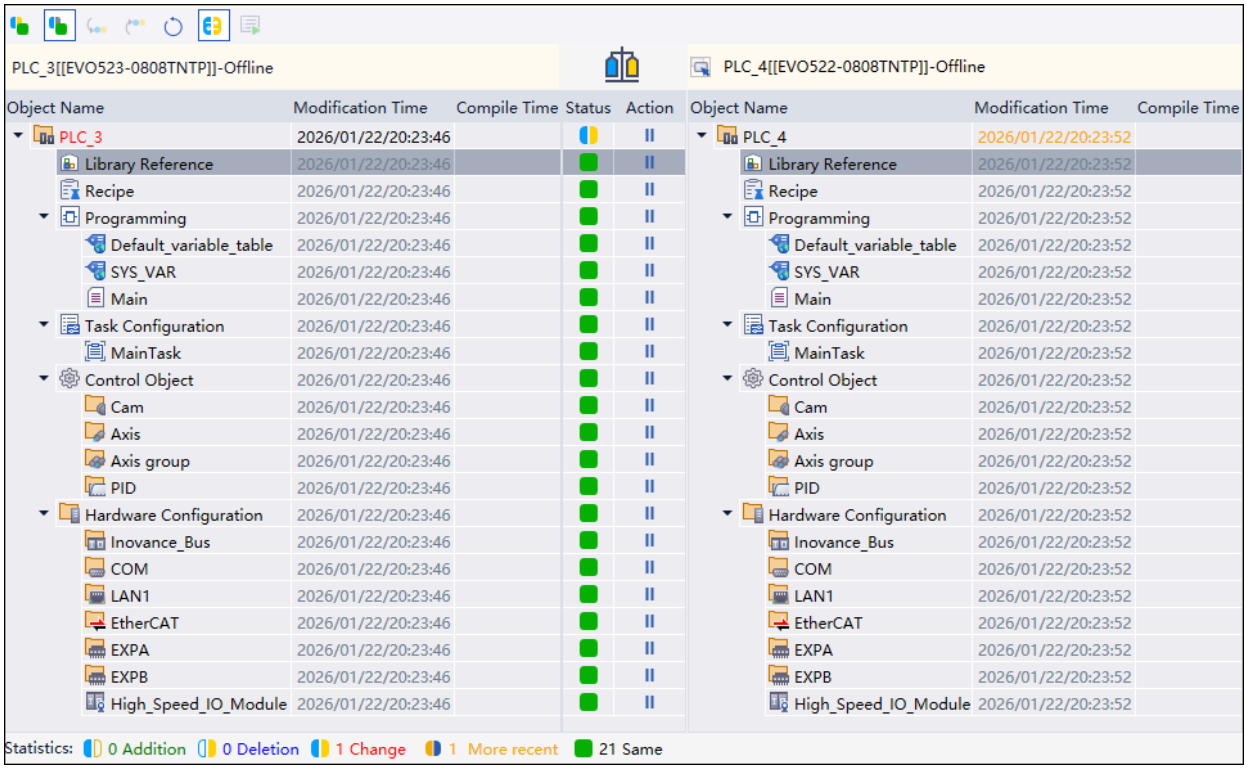
1. Right-click a PLC node in the project tree and select "Offline/offline comparison" to enter the comparison interface.



2. Drag the project to be compared into the right area.



Function view



10.12.3 Offline/Online Comparison

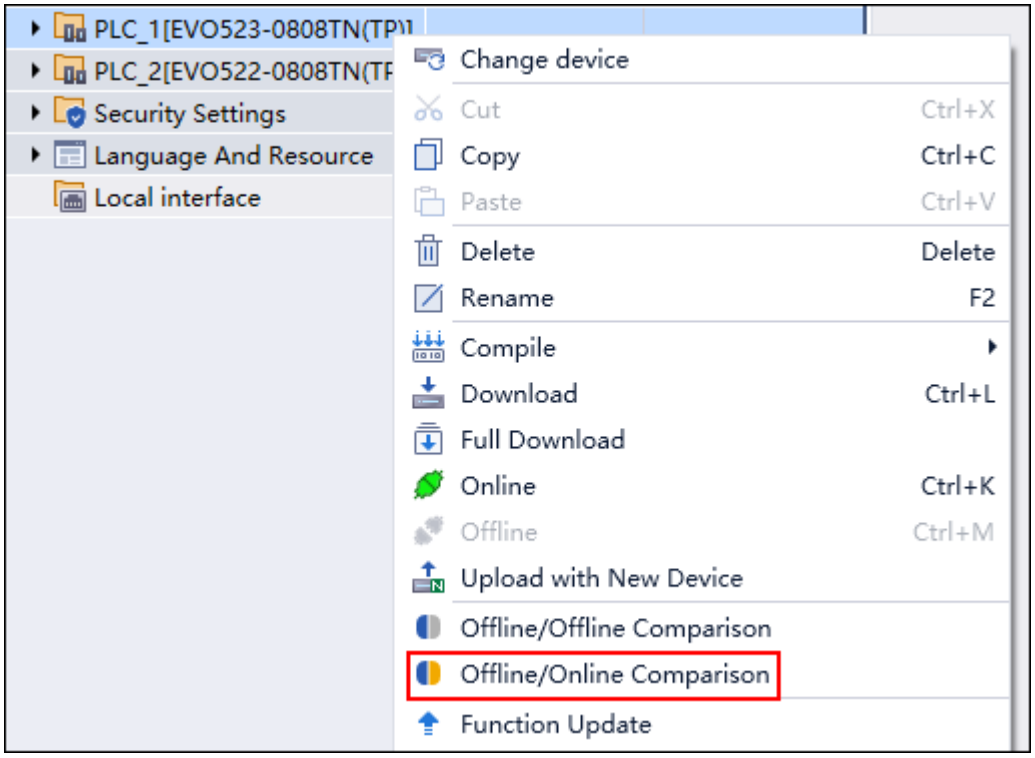
Offline/online comparison is suitable for comparing offline and online PLC projects with the same product series.

Prerequisites

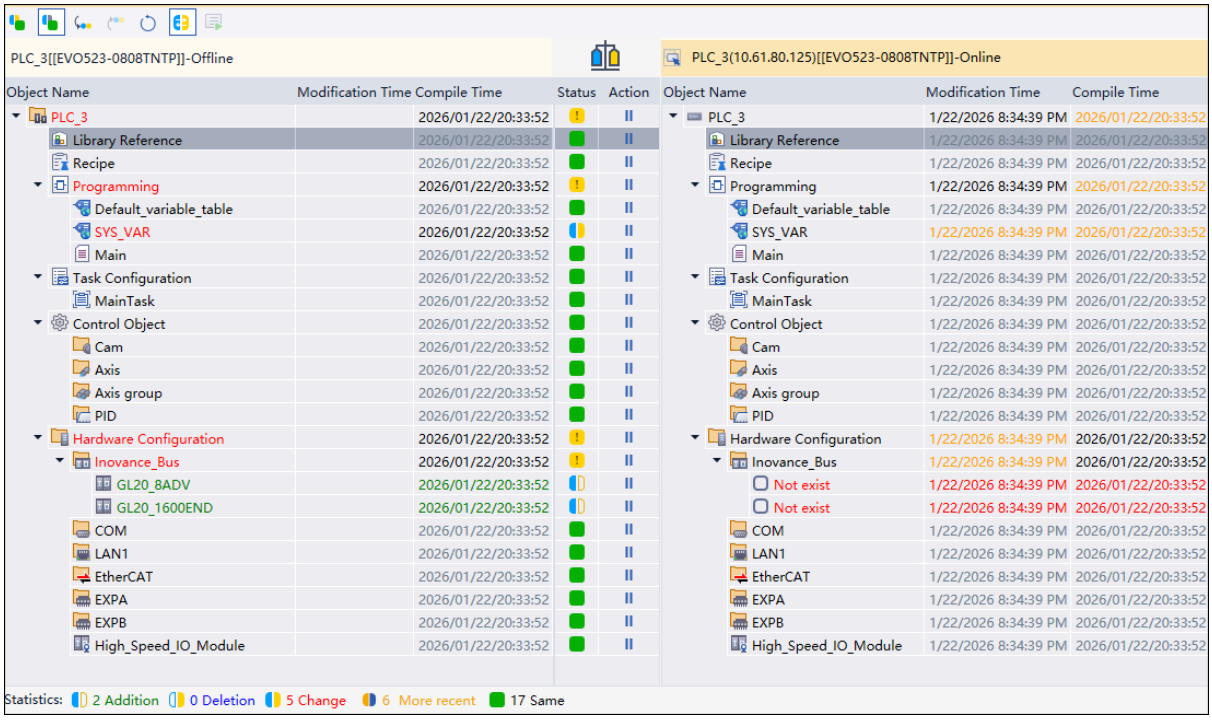
- At least one PLC project node exists in the project tree.
- The PLC project can successfully connect to the PLC, or this project is already in online state.
- Comparison is only supported between PLC projects of the same product series.

Procedure

1. Right-click a PLC node in the project tree and select "Offline/online comparison".



2. Enter the comparison interface and perform object comparison.



10.12.4 Viewing and Processing Comparison Results

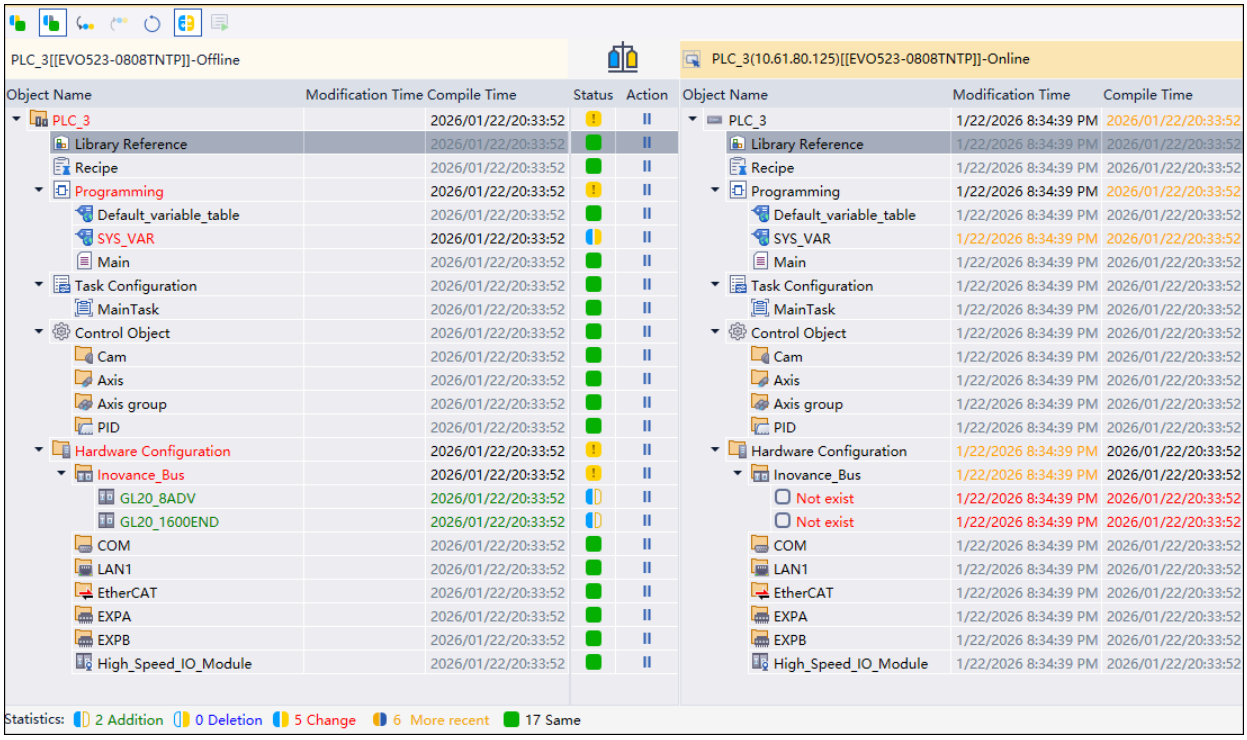
This function is mainly used to view and process the comparison objects and comparison blocks.

Comparison object

The comparison results for various program objects can be viewed and processed.

Viewing method

Right-click a PLC node in the project tree and select "Offline/offline comparison" or "Offline/online comparison" to open the comparison view and directly view the comparison results. Take offline/online comparison as an example.

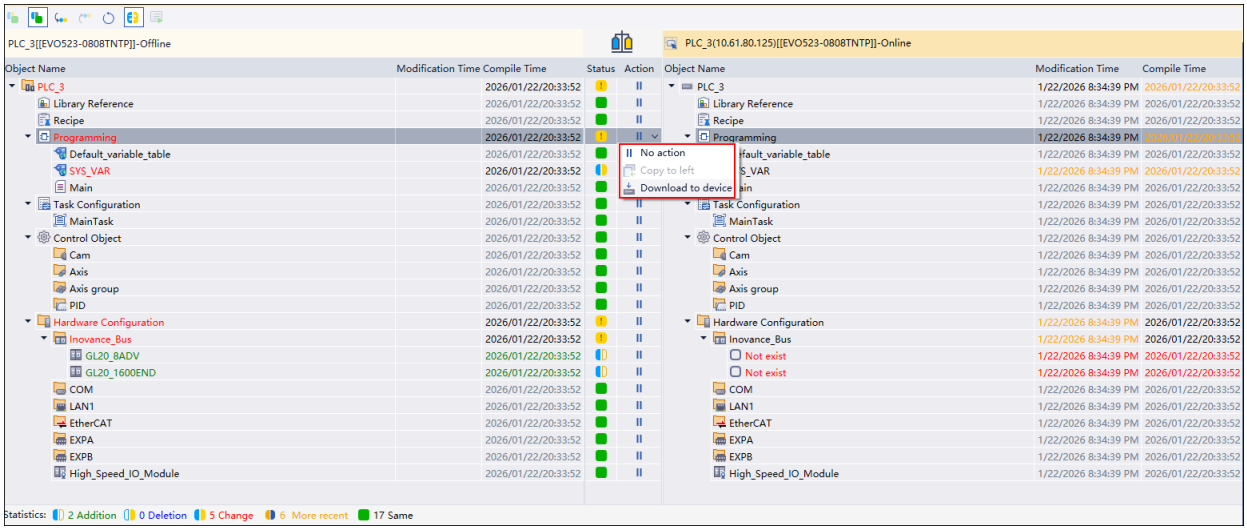


For details about the toolbar, status area, action area, and result area, see 10.12.1 Overview.

Processing method

Select the objects with differences, and click an action to perform the corresponding action.

Offline/offline comparison results cannot be downloaded to the device. For details about actions, see 10.12.1 Overview.



NOTE

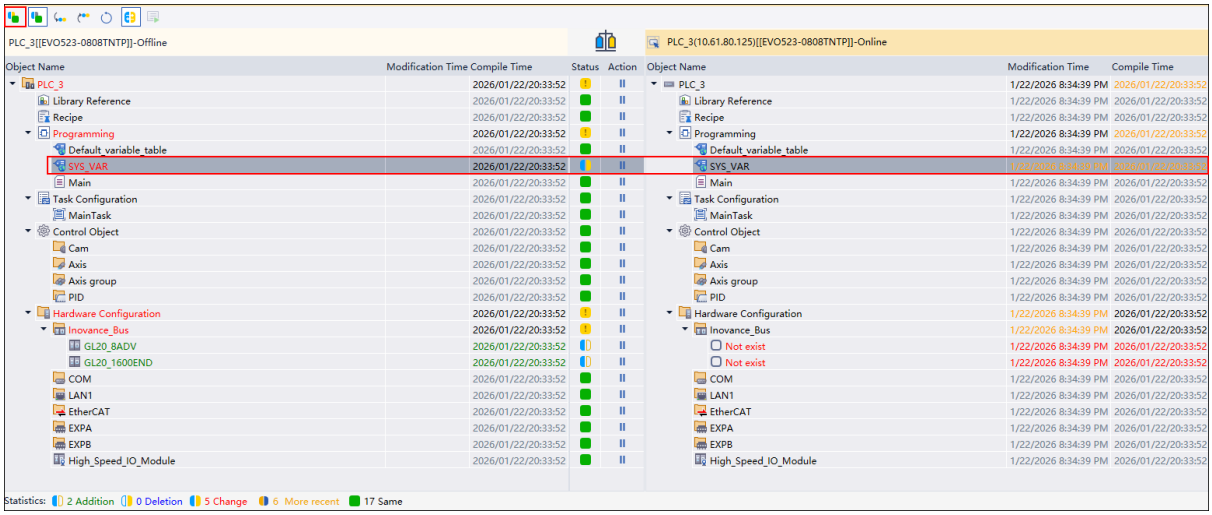
The selected action can be applied to an individual node or the entire branch. When selecting a higher-level branch, the selected action will be synchronized for all its sub-branches.

Comparison blocks

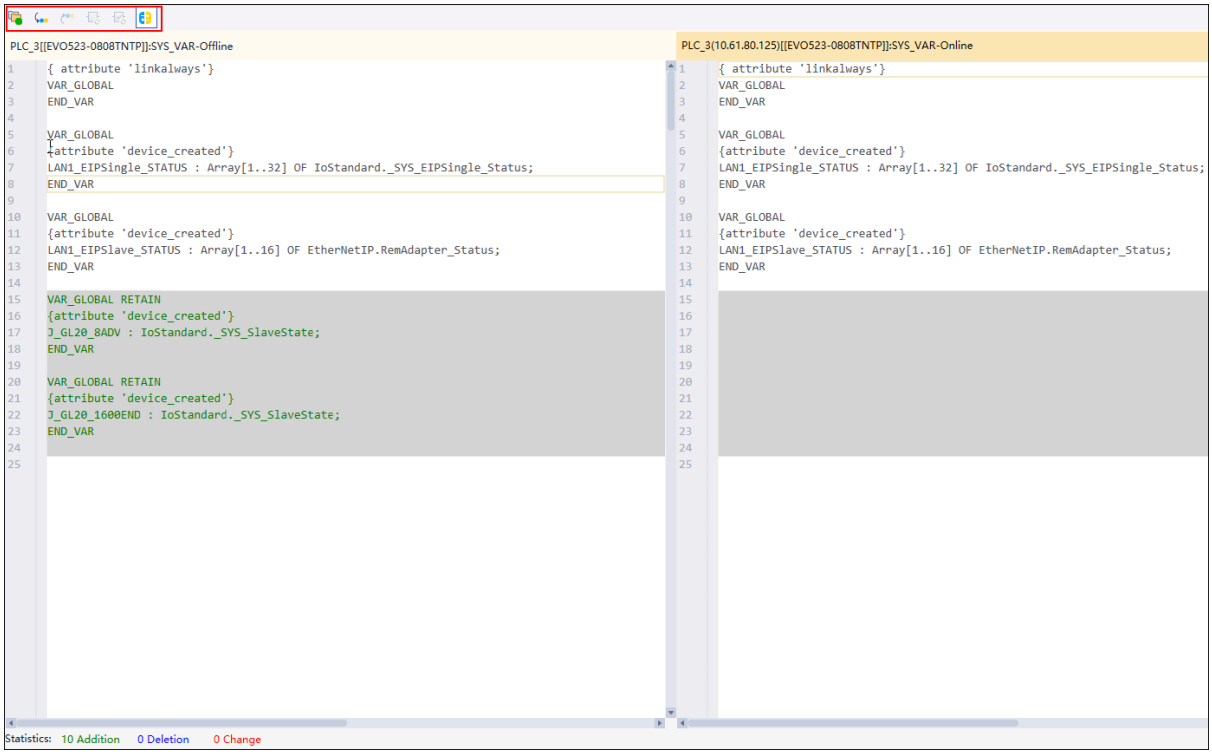
Project comparison supports detailed views of comparison objects, allowing viewing and processing results for each comparison block.

Procedure

1. Right-click a PLC node in the project tree and select "Offline/offline comparison" or "Offline/online comparison" to open the comparison view.
2. Click the "Shows differences in objects." icon in the toolbar or double-click the comparison object to open the detailed view of each comparison object



3. Modify the comparison block using the icons in the toolbar.



Descriptions of the toolbar

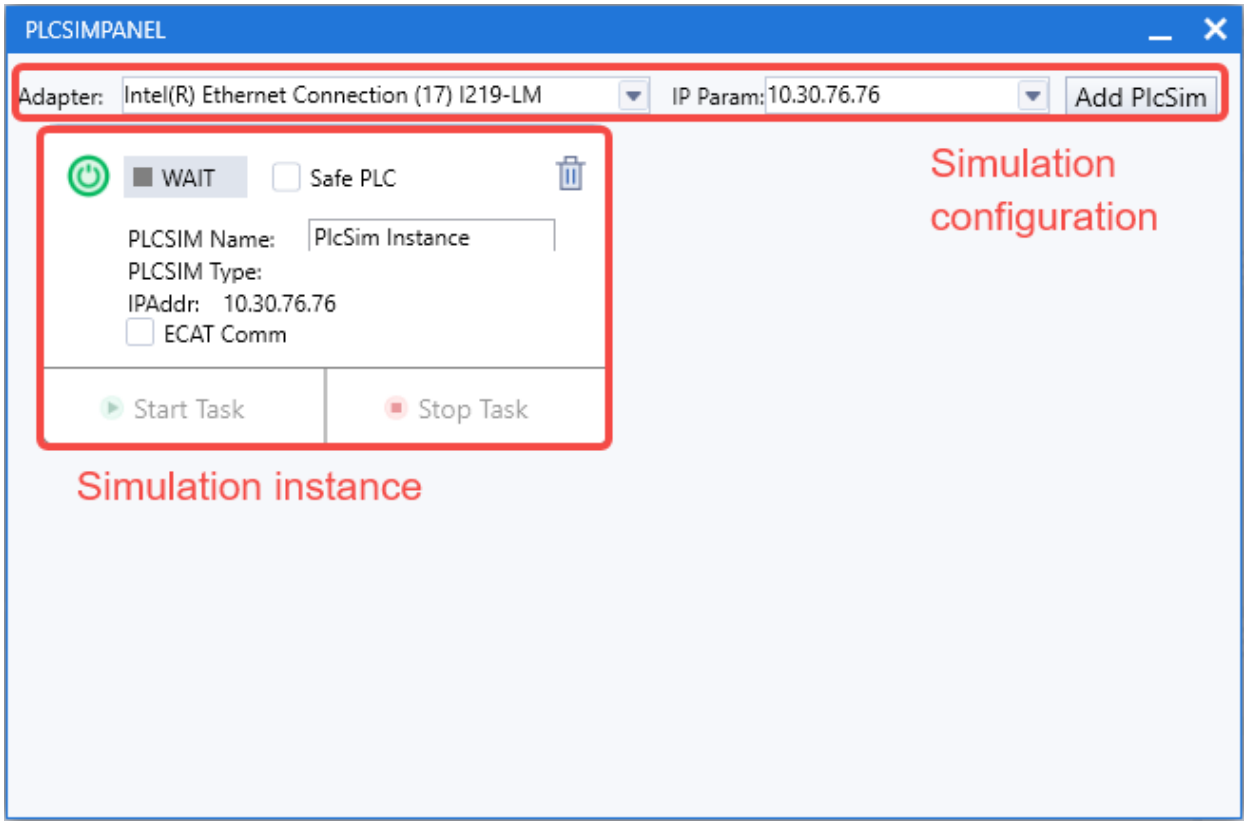
Icon	Description	Description
	Shows differences in the projects.	Switches to the detailed view of project comparison.
	Next difference	Selects the next different object.
	Previous difference	Selects the previous different object.
	Synchronize block	Synchronizes all different blocks.

Icon	Description	Description
	Synchronize selected	Synchronizes the selected different blocks.
	Synchronizes scrolling	Enables/disables synchronized scrolling between left and right comparison blocks; this icon remains synchronized with the "Synchronize scrolling" icon in the comparison overview view.

10.13 PLC Simulation Mode

10.13.1 Simulation Panel Overview

Click in the iFA toolbar to open the simulation panel. The simulation panel consists of two parts: the simulation configuration area and the simulation instance management area, as shown in the figure below:



- Simulation configuration area
  Select the physical network card and IP address to establish the simulation network connection. Click "Add simulation" to create a simulation instance.
- Simulation instance management area
  Manage simulation instances, control simulation execution, and view simulation status.

Function	Description
Start/Stop	Starts and stops the virtual PLC. Start: Click and the virtual PLC enters the Ready state. The PLC can be used normally at this time. Users can perform scanning, connecting, downloading, and online operations via the iFA software. Stop: Click to stop the PLC simulation. All connections to iFA are disconnected, and all ongoing simulation operations are terminated.
Status indicator	Indicates the current status of the simulated PLC. WAIT: Wait state. After the virtual PLC is created, the indicator is in the wait state. It indicates that it has not been started and is waiting for users to power it on. READY: Ready state. After the virtual PLC is powered on, the indicator changes to the ready state. It indicates that it is ready and users can download the PLC program for simulation. STOP: Stop state. After the PLC program is compiled and downloaded, the indicator is in stop state. It indicates that the program is loaded but not running. Users can click "Start task" to begin simulation. START: Run state. When the PLC program is running, the indicator is in the run state. It indicates that simulation is in progress.
Safety PLC	Enables the Safety PLC. This option is unchecked by default.
Delete	Deletes the virtual PLC instance object. When the current simulation object is no longer needed, this icon can be used to delete the object.
PLCSIM name	User-defined name of the virtual PLC simulation.
PLCSIM simulation model	Displays the model of the current simulation. After the virtual PLC simulation is successfully created, the simulation model will be confirmed by the system and displayed in the simulation object.
IP address	Displays the IP address bound to the simulation upon creation.
ECAT communication	Check to enable ECAT communication. Disabled by default. During ECAT axis simulation, both real axes and virtual axes are simulated. If ECAT communication is not enabled, all axes in the iFA project will be simulated in virtual axis mode. If ECAT communication is enabled, axes will be simulated according to their actual configuration in the iFA project.
Starts task	After a virtual PLC instance object is created, the PLC logic program is built through software. Once the iFA program is completed, it is downloaded to the virtual PLC object, and this button can be used to start the PLC logic program. If an invalid PLC logic program is downloaded to the current virtual PLC or if the PLC logic program is already running, this button will be disabled and operation is prohibited.
Stop task	After a virtual PLC instance object is created, the PLC logic program is built through software. Once the iFA program is completed, it is downloaded to the virtual PLC object, and this button can be used to pause the PLC logic program. If an invalid PLC logic program is downloaded to the current virtual PLC or if the PLC logic program is already in a stopped state, this button will be disabled.

10.13.2 Establishing a Simulation Environment

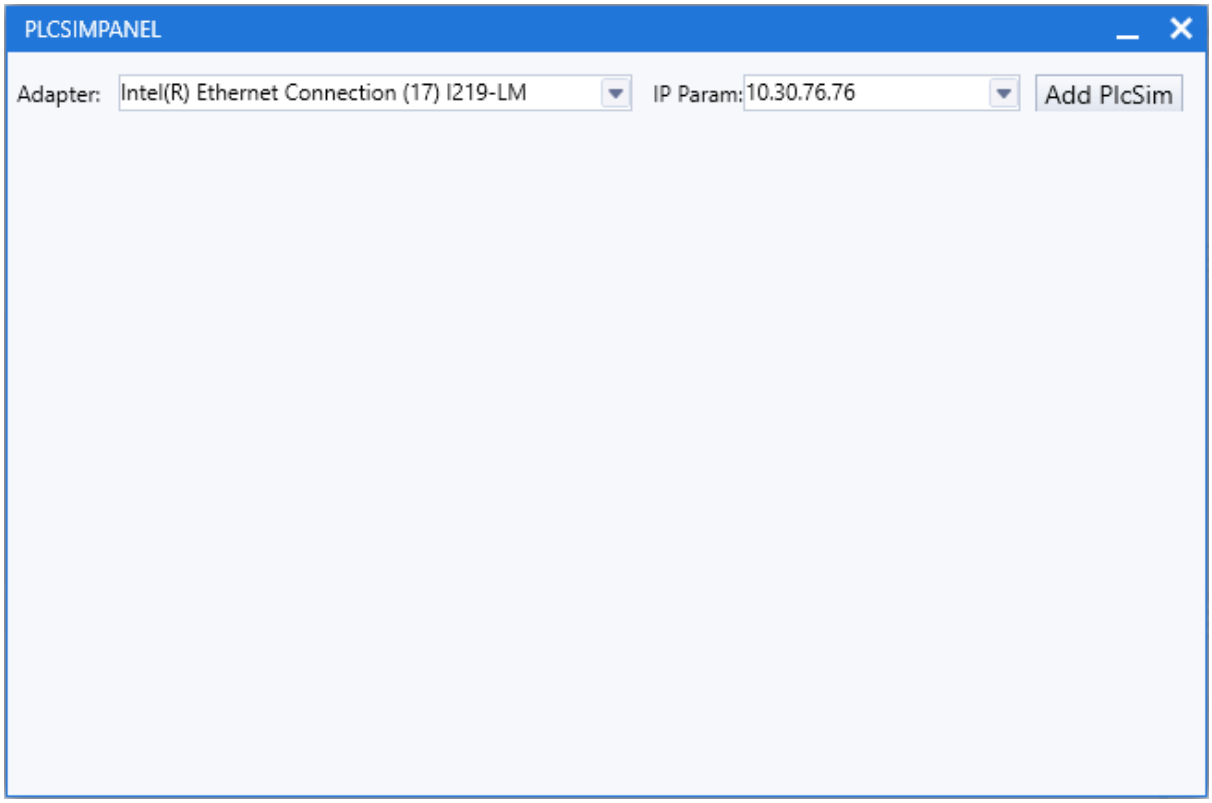
PLCSIM virtual PLC simulation relies on iFA integrated software and must be started by iFA. The simulation environment must be set up before performing simulation operations.

Prerequisites

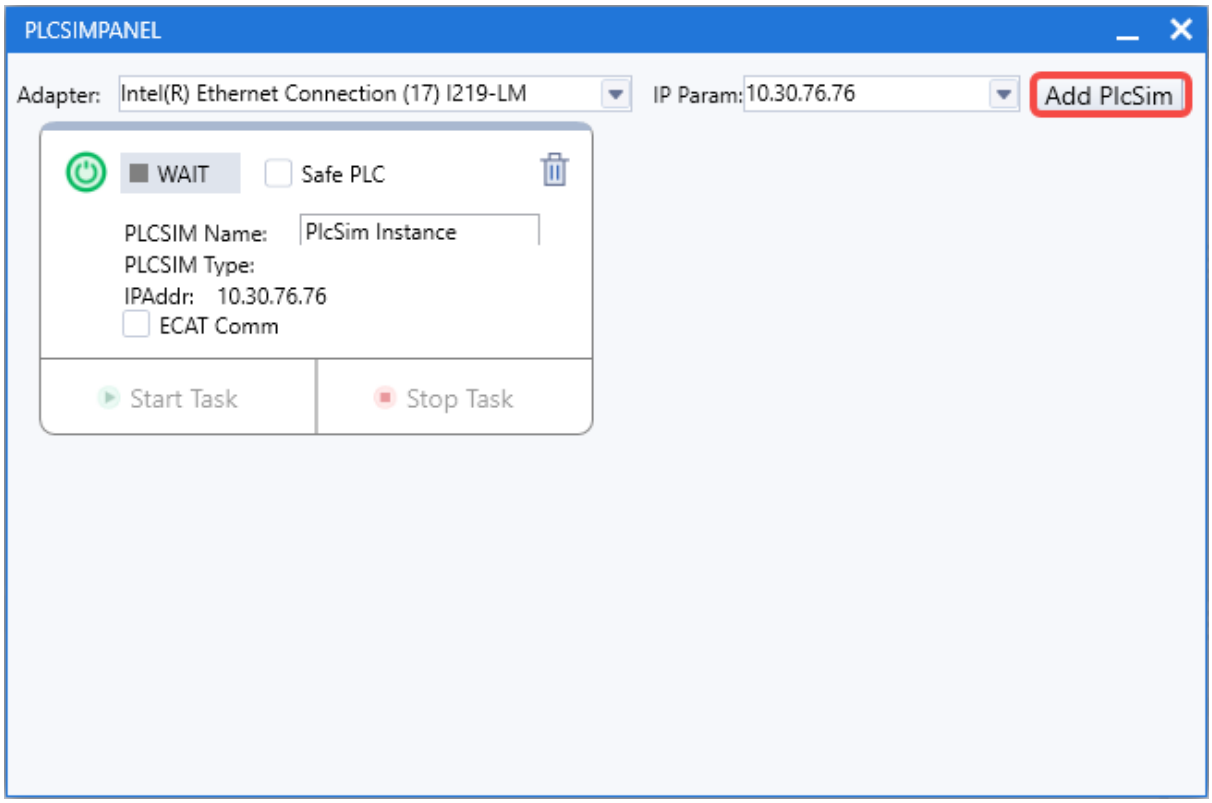
- A project has already been created in iFA, and a PLC device has been added.
- Configure the PLC logic execution program in the program editing interface.

Procedure

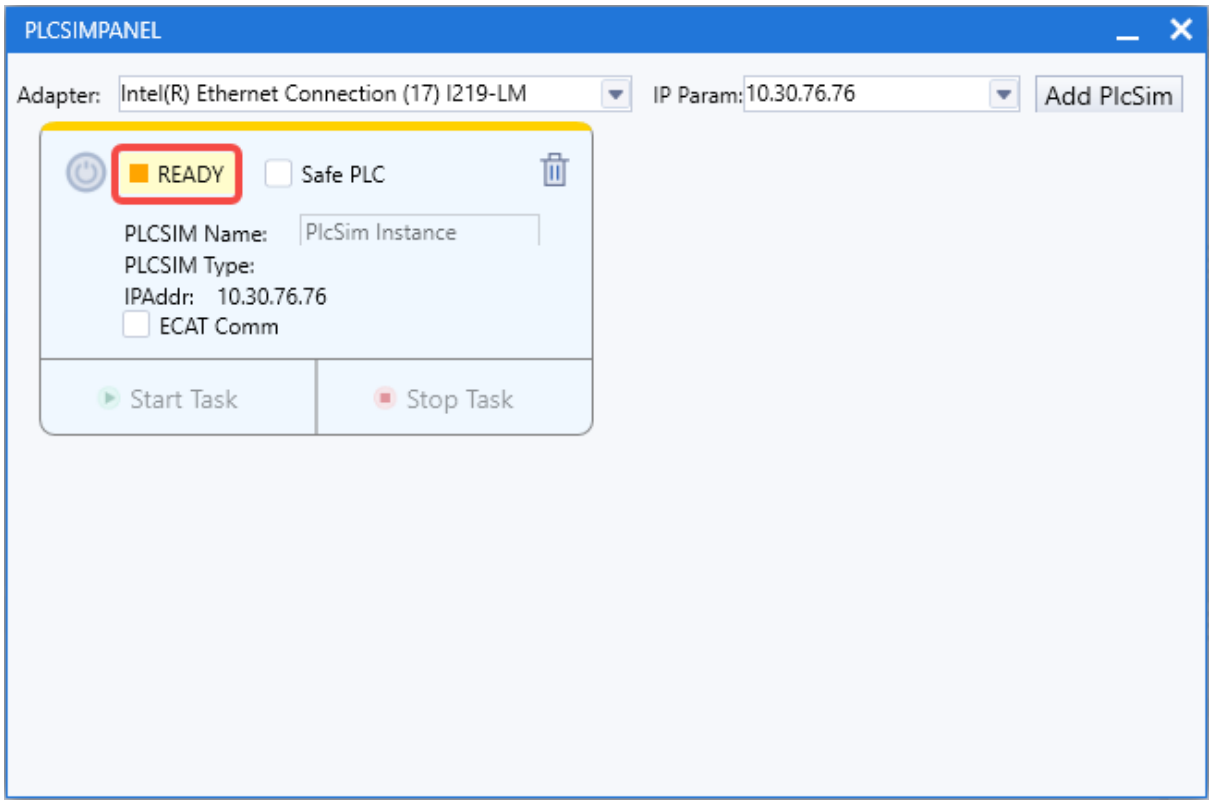
1. Click "Start simulation" on the toolbar to open the simulation panel.



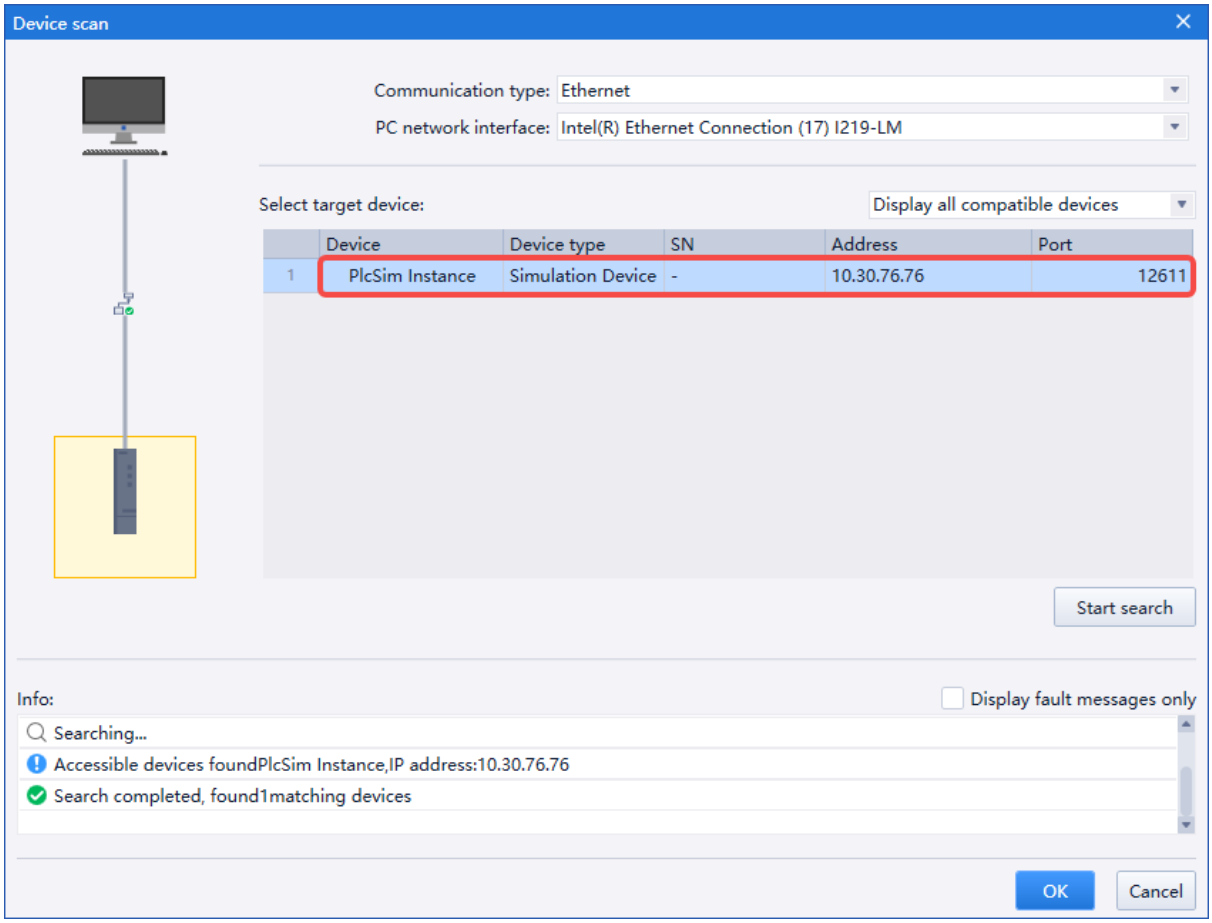
2. Click "Add simulation" to create a virtual PLC simulation instance object.



3. Click and the virtual PLC enters the Ready state. The PLC can be used normally at this time.

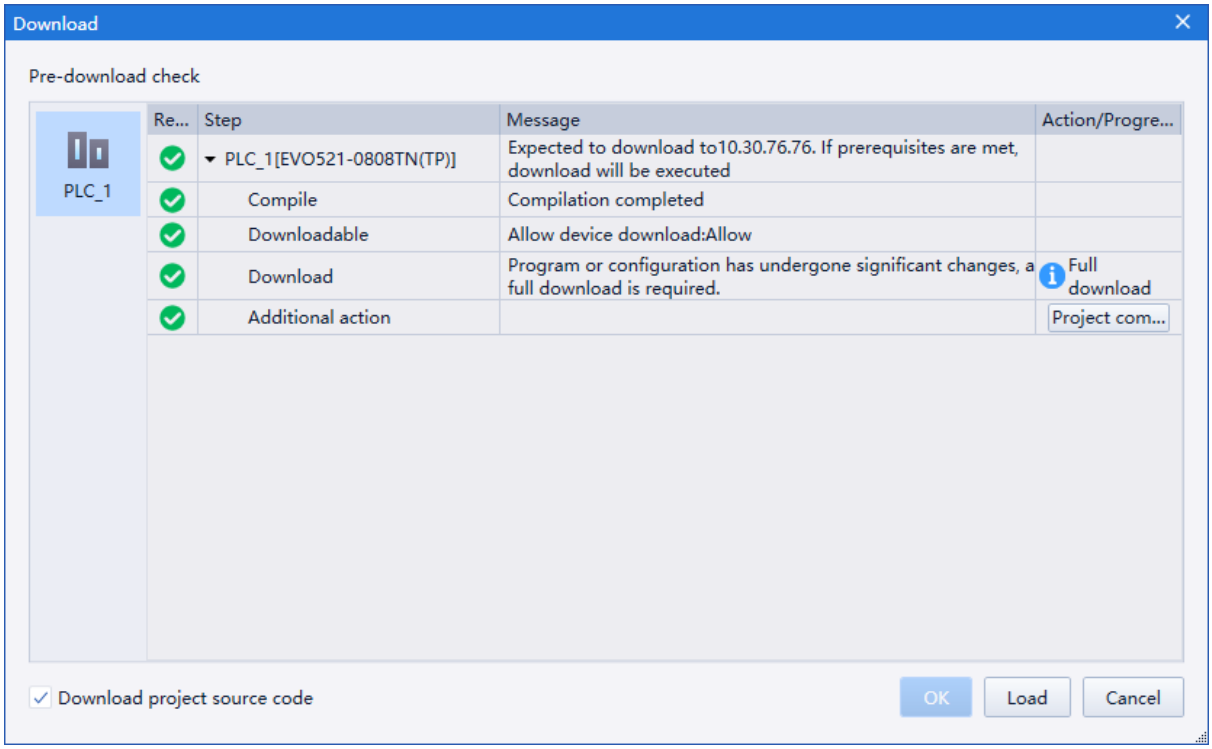


4. Select "Online > Device scan" from the menu bar. In the "Device scanning" window, click "Start search", select the simulation device of the virtual PLC, and click "OK".



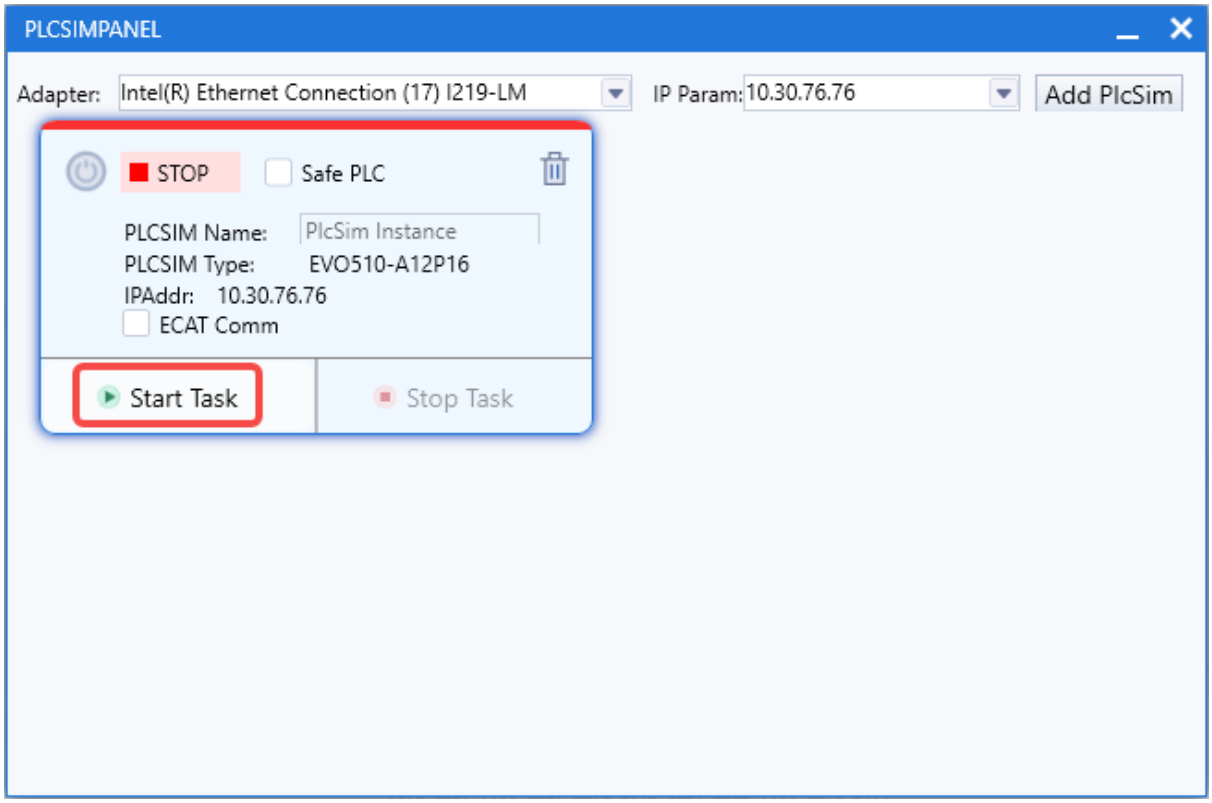
5. Click the "Compile" icon on the toolbar to compile the PLC logic program.

6. Click the "Download" icon on the toolbar. In the "Download" window, click "Load" to download the successfully compiled program to the virtual PLC.



7. Click the "Online" icon on the toolbar to go online.

8. Click "Start task" in the simulation panel to execute the program in the virtual PLC and start logic simulation.



10.13.3 Simulation Function Description

Supported simulation features

The main functions of PLCSIM are embodied in the virtual PLC instance object, specifically including the following functions:

Simulation Functions	Description
Logic and program simulation	The virtual PLC object can simulate the logical ST programs and LD programs of the PLC, and supports software algorithm library instructions, PID algorithm instructions, axis motion control instructions, and compiler algorithm instructions. It also supports logical power-off commissioning for ST programs, power failure retention, as well as simulation of oscilloscope and recipe functions.

Simulation Functions	Description	
Communication simulation	EIP simulation	EIP protocol supports the virtual PLC to act as a scanner and adapter for communication simulation. The virtual PLC can work as an adapter slave or a scanner master. When the virtual PLC acts as an adapter slave device, it supports Class1 implicit I/O connections and label connections, enabling data exchange access for I/O; it also supports Class3 and UCMM explicit communication with label access, allowing the scanner to access public label variables within the virtual PLC. When the virtual PLC acts as a scanner master, it supports importing EDS device files into the scanner and adding slave device connections or label connections, and building explicit Class3 and UCMM service label communication to access public labels variables on the adapter side.
	Modbus simulation	The virtual PLC supports simulation of Modbus protocol as both master and slave. The virtual PLC simulation object can operate independently as a master or slave, or simultaneously as both. When simulating as a Modbus master or slave, it supports standard Modbus function codes, allowing read and write operations on memory areas and status areas.
	OPC UA simulation	The virtual PLC can simulate an OPC UA protocol service device. During OPC UA simulation, the virtual PLC currently only supports label access from the OPC UA clients, enabling read and write operations on public network variables in the virtual PLC program.
	EtherCAT simulation	The virtual PLC supports simulation of an EtherCAT as a master. Slaves can be added through the EtherCAT master. Currently, only servo EtherCAT slaves are supported, while I/O is not supported. It can drive the motion of servo EtherCAT slave physical axes via EtherCAT, and the current servo master supports motion control simulation for up to eight axes with a 10 ms cycle.

Simulation Functions	Description
Axis simulation	The virtual PLC axis simulation supports two modes: real axis and virtual axis. Real axis: Simulates the motion control for physical servo axes of devices. Virtual axis: Simulates the motion control of theoretical axes, without the need for external physical servo axes. Currently, the virtual PLC supports simulation of axis control instructions and component configuration. It can build axis control configuration programs through the motion control library and perform program simulation. It also supports axis group simulation, enabling motion control simulation of multiple axes in one group. In real axis simulation, the virtual PLC simulates not only EtherCAT real axis devices but also virtual EtherCAT slave devices, enabling motion control simulation of virtual devices.

NOTE

In communication simulation, the simulated instance object must have the same IP address as the current PC.

Use restrictions

PLCSIM simulation is implemented on the Windows system platform. Some functions cannot be supported for simulation, and certain functions may have boundary errors during simulation, as shown below.

- Task priority
  Windows is a non-real-time system, and the system priority settings differ significantly from those in Linux, so task priority simulation is not supported.
- Oscilloscope
  When the execution cycle of an IEC task is less than 40 ms, the values of POU variables obtained by the oscilloscope will have errors, with larger errors occurring at smaller task cycles.
- Instruction
  - There are differences in the instruction results between the X86 and ARM platforms. The differing instructions are as follows:

Differing Instruction	Note
REAL_TO_<TYPE>	<TYPE> data types include: BYTE, SINT, USINT, INT, UINT, DINT, UDINT, LINT, WORD, DWORD, TIME, DATE, DT.
LREAL_TO_<TYPE>	<TYPE> data types include: BYTE, SINT, USINT, INT, UINT, DINT, UDINT, LINT, WORD, DWORD, TIME, DATE, DT, TOD, STRING, WSTRING.
TRUNC	ST language operator for rounding.
TRUNC_INT	ST language operator for rounding.

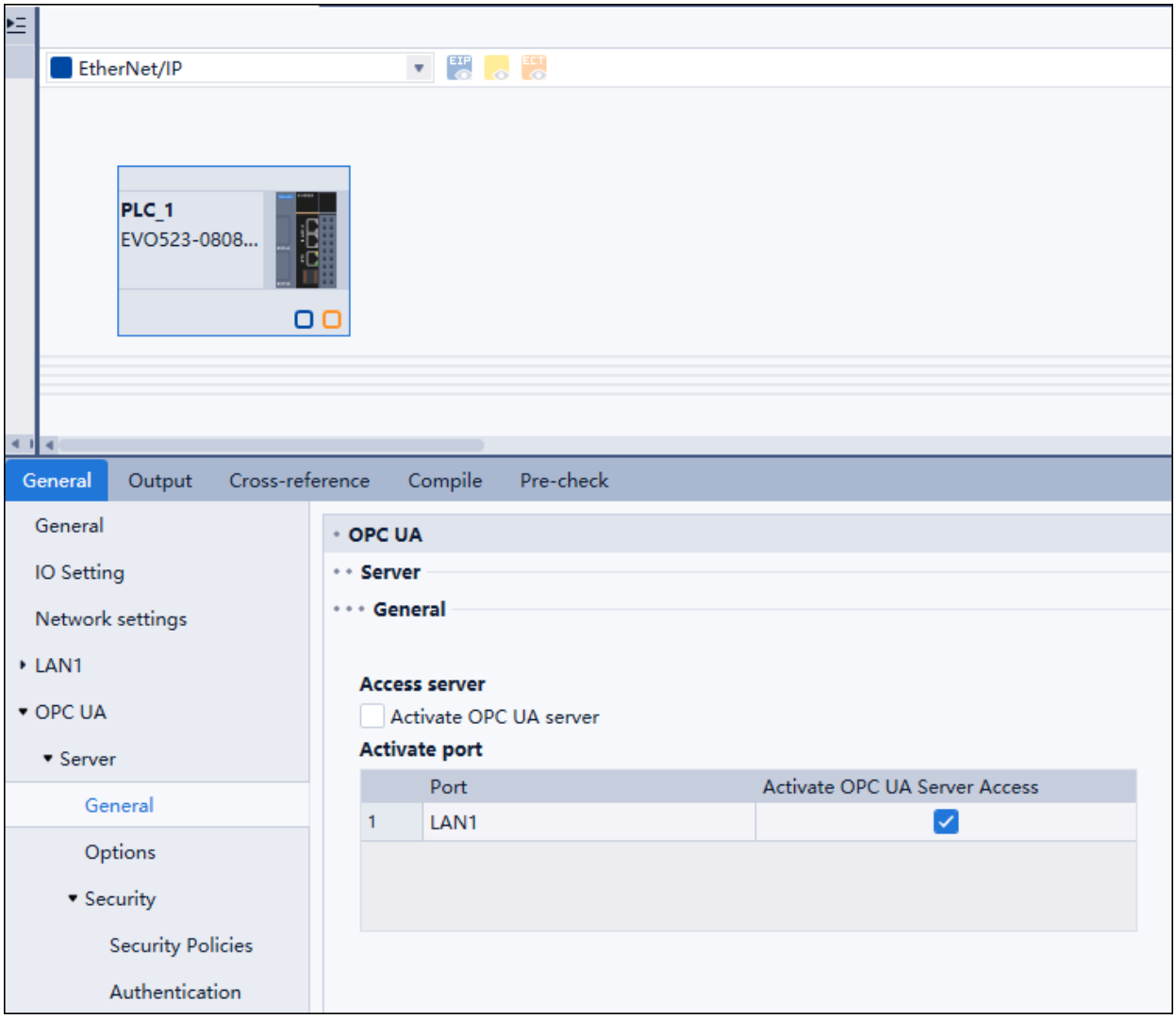
- The following instructions in the library do not support simulation:
  CANBus, CIA405, CmpHCHighSpeedIo, ExtendIo, HC_FINS, IOBlockHandle, MelsecServer, NetFreePort, S7Comm, SerialCom, SftpClient, SysHCPlcInfo, TcpUdpFreeProtocol.

- POU call
  In multi-task simulation mode, calling the same POU from multiple tasks is not supported. If no synchronization lock exists in the logic for the POU, resource contention issues may occur.
- Simulation duration
  After PLCSIM creates a PLC simulation, the simulation duration cannot exceed 8 hours. If it exceeds 8 hours, the PLC simulation will automatically exit.

10.13.4 Simulation Commissioning

10.13.4.1 OPCUA Communication Protocol Simulation

The simulation virtual PLC has OPC UA service capabilities, enabling access by OPC UA client to public network variables in the PLC program. The OPC UA communication protocol is enabled and its related parameters are configured via the iFA software, then downloaded to the virtual PLC instance object in PLCSIM, achieving OPC UA communication protocol simulation for the virtual PLC.
The OPC UA protocol stack parameter configuration is shown in the following figure:



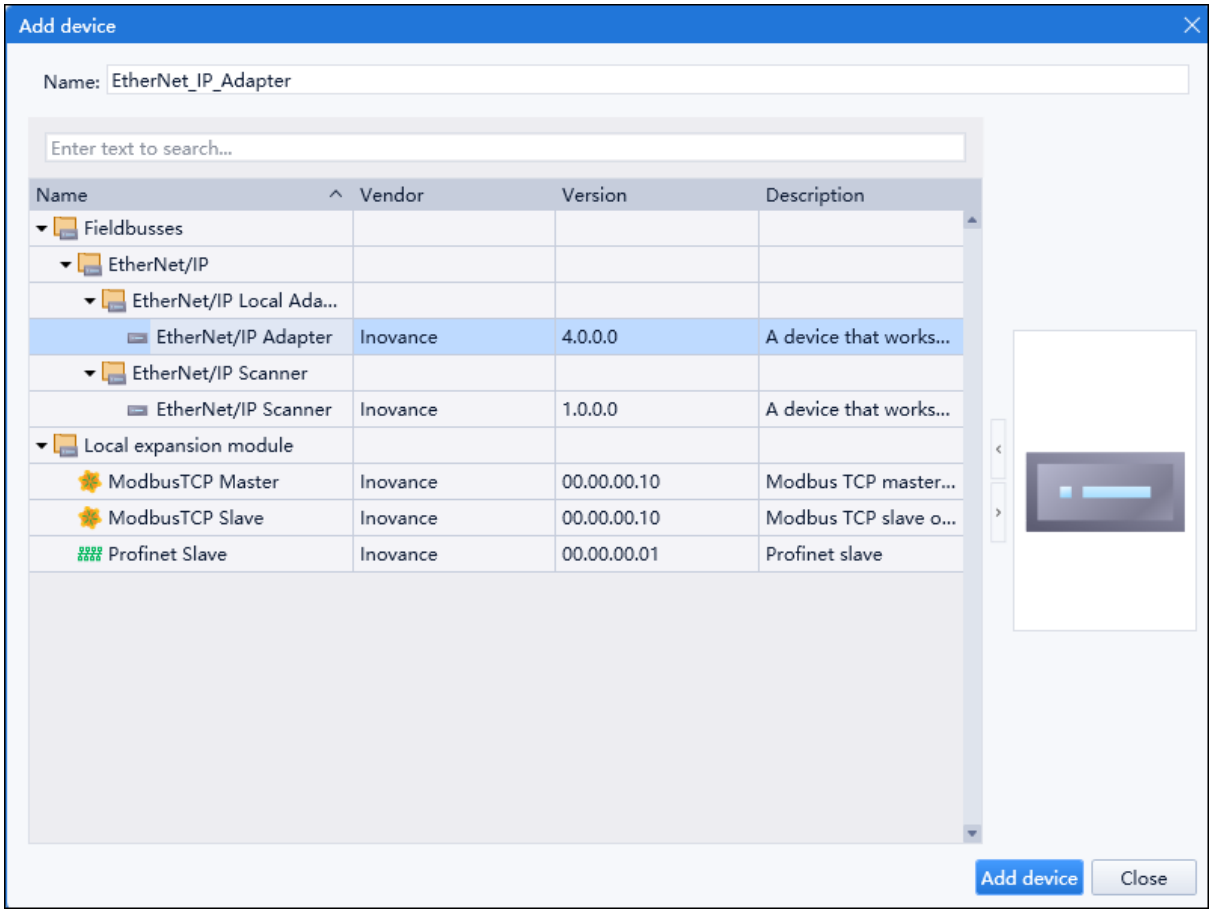
NOTE

OPC UA functionality depends on the .ini configuration file. Currently, OPC UA does not support Chinese paths; therefore, when using OPC UA functionality, PLCSIM must not be installed using a path containing Chinese characters.

10.13.4.2 EtherNet/IP Communication Protocol Simulation

EtherNet/IP (short for EIP) network communication operates in master-slave mode. To simulate EIP communication functions, it is necessary to build a master or slave for the EIP device. Configure the IP address of the EIP device in the slave or master device, and establish a Class1 or Class3 connection. I/O or label data for Class1 and Class3 connections are displayed in the I/O mapping area as communication data. The EIP device configuration is shown in the figure below:

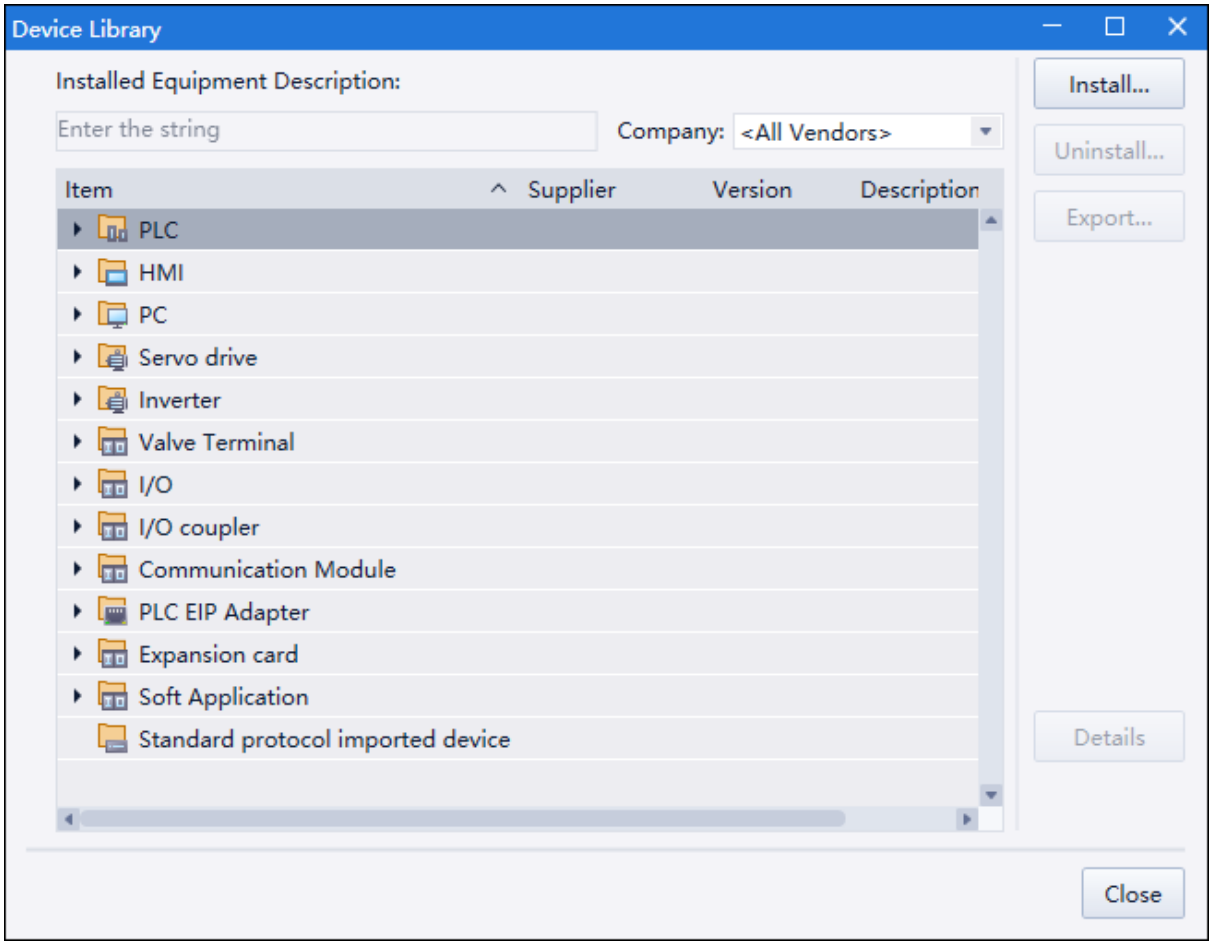
1. Right-click "LAN1 (EtherNet)" under "Hardware configuration" in the project tree and select "Add device". EIP devices include the master scanner and the slave adapter. A virtual PLC can function as both a master and a slave. The EIP device addition interface is shown in the figure below.



2. Select to add an EIP slave device adapter, and add an I/O connection within the adapter. After the connection is created, export the EDS file from the adapter.

   After the simulation program starts, the master device can connect to the slave device of the adapter, enabling I/O data exchange.

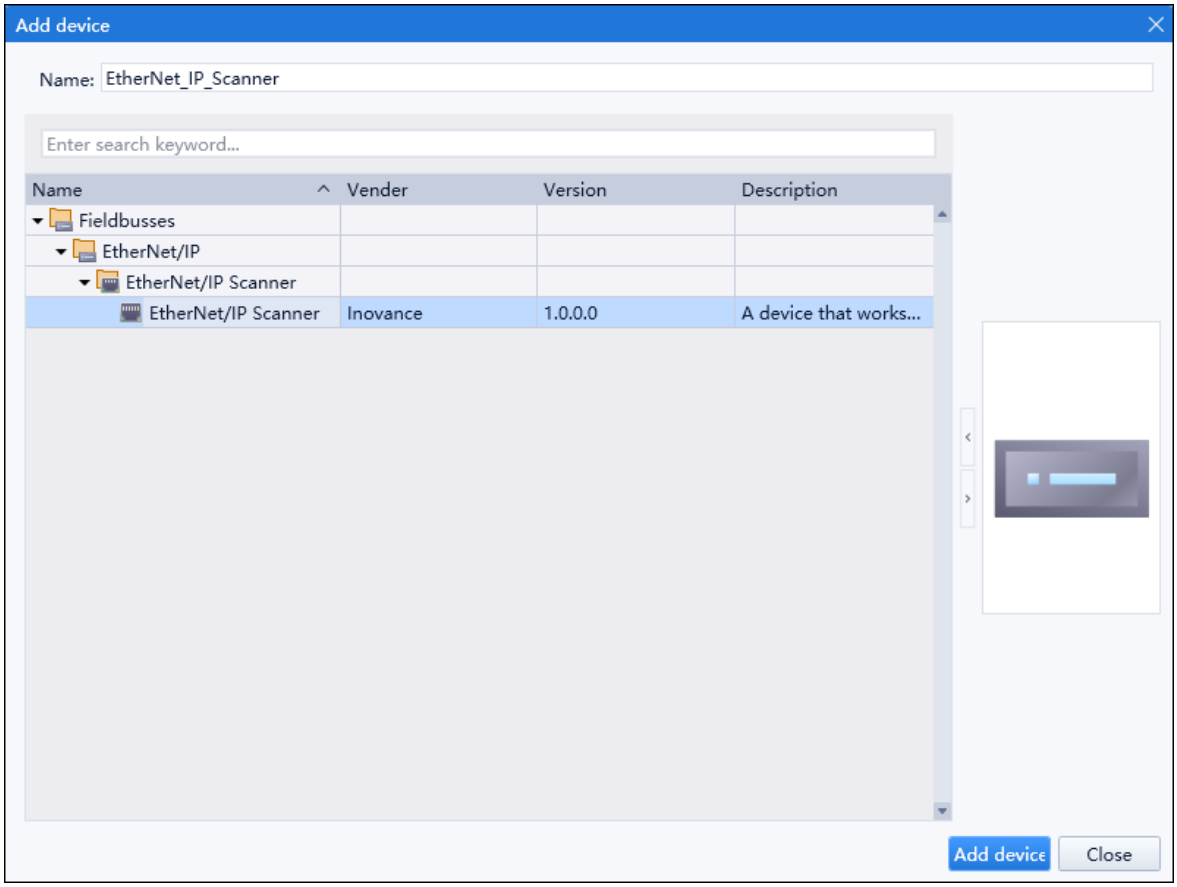
3. Add the EDS file exported in Step 2. Select "Tools > Device library" in the menu bar, as shown in the figure below.



4. After adding the EDS device file, add an EIP master scanner to the network node, and add a slave under the scanner device.

   a. Right-click "LAN1 (EtherNet)" in the project tree and select "Add device".

   b. Select "EtherNet/IP Scanner" and click "Add device".



5. After adding the EIP slave device, double-click the slave device, set the slave IP address in the "General" tab, and add I/O connections in the "Connection" interface.

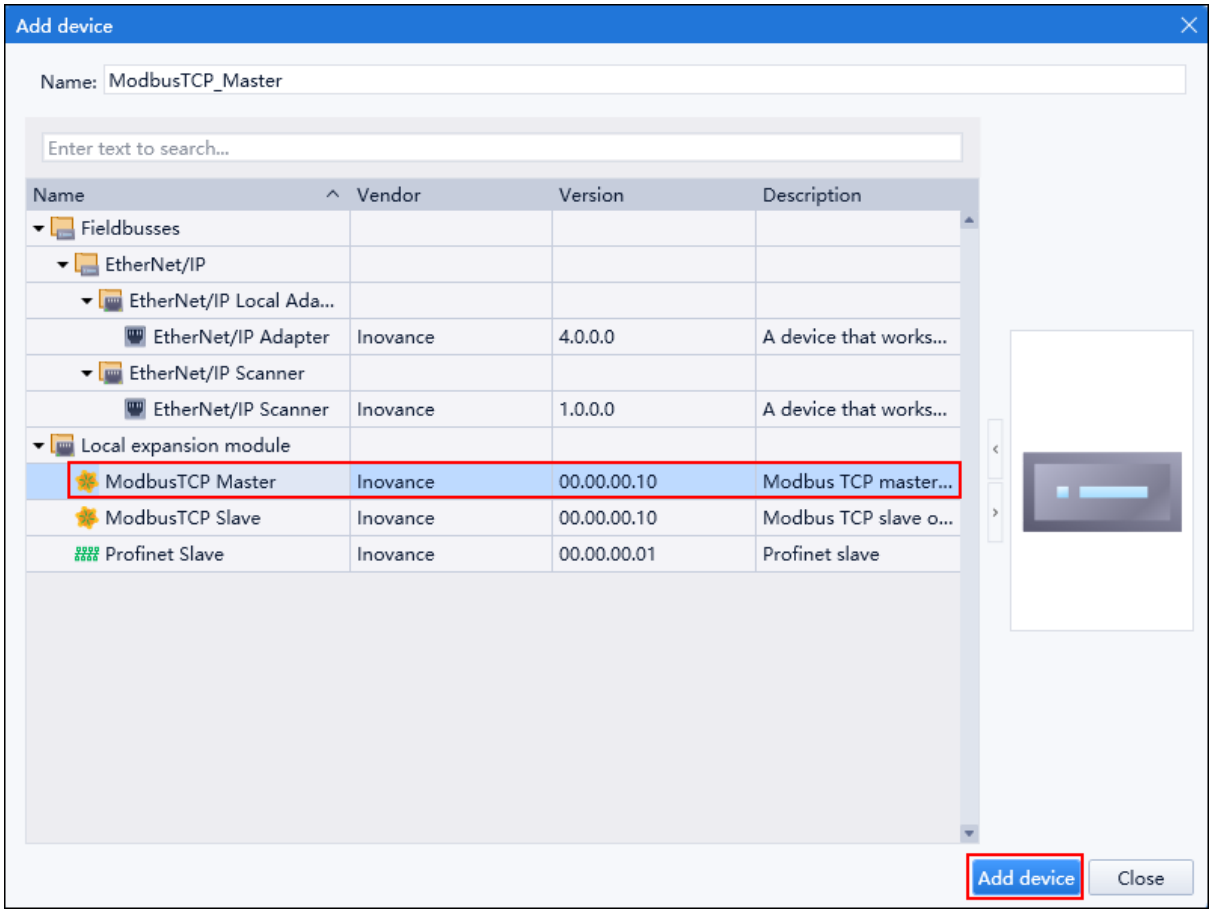
After configuring the remote slave IP and connection settings, re-download the program and start the simulation to establish the EIP I/O connection and complete I/O data exchange.

10.13.4.3 Modbus TCP Communication Protocol Simulation

ModbusTCP devices and EIP devices both use a master-slave communication mode.

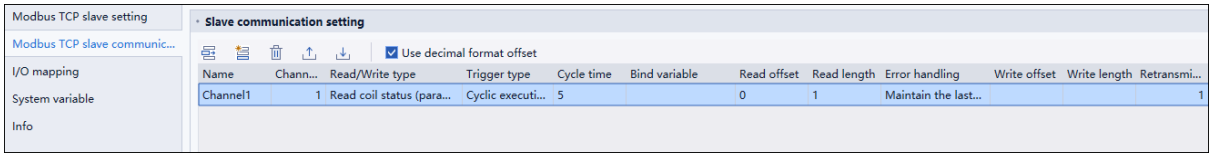
The procedure of adding a ModbusTCP device is similar to adding an EIP device. Right-click the network node, select the ModbusTCP master/slave to add it to the iFA device tree. The creation and operation of a ModbusTCP device are as follows.

1. Right-click "LAN1 (EtherNet)" in the left project tree and add a ModbusTCP master and slave.



2. To add a slave device to the ModbusTCP master, right-click the "ModbusTCP_Master" node and select "Add device".

3. Add I/O communication channels to the slave device. The communication channels of the slave device support instruction codes 01, 02, 03, 04, 05, 06, 15, and 16. Re-download the virtual PLC to enable ModbusTCP simulation communication. The I/O configuration for Modbus device communication is shown in the following figure.

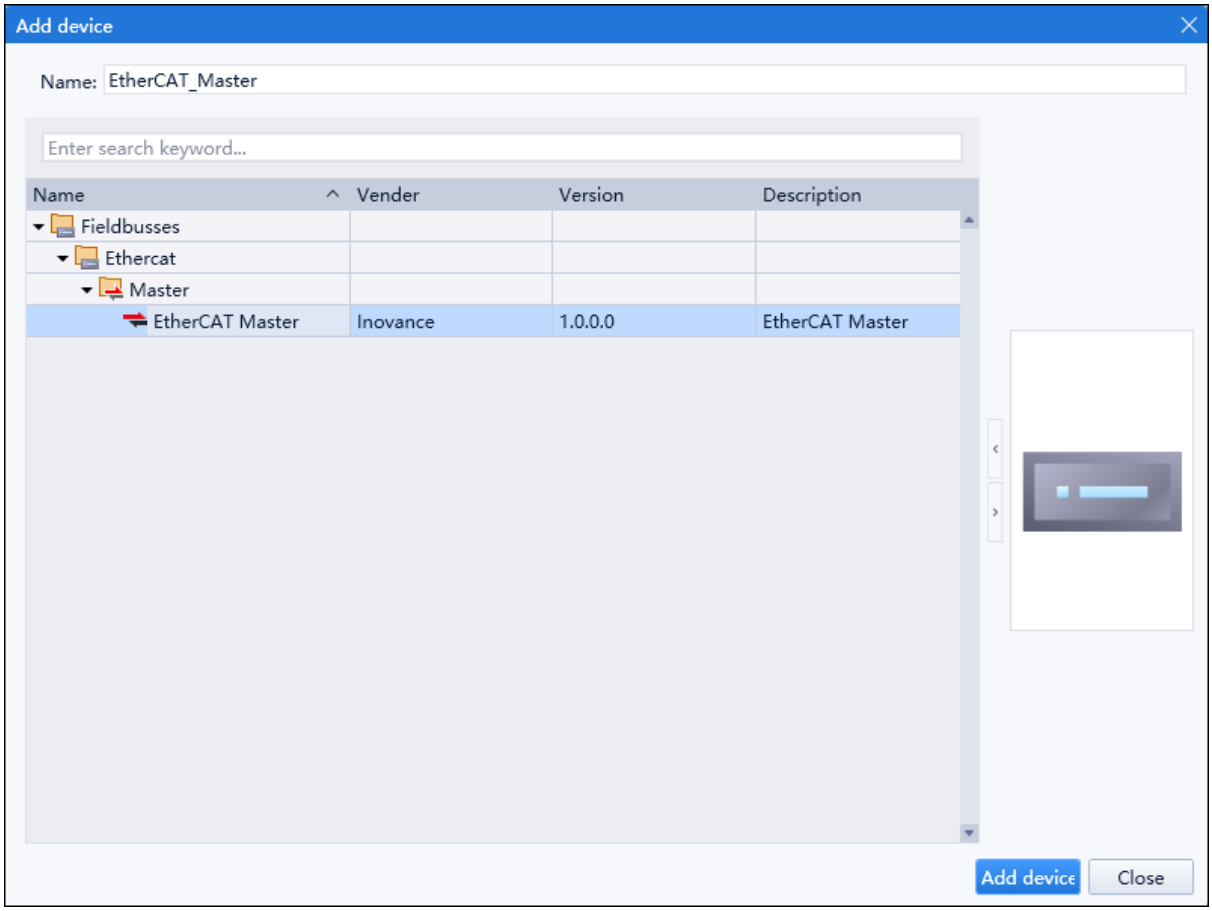


10.13.4.4 EtherCAT Communication Protocol Simulation

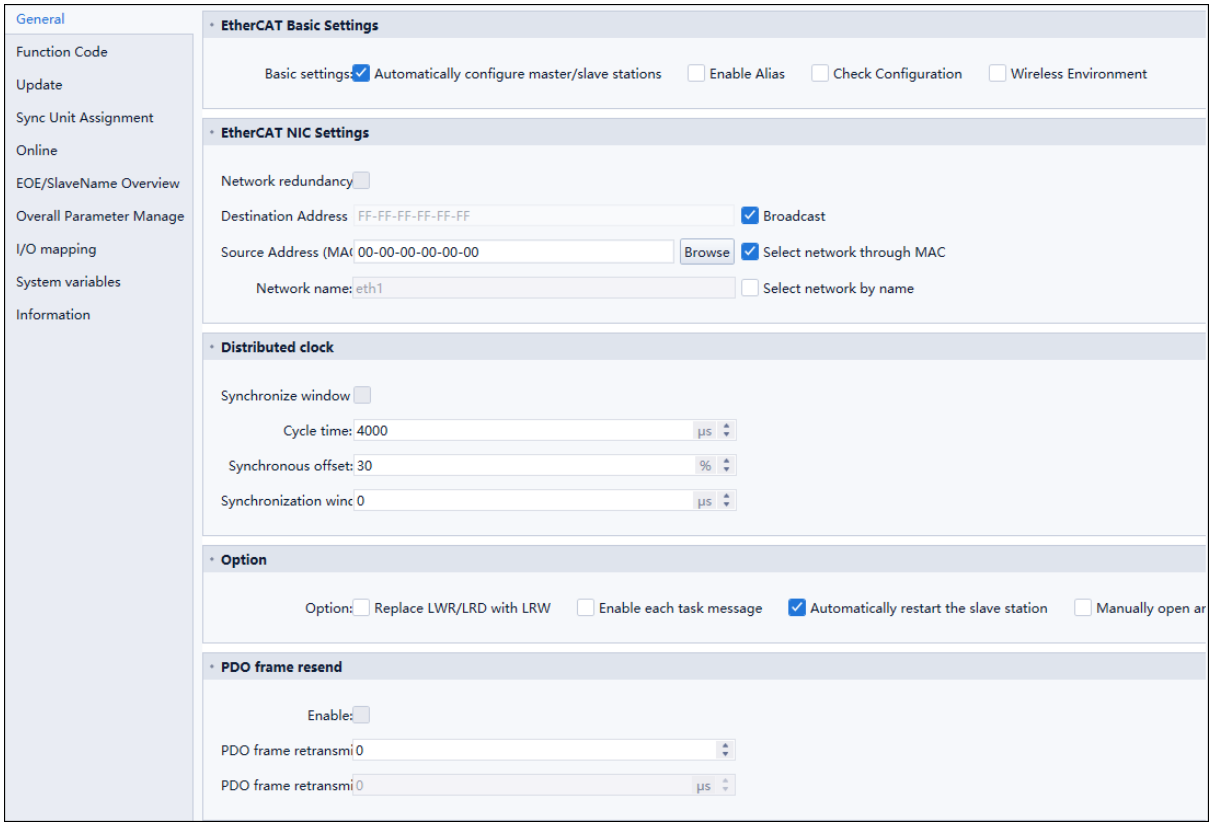
EtherCAT simulation supports only masters and servo devices, not I/O coupler devices.

The simulation cycle of the PLCSIM EtherCAT master supports a minimum of 10 ms. For EtherCAT simulation of drive real axes, up to 8 physical real axes can be simulated. The configuration steps for EtherCAT master in iFA are as follows:

1. Add an EtherCAT device: Right-click the "EtherCAT" node under "Hardware configuration" in the project tree, select "Add device", and select the EtherCAT master device, as shown in the figure below:



2. Configure the EtherCAT master parameters: Select the physical network card for EtherCAT communication, and set the EtherCAT device cycle to no less than 10 ms. The configuration of the EtherCAT master network card is shown in the figure below:



3. Add servo devices and configure the relevant parameters on the servo node.

4. Re-download the program to perform simulation of the servo real axes.

> NOTE
>
> The EtherCAT network cannot be selected using the network names.

10.14 Multi-User Collaboration

10.14.1 Overview

The multi-user collaboration feature allows multiple users to connect to the same PLC simultaneously to complete configuration and program development tasks. It mainly supports operations such as monitoring, commissioning, online modification, online comparison, program upload, and download.

> NOTE
>
> - This feature is only applicable to the EVO700 series PLCs with firmware version V1.7 or above
> - Multi-user collaboration is only supported for projects from the same source. Collaboration is not allowed for projects from different sources.
> - Same-source projects share the same original project, while different-source projects do not. If the original project is opened after a device version change or software upgrade, it will no longer be considered a same-source project.

During multi-user collaboration, the PLC acts as the server, allowing different programmers to download their control logic to the PLC via the iFA software. Other programmers can upload the latest PLC logic back to iFA. It is recommended that team members plan the division of work in advance to complete programming tasks efficiently and orderly through teamwork. For example, after member A finishes programming and downloads the program to the PLC, member B uploads the PLC program to their local environment and continues programming, and so on.

10.14.2 Multi-User Online Editing

To accommodate multi-user collaboration scenarios, the system separates device online operations from program download. When users perform online operations, iFA only logs in to the selected device without triggering a download. When online conditions are met, iFA can directly log in to the PLC. Compared to the previous online download mode, the new mode has the following features:

- The operation involves additional steps, but the compilation and download speed is significantly improved.
- Avoids program overwrite conflicts caused by multiple user logins; supports real-time online collaboration among multiple users (including program modification, real-time monitoring, and synchronized download).
- User modifications can be automatically synchronized across multiple terminals in real time.

Restrictions

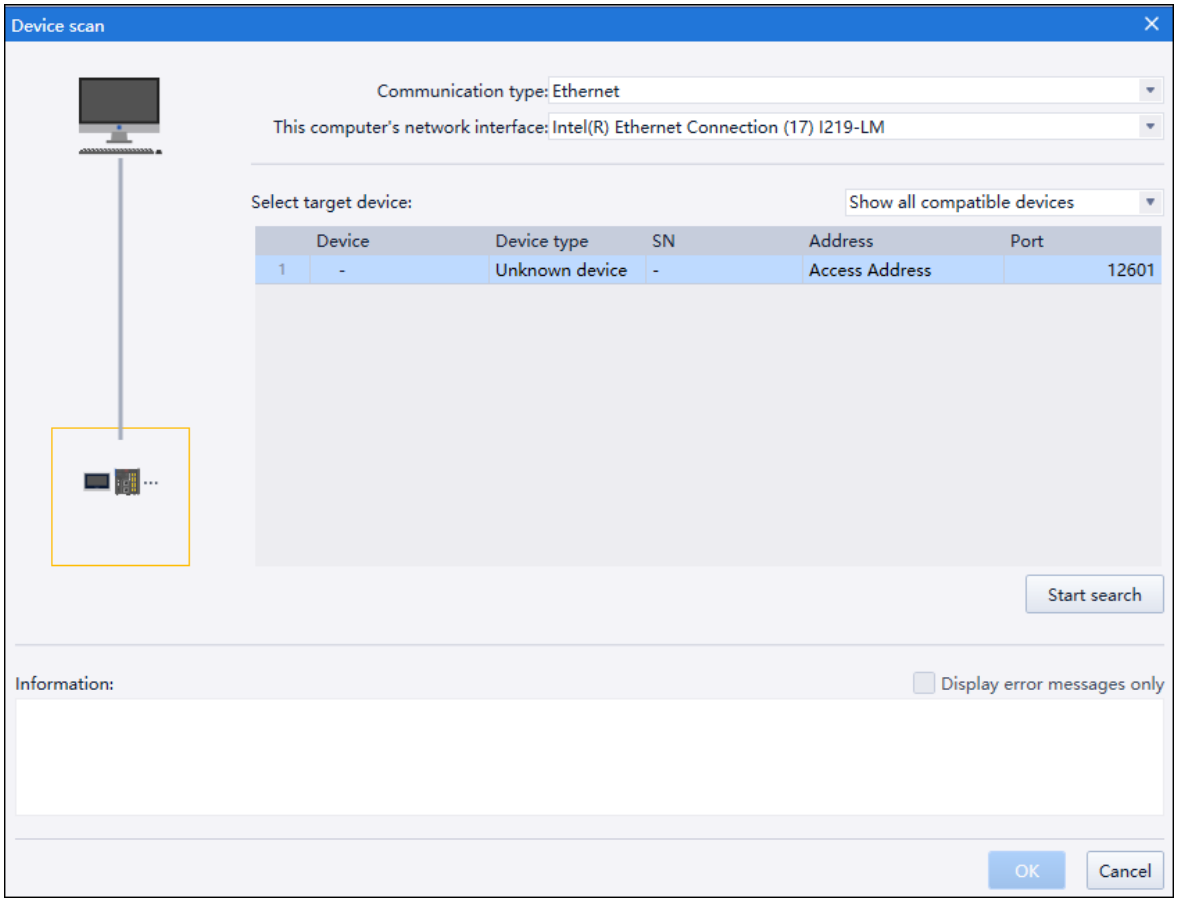
A device supports up to five users logging in simultaneously. If it is exceeded, the system displays a pop-up message, indicating that the login limit is reached and the device supports a maximum of five concurrent logins.

Prerequisites

- Device types are consistent and versions are compatible.
- Project files are from the same source project.

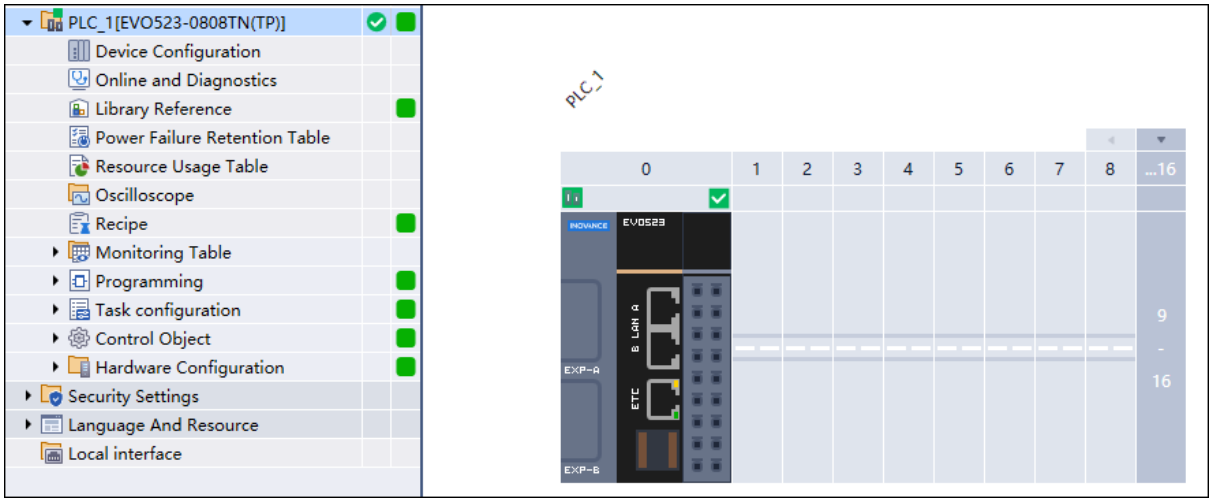
Procedure

1. Go online.
   - After selecting the device node in the project tree, click the "Online" icon in the menu bar.
   - Right-click the device node in the project tree and select "Online".
   - After selecting the device, use the Ctrl+K shortcut key to go online; use the Ctrl+M shortcut key to go offline.
   - In the network view/device view, right-click the device and select "Online".
2. (Optional) If a device connection has not been established, configure the device connection.
   - In the pop-up "Device scanning" window, click "Start search".



- Select the target device and click "OK".

3. (Optional) If online conditions are not met, an error prompt will appear. Troubleshoot and resolve the issue based on the prompted reasons.
4. After going online, the device identifiers in the project tree and device configuration view will change, as shown in the figure below.



Result

Users can view the current number of logged-in users in real time, compare differences between the local project and the controller project, and perform online editing.

- Number of logged-in users

Users can view the current number of online users in the status bar of the software interface. For details, see 5.2 Project View.

- Project comparison

After the device is online, the status bar on the right of the project tree will display the difference status between the local project and the controller project in real time. If a node supports detailed comparison, double-click its status icon to jump to the corresponding comparison view. The descriptions of each status icon are shown in the table below.

Status	Description
	Projects are consistent.
	Projects are inconsistent; double-click to view the detailed comparison for this node.
	Exists only locally.
	Exists only on the controller.
	Content or number of child nodes are inconsistent.



In addition to the above methods, you can also view project differences through the project comparison function. For details, see 10.12.3 PLC Project Comparison > Offline/Online Comparison.

- Online editing

After the device is online, users can perform online editing to modify variables and programs in real time.

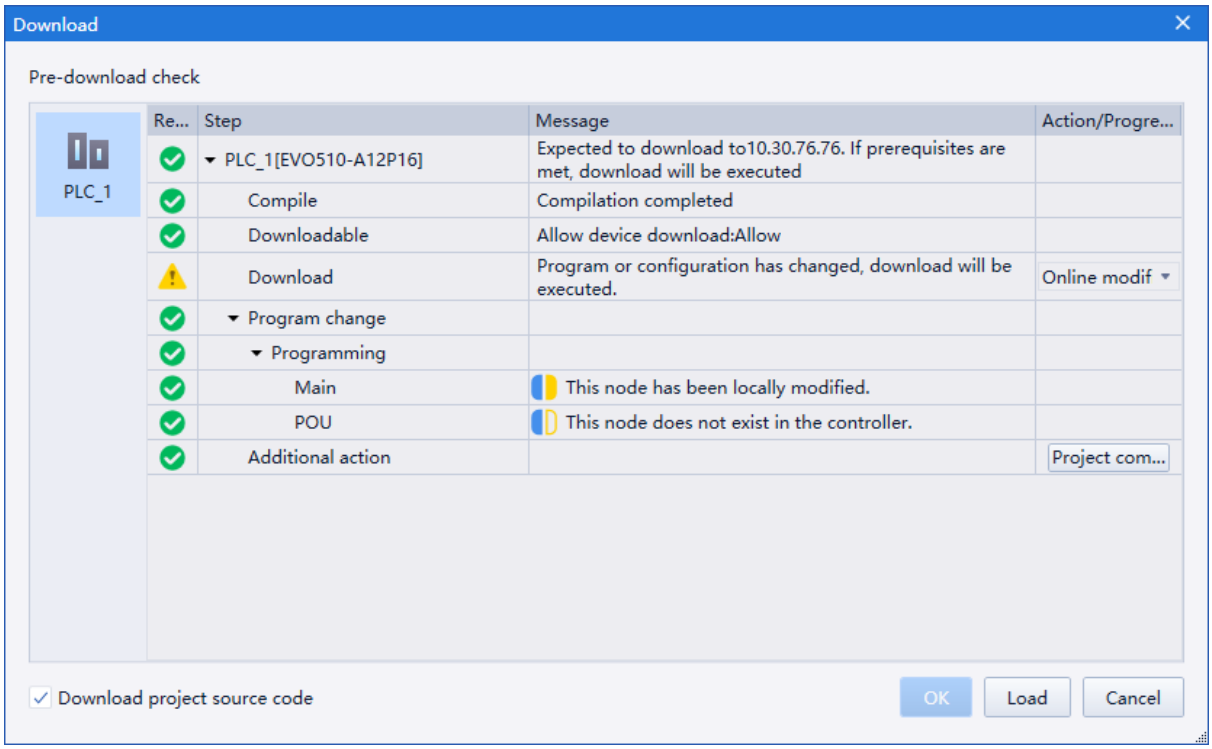
10.14.3 Multi-User Download Synchronization

Multi-user download is a core feature in multi-user collaborative scenarios. It is used to resolve potential conflicts and overwriting issues when multiple users download projects to the same controller at the same time. This feature allows multiple users to initiate download tasks simultaneously. The system performs conflict detection during execution and provides clear status prompts to guide users, ensuring the consistency and correctness of the project version in the controller.

Procedure

1. Click  on the toolbar to open the "Download" window. Before the download, compilation and a series of checks are automatically performed.
   When the current project is inconsistent with the project in the controller, the prompt "Program or

configuration has changed, download will be executed" is displayed.



2. Under "Program change", view the program change details and select the appropriate action (currently only changes to programming and task configuration nodes are displayed).
   - Download on sync: Synchronize and download the local program or configuration to the controller.
   - Upload on sync: Synchronize and upload the controller program or configuration to the local system.

Icon	Prompt	Action/Progress
	This node does not exist in the controller.	Default: Download on sync; not modifiable.
	This node does not exist in the project tree.	Default: Upload on sync; not modifiable.
	The node has been modified by others.	Default: Upload on sync; not modifiable.
	This node has been locally modified.	Default: Download on sync; not modifiable.
	This node has been modified by multiple parties.	Options: Download on sync or upload on sync.

3. Click "Download after sync" or "Download without sync".
   - Download after sync: Based on the controller's program or configuration, locally synchronize nodes marked as "Upload on sync"; download the program or configuration of nodes marked as "Download on sync" to the controller.
   - Download without sync: Download the current local program or configuration to the controller, overwriting the controller configuration. This action is independent of the node actions in "Program change".
4. After download, a success message appears. If the download fails, an error prompt will appear. Please troubleshoot and handle the issue according to the prompt.

NOTE

- Before downloading, please carefully confirm that the synchronization items are correctly selected. Synchronization operations cannot be undone. Please proceed with caution!
- Before downloading, please verify the code. In multi-user collaboration, promptly synchronize modifications by others to avoid overwriting.
- In multi-user collaboration, in addition to the automatic project synchronization download method described above, manual synchronization can also be used: Manually upload the programs generated by other users to the local system, and then download the entire project from the local system to the PLC. There are three ways to manually synchronize the project. You can flexibly choose based on your actual situation.
  - Synchronize the current project information in the PLC by uploading only differences. For specific operations, see 10.14.5 Upload Only Differences.
  - Synchronize the current project information in the PLC using the project comparison feature. For specific operations, see 10.12.3 Offline/Online Comparison and 10.12.4 Viewing and Processing Comparison Result in the "PLC Project Comparison" chapter.
  - Synchronize the current project information in the PLC using the project comparison feature in the "Download" window that pops up when logging in to the PLC. For operations after project comparison, see 10.12.3 Offline/Online Comparison and 10.12.4 Viewing and Processing Comparison Result in the "PLC Project Comparison" chapter.

10.14.4 Multi-User Commissioning

When multiple users are simultaneously performing online commissioning on the same controller, the system ensures orderly operation and data consistency through collaboration rules and state management mechanisms, preventing conflicts between commissioning operations. The following table lists all mutually exclusive operations. When performing an operation in the left column, ensure that no other user is performing the corresponding operation in the right column.

When performing this operation	Ensure no other user is performing the following operations
Download	Download, upload, force, release, write, commissioning, clear user program, start, stop, sync source code, sync compilation information, online login, incremental file upload
Incremental file upload	Download, clear user program, sync source code, sync compilation information
Source code upload (Upload as new device)	Download, upload source code, clear user program, sync source code
Forcing a value	Download, clear user program
Releasing a value	Download, clear user program
Writing a value	Download
Execute	Download, clear user program
Stop	Download, clear user program

When performing this operation	Ensure no other user is performing the following operations
Commissioning	-
Clearing user programs	Download
Cold reset	Download, clear user program
Hot reset	Download, clear user program
Sync source code (Download iFA project to PLC)	Download, go online, sync source code, upload source code, incremental file upload
Sync compilation information (Download iFA compilation information to PLC)	Download, sync compilation information, go online, incremental file upload
Go online	Download, sync source code, sync compilation information

Axis commissioning

In a multi-user collaborative commissioning environment, the system uses permission control to ensure the security and data consistency of the axis commissioning process.

Restrictions

- Allowed operations: Different users can perform commissioning on different axes simultaneously.
- Prohibited operations: Only one user can perform commissioning on the same axis simultaneously. If another user attempts to perform commissioning on an axis that is already occupied, the system will prohibit the operation and display a prompt.

Precautions

- When User A is performing commissioning on an axis and communication is abnormally interrupted (e.g., iFA unexpectedly closed or LAN cable disconnected), the system will automatically exit User A's commissioning state, allowing User B to perform commissioning
- If User A has opened axis X and entered the commissioning state, User B's commissioning attempt on the same axis X will be prohibited. User B must wait until User A exits the commissioning state.

10.14.5 Uploading Only Differences

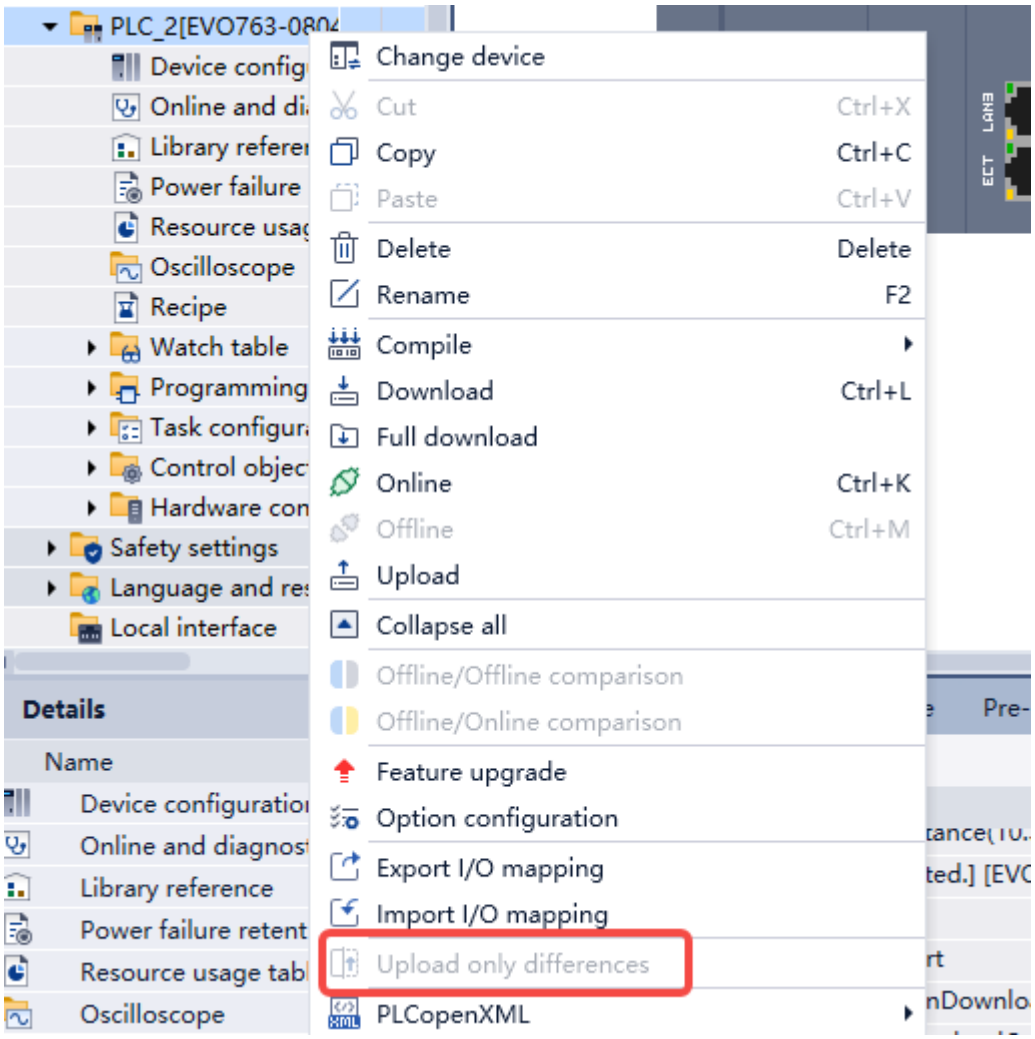
This feature is used to synchronize changes in the controller to the local project.

Prerequisites

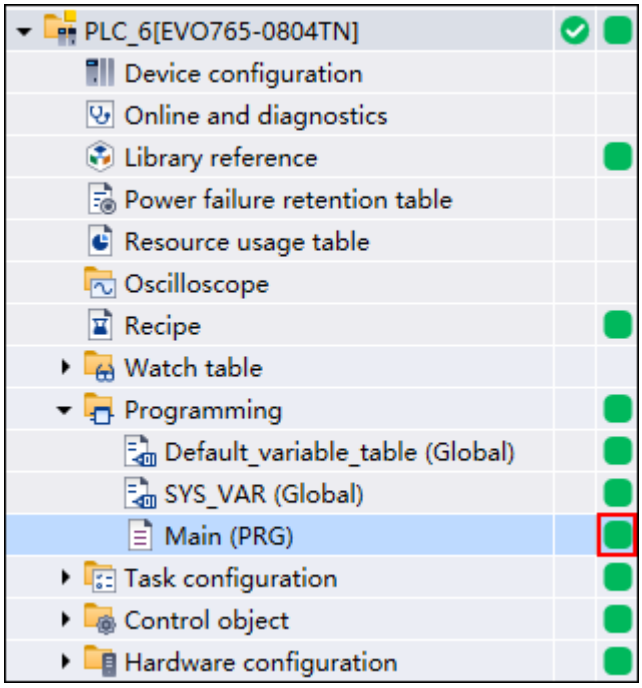
- Currently, the node types that support difference synchronization are programming nodes, PID configuration nodes, and task configuration nodes in control objects.
- The node exists only in the controller and is not present in the local project.
- The node has not been modified in the local project, but it has been modified in the controller.

Procedure

1. Right-click the device name in the project tree and select "Upload only differences".



2. After execution, the comparison status icon in the project tree changes to , indicating that the local project is consistent with the project in the controller.



11 Safety PLC Programming

Safety PLC programming complies with the following standards:

- LD and FBD languages defined by IEC 61131-3
- T3 tool requirements defined by IEC 61508-3
- IEC 62061 SIL CL 3 and ISO 13849-1 PL e/Cat.4 certifications

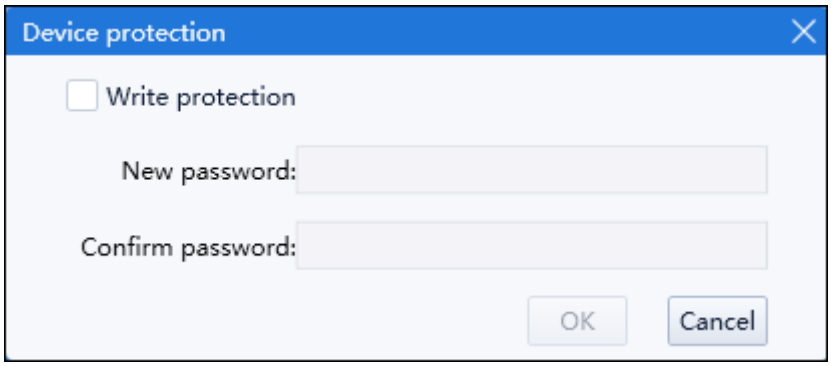
11.1 Safety PLC Access Protection

11.1.1 PLC Configuration Write Protection

In the iFA project, the safety PLC supports individual write protection settings. The write protection ensures that all content within this safety PLC (including hardware configuration and programs) can only be viewed and cannot be modified.

Setting write protection

In the left project tree, right-click a safety PLC project node, and select "Set/Delete access protection" to open the device protection interface.



Check "Write protection" to activate the password input fields. Enter a valid and consistent password, confirm it, and click "OK" to apply the setting.

Deleting write protection

- Method 1: In the left project tree, right-click a safety PLC project node, and select "Set/Delete access protection". Uncheck "Write protection", a password verification window will appear. Enter the correct password and click "OK" to apply.
- Method 2: In the left project tree, right-click a safety PLC project node, and select "Disable/Enable access protection". A password verification window will appear. Enter the correct password and click "OK" to apply.

11.1.2 PLC Permission Configuration

The method and criteria for configuring PLC permissions in a safety PLC are the same as those for a standard PLC. For details, see 10.11.6 PLC Permission Configuration in the "PLC Programming" chapter.

11.2 Programming Basics

11.2.1 Variables

11.2.1.1 Data Types

11.2.1.1.1 Standard Data Types

Data Type	Value Range	Memory Usage (Byte)	Input Example
BOOL	TRUE, FALSE	1	TRUE FALSE
BYTE	0 to 255	1	BYTE#2#00001111
WORD	0 to 65535	2	WORD#61680 DWORD#2#0000_0000_1111_0000_1111_1111_0000_1111
DWORD	0 to 4294967295	4	
INT	-32768 to 32767	2	100 INT#2#00000000_01100100 INT#8#144 INT#10#100 INT#16#64 -100
DINT	-2147483648 to 2147483647	4	
TIME	T#0d0h0m0s0ms to T#49d17h2m47s295ms	4	T#10d_20h_30m_20s_630ms T#61m5s T#3s20ms TIME#10d_20h_30m_20s_630ms TIME#6d_10m
DATE_AND_TIME	DT#1970-01-01-00:00:00 to DT#2106-02-07-06:28:15	4	Start with DT# or dt#, and followed by a date and time in yyyy-mm-dd-hh:mm:ss format. dt#2007-07-01-05:40:25

11.2.1.1.2 Safety Data Types

Data Type	Value Range	Memory Usage (Byte)	Input Example
SAFEBOOL	TRUE, FALSE	1	TRUE FALSE

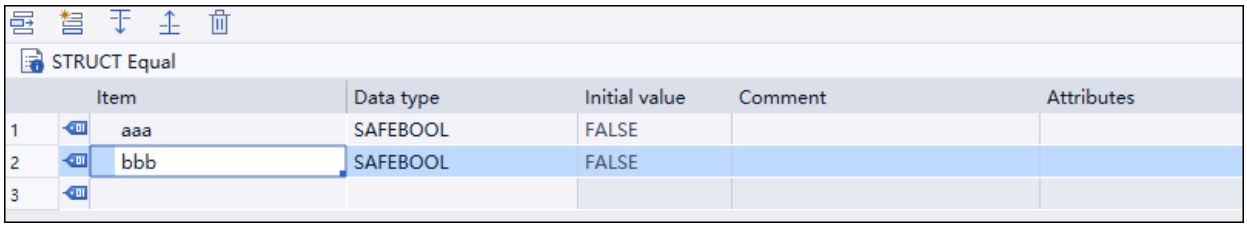
Data Type	Value Range	Memory Usage (Byte)	Input Example
SAFEBYTE	0 to 255	1	BYTE#2#00001111
SAFEWORD	0 to 65535	2	WORD#61680 DWORD#2#0000_0000_1111_0000_1111_1111_0000_1111
SAFEDWORD	0 to 4294967295	4	
SAFEINT	-32768 to 32767	2	INT#2#00000000_01100100 INT#10#100 INT#16#64
SAFEDINT	-2147483648 to 2147483647	4	
SAFETIME	T#0d0h0m0s0ms to T#49d17h2m47s295ms	4	T#10d_20h_30m_20s_630ms T#61m5s T#3s20ms TIME#10d_20h_30m_20s_630ms TIME#6d_10m
SAFE_DATE_AND_TIME	DT#1970-01-01-00:00:00 to DT#2106-02-07-06:28:15	4	Start with DT# or dt#, and followed by a date and time in yyyy-mm-dd-hh:mm:ss format. dt#2007-07-01-05:40:25

11.2.1.1.3 User-Defined Types

In safety PLC programming, custom data types only support structures (STRUCT) with one level.

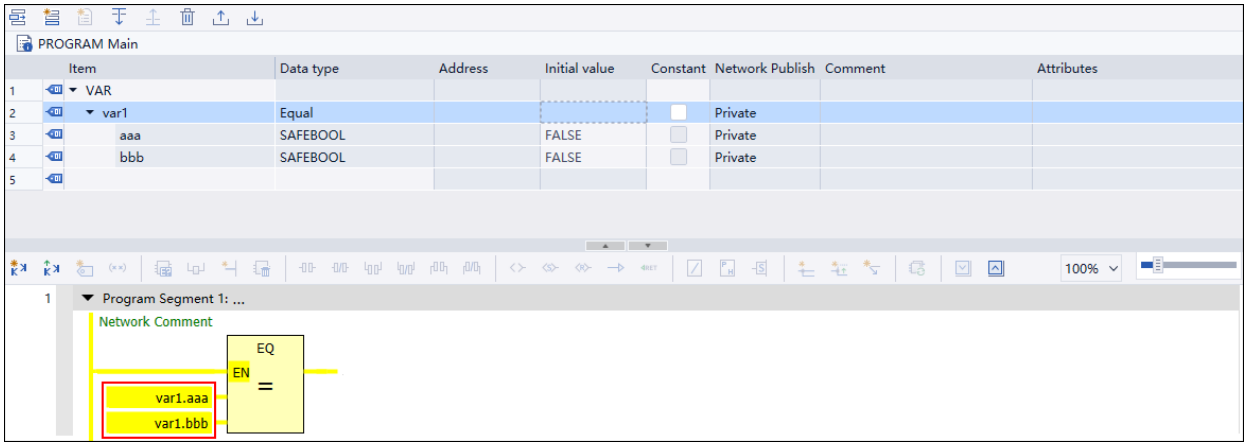
Structure definition

In the project tree, right-click "Programming" and select "Add user-defined type". Select "Struct" in the pop-up window. Click "OK" and the DUT editor is automatically opened. Create a structure named Equal in the DUT editor, as shown below.



Structure usage

After a structure is created, create a variable in the program with the type set to the newly created structure name, that is Equal. In the "PROGRAM Main" window, enter "VariableName." (the variable name followed by a dot). The system will automatically display a dropdown list of the structure's members. Select to perform read or write operations on the structure, as shown below.



11.2.1.1.4 Data Type Conversion

The safety PLC data type conversion is as follows:

- Safety data remains a safety type after data type conversion. Standard data remains a standard type after conversion.
- Safety type can be automatically converted to standard type, but standard type cannot be converted to safety type.
- Conversion of specified data types is supported. The iFA software provides explicit conversion instructions for users. For details, see 11.10.1.6 Conversion Operators.

> NOTE
>
> Signal from a safety type variable can be input into a standard type variable, but signal from a standard type variable cannot be input into a safety type variable.

11.2.1.2 Variable Declaration

This section only covers the functional differences between variable declarations in safety PLCs and standard PLCs; for other general functions, see 10.1.1.9 Variable Declaration in the "PLC Programming" chapter.

11.2.1.2.1 Variable Declaration Properties

A variable has eight properties: ID (name), data type, address, initial value, constant, network public property, comment, and property.

- ID
  ID is the variable's name. When naming variables, keep the following in mind:
  - It cannot include spaces or special characters.
  - It cannot be a predefined keyword.
  - Names are case-insensitive.
  - The maximum length is 128 characters.
  - Names cannot be defined repeatedly.

- Data type

  Refer to 11.2.1.1 Data Type for the data types supported by the safety PLC.

- Address

  Address is displayed only and not editable; it is automatically generated by the software.

- Initial Value

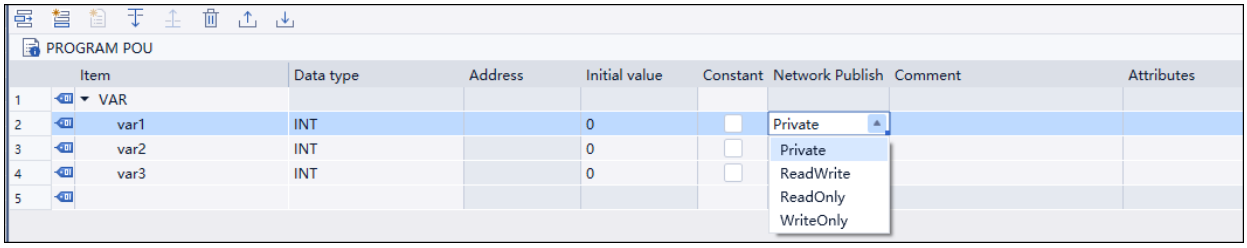
  The initial value of a variable varies depending on its data type and can be customized during variable declaration.

- Constant

  A constant is a value that does not change.

- Network public property

  The network public property can be used to display the network public status of a variable.



  - When the "Private" option is selected, the variable does not support access by third-party equipment or software.
  - When the "Read/Write", "Read-Only", or "Write-Only" option is selected, the variable supports access by third-party equipment or software.
    - Read/Write: The data of the variable can be modified or viewed.
    - WriteOnly: The data of the variable can only be modified, but cannot be viewed or read.
    - ReadOnly: The data of the variable can only be viewed or accessed, but cannot be modified.

    Network public property restrictions:
  - The number of public read-only variables is limited to 512, so is the number of public write-only variables.
  - The above public variables include both safety variables and standard variables.
  - For the above read-only and write-only variables with public property "Read/Write", each type occupies one count.

- Comment

  Used to add annotations to variables, making them easier to understand.

- Properties

  Properties are used for differentiated handling of individual variables. For example, when a user needs to permanently display a certain variable in decimal format, this can be achieved by setting the variable's attribute 'displaymode' := 'dec'. The following are commonly used properties:

No.	Keyword	Value	Note
1	displaymode	'bin', 'dec', 'hex'	Specifies the base for displaying the online value of a variable, supporting binary, decimal, and hexadecimal formats.

No.	Keyword	Value	Note
2	visibility	none	Visibility check for variables/notations; if it includes "none," it is hidden.
3	qualified_only	/	When this marking is added to a global variable, it must be accessed using the global variable group name followed by a dot.

11.2.1.2.2 Variable Address Range

The following table shows the address ranges of PLC variables.

Storage Area	Address Range (Byte)
I area	0 to 8191
Q area	0 to 8191

11.2.1.2.3 Default Variable Table

When adding a safety PLC device to the project, a "Default variable table (Global)" node is automatically created under the device. Double-click this node to open the corresponding window.

The default variable table does not support copying or deletion, and the node name cannot be modified.

The default variable table supports import and export operations. For details, see sections 10.1.1.10.6 Variable Input and 10.1.1.10.7 Variable Output.

When creating new variables in I/O mapping, they are automatically added to this table.

11.2.1.2.4 System Variable Table

When adding a safety PLC device to the project, a "SYS_VAR(Global)" node is automatically created under the device. Double-click this node to open the corresponding window.

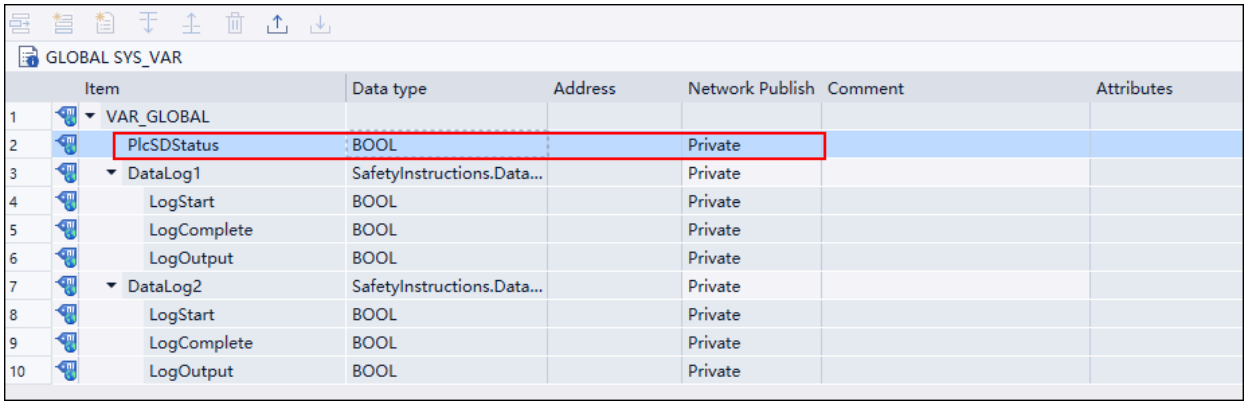
The system variable table is the variable table created by default, which does not support copying or deletion, and the node name cannot be modified.

The system variable table supports export. For details, see section 10.1.1.10.7 Variable Export. Variable import is not supported.

System variables consist of two parts:

- SD card status variables

Variables related to the PLC hardware configuration of the SD card status. By default, the corresponding variable PlcSDStatus is created in the system variable table. Variable name and data type cannot be modified. Users can modify the network public property of variables according to requirements. For details, see 11.2.1.2.1 Variable Declaration Property.



- Data record status variables

Variables related to the PLC hardware data record. By default, corresponding variables are created in the system variable table, named PlcSDStatus, DataLog1, and DataLog2. Variable names and data types cannot be modified, and their network public properties only support private and read-only. Users can modify the network public property of variables according to requirements. For details, see 11.2.1.2.1 Variable Declaration Property. This requires that users has configured data record and enabled this feature. The current data record operation status can be obtained through the structure SafetyInstructions.DataLog, and variables LogStart, LogComplete, and LogOutput.

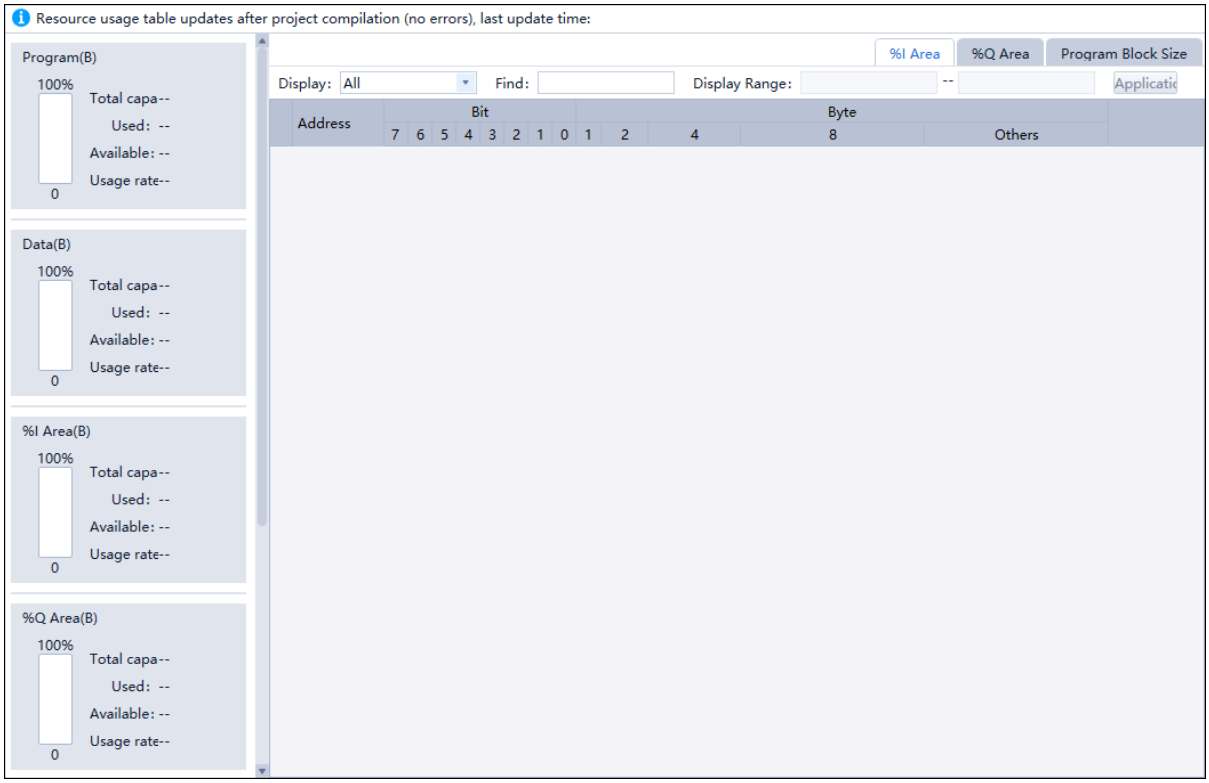
Table11-2-1-2-4-1 Descriptions of safety data record system variables

Variable Name	Data Type	Description
PlcSDStatus	BOOL	TRUE: SD card status of the communication unit is normal FALSE: SD card status of the communication unit is faulty
DataLog1.LogStart	BOOL	TRUE: Safety data record 1 is currently running FALSE: Safety data record 1 has stopped
DataLog1.LogComplete	BOOL	TRUE: Safety data record 1 has completed FALSE: Safety data record 1 has not completed
DataLog1.LogOutput	BOOL	TRUE: Safety data record 1 has completed writing to file FALSE: Safety data record 1 has not completed writing to file
DataLog2.LogStart	BOOL	TRUE: Safety data record 2 is currently running FALSE: Safety data record 2 has stopped
DataLog2.LogComplete	BOOL	TRUE: Safety data record 2 has completed FALSE: Safety data record 2 has not completed
DataLog2.LogOutput	BOOL	TRUE: Safety data record 2 has completed writing to file FALSE: Safety data record 2 has not completed writing to file

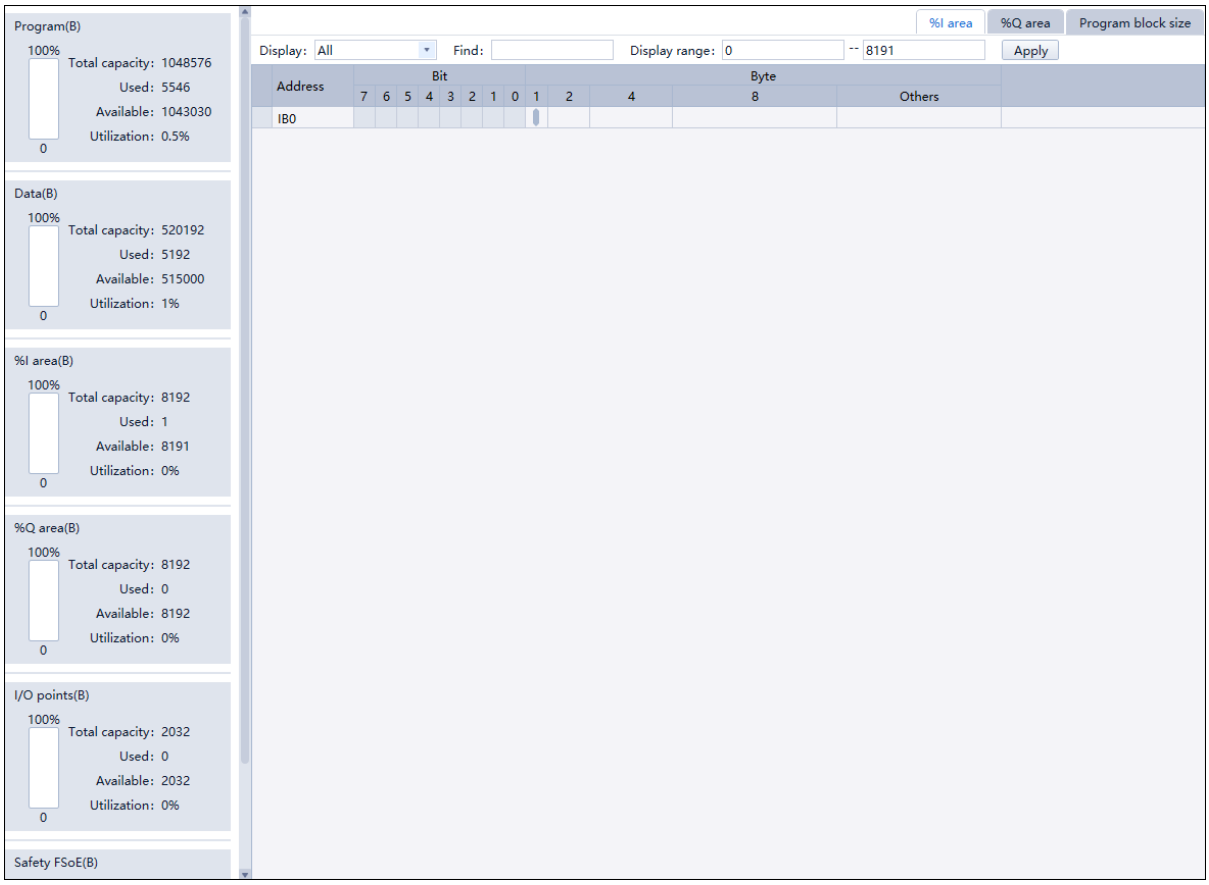
11.2.1.2.5 Resource Usage Table

The resource usage table is mainly used to display the status of the program, including the overall resource usage of the program, data, I area, Q area, I/O connections, and the safety CIP/FSoE, as well as specific displays of the I area, Q area, and program block size.

- Equipped with an EIP communication unit

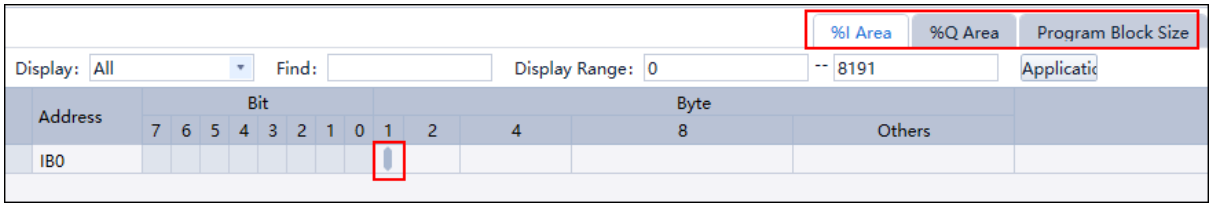


- Equipped with an ECT communication unit



1. The resource usage table is not generated in real-time but is updated after the project is compiled (without faults). The updated results after compilation need to be saved so that they can be viewed the next time the project is opened. Variables that have not been assigned absolute addresses will not be counted or displayed in the resource usage table. The resource usage table only shows the usage of defined variables in the program. Undefined and unused variables are not displayed in the address conflict table on the right side, but only in the statistical information on the left side. Variables declared with address information AT are also included in the display and statistics.

2. The left side displays the overall resource usage using six cards, including program, data, I area, Q area, I/O connections, and safety CIP. The "Display" filter in the tabs of the I area, and Q area can be divided into four categories:
   - All: Lists all defined (configured, used, or coupled) variables.
   - Used: Lists all used variables and addresses (including direct addresses).
   - Idle: Lists all defined but unused variables.
   - Conflict: Lists all variables detected to have address conflicts or overlaps.
3. The right side displays the resource usage status of the I area, Q area, and program block sizes.



4. Hover the mouse over a variable to view the variable name. Right-click a variable and select "Go to definition" to jump to the declaration location of this variable; select "Cross-reference table" to view the usage information of this variable.

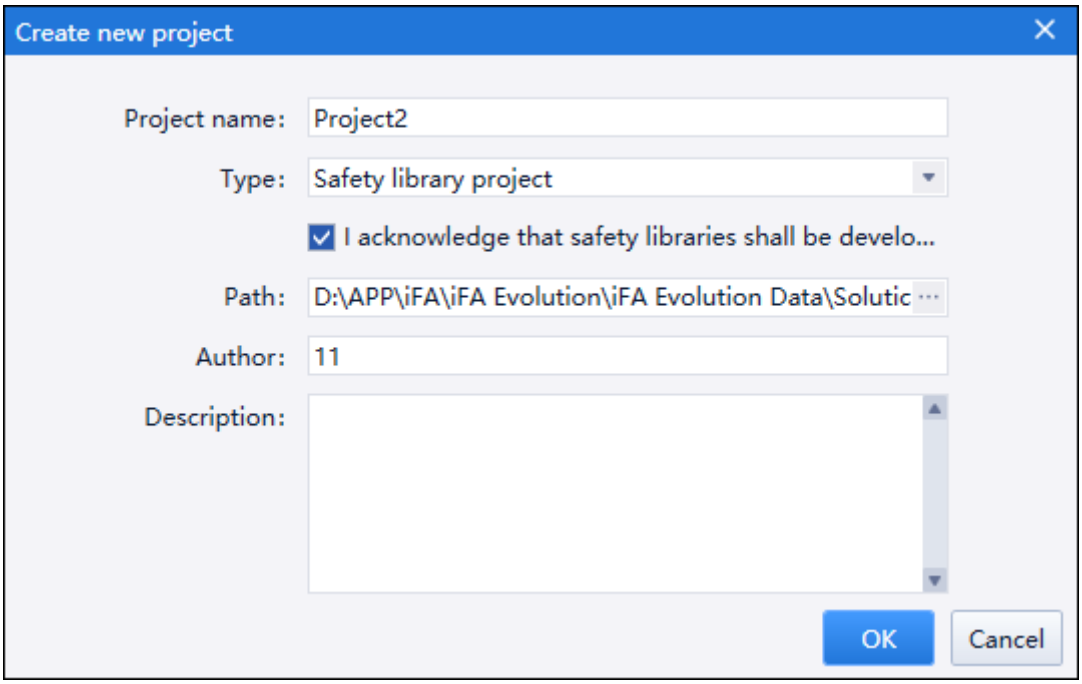
11.2.1.3 Variable Operations

The variable operations in safety PLCs are the same as those in standard PLCs. For details, see 10.1.1.10 Variable Operationsin the "PLC Programming" chapter.

11.2.2 Library Reference

The library reference operations in safety PLCs are the same as those in standard PLCs. For details, see 12.3 Library Reference in the "PLC Programming" chapter.

The safety PLC supports the use of safety libraries. After selecting a safety library project, the user must check the risk acknowledgement alarm before clicking "OK".



11.2.3 Program Block

This section only covers the functional differences between the program blocks in safety PLCs and standard PLCs; for other general functions, see 10.1.2 Program Block in the "PLC Programming" chapter.

11.2.3.1 Program

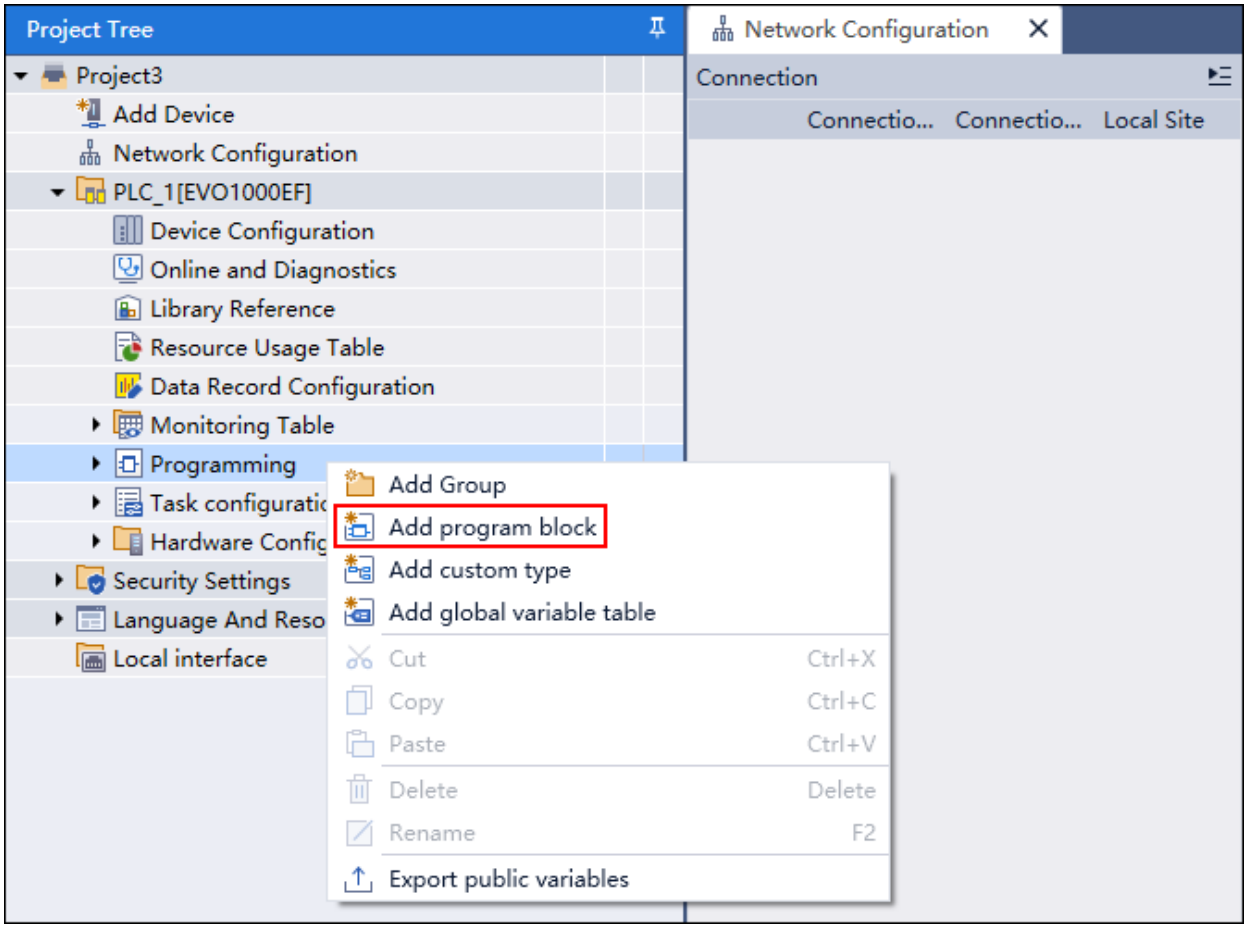
11.2.3.1.1 Adding a User Program

Prerequisites

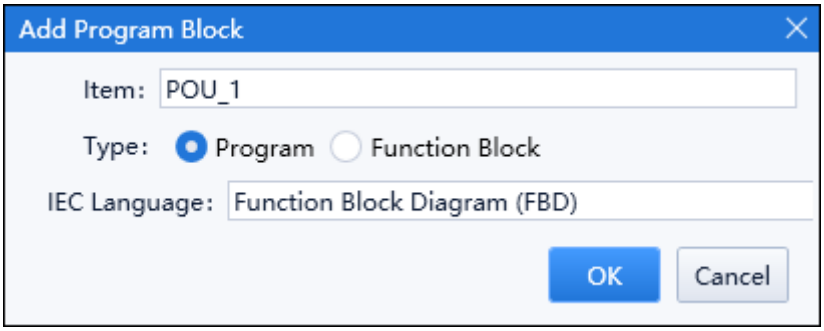
- A project has been created.
- A device has been added.

Procedure

1. Right-click "Programming > Add program block".



2. In the "Add program block" window, set the program name, select "Program" as the type, and select the corresponding IEC language, as shown in the figure below.



Property	Description
Name	Indicates the program name. It supports letters, numbers, and underscores (_), and must start with a letter or underscore. Underscores cannot be used consecutively or at the end. The program name is case-insensitive, with a maximum length of 128 bytes. For example, __POU, PO__U, and POU_ are invalid names, while P_O_U is considered valid.
Type	Indicates the program block type. To add a user program, select "Program" here.
IEC Language	Supports LD and FBD programming languages.

3. Click "OK" to add the user program POU. The user program has a "(PRG)" suffix. When a user program is added, iFA will automatically open its editor. The editor is divided into the variable declaration area and the program editing area, which can be collapsed and expanded using and .

NOTE

When a POU is being compiled, the system will check the upper limit based on device requirements. For specific limitations, see the device specifications.

11.2.3.1.2 Calling a User Program

Before a user program is downloaded to the controller, it needs to be called first; otherwise, the program will not be executed.

User program calls are mainly managed in "Task configuration". For details, see 11.4.6 Task Configuration.

NOTE

If a user program is added without being called, and the compiler detects that the program has not changed (using the incremental download mechanism), the program download will not be triggered.

11.2.3.2 Function Block

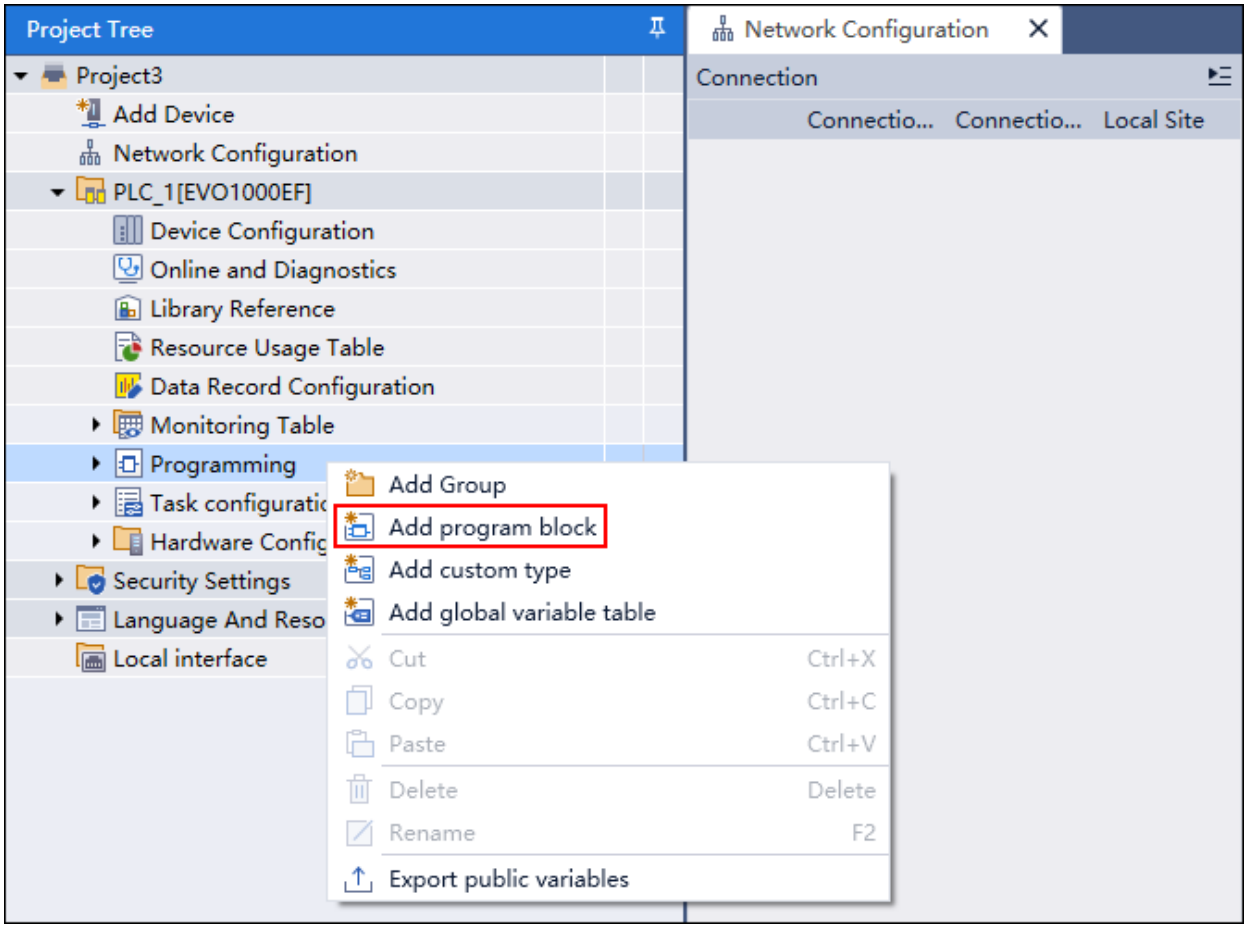
11.2.3.2.1 Adding a User Function Block

Prerequisites

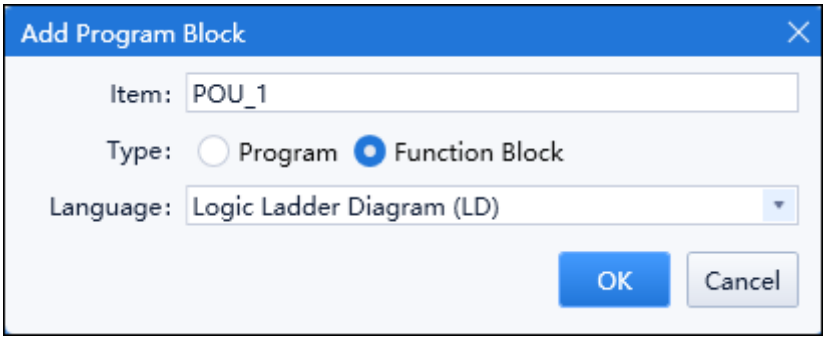
- A project has been created.
- A device has been added.

Procedure

1. Right-click "Programming > Add program block".



2. In the "Add program block" window, set the function block name, select "Function block" as the type, and select the corresponding IEC language, as shown in the figure below.



Property	Description
Name	Indicates the name of the function block. It supports letters, numbers, and underscores (_), and must start with a letter or underscore. Underscores cannot be used consecutively or at the end. The program name is case-insensitive, with a maximum length of 128 bytes. For example, __POU, PO__U, and POU_ are invalid names, while P_O_U is considered valid.
Type	Indicates the program block type. To add a user function, select "Function block" here.
IEC Language	Supports LD and FBD programming languages.

3. Click "OK" to add the user function block POU. The user function block has a "(FB)" suffix. When a user function block is added, iFA will automatically open its editor. The editor is divided into the variable declaration area and the program editing area, which can be collapsed and expanded using and .

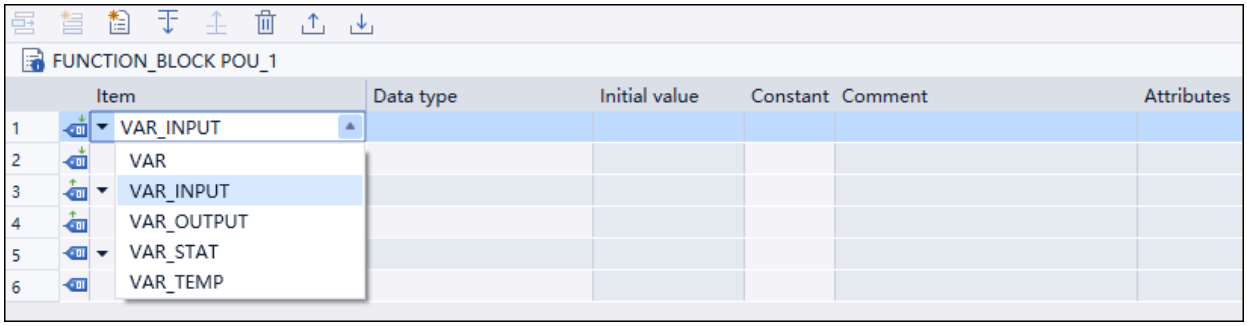
NOTE

When a POU is being compiled, the system will check the upper limit based on device requirements. For specific limitations, see the device specifications.

11.2.3.2.2 Using a User Function Block

Variable declaration in function blocks

The variable types that can be declared in FB include local variables (VAR), input/output variables (VAR_IN_OUT), input variables (VAR_INPUT), output variables (VAR_OUTPUT), static variables (VAR_STAT), and temporary variables (VAR_TEMP). All properties of the variables are configurable.



Instantiation of function blocks

A function block is an abstract structure definition in the IEC61131-3 standard and must be instantiated to become an executable data entity. An uninstantiated function block cannot be called or executed by a program. An instantiated function block has the following characteristics:

- Possesses private data storage space.
- Executes specific functions according to predefined logic.
- Equipped with fully encapsulated and independent structural variables.

Through the instantiation process, the function block is transformed from a type definition into a specific data entity, thereby achieving its designed functionality.

The instantiation of a function block is like defining a variable. When the program needs to call this function block, it only needs to declare a variable of the FB type. The program reads and writes to the function block through this variable during execution.



A POU can only call the same FB instance once.

11.2.4 Safety LD Language Programming

11.2.4.1 Overview

A ladder diagram is a graphical programming language.

It features two vertical power rails on its left and right sides. The left power rail provides a power flow that passes through various contacts, functions, function blocks, coils, and other components along horizontal rungs from left to right. The right power rail serves as the endpoint of the power flow.

Each contact represents the status of a Boolean variable, each coil represents the status of an actual device, and functions or function blocks correspond to the standard libraries in IEC 61131-3 or user-created functions or function blocks.

In ladder diagram programming, although relay components such as coils and contacts are used, they are not actual physical relays but rather storage units (soft relays) within the PLC. Each soft relay corresponds to a storage unit in the PLC's memory mapped register, and its status (TRUE/FALSE) determines the logical behavior of the soft relay:

- TRUE (ON) status: When the storage unit is in TRUE status, the corresponding soft relay coil is "energized", with its normally open contacts closed and its normally closed contacts opened.
- FALSE (OFF) status: When the storage unit is in FALSE status, the soft relay coil is "de-energized", with its normally open contacts remain opened, and its normally closed contacts remain closed.

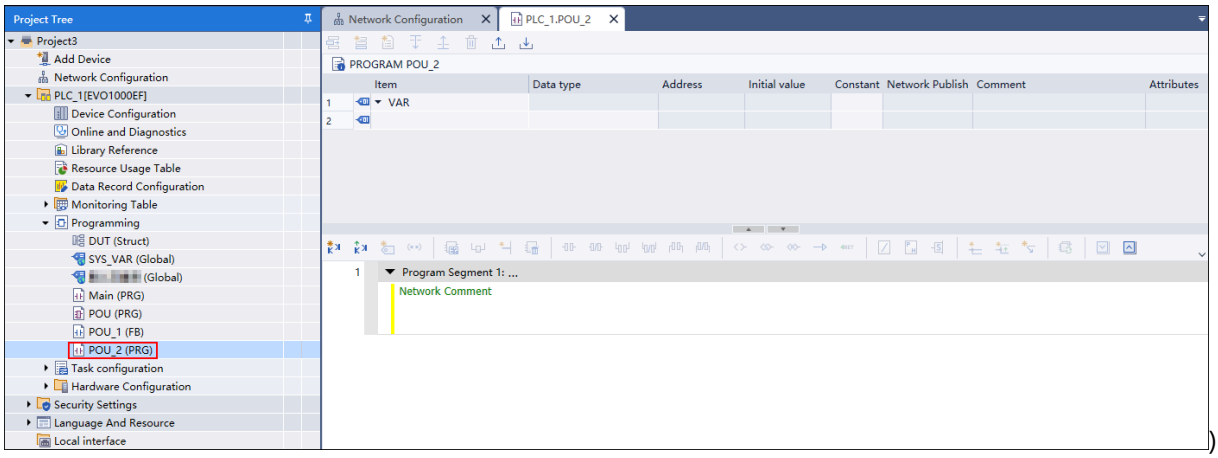
Since these components exist only in the program, they are typically referred to as programming elements, not actual physical relays.

11.2.4.2 Opening the Safety LD Editor

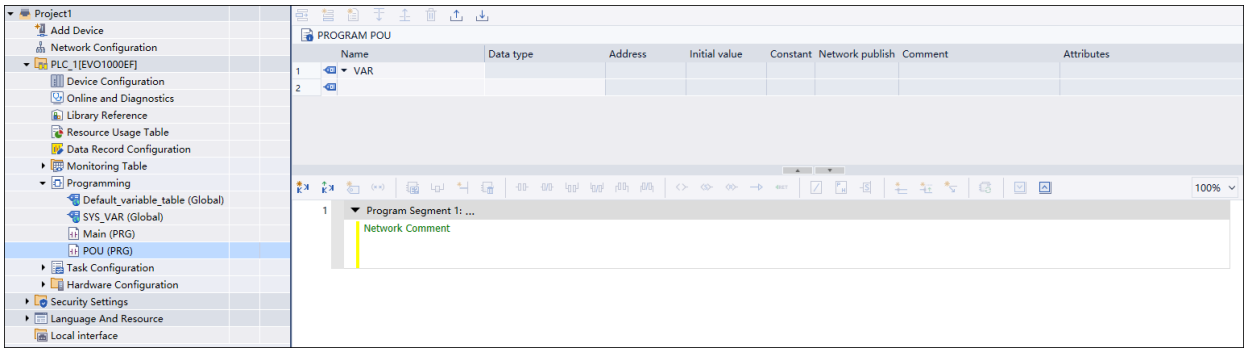
Prerequisites

A safety LD program block has been added.

Procedure

- Double-click a safety LD language program block under the "Programming" node (node identifier: 

) to open the safety LD editor.
- Right-click a safety LD language program block under the "Programming" node and select "Open" to open the safety LD editor.



The safety LD editor is divided into the variable declaration area and the program editing area, which can be collapsed and expanded using and .

11.2.4.3 Setting the Safety LD Editor

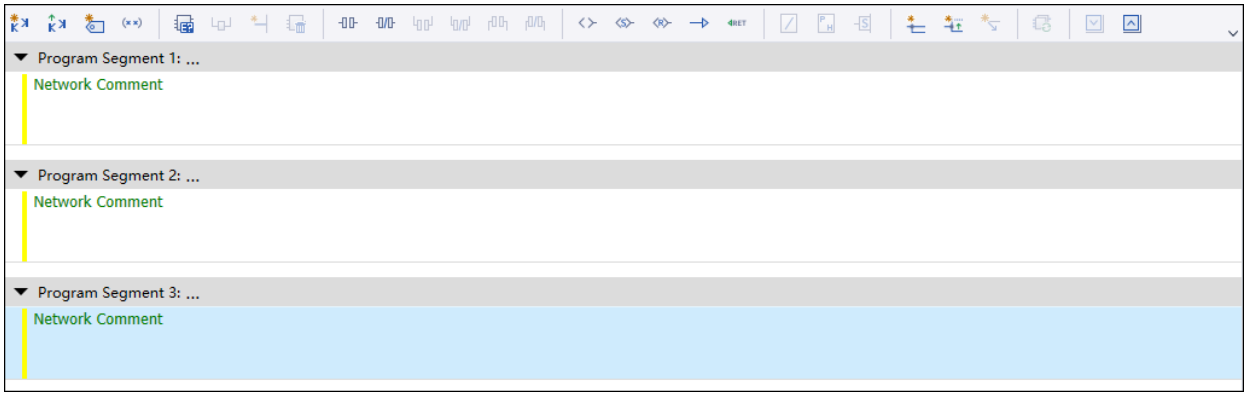
The method and standard for setting the safety LD editor are the same as those for the standard PLC. For details, see 10.1.3.3.2 LD Editor Settings.

11.2.4.4 Safety LD Elements

Main elements

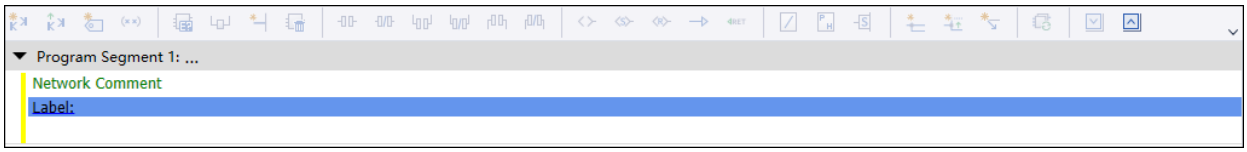
- **Network**

  Each network element contains a program, and the graphical configuration result of the entire network is also a program. Although the entire user program can be implemented within a single network, this is not recommended. To comply with programming standards and facilitate program inspection, it is recommended to split the user program into different networks by function, with each network implementing a specific function. This not only improves programming efficiency but also enhances system stability and scalability.



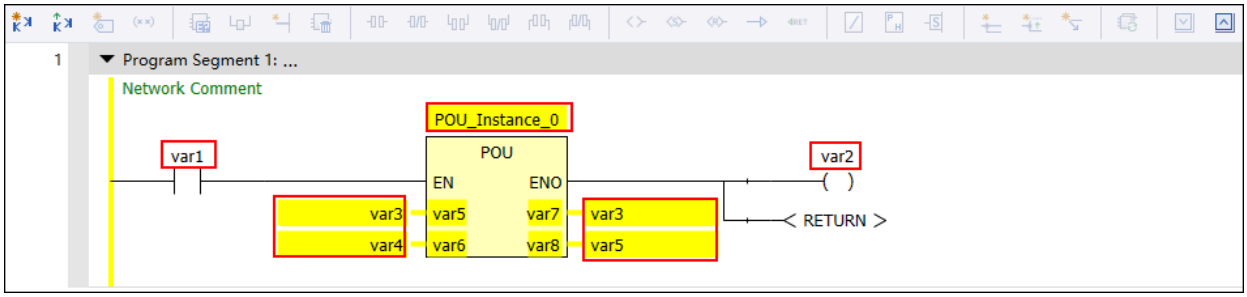
- **Label**

  Labels are used to indicate the specific location to which the program jumps.



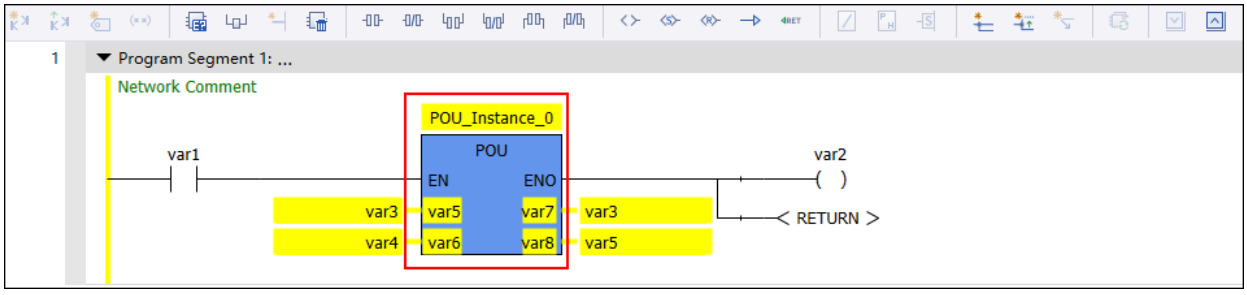
- **Operand**

  Operands are the editable text data, such as variable text corresponding to contacts, coils, block input and output pins, type text of the block, and instance text of the block. Users can change operands to target variables as needed.



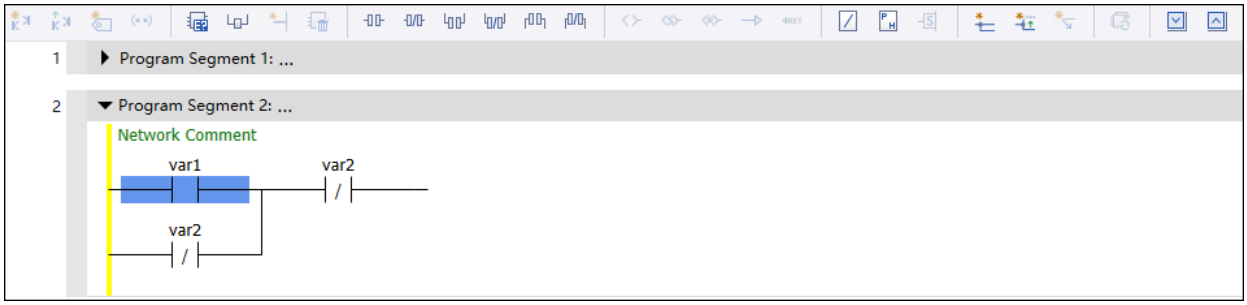
- **Block**

  A block is a packaging of functionality, used to simplify program complexity. It is recommended to encapsulate major functions into FBs and call them in the program. This not only reduces the reading difficulty of the main program but also improves the reusability of functions.



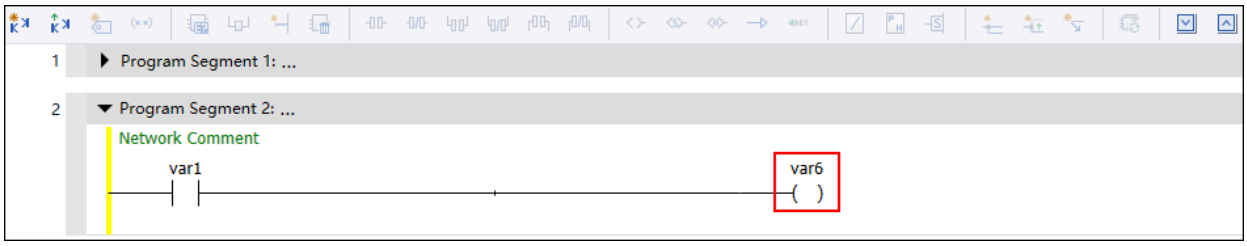
- **Contact**

  A contact is an element controlled by a BOOL or SAFEBOOL variable, determining whether the logic proceeds to the right. When the variable corresponding to the contact is TRUE, the program proceeds to the right; otherwise, execution is interrupted at the contact.



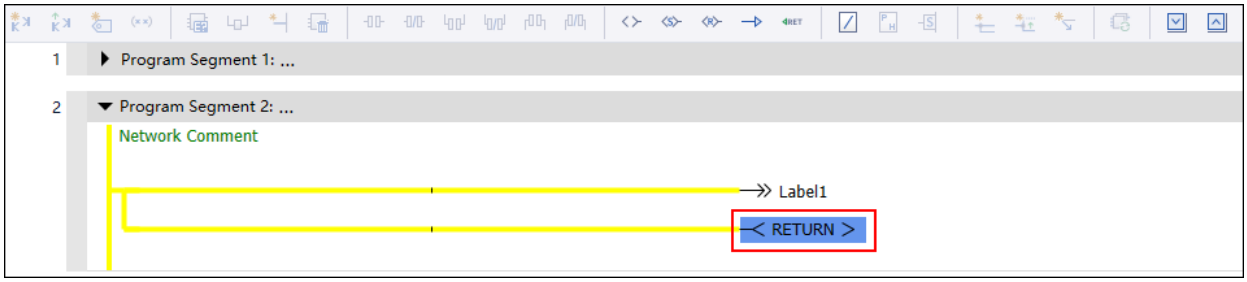
- **Coil**

  A coil is an element used to receive data from the previous configuration result. It assigns the configuration result from the left to the coil variable, achieving data transfer and assignment.



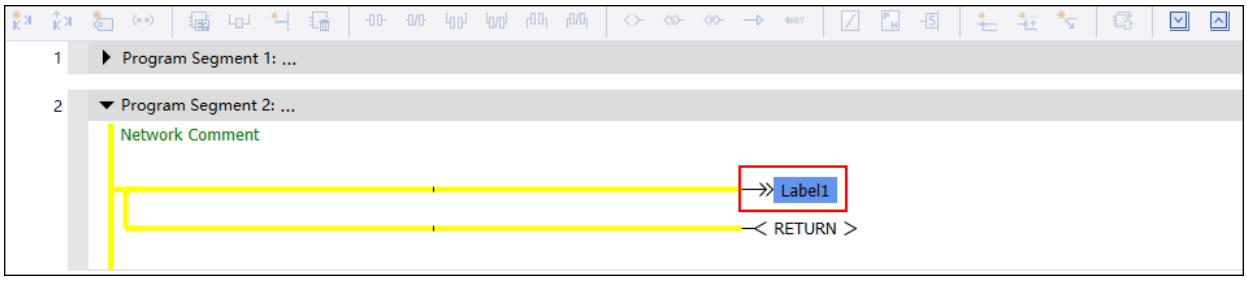
- **Return**

  The return element is used to terminate program execution. Once a return element is encountered, all subsequent program code will no longer be executed.



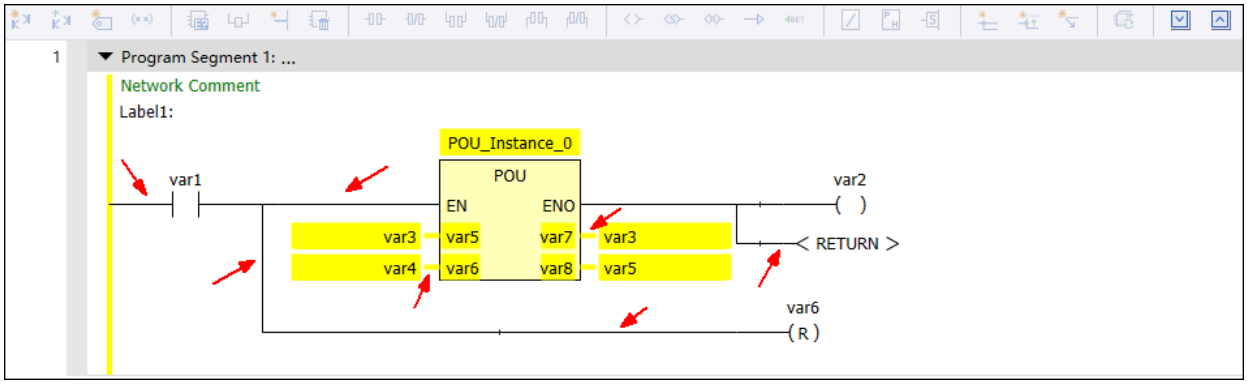
- **Jump**

  The jump element is used to transfer program execution from the current position to another specified location.



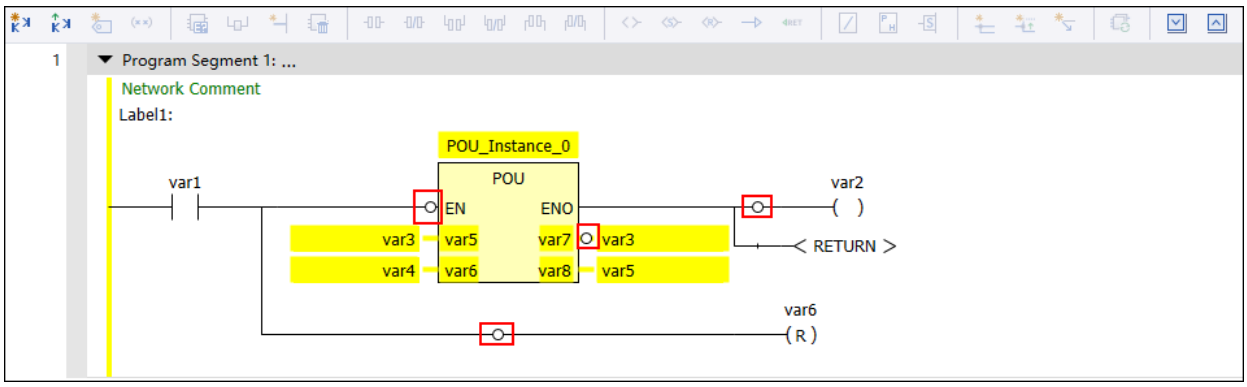
- **Connection**

  A connection is a line that connects two positions. Connection lines mainly include contact connections, input/output connections of operation blocks, connections between coils, terminal connections, branch terminal connections, and vertical connections.

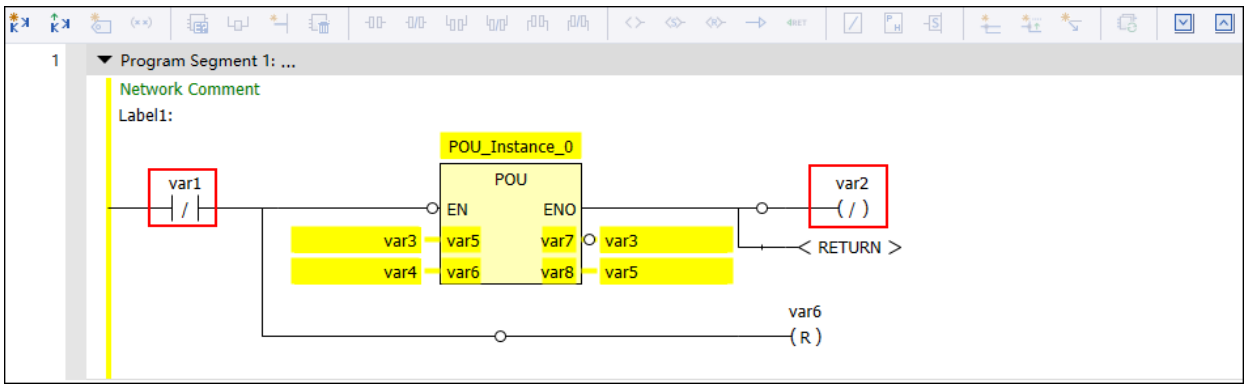


Auxiliary element

- **Inversion**
  - Supports inverting the block pin of BOOL or SAFEBOOL type.

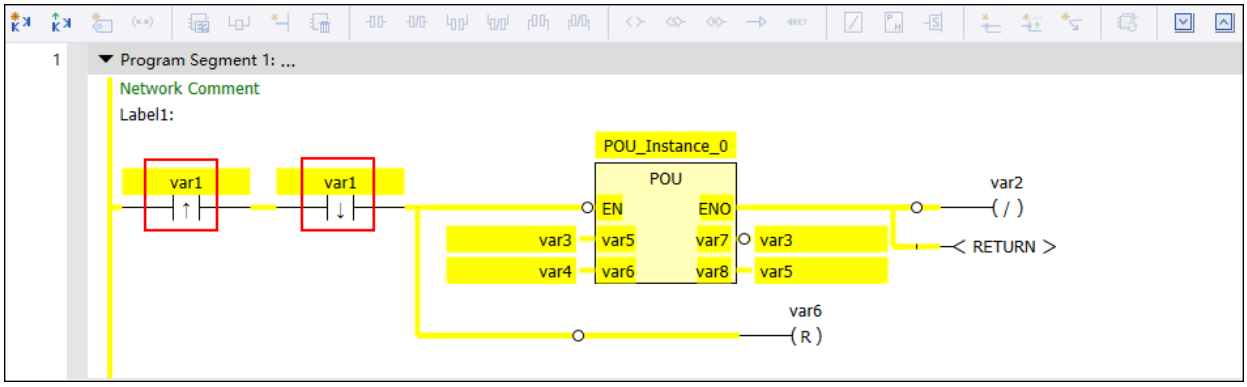


  - Supports inverting contacts and coils.



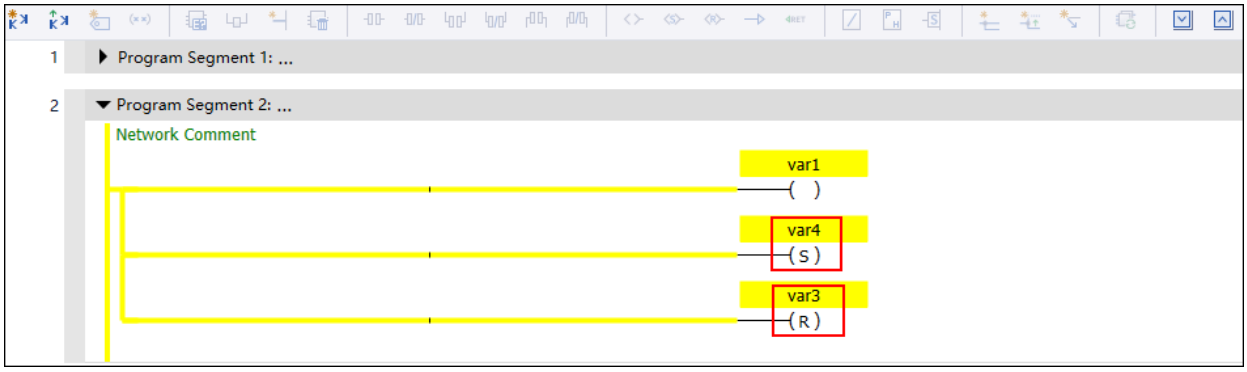
- **Edge detection**

  Supports setting rising/falling edges for contacts.



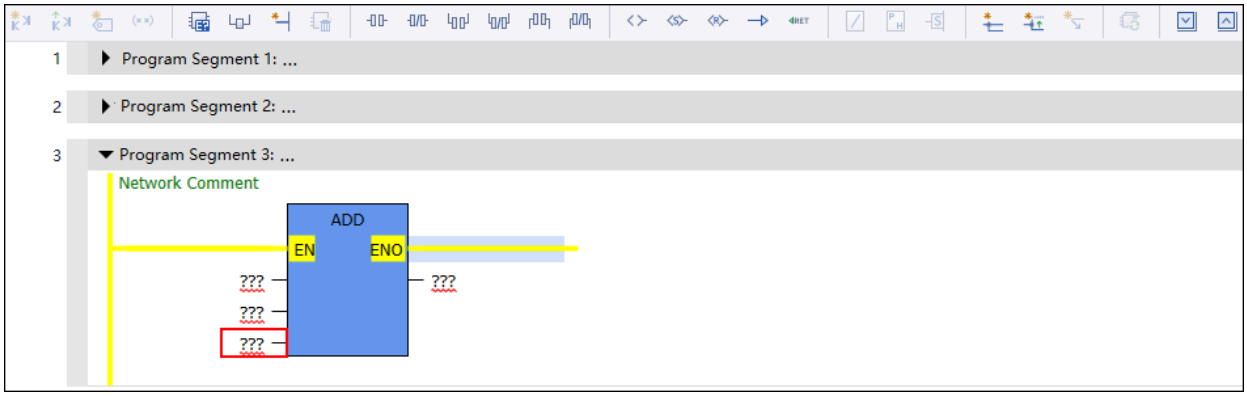
- **Set/Reset**

  Supports set and reset operations for coils.



- **Input pin**

  For blocks supporting multiple inputs, input pins can be added.



11.2.4.5 Safety LD Programming

Safety LD allows users to add required elements to the network using commands from the toolbar. It also supports directly dragging user-defined program blocks from the toolbox and dropping them into the network to create corresponding block instances.

Adding elements by commands

Elements that can be added via commands include: network, label, block, input pin, contact, coil, return, jump, inversion, edge detection, set/reset, and branch.

- Commands in the toolbar



  Commands in the toolbar are fixed. Depending on the selected element, different commands are enabled. Only enabled commands support adding corresponding elements around the selected element, which cannot be achieved by disabled commands by default.

- Commands in the right-click menu

  Commands in the right-click menu vary depending on the selected element. Only the commands that are visible for the selected element can be used to add corresponding elements around it.

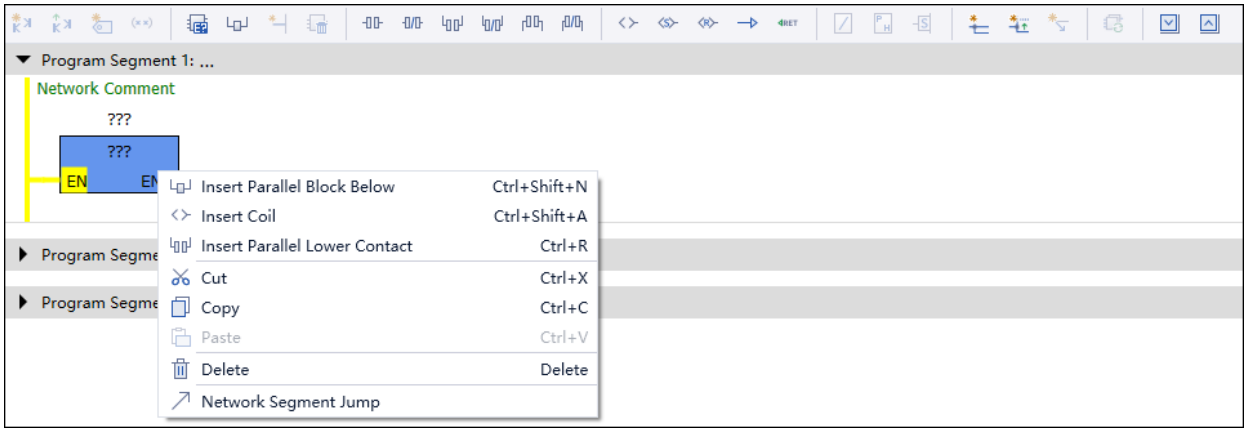


Table11-2-4-5-1 Icons and descriptions of commands

Type	Icon	Description
Add/Insert network		Adds/Inserts a network.
Insert label		Adds a label to network.
Toggle network comment		Comments or uncomments the selected network.
Insert EN/ENO empty block		-
Insert parallel block below		-
Add input		-
Remove unused pins		Deletes unconfigured pins for FB call blocks.
Insert an NO contact		Inserts a normally open, normally closed, or upper/lower normally open/closed contact.
Insert a coil		Inserts a coil, set coil, or reset coil.
Insert jump		-
Insert return		-
Inversion		Inverts a contact or a BOOL connection.
Edge detection		Sets edge detection for a contact.
Set/Reset		Performs set/reset operation for a coil.
Insert a branch		Inserts a branch above or below.
Set branch start point		-
Update all parameters		When parameters are changed at the FB declaration but not at the call site, this icon can update the block at the call site.

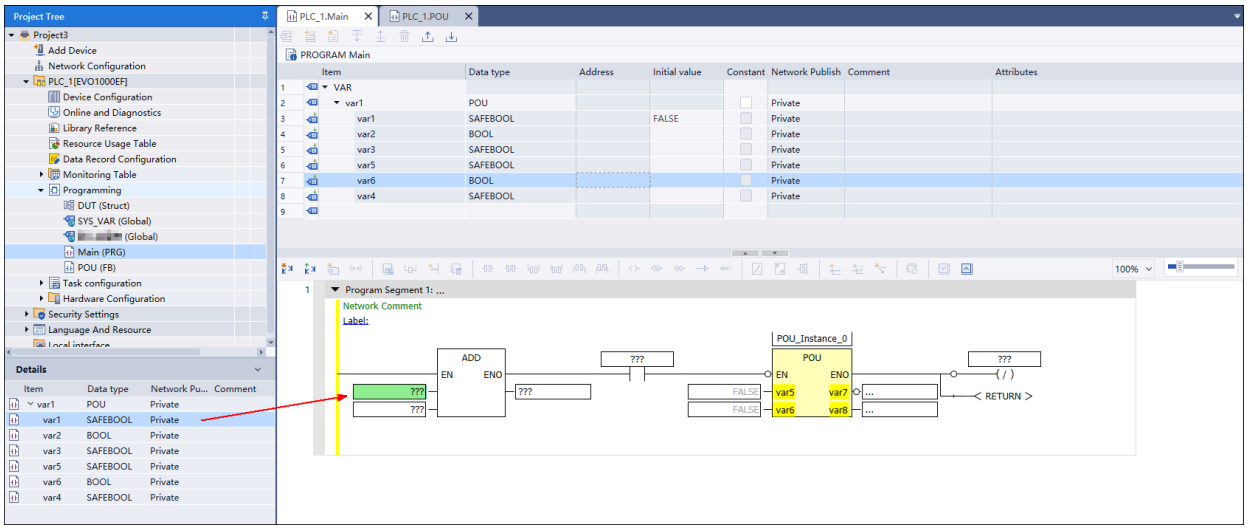
Type	Icon	Description
Expand all networks		-
Collapse all networks		-

Adding elements by dragging

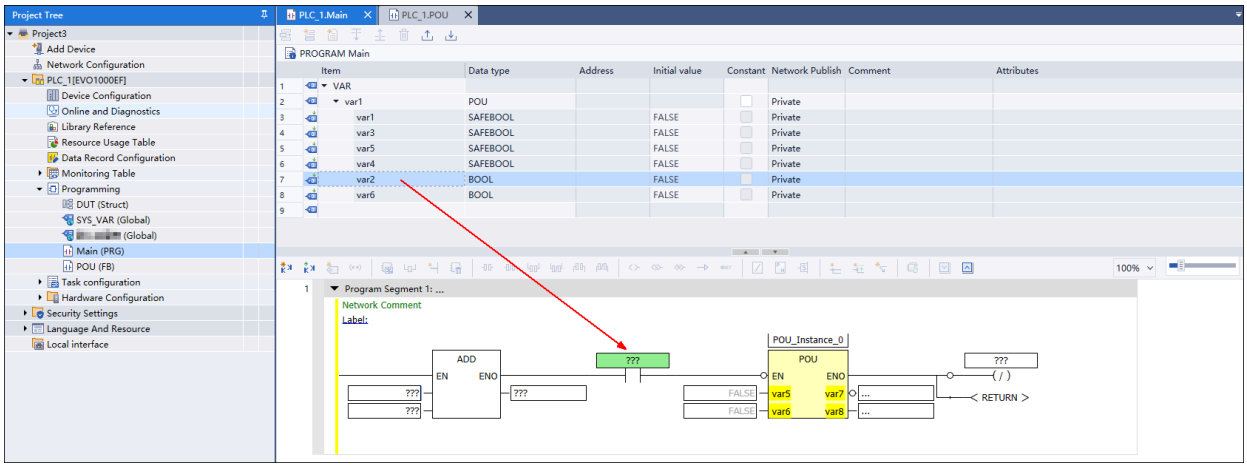
Instance variables from the "Details" window and the variable declaration area can be dragged to block input/output pins, contact, or coil variables for addition or modification. Blocks from the project tree or toolbox can be dragged to the editor for addition. Elements within the editor to another location in the editor. Currently, variables within the editor cannot be dragged.

- Dragging variables

  Drag instance variables from the "Details" window to the operand variable locations in the editor.

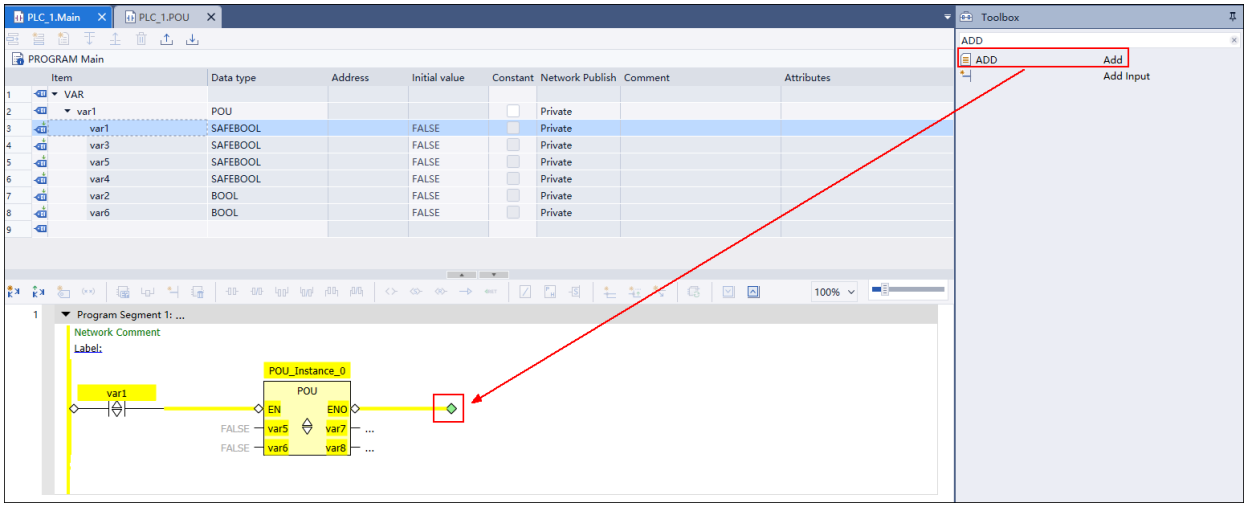


  Drag variables from the variable declaration area to the operand variable locations in the editor.



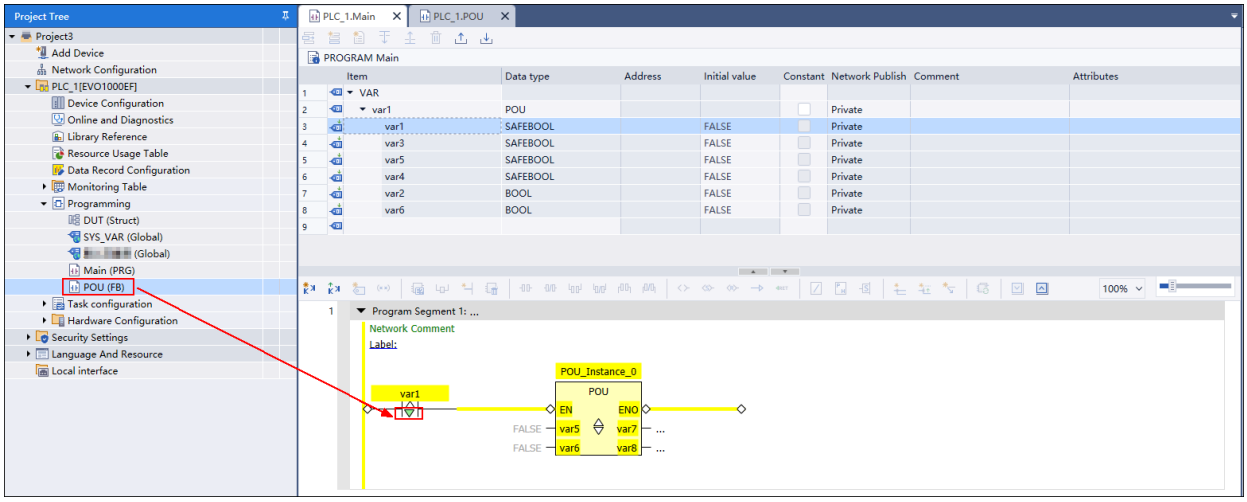
- Dragging elements from the toolbox

  When dragging an element from the toolbox to the editor, the available locations are highlighted with a diamond-shaped mark. Hover over these locations and they will be displayed in green. Release the mouse to add the selected element to the corresponding location.



- Dragging elements from the project tree

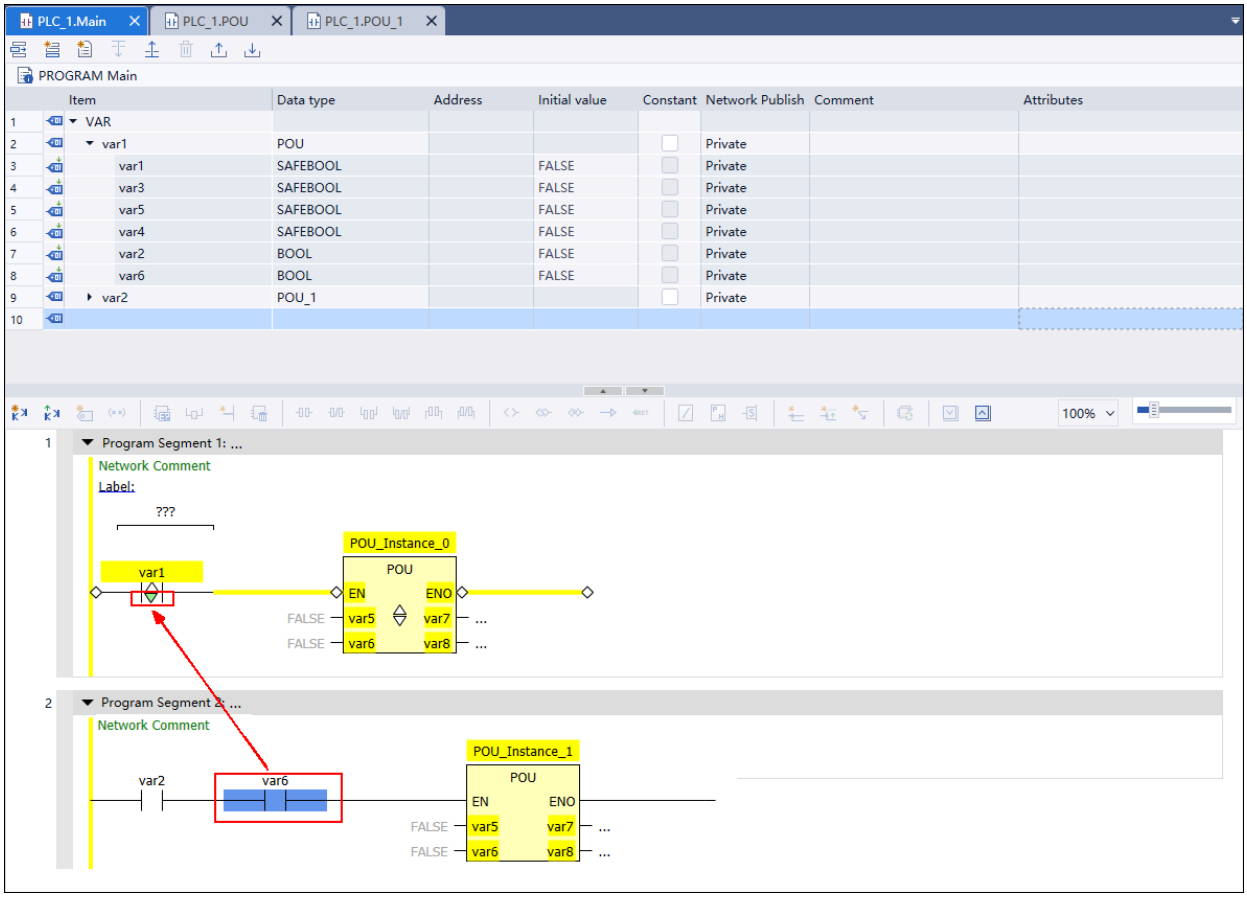
  When dragging an element from the project tree to the editor, the available locations are highlighted with a diamond-shaped mark. Hover over these locations and they will be displayed in green. Release the mouse to add the selected element to the corresponding location.



- Dragging elements in the editor

  When dragging an element in the editor to another network or another location within the same network, the available locations in the target network are highlighted with a diamond-shaped mark. Hover over these locations and they will be displayed in green. Release the mouse to move the selected element to the corresponding location.

  If you keep pressing Ctrl during dragging and dropping, the element at the source location will not be deleted and a new element will be added at the target location after. If Ctrl is not pressed during dragging and dropping, the element at the source location will be deleted.



11.2.4.6 Graphic Configuration Modification

11.2.4.6.1 Modifying Text Data

After adding a network, contact, coil, or block, the system provides corresponding text data for them:

- For networks: Includes network title and comment.
- For contacts and coils: Includes variable text.
- For blocks: Includes block type, block instance, and block input/output text.

Users can modify these text data as follows:

- For network titles and labels, double-click the title or label to modify the text.

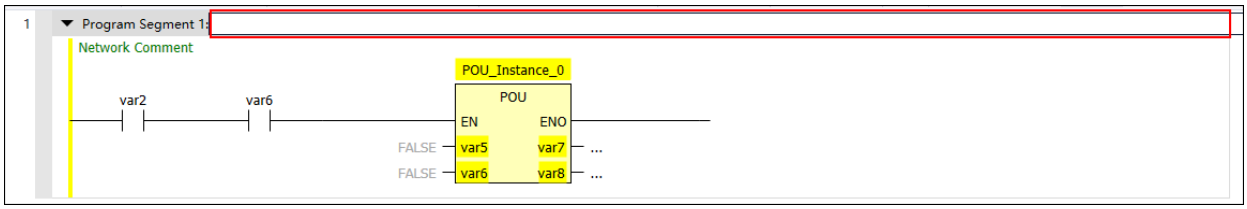


Figure 11-2-4-6-1-1 Modifying network title

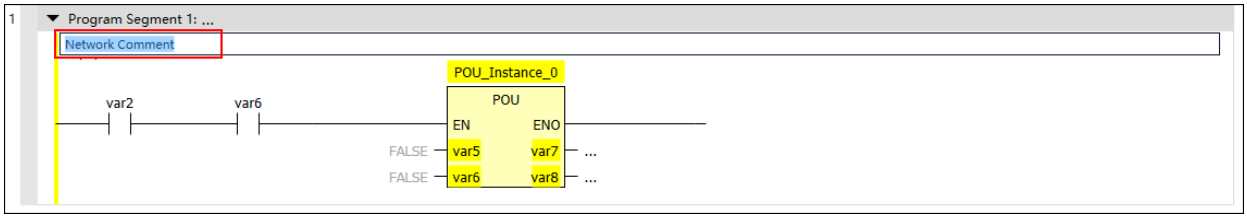


Figure 11-2-4-6-1-2 Modifying network label

- For text of contacts, coils, and blocks, drag a variable to the corresponding location or double-click the text for modification. After modification, the text will be successfully updated.

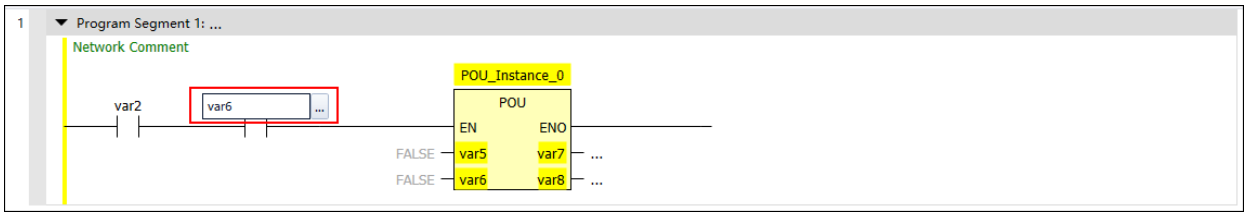


Figure 11-2-4-6-1-3 Modifying contact text

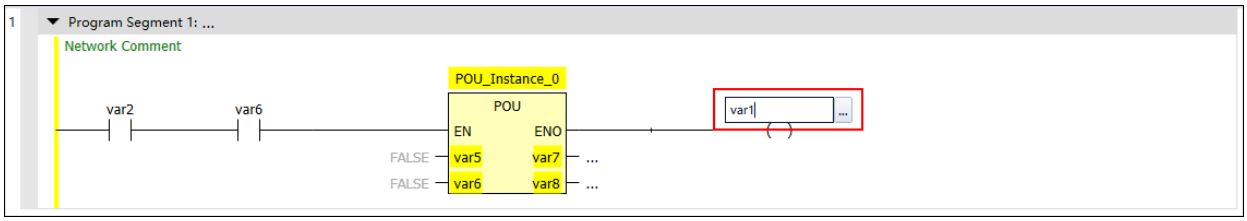


Figure 11-2-4-6-1-4 Modifying coil text

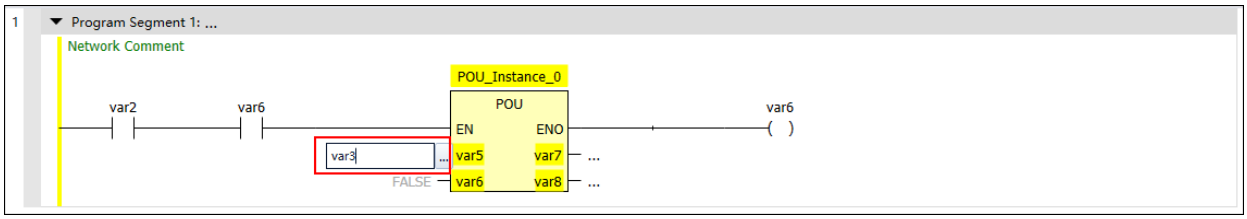
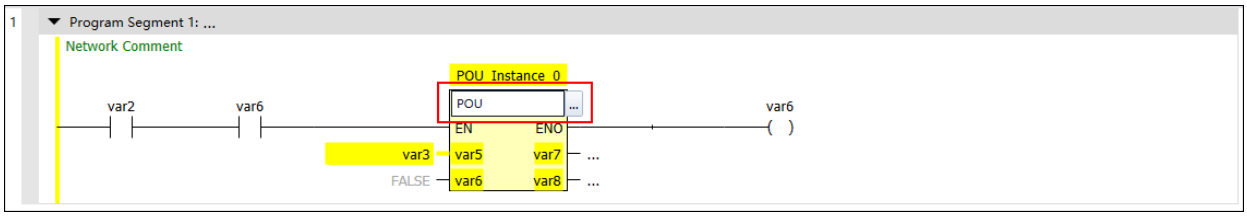


Figure 11-2-4-6-1-5 Modifying block text

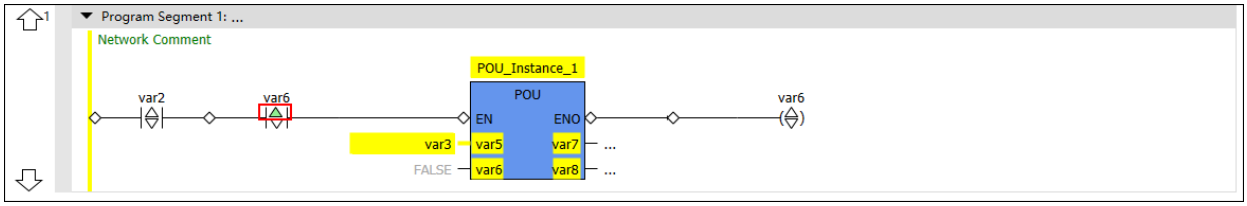
11.2.4.6.2 Modifying Block Type

Double-click a block type to modify its type.



11.2.4.6.3 Moving an Element

To move an element in the editor, you can delete it and then add it to the target position, or directly drag it to the target position. As shown in the figure, a block POU can be dragged to another location.



11.2.5 Safety FBD Language Programming

11.2.5.1 Overview

Function Block Diagram (FBD) is a widely used and intuitive graphical programming language. It uses graphical symbols (such as boxes and lines) to represent program logics, similar to electronic circuit diagrams.

The basic building blocks of a program are function blocks. Each function block represents a predefined operation or function, such as logical operations (AND, OR, NOT), mathematical operations (ADD, SUB, MUL, DIV), timers (TON, TOF), counters (CTU, CTD), comparators (GT, LT, EQ), and user-defined complex functions.

Function blocks have input terminals (usually on the left) and output terminals (usually on the right). The logic of program execution is represented by connection lines, which connect the output of one function block to the input of another, clearly illustrating the direction of data flow (from left to right or from top to bottom). The execution order of the program is primarily determined by data flow.

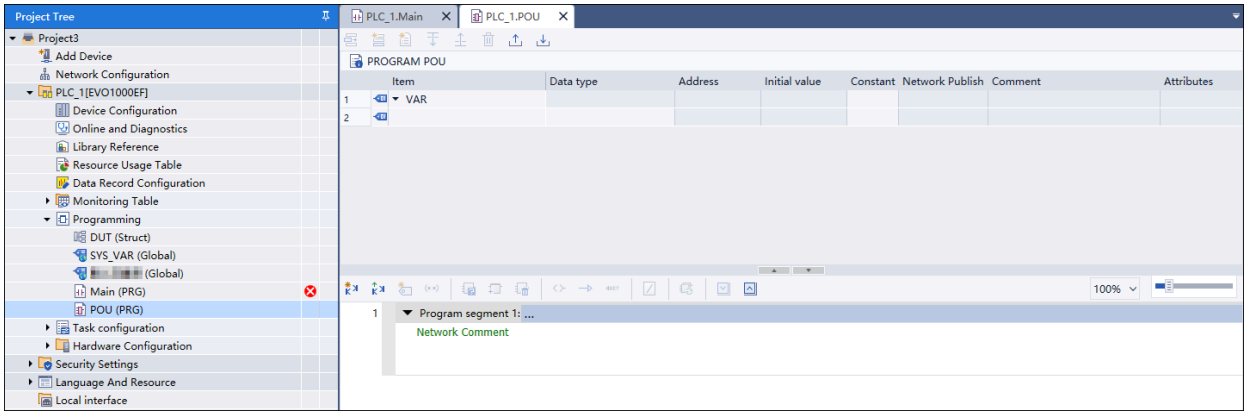
11.2.5.2 Opening the Safety FBD Editor

Prerequisites

The safety FBD program block has been added.

Procedure

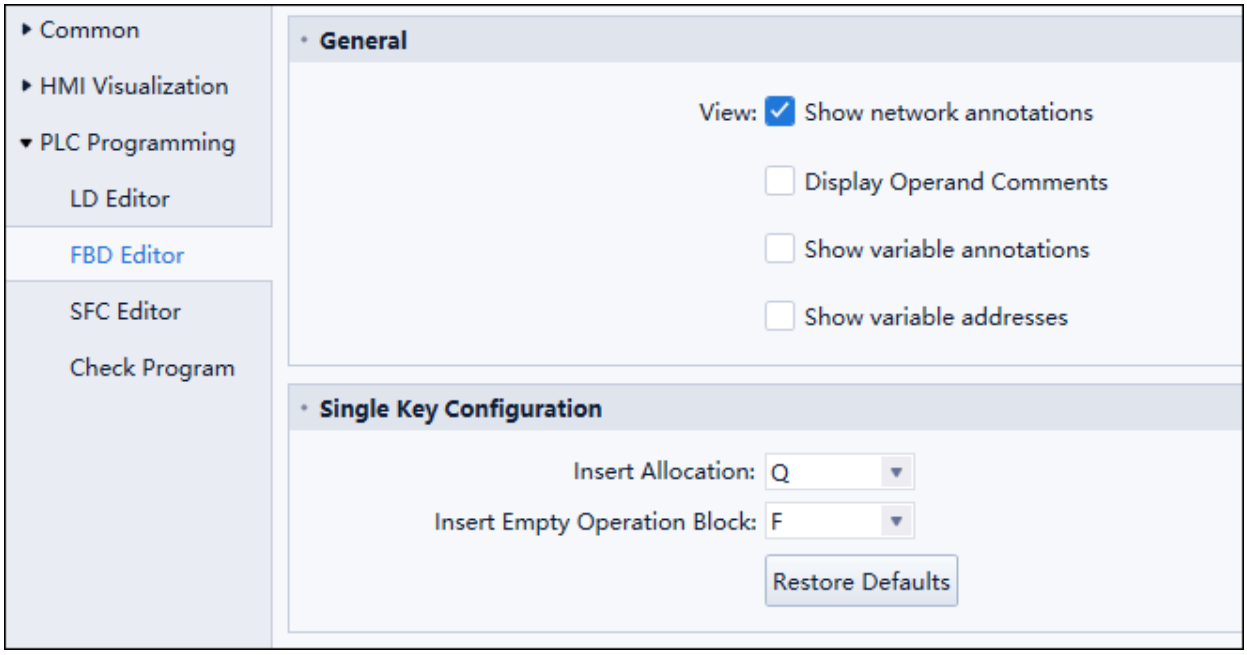
- Double-click a safety FBD language program block under the "Programming" node (node identifier: ) to open the safety FBD editor.
- Right-click a safety FBD language program block under the "Programming" node and select "Open" to open the safety FBD editor.



The safety FBD editor is divided into the variable declaration area and the program editing area, which can be collapsed and expanded using and .

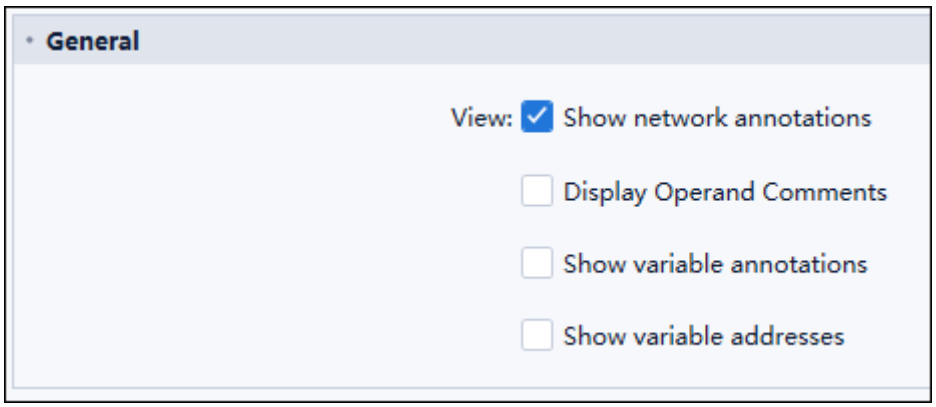
11.2.5.3 Setting the Safety FBD Editor

In the software menu, select "Tools > Options > FBD editor".

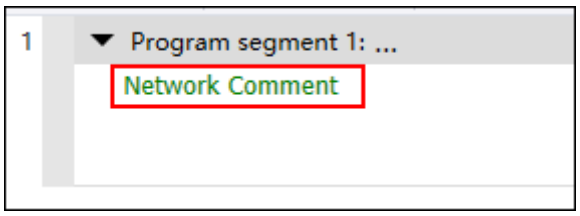


General

Sets the view range of the information in the FBD editor.

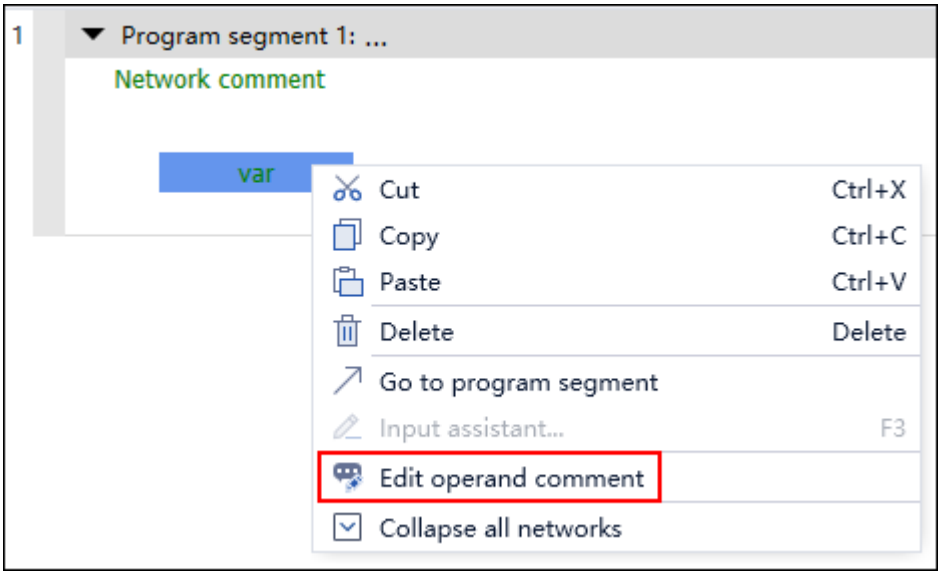


- Show network annotations: Sets the visibility of network annotations.

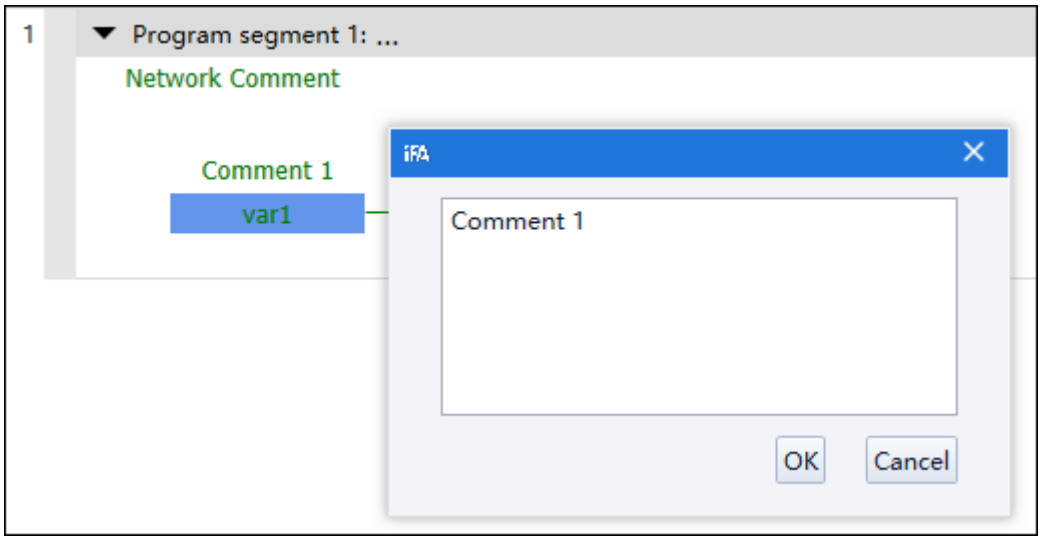


- Show operand annotations: Sets the visibility of operand annotations.

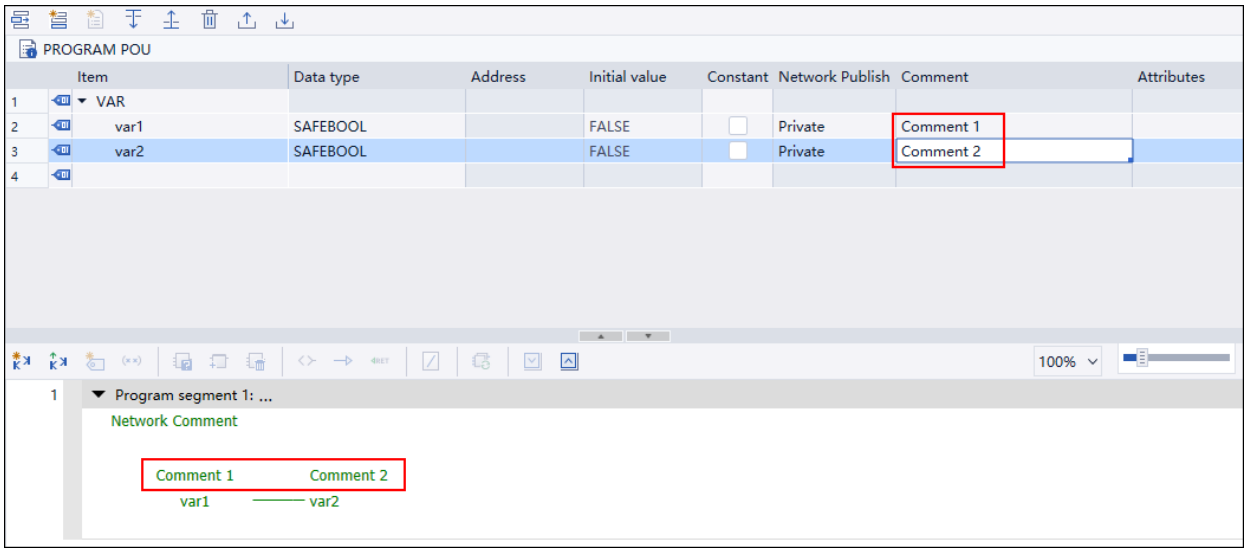
  1. After checking "Display operand comment", right-click an operand in the editor and select "Edit operand comment".



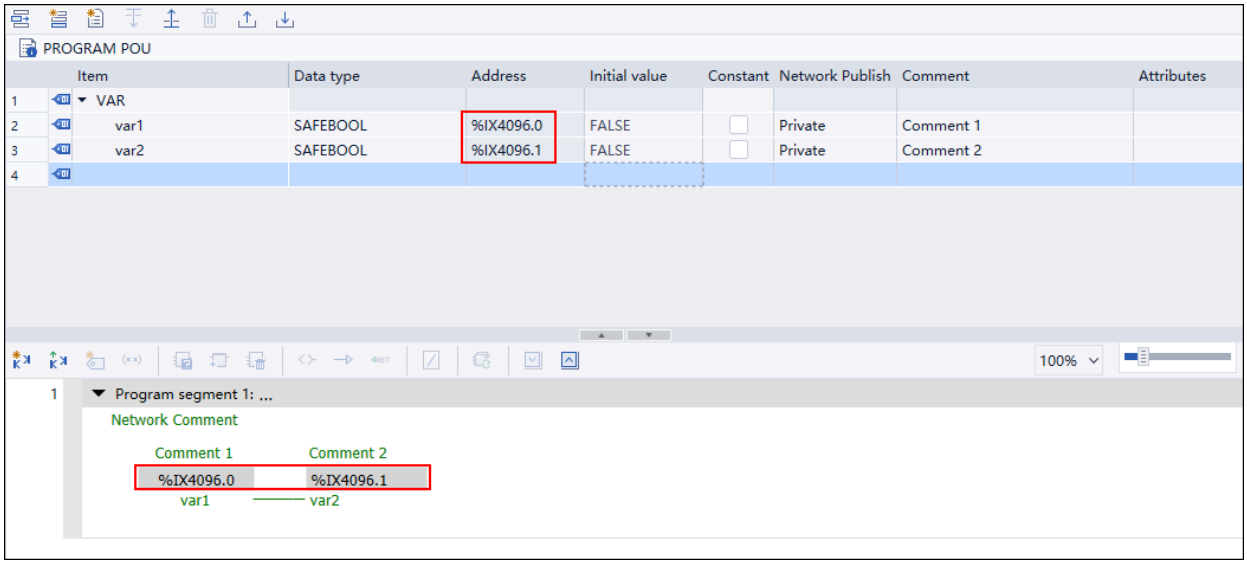
2. The entered annotation will be displayed above the operand. If "Display operand comment" is unchecked, the annotation will not be displayed.



- Show variable annotations: Sets the visibility of variable annotations.

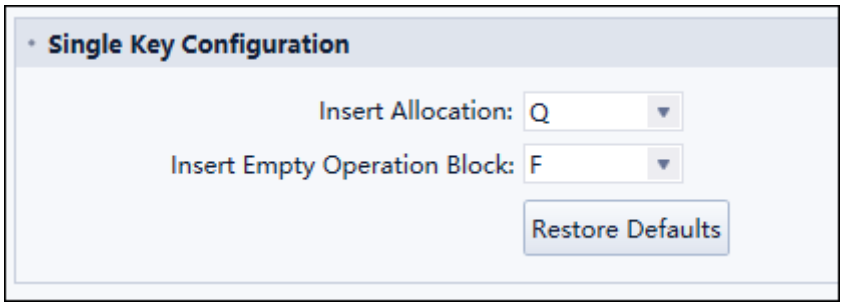


- Show variable addresses: Sets the visibility of variable addresses.



Single key configuration

The items in the "Single key configuration" section sets commonly used shortcut keys for FBD program editing, selected from drop-down lists. Click "Restore defaults" to reset all shortcut keys to their default values.

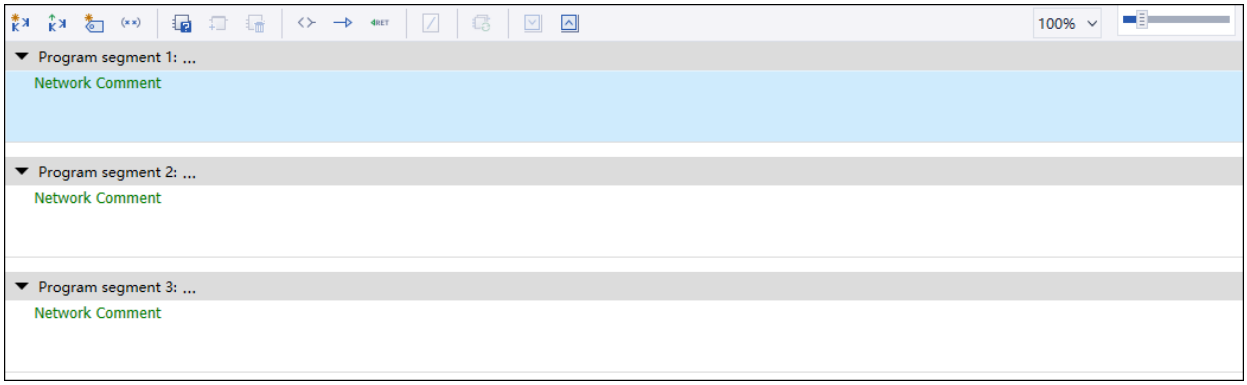


11.2.5.4 Safety FBD Elements

Main elements

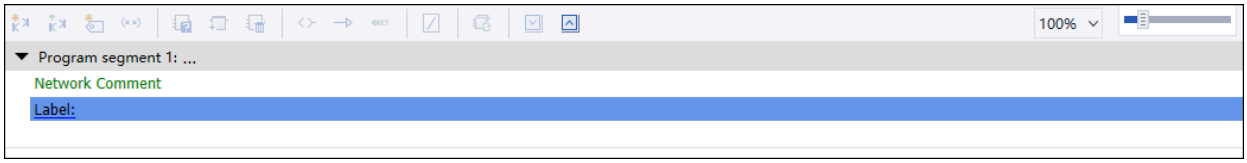
- **Network**

  Each network element contains a program, and the graphical configuration result of the entire network is also a program. Although the entire user program can be implemented within a single network, this is not recommended. To comply with programming standards and facilitate program inspection, it is recommended to split the user program into different networks by function, with each network implementing a specific function. This not only improves programming efficiency but also enhances system stability and scalability.



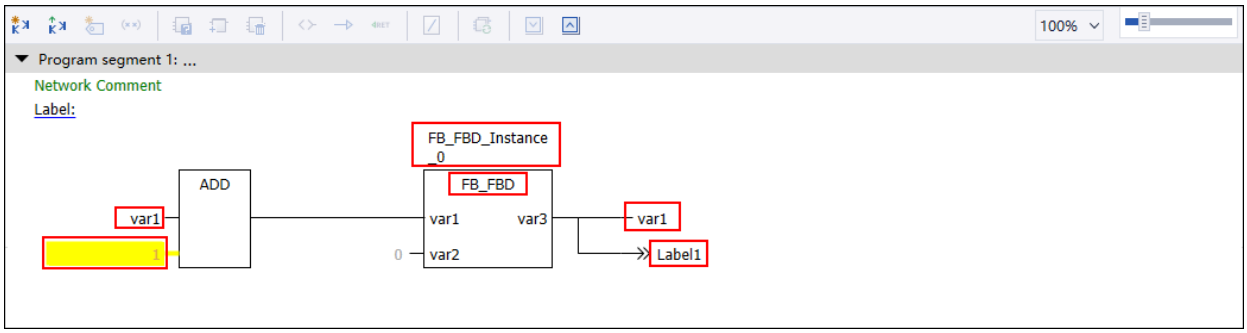
- **Label**

Labels are used to indicate the specific location to which the program jumps.



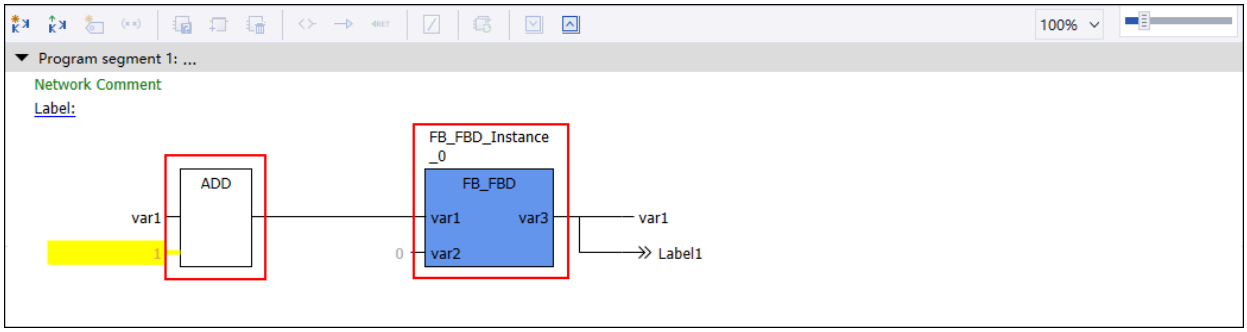
- **Operands**

  Operands are the editable text data, such as variable text corresponding to block input and output pins, type text of the block, and instance text of the block. These texts are all editable, allowing users to change them to target variables as needed.



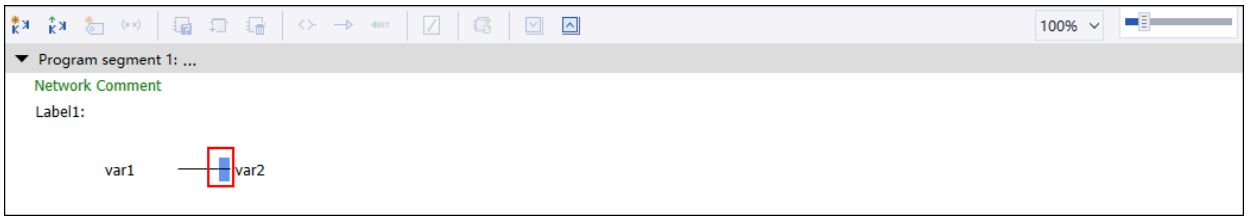
- **Block**

  A block is a packaging of functionality, used to simplify program complexity. It is recommended to encapsulate major functions into FBs and call them in the program. This not only reduces the reading difficulty of the main program but also improves the reusability of functions.



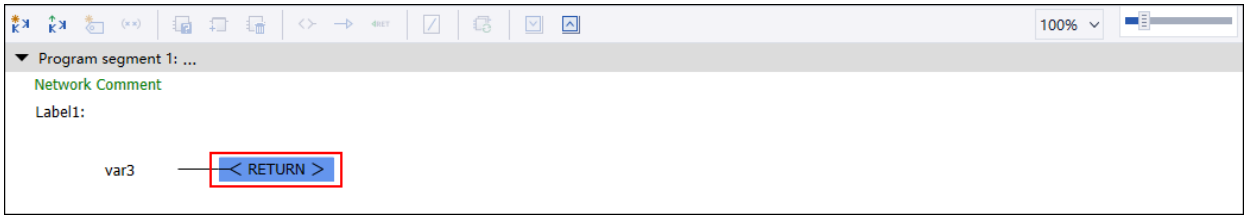
- **Assignment**

  Used to assign a value to another value. As shown in the figure below, var2 := var1 means to assign the value to var1 to var2.



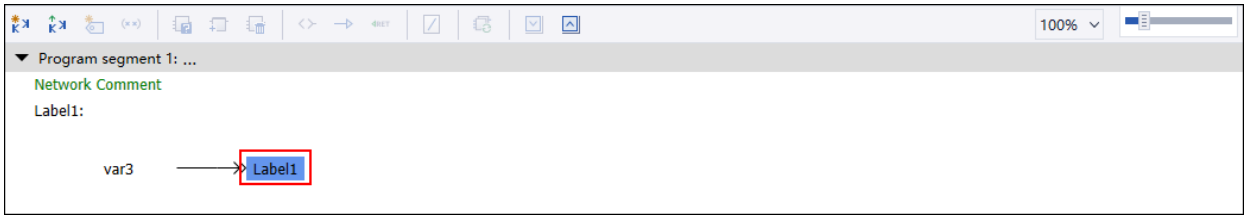
- **Return**

  The return element is used to terminate program execution. Once a return element is encountered, all subsequent program code will no longer be executed.



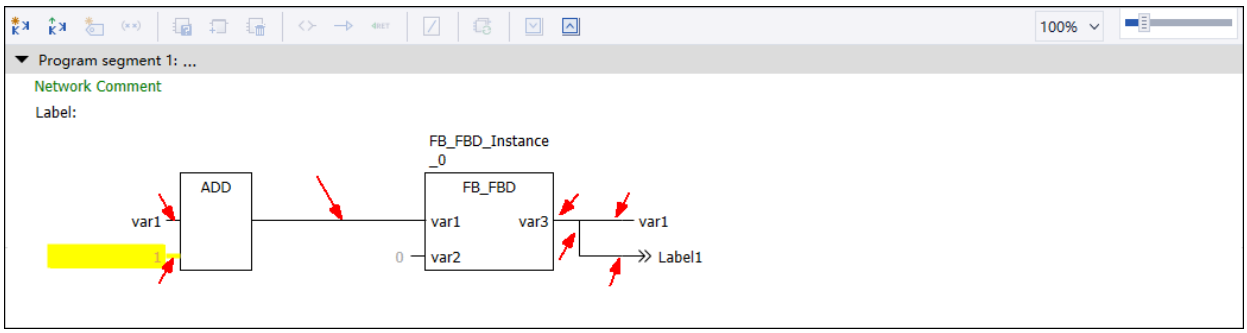
- **Jump**

  The jump element is used to transfer program execution from the current position to another specified location.



- **Connection**

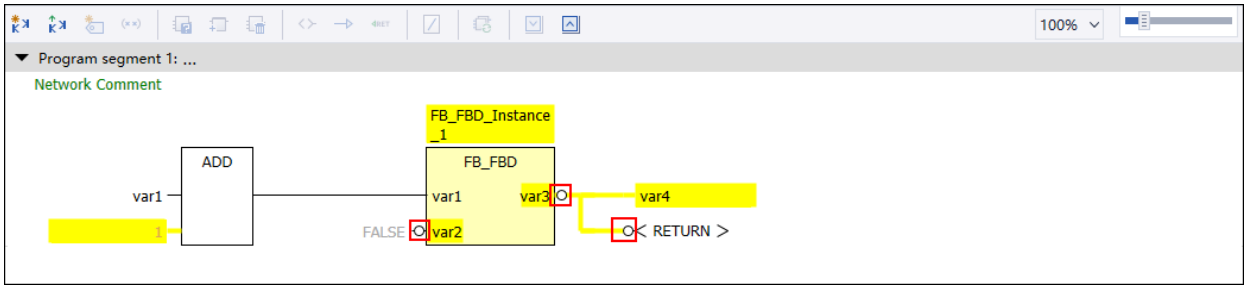
  A connection is a line that connects two positions. Connection lines mainly include contact connections, input/output connections of operation blocks, connections between coils, terminal connections, branch terminal connections, and vertical connections.



Auxiliary element

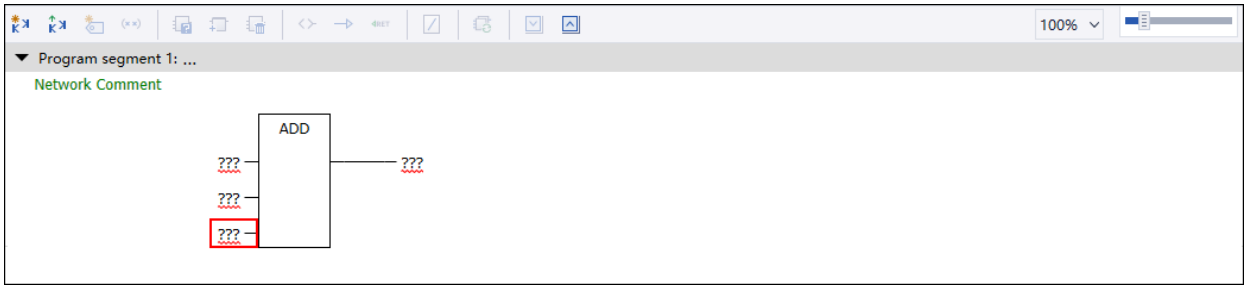
- **Inversion**

  Supports inverting the block pin of BOOL or SAFEBOOL type.



- **Input pin**

  For blocks supporting multiple inputs, input pins can be added.



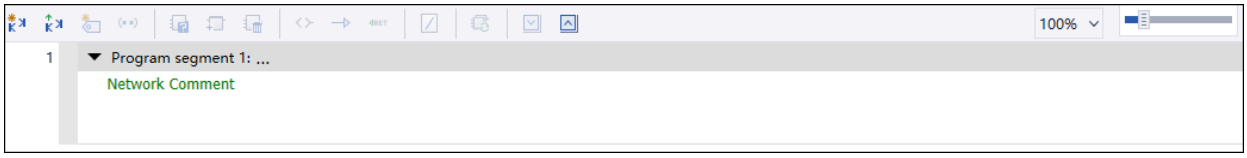
11.2.5.5 Safety FBD Programming

Safety FBD allows users to add required elements to the network using commands from the toolbar. It also supports directly dragging user-defined program blocks from the toolbox and dropping them into the network to create corresponding block instances.

Adding elements by commands

Elements that can be added by commands include: network, label, block, input pin, assignment, return, jump, and inversion.

- Commands in the toolbar



Commands in the toolbar are fixed. Depending on the selected element, different commands are enabled. Only enabled commands support adding corresponding elements around the selected element, which cannot be achieved by disabled commands by default.

- Commands in the right-click menu

Commands in the right-click menu vary depending on the selected element. Only the commands that are visible for the selected element can be used to add corresponding elements around it.

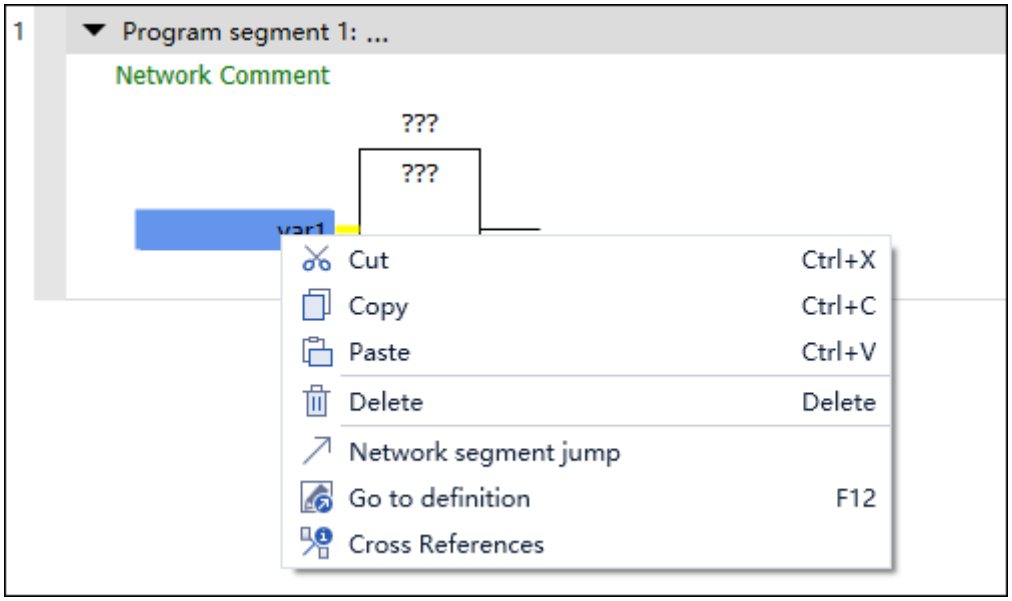


Table11-2-5-5-1 Icons and descriptions of commands

Type	Icon	Description
Add/Insert network		Adds/Inserts a network.
Insert label		Adds a label to network.
Toggle network comment		Comments or uncomments the selected network.
Insert empty function block		-
Add input		-

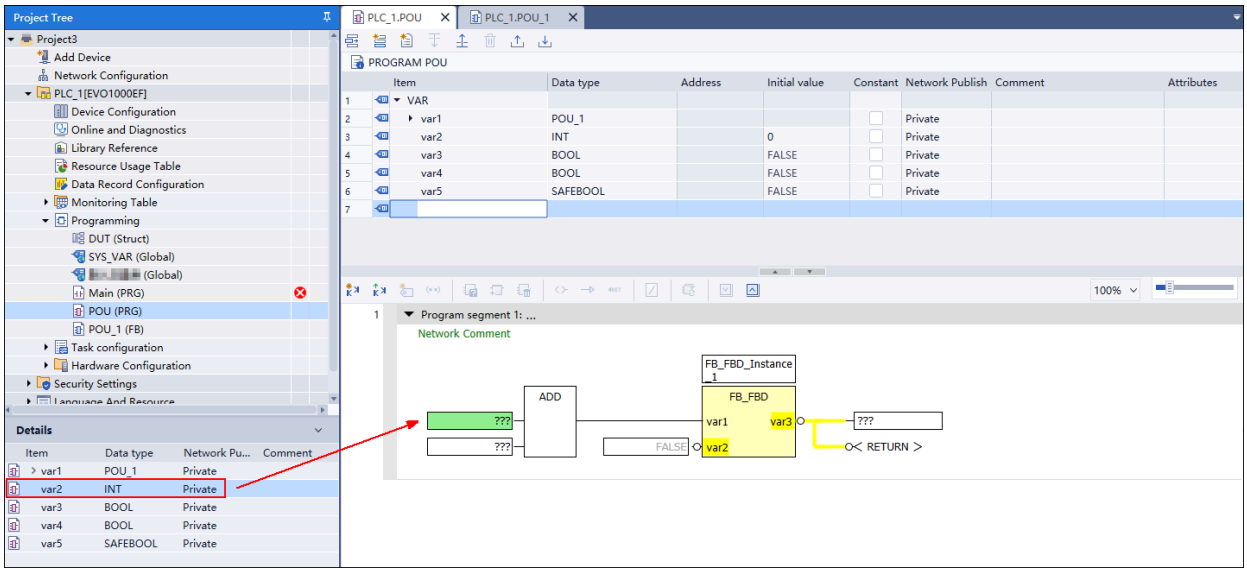
Type	Icon	Description
Insert input		-
Remove unused pins		Deletes unconfigured pins for FB call blocks.
Insert assignment		-
Insert jump		-
Insert return		-
Inversion		Inverts a contact or a BOOL connection.
Update all parameters		When parameters are changed at the FB declaration but not at the call site, this icon can update the block at the call site.
Expand all networks		-
Collapse all networks		-

Adding elements by dragging

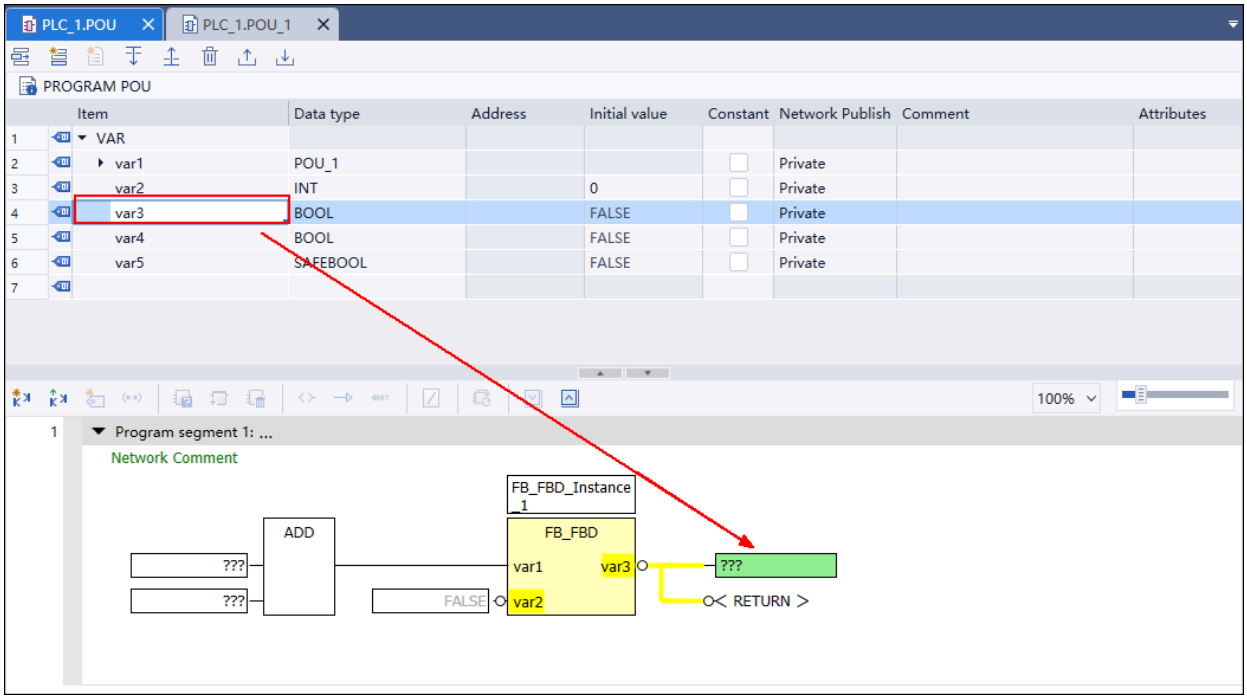
Instance variables from the "Details" window and the variable declaration area can be dragged to block input/output pins for addition or modification. Blocks from the project tree or toolbox can be dragged to the editor for addition. Elements within the editor to another location in the editor. Currently, variables within the editor cannot be dragged.

- Dragging variables

  Drag instance variables from the "Details" window to the operand variable locations in the editor.

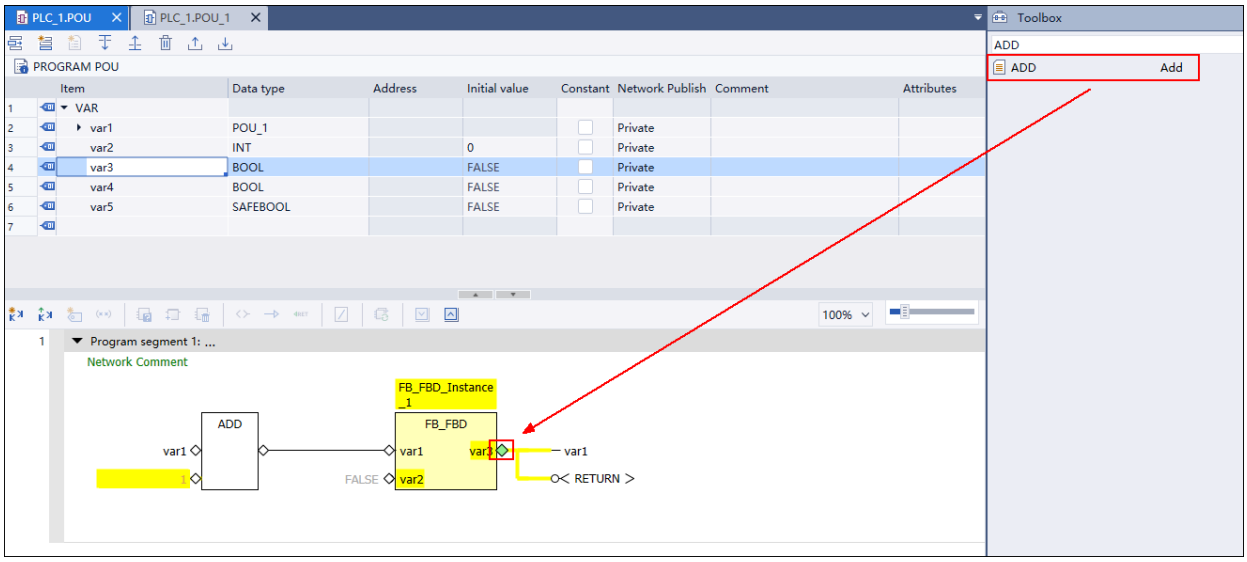


  Drag variables from the variable declaration area to the operand variable locations in the editor.



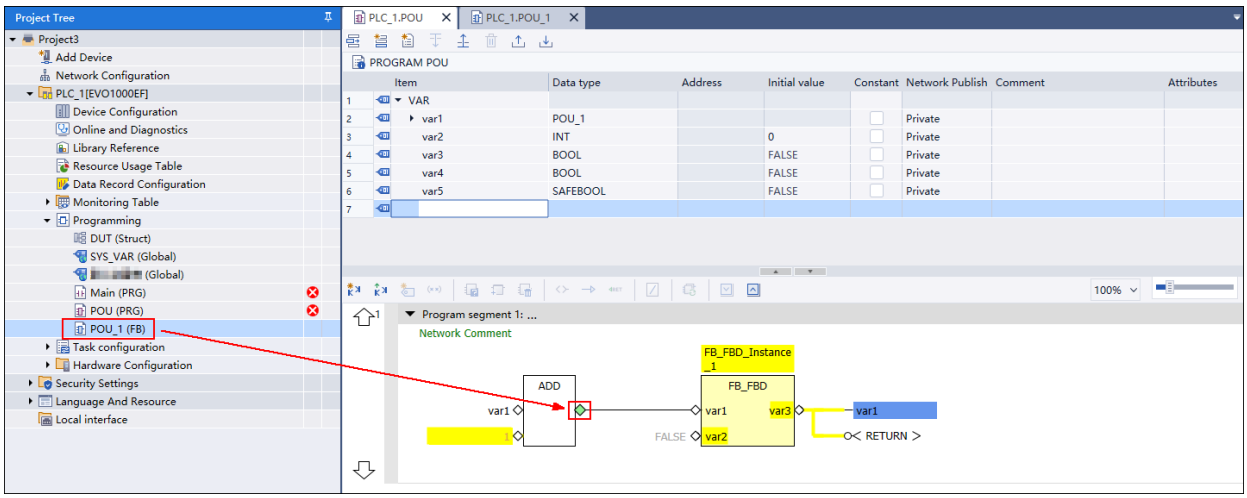
- Dragging elements from the toolbox

  When dragging an element from the toolbox to the editor, the available locations are highlighted with a diamond-shaped mark. Hover over these locations and they will be displayed in green. Release the mouse to add the selected element to the corresponding location.



- Dragging elements from the project tree

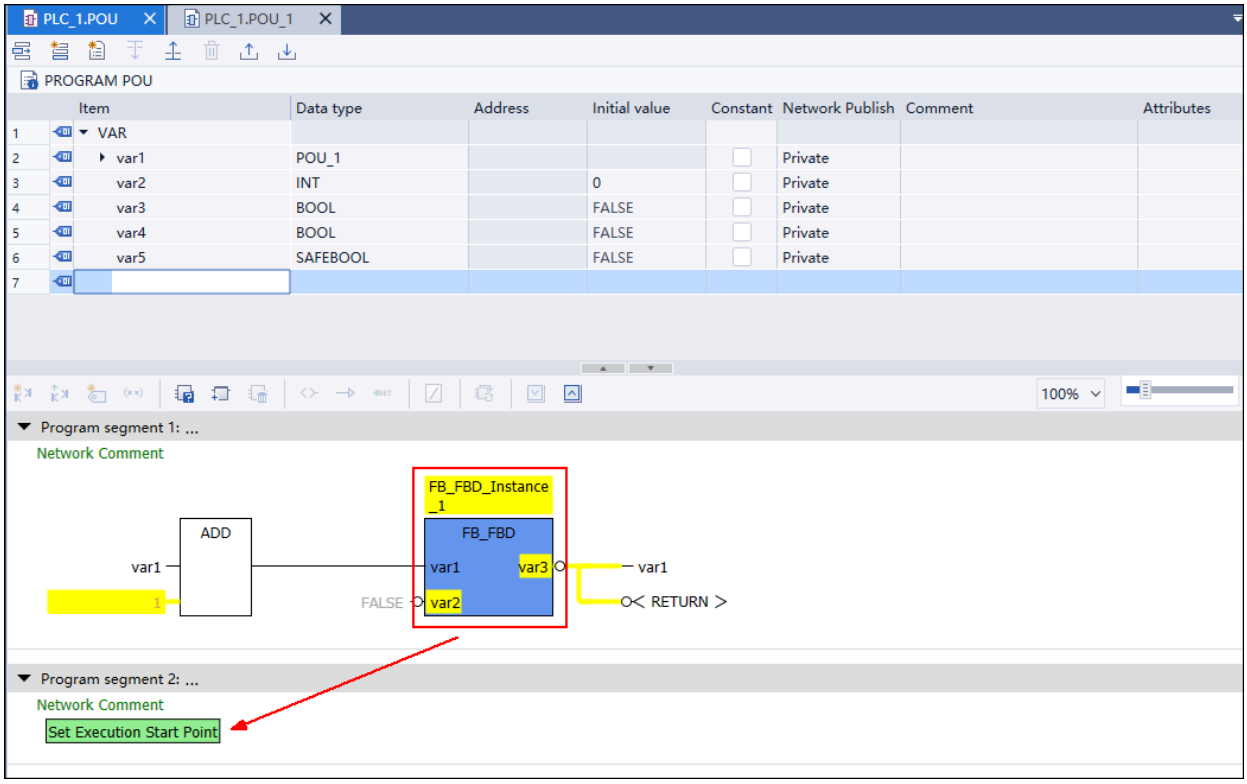
  When dragging an element from the project tree to the editor, the available locations are highlighted with a diamond-shaped mark. Hover over these locations and they will be displayed in green. Release the mouse to add the selected element to the corresponding location.



- Dragging elements in the editor

  When dragging an element in the editor to another network or another location within the same network, the available locations in the target network are highlighted with a diamond-shaped mark. Hover over these locations and they will be displayed in green. Release the mouse to move the selected element to the corresponding location.

  If you keep pressing Ctrl during dragging and dropping, the element at the source location will not be deleted and a new element will be added at the target location after. If Ctrl is not pressed during dragging and dropping, the element at the source location will be deleted.



11.2.5.6 Graphic Configuration Modification

11.2.5.6.1 Modifying Text Data

After adding a network, assignment, or block, the system provides corresponding text data for them:

- For networks: Includes network title, comment, and label text.
- For assignments: Includes variable text.

- For blocks: Includes block type, block instance, and block input/output text.

Users can modify these text data as follows:

- For network titles and labels, double-click the title or label to modify the text.

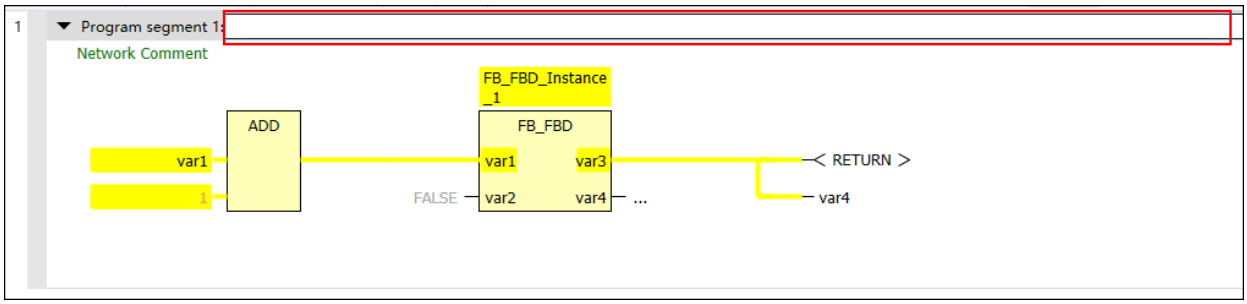


Figure 11-2-5-6-1-1 Modifying network title

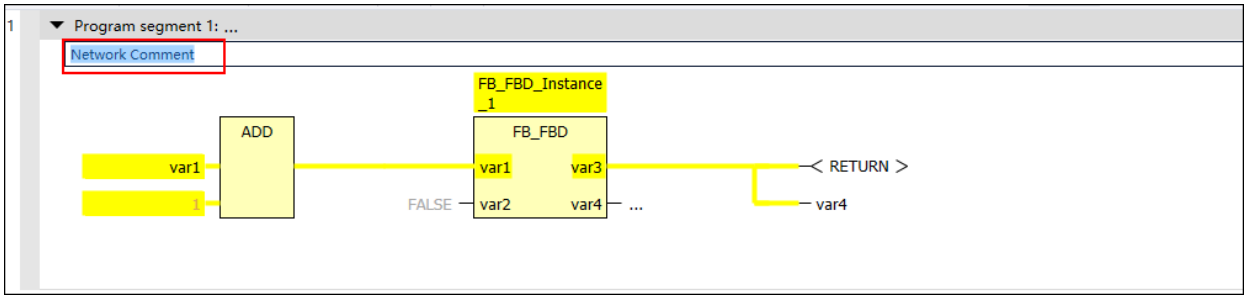


Figure 11-2-5-6-1-2 Modifying network label

- For assignment and block text, drag a variable to the corresponding location or double-click the text for modification.

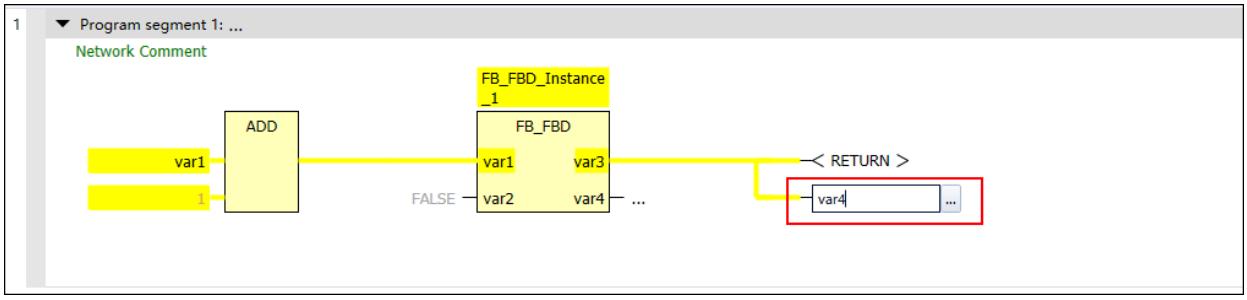


Figure 11-2-5-6-1-3 Modifying assignment text

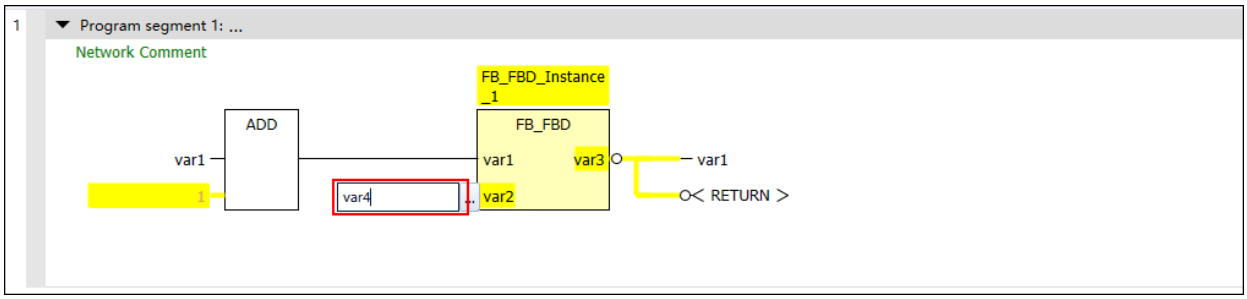
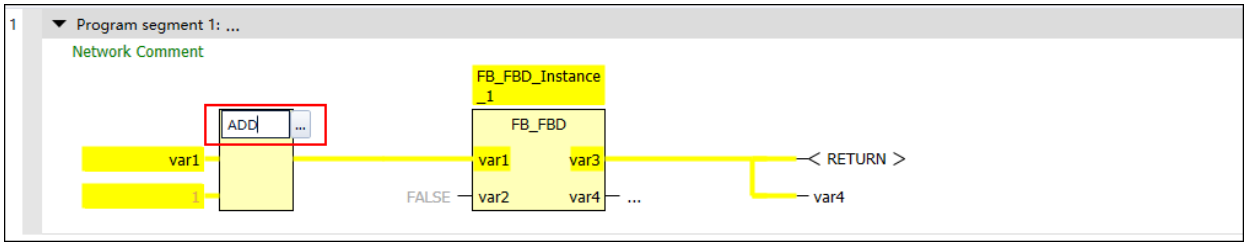


Figure 11-2-5-6-1-4 Modifying block text

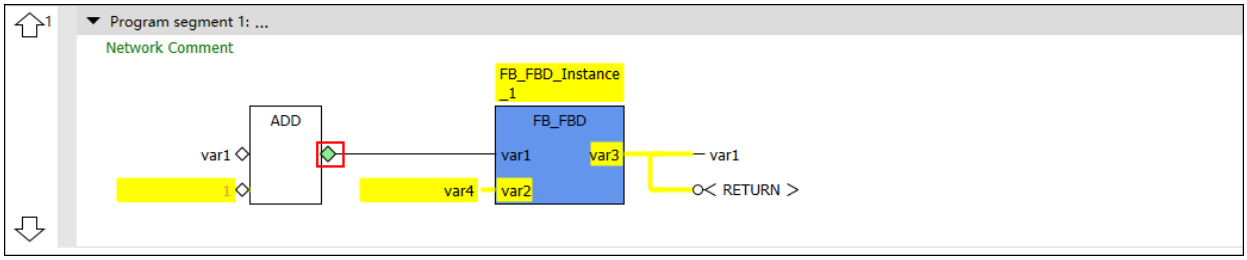
11.2.5.6.2 Modifying Block Type

Double-click a block type to modify its type.



11.2.5.6.3 Moving an Element

To move an element in the editor, you can delete it and then add it to the target position, or directly drag it to the target position. As shown in the figure, a block POU can be dragged to another location.



11.3 Data Record Configuration

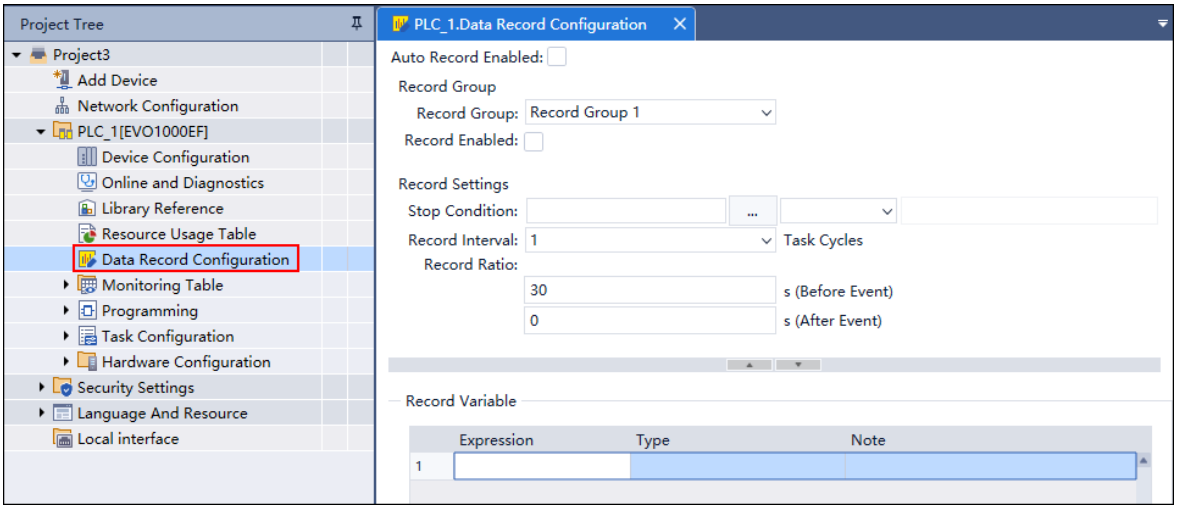
The safety PLC supports recording historical values of variables. During operation, the PLC records variable values to the SD card in the communication unit according to conditions predefined in the "Data record configuration". The recordable data range includes safety PLC's external public variables and device variables. Each variable can record up to 3,000 pieces of data. The iFA Evolution software only supports setting data recording conditions and does not support reading or displaying recorded historical data.

Prerequisites

- A qualified SD card (with more than 10% available space) has been inserted into the communication unit of the safety PLC.
- A project has been created and a safety PLC device has been added.
- A program has been added, and variables have been created.

Procedure

1. Double-click "Data record configuration" in the project tree.
2. Set data recording conditions.



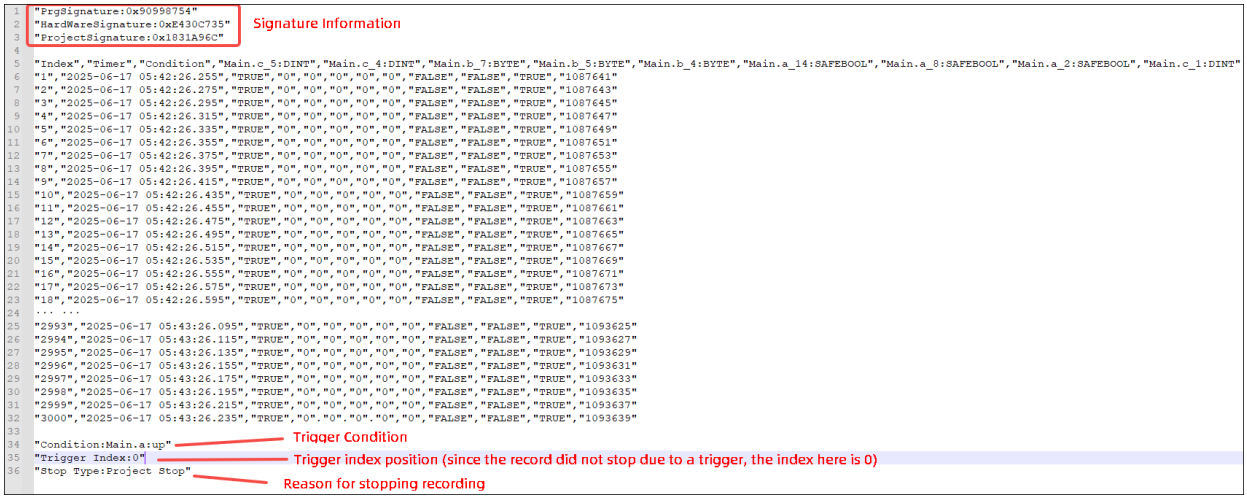
Item		Description
Auto recording enabled		If checked, the safety data will be automatically recorded after the PLC program starts. Both record group configurations will take effect simultaneously.
Record group	Record group	Select the record group to be configured.
	Recording enabled	Set whether to enable the selected recording group configuration.
Recording settings	Stop recording condition	Set the trigger condition for stopping recording. Each recording group can specify a variable. When the condition is met, recording will complete the remaining according to the set recording rate before stopping. ◦ BOOL variable: TRUE (rising edge), FALSE (falling edge). ◦ Non-BOOL variables: Select one of the following options and set a constant value. ▪ [=]: Equal to constant value ▪ [≠]: Not equal to constant value ▪ [>]: Greater than constant value ▪ [<]: Less than constant value ▪ [≥]: Greater than or equal to constant value ▪ [≤]: Less than or equal to constant
	Recording interval	Set the interval period between two consecutive recordings, with a range of 1 to 10 task cycles.
	Recording ratio	The data recorded before and after the moment when the stop recording condition is met. By default, only the data before the stop condition is recorded; no data is recorded after the condition. When one time (before or after the event) is modified, the other is automatically calculated. The sum of both times should not exceed the total recordable time of the buffer. Total recordable time = 3000 × recording cycle × recording interval. For example, with a recording cycle of 10 ms and a recording interval of 1, the total recordable duration for 3000 variables is 30 seconds.

Item	Description
Record variable	Set the variables to be recorded (up to 100 variables allowed). Variables are added from the safety PLC variable table. Only public variables or device variables can be selected. The data type and comment of the selected variables are referenced from the variable table and cannot be edited. ◦ Single variable addition is supported via intelligent suggestion input and input assistant. ◦ Batch variable addition is supported by dragging one or multiple variables from the variable table or "Details" window to the record variable table. **NOTE** Direct addition of complex variables with parent-child structures is not supported. In such cases, variables can only be added individually.

3. After configuration, download the project (including the safety data recording configuration file) to the safety PLC. After the PLC starts running, it will execute the safety data recording according to the configuration.

Result

After safety data recording, a TXT file is generated in a fixed format and stored on the SD card in the communication unit. Recorded information includes project signature, data information, recording time, trigger condition, and reason for stopping recording. Open the generated TXT file. The recorded content is shown in the figure below.



11.4 PLC Configuration

11.4.1 Hardware Configuration and Device Management

11.4.1.1 Adding a Safety PLC and Communication Unit

Taking the EVO1000EF as an example for the safety PLC and the EVO1000EF EIP as an example for the communication unit, the method for adding other models of safety PLCs and communication units is the same.

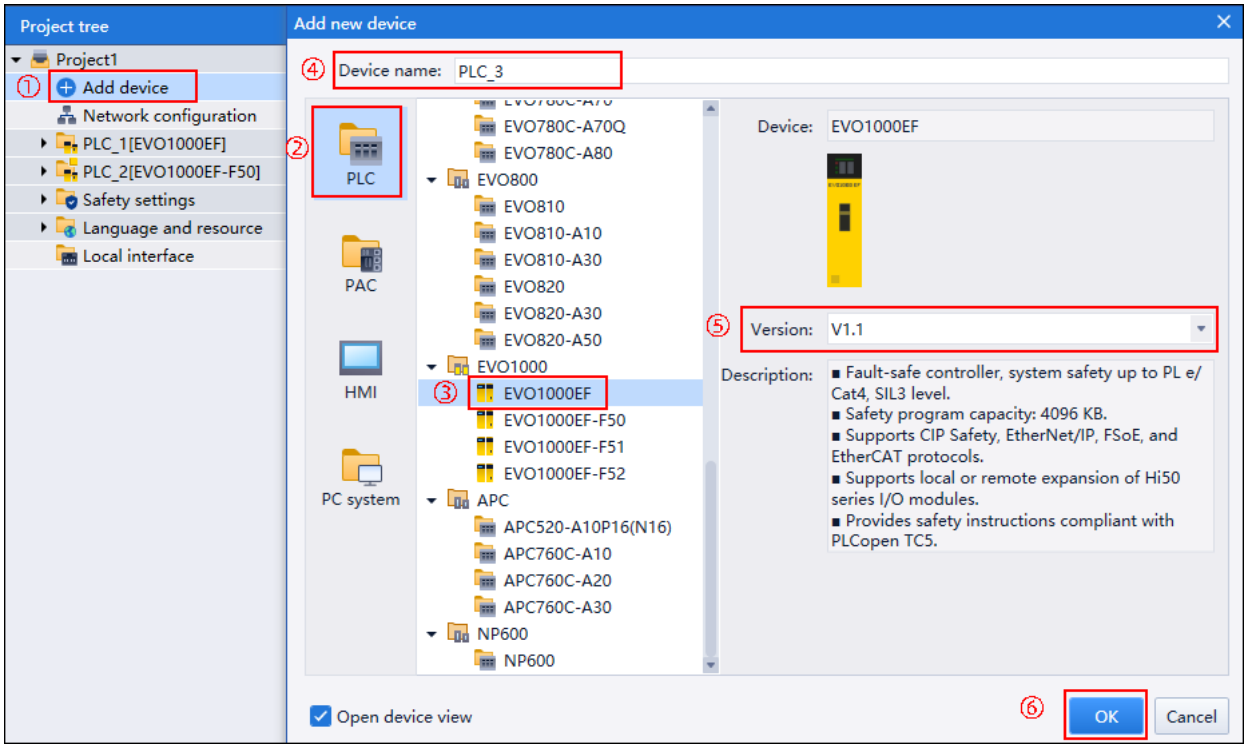
> NOTE
>
> During EtherCAT communication, local failsafe I/O and remote failsafe I/O (including couplers) can only be created, deleted, and renamed under a standard PLC. Whether it is local or remote I/O for the safety PLC, its slot number must be displayed in the project tree.

Adding a safety PLC

To add a safety PLC, follow the steps below. For details, see 7.2.1 Adding a Device.

1. Double-click the "Add device" node in the project tree.
2. Select the PLC device to add, such as "PLC > EVO1000 > EVO1000EF".
3. Modify the "Device name" as needed, select the "Version", and click "OK" to add the new device.

> NOTE
>
> - Version V1.0: The safety PLC is equipped with an EIP communication unit.
> - Version V1.1: You must manually add an EIP or ECT communication unit.

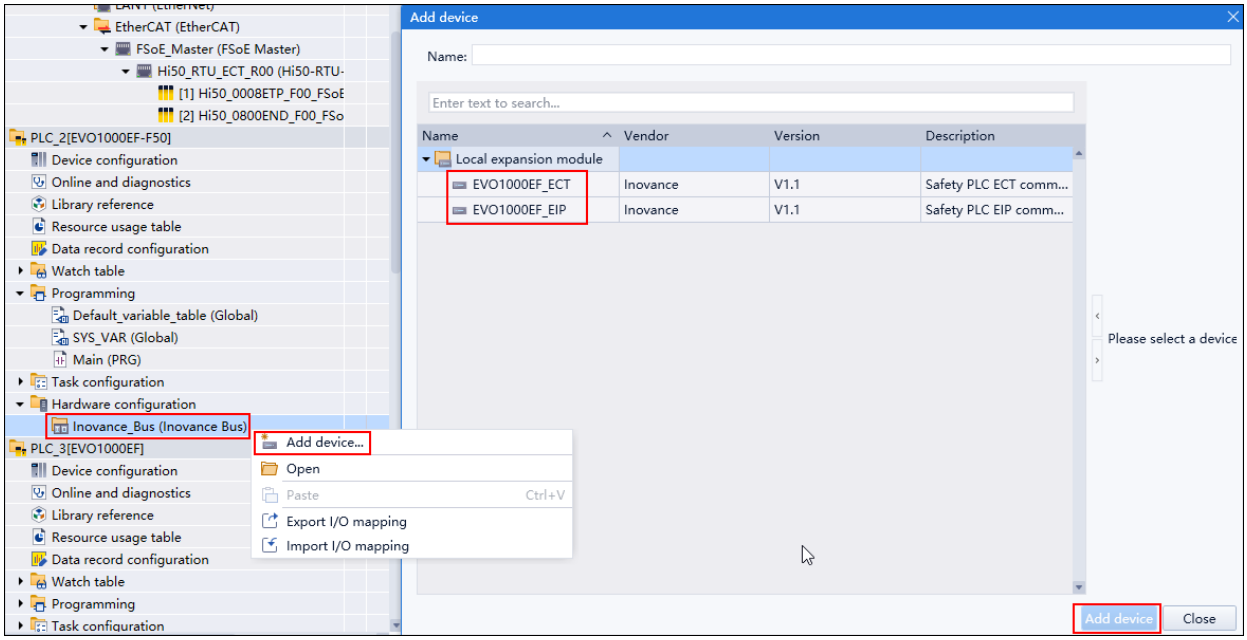


Communication Module Addition

For safety PLC V1.1, you must manually add an EIP or ECT communication unit. Communication units can be added via the project tree view or the device view.

Method 1: Adding via the project tree

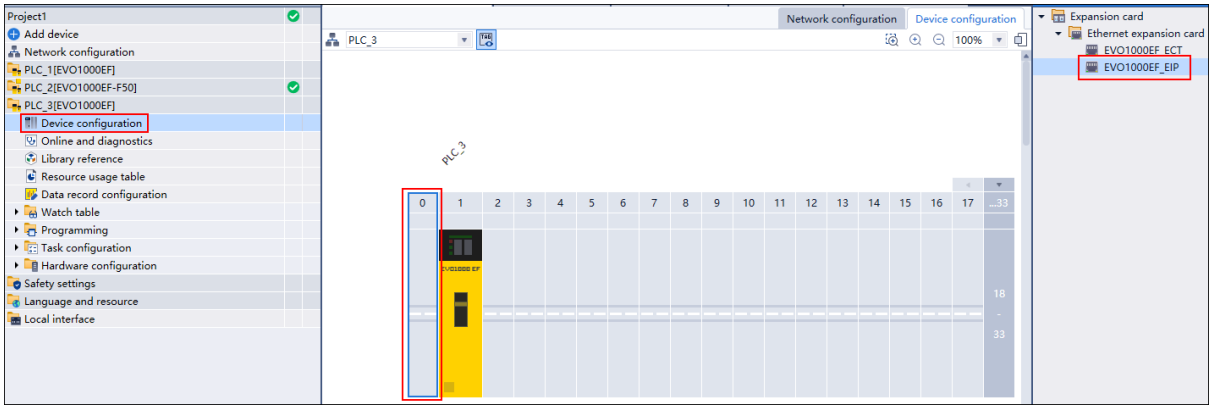
1. Right-click the "Inovance_Bus" node under "Hardware configuration" node of the safety PLC device, and select "Add device".
2. Select the communication unit as needed, such as "EVO1000EF_EIP", and click the "Add device" button.



Method 2: Adding via the device view

1. Double-click "Device configuration" of the safety PLC device in the project tree.
2. Select a communication unit under "Device list > Expansion card > Ethernet expansion card", such as the "EVO1000EF_EIP" module. When the module is selected, the device view highlights the border of the slot, indicating where the module can be inserted.

3. Double-click a module to add it to the nearest available slot on the PLC, or select a module and drag it to the desired location in the device view, then release the mouse button to complete the addition.



11.4.1.2 Device Scanning

The device scanning method in the safety PLC is the same as that in the standard PLC. Only the functional differences are described here. For other general functions, see 10.6.1.1 Device Scanning in the "PLC Programming" chapter.

For safety PLCs, the "Communication type" adds the "EOE" method. The operation method is the same as the "Ethernet" method.

11.4.2 Failsafe I/O Module Configuration

11.4.2.1 Rack Overview

To mount modules after the main device, a rack (i.e., a mounting rail) must be used to secure the modules to the rack and connect them to the CPU, couplers, power supply, or other modules via the backplate bus.

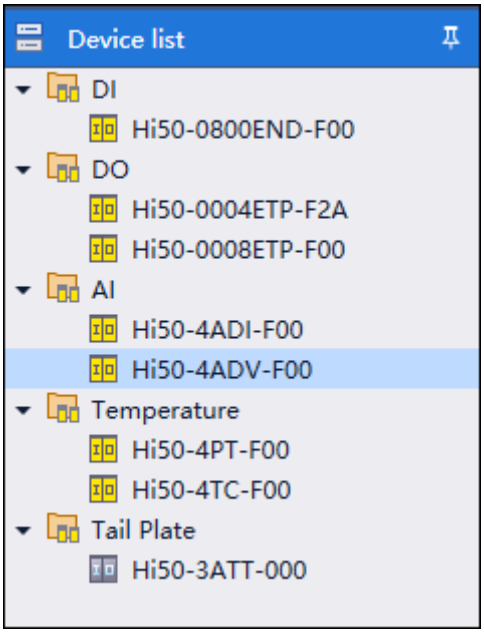
Creating a rack

After adding a device, a rack suitable for the selected device is automatically created. The available slots in the rack are displayed in the device configuration view, and the number of available slots depends on the type of device used.

Rack structure

There are empty slots to the right of the device. The slot numbers are located above the slots and a brief description is displayed above the inserted device and modules. Use the "Display label" icon on the toolbar to display or hide the brief description of the device.

In the right device list, the slots that allow insertion of these modules are displayed.



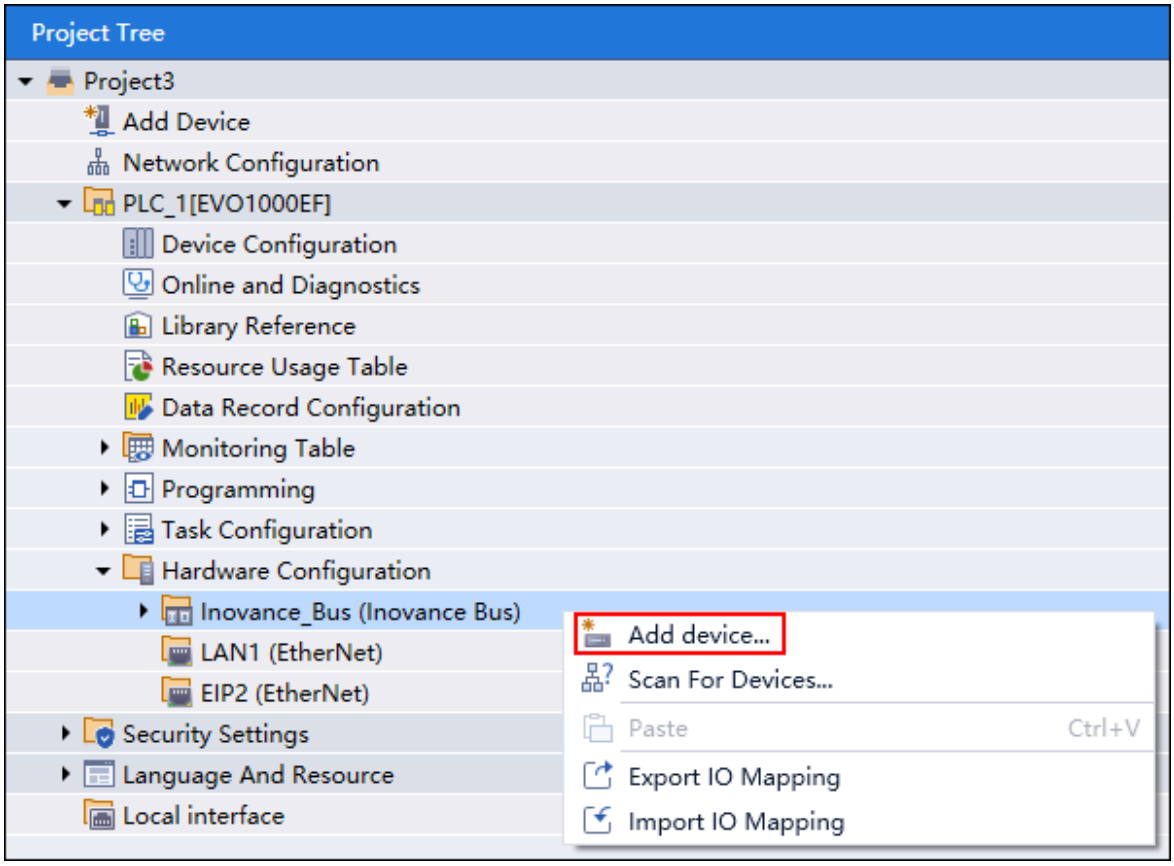
PLC local rack I/O modules are referred to as local I/O modules. For EVO1000 series PLC, the CPU rack defaults to supporting up to 31 local I/O modules, including safety digital input modules (DI), safety digital output modules (DO), safety current analog input modules (ADI), safety voltage analog input modules (ADV), safety thermal resistor analog input modules (PT), and safety thermocouple analog input modules (TC).

11.4.2.2 Adding a Module

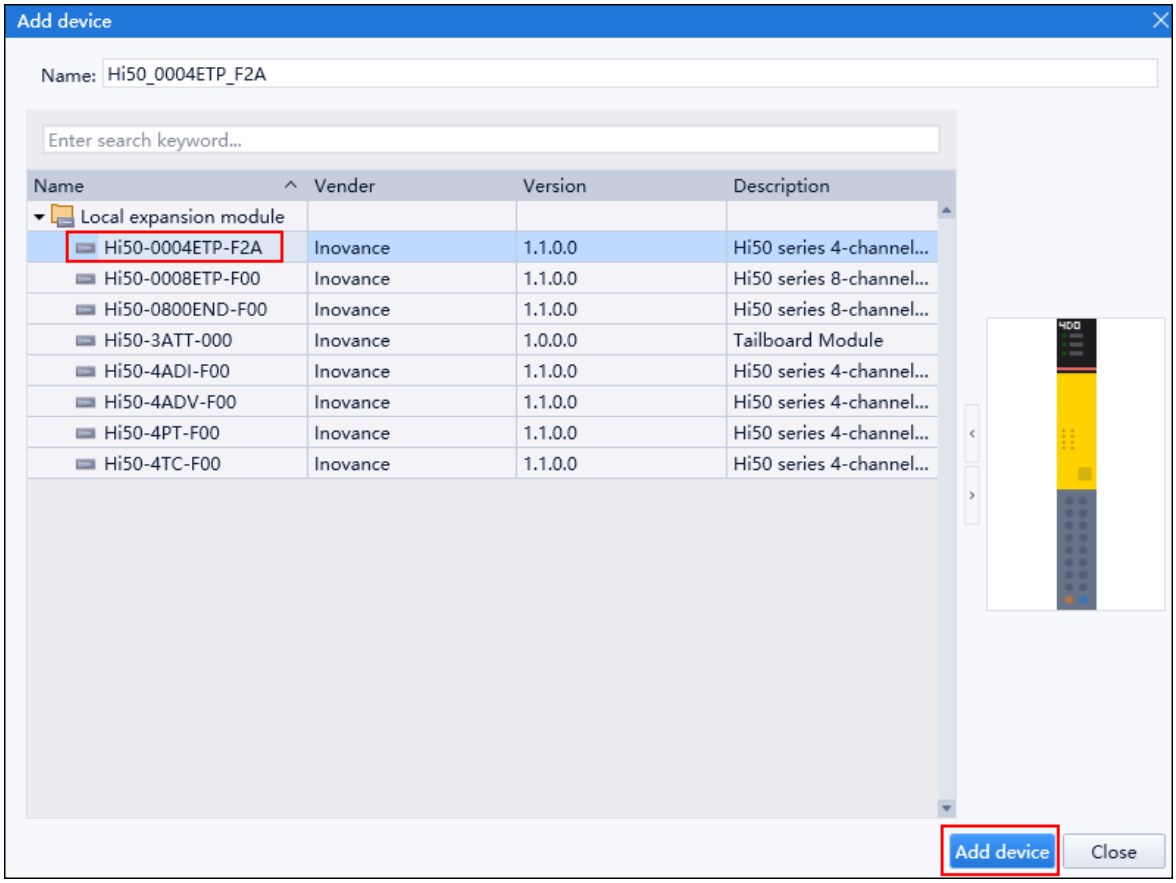
The following describes the methods for adding modules via the project tree, device configuration view, and device scanning, as well as specific restrictions for adding tail plate modules:

Method 1: Adding a module through the project tree

1. Expand the "Hardware configuration" list in the left project tree.
2. Right-click "Inovance_Bus (Inovance Bus)" and select "Add device".



3. In the pop-up window, select the module and click "Add device", or double-click the module to complete the addition.

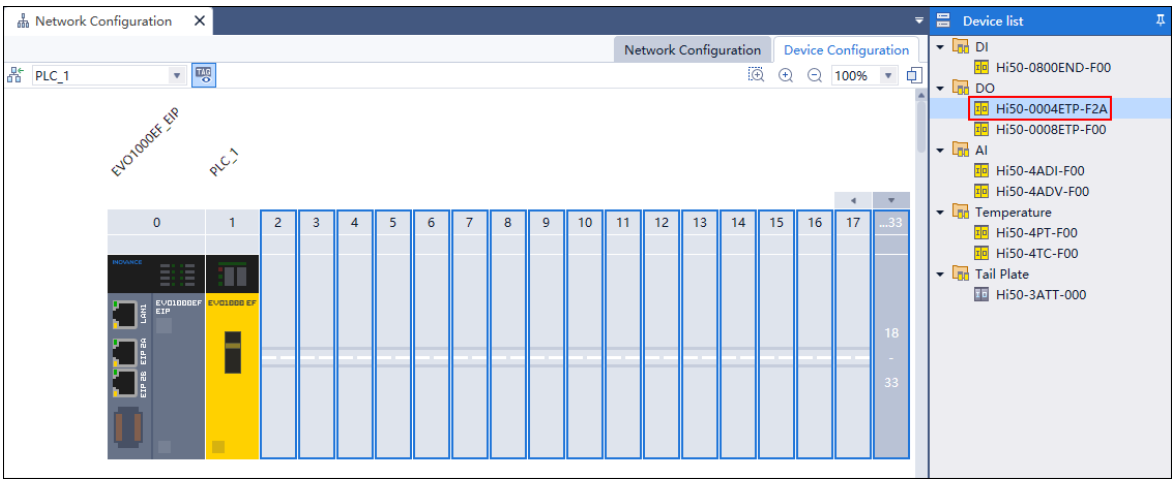


4. After the module is added, the result will be synchronized in the device configuration view.

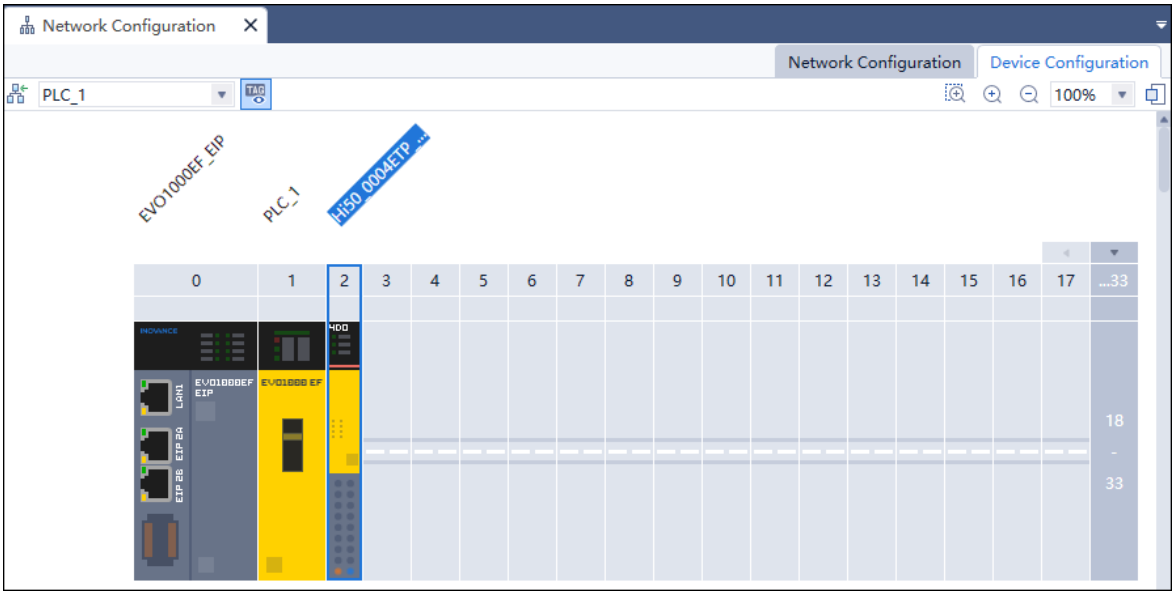
Method 2: Adding a module through the device configuration view

1. Double-click the "Device configuration" node in the left project tree.

2. In the right device list, select the module. When a module is selected, the insertable slots for the module are highlighted in the configuration diagram.



3. In the device list, double-click a module to add it to the nearest available slot adjacent to the PLC. You can also drag the module from the device list to an available slot in the device configuration view.



4. To change the module slot, drag the module in the device configuration view to the target slot.

5. The added module can be copied and pasted using the right-click menu or keyboard shortcuts.
   Only one module can be copied and pasted in the device list. Multiple modules can be copied and pasted in the device configuration view.

6. After the module is added, the result will be synchronized under the "Hardware configuration > Inovance_Bus (Inovance Bus)" node in the project tree.

Method 3: Adding a module through device scanning

The iFA Evolution software provides an automatic scanning function for safety I/O modules, allowing users to directly add modules by scanning.

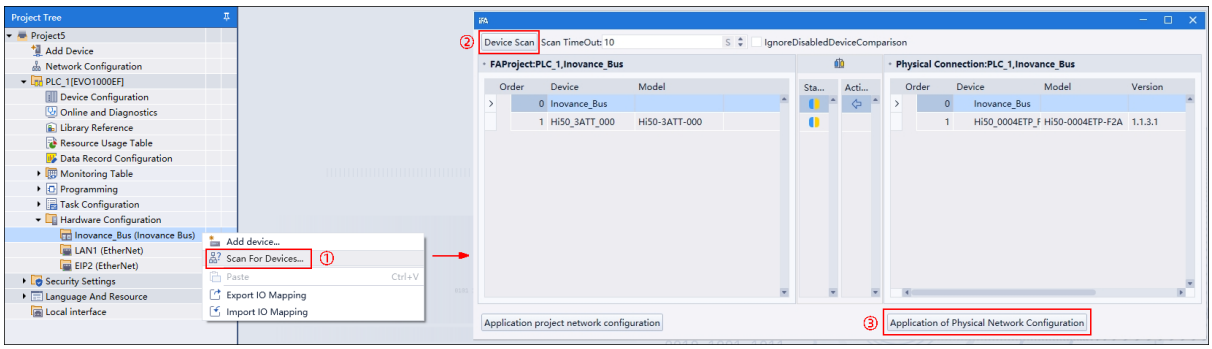
Prerequisites

- A blank project containing the "CIP_Safety_Scanner" node has been deployed to the target safety PLC.
- The user program of the safety PLC is in the stopped state.

Procedure

1. Right-click the "Inovance_Bus" node under "Hardware configuration" node, and select "Scan for devices".

2. In the pop-up window, click "Device scanning" and synchronize all scanned I/O modules into the current project.
3. In the pop-up confirmation window, click "Yes" to restore the synchronized device configuration to default.
4. Click "OK" to add the safety I/O module to the project.



Adding a backplate module

The Hi50-3ATT-000 module is a backplate module for the CPU rack of the safety PLC and the I/O modules of the Hi50 coupler rack. This module is only used for display in the device view and requires no configuration. Additionally, no information about this module will be included when downloading the project.

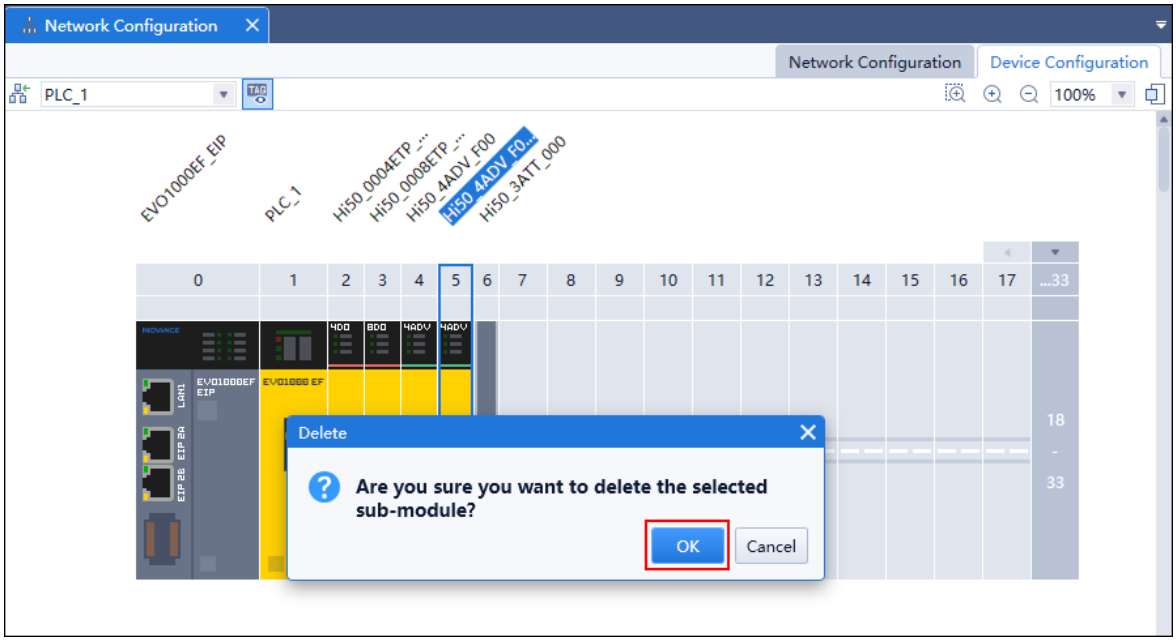
The method for adding a backplate module is the same as for other I/O modules. The following are the restrictions for this module:

- Only one backplate module is supported in a rack.
- The backplate module can only be added to the slot immediately to the right of the last I/O module.
- If no I/O modules are added to the rack, the backplate module can only be added to the slot immediately to the right of the CPU module (slot 2).
- After adding a backplate module to the rack, no other I/O modules can be added to the slots to the right of this module. To add more modules, the backplate module must first be manually removed.
- After a backplate module is added, it cannot be moved. When there are no available slots to the left of the backplate module, no other I/O modules can be inserted. To insert another module, you must either manually delete the backplate module or delete the modules on the left to free up a slot.
- The last slot in the rack (slot 33) is reserved for the backplate module; no other I/O modules can be inserted into this slot.
- If a backplate module is not manually added during configuration, it will be automatically added during compilation.

11.4.2.3 Deleting a Module

The deletion procedure of a module is the same as that of a regular slave device. The recommended method for deleting a module is as follows:

1. Double-click the "Device configuration" node in the left project tree.
2. In the device configuration view, select a module on the PLC rack; multiple selections are supported.
3. Press the Delete key, then click "OK" in the pop-up window to delete the module.



When a module is deleted, the variables mapped to its channels will be released. At the same time, the deleted module will be removed from the "Hardware configuration > Inovance_Bus (Inovance Bus)" node in the project tree.

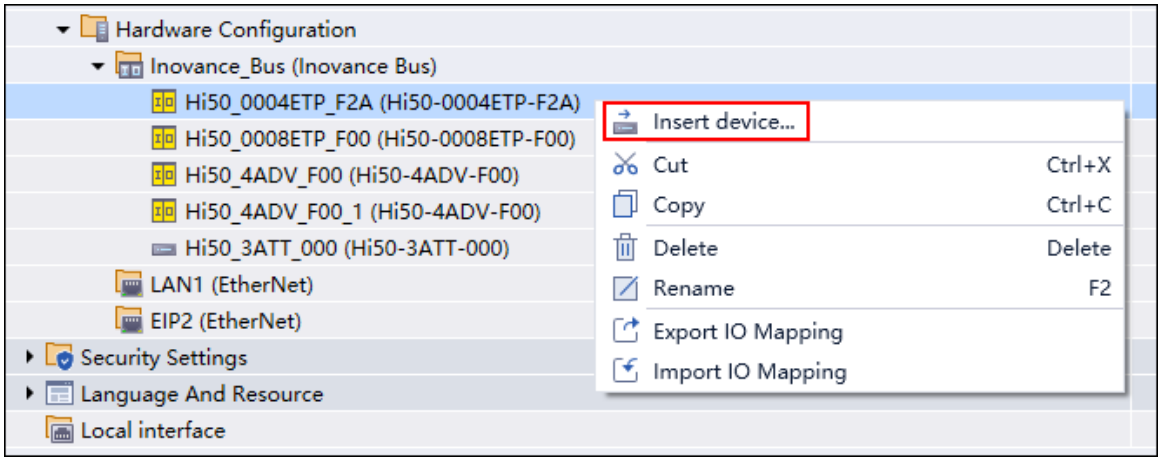
> **NOTE**
>
> After deleting an I/O module, the addresses of the associated I/O variables will be rearranged.

11.4.2.4 Inserting a Module

The module can be inserted in the following three ways:

- **Inserting through the project tree**

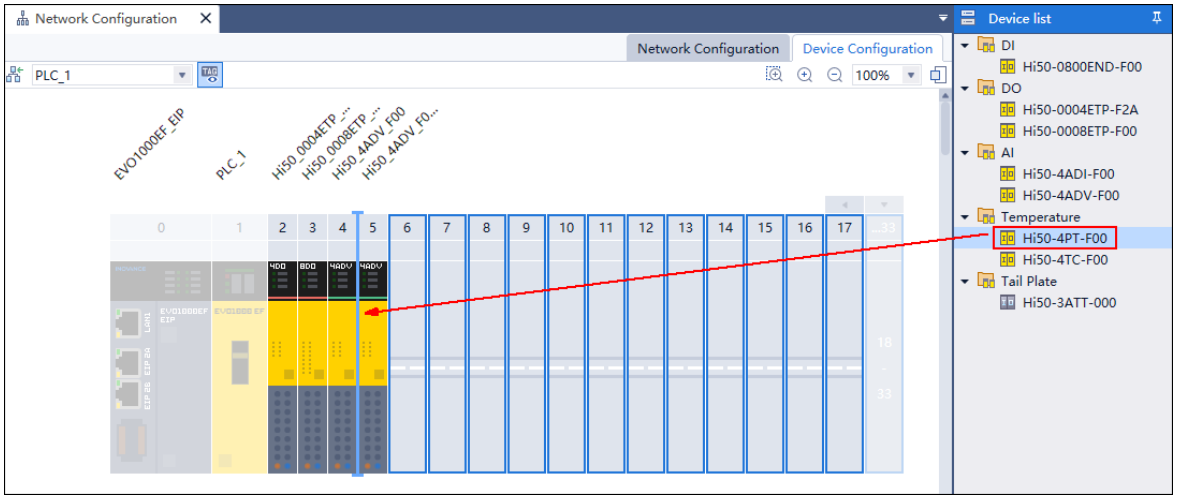
  Right-click an existing module under "Hardware configuration > Inovance_Bus (Inovance Bus)" and select "Insert device".



  The inserted module is placed by default in the slot to the left of the selected module.

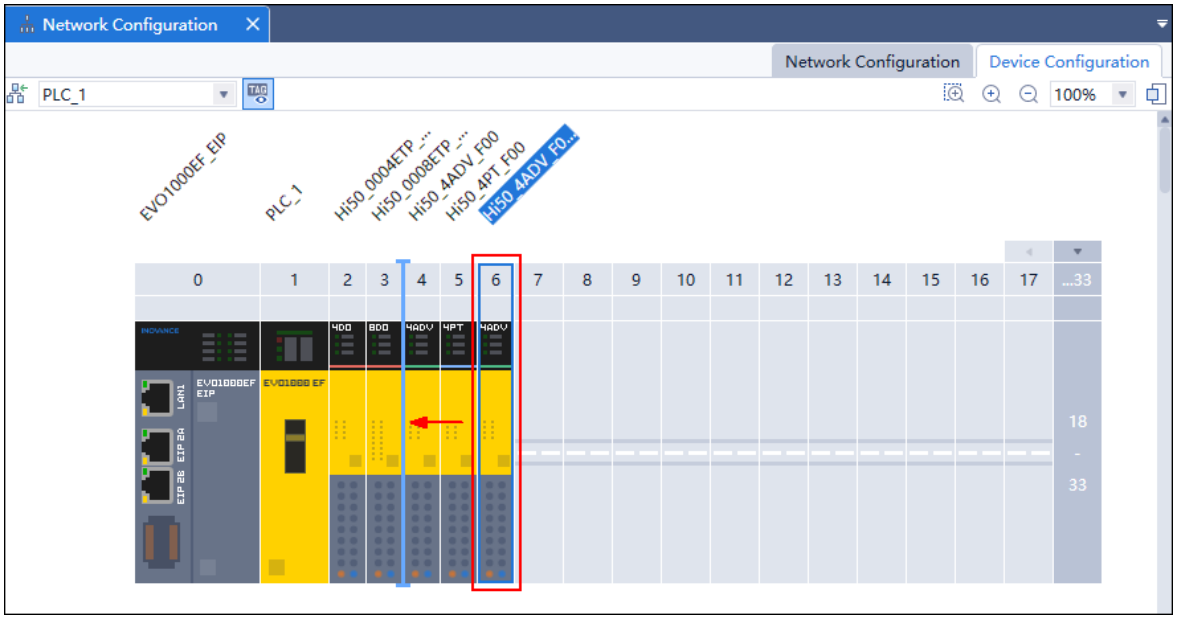
- **Dragging and dropping from the device list**

  Drag a sub-module from the device list to insert it into existing slots.



After successful insertion, the module whose location has been adjusted is highlighted.

- **Dragging and dropping from the device view**
  Drag a sub-module from the device configuration view and insert it into another slot.



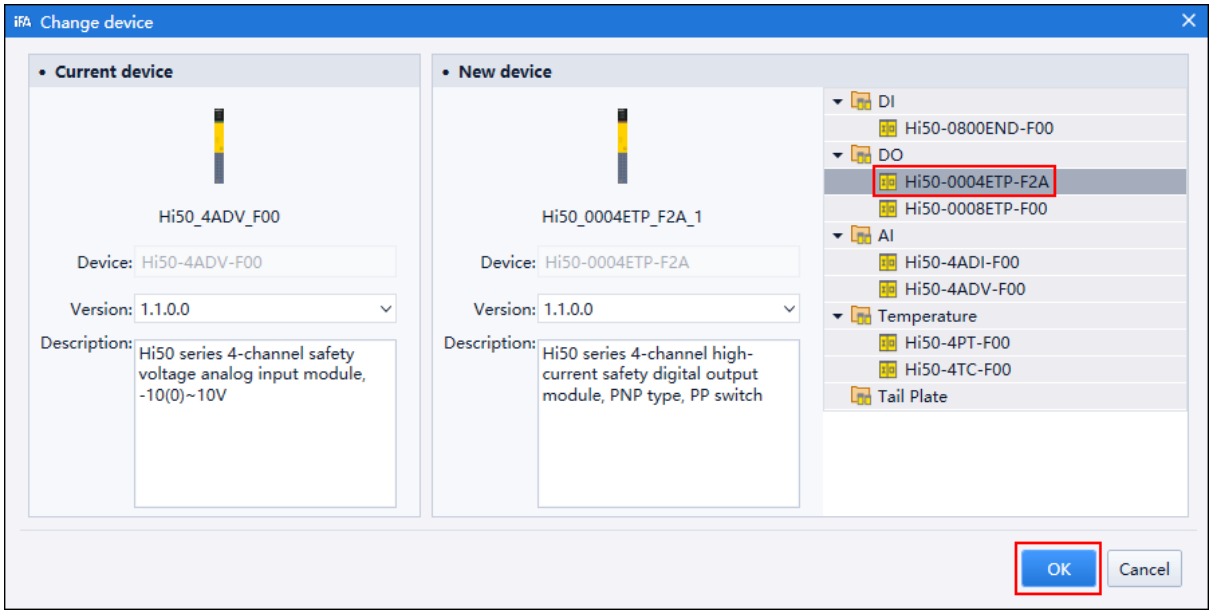
NOTE

- After inserting an I/O module, the addresses of the associated I/O variables will be rearranged.
- When a backplate module has been added to the rack, no other I/O modules can be inserted to the right of the backplate module. If there are no available slots to the left of the backplate module, no other I/O modules can be inserted either.

11.4.2.5 Replacing a Module

The module can be replaced in the following two ways:

- **Replacing through the right-click menu**
  In the device view, right-click the module to be replaced and select "Change device". Select the new device in the pop-up dialog box, and click "OK" to complete the module replacement.



- **Dragging and dropping from the device view**
  In the device list, drag the new module onto the module to be replaced in the device configuration view. After releasing the mouse, the "Change device" dialog box appears. Click "OK" to complete the module replacement.

11.4.2.6 Moving a Module

The module movement includes the following two scenarios:

- **Moving to an empty slot**
  In the device configuration view, drag a module to move it to another empty slot.
- **Moving to a non-empty slot**
  In the device configuration view, drag a module to move it to another non-empty slot to switch the positions of the two modules. After releasing the mouse, the modules whose positions have been changed are highlighted.

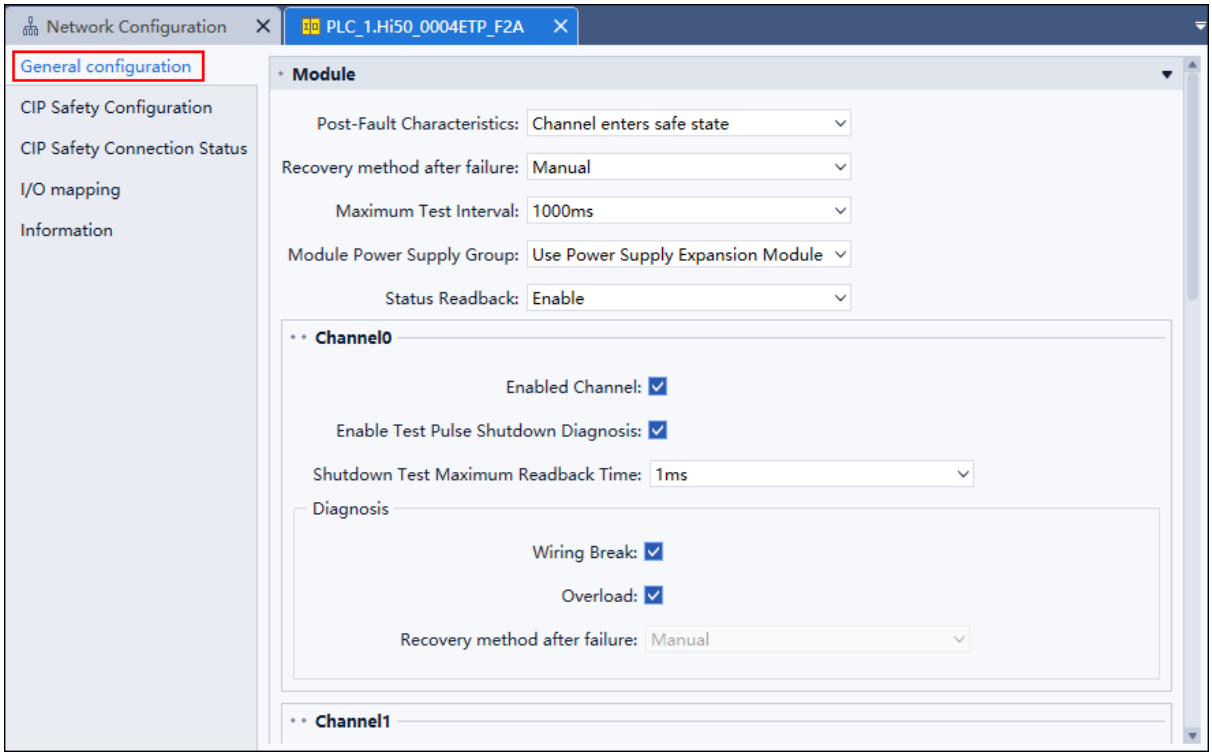
11.4.2.7 Failsafe I/O Module Configuration

The iFA Evolution software supports parameter configuration for each failsafe I/O module. The following describes the specific configuration items:

General configuration

This tab is used to configure module-level general settings and channel settings.

1. In the project tree or device configuration view, double-click the failsafe I/O module to be configured. The following takes the Hi50-0004ETP-F2A module as an example.
2. In the configuration page, select the "General configuration" tab to configure module parameters and channel parameters.



Parameter		Description
Module general configuration	Post-fault character	Includes two modes: module enters safe state/channel enters safe state
	Recovery method after fault	Includes three modes: custom/manual/automatic ◦ When set to manual or automatic, the channel parameter "Recovery method after fault" is consistent with the setting. ◦ When set to custom, the channel parameter "Recovery method after fault" can be set to automatic or manual. When "Post-fault character" is set to "Module enters safe state", the "Custom" option is not visible.
	Maximum test interval	Options: 100 ms, 200 ms, 500 ms, 1000 ms (default), 2000 ms, 5000 ms, 10000 ms
	Module power supply group	Supports using power supply expansion modules (default) or power supply start modules.
	Status readback	Enables or disables the status readback function. When enabled, input connections of output modules are activated.
Channel (0 to 3) configuration	Enable channel	Sets whether the channel is enabled.
	Enable shutdown test pulse diagnosis	Enables or disables the shutdown test pulse diagnosis function.
	Shutdown test maximum readback time	Options: 0.5 ms, 1 ms (default), 2 ms, 3 ms, 4 ms, 5 ms
	Wiring break	Enables or disables disconnection diagnosis.
	Overload	Enables or disables overload diagnosis.

Parameter		Description
	Recovery method after fault	◦ When the module parameter "Post-fault recovery method" is set to automatic or manual, this setting follows accordingly and cannot be modified. ◦ When the module parameter "Post-fault recovery method" is set to custom, this setting supports manual selection between automatic and manual.
Safety signature	Safety configuration CRC	Displays the safety configuration CRC information of the module.
	Safety configuration timestamp	Displays the safety configuration timestamp information of the module.

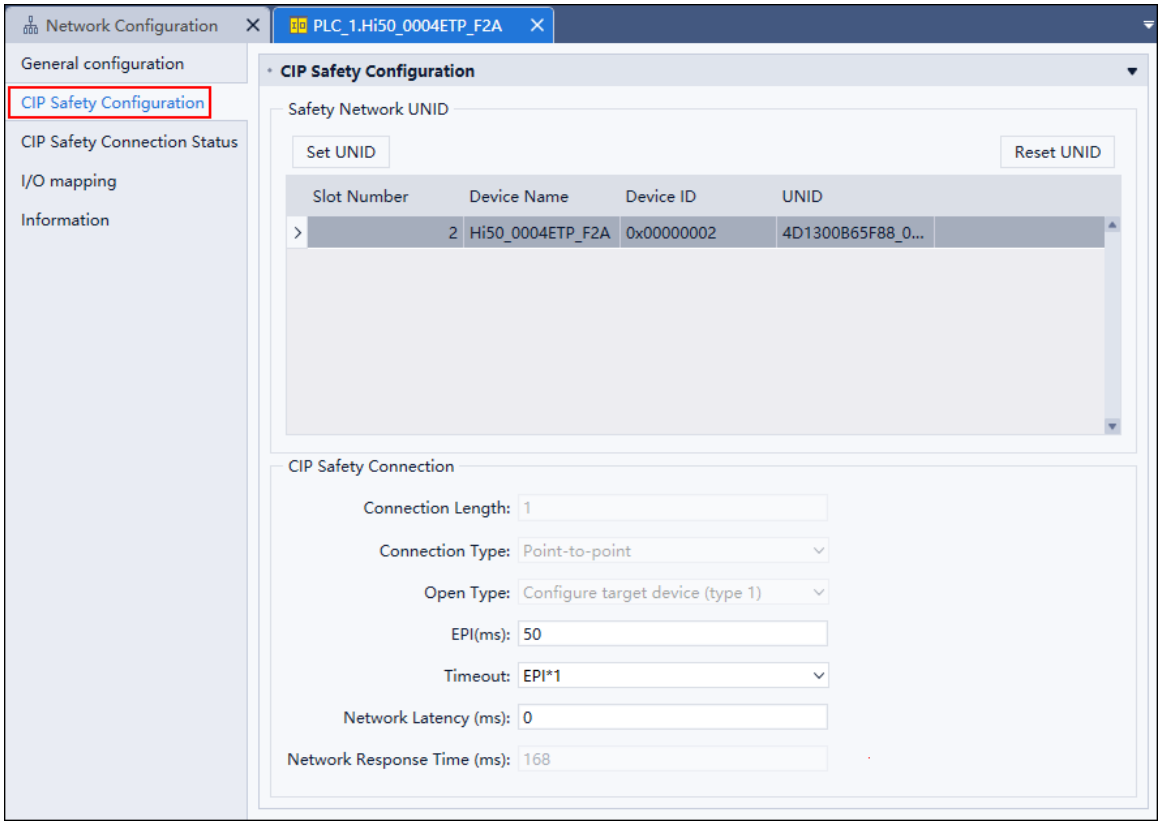


See the Hi50 series failsafe I/O module manuals for parameter definitions and functions of each failsafe I/O module.

CIP Safety configuration

This tab is used to configure the CIP Safety communication parameters for failsafe I/O modules, including EPI, timeout multiplier, network latency, and so on.

1. Double-click the failsafe I/O module to be configured in the project tree or device configuration view.
2. In the configuration page, select the "CIP Safety configuration" tab to configure the CIP Safety connection parameters of the module.
   - EPI (ms): Indicates the cycle of CIP Safety communication between the safety PLC and the safety I/O module. This parameter must be an integer multiple of the task cycle. For example, if the task cycle is 50 ms, this parameter must be 50 ms, 100 ms, 150 ms, and so on. By default, this parameter equals the task cycle.
   - Timeout: Indicates the timeout multiplier for CIP Safety communication between the safety PLC and the safety I/O module. Options include EPI*1 (default), EPI*2, EPI*3, and EPI*4.
   - Network latency (ms): Indicates the network delay time, which should be configured based on the network environment. Default is 0 ms.
   - Network response time (ms): Indicates the timeout time for CIP Safety communication between the safety PLC and the safety I/O module. If no communication response is received within this time, the safety PLC and safety I/O module will enter a safe status and cannot be modified. This parameter is used to calculate the system response time.

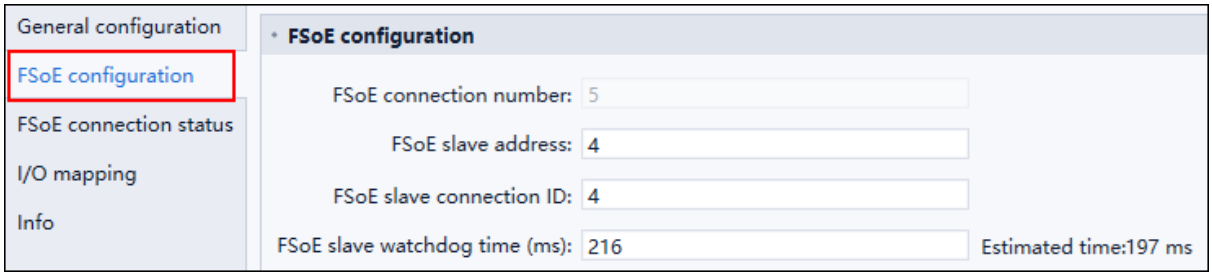


FSoE configuration

This tab is used to configure the FSoE communication parameters for the failsafe I/O module.

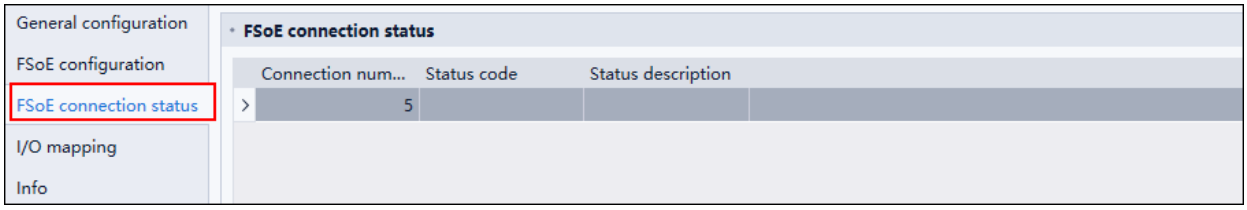
1. Double-click the failsafe I/O module to be configured in the project tree or device configuration view.
2. On the "FSoE configuration" tab, configure the FSoE connection parameters for the module.

Parameter	Value Range	Note
FSoE connection ID	1 to 65535	The connection ID established in the safety PLC is automatically assigned and cannot be modified by the user.
FSoE slave address	1 to 65535	Entered manually by the user.
FSoE slave connection ID	1 to 65535	Generated by the iFA. It must be unique in the FSoE network and can be modified by the user.
FSoE slave watchdog time (ms)	1 to 65535	The default value is calculated by a formula. It can be modified by the user but cannot be less than the estimated value.



CIP Safety connection status / FSoE connection status

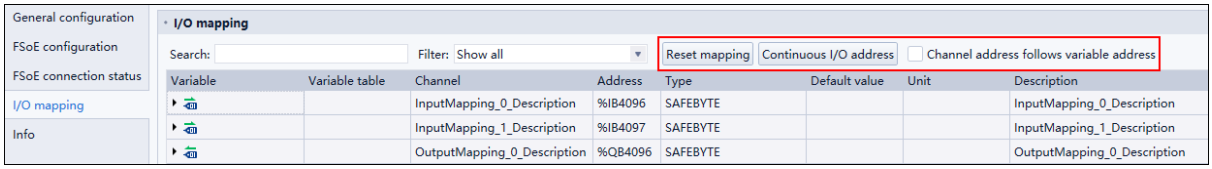
This tab displays the current connection status of the module, including status codes and descriptions.



I/O Mapping

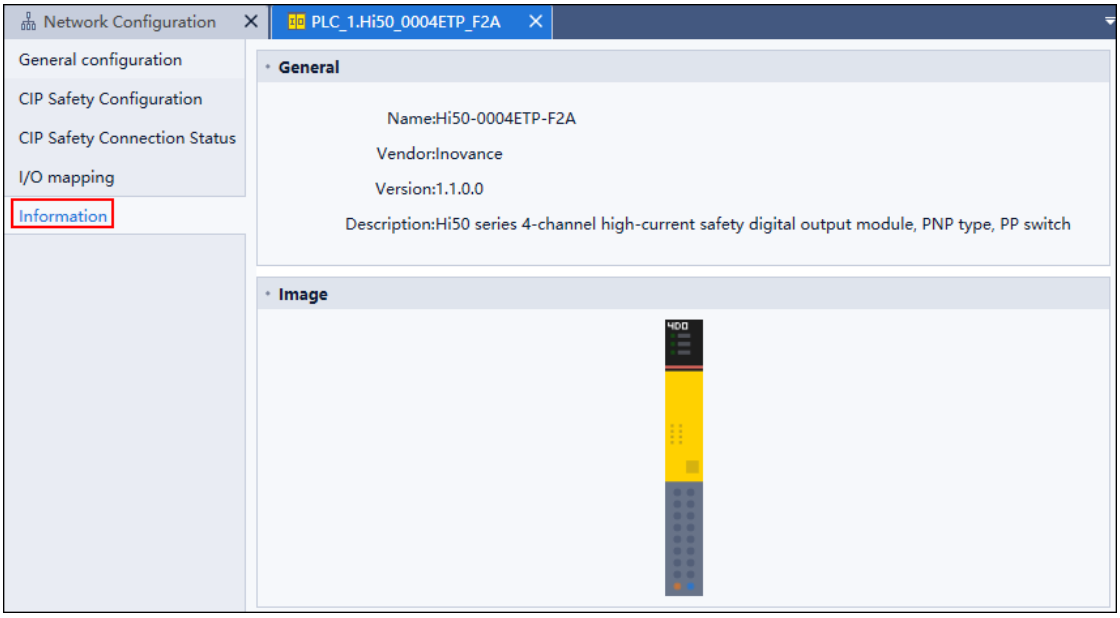
This tab is used to view I/O mapping variable information for on-site signal acquisition. The information supports keyword search, condition-based filtering, and resetting. For detailed configuration parameter descriptions, see 10.6.3.3.9 I/O Mapping. A structure can be directly bound to failsafe I/O mappings for quick binding.

- Binding a structure is only supported when "Channel address follows variable address" is selected.
- For a successful binding, the member layout order of the defined structure must exactly match the variable order and data length of the I/O mapping.
- For safety variables, the variable property, variable type, and data length must be completely consistent.
- Cross-group matching between structures and I/O mappings is supported. For example, you can define a structure variable whose members have the same order and types as the variables in the I/O mapping area, allowing all I/O mappings to be bound in one step.
- After successful binding, the default address assigned to this I/O mapping becomes invalid.



Information

This tab displays general information and images of the I/O module.



11.4.3 EtherNet/IP Communication Configuration

11.4.3.1 Standard EtherNet/IP Communication Configuration

11.4.3.1.1 EtherNet/IP Communication Between PLCs

The EVO1000EF series safety PLC supports communication with standard PLCs via the EtherNet/IP protocol. Users can configure communication through network configuration. Communication configuration includes the following two scenarios:

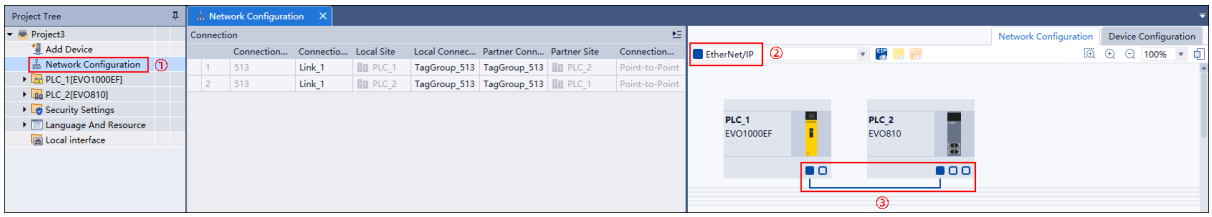
- The safety PLC and standard PLC devices are in the same project.
- The safety PLC and standard PLC devices are in different projects.

NOTE

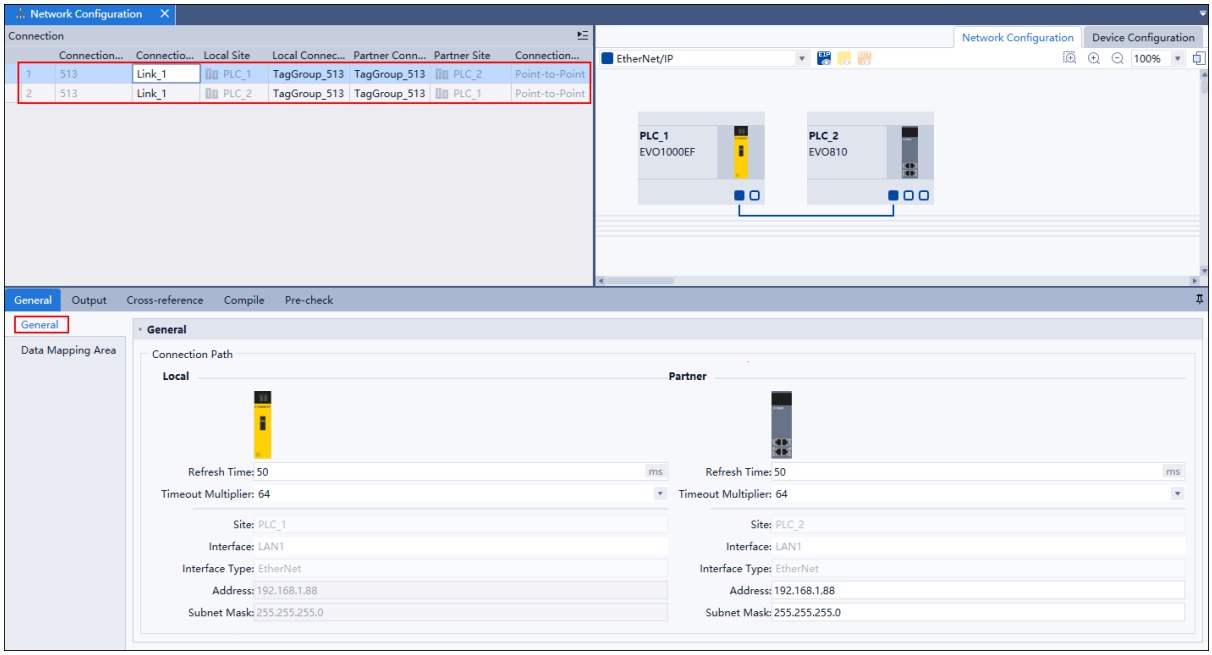
- Only the LAN1 port of the safety PLC supports EtherNet/IP communication with standard PLCs.
- Data sent from the safety PLC to the standard PLC can be either of safety type or standard type.
- Data received by the safety PLC from the standard PLC can only be of standard type; otherwise, the safety PLC program will report a compilation fault.
- The LAN1 port of the safety PLC and the network port of the standard PLC are in the same network segment.
- The LAN1 port and the EIP2 network port of the safety PLC can be in the same network segment, but their IP addresses must not be the same.

EtherNet/IP communication configuration for PLCs in the same project

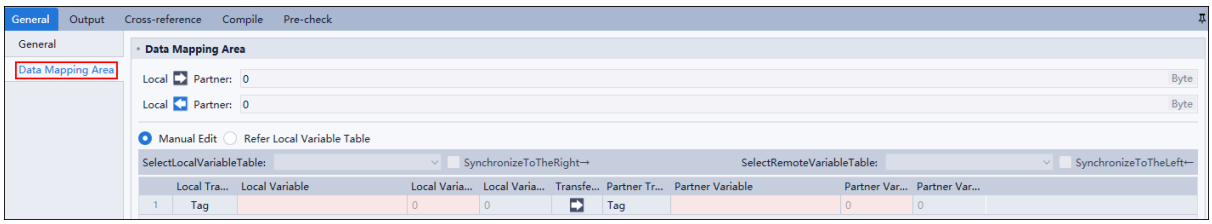
1. Create a project and add a safety PLC and standard PLC (using the EVO800 series large-scale PLC as an example).
2. Double-click "Network configuration" in the project tree, and select "EtherNet/IP" communication from the drop-down menu.
3. Select the LAN1 port of the safety PLC, drag and connect it to the network port of the large-scale PLC to establish an EtherNet/IP connection.



4. In the "Connection" area, the newly created EtherNet/IP connection information is displayed, including connection ID, connection name, local/partner site, local/partner connection ID, and connection method.
5. Select the connection, and in the "General" tab below, configure the EtherNet/IP communication refresh cycle and timeout multiplier, and verify the IP address and subnet mask information.



6. In the "Data mapping area" tab, configure the data mapping relationship between the communication parties by binding variables. Click the cell in the "Transfer direction" column to switch the data transfer direction.

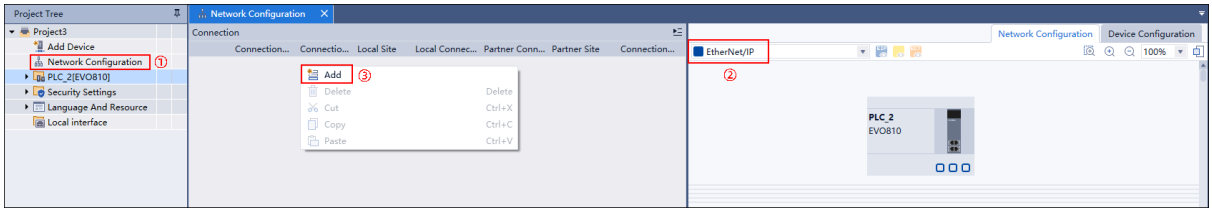


> **NOTE**
>
> The above drag-and-drop connection method only applies to "point-to-point" EtherNet/IP communication.

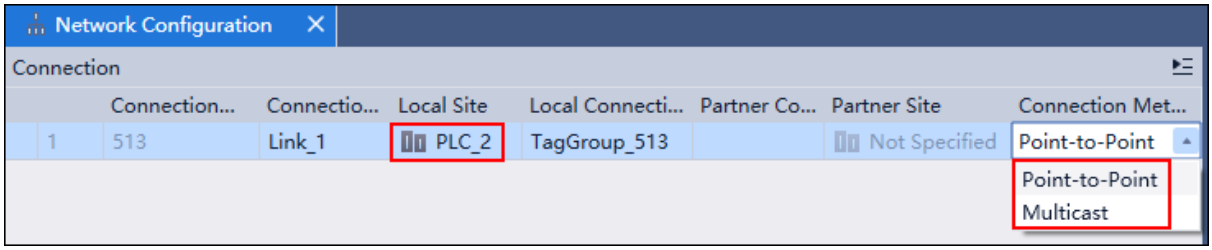
EtherNet/IP communication configuration for PLCs in different projects

- **Standard PLC configuration (using the EVO800 series large-scale PLC as an example)**

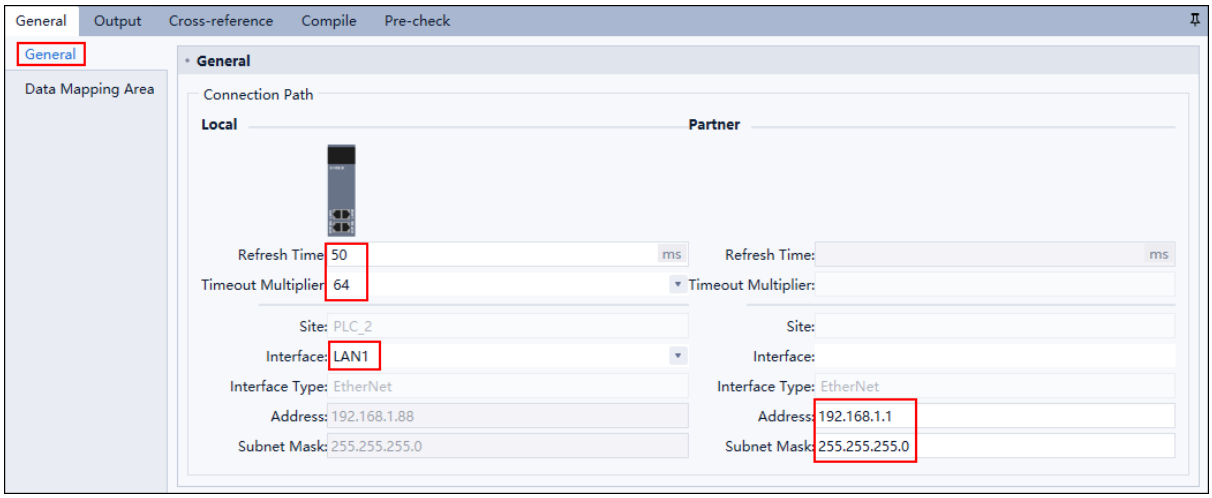
1. Create a project and add the EVO800 series large-scale PLC.
2. Double-click "Network configuration" in the project tree, and select "EtherNet/IP" communication from the drop-down menu.
3. Right-click a blank area in the "Connection" area, and select "Add" to create an EtherNet/IP connection.



4. In the newly created EtherNet/IP connection, select the large-scale PLC used for communication as the "Local site", and select "Point-to-point" or "Multicast" as the "Connection method".



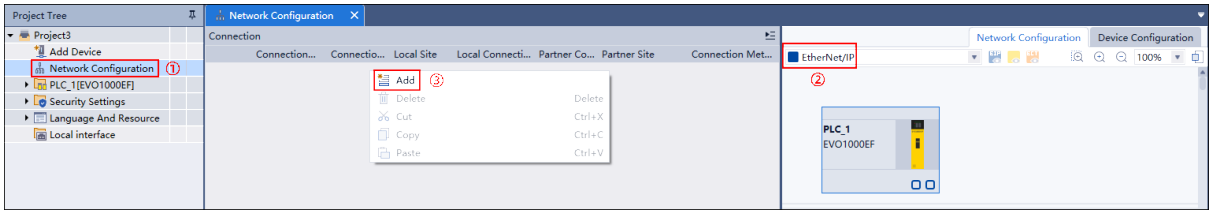
5. Select the connection, and in the "General" tab below, configure the EtherNet/IP communication refresh cycle and timeout multiplier.
6. In the "Interface" drop-down menu, select the network port for communication from "LAN1", "LAN2", or "EIP3".
7. Enter the IP address and subnet mask of the target safety PLC in the fields for partner address and subnet mask.



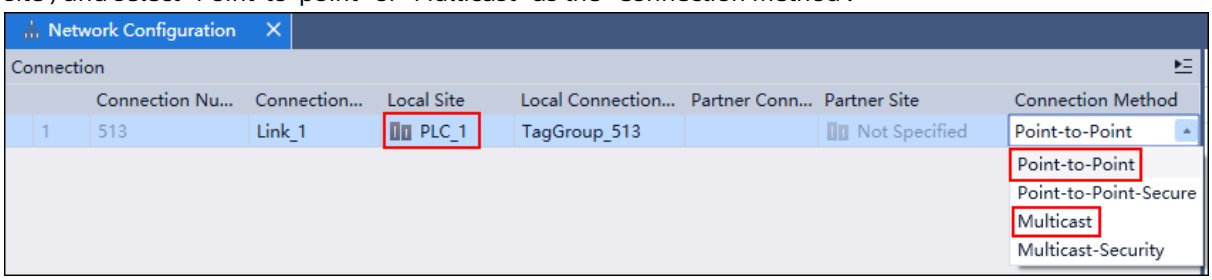
8. In the "Data mapping area" tab, configure the data mapping relationship between the communication parties by binding variables. Click the cell in the "Transfer direction" column to switch the data transfer direction.

- **EVO1000EF series safety PLC configuration**

1. Create a project and add a safety PLC device.
2. Double-click "Network configuration" in the project tree, and select "EtherNet/IP" communication from the drop-down menu.
3. Right-click a blank area in the "Connection" area, and select "Add" to create an EtherNet/IP connection.



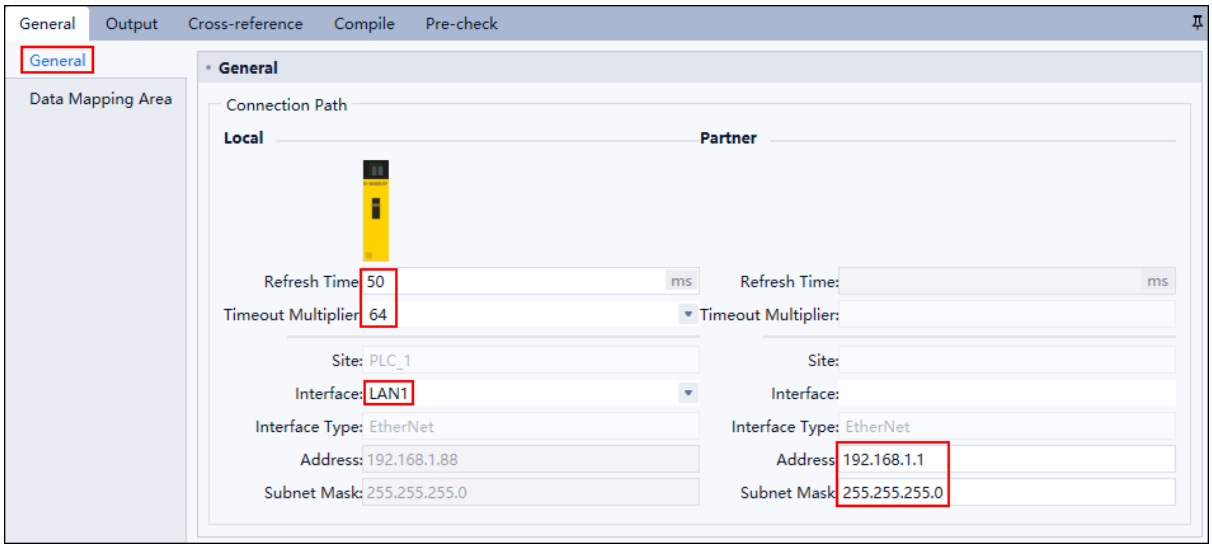
4. In the newly created EtherNet/IP connection, select the safety PLC used for communication as the "Local site", and select "Point-to-point" or "Multicast" as the "Connection method".



NOTE

The "Point-to-point-safety" and "Multicast-safety" connection methods indicate communication based on the CIP Safety protocol. The CIP Safety protocol is supported only by safety PLCs. For details, see 11.4.3.2.2 Safety CIP Safety Communication.

5. Select the connection, and in the "General" tab below, configure the EtherNet/IP communication refresh cycle and timeout multiplier.
6. In the "Interface" drop-down menu, only "LAN1" can be selected.
7. Enter the IP address and subnet mask of the target large-scale PLC in the fields for partner address and subnet mask.



8. In the "Data mapping area" tab, configure the data mapping relationship between the communication parties by binding variables. Click the cell in the "Transfer direction" column to switch the data transfer direction.

11.4.3.1.2 EtherNet/IP Communication with HMI

The safety PLC only supports connecting to HMI via the LAN1 port of the EIP or ECT communication unit. When the safety PLC communicates with HMI via EtherNet/IP, an EIP Adapter service node is generated under the LAN1 port of the safety PLC. The configuration method for EtherNet/IP communication between the safety PLC and HMI is the same as those for the standard PLC. Only the functional differences are described here. For other common functions, see 10.6.4.2.7 Communication with HMI in the "PLC Programming" chapter.

NOTE

- The HMI supports reading and writing standard data type variables in the safety PLC, but only supports reading safety data type variables.
- When the HMI attempts to write to a safety variable, the safety PLC runtime rejects the operation and returns a failure.
- For the first download, the safety PLC project configured with the EtherNet/IP Adapter must be downloaded before the HMI can be downloaded.

11.4.3.2 Safety CIP Safety Communication Configuration

11.4.3.2.1 Overview

CIP Safety

CIP Safety is a safety communication extension based on CIP (Common Industrial Protocol), primarily used for secure data transmission between safety devices in industrial automation. It is especially suitable for communication scenarios between safety PLCs and other safety devices (including but not limited to safety PLCs), with transmitted data limited to safety types (such as emergency stop signals, safety door status, and safety interlock commands). It is designed to enable reliable interaction of safety data over standard CIP networks (such as EtherNet/IP) without requiring a dedicated safety network, while strictly ensuring communication security.

Core features

- Communication objects and data types
  CIP Safety is specifically designed for interaction between safety devices (such as safety PLCs and safety I/O modules). It only transmits safety-related data (non-safety data is transmitted via standard CIP channels), ensuring logical independence of safety data.
- Safety mechanisms
  Based on the characteristics of safety data, CIP Safety applies CRC check, device identity authentication, timestamp anti-replay, and timeout detection to ensure data integrity, authenticity, and timeliness, eliminating safety risks caused by communication anomalies (such as data tampering, delay, or forgery).
- Core roles in safety PLCs
  CIP Safety serves as a critical channel for transmitting safety commands and statuses between the safety PLC and upstream/downstream safety devices (such as another safety PLC or safety I/O modules), ensuring the reliable implementation of core safety functions such as emergency stop and safety interlock. By limiting the scope of safety data transmission and enhancing the safety interaction mechanism between devices, CIP Safety provides efficient and compliant communication assurance for safety PLCs within safety control systems, serving as a key technology for industrial functional safety systems.

CIP Safety Scanner

CIP Safety Scanner is a core component responsible for managing and controlling safety device networks in industrial automation safety communications. Integrated in safety PLCs, it acts as the "master device" in the CIP Safety communication architecture. It is primarily used to configure and monitor safety slave devices (such as Hi50 couplers and remote safety I/O modules) in the network, establish and maintain safety communication connections with these devices. Through standardized safety communication logic, it coordinates data interaction between safety devices and ensures the functional safety of industrial control systems.

11.4.3.2.2 CIP Safety Communication Between PLCs

CIP Safety communication is supported among multiple safety PLCs. Users can configure communication via network configuration. Communication configuration includes the following two scenarios:

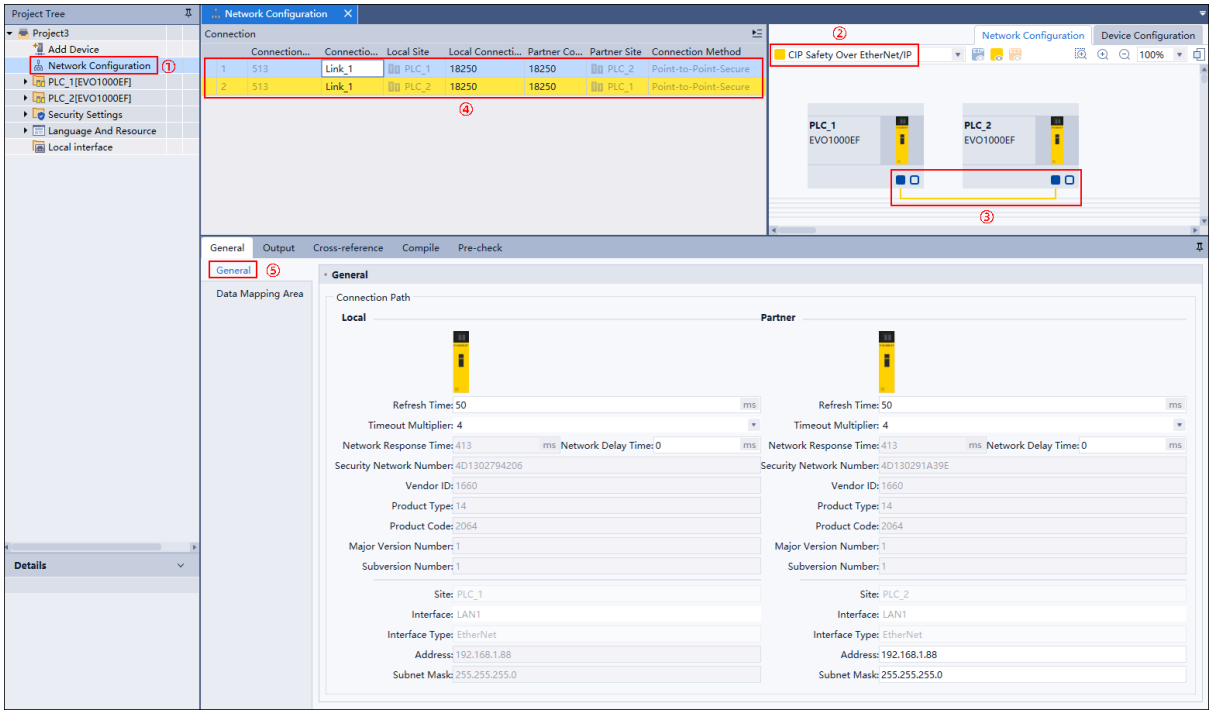
- Multiple safety PLCs are in the same project.

- Multiple safety PLCs are in different projects.

> **NOTE**
>
> - When safety PLC devices communicate via CIP Safety, only safety-type data can be transmitted.
> - Both the LAN1 and EIP2 ports of the safety PLC support CIP Safety communication, with each port supporting up to 16 safety connections.

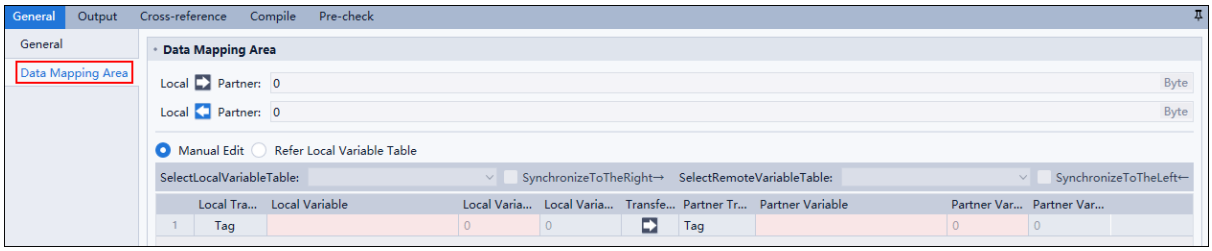
CIP Safety communication configuration for safety PLCs in the same project

1. Create a project and add two safety PLC devices.
2. Double-click "Network configuration" in the project tree, and select "CIP Safety Over EtherNet/IP" communication from the drop-down menu.
3. Connect the LAN1 or EIP2 ports of the two safety PLCs by dragging to establish CIP Safety communication between the two safety PLCs. The following takes the LAN1 port as an example.
4. In the "Connection" area, the newly created CIP Safety connection information is displayed, including connection ID, connection name, local/partner site, local/partner connection ID, and connection method.
5. Select the connection, and in the "General" tab below, configure the CIP Safety communication parameters, including refresh time, timeout multiplier, network latency time, IP address, and subnet mask.



> **CAUTION**
>
> The "Refresh time" must be a common multiple of the task cycles of both safety PLCs; otherwise, communication timeout may occur.

6. In the "Data mapping area" tab, configure the data mapping relationship between the communication parties by binding variables. Click the cell in the "Transfer direction" column to switch the data transfer direction.

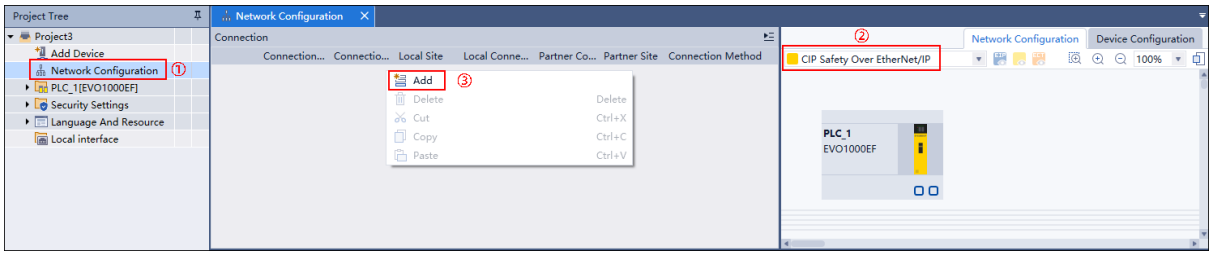


NOTE

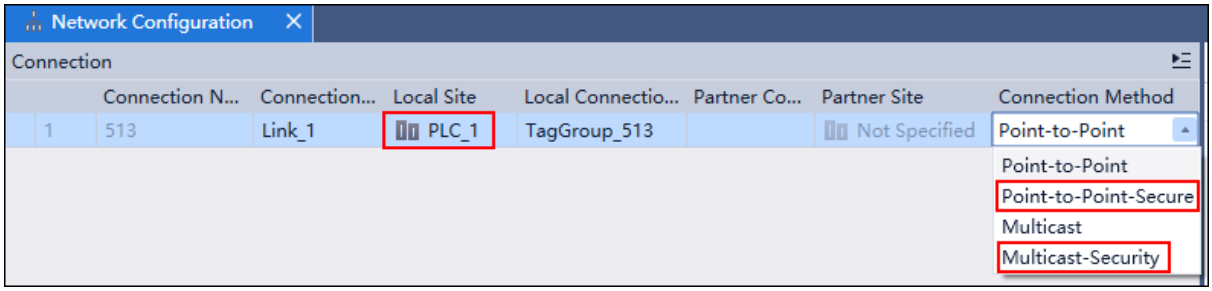
The above drag-and-drop connection method only applies to "Point-to-point-safety" CIP Safety communication.

CIP Safety communication configuration for safety PLCs in different projects

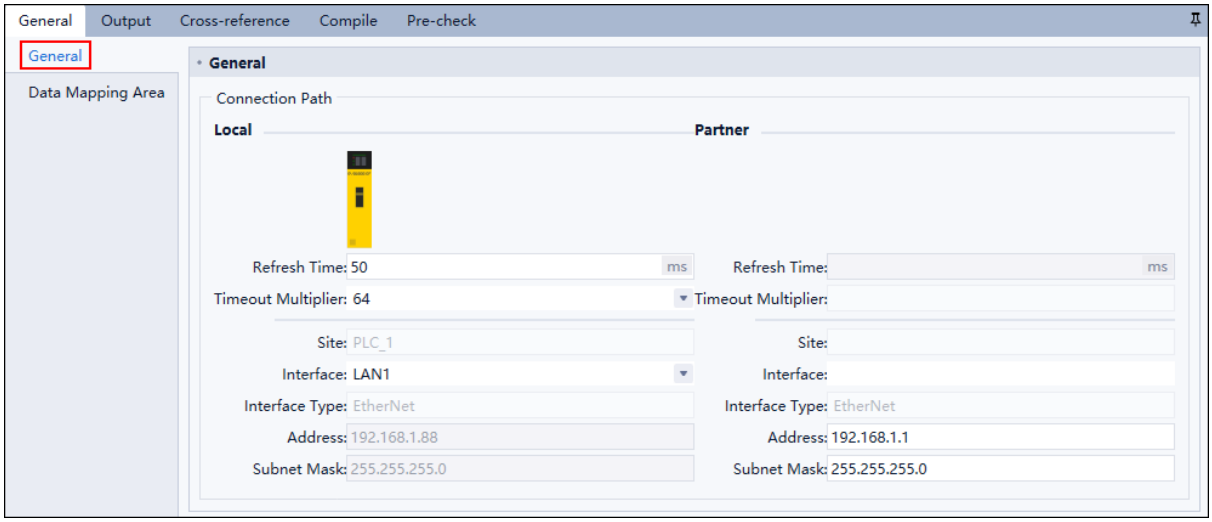
1. Create a project and add a safety PLC device.
2. Double-click "Network configuration" in the project tree, and select "CIP Safety Over EtherNet/IP" communication from the drop-down menu.
3. Right-click a blank area in the "Connection" area, and select "Add" to create a CIP Safety connection.



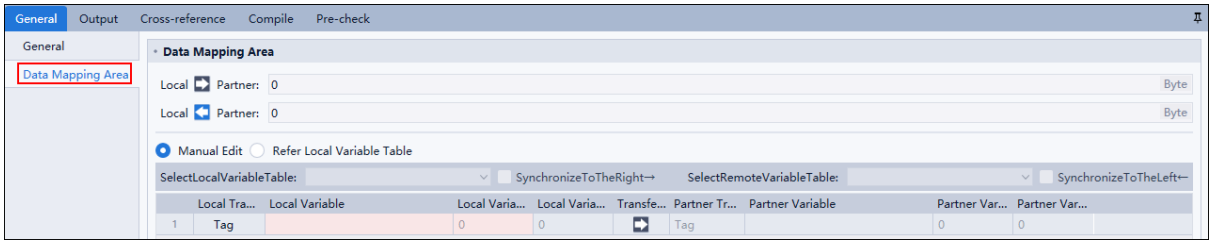
4. In the newly created connection, select the safety PLC used for communication as the "Local site," and select "Point-to-point-safety" or "Multicast-safety" as the "Connection method".



5. Select the connection, and in the "General" tab below, configure the communication parameters for the local and partner PLCs.
   The communication parameters for the local PLC include refresh time, timeout multiplier, network delay time, and interface. The device parameters for the partner PLC include safety network number, vendor ID, product type, IP address and subnet mask.



6. In the "Data mapping area" tab, configure parameters such as local variables and transfer direction.

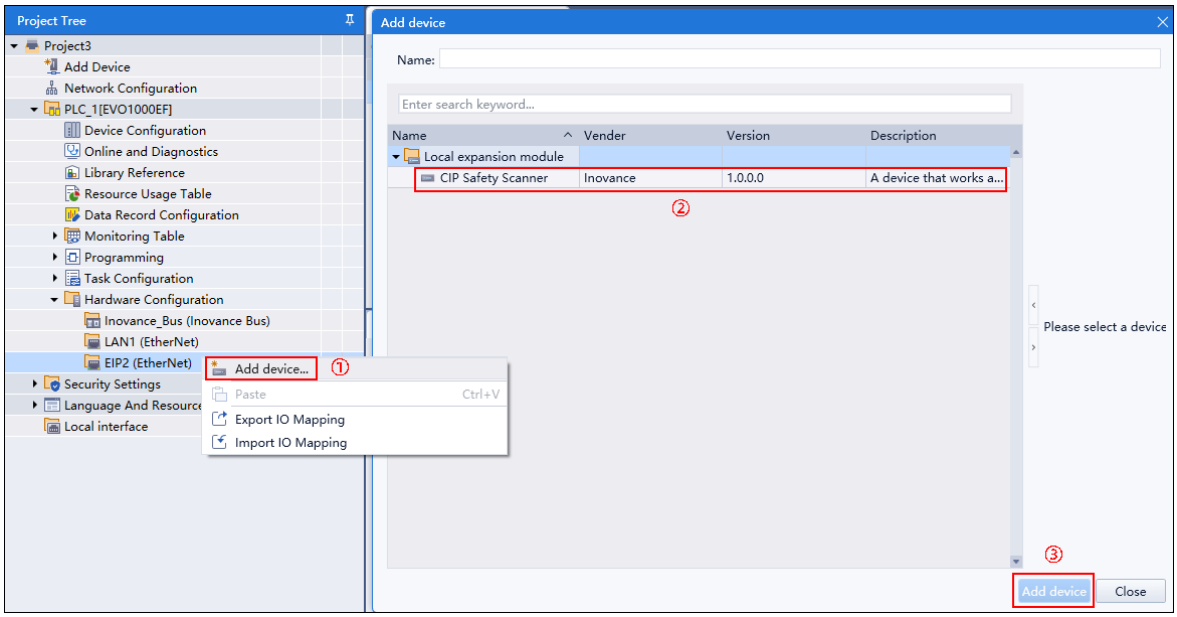


11.4.3.2.3 Adding a Remote CIP Safety Device

The Hi50-RTU-EIP-R00 module is a Hi50 series EIP coupler, used to add and manage remote safety I/O modules. The addition methods for the coupler include manual addition in the project tree and device scanning. Besides these two methods, remote safety I/O modules can also be added via the device configuration view.

Manual addition

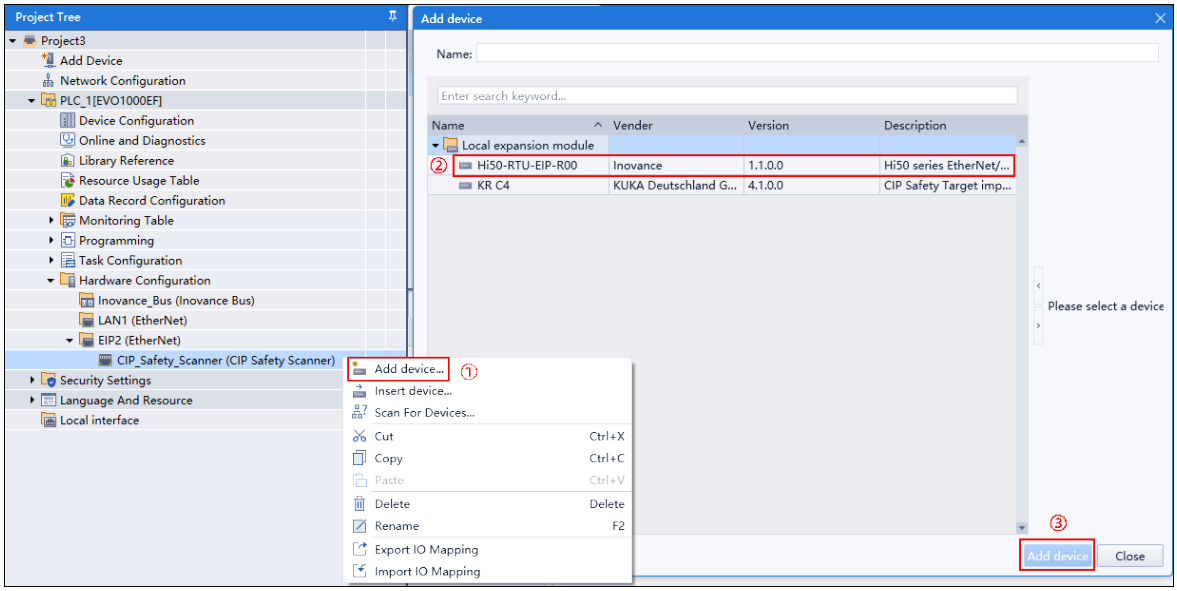
1. Add a "CIP_Safety_Scanner" node.
   a. Right-click the "EIP2 (EtherNet)" node under "Hardware configuration" node in the project tree, and select "Add device".
   b. In the pop-up window, select "CIP_Safety_Scanner", and click "Add device".



2. Add the Hi50-RTU-EIP-R00 coupler.

   a. Right-click the "CIP_Safety_Scanner" node, and select "Add device".

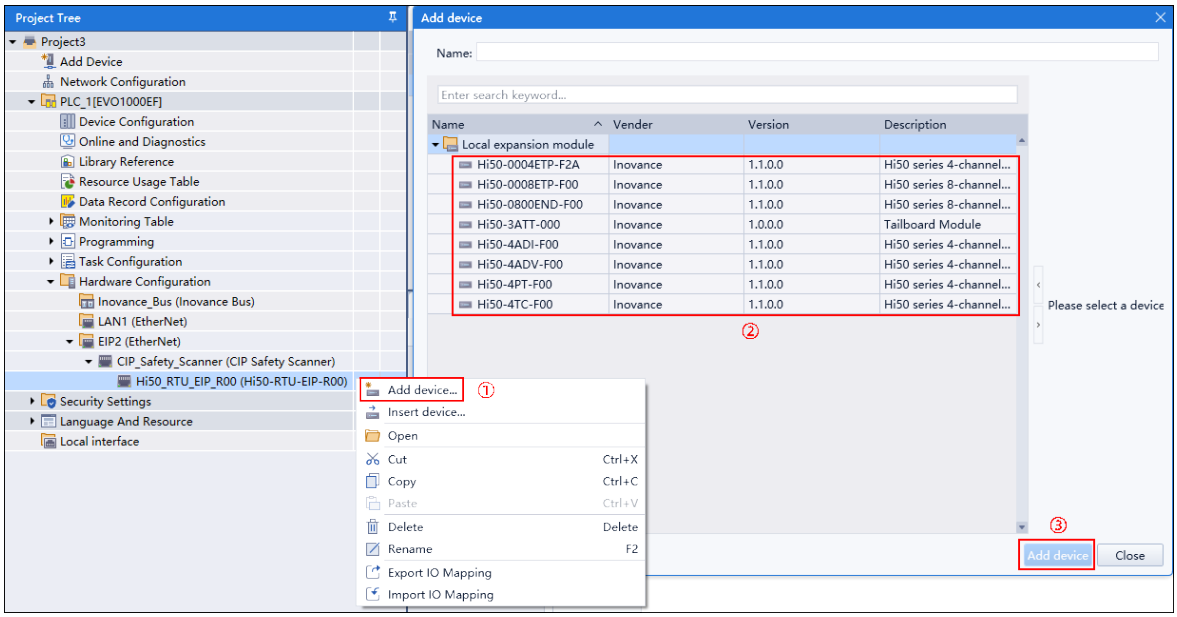
   b. In the pop-up window, select "Hi50-RTU-EIP-R00", and click "Add device".



3. Add a remote safety I/O module.

   a. Right-click the "Hi50-RTU-EIP-R00" node, and select "Add device".

   b. In the pop-up window, select the required safety I/O module, and click "Add device".



Addition via device scanning

The iFA Evolution software provides an automatic scanning function for safety I/O modules, allowing users to directly add modules by scanning.

Prerequisites

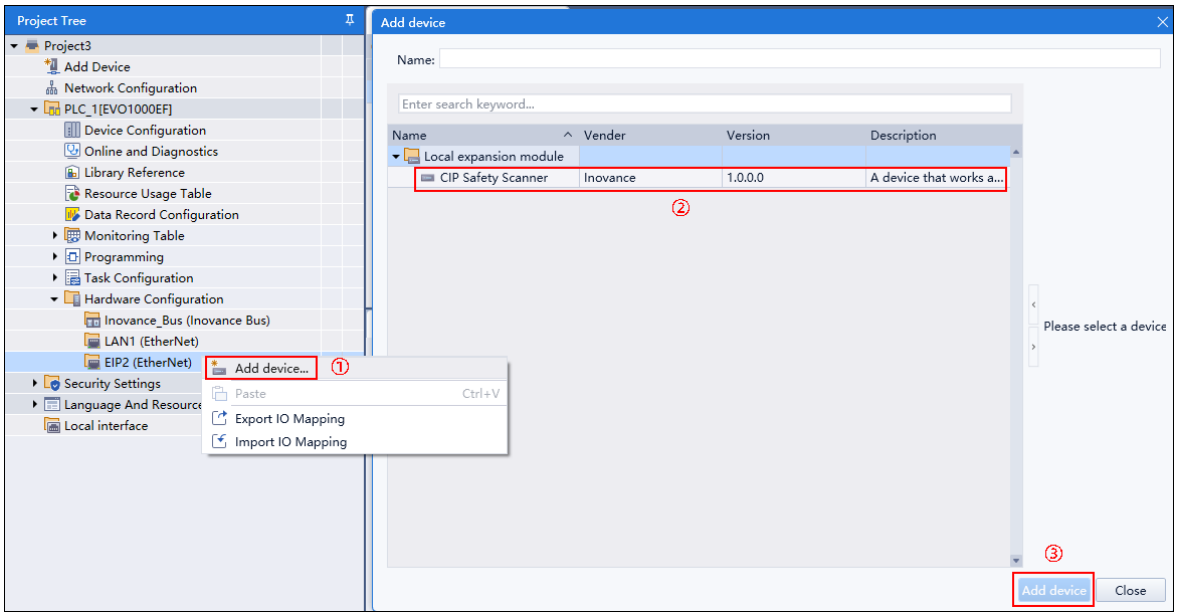
- A blank project containing the "CIP_Safety_Scanner" node has been deployed to the target safety PLC.
- The user program of the safety PLC is in the stopped state.

Procedure

1. Add a "CIP_Safety_Scanner" node.

   a. Right-click the "EIP2 (EtherNet)" node under "Hardware configuration" node in the project tree, and select "Add device".

b. In the pop-up window, select "CIP_Safety_Scanner", and click "Add device".



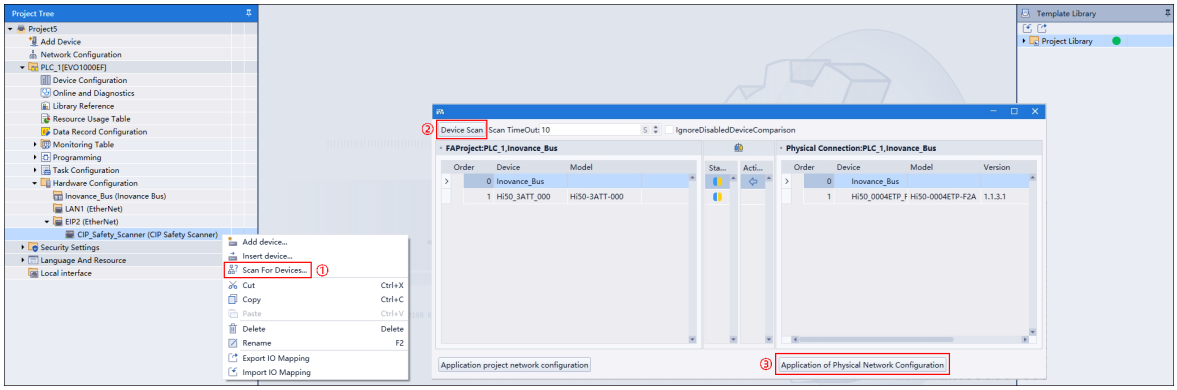
2. Add a remote safety I/O module via device scanning.

a. Right-click the "CIP_Safety_Scanner" node, and select "Scan for devices".

b. In the pop-up window, click "Device scanning" and synchronize all scanned I/O modules into the current project.

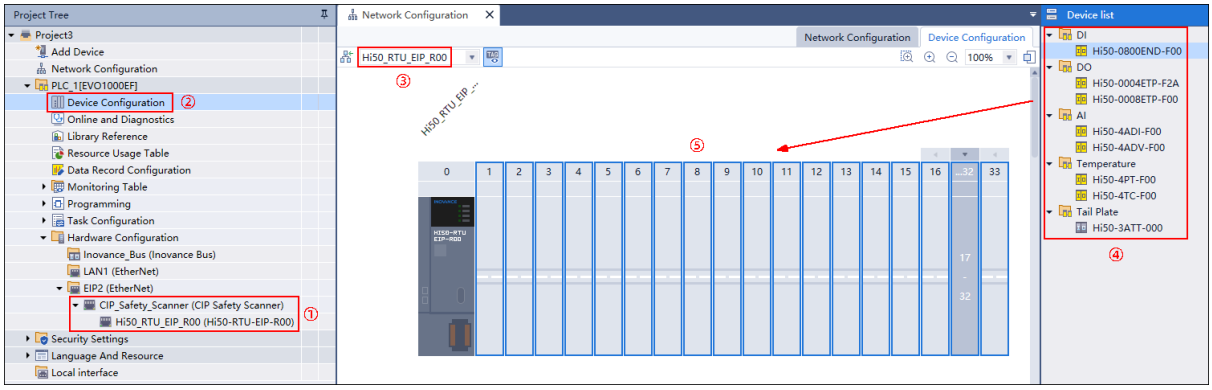
c. In the pop-up confirmation window, click "Yes" to restore the synchronized device configuration to default.

d. Click "OK" to add the safety I/O module to the project.



Addition via device configuration view

1. Add the "CIP_Safety_Scanner" node and the "Hi50-RTU-EIP-R00" coupler under "Hardware configuration > EIP2 (EtherNet)" of the corresponding device in the project tree.
2. Double-click the "Device configuration" node in the project tree.
3. In the device configuration view, select the "Hi50-RTU-EIP-R00" coupler device.
4. In the right device list, select the module. When a module is selected, the insertable slots for the module are highlighted in the configuration diagram.
5. In the device list, double-click a module to add it to the nearest available slot adjacent to the PLC. You can also drag the module from the device list to an available slot in the device configuration view.

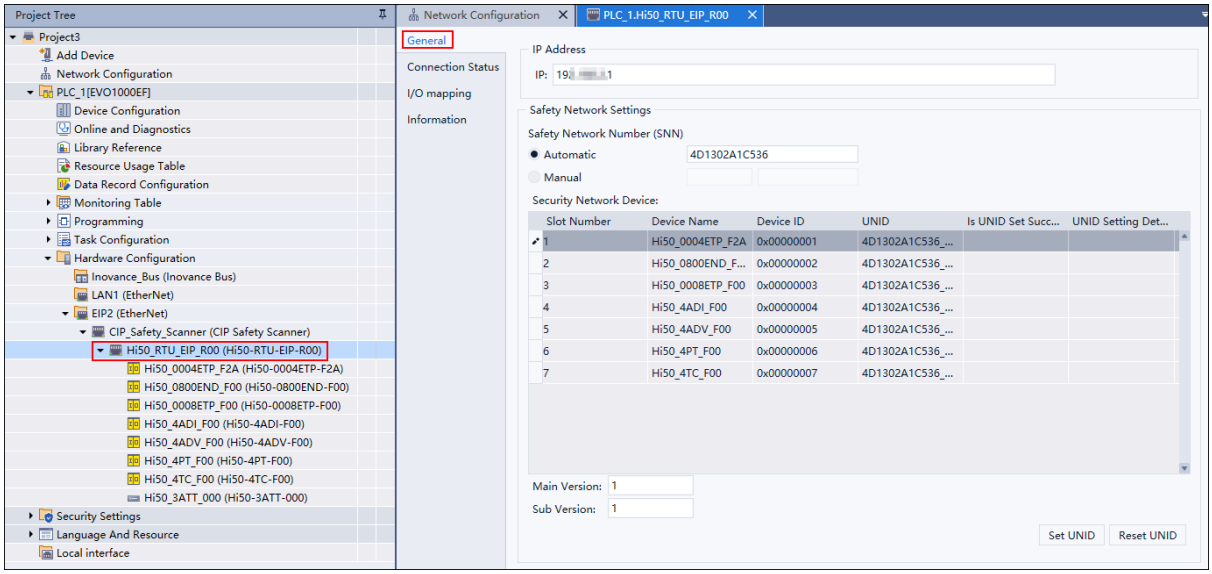


11.4.3.2.4 Configuring a Remote CIP Safety Device

The "CIP_Safety_Scanner" node is used to add safety Hi50 couplers (such as Hi50-RTU-EIP-R00) and configure remote safety I/O modules. The safety Hi50 coupler supports configuration of IP address, safety network number (SNN), and setting UNID for remote safety I/O modules. Configuration of remote safety I/O modules is the same as for local safety I/O modules. See 11.4.2.7 Safety I/O Module Configuration for details. The following describes the configuration related to the safety Hi50 coupler.

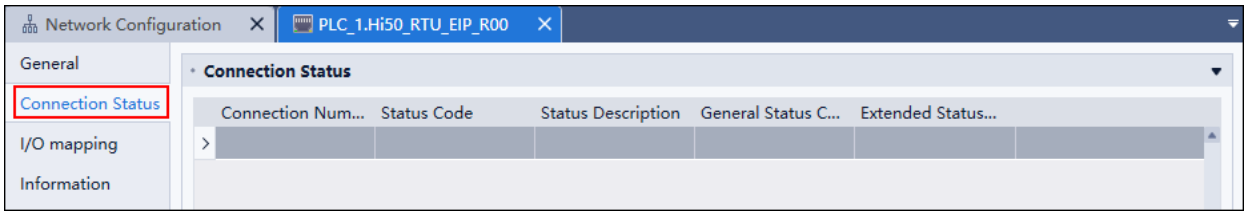
General settings

1. Double-click the "Hi50-RTU-EIP-R00" node under "Hardware configuration > EIP2" in the project tree.
2. In the pop-up coupler configuration page, select the "General" tab.
3. IP address: The coupler's IP address must be in the same subnet as the safety PLC device.
4. Safety network number (SNN): Supports automatic and manual assignment methods.
   - Automatic assignment: SNN is automatically generated based on the current system time.
   - Manual assignment: The first input box is fixed at 1, and the second allows values from 1 to 9999.
5. The safety network device list below displays slot numbers, device names, device IDs, UNIDs, and other information for each remote safety I/O module.
   When adding remote safety I/O modules, UNIDs are automatically assigned. After modifying the SNN, the UNID must be reset before configuration so that the new UNID can be successfully assigned to each module.
6. Click the "Reset UNID" button and confirm that the UNIDs of all remote safety I/O modules have been successfully reset.
7. Click the "Set UNID" button and confirm that the UNIDs of all remote safety I/O modules have been successfully set.



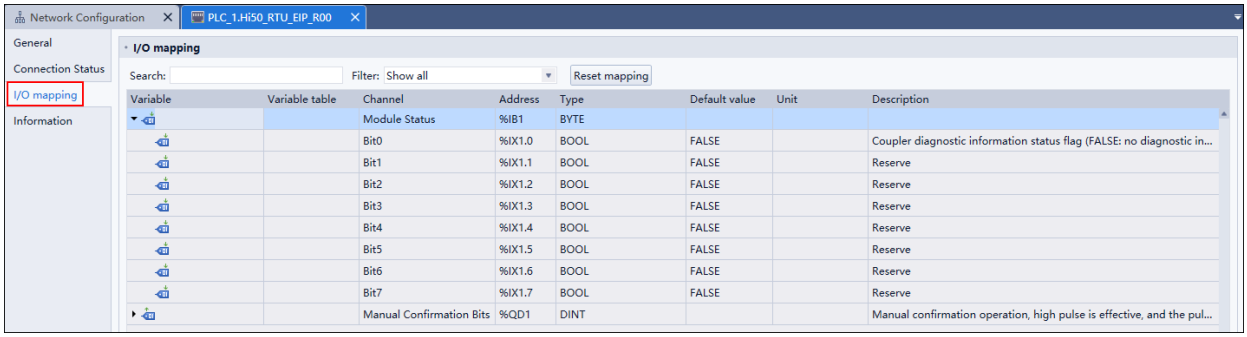
Connection status

This tab displays the coupler's connection status, including status codes and descriptions.



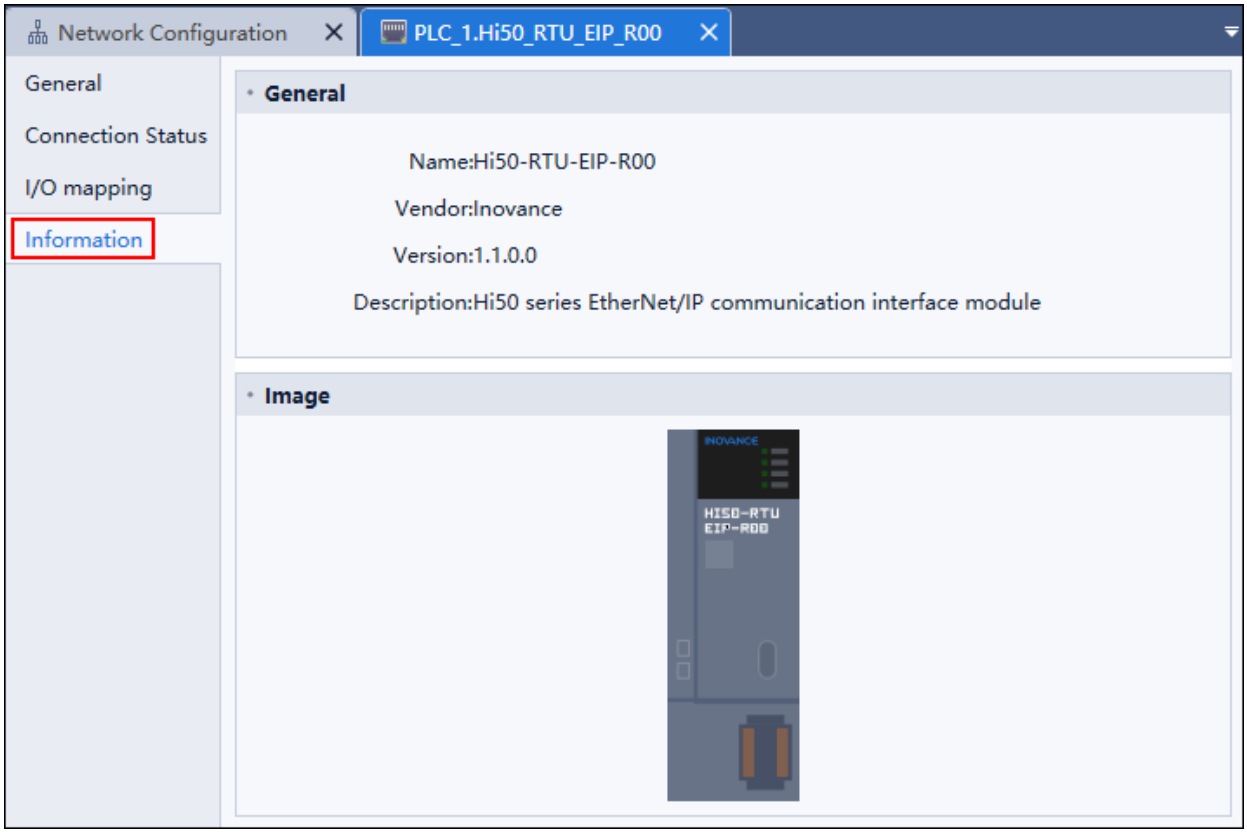
I/O Mapping

This tab is used to view I/O mapping variable information for on-site signal acquisition. The information supports keyword search, condition-based filtering, and resetting. For detailed configuration parameter descriptions, see 10.6.3.2.8 I/O Mapping.



Information

This tab displays the coupler's general information and image.



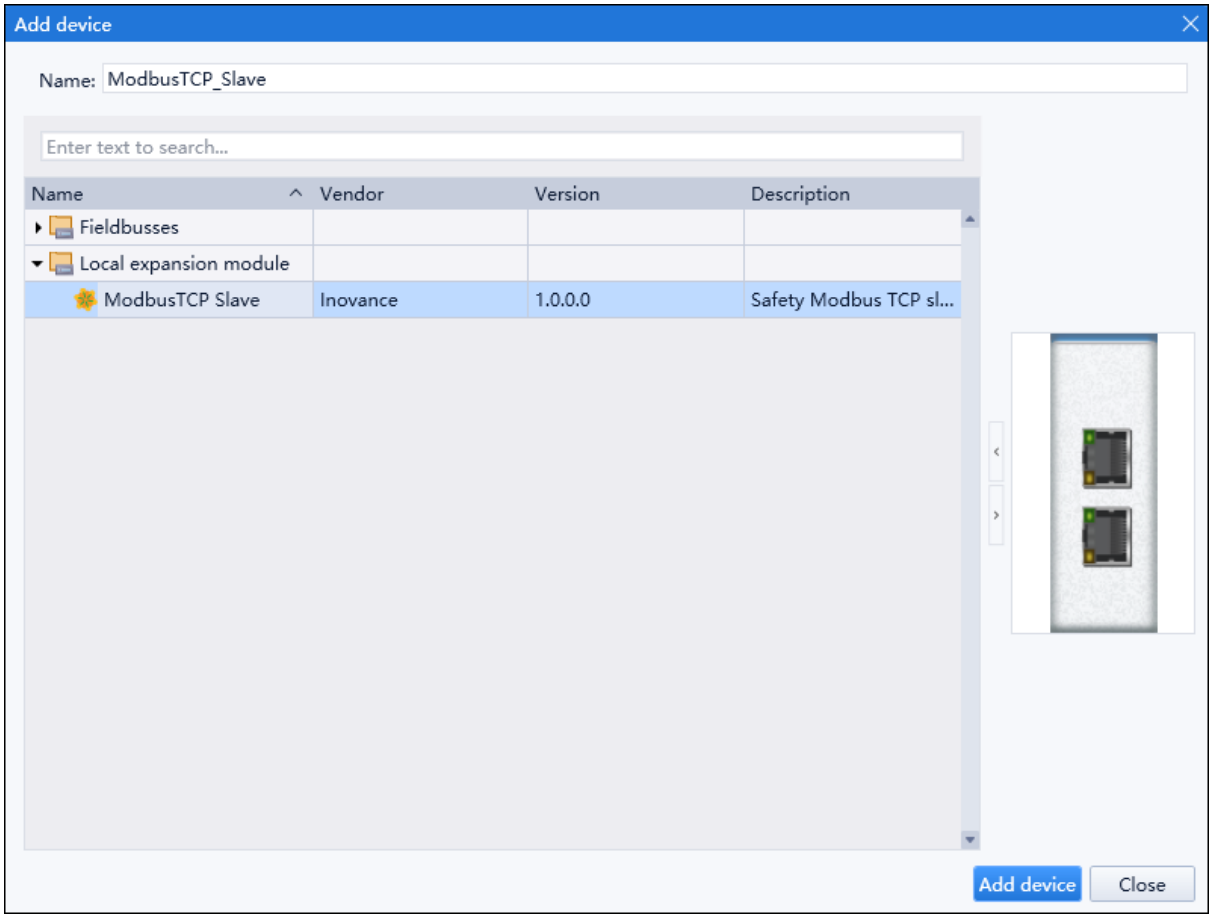
11.4.3.3 Modbus TCP Communication Configuration

The Modbus TCP communication configuration method for a safety PLC is the same as that of a standard PLC. Only the functional differences are introduced here. For other common functions, see 10.6.4.1 Modbus-TCP Communication Configuration in the "PLC Programming" chapter.

- The Modbus TCP Slave only supports configuration-based operation and does not support instruction-based operation.
- The maximum number of variables that the Modbus TCP Master can access is the number of public variables of the safety PLC.

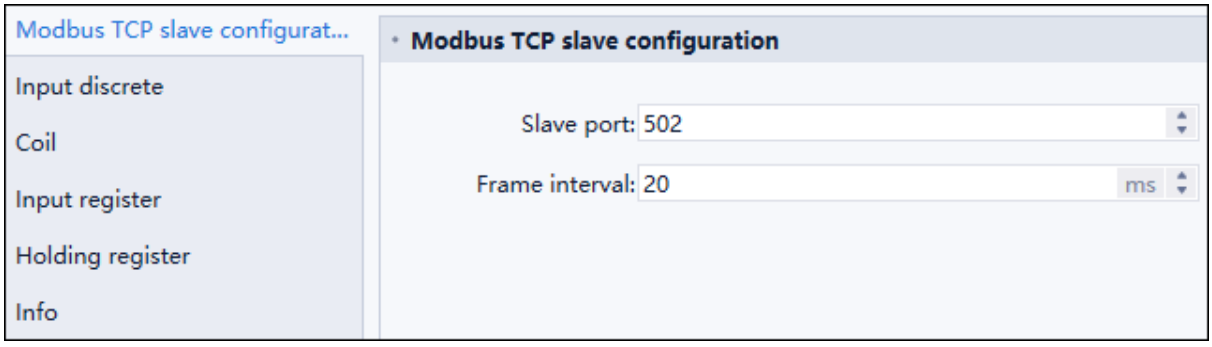
Adding Modbus TCP Slave

1. In the project tree on the left, select "Hardware configuration > Inovance_Bus > LAN1 (EtherNet)" of a safety PLC, and select "Add device".
2. Double-click "ModbusTCP slave" or select it and click "Add device".



Configuring Modbus TCP Slave

1. Double-click the added "Modbus TCP Slave" node to open the "Modbus TCP slave configuration" page by default.



2. Configure the related parameters as needed.

Variables	Requirements
Starting Address	◦ Input discrete values: The input range is 0 to 65535, which is automatically converted to 1x0 to 1x65535. ◦ Coils: The input range is 0 to 65535, which is automatically converted to 0x0 to 0x65535. ◦ Input registers: The input range is 0 to 65535, which is automatically converted to 3x0 to 3x65535. ◦ Holding registers: The input range is 0 to 65535, which is automatically converted to 4x0 to 4x65535.

Variables	Requirements
Modbus address	Automatically generated and can be modified.
Variable	◦ Input discrete values: Select BOOL and SAFEBOOL variables with public property of "Read-Only" or "Read/Write". ▪ Binding method: Drop-down selection, drag-and-drop from the detail view, copy-paste from the variable table. ▪ Exception: If the public property of the current variable does not meet the requirements, or if the selected type is not BOOL or SAFEBOOL, a fault will occur. ◦ Coils: Select BOOL variables with public property of "Write-Only" or "Read/Write". If the public property of the current variable does not meet the requirements, or if the selected type is not BOOL, a fault will occur. ◦ Input registers: Select WORD and INT variables with public property of "Write-Only" or "Read/Write". If the public property of the current variable does not meet the requirements, or if the selected type is not WORD or INT, a fault will occur. ◦ Holding registers: Select WORD and INT variables with public property of "Write-Only" or "Read/Write". If the public property of the current variable does not meet the requirements, or if the selected type is not WORD or INT, a fault will occur.
Data Type	Automatically filled based on the selected variable name, and cannot be modified.
Comment	Automatically filled based on the selected variable name, and cannot be modified.
Variable table	Automatically filled based on the selected variable name, and cannot be modified.
Data read/ write requirements	◦ Modbus Master can perform read and write operations on standard data type variables within the safety PLC. ◦ Modbus Master can only perform read operations on safety data type variables within the safety PLC.

11.4.4 EtherCAT Communication Configuration

Based on the relationship between FSoE and the EtherCAT protocol, and the fact that an EtherCAT network has only one master, the EVO1000EF must be connected to a standard PLC that supports the EtherCAT protocol. In a safety PLC project, the standard PLC serves as the EtherCAT master, while the safety PLC, Hi50 coupler, GL20, GL10, and other standard I/O couplers serve as EtherCAT slaves. The safety PLC also serves as the FSoE master, with failsafe I/O modules or safety servo modules serving as FSoE slaves.

11.4.4.1 Configuring a Safety PLC as an EtherCAT Slave of a Standard PLC

A standard PLC and up to 8 safety PLCs form a control system. The supported standard PLC series are: EVO500, EVO700, EVO700C, and APC. The EtherCAT ports of the standard PLCs listed above that can act as the

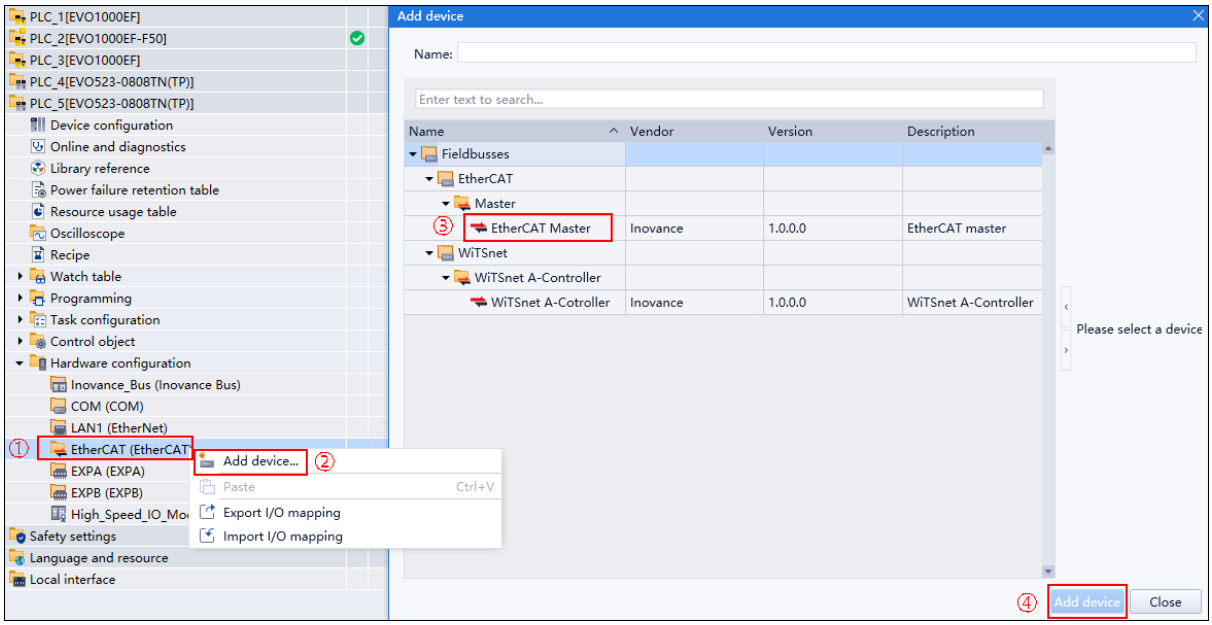
EtherCAT master must support configuring multiple models of the EVO1000EF series safety PLCs as EtherCAT slaves.

- After an independent safety PLC is configured as an EtherCAT slave under a standard PLC, the name of the slave and the name of the safety PLC in the project tree are automatically synchronized when either is modified.
- If the safety PLC node of the EtherCAT slave under the standard PLC is deleted, the slave relationship will be terminated, but the hardware and software of the independent safety PLC in the project tree will remain unchanged.
- If the independent safety PLC in the project tree is deleted, the safety PLC node of the EtherCAT slave under the standard PLC will be automatically deleted.
- Under an independent safety PLC in the project tree, the following devices cannot be added, deleted, or renamed: communication units, local I/O, remote couplers, and remote I/O. Independent safety PLCs that are not configured as EtherCAT slaves of a standard PLC can freely add local and remote slaves (servos, couplers, I/O).
- The EtherCAT slave node of a safety PLC under a standard PLC does not support copy and paste. An independent safety PLC not configured as an EtherCAT slave under a standard PLC supports copy and paste.
- Device changes to the independent safety PLC in the project tree will be automatically reflected in its slave node under the standard PLC.

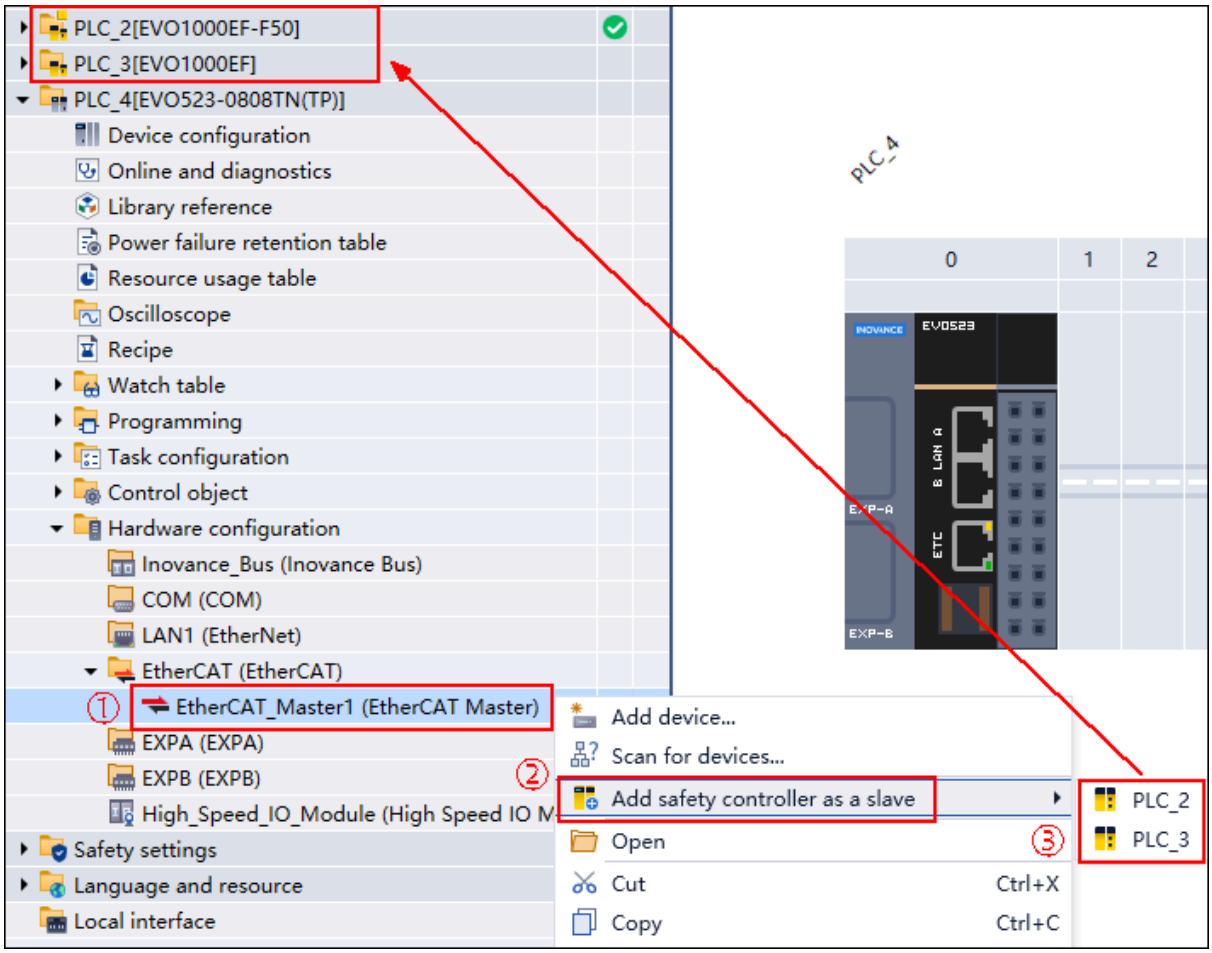
11.4.4.1.1 Configuring an Independent Safety PLC as an EtherCAT Slave

Method 1: Configuring via the project tree

1. Create a project and add a safety PLC device and a standard PLC device.
2. Right-click "EtherCAT" under "Hardware configuration" of the standard PLC in the project tree, and select "Add device".
3. Select "EtherCAT Master" and click "Add device".



4. Right-click the added "EtherCAT Master" and select "Add safety controller as a slave".

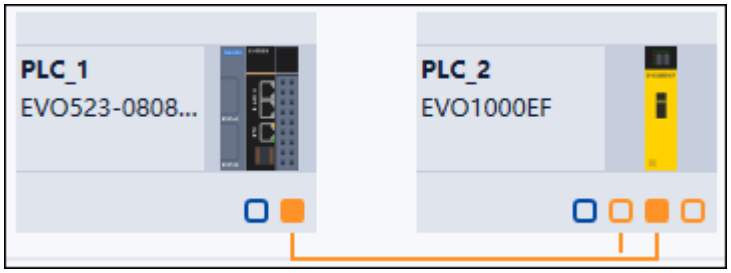


5. Select the independent safety PLC created in the project, with the following specific requirements:
   - The safety PLC has not been associated any EtherCAT master.
   - The safety PLC has been configured with an ECT communication unit.
   - The model of the safety PLC is the same as the EVO1000EF slave model.

> **NOTE**
>
> If the current safety PLC has been configured with local or remote I/O modules, when you add the safety PLC as slave, a prompt message will appear, indicating that the I/O devices of PLC_2 will be deleted. After the user confirms, the association will be established.

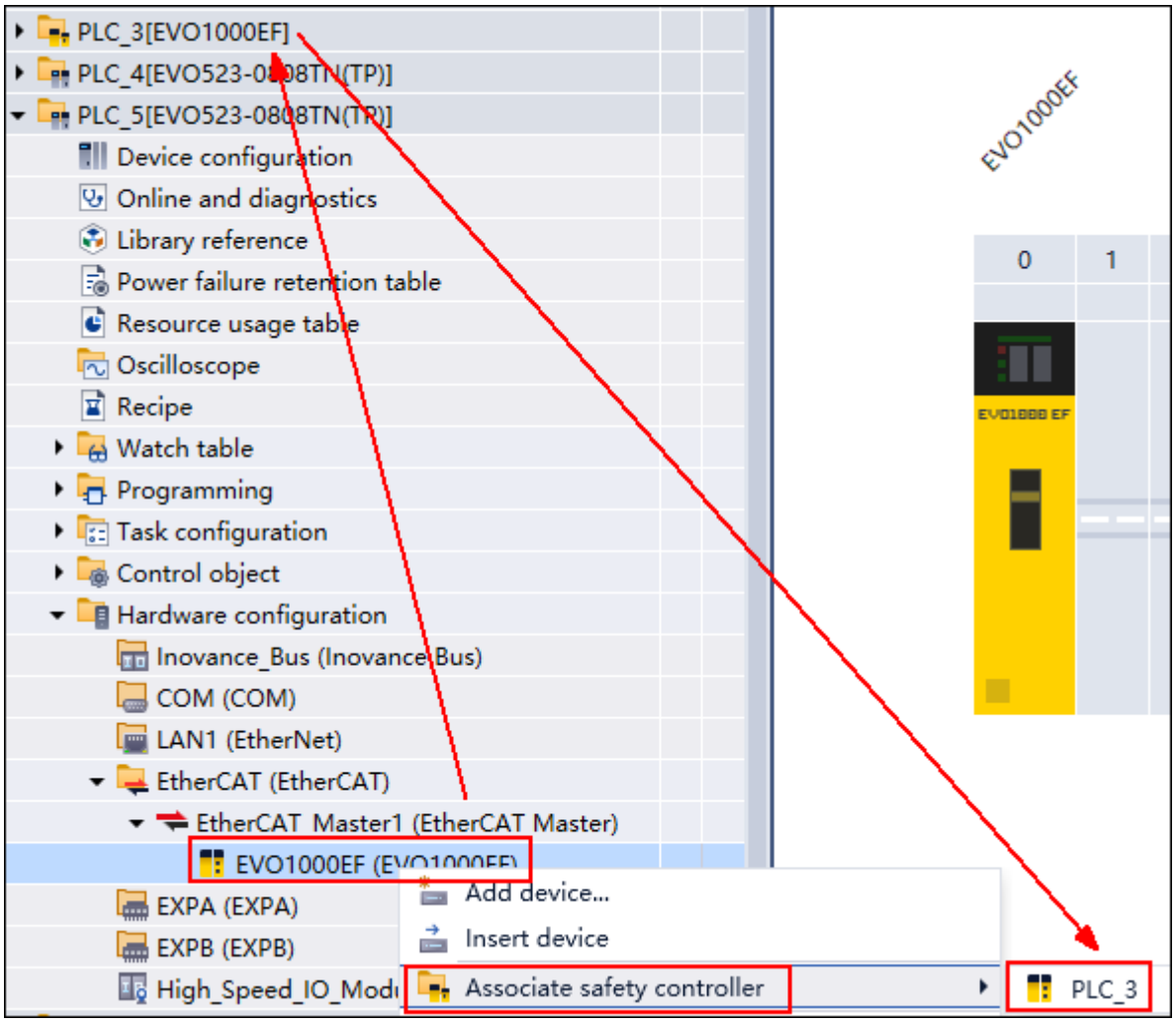
Method 2: Configuring via network view

1. Create a project and add a safety PLC device and a standard PLC device.
2. Double-click "Network configuration" in the project tree to connect the EtherCAT ports of the standard PLC and the safety PLC.



11.4.4.1.2 Directly Creating a Safety PLC as an EtherCAT Slave

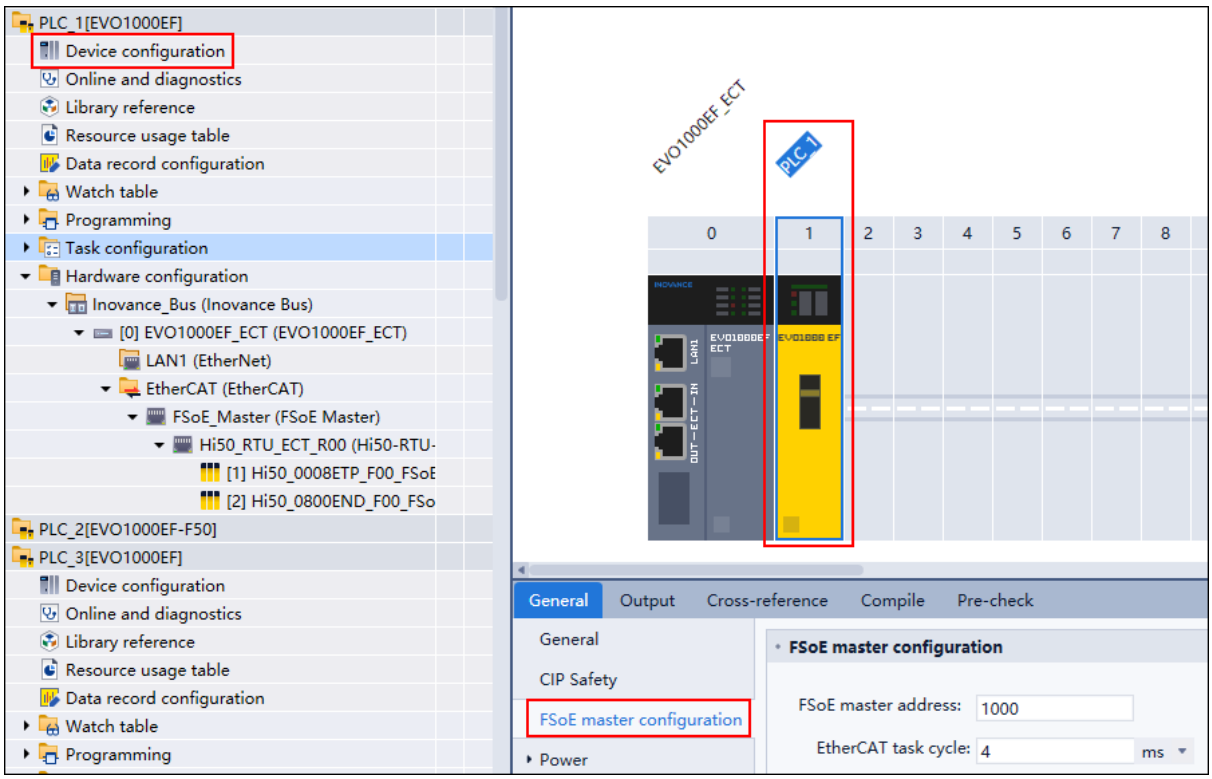
1. Create a project and add a safety PLC device and a standard PLC device.
2. Right-click "EtherCAT" under "Hardware configuration" of the standard PLC in the project tree, and select "Add device".
3. Select "EtherCAT Master" and click "Add device".
4. Right-click "EtherCAT Master" and select "Add device".
5. Select the safety PLC to be added, such as "EVO1000EF", and click "Add device". The specific requirements are as follows:
   - There is an independent safety PLC in the project that is not associated.
   - The independent safety PLC has been configured with an ECT communication unit.
   - The added safety PLC has the same model as the independent safety PLC in the project.
6. Right-click the added safety PLC under "EtherCAT Master", select "Associate safety controller", and select the independent safety PLC created in the project.



11.4.4.1.3 Configuring Communication Parameters of Safety PLC

1. Double-click "Device configuration" under the safety PLC in the project tree, click the safety PLC to open the monitoring area.
2. Configure parameters in "General > FSoE master configuration".

Parameter	Value Range	Default Value	Note
FSoE master address	1 to 65535	1000	◦ As the FSoE master address, it must be unique within the EtherCAT network. ◦ The iFA provides an automatically assigned address, which can be modified manually. ◦ It ensures the uniqueness of the FSoE master/slave address across the entire EtherCAT network to avoid master-master conflicts, slave-slave conflicts, and master-slave conflicts. ◦ During safety PLC compilation, the uniqueness of the master address and the FSoE addresses of its assigned slaves is checked.
EtherCAT task cycle	◦ For EIP communication unit, the FSoE option is grayed out. ◦ For ECT communication unit, both CIP Safety and FSoE need to be configured.	4 millisconds	◦ The task cycle can be defined by users. It must be consistent with the EtherCAT task cycle of the safety PLC's EtherCAT master. ◦ For standard PLCs, the EtherCAT task cycle can be selected in ms or us. ◦ The safety PLC must have the same EtherCAT task cycle value and unit as the standard PLC; otherwise, a fault level message will be displayed during the safety PLC compilation. ◦ Double-click "Device configuration" under the safety PLC in the project tree, click the safety PLC to open the monitoring area.



11.4.4.2 Configuring I/O Modules

I/O modules include local failsafe I/O modules and remote I/O modules, which can only be created under a standard PLC.

> **NOTE**
>
> During EtherCAT communication, local failsafe I/O and remote failsafe I/O (including couplers) can only be created, deleted, and renamed under a standard PLC. Whether it is local or remote I/O for the safety PLC, its slot number must be displayed in the project tree.

11.4.4.2.1 Adding a Local Failsafe I/O Module

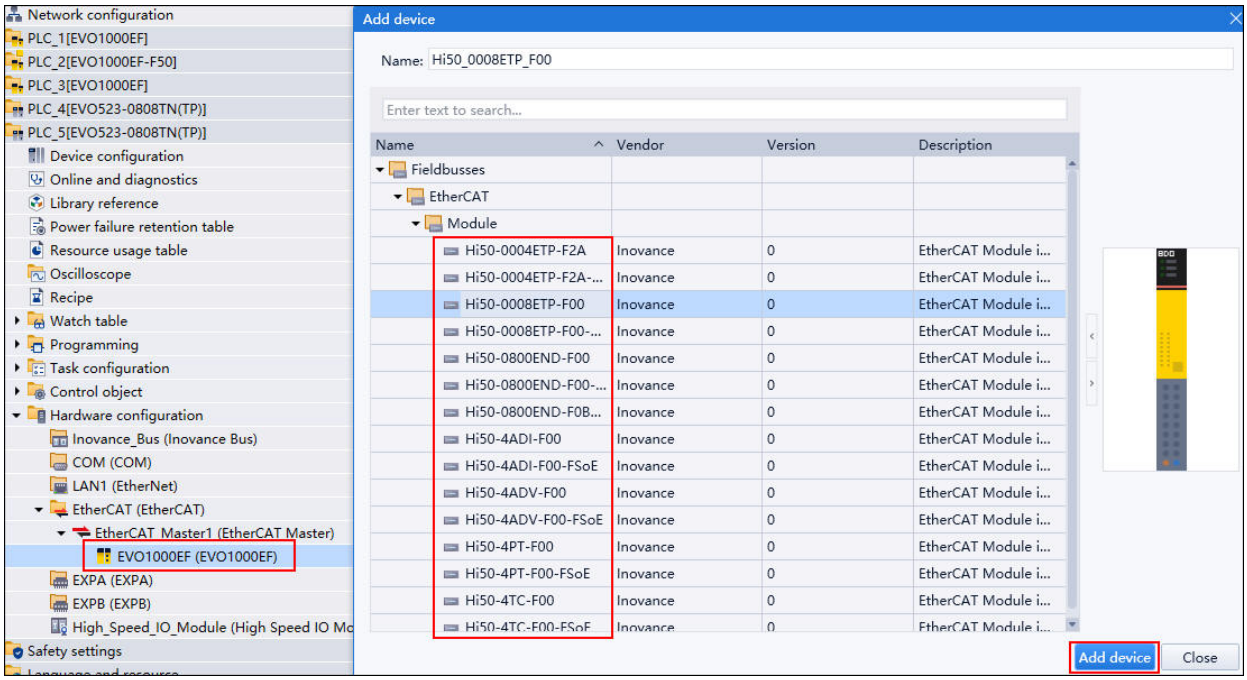
Introduction

After adding a local failsafe I/O module, the local failsafe I/O module under the standard PLC must be synchronized to the independent safety PLC rack in the project tree.

- Local failsafe I/O can only belong to this safety PLC.
- In the project tree, under the standard PLC and safety PLC, the I/O modules need to display the slot number.

Procedure

1. Right-click the safety PLC under "Hardware configuration > EtherCAT > EtherCAT Master" of the standard PLC and select "Add device".
2. Select the failsafe I/O module to be added and click "Add device".



11.4.4.2.2 Adding a Remote Failsafe I/O Module

Introduction

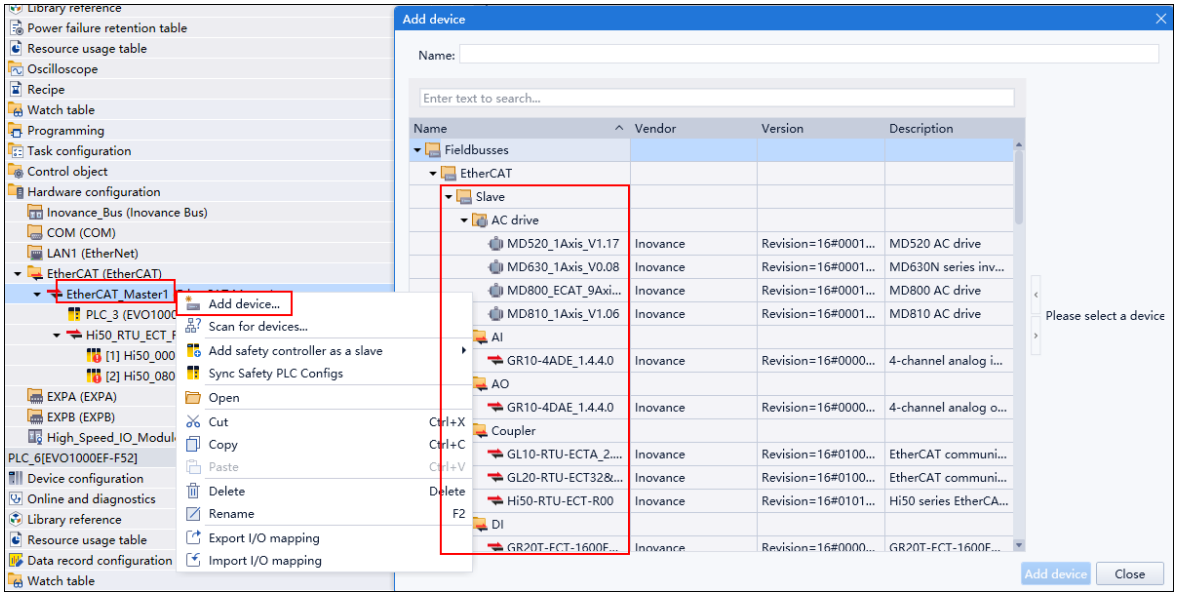
Under the EtherCAT Master of a standard PLC, you can add standard I/O couplers such as GL10, GL20, and Hi50. You can also add standard I/O modules to the couplers. Standard couplers and standard I/O modules are automatically and exclusively subordinate to the standard PLC. In the EtherCAT network, the standard PLC is presented as the EtherCAT master, under which the safety PLC, failsafe I/O couplers, standard I/Os, and other EtherCAT slaves are connected.

When there are multiple safety PLCs under a standard PLC, the newly created remote failsafe I/O module defaults to be the first safety PLC. If there is currently no safety PLC, the failsafe I/O module is in an unassigned state.

Procedure

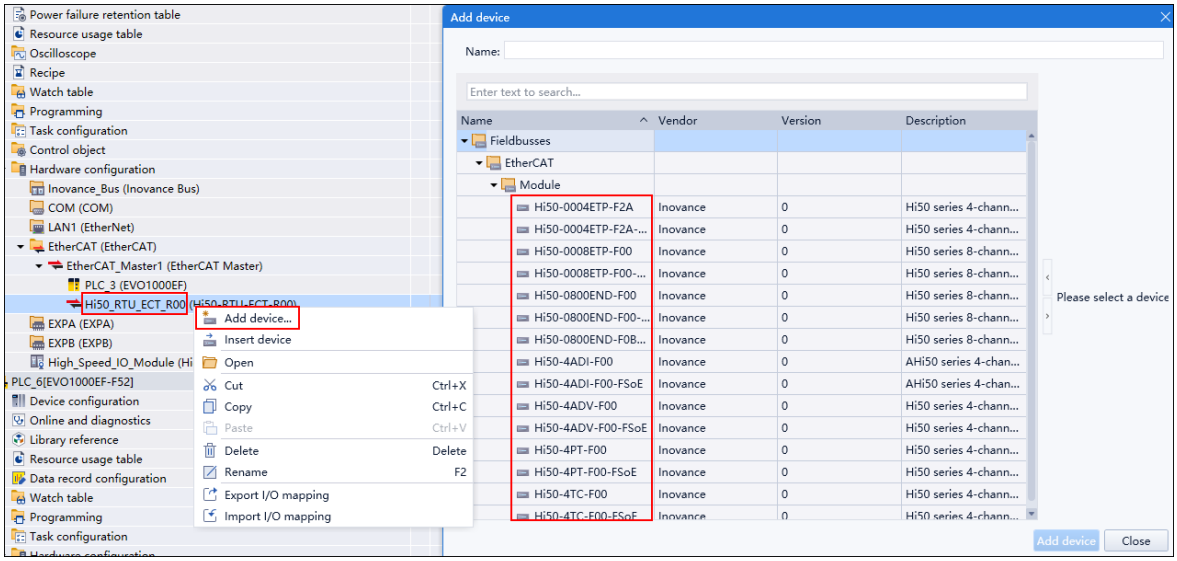
1. Add "Hi50-RTU-ECT-R00".
   a. Right-click "EtherCAT Master" under "Hardware configuration > EtherCAT" of the standard PLC and select "Add device".

   b. Select "Hi50-RTU-ECT-R00" and click "Add device".



2. Add a remote failsafe I/O module.
   a. Right-click "Hi50-RTU-ECT-R00" and select "Add device".

   b. Select the failsafe I/O module to be added and click "Add device".



11.4.4.2.3 Assigning FSoE Master to Failsafe I/O Modules

Introduction

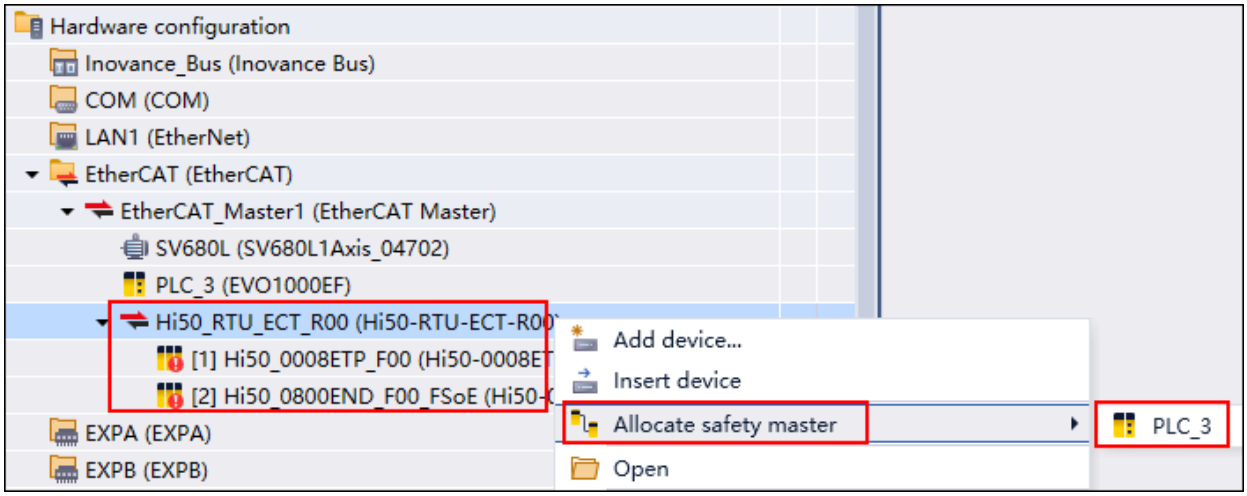
In EtherCAT and FSoE networks, there are multiple safety PLCs and multiple remote failsafe I/Os, so it is necessary to assign an FSoE master to the failsafe I/O modules on the coupler. Newly added I/O modules belong to the first safety PLC by default, which is the safety PLC closest to the standard PLC in the EtherCAT network.

- Hi50-RTU-ECT-R00 is a non-safety module, and its master is fixed as the standard PLC, requiring no master assignment.
- Failsafe I/O modules on the local rack of the safety PLC automatically and are permanently associated with the safety PLC on the same rack, requiring no master assignment.
- For servo modules with safety signals, it is necessary to support assigning a safety FSoE master to them.

Procedure

Right-click the added failsafe I/O module or coupler, select "Allocate safety master" to assign an FSoE master to the remote failsafe I/O.

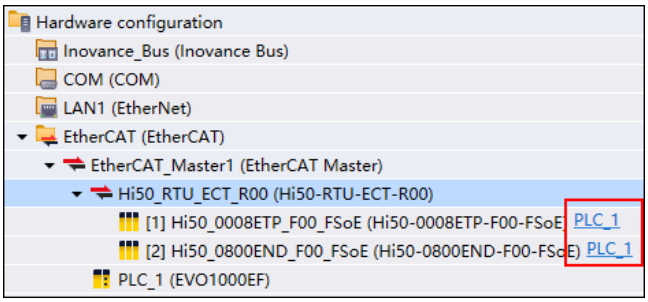
- If an associated safety PLC (FSoE master) is renamed or its device model is changed, the master-slave relationship displayed for the I/O modules in the project tree shall be updated accordingly.
- When a safety PLC is deleted, the failsafe I/Os previously assigned to that PLC are automatically assigned to the default safety PLC.
- If there are no safety PLCs, the I/O modules are in an unassigned master state.
- When a failsafe I/O module is deleted, it shall also be synchronously updated (deleted) under the safety PLC to which it belongs.



Result

Remote failsafe I/O modules display the name of the safety PLC (FSoE master) to which they belong.

- If at least one failsafe I/O module on a given coupler belongs to that safety PLC, then the coupler and that failsafe I/O module shall be displayed under the safety PLC.
- If none of the failsafe I/O modules on a given coupler belongs to that safety PLC, then neither the coupler nor its failsafe I/O modules shall be displayed under that safety PLC.



11.4.4.2.4 Reading FSoE DIP Switch Addresses of Failsafe I/O Modules

Introduction

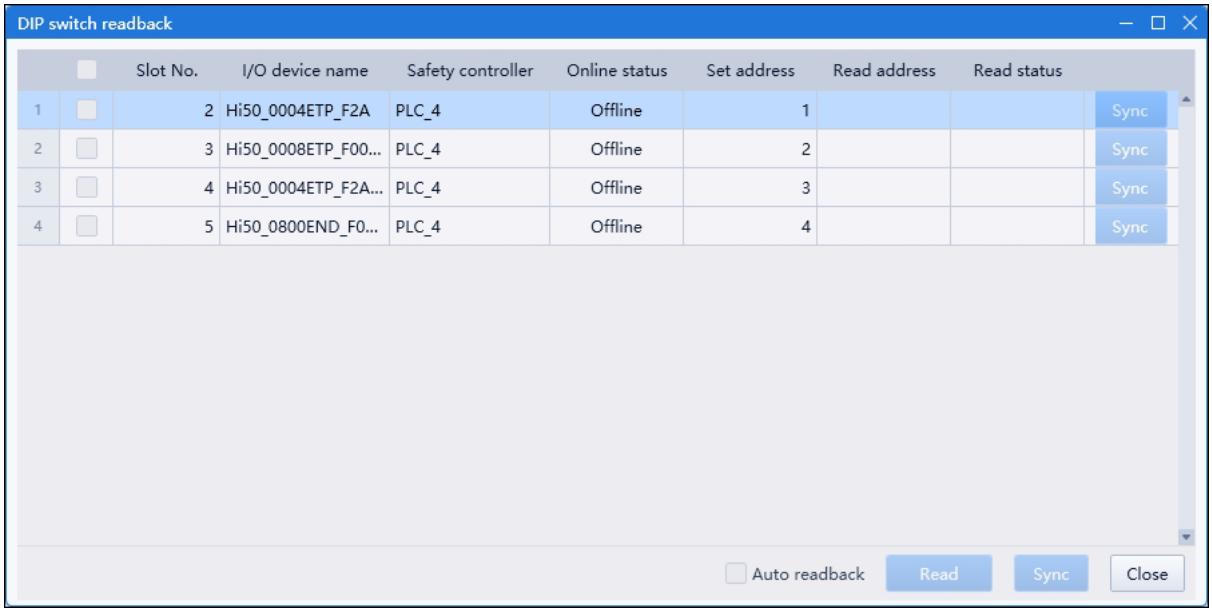
The DIP switch readback function is designed for configuration verification and synchronization of slave failsafe I/O modules. It is primarily used to read the DIP switch address from the FSoE configuration of the failsafe I/O module and update the address to the "FSoE configuration > FSoE slave address" of the corresponding I/O module, either individually or in batch.

Prerequisites

The standard PLC must be in an online running state.

Procedure

1. With the standard PLC online, right-click the target I/O module under "Hardware configuration > EtherCAT > EtherCAT Master" of the standard PLC and select "DIP switch readback".



2. The DIP switch readback information list is dynamically adapted for display. It automatically reads the current I/O modules and generates corresponding information entries. If no I/O modules are present, the DIP switch readback function will not be displayed.
   - Slot number: The installation slot number corresponding to the I/O module.
   - IO device name: The currently mounted I/O device.
   - Safety controller: The associated safety PLC device.
   - Online status: The running status of the corresponding safety PLC, including online and offline.
   - Set address: The "FSoE configuration > FSoE slave address" of the slave I/O module.
   - Readback address: The actual DIP switch address of the I/O module read via the DIP switch readback function.
   - Readback status: The result of the DIP switch address readback for the current I/O module (Success/Failure).
3. Execute DIP switch readback and synchronization as needed.
   - Automatic readback: The system will automatically and cyclically execute DIP switch address readback for all I/O modules in the list order, continuously refreshing the readback addresses and statuses of the I/O modules until the automatic readback task is manually canceled.
   - DIP switch readback: The system will perform a single round of DIP switch address readback for all I/O modules in the list. After reading the addresses for all modules and refreshing the list data, it automatically stops without looping.
   - Synchronization: Synchronizes the successfully read-back DIP switch addresses to "FSoE configuration > FSoE slave address" of the corresponding I/O modules. Two modes are available:
     - Single synchronization: Each I/O module entry in the list has an independent "Synchronize" button for address synchronization of that single module.
     - Batch synchronization: A global "Synchronize" button is provided at the bottom right, allowing batch synchronization of the selected modules in the list.
4. Diagnosis of abnormal readback results.
   - When the device readback status is failed, or when the readback address completely matches the set address, the synchronization button for the I/O module is grayed out and disabled, the corresponding checkbox in the list is disabled, and synchronization cannot be performed.

- Synchronization permissions are restricted based on the online status of the safety controller:
  - Online status is "Online": The synchronization function for all I/O modules is disabled; checkboxes in the list cannot be operated; address synchronization is not allowed.
  - Online status is "Offline": When the readback address of the I/O module does not match the set address, address synchronization is allowed.
- During the system's DIP switch readback process, the list checkboxes and synchronization-related functions are globally locked, and all operations are temporarily unavailable. The lock is automatically released after the readback task is completed.

11.4.4.2.5 Configuring Failsafe I/O Module Parameters

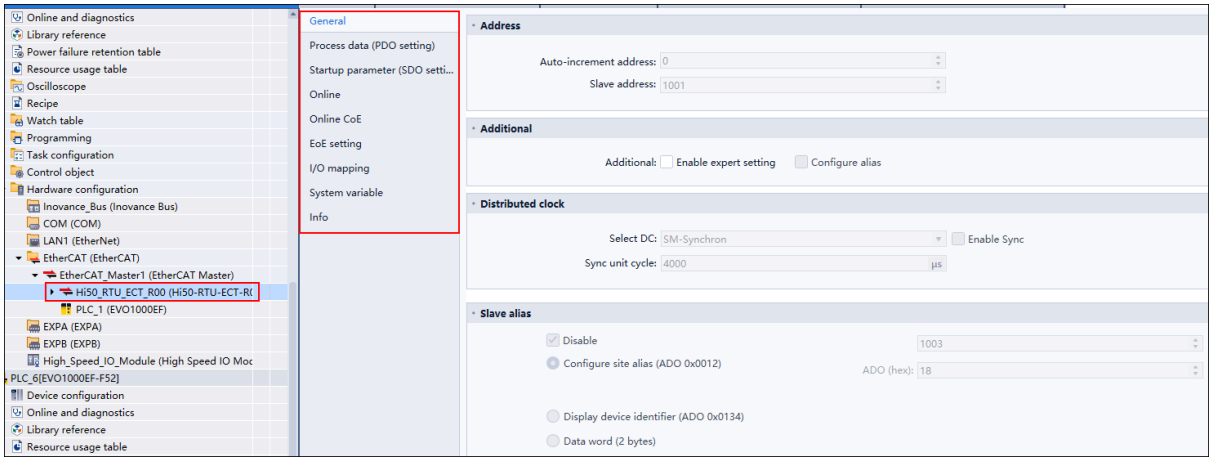
Local failsafe I/O modules on the safety PLC rack and remote failsafe I/O modules on the coupler both have EtherCAT and FSoE parameters. For failsafe I/O modules supporting CIP Safety and FSoE, if an ECT communication unit or coupler is configured, the configuration page displays FSoE information; if an EIP communication unit or coupler is configured, the configuration page displays CIP Safety information. Automatic adaptation is supported. Failsafe I/O modules that support FSoE only can only be used with an ECT communication unit or coupler.

When the iFA project file downloaded to the safety PLC remains unchanged, I/O modules that support both CIP Safety and FSoE in the EtherCAT/FSoE network can be directly interchanged with pure FSoE protocol failsafe I/O modules. For example, if the configured module in the project is Hi50-0800END-F00, it can be replaced with Hi50-0800END-F00-FSoE. All other failsafe I/O modules of the same type follow this rule.

> NOTE
>
> Module replacement must follow the firmware forward compatibility principle. If the project configuration firmware version is Hi50-0800END-F00 V2.0, it can be replaced with Hi50-0800END-F00-FSoE of V1.1, but not of V1.0.

Configuring EtherCAT parameters for failsafe I/O modules

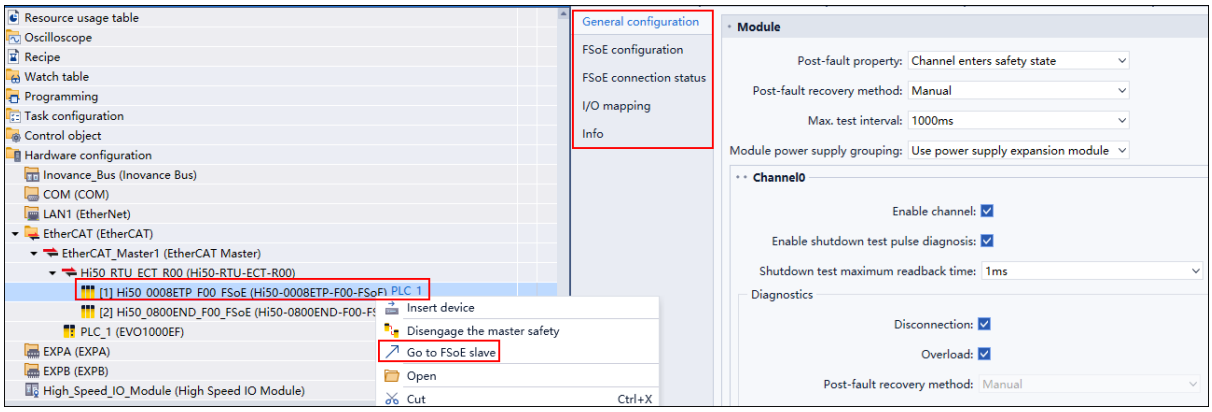
1. Double-click the target coupler under the standard PLC to open the "General configuration" page by default.



2. For EtherCAT parameter configuration of failsafe I/O modules, see the 10.6.3.3 Configuring Slave Parameters.

Configuring FSoE parameters for I/O modules

1. Double-click the target failsafe I/O module under a safety PLC, or right-click the target I/O module under a standard PLC and select "Go to FSoE slave" to open the "General configuration" page.



2. For FSoE parameter configuration of failsafe I/O modules, see 11.4.2.7 Safety I/O Module Configuration.
3. After completing parameter configuration, right-click "EtherCAT Master" under the standard PLC in the project tree and select "Sync safety PLC variable mappings".

11.4.4.3 Configuring Safety Servo Modules

Introduction

Inovance servo devices with FSoE function: SV670NS and SV680-INT.

- Servo modules with FSoE function (SV670NS and SV680-INT) can be added to a safety PLC EtherCAT communication unit.
- The safety servo parameters are downloaded to the servo device by the standard PLC via EoE.
- Safety parameters of the servo with safety functions can be configured in the detailed view. For details, see 17.3.15 Safety Parameter.

Prerequisites

The iFA Evolution software supports automatic scanning for servo drives, which requires the following conditions to be met:

- The safety PLC project has be downloaded.
- The safety PLC device is in a stopped state.
- When the current PLC needs to connect to devices such as PLCs, HMIs, PACs, PCs, and drives, this function can scan all compatible devices within the same subnet, allowing users to select the target device. For IP address modification, see the "PLC Programming > Online Controller Operations > Network Settings" in the iFA Information System Guide.

Procedure

1. Manually add a safety servo module.
   a. Right-click "EtherCAT Master" under "Hardware configuration > EtherCAT" of the standard PLC and select "Add device".
   b. Select the servo drive to be added and click "Add device".

2. Scan to add a safety servo module.
   a. Right-click "EtherCAT Master" under "Hardware configuration > EtherCAT" of the standard PLC and select "Scan for devices".
   b. In the "Device scanning" window, click "Scan for devices". The scanned devices will be displayed in the "Physical connection" list.
   c. Select the servo drive to be added, and click "Apply physical network configuration".
3. Right-click the added safety servo module, select "Allocate safety master", and assign an FSoE master to the safety servo.
4. Double-click the servo drive under "Standard PLC > Hardware configuration > EtherCAT > EtherCAT Master". For parameter configuration, see 10.6.3.3 Configuring Slave Parameters.
5. For EoE configuration of the safety servo module, see 17.1 EtherCAT Configuration.
6. After completing parameter configuration, right-click "EtherCAT Master" under the standard PLC in the project tree and select "Sync safety PLC variable mappings".

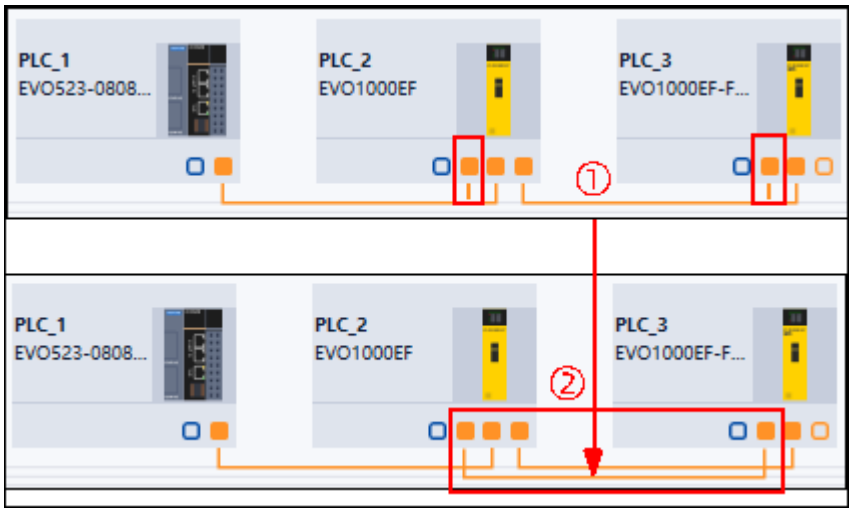
11.4.4.4 FSoE Communication Between Safety PLCs

EVO1000EF series safety PLCs support communication via FSoE, configurable through network configuration. Two communication configuration methods are supported:

- Multiple safety PLCs deployed in the same project.
- Multiple safety PLCs deployed in different projects.

FSoE communication between safety PLCs in the same project

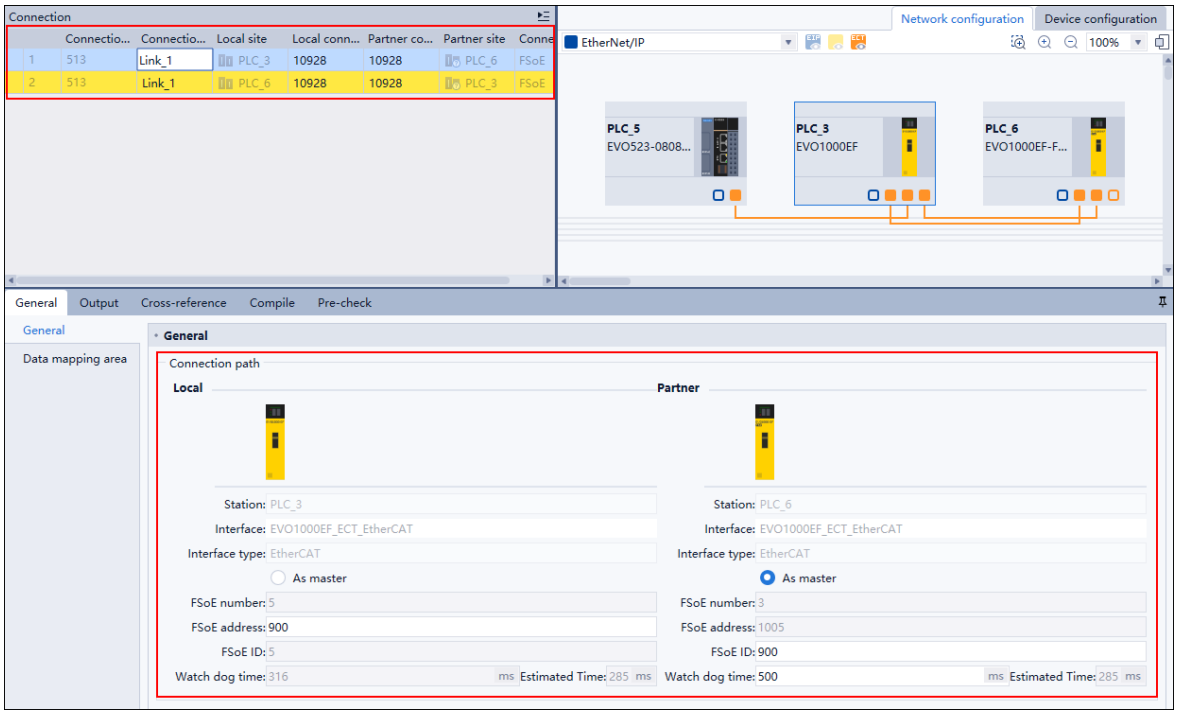
1. Create a project and add two safety PLCs and a standard PLC.
2. Double-click "Network configuration" in the project tree to open the network configuration page.
3. In the same EtherCAT network, press and hold the left mouse button to connect the EtherCAT ports of the two safety PLCs to create an FSoE connection.



4. Configure parameters for the local PLC and partner PLC.
   a. Select the generated connection. On the "General" tab in the monitoring area, configure the connection parameters for the local PLC and partner PLC.

Parameter	Value Range	Default Value	Note
As master	Yes or No	Yes	-

Parameter	Value Range	Default Value	Note
FSoE connection ID	1 to 65535	-	Cannot be changed. The connection IDs of the two safety PLCs can be different.
FSoE address	1 to 65535	1	▪ If acting as a master, the "FSoE master address" (unmodifiable) can be viewed under "General > FSoE master configuration" of the safety PLC. ▪ If acting as a slave, the "FSoE master address" must be entered manually.
FSoE connection ID	1 to 65535	-	Must be unique within the FSoE network.
Watchdog time	1 to 65535	-	Must not be less than the estimated watchdog time.



b. Switch to the "Data mapping area" tab to configure the related parameters for the connection.

> **NOTE**
>
> The length of data mapping area in each direction cannot exceed 240 bytes; otherwise, a fault level message will be displayed during compilation.

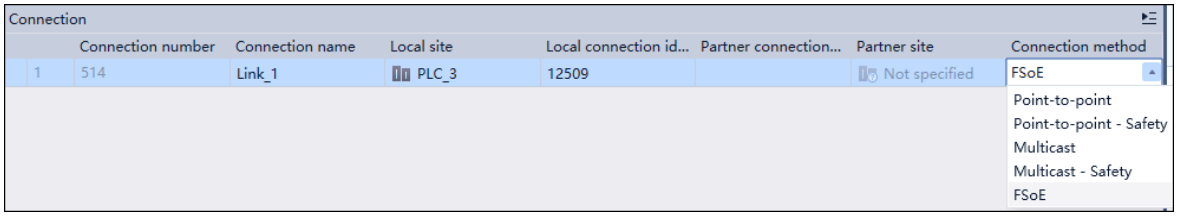
c. Configure the "Local variable", "Transfer direction", and "Partner variable" to send and receive communication data.

5. After completing parameter configuration, right-click "EtherCAT Master" under the standard PLC in the project tree and select "Sync safety C2C configuration".

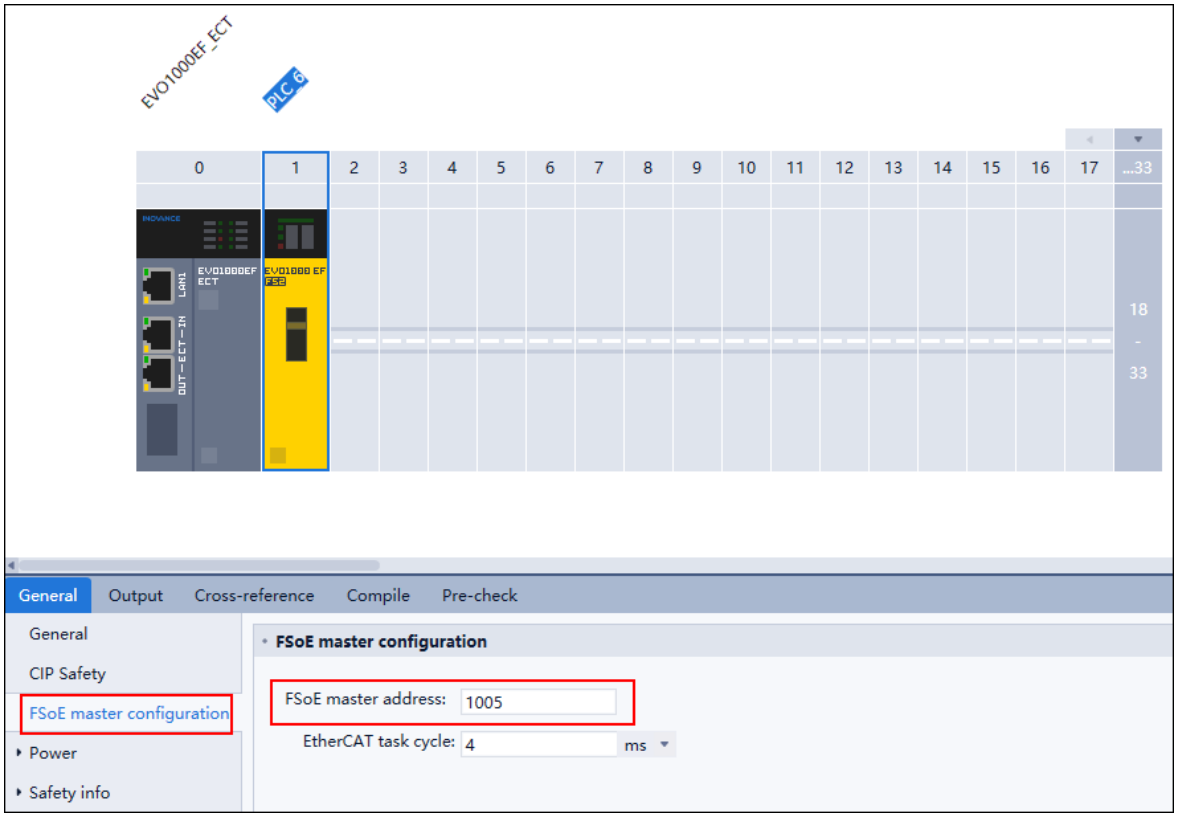
FSoE communication between safety PLCs in different projects

The configuration for the two safety PLCs are the same as above.

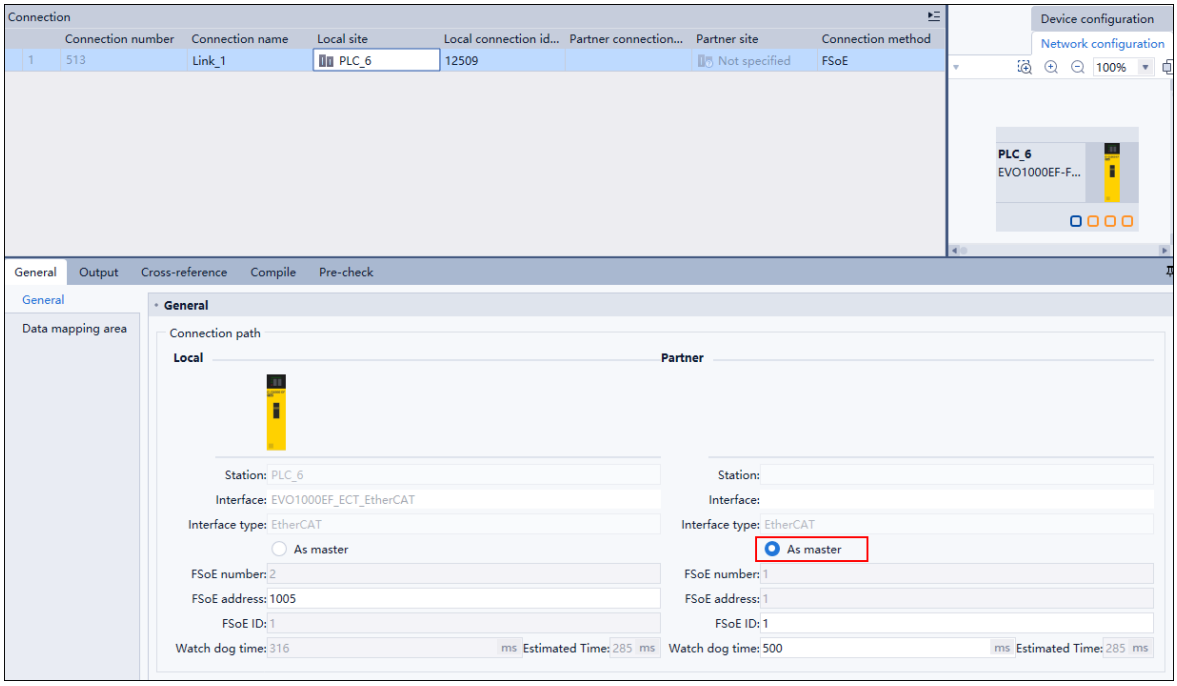
1. Create a project and add a safety PLC device and a standard PLC device.
2. Double-click "Network configuration" in the project tree to open the network configuration page.
3. Right-click the blank area in the "Connection" interface and select "Add" to add a connection.
4. Configure the connection parameters.
   - Select the configured safety PLC for "Local site".
   - Select "FSoE" for "Connection method".
   - Keep the "Local connection identifier" of the two PLCs in different projects consistent.



5. View the FSoE master address.
   a. Double-click "Device configuration" of the safety PLC in the project tree.
   b. Click the safety PLC in the "Device configuration" interface to open the "General" tab in the monitoring area.
   c. Select "FSoE master configuration" to view the "FSoE master address".



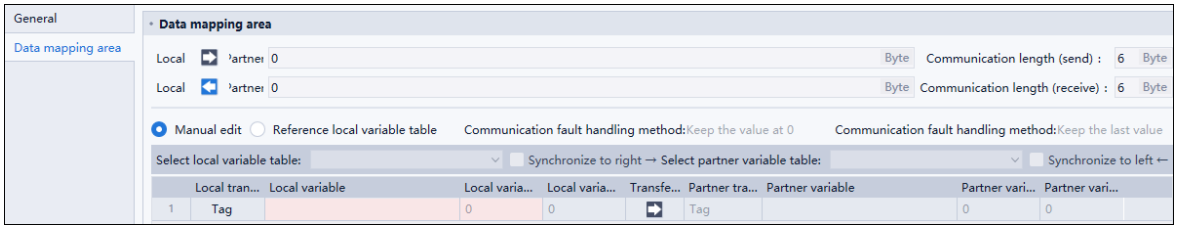
6. Select the generated connection. On the "General" tab in the monitoring area, configure the parameters for the local PLC and partner PLC.
   - Check "As master" for the partner PLC.
   - When the partner PLC is an FSoE master, ensure that the "FSoE address" of the local PLC is different from all master and slave addresses in the FSoE network.



7. Switch to the "Data mapping area" and configure the related parameters for the local PLC and partner PLC.

   a. On the "Data mapping area" tab, right-click the blank area and select "Add" to add a local transmission area.

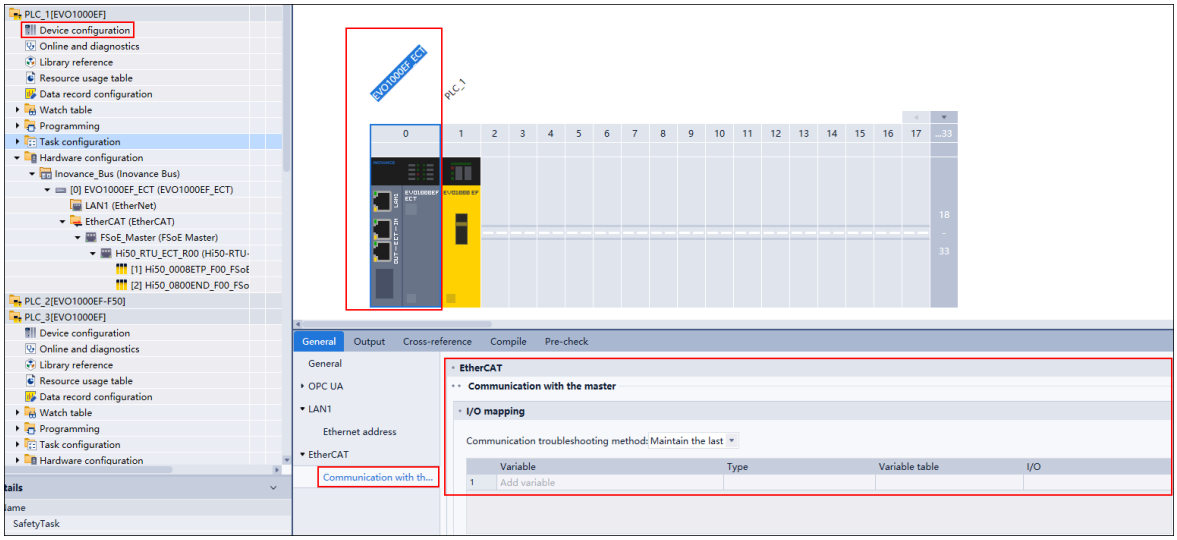
   b. Configure the "Local variable" and "Transfer direction" to send and receive communication data.



8. After completing parameter configuration, right-click "EtherCAT Master" under the standard PLC in the project tree and select "Sync safety C2C configuration".

11.4.4.5 EtherCAT Communication Between Safety PLC and Standard PLC

1. Double-click "Device configuration" under the safety PLC in the project tree, click the communication unit to open the monitoring area.
2. Configure the "I/O mapping" parameters under "General > EtherCAT > Communication with the master".
   - Variables: Users can manually add mappings and bind them to safety variables or standard variables defined in the safety PLC.
     - Up to 512 variables can be configured. If exceeded, a compilation fault will be output.
     - When the total length of PDOs exceeds the limit, it needs to be checked during compilation in the standard PLC.
   - Data type and variable table: Automatically displayed based on the variable name, and cannot be modified.
   - Transmission direction: The standard PLC master can read both safety and standard data types, and can write only the standard data types.



3. After completing parameter configuration, right-click "EtherCAT Master" under the standard PLC in the project tree and select "Sync safety PLC variable mappings".

11.4.5 PLC Property Settings

11.4.5.1 PLC Configuration

Each device in the CIP Safety network has a unique node ID, known as the UNID (Unique Node Identifier), which uniquely identifies the device within the CIP Safety network during communication. The Safety Network Number (SNN) is a core component of the UNID. A complete UNID consists of the following two parts:

- SNN (6 bytes): Identifies the safety network domain to which the device belongs.
- UNID (4 bytes): Contains device type characteristics, such as slot address or port information.

The PLC configuration interface is used to set the UNID for local safety I/O modules, which are backplate modules, so the UNID is the module's slot address. Before the safety PLC starts running, the UNID assigned by the iFA Evolution software must be set to each safety I/O module.

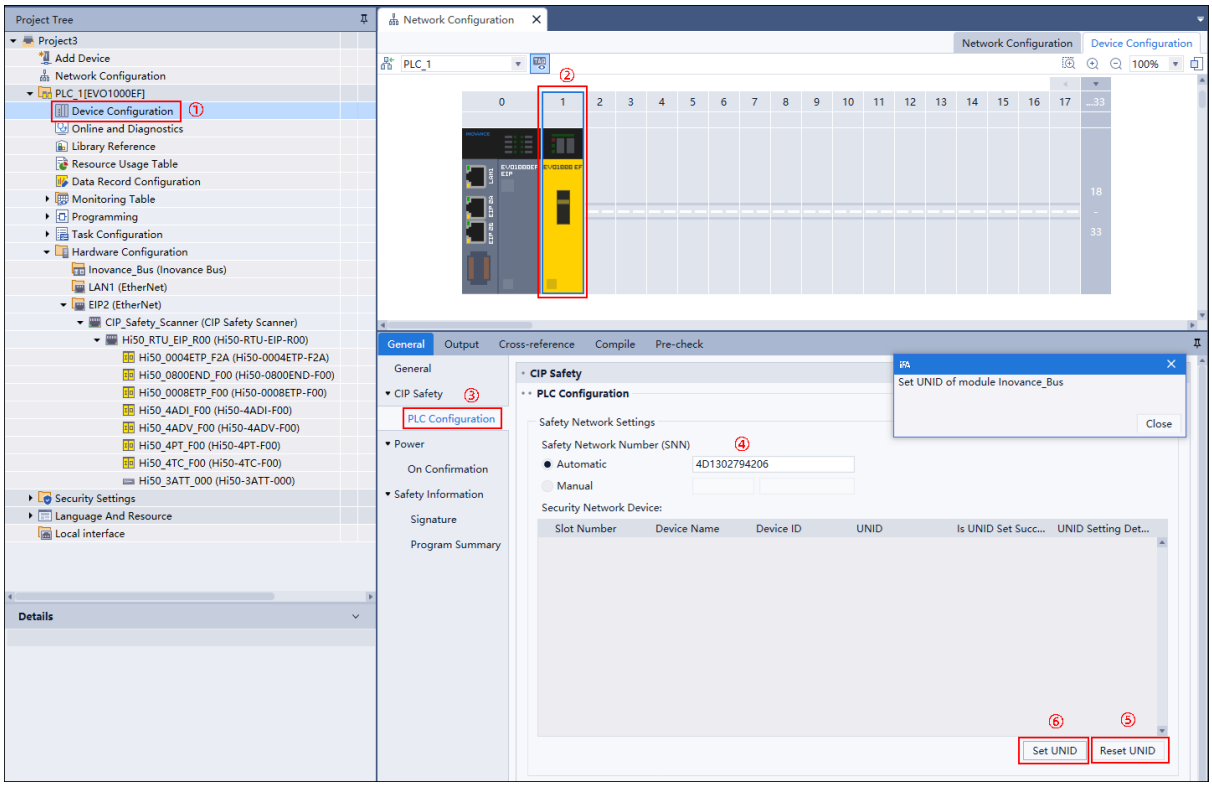
Prerequisites

The safety PLC device is in offline status.

Procedure

1. In the project tree, double-click the "Device configuration" node under the safety PLC device.
2. In the device configuration interface, click the safety PLC device.
3. In the "General > CIP safety > PLC configuration" tab at the bottom of the interface, configure the SNN automatically or manually.
   - Automatic assignment: Generates the SNN automatically based on the current system time.
   - Manual assignment: The first input field is fixed at 1 and cannot be modified. The second input field accepts values from 1 to 9999.
4. The safety network device list below displays slot numbers, device names, device IDs, UNIDs, and other information for each safety I/O module.
   When adding safety I/O modules, UNIDs are automatically assigned. After modifying the SNN, the UNID must be reset before configuration so that the new UNID can be successfully assigned to each module.

5. Click the "Reset UNID" button and confirm that the UNIDs of all safety I/O modules have been successfully reset.
6. Click the "Set UNID" button and confirm that the UNIDs of all safety I/O modules have been successfully set.



CAUTION

- The UNID structure used by the Hi50 series safety I/O modules is: SNN (6 bytes) + physical slot address (4 bytes).
- Users are responsible for ensuring that all interconnected CIP Safety devices have globally unique UNIDs.

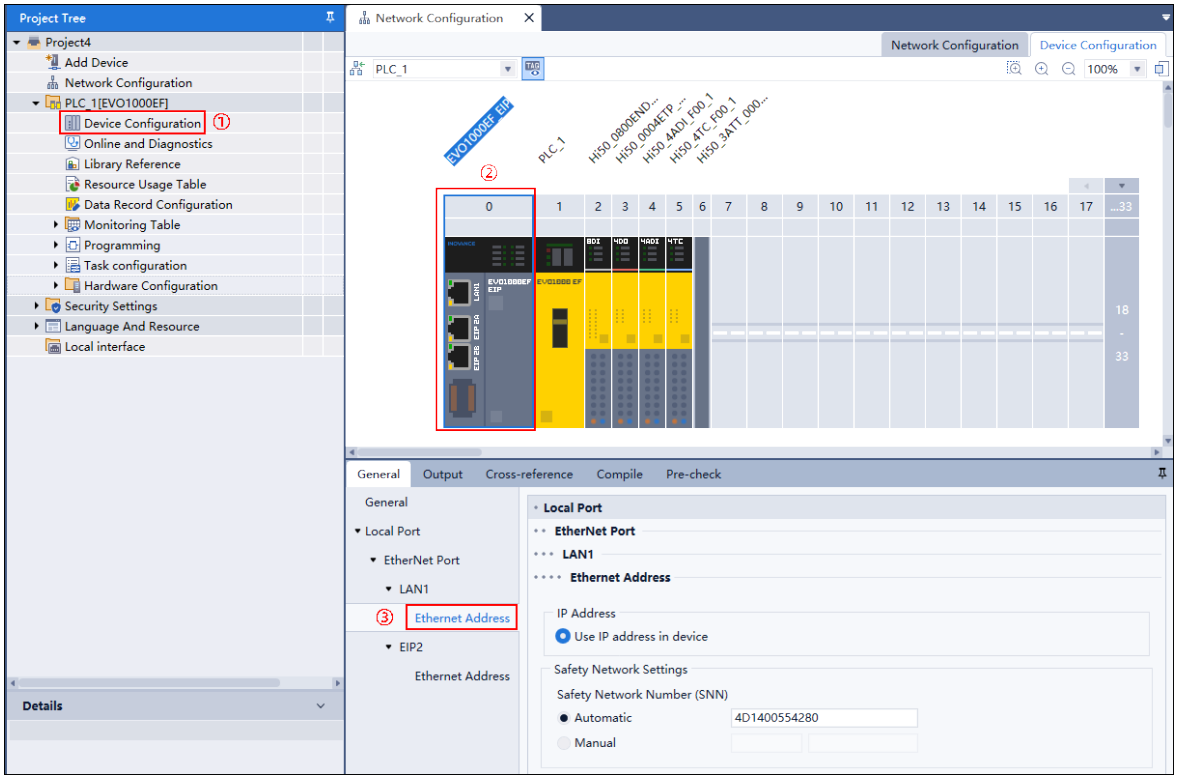
11.4.5.2 Ethernet Address and Safety Network Settings

The safety PLC device communicates via the communication unit with the safety I/O modules, safety gate system, and other safety PLCs on the Hi50 coupler using CIP Safety. Therefore, each network port of the communication unit must be configured with an SNN to assign a UNID to third-party CIP Safety devices connected to the LAN1 or EIP2 port. The UNID structure used is: SNN (6 bytes) + IP address of the device connected to the port (4 bytes).

Procedure

1. In the project tree, double-click the "Device configuration" node under the safety PLC device.
2. In the device configuration interface, click the safety PLC device.
3. Select the "General > Local ports > LAN1 > Ethernet address" tab at the bottom of the interface.
4. The IP address defaults to the device's IP address and cannot be modified.

5. The SNN can be assigned automatically and manually.
   - Automatic assignment: Generates the SNN automatically based on the current system time.
   - Manual assignment: When the first input box is set to 4, the allowable range for the second input box is 1 to 9999; when the first input box is set to 11688 to 65534, the allowable range for the second input box is 0 to 86399999.



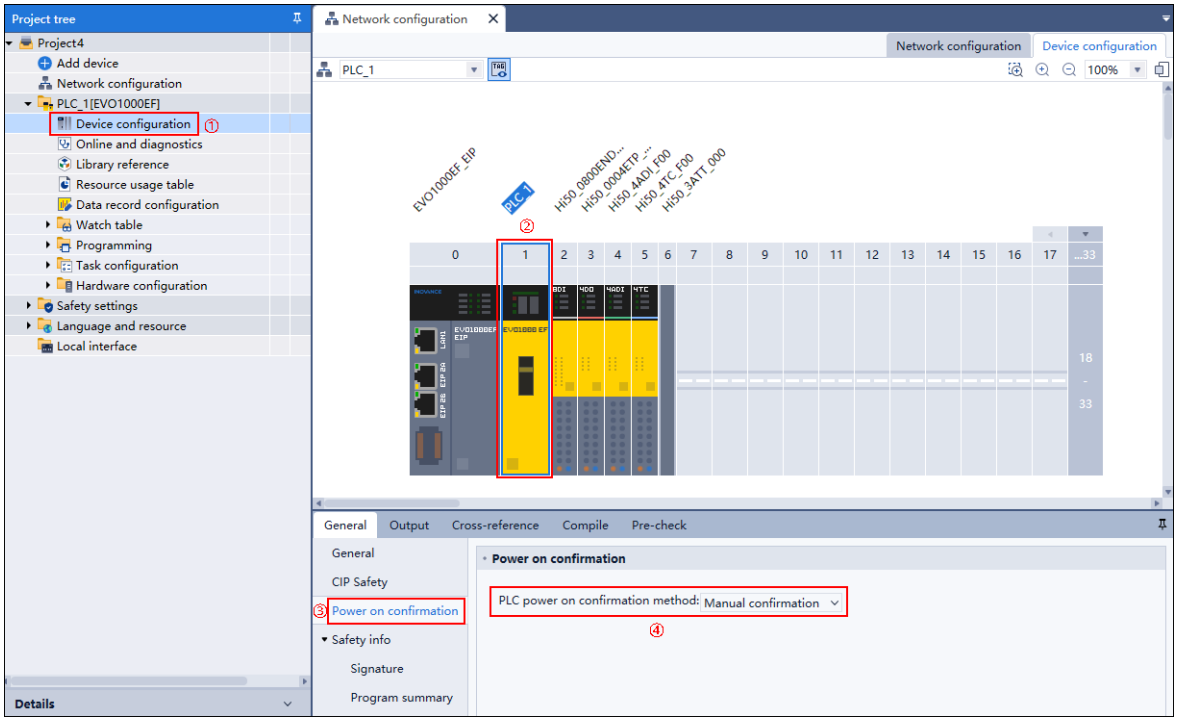
6. The configuration method for EIP2 is the same as for LAN1.

11.4.5.3 Power-On Confirmation

The power-on confirmation mechanism of the safety PLC provides two methods: manual confirmation and automatic confirmation. After the safety confirmation of the safety PLC program, it meets the operating conditions each time it is powered on and passes self test to enter the safety status. At this point, users can choose either manual confirmation or automatic confirmation. If automatic confirmation is selected, the PLC will directly enter the operational mode; if manual confirmation is selected, the PLC enters the operational mode before manual confirmation by users. The system defaults to manual confirmation.

Procedure

1. In the project tree, double-click the "Device configuration" node under the safety PLC device.
2. In the device configuration interface, click the safety PLC device.
3. In the "General > Power-on confirmation" tab, configure the confirmation method.
   - Manual Confirmation: After the safety PLC is powered on, it enters the manual confirmation mode. At this time, the safety PLC is in a safe status, with all inputs and outputs at the safe value "0". The system will only enter the operational mode after users manually confirm that the safety conditions are met.
   - Automatic Confirmation: After the safety PLC is powered on, it automatically checks whether the safety conditions are met. If so, it automatically enters the operational mode without requiring manual confirmation.



11.4.5.4 Safety Information

The safety program summary contains safety-related data (including project name, project signature, safety program signature, safety hardware signature) to support users during system acceptance. It supports generating the safety program summary in electronic format (such as in PDF format).

A safety signature is a unique identifier for project, hardware, and programs, divided into offline signatures and online signatures. An offline signature is generated after successful compilation, while an online signature is generated after successful online connection.

Before generating the safety program summary, confirm the safety project's signature generation time and verify that the comparison results for the project signature, safety program signature, and safety hardware signature are consistent.

Prerequisites

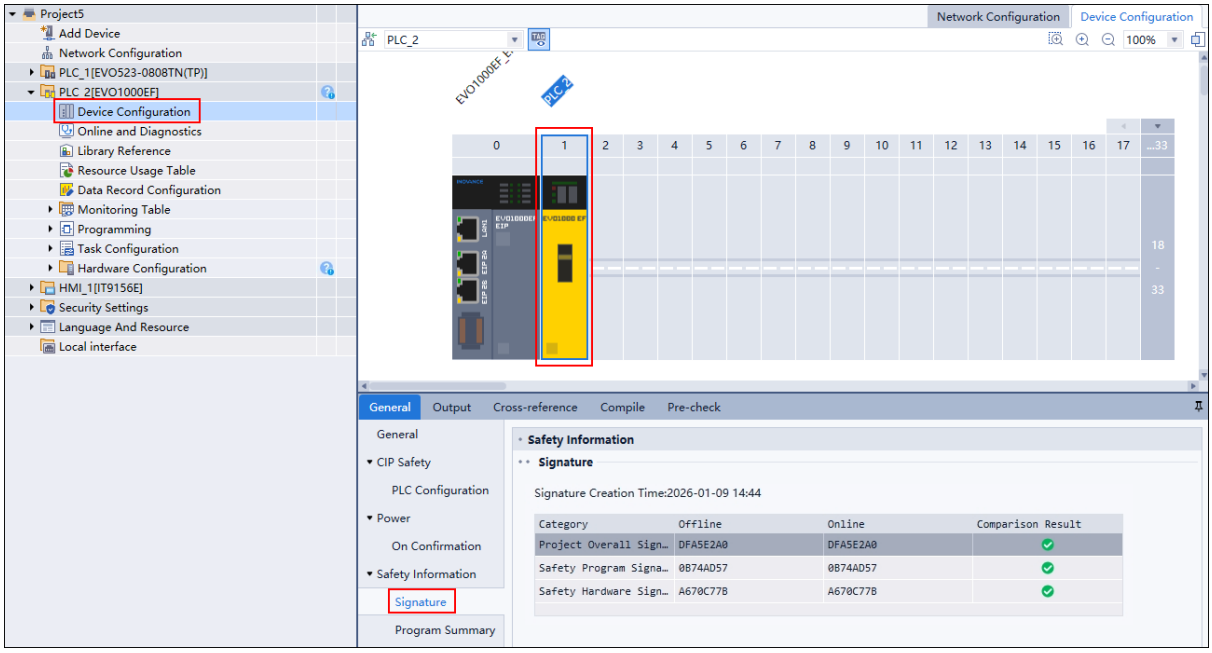
The safety PLC device is online.

Procedure

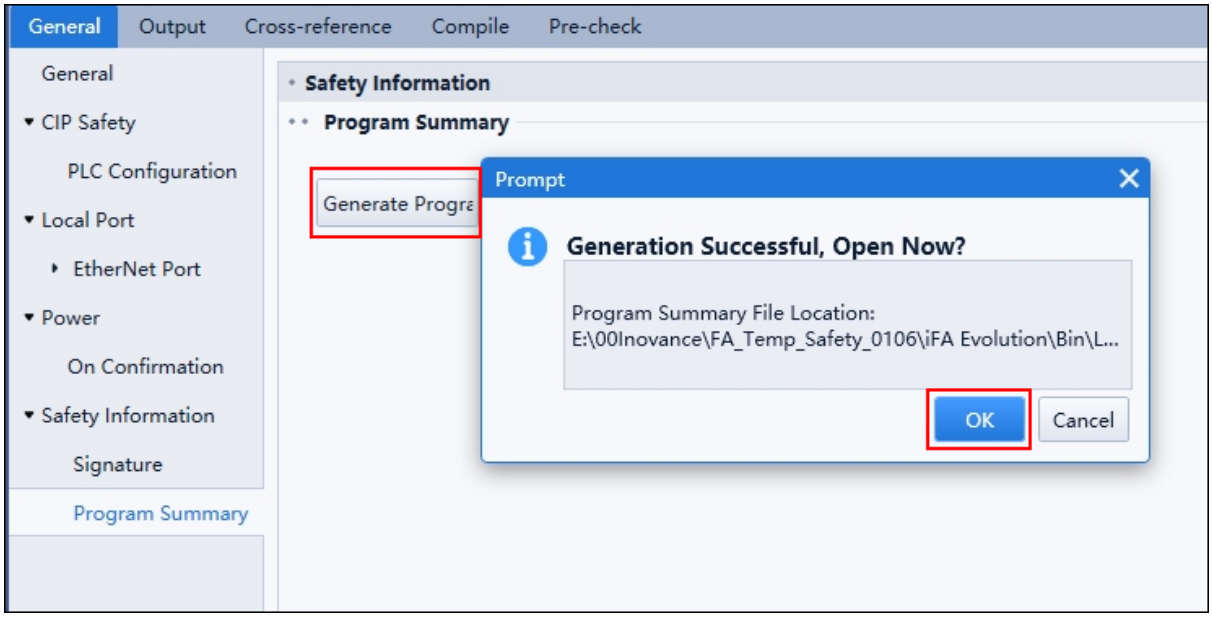
1. In the project tree, double-click the "Device configuration" node under the safety PLC device.
2. In the device configuration interface, click the safety PLC device.
3. In the "General > Safety information > Signature" tab, view signature information.

Item	Description
Signature creation time	Displays the date and time when the signature was generated.
Project overall signature	Displays online/offline signature information and comparison results. This signature is updated when the safety PLC program or hardware configuration changes and the compilation is successful.

Item	Description
Safety program signature	Displays online/offline signature information and comparison results. This signature is updated when the safety PLC program changes and compilation is successful. The scope of the safety program includes: SafeTask, program (PRG, FB), global variable table, and system variable table.
Safety hardware signature	Displays online/offline signature information and comparison results. This signature is updated when the hardware configuration of the safety PLC changes and the compilation is successful. The scope of safety hardware includes: hardware configuration (I/O modules added to local or coupling racks), I/O module parameters, and CIP Safety configuration information for I/O.



4. Confirm that the comparison results for the project overall signature, safety program signature, and safety hardware signature are correct.
5. Select the "Safety information > Program summary" tab.
6. Click "Generate program summary" to generate a PDF file of safety program summary.
7. Click "OK" in the pop-up window to open the summary file.



11.4.6 Task Configuration

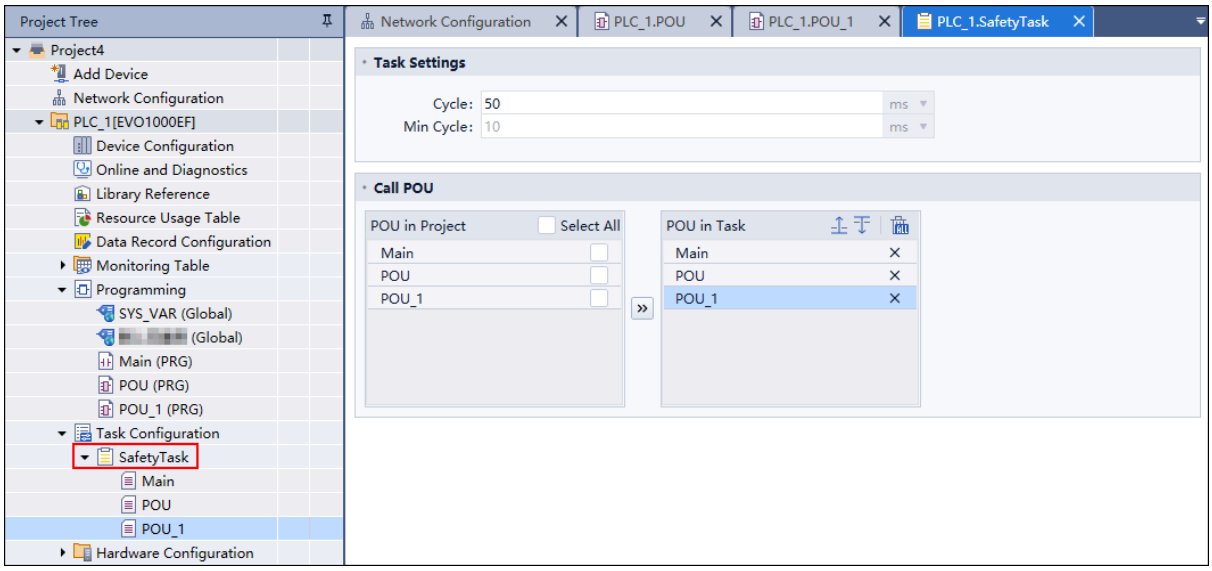
Tasks are used to manage user program calls and are essential in PLC programming. When adding a safety PLC device, a default cycle task is automatically created. Users can configure the task cycle time and add a user program to be called. The device processes user programs under the task configuration; safety PLC devices do not support adding new tasks.

Prerequisites

A user program (PRG) has been created.

Procedure

1. In the left project tree, double-click "Task configuration > SafetyTask" under the safety PLC device.



2. Set the task cycle time.
   The default cycle time is 50 ms, with a configurable range of 10 ms to 100 ms. The minimum cycle is the minimum task cycle automatically calculated by the IDE; the actual task cycle must be greater than or equal to the minimum cycle.

3. Set user program calls.
   a. In the "POU in project" list, check the user programs to be called. Selecting all is supported.
   b. Click » to add them to the "POU in task" list, thereby adding the user programs to the task.

c. Click  to move up/down the selected item in the "POU in task" list. User programs in the list will be executed in order from top to bottom.

d. Click  to the right of a program to remove it from the task. Click  in the upper right corner of the list to remove all added programs.

> NOTE
>
> Considering future expansion of safety control systems or safety programs, it is recommended to set a safety task cycle time with some margin.

11.5 PLC Compilation

The compilation operations in safety PLCs are the same as those in standard PLCs. For details, see 10.7 PLC Compilation in the "PLC Programming" chapter.

> NOTE
>
> The configuration items for safety PLC compilation differ from those in the standard PLC. For details, see the current software interface.

11.6 PLC Download

This section only covers the functional differences between the download of safety PLCs and standard PLCs; for other general functions, see 10.8 PLC Download in the "PLC Programming" chapter.

11.6.1 Full Download

Prerequisites

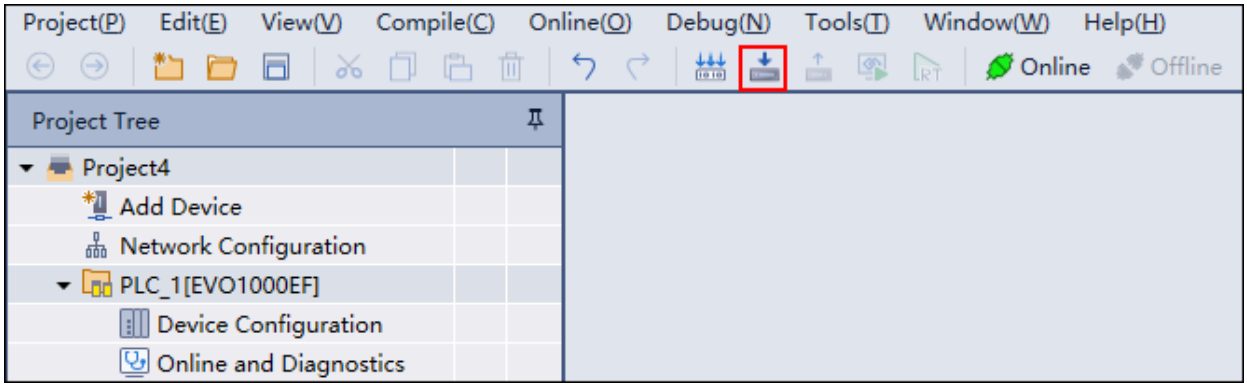
- A program or hardware configuration has changed.
- The project is in offline mode.

Procedure

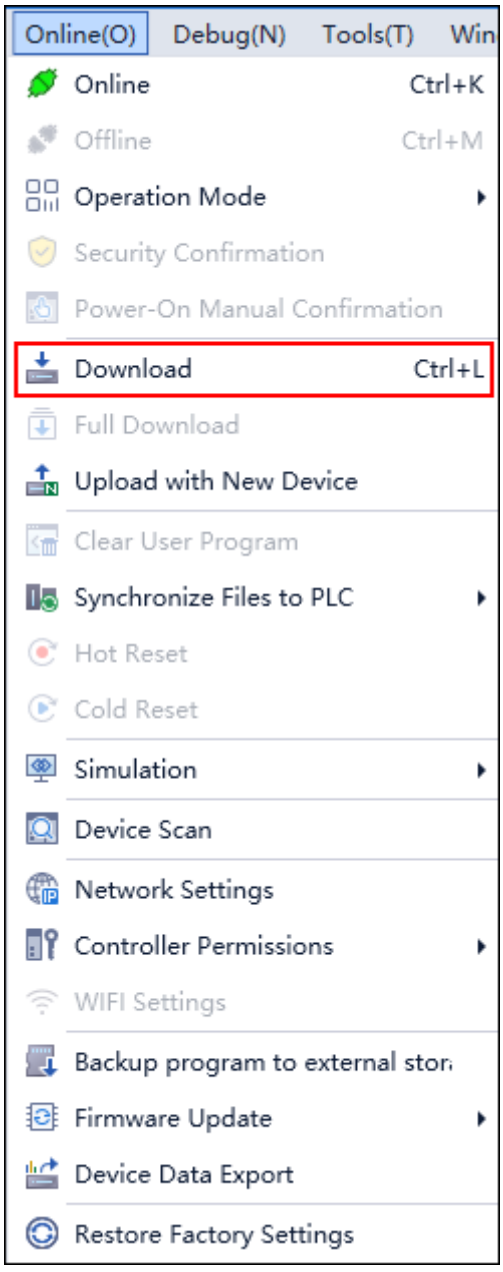
1. Open the download interface.

   There are three download entries: the toolbar, the menu bar, and the right-click menu of the PLC node in the project tree. The download button becomes enabled only when a PLC node or its child node is selected.

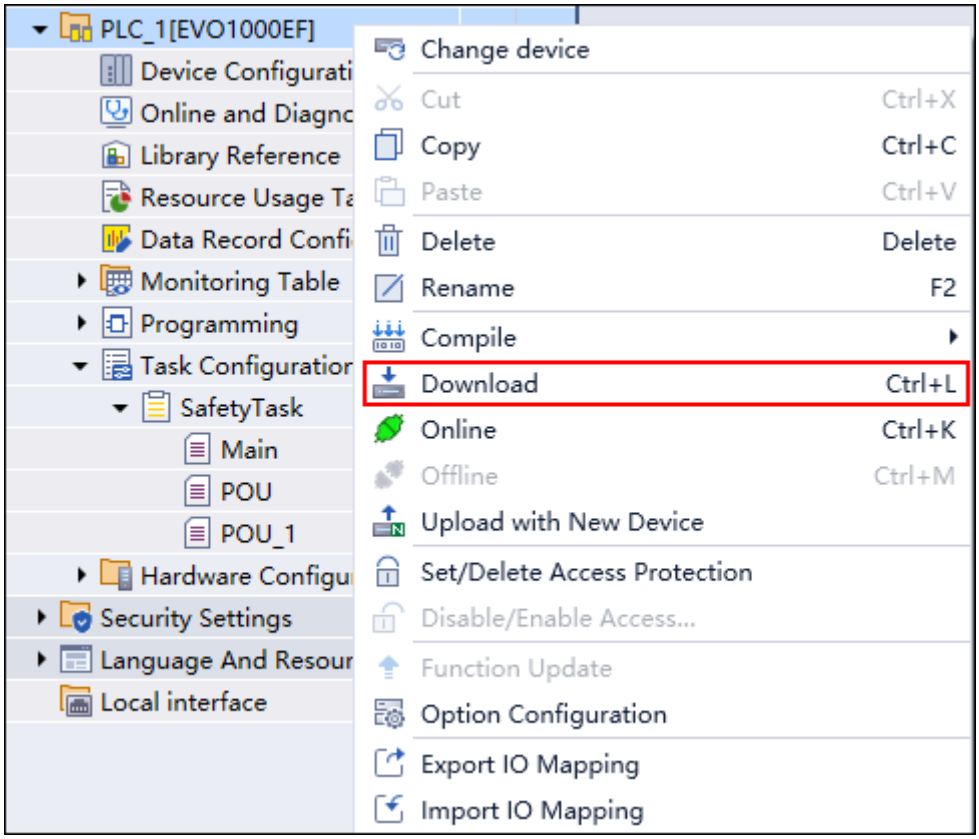
   - Toolbar: Select the corresponding PLC project in the project tree, then click the "Download" icon in the Toolbar, as shown in the figure below.



◦ Menu bar: Select the corresponding PLC project in the project tree, then click "Online > Download" in the menu bar, as shown in the figure below.

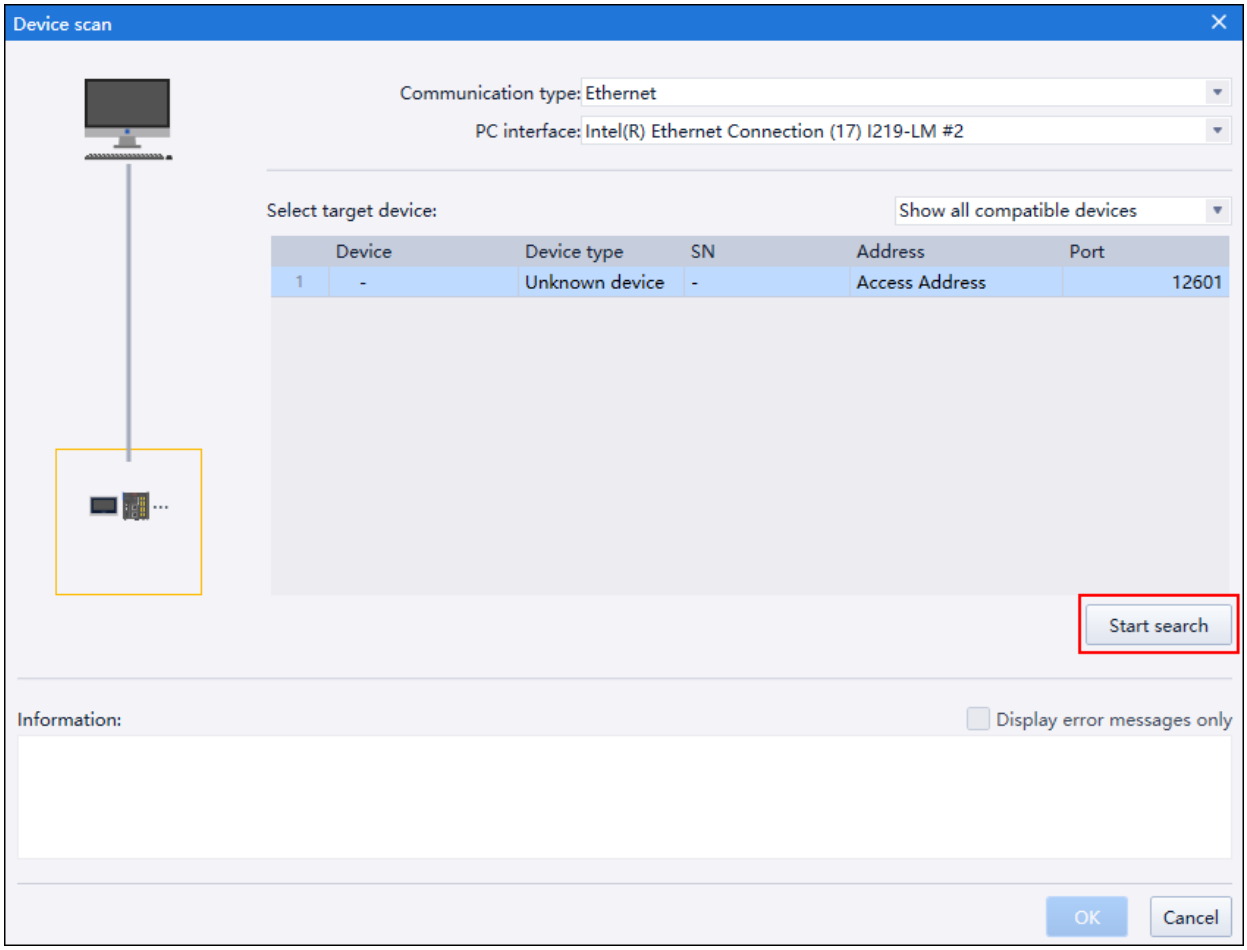


◦ Right-click menu: Right-click the corresponding PLC project in the project tree, then select "Download", as shown in the figure below.



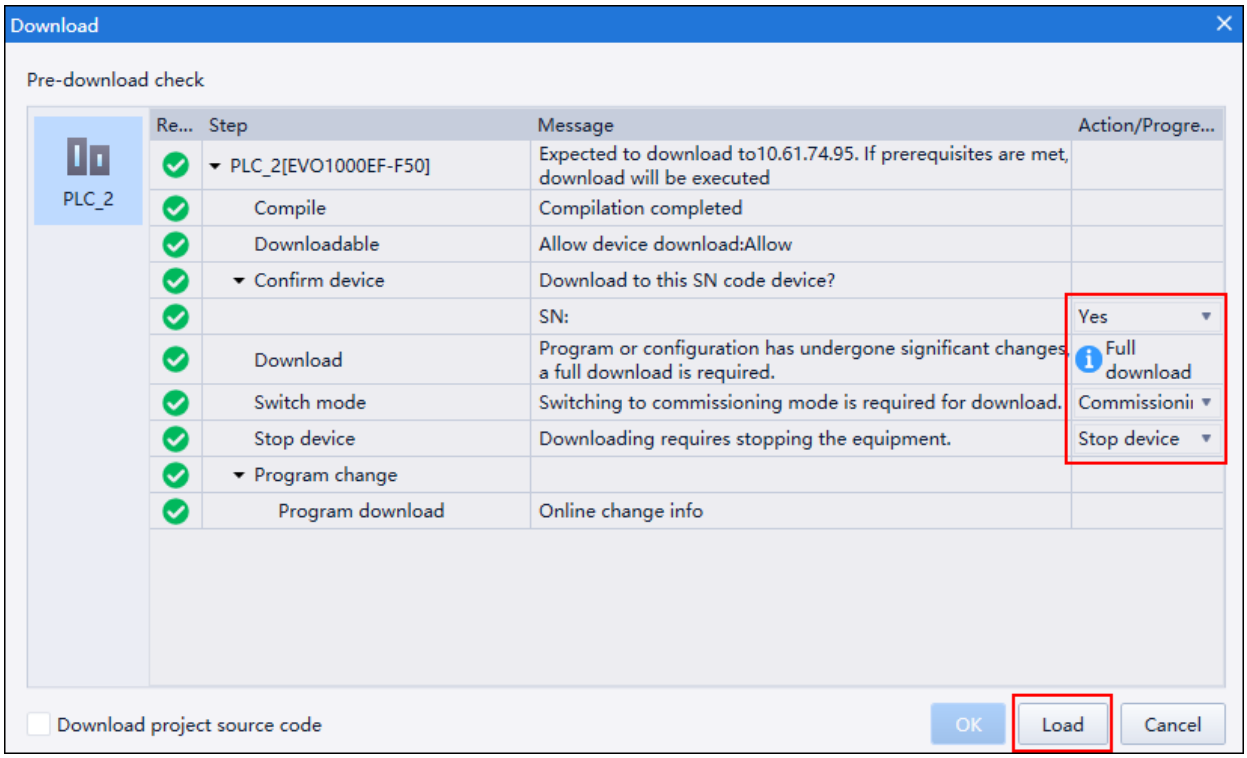
2. Download the configuration.

In the pop-up device scanning window, click "Start search". Select the target device to download, then click "OK", as shown in the figure below.



3. Pre-download checks

A series of checks will be performed before downloading, and relevant information will be prompted. After configuration, click "Load", as shown in the figure below.



The main check items are as follows:

- Check for the device

  Before download, users must confirm that the SN code of the safety PLC matches the target safety PLC to be downloaded.

- Check if mode switching is required

  Before download, if the current PLC is in run mode, users must switch the device to commissioning mode before download.

- Check if shutdown is required

  For programs that must stop the PLC before downloading, users must select the "Stop device" action before downloading.

- Perform automatic compilation

  If compilation has not been performed before downloading, compilation will be automatically executed.

- Check for encryption permission downgrade

  A security prompt will be issued if the encryption permissions are downgraded before downloading.

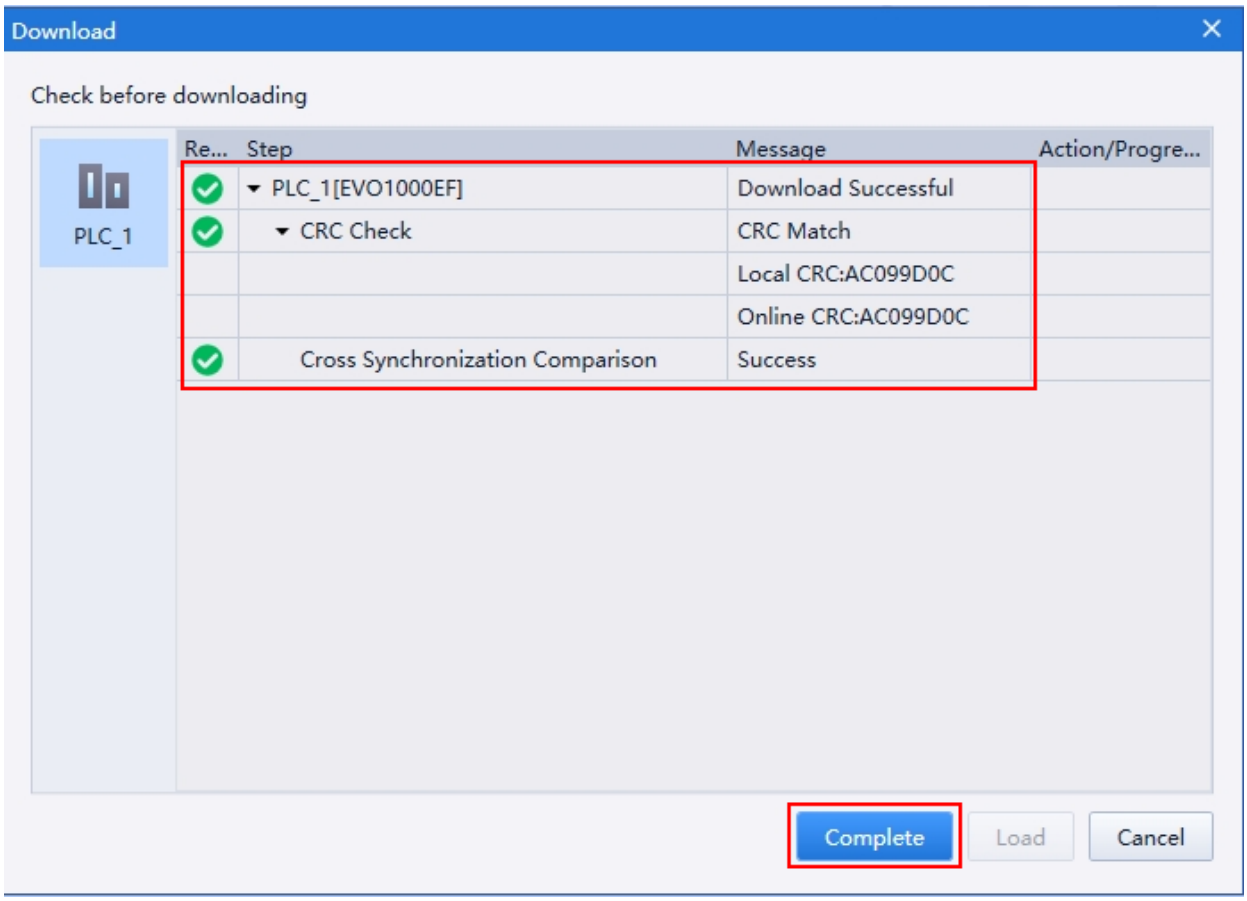
- Check for password

  If a password has been set for the PLC, users must enter the password before downloading; otherwise, downloading is not allowed.

4. Post-download actions

   After the download is completed, the system will perform offline and online CRC checks and cross-synchronization verification. The download will only be considered successful if both verifications

succeed. If either fails, a download failure will be reported As shown in the figure below, confirm that all download results are correct to complete the download.



11.7 PLC Commissioning

11.7.1 PLC Start/Stop

The start/stop operations in safety PLCs are the same as those in standard PLCs. For details, see 10.10.1 PLC Start/Stop in the "PLC Programming" chapter.

11.7.2 PLC Online/Offline

The online/offline operations in safety PLCs are the same as those in standard PLCs. For details, see 10.10.2 PLC Online/Offline in the "PLC Programming" chapter.

11.7.3 Value Display and Calculation

The methods for value display and calculation in safety PLCs are the same as those in standard PLCs. For details, see 10.10.3 Value Display and Calculation in the "PLC Programming" chapter.

11.8 PLC Online Operations

This section only covers the functional differences between the online operations of safety PLCs and standard PLCs; for other general functions, see 10.11 PLC Online Operation in the "PLC Programming" chapter.

- The EIP2 port of the EIP communication unit supports DLR ring networking. For specific operations, see 10.11.16 DLR Ring Network Management of the standard PLC.
- The EIP2 port of the EIP communication unit in the safety PLC supports IP address configuration in the same subnet as the LAN1 port. The standard PLC maintains its original rules – different network ports must have IP addresses in different subnets.

11.8.1 Operation Modes

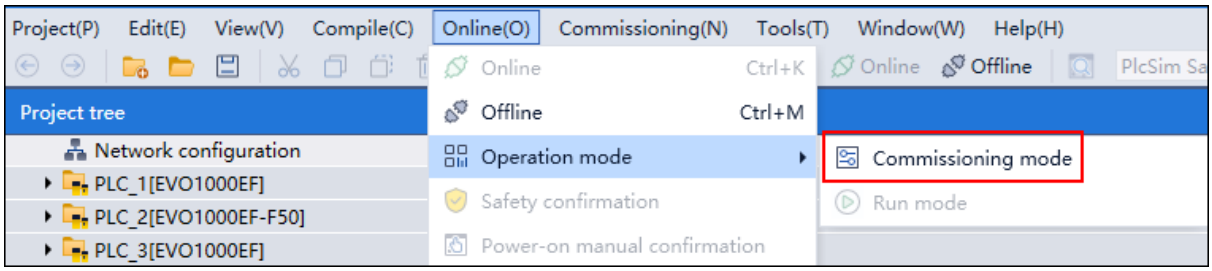
The operation modes of safety PLCs include startup mode, debug mode, run mode, recovery mode, and safety status mode. Manual switching is supported between debug mode and run mode. Startup mode, recovery mode, and safety mode are automatically determined and switched by the PLC based on the device status, and manual switching is not supported.

Prerequisites

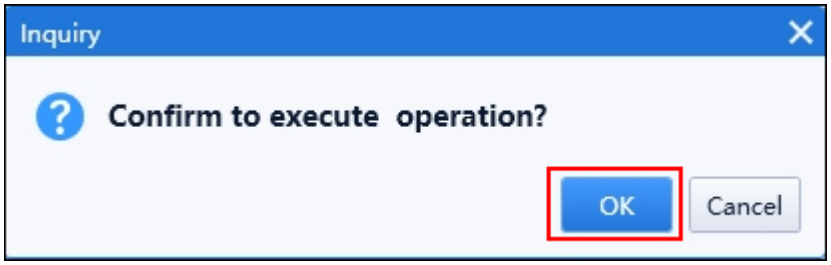
The safety PLC has switched to online status.

Procedure

1. Select "Online > Operation mode" from the menu bar to view the current operation mode of the device.
2. If the current mode is operational mode, click "Commissioning mode" to switch.



3. Click "OK" in the pop-up window to switch the operation mode.



4. If the current operation mode is debug mode, 11.8.2 safety confirmation must be performed before manually switching to run mode.

> **WARNING**
>
> When switching the safety PLC operation mode to debug mode, ensure that all personnel on site have evacuated the hazardous area. Failure to do so may result in output activation, leading to personal injury.

Description of safety PLC operation modes

Operation Mode	Description
Startup mode	The safety PLC initializes hardware and software and performs a power-on self test.
Debug mode (stopped)	Users verify and debug the project in this mode. This mode supports project modification, download, online debugging, variable forcing, releasing and writing. Safety communication is not executed.
Debug mode (running)	Users verify and debug the project in this mode. This mode supports project modification, download, online debugging, variable forcing, releasing and writing. Safety communication is executed.
Run mode	Normal operation mode of the safety PLC. This mode does not support project download, variable forcing, or releasing.
Safety status mode (manual confirmation)	After project safety confirmation and power-on, the system enters this mode and waits for manual confirmation from users. In this mode, safety communication is executed, and input/output values are set to safety values.
Safety status mode (fault)	The system enters this mode after the safety PLC detects a fault in safety diagnostics. In this mode, input/output values are set to safety values.
Recovery mode	The safety PLC enters the recovery mode via the DIP switch. In this mode, backup data is restored to the safety PLC. After recovery, the system waits for power-off and restart.



When the safety PLC enters safety mode, the VAD indicator on the device's segment display maintains the status before entering safety mode.

Cycle communication status of the safety PLC under various operation modes

Operation Mode	Safety Input (CIP Safety)	Standard Input	Safety Output (CIP Safety)	Standard Output	Safety Status
Startup mode	Ignore	Ignore	Ignore	Ignore	Refresh
Debug mode (stopped)	Safety value	Ignore	Safety value	Ignore	Refresh
Debug mode (running)	Refresh	Refresh	Refresh	Refresh	Refresh
Run mode	Refresh	Refresh	Refresh	Refresh	Refresh
Safety status mode (manual confirmation)	Safety value	Refresh	Safety value	Refresh	Remain
Safety status mode (fault)	Safety value	Refresh	Safety value	Refresh	Remain

Operation Mode	Safety Input (CIP Safety)	Standard Input	Safety Output (CIP Safety)	Standard Output	Safety Status
Recovery mode	Ignore	Ignore	Ignore	Ignore	Ignore

Actions during safety PLC mode switching

Previous Operating Mode	New Operating Mode	Action
Startup Mode	Debug mode (stopped)	No project in safety PLC; no other actions
Startup Mode	Safety status mode (manual confirmation)	1. Starts safety communication; input/output values are set to safety values. 2. Starts scheduling and execution of user tasks.
Startup Mode	Safety status mode (fault)	Remains in safety mode; input/output values are set to safety values.
Startup Mode	Run mode	1. Starts safety communication; input/output values are set to real-time values. 2. Starts scheduling and execution of user tasks.
Startup Mode	Recovery mode	Starts restoring backup data from the SD card in the communication unit.
Debug mode (stopped)	Debug mode (running)	1. Starts safety communication; input/output values are set to real-time values. 2. Starts scheduling and execution of user projects.
Debug mode (running)	Debug mode (stopped)	1. Stops safety communication. 2. Stops scheduling and execution of user tasks.
Debug mode	Safety status mode (fault)	Input/output values are set to safety values.
Run mode	Safety status mode (fault)	Input/output values are set to safety values.
Debug mode (running)	Run mode	User project remains running.
Run mode	Debug mode (stopped)	1. Deletes safety confirmation file. 2. Stops safety communication. 3. Stops scheduling and execution of user tasks.

Functions supported by safety PLC in each operation mode

Description	Startup Mode	Debug Mode		Run Mode	Safety Status Mode	Recovery Mode
		Stop	Run			
User program execution	×	×	√	√	√	×
Safety communication	×	×	√	√	√	×
Backplate communication	√	√	√	√	√	√
Variable forcing	×	√	√	×	×	×
Variable releasing	×	√	√	×	×	×
Variable writing	×	√	√	×	×	×
Project download	×	√	√	×	×	×
Safety PLC status monitoring	×	√	√	√	√	√
Program monitoring	×	√	√	√	√	√
Safety data recovery	×	×	×	×	×	√
Restoring factory settings	×	√	√	×	×	×
Clearing User Programs	×	√	√	×	×	×
Firmware upgrade	×	√	√	×	×	×
Cold Reset	×	√	√	×	×	×
Time Settings	×	√	×	×	×	×
Safety confirmation	×	√	√	×	×	×
Device permissions	×	√	√	×	×	×
Power-On Manual Confirmation	×	×	×	×	√[1]	×

[1]: Power-on manual confirmation is only supported when the safety PLC enters the safety mode (manual confirmation) after power-on.

11.8.2 Safety Confirmation

Users must perform a safety confirmation test to confirm that all safety functions, including safety sensors, safety I/O modules, safety PLCs, and safety actuators, meet the safety system requirements. After safety confirmation test, if all safety functions meet the requirements, users must attach the confirmation information to the safety program. This action is called safety confirmation.

After the user program has been debugged, safety confirmation can be performed on the user program. Only user programs that have undergone safety confirmation are allowed to be loaded and executed when the safety PLC is powered on again; otherwise, the user program will be lost.

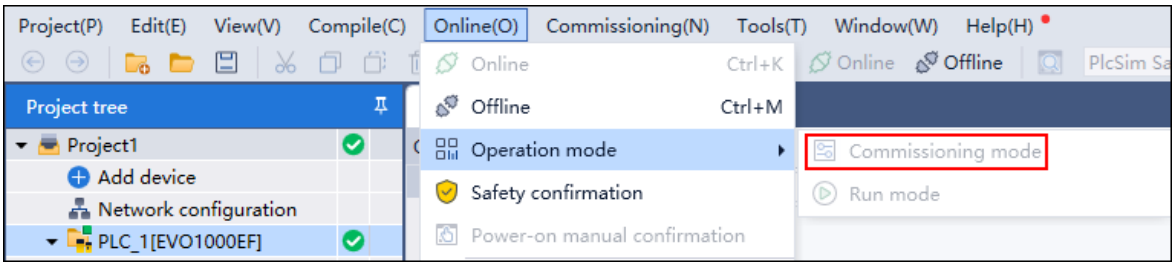
> **WARNING**
>
> For all safety connections, confirm that the calculated response time meets the requirements; otherwise, safety functions may be compromised, potentially resulting in personal injury.

Prerequisites

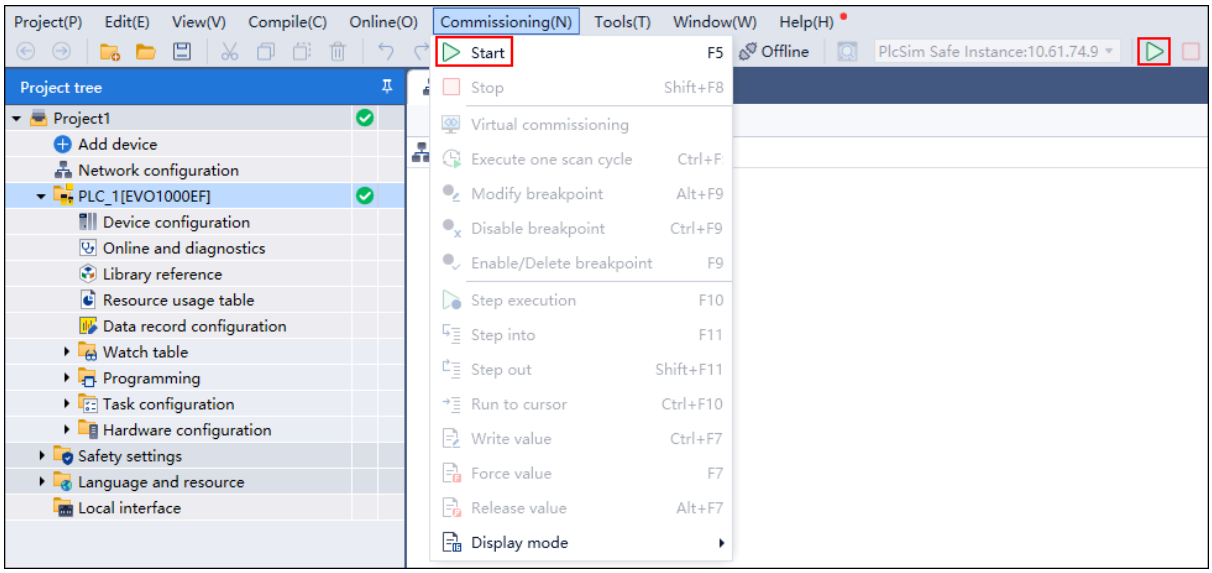
- The safety PLC has switched to online status.
- Before performing safety confirmation on the safety program, complete program debugging first.

Procedure

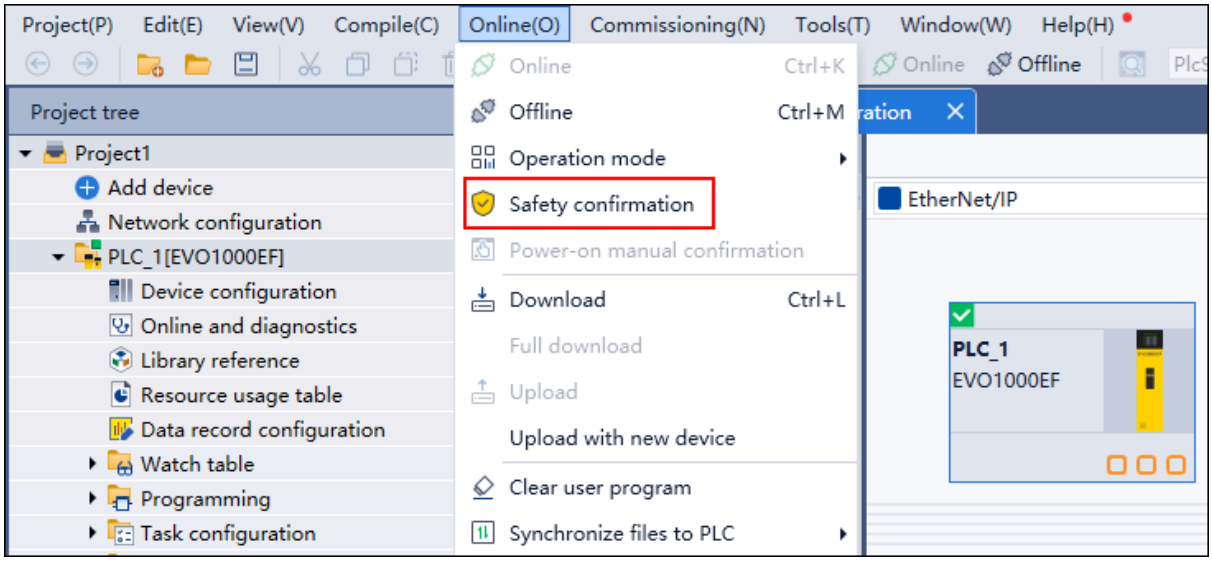
1. In the menu bar, select "Online > Operation mode" and confirm that the safety PLC is in "Commissioning mode".



2. In the menu bar, select "Commissioning > Start", or click , then click "OK" in the pop-up window to switch the device to run mode.



3. Click "Online > Safety confirmation" in the menu bar.



4. Click "OK" in the pop-up dialog box to complete the safety confirmation.



11.8.3 Power-On Manual Confirmation

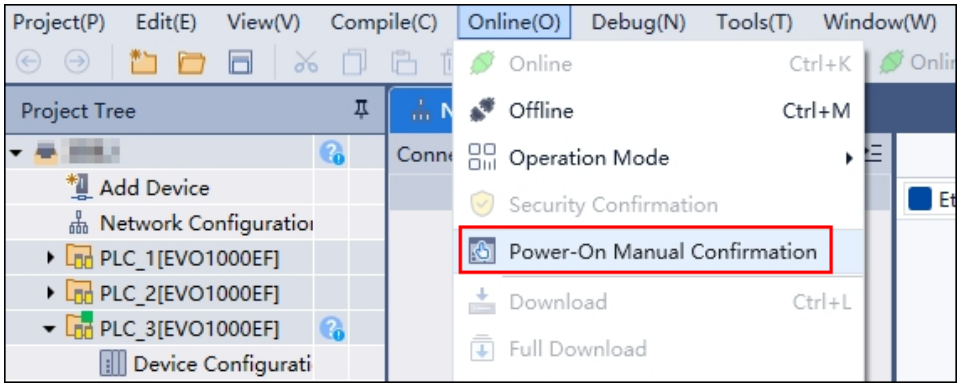
After safety confirmation of the safety program in the safety PLC, when the safety PLC is powered on and completes self test, it is ready to run. Users can choose either manual or automatic confirmation. If automatic confirmation is selected, the safety PLC will directly enter the run mode; if manual confirmation is selected, the PLC enters the run mode before manual confirmation by users. The system defaults to manual confirmation mode.

Prerequisites

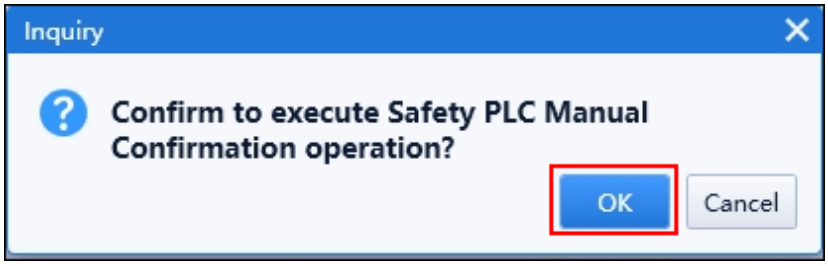
- After the safety PLC clears faults and is powered on again, the device is in safety mode, waiting for manual confirmation.
- The safety PLC is switched to online status using the iFA Evolution software (the current projects in use must match the projects in the safety PLC).

Procedure

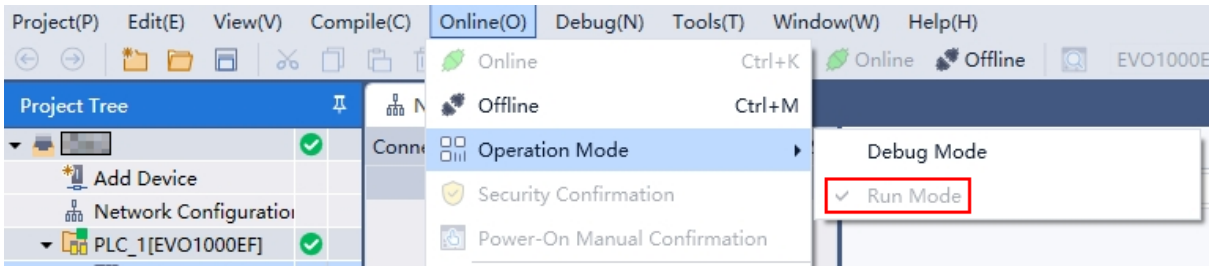
1. Click "Online > Power-on manual confirmation" in the menu bar.



2. Click "OK" in the pop-up window to complete the power-on manual confirmation of the safety PLC.



3. Select "Online > Operation mode" from the menu bar to check the current operation mode of the safety PLC. If the current mode is run mode, it indicates that the power-on manual confirmation is successful.



11.8.4 Operation Log

This section only covers the functional differences between operation logs in safety PLCs and standard PLCs; for other general functions, see 10.11.21 Operation Log in the "PLC Programming" chapter.

> NOTE
>
> Exporting and clearing logs require permission control. If the safety PLC has permission protection enabled, permission verification must be passed before execution.

11.9 Safety PLC Simulation Mode

This section only covers the functional differences between the simulation mode in safety PLCs and standard PLCs; for other general functions, see 10.13 PLC Simulation Mode in the "PLC Programming" chapter.

- Supports logic simulation.
- Supports online program commissioning.

11.10 Common Safety Instructions

11.10.1 Basic Instructions

11.10.1.1 Comparison Operations

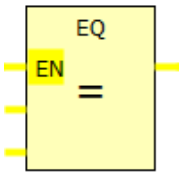
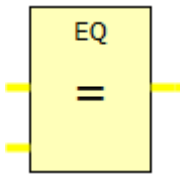
11.10.1.1.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
11.10.1.1.2 EQ	FC	Equal to	SafetyInstructions	Applicable to library version 1.0.0.0/1.1.0.0
11.10.1.1.3 GE	FC	Greater than or equal to		
11.10.1.1.4 GT	FC	Greater than		
11.10.1.1.5 LE	FC	Smaller than or equal to		
11.10.1.1.6 LT	FC	Less than		
11.10.1.1.7 NE	FC	Not equal		

11.10.1.1.2 EQ

EQ determines whether the two input values are equal. If the first input value equals the second, the output is TRUE; otherwise, the output is FALSE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
EQ	Equal to	FC		

- **Related variables**

Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Comparison object	Follows data type	–	The value to be compared.
	In2	Comparison object	Follows data type	–	The value to be compared.
Output	Out	Comparison result	Follows data type	–	Comparison result

Data types

	Boolean		Bit String				Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In1	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	√	–	–	–	–	–	–	–	–	–	–

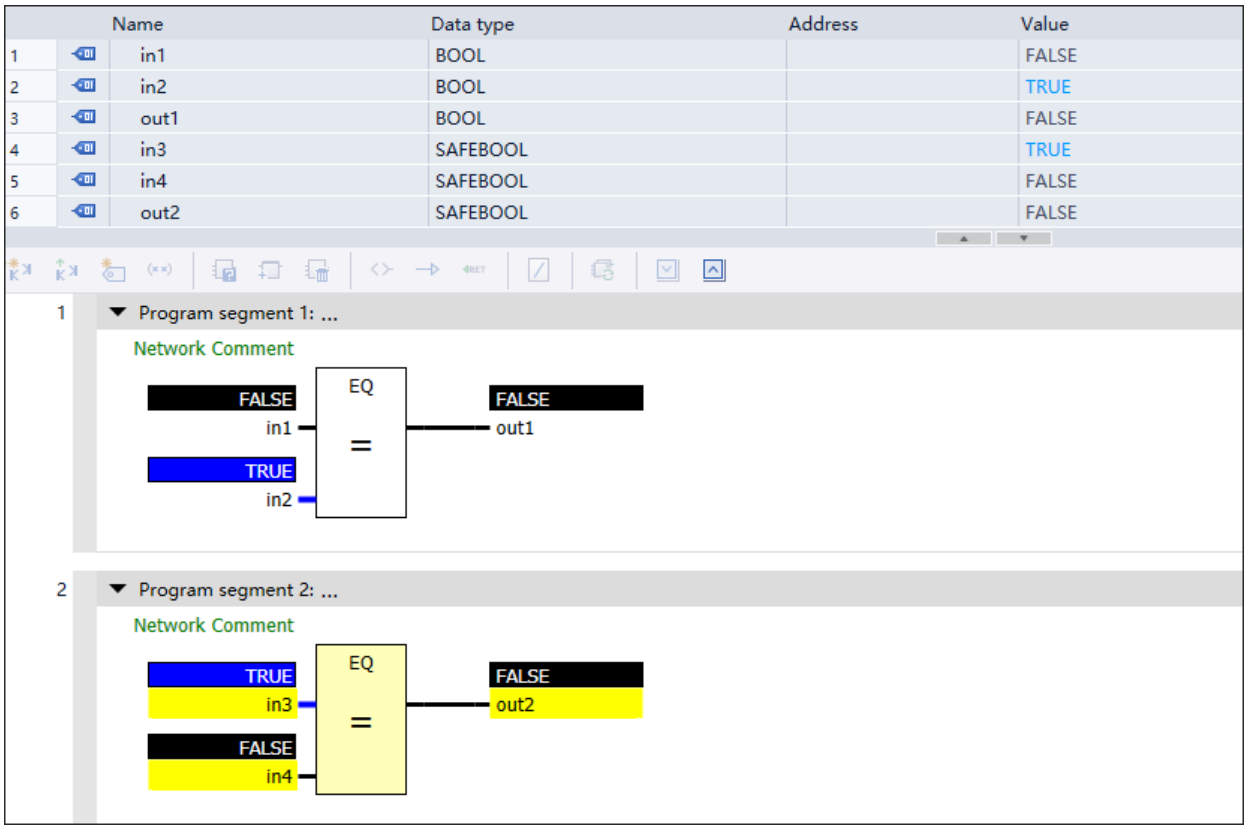
- **Function description**

  Check whether the values of the two inputs, "In1" and "In2", are equal.

  If they are equal, the comparison result "Out" is TRUE. In all other cases, the value of "Out" is FALSE.

- **Program example**

  FBD:



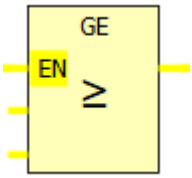
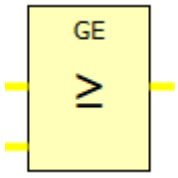
- **Precautions**

  If the input/output parameter is omitted, a fault will occur during compilation. However, if the output terminal is unused or the input terminal for the next instruction is not connected, no fault will occur.

11.10.1.1.3 GE

GE compares the two input values. If the first input value is greater than or equal to the second, the output is TRUE; otherwise, the output is FALSE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
GE	Greater than or equal to	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Comparison object	Follows data type	–	The value to be compared.
	In2	Comparison object	Follows data type	–	The value to be compared.
Output	Out	Comparison result	Follows data type	–	Comparison result

  Data types

	Boolean		Bit String				Integer				Time Format	
	BOOL	SAFE BOOL	BYTE	SAFE BYTE	WORD	SAFE WORD	INT	SAFEINT	DINT	SAFE DINT	TIME	SAFE TIME
In1	–	–	–	–	–	–	√	√	√	√	√	√
In2	–	–	–	–	–	–	√	√	√	√	√	√
Out	√	√	–	–	–	–	–	–	–	–	–	–

- **Function description**

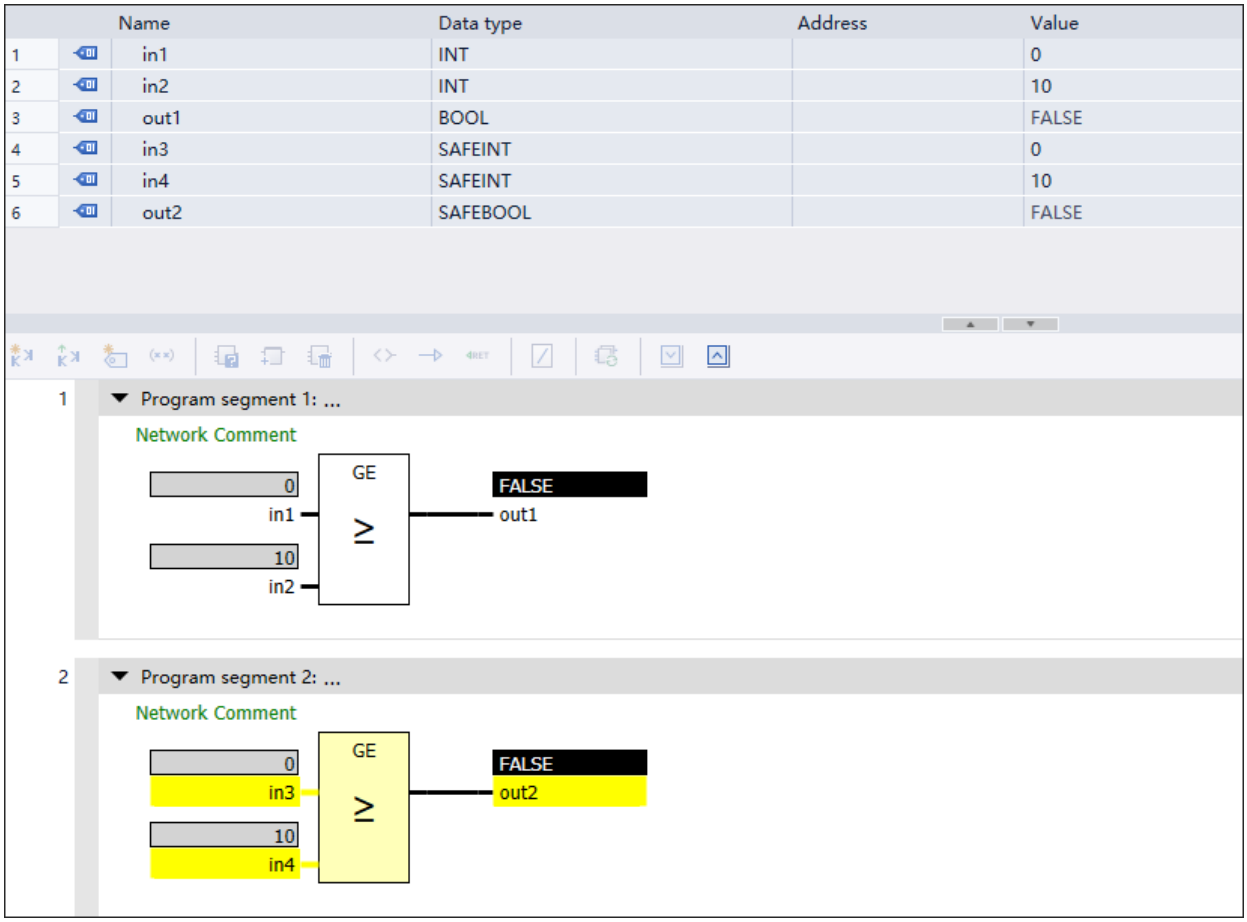
  Compare the values of the two inputs, "In1" and "In2".

  If "In1" ≥ "In2", the output is TRUE; otherwise, the output is FALSE.

Input/Output Data Type	Comparison Basis
INT, SAFEINT, DINT, SAFEDINT	Compare signed values.
TIME, SAFETIME	Compare days, hours, minutes, seconds, and milliseconds.

- **Program example**

  FBD:

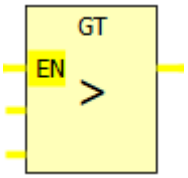
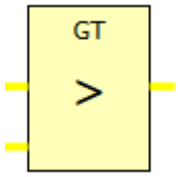


- **Precautions**
  - If the input/output parameter is omitted, a fault will occur during compilation. However, if the output terminal is unused or the input terminal for the next instruction is not connected, no fault will occur.
  - "In1" and "In2" follow the same comparison rule as safety and standard data types: they can be compared if they have the same format and size, but cannot be compared if they are of WORD and INT types. However, for integer types, comparisons between INT/DINT and SAFEINT are supported.
  - The input condition varies depending on whether the output is a safety or standard data type. If the condition is not met, a compilation fault will occur.
  - When assigning a safety data type variable to the output terminal, set both input terminals to safety data type variables; otherwise, a compilation fault will occur.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.1.4 GT

GT compares the two input values. When the first input value is greater than the second, the output is TRUE; otherwise, the output is FALSE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
GT	Greater than	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Comparison object	Follows data type	–	The value to be compared.
	In2	Comparison object	Follows data type	–	The value to be compared.
Output	Out	Comparison result	Follows data type	–	Comparison result

  Data types

	Boolean		Bit String				Integer				Time Format	
	BOOL	SAFE BOOL	BYTE	SAFE BYTE	WORD	SAFE WORD	INT	SAFEINT	DINT	SAFE DINT	TIME	SAFE TIME
In1	–	–	–	–	–	–	√	√	√	√	√	√
In2	–	–	–	–	–	–	√	√	√	√	√	√
Out	√	√	–	–	–	–	–	–	–	–	–	–

- **Function description**

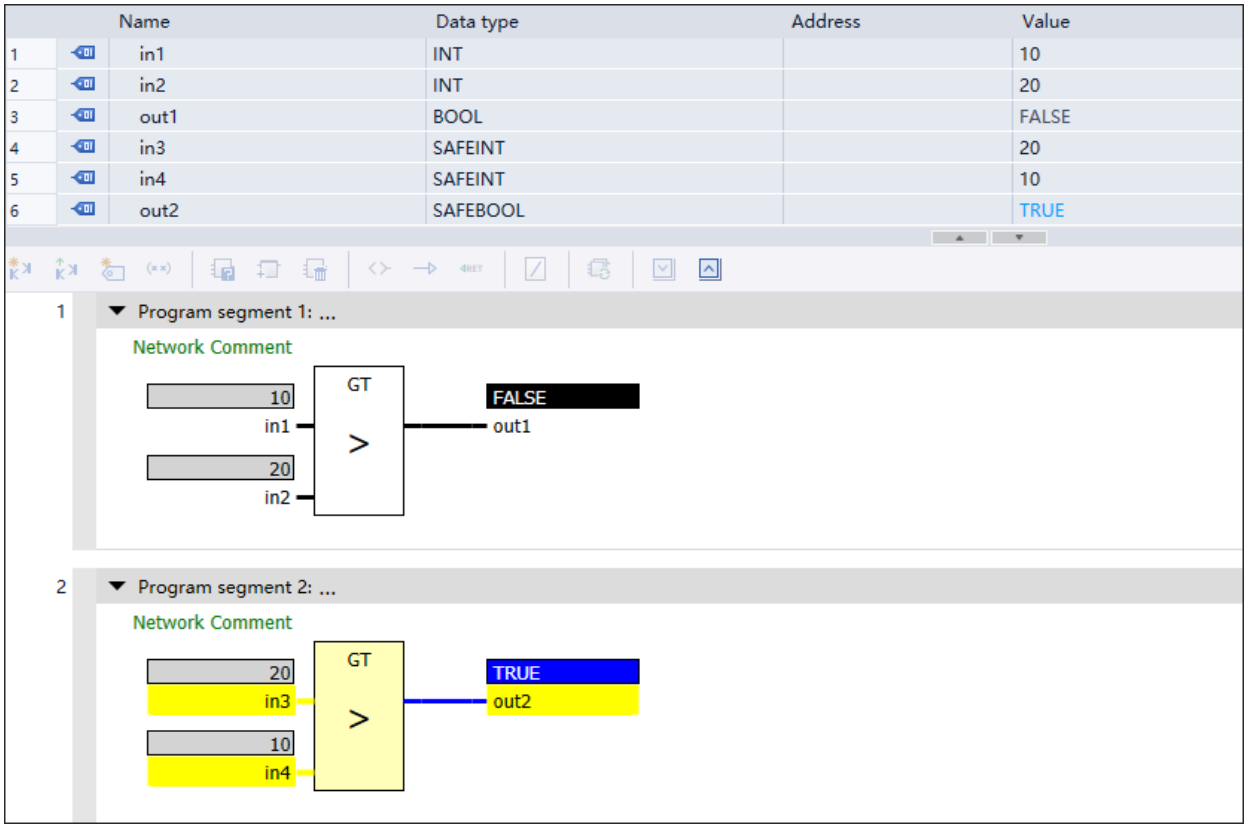
  Compare the values of the two inputs, "In1" and "In2".

  If "In1" > "In2", the output is TRUE; otherwise, the output is FALSE.

Input/Output Data Type	Comparison Basis
INT, SAFEINT, DINT, SAFEDINT	Compare signed values.
TIME, SAFETIME	Compare days, hours, minutes, seconds, and milliseconds.

- **Program example**

  FBD:

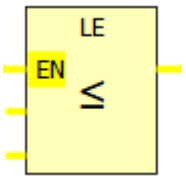
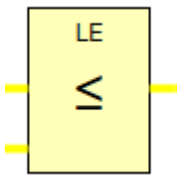


- **Precautions**
  - If the input/output parameter is omitted, a fault will occur during compilation. However, if the output terminal is unused or the input terminal for the next instruction is not connected, no fault will occur.
  - "In1" and "In2" follow the same comparison rule as safety and standard data types: they can be compared if they have the same format and size, but cannot be compared if they are of WORD and INT types. However, for integer types, comparisons between INT/DINT and SAFEINT are supported.
  - The input condition varies depending on whether the output is a safety or standard data type. If the condition is not met, a compilation fault will occur.
  - When assigning a safety data type variable to the output terminal, set both input terminals to safety data type variables; otherwise, a compilation fault will occur.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.1.5 LE

LE compares the two input values. When the first input value is smaller than the second, the output is TRUE; otherwise, the output is FALSE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
LE	Smaller than or equal to	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Comparison object	Follows data type	–	The value to be compared.
	In2	Comparison object	Follows data type	–	The value to be compared.
Output	Out	Comparison result	Follows data type	–	Comparison result

Data types

	Boolean		Bit String				Integer				Time Format	
	BOOL	SAFE BOOL	BYTE	SAFE BYTE	WORD	SAFE WORD	INT	SAFEINT	DINT	SAFE DINT	TIME	SAFE TIME
In1	–	–	–	–	–	–	√	√	√	√	√	√
In2	–	–	–	–	–	–	√	√	√	√	√	√
Out	√	√	–	–	–	–	–	–	–	–	–	–

- **Function description**

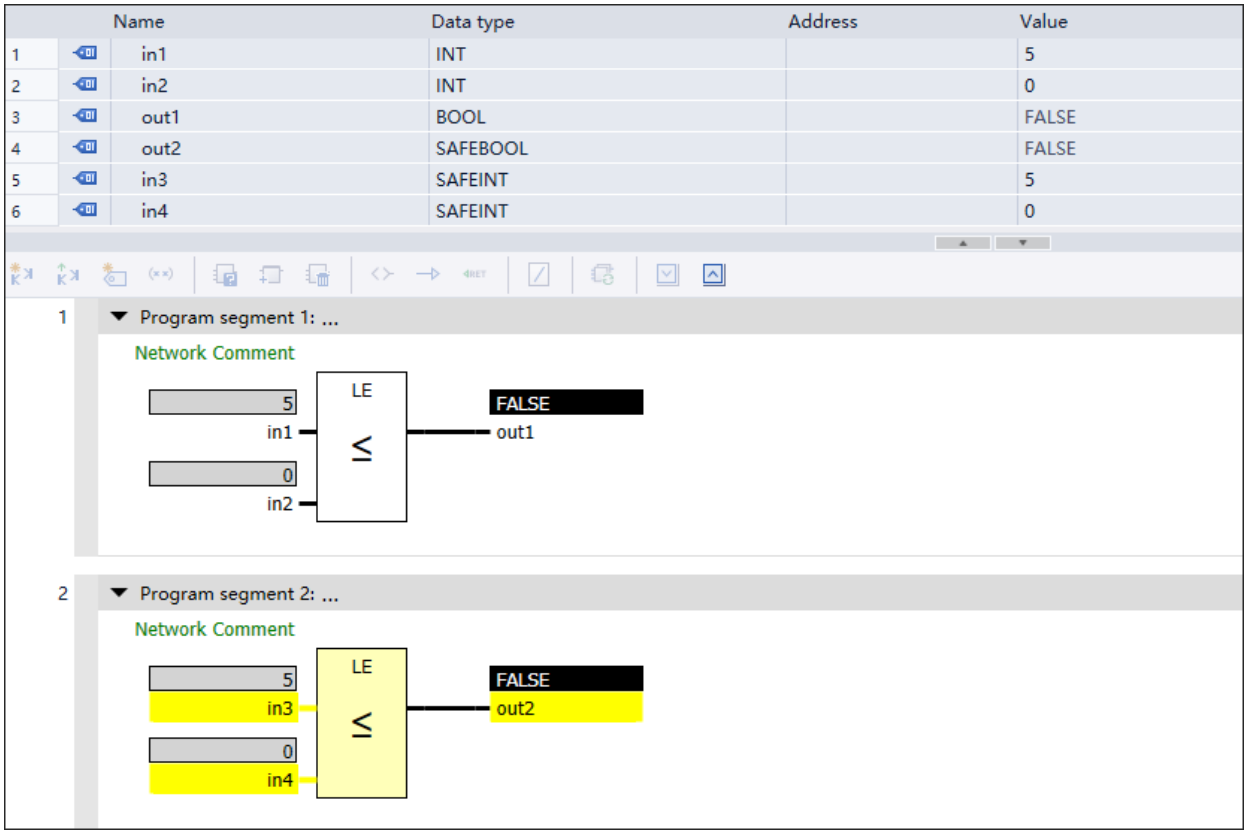
  Compare the values of the two inputs, "In1" and "In2".

  If "In1" ≤ "In2", the output is TRUE; otherwise, the output is FALSE.

Input/Output Data Type	Comparison Basis
INT, SAFEINT, DINT, SAFEDINT	Compare signed values.
TIME, SAFETIME	Compare days, hours, minutes, seconds, and milliseconds.

- **Program example**

  FBD:



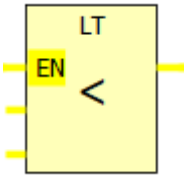
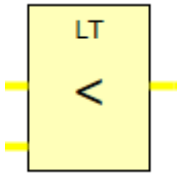
- **Precautions**
  - If the input/output parameter is omitted, a fault will occur during compilation. However, if the output terminal is unused or the input terminal for the next instruction is not connected, no fault will occur.
  - "In1" and "In2" follow the same comparison rule as safety and standard data types: they can be compared if they have the same format and size,

    but cannot be compared if they are of WORD and INT types. However, for integer types, comparisons between INT/DINT and SAFEINT are supported.
  - The input condition varies depending on whether the output is a safety or standard data type. If the condition is not met, a compilation fault will occur.
  - When assigning a safety data type variable to the output terminal, set both input terminals to safety data type variables; otherwise, a compilation fault will occur.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.1.6 LT

LT compares the two input values. When the first input value is smaller than the second, the output is TRUE; otherwise, the output is FALSE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
LT	Less than	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Comparison object	Follows data type	–	The value to be compared.
	In2	Comparison object	Follows data type	–	The value to be compared.
Output	Out	Comparison result	Follows data type	–	Comparison result

  Data types

	Boolean		Bit String				Integer				Time Format	
	BOOL	SAFE BOOL	BYTE	SAFE BYTE	WORD	SAFE WORD	INT	SAFEINT	DINT	SAFE DINT	TIME	SAFE TIME
In1	–	–	–	–	–	–	√	√	√	√	√	√
In2	–	–	–	–	–	–	√	√	√	√	√	√
Out	√	√	–	–	–	–	–	–	–	–	–	–

- **Function description**

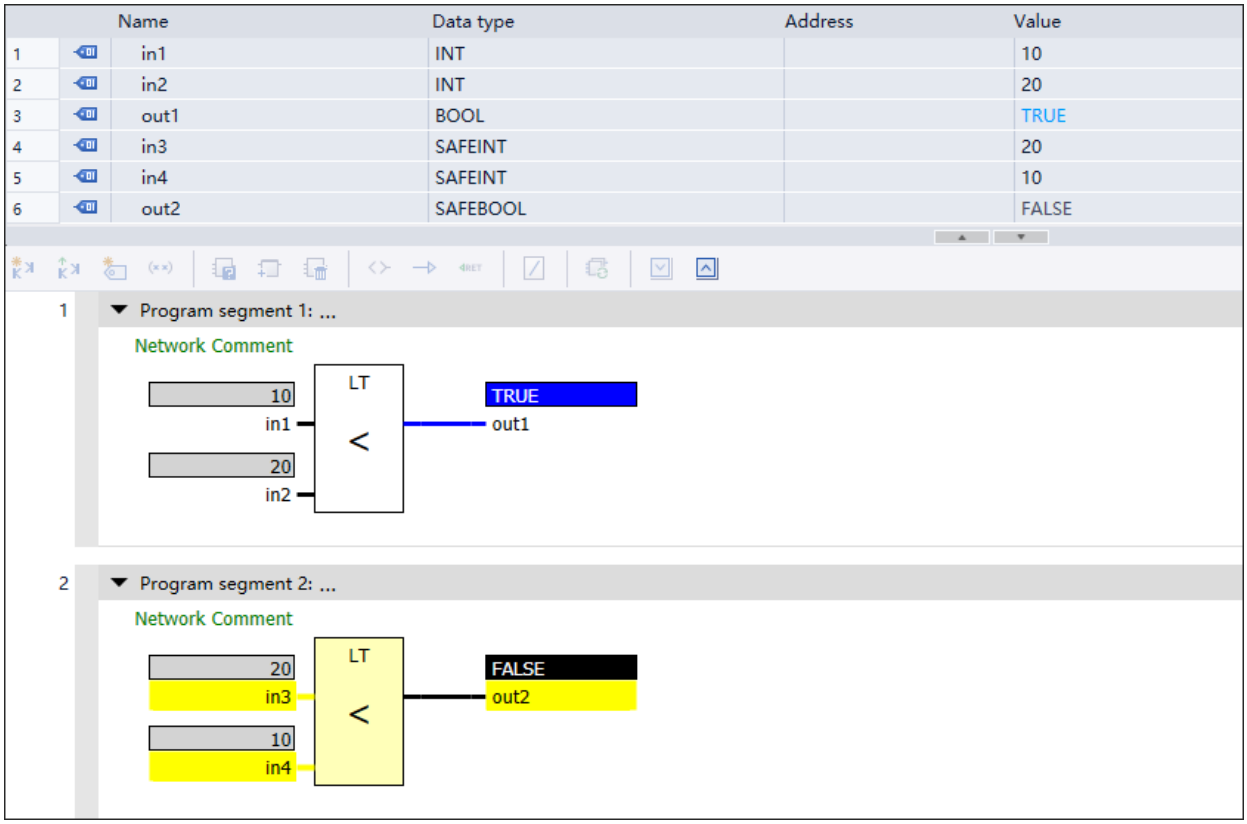
  Compare the values of the two inputs, "In1" and "In2".

  If "In1" < "In2", the output is TRUE; otherwise, the output is FALSE.

Input/Output Data Type	Comparison Basis
INT, SAFEINT, DINT, SAFEDINT	Compare signed values.
TIME, SAFETIME	Compare days, hours, minutes, seconds, and milliseconds.

- **Program example**

  FBD:

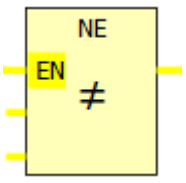
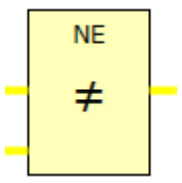


- **Precautions**
  - If the input/output parameter is omitted, a fault will occur during compilation. However, if the output terminal is unused or the input terminal for the next instruction is not connected, no fault will occur.
  - "In1" and "In2" follow the same comparison rule as safety and standard data types: they can be compared if they have the same format and size, but cannot be compared if they are of WORD and INT types. However, for integer types, comparisons between INT/DINT and SAFEINT are supported.
  - The input condition varies depending on whether the output is a safety or standard data type. If the condition is not met, a compilation fault will occur.
  - When assigning a safety data type variable to the output terminal, set both input terminals to safety data type variables; otherwise, a compilation fault will occur.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.1.7 NE

NE compares the two input values. When the first input value is not equal to the second, the output is TRUE; otherwise, the output is FALSE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
NE	Not equal	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Comparison object	Follows data type	–	The value to be compared.
	In2	Comparison object	Follows data type	–	The value to be compared.
Output	Out	Comparison result	Follows data type	–	Comparison result

Data types

	Boolean		Bit String				Integer				Time Format	
	BOOL	SAFE BOOL	BYTE	SAFE BYTE	WORD	SAFE WORD	INT	SAFEINT	DINT	SAFE DINT	TIME	SAFE TIME
In1	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	√	–	–	–	–	–	–	–	–	–	–

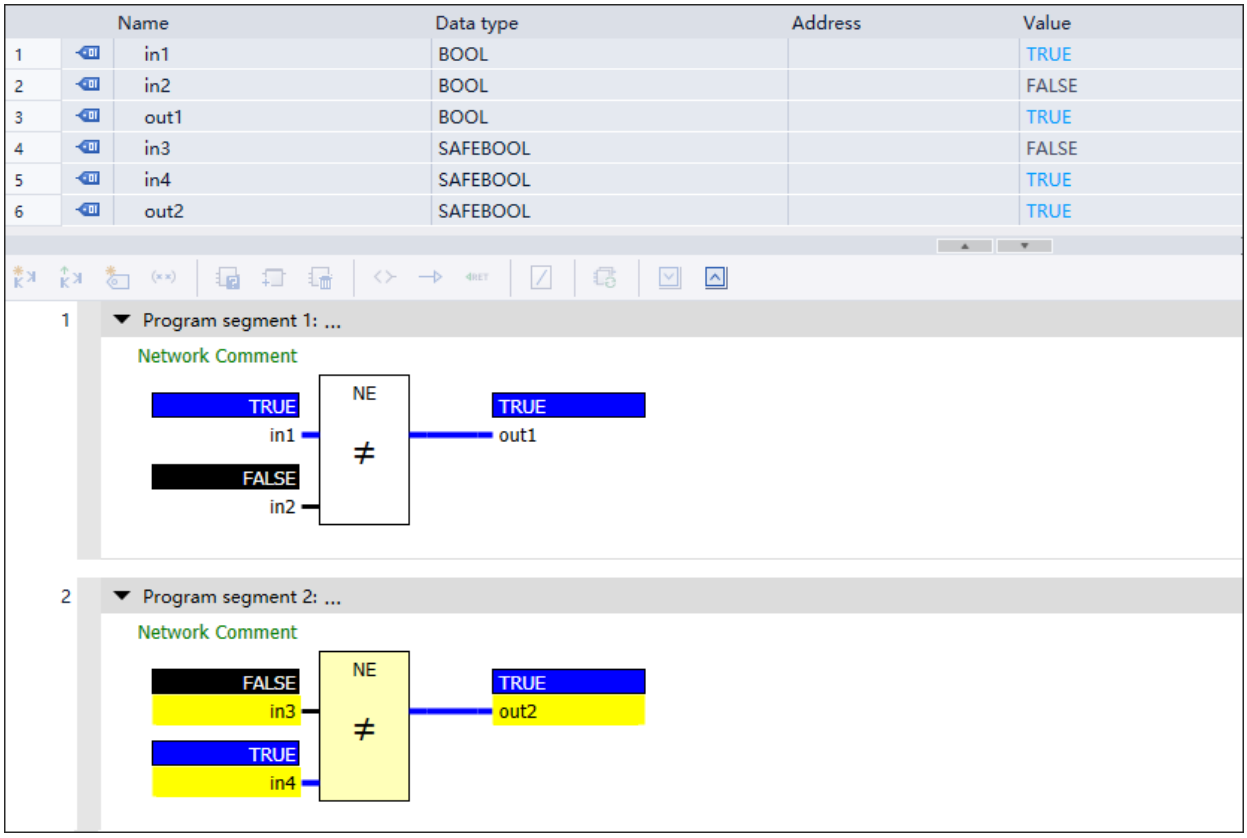
- **Function description**

  Check whether the values of the two inputs, "In1" and "In2", are not equal.

  If they are not equal, the comparison result "Out" is TRUE. If they are equal, the value of "Out" is FALSE.

- **Program example**

  FBD:



- **Precautions**
  - If the input/output parameter is omitted, a fault will occur during compilation. However, if the output terminal is unused or the input terminal for the next instruction is not connected, no fault will occur.

- "In1" and "In2" follow the same comparison rule as safety and standard data types: they can be compared if they have the same format and size, but cannot be compared if they are of WORD and INT types. However, for integer types, comparisons between INT/DINT and SAFEINT are supported.
- The input condition varies depending on whether the output is a safety or standard data type. If the condition is not met, a compilation fault will occur.
- When assigning a safety data type variable to the output terminal, set both input terminals to safety data type variables; otherwise, a compilation fault will occur.
- When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.2 Logic Operations

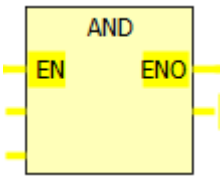
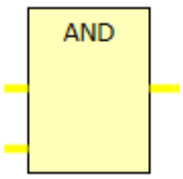
11.10.1.2.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
11.10.1.2.2 AND	FC	and	SafetyInstructions	Applicable to library version 1.0.0.0/1.1.0.0
11.10.1.2.3 NOT	FC	Not		
11.10.1.2.4 OR	FC	Or		
11.10.1.2.5 XOR	FC	Exclusive OR		

11.10.1.2.2 AND

Bitwise AND: When the same bit in all inputs is 1, the corresponding output bit is 1; if not, the output bit is 0.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
AND	and	FC		

- **Related variables**

Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1 to InN	Operand	FALSE, TRUE	FALSE	Operand
Output	Out	Operation result	FALSE, TRUE	–	Operation result

Data types

	Boolean		Bit String				Integer				Time Format	
	BOOL	SAFE BOOL	BYTE	SAFE BYTE	WORD	SAFE WORD	INT	SAFEINT	DINT	SAFE DINT	TIME	SAFE TIME
In0 to InN	√	√	–	–	–	–	–	–	–	–	–	–
Out	√	√	–	–	–	–	–	–	–	–	–	–

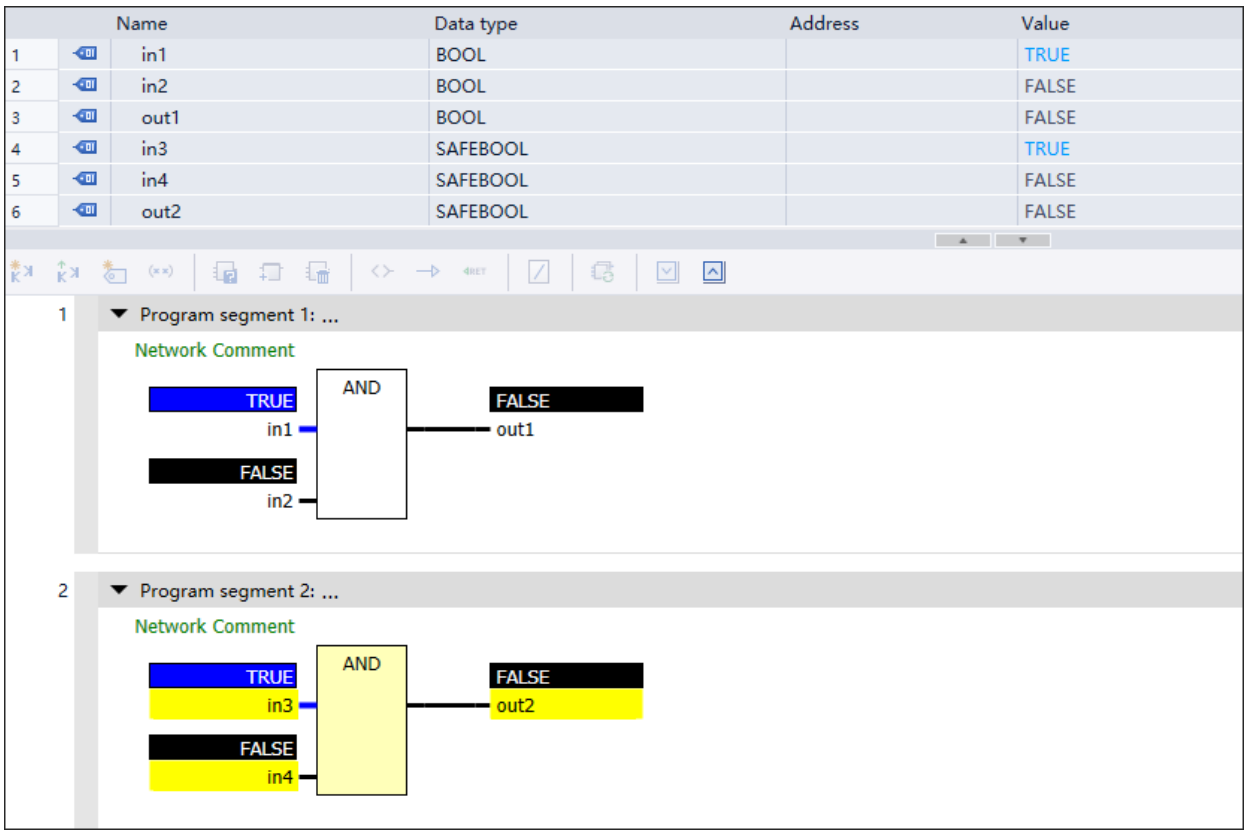
- **Function description**

  AND operation is performed on multiple Boolean operands "In1 to InN".

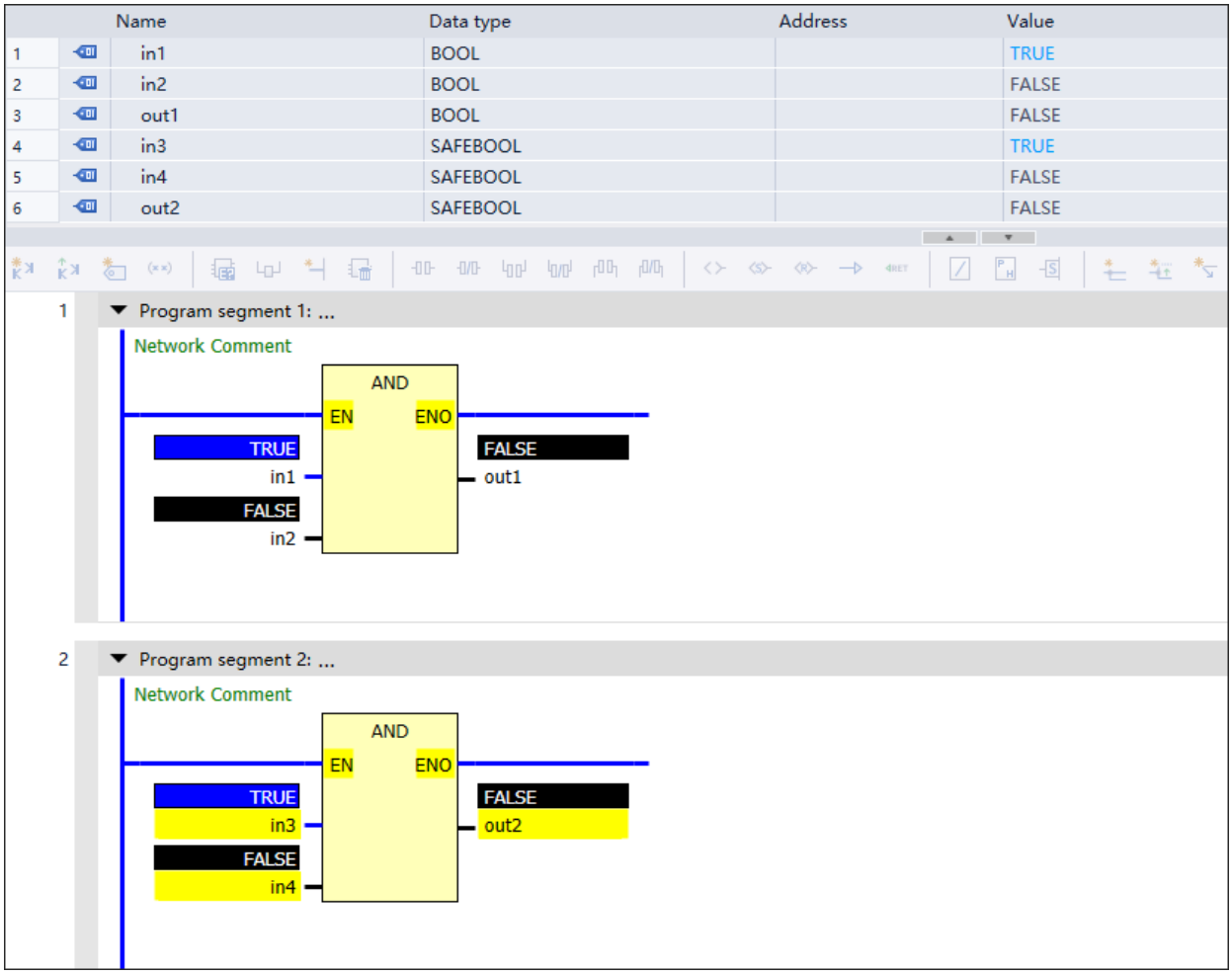
  The operation result is TRUE when all input operands are TRUE. Otherwise, the operation result is FALSE.

- **Program example**

  FBD:



  LD:

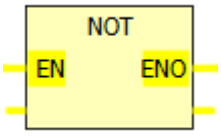
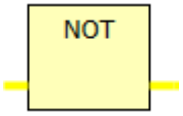


- **Precautions**
  - The data types of "In1" to "InN" and "Out" can only be set to Boolean.
  - The input condition varies depending on whether the output is a safety or standard data type. If the condition is not met, a compilation fault will occur.
  - When assigning a safety data type variable to the output terminal, set at least one safety data type variable for the input terminals.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.2.3 NOT

NOT inverts the bits of a Boolean value.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
NOT	Not	FC		

- **Related variables**

Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Operand	FALSE, TRUE	–	Operand

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Operation result	FALSE, TRUE	FALSE	Operation result

Data types

	Boolean		Bit String				Integer				Time Format	
	BOOL	SAFE BOOL	BYTE	SAFE BYTE	WORD	SAFE WORD	INT	SAFEINT	DINT	SAFE DINT	TIME	SAFE TIME
In	√	√	–	–	–	–	–	–	–	–	–	–
Out	√	√	–	–	–	–	–	–	–	–	–	–

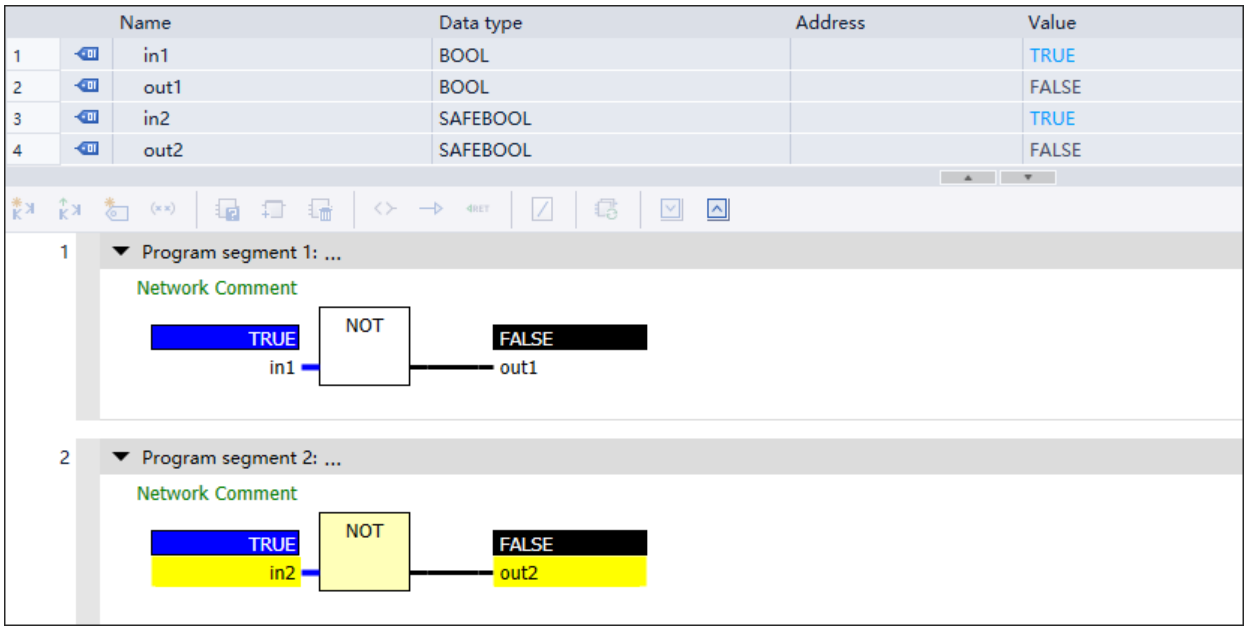
- **Function description**

  Inverts FALSE and TRUE for Boolean operand "In".

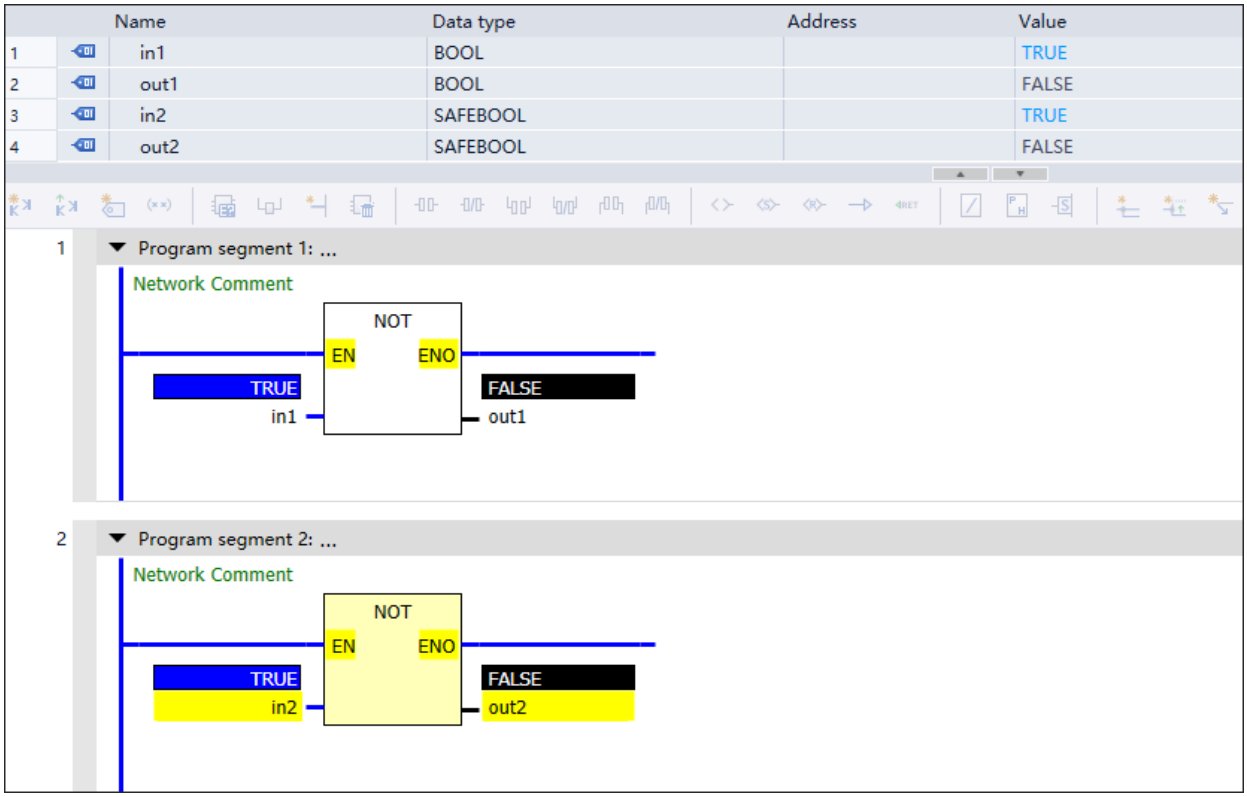
Bits of In	Bits of Out
FALSE	TRUE
TRUE	FALSE

- **Program example**

  FBD:



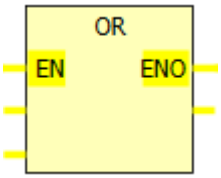
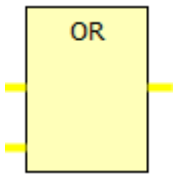
  LD:



11.10.1.2.4 OR

Bitwise OR: If at least one of the same bit of all inputs is non-zero, the corresponding output bit is 1; otherwise, it is 0.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
OR	Or	FC		

- **Related variables**

Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1 to InN	Operand	FALSE, TRUE	FALSE	Operand
Output	Out	Operation result	FALSE, TRUE	–	Operation result

Data types

	Boolean		Bit String				Integer				Time Format	
	BOOL	SAFE BOOL	BYTE	SAFE BYTE	WORD	SAFE WORD	INT	SAFEINT	DINT	SAFE DINT	TIME	SAFE TIME
In1 to InN	√	√	–	–	–	–	–	–	–	–	–	–
Out	√	√	–	–	–	–	–	–	–	–	–	–

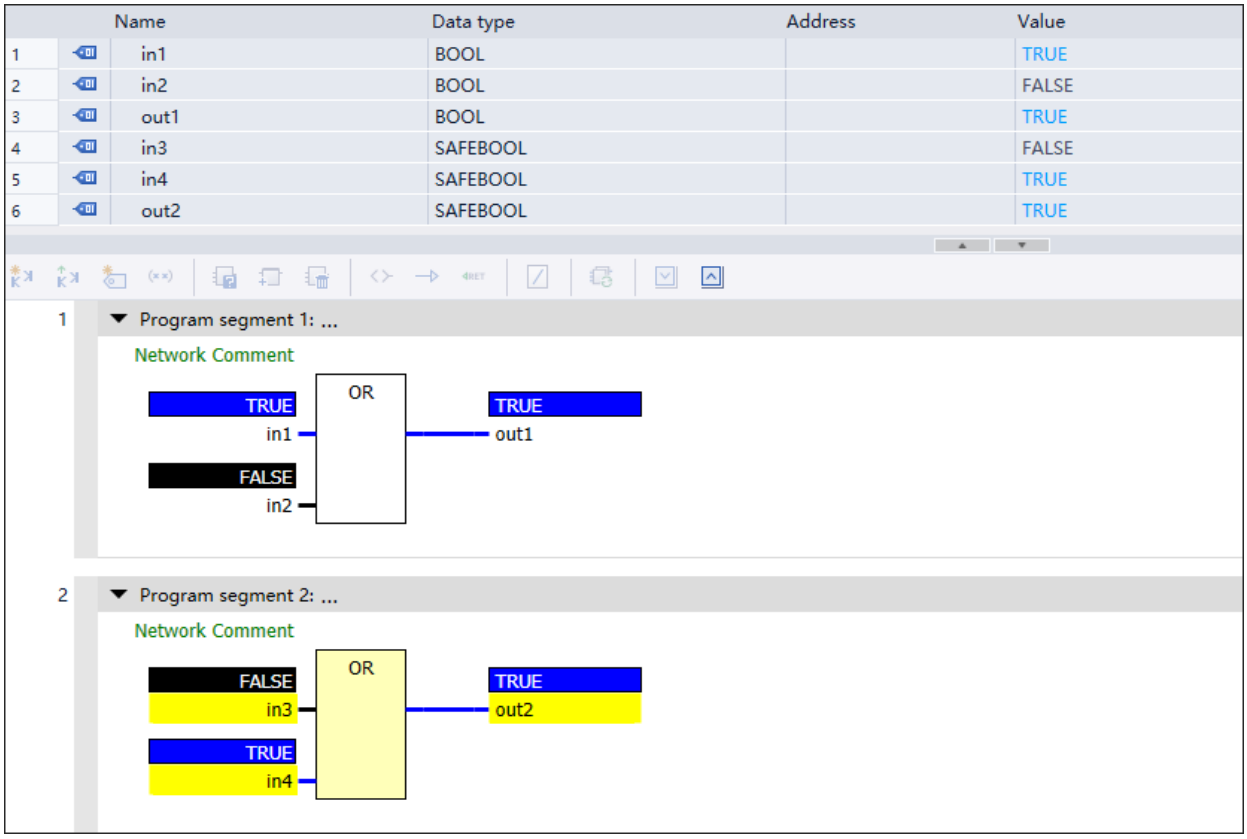
- **Function description**

AND operation is performed on multiple Boolean operands "In1 to InN".

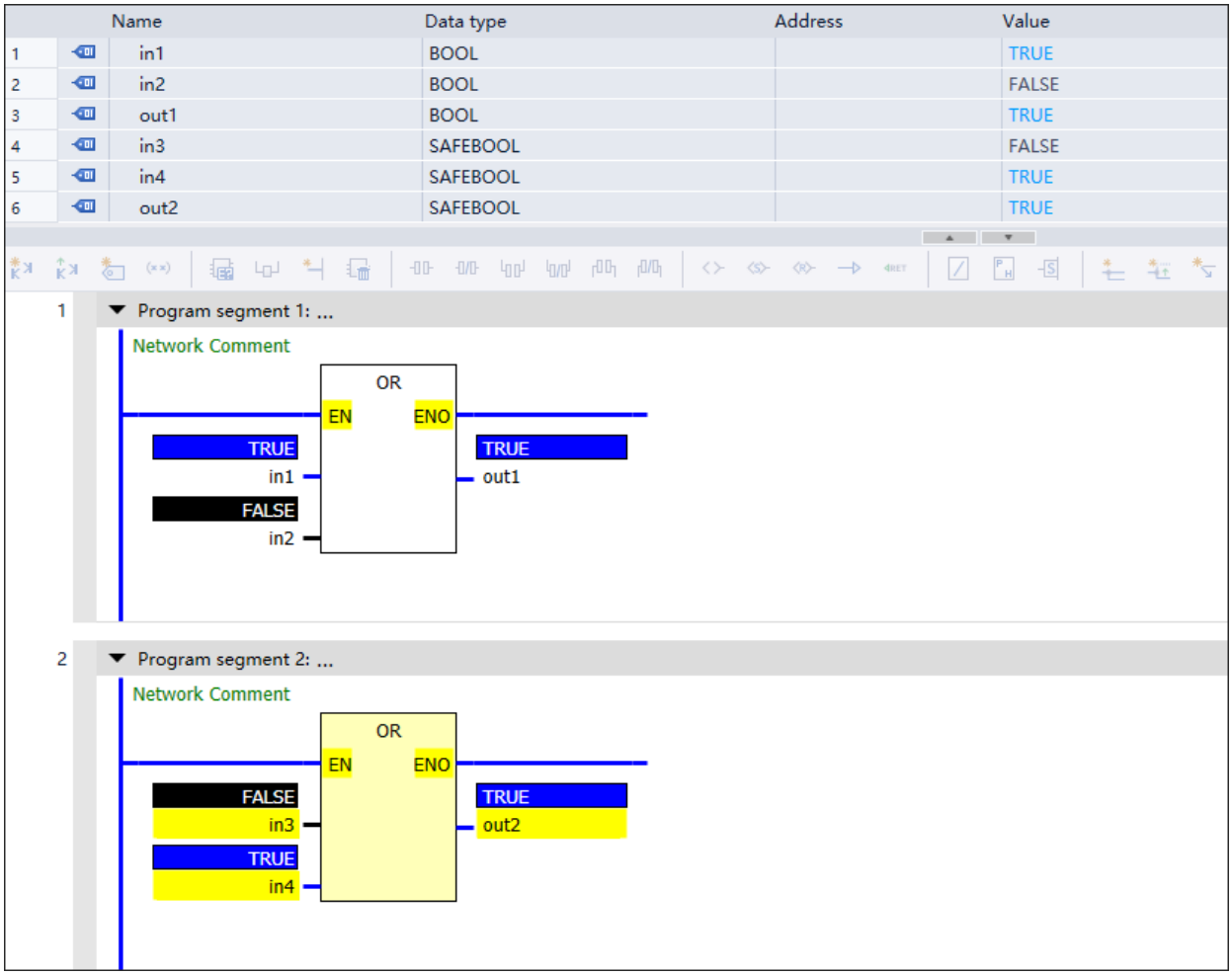
The operation result is FALSE when all input operands are FALSE. Otherwise, the operation result is TRUE.

- **Program example**

FBD:



LD:

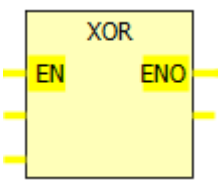
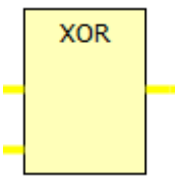


- **Precautions**
  - The data types of "In1" to "InN" and "Out" can only be set to Boolean.
  - The input condition varies depending on whether the output is a safety or standard data type. If the condition is not met, a compilation fault will occur.
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.2.5 XOR

Bitwise XOR: When one input is 1 and the other is 0 for the same bit of the two inputs, the corresponding output bit is 1; otherwise, when the two input values are the same for the same bit, the output bit is 0.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
XOR	Exclusive OR	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1 to InN	Operand	FALSE, TRUE	FALSE	Operand
Output	Out	Operation result	FALSE, TRUE	–	Operation result

Data types

	Boolean		Bit String				Integer				Time Format	
	BOOL	SAFE BOOL	BYTE	SAFE BYTE	WORD	SAFE WORD	INT	SAFEINT	DINT	SAFE DINT	TIME	SAFE TIME
In1 to InN	√	√	–	–	–	–	–	–	–	–	–	–
Out	√	√	–	–	–	–	–	–	–	–	–	–

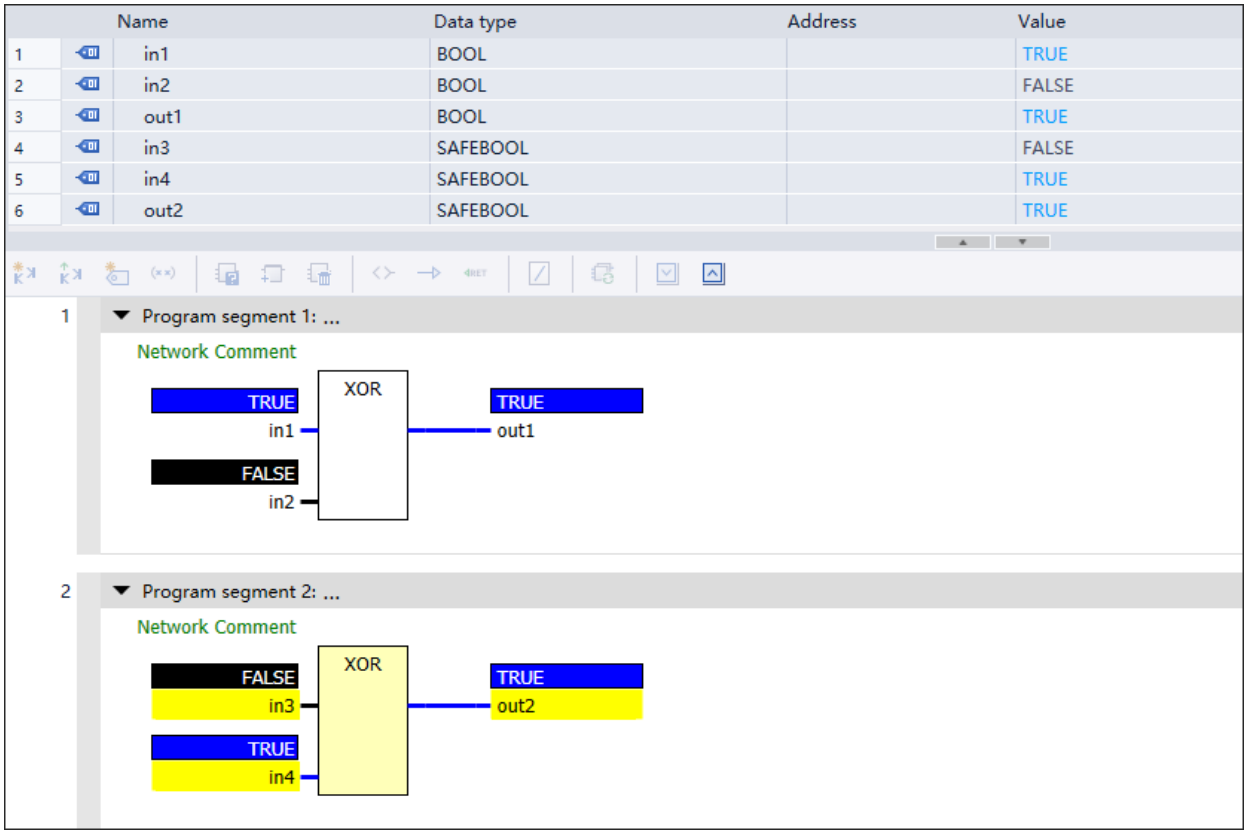
- **Function description**

  AND operation is performed on multiple Boolean operands "In1 to InN".

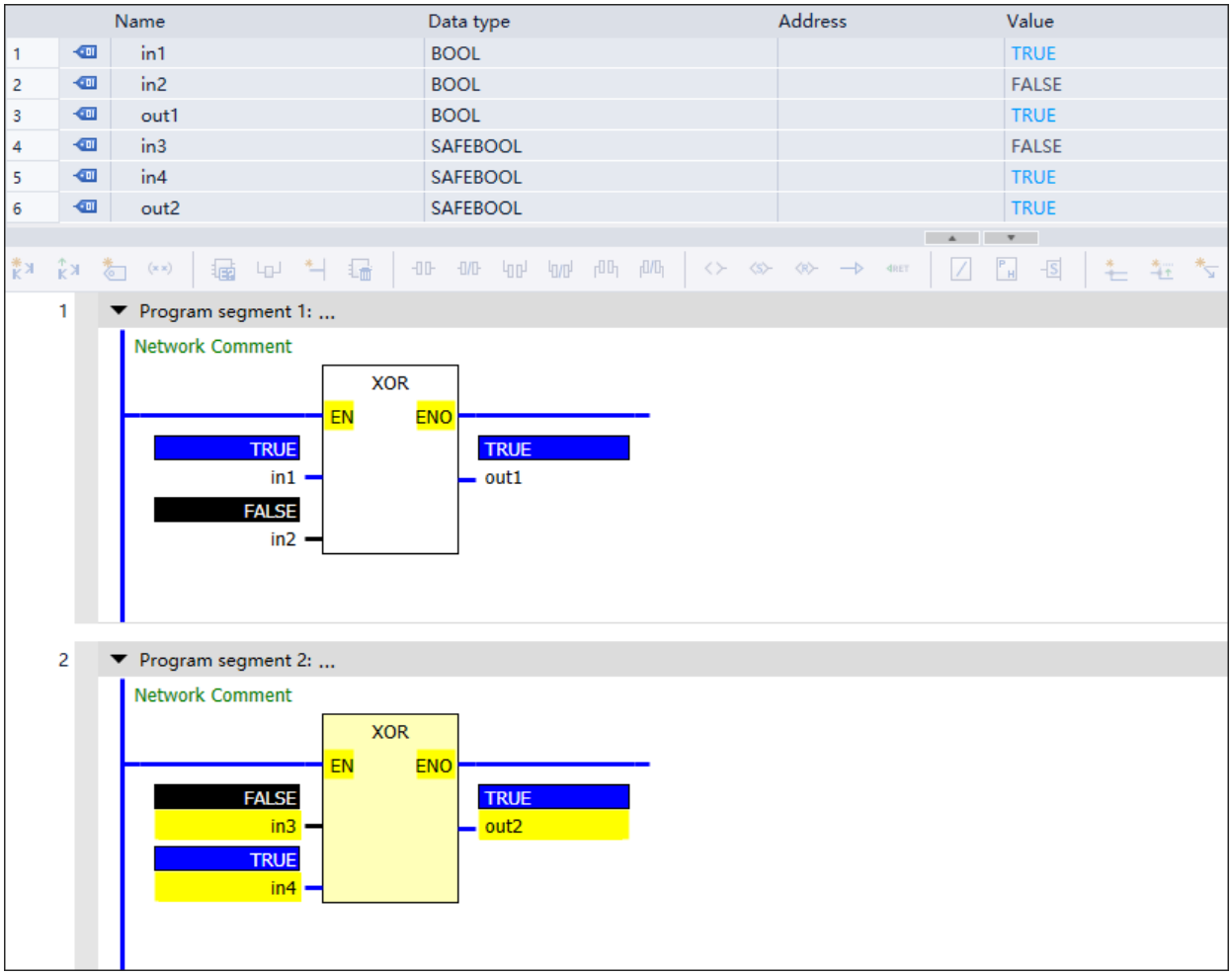
  The operation result is FALSE when the two input operands have the same value. The operation result is TRUE when they are different.

- **Program example**

  FBD:



  LD:



- **Precautions**
  - The data types of "In1" to "InN" and "Out" can only be set to Boolean.
  - The input condition varies depending on whether the output is a safety or standard data type. If the condition is not met, a compilation fault will occur.
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.3 Mathematical Operations

11.10.1.3.1 Instruction List

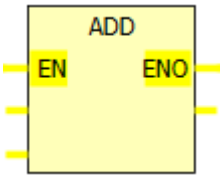
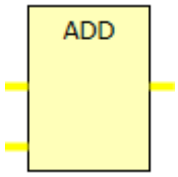
Instruction	FB/FC	Function	Library	Applicable Library Version
11.10.1.3.2 ADD	FC	Add	SafetyInstructions	Applicable to library version 1.0.0.0/1.1.0.0
11.10.1.3.3 SUB	FC	Subtract		
11.10.1.3.4 MUL	FC	Multiply		

Instruction	FB/FC	Function	Library	Applicable Library Version
11.10.1.3.5 DIV	FC	Divide		

11.10.1.3.2 ADD

ADD operation is performed on integers and durations.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
ADD	Add	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1 to InN	Addend	Follows data type	-	Values to be added
Output	Out	Output value	Follows data type	-	Output value

  Data types

	Boolean		Bit String				Integer				Time Format	
	BOOL	SAFE BOOL	BYTE	SAFE BYTE	WORD	SAFE WORD	INT	SAFEINT	DINT	SAFE DINT	TIME	SAFE TIME
In1 to InN	–	–	–	–	–	–	√	√	√	√	√	√
Out	–	–	–	–	–	–	√	√	√	√	√	√

- **Function description**
  - Output the result of adding integers or durations.
  - Overflow occurs when the sum of "In1" to "InN" exceeds the valid range of the data type of the result. When overflow occurs, the relationship among the data types of "In1" to "InN", the data type of the result, and the value of the result is shown in the table below.

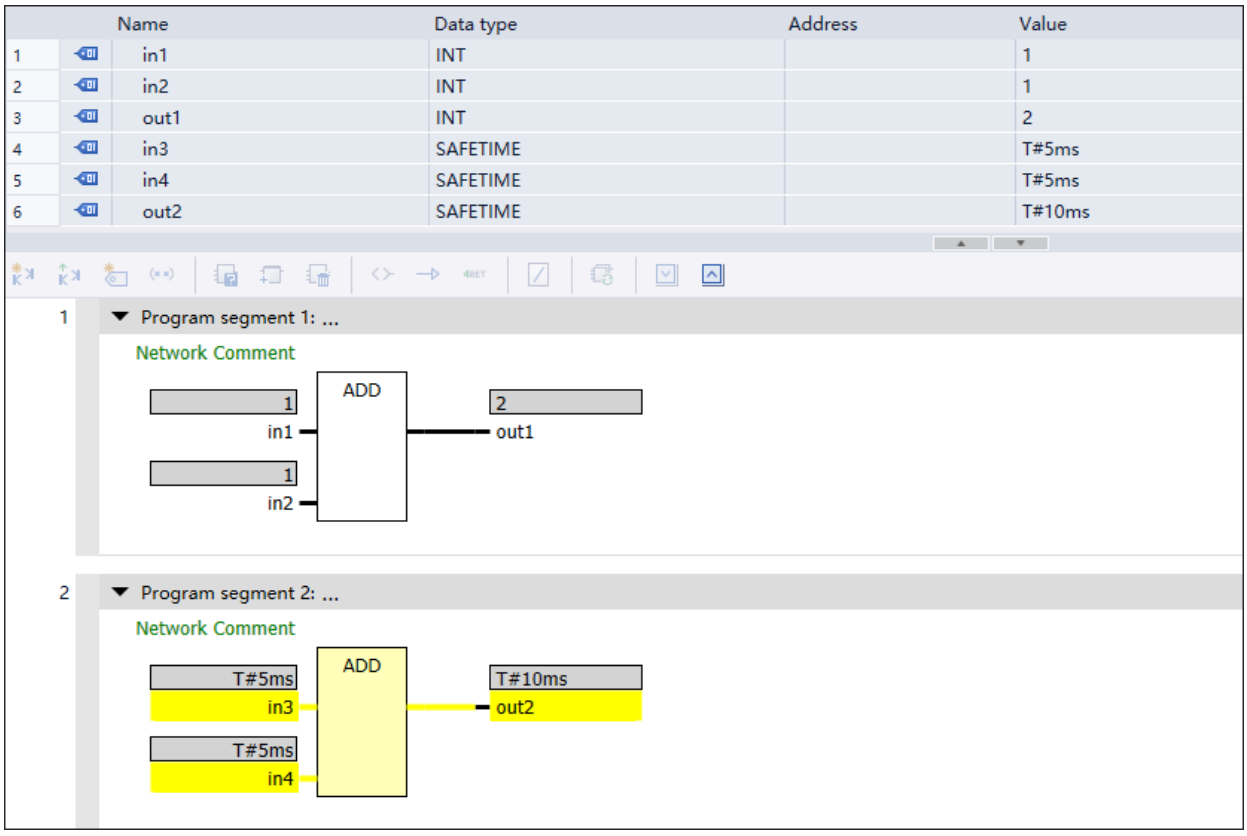
Input/Output Data Type	Addition Result
Integer	The sum of In1 to InN is represented by the data type of the addition result [1]. When both inputs are of the same type, the output must be of the same type, such as INT + INT = INT. When the two inputs are of different types, the output must be the type with the wider bit width, such as INT + DINT = DINT. The result of safety type input and standard type input is a standard type. The result of safety type inputs can be either a standard or safety data type. The result of standard type inputs is a standard type.
Time format	The sum of TIME type values is represented by DWORD. Overflow is truncated according to bit width [2].

[1]: For example, if the value of "In1" is INT#32767 and the value of "In2" is INT#3, the value of the addition result is 32770. When the output is set to INT type, the value of the addition result is -32766, which is the lower 16 bits of the sum 32770. When the output is set to DINT type, the result is DINT#32770.

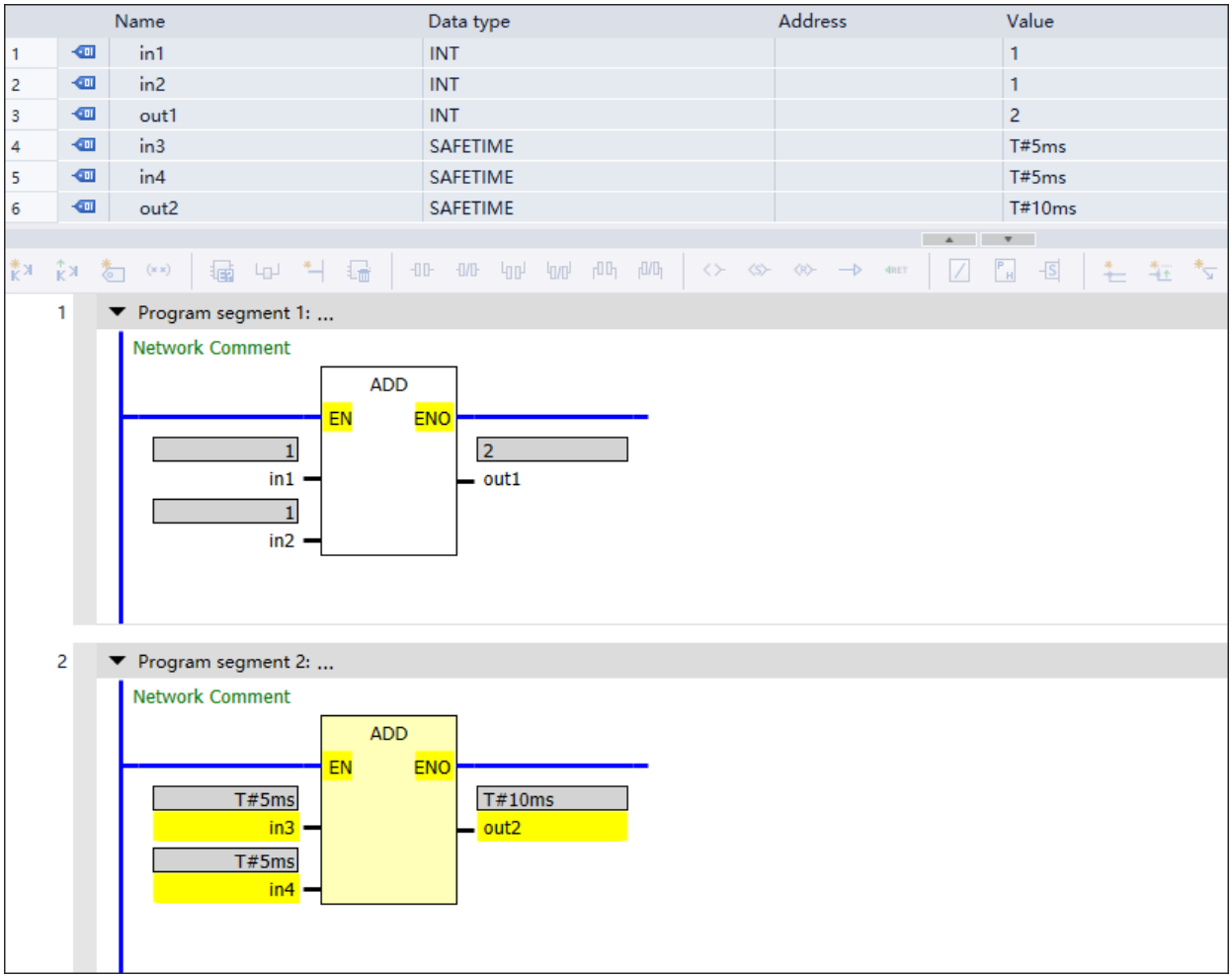
[2]: For example, if the value of "In1" is TIME#49d17h2m47s295ms and the value of "In2" is TIME#5ms, the sum is TIME#49d17h2m47s30ms. However, the maximum value of TIME type is the same as DWORD, which is 4294967295. Therefore, the value of the addition result is TIME#4ms, representable in 32 bits.

- **Programming example**

  FBD:



  LD:

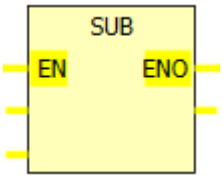
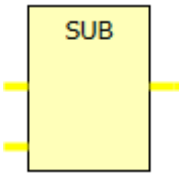


- **Precautions**
  - If the input/output parameter is omitted, a fault will occur during compilation. However, if the output terminal is unused or the input terminal for the next instruction is not connected, no fault will occur.
  - "In0" to "InN", and "Out" can be of different data types. In this case, the operation is processed using the data type that includes all the above input parameters. For example, if "In0" is INT type and "In1" is DINT type, the operation is processed using DINT type. Therefore, the addition result "Out" is of DINT type.
  - The input condition varies depending on whether the output is a safety or standard data type. If the condition is not met, a compilation fault will occur.
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.3.3 SUB

SUB operation is performed on integers and durations.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SUB	Subtract	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Subtrahend	Follows data type	–	Subtrahend
	In2	Minuend	Follows data type	–	Minuend
Output	Out	Output value	Follows data type	–	Output value

  Data types

	Boolean		Bit String				Integer				Time Format	
	BOOL	SAFE BOOL	BYTE	SAFE BYTE	WORD	SAFE WORD	INT	SAFEINT	DINT	SAFE DINT	TIME	SAFE TIME
In1	–	–	–	–	–	–	√	√	√	√	√	√
In2	–	–	–	–	–	–	√	√	√	√	√	√
Out	–	–	–	–	–	–	√	√	√	√	√	√

- **Function description**
  - Output the result of subtracting integers or durations.
  - Overflow occurs when the subtraction result of "In1" and "In2" exceeds the valid range of the data type of the result. When overflow occurs, the relationship among the data types of "In1" and "In2", the data type of the result, and the value of the result is shown in the table below.

Input/Output Data Type	Subtraction Result
Integer	The value difference between In1 and In2 is represented by the data type of the subtraction result [1]. When both inputs are of the same type, the output must be of the same type, such as INT - INT = INT. When the two inputs are of different types, the output must be the type with the wider bit width, such as INT - DINT = DINT. The result of safety type input and standard type input is a standard type. The result of safety type inputs can be either a standard or safety data type. The result of standard type inputs is a standard type.

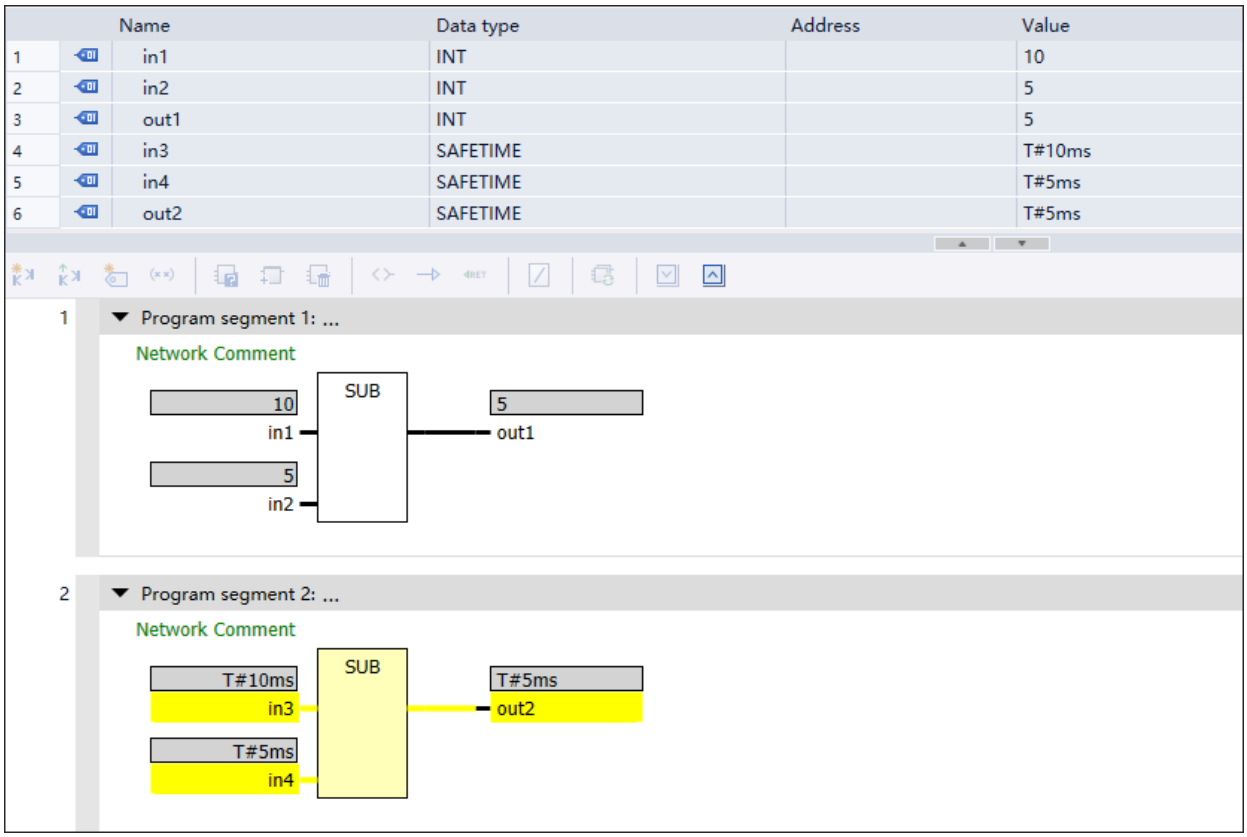
Input/Output Data Type	Subtraction Result
Time Format	The subtraction of TIME type values is represented by DWORD. Overflow is truncated according to bit width [2].

[1]: For example, if the value of "In1" is INT#-5 and the value of "In2" is INT#32767, the value of the addition result is -32772. When the output is set to INT type, the value of the subtraction result is 32764, which is the lower 16 bits of -32772. When the output is set to DINT type, the result is DINT#-32772.

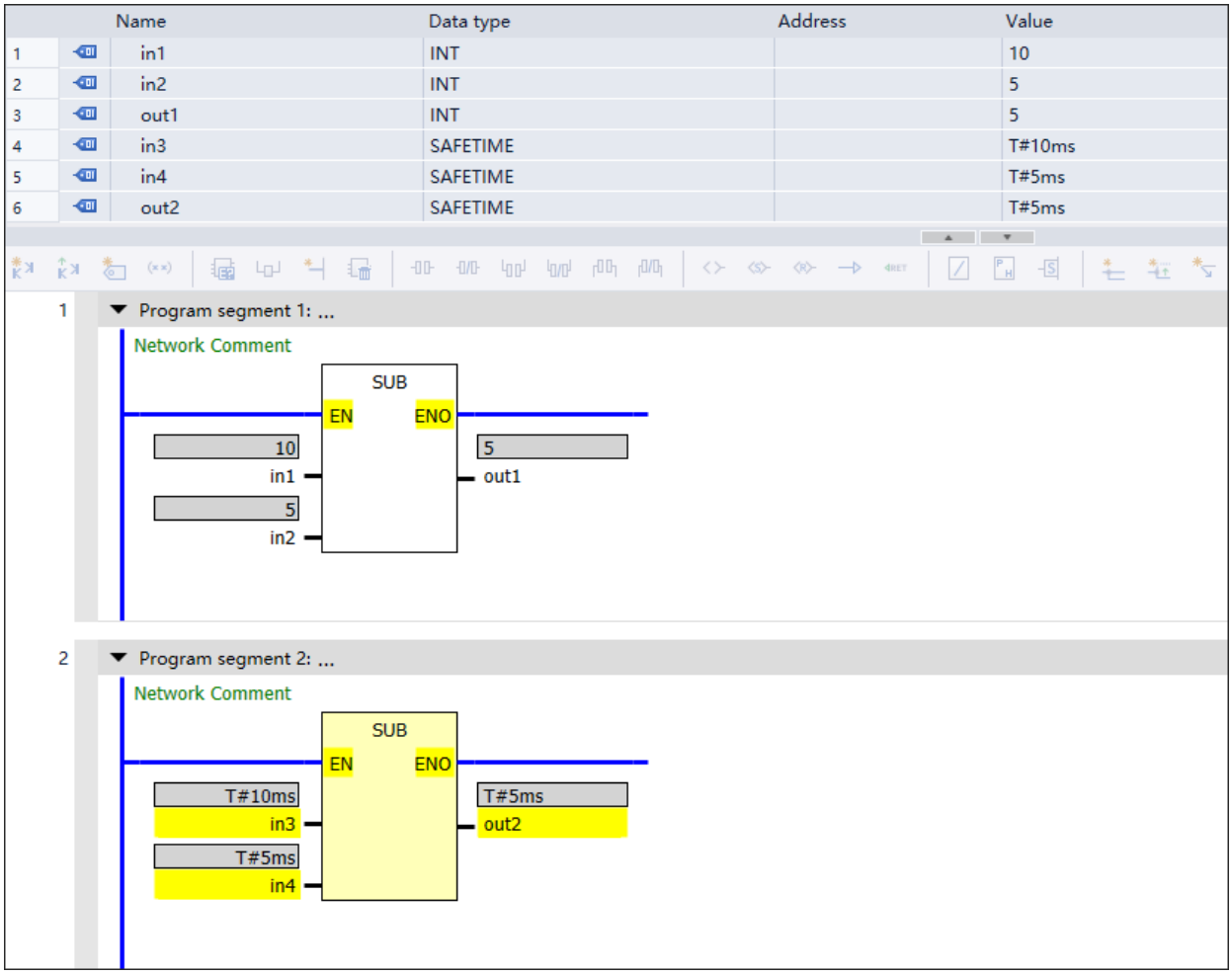
[2]: For example, when the value of "In1" is TIME#10ms and the value of "In2" is TIME#14ms, the value of the subtraction result is TIME#-4ms (although negative time does not exist, negative values are used for representation). However, since the maximum value of TIME is the same as DWORD, which is 4294967295, the subtraction result is T#49d17h2m47s292ms, representable within 32 bits.

- **Programming example**

  FBD:



  LD:

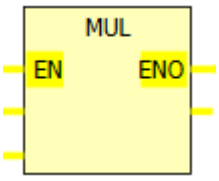
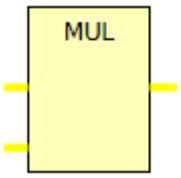


- **Precautions**
  - If the input/output parameter is omitted, a fault will occur during compilation. However, if the output terminal is unused or the input terminal for the next instruction is not connected, no fault will occur.
  - "In1", "In2", and "Out" can be of different data types. In this case, the operation is processed using the data type that includes all the above input parameters. For example, if "In0" is INT type and "In1" is DINT type, the operation is processed using DINT type. Therefore, the subtraction result "Out" is of DINT type.
  - The input condition varies depending on whether the output is a safety or standard data type. If the condition is not met, a compilation fault will occur.
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.3.4 MUL

MUL operation is performed on integers and durations.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
MUL	Multiply	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1 to InN	Multiplier	Follows data type	–	Multiplier
Output	Out	Output value	Follows data type	–	Output value

  Data types

	Boolean		Bit String				Integer				Time Format	
	BOOL	SAFE BOOL	BYTE	SAFE BYTE	WORD	SAFE WORD	INT	SAFEINT	DINT	SAFE DINT	TIME	SAFE TIME
In1	–	–	–	–	–	–	√	√	√	√	√	√
In2 to InN	–	–	–	–	–	–	√	√	√	√	–	–
Out	–	–	–	–	–	–	√	√	√	√	√	√

- **Function description**
  - Output the result of multiplying integers or durations.
  - Overflow occurs when the product of "In1" to "InN" exceeds the valid range of the data type of the result. When overflow occurs, the relationship among the data types of "In1" to "InN", the data type of the result, and the value of the result is shown in the table below.

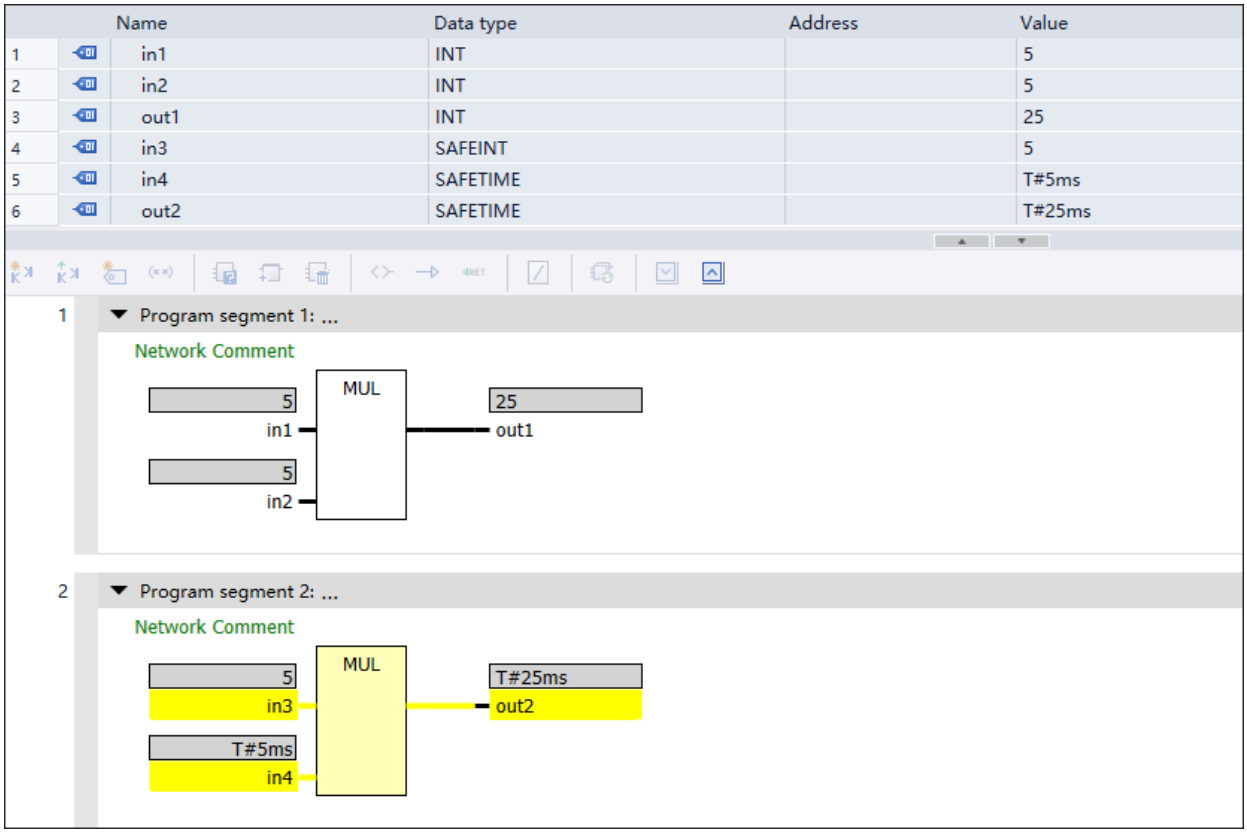
Input/Output Data Type	Product Result
Integer	The product of In1 to InN is represented by the data type of the product result [1]. When both inputs are of the same type, the output must be of the same type, such as INT × INT = INT. When the two inputs are of different types, the output must be the type with the wider bit width, such as INT × DINT = DINT. The result of safety type input and standard type input is a standard type. The result of safety type inputs can be either a standard or safety data type. The result of standard type inputs is a standard type.
Time Format	The product of TIME type values is represented by DWORD. Overflow is truncated according to bit width [2].

[1]: For example, if the value of "In1" is INT#16390 and the value of "In2" is INT#2, the product value is 32780. When the output is set to INT type, the product value is -32756, which is the lower 16 bits of 32780. When the output is set to DINT type, the result is DINT#32780.

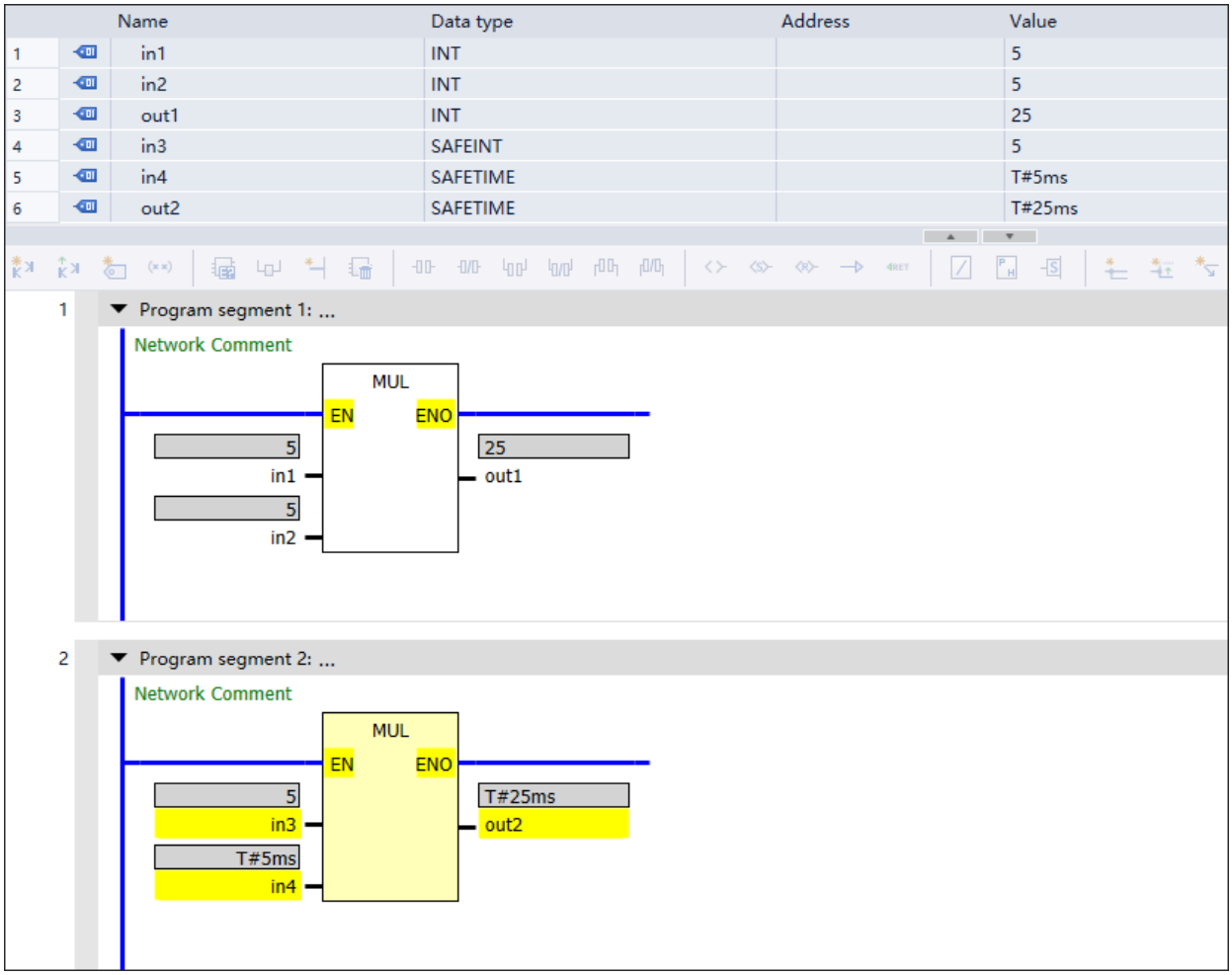
[2]: For example, if the value of "In1" is T#24d20h31m23s649ms and the value of "In2" is INT#2, the product result is T#49d17h2m47s298ms. However, the maximum value of TIME type is the same as DWORD, which is 4294967295. Therefore, the product result is T#2ms, representable in 32 bits.

- **Programming example**

  FBD:



  LD:

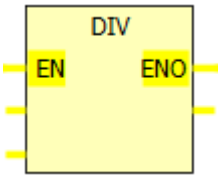
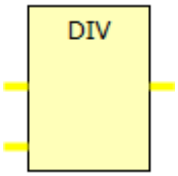


- **Precautions**
  - If the input/output parameter is omitted, a fault will occur during compilation. However, if the output terminal is unused or the input terminal for the next instruction is not connected, no fault will occur.
  - MUL operation cannot be performed on multiple time format data. When multiplying time format data, set the time format for "In1" and "Out", and set integer type for "In2" to "InN".
  - When multiplying integer types, "In1" to "InN" and "Out" can be of different data types. In this case, the operation is processed using the data type that includes all the above input parameters. For example, if "In1" is INT type and "In2" is DINT type, the operation is processed using DINT type. Therefore, the product result "Out" is of DINT type.
  - When using time format for the multiplier, set the time format for any one of "In1" to "InN" and for "Out".
  - The input condition varies depending on whether the output is a safety or standard data type. If the condition is not met, a compilation fault will occur.
  - When assigning a safety data type variable to the output terminal, set all input terminals to safety data type variables; otherwise, a compilation fault will occur.

11.10.1.3.5 DIV

DIV operation is performed on integers and durations.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
DIV	Divide	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Dividend	Follows data type	–	Dividend
	In2	Divider	Follows data type	–	Divider
Output	Out	Output value	Follows data type	–	Output value

  Data types

	Boolean		Bit String				Integer				Time Format	
	BOOL	SAFE BOOL	BYTE	SAFE BYTE	WORD	SAFE WORD	INT	SAFEINT	DINT	SAFE DINT	TIME	SAFE TIME
In1	–	–	–	–	–	–	√	√	√	√	√	√
In2	–	–	–	–	–	–	√	√	√	√	–	–
Out	–	–	–	–	–	–	√	√	√	√	√	√

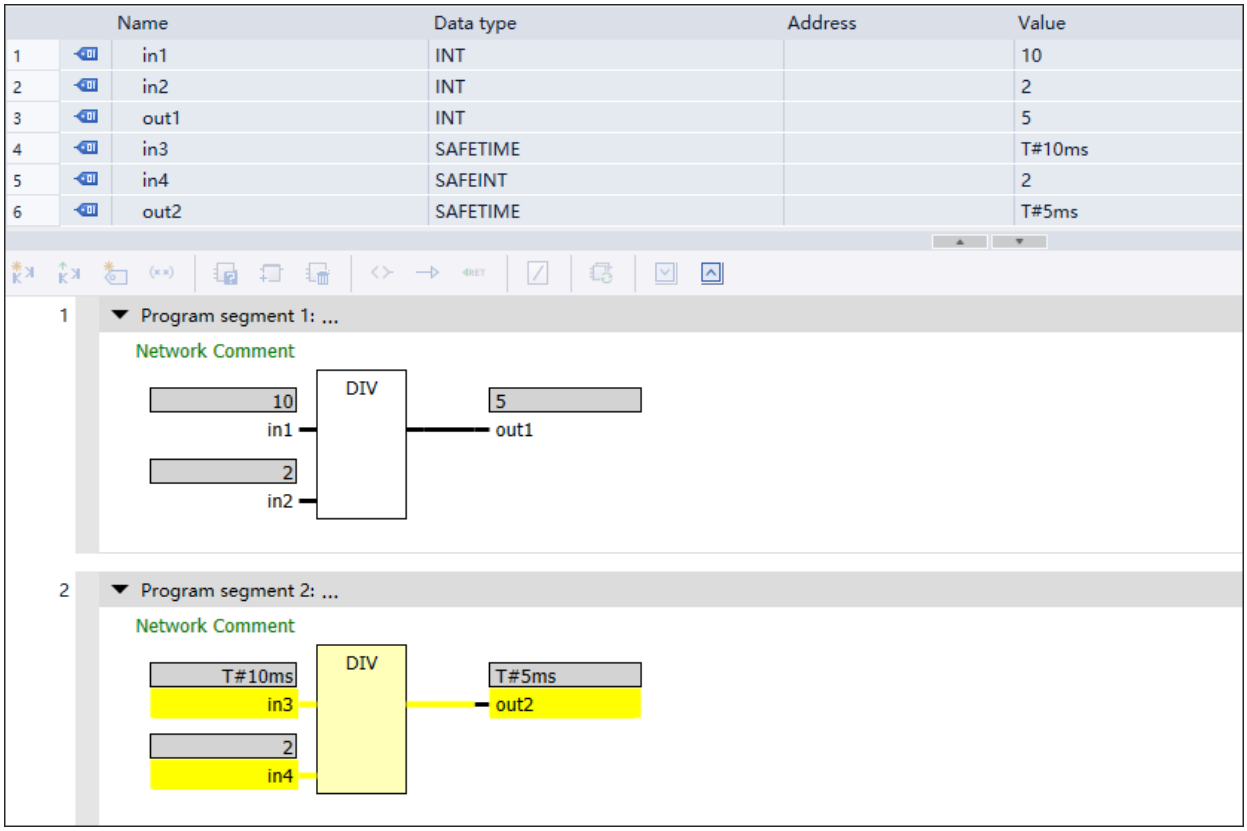
- **Function description**
  - Output the result of dividing integers or durations.
  - When division is not exact, the remainder is discarded.
  - Overflow occurs when the division result of "In1" and "In2" exceeds the valid range of the data type of the result. When overflow occurs, the relationship among the data types of "In1" and "In2", the data type of the result, and the value of the result is shown in the table below.

Input/Output Data Type	Division Result
Integer	The quotient of In1 and In2 is represented by the data type of the division result. When both inputs are of the same type, the output must be of the same type, such as INT ÷ INT = INT. When the two inputs are of different types, the output must be the type with the wider bit width, such as INT ÷ DINT = DINT. The result of safety type input and standard type input is a standard type. The result of safety type inputs can be either a standard or safety data type. The result of standard type inputs is a standard type.

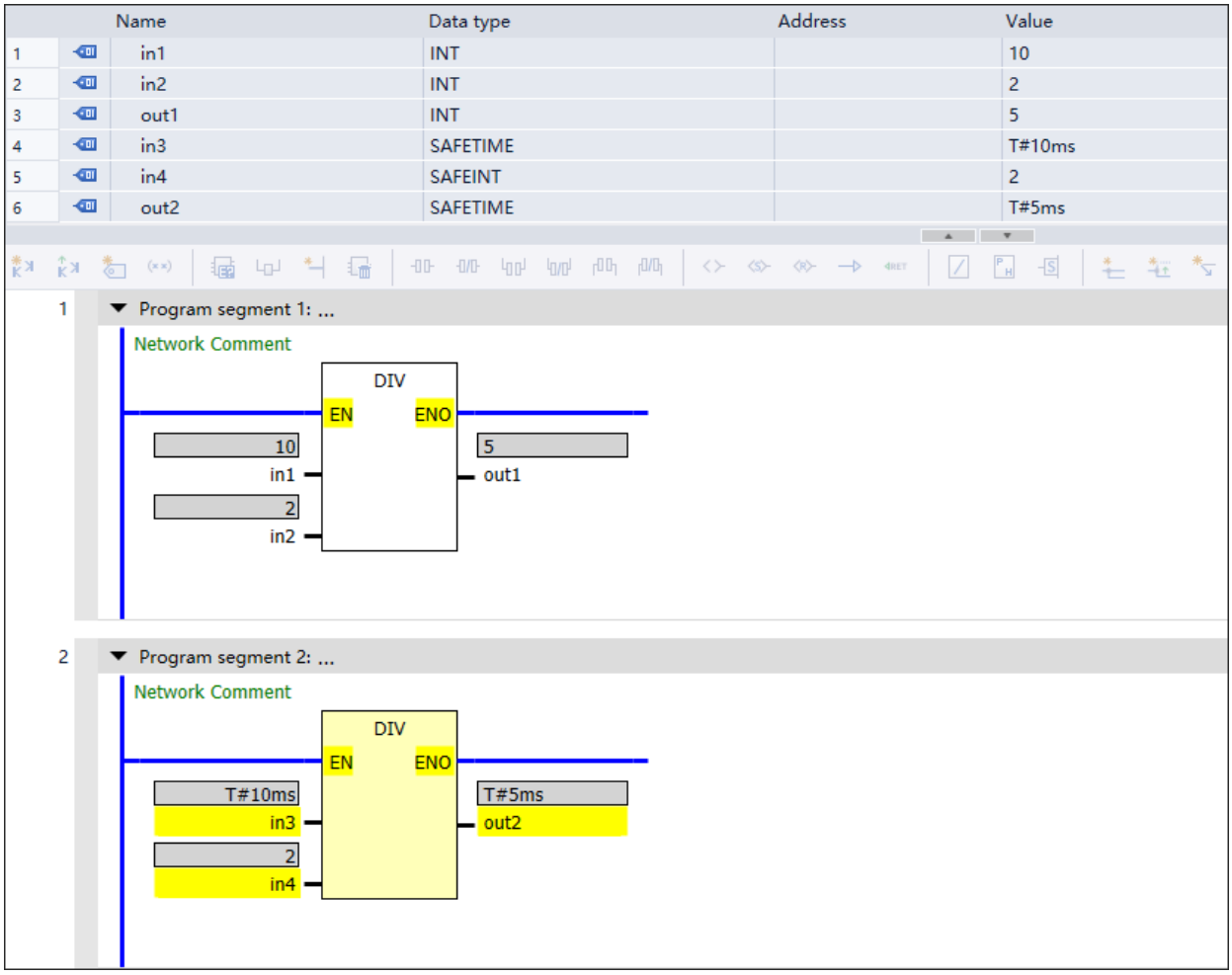
Input/Output Data Type	Division Result
Time Format	The quotient of TIME type is represented by DWORD, and overflow is truncated according to bit width.

- **Program example**

FBD:



LD:



- **Precautions**
  - If the input/output parameter is omitted, a fault will occur during compilation. However, if the output terminal is unused or the input terminal for the next instruction is not connected, no fault will occur.
  - MUL operation cannot be performed on multiple time format data. When dividing time format data, set the time format for "In1" and "Out", and set integer type for "In2" to "InN".
  - The divider must not be 0. Using 0 as the divider causes a "division by zero" exception, and the program will stop.
  - When dividing integer types, "In1" to "InN" and "Out" can be of different data types. In this case, the operation is processed using the data type that includes all the above input parameters. For example, if "In1" is INT type and "In2" is DINT type, the operation is processed using DINT type. Therefore, the division result "Out" is of DINT type.
  - The input condition varies depending on whether the output is a safety or standard data type. If the condition is not met, a compilation fault will occur.
  - When assigning a safety data type variable to the output terminal, set all input terminals to safety data type variables; otherwise, a compilation fault will occur.

11.10.1.4 Selection Operations

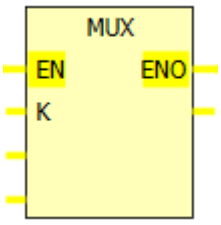
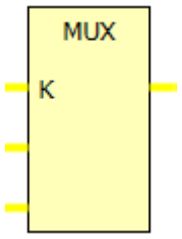
11.10.1.4.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
11.10.1.4.2 MUX	FC	Select one from multiple	SafetyInstructions	Applicable to library version 1.0.0.0/1.1.0.0
11.10.1.4.3 SEL	FC	One out of two		

11.10.1.4.2 MUX

MUX operation selects one from multiple options.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
MUX	Select one from multiple	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	K	Selector	Follows data type	–	If K's value is N, select InN.
	In0 to InN	Options	Follows data type	–	Option
Output	Out	Selection result	Follows data type	–	Selection result

  Data types

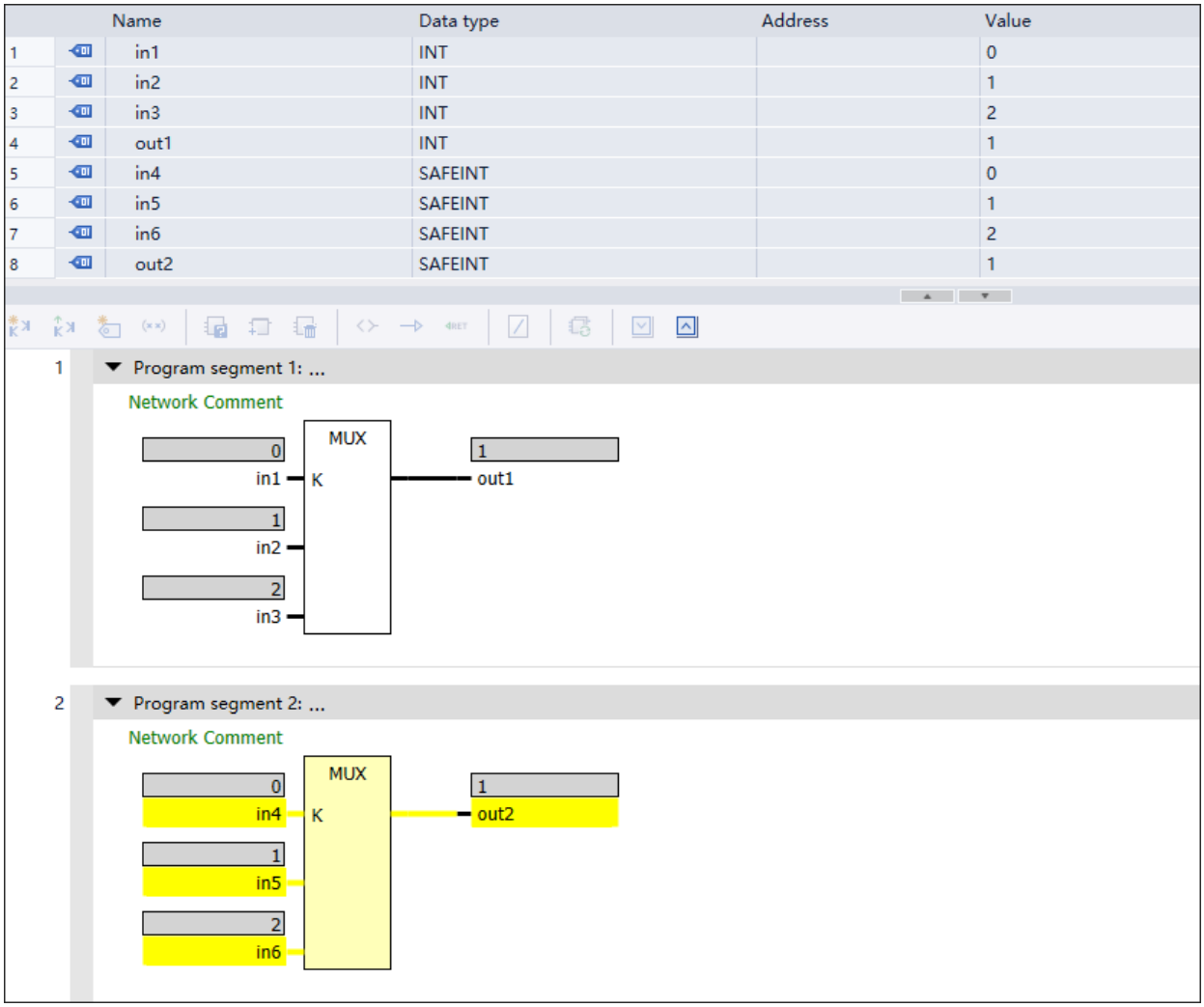
	Boolean		Bit String				Integer				Time Format	
	BOOL	SAFE BOOL	BYTE	SAFE BYTE	WORD	SAFE WORD	INT	SAFEINT	DINT	SAFE DINT	TIME	SAFETIME
K	–	–	–	–	–	–	√	√	√	√	–	–
In0 to InN	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	√	√	√	√	√	√	√	√	√	√	√

- **Function description**
  - Specify one from the two to N options ("In0" to "InN").

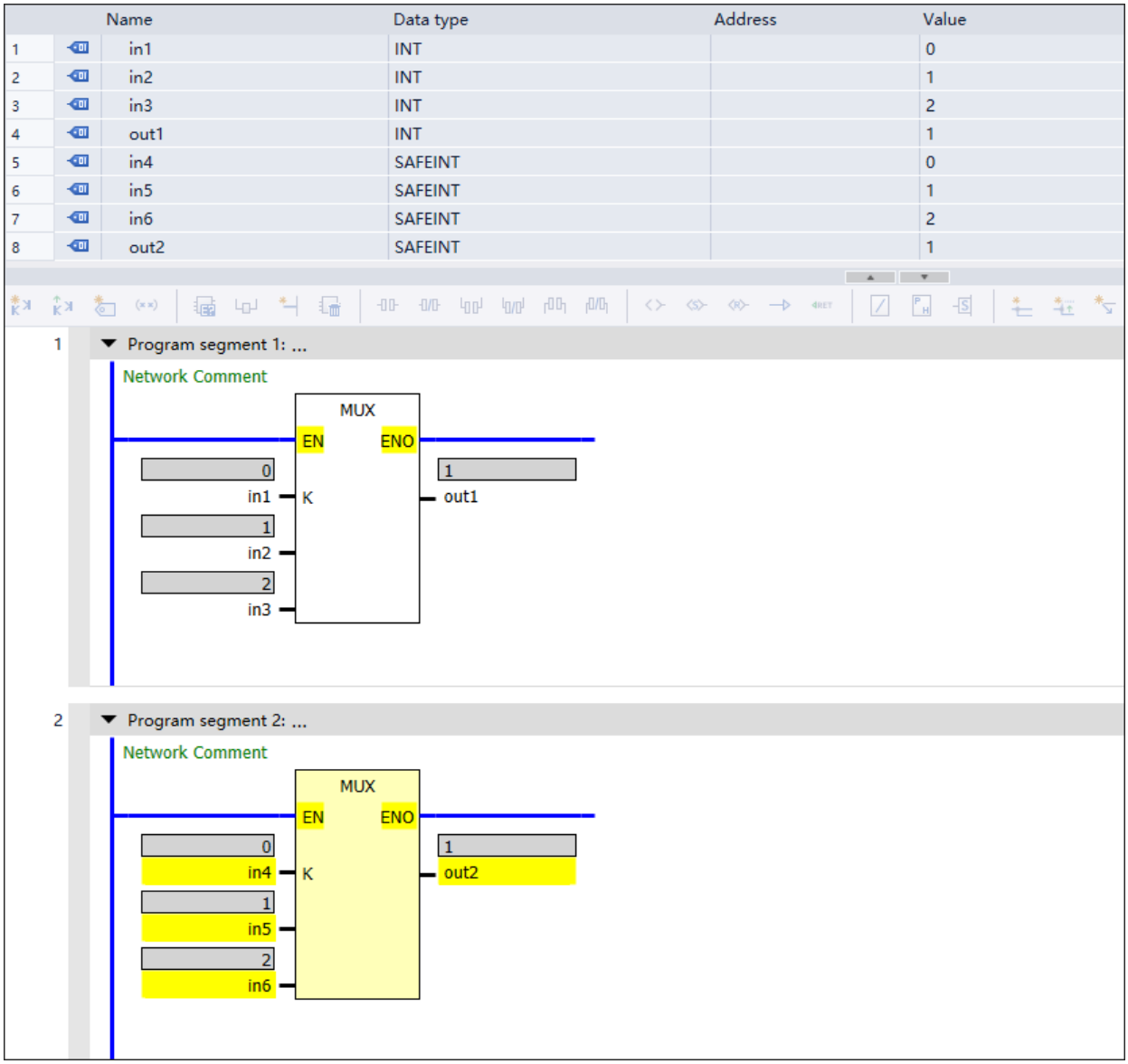
- When selecting one from "In0" to "InN", specify it using the selector "K".
- When the value of "K" is 0, assign "In0" to "Out"; when it is N, assign "InN" to "Out".

- **Programming example**

  FBD:



  LD:

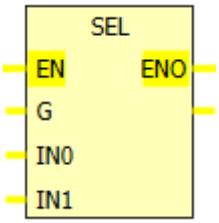
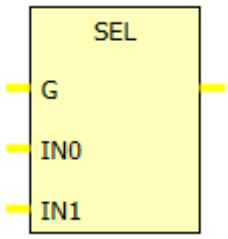


- **Precautions**
  - If the input/output parameter is omitted, a fault will occur during compilation. However, if the output terminal is unused or the input terminal for the next instruction is not connected, no fault will occur.
  - "In0", "In1", and "Out" can be of different data types, but note the following:
    - The valid range of "Out" must include the valid ranges of "In0" to "InN".
    - If the value of "K" is outside the valid range (less than 0 or greater than N), a "MUX exception" will occur and the program will stop.
    - If the output terminal of MUX is of safety type, all input parameters corresponding to the instruction must also be of safety type.

11.10.1.4.3 SEL

SEL operation selects one from two options.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SEL	One out of two	FC		

- **Related variables**

  Variables

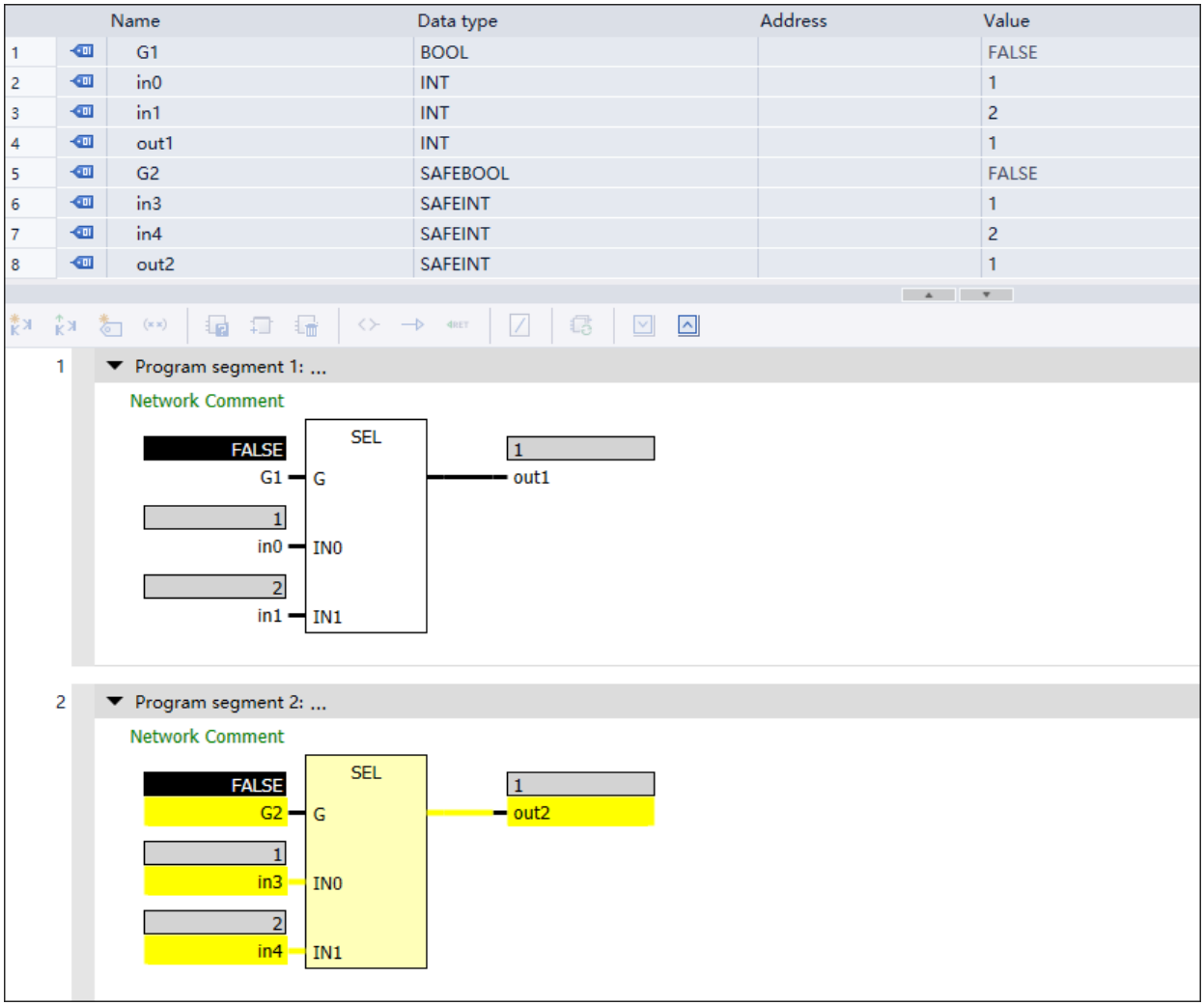
Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	G	Selector gate	Follows data type	–	FALSE: Selects In0. TRUE: Selects In1.
	In0	Option	Follows data type	–	Option
	In1	Option	Follows data type	–	Option
Output	Out	Selection result	Follows data type	–	Selection result

  Data types

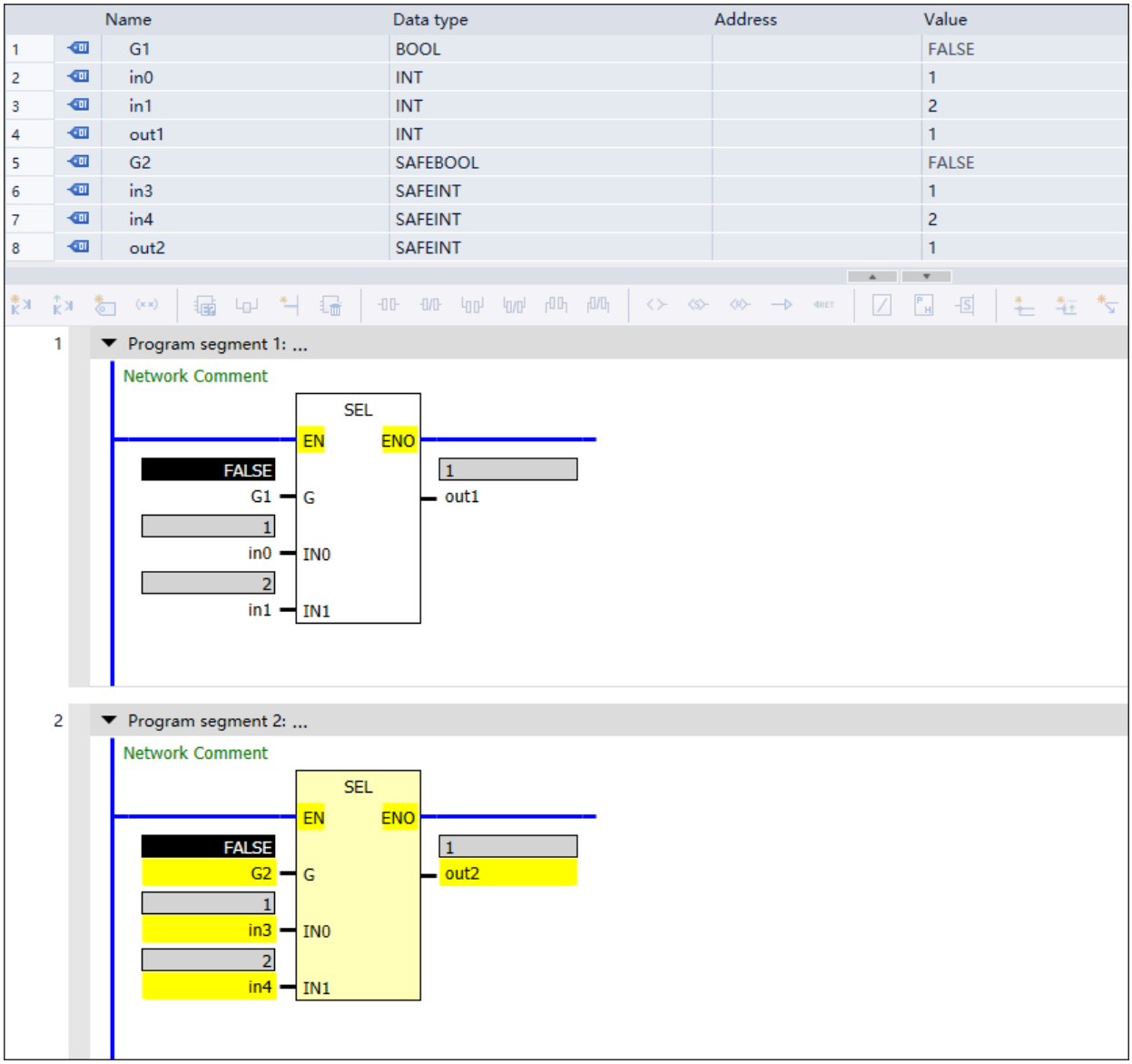
	Boolean		Bit String				Integer				Time Format	
	BOOL	SAFE BOOL	BYTE	SAFE BYTE	WORD	SAFE WORD	INT	SAFEINT	DINT	SAFE DINT	TIME	SAFE TIME
G	√	√	–	–	–	–	–	–	–	–	–	–
In0	√	√	√	√	√	√	√	√	√	√	√	√
In1	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	√	√	√	√	√	√	√	√	√	√	√

- **Function description**
  - Specify one from the two options "In0" and "In1".
  - When selecting one from "In0" and "In1", specify through the selector gate "G".
  - When the value of "G" is FALSE, assign "In0" to "Out"; when it is TRUE, assign "In1" to "Out".
- **Programming example**

  FBD:


Name
Data type
Address
Value
1 G1 BOOL FALSE
2 in0 INT 1
3 in1 INT 2
4 out1 INT 1
5 G2 SAFEBOOL FALSE
6 in3 SAFEINT 1
7 in4 SAFEINT 2
8 out2 SAFEINT 1
1
Program segment 1: ...
Network Comment
FALSE
G1
G
SEL
1
out1
1
in0
IN0
2
in1
IN1
2
Program segment 2: ...
Network Comment
FALSE
G2
G
SEL
1
out2
1
in3
IN0
2
in4
IN1

LD:



- **Precautions**
  - If the input/output parameter is omitted, a fault will occur during compilation. However, if the output terminal is unused or the input terminal for the next instruction is not connected, no fault will occur.
  - "In0", "In1", and "Out" can be of different data types. In this case, the operation is processed using the data type that includes all the above input parameters. For example, if "In0" is INT type and "In1" is DINT type, the operation is processed using DINT type. Therefore, the division result "Out" is of DINT type.
  - The input condition varies depending on whether the output is a safety or standard data type. If the condition is not met, a compilation fault will occur.
  - When input parameter "G" is of a safety data type, the output variable type can be either safety or standard data type; however, when "G" is of a standard data type, the output parameter can only be of standard data type.
  - If the output terminal of SEL is of safety type, all input parameters corresponding to the instruction must also be of safety type.

11.10.1.5 Shift Operations

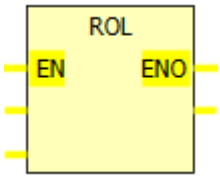
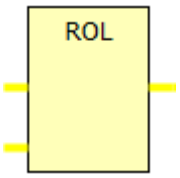
11.10.1.5.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
11.10.1.5.2 ROLE	FC	Cyclic left-shift	SafetyInstructions	Applicable to library version 1.0.0.0/1.1.0.0
11.10.1.5.3 ROR	FC	Cyclic right-shift		
11.10.1.5.4 SHL	FC	Left shift		
11.10.1.5.5 SHR	FC	Right shift		

11.10.1.5.2 ROL

ROL operation performs a bitwise cyclic left shift on the inputs.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
ROLE	Cyclic left-shift	FC		

- **Related variables**

Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN1	Data Source	Follows data type	–	Data source
	NUM	Number of shifted bits	Follows data type	–	Number of shifted bits
Output	Out	Return value	Follows data type	–	Shifted result

Data types

	Boolean		Bit String				Integer				Time Format	
	BOOL	SAFE BOOL	BYTE	SAFE BYTE	WORD	SAFE WORD	INT	SAFEINT	DINT	SAFE DINT	TIME	SAFE TIME
IN1	–	–	√	√	√	√	–	–	–	–	–	–
NUM	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	√	√	√	√	–	–	–	–	–	–

- **Function**

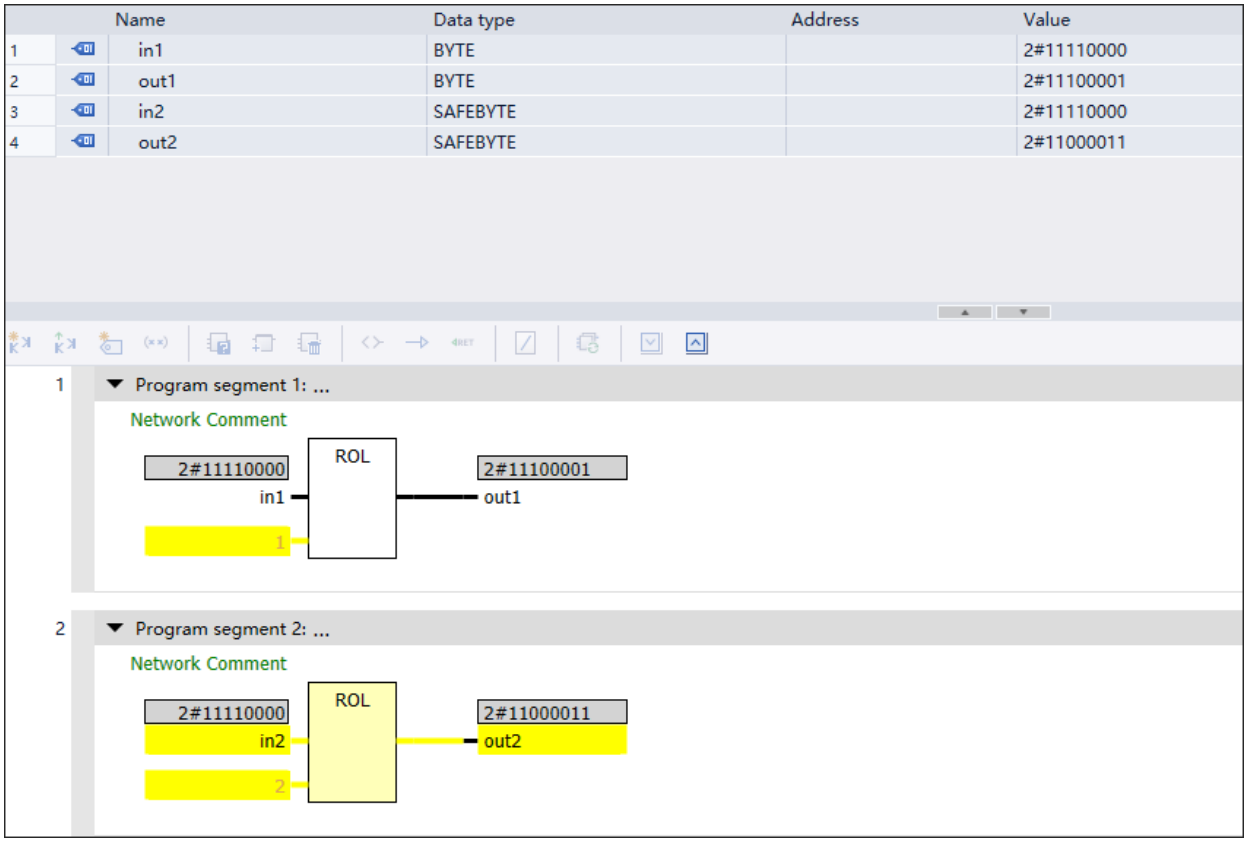
  Rotates the input value to the left, with the bits shifted out from the left directly appended to the rightmost bit.

- **Program example**

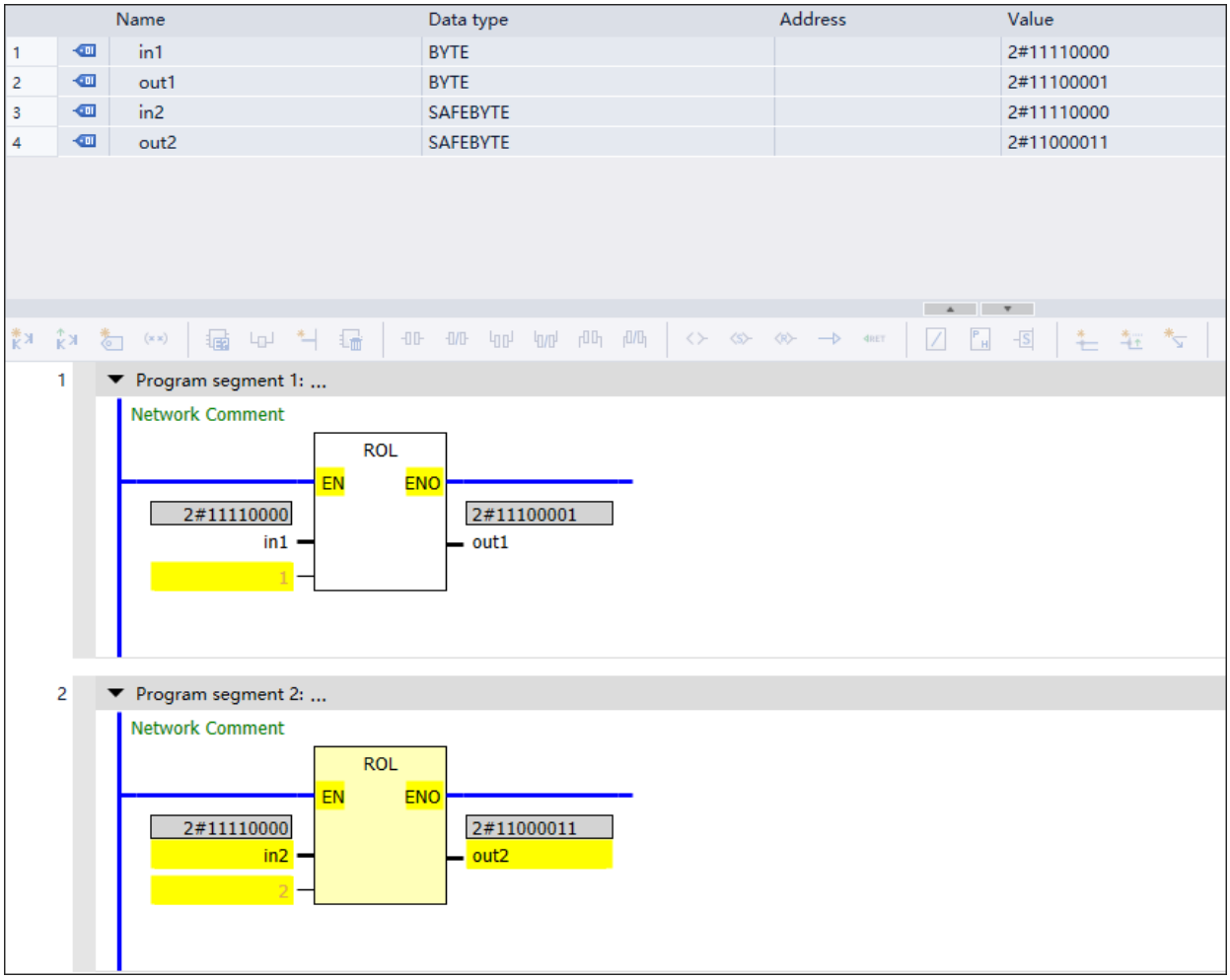
  When the source data is 2#11110000 and the number of shifted bits is 1, the output result is 2#11100001, with the valid data shifted left by one bit.

  When the source data is 2#11110000 and the number of shifted bits is 2, the output signal result is 2#11000011, with the valid data shifted left by two bits.

  FBD:



  LD:

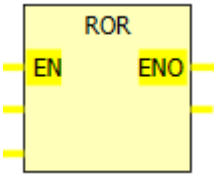
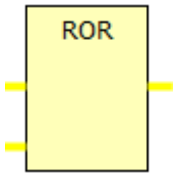


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - NUM only supports immediate values.

11.10.1.5.3 ROR

ROR operation performs a bitwise cyclic right shift on the inputs.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
ROR	Cyclic right-shift	FC		

- **Related variables**

Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN1	Data Source	Follows data type	–	Data source

Input/Output	Variable	Name	Valid Range	Default Value	Description
	NUM	Number of shifted bits	Follows data type	–	Number of shifted bits
Output	Out	Return value	Follows data type	–	Shifted result

Data types

	Boolean		Bit String				Integer				Time Format	
	BOOL	SAFE BOOL	BYTE	SAFE BYTE	WORD	SAFE WORD	INT	SAFEI NT	DINT	SAFE DINT	TIME	SAFE TIME
IN1	–	–	√	√	√	√	–	–	–	–	–	–
NUM	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	√	√	√	√	–	–	–	–	–	–

- **Function description**

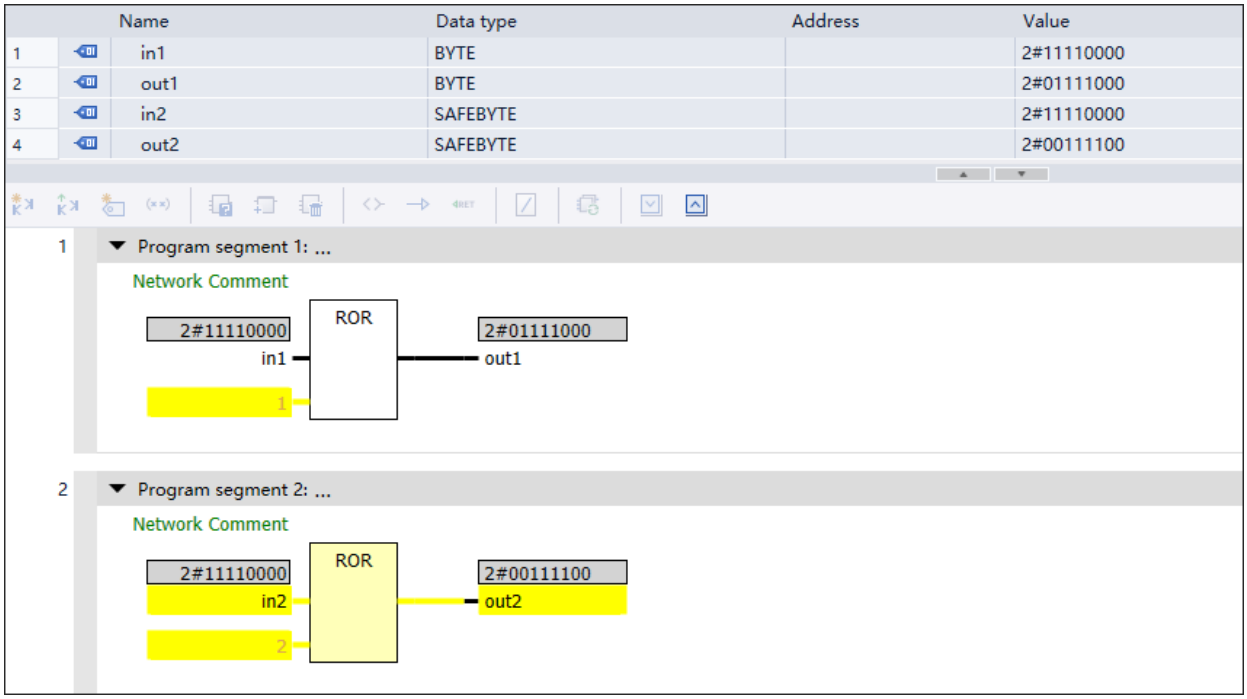
  Perform a bitwise cyclic right shift on the input value, with the bit shifted out from the right directly appended to the leftmost bit.

- **Program example**

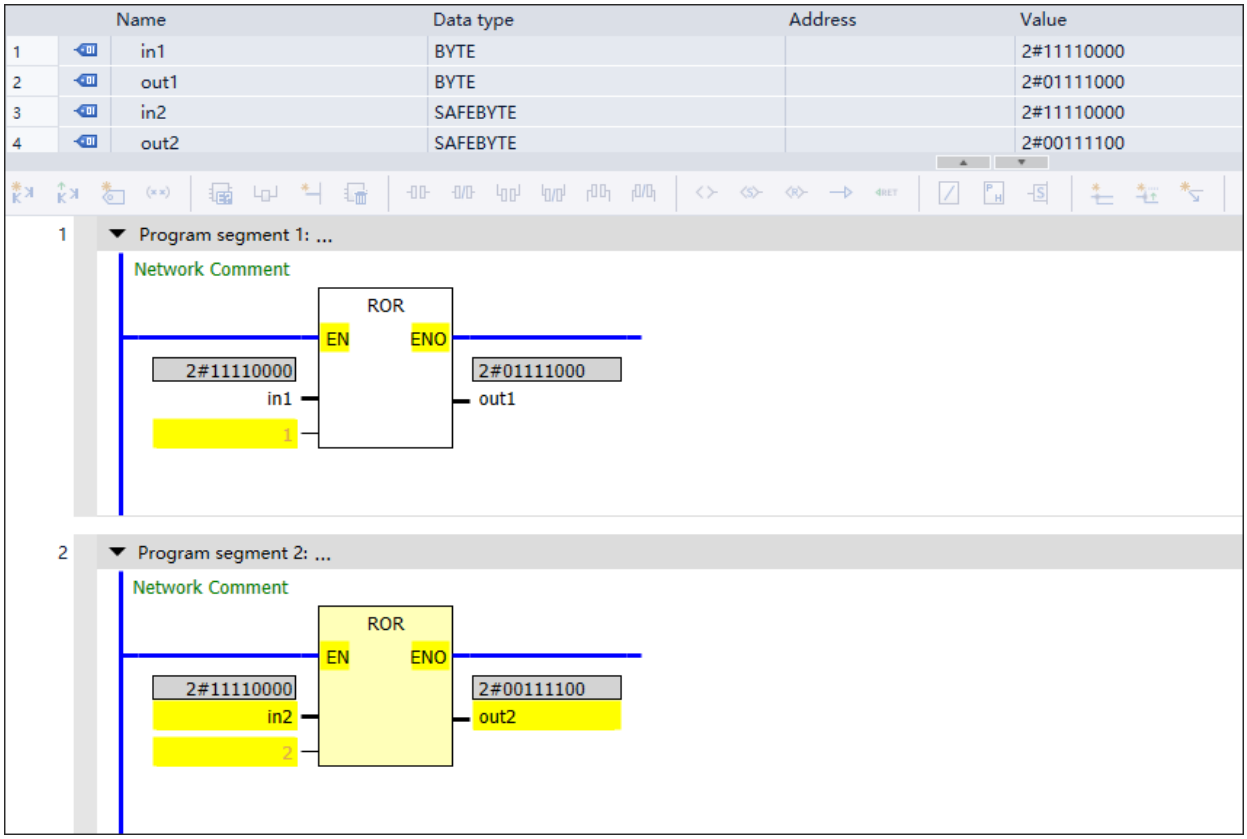
  When the source data is 2#11110000 and the number of shifted bits is 1, the output result is 2#01111000, with the valid data shifted right by one bit.

  When the source data is 2#11110000 and the number of shifted bits is 2, the output signal result is 2#00111100, with the valid data shifted right by two bits.

  FBD:



  LD:

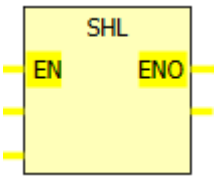
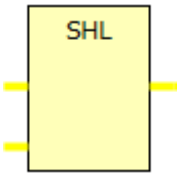


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - NUM only supports immediate values.

11.10.1.5.4 SHL

Left shift instruction

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SHL	Left shift	FC		

- **Related variables**

Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN1	Data Source	Follows data type	–	Data source
	NUM	Number of shifted bits	Follows data type	–	Number of shifted bits

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Return value	Follows data type	–	Shifted result

Data types

	Boolean		Bit String				Integer				Time Format	
	BOOL	SAFE BOOL	BYTE	SAFE BYTE	WORD	SAFE WORD	INT	SAFEINT	DINT	SAFE DINT	TIME	SAFE TIME
IN1	–	–	√	√	√	√	–	–	–	–	–	–
NUM	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	√	√	√	√	–	–	–	–	–	–

- **Function description**

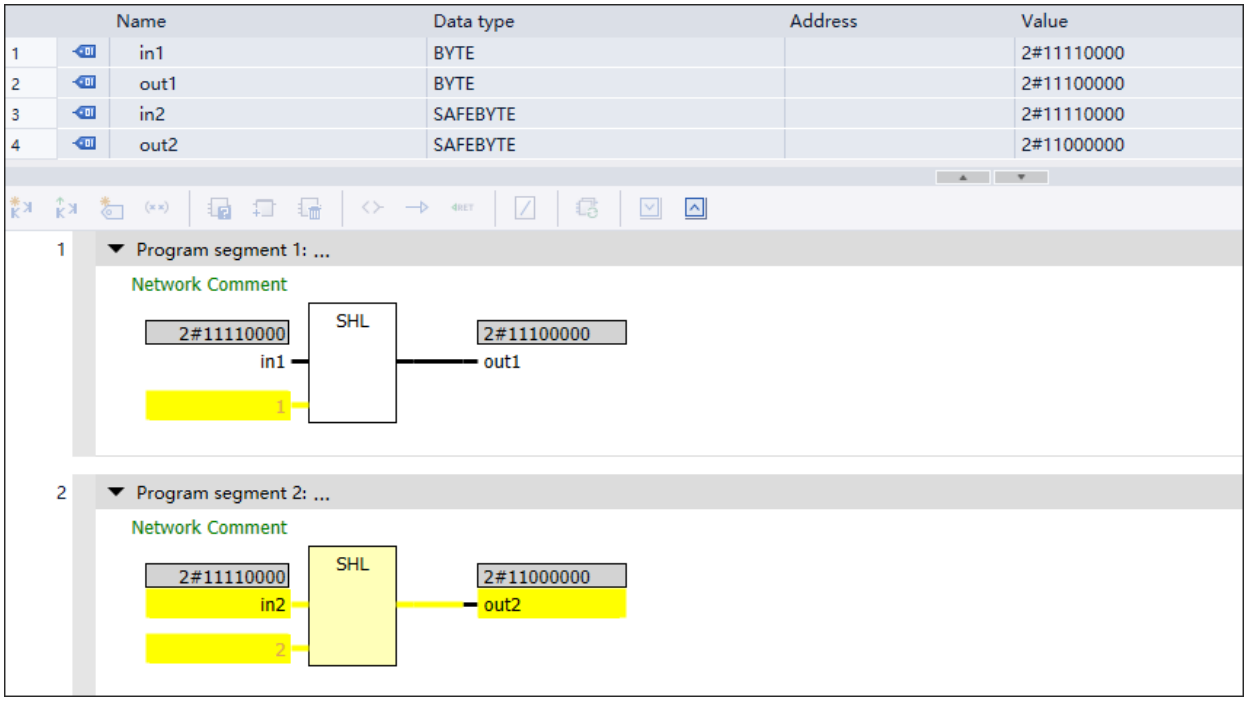
  Perform a bitwise left shift on the input value. The bits shifted out from the left are discarded, and zeros are automatically filled in on the right.

- **Program example**

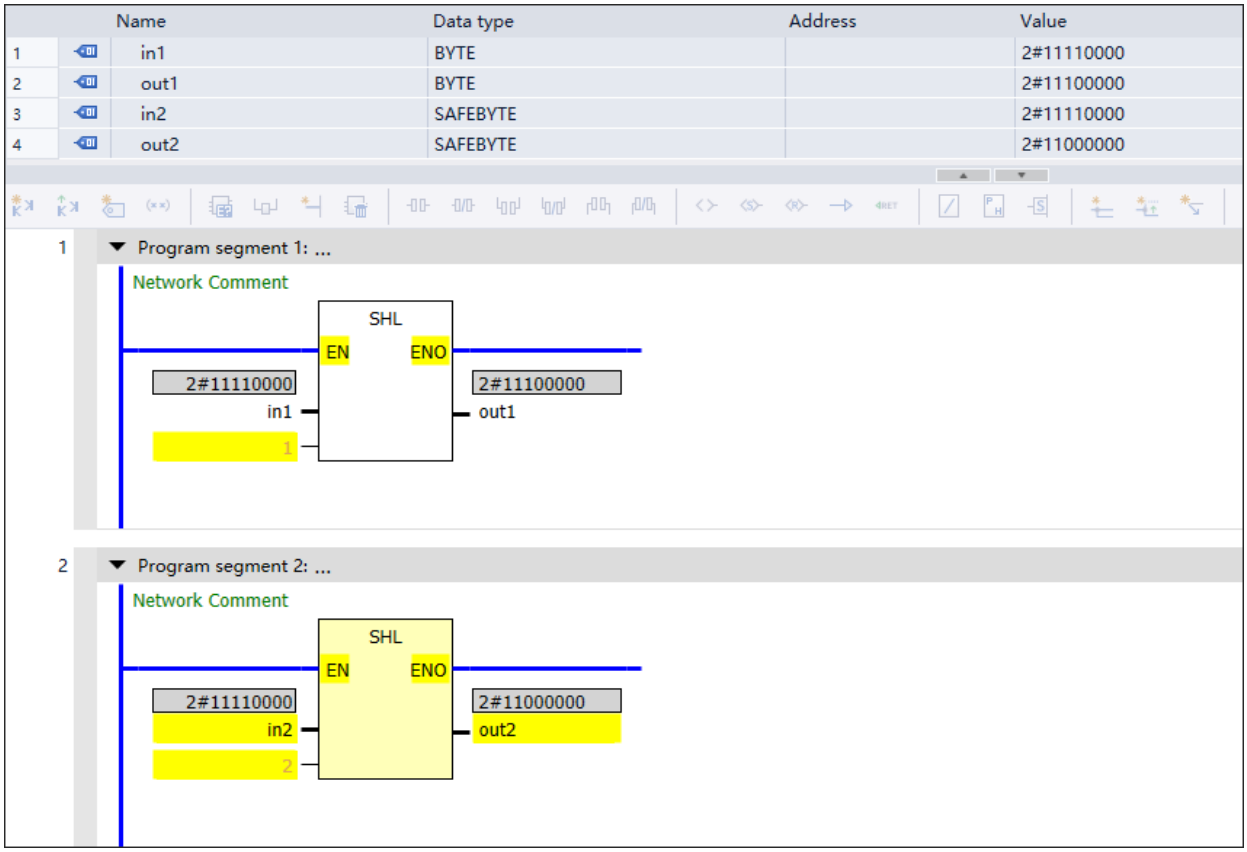
  When the source data is 2#11110000 and the number of shifted bits is 1, the output result is 2#11100000, with the valid data shifted left by one bit.

  When the source data is 2#11110000 and the number of shifted bits is 2, the output signal result is 2#11000000, with the valid data shifted left by two bits.

  FBD:



  LD:

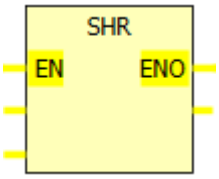
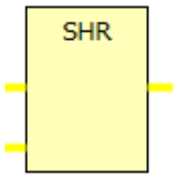


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - NUM only supports immediate values.

11.10.1.5.5 SHR

Right shift instruction

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SHR	Right shift	FC		

- **Related variables**

Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN1	Data Source	Follows data type	–	Data source
	NUM	Number of shifted bits	Follows data type	–	Number of shifted bits

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Return value	Follows data type	–	Shifted result

Data types

	Boolean		Bit String				Integer				Time Format	
	BOOL	SAFE BOOL	BYTE	SAFE BYTE	WORD	SAFE WORD	INT	SAFEINT	DINT	SAFE DINT	TIME	SAFE TIME
IN1	–	–	√	√	√	√	–	–	–	–	–	–
NUM	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	√	√	√	√	–	–	–	–	–	–

- **Function description**

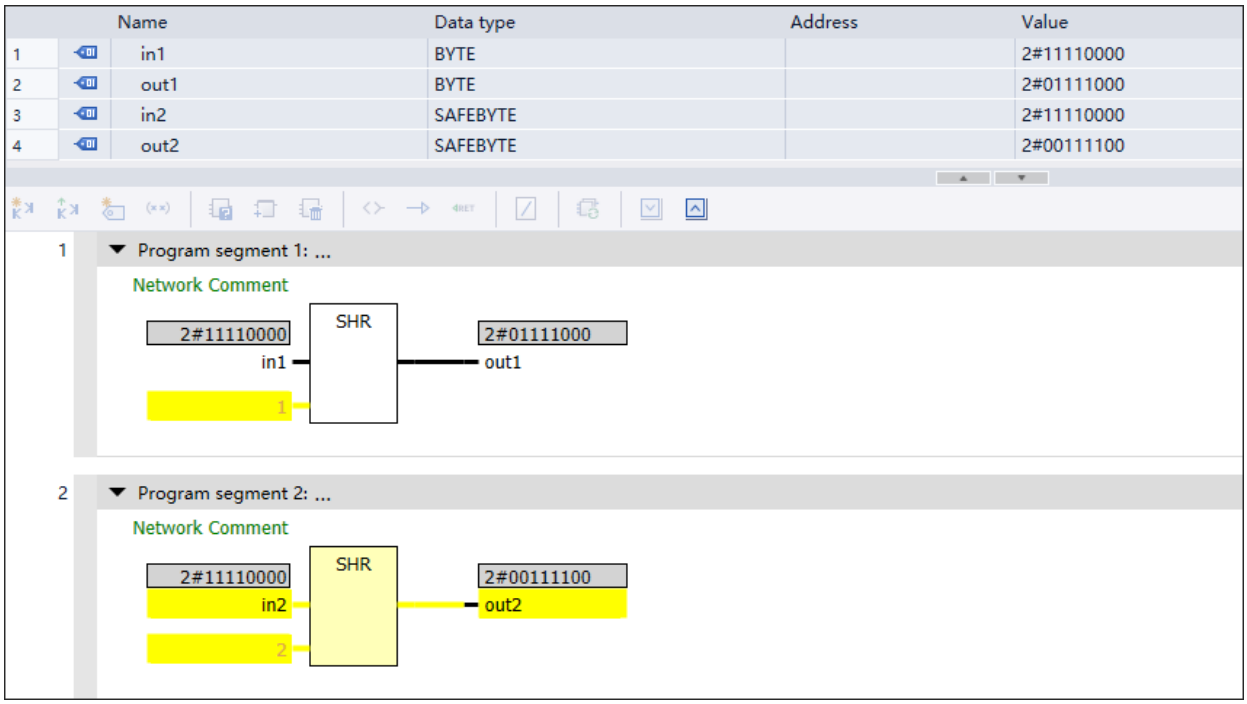
  Perform a bitwise right shift on the input value. The bits shifted out from the right are discarded, and zeros are automatically filled in on the left.

- **Program example**

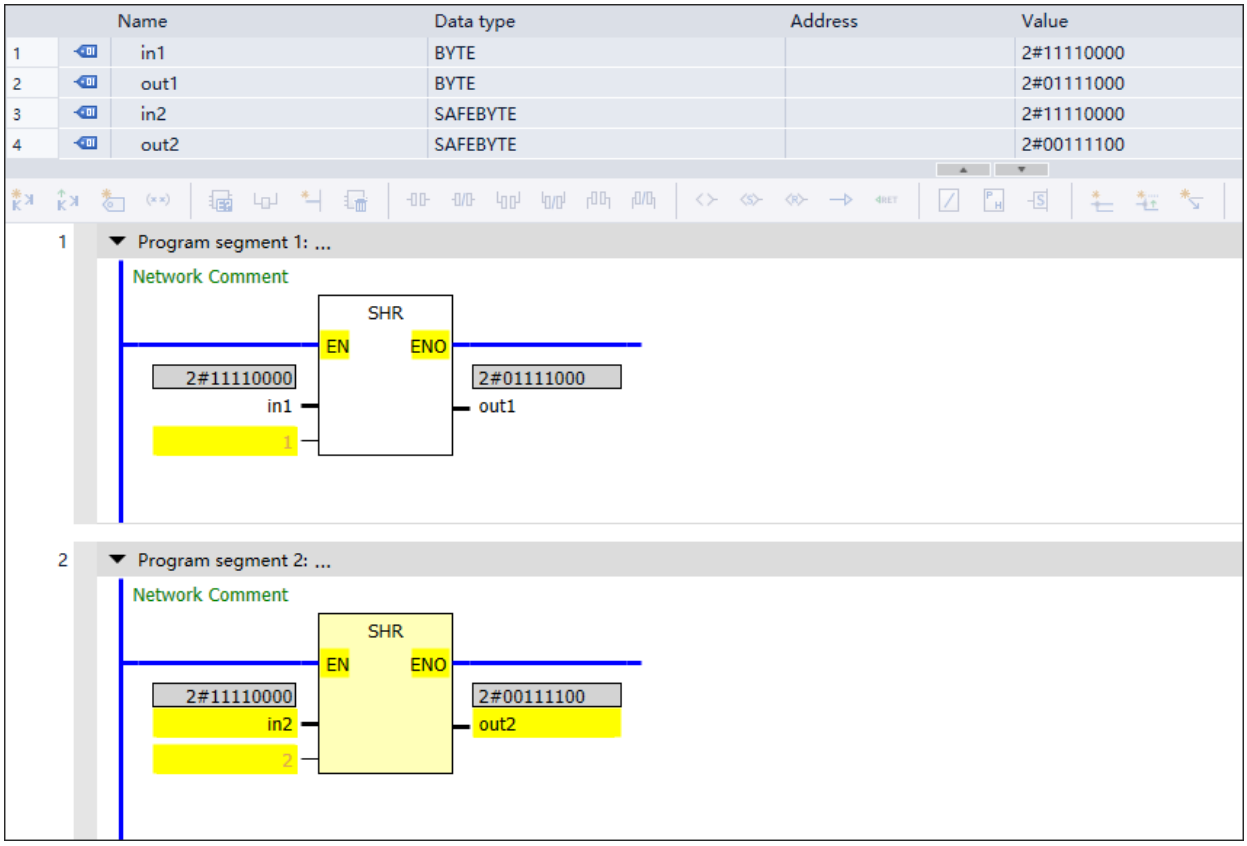
  When the source data is 2#11110000 and the number of shifted bits is 1, the output result is 2#01111000, with the valid data shifted right by one bit.

  When the source data is 2#11110000 and the number of shifted bits is 2, the output signal result is 2#00111100, with the valid data shifted right by two bits.

  FBD:



  LD:



- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - NUM only supports immediate values.

11.10.1.6 Conversion Operators

11.10.1.6.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
11.10.1.6.2 TIME_TO_WORD	FC	Convert data of TIME type to WORD type	SafetyInstructions	Applicable to library version 1.0.0.0/1.1.0.0
11.10.1.6.3 INT_TO_TIME	FC	Convert INT type to TIME type		
11.10.1.6.4 DINT_TO_WORD	FC	Convert DINT type to WORD type		
11.10.1.6.5 TIME_TO_DINT	FC	Convert TIME type to DINT type		
11.10.1.6.6 TIME_TO_BYTE	FC	Convert TIME type to BYTE type		
11.10.1.6.7 BOOL_TO_INT	FC	Convert BOOL type to INT type		

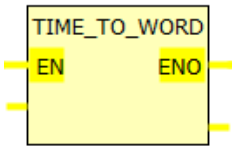
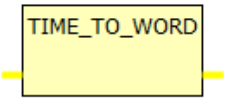
Instruction	FB/FC	Function	Library	Applicable Library Version
11.10.1.6.8 WORD_TO_INT	FC	Convert WORD type to INT type		
11.10.1.6.9 INT_TO_BYTE	FC	Convert INT type to BYTE type		
11.10.1.6.10 INT_TO_BOOL	FC	Convert INT type to BOOL type		
11.10.1.6.11 DINT_TO_TIME	FC	Convert DINT type to TIME type		
11.10.1.6.12 DINT_TO_BYTE	FC	Convert DINT type to BYTE type		
11.10.1.6.13 BOOL_TO_TIME	FC	Convert BOOL type to TIME type		
11.10.1.6.14 BYTE_TO_DINT	FC	Convert BYTE type to DINT type		
11.10.1.6.15 BOOL_TO_WORD	FC	Convert BOOL type to WORD type		
11.10.1.6.16 BYTE_TO_WORD	FC	Convert BYTE type to WORD type		
11.10.1.6.17 BYTE_TO_INT	FC	Convert BYTE type to INT type		
11.10.1.6.18 BOOL_TO_DINT	FC	Convert BOOL type to DINT type		
11.10.1.6.19 BYTE_TO_TIME	FC	Convert BYTE type to TIME type		
11.10.1.6.20 INT_TO_WORD	FC	Convert INT type to WORD type		
11.10.1.6.21 BOOL_TO_BYTE	FC	Convert BOOL type to BYTE type		
11.10.1.6.22 DINT_TO_BOOL	FC	Convert DINT type to BOOL type		
11.10.1.6.23 WORD_TO_BOOL	FC	Convert WORD type data to BOOL type		
11.10.1.6.24 BYTE_TO_BOOL	FC	Convert BYTE type data to BOOL type		
11.10.1.6.25 INT_TO_DINT	FC	Convert INT type data to DINT type		
11.10.1.6.26 WORD_TO_BYTE	FC	Convert WORD type data to BYTE type		

Instruction	FB/FC	Function	Library	Applicable Library Version
11.10.1.6.27 WORD_TO_TIME	FC	Convert WORD type data to TIME type		
11.10.1.6.28 WORD_TO_DINT	FC	Convert WORD type data to DINT type		
11.10.1.6.29 DINT_TO_INT	FC	Convert DINT type data to INT type		
11.10.1.6.30 TIME_TO_INT	FC	Convert TIME type data to INT type		
11.10.1.6.31 TIME_TO_BOOL	FC	Convert TIME type data to BOOL type		
11.10.1.6.32 BOOL_TO_DWORD	FC	Convert BOOL type data to DWORD type		Applicable to library version 1.1.0.0
11.10.1.6.33 BYTE_TO_DWORD	FC	Convert BYTE type data to DWORD type		
11.10.1.6.34 DINT_TO_DWORD	FC	Convert DINT type data to DWORD type		
11.10.1.6.35 DWORD_TO_BOOL	FC	Convert DWORD type data to BOOL type		
11.10.1.6.36 DWORD_TO_BYTE	FC	Convert DWORD type data to BYTE type		
11.10.1.6.37 DWORD_TO_DINT	FC	Convert DWORD type data to DINT type		
11.10.1.6.38 DWORD_TO_INT	FC	Convert DWORD type data to INT type		
11.10.1.6.39 DWORD_TO_TIME	FC	Convert DWORD type data to TIME type		
11.10.1.6.40 DWORD_TO_WORD	FC	Convert DWORD type data to WORD type		
11.10.1.6.41 INT_TO_DWORD	FC	Convert INT type data to DWORD type		
11.10.1.6.42 TIME_TO_DWORD	FC	Convert TIME type data to DWORD type		
11.10.1.6.43 WORD_TO_DWORD	FC	Convert WORD type data to DWORD type		

11.10.1.6.2 TIME_TO_WORD

Convert data of TIME type to WORD type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
TIME_TO_WORD	Convert data of TIME type to WORD type	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	T#0ms to T#65s535ms	T#0ms	Conversion object
Output	Out	Conversion result	WORD#16#0000 to FFFF	WORD#16#0000	Conversion result

  Data types

	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	-	-	-	-	-	-	√	√
Out	-	-	-	-	√	√	-	-	-	-	-	-	-	-

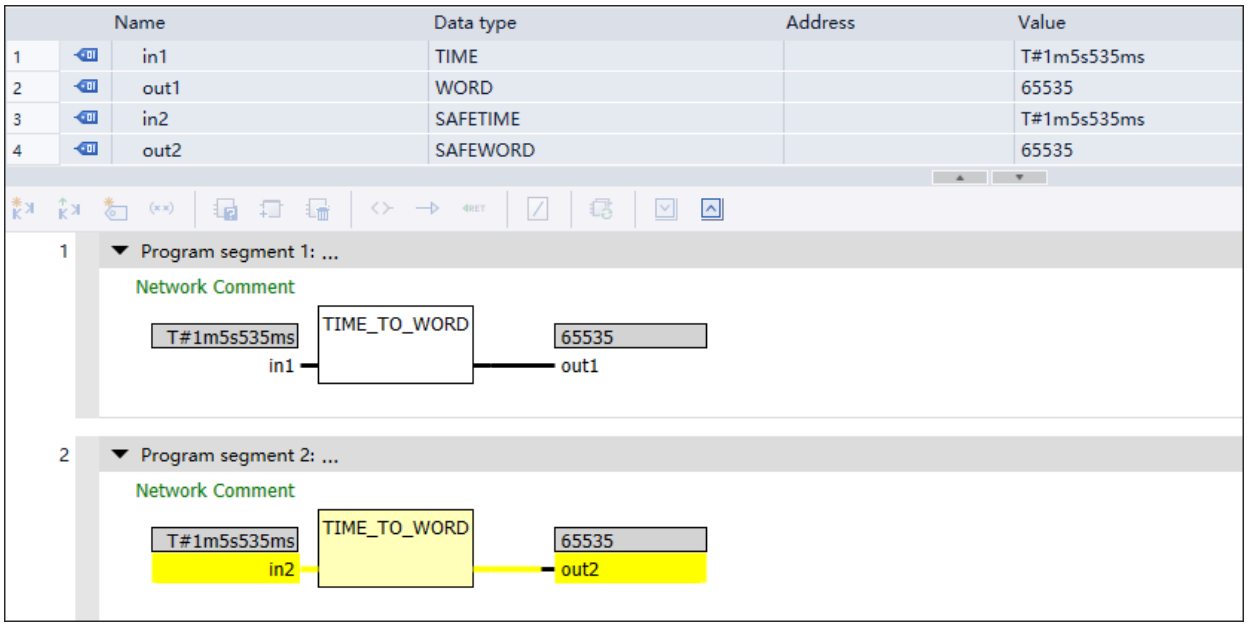
- **Function description**

  Convert "In" of TIME type to the WORD-type bit string "Out".

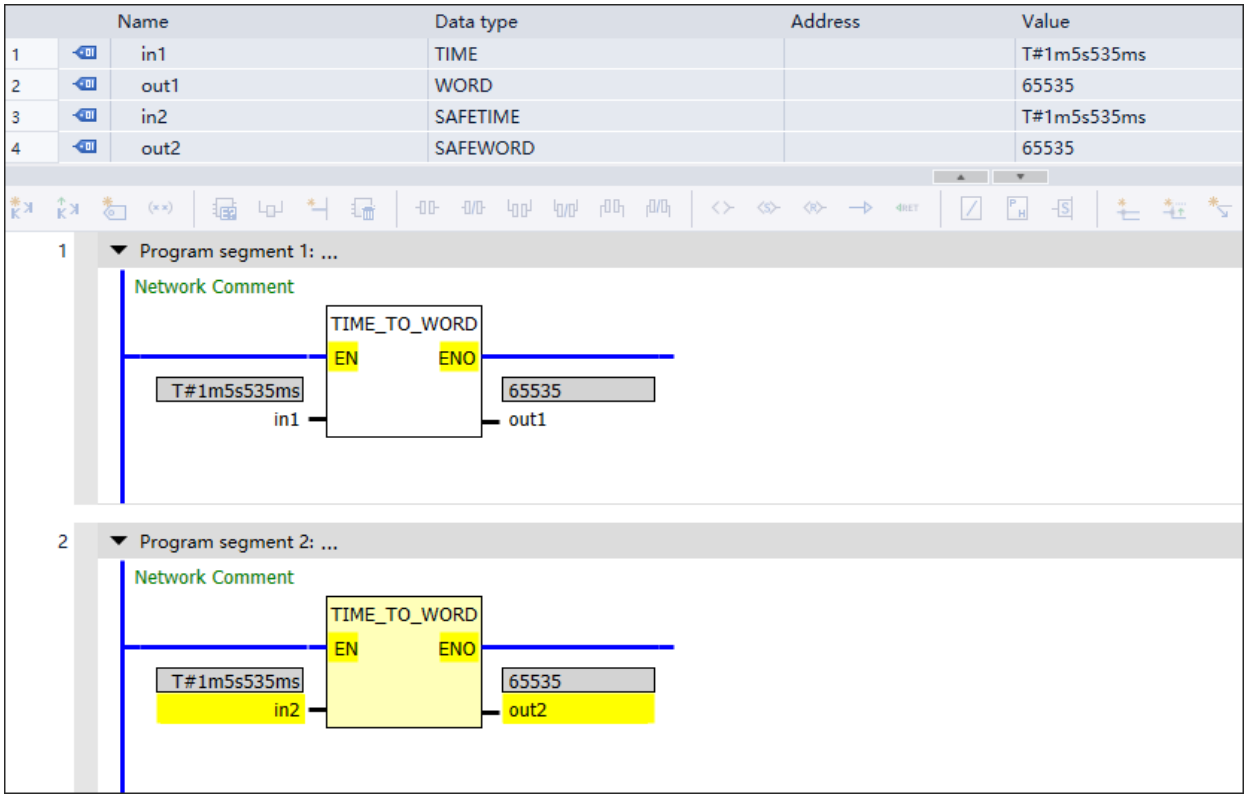
  Example: When the value of "In" is T#50s505ms, the value of "Out" is WORD#16#C549.

- **Programming example**

  FBD:



  LD:

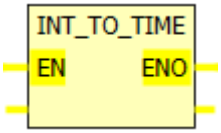
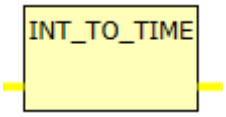


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - If a value outside the range of TIME#0ms to 66535ms is input, a "conversion exception" will occur and the program will stop.

11.10.1.6.3 INT_TO_TIME

Convert data of INT type to TIME type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
INT_TO_TIME	Convert INT type to TIME type	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	INT#0 to 32767	INT#0	Conversion object
Output	Out	Conversion result	T#0ms to T#32s767ms	T#0ms	Conversion result

  Data types

	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	-	-	√	√	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	√	√

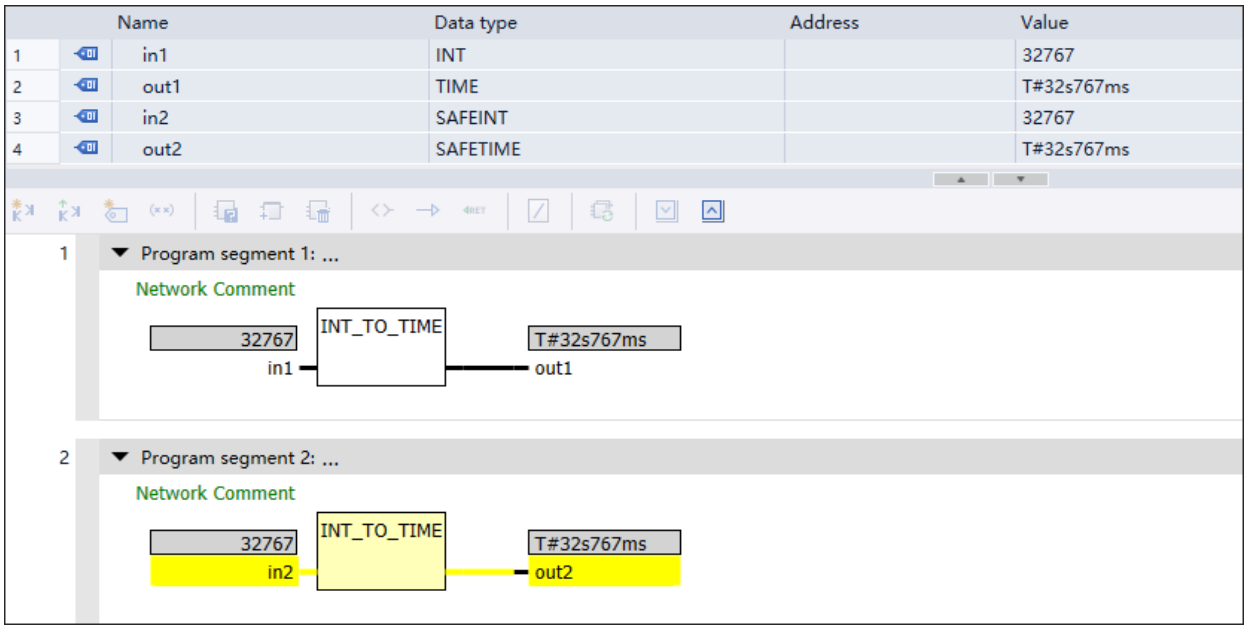
- **Function description**

  Convert "In" of INT type to "Out" of TIME type.

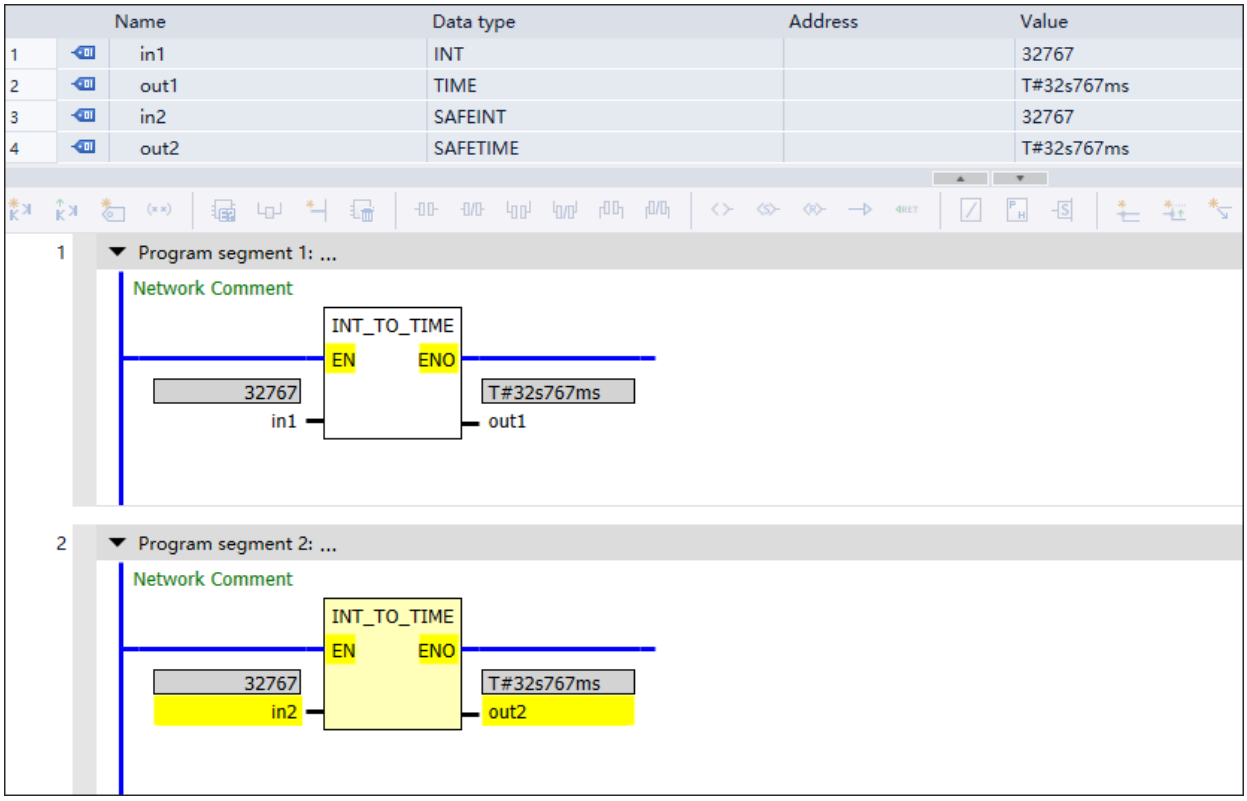
  Example: When the value of "In" is INT#10500, the value of "Out" is T#10s500ms.

- **Programming example**

  FBD:



  LD:

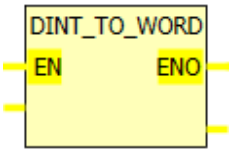
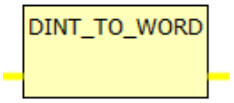


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - If a value outside the range of INT#0 to 32767 is input, a "conversion exception" will occur and the program will stop.

11.10.1.6.4 DINT_TO_WORD

Convert data of DINT type to WORD type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
DINT_TO_WORD	Convert DINT type to WORD type	FC		

- **Related variables**

Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	DINT#0 to 65535	DINT#0	Conversion object
Output	Out	Conversion result	WORD#16#0000 to FFFF	WORD#16#0000	Conversion result

Data types

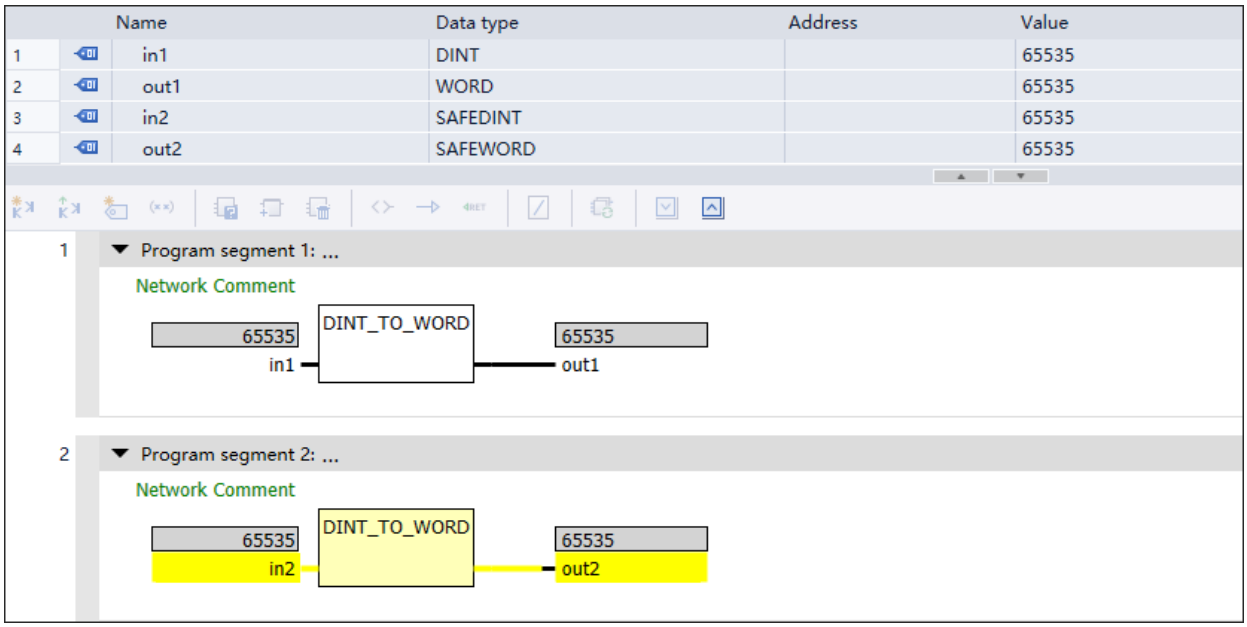
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	-	-	-	-	√	√	-	-
Out	-	-	-	-	√	√	-	-	-	-	-	-	-	-

- **Function description**

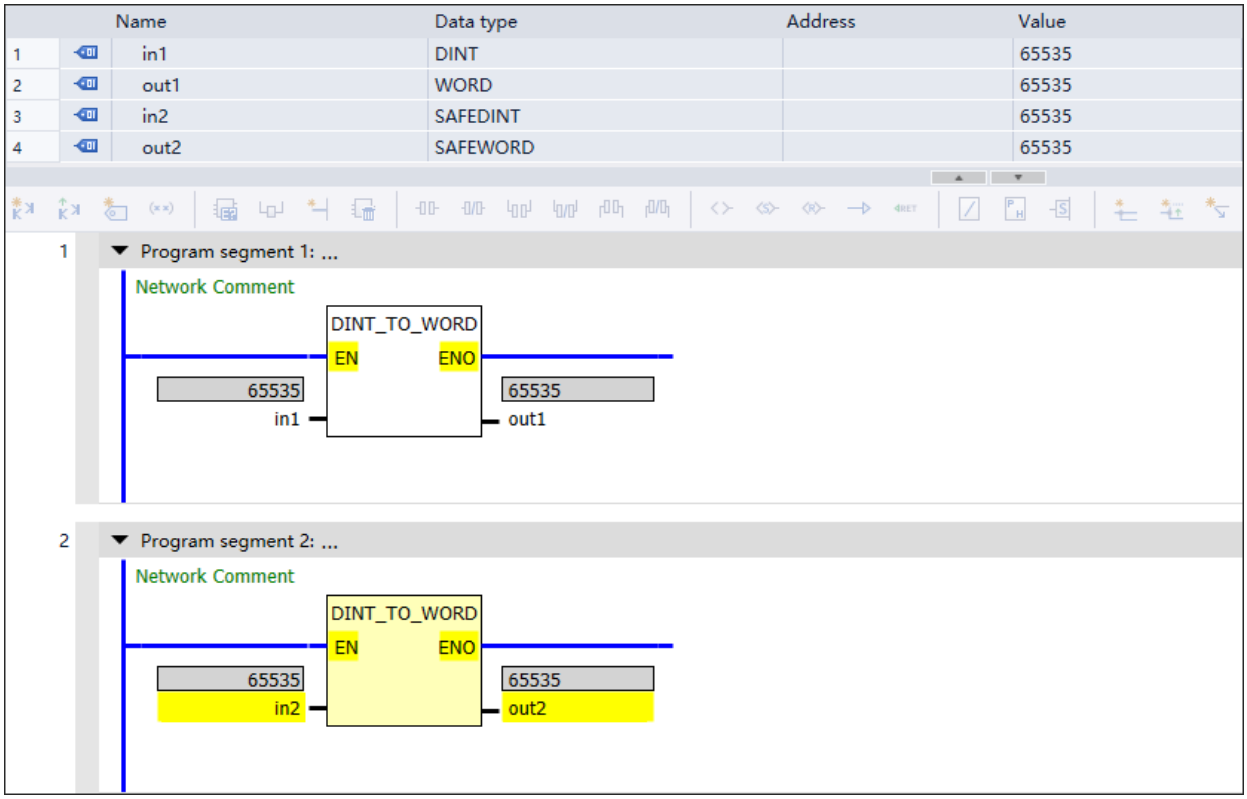
  Convert "In" of DINT type to the bit string "Out" of WORD type.

- **Programming example**

  FBD:



  LD:

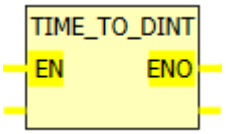
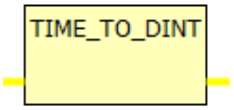


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - If a value outside the range of DINT#0 to 65535 is input, a "conversion exception" will occur and the program will stop.

11.10.1.6.5 TIME_TO_DINT

Convert data of TIME type to DINT type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
TIME_TO_DINT	Convert TIME type to DINT type	FC		

- **Related variables**

Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	T#0ms to T#49d17h2m47s295ms	T#0ms	Conversion object
Output	Out	Conversion result	DINT#0 to 2147483647	DINT#0	Conversion result

Data types

	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	-	-	-	-	-	-	√	√
Out	-	-	-	-	-	-	-	-	-	-	√	√	-	-

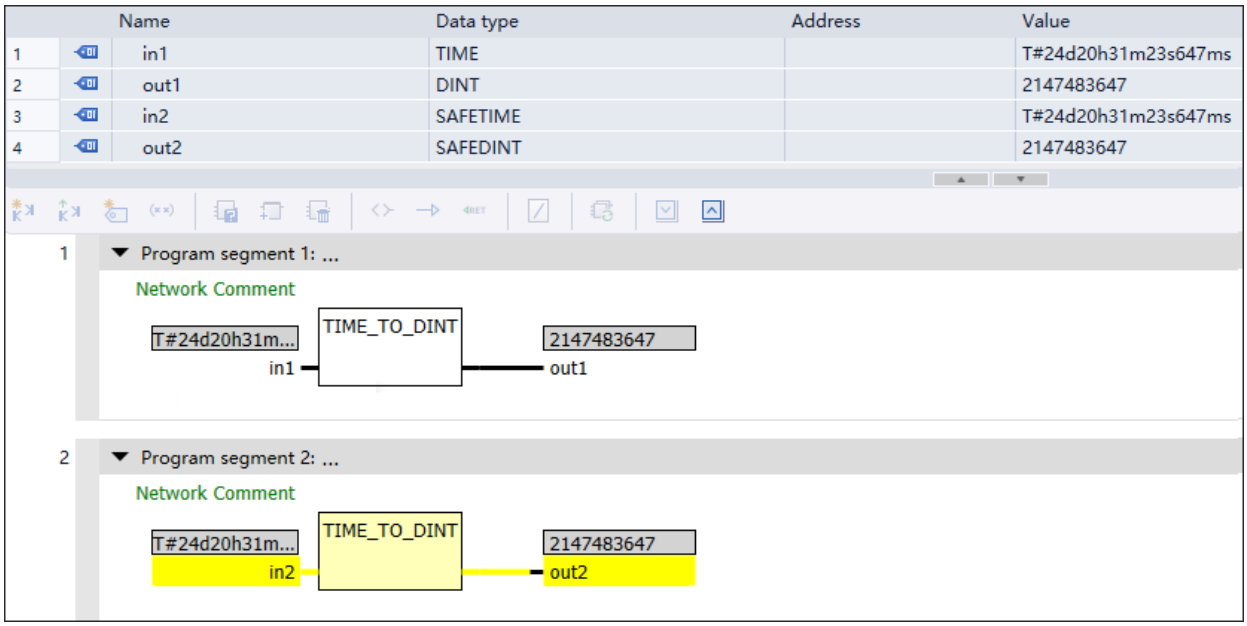
- **Function description**

  Convert "In" of TIME type to "Out" of DINT type.

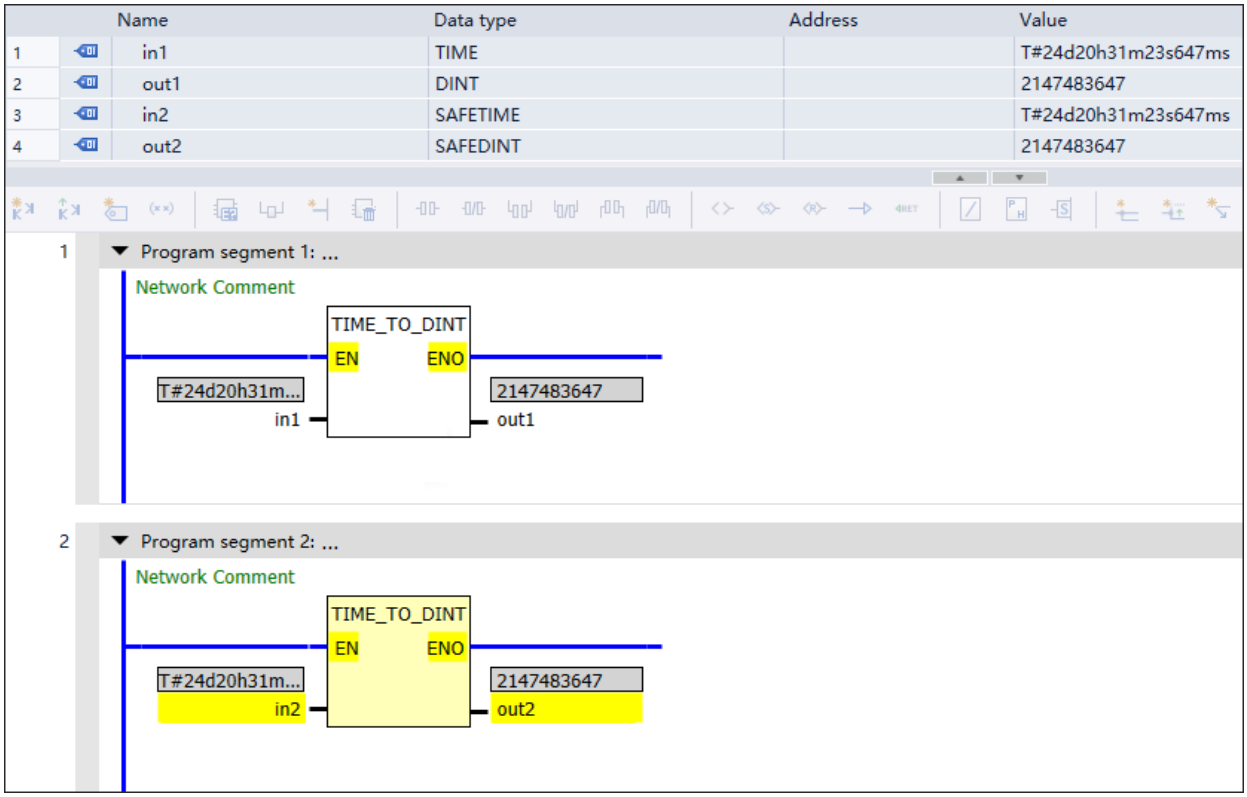
  Example: When the value of "In" is T#1d1h1m30s090ms, the value of "Out" is DINT#90090090.

- **Programming example**

  FBD:



  LD:

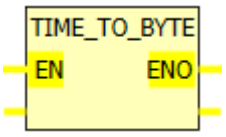
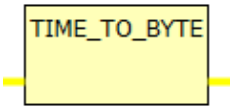


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - If a value outside the range of TIME#0ms to 2147483647 ms is input, a "conversion exception" will occur and the program will stop.

11.10.1.6.6 TIME_TO_BYTE

Convert data of TIME type to BYTE type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
TIME_TO_BYTE	Convert TIME type to BYTE type	FC		

- **Related variables**

Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	T#0ms to T#255ms	T#0ms	Conversion object
Output	Out	Conversion result	BYTE16#00 to FF	BYTE16#00	Conversion result

Data types

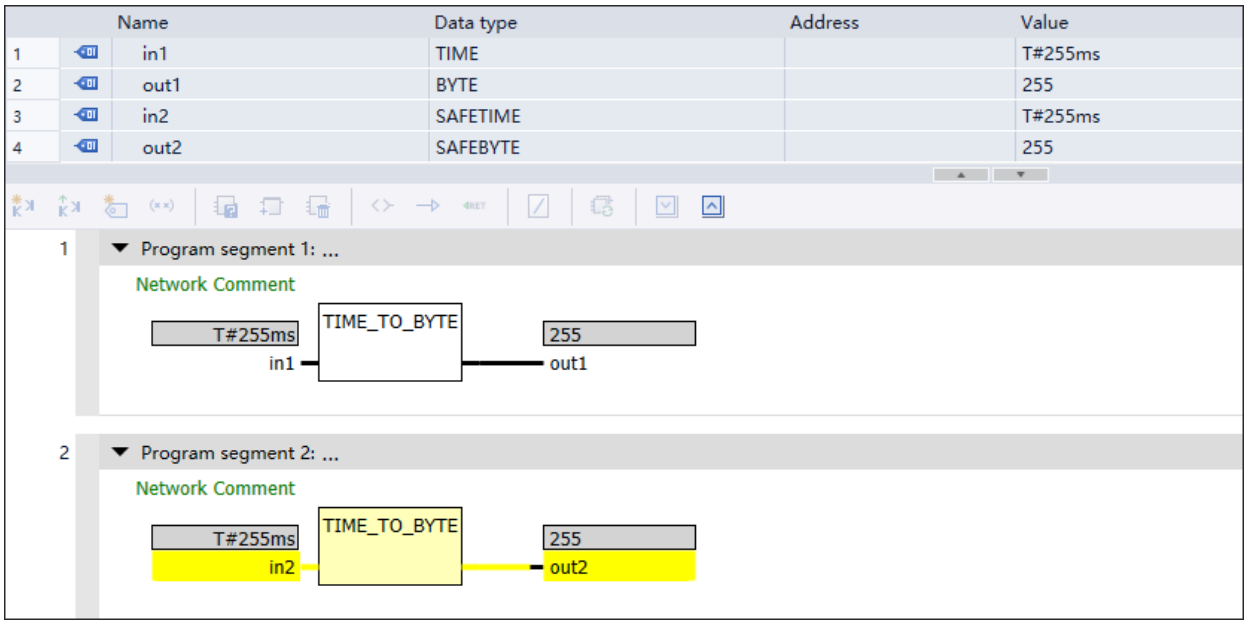
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	-	-	-	-	-	-	√	√
Out	-	-	√	√	-	-	-	-	-	-	-	-	-	-

- **Function description**

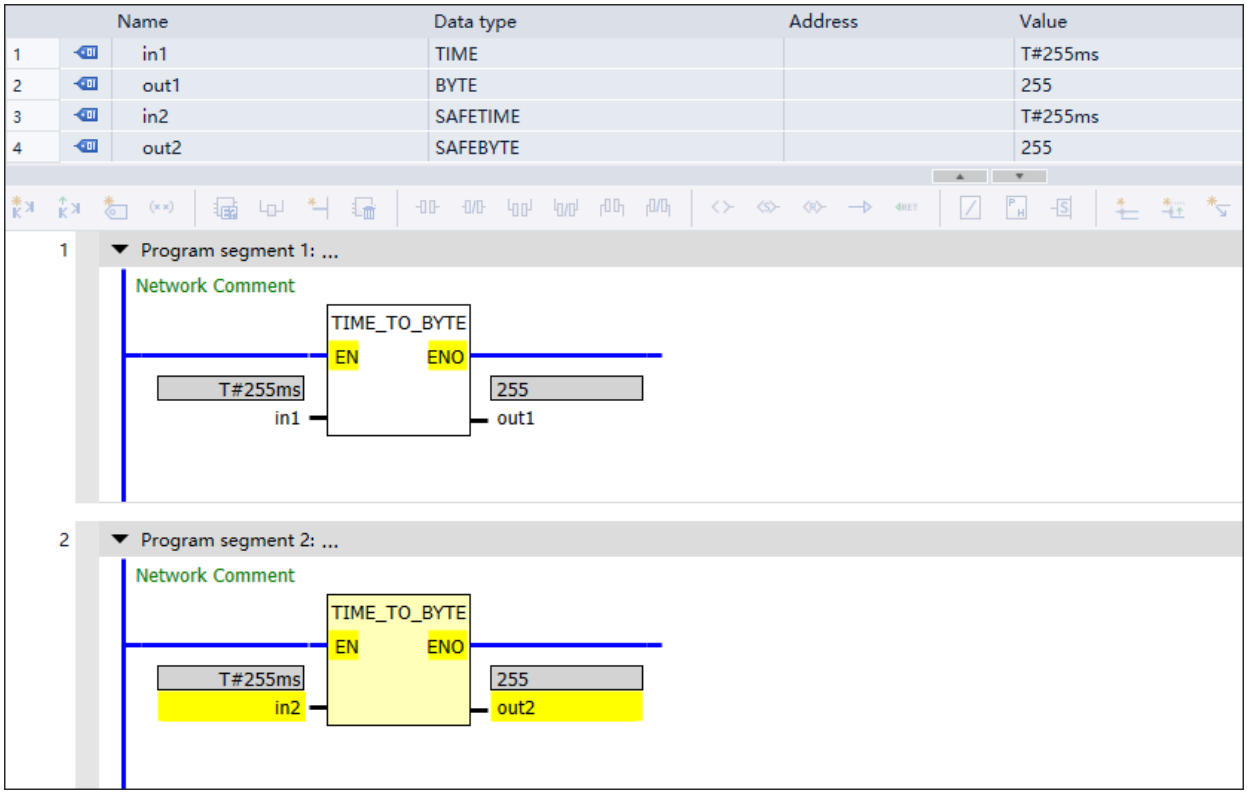
  Convert "In" of TIME type to the bit string "Out" of BYTE type.

- **Programming example**

  FBD:



  LD:

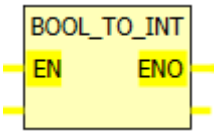
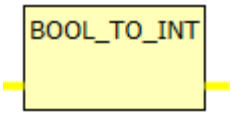


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - If a value outside the range of TIME#0ms to 255 ms is input, a "conversion exception" will occur and the program will stop.

11.10.1.6.7 BOOL_TO_INT

Convert data of BOOL type to INT type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
BOOL_TO_INT	Convert BOOL type to INT type	FC		

- **Related variables**

Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	FALSE, TRUE	FALSE	Conversion object
Output	Out	Conversion result	INT#0, INT#1	INT#0	Conversion result

Data types

	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	√	√	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	√	√	-	-	-	-

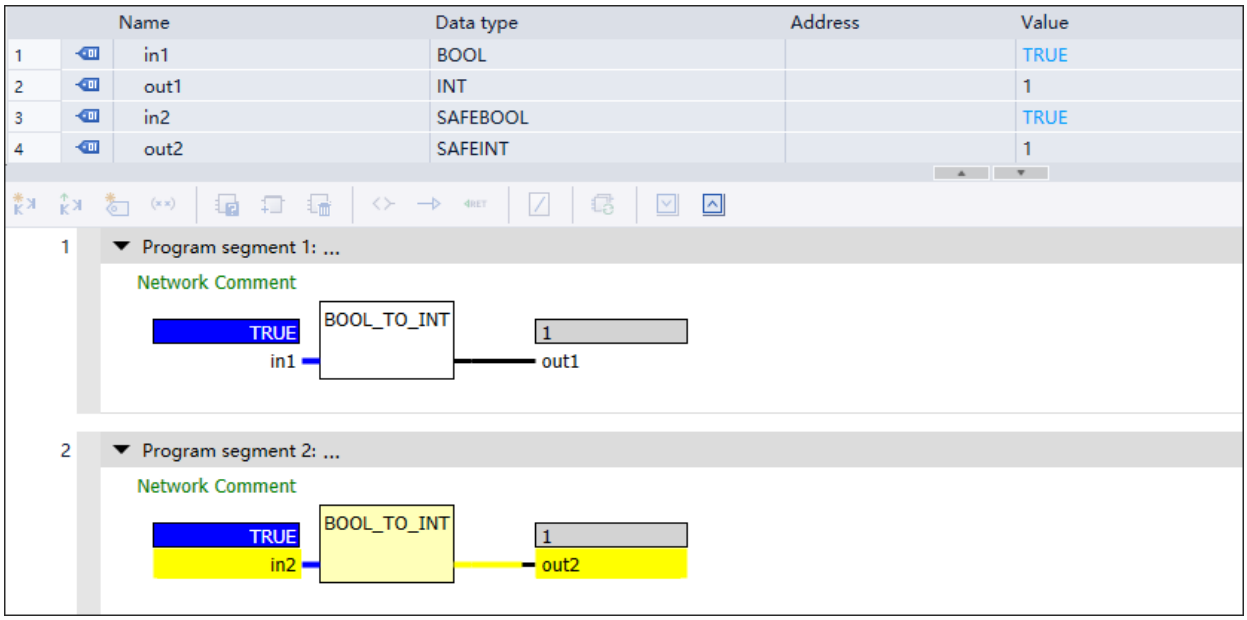
- **Function description**

  Convert "In" of BOOL type to "Out" of INT type.

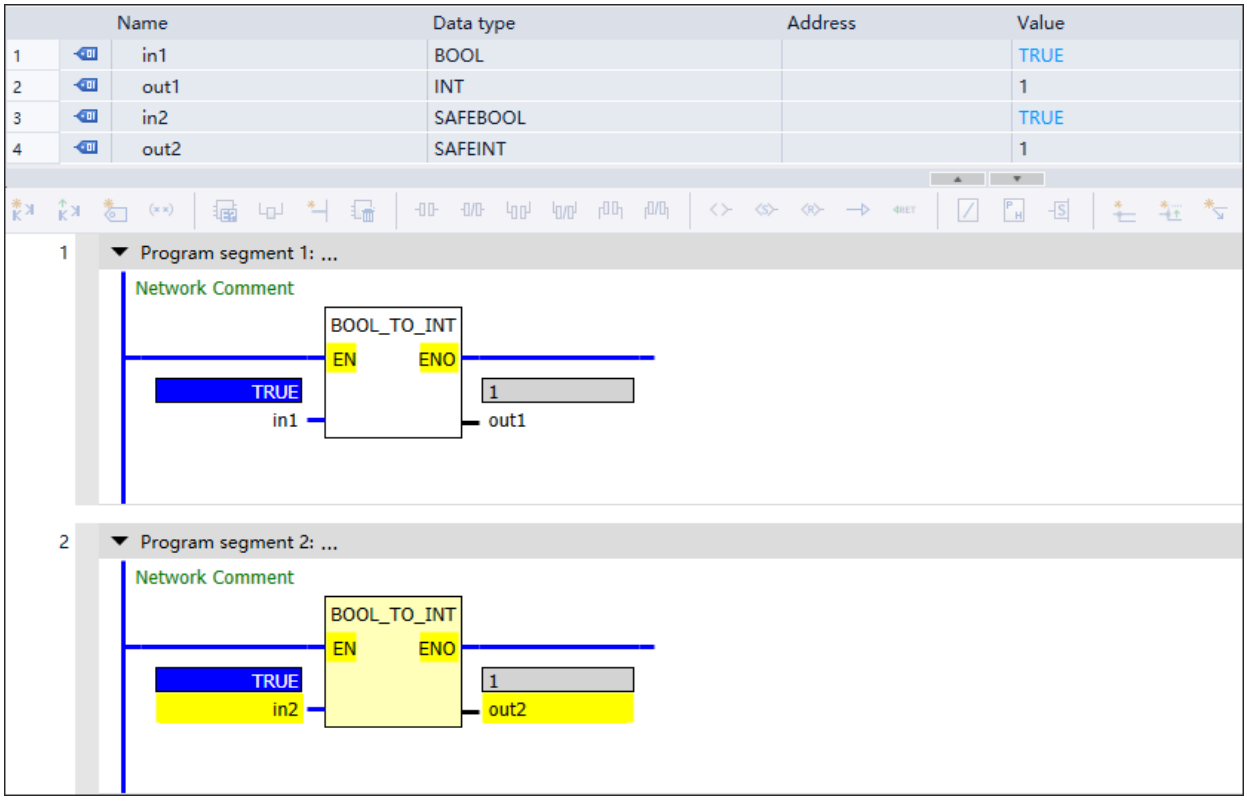
  - When the value of "In" is FALSE, the value of "Out" is INT#0.
  - When the value of "In" is TRUE, the value of "Out" is INT#1.

- **Programming example**

  FBD:



  LD:

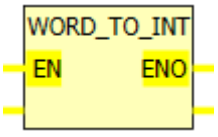
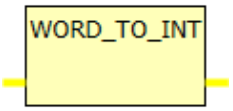


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.6.8 WORD_TO_INT

Convert data of WORD type to INT type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
WORD_TO_INT	Convert WORD type to INT type	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	WORD#16#0000 to FFFF	WORD#16#0000	Conversion object
Output	Out	Conversion result	INT#-32768 to 32767	INT#0	Conversion result

  Data types

	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	√	√	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	√	√	-	-	-	-

- **Function description**

  Convert the bit string "In" of WORD type to "Out" of INT type.

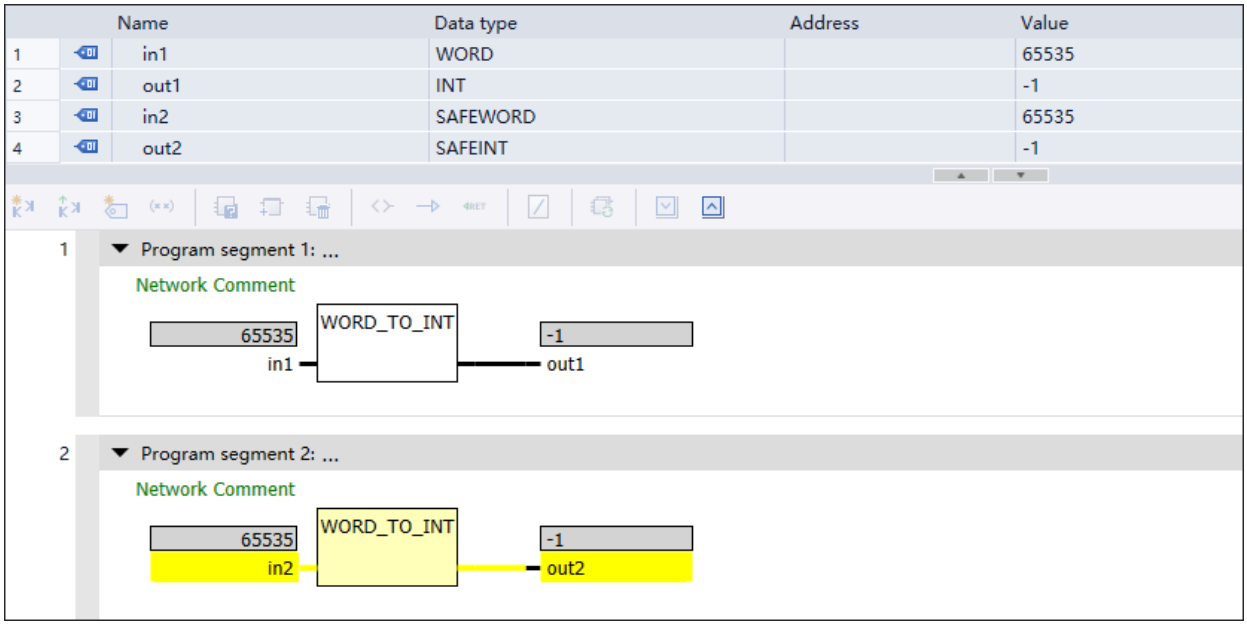
  - Example when the value of "Out" is positive (INT#0 to 32767):

    When the value of "In" is WORD#16#0000 to 7FFF, the value of "Out" is INT#0 to 32767.

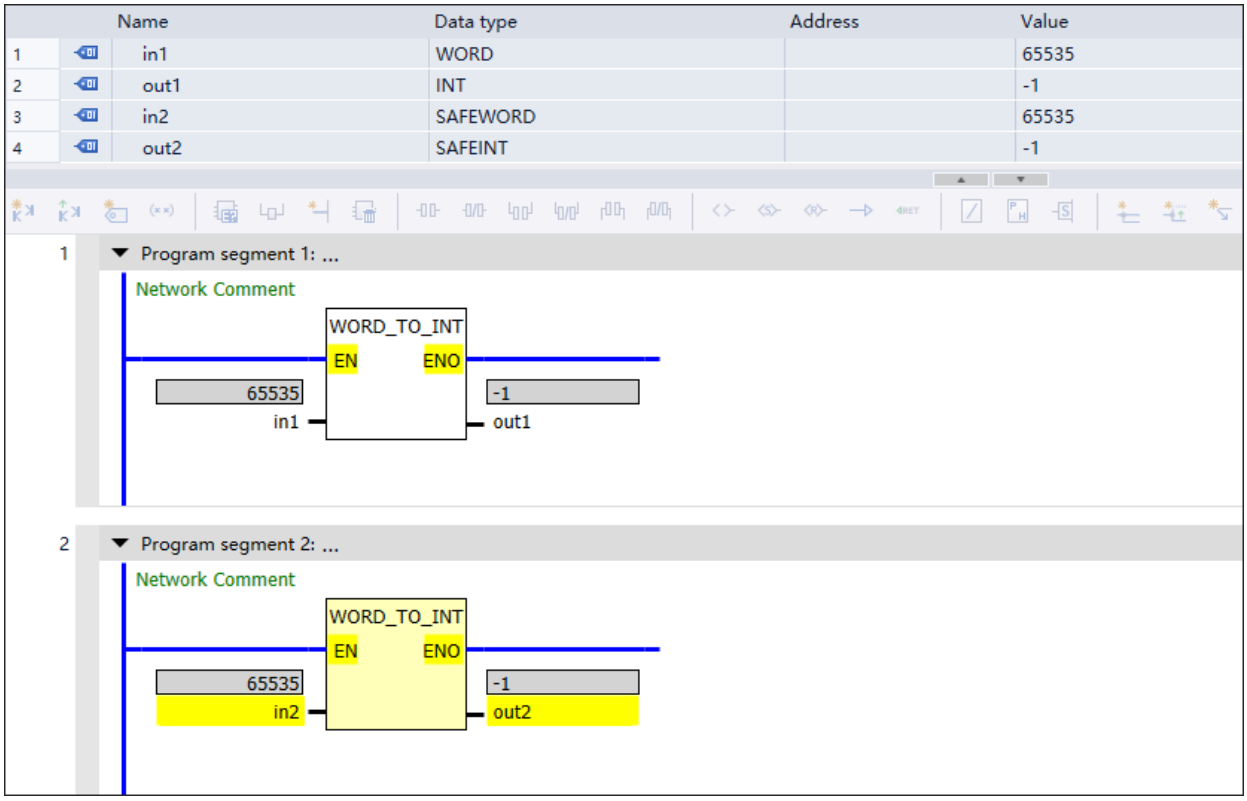
  - Example when the value of "Out" is negative (INT#-32768 to -1):
    - When the value of "In" is WORD#16#8000, which is 1000 0000 0000 0000 (BIN), the value of "Out" is INT#32768, represented as INT#-32768 using 2 bytes.
    - When the value of "In" is WORD#16#FFFF, which is 1111 1111 1111 1111 (BIN), the value of "Out" is INT#65535, represented as INT#-1 using 2 bytes.

- **Programming example**

  FBD:



  LD:

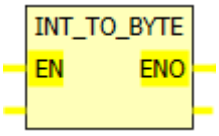
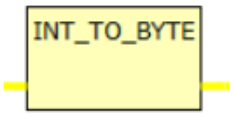


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.6.9 INT_TO_BYTE

Convert data of INT type to BYTE type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
INT_TO_BYTE	Convert INT type to BYTE type	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	INT#0 to 255	INT#0	Conversion object
Output	Out	Conversion result	BYTE#16#00 to FF	BYTE#16#00	Conversion result

  Data types

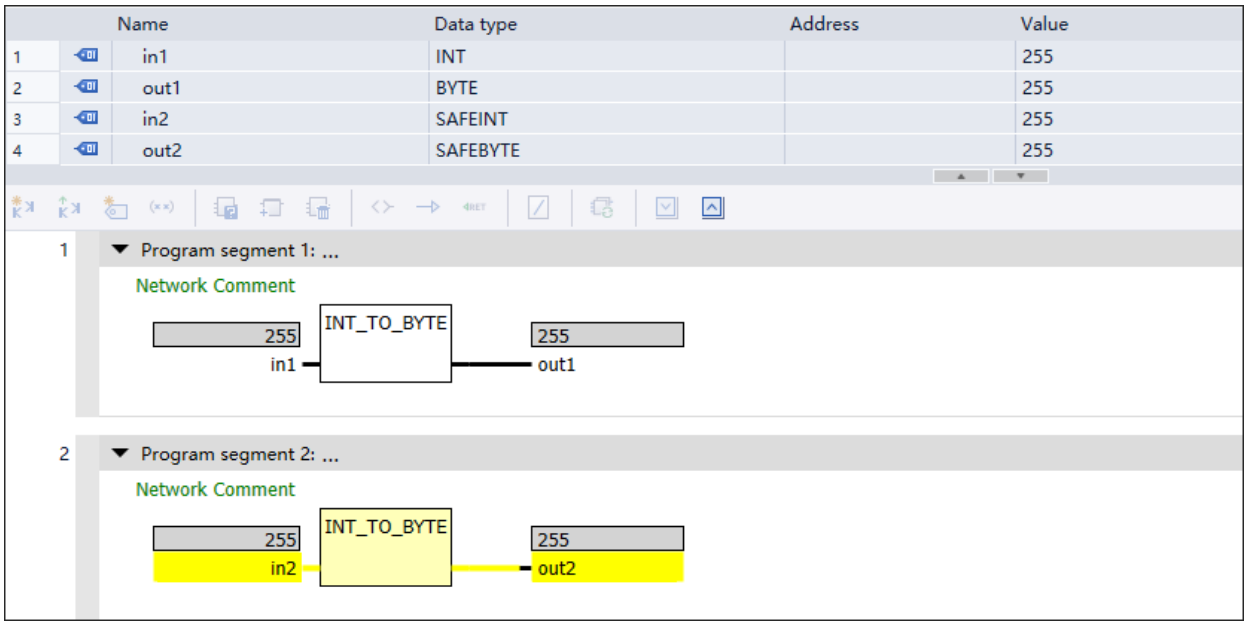
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	-	-	√	√	-	-	-	-
Out	-	-	√	√	-	-	-	-	-	-	-	-	-	-

- **Function description**

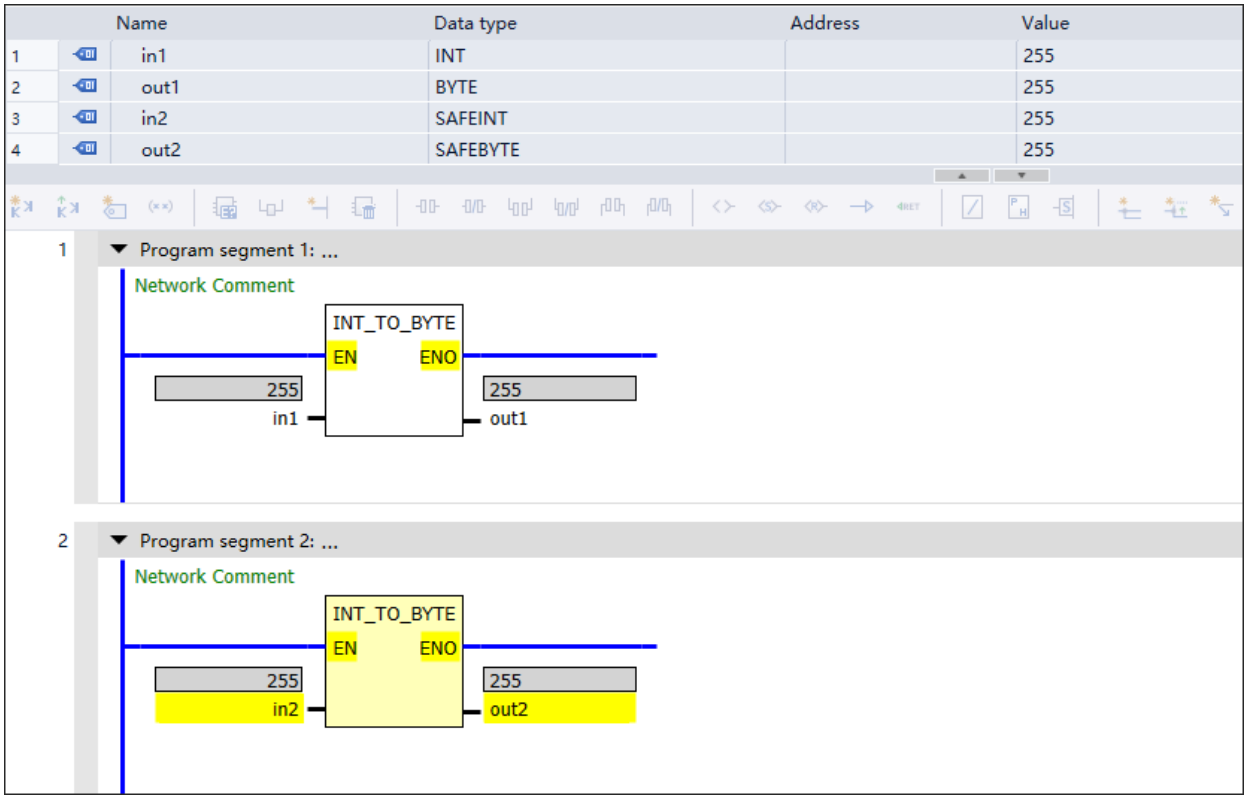
  Convert "In" of INT type to the bit string "Out" of BYTE type.

- **Programming example**

  FBD:



  LD:

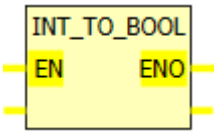
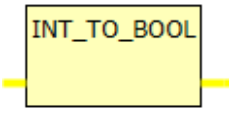


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - If a value outside the range of INT#0 to 255 is input, a "conversion exception" will occur and the program will stop.

11.10.1.6.10 INT_TO_BOOL

Convert data of INT type to BOOL type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
INT_TO_BOOL	Convert INT type to BOOL type	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	INT#-32768 to 32767	INT#0	Conversion object
Output	Out	Conversion result	FALSE, TRUE	FALSE	Conversion result

  Data types

	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	-	-	√	√	-	-	-	-
Out	√	√	-	-	-	-	-	-	-	-	-	-	-	-

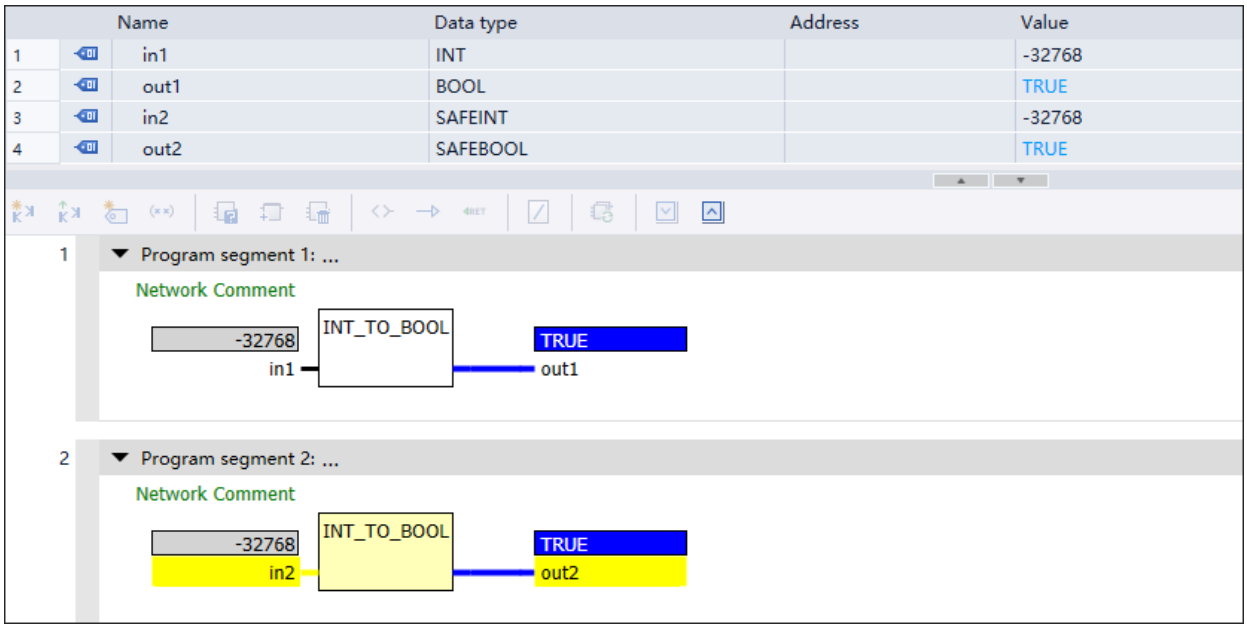
- **Function description**

  Convert "In" of INT type to "Out" of BOOL type.

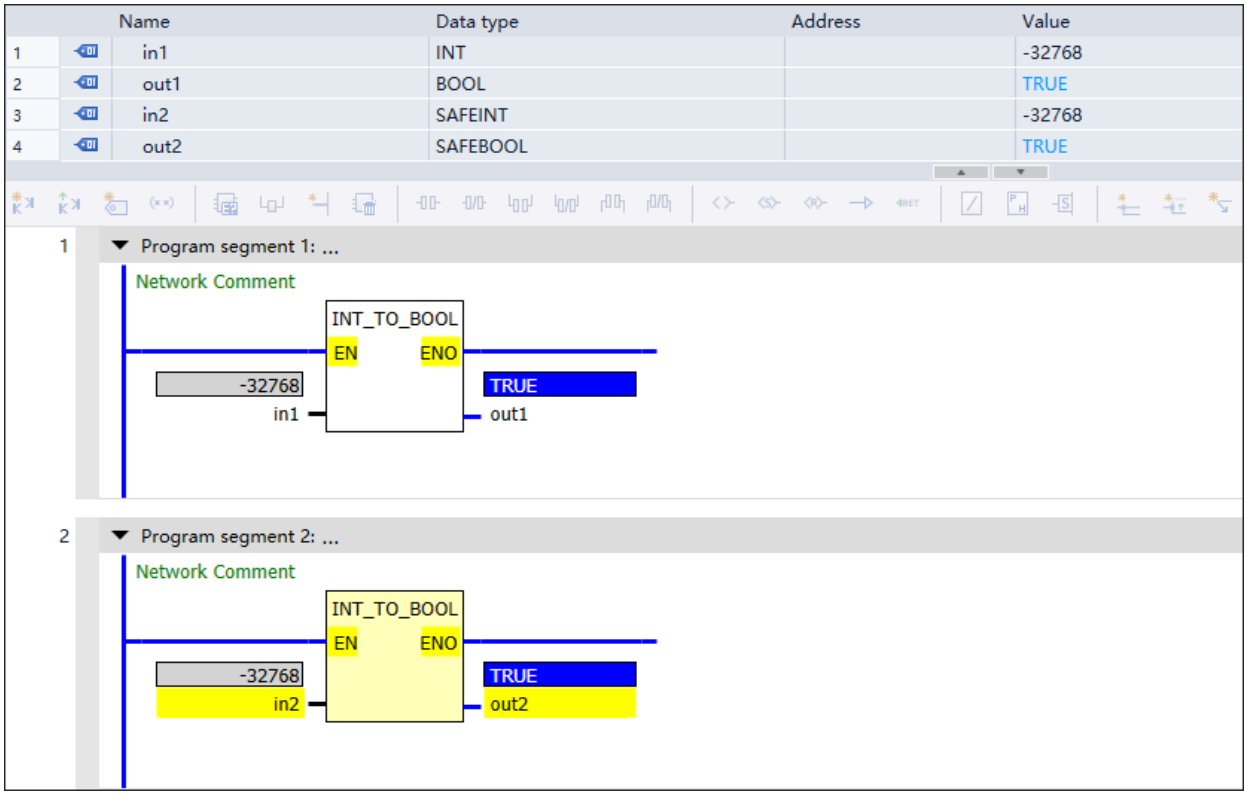
  When the value of "In" is INT#0, the value of "Out" is FALSE; when the value of "In" is INT#-32768 to -1 or INT#1 to 32767 (excluding INT#0), the value of "Out" is TRUE.

- **Programming example**

  FBD:



  LD:

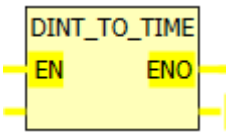
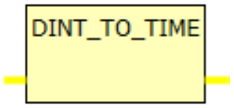


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - If a value outside the range of INT#-32768 to 32767 is input, a "conversion exception" will occur and the program will stop.

11.10.1.6.11 DINT_TO_TIME

Convert data of DINT type to TIME type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
DINT_TO_TIME	Convert DINT type to TIME type	FC		

- **Related variables**

Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	DINT#0 to 2147483647	DINT#0	Conversion object
Output	Out	Conversion result	T#0ms to T#24d20h31m23s647ms	T#0ms	Conversion result

Data types

	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	-	-	-	-	√	√	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	√	√

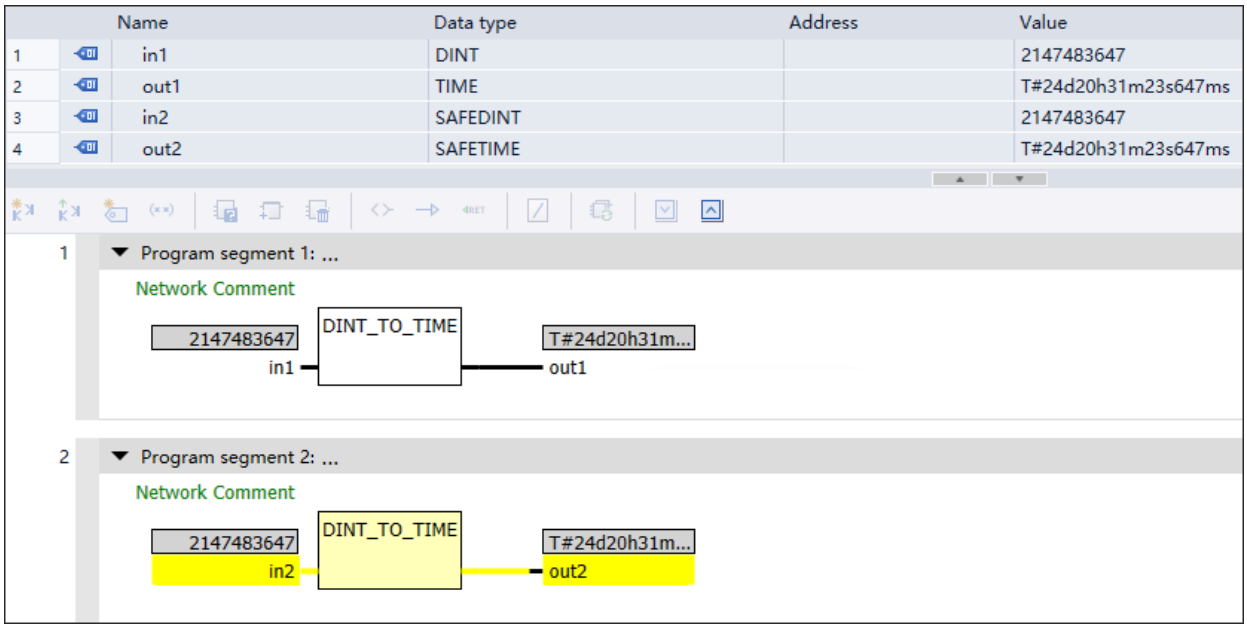
- **Function description**

  Convert "In" of DINT type to "Out" of TIME type.

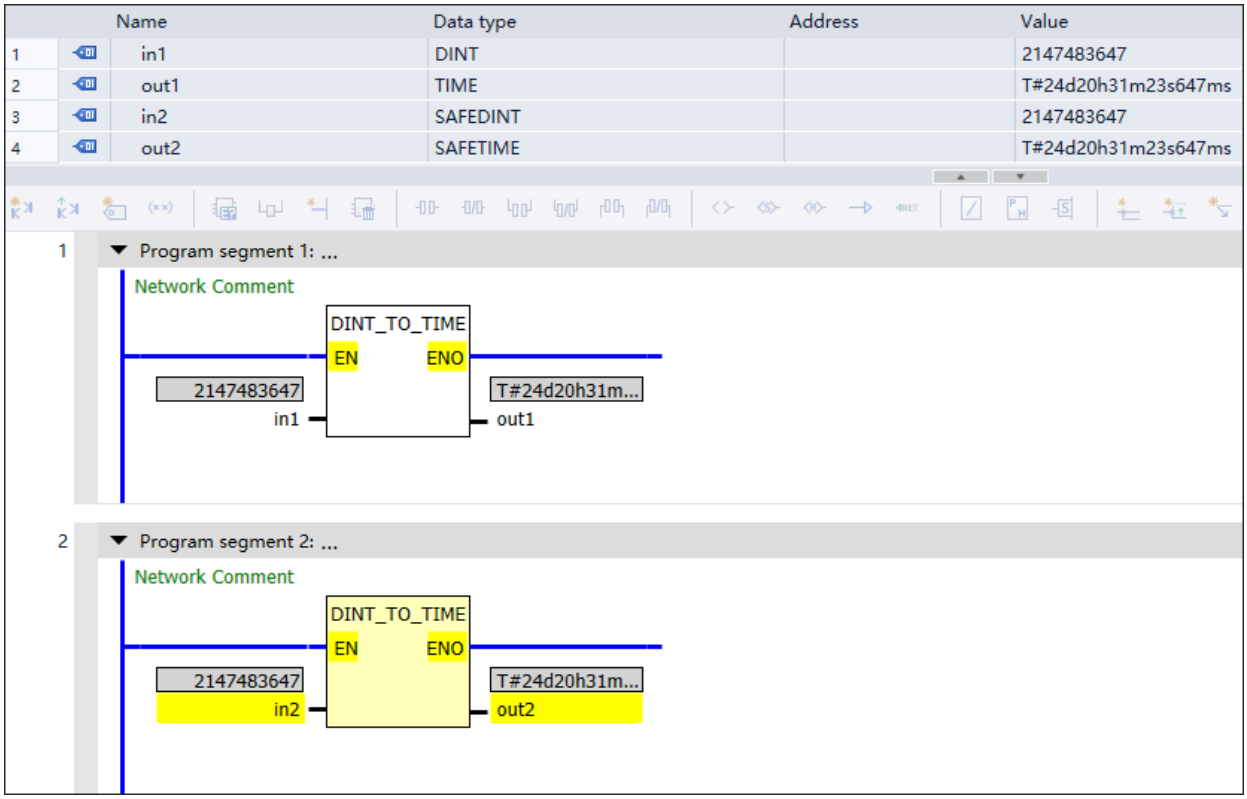
  Example: When the value of "In" is DINT#90090090, the value of "Out" is T#1d1h1m30s090ms.

- **Programming example**

  FBD:



  LD:

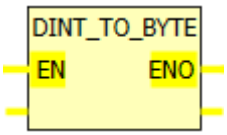
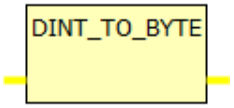


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - If a value outside the range of DINT#0 to 2147483647 is input, a "conversion exception" will occur and the program will stop.

11.10.1.6.12 DINT_TO_BYTE

Convert data of DINT type to BYTE type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
DINT_TO_BYTE	Convert DINT type to BYTE type	FC		

- **Related variables**

Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	DINT#0 to 255	DINT#0	Conversion object
Output	Out	Conversion result	BYTE#16#00 to FF	BYTE#16#00	Conversion result

Data types

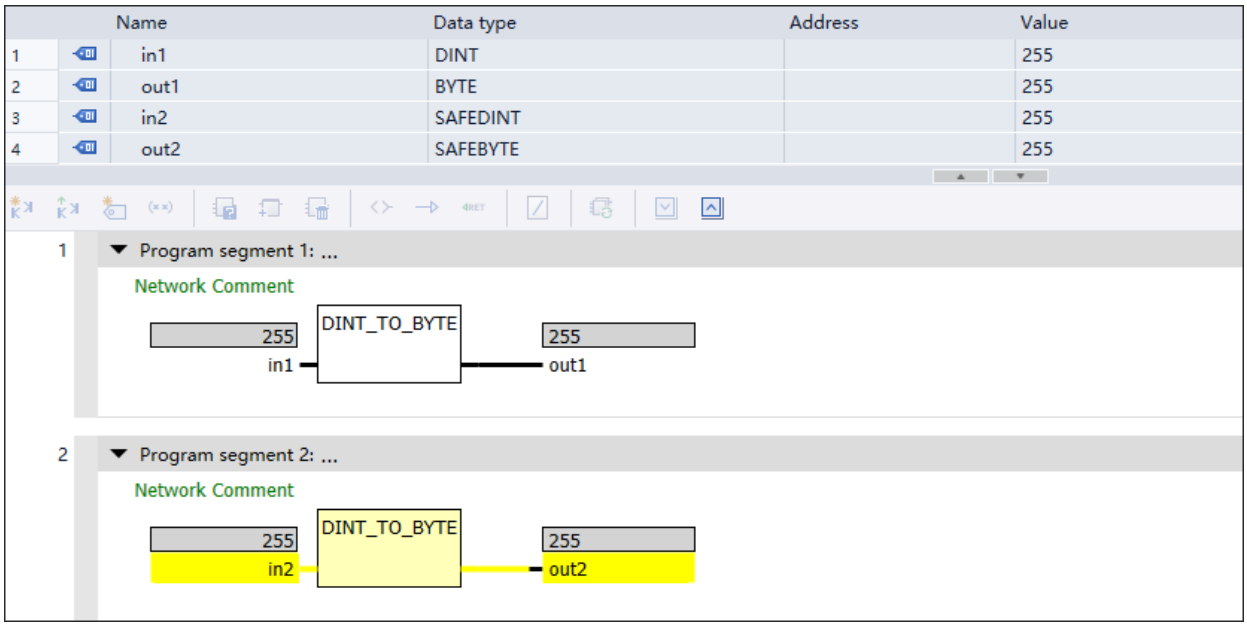
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	-	-	-	-	√	√	-	-
Out	-	-	√	√	-	-	-	-	-	-	-	-	-	-

- **Function description**

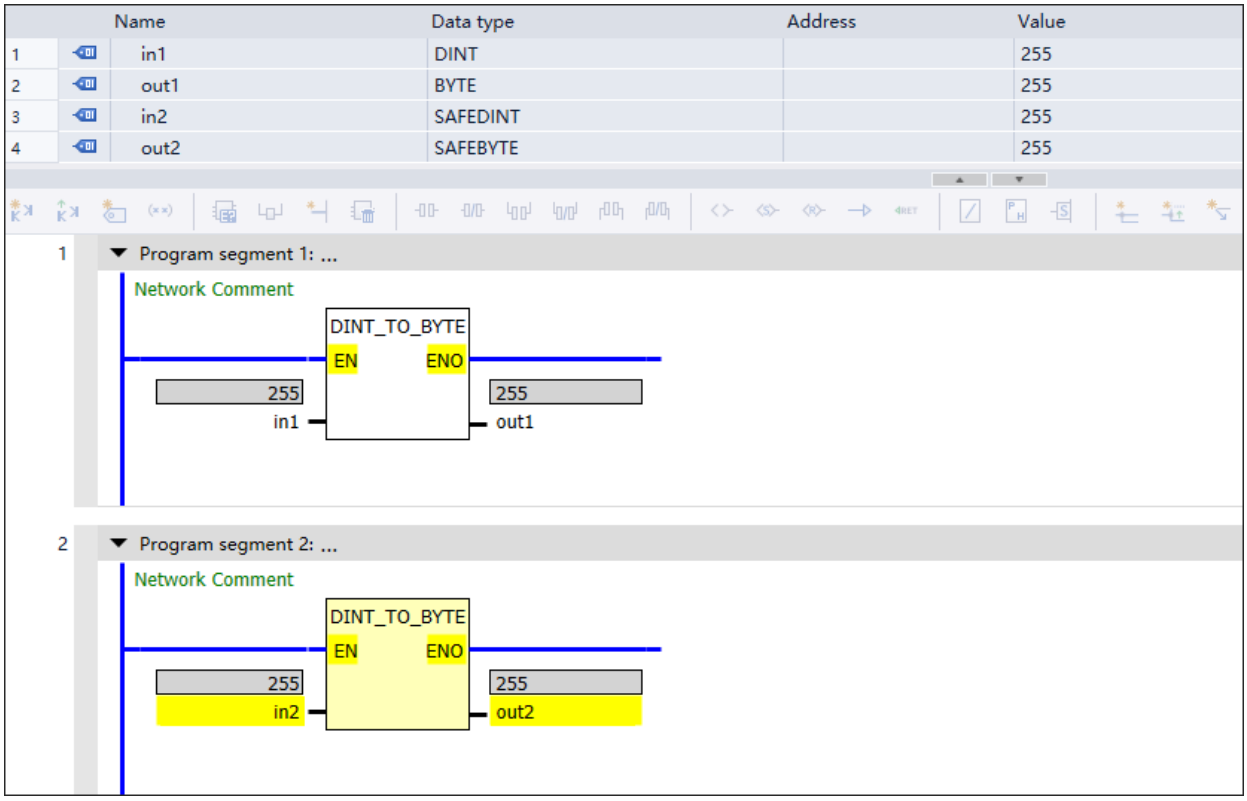
  Convert "In" of DINT type to the bit string "Out" of BYTE type.

- **Programming example**

  FBD:



  LD:

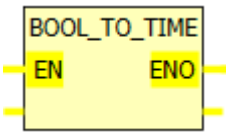
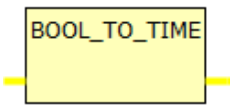


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - If a value outside the range of DINT#0 to 255 is input, a "conversion exception" will occur and the program will stop.

11.10.1.6.13 BOOL_TO_TIME

Convert data of BOOL type to TIME type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
BOOL_TO_TIME	Convert BOOL type to TIME type	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	FALSE, TRUE	FALSE	Conversion object
Output	Out	Conversion result	T#0ms, T#1ms	T#0ms	Conversion result

  Data types

	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	√	√	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	√	√

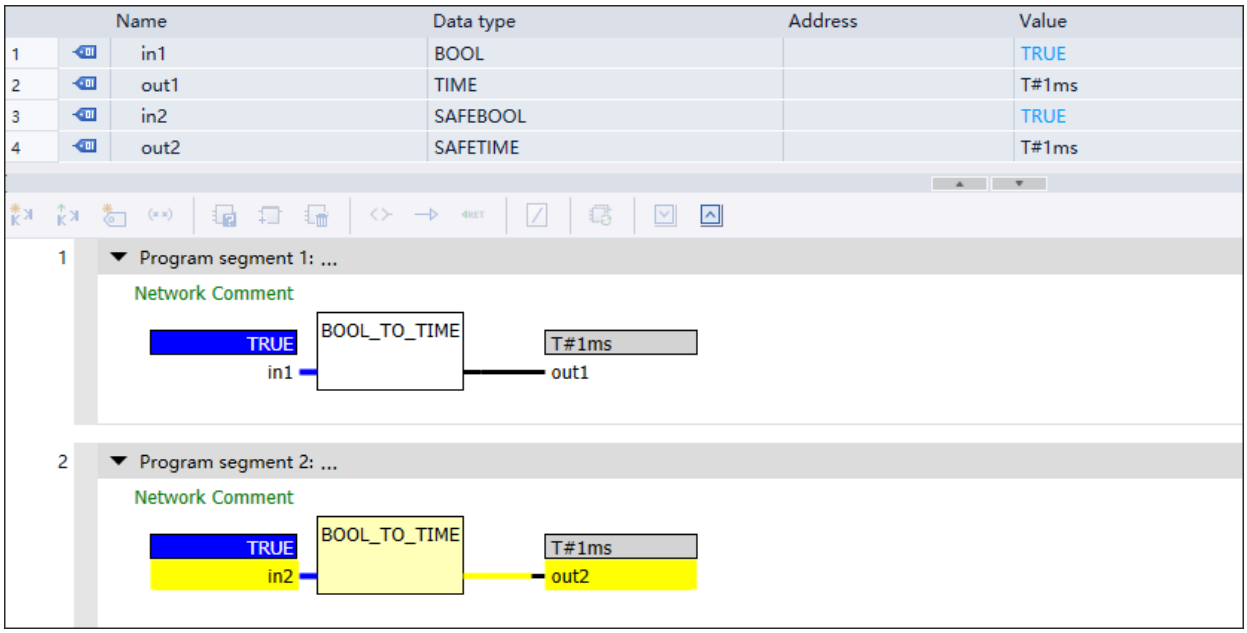
- **Function description**

  Convert the bit string "In" of BOOL type to "Out" of TIME type.

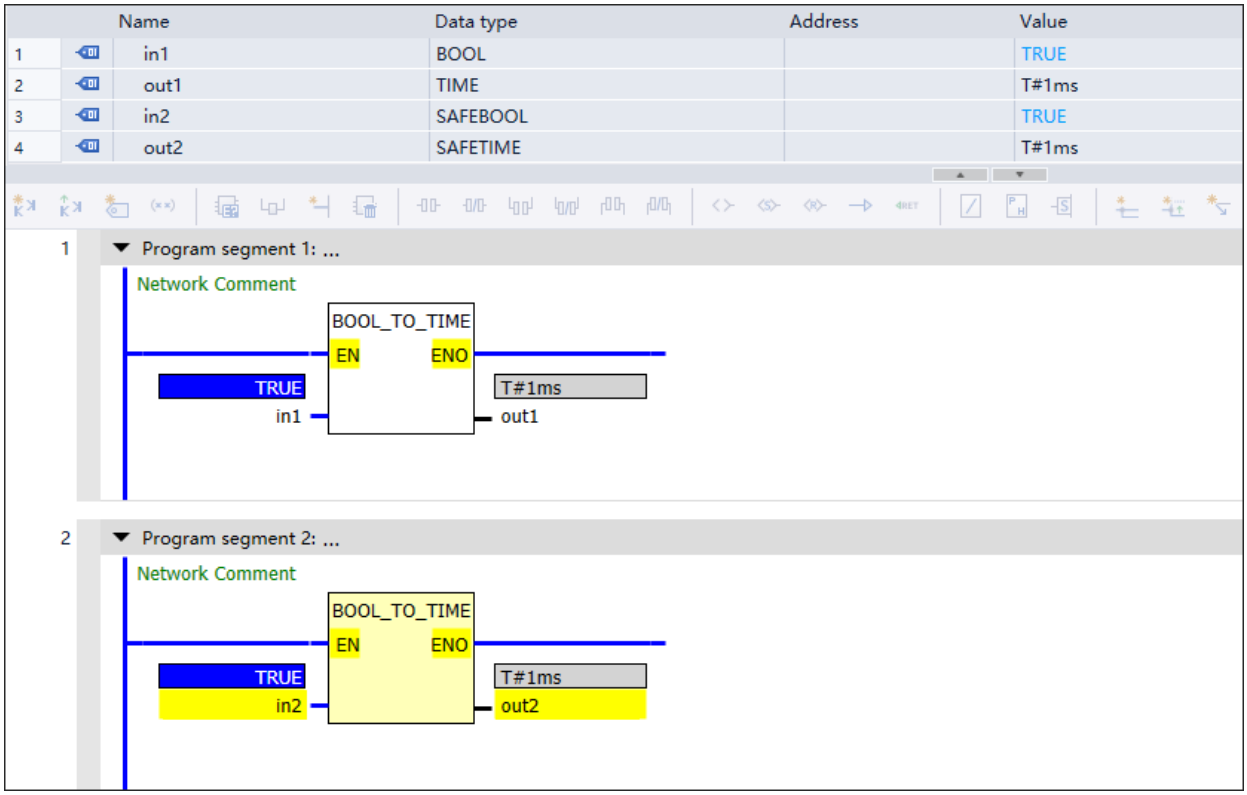
  - When the value of "In" is FALSE, the value of "Out" is T#0ms.
  - When the value of "In" is TRUE, the value of "Out" is T#1ms.

- **Programming example**

  FBD:



  LD:

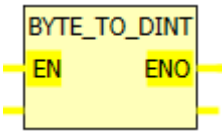
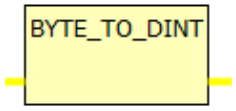


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.6.14 BYTE_TO_DINT

Convert data of BYTE type to DINT type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
BYTE_TO_DINT	Convert BYTE type to DINT type	FC		

- **Related variables**

Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	BYTE#16#00 to FF	BYTE#16#00	Conversion object
Output	Out	Conversion result	DINT#0 to 255	DINT#0	Conversion result

Data types

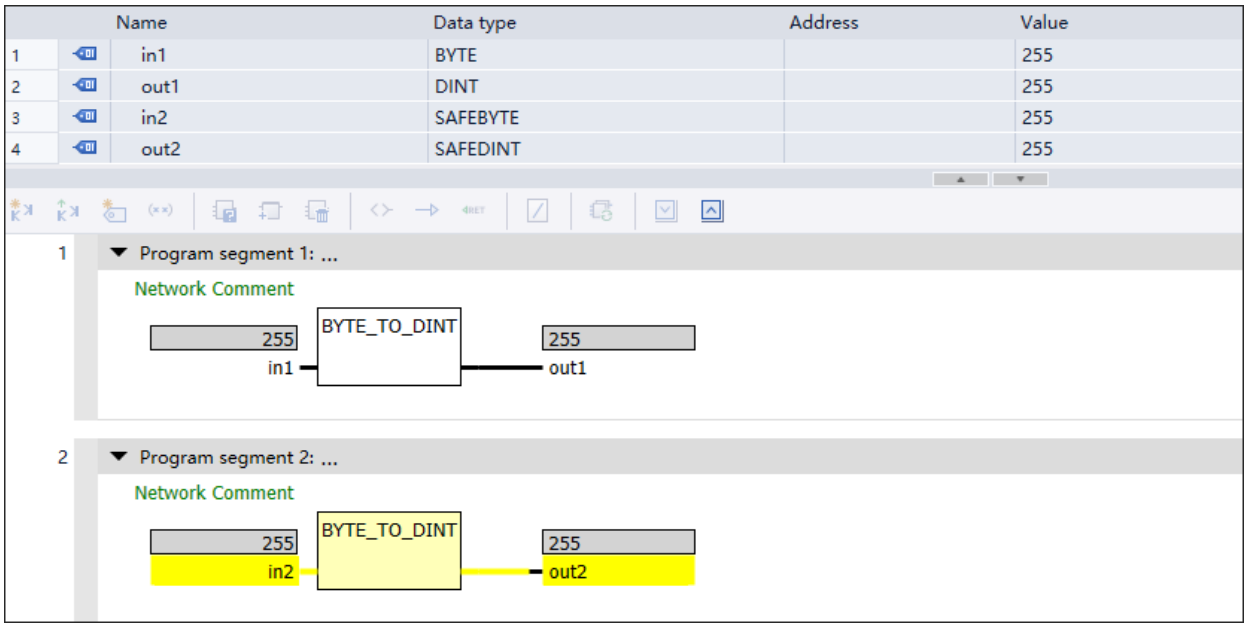
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	√	√	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	√	√	-	-

- **Function description**

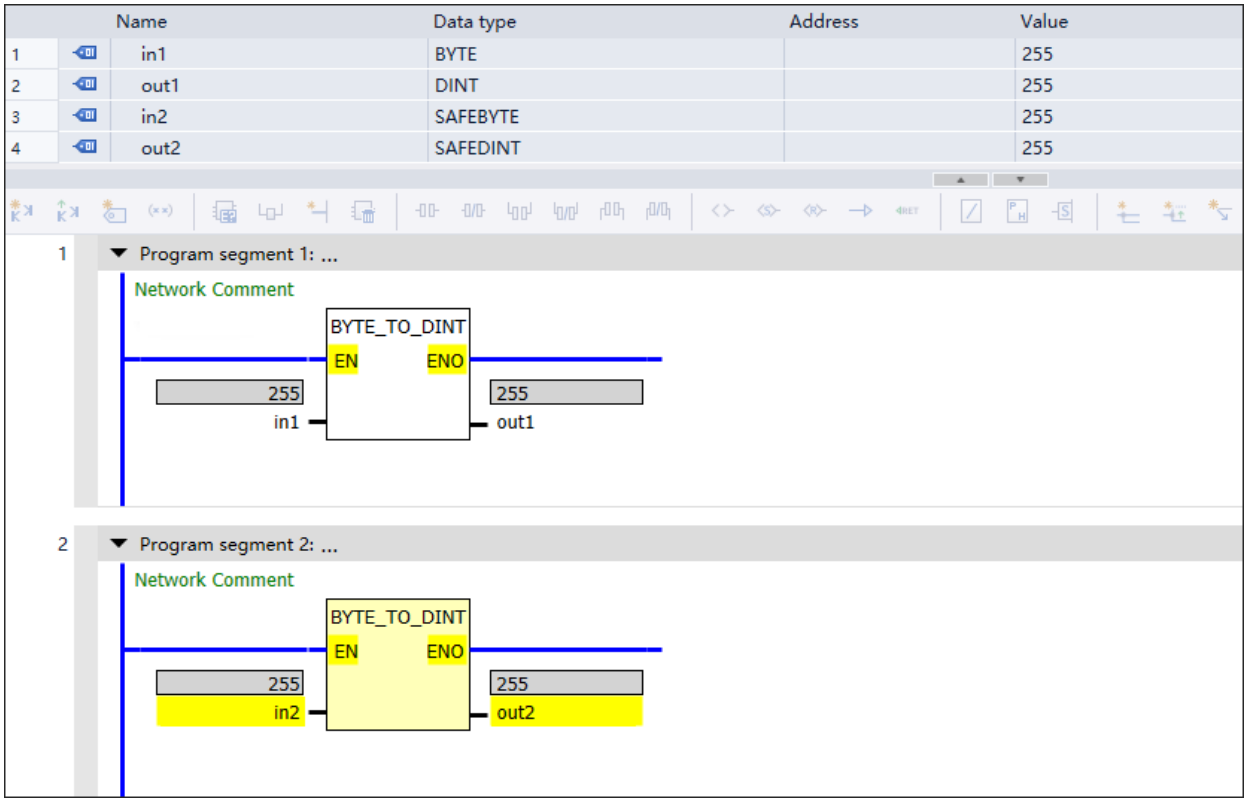
  Convert the bit string "In" of BYTE type to "Out" of DINT type.

- **Programming example**

  FBD:



  LD:

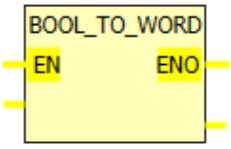
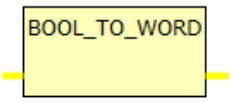


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.6.15 BOOL_TO_WORD

Convert data of BOOL type to WORD type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
BOOL_TO_WORD	Convert BOOL type to WORD type	FC		

- **Related variables**

  Variables

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	FALSE, TRUE	FALSE	Conversion object
Output	Out	Conversion result	WORD#16#0000, WORD#16#0001	WORD#16#0000	Conversion result

  Data types

	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	√	√	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	√	√	-	-	-	-	-	-	-	-

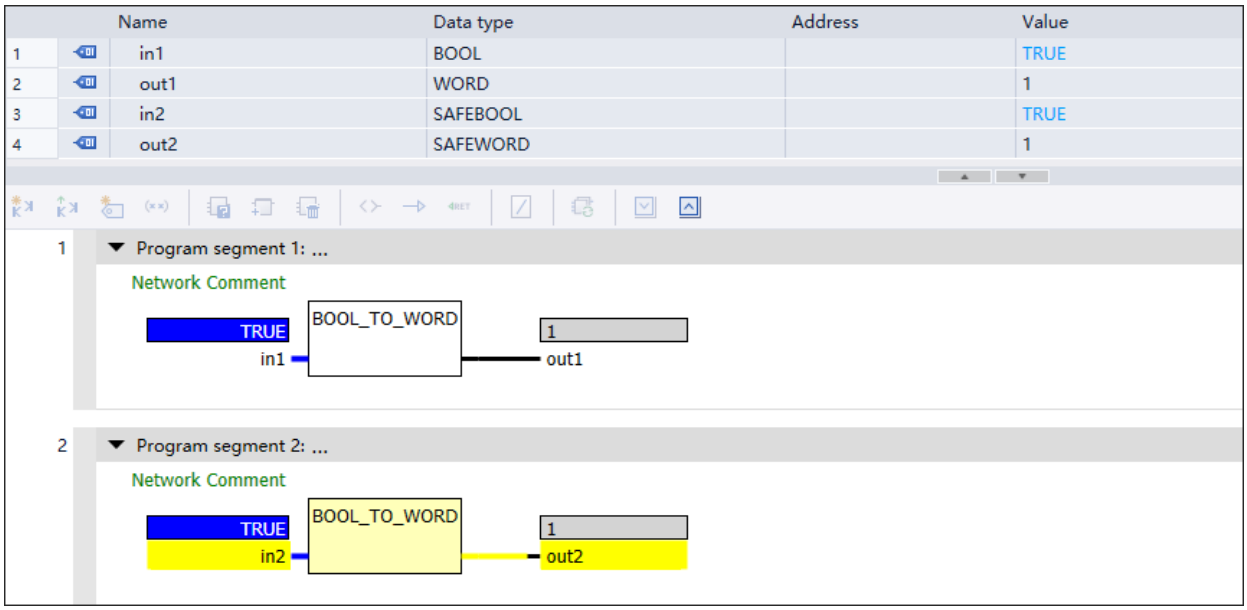
- **Function description**

  Convert "In" of BOOL type to the bit string "Out" of WORD type.

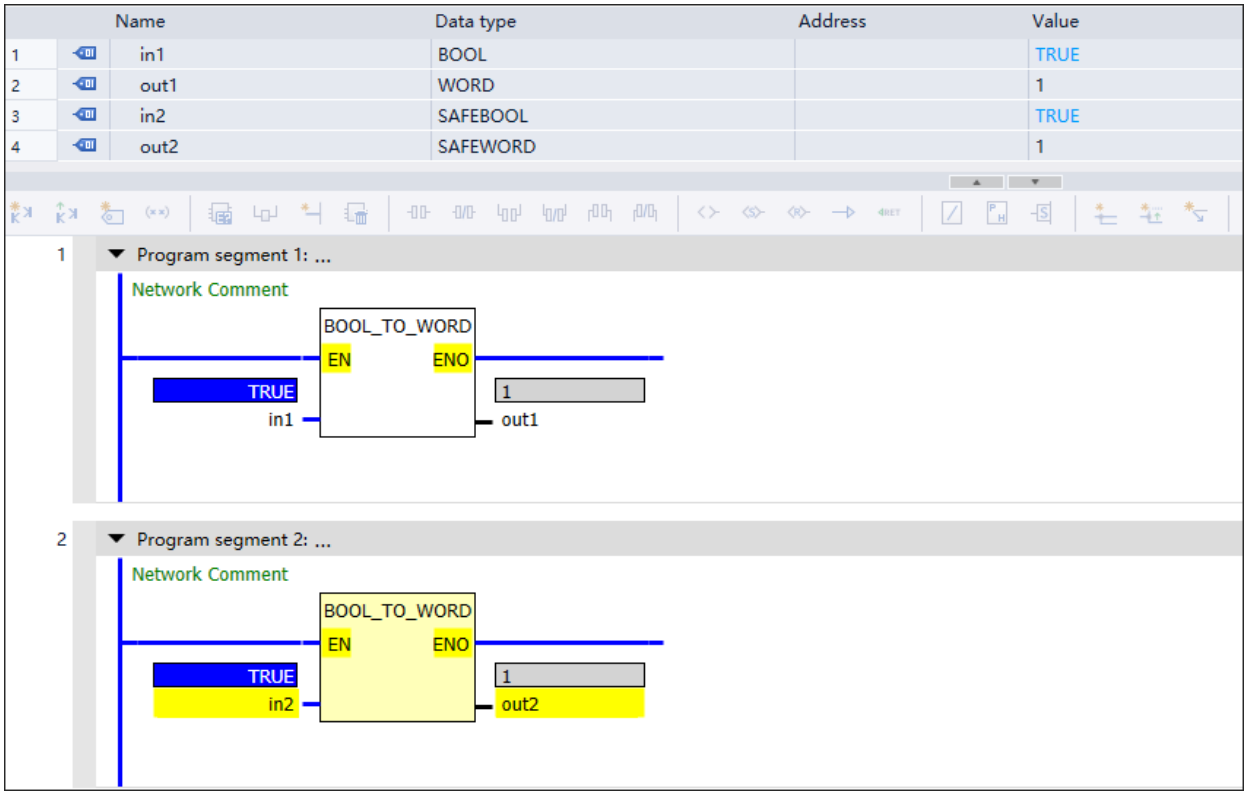
  - When the value of "In" is FALSE, the value of "Out" is WORD#16#0000.
  - When the value of "In" is TRUE, the value of "Out" is WORD#16#0001.

- **Programming example**

  FBD:



  LD:

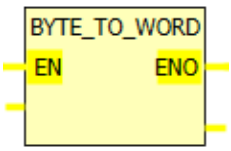
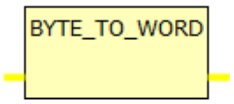


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.6.16 BYTE_TO_WORD

Convert data of BYTE type to WORD type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
BYTE_TO_WORD	Convert BYTE type to WORD type	FC		

- **Related variables**

Variables

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	BYTE#16#00 to FF	BYTE#16#00	Conversion object
Output	Out	Conversion result	WORD#16#0000 and WORD#16#00FF	WORD#16#0000	Conversion result

Data types

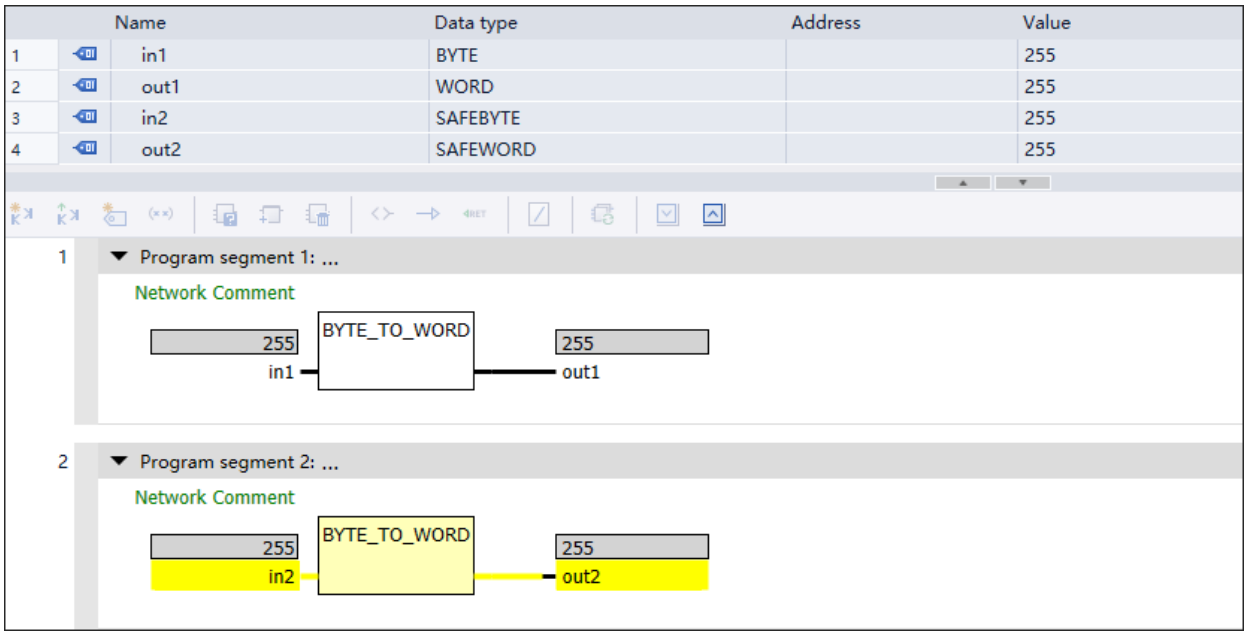
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	√	√	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	√	√	-	-	-	-	-	-	-	-

- **Function description**

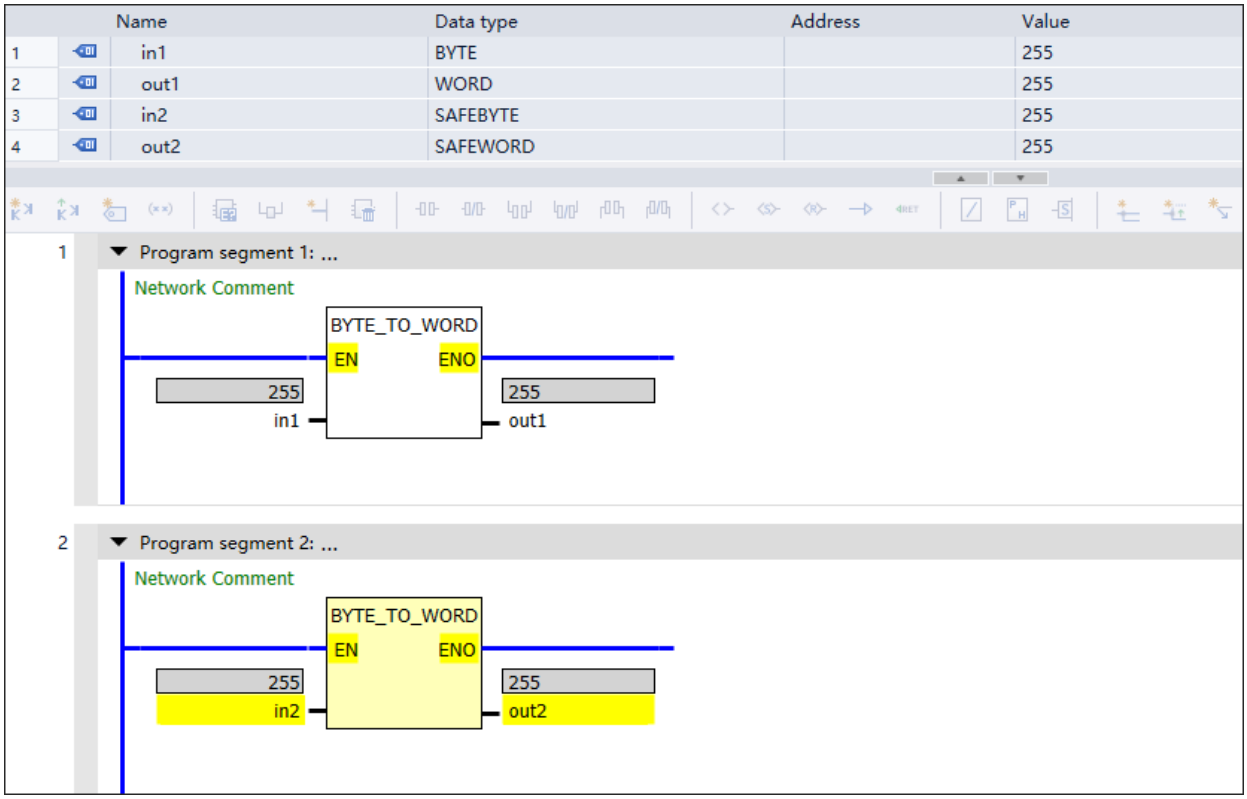
  Convert the bit string "In" of BYTE type to the bit string "Out" of WORD type.

- **Programming example**

  FBD:



  LD:

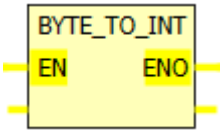
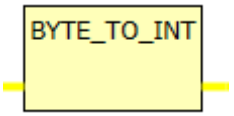


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.6.17 BYTE_TO_INT

Convert data of BYTE type to INT type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
BYTE_TO_INT	Convert BYTE type to INT type	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	BYTE#16#00 to FF	BYTE#16#00	Conversion object
Output	Out	Conversion result	INT#0 to 255	INT#0	Conversion result

  Data types

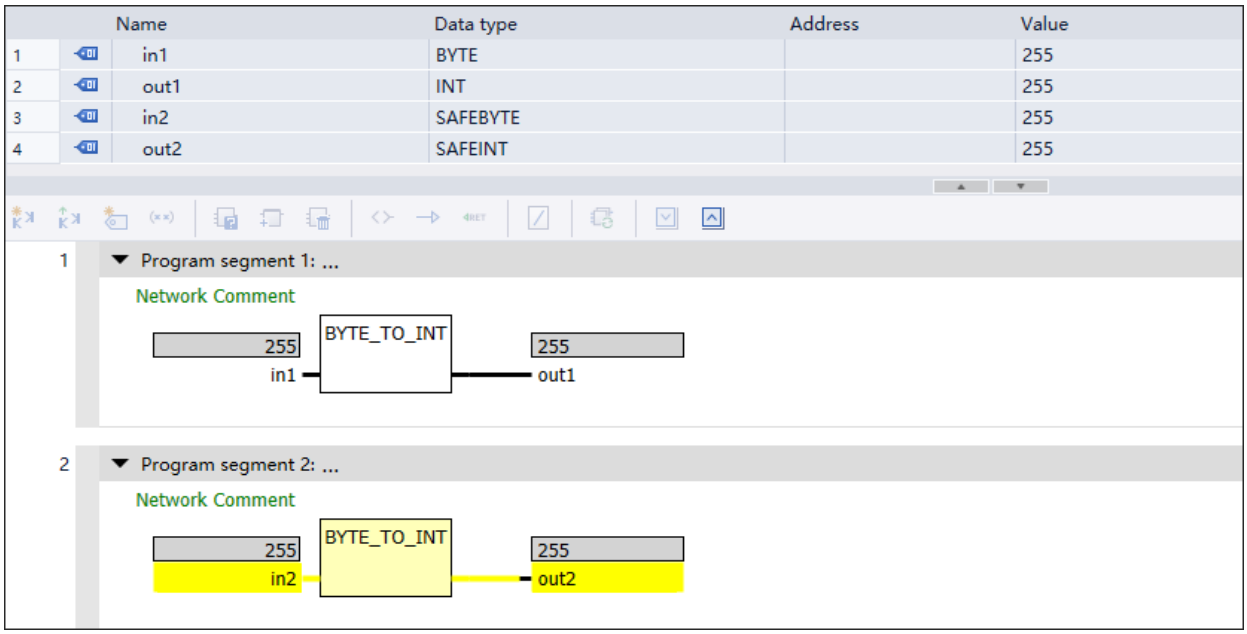
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	√	√	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	√	√	-	-	-	-

- **Function description**

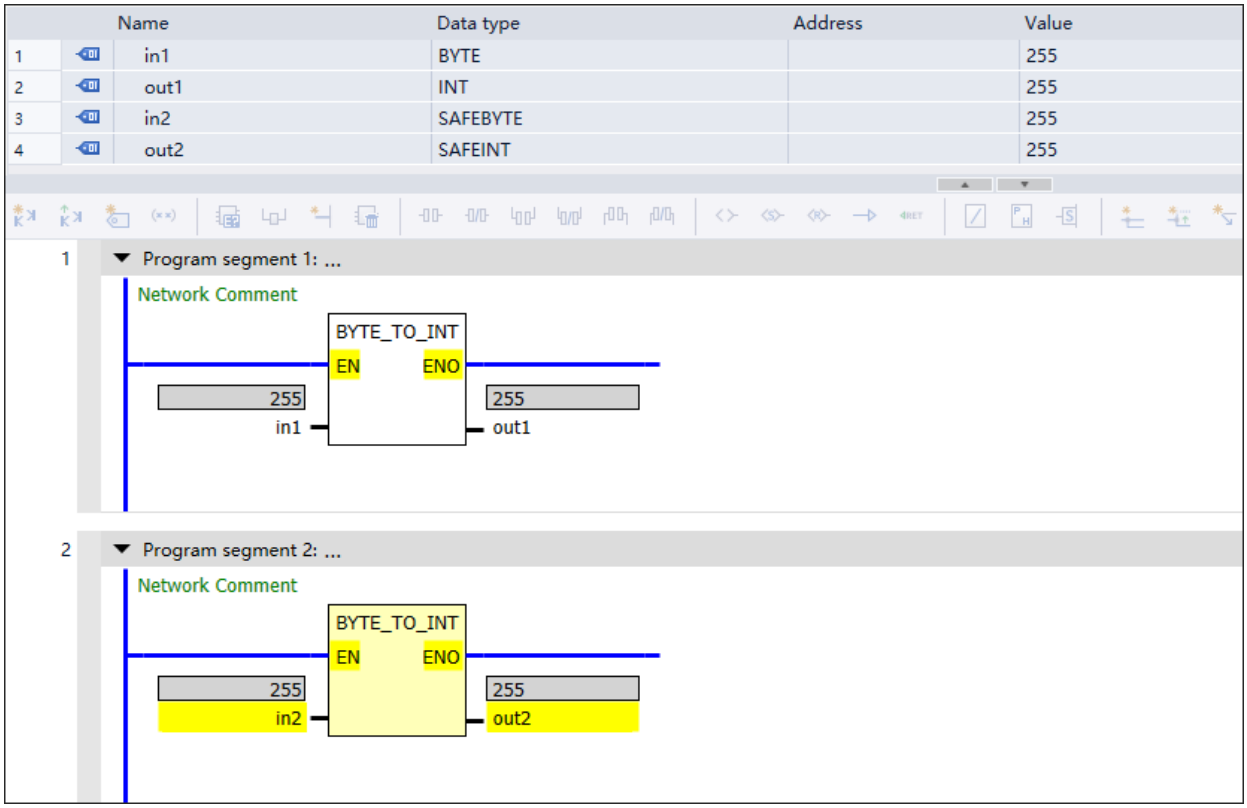
  Convert the bit string "In" of BYTE type to "Out" of INT type.

- **Programming example**

  FBD:



  LD:

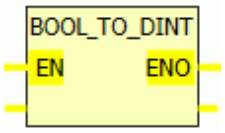
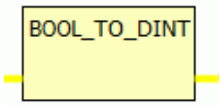


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.6.18 BOOL_TO_DINT

Convert data of BOOL type to DINT type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
BOOL_TO_DINT	Convert BOOL type to DINT type	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	FALSE, TRUE	FALSE	Conversion object
Output	Out	Conversion result	DINT#0 and DINT#1	DINT#0	Conversion result

  Data types

	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	√	√	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	√	√	-	-

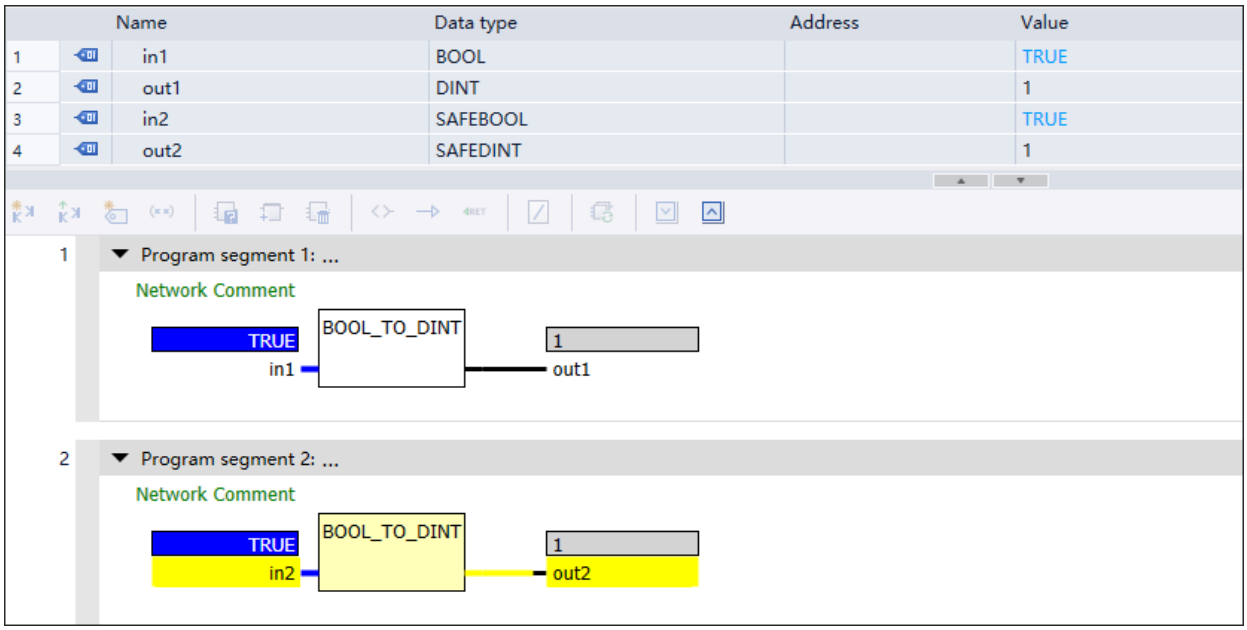
- **Function description**

  Convert "In" of BOOL type to "Out" of DINT type.

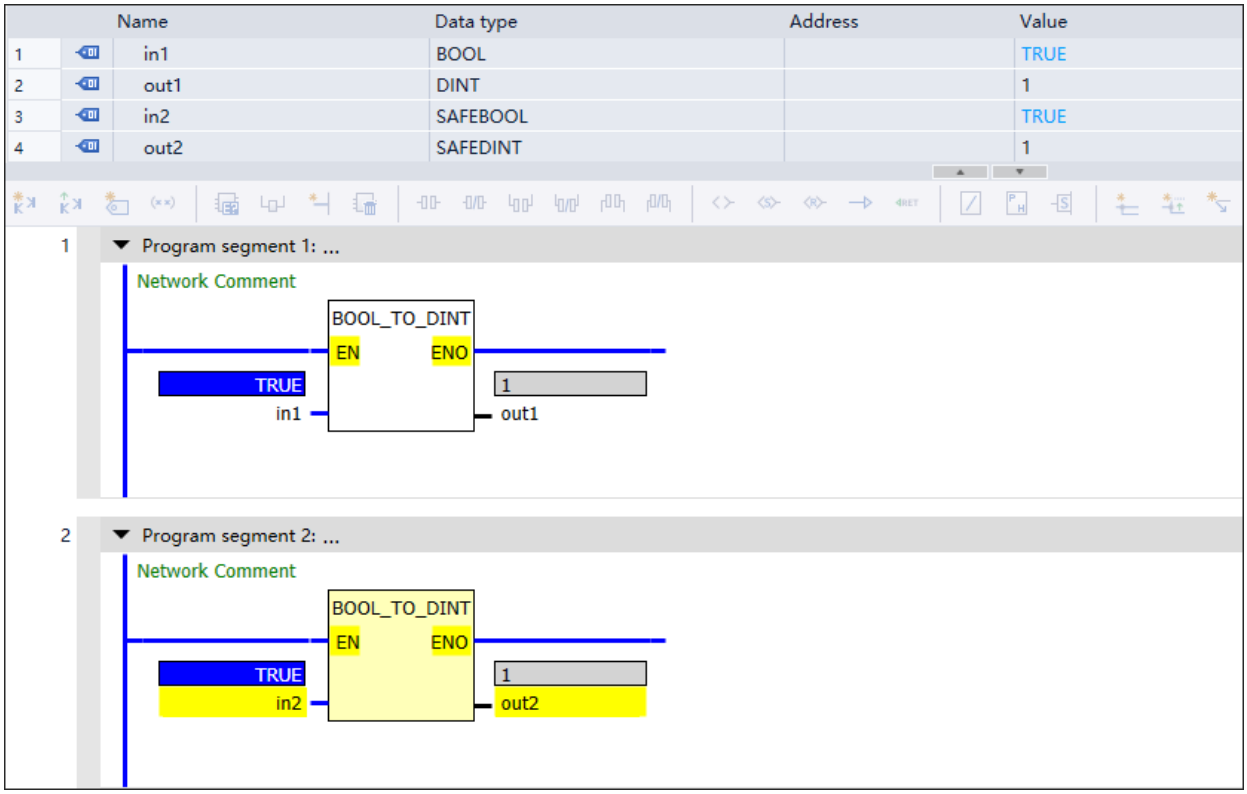
  - When the value of "In" is FALSE, the value of "Out" is DINT#0.
  - When the value of "In" is TRUE, the value of "Out" is DINT#1.

- **Programming example**

  FBD:



  LD:

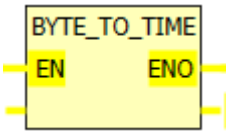
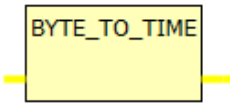


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.6.19 BYTE_TO_TIME

Convert data of BYTE type to TIME type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
BYTE_TO_TIME	Convert BYTE type to TIME type	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	BYTE#16#00 to FF	BYTE#16#00	Conversion object
Output	Out	Conversion result	T#0ms to T#255ms	T#0ms	Conversion result

  Data types

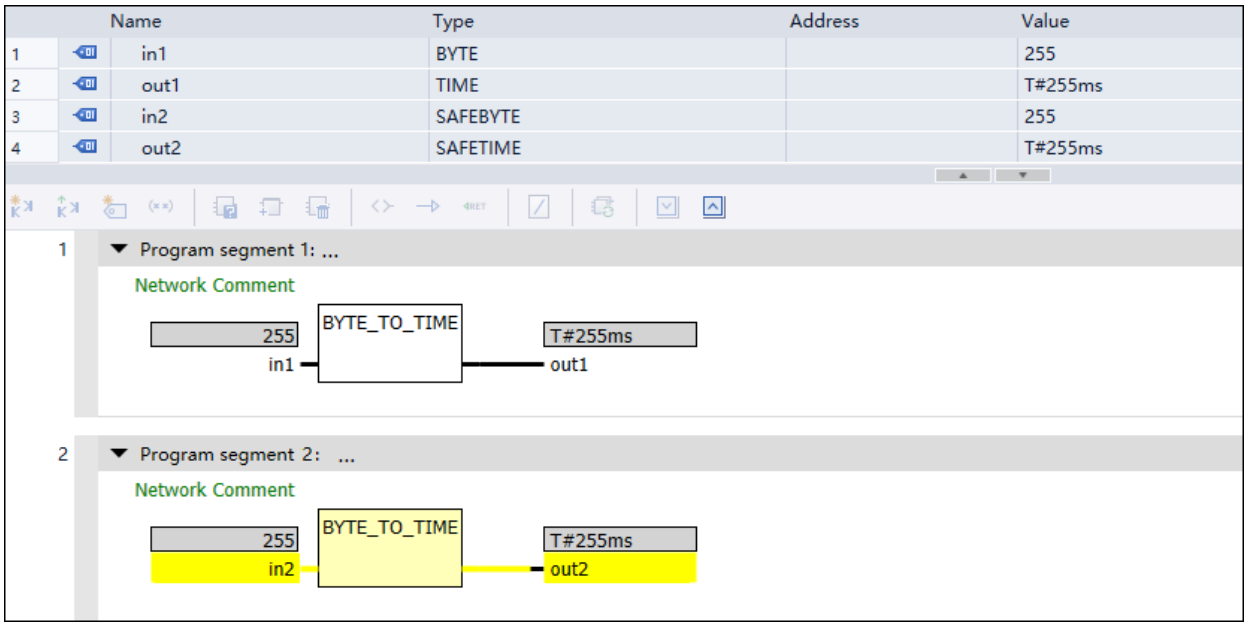
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	√	√	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	√	√

- **Function description**

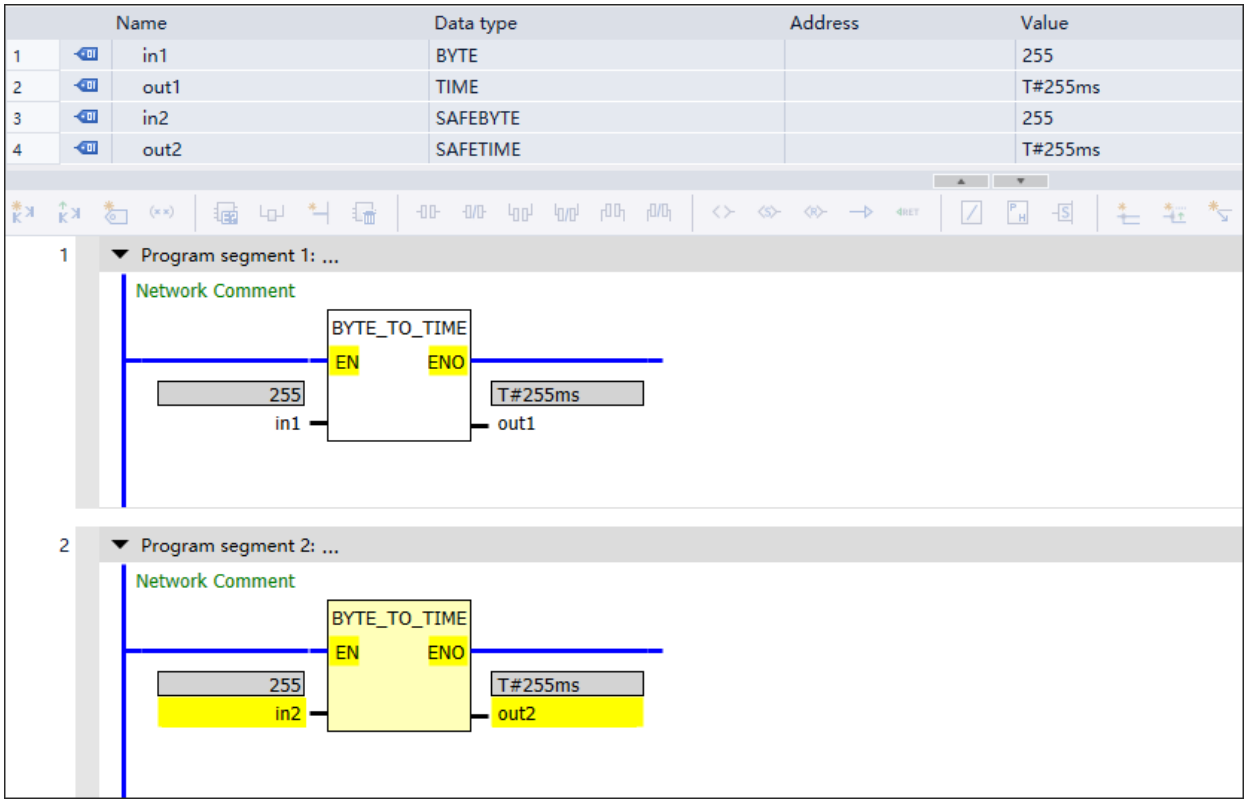
  Convert the bit string "In" of BYTE type to "Out" of TIME type.

- **Programming example**

  FBD:



  LD:

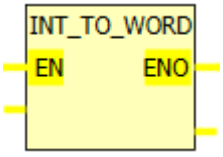
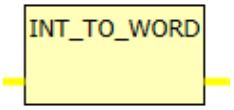


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.6.20 INT_TO_WORD

Convert data of INT type to WORD type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
INT_TO_WORD	Convert INT type to WORD type	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	INT#-32768 to 32767	INT#0	Conversion object
Output	Out	Conversion result	WORD#16#0000 to FFFF	WORD#16#0000	Conversion result

  Data types

	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	-	-	√	√	-	-	-	-
Out	-	-	-	-	√	√	-	-	-	-	-	-	-	-

- **Function description**

  Convert "In" of INT type to the bit string "Out" of WORD type.

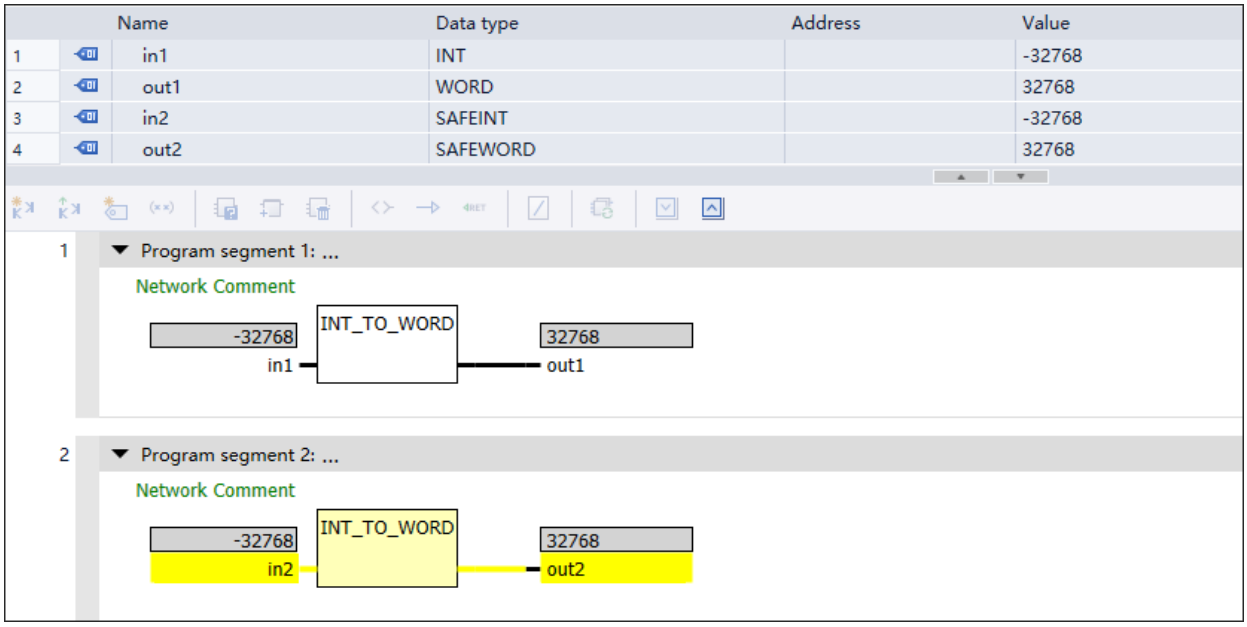
  - Example when the value of "In" is positive (INT#0 to 32767):

    When the value of "In" is INT#0 to 32767, the value of "Out" is WORD#16#0000 to 7FFF.

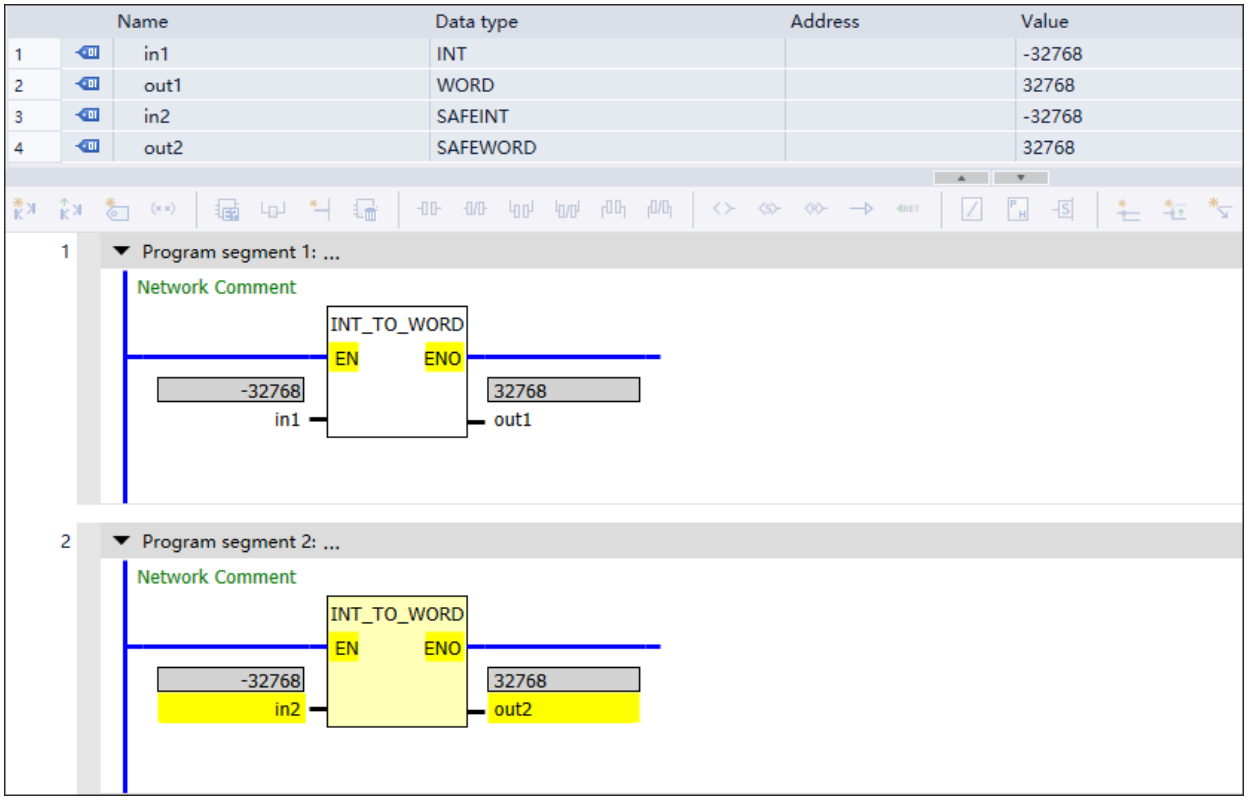
  - Example when the value of "In" is negative (INT#-32768 to -1):
    - When the value of "In" is INT#-32768, which is 1000 0000 0000 0000 (BIN), the value of "Out" is WORD#16#8000.
    - When the value of "In" is INT#-1, which is 1111 1111 1111 1111 (BIN), the value of "Out" is WORD#16#FFFF.

- **Programming example**

  FBD:



  LD:

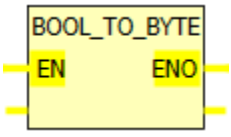
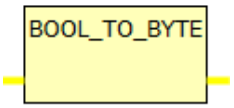


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.6.21 BOOL_TO_BYTE

Convert data of BOOL type to BYTE type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
BOOL_TO_BYTE	Convert BOOL type to BYTE type	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	FALSE, TRUE	FALSE	Conversion object
Output	Out	Conversion result	BYTE#0 and BYTE#1	BYTE#0	Conversion result

  Data types

	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	√	√	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	√	√	-	-	-	-	-	-	-	-	-	-

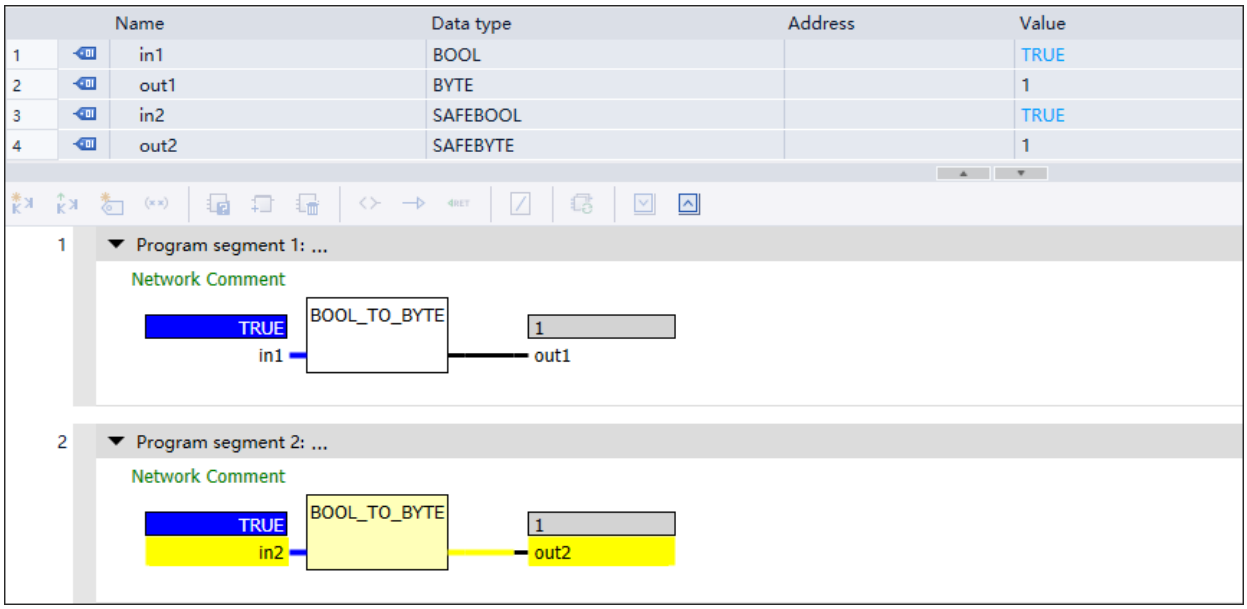
- **Function**

  Convert "In" of BOOL type to the bit string "Out" of BYTE type.

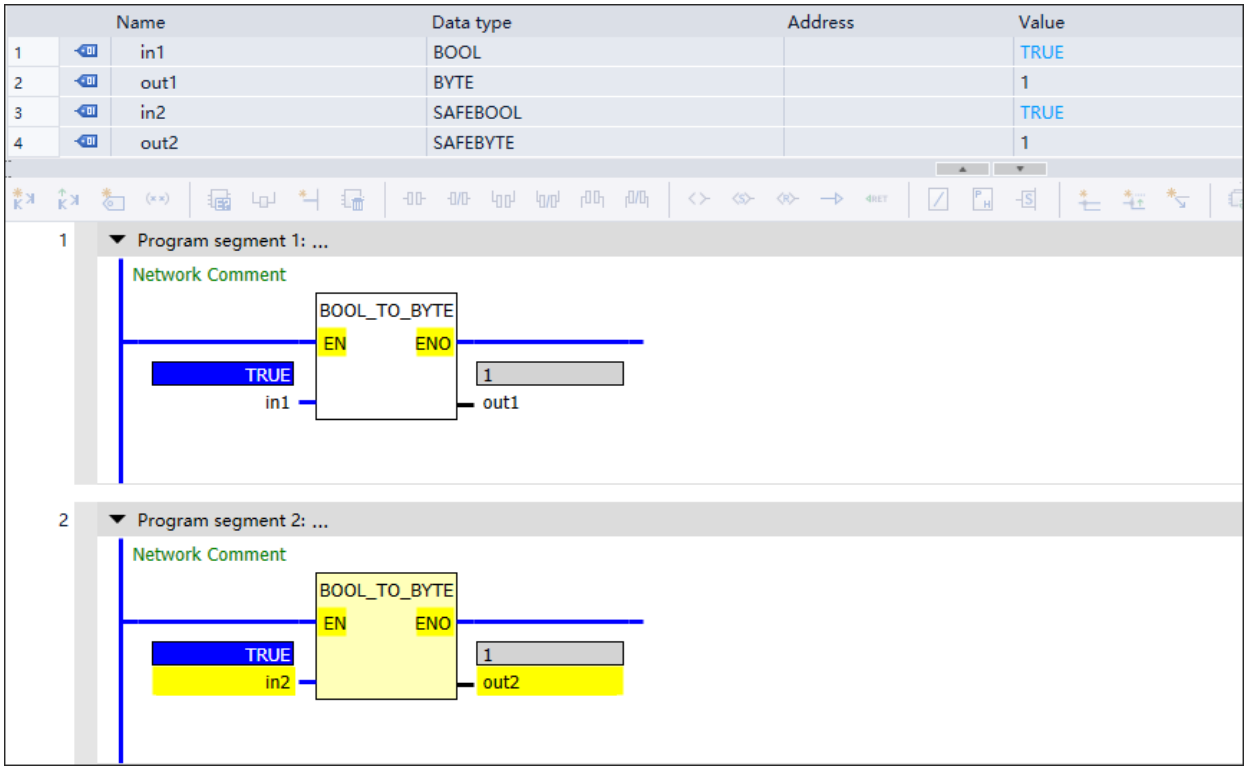
  - When the value of "In" is FALSE, the value of "Out" is BYTE#0.
  - When the value of "In" is TRUE, the value of "Out" is BYTE#1.

- **Programming example**

  FBD:



  LD:

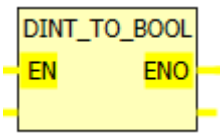
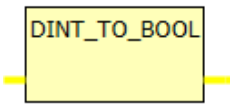


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.6.22 DINT_TO_BOOL

Convert data of DINT type to BOOL type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
DINT_TO_BOOL	Convert DINT type to BOOL type	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	DINT#-2147483648 to 2147483647	DINT#0	Conversion object
Output	Out	Conversion result	FALSE, TRUE	FALSE	Conversion result

  Data types

	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	-	-	-	-	√	√	-	-
Out	√	√	-	-	-	-	-	-	-	-	-	-	-	-

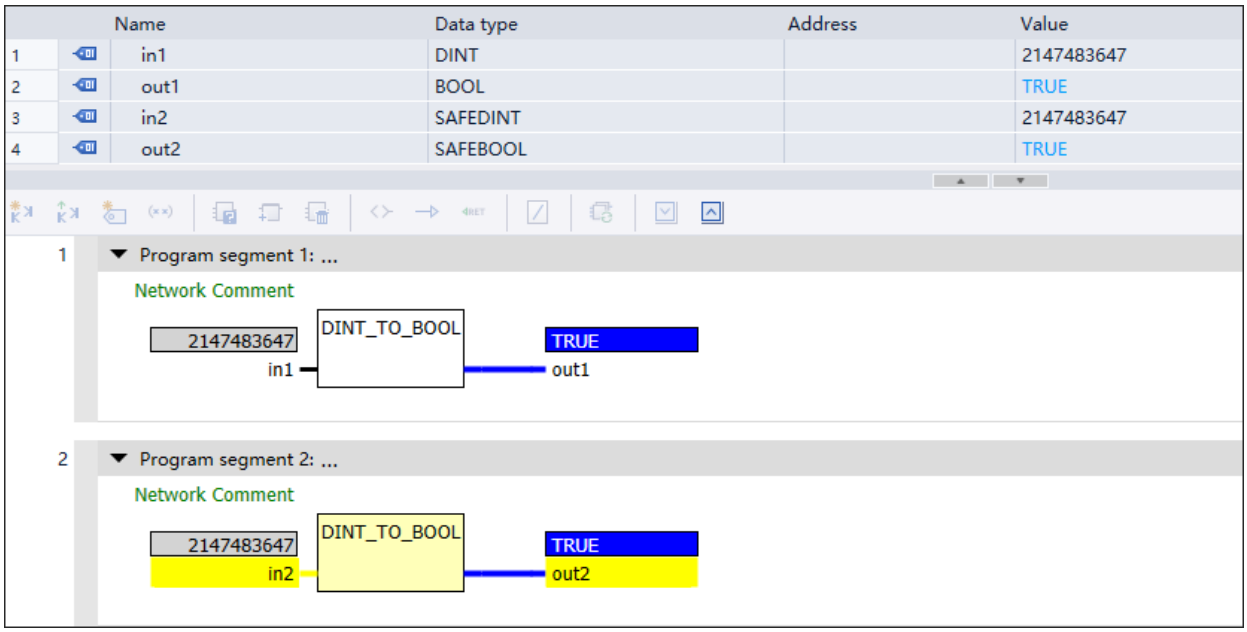
- **Function description**

  Convert "In" of DINT type to "Out" of BOOL type.

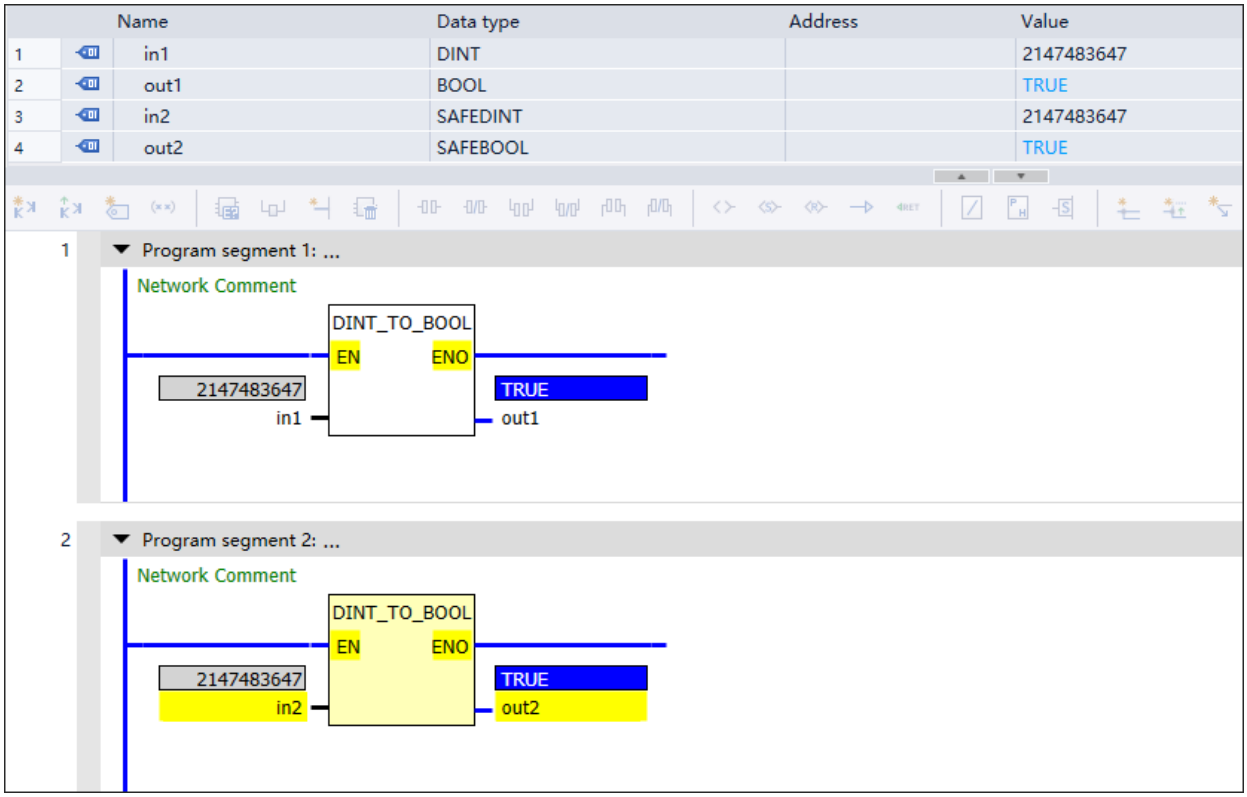
  - When the value of "In" is DINT#0, the value of "Out" is FALSE.
  - When the value of "In" is DINT#-2147483648 to -1 and DINT#1 to 2147483647 (excluding DINT#0), the value of "Out" is TRUE.

- **Programming example**

  FBD:



  LD:

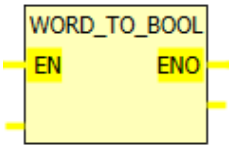
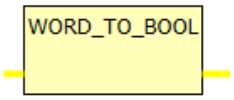


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - If a value outside the range of DINT#-2147483648 to 2147483647 is input, a "conversion exception" will occur and the program will stop.

11.10.1.6.23 WORD_TO_BOOL

Convert data of WORD type to BOOL type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
WORD_TO_BOOL	Convert WORD type data to BOOL type	FC		

- **Related variables**

Variables

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	WORD#16#0000 to FFFF	WORD#16#0000	Conversion object
Output	Out	Conversion result	FALSE, TRUE	FALSE	Conversion result

Data types

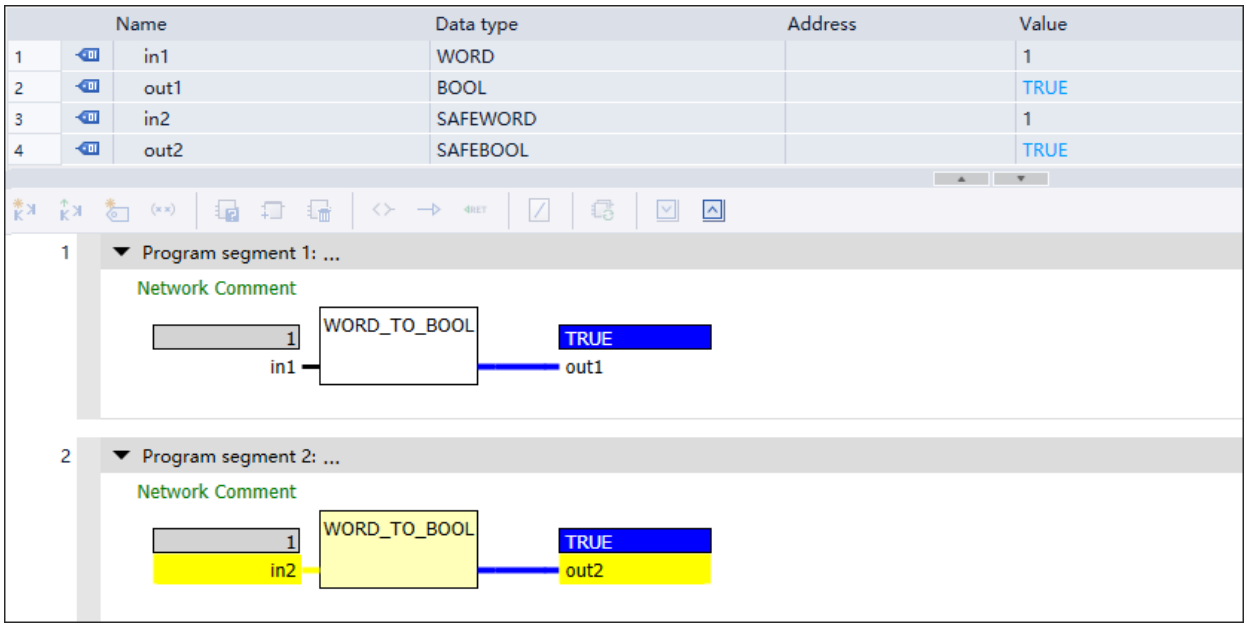
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	√	√	-	-	-	-	-	-	-	-
Out	√	√	-	-	-	-	-	-	-	-	-	-	-	-

- **Function description**

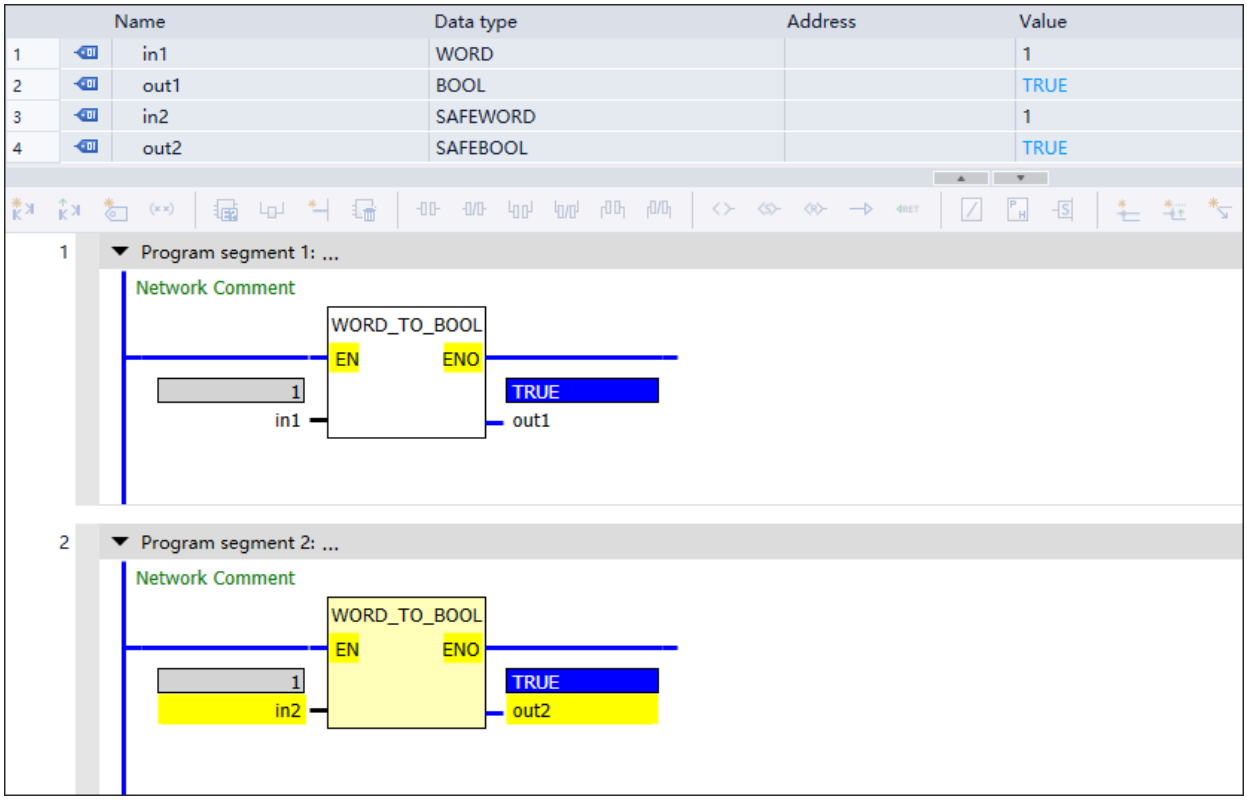
  Convert the bit string "In" of WORD type to "Out" of BOOL type.

- **Programming example**

  FBD:



  LD:

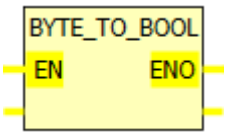
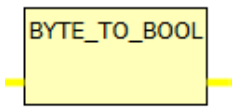


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - If a value outside the range of WORD#0 to 1 is input, a "conversion exception" will occur and the program will stop.

11.10.1.6.24 BYTE_TO_BOOL

Convert data of BYTE type to BOOL type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
BYTE_TO_BOOL	Convert BYTE type data to BOOL type	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	BYTE#16#00 to FF	BYTE#16#00	Conversion object
Output	Out	Conversion result	FALSE, TRUE	FALSE	Conversion result

  Data types

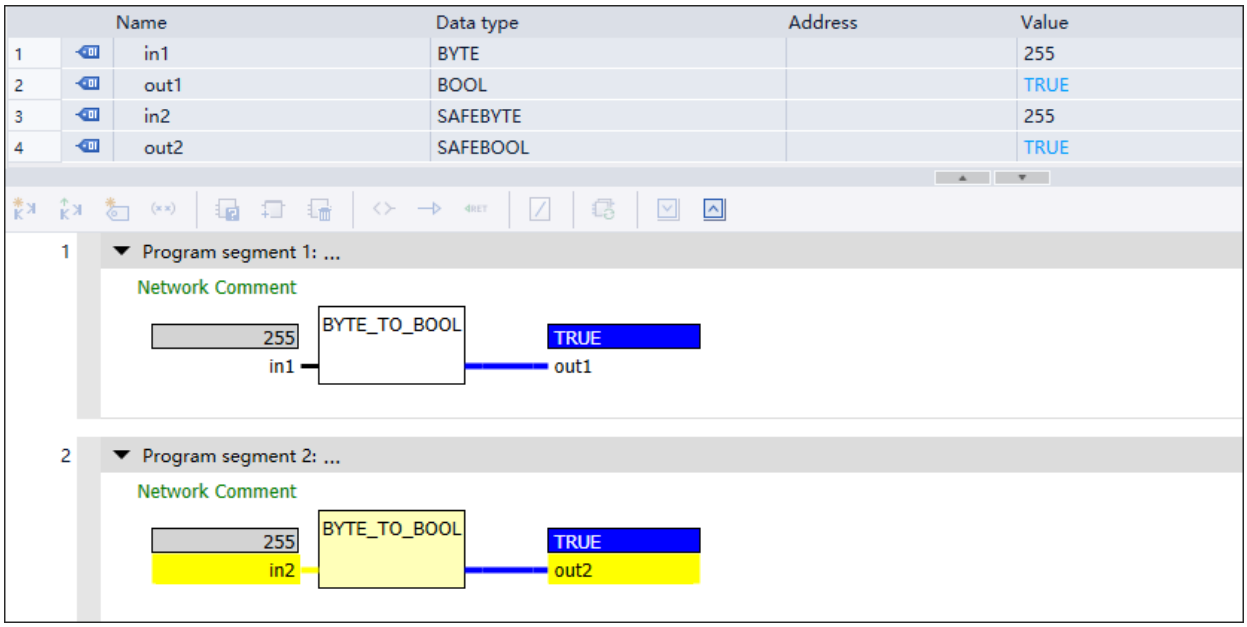
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	√	√	-	-	-	-	-	-	-	-	-	-
Out	√	√	-	-	-	-	-	-	-	-	-	-	-	-

- **Function description**

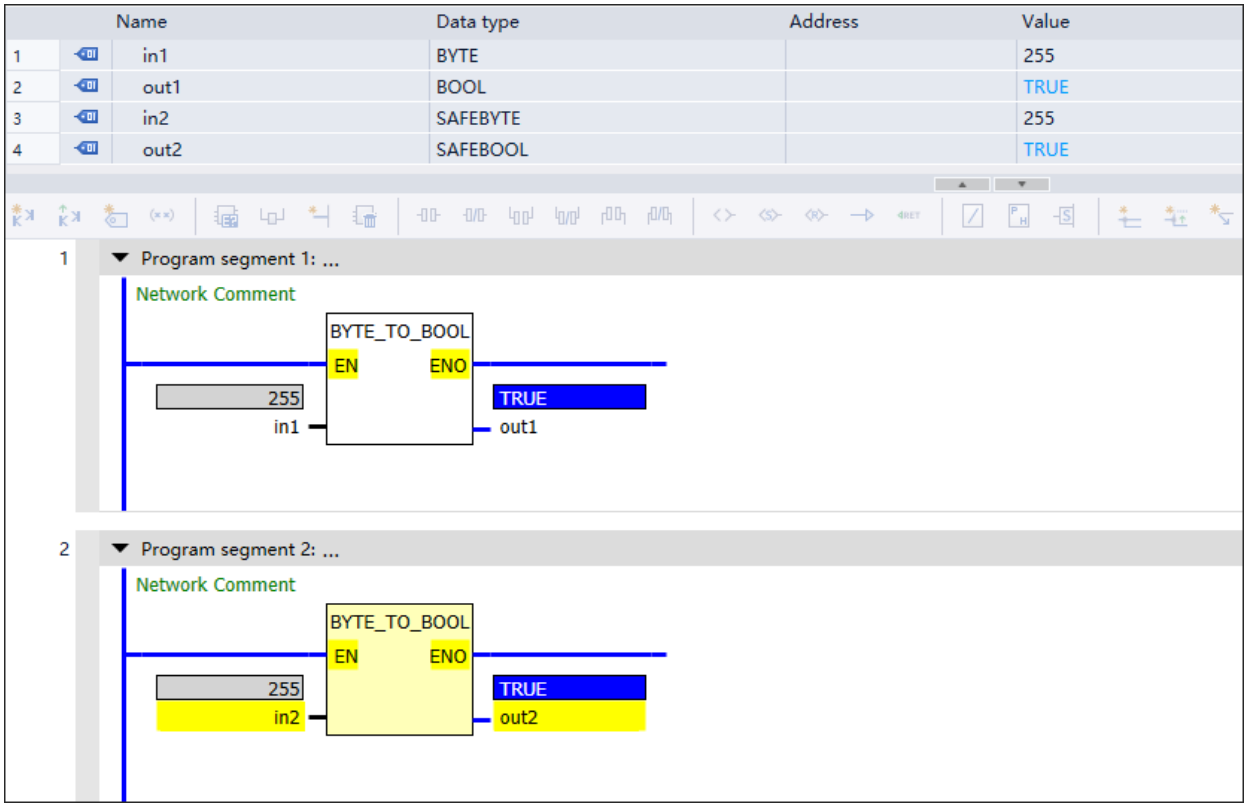
  Convert the bit string "In" of BYTE type to "Out" of BOOL type.

- **Programming example**

  FBD:



  LD:

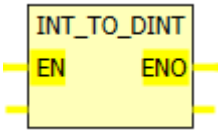
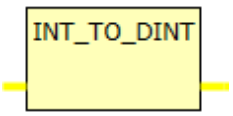


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.6.25 INT_TO_DINT

Convert data of INT type to DINT type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
INT_TO_DINT	Convert INT type data to DINT type	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	INT#-32768 to 32767	INT#0	Conversion object
Output	Out	Conversion result	DINT#-32768 to 32767	DINT#0	Conversion result

  Data types

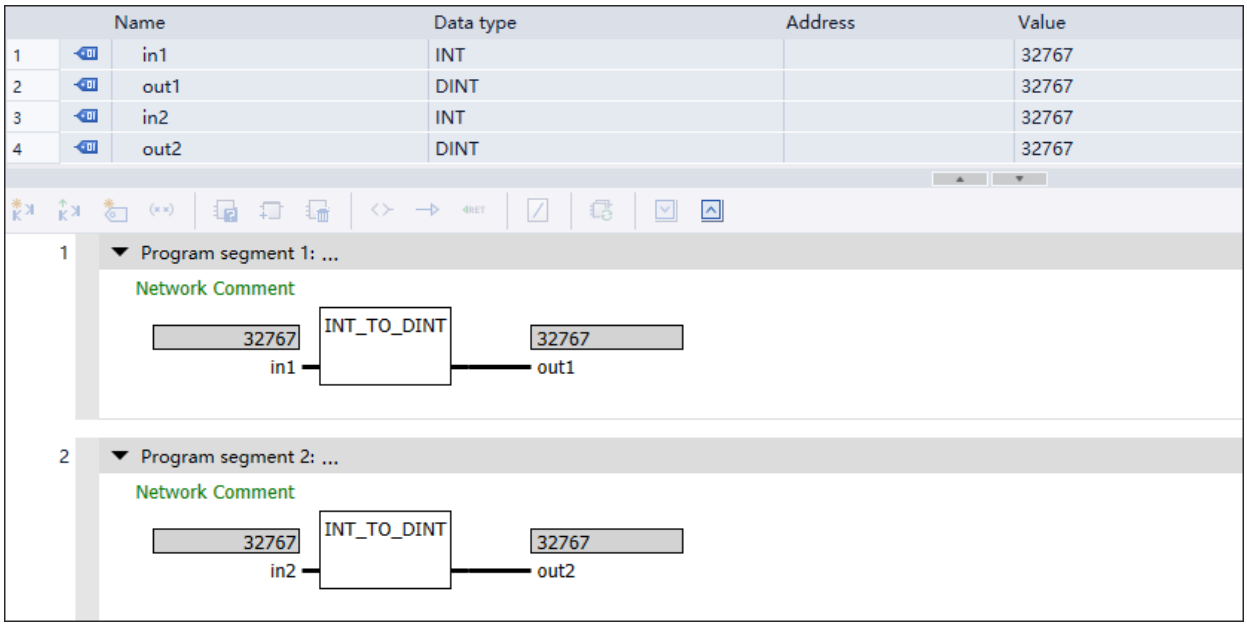
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	-	-	√	√	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	√	√	-	-

- **Function description**

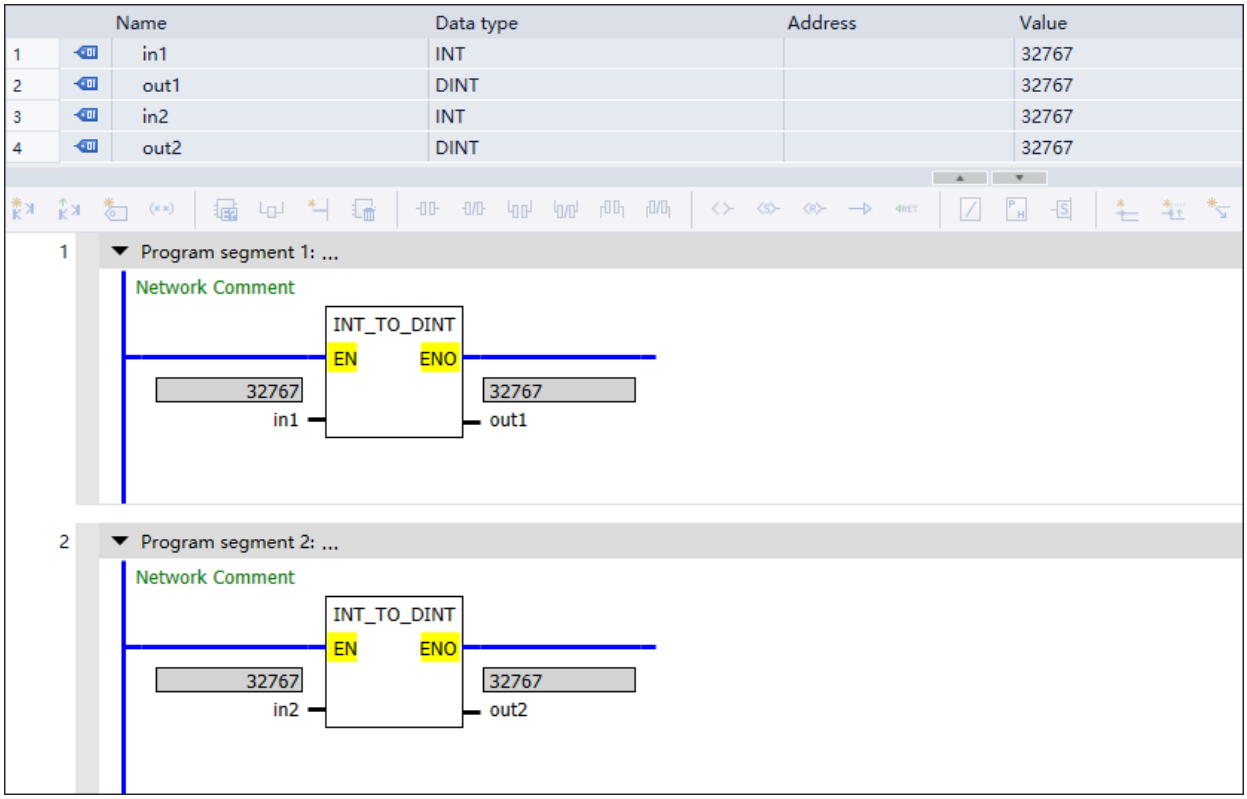
  Convert "In" of INT type to "Out" of DINT type.

- **Programming example**

  FBD:



  LD:

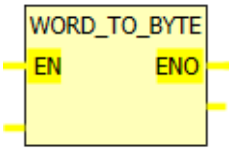
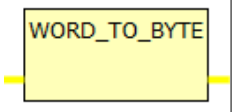


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.6.26 WORD_TO_BYTE

Convert data of WORD type to BYTE type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
WORD_TO_BYTE	Convert WORD type data to BYTE type	FC		

- **Related variables**

Variables

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	WORD#16#0000 to WORD#16#00FF	WORD#16#0000	Conversion object
Output	Out	Conversion result	BYTE#16#00 to FF	BYTE#16#00	Conversion result

Data types

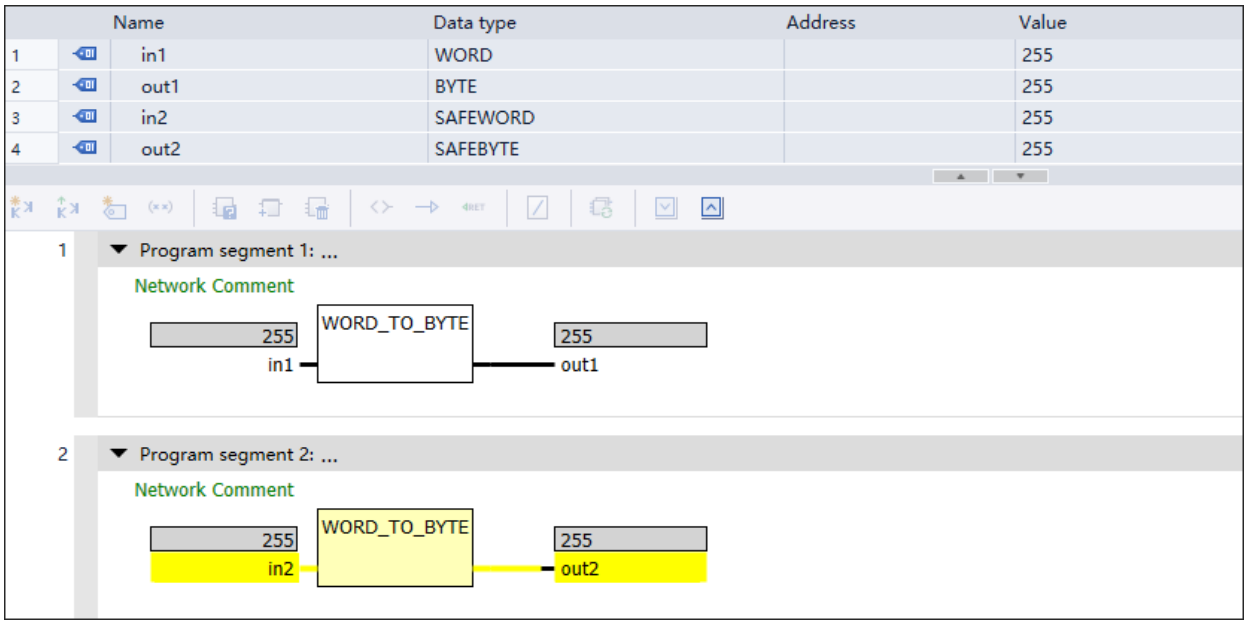
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	√	√	-	-	-	-	-	-	-	-
Out	-	-	√	√	-	-	-	-	-	-	-	-	-	-

- **Function description**

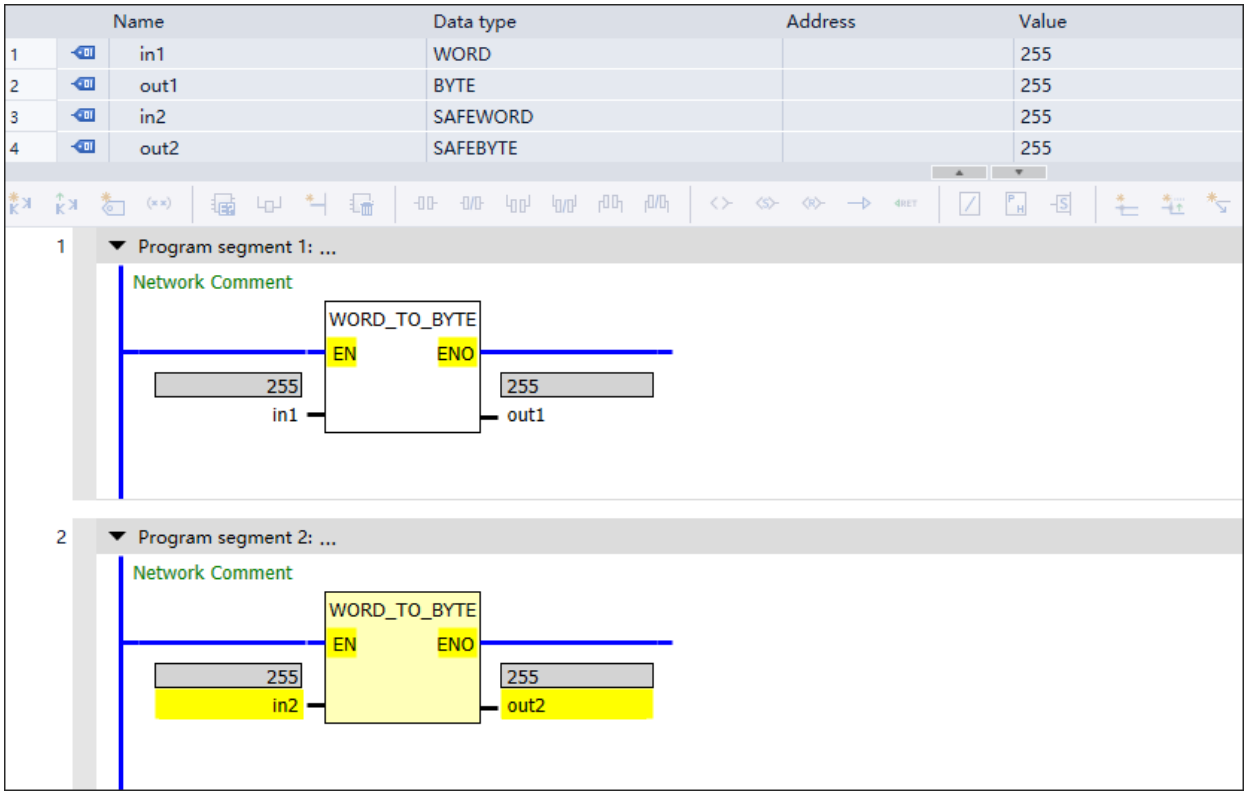
  Convert the bit string "In" of WORD type to the bit string "Out" of BYTE type.

- **Programming example**

  FBD:



  LD:

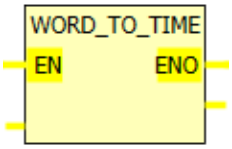
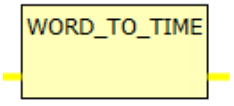


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - If a value outside the range of WORD#16#0000 to 00FF is input, a "conversion exception" will occur and the program will stop.

11.10.1.6.27 WORD_TO_TIME

Convert data of WORD type to TIME type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
WORD_TO_TIME	Convert WORD type data to TIME type	FC		

- **Related variables**

Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	WORD#16#0000 to FFFF	WORD#16#0000	Conversion object
Output	Out	Conversion result	T#0ms to T#1m5s535ms	T#0ms	Conversion result

Data types

	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	√	√	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	√	√

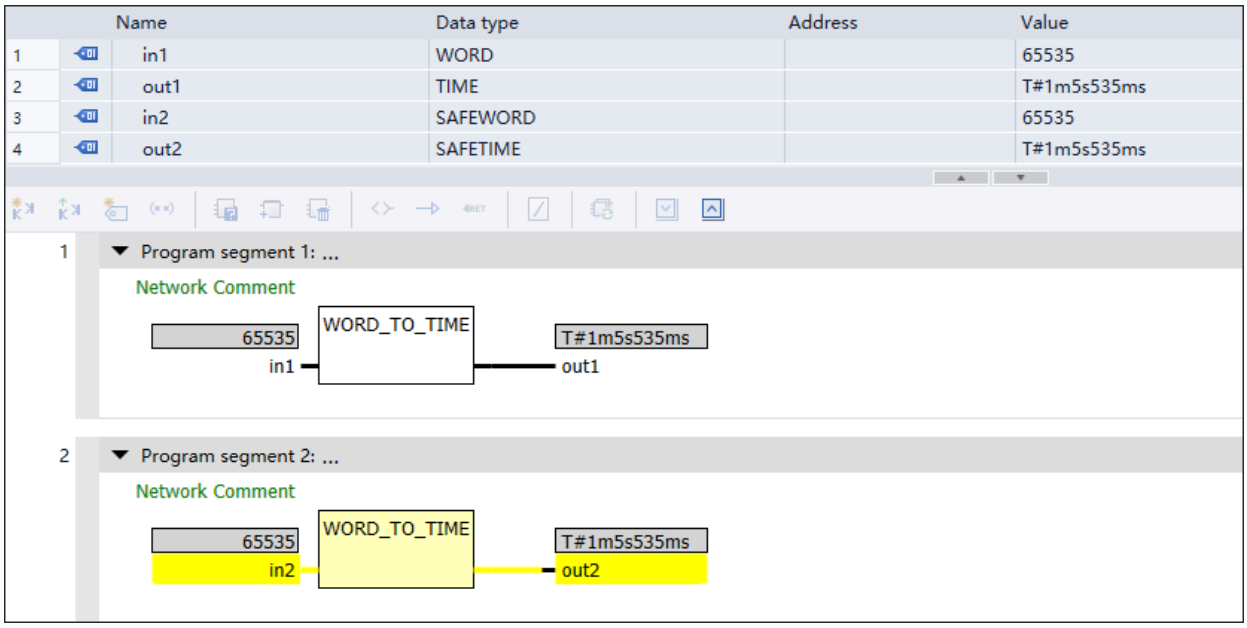
- **Function description**

  Convert the bit string "In" of WORD type to "Out" of TIME type.

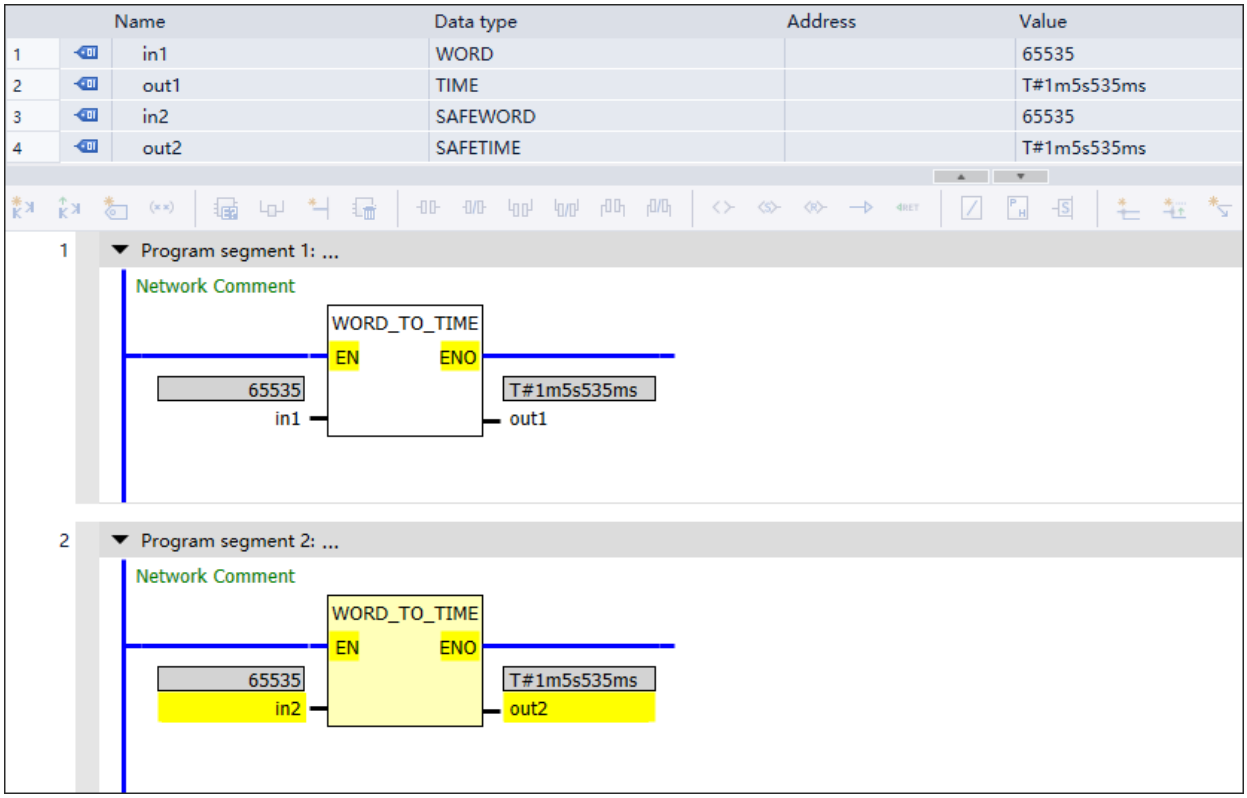
  Example: When the value of "In" is WORD#16#C549, the value of "Out" is T#50s505ms.

- **Programming example**

  FBD:



  LD:

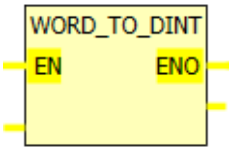
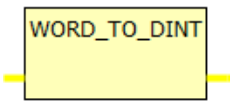


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.6.28 WORD_TO_DINT

Convert data of WORD type to DINT type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
WORD_TO_DINT	Convert WORD type data to DINT type	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	WORD#16#0000 to FFFF	WORD#16#0000	Conversion object
Output	Out	Conversion result	DINT#0 to 65535	DINT#0	Conversion result

  Data types

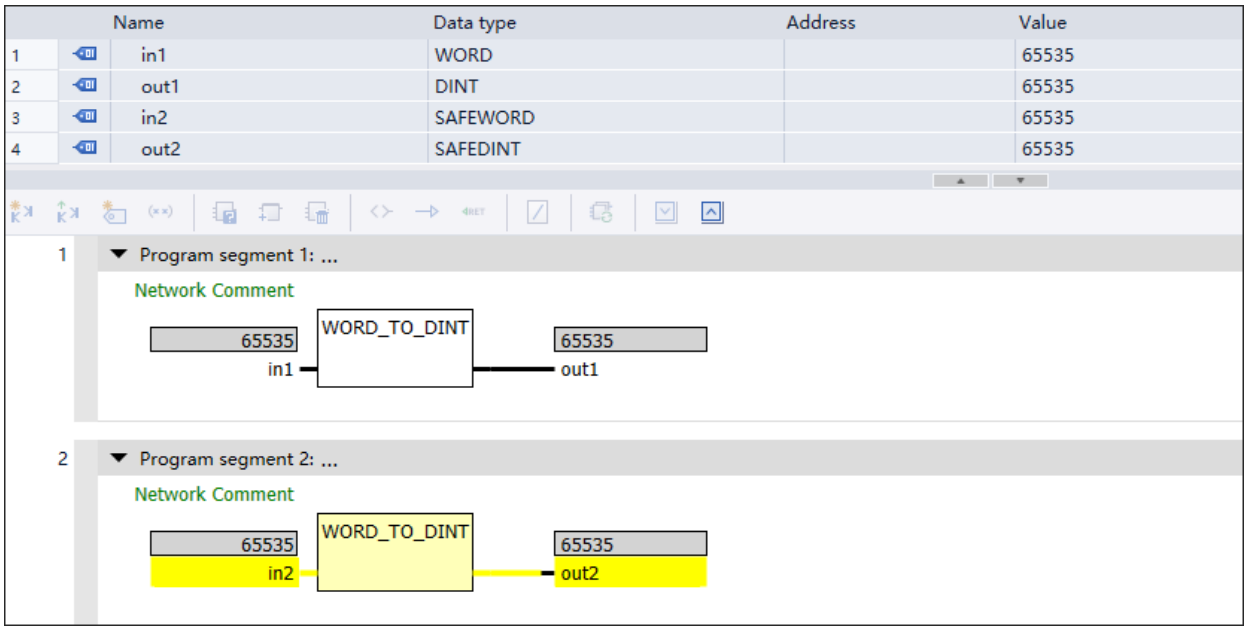
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	√	√	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	√	√	-	-

- **Function description**

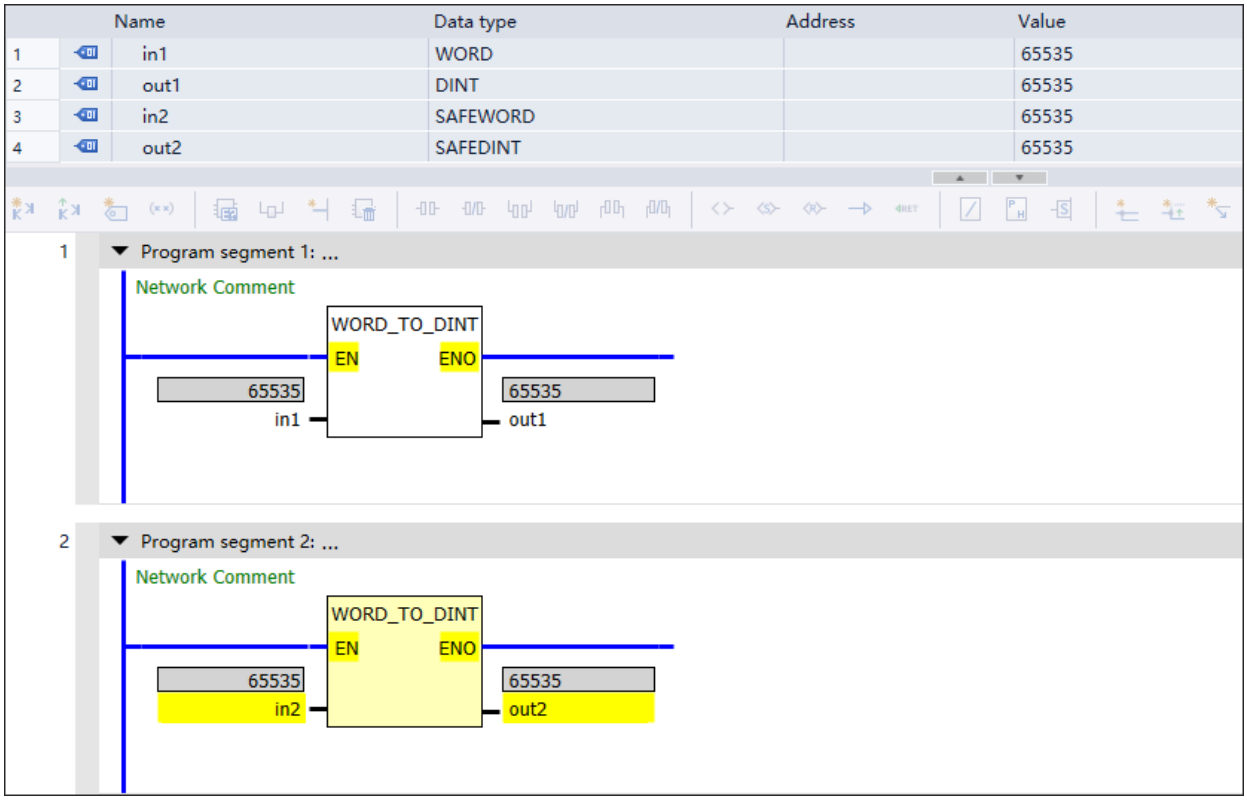
  Convert "In" of WORD type to "Out" of DINT type.

- **Programming example**

  FBD:



  LD:

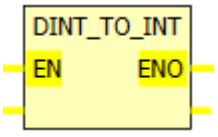
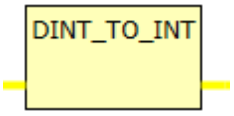


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.6.29 DINT_TO_INT

Convert data of DINT type to INT type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
DINT_TO_INT	Convert DINT type data to INT type	FC		

- **Related variables**

Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	DINT#-32768 to 32767	DINT#0	Conversion object
Output	Out	Conversion result	INT#-32768 to 32767	INT#0	Conversion result

Data types

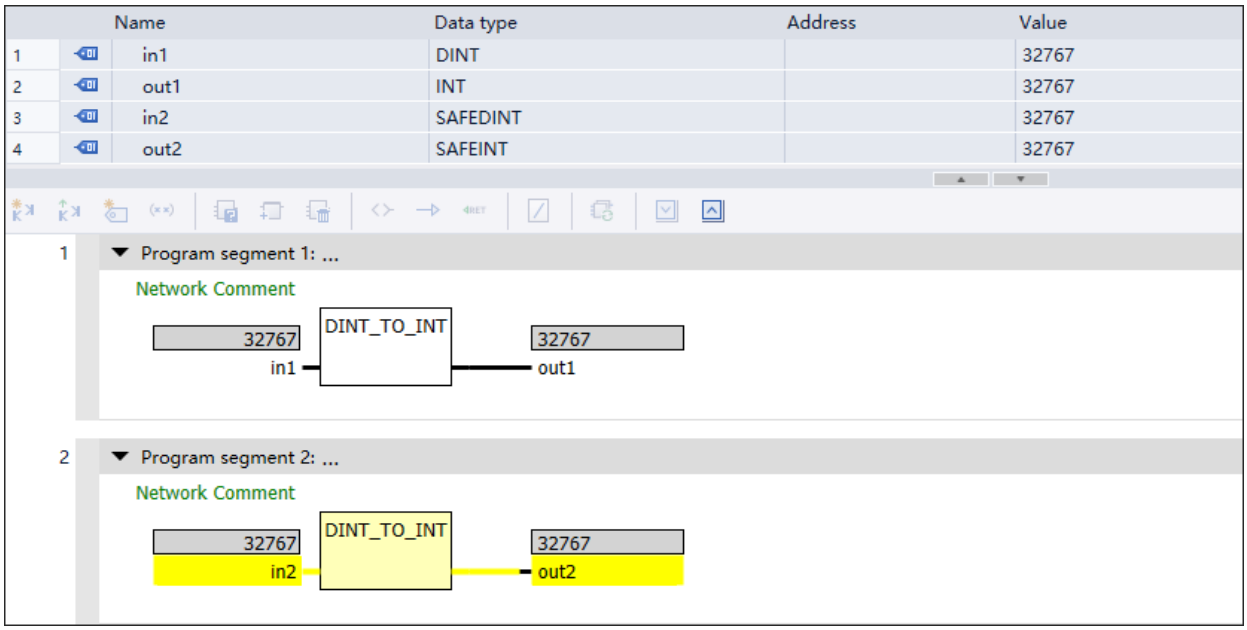
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	-	-	-	-	√	√	-	-
Out	-	-	-	-	-	-	-	-	√	√	-	-	-	-

- **Function description**

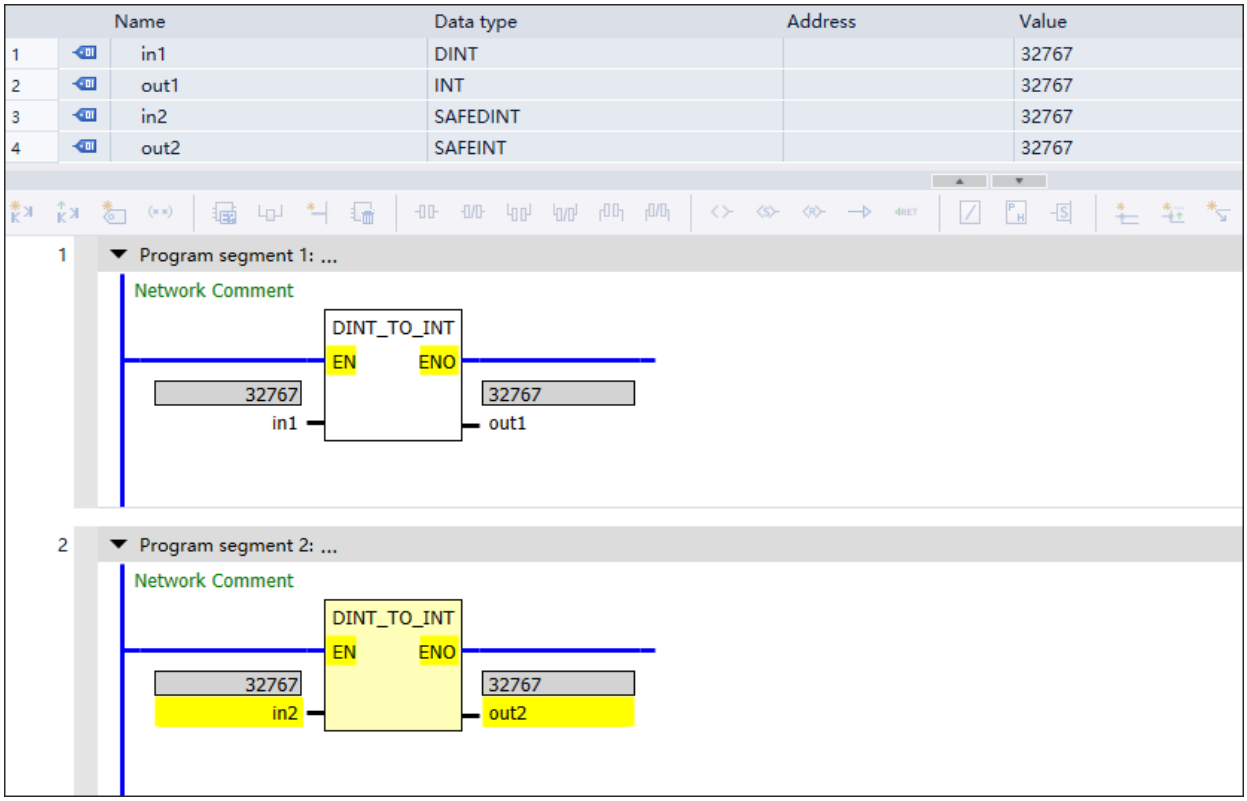
  Convert "In" of DINT type to "Out" of INT type.

- **Programming example**

  FBD:



  LD:

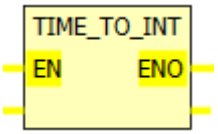
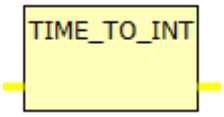


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - If a value outside the range of DINT#-32768 to 32767 is input, a "conversion exception" will occur and the program will stop.

11.10.1.6.30 TIME_TO_INT

Convert data of TIME type to INT type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
TIME_TO_INT	Convert TIME type data to INT type	FC		

- **Related variables**

  Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	T#0ms to T#32s767ms	T#0ms	Conversion object
Output	Out	Conversion result	INT#0 to 32767	INT#0	Conversion result

  Data types

	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	-	-	-	-	-	-	√	√
Out	-	-	-	-	-	-	-	-	√	√	-	-	-	-

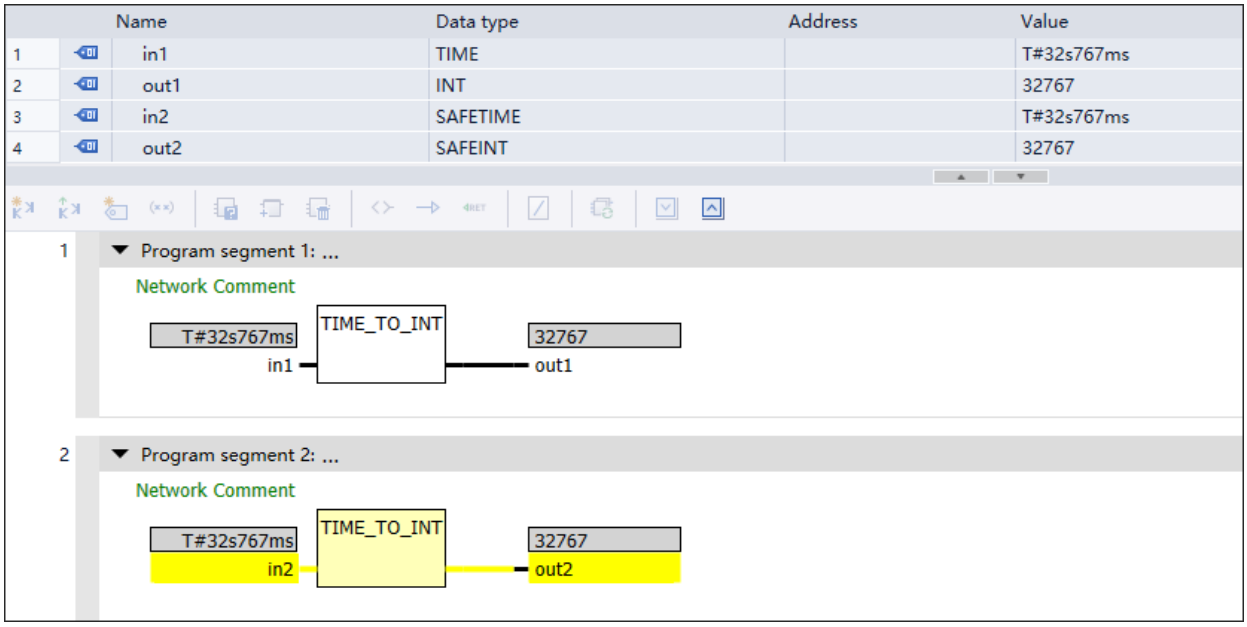
- **Function description**

  Convert "In" of TIME type to "Out" of INT type.

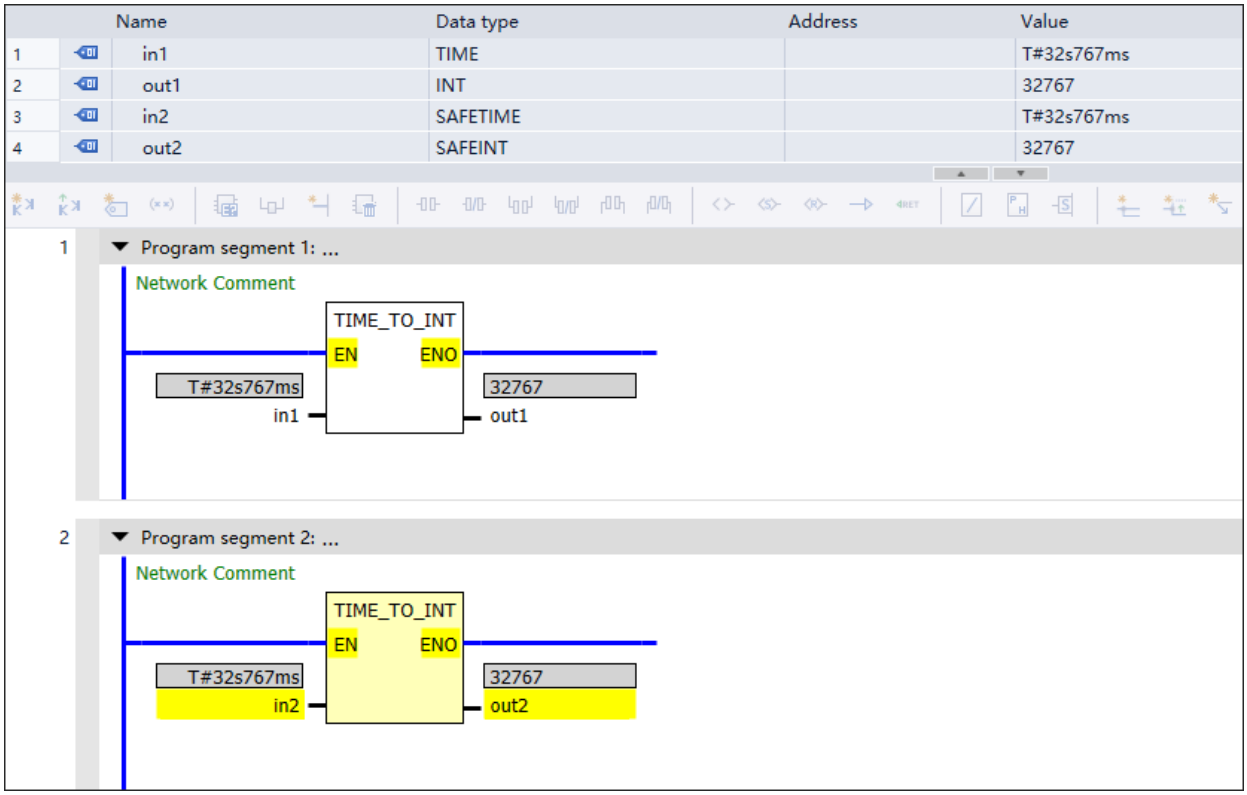
  Example: When the value of "In" is T#10s500ms, the value of "Out" is INT#10500.

- **Programming example**

  FBD:



  LD:

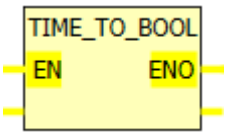
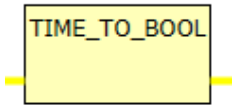


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - If a value outside the range of TIME#0ms to 32s767ms is input, a "conversion exception" will occur and the program will stop.

11.10.1.6.31 TIME_TO_BOOL

Convert TIME type data to BOOL type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
TIME_TO_BOOL	Convert TIME type data to BOOL type	FC		

- **Related variables**

Variables

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	T#0ms to T#49d17h2m47s295ms	T#0ms	Conversion object
Output	Out	Conversion result	FALSE, TRUE	FALSE	Conversion result

Data types

	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	-	-	-	-	-	-	√	√
Out	√	√	-	-	-	-	-	-	-	-	-	-	-	-

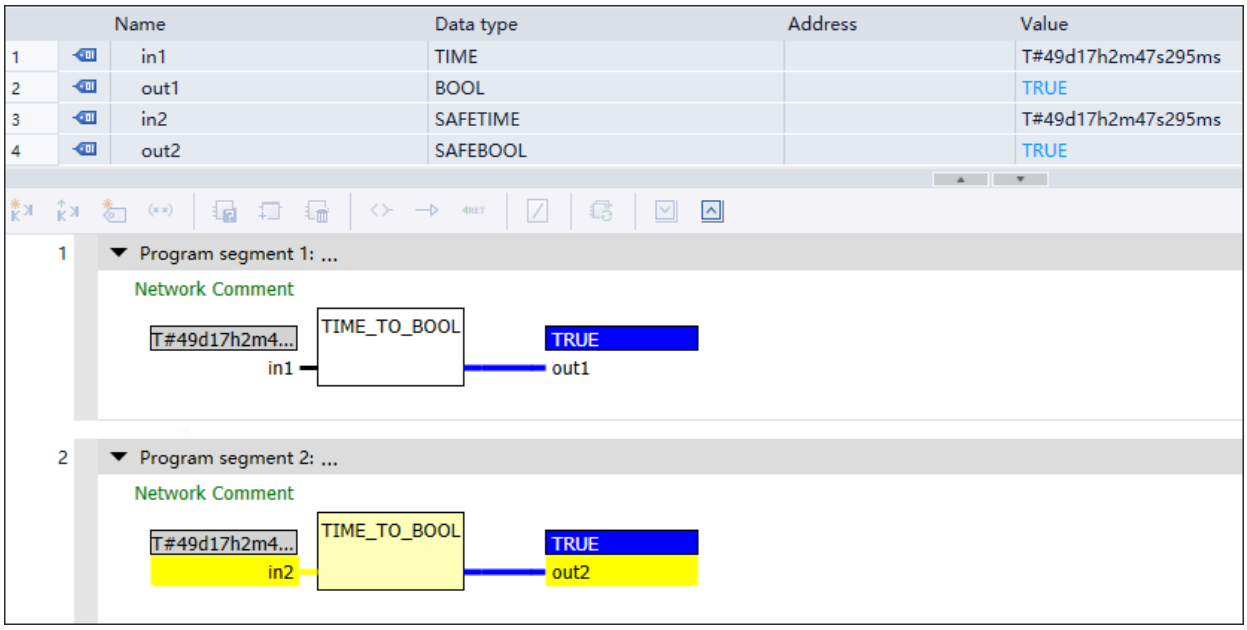
- **Function description**

  Convert "In" of TIME type to "Out" of BOOL type.

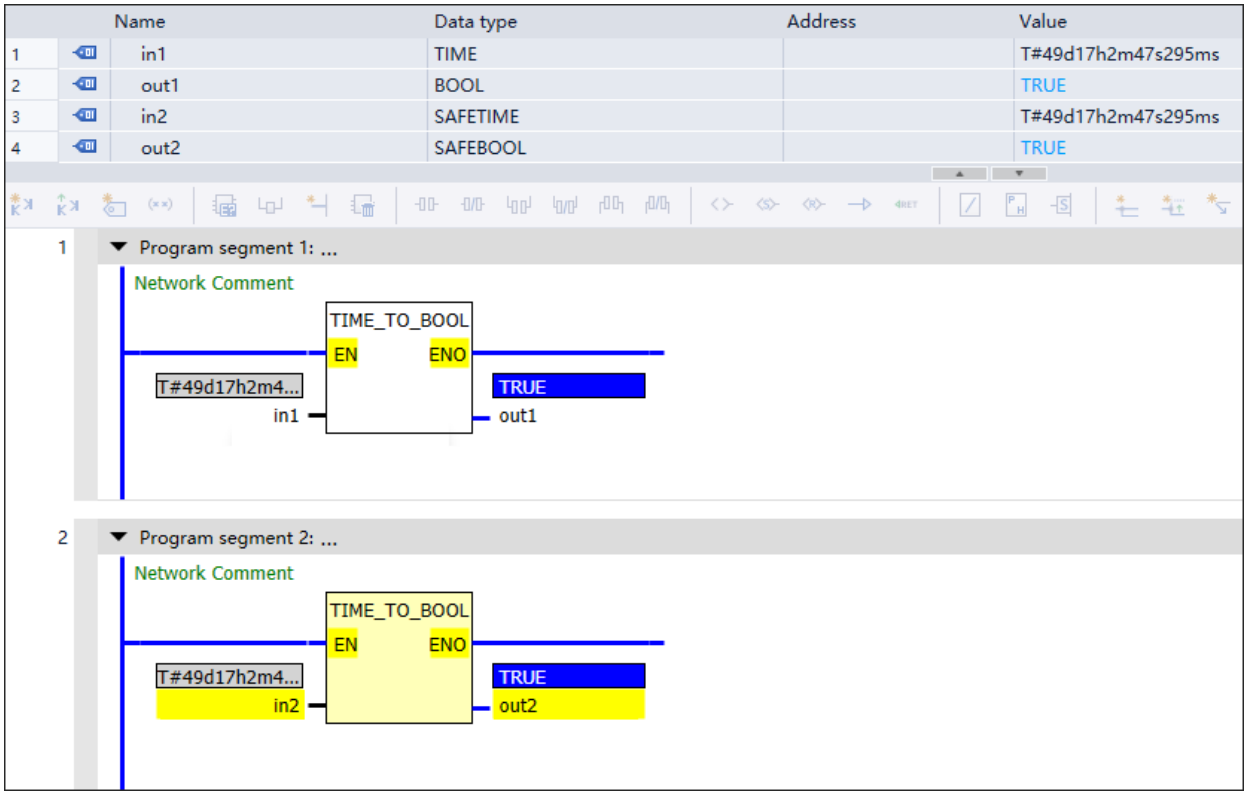
  - When the value of "In" is T#0ms, the value of "Out" is FALSE.
  - When the value of "In" is T#1ms to T#49d17h2m47s295ms (other than T#0ms), the value of "Out" is TRUE.

- **Programming example**

  FBD:



  LD:

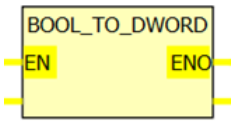
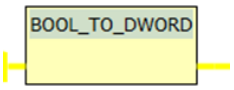


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.6.32 BOOL_TO_DWORD

Convert BOOL type data to DWORD type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
BOOL_TO_DWORD	Convert BOOL type data to DWORD type	FC		

- **Related variables**

  Variables

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	FALSE, TRUE	FALSE	Conversion object
Output	Out	Conversion result	DWORD#0, DWORD #1	DWORD #0	Conversion result

  Data types

	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	√	√	-	-	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	√	√	-	-	-	-	-	-

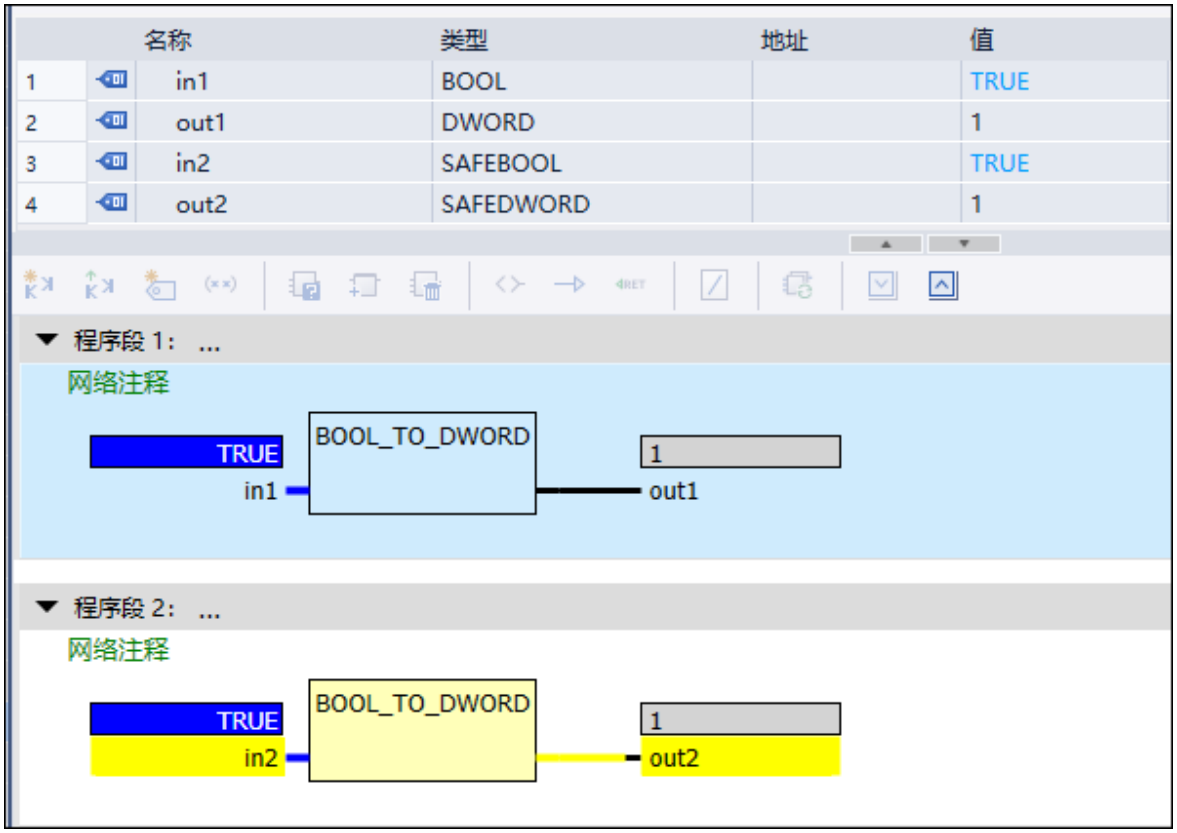
- **Function description**

  Convert "In" of BOOL type to the bit string "Out" of WORD type.

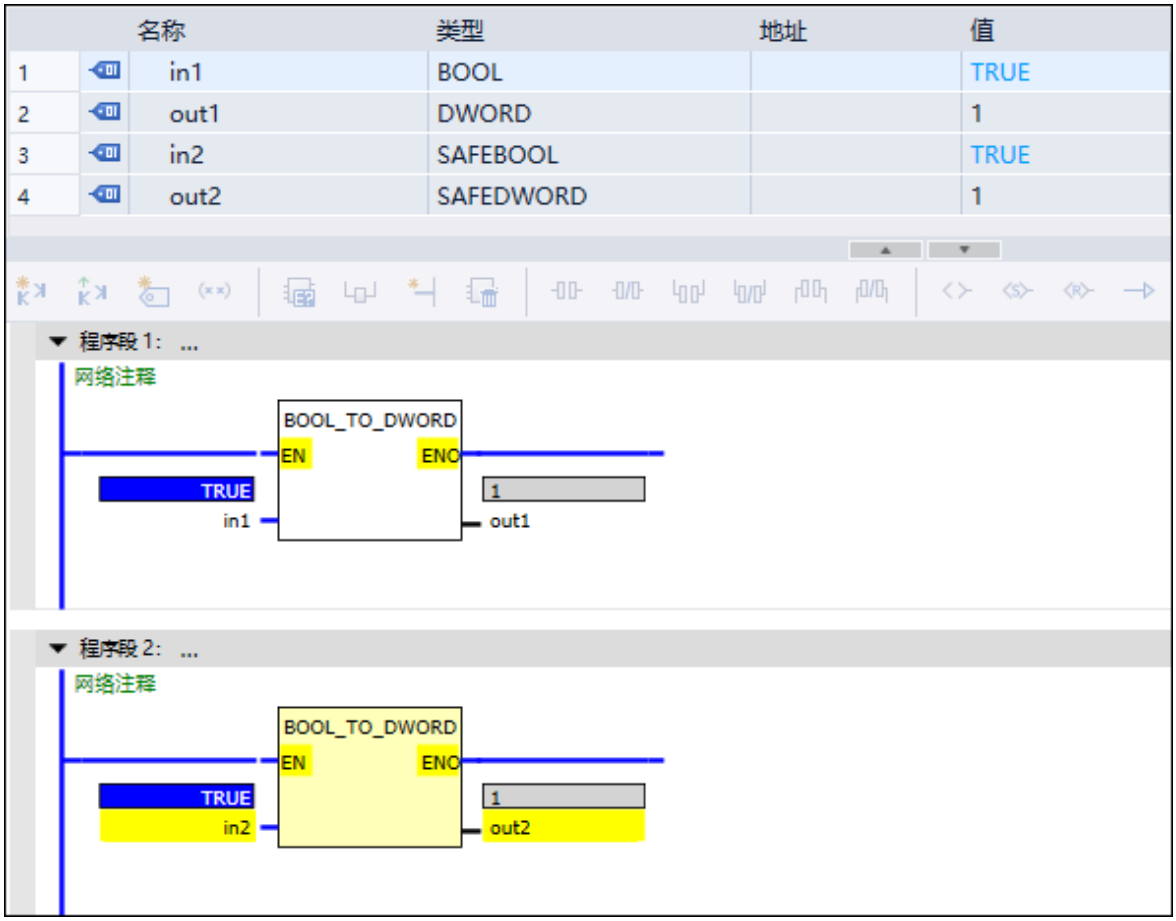
  - When the value of "In" is FALSE, the value of "Out" is DWORD#0.
  - When the value of "In" is TRUE, the value of "Out" is DWORD #1.

- **Programming example**

  FBD:



  LD:

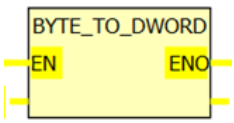
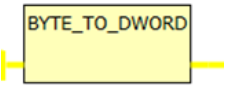


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.6.33 BYTE_TO_DWORD

Convert BYTE type data to DWORD type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
BYTE_TO_DWORD	Convert BYTE type data to DWORD type	FC		

- **Related variables**

Variables

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	BYTE#16#00 to FF	BYTE#16#00	Conversion object

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Conversion result	DWORD#0 to DWORD#255	DWORD#0	Conversion result

Data types

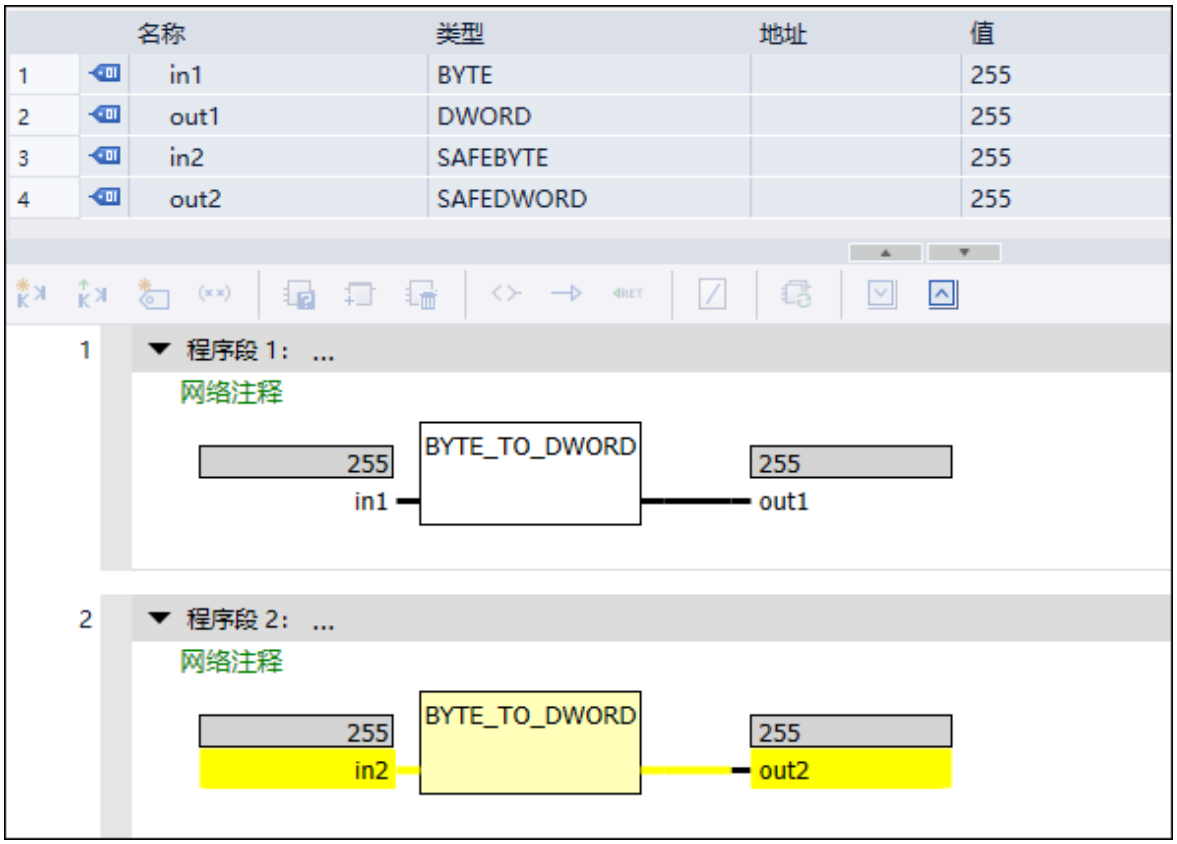
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	√	√	-	-	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	√	√	-	-	-	-	-	-

- **Function description**

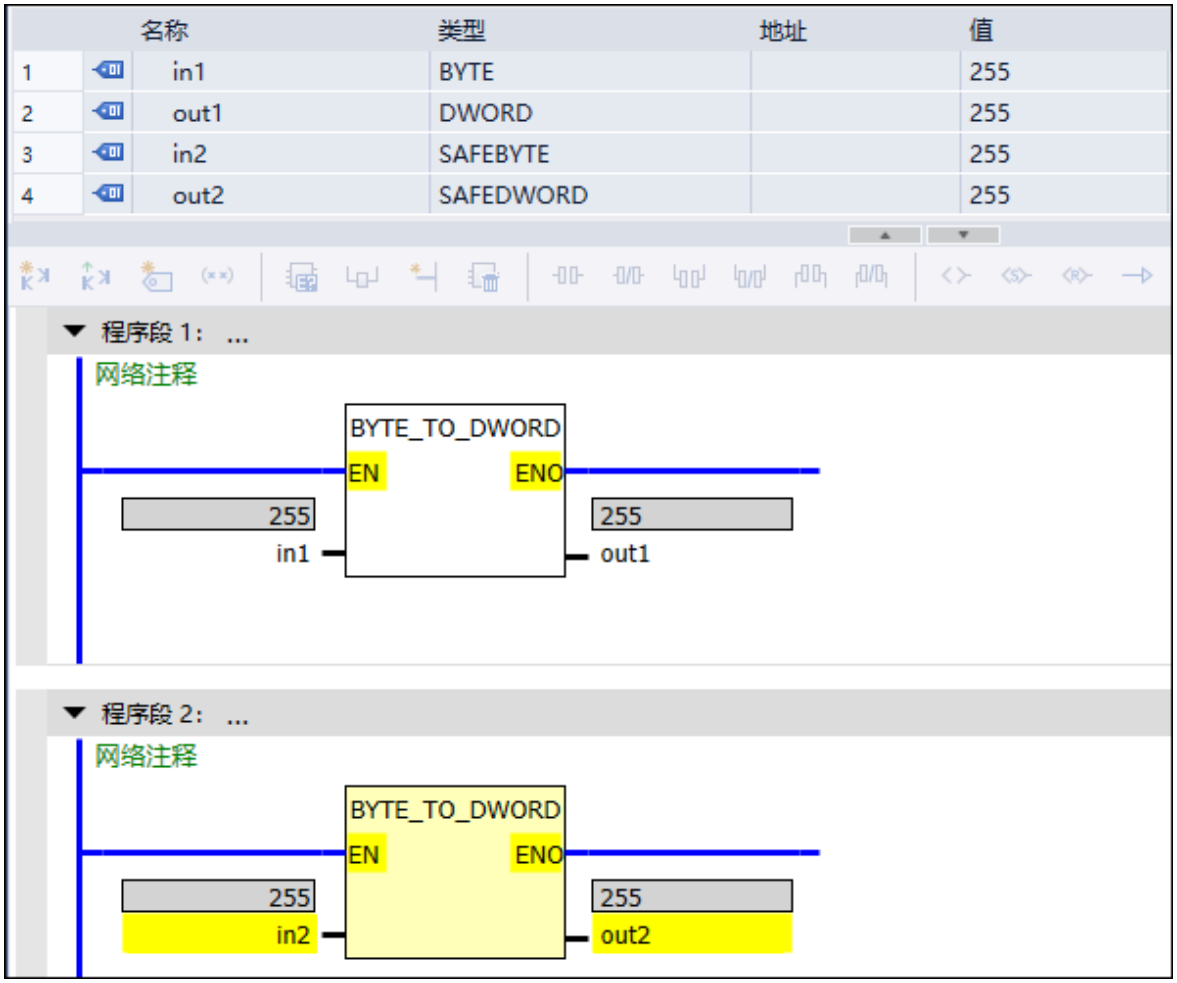
  Convert the bit string "In" of BYTE type to the bit string "Out" of DWORD type.

- **Programming example**

  FBD:



  LD:

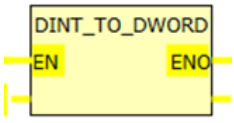
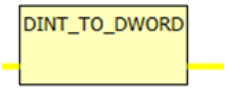


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.6.34 DINT_TO_DWORD

Convert DINT type data to DWORD type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
DINT_TO_ DWORD	Convert DINT type data to DWORD type	FC		

- **Related variables**

Variables

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	DINT#0 to 2147483647	DINT#0	Conversion object

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Conversion result	DWORD#0 to 2147483647	DWORD#0	Conversion result

Data types

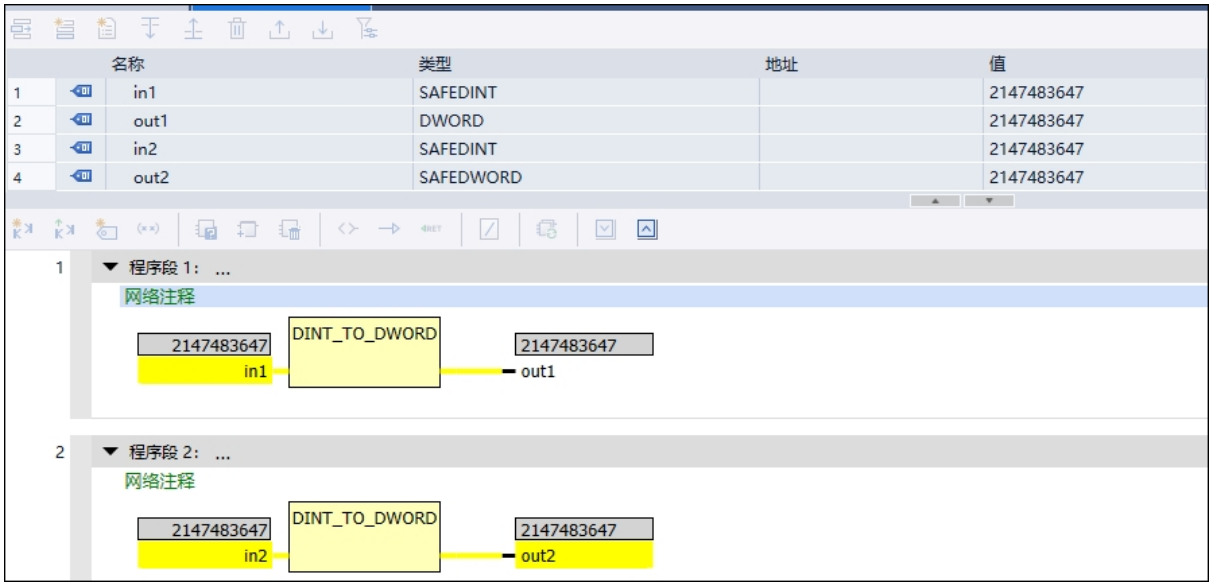
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	-	-	-	-	√	√	-	-
Out	-	-	-	-	-	-	√	√	-	-	-	-	-	-

- **Function description**

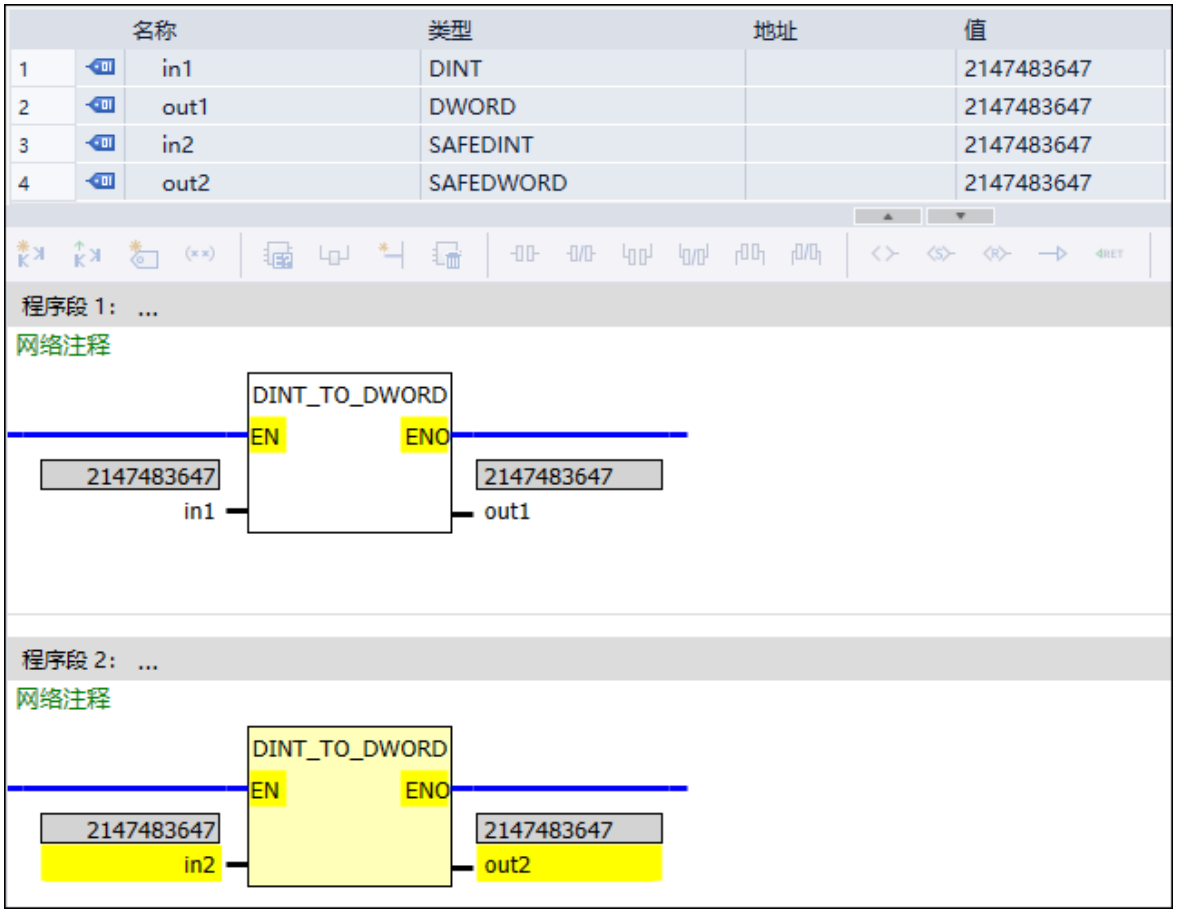
  Convert "In" of DINT type to the bit string "Out" of DWORD type.

- **Programming example**

  FBD:



  LD:

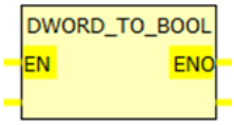
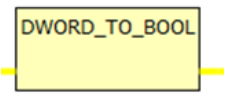


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - If a value outside the range of DINT#0 to 2147483647 is input, a "conversion exception" will occur and the program will stop.

11.10.1.6.35 DWORD_TO_BOOL

Convert DWORD type data to BOOL type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
DWORD_TO_BOOL	Convert DWORD type data to BOOL type.	FC		

- **Related variables**

Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	DWORD#0 to 1	DWORD#0	Conversion object

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Conversion result	FALSE, TRUE	FALSE	Conversion result

Data types

	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	√	√	-	-	-	-	-	-
Out	√	√	-	-	-	-	-	-	-	-	-	-	-	-

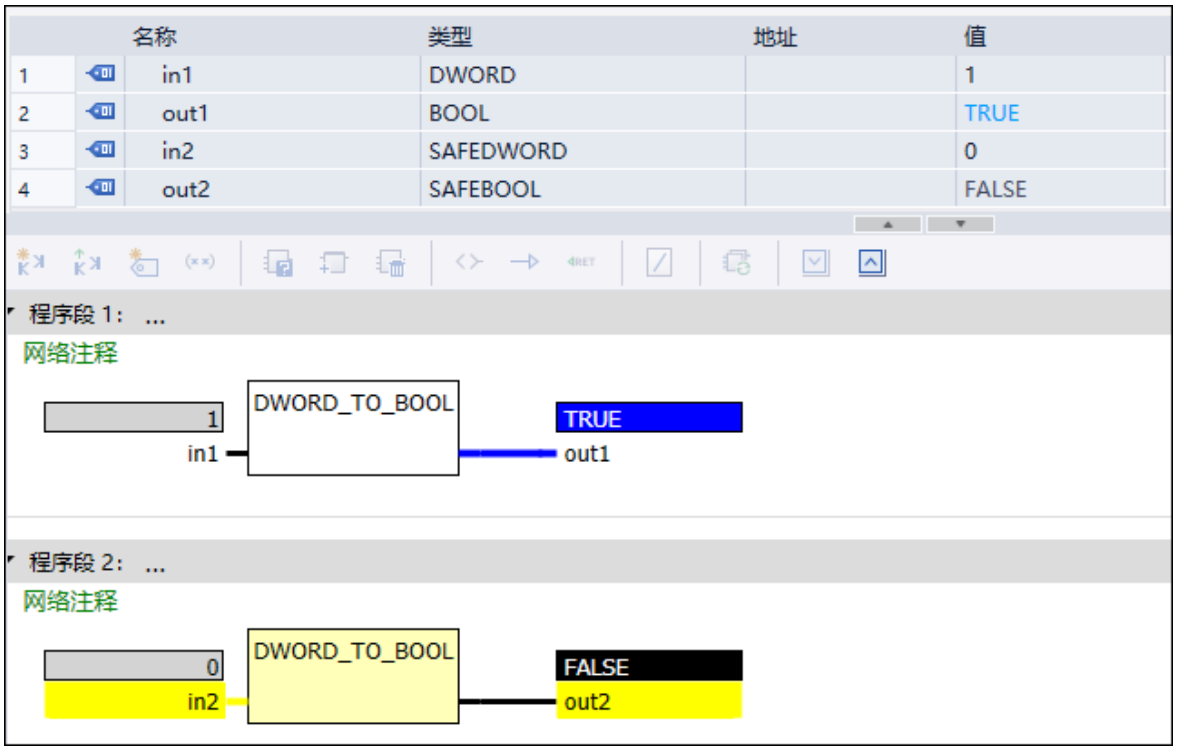
- **Function description**

  Convert the bit string "In" of WORD type to "Out" of BOOL type.

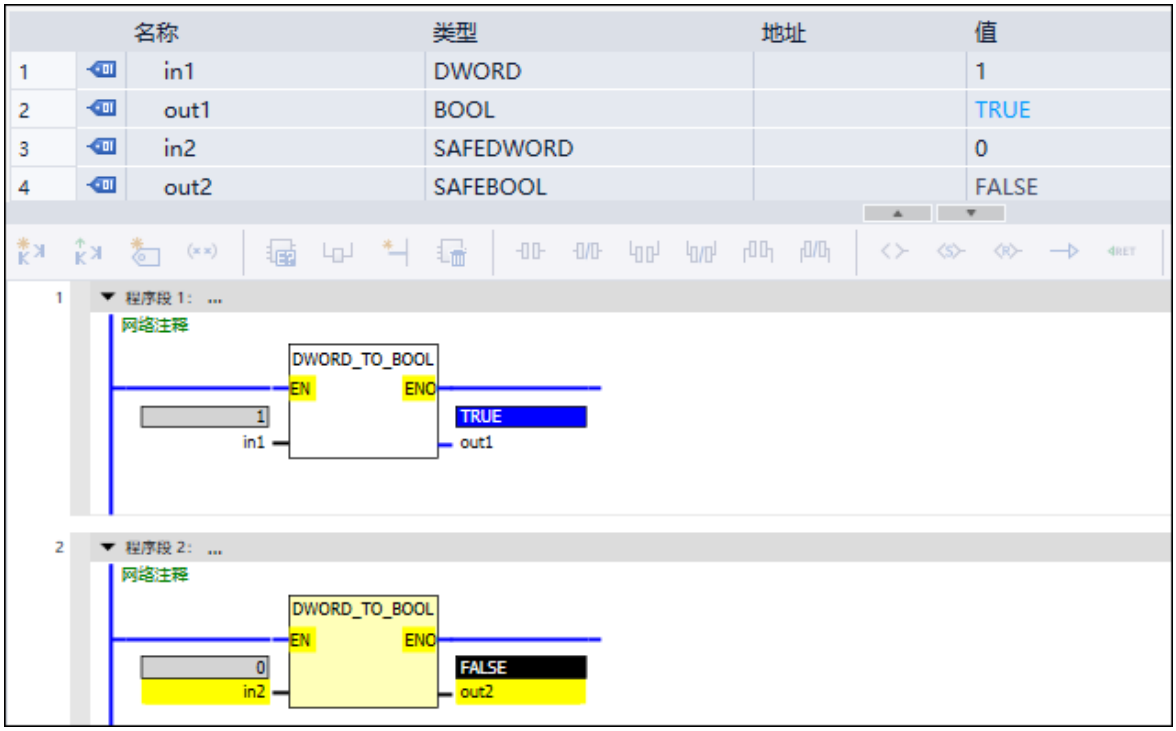
  - When the value of "In" is 0, the value of "Out" is FALSE.
  - When the value of "In" is 1, the value of "Out" is TRUE.

- **Programming example**

  FBD:



  LD:

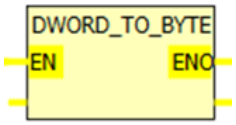
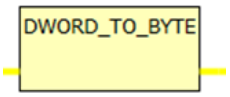


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.6.36 DWORD_TO_BYTE

Convert DWORD type data to BYTE type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
DWORD_TO_BYTE	Convert DWORD type data to BYTE type	FC		

- **Related variables**

Variables

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	DWORD#0 to 255	DWORD#0	Conversion object
Output	Out	Conversion result	BYTE16#00 to FF	BYTE16#00	Conversion result

Data types

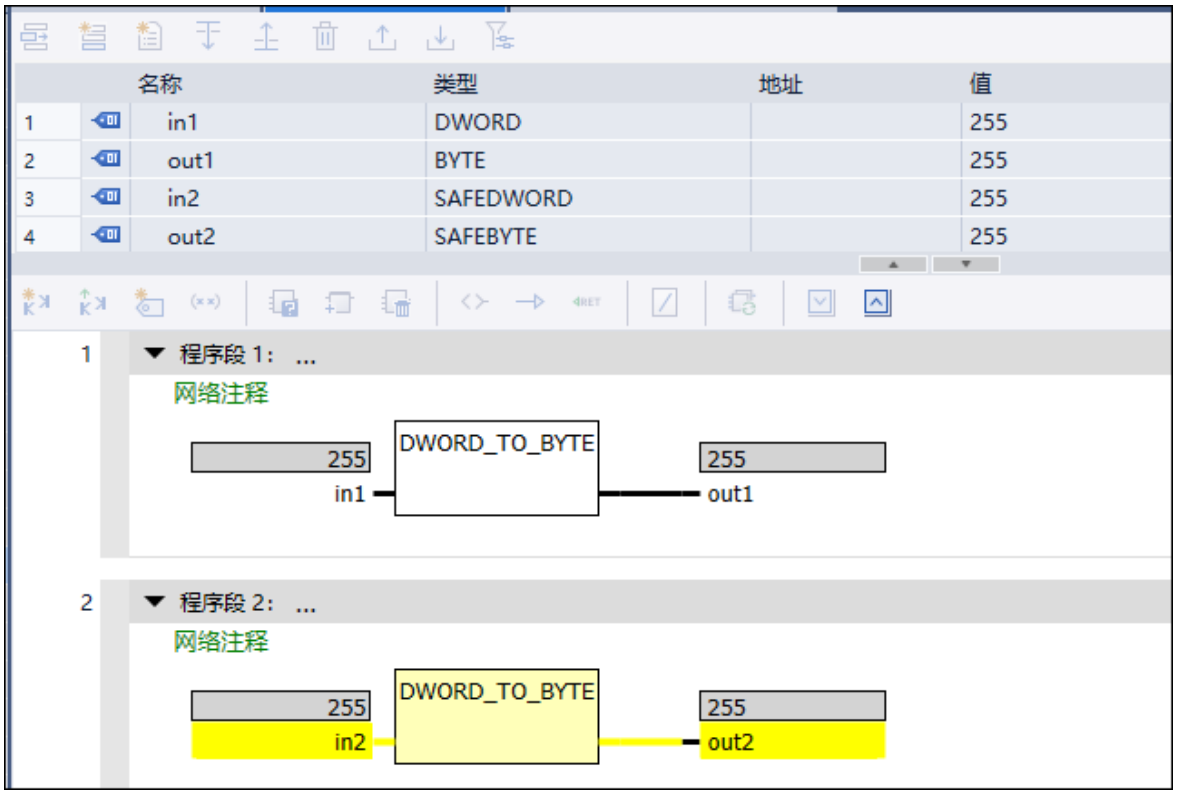
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	√	√	-	-	-	-	-	-
Out	-	-	√	√	-	-	-	-	-	-	-	-	-	-

- **Function description**

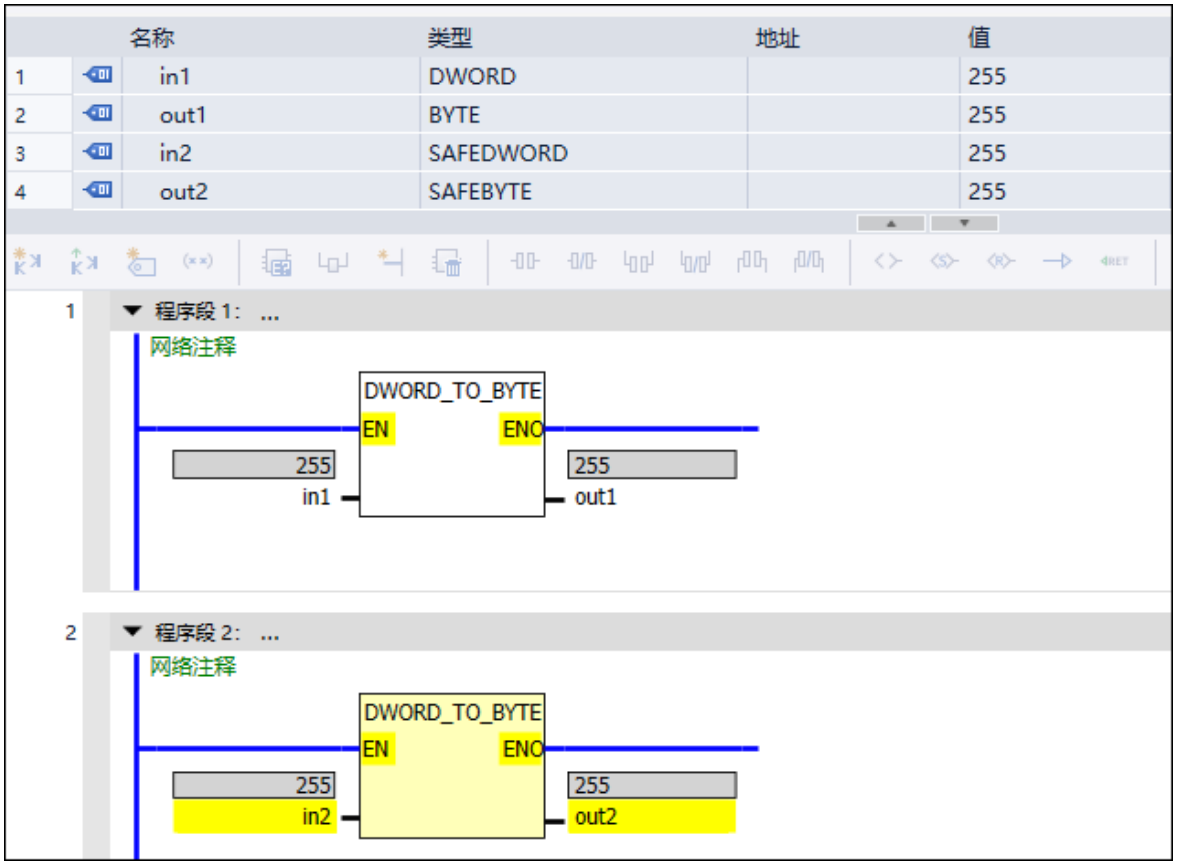
  Convert the bit string "In" of DWORD type to the bit string "Out" of BYTE type.

- **Programming example**

  FBD:



  LD:

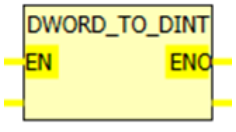
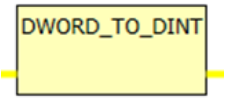


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - If a value outside the range of DWORD#0 to 255 is input, a "conversion exception" will occur and the program will stop.

11.10.1.6.37 DWORD_TO_DINT

Convert DWORD type data to DINT type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
DWORD_TO_DINT	Convert DWORD type data to DINT type	FC		

- **Related variables**

Variables

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	DWORD#0 to 2147483647	DWORD#0	Conversion object

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Conversion result	DINT#0 to 2147483647	DINT#0	Conversion result

Data types

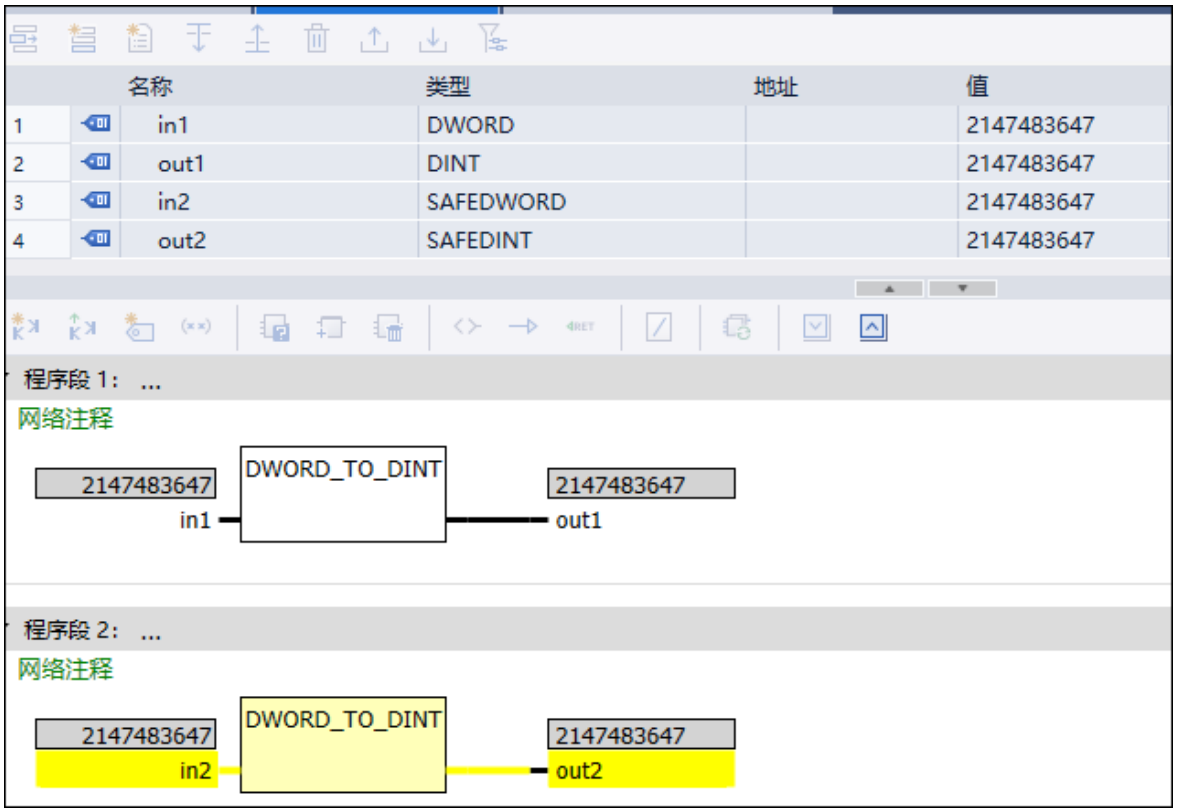
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	√	√	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	√	√	-	-

- **Function description**

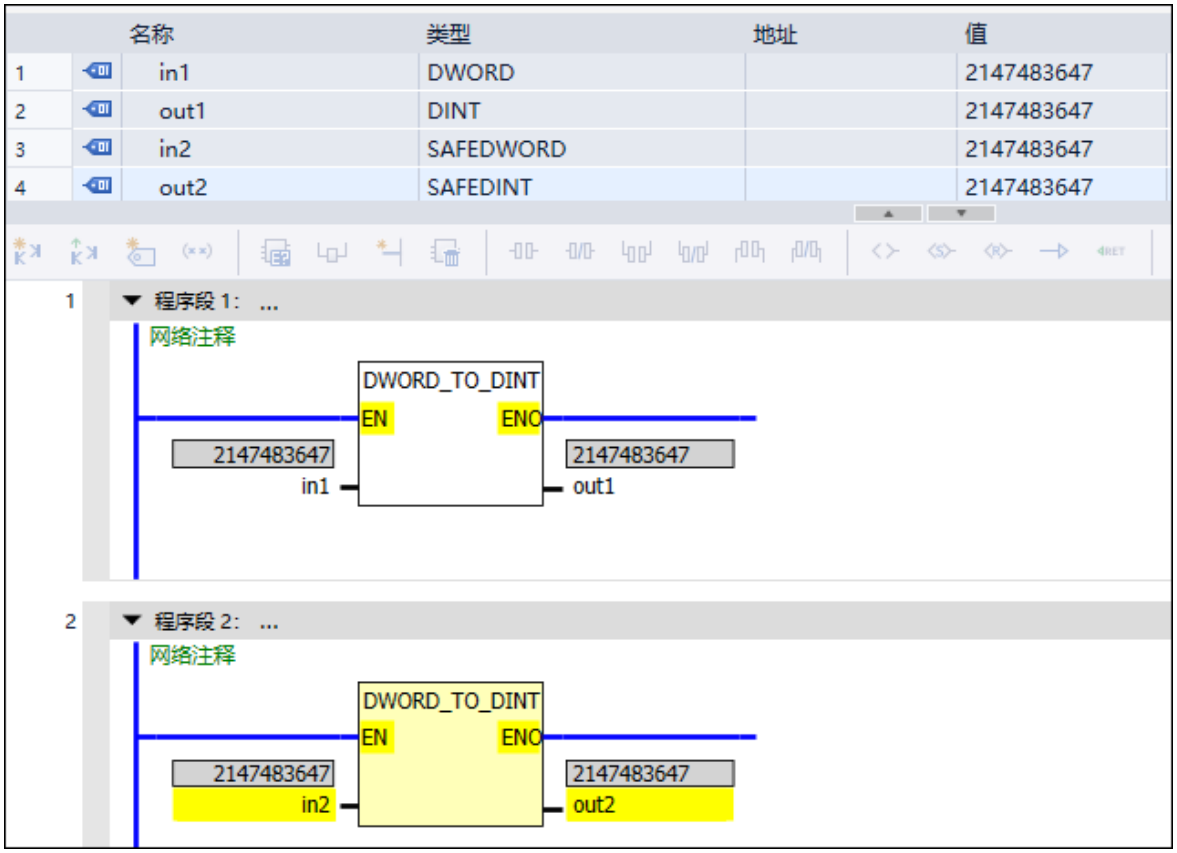
  Convert the bit string "In" of DWORD type to the integer "Out" of DINT type.

- **Programming example**

  FBD:



  LD:

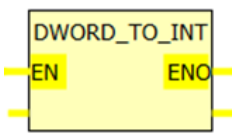
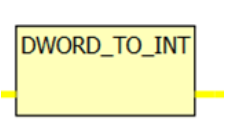


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - If a value outside the range of DWORD#0 to 2147483647 is input, a "conversion exception" will occur and the program will stop.

11.10.1.6.38 DWORD_TO_INT

Convert DWORD type data to INT type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
DWORD_TO_INT	Convert DWORD type data to INT type	FC		

- **Related variables**

Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	DWORD#0 to 32767	DWORD#0	Conversion object

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Conversion result	INT#0 to 32767	INT#0	Conversion result

Data types

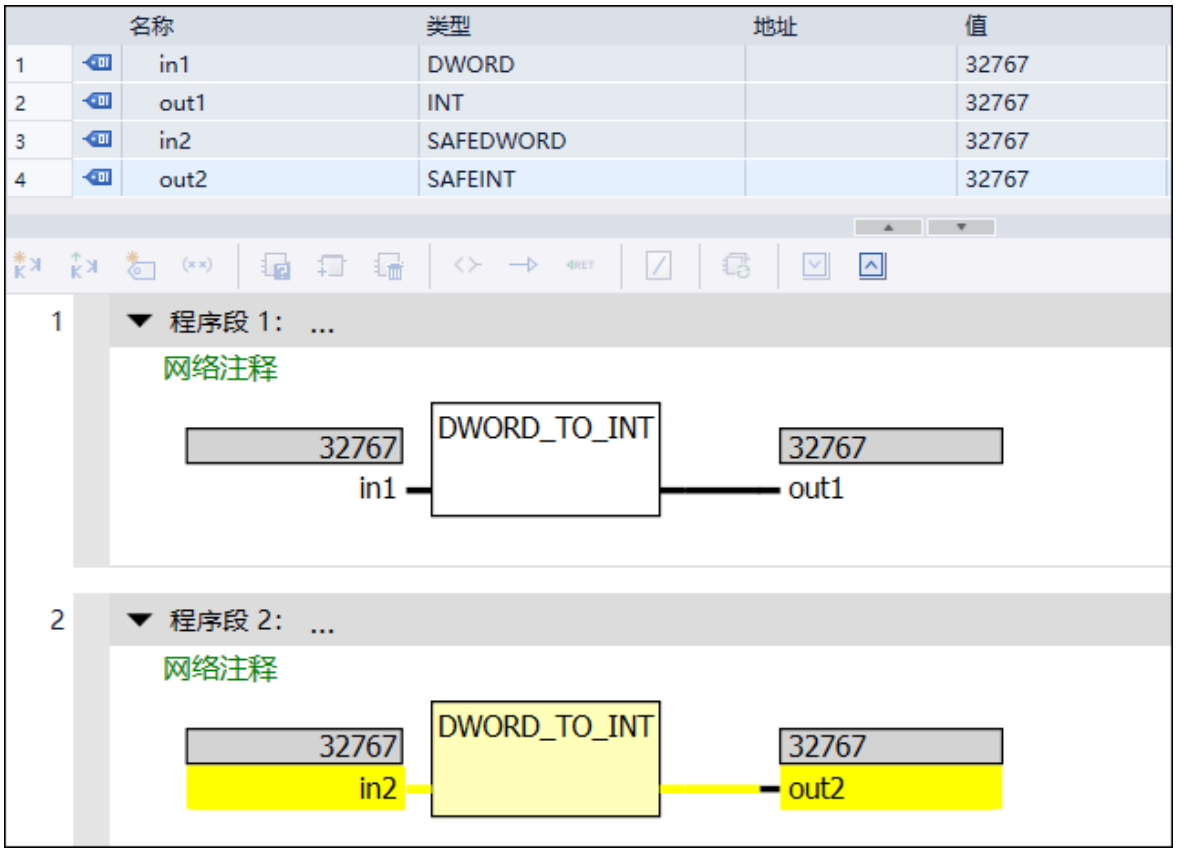
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	√	√	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	√	√	-	-	-	-

- **Function description**

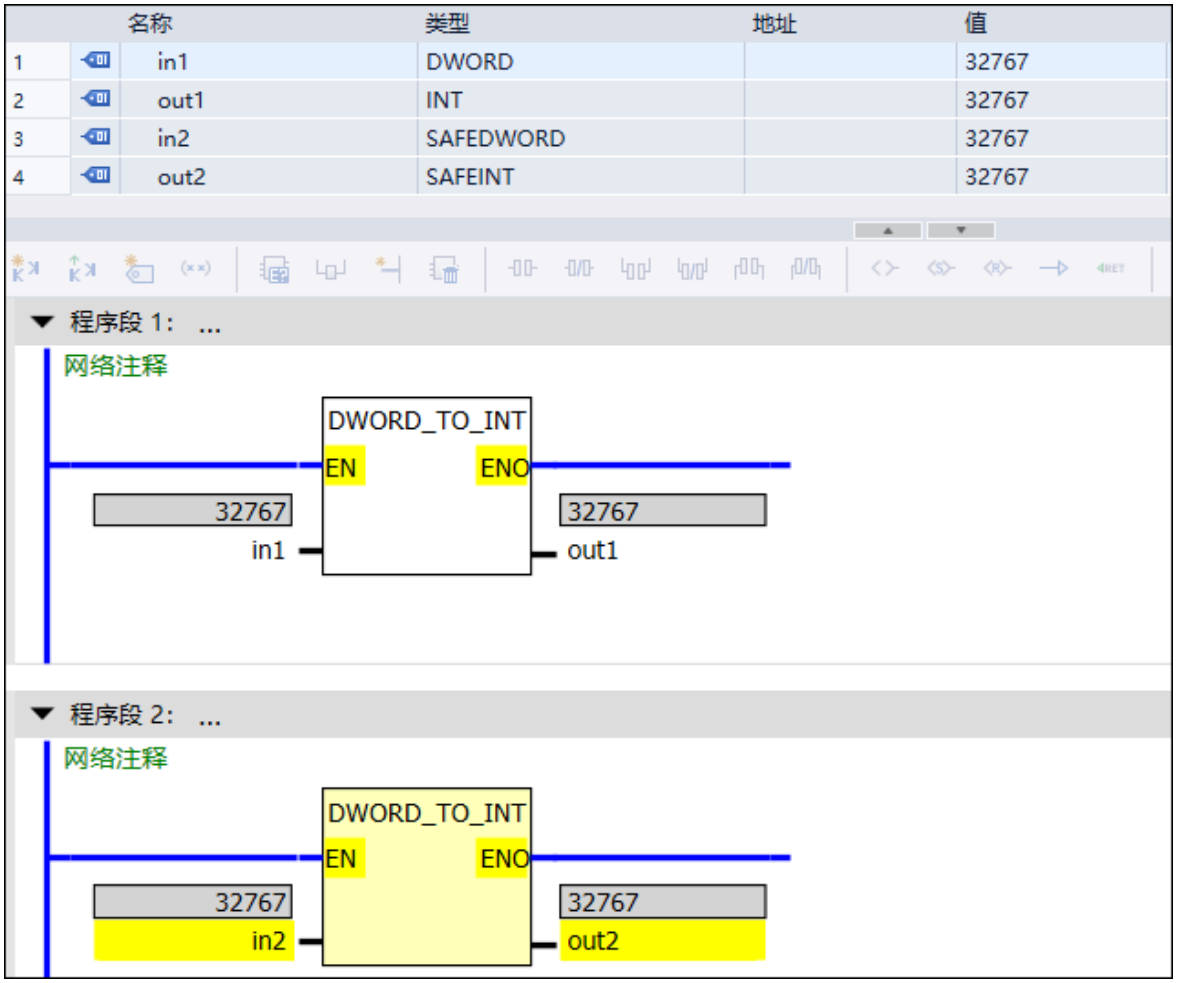
  Convert the bit string "In" of DWORD type to the integer "Out" of INT type.

- **Programming example**

  FBD:



  LD:

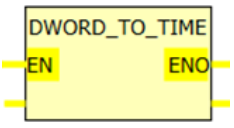
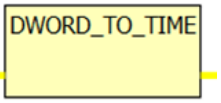


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - If a value outside the range of DWORD#0 to 32767 is input, a "conversion exception" will occur and the program will stop.

11.10.1.6.39 DWORD_TO_TIME

Convert DWORD type data to TIME type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
DWORD_TO_TIME	Convert DWORD type data to TIME type	FC		

- **Related variables**

  Variables

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	DWORD#0 to 4294967295	DWORD#0	Conversion object
Output	Out	Conversion result	T#0ms to T#49d17h2m47s295ms	T#0ms	Conversion result

Data types

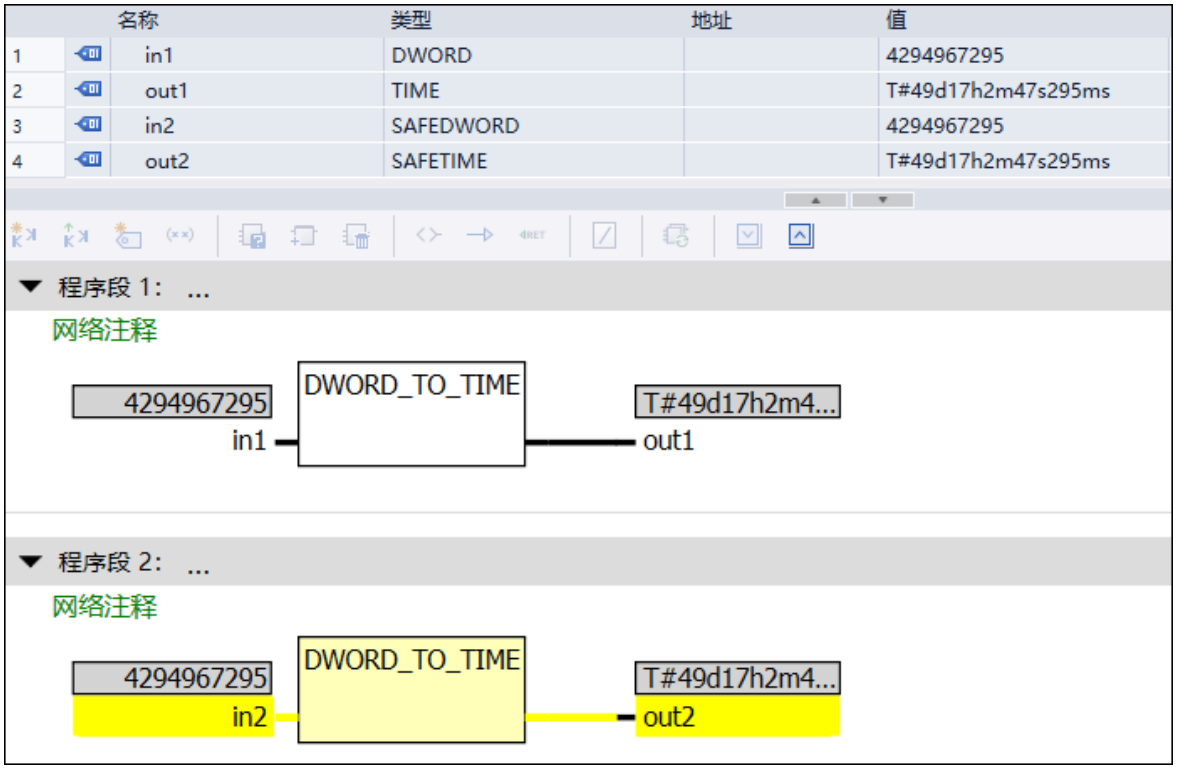
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	√	√	-	-	-	-	-	-
Out	-	-	-	-	-	-	-	-	-	-	-	-	√	√

- **Function description**

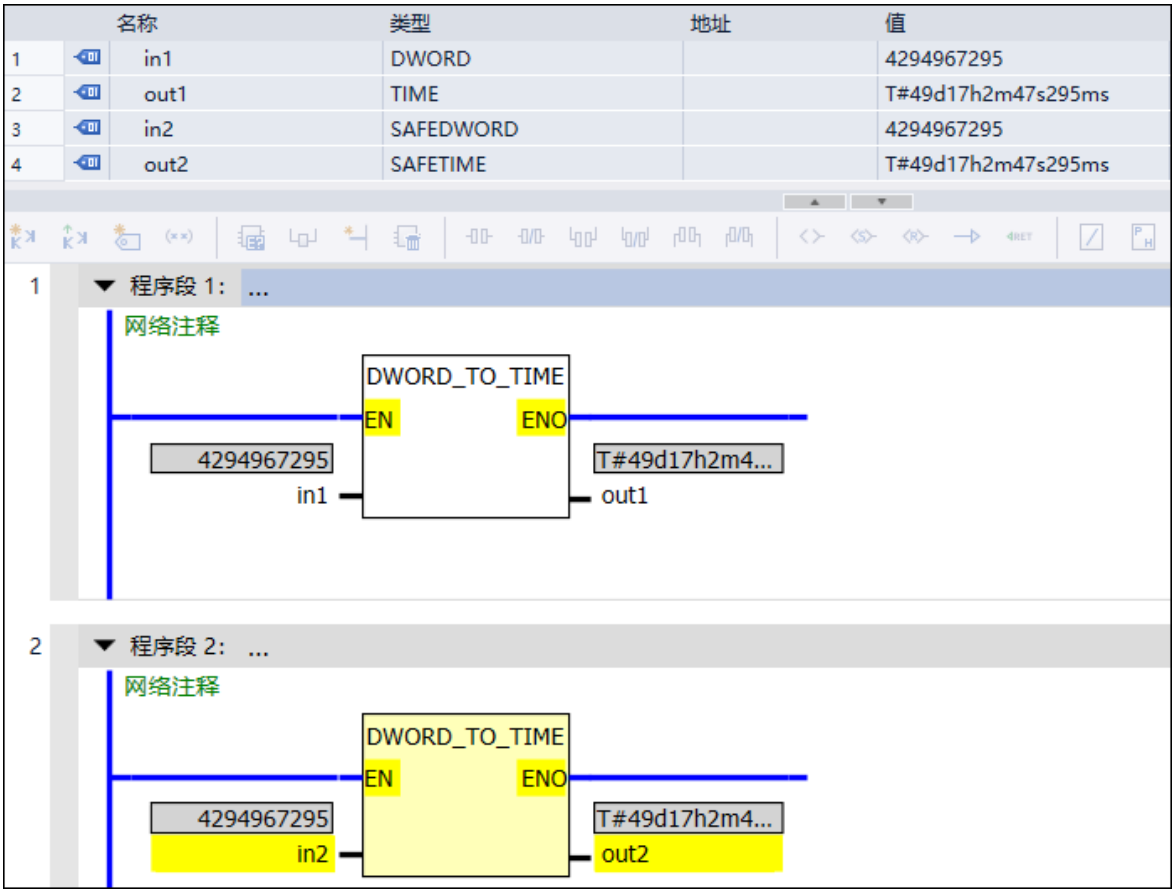
  Convert the bit string "In" of DWORD type to "Out" of TIME type.

- **Programming example**

  FBD:



  LD:

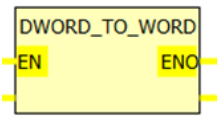
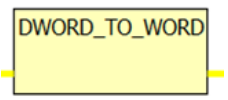


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.6.40 DWORD_TO_WORD

Convert DWORD type data to WORD type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
DWORD_TO_WORD	Convert DWORD type data to WORD type	FC		

- **Related variables**

Variables

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	DWORD#0 to T#65535	DWORD#0	Conversion object
Output	Out	Conversion result	WORD#16#0000 to FFFF	WORD#16#0000	Conversion result

Data types

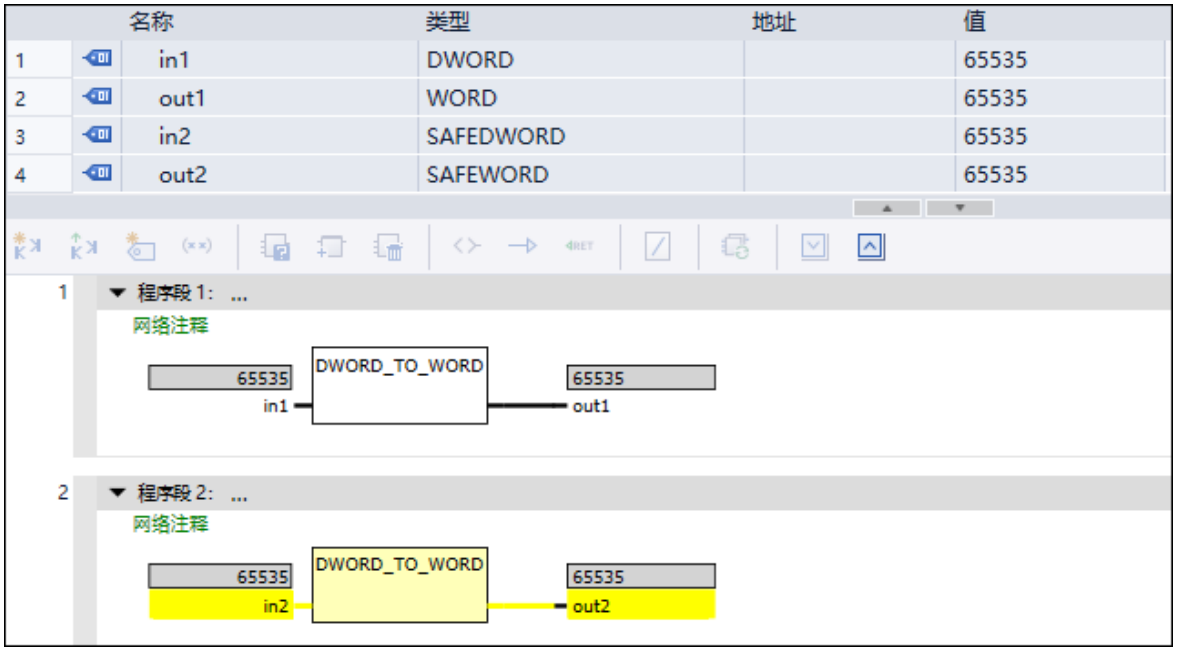
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	√	√	-	-	-	-	-	-
Out	-	-	-	-	√	√	-	-	-	-	-	-	-	-

- **Function description**

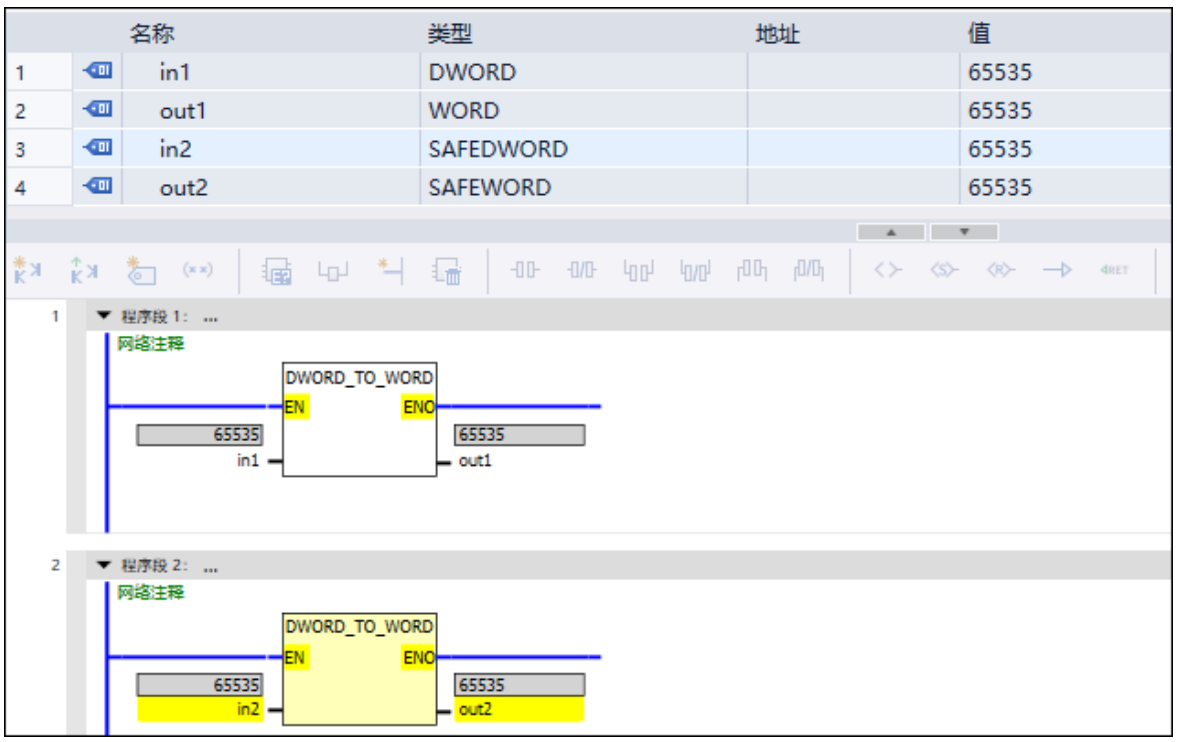
  Convert the bit string "In" of DWORD type to the bit string "Out" of WORD type.

- **Programming example**

  FBD:



  LD:

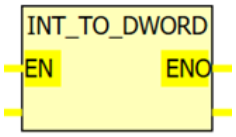
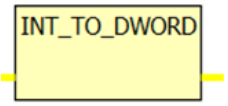


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - If a value outside the range of DWORD#0 to 66535 is input, a "conversion exception" will occur and the program will stop.

11.10.1.6.41 INT_TO_DWORD

Convert INT type data to DWORD type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
INT_TO_DWORD	Convert INT type data to DWORD type	FC		

- **Related variables**

Variables

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	INT#0 to 32767	INT#0	Conversion object
Output	Out	Conversion result	DWORD#0 to 32767	DWORD#0	Conversion result

Data types

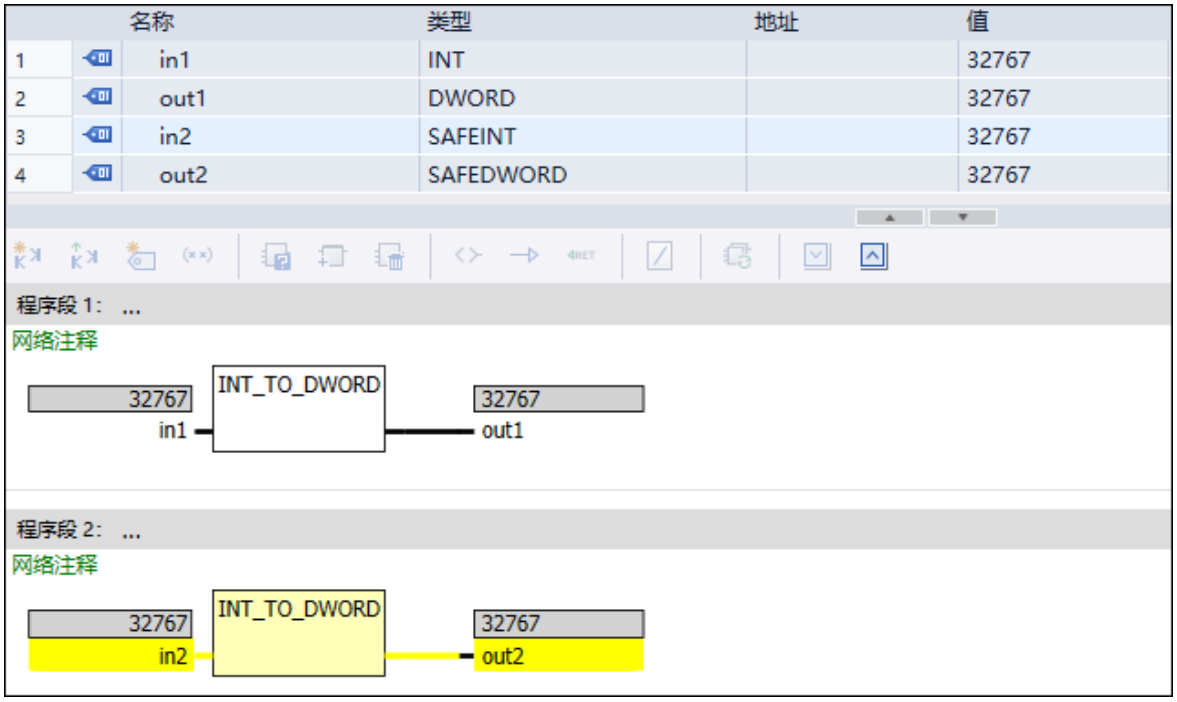
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	-	-	√	√	-	-	-	-
Out	-	-	-	-	-	-	√	√	-	-	-	-	-	-

- **Function description**

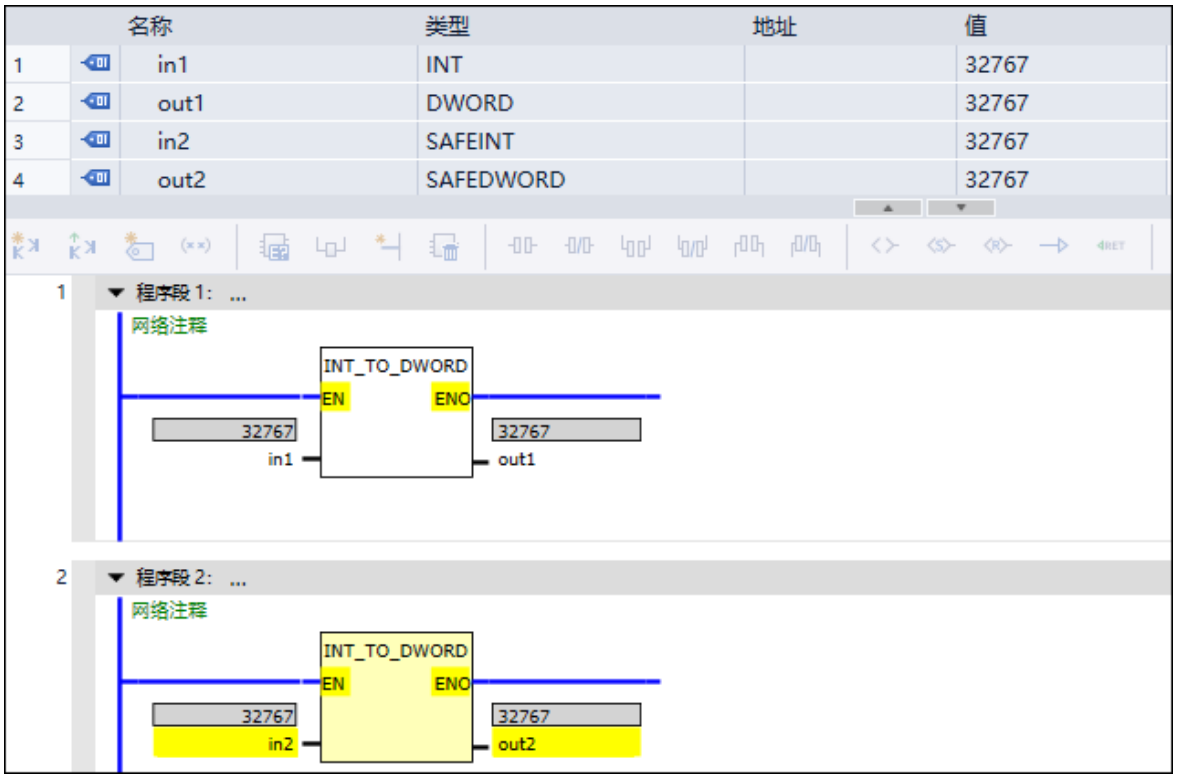
  Convert "In" of INT type to the bit string "Out" of DWORD type.

- **Programming example**

  FBD:



LD:

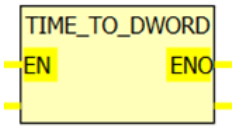
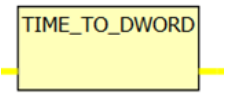


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.
  - If a value outside the range of INT#0 to 32767 is input, a "conversion exception" will occur and the program will stop.

11.10.1.6.42 TIME_TO_DWORD

Convert TIME type data to DWORD type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
TIME_TO_DWORD	Convert TIME type data to DWORD type	FC		

- **Related variables**

Variables

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	T#0ms to T#49d17h2m47s295ms	T#0ms	Conversion object
Output	Out	Conversion result	DWORD#0 to 4294967295	DWORD#0	Conversion result

Data types

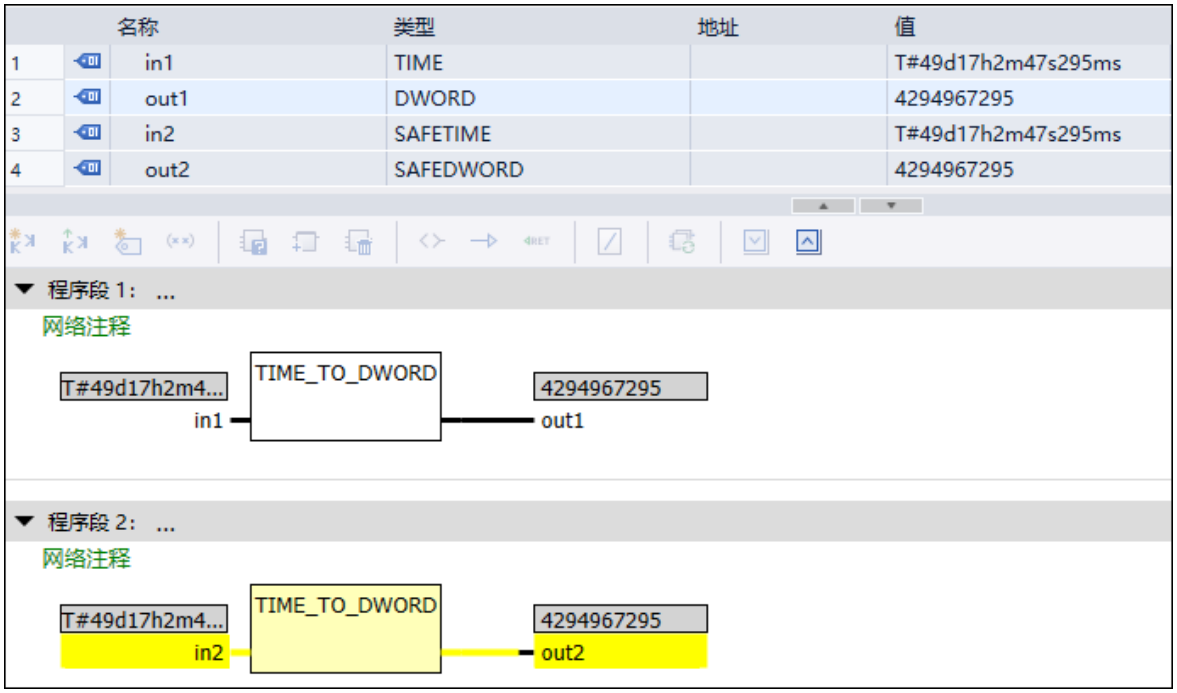
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	-	-	-	-	-	-	-	-	√	√
Out	-	-	-	-	-	-	√	√	-	-	-	-	-	-

- **Function description**

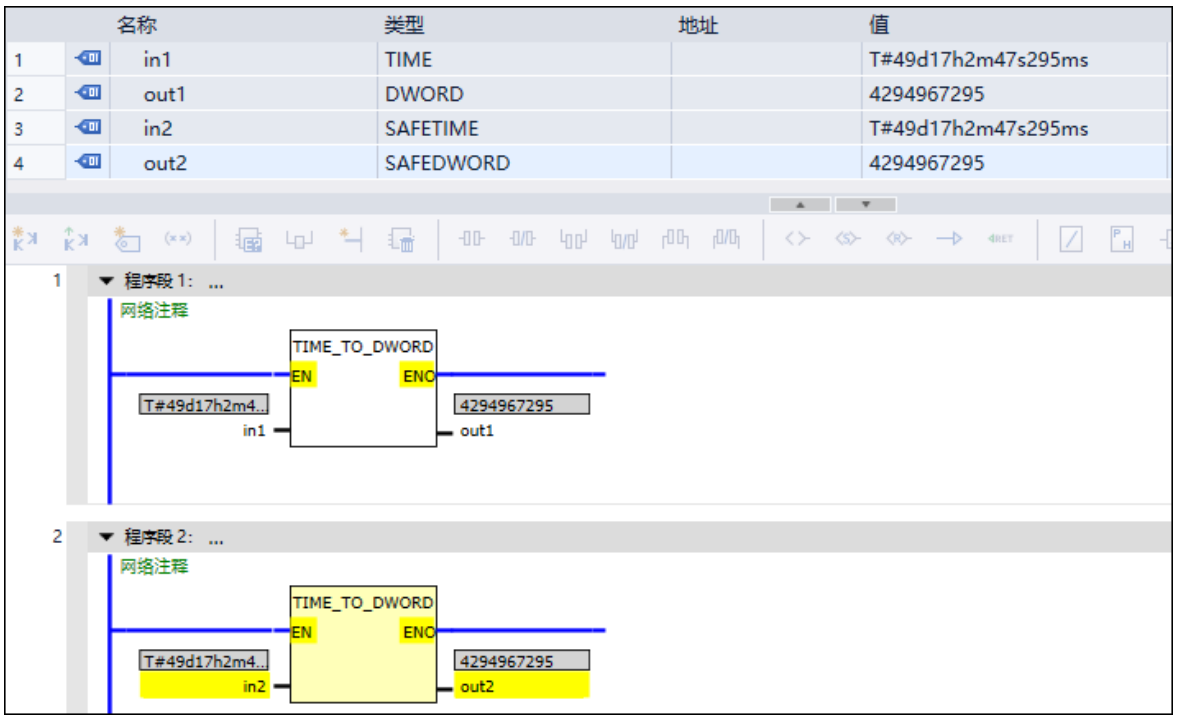
Convert "In" of TIME type to the bit string "Out" of DWORD type.

- **Programming example**

FBD:



LD:

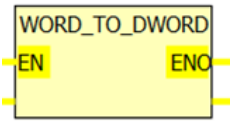
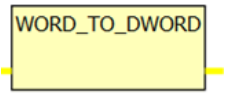


- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.1.6.43 WORD_TO_DWORD

Convert WORD type data to DWORD type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
WORD_TO_DWORD	Convert WORD type data to DWORD type	FC		

- **Related variables**

  Variables

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	WORD#16#0000 to FFFF	WORD#16#0000	Conversion object
Output	Out	Conversion result	DWORD#0 to 65535	DWORD#0	Conversion result

  Data types

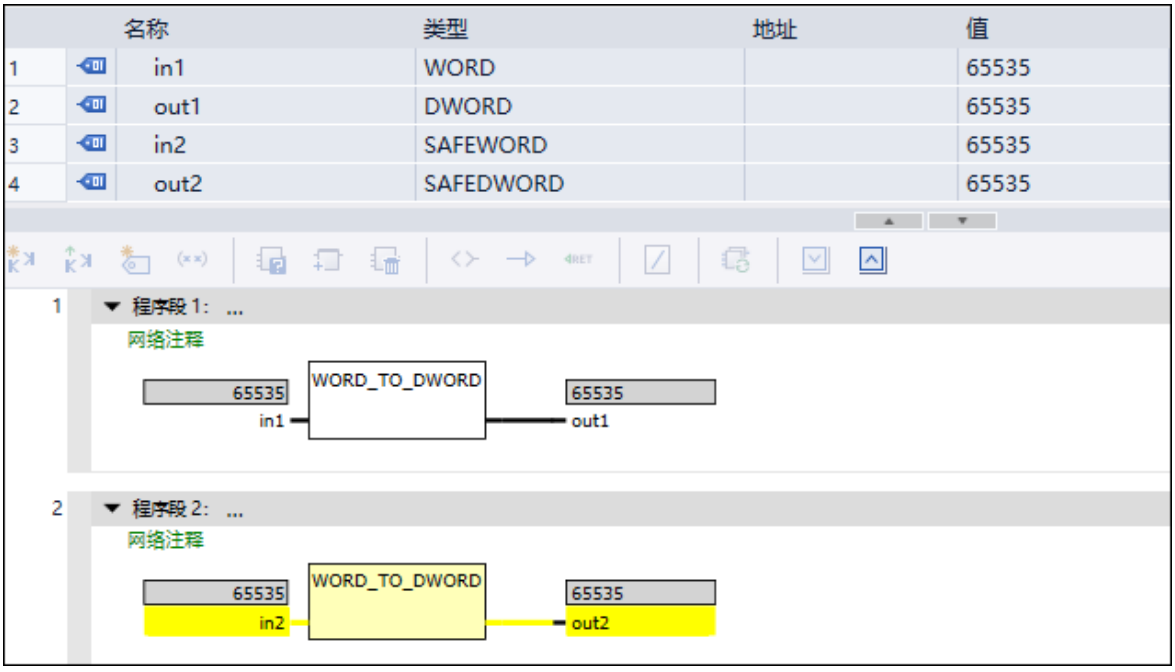
	Boolean		Bit String						Integer				Time Format	
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD	INT	SAFEINT	DINT	SAFEDINT	TIME	SAFETIME
In	-	-	-	-	√	√	-	-	-	-	-	-	-	-
Out	-	-	-	-	-	-	√	√	-	-	-	-	-	-

- **Function description**

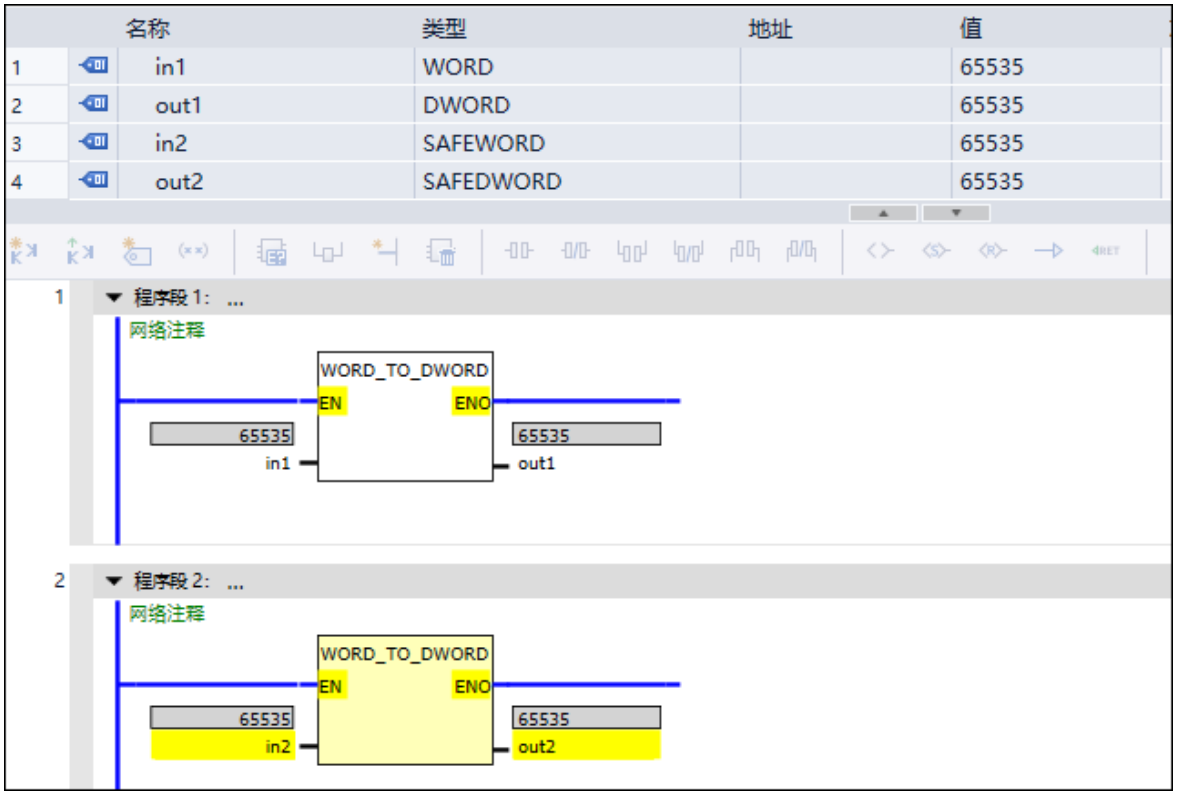
  Convert the bit string "In" of WORD type to the bit string "Out" of DWORD type.

- **Programming example**

  FBD:



  LD:



- **Precautions**
  - When assigning a safety data type variable to the output terminal, the input terminals must all be set to safety data type variables.
  - When assigning a standard data type variable to the output terminal, the input terminals can be set to either safety or standard data type variables.

11.10.2 Safety Instructions

11.10.2.1 Basic Safety Instruction

11.10.2.1.1 Instruction List

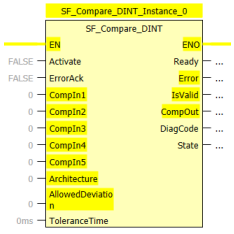
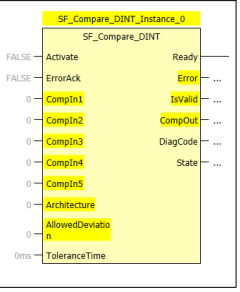
Instruction	FB/FC	Function	Library	Applicable Library Version
11.10.2.1.2 SF_Compare_DINT	FB	Comparison output instruction	SafetyInstructions	Applicable to library version 1.0.0.0/1.1.0.0
11.10.2.1.3 SF_Compare_INT	FB	Comparison output instruction		
11.10.2.1.4 SF_CTD	FB	Subtraction counter		
11.10.2.1.5 SF_CTU	FB	Addition counter		
11.10.2.1.6 SF_CTUD	FB	Addition/ subtraction counter		

Instruction	FB/FC	Function	Library	Applicable Library Version
11.10.2.1.7 SF_F_TRIG	FB	Falling edge triggered		
11.10.2.1.8 SF_Limit_DINT	FB	Limit		
11.10.2.1.9 SF_Limit_INT	FB	Limit		
11.10.2.1.10 SF_R_TRIG	FB	Rising edge-triggered		
11.10.2.1.11 SF_RS	FB	Prioritize the reset value		
11.10.2.1.12 SF_Scaling_DINT	FB	Proportional scaling		
11.10.2.1.13 SF_Scaling_INT	FB	Proportional scaling		
11.10.2.1.14 SF_SR	FB	Prioritize the set value		
11.10.2.1.15 SF_TON	FB	ON-delay timer		
11.10.2.1.16 SF_TOF	FB	OFF-delay timer		
11.10.2.1.17 SF_TP	FB	Time pulse		

11.10.2.1.2 SF_Compare_DINT

The SF_Compare_DINT instruction is used to check whether the two to five input parameters (of SAFEDINT type) are equal or differ by a specified allowable value within a set time range.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_Compare_DINT	Comparison output instruction	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Activate	Activate instruction	BOOL	TRUE, FALSE	FALSE	Determines whether to activate the instruction. FALSE: Instruction is not activated. TRUE: Instruction is activated.
	ErrorAck	Fault acknowledgment	BOOL	TRUE, FALSE	FALSE	Acknowledges faults. When a fault occurs, ErrorAck must be changed from FALSE to TRUE; otherwise, the instruction will remain in a faulty status.
	CompIn1	Comparative value 1	SAFEDINT	-2147483648 to 2147483647	0	The first value to be compared
	CompIn2	Comparative value 2	SAFEDINT	-2147483648 to 2147483647	0	The second value to be compared
	CompIn3	Comparative value 3	SAFEDINT	-2147483648 to 2147483647	0	The third value to be compared
	CompIn4	Comparative value 4	SAFEDINT	-2147483648 to 2147483647	0	The fourth value to be compared
	CompIn5	Comparative value 5	SAFEDINT	-2147483648 to 2147483647	0	The fifth value to be compared
	Architecture	Comparison output mode	SAFEDINT	0, 1, 2	0	Comparison output mode. Out-of-range values are corrected to mode 0.
	AllowedDeviation	Allowed deviation	SAFEDINT	-2147483648 to 2147483647	0	Allowed deviation. The comparison is based on the absolute value of the allowed deviation.
	Tolerance Time	Tolerance time	TIME	Follows data type	T#0ms	The maximum duration for which the input values remain unequal.
Output	Ready	Ready status	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	Error	Faulty status	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is faulty.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	IsValid	Input values are equal	SAFEBOOL	TRUE, FALSE	FALSE	Returns TRUE if the input values are equal (within tolerance range). Returns TRUE if no fault occurs within the allowed time.
	CompOut	Comparison output value	SAFEDINT	-2147483648 to 2147483647	0	Comparison output value. Displays the first value that meets the requirement.
	DiagCode	Instruction information code	16#WORD	Follows data type	0	Displays fault codes when an instruction fault occurs, to indicate the cause of the fault.
	State	Instruction operation status	WORD	Follows data type	0	Indicates the execution status of the instruction. The following statuses are available for the current instruction: 1: Run 2: Stop 4: Error 5: Reset 8: DelayOut

- **Function description**
  - The SF_Compare_DINT instruction is used to check whether the two to five input parameters are equal within a set time range.
  - The comparison mode "Architecture" determines how many input values are used for comparison.

Architecture (Comparison Mode)	Comparison Mode
0: 1oo2	Checks whether the first two input values are equal (within the allowed range).
1: 2oo3	Checks whether two of the first three input values are equal (within the allowed range).
2: 3oo5	Checks whether three of the first five input values are equal (within the allowed range).

    The output parameter "CompOut" returns the first input value that meets the requirement under each comparison mode. If a valid output is found, the output parameter "IsValid" is set to TRUE; if a fault occurs, "IsValid" is reset to FALSE.
  - The instruction information code (DiagCode) is used to display fault codes to indicate the causes.

DiagCode (Instruction Information Code)	Description
0x4098	Input value exceeds the maximum value of the DINT type.
0x4099	Input value is less than the minimum value of the DINT type.
0x0217	Input values remain unequal (within the allowed range) for a time longer than "ToleranceTime".

◦ Status information (State) is used to indicate the execution status of the instruction. In the current instruction, the statuses are: 1 (Run), 2 (Stop), 4 (Error), 5 (Reset), and 8 (DelayOut).

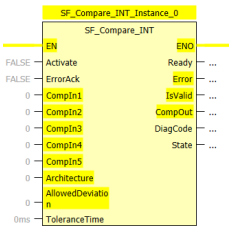
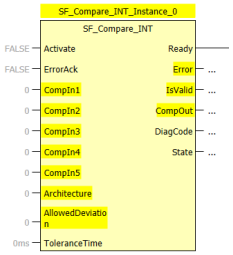
State (Status Information)	Description
1 (Run)	The instruction enters the "Run" status if a sufficient number of input values are equal (within allowable range). Output values are as follows: Error = 0, IsValid = 1, CompOut = CompInX (the first value that meets the requirement)
2 (Stop)	In the "Stop" status, the instruction is not executed. Output values are as follows: Error = 0, IsValid = 0, CompOut = 0.
4 (Error)	In the "Error" status, a fault is detected during instruction execution. Output values are as follows: Error = 1, IsValid = 0, CompOut = 0.
5 (Reset)	In the "Reset" status, no further faults remain and "ErrorAck" changes from FALSE to TRUE. Output values are as follows: Error = 0, IsValid = 0, CompOut = 0.
8 (DelayOut)	If there are not enough input values that are equal (within allowable range), timing will start. If the set time is not exceeded, the instruction enters the "DelayOut" status. Output values are as follows: Error = 0, IsValid = 1, CompOut = retained value.

- **Note**
  : If the instruction execution fails, manual confirmation is required to restart the execution; if no manual confirmation is performed, the instruction status remains 4 (Error) and will not switch to 2 (Stop).

11.10.2.1.3 SF_Compare_INT

The SF_Compare_INT instruction is used to check whether the two to five input parameters (of SAFEINT type) are equal or differ by a specified allowable value within a set time range.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_Compare_INT	Comparison output instruction	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Activate	Activate instruction	BOOL	TRUE, FALSE	FALSE	Determines whether to activate the instruction. FALSE: Instruction is not activated. TRUE: Instruction is activated.
	ErrorAck	Fault acknowledgment	BOOL	TRUE, FALSE	FALSE	Acknowledges faults. When a fault occurs, ErrorAck must be changed from FALSE to TRUE; otherwise, the instruction will remain in a faulty status.
	CompIn1	Comparative value 1	SAFEINT	-32768 to 32767	0	The first value to be compared
	CompIn2	Comparative value 2	SAFEINT	-32768 to 32767	0	The second value to be compared
	CompIn3	Comparative value 3	SAFEINT	-32768 to 32767	0	The third value to be compared
	CompIn4	Comparative value 4	SAFEINT	-32768 to 32767	0	The fourth value to be compared
	CompIn5	Comparative value 5	SAFEINT	-32768 to 32767	0	The fifth value to be compared
	Architecture	Comparison output mode	SAFEINT	0, 1, 2	0	Comparison output mode. Out-of-range values are corrected to mode 0. 0: 1oo2 1: 2oo3 2: 3oo5

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	AllowedDeviation	Allowed deviation	SAFEINT	-32768 to 32767	0	Allowed deviation. The comparison is based on the absolute value of the allowed deviation.
	ToleranceTime	Tolerance time	TIME	Follows data type	T#0ms	The maximum duration for which the input values remain unequal.
Output	Ready	Ready status	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	Error	Faulty status	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is faulty.
	IsValid	Input values are equal	SAFEBOOL	TRUE, FALSE	FALSE	Returns TRUE if the input values are equal (within tolerance range). Returns TRUE if no fault occurs within the allowed time.
	CompOut	Comparison output value	SAFEINT	-32768 to 32767	0	Comparison output value. Displays the first value that meets the requirement.
	DiagCode	Instruction information code	16#WORD	Follows data type	0	Displays fault codes when an instruction fault occurs, to indicate the cause of the fault.
	State	Instruction operation status	WORD	Follows data type	0	Indicates the execution status of the instruction. The following statuses are available for the current instruction: 1: Run 2: Stop 4: Error 5: Reset 8: DelayOut

- **Function description**
  - The SF_Compare_INT instruction is used to check whether the two to five input parameters are equal within a set time range.
  - The comparison mode "Architecture" determines how many input values are used for comparison.

Architecture (Comparison Mode)	Comparison Mode
0: 1oo2	Checks whether the first two input values are equal (within the allowed range).
1: 2oo3	Checks whether two of the first three input values are equal (within the allowed range).
2: 3oo5	Checks whether three of the first five input values are equal (within the allowed range).

The output parameter "CompOut" returns the first input value that meets the requirement under each comparison mode. If a valid output is found, the output parameter "IsValid" is set to TRUE; if a fault occurs, "IsValid" is reset to FALSE.

◦ The instruction information code (DiagCode) is used to display fault codes to indicate the causes.

DiagCode (Instruction Information Code)	Description
0x4098	Input value exceeds the maximum value of the INT type.
0x4099	Input value is less than the minimum value of the INT type.
0x0217	Input values remain unequal (within the allowed range) for a time longer than "ToleranceTime".

◦ Status information (State) is used to indicate the execution status of the instruction. In the current instruction, the statuses are: 1 (Run), 2 (Stop), 4 (Error), 5 (Reset), and 8 (DelayOut).

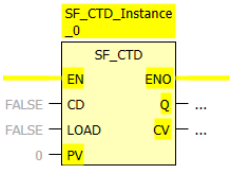
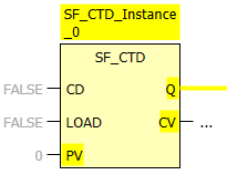
State (Status Information)	Description
1 (Run)	The instruction enters the "Run" status if a sufficient number of input values are equal (within allowable range). Output values are as follows: Error = 0, IsValid = 1, CompOut = CompInX (the first value that meets the requirement).
2 (Stop)	In the "Stop" status, the instruction is not executed. Output values are as follows: Error = 0, IsValid = 0, CompOut = 0.
4 (Error)	In the "Error" status, a fault is detected during instruction execution. Output values are as follows: Error = 1, IsValid = 0, CompOut = 0.
5 (Reset)	In the "Reset" status, no further faults remain and "ErrorAck" changes from FALSE to TRUE. Output values are as follows: Error = 0, IsValid = 0, CompOut = 0.
8 (DelayOut)	If there are not enough input values that are equal (within allowable range), timing will start. If the set time is not exceeded, the instruction enters the "DelayOut" status. Output values are as follows: Error = 0, IsValid = 1, CompOut = retained value.

- **Note**
  : If the instruction execution fails, manual confirmation is required to restart the execution; if no manual confirmation is performed, the instruction status remains 4 (Error) and will not switch to 2 (Stop).

11.10.2.1.4 SF_CTD

SF_CTD is a subtraction counter.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_CTD	Subtraction counter	FB		

- **Related variables**
  Variables

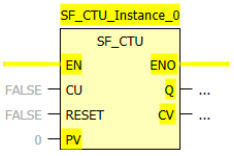
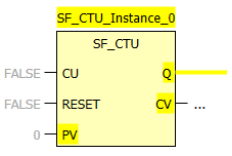
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	CD	Counter input	BOOL	TRUE, FALSE	FALSE	Counter input
	LOAD	Load signal	BOOL	TRUE, FALSE	FALSE	TRUE: Loads "PV" into "CV".
	PV	Preset value	SAFEINT	0 to 32767	0	Preset value of the counter
Output	Q	Counter output	SAFEBOOL	TRUE, FALSE	FALSE	Counter output
	CV	Count value	SAFEINT	0 to 32767	0	The current value of the counter

- **Function description**
  - SF_CTD is a subtraction counter. The data type of preset value and count value is SAFEINT.
  - When the load signal "LOAD" is set to TRUE, the preset value "PV" is loaded into the count value "CV", and the counter output "Q" becomes FALSE. When the value of "CV" is less than or equal to 0, the value of "Q" becomes TRUE.
  - A subtraction operation is triggered for "CV" when the input signal "CU" is at the rising edge.
  - When the value of "CV" is less than or equal to 0, even if "CD" is at a rising edge, "CV" will not change.
  - When "LOAD" is TRUE, "CD" is ignored. "CV" does not perform a subtraction operation.

11.10.2.1.5 SF_CTU

SF_CTU is an addition counter.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_CTU	Addition counter	FB		

- **Related variables**

  Variables

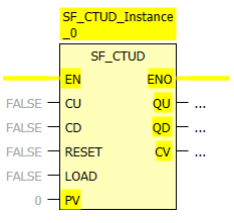
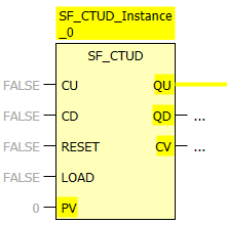
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	CU	Counter input	BOOL	TRUE, FALSE	FALSE	Counter input
	RESET	Reset signal	BOOL	TRUE, FALSE	FALSE	TRUE: Resets "CV" to 0.
	PV	Preset value	SAFEINT	0 to 32767	0	Preset value of the counter
Output	Q	Counter output	SAFEBOOL	TRUE, FALSE	FALSE	Counter output
	CV	Count value	SAFEINT	0 to 32767	0	The current value of the counter

- **Function description**
  - SF_CTU is an addition counter. The data type of preset value and count value is SAFEINT.
  - When the reset signal "RESET" is set to TRUE, the count value "CV" becomes 0, and the counter output "Q" becomes FALSE. When the value of "CV" is greater than or equal to the preset value "PV", the value of "Q" becomes TRUE.
  - An addition operation is triggered for "CV" when the input signal "CU" is at the rising edge.
  - When the value of "CV" exceeds the value of "PV", "CV" continues to perform addition operations at each rising edge of "CU" until it reaches the maximum value of 32767.
  - When "RESET" is TRUE, "CU" is ignored. "CV" does not perform an addition operation.

11.10.2.1.6 SF_CTUD

SF_CTUD is an addition/subtraction counter.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_CTUD	Addition/ subtraction counter	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	CU	Addition counter input	BOOL	TRUE, FALSE	FALSE	Addition counter input
	CD	Subtraction counter input	BOOL	TRUE, FALSE	FALSE	Subtraction counter input
	RESET	Reset signal	BOOL	TRUE, FALSE	FALSE	TRUE: Resets "CV" to 0.
	LOAD	Load signal	BOOL	TRUE, FALSE	FALSE	TRUE: Loads "PV" into "CV".
	PV	Preset value	SAFEINT	0 to 32767	0	The positive count value of the addition counter and the default value of the subtraction counter.
Output	QU	Addition counter output	SAFEBOOL	TRUE, FALSE	FALSE	Addition counter output
	QD	Subtraction counter output	SAFEBOOL	TRUE, FALSE	FALSE	Subtraction counter output
	CV	Count value	SAFEINT	0 to 32767	0	The current value of the counter

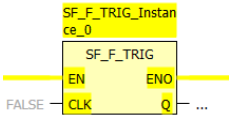
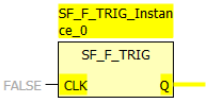
- **Function description**
  - CTUD is a counter that performs addition and subtraction operations based on the input signals. It combines the functions of both the addition counter and the subtraction counter. The data type of preset value and count value is SAFEINT.
  - Addition counter function
    - When the reset signal "RESET" is set to TRUE, the count value "CV" becomes 0, and the addition counter output "QU" becomes FALSE. When the value of "CV" is greater than or equal to the preset value "PV", the value of "QU" becomes TRUE.
    - An addition operation is triggered for "CV" when the input signal "CU" is at the rising edge.
    - When the value of "CV" exceeds the value of "PV", "CV" continues to perform addition operations at each rising edge of "CU" until it reaches the maximum value of 32767.
  - Subtraction counter function
    - When the load signal "LOAD" is set to TRUE, the preset value "PV" is loaded into the count value "CV", and the subtraction counter output "QD" becomes FALSE. When the value of "CV" is less than or equal to 0, the value of "QD" becomes TRUE.
    - A subtraction operation is triggered for "CV" when the input signal "CD" is at the rising edge.
    - When the value of "CV" is less than or equal to 0, even if "CD" is at a rising edge, "CV" will not change.
  - Common functions of the addition/subtraction counter
    - When "LOAD" or "RESET" is TRUE, "CU" and "CD" are ignored, and "CV" does not perform addition or subtraction operations.
    - When "CU" and "CD" are both at the rising edge, "CV" does not change.
    - When both "RESET" and "LOAD" are TRUE, "RESET" takes priority, and the value of "CV" becomes 0.
    - When "RESET" is set to TRUE, the value of "CV" becomes 0, and the value of "QD" becomes TRUE.

- When "LOAD" is set to TRUE, the value of "CV" becomes equal to "PV", and the value of "QU" becomes TRUE.

11.10.2.1.7 SF_F_TRIG

SF_F_TRIG is a falling edge-triggered instruction.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_F_TRIG	Falling edge triggered	FB		

- **Related variables**

  Variables

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	CLK	Input signal	SAFEBOOL	TRUE, FALSE	FALSE	Input signal
Output	Q	Output signal	SAFEBOOL	TRUE, FALSE	FALSE	Output signal

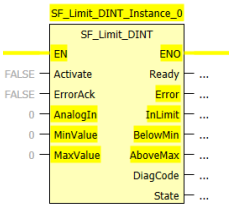
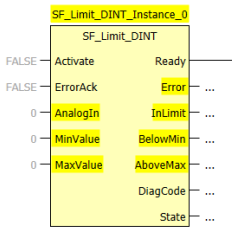
- **Function description**

  When the input signal "CLK" is at a falling edge (TRUE→FALSE), the output signal "Q" is set to TRUE for only one task cycle. In all other cases, the value of "Q" is FALSE.

11.10.2.1.8 SF_Limit_DINT

The SF_Limit_DINT instruction checks whether the input parameter AnalogIn (of SAFEDINT type) is within the range defined by the input parameters MinValue and MaxValue. If the value of AnalogIn is within the limits, the output parameter InLimit is set to TRUE. Otherwise, depending on the value of AnalogIn, the output parameter BelowMin or AboveMax is set to TRUE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_Limit_DINT	Limit	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Activate	Activate instruction	BOOL	TRUE, FALSE	FALSE	Determines whether to activate the instruction. FALSE: Instruction is not activated. TRUE: Instruction is activated.
	ErrorAck	Fault acknowledgment	BOOL	TRUE, FALSE	FALSE	Acknowledges faults. When a fault occurs, ErrorAck must be changed from FALSE to TRUE; otherwise, the instruction will remain in a faulty status.
	AnalogIn	Input value	SAFEDINT	-2147483648 to 2147483647	0	The value for range check
	MinValue	Minimum value	SAFEDINT	-2147483648 to 2147483647	0	Minimum value
	MaxValue	Maximum value	SAFEDINT	-2147483648 to 2147483647	0	Maximum value
Output	Ready	Ready status	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	Error	Faulty status	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is faulty.
	InLimit	Input range flag	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether the input value is within the set range.
	BelowMin	Input below the lower limit	SAFEBOOL	TRUE, FALSE	FALSE	The value of "AnalogIn" is less than the set minimum value.
	AboveMax	Input exceeding the upper limit	SAFEBOOL	TRUE, FALSE	FALSE	The value of "AnalogIn" exceeds the set maximum value.
	DiagCode	Instruction information code	16#WORD	Follows data type	0	Displays fault codes when an instruction fault occurs, to indicate the cause of the fault.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	State	Instruction operation status	WORD	Follows data type	0	Indicates the execution status of the instruction. The following statuses are available for the current instruction: 1: Run 2: Stop 3: Safe 4: Error 5: Reset

- **Function description**
  - The SF_Limit_DINT instruction checks whether the input parameter AnalogIn is within the range defined by the input parameters MinValue and MaxValue. If the value of AnalogIn is within the limits, the output parameter InLimit is set to TRUE. Otherwise, depending on the value of AnalogIn, the output parameter BelowMin or AboveMax is set to TRUE.
  - The instruction information code (DiagCode) is used to display fault codes to indicate the causes.

DiagCode (Instruction Information Code)	Description
0x4090	The input parameter MinValue is greater than MaxValue.

  - Status information (State) is used to indicate the execution status of the instruction. In the current instruction, the statuses are: 1 (Run), 2 (Stop), 3 (Safe), 4 (Error), and 5 (Reset).

State (Status Information)	Description
1 (Run)	If AnalogIn is greater than or equal to MinValue and less than or equal to MaxValue, the instruction enters the "Run" status. Output values are as follows: Error = 0; InLimit = 1; BelowMin = 0; AboveMax = 0
2 (Stop)	In the "Stop" status, the instruction is not executed. Output values are as follows: Error = 0, InLimit = 0, BelowMin = 0, AboveMax = 0.
3 (Safe)	In the "Safe" status, AnalogIn is less than MinValue or greater than MaxValue. Output values are as follows: Error = 0, InLimit = 0, BelowMin = (less than MinValue), AboveMax = (greater than MaxValue).
4 (Error)	In the "Error" status, a fault is detected during instruction execution. Output values are as follows: Error = 1, InLimit = 0, BelowMin = 0, AboveMax = 0

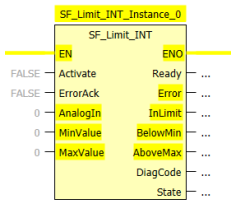
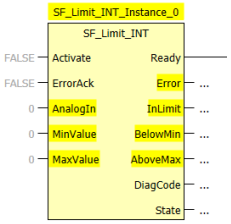
State (Status Information)	Description
5 (Reset)	In the "Reset" status, no further faults remain and "ErrorAck" changes from FALSE to TRUE. Output values are as follows: Error = 0, InLimit = 0, BelowMin = 0, AboveMax = 0.

- **Note**
  : If the instruction execution fails, manual confirmation is required to restart the execution; if no manual confirmation is performed, the instruction status remains 4 (Error) and will not switch to 2 (Stop).

11.10.2.1.9 SF_Limit_INT

The SF_Limit_INT instruction checks whether the input parameter AnalogIn (of SAFEINT type) is within the range defined by the input parameters MinValue and MaxValue. If the value of AnalogIn is within the limits, the output parameter InLimit is set to TRUE. Otherwise, depending on the value of AnalogIn, the output parameter BelowMin or AboveMax is set to TRUE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_Limit_INT	Limit	FB		

- **Related variables**

Variables

Input / Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Activate	Activate instruction	BOOL	TRUE, FALSE	FALSE	Determines whether to activate the instruction. FALSE: Instruction is not activated. TRUE: Instruction is activated.
	ErrorAck	Fault acknowledgment	BOOL	TRUE, FALSE	FALSE	Acknowledges faults. When a fault occurs, ErrorAck must be changed from FALSE to TRUE; otherwise, the instruction will remain in a faulty status.
	AnalogIn	Input value	SAFEINT	-32768 to 32767	0	The value for range check
	MinValue	Minimum value	SAFEINT	-32768 to 32767	0	Minimum value

Input / Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	MaxValue	Maximum value	SAFEINT	-32768 to 32767	0	Maximum value
Output	Ready	Ready status	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	Error	Faulty status	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is faulty.
	InLimit	Input range flag	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether the input value is within the set range.
	BelowMin	Input below the lower limit	SAFEBOOL	TRUE, FALSE	FALSE	The value of "AnalogIn" is less than the set minimum value.
	AboveMax	Input exceeding the upper limit	SAFEBOOL	TRUE, FALSE	FALSE	The value of "AnalogIn" exceeds the set maximum value.
	DiagCode	Instruction information code	16#WORD	Follows data type	0	Displays fault codes when an instruction fault occurs, to indicate the cause of the fault.
	State	Instruction operation status	WORD	Follows data type	0	Indicates the execution status of the instruction. The following statuses are available for the current instruction: 1: Run 2: Stop 3: Safe 4: Error 5: Reset

- **Function description**
  - The SF_Limit_INT instruction checks whether the input parameter AnalogIn is within the range defined by the input parameters MinValue and MaxValue. If the value of AnalogIn is within the limits, the output parameter InLimit is set to TRUE. Otherwise, depending on the value of AnalogIn, the output parameter BelowMin or AboveMax is set to TRUE.
  - The instruction information code (DiagCode) is used to display fault codes to indicate the causes.

DiagCode (Instruction Information Code)	Description
0x4090	The input parameter MinValue is greater than MaxValue.

Status information (State) is used to indicate the execution status of the instruction. In the current instruction, the statuses are: 1 (Run), 2 (Stop), 3 (Safe), 4 (Error), and 5 (Reset).

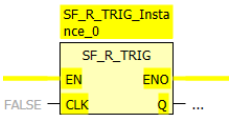
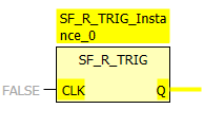
State (Status Information)	Description
1 (Run)	If AnalogIn is greater than or equal to MinValue and less than or equal to MaxValue, the instruction enters the "Run" status. Output values are as follows: Error = 0; InLimit = 1; BelowMin = 0; AboveMax = 0
2 (Stop)	In the "Stop" status, the instruction is not executed. Output values are as follows: Error = 0, InLimit = 0, BelowMin = 0, AboveMax = 0.
3 (Safe)	In the "Safe" status, AnalogIn is less than MinValue or greater than MaxValue. Output values are as follows: Error = 0, InLimit = 0, BelowMin = (less than MinValue), AboveMax = (greater than MaxValue).
4 (Error)	In the "Error" status, a fault is detected during instruction execution. Output values are as follows: Error = 1, InLimit = 0, BelowMin = 0, AboveMax = 0
5 (Reset)	In the "Reset" status, no further faults remain and "ErrorAck" changes from FALSE to TRUE. Output values are as follows: Error = 0, InLimit = 0, BelowMin = 0, AboveMax = 0.

- **Note**
  : If the instruction execution fails, manual confirmation is required to restart the execution; if no manual confirmation is performed, the instruction status remains 4 (Error) and will not switch to 2 (Stop).

11.10.2.1.10 SF_R_TRIG

SF_R_TRIG is a rising edge-triggered instruction.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_R_TRIG	Rising edge-triggered	FB		

- **Related variables**

  Variables

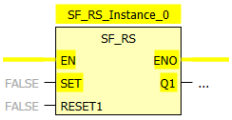
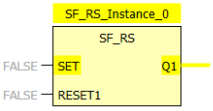
Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	CLK	Input signal	SAFEBOOL	TRUE, FALSE	FALSE	Input signal
Output	Q	Output signal	SAFEBOOL	TRUE, FALSE	FALSE	Output signal

- **Function description**
  When the input signal "CLK" is at a rising edge (FALSE→TRUE), the output signal "Q" is set to TRUE for only one task cycle. In all other cases, the value of "Q" is FALSE.

11.10.2.1.11 SF_RS

SF_RS is a reset-priority self-holding instruction.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_RS	Prioritize the reset value	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	SET	Set a bit	SAFEBOOL	TRUE, FALSE	FALSE	Set input
	RESET1	Reset a bit	BOOL	TRUE, FALSE	FALSE	Reset input (prioritized)
Output	Q1	Self-holding	SAFEBOOL	TRUE, FALSE	FALSE	Self-holding output

- **Function description**

  SF_RS is a reset-priority self-holding instruction. The input/output relationship is as follows.

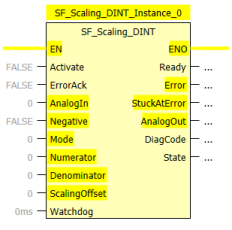
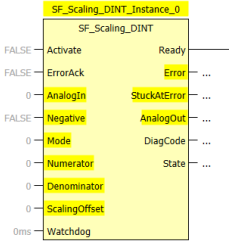
"SET" Value	"RESET1" Value	"Q1" Value
TRUE	TRUE	FALSE
TRUE	FALSE	TRUE
FALSE	TRUE	FALSE
FALSE	FALSE	Remain

  - As long as the value of "RESET1" is TRUE, the value of "Q1" is FALSE.
  - When the value of "RESET1" is FALSE and the value of "SET" is TRUE, the value of "Q1" is TRUE.

11.10.2.1.12 SF_Scaling_DINT

The SF_Scaling_DINT function performs the calculation "AnalogIn * Numerator / Denominator + ScallingOffset" on the input value AnalogIn (of SAFEINTDINT type), and rounds the result according to the Mode (rounding method), outputting the final calculated value through AnalogOut.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_Scaling_DINT	Proportional scaling	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Activate	Activate instruction	BOOL	TRUE, FALSE	FALSE	Determines whether to activate the instruction. FALSE: Instruction is not activated. TRUE: Instruction is activated.
	ErrorAck	Fault acknowledgment	BOOL	TRUE, FALSE	FALSE	Acknowledges faults. When a fault occurs, ErrorAck must be changed from FALSE to TRUE; otherwise, the instruction will remain in a faulty status.
	AnalogIn	Value to be scaled	SAFEDINT	-2147483648 to 2147483647	0	The value that needs to be scaled proportionally
	Negative	Inversion flag	SAFEBOOL	TRUE, FALSE	FALSE	Inversion flag. FALSE: The input value is not inverted. TRUE: The input value is inverted first and then scaled proportionally.
	Mode	Rounding method	SAFEDINT	0, 1, 2	0	Rounding method 0: Round down 1: Round up 2: Round to nearest
	Numerator	Numerator of the scaling ratio	SAFEDINT	-2147483648 to 2147483647	1	Numerator of the scaling ratio
	Denominator	Denominator of the scaling ratio	SAFEDINT	-2147483648 to 2147483647	1	Denominator of the scaling ratio
	ScalingOffset	Scaling offset	SAFEDINT	-2147483648 to 2147483647	0	Scaling offset

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Watchdog	Detection time	TIME	Follows data type	T#0ms	Detection time. Used to detect whether the input value has changed within the time range.
Output	Ready	Ready status	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	Error	Faulty status	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is faulty.
	StuckAtError	Input stable flag	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether the input value has remained unchanged within the specified time range.
	AnalogOut	Output value	SAFEDINT	-2147483648 to 2147483647	0	Output value after scaling.
	DiagCode	Instruction information code	16#WORD	Follows data type	0	Displays fault codes when an instruction fault occurs, to indicate the cause of the fault.
	State	Instruction operation status	WORD	Follows data type	0	Indicates the execution status of the instruction. The following statuses are available for the current instruction: 1: Run 2: Stop 4: Error 5: Reset

- **Function description**
  - The SF_Scaling_DINT function performs the calculation "AnalogIn * Numerator / Denominator + ScallingOffset" on the input value AnalogIn, and rounds the result according to the Mode (rounding method), outputting the final calculated value through AnalogOut. The input parameter "Negative" can be used to invert "AnalogIn" before scaling.
  - The input parameter "Mode" can determine the rounding method after scaling. If set to a value greater than 2, it will take effect as 0.

Mode (Rounding Mode)	Rounding Method
0: Round up	Returns the closest integer that is greater than or equal to the input. For example, returns 4 for 3.4 and -3 for -3.4.

Mode (Rounding Mode)	Rounding Method
1: Round down	Returns the closest integer that is less than or equal to the input. For example, returns 3 for 3.4 and -4 for -3.4.
2: Round to nearest	Returns the closest integer. For example, returns 3 for 3.4, 4 for 3.5, -3 for -3.4, and -4 for -3.5.

◦ The instruction information code (DiagCode) is used to display fault codes to indicate the causes.

DiagCode (Instruction Information Code)	Description
0xC0B1	The output result exceeds the maximum value of the DINT type.
0xC0B2	The output result is less than the minimum value of the DINT type.

◦ Status information (State) is used to indicate the execution status of the instruction. In the current instruction, the statuses are: 1 (Run), 2 (Stop), 4 (Error), and 5 (Reset).

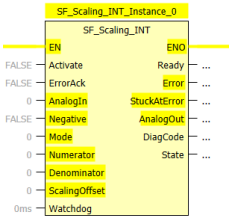
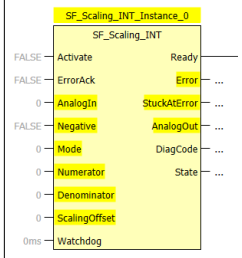
State (Status Information)	Description
1 (Run)	In the "Run" status, the scaling result is output to "AnalogOut". Output values are as follows: Error = 0; AnalogOut = Scaling result; StuckAtError = Detected StuckAtError
2 (Stop)	In the "Stop" status, the instruction is not executed. Output values are as follows: Error = 0; AnalogOut = 0; StuckAtError = 0
4 (Error)	In the "Error" status, a fault is detected during instruction execution. Output values are as follows: Error = 1; AnalogOut = 0; StuckAtError = 0
5 (Reset)	In the "Reset" status, no further faults remain and "ErrorAck" changes from FALSE to TRUE. Output values are as follows: Error = 0; AnalogOut = 0; StuckAtError = 0

- **Precautions**
  - ◦ If the instruction execution fails, manual confirmation is required to restart the execution; if no manual confirmation is performed, the instruction status remains 4 (Error) and will not switch to 2 (Stop).
  - ◦ If the denominator of the scaling ratio is bound to a variable or is an immediate value, the value will be automatically corrected to 1 when it is 0, but the corrected value will not be visible in the iFA interface.

11.10.2.1.13 SF_Scaling_INT

The SF_Scaling_INT function performs the calculation "AnalogIn * Numerator / Denominator + ScallingOffset" on the input value AnalogIn (of SAFEINT type), and rounds the result according to the Mode (rounding method), outputting the final calculated value through AnalogOut.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_Scaling_INT	Proportional scaling	FB		

- **Related variables**

Variables

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Activate	Activate instruction	BOOL	TRUE, FALSE	FALSE	Determines whether to activate the instruction. FALSE: Instruction is not activated. TRUE: Instruction is activated.
	ErrorAck	Fault acknowledgment	BOOL	TRUE, FALSE	FALSE	Acknowledges faults. When a fault occurs, ErrorAck must be changed from FALSE to TRUE; otherwise, the instruction will remain in a faulty status.
	AnalogIn	Value to be scaled	SAFEINT	-32768 to 32767	0	The value that needs to be scaled proportionally
	Negative	Inversion flag	SAFEBOOL	TRUE, FALSE	FALSE	Inversion flag. FALSE: The input value is not inverted. TRUE: The input value is inverted first and then scaled proportionally.
	Mode	Rounding method	SAFEINT	0, 1, 2	0	Rounding method 0: Round down 1: Round up 2: Round to nearest
	Numerator	Numerator of the scaling ratio	SAFEINT	-32768 to 32767	1	Numerator of the scaling ratio

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Denominator	Denominator of the scaling ratio	SAFEINT	-32768 to 32767	1	Denominator of the scaling ratio
	ScalingOffset	Scaling offset	SAFEINT	-32768 to 32767	0	Scaling offset
	Watchdog	Detection time	TIME	Follows data type	T#0ms	Detection time. Used to detect whether the input value has changed within the time range.
Output	Ready	Ready status	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	Error	Faulty status	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is faulty.
	StuckAtError	Input stable flag	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether the input value has remained unchanged within the specified time range.
	AnalogOut	Output value	SAFEINT	-32768 to 32767	0	Output value after scaling.
	DiagCode	Instruction information code	16#WORD	Follows data type	0	Displays fault codes when an instruction fault occurs, to indicate the cause of the fault.
	State	Instruction operation status	WORD	Follows data type	0	Indicates the execution status of the instruction. The following statuses are available for the current instruction: 1: Run 2: Stop 4: Error 5: Reset

- **Function description**
  - The SF_Scaling_INT function performs the calculation "AnalogIn * Numerator / Denominator + ScallingOffset" on the input value AnalogIn, and rounds the result according to the Mode (rounding method), outputting the final calculated value through AnalogOut. The input parameter "Negative" can be used to invert "AnalogIn" before scaling.
  - The input parameter "Mode" can determine the rounding method after scaling. If set to a value greater than 2, it will take effect as 0.

Mode (Rounding Mode)	Rounding Method
0: Round up	Returns the closest integer that is greater than or equal to the input. For example, returns 4 for 3.4 and -3 for -3.4.
1: Round down	Returns the closest integer that is less than or equal to the input. For example, returns 3 for 3.4 and -4 for -3.4.
2: Round to nearest	Returns the closest integer. For example, returns 3 for 3.4, 4 for 3.5, -3 for -3.4, and -4 for -3.5.

◦ The instruction information code (DiagCode) is used to display fault codes to indicate the causes.

DiagCode (Instruction Information Code)	Description
0xC0B1	Output result exceeds the maximum value of the INT type.
0xC0B2	Output result is less than the minimum value of the INT type.

◦ Status information (State) is used to indicate the execution status of the instruction. In the current instruction, the statuses are: 1 (Run), 2 (Stop), 4 (Error), and 5 (Reset).

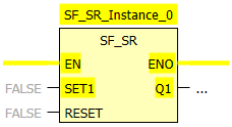
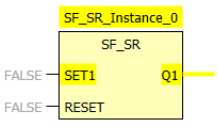
State (Status Information)	Description
1 (Run)	In the "Run" status, the scaling result is output to "AnalogOut". Output values are as follows: Error = 0; AnalogOut = Scaling result; StuckAtError = Detected StuckAtError
2 (Stop)	In the "Stop" status, the instruction is not executed. Output values are as follows: Error = 0; AnalogOut = 0; StuckAtError = 0
4 (Error)	In the "Error" status, a fault is detected during instruction execution. Output values are as follows: Error = 1; AnalogOut = 0; StuckAtError = 0
5 (Reset)	In the "Reset" status, no further faults remain and "ErrorAck" changes from FALSE to TRUE. Output values are as follows: Error = 0; AnalogOut = 0; StuckAtError = 0

- **Precautions**
  - ◦ If the instruction execution fails, manual confirmation is required to restart the execution; if no manual confirmation is performed, the instruction status remains 4 (Error) and will not switch to 2 (Stop).
  - ◦ If the denominator of the scaling ratio is bound to a variable or is an immediate value, the value will be automatically corrected to 1 when it is 0, but the corrected value will not be visible in the iFA interface.

11.10.2.1.14 SF_SR

SF_SR is a set-priority self-holding instruction.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_SR	Prioritize the set value	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	SET1	Set a bit	SAFEBOOL	TRUE, FALSE	FALSE	Set input (prioritized)
	RESET	Reset a bit	BOOL	TRUE, FALSE	FALSE	Reset output
Output	Q1	Self-holding	SAFEBOOL	TRUE, FALSE	FALSE	Self-holding output

- **Function description**

  SF_SR is a set-priority self-holding instruction. The input/output relationship is as follows.

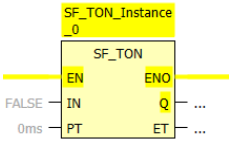
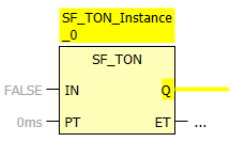
"SET1" Value	"RESET" Value	"Q1" Value
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	FALSE
FALSE	FALSE	Self-holding

  - As long as the value of "SET1" is "TRUE", the value of "Q1" is "TRUE".
  - When the value of "SET1" is "FALSE" and the value of "RESET" is "TRUE", the value of "Q1" is "FALSE".

11.10.2.1.15 SF_TON

SF_TON is a timer that outputs TRUE after a set time since startup.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_TON	ON-delay timer	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	IN	Timer input	BOOL	TRUE, FALSE	FALSE	TRUE: Starts timer. FALSE: Resets timer.

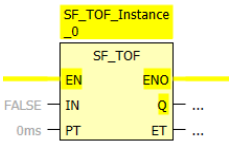
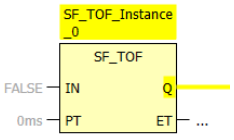
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	PT	Set time	TIME	Follows data type	TIME#0ms	The time from when the timer starts to when "Q" becomes TRUE
Output	Q	Timer output	SAFEBOOL	TRUE, FALSE	FALSE	TRUE: "ET" reaches "PT". FALSE: "ET" is less than "PT" when "IN" is TRUE or after the timer starts.
	ET	Elapsed time	TIME	Follows data type	TIME#0ms	Elapsed time after timer starts

- **Function description**
  - SF_TON is a timer that outputs TRUE after a set time since startup. The set time unit is ms.
  - When the timer input "IN" changes from FALSE to TRUE, the timer starts and "ET" continuously increases until it reaches the preset time "PT".
  - When "ET" reaches the set time "PT", the timer output "Q" becomes TRUE, and at this point, "ET" stops increasing.
  - When "IN" becomes FALSE, the timer resets. "ET" becomes 0, and "Q" becomes FALSE. Even before "ET" reaches "PT", if "IN" becomes FALSE after the timer starts, the timer resets.

11.10.2.1.16 SF_TOF

SF_TOF is a timer that outputs FALSE after a set time since startup.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_TOF	OFF-delay timer	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	IN	Timer input	BOOL	TRUE, FALSE	FALSE	TRUE: Resets timer. FALSE: Starts timer.
	PT	Set time	TIME	Follows data type	TIME#0ms	The time from when the timer starts to when "Q" becomes FALSE

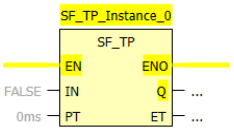
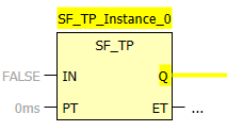
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	Q	Timer output	SAFEBOOL	TRUE, FALSE	FALSE	TRUE: "ET" is less than "PT" when "IN" is TRUE or after the timer starts. FALSE: "ET" reaches "PT".
	ET	Elapsed time	TIME	Follows data type	TIME#0ms	Elapsed time after timer starts

- **Function description**
  - SF_TOF is a timer that outputs FALSE after a set time since startup. The set time unit is ms.
  - When the timer input "IN" changes from TRUE to FALSE, the timer starts and "ET" continuously increases until it reaches the preset time "PT".
  - When "ET" reaches "PT", the timer output "Q" becomes FALSE. At this point, "ET" stops increasing.
  - When "IN" becomes TRUE, the timer resets. "ET" becomes 0, and "Q" becomes TRUE. Even before "ET" reaches "PT", if "IN" becomes TRUE after the timer starts, the timer resets.

11.10.2.1.17 SF_TP

SF_TP is a timer that outputs TRUE only for a set time after startup.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_TP	Time pulse	FB		

- **Related variables**

  Variables

Input / Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	IN	Timer input	BOOL	TRUE, FALSE	FALSE	TRUE: Starts timer. FALSE: Resets timer.
	PT	Set time	TIME	Follows data type	TIME#0ms	The time "Q" remains TRUE
Output	Q	Timer output	SAFEBOOL	TRUE, FALSE	FALSE	TRUE: "ET" is less than "PT" when "IN" is TRUE or after the timer starts. FALSE: "ET" reaches "PT".
	ET	Elapsed time	TIME	Follows data type	TIME#0ms	Elapsed time after timer starts

- **Function description**
  - SF_TP is a timer that outputs TRUE only for a set time after startup. The set time unit is ms.
  - When the timer input "IN" changes from FALSE to TRUE, the timer starts and timer output "Q" becomes TRUE. Elapsed time "ET" continuously increases until it reaches the set time "PT".
  - When "ET" reaches "PT", the timer output "Q" becomes FALSE. At this point, "ET" stops increasing.
  - When "IN" is FALSE, the timer resets and "ET" becomes 0. However, after the timer starts, if "IN" becomes FALSE before "ET" reaches "PT", the timer does not reset.

11.10.2.2 Safety Function Blocks

11.10.2.2.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
11.10.2.2.3 SF_Antivalent	FB	Inequivalent input	SafetyInstructions	Applicable to library version 1.0.0.0/1.1.0.0
11.10.2.2.4 SF_CamMonitoring	FB	Cam monitoring		
11.10.2.2.5 SF_CamshaftMonitor	FB	Cam shaft monitoring function		
11.10.2.2.6 SF_CycleControl	FB	Cycle control		
11.10.2.2.7 SF_EDM	FB	External monitoring		
11.10.2.2.8 SF_EmergencyStop	FB	Emergency stop.		
11.10.2.2.9 SF_EnableSwitch	FB	Enable switch		
11.10.2.2.10 SF_Equivalent	FB	Equivalent input		
11.10.2.2.11 SF_ESPE	FB	Electrically sensitive protective equipment		
11.10.2.2.12 SF_FootSwitch	FB	Foot switch		
11.10.2.2.13 SF_Guard	FB	Monitoring of safety-related protective devices		
11.10.2.2.14 SF_GuardLocking	FB	Safety door lock with interlock device		
11.10.2.2.15 SF_ModeSelector	FB	Mode selector		
11.10.2.2.16 SF_MutingPar	FB	Perform parallel muting with four muting sensors.		
11.10.2.2.17 SF_MutingPar_2Sensor	FB	Perform parallel muting with two muting sensors		
11.10.2.2.18 SF_MutingSeq	FB	Perform sequential muting with four muting sensors		

Instruction	FB/FC	Function	Library	Applicable Library Version
11.10.2.2.19 SF_OutControl	FB	Output control		
11.10.2.2.20 SF_Override	FB	Override muting function		
11.10.2.2.21 SF_PressControl	FB	Press control		
11.10.2.2.22 SF_PSE	FB	Pressure-sensitive equipment		
11.10.2.2.23 SF_ResetButton	FB	Reset button		
11.10.2.2.24 SF_SafetyRequest	FB	Safety request		
11.10.2.2.25 SF_ManualAck	FB	Safety confirmation		
11.10.2.2.26 SF_TempAlarm	FB	Temperature alarm		
11.10.2.2.27 SF_DataLog	FB	Safety logging function		
11.10.2.2.28 SF_TestableSafetySensor	FB	Testable safety sensors		
11.10.2.2.29 SF_TwoHandControlTypeII	FB	Type II two-hand control		
11.10.2.2.30 SF_TwoHandControlTypeIII	FB	Type III two-hand control		
11.10.2.2.31 SF_SafetyConnStatus	FB	Get safety C2C (CIP Safety) connection status		
11.10.2.2.32 SF_SafetyTaskInfo	FB	Get safety task information		
11.10.2.2.33 SF_GetSafetySignature	FB	Get safety signature		
11.10.2.2.34 SF_GetSystemMode	FB	Get safety PLC status		
11.10.2.2.35 SF_GetSafetyConfirmInfo	FB	Get safety confirmation information (including safety signature, safety confirmation time)		Applicable to library version 1.1.0.0
11.10.2.2.36 SF_GetCouplerState	FB	Get coupler status (EIP connection)		
11.10.2.2.37 SF_GetGlBusState	FB	Get local bus connection status (EIP device)		
11.10.2.2.38 SF_GetEtherNetIpStateCode	FB	Get EtherNet/IP (Scanner) connection status		
11.10.2.2.39 SF_GetFSoeStateCode	FB	Get FSoE connection status		
11.10.2.2.40 SF_GetCIPSafetyStateCode	FB	Get CIP Safety connection status code		

11.10.2.2.2 General Description of Safety Function Blocks

11.10.2.2.2.1 General Input Variables of Safety Function Blocks

Input Parameter	Data Type	Description
Activate	BOOL	The "Activate" parameter is a Boolean (BOOL) variable or constant used to control the activation status of the function block (FB), with an initial value of FALSE. This parameter must be connected to a variable reflecting the status (activated/deactivated) of the associated safety device. Dynamic association with the device status prevents irrelevant diagnostic information when the device is disabled. When the parameter value is FALSE, all output variables of the function block are forced to reset to their initial values, thereby achieving output isolation in a safety status. If no physical device is connected in the system, a constant TRUE signal must be assigned to this parameter to ensure the function block remains enabled in scenarios without device connection.
S_StartReset	SAFEBOOL	FALSE (initial value): Manual reset is required when the programmable electronic system (PES) is started (either warm start or cold start). TRUE: Automatic reset is performed when the PES is started (either warm start or cold start). This function may only be enabled when it is ensured that no hazards will occur during PES startup. Therefore, if the automatic circuit reset feature of the function block is to be used, other system or application measures must be implemented to prevent accidental (or unintended) startup.
S_AutoReset	SAFEBOOL	The reset mode is determined by a variable or constant. When the initial value is set to FALSE, manual reset is required after the emergency stop button is released; if set to TRUE, automatic reset is triggered after the emergency stop button is released. Note that the automatic reset function shall only be activated on the premise that releasing the emergency stop button will not cause unintended device restart. Therefore, when using the automatic circuit reset feature of the function block, other system-level or application-level protective measures must be implemented to eliminate the risk of unintended (or unplanned) device restart.

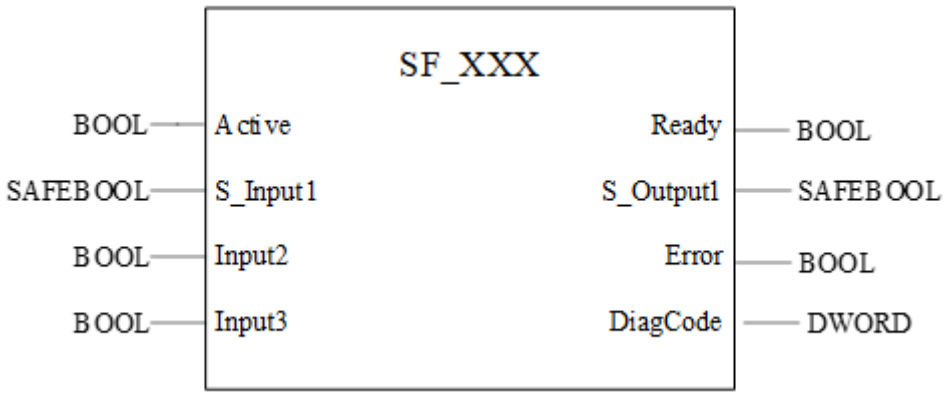
Input Parameter	Data Type	Description
Reset	SAFEBOOL	The Reset parameter is controlled by a variable, with an initial value of FALSE. Depending on specific functional requirements, this parameter input can be used for various purposes, such as resetting the state machine and associated fault/status information (indicated by DiagCode), provided the fault cause has been eliminated. This type of reset behavior is designed as a confirmation of fault troubleshooting. The interlock can be restarted by manual reset (refer to EN 954-1 standard), which is considered a functional reset. Additional reset functions can also be implemented for specific function blocks. Note that this function responds only to signal transitions from FALSE to TRUE. A sustained TRUE signal will not initiate any operation and may be considered a fault in certain function blocks.

11.10.2.2.2.2 General Output Variables of Safety Function Blocks

Output Parameter	Data Type	Description
Ready	BOOL	When the input is TRUE, it indicates that the function block is activated and the output result is valid (equivalent to the "POWER" indicator logic of a safety relay). Under this condition, the function block enters normal operational mode, with its internal logic executed in compliance with the predefined Safety Integrity Level (SIL). The output signals shall satisfy the fail-safe principles specified in IEC 61508/ISO 13849 via a dual-channel validation scheme. When the input is FALSE, the function block is not activated, program logic execution is suspended, and all safety outputs are forced into a predefined fail-safety status (such as degraded mode or shutdown lock), preventing uncontrolled operation.
SafetyDemand	BOOL	This parameter indicates that the function block is activated and the primary safety function has been requested (e.g., associated with safety function logic). This signal only reflects the basic safety requests and does not include the results of other safety-related input parameters (such as SafetyActive and EDM). When the output is TRUE, it indicates that the safety circuit fails to form a closed-loop logic, forcing the associated safety outputs into a predefined safety status (such as power-off lock or torque limitation mode), with no fault record generated at this time. When the output is FALSE, it indicates no safety request is present, and the safety outputs can return to normal operational status.

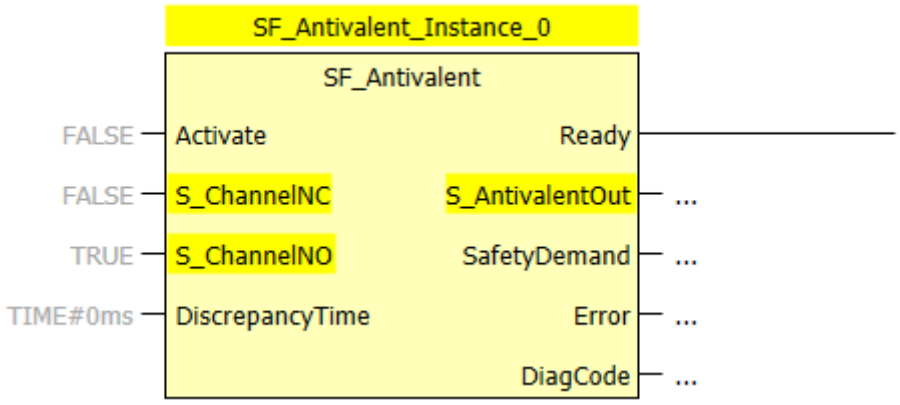
Output Parameter	Data Type	Description
ResetRequest	BOOL	This output prompts the operator to press the reset button to proceed with subsequent operations. When the output is TRUE, it indicates a reset request is present. When the output is FALSE, it indicates no reset request is present.
Error	BOOL	This output is a fault flag (equivalent to the "K1/K2" fault indicator on a safety relay) used to indicate the operational status of the function block. When the flag is TRUE, it indicates a fault has been detected and the function block has entered a faulty status. The corresponding fault code will be mapped in real time via the DiagCode output interface (e.g., standard fault codes E01~E99). When the flag is FALSE, it indicates no fault is present, and the function block is in normal operational or standby status, with DiagCode updated synchronously to the corresponding status code at this time.
DiagCode	WORD	DiagCode fully maps all statuses of the function block (activated, inactivated, faulty). At runtime, the system reflects only a single valid code at any given moment; if multiple faults occur, the DiagCode output will prioritize displaying the first detected fault (following the fault priority mechanism of ISO 13849-1).

11.10.2.2.2.3 General Representation of Safety Function Blocks



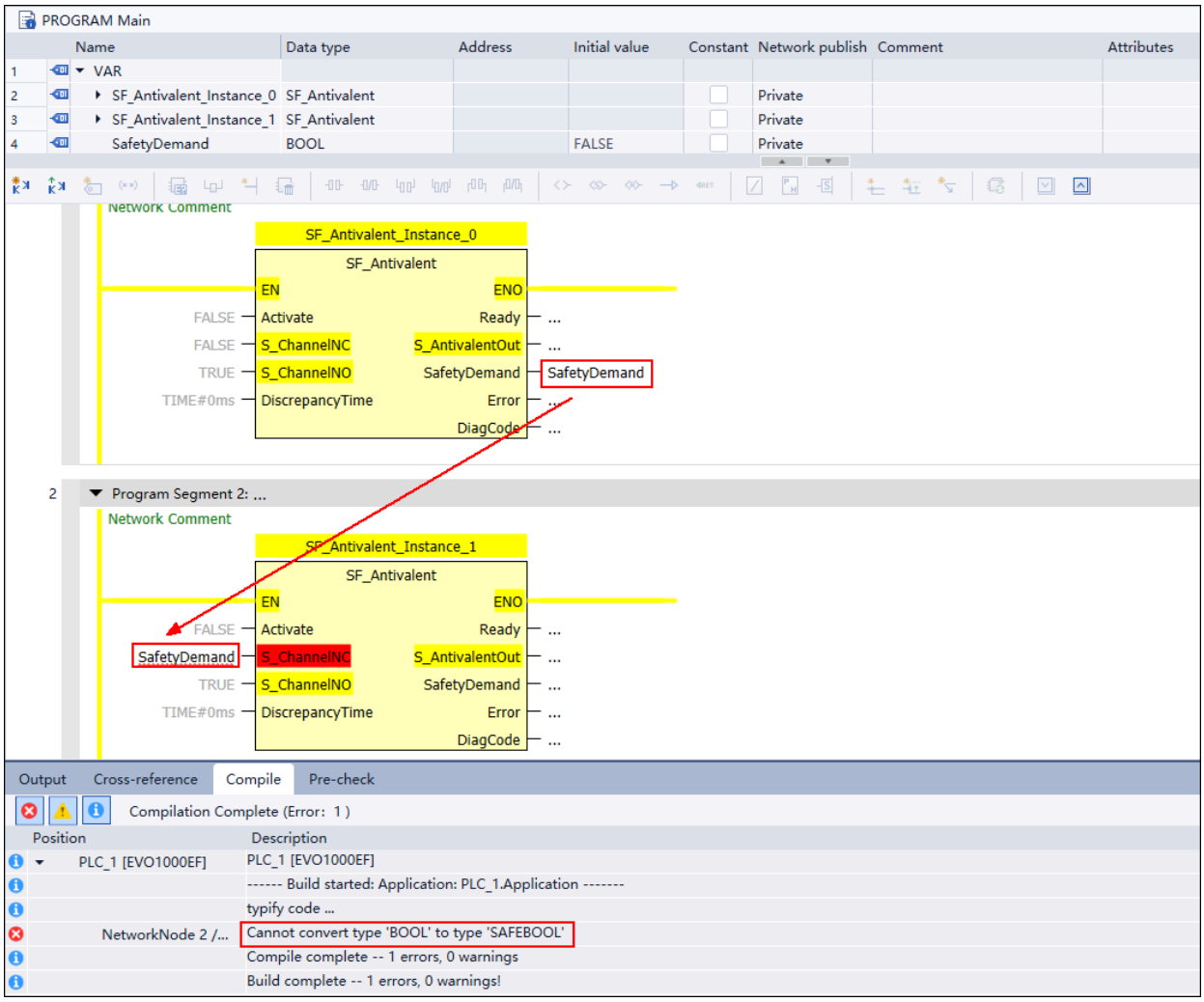
The SF_XXX function block consists of a function name, input parameters, and output parameters. Different function blocks have different names, as well as different input and output parameters. Based on engineering requirements, users shall select the appropriate safety function blocks compliant with relevant standards.

The input and output parameters of safety function blocks support the simultaneous use of safety-related types (such as SAFEBOOL) and standard types (such as BOOL), which are clearly distinguished by whether the parameter name includes the "S_" prefix. For example, the SF_Antivalent function block is displayed entirely in yellow in the programming interface; if its input and output parameters include the "S_" prefix, they are also highlighted in yellow, contrasting with standard type parameters with no prefix.



CAUTION

Although function blocks allow coexistence of safety-related parameters and standard parameters, a standard output of a function block cannot be directly connected to a safety input of another safety function block. If such an operation is attempted during programming, the compiler will report a fault, indicating that the operand 'SafetyDemand' expects a 'SAFEBOOL' input, and is unable to convert 'BOOL' to 'SAFEBOOL', as shown in the figure below.



11.10.2.2.2.4 Diagnostic Codes of Safety Function Blocks

A function block provides detailed diagnostic information regarding faults and statuses, including information about the conditions for status transition, and the output parameter values corresponding to each status.

Information Code	Description
0000(HEX)	The function block is not activated. This status code corresponds to the idle status. Taking a general-purpose function block as an example, its input/output (I/O) configuration can be set as: Activate = FALSE S_Input1 = FALSE or TRUE Input2 = FALSE Input3 = FALSE Ready = FALSE S_Output1 = FALSE Error = FALSE
8000(HEX)	When the function block is activated with no fault or condition triggering a safety output degradation, the system is in the default operational status. In this status, the safety output S_Output1 remains TRUE during normal operation. Taking a general-purpose function block as an example, its input/output (I/O) configuration can be set as: Activate = TRUE S_Input1 = TRUE Input2 = TRUE Input3 = TRUE Ready = TRUE S_Output1 = TRUE Error = FALSE

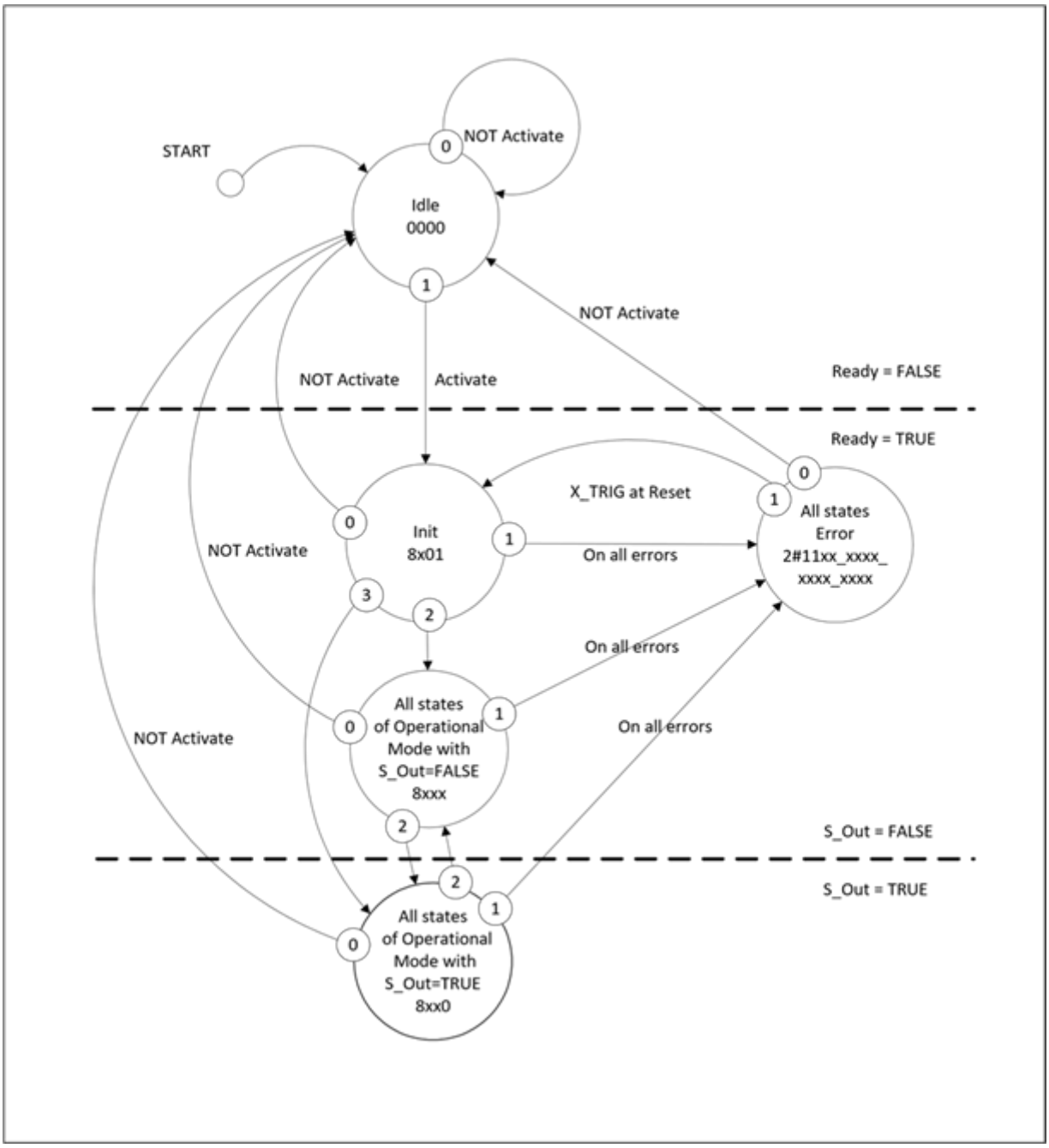
Information Code	Description
8401(HEX)	When the function block detects the activation signal and completes activation, but the safety output S_Out is still forced to FALSE, this status code indicates the system is in the initialization status of the operational mode. Taking a general-purpose safety function block as an example, its input/output (I/O) configuration can be set as: Activate = TRUE S_Input1 = TRUE Input2 = TRUE Input3 = TRUE Ready = TRUE S_Output1 = TRUE Error = FALSE

11.10.2.2.2.5 General Status Transition Diagram of Safety Function Blocks

The status transition diagram shows the conditions and corresponding diagnostic codes for the functional block as it transitions between different statuses.

Three areas in the diagram below: the top area represents the functional block in an inactivated status, the middle area represents the functional block in an activated status, and the bottom area represents the functional block in a normal status. The first horizontal line represents the transition from the inactivated status to the activated status, and the second horizontal line represents the transition from the non-safety status to the safety status. The priority of parallel transition conditions are indicated by numbers, with smaller numbers indicating higher priority. The status bubbles contain the status name and the hexadecimal

diagnostic code (DiagCode).



This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.

11.10.2.2.2.6 Precautions of Safety Function Blocks

For safety function blocks, multiple safety function blocks are typically combined according to operating conditions. The following function blocks must not be used alone; they must be used together with other function blocks to accomplish a specific function.

- SF_Override: Used to override muting functions. The override function for manual operation may be required to remove obstructions generated during the muting process from the safety area. If the override function is activated, stop requests issued by safety devices will not take effect. The SF_Override function block can only be used in conjunction with muting function blocks (SF_MutingXXX).

  In the application, the muting function block must be processed first, followed by the override function block.

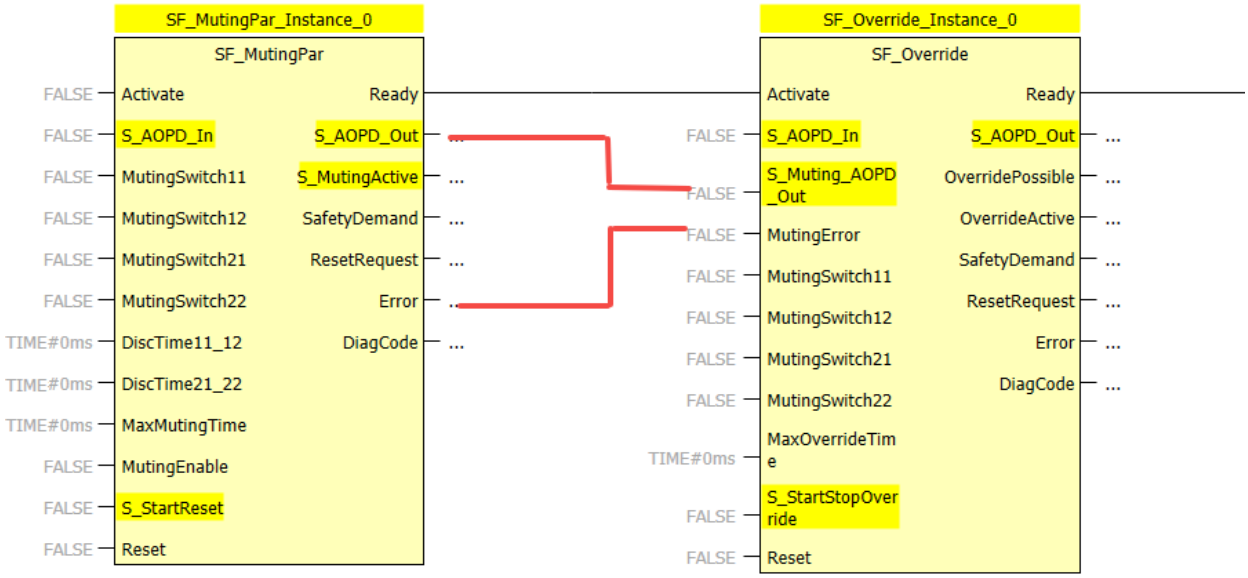
> **CAUTION**
>
> The output faults and DiagCode from the connected muting function block will not be transmitted to the output faults and DiagCode of the SF_Override function block.

It may be required to release the safety zone lockout generated during the muting process via a manually operated override muting function. When the bypass is activated, stop requests from safety devices will be invalid.

The SF_Override function block must be used in conjunction with a muting function block. For example, it can be used in combination with an SF_MutingPar function block, whose safety output parameters serve as the safety input for the SF_Override function block.
In the application, the muting function block must be executed first, followed by the override function block.

> **CAUTION**
>
> The Error and DiagCode outputs from the preceding muting function block will not be passed to the corresponding output terminals of the SF_Override function block.

The following figure shows the combination of the SF_Override function block with the SF_MutingXXX function block in the programming interface.

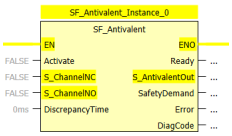
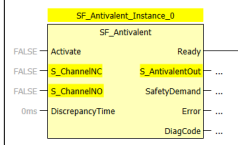


- SF_PressControl: This function block provides three modes for press control and is used with the function blocks "SF_ModeSelector" and "SF_CamMonitor". This function block implements the single stroke function in manual and setting modes. Only one of the three modes can be selected through the function block "SF_ModeSelector".

11.10.2.2.3 SF_Antivalent

SF_Antivalent monitors the discrepancy time between two inequivalent safety input values.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_Antivalent	Inequivalent input	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.
	S_ChannelNC	SAFEBOOL	TRUE, FALSE	FALSE	Indicates the input for a normally closed (NC) contact. FALSE: NC contact is open. TRUE: NC contact is closed.
	S_ChannelNO	SAFEBOOL	TRUE, FALSE	TRUE	Indicates the input for a normally open (NO) contact. FALSE: NO contact is open. TRUE: NO contact is closed.
	DiscrepancyTime	TIME	Follows data type	T#0ms	Used to set the maximum discrepancy time between the two input statuses.
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	S_AntivalentOut	SAFEBOOL	TRUE, FALSE	FALSE	Safety-related output FALSE: At least one input signal is inactivated or the status change occurs outside the monitoring time. TRUE: Both input signals are activated and the status changes occurs within the monitoring time.
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**
  - This function block continuously monitors the logical exclusivity of two SAFEBOOL input signals (S_ChannelNC and S_ChannelNO), and outputs the evaluation result as a SAFEBOOL signal. The two input channels are interdependent. The function block output depends on the logical comparison

result between the two channels: the output is valid when the two channels are in inequivalent statuses; a safety response is triggered when the two channels are in equivalent statuses

- When a signal transition in either channel causes the two channels to enter an equivalent status (i.e., S_ChannelNC == S_ChannelNO), the function block immediately sets the safety output S_AntivalentOut to FALSE and starts the discrepancy timer. If the inequivalent statuses of the two channels are not restored within the time window predefined by DiscrepancyTime (which must be greater than the safety task cycle), a fault occurs.
- Set DiscrepancyTime to a value greater than the safety task cycle.

- **Status transition diagram**

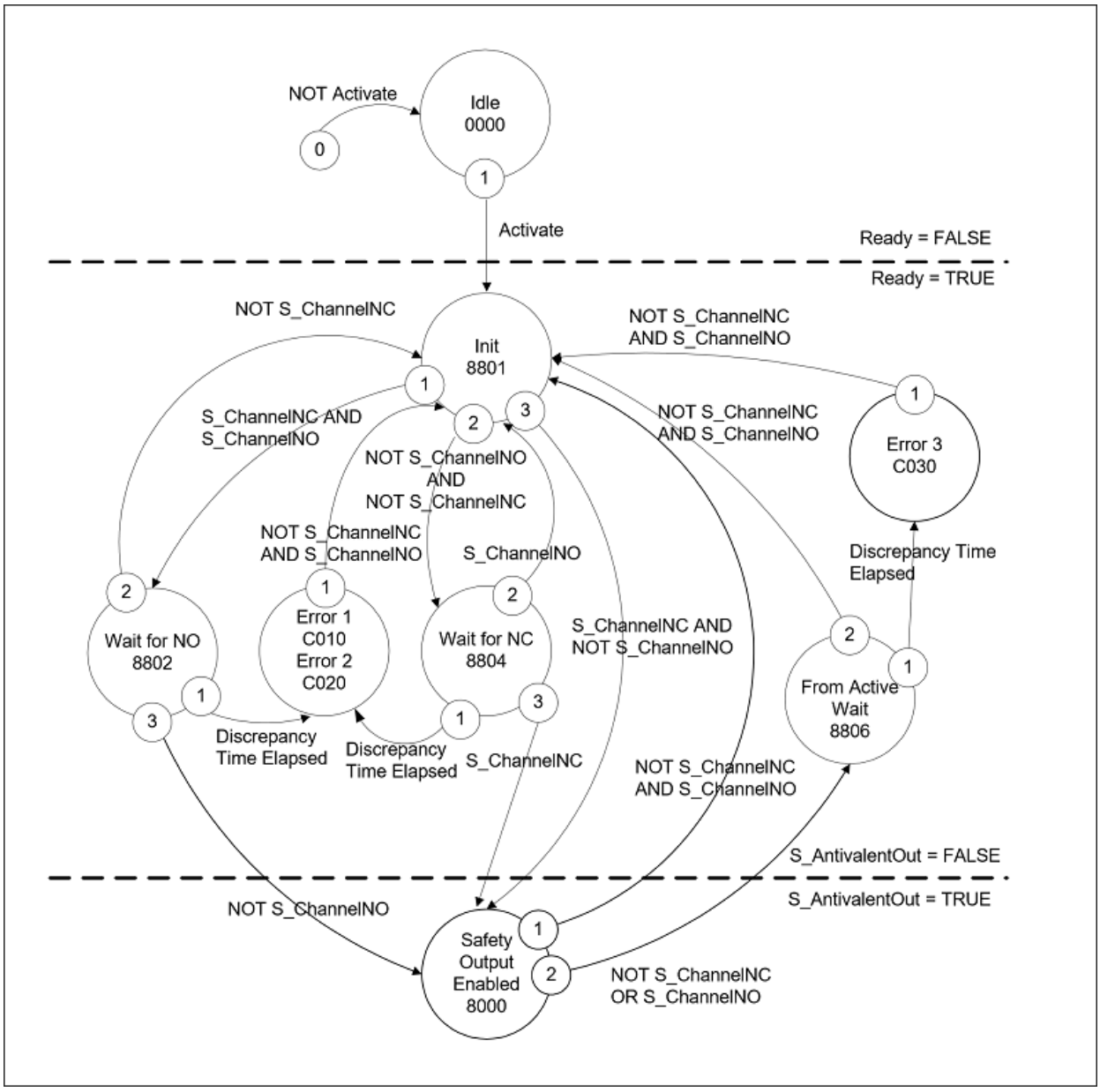
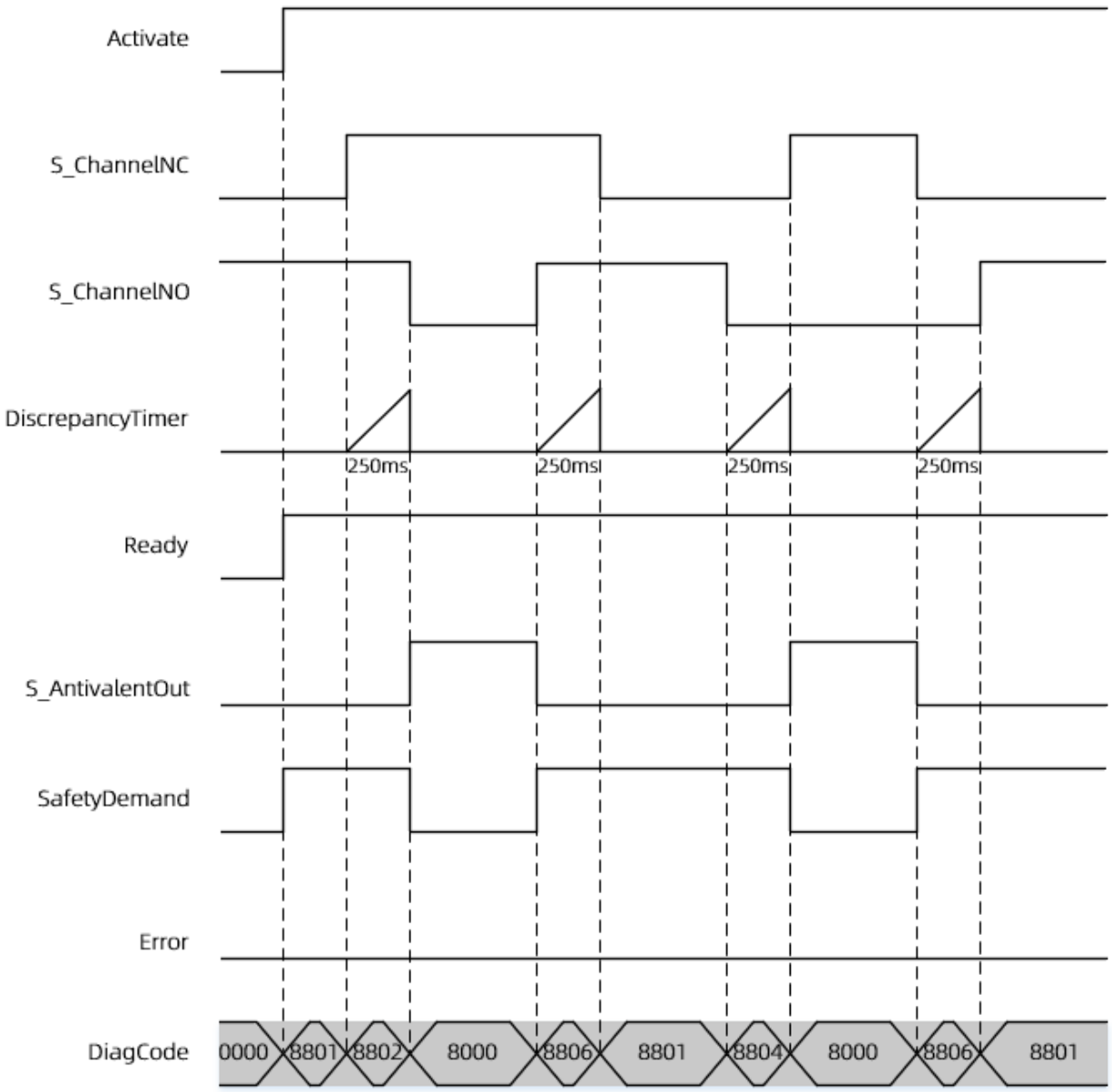
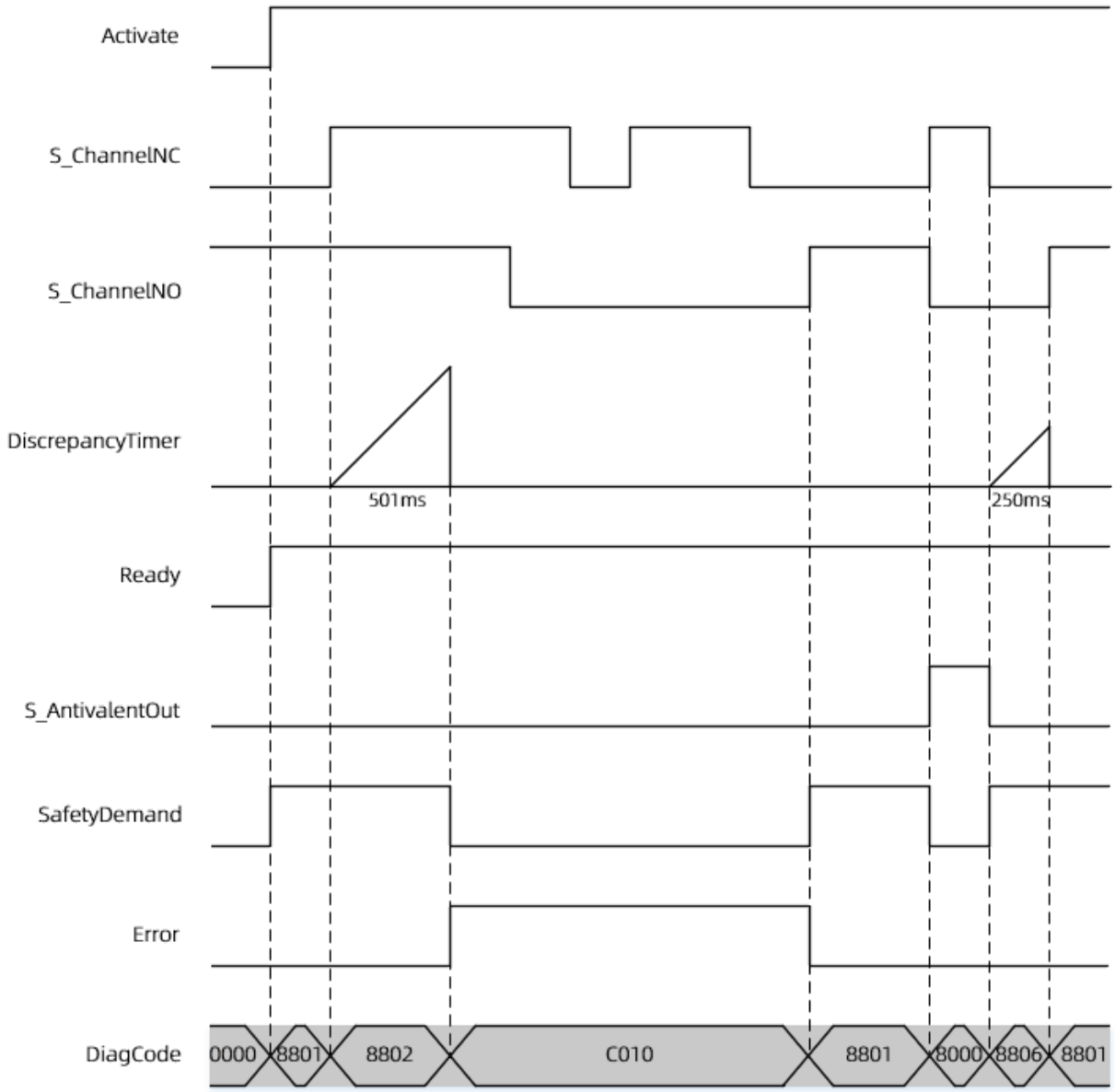


Figure 11-10-2-2-3-1 Status transition diagram for SF_Antivalent

- This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.
- The transition from any status to the idle status when Activate = FALSE is not indicated, but the priority (0) for transitioning to the idle status is the highest.

- **Sequence diagram**


Activate
S_ChannelNC
S_ChannelNO
DiscrepancyTimer
250ms
250ms
250ms
250ms
Ready
S_AntivalentOut
SafetyDemand
Error
DiagCode
0000
8801
8802
8000
8806
8801
8804
8000
8806
8801



- **Fault detection**
  The function block monitors the discrepancy time between the NO channel and the NC channel.
- **Fault behavior**
  When a fault occurs, reset operation must be performed based on the fault code indicated by DiagCode, combined with the status transition diagram.
- **Function block-specific fault codes and status codes**
  - Fault code

DiagCode	Status Name	Description and Output
C010	Error 1	Discrepancy time timeout in status 8802. Ready = TRUE S_AntivalentOut = FALSE SafetyDemand = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
C020	Error 2	Discrepancy time timeout in status 8804. Ready = TRUE S_AntivalentOut = FALSE SafetyDemand = FALSE Error = TRUE
C030	Error 3	Discrepancy time timeout in status 8806. Ready = TRUE S_AntivalentOut = FALSE SafetyDemand = FALSE Error = TRUE

◦ Status code

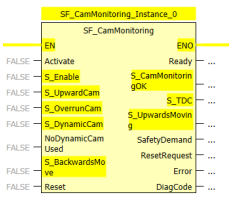
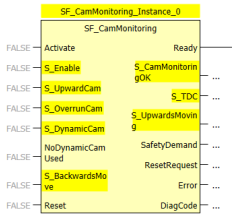
DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_AntivalentOut = FALSE SafetyDemand = FALSE Error = FALSE
8801	Init (Initialization)	Function block is activated. Ready = TRUE S_AntivalentOut = FALSE SafetyDemand = TRUE Error = FALSE
8000	Safety Output Enabled	Input switches to TRUE in equivalent mode. Ready = TRUE S_AntivalentOut = TRUE SafetyDemand = FALSE Error = FALSE
8802	Wait for NO	NC channel has been switched to TRUE and is waiting for NO channel to switch to FALSE; the discrepancy timer has been started. Ready = TRUE S_AntivalentOut = FALSE SafetyDemand = TRUE Error = FALSE

DiagCode	Status Name	Description and Output
8804	Wait for NC	NO channel has been switched to FALSE and is waiting for NC channel to switch to TRUE; the discrepancy timer has been started. Ready = TRUE S_AntivalentOut = FALSE SafetyDemand = TRUE Error = FALSE
8806	From Active Wait	One channel has been switched to FALSE and is waiting for the other channel to switch to FALSE. Ready = TRUE S_AntivalentOut = FALSE SafetyDemand = TRUE Error = FALSE

11.10.2.2.4 SF_CamMonitoring

SF_CamMonitoring provides cam monitoring function. The cam signal must have a defined sequence.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_CamMonitoring	Cam monitoring	FB		

- **Related variables**

Variables

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.
	S_Enable	SAFEBOOL	TRUE, FALSE	FALSE	Control signal used to activate cam monitoring. FALSE: Cam monitoring is not activated. TRUE: Cam monitoring is activated.

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
	S_UpwardCam	SAFEBOOL	TRUE, FALSE	FALSE	Input for upward cam. FALSE: Cam is activated. TRUE: Cam is not activated.
	S_OverrunCam	SAFEBOOL	TRUE, FALSE	FALSE	Input for overrun cam. FALSE: Cam is activated. TRUE: Cam is not activated.
	S_DynamicCam	SAFEBOOL	TRUE, FALSE	FALSE	Input for dynamic cam. FALSE: Cam is activated. TRUE: Cam is not activated.
	NoDynamicCamUsed	BOOL	TRUE, FALSE	FALSE	Input to disable the use of dynamic cam. FALSE: Enable dynamic cam. TRUE: Disable dynamic cam.
	S_BackwardsMove	SAFEBOOL	TRUE, FALSE	FALSE	Input to enable backward movement for the press. FALSE: Press moves forward. TRUE: Press moves backward.
	Reset	BOOL	TRUE, FALSE	FALSE	Manual reset
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	S_CamMonitoringOK	SAFEBOOL	TRUE, FALSE	FALSE	Output indicating the cam sequence. FALSE: Incorrect cam sequence or signal fault. TRUE: Correct cam sequence with no signal fault or deactivation.
	S_TDC	SAFEBOOL	TRUE, FALSE	FALSE	Output indicating whether the press is at top dead center (TDC). FALSE: The press is not at TDC. TRUE: The press is at TDC.

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
	S_UpwardsMoving	SAFEBOOL	TRUE, FALSE	FALSE	Output indicating whether the press is moving upward (from BDC to TDC). FALSE: The press is not moving upward. TRUE: The press is moving upward.
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	ResetRequest	BOOL	TRUE, FALSE	FALSE	Flag indicating whether a reset is required.
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**
  - This function block provides cam monitoring function. A defined cam signal sequence is required. The function block also provides information indicating whether the press is at top dead center (S_TDC) or whether it is moving upward (S_UpwardsMoving). Operation can be performed using two cams (upward and overrun) or three cams (upward, overrun, and dynamic).
  - The monitoring function of the function block can be enabled or disabled via the S_Enable input.

    Press movement sequence:

    1. Enable the monitoring function (activate function block).
    2. The press is in the TDC area (status code 8000)
       a. S_UpwardCam = FALSE
       b. S_OverrunCam = TRUE
       c. S_DynamicCam = FALSE
       d. S_TDC = TRUE
       e. S_UpwardsMoving = FALSE
    3. Overrun cam ends.
       a. The press is moving out of TDC area.
       b. F_TRIG at S_OverrunCam 8010
       c. S_UpwardCam = FALSE
       d. S_OverrunCam = FALSE
       e. S_DynamicCam = FALSE
       f. S_TDC = FALSE
       g. S_UpwardsMoving = FALSE

4. Upward cam starts.

   a. The press is in BDC area.

   b. R_TRIG at S_UpwardCam 8020

   c. S_UpwardCam = TRUE

   d. S_OverrunCam = FALSE

   e. S_DynamicCam = TRUE

   f. S_TDC = FALSE

   g. S_UpwardsMoving = TRUE

5. Overrun cam starts.

   a. R_TRIG at S_OverrunCam 8030

   b. S_UpwardCam = TRUE

   c. S_OverrunCam = TRUE

   d. S_DynamicCam = FALSE

   e. S_TDC = TRUE F_TRIG at S_DynamicCam

   f. S_UpwardsMoving = FALSE F_TRIG at S_DynamicCam

6. Upward cam ends.

   a. The press is in TDC area.

   b. F_TRIG at S_UpwardCam 8000

   c. S_UpwardCam = FALSE

   d. S_OverrunCam = TRUE

   e. S_DynamicCam = FALSE

   f. S_TDC = TRUE

   g. S_UpwardsMoving = FALSE

7. The press moves backwards.

   a. Signal to move backwards

   b. S_BackwardsMove = TRUE 8040

   c. S_Upwardcam = FALSE

   d. S_OverrunCam = FALSE

   e. S_DynamicCam = FALSE

   f. S_TDC = FALSE

   g. S_UpwardsMoving = FALSE

   h. R_TRIG at S_OverrunCam 8000

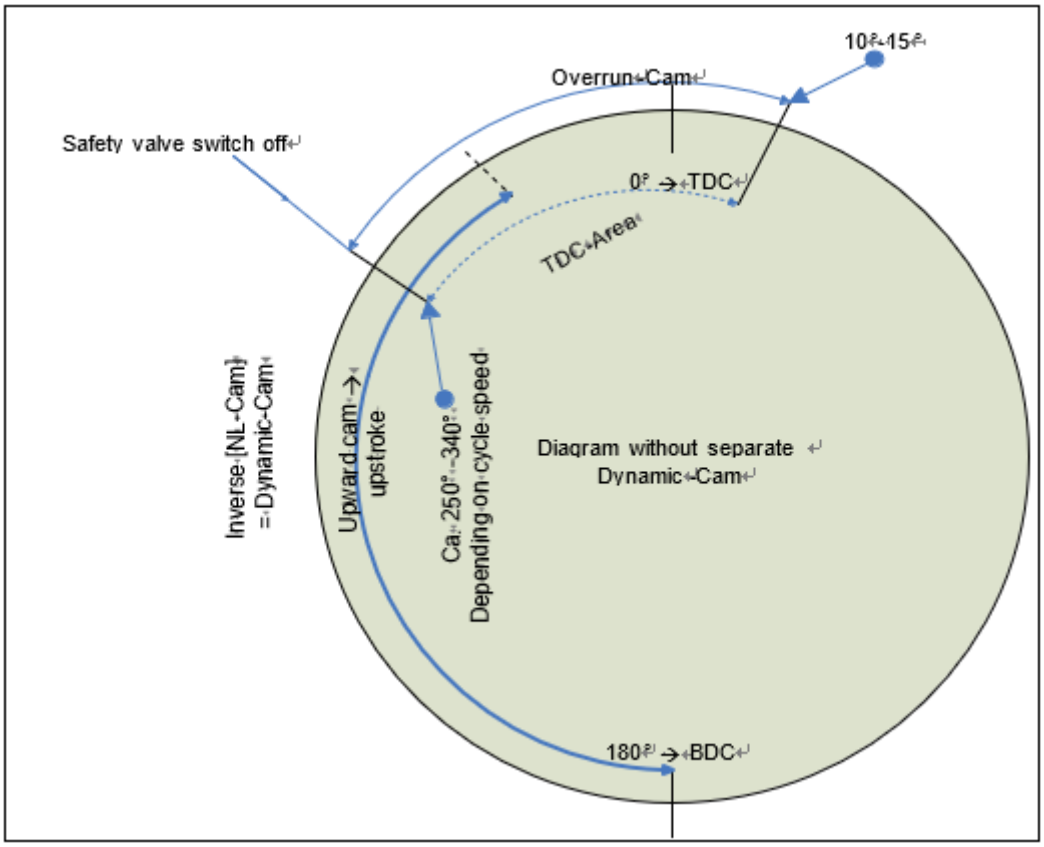


Figure 11-10-2-2-4-1 SF_CamMonitoring sequence diagram

- **Status transition diagram**

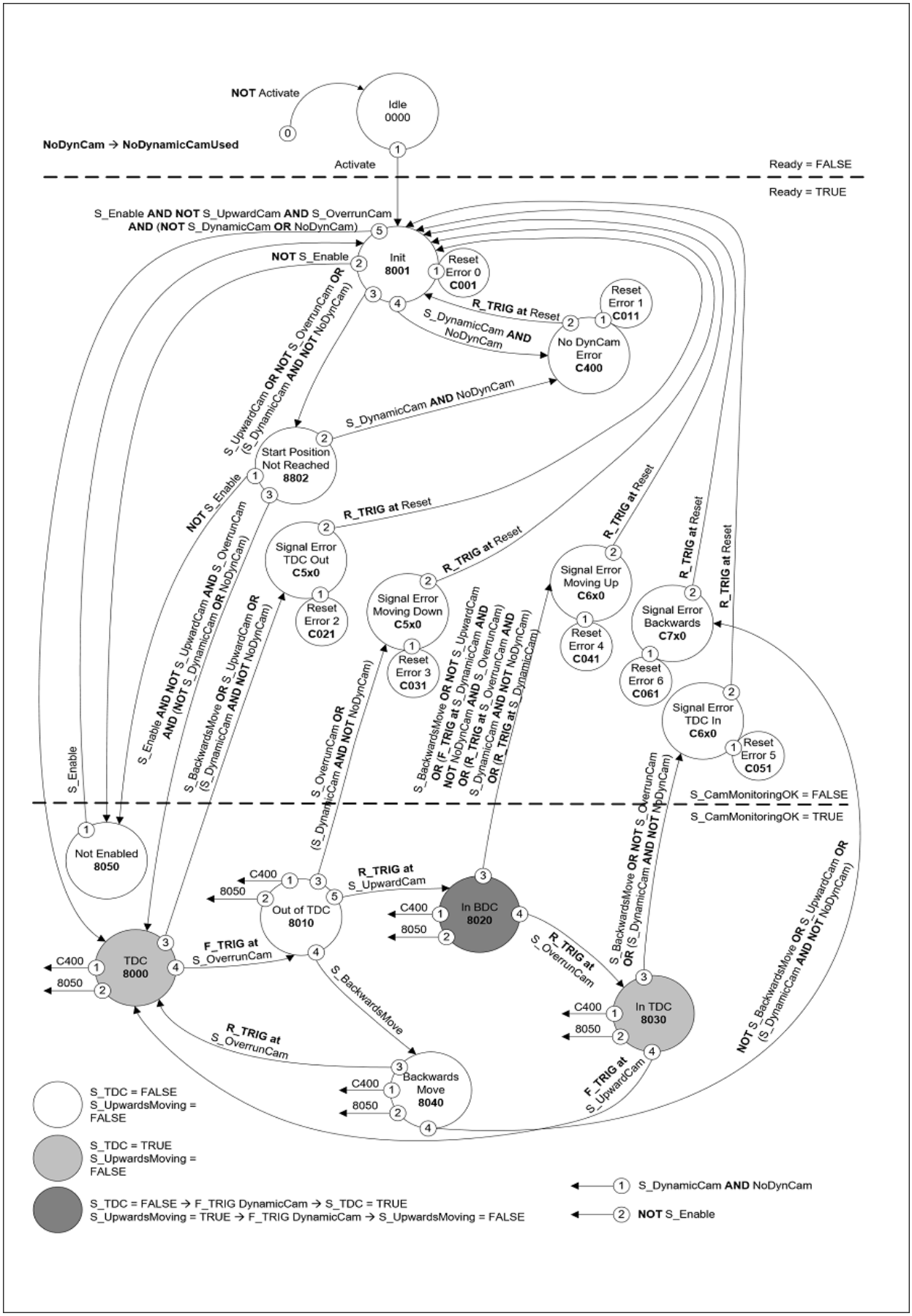
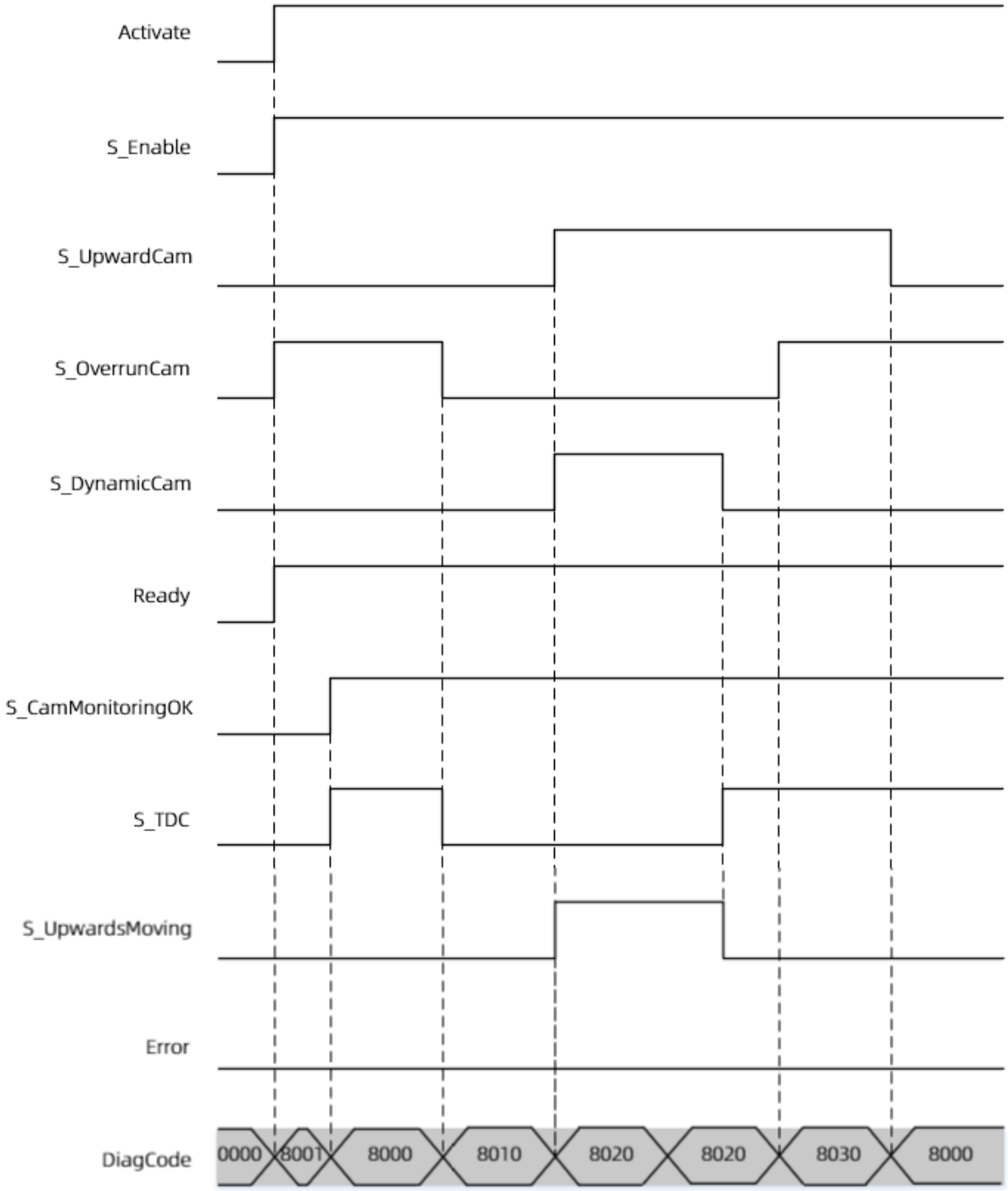


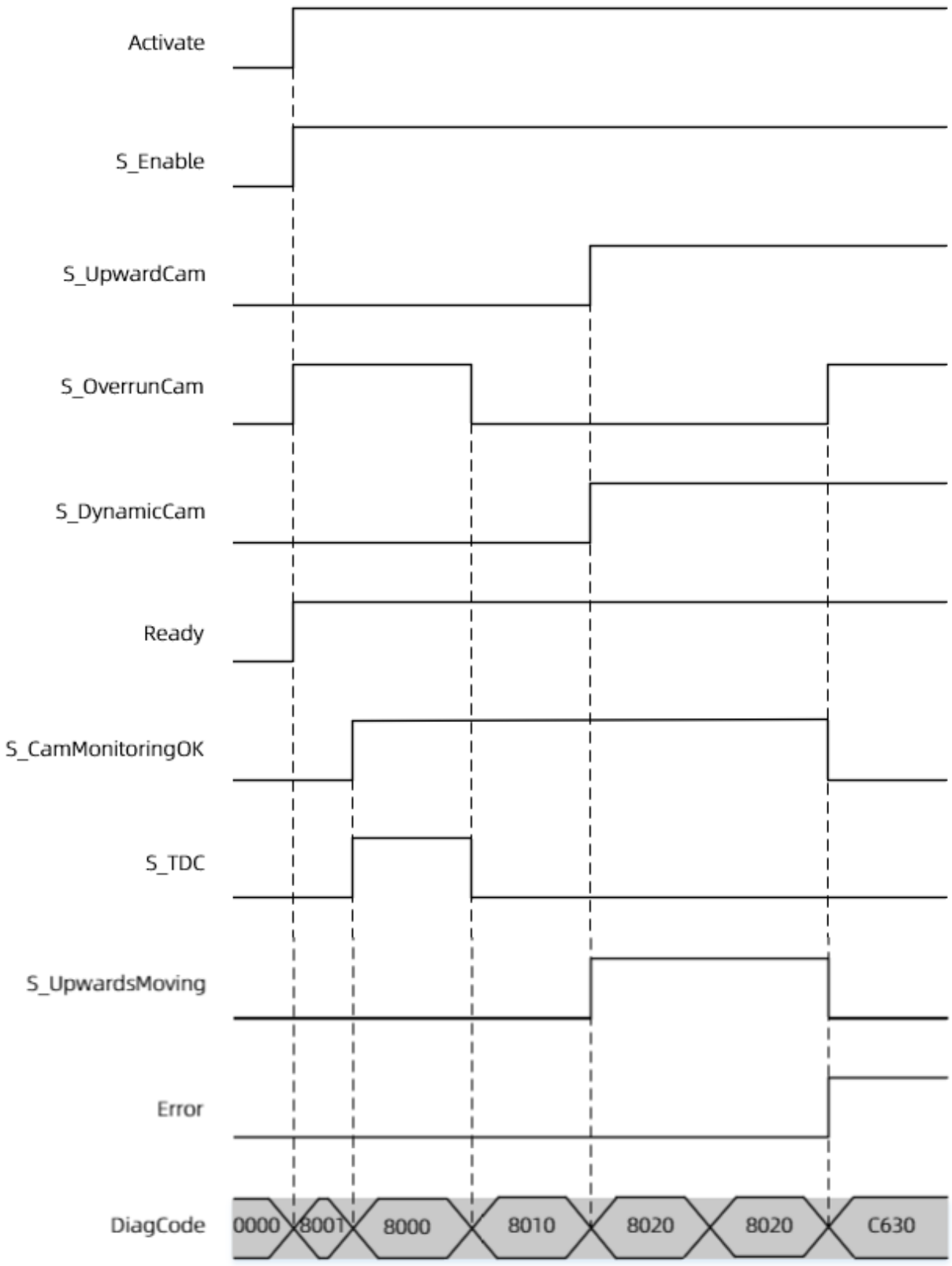
Figure 11-10-2-2-4-2 Status transition diagram for SF_CamMonitorin State Transition Diagram

◦ This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.

◦ The transition from any status to the idle status when Activate = FALSE is not indicated, but the priority (0) for transitioning to the idle status is the highest.

- **Sequence diagram**





- **Fault detection**

  The faulty status is triggered when a static TRUE signal of the Reset input is detected.

- **Fault behavior**

  When a fault occurs, the S_CamMonitoringOK output is set to FALSE. The DiagCode output indicates the relevant fault code, and the Error output is set to TRUE.

  Restart is prohibited until the fault condition is cleared and the operator confirms the safety status via Reset.

- **Function block-specific fault codes and status codes**
  - Fault code

DiagCode	Status Name	Description and Output
C400	No DynCam Error	Dynamic cam configuration is incorrect. Ready = TRUE S_CamMonitoringOK = FALSE S_TDC = FALSE S_UpwardsMoving = FALSE SafetyDemand = FALSE ResetRequest = TRUE Error = TRUE
C5x0	Signal Error TDC Out	An incorrect cam signal combination is detected during downward movement. If S_BackwardsMove is TRUE, the fault code is C500. If S_UpwardCam is TRUE, the fault code is C510. If S_DynamicCam is TRUE and NoDynCam is FALSE, the fault code is C520. Ready = TRUE S_CamMonitoringOK = FALSE S_TDC = FALSE S_UpwardsMoving = FALSE SafetyDemand = FALSE ResetRequest = TRUE Error = TRUE
C5x0	Signal Error Moving Down	An incorrect cam signal combination is detected during downward movement. If S_OverrunCam is TRUE, the fault code is C530. If S_DynamicCam is TRUE and NoDynCam is FALSE, the fault code is C540. Ready = TRUE S_CamMonitoringOK = FALSE S_TDC = FALSE S_UpwardsMoving = FALSE SafetyDemand = FALSE ResetRequest = TRUE Error = TRUE

DiagCode	Status Name	Description and Output
C6x0	Signal Error Moving Up	An incorrect cam signal combination is detected during upward movement. If S_BackwardsMove is TRUE, the fault code is C600. If S_UpwardCam is FALSE, the fault code is C610. If S_DynamicCam is triggered at the falling edge, NoDynCam is FALSE, and S_OverrunCam is TRUE, the fault code is C620. If S_OverrrunCam is triggered at the rising edge, S_DynamicCam is TRUE, and NoDynCam is FALSE, the fault code is C630. Ready = TRUE S_CamMonitoringOK = FALSE S_TDC = FALSE S_UpwardsMoving = FALSE SafetyDemand = FALSE ResetRequest = TRUE Error = TRUE
C6x0	Signal Error TDC In	An incorrect cam signal combination is detected when the press enters the TDC area If S_BackwardsMove is TRUE, the fault code is C640. If S_OverrunCam is FALSE, the fault code is C650. If S_DynamicCam is TRUE and NoDynCam is FALSE, the fault code is C660. Ready = TRUE S_CamMonitoringOK = FALSE S_TDC = FALSE S_UpwardsMoving = FALSE SafetyDemand = FALSE ResetRequest = TRUE Error = TRUE

DiagCode	Status Name	Description and Output
C7x0	Signal Error Backwards	An incorrect cam signal combination is detected during backward movement. If S_BackwardsMove is FALSE, the fault code is C700. If S_UpwardCam is TRUE, the fault code is C710. If S_DynamicCam is TRUE and NoDynCam is FALSE, the fault code is C720. Ready = TRUE S_CamMonitoringOK = FALSE S_TDC = FALSE S_UpwardsMoving = FALSE SafetyDemand = FALSE ResetRequest = TRUE Error = TRUE
C001	Reset Error 0	In the 8001 status, Reset is triggered at the non-rising edge. Ready = TRUE S_CamMonitoringOK = FALSE S_TDC = FALSE S_UpwardsMoving = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C011	Reset Error 1	In the C400 status, Reset is triggered at the non-rising edge. Ready = TRUE S_CamMonitoringOK = FALSE S_TDC = FALSE S_UpwardsMoving = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
C021	Reset Error 2	In the C5x0 status, Reset is triggered at the non-rising edge. Ready = TRUE S_CamMonitoringOK = FALSE S_TDC = FALSE S_UpwardsMoving = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C031	Reset Error 3	In the C5x0 status, Reset is triggered at the non-rising edge. Ready = TRUE S_CamMonitoringOK = FALSE S_TDC = FALSE S_UpwardsMoving = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C041	Reset Error 4	In the C6x0 status, Reset is triggered at the non-rising edge. Ready = TRUE S_CamMonitoringOK = FALSE S_TDC = FALSE S_UpwardsMoving = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
C051	Reset Error 5	In the C6x0 status, Reset is triggered at the non-rising edge. Ready = TRUE S_CamMonitoringOK = FALSE S_TDC = FALSE S_UpwardsMoving = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C061	Reset Error 6	In the C70 status, Reset is triggered at the non-rising edge. Ready = TRUE S_CamMonitoringOK = FALSE S_TDC = FALSE S_UpwardsMoving = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

◦ Status code

DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_CamMonitoringOK = FALSE S_TDC = FALSE S_UpwardsMoving = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8000	TDC	The press is at TDC. Ready = TRUE S_CamMonitoringOK = TRUE S_TDC = TRUE S_UpwardsMoving = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8001	Init (Initialization)	The function block is activated and initialized. Ready = TRUE S_CamMonitoringOK = FALSE S_TDC = FALSE S_UpwardsMoving = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8010	Out of TDC	The press is moving downward. Ready = TRUE S_CamMonitoringOK = TRUE S_TDC = FALSE S_UpwardsMoving = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

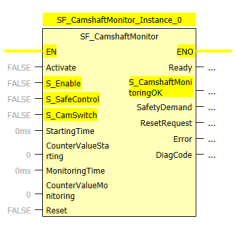
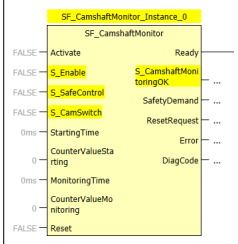
DiagCode	Status Name	Description and Output
8020	In BDC	The press is moving upward. Ready = TRUE S_CamMonitoringOK = TRUE S_TDC = FALSE S_UpwardsMoving = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE As long as the S_Dynamic input is not triggered at the falling edge, S_TDC remains FALSE and S_UpwardsMoving remains TRUE.
8030	In TDC	The press is in the TDC area. Ready = TRUE S_CamMonitoringOK = TRUE S_TDC = TRUE S_UpwardsMoving = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8040	Backwards Move	The press is moving backwards. Ready = TRUE S_CamMonitoringOK = TRUE S_TDC = FALSE S_UpwardsMoving = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8050	Not Enabled	The function block is not enabled. Ready = TRUE S_CamMonitoringOK = TRUE S_TDC = FALSE S_UpwardsMoving = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8802	Start Position Not Reached	Cam start signal is not ready. Ready = TRUE S_CamMonitoringOK = FALSE S_TDC = FALSE S_UpwardsMoving = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE

11.10.2.2.5 SF_CamshaftMonitor

SF_CamshaftMonitor provides cam shaft monitoring function. A certain amount of signal changes must occur within the specified period.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_CamshaftMonitor	Cam shaft monitoring function	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
	S_Enable	SAFEBOOL	TRUE, FALSE	FALSE	Control signal used to activate cam shaft monitoring. FALSE: Cam shaft monitoring is not activated. TRUE: Cam shaft monitoring is activated.
	S_SafeControl	SAFEBOOL	TRUE, FALSE	FALSE	Input used to indicate that the press is moving and monitoring needs to be performed. It is typically connected to a control signal or feedback signal of the press safety valve. FALSE: The press is not moving - monitoring is not activated. TRUE: The press is moving - monitoring is activated.
	S_CamSwitch	SAFEBOOL	TRUE, FALSE	FALSE	Input for the cam switch. FALSE: Cam is not activated. TRUE: Cam is activated.
	StartingTime	TIME	Follows data type	T#0ms	Ranges from 0 to X ms. This input indicates the device start time. The number of signal changes must reach CounterValueStarting within this time period.
	CounterValueStarting	INT	Follows data type	INT#0	Ranges from 0 to 40. Number of signal changes required to be reached during startup.
	MonitoringTime	TIME	Follows data type	T#0ms	Ranges from 0 to X ms. Monitoring time must be less than StartingTime. This input indicates the monitoring time during cycling operation. The number of signal changes must reach CounterValueMonitoring within this time period.
	CounterValueMonitoring	INT	Follows data type	INT#0	Ranges from 0 to 40. Number of signal changes required to be reached during cycling operation.
	Reset	BOOL	TRUE, FALSE	FALSE	Manual reset

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	S_Cam shaft MonitoringOK	SAFEBOOL	TRUE, FALSE	FALSE	Output indicating cam shaft monitoring status. FALSE: Incorrect cam shaft sequence or signal fault. TRUE: Cam shaft monitoring sequence is correct or not activated.
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	ResetRequest	BOOL	TRUE, FALSE	FALSE	Flag indicating whether a reset is required.
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.
If multiple cam switches are used, multiple instances of this function must be used. This function block can handle up to 20 strokes (40 signal changes). A signal change is the rising/falling edge of S_CamSwitch.					

- **Function description**
  SF_CamshaftMonitor provides cam shaft monitoring function. A series of signal changes of S_CamSwitch must be input within the specified time period.
  The monitoring function of the function block can be enabled or disabled via the S_Enable input.
  Monitoring sequence:
  1. Enable the monitoring function.
  2. The press is moving (S_SafeControl input is TRUE).
  3. Start the StartingTime timer and check the number of signal changes.
  4. Start the MonitoringTime timer and check the number of signal changes.
  5. Repeat step 4.
- **Note:**
  If a dynamic cam is not used, the input NoDynamicCamUsed must be set to TRUE.
- **Status transition diagram**

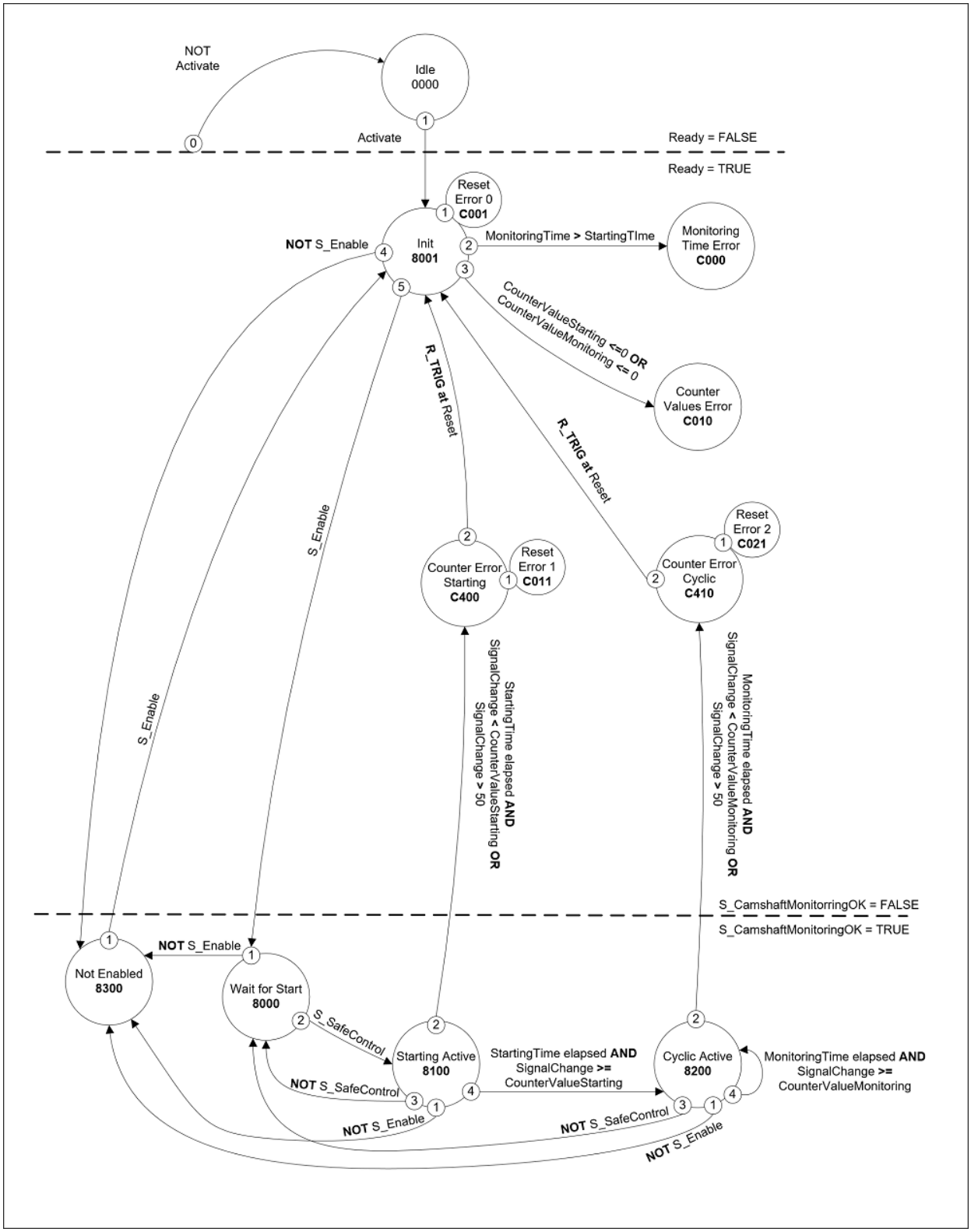


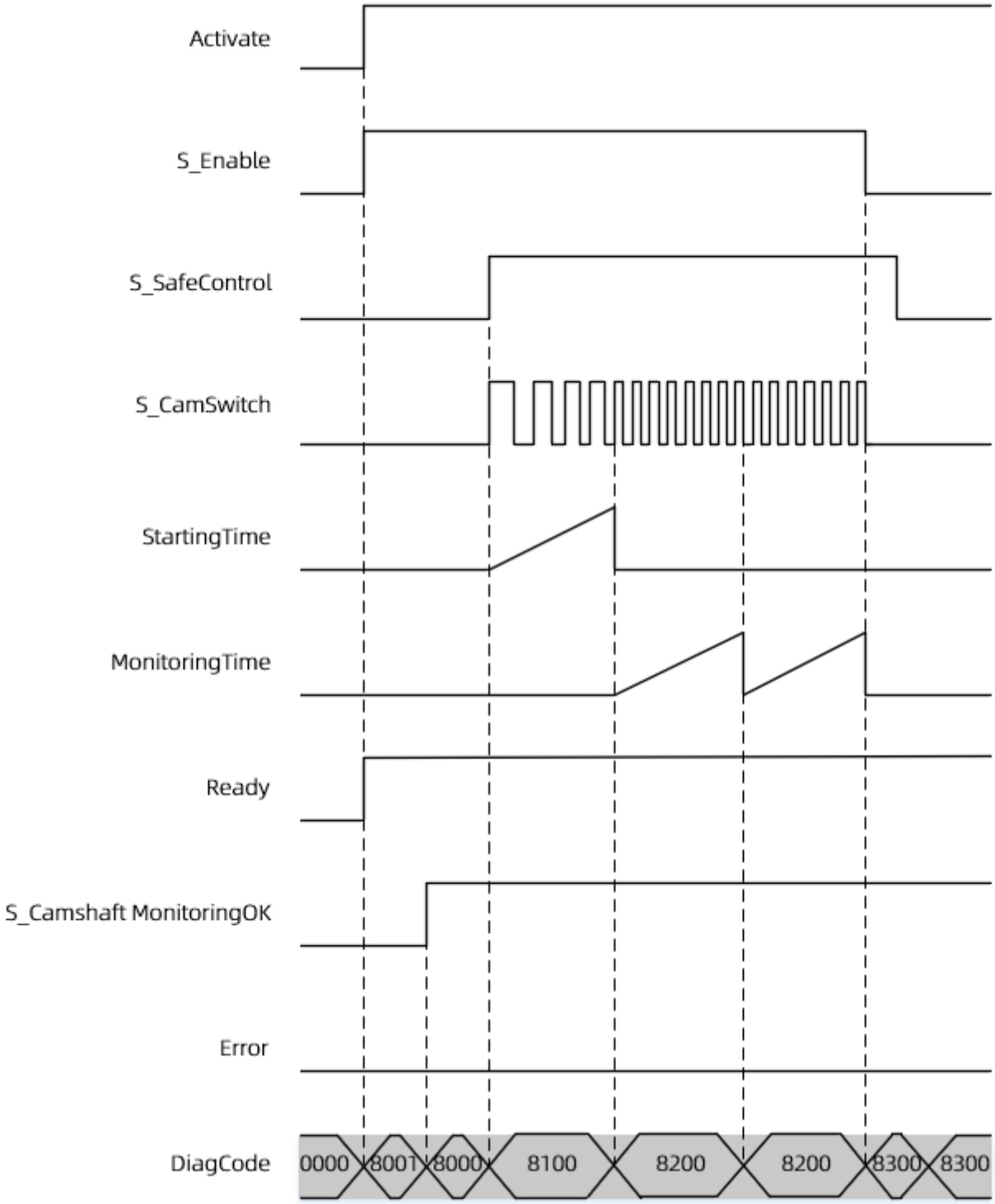
Figure 11-10-2-2-5-1 Status transition diagram for SF_CamshaftMonitor

- This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.
- The transition from any status to the idle status when Activate = FALSE is not indicated, but the priority (0) for transitioning to the idle status is the highest.

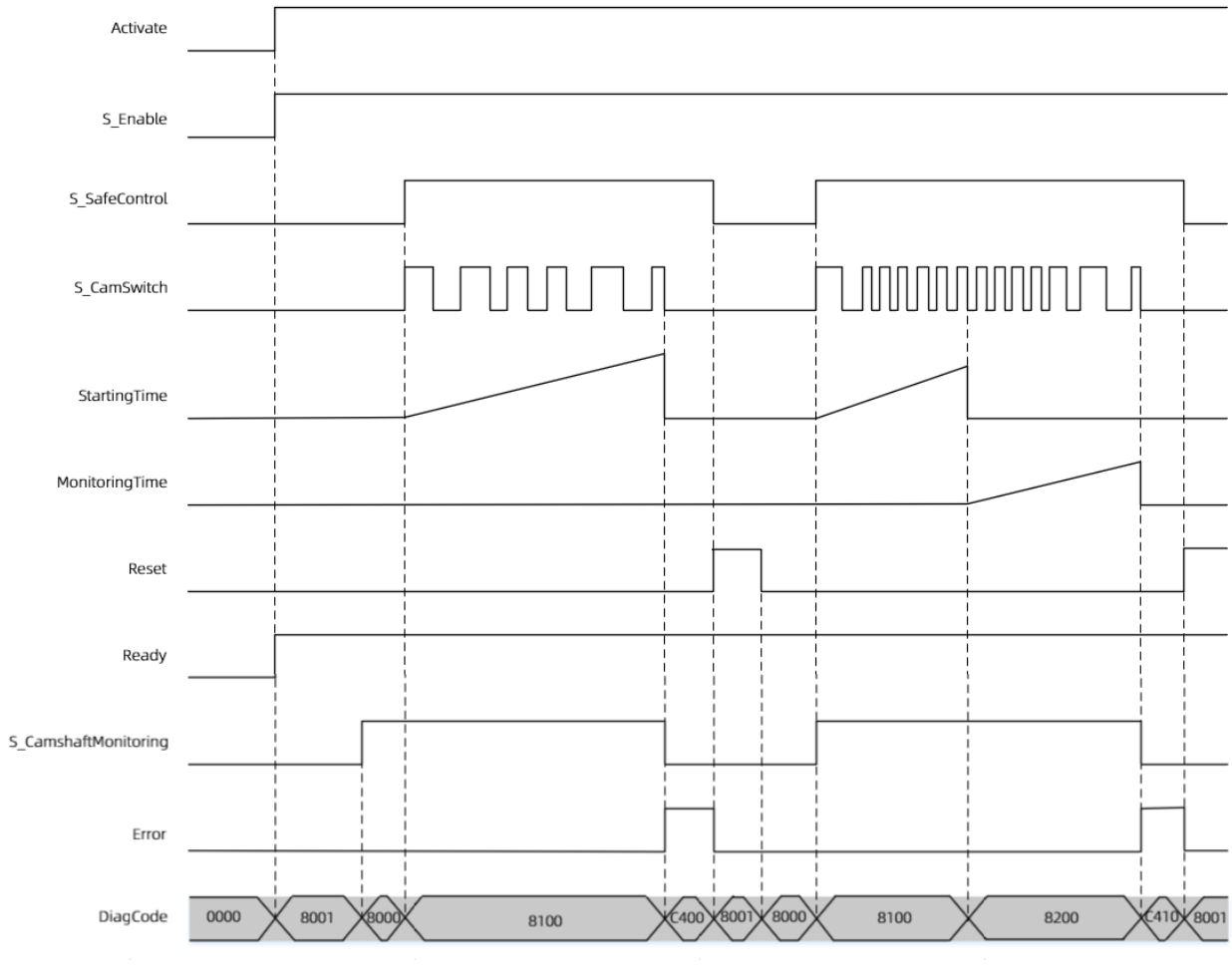
- **Sequence diagram**

  SafetyDemand and ResetRequest are not shown in the sequence diagram.

  CounterValueStarting = 9, CounterValueMonitoring = 15



CounterValueStarting = 14, CounterValueMonitoring = 28



- **Fault detection**
  The faulty status is triggered when a static TRUE signal of the Reset input is detected.
- **Fault behavior**
  When a fault occurs, the S_CamshaftMonitoringOK output is set to FALSE. The DiagCode output indicates the relevant fault code, and the Error output is set to TRUE.

  Restart is prohibited until the fault condition is cleared and the operator confirms the safety status via Reset.
- **Function block-specific fault codes and status codes**
  - Fault code

DiagCode	Status Name	Description and Output
C400	Counter Error Starting	The number of signal changes during startup is insufficient or exceeds 50. Ready = TRUE S_CamshaftMonitoring = FALSE SafetyDemand = FALSE ResetRequest = TRUE Error = TRUE

DiagCode	Status Name	Description and Output
C410	Counter Error Cyclic	The number of signal changes during cycling monitoring is insufficient or exceeds 50. Ready = TRUE S_CamshaftMonitoring = FALSE SafetyDemand = FALSE ResetRequest = TRUE Error = TRUE
C000	MonitoringTime Error	MonitoringTime is greater than StartingTime. Ready = TRUE S_CamshaftMonitoring = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C001	Reset Error 0	In the 8001 status, Reset is triggered at the non-rising edge. Ready = TRUE S_CamshaftMonitoring = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C010	Counter Values Error	The value of CounterValueStarting or CounterValueMonitoring is invalid. Ready = TRUE S_CamshaftMonitoring = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C011	Reset Error 1	In the C400 status, Reset is triggered at the non-rising edge. Ready = TRUE S_CamshaftMonitoring = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
C021	Reset Error 2	In the C410 status, Reset is triggered at the non-rising edge. Ready = TRUE S_CamshaftMonitoring = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

◦ Status code

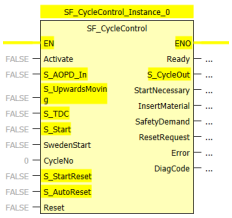
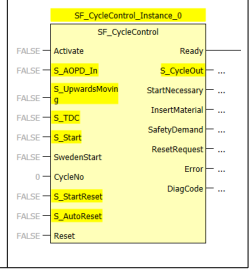
DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_AntivalentOut = FALSE SafetyDemand = FALSE Error = FALSE
8000	Wait for Start	Function block is activated. Ready = TRUE S_AntivalentOut = FALSE SafetyDemand = TRUE Error = FALSE
8001	Init (Initialization)	Input switches to TRUE in equivalent mode. Ready = TRUE S_AntivalentOut = TRUE SafetyDemand = FALSE Error = FALSE
8100	Starting Active	NC channel has been switched to TRUE and is waiting for NO channel to switch to FALSE; the discrepancy timer has been started. Ready = TRUE S_AntivalentOut = FALSE SafetyDemand = TRUE Error = FALSE

DiagCode	Status Name	Description and Output
8200	Wait for NC	NO channel has been switched to FALSE and is waiting for NC channel to switch to TRUE; the discrepancy timer has been started. Ready = TRUE S_AntivalentOut = FALSE SafetyDemand = TRUE Error = FALSE
8300	From Active Wait	One channel has been switched to FALSE and is waiting for the other channel to switch to FALSE. Ready = TRUE S_AntivalentOut = FALSE SafetyDemand = TRUE Error = FALSE

11.10.2.2.6 SF_CycleControl

SF_CycleControl is used to control the ESPE cycle mode (1 to n cycles).

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_CycleControl	Cycle control	FB		

- **Related variables**

  Variables

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.
	S_AOPD_In	SAFEBOOL	TRUE, FALSE	FALSE	OSSD signal from AOPD. FALSE: Protective field is interrupted. TRUE: Protective field is not interrupted.

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
	S_UpwardsMoving	SAFEBOOL	TRUE, FALSE	FALSE	Signal from the press, indicating the return stroke (e.g., from a cam controller). If the press return stroke is not hazardous and it is not desired for light curtain interruption to cause shutdown during the return stroke, this parameter must be connected to the relevant variable. FALSE: The press is not in return mode. TRUE: The press is in return mode. If the press return stroke is hazardous and/or no intervention is permitted during the return stroke, this parameter must be connected to a constant value of FALSE.
	S_TDC	SAFEBOOL	TRUE, FALSE	FALSE	Machine status signal (e.g., from a cam controller) indicating whether the machine is in the normal position. FALSE: The machine is not in the normal position. TRUE: The machine is in the normal position.
	S_Start	SAFEBOOL	TRUE, FALSE	FALSE	Edge-triggered signal used to start hazardous movement.
	SwedenStart	BOOL	TRUE, FALSE	FALSE	Specifies the start behavior (initialization of the function). FALSE: Standard start. TRUE: SwedenStart.
	CycleNo	INT	Follows data type	INT#0	Indicates the number of cycles (1 to n) that cause the safety output to be set to TRUE in one operating cycle. A single cycle is defined as a signal transition on S_AOPD_In from FALSE to TRUE.
	S_StartReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to start reset.
	S_AutoReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to automatically reset.
	Reset	BOOL	TRUE, FALSE	FALSE	Manual reset

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	S_CycleOut	SAFEBOOL	TRUE, FALSE	FALSE	Signal controlling the cycle movement.
	StartNecessary	BOOL	TRUE, FALSE	FALSE	Indicates whether it is necessary to start or maintain the cycle mode. For initialization or maintaining the cycle mode, S_AOPD_In must be set according to the value of CycleNo (TRUE -> FALSE -> TRUE).
	InsertMaterial	BOOL	TRUE, FALSE	FALSE	Indicates whether it is necessary to insert material to start or maintain the cycle mode. For initialization or maintaining the cycle mode, S_AOPD_In must be set according to the value of CycleNo (TRUE -> FALSE -> TRUE).
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	ResetRequest	BOOL	TRUE, FALSE	FALSE	Flag indicating whether a reset is required.
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**

  This function block is used for applications where access to hazardous areas during cyclic processes needs to be controlled by safety devices.

  - Normal operation: Manually start/initialize the first cycle/machine cycle. Two operating modes are available for starting this function (cycling function initialization): SwedenStart and normal start. After the first cycle/machine cycle (leaving TDC and reaching TDC again), subsequent machine cycles can be controlled by safety devices as long as MaxOperatorTime is not exceeded. If MaxOperatorTime is exceeded, the cyclic mode must be re-initialized. To start the cyclic mode, the machine must be in the normal position (TDC) in safety status.

    - Cyclic mode initialization (normal start):
      - The machine is at TDC (normal position).
      - The safety output is FALSE.
      - Interrupt ESPE (workpiece insertion) while the machine is in the normal position.
      - Interrupt ESPE (workpiece insertion) according to the CycleNo specification.
      - The time between two interventions must not exceed 30 seconds.
      - The safety output remains set to FALSE.

- The operator must switch S_Start from FALSE to TRUE within 30 seconds.

- Cyclic mode initialization (SwedenStart):
  - The machine is at TDC (normal position).
  - The safety output is FALSE.
  - The operator switches S_Start from FALSE to TRUE (starts 30-second timer).
  - The safety output remains set to FALSE.
  - Interrupt ESPE (workpiece insertion) according to the CycleNo specification. The time between two interventions must not exceed 30 seconds.
  - After reaching the number of cycles specified by CycleNo, the machine cycle is started.
- Cyclic mode operation/re-operation:
  - The machine is at TDC (normal position) or in return mode.
  - The time elapsed since the last initialization must not exceed 30 seconds.
    - TDC:
      - ESPE (workpiece insertion) must be interrupted.
      - Interrupt ESPE (workpiece insertion) according to the CycleNo specification.
      - The time between two interventions must not exceed 30 seconds.
    - Return stroke:
      - The number of interrupts on S_AOPD_In (from 1 to n) is counted by cycle counter.
      - Interrupt ESPE (workpiece insertion) according to the CycleNo specification.
      - The time between two interventions must not exceed 30 seconds.
  - After reaching the number of interrupts specified by CycleNo, the machine cycle is started.
  - Stop the time monitoring (T1) of material supply by MaxOperatorTime.
  - Set the safety output to TRUE. The machine moves away from TDC.
  - The machine reaches TDC (normal position) again. The safety output is set to FALSE again.

- Abnormal operation: If the safety output is TRUE and ESPE is interrupted, the safety output will be turned off and a fault message will be triggered under the following conditions:
  - The machine is moving downward.
  - Return stroke (S_UpwardsMoving) is dangerous for the operator.

  To exit the faulty status, a signal change from FALSE to TRUE on Reset may be required, and the cycle must be reinitialized.

- **Status transition diagram**

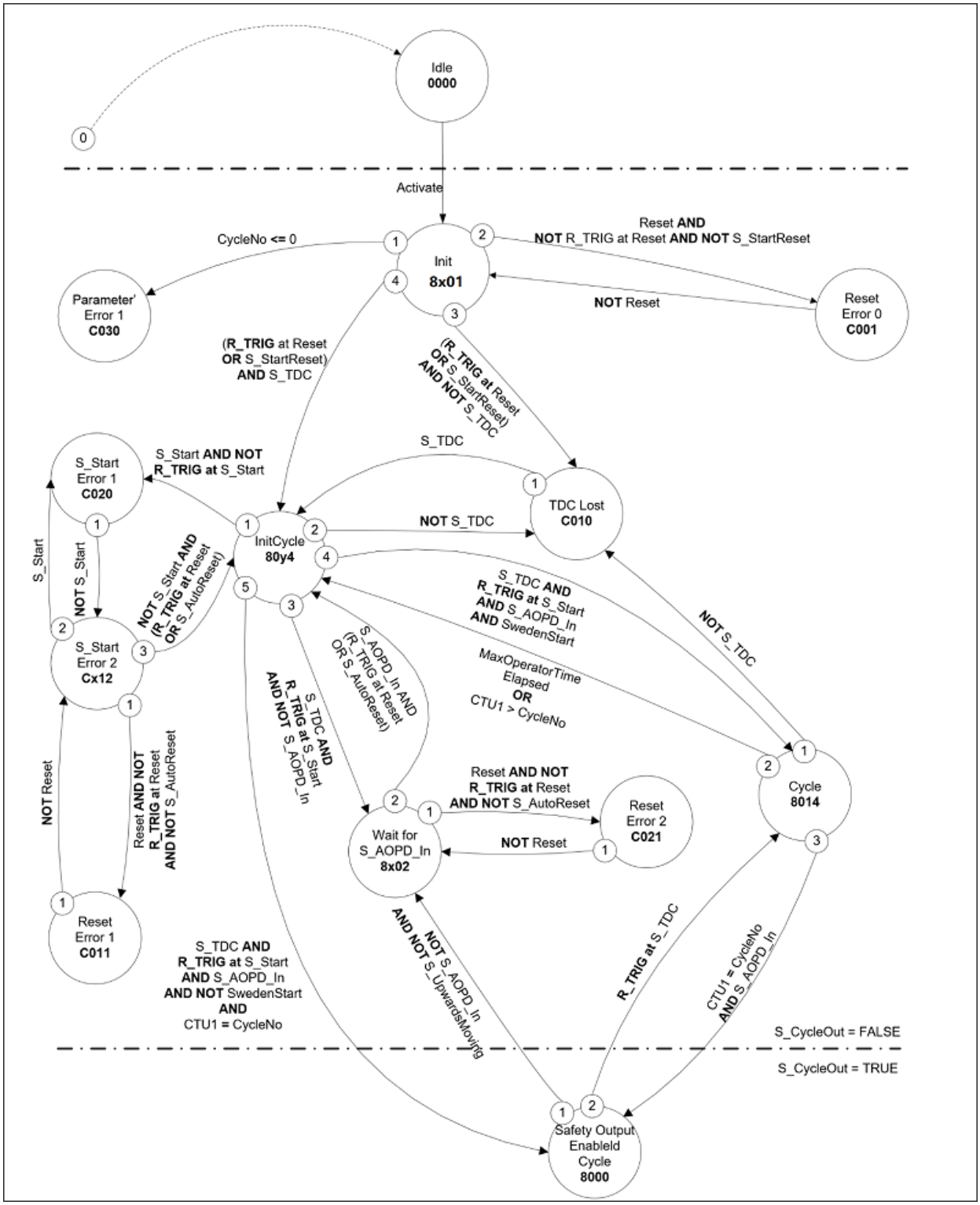
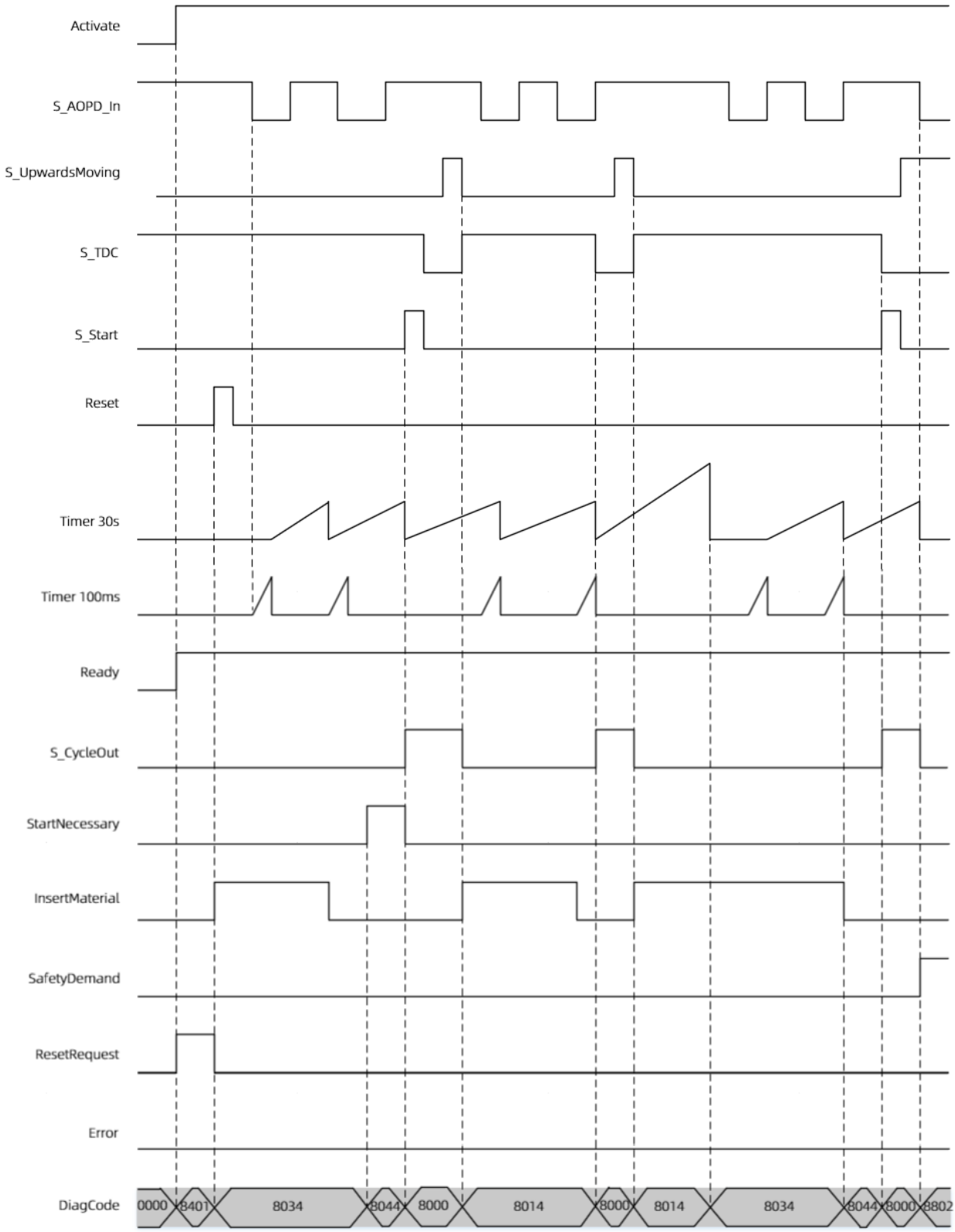


Figure 11-10-2-2-6-1 Status transition diagram for SF_CycleControl

- This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.
- The transition from any status to the idle status when Activate = FALSE is not indicated, but the priority (0) for transitioning to the idle status is the highest.

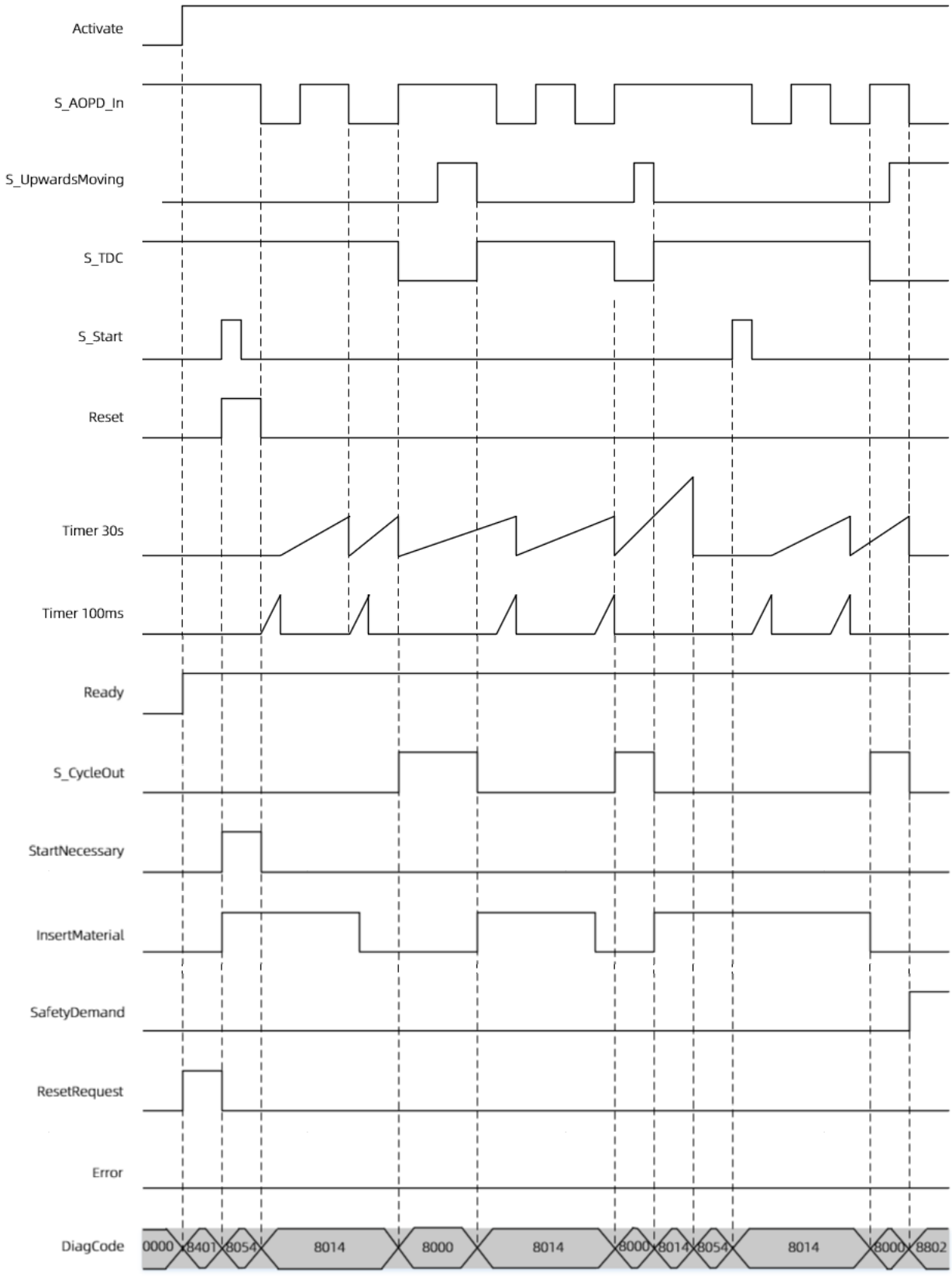
- **Sequence diagram**

  S_StartReset = FALSE, S_AutoReset = FALSE, SwedenStart = FALSE, CycleNo = INT#2



1. Initialize the function block.
2. Initialize the first cycle for standard start.
3. The machine operates normally.
4. The maximum operating time (30 seconds) has been reached; it is recommended to perform a new initialization cycle.
5. S_AOPD_In safety photoelectric device input signal is lost between stamping cycles.

S_StartReset = FALSE, S_AutoReset = FALSE, SwedenStart = TRUE, CycleNo = INT#2



1. Initialize the function block.
2. Initialize the first cycle for SwedenStart.
3. The machine operates normally.
4. The maximum operating time (30 seconds) has been reached; it is recommended to perform a new initialization cycle.

5. S_AOPD_In safety photoelectric device input signal is lost between stamping cycles.

- **Function block-specific fault codes and status codes**
  - Fault code

DiagCode	Status Name	Description and Output
C001	Reset Error 0	In the 8001 status, Reset is triggered at the non-rising edge. Ready = TRUE S_CycleOut = FALSE SafetyDemand = TRUE InsertMaterial = FALSE StartNecessary = FALSE ResetRequest = FALSE Error = TRUE
C011	Reset Error 1	In the C012 status, Reset is triggered at the non-rising edge. Ready = TRUE S_CycleOut = FALSE SafetyDemand = TRUE InsertMaterial = FALSE StartNecessary = FALSE ResetRequest = FALSE Error = TRUE
C021	Reset Error 2	In the 8002 status, Reset is triggered at the non-rising edge. Ready = TRUE S_CycleOut = FALSE SafetyDemand = TRUE InsertMaterial = FALSE StartNecessary = FALSE ResetRequest = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
C010	TDC Lost	The machine moves away from the normal position illegally. Ready = TRUE S_CycleOut = FALSE SafetyDemand = TRUE InsertMaterial = FALSE StartNecessary = FALSE ResetRequest = FALSE Error = TRUE
C020	S_Start Error 1	In the 8004 status, S_Start is triggered at the non-rising edge. Ready = TRUE S_CycleOut = FALSE SafetyDemand = TRUE InsertMaterial = FALSE StartNecessary = FALSE ResetRequest = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
Cx12	S_Start Error 2	If S_AutoReset is FALSE, the fault code is C412; otherwise, it is C012. • When the fault code is C012: Ready = TRUE S_CycleOut = FALSE SafetyDemand = FALSE InsertMaterial = FALSE StartNecessary = FALSE ResetRequest = FALSE Error = TRUE • When the fault code is C412: Ready = TRUE S_CycleOut = FALSE SafetyDemand = FALSE InsertMaterial = FALSE StartNecessary = FALSE ResetRequest = TRUE Error = TRUE
C030	Parameter Error	The value of CycleNo cannot be less than or equal to 0. Ready = TRUE S_CycleOut = FALSE SafetyDemand = TRUE InsertMaterial = FALSE StartNecessary = FALSE ResetRequest = FALSE Error = TRUE

◦ Status code

DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_CycleOut = FALSE SafetyDemand = FALSE InsertMaterial = FALSE StartNecessary = FALSE ResetRequest = FALSE Error = FALSE
8000	Safety Output Enable Cycle	Function block accepts cycle control. Ready = TRUE S_CycleOut = TRUE SafetyDemand = FALSE InsertMaterial = FALSE StartNecessary = FALSE ResetRequest = FALSE Error = FALSE • When S_UpwardMoving = FALSE: The cycle counter (CTU1) will be reset. S_AOPD_In = FALSE is considered an invalid interrupt. • When S_UpwardMoving = TRUE: Each signal transition of S_AOPD_In(FALSE) from FALSE to TRUE will be counted as a valid cycle (CTU1 count). The S_AOPD_In low-level signal for each interrupt must last for ≥ 100 ms (short-duration low-level signals that do not meet the requirement are not counted as interrupts). The maximum allowed interval between two interrupts/between cycle initialization and the next interrupt request is 30 seconds (T1). When the interval between two interrupts exceeds 30 seconds, the counter (CTU1) will be automatically reset to zero.

DiagCode	Status Name	Description and Output
8x01	Init (Initialization)	The function block is activated and initialized. A rising edge trigger is required on Reset. The counter (CTU1) and timer (T1) will be reset. If S_AutoReset = FALSE, the status code is 8401; otherwise, it is 8001. • When the status code is 8001: Ready = TRUE S_CycleOut = FALSE SafetyDemand = FALSE InsertMaterial = FALSE StartNecessary = FALSE ResetRequest = FALSE Error = FALSE • When the status code is 8401: Ready = TRUE S_CycleOut = FALSE SafetyDemand = FALSE InsertMaterial = FALSE StartNecessary = FALSE ResetRequest = TRUE Error = FALSE

DiagCode	Status Name	Description and Output
8x02	Wait for S_AOPD_In	The protective field is interrupted irregularly. The counter (CTU1) and timer (T1) will be reset. If S_AutoReset is FALSE and S_AOPD_In is TRUE, the status code is 8402; otherwise, it is 8802. • When the status code is 8802: Ready = TRUE S_CycleOut = FALSE SafetyDemand = TRUE InsertMaterial = FALSE StartNecessary = FALSE ResetRequest = FALSE Error = FALSE • When the status code is 8402: Ready = TRUE S_CycleOut = FALSE SafetyDemand = TRUE InsertMaterial = FALSE StartNecessary = FALSE ResetRequest = TRUE Error = FALSE

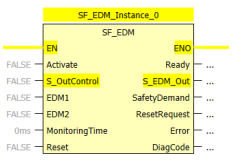
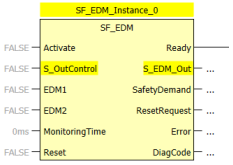
80y4	Init Cycle	Cycle control is enabled. When the first control cycle reaches status 8004, the cycle counter (CTU1) is automatically reset. • If SwedenStart is FALSE and CTU < CycleNo, the status code is 8034: Ready = TRUE S_CycleOut = FALSE SafetyDemand = FALSE InsertMaterial = TRUE StartNecessary = FALSE ResetRequest = FALSE Error = FALSE A signal transition of S_AOPD_In from FALSE to TRUE is counted as a valid cycle (CTU1 count). The low-level interrupt signal must last for ≥ 100 ms; short-duration low-level signals are not counted as valid interrupts. If the interval between two interrupts is ≤ 30s (T1), the counter will be reset to zero upon timeout. • If SwedenStart is FALSE and CTU = CycleNo, the status code is 8044: Ready = TRUE S_CycleOut = FALSE SafetyDemand = FALSE InsertMaterial = FALSE StartNecessary = TRUE ResetRequest = FALSE Error = FALSE The time interval between the last S_AOPD_In interrupt (used for standard initialization, CTU = CycleNo) and the signal transition of S_Start from FALSE to TRUE must not exceed 30 seconds. After timeout, the counter (CTU1) resets to 0 (diagnostic code 8034), and the full initialization procedure must be re-executed.

DiagCode	Status Name	Description and Output
		Each transition of S_AOPD_In from FALSE to TRUE (when S_AOPD_In = FALSE) is counted as a cycle (CTU1 count). The low-level interrupt signal must last for ≥ 100 ms; short-duration low-level signals are not counted as valid interrupts. If CTU > CycleNo, CTU is reset to zero and diagnostic code 8034 is triggered, requiring a complete re-initialization. • If SwedenStart is TRUE, the status code is 8054: Ready = TRUE S_CycleOut = FALSE SafetyDemand = FALSE InsertMaterial = FALSE StartNecessary = TRUE ResetRequest = FALSE Error = FALSE The time interval between the S_Start signal transition (FALSE→TRUE) and the first S_AOPD_In interrupt must be ≤ 30s.
8014	Cycle	Cycle control is activated. The device is in the start position. Material insertion must be performed. Ready = TRUE S_CycleOut = FALSE SafetyDemand = FALSE InsertMaterial = TRUE StartNecessary = FALSE ResetRequest = FALSE Error = FALSE A signal transition of S_AOPD_In from FALSE to TRUE is counted as a cycle (CTU1 count). The low-level interrupt signal must last for ≥ 100 ms; short-duration low-level signals are considered invalid. The interval between two interrupts must be ≤ 30 seconds (T1 timer monitoring); otherwise, the cycle control must be reinitialized.

11.10.2.2.7 SF_EDM

SF_EDM monitors inputs from external devices.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_EDM	External monitoring	FB		

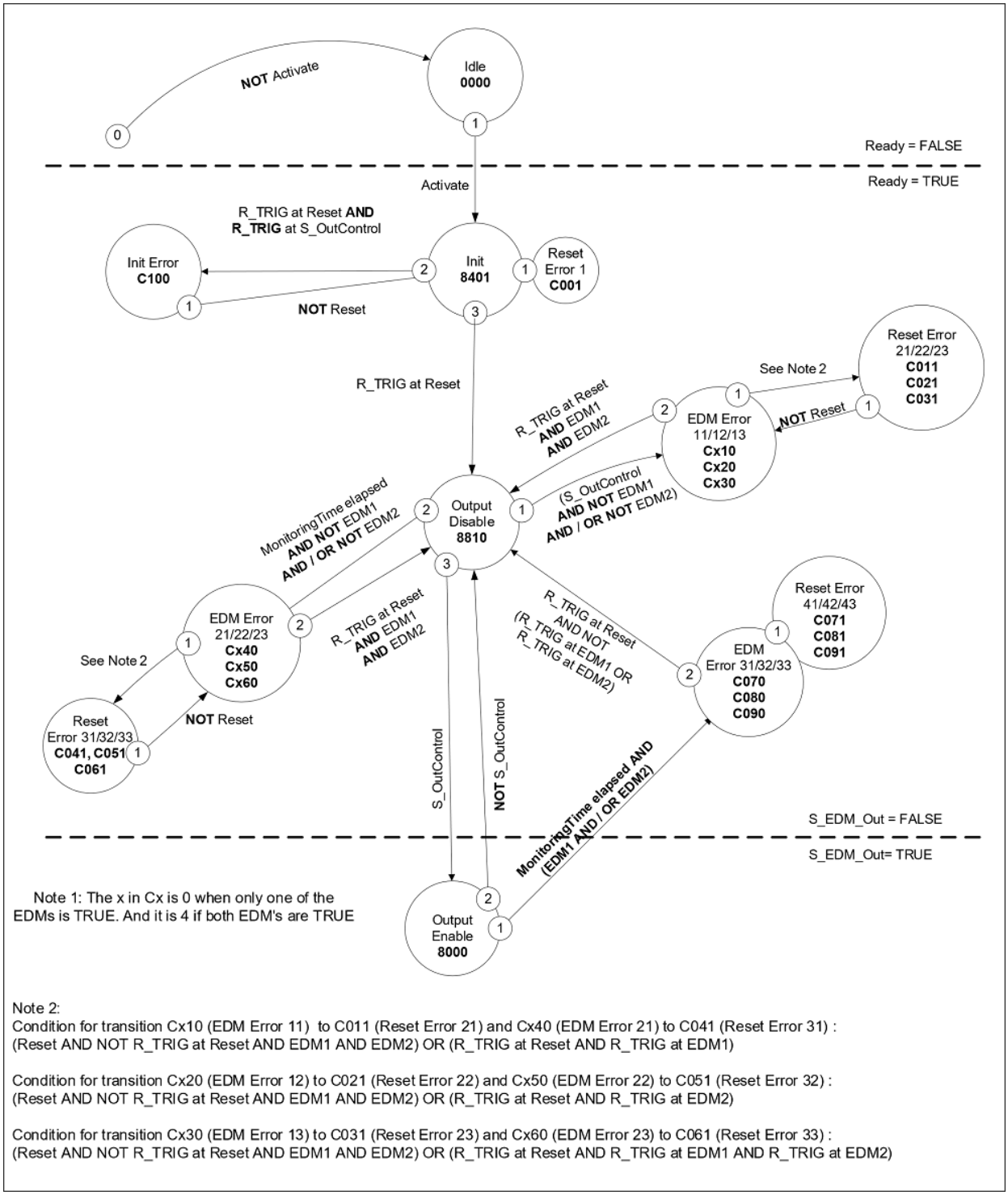
- **Related variables**

  Variables

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.
	S_OutControl	SAFEBOOL	TRUE, FALSE	FALSE	Control signal from other instructions. (such as SF_OutControl, SF_TwoHandControlTypeII, etc.). FALSE: Disable safety output (S_EDM_Out). TRUE: Enable safety output (S_EDM_Out).
	EDM1	BOOL	TRUE, FALSE	FALSE	Feedback signal from the first connected actuator. FALSE: The first connected actuator is in switch status. TRUE: The first connected actuator is in initial status.
	EDM2	BOOL	TRUE, FALSE	FALSE	Feedback signal from the second connected actuator. FALSE: The second connected actuator is in switch status. TRUE: The second connected actuator is in initial status.
	MonitoringTime	TIME	Follows data type	#0ms	Maximum response time for monitoring the actuator.
	Reset	BOOL	TRUE, FALSE	FALSE	Manual reset

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	S_EDM_Out	SAFEBOOL	TRUE, FALSE	FALSE	Safety output. FALSE: Disable the connected actuator. TRUE: Enable the connected actuator.
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	ResetRequest	BOOL	TRUE, FALSE	FALSE	Flag indicating whether a reset is required.
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**
  - The SF_EDM function block is used to control safety outputs and monitor the status of controlled actuators. This function block monitors the initial status of the actuator before it is enabled through feedback signals (EDM1 and EDM2), ensuring the actuator is in a predefined safe position (e.g., valve closed or motor braked). If the feedback signal does not match the expected status during this phase (e.g., EDM1 = TRUE and EDM2 = FALSE), a fault is triggered.
  - After the function block activates the actuator, it continuously monitors the actuator's switching status within the time window specified by the MonitoringTime parameter. If the expected transition (e.g., from FALSE->TRUE) of EDM1/EDM2 is not detected within this time window, the actuator is deemed to have a motion failure, and a fault is reported.
  - Feedback signal configuration rules: For precise diagnosis mode, two independent feedback signals (EDM1 and EDM2 connected to different detection points) must be configured for each actuator. For simplified diagnosis mode, two actuators can share one feedback signal, which must be connected to both EDM1 and EDM2 (i.e., EDM1=EDM2). In this mode, the system can only detect overall abnormalities of the actuator group but cannot identify the specific faulty unit.
  - Switching devices used for safety functions shall be selected from the categories specified in risk analysis (EN ISO 13849-1).
- **Status transition diagram**



Note 2:
Condition for transition Cx10 (EDM Error 11) to C011 (Reset Error 21) and Cx40 (EDM Error 21) to C041 (Reset Error 31) :
(Reset AND NOT R_TRIG at Reset AND EDM1 AND EDM2) OR (R_TRIG at Reset AND R_TRIG at EDM1)

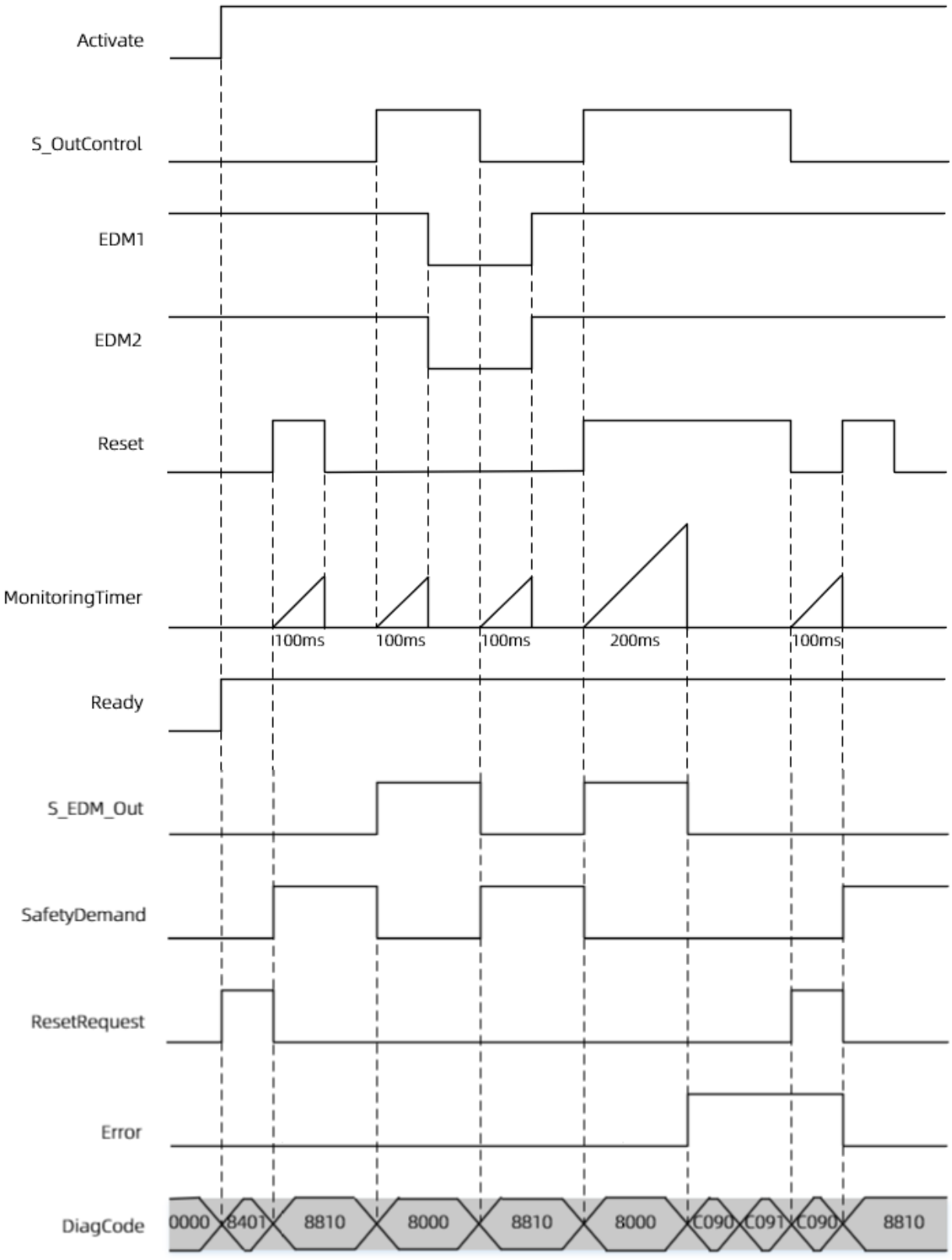
Condition for transition Cx20 (EDM Error 12) to C021 (Reset Error 22) and Cx50 (EDM Error 22) to C051 (Reset Error 32) :
(Reset AND NOT R_TRIG at Reset AND EDM1 AND EDM2) OR (R_TRIG at Reset AND R_TRIG at EDM2)

Condition for transition Cx30 (EDM Error 13) to C031 (Reset Error 23) and Cx60 (EDM Error 23) to C061 (Reset Error 33) :
(Reset AND NOT R_TRIG at Reset AND EDM1 AND EDM2) OR (R_TRIG at Reset AND R_TRIG at EDM1 AND R_TRIG at EDM2)

Figure 11-10-2-2-7-1 Status transition diagram for SF_EDM

- This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.
- The transition from any status to the idle status when Activate = FALSE is not indicated, but the priority (0) for transitioning to the idle status is the highest.
- The transition condition from status 8810 to status 8000 is S_OutControl being TRUE; this condition implicitly includes S_OutControl being TRUE, and both EDM1 and EDM2 being TRUE.

- **Sequence diagram**



- **Fault detection**
  - Detects that a TRUE value of Reset is triggered at the non-rising edge.
  - Detects invalid EDM signal during the process.
  - Due to a programming fault, S_OutControl and Reset are incorrectly connected to each other.
- **Fault behavior**
  When a fault occurs, reset operation must be performed based on the fault code indicated by DiagCode, combined with the status transition diagram.

- **Function block-specific fault codes and status codes**
  - Fault code

DiagCode	Status Name	Description and Output
C001	Reset Error 1	In the 8401 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_EDM_Out = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C011	Reset Error 21	In the C010 status, a TRUE value of Reset is triggered at the non-rising edge or both EDM1 and Reset are triggered at the rising edge simultaneously. Ready = TRUE S_EDM_Out = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C021	Reset Error 22	In the C020 status, a TRUE value of Reset is triggered at the non-rising edge or both EDM2 and Reset are triggered at the rising edge simultaneously. Ready = TRUE S_EDM_Out = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C031	Reset Error 23	In the C030 status, a TRUE value of Reset is triggered at the non-rising edge, or EDM1, EDM2, and Reset are all triggered at the rising edge simultaneously. Ready = TRUE S_EDM_Out = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
C041	Reset Error 31	In the C040 status, a TRUE value of Reset is triggered at the non-rising edge or both EDM1 and Reset are triggered at the rising edge simultaneously. Ready = TRUE S_EDM_Out = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C051	Reset Error 32	In the C050 status, a TRUE value of Reset is triggered at the non-rising edge or both EDM2 and Reset are triggered at the rising edge simultaneously. Ready = TRUE S_EDM_Out = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C061	Reset Error 33	In the C060 status, a TRUE value of Reset is triggered at the non-rising edge, or EDM1, EDM2, and Reset are all triggered at the rising edge simultaneously. Ready = TRUE S_EDM_Out = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C071	Reset Error 41	In the C070 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_EDM_Out = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
C081	Reset Error 42	In the C080 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_EDM_Out = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C091	Reset Error 43	In the C090 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_EDM_Out = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C010	EDM Error 11	EDM1 signal is invalid in the initial actuator status. In the 8810 status, when S_OutControl is enabled, EDM1 signal is FALSE and EDM2 signal is TRUE. Ready = TRUE S_EDM_Out = FALSE SafetyDemand = FALSE ResetRequest = R1 Error = TRUE
C020	EDM Error 12	EDM2 signal is invalid in the initial actuator status. In the 8810 status, when S_OutControl is enabled, EDM1 signal is TRUE and EDM2 signal is FALSE. Ready = TRUE S_EDM_Out = FALSE SafetyDemand = FALSE ResetRequest = R1 Error = TRUE

DiagCode	Status Name	Description and Output
C030	EDM Error 13	EDM1 and EDM2 signals are invalid in the initial actuator status. In the 8810 status, when S_OutControl is enabled, EDM1 and EDM2 signals are both FALSE. Ready = TRUE S_EDM_Out = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C040	EDM Error 21	EDM1 signal is invalid in the initial actuator status. In the 8810 status, the EDM1 signal is FALSE and exceeds the monitoring time. Ready = TRUE S_EDM_Out = FALSE SafetyDemand = FALSE ResetRequest = R1 Error = TRUE
C050	EDM Error 22	EDM2 signal is invalid in the initial actuator status. In the 8810 status, the EDM2 signal is FALSE and exceeds the monitoring time. Ready = TRUE S_EDM_Out = FALSE SafetyDemand = FALSE ResetRequest = R1 Error = TRUE
C060	EDM Error 23	EDM1 and EDM2 signals are invalid in the initial actuator status. In the 8810 status, the EDM1 and EDM2 signals are both FALSE and exceed the monitoring time. Ready = TRUE S_EDM_Out = FALSE SafetyDemand = FALSE ResetRequest = R1 Error = TRUE

DiagCode	Status Name	Description and Output
C070	EDM Error 31	EDM1 signal is invalid in the actuator switching status. In the 8000 status, the EDM1 signal is TRUE, the EDM2 signal is FALSE and exceeds the monitoring time. Ready = TRUE S_EDM_Out = FALSE SafetyDemand = FALSE ResetRequest = NOT RESET Error = TRUE
C080	EDM Error 32	EDM2 signal is invalid in the actuator switching status. In the 8000 status, the EDM2 signal is TRUE, the EDM1 signal is FALSE and exceeds the monitoring time. Ready = TRUE S_EDM_Out = FALSE SafetyDemand = FALSE ResetRequest = NOT RESET Error = TRUE
C090	EDM Error 33	EDM1 and EDM2 signals are invalid in the actuator switching status. In the 8000 status, the EDM1 and EDM2 signals are both TRUE and exceed the monitoring time. Ready = TRUE S_EDM_Out = FALSE SafetyDemand = FALSE ResetRequest = NOT RESET Error = TRUE
C100	Init Error	S_OutControl and Reset are both triggered at the rising edge at the same time. Ready = TRUE S_EDM_Out = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

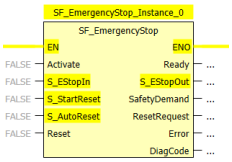
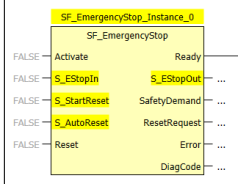
◦ Status code

DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_EDM_Out = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8401	Init (Initialization)	When the function block is activated, start inhibit is in an activated status. Reset is required. Ready = TRUE S_EDM_Out = FALSE SafetyDemand = FALSE ResetRequest = TRUE Error = FALSE
8810	Output Disable	EDM control is not activated. Start timer upon entering the status. Ready = TRUE S_EDM_Out = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8000	Output Enable	EDM control is activated. Start timer upon entering the status. Ready = TRUE S_EDM_Out = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

11.10.2.2.8 SF_EmergencyStop

SF_EmergencyStop monitors the input of the emergency stop button.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_EmergencyStop	Emergency stop.	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.
	S_EStopIn	SAFEBOOL	TRUE, FALSE	FALSE	Safety requirement input. FALSE: Request for safety-related response (e.g., emergency stop button being pressed). TRUE: No request for safety-related response (e.g., emergency stop button not pressed).
	S_StartReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to start reset.
	S_AutoReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to automatically reset.
	Reset	BOOL	TRUE, FALSE	FALSE	Manual reset
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	S_EStopOut	SAFEBOOL	TRUE, FALSE	FALSE	Output for safety-related response. FALSE: Safety output is disabled. A safety-related response is requested (e.g., emergency stop button pressed, reset required, or internal fault is active). TRUE: Safety output is enabled. No safety-related response is requested (e.g., emergency stop button not pressed, no internal fault is active).
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	ResetRequest	BOOL	TRUE, FALSE	FALSE	Flag indicating whether a reset is required.
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**
  - The S_EStopOut enabling signal is immediately reset to FALSE when the S_EStopIn input is set to FALSE. The S_EStopOut enabling signal only returns to TRUE when S_EStopIn input is set to TRUE and a reset operation occurs. The enabling of the reset logic depends on the defined inputs S_StartReset, S_AutoReset, and Reset.
    - If S_AutoReset = TRUE, reset is automatically executed.
    - If S_AutoReset = FALSE, reset is executed by a rising edge trigger of the Reset input.
    - If S_StartReset = TRUE, reset is automatically executed when the PES (Programmable Electronic System) starts for the first time.
    - If S_StartReset = FALSE, reset is executed by a rising edge trigger of the Reset input.
  - The S_StartReset and S_AutoReset inputs can only be activated when it is ensured that PES startup does not cause a hazardous status.
  - The SF_EmergencyStop function block can be used to monitor single-channel or dual-channel emergency stop buttons. For example, in dual-channel applications, the SF_Equivalent function block can be used to detect whether contact synchronization exceeds limits. The category classification according to EN ISO 13849-1 depends on the actual end components used.
  - SF_EmergencyStop automatically detects the static TRUE status of the Reset signal. Other fault detection (such as open circuit and short circuit) relies on the dedicated hardware used.
- **Status transition diagram**

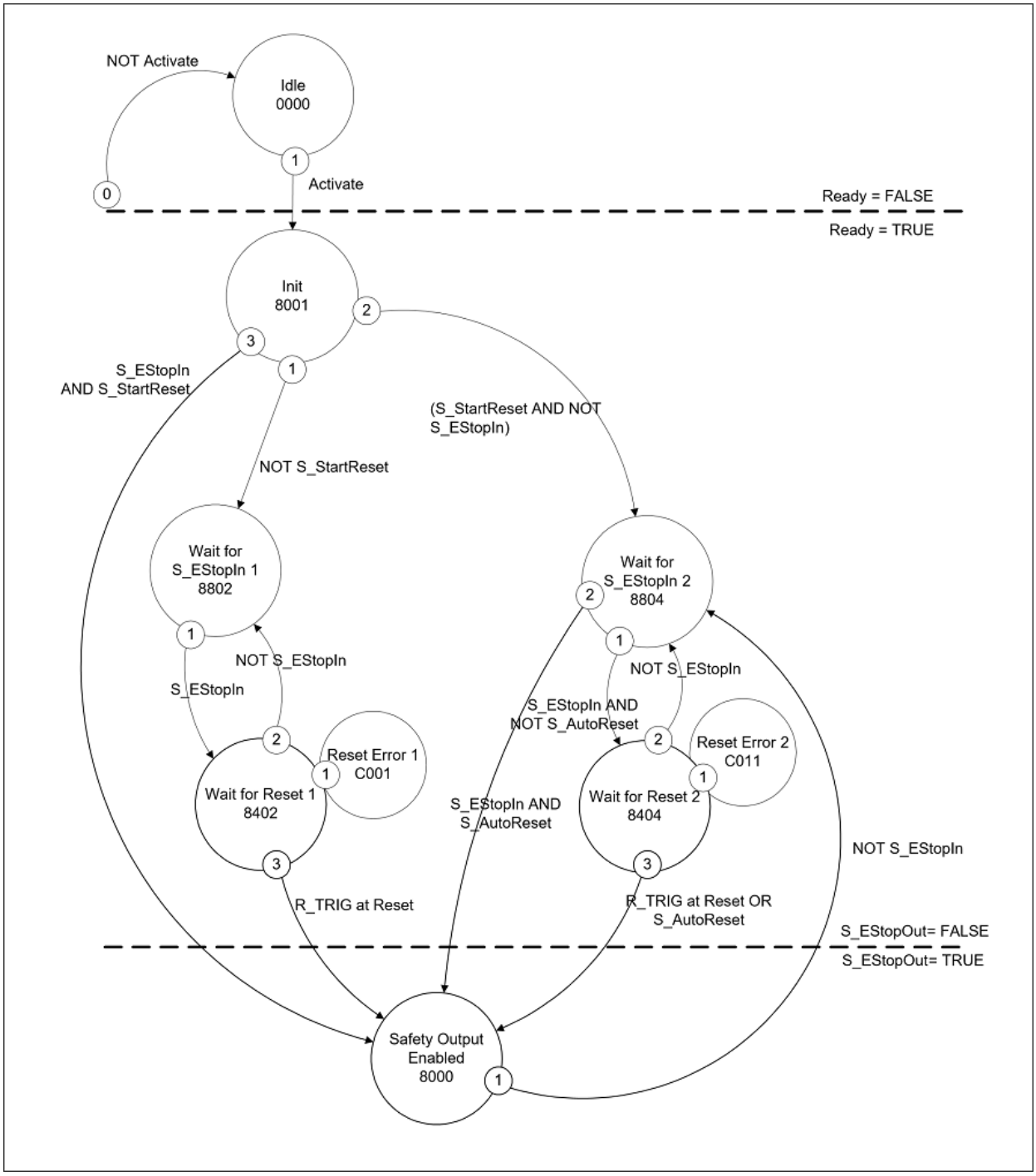
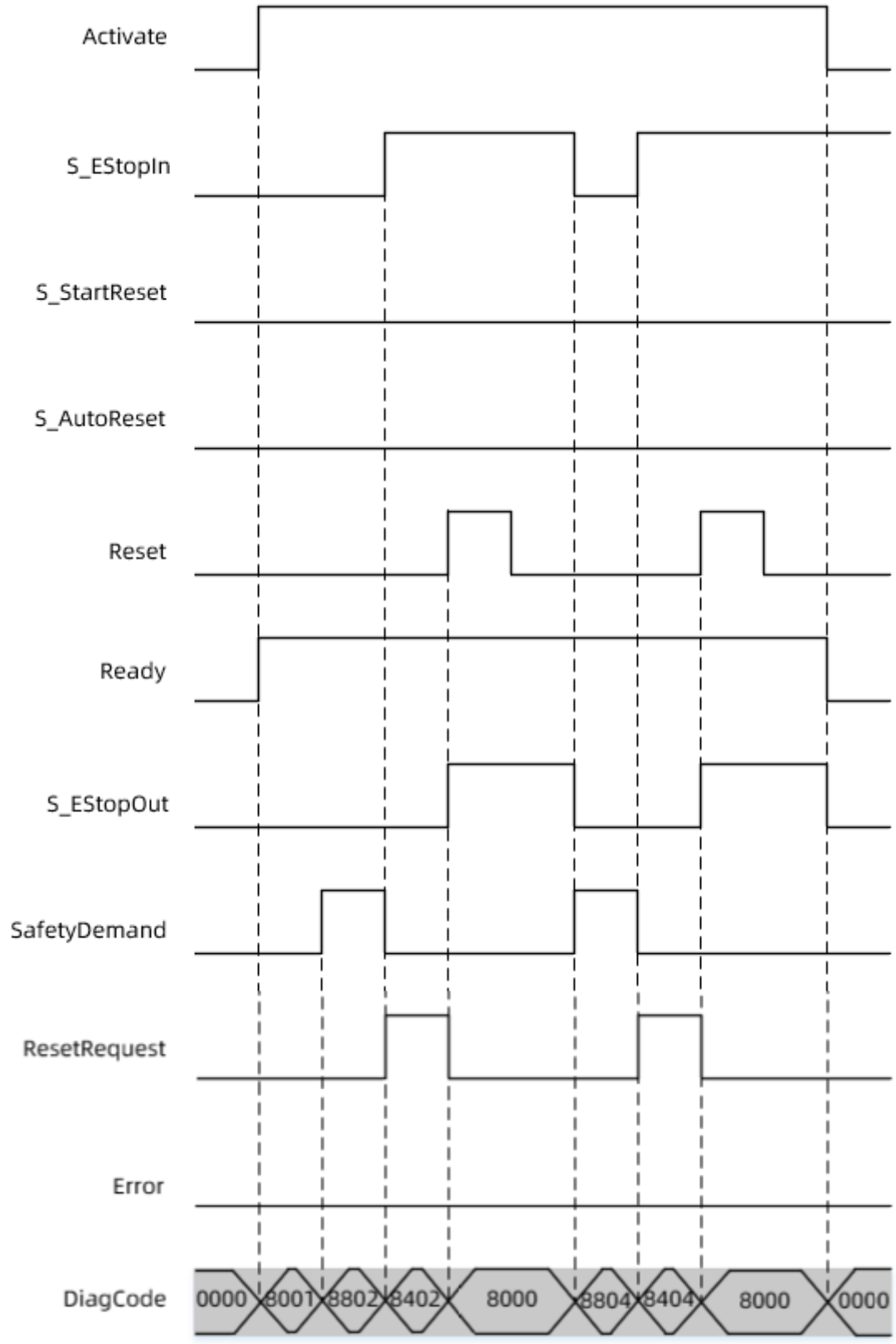


Figure 11-10-2-2-8-1 Status transition diagram for SF_EmergencySto instruction status transition diagram

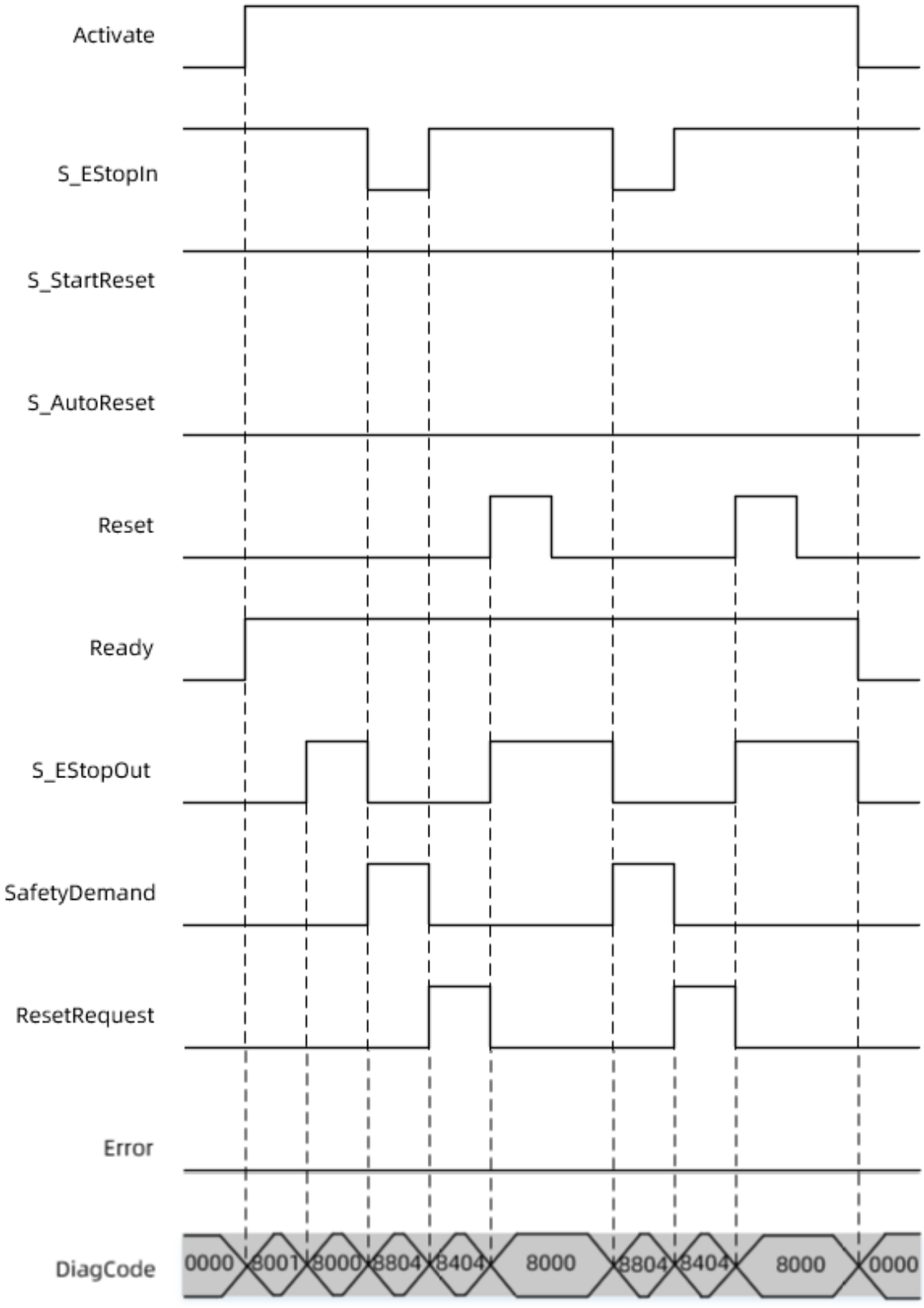
- This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.
- The transition from any status to the idle status when Activate = FALSE is not indicated, but the priority (0) for transitioning to the idle status is the highest.

- **Sequence diagram**

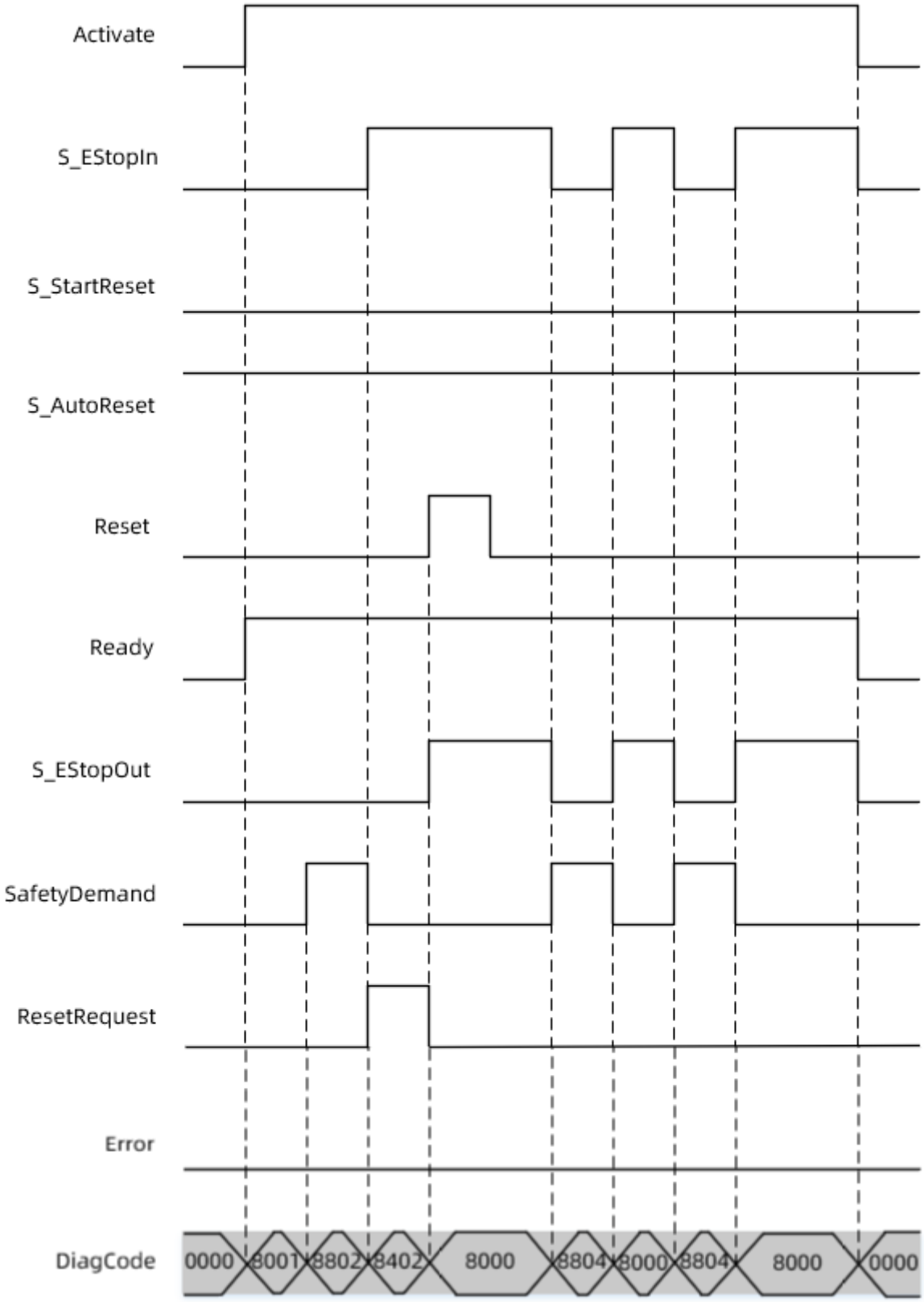
  S_StartReset=FALSE, S_AutoReset=FALSE



S_StartReset=TRUE, S_AutoReset=FALSE



S_StartReset = FALSE, S_AutoReset = TRUE



- **Fault detection**
  Detects that a TRUE value of Reset is triggered at the non-rising edge.
- **Fault behavior**
  When a fault occurs, reset operation must be performed based on the fault code indicated by DiagCode, combined with the status transition diagram.
- **Function block-specific fault codes and status codes**
  - Fault code

DiagCode	Status Name	Description and Output
C001	Reset Error 1	Reset is TRUE while waiting for S_EStopIn = TRUE. Ready = TRUE S_EStopOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C011	Reset Error 2	Reset is TRUE while waiting for S_EStopIn = TRUE. Ready = TRUE S_EStopOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

◦ Status code

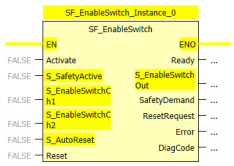
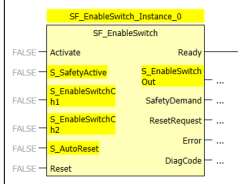
DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_EStopOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8001	Init (Initialization)	Function block is enabled, Check if S_StartReset is required. Ready = TRUE S_EStopOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8802	Wait for S_EstopIn 1	Check whether Reset is FALSE and wait for S_EStopIn = TRUE. Ready = TRUE S_EStopOut = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8402	Wait for Reset 1	S_EStopIn = TRUE. Wait for the rising edge of Reset. Ready = TRUE S_EStopOut = FALSE SafetyDemand = FALSE ResetRequest = TRUE Error = FALSE
8804	Wait for S_EstopIn 2	The function block detected a safety request. Check whether Reset is FALSE and wait for S_EStopIn = TRUE. Ready = TRUE S_EStopOut = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8404	Wait for Reset 2	S_EStopIn = TRUE. Check S_AutoReset or wait for the rising edge trigger of Reset. Ready = TRUE S_EStopOut = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = FALSE
8000	Safety Output Enabled	S_EStopIn = TRUE. In function mode, S_EStopOut = TRUE. Ready = TRUE S_EStopOut = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

11.10.2.2.9 SF_EnableSwitch

SF_EnableSwitch evaluates the signals of the enable switch with three positions.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_EnableSwitch	Enable switch	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.
	S_SafetyActive	SAFEBOOL	TRUE, FALSE	FALSE	Confirm safety mode (limiting operating speed, power, or range). FALSE: Safety mode is not activated. TRUE: Safety mode is activated.
	S_EnableSwitchCh1	SAFEBOOL	TRUE, FALSE	FALSE	Signal connected to the E1 and E2 contacts of the enable switch. FALSE: The connected switch is open. TRUE: The connected switch is closed.
	S_EnableSwitchCh2	SAFEBOOL	TRUE, FALSE	FALSE	Signal connected to the E3 and E4 contacts of the enable switch. FALSE: The connected switch is open. TRUE: The connected switch is closed.
	S_AutoReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to automatically reset.
	Reset	BOOL	TRUE, FALSE	FALSE	Manual reset
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	S_EnableSwitchOut	SAFEBOOL	TRUE, FALSE	FALSE	Safety-related output indicating whether the safety protection function is paused. FALSE: The safety protection function fails to be paused. TRUE: The safety protection function is paused.
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	ResetRequest	BOOL	TRUE, FALSE	FALSE	Flag indicating whether a reset is required.

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**
  - The SF_EnableSwitch function block supports pausing safety protective function (DIN EN 60204 section 9.2.4) via an enable switch (DIN EN 60204 section 9.2.5.8), provided that the relevant operating mode has been selected and activated. The relevant operating mode (limiting operating speed, power, or range) must be selected outside the SF_EnableSwitch function block.
  - The SF_EnableSwitch function block evaluates the signals of the three-position enable switch conforming to DIN EN 60204 section 9.2.5.8. Input parameters S_EnableSwitchCh1 and S_EnableSwitchCh2 are used to process signal levels from contacts E1 to E4, as shown below.

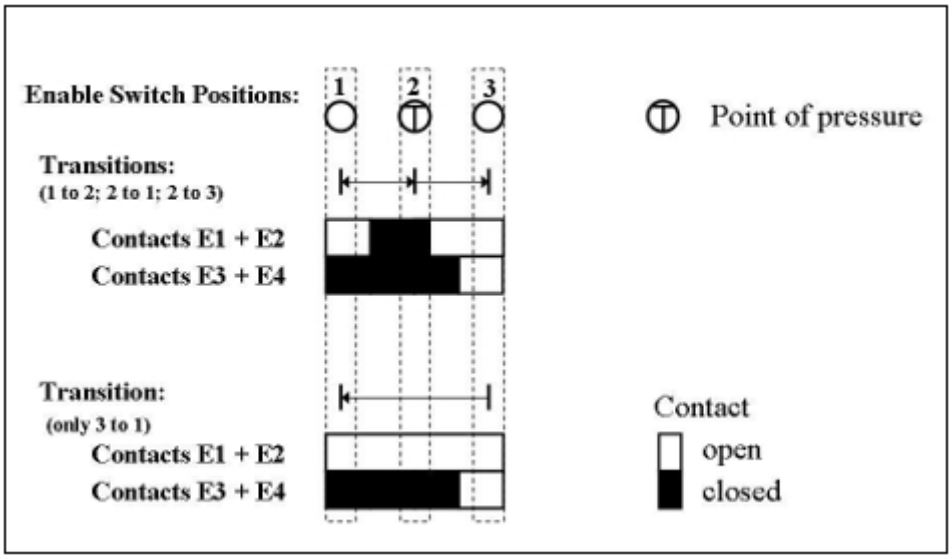


Figure 11-10-2-2-9-1 Switch position

  - Signals E1 and E2 must be connected to parameter S_EnableSwitchCh1; signals E3 and E4 must be connected to parameter S_EnableSwitchCh2. The function block detects the enable switch position status through this signal sequence.
  - The transition from position 2 to position 3 may differ from the method described here. The function block can detect the switch direction (position 1 -> position 2/position 3 -> position 2) using predefined enable switch contact signal sequences. The safety protective function can be paused only when the enable switch transitions from position 1 to position 2; other transition directions or positions are invalid. This measure complies with requirement EN 60204 section 9.2.5.8.
  - To meet the requirements of DIN EN 60204 section 9.2.4, users must select a compliant switching device and ensure that the relevant operating mode is selected in the application (disable automatic operation in this mode by appropriate measures, which is in compliance with DIN EN 60204 section 9.2.3).
  - Operating modes are typically configured using the operating mode selector switch with the SF_ModeSelector function block and the SF_SafeRequest or SF_SafelyLimitedSpeed function blocks.

The SF_EnableSwitch function block confirms the safety mode status via the S_SafetyActive parameter. If the safety mode is not confirmed in the application, a static TRUE signal must be connected to the S_SafetyActive parameter.

- The S_AutoReset input can only be activated when it is ensured that the PES startup will not cause a hazardous status.

- **Status transition diagram**

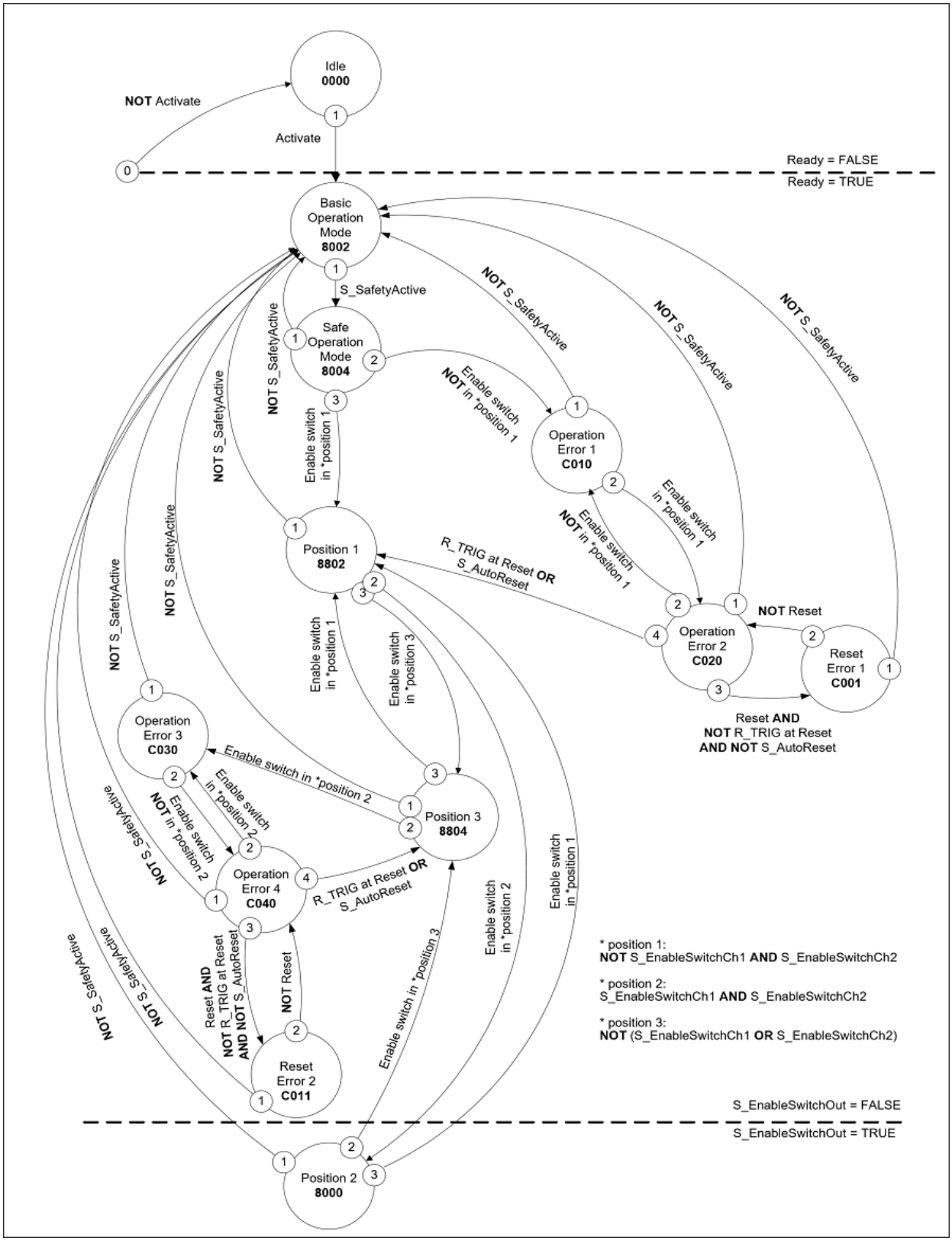
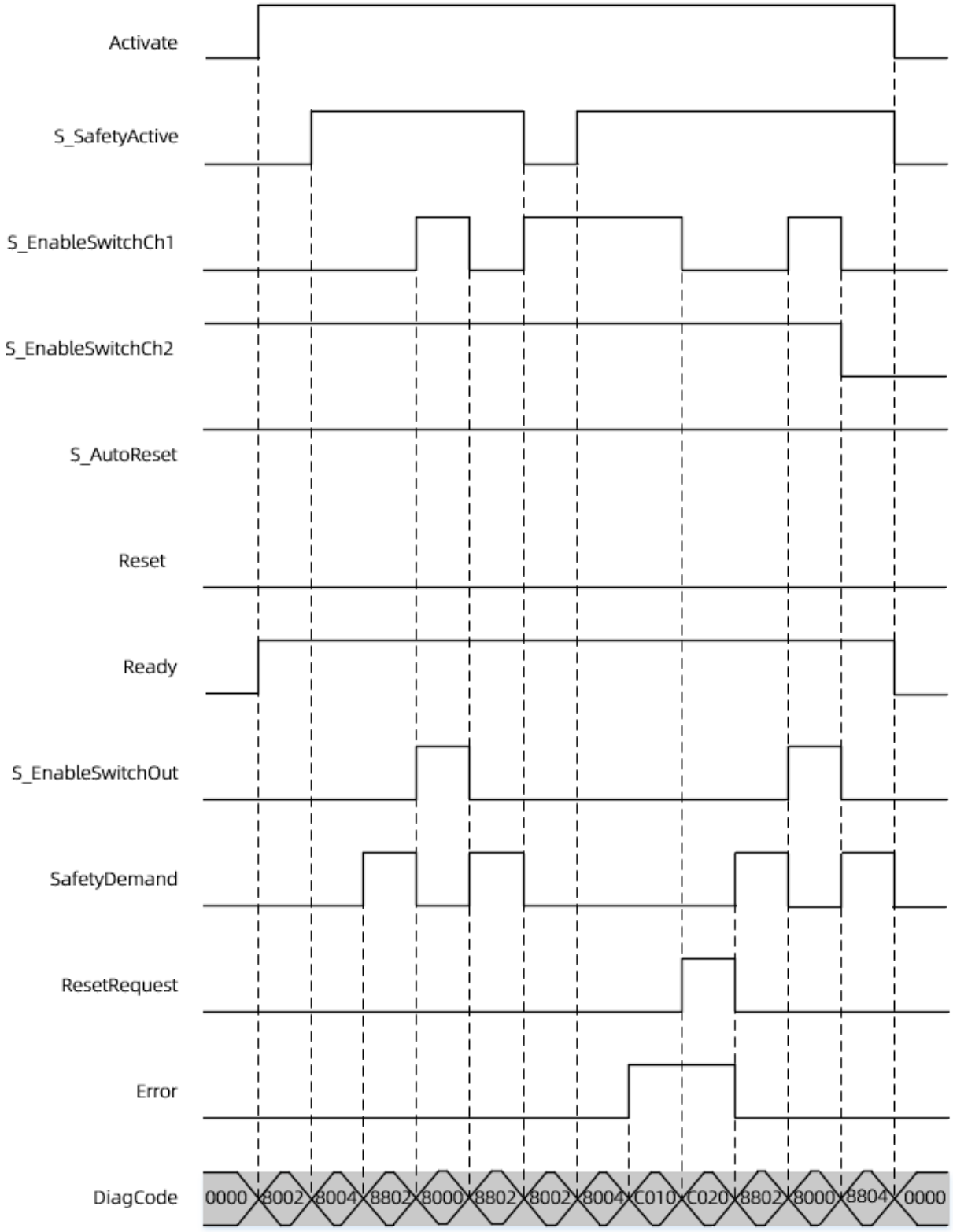


Figure 11-10-2-2-9-2 Status transition diagram for SF_EnableSwitch

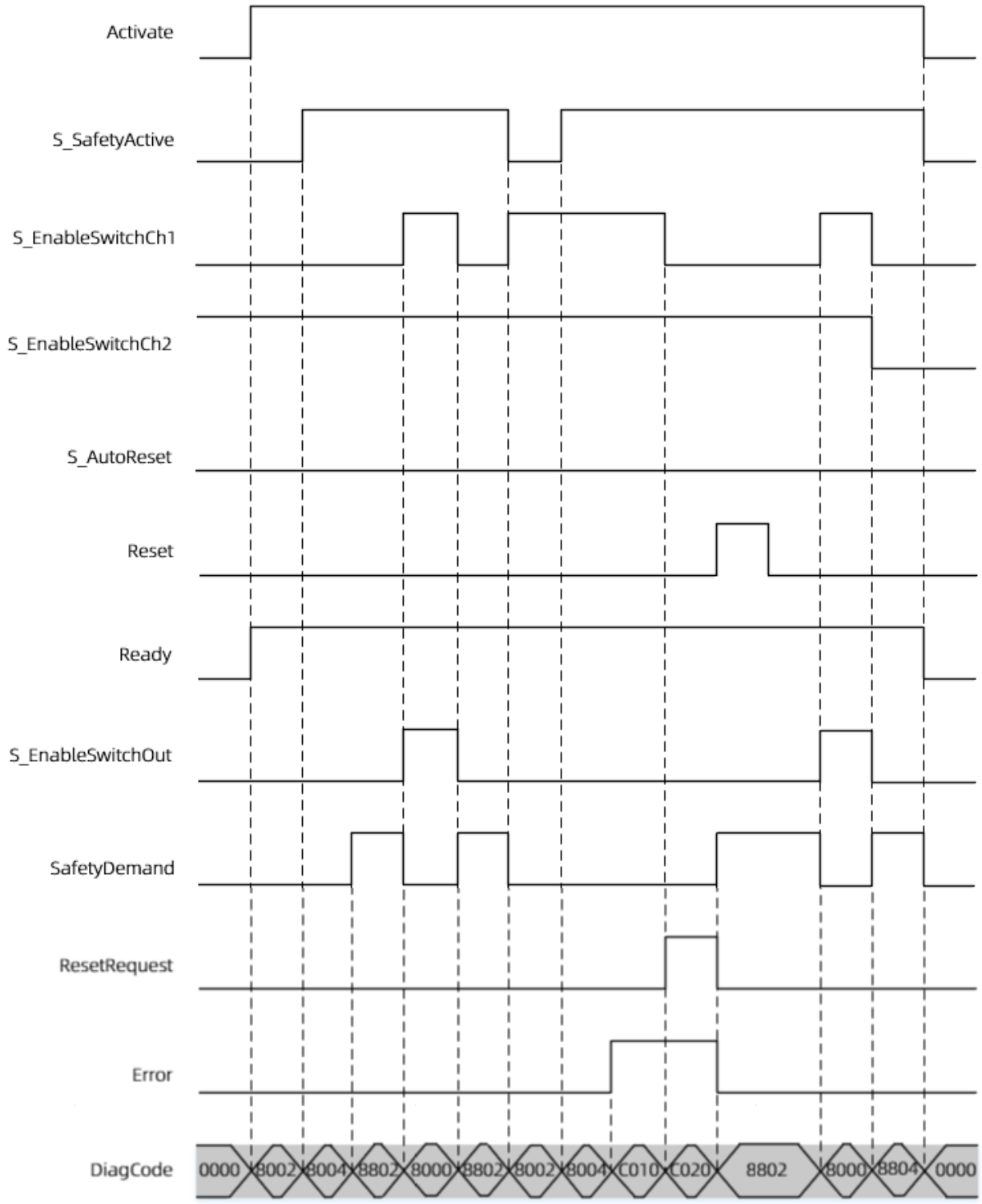
- This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.
- The transition from any status to the idle status when Activate = FALSE is not indicated, but the priority (0) for transitioning to the idle status is the highest.

- **Sequence diagram**

  S_AutoReset = TRUE



S_AutoReset = FALSE



- **Fault detection**

  The faulty status is triggered by the following conditions:

  - Detects that a TRUE value of Reset is triggered at the non-rising edge.
  - The switch position is invalid.

- **Fault behavior**

  - When a fault occurs, the S_EnableSwitchOut safety output is set to FALSE and remains in this safety status. Unlike other function blocks, the faulty status can be cleared by setting Reset to FALSE, or on this basis, by setting S_SafetyActive to FALSE.

- After the fault is cleared, the enable switch must be in the initial position specified by the process flow before the S_EnableSwitchOut output can be set to TRUE via the enable switch. If S_AutoReset = FALSE, a TRUE value of Reset shall be triggered at the rising edge.

- **Function block-specific fault codes and status codes**
  - Fault code

DiagCode	Status Name	Description and Output
C001	Reset Error 1	In the C020 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_EnableSwitchOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C011	Reset Error 2	In the C040 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_EnableSwitchOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C010	Operation Error 1	The enable switch is not in position 1 when S_SafetyActive is activated. Ready = TRUE S_EnableSwitchOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C020	Operation Error 2	The enable switch is in position 1 after status C010. Ready = TRUE S_EnableSwitchOut = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = TRUE

DiagCode	Status Name	Description and Output
C030	Operation Error 3	The enable switch is in position 2 after position 3. Ready = TRUE S_EnableSwitchOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C040	Operation Error 4	The enable switch is not in position 2 after status C030. Ready = TRUE S_EnableSwitchOut = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = TRUE

◦ Status code

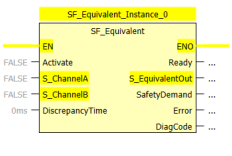
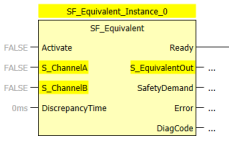
DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_EnableSwitchOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8002	Basic Operation Mode	The safety operation mode is not activated. Ready = TRUE S_EnableSwitchOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8004	Safe Operation Mode	The safety operation mode is activated. Ready = TRUE S_EnableSwitchOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8802	Position 1	The safety operation mode is activated and the enable switch is in position 1. Ready = TRUE S_EnableSwitchOut = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8804	Position 3	The safety operation mode is activated and the enable switch is in position 3. Ready = TRUE S_EnableSwitchOut = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8000	Position 2	The safety operation mode is activated and the enable switch is in position 2. Ready = TRUE S_EnableSwitchOut = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

11.10.2.2.10 SF_Equivalent

SF_Equivalent monitors the discrepancy time between two equal safety input values.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_Equivalent	Equivalent input	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
	S_ChannelA	SAFEBOOL	TRUE, FALSE	FALSE	Input A of the logical connection. FALSE: Contactor A is open. TRUE: Contactor A is closed.
	S_ChannelB	SAFEBOOL	TRUE, FALSE	FALSE	Input B of the logical connection. FALSE: Contactor B is open. TRUE: Contactor B is closed.
	DiscrepancyTime	TIME	Follows data type	T#0ms	Used to set the maximum discrepancy time between the two input statuses.
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	S_EquivalentOut	SAFEBOOL	TRUE, FALSE	FALSE	Safety-related output FALSE: At least one input signal is FALSE or the status change exceeds the monitoring time. TRUE: Both input signals are activated and the status change is within the monitoring time.
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**
  - This function block converts two equivalent SAFEBOOL inputs into a SAFEBOOL output with discrepancy time monitoring. Input channels A and B are interdependent. The function block output displays the evaluation result of both channels. If the signal of either channel changes from TRUE to FALSE, the output will be immediately shut down (FALSE) for safety reasons.
  - Discrepancy time monitoring: The discrepancy time is the maximum duration during which the two inputs are allowed to remain in different statuses without triggering the function block's fault detection. Discrepancy time monitoring starts immediately whenever the status of either input changes. If the two inputs have not reached the same status by the end of the monitoring, the function block detects a fault.

◦ Inputs must switch symmetrically. This means the monitoring process applies to both the turn-on (switching to TRUE) and turn-off (switching to FALSE) operations of the input signals.

- **Status transition diagram**

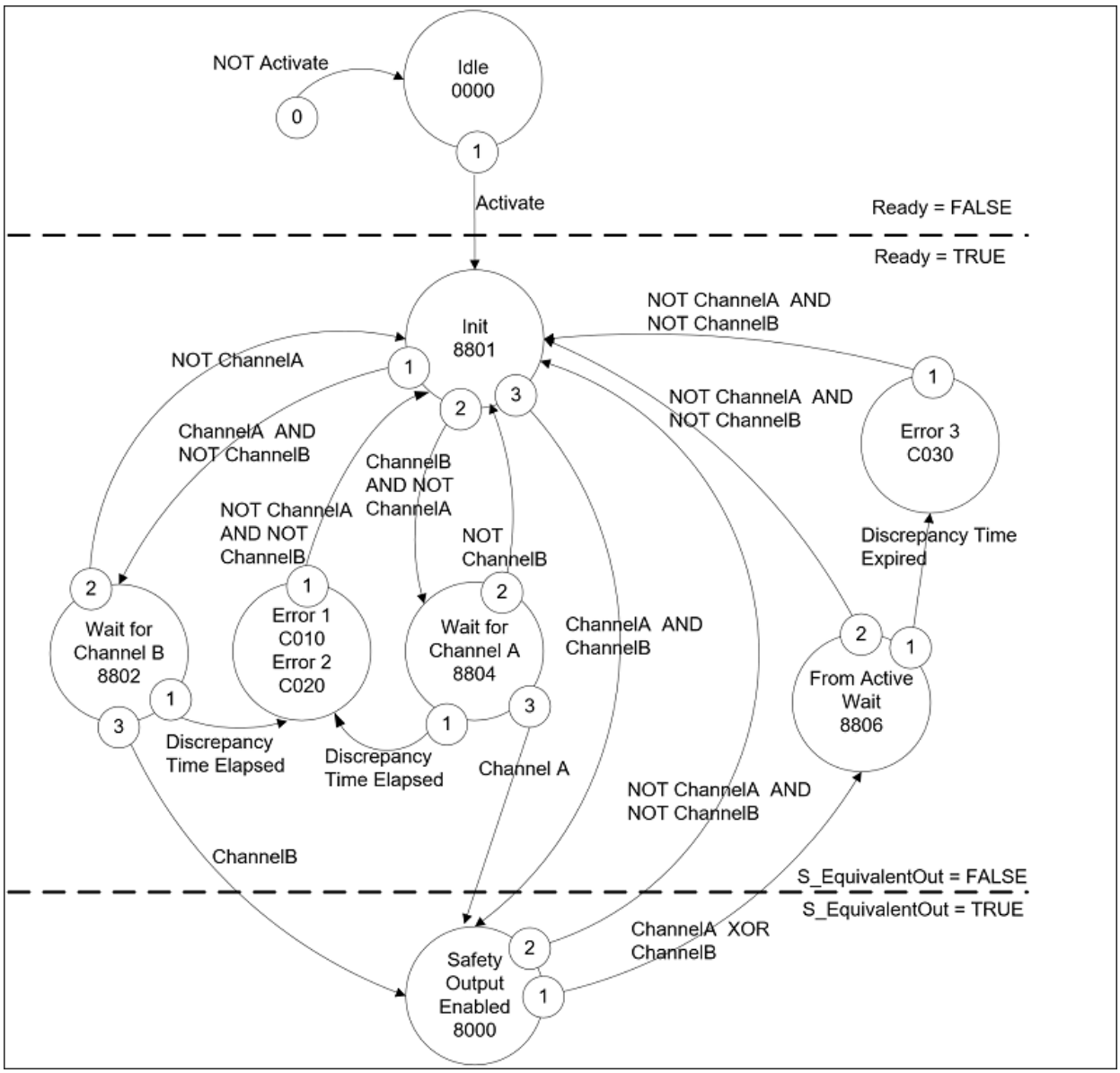
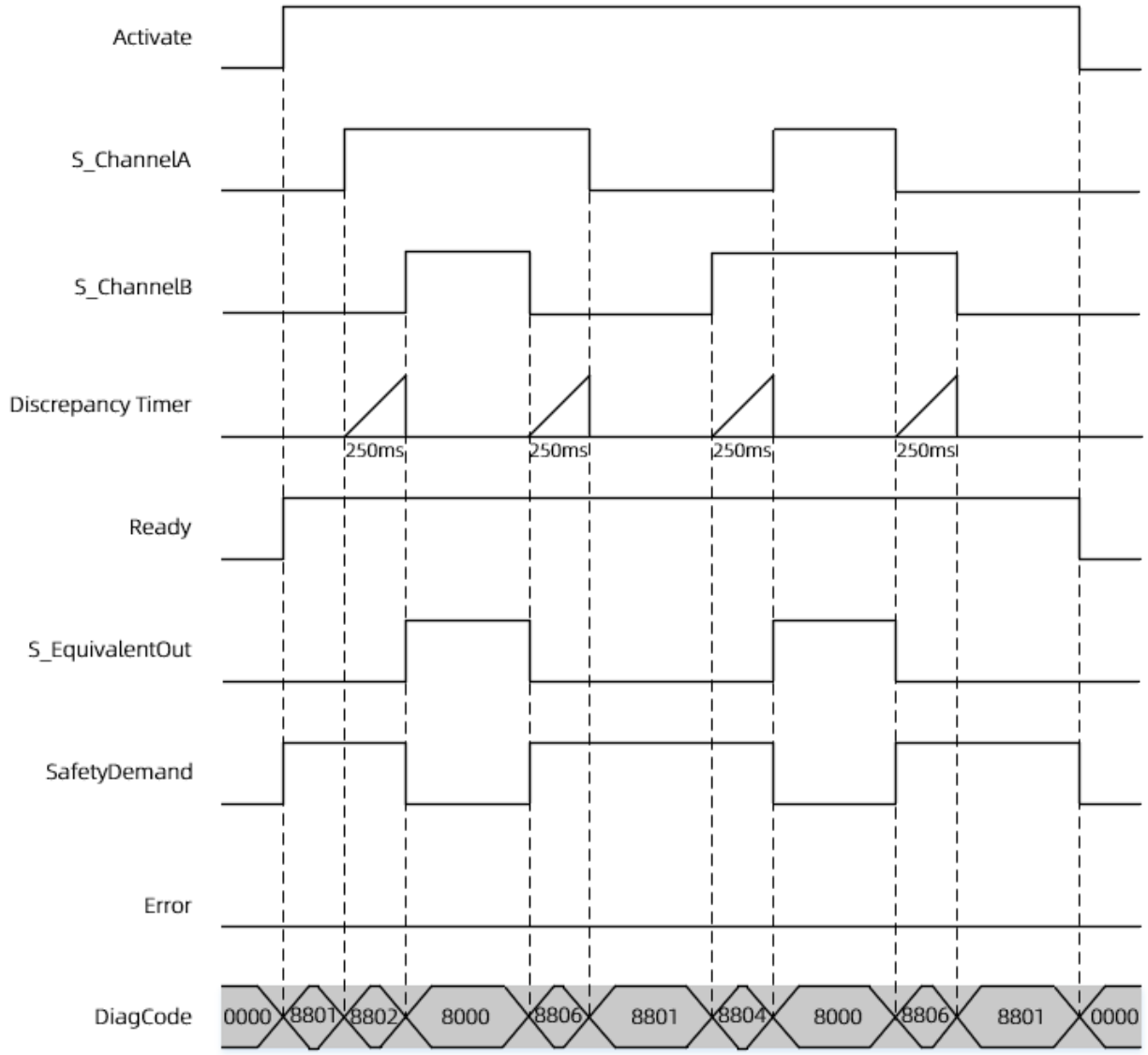


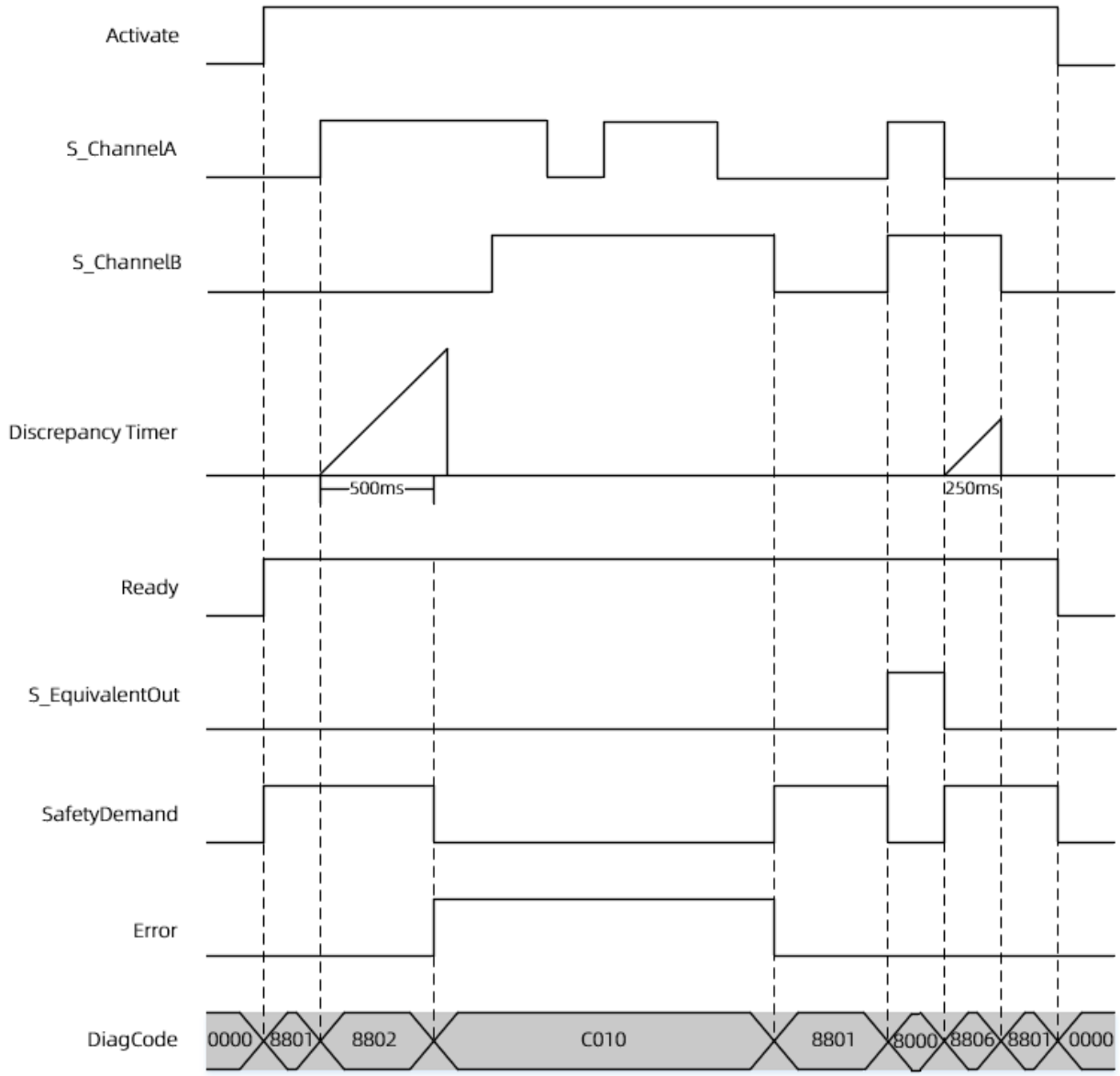
Figure 11-10-2-2-10-1 Status transition diagram for SF_Equivalent

◦ This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.

◦ The transition from any status to the idle status when Activate = FALSE is not indicated, but the priority (0) for transitioning to the idle status is the highest.

- **Sequence diagram**


Activate
S_ChannelA
S_ChannelB
Discrepancy Timer
250ms
250ms
250ms
250ms
Ready
S_EquivalentOut
SafetyDemand
Error
DiagCode
0000
8801
8802
8000
8806
8801
8804
8000
8806
8801
0000



- **Fault detection**

  The function block monitors the discrepancy time between channel A and channel B.

- **Fault behavior**

  When a fault occurs, reset operation must be performed based on the fault code indicated by DiagCode, combined with the status transition diagram.

- **Function block-specific fault codes and status codes**

  - Fault code

DiagCode	Status Name	Description and Output
C010	Error 1	Discrepancy time timeout in status 8802. Ready = TRUE S_EquivalentOut = FALSE SafetyDemand = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
C020	Error 2	Discrepancy time timeout in status 8804. Ready = TRUE S_EquivalentOut = FALSE SafetyDemand = FALSE Error = TRUE
C030	Error 3	Discrepancy time timeout in status 8806. Ready = TRUE S_EquivalentOut = FALSE SafetyDemand = FALSE Error = TRUE

◦ Status code

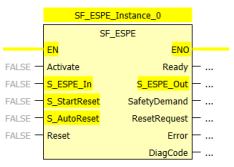
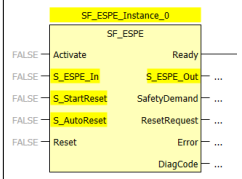
DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_EquivalentOut = FALSE SafetyDemand = FALSE Error = FALSE
8801	Init (Initialization)	The function block has detected activation and is activated. Ready = TRUE S_EquivalentOut = FALSE SafetyDemand = TRUE Error = FALSE
8000	Safety Output Enabled	The input is switched to TRUE in equivalent mode. Ready = TRUE S_EquivalentOut = TRUE SafetyDemand = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8802	Wait for Channel B	Channel A is switched to TRUE and is waiting for Channel B. Discrepancy timer is started. Ready = TRUE S_EquivalentOut = FALSE SafetyDemand = TRUE Error = FALSE
8804	Wait for Channel A	Channel B is switched to TRUE and is waiting for Channel A. Discrepancy timer is started. Ready = TRUE S_EquivalentOut = FALSE SafetyDemand = TRUE Error = FALSE
8806	From Active Wait	A channel is switched to FALSE and is waiting for the second channel to be switched to FALSE. Discrepancy timer is started. Ready = TRUE S_EquivalentOut = FALSE SafetyDemand = TRUE Error = FALSE

11.10.2.2.11 SF_ESPE

SF_ESPE monitors the electrically sensitive protective equipment (ESPE).

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_ESPE	Electrically sensitive protective equipment	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.
	S_ESPE_In	SAFEBOOL	TRUE, FALSE	FALSE	Safety request input. FALSE: Safety mode is not activated. TRUE: Safety mode is activated.

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
	S_StartReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to start reset.
	S_AutoReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to automatically reset.
	Reset	BOOL	TRUE, FALSE	FALSE	Manual reset
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	S_ESPE_Out	SAFEBOOL	TRUE, FALSE	FALSE	Output for safety-related response. FALSE: Safety output is disabled. Safety-related response is required (e.g., reset is required or internal fault is active). TRUE: Safety output is enabled. No safety-related response is required.
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	ResetRequest	BOOL	TRUE, FALSE	FALSE	Flag indicating whether a reset is required.
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**
  - This function block is used to monitor the electrical sensitive protective equipment (ESPE), and its function is identical to SF_EmergencyStop. When the S_ESPE_In input is set to FALSE, the S_ESPE_Out output signal is immediately set to FALSE. The S_ESPE_Out output signal is set to TRUE only when the S_ESPE_In input is set to TRUE and a reset is performed. The enabling of the reset depends on the defined inputs S_StartReset, S_AutoReset and Reset.
    - If S_AutoReset = TRUE, reset is automatically executed.
    - If S_AutoReset = FALSE, reset is executed by a rising edge trigger of the Reset input.
    - If S_StartReset = TRUE, reset is automatically executed when the PES (Programmable Electronic System) starts for the first time.
    - If S_StartReset = FALSE, reset is executed by a rising edge trigger of the Reset input.
  - The S_StartReset and S_AutoReset inputs can only be activated when it is ensured that PES startup does not cause a hazardous status.
  - ESPE selection must comply with EN IEC 61496-1, -2, and -3 standards, and the corresponding category must be determined according to EN 954-1 requirements.
- **Status transition diagram**

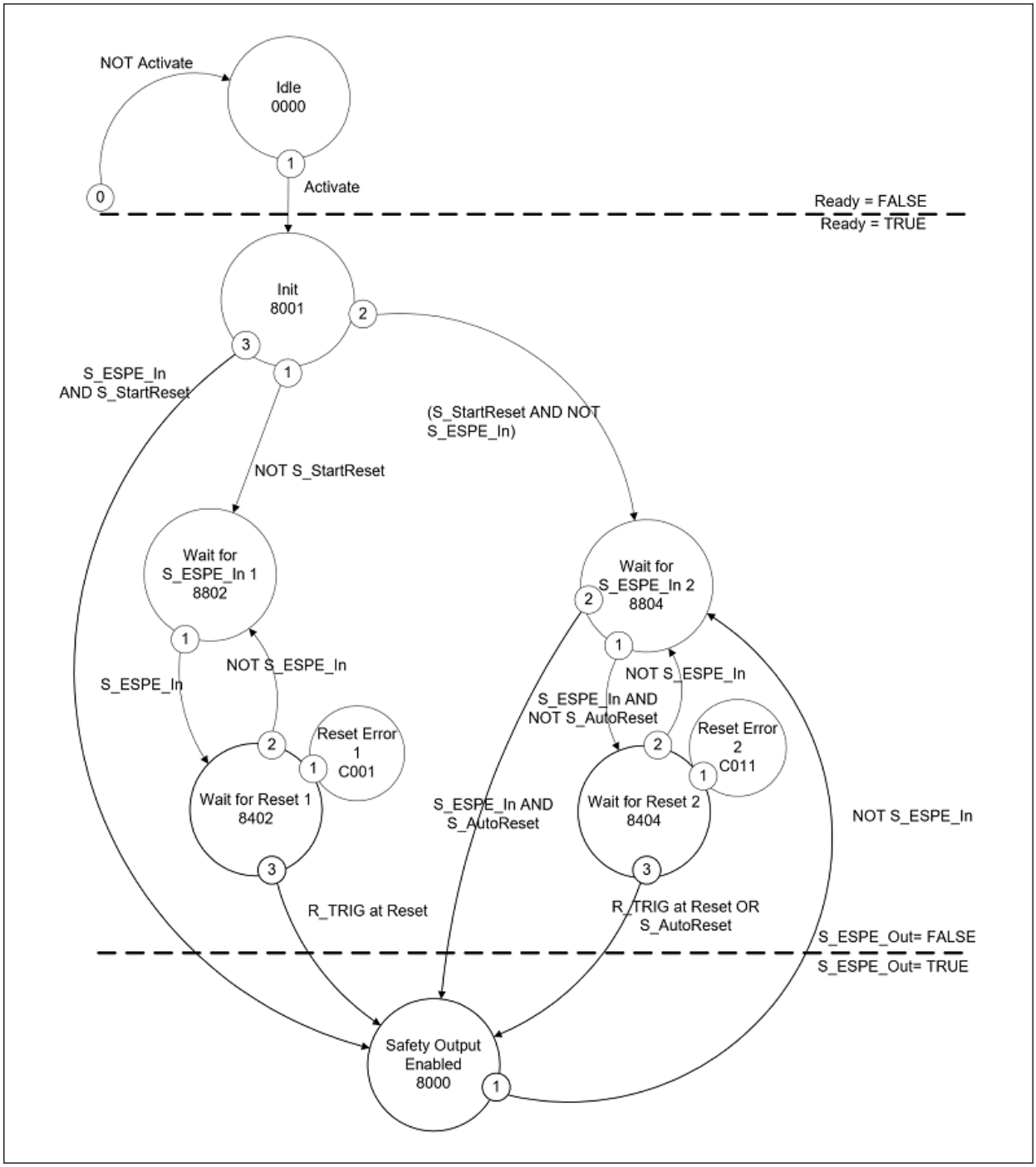
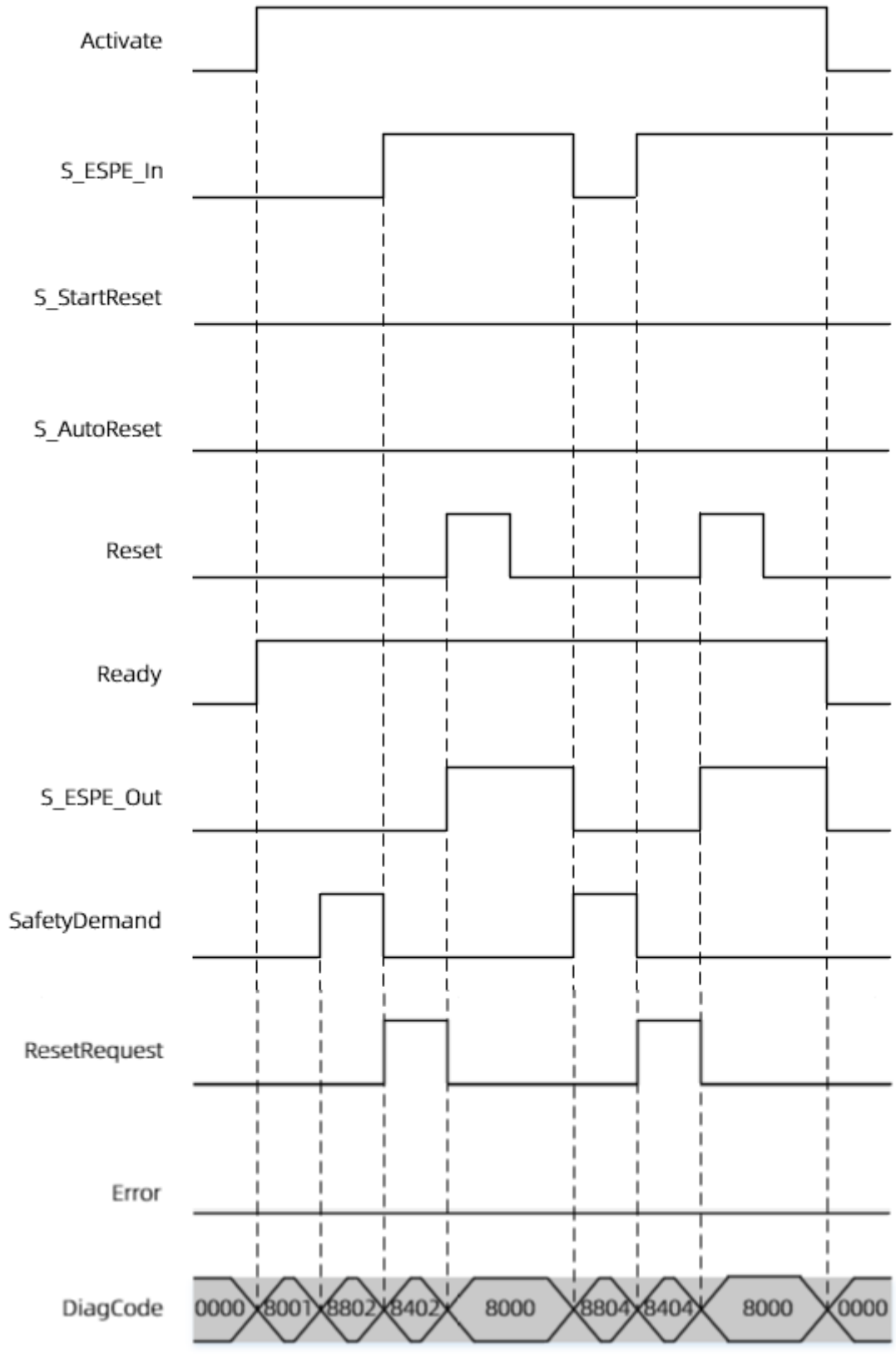


Figure 11-10-2-2-11-1 Status transition diagram for SF_ESPE

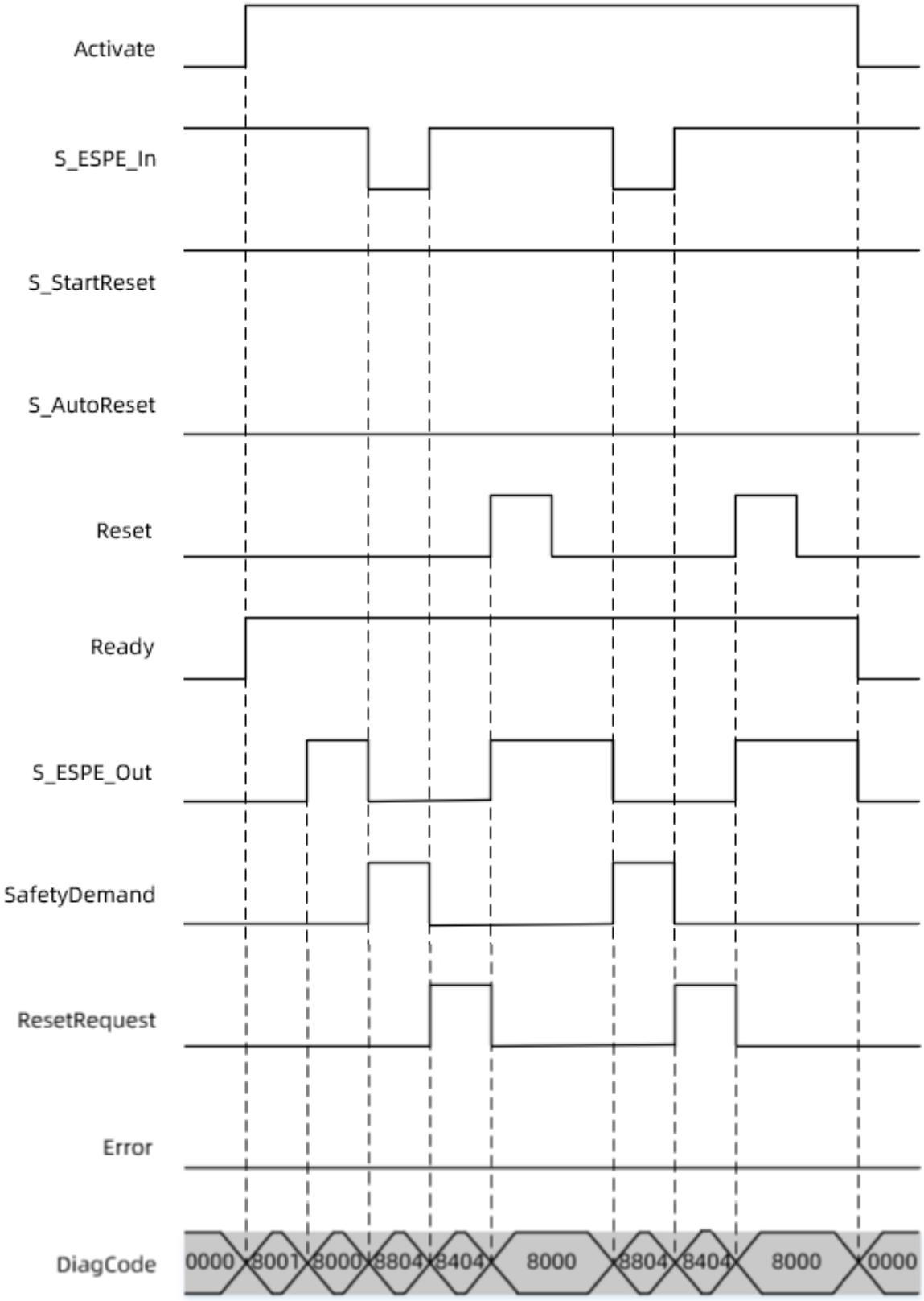
This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.

- **Sequence diagram**

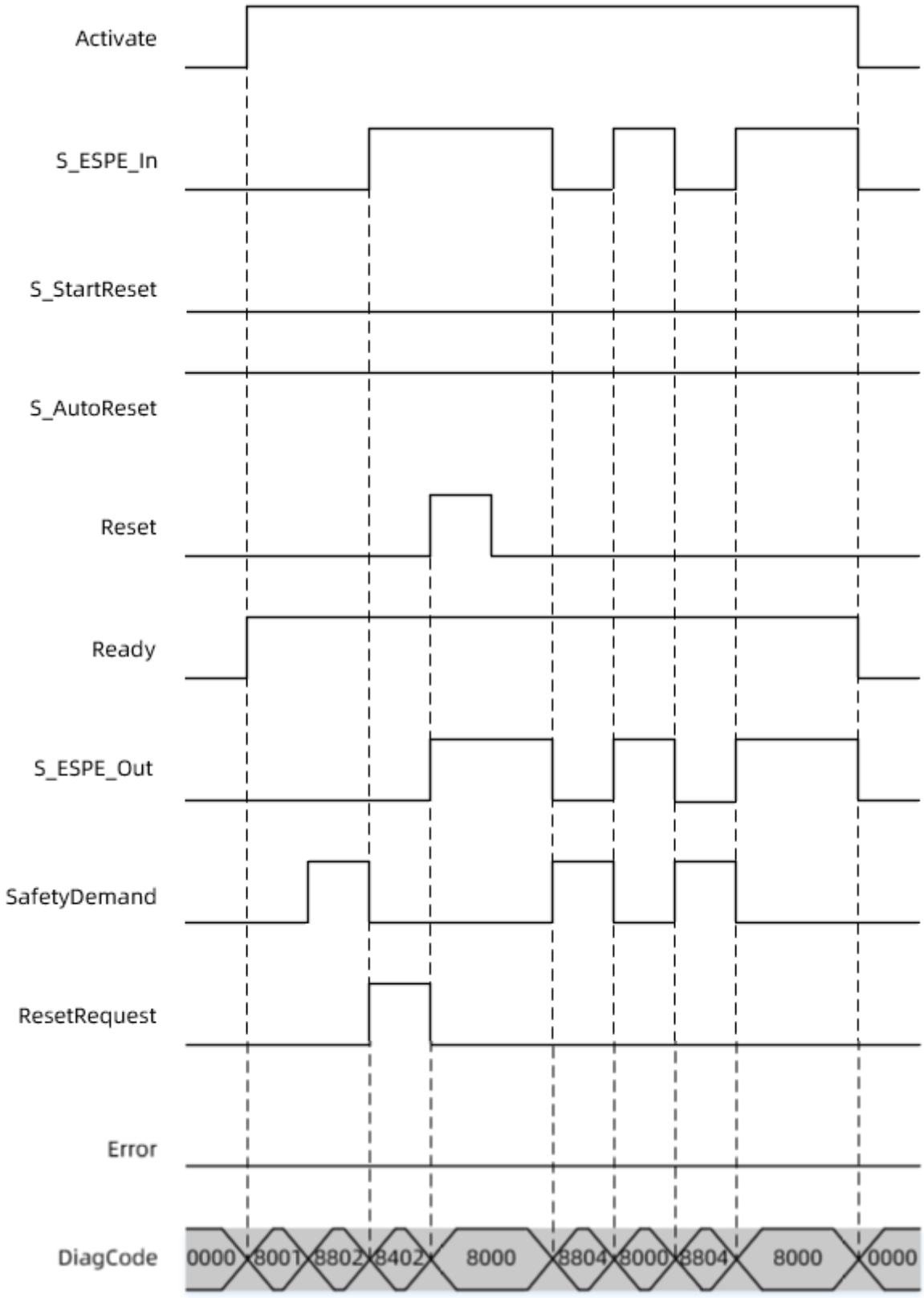
  S_StartReset=FALSE, S_AutoReset=FALSE



S_StartReset=TRUE, S_AutoReset=FALSE



S_StartReset=FALSE, S_AutoReset=TRUE



- **Fault detection**

  Detects that a TRUE value of Reset is triggered at the non-rising edge.

- **Fault behavior**

  S_ESPE_Out is set to FALSE. When the Reset input is a static TRUE signal, the DiagCode output indicates the corresponding fault code, and the Error output is set to TRUE.

  To exit the faulty status, Reset must be set to FALSE.

- **Function block-specific fault codes and status codes**
  - Fault code

DiagCode	Status Name	Description and Output
C001	Reset Error 1	Reset is TRUE while waiting for S_ESPE_In = TRUE. Ready = TRUE S_ESPE_Out = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C011	Reset Error 2	Reset is TRUE while waiting for S_ESPE_In = TRUE. Ready = TRUE S_ESPE_Out = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

  - Status code

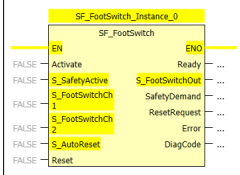
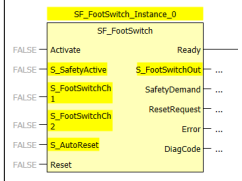
DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_ESPE_Out = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8001	Init (Initialization)	Function block is enabled, Check if S_StartReset is required. Ready = TRUE S_ESPE_Out = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8802	Wait for S_ESPE_In 1	Check whether Reset is FALSE and wait for S_ESPE_In = TRUE. Ready = TRUE S_ESPE_Out = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8402	Wait for Reset 1	S_ESPE_In = TRUE. Wait for the rising edge of Reset. Ready = TRUE S_ESPE_Out = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = FALSE
8804	Wait for S_ESPE_In 2	The function block detected a safety request. Check whether Reset is FALSE and wait for S_ESPE_In = TRUE. Ready = TRUE S_ESPE_Out = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8404	Wait for Reset 2	S_ESPE_In = TRUE. Check S_AutoReset or wait for the rising edge trigger of Reset. Ready = TRUE S_ESPE_Out = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = FALSE
8000	Safety Output Enabled	S_ESPE_In = TRUE. In function mode, S_ESPE_Out = TRUE. Ready = TRUE S_ESPE_Out = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

11.10.2.2.12 SF_FootSwitch

SF_FootSwitch evaluates the signals of a foot switch with three positions.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_FootSwitch	Foot switch	FB		

- **Related variables**

Variables

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.
	S_SafetyActive	SAFEBOOL	TRUE, FALSE	FALSE	Confirm safety mode (limiting operating speed, power, or range). FALSE: Safety mode is not activated. TRUE: Safety mode is activated.
	S_FootSwitchCh1	SAFEBOOL	TRUE, FALSE	FALSE	Signal connected to the E1 and E2 contacts of the foot switch. FALSE: The connected switch is open. TRUE: The connected switch is closed.
	S_FootSwitchCh2	SAFEBOOL	TRUE, FALSE	FALSE	Signal connected to the E3 and E4 contacts of the foot switch. FALSE: The connected switch is open. TRUE: The connected switch is closed.
	S_AutoReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to automatically reset.
	Reset	BOOL	TRUE, FALSE	FALSE	Manual reset
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	S_FootSwitchOut	SAFEBOOL	TRUE, FALSE	FALSE	Safety-related output indicating whether the safety protection function is paused. FALSE: The safety protection function fails to be paused. TRUE: The safety protection function is paused.

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	ResetRequest	BOOL	TRUE, FALSE	FALSE	Flag indicating whether a reset is required.
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**
  - The SF_FootSwitch function block supports pausing safety protection (DIN EN 60204 section 9.2.4) via a foot switch (DIN EN 60204 section 9.2.5.8), provided that the relevant operating mode has been selected and activated. The relevant operating mode (limiting operating speed, power, or range) must be selected outside the SF_FootSwitch function block.
  - The SF_FootSwitch function block evaluates the signals of a foot switch with three positions (DIN EN 60204 section 9.2.5.8).

    Input parameters S_FootSwitchCh1 and S_FootSwitchCh2 are used to process signal levels from contacts E1 to E4, as shown below.

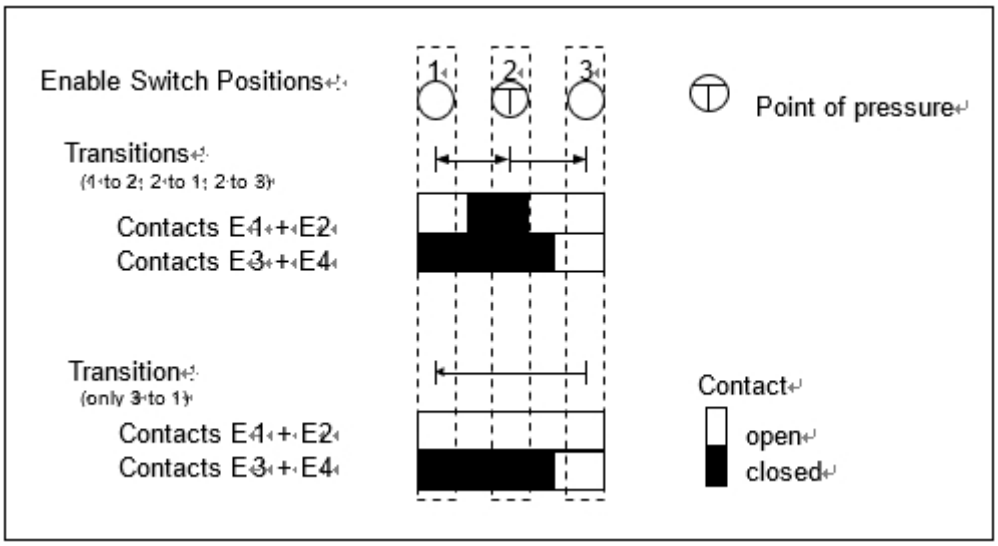


Figure 11-10-2-2-12-1 Switch position

  - Signals E1 and E2 must be connected to parameter S_FootSwitchCh1. Signals E3 and E4 must be connected to parameter S_FootSwitchCh2. The foot switch position is detected by the function block based on the time sequence of signals. However, the transition from position 2 to 3 may differ from this diagram.
  - The function block can detect the switch transition (position 1 -> position 2/position 3 -> position 2) using predefined foot switch contact signal sequences. The safety protective function can only be paused when the foot switch transitions from position 1 to position 2. Other transition directions or switch positions cannot be used to pause the safety protective function. This measure complies with requirement 9.2.5.8 of EN 60204.

- To meet the requirements of section 9.2.4 of DIN EN 60204, users shall use appropriate switch devices. Additionally, users must ensure that the relevant operating mode is selected in the application (section 9.2.3 of DIN EN 60204) (in this operating mode, automatic operation must be disabled by appropriate measures).
- The operating mode is typically switched to a safety status using instructions such as SF_ModeSelector and SF_SafetyRequest.
- The SF_EnableSwitch function block confirms the safety mode through S_SafetyActive. If the safety mode is not confirmed before implementation, S_SafetyActive will be TRUE.
- Input S_AutoReset shall only be activated when it is ensured that no hazardous situation will occur during PES startup.

- **Status transition diagram**

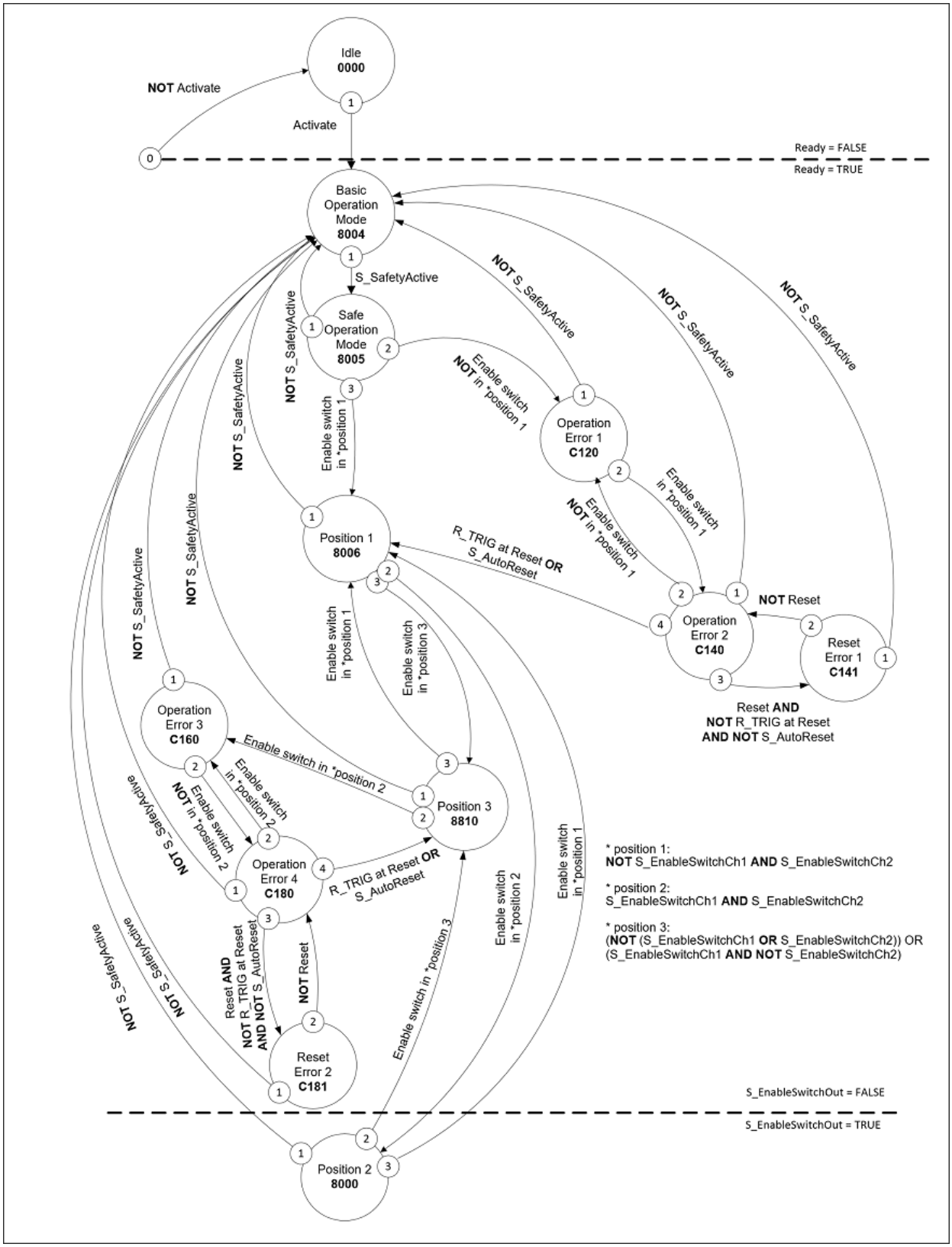
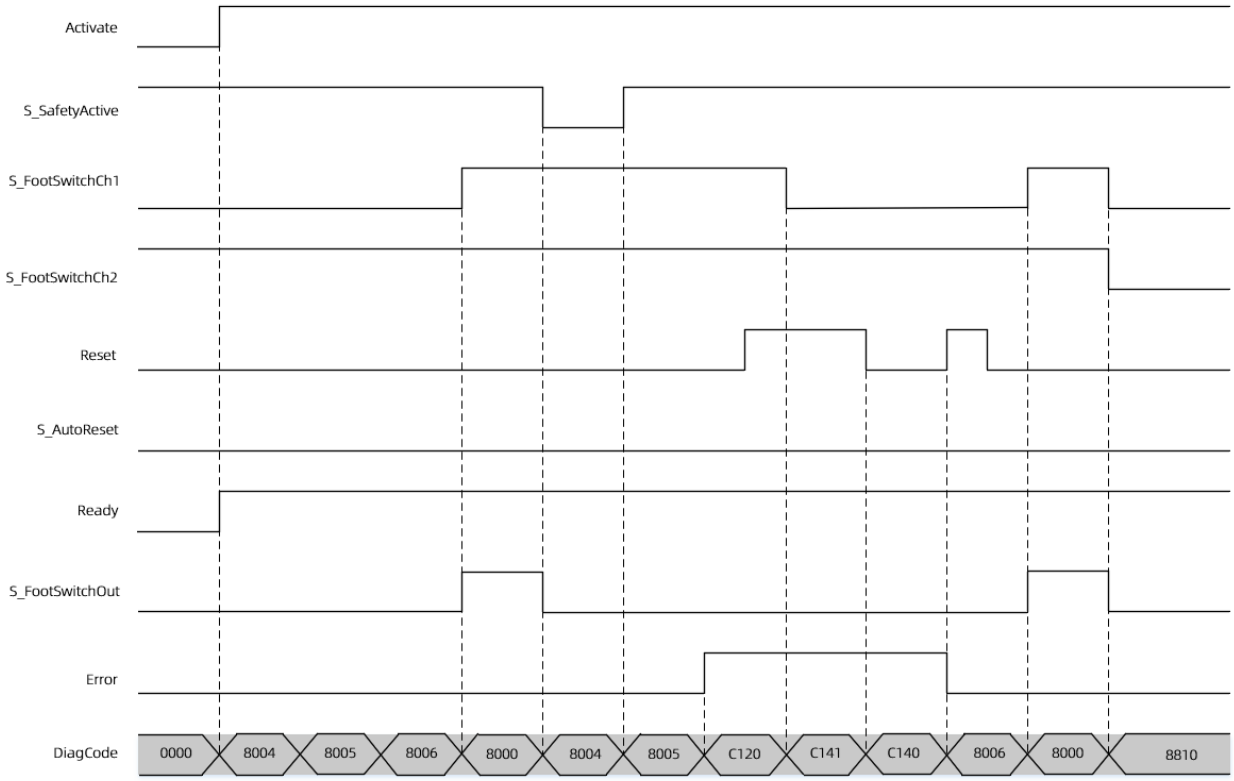


Figure 11-10-2-2-12-2 Status transition diagram for SF_FootSwitch

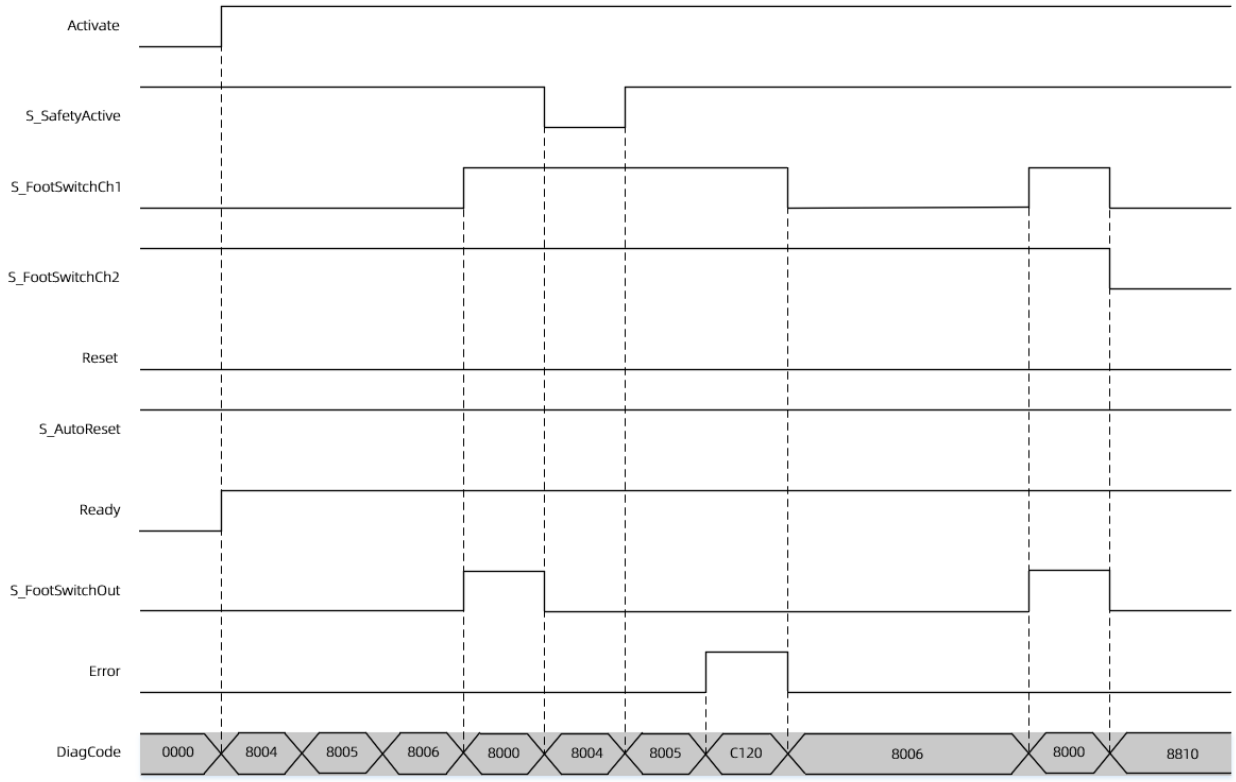
- This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.
- The transition from any status to the idle status when Activate = FALSE is not indicated, but the priority (0) for transitioning to the idle status is the highest.

- **Sequence diagram**

  S_AutoReset = FALSE



S_AutoReset = TRUE



- **Fault detection**

  The faulty status is triggered by the following conditions:

  - Detects that a TRUE value of Reset is triggered at the non-rising edge.
  - The switch position is invalid.

- **Fault behavior**
  - When a fault occurs, the S_FootSwitchOut safety output is set to FALSE and remains in this status. Unlike other function blocks, the faulty status can be cleared by setting Reset to FALSE, or on this basis, by setting S_SafetyActive to FALSE.
  - After the fault is cleared, the valid switch must be set to the initial position defined in the process before the S_FootSwitchOut output can be set to TRUE via the foot switch.
  - If S_AutoReset = FALSE, a rising edge trigger on Reset is required.
- **Function block-specific fault codes and status codes**
  - Fault code

DiagCode	Status Name	Description and Output
C141	Reset Error 1	In the C140 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_FootSwitchOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C181	Reset Error 2	In the C180 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_FootSwitchOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C120	Operation Error 1	The foot switch is not in position 1 when S_SafetyActive is activated. Ready = TRUE S_FootSwitchOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
C140	Operation Error 2	The foot switch is in position 1 after status C120. Ready = TRUE S_FootSwitchOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C160	Operation Error 3	The foot switch is in position 2 after position 3. Ready = TRUE S_FootSwitchOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C180	Operation Error 4	The foot switch is not in position 2 after status C160. Ready = TRUE S_FootSwitchOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

◦ Status code

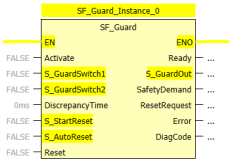
DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_FootSwitchOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8004	Basic Operation Mode	The safety operation mode is not activated. Ready = TRUE S_FootSwitchOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8005	Safe Operation Mode	The safety operation mode is activated. Ready = TRUE S_FootSwitchOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8006	Position 1	The safety operation mode is activated and the foot switch is in position 1. Ready = TRUE S_FootSwitchOut = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8810	Position 3	The safety operation mode is activated and the foot switch is in position 3. Ready = TRUE S_FootSwitchOut = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8000	Position 2	The safety operation mode is activated and the foot switch is in position 2. Ready = TRUE S_FootSwitchOut = TRUE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE

11.10.2.2.13 SF_Guard

SF_Guard monitors safety-related guard devices. It is equipped with two independent input parameters corresponding to the two switches on the guard device, and includes a discrepancy time (DiscrepancyTime) setting for disabling the guard device.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_Guard	Monitoring of safety-related protective devices	FB		

- **Related variables**

  Variables

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.
	S_GuardSwitch1	SAFEBOOL	TRUE, FALSE	FALSE	Protective switch 1 input. FALSE: The guard device is not disabled. TRUE: The guard device is disabled.
	S_GuardSwitch2	SAFEBOOL	TRUE, FALSE	FALSE	Protective switch 2 input. FALSE: The guard device is not disabled. TRUE: The guard device is disabled.
	DiscrepancyTime	TIME	Follows data type	T#0ms	Monitoring synchronization time between S_GuardSwitch1 and S_GuardSwitch2.
	S_StartReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to start reset.
	S_AutoReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to automatically reset.
	Reset	BOOL	TRUE, FALSE	FALSE	Manual reset
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	S_GuardOut	SAFEBOOL	TRUE, FALSE	FALSE	Output indicating that the guard device is disabled and the guarded area is safe. FALSE: The guard device is enabled. TRUE: Both S_GuardSwitch1 and S_GuardSwitch2 are TRUE, with no fault.
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	ResetRequest	BOOL	TRUE, FALSE	FALSE	Flag indicating whether a reset is required.

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**
  - This function block requires two input parameters to monitor the position of a safety guard device with two switches (in compliance with ISO 14119 standard), and includes a DiscrepancyTime input and a Reset input. If the safety guard device has only one switch, the S_GuardSwitch1 and S_GuardSwitch2 inputs can be bridged. Discrepancy time is the maximum time required for both switches to respond when disabling the safety guard device. Reset, S_StartReset, and S_AutoReset inputs are used to determine the reset method of the function block after the safety guard device is enabled.
    - When enabling the guard device: both S_GuardSwitch1 and S_GuardSwitch2 inputs should switch to FALSE. When either switch is set to FALSE, the S_GuardOut output immediately switches to FALSE.
    - When disabling the guard device: both S_GuardSwitch1 and S_GuardSwitch2 inputs should switch to TRUE.
  - This function block monitors the symmetry of the operations of the two switches: if only one contact completes the open/close process, the S_GuardOut output remains FALSE.
  - Relationship between the output behavior and discrepancy time:
    - When the input statuses of S_GuardSwitch1 and S_GuardSwitch2 input statuses are inconsistent, the discrepancy time monitoring is immediately initiated.
    - If the input remains inconsistent after the discrepancy time expires, the S_GuardOut output remains FALSE.
    - If the corresponding second input of S_GuardSwitch1/S_GuardSwitch2 switches to TRUE within the discrepancy time, the S_GuardOut output will be set to TRUE after verification.
  - Inputs S_StartReset and S_AutoReset can only be activated when it is ensured that the activation of the programmable electronic system (PES) will not cause hazardous conditions.
  - Set DiscrepancyTime to a value greater than the safety task cycle.
- **Status transition diagram**

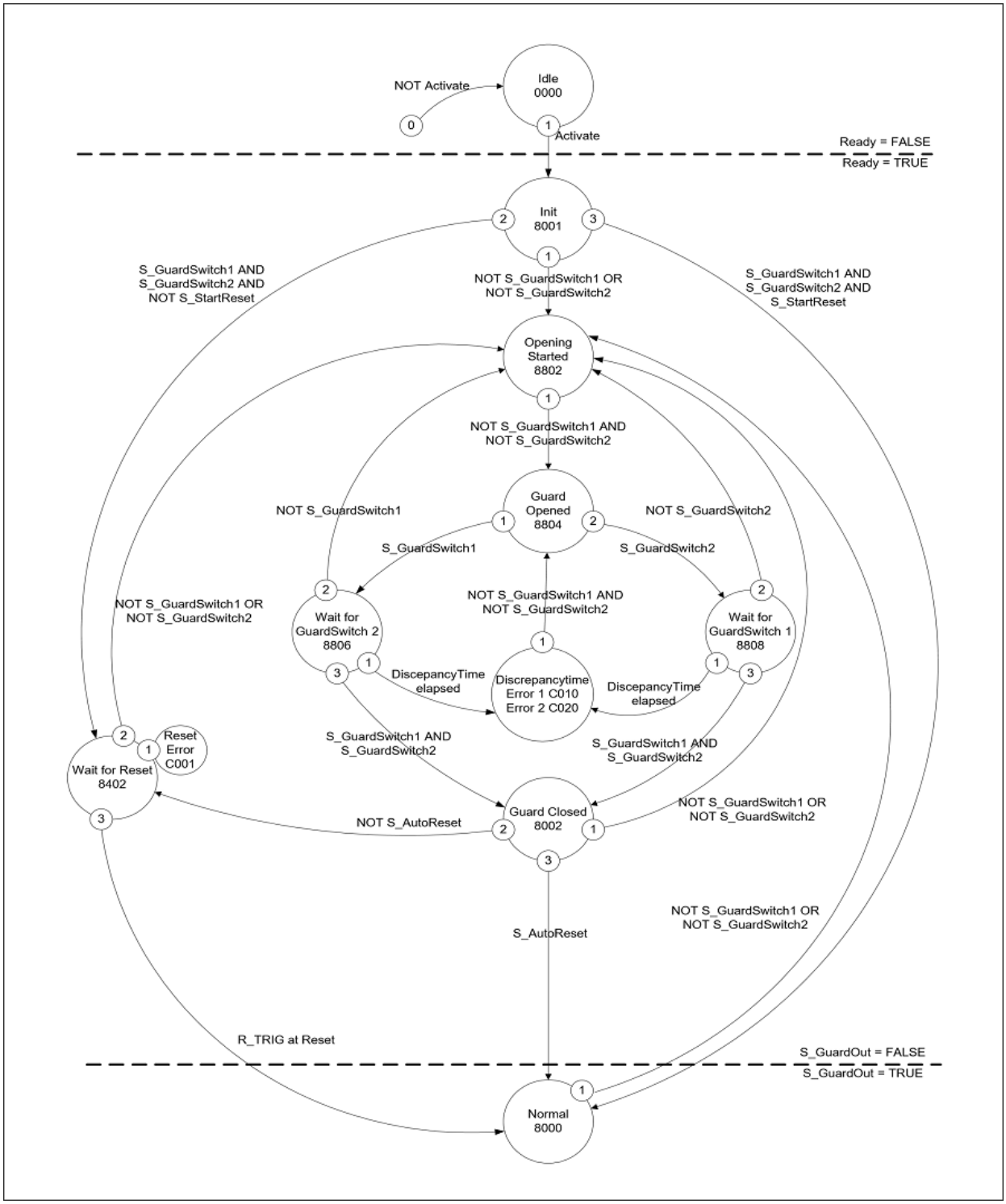
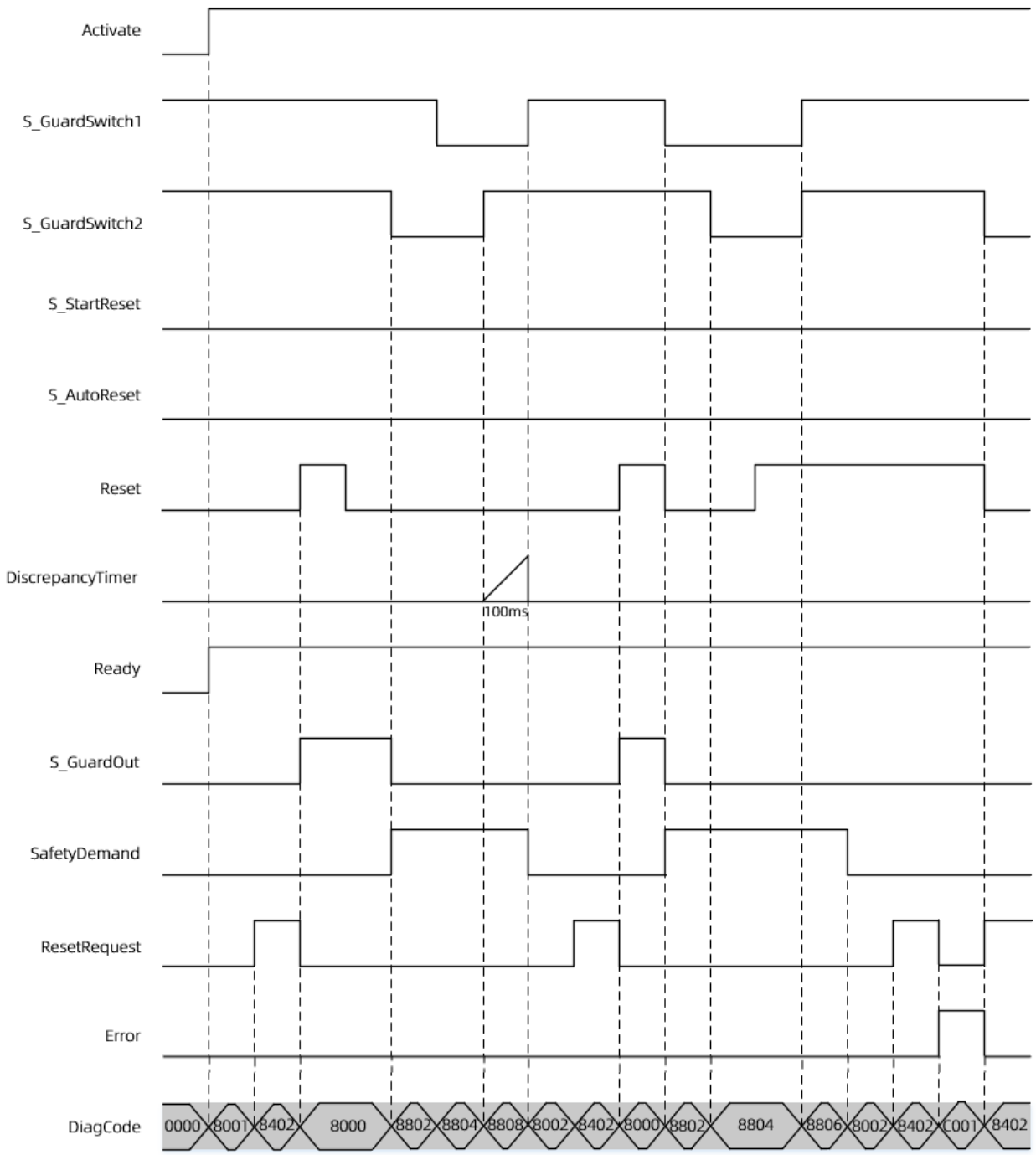
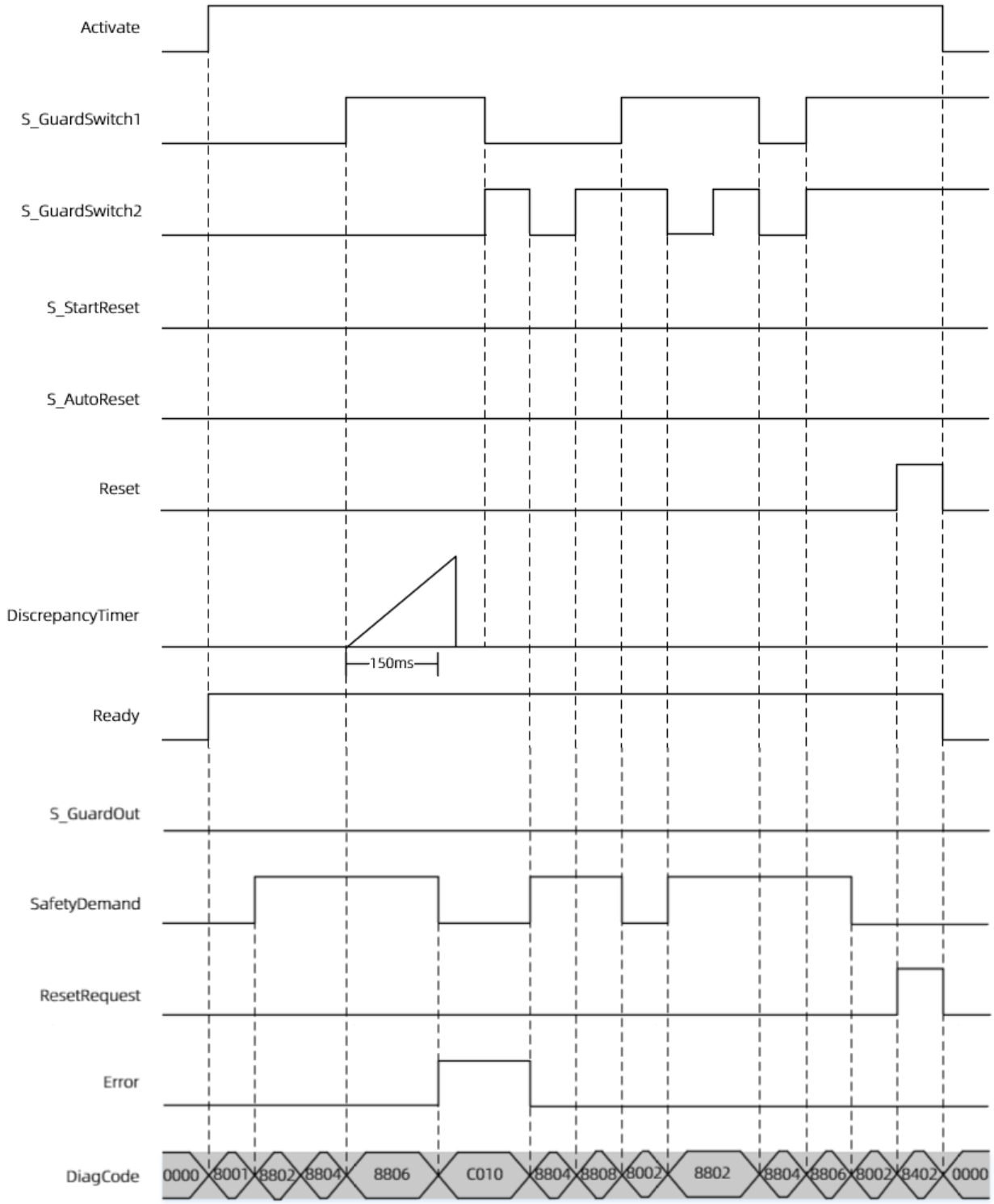


Figure 11-10-2-2-13-1 Status transition diagram for SF_Guard

- This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.
- The transition from any status to the idle status when Activate = FALSE is not indicated, but the priority (0) for transitioning to the idle status is the highest.

- **Sequence diagram**


Activate
S_GuardSwitch1
S_GuardSwitch2
S_StartReset
S_AutoReset
Reset
DiscrepancyTimer
100ms
Ready
S_GuardOut
SafetyDemand
ResetRequest
Error
DiagCode
0000
8001
8402
8000
8802
8804
8808
8002
8402
8000
8802
8804
8806
8002
8402
C001
8402



- **Fault detection**
  - External signal characteristics: SAFEBOOL input possess inherent fault detection capability.
  - Mechanical configuration: Apply the mechanical combination of opening and closing switches according to EN 954 standard (safety guard devices with dual switches).
  - Discrepancy time monitoring: According to EN 954 standard, the time lag between the responses of two mechanical switches is monitored (considered as "application fault" detection, i.e., generated by the application). If the response time lag between the first S_GuardSwitch1/S_GuardSwitch2 input and the second input exceeds the value set by DiscrepancyTime, a fault is detected and the Error output is set to TRUE.
  - Static signal detection: The function block can detect a static TRUE signal at the RESET input.

- **Fault behavior**

  When a fault occurs, the S_GuardOut output is set to FALSE. If both S_GuardSwitch1 and S_GuardSwitch2 inputs are bridged, no fault will be detected.

  To exit the faulty status, Reset must be set to FALSE.

  To clear the discrepancy time fault, both S_GuardSwitch1 and S_GuardSwitch2 inputs must be set to FALSE.

- **Function block-specific fault codes and status codes**

  ◦ Fault code

DiagCode	Status Name	Description and Output
C001	Reset Error	In the 8402 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_GuardOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C010	Discrepancytime Error 1	In the 8806 status, DiscrepancyTime is timeout. Ready = TRUE S_GuardOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C020	Discrepancytime Error 2	In the 8808 status, DiscrepancyTime is timeout. Ready = TRUE S_GuardOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

  ◦ Status code

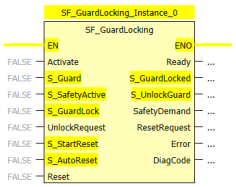
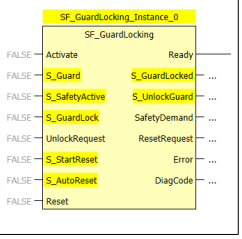
DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_GuardOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8000	Normal	The safety guard device is disabled and the safety status is confirmed. Ready = TRUE S_GuardOut = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8001	Init (Initialization)	Function block is activated. Ready = TRUE S_GuardOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8802	Opening Started	A complete switching sequence must be executed. Ready = TRUE S_GuardOut = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8402	Wait for Reset	Wait for the rising-edge signal of Reset. Ready = TRUE S_GuardOut = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = FALSE

DiagCode	Status Name	Description and Output
8804	Guard Opened	The safety guard device is fully enabled. Ready = TRUE S_GuardOut = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8806	Wait for GuardSwitch2	Wait for S_GuardSwitch2 to become TRUE; S_GuardSwitch1 is already TRUE; the discrepancy timer has started. Ready = TRUE S_GuardOut = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8808	Wait for GuardSwitch1	Wait for S_GuardSwitch2 to become TRUE; S_GuardSwitch1 is already TRUE; the discrepancy timer has started. Ready = TRUE S_GuardOut = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8002	Guard Closed	The guard device has been disabled. If S_AutoReset = FALSE, wait for the Reset signal. Ready = TRUE S_GuardOut = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

11.10.2.2.14 SF_GuardLocking

SF_GuardLocking controls the access to the hazardous area using an interlocked guard device with a lock.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_GuardLocking	Safety door lock with interlock device	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.
	S_Guard	SAFEBOOL	TRUE, FALSE	FALSE	Monitors the interlock of the guard device. FALSE: The guard device is enabled. TRUE: The guard device is disabled and the guarded area is safe.
	S_SafetyActive	SAFEBOOL	TRUE, FALSE	FALSE	Status of hazardous area (EDM), for example, based on speed monitoring or safety time delay. FALSE: Machine is in an unsafe status. TRUE: Machine is in a safety status.
	S_GuardLock	SAFEBOOL	TRUE, FALSE	FALSE	Status of the guard device lock. FALSE: The guard device is not locked. TRUE: The guard device is locked.
	UnlockRequest	BOOL	Follows data type	T#0ms	Operator intervention to request to unlock the guard device. FALSE: No request. TRUE: Unlock requested.
	S_StartReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to start reset.
	S_AutoReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to automatically reset.
	Reset	BOOL	TRUE, FALSE	FALSE	Manual reset
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	S_GuardLocked	SAFEBOOL	TRUE, FALSE	FALSE	Output for safety-related response. FALSE: Non-safety status. TRUE: Safety status.

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
	S_UnlockGuard	SAFEBOOL	TRUE, FALSE	FALSE	Used to unlock the guard device. FALSE: Disable the guard device. TRUE: Unlock the guard device.
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	ResetRequest	BOOL	TRUE, FALSE	FALSE	Flag indicating whether a reset is required.
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**
  - This function block controls the safety lock and monitors the guard device and lock position. It can also be used with mechanical lock switches.
  - When an operator requests access to a hazardous area, the safety device can only be unlocked when the hazardous area is in a safety status. The locking operation can be performed when the guard device is disabled. The machine can be started only when the guard device is disabled and locked. In safety-critical situations, a guard device that is not disabled or not locked will be detected.
  - Inputs S_StartReset and S_AutoReset can only be activated when it is ensured that the activation of the programmable electronic system (PES) will not cause hazardous conditions.

    Operation sequence:

1	Surface cleanness	Request to restore the hazardous area to a safety status – not performed within this function block.
2	Input	Feedback from the corresponding hazardous area in a safety status (via S_SafetyActive).
3	Input	Operator requests unlocking the protective device (via UnlockRequest).
4	Output	The protective device can be unlocked (via S_UnlockGuard).
5	Input	The protective device is unlocked (via S_GuardLock). The protective device can now be enabled (S_GuardLocked = FALSE).
6	Input	Monitor the guard device status via S_Guard signal when the guard device is disabled again.
7	Input	Restart the hazardous area (via Reset) according to the operator's feedback.
8	Output	Lock the protection (S_UnlockGuard).
9	Input	Check if protection is locked (S_GuardLock).
10	Output	Restore the hazardous area to operational status (S_GuardLocked = TRUE).

11	Surface cleanness	Restore operation in hazardous area.

- **Status transition diagram**

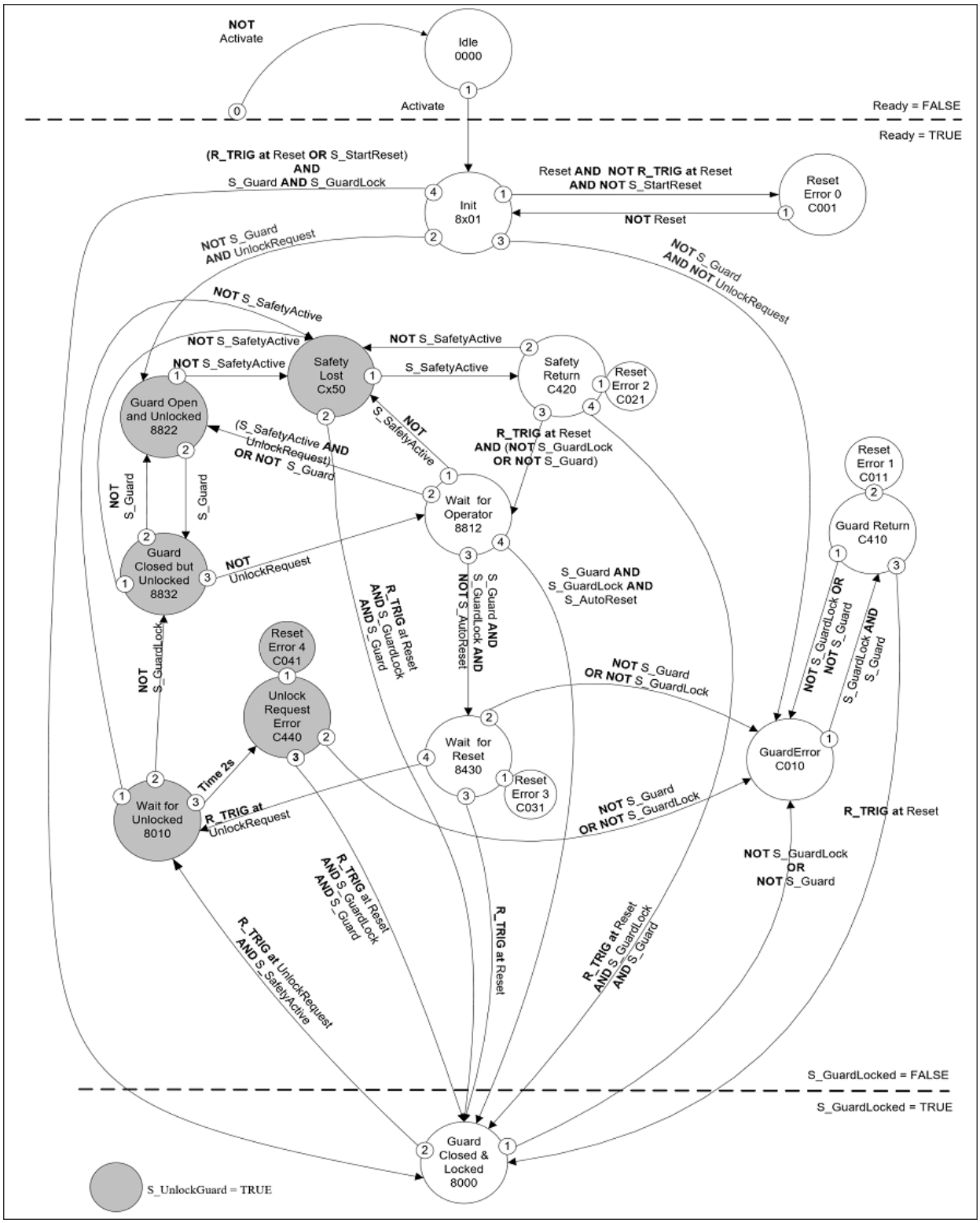
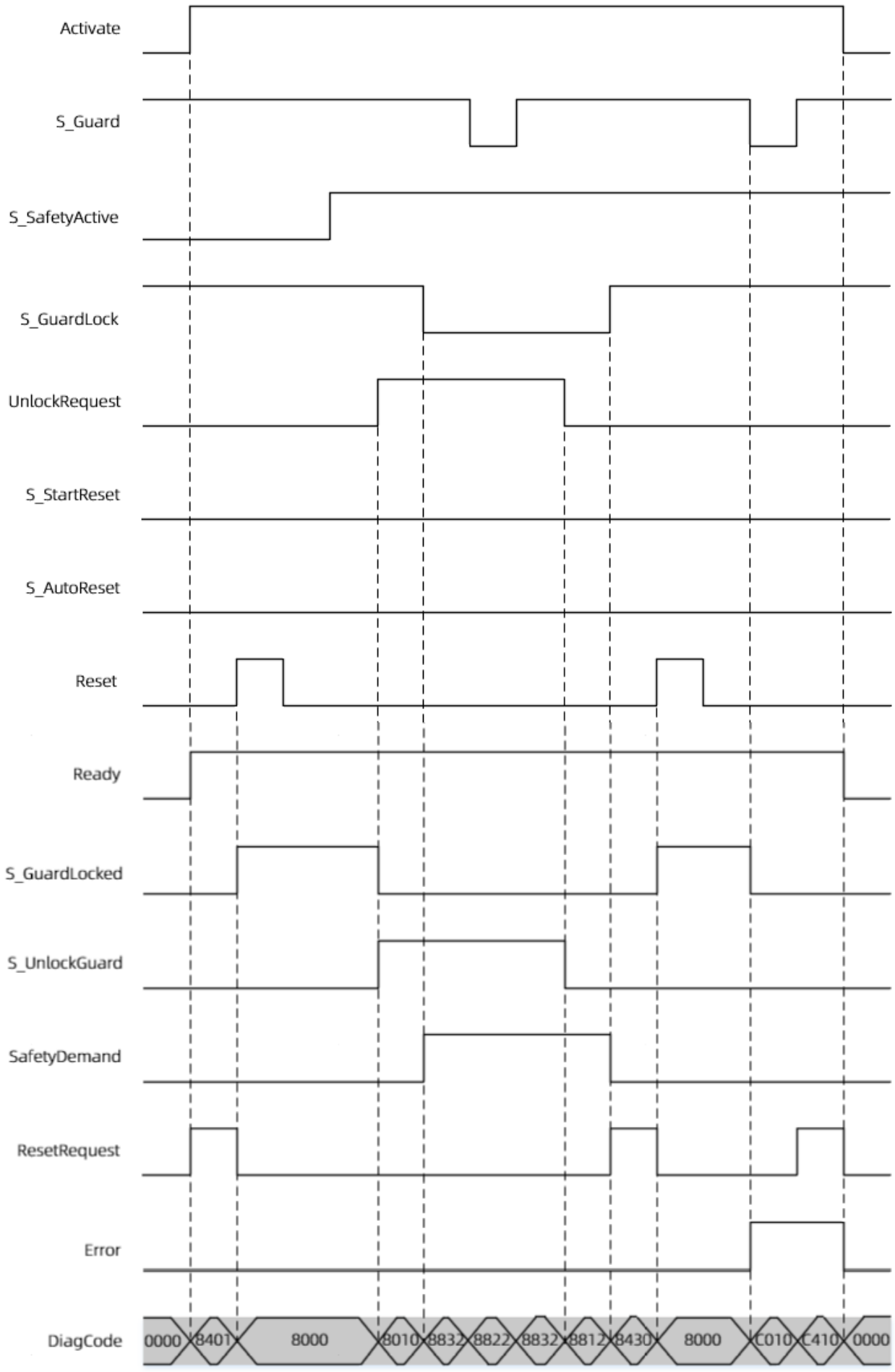


Figure 11-10-2-2-14-1 Status transition diagram for SF_GuardLocking

  - This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.
  - The transition from any status to the idle status when Activate = FALSE is not indicated, but the priority (0) for transitioning to the idle status is the highest.

- **Sequence diagram**



- **Fault detection**
  Detects whether a static signal is present on the Reset input. Detects faults at the protective device switch.

- **Fault behavior**
  - When a fault occurs, both S_GuardLocked and S_UnlockGuard outputs are set to FALSE, the DiagCode output displays the relevant fault code, and the Error output is set to TRUE.
  - The fault must be reset by a rising edge trigger of the Reset input.
- **Function block-specific fault codes and status codes**
  - Fault code

DiagCode	Status Name	Description and Output
C001	Reset Error 0	In the 8x01 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C010	Guard Error	S_GuardLock and S_Guard are not in TRUE status despite no door open command being issued. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C011	Reset Error 1	In the C410 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
C410	Guard Return	S_GuardLock and S_Guard are reactivated to TRUE after signal loss (fault code C010). Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = TRUE
Cx50	Safety Lost	Safety confirmation signal is lost. If both S_Guard and S_GuardLock are TRUE, the fault code is C450; otherwise, it is C050. • When the fault code is C450: Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = TRUE SafetyDemand = FALSE ResetRequest = NOT Reset Error = TRUE • When the fault code is C050: Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C021	Reset Error 2	In the C420 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
C420	Safety Return	The safety confirmation signal is reactivated after being lost (Cx50). Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = TRUE
C031	Reset Error 3	In the 8430 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C440	Unlock Request Error	The time waiting for unlock has timed out. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = TRUE SafetyDemand = FALSE ResetRequest = NOT Reset Error = TRUE
C041	Reset Error 4	In the C440 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

◦ Status code

DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_GuardLocked = FALSE S_UnlockGuard = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8000	Guard Closed and Locked	The guard device is disabled and locked. Ready = TRUE S_GuardLocked = TRUE S_UnlockGuard = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8x01	Init (Initialization)	The function block is activated and initialized. If both S_Guard and S_GuardLock are TRUE, the status code is 8401; otherwise, it is 8801. • When the status code is 8401: Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = FALSE • When the status code is 8801: Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE

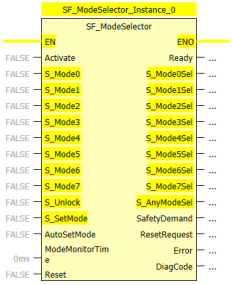
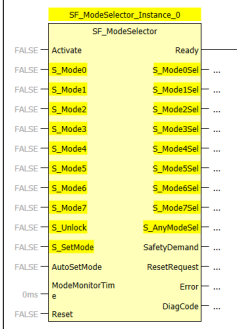
DiagCode	Status Name	Description and Output
8430	Wait for Reset	The door is closed and locked, waiting for the operator to reset. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = FALSE
8812	Wait for Operator	Waiting for the operator to request opening the door (unlock request). Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8822	Guard Open and Unlocked	The guard device is enabled and locked. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = TRUE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8832	Guard Closed but Unlocked	The guard device is disabled and not locked. Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = TRUE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8010	Wait for Unlocked	S_UnlockGuard is TRUE, but the S_GuardLock input signal is still TRUE (waiting for S_GuardLock signal to become FALSE). Ready = TRUE S_GuardLocked = FALSE S_UnlockGuard = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

11.10.2.2.15 SF_ModeSelector

SF_ModeSelector selects the system's operating mode (automatic, manual, semi-automatic, etc.).

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_ModeSelector	Mode selector	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.
	S_Mode0	SAFEBOOL	TRUE, FALSE	FALSE	Input 0 of the mode selector switch. FALSE: Mode 0 is not requested. TRUE: Mode 0 is requested.
	S_Mode1	SAFEBOOL	TRUE, FALSE	FALSE	Input 1 of the mode selector switch. FALSE: Mode 1 is not requested. TRUE: Mode 1 is requested.

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
	S_Mode2	SAFEBOOL	TRUE, FALSE	FALSE	Input 2 of mode selector switch. FALSE: Mode 2 is not requested. TRUE: Mode 2 is requested.
	S_Mode3	SAFEBOOL	TRUE, FALSE	FALSE	Input 3 of mode selector switch. FALSE: Mode 3 is not requested. TRUE: Mode 3 is requested.
	S_Mode4	SAFEBOOL	TRUE, FALSE	FALSE	Input 4 of mode selector switch. FALSE: Mode 4 is not requested. TRUE: Mode 4 is requested.
	S_Mode5	SAFEBOOL	TRUE, FALSE	FALSE	Input 5 of mode selector switch. FALSE: Mode 5 is not requested. TRUE: Mode 5 is requested.
	S_Mode6	SAFEBOOL	TRUE, FALSE	FALSE	Input 6 from mode selector switch. FALSE: Mode 6 is not requested. TRUE: Mode 6 is requested.
	S_Mode7	SAFEBOOL	TRUE, FALSE	FALSE	Input 7 from mode selector switch. FALSE: Mode 7 is not requested. TRUE: Mode 7 is requested.
	S_Unlock	SAFEBOOL	TRUE, FALSE	FALSE	Locks/unlocks the selected mode. FALSE: The actual "S_ModeXSel" output is locked, so even if the "S_ModeX" input changes due to a rising edge of "S_SetMode", the "S_ModeXSel" output will not change. TRUE: The selected "S_ModeXSel" output is not locked, and can be changed.

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
	S_SetMode	SAFEBOOL	TRUE, FALSE	FALSE	When "AutoSetMode" is FALSE, S_SetMode changes the selected mode. TRUE: Select the mode by rising edge setting. FALSE: When "AutoSetMode" is TRUE, S_SetMode is set to FALSE. When the operator toggles the mode selector switch, "S_AnyModeSel" and "S_ModeXSel" become FALSE. Triggered by the rising edge of "S_SetMode", "S_ModeXSel" becomes TRUE, and the mode is changed accordingly.
	AutoSetMode	BOOL	TRUE, FALSE	FALSE	Setting for mode confirmation. FALSE: The mode can only be changed by the operator through "S_SetMode". TRUE: The operator does not need to confirm the change through "S_SetMode"; the "S_ModeXSel" output automatically changes according to the "S_ModeX" input. ("AutoSetMode" is equivalent to FALSE when the mode is locked using "S_Unlock")
	ModeMonitorTime	TIME	Follows data type	T#0ms	Maximum allowable time for changing the mode.
	Reset	BOOL	TRUE, FALSE	FALSE	Manual reset
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	S_Mode0Sel	SAFEBOOL	TRUE, FALSE	FALSE	Indicates mode 0 is selected and confirmed. FALSE: Mode 0 is not selected, or is not confirmed after selection. TRUE: Mode 0 is selected and confirmed.

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
	S_Mode1Sel	SAFEBOOL	TRUE, FALSE	FALSE	Indicates mode 1 is selected and confirmed. FALSE: Mode 1 is not selected, or is not confirmed after selection. TRUE: Mode 1 is selected and confirmed.
	S_Mode2Sel	SAFEBOOL	TRUE, FALSE	FALSE	Indicates mode 2 is selected and confirmed. FALSE: Mode 2 is not selected, or is not confirmed after selection. TRUE: Mode 2 is selected and confirmed.
	S_Mode3Sel	SAFEBOOL	TRUE, FALSE	FALSE	Indicates mode 3 is selected and confirmed. FALSE: Mode 3 is not selected, or is not confirmed after selection. TRUE: Mode 3 is selected and confirmed.
	S_Mode4Sel	SAFEBOOL	TRUE, FALSE	FALSE	Indicates mode 4 is selected and confirmed. FALSE: Mode 4 is not selected, or is not confirmed after selection. TRUE: Mode 4 is selected and confirmed.
	S_Mode5Sel	SAFEBOOL	TRUE, FALSE	FALSE	Indicates mode 5 is selected and confirmed. FALSE: Mode 5 is not selected, or is not confirmed after selection. TRUE: Mode 5 is selected and confirmed.
	S_Mode6Sel	SAFEBOOL	TRUE, FALSE	FALSE	Indicates mode 6 is selected and confirmed. FALSE: Mode 6 is not selected, or is not confirmed after selection. TRUE: Mode 6 is selected and confirmed.

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
	S_Mode7Sel	SAFEBOOL	TRUE, FALSE	FALSE	Indicates mode 7 is selected and confirmed. FALSE: Mode 7 is not selected, or is not confirmed after selection. TRUE: Mode 7 is selected and confirmed.
	S_AnyModeSel	SAFEBOOL	TRUE, FALSE	FALSE	Indicates one of the eight modes is selected and confirmed. FALSE: No mode is selected. TRUE: One of the eight modes is selected and confirmed.
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	ResetRequest	BOOL	TRUE, FALSE	FALSE	Flag indicating whether a reset is required.
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**
  - Default status and safety status:
    The default status after the function block is activated is the mode changed status (ModeChanged), at which point all S_ModeXSel and S_AnyModeSel outputs are FALSE.
  - Mode changing logic:

    When the function block is in the ModeChanged status:

    - If AutoSetMode = FALSE: A rising edge on the S_SetMode input is required to confirm the new S_ModeX input, thereby generating the new S_ModeXSel output.
    - If AutoSetMode = TRUE: The new S_ModeX input directly generates the new S_ModeXSel output.

      Status transition condition: A transition from status 8802 to 8000 is only possible when at least one S_ModeX input is TRUE; if all S_ModeX inputs are FALSE, the function block remains in status 8802 even if S_SetMode is triggered.
  - Status transition monitoring:

    The transition from the ModeChanged status to ModeSelected (i.e., S_SetMode is triggered) is not monitored by a timer.

    If the function block is in the ModeSelected status: A transition back to the ModeChanged status occurs if a new S_ModeX input (with high priority) and a NOT S_Unlock signal (with low priority) occur simultaneously.

◦ Parameter configuration requirements:

S_ModeX inputs not used for mode selection shall be set to the default value FALSE to simplify program verification.

The AutoSetMode input can only be activated when it is ensured that the activation of the programmable electronic system (PES) will not cause hazardous conditions.

- **Status transition diagram**

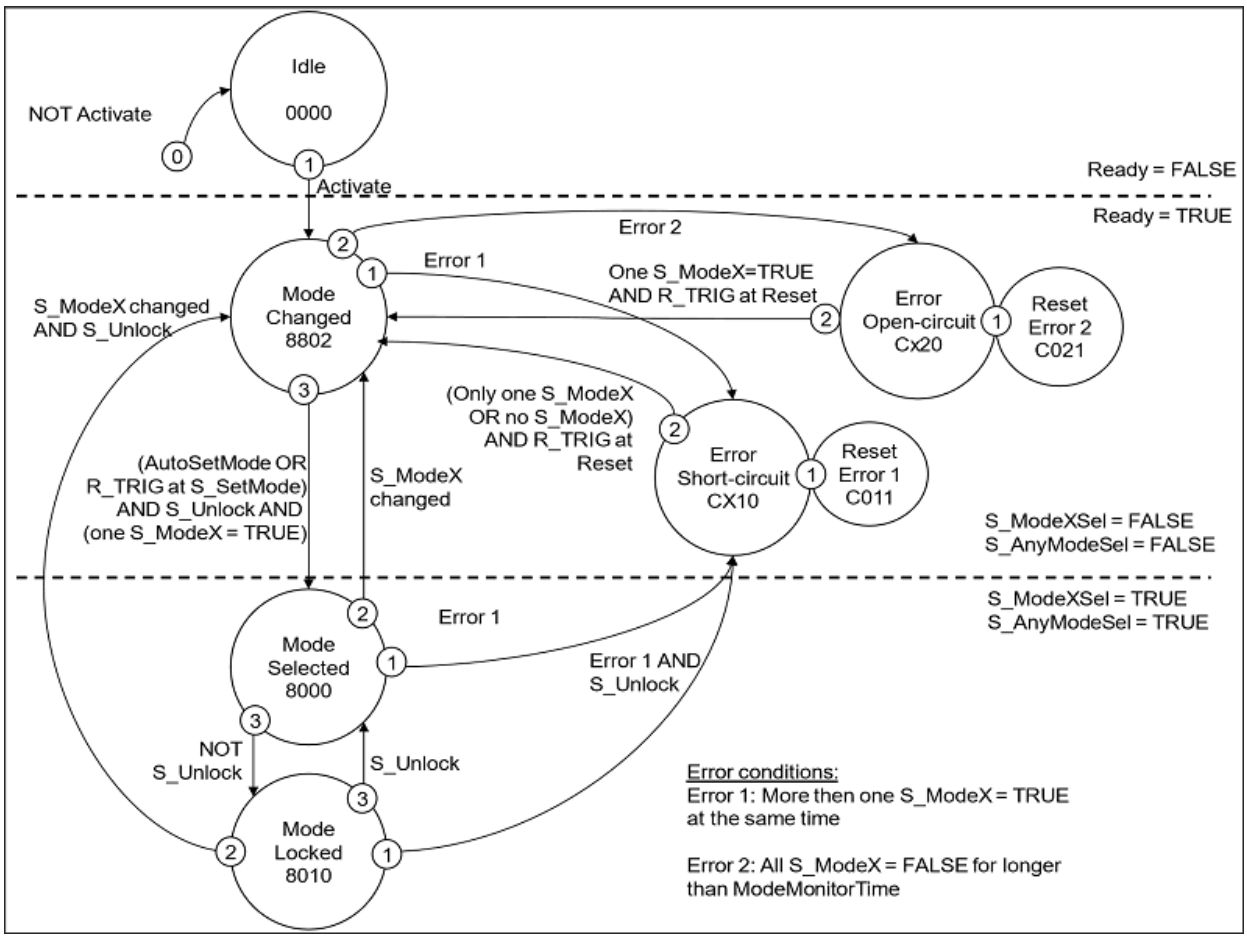
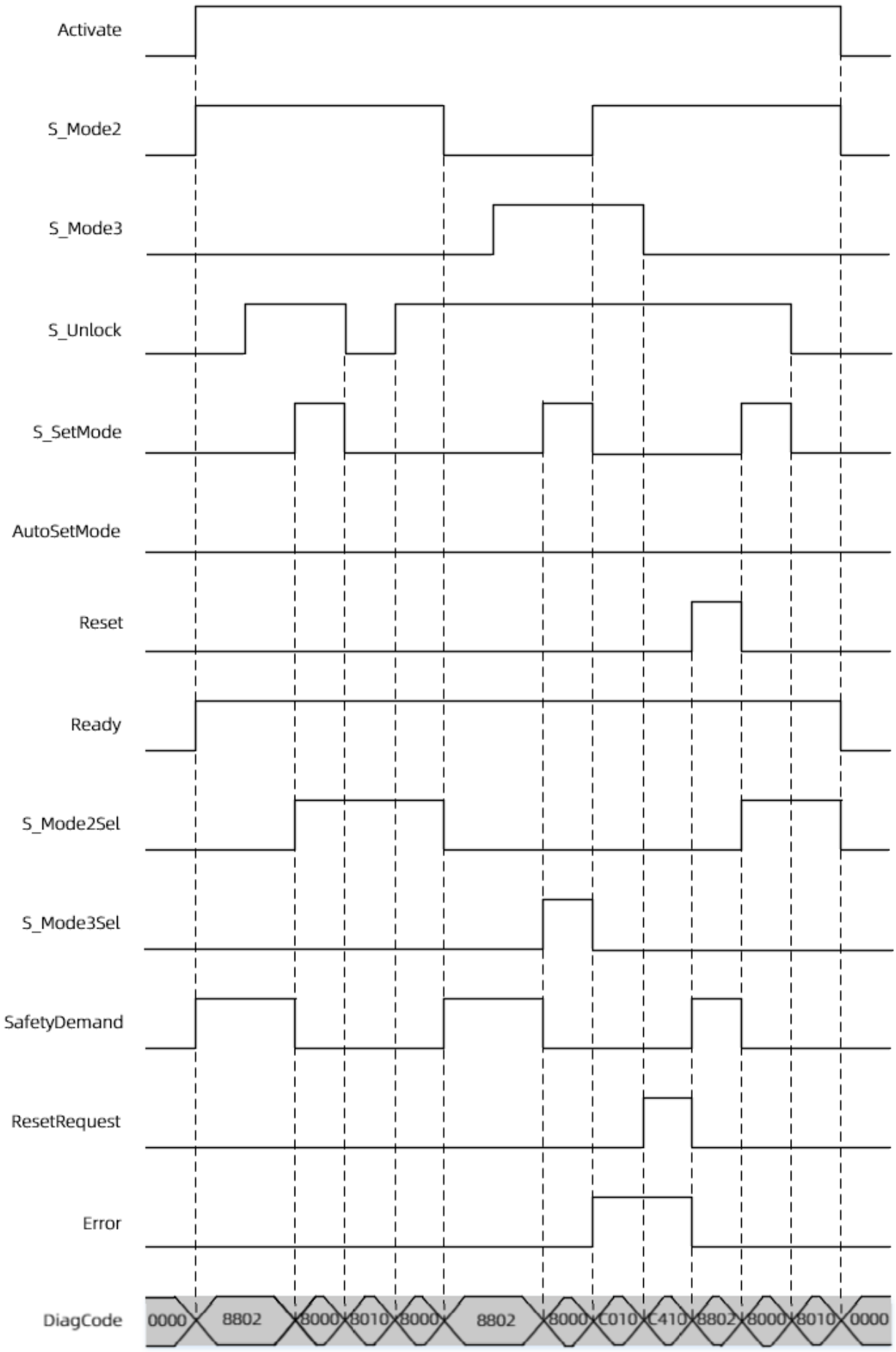


Figure 11-10-2-2-15-1 Status transition diagram for SF_ModeSelector

◦ This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.

◦ The transition from any status to the idle status when Activate = FALSE is not indicated, but the priority (0) for transitioning to the idle status is the highest.

- **Sequence diagram**



- **Fault detection**

Detects whether no mode input is selected. If no mode is selected after the ModeMonitorTime timeout, the status is invalid. Each time an S_ModeX input is triggered at the falling edge, ModeMonitorTime is reset, and the function block transitions to the ModeChanged status after activation.

The function block also directly detects whether multiple S_ModeX inputs are selected simultaneously.

When the function block is in faulty status C011 or C021, a static reset condition is detected.

- **Fault behavior**

  Upon fault occurrence, the S_ModeXSel and S_AnyModeSel outputs are set to the safety status FALSE. The DiagCode output indicates the relevant fault code, and the Error output is set to TRUE. The fault must be reset by a rising edge trigger of the Reset input. The function block transitions from the faulty status to the ModeChanged status.

- **Function block-specific fault codes and status codes**

  ◦ Fault code

DiagCode	Status Name	Description and Output
Cx10	Error Short-circuit	Detects that two or more S_ModeX inputs are TRUE due to cable short circuit or other causes. If any S_ModeX (X = 0 to 7) is TRUE or all S_ModeX inputs are FALSE, the fault code is C410; otherwise, it is C010. • When the fault code is C410: Ready = TRUE SafetyDemand = FALSE ResetRequest = NOT Reset Error = TRUE S_AnyModeSel = FALSE All S_ModeXSel = FALSE • When the fault code is C010: Ready = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE S_AnyModeSel = FALSE All S_ModeXSel = FALSE

DiagCode	Status Name	Description and Output
Cx20	Error Open-circuit	Detects that all S_ModeX inputs are FALSE: The time period after S_ModeX is triggered has exceeded ModeMonitorTime due to cable open circuit or other causes. If any S_ModeX is TRUE, the fault code is C420; otherwise, it is C020. • When fault code is C420: Ready = TRUE SafetyDemand = FALSE ResetRequest = NOT Reset Error = TRUE S_AnyModeSel = FALSE All S_ModeXSel = FALSE • When the fault code is C020: Ready = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE S_AnyModeSel = FALSE All S_ModeXSel = FALSE
C011	Reset Error 1	A static reset signal was detected in status C410. Ready = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE S_AnyModeSel = FALSE All S_ModeXSel = FALSE
C021	Reset Error 2	A static reset signal was detected in status C420. Ready = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE S_AnyModeSel = FALSE All S_ModeXSel = FALSE

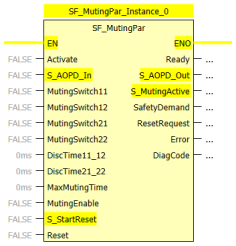
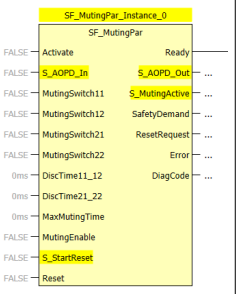
◦ Status code

DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE S_AnyModeSel = FALSE All S_ModeXSel = FALSE
8802	ModeChanged	After function block activation, the mode selected by S_ModeX (unless locked) is reset or the faulty status is reset. Ready = TRUE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE S_AnyModeSel = FALSE All S_ModeXSel = FALSE
8000	ModeSelected	A valid mode is selected, but not yet locked. Ready = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE S_AnyModeSel = TRUE S_ModeXSel = TRUE (X is the selected S_ModeX, the inputs of other unselected modes are FALSE)
8010	ModeLocked	A valid mode is selected and locked. Ready = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE S_AnyModeSel = TRUE S_ModeXSel = TRUE (X is the selected S_ModeX, the inputs of other unselected modes are FALSE)

11.10.2.2.16 SF_MutingPar

SF_MutingPar intentionally inhibits the safety function. In this function block, parallel muting with four muting sensors is specified.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_MutingPar	Perform parallel muting with four muting sensors.	FB		

- **Related variables**

  Variables

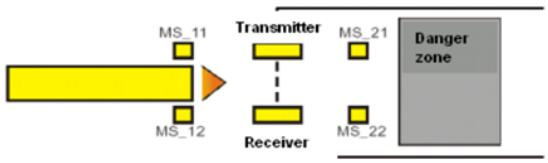
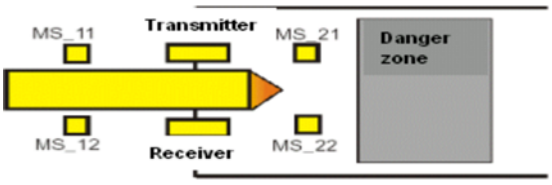
Input / Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.
	S_AOPD_In	SAFEBOOL	TRUE, FALSE	FALSE	OSSD signal from AOPD. FALSE: The guarded area is invaded. TRUE: The guarded area is not invaded.
	MutingSwitch11	BOOL	TRUE, FALSE	FALSE	Status of muting sensor 11. FALSE: Muting sensor 11 is not activated. TRUE: Muting sensor 11 is activated.
	MutingSwitch12	BOOL	TRUE, FALSE	FALSE	Status of muting sensor 12. FALSE: Muting sensor 12 is not activated. TRUE: Muting sensor 12 is activated.
	MutingSwitch21	BOOL	TRUE, FALSE	FALSE	Status of muting sensor 21. FALSE: Muting sensor 21 is not activated. TRUE: Muting sensor 21 is activated.

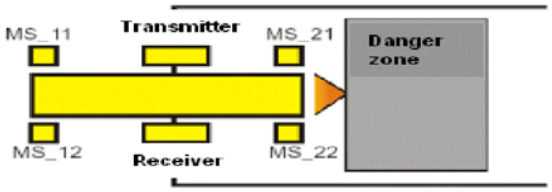
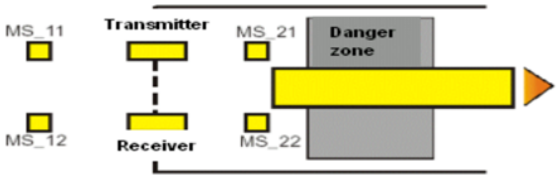
Input / Output	Variable	Data Type	Valid Range	Default Value	Description
	MutingSwitch22	BOOL	TRUE, FALSE	FALSE	Status of muting sensor 22. FALSE: Muting sensor 22 is not activated. TRUE: Muting sensor 22 is activated.
	DiscTime11_12	TIME	Follows data type	T#0ms	Maximum discrepancy time between MutingSwitch11 and MutingSwitch12, with a range of 0 to 4s.
	DiscTime21_22	TIME	Follows data type	T#0ms	Maximum discrepancy time between MutingSwitch21 and MutingSwitch22, with a range of 0 to 4s.
	MaxMutingTime	TIME	Follows data type	T#0ms	Maximum time for the complete muting sequence, with a range of 0 to 120 minutes.
	MutingEnable	BOOL	TRUE, FALSE	FALSE	Control system command used to activate the muting function when required. FALSE: Muting is invalid. TRUE: Muting is valid.
	S_StartReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to start reset.
	Reset	BOOL	TRUE, FALSE	FALSE	Manual reset
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	S_AOPD_Out	SAFEBOOL	TRUE, FALSE	FALSE	Safety-related output indicating the status of the muting guard device. FALSE: The AOPD guarded area is invaded and muting is invalid. TRUE: The AOPD guarded area is not invaded and muting is valid.
	S_MutingActive	SAFEBOOL	TRUE, FALSE	FALSE	Indicates the status of the muting process. FALSE: Muting is invalid. TRUE: Muting is valid.
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	ResetRequest	BOOL	TRUE, FALSE	FALSE	Flag indicating whether a reset is required.

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**
  - Muting refers to the intentional inhibition of the safety function. For example, SF_MutingPar can be used when feeding material into a hazardous area without causing the machine to stop. Muting is triggered by muting sensors and requires two or four muting sensors with proper integration into the production process to ensure no personnel enters the hazardous area during the muting period. Muting sensors may include non-safety proximity switches, photoelectric barriers, limit switches, and so on. The activation of muting must be indicated by an indicator.
  - The muting process is divided into sequential muting and parallel muting. This function block adopts parallel muting with four muting sensors and supports bidirectional use in both forward and backward directions. To prevent misoperation, muting shall be activated by the MutingEnable signal from the process.
  - Input parameters of this function block include:
    - Four muting sensor signals (MutingSwitch11...MutingSwitch22).
    - OSSD signal from the active photoelectric guard device (S_AOPD_In).
    - Three parameterizable times (DiscTime11_12, DiscTime21_22, and MaxMutingTime).
  - The S_StartReset input can only be activated when it is ensured that the activation of the programmable electronic system (PES) will not cause hazardous conditions.

Table11-10-2-2-16-1 Example of forward muting process with four muting sensors

No.	Figure	Explanation
1		During the time set by DiscTime11_12, when MutingSwitch11 (MS_11) and MutingSwitch12 (MS_12) are turned ON due to the workpiece passage, the muting mode becomes active (S_MutingActive = TRUE).
2		The muting mode remains active while MutingSwitch11 (MS_11) and MutingSwitch12 (MS_12) are turned ON due to the workpiece. The workpiece can pass through the light curtain without stopping the machine.

No.	Figure	Explanation
3		Before MutingSwitch11 (MS_11) and MutingSwitch12 (MS_12) are turned OFF, MutingSwitch21 (MS_21) and MutingSwitch22 (MS_22) must be turned ON. This ensures that the muting mode remains ON. The time discrepancy between MutingSwitch21 and MutingSwitch22 is monitored using the time set by DiscTime21_22.
4		The muting mode ends only when MutingSwitch22 (MS_22) is turned ON. The muting mode ends when either MutingSwitch21 (MS_21) or MutingSwitch22 (MS_22) is turned OFF due to the workpiece. The maximum time for which the muting mode remains active is set by MaxMutingTime.

- **Status transition diagram**

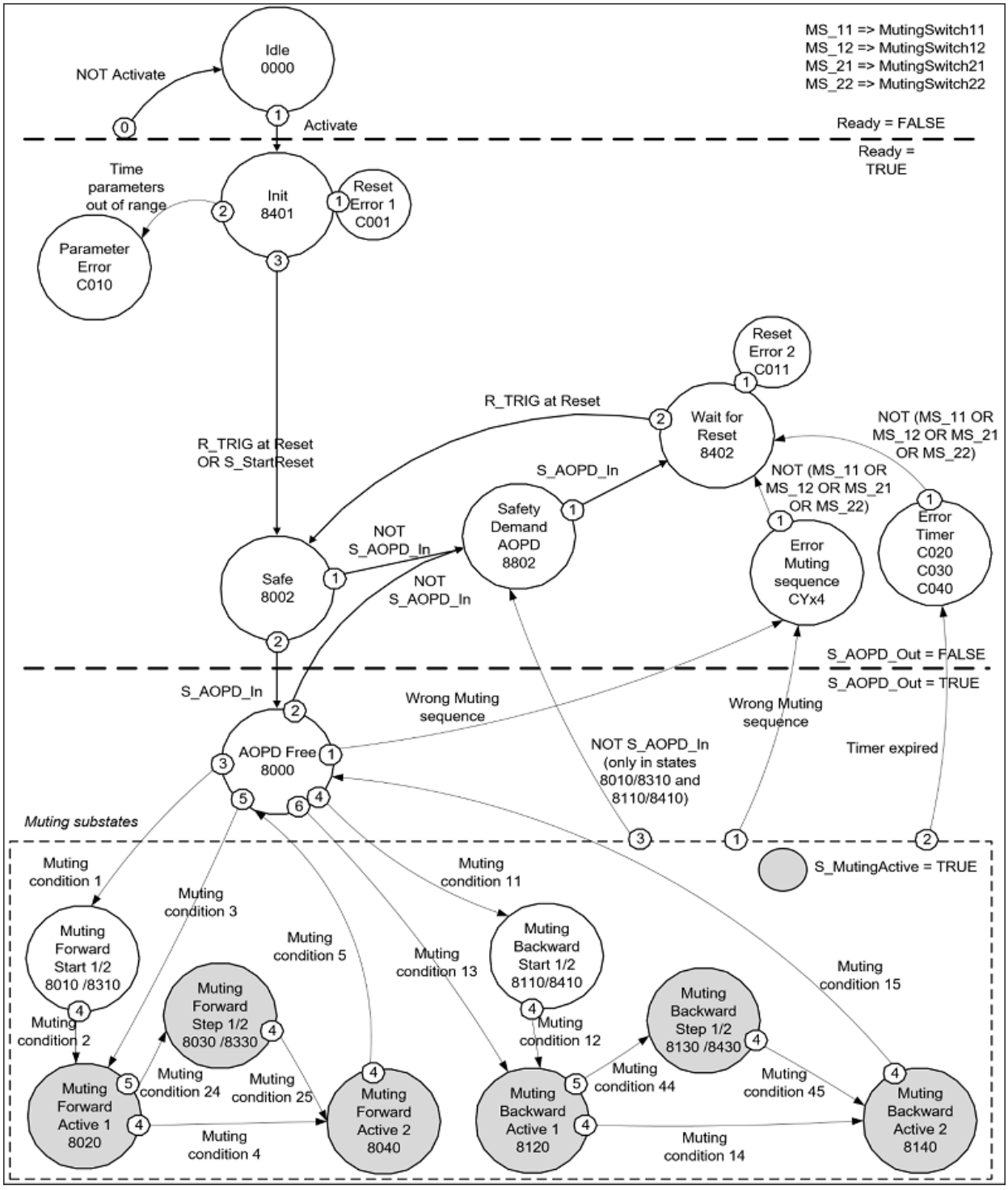


Figure 11-10-2-2-16-1 State transition diagram for SF_MutingPar

- This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.
- The transition from any status to the idle status with Activate being FALSE is not shown. However, these transitions have the highest priority (0).
- In the muting status, AOPD transitions triggered by an incorrect muting sequence (priority 1), a faulty timer (priority 2), and a safety request (priority 3) have higher priority than transitions to the muting status (priority 4 or 5).
- The muting conditions are defined as follows.
  - Forward direction

Muting condition 1 (8000 → 8010, MS_11 as the first start switch): When timers MaxMutingTime and DiscTime11_12 start: MutingEnable AND (R_TRIG at MS_11 AND NOT MS_12 AND NOT MS_21 AND NOT MS_22)

Muting condition 1 (8000 → 8310, MS_12 as the first start switch): When timers MaxMutingTime and DiscTime11_12 start: MutingEnable AND (NOT MS_11 AND R_TRIG at MS_12 AND NOT MS_21 AND NOT MS_22)

Muting condition 2 (8010 → 8020, MS_12 as the second start switch): When timer DiscTime11_12 stops: MutingEnable AND (MS_11 AND R_TRIG at MS_12 AND NOT MS_21 AND NOT MS_22)

Muting condition 2 (8310 → 8020, MS_11 as the second start switch): When timer DiscTime11_12 stops: MutingEnable AND (R_TRIG at MS_11 AND MS_12 AND NOT MS_21 AND NOT MS_22)

Muting condition 3 (two switches simultaneously activated): When timer MaxMutingTime starts: MutingEnable AND (R_TRIG at MS_11 AND R_TRIG at MS_12 AND NOT MS_21 AND NOT MS_22)

Muting condition 4 (all switches activated): MS_11 AND MS_12 AND MS_21 AND MS_22

Muting condition 24 (8020 → 8030, MS_21 as the first start switch): When timer DiscTime21_22 starts: MS_11 AND MS_12 AND R_TRIG at MS_21 AND NOT MS_22

Muting condition 24 (8020 → 8330, MS_22 as the first start switch): When timer DiscTime21_22 starts: MS_11 AND MS_12 AND NOT MS_21 AND R_TRIG at MS_22

Muting condition 25 (8030→8040, MS_22 as the second start switch): When timer DiscTime21_22 stops: MS_11 AND MS_12 AND MS_21 AND R_TRIG at MS_22

Muting condition 25 (8330→8040, MS_21 as the second start switch): When timer DiscTime21_22 stops: MS_11 AND MS_12 AND R_TRIG at MS_21 AND MS_22

Muting condition 5 (a stop switch released): When timer MaxMutingTime stops: NOT MS_11 AND NOT MS_12 AND (F_TRIG at MS_21 OR F_TRIG at MS_22)

- Backward direction

  Muting condition 11 (8000 → 8110, MS_21 as the first start switch): When timers MaxMutingTime and DiscTime21_22 start: MutingEnable AND (NOT MS_22 AND R_TRIG at MS_21 AND NOT MS_11 AND NOT MS_12)

  Muting condition 11 (8000 → 8410, MS_22 as the first start switch): When timers MaxMutingTime and DiscTime21_22 start: MutingEnable AND (R_TRIG at MS_22 AND NOT MS_21 AND NOT MS_11 AND NOT MS_12)

  Muting condition 12 (8110 → 8120, MS_22 as the second start switch): When timer DiscTime21_22 stops: MutingEnable AND (MS_21 AND R_TRIG at MS_22 AND NOT MS_11 AND NOT MS_12)

  Muting condition 12 (8410 → 8120, MS_21 as the second start switch): When timer DiscTime21_22 stops: MutingEnable AND (R_TRIG at MS_21 AND MS_22 AND NOT MS_11 AND NOT MS_12)

  Muting condition 13 (two switches activated simultaneously): When timer MaxMutingTime starts: MutingEnable AND (R_TRIG at MS_21 AND R_TRIG at MS_22 AND NOT MS_11 AND NOT MS_12)

  Muting condition 14 (all switches activated): MS_11 AND MS_12 AND MS_21 AND MS_22

Muting condition 44 (8120→8130, MS_11 as the first start switch): When timer DiscTime11_12 starts: MS_21 AND MS_22 AND R_TRIG at MS_11 AND NOT MS_12

Muting condition 44 (8120→8430, MS_12 as the first start switch): When timer DiscTime11_12 starts: MS_21 AND MS_22 AND NOT MS_11 AND R_TRIG at MS_12

Muting condition 45 (8130 → 8140, MS_12 as the second start switch): When timer DiscTime11_12 stops: MS_21 AND MS_22 AND MS_11 AND R_TRIG at MS_12

Muting condition 45 (8430 → 8140, MS_11 as the second start switch): When timer DiscTime11_12 stops: MS_21 AND MS_22 AND R_TRIG at MS_11 AND MS_12

Muting condition 15 (a stop switch released): When timer MaxMutingTime stops: NOT MS_21 AND NOT MS_22 AND (F_TRIG at MS_11 OR F_TRIG at MS_12)

- Incorrect muting sequence

  Status 8000:

  (MutingEnable = FALSE when the muting sequence starts) OR

  ((MS_11 OR MS_12) AND (MS_21 OR MS_22)) OR

  (R_TRIG at MS_11 AND MS_12 AND NOT R_TRIG at MS_12) OR

  (R_TRIG at MS_12 AND MS_11 AND NOT R_TRIG at MS_11) OR

  (R_TRIG at MS_21 AND MS_22 AND NOT R_TRIG at MS_22) OR

  (R_TRIG at MS_22 AND MS_21 AND NOT R_TRIG at MS_21) OR

  ((MS_11 AND NOT R_TRIG at MS_11) AND (MS_12 AND NOT R_TRIG at MS_12)) OR

  ((MS_21 AND NOT R_TRIG at MS_21) AND (MS_22 AND NOT R_TRIG at MS_22))

  Status 8010: NOT MutingEnable OR NOT MS_11 OR MS_21 OR MS_22

  Status 8310: NOT MutingEnable OR NOT MS_12 OR MS_21 OR MS_22

  Status 8020: NOT MS_11 OR NOT MS_12

  Status 8040:

  R_TRIG at MS_21 OR R_TRIG at MS_22

  OR ((MS_11 OR MS_12) AND (F_TRIG at MS_21 OR F_TRIG at MS_22))

  OR ((F_TRIG at MS_21 OR F_TRIG at MS_22))

  Status 8030: NOT MS_11 OR NOT MS_12 OR NOT MS_21

  Status 8330: NOT MS_11 OR NOT MS_12 OR NOT MS_22

  Status 8110: NOT MutingEnable OR MS_11 OR MS_12 OR NOT MS_21

  Status 8410: NOT MutingEnable OR MS_11 OR MS_12 OR NOT MS_22

  Status 8120: NOT MS_21 OR NOT MS_22

  Status 8410:

  R_TRIG at MS_21 OR R_TRIG at MS_22

  OR ((MS_21 OR MS_22) AND (F_TRIG at MS_11 OR F_TRIG at MS_12))

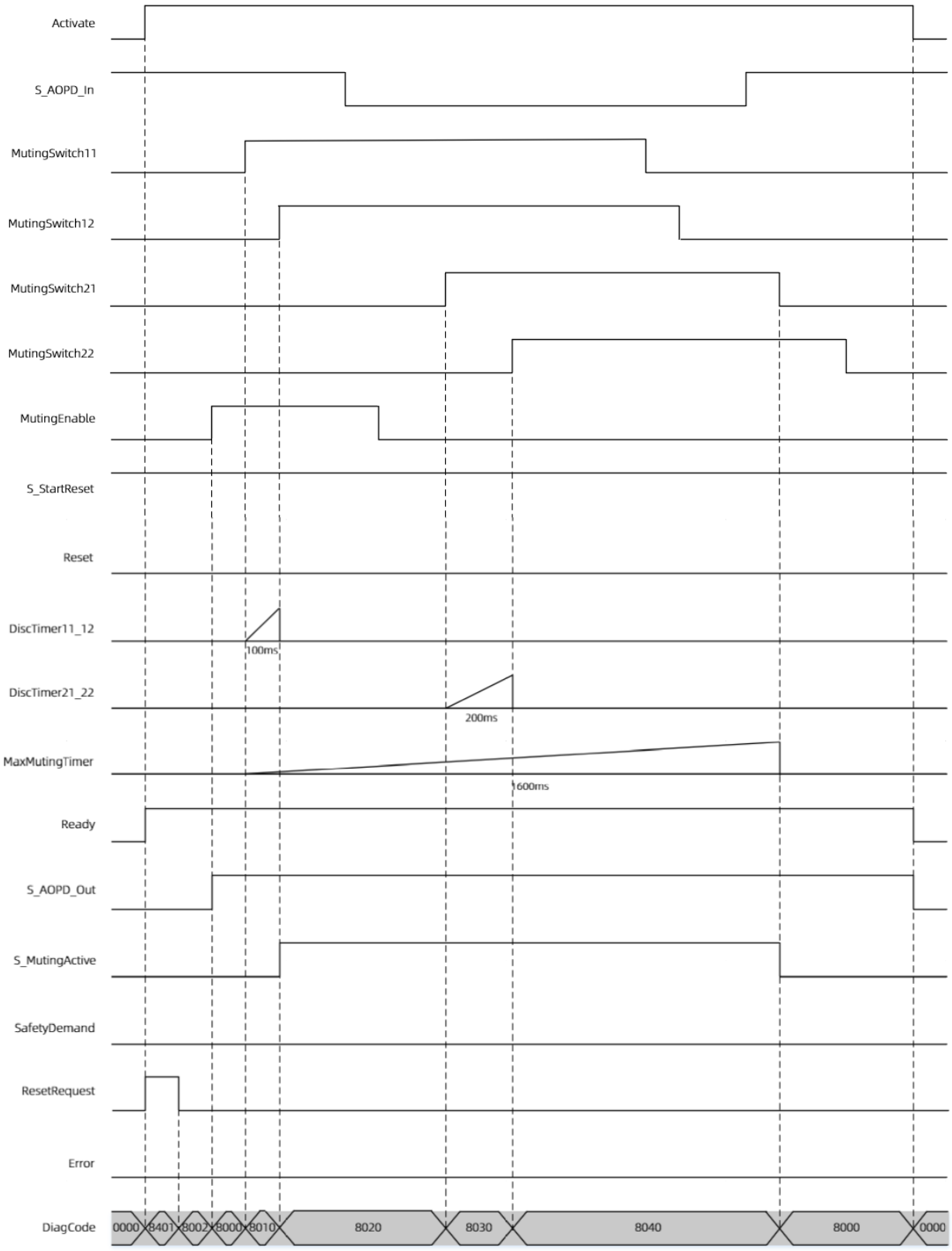
  OR ((MS_21 OR MS_22) AND (F_TRIG at MS_11 OR F_TRIG at MS_12))

  OR (AND (F_TRIG at MS_21 OR F_TRIG at MS_22))

Status 8130: NOT MS_21 OR NOT MS_22 OR NOT MS_11

Status 8430: NOT MS_21 OR NOT MS_22 OR NOT MS_12

- **Sequence diagram**



- **Fault detection**

  This function block detects the following fault conditions:

  - DiscTime11_12 and DiscTime21_22 are set to values less than T#0s or greater than T#4s.
  - MaxMutingTime is set to a value less than T#0s or greater than T#120min.

- The discrepancy time for MutingSwitch11/MutingSwitch12 or MutingSwitch21/MutingSwitch22 sensor pairs exceeds the timeout setting.
- The muting function (S_MutingActive = TRUE) exceeds the maximum muting time MaxMutingTime.
- The muting sensors MutingSwitch11, MutingSwitch12, MutingSwitch21, and MutingSwitch22 are activated in an incorrect sequence.
- The muting sequence is started without being enabled by MutingEnable.
- A static Reset condition is detected in statuses 8401 and 8402.

- **Fault behavior**

  In case of a fault, the S_AOPD_Out and S_MutingActive outputs are set to FALSE. The DiagCode output indicates the relevant fault code, and the Error output is set to TRUE. Restart is prohibited until the fault condition is cleared and the safety status is confirmed by the operator via Reset.

- **Function block-specific fault codes and status codes**
  - Fault code

DiagCode	Status Name	Description and Output
C001	Reset Error 1	In the 8401 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C011	Reset Error 2	In the 8402 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

CYx4	Error Muting sequence	Fault detected in muting sequence statuses 8000, 8010, 8310, 8020, 8040, 8030, 8330, 8110, 8410, 8120, 8140, 8130, or 8430. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE Y indicates the fault occurred under different status codes. Fault code C0x4 indicates the fault occurred in status 8000. Fault code C1x4 indicates the fault occurred in status 8010. Fault code C2x4 indicates the fault occurred in status 8310. Fault code C3x4 indicates the fault occurred in status 8020. Fault code C4x4 indicates the fault occurred in status 8030. Fault code C5x4 indicates the fault occurred in status 8330. Fault code C6x4 indicates the fault occurred in status 8040. Fault code C7x4 indicates the fault occurred in status 8110. Fault code C8x4 indicates the fault occurred in status 8410. Fault code C9x4 indicates the fault occurred in status 8120. Fault code CAx4 indicates the fault occurred in status 8130. Fault code CBx4 indicates the fault occurred in status 8430. Fault code CCx4 indicates the fault occurred in status 8140. Fault code CCx4 indicates the fault occurred when MutingEnable is FALSE. x reflects the sequence of sensor faults (4 bits: MS_11, MS_12, MS_21, MS_22 from low to high).

DiagCode	Status Name	Description and Output
C010	Parameter Error	Timeout of timer DiscTime11_12, DiscTime21_22, or MaxMutingTime. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C020	Error Timer MaxMuting	The active muting time (when S_MutingActive = TRUE) exceeds MaxMutingTime. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C030	Error Timer MS11_12	The discrepancy time between the switching of MutingSwitch11 and MutingSwitch12 exceeds DiscTime11_12. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C040	Error Timer MS21_22	The discrepancy time between the switching of MutingSwitch21 and MutingSwitch22 exceeds DiscTime21_22. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

◦ Status code

DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8000	AOPD Free (Active photoelectric guard device is idle)	Muting is not activated and there is no safety demand for AOPD. If the subsequent muting timer is still running, it will be stopped. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8401	Init (Initialization)	Function block is activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = FALSE
8802	Safety Demand AOPD	A safety demand for AOPD is detected and muting is not activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8402	Wait for Reset	A safety demand or fault is detected, and is now cleared. Operator reset confirmation is required. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = FALSE
8002	Safe	The safety function is activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8010	Muting Forward Start 1	The muting forward sequence starts after MutingSwitch 11 is triggered. Monitoring of DiscTime11_12 is activated. Monitoring of MaxMutingTime is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8310	Muting Forward Start 2	The muting forward sequence starts after MutingSwitch 12 is triggered. Monitoring of DiscTime11_12 is activated. Monitoring of MaxMutingTime is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8020	Muting Forward Active 1	The muting forward sequence is activated when the trigger of the second entry of MutingSwitch 12 or 11 is detected, or when both MutingSwitch 11 and 12 are activated in the same cycle. Monitoring of DiscTime11_12 is stopped. If transitioning from status 8000, monitoring of MaxMutingTime is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8030	Muting Forward Step 1	The muting forward sequence is activated. MutingSwitch21 is the first exit switch being activated. Monitoring of DiscTime21_22 is started. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8330	Muting Forward Step 2	The muting forward sequence is activated. MutingSwitch22 is the first exit switch being activated. Monitoring of DiscTime21_22 is started. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8040	Muting Forward Active 2	The muting forward sequence is activated. Both exit switches MutingSwitch21 and 22 are activated. Monitoring of DiscTime21_22 is stopped. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8110	Muting Backward Start 1	The muting backward sequence starts after MutingSwitch 21 is triggered. Monitoring of DiscTime21_22 is activated. Monitoring of MaxMutingTime is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

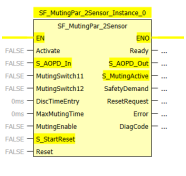
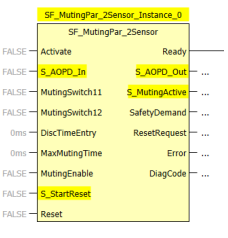
DiagCode	Status Name	Description and Output
8410	Muting Backward Start 2	The muting backward sequence starts after MutingSwitch 22 is triggered. Monitoring of DiscTime21_22 is activated. Monitoring of MaxMutingTime is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8120	Muting Backward Active 1	The muting backward sequence is activated when the trigger of the second entry of MutingSwitch 21 or 22 is detected, or when both MutingSwitch 21 and 22 are activated in the same cycle. Monitoring of DiscTime21_22 is stopped. If transitioning from status 8000, monitoring of MaxMutingTime is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8130	Muting Backward Step 1	The muting backward sequence is activated. MutingSwitch11 is the first exit switch being activated. Monitoring of DiscTime11_12 is started. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8430	Muting Backward Step 2	The muting backward sequence is activated. MutingSwitch12 is the first exit switch being activated. Monitoring of DiscTime11_12 is started. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8140	Muting Backward Active 2	The muting backward sequence is activated. Both exit switches MutingSwitch11 and 12 are activated. Monitoring of DiscTime11_12 is stopped. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

11.10.2.2.17 SF_MutingPar_2Sensor

SF_MutingPar_2Sensor intentionally inhibits the safety function. In this function block, parallel muting with two muting sensors is specified.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_MutingPar_2Sensor	Perform parallel muting with two muting sensors	FB		

- **Related variables**

  Variables

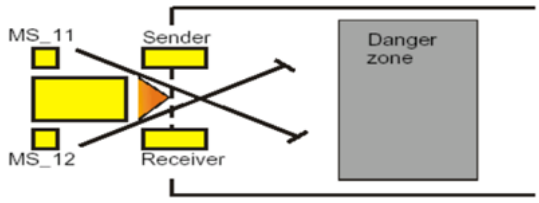
Input / Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
	S_AOPD_In	SAFEBOOL	TRUE, FALSE	FALSE	OSSD signal from AOPD. FALSE: The guarded area is invaded. TRUE: The guarded area is not invaded.
	MutingSwitch11	BOOL	TRUE, FALSE	FALSE	Status of muting sensor 11. FALSE: Muting sensor 11 is not activated. TRUE: Muting sensor 11 is activated.
	MutingSwitch12	BOOL	TRUE, FALSE	FALSE	Status of muting sensor 12. FALSE: Muting sensor 12 is not activated. TRUE: Muting sensor 12 is activated.
	DiscTimeEntry	TIME	Follows data type	T#0ms	Maximum discrepancy time between MutingSwitch11 and MutingSwitch12, with a range of 0 to 4s.
	MaxMutingTime	TIME	Follows data type	T#0ms	Maximum time for the complete muting sequence, with a range of 0 to 120 minutes.
	MutingEnable	BOOL	TRUE, FALSE	FALSE	Control system command used to activate the muting function when required. FALSE: Muting is invalid. TRUE: Muting is valid.
	S_StartReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to start reset.
	Reset	BOOL	TRUE, FALSE	FALSE	Manual reset
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	S_AOPD_Out	SAFEBOOL	TRUE, FALSE	FALSE	Safety-related output indicating the status of the muting guard device. FALSE: The AOPD guarded area is invaded and muting is invalid. TRUE: The AOPD guarded area is not invaded and muting is valid.

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
	S_MutingActive	SAFEBOOL	TRUE, FALSE	FALSE	Indicates the status of the muting process. FALSE: Muting is invalid. TRUE: Muting is valid.
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	ResetRequest	BOOL	TRUE, FALSE	FALSE	Flag indicating whether a reset is required.
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**
  - Muting refers to the intentional inhibition of the safety function. For example, SF_MutingPar can be used when feeding material into a hazardous area without causing the machine to stop. Muting is triggered by muting sensors and requires two muting sensors with proper integration into the production process to ensure no personnel enters the hazardous area during the muting period. Muting sensors may include non-safety proximity switches, photoelectric barriers, limit switches, and so on. The activation of muting must be indicated by an indicator.
  - The muting process is divided into sequential muting and parallel muting. In this function block, parallel muting with two muting sensors is used. The positioning of the sensors shall comply with the requirements specified in Appendix F.7 of IEC 62046 (CD 2005 edition). The function block can be used in both forward and backward directions, but the actual direction cannot be identified. To prevent misoperation, muting shall be activated by the MutingEnable signal from the process.
  - Input parameters of this function block include:
    - Two muting sensor signals (MutingSwitch11 and MutingSwitch12).
    - OSSD signal from the active photoelectric guard device (S_AOPD_In).
    - Two parameterizable times (DiscTimeEntry and MaxMutingTime).
  - The S_StartReset input can only be activated when it is ensured that the activation of the programmable electronic system (PES) will not cause hazardous conditions.

Table11-10-2-2-17-1 Example of forward muting process with two muting sensors

No.	Figure	Explanation
1		When using retroreflective sensors as muting sensors, it is generally configured diagonally. Typically, this configuration requires two photoelectric sensors, and the assignment of S_MutingSwitch11 (MS_11) and S_MutingSwitch12 (MS_12).

- **Status transition diagram**

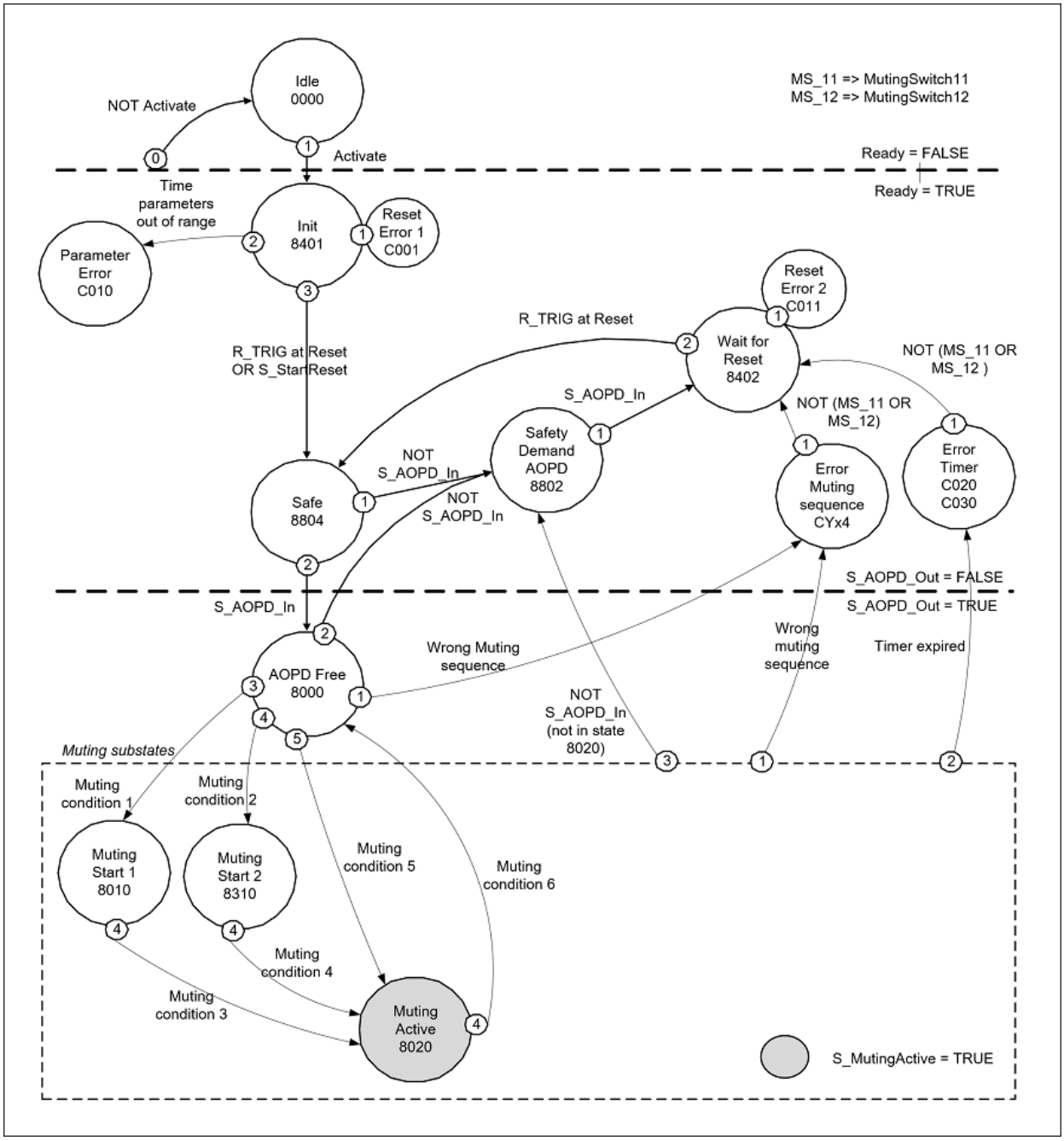


Figure 11-10-2-2-17-1 Status transition diagram SF_MutingPar_2Senso

◦ This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.

- The transition from any status to the idle status with Activate being FALSE is not shown. However, these transitions have the highest priority (0).
- In the muting status, AOPD transitions triggered by an incorrect muting sequence (priority 1), a faulty timer (priority 2), and a safety request (priority 3) have higher priority than transitions to the muting status (priority 4 or 5).
- The muting conditions are defined as follows.
  - Muting conditions

    Muting condition 1 (8000 → 8010, MS_11 as the first start switch): When timers DiscTimeEntry and MaxMutingTime start: MutingEnable AND R_TRIG at MS_11 AND NOT MS_12

    Muting condition 2 (8000 → 8310, MS_12 as the first start switch): When timers DiscTimeEntry and MaxMutingTime start: MutingEnable AND NOT MS_11 AND R_TRIG at MS_12

    Muting condition 3 (8010 → 8020, MS_12 as the second start switch): When timer DiscTimeEntry stops: MutingEnable AND MS_11 AND R_TRIG at MS_12

    Muting condition 4 (8310 → 8020, MS_11 as the second start switch): When timer DiscTimeEntry stops: MutingEnable AND R_TRIG at MS_11 AND MS_12

    Muting condition 5 (8000 → 8020, both switches activated simultaneously): When timer MaxMutingTime starts: MutingEnable AND R_TRIG at MS_11 AND R_TRIG at MS_12

    Muting condition 6 (8020 → 8000, both switches are released simultaneously or MS_11 and MS_12 released consecutively): When timer MaxMutingTime stops: NOT MS_11 OR NOT MS_12
  - Incorrect muting sequence

    Status 8000:

    (R_TRIG at MS_11 AND MS_12 AND NOT R_TRIG at MS_12)

    OR (R_TRIG at MS_12 AND MS_11 AND NOT R_TRIG at MS_11)

    OR ((MS_11 AND NOT R_TRIG at MS_11) AND (MS_12 AND NOT R_TRIG at MS_12))

    OR (NOT MutingEnable AND R_TRIG at MS_11)

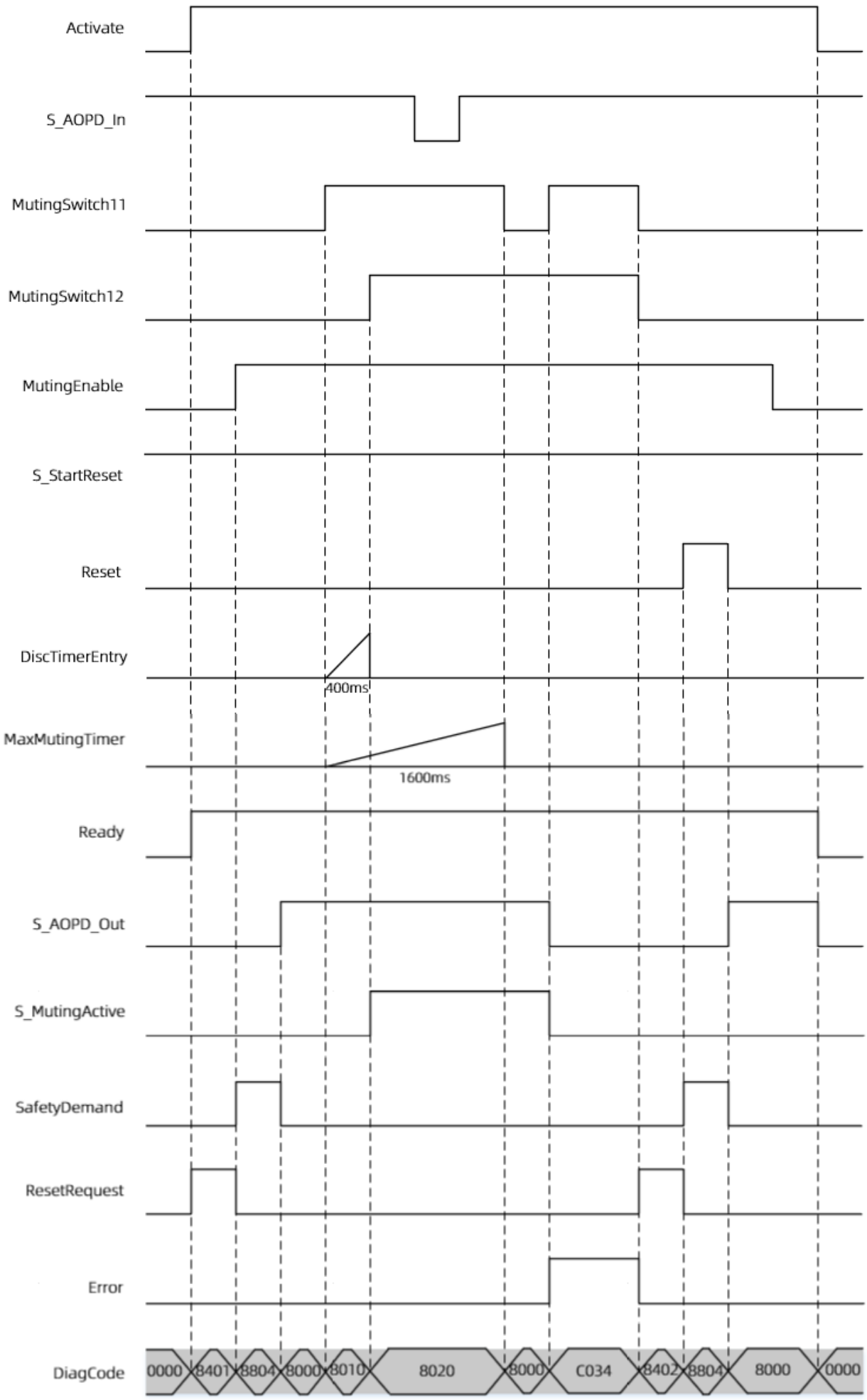
    OR (NOT MutingEnable AND R_TRIG at MS_12)

    Status 8010: NOT MutingEnable OR NOT MS_11

    Status 8310: NOT MutingEnable OR NOT MS_12

    Status 8020: No incorrect muting sequence

- **Sequence diagram**


Activate
S_AOPD_In
MutingSwitch11
MutingSwitch12
MutingEnable
S_StartReset
Reset
DiscTimerEntry
400ms
MaxMutingTimer
1600ms
Ready
S_AOPD_Out
S_MutingActive
SafetyDemand
ResetRequest
Error
DiagCode
0000
8401
8804
8000
8010
8020
8000
C034
8402
8804
8000
0000

- **Fault detection**

  This function block detects the following fault conditions:

  - DiscTimeEntry is set to a value less than T#0s or greater than T#4s.
  - MaxMutingTime is set to a value less than T#0s or greater than T#120min.
  - The discrepancy time between the MutingSwitch11/MutingSwitch12 sensor pair exceeds the timeout.
  - The muting function (S_MutingActive = TRUE) exceeds the maximum muting time MaxMutingTime.
  - The muting sensors MutingSwitch11 and MutingSwitch12 are activated in the wrong order.
  - The muting sequence is started without being enabled by MutingEnable.
  - Static signals of the muting sensor exist.
  - A static Reset condition is detected in statuses 8401 and 8402.

- **Fault behavior**

  In case of a fault, the S_AOPD_Out and S_MutingActive outputs are set to FALSE. The DiagCode output indicates the relevant fault code, and the Error output is set to TRUE. Restart is prohibited until the fault condition is cleared and the safety status is confirmed by the operator via Reset.

- **Function block-specific fault codes and status codes**

  - Fault code

DiagCode	Status Name	Description and Output
C001	Reset Error 1	In the 8401 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C011	Reset Error 2	In the 8402 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
CYx4	Error Muting sequence	A fault is detected in muting sequence in statuses 8000, 8010, or 8310. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE Y indicates the fault occurred under different status codes. Fault code C0x4 indicates the fault occurred in status 8000. Fault code C1x4 indicates the fault occurred in status 8010. Fault code C2x4 indicates the fault occurred in status 8310. Fault code CCx4 indicates the fault occurred when MutingEnable is FALSE. x reflects the sequence of sensor faults (4 bits: MS_11 and MS_12 from low to high).
C010	Parameter Error	The value of DiscTimeEntry or MaxMutingTime is invalid. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C020	Error Timer MaxMuting	The active muting time (when S_MutingActive = TRUE) exceeds MaxMutingTime. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
C030	Error Timer Entry	The discrepancy time between the switching of MutingSwitch11 and MutingSwitch12 exceeds DiscTimeEntry. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

◦ Status code

DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8000	AOPD Free (Active photoelectric guard device is idle)	Muting is not activated and there is no safety demand for AOPD. If the subsequent muting timer is still running, it will be stopped. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8401	Init (Initialization)	Function block is activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = FALSE

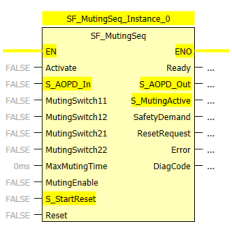
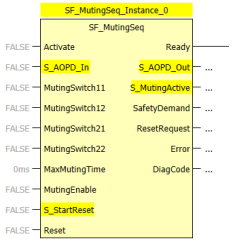
DiagCode	Status Name	Description and Output
8802	Safety Demand AOPD	A safety demand for AOPD is detected and muting is not activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8402	Wait for Reset	A safety demand or fault is detected, and is now cleared. Operator reset confirmation is required. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = FALSE
8004	Safe	The safety function is activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8010	Muting Start 1	The muting sequence starts after MutingSwitch 11 is triggered. Monitoring of DiscTimeEntry is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8310	Muting Start 2	The muting sequence starts after MutingSwitch 12 is triggered. Monitoring of DiscTimeEntry is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8020	Muting Active	The muting sequence is activated when the trigger of the second entry of MutingSwitch 12 or 11 is detected, or when both MutingSwitch 11 and 12 are activated in the same cycle. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

11.10.2.2.18 SF_MutingSeq

SF_MutingSeq intentionally inhibits the safety function. In this function block, sequential muting with four muting sensors is specified.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_MutingSeq	Perform sequential muting with four muting sensors	FB		

- **Related variables**

  Variables

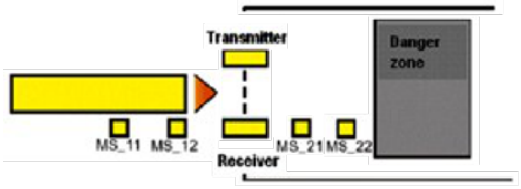
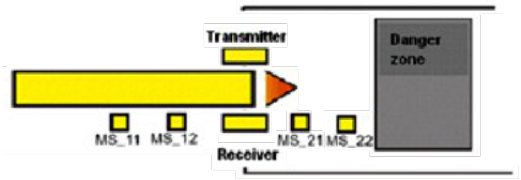
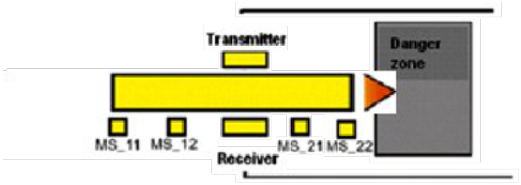
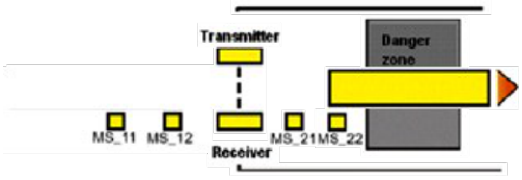
Input / Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
	S_AOPD_In	SAFEBOOL	TRUE, FALSE	FALSE	OSSD signal from AOPD. FALSE: The guarded area is invaded. TRUE: The guarded area is not invaded.
	MutingSwitch11	BOOL	TRUE, FALSE	FALSE	Status of muting sensor 11. FALSE: Muting sensor 11 is not activated. TRUE: Muting sensor 11 is activated.
	MutingSwitch12	BOOL	TRUE, FALSE	FALSE	Status of muting sensor 12. FALSE: Muting sensor 12 is not activated. TRUE: Muting sensor 12 is activated.
	MutingSwitch21	BOOL	TRUE, FALSE	FALSE	Status of muting sensor 21. FALSE: Muting sensor 21 is not activated. TRUE: Muting sensor 21 is activated.
	MutingSwitch22	BOOL	TRUE, FALSE	FALSE	Status of muting sensor 22. FALSE: Muting sensor 22 is not activated. TRUE: Muting sensor 22 is activated.
	MaxMutingTime	TIME	Follows data type	T#0ms	Maximum time for the complete muting sequence, with a range of 0 to 120 minutes.
	MutingEnable	BOOL	TRUE, FALSE	FALSE	Control system command used to activate the muting function when required. FALSE: Muting is invalid. TRUE: Muting is valid.
	S_ StartReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to start reset.
	Reset	BOOL	TRUE, FALSE	FALSE	Manual reset
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
	S_AOPD_Out	SAFEBOOL	TRUE, FALSE	FALSE	Safety-related output indicating the status of the muting guard device. FALSE: The AOPD guarded area is invaded and muting is invalid. TRUE: The AOPD guarded area is not invaded and muting is valid.
	S_MutingActive	SAFEBOOL	TRUE, FALSE	FALSE	Indicates the status of the muting process. FALSE: Muting is invalid. TRUE: Muting is valid.
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	ResetRequest	BOOL	TRUE, FALSE	FALSE	Flag indicating whether a reset is required.
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**
  - Muting refers to the intentional inhibition of the safety function. For example, SF_MutingPar can be used when feeding material into a hazardous area without causing the machine to stop. Muting is triggered by muting sensors and requires two or four muting sensors with proper integration into the production process to ensure no personnel enters the hazardous area during the muting period. Muting sensors may include non-safety proximity switches, photoelectric barriers, limit switches, and so on. The activation of muting must be indicated by an indicator.
  - The muting process is divided into sequential muting and parallel muting. In this function block, sequential muting with four muting sensors is used (the forward transport direction is used as an example). The function block can be used in both forward and backward directions. To prevent misoperation, muting shall be activated by the MutingEnable signal from the process. If the MutingEnable signal is not available, this input must be set to TRUE.
  - Input parameters of this function block include:
    - Four muting sensor signals (MutingSwitch11...MutingSwitch22).
    - OSSD signal from the active photoelectric guard device (S_AOPD_In).
  - The S_StartReset input can only be activated when it is ensured that the activation of the programmable electronic system (PES) will not cause hazardous conditions.

Table11-10-2-2-18-1 Example of forward muting process with four muting sensors

No.	Figure	Explanation
1		The muting mode remains active when MutingSwitch12 (MS_12) is turned ON due to the workpiece passage after MutingSwitch11 (MS_11).
2		The muting mode remains active while MutingSwitch11 (MS_11) and MutingSwitch12 (MS_12) are turned ON due to the workpiece. The workpiece can pass through the light curtain without stopping the machine.
3		Before MutingSwitch11 (MS_11) and MutingSwitch12 (MS_12) are turned OFF, MutingSwitch21 (MS_21) and MutingSwitch22 (MS_22) must be turned ON. This ensures that the muting mode remains ON.
4		The muting mode ends only when MutingSwitch22 (MS_22) is turned ON due to the workpiece.

- **Status transition diagram**

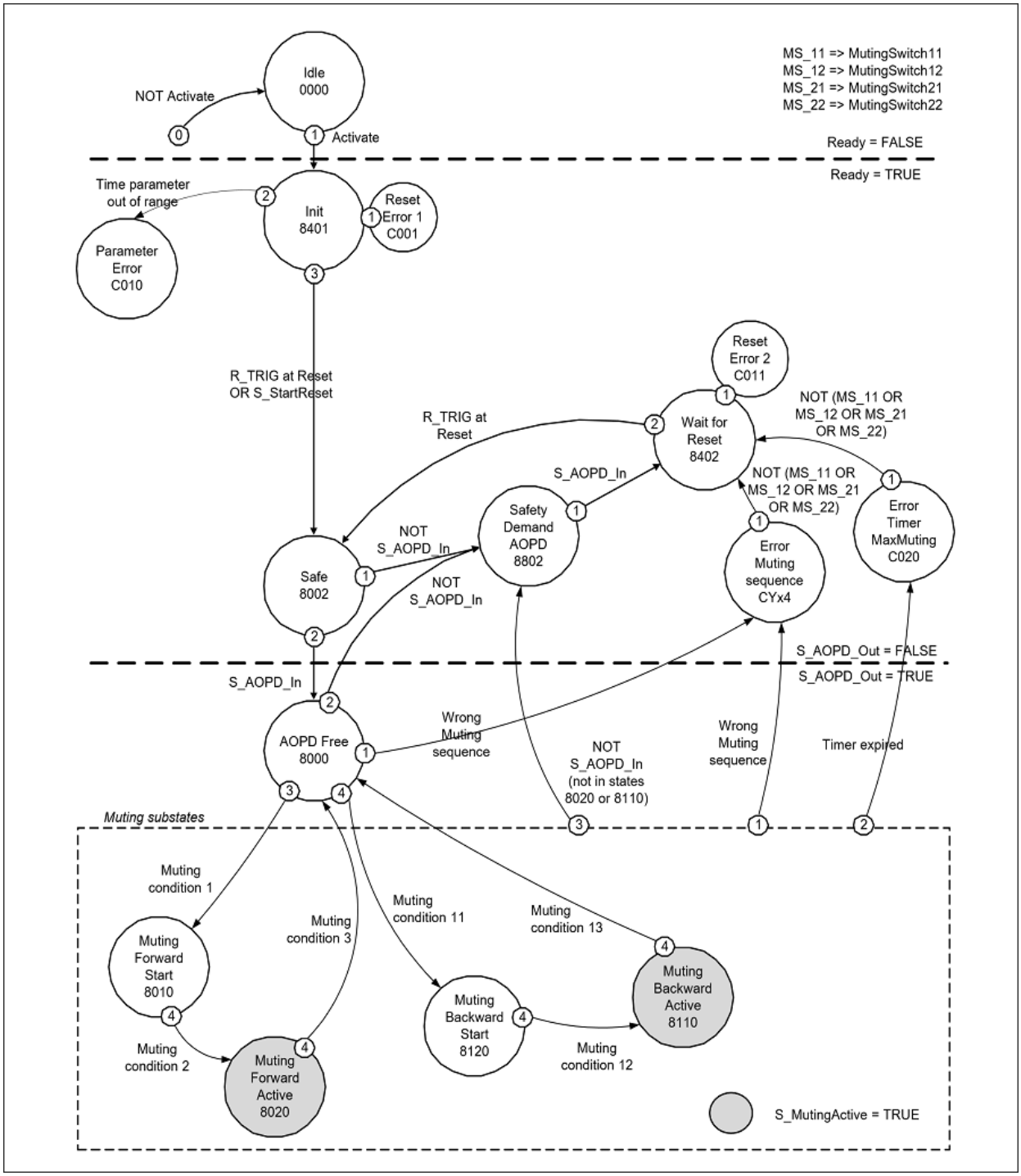


Figure 11-10-2-2-18-1 Status transition diagram for SF_MutingSeq

- This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.
- The transition from any status to the idle status with Activate being FALSE is not shown. However, these transitions have the highest priority (0).
- In the muting status, AOPD transitions triggered by an incorrect muting sequence (priority 1), a faulty timer (priority 2), and a safety request (priority 3) have higher priority than transitions to the muting status (priority 4 or 5).
- The muting conditions are defined as follows.
  - Forward direction

    Muting condition 1 (8000 → 8010, MS_11 as the first start switch): When timer MaxMutingTime starts: MutingEnable AND (R_TRIG at MS_11 AND NOT MS_12 AND NOT MS_21 AND NOT MS_22)

Muting condition 2 (8010 → 8020, MS_12 as the second start switch): When timer MaxMutingTime starts: MutingEnable AND (MS_11 AND R_TRIG at MS_12 AND NOT MS_21 AND NOT MS_22)

Muting condition 3 (8020 → 8000, MS_21 as the first override switch): When timer MaxMutingTime stops: NOT MS_11 AND NOT MS_12 AND F_TRIG at MS_21 AND MS_22

- Backward direction

  Muting condition 11 (8000 → 8120, MS_22 as the first start switch): When timer MaxMutingTime starts: MutingEnable AND (NOT MS_11 AND NOT MS_12 AND NOT MS_21 AND R_TRIG at MS_22)

  Muting condition 12 (8120 → 8110, MS_21 as the second start switch): When timer MaxMutingTime starts: MutingEnable AND (NOT MS_11 AND NOT MS_12 AND R_TRIG at MS_21 AND MS_22)

  Muting condition 13 (MS_12 as the first override switch). When timer MaxMutingTime starts: MS_11 AND F_TRIG at MS_12 AND NOT MS_21 AND NOT MS_22

- Incorrect muting sequence

  Status 8000:

  (NOT MutingEnable AND R_TRIG at MS_11)

  OR (NOT MutingEnable AND R_TRIG at MS_22)

  OR (MS_12 OR MS_21) OR (MS_11 AND MS_22)

  Status 8010: NOT MutingEnable OR NOT MS_11 OR MS_21 OR MS_22

  Status 8020:

  R_TRIG at MS_11 OR R_TRIG at MS_12 OR F_TRIG at MS_22

  OR (MS_11 AND F_TRIG at MS_12)

  OR ((MS_11 OR MS_12) AND (F_TRIG at MS_21))

  OR ((NOT MS_11 OR NOT MS_12) AND NOT MS_22)

  OR ((NOT MS_11 OR NOT MS_12 OR NOT MS_21) AND R_TRIG at MS_22)

  OR ((MS_11 AND MS_22) AND (NOT MS_12 OR NOT MS_21))

  OR (R_TRIG at MS_21 AND R_TRIG at MS_22)

  OR (F_TRIG at MS_11 AND F_TRIG at MS_12)

  OR (F_TRIG at MS_12 AND F_TRIG at MS_21)

  OR (NOT MS_11 AND MS_12 AND NOT MS_21)

  Status 8120: NOT MutingEnable OR MS_11 OR MS_12 OR NOT MS_22

  Status 8110:

  F_TRIG at MS_11 OR R_TRIG at MS_21 OR R_TRIG at MS_22

  OR (MS_22 AND F_TRIG at MS_21)

  OR ((MS_22 OR MS_21) AND (F_TRIG at MS_12))

  OR ((NOT MS_22 OR NOT MS_21) AND NOT MS_11)

  OR ((NOT MS_22 OR NOT MS_21 OR NOT MS_12) AND R_TRIG at MS_11)

  OR ((MS_11 AND MS_22) AND (NOT MS_12 OR NOT MS_21))

OR (R_TRIG at MS_11 AND R_TRIG at MS_12)

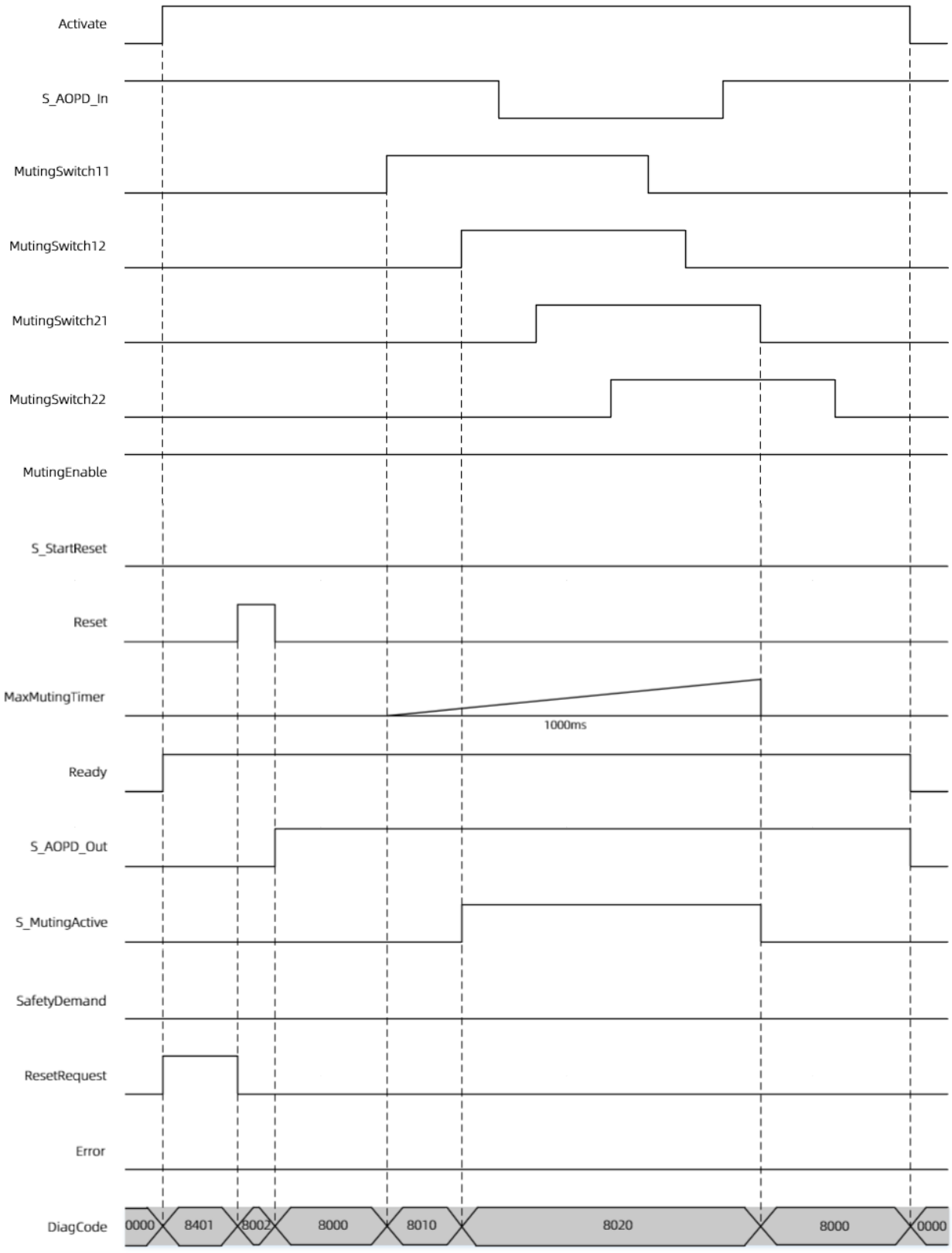
OR (F_TRIG at MS_22 AND F_TRIG at MS_21)

OR (F_TRIG at MS_21 AND F_TRIG at MS_12)

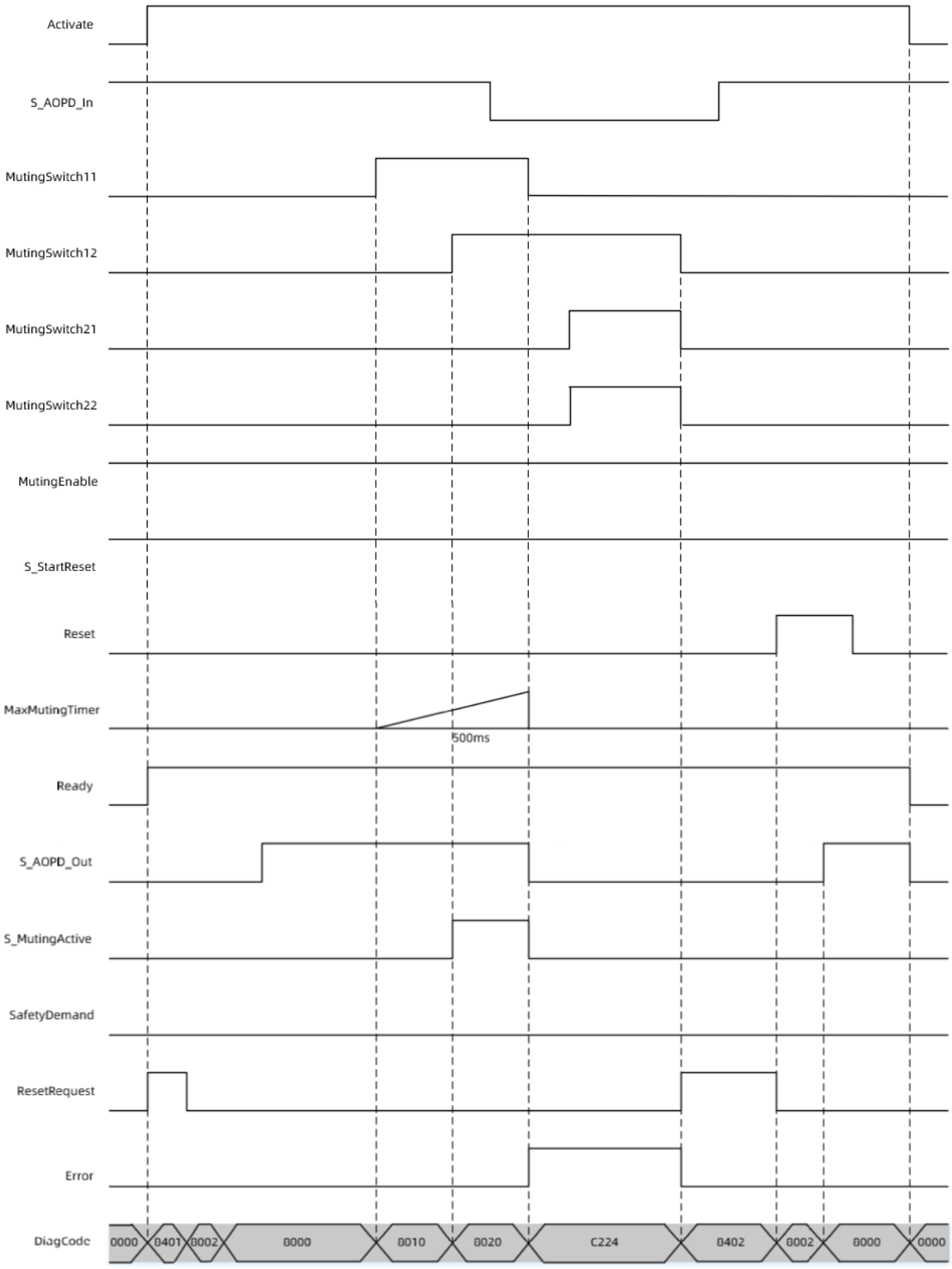
OR (NOT MS_12 AND MS_21 AND NOT MS_22)

- **Sequence diagram**

Manual reset



S_StartReset = TRUE



- **Fault detection**

  This function block detects the following fault conditions:

  - The muting sensors MutingSwitch11, MutingSwitch12, MutingSwitch21, and MutingSwitch22 are activated in an incorrect sequence.
  - The muting sequence is started without being enabled by MutingEnable.
  - A static Reset condition exists.
  - MaxMutingTime is set to a value less than T#0s or greater than T#120min.

- The muting function (S_MutingActive = TRUE) exceeds the maximum muting time MaxMutingTime.

- **Fault behavior**

  In case of a fault, the S_AOPD_Out and S_MutingActive outputs are set to FALSE. The DiagCode output indicates the relevant fault code, and the Error output is set to TRUE. Restart is prohibited until the fault condition is cleared and the safety status is confirmed by the operator via Reset.

- **Function block-specific fault codes and status codes**

  - Fault code

DiagCode	Status Name	Description and Output
C001	Reset Error 1	Detects that a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C011	Reset Error 2	In the 8402 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
CYx4	Error Muting sequence	A fault is detected in muting sequence in statuses 8000, 8010, 8020, 8120, or 8110 Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE Y indicates the fault occurred under different status codes. Fault code C0x4 indicates the fault occurred in status 8000. Fault code C1x4 indicates the fault occurred in status 8010. Fault code C2x4 indicates the fault occurred in status 8020. Fault code C3x4 indicates the fault occurred in status 8120. Fault code C4x4 indicates the fault occurred in status 8110. Fault code CCx4 indicates the fault occurred when MutingEnable is FALSE. x reflects the sequence of sensor faults (4 bits: MS_11, MS_12, MS_21, MS_22 from low to high).
C010	Parameter Error	MaxMutingTime value is out of range. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
C020	Error Timer MaxMuting	The active muting time (when S_MutingActive = TRUE) exceeds MaxMutingTime. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

◦ Status code

DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8000	AOPD Free	Muting is not activated and there is no safety demand for AOPD. If the subsequent muting timer is still running, it will be stopped. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8401	Init (Initialization)	Function block is activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = FALSE

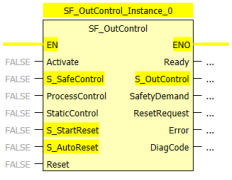
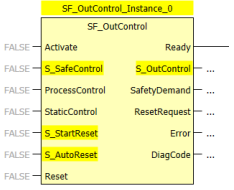
DiagCode	Status Name	Description and Output
8802	Safety Demand AOPD	A safety demand for AOPD is detected and muting is not activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8402	Wait for Reset	A safety demand or fault is detected, and is now cleared. Operator reset confirmation is required. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = FALSE
8002	Safe	The safety function is activated. Ready = TRUE S_AOPD_Out = FALSE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8010	Muting Forward Start	The muting forward sequence is in the start phase, with no safety demand. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8020	Muting Forward Active	The muting forward sequence is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8110	Muting Backward Start	The muting backward sequence is in the start phase, with no safety demand. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8120	Muting Backward Active	The muting backward sequence is activated. Ready = TRUE S_AOPD_Out = TRUE S_MutingActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

11.10.2.2.19 SF_OutControl

SF_OutControl controls safety outputs based on control signals and safety signals issued according to functional applications.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_OutControl	Output control	FB		

- **Related variables**

  Variables

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.
	S_SafeControl	SAFEBOOL	TRUE, FALSE	FALSE	Safety request input.
	ProcessControl	BOOL	TRUE, FALSE	FALSE	Control signal from functional application.
	StaticControl	BOOL	TRUE, FALSE	FALSE	Status of process control.
	S_StartReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to start reset.
	S_AutoReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to automatically reset.
	Reset	BOOL	TRUE, FALSE	FALSE	Manual reset
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	S_OutControl	SAFEBOOL	TRUE, FALSE	FALSE	Safety output.
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	ResetRequest	BOOL	TRUE, FALSE	FALSE	Flag indicating whether a reset is required.
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**
  - The SF_OutControl function block serves as a drive for safety output control. Its input signals include:
    - Process application signal: ProcessControl/BOOL (used to control process flow).
    - Safety application signal: S_SafeControl/SAFEBOOL (used to control safety functions).
  - Optional conditions for ProcessControl:
    - Condition 1: After the function block is activated or the safety signal (S_SafeControl) is fed back, ProcessControl must transition from FALSE to TRUE to trigger the additional function; a static TRUE signal cannot set S_OutControl to TRUE.
    - Condition 2: After the function block is activated or the safety signal (S_SafeControl) is fed back, ProcessControl is not required to transition from FALSE to TRUE for the additional function; a static TRUE signal can set S_OutControl to TRUE if other conditions are met.
  - Optional start inhibit conditions:
    - Start inhibit after function block activation
    - Start inhibit after guard device interruption

◦ Inputs StaticControl, S_StartReset, and S_AutoReset can only be activated when no hazardous situation will occur during PES startup.

- **Status transition diagram**

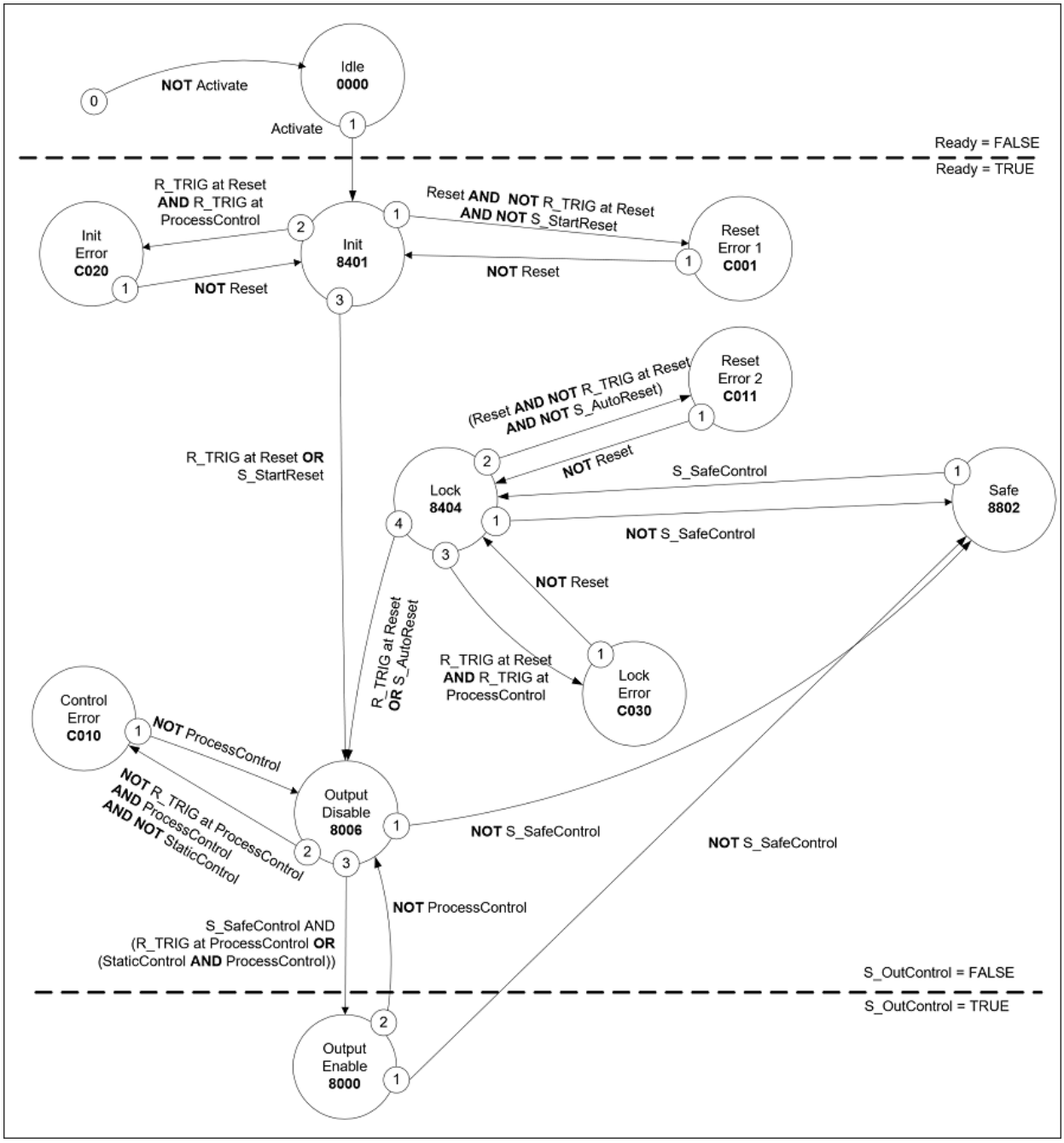


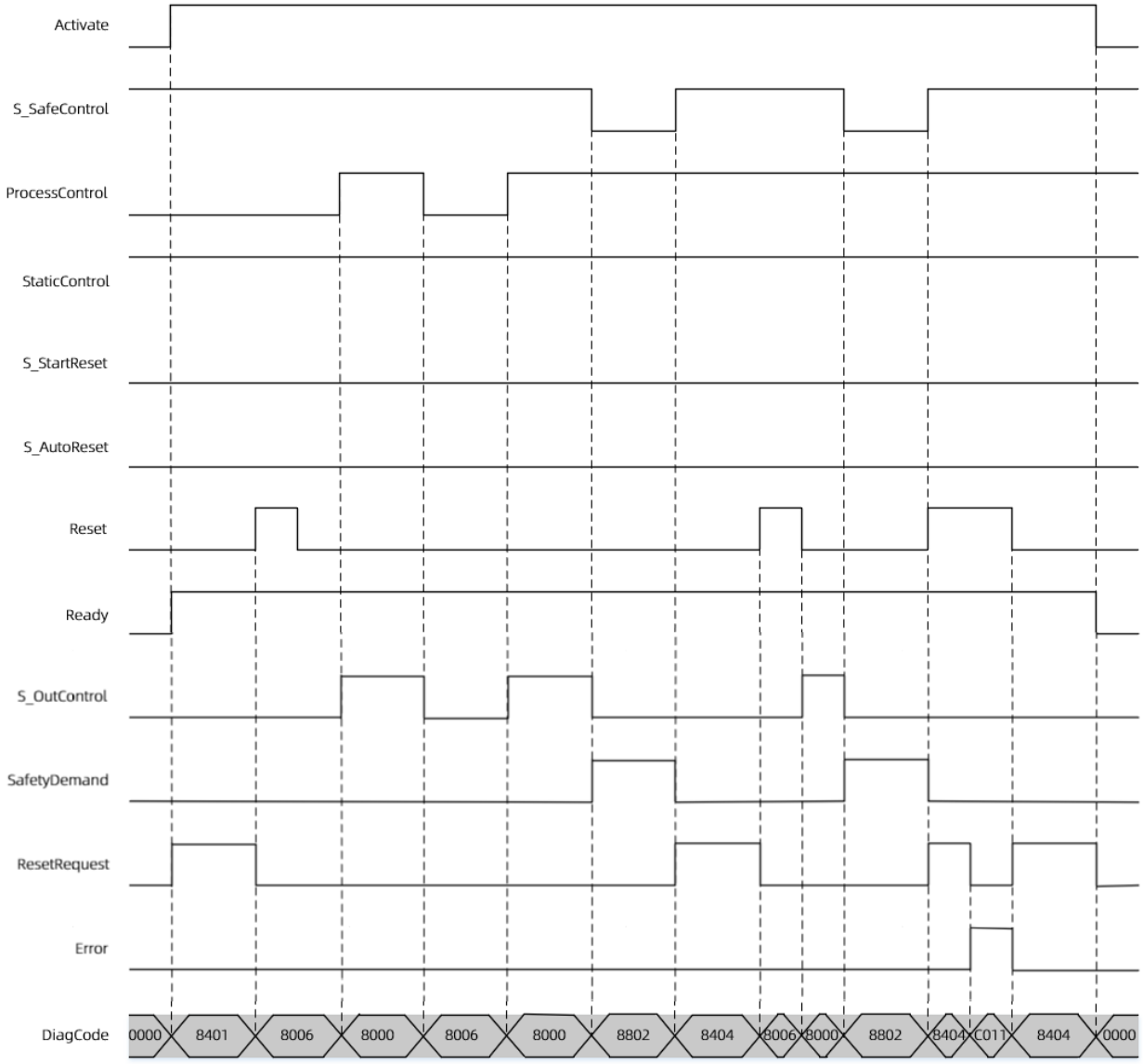
Figure 11-10-2-2-19-1 Status transition diagram for SF_OutControl

◦ This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.

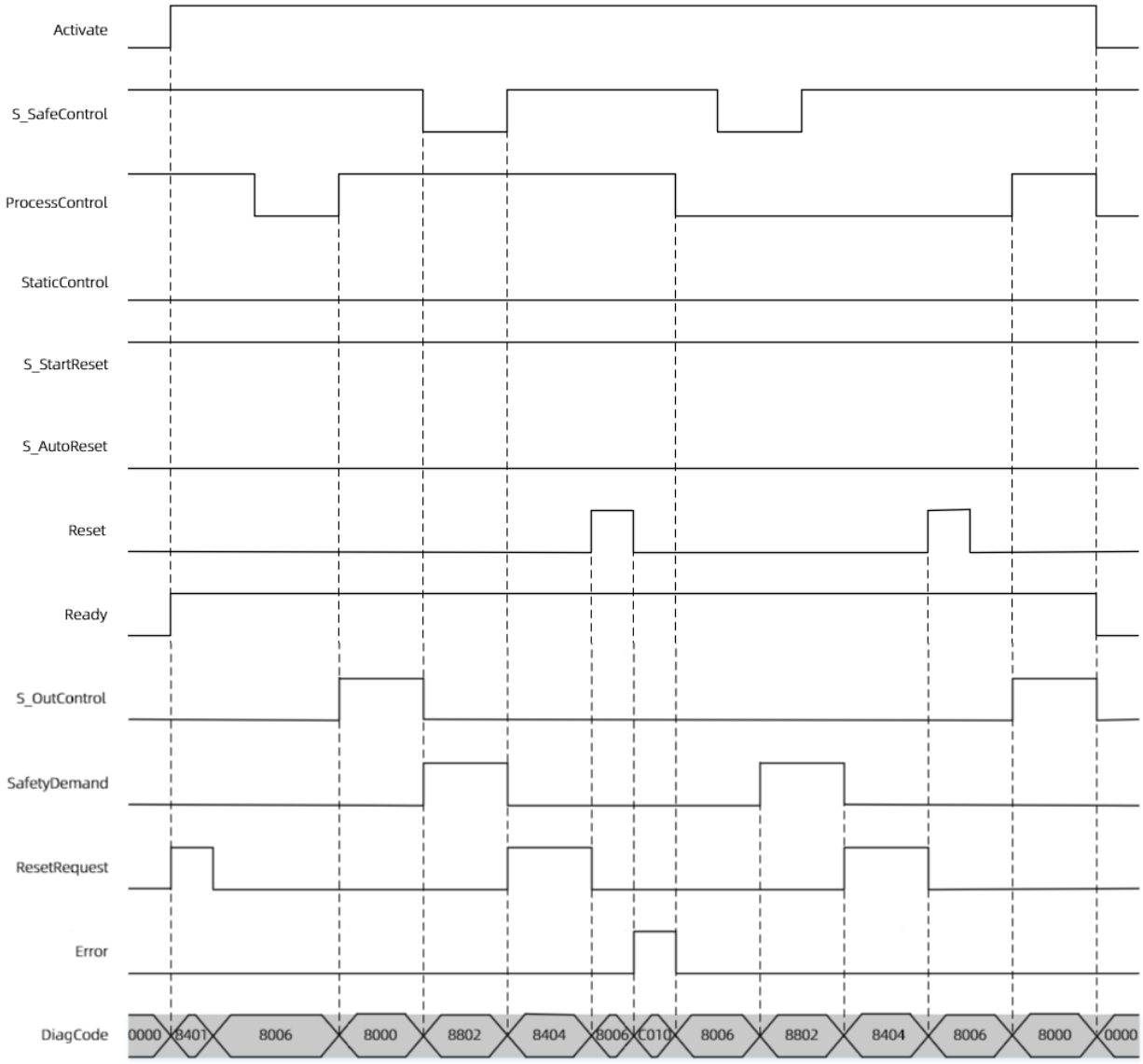
◦ The transition from any status to the idle status when Activate = FALSE is not indicated, but the priority (0) for transitioning to the idle status is the highest.

- **Sequence diagram**

S_StartReset = FALSE



S_StartReset = TRUE



- **Fault detection**

  The functional block forces a transition to the faulty status under the following conditions:

  - An invalid static Reset signal is detected.
  - An invalid static ProcessControl signal is detected.
  - ProcessControl and Reset are incorrectly interconnected due to a programming fault.

- **Fault behavior**

  When a fault occurs, S_OutControl is set to FALSE and remains in the safety status.

  - Fault clearance conditions:

    The Reset input must be set to FALSE to exit the Reset, Init, or Lock faulty status.

    The ProcessControl input must be set to FALSE to exit the Control faulty status.

  - Start inhibit reset method:

    After S_SafeControl switches to TRUE, the optional start inhibit can be reset by a rising edge at the Reset input.

    After the function block is activated, the optional start inhibit can be reset by a rising edge at the Reset input.

- **Function block-specific fault codes and status codes**
  - Fault code

DiagCode	Status Name	Description and Output
C001	Reset Error 1	In the 8401 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_OutControl = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C011	Reset Error 2	In the 8404 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_OutControl = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C010	Control Error	In the 8006 status, a TRUE value of ProcessControl is triggered at the non-rising edge. Ready = TRUE S_OutControl = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C020	Init Error	Reset and ProcessControl are triggered simultaneously in the 8401 status. Ready = TRUE S_OutControl = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
C030	Lock Error	Reset and ProcessControl are triggered simultaneously in the 8404 status. Ready = TRUE S_OutControl = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

◦ Status code

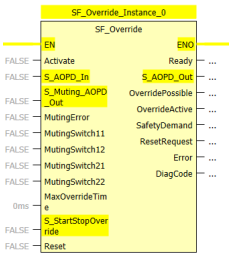
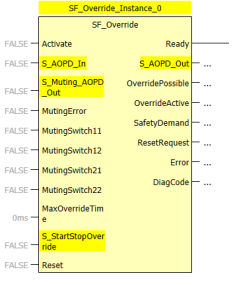
DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_OutControl = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8401	Init (Initialization)	Start inhibit of the function block is activated. Reset is required. Ready = TRUE S_OutControl = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = FALSE
8802	Safe	Safety function is triggered. Ready = TRUE S_OutControl = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8404	Lock	Safety function start inhibit is activated. Reset is required. Ready = TRUE S_OutControl = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = FALSE
8006	Output Disable	Process control is not activated. Ready = TRUE S_OutControl = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8000	Output Enable	Process control is activated and safety function is enabled. Ready = TRUE S_OutControl = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

11.10.2.2.20 SF_Override

SF_Override overrides the muting function. It allows product movement on the production line even if the muting function is aborted due to a fault. This function block is applicable only when used with the SF_MutingXXX instruction.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_Override	Override muting function	FB		

- **Related variables**

  Variables

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.
	S_AOPD_In	SAFEBOOL	TRUE, FALSE	FALSE	A OSSD signal from AOPD, indicating safety request input. FALSE: Protective field is interrupted. TRUE: Protective field is not interrupted.
	S_Muting_AOPD_Out	SAFEBOOL	TRUE, FALSE	FALSE	FALSE/TRUE: Depends on the status of the safety-related output S_AOPD_Out of the previous muting function block.
	MutingError	BOOL	TRUE, FALSE	FALSE	Fault output of the previous connected muting function block. FALSE: No fault occurs in the muting process. TRUE: A fault occurs in the muting process.
	MutingSwitch11	BOOL	TRUE, FALSE	FALSE	Status of the muting sensor signal at the MutingSwitch11 input of the previous muting function block. FALSE: Muting sensor 11 is not activated. TRUE: Muting sensor 11 is activated.
	MutingSwitch12	BOOL	TRUE, FALSE	FALSE	Status of the muting sensor signal at the MutingSwitch12 input of the previous muting function block. FALSE: Muting sensor 12 is not activated. TRUE: Muting sensor 12 is activated.
	MutingSwitch21	BOOL	TRUE, FALSE	FALSE	Status of the muting sensor signal at the MutingSwitch21 input of the previous muting function block. FALSE: Muting sensor 21 is not activated. TRUE: Muting sensor 21 is activated.

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
	MutingSwitch22	BOOL	TRUE, FALSE	FALSE	Status of the muting sensor signal at the MutingSwitch22 input of the previous muting function block. FALSE: Muting sensor 22 is not activated. TRUE: Muting sensor 22 is activated.
	MaxOverrideTime	TIME	Follows data type	T#0ms	Ranges from 0 to 10 minutes. Indicates the total time of the override process. Timer starts when the override start condition becomes available. The timer continues running even after all muting sensors are no longer active, until the muting process ends.
	S_StartStopOverride	SAFEBOOL	TRUE, FALSE	FALSE	Signal to start and stop the override function. A rising edge is required to start the override function. TRUE: If all override conditions are met, the override process begins. At this point, the timer for MaxOverrideTime also starts. FALSE: The override process stops. The timer for MaxOverrideTime continues running until the muting process ends.
	Reset	BOOL	TRUE, FALSE	FALSE	Manual reset
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	S_AOPD_Out	SAFEBOOL	TRUE, FALSE	FALSE	Safety-related output Indicates the status of the muted protective device or override signal. FALSE: The AOPD protective field is interrupted and the muting function is not activated, or the override function is not activated. TRUE: The AOPD protective field is not interrupted or the muting function is activated, or the override function is activated.

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
	OverridePossible	BOOL	TRUE, FALSE	FALSE	Status signal indicating whether the override function can be activated. FALSE: The override function cannot be activated. TRUE: The override function can be activated.
	OverrideActive	BOOL	TRUE, FALSE	FALSE	Indicates the status of the override process. FALSE: The override function is not activated. TRUE: The override function is activated.
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	ResetRequest	BOOL	TRUE, FALSE	FALSE	Flag indicating whether a reset is required.
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**
  - The override function for manual operation may be required to remove obstructions generated during the muting process from the safety area. If the override function is activated, stop requests issued by safety devices will not take effect. The function block SF_Override can only be used in combination with SF_MutingXXX. In the application, the SF_MutingXXX function block must be processed first, followed by the SF_Override function block.



The output faults and DiagCode from the connected muting function block will not be transmitted to the output faults and DiagCode of the SF_Override function block.

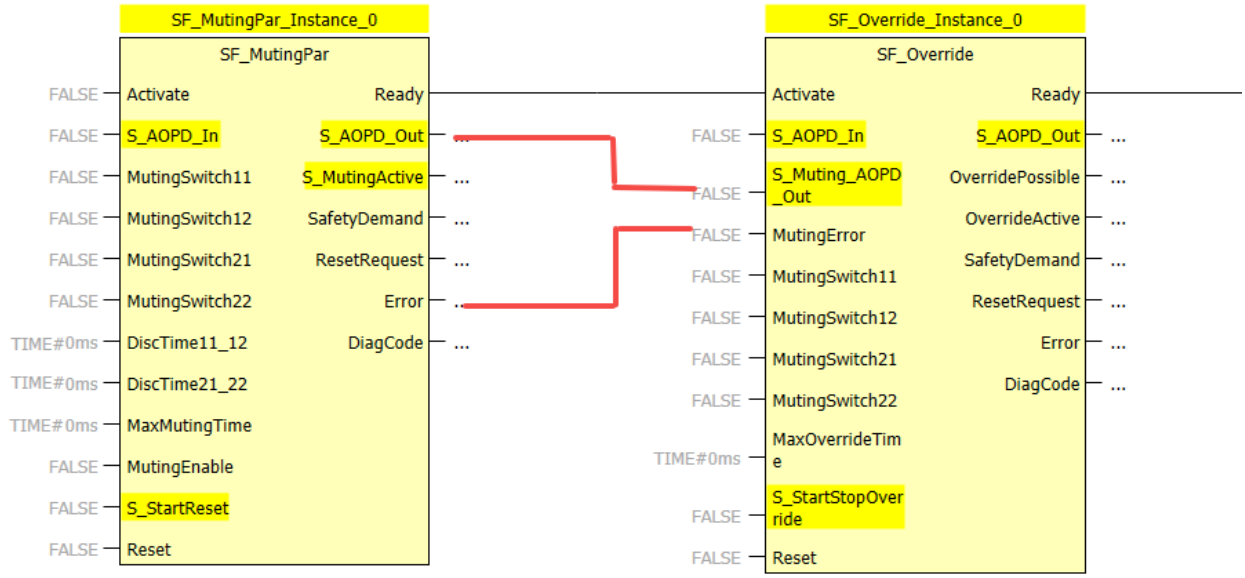


Figure 11-10-2-2-20-1 Combination of SF_Override and SF_Muting_Par with four sensors

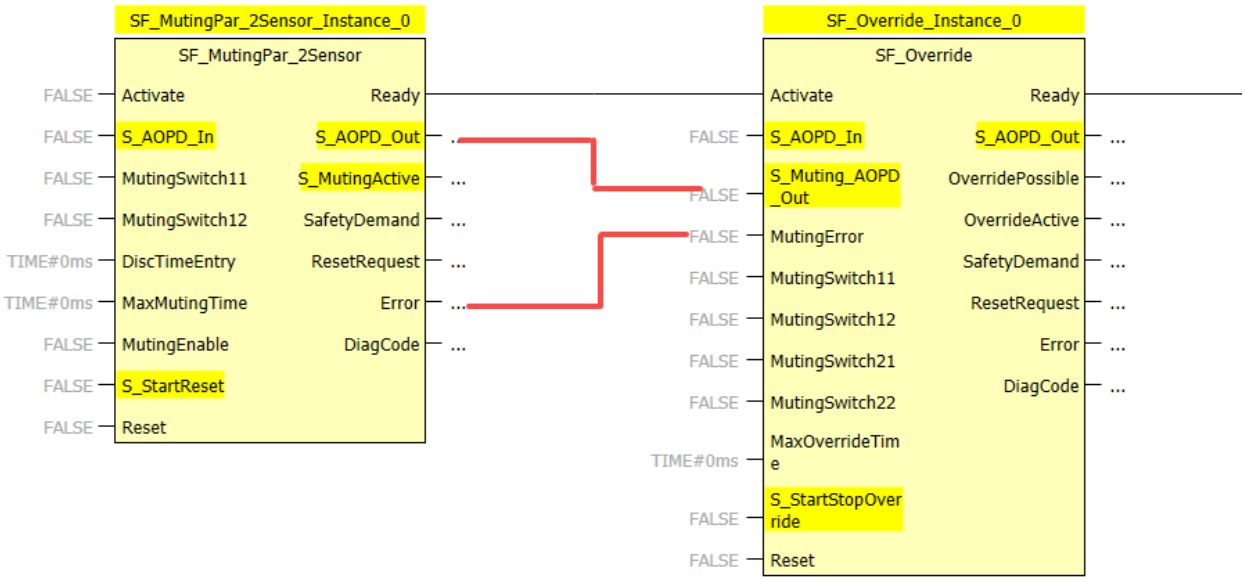


Figure 11-10-2-2-20-2 Combination of SF_Override and SF_Muting_Seq with two sensors

- The output parameter S_AOPD_Out of this function block will be set to TRUE under the following conditions:
  - The preceding muting function reports a fault.
  - A TRUE value of the input parameter S_StartStopOverride is triggered at the rising edge and remains TRUE.
  - The protective device (e.g., light curtain) is interrupted and/or at least one muting sensor is blocked.
- The output parameter S_AOPD_Out of this function block will be set to FALSE under the following conditions:
  - All muting sensors are "cleared" and the protective device (e.g., light curtain) is not interrupted.
  - The override time has exceeded the value set by MaxOverrideTime.
  - The signal of the input parameter S_StartStopOverride is FALSE.

- **Status transition diagram**

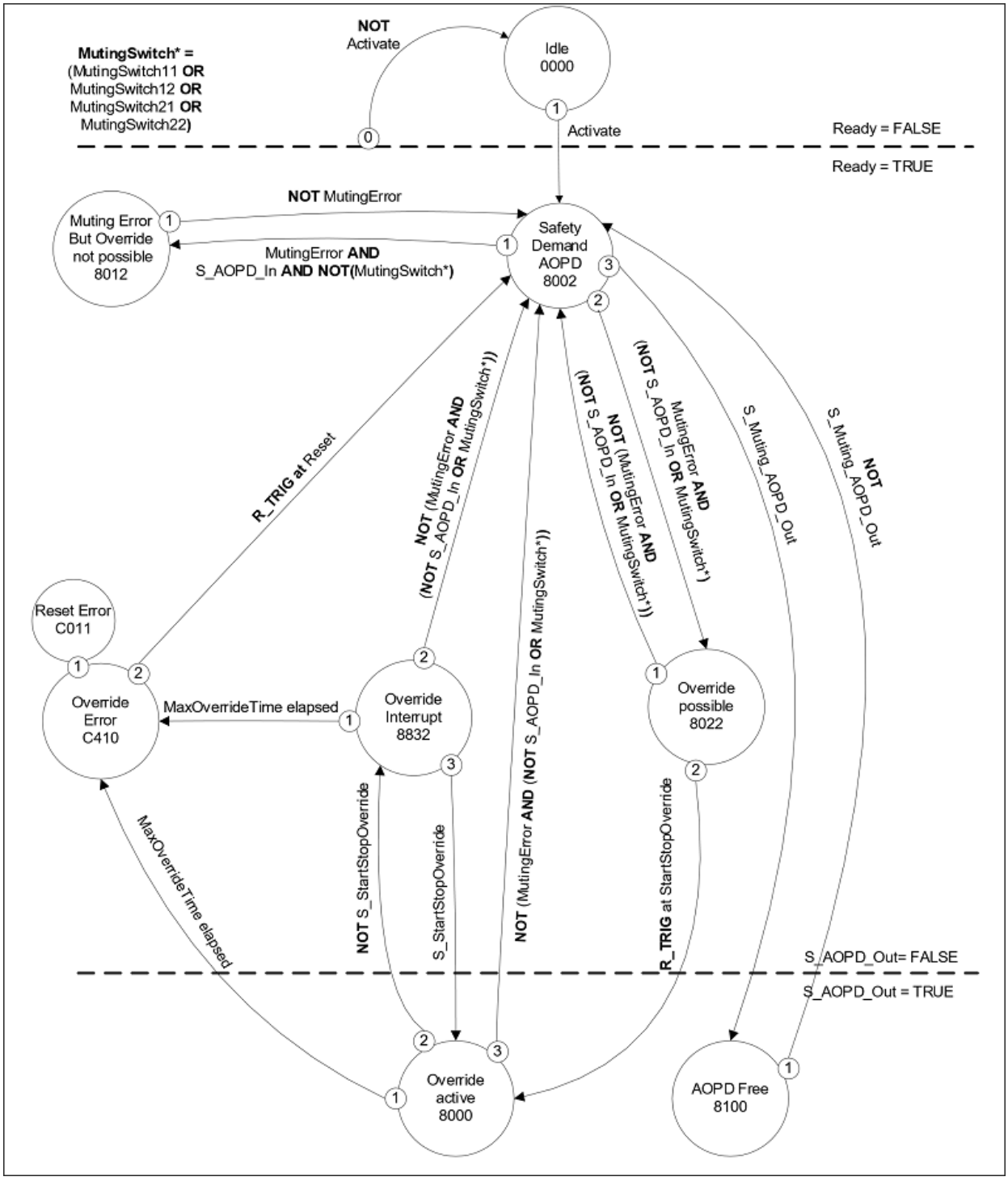
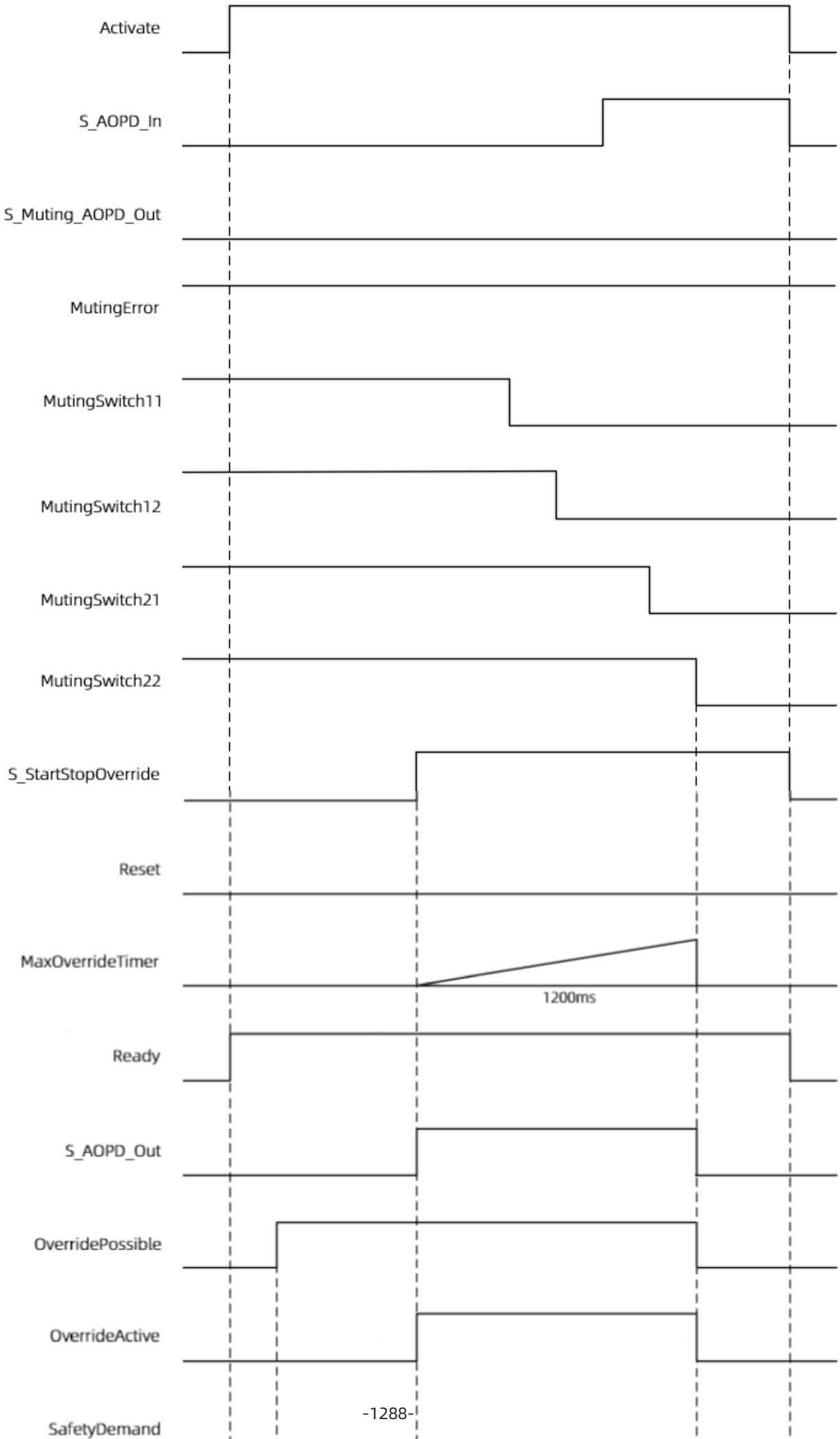


Figure 11-10-2-2-20-3 Status transition diagram for SF_Override

- This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.
- The transition from any status to the idle status when Activate = FALSE is not indicated, but the priority (0) for transitioning to the idle status is the highest.

- **Sequence diagram**


Activate
S_AOPD_In
S_Muting_AOPD_Out
MutingError
MutingSwitch11
MutingSwitch12
MutingSwitch21
MutingSwitch22
S_StartStopOverride
Reset
MaxOverrideTimer
1200ms
Ready
S_AOPD_Out
OverridePossible
OverrideActive
SafetyDemand

- **Fault detection**

  The faulty status is triggered by the following conditions:

  - Detects that a TRUE value of Reset is triggered at the non-rising edge.
  - The maximum override time is exceeded.

- **Fault behavior**
  When a fault occurs, the Error output is set to TRUE, OverridePossible and OverrideActive are disabled, and the DiagCode output displays the relevant fault code. The fault must be reset by a rising edge trigger of the Reset input.

- **Function block-specific fault codes and status codes**

  - Fault code

DiagCode	Status Name	Description and Output
C011	Reset Error	A static TRUE value of Reset is detected after the function block is activated. Ready = TRUE S_AOPD_Out = FALSE OverridePossible = FALSE OverrideActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C410	Override Error	The maximum override time is exceeded. Ready = TRUE S_AOPD_Out = FALSE OverridePossible = FALSE OverrideActive = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = TRUE

  - Status code

DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_AOPD_Out = FALSE OverridePossible = FALSE OverrideActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8002	Safety Demand AOPD	The protective field is interrupted and the muting function is inactive, or the override function is inactive and the MaxOverrideTime timer will be reset. Ready = TRUE S_AOPD_Out = FALSE OverridePossible = FALSE OverrideActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8012	Muting Error but Override not possible	The previous connected muting function shows a fault, but the protective device (e.g., light curtain) is not interrupted and no muting sensor is blocked. Ready = TRUE S_AOPD_Out = FALSE OverridePossible = FALSE OverrideActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

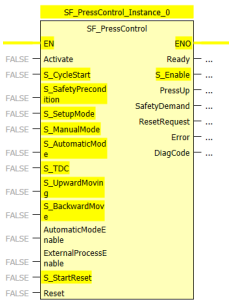
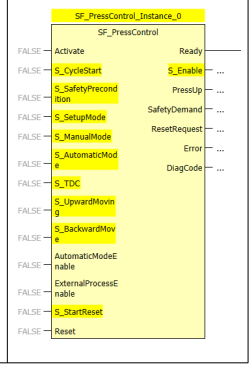
DiagCode	Status Name	Description and Output
8022	Override Possible	The previous connected muting function shows a fault, and the protective device (e.g., light curtain) is interrupted and/or at least a muting sensor is blocked. Ready = TRUE S_AOPD_Out = FALSE OverridePossible = TRUE OverrideActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8832	Override Interrupt	The override start signal is set to FALSE during the override process. The timer for MaxOverrideTime is still running. Ready = TRUE S_AOPD_Out = FALSE OverridePossible = TRUE OverrideActive = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8000	Override Active	The override function is activated and the timer for MaxOverrideTime starts running. Ready = TRUE S_AOPD_Out = TRUE OverridePossible = TRUE OverrideActive = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8100	AOPD Free	The S_AOPD_Out of the previous connected function block is TRUE. Ready = TRUE S_AOPD_Out = TRUE OverridePossible = FALSE OverrideActive = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

11.10.2.2.21 SF_PressControl

SF_PressControl controls the press.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_PressControl	Press control	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
	S_CycleStart	SAFEBOOL	TRUE, FALSE	FALSE	This parameter starts the cycle mode according to the operational mode. A transition from FALSE to TRUE starts the press cycle independently of the operational mode. Setting mode: A FALSE value sets the S_Enable signal to FALSE. Manual mode: A FALSE signal sets S_Enable to FALSE, as long as the slide is not moving upward (i.e., the S_UpwardMoving output of SF_CamMonitoring is FALSE). Automatic mode: A transition from FALSE to TRUE starts the cycle. Subsequent FALSE signal will not stop the process. To stop the process, set the input AutomaticModeEnable to FALSE. FALSE: No request to start the process. TRUE: Request to start the process.
	S_SafetyPrecondition	SAFEBOOL	TRUE, FALSE	FALSE	Signal from the previous safety function block (such as SF_EmergencyStop and SF_GuardMonitoring) FALSE: The safety-related prerequisite has not been met. TRUE: The safety-related prerequisite has been met.
	S_SetupMode	SAFEBOOL	TRUE, FALSE	FALSE	Setting mode is selected through the SF_ModeSelector function block. FALSE: Setting mode is not selected. TRUE: Setting mode is selected.
	S_ManualMode	SAFEBOOL	TRUE, FALSE	FALSE	Setting mode is selected through the SF_ModeSelector function block. FALSE: Manual mode is not selected. TRUE: Manual mode is selected.
	S_AutomaticMode	SAFEBOOL	TRUE, FALSE	FALSE	Automatic mode is selected through the SF_ModeSelector function block. FALSE: Automatic mode is not selected. TRUE: Automatic mode is selected.

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
	S_TDC	SAFEBOOL	TRUE, FALSE	FALSE	Signal from the previous safety function block "SF_CamMonitor". FALSE: The press is not in the TDC area. TRUE: The press is in the TDC area.
	S_UpwardMoving	SAFEBOOL	TRUE, FALSE	FALSE	Signal from the previous safety function block "SF_CamMonitor". FALSE: The press is not moving upward. TRUE: The press is moving upward.
	S_BackwardMove	SAFEBOOL	TRUE, FALSE	FALSE	Allows the slide to move backward. It applies only in setting mode via individual switch input. FALSE: Backward movement is not requested. TRUE: Backward movement is requested.
	AutomaticModeEnable	BOOL	TRUE, FALSE	FALSE	Control signal from the functional application, used to enable/disable the process in automatic mode at the next TDC. FALSE: Automatic mode is not enabled, or the cycle is completed at TDC. TRUE: Automatic mode is enabled.
	ExternalProcessEnable	BOOL	TRUE, FALSE	FALSE	Control signal from the functional application, used as the prerequisite to enable the process in all modes. It is typically not safety-critical. Restart can only be performed via CycleStart. FALSE: Request to set S_Enable to FALSE. TRUE: Request to set S_Enable to TRUE.
	S_StartReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to start reset.
	Reset	BOOL	TRUE, FALSE	FALSE	Manual reset
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	S_Enable	SAFEBOOL	TRUE, FALSE	FALSE	Safety-related output FALSE: Disable the connected valve. TRUE: Enable the connected valve.

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
	PressUp	BOOL	TRUE, FALSE	FALSE	Output signal for diagnostics. FALSE: The press is not moving upward. TRUE: The press is moving upward.
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	ResetRequest	BOOL	TRUE, FALSE	FALSE	Flag indicating whether a reset is required.
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**
  - This function block provides three modes for press control and is used with the function blocks "SF_ModeSelector" and "SF_CamMonitor".
  - This function block implements the single stroke function in manual and setting modes.
  - Only one of the three modes can be selected through the function block "SF_ModeSelector".
  - The optional modes include:
    - Manual mode

      In manual mode, the single stroke function is activated.

      - A stroke is initiated at the rising edge (0 -> 1) of S_CycleStart when inputs S_TDC, S_SafetyPrecondition, and ExternalProcessEnable are all TRUE.
      - After initiation, S_CycleStart must remain TRUE until the rising edge of UpwardMoving. Otherwise, S_Enable is set to FALSE.
      - If a stroke is initiated between the TDC area and BDC, and S_CycleStart is set to FALSE, the stroke can be re-initiated outside the TDC area.
        An overtravel fault prevents the initiation of a stroke.
      - When TDC zone is reached, the output S_Enable is set to FALSE. The stroke is fully completed, and the process can be restarted.
      - The cycle overtravel control performs monitoring, and only allows restarting from the TDC position after the rising edge of the TDC signal is detected.
      - Overtravel fault can only be reset in setting mode.

    - Setting mode
      - Setting mode (clockwise)
        - A stroke can be initiated from any position.
        - The single stroke function is activated.
          - For details, see "Manual mode".
          - Overtravel faults are for diagnostics only.

        - Muting (BDC signal changing from FALSE to TRUE) is not activated (S_CycleStart must be TRUE throughout the entire cycle).
    - Setting mode (counterclockwise)
        - A stroke can be initiated from any position.
        - The single stroke function is not activated.
        - The stroke automatically stops at the rising edge of S_TDC.
        - Movement can only be performed with the two-hand control device.
    - Automatic mode
        - A stroke can be initiated from any position.
        - The single stroke function is not activated.
        - Muting (BDC signal changing from FALSE to TRUE) is not activated.
        - If both S_SafetyPrecondition and AutomaticModeEnable are TRUE, the first stroke starts at the rising edge of S_CycleStart (S_CycleStart can be FALSE after the rising edge).
        - The stroke is stopped by setting S_SafetyPrecondition to FALSE or AutomaticModeEnable to FALSE.
        - If automatic mode is stopped while AutomaticModeEnable = FALSE, the cycle will continue until reaching TDC.
        - If S_SafetyPrecondition = FALSE, movement stops immediately.

    In all modes, if S_SafetyPrecondition = FALSE, movement stops immediately; if S_SafetyPrecondition is set to TRUE, a new press cycle can be triggered when Reset is triggered at the non-rising edge.

- **Status transition diagram**

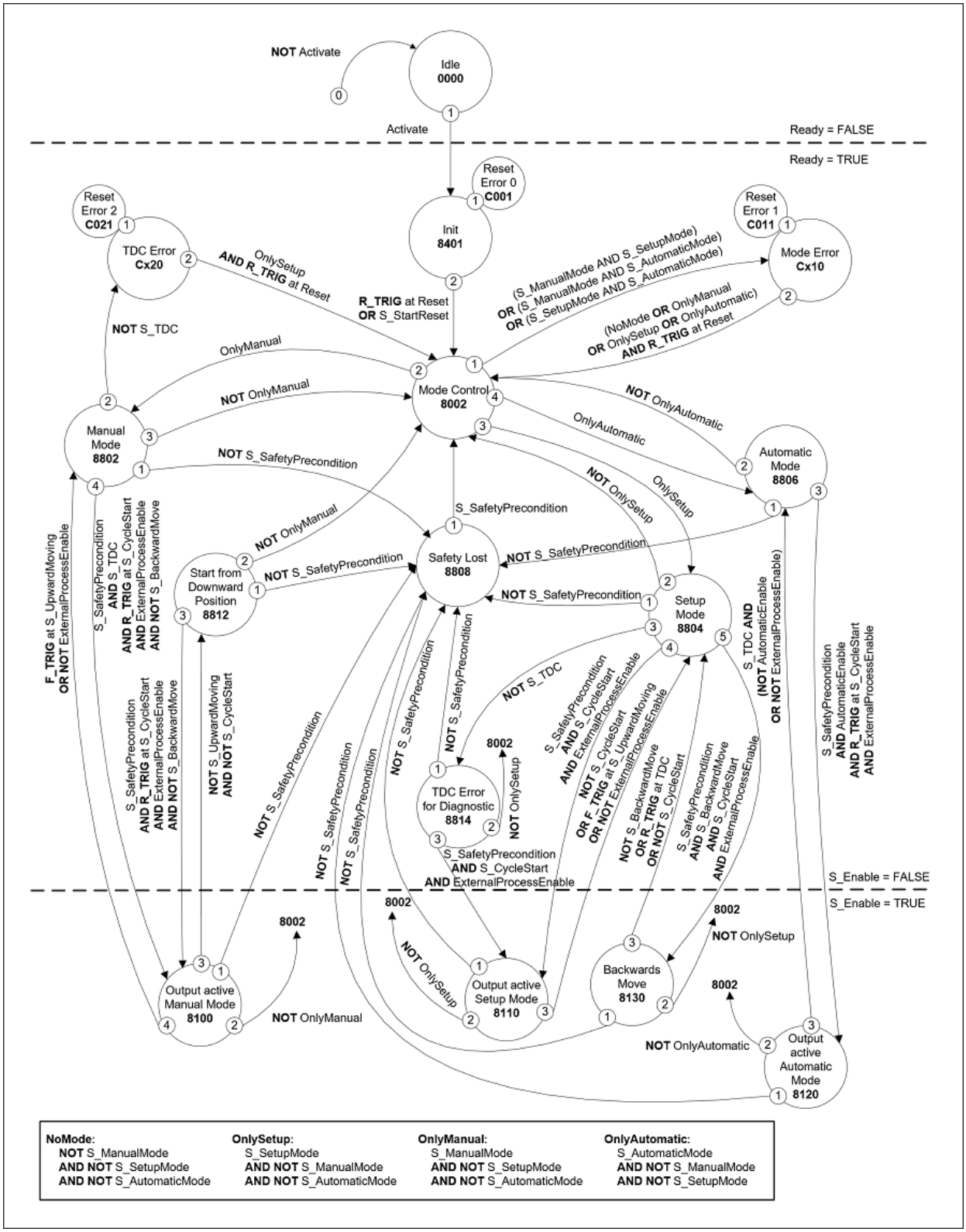
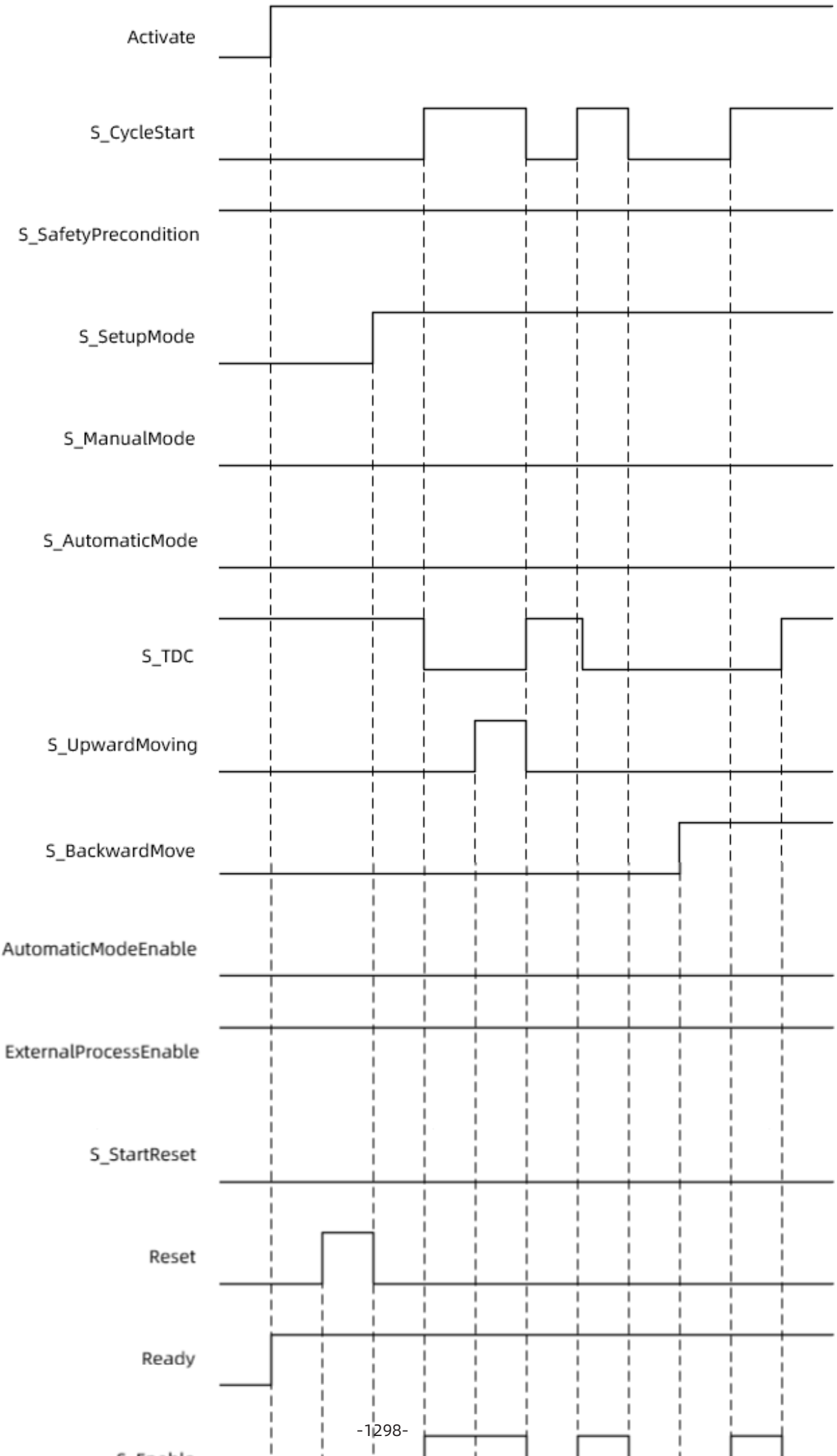


Figure 11-10-2-2-21-1 Status transition diagram for SF_PressControl

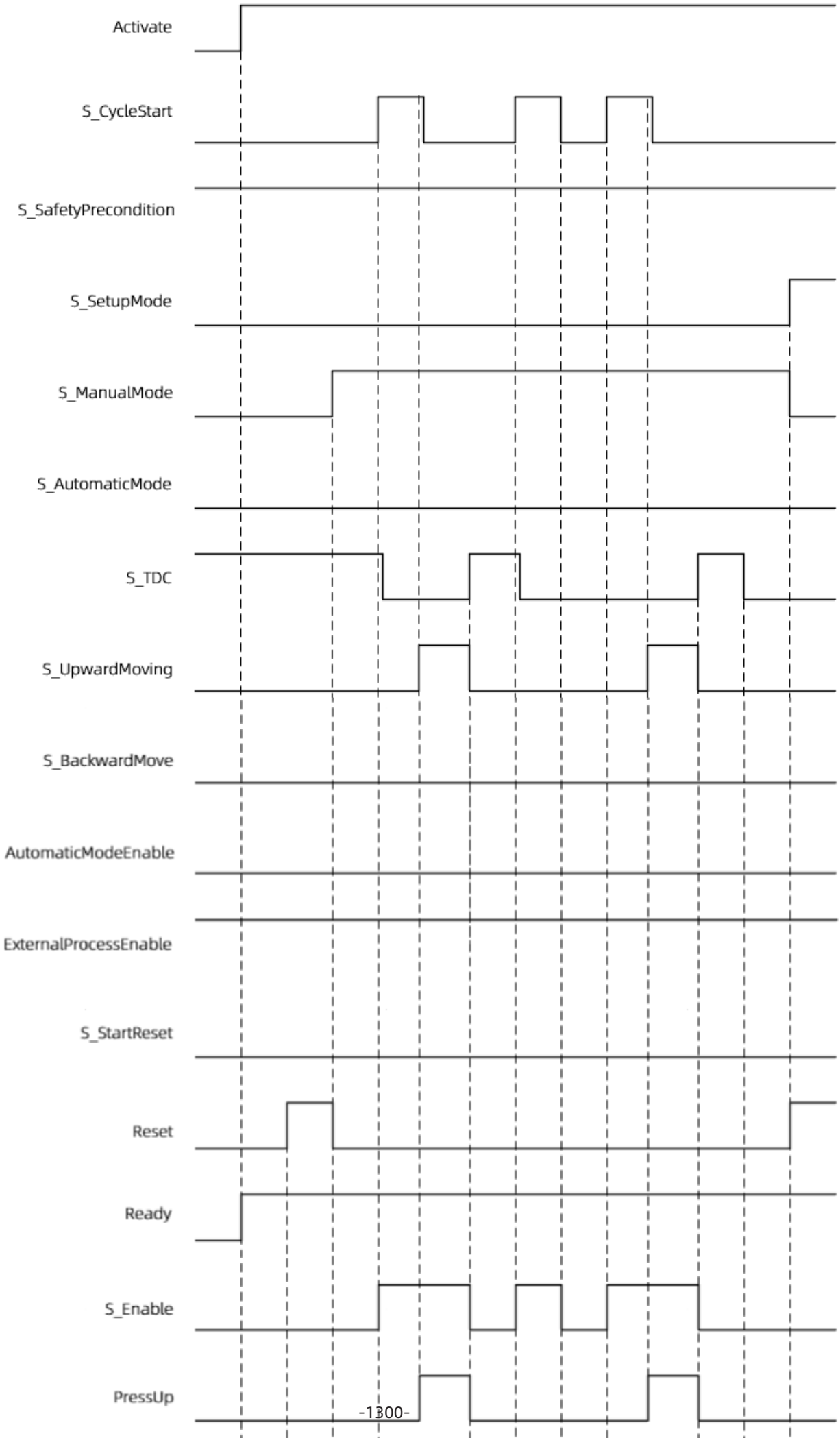
- This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.
- The transition from any status to the idle status when Activate = FALSE is not indicated, but the priority (0) for transitioning to the idle status is the highest.

- **Sequence diagram**

  Setting mode

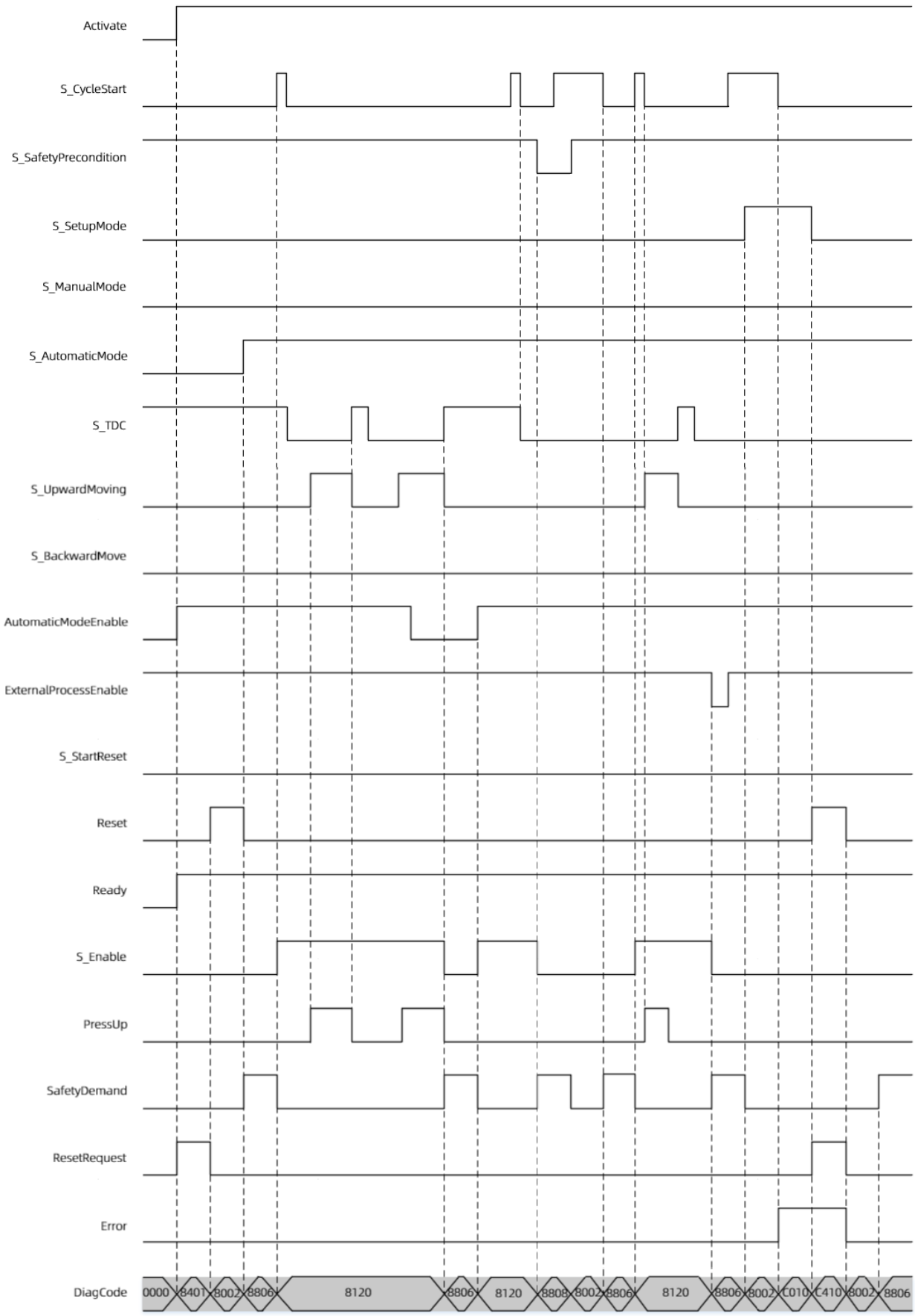

Activate
S_CycleStart
S_SafetyPrecondition
S_SetupMode
S_ManualMode
S_AutomaticMode
S_TDC
S_UpwardMoving
S_BackwardMove
AutomaticModeEnable
ExternalProcessEnable
S_StartReset
Reset
Ready

Manual mode


Activate
S_CycleStart
S_SafetyPrecondition
S_SetupMode
S_ManualMode
S_AutomaticMode
S_TDC
S_UpwardMoving
S_BackwardMove
AutomaticModeEnable
ExternalProcessEnable
S_StartReset
Reset
Ready
S_Enable
PressUp

1. Initial setup and startup.
2. Single stroke in manual mode.
3. Single stroke stopped between TDC and BDC, and restarting from the lower position.
4. Overtravel fault after the last stroke, reset in setting mode.

Automatic mode



- **Fault detection**

  The faulty status is triggered by the following conditions:

  - A static TRUE signal is detected at the Reset input.

- TDC is not activated after press cycle (overtravel fault).
- More than one mode has been selected.

- **Fault behavior**
  - When a fault occurs, the output S_Enable is set to FALSE. The DiagCode output indicates the relevant fault code, and the Error output is set to TRUE.
  - Restart is prohibited until the fault condition is cleared and the operator confirms the safety status via Reset.
- **Function block-specific fault codes and status codes**
  - Fault code

DiagCode	Status Name	Description and Output
C001	Reset Error 0	A static Reset input is detected in status 8401. Ready = TRUE S_Enable = FALSE PressUp = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
Cx10	Mode Error	More than one mode has been selected. If NoMode, OnlyManual, OnlySetup, or OnlyAutomatic is TRUE, the fault code is C410; otherwise, it is C010. • When the fault code is C410: Ready = TRUE S_Enable = FALSE PressUp = FALSE SafetyDemand = FALSE ResetRequest = TRUE Error = TRUE • When fault code is C010: Ready = TRUE S_Enable = FALSE PressUp = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
C011	Reset Error 1	A static Reset input is detected in status Cx10. Ready = TRUE S_Enable = FALSE PressUp = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
Cx20	TDC Error	TDC is not activated after press cycle. (Overtravel) If OnlySetup is TRUE, the fault code is C420; otherwise, it is C020. • When fault code is C420: Ready = TRUE S_Enable = FALSE PressUp = FALSE SafetyDemand = FALSE ResetRequest = TRUE Error = TRUE • When the fault code is C020: Ready = TRUE S_Enable = FALSE PressUp = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C021	Reset Error 2	A static Reset input is detected in status Cx20. Ready = TRUE S_Enable = FALSE PressUp = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

◦ Status code

DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_Enable = FALSE PressUp = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8410	Init (Initialization)	The function block is activated and initialized. Ready = TRUE S_Enable = FALSE PressUp = FALSE SafetyDemand = FALSE ResetRequest = TRUE Error = FALSE
8002	Mode Control	Analyzing the selected mode. Ready = TRUE S_Enable = FALSE PressUp = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8802	Manual Mode	Manual mode has been selected, and the function block is waiting for the press cycle to start. Ready = TRUE S_Enable = FALSE PressUp = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8804	Setup Mode	Setting mode has been selected, and the function block is waiting for the press cycle to start. Ready = TRUE S_Enable = FALSE PressUp = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8806	Automatic Mode	Automatic mode has been selected, and the function block is waiting for the press cycle to start. Ready = TRUE S_Enable = FALSE PressUp = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8808	Safety Lost	S_SafetyPreconditions is not TRUE. Ready = TRUE S_Enable = FALSE PressUp = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8812	Start from Downward Position	The press has passed TDC, and the S_CycleStart signal is lost before the press reaches BDC. Ready = TRUE S_Enable = FALSE PressUp = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE

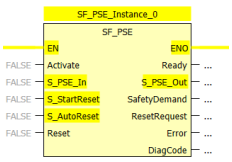
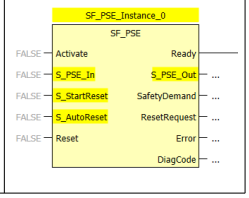
DiagCode	Status Name	Description and Output
8814	TDC Error for Diagnostic	This is not a fault in setting mode, used only for diagnostics. Ready = TRUE S_Enable = FALSE PressUp = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8100	Output Active Manual Mode	The press cycle is started in manual mode. Ready = TRUE S_Enable = TRUE PressUp = TRUE when the rising edge of S_UpwardMoving is detected. SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8110	Output Active Setup Mode	The press cycle is started in setting mode. Ready = TRUE S_Enable = TRUE PressUp = TRUE when the rising edge of S_UpwardMoving is detected. SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8120	Output Active Automatic Mode	The press cycle is started in automatic mode. Ready = TRUE S_Enable = TRUE PressUp = TRUE when the rising edge of S_UpwardMoving is detected. SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8130	Backwards Move	A press cycle is initiated in setting mode, and the press moves backwards. Ready = TRUE S_Enable = TRUE PressUp = TRUE when S_UpwardMoving is FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

11.10.2.2.22 SF_PSE

SF_PSE monitors pressure-sensitive equipment (PSE), such as safety mats, buffers, etc.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_PSE	Pressure-sensitive equipment	FB		

- **Related variables**

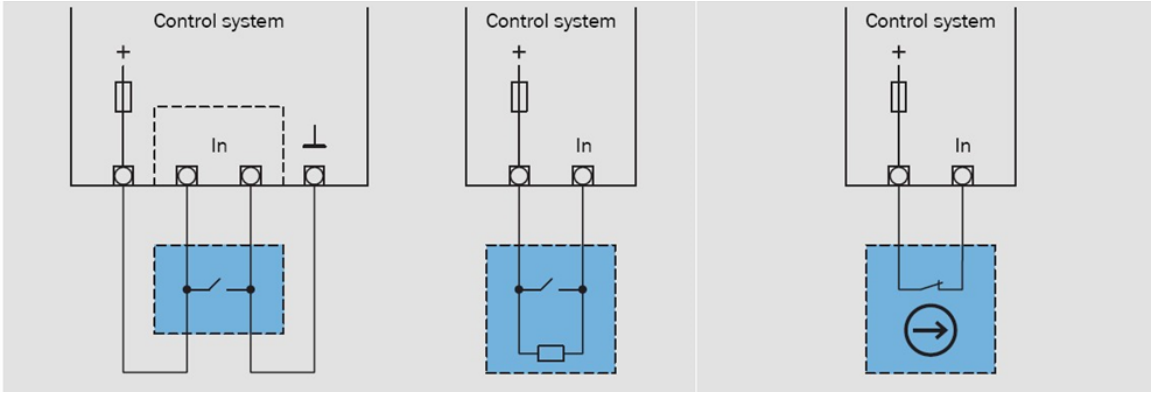
  Variables

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.
	S_PSE_In	SAFEBOOL	TRUE, FALSE	FALSE	Safety requirement input. FALSE: PSE is activated, and a safety-related requirement is input. TRUE: PSE is not activated, and no safety-related requirement is input.
	S_StartReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to start reset.
	S_AutoReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to automatically reset.
	Reset	BOOL	TRUE, FALSE	FALSE	Manual reset
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	S_PSE_Out	SAFEBOOL	TRUE, FALSE	FALSE	Safety-related output FALSE: Safety output is disabled. TRUE: Safety output is enabled.

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	ResetRequest	BOOL	TRUE, FALSE	FALSE	Flag indicating whether a reset is required.
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**
  - This function block is used to monitor pressure-sensitive equipment (PSE), such as safety mats and buffers.

Table11-10-2-2-22-1 Different PSE configurations in practice

Short-circuit forming designs (energise to trip principle)		**Positive opening contact design (deenergise to trip principle)**
4-wire variant	Resistance variant	
		
A short circuit is formed when the protective device is activated. For the 4-wire variant, one circuit is shorted (with a resistance in the range of several ohms). For the resistance variant, a change in the set resistance (in the range of kiloohms) is detected.		This design is more universal and advantageous. When the protective device is activated, a switch contact on the safety switch is opened. By special wiring, short circuits are avoided.

  - This function block requires a FALSE signal to activate the safety function. Therefore, as shown on the right of the figure above, PSE with positive opening contact design can be directly connected to a safety input device. However, the other two designs, as shown on the left of the figure above, require an evaluation unit to generate a suitable FALSE signal when the PSE is activated.
  - This function is identical to SF_EmergencyStop (except for two additional outputs: SafetyDemand and ResetRequest). Once the S_PSE_In input is set to FALSE, the S_PSE_Out is set to FALSE. The S_PSE_Out output signal is set to TRUE only when the S_PSE_In input is set to TRUE and a reset is performed. The enabling of the reset depends on the defined inputs S_StartReset, S_AutoReset and Reset.
    - If S_AutoReset = TRUE, reset is automatically confirmed.
    - If S_AutoReset = FALSE, reset is confirmed by a rising edge trigger of the Reset input.

- If S_StartReset = TRUE, reset is automatically confirmed upon the first startup of the PES.
- If S_StartReset = FALSE, reset is confirmed by a rising edge trigger of the Reset input.

S_StartReset and S_AutoReset inputs can only be activated when it is ensured that no hazardous conditions occur when the PES starts up.

◦ SF_PSE must be selected according to the product standards EN 1760-1, -2, and -3 and the performance level requirements of ISO 13849-1:2008.

- **Status transition diagram**

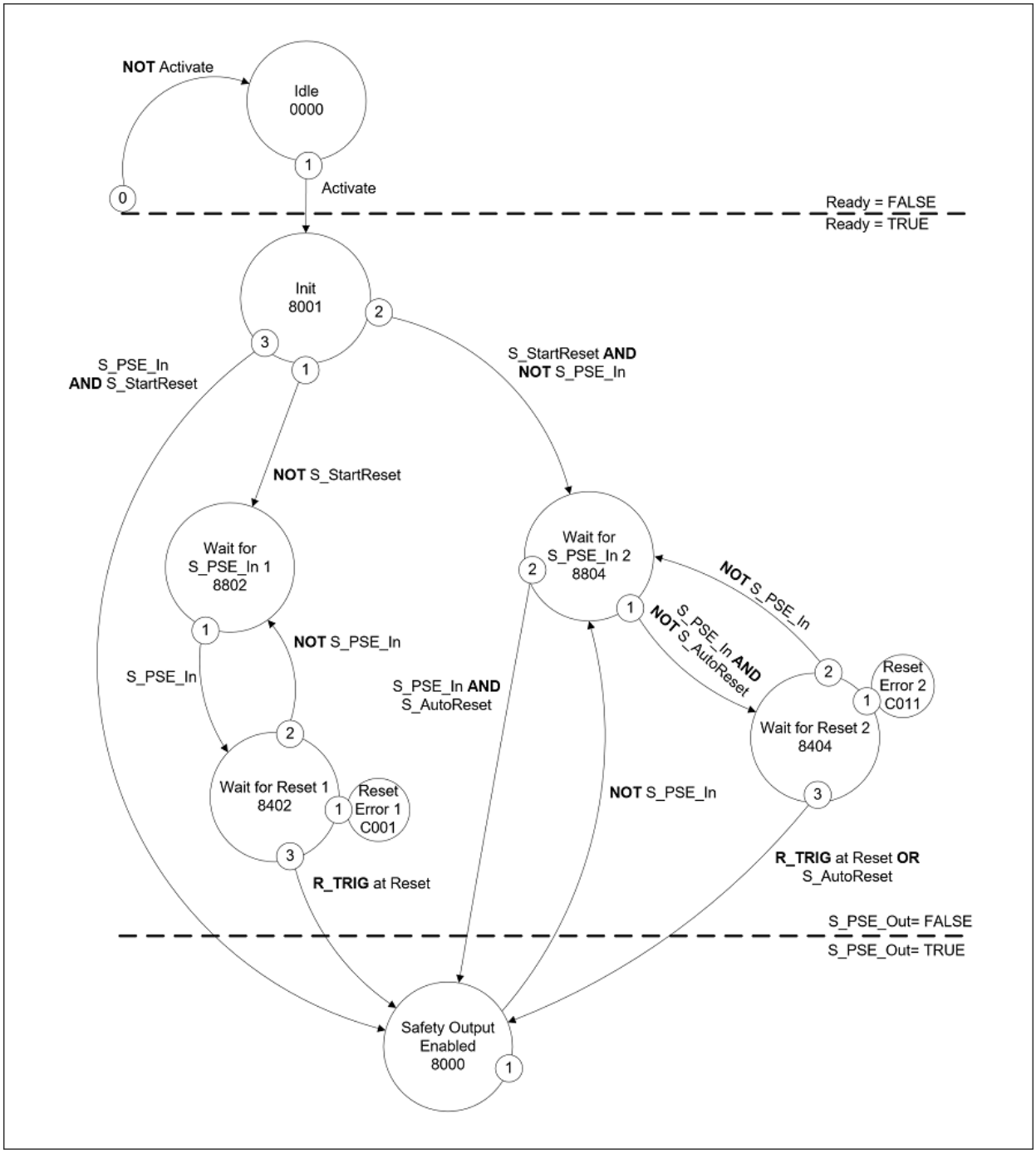


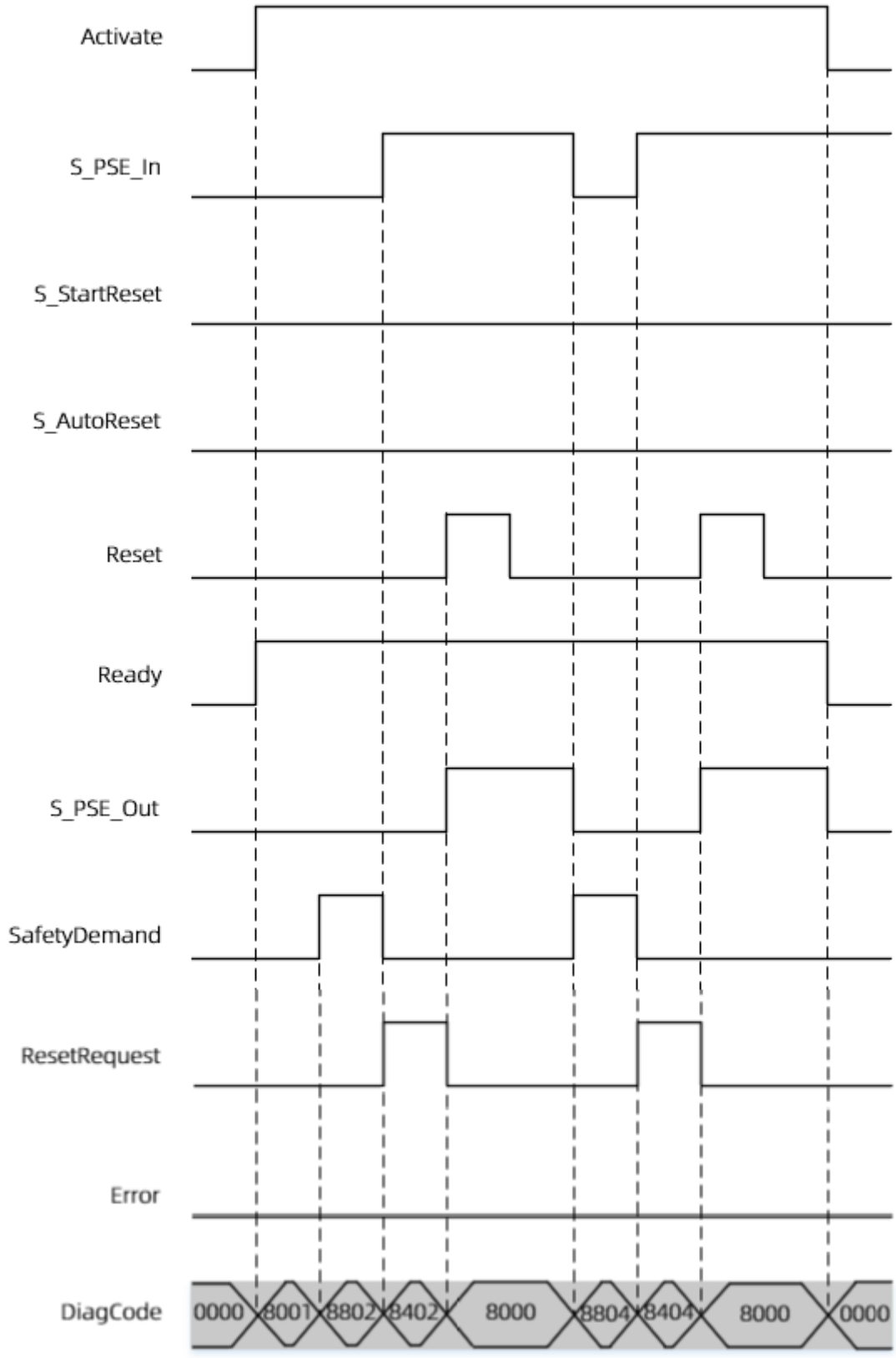
Figure 11-10-2-2-22-1 Status transition diagram for SF_PSE

◦ This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.

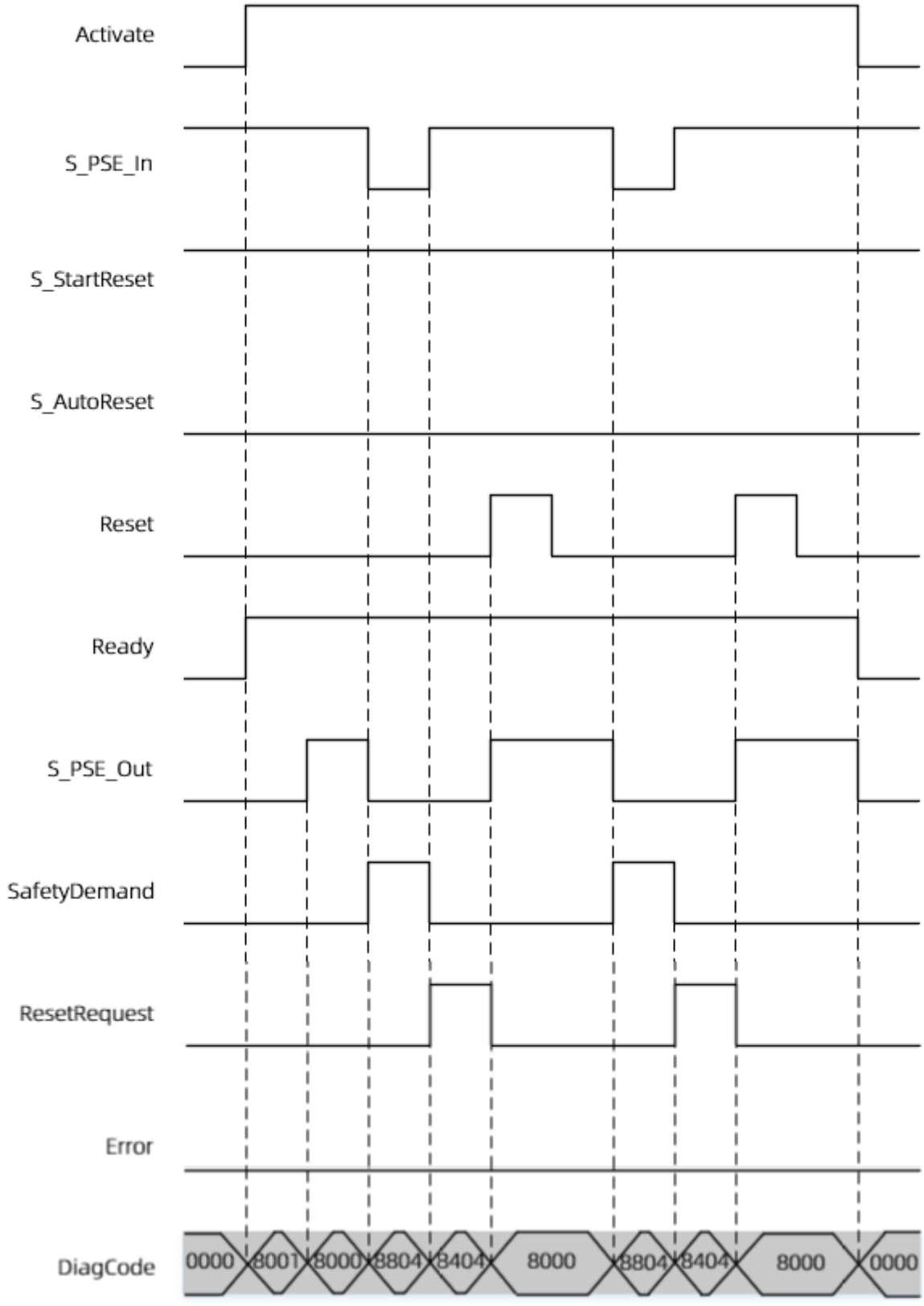
◦ The transition from any status to the idle status when Activate = FALSE is not indicated, but the priority (0) for transitioning to the idle status is the highest.

- **Sequence diagram**

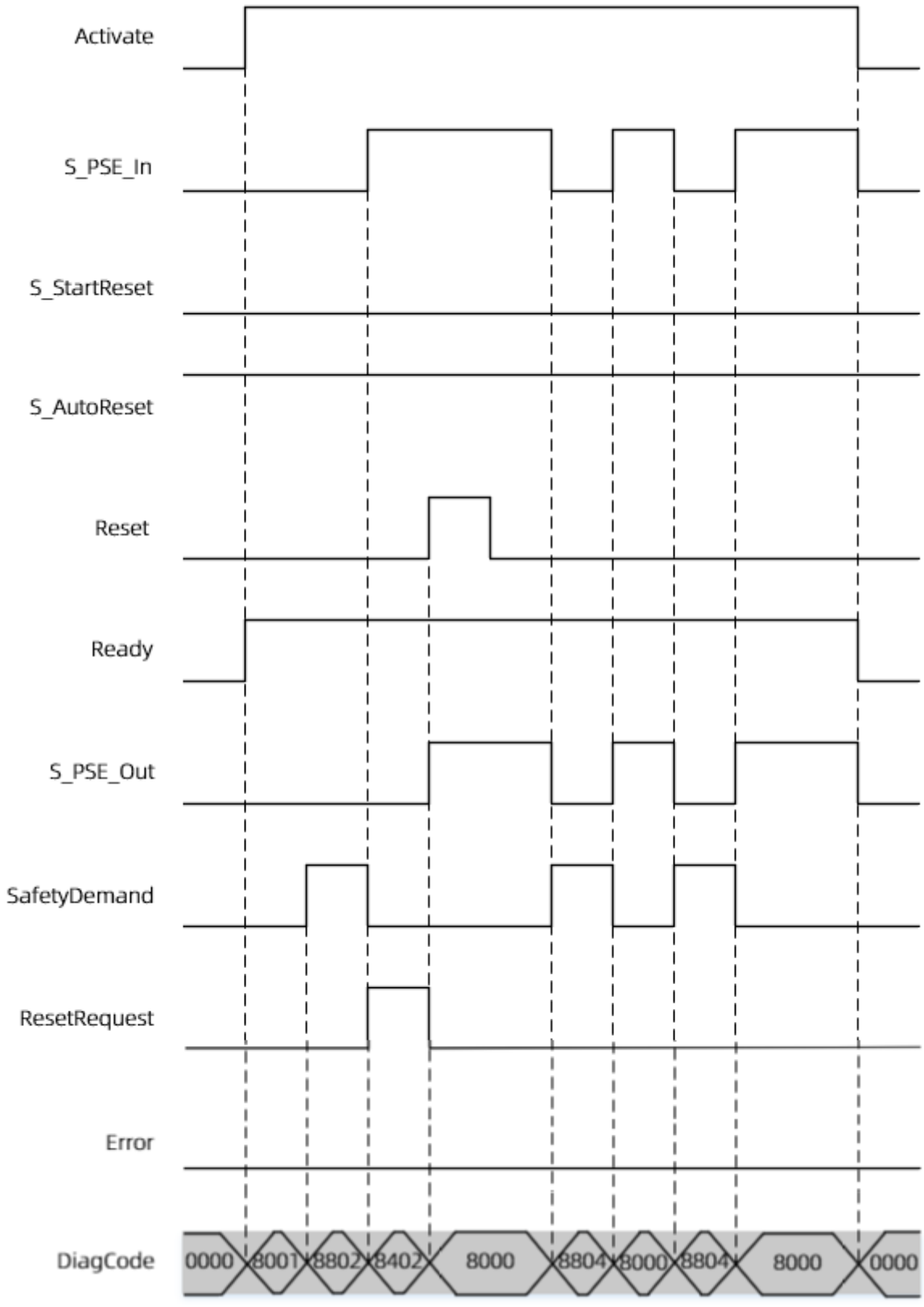
S_StartReset = FALSE, S_AutoReset = FALSE



S_StartReset = TRUE, S_AutoReset = FALSE



S_StartReset = FALSE, S_AutoReset = TRUE



- **Fault detection**

  The faulty status is triggered when a TRUE value of Reset is detected at the non-rising edge.

- **Fault behavior**

  S_PSE_Out is set to FALSE. If a static TRUE signal is detected at the Reset input, the DiagCode output displays the relevant fault code, and the Error output is set to TRUE. To exit the faulty status, Reset must be set to FALSE.

- **Function block-specific fault codes and status codes**
  - Fault code

DiagCode	Status Name	Description and Output
C001	Reset Error 1	Reset is set to TRUE while waiting for S_PSE_In = TRUE. Ready = TRUE S_PSE_Out = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C011	Reset Error 2	Reset is set to TRUE while waiting for S_PSE_In = TRUE. Ready = TRUE S_PSE_Out = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

  - Status code

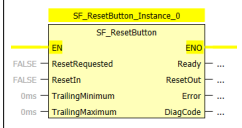
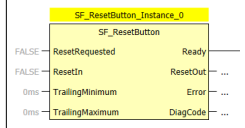
DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_PSE_Out = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8001	Init (Initialization)	Function block is enabled, Check if S_StartReset is required. Ready = TRUE S_PSE_Out = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8802	Wait for S_PSE_In (Wait for S_PSE 1)	Check if Reset is faulty, and wait for S_PSE_In = FALSE. Ready = TRUE S_PSE_Out = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8402	Wait for Reset 1	S_PSE_In = FALSE. Wait for the rising edge of Reset. Ready = TRUE S_PSE_Out = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = FALSE
8804	Wait for S_PSE_In (Wait for S_PSE 2)	The function block detected a safety request. Check if Reset is faulty, and wait for S_PSE_In = FALSE. Ready = TRUE S_PSE_Out = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8404	Wait for Reset 2	S_PSE_In = FALSE. Check S_AutoReset or wait for the rising edge of Reset. Ready = TRUE S_PSE_Out = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = FALSE
8000	Safety Output Enabled	S_PSE_In = FALSE. In function mode, S_PSE_Out = FALSE. Ready = TRUE S_PSE_Out = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

11.10.2.2.23 SF_ResetButton

SF_ResetButton adds a falling edge function to function blocks that originally only detect rising-edge triggers for reset. This function is compliant with the EN ISO 13849-1:2015 standard.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_ResetButton	Reset button	FB		

- **Related variables**

  Variables

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
Input	ResetRequested	BOOL	TRUE, FALSE	FALSE	Input connected to the ResetRequest output of the paired function block. TRUE: Reset requested. FALSE: No reset request / ResetIn not monitored.
	ResetIn	BOOL	TRUE, FALSE	FALSE	Input of the reset button. FALSE: Reset button released. TRUE: Reset button activated.
	TrailingMinimum	TIME	Follows data type	T#350ms	Valid at trailing edge. Minimum time the reset button must be activated. If the reset button is pressed for less than this time, the reset is ignored. Typical value is 350 ms. The absolute minimum is 100 ms, under which the task cycle time shall be less than or equal to TrailingMinimum / 2.
	TrailingMaximum	TIME	Follows data type	T#2s	Valid at trailing edge. Maximum time the reset button can be activated. Typical value is 2s. If the reset button is pressed for longer than this time, the reset is ignored.
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
	ResetOut	BOOL	TRUE, FALSE	FALSE	Pulse used to initiate the reset process. This pulse is generated after the falling edge. A rising-edge pulse output is generated first, which lasts for at least one cycle.
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**

  This function block is used in automated control systems to provide a safety-compliant reset mechanism. It allows the system to safely resume operation after an emergency stop or safety event through intentional and explicit user operations. It detects the rising and falling edges of the reset button, ensuring that the reset action is executed only when specific safety conditions are met, thus preventing unintended operations and accidental startups, and safeguarding personnel and equipment.

Fault	No request	Request enabled	↓	↑↓
Switch jam		OK	OK	OK
Abnormal voltage-high-level latching			OK	OK

OK Module/signal in pre-operational status

↑ Rising-edge triggering

↓ Falling-edge triggering

↑↓ Reset on rising edge and fault detection on falling edge

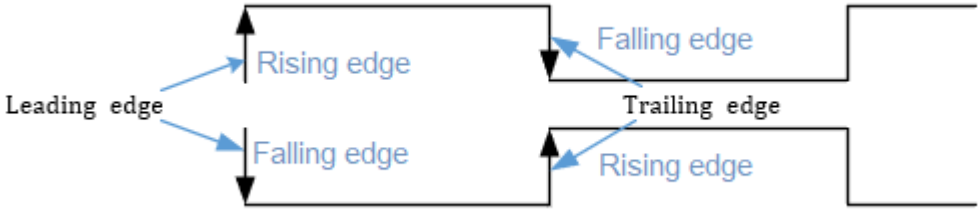


Figure 11-10-2-2-23-1 Rising edge, falling edge, and trailing edge

- **Status transition diagram**

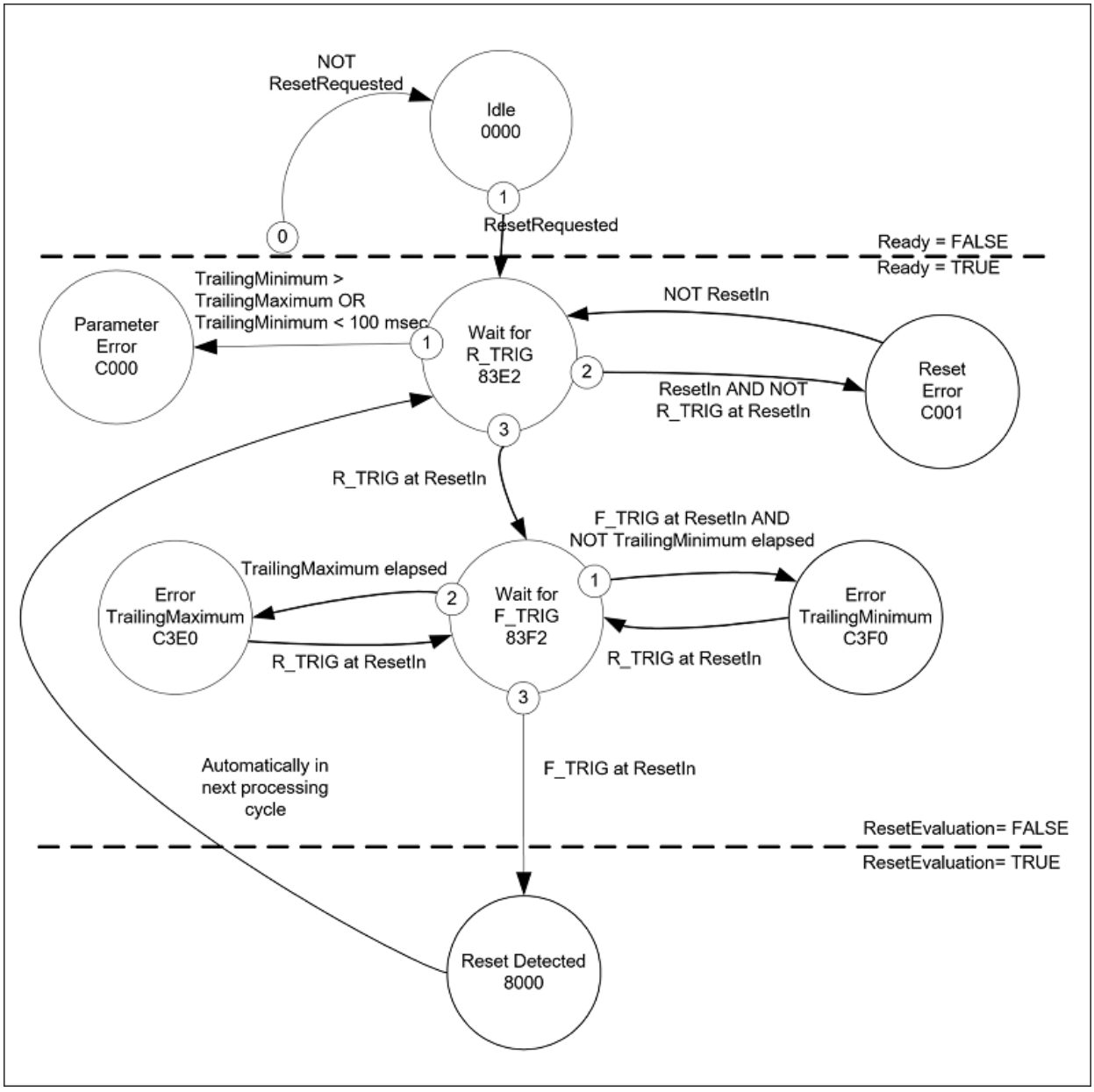
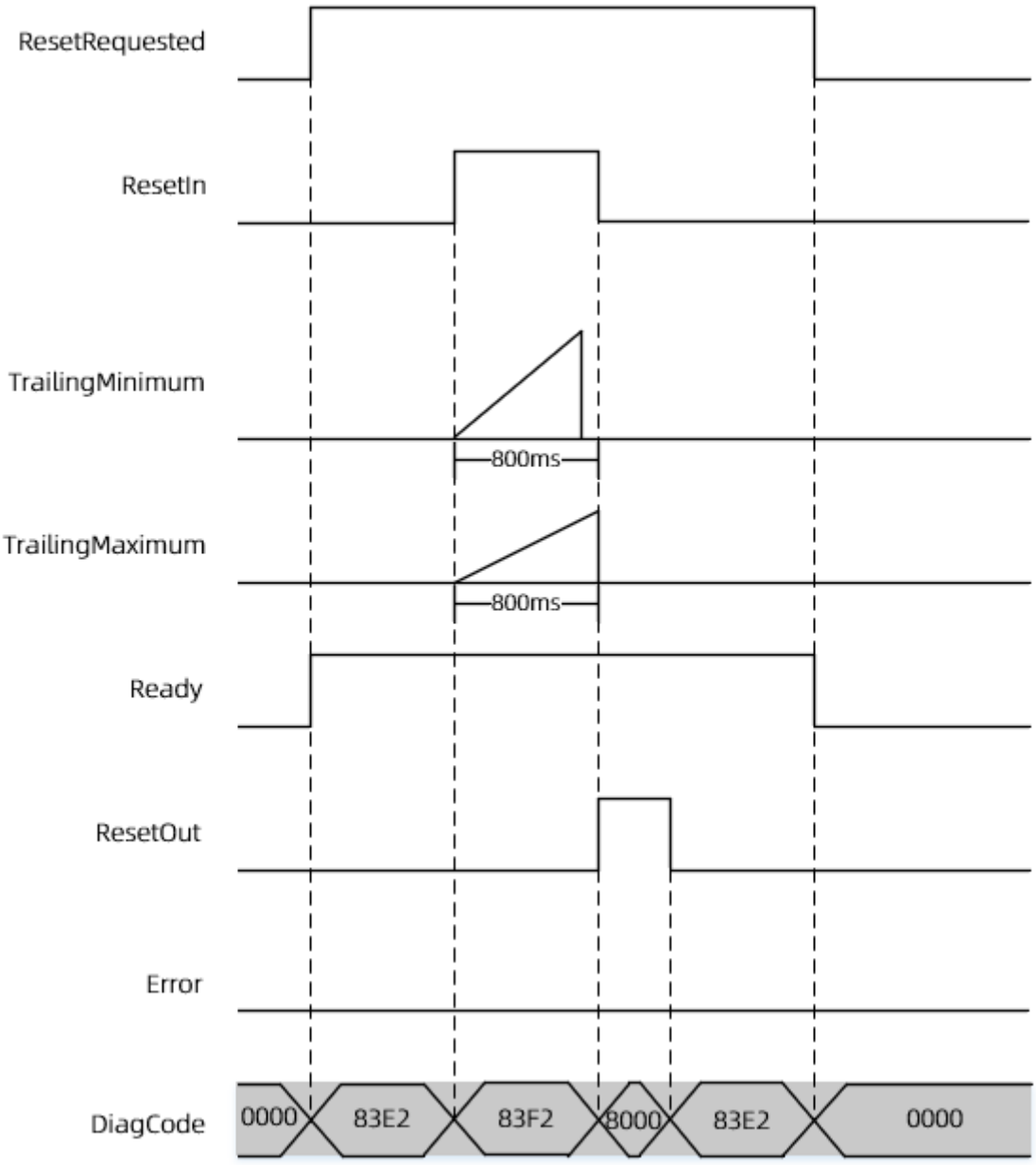


Figure 11-10-2-2-23-2 Status transition diagram for SF_ResetButton

  - This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.
  - The transition from any status to the idle status when ResetRequested = FALSE is not indicated, but the priority (0) for transitioning to the idle status is the highest.

- **Sequence diagram**



- **Fault detection**

  The faulty status is triggered by the following conditions:

  - ResetIn is already TRUE when ResetRequested becomes TRUE.
  - ResetRequested and ResetIn are TRUE, and the reset button is pressed for a time less than TrailingMinimum or longer than TrailingMaximum.

- **Fault behavior**

  When the ResetIn input is a static TRUE signal, the DiagCode output indicates the corresponding fault code, and the Error output is set to TRUE.

- **Function block-specific fault codes and status codes**

  - Fault code

DiagCode	Status Name	Description and Output
C000	Parameter Error	TrailingMinimum > TrailingMaximum or TrailingMinimum < 100 ms. Ready = TRUE ResetOut = FALSE Error = TRUE
C001	Reset Error	ResetIn is triggered at the non-rising edge. Ready = TRUE ResetOut = FALSE Error = TRUE
C3E0	Error Trailing Maximum	The maximum time for the trailing edge has been exceeded before the falling edge of ResetIn is detected. Ready = TRUE ResetOut = FALSE Error = TRUE
C3F0	Error Trailing Minimum	The minimum time for the trailing edge has not been exceeded before the falling edge of ResetIn is detected. Ready = TRUE ResetOut = FALSE Error = TRUE

◦ Status code

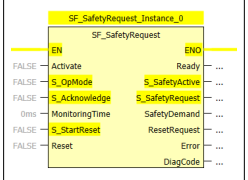
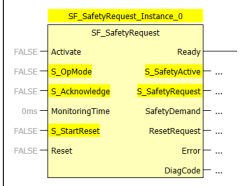
DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE ResetOut = FALSE Error = FALSE
83E2	Wait for R_TRIG (Wait for rising edge)	Function block is enabled, and is waiting for rising edge of ResetIn. Ready = TRUE ResetOut = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
83F2	Wait for F_TRIG (waiting for falling edge)	Function block is enabled, and is waiting for falling edge of ResetIn. Ready = TRUE ResetOut = FALSE Error = FALSE
8000	Reset Detected	Valid reset action is detected. This status is valid for at least one cycle and will automatically transition to the Wait for R_TRIG status. Ready = TRUE ResetOut = TRUE Error = FALSE

11.10.2.2.24 SF_SafetyRequest

SF_SafetyRequest targets safety-enabled actuators (such as drives and valves) to request and monitor their safety statuses.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_SafetyRequest	Safety request	FB		

- **Related variables**

  Variables

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.
	S_OpMode	SAFEBOOL	TRUE, FALSE	FALSE	Requested mode of the general-purpose safety actuator. FALSE: Request safety mode. TRUE: Request operational mode.
	S_Acknowledge	SAFEBOOL	TRUE, FALSE	FALSE	Confirm that the general-purpose actuator is in safety status. FALSE: Operational mode (non-safe). TRUE: Safe mode.

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
	MonitoringTime	TIME	Follows data type	T#0ms	Response time between the safety function request (S_OpMode set to FALSE) and actuator confirmation (S_Acknowledge set to TRUE).
	S_StartReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to start reset.
	Reset	BOOL	TRUE, FALSE	FALSE	Manual reset
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	S_SafetyActive	SAFEBOOL	TRUE, FALSE	FALSE	Confirmation of safety status. FALSE: Non-safety status. TRUE: Safety status.
	S_SafetyRequest	SAFEBOOL	TRUE, FALSE	FALSE	Request to set the actuator in safety status. FALSE: Request safety status. TRUE: Request operational mode.
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	ResetRequest	BOOL	TRUE, FALSE	FALSE	Flag indicating whether a reset is required.
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**
  - This function block provides an interface between safety-related systems and general-purpose actuators, so that the safety-related functions of actuators can be called through the application. The actuator's safety status is controlled by only two binary signals: one for requesting and one for receiving confirmation.
    Safety functions are implemented by the actuators. Therefore, this function block is responsible only for initiating the request, monitoring the request, and setting the output when the actuator confirms the safety status (indicated by the S_SafetyActive output).
  - Parameter definition:
    This function block does not define parameters dedicated to general-purpose actuators; such parameters shall be defined within the actuators themselves.
    The function block is used to switch the general-purpose actuator from the operational mode to the safety status.
  - Additional input function:

    The S_StartReset input provides an automatic cold start function. Setting it to FALSE ensures compatibility with the original function.

- **Status transition diagram**

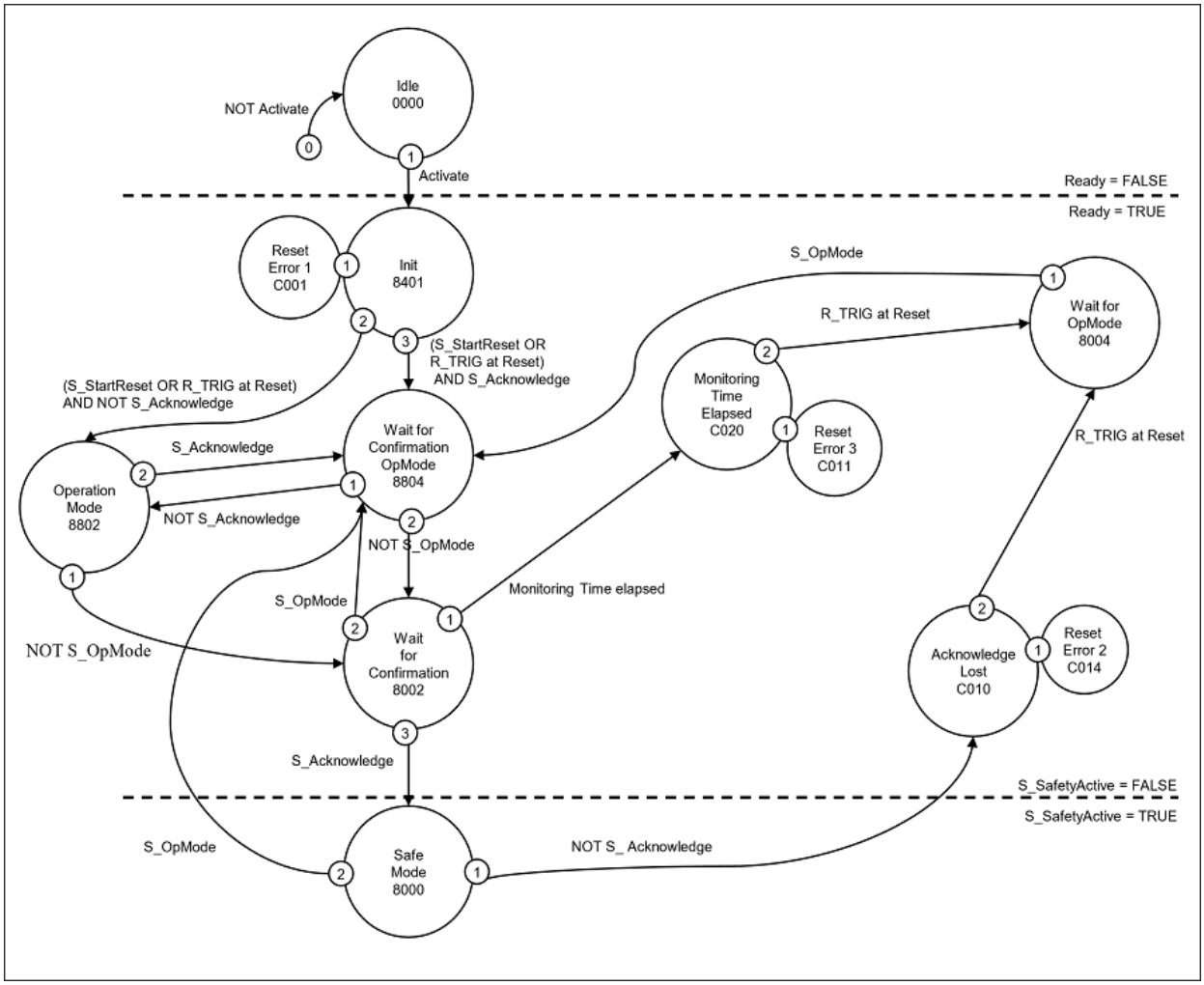
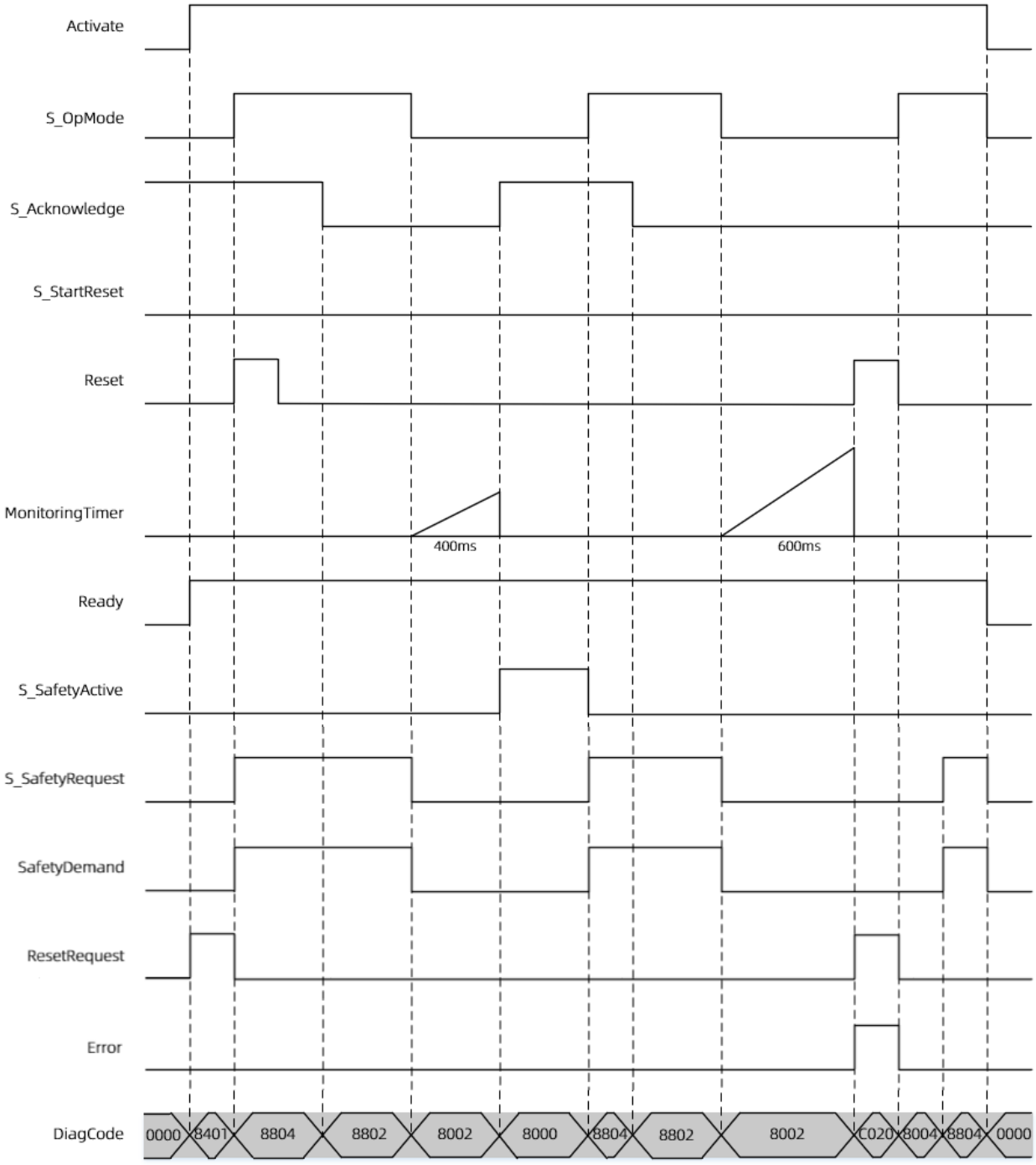


Figure 11-10-2-2-24-1 State transition diagram for SF_SafetyRequest

  - This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.
  - The transition from any status to the idle status when Activate = FALSE is not indicated, but the priority (0) for transitioning to the idle status is the highest.

- **Sequence diagram**



- **Fault detection**
  - Detects that the actuator fails to enter the safety status within the monitoring time.
  - Detects that no confirmation signal is sent before the request becomes invalid.
  - Detects that a TRUE value of Reset is triggered at the non-rising edge.
- **Fault behavior**

  When a fault occurs, the S_SafetyActive output is set to FALSE.

  The fault must be reset by a rising edge trigger of the Reset input. To continue operating the function block after this reset, set the S_OpMode request to TRUE or set S_Acknowledge to TRUE.
- **Function block-specific fault codes and status codes**
  - Fault code

DiagCode	Status Name	Description and Output
C010	Acknowledge Lost	The confirmation signal is lost in the safety status. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = TRUE
C020	MonitoringTime Elapsed	S_OpMode request cannot be completed within the monitoring time. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = TRUE
C001	Reset Error 1	In the 8401 status, a TRUE value of Reset is detected at the non-rising edge. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C014	Reset Error 2	In the C010 status, a TRUE value of Reset is detected at the non-rising edge. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
C011	Reset Error 3	In the C020 status, a TRUE value of Reset is detected at the non-rising edge. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

◦ Status code

DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_SafetyActive = FALSE S_SafetyRequest = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8000	Safe Mode	The actuator is in safe mode. Ready = TRUE S_SafetyActive = TRUE S_SafetyRequest = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

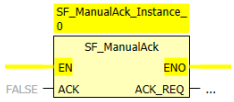
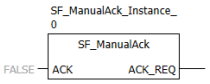
DiagCode	Status Name	Description and Output
8401	Init (Initialization)	The function block enters the initialization status when it is activated or when a TRUE value of the Reset input is detected at the rising edge. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = FALSE
8802	Operation Mode	Indicates the operational mode without safe mode confirmation. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = TRUE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8804	Wait for Confirmation OpMode	Indicates the operational mode with safe mode confirmation. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = TRUE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8002	Wait for Confirmation	The function block is waiting for confirmation from the actuator (system interface). Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8004	Wait for OpMode	Faults have been removed. However, the function block can only be continued with S_OpMode or S_Acknowledge set to TRUE. Ready = TRUE S_SafetyActive = FALSE S_SafetyRequest = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

11.10.2.2.25 SF_ManualAck

SF_ManualAck is a function block used for manual confirmation.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_ManualAck	Safety confirmation	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
Input	ACK	BOOL	TRUE, FALSE	FALSE	Manual confirmation operation. TRUE: Manual confirmation operation performed. FALSE: Manual confirmation operation not performed.
Output	ACK_REQ	BOOL	TRUE, FALSE	FALSE	Safety confirmation request. TRUE: Safety confirmation request exists. FALSE: No safety confirmation request.

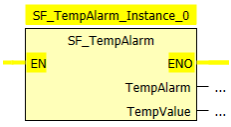
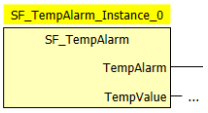
- **Function description**

  This function block contains the input parameter "ACK" for confirmation operation and output parameter "ACK_REQ" for confirmation request. "ACK_REQ" being TRUE indicates a manual confirmation request is performed. It indicates that users confirm manually by writing TRUE to "ACK", with the TRUE status lasting for more than 1s and less than 10s. The safety PLC identifies the manual confirmation operation. When the confirmation is successful, ACK_REQ becomes FALSE.

11.10.2.2.26 SF_TempAlarm

SF_TempAlarm is a function block used for temperature alarm.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_TempAlarm	Temperature alarm	FB		

- **Related variables**

  Variables

Input/Output	Variable	Data Type	Valid Range	Default Value	Description
Output	TempAlarm	BOOL	TRUE, FALSE	FALSE	Temperature alarm. TRUE: Temperature exceeds the acceptable range. FALSE: Temperature is within the acceptable range.
	TempValue	DINT	Follows data type.	0	Indicates temperature display value. Displays device temperature.

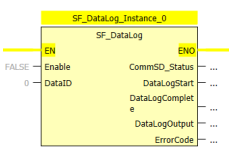
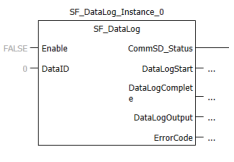
- **Function description**

  This function block is used to monitor device temperature. The device temperature is displayed via the output parameter "TempValue", and an alarm is triggered via the output parameter "TempAlarm" when the temperature exceeds the acceptable range.

11.10.2.2.27 SF_DataLog

SF_DataLog is a function block used for safety data logging.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_DataLog	Safety logging function	FB		

- **Related variables**

  Variables

Input / Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Enable	BOOL	TRUE, FALSE	FALSE	Rising edge: Start safety data logging function. Falling edge: Stop safety data logging function.
	DataID	DINT	1, 2	1	Safety data logging group number.
Output	CommSD_Status	BOOL	TRUE, FALSE	FALSE	SD card status. TRUE: SD card status is normal. FALSE: SD card status is faulty.
	DataLogStart	BOOL	TRUE, FALSE	FALSE	Safety data logging start flag. TRUE: Safety data logging is in progress. FALSE: Safety data logging has stopped.
	DataLogComplete	BOOL	TRUE, FALSE	FALSE	Safety data logging completion flag. TRUE: Safety data logging has been completed. FALSE: Safety data logging has not been completed.
	DataLogOutput	BOOL	TRUE, FALSE	FALSE	Safety data logging writing flag. TRUE: Safety data logging is written to file. FALSE: Safety data logging is not written to file.
	ErrorCode	INT	Follows data type	0	Fault code when safety data logging encounters a fault.

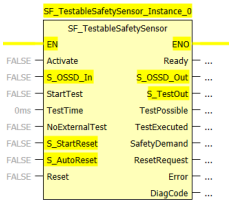
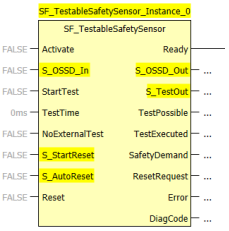
- **Function description**

  This function block is used to start/stop safety data logging. It displays the safety data logging status: SD card status, logging start status, logging completion status, output file status.

11.10.2.2.28 SF_TestableSafetySensor

SF_TestableSafetySensor is a function block used to detect faults such as sensing unit detection capability failure, excessive response time, and static ON signals in single-channel sensor systems. It is applicable to externally testable safety sensors, such as electro-sensitive protective equipment (ESPE, such as light curtains).

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_TestableSafetySensor	Testable safety sensors	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.
	S_OSSD_In	SAFEBOOL	TRUE, FALSE	FALSE	Output status of the sensor (such as light curtains). FALSE: Safety sensor is in test status or requests a safety-related response. TRUE: Safety sensor is in normal operating status.
	StartTest	BOOL	TRUE, FALSE	FALSE	Input to start sensor test. Set "S_TestOut" and start internal time monitoring function. FALSE: No test request. TRUE: Test request.
	TestTime	TIME	Follows data type	T#0ms	Ranges from 0 to 150 ms. Test time for safety sensor.
	NoExternalTest	BOOL	TRUE, FALSE	FALSE	Indicates whether external manual sensor test is supported. FALSE: External manual sensor test is supported. Automatic sensor test cannot be performed again after test failure until the manual sensor switch sequence is completed. TRUE: External manual sensor test is not supported.
	S_StartReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to start reset.
	S_AutoReset	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether to automatically reset.
	Reset	BOOL	TRUE, FALSE	FALSE	Manual reset
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
	S_OSSD_Out	SAFEBOOL	TRUE, FALSE	FALSE	Safety-related output indicating the ESPE status. FALSE: A safety-related action request or test fault is present in the sensor. TRUE: No safety-related action request and no test fault in the sensor.
	S_TestOut	SAFEBOOL	TRUE, FALSE	FALSE	Connected to the sensor's test input. Although specified as SAFEBOOL, this signal is typically connected to a BOOL output in practice. FALSE: A test request is issued. TRUE: No test request is issued.
	TestPossible	BOOL	TRUE, FALSE	FALSE	Feedback signal process. FALSE: Automatic sensor test cannot be executed. TRUE: Automatic sensor test can be executed.
	TestExecuted	BOOL	TRUE, FALSE	FALSE	Indicates the successful execution of automatic sensor test. FALSE: Automatic sensor test has not been executed, is in progress, or is faulty. TRUE: Automatic sensor test has been successfully executed.
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	ResetRequest	BOOL	TRUE, FALSE	FALSE	Flag indicating whether a reset is required.
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**
  - This function block (Type 2 ESPE) must be equipped with cyclic test capability to detect dangerous faults (such as sensing unit detection capability failure, excessive response time, etc.).
  - Test signals shall simulate the operation of the sensing device, and the cyclic test must not last for longer than 150 ms. The test verifies that each beam operates as specified by the supplier. If the cyclic

test needs to be triggered by an external safety-related control system (such as a machine equipment), the ESPE shall be equipped with appropriate input devices (such as terminals).

- The selection of the ESPE shall comply with the EN IEC 61496-1, -2, and -3 standards, and the test shall be monitored through independent functions to see if it is initiated at appropriate intervals
- Inputs S_StartReset and S_AutoReset can only be activated when it is ensured that the activation of the programmable electronic system (PES) will not cause hazardous conditions.



Power-on test of the ESPE is outside the scope of this function block and must be handled by users within the application.

- Test mode:
  1. StartTest = TRUE: S_TestOut is set to FALSE and monitoring timing is started.
  2. S_TestOut signal stops the transmitter (start TestTime monitoring for the first time).
  3. S_OSSD_In changes from TRUE to FALSE (start TestTime monitoring for the second time).
  4. S_TestOut changes from FALSE to TRUE.
  5. Start the transmitter.
  6. S_OSSD_In changes from FALSE to TRUE.
  7. Monitoring timing is stopped.
  8. During testing, S_OSSD_Out is set to TRUE.
- Optional start inhibit:
  - Startup is inhibited after the function block is activated.
  - Startup is inhibited after the protective device is interrupted.

- **Status transition diagram**

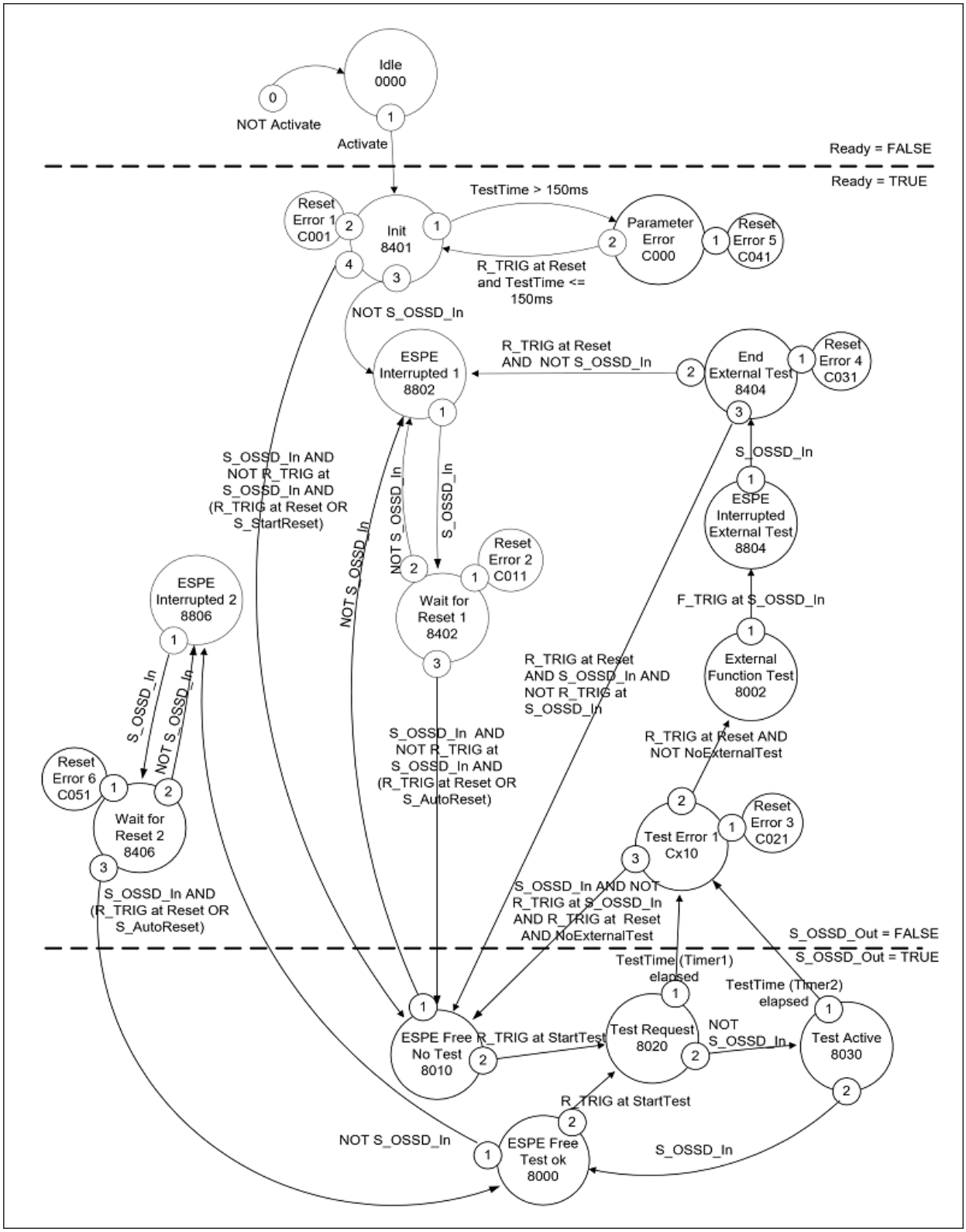
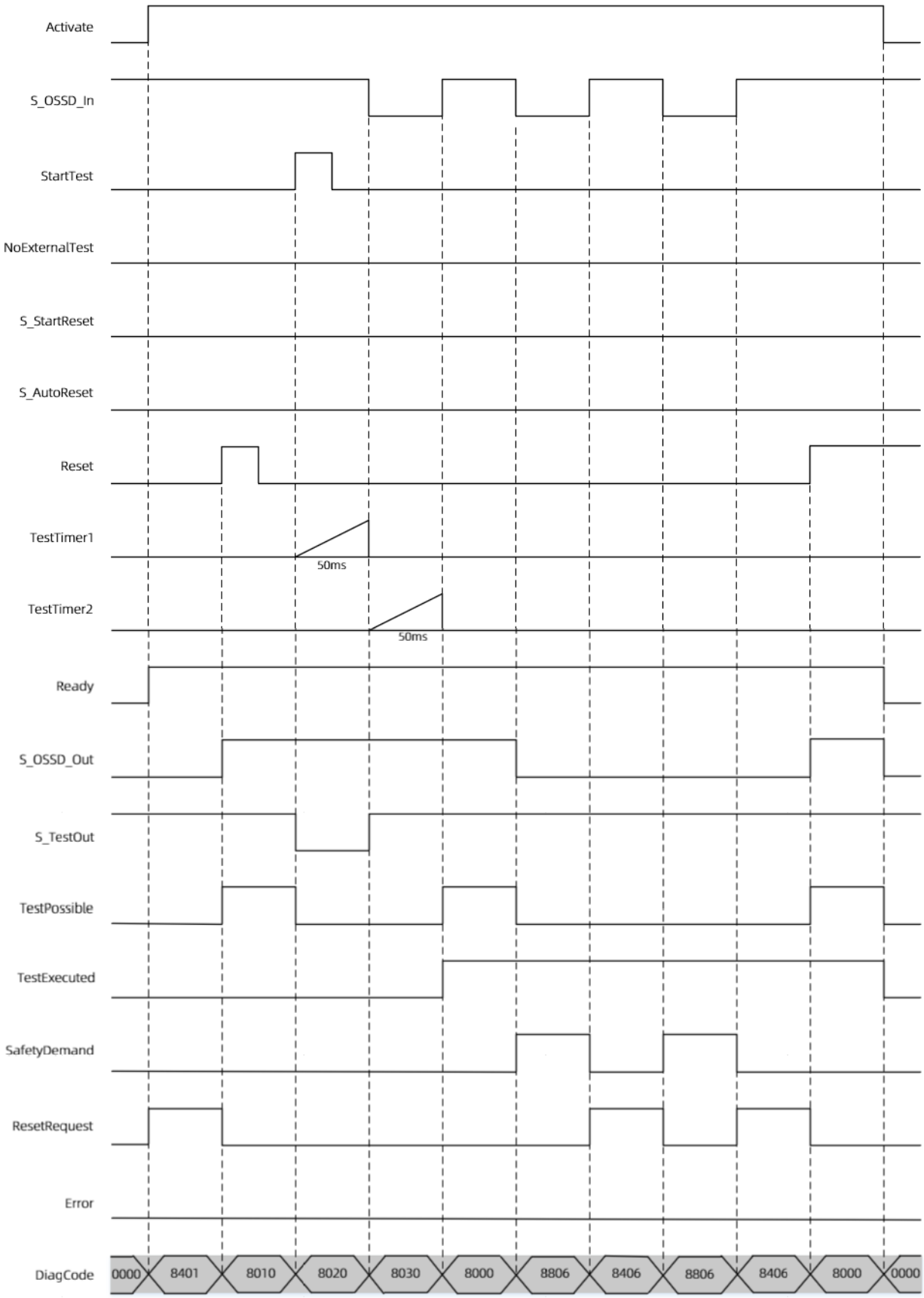


Figure 11-10-2-2-28-1 Status transition diagram for SF_TestableSafetySensor

- This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.
- The transition from any status to the idle status when Activate = FALSE is not indicated, but the priority (0) for transitioning to the idle status is the highest.

- **Sequence diagram**



- **Fault detection**

  The functional block forces a transition to the faulty status under the following conditions:

  - Test timeout with no sensor feedback delay.
  - No sensor signal feedback during testing.
  - Invalid static reset signal present during testing.

- The validity check of the monitoring time setting fails.

- **Fault behavior**

  When a fault occurs, S_OSSD_Out is set to FALSE and remains in the safety status.

  - Troubleshooting and resetting conditions:

    After the fault is cleared and the sensor is enabled (S_OSSD_In = TRUE), the reset operation clears the faulty status and sets S_OSSD_Out to TRUE.

    If S_AutoReset = FALSE, a TRUE value of Reset shall be triggered at the rising edge.

  - Start inhibit reset method:

    After S_OSSD_In switches to TRUE, the optional start inhibit can be reset by a rising edge at the Reset input.

    After the function block is activated, the optional start inhibit can be reset by a rising edge at the Reset input.

- **Function block-specific fault codes and status codes**

  - Fault code

DiagCode	Status Name	Description and Output
C000	Parameter Error	Invalid value in the TestTime parameter. Valid values are between 0 ms and 150 ms. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C001	Reset Error 1	After the function block is activated, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
C011	Reset Error 2	In the 8402 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C021	Reset Error 3	In the Cx10 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C031	Reset Error 4	In the 8404 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
C041	Reset Error 5	In the C000 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE
C051	Reset Error 6	In the 8406 status, a TRUE value of Reset is triggered at the non-rising edge. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
Cx10	Test Error 1	Test timeout in status 8020 or 8030. If S_OSSD_IN is TRUE and NoExternalTest is TRUE, the fault code is C410; otherwise, it is C010. • When the fault code is C410: Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = TRUE • When the fault code is C010: Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = TRUE

◦ Status code

DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8401	Init (Initialization)	Function block is activated. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = FALSE
8802	ESPE Interrupted 1	The function block detected a safety request. Automatic testing of the switch has not yet been performed. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8402	Wait for Reset 1	Wait for a TRUE value of Reset to be triggered at the rising edge after entering status 8802 Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = FALSE

DiagCode	Status Name	Description and Output
8002	External Function Test	Automatic sensor test is faulty. External manual sensor test is required. Necessary external manual sensor testing is supported in FB (NoExternalTest = FALSE). The sensor shall be triggered at the falling edge. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8804	ESPE Interrupted External Test	Automatic sensor test is faulty. External manual sensor test is required. Necessary external manual sensor testing is supported in FB (NoExternalTest = FALSE). TRUE signal is required at the sensor. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8404	End External Test	Automatic sensor test is faulty. External manual sensor test is required. Necessary external manual sensor testing is supported in FB (NoExternalTest = FALSE). External manual test is completed. A complete sensor switching cycle (externally controlled) is detected. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE SafetyDemand = FALSE ResetRequest = NOT Reset Error = FALSE
8010	ESPE Free No Test	No safety request is detected by the function block. Automatic testing of the sensor has not yet been performed. Ready = TRUE S_OSSD_Out = TRUE S_TestOut = TRUE TestPossible = TRUE TestExecuted = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

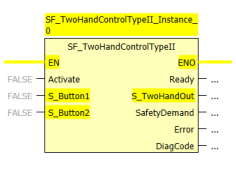
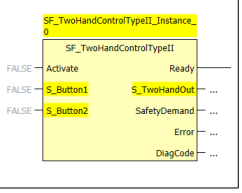
DiagCode	Status Name	Description and Output
8806	ESPE Interrupted 2	The function block detected a safety request. Automatic testing of the switch has been performed. Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = TRUE SafetyDemand = TRUE ResetRequest = FALSE Error = FALSE
8406	Wait for Reset 2	Wait for a TRUE value of Reset to be triggered at the rising edge after entering status 8806 Ready = TRUE S_OSSD_Out = FALSE S_TestOut = TRUE TestPossible = FALSE TestExecuted = TRUE SafetyDemand = FALSE ResetRequest = NOT Reset Error = FALSE
8020	Test Request	The automatic sensor test is active. The test timer is started for the first time. The transmitter signal of the sensor is turned off by the function block. The receiver signal must follow the transmitter signal. Ready = TRUE S_OSSD_Out = TRUE S_TestOut = FALSE TestPossible = FALSE TestExecuted = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

DiagCode	Status Name	Description and Output
8030	Test Active	The automatic sensor test is active. The test timer is started for the second time. Timer 1 has stopped. The transmitter signal of the sensor is activated by the function block, and the receiver signal must follow the transmitter signal. Ready = TRUE S_OSSD_Out = TRUE S_TestOut = TRUE TestPossible = FALSE TestExecuted = FALSE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE
8000	ESPE Free Test ok	No safety request is detected by the function block. Timer 2 has stopped. Automatic testing of the sensor has been performed. Ready = TRUE S_OSSD_Out = TRUE S_TestOut = TRUE TestPossible = TRUE TestExecuted = TRUE SafetyDemand = FALSE ResetRequest = FALSE Error = FALSE

11.10.2.2.29 SF_TwoHandControlTypeII

SF_TwoHandControlTypeII is a function block used for Type II two-hand control. It controls the safety output via signals from the function application and safety signals with optional start inhibit function.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_TwoHandControlTypeII	Type II two-hand control	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.
	S_Button1	SAFEBOOL	TRUE, FALSE	FALSE	Represents the input of button 1. FALSE: Button 1 is released. TRUE: Button 1 is activated.
	S_Button2	SAFEBOOL	TRUE, FALSE	FALSE	Input for button 2. FALSE: Button 2 is released. TRUE: Button 2 is activated.
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	S_TwoHandOut	SAFEBOOL	TRUE, FALSE	FALSE	Safety-related output FALSE: Correct two-hand operation is not performed. TRUE: S_Button1 and S_Button2 inputs are TRUE, with no fault. The two-hand operation is correct.
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**
  - The SF_OutControl function block is the driver for safety outputs, which are controlled via S_OutControl using signals from the functional application (ProcessControl/BOOL for process control) and signals from the safety application (S_SafeControl/SAFEBOOL for safety function control).
  - Optional conditions for ProcessControl:
    - After the function block is activated or the safety signal (S_SafeControl) is fed back, an additional function start is required (ProcessControl changes from FALSE to TRUE). A static TRUE signal at ProcessControl will not set S_OutControl to TRUE.
    - After the function block is activated or the safety signal (S_SafeControl) is fed back, no additional function start is required (ProcessControl changes from FALSE to TRUE). If other conditions are met, a static TRUE signal at ProcessControl will set S_OutControl to TRUE.

- Optional start inhibit:
  - Start inhibit after function block activation
  - Start inhibit after protective device interruption
- The StaticControl, S_StartReset, and S_AutoReset inputs should only be activated when no hazardous situation will occur during PES startup.

- **Status transition diagram**

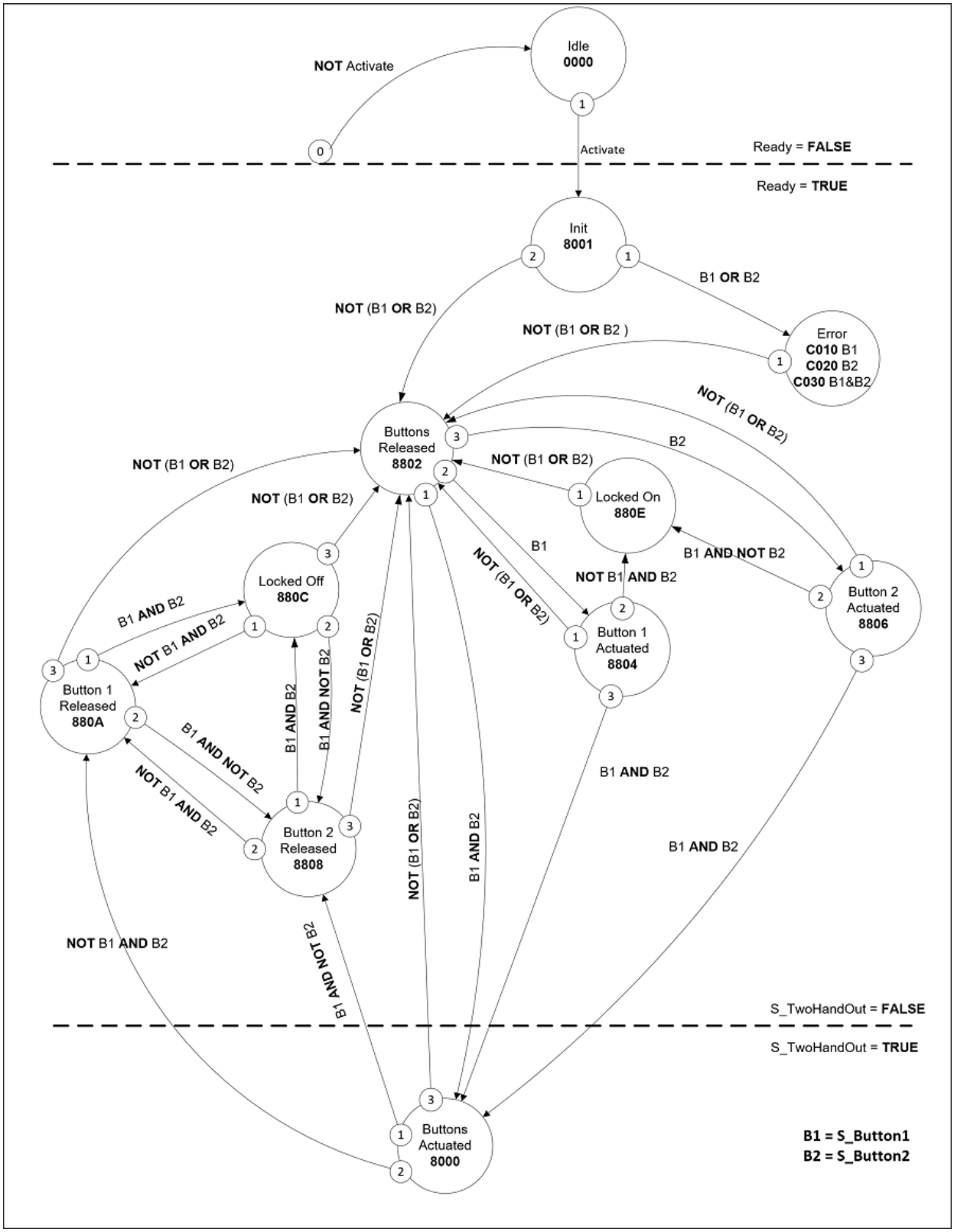
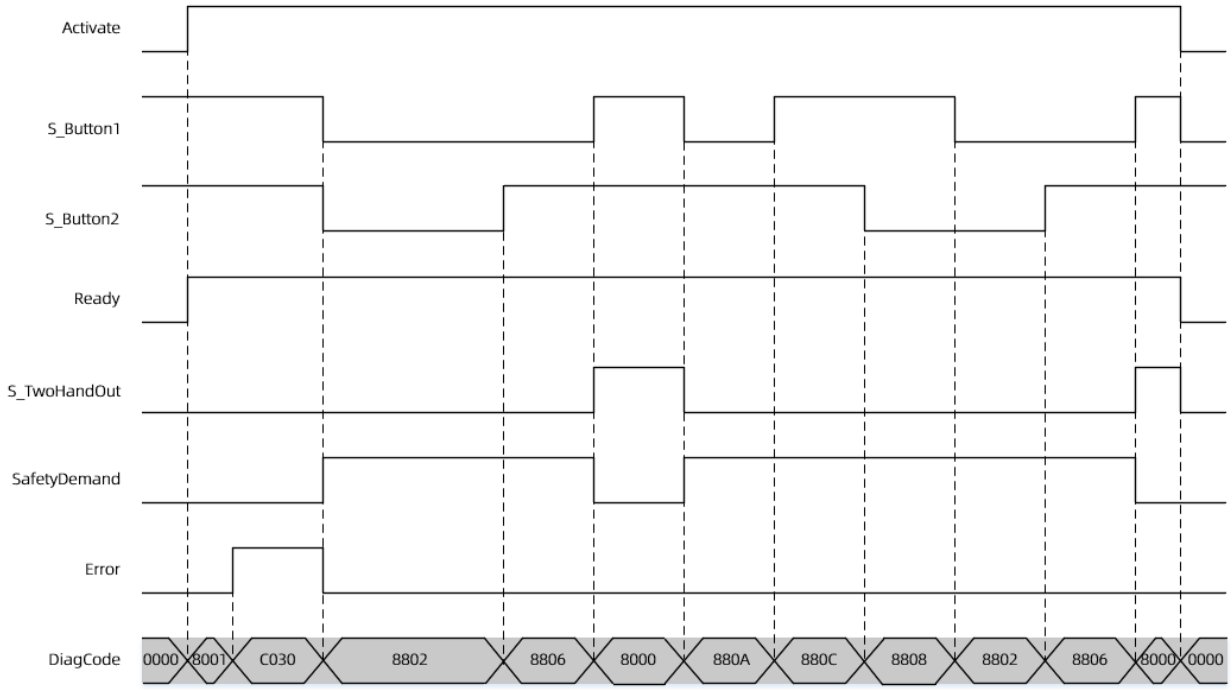


Figure 11-10-2-2-29-1 Status transition diagram for SF_TwoHandControlTypeII

- This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.
- The transition from any status to the idle status when Activate = FALSE is not indicated, but the priority (0) for transitioning to the idle status is the highest.

- **Sequence diagram**



- **Fault detection**
  Detects if a TRUE signal is mistakenly sent to the input parameter after the function block is activated.
- **Fault behavior**

  When a fault occurs, the "S_TwoHandOut" output is set to FALSE and maintained in a safety status.

  After both buttons are released (set to FALSE), the faulty status is cleared.
- **Function block-specific fault codes and status codes**
  - Fault code

DiagCode	Status Name	Description and Output
C010	Error 1 B1	S_Button1 is TRUE when the function block is activated. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = FALSE Error = TRUE
C020	Error 1 B2	S_Button2 is TRUE when the function block is activated. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
C030	Error 1 B1&B2	S_Button1 and S_Button2 are TRUE when the function block is activated. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = FALSE Error = TRUE

◦ Status code

DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_TwoHandOut = FALSE SafetyDemand = FALSE Error = FALSE
8000	Buttons Actuated	Both buttons are actuated correctly. Safety-related output is valid. Ready = TRUE S_TwoHandOut = TRUE SafetyDemand = FALSE Error = FALSE
8001	Init	Function block is activated and is in initialization status. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = FALSE Error = FALSE
8802	Buttons Released	No buttons have been actuated. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = TRUE Error = FALSE

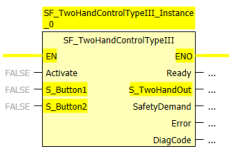
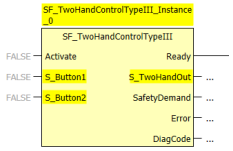
DiagCode	Status Name	Description and Output
8804	Button 1 Actuated	Only Button 1 has been actuated. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = TRUE Error = FALSE
8806	Button 2 Actuated	Only Button 2 has been actuated. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = TRUE Error = FALSE
8808	Button 2 Released	Safety output was set to valid, and then set to invalid. After setting the safety output to invalid, neither S_Button1 nor S_Button2 changes to FALSE. In this status, after setting the safety output to invalid, S_Button1 is TRUE and S_Button2 is FALSE. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = TRUE Error = FALSE
880 A	Button 1 Released	Safety output was set to valid, and then set to invalid. After setting the safety output to invalid, neither S_Button1 nor S_Button2 changes to FALSE. In this status, after setting the safety output to invalid, S_Button1 is FALSE and S_Button2 is TRUE. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = TRUE Error = FALSE

DiagCode	Status Name	Description and Output
880C	Locked Off	Safety output was set to valid, and then set to invalid. After setting the safety output to invalid, neither S_Button1 nor S_Button2 changes to FALSE. In this status, after setting the safety output to invalid, S_Button1 is TRUE and S_Button2 is TRUE. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = TRUE Error = FALSE
880E	Locked On	Incorrect button operation. Wait for both buttons to be released. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = TRUE Error = FALSE

11.10.2.2.30 SF_TwoHandControlTypeIII

SF_TwoHandControlTypeIII is a function block used for Type III two-hand control. It controls the safety output via signals from the function application and safety signals with optional start inhibit function.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_TwoHandControlTypeIII	Type III two-hand control	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Activate	BOOL	TRUE, FALSE	FALSE	Activates the instruction.
	S_Button1	SAFEBOOL	TRUE, FALSE	FALSE	Represents the input of button 1. FALSE: Button 1 is released. TRUE: Button 1 is activated.
	S_Button2	SAFEBOOL	TRUE, FALSE	FALSE	Input for button 2. FALSE: Button 2 is released. TRUE: Button 2 is activated.

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
Output	Ready	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is in ready status.
	S_TwoHandOut	SAFEBOOL	TRUE, FALSE	FALSE	Safety-related output FALSE: Correct two-hand operation is not performed. TRUE: S_Button1 and S_Button2 inputs are TRUE, with no fault. The two-hand operation is correct.
	SafetyDemand	BOOL	TRUE, FALSE	FALSE	Activates or maintains the status according to the instruction status.
	Error	BOOL	TRUE, FALSE	FALSE	Faulty status
	DiagCode	WORD	Follows data type	16#0000	Status code: Indicates fault code when instruction fails; indicates instruction status when instruction is running.

- **Function description**
  - This function block supports two-hand control compliant with Type III of EN 574 Section 4.
  - If S_Button1 and S_Button2 are set to TRUE within 500 ms in the correct sequence, then S_TwoHandOut output is also set to TRUE.
  - The function block also controls the release of both buttons, and then allows the S_TwoHandOut output to be set to TRUE again after proper re-trigger operations.
- **Status transition diagram**

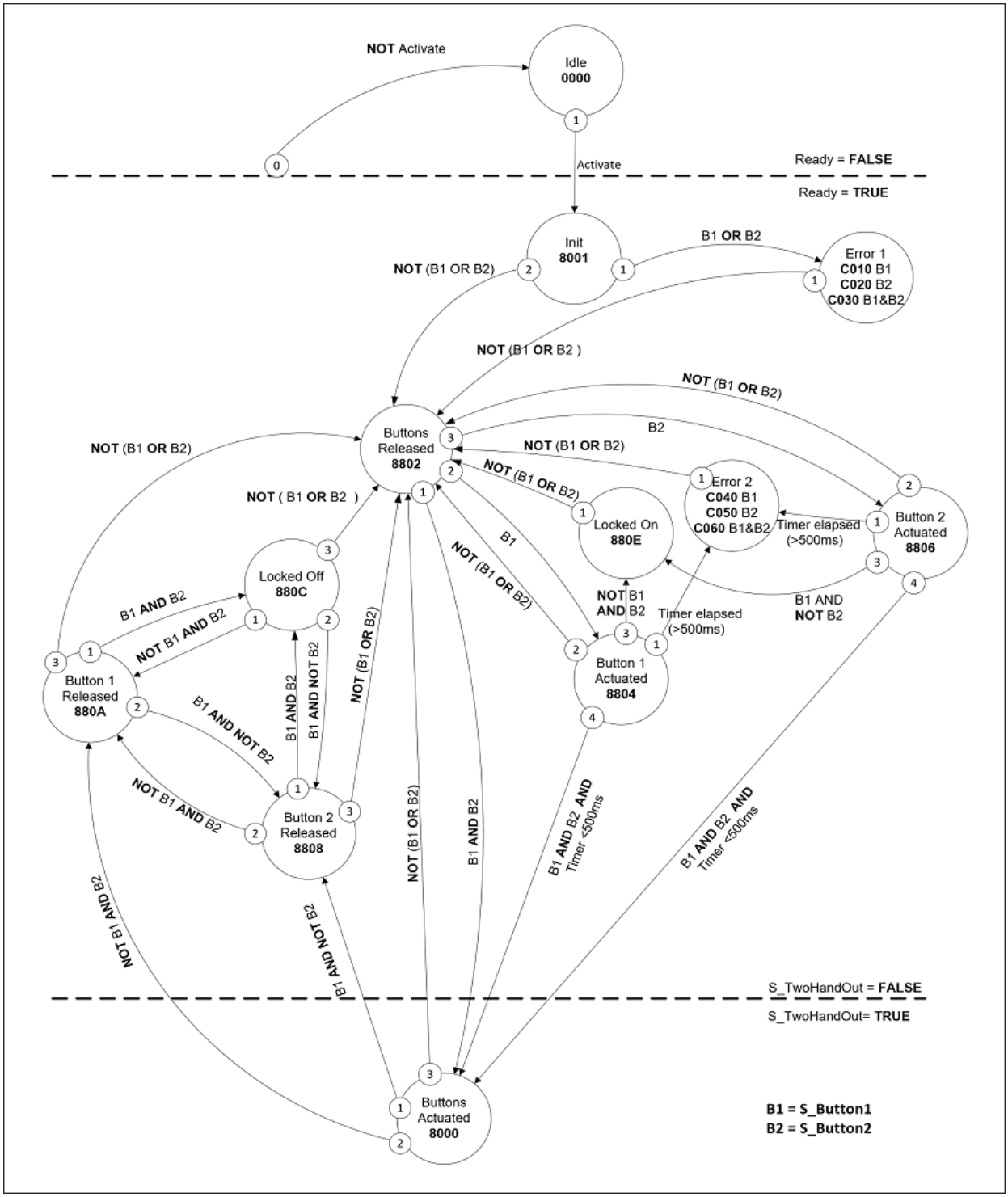
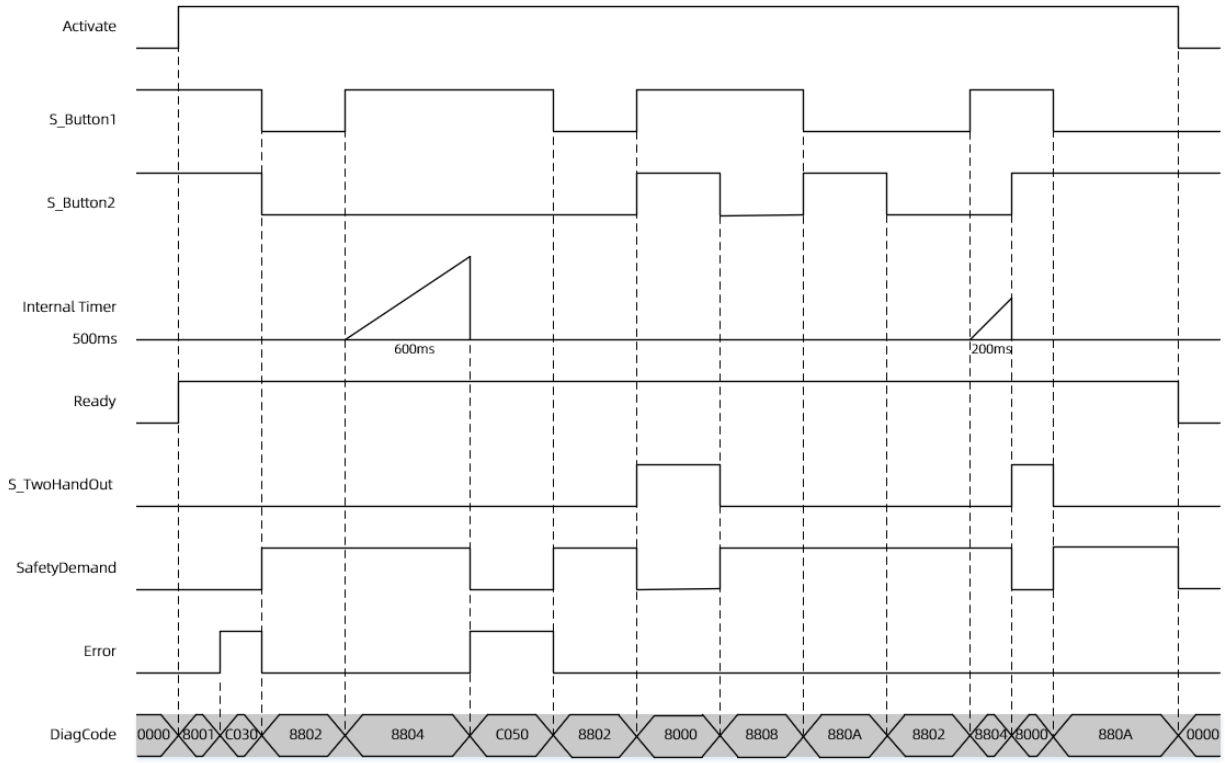


Figure 11-10-2-2-30-1 Status transition diagram for SF_TwoHandControlTypeIII

- This diagram is referenced from the PLCopen specification, and the copyright belongs to the PLCopen organization.
- The transition from any status to the idle status when Activate = FALSE is not indicated, but the priority (0) for transitioning to the idle status is the highest.

- **Sequence diagram**



- **Fault detection**
  Detects if a TRUE signal is mistakenly sent to the input parameter after the function block is activated.
  Detects if the discrepancy time between the two button inputs exceeds 500 ms.
- **Fault behavior**

  When a fault occurs, the "S_TwoHandOut" output is set to FALSE and maintained in a safety status.

  After both buttons are released (set to FALSE), the faulty status is cleared.
- **Function block-specific fault codes and status codes**
  - Fault code

DiagCode	Status Name	Description and Output
C010	Error 1 B1	S_Button1 is TRUE when the function block is activated. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = FALSE Error = TRUE
C020	Error 1 B2	S_Button2 is TRUE when the function block is activated. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = FALSE Error = TRUE

DiagCode	Status Name	Description and Output
C030	Error 1 B1&B2	S_Button1 and S_Button2 are TRUE when the function block is activated. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = FALSE Error = TRUE
C040	Error 2 B1	After 500 ms in status 8804, S_Button1 becomes FALSE, and S_Button2 becomes TRUE. Ready = TRUE SafetyDemand = FALSE Error = TRUE S_TwoHandOut = FALSE
C050	Error 2 B2	After 500 ms in status 8806, S_Button1 becomes TRUE, and S_Button2 becomes FALSE. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = FALSE Error = TRUE
C060	Error 2 B1&B2	After 500 ms in status 8804 or 8806, S_Button1 becomes TRUE, and S_Button2 becomes TRUE. Condition to reach this status: Within the 500 ms timer cycle, the statuses of S_Button1 and S_Button2 change from different to same (both TRUE). Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = FALSE Error = TRUE

◦ Status code

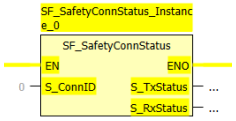
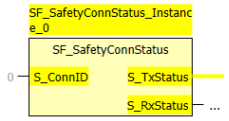
DiagCode	Status Name	Description and Output
0000	Idle	Function block is not activated (initial status). Ready = FALSE S_TwoHandOut = FALSE SafetyDemand = FALSE Error = FALSE
8000	Buttons Actuated	Both buttons are actuated correctly. Safety-related output is valid. Ready = TRUE S_TwoHandOut = TRUE SafetyDemand = FALSE Error = FALSE
8001	Init	Function block is activated and is in initialization status. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = FALSE Error = FALSE
8802	Buttons Released	No buttons have been actuated. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = TRUE Error = FALSE
8804	Button 1 Actuated	Only Button 1 has been actuated. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = TRUE Error = FALSE
8806	Button 2 Actuated	Only Button 2 has been actuated. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = TRUE Error = FALSE

DiagCode	Status Name	Description and Output
8808	Button 2 Released	Safety output was set to valid, and then set to invalid. After setting the safety output to invalid, neither S_Button1 nor S_Button2 changes to FALSE. In this status, after setting the safety output to invalid, S_Button1 is TRUE and S_Button2 is FALSE. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = TRUE Error = FALSE
880 A	Button 1 Released	Safety output was set to valid, and then set to invalid. After setting the safety output to invalid, neither S_Button1 nor S_Button2 changes to FALSE. In this status, after setting the safety output to invalid, S_Button1 is FALSE and S_Button2 is TRUE. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = TRUE Error = FALSE
880C	Locked Off	Safety output was set to valid, and then set to invalid. After setting the safety output to invalid, neither S_Button1 nor S_Button2 changes to FALSE. In this status, after setting the safety output to invalid, S_Button1 is TRUE and S_Button2 is TRUE. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = TRUE Error = FALSE
880E	Locked On	Incorrect button operation. Wait for both buttons to be released. Ready = TRUE S_TwoHandOut = FALSE SafetyDemand = TRUE Error = FALSE

11.10.2.2.31 SF_SafetyConnStatus

SF_SafetyConnStatus is a function block used to get the safety C2C (CIP Safety) connection status.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_SafetyConnStatus	Gets safety C2C (CIP Safety) connection status.	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
Input	S_ConnID	SAFEWORD	0 and positive numbers	0	Connection ID for safety C2C communication
Output	S_TxStatus	SAFEBOOL	TRUE, FALSE	FALSE	Transmission status of the connection TRUE: Connection status is normal. FALSE: Connection status is faulty.
	S_RxStatus	SAFEBOOL	TRUE, FALSE	FALSE	Receiving status of the connection TRUE: Connection status is normal. FALSE: Connection status is faulty.

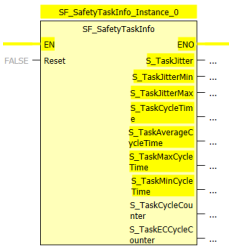
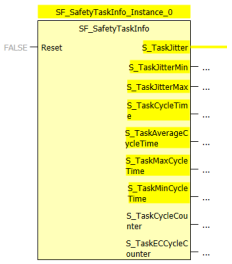
- **Function description**

  This function block is used to obtain the safety C2C connection status. If the connection is only configured for one-direction communication (transmitting or receiving), the status of the other direction is fixed as FALSE.

11.10.2.2.32 SF_SafetyTaskInfo

SF_SafetyTaskInfo is a function block used to obtain safety task information.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_SafetyTaskInfo	Obtains safety task information.	FB		

- **Related variables**

  Variables

Input/ Output	Variable	Data Type	Valid Range	Default Value	Description
Input	Reset	BOOL	TRUE, FALSE	FALSE	Resets task data.
Output	S_TaskJitter	SAFEDINT	Follows data type	0	Cycle jitter
	S_TaskJitterMin	SAFEDINT	Follows data type	0	Minimum cycle jitter
	S_TaskJitterMax	SAFEDINT	Follows data type	0	Maximum cycle jitter
	S_TaskCycleTime	SAFEWORD	Follows data type	0	Last execution time
	S_TaskAverageCycleTime	SAFEWORD	Follows data type	0	Average execution time
	S_TaskMaxCycleTime	SAFEDINT	Follows data type	0	Maximum execution time
	S_TaskMinCycleTime	SAFEWORD	Follows data type	0	Minimum execution time
	S_TaskCycleCounter	DWORD	Follows data type	0	System scheduling count
	S_TaskECCycleCounter	DWORD	Follows data type	0	User task execution count

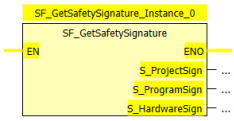
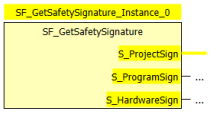
- **Function description**

  This function block is used to obtain safety task information, which can be reset upon the rising edge (FALSE->TRUE) of the input parameter "Reset".

11.10.2.2.33 SF_GetSafetySignature

SF_GetSafetySignature is a function block used to obtain the safety signature.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_GetSafetySignature	Obtains safety signature.	FB		

- **Related variables**

  Variables

Input/Output	Variable	Data Type	Valid Range	Default Value	Description
Output	S_ProjectSign	SAFEDINT	Follows data type	0	Represents the overall project signature.
	S_ProgramSign	SAFEDINT	Follows data type	0	Represents the safety program signature.
	S_HardwareSign	SAFEDINT	Follows data type	0	Represents the safety hardware signature.

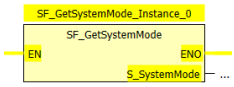
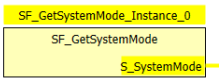
- **Function description**

  This function block is used to obtain the overall project signature, safety program signature, and safety hardware signature.

11.10.2.2.34 SF_GetSystemMode

SF_GetSystemMode is a function block used to obtain the safety PLC status.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_GetSystemMode	Obtains safety PLC status.	FB		

- **Related variables**

  Variables

Input/Output	Variable	Data Type	Valid Range	Default Value	Description
Output	S_SystemMode	SAFEBYTE	Follows data type	0	Represents the system status of the safety PLC. 1: Commissioning mode 2: Run mode 3: Fault mode 4: Pending mode

- **Function description**

  This function block is used to obtain the system status of the safety PLC.

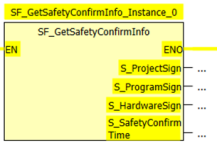
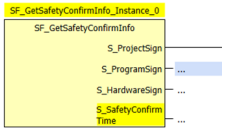
  - 1: Commissioning mode
  - 2: Run mode

- 3: Fault mode
- 4: Pending mode

11.10.2.2.35 SF_GetSafetyConfirmInfo

SF_GetSafetyConfirmInfo is a function block used to get safety confirmation information (including safety signature, safety confirmation time).

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_GetSafetyConfirmInfo	Get safety confirmation information (including safety signature, safety confirmation time)	FB		

- **Related variables**

  Output variables

Output Variable	Variable	Data Type	Valid Range	Default Value	Description
S_ProjectSign	Project overall signature	SAFEDWORD	Follows data type	0	Project overall signature
S_ProgramSign	Safety program signature	SAFEDWORD	Follows data type	0	Safety program signature
S_HardwareSign	Safety hardware signature	SAFEDWORD	Follows data type	0	Safety hardware signature
S_SafetyConfirmTime	Safety confirmation time	SAFEDATE_AND_TIME	Follows data type	1970-01-01-0:0:0	Safety confirmation time

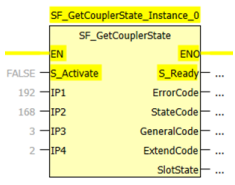
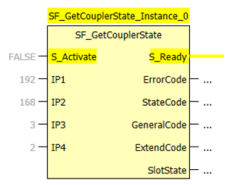
- **Function**

  This function block is used to get safety confirmation information (including safety signature, safety confirmation time).

11.10.2.2.36 SF_GetCouplerState

SF_GetCouplerState is a function block used to get the coupler status (EIP connection).

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_GetCouplerState	Get coupler status (EIP connection)	FB		

- **Related variables**

  Input variables

Input Variable	Variable	Data Type	Valid Range	Default Value	Description
S_Activate	Instruction activation status	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is activated.
IP1	IP field 1	BYTE	Follows data type	192	Coupler IP field 1
IP2	IP field 2	BYTE	Follows data type	168	Coupler IP field 2
IP3	IP field 3	BYTE	Follows data type	3	Coupler IP field 3
IP4	IP field 4	BYTE	Follows data type	2	Coupler IP field 4

  Output variables

Output Variable	Name	Data Type	Value Range	Default Value	Description
S_Ready	Instruction ready status	SAFEBOOL	TRUE, FALSE	FALSE	TRUE: Instruction is ready. FALSE: Instruction is not ready.
ErrorCode	Instruction status code	WORD	Follows data type	0	Status code: Indicates the instruction status. 0: Normal; 1: Abnormal
StateCode	Coupler EtherNet/IP communication status code	BYTE	Follows data type	0	Connection status code
GeneralCode		BYTE	Follows data type	0	General status code
ExtendCode		WORD	Follows data type	0	Extended status code
SlotState[32]	I/O module diagnostic status for each slot	SlotState	Follows data type	0	I/O module diagnostic status for each slot

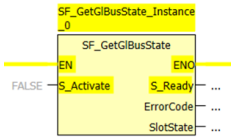
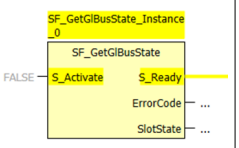
- **Function**

This function block is used to get the coupler status under an EIP connection. Input the coupler IP address to query.

11.10.2.2.37 SF_GetGlBusState

SF_GetGlBusState is a function block used to obtain local bus connection status (EIP device).

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_GetGlBusState	Get local bus connection status (EIP device)	FB		

- **Related variables**

  Input variables

Input Variable	Variable	Data Type	Valid Range	Default Value	Description
S_Activate	Instruction activation status	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is activated.

  Output variables

Output Variable	Name	Data Type	Value Range	Default Value	Description
S_Ready	Instruction ready status	SAFEBOOL	TRUE, FALSE	FALSE	TRUE: Instruction is ready. FALSE: Instruction is not ready.
ErrorCode	Instruction status code	WORD	Follows data type	0	Status code: Indicates the instruction status. 0: Normal; 1: Abnormal
SlotState[32]	I/O module diagnostic status for each slot	SlotState	Follows data type	0	I/O module diagnostic status for each slot

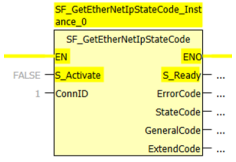
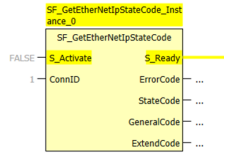
- **Function**

  This function block is used to get the local bus connection status (local EIP device).

11.10.2.2.38 SF_GetEtherNetIpStateCode

SF_GetEtherNetIpStateCode is a function block used to get the EtherNet/IP (Scanner) connection status.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_GetEtherNetIpStateCode	Get EtherNet/IP (Scanner) connection status	FB		

- **Related variables**

  Input variables

Input Variable	Variable	Data Type	Valid Range	Default Value	Description
S_Activate	Instruction activation status	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is activated.
ConnID	EtherNet/IP connection ID	WORD	Follows data type	1	EtherNet/IP connection ID

  Output variables

Output Variable	Name	Data Type	Value Range	Default Value	Description
S_Ready	Instruction ready status	SAFEBOOL	TRUE, FALSE	FALSE	TRUE: Instruction is ready. FALSE: Instruction is not ready.
ErrorCode	Instruction status code	WORD	Follows data type	0	Status code: Indicates the instruction status. 0: Normal; 1: Invalid connection ID
StateCode	EtherNet/IP connection status code	BYTE	Follows data type	0	Connection status code
GeneralCode		BYTE	Follows data type	0	General status code
ExtendCode		WORD	Follows data type	0	Extended status code

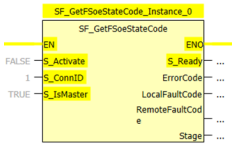
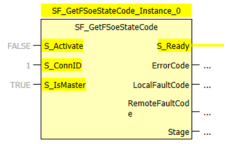
- **Function**

  This function block is used to get the EtherNet/IP (Scanner) connection status. Input the connection ID to query.

11.10.2.2.39 SF_GetFSoeStateCode

SF_GetFSoeStateCode is a function block used to get the FSoE connection status.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_GetFSoeStateCode	Get FSoE connection status	FB		

- **Related variables**

Input variables

Input Variable	Variable	Data Type	Valid Range	Default Value	Description
S_Activate	Instruction activation status	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is activated.
S_ConnID	Connection ID	SAFEWORD	Follows data type	1	Connection ID for FSoE communication.
S_IsMaster	Specifies whether to query the device status as master or slave.	SAFEBOOL	TRUE, FALSE	TRUE	TRUE: Query the device status as master; FALSE: Query the device status as slave.

Output variables

Output Variable	Name	Data Type	Value Range	Default Value	Description
S_Ready	Instruction ready status	SAFEBOOL	TRUE, FALSE	FALSE	TRUE: Instruction is ready. FALSE: Instruction is not ready.
ErrorCode	Instruction status code	WORD	Follows data type	0	Status code: Indicates the instruction status. 0: Normal; 1: Incorrect connection ID; 2: Function block exception

LocalFaultCode	FSoE connection fault code (local)	WORD	Follows data type	0	FSoE connection fault code (local) 0: Normal; 1: Unexpected instruction error; 2: Unknown instruction; 3: Invalid connection ID; 4: CRC error; 5: Watchdog timeout; 6: Invalid slave address; 7: Invalid safety data; 8: Invalid communication parameter length; 9: Invalid communication parameter data; 10: Invalid application parameter length; 11: Invalid application parameter data
RemoteFaultCode	FSoE connection fault code (remote)	WORD	Follows data type	0	FSoE connection fault code (remote) 0: Normal; 1: Unexpected instruction error; 2: Unknown instruction; 3: Invalid connection ID; 4: CRC error; 5: Watchdog timeout; 6: Invalid slave address; 7: Invalid safety data; 8: Invalid communication parameter length; 9: Invalid communication parameter data; 10: Invalid application parameter length; 11: Invalid application parameter data
Stage	Current FSoE connection stage	WORD	Follows data type	0	Current FSoE connection stage 0: No state; 1: Reset stage; 2: Session stage; 3: Connection stage; 4: Parameter stage; 5: Data stage

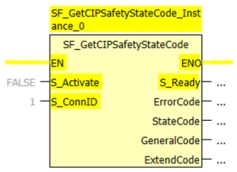
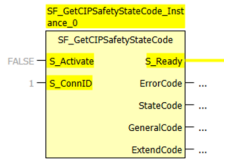
- **Function**

  This function block is used to get the FSoE connection status. Input the connection ID to query.

11.10.2.2.40 SF_GetCIPSafetyStateCode

SF_GetCIPSafetyStateCode is a function block used to get the CIP Safety connection status code.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SF_GetCIPSafetyStateCode	Get CIP Safety connection status code	FB		

- **Related variables**

Input variables

Input Variable	Variable	Data Type	Valid Range	Default Value	Description
S_Activate	Instruction activation status	SAFEBOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is activated.
S_ConnID	CIP Safety connection ID	SAFEWORD	Follows data type	1	CIP Safety connection ID

Output variables

Output Variable	Name	Data Type	Value Range	Default Value	Description
S_Ready	Instruction ready status	SAFEBOOL	TRUE, FALSE	FALSE	TRUE: Instruction is ready. FALSE: Instruction is not ready.
ErrorCode	Instruction status code	WORD	Follows data type	0	Status code: Indicates the instruction status. 0: Normal; 1: Incorrect connection ID; 2: Function block exception
StateCode	CIP Safety connection status code	BYTE	Follows data type	0	Connection status code
GeneralCode		BYTE	Follows data type	0	General status code

ExtendCode		WORD	Follows data type	0	Extended status code

- **Function**
  This function block is used to get the CIP Safety connection status code. Input the connection ID to query.

11.10.2.3 Extended Function Blocks

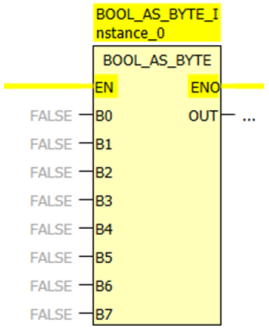
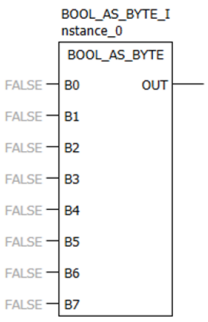
11.10.2.3.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
11.10.2.3.2 BOOL_AS_BYTE	FB	Convert BOOL type data to BYTE type	SafetyInstructions	Applicable to library version 1.1.0.0
11.10.2.3.3 BOOL_AS_WORD	FB	Convert BOOL type data to WORD type		
11.10.2.3.4 BOOL_AS_DWORD	FB	Convert BOOL type data to DWORD type		
11.10.2.3.5 SAFEBOOL_AS_SAFEBYTE	FB	Convert SAFEBOOL type data to SAFEBYTE type		
11.10.2.3.5 SAFEBOOL_AS_SAFEBYTE	FB	Convert SAFEBOOL type data to SAFEWORD type		
11.10.2.3.7 SAFEBOOL_AS_SAFEDWORD	FB	Convert SAFEBOOL type data to SAFEDWORD type		
11.10.2.3.8 BYTE_AS_BOOL	FB	Convert BYTE type data to BOOL type		
11.10.2.3.9 WORD_AS_BOOL	FB	Convert WORD type data to BOOL type		
11.10.2.3.10 DWORD_AS_BOOL	FB	Convert DWORD type data to BOOL type		
11.10.2.3.11 SAFEBYTE_AS_SAFEBOOL	FB	Convert SAFEBYTE type data to SAFEBOOL type		
11.10.2.3.12 SAFEWORD_AS_SAFEBOOL	FB	Convert SAFEWORD type data to SAFEBOOL type		
11.10.2.3.13 SAFEDWORD_AS_SAFEBOOL	FB	Convert SAFEDWORD type data to SAFEBOOL type		

11.10.2.3.2 BOOL_AS_BYTE

Converts eight input values of BOOL type into values of BYTE type, and outputs them from the right side.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
BOOL_AS_BYTE	Convert BOOL type data to BYTE type	FB		

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	B0 to B7	BOOL values to be converted	BOOL	Follows data type	FALSE	BOOL values to be converted, corresponding to each bit
Output	OUT	Converted BYTE value	BYTE	Follows data type	0	Conversion result

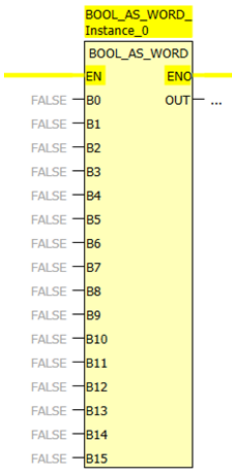
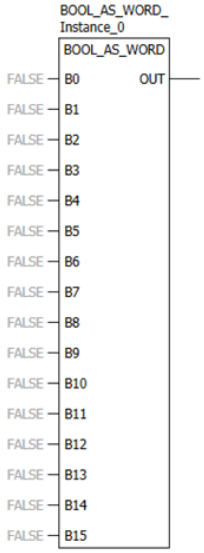
- **Data type range**

	Boolean		Bit String					
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD
IN	√	-	-	-	-	-	-	-
OUT	-	-	√	-	-	-	-	-

11.10.2.3.3 BOOL_AS_WORD

Converts 16 input values of BOOL type into values of WORD type, and outputs them from the right side.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
BOOL_AS_WORD	Convert BOOL type data to WORD type	FB		

- **Related variables**

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	B0 to B15	BOOL values to be converted	BOOL	Follows data type	FALSE	BOOL values to be converted, corresponding to each bit
Output	OUT	Converted WORD value	WORD	Follow data type	0	Conversion result

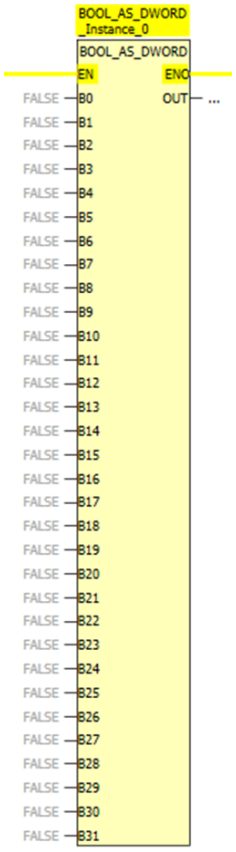
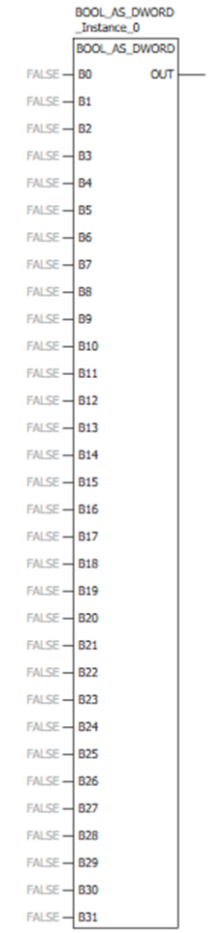
- **Data type range**

	Boolean		Bit String					
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD
IN	√	-	-	-	-	-	-	-
OUT	-	-	-	-	√	-	-	-

11.10.2.3.4 BOOL_AS_DWORD

Converts 32 input values of BOOL type into values of DWORD type, and outputs them from the right side.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
BOOL_AS_DWORD	Convert BOOL type data to DWORD type	FB		

- **Related variables**

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	B0 to B31	BOOL values to be converted	BOOL	Follows data type	FALSE	BOOL values to be converted, corresponding to each bit
Output	OUT	Converted DWORD value	DWORD	Depends on data type	0	Conversion result

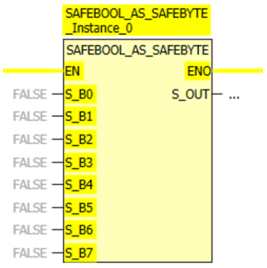
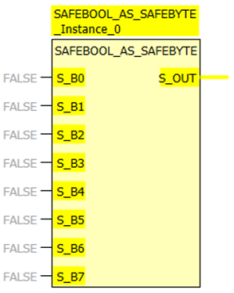
- **Data type range**

	Boolean		Bit String					
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD
IN	√	-	-	-	-	-	-	-
OUT	-	-	-	-	-	-	√	-

11.10.2.3.5 SAFEBOOL_AS_SAFEBYTE

Converts eight input values of SAFEBOOL type into values of SAFEBYTE type, and outputs them from the right side.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SAFEBOOL_AS_SAFEBYTE	Convert SAFEBOOL type data to SAFEBYTE type	FB		

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	S_B0 to S_B7	SAFEBOOL value to be converted	SAFEBOOL	Follows data type	FALSE	SAFEBOOL values to be converted, corresponding to each bit
Output	S_OUT	Converted SAFEBYTE value	SAFEBYTE	Follows data type	0	Conversion result

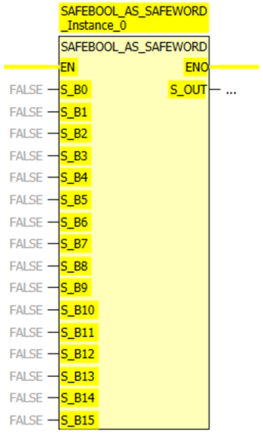
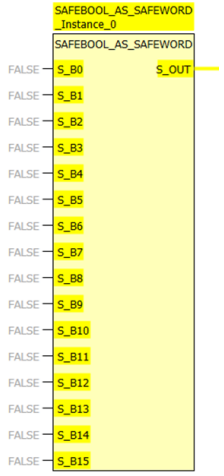
- **Data type range**

	Boolean		Bit String					
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD
IN	-	√	-	-	-	-	-	-
OUT	-	-	-	√	-	-	-	-

11.10.2.3.6 SAFEBOOL_AS_SAFEWORD

Converts 16 input values of SAFEBOOL type into values of SAFEWORD type, and outputs them from the right side.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SAFEBOOL_AS_SAFEWORD	Convert SAFEBOOL type data to SAFEWORD type	FB		

- **Related variables**

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	S_B0 to S_B15	SAFEBOOL value to be converted	SAFEBOOL	Follows data type	FALSE	SAFEBOOL values to be converted, corresponding to each bit
Output	S_OUT	Converted SAFEWORD value	SAFEWORD	Follows data type	0	Conversion result

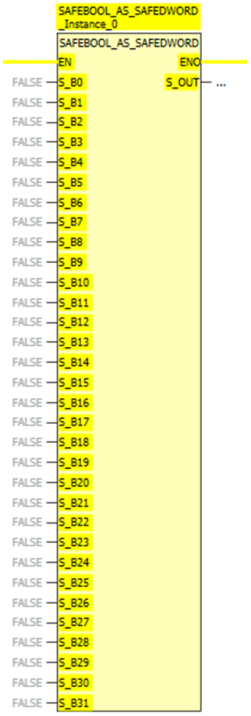
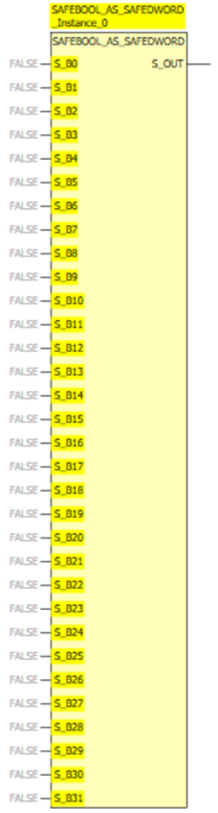
- **Data type range**

	Boolean		Bit String					
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD
IN	-	√	-	-	-	-	-	-
OUT	-	-	-	-	-	√	-	-

11.10.2.3.7 SAFEBOOL_AS_SAFEDWORD

Converts 32 input values of SAFEBOOL type into values of SAFEWORD type, and outputs them from the right side.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SAFEBOOL_AS_SAFEDWORD	Convert SAFEBOOL type data to SAFEDWORD type	FB		

- **Related variables**

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	S_B0 to S_B31	SAFEBOOL value to be converted	SAFEBOOL	Follows data type	FALSE	SAFEBOOL values to be converted, corresponding to each bit
Output	S_OUT	Converted SAFEDWORD value	SAFEDWORD	Follows data type	0	Conversion result

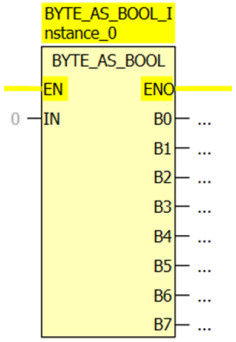
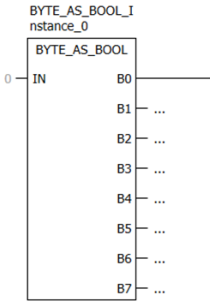
- **Data type range**

	Boolean		Bit String					
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD
IN	-	√	-	-	-	-	-	-
OUT	-	-	-	-	-	-	-	√

11.10.2.3.8 BYTE_AS_BOOL

Converts the input value of BYTE type into eight values of BOOL type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
BYTE_AS_BOOL	Convert BYTE type data to BOOL type	FB		

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	IN	BYTE value to be converted	BYTE	Follows data type	0	BYTE value to be converted
Output	B0 to B7	Converted return value	BOOL	Follows data type	FALSE	Conversion result

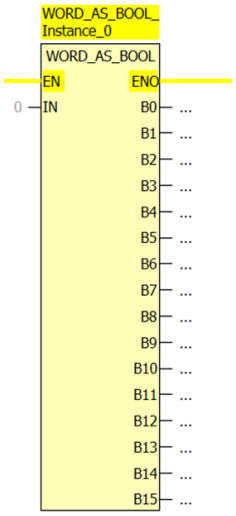
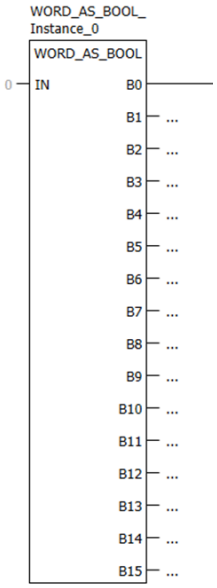
- **Data type range**

	Boolean		Bit String					
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD
IN	-	-	√	-	-	-	-	-
OUT	√	-	-	-	-	-	-	-

11.10.2.3.9 WORD_AS_BOOL

Converts the input value of WORD type into 16 values of BOOL type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
WORD_AS_BOOL	Convert WORD type data to BOOL type	FB		

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	IN	WORD value to be converted	WORD	Follow data type	0	WORD value to be converted
Output	B0 to B15	Converted return value	BOOL	Follows data type	FALSE	Conversion result

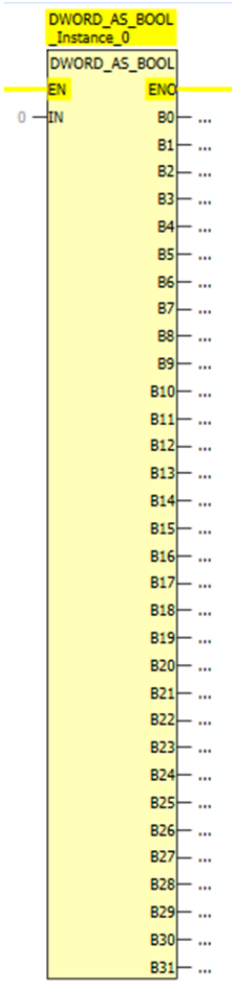
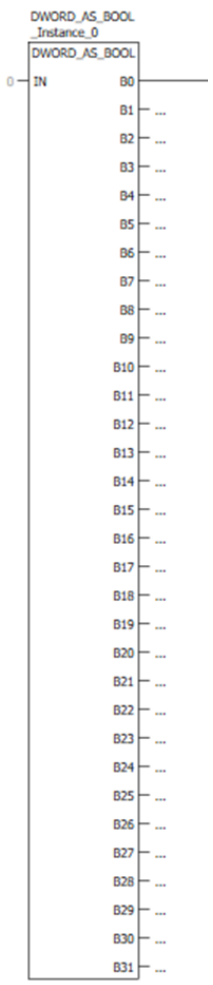
- **Data type range**

	Boolean		Bit String					
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD
IN	-	-	-	-	√	-	-	-
OUT	√	-	-	-	-	-	-	-

11.10.2.3.10 DWORD_AS_BOOL

Converts the input value of DWORD type into 32 values of BOOL type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
DWORD_AS_BOOL	Convert DWORD type data to BOOL type	FB		

- **Related variables**

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	IN	DWORD value to be converted	DWORD	Depends on data type	0	DWORD value to be converted
Output	B0 to B31	Converted return value	BOOL	Follows data type	FALSE	Conversion result

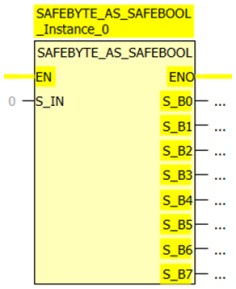
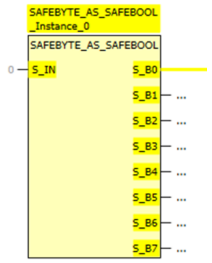
- **Data type range**

	Boolean		Bit String					
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD
IN	-	-	-	-	-	-	√	-
OUT	√	-	-	-	-	-	-	-

11.10.2.3.11 SAFEBYTE_AS_SAFEBOOL

Converts the input value of SAFEBYTE type into eight values of SAFEBOOL type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SAFEBYTE_AS_SAFEBOOL	Convert SAFEBYTE type data to SAFEBOOL type	FB		

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	S_IN	SAFEBYTE value to be converted	SAFEBYTE	Follows data type	0	SAFEBYTE value to be converted
Output	S_B0 to S_B7	Converted return value	SAFEBOOL	Follows data type	FALSE	Conversion result

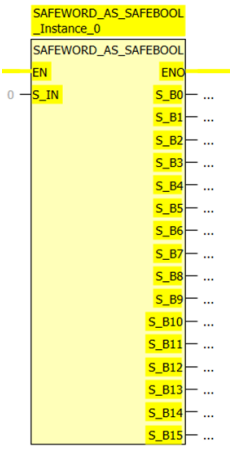
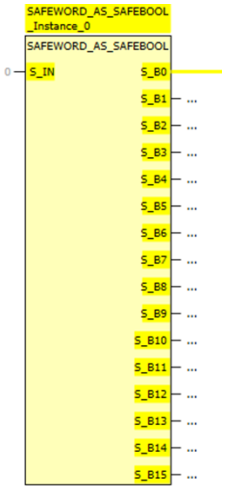
- **Data type range**

	Boolean		Bit String					
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD
IN	-	-	-	√	-	-	-	-
OUT	-	√	-	-	-	-	-	-

11.10.2.3.12 SAFEWORD_AS_SAFEBOOL

Converts the input value of SAFEWORD type into 16 values of SAFEBOOL type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SAFEWORD_AS_SAFEBOOL	Convert SAFEWORD type data to SAFEBOOL type	FB		

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	S_IN	SAFEWORD value to be converted	SAFEWORD	Follows data type	0	SAFEWORD value to be converted
Output	S_B0 to S_B15	Converted return value	SAFEBOOL	Follows data type	FALSE	Conversion result

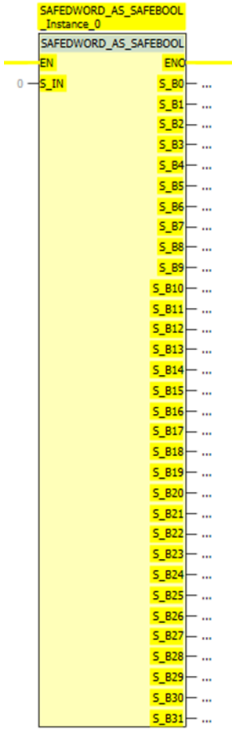
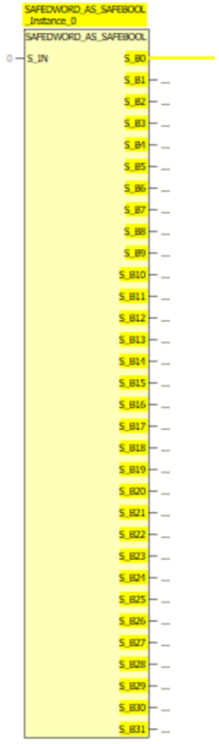
- **Data type range**

	Boolean		Bit String					
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD
IN	-	-	-	-	-	√	-	-
OUT	-	√	-	-	-	-	-	-

11.10.2.3.13 SAFEDWORD_AS_SAFEBOOL

Converts the input value of SAFEWORD type into 32 values of SAFEBOOL type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	FBD Representation
SAFEDWORD_AS_SAFEBOOL	Convert SAFEDWORD type data to SAFEBOOL type	FB		

- **Related variables**

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	S_IN	SAFEDWORD value to be converted	SAFEDWORD	Follows data type	0	SAFEDWORD value to be converted
Output	S_B0 to S_B31	Converted return value	SAFEBOOL	Follows data type	FALSE	Conversion result

- **Data type range**

	Boolean		Bit String					
	BOOL	SAFEBOOL	BYTE	SAFEBYTE	WORD	SAFEWORD	DWORD	SAFEDWORD
IN	-	-	-	-	-	-	-	√
OUT	-	√	-	-	-	-	-	-

12 Library Management

12.1 Library Editing

12.1.1 Overview

Library editing refers to editing a library project, i.e., defining a library, specifically including:

- Adding, deleting, and modifying POU, customized types, global variable tables, parameter tables, and other elements in a library project, as well as configuration logic inside the POU.
- Modifying the library reference.
- Modifying the project information, including company name, library name, version, author, and description.
- Library projects support the project protection feature, which protects library project data through user management to prevent unauthorized access. For details, see 9.15 Library Project Safety Protection.

12.1.2 Library Update

Library update refers to editing or modifying the library project information based on an existing library to distinguish differences between versions before and after the changes.

Prerequisites

A saved library project is opened.

Procedure

1. Modify the library reference information or program configuration logic.
2. Modify the library project information (company name, library name, version, author, or description).
3. Save and load into library.

Result

Through the Library Manager, you can view the information about multiple libraries, including the libraries before and after the upgrade.

12.1.3 Compiled Library

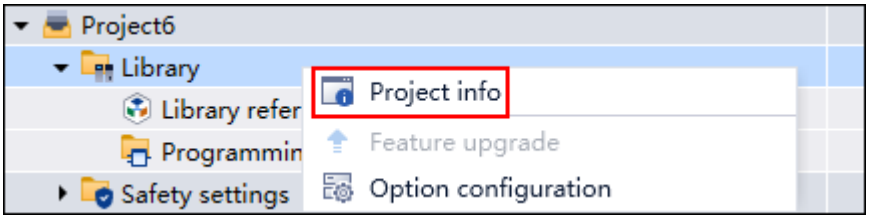
This is a compiled binary library, usually in .inoblib file format.

Prerequisites

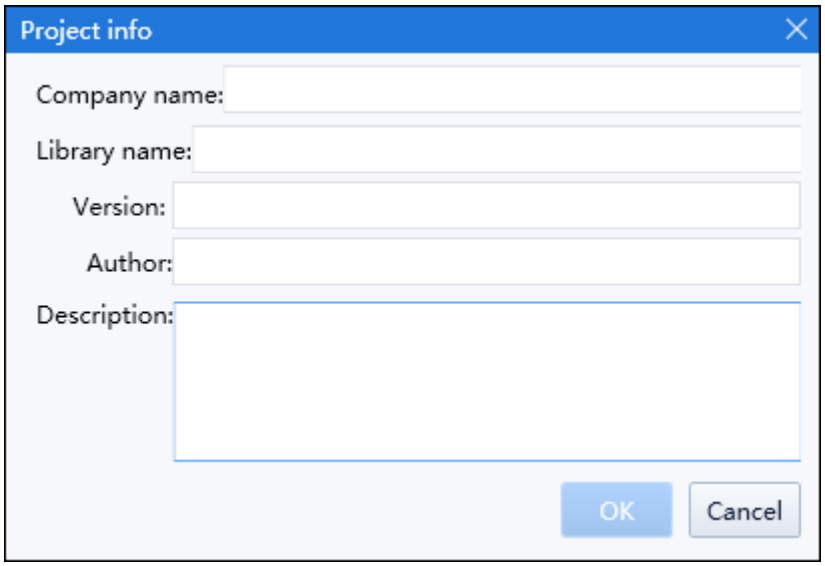
A library project has been created or opened.

Procedure

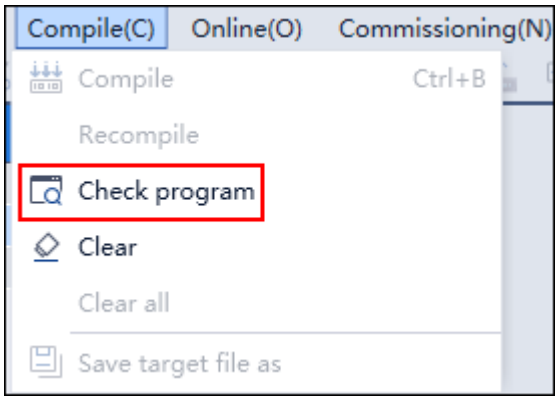
1. Add the program logic, such as program blocks and GV, and perform the logic configuration.
2. Right-click "Library" and then click "Project info".



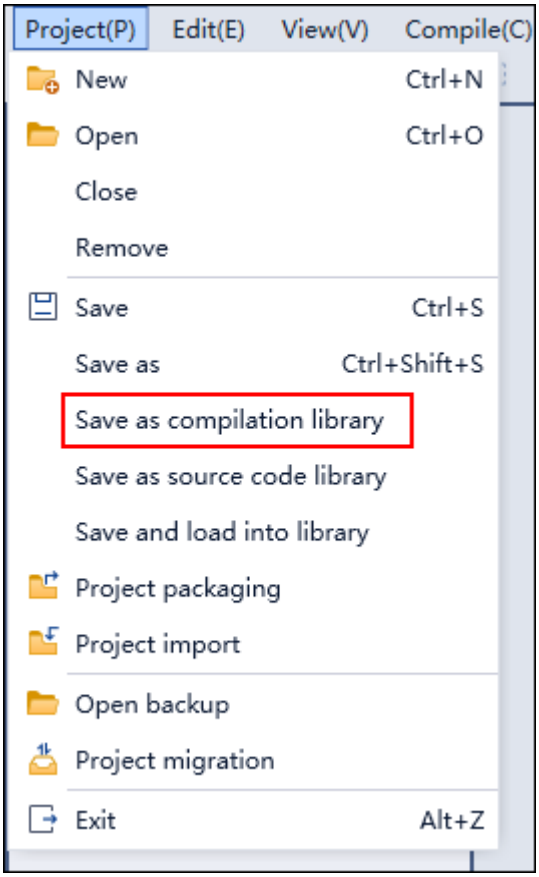
3. On the opened "Project info" interface, enter the company name, library name, version (such as 1.0.0.1), author, and description. The library name serves as the unique identifier for the library, and a library may have multiple version numbers.



4. Choose "Compile > Check program" from the menu bar.



5. Choose "Project > Save as compilation library" from the menu bar after confirming no compilation errors.



6. Select a path, enter a filename, and then save it. The file format is .inoblib.

Result

A file (xxx.inoblib) is generated in the specified folder.

12.1.4 Source Code Library

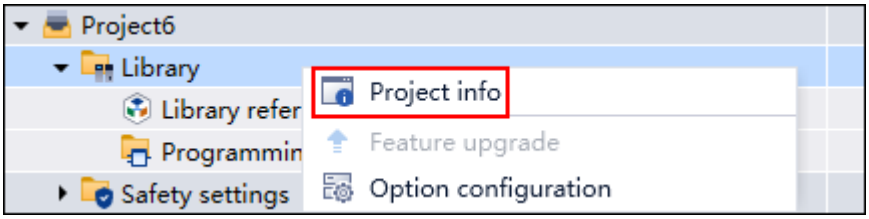
The source code library refers to uncompiled library files (typically in .inolib format) for a project, and can be used for project commissioning.

Prerequisites

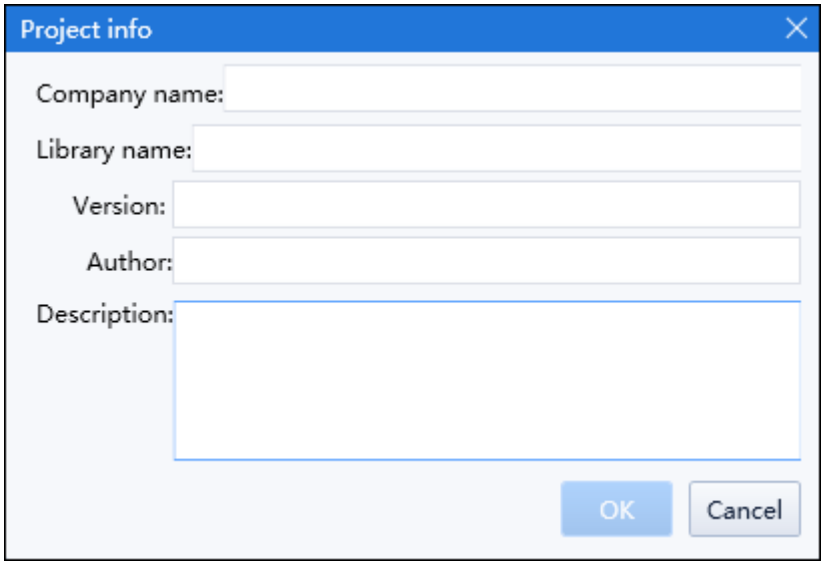
A library project has been created or opened.

Procedure

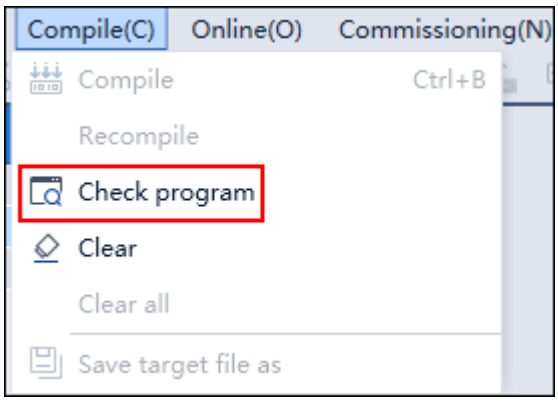
1. Add the program logic, such as program blocks and GV, and perform the logic configuration.
2. Right-click "Library" and then click "Project info".



3. On the opened "Project info" interface, enter the company name, library name, version (such as 1.0.0.1), author, and description. The library name serves as the unique identifier for the library, and a library may have multiple version numbers.

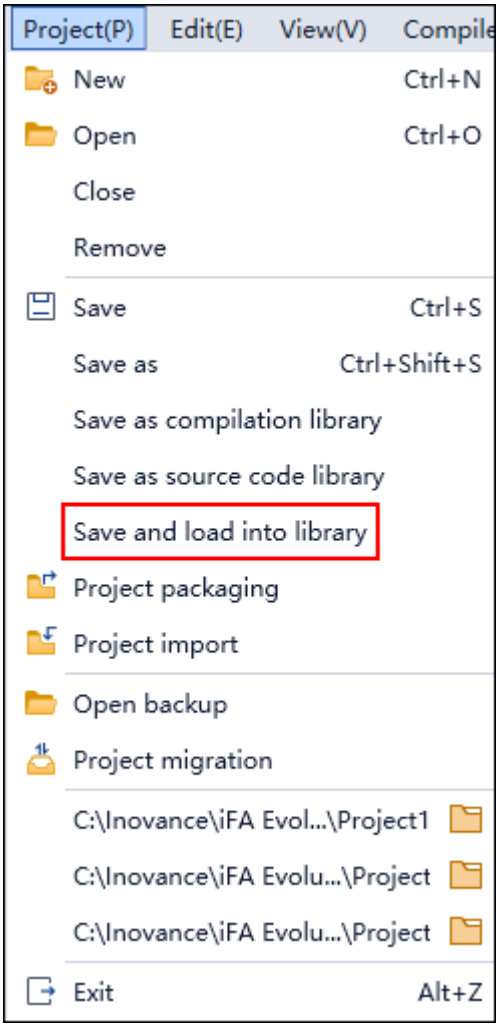


4. Choose "Compile > Check program" from the menu bar.

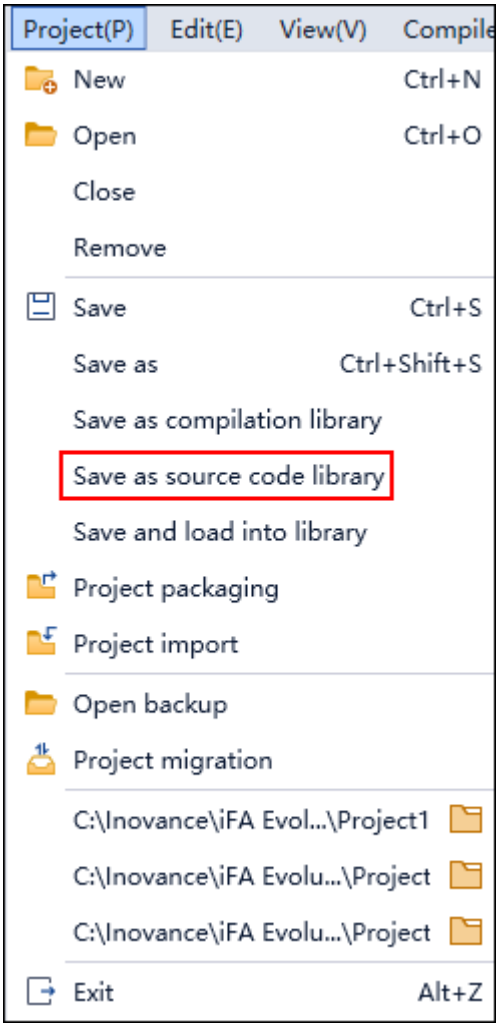


5. Save the source code library in two ways after confirming no compilation errors:

   - Method 1: Save and load into library
     Choose "Project > Save and load into library" from the menu bar. If a library with the same name and version number already exists, a prompt will appear asking whether to overwrite it. This method directly saves the source code library to the system directory, allowing users to view and use it through the "Reference manager".



- Method 2: Save as source code library
  Choose "Project > Save as source code library" from the menu bar, select a path, enter a filename, and then save it. The file format is .inolib. This method saves the source code library to a user-defined path, enabling flexible export and management of files.



Results

- Method 1: Save and load into library
  - A file (.inolib) is generated in the project folder XXX\Repository\Libraries\XXX\.
  - The library information, including library name (entered in Step 3), version number, and author, can be viewed in the "Reference manager".
- Method 2: Save as source code library
  A file (xxx.inolib) is generated in the specified folder.

12.2 Library Manager

The Library Manager allows one-click installation/uninstallation and supports quick import of self-defined libraries into the current project, facilitating direct invocation of installed libraries and improving the development efficiency.

Installation

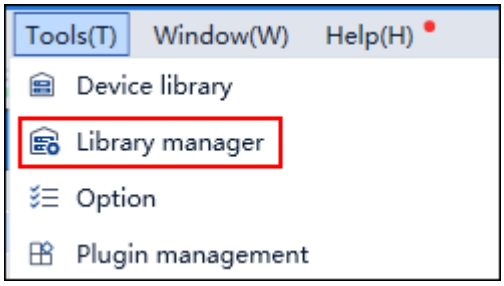
Installation refers to importing user-defined algorithm libraries into the current IDE environment for building user programs in a project.

Prerequisites

User-defined algorithm libraries (source code libraries or compiled libraries) are available in a computer, which means the .inolib file or .inoblib file is present.

Procedure

1. Choose "Tools > Library manager" from the menu bar to access the "Library manager" configuration interface.



2. In the opened "Library manager" interface, click the "Install" button.
3. Select the path of the custom algorithm library.
4. Select .inolib or .inoblib file, and then click the "Open" button in the dialog box.

Result

The installed user-defined libraries are displayed under the "Users" node of the Library Manager.

Uninstallation

Uninstallation refers to removing user-defined libraries from the current IDE environment, after which no information related to these libraries can be identified in the system. System libraries cannot be uninstalled.

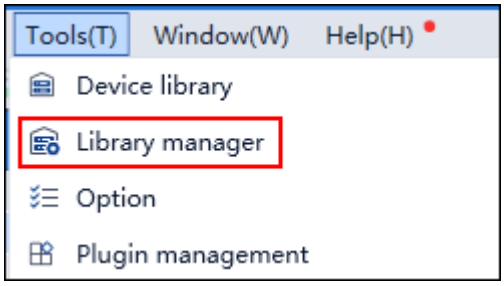
> CAUTION
>
> Uninstalling a referenced library may cause a compilation error in the project program.

Prerequisites

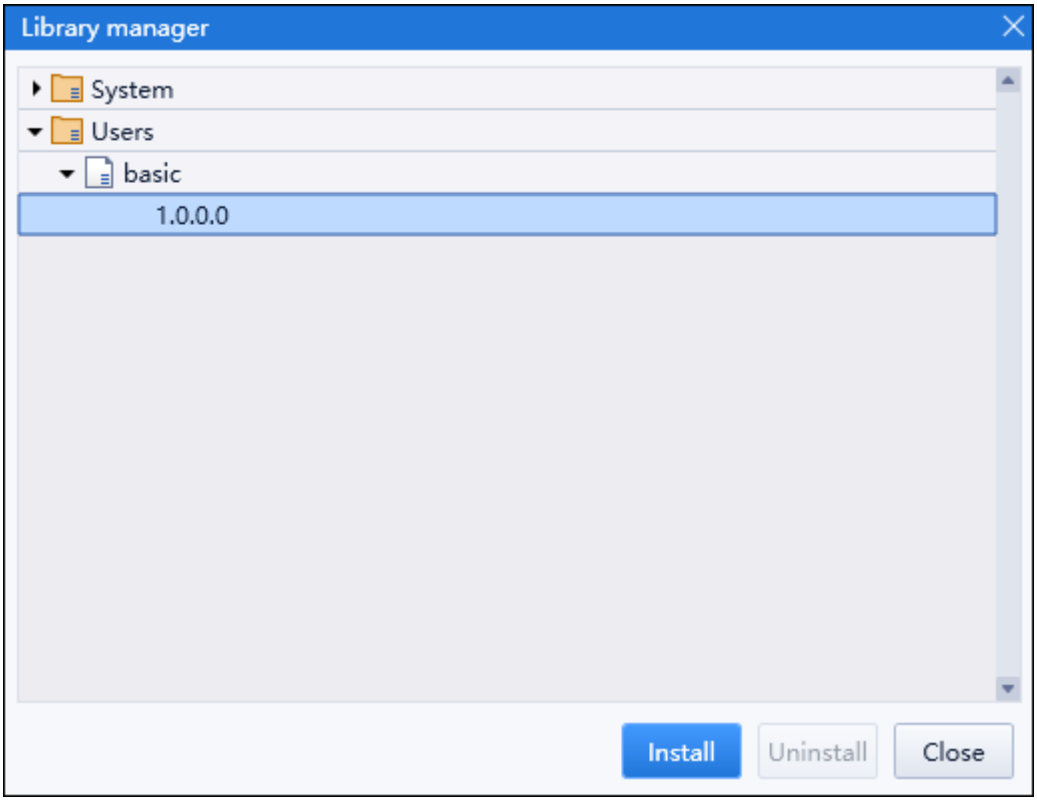
User-defined algorithm libraries (source code libraries or compiled libraries) have been installed.

Procedure

1. Choose "Tools > Library manager" from the menu bar to access the "Library manager" configuration interface.



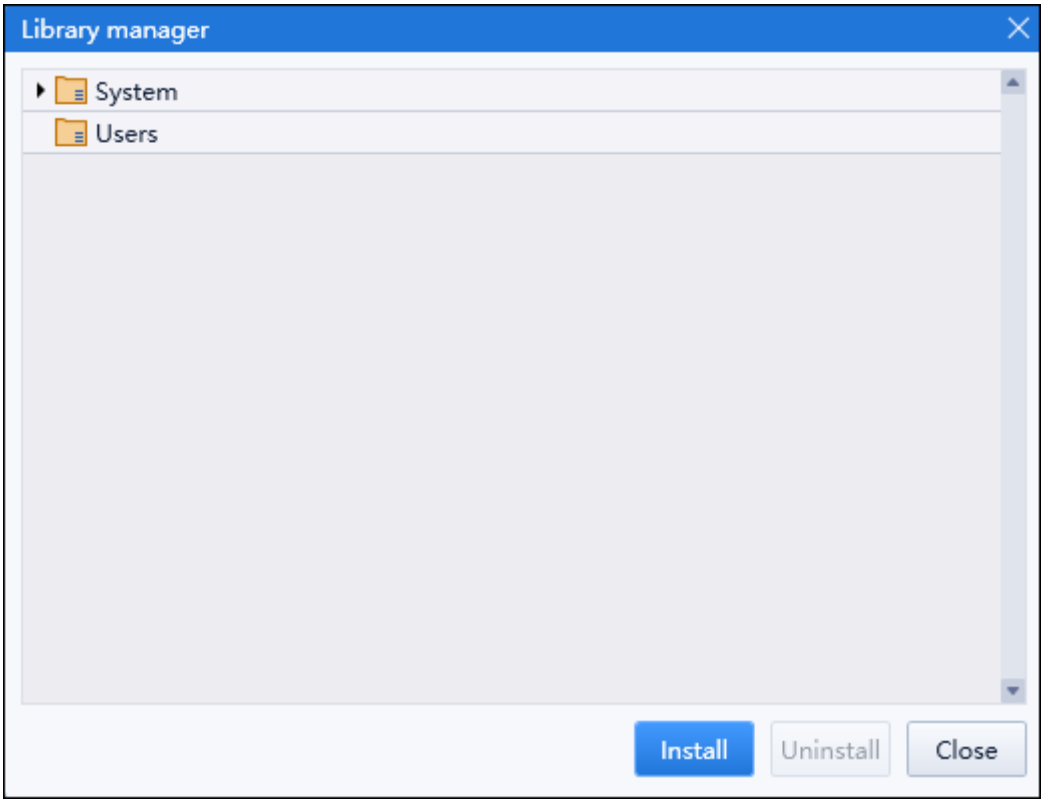
2. In the opened "Library manager" interface, select the version numbers of the libraries under the "Users" node.



3. Click the "Uninstall" button.

Result

- The uninstalled user-defined libraries are no longer displayed under the "Users" node of the Library Manager.
- The program cannot invoke any uninstalled libraries.



12.3 Library Reference

Library reference manages the libraries used in a project, supporting operations on system libraries or self-defined libraries, including referencing, dereferencing, one-click upgrade/downgrade, and library version switching. It applies to projects, including library projects and engineering projects. You can reference different libraries or different versions of the same library for different projects based on project requirements.

- Library referencing introduces the Lib symbols in the current IDE environment into the current project environment, facilitating the user configuration logic function. The referenced library information is saved in the project file to ensure recognition and loading next time when the project is opened.
- Dereferencing removes the referenced libraries from the current project, along with their symbols. If the symbol of a dereferenced library is being used, the system will send an error message.



Dereferencing may cause a software runtime exception or compilation error.

- One-click upgrade/downgrade automatically adjusts all library versions to the latest versions supported by the current device version after the device version switching, preventing conflicts caused by version incompatibility.
- Library version switching allows switching the library versions in a project as needed during development. Available library versions include all versions supported by the current device version, as well as the * version, where "*" indicates the latest version supported by the device version.

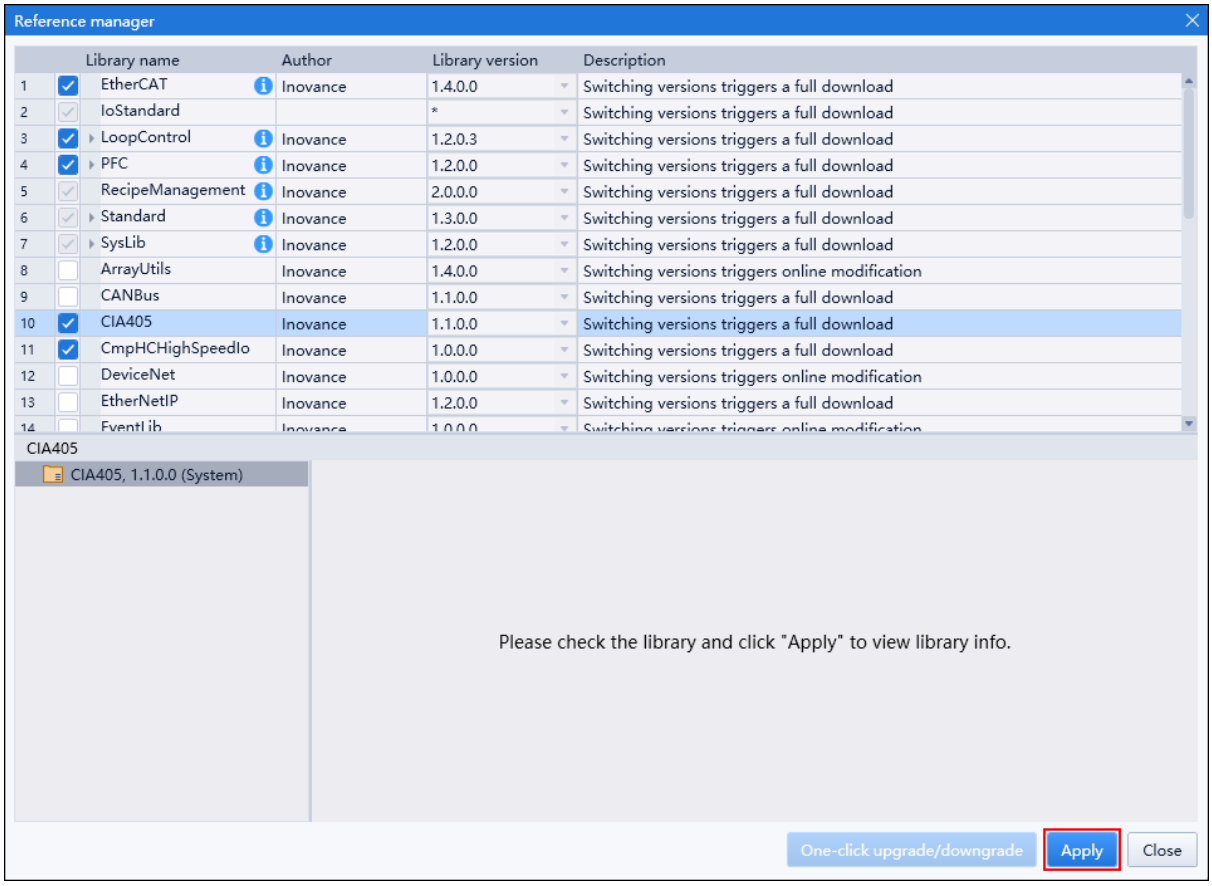
Library referencing and dereferencing

Prerequisites

- System libraries are available, or self-defined libraries have been installed.
- A library project has been opened, or a PLC has been added.

Procedure

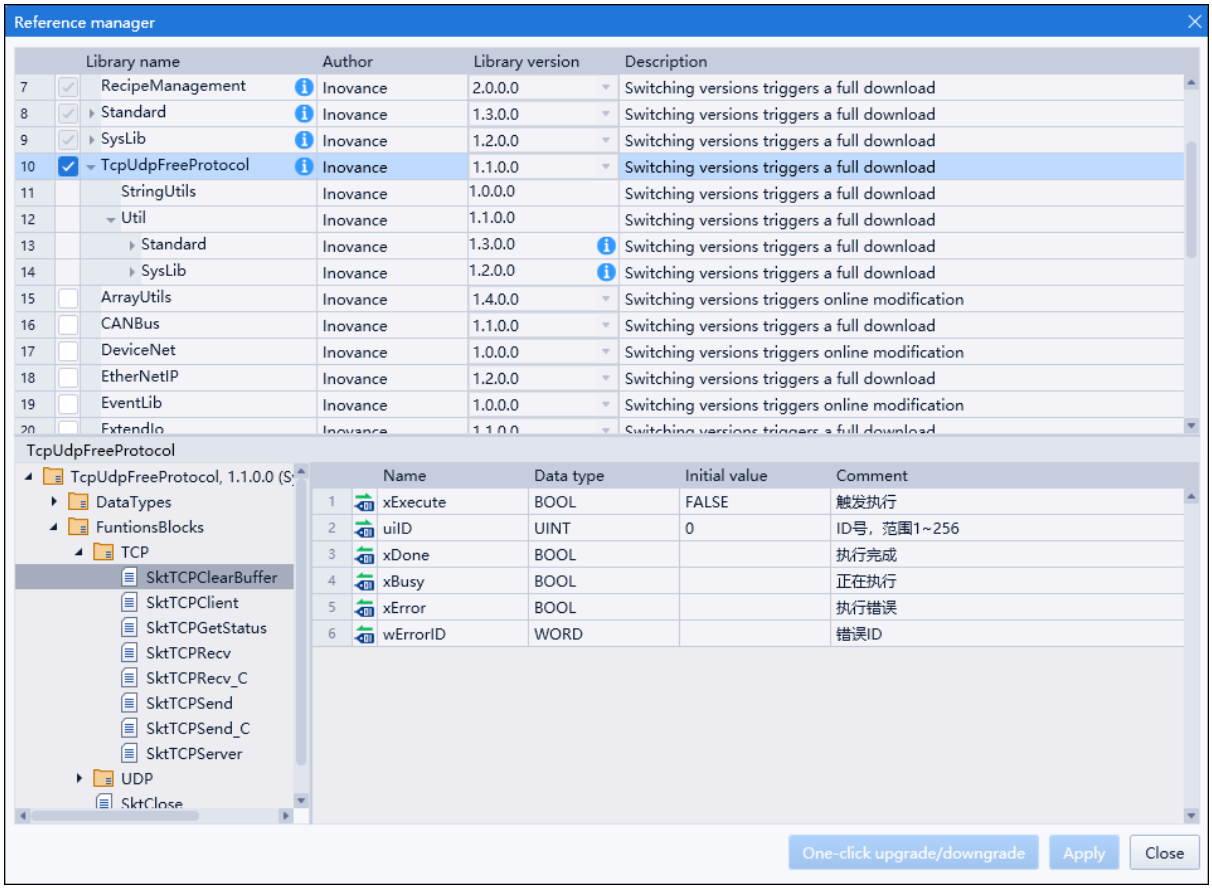
1. Double-click "Library reference" in the left project tree to access the "Reference manager" configuration interface.
2. Check/Uncheck appropriate libraries for reference/dereference.



3. Click the "Apply" button.

Result

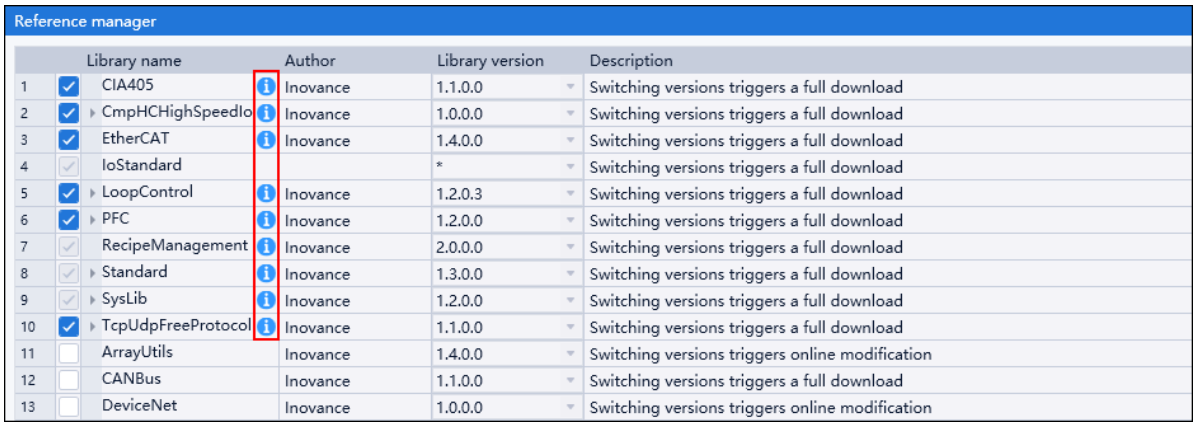
- Referencing: Introduces a selected library into the current project environment, and allows viewing details about supported instructions, variables, user-defined types, and dependent libraries.



- Dereferencing: Removes a referenced library from the current project, along with its symbols.

NOTE

- On the "Reference manager" interface, all installed system libraries and user-defined libraries are displayed, including those referenced for the project. The default library version is the latest, and the "*" in the drop-down list indicates that the latest library version is automatically applied when a new version is installed.
- When referencing system libraries in an engineering project, all versions except the "*" version are set as placeholder libraries by default.



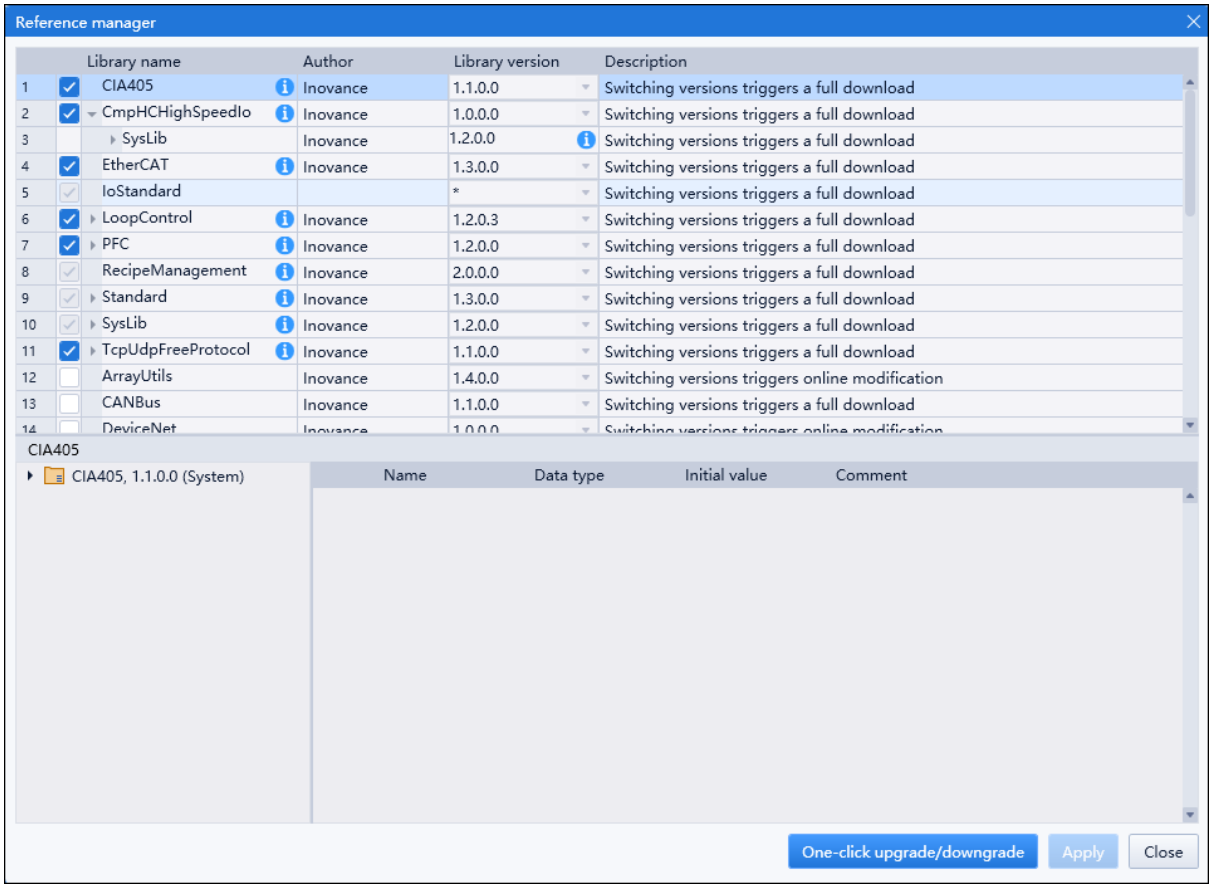
One-click upgrade/downgrade

Prerequisites

A new device version is available.

Procedure

1. Double-click "Library reference" in the left project tree to access the "Reference manager" configuration interface.



2. Click the "One-click Upgrade/Downgrade" button, and then a dialog stating that "Do you want to perform one-click upgrade/downgrade? Click Apply after the upgrade/downgrade" appears. Click "Yes" and then click the "Apply" button.

Result

All libraries supported by this device (irrespective of the selection status) are upgraded to the latest versions (* version remains unchanged).

NOTE

- This feature only supports one-click upgrade/downgrade of system libraries; user-defined libraries are unaffected.
- If any library file is missing after the device version switching, the library version will remain unchanged after the upgrade/downgrade, and an error message will appear during compilation.

Library version switching

Prerequisites

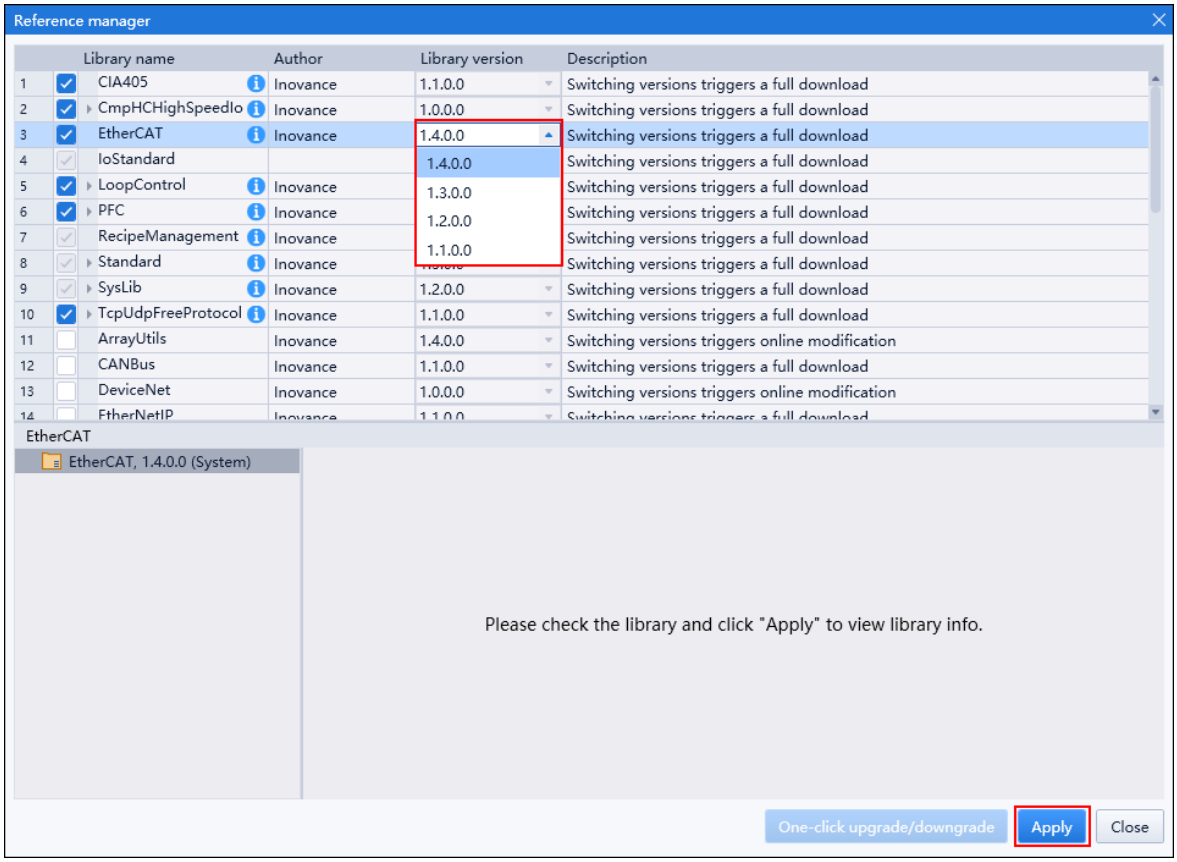
- A library project has been opened, or a PLC has been added.
- System libraries or user-defined libraries have been installed.

Procedure

1. Double-click "Library reference" in the left project tree to access the "Reference manager" configuration interface.

2. You can switch the library version in two ways:

   - Method 1: Switch the library version directly via the drop-down list
     Check a library whose version you want to switch, then select the desired library version from the drop-down list under "Library Version" ("*" indicates using the latest version), and click the "Apply" button.



> **NOTE**
>
> This method only supports modifying the version of top-level libraries via the drop-down list; dependent library versions cannot be directly modified.

   - Method 2: Mark as placeholder library or unmark
     Selected a library, right-click "Mark as placeholder library" or "Unmark," and then a dialog stating that "Confirm that marking will adjust all library versions in the system to the same version" or "Confirm that umarking will restore the library versions in the system to the default version" appears. Follow the prompts to proceed.

> **NOTE**
>
> This method only supports system engineering projects.

Result

The library version in the project is switched to a specified version or the latest version.



Due to differences between library versions, changing the referenced library version may cause a software runtime exception or compilation error.

12.4 Library Properties

Library properties refer to library project information, including company name, library name, version, author, and description.

12.5 Library Permissions

Library permissions refer to read-write protection permissions managed by password for individual POU within a library project, following the read-write permissions of the POU. For details, see 10.1.2.5 Program Block Encryption.

13 Visual Design

13.1 Establishing a Connection

- 13.1.1 Connecting Devices
- 13.1.2 Connecting External Devices
- 13.1.4 Modifying a Connection
- 13.1.5 Copying a Connection
- 13.1.6 Supported Device Drivers

13.1.1 Connecting Devices

- 13.1.1.1 Establishing a Device Connection
- 13.1.1.2 Setting Device Connection Property
- 13.1.1.3 Deleting a Device Connection

13.1.1.1 Establishing a Device Connection

Connection is used to configure the communication link between monitoring devices and other devices (e.g., PLCs), supporting both serial and Ethernet link. A monitoring device can establish up to 32 such links.

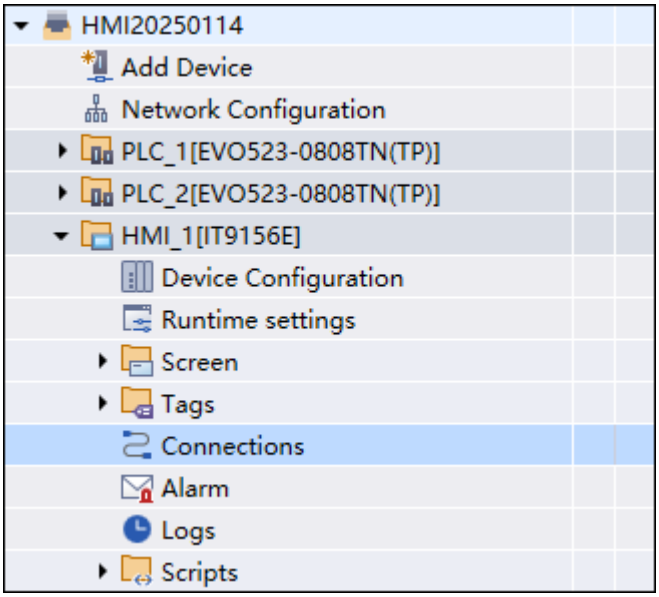
Communication connections between monitoring devices and other devices (e.g., PLC) can be established in two ways: manual and automatic.

Manual: You can configure on the "Connections" page, supporting both serial and Ethernet link.

Automatic: When you configure relevant settings for other devices (e.g., PLCs) on the network view, screen or variable table (limited to default and user-defined variable tables) page, the system automatically triggers the connection. This method supports only Ethernet links.

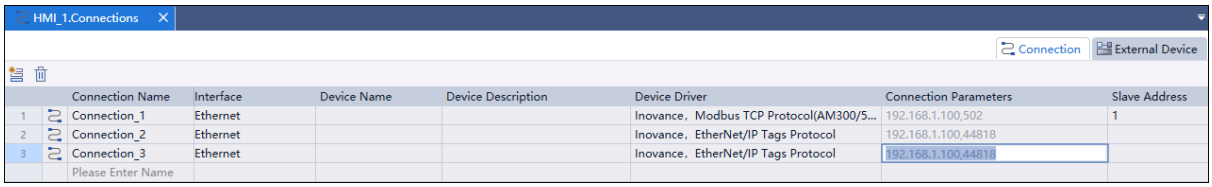
Manually establishing a connection on the "Connections" page

1. Open the "Connections" window.
   Double-click the **Connections** node in the project tree to open the corresponding window.



2. Add a connection.

   Click the icon  to add a connection row. Each added row indicates that the monitored device has established a communication link with a PLC, identified by .



3. Configure parameters.

   The communication properties vary depending on the device driver. For details on property settings, see 13.1.6 Supported Device Drivers.

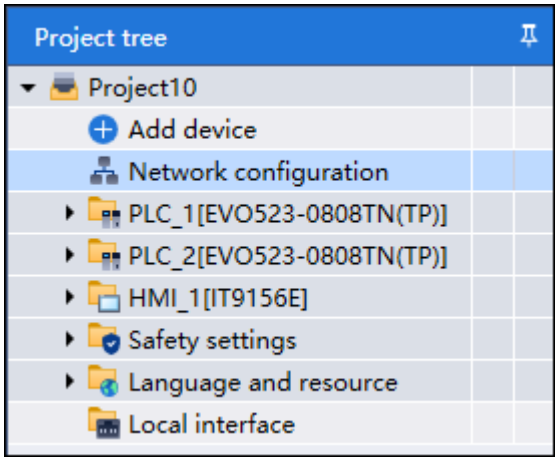
The Ethernet connection between the current monitoring device and a PLC has been successfully established.

Automatically establishing a connection via "Network configuration"

This method is applicable when the PLC and monitoring device belong to the same project and are connected via Ethernet.

1. Open the "Network configuration" window.

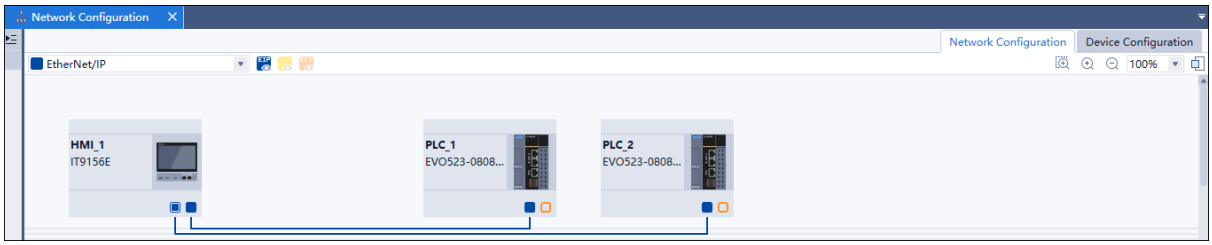
   Double-click the **device and network** node in the project tree to open the "Network configuration" window.



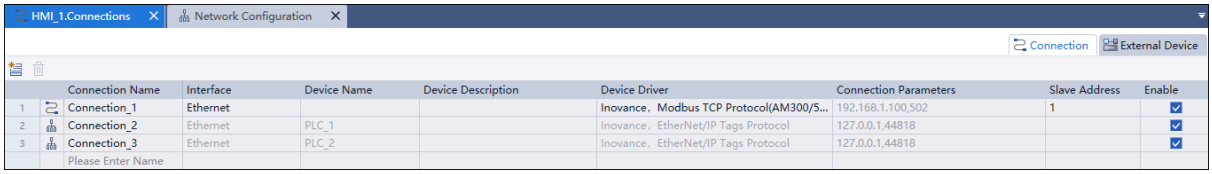
2. Connect devices.

   Manually connect the Ethernet ports of devices. As shown in the figure below, HMI_1 connects to PLC_1

and PLC_2 respectively.



The connection information will be synchronized to the connection window of the monitoring device (HMI_1) and will be identified with , e.g., Connection_2 and Connection_3.



For automatically established connections, the device driver defaults to the 13.1.6.1.7 EtherNet/IP Tags Protocol.

The Ethernet connection between the current monitoring device and a PLC has been successfully established.

Automatically establishing a connection via variable table

This method is applicable when the PLC and monitoring device belong to the same project and are connected via Ethernet.

1. Open the variable table of the monitoring device.
   Double-click the **default variable table** or user variable table node under the variable to open the corresponding window.
2. Add variables.
3. Configure reference variables.
   Double-click the reference variable cell, and select the public variable of the target PLC in the drop-down menu. A connection is established between the current monitoring device and the target PLC. Connection details will be synchronized to the project's "Network view" and the "Connection" window of the monitoring device.
4. Configure connection parameters.
   The connection information identifier is automatically created in the connection window. For parameter settings, see 13.1.6.1.7 EtherNet/IP Tags Protocol.

The Ethernet connection between the current monitoring device and a PLC has been successfully established.

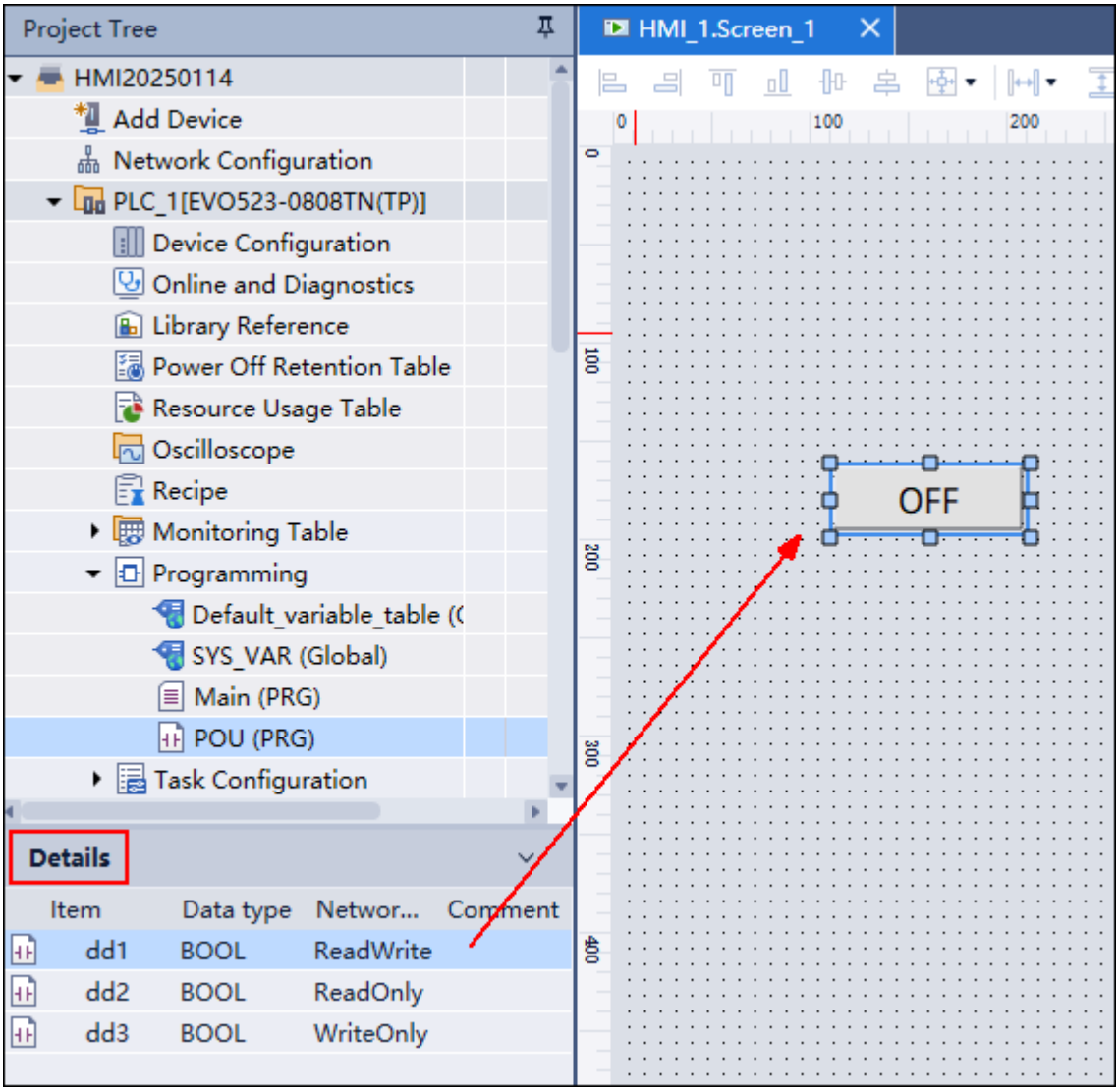
Automatically establishing a connection via screen

This method is applicable when the PLC and monitoring device belong to the same project and are connected via Ethernet.

1. Define variables in PLC_1.
   For the variable definition process, see 10.1.1.1 Global Variables or 10.1.1.2 Local Variables.
2. Set the variable as a public variable.
   Set the variable's "Network public" property to any value other than "Private." For details, see 10.1.1.9.3 Variable Declaration Property.

3. Add a screen to HMI_1.
   For operations, see 13.3.2 Adding a Screen.

4. Drag the variable from PLC_1 to the screen in HMI_1.
   For example, in the figure below, the local variable **dd1** of the POU is dragged from the detailed view to **Screen_1**. The connection between PLC_1 and HMI_1 is successfully established.

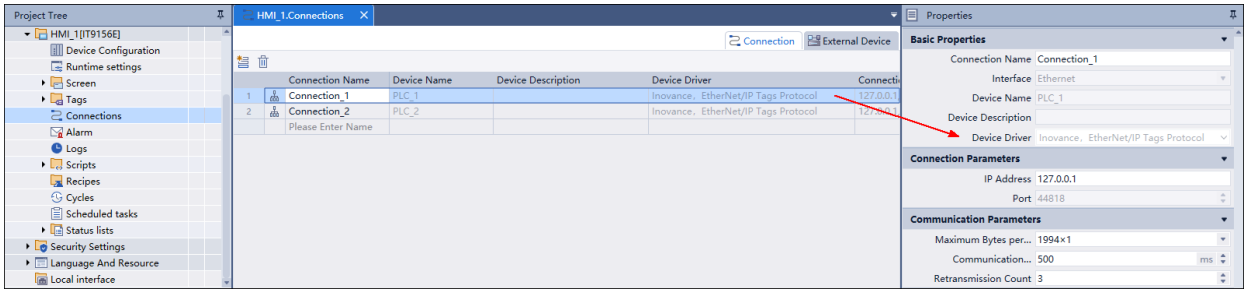


5. Configure connection parameters.
   The connection information identifier is automatically created in the connection window. For parameter settings, see 13.1.6.1.7 EtherNet/IP Tags Protocol.

6. The Ethernet connection between the current monitoring device and a PLC has been successfully established.

13.1.1.2 Setting Device Connection Property

When communication parameters for a monitoring device connected to another device (e.g., PLC) need to be adjusted, double-click the serial number of the target connection row to open the corresponding communication property window.

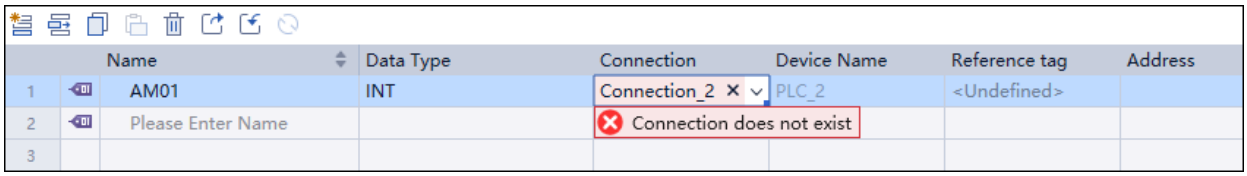


Different selections of device drive will result in differences in the number and content of the corresponding connection property. For details, see the respective sections.

Product	Device Driver	Reference Section
Inovance	TCP Monitoring Protocol (H5U/Easy Series)	For details, see the corresponding section according to the product series. 13.1.6.1.1 Easy Series PLCs and 13.1.6.1.2 H5U Series PLCs
	QLink TCP Protocol (H5U Series)	13.1.6.1.2 H5U Series PLCs
	QLink RTU Protocol (H5U Series)	
	Modbus TCP Protocol (AM400/600 Series)	13.1.6.1.5 AM400 and AM600 Series PLCs
	Modbus RTU Protocol (AM400/600 Series)	
	QLink TCP Protocol (AM400/600 Series)	
	Modbus TCP Protocol (AM300/500, AC700/800 Series)	13.1.6.1.4 AM300 and AM500 Series PLCs, 13.1.6.1.6 AC700 and AC800 Series PLCs The content of the connection property is the same, but there are differences in the variable address configuration.
	Modbus RTU Protocol (AM300/500, AC700/800 Series)	
	QLink TCP Protocol (AM300/500, AC700/800 Series)	
	Modbus TCP Protocol (EVO Series)	13.1.6.1.3 EVO Series PLCs
	Modbus RTU Protocol (EVO Series)	
	EtherNet/IP Tags Protocol	13.1.6.1.7 EtherNet/IP Tags Protocol
	AC800 Tags Protocol (AC/AI/AM/AP Series)	13.1.6.1.8 AC, AI, AM, and AP Series Products
Omron	EtherNet/IP (NJ/NX/NY Series)	13.1.6.4 Communicating with Omron Devices
Siemens	S7-300	13.1.6.2.1 S7-300 PLCs
	S7-1200/1500	13.1.6.2.2 S7-1200 and S7-1500 PLCs
OPC UA	OPC UA Client	13.1.6.5 OPC UA Client
Modbus IDA	Modbus TCP Client	13.1.6.6.1 Modbus TCP Protocol
	Modbus TCP Server	
	Modbus RTU Master	13.1.6.6.2 Modbus RTU Protocol
	Modbus RTU Slave	

13.1.1.3 Deleting a Device Connection

If variables for monitoring devices in the variable table are associated with connections to other devices (e.g., PLC), deleting such connections will cause the corresponding variable rows to display abnormally.



There are three ways to delete other device (e.g., PLC) connections:

1. Select the target row in the "Connections" window, and complete the operation via a command, icon , or the **Delete** key.
2. Delete the connections between devices in the network view to remove the connection to other devices (e.g., PLC).
3. Deleting the connected PLCs in the project tree can also remove the connection to other devices (e.g., PLC).

When a connection to another device (e.g., PLC) is deleted, its corresponding system variables will also be deleted automatically. If a connection with the same name is re-established later, the variable references must be re-established.

13.1.2 Connecting External Devices

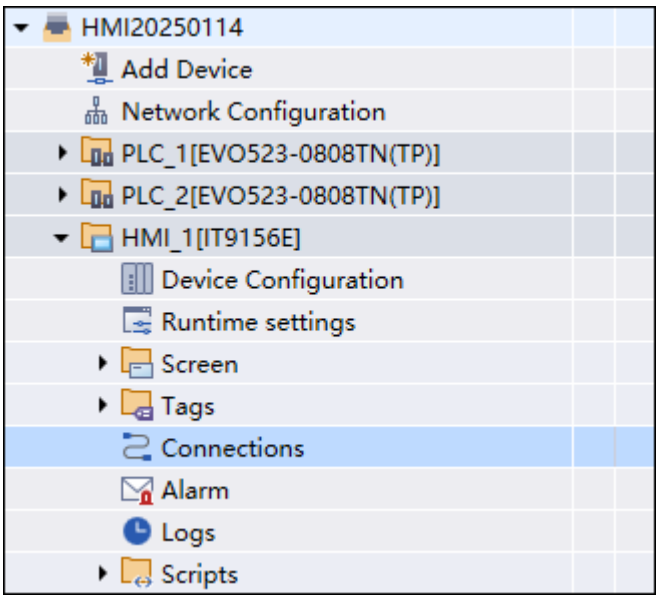
- 13.1.2.1 Establishing an External Device Connection
- 13.1.2.2 Deleting an External Device Connection

13.1.2.1 Establishing an External Device Connection

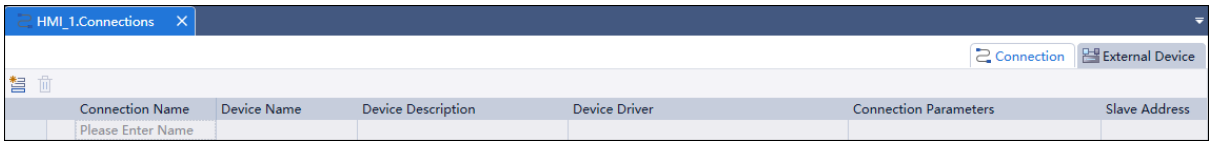
The communication link between the monitoring device and the card reader or barcode scanner must be configured. The devices are connected via USB ports.

Establishing a connection

1. Open the "Connections" window.
   Double-click the **Connections** node in the project tree to open the corresponding window.



2. Switch to the "External device" tab.



3. Add a connection.

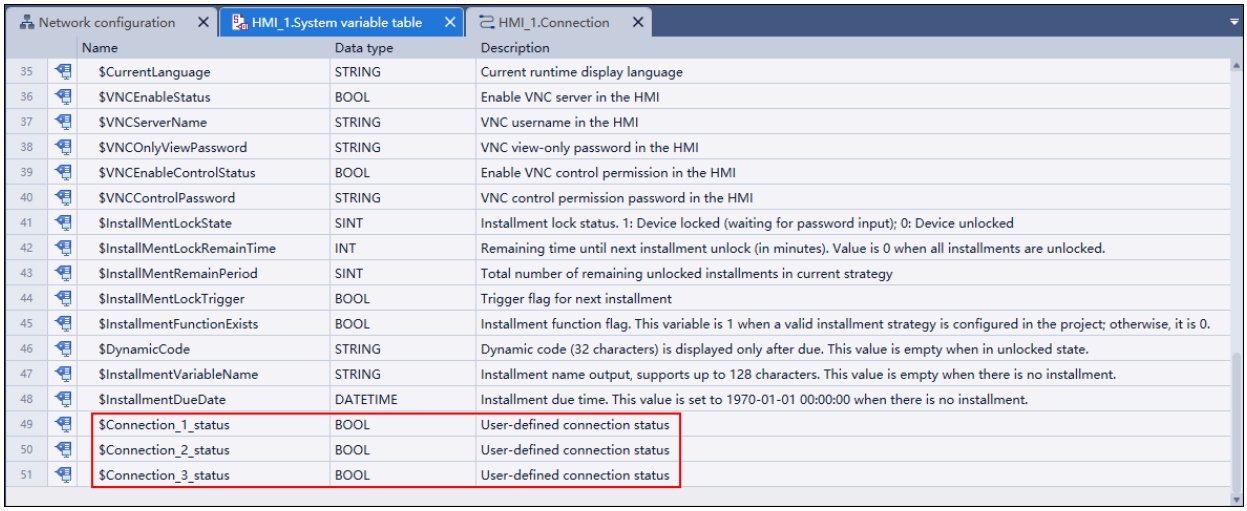
   Click the icon to add a connection row. Each added row indicates that a communication link has been established with an external device, identified by .

4. Configure connection parameters.

Property Category		Description
Basic property	Connection name	Connection identifier supports combinations of Chinese characters, English letters, numbers, `_`, and `-`, with a maximum length of 64 English characters. Connection name must be unique within the same monitoring device. Default name automatically increments based on the number of existing connections (including 13.1.1.1 Connecting Devices), ensuring no duplicates.
	Link type	USB supported
	Device name	Name of the connected object supports combinations of Chinese characters, English letters, numbers, `_`, and `-`, with a maximum length of 64 English characters. Device names do not support duplicates within the same monitoring device.
	Device description	Device description supports a maximum length of 128 characters.
	Device driver	Communication method
Link parameters	Communication port	Connection port selection.
Activate	Enable/Disable	Enable or disable the connection.

Connecting system variables

A corresponding system variable is automatically generated in the "System Variable Table" when an external device connection is established, named as "$Connection Name _status" to indicate the connection status.

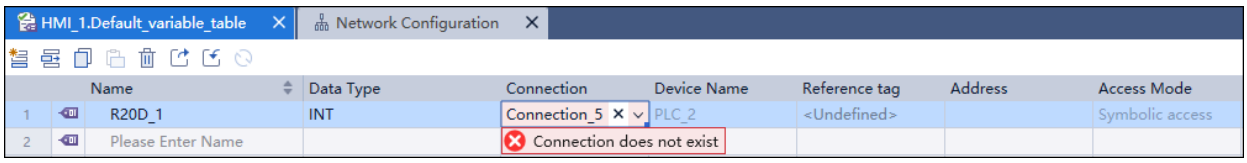


	Name	Data type	Description
35	$CurrentLanguage	STRING	Current runtime display language
36	$VNCEnableStatus	BOOL	Enable VNC server in the HMI
37	$VNCServerName	STRING	VNC username in the HMI
38	$VNCOnlyViewPassword	STRING	VNC view-only password in the HMI
39	$VNCEnableControlStatus	BOOL	Enable VNC control permission in the HMI
40	$VNCControlPassword	STRING	VNC control permission password in the HMI
41	$InstallMentLockState	SINT	Installment lock status. 1: Device locked (waiting for password input); 0: Device unlocked
42	$InstallMentLockRemainTime	INT	Remaining time until next installment unlock (in minutes). Value is 0 when all installments are unlocked.
43	$InstallMentRemainPeriod	SINT	Total number of remaining unlocked installments in current strategy
44	$InstallMentLockTrigger	BOOL	Trigger flag for next installment
45	$InstallmentFunctionExists	BOOL	Installment function flag. This variable is 1 when a valid installment strategy is configured in the project; otherwise, it is 0.
46	$DynamicCode	STRING	Dynamic code (32 characters) is displayed only after due. This value is empty when in unlocked state.
47	$InstallmentVariableName	STRING	Installment name output, supports up to 128 characters. This value is empty when there is no installment.
48	$InstallmentDueDate	DATETIME	Installment due time. This value is set to 1970-01-01 00:00:00 when there is no installment.
49	$Connection_1_status	BOOL	User-defined connection status
50	$Connection_2_status	BOOL	User-defined connection status
51	$Connection_3_status	BOOL	User-defined connection status

For usage constraints, see 13.2.1 System Variable Table.

13.1.2.2 Deleting an External Device Connection

In the variable table, if variables of the monitoring device are associated with an external device connection, deleting the connection will cause the corresponding variable rows to display abnormally.

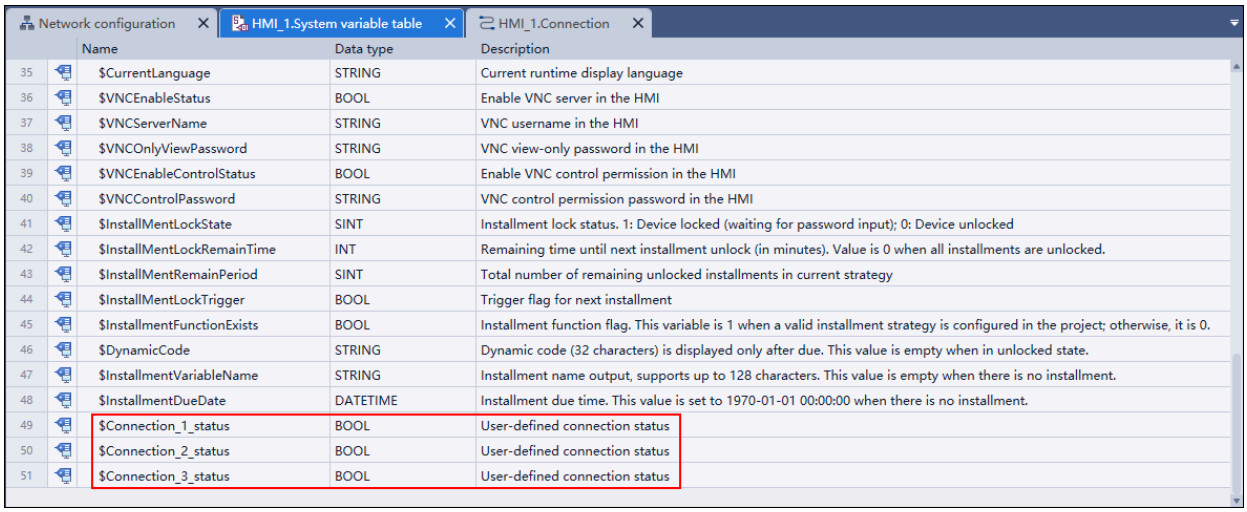


Select the target row in the "Connections" window, and complete the delete operation via a command, icon , or the **Delete** key.

When a connection to external device is deleted, its corresponding system variables will also be deleted automatically. If a connection with the same name is re-established later, the variable references must be re-established.

13.1.3 Adding a System Variable

A corresponding system variable is automatically generated in the "System variable table" when each connection is established, named as "$Connection Name_status" to indicate the connection status.



	Name	Data type	Description
35	$CurrentLanguage	STRING	Current runtime display language
36	$VNCEnableStatus	BOOL	Enable VNC server in the HMI
37	$VNCServerName	STRING	VNC username in the HMI
38	$VNCOnlyViewPassword	STRING	VNC view-only password in the HMI
39	$VNCEnableControlStatus	BOOL	Enable VNC control permission in the HMI
40	$VNCControlPassword	STRING	VNC control permission password in the HMI
41	$InstallMentLockState	SINT	Installment lock status. 1: Device locked (waiting for password input); 0: Device unlocked
42	$InstallMentLockRemainTime	INT	Remaining time until next installment unlock (in minutes). Value is 0 when all installments are unlocked.
43	$InstallMentRemainPeriod	SINT	Total number of remaining unlocked installments in current strategy
44	$InstallMentLockTrigger	BOOL	Trigger flag for next installment
45	$InstallmentFunctionExists	BOOL	Installment function flag. This variable is 1 when a valid installment strategy is configured in the project; otherwise, it is 0.
46	$DynamicCode	STRING	Dynamic code (32 characters) is displayed only after due. This value is empty when in unlocked state.
47	$InstallmentVariableName	STRING	Installment name output, supports up to 128 characters. This value is empty when there is no installment.
48	$InstallmentDueDate	DATETIME	Installment due time. This value is set to 1970-01-01 00:00:00 when there is no installment.
49	$Connection_1_status	BOOL	User-defined connection status
50	$Connection_2_status	BOOL	User-defined connection status
51	$Connection_3_status	BOOL	User-defined connection status

For usage constraints, see 13.2.1 System Variable Table.

13.1.4 Modifying a Connection

In the variable table, when variables of the monitoring device are associated with a connection, the impact of modifying connection parameters are as follows:

- Connection name change: The iFA software will automatically synchronize the change to the variable table (including the system variable table).
- Device driver change: The addresses in the variable table (excluding the system variable table) will revert to default values. For the address description, see variable Connection property.

13.1.5 Copying a Connection

Other devices (e.g., PLC) connected internally to the monitoring device do not support copy operations. For example, Connection_1 of HMI_1 device does not support copying.

When the entire device (HMI_1) is copied, the copied objects include identifiers  or connection information.

13.1.6 Supported Device Drivers

Different driver enable different device connections. You can choose a specific communication interface, configuration and transmission speed for an individual device. Supported protocol types in the visualization platform are shown in the table below:

Table13-1-6-1

Connection Type	Device Manufacturer/ Organization	Device Model	Supported Protocol	Visualization Device Type
Serial port	Inovance Technology	AM300, AM500 Series	Modbus-RTU	HMI
		AM400, AM600 Series	Modbus-RTU Protocol	HMI
		AC700, AC800 Series	Modbus-RTU Protocol	HMI
		EVO Series	Modbus-RTU Protocol	HMI
		H5U Series	QLink RTU Protocol	HMI
		MD Series	Modbus RTU Protocol	HMI
	Modbus IDA	/	Modbus RTU Master	HMI
		/	Modbus RTU Slave	HMI
	Mitsubishi	FX3U/FX3UC Series	FX485BD Protocol	HMI
Ethernet	Inovance Technology	AC700, AC800 Series	QLink TCP Protocol AC800 Tags Protocol Modbus TCP Protocol EtherNet/IP Tags Protocol	HMI

Connection Type	Device Manufacturer/ Organization	Device Model	Supported Protocol	Visualization Device Type
		AI800	AC800 Tags Protocol EtherNet/IP Tags Protocol	HMI
		AP700 Series	AC800 Tags Protocol EtherNet/IP Tags Protocol	HMI
		AM300, AM500 Series	QLink TCP Protocol AC800 Tags Protocol Modbus TCP Protocol EtherNet/IP Tags Protocol	HMI
		AM400, AM600 Series	QLink TCP Protocol AC800 Tags Protocol Modbus TCP Protocol EtherNet/IP Tags Protocol	HMI
		Easy series PLCs	TCP Monitoring Protocol EtherNet/IP Tags Protocol	HMI
		EVO Series	Modbus TCP Protocol EtherNet/IP Tags Protocol	HMI
		H5U Series	TCP Monitoring Protocol QLink TCP Protocol EtherNet/IP Tags Protocol	HMI
	Omron	NJ Series, NX Series, or XY Series	EtherNet/IP	HMI
	Siemens	S7-1200 or S7-1500	/	HMI
		S7-300	/	HMI
	Rockwell (AB)	/	EtherNet/IP	HMI
	Modbus IDA	/	Modbus TCP Client	HMI
		/	Modbus TCP Server	HMI
	OPC UA	/	OPC UA Client	HMI
	Mitsubishi	FX5U	FX5U (Binary) Protocol	HMI
		Q Series	MelSec-Q/L (Binary) Protocol	HMI
USB	/		Card reader	HMI
			Barcode scanner	HMI

13.1.6.1 Communicating with Inovance PLCs

13.1.6.1.1 Easy Series PLCs

When an HMI device communicates with an Easy series PLC, select the device driver labeled "Easy" to establish the connection. The connection link is Ethernet.

Communication link configuration

The Ethernet communication connection property of Easy series PLCs is the same as those of the AM300 series. For details, see Ethernet Connection Property of AM300 series.

Variable address configuration

To ensure correct data transmission across protocols, see the table below when configuring variable addresses.

Table13-1-6-1-1-1 Easy series PLCs – Modbus communication address lookup table

Access Type		Function Code			Address Range	Quantity	PLC Memory	
		Read	Single Write	Multiple Write			Type	Range
Bit access	Coil	0x01 or 0x02	0x05	0x0F	0x0000 to 0x1F3F	8000	M	M0 to M7999
					0x3000 to 0xAFFF	32768	B	B0 to B32767
					0xE000 to 0xEFFF	4096	S	S0 to S4095
					0xF800 to 0xFBFF	1024	X	X0 to X1777 (octal)
					0xFC00 to 0xFFFF	1024	Y	Y0 to Y1777 (octal)
Word access	Holding register	0x03 or 0x04	0x06	0x10	0x0000 to 0x1F3F	8000	D	D0 to D7999
					0x3000 to 0xAFFF	32768	R	R0 to R32767

13.1.6.1.2 H5U Series PLCs

When an HMI device communicates with an H5U series PLC, select the device driver labeled "H5U" to establish the connection. Connection links are divided into Ethernet and serial port, and the connection properties differ.

Communication link configuration

The Ethernet communication connection property of H5U series PLCs is the same as those of the AM300 series. For details, see Ethernet Connection Property of AM300 series.

The serial port communication connection property of H5U series PLCs is the same as those of the AM300 series. For details, see Serial Port Connection Property of the AM300 series.

Variable function code configuration

When an H5U series PLC communicates with an HMI using the Modbus protocol, the function codes are the same as those of the Easy series PLCs. For the correspondence between function codes and addresses, see Easy series PLCs – Modbus communication address lookup table .

13.1.6.1.3 EVO Series PLCs

When an HMI device communicates with an EVO series PLC, select the device drive labeled "EVO" to establish the connection. Connection links are divided into Ethernet and serial port, and the connection properties differ.

Communication link configuration

The Ethernet communication connection property of EVO series PLCs is the same as those of the AM300 series. For details, see Ethernet Connection Property of AM300 series.

The serial port communication connection property of EVO series PLCs is the same as those of the AM300 series. For details, see Serial Port Connection Property of AM300 series.

Variable address configuration

To ensure correct data transmission across protocols, see the table below when configuring variable addresses.

Table13-1-6-1-3-1 EVO series PLCs – Modbus communication address lookup table

Register Type	Function Code			Address Range	Quantity	PLC Memory	
	Read	Single Write	Multiple Write			Type	Range
Discrete input	0x02	/	/	0x0000 to 0xFFFF	65536	I[1]	IX0.0 to IX8191.7
Coil	0x01	0x05	0x0F	0x0000 to 0xFFFF	65536	Q[2]	QX0.0 to QX8191.7
Input register	0x04	/	/	0x0000 to 0xFFFF	65536	MW	MW0 to MW65535
Holding register	0x03	0x06	0x10	0x0000 to 0xFFFF	65536	MW	MW0 to MW65535

[1]: The I area supports two data access modes: bit access (IX) and word access (IW). The bit access address range is shown in the table above; the word access address ranges from IW0 to IW8192. Each word contains two bytes. For example, IW0 covers the range from IW0.0 to IW0.7, and from IW1.0 to IW1.7.

[2]: The Q area supports two data access modes: bit access (QX) and word access (QW). The bit access address range is shown in the table above; the word access address ranges from QW0 to QW8192. Each word contains two bytes. For example, QW0 covers the range from QW0.0 to QW0.7, and from QW1.0 to QW1.7.

13.1.6.1.4 AM300 and AM500 Series PLCs

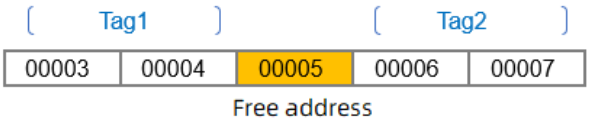
When an HMI device communicates with an AM300 or AM500 series PLC, select the device drive labeled "AM300/500" to establish the connection. Connection links are divided into Ethernet and serial port, and the connection properties differ.

Communication link configuration

Ethernet configuration

Table13-1-6-1-4-1 Description of Ethernet link connection property

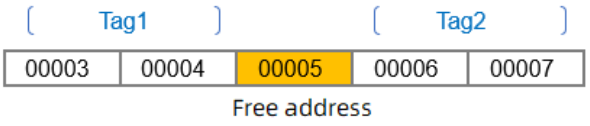
Property Category		Description
Basic property	Connection name	Connection identifier supports combinations of Chinese characters, English letters, numbers, `_`, and `-`, with a maximum length of 64 English characters. Connection name must be unique within the same monitoring device. Default name automatically increments based on the number of existing connections (including 13.1.2.1 Connecting External Devices), ensuring no duplicates.
	Link type	Ethernet
	Device name	Name of the connected object supports combinations of Chinese characters, English letters, numbers, `_`, and `-`, with a maximum length of 64 English characters. Device names do not support duplicates within the same monitoring device.
	Device description	Device description supports a maximum length of 128 characters.
	Device driver	Modbus TCP protocol (AM300/500, AC700/800 series) or QLink TCP protocol (AM300/500, AC700/800 series)
Link parameters	IP address	The access address of the connected object.
	Port number	Setting range: 0 to 65535. Default: 502.
	Slave address	Unique number of a slave device. The master station communicates with a specific slave device via the station number. The setting range of station number is related with **Device Driver**. The default value is 1. For example, the station range for Modbus protocol is 1 to 255.

Property Category		Description
Communication parameter	Address interval	Applicable to both Modbus protocol and QLink protocol. This parameter refers to the number of idle addresses between adjacent data (addresses). When the number of idle addresses is less than or equal to the setting value, adjacent data will be merged into a single packet for transmission. The number of data packets affects the data transfer rate. More data packets result in a lower transfer rate. Setting range: 0 to 24, where 0 indicates each packet contains only one data. If set to 3, when reading two words of data Tag1 and Tag2 from address 0x00003 and 0x00006 respectively, with only one idle address 0x00005 in between, the packet merging condition is satisfied, meaning Tag1 and Tag2 can be sent in one combined packet. 
	Max. read (bytes)	The maximum number of bytes per data packet. The setting range is determined by **Device Driver**.
	Max. write (bytes)	
	Connection timeout	The client does not receive a response from the server within the specified time after sending a connection request during the establishment of a Modbus TCP/IP connection. Time unit: milliseconds (ms). Setting range: 1 to 3600000.
	Communication timeout	The master station does not receive a response from the slave within the specified time after sending a request. Time unit: milliseconds (ms). Setting range: 1 to 3600000.
	Retransmission count	Number of retries for resending commands when device response (e.g., PLC) data is abnormal. Up to 10 attempts are supported.
	Connection attempt interval	Reconnection interval after communication is disconnected. Time unit: seconds (s). Setting range: 1 to 3600000.
Activate	Enable/Disable	Enable or disable the connection.

Serial port configuration

Table13-1-6-1-4-2 Description of serial link connection property

Property Category		Description
Basic property	Connection name	Connection identifier supports combinations of Chinese characters, English letters, numbers, `_`, and `-`, with a maximum length of 64 English characters. Connection name must be unique within the same monitoring device. Default name automatically increments based on the number of existing connections (including 13.1.2.1 Connecting External Devices), ensuring no duplicates.
	Link type	Serial port
	Device name	Name of the connected object supports combinations of Chinese characters, English letters, numbers, `_`, and `-`, with a maximum length of 64 English characters. Device names do not support duplicates within the same monitoring device.
	Device description	Device description supports a maximum length of 128 characters.
	Device driver	Modbus RTU protocol (AM300/500, AC700/800 series)
Link parameters	Communication port	Supports COM1 to COM4.
	Type	Supports RS232, RS422 and RS485. The communication port is automatically adapted without manual configuration. COM1 in RS422 mode and COM4 in RS485 mode cannot be used simultaneously.
	Baud rate	Default: 9600 bps.
	Data bit	Supports 7 and 8. Defaults to 8.
	Stop bit	Supports 1 and 2. Defaults to 1.
	Parity check	Supports no parity, odd parity, and even parity. The default is no parity.
	Slave address	Unique number of a slave device. The master station communicates with a specific slave device via the station number. Station number range:1 to 255. Default: 1.

Property Category		Description
Communication parameter	Address interval	This parameter refers to the number of idle addresses between adjacent data (addresses). When the number of idle addresses is less than or equal to the setting value, adjacent data will be merged into a single packet for transmission. The number of data packets affects the data transfer rate. More data packets result in a lower transfer rate. Setting range: 0 to 24, where 0 indicates each packet contains only one data. If set to 3, when reading two words of data Tag1 and Tag2 from address 0x00003 and 0x00006 respectively, with only one idle address 0x00005 in between, the packet merging condition is satisfied, meaning Tag1 and Tag2 can be sent in one combined packet. 
	Max. read (bytes)	The maximum number of bytes per data packet. The setting range is determined by **Device Driver**.
	Max. write (bytes)	
	Max. bytes per packet	The maximum number of bytes per packet for EtherNet/IP Tags protocol.
	Communication timeout	The master station does not receive a response from the slave within the specified time after sending a request. Time unit: milliseconds (ms). Setting range: 1 to 3600000.
	Retransmission count	Number of retries for resending commands when PLC response data is abnormal. Up to 10 attempts are supported.
	Connection attempt interval	Reconnection interval after communication is disconnected. Time unit: seconds (s). Setting range: 1 to 36000.
	Device detection cycle	Device detection cycle after the device goes offline. Time unit: seconds (s). Setting range: 1 to 36000.
Activate	Enable/Disable	Enable or disable the connection.

Variable address configuration

To ensure correct data transmission across protocols, see the table below when configuring variable addresses.

Modbus communication address

Table13-1-6-1-4-3 AM 300 and AM 500 series PLCs – Modbus communication address lookup table

Access Type		Function Code			Address Range	Quantity	PLC Memory	
		Read	Single Write	Multiple Write			Type	Range
Bit Access	Discrete input	0x02	/	/	0x0000 to 0xFFFF	65536	IX	IX0.0 to IX8191.7
	Coil	0x01	0x05	0x0F	0x0000 to 0xFFFF	65536	QX	QX0.0 to QX8191.7
	Extension (coil)	0x21	0x25	0x2F	0x0000 to 0xFFFF	65536	MX	MX0.0 to MX8191.7
	Extension (coil)	0x51	0x55	0x5F	0x0000 to 0xFFFF	65536	MX	MX131072 to MX139263.7
Word access	Holding register	0x03	0x06	0x10	0x0000 to 0xFFFF	65536	MW	MW0 to MW65535
	Extension (holding register)	0x53	0x56	0x50	0x0000 to 0xFFFF	65536	MW	MW65536 to MW131071

QLink TCP communication address

Table13-1-6-1-4-4 AM 300 and AM 500 series PLCs – QLink TCP communication address lookup table

Access Type		Function Code			Address Range	Quantity	PLC Memory	
		Read	Single Write	Multiple Write			Type	Range
Bit Access	Discrete input	0x02	/	/	0x0000 to 0xFFFF	65536	IX	IX0.0 to IX8191.7
	Coil	0x01	0x05	0x0F	0x0000 to 0xFFFF	65536	QX	QX0.0 to QX8191.7
Word access	Holding register	0x03	0x06	0x10	0x0000 to 0xFFFF	65536	MW	MW0 to MW65535

13.1.6.1.5 AM400 and AM600 Series PLCs

When an HMI device communicates with an AM400 or AM600 series PLC, select the device drive labeled "AM400/600" to establish the connection. Connection links are divided into Ethernet and serial port, and the connection properties differ.

Communication link configuration

The Ethernet communication connection property of AM400/AM600 series PLCs is the same as those of the AM300 series. For details, see Ethernet Connection Property of AM300 series.

The serial port communication connection property of AM400/AM600 series PLCs is the same as those of the AM300 series. For details, see Serial Port Connection Property of AM300 series.

Variable address configuration

To ensure correct data transmission across protocols, see the table below when configuring variable addresses.

Modbus communication address

Table13-1-6-1-5-1 AM 400 and AM 600 series PLCs – Modbus communication address lookup table

Access Type		Function Code			Address Range	Quantity	PLC Memory	
		Read	Single Write	Multiple Write			Type	Range
Bit Access	Discrete input	0x02	/	/	0x0000 to 0xFFFF	65536	IX	IX0.0 to IX8191.7
	Coil	0x01	0x05	0x0F	0x0000 to 0xFFFF	65536	QX	QX0.0 to QX8191.7
	Extension (coil)	0x21	0x25	0x2F	0x0000 to 0xFFFF	65536	MX	MX0.0 to MX8191.7
	Extension (coil)	0x31	0x35	0x3F	0x0000 to 0x1F3F	8000	SM	SM0 to SM7999
Word access	Holding register	0x03	0x06	0x10	0x0000 to 0xFFFF	65536	MW	MW0 to MW65535
	Extension (holding register)	0x33	0x36	0x40	0x0000 to 0x1F3F	8000	SD	SD0 to SD7999

QLink TCP communication address

Table13-1-6-1-5-2 AM 400 and AM 600 series PLCs – QLink TCP communication address lookup table

Access Type		Function Code			Address Range	Quantity	PLC Memory	
		Read	Single Write	Multiple Write			Type	Range
Bit Access	Discrete input	0x02	/	/	0x0000 to 0xFFFF	65536	IX	IX0.0 to IX8191.7
	Coil	0x01	0x05	0x0F	0x0000 to 0xFFFF	65536	QX	QX0.0 to QX8191.7
Word access	Holding register	0x03	0x06	0x10	0x0000 to 0xFFFF	65536	MW	MW0 to MW65535

13.1.6.1.6 AC700 and AC800 Series PLCs

When an HMI device communicates with an AC700 or AC800 series PLC, select the device drive labeled "AC700/800" to establish the connection. Connection links are divided into Ethernet and serial port, and the connection properties differ.

Communication link configuration

The Ethernet communication connection property of AC700/800 series PLCs are the same as those of the AM300 series. For details, see Ethernet Connection Property of AM300 series.

The serial port communication connection property of AC700/AC800 series PLCs is the same as those of the AM300 series. For details, see Serial Port Connection Property of AM300 series.

Variable address configuration

To ensure correct data transmission across protocols, see the table below when configuring variable addresses.

Modbus communication address

Table13-1-6-1-6-1 AC 700 and AC 800 series PLCs – Modbus communication address lookup table

Access Type		Function Code			Address Range	Quantity	PLC Memory	
		Read	Single Write	Multiple Write			Type	Range
Bit Access	Discrete input	0x02	/	/	0x0000 to 0xFFFF	65536	IX	IX0.0 to IX8191.7
	Coil	0x01	0x05	0x0F	0x0000 to 0xFFFF	65536	QX	QX0.0 to QX8191.7
	Extension (coil)	0x21	0x25	0x2F	0x0000 to 0x0FFF	4096	MX	MX0.0 to MX8191.7
	Extension (coil)	0x51	0x55	0x5F	0x0000 to 0x0FFF	4096	MX	MX131072 to MX139263.7
Word access	Holding register	0x03	0x06	0x10	0x0000 to 0xFFFF	65536	MW	MW0 to MW65535
	Extension (holding register)	0x53	0x56	0x50	0x0000 to 0x0FFF	4096	MW	MW65536 to MW131071

QLink TCP communication address

Table13-1-6-1-6-2 AC 700 and AC 800 series PLCs – QLink TCP communication address lookup table

Access Type		Function Code			Address Range	Quantity	PLC Memory	
		Read	Single Write	Multiple Write			Type	Range
Bit Access	Discrete input	0x02	/	/	0x0000 to 0xFFFF	65536	IX	IX0.0 to IX8191.7
	Coil	0x01	0x05	0x0F	0x0000 to 0xFFFF	65536	QX	QX0.0 to QX8191.7
Word access	Holding register	0x03	0x06	0x10	0x0000 to 0xFFFF	65536	MW	MW0 to MW65535

13.1.6.1.7 EtherNet/IP Tags Protocol

The EtherNet/IP Tags protocol is applicable to all Inovance series PLCs, with Ethernet as the communication link. This protocol supports display messages in source/destination mode, with the message type being Class 3.

For the setting requirements of connection property, see the descriptions in the table.

Table13-1-6-1-7-1 Connection property description

Property Category		Description
Basic property	Connection name	Connection identifier supports combinations of Chinese characters, English letters, numbers, `_`, and `-`, with a maximum length of 64 English characters. Connection name must be unique within the same monitoring device. Default name automatically increments based on the number of existing connections (including 13.1.2.1 Connecting External Devices), ensuring no duplicates.
	Link type	Supports Ethernet
	Device name	Name of the connected object supports combinations of Chinese characters, English letters, numbers, `_`, and `-`, with a maximum length of 64 English characters. Device names do not support duplicates within the same monitoring device.
	Device description	Device description supports a maximum length of 128 characters.
	Device driver	Inovance Ethernet/IP Tags Protocol
Link parameters	IP address	The access address of the connected object.
	Port number	44818, and not editable.
Communication parameter	Max. bytes per packet	Multiple specifications are available for the max. bytes per packet. If the setting here exceeds the maximum specification of the PLCs, the software will automatically split and reassemble packets according to the PLC specifications to ensure normal data transmission. During the packet splitting process, the system will trigger a prompt and display it in the system alarm pop-up window.
	Communication timeout	The master station does not receive a response from the slave within the specified time after sending a request. Time unit: milliseconds (ms). Setting range: 1 to 3600000.
	Retransmission count	Number of retries for resending commands when PLC response data is abnormal. Up to 10 attempts are supported.
Tag	Import tags	Import XML file of tag variables.

Property Category		Description
	Target tag variable table	Displays the name of the connection to which the current tag belongs. For example: Connection_5. It will also be automatically updated when the tag variable table is renamed.
Activate	Enable/Disable	Enable or disable the connection.

13.1.6.1.8 AC, AI, AM, and AP Series Products

The AC, AI, AM, and AP series devices include the following models:

- AC Series: AC700 and AC800 Series
- AI Series: AI800 Series
- AM Series: AM300, AM500, AM400, and AM600 Series
- AP Series: AP700 Series

When an HMI device communicates with these series of devices select the device drive labeled "AC800 Tags" to establish the connection. The connection link is Ethernet.

For the setting requirements of connection property, see the descriptions in the table.

Table13-1-6-1-8-1 Connection property description

Property Category		Description
Basic property	Connection name	Connection identifier supports combinations of Chinese characters, English letters, numbers, `_`, and `-`, with a maximum length of 64 English characters. Connection name must be unique within the same monitoring device. Default name automatically increments based on the number of existing connections (including 13.1.2.1 Connecting External Devices), ensuring no duplicates.
	Link type	Supports Ethernet
	Device name	Name of the connected object supports combinations of Chinese characters, English letters, numbers, `_`, and `-`, with a maximum length of 64 English characters. Device names do not support duplicates within the same monitoring device.
	Device description	Device description supports a maximum length of 128 characters.
	Device driver	Inovance AC800 Tags Protocol (AC/AI/AM/AP Series Products)
Link parameters	IP address	Access address of the connected object

Property Category		Description
Authentication settings	Authentication method	Supports anonymous access and password authentication.
	Username	Password is displayed in ciphertext.
	Password	
Tag	Encoding format	Supports GB2312 and UTF-8.
	Import tags	Import XML file of tag variables.
	Target tag variable table	Displays the name of the connection to which the current tag belongs. For example: Connection_5. It will also be automatically updated when the tag variable table is renamed.
Activate	Enable/ Disable	Enable or disable the connection.

13.1.6.2 Communicating with Siemens S7 Series PLCs

13.1.6.2.1 S7-300 PLCs

When an HMI device communicates with an S7-300 PLC, select the "Siemens S7-300" device driver to establish the connection. For the setting requirements of connection property, see the descriptions in the table.

Connection property

Table13-1-6-2-1-1 Description of S7-300 connection property

Property Category		Description
Basic property	Connection name	Connection identifier supports combinations of Chinese characters, English letters, numbers, `_`, and `-`, with a maximum length of 64 English characters. Connection name must be unique within the same monitoring device. Default name automatically increments based on the number of existing connections (including 13.1.2.1 Connecting External Devices), ensuring no duplicates.

Property Category		Description
	Connection type	Supports serial port and Ethernet.
	Device name	Name of the connected object supports combinations of Chinese characters, English letters, numbers, `_`, and `-`, with a maximum length of 64 English characters. Device names do not support duplicates within the same monitoring device.
	Device description	Device description supports a maximum length of 128 characters.
	Device driver	Communication protocol
Link parameters	IP address	The access address of the connected object.
	Port number	Setting range: 0 to 65535. Default: 102.
Communication parameter	Connection timeout	When establishing a connection, no response is received from the server within the specified time after the client sends a connection request. Time unit: milliseconds (ms). Setting range: 1 to 3600000.
	Communication timeout	The master station does not receive a response from the slave within the specified time after sending a request. Time unit: milliseconds (ms). Setting range: 1 to 3600000.
	Retransmission count	Number of retries for resending commands when PLC response data is abnormal. Up to 10 attempts are supported.
	Connection attempt interval	Reconnection interval after communication is disconnected. Time unit: seconds (s). Setting range: 1 to 3600000.
Activate	Enable/ Disable	Enable or disable the connection.

Setting variable address

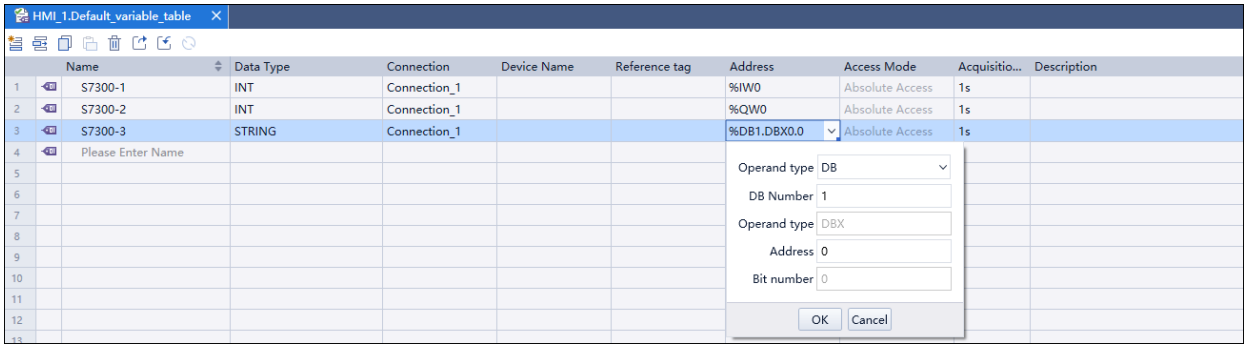
When an HMI product communicates with an S7-300 PLC, the data types supported for communication between the PLC memory areas and the HMI are listed in the table below.

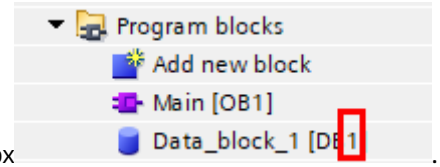
S7-300 Memory Area	Data Type	Remarks
I area	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, and LREAL	
Q area		
M area		
Zone DB	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, and STRING	

CAUTION

The STRING type on the HMI side uses UTF-8 encoding, while the PLC side uses GBK encoding.

In the HMI-side variable table, the addresses of S7-300 PLC variables need to be configured. The configuration requirements for each parameter are as follows.



Parameter	Data Type
Area	PLC variable memory area
DB number	Data block (DB) number, e.g. the selected number in the box  .
Operand type	Software automatically matches.
Address	Absolute address
Bit	Bit number of the Boolean variable.

13.1.6.2.2 S7-1200 and S7-15001500 PLCs

When an HMI device communicates with an S7-1200 or S7-1500 PLC, select the "Siemens S7-1200/1500" device driver to establish the connection.

Connection property

The connection properties of the "Siemens S7-1200/1500" device driver are the same as those of "Siemens S7-300". For details, see S7-300 Connection property.

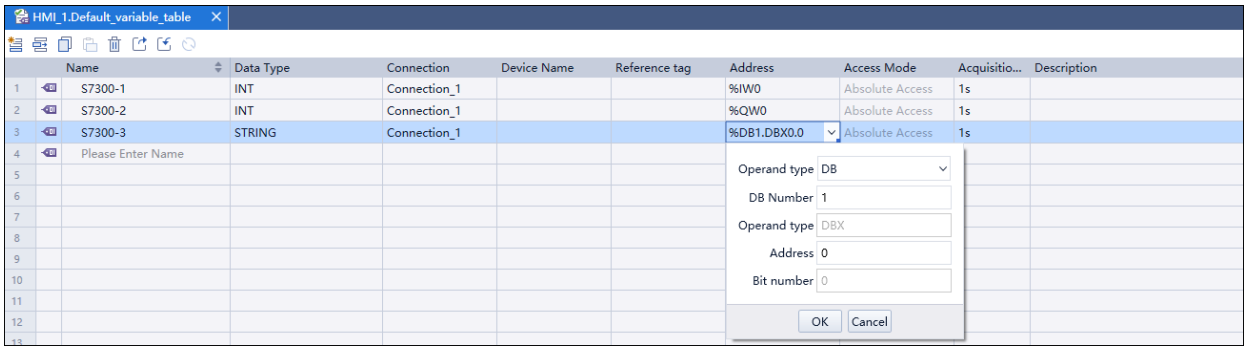
Variable address

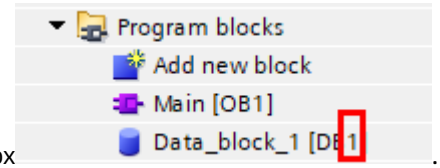
The data types supported by each PLC memory area for communication with the HMI are listed in the table below.

PLC Memory Area	Data Type	Remarks
I area	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, and LREAL	
Q area		
M area		

PLC Memory Area	Data Type	Remarks
Zone DB	BOOL, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL, STRING, and WSTRING	For the STRING type, the maximum length is 256 bytes, with the first two bytes not storing text. For the WSTRING type, the maximum length is 256 bytes, with the first two characters not storing text.

In the HMI-side variable table, the addresses of S7-12/1500 PLC variables need to be configured. The configuration requirements for each parameter are as follows.



Parameter	Data Type
Area	PLC variable memory area
DB number	Data block (DB) number, e.g. the selected number in the box  .
Operand type	Software automatically matches.
Address	Absolute address
Bit	Bit number of the Boolean variable.

13.1.6.3 Communicating with Rockwell (AB) Devices

- 13.1.6.3.1 Configuring Connection Properties
- 13.1.6.3.2 Importing Tag Variables

13.1.6.3.1 Configuring Connection Properties

When an HMI device communicates with a Rockwell ControlLogix series 1756 PLC, select the "Rockwell (AB) EtherNet/IP" device driver to establish the connection. For the setting requirements of connection property, see the descriptions in the table.

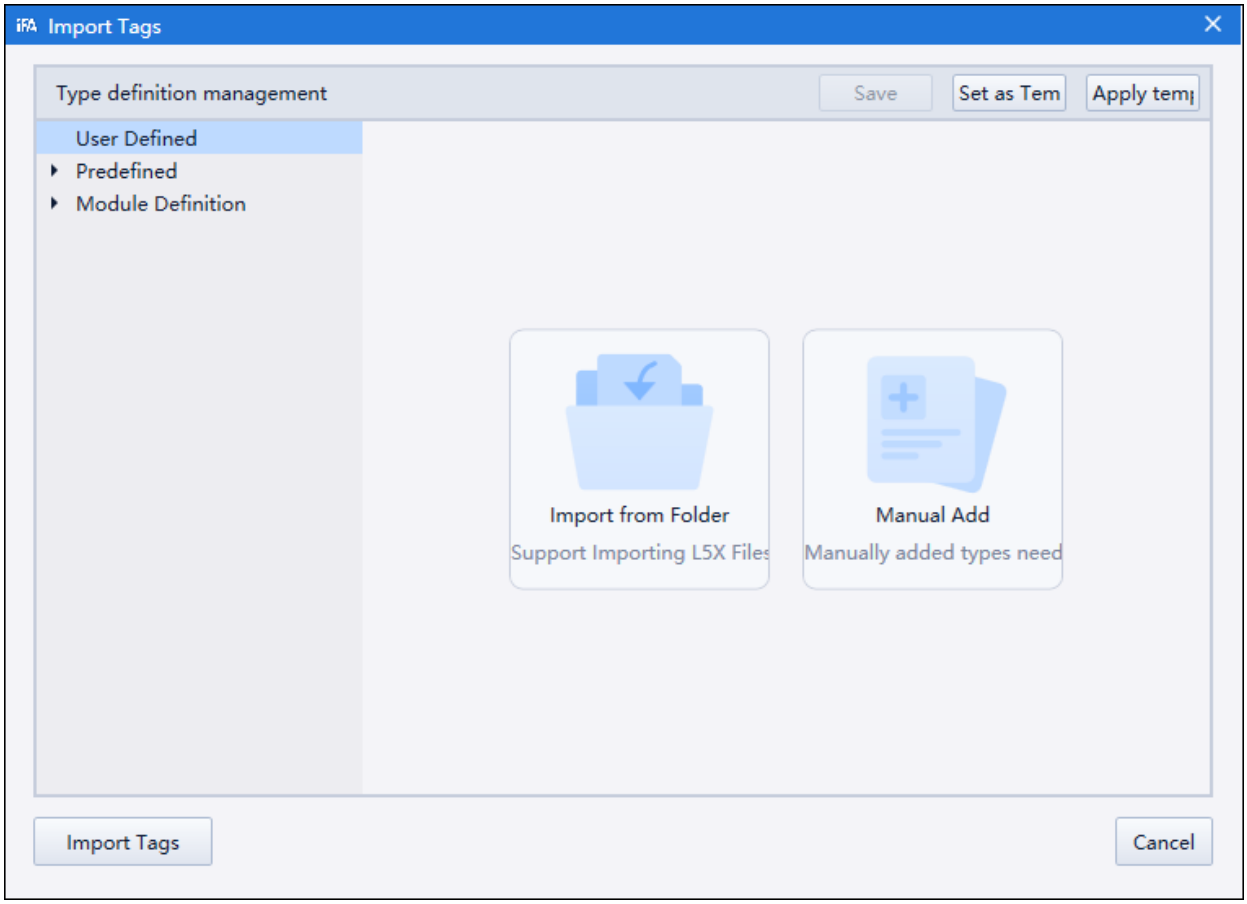
Table13-1-6-3-1-1 Connection property description

Property Category		Description
Basic property	Connection name	Connection identifier supports combinations of Chinese characters, English letters, numbers, `_`, and `-`, with a maximum length of 64 English characters. Connection name must be unique within the same monitoring device. Default name automatically increments based on the number of existing connections (including 13.1.2.1 Connecting External Devices), ensuring no duplicates.
	Connection type	Supports Ethernet
	Device name	Name of the connected object supports combinations of Chinese characters, English letters, numbers, `_`, and `-`, with a maximum length of 64 English characters. Device names do not support duplicates within the same monitoring device.
	Device description	Device description supports a maximum length of 128 characters.
	Device driver	Communication protocol
Link parameters	IP address	The access address of the connected object.
	Port number	Setting range: 0 to 65535. Default: 44818.
Communication parameter	Communication timeout	The master station does not receive a response from the slave within the specified time after sending a request. Time unit: milliseconds (ms). Setting range: 1 to 3600000.
	Retransmission count	Number of retries for resending commands when PLC response data is abnormal. Up to 10 attempts are supported.
Tag	Import tags	For related operations, see 13.1.6.3.2 Importing Tag Variables.
	Target tag variable table	Displays the name of the connection to which the current tag belongs. For example: Connection_5. It will also be automatically updated when the tag variable table is renamed.
Activate	Enable/Disable	Enable or disable the connection.

13.1.6.3.2 Importing Tag Variables

The tag variable import method for the Rockwell (AB) EtherNet/IP device driver differs from that for other EtherNet/IP device drivers. It imports structure definitions and structure variables separately.

Select the "Import tag" button in the property area to open the corresponding window.



Operations related to structure definitions

These operations are completed in the "Type definition management" area.

Three types of structure definitions:

User-defined: Struct types created by users according to control requirements, corresponding to the struct types under the “Data Types–User-defined” node in Studio 5000 software.

Predefined: Corresponds to the built-in types in Studio 5000 software, i.e., the structure types under the node Data Types – Predefined.

Module-defined: Structure types associated with specific hardware modules, corresponding to types compatible with Rockwell ControlLogix series 1756 PLC, excluding basic types.

Table13-1-6-3-2-1 Related operations and requirements

Operation	Applicable Objects	Requirements
Import	User-defined type	Scope: Definitions where the read/write flag is not "None". Method: Incremental import. Duplicates will be overwritten. File Format: L5X format.
Manual addition	Supported by all three types	Directly copy structure member definitions from Studio 5000 software and paste into the current window.

Operation	Applicable Objects	Requirements
New	Supported by all three types	Create a new structure definition in the current window. Naming requirements: Type name consists of letters, numbers, "_", and ":", and must not begin with a number or a ":". Name must be unique. Maximum name length is 100 characters. The member name consists of letters, digits, and "_", and must not begin with an underscore. Maximum name length is 100 characters. Up to 512 members supported.
Edit	Supported by all three types	Modify members of existing types.
Set as template	All types	Encapsulate all current types into a template, which will be updated into the current software. That is, recreate the project in the current software.
Apply template	Connection	"Apply template" operation automatically updates the current connection with the manually set template.

Importing structure variables

Click the "Import tag" button in the lower-left corner of the window to import. The file format is CSV. Batch import is supported.

Data type

Data type correspondence before and after import

Before Import		After Import
Integer	SINT	Unchanged
	INT	Unchanged
	DINT	Unchanged
	LINT	Unchanged
	USINT	Unchanged
	UINT	Unchanged
	UDINT	Unchanged
	ULINT	Unchanged
Decimal	REAL	Unchanged
	LREAL	Unchanged
Boolean	BOOL	Unchanged
Text	STRING	Fixed 82 characters
	STRING_16	Fixed 16 characters
	STRING_**32**	Fixed 32 characters

Before Import		After Import
Array	ARRAY	DINT[2,3,36] before import is converted to ARRAY OF format. All array sizes must not exceed 2M, with a maximum quantity of 65535.

13.1.6.4 Communicating with Omron Devices

When HMI devices communicate with Omron NJ, NX, and NY series PLCs, select the "Omron, EtherNet/IP (NJ/NX/NY series)" device drivers to establish the connection. For the setting requirements of connection property, see the descriptions in the table.

Table13-1-6-4-1 Connection property description

Property Category		Description
Basic property	Connection name	Connection identifier supports combinations of Chinese characters, English letters, numbers, `_`, and `-`, with a maximum length of 64 English characters. Connection name must be unique within the same monitoring device. Default name automatically increments based on the number of existing connections (including 13.1.2.1 Connecting External Devices), ensuring no duplicates.
	Link type	Supports Ethernet
	Device name	Name of the connected object supports combinations of Chinese characters, English letters, numbers, `_`, and `-`, with a maximum length of 64 English characters. Device names do not support duplicates within the same monitoring device.
	Device description	Device description supports a maximum length of 128 characters.
	Device driver	Communication protocol
Link parameters	IP address	The access address of the connected object.
	Port number	Setting range: 0 to 65535. Default: 44818.
Communication parameter	Communication rate	Standard rate: Data transmission is based on the standard EtherNet/IP protocol. High-speed rate: Under the same conditions, data transmission rate is higher than standard rate.
	Communication timeout	The master station does not receive a response from the slave within the specified time after sending a request. Time unit: milliseconds (ms). Setting range: 1 to 3600000.
	Retransmission count	Number of retries for resending commands when PLC response data is abnormal. Up to 10 attempts are supported.

Property Category		Description
Tag	Import tags	Import the text file (.txt) containing tag variable definitions. The file must be encoded in UTF-8.
	Target tag variable table	Displays the name of the connection to which the current tag belongs. For example: Connection_5. It will also be automatically updated when the tag variable table is renamed.
Activate	Enable/Disable	Enable or disable the connection.



When multiple EtherNet/IP (NJ/NX/NY series) protocol connections are established between the same HMI device and the same Omron PLC, ensure that the communication rate configuration is consistent across all connections. That is, set all to high-speed rate or all to standard rate. Do not mix two communication rates to avoid possible communication abnormalities.

13.1.6.5 OPC UA Client

OPC UA (OPC Unified Architecture) is a widely used communication protocol in industrial automation, enabling secure and reliable data exchange. When an HMI communicates with a PLC, the HMI acts as the client.

For the configuration requirements of OPC UA communication properties, see the descriptions in the table.

Table13-1-6-5-1 Communication property description

Property Category		Description
Basic property	Connection name	Connection identifier supports combinations of Chinese characters, English letters, numbers, `_`, and `-`, with a maximum length of 64 English characters. Connection name must be unique within the same monitoring device. Default name automatically increments based on the number of existing connections (including 13.1.2.1 Connecting External Devices), ensuring no duplicates.
	Link type	Supports serial port and Ethernet.
	Device name	Name of the connected object supports combinations of Chinese characters, English letters, numbers, `_`, and `-`, with a maximum length of 64 English characters. Device names do not support duplicates within the same monitoring device.
	Device description	Device description supports a maximum length of 128 characters.

Property Category		Description
	Device driver	The role of HMI device in communication. Select "OPC UA, OPC UA Client".
Link parameters	URL	OPC server address The address prefix is fixed as **opc.tcp://**, and the address length supports up to 256 bytes. Valid address example 1:: opc.tcp://192.168.1.99:48010 Valid address example 2: opc.tcp://test01:4840 URL
Security settings	Security policy	If either the security policy or security mode is set to "None", the other must also be set to "None". In other cases, the two parameters can be set to different options independently.
	Safety mode	
Authentication settings	Authentication method	Supports anonymous access and password authentication.
	Username	When authentication method is username/password, the password is displayed in ciphertext.
	Password	
Data access	Access mode	Supports subscription and active reading; subscription mode requires configuration of **publishing interval**.
	Publish interval	The subscription parameter is in milliseconds (ms). Setting range: 1 to 3600000.
Tag	Import tags	Import XML file of tag variables.
	Target tag variable table	Displays the name of the connection to which the current tag belongs. For example: Connection_5. It will also be automatically updated when the tag variable table is renamed.
Activate	Enable/Disable	Enable or disable the connection.

13.1.6.6 Modbus IDA

- 13.1.6.6.1 Modbus TCP Protocol
- 13.1.6.6.2 Modbus RTU Protocol

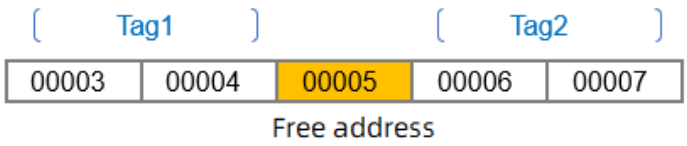
13.1.6.6.1 Modbus TCP Protocol

Modbus TCP protocol is an industry standard protocol widely used in the field of industrial automation. Modbus TCP protocol follows a unified communication specification, simplifying the configuration and maintenance processes when communicating with third-party devices or systems.

Modbus TCP protocol uses a Client-Server architecture, where the communication properties of the two parts differ. For details on respective settings, see the descriptions in the table.

Table13-1-6-6-1-1 Connection property description

Property Category		Description	Server	Client
Basic property	Connection name	Connection identifier supports combinations of Chinese characters, English letters, numbers, `_`, and `-`, with a maximum length of 64 English characters. Connection name must be unique within the same monitoring device. Default name automatically increments based on the number of existing connections (including 13.1.2.1 Connecting External Devices), ensuring no duplicates.	Applicable	Applicable
	Connection type	Ethernet	Applicable	Applicable
	Device name	Name of the connected object supports combinations of Chinese characters, English letters, numbers, `_`, and `-`, with a maximum length of 64 English characters. Device names do not support duplicates within the same monitoring device.	Applicable	Applicable
	Device description	Device description supports a maximum length of 128 characters.	Applicable	Applicable
	Device driver	The role of HMI device in communication. Modbus TCP Server indicates that the HMI device acts as the Server side. Modbus TCP Client indicates that the HMI device acts as the Client side.	Applicable	Applicable
Link parameters	IP address	The access address of the connected object.	Applicable	Applicable
	Port number	Setting range: 0 to 65535. Default: 502.	Applicable	Applicable
	Slave address	Unique number of a slave device. The master station communicates with a specific slave device via the station number. Station number range:1 to 255. Default: 1.	Not applicable	Applicable

Property Category		Description	Server	Client
Communication parameter	Address interval	This parameter refers to the number of idle addresses between adjacent data (addresses). When the number of idle addresses is less than or equal to the setting value, adjacent data will be merged into a single packet for transmission. The number of data packets affects the data transfer rate. More data packets result in a lower transfer rate. Setting range: 0 to 24, where 0 indicates each packet contains only one data. If set to 3, when reading two words of data Tag1 and Tag2 from address 0x00003 and 0x00006 respectively, with only one idle address 0x00005 in between, the packet merging condition is satisfied, meaning Tag1 and Tag2 can be sent in one combined packet. 	Not applicable	Applicable
	Max. read (bytes)	The maximum number of bytes per data packet. The setting range is determined by **device driver**.	Not applicable	Applicable
	Max. write (bytes)		Not applicable	Applicable
	Connection timeout	The client does not receive a response from the server within the specified time after sending a connection request during the establishment of a Modbus TCP/IP connection. Time unit: milliseconds (ms). Setting range: 1 to 3600000.	Not applicable	Applicable
	Communication timeout	The master station does not receive a response from the slave within the specified time after sending a request. Time unit: milliseconds (ms). Setting range: 1 to 3600000.	Not applicable	Applicable
	Retransmission count	Number of retries for resending commands when device response (e.g., PLC) data is abnormal. Up to 10 times are supported.	Not applicable	Applicable
	Connection attempt interval	Reconnection interval after communication is disconnected. Time unit: seconds (s). Setting range: 1 to 3600000.	Not applicable	Applicable
Activate	Enable/ Disable	Enable or disable the connection.	Applicable	Applicable

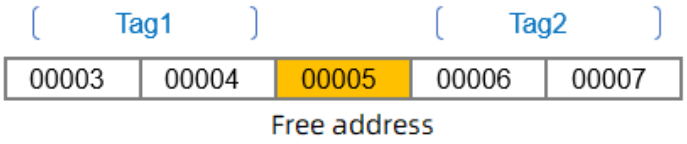
13.1.6.6.2 Modbus RTU Protocol

Modbus RTU protocol is an industry standard protocol widely used in industrial automation, especially in low-latency and high-precision applications. Modbus RTU protocol follows a unified communication specification, simplifying the configuration and maintenance processes when communicating with third-party devices or systems.

Modbus RTU protocol uses a Client-Server architecture, where the communication properties of the two parts differ. For details on respective settings, see the descriptions in the table.

Table13-1-6-6-2-1 Connection property description

Property Category		Description	Server	Client
Basic property	Connection name	Connection identifier supports combinations of Chinese characters, English letters, numbers, `_`, and `-`, with a maximum length of 64 English characters. Connection name must be unique within the same monitoring device. Default name automatically increments based on the number of existing connections (including 13.1.2.1 Connecting External Devices), ensuring no duplicates.	Applicable	Applicable
	Link type	Serial port	Applicable	Applicable
	Device name	Name of the connected object supports combinations of Chinese characters, English letters, numbers, `_`, and `-`, with a maximum length of 64 English characters. Device names do not support duplicates within the same monitoring device.	Applicable	Applicable
	Device description	Device description supports a maximum length of 128 characters.	Applicable	Applicable
	Device driver	The role of HMI device in communication. Modbus RTU Server indicates that the HMI device acts as the Server side. Modbus RTU Client indicates that the HMI device acts as the Client side.	Applicable	Applicable
Link parameters	Communication port	The number of ports depends on the specific product model. For example, the IT9156E touch screen supports COM1 to COM5 ports.	Applicable	Applicable
	Type	Select the type according to on-site requirements.	Applicable	Applicable
	Baud rate	Default: 9600 bps.	Applicable	Applicable
	Data bit	Supports 6, 7 and 8. Defaults to 8.	Applicable	Applicable

Property Category		Description	Server	Client
	Stop bit	Supports 1 and 2. Defaults to 1.	Applicable	Applicable
	Parity check	Supports no parity, odd parity, and even parity. The default is odd parity.	Applicable	Applicable
	Slave address	Unique number of a slave device. The master station communicates with a specific slave device via the station number. The setting range of station number is related with **device driver**. The default value is 1. For example, the station number range for the Modbus protocol is 1 to 255.	Applicable	Applicable
Communication parameter	Address interval	This parameter refers to the number of idle addresses between adjacent data (addresses). When the number of idle addresses is less than or equal to the setting value, adjacent data will be merged into a single packet for transmission. The number of data packets affects the data transfer rate. More data packets result in a lower transfer rate. Setting range: 0 to 24, where 0 indicates each packet contains only one data. If set to 3, when reading two words of data Tag1 and Tag2 from address 0x00003 and 0x00006 respectively, with only one idle address 0x00005 in between, the packet merging condition is satisfied, meaning Tag1 and Tag2 can be sent in one combined packet. 	Applicable	Applicable
	Max. read (bytes)	The maximum number of bytes per data packet. The setting range is determined by **device driver**.	Applicable	Applicable
	Max. write (bytes)		Applicable	Applicable
	Communication timeout	The master station does not receive a response from the slave within the specified time after sending a request. Time unit: milliseconds (ms). Setting range: 1 to 3600000.	Applicable	Applicable
	Retransmission count	Number of retries for resending commands when device response (e.g., PLC) data is abnormal. Up to 10 attempts are supported.	Applicable	Applicable
	Connection attempt interval	Reconnection interval after communication is disconnected. Time unit: seconds (s). Setting range: 1 to 36000.	Applicable	Applicable

Property Category		Description	Server	Client
	Device detection cycle	Device detection cycle after the device goes offline. Time unit: seconds (s). Setting range: 1 to 36000.	Applicable	Applicable
Activate	Enable/ Disable	Enable or disable the connection.	Applicable	Applicable

13.2 Using Variables

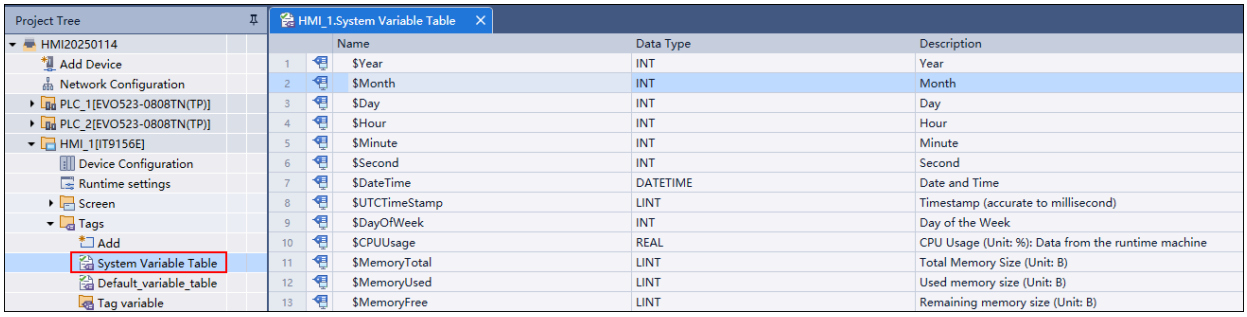
- 13.2.1 System Variable Table
- 13.2.2 Default Variable Table
- 13.2.3 Tag Variables
- 13.2.4 Using a Variable Table
- 13.2.5 Configuring Variables
- 13.2.6 Operating Variables

13.2.1 System Variable Table

The system variable table displays variables related to system operation, which can show real-time data of the current system while running on the screen, making it easier for you to monitor the system status.

The system variable table does not support editing or deletion. The variables in the table do not support writing values or recording historical data during runtime.

When adding a PC/HMI device to a project, a "System variable table" node is automatically created under the device, which can be opened by double-clicking.



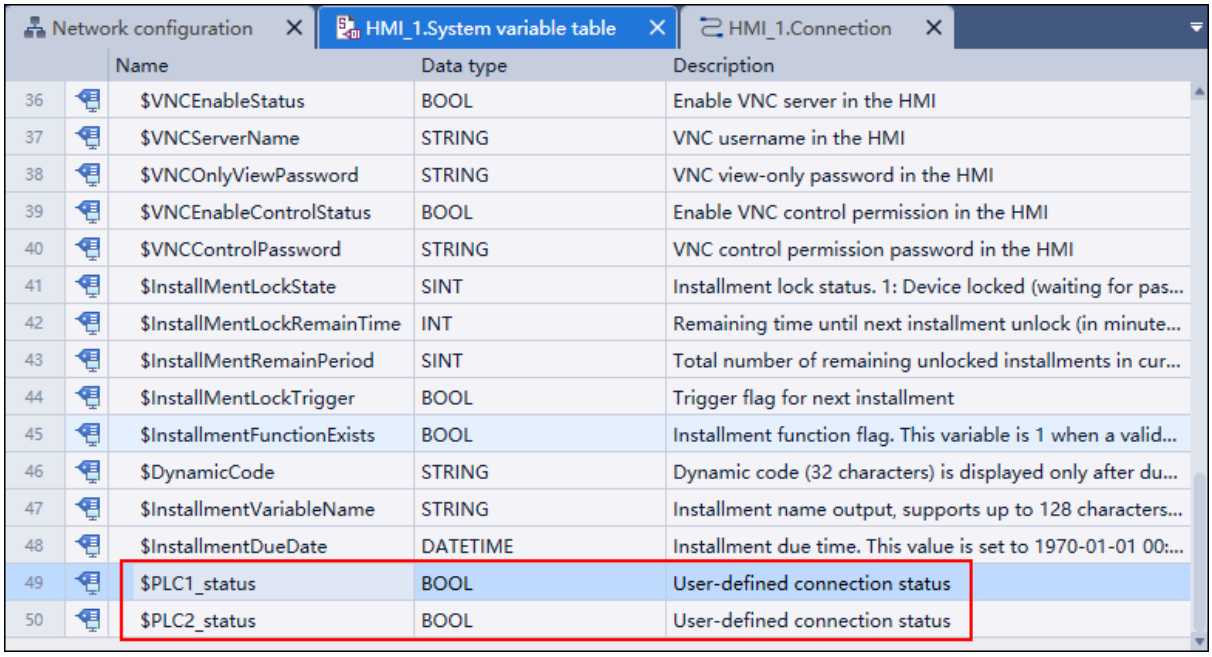
System variables consist of two parts:

- Variables related to the hardware configuration and operation of monitoring device
  These variables are automatically generated when adding a device, typically distributed in the upper half of the variable table (e.g., rows 1 to 40), with the exact number varying depending on the hardware model. The IT9156E is used to demonstrate system variable distribution:
  Rows 1 to 9: variables related to system time;
  Rows 10 to 23: variables related to CPU operation;
  Rows 24 to 31: variables related to physical network ports;
  Rows 32 to 35: variables related to display and user permissions;
  Rows 36 to 40: variables related to VNC access permissions

No.	Category	Name	Data Type	Description		Applicable Device
1	System time	$Year	INT	Year		PC/HMI
2		$Month	INT	Month		PC/HMI
3		$Day	INT	Day		PC/HMI
4		$Hour	INT	Hour		PC/HMI
5		$Minute	INT	Minute		PC/HMI
6		$Second	INT	Second		PC/HMI
7		$DateTime	DATETIME	Date and time		PC/HMI
8		$UTCTimeStamp	LINT	Timestamp (millisecond)		PC/HMI
9		$DayOfWeek	INT	Week		PC/HMI
10	CPU running	$CPUUsage	REAL	CPU usage (unit: %)		PC/HMI
11		$MemoryTotal	LINT	Total memory size. Unit: byte		PC/HMI
12		$MemoryUsed	LINT	Used memory size (unit: byte)		PC/HMI
13		$MemoryFree	LINT	Free memory size. Unit: byte		PC/HMI
14		$MemoryUsage	REAL	Memory usage (unit: %)		PC/HMI
15		$LocalMachineName	STRING	Runtime machine name		PC/HMI
16		$OSVersion	STRING	Runtime machine operating system version		PC/HMI
17		$CPUTemperatureHigh	BOOL	CPU temperature exceeds 83°C		HMI
18		$CPUTemperatureTooHigh	BOOL	CPU temperature exceeds 115°C		HMI
19		$DiskTotal	LINT	Total disk size. Unit: byte	Disk of the touch screen refers to the inodata directory. Disk of a PC device refers to the disk where the runtime data storage path is located, such as Disk D.	PC/HMI
20		$DiskUsed	LINT	Used disk size. Unit: byte		PC/HMI
21		$DiskFree	LINT	Free disk size. Unit: byte		PC/HMI
22		$DiskUsage	REAL	Disk usage (unit: %)		PC/HMI
23		$RTCVoltage	INT	RTC battery voltage (unit: mV)		HMI
24	Physical network port	$IPAddress_1	STRING	IP address of physical network card 1		PC/HMI
25		$Netmask_1	STRING	Subnet mask of physical network card 1		PC/HMI

No.	Category	Name	Data Type	Description	Applicable Device
26		$Gateway_1	STRING	Gateway of physical network card 1	PC/HMI
27		$MACAddress_1	STRING	MAC address of physical network card 1	PC/HMI
28		$IPAddress_2	STRING	IP address of physical network card 2	PC/HMI
29		$Netmask_2	STRING	Subnet mask of physical network card 2	PC/HMI
30		$Gateway_2	STRING	Gateway of physical network card 2	PC/HMI
31		$MACAddress_2	STRING	MAC address of physical network card 2	PC/HMI
32	Other	$CurrentScreenId	LINT	Current screen ID	PC/HMI
33		$CurrentUserName	STRING	Current user name	PC/HMI
34		$CurrentRoleName	STRING	Current role name	PC/HMI
35		$CurrentLanguage	STRING	Current runtime display language	HMI
36	VNC	$VNCEnableStatus	BOOL	Enable VNC server in the HMI	HMI
37		$VNCServerName	STRING	VNC username for the HMI	HMI
38		VNCOnlyViewPassword	STRING	VNC monitoring password for the HMI	HMI
39		VNCEnableControlStatus	BOOL	Enable the VNC operation permission for the HMI	HMI
40		VNCControlPassword	STRING	VNC operation password for the HMI	HMI

- Connection variables
  These variables are generated when you create a connection. They are appended to the end of the table by default (e.g., after row 35). Connection variables are named in the format $ConnectionName_Status and are used to indicate the connection status.



	Name	Data type	Description
36	$VNCEnableStatus	BOOL	Enable VNC server in the HMI
37	$VNCServerName	STRING	VNC username in the HMI
38	$VNCOnlyViewPassword	STRING	VNC view-only password in the HMI
39	$VNCEnableControlStatus	BOOL	Enable VNC control permission in the HMI
40	$VNCControlPassword	STRING	VNC control permission password in the HMI
41	$InstallMentLockState	SINT	Installment lock status. 1: Device locked (waiting for pas...
42	$InstallMentLockRemainTime	INT	Remaining time until next installment unlock (in minute...
43	$InstallMentRemainPeriod	SINT	Total number of remaining unlocked installments in cur...
44	$InstallMentLockTrigger	BOOL	Trigger flag for next installment
45	$InstallmentFunctionExists	BOOL	Installment function flag. This variable is 1 when a valid...
46	$DynamicCode	STRING	Dynamic code (32 characters) is displayed only after du...
47	$InstallmentVariableName	STRING	Installment name output, supports up to 128 characters...
48	$InstallmentDueDate	DATETIME	Installment due time. This value is set to 1970-01-01 00:...
49	$PLC1_status	BOOL	User-defined connection status
50	$PLC2_status	BOOL	User-defined connection status

13.2.2 Default Variable Table

Each PC/HMI device in the project has a default variable table with a node ID , which does not support deletion or moving to a variable group.

The node name can be renamed. It consists of Chinese characters, English letters, digits, "_", and "-", with a maximum length of 128 characters.

The default variable table supports copy/paste operations. For details, see 13.2.4.4 Copying and Pasting a Variable Table. Variables support import/export operations. For details, see 13.2.6.2 Importing and Exporting Variables.

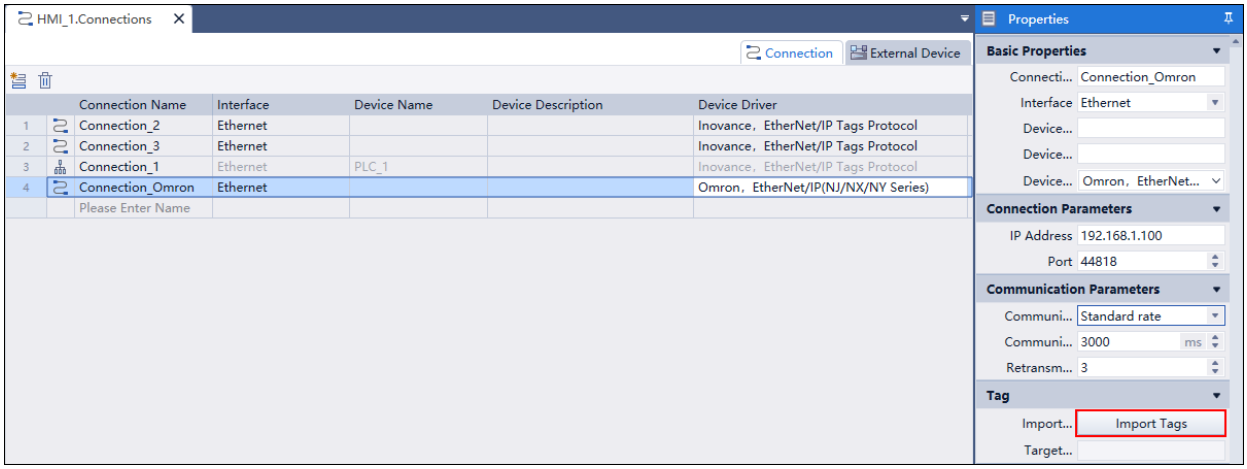
13.2.3 Tag Variables

When monitoring devices communicate with other devices via the EtherNet/IP protocol, data transmission is achieved through tag variables.

Tag variables are imported into the iFA software in XML or TXT format. You can not edit the variable information.

Importing tag variables

Tag variables can be imported into the iFA software via the connection property panel (see figure below). For details on establishing a connection, see 13.1.1.1 Establishing a Device Connection.



A single connection supports a maximum of 500,000 imported tags. If this limit is exceeded, the entire import will fail.

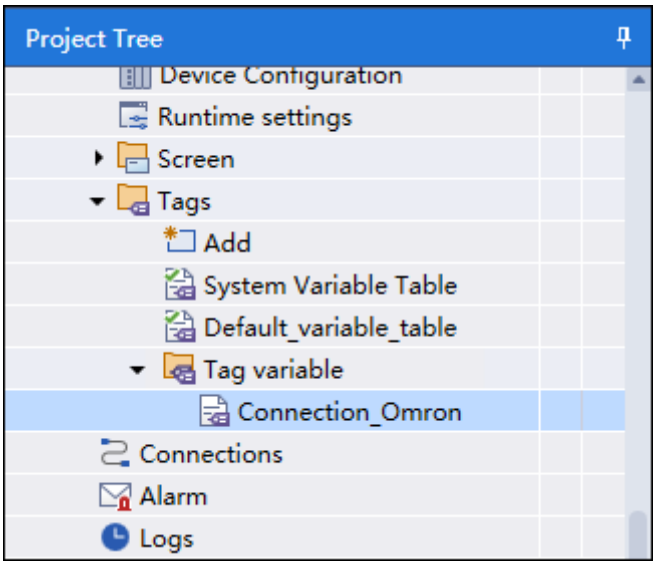
Variable count calculation: simple variables are counted as 1 each; complex variables (such as arrays or user-defined types) are counted based on their minimum number of sub-elements. For example, an array with 5 sub-elements is counted as 5.

During the import of time and text variables into the iFA software, their data types are converted to match existing data types. The following table lists the data type mappings for the three protocols that require conversion.

Data Type Before Import	Data Type After Import via AC800 Tags Protocol	Data Type After Import via EtherNet/IP Tags Protocol	Data Type After Import via Omron EtherNet/IP Protocol
TIME	UDINT	32-bit. Range: 0 **ms** to 71582 **m** 47 **s** 295 **ms**	LINT
LTIME	ULINT	64-bit. Range: 0 **ms** to 213503 **d** 23 **h** 34 **m** 33 **s** 709 **ms** 551 **µs** 615 **ns**	N/A
TIME_OF_DAY(TOD)	UDINT	32-bit. Range: 00:00:00.000 to 23:59:59.999	UDINT
DATE	UDINT	32-bit. Range: 1970-01-01 to 2106-02-07	UDINT
DATE_AND_TIME(DT)	UDINT	32-bit. Range: 1970-01-01 00:00:00 to 2106-02-07 06:28:15	ULINT
STRING	Encoding format: ASCII (Chinese characters not supported). Default length: 80.	Encoding format: UTF-8. Default length: 80.	Encoding format: UTF-8. Default length: 80. Max. length:1986.
WSTRING	Unchanged	Encoding format: UTF-16. Default length: 80.	Not involved

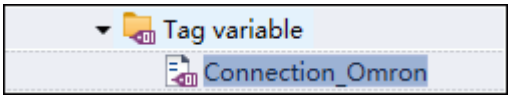
Data Type Before Import	Data Type After Import via AC800 Tags Protocol	Data Type After Import via EtherNet/IP Tags Protocol	Data Type After Import via Omron EtherNet/IP Protocol
ARRAY	Unchanged	Supports one-dimensional and multi-dimensional arrays as well as nested arrays, with a maximum of 65,536 array elements.	Supports three-dimensional arrays but does not support nested arrays, with a maximum of 65,536 array elements.

Upon successful tag import, corresponding child nodes (variable tables) are generated under the **tag variable** node, and are named after the **connection name** by default.

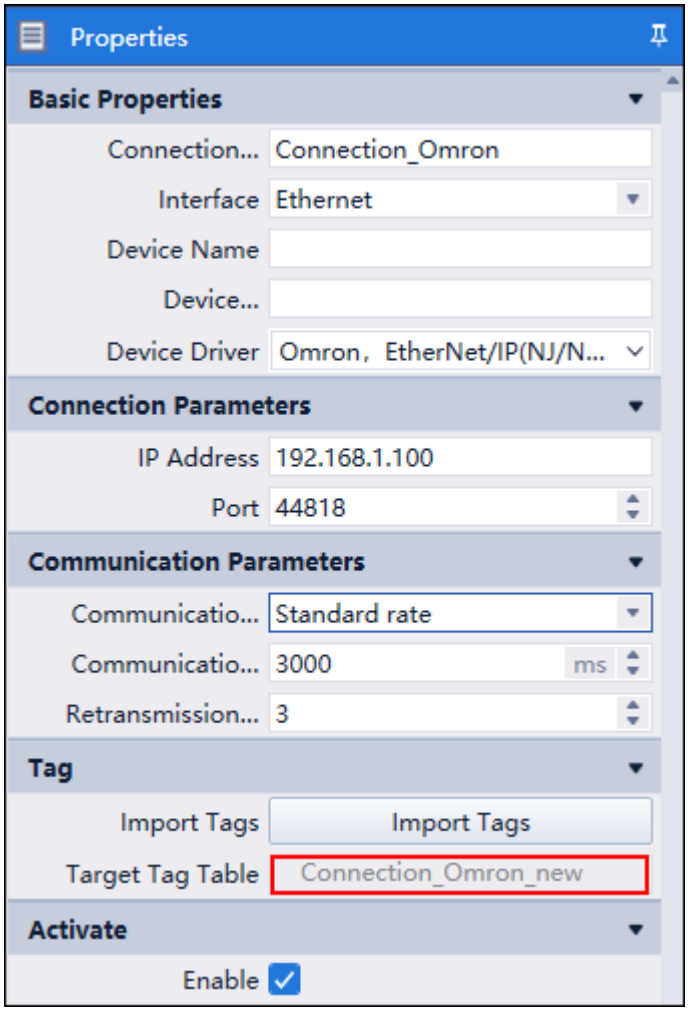


Operations related to tag variable table

- Rename a variable table
  Select the tag variable table node, and then select "Rename" from the right-click menu to put the name into edit mode.

  Enter the new name and press **Enter** to confirm. The new name will be automatically synchronized and updated in the connection property panel.



- Update tags
  Tag information in the tag variable table does not support editing.
  If a tag file with the same name is imported multiple times into the same connection, the tag content from the latest import takes precedence.
  The validity of tags referenced on the screen depends on the tag name. If the tag name remains unchanged, the reference remains valid; otherwise, it becomes invalid.
- Reference screens
  iFA software supports referencing tag variables through elements or graphic symbols on the screen.
- Delete a variable table
  Select the tag variable table node, and then choose "Delete" from the right-click menu to remove the variable table.
  After a tag is deleted, its references on the screen still exist and must be removed manually.

13.2.4 Using a Variable Table

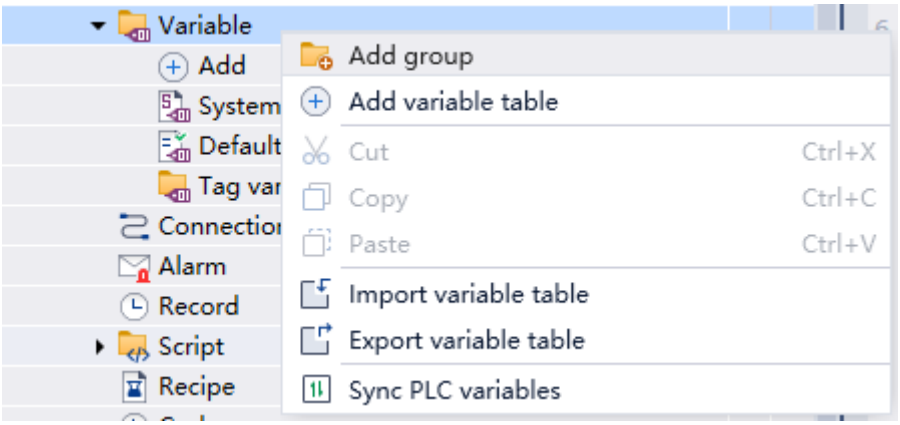
- 13.2.4.1 Variable Table Groups
- 13.2.4.2 Adding a Variable Table
- 13.2.4.3 Importing and Exporting a Variable Table
- 13.2.4.4 Copying and Pasting a Variable Table
- 13.2.4.5 Deleting a Variable Table

13.2.4.1 Variable Table Groups

Regular variable tables are grouped and managed based on the properties of the variables.

A first-level group can be created under the "Variable" node, which supports expanding multiple levels of subgroups.

In the right-click menu of the "Variable" node, select "Add group" to add a variable group, with the node identifier .



Variable group names support combinations of Chinese, English, numbers, "_", and "-", with a maximum length of 64 characters. The default name is "Variable Group_n", with the sequence number incrementing automatically.

Variable group name must be unique within the same device.

13.2.4.2 Adding a Variable Table

The variable tables are used to store user-defined variables, and are divided into default variable table and regular variable table. The default variable table is automatically generated when adding a monitoring device, while the regular variable table is created by yourself.

Regular variable tables can be added under both the "Variable" node and the "Variable group" node. For details on creating variable groups, see 13.2.4.1 Variable Table Groups.

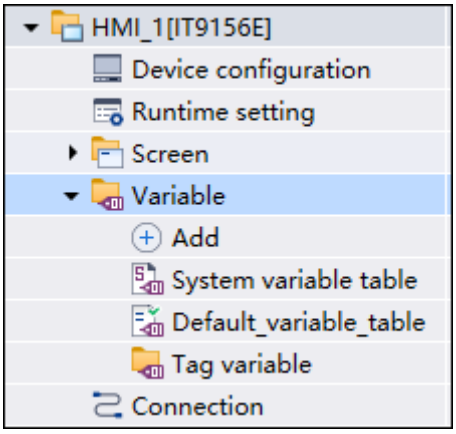


Figure 13-2-4-2-1 Add location - variable node

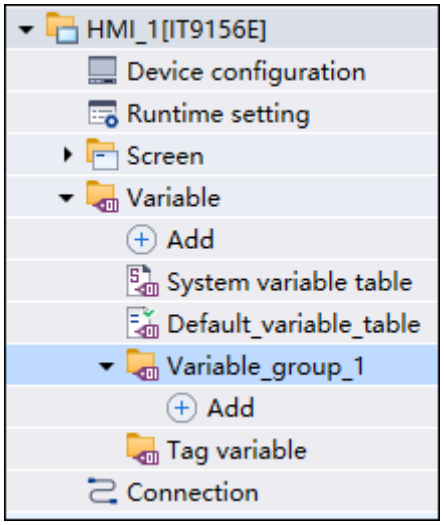


Figure 13-2-4-2-2 Add location - variable group

Add in the following three methods:

- Double-click the "Add" node.
- In the right-click menu of the parent node, select the "Add variable table" command.
- If the name of the imported variable table is different from existing variable tables in the project tree, corresponding node will be created automatically during import.

The name of variable table node identifier supports combinations of Chinese, English, numbers, "_", and "-", with a maximum length of 64 characters. The default name is "Variable Table_n", where n starts at 1 and increments automatically.

Variable table name must be unique within the same device.

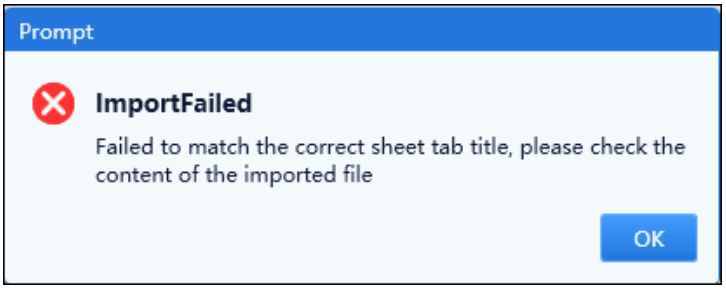
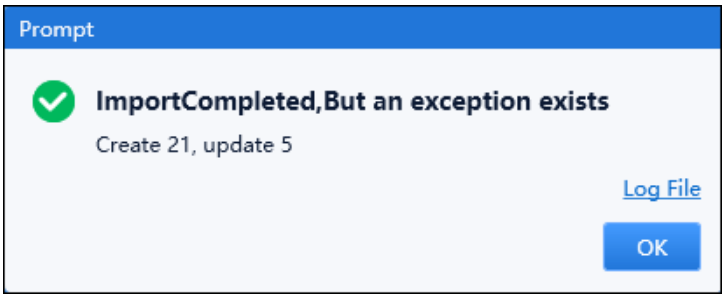
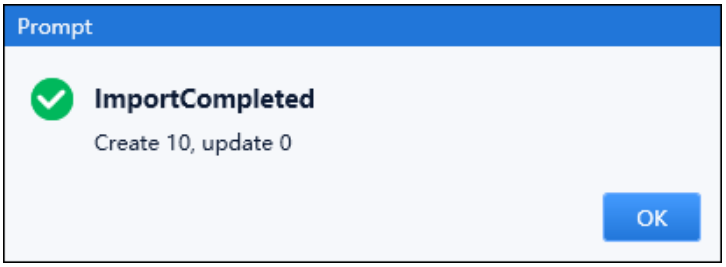
13.2.4.3 Importing and Exporting a Variable Table

The import and export function of the variable table enables cross-device reuse of variable tables and reduces configuration time.

Importing a variable table

This section details the specifications and import procedure for variable table file. For import rules, see Importing variables.

<table>
<tr><th>Step</th><th>Operation</th><th>Description</th></tr>
<tr><td>1</td><td>Prepare variable table</td><td>Basic requirements for the variable table:
(1) File format must be xlsx.
(2) Sheet name length must be less than 64 characters and consist of Chinese, English, numbers, "_", and "-".
(3) The table must contain only one sheet page named "Variable".
(4) The content and order of the first row cannot be modified. The following table shows the property names for each column.
<table>
<tr><td>A</td><td>Name</td><td>G</td><td>Access mode</td><td>M</td><td>Range upper limit</td><td>S</td><td>PLC end value</td></tr>
<tr><td>B</td><td>Data type</td><td>H</td><td>Acquisition cycle</td><td>N</td><td>Range lower limit</td><td>T</td><td>HMI start value</td></tr>
<tr><td>C</td><td>Connection</td><td>I</td><td>Description</td><td>O</td><td>Constant range upper limit</td><td>U</td><td>HMI end value</td></tr>
<tr><td>D</td><td>Device name</td><td>J</td><td>Length</td><td>P</td><td>Constant range lower limit</td><td>V</td><td>Initial value</td></tr>
<tr><td>E</td><td>Referenced variable</td><td>K</td><td>Alarm group</td><td>Q</td><td>Enable linear conversion</td><td>W</td><td>Internal variable retention</td></tr>
<tr><td>F</td><td>Address</td><td>L</td><td>High-low byte swap</td><td>R</td><td>PLC start value</td><td></td><td></td></tr>
</table>
(5) The "name" column must contain variable names, with at least one entry.
(6) The content in the "Data type" column must match the format used in the software, for example, Bool is valid, but bool is invalid.
(7) For array-type variables, only the type name needs to be configured, such as <code>ARRAY[2..5] OF INT</code>. Individual data elements do not need to be listed.
If any of the first five conditions are not met, the table import will fail.</td></tr>
<tr><td>2</td><td>Select import command</td><td>Select the command based on the import location.
Select "Import variable table" in the right-click menu of the parent node or select "Import" in the right-click menu of the same-level variable table node.</td></tr>
<tr><td>3</td><td>Select import target</td><td>Select the target variable table.</td></tr>
</table>

Step	Operation	Description
4	Confirm import result	The result of importing variables will be displayed in a pop-up window. • Import failed Caused by the sheet tab title not being "Variable".  • Import completed with errors Variable information in the table is abnormal (e.g., incomplete data).; Details are recorded in the log file.  • Import completed Variable information in the table is valid. All is imported. 
5	Import ended	If there are errors in the import result, handle according to the log content. When the import target conflicts with the name of an existing variable table, only the variable will be imported; when there is no conflict, the corresponding node will be created and the variable will be imported. For variable import rules, see 13.2.6.2 Importing and Exporting Variables.

Exporting a variable table

Variable tables support single and batch export, with export format as xlsx.

Batch selection of variable tables and variable groups supports Ctrl or Shift keys. Variable table export is performed via the corresponding command in the right-click menu. If a table with the same name exists in the storage path, it will be overwritten.

13.2.4.4 Copying and Pasting a Variable Table

Copy

Select the object to be copied in the project tree, and trigger copying via the right-click menu command or Ctrl+C.

Paste

Pasting can be performed by selecting the "Variable" node or "Variable group" node.

When pasting a variable table, "_n" is appended to the name by default, with n starting at 1 and incrementing automatically.

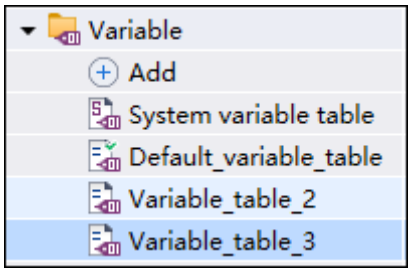


Figure 13-2-4-4-1 Paste result

13.2.4.5 Deleting a Variable Table

Regular variable tables support deletion. When a variable table is deleted, all variables within it will also be deleted.

Variable tables support single or batch deletion. Select multiple target variable tables using the **Ctrl** or **Shift** key, and then delete via the right-click menu command or the **Delete** key.

13.2.5 Configuring Variables

- 13.2.5.1 Adding an Internal Variable
- 13.2.5.2 Adding an External Variable
- 13.2.5.3 Setting Variable Properties
- 13.2.5.4 Setting Properties in Batch
- 13.2.5.5 Multiplex Variables
- 13.2.5.2.3 Synchronizing Variable Names
- 13.2.5.6 Storing Variable Data
- 13.2.5.7 Configuring Variable Events

13.2.5.1 Adding an Internal Variable

Internal variables are stored within the HMI device and have no connection to the PLC. Internal variables are used to monitor internal system calculations.

Internal and external variables are distinguished by **connection** property. For example, in the figure below, Var1 is an internal variable and Var2 is an external variable.

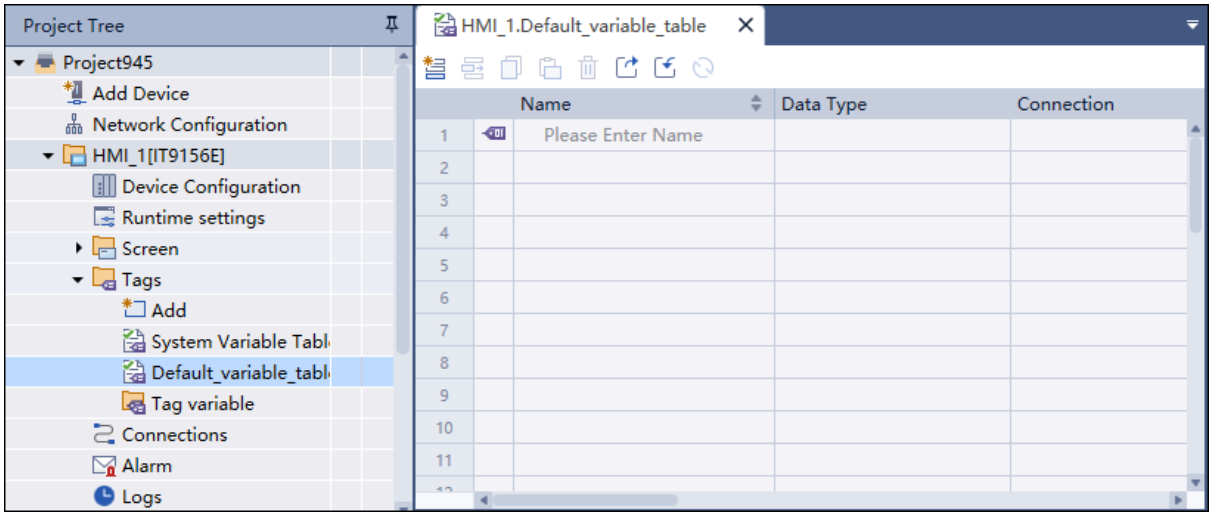


Both the 13.2.2 Default Variable Table and the regular variable table support adding internal variables. For details on adding a regular table, see 13.2.4.2 Adding a Variable Table.

This section takes the default variable table as an example to illustrate the method for adding internal variables.

1. Open the variable table editing window.

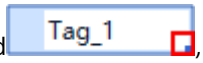
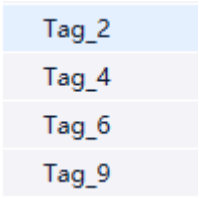
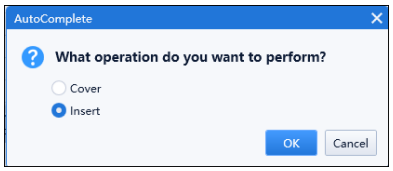
   Double-click the "Default variable table" node in the **project tree**.



2. Add variables.

   Variables are managed in a list format, with each row representing one variable, identified by .

Number	Procedure	Example
One	Manual input: Double-click the text box Please Enter Name, enter a valid variable name, and press Enter to add the variable. The software will automatically fill in the **data type** with the default value.	Var_example INT
One	Command input: Select or to add a variable. The variable name and data type are filled with default values. The name is named as Tag_n, where n starts at 1 and increments automatically.	Tag_1 INT

Number	Procedure	Example
Multiple	Click on the name of an existing variable, drag the square at the lower-right corner of the cell downward , and select "Insert new variable" to add a new variable. The new variable will copy all properties from the target variable (e.g., Tag_1), including alarm, recording, and event configurations. The software automatically checks for duplicate variable names. Non-consecutive variable numbers indicate missing entries exist in other variable tables of the current device.  Add or insert new variables in the drop-down area. Existing variables will shift down automatically.	
Multiple	Variables can also be added by batch import. For details, see Importing variables.	-

Once an internal variable is added, it can be applied in the project configuration.

13.2.5.2 Adding an External Variable

- 13.2.5.2.1 Adding an External Variable
- 13.2.5.2.2 Naming an External Variable
- 13.2.5.2.3 Synchronizing Variable Names

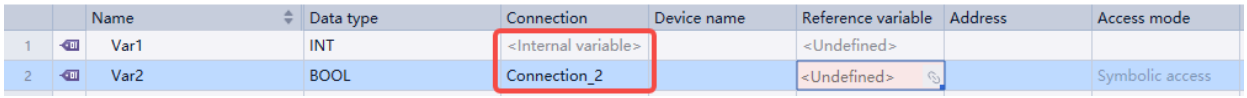
13.2.5.2.1 Adding an External Variable

Introduction

External variables can access PLC addresses through PLC variables. Addressing is performed in the following ways:

- Symbolic addressing
- Absolute addressing

Internal and external variables are distinguished by **connection** property. For example, in the figure below, Var1 is an internal variable and Var2 is an external variable.



Both the 13.2.2 Default Variable Table and the regular variable table support adding external variables. For details on adding a regular table, see 13.2.4.2 Adding a Variable Table.

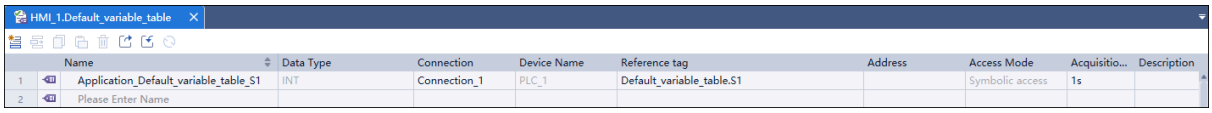
Requirements

- The HMI and PLC belong to the same project.
- The HMI has been added.
- The PLC has been added and variables have been made public.

Add by drag-and-drop

This section takes the default variable table as an example to illustrate the method for adding external variables.

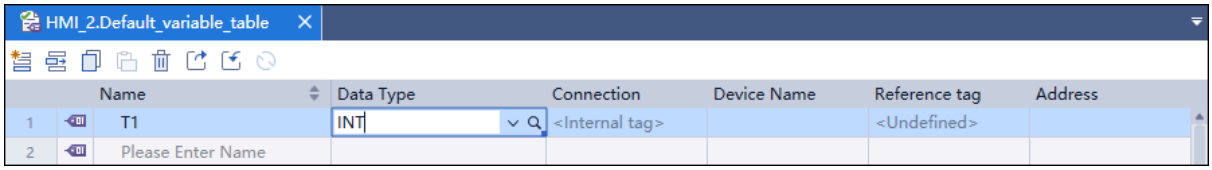
1. Open the default variable table on the HMI side
2. In the project tree, select the node containing the PLC public variables. The corresponding variable list will be displayed in the "Details" area. For example, PLC variable S1 is of INT type and public read-only.
3. Select the desired PLC public variables and drag them to a blank row in the default variable table on the HMI side.
   The system will automatically generate corresponding external variables. The default variable name is generated based on the connection configuration. For details, see 13.2.5.2.2 Naming an External Variable. For example, the name **Application_Default Variable Table_S1** may be automatically generated.



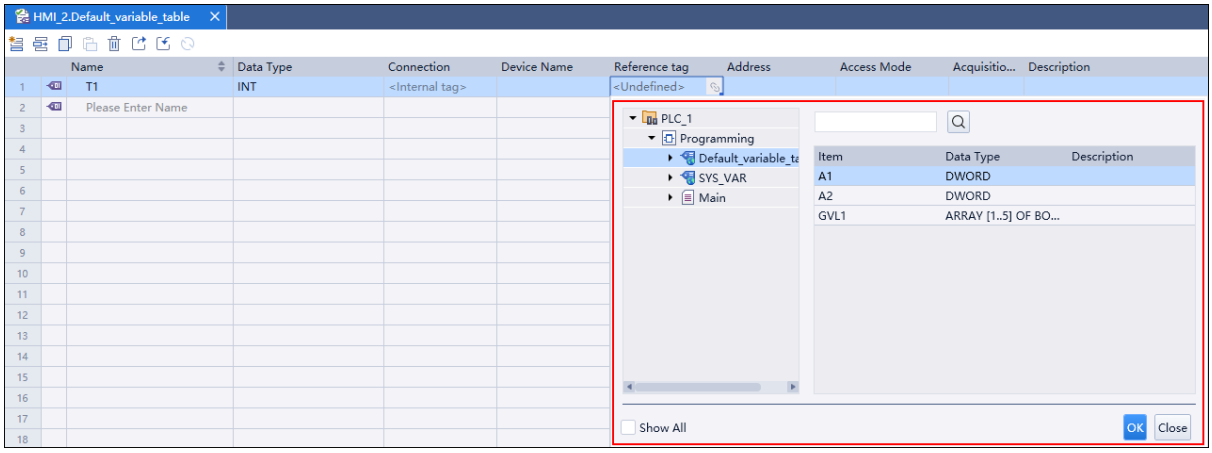
Add by reference

This section takes the default variable table as an example to illustrate the method for adding external variables.

1. 13.2.5.1 Add an internal variable
   For example, add an INT variable T1.

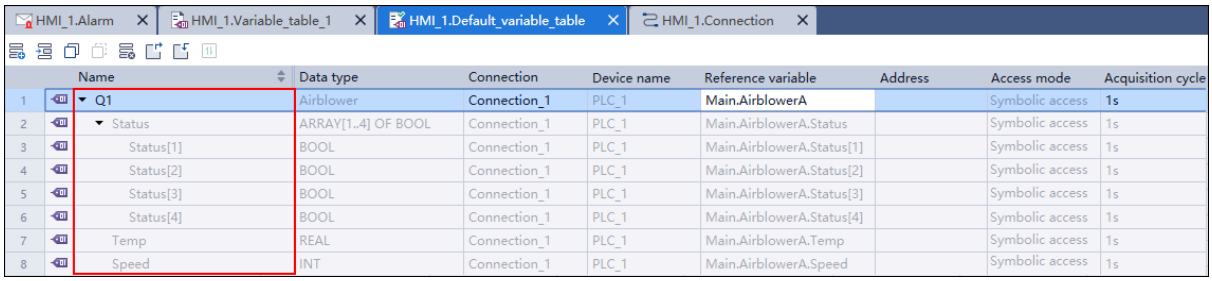


2. Select a reference variable
   Select the reference variable cell, and click to open the variable selection window.



3. Select the target variable and confirm
   The data type of the internal variable will automatically adjust according to the type of the referenced variable to ensure consistency in data types. For example, in step 2, if T1 references a variable A1 of type

DWORD, the data type of T1 will automatically be set to UDINT.
For array types (including arrays of basic types, arrays of structures, and nested arrays within structures), the total number of referenced elements must not exceed 2,000.
For structure types, the total number of referenced members must not exceed 20,000.
For example, if the internal variable Q1 references an array of structures named AirblowerA, the number of sub-members is 6.

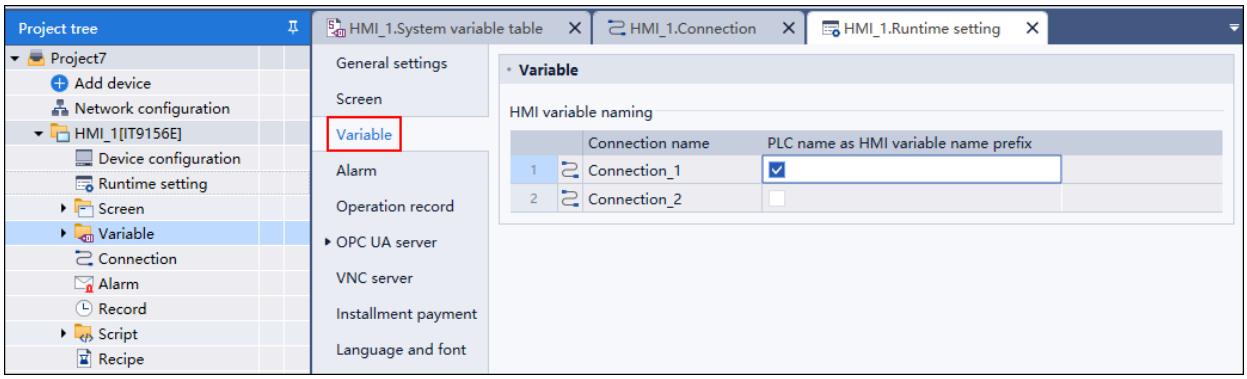


4. Click the synchronization button to make the HMI-side variable name consistent with the referenced variable name. For details, see 13.2.5.2.3 Synchronizing Variable Names.

An external variable has been added and mapped to a variable in the PLC.

13.2.5.2.2 Naming an External Variable

External variable names in the HMI can be configured. The configuration is done on the "Runtime setting" interface of the device.



For HMI and PLC in the same project, the HMI external variable name depends on the "PLC name as HMI variable name prefix" setting and how it is added.

Add by direct drag-and-drop

When you drag variables from PLC to the HMI variable table or screen, the HMI variable naming is determined by the "PLC name as HMI variable name prefix" setting.

- Checked: Variable name prefix will be added

Prefix	Naming Format	Applicable Device Driver
PLC device name	**PLC device name**_Application_PLC variable table name_PLC variable name For example, PLC1_Application_GVL_Relay connect	• Inovance Ethernet/IP Tags Protocol • Inovance AC800 Tags Protocol (AC/AI/AM/AP Series)

Prefix	Naming Format	Applicable Device Driver
PLC device name	**PLC device name**_PLC variable name For example, PLC1_Relay connect	• Omron EtherNet/IP (NJ/NX/NY Series) • Rockwell (AB) EtherNet/IP • OPC UA, OPC UA Client

- Unchecked: No variable name prefix will be added

Add by manual association

When you manually associate HMI internal variables with PLC variables, the HMI variable naming is not affected by the"PLC name as HMI variable name prefix" setting.

You can use the "13.2.5.2.3 Synchronizing Variable Names" command to automatically add a prefix to match the PLC variable name.

13.2.5.2.3 Synchronizing Variable Names

When the PLC and HMI are in the same project, the "Sync PLC variables" function ensures HMI variable names match PLC variable names.

Requirements

- The PLC variable name and HMI variable name for an external variable are inconsistent.
- The "Network public" property of the PLC variable is not "Private".
- Synchronization with PLC tag variable names is not supported.

Synchronization result

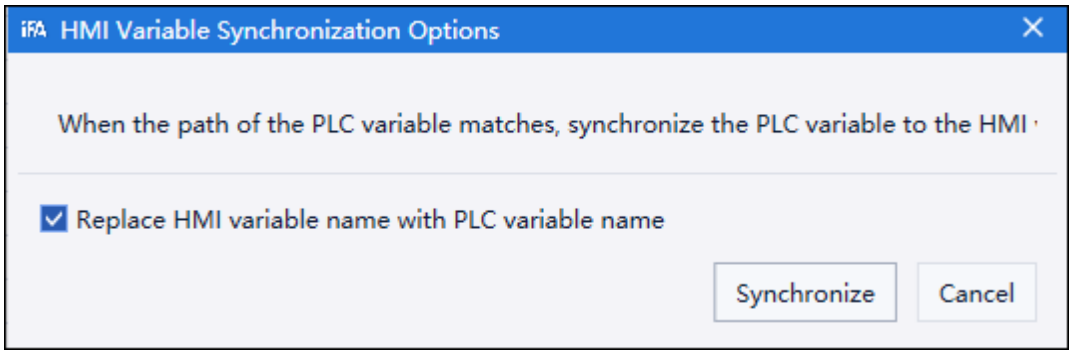
- Re-establish the reference relationship from HMI variables to PLC variables.
- Rename the HMI variables using the PLC variable names, with the naming format determined by the device configuration. For details, see 13.2.5.2.2 Naming an External Variable.
- Synchronize the number of structure variable members.

Synchronization operation for a single variable table

1. Select the external variables to be synchronized from the variable table.

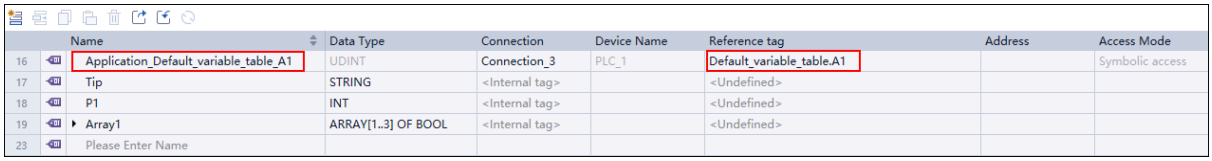
	Name	Data Type	Connection	Device Name	Reference tag	Address	Access Mode
16	T1	UDINT	Connection_3	PLC_1	Default_variable_table.A1		Symbolic access
17	Tip	STRING	<Internal tag>		<Undefined>		
18	P1	INT	<Internal tag>		<Undefined>		
19	Array1	ARRAY[1..3] OF BOOL	<Internal tag>		<Undefined>		
23	Please Enter Name						

2. Click the button to set the synchronization scope.
   Click the "Synchronize" button to open the synchronization option settings window.



If synchronization options are unchecked, only variable references are re-established. If checked, variable names are also updated in addition to reference setup.

3. Complete the synchronization
   For example, the variable name replacement result is shown in the figure below.

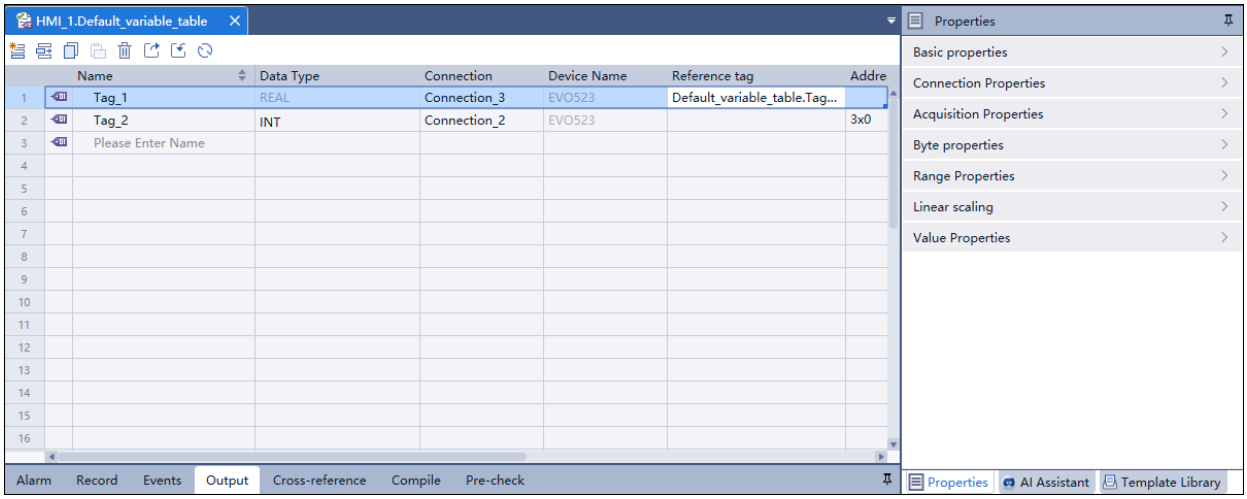


Synchronization operation for the entire device

1. Select the variable node of the HMI device.
2. In the quick menu, select "Sync PLC variables".
3. Synchronize the corresponding content based on the option settings. For detailed description, see step 2 in Synchronization operation for a single variable table.
4. Check the synchronization result.

13.2.5.3 Setting Variable Properties

All properties of variables added in the default variable table and the regular variable table are displayed in the right-side panel. You can modify variable information in the table or in the right-side property panel.



Internal and external variables have different property items. The following is a comprehensive description of their property items. You can select the corresponding parameter description based on the selected variable type.

Basic properties

- Name
  It is the unique identifier of variables in the software, and the naming rules are:
  (1) Consist of Chinese characters, letters, numbers, spaces, and special characters
  (2) Spaces cannot be placed at the beginning or end
  (3) Support a maximum length of 128 characters
  (4) Names are case-insensitive
  (5) The default is "Tag_n", where n starts at 1 and increments automatically.

- Description
  The description of variables. Up to 128 characters is supported.
- Data type
  Data type supports basic types and complex types. If the data types of internal variables and referenced variables are inconsistent, the internal variable will automatically adjust to ensure consistency.
  - Basic types

Category	Data Type	Value Range	Length	Memory Usage
Boolean	BOOL	TRUE, FALSE	1-bit	1 bytes
Integer	SINT	-128 to 127	8-bit	1 byte
	USINT	0 to 255	8-bit	1 byte
	INT	-32768 to 32767	16-bit	2 bytes
	UINT	0 to 65535	16-bit	2 bytes
	DINT	-2147483648 to 2147483647	32-bit	4 bytes
	UDINT	0 to 4294967295	32-bit	4 bytes
	LINT	-9223372036854775808 to 9223372036854775807	64-bit	8 bytes
	ULINT	0 to 18446744073709551615	64-bit	8 bytes
Floating-point number	REAL	-3.402823e+38 to -1.175495e-38 ±0.0 +1.175495e-38 to +3.402823e+38 +∞/-∞	32-bit	4 bytes
	LREAL	-1.7976931348623157e+308 to -2.2250738585072014e-308 ±0.0 +2.2250738585072014e-308 to +1.7976931348623157e+308 +∞/-∞	64-bit	8 bytes

Category	Data Type	Value Range	Length	Memory Usage
String	STRING	Encoding format is UTF-8. Supports 0 to 255 bytes. Any excess will be truncated.	/	Each character occupies 1 to 4 bytes. English letters and numbers generally occupy 1 byte.
	WSTRING	Encoding format is UNICODE. Supports 0 to 255 characters. Any excess will be truncated.	/	Each character occupies 2 bytes.
Time	DATETIME	1970-01-01-00:00:00.000 to 9999-12-31-23:59:59.999 Default format: yyyy-mm-dd-hh:mm:ss	4 bytes	4 bytes
	TIME	0ms to 71582m47s295ms	32-bit	4 bytes
	LTIME	0ms to 213503d23h34m33s709ms551μs615ns	64-bit	8 bytes
	TIME_OF_DAY(TOD)	00:00:00.000 to 23:59:59.999	32-bit	4 bytes
	DATE	1970-01-01 to 2106-02-07	32-bit	4 bytes
	DATE_AND_TIME(DT)	1970-01-01 00:00:00 to 2106-02-07 06:28:15	32-bit	4 bytes

NOTE

HMI devices cannot directly create internal variables of the time types: TIME, LTIME, TIME_OF_DAY (TODAY), DATE, and DATE_AND_TIME (DT). However, these types can be added successfully when referencing public variables from EVO series PLCs.

◦ Complex types

Category	Restrictions
Array	Both internal and external variables can be defined as arrays. When configuring external variables as an array, the following conditions must be met. ▪ Total number of elements does not exceed 20,000. ▪ Basic type elements are supported (see the table above). ▪ Nested arrays are not supported. ▪ Communication driver is the EtherNet/IP protocol.
Alias	The data type of external variables will be displayed as the original type of the alias. Alias variables for HMI communication are only supported for PLCs that are configured in the iFA software.
Enumeration	The data type of external variables will be displayed as integer types defined by enumeration, such as INT, LINT, and SINT. Enumeration variables for HMI communication are only supported for PLCs that are configured in the iFA software.
Structure	Either internal or external variables can be defined as structure types. Internal variables: Created by instantiating HMI User-Defined Types (located in the template library). External variables: Reference the PLC's structure variables. The communication driver must be the EtherNet/IP protocol.

- Alarm group
  Select the target alarm group. For details on adding alarm groups, see 13.5.5 Alarm Group Configuration.
- Length
  The length of string type can be configured, with a range of 1 to 255 characters, defaulting to 80.

Connection property

- Reference variable
  The associated variable must meet the following conditions to establish a mapping relationship between the current variable and the PLC variable.
  ◦ When the PLC where the variable resides and the monitoring device belong to the same project, the public property of the variable must be set to a value other than "Private".
  ◦ Tag variables import are supported.
  ◦ Only a single element of an array can be associated.

  When the PLC is not connected to the monitoring device, the software automatically creates a connection between them by associating a variable of the PLC. For more information, see Automatically establishing a connection via variable table.
- Connection
  The connection name is automatically filled based on the referenced variable.

- Device name
  The device name is automatically filled based on the referenced variable.
- Address
  The device driver type of the connection determines the address settings. Address configuration is supported when the access mode is set to absolute access.
  The address format generated by different **device drivers** (communication protocols) varies:

Communication Object	Address Format	Example
Mitsubishi FX3U/FX3UC/FX5U/Q series	Area address	SM 0
Siemens S7-1200/S7-1500 series	%Area address %Area address.Bit	%MW0 %QX0.5
Inovance AM/AC/MD/EVO series		
Inovance H5U/Easy series	Area address	R17

  When the connected device driver is changed, the address automatically reverts to its initial value. For example, **%QX3.0** in the figure automatically adapts to **%QX0.0**.

	Name	Data Type	Connection	Device Name	Reference tag	Address	Access Mode
1	Fan_start	BOOL	Connection_PLC2	PLC2		%QX3.0	Absolute Access

- Access mode
  The access mode depends on the connection creation method and the connected device driver.
  Symbolic access: When a variable is associated with an identified connection or the connected device driver supports EtherNet/IP protocol, the access mode is displayed as symbolic access.
  Absolute access: When a variable is associated with an identified connection or the connected device driver supports Modbus TCP protocol, the access mode is displayed as absolute access.

Range property

Range upper/lower limit: Sets the measuring range.

Acquisition property

Acquisition cycle: The time interval at which the HMI device reads data from other devices. Configuration range: 50 ms to 3,600,000 ms.

CAUTION

When an IT9000 series device communicates with an EVO1000 series PLC, if the acquisition cycle of the communication variable is set to less than 500 ms, the new value may flicker after being written via the screen (symptom: the new value briefly jumps back to the old value before being written, and then returns to the new value). To avoid this phenomenon, it is recommended to set the acquisition cycle to 500 ms or longer.

Byte property

High/Low byte swap: Set as required for the field application. Supported data types: UINT, INT, UDINT, DINT, ULINT, LINT, REAL, and LREAL.

Linear scaling

Configures the value ranges for connection object variables and HMI variables to derive the conversion relationship between value. Supported data types: USINT, SINT, UINT, INT, UDINT, DINT, ULINT, LINT, REAL, and LREAL.

Numeric property

Initial value: Value displayed between downloading the variable configuration to the monitoring device and acquiring the first set of data.

Retain

Retention of internal variables: Determines how internal variable values are selected when the device recovers from an error.  Restore to value before error; Restore to initial value

These settings affect the behavior of the **"Retain the variable's current value"** option in the "Save As Target File" dialog. For details, see 13.13.2 Saving As Compiled File.

13.2.5.4 Setting Properties in Batch

Variable properties support batch processing. You can hold and pull down the square 

in the lower right corner of the cell () to select the range for batch processing.

In the pop-up "Reminder" window, select "**Cover**" to complete batch modifications of multiple rows of properties.

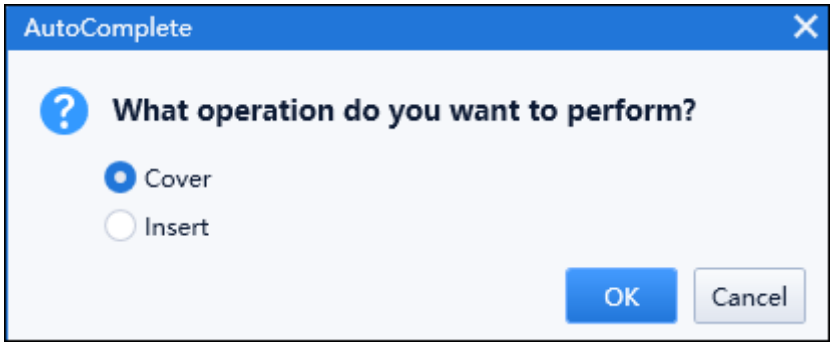


Figure 13-2-5-4-1 Set operation method

13.2.5.5 Multiplex Variables

Multiplexed variables access multiple PLC memory locations within an address range using a single variable, enabling read/write operations without defining individual variable for each address.

When multiplexing with symbolic addressing, multiplexed variables and index variables can be used to access different elements of array variables of the connected PLC.

Prerequisites

- Support one-dimensional arrays with standard data types.
- Up to two index variables can be configured by default, which are the first two indices of the array, for example, 1 and 2 in **Struct1_inst[1].A[2].**

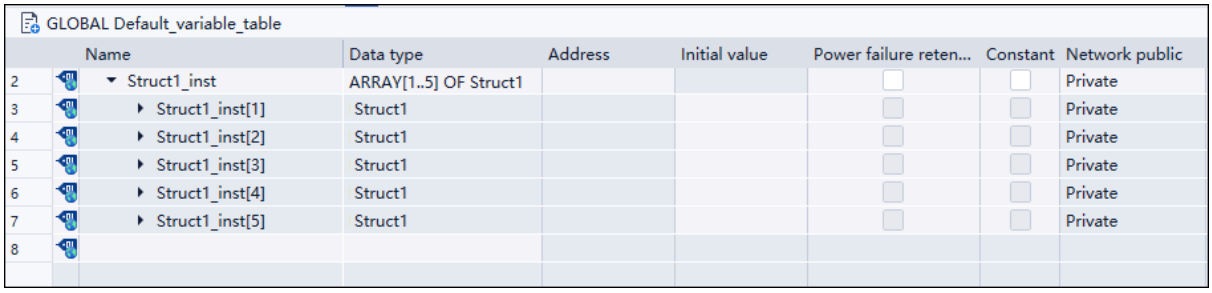
- Support communication protocols: Inovance EtherNet/IP Tags protocol and AC 800 Tags protocol.

Requirements

The PLC device and HMI device have been added.

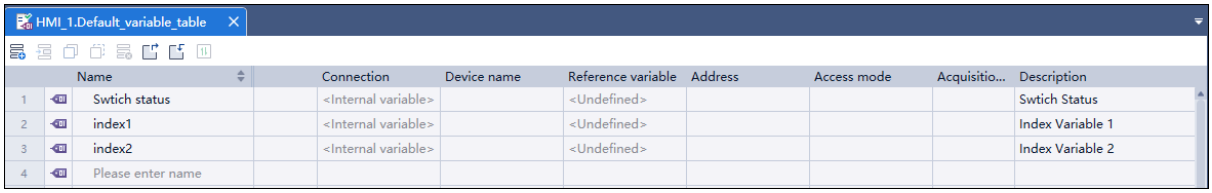
Operation

1. Add a one-dimensional array variable in the PLC
   Add a one-dimensional array variable in the variable table under the "Programming" node of the PLC device. For example, add a one-dimensional array named **Struct1_inst** in the default variable table, with data type as user-defined structure **Struct1**.

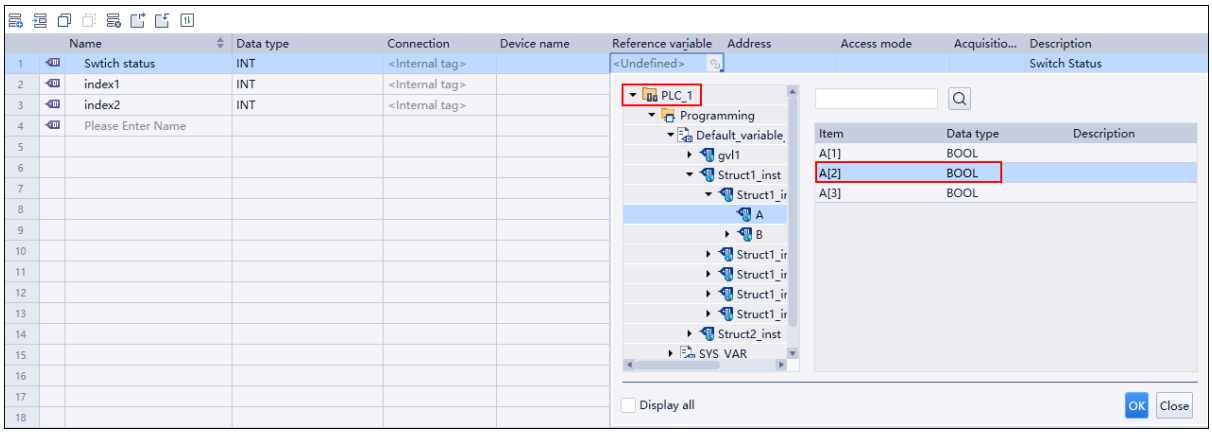


2. Set the "Network public" property of the one-dimensional array, for example, set it to **read-only**.

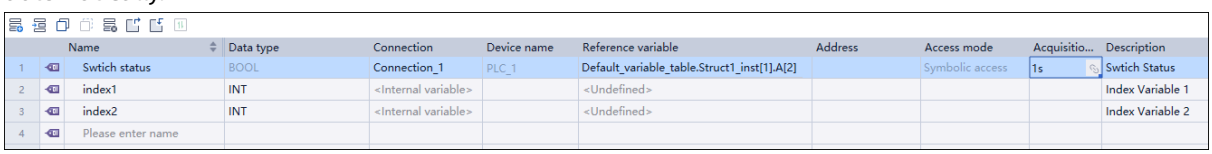
3. In the variable table under the "Variable" node of the HMI device, add three regular variables. For example, add regular variables named **Switch status, index1**, and **index2** in the default variable table.



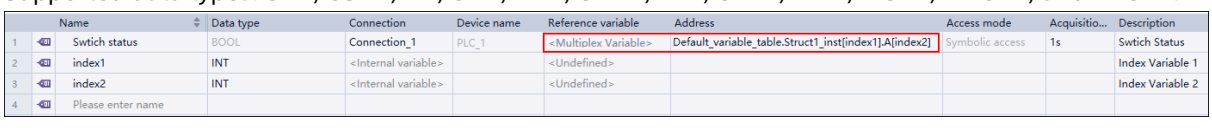
4. Associate PLC variables with reference variables, for example, associate **Switch status** with the variable **Struct1_inst[1].A[2]** in the **default variable table** of PLC_1 device.



Select "OK" and enter the "Reference variable". The connection between PLC_1 and HMI_1 will be created automatically.



5. Set the index address of the reference variable
   In the address cell of **Switch status**, use variables index1 and index2 to control access to array element indexes.
   Supported data types: SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, BYTE, WORD, DWORD, and LWORD.



The reference variable location for **Switch status** is identified as **<Multiplex Variables>**.

> NOTE
>
> If an index variable is accidentally deleted during configuration without error clearing, compilation or saving, its association will be automatically restored when a new variable with the same name is created or an existing variable is renamed to the index variable name.

During online operation, if the index variable value exceeds the sub-element index range of the multiplexed variable:

- The multiplexed variable value is abnormal and marked .
- In the F(t) curve view, the historical trend is connected by a line from the moment the range is exceeded until it returns to normal.

13.2.5.6 Storing Variable Data

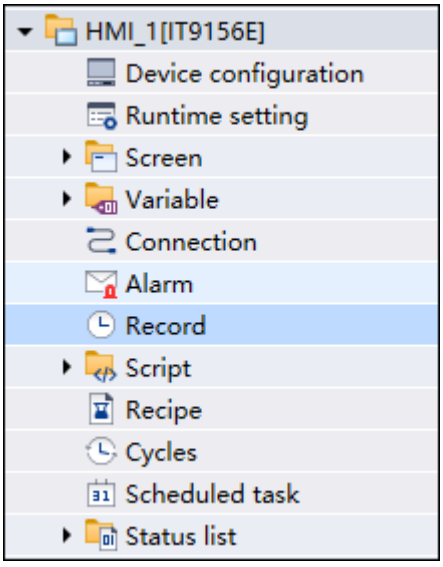
- 13.2.5.6.1 Setting Recording Rules for Data
- 13.2.5.6.2 Variable-Associated Recording Rules

13.2.5.6.1 Setting Recording Rules for Data

Data logging is used to collect, process, and record process data from field devices, reflecting device status such as temperature or motor run/stop status.

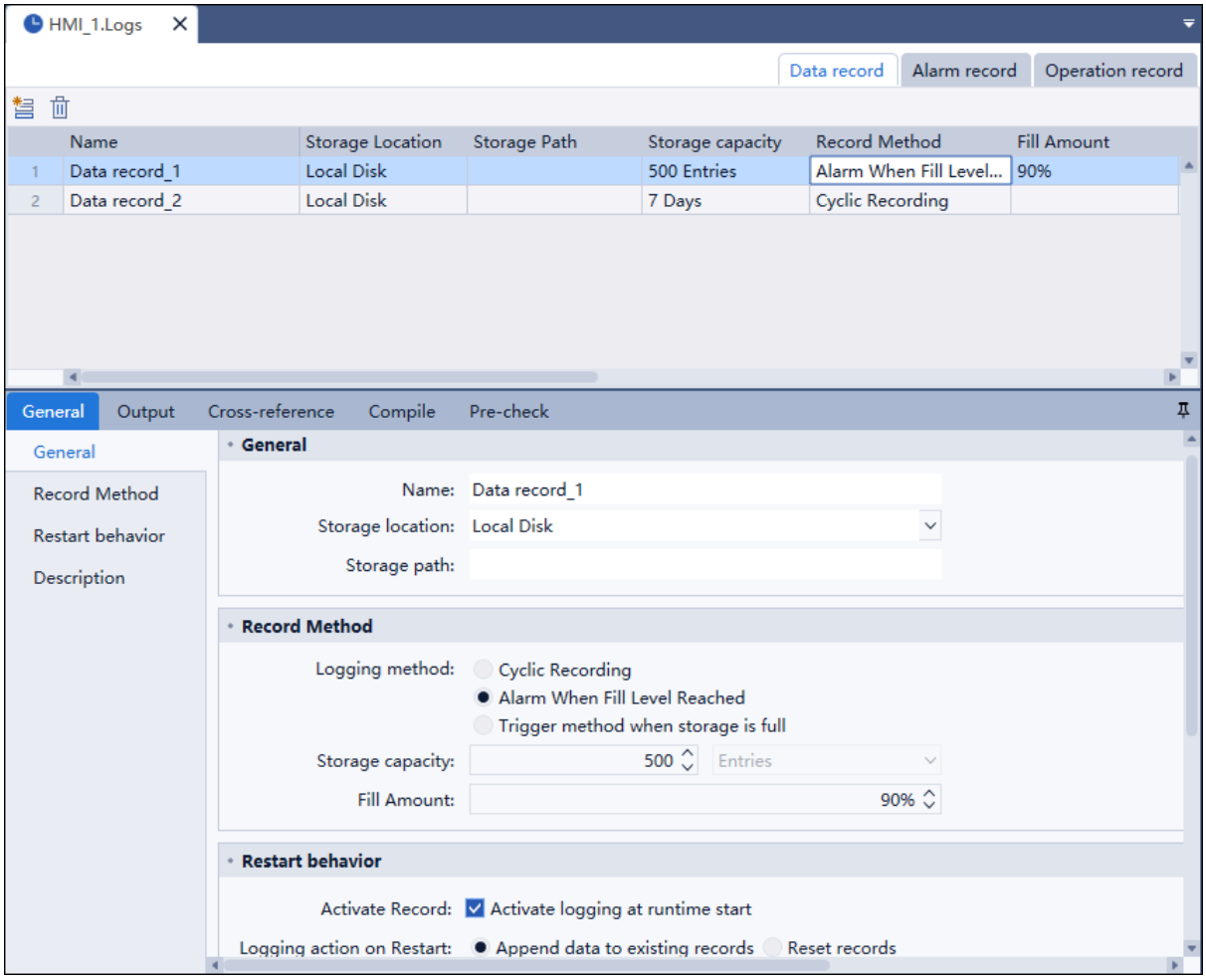
Logging rules specify the storage location and the handling method when the storage space is full.

1. Double-click the "Record" node in the project tree to open the settings interface.



2. Switch to the "Data record" tab.
3. Add logging rules.

   Use right-click commands or icons in the editing area. A maximum of 256 rows can be added. Each row represents one logging rule, and rules are independent of each other.
4. Set storage parameters.
   Detailed parameters for data recording are displayed in the monitoring area below the window.



The following table introduces the properties of monitoring area in top-to-bottom order.

Type	Parameter	Description
General Settings	Name	By default, named as "Data record_n", where n starts at 1 and increments automatically. For details, see Naming rules.
	Storage location	The options for storage location depend on the device hardware. There are three typical categories: ◦ Local disk: The disk path of the monitoring device hardware, which does not need to be specified. Touch screen: Default path is **\Inodata**; PC: Default path is **D:\Logs**. ◦ SD card: Default path is **/Logs**. ◦ USB: Default path is **/Logs**.
	Storage path	

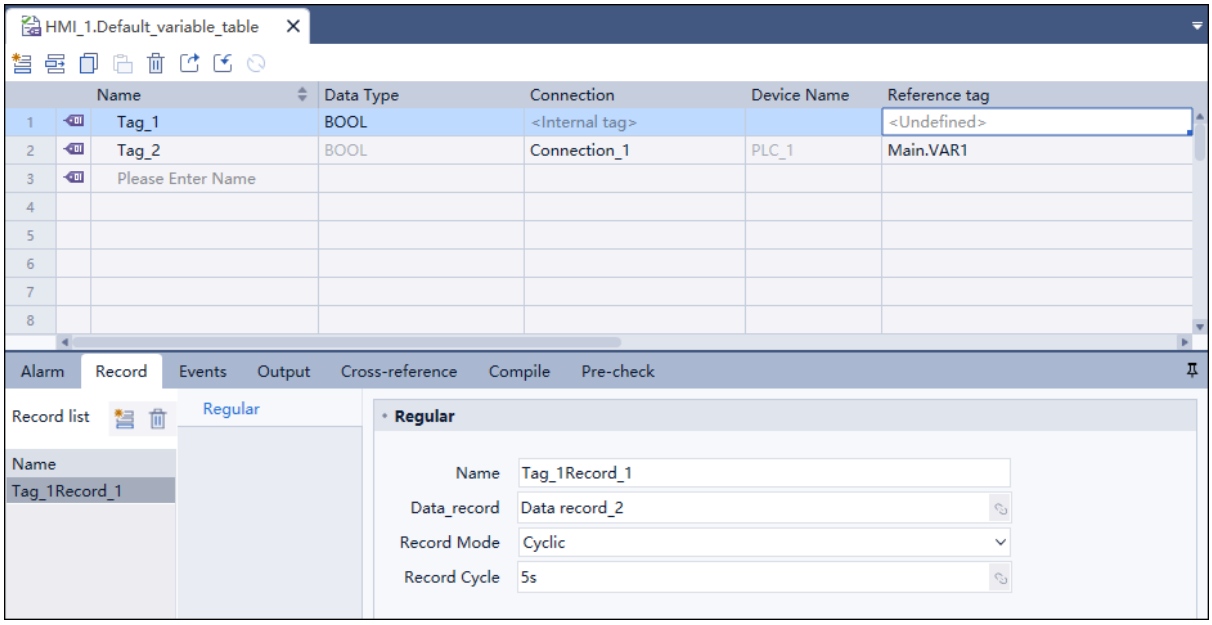
Type	Parameter	Description
Recording method	Recording method	◦ Cyclic recording: When storage space is full, the oldest data is overwritten by the latest data. ◦ Alarm at fill level: When storage usage reaches a high threshold (e.g., 90% as configured), the iFA software triggers a system alarm to indicate near-full storage and prompt space release. After the system alarm is triggered, data logging stops until the space usage drops below the fill level. ◦ Trigger at full storage: An overflow event is triggered when storage is full, and data logging stops. The predefined overflow event is used to free up storage space. Common events include 19.6.1 Clearing a Record, 19.6.6 Exporting a Data Record, 19.6.2 Starting Recording, and 19.6.3 Stopping Recording. For details, see 19.6 Record.
	Storage capacity	Set according to the number of data entries or storage days, which need to be estimated based on data volume. Number of records: 1 to 5,000,000; default: 500 Days range: 1 to 10,000; default: 7
	Fill level	Storage space utilization. The setting range is 1% to 100%.
Restart behavior	Activate record	Enable data recording when the system starts. This option is selected by default.
	Record handling on restart	Handling of logged data when the monitoring device restarts. Append data to existing records: Retain existing data and add new data. Reset records: Clear existing data and start recording anew.
Description	Description	Remarks on the current logging rules.

Specify the storage location and configure the handling mechanism for full storage space.

13.2.5.6.2 Variable-Associated Recording Rules

To store values of variables in the default and regular variable tables, you must associate variables with applicable logging rules, so target variable values are saved to the specified location.

1. Open the variable table.
2. Select and set the target variable.
   Click the target variable row and its storage properties are summarized in the "Record" tab of the monitoring area.



You can click the icon to establish storage conditions for variables. Up to 16 conditions are supported per variable.

Parameter	Description
Name	Name of the variable logging condition. The default name isVariableName_Record_n, where n starts at 1 and increments automatically.
Data record	Associated with a predefined logging rule.
Recording mode	◦ Cyclic logging: Logs values according to the time cycle specified in "Record cycle". A variable supports data logging at multiple time intervals, but duplicate configurations at the same time are not allowed. ◦ Change-based logging: Data is logged only when the variable value changes. A variable can be configured only once. ◦ On-demand logging: The logging action is dynamically controlled by user-preset trigger conditions. Each trigger condition is judged based on the specific bit status of one variable (trigger variable). Multiple trigger conditions can be configured for one variable. For example, when bit 3 of variable A changes to 0, the current value of variable C is logged.

Parameter	Description
Trigger mode	Adapt to **on-demand logging** mode. When the value of the specified bit of the trigger variable changes (e.g., 0→1) and the trigger mode is satisfied (e.g., rising edge), the value of the target variable is logged once. The data type of the trigger variable supports integer and BOOL types: ◦ For BOOL variable, the bit number is set to 0. ◦ For integer variable, the lower limit of the bit number is 0, and the upper limit is the total number of occupied bits minus 1. For example, the upper limit for INT is 16-1=15, so the bit number configuration range is 0 to 15.
Trigger variable	
Bit number	

The variable logging conditions have been created and set successfully.

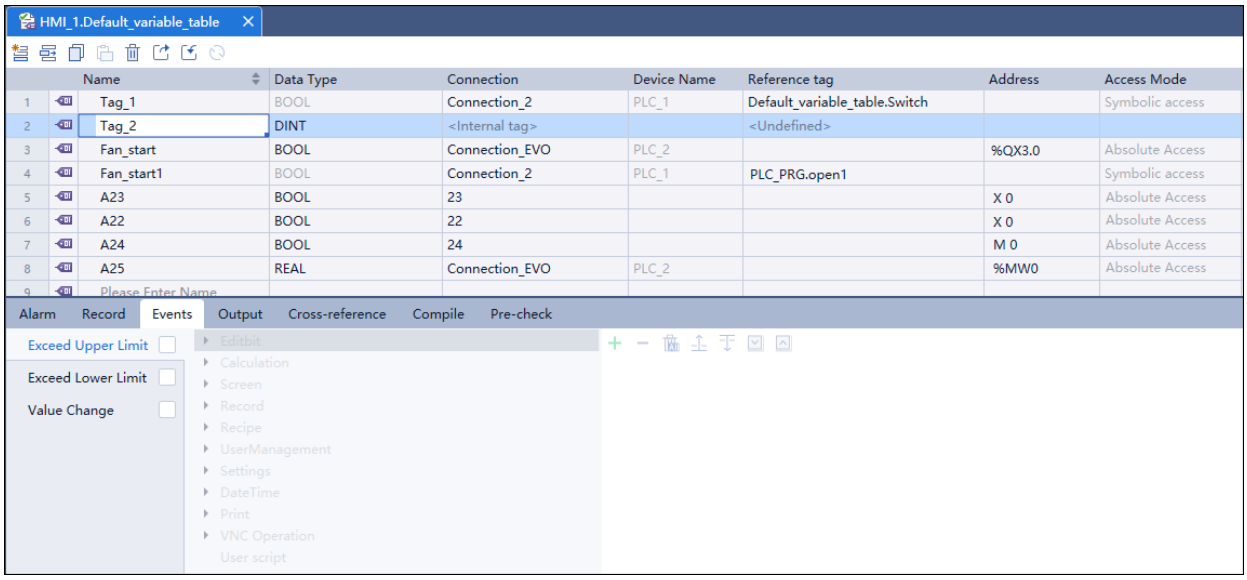
NOTE

Record variables with the same logging rule in the same data record to reduce redundancy.

13.2.5.7 Configuring Variable Events

The status of the variable value serves as the trigger condition. The preset operation is executed when the condition is met.

Select any variable row in the variable table, and its corresponding event properties will be displayed in the monitoring area.



Trigger condition

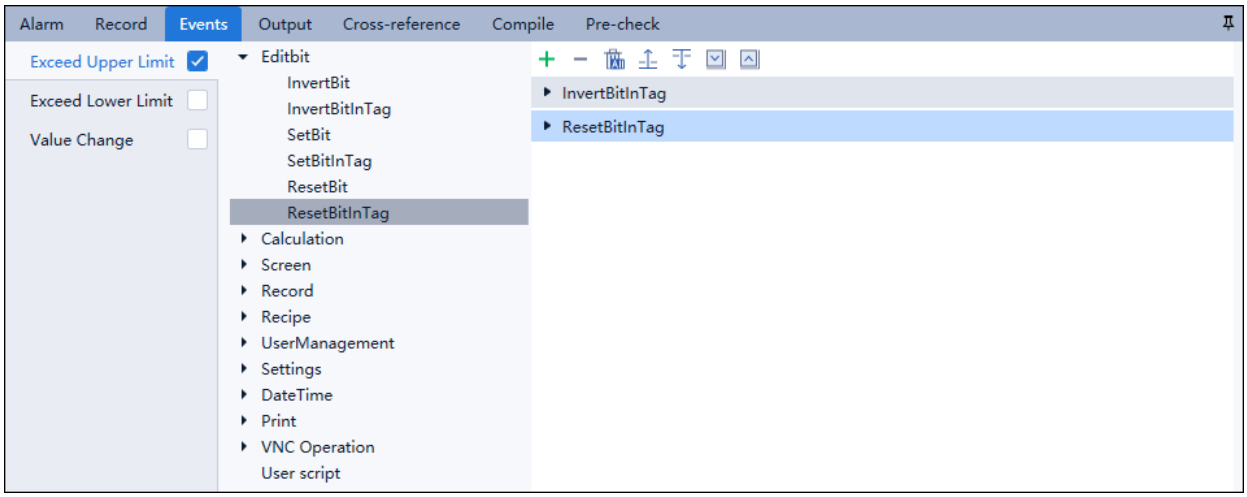
The trigger condition for the event, which is set based on the variable value.

Out of upper/lower limit: The event configuration is triggered when the real-time value of the variable exceeds the upper or lower limit of the measuring range. Applicable to integer variables.

Value change: The event configuration is triggered when the real-time value of the variable changes. Applicable to integer and Boolean-type variables.

Event configuration

A variable supports adding multiple events. When the trigger condition is met, events are executed in top-to-bottom order.



13.2.6 Operating Variables

13.2.6.1 Copying and Pasting Variables

Copying variables

Copying variables means duplicating all property information of the target variable, including alarms, records, and event configurations.

1. Select the target variable row.

   Click the identifier  of the variable row to select the target variable. For multiple selections, use the **Ctrl** or **Shift** key to pick objects (or by dragging).

2. Command source.

   Click  or select "Copy" from the right-click menu.

Pasting variables

In the target variable table, click the icon  or select "Paste" from the right-click menu. The pasted variable is placed below an existing variable, with the variable name appended with a "_n" suffix, where n starts at 1 and increments.

The paste operation is supported both within the same table and across pages. Cross-table operation refers to copy a variable from Variable Table A and pasting it into Variable Table B.

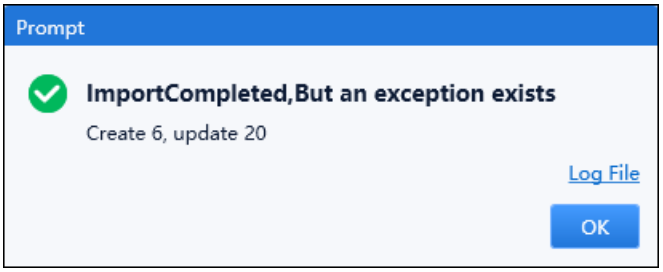
13.2.6.2 Importing and Exporting Variables

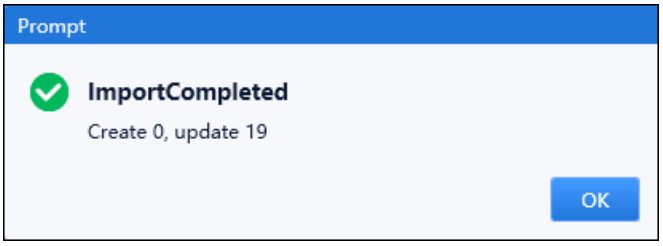
The import and export function of the variable enables cross-device reuse of variables and reduces configuration time.

Importing variables

Variables are imported into the iFA software in table file format (.xlsx). For constraints of the table file, see Importing a variable table.

Provided that the variable table meets import requirements, the following mainly describes the processing rules for abnormal variable information during the import process.

<table>
<tr><th>No.</th><th>Import File</th><th>Rules</th><th>Example</th></tr>
<tr><td>1</td><td>Variable properties not configured</td><td>If the variable table provides only the variable name with all other items empty, the software will automatically fill with default values after import.</td><td rowspan="3">

Storage path of log file: Inovance\iFA\Bin\config\ExImport under the iFA software installation directory.</td></tr>
<tr><td>2</td><td>Variable properties written incorrectly</td><td>Property errors in the variable table, such as writing "bool" instead of "Bool" for the data type, will be automatically filled with default values by the software after import.</td></tr>
<tr><td>3</td><td>Variable name conflict</td><td>From top to bottom, only the first occurrence of a variable is imported. Subsequent variables with the same name are discarded and recorded in the log file.</td></tr>
<tr><td>4</td><td>Conflict with existing variables in the software</td><td>Existing variables with the same name are overwritten after import.</td><td>
</td></tr>
</table>

Exporting variables

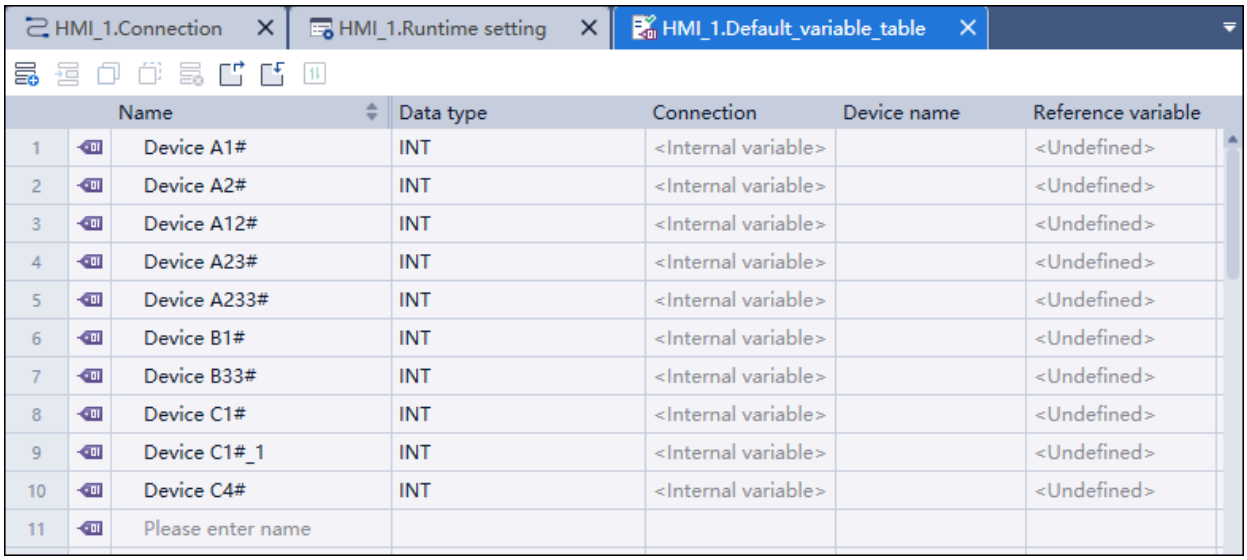
The operation for exporting a variable is the same as that for exporting a variable table. For details, see 13.2.4.3 Importing and Exporting a Variable Table.

When exporting multiplexed variables for symbolic access, the "Reference variable" column records the initial state of the referenced PLC variable, while the "Address" column records the state of the configured index variable. The differences are shown in the figure below.

Name	Data Type	Connection	Device name	Reference variable	Address	Access mode
Swtich status	BOOL	Connection_1	PLC_1	Application.Default_variable_table.Struct1_inst[1].A[2]	Application.Default_variable_table.Struct1_inst[index1].A[index2]	Symbol access
index1	INT					
index2	INT					

13.2.6.3 Sorting Variables

New variables added to the variable table are sorted by the order in which they were added by default. The figure below shows an example of ascending order sorting.



To adjust the sorting, you can click the header of the "Name" column to switch between the following three sorting methods.

- Default sorting: Sort variables in the order they were added, with identifier .
- Name ascending: Sort variables in ascending order based on the content of names (same rules as sorting 13.3.4 screen names), with identifier .
- Name descending: Sorts variables in descending order based on the content of names, with identifier .

You can click the header to cycle through the three methods, and quickly locate and organize variables.

13.3 Creating a Screen

- 13.3.1 Screen Types
- 13.3.2 Adding a Screen
- 13.3.3 Screen Properties
- 13.3.5 Selecting Objects
- 13.3.6 Gallery Usage
- 13.3.7 Configuring a Screen
- 13.3.8 Configuring Animation
- 13.3.9 Using Layers
- 13.3.10 Screen Management

13.3.1 Screen Types

Screens are divided into four types according to application scenarios and displayed content: main screen, picture-in-picture, pop-up screen, and template screen.

- Main screen
  The main screen serves as the basic screen and provides the user interface for monitoring and operation. It is primarily used for centralized monitoring of key processes and data. Applications are created by calling up pop-up screens from the main screen or switching to other main screens. A main screen can have up to 500 objects configured.
- Picture-in-picture
  Picture-in-picture screens are only displayed and switched within the 13.3.5.2.16 Picture-in-Picture View, which is usually configured on the main screen. You can monitor the main screen while browsing content in the picture-in-picture view.
- Pop-up screen
  Pop-up screens are used to configure additional content for the main screen, such as displaying parameters of a specific device type, and are called up from the main screen during runtime.
  Pop-up screens are template screens based on the concept of "Page parameter". When configuring screen elements, you can either choose page parameters as a priority or select variables from the variable table for configuration. When a pop-up screen is opened at runtime, variables are passed to replace the page parameters in the pop-up screen.
- Template screen
  A template screen can be regarded as a common screen for main screens. Objects configured in the template are displayed on all main screens that reference this template.
  If the content of the template changes, all referenced screens will automatically update.

13.3.2 Adding a Screen

Screens are classified into four types by purpose: main screen, picture-in-picture, pop-up screen, and template screen. They are all located under the **"Screen"** node in the project tree. All types except the main screen are managed in separate categories.

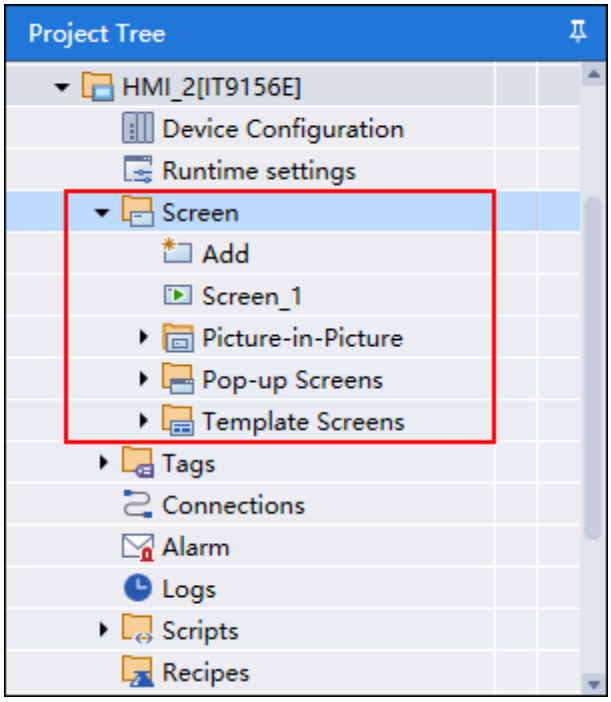


Figure 13-3-2-1 Screen node

A single HMI device can be configured with up to 2000 screens, regardless of screen type. All screen types are added via the project tree, with two supported methods:

- In the right-click menu of the parent node, select the add command, such as "Add screen".
- Double-click the "Add" node under the parent node.

The result is shown in the figure below. The screens named **Screen_1** and **Screen_2** in the figure are main screens.

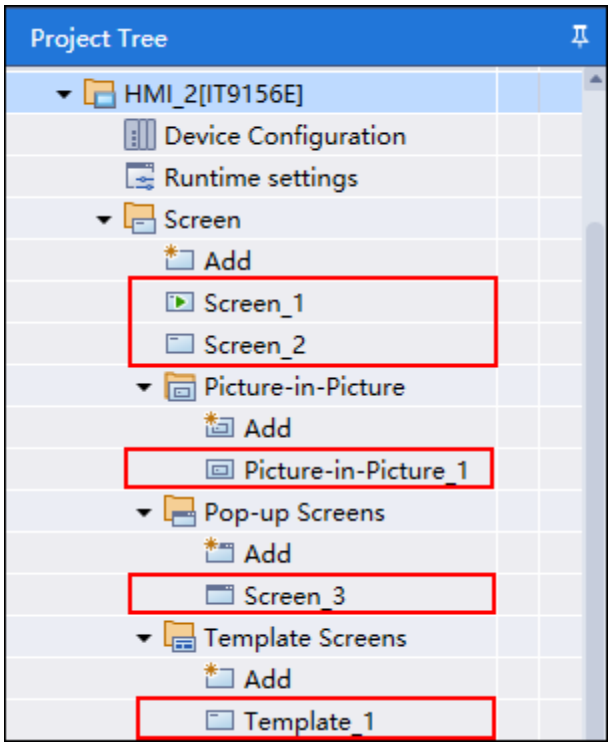
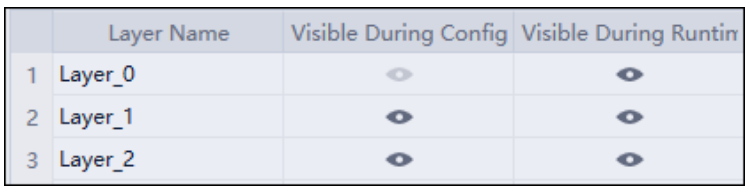


Figure 13-3-2-2 Added screens of each type

13.3.3 Screen Properties

After a screen is added, you need to configure its properties according to the control strategy. Property items vary by screen type. The table below summarizes all screen properties, with notes on the applicable screen objects for easy reference.

Table13-3-3-1 Screen properties

Item		Description	Applicable Screen Type
Basic settings	Dimension	Screen format size. Only picture-in-picture and pop-up screens support custom sizing. For picture-in-picture screens, height range: (60, 4 × screen height), default: ½ screen height; width range: (100, 2 × screen width), default: ½ screen width The sizes of other screens are set automatically according to the screen size of the monitoring device.	Applicable to all four types of screens
Window	Background color	Background color of the screen	Applicable to all four types of screens
	Border style	Style of pop-up window during runtime	Applicable to pop-up screens
	Reference template	Configuration of screen template reference	Applicable to main screens
	Reference screen	Screen reference configuration	Applicable to template screens
Layer	Configuration fully visible	All layers in the screen are visible during configuration. This option is selected by default.	Applicable to all four types of screens
	Runtime fully visible	All layers in the screen are visible during runtime. This option is selected by default.	Applicable to all four types of screens
	Active layer	Selects the current active layer. For details, see 13.3.9.3 Setting an Active Layer.	Applicable to all four types of screens
	Display/ hide configuration	Sets whether each layer is visible during configuration and runtime. Click the icon to toggle the visibility state. For more information, see 13.3.9.4 Displaying or Hiding A Layer.  	Applicable to all four types of screens

Item		Description	Applicable Screen Type
Page parameters	Page parameters	Configures screen parameter. In a pop-up screen, all associated symbols can be associated with either regular variables in monitoring or page parameters of this screen.	Applicable to pop-up screens
Basic information	Display title	The title is displayed in the title bar of the pop-up screen window. This option is selected by default.	Applicable to pop-up screens
	Close button	The close button is displayed in the title bar of the pop-up screen window. This option is selected by default.	Applicable to pop-up screens
	Resizable	The size of the pop-up screen window can be adjusted by dragging. This feature is not supported by default.	Applicable to pop-up screens
	Movable	The pop-up screen can be moved. This option is selected by default.	Applicable to pop-up screens
	Modal	The pop-up screen opens in modal mode. This feature is not supported by default.	Applicable to pop-up screens
	Title text	The title content is displayed in the title bar of the pop-up screen.	Applicable to pop-up screens
	Screen ID	ID must be unique, with a range of 1 to 65535. The current screen ID can be read via the system variable $CurrentScreenId.	Applicable to all four types of screens
	Name	Name of the screen.	Applicable to all four types of screens

13.3.4 Renaming a Screen

When a screen is created, its default name is "Screen_SequenceNumber". The name can be changed according to the following rules.

Naming rules

The naming rules must satisfy all the following conditions.

- Letters (case-insensitive), numbers (0 to 9), Chinese characters, and spaces are supported.
- The length cannot exceed 128 characters.
- The name cannot start or end with a space.

- Invalid characters: \ / : * ? " < > |
- Screen names must be unique.

Sorting rules

After renaming a screen, its order in the project tree is sorted in ascending order based on the following priorities.

Precedence	Character Type	Note
1	English symbol	English punctuations (space, -, etc.)
2	Number	Sorted by actual numeric value, not by character-by-character comparison For example, in ascending order: A1→A2→A11.
3	Letter	Sorted in a-z order
4	Chinese symbol	Chinese punctuations
5	Chinese characters	Sorted by the first letter of the Pinyin of the first character. If the first character is the same, compare the second character, and so on.
6	Special character	Characters not within the above ranges are sorted last.

13.3.5 Object Selection

Objects are an important part of process screens. They are divided into two parts: elements and graphic symbols.

Elements visually display equipment operating data and status, enhance monitoring, simplify operation and interaction, support alarms and notifications, and enable customized visualization. Elements are stored in the "Toolbox" and categorized into five types based on application scenarios.

- 13.3.5.1 Basic Elements
- 13.3.5.2 Control and Display
- 13.3.5.3 Data Visualization
- 13.3.5.4 Production and Management
- 13.3.5.5 Others

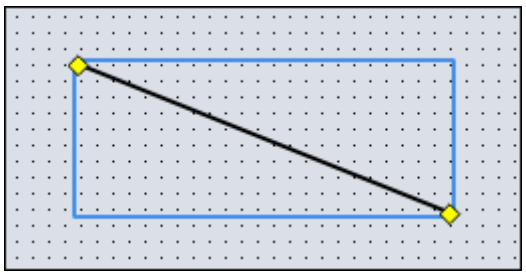
13.3.5.1 Basic Elements

13.3.5.1.1 Line

The "line" is an open object.

The length and slope of a line are defined by the height and width of the bounding box.

There is a yellow control point at each end of the line, which can be directly dragged to change their position.



- **Line creation method**

There are three methods to create a "line":

  - Method 1: Select the object from the toolbox, drag it into the screen area, and release the mouse button to create a default-sized element.
  - Method 2: Click the object in the toolbox, and then click the screen to create a default-sized element.
  - Method 3: Click the object in the toolbox, click the screen to define the starting position of the element, and drag to the end position before releasing. This creates an element of the desired size.

- **Line properties**
  - Basic Settings

Item	Description
Position	Sets the position coordinates (i.e., top-left coordinates) of the element on the screen in pixels: X: X-axis coordinate. Y: Y-axis position coordinate. The coordinate value must be smaller than the screen boundary coordinate value to prevent the elements from extending beyond the screen.
Dimension	Indicates the size of the element on the screen in pixels. The size can be adjusted by manually dragging the control points at both ends. The minimum size can be adjusted to 0 (width) * 0 (height).
Start position	Sets the starting position of the element on the screen in pixels: X: X-axis, position coordinate at the top of the screen. Y: Y-axis, position coordinate at the left of the screen.
End position	Sets the ending position of the element on the screen in pixels: X: X-axis, position coordinate at the top of the screen. Y: Y-axis, position coordinate at the left of the screen.

  - Stroke:

Property	Description
Stroke color	Line color and transparency.
Line width	The range is 1 to 10, with a default of 2.

Property	Description
Line type	Line type
Start arrow	Start arrow style of the line
End arrow	End arrow style of the line

◦ Basic information:

Item	Description
Graph element ID	Global identifier number of the element in the system. (automatically assigned by default)
Type	Type information of the element.
Name	Name of the element. The default name is "Element_1", for example "Line_1".
Enable tooltip	Sets whether to use tooltip information: • Selected: Used. • Unselected: Not used. No tooltip by default.
Runtime display	Specifies whether the element is visible in runtime. This option is selected by default.
Enable	Sets whether the element can be operated in the runtime. This option is selected by default.
Permissions	Sets the element permission.
Tooltip	The tooltip information is displayed when the mouse hovers over this element in the runtime. It is empty by default.
Layer	Sets the layer to which the element belongs. The default is layer_0.

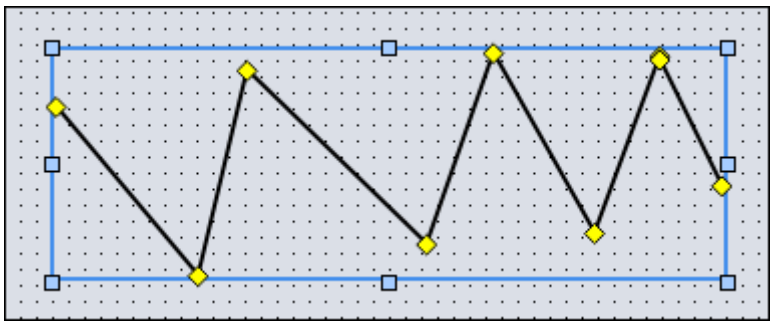
◦ Animation: For details, see 13.3.8 Animation Configuration.

◦ Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.1.2 Polyline

A "polyline" consists of connected segments and can have indefinite number of control points (numbered in the order in which they were created).

There is a yellow control point at each corner of the polyline, which can be directly dragged to change their position.



- **Polyline creation**

There are two ways to create a "polyline":

  - Method 1: Select the object in the toolbox without releasing the mouse, drag it into the screen area, and then release the mouse to obtain a default-sized element.
  - Method 2:
    1. Click the target element in the toolbox and move the mouse to the screen.
    2. Click the left mouse button to set the starting position of the polyline, drag the mouse to the target position of the first line segment, and then click the left mouse button again to complete the first segment of the polyline.
    3. Repeat the above steps until the final segment is drawn, and then right-click to generate the desired element.

- **Polyline properties**
  - Basic settings:

Item	Description
Position	Sets the position coordinates (i.e., top-left coordinates) of the element on the screen in pixels: X: X-axis coordinate. Y: Y-axis position coordinate. The coordinate value must be smaller than the screen boundary coordinate value to prevent the elements from extending beyond the screen.
Dimension	Sets the size of the element on the screen in pixels. Setting range: 0 to 10000
Rotation	Sets the rotation angle of the element. Range: (-180 to 180].
Relative center	Sets the rotation center position of the object. This position is an offset relative to the object physical center.
Absolute center	Sets the rotation center position of the object. This position is an offset relative to the entire screen.

  - Stroke:

Property	Description
Stroke color	Polyline color and transparency.
Line width	Polyline line width. The range is 1 to 10, with a default value of 2.
Line type	Line type.
Start arrow	Start arrow style.
End arrow	End arrow style.
Corner point	Sets the corner style of the polyline.

- Point set:

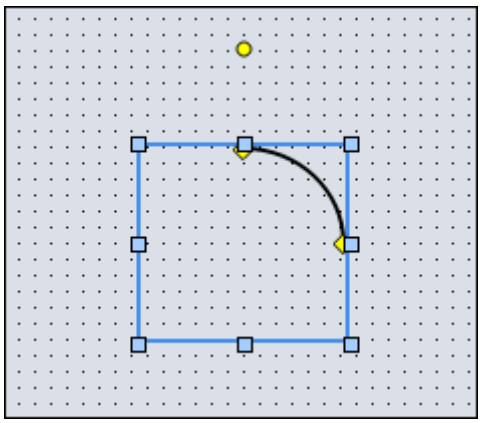
Property	Description
Add point set	Click the "Add" button to add a polyline segment.
Delete point set	Click the "Delete" button to remove a polyline segment.
Edit point set	Double-click the X, Y coordinates in the point set to modify the coordinate information of each segment.

- Basic information: For details, see Line > Basic Information.
- Animation: For details, see 13.3.8 Animation Configuration.
- Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.1.3 Arc

The "arc" displays curves of all types.

There is a yellow control point at each end of the arc, which can be directly dragged to change their position.



- **Arc creation**

  There are three ways to create an "arc". For details, see the line creation method.

- **Arc properties**
  - Basic settings:

Property	Description
Position	Sets the position coordinates (i.e., top-left coordinates) of the element on the screen in pixels: X: X-axis coordinate. Y: Y-axis position coordinate. The coordinate value must be smaller than the screen boundary coordinate value to prevent the elements from extending beyond the screen.
Dimension	Sets the size of the element on the screen in pixels. Setting range: 1 to 10000
Rotation	Sets the rotation angle of the element. Range: (-180 to 180].
Relative center	Sets the rotation center position of the object. This position is an offset relative to the object physical center.
Absolute center	Sets the rotation center position of the object. This position is an offset relative to the entire screen.

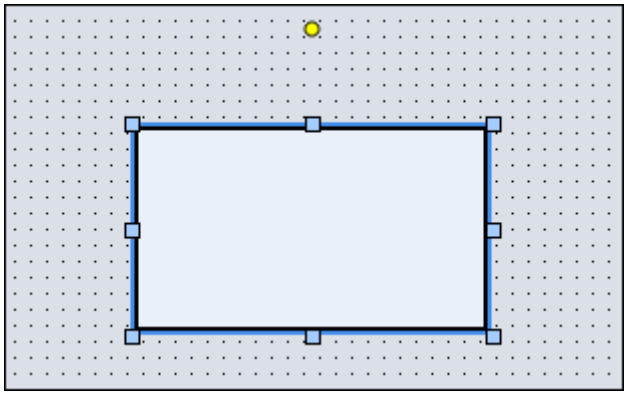
- Stroke properties: For detail, see Line > Stroke.
- Angle:

Item	Description
Start angle	The angle is measured clockwise from the X-axis to the starting point of the arc (in degrees).
Sweep angle	The angle is measured clockwise from the startAngle parameter to the ending point of the arc (in degrees).

- Basic information: For details, see Line > Basic Information.
- Animation: For details, see 13.3.8 Animation Configuration.
- Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.1.4 Rectangle

The "rectangle" is a closed object that can be filled with a background color.



- **Rectangle creation**

  There are three ways to create a "rectangle". For details, see the line creation method.

- **Rectangle properties**

  - Basic settings:

Item	Description
Position	Sets the position coordinates (i.e., top-left coordinates) of the element on the screen in pixels: X: X-axis coordinate. Y: Y-axis position coordinate. The coordinate value must be smaller than the screen boundary coordinate value to prevent the elements from extending beyond the screen.
Dimension	Sets the size of the element on the screen in pixels. Setting range: 1 to 10000
Rotation	Sets the rotation angle of the element. Range: (-180 to 180].
Relative center	Sets the rotation center position of the object. This position is an offset relative to the object physical center.
Absolute center	Sets the rotation center position of the object. This position is an offset relative to the entire screen.
Rounded corner	Sets the rounded corner width and height of the rectangle. A rounded rectangle is created by changing the rounded corner width and height.

  - Style:

Property	Description
Draw options	Sets the drawing options for the element. Options: draw, fill, and draw and fill.

  - Percentage fill:

Property		Description
Percentage fill mode		There are two fill modes: absolute direction fill and relative direction fill. Absolute direction fill: Regardless of the orientation of the element, the fill direction is always aligned with the absolute vertical direction. Relative direction fill: When the shape of the element changes, the fill direction remains relative to the vertical orientation of the element.
Percentage fill brush	Solid brush	Color: Sets the solid brush color.
		Transparency: Sets the brush transparency to change the brush display effect (0% to 100%).
	Hatch brush	Fill style: Sets the hatch pattern for filling the element.
		Foreground color: Sets the foreground color of the hatch brush to achieve foreground filling of the element.
		Foreground transparency: Sets the brush transparency to change the brush display effect (0% to 100%).
		Background color: Sets the background color of the hatch brush to achieve background filling of the element.
		Background transparency: Sets the brush opacity to change the brush display effect (0% to 100%).
	Texture brush	Fill style: Sets the texture pattern for filling the element.
		Fill mode: Sets the texture fill mode (including normal, tile, and stretch).
		Transparency: Sets the brush transparency to change the brush display effect (0% to 100%).
	Linear brush	Fill style: Sets the linear pattern for filling the element.
		Cycle count: Sets the number of linear brush fill repetitions.
		Rotation angle: Sets the fill rotation angle.
		Start color: Sets the starting color of the linear brush.
		Start color transparency: Sets the brush transparency to change the brush display effect (0% to 100%).
		End color: Sets the ending color of the linear brush.
		End color transparency: Sets the brush transparency to change the brush display effect (0% to 100%).
	Path brush	Gradient style: Sets the gradient style of the path brush (including circular and rectangular).

Property		Description
		Cycle count: Sets the number of path brush fill repetitions.
		Center size: Sets the center size of the path brush (0% to 100%).
		Start color: Sets the starting color of the linear brush.
		Start color transparency: Sets the brush transparency to change the brush display effect (0% to 100%).
		End color: Sets the ending color of the linear brush.
		End color transparency: Sets the brush transparency to change the brush display effect (0% to 100%).
Vertical fill direction		Specifies the fill direction: fill from top to bottom or from bottom to top.
Horizontal fill direction		Specifies the fill direction: fill from left to right or from right to left.
Vertical fill percentage		Sets the vertical fill percentage value of the element (0% to 100%).
Horizontal fill percentage		Sets the horizontal fill percentage value of the element (0% to 100%).

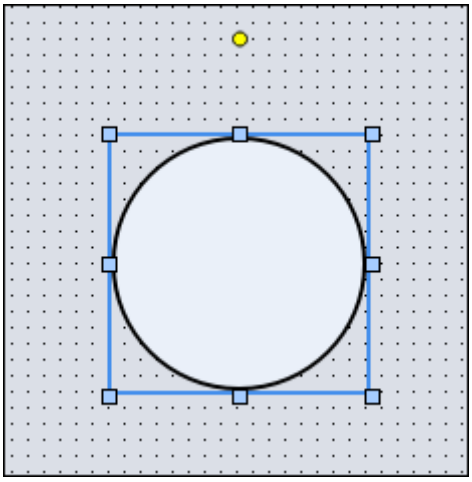
- Fill: For details, see Percentage Fill > Percentage fill brush.
- Border:

Property	Description
Border color	Rectangle color and transparency.
Line width	Rectangle line width. The range is 1 to 10, with a default value of 2.
Line type	Rectangle line type.

- Basic information: For details, see Line > Basic information .
- Animation: For details, see 13.3.8 Animation Configuration.
- Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.1.5 Circle

The "circle" is a closed object that can have a background color filled.



- **Circle creation**

  There are three ways to create a "circle". For details, see the line creation method.

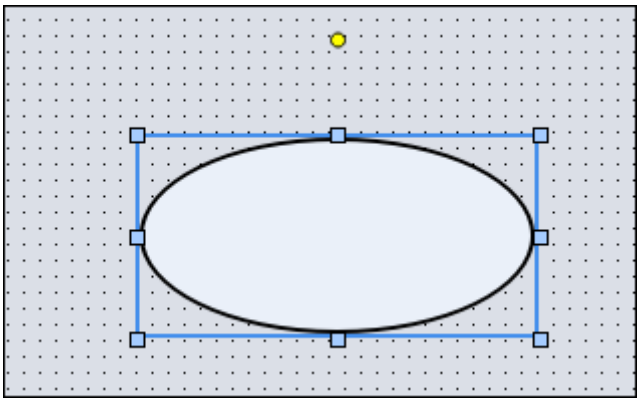
- **Circle properties**
  - Basic settings:

Item	Description
Position	Sets the position coordinates (i.e., top-left coordinates) of the element on the screen in pixels: X: X-axis coordinate. Y: Y-axis position coordinate. The coordinate value must be smaller than the screen boundary coordinate value to prevent the elements from extending beyond the screen.
Dimension	Sets the size of the element on the screen in pixels. Setting range: 1 to 10000
Rotation	Sets the rotation angle of the element. Range: (-180 to 180].
Relative center	Sets the rotation center position of the object. This position is an offset relative to the object physical center.
Absolute center	Sets the rotation center position of the object. This position is an offset relative to the entire screen.
Center point	Sets the position of the circle center.

  - Style: For details, see Rectangle > Ssyle .
  - Percentage fill: For details, see Rectangle > Percentage fill.
  - Fill: For details, see Rectangle > Fill.
  - Border: For details, see Rectangle > Border.
  - Basic information: For details, see Line > Basic information.
  - Animation: For details, see 13.3.8 Animation Configuration.
  - Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.1.6 Ellipse

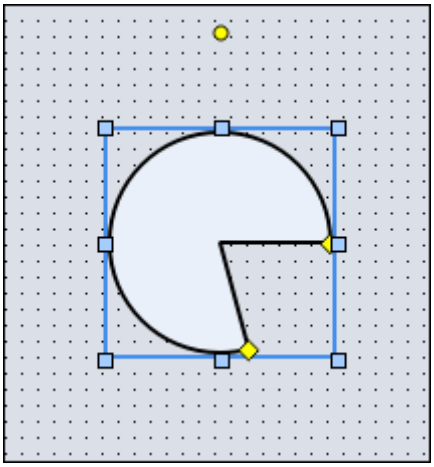
The "ellipse" is a closed object that can be filled with a background color.



- **Ellipse creation**
  There are three methods to create an "ellipse". For details, see the line creation method.
- **Ellipse properties**
  - Basic settings: For details, see Circle > Basic settings.
  - Style: For details, see Rectangle > Style.
  - Percentage fill: For details, see Rectangle > Percentage fill.
  - Fill: For details, see Rectangle > Fill.
  - Border: For details, see Rectangle > Border.
  - Basic information: For details, see Line > Basic information.
  - Animation: For details, see 13.3.8 Animation Configuration.
  - Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.1.7 Sector

The "sector" is a closed object that can be filled with a background color.



- **Sector creation**
  There are three ways to create a "sector". For details, see the line creation method.
- **Sector properties**
  - Basic settings: For details, see Arc > Basic settings.

- Style: For details, see Rectangle > Style.
- Data:

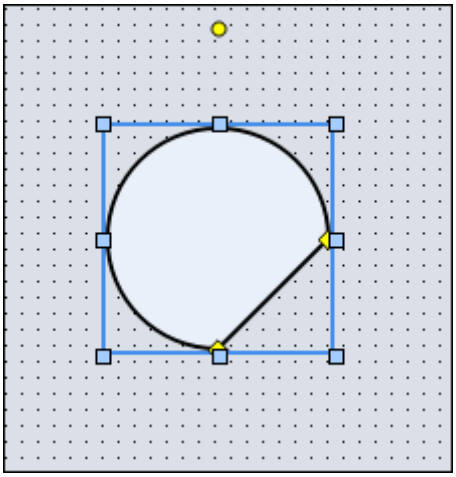
Item		Description
Start angle		The angle is measured clockwise from the X-axis to the starting point of the arc (in degrees).
Sweep angle		The angle is measured clockwise from the startAngle parameter to the ending point of the arc (in degrees).
Percent fill mode		There are two fill modes: absolute direction fill and relative direction fill. Absolute direction fill: Regardless of the orientation of the element, the fill direction is always aligned with the absolute vertical direction. Relative direction fill: When the shape of the element changes, the fill direction remains relative to the vertical orientation of the element.
Percent fill brush	Solid color brush	Color: Sets the solid brush color.
		Transparency: Sets the brush transparency to change the brush display effect (0% to 100%).
	Hatch brush	Fill style: Sets the hatch pattern for filling the element.
		Foreground color: Sets the foreground color of the hatch brush to achieve foreground filling of the element.
		Foreground transparency: Sets the brush transparency to change the brush display effect (0% to 100%).
		Background color: Sets the background color of the hatch brush to achieve background filling of the element.
		Background transparency: Sets the brush opacity to change the brush display effect (0% to 100%).
	Texture brush	Fill style: Sets the texture pattern for filling the element.
		Fill mode: Sets the texture fill mode (including normal, tile, and stretch).
		Transparency: Sets the brush transparency to change the brush display effect (0% to 100%).
	Linear brush	Fill style: Sets the linear pattern for filling the element.
		Cycle count: Sets the number of linear brush fill repetitions.
		Rotation angle: Sets the fill rotation angle.
		Start color: Sets the starting color of the linear brush.

Item		Description
		Start color transparency: Sets the brush transparency to change the brush display effect (0% to 100%).
		End color: Sets the ending color of the linear brush.
		End color transparency: Sets the brush transparency to change the brush display effect (0% to 100%).
	Path brush	Gradient style: Sets the gradient style of the path brush (including circular and rectangular).
		Cycle count: Sets the number of path brush fill repetitions.
		Center size: Sets the center size of the path brush (0% to 100%).
		Start color: Sets the starting color of the linear brush.
		Start color transparency: Sets the brush transparency to change the brush display effect (0% to 100%).
		End color: Sets the ending color of the linear brush.
		End color transparency: Sets the brush transparency to change the brush display effect (0% to 100%).
Vertical fill direction		Specifies the fill direction: fill from top to bottom or from bottom to top.
Horizontal fill direction		Specifies the fill direction: fill from left to right or from right to left.
Vertical fill percentage		Sets the vertical fill percentage value of the element (0% to 100%).
Horizontal fill percentage		Sets the horizontal fill percentage value of the element (0% to 100%).

- Fill: For details, see Rectangle > Fill.
- Border: For details, see Rectangle > Border.
- Basic information: For details, see Line > Basic information.
- Animation: For details, see 13.3.8 Animation Configuration.
- Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.1.8 Circular Segment

The "circular segment" is a closed object that can be filled with a background color.



- **Circular segment creation**

  There are three ways to create a "circular segment". For details, see the line creation method.

- **Properties of circular segment**
  - Basic settings: For details, see Arc > Basic settings.
  - Style: For details, see Rectangle > Style.
  - Data:

Item		Description
Start angle		The angle is measured clockwise from the X-axis to the starting point of the arc (in degrees).
Sweep angle		The angle is measured clockwise from the startAngle parameter to the ending point of the arc (in degrees).
Percentage fill mode		There are two fill modes: absolute direction fill and relative direction fill. Absolute direction fill: Regardless of the orientation of the element, the fill direction is always aligned with the absolute vertical direction. Relative direction fill: When the shape of the element changes, the fill direction remains relative to the vertical orientation of the element.
Percentage fill brush	Solid brush	Color: Sets the solid brush color.
		Transparency: Sets the brush transparency to change the brush display effect (0% to 100%).
	Hatch brush	Fill style: Sets the hatch pattern for filling the element.
		Foreground color: Sets the foreground color of the hatch brush to achieve foreground filling of the element.
		Foreground transparency: Sets the brush transparency to change the brush display effect (0% to 100%).
		Background color: Sets the background color of the hatch brush to achieve background filling of the element.

Item		Description
		Background transparency: Sets the brush opacity to change the brush display effect (0% to 100%).
	Texture brush	Fill style: Sets the texture pattern for filling the element.
		Fill mode: Sets the texture fill mode (including normal, tile, and stretch).
		Transparency: Sets the brush transparency to change the brush display effect (0% to 100%).
	Linear brush	Fill style: Sets the linear pattern for filling the element.
		Cycle count: Sets the number of linear brush fill repetitions.
		Rotation angle: Sets the fill rotation angle.
		Start color: Sets the starting color of the linear brush.
		Start color transparency: Sets the brush transparency to change the brush display effect (0% to 100%).
		End color: Sets the ending color of the linear brush.
		End color transparency: Sets the brush transparency to change the brush display effect (0% to 100%).
	Path brush	Gradient style: Sets the gradient style of the path brush (including circular and rectangular).
		Cycle count: Sets the number of path brush fill repetitions.
		Center size: Sets the center size of the path brush (0% to 100%).
		Start color: Sets the starting color of the linear brush.
		Start color transparency: Sets the brush transparency to change the brush display effect (0% to 100%).
		End color: Sets the ending color of the linear brush.
		End color transparency: Sets the brush transparency to change the brush display effect (0% to 100%).
Vertical fill direction		Specifies the fill direction: fill from top to bottom or from bottom to top.
Horizontal fill direction		Specifies the fill direction: fill from left to right or from right to left.
Vertical fill percentage		Sets the vertical fill percentage value of the element (0% to 100%).

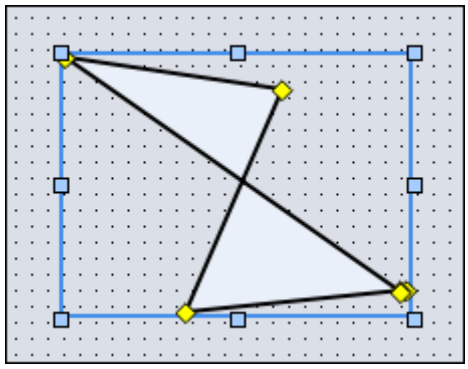
Item	Description
Horizontal fill percentage	Sets the horizontal fill percentage value of the element (0% to 100%).

- Fill: For details, see Rectangle > Fill.
- Border: For details, see Rectangle > Border.
- Basic information: For details, see Line > Basic information.
- Animation: For details, see 13.3.8 Animation Configuration.
- Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.1.9 Polygon

The "polygon" is a closed object that can be filled with a background color.

The corners of "polygon" are numbered sequentially in the order of creation. There is a yellow control point at each corner, which can be directly dragged to change their position.



- **Polygon creation**

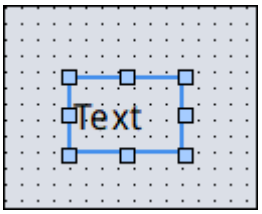
  There are two ways to create a "polygon":

  - Method 1: Select the object in the toolbox without releasing the mouse, drag it into the screen area, and then release the mouse to obtain a default-sized element.
  - Method 2:
    1. Click the target graphic element in the toolbox and move the mouse to the screen.
    2. Click the left button and drag to create a polygon.
    3. Right-click to generate the target element.

- **Polygon properties**
  - Basic settings: For details, see Polyline > Basic settings.
  - Style: For details, see Rectangle > Style.
  - Data: For details, see Rectangle > Percentage fill.
  - Fill: For details, see Rectangle > Fill.
  - Border properties: For details, see Rectangle > Border.
  - Properties of point set : For details, see Polyline > Point set.
  - Basic information: For details, see Line > Basic information.
  - Animation: For details, see 13.3.8 Animation Configuration.
  - Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.1.10 Text

The "text" is a closed object that allows one or more lines of text input, as well as font and color definition. You can also fill the text background and enter text for all configured languages.



- **Text creation**

  There are two methods to create a "text":

  - Method 1: Select the object from the toolbox, drag it into the screen area, and release the mouse button to create a default-sized element.
  - Method 2: Click the object in the toolbox, and then click the screen to create a default-sized element.

- **Text properties**
  - Basic settings:

Property	Description
Position	Sets the position coordinates (i.e., top-left coordinates) of the element on the screen in pixels: X: X-axis coordinate. Y: Y-axis position coordinate. The coordinate value must be smaller than the screen boundary coordinate value to prevent the elements from extending beyond the screen.
Dimension	The text size automatically adjusts according to the content. Manual setting is not supported.
Rotation	Sets the rotation angle of the element. Range: (-180 to 180].
Relative center	Sets the rotation center position of the object. This position is an offset relative to the object physical center.
Absolute center	Sets the rotation center position of the object. This position is an offset relative to the entire screen.

  - Text:

Item	Description
Text content	Sets the text content.
Font	Sets the text font. The default font is Droid Sans Fallback.
Font size	Sets the text size. The default size is 18.

Item	Description
Font style	Sets the text font style. Supports regular, bold, italic, and underline. The default is regular. Supports left alignment, center alignment, and right alignment. The default is left alignment.
Fit size	Specifies whether the text box size automatically adapts to the text content. When selected, the size adapts automatically. When unselected, the size is fixed and must be adjusted manually if needed.

- Border

Item	Description
Border width	Changes the border line thickness. Setting range: 0 to 10, where 0 means no border.
Border color	Changes the border line color.

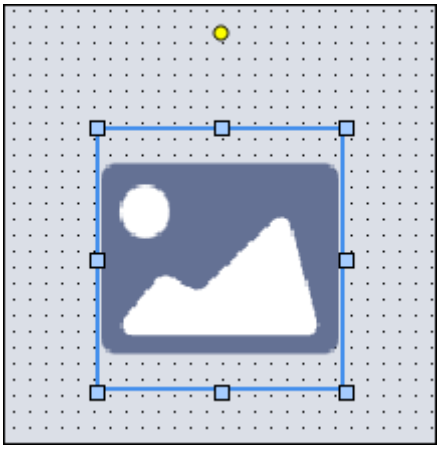
- Fill:

Item	Description
Text color	Sets the text color. The default is black.
Text transparency	Sets the text color transparency from 0% to 100%. The default is 0%.
Background color	Sets the background color. The default is gray.
Background color transparency	Sets the transparency of background color from 0% to 100%. The default is 0%.

- Basic information: For details, see Line > Basic information.
- Animation: For details, see 13.3.8 Animation Configuration.
- Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.1.11 Image

The "image" is a element used to display specified image information.



The image supports displaying the following image types: ico, png, bmp, jpg, jpeg, gif, svg, and tiff.

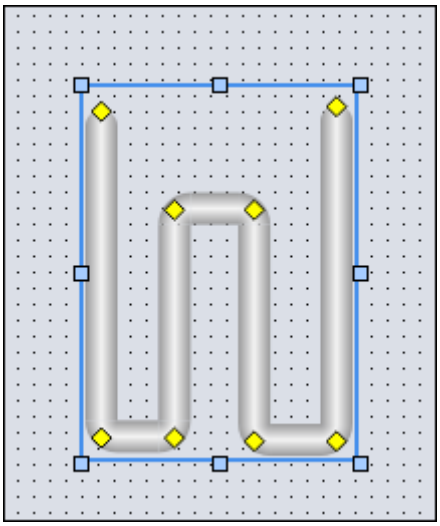
- **Image creation**
  There are three methods to create an image. For details, see the line creation method.
- **Image properties**
  - Basic settings: For details, see Arc > Basic settings.
  - Style:

Property	Description
Picture	Used to associate and clear specific image file. Image file sources from the system gallery, project gallery, and user gallery. After successful association in a 13.3.2 screen, the image is saved to the 13.3.6.4 project gallery. After successful association in a 13.4.3.2 symbol template, the image is saved to the 13.3.6.5 local gallery.
Display original size	Used to control the display mode of the image. When this option is checked, the image is displayed at its original aspect ratio; when unchecked, the image is stretched or scaled to fit the control's dimensions.

  - Basic information: For details, see Line > Basic information.
  - Animation: For details, see 13.3.8 Animation Configuration.
  - Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.1.12 Pipe

The "pipe" is a graphic with an outer wall and internal flow capability.



- **Pipe creation**

  There are two ways to create a "pipe":

  - Method 1: Select the object in the toolbox without releasing the mouse, drag it into the screen area, and then release the mouse to obtain a default-sized graphic element.
  - Method 2:
    1. Click the target element in the toolbox and move the mouse to the screen.

2. Click the left mouse button to set the starting position of the pipe, drag the mouse to the target position of the first pipe segment, and then click the left mouse button again to complete the first segment of the pipe.
3. Repeat the above steps until the final segment is drawn, and then right-click to generate the desired element.

- **Pipe properties**
  - Basic settings:

Property	Description
Position	Sets the position coordinates (i.e., top-left coordinates) of the element on the screen in pixels: X: X-axis coordinate. Y: Y-axis position coordinate. The coordinate value must be smaller than the screen boundary coordinate value to prevent the elements from extending beyond the screen.
Dimension	Sets the size of the element on the screen in pixels. When the pipe contains a set of three or more points, the width can be configured. Setting range: 16 to 10000
Rotation	Sets the rotation angle of the element. Range: (-180 to 180].
Relative center	Sets the rotation center position of the object. This position is an offset relative to the object physical center.
Absolute center	Sets the rotation center position of the object. This position is an offset relative to the entire screen.

  - Stroke:

Item	Description
Inner line	Sets the color and transparency (0% to 100%) of the internal flow line during pipe operation.
Inner line width	Sets the width of the internal flow line, including Thiner, Thin, Mid, Wide, and Wider, with Wide as the default.
Outer line	Sets the color and transparency (0% to 100%) of the pipe outer line.
External line width	Sets the external line width of the pipe.

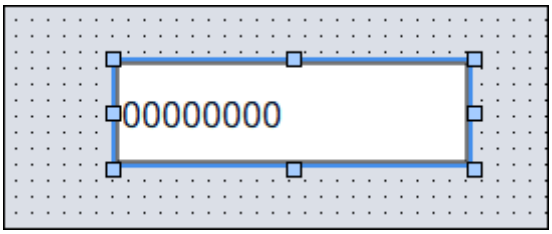
  - Point set: For details, see Polyline > Point set.
  - Basic information: For details, see Line > Basic information.
  - Animation: For details, see 13.3.8 Animation Configuration.
  - Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.2 Control and Display

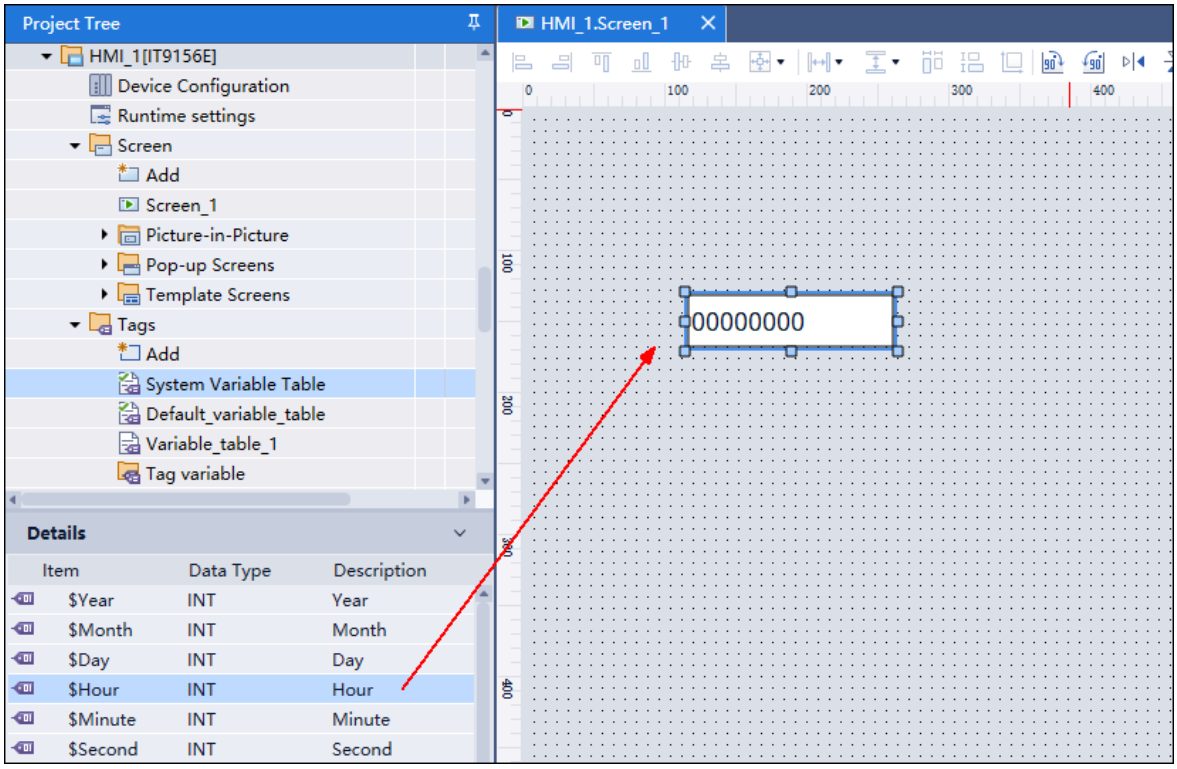
- 13.3.5.2.1 Numeric I/O Field
- 13.3.5.2.2 Character I/O Field
- 13.3.5.2.3 Date-Time Field
- 13.3.5.2.4 Button
- 13.3.5.2.5 Bit Button
- 13.3.5.2.6 Text Button
- 13.3.5.2.7 Bit Indicator
- 13.3.5.2.8 Text Indicator
- 13.3.5.2.9 Text List
- 13.3.5.2.10 Combobox
- 13.3.5.2.11 List Box
- 13.3.5.2.12 Text Switch
- 13.3.5.2.13 Radio Button
- 13.3.5.2.14 Checkbox
- 13.3.5.2.15 Table
- 13.3.5.2.16 Picture-in-Picture View

13.3.5.2.1 Numeric I/O Field

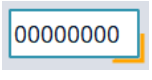
The "numeric I/O field" is used to modify/display variable values.



- **Numeric I/O field creation**
  - There are three methods to create a "numeric I/O field". For details, see the line creation method.
  - Direct creation with variable: Click the variable table node in the project tree, click a variable in the detail view, and drag it onto the screen. The corresponding numeric I/O field is created.
    The data type of variable does not support DATETIME, BOOL, or STRING.
    For example, dragging the system variable $Hour onto the screen generates the corresponding numeric I/O field as shown in the figure below.



- **Properties** of **numeric I/O field**
  - Basic settings:

Item	Description
Position	Sets the position coordinates (i.e., top-left coordinates) of the element on the screen in pixels: X: X-axis coordinate. Y: Y-axis position coordinate. The coordinate value must be smaller than the screen boundary coordinate value to prevent the elements from extending beyond the screen.
Dimension	Sets the size of the element on the screen in pixels. Setting range: 1 to 10000
Corner radius	Specifies the degree of curvature or smoothness at the corners of the numeric I/O field. The maximum corner radius is half of the shortest side length of the numeric I/O field. For example, for the control in the figure with dimensions 110 x 46, the shortest side length is 46, so the maximum corner radius is 23.  The two sides of the orange corner are the radius of the rounded corner, and the vertex of the corner is the center point.

  - Interface properties:

Property	Description
Mode	Value range: Input, Input/Output, Output; Default value: Input/Output. Special description: 1. Input mode and Input/Output mode:Any attempt to input illegal values or values outside the range, as well as setting such values via script, will be intercepted and the value change will not be executed. 2. Read-only mode: Data cannot be input on the screen. Script function can modify the value. 3. Input mode: Project unit is cleared and hidden. Positive/negative sign is false and hidden.
LCD display	Effective only in "Output" mode and is disabled by default. The LCD display style is . The LCD display does not support displaying in encrypted form.
Same read/ write variable	This option is selected by default. When selecteded, the read and write variables are the same, and the write variable is grayed out. When unselected, the read and write variables can be set separately.
Read/Write variable	Sets the read/ write variables. When the PLC associated with the variable is disconnected, the value remains the previous value, input is disabled, the color changes, and an icon indicates the status. The following figure shows the normal and disconnected states:  In decimal format, if a float exceeds 7 significant digits or a double exceeds 16 significant digits, the float is rounded to 7 significant digits and the double to 16 significant digits. If the set number of integer digits is less than the current number of integer digits in the variable, the variable displays "###".
Project unit	Manual input, up to 64 characters.

Display format

Format range: Hexadecimal / Decimal / Binary / BCD. Default value: Decimal.

BCD: 8421 BCD code is used. Data processed in BCD format does not include a sign bit, meaning that negative numbers cannot be represented in BCD. If it is a negative number, display "###".

Data Type	Display Format
USINT	Decimal, Hexadecimal, Binary, BCD
SINT	Decimal, Hexadecimal, Binary, BCD
UINT	Decimal, Hexadecimal, Binary, BCD
INT	Decimal, Hexadecimal, Binary, BCD
UDINT	Decimal, Hexadecimal, Binary, BCD
DINT	Decimal, Hexadecimal, Binary, BCD
ULINT	Decimal, Hexadecimal, Binary, BCD
LINT	Decimal, Hexadecimal, Binary, BCD
REAL	Decimal
LREAL	Decimal

- The display format takes effect for both input and output of numeric I/O fields:

  During offline configuration, when in input/output or output mode, the display format is validated against the data type of the read variable. If invalid, the user is prompted and a compilation error is reported. In input mode, the display format is validated against the data type of the write variable. If invalid, the user is prompted and a compilation error is reported.

  During runtime, data can only be input and displayed in this format.

- The display format affects the range of values that can be displayed and input in the numeric I/O control:

  If the integer digit count is set to 4 and the format is decimal, the range is -9999 to 9999;

  If the number of integers is set to 4 and the format is BCD, the range is 0 to 7999. (For UINT type, the upper limit is 9999.)

  If the number of integers is set to 4 and the format is hexadecimal, the range is 0 to FFFF.

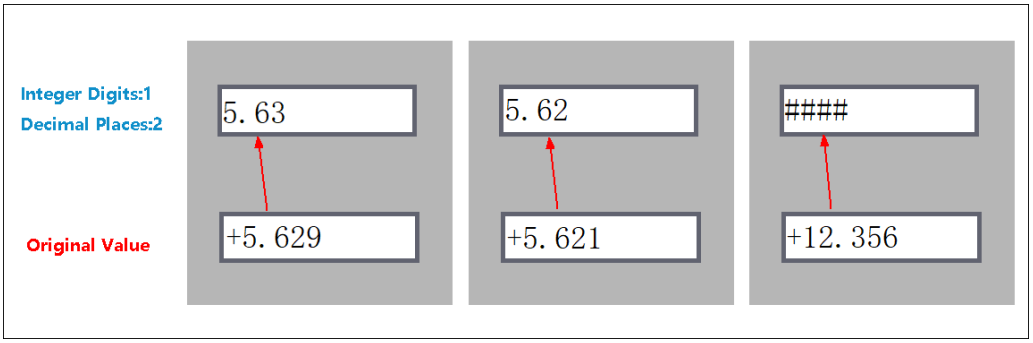
  If the number of integers is set to 4 and the format is binary, the range is 0 to 1111.

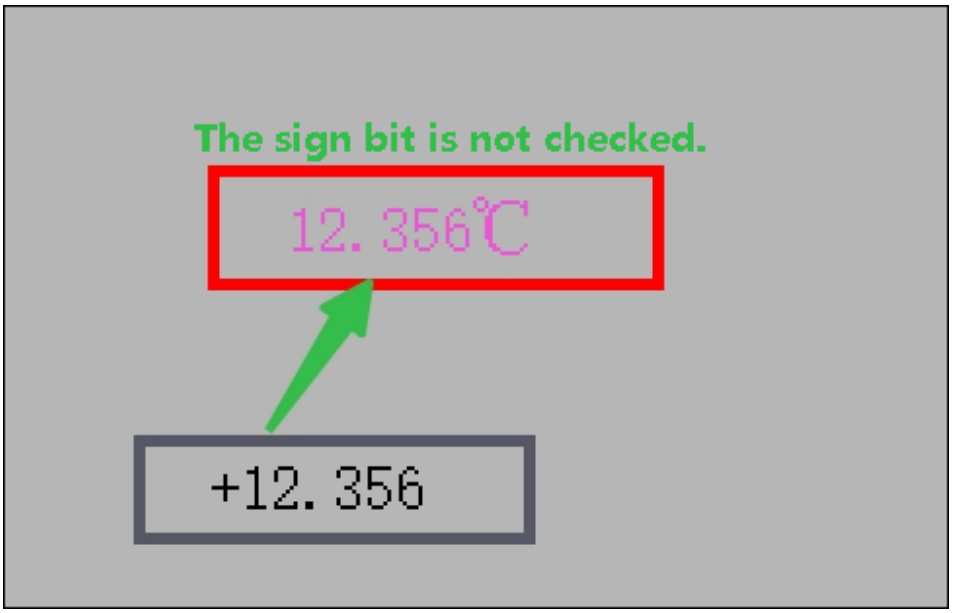
- The display format affects the display format and range of the variable upper and lower limits:

  For INT type variables, if upper and lower limits are not configured, the default limits are the data type limits: -32768 to 32767.

  If the format is decimal, the limits are -32768 to 32767.

Property	Description
	When the format is BCD, the limits are 0 to 7999. (For UINT type, the upper limit is 9999.) If there is no corresponding BCD value for the limits, the nearest valid value below the lower limit is selected, and the nearest valid value above the upper limit is selected. If the value is negative, it is treated as 0. When the format is binary, the limits are 0 to 1111111111111111. When the format is hexadecimal, the limits are 0 to FFFFF. ➢When the display format is set to BCD, BCD data must be converted to decimal before writing to the variable, and the decimal variable value must be converted to BCD for display. For example, for an INT type variable, if you input 58, the corresponding BCD code is 0101 1000, which converts to decimal 88, so the variable value should be written as 88. Similarly, when the variable value is 88, it should be displayed as 58 in BCD format.
Number of integers	Sets the number of integers. The range is 1 to 32 and the default value is 8. The number of integers must match the display format and applies to both input and output of the numeric I/O field. During runtime, data can only be input and displayed in this format. After configuring the display format, number of integers, and number of decimal places, the corresponding effect is synchronized in the IDE configuration interface: If the display format is decimal or BCD, the number of integers is 2, and the number of decimal places is 2, the IDE interface displays 00.00. If the display format is hexadecimal and the number of integers is 2, the IDE interface displays FF. If the display format is binary and the number of integers is 3, the IDE interface displays 111.

Property	Description
Decimal points	Sets the number of decimal places. The range is 0 to 16 and the default value is 0. When the display format is Binary, Hexadecimal, or BCD, the number of decimal places is reset to the default value 0, grayed out, and disabled for editing. Decimal places support rounding. That is, when the actual number of decimal places exceeds the configured number, the decimal part is rounded, as shown in the figure below:  ➢After configuring integer and decimal places for a variable, when inputting data through a numeric IO field, the input data must adhere to the user-configured limits for the lengths of integer places and decimal places. For example, for a UINT-type variable configured with 2 integers and 2 decimal places, a maximum of 2 integers and 2 decimal digits can be input. ➢After configuring the integer and decimal places for a variable, the display range of the variable is affected. For example, for a UINT-type variable configured with 2 integers and 2 decimal places, the display range of the I/O field is 0.00 to 99.99. When the variable value is 10001, it should be displayed as 100.01, which exceeds the display range of I/O field. "###" is displayed instead. ➢For non-floating-point variables, configuring decimal places is equivalent to scaling the variable, which affects the displayed upper and lower limits: For INT type variables, if upper and lower limits are not configured, the default limits are the data type limits: -32768 to 32767. When the format is decimal and the number of decimal places is 1, the limits are displayed as -3276.8 to 3276.7. When the format is BCD and the number of decimal places is 1, the limits are displayed as 0.0 to 999.9. The binary format is not applicable. The hexadecimal format is not applicable. ➢For floating-point variables, configuring decimal places does not scale the variable, but only affects the number of digits that can be input after the decimal point.

Property	Description
Sign symbol	Specifies whether to display the sign (positive/negative). This option is unselected by default (sign not displayed) and is configurable when the format is decimal. The sign symbol can be selected for signed data types. For other data types, the sign option is unselected and grayed out. ➢When the sign symbol is selected, the sign symbol is displayed. ➢When the sign symbol is not selected: 1. When the actual value is negative, a minus sign (-) is displayed on the screen. 2. When the actual value is positive, no plus sign (+) is displayed on the screen. 3. Negative values can be input because signed data types include negative values in their range, such as integers. 
Leading zero	When selected, leading zeros are automatically added if the display length is insufficient. Editing is only available in decimal/BCD format. This option is not selected by default. When selected, leading zeros are added to the integer part and the decimal part respectively. For example, with 5 integer digits and 3 decimal digits, 12.3 is displayed as 00012.300.
Cipher display	This option cannot be selected simultaneously with "LCD display". Cipher display is only used for numeric values. Project units are displayed in plain text. For display effect, see 13.15.1 Real-Time Variable Monitoring. Cipher text uses * (asterisk) to represent each character. For example, 26.5 is displayed as "****" in cipher text.

◦ Text:

Property	Description
Font	Sets the text font. The default font is Droid Sans Fallback.
Font size	Sets the text size. The default size is 18.

Property	Description
Font style	Sets the text font style. Supports regular, bold, italic, and underline. The default is regular. Supports left alignment, center alignment, and right alignment. The default is left alignment.

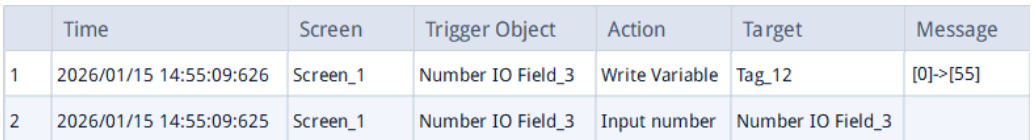
- Border:

Property	Description
Border width	The value range is 0 to 10 and the default is 1. The value can be manually input, or changed by clicking the up/down increment/decrement buttons.
Border color	Sets the border color. The default is black #000000.

- Fill:

Property	Description
Text color	Sets the text color and transparency of the control, defaulting to black with 0%. A higher transparency value indicates a more transparent text.
Background color	Sets the background color and transparency of the control, defaulting to white with 0%. A higher transparency value indicates a more transparent background.

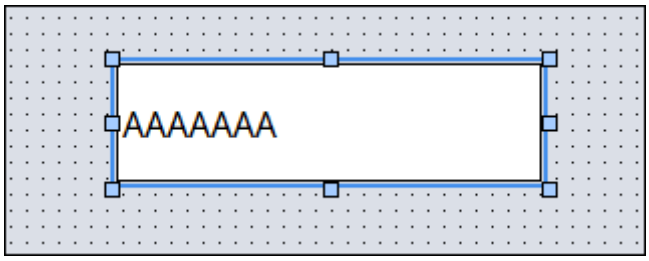
- Operation record:

Property	Description
Record	When the current control performs a write operation, this action is not recorded by default. If recorded, two records will be generated: one for the input value action and another for the variable write action. For example, the two logs generated when "Numeric I/O field_3" writes "55" are shown in the figure below.  Time / Screen / Trigger Object / Action / Target / Message 1 / 2026/01/15 14:55:09:626 / Screen_1 / Number IO Field_3 / Write Variable / Tag_12 / [0]->[55] 2 / 2026/01/15 14:55:09:625 / Screen_1 / Number IO Field_3 / Input number / Number IO Field_3 /
Action description	Description can be up to 256 characters, which will be displayed in the 13.3.5.4.8 Operation Records.

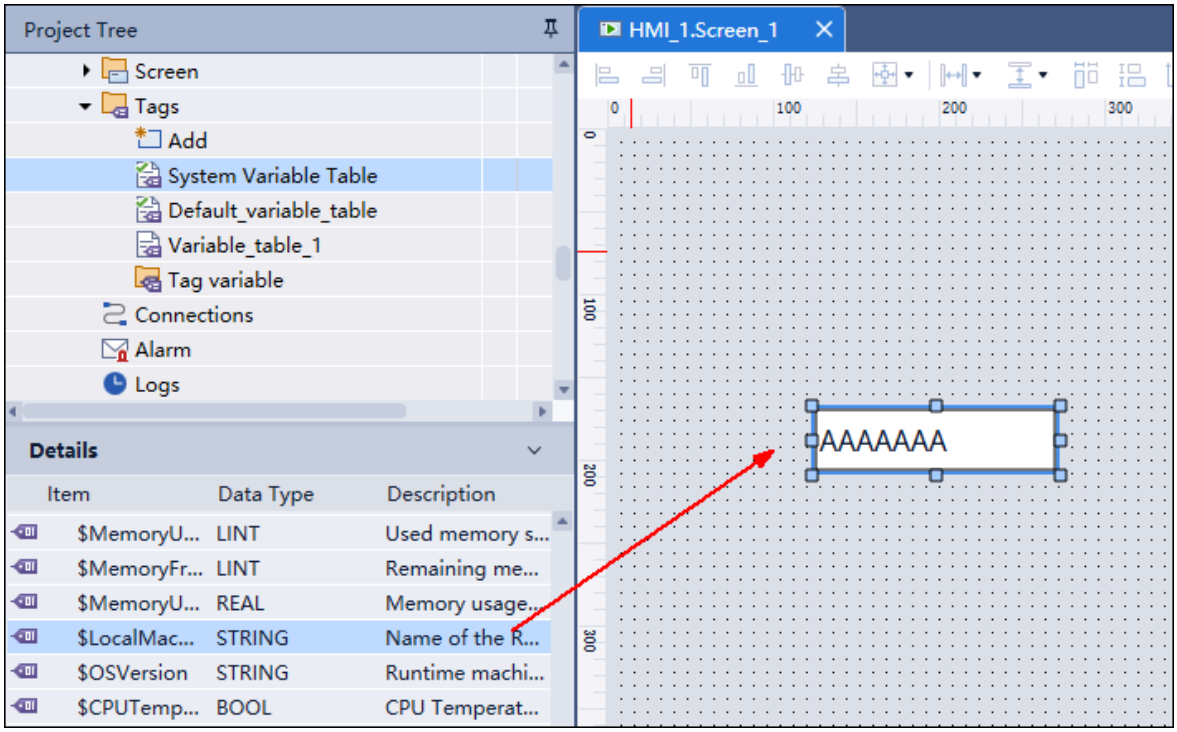
- Basic information: For details, see Line > Basic information.
- Animation: For details, see 13.3.8 Animation Configuration.
- Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.2.2 Character I/O Field

The "character I/O field" reads/displays string variables.



- **Character I/O field creation**
  - There are three methods to create a "numeric I/O field". For details, see the line creation method.
  - Direct creation with variable: Click the variable table node in the project tree, click a STRING-type variable in the detail view, and drag it onto the screen. The corresponding character I/O field is created.
    For example, dragging the system variable $LocalMachineName onto the screen generates the corresponding character I/O field as shown in the figure below.



- Properties of **character I/O field**
  - For basic settings, see Numeric I/O Field > Basic settings.
  - Interface:

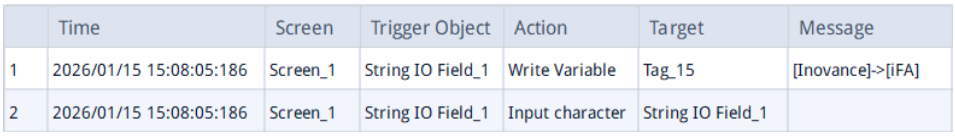
Item	Description
Mode	Value range: Input, Input/Output, Output. Default value: Input/Output. Note 1 (Input mode and Input/Output mode): Any attempt to input illegal values, set illegal values via script, or set values outside the range will be intercepted, and the value change will not be executed. Note 2 (Read-only mode): Data cannot be input on the screen. Script function can modify the value.

Item	Description
Same read/write variable	This option is selected by default. When selecteded, the read and write variables are the same, and the write variable is grayed out. When unselected, the read and write variables can be set separately.
Read variable	Sets the read variable.
Write variable	Sets the write variable.
Cipher display	Cipher text uses * (asterisk) to represent each character. For example, "Overtemperature" is displayed as "****" in cipher text. For display effect, see 13.15.1 Real-Time Variable Value Monitoring.

- Text: For details, see Numeric I/O Field > Text.
- Border:

Item	Description
Border width	The value range is 0 to 10 and the default is 1. The value can be manually input, or changed by clicking the up/down increment/decrement buttons.
Border color	Sets the border color. The default is black.

- Fill: For details, see Numeric I/O Field > Fill.
- Operation record:

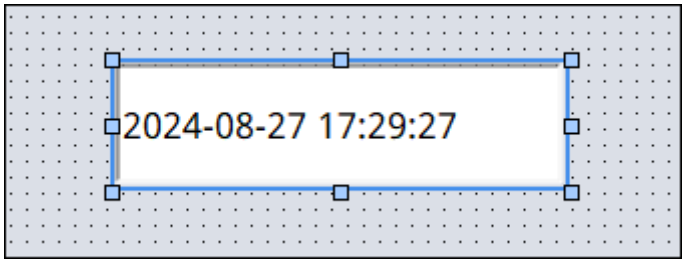
Item	Description
Record	When the current control performs a write operation, this action is not recorded by default. If recorded, two records will be generated: one for the input character action and another for the variable write action. For example, the two logs generated when "Character I/O field_1" writes "iFA" are shown in the figure below. 
Action description	Description can be up to 256 characters, which will be displayed in the 13.3.5.4.8 Operation Records.

	Time	Screen	Trigger Object	Action	Target	Message
1	2026/01/15 15:08:05:186	Screen_1	String IO Field_1	Write Variable	Tag_15	[Inovance]->[iFA]
2	2026/01/15 15:08:05:186	Screen_1	String IO Field_1	Input character	String IO Field_1	

- Basic information: For details, see Line > Basic information.
- Animation: For details, see 13.3.8 Animation Configuration.
- Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.2.3 Date-Time Field

The "date-time field" is used to display time.



- **Date-time field creation**

  There are three methods to create a "date-time field". For details, see the line creation method.

- **Properties of date-time field**
  - Basic settings: For details, see Numeric I/O Field > Basic settings.
  - Text: For details, see Numeric I/O Field > Text.
  - Interface:

Item	Description
Mode	Mode support: Output, Input/Output. When the **time source** is set to **variables**, both modes are supported. The default is Input/Output. When the **time source** is set to **system time**, only output mode is supported.
Time source	You can choose to display system time or use variables. The default is to display system time.
Variable	When selected, the time of the configured variable is displayed. It is empty by default. Supported data type is DATETIME.
Format	Time format is composed of date and time values.

  - Border:

Item	Description
Border width	The value range is 0 to 10 and the default is 1. The value can be manually input, or changed by clicking the up/down increment/decrement buttons.
Border color	Sets the border color. The default is black #000000.

  - Fill:

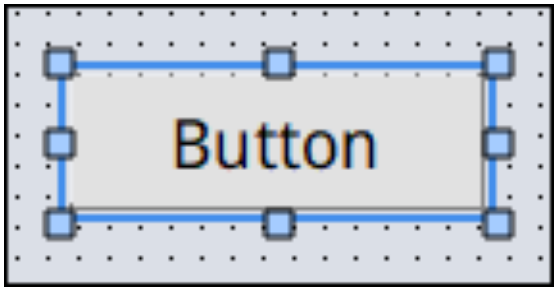
Item	Description
Text color	Sets the text color and transparency of the control, defaulting to black with 0%. A higher transparency value indicates a more transparent text.

Item	Description
Background color	Sets the background color and transparency of the control, defaulting to white with 0%. A higher transparency value indicates a more transparent background.

- Basic information: For details, see Line > Basic information.
- Animation: For details, see 13.3.8 Animation Configuration.
- Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.2.4 Button

The "button" is used to execute a command at run time. Click the button to execute the command.

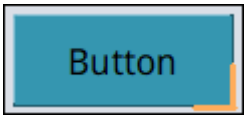


- **Button creation**

There are three methods to create a "button". For details, see the line creation method.

- **Button properties**
  - Basic settings:

Item	Description
Position	Sets the position coordinates (i.e., top-left coordinates) of the element on the screen in pixels: X: X-axis coordinate. Y: Y-axis position coordinate. The coordinate value must be smaller than the screen boundary coordinate value to prevent the elements from extending beyond the screen.
Dimension	Sets the size of the element on the screen in pixels. Setting range: 1 to 10000

Item	Description
Corner radius	Specifies the degree of curvature or smoothness at the corners of the button (flat style). The maximum corner radius is half of the shortest side length of the button. For example, for the control in the figure with dimensions 110 x 46, the shortest side length is 46, so the maximum corner radius is 23.  The two sides of the orange corner are the radius of the rounded corner, and the vertex of the corner is the center point.

◦ Text:

Item	Description
Text content	Sets the text displayed on the button.
Display mode	Display modes include single line and multi-line. The default is single line display.
Font	Sets the text font. The default font is Droid Sans Fallback.
Font size	Sets the text size. The default size is 18.
Font style	Sets the text font style. Supports regular, bold, italic, and underline. The default is regular.

◦ Fill:

Fill style: Sets the fill style of the default background color for buttons.

◦ Style:

Property	Description
Icon	Displays the image on the button.
Icon size	Sets the display size of the image.
Button style	Normal: 3D display. Flat: Flat display. Flat-style buttons can adjust the corner radius through the "Corner radius" option in "Basic setting".
Icon layout	Image layout on the button: Icon Only: Displays the image with no text. Left/Right/Top/Bottom: Position of the image around the text.

◦ Border:

Item	Description
Border width	Applies to flat-style buttons. The range of border width is 0 to10, where 0 indicates no border.
Border color	Sets the border line color and transparency.
Default text color	Sets the default color and transparency of the button text.
Default background color	Sets the default background color and transparency of the button.
Pressed background color	Sets the background color and transparency when the button is pressed.
Disabled background color	Sets the background color and transparency when the button is disabled.

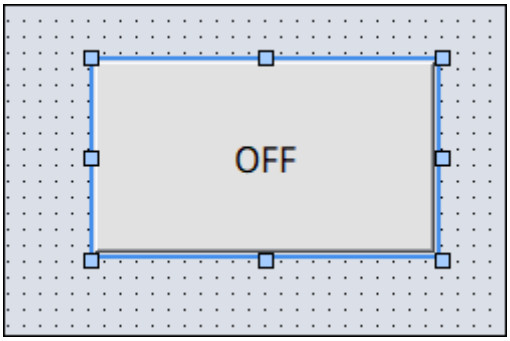
- Operation record:

Item	Description
Record	When the current control performs a click operation, this action is not recorded by default. If recorded, a record will be generated. For example, the following figure shows the log generated by clicking "Button_2". | | Time | Screen | Trigger Obje | Action | Target | | 1 | 2026/01/15 15:14:13:597 | Screen_1 | Button_2 | Click the Button , open screen_2 | Button_2 |
Action description	Description can be up to 256 characters, which will be displayed in the 13.3.5.4.8 Operation Records.

- Basic information: For details, see Line > Basic information.
- Animation: For details, see 13.3.8 Animation Configuration.
- Event: For details, see 13.3.7.14 Adding an Event.

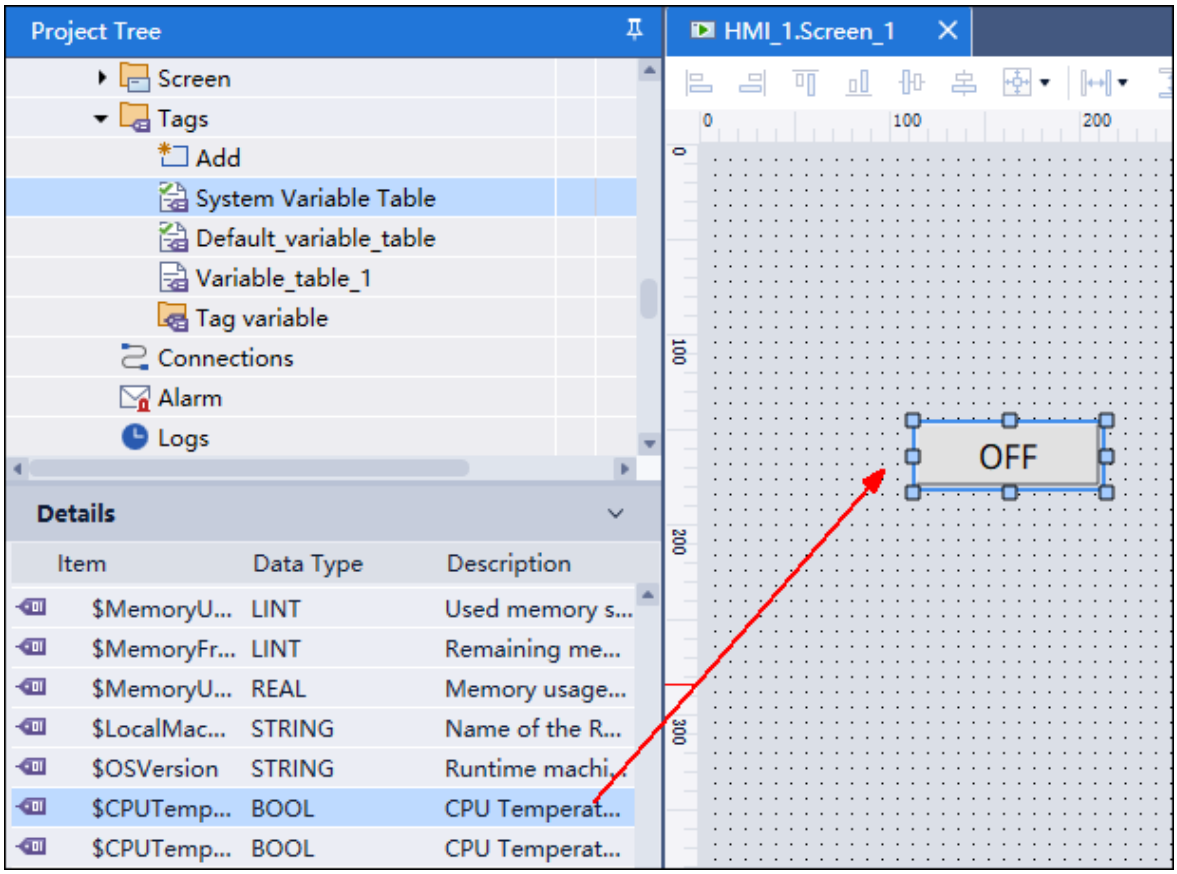
13.3.5.2.5 Bit Button

The "bit button" sets, modifies and displays the value of a variable at a specified bit position.



- **Bit button creation**
  - There are three methods to create a "bit button". For details, see the line creation method.

◦ Direct creation with variable: Click the variable table node in the project tree, click a BOOL-type variable in the detail view, and drag it onto the screen. The corresponding bit button is created. For example, dragging the system variable $CPUTemperatureHigh onto the screen generates the corresponding bit button as shown in the figure below.



- **Properties of bit button**

◦ For basic settings, see Button > Basic settings.

◦ Data:

Item	Description
Same read/ write variable	Specifies whether the read value and write value of the bit button are associated with the same variable. This option is selected by default.
Invert read output	Specifies whether the bit button state is inverted relative to the associated variable value. This option is selected by default.
Read variable	Selects the read variable to associate with the control.
Write variable	Selects the write variable to associate with the control.
Write mode	Set: Set the bound variable to 1 when the button is released. Reset: Set the bound variable to 0 when the button is released. Invert: Write the inverted value. Press ON: Set to 1 when pressed; set to 0 when released. Press OFF: Set to 0 when pressed; set to 1 when released.

Item	Description
Read bit number	When a variable of integer data type is selected as the read variable, the bit number is enabled. The default value is 0. The value range changes according to the variable type. For example, when the associated variable is INT, the range is 0 to 15; when the variable type is LINT, the range is 0 to 63.
Write bit number	When a variable of integer data type is selected as the write variable, the bit number is enabled. The default value is 0. The value range changes according to the variable type. For example, when the associated variable is INT, the range is 0 to 15; when the variable type is LINT, the range is 0 to 63. When the read/write variables are the same, this option is grayed out and cannot be edited. The write bit number is automatically synchronized with the read bit number.
Click animation	The animation is scaled when you click this button in runtime. This option is selected by default.
Minimum press duration	Specifies the minimum press duration (in ms) required to trigger the associated event. Used to prevent accidental operation on the touch screen. Value range: 0 to 60,000.
Auto repeat	When the button is pressed, the event can be repeatedly triggered. This option is not selected by default.

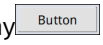
◦ Text:

Item	Description
Text position	The position of text display on the button. Position options: top-left, top-center, top-right, middle-left, center, middle-right, bottom-left, bottom-center, bottom-right, text left-aligned, text right-aligned, text top-aligned, and text bottom-aligned.
Font	Sets the text font. The default font is Droid Sans Fallback.
Font size	Sets the text size. The default size is 18.
Font style	Sets the text font style. Supports regular, bold, italic, and underline. The default is regular. Supports left alignment, center alignment, and right alignment. The default is left alignment.

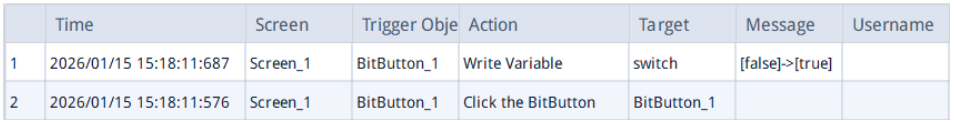
◦ Bit state settings:

Item	Description
State [0]	The state displayed on the button when the specified bit value of the associated variable is 0. Text: The text content on the button when the associated variable value is 0. Flashing: The flashing state when the associated variable value is 0. Flashing options: No, Text Color Alternation, and Display/Hide. The default is No. Style: Effect Preview Text color: The color of the text on the button when the associated variable value is 0. Background color: The background color of the button when the associated variable value is 0. Background image: The image displayed on the button when the associated variable value is 0.
State[1]	The state displayed on the button when the specified bit value of the associated variable is 1. Text: The text content on the button when the associated variable value is 1. Flashing: The flashing state when the associated variable value is 1. Flashing options: No, Text Color Alternation, and Display/Hide. The default is No. Style: effect preview Text color: The color of the text on the button when the associated variable value is 1. Background color: The background color of the button when the associated variable value is 1. Background image: The image displayed on the button when the associated variable value is 1.

- Border: For details, see Button > Border.
- Fill:

Item	Description
Fill style	Sets the background fill style of the button.
Button style	Normal: 3D display  Flat: Flat display  Flat-style buttons can adjust the corner radius through the "Corner radius" option in "Basic setting".

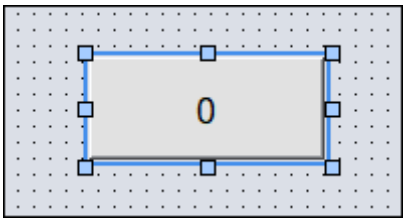
- Operation record:

Item	Description
Record	When the current control performs a write operation, this action is not recorded by default. If recorded, two records will be generated: one for the click action and another for the variable write action. For example, the two logs shown in the figure below are generated when the state of "Bit button_1" switches to "ON".  | | Time | Screen | Trigger Obje | Action | Target | Message | Username | | 1 | 2026/01/15 15:18:11:687 | Screen_1 | BitButton_1 | Write Variable | switch | [false]->[true] | | | 2 | 2026/01/15 15:18:11:576 | Screen_1 | BitButton_1 | Click the BitButton | BitButton_1 | | |
Action description	Description can be up to 256 characters, which will be displayed in the 13.3.5.4.8 Operation Records.

- Basic information: For details, see Line > Basic information.
- Animation: For details, see 13.3.8 Animation Configuration.
- Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.2.6 Text Button

The "text button" sets, modifies, and displays an integer variable value.



- **Text button creation**

There are three methods to create a "text button". For details, see the line creation method.

- **Properties of word button**
  - For basic settings, see Button > Basic settings.
  - Data:

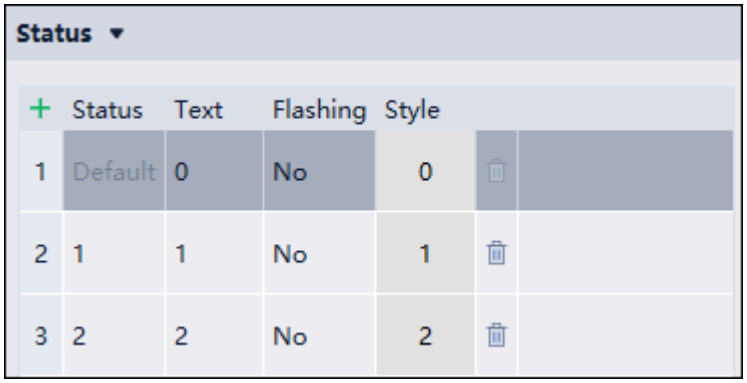
Item	Description
Same read/write variable	Specifies whether the read value and write value of the word button are associated with the same variable. This option is selected by default.
Read variable	Selects the read variable to associate with the control. Only integer variables are supported.
Write variable	Selects the write variable to associate with the control. Only integer variables are supported.

Item	Description
Mode	1. Increment: The write variable increases by a target value (variable or constant) each time. 2. Decrement: The write variable decreases by a target value (variable or constant) each time. 3. Cycle increment: The write variable increases by the next value in the ordered sequence of states. If the write variable value is outside the state range, the last state value is used. When the current value is the last one, it increments to the first value. 4. Cycle decrement: The write variable decreases by the next value in the ordered sequence of states. If the write variable value is outside the state range, the last state value is used. When the current value is the first one, it decrements to the last value. 5. State increment: The write variable increases by the next value in the ordered sequence of states. If the write variable value is outside the state range, the last state value is used. Increment stops when the current value is the last value. 6. State decrement: The write variable decreases by the next value in the ordered sequence of states. If the write variable value is outside the state range, the last state value is used. Decrement stops when the current value is the first value. 7. Custom: After clicking the button, set the write variable value to a custom target value, which can be a constant or variable. The default mode is increment. When same read/write variable is not selected, only increment, decrement and custom mode are supported.
Constant	Sets the target values for increment, decrement, and custom, which can be any real number. The default value is 1.
Variable	Sets the target variable for increment, decrement, and custom.
Click animation	The animation is scaled when you click this button in runtime. This option is selected by default.
Minimum press duration	Specifies the minimum press duration (in ms) required to trigger the associated event. Used to prevent accidental operation on the touch screen. Value range: 0 to 60,000.
Auto repeat	When the button is pressed, the event can be repeatedly triggered. This option is not selected by default.

- Fill: For details, see Bit Button > Fill.
- Border: For details, see Button > Border.
- Operation record:

Item	Description
Record	When the current control performs a write operation, this action is not recorded by default. If recorded, two records will be generated: one for the click action and another for the variable write action. For example, the two logs shown in the figure below are generated when the state of "Text button_2" switches to "3". | | Time | Screen | Trigger Object | Action | Target | Message | | 1 | 2026/01/15 15:23:43:983 | Screen_1 | WordButton_2 | Write Variable | Tag_3 | [2]->[3] | | 2 | 2026/01/15 15:23:43:866 | Screen_1 | WordButton_2 | Click the WordButton | WordButton_2 | |
Action description	Description can be up to 256 characters, which will be displayed in the 13.3.5.4.8 Operation Records.

- Text: For details, see Bit Button > Text.
- Word state: Multiple word states can be added to the word state property. Click the "Add" button to add a state. Each text indircator control has a default state that cannot be deleted. The default state value is "Default", and the default text is 0, with a maximum of 256 characters.



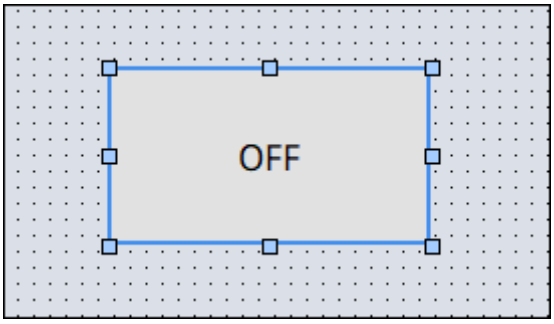
Property		Description
Status		Sets the state value, supporting single values or value range. When the variable value matches the state value, or falls within the value range, the button displays the text content.
Text		Sets the text content to be displayed when the state value is met.
Flashing		Sets the flashing method when the state value is met. Flashing methods: ▪ No: No flashing. ▪ Text Color Alternation: The text content flashes, while the button does not. ▪ Display/Hide: The entire button flashes.
Style	Text color	Text color for state value display
	Background color	Background color for state value display

Property		Description
	Background image	Background image for state value display

- For basic information, see Line > Basic information.
- For animation, see 13.3.8 Animation Configuration.
- Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.2.7 Bit Indicator

The "bit indicator" displays the value of a variable at a specified bit position.



- **Bit indicator creation**

  There are three methods to create a "bit indicator". For details, see the line creation method.

- **Properties of bit status display**
  - For basic settings, see Button > Basic settings.
  - Data:

Item	Description
Invert read output	Specifies whether the bit indicator is inverted relative to the read value of the associated variable. This option is selected by default.
Read variable	Selects the read variable to associate with the control.
Bit number	When the read variable is of integer data type, bit enable is active by default, with a default value of 0. The value range changes according to the variable type. For example, when the associated variable is INT, the range is 0 to 15; when the variable type is LINT, the range is 0 to 63.

  - Text: For details, see Bit Button > Text.
  - Bit state settings: For details, see Bit Button > Bit state.
  - Border:

Item	Description
Border width	The range of border width is 0 to10, where 0 indicates no border.
Border color	Sets the border line color and transparency.

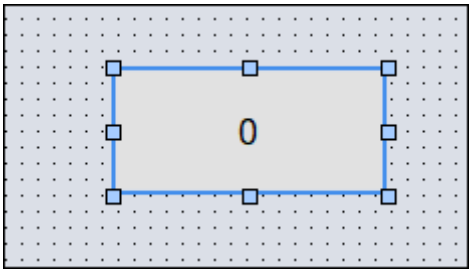
  - Fill properties:

Item	Description
Fill style	Sets the background fill style of the button.

- Basic information: For details, see Line > Basic information.
- Animation: For details, see 13.3.8 Animation Configuration.

13.3.5.2.8 Text Indicator

The "text indicator" is used to display variable values.



- **Text indicator creation**

  There are three methods to create a "text indicator". For details, see the line creation method.

- **Properties of text indicator**
  - Basic Settings: For details, see Button > Basic settings.
  - Data: Read variable is used to specify the variable associated with the text indicator.
  - Text: For details, see Bit Button > Text.
  - Text indicator settings: For details, see Text Button > Text state.
  - Border:

Item	Description
Border width	The range of border width is 0 to10, where 0 indicates no border.
Border color	Sets the border line color and transparency.

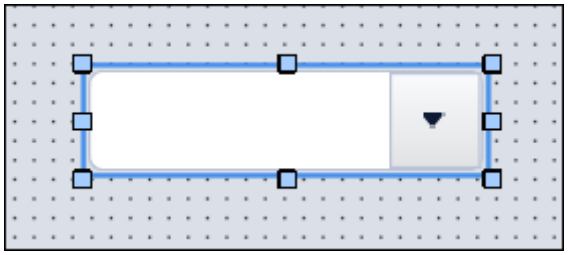
  - Fill:

Item	Description
Fill style	Sets the background fill style of the button.

  - Basic information: For details, see Line > Basic information.
  - Animation: For details, see 13.3.8 Animation Configuration.

13.3.5.2.9 Text List

The "text list" is used to edit its associated variables based on its setpoints, and to display the corresponding text according to the current values of the associated variables.



- **Text list creation**

  There are three methods to create a "text list". For details, see the line creation method.

- **Properties of text list**

  - For basic settings, see Numeric I/O Field > Basic settings.
  - Options:

Item	Description
Text list	Selects the text list to bind. Text lists with empty content are not supported.
Mode	Input/Output mode: control style  Output mode: control style  The default value is "Input/Output" mode.
Visible entries	The number of texts available for user selection. It is displayed when "Mode" is set to "Input/Output". The range is 1 to 100 with a default value of 3.
Variable	The correspondence between the variable values and entry values (or Bit No.) is: ▪ Integer variable: If the "Selection" value of the text is "Bit (0, n)": Bit n corresponds to 2^n. For example, bit 0 corresponds to 2^0, which is 1. If the "Selection" value of the text is "Value/Range": Item values corresponds one-to-one with the variable values. ▪ Boolean type: If the "Selection" value of the text is "Bit (0,1)": Bit 0 corresponds to the variable value False, and bit 1 corresponds to the variable value True.

  - Text: For details, see Numeric I/O Field > Text.
  - Stroke:

Item	Description
Border	Sets whether the text list displays a border. This option checked by default.
Border width	Sets the border width range from 0 to 10. The default is 1.
Border color	Sets the border color of the text list. The default is #B9C2D5.

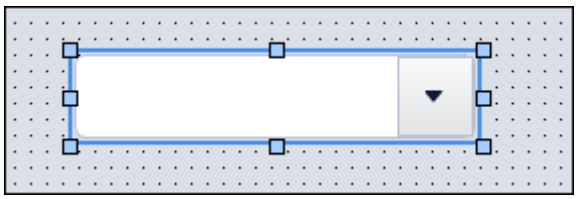
  - Fill:

Item	Description
Text color	Specifies the text color. The default is #0C1A34.
Background color	Specifies the color value and transparency (0% to 100%). The default color value is #FFFFFF and transparency is 0%.

- Basic information: For details, see Line > Basic information.
- Animation: For details, see 13.3.8 Animation Configuration.
- Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.2.10 Combobox

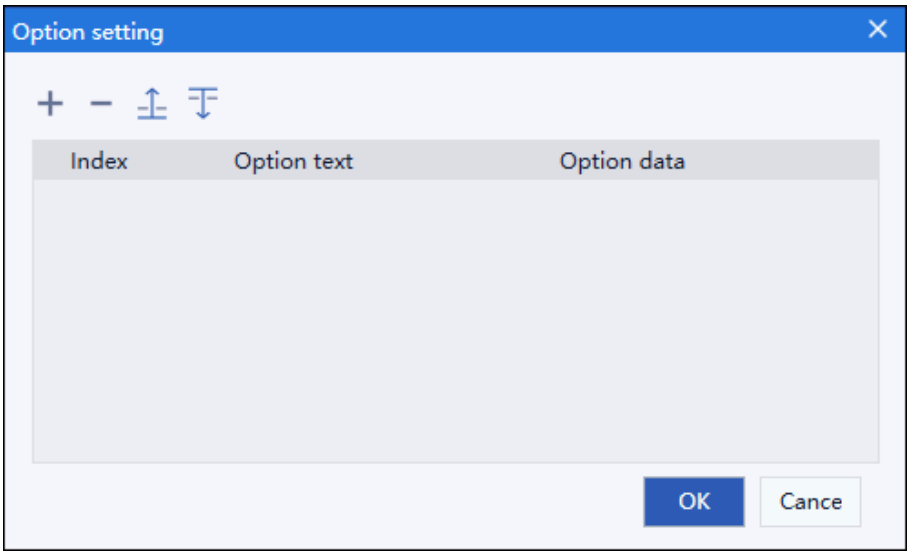
The "combobox" is used to display and select specific values.



- **Combobox creation**

  There are three methods to create a "combobox". For details, see the line creation method.

- **Combobox properties**
  - For basic settings, see Numeric I/O Field > Basic settings.
  - Text: For details, see Numeric I/O Field > Text.
  - Options:

Item	Description
Option settings	Configures the combobox options.  ▪ Click the "Add/Delete" button:  Adds or deletes options. New options are named "Option n" by default. Option indices are automatically generated and cannot be modified, starting from 0 and incrementing. The maximum number of options is 32 (index row range: 0 to 31). ▪ Click the "Move up/Move down" button:  Adjusts the order of options. Only the option text is rearranged. The option index positions remain unchanged.
Variable bound to option	During runtime, the option data corresponding to the selected text in the drop-down list will be output to the bound variable here. Supports inversion display of the bound variable: when the bound variable changes, the selected item of the control automatically follows the change. If the variable value does not match any option data of the current control and inversion display cannot be performed, the combobox displays "***".
Visible entries	The number of visible entries dropdown at runtime. The range is 1 to 32 with a default of value of 3.

◦ Fill:

Item	Description
Text color	Sets the text color of the control. The default is black.
Background color	Sets the background color of the control. The default is white.

◦ Border:

Item	Description
Border	Sets whether the combobox displays a border. This option is checked by default.
Border width	Sets the combobox width range from 0 to 10. The default is 1.
Border color	Sets the border color of the combobox.

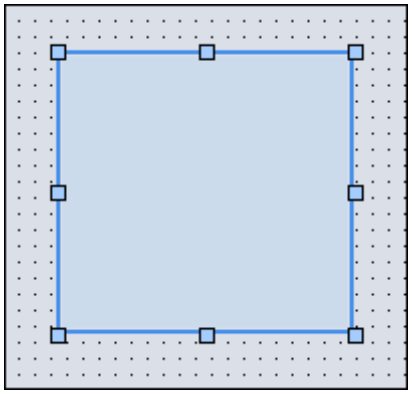
- Operation record:

Item	Description
Record	When the current control performs a write operation, this action is not recorded by default. If recorded, two records will be generated: one for the click action and another for the variable write action. For example, the two logs generated when selecting "Brown" in the initial state of "Combobox_1" are shown in the figure below. Time / Screen / Trigger Obje / Action / Target / Message 1 / 2026/01/15 15:31:00:536 / Screen_1 / ComboBox_1 / Write Variable / Tag_3 / [0]->[145] 2 / 2026/01/15 15:31:00:535 / Screen_1 / ComboBox_1 / Click the ComboBox / ComboBox_1 / [***]->[Brown]
Action description	Description can be up to 256 characters, which will be displayed in the 13.3.5.4.8 Operation Records.

- Basic information: For details, see Line > Basic information.
- Animation: For details, see 13.3.8 Animation Configuration.
- Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.2.11 List Box

The "list box" is used to display and select specific values.



- **List box creation**

  There are three methods to create a "list box". For details, see the line creation method.

- **Properties of list box**
  - For basic settings, see Numeric I/O Field > Basic settings.
  - Text: For details, see Numeric I/O Field > Text.

◦ Options:

Item	Description
Option settings	Configures the list box options. ▪ Click the Add/Delete buttons: Adds or deletes options. New options are named "Option n" by default. Option indices are automatically generated and cannot be modified, starting from 0 and incrementing. The maximum number of options is 32 (index row range: 0 to 31). ▪ Click the Direction button: Adjusts the order of the option text without affecting the index numbers.
Variable bound to option	During runtime, the option data corresponding to the selected text in the drop-down list will be output to the bound variable here. Supports inversion display of the bound variable: when the bound variable changes, the selected item of the control automatically follows the change. If the variable value does not match any option data of the current control and inversion display cannot be performed, the list box displays no selection.

◦ Fill:

Item	Description
Text color	Sets the text display color of the list box.
Background color	Sets the background color of the list box.
Selected item color	Sets the selection color for the selected item in the list box.

◦ Border:

Item	Description
Border	Manages the validity of border property.
Border width	Selects the line width for the list box outline. The line width range is 1 to 10, with a default value of 1.
Border color	Selects the color for the list box outline.

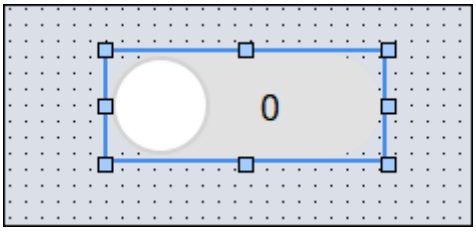
◦ Operation record:

Item	Description
Record	When the current control performs a write operation, this action is not recorded by default. If recorded, two records will be generated: one for the click action and another for the variable write action. For example, the two logs generated when "List box_1" selects "Blue" are shown in the figure below. | | Time | Screen | Trigger Object | Action | Target | Message | | --- | --- | --- | --- | --- | --- | --- | | 1 | 2026/01/16 10:56:45:713 | Screen_1 | Listbox_1 | Write Variable | Tag_3 | [5]->[7] | | 2 | 2026/01/16 10:56:45:712 | Screen_1 | Listbox_1 | Click the ListControl | Listbox_1 | [Red]->[Blue] |
Action description	Description can be up to 256 characters, which will be displayed in the 13.3.5.4.8 Operation Records.

- Basic information: For details, see Line > Basic information.
- Animation: For details, see 13.3.8 Animation Configuration.
- Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.2.12 Text Switch

The "text switch" is associated with a Boolean value and displays the project meaning of whether the Boolean value is set to TRUE or FALSE.



- **Text switch creation**

  There are three methods to create a "text switch". For details, see the line creation method.

- **Properties of text switch**
  - For basic settings, see Numeric I/O Field > Basic settings.
  - Text: For details, see Numeric I/O Field > Text.
  - Interface:

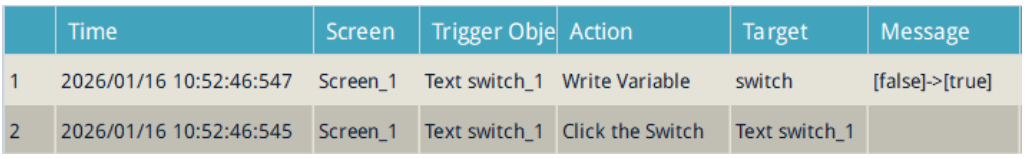
Item	Description
Switch	Specifies whether to display the switch button. This option is selected by default.
Click animation	Animation enabled. This option is selected by default.
ON state text	ON state text. The default value is 1. The text length is limited to 128.

Item	Description
OFF state text	OFF state text. The default value is 0. The text length is limited to 128.
Delay (ms)	Animation delay. The value range is 0 to 60000 ms.
Variable	Sets the reference variable.

- Fill:

Item	Description
Text color	Sets the text color of the control. The default is black.
Background color	Sets the background color of the control. The default is #ECE9DB.

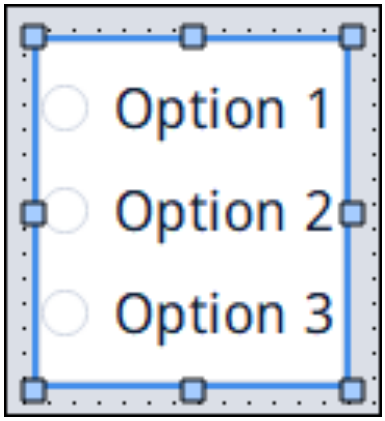
- Operation record:

Item	Description
Record	When the current control performs a write operation, this action is not recorded by default. If recorded, two records will be generated: one for the click action and another for the variable write action. For example, the two logs shown in the figure below are generated when the state of "Text switch_1" switches to "Running".  | | Time | Screen | Trigger Obje | Action | Target | Message | | 1 | 2026/01/16 10:52:46:547 | Screen_1 | Text switch_1 | Write Variable | switch | [false]->[true] | | 2 | 2026/01/16 10:52:46:545 | Screen_1 | Text switch_1 | Click the Switch | Text switch_1 | |
Action description	Description can be up to 256 characters, which will be displayed in the 13.3.5.4.8 Operation Records.

- Basic information: For details, see Line > Basic information.
- Animation: For details, see 13.3.8 Animation Configuration.
- Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.2.13 Radio Button

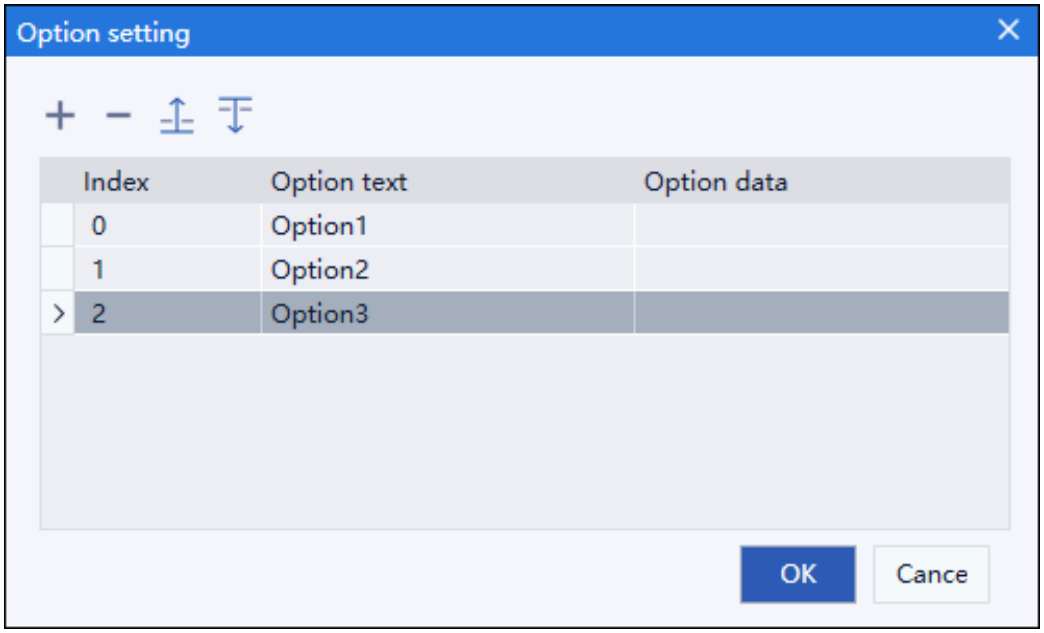
The radio button is used to select a specific value.



- **Radio button creation**

There are three methods to create a radio button. For details, see the line creation method.

- **Radio button properties**
  - For basic settings, see Numeric I/O Field > Basic settings.
  - Text: For details, see Numeric I/O Field > Text.
  - Options:

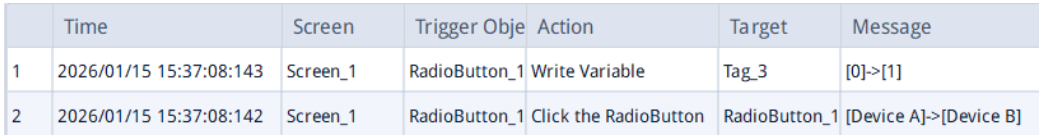
Property	Description
Option settings	Configures the options of the radio button.  A radio button has three options by default. ▪ Click the "Add/Delete" button:  Adds or deletes options. New options are named "Option n" by default. Option indices are automatically generated and cannot be modified, starting from 0 and incrementing. The maximum number of options is 32 (index row range: 0 to 31). ▪ Click the "Move up/Move down" button:  Adjusts the order of options. Only the option text is rearranged. The option index positions remain unchanged.

Property	Description
Variable bound to option	The bound variable types include signed integer arrays, unsigned integer arrays, and one-dimensional boolean arrays. • Signed integer data: The variable value corresponds one-to-one with the option index. For example, if the variable value is 5, the sixth option from the top is selected. • Unsigned integer data: The options correspond one-to-one with bits in the integer variable. For example, option 1 corresponds to bit 0, and option 2 corresponds to bit 1. When an option is selected, the corresponding bit is set. • Boolean one-dimensional array: The array elements (from top to bottom) correspond one-to-one with the option index numbers. The first array element corresponds to the option with index 0. When the second array element is true, the second option is selected.
Option docking	Sets the option docking mode, including left and right. The default is left.
Option layout	Sets the layout mode for options, including vertical and horizontal, with vertical as default.

◦ Fill:

Property	Description
Text color	Sets the color of the radio button text.
Background color	Sets the background color and transparency (0% to 100%) of the group box.

◦ Operation record:

Record	When the current control performs a write operation, this action is not recorded by default. If recorded, two records will be generated: one for the click action and another for the variable write action. For example, the two logs generated when the "Radio button_1" selects the option "Device B" are shown in the figure below.  Time / Screen / Trigger Obje / Action / Target / Message 1 / 2026/01/15 15:37:08:143 / Screen_1 / RadioButton_1 / Write Variable / Tag_3 / [0]->[1] 2 / 2026/01/15 15:37:08:142 / Screen_1 / RadioButton_1 / Click the RadioButton / RadioButton_1 / [Device A]->[Device B]
Action description	Description can be up to 256 characters, which will be displayed in the 13.3.5.4.8 Operation Records.

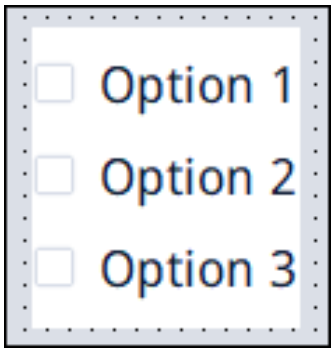
◦ Basic information: For details, see Line > Basic information.

◦ Animation: For details, see 13.3.8 Animation Configuration.

◦ Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.2.14 Checkbox

The "checkbox" allow multiple options to be selected at once.



- **Checkbox creation**

  There are three methods to create a "checkbox". For details, see the line creation method.

- **Checkbox properties**
  - For basic settings, see Numeric I/O Field > Basic settings.
  - Text: For details, see Numeric I/O Field > Text.
  - Options: Checkboxes support binding variable types of Boolean one-dimensional arrays and unsigned integers. Other property settings are the same as for radio buttons. For details, see Radio Button > Options.

> **NOTE**
>
> If the number of options added in the option settings exceeds the number of bits of the bound variable, the excess options will fail during online operation. For example, if 10 options are added and the bound variable is of USINT type, then the last two options of the checkbox will not be operable online.

  - Fill: For details, see Radio Button > Fill.
  - Operation record:

Item	Description
Record	When the current control performs a write operation, this action is not recorded by default. If recorded, two records will be generated: one for the click action and another for the variable write action. For example, the two logs generated when the "Checkbox_1" checks the option "Behind" are shown in the figure below. | | Time | Screen | Trigger Obje | Action | Target | Message | | 1 | 2026/01/16 09:37:12:740 | Screen_1 | CheckBox_1 | Write Variable | Array1 | [true][false][false][false]->[true][false][false][true] | | 2 | 2026/01/16 09:37:12:740 | Screen_1 | CheckBox_1 | Click the CheckBo | CheckBox_1 | [Left]->[Left][Back] |
Action description	Description can be up to 256 characters, which will be displayed in the 13.3.5.4.8 Operation Records.

- Basic information: For details, see Line > Basic information.
- Animation: For details, see 13.3.8 Animation Configuration.
- Event: For details, see 13.3.7.14 Adding an Event.

13.3.5.2.15 Table

The "table" control is used to arrange controls on the screen.



- **Table creation**

  There are three methods to create a "table". For details, see the line creation method.

- **Table properties**
  - For basic settings, see Numeric I/O Field > Basic settings.
  - Border:

Property	Description
Border color	Specifies the border color.
Line width	Sets the line width inside the table. The range is 1 to 10, with a default value of 2.
Line type	Sets the line type inside the table.

  - Fill:

Property	Description
Background color	Sets the table background color and transparency (0% to 100%).

  - Interface properties:

Property	Description
Row count	Sets the number of table rows (range: 1 to 64). When the table spacing mode is set to "Equal", the number of rows can be modified in the property bar. When the spacing mode is set to Custom, the number of rows cannot be modified in the property bar; rows can only be dynamically added or deleted via the right-click menu during table editing.
Column count	Sets the number of table columns (range: 1 to 64). When the table spacing mode is set to "Equal", the number of columns can be modified in the property bar. When the spacing mode is set to Custom, the number of columns cannot be modified in the property bar; rows can only be dynamically added or deleted via the right-click menu during table editing.

  - Alignment:

Property	Description
Text alignment	Sets the position of the control within the cell relative to the cell. ▪ Left align: The control aligns with the left border of the cell. ▪ Center: The control is centered within the cell. ▪ Right align: The control aligns with the right border of the cell. The settings apply to all controls in all table cells.

◦ Spacing:

Property	Description
Spacing	Sets the spacing mode for row height and column width: Equal / Custom. The default is Equal. ▪ Equal: Distributes the row height and column width evenly based on the overall width and height of the table. ▪ Custom: Allows dynamic adjustment of each row height (column width). Each row height (column width) can be different.

◦ Cell properties:

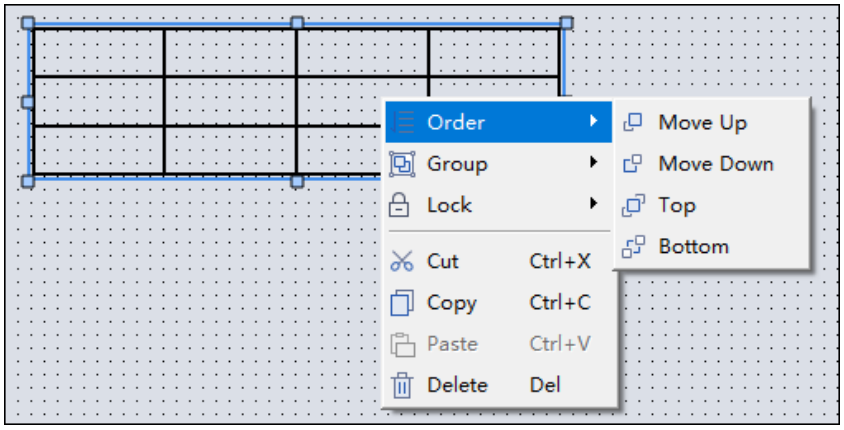
Property	Description
Background color	Sets the background color and transparency of selected cells.
Alignment	Sets the layout position of controls inside selected cells relative to the cell. ▪ Left align: The control aligns with the left border of the cell. ▪ Center: The control is centered within the cell. ▪ Right align: The control aligns with the right border of the cell. The settings apply to controls in selected cells only.

◦ Basic information: For details, see Line > Basic information.

◦ Animation: For details, see 13.3.8 Animation Configuration.

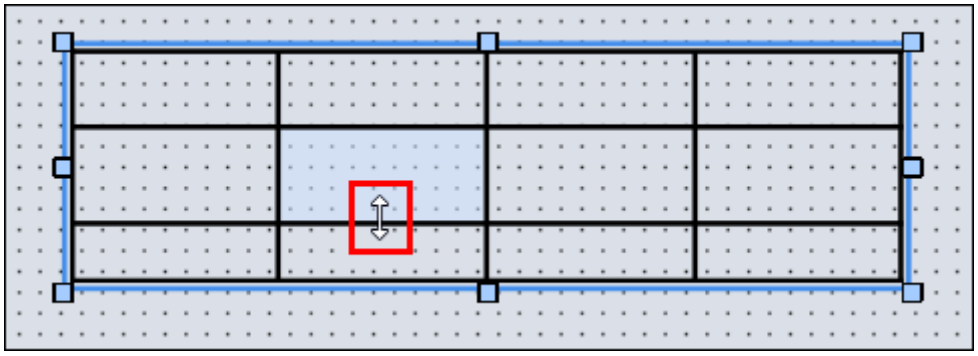
- **Table editing**

◦ In design mode, the table control can be edited, including adding or removing rows and columns, merging or splitting cells, and adjusting row height and column width inside the table. Double-click the table to enter edit mode, where you can perform editing operations. Right-click to open the edit menu:

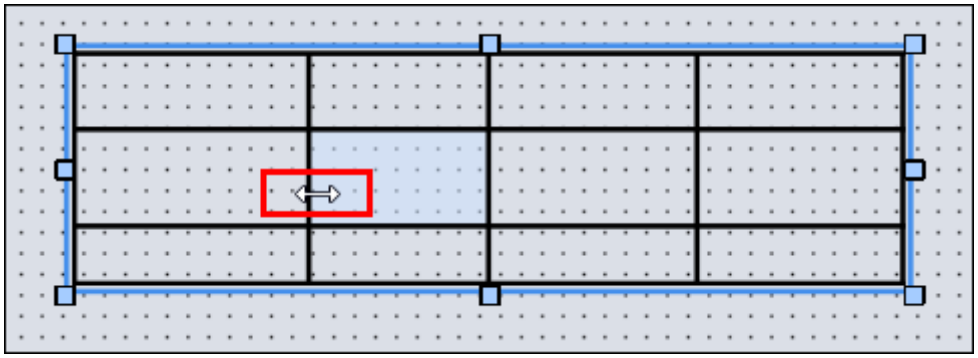


- Insert row/column: Adds a row above or below the selected cell, or add a column to the left or right of the selected cell.
- Merge cells: Merges selected cells into a single cell. The merged cell can continue to be merged with a single cell.
- Split cell: Splits a merged cell back into individual cells, down to the smallest possible cells.
- Delete row/column: Deletes the row or column containing the selected cell.

◦ When the table spacing mode is set to Custom, row height can be adjusted dynamically. Double-click the table to enter edit mode. Move the mouse near the row adjustment area. When the mouse pointer changes to the icon shown below, drag the mouse to adjust the row height.



◦ When the table spacing mode is set to Custom, column width can be adjusted dynamically. Double-click the table to enter edit mode. Move the mouse near the column adjustment area. When the mouse pointer changes to the icon shown below, drag the mouse to adjust the column width.

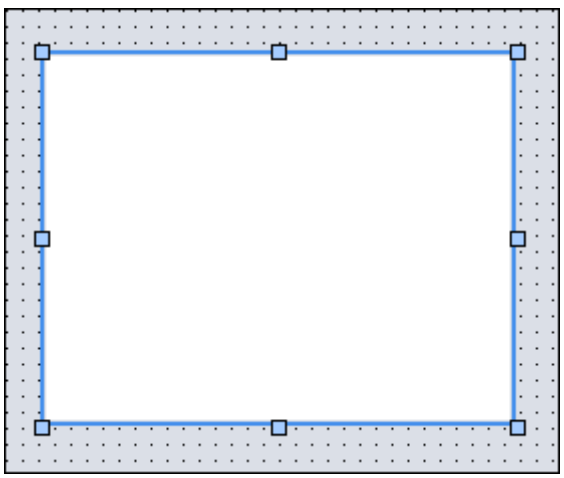


- **Layout rules for tables and other controls**
  - ◦ All controls except tables can be dragged into the table for layout.
  - ◦ When moving a control within the table cell range, its position is automatically adjusted according to the configured alignment after moving. Controls move with the table when dragging.
  - ◦ If a control already exists in a cell, no additional control can be added to it. Even if dragged into the table, its position will not be adjusted automatically, and it will not move when the table is dragged.

- If the display area of a table cell is smaller than that of the control, the control will not be scaled. For better appearance, scaling the control manually is recommended.
- When the entire table is moved or scaled, the control positions inside the table automatically adjust according to the cell positions. Only display positions are adjusted without scaling.
- Move a control inside the table.
  - The control position in the cell is automatically adjusted according to the configured alignment after movement.
  - When moved out of the table, the control is removed from the cell layout.
- Deleting the table does not delete the controls laid out inside the table.
- When copying a table, only the table is copied. The controls laid out inside the table are not copied.
- Controls inside a table cannot be grouped with other controls.
- Controls already laid out inside a table cannot be grouped with each other.
- Grouped graphic elements laid out inside a table cannot be ungrouped.
- In runtime mode, only the table is drawn. All table operations in design mode are not supported in runtime mode.
- After the table is locked, it cannot enter edit mode.
- Graphic elements laid out within a table cannot be moved using the keyboard. Their position and size cannot be modified via the property bar. Alignment, equal size, and equal spacing operations are not allowed.
- Graphic elements can only be placed into the table by mouse dragging. Positioning into the table via other methods (keyboard movement, property bar position modification, alignment, etc.) will not integrate the graphic elements into the table layout, nor will it move with the table.
- After a graphic element laid out in the table has undergone animation such as scaling, moving, or rotating, its final position and size after animation take precedence, and it is no longer controlled by the layout position.

13.3.5.2.16 Picture-in-Picture View

The "picture-in-picture view" control allows multiple views to be displayed simultaneously on the same screen, with the sub-picture placed within the main screen.



> **NOTE**
>
> Only the current screen supports picture-in-picture view. The picture-in-picture control is hidden when other screens are opened.

- **Picture-in-Picture view creation**

  There are three methods to create a "picture-in-picture view". For details, see the line creation method.

- **Properties of picture-in-picture view**
  - For basic settings, see Numeric I/O Field > Basic settings.
  - Interface:

Item	Description
Picture-in-picture	Includes the options of "screen" and "variable".
Screen	Selects a screen via the screen selection box to display the established picture-in-picture view.
Variable	Selects the variable via the variable selection box. Supported data types: USINT, UINT, UDINT, ULINT. When the variable value corresponds to a different screen ID, the picture-in-picture view switches to the corresponding screen.
Horizontal scrollbar	Sets whether a horizontal scrollbar is needed, including "As needed", "Always off", and "Always show". The default is "As needed".
Vertical scrollbar	Sets whether a vertical scrollbar is needed, including "As needed", "Always off", and "Always show". The default is "As needed".
Start position	Sets the starting view position, including "Top-left, Top-center, Top-right, Middle-left, Center, Middle-right, Bottom-left, Bottom-center, and Bottom-right". The default is "Center".

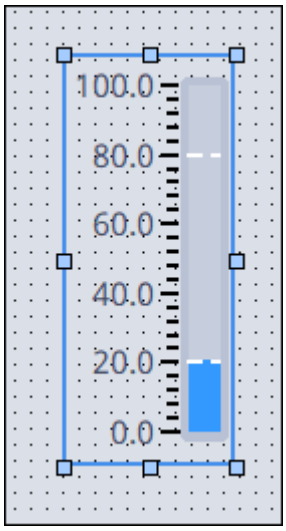
  - Basic information: For details, see Line > Basic information.

13.3.5.3 Data Visualization

- 13.3.5.3.1 Bar graph
- 13.3.5.3.2 Slider
- 13.3.5.3.3 Progress bar
- 13.3.5.3.4 Circular progress bar
- 13.3.5.3.5 Pie chart
- 13.3.5.3.6 F(t) trend curve
- 13.3.5.3.7 F(x) trend curve
- 13.3.5.3.8 Scale

13.3.5.3.1 Bar Graph

The "bar graph" control is used to monitor process values in a predefined range.



- **Bar creation**

  There are three methods to create a "bar graph". For details, see the line creation method.

- **Properties of bar graph**

  - For basic settings, see Numeric I/O Field > Basic settings.
  - Style:

Property	Description
Text color	Sets the text display color of the bar graph.
Background color	Sets the background color of the bar graph.
Fill style	Sets the background fill style of the bar graph.
Font	Sets the text font of the bar graph. The default font is Droid Sans Fallback.
Font size	Sets the text font size of the bar graph. The default size is 16.
Font style	Sets the font style of the bar graph text. Supports regular, bold, italic, and underline. The default is regular.

  - Interface:

Property	Description
Max. value (static)	Sets the maximum value of the bar scale. The default value is 100.

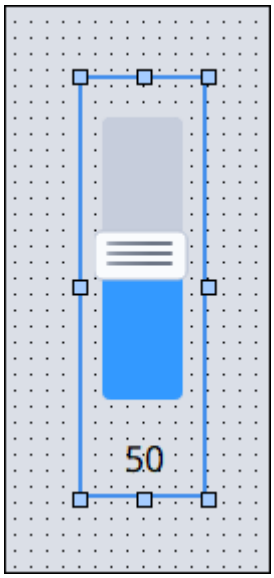
Property	Description
Max. value (variable)	Sets the variable reference for the maximum value of the bar scale. When a variable is associated, the property Max. value (Constant) is not editable. NOTE When the Max. value (variable) is associated with a variable, the Max. value (static) is grayed out and cannot be set.
Min. value (static)	Sets the minimum value of the bar scale. The default value is 0.
Min. value (variable)	Sets the variable reference for the minimum value of the bar scale. When a variable is associated, the property Min. value (Constant) is not editable. NOTE When the Min. value (variable) is associated with a variable, the Min. value (static) is grayed out and cannot be set.
Reference variable	Variable used to display the process value on the bar graph.
Progress bar inversion	Specifies whether the normal value color is displayed on top. This option is not selected by default.
Scale label display	Display scale values when selected. This option is selected by default.
Limit line display	Display limit lines when selected. This option is selected by default.
Limit mark display	Display limit marks when selected. This option is not selected by default.
Major scale value	Number of intervals for major scale values. The default value is 5.
Minor scale value	Number of intervals for minor scale value. The default value is 5.
Decimal places	Number of decimal places for display text. The value range is 0 to 16 with a default value of 1.
Scale position	Sets the position for scale label display. Value range: No/Left/Right/Top/Bottom. The default value is Left.
Background color of bar graph	Sets the background color of the bar graph.
Normal color	Sets the color displayed when the process value is within the normal range.

Property	Description
Color above upper limit	The color displayed when the process tag value exceeds the upper limit. The default is red.
Color below lower limit	The color displayed when the process tag value exceeds the lower limit. The default is yellow.

- Basic information: For details, see Line > Basic information.
- Animation: For details, see 13.3.8 Animation Configuration.

13.3.5.3.2 Slider

The "slider" is used to adjust and display process variable value information.



- **Slider creation**

  There are three methods to create a "slider". For details, see the line creation method.

- **Slider properties**
  - For basic settings, see Numeric I/O Field > Basic settings.
  - Text:

Property	Description
Font	Sets the text font. The default font is Droid Sans Fallback.
Font size	Sets the text size. The default size is 18.
Font style	Sets the text font style. Supports regular, bold, italic, and underline. The default is regular.
Font color	Sets the font color. The default is black.

  - Interface:

Property	Description
Maximum value	Maximum value type of slider: Static, Variable. Static (default value): The maximum value is a static value. The property is not editable. Variable: The maximum value reads from a variable. The property is not editable.
Max. value (static)	Slider maximum value. The range is -9998 to 99999 with a default value of 100.
Max. value (read variable)	Variable associated with the slider's maximum value.
Minimum value	Minimum value type of slider: Static, Variable. Static (default value): The minimum value is a static value. The property is not editable. Variable: The minimum value reads from a variable. The property is not editable.
Min. value (static)	Slider minimum value. The range is -9999 to 99998 with a default value of 0.
Min. value (read variable)	Variable associated with the slider's minimum value.
Process value	Variable associated with the slider's process value.
Slider direction	Slider display direction: left to right, right to left, bottom to top, and top to bottom. Default: bottom to top.
Process value visible	Specifies whether to display the associated variable value. This option is selected by default.
Scale visible	Specifies whether to display the slider scale. This option is selected by default.
Process value decimal places	Number of decimal places for displaying the process value. The range is 0 to 16 with a default value of 0.
Scale position	Sets the scale position after "Scale visible" is selected. For horizontal slider, top/bottom are available; for vertical slider, left/right are available.
Scale decimal places	Sets the number of decimal places retained for the scale after "Scale visible" is selected. The default value is 0.
Major scale divisions	Sets the number of major scale divisions after "Scale visible" is selected. The default value is 5.
Minor scale divisions	Sets the number of minor scale divisions after "Scale visible" is selected. The default value is 5.

- Style:

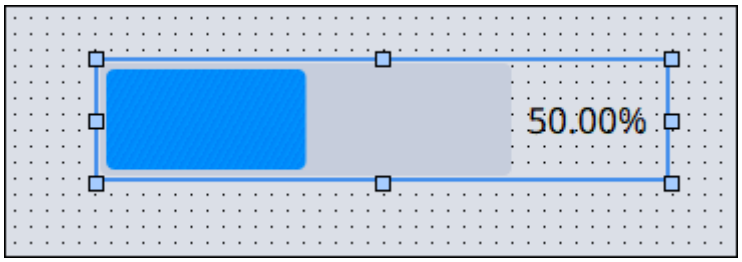
Property	Description
Fill	Sets the fill style of the slider. For specific style settings, see Rectangle > Fill.

Property	Description
Bar background color	Sets the color of the slider bar part.
Progress color	Sets the color of the slider progress part.
Slider color	Sets the slider color.
Slider border color	Sets the slider border color.
Color above upper limit	Color displayed by progress bar when variable value exceeds maximum value.
Color below lower limit	Color displayed by progress bar when variable value exceeds minimum value.

- Basic information: For details, see Line > Basic information.
- Animation: For details, see 13.3.8 Animation Configuration.

13.3.5.3.3 Progress Bar

The "progress bar" is used to monitor and adjust a process value within a defined range.



- **Progress bar creation**

  There are three methods to create a "progress bar". For details, see the line creation method.

- **Progress bar properties**
  - For basic settings, see Numeric I/O Field > Basic settings.
  - Interface properties:

Property	Description
Max. value (constant)	Maximum value of the progress range. The default is 100.
Max. value (variable)	Variable associated with the maximum value of the progress range. When a variable is associated, the property Max. value (Constant) is not editable. NOTE Variables with data types STRING, DATETIME, or BOOL are not supported for association.

Property	Description
Min. value (constant)	Minimum value of the progress range. The default is 0.
Min. value (variable)	Variable associated with the minimum value of the progress range. When a variable is associated, the property Min. value (Constant) is not editable. NOTE Variables with data types STRING, DATETIME, or BOOL are not supported for association.
Reference variable	Sets the associated variables.
Direction	Sets the direction of the progress bar, including horizontal and vertical. The default is horizontal.
Text visible	Sets whether to display progress text. This option is selected by default.
Display decimal point	Sets whether progress information displays decimal points. Two decimal places are displayed by default.

- Text:

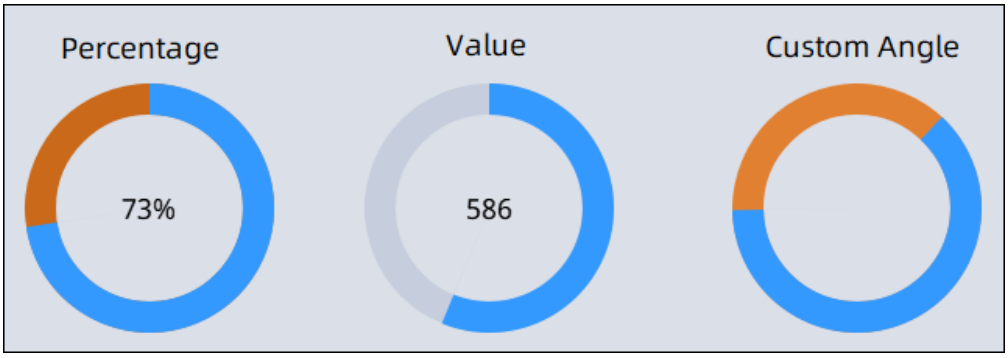
Property	Description
Font	Sets the text font. The default font is Droid Sans Fallback.
Font size	Sets the text size. The default size is 18.
Font style	Sets the text font style. Supports regular, bold, italic, and underline. The default is regular.
Text position	Outside the graphic: Progress information is displayed outside the graphic. When the direction property is horizontal, it is displayed on the right; when the direction property is vertical, it is displayed below. Inside the graphic: Progress information is displayed inside the graphic, centered by default.
Text color	Sets the display color of progress text.
Background color	Sets the background color.
Fill	Sets the fill style of the progress bar. For specific style settings, see Rectangle > Fill.
Color above upper limit	Color displayed by progress bar when variable value exceeds maximum value.

Property	Description
Color below lower limit	Color displayed by progress bar when variable value exceeds minimum value.

- Basic information: For details, see Line > Basic information.
- Animation: For details, see 13.3.8 Animation Configuration.

13.3.5.3.4 Circular Progress Bar

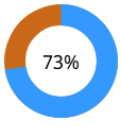
The "circular progress bar" control offers three display modes for visualizing process variables such as level, progress and completion.



- **Circular progress bar creation**

  There are three methods to create a "circular progress bar". For details, see the line creation method.

- **Properties of circular progress bar**
  - For basic settings, see Numeric I/O Field > Basic settings.
  - Data:

Property	Description
Display mode	Percentage: The fill percentage is calculated automatically based on the set range limits and real-time values. It is filled in a clockwise direction. Fill percentage = [(Real-time value - Minimum value) / (Maximum value - Minimum value)] * 100% 
	Value: The real-time value is directly displayed, and the fill range conversion formula is the same as for percentage. It is filled in a clockwise direction. 

Property	Description
	Custom angle: The fill range of the progress bar is determined by user-defined start and end angles. Filling proceeds clockwise from the start angle to the end angle. For example, the reference 0° is at 90° (3 o'clock direction), the start angle is 0°, and the end angle is 180°, with the fill range shown as the blue area in the figure. 
Max. value (constant/ variable)	Maximum fill value. Default: 100. Setting range: -9999 to 99999 Variable types supported: UINT, INT, USINT, SINT, UDINT, DINT, ULINT, LINT, REAL, and LREAL.
Min. value (constant/ variable)	Minimum fill value. Default: 100. Setting range: -9999 to 99999 Variable types supported: UINT, INT, USINT, SINT, UDINT, DINT, ULINT, LINT, REAL, and LREAL.
Process variables	Data types supported for display variables: UINT, INT, USINT, SINT, UDINT, DINT, ULINT, LINT, REAL, and LREAL.
Start angle	Absolute angle relative to the "Reference 0°", used as the fill start point.
End angle	Absolute angle relative to the "Reference 0°", used as the fill end point.
Reference 0°	It defines the absolute reference point for the fill start and end points. Setting range: 0° to 360°, where 0° is at the 12 o'clock direction and 90° is at the 3 o'clock direction. Default: 0°.

- Text:

Property	Description
Font	Sets the text font. The default font is Droid Sans Fallback.
Font size	Sets the text size. The default size is 18.
Font style	Sets the text font style. Supports regular, bold, italic, and underline. The default is regular.

- Appearance:

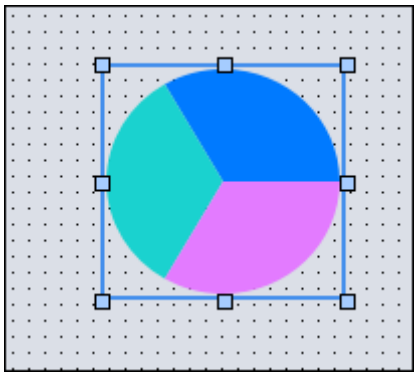
Property	Description
Text visible	Sets whether to display progress text. This option is selected by default.
Display decimal point	Sets whether progress information displays decimal points. Two decimal points are displayed by default.
Decimal point	Number of decimal points displayed for progress, range 0 to 16, default 2.

Property	Description
Outer ring size	Ring size, range 1 to 999, default 12. Maximum value is half the control width. Display style at maximum outer ring size: 
Text color	Sets the display color of progress text.
Background color	Sets the background color.
Progress color	Sets the color of progress bar.
Color above upper limit	Color displayed by progress bar when process variable value exceeds max imum value.
Color below lower limit	Color displayed by progress bar when process variable value is less than mi nimum value.

- For basic information, see Line > Basic information.
- Animation: For details, see 13.3.8 Animation Configuration.

13.3.5.3.5 Pie Chart

The "pie chart" is used to display data proportions.



- **Pie chart creation**

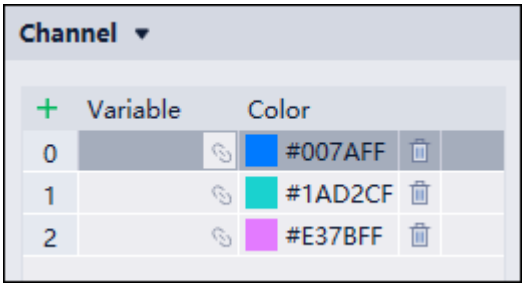
  There are three methods to create a "pie chart". For details, see the line creation method.

- **Pie chart properties**
  - For basic settings, see Numeric I/O Field > Basic settings.
  - Text:

Property	Description
Font	Sets the text font. The default font is Droid Sans Fallback.
Font size	Sets the text size. The default size is 18.

Property	Description
Font style	Sets the text font style. Supports regular, bold, italic, and underline. The default is regular.

◦ Interface:

Property	Description
Style	Sets the text to be displayed on the pie chart. Options include no data, value, and percentage. Default: no data.
Number of channels	Sets the number of pie chart segments (up to 8 segments can be set). The default value is 3.
Channel settings	Sets the variable associated with the pie chart channel and the display color.  Channel: A pie chart has three channels by default. ▪ Click the button to add a channel. ▪ Click the button to delete the selected channel. Variable: Set the variable associated with the channel. Supported data types for associated variables: USINT, UINT, UDINT, ULINT. Color: Set the display color of the channel.

◦ Fill:

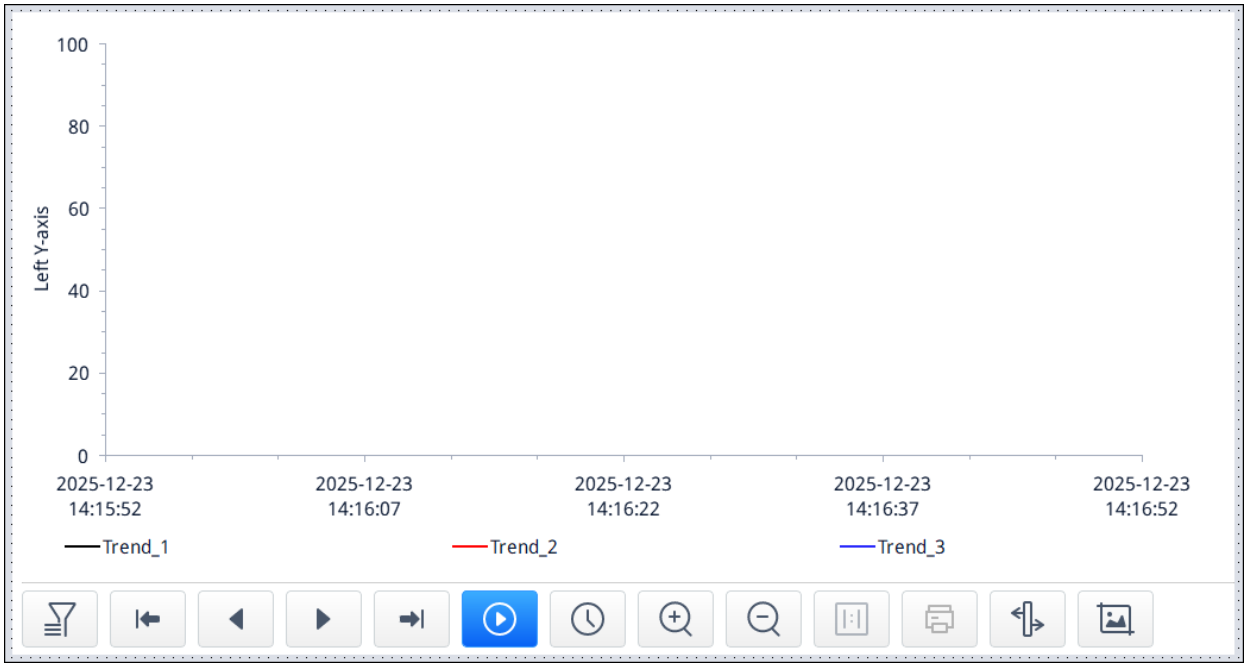
Property	Description
Text color	Specifies the text color of pie chart.
Background color of outer frame	Specifies the background color of pie chart with transparency (0% to 100%).
Scaling ratio	Sets the scaling ratio when drawing the pie chart (0% to 100%).

◦ For basic information, see Line > Basic information.

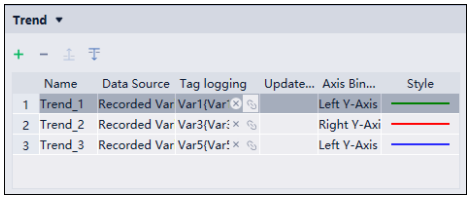
◦ Animation: For details, see 13.3.8 Animation Configuration.

13.3.5.3.6 F(t) Trend Curve

The "F(t) trend curve" is an enhanced screen object used to display the trend of a variable over time.



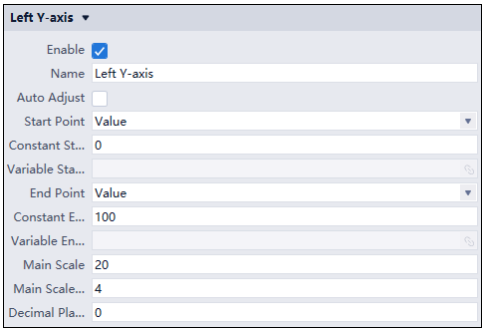
- **F(t) trend curve creation There are three methods to create an "F(t) trend curve". For details, see the** line creation method.
- **F(t) trend curve properties**
  - For basic settings, see Numeric I/O Field > Basic Settings.
  - Trend:

Item	Description	Example
Name	The legend name of the curve. One name corresponds to one curve. New curves are named as Trend_n by default, with n starting from 1. Addition and deletion of curves are performed by clicking icons + and −, supporting up to 100 curves.	
Data source	The location from which variable values are retrieved during online operation. Real-time variable: Retrieves real-time values from the database. Record variable: Retrieves historical values from data records.	
Variable recording	Associated variables	
Update interval	Applicable when a real-time variable is used as the data source. The period at which the current control acquires the real-time value.	
Axis binding	Binds to the Y-axis. Supports configuration when the left or right Y-axis is enabled.	
Style	Configuration of line style and color.	

◦ Time axis:

Item	Description	Example
Time span	The time interval displayed in the view window.	
Horizon value		
Time format	Time display format	
Time format	Supports 24-hour format and 12-hour format.	
Scale value	Manages the display and hiding of time text on the time axis.	

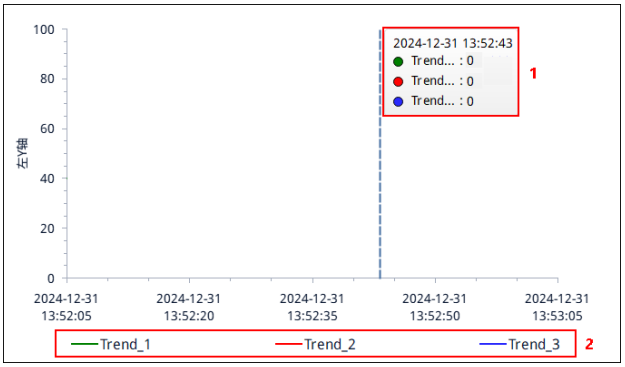
◦ Left/Right Y-axis:

Item	Description	Example
Enable	Check this option to display the current axis on the control and enables editing of related parameters.	
Name	Y-Axis name	
Automatic adjustment	By default, three decimal points are retained based on the start and end values, and the display space for Y-axis scale values is automatically adjusted.	
Start/End	Y-axis tick range	
Constant start/end values	Supports constant or variable values as the maximum and minimum scale values.	
Variable start/end values		
Major scale	Sets the unit of scale length and precision of scale values.	
Major scale segments		
Decimal points		

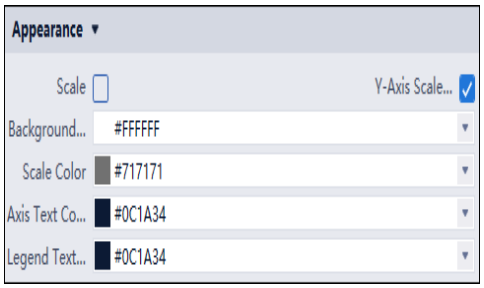
◦ Screenshot:

Item	Description	Example
Storage location	Supported storage locations include local disk, SD card, and USB, with SD card and USB applicable to touch screen devices. ▪ Local disk: The maximum number of screenshot files stored is 100. For more details, see 13.12.1 Screen Display Configuration. ▪ SD card and USB: The number of stored screenshots depends on the file size and available storage space. If the storage is full, the screenshot operation will fail.	
Constant path	Settings for the storage path of screenshot files. ▪ Local disk: The storage path is fixed as the screenshot directory and cannot be changed. ▪ SD card and USB: Only the folder name needs to be entered, such as **screen**, without entering the full path. If the folder **screen** does not exist, the software will automatically create it.	

◦ Text:

Item	Description	Example
Axis text	Font, size, and style of the text on the coordinate axes.	
Ruler text	Font, size, and style of the variable information text (position 1) on the ruler when the control is running online.	
Legend text	Font, size, and style of the legend text (position 2).	

◦ Appearance:

Item	Description	Example
Ruler	Specifies whether to display the ruler button  when the control runs online. Selected: display; not selected: hide.	
Y-axis scale value	Manages the display and hiding of Y-axis scale values.	

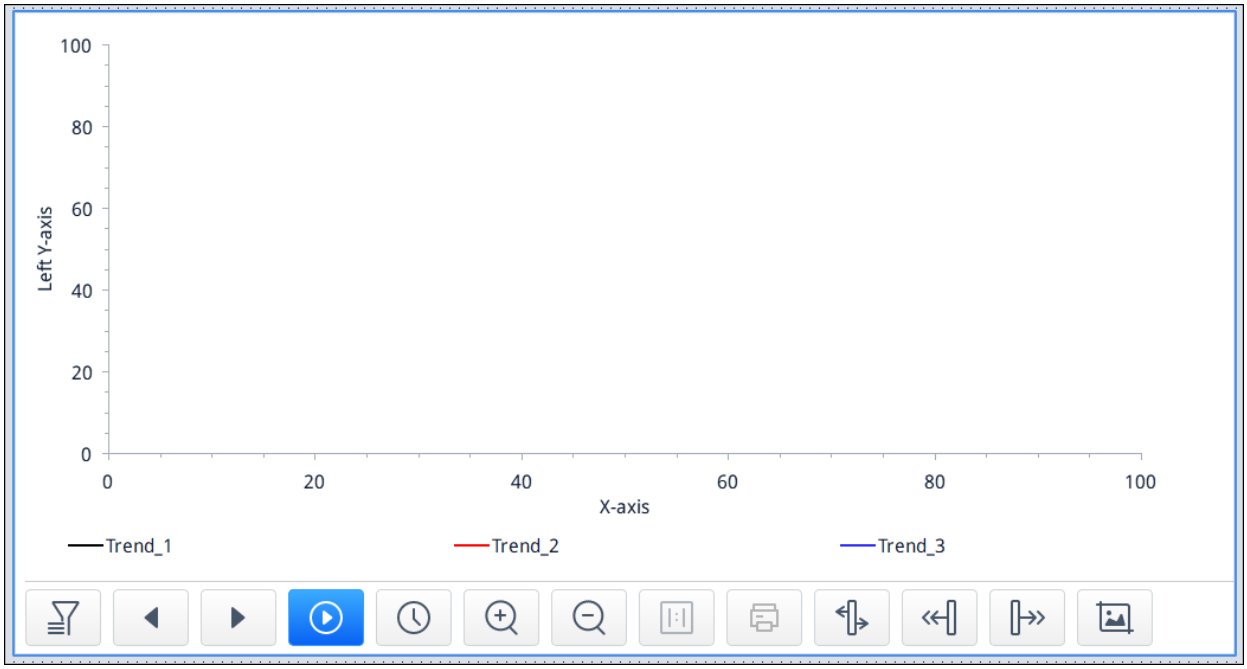
Item	Description	Example
Background color	Sets the trend chart background color. The default is white #FFFFFF.	
Ruler color	Sets the ruler color. The default is #717171.	
Axis text color	Sets the text display color on the X/Y axis. The default is #0C1A34.	
Legend text color	Sets the legend text color. The default is #0C1A34.	

  - For basic information, see Line > Basic information.
  - For animation, see 13.3.8 Animation Configuration.

- **Online operations of the**
  F(t) trend curve For details on supported operations, see 13.15.3 Online Historical Data Viewing.

13.3.5.3.7 F(x) Trend Curve

The "F(x) trend curve" is an enhanced screen object used to display the changing trends among multiple variables.



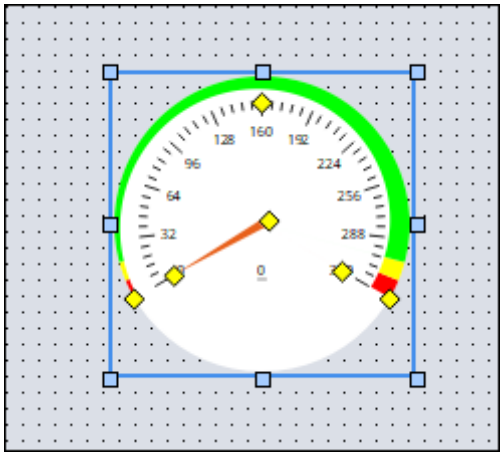
- **F(x) trend curve creation**
  There are three methods to create an "F(x) trend curve". For details, see the line creation method.
- **F(x) trend curve properties**
  - For basic settings, see Numeric I/O Field > Basic information.
  - For trend, see F(t) Trend Curve > Trend.
  - For time axis, see F(t) Trend Curve > Time axis.
  - For screenshot, see F(t) Trend Curve > Trend.

- For text, see F(t) Trend Curve > Text.
- For appearance, see F(t) Trend Curve > Appearance.
- For basic information, see Line > Basic information.
- For animation, see 13.3.8 Animation Configuration.

- **Online operations of the F(x) trend curve**
  For details on supported operations, see 13.15.3 Online Historical Data Viewing.

13.3.5.3.8 Scale

The "scale" is an animated graphic that simulates a rotating pointer instrument, used to display the value of the connected variable. The pointer continuously changes position based on the variable value, and the scale pointed to by the pointer represents the value of the connected variable.



- **Scale creation**

  There are three methods to create a "scale". For details, see the line creation method.

- **Scale properties**
  - For basic settings, see Numeric I/O Field > Basic settings.
  - Style:

Property	Description
Draw options	Sets the drawing style for the scale background fill, including drawing fill, fill, and drawing.
Fill brush	Sets the background color fill style of the scale. For specific style settings, see Rectangle > Fill.
Border color	Sets the border color and transparency of the scale background fill.
Line width	Sets the border width of the scale background fill.
Line type	Sets the border line style of the scale background fill.

  - Dial information:

Property	Description
Background color	Sets the fill background color and transparency within the dial.

Property	Description
Font color	Sets the font color for the label text, unit text, and current value within the dial.
Font	Sets the text font. The default font is Droid Sans Fallback.
Font size	Sets the text size. The default size is 16.
Label text	Sets the dial label text.
Unit text	Sets the dial unit text.
Current value variable	Sets the variables associated with the current value of the dial. Supported data types: SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL, LREAL.
Display current value	Specifies whether to display the current value of the dial. Enabled by default.
Hide background	Specifies whether to hide the dial background. Disabled by default.

◦ Start data:

Property	Description
Start angle	The start position is determined by the starting angle. The default value for the starting angle is 330, and the start angle range is 0 to 360. When entering values, the input box displays the range. If the value exceeds the range, it cannot be entered.
Start value	The start value corresponds to the value at the start angle. The default value for the start value is 320, and the value range is -2147483648 to 2147483647. When entering values, the input box displays the range. If the value exceeds the range, it cannot be entered.
Warning value	The warning value corresponds to the value at the start angle. The default value for the warning value is 300, and the value range is -2147483648 to 2147483647. When entering values, the input box displays the range. If the value exceeds the range, it cannot be entered.
Error value	The error value corresponds to the value at the start angle. The default value for the error value is 310, and the value range is -2147483648~2147483647. When entering values, the input box displays the range. If the value exceeds the range, it cannot be entered.

◦ End data:

Property	Description
End angle	The end position is determined by the end angle. The default value for the end angle is 210, and the end angle range is 0 to 360. When entering values, the input box displays the range. If the value exceeds the range, it cannot be entered.
End value	The end value corresponds to the value at the end angle. The default value for the end value is 0, and the value range is -2147483648 to 2147483647. When entering values, the input box displays the range. If the value exceeds the range, it cannot be entered.
Warning value	The warning value corresponds to the value at the end angle. The default value for the warning value is 20, and the value range is -2147483648 to 2147483647. When entering values, the input box displays the range. If the value exceeds the range, it cannot be entered.
Error value	The error value corresponds to the value at the end angle. The default value for the error value is 10, and the value range is -2147483648 to 2147483647. When entering values, the input box displays the range. If the value exceeds the range, it cannot be entered.

◦ Scale information:

Property		Description
Major scale	Bisection	Sets the number of major scales. Data type is UINT. Value range is 0 to 255. Default value is 10.
	Color	Sets the color of the major scale. Default is black #000000.
	Width and height	Sets the width and height of the major scale. Data type is UINT. Value range is 0 to 255. Default width is 2 and height is 16.
Minor scale	Bisection	Sets the number of minor scales. Data type is UINT. Value range is 0 to 255. Default value is 5.
	Color	Sets the color of the minor scale. Default is black #000000.
	Width and height	Sets the width and height of the minor scale. Data type is UINT. Value range is 0 to 255. Default width is 2 and height is 8.
Scale radius		Distance from the center axis to the outer edge of the scale. Data type is UINT. Default value is 145.
Annotation radius		Distance from the center axis to the outer edge of the label font. Data type is UINT. Default value is 110.
Font color		Sets the dial color. Default is black #000000.
Font size		Sets the font size. Default is 16, unit: px.

Property	Description
Coincident display	Select the value to display when values overlap on the scale. Options: maximum value or minimum value.
Display text	Specifies whether to display the text within the dial. Enabled by default.
Callout interval display	Specifies whether to enable interval display of scale values. Disabled by default.

◦ Pointer information:

Property	Description
Pointer color	Configures the fill color of the pointer. Default is red #E8601A.
Pointer width	Sets the width of the pointer and the center axis width. Data type: INT; Range: 0 to 255; Default width: 10.
Pointer radius	Sets the long radius value of the pointer. Data type is INT. Range is 0 to 255. Default length is 120.
Short pointer radius	Sets the short radius value of the pointer. Data type is INT. Range is 0 to 255. Default length is 0.
Axis color	Configures the fill color of the pivot point. Default is white #FFFFFF.
Pivot radius	Configures the radius of the pivot point. Default value is 2.

◦ Warning ring information:

Property	Description
Start width	Sets the width at the start of the scale ring. Default value is 26.
End width	Sets the width at the end of the scale ring. Default value is 4.
Normal color	Color when the process variable value is within the normal range. Default is green #00FF00.
Warning color	Color when the process variable value is within the alarm range. Default is yellow #FFFF00.
Error color	Color when the process variable value is within the error range. Default is red #FF0000.

◦ Basic information: For details, see Line > Basic information.

◦ Animation: For details, see 13.3.8 Animation Configuration.

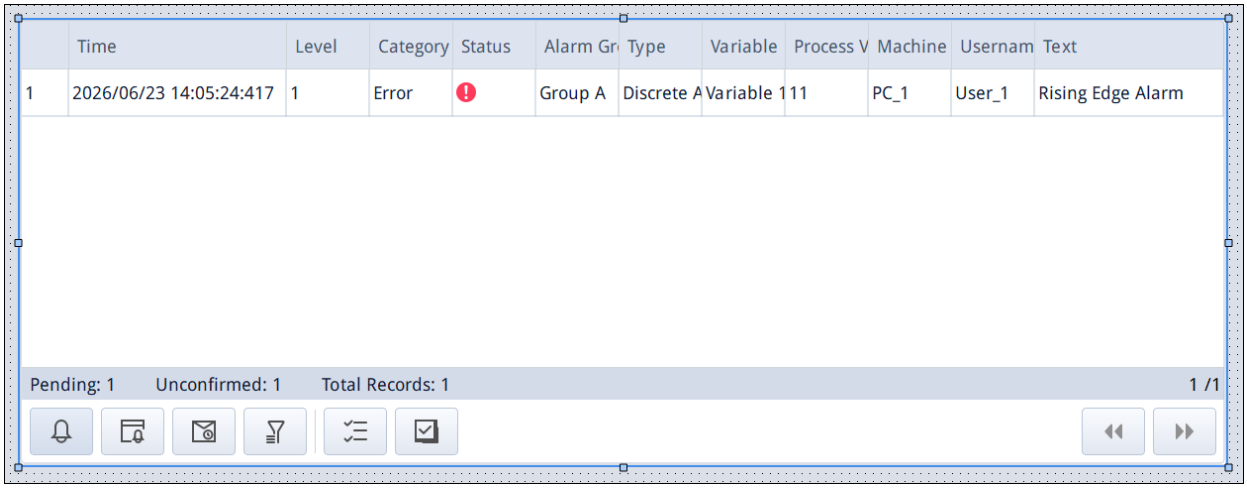
13.3.5.4 Production and Management

- 13.3.5.4.1 Alarm View
- 13.3.5.4.2 Alarm Bar

- 13.3.5.4.3 Recipe View
- 13.3.5.4.4 User Login
- 13.3.5.4.5 User View
- 13.3.5.4.6 Variable Log View
- 13.3.5.4.7 System Diagnostics
- 13.3.5.4.8 Operation Log
- 13.3.5.4.9 SFC View
- 13.3.5.4.10 Code View

13.3.5.4.1 Alarm View

The "Alarm view" window dynamically displays information of a specified category that meets certain conditions in the running system.



- **Alarm view creation**

  There are three methods to create an alarm view. For details, see the line creation method.

- **Alarm view properties**

  - For basic settings, see Numeric I/O Field > Basic settings.
  - Data:

Property	Description
Column header	Sets whether to display the header row of the alarm view. Default is selected.
Toolbar	Sets whether to display the toolbar of the alarm view. Default is selected.
Status bar	Sets whether to display the status bar of the alarm view. Default is selected.
Time (Milliseconds)	Sets whether to display the time column in milliseconds; selected by default.
Show grid	Sets whether to display grid lines. Default is selected.

Property	Description
Default display	Sets the content to be displayed by default in the alarm view. Options: Real-time alarms, real-time alarm events, and historical alarm events. Default is real-time alarms.
Sorting method	Options: Sorting by time descending, sorting by level ascending, and sorting by level descending. Default is sorting by time descending. When sorting by level, alarms with the same level are sorted by time in descending order. During runtime, clicking the headers of Time and Level columns can switch the sorting methods. When not sorting by level, click the header of the Level column to sort the alarms by level, and then click again to switch to descending order.
Column setting	Sets the columns to be displayed in the alarm view. Default display shows "<Select columns to display>". All columns are displayed by default. After column width configuration, if there is remaining width in the table, the last column automatically stretches; if the table width is insufficient, a horizontal scroll bar is automatically displayed.
Header row height	Sets the height of the header row in the alarm view.
Content row height	Sets the height of the content rows in the alarm view.
Filter conditions	Sets the filter conditions for displayed alarms in the alarm view.

◦ Fill:

Property	Description
Header text color	Sets the foreground color of the header row. Default is black #000000.
Header background color	Sets the background color of the header row. Default is #E5E6EC.
Selected row color	Sets the color of selected content rows. Default is #0078D7.
Grid line color	Sets the grid line color. Default is #CFD5D8.

◦ Text:

Property	Description
Font	Sets the font for the header row. Default is Droid Sans Fallback.
Font size	Sets the font size for the header row. Default is 16.
Font style	Sets the font style for the header row. Supports regular, bold, italic, and underline. Default is regular.

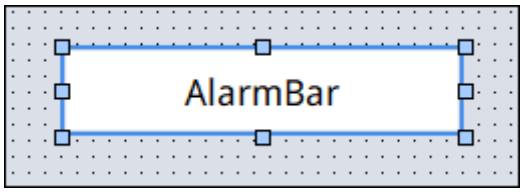
Property	Description
Content font	Sets the font for the content row. Default is Droid Sans Fallback.
Content font size	Sets the font size for the content row. Default is 16.
Content font style	Sets the font style for the content row. Supports regular, bold, italic, and underline. Default is regular.

  - Basic information: Refer to Line > Basic information.
  - Animation: For details, see 13.3.8 Animation Configuration.

- Online operations
  For the operations supported when the alarm view is running online, see 13.15.4 Online Alarm View Operations.

13.3.5.4.2 Alarm Bar

The alarm bar is used to display the active alarm.



- **Alarm bar creation**

  There are three methods to create an alarm bar. For details, see the line creation method.

- **Alarm bar properties**
  - For basic settings, see Numeric I/O Field > Basic settings.
  - Text:

Property	Description
Font	Sets the font for the alarm bar text. Default is Droid Sans Fallback.
Font size	Sets the font size for the alarm bar text. Default is 18.
Font style	Sets the font style of the alarm bar text. Supports regular, bold, italic, and underline. Default is regular.

  - Flashing:

Property	Description
Process variables	Sets the standard flashing-associated variable. When the variable value is TRUE, the alarm bar flashes. When the variable value is FALSE, the alarm bar does not flash.

Property	Description
Flashing mode	Sets the flashing mode to no flashing or standard flashing. Default is no flashing. No flashing: The alarm bar does not flash during system operation. Standard flashing: The alarm bar flashes during system operation.

- Display:

Property	Description
Display date	Sets whether to display the date when an alarm is generated. Default is selected.
Display message	Sets whether to display the alarm message. Default is selected.
Display time	Sets whether to display the time when an alarm is generated. Default is selected.
Auto hide	Sets whether the alarm bar is displayed only when there is a message. Default is not selected.
Fault	Sets whether to display messages of error type. Default is selected.
Warning	Sets whether to display messages of warning type. Default is selected.
System	Sets whether to display messages of system type. Default is selected.
User operation	Sets whether to display user operation information. Default is selected.
Scroll speed	Sets the scroll speed of text when an alarm is generated in pixels. The configurable value ranges from 0 to 25. Default is 1.
Scroll direction	Sets the scroll direction of alarm text. Options include RTL (right to left), LTR (left to right). Default is RTL.
Alarm sequence	Sets the alarm display sequence. Options include time order (default) and time reverse order.

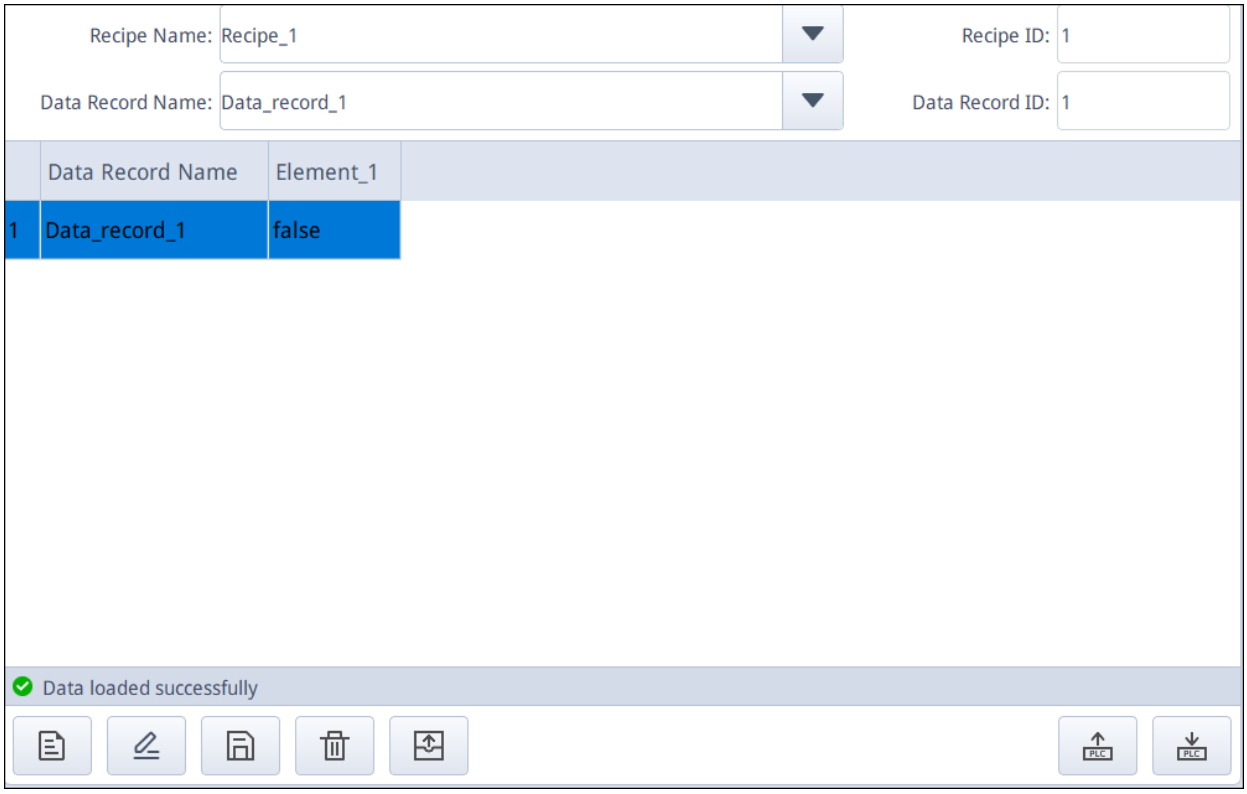
- Fill:

Property	Description
Text color	Sets the alarm bar text color.
Background color	Sets the alarm bar background color.
Fill style	Sets the alarm bar fill mode.

- Basic information: For detaIils, see Line > Basic information.
- Animation: For details, see 13.3.8 Animation Configuration.

13.3.5.4.3 Recipe View

The recipe view is a general-purpose enhanced screen object for displaying and editing recipes, and managing recipe data records. It presents data records in tables. In the recipe view, the values displayed and input can be stored in data records, which exchange data with the PLC through system features.



- **Recipe view creation**

  There are three methods to create a recipe view. For details, see the line creation method.

- **Recipe view properties**

  - For basic settings, see Numeric I/O Field > Basic settings.
  - Interface:

Property	Description
Recipe name	The recipe name can be selected. The selected recipe name can be cleared using the clear icon. Note: When selecting a recipe name, the runtime system can only operate on the currently selected recipe; when no recipe name is selected, the runtime system can operate on any recipe.
Recipe variable	Select an integer variable for the recipe number. The runtime system dynamically controls the selected recipe number by modifying the variable value, thereby controlling the selected recipe. Note: There are several ways to switch recipes: 1. Switch the recipe number by changing the variable value. 2. Switch recipes manually by selection. 3. Switch recipes by changing the recipe number.

Property	Description
Data record variables	Select an integer variable for the data record number. The runtime system dynamically controls the selected recipe data record by modifying the variable value. Note: After selecting a variable, the runtime system can select a data record by changing the variable value, or manually switch the selected data record.
Activate edit mode	Default is selected. When selected, the runtime system can edit recipe data records.

◦ Display:

Property	Description
Alternate table display	Sets whether the table displays alternately. Default is selected.
Show grid	Sets whether to display grid lines. Default is selected.
Toolbar display	Sets the buttons to display in the recipe view toolbar, including "Create data record", "Save data record", "Delete data record", "Export recipe", "Read from PLC", and "Write to PLC". By default, all buttons are displayed.

◦ Fill:

Property	Description
Background color	Sets the recipe view background color. Default is white #FFFFFF.
Text color	Sets the recipe view text color. Default is #454E66.
Header background color	Sets the header background color. Default is #E2E8F2.
Header foreground color	Sets the header foreground color. Default is black #000000.
Table background color	Sets the table background color. Default is white #FFFFFF.
Table foreground color	Sets the table foreground color. Default is #000000.
Table alternate color	Sets the alternate color for table row display. Default is #F1F4F8.
Grid line color	Sets the grid line color. Default is #CFD5D8.
Selected row color	Sets the color of selected content rows. Default is #0078D7.

◦ Text:

Property	Description
Font	Sets the font for the recipe view text. Default is Droid Sans Fallback.

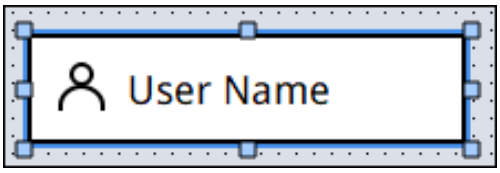
Property	Description
Font size	Sets the font size for the recipe view text. Default is 18.
Font style	Sets the font style of the recipe view text. Supports regular, bold, italic, and underline. Default is regular.

- Basic information: For details, see Line > Basic Information.
- Animation: For details, see 13.3.8 Animation Configuration.

- **Recipe view online operations**
  For the operations supported when the recipe view is running online, see 13.15.2 Online Recipe View Operations.

13.3.5.4.4 User Login

User login management control.



- **User login control creation**

  There are three methods to create a user login control. For details, see the line creation method.

- **User login control properties**
  - For basic settings, see Numerical I/O Field > Basic Settings .
  - Text:

Property	Description
Font	Sets the font for the user login text. Default is Droid Sans Fallback.
Font size	Sets the font size for the user login text. Default is 18.
Font style	Sets the font style for the user login text. Supports regular, bold, italic, and underline. Default is regular.

  - Interface:

Property	Description
Verify identity when logout	Sets whether to verify identity when a user logs out. Default is selected.
Display login role	Sets whether to display the role information of the logged-in user. Default is not selected.
Display login time	Sets whether to display the user login time. Default is not selected.

  - Fill:

Property	Description
Text color	Sets the text color of the user login control.
Background color	Sets the background color of the user login control.

- Border:

Property	Description
Border color	Sets the border color and transparency of the user login control.
Line width	Sets the border line width of the user login control. Configurable range is 1 to 10. Default is 2.
Line type	Sets the border line type of the user login control.

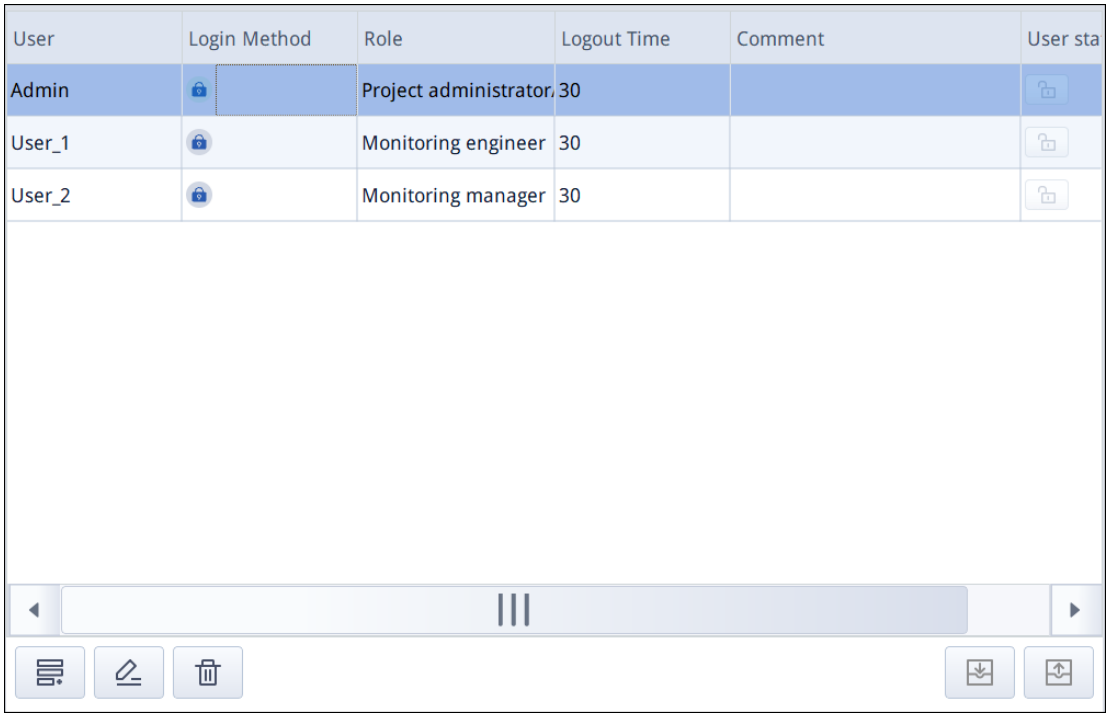
- Basic information: For details, see Line > Basic Information.
- Animation: For details, see 13.3.8 Animation Configuration.

NOTE

The complexity requirements and validation rules for user login passwords are configured and managed under "Safety settings > Settings". For related instructions, see 9.6 HMI Security Authentication Policy.

13.3.5.4.5 User View

The user view control is used to display and manage user information for the current device. When the user view is in operation mode, it manages user login information, allowing users to add/delete users and import/export user data.



- **User view creation**

There are three methods to create a user view. For details, see the line creation method.

- **User view properties**
  - For size and layout settings: see Numerical I/O Field > Basic Settings.
  - Table appearance:

Property	Description
Alternate table display	Sets whether the table row background color alternates. Default is selected.
Show grid	Sets whether to display grid lines. Default is selected.
Background color	Sets the table background color. Default is white #FFFFFF.
Header text color	Sets the table header text color. Default is black #000000.
Header background color	Sets the table header background color. Default is #E5E6EC.
Table text color	Sets the table text color. Default is black #000000.
Odd row background color	Sets the odd row background color. Default is white #FFFFFF.
Even row background color	Sets the even row background color. Default is #EDEEF1.
Grid line color	Sets the grid line color. Default is #CFD5D8.
Selected row color	Sets selected row background color. Default is #A0BBE9.

  - Table text:

Property	Description
Header text	Sets the font, size, and style of the table header text. For details, see Number I/O Filed > Text.
Table text	Sets the font, size, and style of the table text. For details, see Number I/O Filed > Text.

  - Basic information: For details, see Line > Basic Information.
  - Animation: For details, see 13.3.8 Animation Configuration.
- **User view online operations**
  For the operations supported when the user view is running online, see 13.15.5 Online User View Operations.

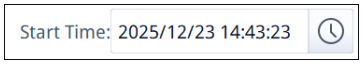
13.3.5.4.6 Variable Log View

The variable log view is used for statistics. It displays archived records of variables over a certain period of time.

- **Variable log view creation**

There are three methods to create a variable log view. For details, see the line creation method.

- **Icons in the variable log view during operation**

Item	Icon	Description
Starting time		Variable start time (default is 1 minute before the current system time)
End time		Variable end time (default is the current system time). Note: The end time cannot be earlier than or equal to the start time.
Query		Click to query the variables within the range of start time and end time.
Configuration		Sets whether to display configured fields in edit mode during runtime.
Statistics		Displays statistical query results (including the list of queried variables, maximum value, minimum value, average value, and cumulative value).
Export		Exports the queried data.
Ascending		Sorts query results in ascending order based on the time field.
Descending		Sorts query results in descending order based on the time field.
Back		Goes to the previous page of query results. This icon is grayed out when the current page is the first page.
Next		Goes to the next page of query results. This icon is grayed out when the current page is the last page.

- **Variable log view properties**
  - For basic settings, see Numerical I/O Field > Basic Settings .
  - Text:

Property	Description
Header font	Sets the header font. Default is Droid Sans Fallback.
Header font size	Sets the header font size. Default is 18.

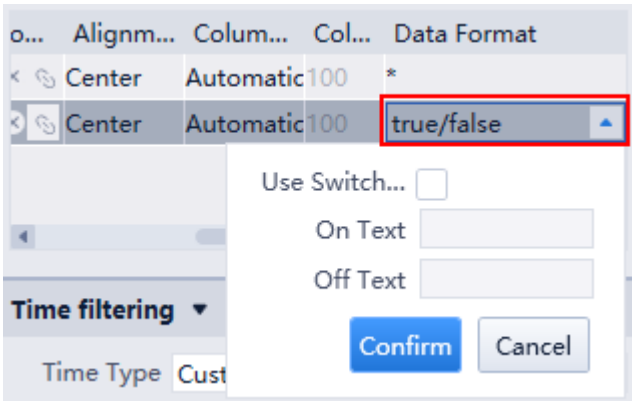
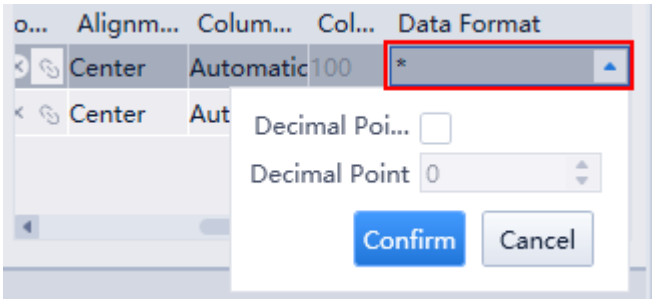
Property	Description
Header font style	Sets the header font style. Supports regular, bold, italic, and underline. Default is regular.
Content font	Sets the font for the content row. Default is Droid Sans Fallback.
Content font size	Sets the font size for the content row. Default is 18.
Content font style	Sets the font style for the content row. Supports regular, bold, italic, and underline. Default is regular.

◦ Interface:

Property	Description
Data record	Select the data record to query. All variables associated with the fields in the table belong to this data record.
Statistics	Sets whether the statistics function is available at runtime. Default is selected.
Refresh interval	Sets the automatic refresh interval (in seconds) for displaying historical data online. This setting takes effect when "Time type" is set to "All records" or "Fixed time". Range: 0 to 65535. 0 indicates historical data is refreshed only once when the screen is opened, with no further automatic refresh.

◦ Record variables:

Property	Description
Name	Name of the added field. Default is Field x (x starts from 1 and increments automatically). Double-click to modify the name.
Variable recording	Associated variable.
Alignment	Alignment of field content. Supports left alignment, center alignment (default), and right alignment.
Column width adjustment method	Supports automatic and manual adjustment. Default is automatic.
Column width	When column width adjustment method is set to manual, column width can be set. The value range is 5 to 5000. Default is 100.

Property	Description
Data format	Sets the data display format. Only supported when the field-associated variable is of BOOL, REAL, or LREAL type. • When the field-associated variable is of BOOL type, the following options are displayed. Users can set the text to show when the variable is True or False.  • When the field-associated variable is of REAL or LREAL type, the following options are displayed to configure the formatted display of the data. If not configured, the value is displayed according to the default settings. The decimal place range is 0 to 7, with a default value of 0.  • When the field-associated variable is of time type, display format is yyyy-MM-dd hh:mm:ss (not editable).

◦ Time filtering:

Property	Description
Time type	Sets the time conditions for historical data filtering. Options include all records, fixed time, and custom time (default). • All records: In runtime, displays data from all records of the statistical variables. • Fixed time: In runtime, displays data from records within the configured time range. Fixed time can be either a constant or a variable, with a maximum supported range of 2 years. ◦ For a constant, seconds, minutes, hours, days, weeks, and months are supported, with a default value of 60 minutes. For a variable, unsigned integer variables are supported. If the runtime variable value exceeds the 2-year range, statistics are calculated based on the maximum 2-year range. • Custom time: The custom time can be set by the default time or variables. ◦ Default time: In runtime, the query range is set based on the start and end time setting controls in the record table. ◦ Variables: The start and end time of the query range are associated with specific time variables.

◦ Column settings:

Property	Description
Time	Sets whether the time column is displayed for runtime data. Default is selected.
ID	Sets whether the ID column is displayed for runtime data. Selected by default.
Header row height	Sets the header row height for runtime data. Default is 28.
Content row height	Sets the content row height for runtime data. Default is 42.

◦ Display: Sets whether to display the grid lines.

◦ Fill:

Property	Description
Background color	Set the background color of control. Default is white #FFFFFF.
Text color	Sets the text color. Default is black #000000.
Header text color	Sets the table header text color. Default is black #000000.
Header background color	Sets the table header background color. Default is #E5E6EC.
Text color	Sets the table text color. Default is black #000000.
Table background color	Set the table background color. Default is white #FFFFFF.
Alternate background color	Set the alternate background color. Default is gray #F1F4F8.

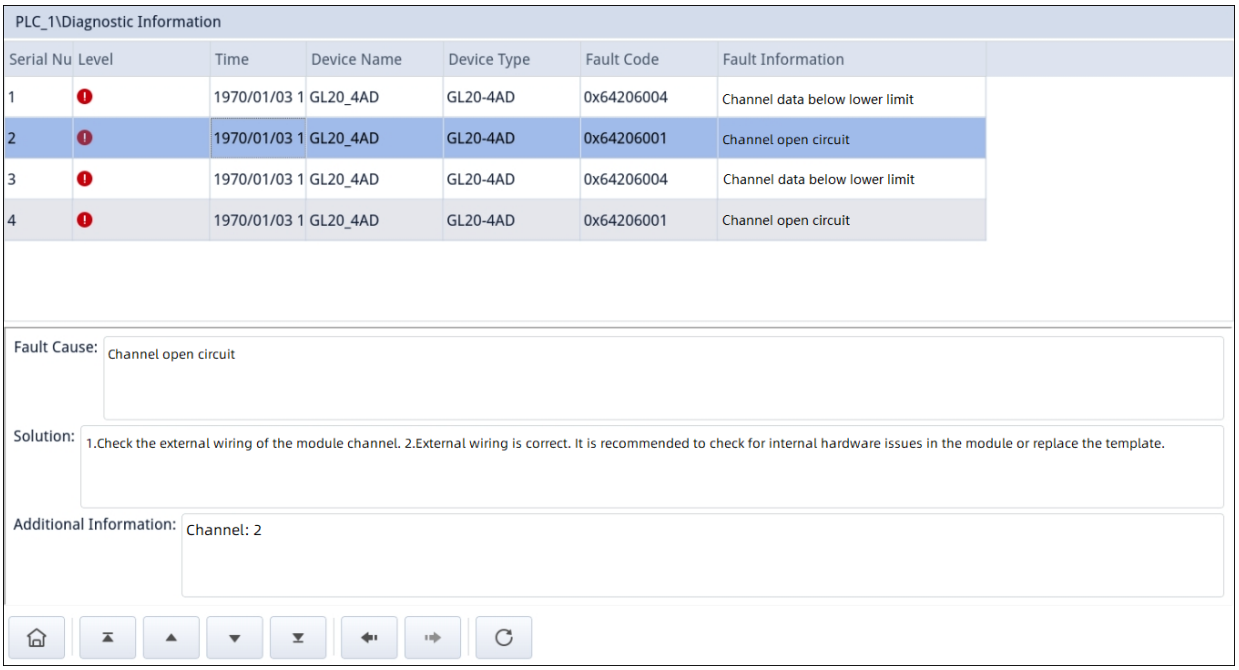
Property	Description
Grid line color	Sets the grid line color of the table. Default is gray #E5E6EC.
Selected row color	Sets selected row background color. Default is #A0BBE9.

- Basic information: For details, see Line > Basic Information.
- Animation: For details, see 13.3.8 Animation Configuration.

- **Record variable table online operation**
  For the operations supported by the record variable table during online runtime, see 13.15.3 Online Historical Data Viewing.

13.3.5.4.7 System Diagnostics

The system diagnostics control is used to display the PLC operating status, as well as diagnostic information for I/O modules and coupler modules. The diagnostic information displayed is synchronized with the content under the 10.11.20 "Online and diagnostics" node in the PLC device.



- **System diagnostics control creation**

  There are three methods to create a system diagnostics control. For details, see the line creation method.

- **System diagnostics control properties**
  - For basic settings, see Numerical I/O Field > Basic Settings .
  - Column settings:

Property	Description
Column setting	Sets the column content and column width of the system diagnostics table.
Header row height	Sets the header row height of the system diagnostics table.
Content row height	Sets the content row height of the system diagnostics table.

  - Fill:

Property	Description
Header text color	Sets the table header text color. Default is black #000000.
Header background color	Sets the table header background color. Default is #E5E6EC.
Header text color	Sets the table header text color. Default is black #000000.
Header background color	Sets the table header background color. Default is #E5E6EC.
Content text color	Sets the table content text color. Default is black #000000.
Content background color	Set the table content background color. Default is white #FFFFFF.
Selected row color	Sets selected row background color. Default is #A0BBE9.
Grid line color	Sets the grid line color of the table. Default is gray #CFD5D8.

  - Text properties:

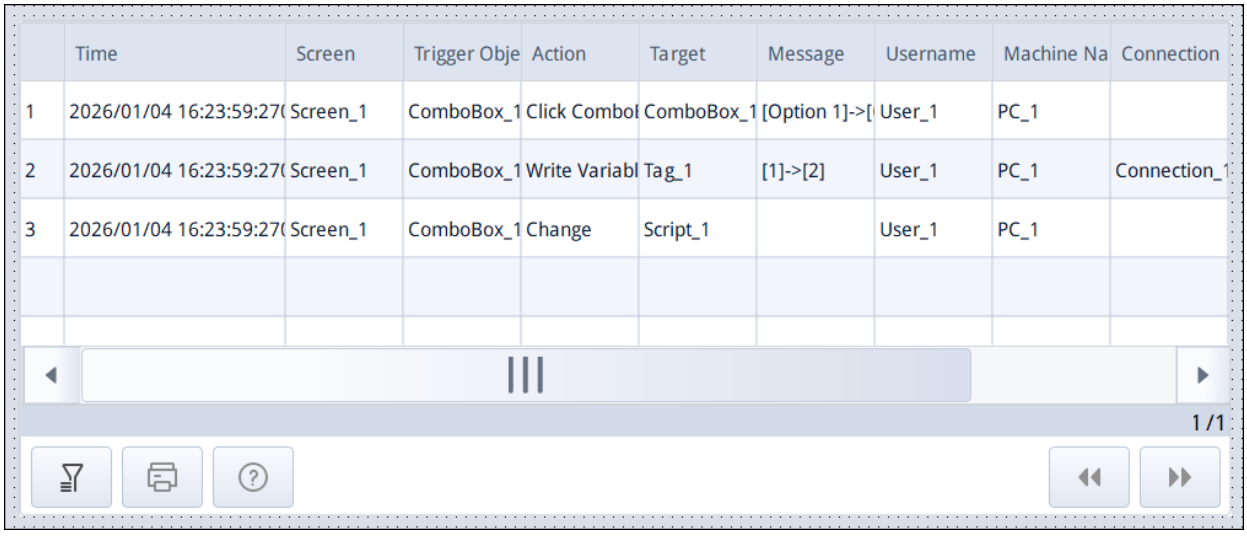
Property	Description
Title text	Sets the font, size, and style of the title text.
Header text	Sets the font, size, and style of the header text.
Content text	Sets the font, size, and style of the content text.

  - Basic information: For details, see Line > Basic information.
  - Animation: For details, see 13.3.8 Animation Configuration.

- **System diagnostics online operation**
  For the operations supported by the system diagnostics control during runtime, see 13.15.7 Online System Diagnostics.

13.3.5.4.8 Operation Records

The operation log is used to display user operation logs and can be synchronously updated in real time.



- **Operation log creation**

  There are three methods to create an operation log. For details, see the line creation method.

- **Operation log properties:**

  - For basic settings, see Numerical I/O Field > Basic Settings .
  - Column settings:

Item	Description
Visible	Controls the visibility of corresponding columns in the view window.
Name	Default column name, not editable.
Width	Sets the column width with a range of 20 to 1000.
Display name	Column name displayed in the view window, with a length of 0 to 256. Default name is displayed when the display name is empty.

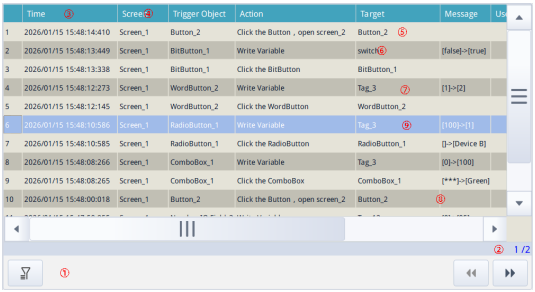
  - Display:

    Sets the table style and data arrangement in the view window.

Property	Description
Column header	Sets the visibility of the table header.
Toolbar	Sets the visibility of the toolbar.
Pagination bar	Sets the visibility of page numbers.
Time (ms)	Sets whether operation time is displayed to the millisecond level. When selected, the content is displayed as 2025/05/30 14:34:39:059 ; when not selected, the content is displayed as 2025/05/30 14:34:39.
Show grid	Sets the visibility of grid lines in the operation log table.
Refresh interval (s)	Data refresh interface, with a range of 2 to 65535.
Sort	Default sorting method for operation logs.
Date range	Default display range for operation logs.
Header row height	Height of the table header row.
Content row height	Height of the table content rows.

◦ Function:
Sets the function icons displayed in the view window.

◦ Fill:
Sets the color of the view window.

Property	Description	
Background color	Sets the background color of the area outside the operation log table in the view window, as shown in position ①.	
Text color	Page navigation text color, as shown in position ②.	
Header background color	Table header background color, as shown in position ③.	
Header foreground color	Table header text color, as shown in position ④.	
Table background color	Table background color, as shown in position ⑤.	
Table foreground color	Table text color, as shown in position ⑥.	
Alternate background color	Alternate background color of the table, as shown in position ⑦.	
Grid line color	Inner border line color of the table, as shown in position ⑧.	
Selected row color	Selected row color of the table, as shown in position ⑨.	

- Text:
  Sets the font, size, and style for header text and content text.
- Basic information: For details, see Line > Basic Information.
- Animation: For details, see 13.3.8 Animation Configuration.

13.3.5.4.9 SFC View

The "SFC view" control is used to display the current execution status of a sequential control program in real time. If an error occurs during execution, the system indicates the exception at the corresponding step. SFC view needs to be used with the 13.3.5.4.10 code view control.



- **Requirements**
  - The PLC has at least one instance of a sequential control program. The PLC and HMI are in the same project.
  - The network public property of the instance variables of the sequential control program is "Public".
- **Creating an SFC view**

  There are three methods to create an SFC view. For details, see Line creation method.
- **SFC view property**
  - Basic settings:

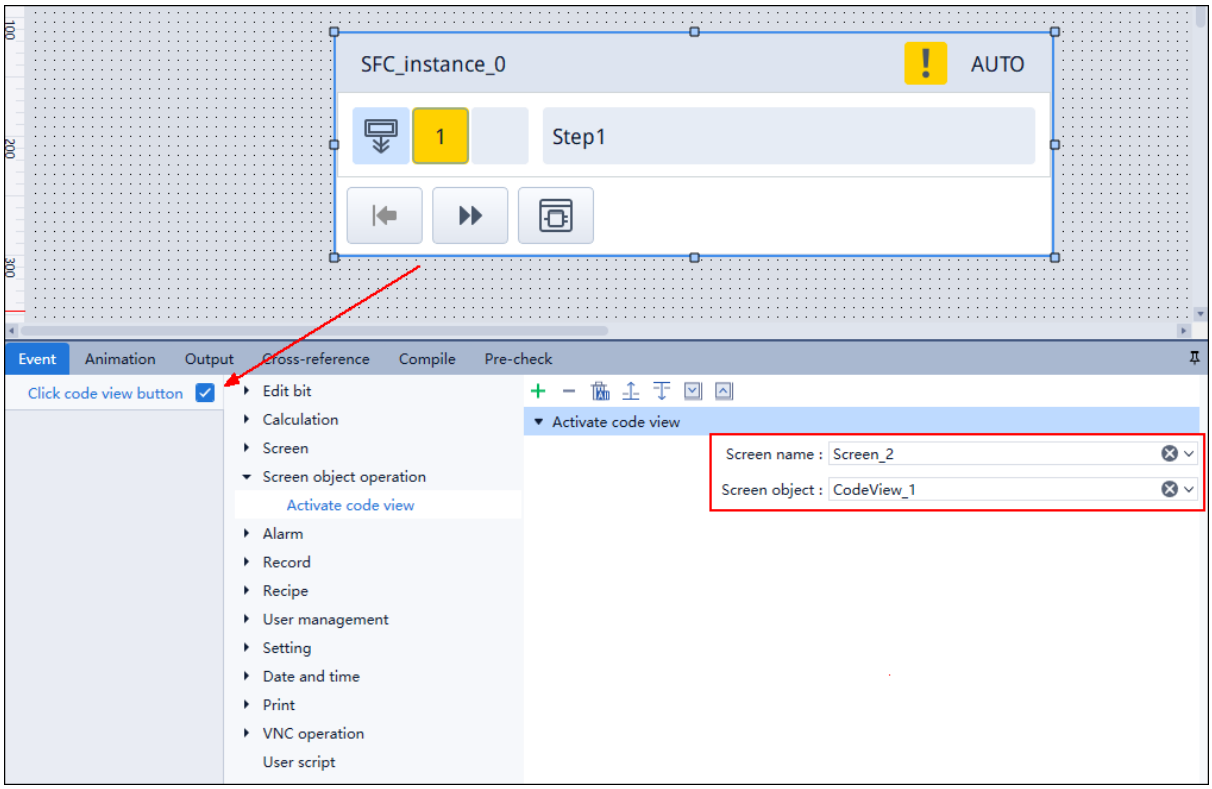
Property	Description
Position	Sets the position coordinates (i.e., top-left coordinates) of the element on the screen in pixels: X: X-axis coordinate. Y: Y-axis position coordinate. The coordinate value must be smaller than the screen boundary coordinate value to prevent the elements from extending beyond the screen.
Dimension	Sets the size of the element on the screen in pixels. Width range: 1 to 10000 Height is fixed and cannot be modified.

  - Variable:
    Process variable: Associates a sequential control program instance from the PLC.
  - Fill:
    Sets the colors for various display objects in the control view (e.g., background color, text color).
  - Text:
    Sets the font and font size for the table header and step description text.
  - Basic information: For details, see Line > Basic Information.
- **Associating a 13.3.5.4.10 code view control**

  The SFC view and code view must be used together. Both must be placed on a process screen. One SFC view can be associated with multiple code views. The related operations are as follows.

  1. Add a code view control to the current view.
  2. Double-click the SFC view control to open the "Event" tab.
  3. Associate a code view.

Add a specific event and associate the current screen and the code view instance.



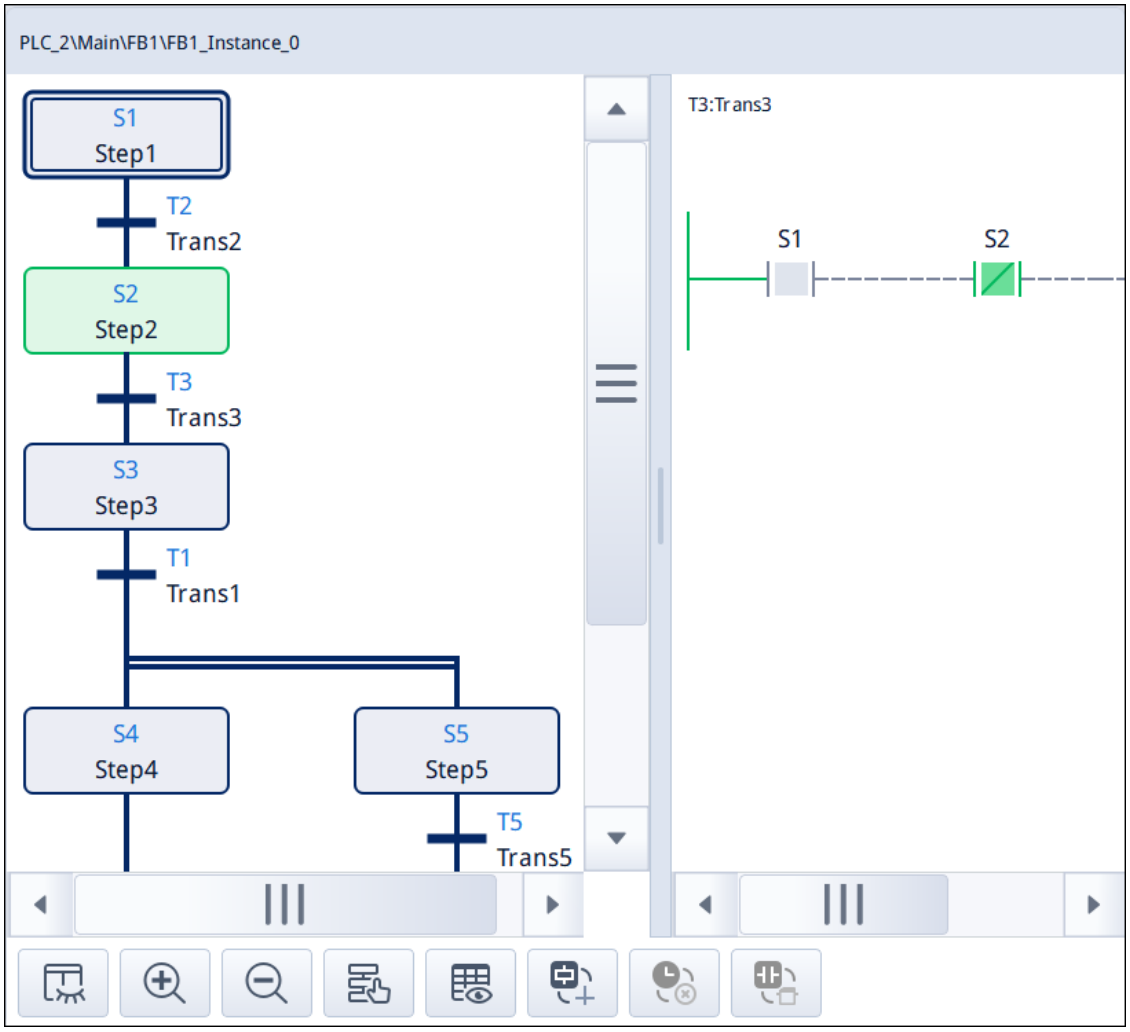
The SFC view and code view controls are successfully associated.

- **SFC program online operation**
  For details on the online interaction between the SFC view and code view, see 13.15.8 Online PLC Program Viewing.

13.3.5.4.10 Code View

The code view control is used to dynamically display the sequential control program (in SFC and LD languages) of the PLC during runtime. By synchronizing real-time data, it shows the execution status of the program logic. Users can intuitively monitor the running status of the sequential control program for easier commissioning and diagnostics.

The code view must be used with the 13.3.5.4.9 SFC view control.



- **Requirement**
  An SFC view control has been added, and its "Process variable" property is associated with a valid variable.
- **Creating a code view**
  There are three methods to create a code view. For details, see the line creation method.
- **Code view properties**
  - Basic settings:

Property	Description
Position	Sets the position coordinates (i.e., top-left coordinates) of the element on the screen in pixels: X: X-axis coordinate. Y: Y-axis position coordinate. The coordinate value must be smaller than the screen boundary coordinate value to prevent the elements from extending beyond the screen.
Dimension	Sets the size of the element on the screen in pixels. Width range: 1 to 10000 Height range: 132 to 10000

Property	Description
Detail view width	Column width ratio of the detail view display area.

- Fill:

  Sets the colors for various display objects in the control view (e.g., background color, text color).

- Text:

  Sets the font and font size for the table header and step description text.

- Symbol table:

  Displays the internal variable information that activates transition condition in table form.

Property	Description
Item	Information about variables in the transition condition. Symbol name: variable name Absolute address: storage address of a variable Description: Variable description
Displayed row count	Number of rows displayed in the table online. Range: 1 to 20
Grid line	Sets the visibility of grid lines in the table.

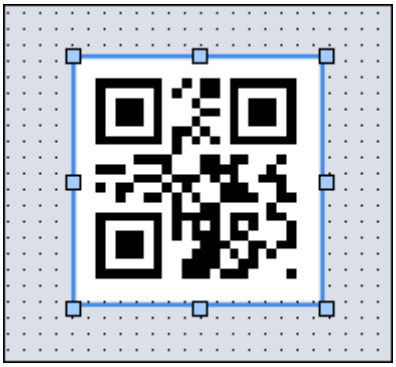
- **Associating an SFC view control**
  The SFC view and code view must be used together. Both must be placed on a process screen. One SFC view can be associated with multiple code views. For related operations, see SFC View.
- **SFC program online operation**
  For details on the online interaction between the SFC view and code view, see 13.15.8 Online PLC Program Viewing.

13.3.5.5 Others

- 13.3.5.5.1 QR Code
- 13.3.5.5.2 Barcode
- 13.3.5.5.3 PDF
- 13.3.5.5.4 Clock
- 13.3.5.5.5 Browser

13.3.5.5.1 QR Code

The QR code control encodes the associated variable's real-time value into a QR graphic for scanning.



- **QR code creation**

  There are three methods to create a QR code. For details, see the line creation method.

- **QR code properties**

  - For basic settings, see Numeric I/O Field > Basic Settings.
  - Data:

Property	Description
Fault tolerance	Sets the fault tolerance level of the QR code. The required fault tolerance level depends on the expected damages the QR code may endure in its intended environment. Options include: L (7%), M (15%), Q (25%), and H (30%). The percentages in parentheses indicate the maximum damage ratio. Default: H (30%)
Encoding	Sets the parsing type of the QR code. Currently supported types are: UTF-8 (default), ISO-8859-1, and GBK.

  - Interface:

Property	Description
Static value	Static text: At runtime, the QR code corresponding to the static text is displayed. Default is "qrcode"; other text can be entered.
Read variable	Supports basic variable types. At runtime, the QR code corresponding to the variable value is displayed. For details, see 13.2.5.3 Setting Variable Properties.

  - Fill:

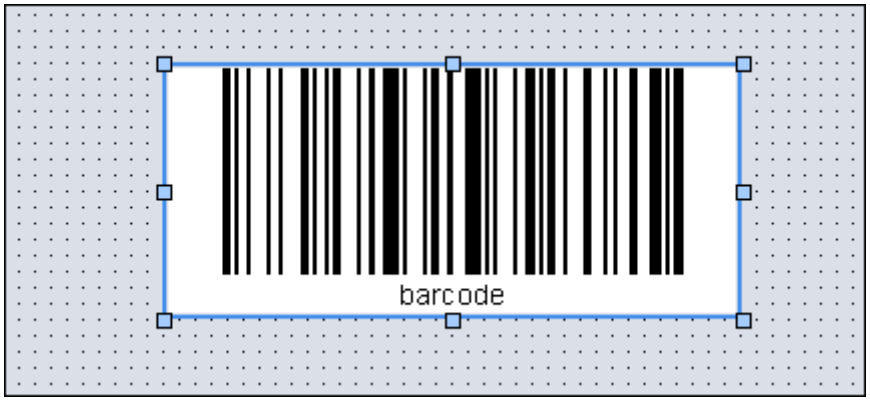
Property	Description
Foreground color	Sets the color of the QR code. Default is black.
Background color	Sets the background color of the QR code. Default is white. Sets the transparency of the QR code background (range: 0% to 100%).
Text transparency	Sets the transparency of the QR code from 0% to 100%. Default is 0%. Integers and decimals can be entered.

Property	Description
Border margin	Sets the border margin of the QR code from 1 to 4 pixels. Default is 2 pixels.

- Basic information: For details, see Line > Basic Information.
- Animation: For details, see 13.3.8 Animation Configuration.

13.3.5.5.2 Barcode

The barcode control encodes the associated variable's real-time value into a barcode graphic for scanning.



- **Barcode creation**

  There are three methods to create a barcode. For details, see the line creation method.

- **Barcode properties**
  - For basic settings, see Numerical I/O Field > Basic Settings .
  - Text:

Property	Description
Text display	Sets whether to display text. Default is selected.
Text position	Sets the position of the text, either at the bottom or top. If text is not displayed, this option is grayed out and cannot be edited.

  - Barcode type:

Property	Description
Barcode type	Sets the decoding type of the barcode, supporting Code128.

  - Interface:

Property	Description
Static value	Static text: Conforms to the character limits of Code128. It consists of uppercase letters, numbers, and control characters (such as #, /, -). A maximum of 9 characters is supported. The barcode corresponding to the static text is displayed during runtime.

Property	Description
Read variable	Supports basic variable types. At runtime, the barcode corresponding to the variable value is displayed. For details, see 13.2.5.3 Setting Variable Properties.

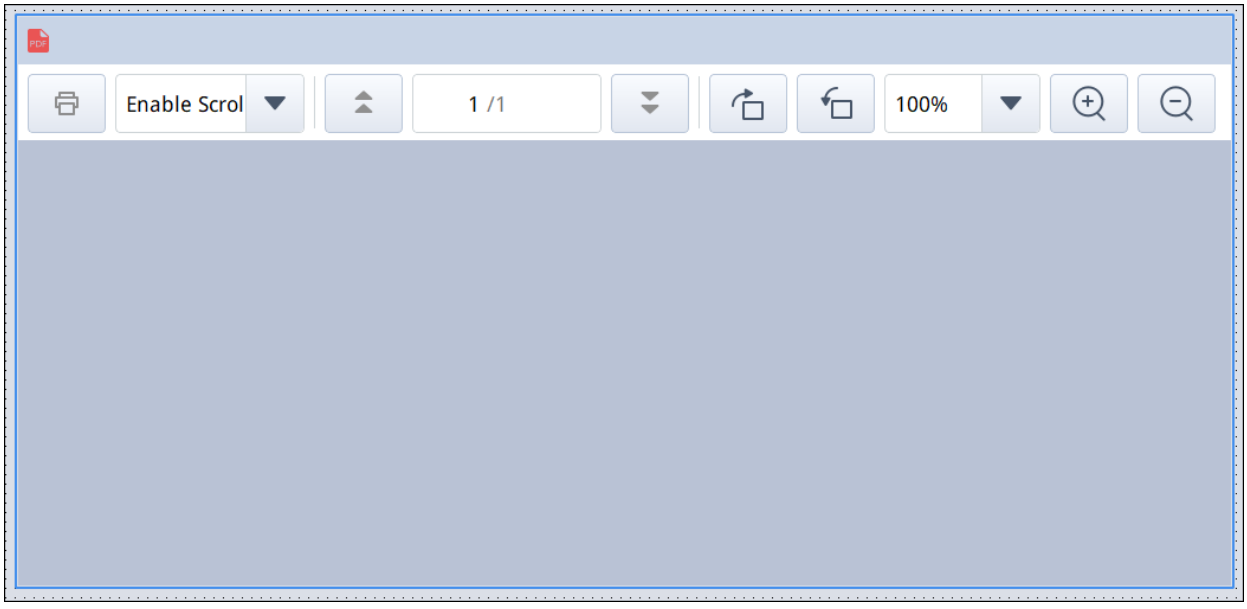
- Fill properties:

Property	Description
Text color	Sets the color of the barcode. Default is black.
Background color	Sets the barcode background color. Default is white. Sets the background transparency level from 0% to 100%. Default is 0%.
Text transparency	Sets the transparency of the barcode from 0% to 100%. Default is 0%. Integers and decimals can be entered.

- Basic information: For details, see Line > Basic Information.
- Animation: For details, see 13.3.8 Animation Configuration.

13.3.5.5.3 PDF

The PDF control is used to display PDF documents.

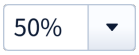


- **PDF creation**

  There are three methods to create a PDF control. For details, see the line creation method.

- **Icons of the PDF control during operation**

Item	Icon	Description
Column header		Displays the PDF file name. Default is empty.
Print		Prints the file.

Item	Icon	Description
Page display (drop-down box)		Sets single-page view or enables scrolling mode for PDF display. Default is scrolling mode.
Back		Displays the previous PDF page. Disabled when the current page is the first page.
Current page / Total pages		Displays the current page. You can enter a page number and press Enter to jump to the specified page. A prompt is displayed if the entered number exceeds the total number of pages.
Next		Displays the next PDF page. Disabled when the current page is the last page.
Clockwise rotation		Rotates all PDF pages clockwise (zoom ratio remains unchanged). Click once to rotate 90 degrees clockwise.
Counterclockwise rotation		Rotates all PDF pages counterclockwise (zoom ratio remains unchanged). Click once to rotate 90 degrees counterclockwise.
Scale ratio		Selects the PDF display ratio from the drop-down menu. Options include 10%, 25%, 50%, 75%, 100%, 125%, 150%, 200%, and 300%.
Zoom in		Zooms in on the entire PDF. Click once to zoom in one level. Options include 10%, 25%, 50%, 75%, 100%, 125%, 150%, 200%, and 300%.
Zoom out		Zooms out on the entire PDF. Click once to zoom out one level. Options include 10%, 25%, 50%, 75%, 100%, 125%, 150%, 200%, and 300%.
Vertical scroll bar		Automatically follows the vertical position of the currently displayed page.
Horizontal scroll bar		Automatically follows the horizontal position of the currently displayed page.

- **PDF properties**
  - For basic settings, see Numerical I/O Field > Basic Settings .
  - Interface properties:

Property	Description
Path source	Sets the source path of the PDF, supporting a constant and a variable.

Property	Description
Storage location	Sets the storage location of the PDF. Options include SD card, USB1, and USB2.
Constant path	Enter the absolute path where the PDF file is stored, without including the root directory. The specific filename (including the .pdf extension) must be included.
Variable path	Select a STRING type variable associated with the PDF path.

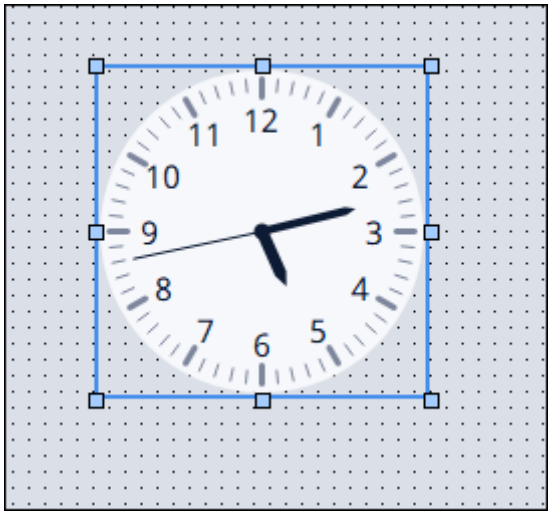
- Function configuration:

Property	Description
File selection	Sets whether to allow selecting a specific PDF file for browsing at runtime. Default is not selected.
Print	Sets whether PDF supports printing. Default is selected.

- Basic information: For details, see Line > Basic Information.
- Animation: For details, see 13.3.8 Animation Configuration.

13.3.5.5.4 Clock

The clock control is used to display the current system time.



- **Clock creation**

  There are three methods to create a clock. For details, see the line creation method.

- **Clock properties**
  - For basic settings, see Numerical I/O Field > Basic settings .
  - Properties:

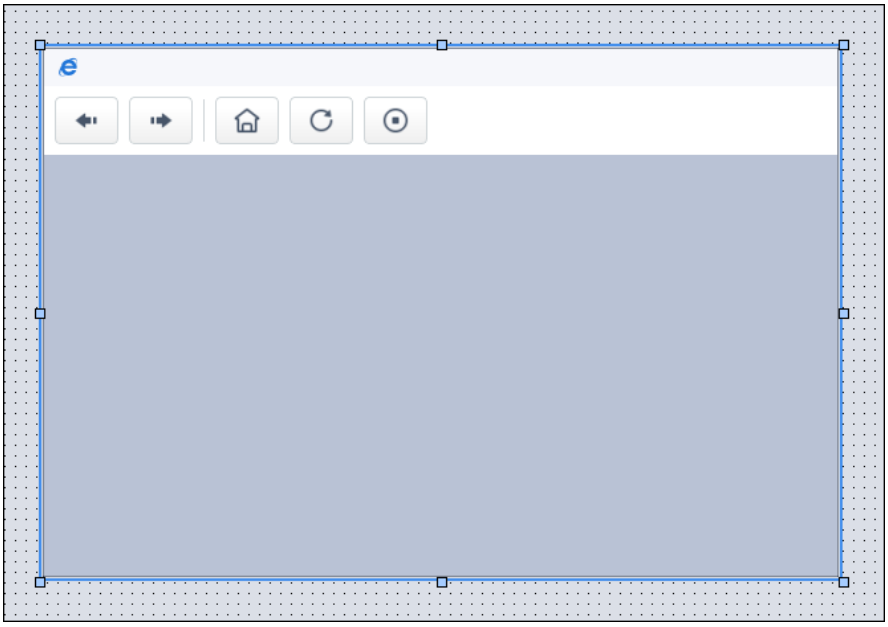
Property	Description
Text color	Sets the clock text color and transparency (0% to 100%).

Property	Description
Background color	Sets the clock background color and transparency (0% to 100%).
Tick color	Sets the clock tick color and transparency (0% to 100%).
Pointer fill color	Sets the fill colors and transparency (0% to 100%) for the hour, minute, and second hands.
Font	Sets the clock text font. Default is Droid Sans Fallback.
Font size	Sets the clock text size. Default is 16.
Font style	Sets the clock text font style. Supports regular, bold, italic, and underline. Default is regular.

- Basic information: Refer to Line > Basic information.
- Animation: For details, see 13.3.8 Animation Configuration.

13.3.5.5.5 Browser

The browser control is used to display the page corresponding to a specified URL link. At runtime, it functions as an embedded browser, allowing users to interact with the displayed page.



- **Browser creation**

  There are three methods to create a browser. For details, see the line creation method.

- **Icons of the browser during operation**

Item	Icon	Description
Back		Switches to the previous page. If the current page is the home page, this icon is grayed out.

Item	Icon	Description
Forward		Switches to the next page. The operation takes effect only if the next page is cached; otherwise, this icon is grayed out.
Refresh		Reloads the current URL page.
Stop		Stops loading the current URL page.
Homepage		Returns to the URL link page set in edit mode.

- **Browser properties**
  - For basic settings, see Numerical I/O Field > Basic Settings .
  - Interface properties:

Property	Description
URL	Sets the static URL address of the page to access. Default is empty. The length must not exceed 4096 characters.
URL variable	URL variables can only be of STRING type. • If the URL variable is not set, the static URL takes effect. • If the URL variable is set, the URL variable takes effect.
Toolbar	By default, the toolbar is selected during runtime. 

  - Border properties:

Property	Description
Border	Default is selected to enable the border function.
Border width	Default is 1, with a value range of 0 to 10. It can be entered manually, or adjusted by clicking the up/down icons. When "Border" is not selected, the border width is grayed out and cannot be edited. When "Border" is selected, the border width can be edited.
Color value	Sets the border color. Default is black. When "Border" is not selected, the border color is grayed out and cannot be edited. When "Border" is selected, the border color can be edited.

  - Basic information: For details, see Line > Basic Information.
  - Animation: For details, see 13.3.8 Animation Configuration.

13.3.5.5.6 Media Player

The media player control is used to display the page corresponding to a specified URL link. At runtime, it functions as an embedded browser, allowing users to interact with the displayed page.



Supported file formats include MP4, MKV, AVI, MOV, FLV, ASF/WMV, WAV, MP3, FLAC, and AAC.

Use restrictions

- Applicable screen: Only supported in the main screen.
- Configuration count: Only one media player can be added per main screen.

Control configuration

Media player creation

There are three methods to create a media player. For details, see the line creation method.

CAUTION

- When arranging the screen, other controls or graphic elements (including those on higher layers) must not obscure the display area of the multimedia control.
- Only pop-ups triggered at runtime are allowed to temporarily overlay the multimedia control.

Media player properties

- **Basic settings**

Item	Description
Position	Sets the position coordinates (top-left corner) of the element on the screen in pixels. The coordinate values must be within the screen boundaries to prevent the element from extending beyond the visible area.
Size	Sets the size of the element on the screen in pixels. Minimum size (W*H): 336*131.

- **Interface property**

Property	Description
Path source	Sets the input method for the path. Constant: A fixed path. Variable: A variable path.
Storage location	Sets the path where the media file is stored.
Constant path	Enter an absolute path.
Variable path	Select a variable of STRING/WSTRING type that is associated with the media file. The file path is determined by the variable value. An absolute path must be entered.

- **Playback control**

  The playback mode and volume are controlled via variables.

Property	Description	Remarks
Play/Pause	Supports BOOL type variables. 0: Pause; 1: Play	If no variable is associated, the video is controlled by the icons during runtime.
Replay control	Supports BOOL type variables. 0 -> 1: Replay 1 -> 0: No operation	
Loop playback control	Supports INT type variables. 0: No loop 1: Loop single file 2: Loop folder contents	
Volume control	Supports INT type variables. The value range is 0 to 100, corresponding to minimum and maximum volume.	

- **Output Parameters**

Property	Description
Current file name	Writes the name of the currently playing file into the specified variable. Supports STRING/WSTRING types.
Playback status	Associates with an INT type variable to feed back video playback status to the variable. 0: Playing 1: Paused 2: Playback completed

- **Playback settings**

Property	Description
Resume mode	Sets whether the video resumes from the last playback position after switching screens. Resume from current position: Continues from where playback left off before the screen switch. Replay: Restarts playback from the beginning of the video.
Play mode	Loop disabled: Plays the current file and stops at the end. Folder loop: Plays all videos in the same folder sequentially and repeats. Single file loop: Repeats the current file continuously.
Default volume	Sets the volume used when the control is online.

- **Functions**
  Sets the operation icons on the video control.
- **Fill**
  Sets the background color and display angle of the video control.
- **Basic information**: For details, see Line > Basic Information.
- **Animation**: For details, see 13.3.8 Animation Configuration.
- **Event**: For details, see 13.3.7.14 Adding an Event.

13.3.6 Gallery Usage

- 13.3.6.1 Overview
- 13.3.6.4 Project Gallery Usage
- 13.3.6.2 System Gallery Usage
- 13.3.6.3 User Gallery Usage
- 13.3.6.5 Local Gallery Usage

13.3.6.1 Overview

Gallery contains a wealth of graphic resources to enhance the intuitiveness and professionalism of process screens. Based on the graphic sources, there are project gallery, system gallery, and user gallery.

Project gallery

Graphics used within a project are automatically added to the project gallery for unified management. The project gallery can be used across all HMI devices in the current project.

System gallery

The system gallery is installed along with the software monitoring development environment. Its path follows the that of the development environment and cannot be specified. The main categories in the system gallery include: buttons, indicators, alarm indicators, pipes, valves, elbows, gas pipes, gas pipe connectors, fans, water pumps, motors, carriers, etc.

User gallery

The user gallery is a user-defined gallery that supports importing images in formats (bmp, ico, gif, tiff, svg, jpeg, jpg, png) that meet the requirements, and using them when creating screens.

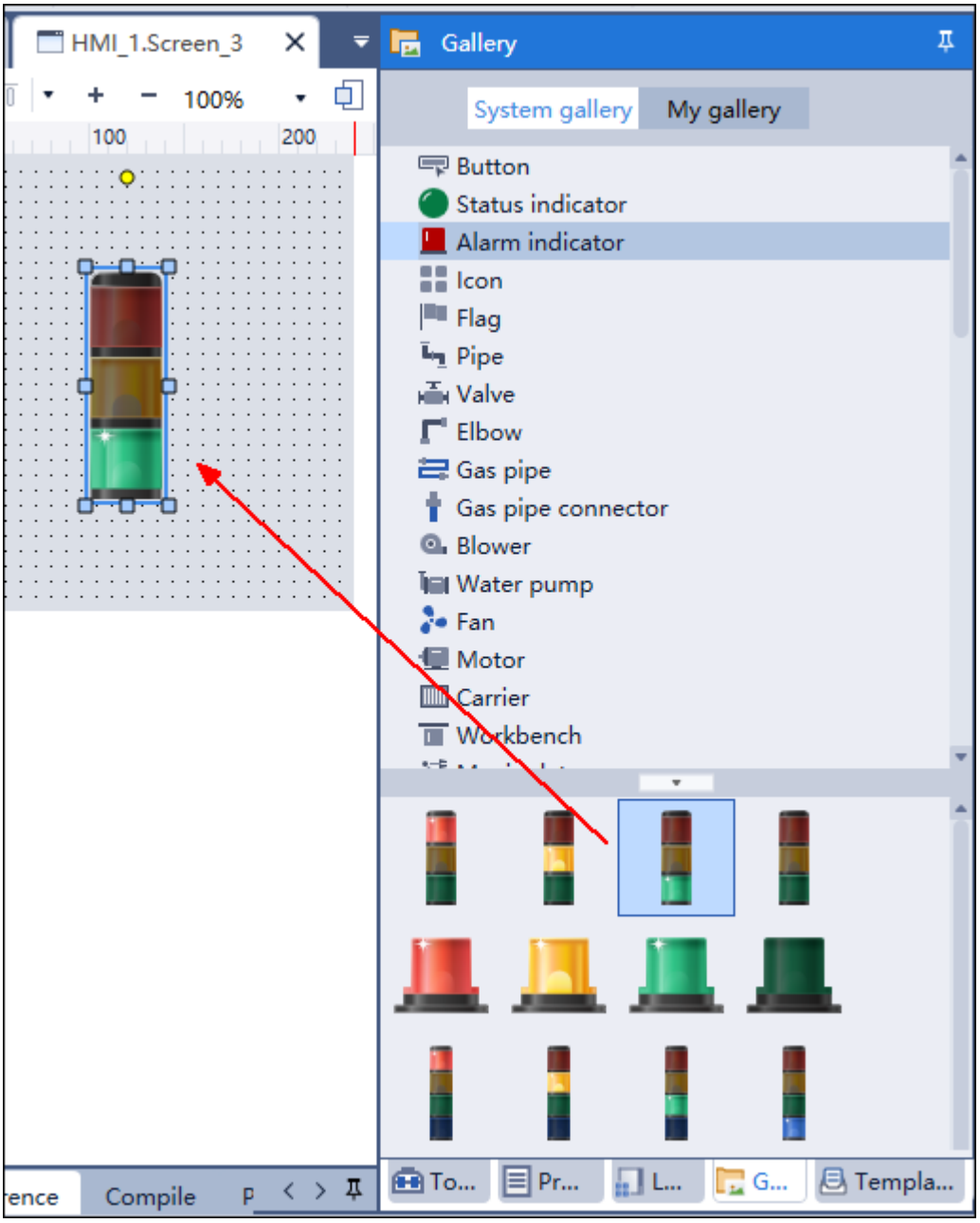
Local gallery

The local gallery is used to store image resources referenced in symbol templates. Each symbol template has its own local gallery, where all images referenced by all versions of that template are stored. These galleries are independent of each other.

13.3.6.2 System Gallery Usage

Procedure

1. Select an image from the gallery and directly drag it into the screen. The width and height of the image are its actual size in the gallery (including SVG format). Once placed, its appearance and size can be adjusted directly using the mouse.
   - To replace an image, select the new image from the gallery and drag it over the original one to overwrite it.
   - To overlap objects from different galleries, place the second image near the original one, and adjust its position for stacking.



NOTE

- During software uninstallation, the system gallery is removed together with the software. User-defined galleries are unaffected by software uninstallation and remain in their original directories.
- During software updates, user-defined gallery settings are preserved. Whether the system gallery is updated depends on whether the update package contains changes to the system gallery.

2. Click a node in the system gallery to display the gallery information under each category:

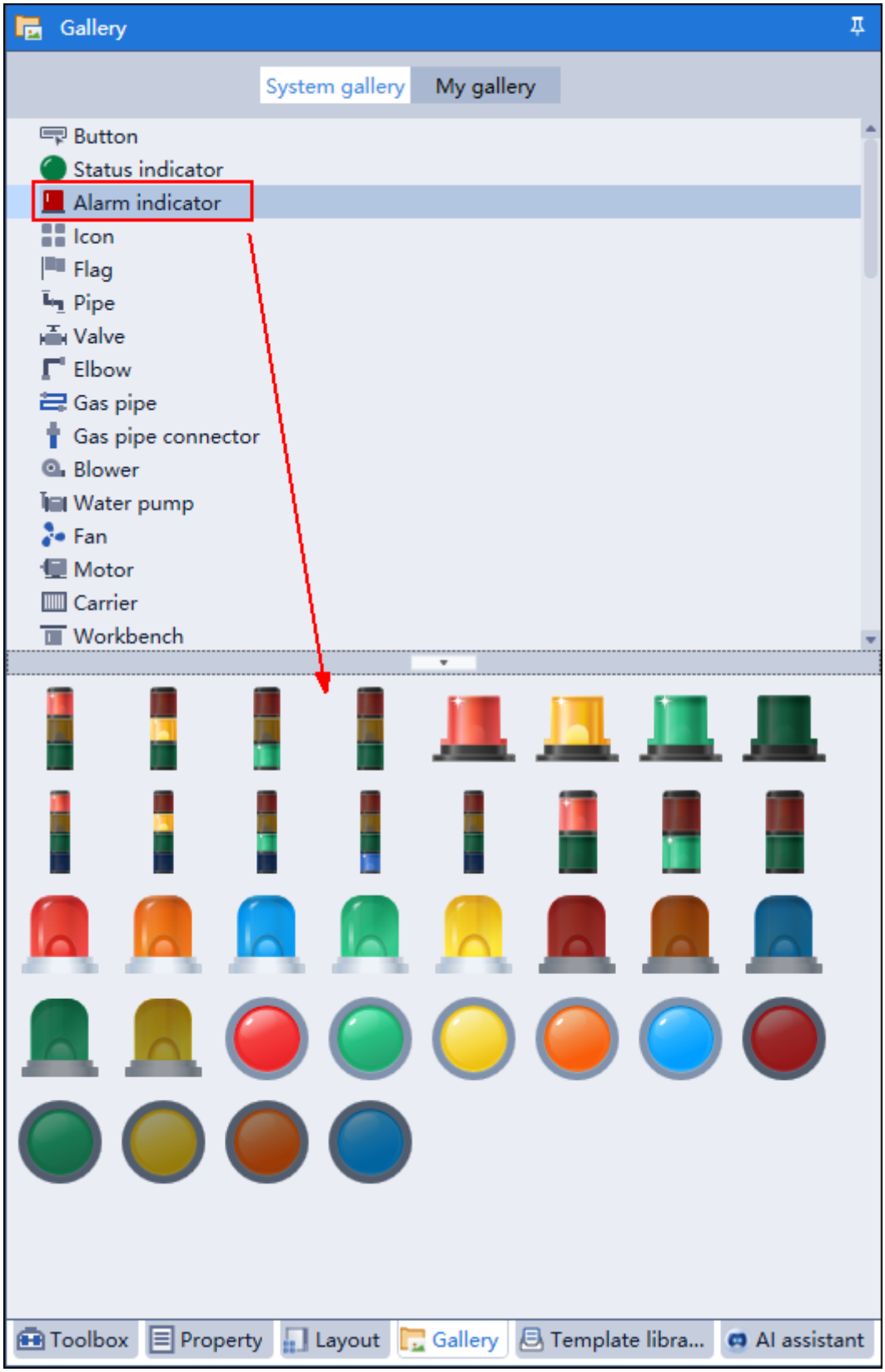


Image properties

- Basic settings

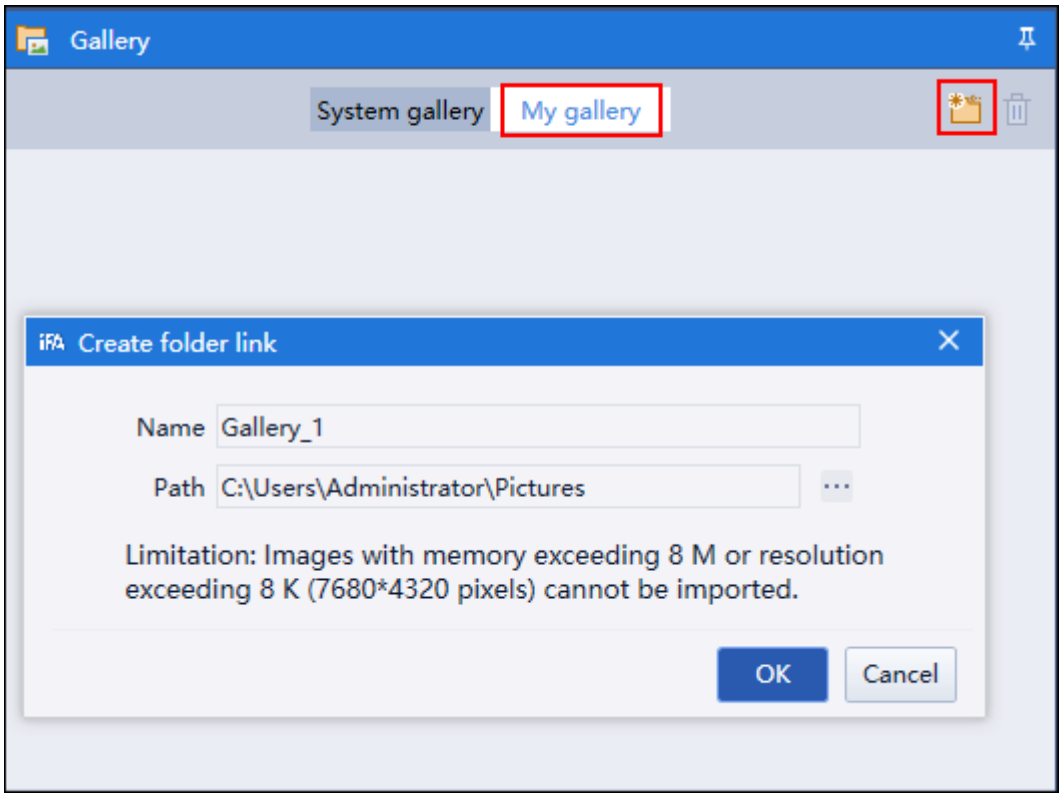
Item	Description
Position	Sets the position coordinates (i.e., top-left coordinates) of the element on the screen in pixels: • X: X-axis coordinate. • Y: Y-axis position coordinate. The coordinate value must be smaller than the screen boundary coordinate value to prevent the elements from extending beyond the screen.
Size	Sets the size of the graphic element on the screen in pixels: • W: Element width. • H: Element height. The minimum size for the graphic element is 1*1.
Rotation	Sets the rotation angle of the graphic element, with a range from -180 to 180.

- Basic information: For details, see Line > Basic Information.
- Animation: For details, see 13.3.8 Animation Configuration.
- Event: For details, see 13.3.7.14 Adding an Event.

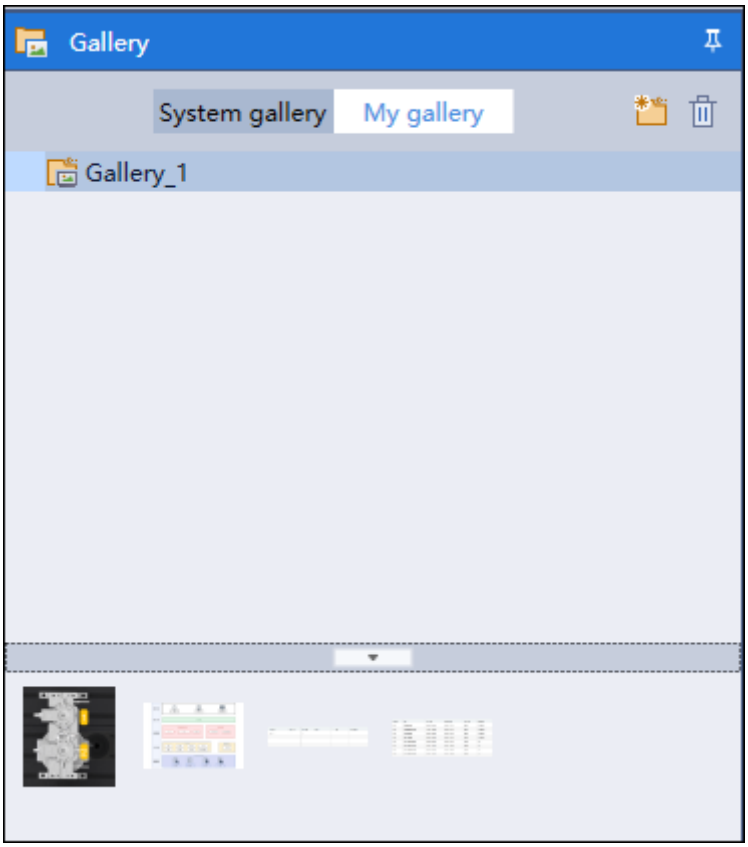
13.3.6.3 User Gallery Usage

Procedure

1. Select "My gallery", click the "Add" icon in the upper right corner, and the "Create folder link" window pops up:
   - Name: The name of the created gallery (default: Graphics_1, Graphics_2, …, Graphics_n).
   - Path: Points to a Windows folder. After creation, the software will automatically display supported images from this folder as previews in the gallery.



2. After successful addition, the gallery information can be viewed in "My gallery". Elements in the gallery can be added to screens.



> **NOTE**
>
> Images with memory exceeding 8 M or resolution exceeding 8 K (7680*4320 pixels) cannot be imported.

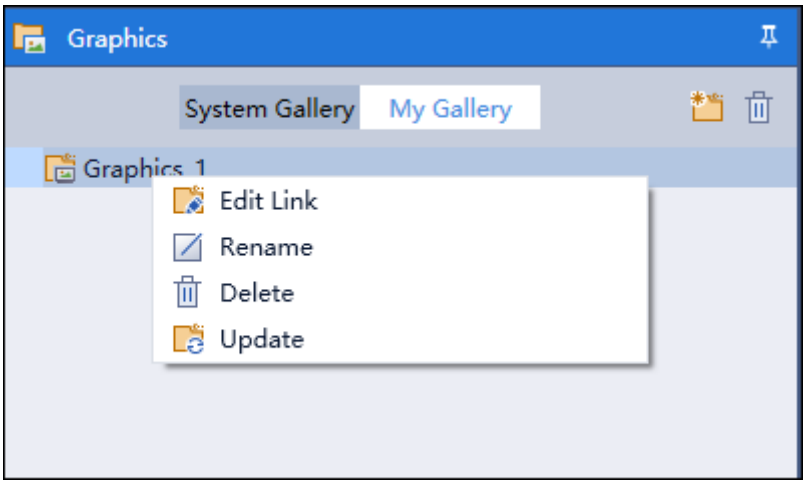
Image properties

- Basic settings

Item	Description
Position	Sets the position coordinates (i.e., top-left coordinates) of the element on the screen in pixels: • X: X-axis coordinate. • Y: Y-axis position coordinate. The coordinate value must be smaller than the screen boundary coordinate value to prevent the elements from extending beyond the screen.
Size	Sets the size of the graphic element on the screen in pixels: • W: Element width. • H: Element height. The minimum size for the graphic element is 1*1.
Rotation	Sets the rotation angle of the graphic element, with a range from -180 to 180.

- Basic information: For details, see Line > Basic Information.

- Animation: For details, see 13.3.8 Animation Configuration.
- Event: For details, see 13.3.7.14 Event Configuration.
- Right-click a folder in "My gallery" to edit:



  - Edit link: Modify the gallery name and path information.
  - Rename: Modify the gallery name.
  - Delete: Delete the selected gallery.
  - Update: Refresh the gallery information. When images are added or removed in the specified Windows folder, the preview will be refreshed to list valid images.

Gallery data retention

When uninstalling the iFA software, select how to handle user gallery data via the "Keep local settings" option:



Checked: After uninstallation, images and access paths in the user gallery will be retained. If the iFA software is reinstalled on the same computer, the gallery data and related settings will be automatically restored.

Unchecked: User gallery data will be deleted simultaneously during uninstallation. After reinstalling, you will need to rebuild the gallery.

13.3.6.4 Project Gallery Usage

Image specification limits

Each referenced image must be smaller than 8 MB in size and have a resolution below 8 K (7680 x 4320 pixels).

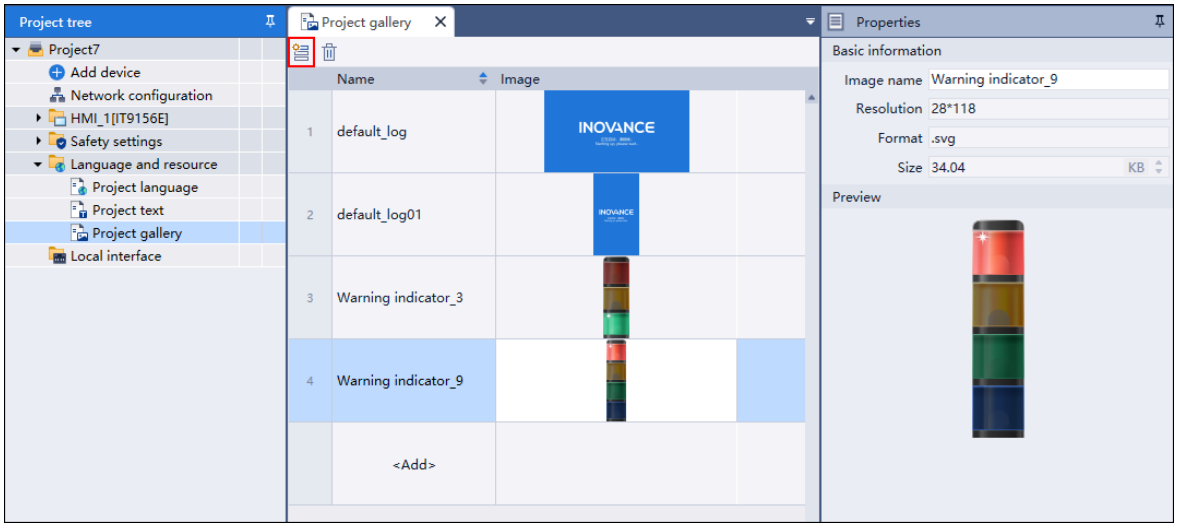
Storage capacity limits

Whichever limit is reached first will trigger a system alert:

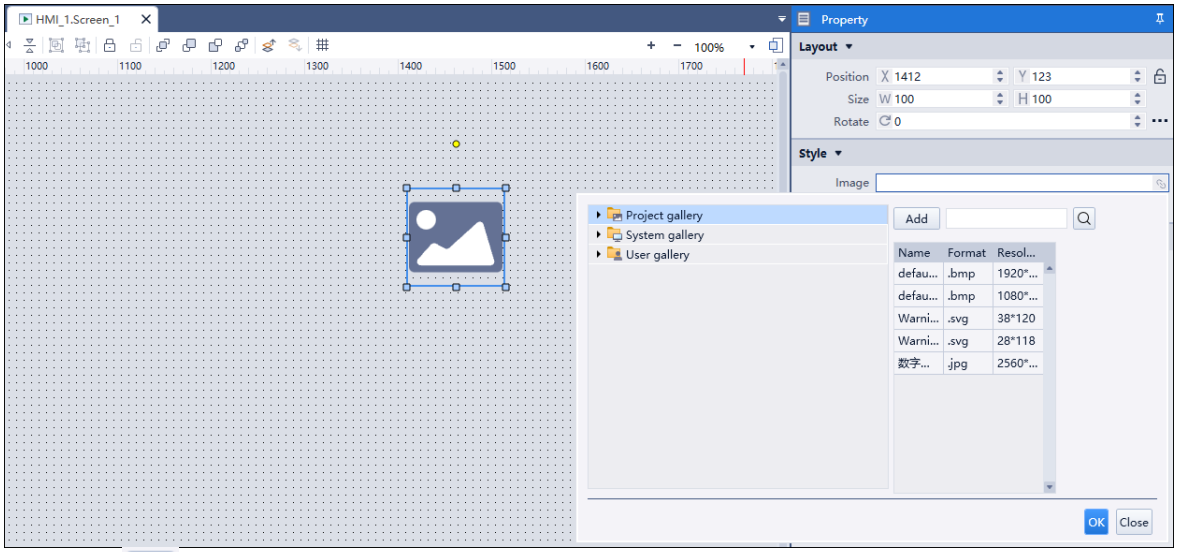
- Maximum number of images: 10000.
- Maximum total image size: 1024 MB.

Procedure

1. Double-click "Project gallery" under "Language and resources" in the left project tree to open the project gallery.
2. Add images.
   - Manual addition: Click , select the images to be added, and click "Open" to import the images into the project gallery.



   - Automatic addition: Graphics used within a project are automatically added to the project gallery for unified management.
     Example: When adding an image via the 13.3.5.1.11 Image control, all related operations are performed in the property tab of the 13.3.5.1.11 Image control. The image will be automatically added to the project gallery.



- Search

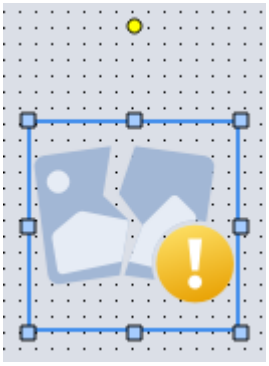
  Search based on the entered keyword in the selected gallery on the left.

- Add

  Add resources from local PC to the local gallery.

> **NOTE**
>
> The project gallery can be used across all HMI devices in the current project.

3. Delete an image. Select an image and click to delete it from the project gallery. After deletion, the corresponding image control is as follows.



13.3.6.5 Local Gallery Usage

The local gallery is used to store the specific images associated with the 13.3.5.1.11 Image control in the symbol template. Each 13.4.3.2 Symbol Template has its own local gallery, independent of each other.

Image specification limits

Each referenced image must be smaller than 8 MB in size and have a resolution below 8 K (7680 x 4320 pixels).

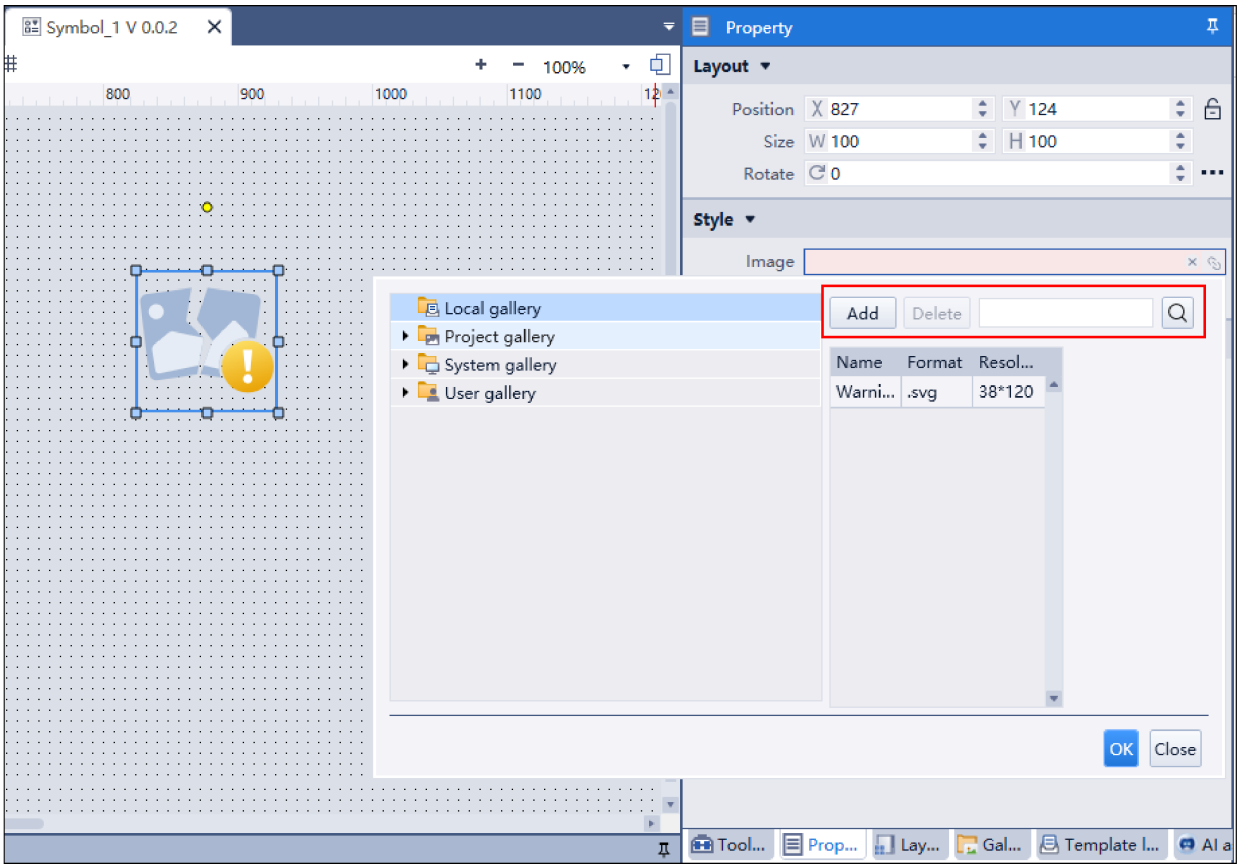
Storage capacity limits

Whichever limit is reached first will trigger a system alert:

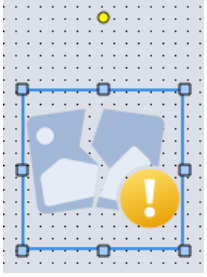
- Maximum number of images: 100.
- Maximum total image size: 32 MB.

Procedure

Operations related to the local gallery are performed on the property tab of the 13.3.5.1.11 Image control.



- Search
  Search based on the entered keyword in the selected gallery on the left.
- Add
  Add resources from local PC to the local gallery.
- Delete
  Delete images from the local gallery. After deletion, the corresponding image control is as follows.



13.3.7 Screen Configuration

- 13.3.7.1 Creating an Object

- 13.3.7.2 Renaming an Object
- 13.3.7.3 Copying/Pasting an Object
- 13.3.7.4 Cutting/Pasting an Object
- 13.3.7.5 Deleting an Object
- 13.3.7.6 Rotating an Object
- 13.3.7.7 Flipping an Object
- 13.3.7.8 Aligning Objects
- 13.3.7.9 Grouping Objects
- 13.3.7.10 Ungrouping Objects
- 13.3.7.11 Locking an Object
- 13.3.7.12 Unlocking an Object
- 13.3.7.13 Adding an Animation
- 13.3.7.14 Adding an Event
- 13.3.7.15 Configuring a Picture-in-Picture Screen
- 13.3.7.16 Resizing an Object
- 13.3.7.17 Moving an Object
- 13.3.7.18 Adjusting Object Spacing
- 13.3.7.19 Configuring a Pop-up Screen

13.3.7.1 Creating an Object

A screen supports up to 500 objects, with grouped objects counted by their members and picture-in-picture counted as one object. For example, a grouped object containing four lines and one text element is counted as five objects.

- There are three ways to create an object with default size:
  - Select an object in the toolbox, then click on the screen to create a default-sized element at the specified location.
  - Select an object in the toolbox and drag it into the screen to create a default-sized element.
  - Drag a variable from the detailed view into the screen to automatically create the corresponding element and establish a link between the variable and the element. For example, a BOOL variable corresponds to a bit button, and a REAL variable corresponds to a numeric I/O field.
- To create an object with custom size, follow these steps:
  1. Select an object in the toolbox.
  2. Click and hold the mouse at the start position on the screen, drag to the end position, and then release.
- To create a polyline, polygon, and pipeline object, follow these steps:
  1. Select the target element in the toolbox.
  2. Click at the start position on the screen, move to the end of the first segment, and click again to complete the first segment.
  3. Repeat the above steps for each segment, and then right-click to generate the desired element.

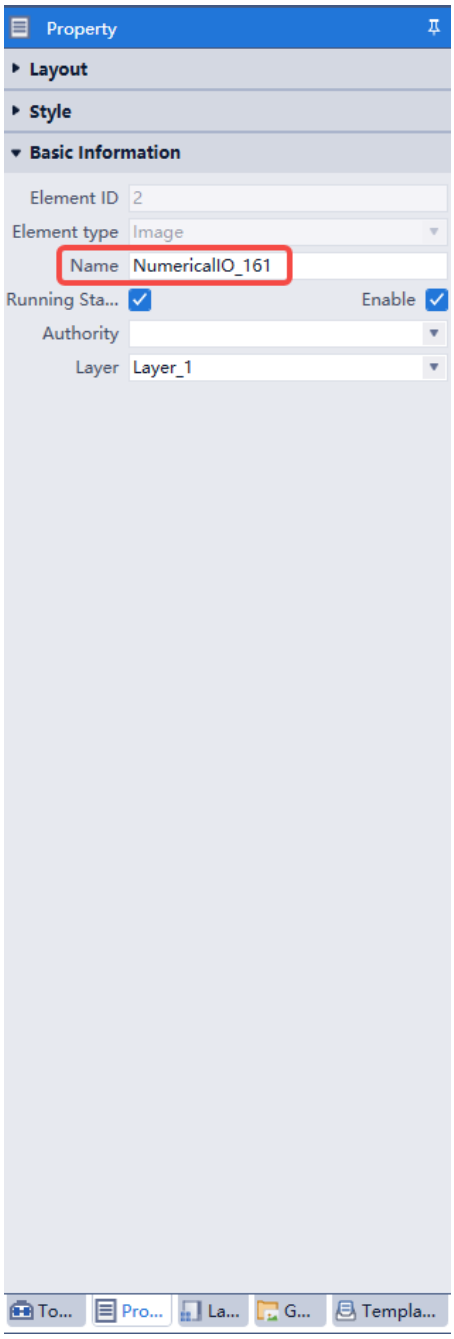
13.3.7.2 Renaming an Object

Each object on the screen has a unique name, which is automatically assigned by the software in the order they are created. The default naming format is "Object name (Name in toolbar)_Number", such as "Line_1".

Objects can be renamed. The new name can consist of Chinese characters, English letters, numbers, and special characters. It cannot start or end with a space, and supports up to 128 characters.

Renaming steps:

- Object properties
  1. Double-click an object on the screen, such as a numeric I/O field.
  2. Enter the new name in "Name".



  3. Press **Enter** to confirm.

- Screen layout
  1. Open the "Layout" tab of the screen.
  2. Select an object node in the layer tree.

3. Press **F2** or select "Rename" from the right-click menu.
4. After entering the new name, press **Enter** to confirm.

13.3.7.3 Copying/Pasting an Object

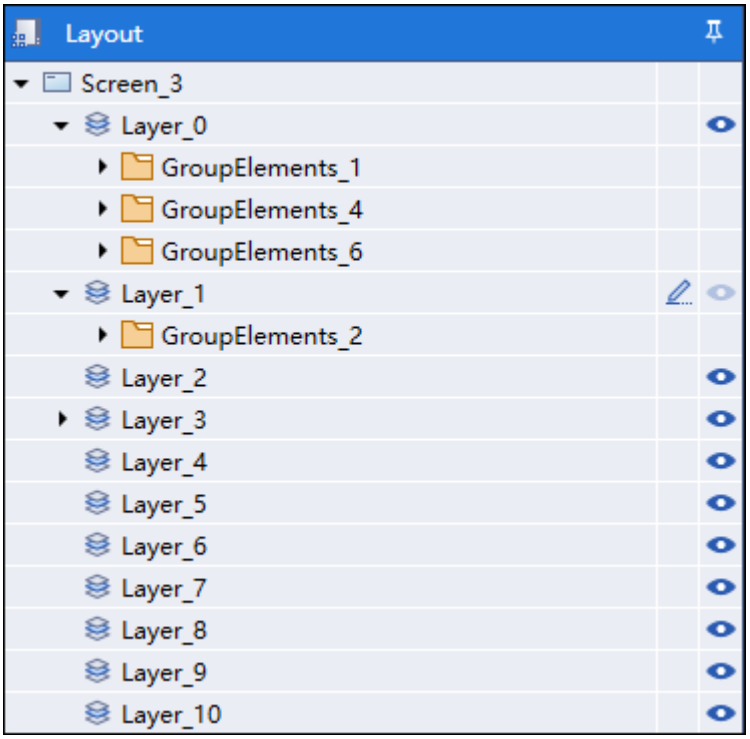
Objects on the screen can be copied and pasted within the same screen or across different screens.

Operating on the screen

1. Select the target object on the screen.
   When copying multiple objects, use the **Ctrl** key or box selection to select multiple objects.
2. Copy the object.
   Press **Ctrl+C** or click "Copy" in the right-click menu to copy the selected object along with all its properties, events, and animation settings.
3. Paste the object.
   Press **Ctrl+V** or click "Paste" in the right-click menu to paste the object onto the currently active layer. Multiple pastes are supported. Object names will automatically be adjusted to avoid duplicates. If the copied object is locked, it will be automatically unlocked after pasting.

Operating in the layout

1. Select the target object in the layout.
   When copying multiple objects, use the **Ctrl** key to select multiple objects.



2. Copy the object.
   Press **Ctrl+C** or click "Copy" in the right-click menu to copy the selected object along with all its properties, events, and animation settings.
3. Paste the object.
   Press **Ctrl+V** or click "Paste" in the right-click menu to paste the object onto the selected layer. Multiple pastes are supported. Object names will automatically be adjusted to avoid duplicates. If the copied object is locked, it will be automatically unlocked after pasting.

13.3.7.4 Cutting/Pasting an Object

Objects on the screen can be cut and within the same screen or across different screens.

1. Select the target object on the screen.
   When cutting multiple objects, use the **Ctrl** key or box selection to select multiple objects.
2. Copy the object.
   Press **Ctrl+X** or click "Cut" in the right-click menu to cut the selected object along with all its properties, events, and animation settings.
3. Paste the object.
   Press **Ctrl+V** or click "Paste" in the right-click menu to paste the object. Objects are typically pasted onto the active layer.
   Supports cutting once and pasting multiple times. When pasting, all properties, events, and animation settings of the object are included.
   When pasting multiple times, object names will automatically be adjusted to avoid duplicates.

13.3.7.5 Deleting an Object

Used to delete one or more objects, without affecting their associated variables or scripts.

Operation:

1. Select the object to delete from the screen or layout tree.
   When selecting from the layout tree, to delete one of the objects from a group containing two objects, 13.3.7.10 ungroup it first.
2. Click "Delete" from the right-click menu or press **Delete** to delete the selected object.

13.3.7.6 Rotating an Object

There are three ways to rotate an object:

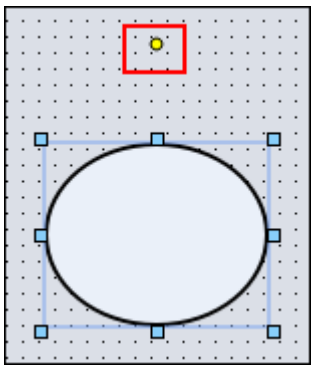
- Click the rotate icons on the toolbar to rotate.

Icon	Description
	Rotates the selected object clockwise by 90 degrees around its center point.
	Rotates the selected object counterclockwise by 90 degrees around its center point.

  Applicable objects include:
  - Basic element: 13.3.5.1.3 arc, 13.3.5.1.4 rectangle, 13.3.5.1.5 circle, 13.3.5.1.6 ellipse, 13.3.5.1.7 sector, 13.3.5.1.8 circular segment, 13.3.5.1.10 text, and 13.3.5.1.11 image;
  - Control and display: 13.3.5.2.1 numeric I/O field, 13.3.5.2.2 character I/O field, 13.3.5.2.3 date/time field, and 13.3.5.2.12 text switch;
  - Data visualization: 13.3.5.3.1 bar chart;

◦ Others: 13.3.5.5.2 barcode.

- Rotate the object by any angle using the mouse.
  1. Select the object to be rotated, and the yellow rotation icon will appear, as shown in the figure below.



  2. Press and hold the rotation icon to rotate by any angle.

- Rotate the object through the property window.

13.3.7.7 Flipping an Object

Select one or more graphic objects to flip, then click the flip icons button on the toolbar. Flipping is supported by only some objects.

Icon	Description
	Displays the selected object in horizontal symmetry.
	Displays the selected object in vertical symmetry.

13.3.7.8 Aligning Objects

Select two or more graphic objects to align, then click the align icons on the toolbar.

Icon	Description
	Aligns the left edges of the object and the reference object.
	Aligns the right edges of the object and the reference object.
	Aligns the top edges of the object and the reference object.
	Aligns the bottom edges of the object and the reference object.
	Aligns the selected objects with the vertical center axis of the reference object.
	Aligns the selected objects with the horizontal center axis of the reference object.

Icon	Description
	Aligns the selected objects with the center point of the reference object.
	Places the geometric center of the selected objects on the horizontal center line of the entire screen, while keeping the vertical position of each object unchanged.
	Places the geometric center of the selected objects on the vertical center line of the entire screen, while keeping the horizontal position of each object unchanged.
	Places the geometric centers of the selected objects at the center point of the entire screen.

13.3.7.9 Grouping Objects

Group two or more objects into a single entity, which is referred to as a group object or group element (default).

Grouping operation

Objects can be grouped on the screen or in the "Layout" tab:

1. Select at least two objects on the screen or in the layer tree.
2. Click "Group" in the right-click menu.

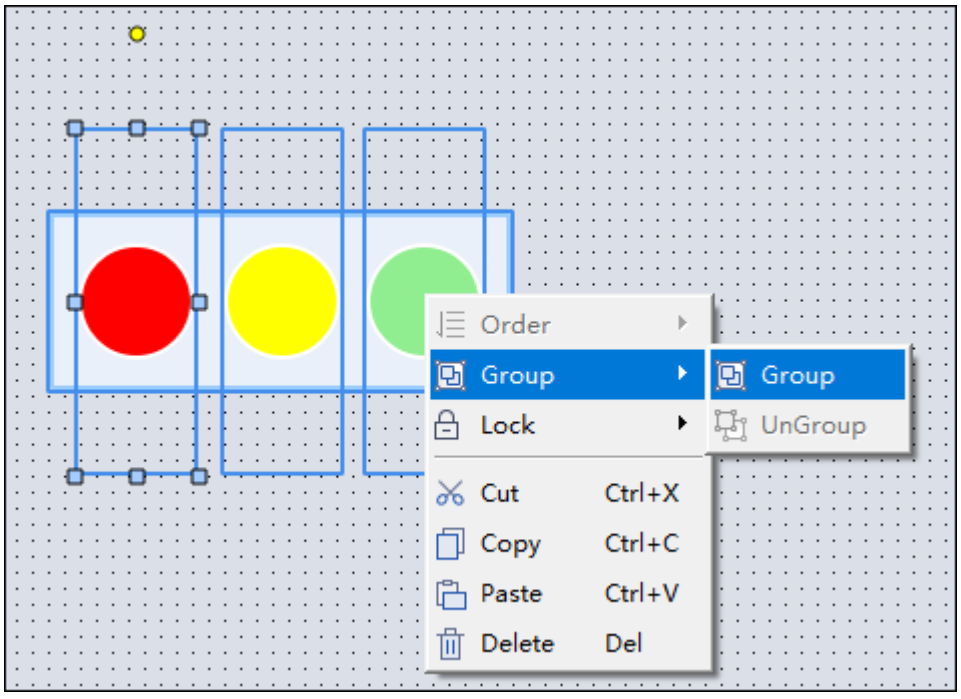


Figure 13-3-7-9-1 Grouping on the screen

If the members for grouping are located on different layers, they will all be moved to the layer with the higher number after grouping.

Group objects are named "GroupElements_n" in the layer tree, which can be renamed. For details, see 13.3.7.2 Renaming an Object.

Adding members to a group

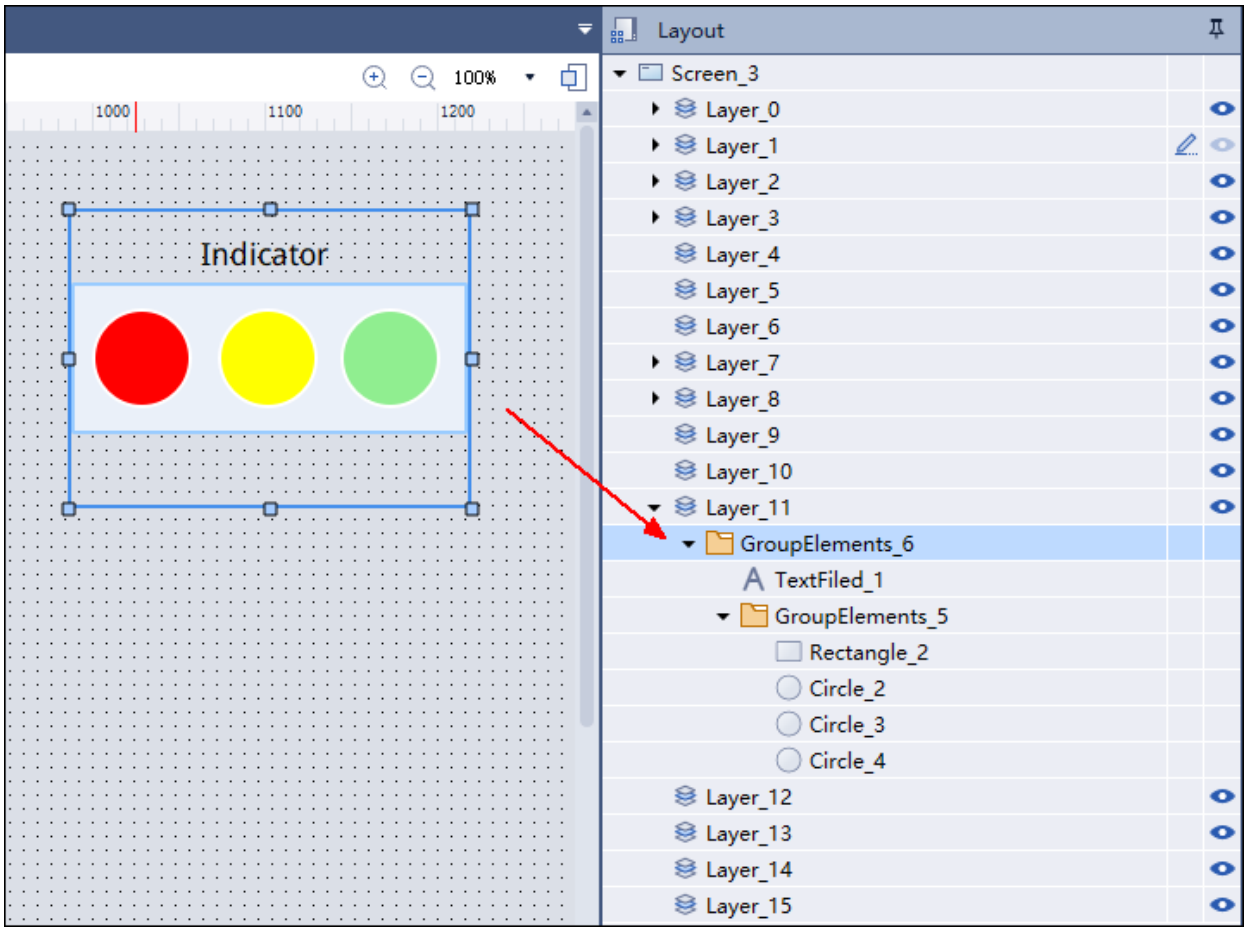
Other objects can be added to an existing group to form a new nested group.

Select an existing group object and the objects to be added on the screen or from the layer tree, then select

"Group" in the right-click menu.

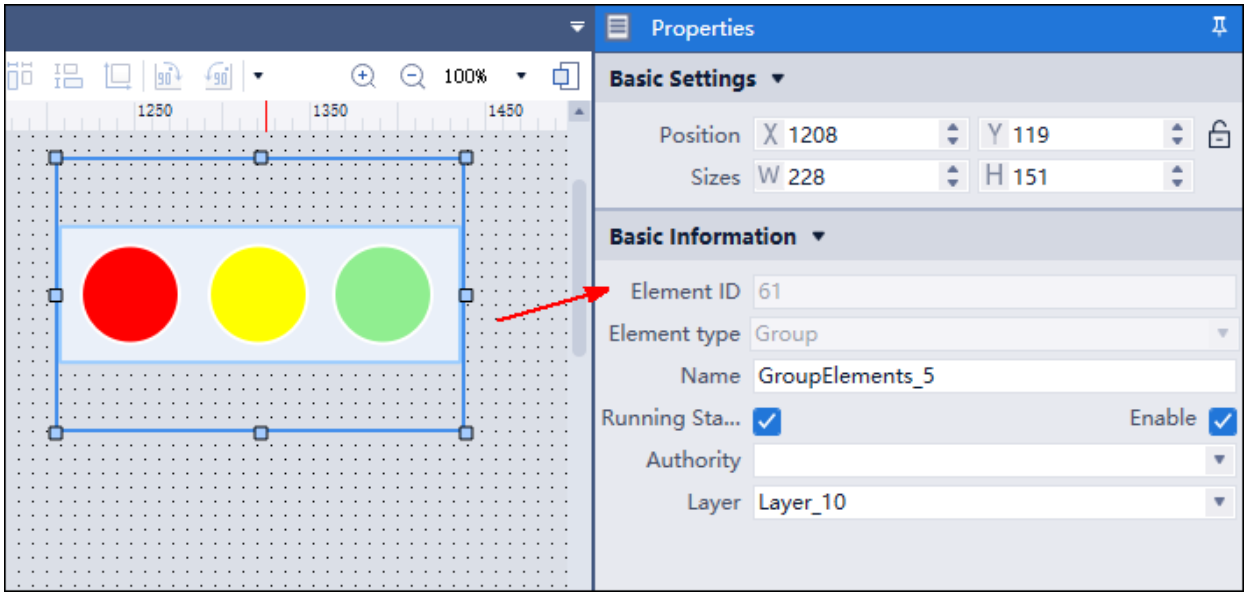
If the members for grouping are located on different layers, they will all be moved to the layer with the higher number after grouping.

The hierarchical relationships among group members can be viewed through the layer tree.



Setting group properties

Groups have their own properties, such as permissions. When a property is set for a group, all its members inherit this property. When a property is set differently on a group member than on the group itself, the member's value takes precedence at runtime. Select a group on the screen, and its related properties will be displayed in the property box.

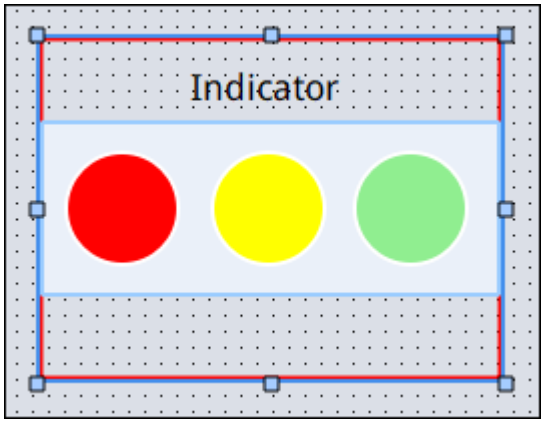


Users can adjust the size and position of the group.

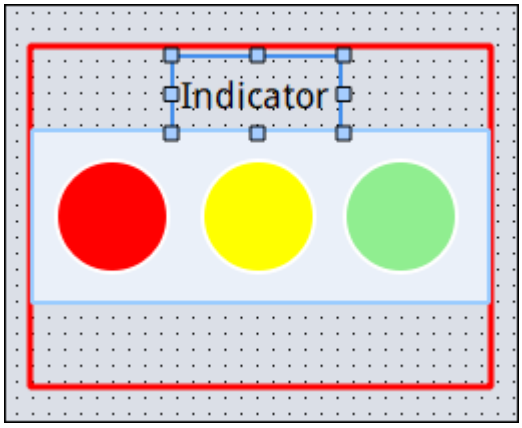
Modifying group member properties

The property values of group members can be modified without ungrouping.

- Modifying on the screen
  1. Double-click a group on the screen and its surrounding rectangle becomes red.



  2. Click the member to modify to open its property box.



  3. Modify the property values of the selected member.
     Some properties cannot be modified, depending on the object type. Refer to the software for details.
- Modifying on the layer tree
  1. Select a group member on the layer node.
  2. Open the "Property" tab.
  3. Modify the property values of the selected member.
     Some properties cannot be modified, depending on the object type. Refer to the software for details.

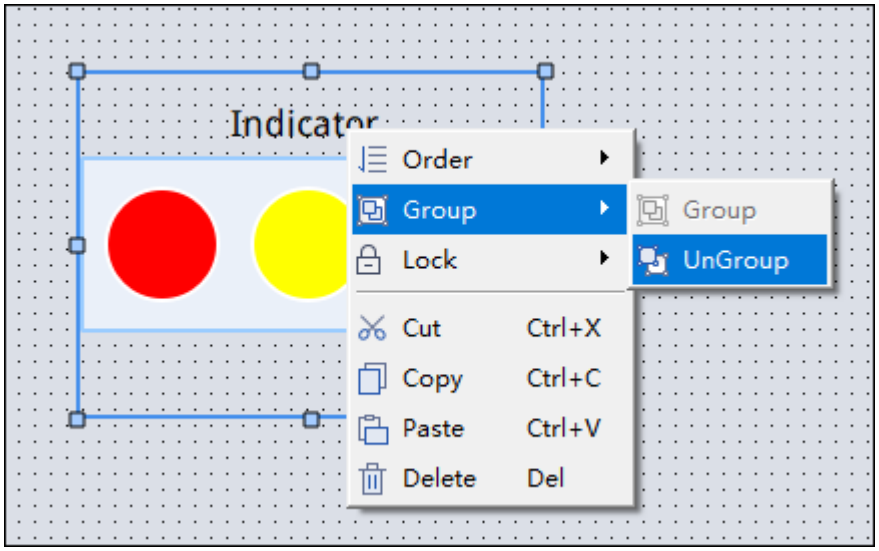
13.3.7.10 Ungrouping Objects

Ungrouping refers to restoring a group to individual objects before grouping. Properties of the group will be automatically canceled during ungrouping.

Ungrouping can be performed in two ways. Nested groups must be ungrouped step by step from the outermost level inward.

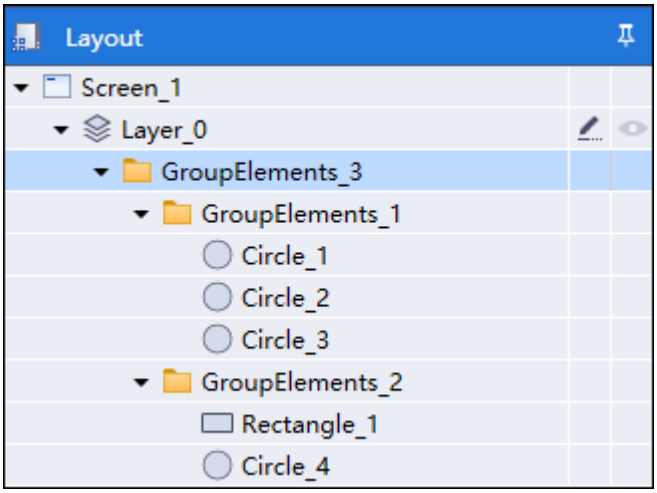
Operation on the screen

1. Select the target group on the screen.
2. Select "Group > Ungroup" in the right-click menu.



Operation in the layout

1. In the "Layer" tab, select the target group node. If the group is nested, select its parent node.

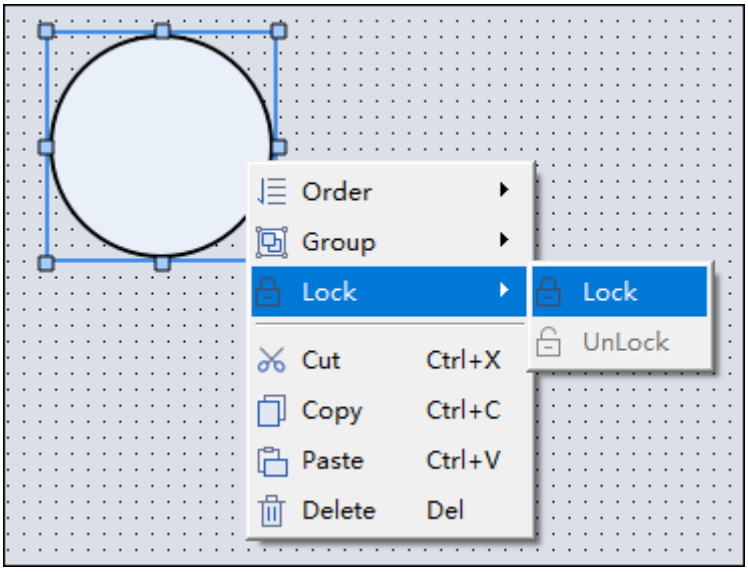


2. Click "Ungroup" in the right-click menu.

13.3.7.11 Locking an Object

Select one or more objects to be locked. Locked objects cannot be moved, scaled, or rotated. There are two locking methods:

- Right-click the target object on the screen and select "Lock > Lock".

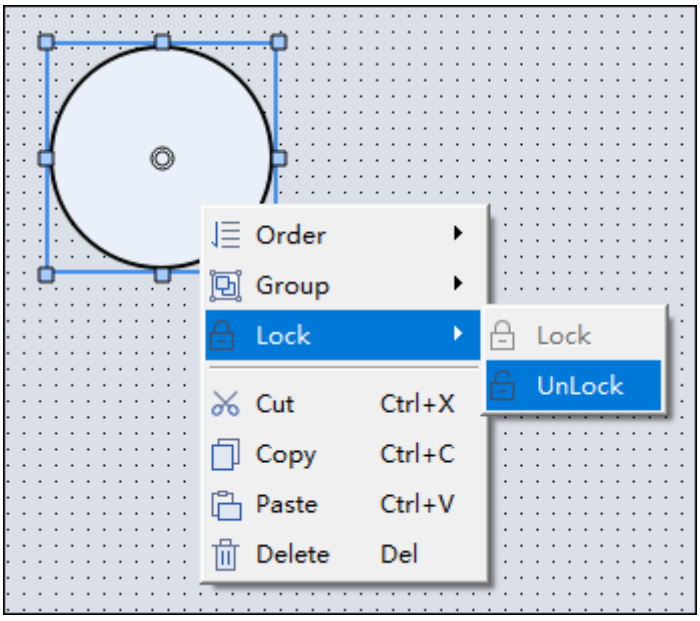


- In the property window of the target object, click the lock-shaped icon to the right of "Position".

13.3.7.12 Unlocking an Object

Select one or more objects to unlock. Unlocked objects can then be moved, scaled, and rotated. There are two unlocking methods:

- Right-click the target object on the screen and select "Lock > Unlock".



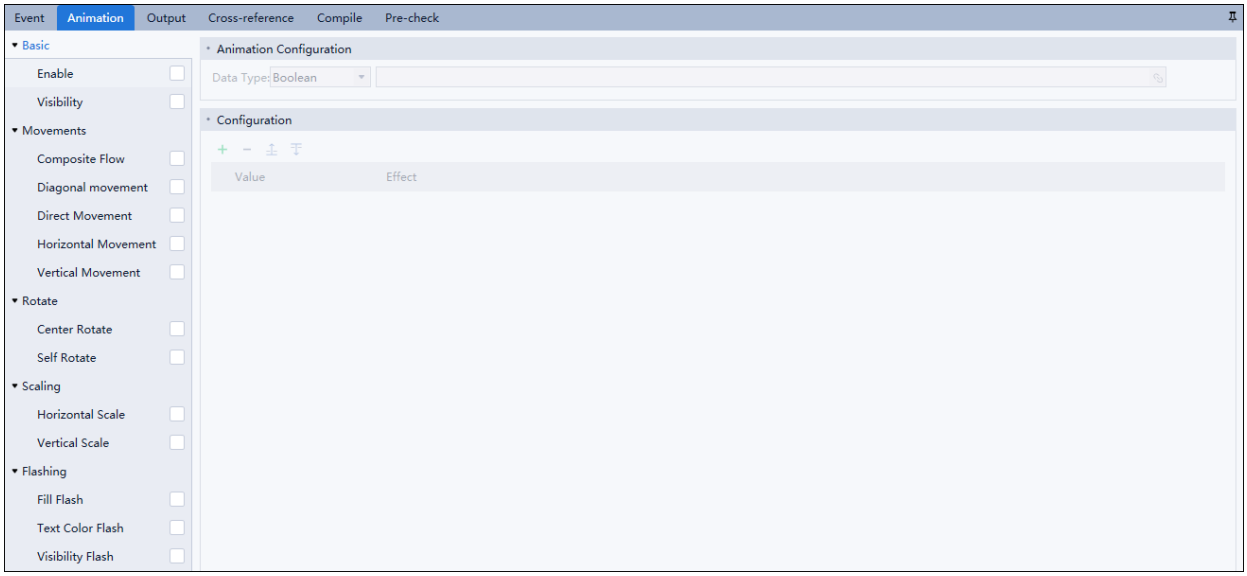
- In the property window of the target object, click the lock-shaped icon to the right of "Position".

13.3.7.13 Adding an Animation

The visualization platform offers rich element animation design for appearance, fill, movement, rotation, and scaling.

Procedure

1. Select the element to edit;
2. Click "Animation" in the monitoring area to enter the animation configuration interface, as shown below:



13.3.7.14 Adding an Event

The visualization platform provides screen events and element events for users.

A screen event refers to the action executed by the screen when entering the runtime environment. It can be triggered when a screen opens, when a screen closes, or by a timer. An element event refers to the action executed by the runtime system when the mouse interacts with an element. They are both implemented by configuring system functions or scripts to define the conditions that trigger the execution of related tasks.

Procedure

The screen event and element event are configured in the same way, as shown below:

1. Select the screen or element to edit;
2. Click "Event" in the monitoring area to display the screen or element event list, as shown below:

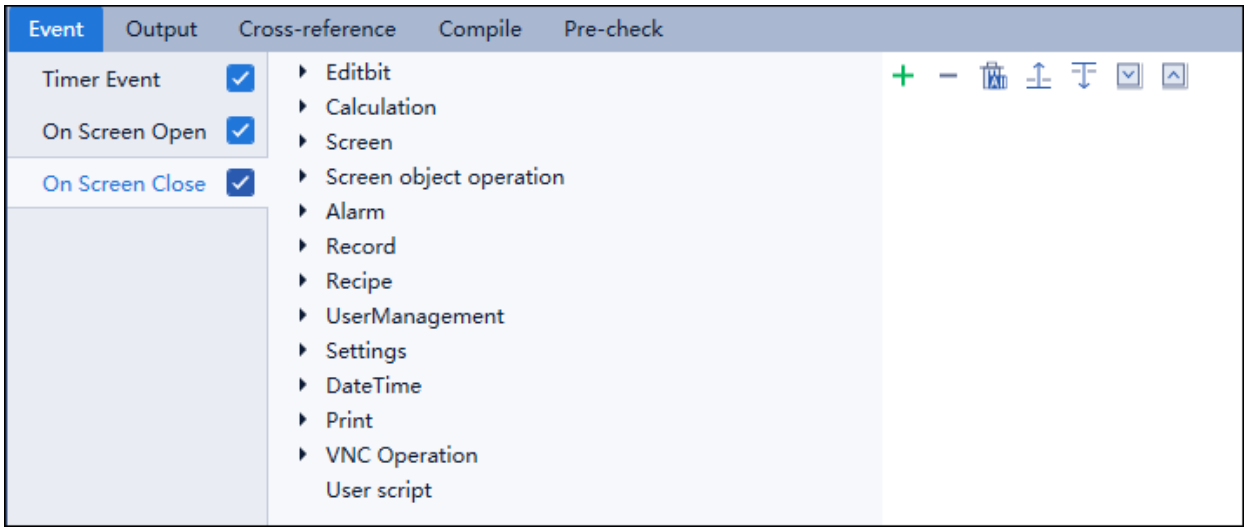


Figure 13-3-7-14-1 Screen event list

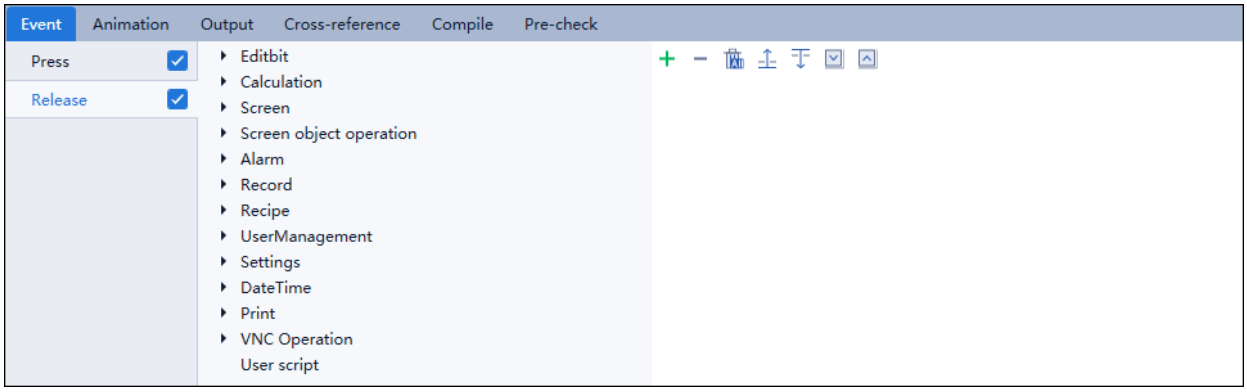


Figure 13-3-7-14-2 Element event list

Function

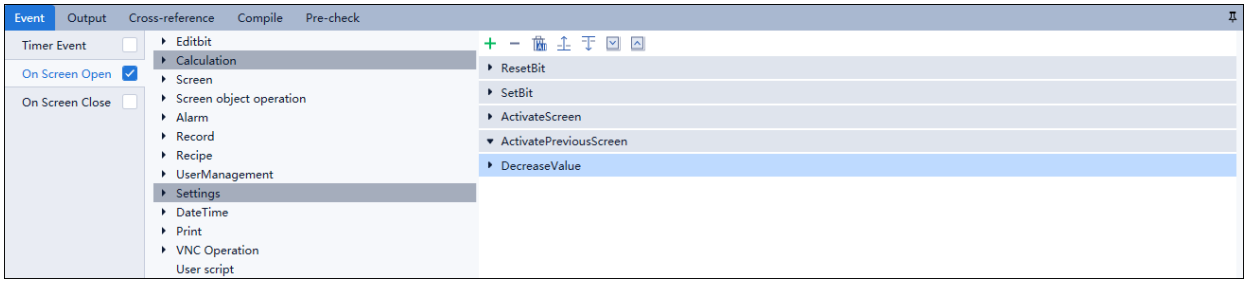
- **Event configuration function**
  Select an event and add the event function using the following two methods:
  - Method 1: Expand a built-in function category and double-click the target function to add it to the selected function list on the right.
  - Method 2: Expand a built-in function category, select the target script function, then click the "+" icon to add it to the selected function list on the right.

  The function list includes string operation functions, bit operation functions, calculation functions, screen functions, recipe operation functions, alarm functions, record functions, date and time functions, user and login management functions, setting functions, and user script functions. Element events and screen events support the same functions, and any function from any category can be added.
- **Element event**
  After configuring an event for an element, the runtime system executes the corresponding function when the element meets the trigger condition.
  - Press: Triggered when the touchscreen or the mouse button is pressed.
  - Release: Triggered when the touchscreen or the mouse button is released.
  - Change: Triggered when the object is modified via mouse or touchscreen operation.
  - Input completed: Triggered when an I/O field input is completed via mouse or touchscreen operation.
  - Open: Triggered when a switch is turned on.
  - Close: Triggered when a switch is turned off.
- **Group event**
  - Group event response rules:
    - Only the group is configured with an event: The event responds to clicks in the group area, but not clicks in blank areas.
    - The group and some sub-elements are configured with events: The sub-element event responds to clicks in the sub-element area; the group event responds to clicks in other area within the group; blank areas do not respond.
    - The group and all sub-elements are configured with events: Only sub-element events respond, and the group event does not respond.
    - To respond to clicks in blank areas, add a background element configured with an event.

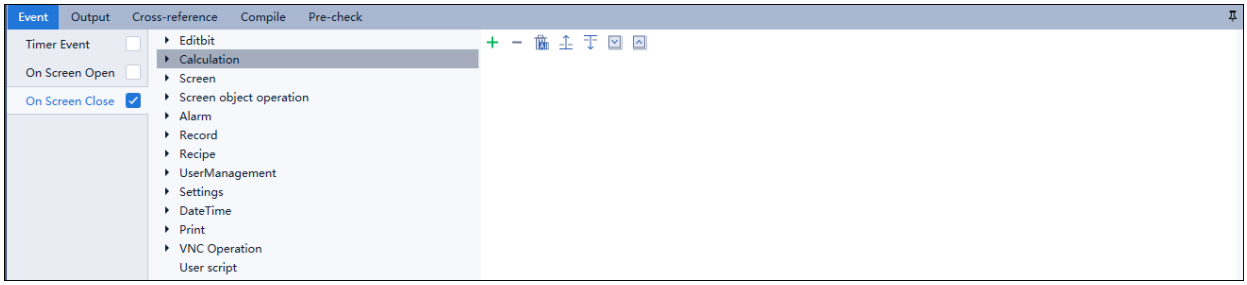
- Impact of visibility on event response: When an element is obscured or hidden, its event cannot be triggered. If all sub-elements of a group are hidden, the group event cannot be triggered either.
- Event response for special control group: When a special control is grouped with a basic element, only the basic element responds to the configured group event, because the special control has its own built-in events.
- Impact of enable on event response:
  - When a sub-element is enabled, clicking its area triggers the sub-element event.
  - When a sub-element is disabled, clicking its area triggers the group event.
- Rules for group selection and event prompt:
  - Sub-element selection: In offline mode, you can select a sub-element via the "Layout" tab or by double-clicking the element within the group.
  - Group operation: In offline or runtime mode, clicking the blank area of the group is treated as operating the entire group.
  - Event prompt: For any element (single or group) configured with an event, the mouse cursor changes to a hand shape only when the event can be triggered.

- **Screen event**

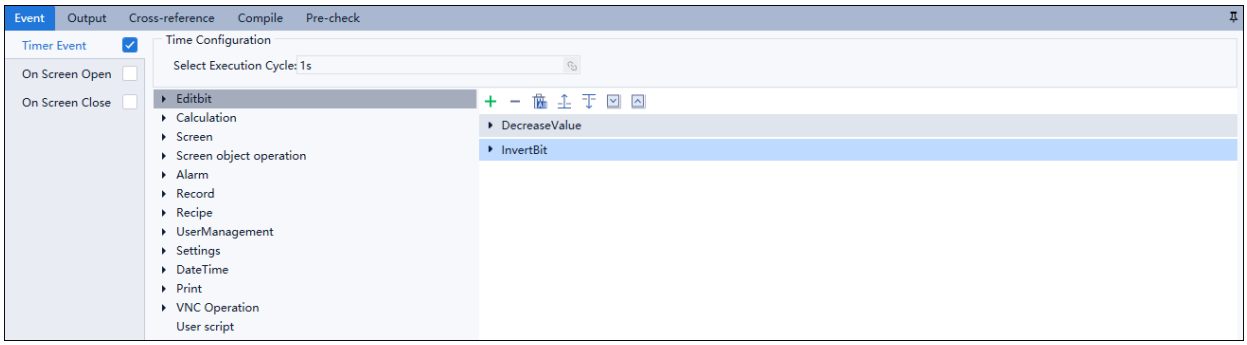
  A screen event can be triggered when a screen opens, when a screen closes, or by a timer.
  - On screen open: Triggered when the screen is opened in runtime.



  - On screen close: Triggered when the screen is closed in runtime.



  - Timer event: Periodically triggered at a defined cycle in runtime.



> **NOTE**
>
> Time configuration: The default execution cycle is 1s. The cycle can be selected from existing cycles, but cannot be manually entered.

13.3.7.15 Configuring a Picture-in-Picture Screen

Before configuring a picture-in-picture in the process screen, first complete the drawing of the target picture-in-picture.

Procedure:

1. Add a picture-in-picture view
   Select the **picture-in-picture view** in the toolbox and drag it to any location on the process screen.

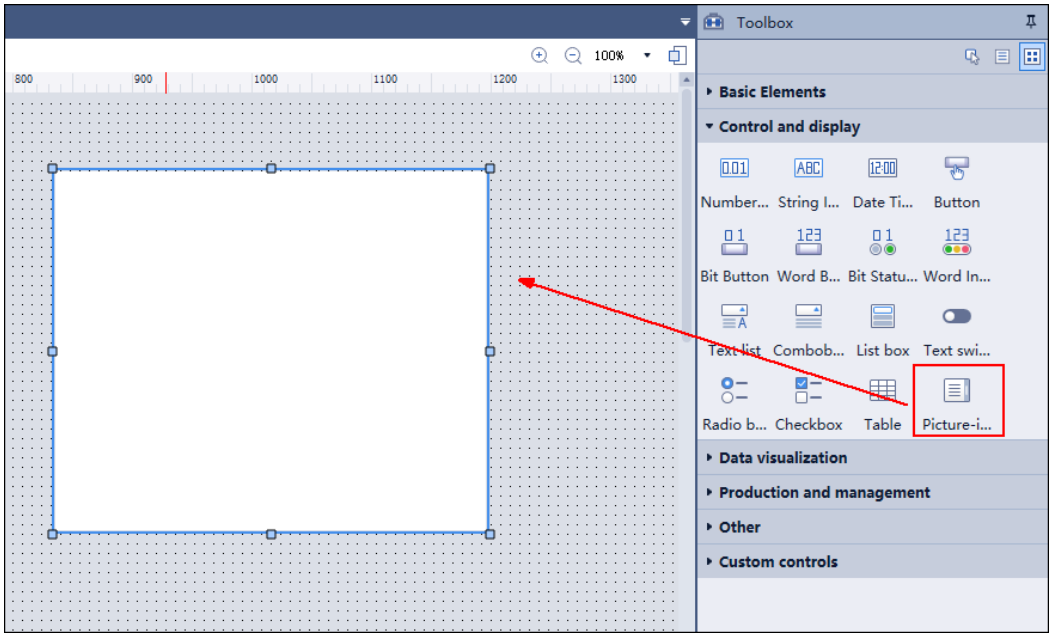


Figure 13-3-7-15-1 Adding a picture-in-picture view

2. Adjust the position and display window size of the picture-in-picture view
   Set the space occupied by the picture-in-picture object in the process screen according to its dimensions.
3. Associate the picture-in-picture
   In the property window of the picture-in-picture view, enter the information of the target picture-in-picture. See 13.3.5.2.16 Picture-in-picture View for details on the property items.
4. Simulation test
   Start simulation mode to preview the display effect of the picture-in-picture.

The configuration of the picture-in-picture view on the process screen is complete.

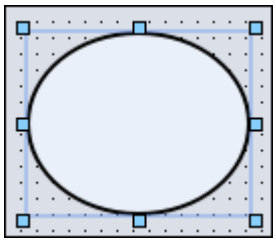
13.3.7.16 Resizing an Object

There are three ways to resize an object:

- Click the resize icons on the toolbar (supports selecting two or more graphic objects).

Icon	Description
	Adjust all selected graphic objects to the same width.
	Adjust all selected graphic objects to the same height.
	Adjust all selected graphic objects to the same width and height.

- Resize by dragging with the mouse.
  1. Select the object to resize, and a selection rectangle appears, as shown below:



  2. Drag a size control point of the rectangle to resize the object.
- Resize through the object property window.

13.3.7.17 Moving an Object

An object can be moved within the same layer and across layers:

Moving within the same layer

The position of an object within the same layer includes coordinate position and front-to-back order.

- Coordinate position
  Move the coordinate position by dragging, using direction keys, or entering coordinate values in the property window.
  Ensure the object is 13.3.7.12 unlocked before moving.

Key	Description
←	Moves the target object to the left.
→	Moves the target object to the right.
↑	Moves the target object up.
↓	Moves the target object down.

- Front-to-back order
  Objects may overlap or intersect within the same layer. Use the icons on the toolbar to adjust the front-to-back order of these objects and change their occlusion relationship.

Icon	Description
	Moves the target object to the top.
	Moves the target object to the bottom.

Icon	Description
	Moves the target object up one layer.
	Moves the target object down one layer.

Moving across layers

Use the icons on the toolbar or drag-and-drop to move an object from one layer to another.

- Icons
  Select an object on the screen and move it to another layer using the following icons.

Icon	Description
	Moves the layer containing the target object up one layer.
	Moves the layer containing the target object down one layer.

- Drag-and-drop
  In the "Layout" window, select the target object and drag it to the target layer.
  Movable objects include individual objects and members of groups. For groups with two members, only moving members into the group is supported.

  The drop position when moving the target object into a group affects its order in the group object tree:
  - When dragging the target object into the group folder node, it will be placed at the end.
  - When dragging the target object into a specific group member node, it will be inserted above that member.

13.3.7.18 Adjusting Object Spacing

Select two or more graphic objects, and click the corresponding toolbar icon to adjust the object spacing.

Icon	Description
	Equalizes horizontal spacing between the geometric centers of multiple selected objects (Note: At least three objects must be selected).
	Equalizes horizontal spacing between multiple selected objects (Note: At least three objects must be selected).
	Reduces horizontal spacing between multiple selected objects until their geometric centers align on the same vertical line.
	Increases horizontal spacing between multiple selected objects.

Icon	Description
	Sets horizontal spacing between multiple selected objects to 0.
	Equalizes vertical spacing between the geometric centers of multiple selected objects (Note: At least three objects must be selected).
	Equalizes vertical spacing between multiple selected objects (Note: At least three objects must be selected).
	Reduces vertical spacing between multiple selected objects until their geometric centers align on the same horizontal line.
	Increases vertical spacing between multiple selected objects.
	Sets vertical spacing between multiple selected objects to 0.

13.3.7.19 Configuring a Pop-up Screen

Before configuring a pop-up window on the screen, first complete the drawing of the pop-up window.

Procedure:

1. Add an object
   Add an object on the screen used to open the pop-up window. Typically, a button is used, but other elements or symbols can also be used depending on project requirements.
   Select the **button** element from the toolbox and drag it to the screen.

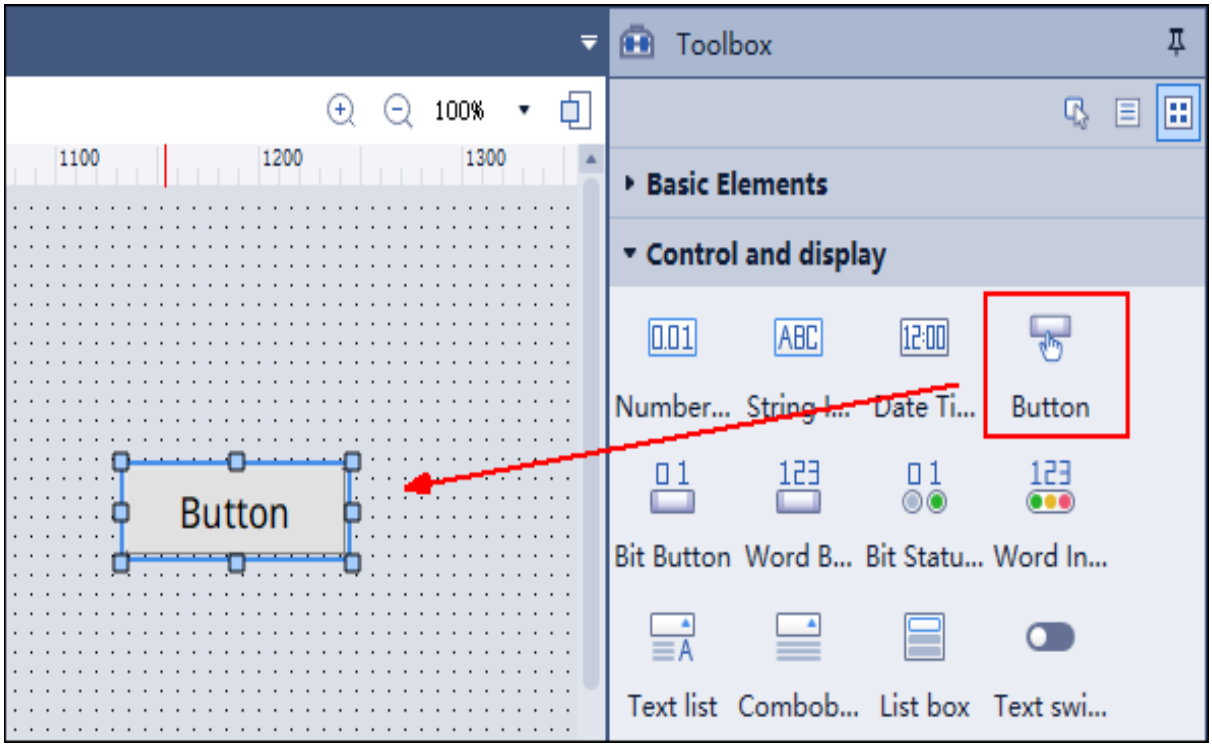


Figure 13-3-7-19-1 Adding a button

2. Move button position

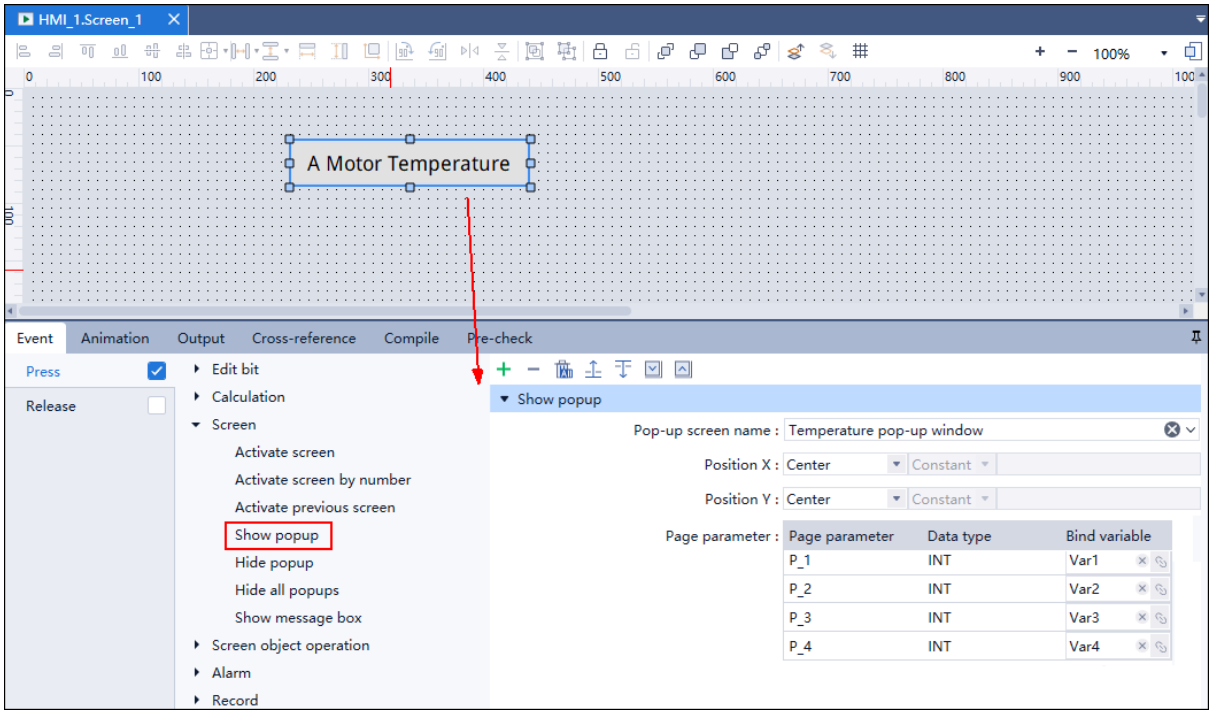
   Move the button to the target position.

3. Set button style

4. Set button event

   Set the trigger condition and target pop-up window (such as the pop-up screen when the button is pressed). The configuration is shown in the figure below.

   Enter the pop-up window information in the "Event" window. See 19.4.4 Showing the Pop-up Screen for descriptions of each parameter.



5. Start simulation

   Start the simulation mode to test the display result of the pop-up window.

The pop-up screen configuration is complete.

13.3.7.20 Setting Properties in Batch

When multiple objects of the same type need to be uniformly adjusted on the screen, batch modification allows users to efficiently edit multiple objects, reducing repetitive operations.

Prerequisites

- The objects are of the same type.
- Object names, events, and animations do not support batch modification.

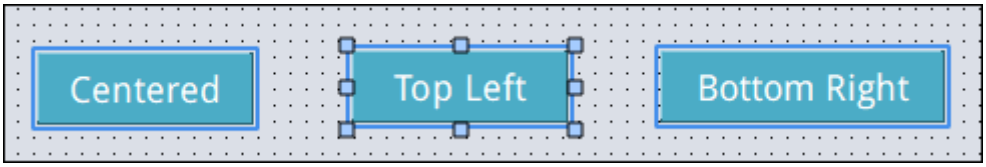
Applicable objects

Supported objects include:

- Basic elements: Line, polyline, arc, rectangle, circle, ellipse, sector, circular segment, polygon, text, image, and pipeline.
- Control and display: Numeric I/O field, character I/O field, time/date field, button, bit button, character button, bit status display, character status display, text list, combo box, list box, text switch, radio button, and checkbox.
- Data visualization: Bar chart, slider, progress bar, circular progress bar, pie chart, and gauge.

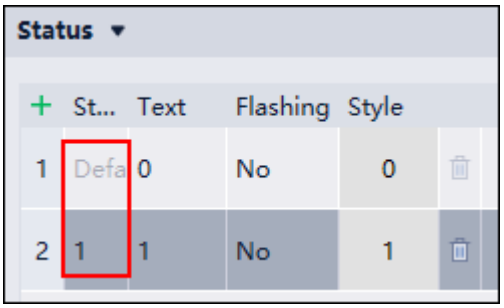
Property range

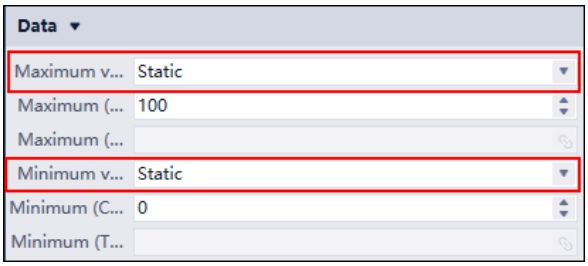
After selecting multiple objects of the same type, the system automatically disables properties that cannot be modified. Users can only modify objects with control points (such as the "Top left" button in the figure below); property changes will automatically synchronize to other selected objects. Modifying other objects will not trigger automatic synchronization.



- Individual objects
  Before batch modifying certain properties of an individual object, specific rules must be met. For example, the "Status" property of a text button requires that the selected objects have identical status values, counts, and order; otherwise, batch modification is not allowed. The following table lists objects with similar property requirements.

Table13-3-7-20-1

Object	Property	Batch Modification Condition
Text button Text indicator		Batch modification of status-related information is supported only when the status values, count, and order must be identical across selected objects.

Object	Property	Batch Modification Condition
Pie Chart		Batch modification of color is supported only when the number of channels must be the same across selected objects.
Bar Graph Slider		Batch modification of maximum /minimum values (static) is supported only when the values are static across all selected objects.

- Group
  When batch modifying multiple groups, consistency in member count or member type is not required.
  The properties that support batch modification include: basic settings and basic information.
  For members of the same type in a group, the batch modification method are the same as for as individual objects.

13.3.8 Animation Configuration

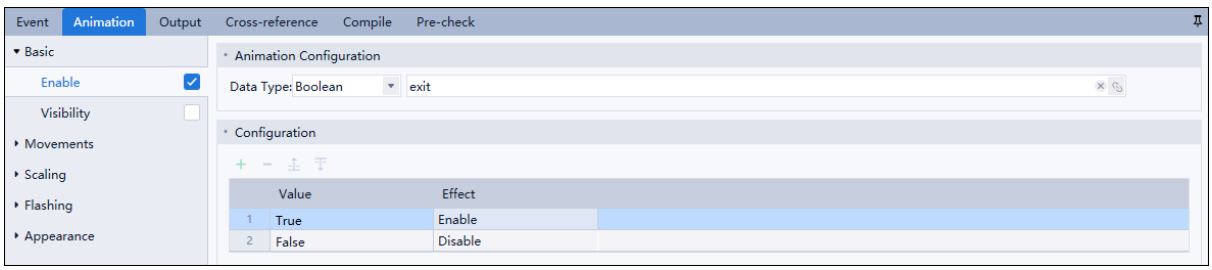
- 13.3.8.1 Basic Animation
- 13.3.8.2.1 Movement Animation
- 13.3.8.3 Rotation Animation
- 13.3.8.4 Scaling Animation
- 13.3.8.5 Flashing Animation
- 13.3.8.6 Appearance Animation

13.3.8.1 Basic Animation

This tab is used to dynamically control the visibility and operability of basic animation objects.

In the runtime, visibility and operability is determined by the configuration variable values.

The following shows the configuration interface for a basic animation object that is enabled:

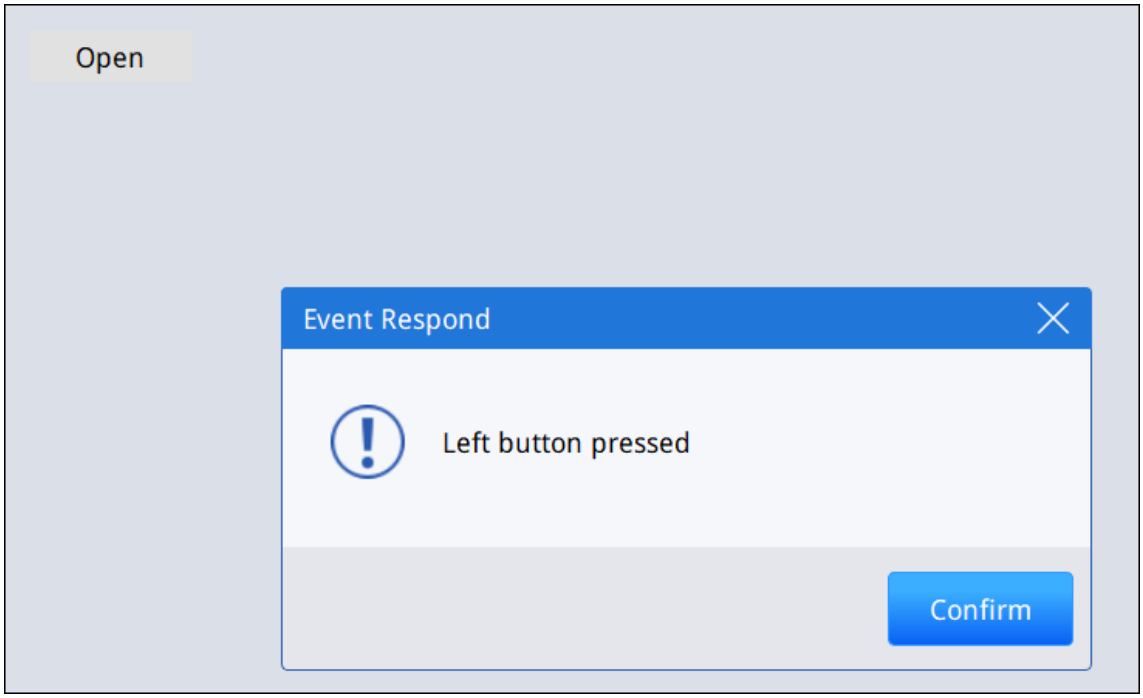
- 

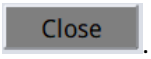
  Configuration is mainly divided into the two parts:
  - Animation configuration: Configures the type of configuration variables, which supports boolean (default), bit, and range (integer).
    - A "Boolean" variable configures the effect of "True" (enable) and "False" (disable).

- A "Bit" variable configures the effect of "1" (enable) and "0" (disable).
- When the variable type is "Range (integer)", click "+" to add a new row, and set the effect of the corresponding "Value" (enable or disable).

- Configuration: Enables/disables the graphic element by selecting TRUE/FALSE. TRUE: Enabled by default; FALSE: Disabled by default.

- Example:
  - Detailed steps for enabling an object are as follows:
    1. Select a button object, open the "Animation" interface under its property view. Check "Enable" to activate the animation configuration.
    2. Add an event. Select "Press > Screen > Show message box", and enter "Left button pressed".
  - After initiating onsite monitoring, the display effect in the simulation environment is as follows:
    - When exit = TRUE, pressing the button shows a "Left button pressed" dialog, as shown below:



    - When exit is FALSE, the button element is grayed out and becomes non-operational 

.

- For details about the "Left button pressed" dialog configuration, see 13.3.7.14 Adding an Event.

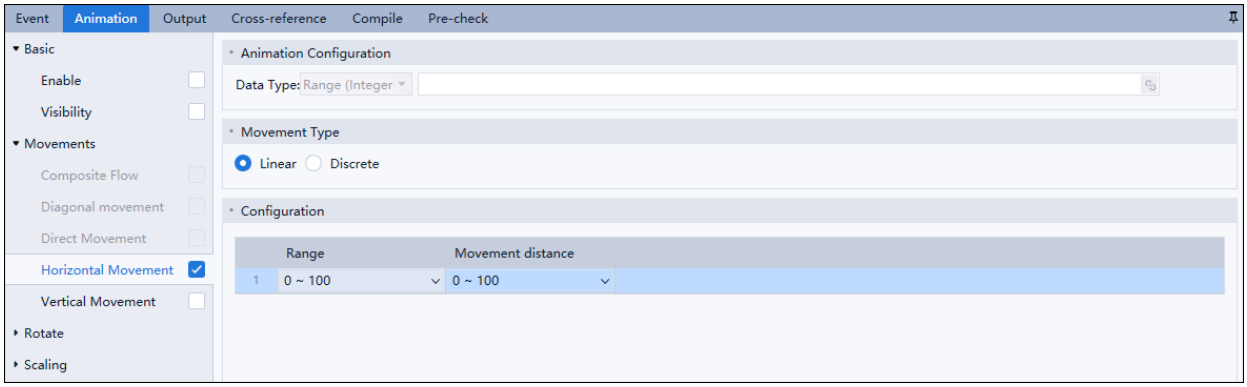
13.3.8.2 Movement Animation

13.3.8.2.1 Movement Animation

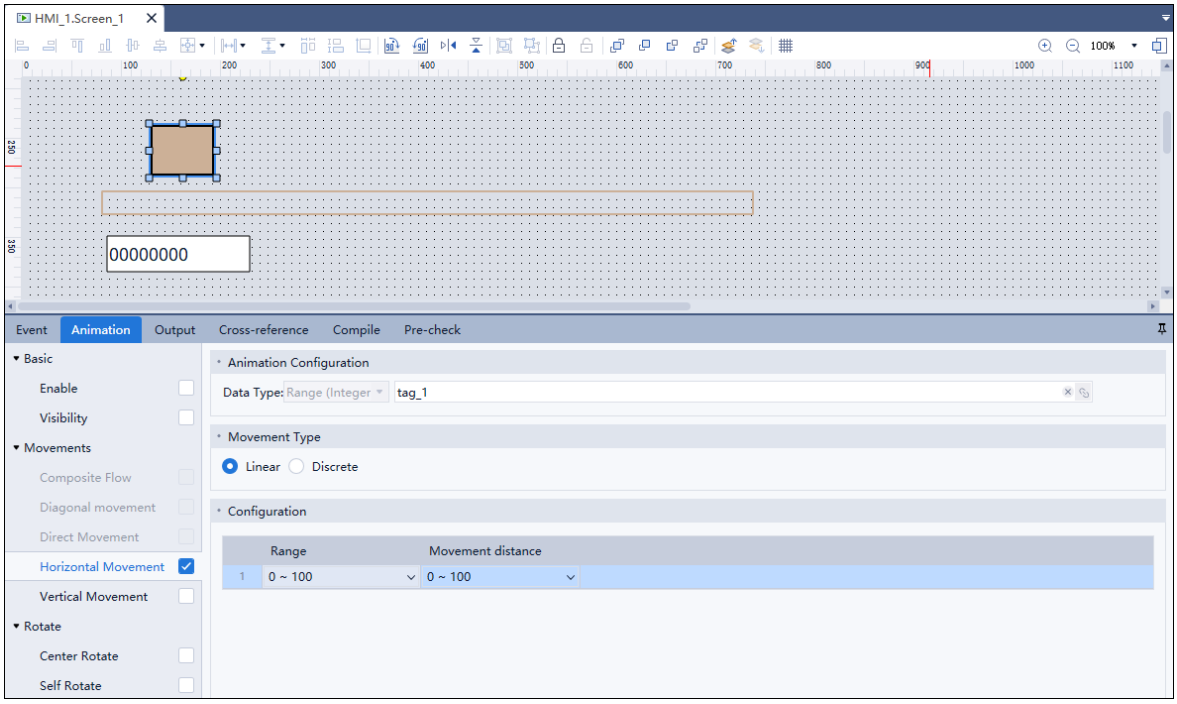
Configuring the movement of an object allows it to move along a specific trajectory. The object movement is controlled by variables. When the variable changes, the object moves along the configured trajectory.

Movement types include: combined movement, diagonal movement, direct movement, horizontal movement, and vertical movement. Horizontal and vertical movements can be used together; the other types can only be used independently.

The following is the configuration interface for "Horizontal movement":

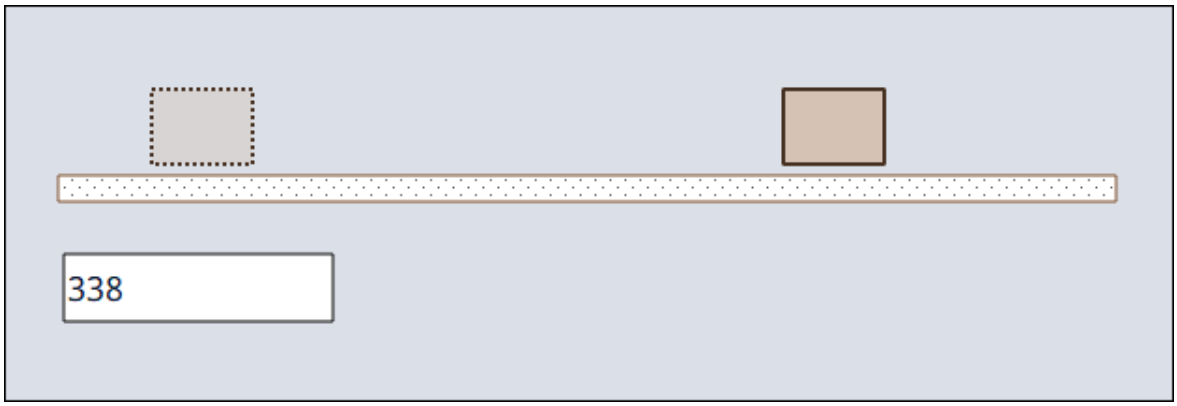


- The configuration consists of three parts:
  - Animation configuration:
    - Data types: Supports range (integer), bit, and boolean. The variable data type supports integer and boolean.
      The configuration here is related to the movement type (if applicable).
  - Movement type:
    - Linear: The object moves continuously along a straight line, with a smooth trajectory, no jumps or pauses.
    - Discrete: The object moves between discrete points, with noticeable intervals or jumps between each step; the movement trajectory is discontinuous.
  - Configuration: Specifies the relationship between variable (bit) values and movement distance.
- Example:
  In the following example, the horizontal movement of a rectangle object is configured. For details, see the following steps.
  1. Select the rectangle object, then expand the "Animation" property in its properties view, and select "Movement > Horizontal movement".
  2. Configure items in "Animation configuration", and select the variable to control object operability, such as tag1.
  3. Configure items in "Configuration" and define the relationship between the variable value (tag1) and the movement distance. The following example shows a 1:1 relationship, meaning the movement distance changes by the same amount as the value change.



4. Enable simulation.

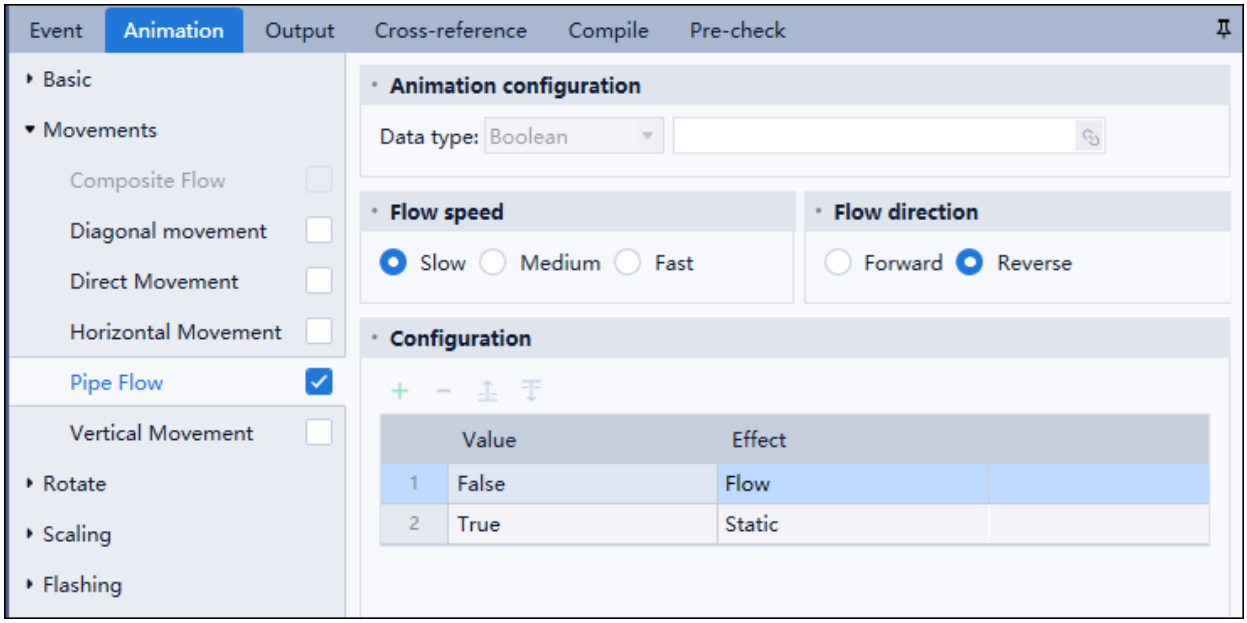
   Enter "338" in the control for variable "tag1", the movement animation is shown in the figure below:



13.3.8.2.2 Pipe Flow

"Pipe flow" animation is a dynamic visualization effect in flowcharts specifically designed for pipeline graphic objects. It simulates the liquid flow state in a pipe, enhancing the intuitiveness and real-time capability of process monitoring.

The following shows the configuration interface for "Pipe flow" animation:



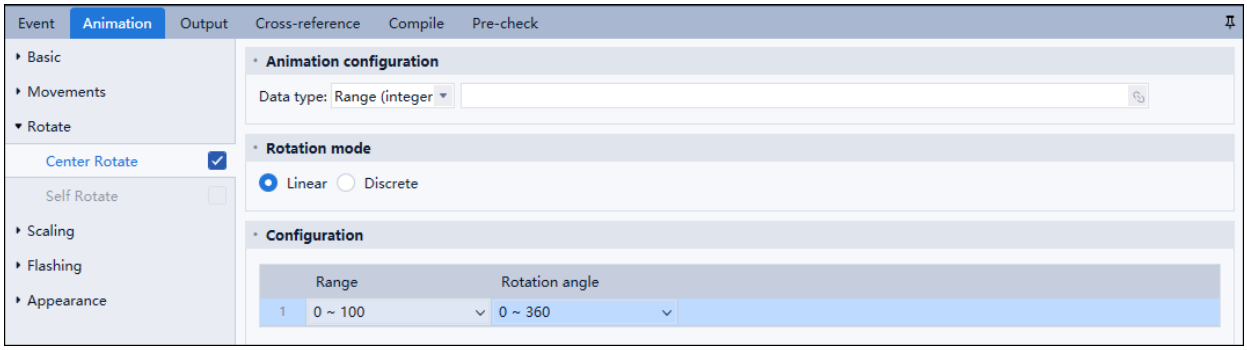
The configuration consists of four parts:

- Animation configuration: Associates one boolean variable or expression to control whether the liquid flows in the pipe (start/stop).
- Flow speed: The speed at which the liquid flows.
- Flow direction:
  - Forward: Liquid flows from left to right or from top to bottom.
  - Reverse: Liquid flows from right to left or from bottom to top.
- Configuration: Sets the animation effect corresponding to the value of the variable or expression.

13.3.8.3 Rotation Animation

20241105113943acaa8d58c16ab49536e915bd49b78844-EN.pngDynamic objects can rotate around a specific point at a specific angle. Users can set the rotation point and angle based on variables.

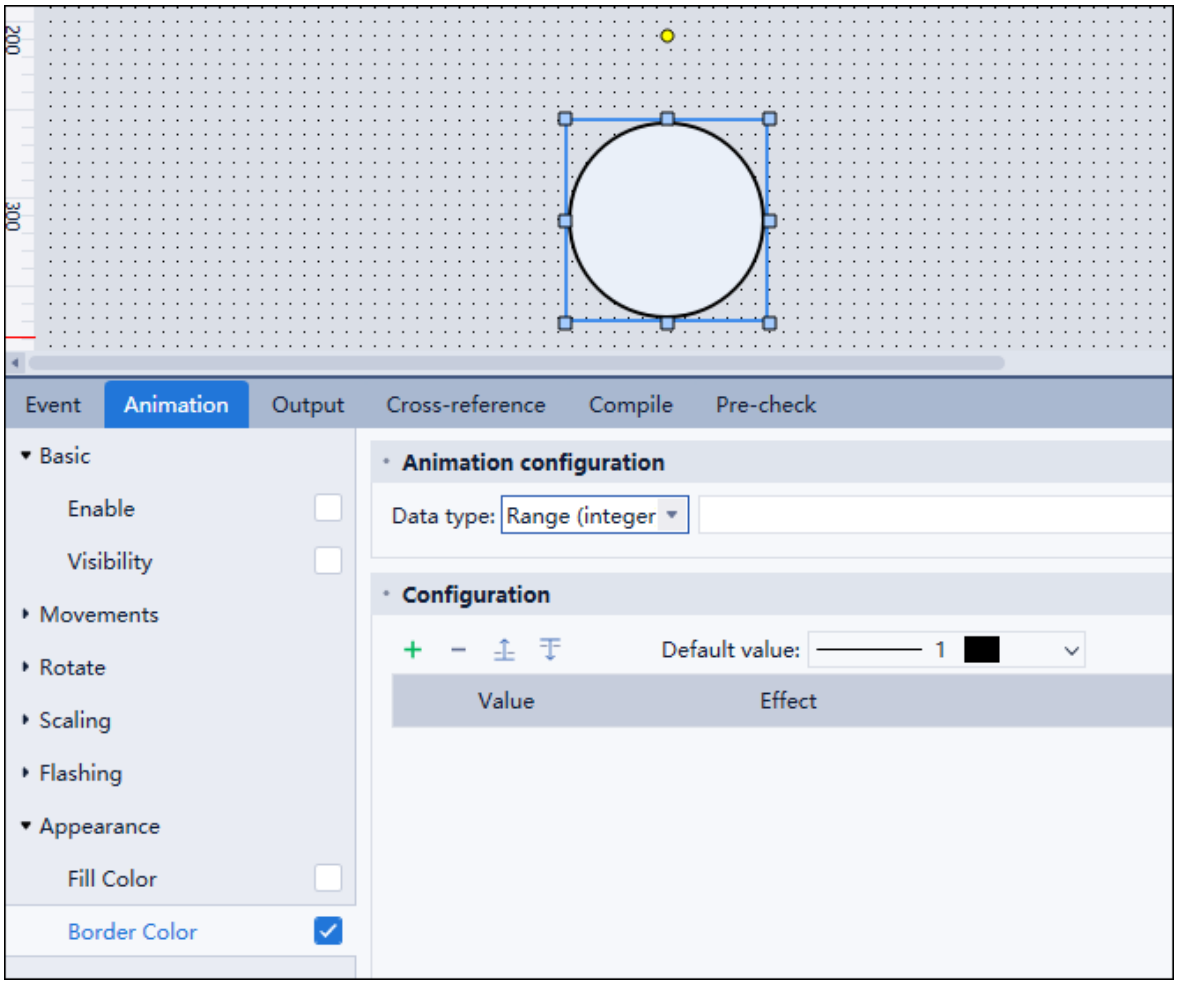
The following is the "Center rotate" configuration interface for an object:



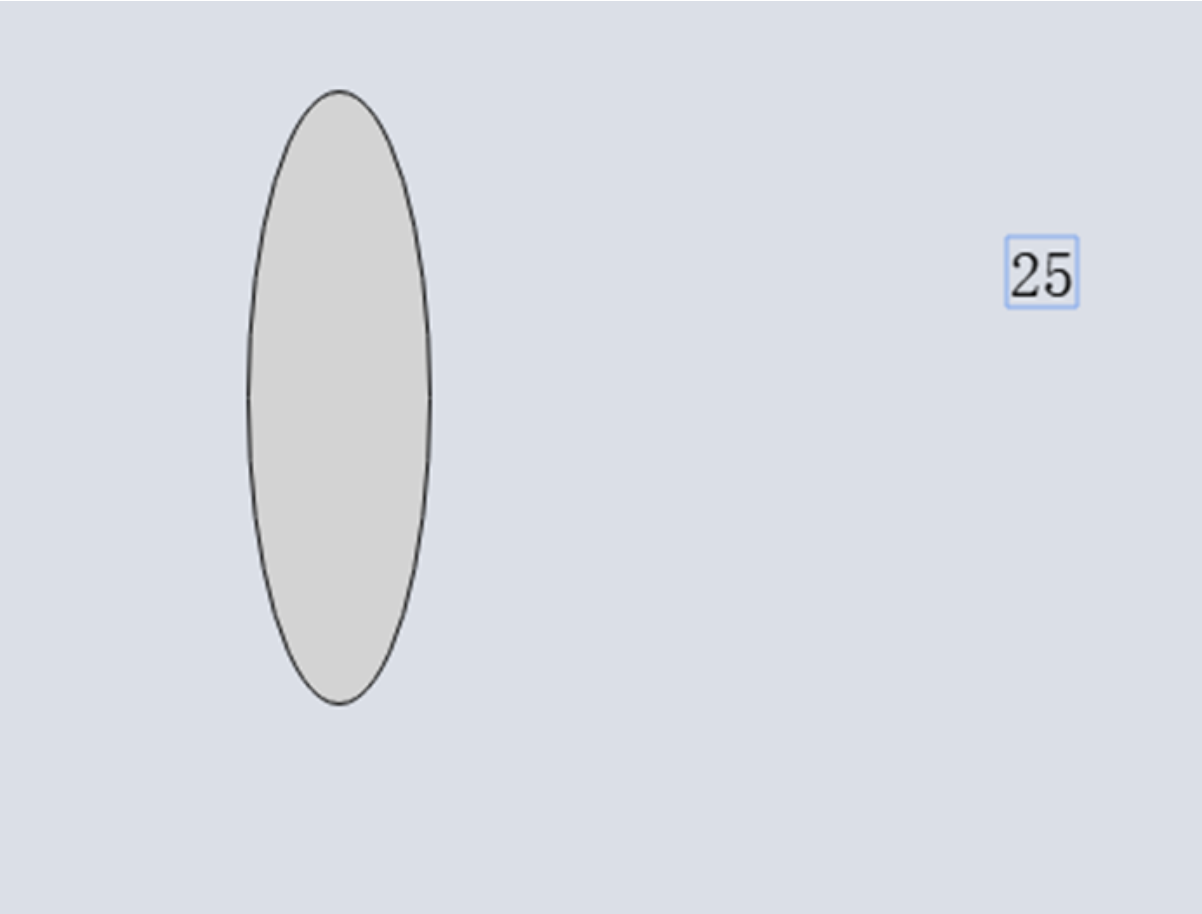
- The configuration mainly consists of the following parts:
  - Animation configuration:
    - Data type: Specifies the data type of the variable, which is fixed to range (integer) and cannot be modified.
    - Variable: Specifies the relative position of the object along the path, supporting integer and boolean types.

- Rotation mode: Specifies the rotation mode of the object, including linear and discrete.
  - Linear: Configures the relationship between input value and rotation angle (start/end value and start/end angle).
  - Discrete: Users need to configure the central rotation table, including comparison operators, values, rotation angles, default values, and so on.
- Configuration:
  - Range: Specifies the start/end angle value for the variable.
  - Rotation angle: Specifies the start/end angle for the variable.

- Example:
  - The following procedure shows how to configure central rotation for an ellipse object.
    1. Select the ellipse object, expand the "Animation" tab in the monitoring area, and select "Central rotation".
    2. Configure the variables on the "Animation" tab. Select the variable to control the object's operability from the "Variable" drop-down list. Select the data type of the variable (in this example, x is a 16-bit integer variable).
    3. Configure "Range" and "Rotation angle".
  - The configuration is as follows:



  - After starting onsite monitoring, the effect shown in the simulator is as follows:
    Enter "25" in the I/O field, the ellipse rotates 180 degrees around its own center.

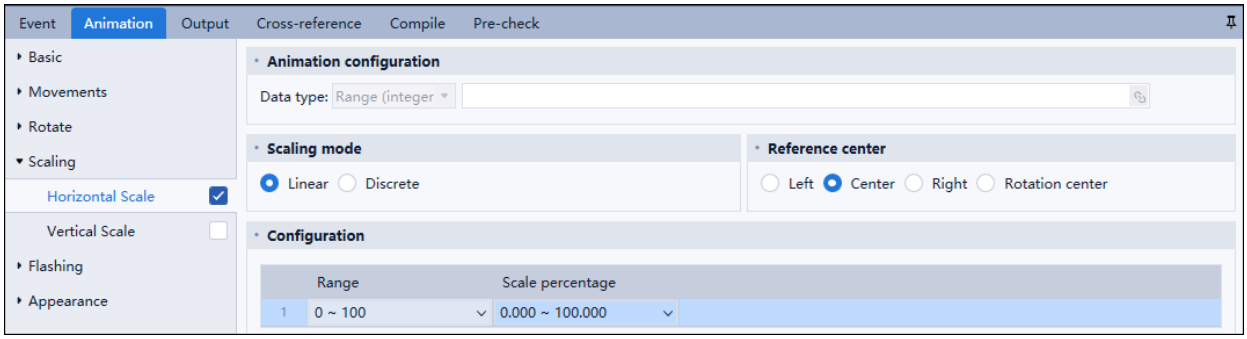


13.3.8.4 Scaling Animation

Scaling animation makes an object to be scaled around a specific center. The scaling is controlled by variables. When the variable changes, the object scales according to the set ratio.

Individual elements support scaling animations, while elements in grouped objects and symbol templates do not.

The following is the configuration interface for "Horizontal scale":

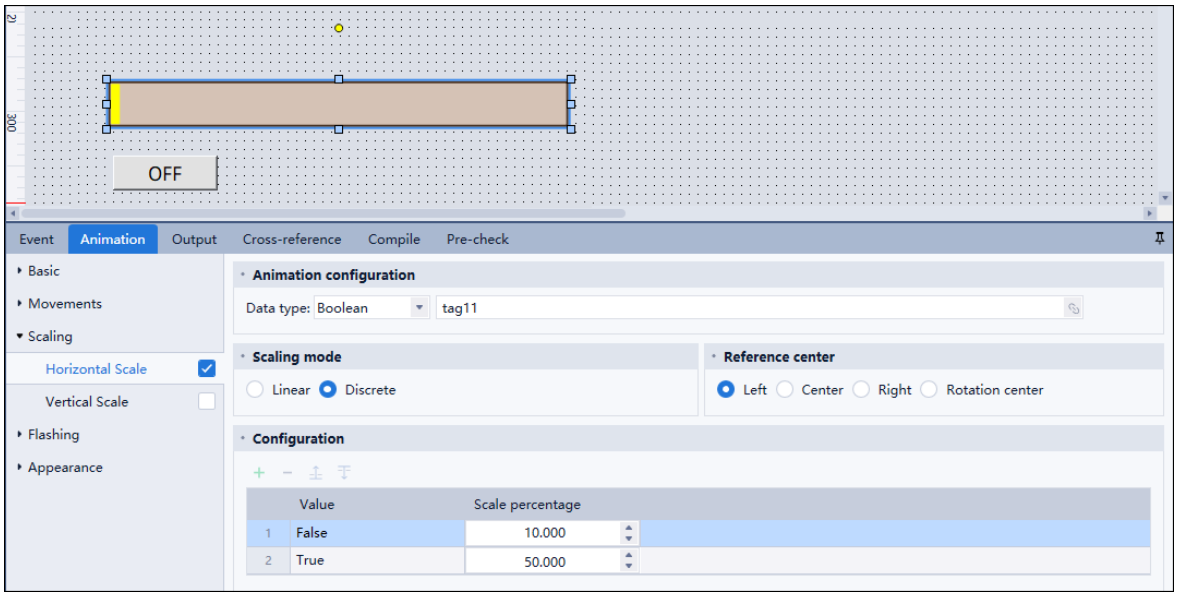


The configuration consists of four parts:

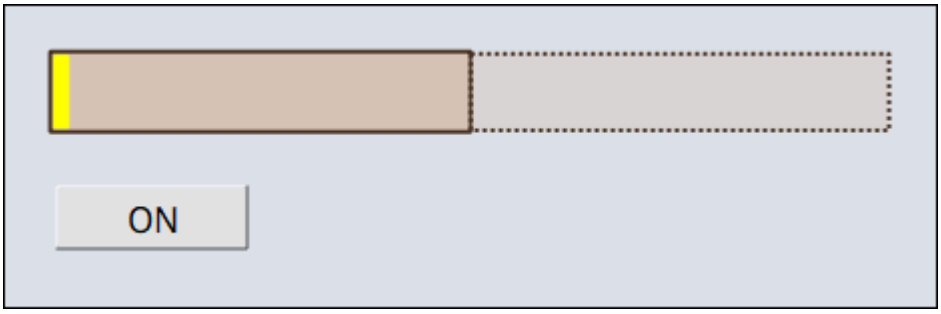
- Animation configuration:
  - Data types: Supports range (integer), bit, and boolean. The variable data type supports integer and boolean.
    The configuration here is related to the scaling mode.
  - Scaling mode:
    - Linear: The scaling ratio changes continuously, with no jumps or pauses in the trajectory.

  - Discrete: The scaling ratio is discontinuous, with noticeable intervals between operations and a discontinuous trajectory.
- Reference center: Specifies the point that remains fixed when the object is scaled, around which the object's shape will be adjusted.
- Configuration: Specifies the relationship between variable (bit) values and scaling ratio.

- Example:
  In the following example, the horizontal scaling of a rectangle object is configured. For details, see the following steps.
  1. Select the rectangle object, then expand the "Animation" property in its properties view, and select "Scaling > Horizontal scaling".
  2. Set the "Scaling mode" and "Animation configuration", and select the variable to control the object's operability.
  3. Set the "Reference center" to the left boundary.
  4. Define the relationship between the variable value (tag11) and scaling ratio. In the example below, when the value is True, it is scaled down to 50% of the original; when the value is False, it is scaled down to 10% of the original.



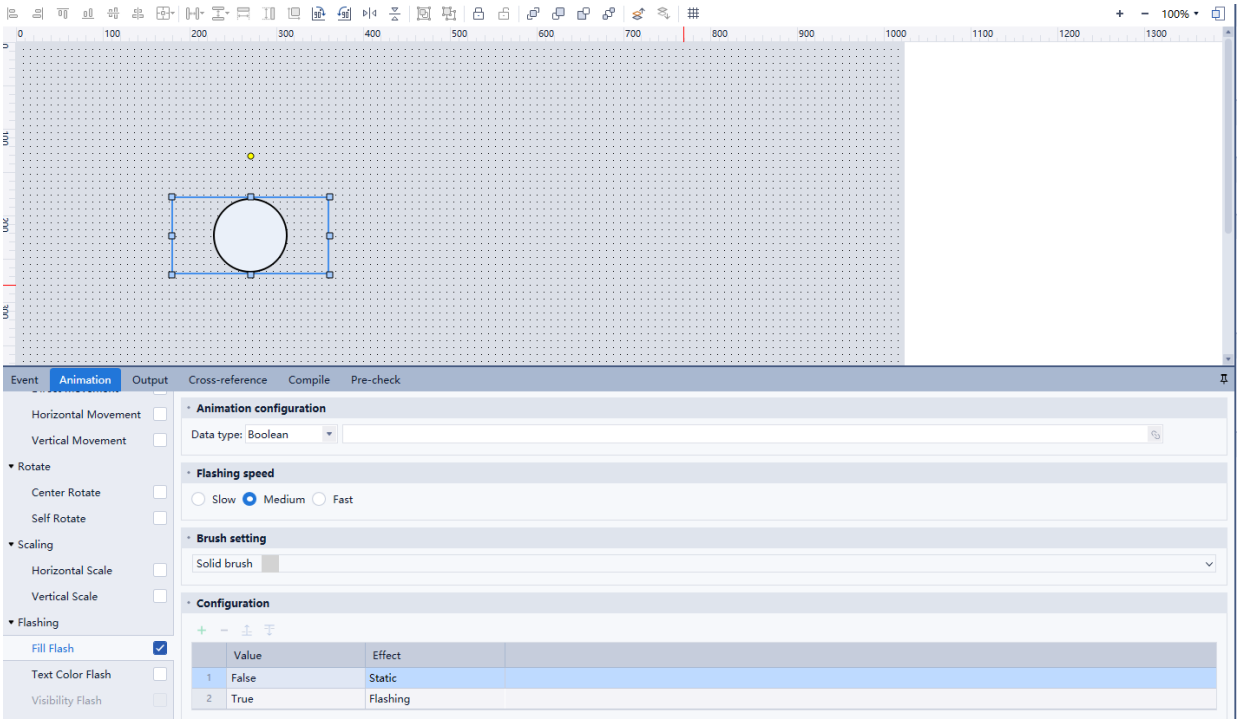
  5. Start simulation.
     When the control for the variable "tag11" is switched to **ON**, the rectangular object is centered on the left boundary and scaled to 50%. The figure below shows the zooming results on the simulator.



13.3.8.5 Flashing Animation

Flashing animation makes the connected element flash when the condition expression is true. There are three flashing types: fill flashing (or color flashing), border flashing, and visibility flashing.

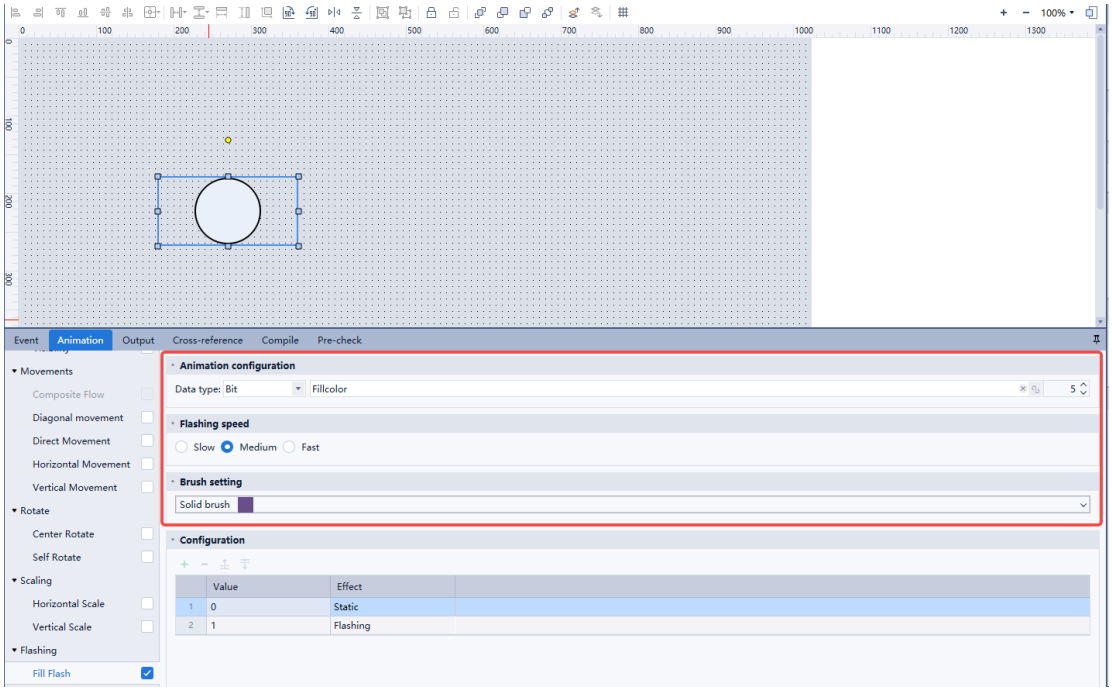
The following is the "Fill flashing" configuration interface for a circle object:



- The configuration mainly consists of the following parts:
  - Animation configuration: Sets whether the element flashes. Supports boolean (default), bit, and range (integer).
  - Flashing speed: Sets the flashing speed of fill color (or pattern). Includes slow, medium (default), and fast.
  - Brush configuration: Sets the brush type and color.
  - Configuration: Sets the color corresponding to the variable or bit value.
- Example:

  To configure the "Fill flashing" properties for the circle object, follow the steps below:

  1. Select the circle object, expand the "Animation" tab in the monitoring area, and select "Flashing > Fill flashing".
  2. Configure "Animation configuration" to specify the data type and the variable of the associated control object that can be operated.
     For example, configure the animation to change based on bit 5 of the INT variable Fillcolor.
  3. Set "Flashing speed" to "Fast".
  4. Set "Brush setting" to a solid color brush with purple fill.



5. Configure the animation effect.

   The animation effect remains at the default setting.

   Result:

   During online operation, the fill color of the circle object changes based on the bit 5 of the Fillcolor variable.

   When the value of Fillcolor is 32 to 63, 96 to 127 (that is, the bit 5 is 1) ..., the area is filled with purple and flashes quickly.

   When the value of Fillcolor is 0 to 31, 64 to 95 (that is, the bit 5 is 0) ..., the default fill color is displayed statically.

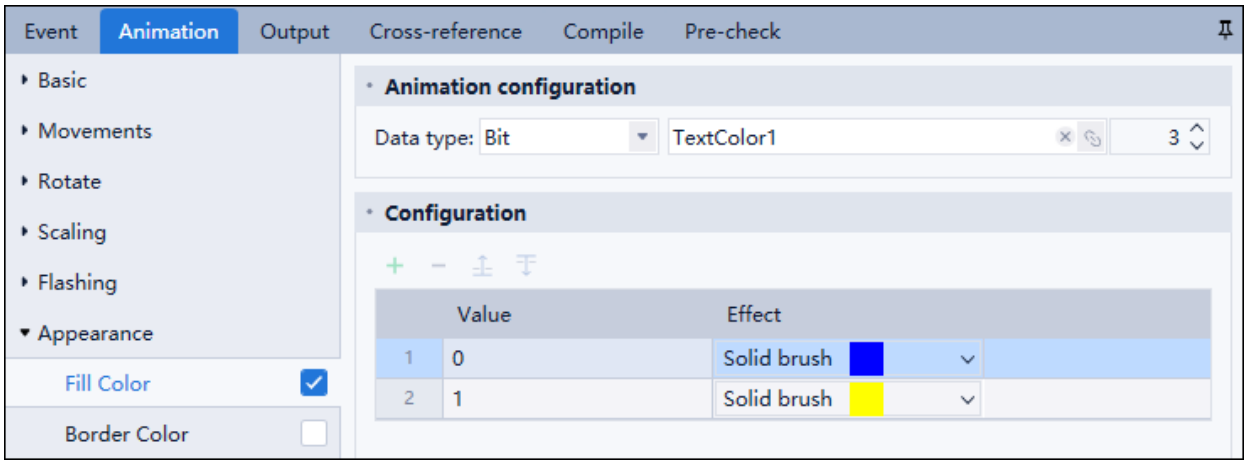
13.3.8.6 Appearance Animation

Appearance animation allows appearance properties such as the border color, fill color, background color, and text color of elements to change dynamically and automatically based on specified conditions (e.g., variable values). The supported properties vary among different elements. See the table below for the supported properties for commonly used elements.

Element	Border Color	Fill Color	Background Color	Text Color
Line	√	×	×	×
Polyline	√	×	×	×
Arc	√	×	×	×
Rectangle	√	√	×	×
Circle	√	√	×	×
Ellipse	√	√	×	×
Sector	√	√	×	×
Circular Segment	√	√	×	×
Polygon	√	√	×	×

Element	Border Color	Fill Color	Background Color	Text Color
Text	×	×	√	√
Numeric I/O field	×	×	√	√
Character I/O field	×	×	√	√
Date-Time Field	×	×	√	√
Button	×	×	√	√

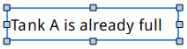
The configuration methods for different areas in the animation are identical. The following uses the "Fill color" configuration of a text object as an example.



- Animation configuration: Configures the type of configuration variables, which supports boolean (default), bit, and range (integer).
  Variable types support boolean and integer.
- Configuration: Sets the color corresponding to the variable or bit value.

Step

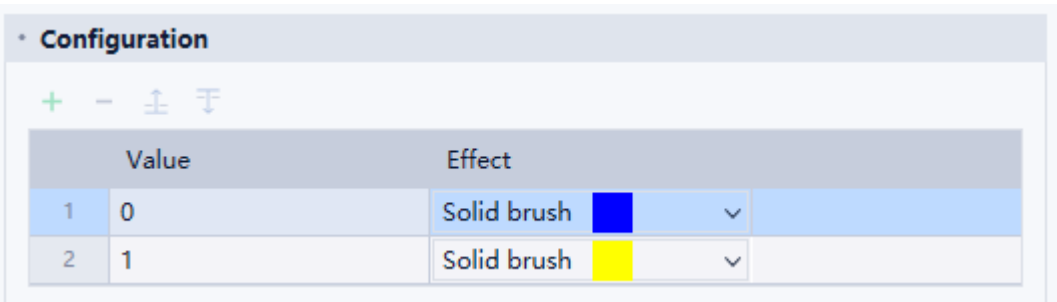
1. Select the screen object to be dynamically controlled. For example, a text box.



2. In the monitoring area, select "Animation > Appearance".
3. Check "Background color" to display the parameter configuration page.
4. Set the data type and associate the variable.
   For example, set the second bit of variable "TextColor1" to control the color.



5. Configure animation colors.



Result:
During online operation, the background color of the text box will change according to the value of the second bit of variable "TextColor1".
When the value of "TextColor1" is 4 to 7, 12 to 15 (that is, the second bit is 1) ..., the text box background is yellow.
When the value of "TextColor1" is 0 to 3, 8 to 11 (that is, the second bit is 0) ..., the text box background is blue.

13.3.9 Layer Usage

- 13.3.9.1 Overview
- 13.3.9.2 Renaming Layers
- 13.3.9.3 Setting Active Layers
- 13.3.9.4 Displaying or Hiding Layers
- 13.3.9.5 Common Operations for Layer Tree

13.3.9.1 Overview

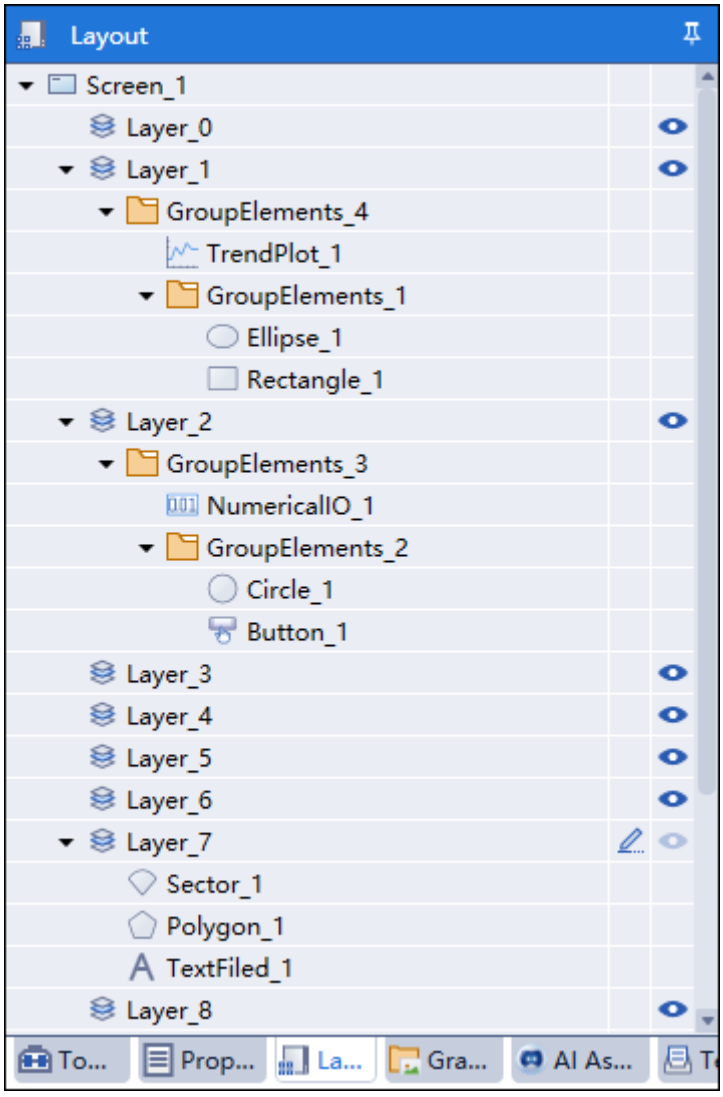
Layers are used to organize, manage, and present different information and data views. With multiple layers, you can group different types of elements (such as text, charts, dashboards, etc.) and display them logically, making the interface clearer and easier to understand.

Main functions of the layer:

- Organization and management: You can organize and categorize information by creating different layers. For example, alarm information, process flow diagrams, and real-time data views during on-site control can be placed on separate layers, making it easier for operators to view and understand.
- Independent operation: Each layer can be edited independently, including adjusting displayed elements, layout, colors, and more. This ensures that modifying a specific part does not affect other parts of the entire interface.

A single screen or symbol template contains 16 layers, numbered 0 to 15, with layer 0 at the bottom (background) and layer 15 at the top (foreground).

Objects within the same layer are arranged in the order they were added, and are displayed in a tree structure within the layout container of the screen or symbol template.

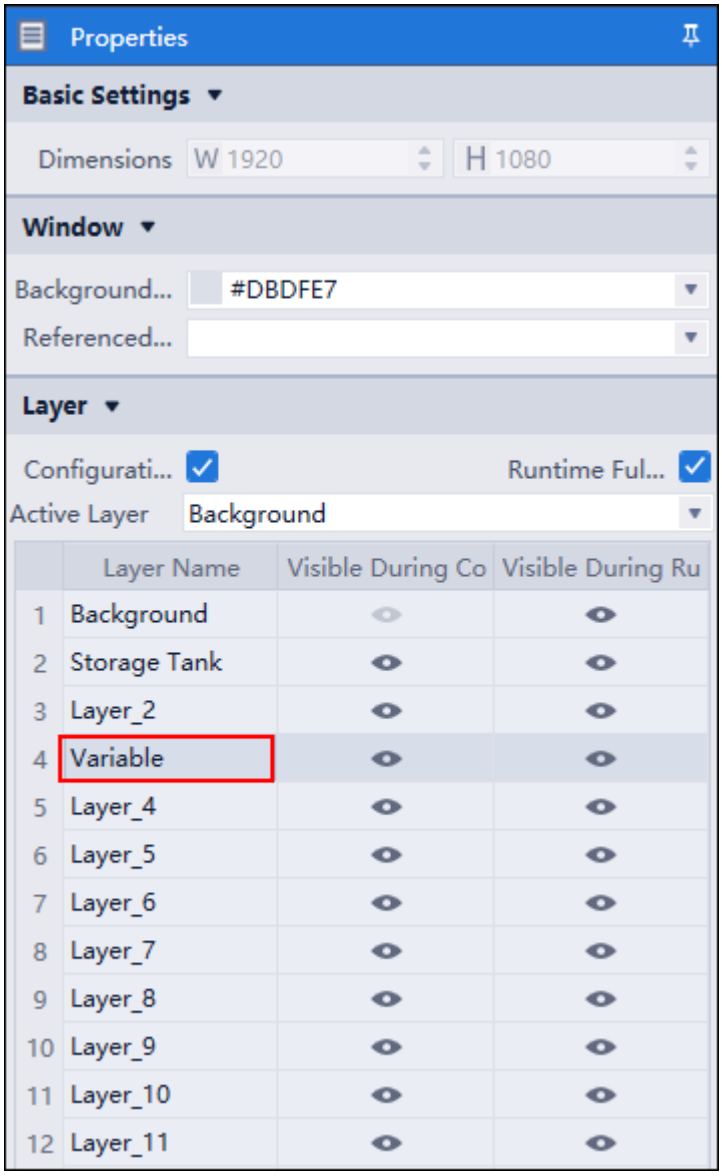


13.3.9.2 Renaming a Layer

The default format for layer names is "Layer_n", where n ranges from 0 to 15. You can customize layer names, which include Chinese, English, numbers, and special characters, but cannot start or end with a space. The maximum length is 128 characters.

Layer renaming can be performed in two different locations. Changes made in either place will be automatically synchronized to the other to ensure consistent naming.

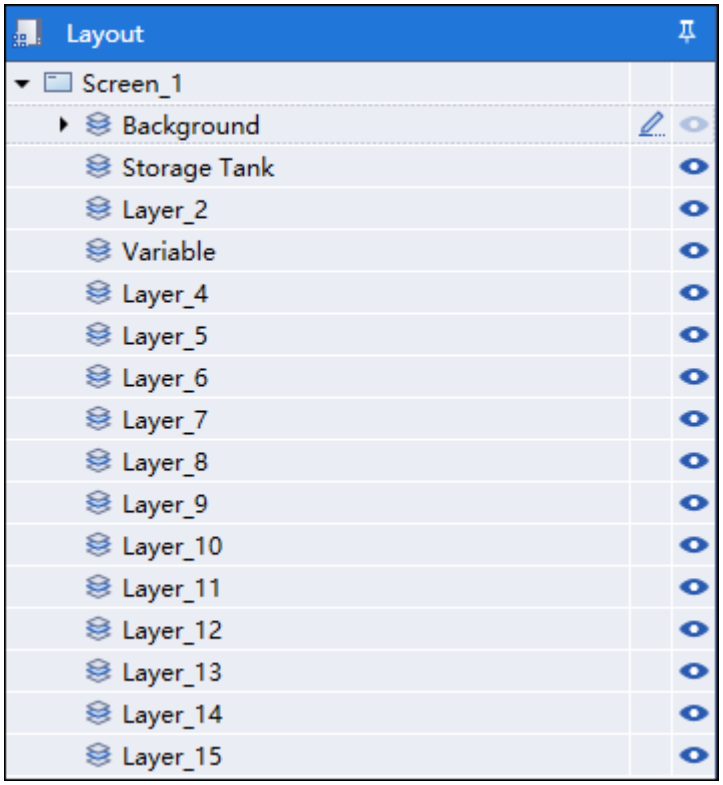
- Property window



Click a layer name, enter the new name, and press Enter to apply.

The new name will be automatically synchronized to the layout container.

- Layout container



1. Select a layer node, press **F2**, or select the "Rename" command from quick menu to enter edit mode.
2. Enter the new name and press Enter to apply.
   The new name will be automatically synchronized to the properties window.

13.3.9.3 Setting an Active Layer

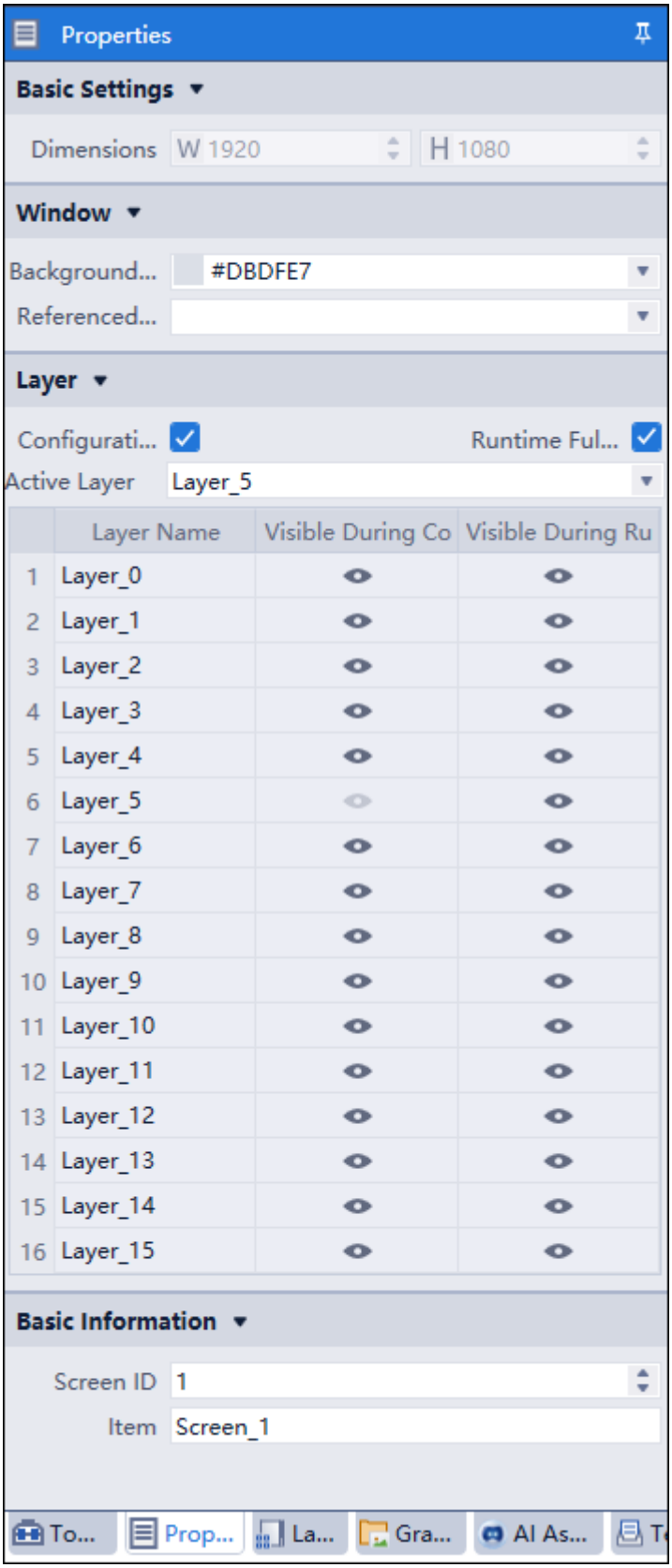
An active layer refers to the layer currently being edited or operated on during offline configuration of a project. When a layer is set as the active layer, you can perform operations on that layer. By setting an active layer, you can focus on the layer currently being edited without affecting operations on other layers.

Layer 0 is the default layer.

There are two ways to set the active layer.

Set in the properties window

1. Double-click the blank area of the screen or symbol template to open the properties window.



2. Set the active layer.

   The active layer is a state during screen configuration. When a layer is set as active, its corresponding configuration indicator appears gray, and hiding the layer is not supported.

   For example, in the figure below, **Layer_5** is the active layer, and the corresponding configuration indicator is gray.

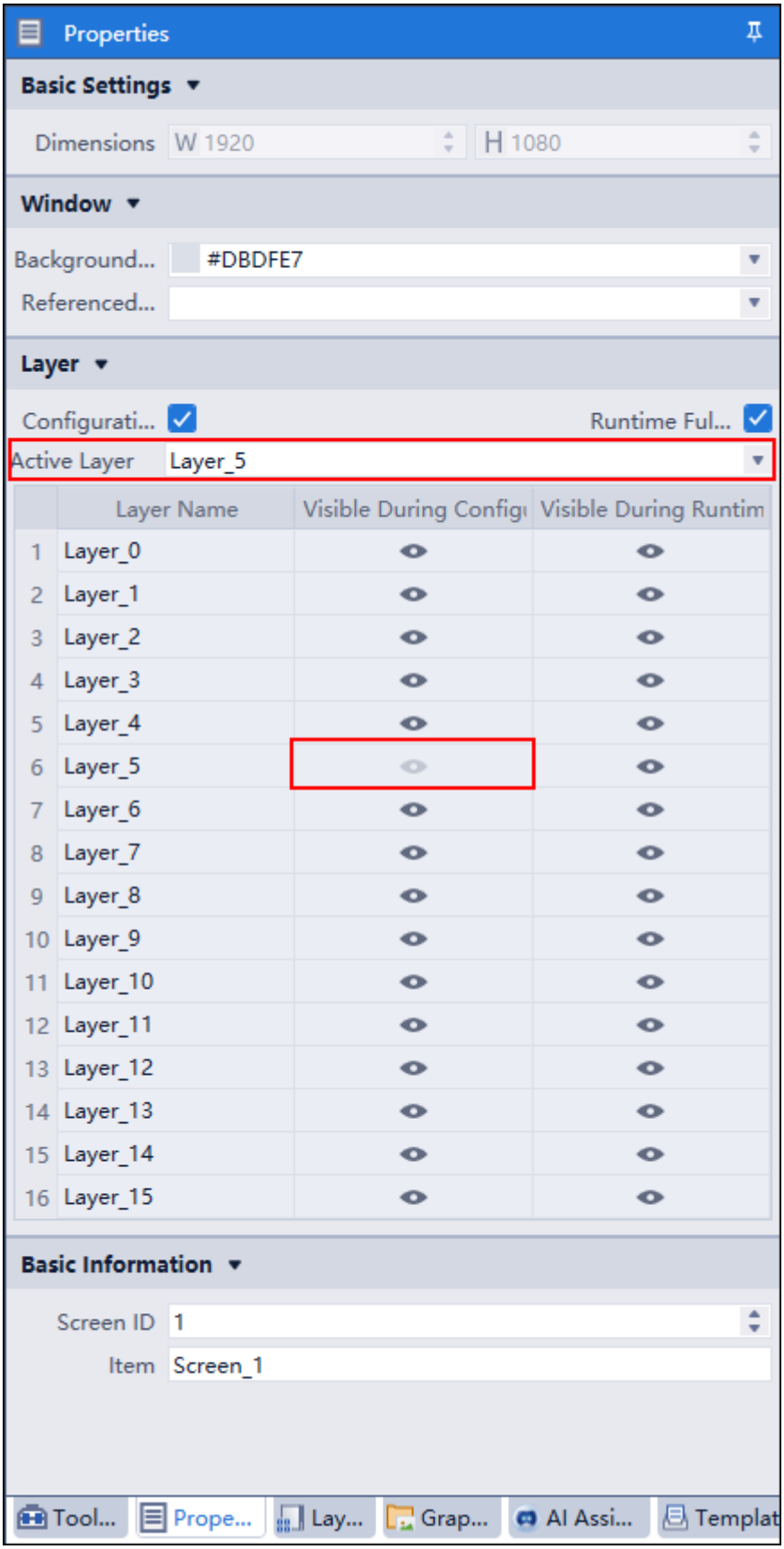


Figure 13-3-9-3-1 Layer_5 active layer

Set in the layout container

1. Open the layout container of the screen or symbol template.
2. Select the shortcut menu for the target layer and select the "Activate layer" command.

   The current active layer is indicated by the icon .

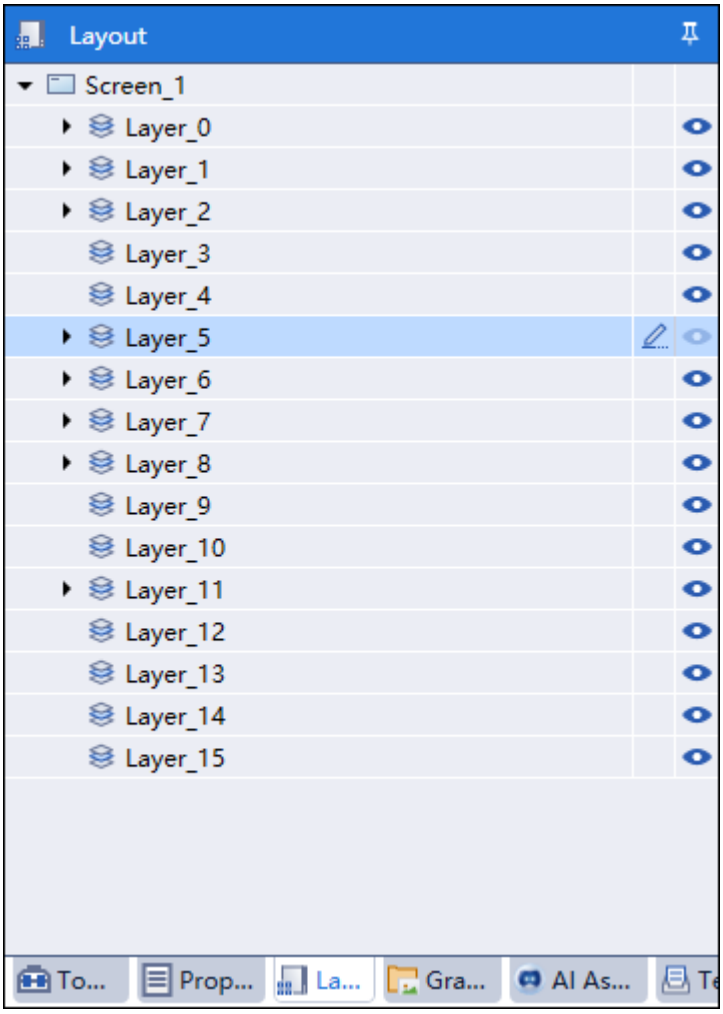


Figure 13-3-9-3-1 Layer_5 active layer

13.3.9.4 Displaying or Hiding a Layer

By selectively displaying or hiding layers, you can focus on the area to be adjusted, reduce visual clutter, and improve work efficiency and design clarity.

Layer visibility settings are supported in both configuration and runtime environments.

Configuration environment

- Single layer

  In the property box or layout box of a screen or symbol template, operate the visibility toggle icon for the specified layer to hide or display the layer and its associated objects.

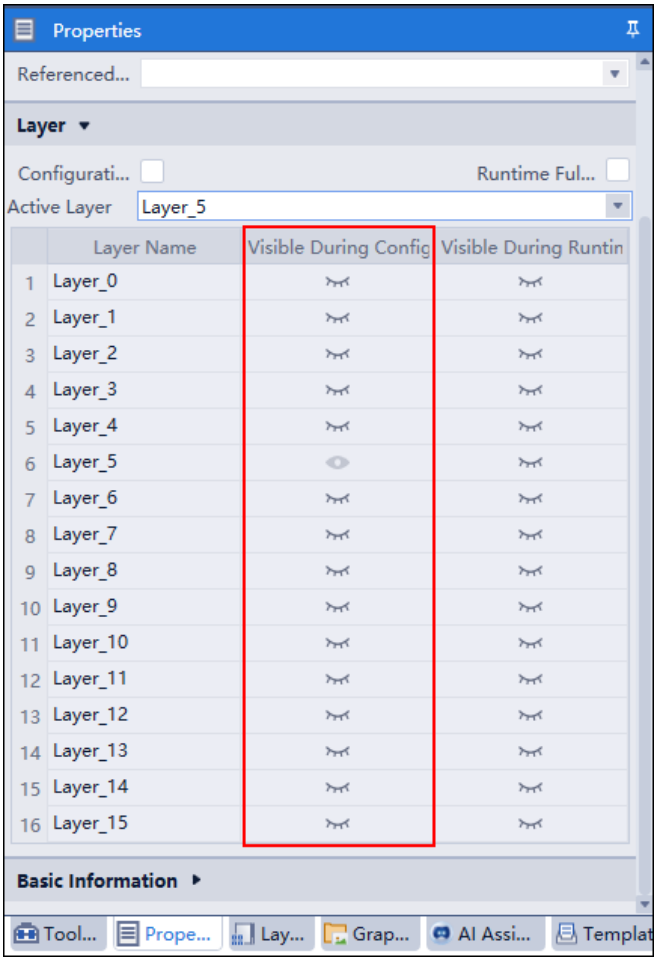


Figure 13-3-9-4-1 Screen properties window

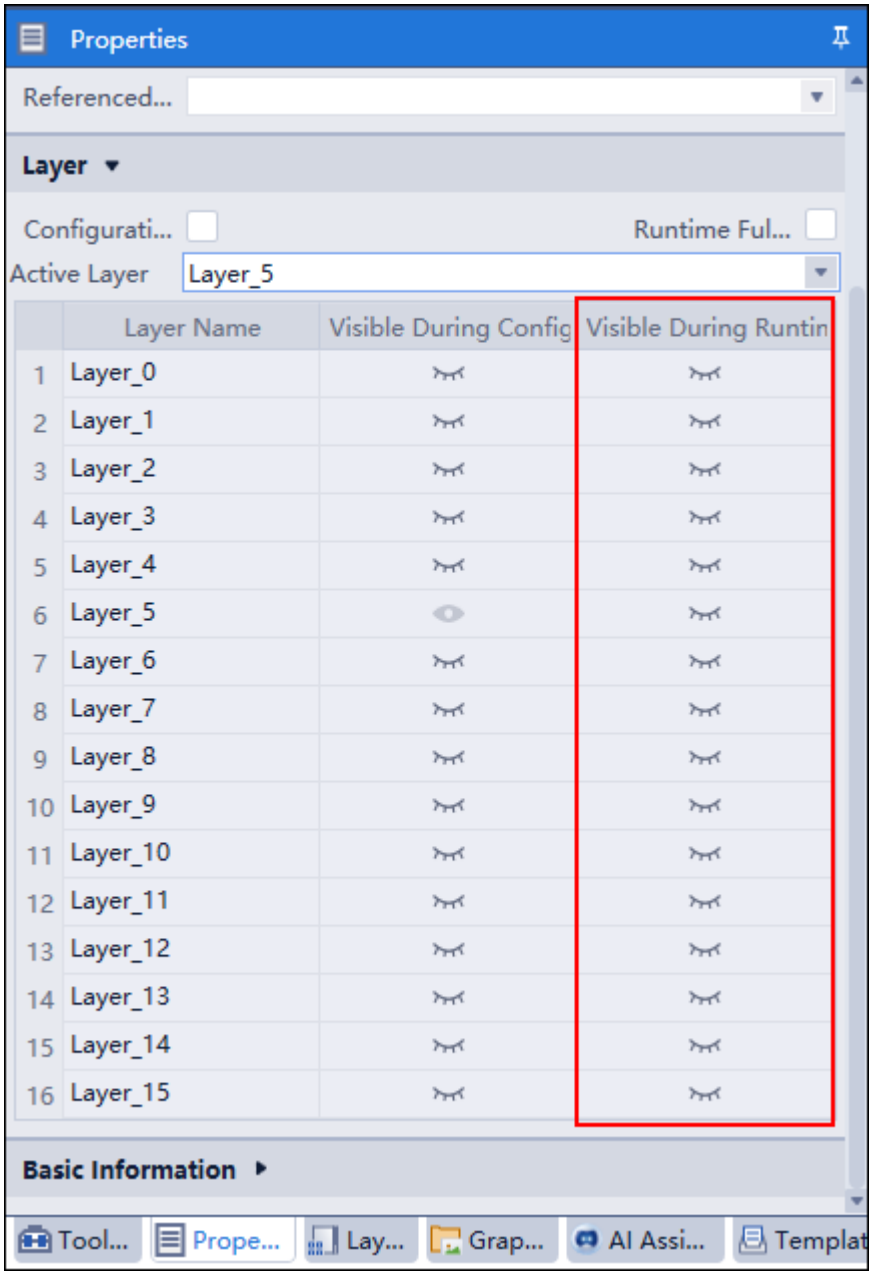
- All layers

  In the property box of a screen or symbol template, operate the radio button "Configuration fully visible" to batch hide or display layers (excluding the active layer) and their associated objects.

Operating environment

- Single layer

  In the property box of a screen or symbol template, operate the visibility toggle icon  for the specified layer to hide or display the layer and its associated objects.



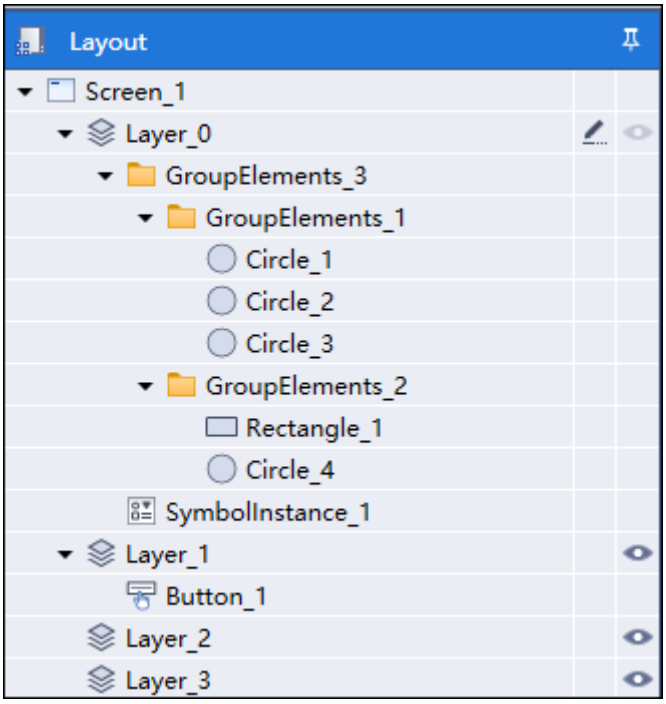
- All layers
  In the property box of a screen or symbol template, operate the radio button "Operation fully visible" to batch hide or display layers (excluding the active layer) and their associated objects.

13.3.9.5 Common Operations for Layer Tree

Objects added to the screen generate child nodes in the corresponding layer. The node name is the object name.

Node naming rules

- Instance of a symbol template: As a whole, the node is named by default as "Symbol instance_n". For example, "Symbol instance_1" under "Layer_0", as shown in the figure below.
- Grouped objects: Displayed as folders, the node is named by default as "Grouped element_n". For example, "Grouped element_1" under "Layer_0", as shown in the figure below.
- Single object: Displayed as a single node, the node is named by default as "Control name_n". For example, "Button_1" under "Layer_1", as shown in the figure below.



Common operations

Object nodes in the layer tree support several common operations, which are described in the 13.3.7 Configuration Screen.

- For operations combining multiple objects, see 13.3.7.9 Grouping Objects.
- For operations to ungroup grouped objects, see 13.3.7.10 Ungrouping Objects.
- For operations to delete objects, see13.3.7.5 Deleting an Object.
- For operations to copy and paste objects, see13.3.7.3 Copying/Pasting an Object.
- For operations to cut and paste objects, see 13.3.7.4 Cutting/Pasting an Object.
- For operations to rename objects13.3.7.2 , see Renaming an Object.
- For operations to change the layer of an object13.3.7.17 , see Moving an Object.

13.3.10 Screen Management

- 13.3.10.1 Screen Grouping
- 13.3.10.2 Zooming a Screen
- 13.3.10.3 Copying a Screen
- 13.3.10.4 Changing Screen Resolution

13.3.10.1 Grouping Screens

The screen configuration function enables visual control of the on-site production process. During configuration, screens can be grouped and managed using screen groups.

Screens and pop-up screens support screen group management.

Create a screen group

Take adding a screen group to a screen as an example:

1. In the project tree, click "Screen > Add group" to create "Screen group_1".

2. Click "Screen group_1 > Add" to add a screen node.

Edit a screen group

Screen group properties can be edited via the right-click menu and keyboard shortcuts: Copy (Ctrl+C), Paste (Ctrl+V), Delete (Delete), and Rename (F2).

Screen groups support undo/redo functionality.

Screen groups support nesting. Screens can be selected and dragged into other groups.

13.3.10.2 Zooming a Screen

The zoom factor for screens in the offline editing environment can be set. When the screen is zoomed, all objects inside it are zoomed proportionally.

The zooming scope can be configured in the iFA software.

Supported screen types include
Main screen, picture-in-picture, pop-up screen, template screen, and symbol screen.

Set the zoom factor for a single screen

1. Select the "Tools > 6.2 Options" command in the iFA software to open the configuration window.
2. On the "Screen" tab, uncheck the "Globally effective" option.
3. Select the zoom button and adjust the display ratio of current screen.

The zoom factor of each screen will then remain independent of each other.

Set the zoom factor for multiple screens

1. Select the "Tools > 6.2 Options" command in the iFA software to open the configuration window.
2. On the "Screen" tab, check the "Globally effective" option.
   Note: When switching from "unchecked" to "checked", the zoom factor of all screens automatically resets to 100%.
3. Select the zoom button and adjust the screen zoom factor.

When the display zoom of any screen is adjusted, the zoom factor of all screens in the project is updated synchronously.

13.3.10.3 Copying a Screen

The screen can be copied and pasted across different projects.

Applicable range

Applicable screen types include:

- Screen
- Picture-in-picture
- Pop-up screen
- Template Screen

> **NOTE**
>
> This feature applies to projects created by iFA Evolution.

Procedure

1. In the project tree of the source project, select the screen to copy.
2. Click "Copy" in the right-click menu.
3. In the project tree of the target project, select the "Paste" command from the right-click menu.
   After pasting, the copied screen retains its original name by default. If there is a screen with the same name in the target project, the system will automatically add a number to it.

Precautions

- Device compatibility
  - If the device model of the target project is different from the source project, check and manually adjust the screen layout.
  - "Screen" and "Template screen" will automatically adapt to the new device size. "Picture-in-picture" and "Pop-up screen" will adjust according to the settings under "Tools > Options".
  - The text font size of all screens will be automatically adjusted based on the preset value for the screen adaptation method.
- Processing of associated templates
  - When copying a screen, its associated template screen is not copied.
  - If a template screen with the same name exists in the target project (even if its content differs), the copied screen will automatically associate with that template. Please check whether the association is correct.

13.3.10.4 Changing Screen Resolution

When the HMI device model in the project changes (e.g., being replaced with a different model) or the device orientation changes (e.g., from landscape to portrait), the resolution of the screen will change. You need to check and adjust the screen layout to ensure proper display.

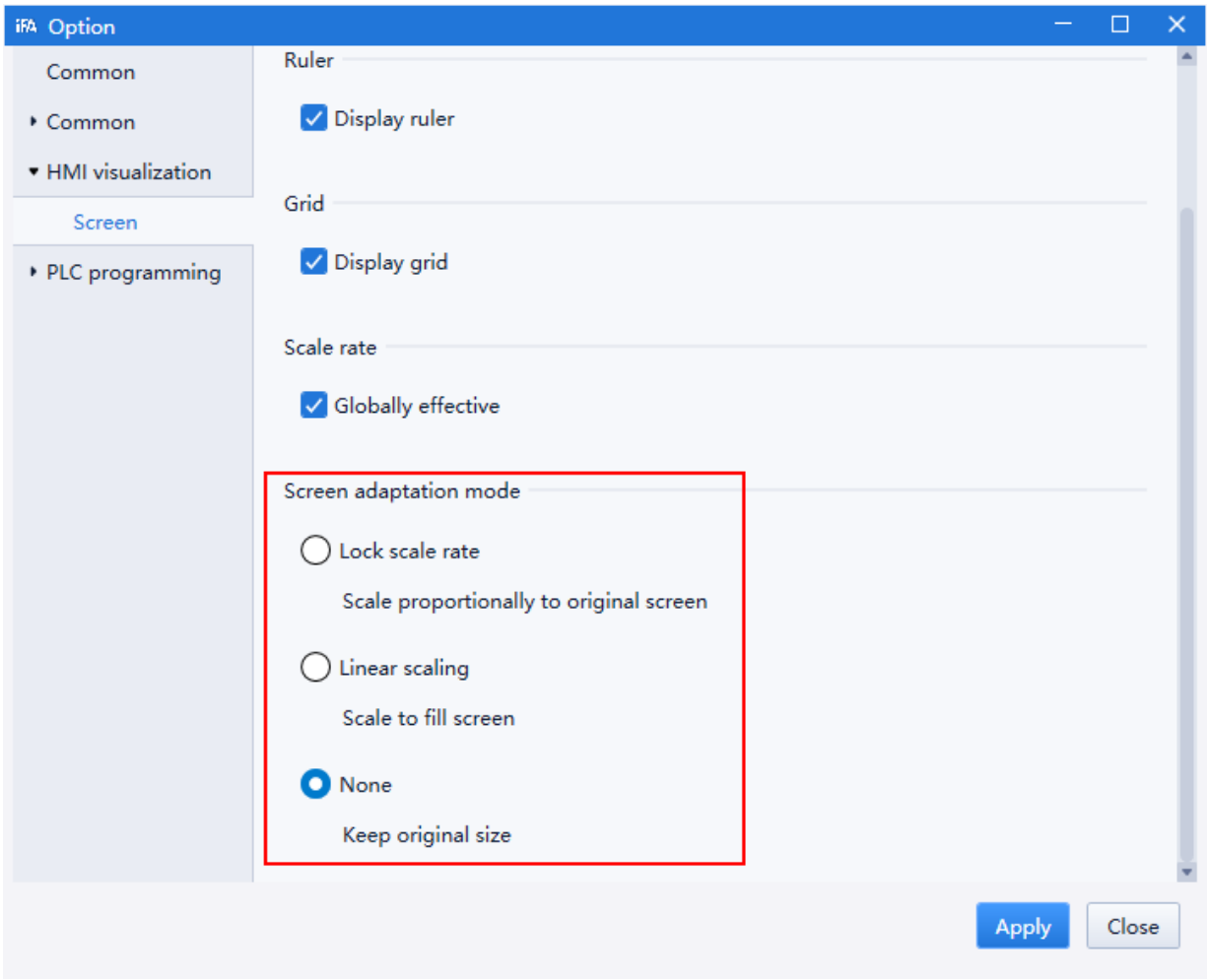
Auto-adjustment of the screen

Screen Types	Adjustment Mode
Screen	Automatically adapts to new device or new orientation.
Template Screen	Automatically adapts to new device or new orientation.
Picture-in-picture	Adjusted according to settings under "Tools > Options".
Pop-up screen	

The text font size of all screens will be automatically adjusted based on the preset value for the screen adaptation mode. The following describes the settings of the screen adaptation mode.

Scaling mode comparison

The adaptation methods for "Picture-in-picture" and "Pop-up screen" can be set under "Tools > Options". There are three options; please select one according to the following instructions.



Adaptation Method	Feature	Applicable Scenario
Lock scale rate	Maintains the screen's aspect ratio; blank areas may appear around the edges.	Prioritizes preserving the aspect ratio; blank margins are acceptable.
Linear scaling	Fills the entire screen area, but the screen may become distorted.	Prioritizes using all available screen space; distortion is acceptable.
/	Maintains the original size; some content may extend beyond the screen.	Requires maintaining original size and layout.

13.4 Template Library Usage

- 13.4.1 Introduction
- 13.4.2 Folder Usage
- 13.4.3 Symbol Template Usage
- 13.4.4 User-Defined Types
- 13.4.5 Template Status Management

- 13.4.6 Common Operations on Template
- 13.4.7 Copying and Pasting a Template Across Projects
- 13.4.8 Copying and Pasting a Template instance Across Projects

13.4.1 Introduction

During the configuration phase, resources with common characteristics are encapsulated into libraries and published as template libraries for reuse across multiple projects in the form of template libraries. Cross-project use must be achieved via the cross-project copy and paste template function.

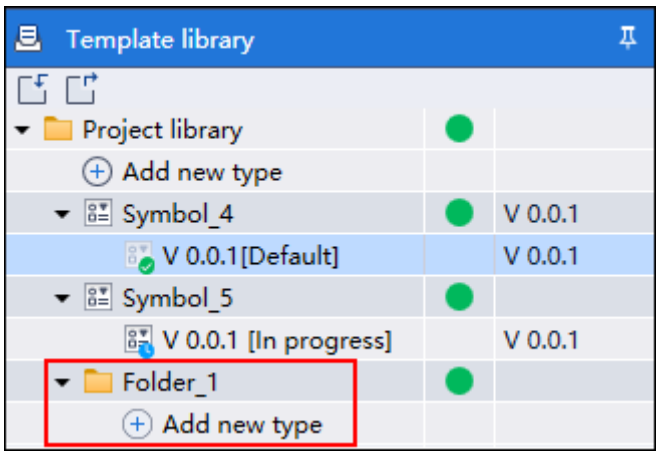
Template libraries support user modifications and editing, and are managed by version through template library versions. Different devices within the same project may use different versions of the template library.

13.4.2 Folder Usage

Folders are used to categorize templates, making them easier to find and manage.

13.4.2.1 Adding a Folder

1. Select the root node "Project library" in the template library.
2. Select "Add new folder" from the right-click menu.



Symbols and user-defined data types can be added to the folder. For detailed instructions, see 13.4.3 Symbol Template Usageand 13.4.4 User-defined Types.

Subfolders can be added within the folder.

13.4.2.2 Deleting a Folder

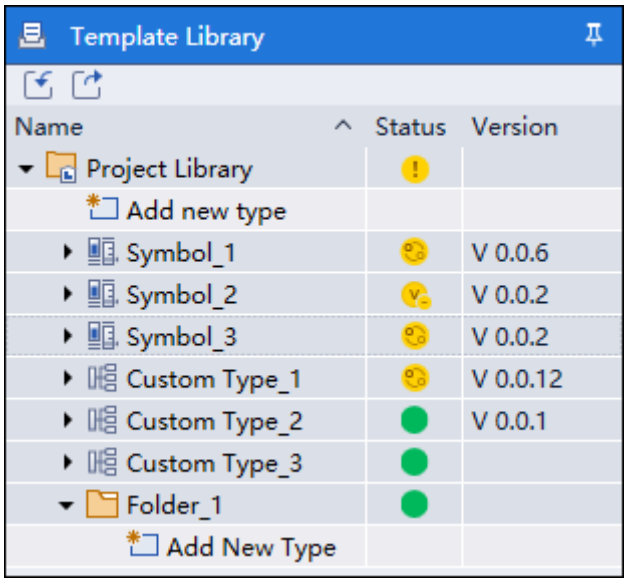
The delete folder operation will clear the folder and all templates within it.

Requirements

- All versions in the folder must be in released state, and the node identifier must be or .
- The data types in the folder cannot be referenced by templates outside the folder.

13.4.2.3 Renaming a Folder

The folder name can be changed, for example, the selected node shown in the figure below.



Naming rules:

- It is composed of letters, numbers, Chinese characters, or spaces. It cannot start or end with a space.
- English is case-insensitive, for example, Aa and AA, aa and aA are considered the same name.
- The maximum name length is 128 characters.
- Unsupported special characters include \ , / , : , * , ?, " , < , > , and |.
- It cannot have the same name as an existing peer folder in the template library.

13.4.3 Symbol Template Usage

The symbols allow you to reuse common HMI screens as templates within the current project. Symbol can be edited, and the modified template can be synchronized to the instance.

13.4.3.1 Overview of Symbol Template

Symbol templates enable efficient and standardized configuration of screens. They allow you to save commonly used configurations, screens, or data objects as reusable standard templates. By using symbol templates, you can ensure consistency of repetitive elements in your project and significantly enhance configuration efficiency.

Symbol templates are stored within a single project and are only available for use within that project.

You can copy, move, and version-control graphic symbol templates using the template library management functions.

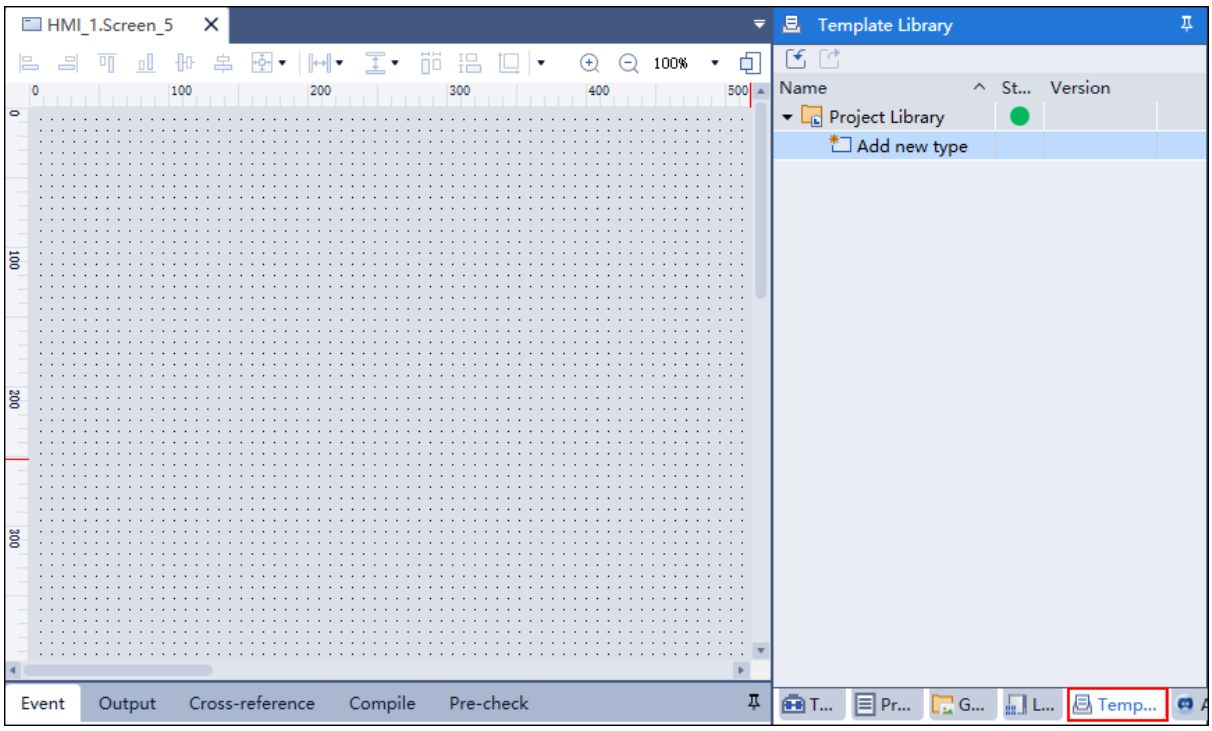
13.4.3.2 Adding a Symbol Template

Introduction

A symbol template is a display and operational object composed of multiple objects. A project can add up to 2000 symbols.

Operation

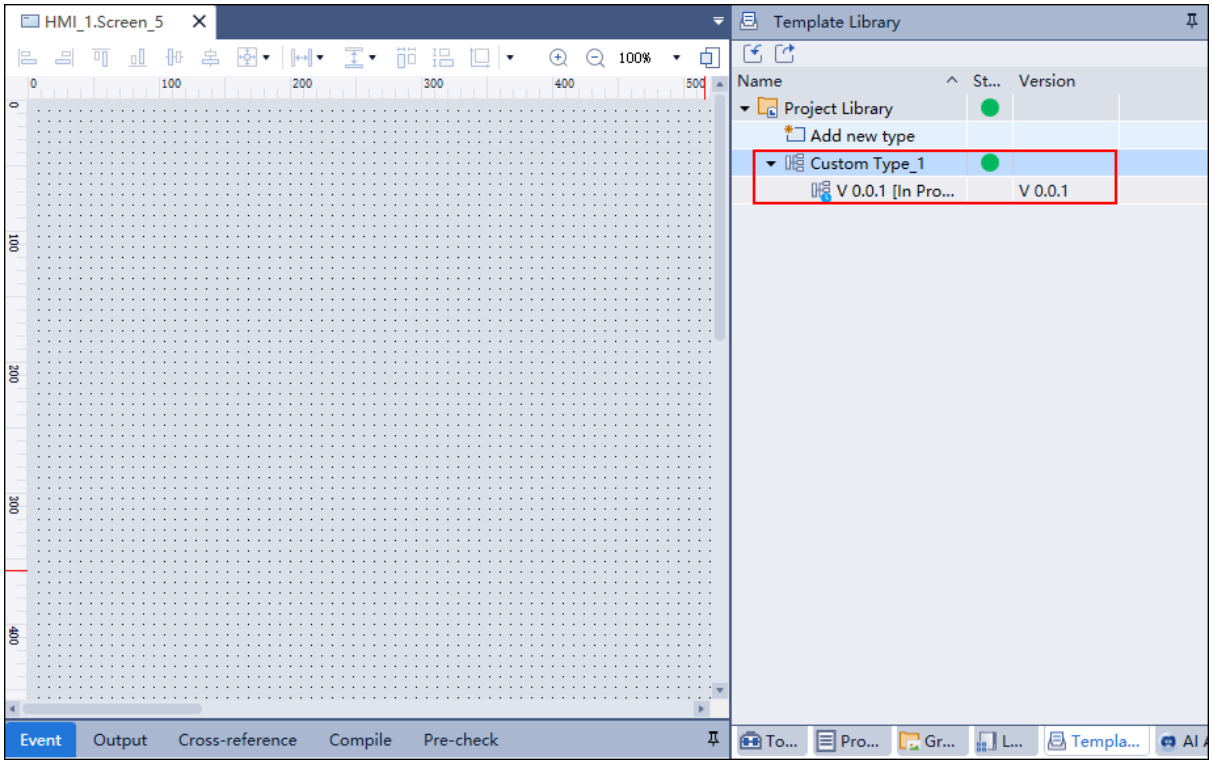
1. Open the default variable table or any user screen
2. Open the "Template library" tab



3. Add a new type

   Double-click the "Add new type" node and enter the symbol name in the pop-up window.

   The new symbol template has been added and is displayed on the project library tree node.



   The version of the new symbol template starts at **0.0.1** and the status is [In Progress].

13.4.3.3 Modifying a Symbol Template

- 13.4.3.3.1 Modifying a Symbol Template
- 13.4.3.3.2 Adding a Variable Interface

- 13.4.3.3.3 Adding a Property Interface
- 13.4.3.3.4 Adding an Event Interface
- 13.4.3.3.5 Adding a Local Variable

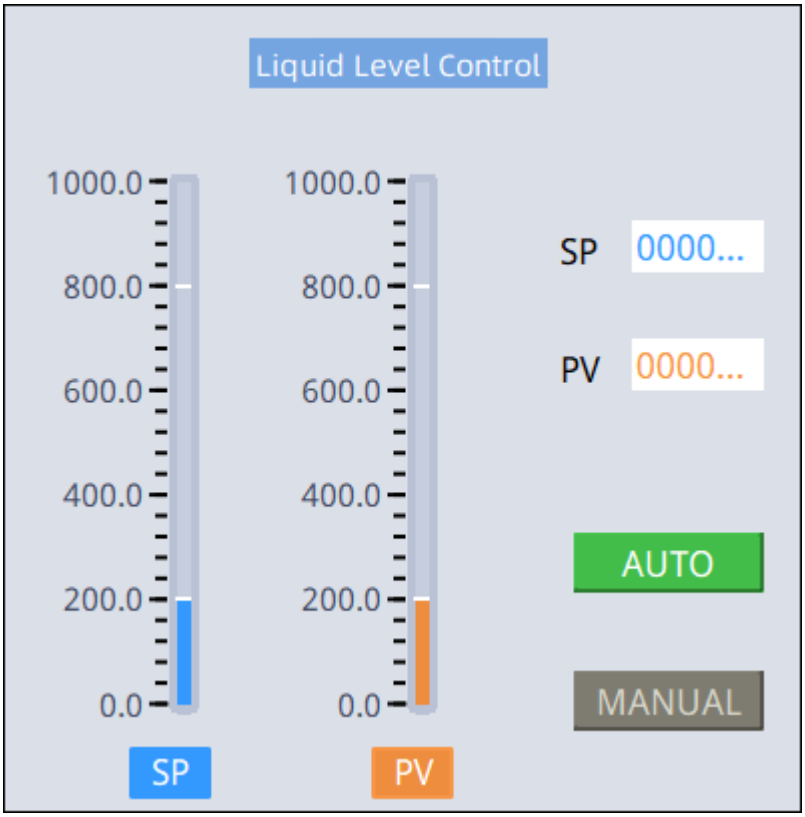
13.4.3.3.1 Modifying a Symbol Template

Requirements

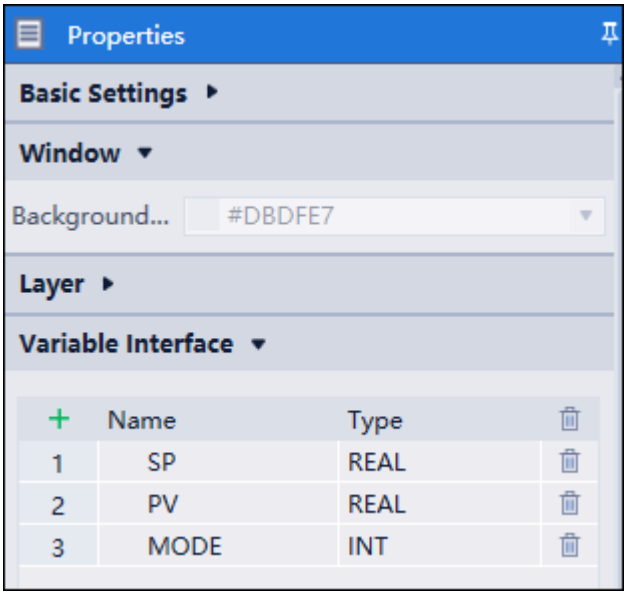
The symbol template has been added to the template library and its status indicator is [In Progress].

Operation

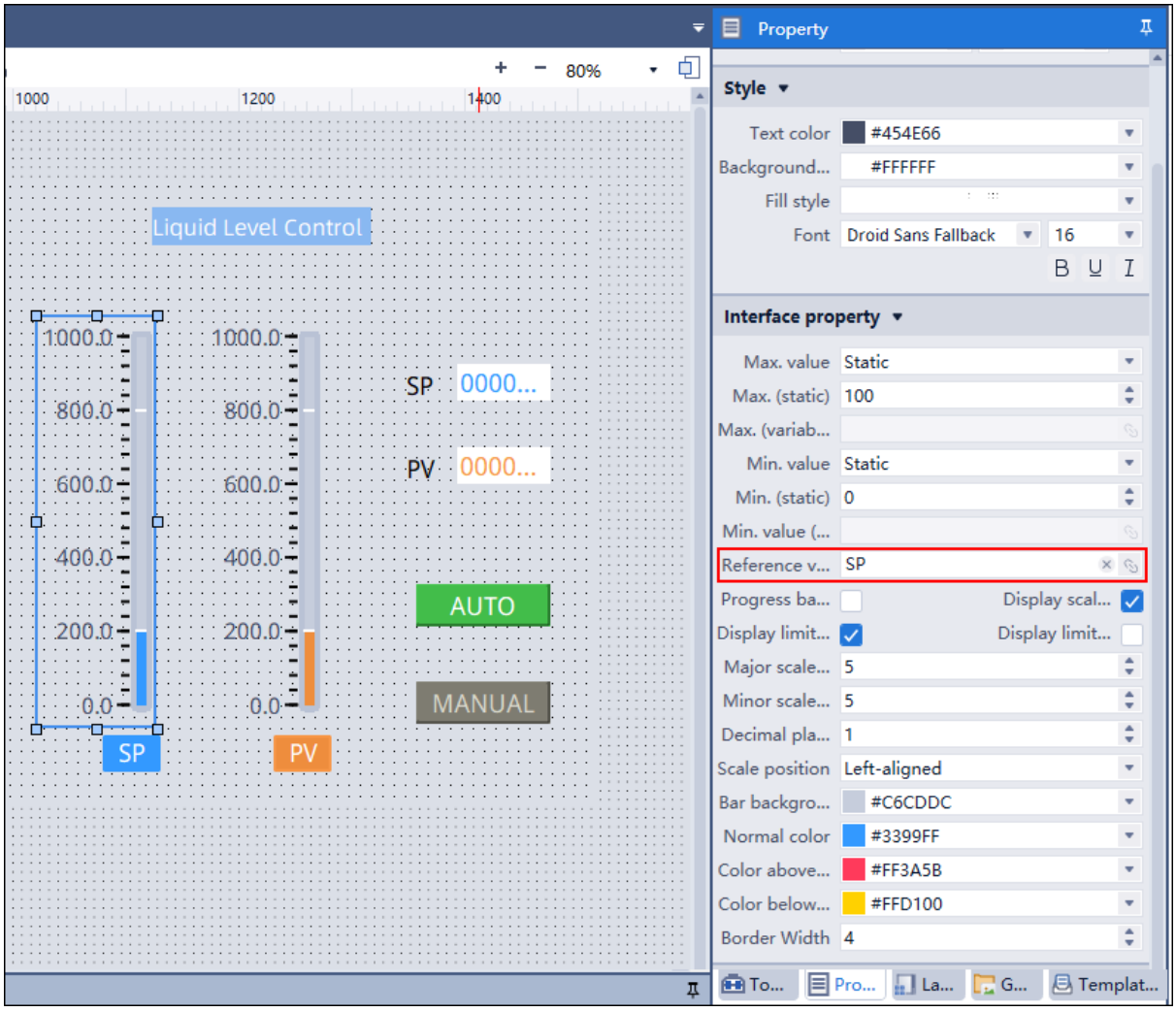
1. Open the symbol editor.
   There are following methods:
   - Select the symbol node, and then choose the "Open" command from the right-click menu.
   - Double-click the symbol version node.
   - Select the symbol version node, and then choose the "Open" command from the right-click menu.
2. Draw objects for display and operation.
   The symbol editor does not support the following six types of controls: Variable Log View, Text List, F(t) Trend, F(x) Trend, Table, Picture-in-Picture.
   For example, you can add bar graph and numeric I/O field in the legend to monitor and operate PV (Process Value) and SP (Setpoint).



3. Add variables to the symbol template
   These variables serve as interfaces for connecting the current symbol to project data. When the symbol is instantiated, they are associated with user-defined variables. Through this association, symbol instances can adjust and respond based on real-time data input from external sources.
   Double-click on the blank area of the symbol screen to open the properties window of symbol. In the "Variable interface" area, click + the button to add a new variable, which will appear in the list.

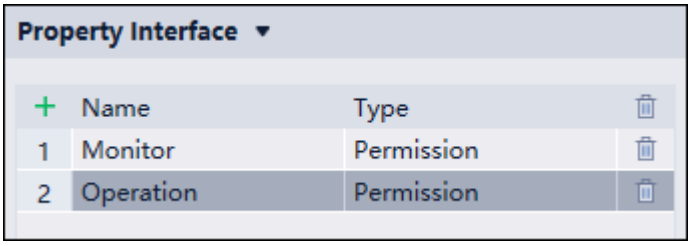


4. Bind the "Variable interface" variables to the respective properties of the objects.

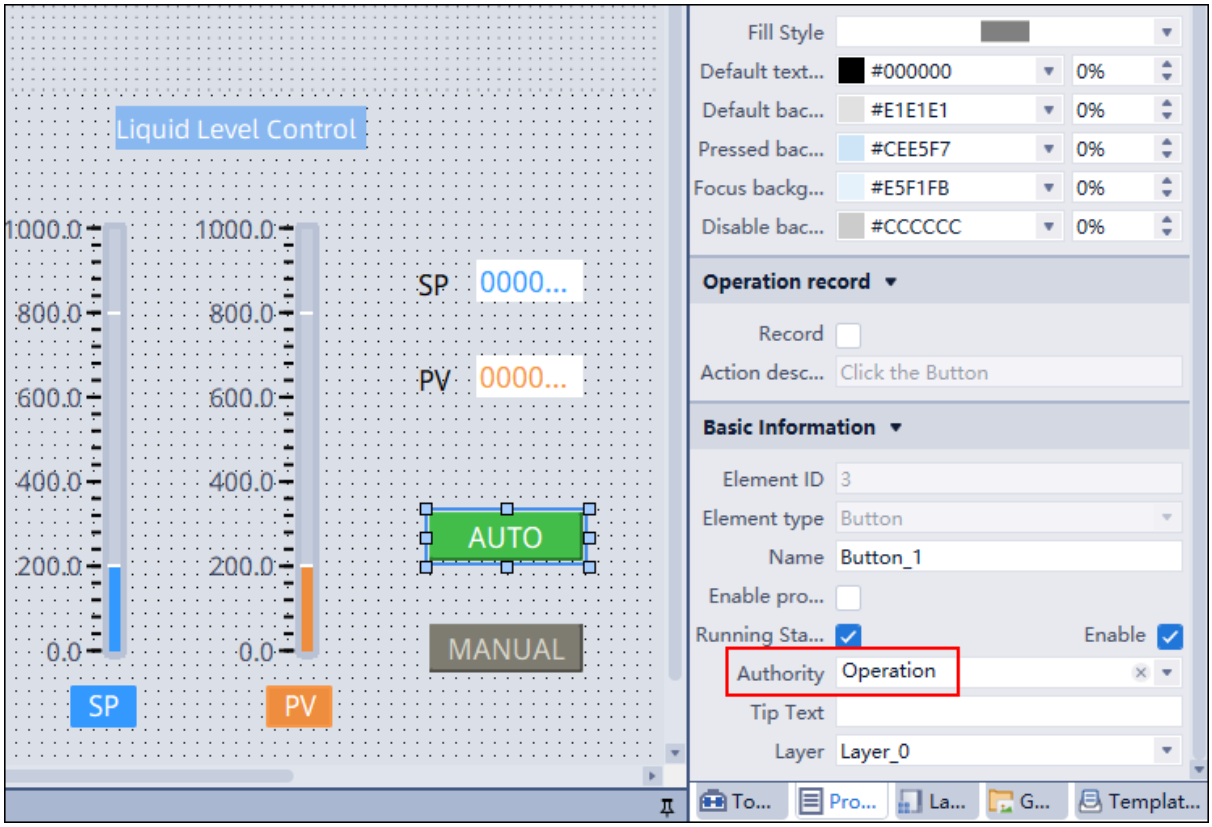


5. Add properties to the symbol template

   In the "Property interface" area, click the button + and specify the property type. The new property will appear in the list.



6. Bind the "Property interface" types to the respective properties of the objects.
   For example, to configure permissions for the operations of two buttons, you need to configure the relevant 13.4.3.3.4 events for the buttons.



7. Release the symbol template.
   After completing the configuration of the symbol template, release it in the template library so that it can be instantiated on the screen.

13.4.3.3.2 Adding a Variable Interface

Create interface variables in the symbol template and bind them to objects within the symbol template. When the template is instantiated, map project variables to interface variables to pass parameter values from project variables to objects.

A symbol template can support up to 20,000 interface variables.

Supported data types for interface variables: basic data types, arrays, and structures. Structures are limited to released versions of 13.4.4 User-defined Types.

Requirements

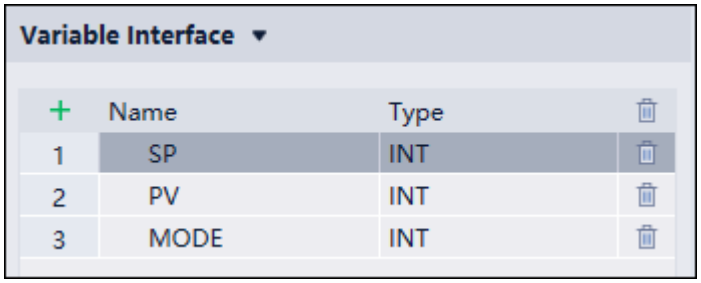
- The symbol template has been added and is in edit mode.
- Restrictions for variables referencing user-defined (struct) types:
  - Do not mix multiple versions of the same type
    In a symbol template, only one version of a type can be used.

For example, if variable 1 uses version V 0.0.2 of DatatypeA, and variable 2 uses version V 0.0.5 of DatatypeA, this is an incorrect operation.

- Do not mix types with nested dependencies
  . If a user-defined type A has a member that references another user-defined type B, both types cannot be used directly as variable types in the same symbol template.
  For example, if a member of type DatatypeA is of type DatatypeB, and variable 1 and variable 2 are declared as these two types respectively, this is an incorrect operation.

Operation

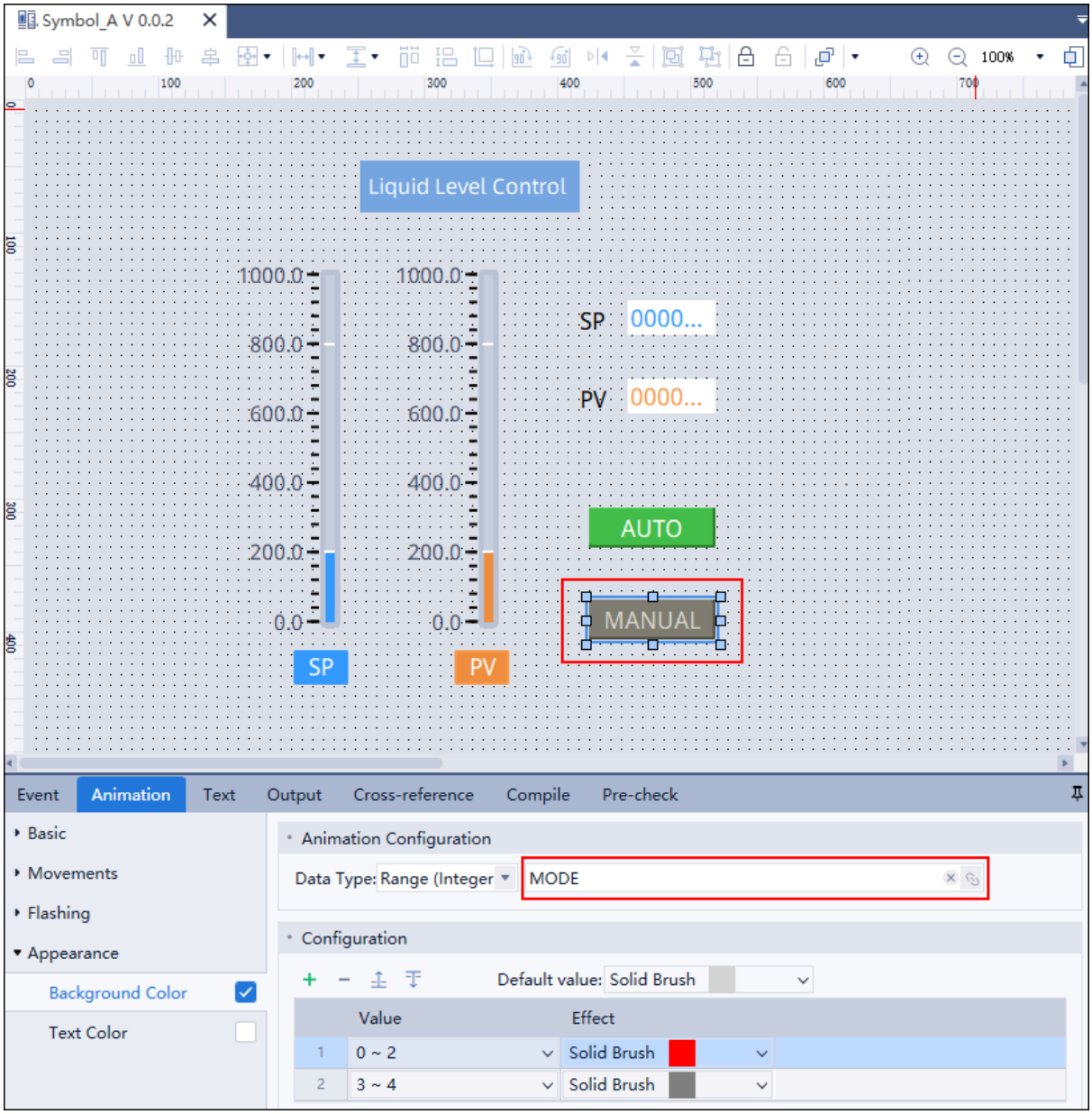
1. Double-click on the blank area of the symbol template to open the properties window of symbol template.
2. Add variables in the "Variable interface" area
   The variable data type can be a basic data type or a user-defined type released in the template library.



NOTE

If the variable type uses a user-defined type from the template library, there is no limit on the number of types, but only one version of each type can be used. Multiple versions cannot be used simultaneously.

3. Double-click on the target object in the symbol template to open the monitoring window (or properties window). For example, the target object is a button.
4. In the button monitoring window, under "Animation > Appearance > Background color," bind the variable added in step 2 and set the color corresponding to the variable value. Then, achieve the function where the button color changes according to the variable value.



5. Repeat the above steps to complete the association of each object with the "Variable interface".

13.4.3.3.3 Adding a Property Interface

Create property interfaces in the symbol template and bind them to objects within the symbol template. When the template is instantiated, map project properties to interface properties to enable customizing object styles within the project.

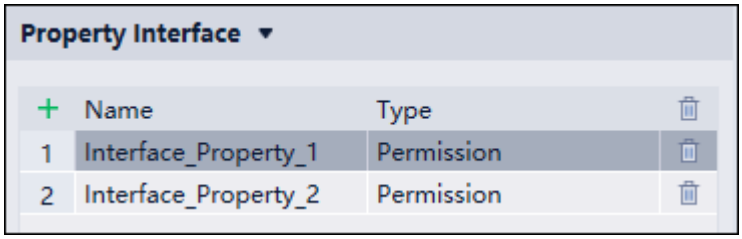
A symbol template can support up to 128 property interfaces.

Requirements

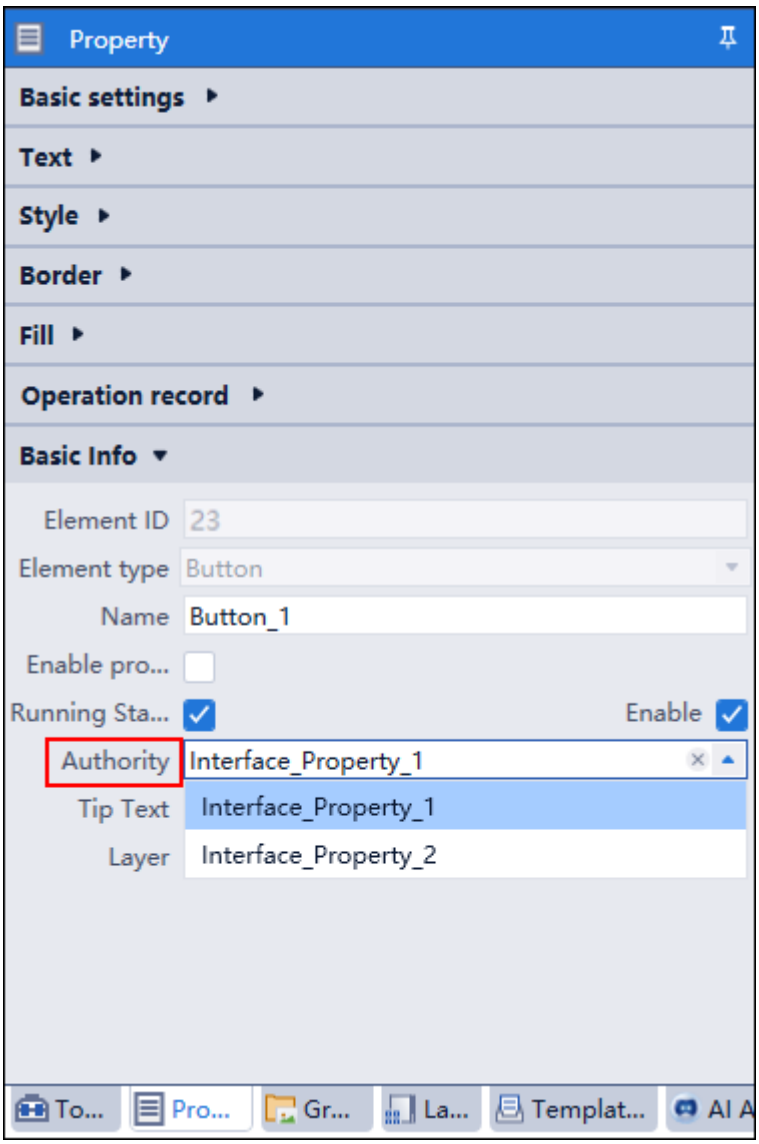
- A graphic symbol template has been added and has a version that is in progress.
- Property interface supports data types: permissions and text list.

Procedure

1. Open the editing interface for the version that is in progress.
2. Double-click the blank area of the workspace to open the property box for the current version.
3. Add properties in the "Property interface" area and assign property types. For example, set the type of **Interface_Property_1** to **Permission**.



4. Bind the property of a target object (e.g., a button) in the template to the interface property.
   For example, bind the "Basic info > Authority" of a button to **Interface_Property_1**.



Property Interface Type	Applicable Object	Associated Property
Authoritiess	All graphic objects	Basic information > Authority
Text list	Text list	Options > Text list

5. Repeat the above steps to bind all object properties to the corresponding "Property interface".
6. Publish the symbol template.

13.4.3.3.4 Adding an Event Interface

Create interface events in the symbol template and bind them to objects within the symbol template. When the template is instantiated, customize object styles in the project through interface properties.

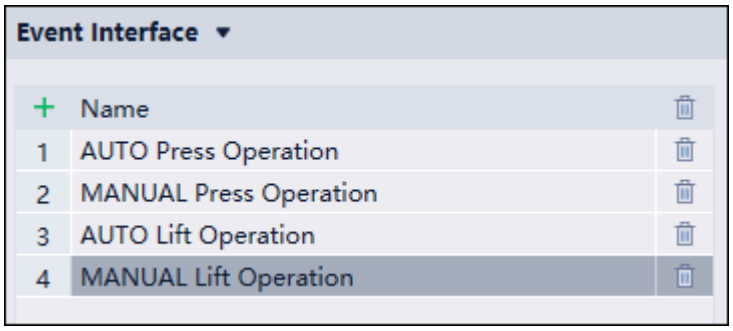
A symbol template supports up to 16 interface events.

Requirements

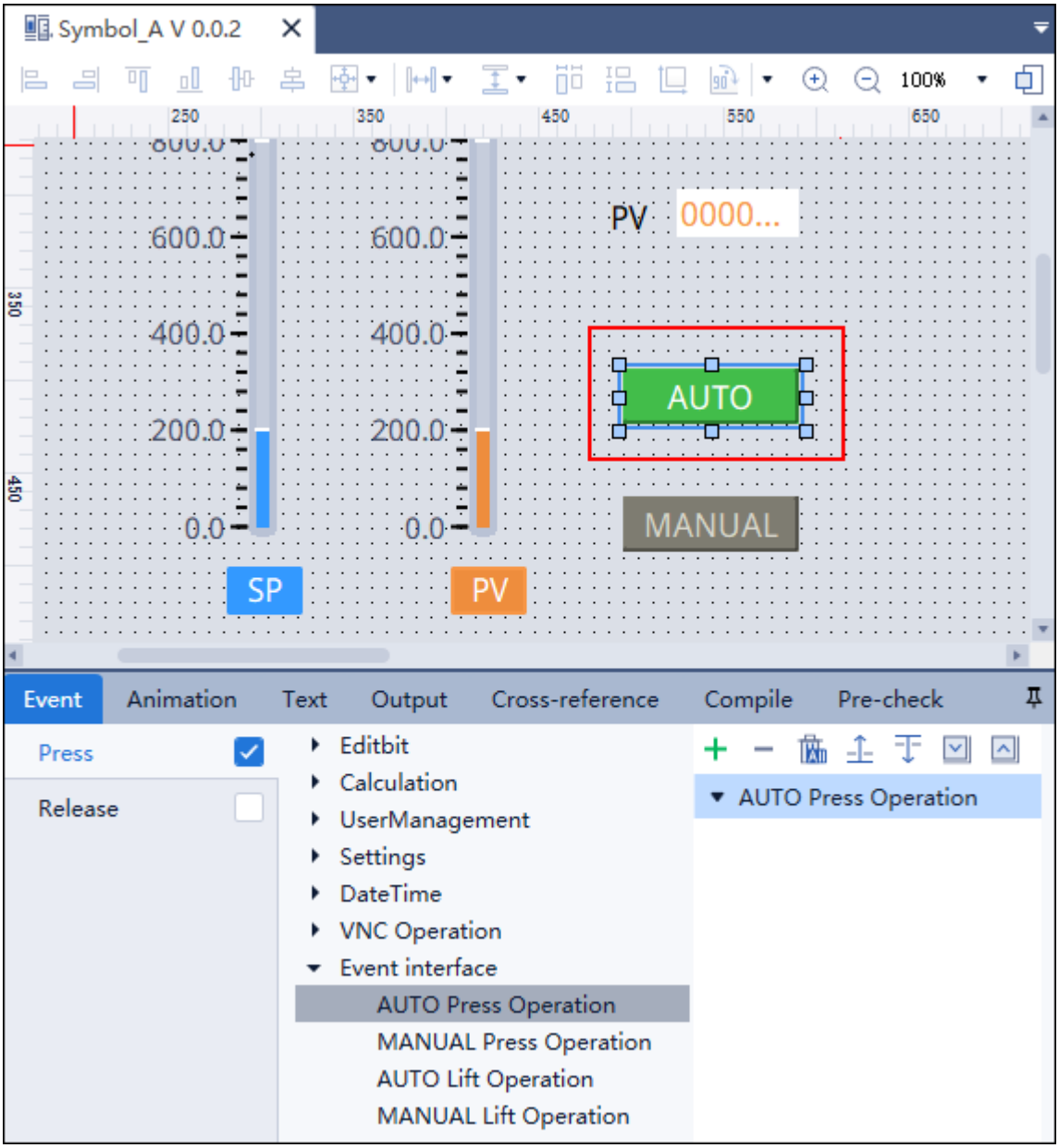
The symbol template has been added and is in edit mode.

Operation

1. Double-click on the blank area of the symbol template to open the properties window of symbol template.
2. Add events in the "Event interface" area and specify the purpose of each event. For example, **AUTO press operation**.



3. Double-click the target object in the symbol template to open the "Event" tab in the monitoring area. For example, the target object is a button.
4. Bind the object to an event. Click "Event > Press" to bind **the AUTO press operation** in step 2.



5. Repeat the above steps to complete the binding of events for all objects with the "Event interface".

13.4.3.3.5 Adding a Local Variable

Local variables are variables used for internal logic processing within a symbol template and do not require association with user variables when the symbol is instantiated.

A symbol template supports up to 128 local variables.

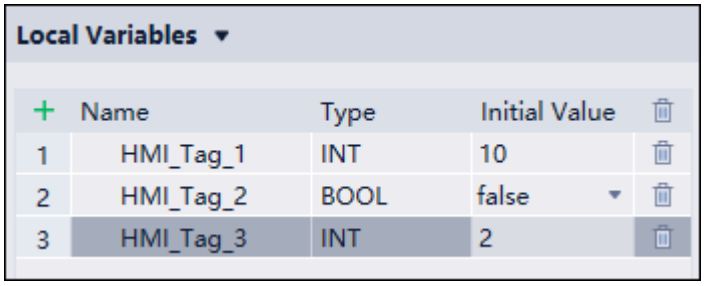
Supported data types for local variables: basic data types and arrays.

Requirements

The symbol template has been added and is in edit mode.

Operation

1. Double-click on the blank area of the symbol template to open the properties window of symbol template.
2. Add variables in the "Local variable" area



The variable name must not be the same as the names in the variable interface and property interface.

13.4.3.3.6 Adding an Animation to Object

Objects in symbol templates support animation configuration, with various dynamic effects such as movement, rotation, and flashing.

Requirements

Identifies the symbol version node and ensure that the object has been added.

Operation

The operation to configure animation for objects in symbol templates is exactly the same as for objects in user interfaces. For details, see 13.3.8 Animation Configuration. The difference lies in that variables associated with the former can only be variable interfaces or local variables, while variables associated with the latter are user variables.

13.4.3.3.7 Adding an Event to Object

Objects in symbol templates support event configuration.

Requirements

Identifies the symbol version node and ensure that the object has been added.

Operation

The operation to configure events for objects in symbol templates is exactly the same as the operation for objects in user interfaces. For details, see 13.3.7.14 Adding an Event. The difference is that the former can configure system functions and event interfaces, while the latter can only configure system functions.

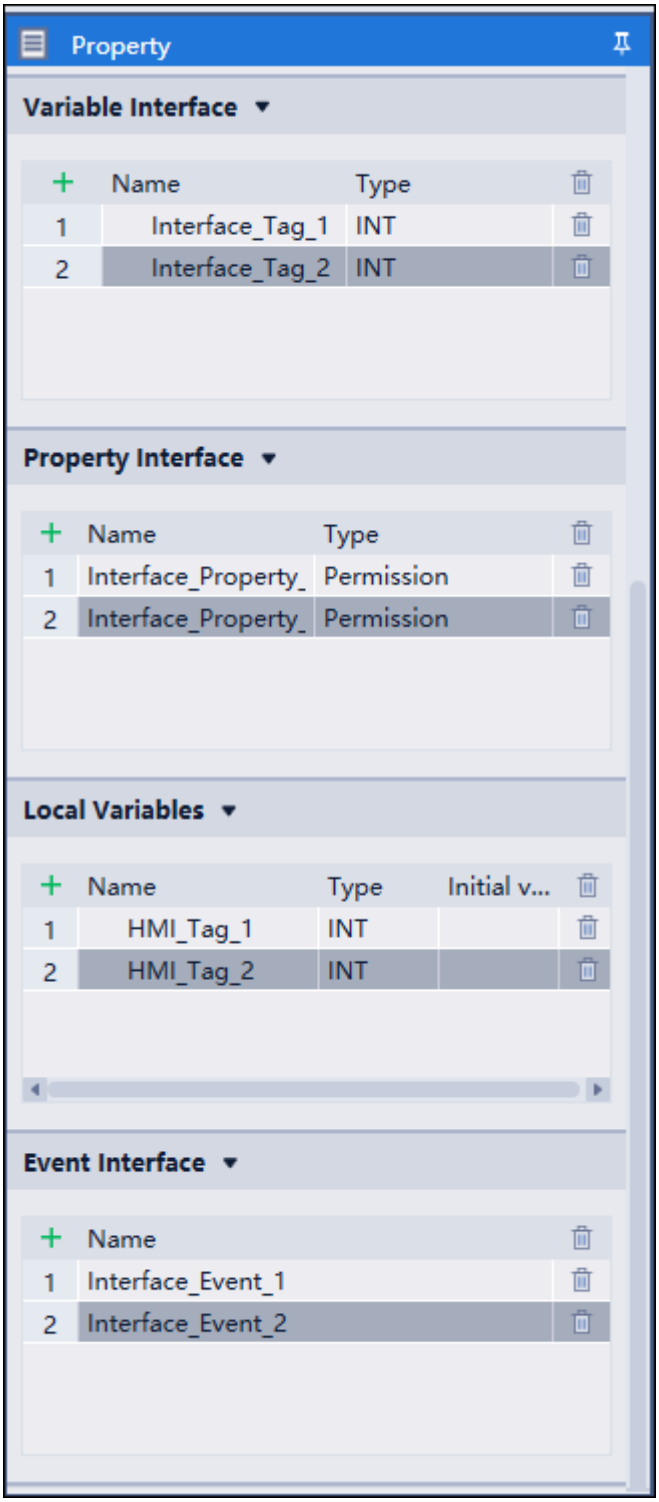
13.4.3.4 Applying a Symbol Template

After the symbol template is published, it can be applied to user screens for instantiation.

User screen refer to the screen type under the "Screen" node in the project tree, such as picture-in-picture screens or pop-up screens.

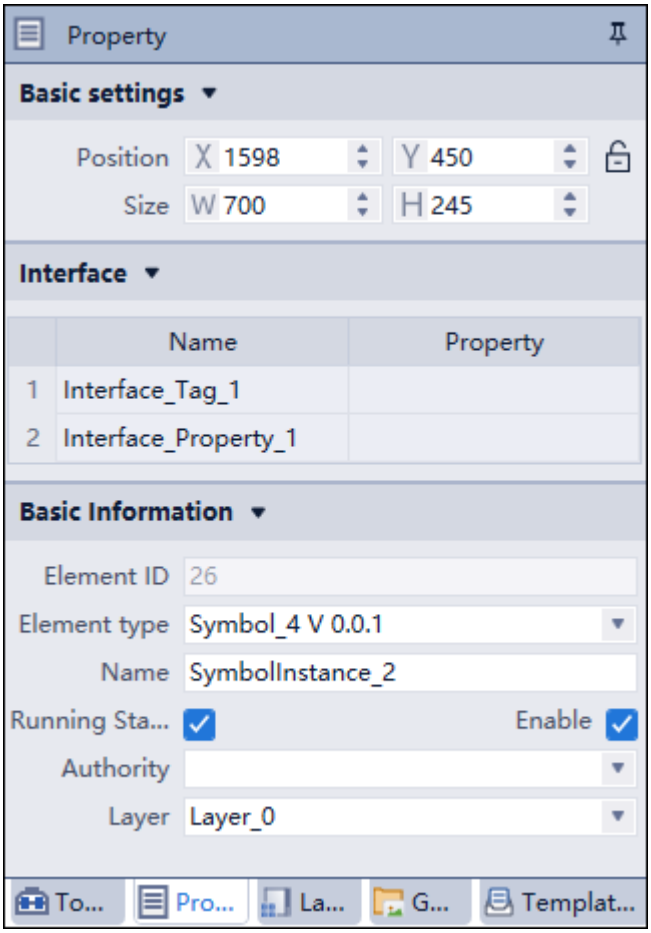
Operation

1. Select the symbol template
   In the template library, select the published version of the desired symbol. The version node is identified , for example, version V0.0.3 of "Symbol_1"
   Open this version to review the interface configuration and ensure full understanding of subsequent steps.



2. Drag the symbol template onto the application screen
   Drag version V 0.0.3 to the template application screen, such as "Screen_1". Instantiate the symbol content.

3. Open the properties tab
   Double-click the symbol template in "Screen_1" to open the "Property" tab.

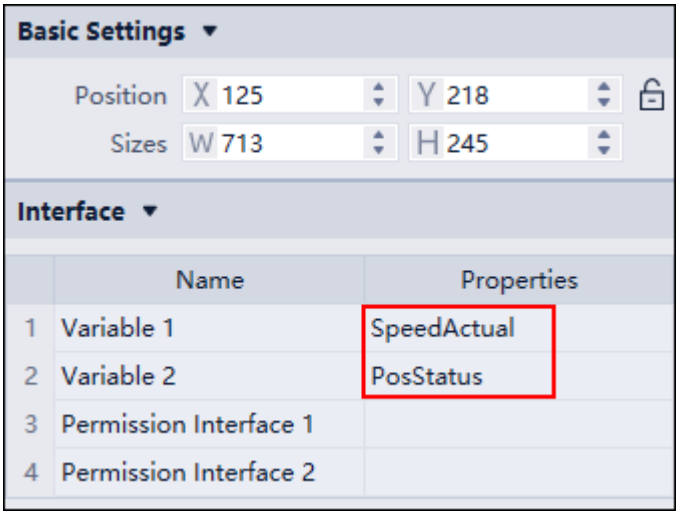


4. Configure variable interface

   When associating the template's variable interface with the associated variable in the project, the data types must be consistent.

   If the variable interface data type is a user-defined structure from PLCs, the associated variable must be an HMI external variable.

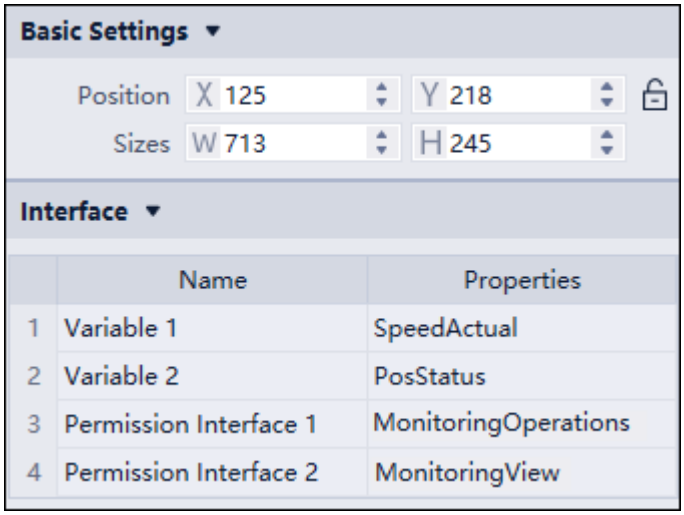
   Additionally, this external variable must be associated with an instance variable of the same structure in the PLCs.



5. Configure permission interface

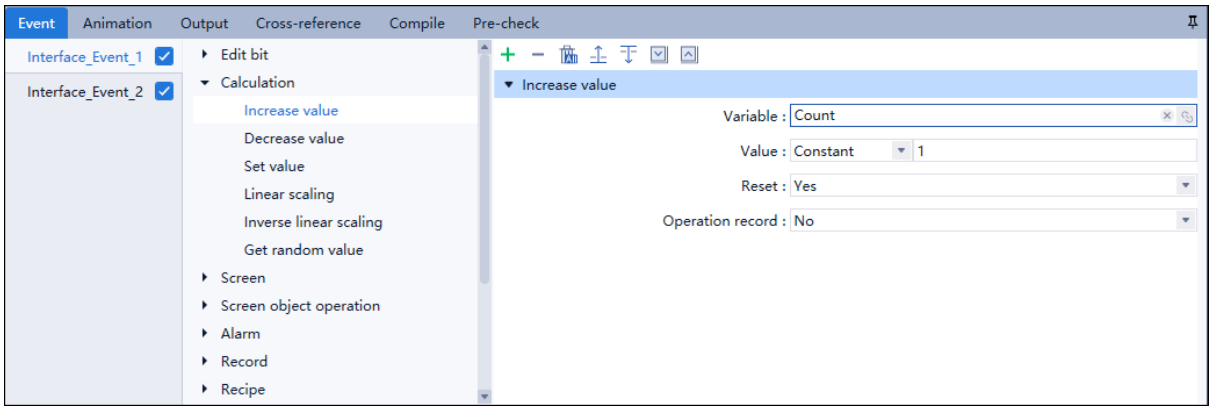
   a. Add user permissions under the "9.8 User and role" node, and specify the target device in the "Device name".

   b. In the properties window, associate the permission interface with the user permissions.



6. Configure event interface

   Double-click the symbol instance in "Screen_1" to open the "Event" tab, and associate the event interface with user events.



7. Save the project

   After completing all configurations, save the project to ensure all changes are applied.

13.4.4 User-Defined Types

A user-defined type is a generic, reusable data type created by the user, similar to a structure type, and contained within a library file, which can be reused across devices.

13.4.4.1 Adding a User-Defined Type

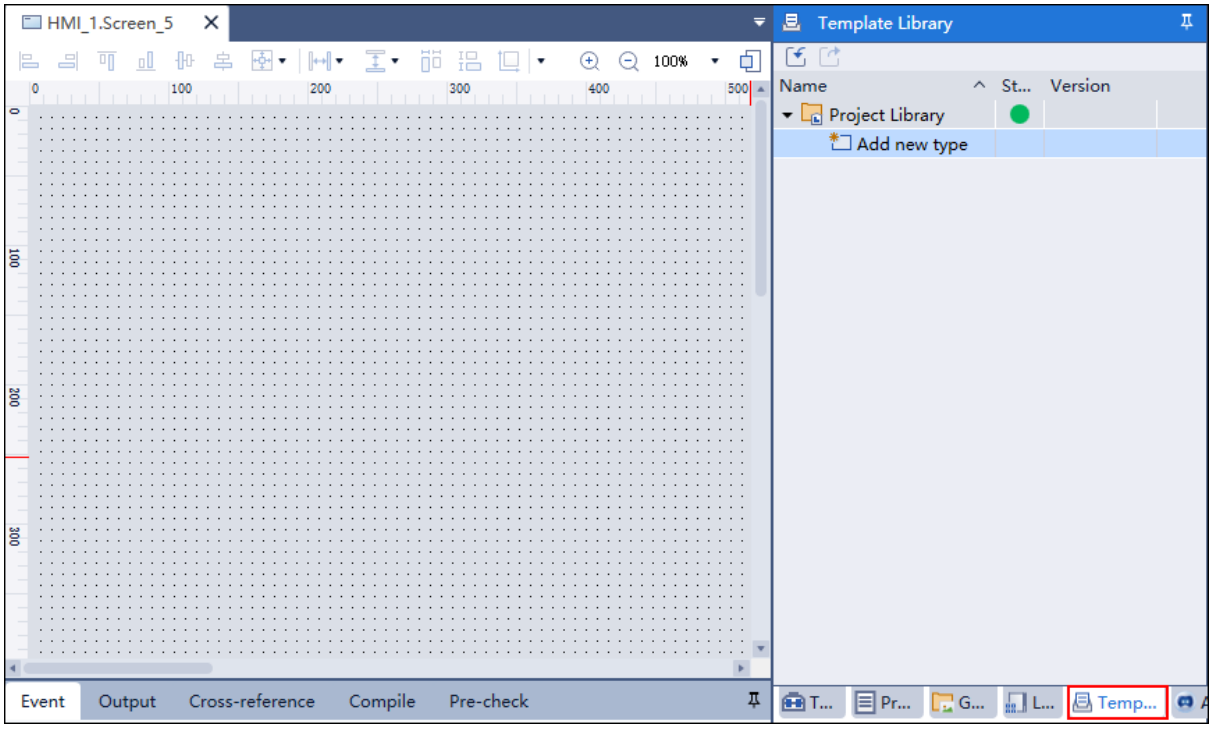
The template library supports two methods for defining struct types, facilitating reuse and management.

HMI user-defined type

You can manually create a structure type within the template library and directly define the structure name and member types.

1. Opening a Project
2. Open the "Template library" tab

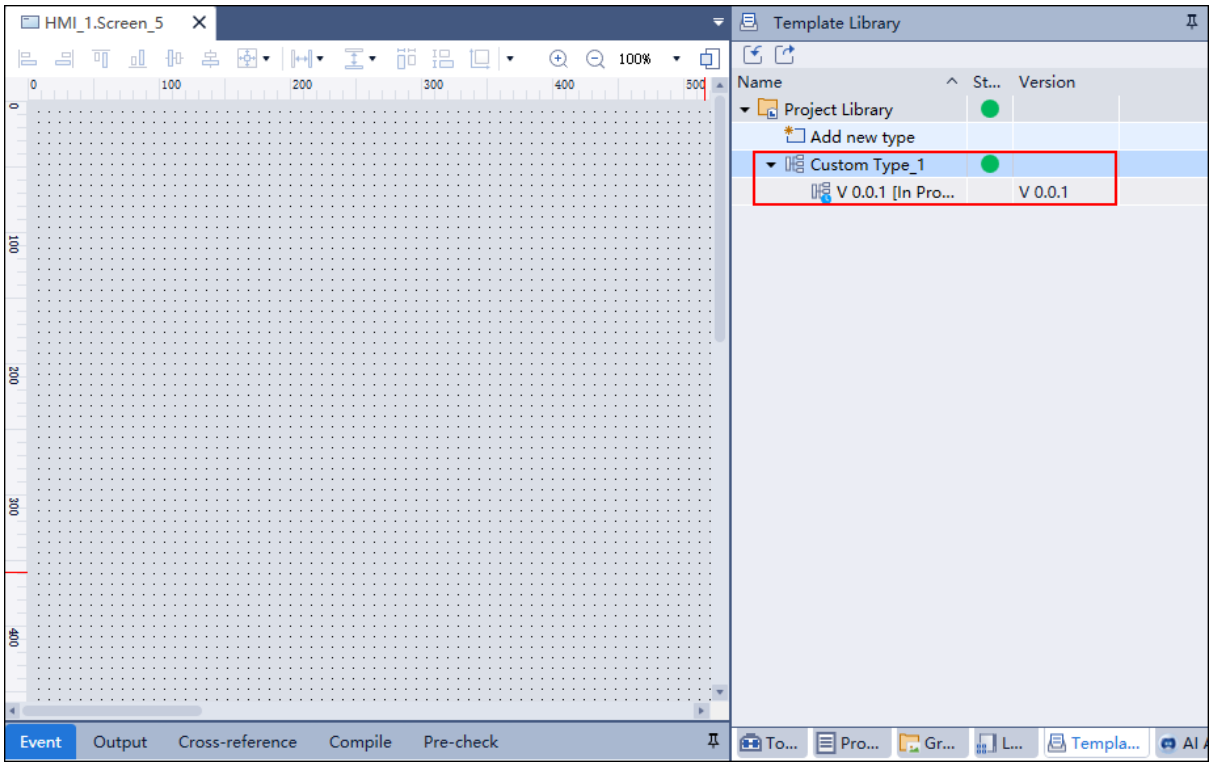
   After opening the user screen, switch to the "Template library" tab. Even if no HMI device has been added to the project, this switch can still be performed in the right area of the interface.



3. Add a new type

   Double-click the "Add new type" node, select "Custom type" in the pop-up window, and enter the data type name.

   The user-defined type has been added and is displayed on the tree node in the project library.



   The version of the user-defined type starts at **0.0.1** and the status is [In Progress].

PLC user-defined type

Drag and drop the user-defined structure created in the PLCs into the template library to template the PLC user-defined structure.

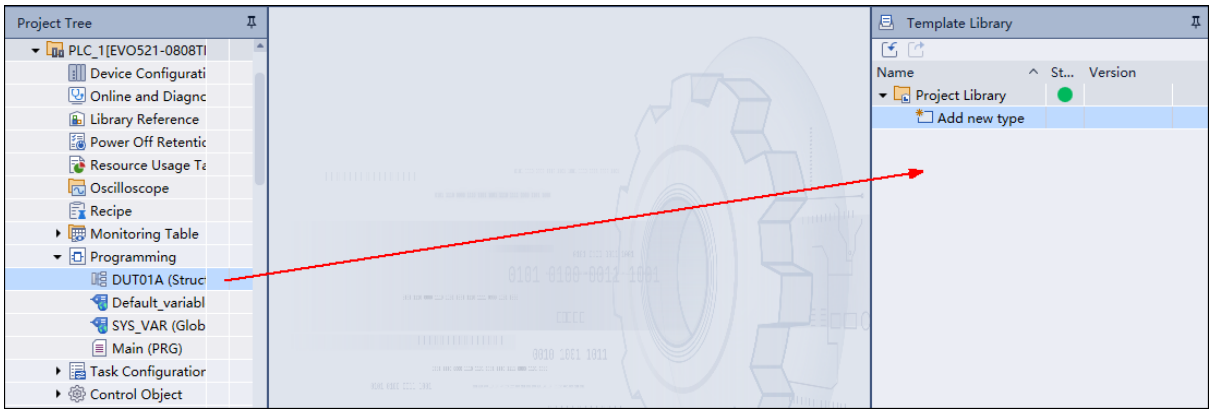
Prerequisites

The user-defined structure types that can be dragged must simultaneously meet the following conditions:

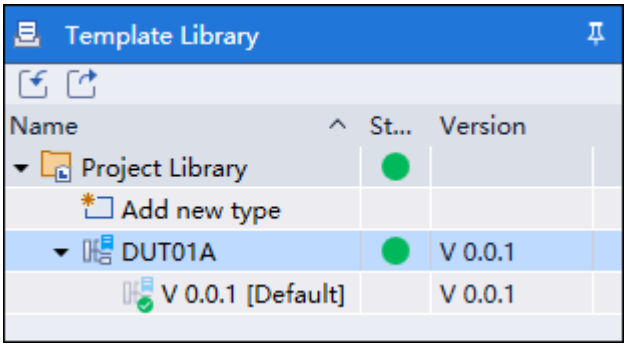
- User-defined data types have been created or imported in the current project.
- Only user-defined 10.1.1.8.1 Standard Data Types and array-type structures are supported for templating. Arrays support multi-dimensional arrays and array nesting, with the number of array elements not exceeding 2000. Standard data types with specified upper/lower limits and arrays with non-numeric boundaries are not supported.
- Enumerations, unions, aliases, FBs, FCs, PRGs, and library types are not supported.
- Structure inheritance is not supported.

Procedure

1. In the PLC device tree, select the structure to reference.
2. Drag the selected structure into the template library.



3. The structure node will automatically appear in the template library and be defined as version V 0.0.1.



Both the node in the PLC device tree and the node in the template library are identified .
During the templating process, if the referenced object contains nested structures and these sub-structures are also of user-defined types, all nested sub-structures will be automatically templated.

> **NOTE**
>
> Once a structure type has been successfully templated, it cannot be templated again if no modifications have been made to the structure members (except for changes to the type name).

13.4.4.2 Modifying a User-Defined Type

After successfully adding a structure type to the template library, the next step is to configure member variables and types.

HMI user-defined type

User-defined type identifier  The user-defined type editor can be opened via the "Open" command in the right-click menu:

1. Select the first row and click the button  to add a new row.
   Add a new data row at the bottom of the table. The default member variable name is number_x where x starts at 1 and increments. The name must be unique and will be renamed automatically if duplicate. The default data type is INT.
2. User-defined data types support basic properties such as name, description, and data type.
3. Creation by entering a name is supported.
4. The data type of members supports basic types, arrays, and structures (excluding user-defined structures referenced from PLCs).

User-defined type members support insertion, copying, pasting, moving up, moving down, deletion, and nesting:

- The rules for inserting, copying, pasting, moving up, moving down, and deleting user-defined type members are the same as those for variables.
- User-defined data types support up to 8 levels of nesting.
- Recursive use is prohibited. User-defined data types that are recursive cannot be used as members of other user-defined types. You will be prompted if recursive use is detected.

PLC user-defined type

For these structures in the template library, except for the struct name and comment, other information cannot be edited and can only be viewed.

13.4.4.3 Impact of User-Defined Type Version Upgrade

To ensure consistency across the template library, upgrading the version of a user-defined type may trigger cascading effects once it is referenced by a symbol template. The specific rules and impacts are as follows.

Version upgrade rules for symbol templates and user-defined (structure) types

The core rule depends on the current state of the symbol template:

- Scenario 1: Linked upgrade
  Condition: Both the symbol template and the referenced user-defined (structure) type are in the "Released" state.
  Result: Once a new version of the user-defined type is released, all released symbol templates referencing it will automatically create a new version (entering the "Editing" state) and update the reference to the latest type version.
- Scenario 2: Independent upgrade
  Condition: The symbol template itself is in the "Editing" state.
  Result: Regardless of how the referenced user-defined (structure) type is upgraded, the symbol template remains unaffected and continues to use the original type version.

Prerequisites

A symbol template and a user-defined (structure) type have already been created.

A relationship has been established: The variable type of the symbol template references a specific version of the user-defined (structure) type.

Procedure

When upgrading the version of the user-defined type, the scope of impact is determined jointly by its internal structure and reference relationships. The key point lies in determining whether the related user-defined type contains nested structures. You must understand this before upgrading.

- Upgrade a structure type without nesting

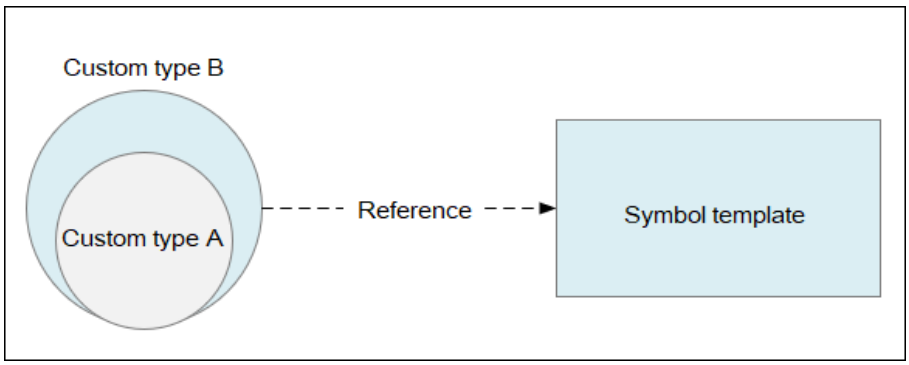
To better illustrate the above upgrade rules, the following table shows typical version evolution scenarios (including linked and independent upgrades) after a reference relationship is established between a user-defined type (e.g., object Datatype) and a symbol template (e.g., object Symbol).

Step	Operation	User-defined type	Symbol Template	Result
1	A reference relationship is established between version V0.0.1 of Datatype and version V0.0.1 of Symbol.	V 0.0.1 released	V 0.0.1 released	The reference relationship is locked
2	The Datatype is upgraded from version V 0.0.1 to V 0.0.2. The Symbol is upgraded to V 0.0.2 automatically.	V 0.0.2 released	V 0.0.2 in progress	The referenced version in the Symbol is automatically upgraded to V 0.0.2.
3	The Datatype is upgraded from version V 0.0.2 to V 0.0.3.	V 0.0.3 released	V 0.0.2 in progress	The version V 0.0.3 of Datatype is not referenced by the Symbol.
4	The version V 0.0.2 of Symbol has been successfully released.	V 0.0.3 released	V 0.0.2 released	The reference relationship between V 0.0.2 of Datatype and V 0.0.2 of Symbol is locked.
5	The Datatype is upgraded from version V 0.0.3 to V 0.0.4. The Symbol is upgraded to V 0.0.3 automatically.	V 0.0.4 released	V 0.0.3 in progress	The referenced version in the Symbol is automatically upgraded to V 0.0.4.

- Upgrade a structure type with nesting

This specifically refers to PLC user-defined types. When a symbol template uses nested PLC user-defined types, upgrading the type affects its entire reference chain.

The reference relationship is shown in the diagram below. The PLC user-defined type B nests type A, and the data type of symbol template 1 is associated (instantiated) with type B.



When type B undergoes a version upgrade, it triggers a linked upgrade of the symbol template and type A. Even if type A itself is not modified, it will be forcibly upgraded to a new version.

The prerequisite for upgrading a symbol template is that it must be in a released state.

Recommendations

- Plan upgrades: Before editing a "Released" user-defined (structure) type, check whether any symbol templates depend on it to avoid unintended chain upgrades. If an unnecessary symbol template upgrade is triggered, use the "Discard Changes" command to undo it.
- State management: Release symbol templates in "In progress" state promptly to maintain synchronization of version dependencies within the project.

13.4.4.4 Applying a User-defined Type

There are two locations in the template library where user-defined types can be applied:

User-defined type	Use Position
HMI user-defined type	Apply within the HMI variable table. Apply within the "Variable interface" property of a symbol template. For details, see 13.4.3.4 Applying a Symbol Template.
PLC user-defined type	Apply within other PLCs. Apply within the "Variable interface" property of a symbol template. For details, see 13.4.3.4 Applying a Symbol Template.

This section explains how to apply PLC user-defined type within other PLCs.

Procedure

1. Template user-defined structure in PLCs
   For example, when a user-defined structure A from PLC_1 device is dragged into the template library, a node with the same name as structure A is created in the template library, and an initial version (V 0.0.1) is generated.
2. Apply in other PLCs
   Select the structure A node in the template library and drag it into the "Programming" node of PLC_2 device. The default version of structure A will be added to PLC_2.
   If multiple versions of structure A exist, you can select a specific version to drag, and that specific version will be added to PLC_2.

Result

The user-defined structure A has been successfully added to PLC_2, with the corresponding node identifier .

Precautions

- The version of the structure dragged into PLC_2 strictly corresponds to the version in the template library. In the template library, all versions are in a released state (non-editable). Therefore, when a new version is released in the template library, manually synchronize that new version to the PLC_2 device if needed.
- Structure definitions applied in PLC_1 and PLC_2 support editing. After editing, the changes can be updated to the template library using the 13.4.6.11 Synchronizaing PLC Types command.

13.4.5 Template Status Management

Templates and versions in the template library use different symbols to indicate their reference state and release state, helping you quickly identify their current state.

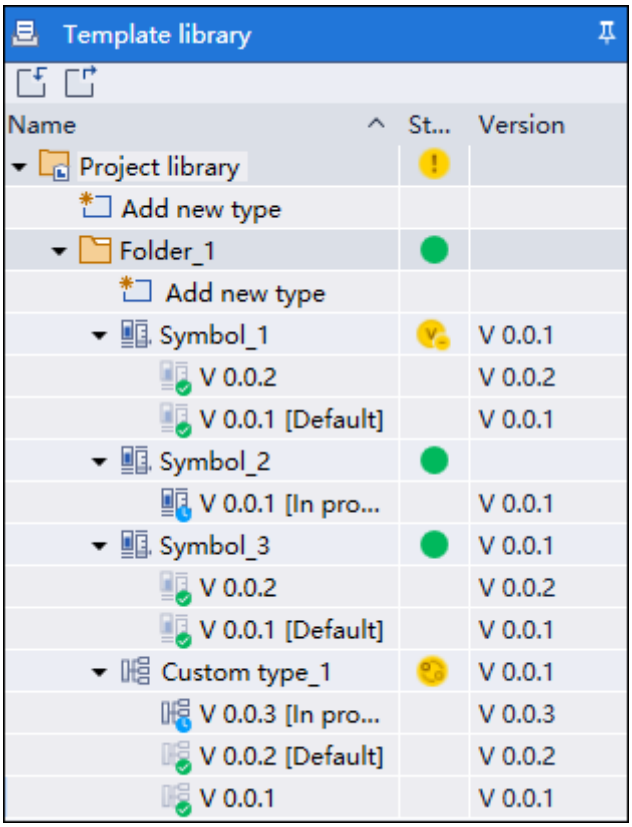
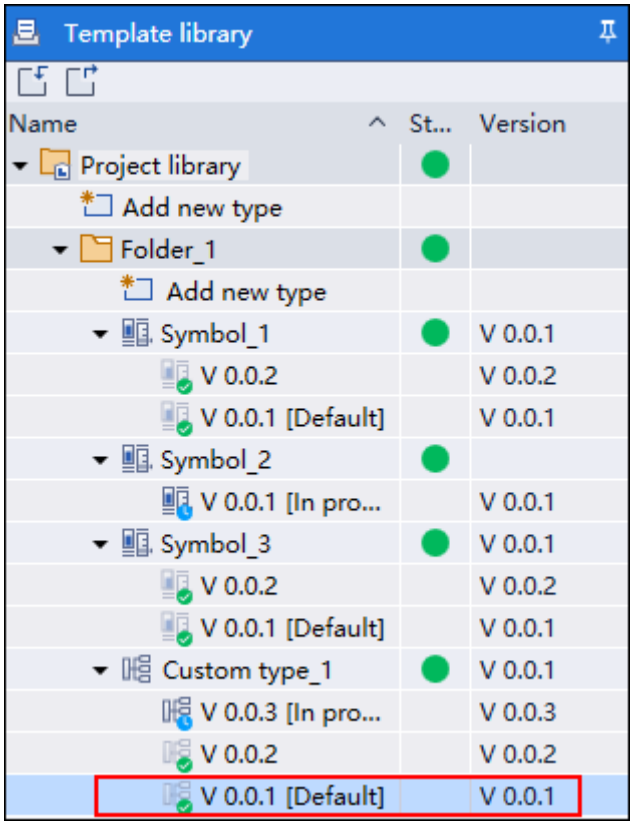


Table13-4-5-1 Description

Category	Label	Description
Project/Template node		The default version pf current template references other templates, but the referenced versions are not the default versions of those templates.
		Does not reference other templates, or References the default versions of other templates
		Non-default version is referenced or instantiated
		Multiple inconsistencies

Category	Label	Description
Version node		Version content is being edited and not yet released. Version number ends with [In Progress].
		Version content has been released. Version number ends without any label or with [Default].

Since the default version can be specified using the command "Set as default", the state of the template node changes along with the default version settings, even if the content remains unchanged. For example, in the figure below, when the default version of "Custom type_1" is changed from V 0.0.2 to V0.0.1, the state of the template node changes.

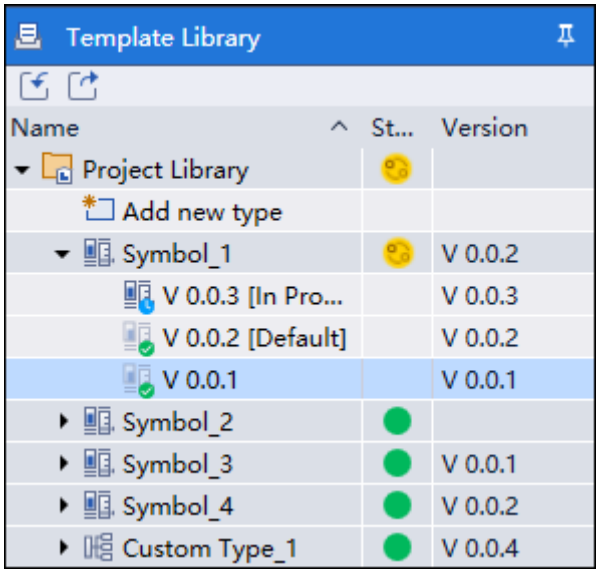


13.4.6 Common Operations on Template

Templates refer generally to symbols and user-defined types in the template library. The operation commands and their results mentioned in the following sections apply to these symbols and user-defined types.

13.4.6.1 Opening a Template

When different nodes are selected in the template, using the "Open" command from the right-click menu produces different results:



Select symbol node

Executing "Open" on a symbol node automatically opens the content of the [Default] version. If there is no default version, the [In Progress] version will be opened. For example, selecting the "Symbol_1" node will open the content of version V 0.0.2 when using the "Open" command.

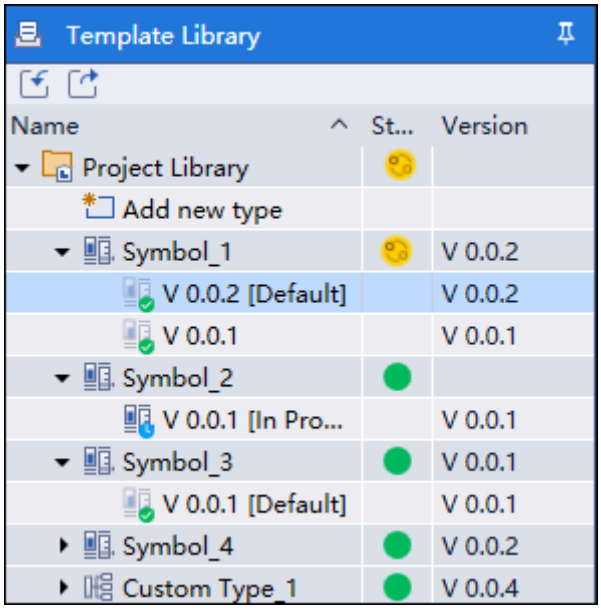
Select symbol version

- No state indicator: For example, for the "V 0.0.1" version of "Symbol_1", the "Open" command will open its content, allowing only viewing, not editing.
- [Default] indicator: For example, for the "V 0.0.2" version of "Symbol_1", the "Open" command will open its content, allowing only viewing, not editing.
- [In Progress] indicator: For example, for the "V 0.0.3" version of "Symbol_1", the "Open" command will open its content and support editing.

13.4.6.2 Modifying a Template

The "Edit" command puts the template into an editable state, to meet the need to modify the template content.

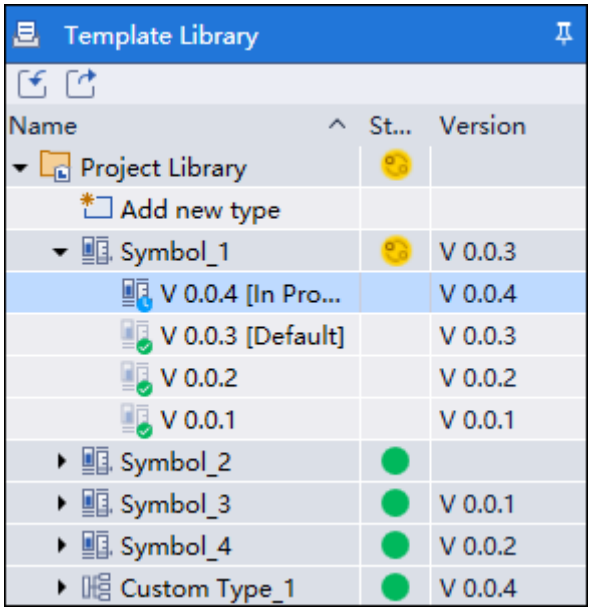
The availability of the "Edit" command in the right-click menu varies depending on the current state of template:



- Template has an [In Progress] version: the Edit command is unavailable. For example, the "V 0.0.1" version of "Symbol_2" is in [In Progress] state, so the Edit command is unavailable.
- Template does not have an [In Progress] version: the Edit command is effective, and you can modify the template content. For example, all versions of "Symbol_1" and "Symbol_3" support editing.
  A new version is automatically generated for the template when using the "Edit" command. The version number of the new version is incremented by 1 from the current version number, and the state indicator is set to [In Progress]. This new version automatically copies the content of the selected version for editing.

13.4.6.3 Publishing a Template

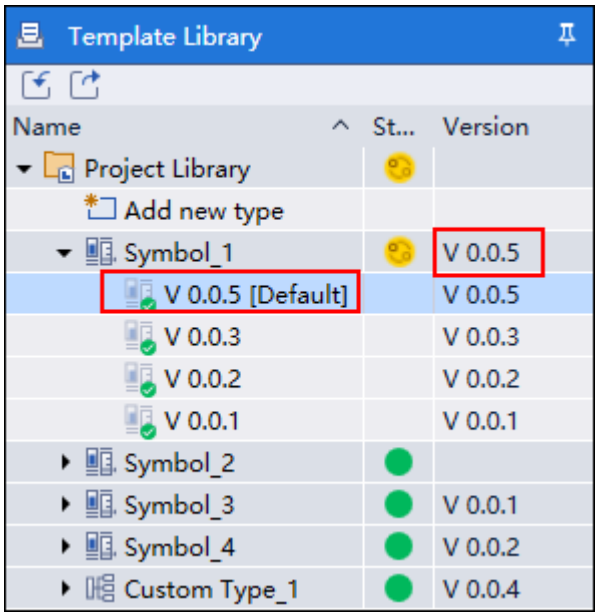
The "Release Template" command verifies the compliance of content in a specific version of a template. Successful verification is considered a successful release. After a template is successfully released, instances can be created in the user interface.



Procedure

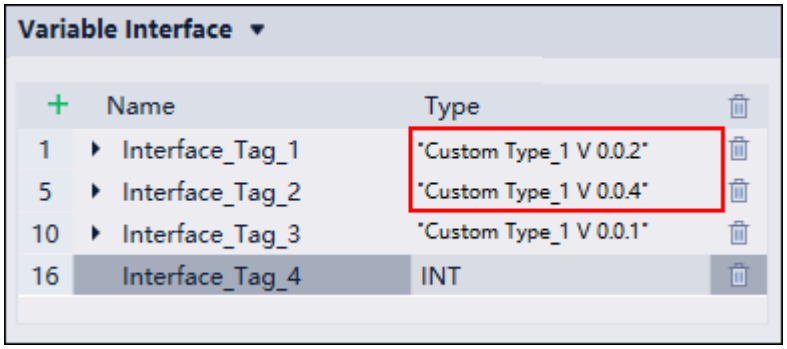
1. Select the version identified as [In Progress].
   For example, select the "V 0.0.4" node of "Symbol_1", identified by .
2. In the right-click menu, select "Publish template".

3. Confirm the symbol name and version number
   In the pop-up window, confirm the symbol name and version number. For example, change the version number to 0.0.5 and confirm.
   The identifier and version number of node have been automatically updated, and the state is identified as [Default].



Results

- Only one version of each symbol can be identified as [Default]. This version is not fixed. You can re-designate from historical versions.
- The default version is synchronized to the parent node version. If instantiation is achieved by dragging the parent node, the default version will be instantiated.
- If the variable type of **variable interface** in the symbol template uses user-defined types from the template library, the number of types is unlimited, but only one version of each type can be used. Multiple versions cannot be used simultaneously. For example, using two versions of "Custom type_1" as shown in the figure below is an incorrect operation.



13.4.6.4 Updating a Template

The "Update template" feature synchronizes the default version from the template library to various instances in devices or projects, ensuring consistency and version control.

The specific logic for the update operation varies depending on the object type in the template library:

Update operation for user-defined symbols

Assume a symbol named Cust_symbol has been created in the template library and two versions, V 0.0.1 and V 0.0.2, have been released. The referenced version of the current instance is V 0.0.2, while the default version set in the template library is V 0.0.1.

- Execute operation
  Select the Cust_symbol node and execute the "Update template" command from the right-click menu.
- Update result
  The referenced version of instance is automatically updated to "Cust_symbol V 0.0.1".

NOTE

When the template version referenced by the symbol instance is updated:

- The display area size of the symbol instance in the screen remains the same as before the update and does not automatically adjust to the new version.
- If the number or size of elements in the new version differs from the original, the instance may be partially cut off or displayed incorrectly. You need to manually adjust the display area to ensure proper display.

Update operation for HMI user-defined types

Assume a user-defined type named Cust_type has been created in the template library, with multiple versions released, including V 0.0.1, V 0.0.2, and V 0.0.3. The variable Var1 in the project currently uses data type V 0.0.1, while the default version set in the template library is V 0.0.2.

- Execute operation
  Select the Cust_type node and execute the "Update template" command from the right-click menu.
- Update result
  The data type of variable Var1 is automatically updated to "Cust_type V 0.0.2".

Update operation for PLC user-defined types

Assume a structure named PLC_1_Stru has been created in the PLC_1 device and is already associated with version V 0.0.2 in the template library. If the members of this structure are edited on the PLC side (e.g., added or removed), and version V 0.0.2 is designated as the default version in the template library.

CAUTION

Since the structure instance in the PLC device is bound to version V 0.0.2 of the template library, the update operation will only take effect when V 0.0.2 is set as the default version. Setting any other version as default will have no effect on this instance.

- Execute operation
  Select the Cust_type node and execute the "Update template" command from the right-click menu.
- Update result
  The member information of the structure PLC_1_Stru in the PLC_1 device is updated to match the content of version V 0.0.2 in the template library.

The above operation ensures that the structure definition in the device remains consistently aligned with the corresponding version in the template library, applicable to scenarios such as rapid version rollback or consistent configuration.

NOTE

When modifying structure member information, ensure that the prerequisite conditions for 13.4.4.1 Adding a User-Defined Typeare met.

13.4.6.5 Deleting a Template

Deleting a template removes all related content of the template from the current project. The delete command varies depending on the state of template.

Delete a released version

For a released version, click the "Delete" command from the right-click menu to remove it.

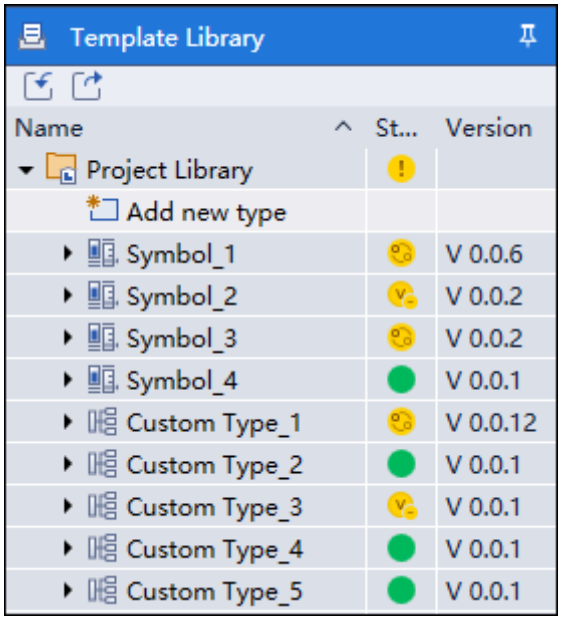
	Delete a Single Version	Delete the Entire Template
Symbol	Condition: Version is in released state with node identifier . Result: Deletes the version node in the template library and removes the instance from the user interface.	Condition: All versions under the symbol node are in released state with node identifier. Result: Deletes the symbol node in the template library and removes all version instances from the interface.
User-defined type	Condition: Version is in released state with node identifier. AND This version is not used in the symbol. Result: • PLC user-defined type: Deleting a specified version severs the connection with the node in PLCs. The identifier in the PLC  changes to . • HMI user-defined type: Deleting a specified version also removes the corresponding variable instance from the HMI variable table (if involved).	Condition: Version is in released state with node identifier . AND This type is not used in the symbol. Result: • PLC user-defined type: Deleting the entire type node severs the connection with the node in the PLCs. The identifier in the PLC  changes to . • HMI user-defined type: Deleting the entire type node also removes the corresponding variable instance from the HMI variable table (if involved).

Delete a version in editing

For a version in editing, identified by [In Progress], click the "Discard changes" command from the right-click menu to remove it.

13.4.6.6 Renaming a Template

The name of a template node can be changed, for example, the collapsed node shown in the figure below.



Naming rules:

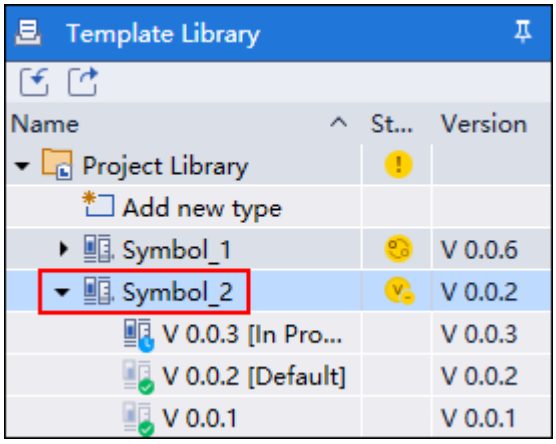
- It is composed of letters, numbers, Chinese characters, or spaces. It cannot start or end with a space.
- English is case-insensitive, for example, Aa and AA, aa and aA are considered the same name.
- The maximum name length is 128 characters.
- Unsupported special characters include \ , / , : , * , ?, " , < , > , and |.
- It cannot be the same as existing symbol and custom type node names in the template library, but can be the same as folder names.

13.4.6.7 Copying a Template

The "Copy template" feature quickly copies the content of a specific version within a template in the same project.

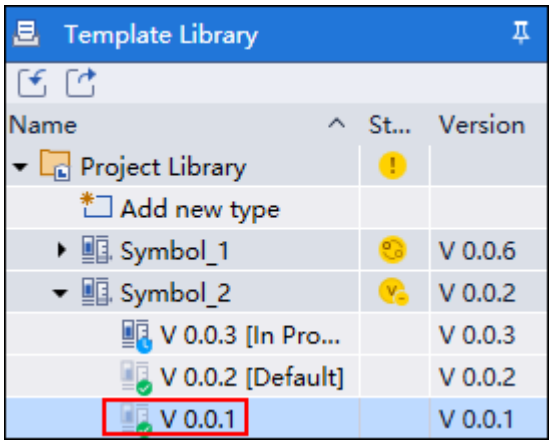
The "Copy template" command is effective in two locations:

- Template node
  1. Execute the "Copy template" command on a template node, such as the "Symbol_2" node shown in the figure.



2. Enter the new template name and version number in the pop-up window.
3. The default version V 0.0.2 of "Symbol_2" will appear in the template library with the new name and version number.

- Version node
  1. Execute the "Copy template" command on a version node, such as the V 0.0.1 version of "Symbol_2" shown in the figure.



  2. Enter the new template name and version number in the pop-up window.
  3. The default version V 0.0.1 of "Symbol_2" will appear in the template library with the new name and version number.

13.4.6.8 Exporting a Template

Templates in the iFA project export objects from the template library into **.fatpl** file format via the export function. This facilitates flexible management and sharing of released objects in the library.

Export scope

- Released versions of templates are supported for export, indicated by an icon in the lower right corner of the version node .
- If the target version references other templates, they will be exported together.

Procedure

1. In the template library, select the version node to export.
   If not selected, all released versions will be exported by default.
2. Right-click the node and select the "Export" command, or the icon.

3. Specify the file path for the exported file.
   The exported file has a **.fatpl** extension, with the default name **Library.fatpl**.

13.4.6.9 Importing a Template

The template library in iFA projects supports importing existing templates, which is applicable for importing templates from other projects into the current project, or for restoring templates that have been deleted from the project.

Import type

- Template import: Import standard templates from other projects for quick reuse.
- Version restore: Restore the project to a previous state by importing an older version file.
- During import, the software determines whether to import based on the template GUID.
  GUID information can be viewed in the properties of each node.

Import rules

- GUID and name match: If the GUID and name of the imported object exactly match those of an existing object, the software will prevent the import of that object.
- GUID matches but name differs: If the GUID matches but the name differs, the name of the imported object will be changed to match the name in the imported file.
- GUID mismatch: If the GUID does not match, the object will be imported. If a name conflict occurs, the software will automatically rename the imported object to avoid conflicts.

Procedure

1. In the template library, select the root node (project library).
2. Right-click the node and select the "Import" command, or the icon .
3. Load the import file.
   The import file has a **.fatpl extension,** with the default name **Library.fatpl**.
4. Click "OK" to complete the import operation.

Precautions

It is recommended to export and back up the current library content before importing.

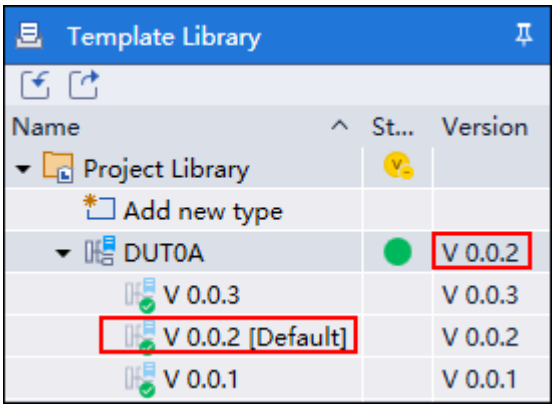
13.4.6.10 Setting as Default

The default version is the version automatically selected by the software when a template node is selected. Quick access to versions without manual selection each time is available.

When a template has multiple released versions, you can specify one of them as the default version. By default, the software automatically sets the most recently released version as the default version.

Procedure

1. Select a released version node of a template that is not identified with [Default].
2. Right-click the node and select "Set as default".
3. The current node is identified with [Default], and the template node also displays the default version number.



13.4.6.11 Synchronizing PLC Types

Synchronizing PLC type refers to synchronizing changes in structure definitions on the PLCs to the template library, enabling templating of the latest definitions.

Any of the following member changes related to structure definitions will trigger a synchronization flag.

- Name
- Data type
- Initial value
- Network public property
- Comment
- Properties
- Quantity

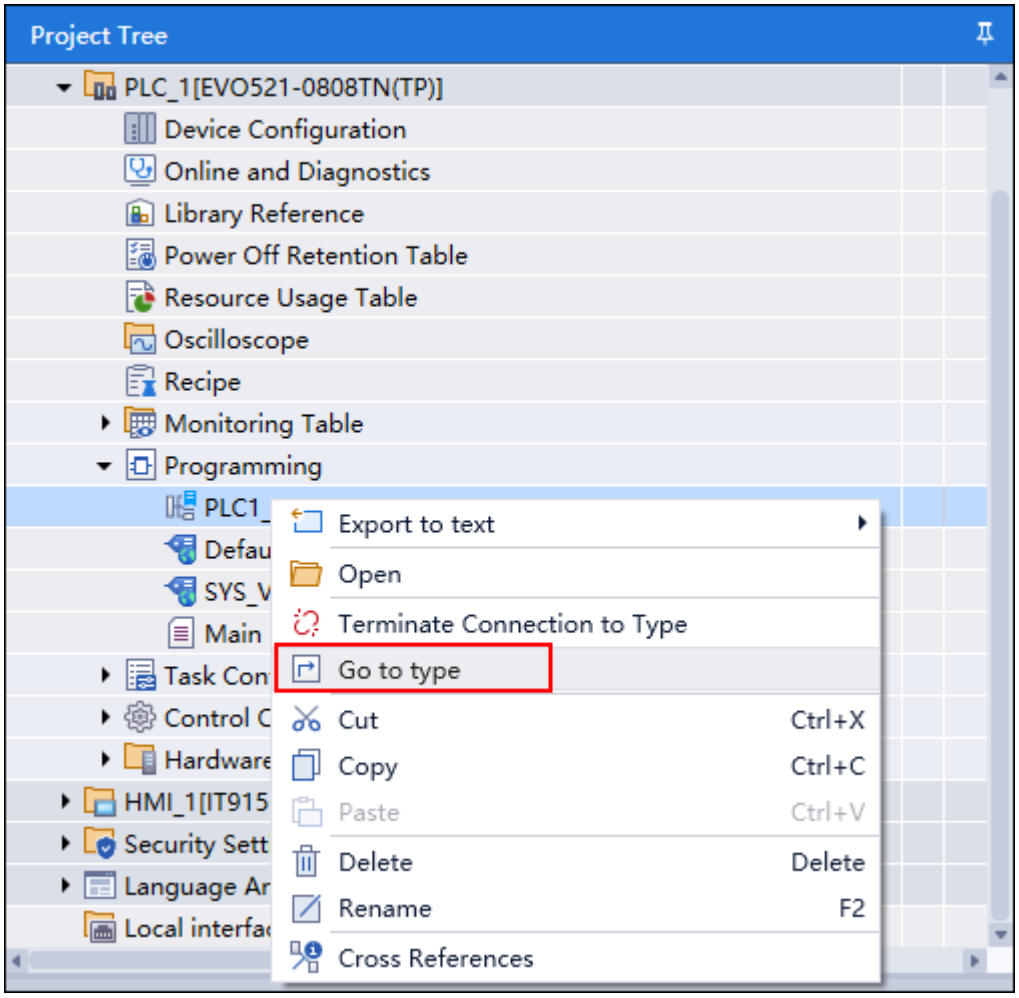
When the definition type of the PLC's structure changes, the corresponding node in the template library will be automatically marked with .

Procedure

1. Select a template version.

   In the template library, select the node version marked with .

   If there are multiple versions marked with , use the "Go to type" command in the PLC to locate the corresponding template version.



2. Initiate the synchronization command.
   Right-click the selected version and select "Sync PLC type".
3. Determine synchronization scope.
   - Select the PLC devices.
     Check the PLC(s) to sync from the list.
     If a version is applied to multiple PLCs, ensure the correct PLCs are selected.
   - Select synchronization mode.
     If the structure type has nested relationships, you can update all related templates or only update the current template.
4. Auto-update.
   After confirming the settings, the system automatically completes the synchronization.

Result

If the changes meet the prerequisites for templatization, the version will automatically be upgraded by one version; otherwise, the upgrade fails.

13.4.7 Copying and Pasting a Template Across Projects

Templates can not only be used within a single project, but also reused across projects. Whether it is an entire template library, a single template type, or a specific version (such as V 0.0.2), they can all be copied and pasted for use in different projects.

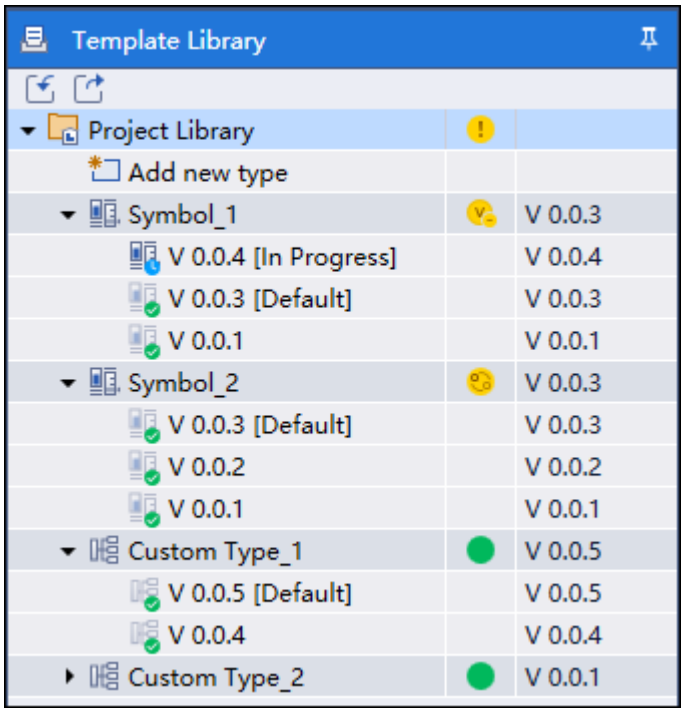
Copy

- Prerequisite
  The copy operation is limited to released templates only, excluding any template instances.

- Procedures

  Take the entire template library as an example to illustrate steps.

  1. Select the root node of the project library



  2. Right-click and select the "Copy" command

Paste

- Procedure

  1. In the template library of the target project, select the root node "Project library".

  2. Right-click and select the "Paste" command.

     First paste of the template

     The software will follow these rules:

     a. First, check whether the GUID of the copied object (viewed in the "Property" window of the type node) is the same as that of an existing type in the target project. If they are different, the paste operation is performed.

     b. Second, check for naming conflicts. If a name conflict occurs, the new type will be renamed during pasting; if no conflict exists, paste directly.

     c. Finally, check whether the property GUIDs of each version of the type are identical. If they differ, paste operation will be performed even if the version numbers (e.g. V0.0.3) are the same, which may result in two identical versions coexisting.

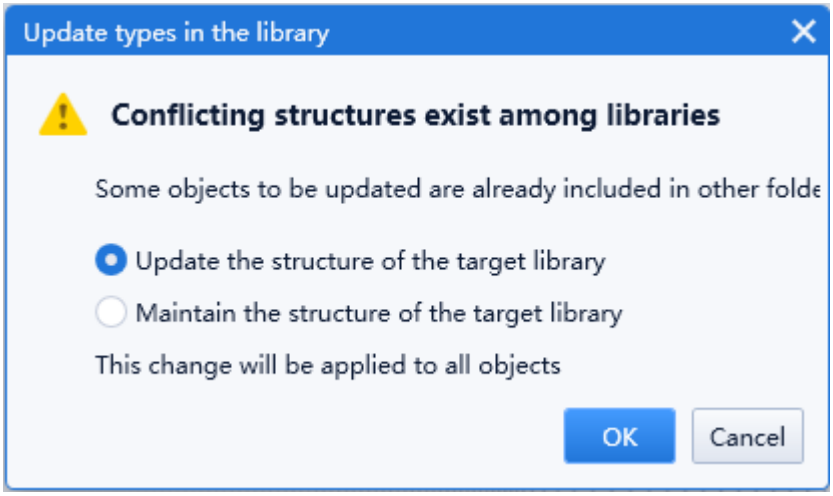
     If the copied type contains nested structures, all related types and versions will be pasted together during the paste operation.

     After the template is pasted, the status related to instance references changes to , while other statuses remain unchanged.

     Subsequent multiple pastes of the same template

     After a template is pasted into the target project, if the template in the source project is updated (for example, a new version is added) and copied to the target project again, you need to specify the update method when the hierarchy of the paste location is different from the node hierarchy in the source project.

     A settings dialog will pop up during pasting



Update the structure of target library: Maintain the node hierarchy of the template consistent with that in the source project.
Keep the structure of target library : Maintain the current node hierarchy of the pasted object within the target library without adjustment.

13.4.8 Copying and Pasting a Template Instance Across Projects

In practical applications, template instances support cross-project copy and paste, which is a key operation for rapid device configuration. During the copy and paste process, the software automatically synchronizes and pastes related templates based on instance dependencies. However, due to differences in template application scope within projects, there are limitations on the templates that can be associated during instance pasting.

Template type and application location

- PLC structure type (template): Used for PLC variable tables to define standardized data types.
- HMI structure data type: Used for HMI variable tables and dynamically bound to screens.
- Symbol template: Used as a visual unit in user screens, forming a unified style for monitoring interfaces.

Copy

1. Select the instance node in the source project tree.
2. Right-click and select the "Copy" command.

Paste

1. On the target project tree, select the node where you want to paste (e.g., the "Screen" node or the "Variable" node under the HMI device).
2. Right-click and select the "Paste" command.

Table13-4-8-1 Differences in paste results

Copied Object	Pasted Result	Remarks
HMI variable table	The referenced types in the variable table and template library will be pasted together.	If external variables are associated with public PLC variables, the association relationships in the variable table need to be manually restored after pasting.

Copied Object	Pasted Result	Remarks
HMI screen	The screen and the associated symbol templates in the template library (including the versions used), as well as the reference types they depend on, will be pasted together.	If the interface variables of the symbol template reference a template type of a PLC structure, they will be pasted together.
PLC variable table	Variable information	Before pasting the variable table, complete the copy-paste of the structure type first. Otherwise, the data type of the variable instance will cause an error, and the "Network public" status will change from "Public" to "Private".

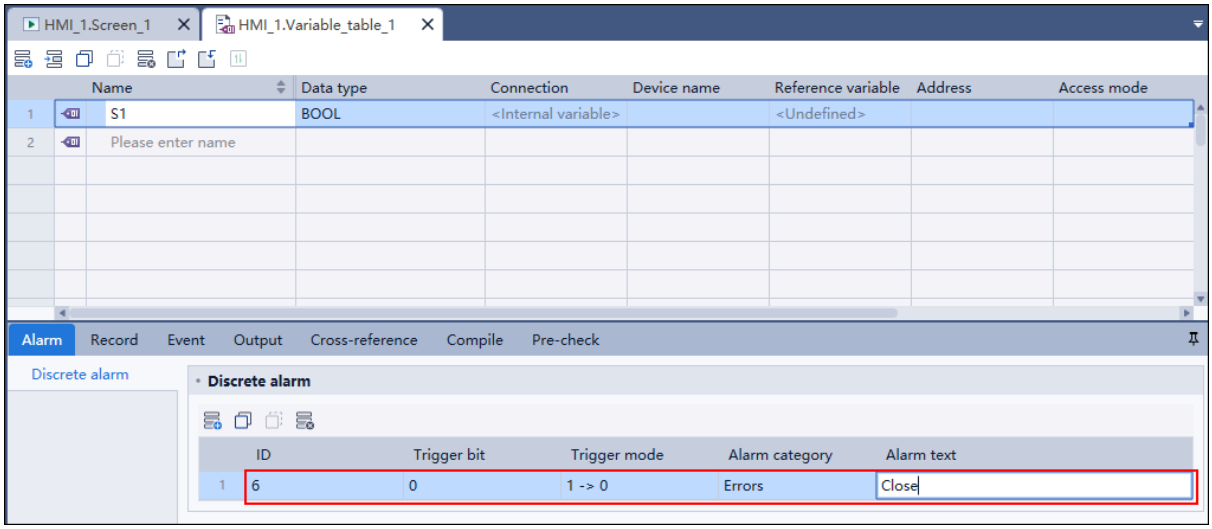
13.5 Alarm Usage

- 13.5.1 Discrete Alarm Configuration
- 13.5.2 Analog Alarm Configuration
- 13.5.3 System Event Configuration
- 13.5.4 Alarm Category Configuration
- 13.5.5 Alarm Group Configuration
- 13.5.6 Historical Alarm Records
- 13.5.8 Alarm Runtime Configuration
- 13.5.9 Alarm Application Example

13.5.1 Discrete Alarm Configuration

Discrete alarms are triggered by changes in the switch state of a variable.
To create a discrete alarm, follow these steps:

1. Double-click the variable table in the project tree to open the variable table editor.
2. Select the boolean or integer variables created in the variable table.
3. Configure the alarm under "Alarm" in the monitoring area.



Property	Description	Restriction
ID	Unique identifier for the alarm.	IDs of discrete and analog alarms must not duplicate within the same HMI device. Supports manual settings. Range: 1 to 2,147,483,647.
Trigger bit	Variable bit that triggers the alarm.	For Boolean variables, the bit number must be 0. For integer variables, the bit number must be within the allowed range of the data type. For example, the bit number can only be set to 0 to 7 for a SINT type variable.
Trigger mode	Alarm trigger mode.	Supports 0->1, 1->0, ==0, and ==1. ◦ 0->1: Rising edge alarm. ◦ 1->0: Falling edge alarm. ◦ ==1: Alarm when the value is 1. ◦ ==0: Alarm when the value is 0.
Alarm category	Categorizes alarms into different types.	Range: "Error", "Alarm", and user-defined alarm types.
Alarm Text	The message displayed when the alarm is triggered	Max. length: 256 English characters.

One variable can have multiple alarm conditions. Each HMI device supports a maximum of 5000 alarm conditions (combined total for discrete and analog alarms).

13.5.2 Analog Alarm Configuration

Users can configure alarms via the "Value alarm" tab. In the variable alarm editor, you can view the configured variable alarms.

Value alarm is used to trigger an alarm when a variable value meets certain conditions.

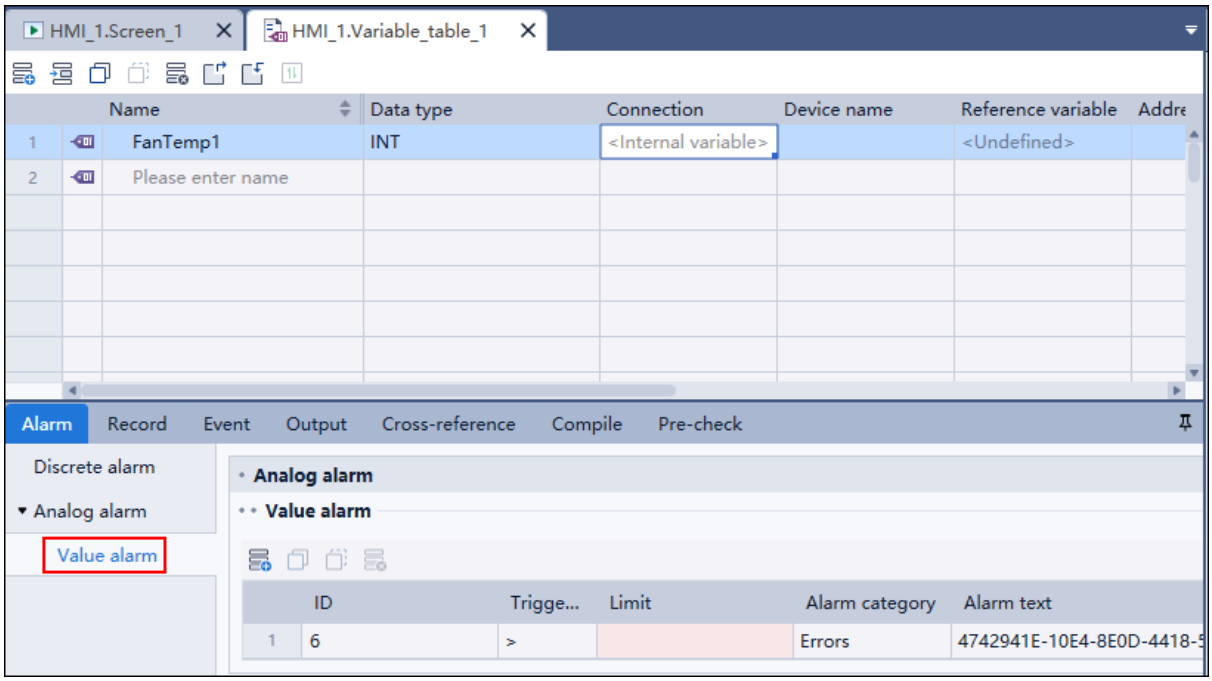
Creating value alarms

Prerequisites

- A project has been created.
- An HMI device has been added.

Procedure

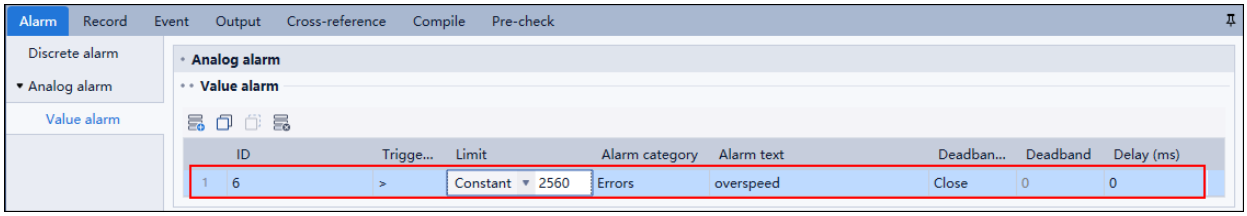
1. Expand "Variable" in the left project tree and double-click "Add" to add a variable table.
2. Add variables in the variable table.
3. Go to "Alarm > Value alarm" in the monitoring area.



4. Add rows to add value alarms.
   One variable can have multiple alarm conditions. Each HMI device supports a maximum of 5000 alarm conditions (combined total for discrete and analog alarms).

Editing value alarms

In the value alarm editor, alarm conditions can be specified.



Add or delete alarm conditions by adding or deleting rows.

The table below lists the properties of value alarms and their related explanations.

Property	Description	Restriction
ID	Unique identifier for the alarm.	IDs of discrete and analog alarms must not duplicate within the same HMI device. Supports manual settings. Range: 1 to 2,147,483,647.

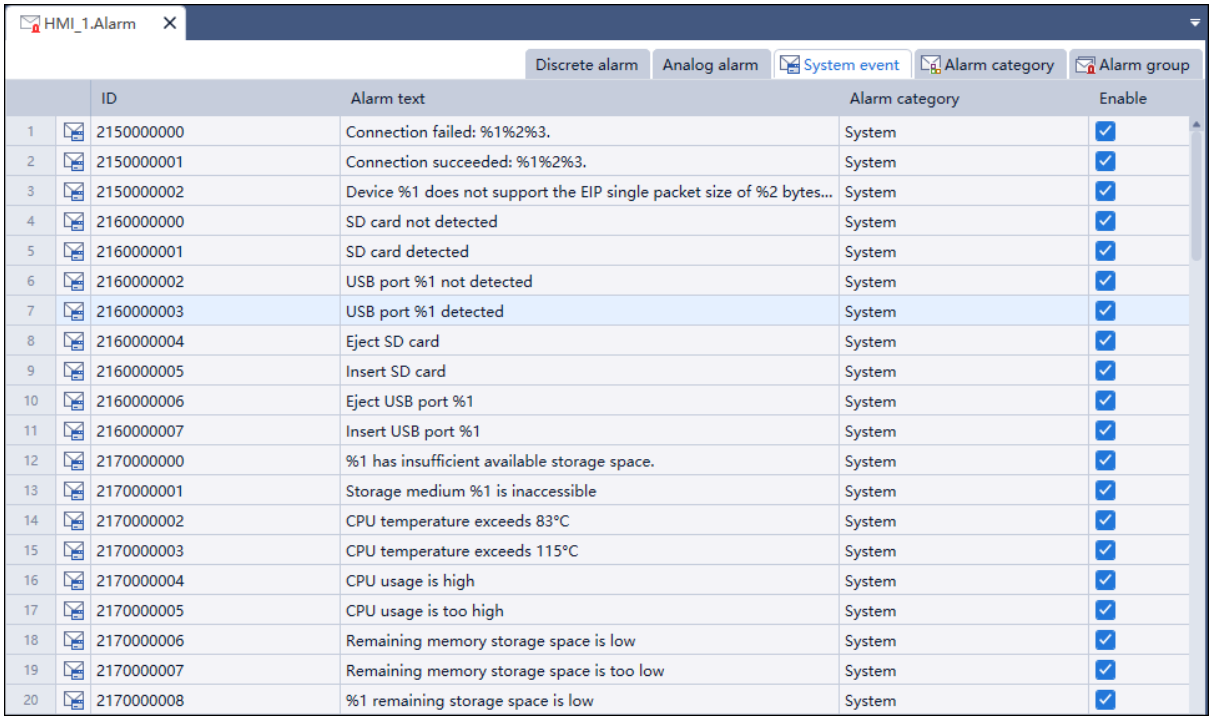
Property	Description	Restriction
Trigger mode	The condition used to compare the variable trigger value with the limit value.	Supports <, > (default), ==, !=, <=, and >=.
Limits	Value to compare against the variable trigger value to determine whether to trigger an alarm.	Default is an empty constant. It can be a constant or a variable. • Constant: The constant must not exceed the range of the data type of the variable configured for the alarm (e.g., if the variable is of type SINT, the limit range is -128 to 127). • Variable: Only integer and floating-point variables are supported, but the variable itself cannot be selected as the limit.
Alarm category	Categorizes alarms into different types.	Synchronizes with the configured alarm categories (excluding system and user-defined alarm categories).
Alarm Text	The message displayed when the alarm is triggered	Max. length: 256 English characters.
Deadband mode	Sets a variable range. If the variable value fluctuates but stays within this range, no alarm is generated.	Options: Disabled (default), "When activated", "When restored", "When activated/restored". The dead zone mode is grayed out when the trigger mode is set to == or !=.
Deadband	Variable range of the deadband.	The maximum value cannot exceed the range of the data type.
Delay (ms)	When the variable meets the alarm condition, the trigger condition is re-evaluated after the delay time. • If the condition is still met, the alarm is triggered. • If the variable value changes and the alarm would recover, the timer stops.	The default value is 0 ms, with a range of 0 to 600000.

13.5.3 System Event Configuration

The HMI project supports system event configuration. System events refer to alarms or status information generated by internal system modules (such as VNC, resources, and scripts). They can also be regarded as system alarms triggered when runtime behavior deviates from expectations.

Double-click "Alarm" under an HMI device in the project tree. On the "System event" tab, check the event types to be monitored and recorded. Only checked events will be monitored and recorded by the system.

1.



	ID	Alarm text	Alarm category	Enable
1	2150000000	Connection failed: %1%2%3.	System	☑
2	2150000001	Connection succeeded: %1%2%3.	System	☑
3	2150000002	Device %1 does not support the EIP single packet size of %2 bytes...	System	☑
4	2160000000	SD card not detected	System	☑
5	2160000001	SD card detected	System	☑
6	2160000002	USB port %1 not detected	System	☑
7	2160000003	USB port %1 detected	System	☑
8	2160000004	Eject SD card	System	☑
9	2160000005	Insert SD card	System	☑
10	2160000006	Eject USB port %1	System	☑
11	2160000007	Insert USB port %1	System	☑
12	2170000000	%1 has insufficient available storage space.	System	☑
13	2170000001	Storage medium %1 is inaccessible	System	☑
14	2170000002	CPU temperature exceeds 83°C	System	☑
15	2170000003	CPU temperature exceeds 115°C	System	☑
16	2170000004	CPU usage is high	System	☑
17	2170000005	CPU usage is too high	System	☑
18	2170000006	Remaining memory storage space is low	System	☑
19	2170000007	Remaining memory storage space is too low	System	☑
20	2170000008	%1 remaining storage space is low	System	☑

The following describes each type of configurable system event. The system event ID consists of 10 digits, with the following encoding rules:

- First three digits: Represent the event category. The specific correspondence is as follows:
  - 215: Communication events
  - 216: External device events
  - 217: System resource events
  - 218: Alarm events
  - 219: Variable events
  - 220: Recording events
  - 221: User management events
  - 222: Script events
  - 223: System events
  - 224: Recipe events
- Last seven digits: Represent specific events within each category.

Communication events (215)

Alarm ID	Alarm Text	Description
2150000000	Connection failed: %1%2%3	Triggered when a connection to another device fails. The first field is the connection name, the second is the device name, and the third is the link information. If the device name is empty, it is omitted. For USB devices, link information is not displayed. **Ethernet port example** Connection failed: Connection_1, PLC_1, 192.168.0.1. Connection failed: Connection_1, 192.168.0.1. **Serial port example** Connection failed: Connection_1, PLC_1, COM1. Connection failed: Connection_1, COM1. **USB example** Connection failed: Reader 1, PLC_1. Connection failed: Reader 1.
2150000001	Connection succeeded: %1%2%3	Triggered when a connection to another device succeeds.
2150000002	Device %1 does not support the EIP packet length (%2 bytes) set by FA. The maximum supported length is %3 bytes.	For Inovance EIP protocol, if the configured maximum packet length exceeds the PLC's supported limit, it is automatically adjusted to the supported range. Example: The maximum packet length for Connection_1 is automatically adjusted to 1994 bytes.

External device events (216)

Alarm ID	Alarm Text	Description
2160000000	SD card not detected	This event is sent at device startup.
2160000001	SD card detected	
2160000002	USB port %1 not detected	This event is sent at device startup. Example: USB interface USB1 not detected.
2160000003	USB port %1 detected	
2160000004	SD card removed	This event is sent when an SD card is removed/inserted during runtime.
2160000005	SD card inserted	
2160000006	USB port %1 removed	This event is sent when a USB storage device is removed/inserted during runtime. Example: USB port USB1 removed.
2160000007	USB port %1 inserted	

System resource events (217)

Alarm ID	Alarm Text	Description
2170000000	%1 has insufficient available storage space	Triggered when there is not enough storage space for data storage operations. Example: The SD card has insufficient available storage space.
2170000001	%1 storage medium is inaccessible	Triggered when the storage medium cannot be accessed. Example: The SD card storage medium is inaccessible.
2170000002	CPU temperature exceeds 83°C	Triggered when the HMI device CPU temperature exceeds 83°C.
2170000003	CPU temperature exceeds 115°C	Triggered when the HMI device CPU temperature exceeds 115°C.
2170000004	CPU usage is high	Triggered when CPU usage is ≥ 80%.
2170000005	CPU usage is too high	Triggered when CPU usage is ≥ 90%.
2170000006	Remaining memory space is low	Triggered when memory usage is ≥ 80%. The meaning of memory space is consistent with the system variable.
2170000007	Remaining memory space is too low	Triggered when memory usage is ≥ 90%.
2170000008	%1 remaining storage space is low	Triggered when storage usage on the runtime data path or recording path is ≥ 80%.
2170000009	%1 remaining storage space is too low	Triggered when storage usage on the runtime data path or recording path is ≥ 90%.

Alarm events (218)

Alarm ID	Alarm Text	Description
2180000000	Overflow, pending alarms will be discarded	Triggered when too many alarms are triggered in a short time. When more than 10,000 alarms are queued for processing, the real-time alarm view will not display all alarms, and alarm data will not be written to the log file. The upper limit varies by specific device.
2180001000	%1 unlocked successfully	-
2180001001	All installments unlocked successfully	All installments unlocked successfully.
2180001002	%1 unlock failed	-
2180001003	System time modification failed	System time modification failed.

Alarm ID	Alarm Text	Description
2180001004	Trusted time recovery succeeded after RTC failure; restored to %1	When the device's internal clock (RTC) resets to its initial value (e.g., 1970) due to battery depletion or other reasons, the system will automatically detect this anomaly. It then uses the last saved trusted time record to restore the current time and generates this alarm. The "%1" in the alarm will be automatically replaced with the actual date and time restored to.

Recording events (220)

Alarm ID	Alarm Text	Description
2200000000	Record: %1, full	Triggered when a data record or alarm record becomes full. Example: Record: Data record_1, full.
2200000001	Record: %1, fill level has reached %2%	Triggered when a data record or alarm record reaches its fill threshold. Example: Record: %1, fill level has reached %2%
2200000002	Storage log task backlog of %1 entries, backlogged data discarded	Example: Storage log task backlog of 1 entry, backlogged data discarded
2200000003	Data record %1 cleared	Example: Data record 1 from the operation record cleared.
2200000004	Failed to clear data record %1	Example: Failed to clear data record 1 from the operation record.
2200000005	Alarm record %1 cleared	Example: Alarm record 1 from the operation record cleared.
2200000006	Failed to clear alarm record %1	Example: Failed to clear alarm record 1 from the operation record.
2200000007	Operation record %1 cleared	Example: Operation record 1 from the operation record cleared.
2200000008	Failed to clear operation record %1	Example: Failed to clear operation record 1 from the operation record.
2200000009	All data records deleted before loading project	-
2200000010	All alarm records deleted before loading project	-
2200000011	All operation records deleted before loading project	-

User management events (221)

Alarm ID	Alarm Text	Description
2210000000	User %1 has logged in	One login event and one logout event are recorded when switching users.
2210000001	User %1 has logged out	
2210000002	Login failed: Invalid user or password	-
2210000003	Login failed: User is locked	-
2210000005	Logout failed	-
2210000006	Password for user %1 has been changed	Example: Password for user User1 has been changed.
2210000007	Role of user %1 has been changed from %2 to %3	Example: Role of user User1 has been changed from monitoring observer to monitoring observer/ monitoring operator.
2210000008	Logout time for user %1 has been changed from %2 to %3	Example: Logout time for user User1 has been changed from 5 minutes to 10 minutes.
2210000009	User %1 has been renamed to %2	Example: User User1 has been renamed to User2.
2210000010	Comment for user %1 has been changed from %2 to %3	Example: Comment for user User1 has been changed from Comment1 to Comment2.
2210000011	User %1 added	Example: User User1 added.
2210000012	User %1 deleted	Example: User User1 deleted.
2210000013	Export users failed	Example: Export users failed.
2210000014	Export users succeeded	Example: Export users succeeded.
2210000018	Import users failed	Example: Import users failed.
2210000019	Import users succeeded	Example: Import users succeeded.
2210000025	User %1 locked for %2	Manual lock. Example: User User1 locked for 1 minute.
2210000026	User %1 unlocked	Manual unlock. Example: User User1 unlocked.
2210000027	User %1 has made %2 consecutive failed login attempts and the account has been automatically locked	Example: User User1 has made 1 consecutive failed login attempt and the account has been automatically locked.
2210000028	Lock time for user %1 has expired, and the account has been automatically unlocked	Example: Lock time for user User1 has expired, and the account has been automatically unlocked.
2210000029	Failed to get current user login information: %1	-

Alarm ID	Alarm Text	Description
2210000030	User management data cleared before loading project	-

Script events (222)

Alarm ID	Alarm Text	Description
2220000000	Overloaded, script <%1> is rejected	Triggered when too many scripts are triggered in a short time. When more than 128 scripts are queued for processing, all subsequent scripts will be rejected. The upper limit varies by specific device. Example: Overloaded, script <script1> is rejected.
2220000001	Error at line %2 in script <%1>, %3	Triggered when an error occurred in the script, and script execution will be aborted. Example: Error at line 10 of script <script1>, division by zero.

System events (223)

Alarm ID	Alarm Text	Description
2230000000	Start runtime system	Recorded when the runtime system starts.
2230000001	Stop runtime system	Recorded when the runtime system stops. This event can be recorded when manually stopped, but cannot be recorded during power failure.
2230000002	Project downloaded from USB %1 succeeded	Example: Project downloaded from USB1 succeeded.
2230000003	Project downloaded from USB %1 failed	Example: Project downloaded from USB1 failed
2230000004	Project downloaded from SD card succeeded	Example: Project downloaded from SD card succeeded.
2230000005	Project downloaded from SD card failed	Example: Project downloaded from SD card failed.
2230000006	Project downloaded from software %1 succeeded	Example: Project downloaded from IDE succeeded.
2230000007	Project downloaded from software %1 failed	Example: Project downloaded from IDE failed.
2230000008	Project uploaded to software %1 succeeded	Example: Project uploaded to IDE succeeded.
2230000009	Project uploaded to software %1 failed	Example: Project uploaded to IDE failed.
2230000010	Firmware updated from USB %1 succeeded	Example: Firmware updated from USB1 succeeded.

Alarm ID	Alarm Text	Description
2230000011	Firmware updated from USB %1 failed	Example: Firmware updated from USB1 failed.
2230000012	Firmware updated from SD card succeeded	Example: Firmware updated from SD card succeeded.
2230000013	Firmware updated from SD card failed	Example: Firmware updated from SD card failed.
2230000014	Firmware updated from software %1 succeeded	Example: Firmware updated from IDE succeeded.
2230000015	Firmware updated from software %1 failed	Example: Firmware updated from IDE failed.
2230000016	Factory reset succeeded (excluding physical keys)	-
2230000017	Factory reset failed (excluding physical keys)	-
2230000018	Device IP changed from %1 to %2	Recorded when the IP address changes. Example: Device IP changed from 192.0.0.1 to 192.0.0.2.
2230000019	Device IP changed from %1 to %2 failed: Invalid IP address	Example: Device IP changed from 192.0.0.1 to 192.0.0.2 failed: Invalid IP address
2230000020	System time changed from %1 to %2	Recorded when the system time changes. Example: System time changed from 2025/08/19 16:00:00 to 2025/08/22 16:00:00.
2230000021	System time changed from %1 to %2 failed: Invalid time	Triggered when the user attempts to change the device time to a non-existent or invalid time value. "%1" represents the original time, and "%2" represents the target time that failed. Example: Failed to change system time from August 1 to August 32 (non-existent).
2230000022	Failed to change system time from %1 to %2: Installment plan or non-permanently locked user exists	Example: Failed to change system time from 2025/08/19 16:00:00 to 2025/08/22 16:00:00: Installment plan or non-permanently locked user exists.

Recipe events (224)

Alarm ID	Alarm Text	Description
2240000000	Recipe %1 import succeeded	Example: Recipe Recipe1 import succeeded.
2240000001	Recipe %1 import failed	Example: Recipe Recipe1 import failed.
2240000002	Recipe data record %1 cleared	Example: Recipe data record Recipe1: Data record1 cleared.

Alarm ID	Alarm Text	Description
2240000003	Failed to clear recipe data record %1	Example: Failed to clear recipe data record Recipe1: Data record1.
2240000004	All recipe records deleted before loading project	-

Steps

1. Double-click "Alarm" in the left project tree to open the alarm editor.
2. Click the "System event" tab.
3. Configure system events according to project requirements.
   By default, all system events are enabled. Users can check/uncheck to modify the enabled status.

13.5.4 Alarm Category Configuration

The alarm category editor supports viewing and editing predefined alarm categories. It also supports creating user-defined alarm categories and configuring properties.

- Predefined alarm categories
  Predefined alarm classes cannot be deleted; they can only be edited within limited scope. There are predefined alarm categories created by the system for each device (in "Alarm > Alarm category").
- User-defined alarm categories
  Users can create new alarm classes in "Alarm > Alarm category" and configure how alarms are displayed.

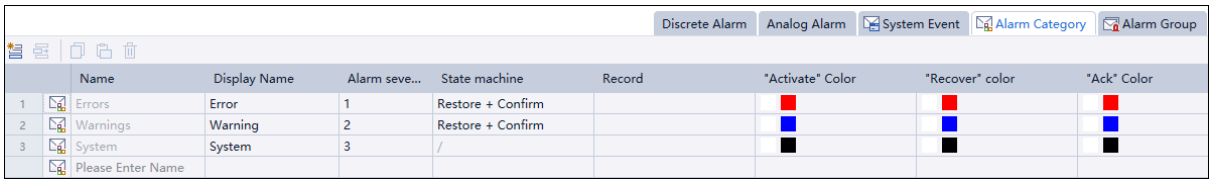
Creating an alarm category

In the "Alarm category" table editor, you can create user-defined alarm categories and configure their properties. A maximum of 32 alarm categories can be created in total.

Procedure

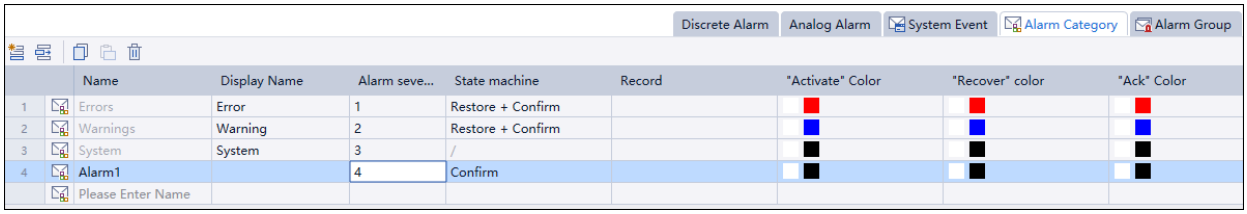
To create an alarm category, follow these steps:

1. Double-click "Alarm" in the left project tree to open the alarm editor.
2. Click the "Alarm category" tab to display predefined alarm categories and existing user-defined alarm categories. The table below lists the predefined alarm categories.



3. Create a new alarm class using the following three methods:
   - Click the "Add row" icon in the toolbar above the table header.
   - Enter a name in the virtual row, then click elsewhere.
   - Right-click and select "Add row".

All of the above methods will automatically create a default alarm category.



Editing an alarm category

In the alarm category editor, you can edit the properties of each category. The properties are described in the following table.

Item	Description
Name	Alarm category name is unique. The name can contain Chinese and English characters, numbers, underscores ("_"), and hyphens ("-"), with a maximum length of 128 characters and cannot be empty. The names of the predefined alarm categories are respectively "Error", "Alarm", and "System". They cannot be modified.
Display name	Descriptive text for the alarm category. Max. length: 128 characters. Default is empty. For predefined categories, the display name defaults to the same value as the name ("Error", "Alarm", and "System").
Alarm level	Used by operators to distinguish the urgency of alarms. The configurable range is 1 to 999, with smaller numbers indicating higher urgency. Default emergency order: Error > Alarm > System.
Alarm reset method	Options include "Restore + Confirm" (default), "Restore", "Confirm", and "None". Restore + Confirm: The alarm is removed from the real-time alarm list only after both the restore condition is met and the user confirms it. Confirm: The alarm is removed from the real-time alarm list only after the user confirms it. Restore: The alarm is automatically removed from the real-time alarm list after the restore condition is met. None: The alarm does not appear in the real-time alarm list; this is used only for "System" alarms.
Record	You can select from the configured alarm records. Different alarm categories can use the same record or different records.
Activated color	Sets the background color, font color, and whether the alarm flashes when it is activated. By default, the background color is white, the font color is black, and the alarm does not flash. The font color for error categories defaults to red, alarm categories defaults to blue, and system categories defaults to black. The background color for all three categories defaults to white, with no flashing.

Item	Description
Restored color	Sets the background color, font color, and whether the alarm flashes when it is restored. The configuration method and default colors are the same as for the activated color. When the alarm reset method is "Restore" or "Restore + Confirm", the restored color can be modified; otherwise, modification is disabled.
Acknowledged color	Sets the background color, font color, and whether the alarm flashes when it is confirmed. The configuration method and default colors are the same as for the activated color. When the alarm reset method is "Restore" or "Restore + Confirm", the acknowledged color can be modified; otherwise, modification is disabled.

The alarm category editor also supports copy, paste, insert, delete, undo/redo, and find/replace functions.

13.5.5 Alarm Group Configuration

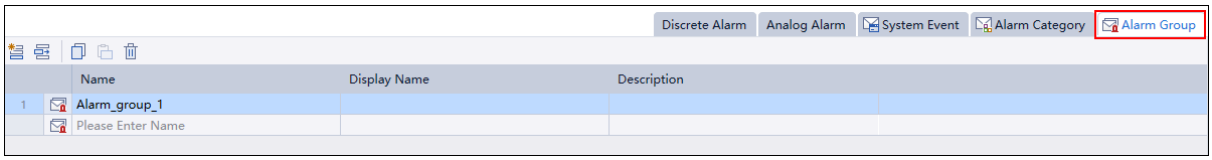
To facilitate viewing, recording, and differentiation, alarm information generated by variables can be divided into different groups.

Assign alarms to various alarm groups based on the actual relationship between alarms. For example, group alarms based on error cause (e.g., "power-off") or error source (e.g., "Motor 1"). Each variable can be configured with one alarm group, and alarm information can be assigned to that alarm group.

Creating an alarm group

Create an alarm group by following these steps:

1. Double-click "Alarm" in the left project tree to open the alarm editor.
2. Click the "Alarm group" tab.



3. There are three ways to create an alarm group.
   - Click the "Add row" icon in the toolbar above the table header.
   - Enter the name in the virtual row (i.e., the last row), then click elsewhere or press the Enter key.
   - Right-click the menu row and select "Add row".

Editing an alarm group

In the alarm group editor, you can edit the properties of the alarm group. The properties are described in the following table.

Item	Description
Name	The name of the alarm group. The name must be unique and cannot be empty. The name can contain Chinese and English characters, numbers, underscores ("_"), and hyphens ("-"), with a maximum length of 128 characters.
Description	Description of the alarm group. The maximum length is 128 characters.

The alarm group editing table also supports copy, paste, insert, and delete operations.

13.5.6 Historical Alarm Records

The alarm records store the alarm information generated during project operation, providing data for problem analysis and troubleshooting of project and device issues.

In "Record > Alarm record", you can create and edit the properties of alarm records.

Record content

The alarm to be recorded is assigned to the alarm record by alarm class. When the alarm status changes, the alarm record will hold the following information for the corresponding alarm:

- Alarm number
- Device name
- Alarm variable name
- Date and time the alarm occurred
- Alarm type
- Alarm level
- Alarm information
- Username
- Alarm value
- Data type of the alarm value
- Alarm group
- Alarm status
- Alarm activation time
- Alarm acknowledge time

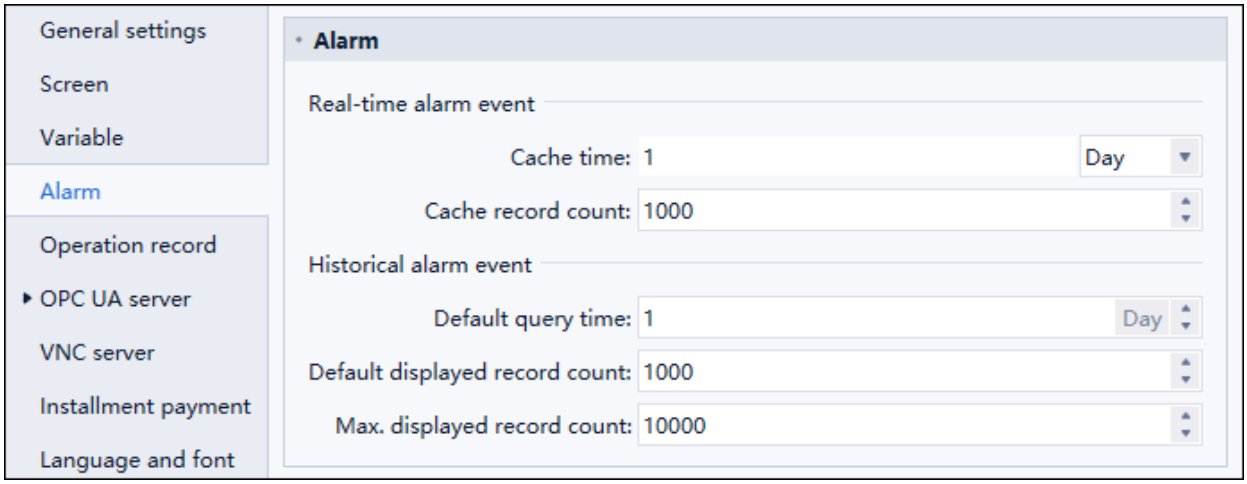
Recording timing

An alarm record will be generated when the alarm occurs, is confirmed and is restored.

Alarm records can be filtered, have their storage content controlled, and be displayed using controls. For details, see 13.3.5.4.1 Alarm View.

Controlling record storage size

Double-click "Runtime setting" in the project tree. On the "Alarm" tab, set "Real-time alarm event" and "Historical alarm event" to save records based on retention time or number of entries.

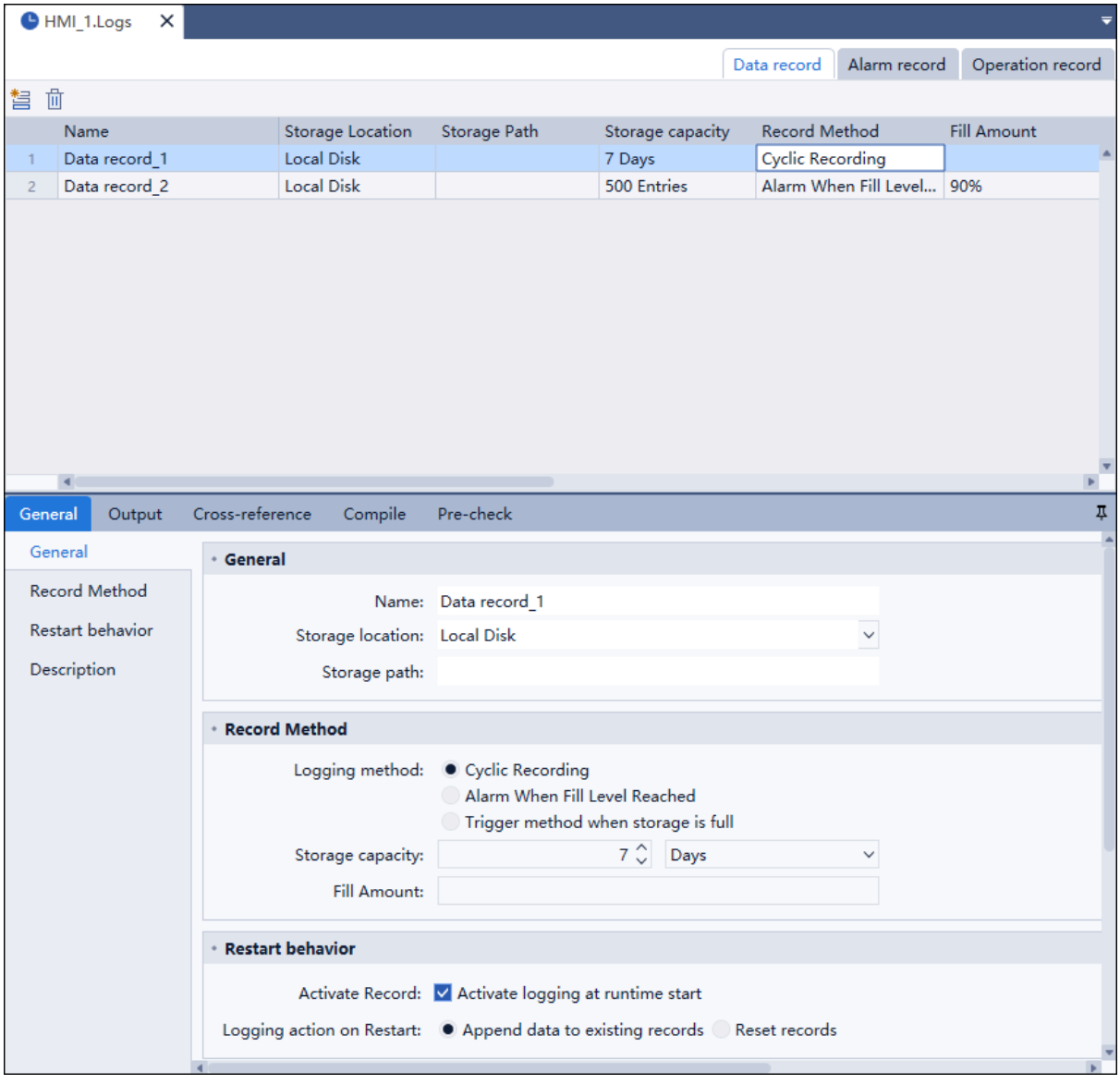


Record setting

1. Add an alarm record

   Double-click the "Record" node in the project tree and select the "Alarm record" tab.

   Click  to add an alarm record.



   The property items of an alarm record are the same as those of a data record. For details, see 13.2.5.6.1 Setting Recording Rules for Data.

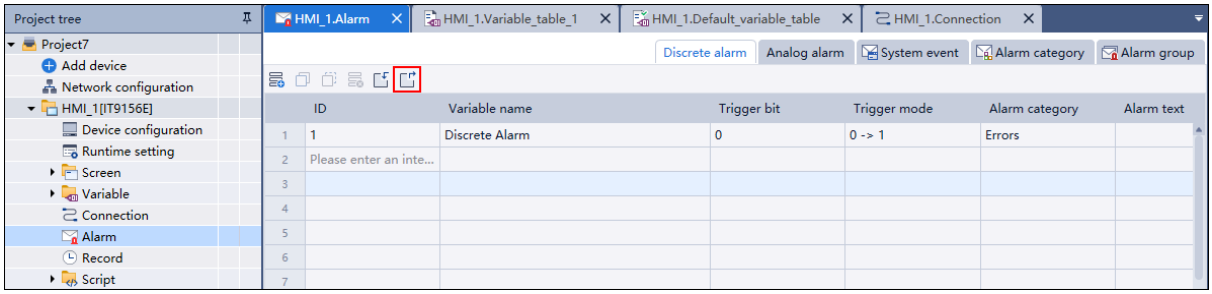
2. Associate alarm records in the alarm category
   Double-click the "Alarm" node, select the "Alarm category" tab, and associate alarm records in the "Record" column.

   After an alarm record is associated with a specific category, the system displays the corresponding alarm information based on the category's level and color. This enables users to quickly identify the urgency or priority of the alarm.

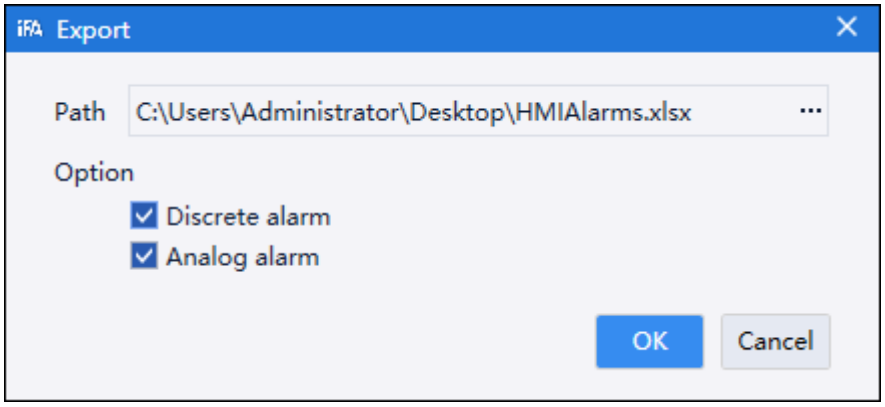
13.5.7 Alarm Import/Export

Exporting alarms

1. Select an HMI device in the left project tree and double-click "Alarm".
2. Click the export icon .

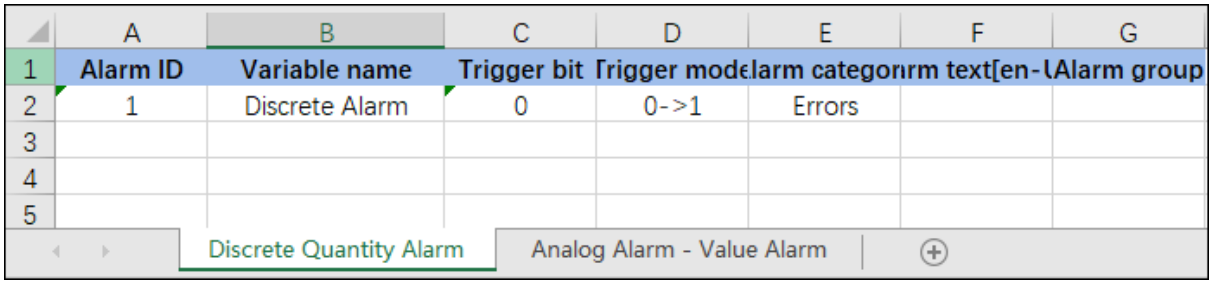


3. Select the file export path and check the alarm categories to export, then click "OK".



Parameter	Description	Additional information
Path	◦ The export path defaults to "C:\Users\Your Username\Desktop". ◦ The file name defaults to "HMIAlarms.xlsx".	◦ Export path and filename allow users to modify. ◦ The export file format is xlsx.
Option	◦ Discrete alarm ◦ Analog alarm	Both types can be exported simultaneously.

4. Wait until the system prompts "Export completed", then click "OK".
5. Open HMIAlarms.xlsx. Users can view the relevant information for each alarm ID in the "Discrete alarm" and "Analog alarm - value alarm" tables respectively. The exported content is as follows:

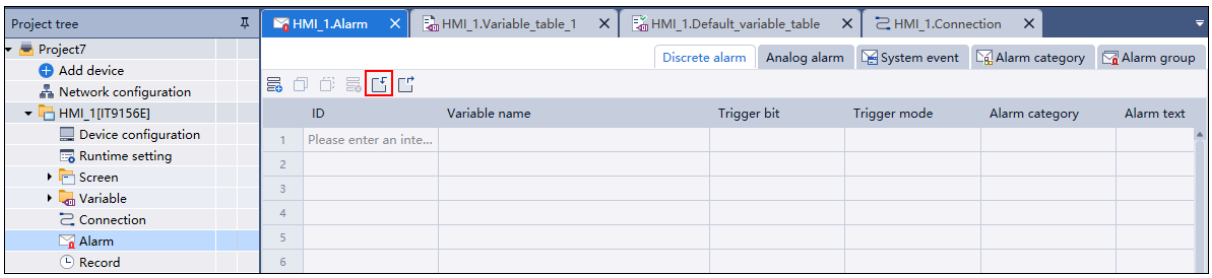


	A	B	C	D	E	F	G
1	Alarm ID	Variable name	Trigger bit	Trigger mode	Alarm categor	rm text[en-U	Alarm group
2	1	Discrete Alarm	0	0->1	Errors		
3							
4							
5							

Discrete Quantity Alarm | Analog Alarm - Value Alarm

Importing alarms

1. Select an HMI device in the left project tree and double-click "Alarm".
2. Click the import icon .



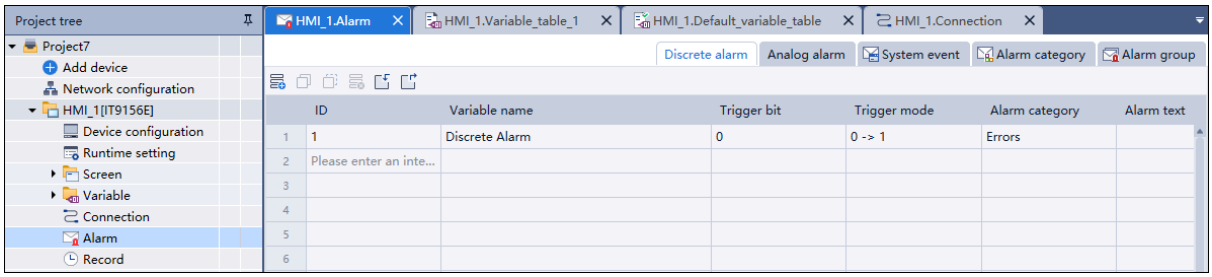
3. Select the xlsx format file to import and click "Open".

> **NOTE**
>
> - Alarm IDs to import must be unique. If the same ID exists, the new alarm will overwrite the old one.
> - The system does not delete existing data; it only performs add or overwrite operations.
> - If an alarm ID is invalid (empty, out of range, or non-numeric), that alarm row will not be imported.

4. Wait until the system prompts "Import completed", then click "OK".
5. (Optional) If there are exceptions during data import, click "Log file" to view the exception log and handle the exceptions as indicated.

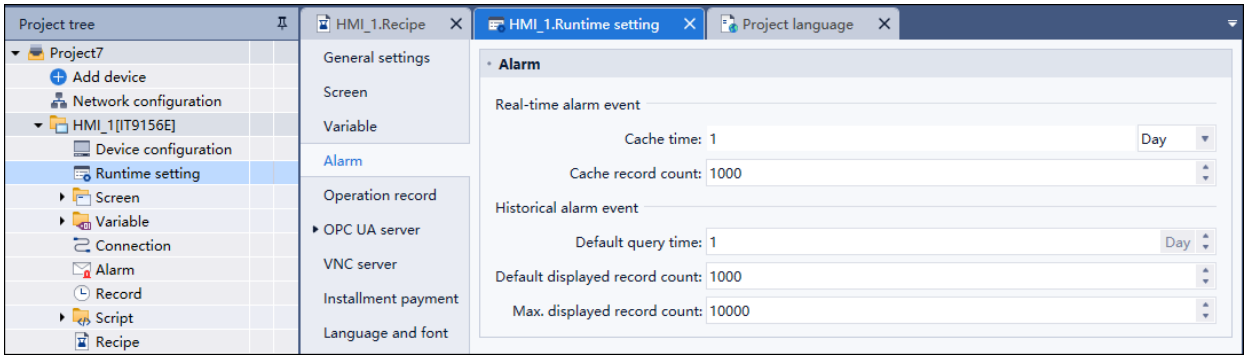


6. The imported content is as follows:



13.5.8 Alarm Runtime Configuration

During online operation of the alarm view, the display settings related to the real-time and historical alarm events are configured on the "Alarm" tab of the "Runtime setting" node.



- Real-time alarm events
  Users can configure the alarm cache duration and the number of events displayed on "Real-time alarm event" tab in the alarm view. This allows users to review the latest alarm events within a certain time.
  Cache time: The duration during which any real-time alarm event is recorded from the moment it is generated. Range: 1 hour to 30 days.
  Cached record count: The maximum number of latest real-time alarm events cached. When the limit is reached, the system overwrites the oldest events with the newest ones. Range: 100 to 10,000.
  Downloading a project clears the real-time alarm event cache, and the cache count and duration for existing events will be reset.
- Historical alarm events
  Users can configure the alarm cache duration, and the default and maximum number of events displayed on "Historical alarm event" tab in the alarm view.
  Default query time: The duration (in days) used to query historical alarm data. This is calculated backwards from the moment the "Historical alarm event" page is opened. Range: 1 to 30.
  Default displayed record count: The number of alarm events displayed on the "Historical alarm event" page when no filter is applied. Range: 100 to 10,000.
  Max. displayed record count: The maximum number of filtered historical alarm events that can be displayed on the page. Range: 10,000 to 1,000,000.

13.5.9 Alarm Application Example

This section demonstrates how to use the alarm function to monitor the rotational speed states of an MS1 servo motor during production.

Assume the variable "rotor_speed" (type: UINT; unit: rpm) monitors the motor's rotational speed. In this case, value alarm and discrete alarm are adopted to provide notifications for abnormal speed states of the motor. First, the value alarm sets the minimum and maximum speed for the motor. When the actual speed exceeds the set range, an alarm is issued. If the motor speed is too slow, slightly slow, too fast, or slightly fast, it will trigger the corresponding value alarm. Second, the discrete alarm detects excessive acceleration based on speed rate levels. If the acceleration exceeds the acceptable limit, a discrete alarm is issued.

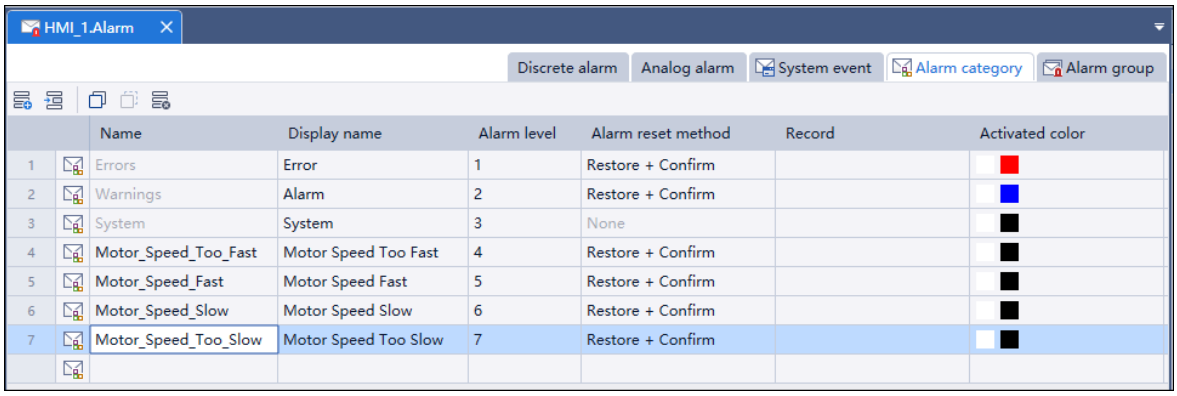
Configure alarms to monitor abnormal states of rotor_speed and use an alarm view control to display alarm information.

Prerequisites

- A variable of type UINT named "rotor_speed" has been created.

Procedure

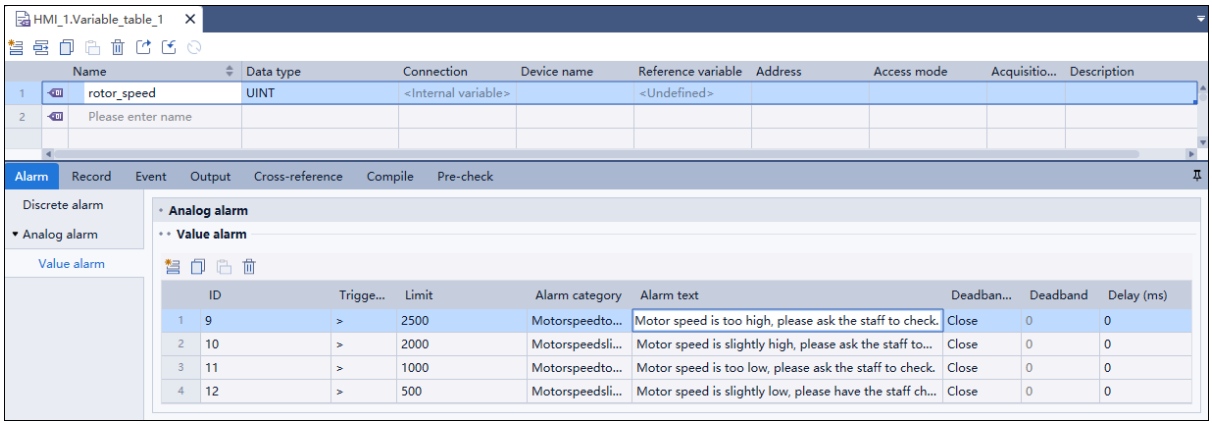
1. Define the alarm category.
   a. Click "Alarm" in the left project tree to open the alarm editor and select the "Alarm category" tab.
   b. Add the following user-defined alarm categories: "Motor speed too fast", "Motor speed fast", "Motor speed slow", and "Motor speed too slow".



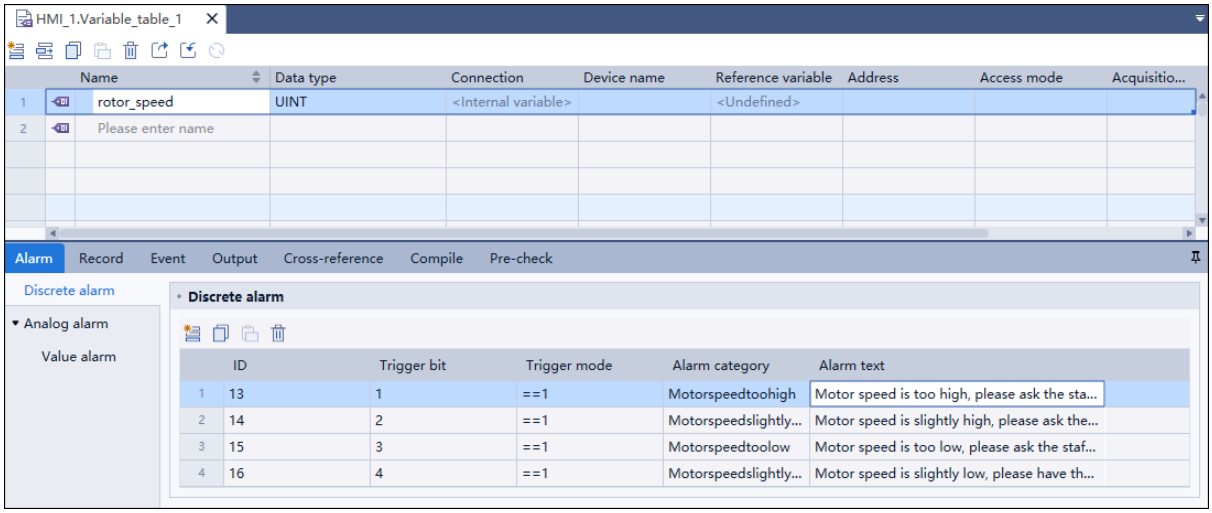
2. Add an alarm group
   On the "Alarm group" tab in the alarm editor, create a new alarm group named "Motor MS1", and add descriptive information based on the specific application scenario.



3. Configure the value alarm
   Open the variable editor for "rotor_speed". In the monitoring area, select "Alarm > Value alarm" and set the value alarm.

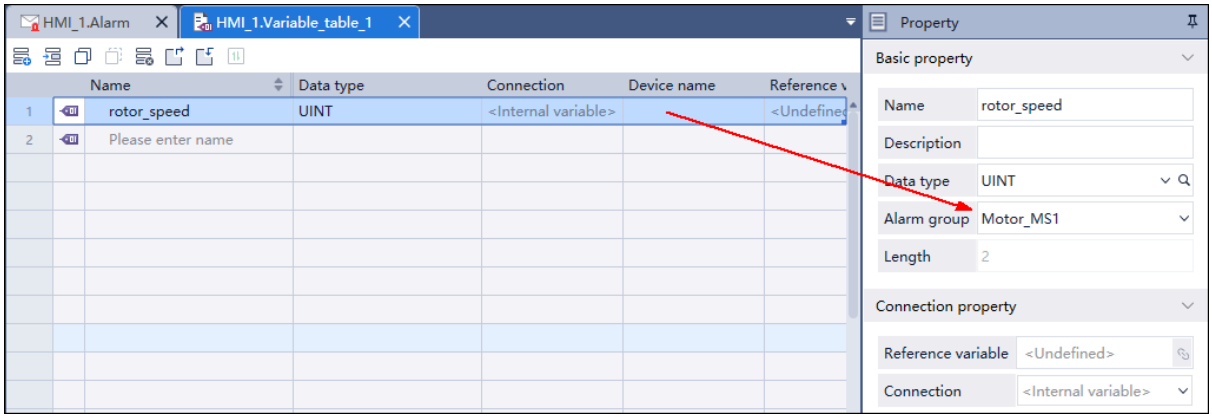


4. Configure the discrete alarm
   In the monitoring area, select "Alarm > Discrete alarm" and set the discrete alarm.



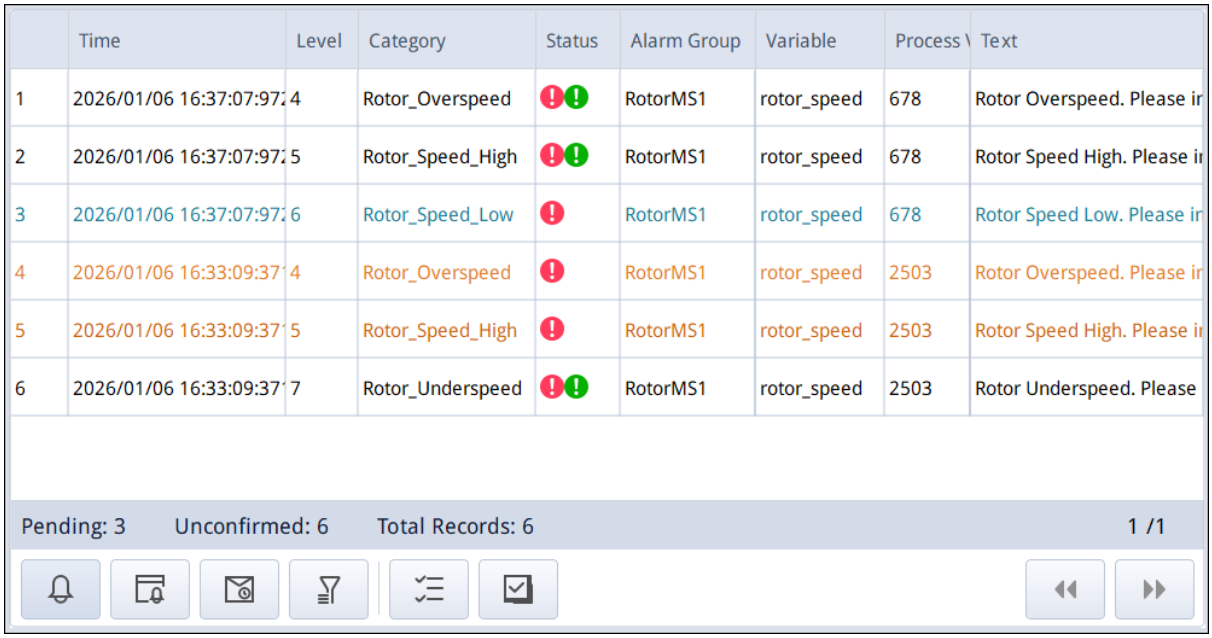
5. Configure the alarm group

   In the right property box of the variable "rotor_speed", set the alarm group to "Motor MS1".



6. Use the alarm control

   Create a new screen and configure the alarm view control during configuration. In runtime, the alarm view control will display abnormal information of variables. Simulate the abnormal condition of the motor speed by modifying the value of the variable "rotor_speed". The figure below shows the process by which the alarm control detects a variable's abnormal state in real time.



13.6 Recipe Usage

This chapter introduces the basic concepts of the recipe function and the offline configuration function.

- 13.6.1 Function Overview
- 13.6.2 Adding a Recipe
- 13.6.3 Modifying a Recipe
- 13.6.4 Deleting a Recipe
- 13.6.5 Copying and Pasting a Recipe
- 13.6.6 Importing and Exporting a Recipe

13.6.1 Function Overview

In the manufacturing field, a recipe refers to the proportional relationship between different raw materials used for production. Essentially, a recipe is a set of parameter values that determine the variables in the production process.

For example, a beverage plant will have a basic recipe for beverage production. This recipe lists all the necessary raw materials and their amounts, such as water, white sugar, fruit juice concentrate, edible flavors, and honey. It will also list a series of optional elements and their amounts, such as high fructose corn syrup and vitamins. These optional elements can be flexibly added to the base recipe according to market demand and product positioning, enabling the production of a variety of different flavored and functional beverages.

The table below illustrates the raw material types and their amounts for different beverages. The data is for example only and is intended to help users understand the meaning and relationships of the terms used in the recipe function. The data has not been scientifically validated and shall not be regarded as the actual content of real product elements.

Table13-6-1-1 Beverage recipe example

Element	Reconstituted Juice	Fruit Drink	Honey Water	Energy Drink
Water	300 g	350 g	360 g	340 g
White sugar	0	80 g	40 g	60 g
Fruit juice concentrate	200 g	50 g	0	30 g
Honey	0	10 g	50 g	0
Edible flavor	0	12 mg	15 mg	0
High fructose corn syrup	0	8 g	40 g	50 g
Vitamin	0	30 mg	0	80 mg

The iFA software provides recipe-related editors and visual controls, enabling recipe management and object configuration within the editor. The following explains the terms involved in the recipe configuration:
Recipe: A collection of elements and their amounts for similar products (beverages). These products share the same elements but differ in amounts. Individual products (beverages) are distinguished by their names. The table above is an overall recipe.

Element: The raw materials contained in the product, such as the first column of the table above (e.g., water, white sugar, etc.).

Data record: Data members in a recipe. A recipe contains multiple data records. These data records have the same structure but different values. In the table above, column 2/3/4 each represents a data record.

13.6.2 Adding a Recipe

Addition of the recipe:

1. Double-click "Recipe" in the project tree to open the recipe editing interface.

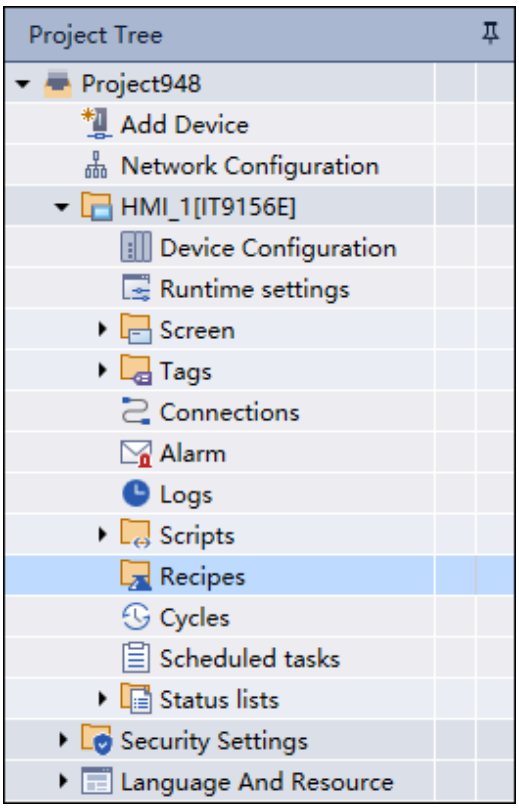


Figure 13-6-2-1 Recipe node

2. Add a recipe.

   Right-click the upper half of the editing interface and select "Add row". You can also enter text in an empty cell to add a recipe.

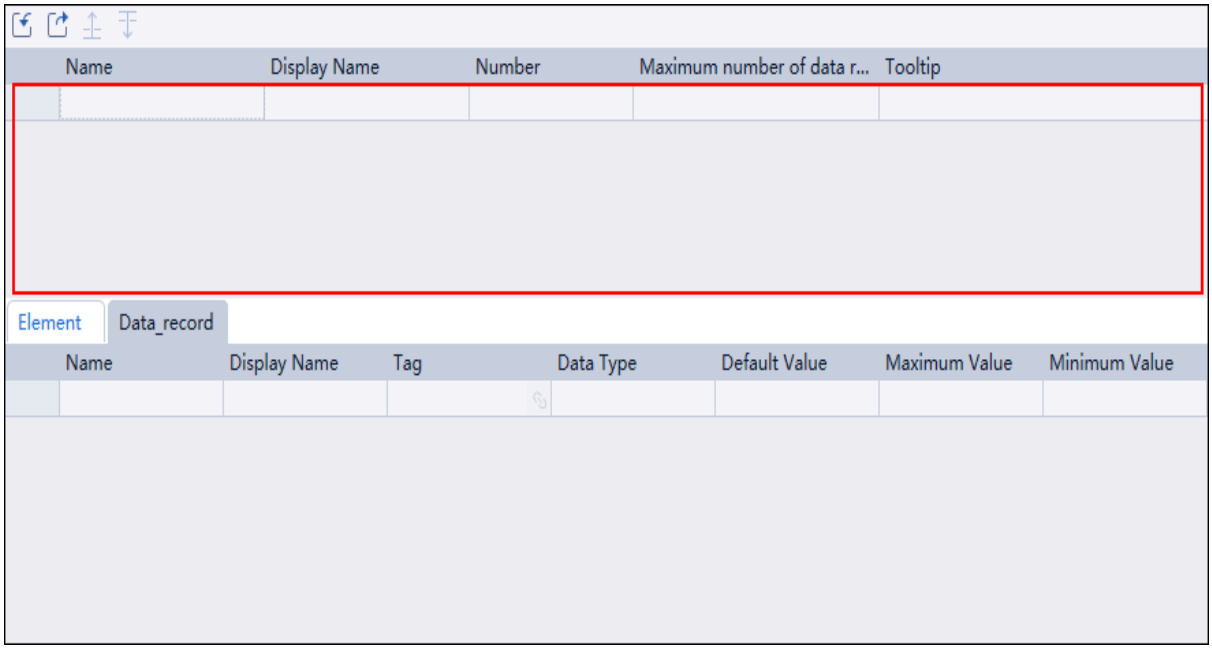


Figure 13-6-2-2 Recipe addition area

A device supports 100 recipes. The following figure shows the result of adding two recipes.

	Name	Display Name	Number	Maximum number of data records	Tooltip
1	Formula_1	Formula_1	1	1000	
2	Formula_2	Formula_2	2	1000	

Figure 13-6-2-3 Recipe addition result

Property	Description
Name	The identifier of a recipe in the software. The name supports Chinese and English characters, numbers, underscores (_), and middle hyphens (-), with a maximum length of 64 characters. Default name is "recipe_n", where n increments starting from 1. Duplicate names are not allowed.
Display name	The name displayed in the recipe control. It supports Chinese and English characters, numbers, underscores (_), and middle hyphens (-), with a maximum length of 64 characters.
ID	Range: 1 to 100.
Max. number of data records	Number of data records in the recipe, with a range from 0 to 1000.
Info text	The description of the recipe. Supports a maximum of 256 characters.

3. Add elements.

   Set the elements in the recipe. A recipe can contain up to 1000 elements.

   On the "Element" tab, select "Add row" from the right-click menu or enter text in an empty cell to add an element.

Element | Data_record

	Name	Display Name	Tag	Data Type	Default Value	Maximum Value	Minimum Value	Decimal Places	Tooltip
1	Water	Water	Water	INT	0	900	0	0	
2	Lemon	Lemon	Lemon	INT	0	50	0	0	
3	Tea	Tea	Tea	INT	0	50	0	0	
4	Syrup	Syrup	Syrup	INT	0	200	0	2	
5	PassionFruit	PassionFruit	PassionFruit	INT	0	50	0	1	

Figure 13-6-2-4 Element addition result

Property	Description
Name	The identifier of an element in the software. The name supports Chinese and English characters, numbers, underscores (_), and middle hyphens (-), with a maximum length of 64 characters. Default name is "Element_n", where n increments starting from 1. Duplicate names are not allowed.
Display name	The name displayed in the recipe control. It supports Chinese and English characters, numbers, underscores (_), and middle hyphens (-), with a maximum length of 64 characters.
Associated variable name	The variable associated with the element. Supports single variables and array variables, with data types BOOL, INT, UINT, REAL, LREAL, STRING, and WSTRING.
Data type	The data type of the associated variable.
Default value	When the values of the elements are set to default in the data record, the recipe control will display this value when it is online.
Maximum value	Range limits of the associated variables.
Minimum value	
Decimal points	Applicable to variable of types INT, UINT, DINT, UDINT, LINT, ULINT, SINT, USINT, REAL, and LREAL. Where: For REAL and LREAL types: Indicates the number of decimal places retained, enhancing numeric precision. For other variable types: Values of data records are displayed according to the number of decimal places. For example, if the configured value of a data record is 400 and the number of decimal places is 1, the data record will be displayed as 40.0 when it is online.
Info text	The description of the element. Supports a maximum of 256 characters.

4. Add data records.
   A recipe can contain a maximum of 1000 data records.
   On the "Data record" tab, select "Add row" from the right-click menu or enter text in an empty cell to add a data record.

Element | Data_record

	Name	Display Name	Number	Water	Lemon	Tea	Syrup	PassionFruit	Tooltip
1	Beverage1	Regular Tea	1	300	0	25	0	0	
2	Beverage2	Passion Fruit Tea	2	300	40	20	50	5	
3	Beverage3	Lemon Tea	3	300	10	28	55	40	
4	Beverage4	Sweet Soup	4	300	0	0	80	0	

Figure 13-6-2-5 Data record addition result

Property	Description
Name	The identifier of a data record in the software. The name supports Chinese and English characters, numbers, underscores (_), and middle hyphens (-), with a maximum length of 64 characters. Default name "Data_record_n", where n increments starting from 1. Duplicate names are not allowed.
Display name	The name displayed in the recipe control. It supports Chinese and English characters, numbers, underscores (_), and middle hyphens (-), with a maximum length of 64 characters.
ID	Range: 1 to 1000.
Element value	Values of each element in different data records.
Info text	The description of the data record. Supports a maximum of 256 characters.

After adding data records, a complete recipe is successfully added.

13.6.3 Modifying a Recipe

After a recipe is created, it may change according to on-site control requirements. Typical changes include:

Adding a recipe/element/data record

Use the "Insert row" or "Add row" command to add a recipe/element/data record.

Insert row: Insert new content above the selected row.

Add row: Add new content to the end of the current table.

Modifying recipe/data record number

During the online operation of the recipe (controller), users can quickly access relevant information through the recipe and data recording numbers.

The numbers support user customization. Duplicates are not allowed within the associated table.

Double-click the number cell, enter the new number, and press the **Enter** key.

Reordering elements

Elements are initially sorted by the order in which they were added. Manual sorting is supported:

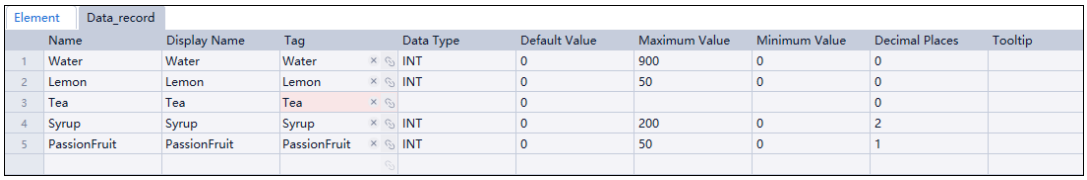
Move up: Click  to move the selected element up by one row.

Move down: Click  to move the selected element down by one row.

Modifying the element-associated variable

If the variable name or data type of an associated variable is changed in the variable table, the element will automatically synchronize the update, requiring no manual modification.

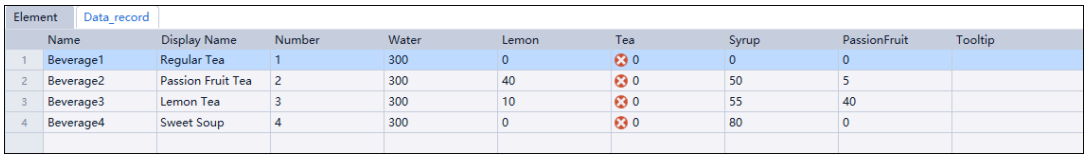
If the changed data type no longer meets the element requirements, the variable will be marked as abnormal on both the "Element" and "Data record" tabs. For example, the data type of the variable "**tea**" associated with the element **green tea** does not meet the constraints, so "**tea**" is marked as abnormal on both the "Element" and "Data record" tabs, as shown below.



Element | Data_record

	Name	Display Name	Tag	Data Type	Default Value	Maximum Value	Minimum Value	Decimal Places	Tooltip
1	Water	Water	Water	INT	0	900	0	0	
2	Lemon	Lemon	Lemon	INT	0	50	0	0	
3	Tea	Tea	Tea		0			0	
4	Syrup	Syrup	Syrup	INT	0	200	0	2	
5	PassionFruit	PassionFruit	PassionFruit	INT	0	50	0	1	

Figure 13-6-3-1 Variable abnormal mark on "Element" tab



Element | Data_record

	Name	Display Name	Number	Water	Lemon	Tea	Syrup	PassionFruit	Tooltip
1	Beverage1	Regular Tea	1	300	0	0	0	0	
2	Beverage2	Passion Fruit Tea	2	300	40	0	50	5	
3	Beverage3	Lemon Tea	3	300	10	0	55	40	
4	Beverage4	Sweet Soup	4	300	0	0	80	0	

Figure 13-6-3-2 Variable abnormal mark on "Data record" tab

13.6.4 Deleting a Recipe

Deleting a recipe will also delete its related elements and data records. Recipes support single deletion and batch deletion (using the Ctrl/Shift keys).

Delete a recipe through the right-click menu or the **Delete** key.

13.6.5 Copying and Pasting a Recipe

Recipes support copy and paste operations within a device.

Both single and batch copy are supported. The pasted name defaults to the original name with a "_n" suffix.

13.6.6 Importing and Exporting a Recipe

Importing recipes

Import recipes to enable cross-device recipe reuse and reduce configuration time.

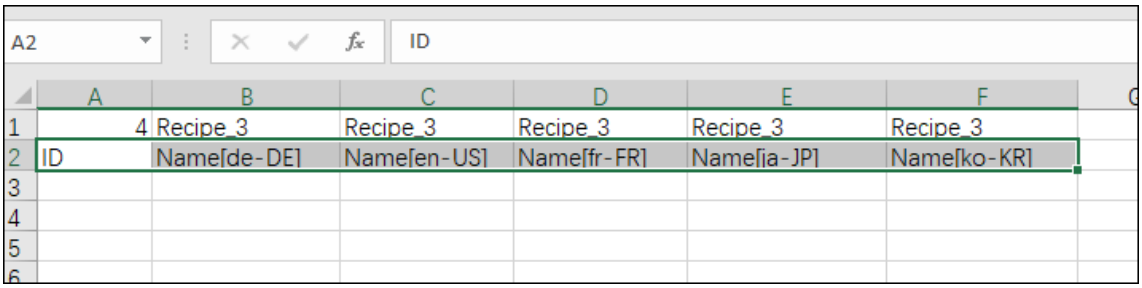
Recipe data is imported into iFA software as in .csv format.

- Prepare the file to be imported. Edit the CSV file in a text editor, ensuring the content meets the following three requirements to avoid import failures.



  - When preparing content for import, it is recommended to first export a recipe file and then add new content to it.
  - Recipe export only supports exporting the multilingual content of the current device. To export other languages, enable them in "Runtime setting > Language and font" for the device before exporting.

- The key fields of a recipe must include the Recipe ID and at least one Language ID. These fields cannot be changed or deleted, as indicated by the blue text in the figure.
  - When importing a recipe file into a new project, the "Name[Language ID]" field will be imported into the corresponding language:



	A	B	C	D	E	F
1	4	Recipe_3	Recipe_3	Recipe_3	Recipe_3	Recipe_3
2	ID	Name[de-DE]	Name[en-US]	Name[fr-FR]	Name[ja-JP]	Name[ko-KR]
3						
4						
5						
6						

  - When importing a recipe file into an existing project, the "Name" field will be imported into all languages activated in the current project:

	1	Recipe_1				
ID		Name	water	lemon	tea	suger	Passion fruit
	1	Ordinary tea	300	0	25	0	0
	2	Passion fruit tea	300	40	20	50	5
	3	Lemon tea	300	10	28	55	40
	4	tea water	350	0	0	80	0

- The Recipe ID must be valid and unique.
- The order and quantity of element data must exactly match those in the data records.
  As shown in the figure above, the order of the five elements (water, lemon, etc.) must exactly match the order in the data records.

- Import the file.
  Click the import icon and select the file to be imported. Only one recipe can be imported at a time. A result prompt appears upon completion. The following table lists common failure prompts and their causes.

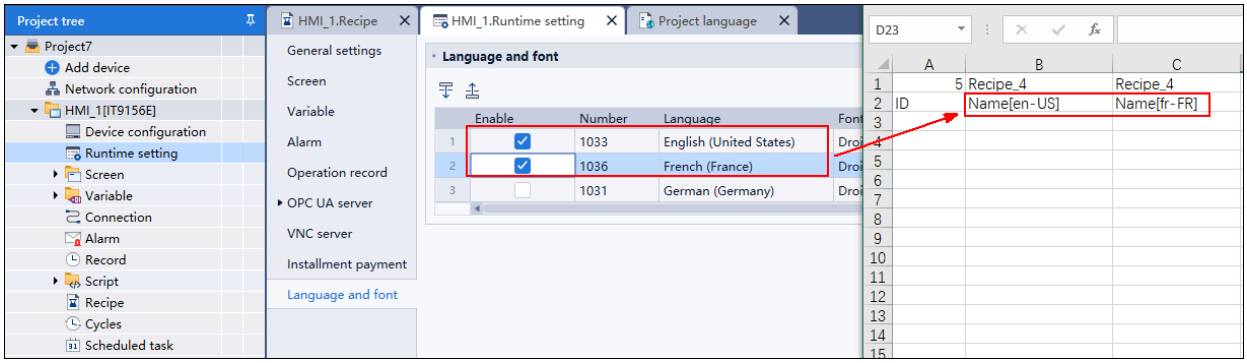
No.	Prompt	Cause
1	Failed to read the file. Please check the modifications made to the import file.	The key fields were modified, deleted, reordered, or the file format was changed.
2	The Recipe ID and Language ID cannot be empty.	The file contains invalid data. See 13.6.2 Adding a Recipe to check the property descriptions.
3	The element data in the data record does not match the data in element management.	The quantity or order of element data in the data record does not match that in element management.

Exporting recipes

Select a recipe and click the export icon to perform the export operation. Only one recipe can be exported at a time.

The exported recipe file (.csv) is named after the recipe and is stored by default in **C:\Users\Administrator\Documents**.

Recipe export only supports exporting the multilingual content of the current device. To export other languages, enable them in "Runtime setting > Language and font" for the device before exporting.



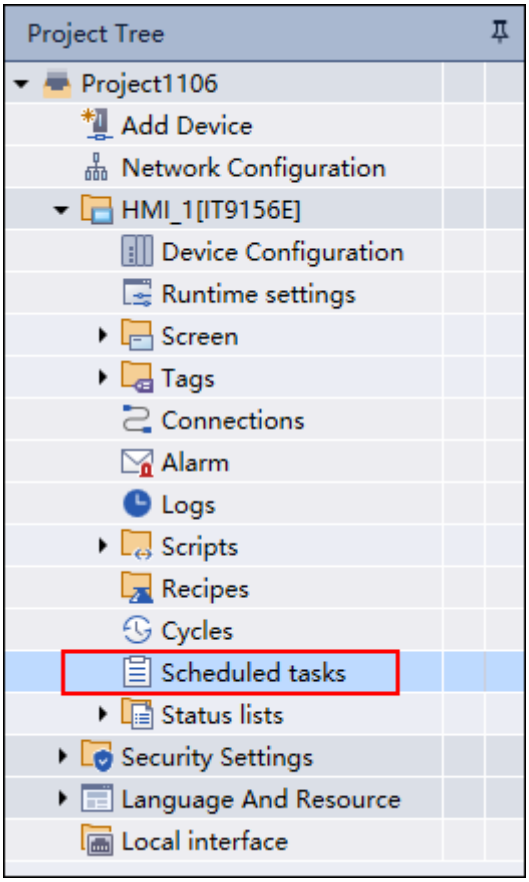
13.7 Scheduled Task

- 13.7.1 Adding a Task
- 13.7.2 Editing a Task
- 13.7.3 Task Properties
- 13.7.4 Event Configuration for Tasks
- 13.7.5 Copying and Pasting a Task
- 13.7.6 Deleting a Task

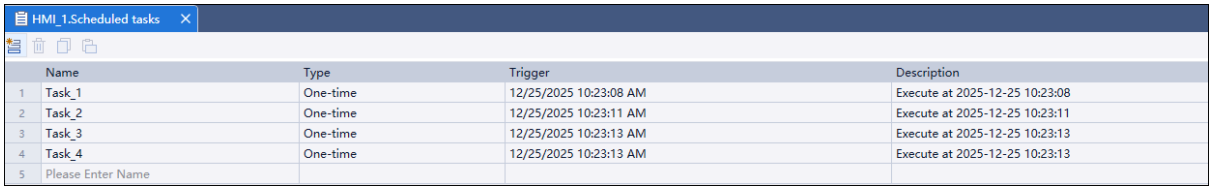
13.7.1 Adding a Task

The visualization platform supports executing tasks based on schedules. The "Scheduled task" window displays all task information in a table format. Different scheduled tasks can use the same configuration parameters. A maximum of 128 tasks can be created. Add a scheduled task as follows:

1. Double-click "Scheduled task" in the project tree to open the "Scheduled task" window.



2. Add a scheduled task in the following three ways:
   - Double-click the "Name" cell in the last blank row to enter the edit mode. Enter the name for the new scheduled task, then click elsewhere or press the Enter key to complete the addition.
   - Click  above the table to automatically add a new scheduled task.
   - Right-click the schedule task table and select "Add row" to automatically add a schedule task.
3. The addition result is shown in the figure below.

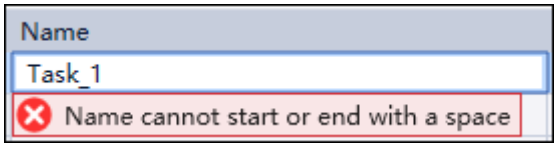


13.7.2 Editing a Task

The scheduled task supports modifications to its name, type, and trigger.

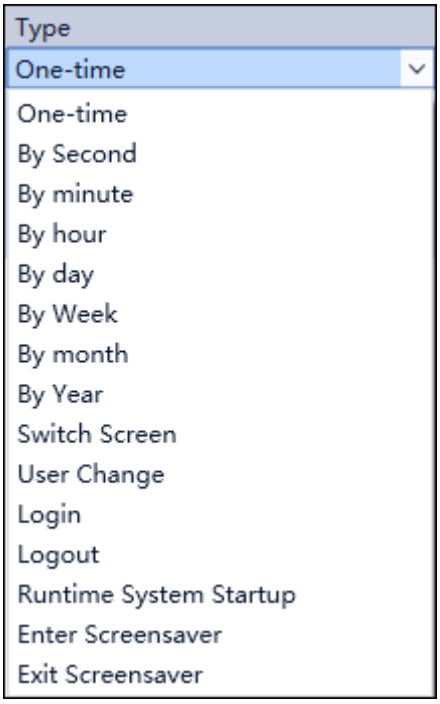
Name modification

Double-click the task name cell to modify to enter the edit mode. Enter the new name, then click elsewhere or press the Enter key. The software automatically checks the validity of the entered content. If the name is invalid, the software displays an error prompt as shown in the figure below, and discards the input.



Type modification

Select the task row to modify, and click  to the right of the cell or double-click the "Type" cell of the task to modify to enter the edit mode. The drop-down list is displayed, as shown below. The current type is selected by default. Select the required type from the drop-down list to complete the modification.



Trigger modification

Select the task row to modify, and click to the right of the cell or double-click the "Type" cell of the task to modify to enter the edit mode. The modification window corresponding to the configured trigger pops up. Enter the new configuration parameters in the window, click elsewhere or press the Enter key.

13.7.3 Task Properties

The basic properties and descriptions of the scheduled tasks are shown in the table below:

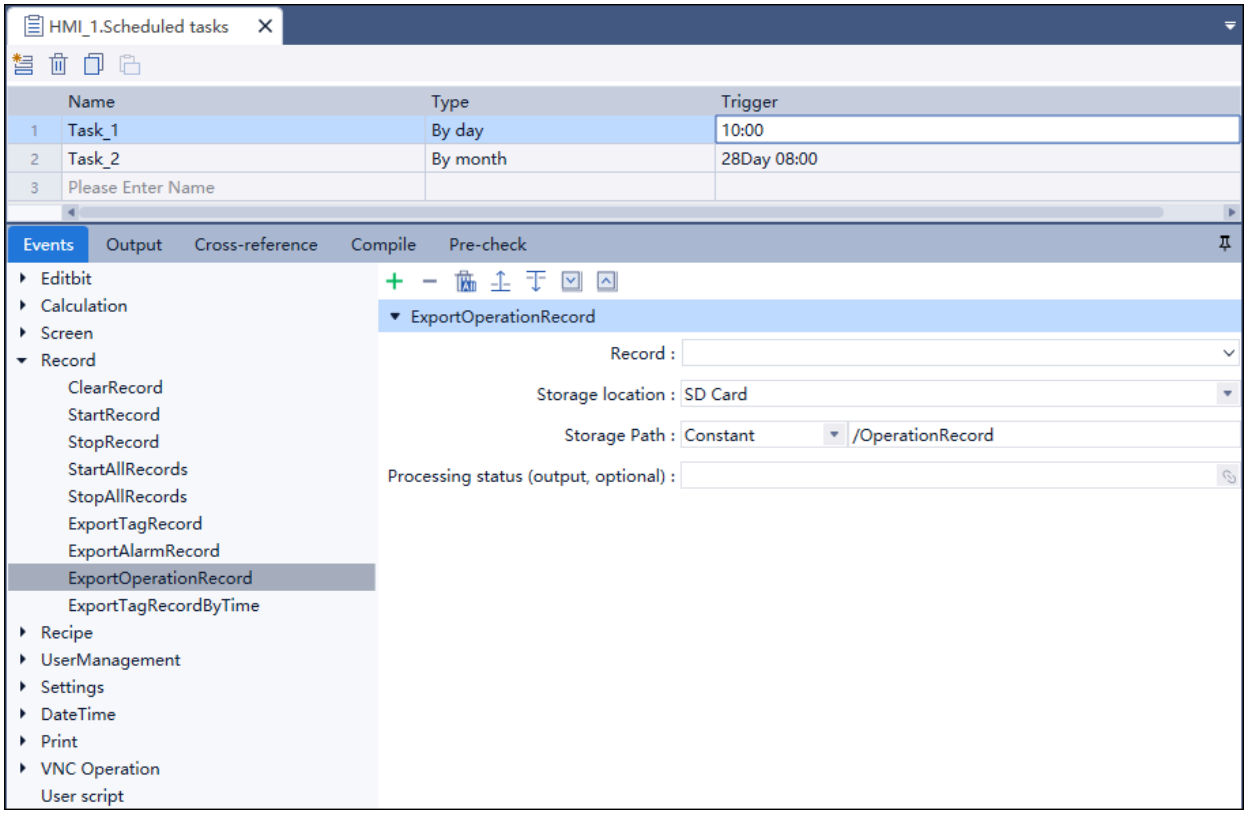
Property	Description
Name	Task name. • Supports letters (case-insensitive), numbers, Chinese characters, and spaces • Supported special characters: ~`!@#$%^&*()-_=+[]{}|\;:,.<>/?! （）、：；“”，。？ • Max length: 128 characters. • Name cannot start or end with a space. The default name is "Task_N", where "N" is an incremental sequence number starting from 1. If there is a gap in the existing name numbers, the smallest available number is used.
Type	Only options within the system's supported range can be selected. Default is "Once".
Flip-flop	Only options within the system's supported range can be selected. Default is the current time.
Description	Description information is automatically generated based on the configured trigger, which does not support manual modification. Default is "Execute at the current time".

The types supported by the visualization platform and their descriptions are shown in the table below. In all descriptions automatically generated by the configuration, the default date and time refer to the current date and time.

Property	Description
Once	Users can configure the date and time for task execution. Example of automatically generated description: Execute on 2024-11-19 15:00:00.
By second	Users can configure the number of seconds, ranging from 1 to 59. Example of automatically generated description: Execute every 1 second.
By minute	Users can configure the number of minutes, ranging from 1 to 59. Example of automatically generated description: Execute every 1 minute.
By hour	Users can configure the minute of the hour, ranging from 0 to 59. Example of automatically generated description: Execute once at minute 5 each hour.
By day	Users can configure the time. Example of automatically generated description: Execute at 15:00 daily.
By week	Users can configure the day of the week and time. Example of automatically generated description: Execute at 15:00 every Tuesday.
By month	Users can configure the date and time. Example of automatically generated description: Execute on day 19 at 15:00 each month.
By year	Users can configure the date and time. Example of automatically generated description: Execute at 15:00 on 11-19 each year.
Switch screen	Triggered once when switching screens.
User change	Triggered once when the user changes.
Login	Triggered once when the user logs in.
Logout	Triggered once when the user logs out.
Runtime system startup	Trigger once when the graphical runtime system starts.
Enter screensaver	Triggered once when the touchscreen enters screensaver mode.
Exit screensaver	Triggered once when the touchscreen exits screensaver mode.

13.7.4 Event Configuration for Tasks

Scheduled task supports event configuration. A scheduled task can be configured with up to 32 functions.



The actions of scheduled task execution events will be recorded in the operation log. The results can be viewed through the 13.15.6 Online Operation Record Operations control.

13.7.5 Copying and Pasting a Task

The scheduled task supports single or batch copy/paste. For batch selection, use the Ctrl or Shift key.

Select the scheduled task row to copy, and click or select "Copy" from the right-click menu to copy the selected task. The copied scheduled task can be pasted by clicking above the table, or by selecting "Copy" from the right-click menu.

13.7.6 Deleting a Task

Scheduled tasks can be deleted individually or in batches. For batch deletion, use the Ctrl or Shift key. Select the scheduled task row to delete, and click or select "Delete" from the right-click menu to delete the selected task.

13.8 Operation Record Usage

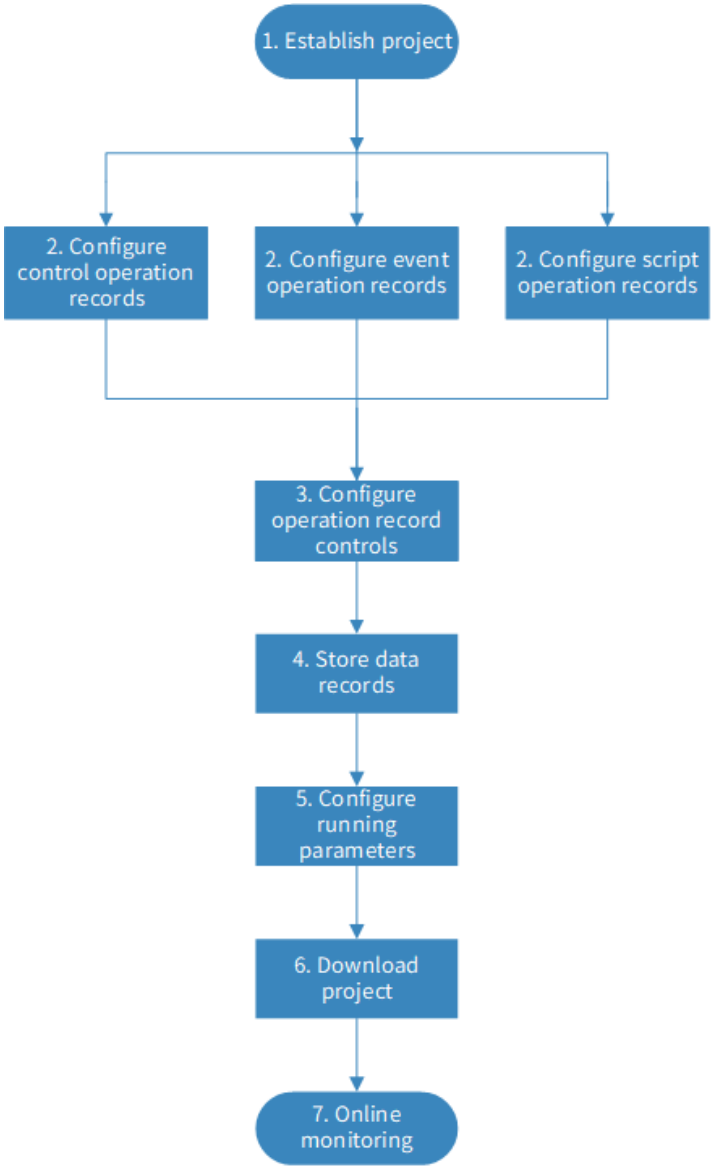
- 13.8.1 Operation Record Configuration Process
- 13.8.2 Operation Records for Stored Projects
- 13.8.3 Operation Record Runtime Configuration

13.8.1 Operation Record Configuration Process

The operation record function displays each step of the user's operations in real time. When an exception occurs, these records can help analyze whether the operation process conforms to the established workflow, and further evaluate and improve the issue.

In the project, changes to variables, screens, controls, events, and other content will generate corresponding operation records. These records are displayed through dedicated controls for user access. Since the configuration process of operation records involves many aspects, see the following flowchart to understand the configuration sequence.

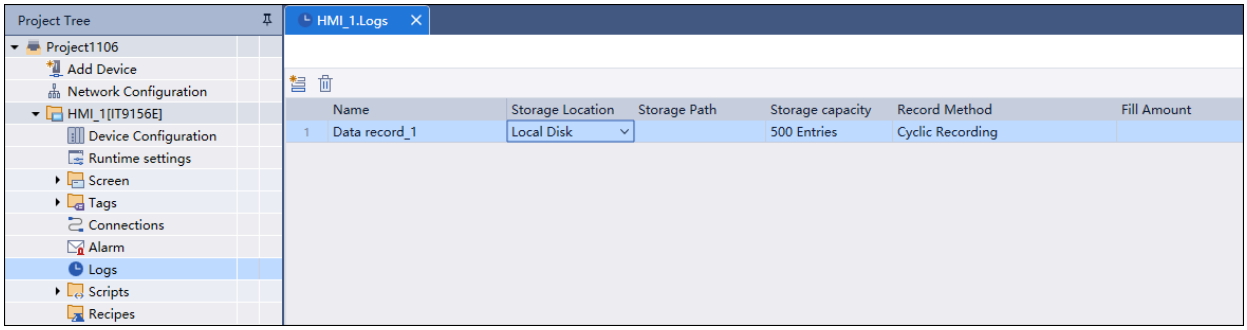
Table13-8-1-1 Operation record configuration process

Key Activity	Description
	1. Add user variables, screens, and other information in the project. 2. Set the storage method and path for log information; see 13.8.2 Operation Records for Stored Projects for details. 3. Configure controls, events, or scripts to record runtime change logs. The configuration scope depends on project requirements; one or more of the following can be configured. • The following controls can be configured with the "Record" property: 13.3.5.2.1 Numeric I/O Field, 13.3.5.2.2 Character I/O Field, 13.3.5.2.4 Button, 13.3.5.2.6 Text Button, 13.3.5.2.5 Bit Button, 13.3.5.2.13 Radio Button, 13.3.5.2.10 Combo Box, 13.3.5.2.11 List Box, 13.3.5.2.14 Checkbox, and 13.3.5.2.12 Text Switch. • Specific system functions in events contain the "Operation record" property. These system functions include: 19.3.1 Increasing Variable Value, 19.3.2 Decreasing Variable Value, 19.3.3 Setting Variable Value, 19.2 Inversion, 19.2 Inverting Bit in Variable, 19.2 Setting, 19.2 Setting Bit in Variable, 19.2 Resetting, and 19.2 Resetting Bit in Variable. • User scripts called in events can be configured with the "Operation record" property.

Key Activity	Description
	4. Add an operation record control on the screen to display the recorded content. For configuration details, see 13.3.5.4.8 Operation Records.
	5. Set the maximum number of logs that the operation record control can display at runtime. See 13.8.3 Operation Record Runtime Configuration for details.
	6. Download the configuration content to the touchscreen to make it effective. See 13.14 Project Download for details.
	7. View the operation logs recorded at runtime. See 13.15.6 Online Operation Record Operations for details.

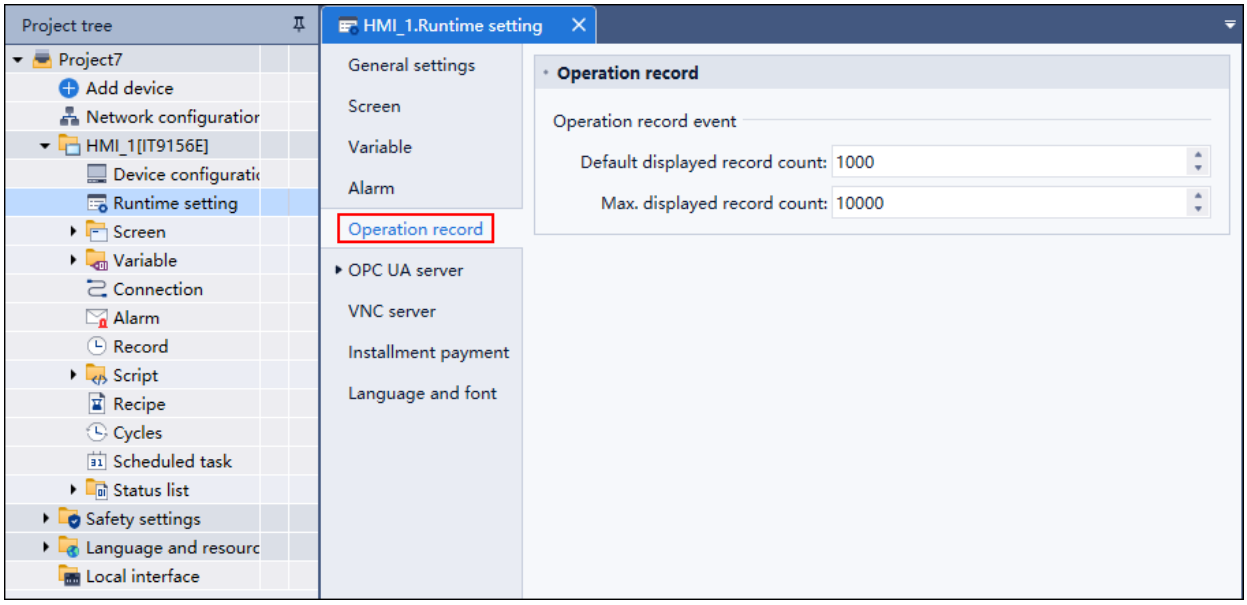
13.8.2 Operation Records for Stored Projects

Each project is automatically configured with an operation record upon creation. One project is configured with one operation record. Adding or deleting records is not supported.



The property items of an operation record are the same as those of a data record. For details, see 13.2.5.6.1 Setting Recording Rules for Data.

13.8.3 Operation Record Runtime Configuration



- Default displayed record count: The number of records displayed in the "Operation record" view with no filter. Range: 100 to 10,000.
- Max. displayed record count: The maximum number of records displayed after filtering according to specified conditions. Range: 10,000 to 1,000,000.

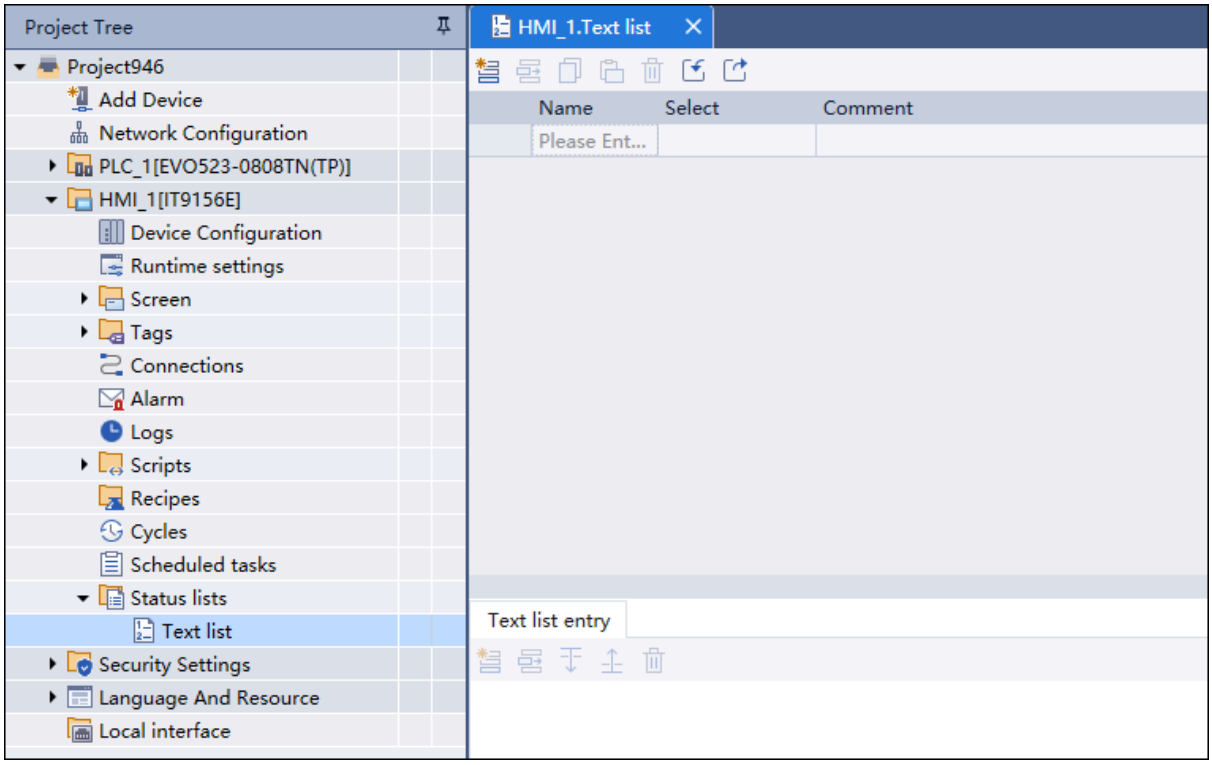
13.9 Text List Usage

- 13.9.1 Adding a Text List
- 13.9.2 Modifying a Text List
- 13.9.3 Deleting a Text List
- 13.9.4 Copying/Pasting a Text List
- 13.9.5 Exporting a Text List
- 13.9.6 Importing a Text List

13.9.1 Adding a Text List

Text lists are used to store specific text. Associate variables with the 13.3.5.2.9 text list control on the screen to monitor data.

1. Open the "Text list" directory in the project tree.
2. Double-click the "Text list" node to open the text list configuration page.



3. Add a text list row.
   - **Add row**

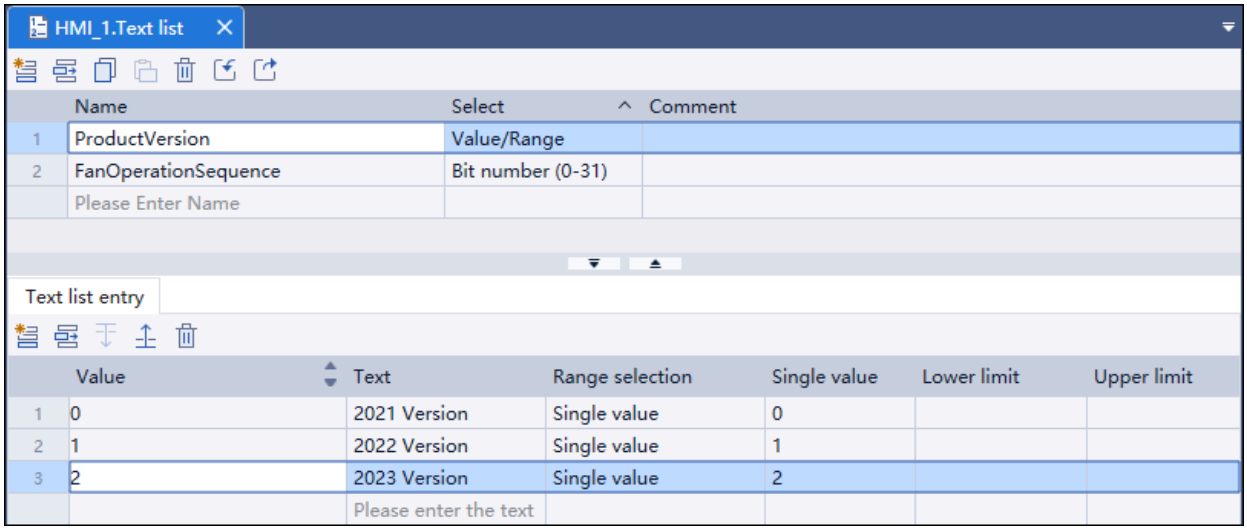
     Click the icon  or select "Add row" in the right-click menu to add a text list entry.
   - **Insert row**

     Select a row, click the icon  or select "Insert row" in the right-click menu to insert a new row above the selected row with default information configured.

A text list supports a maximum of 256 entries.

13.9.2 Modifying a Text List

The modification of a text list consists of two parts: text (rows) and corresponding entries.



Text row

The properties of a text row are determined according to the data types and value ranges of the target variables.

Item	Description
Property	Consists of Chinese characters, English letters, numbers, underscores (_), and hyphens (-), with a maximum of 64 characters.
Select	Value/Range: Supports 2000 entries. Bit number (0 to 31): Supports 32 entries. Bit (0,1): Supports 2 entries.
Comment	Description for the text (row), consisting of 0 to 256 arbitrary characters.

After the name of the text list is modified, if a text list control is associated with this text list, the "Text list" property of the control will automatically update to the new name.

Entry

Select a text (row), and its corresponding entries can be modified on the "Text list entry" tab. The display of the "Text list entry" tab varies according to the settings of the text (row).

- Adding an entry

  Click the icon  or select "Add row" in the right-click menu to add an entry.

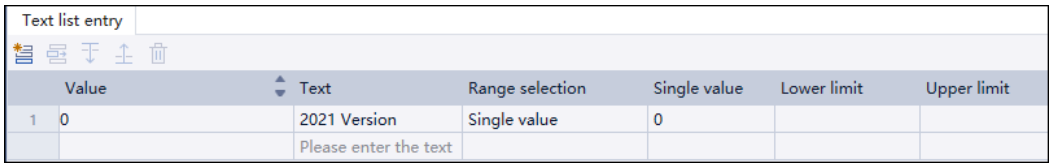
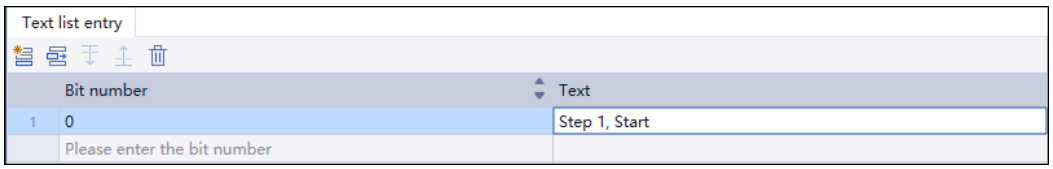


Figure 13-9-2-1 Entry display for "Value/Range" text



Entry display for "Bit number (0 to 31)" text

- Modifying an entry

  The following introduces all properties. Users can selectively view them.

Property Category	Property	Description
Applicable when "Select" = "Value/Range"	Value	The value of the variable. Supports a single numerical value (e.g., 5) or a range (e.g., 2 to 6).
	Range selection	The variable's value is a numerical range, set via upper/lower limits.

Property Category	Property	Description
	Single value	The numerical value corresponding to the entry, within the integer data type range (-2147483648 to 2147483647). Single value: A single numerical value. Upper/Lower limit: Value range.
	Upper/Lower limit	
Applicable when "Select" = "Bit number (0 to 31)"	Bit number	The value of the variable; supports a single numeric value.
Common item	Text	Description of the value or bit number.

- Inserting an entry

  Click the icon  or select "Insert row" in the right-click menu to insert an entry.
  The value or bit number of the new item is automatically filled in based on the current value.

- Reordering entries

  Reorder entries using icons  or . Reordering do not affect the parameter values of the entries.

- Copying/Pasting an entry
  Entries can be copied one by one.
  Pasting entries supports cross-text operations, provided that the "Select" values of the two text rows are the same.

- Deleting an entry

  Select an entry and click the icon  or select "Delete" in the right-click menu to delete the selected entry.

13.9.3 Deleting a Text List

- Select the row containing the single text list, and select the "Delete" icon from the toolbar or select "Delete" from the right-click menu to delete the selected text list.
- Select multiple text list rows, and select the "Delete" icon from the toolbar or select "Delete" from the right-click menu to delete the selected text lists.
- After deletion, all data entries within that text list will be deleted.
- When a text list control is associated with a specific text list, after that text list is deleted, the "Text list" property of the control will be displayed in red.

13.9.4 Copying/Pasting a Text List

Text list supports copy and paste operations within the device.

The copy range includes the text list information and the corresponding entries. Entries can be copied independently.

Before batch copying, select objects using the **Ctrl** or **Shift** keys, then perform the copy operation using **Ctrl + C** or the right-click menu.

Based on the copied content, perform the paste operation at the target location using **Ctrl + V** or the right-click menu. During pasting, duplicate name will be checked and handled automatically.

13.9.5 Exporting a Text List

Text list can be exported.

Method

You can export text information and related entries from the text list one by one using the  icon. Batch export operations are not supported.

Exported file

The **.xlsx** exported file is named after the text list is saved in the C:\Users\Username\Documents directory by default.
The exported content and structure correspond exactly to those on the software interface.

Multilingual support

When exporting, the system automatically includes translations for the corresponding languages based on the languages enabled in the current project.
For example, if the project has enabled German (Switzerland), English (United States), and Chinese (Simplified China), the exported information will by default include translations in these three languages.



Ensure that complete translations for the relevant entries have been added to the project before exporting. If the translation for a certain language of an entry is empty, the corresponding position in the exported file will also be empty.

13.9.6 Importing a Text List

Modify the exported **.xlsx** file and import it to update or create text list entries.

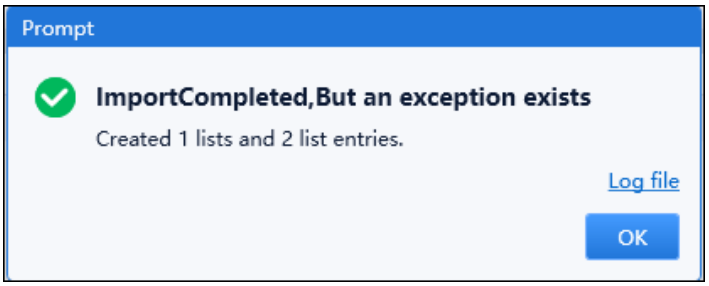
File requirements

- Consistent naming: The file name must exactly match the name of the target text list.
- Valid format: The file structure must match the exported template. The values entered for "Select" fields must be correct, and the number of entries must be within the valid range.
  For example, if "Select" is set to bit numbers (0,1), but the entries contain big numbers other than 0 and 1, it will be considered invalid.

Method

Select an empty row in the text list, then click and select the file to create or update the corresponding entry information in the text list.

Entry information and relevant details regarding import failures will be recorded in the import log.



Multilingual processing

- The system only imports language columns from the file that match the enabled languages for the current project.
- Language columns that do not match will be ignored, and relevant details will be recorded in the import log.

13.10 System Function and Script Usage

- 13.10.1 System Functions
- 13.10.2 User Script

13.10.1 System Functions

Predefined system functions support routine configuration tasks. Users can use these system functions to configure custom functions for various tasks. The main system functions include:

- 19.2 Bit Operations: Performs operations on variable bits.
- 19.3 Calculation: Performs basic calculations on variables.
- 19.4 Screen: Enables switching, displaying, or hiding user screens.
- 19.5 Alarm: Performs alarm-related tasks.
- 19.6 Record: Performs tasks related to data recording and alarm recording.
- 19.7 Recipe: Performs recipe-related tasks.
- 19.8 User Management: Performs permission management for project content.
- 19.9 Configuration: Configures application parameters related to the HMI device, such as backlight duration and screen light on/off.
- 19.10 Date and Time: Sets and reads the HMI device time.
- 19.11.1 Screenshot: Performs the screen capture function during project operation.
- 19.13 Timer: Creates periodic or scheduled tasks.

Users can configure system functions on all graphic objects that are capable of responding to events. System functions are primarily used in screen events, graphic object events, variable events, scheduled tasks, and custom scripts.

13.10.2 User Script

13.10.2.1 Adding a Script

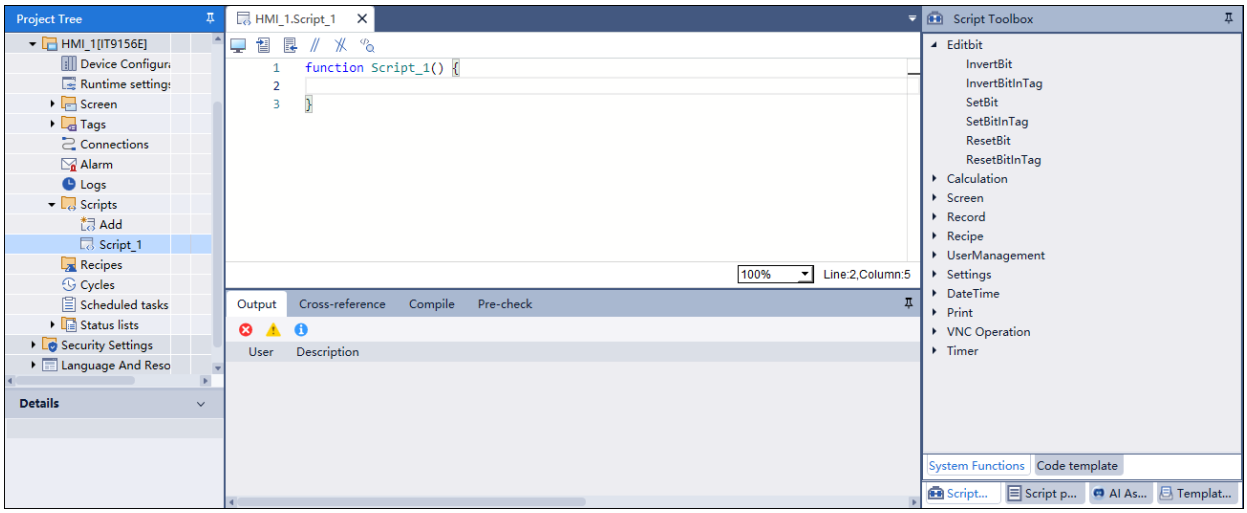
Add a script using the following methods:

- Double-click "Add" under "Script" in the project tree.
- Right-click "Script" in the project tree and select "Add script".

The script name is a combination of English letters, digits, and underscores ("_"). It starts with an English letter, and has a maximum length of 64 characters. Multiple script functions under a single script node cannot have duplicate names.

A maximum of 128 scripts can be created per HMI device.

The initial state of the script after addition is shown in the figure below.



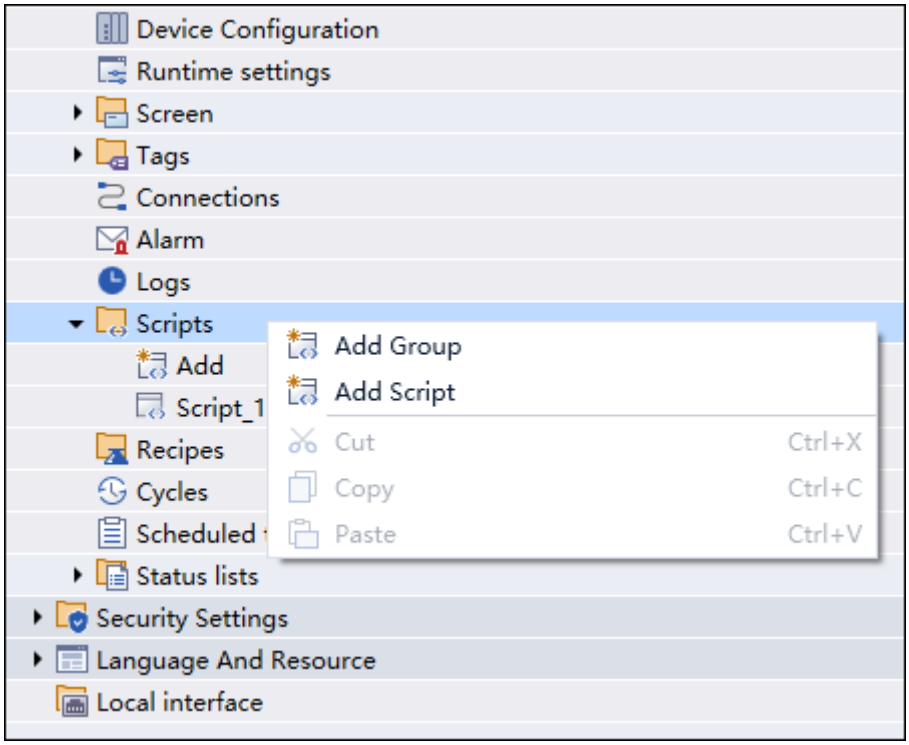
13.10.2.2 Grouping Scripts

User scripts can be grouped and managed based on certain features, such as the controlled object or the implemented control function.

Adding a group

Create a first-level script group under the "Script" node, which supports expanding multiple levels of subgroups.

Right-click "Script" in the project tree and select "Add group" to create a new script group with the node identifier .



Script group names support combinations of Chinese, English, numbers, "_", and "-", with a maximum length of 64 characters. The default name is "Script_Group_n", with the sequence number incrementing automatically.
Script group name must be unique within the same device.

Copying/Pasting

An entire group under the same device supports copy/paste operations, with the content of its scripts unchanged.

Scripts within a group and other scripts outside the group can also be copied and pasted.

Deleting

Delete a group will delete all scripts within it.

13.10.2.3 Modifying a Script

Editing area

Double-click the newly created script, or right-click it and select "Open" to open the script editing area.

```
function Script_1() {
    function calculateAverage(arr) {
        let sum=0;
        for(let i=0;i< arr.length;i++){
            sum +=  arr[i];
        }
        return sum / arr.length;
    }

    // Define an array
    let numbers = [1, 2, 3, 4, 5];

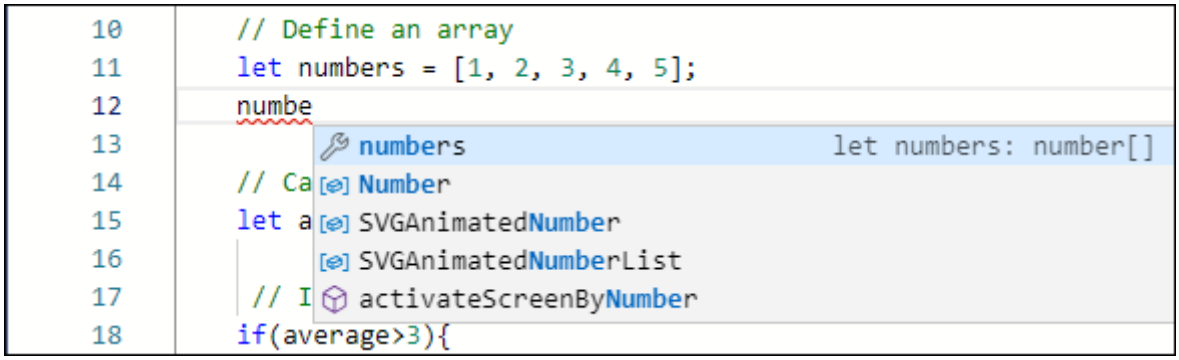
    // Call the function to calculate the average of the array
    let average = calculateAverage(numbers);

     // If the average value is greater than 3, output the corresponding message
    if(average>3){
        console.log("The average value is greater than 3.") ;
    } else {
        console.log("The average value is not greater than3.");
    }
}
```

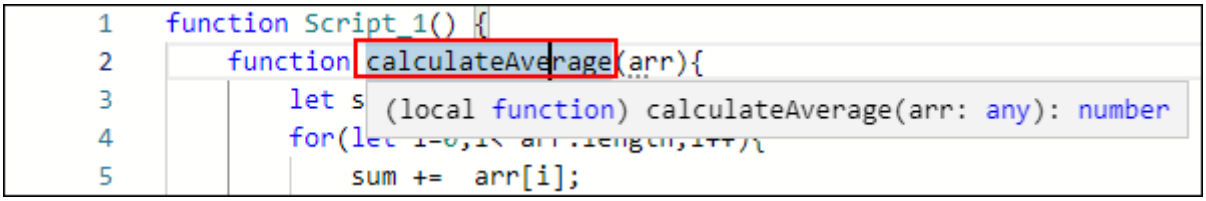
Syntax highlighting distinguishes different parts of the code syntax using different colors, such as keywords, strings, numbers, and comments. These colors are customizable. For details, see Script/text editor.

Intelligent prompt and autocomplete

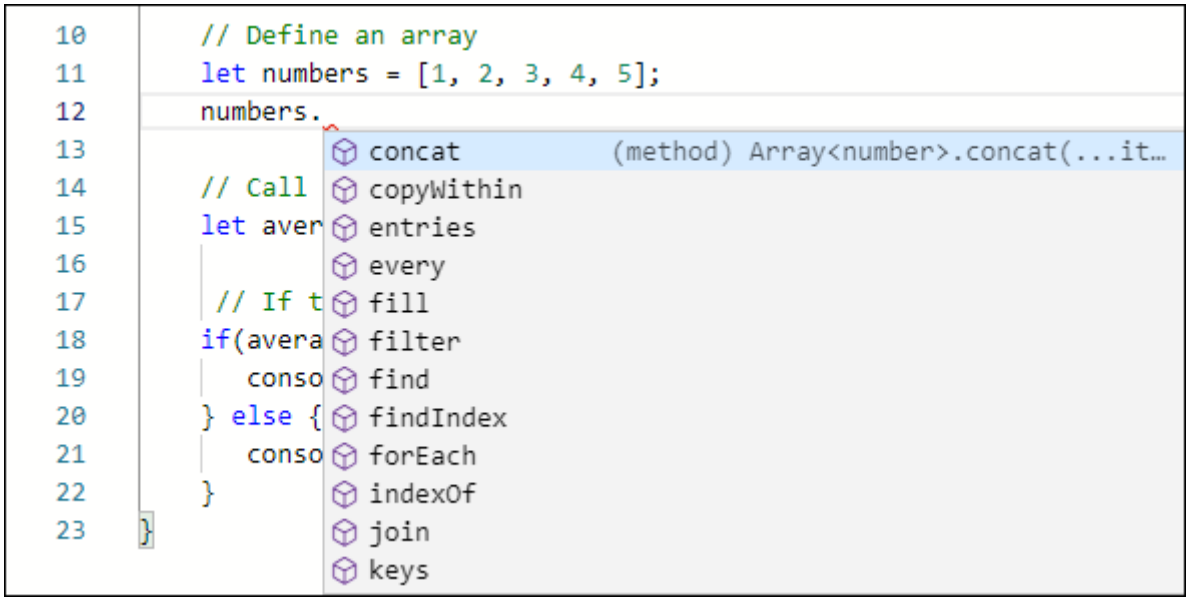
Autocomplete automatically lists all matching system functions in JavaScript object model while you are typing.



When the mouse hovers over a function, a tooltip displays the function signature.

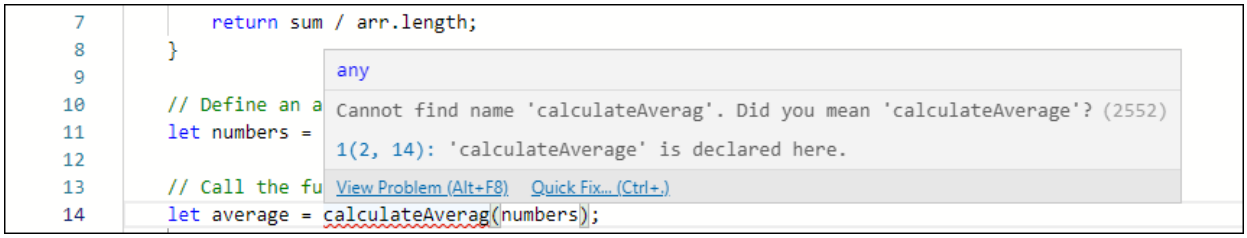


When a user types a "." after a variable, all public properties and methods of that variable are automatically listed, as shown in the figure below.



Syntax and semantic checking

The editor provides real-time syntax and semantic checking.



Toolbar

The toolbar provides commonly used convenient operation commands.

Icon	Command	Description
	System	Enables quick access to system variables. Users can select variable tables for all devices in the current project, including the current device and other devices, as well as built-in system variables. The selected variable is added to the cursor position in the form of smartTags('VariableName'). This has the same effect as function 19.1 Variables.
	Indent +	Increases the leading indentation for one or more selected rows of text.
	Indent -	Decreases the leading indentation for one or more selected lines of text.
	Comment	Adds single-line comments to one or more selected lines of text. Single-line comments are created by adding the comment symbol "//" at the beginning of a text line.
	Uncomment	Removes the comment from a text line.
	Error detection	Checks the syntax and semantics of the script. Error messages are output in the information window of the monitoring area.

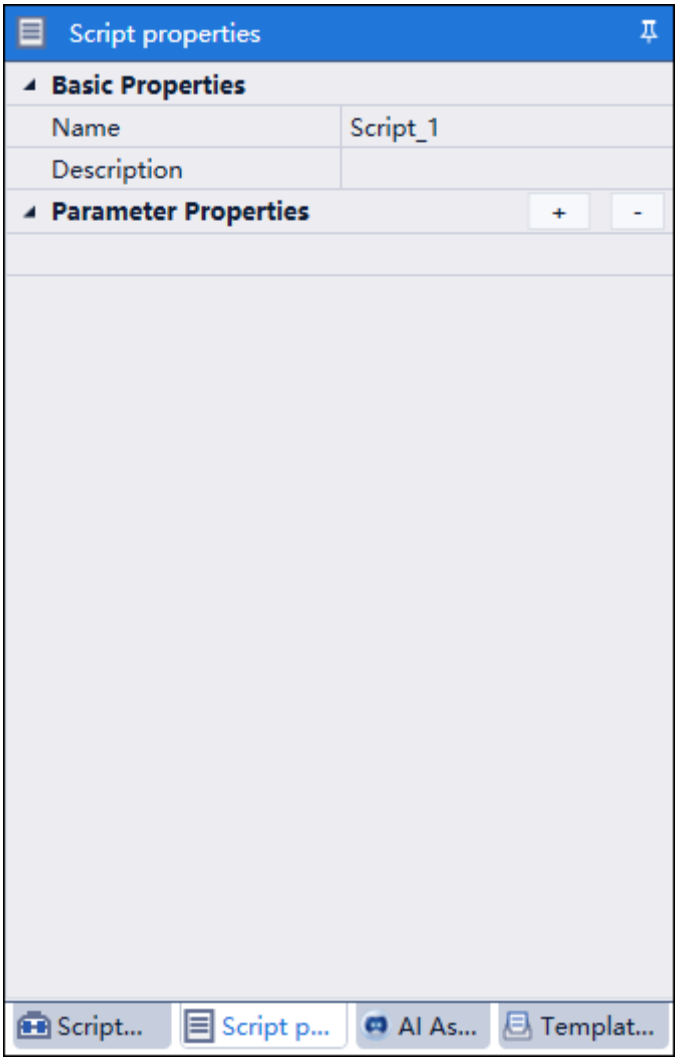
Shortcut

The script editor provides shortcuts almost identical to those in VS Code, greatly improving the efficiency of script developers. Below are some commonly used shortcuts in programming.

Common Command	Shortcut
Formatting	Shift + Alt + F
Error detection	Ctrl + E
Search	Ctrl + F
Replace	Ctrl + F
Indent	Ctrl +]
Outdent	Ctrl + [
Single-line comment	Ctrl + /
Uncomment	Alt + /
Go to line	Ctrl + G
Go to definition	Ctrl + F12
Delete line	Ctrl + Shift + K
Copy line down	Shift + Alt + DownArrow
Move line down	Alt + DownArrow
Insert line below	Ctrl + Enter

13.10.2.4 Script Properties

Script properties include basic properties and parameter properties. They are used to set the function name and parameters of the current script. Script properties can be set through the "Script property" tab of the toolbox on the right, as shown in the figure below.



- Basic property

Name	Description
Name	The name of the script function. The name is a combination of English letters, digits, and underscores ("_"). It starts with an English letter, and has a maximum length of 64 characters.
Description	The description of the script function, which is optional. The description is a combination of Chinese characters, English letters, numbers, "-", and "_". The maximum number of characters is 128.

- Parameter property

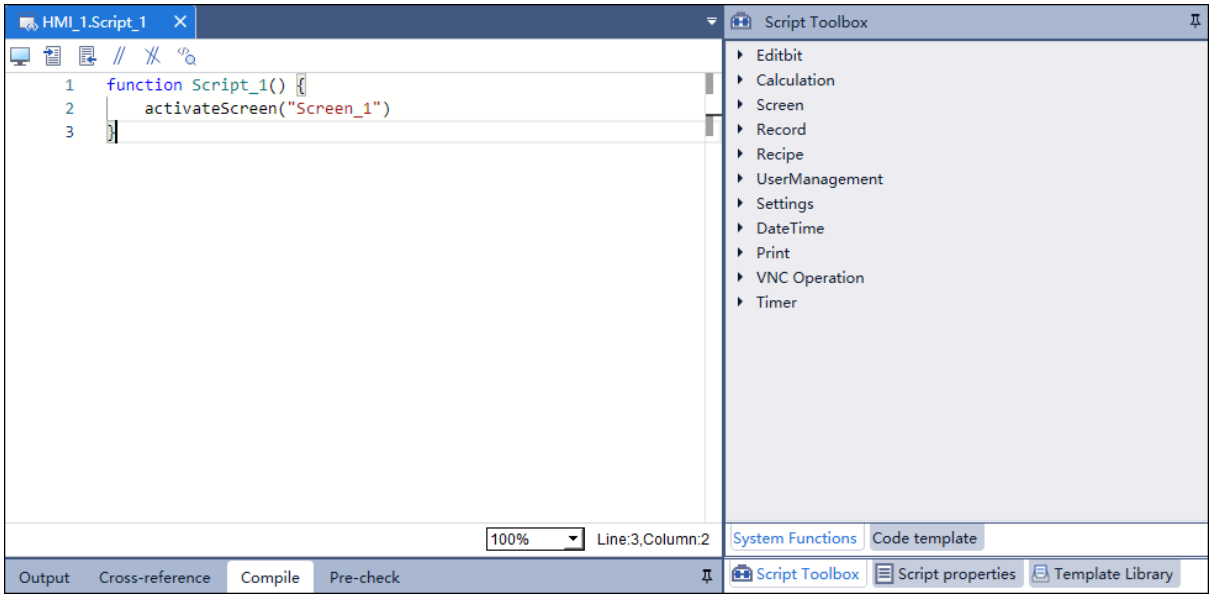
Click "+" and "-" to add or delete a script function parameter.

Property	Description
Parameter	The parameter name of the script function. The name is a combination of English letters, digits, and underscores ("_"). It starts with an English letter, and has a maximum length of 64 characters. Parameter names within the same script function cannot be duplicated.

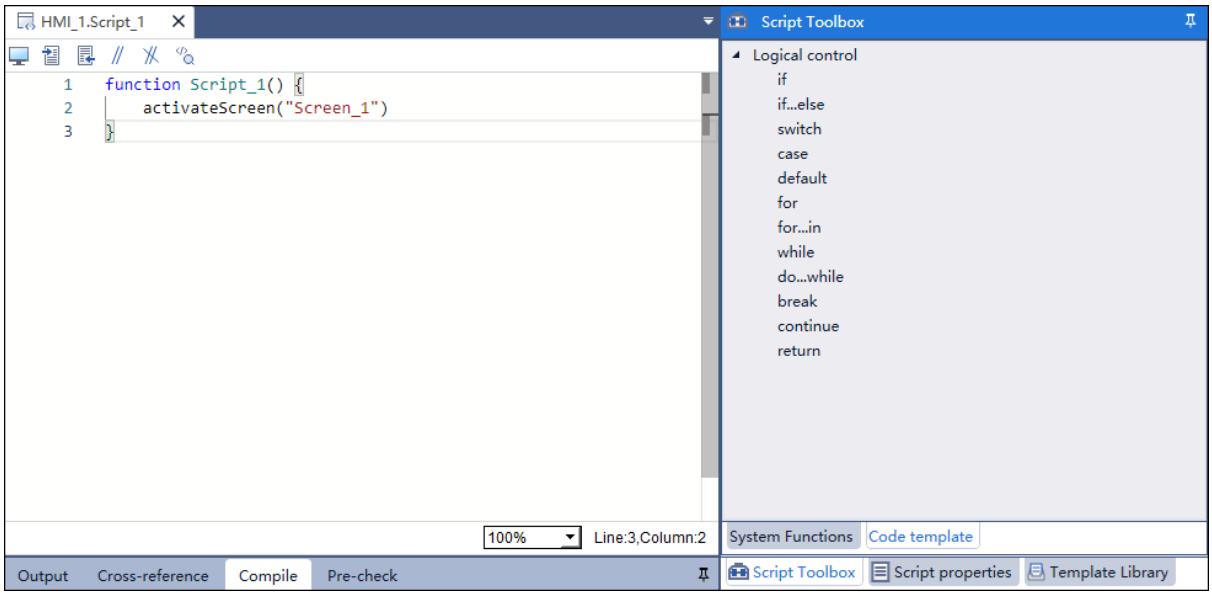
13.10.2.5 Script Toolbox

The script toolbox provides system functions and keywords for use.

- System functions
  The script toolbox contains commonly used functions supported by the script editor, including 12 categories of functions such as bit operations, calculation, and word operations. Detailed usage of these functions is explained in the 19 HMI System Functions chapter.
  Users can double-click a function from the "System function" tab in the script toolbox to insert it at the cursor position.



- Code templates
  The "Code template" tab contains commonly used keywords supported by the script editor, as shown in the figure below. Users can double-click a keyword to insert it at the cursor position.



13.10.2.6 JavaScript Syntax Basics

13.10.2.6.1 Language Overview

The script uses TypeScript as its programming language, with a version of ES6 (ES2015). This allows developers to use modern JavaScript features, including classes, block-scoped variables (let and const), arrow functions, and so on. These features not only enhance code readability and structure but also improve programming flexibility and efficiency. Additionally, the software adopts strict mode specifications to ensure code quality and security. For example, it prohibits undeclared variables and restricts the scope of the eval function to reduce potential risks and errors. At the same time, the software disables alarms for unused function parameters and local variables to help developers keep their code clean and avoid unnecessary distractions. These features and specifications make script development more reliable and efficient.

13.10.2.6.2 Common Operators

The common operators and keywords of JavaScript are managed on the "Code template" tab of the "Script toolbox". The table below lists the common operators, excluding the "Constants" and "Control keywords" nodes.

The sorting of the table remains consistent with the software nodes.



Table13-10-2-6-2-1

Operator		Description	Syntax	Example

Arithmetic	%	Remainder: Returns the remainder when the left operand is divided by the right operand.
		The sign of the result always matches the dividend, and its value is the integer closest to zero.
			a % b	`-13 % 5;  // -3`
				`-9 % 2;  // -1`
				`9 % -2;  // 1`
				`1 % -2;  // 1`
				`Infinity % 2;  // NaN`
				`2 % Infinity;  // 2`

	*	Multiplication: Computes the product of the operands.	a * b	`// Number - Number ->Multiplication`
				`2 * 2; //4`
				`-2 * 2; //4`
				`Infinity * 0; //NaN`
				`Infinity * Infinity //Infinity`

	+	Addition: The operator is overloaded for two different operations: numeric addition and string concatenation.
		Supports:
		• Numeric addition
		• String concatenation
		• Concatenation of a number and a string
		• Concatenation of a string and a boolean value
			a + b	`// Number - Number ->Multiplication`
				`2 * 2; //4`
				`-2 * 2; //4`
				`Infinity * 0; //NaN`
				`Infinity * Infinity //Infinity`
				
				`// Number + Number ->Addition;`
				`1 + 2; //3`
				
				`// Number + string -> Connect`
				`5 + 'foo'; //'5foo'`
				
				`// string + string -> Connect`
				`'foot' + 'ball'; //'football'`
				
				`// string + Boolean -> Connect`
				`'a' + false; //'foofalse'`
				
				`// string + Number -> Connect`
				`'3' + 2; //'32'`

	++	Increment: Increments its operand by one, and returns the value before or after the increment depending on the position of the operator.
		When placed after the operand (e.g., a++), the operand is incremented and the original value is returned.
		When placed before the operand (e.g., ++a), the operand is incremented and the new value is returned.
			a ++
			++ a
				`let x=3;`
				`const y=x++;`
				
				`//x=4`
				`//y=3`
				
				`let x=3;`
				`const y=++x;`
				
				`//x=4`
				`//y=4`

Operator		Description	Syntax	Example

	-	Subtraction: Subtracts the right operand from the left operand and returns the difference.	a - b	`// Number - Number ->Subtraction`
				`5 - 3; // 2`
				`3 - 5; // -2`

	--	Decrement: Decrements its operand by one, and returns the value before or after the decrement depending on the position of the operator.
		When placed after the operand (e.g., a--), the operand is decremented and the original value is returned.
		When placed before the operand (e.g., --a), the operand is decremented and the new value is returned.
			a --
			-- a
				`let x=3;`
				`const y=x--;`
				`//x=2`
				`//y=3`
				`let x=3;`
				`const y=--x;`
				`//x=2`
				`//y=2`

	/	Division: Computes the quotient of the operands, where the left operand is the dividend and the right operand is the divisor.	a / b	`1 / 2; // 0.5`
				`let a = floor (3 / 2); // 1`
				`1.0 / 2.0 ; // 0.5`
				`2.0 / 0; // Infinity`
				`2.0 / 0.0; // Infinity`
				`2.0 / -0.0; // -Infinity`

Assignment	%=	Remainder assignment: Divides the variable by the right operand and assigns the remainder to the variable.	a %= b	`let VAR1 = 23;`
				`VAR1 %= 16; // 7`
				`VAR1 %= 0; // NaN`

	*=	Multiplication assignment: Multiplies the variable by the right operand and assigns the result to the variable.	a *= b	`let VAR1 = 2;`
				`VAR1 *= 6; // 12`
				`VAR1 *= 0; // 0`

	+=	Addition assignment: Adds the right operand value to the variable and assigns the result to the variable. The behavior is determined by the types of the two operands, and may be addition or concatenation.	a += b	`let VAR1 = 5;`
				`VAR1 += 3; // 8`
				`let text = "an";`
				`text += "swer"; // answer`
				`text += true; // antrue`
				`text += 5; // an5`

	-=	Subtraction assignment: Subtracts the right operand value from the variable and assigns the result to the variable.	a -= b	`let VAR1 = 7;`
				`VAR1 -= 2 ; // 5`
				`VAR1 -= -12 ; // -5`

Operator		Description	Syntax	Example

	/=	Division assignment: Divides the variable by the right operand value and assigns the result to the variable.	a /= b	`let VAR1 = 7;`
				`VAR1 /= 2 ;  // 3.5`
				`VAR1 /= 2 ;  // 1.4`
				`VAR1 /= 0;  // Infinity`

	=	Assignment: Assigns a value to a variable.	a = b	`let VAR1 = 7;`
				`let VAR2 = 12;`
				`let VAR3 = 18;`
				
				`VAR1 = 12 ;  //VAR1 is 12`
				`VAR1 = VAR2 = VAR3 ;  //Both VAR1 and VAR2 are 18`

Object	delete	Deletes a property from an object. If the property value is an object and there are no further references to it, the object held by the property will eventually be automatically released.	delete object.property
			delete object[property]
				`//Array`
				`const colors = ["red", "yellow", "green", "black","pink"]`
				`delete colors[3];  // Delete the element with index 3 and return true`
				
				`//Custom Object`
				`let paper ={`
				`size:"A4",`
				`color:"White",`
				`weight:"80g",`
				`};`
				
				`delete paper.size; //Remove the size attribute and return true`

	in	Returns true if the specified property exists in the specified object or its prototype chain.
		The right operand must be an object value.
			prop in object
			prop: String or array index.
			object: Property object.
				`//Array`
				`var colors = ["red", "yellow", "green", "black", "pink"];`
				`2 in colors;  //Returns true`
				`6 in colors;  //Returns flase`
				
				`//Custom Object`
				`let paper = {`
				`size: "A4",`
				`color: "White",`
				`weight: "80g",`
				`};`
				
				`"size" in paper;  //Returns true`
				`delete paper.size;  //Returns true`
				`"size" in paper;  //Returns flase`

	instanceof	Tests whether the prototype property of a constructor appears in the prototype chain of an instance object.	object instanceof constructor
			object: Instance object.
			constructor: Cconstructor function.
				`class Paper {`
				`constructor(size,color){`
				`this.size = size;`
				`this.color = color;`
				
				`}`
				`}`
				`let Notebook = new Paper("A5","blue");`
				`Notebook instanceof Paper;  // true`
				`Notebook instanceof Object;  // true`

Relation	!=	Inequality: Checks whether two operands are not equal and returns a boolean result. Unlike the strict inequality operator (**!==**), it attempts to convert operands of different types before comparison.	a != b	`NaN != NaN;  // true`
				`null != undefined;  // false`
				`true != "1";  // false`
				`7 != "7";  // false`

	!==	Strict inequality: Returns true if the operands are not equal in either value or type.	a !== b	`NaN !== NaN;  // true`
				`null !== undefined;  // true`
				`true !== "1";  // true`
				`7 !== "7";  // true`
				`"an" !== "an";  // false`

Operator	Description	Syntax	Example

<	Less than: Returns true if the left operand is less than the right operand; otherwise returns false.	a < b	`"a" < "b";  // true`
			`true < false;  // false`
			`5 < 3;  // false`
			`NaN < 3;  // false`
			`5 < Infinity;  // true`

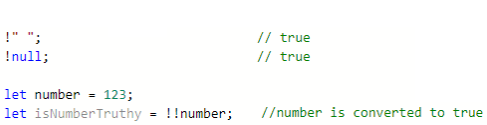
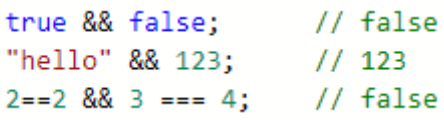
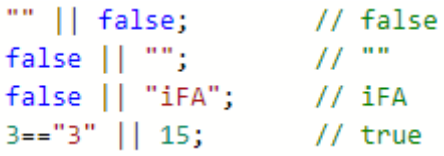
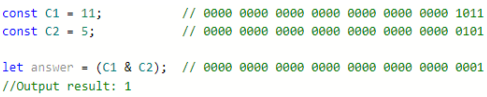
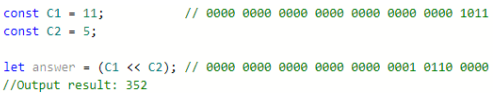
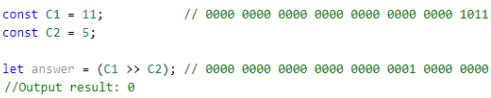
<=	Less than or equal to: Returns true if the left operand is less than or equal to the right operand; otherwise returns false.	a <= b	`"a" <= "b";  // true`
			`true <= false;  // false`
			`5 <= 3;  // false`
			`NaN <= 3;  // false`
			`5 <= Infinity;  // true`

==	Loose equality: Checks whether two operands are equal and returns a boolean result. Unlike the strict equality operator (===), it attempts to convert operands of different types before comparison.	a == b	`NaN == NaN;  //  false`
			`null == undefined;  //  true`
			`true == "1";  //  true`
			`7 == "7";  //  true`

===	Strict equality: Checks whether two operands are equal and returns a boolean result. Operands of different types are always considered unequal.	a === b	`NaN === NaN;  //  false`
			`null === undefined;  //  false`
			`true === "1";  //  false`
			`7 === "7";  //  false`
			`"an" === "an";  //  true`

>	Greater than: Returns true if the left operand is greater than the right operand; otherwise returns false.	a > b	`"a" > "b";  // false`
			`true > false;  // true`
			`5 > 13;  // false`
			`NaN > 3;  // false`
			`3 > NaN;  // false`
			`5 > Infinity;  // false`

>=	Greater than or equal to: Returns true if the left operand is greater than or equal to the right operand; otherwise returns false.	a >= b	`"ab" >= "aa";  // true`
			`"a" >= "3";  // true`
			`true >= 1;  // true`
			`15 >= 13;  // true`
			`NaN >= 3;  // false`
			`3 >= NaN;  // false`
			`5 >= Infinity;  // false`

Operator		Description	Syntax	Example
Logic	!	Logical NOT: Converts true values to false and false values to true, which is often used with boolean values. When used with non-boolean values, returns false if the operand can be converted to true; otherwise returns true.	!expr	
	&&	Logical AND: Returns true if and only if all operands are true; otherwise returns false. Evaluates operands from left to right, returning immediately upon encountering the first false value; if all operands are true, returns the value of the last operand.	expr1 && expr2	
	‖	Logical OR: Returns true if one or more of its operands are true. When used with non-boolean values, returns a non-boolean value. If expr1 can be converted to true, returns expr1; otherwise returns expr2.	expr1 ‖ expr2	
Bit	&	Bitwise AND: The result bit is 1 only if both corresponding bits of the two operands are 1.	a & b	
	<<	Left shift: Shifts the first operand left by the specified number of bits. Bits shifted beyond the left are discarded, and zeros are filled in on the right. Returns a 32-bit integer.	a << b	
	>>	Right shift: Shifts the first operand right by the specified number of bits. Bits shifted beyond the right are discarded, and the leftmost empty bits are filled with the sign bit (the leftmost bit before the shift). Returns a 32-bit integer.	a >> b	

Operator		Description	Syntax	Example

	^	Bitwise XOR: The result bit is 1 if and only if exactly one of the corresponding bits of the two operands is 1.	a ^ b	`const C1 = 11; // 0000 0000 0000 0000 0000 0000 0000 1011`
				`const C2 = 5; // 0000 0000 0000 0000 0000 0000 0000 0101`
				
				`let answer = (C1 ^ C2); // 0000 0000 0000 0000 0000 0000 0000 1110`
				`//Output result: 14`

	|	Bitwise OR: The result bit is 1 if one or both corresponding bits of the two operands are 1.	a | b	`const C1 = 11; // 0000 0000 0000 0000 0000 0000 0000 1011`
				`const C2 = 5; // 0000 0000 0000 0000 0000 0000 0000 0101`
				
				`let answer = (C1 | C2); // 0000 0000 0000 0000 0000 0000 0000 1111`
				`//Output result: 15`

	~	Bitwise NOT: Inverts all bits of the operand.	a ~ b	`const C1 = 11; // 0000 0000 0000 0000 0000 0000 0000 1011`
				`const C2 = 5; // 0000 0000 0000 0000 0000 0000 0000 0101`
				
				`let C3= (~ C1); // 1111 1111 1111 1111 1111 1111 1111 0100`
				`//Output result: -12`
				
				`let C4 = (~ C2); // 1111 1111 1111 1111 1111 1111 1111 1010`
				`//Output result: -6`

13.10.2.6.3 Operator Precedence

The precedence of operators determines the order in which operations are executed in an expression. Operators with higher precedence are executed first.

The table below lists common operators ordered by precedence, from highest (19) to lowest (1).

Precedence	Operator Type	Associativity	Operator
19	Grouping	N/A (Not applicable)	(...)
18	Member access	Left to right	... .
	new (with argument list)	N/A	new ... (...)
	Function call	From left to right	... (...)
17	new (without argument list)	Right to left	new ...
16	Post-increment	N/A	... ++
	Post-decrement		... --
15	Logical NOT (!)	From right to left	! ...
	Bitwise NOT (~)		~ ...
	Pre-increment		++ ...
	Pre-decrement		-- ...
	type of		typeof ...
	void		void ...
	delete		delete ...
	await		await ...
14	Exponentiation (**)	From right to left	... ** ...
13	Multiplication (*)	From left to right	... * ...
	Division (/)		... / ...
	Modulo (%)		... % ...

Precedence	Operator Type	Associativity	Operator
12	Addition (+)	From left to right	... + ...
	Subtraction (-)		... - ...
11	Bitwise left shift (<<)	From left to right	... << ...
	Bitwise right shift (>>)		... >> ...
10	Less than (<)	From left to right	... < ...
	Less than or equal to (<=)		... <= ...
	Greater than (>)		... > ...
	Greater than or equal to (>=)		... >= ...
	in		... in ...
	instanceof		... instanceof ...
9	Equal (==)	From left to right	... == ...
	Not equal (!==)		... != ...
	Strict equality (===)		... === ...
	Strict inequality (!==)		... !== ...
8	Bitwise AND (&)	From left to right	... & ...
7	Bitwise XOR (^)	From left to right	... ^ ...
6	Bitwise OR (|)	From left to right	... | ...
5	Logical AND (&&)	From left to right	... && ...
4	Logical OR (||)	From left to right	... || ...
3	Conditional (ternary) operator	From right to left	... ? ... : ...
2	Assignment	From right to left	... = ...
			... += ...
			... -= ...
			... **= ...
			... *= ...
			... /= ...
			... %= ...
			... <<= ...
			... >>= ...
			... >>>= ...

Precedence	Operator Type	Associativity	Operator
			… &= …
			… ^= …
			… |= …
			… &&= …
			… ||= …
			… ??= …
1	Comma / Sequence	From left to right	… , …

Relationship between expression operator precedence and associativity:

- When an expression contains only operators of different precedence levels, precedence takes priority over associativity and does not affect the output.
- When an expression contains multiple operators with the same precedence level, associativity determines the order of execution;
  For example, the exponentiation operator is right-associative, so **4 ** 2 ** 3** and **4 ** (2 ** 3)** have the same execution order and result, but **(4 ** 2) ** 3** changes the order and yields a different result.

```
4 ** 2 ** 3;     //  65536

4 ** (2 ** 3);   //  65536

(2 ** 2) ** 3;   //  4096
```

13.11 Installment Payment

- 13.11.1 Function Overview
- 13.11.2 Enabling Installment Payment
- 13.11.3 Selecting an Installment Strategy
- 13.11.4 Installment Unlocking Modes
- 13.11.5 Configuring Pre-Due Reminder
- 13.11.6 Configuring Lock Screen
- 13.11.7 Impact of Installment Payment on Other Functions

13.11.1 Function Overview

The installment payment feature provides an embedded payment guarantee solution for equipment sales. You can divide the total price of the equipment into multiple stages, with each stage's payment due dates directly associated with the device's usage permissions, thereby safeguarding the supplier's payment rights at the software level.

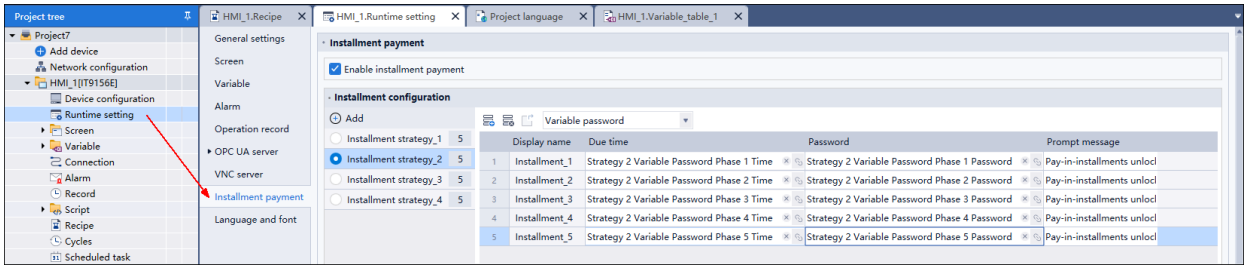
Operation strategy

1. During the project configuration phase, the equipment supplier or owner pre-defines the installment payment strategy, including: the total number of installments, the due date corresponding to each installment, the amount payable, and the unique authorization password for that installment.
2. After the project is put into operation, the device follows the pre-defined installment schedule. When each installment reaches its due date, the system automatically triggers a control mechanism that locks the device's normal functions and forces it into a pending-authorization state.
3. After the user fulfills the current payment obligation, they obtain the activation password for that installment from the supplier. Upon entering the correct password in the device prompt interface, the device is unlocked, full usage rights are restored, and the system proceeds to the next payment cycle.

13.11.2 Enabling Installment Payment

Enabling

The installment payment function is controlled by a master switch. In the device's runtime configuration, there is a top-level "Enable" checkbox, which has the highest priority:



Unchecked: The entire function is disabled. All related settings are invalid, and the device will not perform any expiration checks or locking.

Checked: The function is enabled, and all installment-related options can be configured.

Procedure

After enabling the installment payment function, follow the steps below to ensure proper operation:

1. Configure the installment strategy and term
   Specify the installment strategy, the number of installments, and the due date for each installment. For related instructions, see 13.11.3 Selecting an Installment Strategy.
2. Configure the installment unlocking mode
   There are two installment unlocking modes. For related instructions, see 13.11.4 Installment Unlocking Modes.
3. Configure the pre-due reminder
   Set the time period and reminder frequency for advance notifications. For related instructions, see 13.11.5 Configuring Pre-Due Reminder.
4. Configure the unlocking interface
   Set the lock screen style displayed after the device expires. For related instructions, see 13.11.6 Configuring Lock Screen.
5. Compile and download project
   Compile the project and download it to the HMI device.

6. Function activation and expiration unlock
   After the device is running, the system monitors the time automatically. Upon expiration, the screen will be locked as configured. You must enter or import the correct password to unlock the device.
   When dynamic password or dynamic numeric password are enabled, a periodically changing dynamic code generated by the system will be displayed on the lock screen. The device owner needs to use this dynamic code to generate the corresponding password before unlocking.

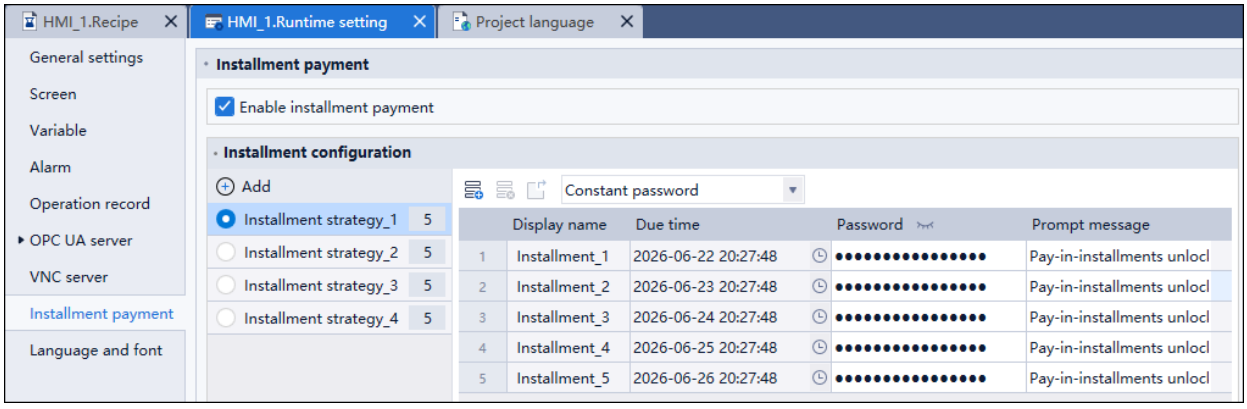
13.11.3 Selecting an Installment Strategy

A device supports configuring up to 10 different installment strategies. Each strategy allows a maximum of 48 installments. Four strategies are supported for installment payments, and their characteristics are compared as follows:

Strategy	Unlock Credential	Feature	Applicable Scenario	Security
Constant password	Pre-set static password	Easy to set up and convenient for users to remember. Once leaked, the entire project must be modified.	Suitable for internal trials or short-term installments with low security requirements.	low
Variable password	Value stored in a specified variable	The password can be modified dynamically without modifying the project. It is convenient and flexible.	Scenarios requiring remote or automatic password changes.	Mid
Dynamic password	Dynamic code generated by the server, consisting of numbers and letters.	The password is valid for one-time use and cannot be reused. It is highly secure.	Scenarios requiring strict control and after-sales service.	Highest
Dynamic numeric password	Dynamic code generated by the server, consisting only of numbers.			high

Constant password

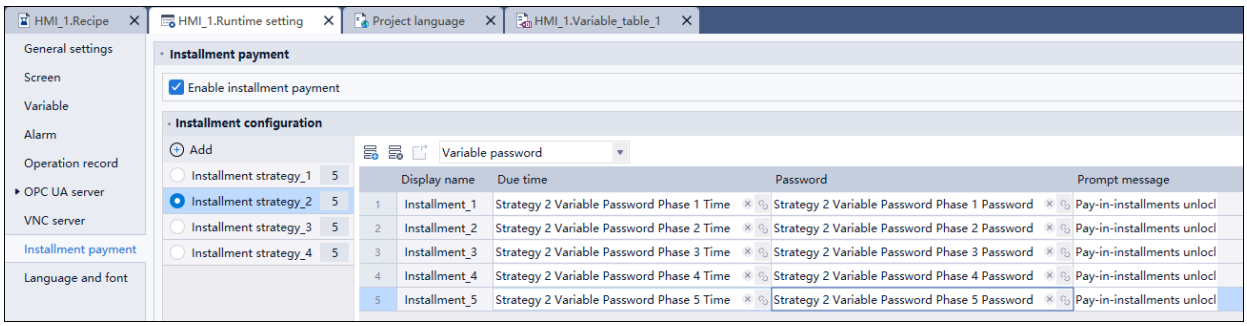
Each installment is configured with a fixed due time and password.



Password: Consists of numbers and letters, with a maximum length of 32 characters.

Variable password

The due time and password are dynamically provided by variables.



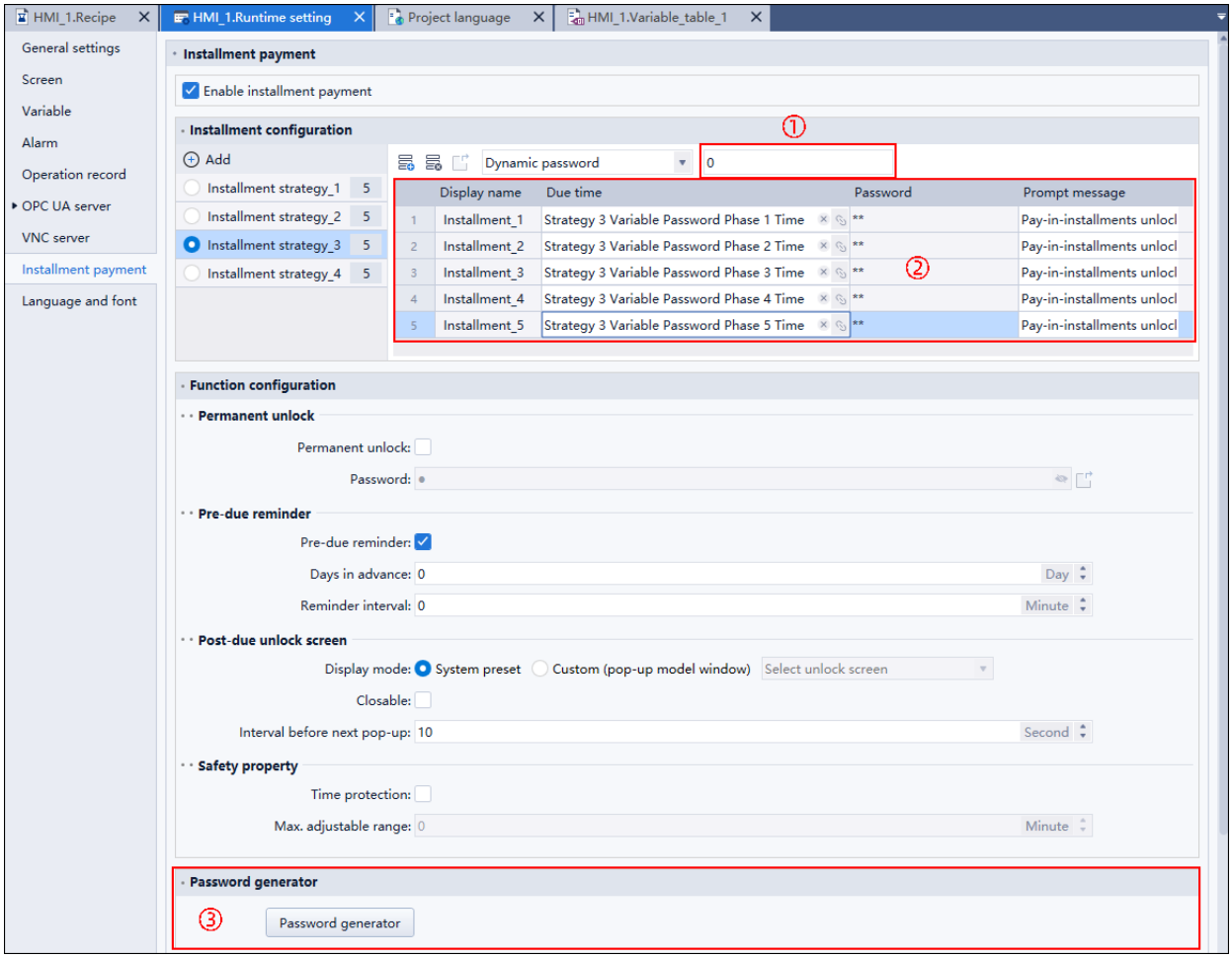
- Due time: Specify a DateTime-type variable to store the authorized due time.
  It is recommended to set the initial value of the variable to a future time later than the project download time.
  If the initial value of this variable is empty, the system automatically uses the default value 1970/01/01 08:00:00. Since this time is far earlier than the current time, the device will be immediately marked as expired upon downloading the project, triggering a lockout.
- Password: Specify a String or WString variable.

> NOTE
>
> A password written online is valid only if it is written before the due time.

Dynamic password

When the device authorization expires, the system automatically generates and displays a dynamic code consisting of numbers and letters on the lock screen. The device vendor or authorized service personnel use a dedicated password generator to obtain a unique, one-time unlock password corresponding to the dynamic code.

The device user unlocks the device interface using this password to restore normal operation.



- Installment configuration

  At Position 1, configure the static passphrase - a preset passphrase managed by the device owner, used to verify the operator's permission.

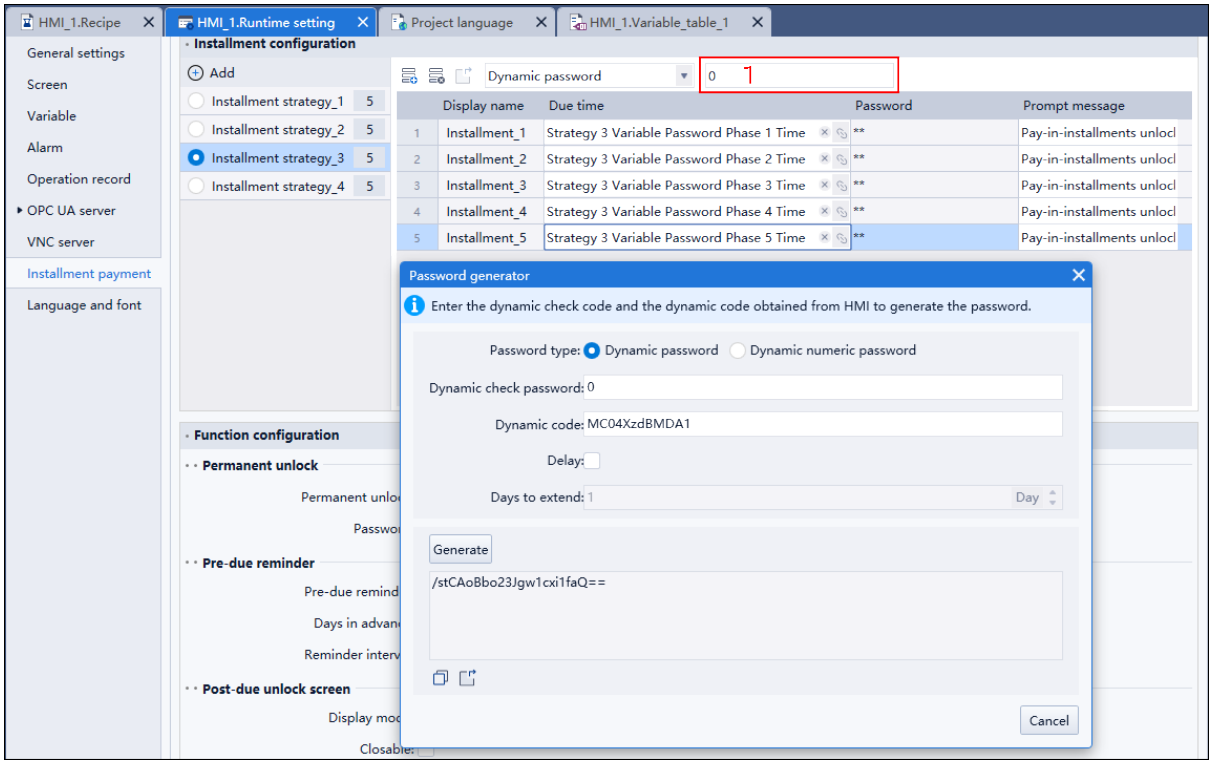
  The passphrase consists of 1 to 9 digits and cannot be empty.

  At Position 2, configure the time variable. For variable requirements, see the Variable password section above.

> NOTE
>
> Each installment must be bound to a different DateTime variable. Initial values must be distinct, and each subsequent due time must be later than the previous one.

- Password generator

  At Position 3, open the password generator. The window is shown in the figure below.



Password generator contains the following configuration items:

Parameter	Description
Password type	Select the corresponding installment strategy.
Dynamic check password	Enter the pre-set passphrase from Position 1.
Dynamic code	Enter the dynamic code displayed on the installment lock screen.
Delay	Extend the current installment authorization period.
Days to extend	
Generate password	Generate the unlock password.
	Copy the unlock password to clipboard.
	Export the unlock password to a txt file. File naming: DynamicPasswordPolicy_InovanceExpiryKey.

Dynamic numeric password

Dynamic numeric password is a specific form of dynamic password, and it works exactly the same. There are two differences:

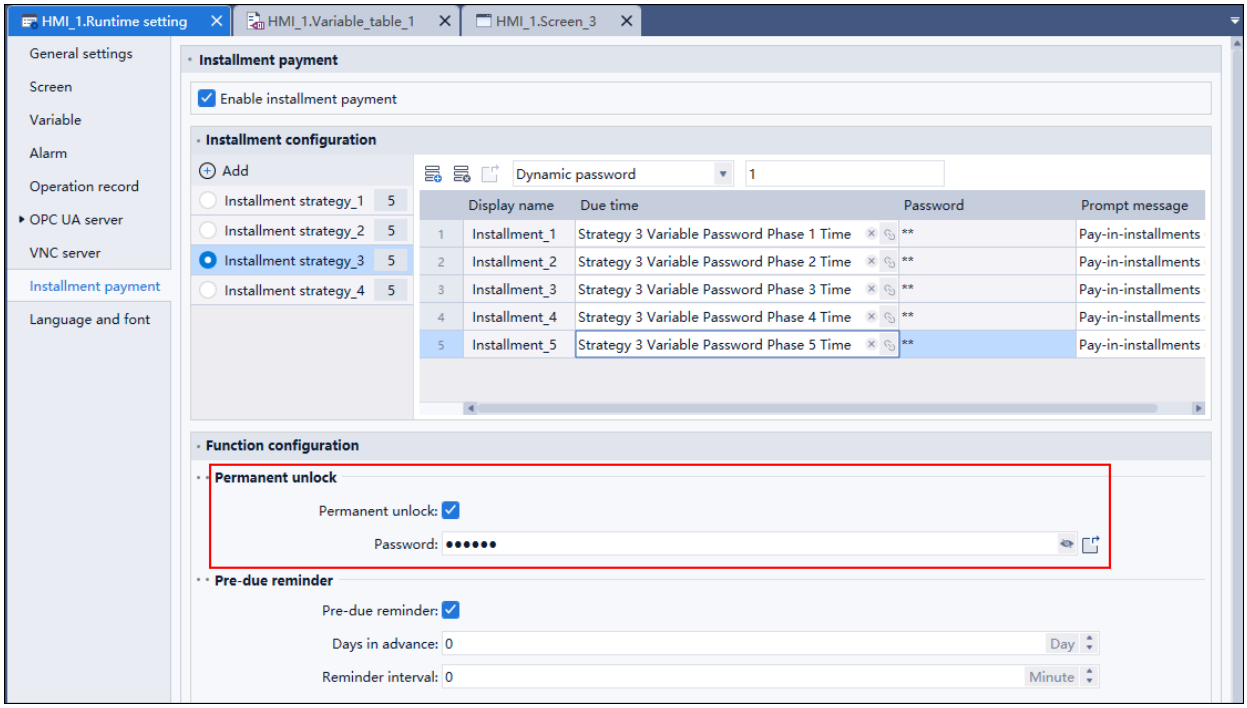
- The dynamic code consists of numeric digits only (no letters).
- The exported txt file is named: DynamicNumericPasswordStrategy_InovanceExpiryKey.

13.11.4 Installment Unlocking Modes

The system provides two authorization modes to conveniently adapt to different authorization delivery and maintenance scenarios. The default is **unlocking the current installment**.

Mode	Trigger Mode	Password Rule	Description
Unlock the current installment	Enter the password for the corresponding installment to unlock.	The password is pre-defined in the installment strategy. The password requirements vary depending on the strategy. For details, see 13.11.3 Selecting an Installment Strategy.	Unlock sequentially in installment order (Installment 1 → Installment 2). The password for each installment is valid only for that specific installment and becomes invalid after unlocking.
Unlock all installments	Enter a global unlock password.	The password consists of 32 digits.	Activate all remaining authorized installments at once.

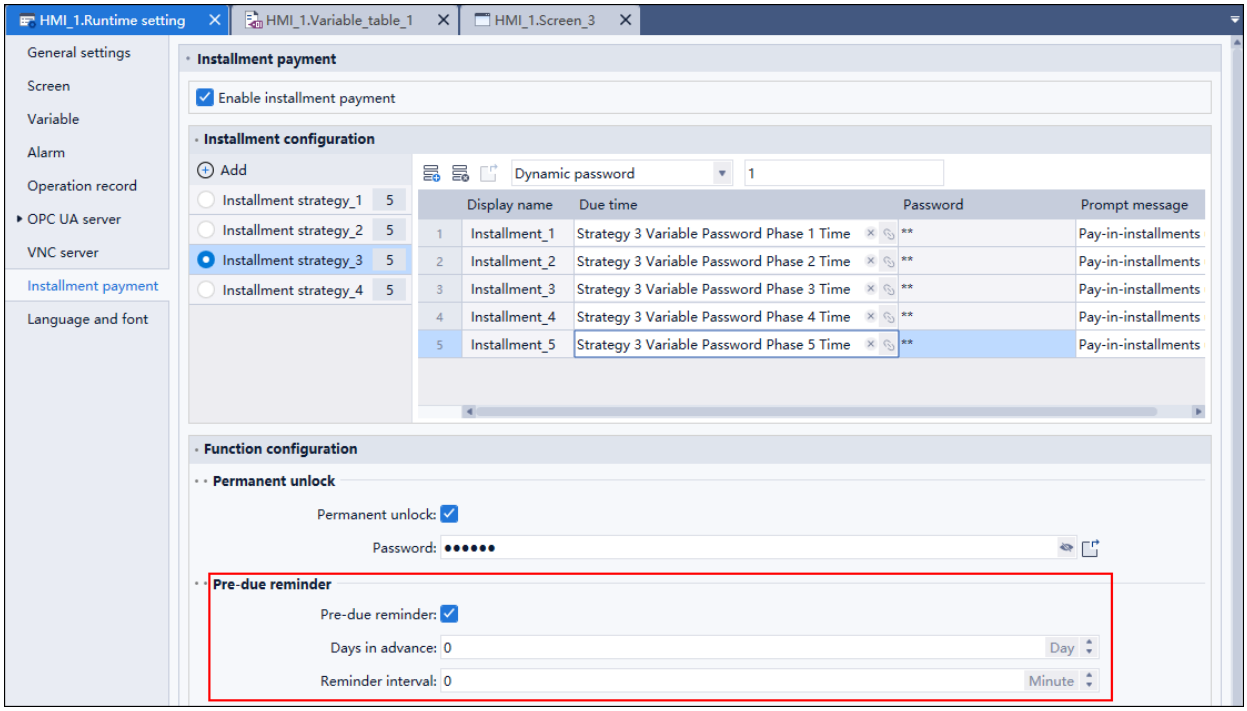
If there is a possibility to unlock all installments later, it is recommended to pre-configure the corresponding passwords in the project as shown in the figure below.



13.11.5 Configuring Pre-Due Reminder

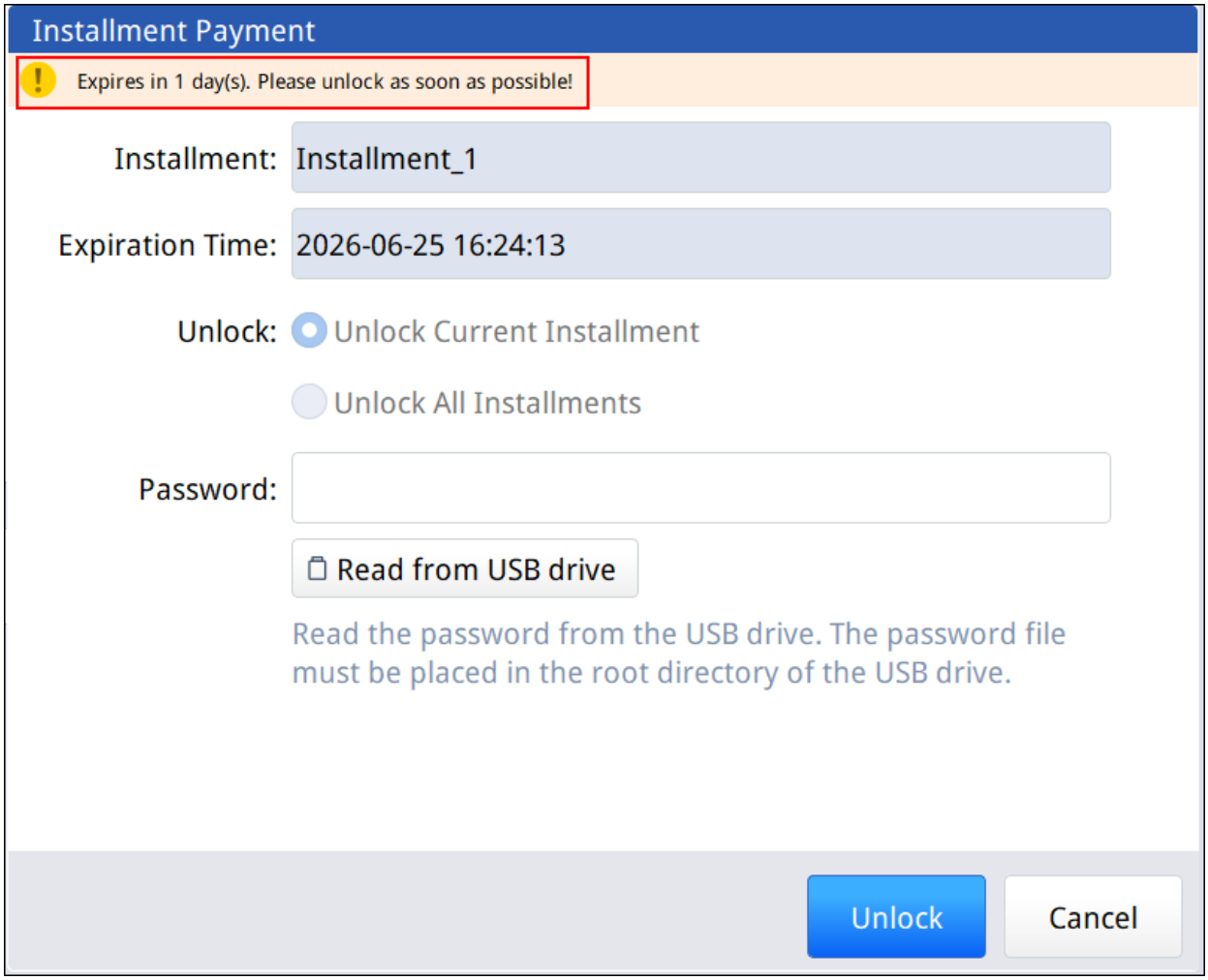
The pre-due reminder is an alarm sent by the system to the user before the authorized due date of each installment. The advance notice period can be flexibly configured, such as one day in advance or two days in advance.

The following figure shows the configuration.



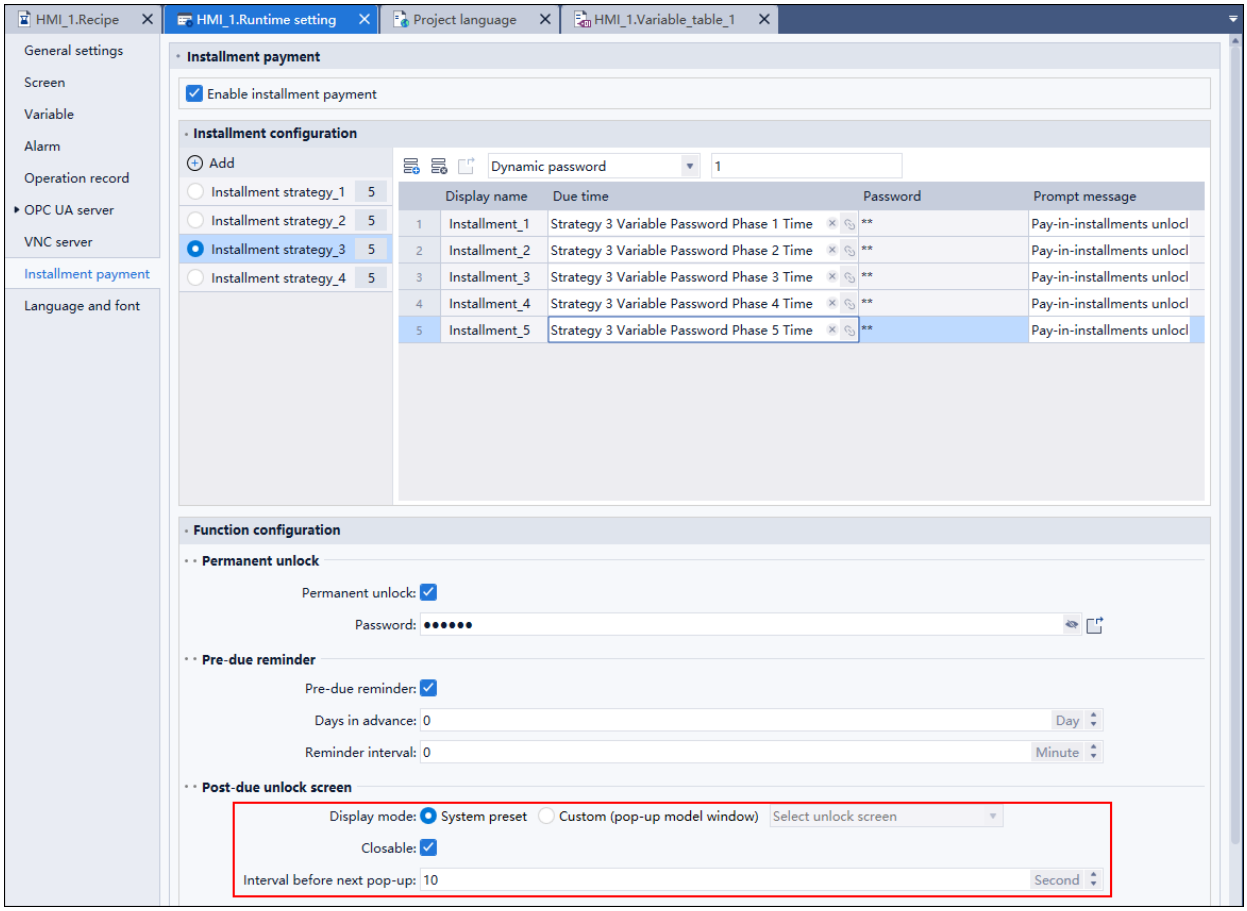
- Days in advance: Ranges from 0 to 20.
- Reminder interval: The time interval between two consecutive reminders within the configured advance days. Range (minutes): 0 to 4320. 0 indicates that only one reminder is sent when the advance days condition is first met.

The reminder position and content in the default system pop-up dialog are shown in the figure below.



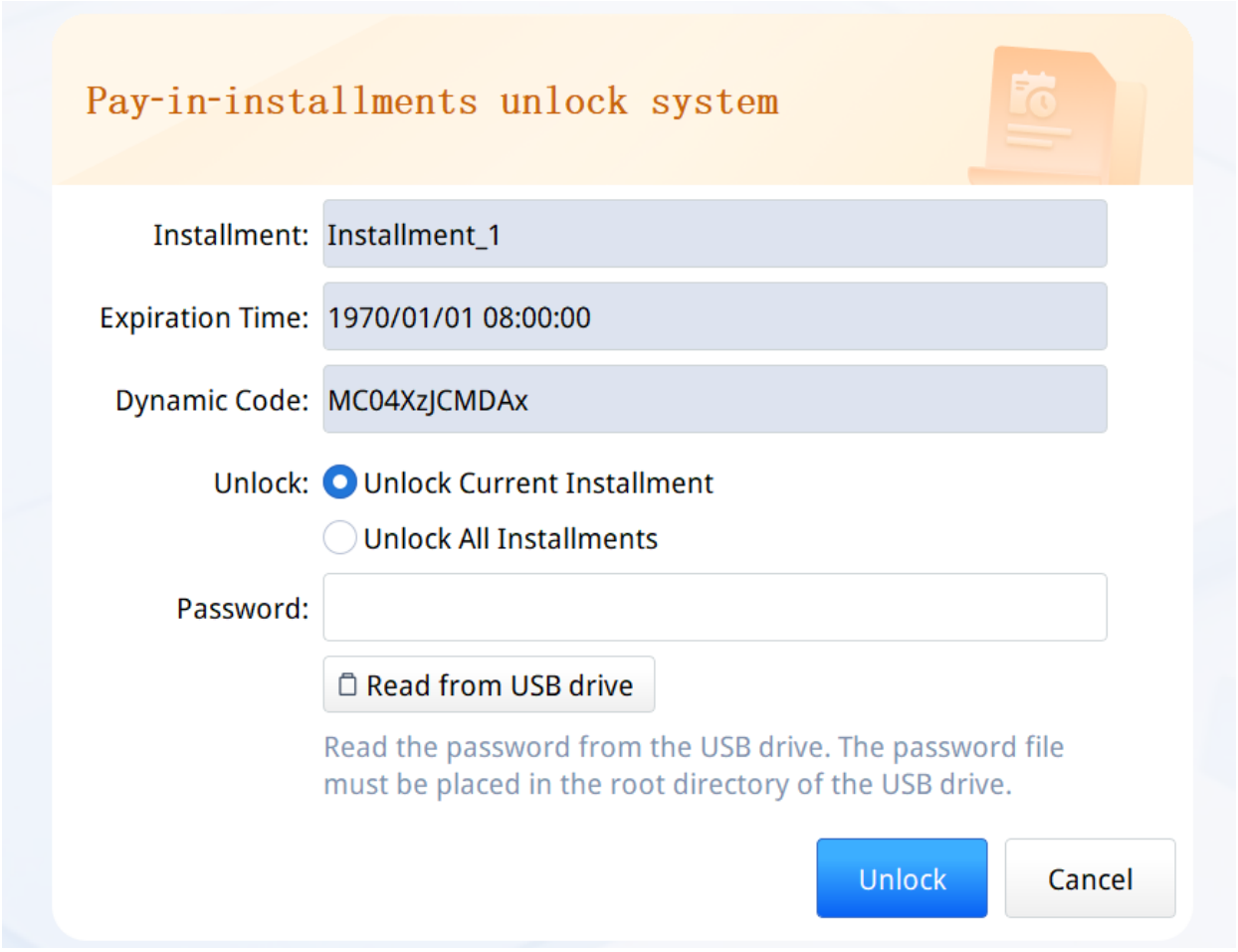
13.11.6 Configuring Lock Screen

After the device authorization expires, the lock screen mechanism will be triggered. You must enter the unlock password to continue using the device. The lock screen is divided into two types: system preset lock screen templates and user-defined screens.



System preset lock screen

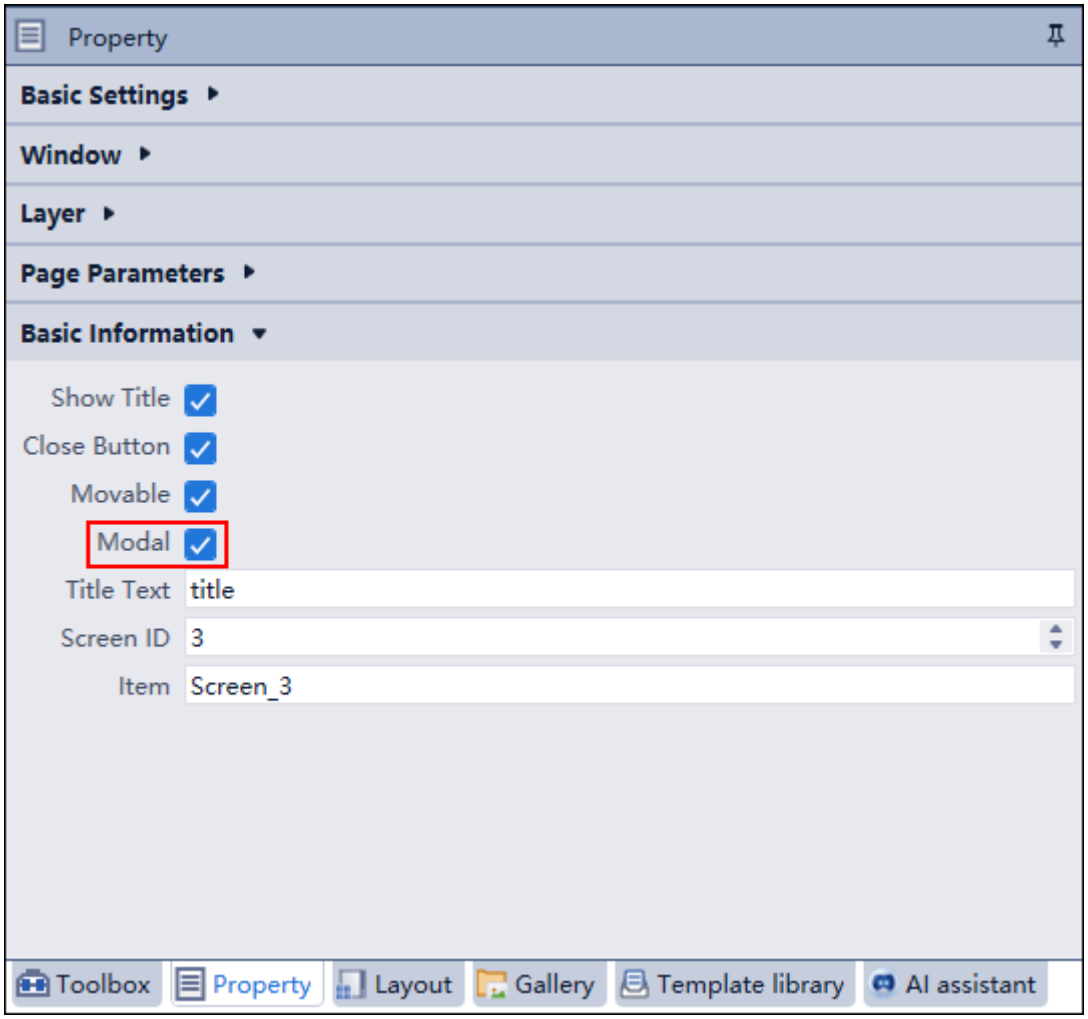
The figure below shows an example of the lock screen for the dynamic password strategy.



The "Cancel" button is only displayed when "Closable" is checked.

User-defined screen

User-defined screens are displayed via a modal pop-up, allowing users to customize the displayed content. The pop-up requires specific properties to be checked.



The pop-up can manage installment payments by monitoring system variables and configuring related system functions.

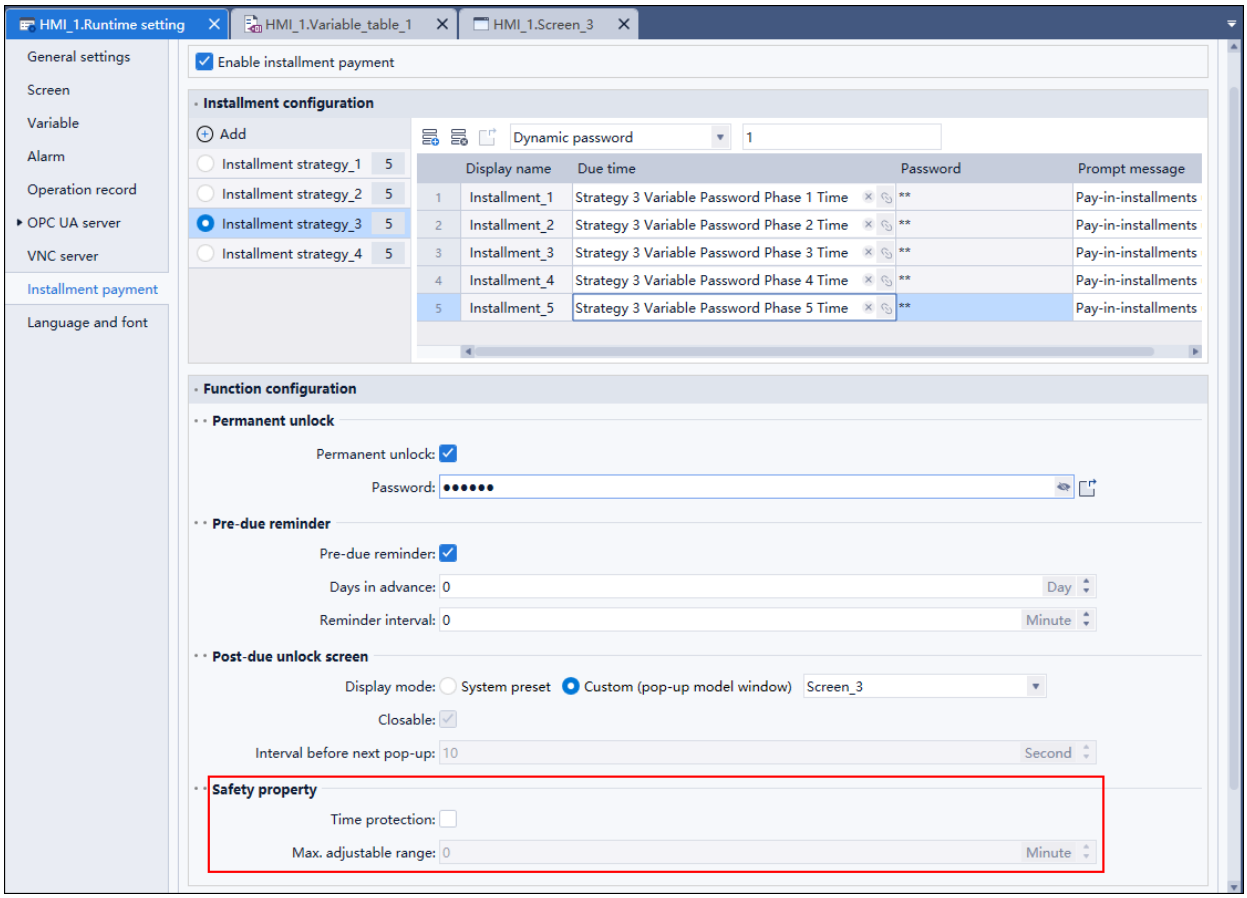
Item		Description
System Variables	$InstallMentLockState	Installment lock status. 1: Device locked (waiting for password input); 0: Device unlocked
	$InstallMentLockRemain Time	Remaining time until next installment unlock (in minutes). Value is 0 when all installments are unlocked.
	$InstallMentRemainPeriod	Total number of remaining unlocked installments in the current strategy.
	$InstallMentLockTrigger	Trigger flag for the next installment.
	$InstallmentFunctionExists	Installment function existence flag. Value is 1 when a valid installment strategy is configured in the project; otherwise, it is 0.
	$DynamicCode	Dynamic code (32 characters). The dynamic code is displayed only after expiration. When the device is unlocked, this value is empty.

Item		Description
	$InstallmentVariableName	Installment name output, supports up to 128 characters. This value is empty when there are no installments.
	$InstallmentDueDate	Installment due time. This value is set to 1970-01-01 00:00:00 when there is no installment.
System Functions	19.9.13 Unlocking Installments	This function is used to manage the installment unlocking options and verify the installment unlocking password in the active installment payment strategy of a device.
	19.9.14 Triggering Installments in Advance	This function is used to unlock operation restrictions before the installment due time. Used to unlock upcoming installments or all remaining installments in advance.

13.11.7 Impact of Installment Payment on Other Functions

During the authorized installment period, the device date and time cannot be modified.

Time protection: When checked, time modification is allowed, but the maximum allowable change must not exceed one day; when unchecked, modification is prohibited.



13.12 Monitoring Runtime Configuration

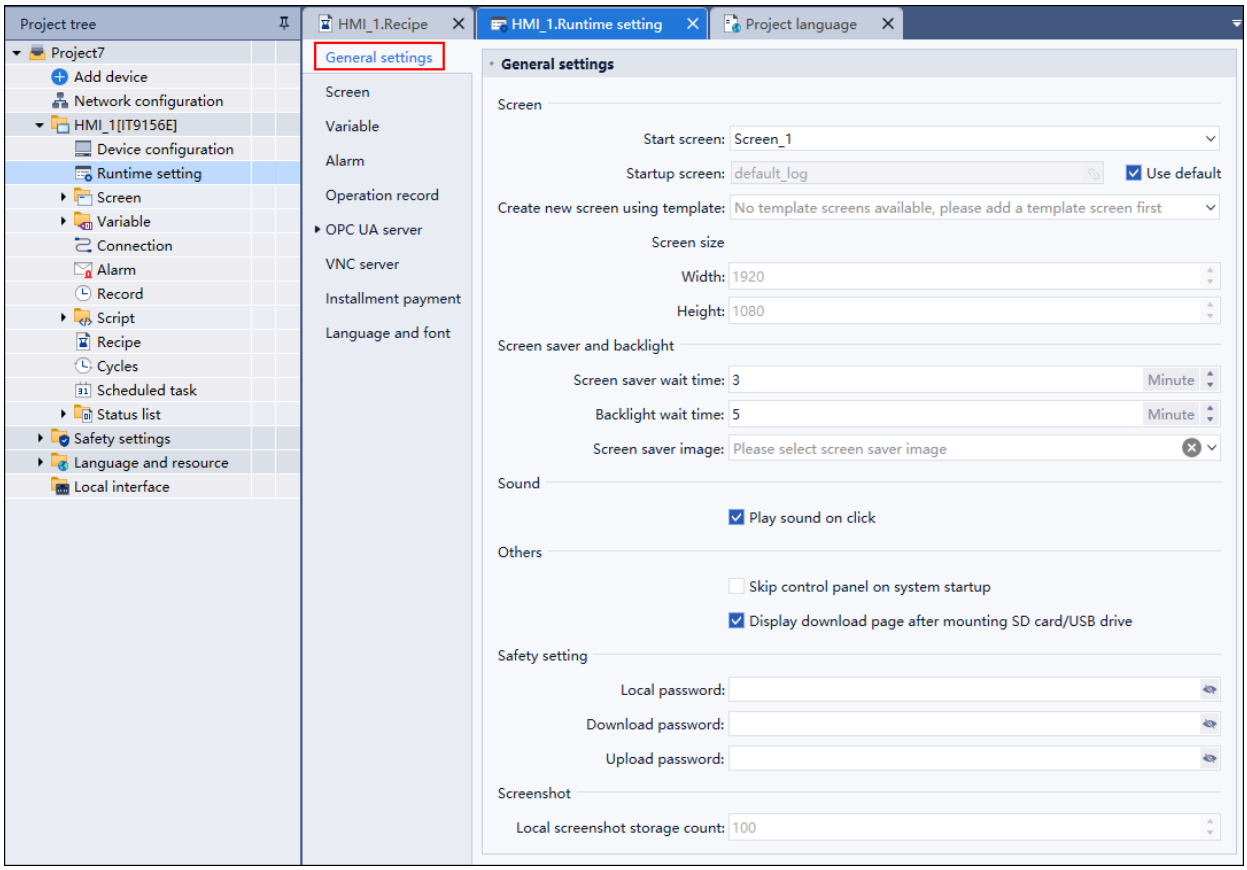
- 13.12.1 Screen Display Configuration
- 13.12.2 Preload Screen Configuration

- 13.12.3 OPC UA Server Configuration
- 13.12.4 VNC Server Configuration

13.12.1 Screen Display Configuration

Before downloading a screen to the monitoring device, certain common properties must be configured, such as the screen saver wait time and the startup screen.

These properties are displayed on the "General settings" tab of the "Runtime setting" node in the project tree.



Category	Item	Description
Screen	Start screen	The first screen displayed when the project is running. The start screen node is identified as . The start screen can also be set through the right-click menu of the screen node.
	Startup screen	The screen displayed on the HMI during startup. The default is the Inovance logo. Supports 8-bit and 24-bit BMP format images, with a maximum image size of 3 MB. The image set here depends on the project gallery. If this image is deleted from the project gallery, a fault will be reported.

Category	Item	Description
	Create new screen using template	Newly created screens will automatically reference the specified template. Only screens created after this setting will use the template.
	Screen size	The screen size of the PC device, in pixels. The range for both width and height is 1 to 10000. Not supported by HMIs.
Screen saver and backlight	Screen saver wait time	The time the HMI waits before automatically starting the screen saver after detecting no user activity. Range: 0 to 255. Value 0 means the screen saver is disabled.
	Backlight wait time	The time after which the HMI turns off the backlight when no user activity is detected. Range: 0 to 255. Value 0 means the backlight is not turned off.
	Screen saver image	The screen displayed when the HMI enters screen saver mode.
Sound	Play sound on click	When a graphic element with a configured event is clicked on the screen, the HMI emits a sound.
Others	Skip control panel on system startup	Controls whether the control panel entry prompt is displayed during HMI startup. Checked: not displayed; unchecked: displayed. This setting takes effect when the project is downloaded to the HMI and remains active until the next download.
	Display download page after mounting SD card/ USB drive	Controls whether the HMI upload/download page of the system control panel is triggered when an SD card or USB drive is inserted. Checked: triggered; unchecked: not triggered.
Safety settings	Local password	Password for accessing the HMI control panel. The password must meet the complexity rules defined in "Safety settings > Settings". For details, see 9.5 Setting Password Policies. Password validation rules are also defined in "Safety settings > Settings". For details, see 9.6 HMI Security Authentication Policy.
	Download password	Password for downloading project data to the monitoring device. The password consists of letters, numbers, and special characters, with a maximum length of 32 characters.
	Upload password	Password for reading project data from the monitoring device. The password consists of letters, numbers, and special characters, with a maximum length of 32 characters.

Category	Item	Description
Screenshot	Local screenshot storage count	The maximum number of online screenshot files that can be stored on the local disk of the HMI. Default is 100; modification is not supported. When the limit is exceeded, the system automatically deletes the oldest historical screenshots in chronological order to free up space.

13.12.2 Preload Screen Configuration

The preload function loads frequently used complex screens into memory during system startup and retains them, improving screen switching responsiveness at runtime.

Use restrictions

- Supports standard screens.
- Does not support pop-up screens, picture-in-picture, and template screens.
- Supports preloading up to 64 screens.

Procedure

1. In the project tree, double-click the "Runtime setting" node of the target device.
2. In the opened window, switch to the "Screen preloading" tab.
3. In the screen list, configure the target screens for preloading.

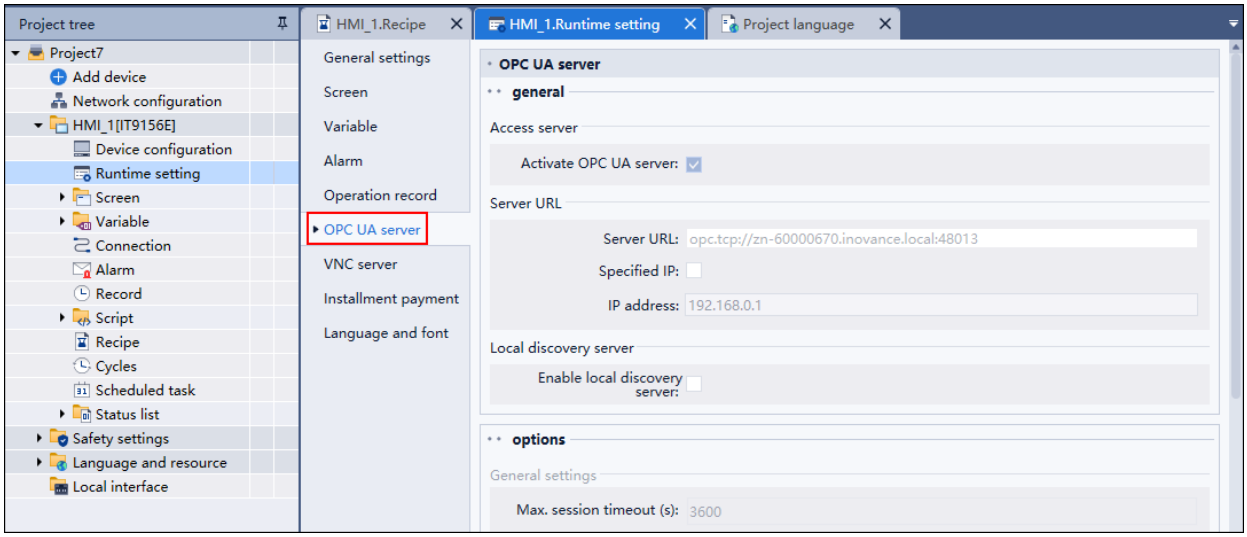
Configuration rules

- The system loads screens in the order listed. Use the shortcut tools above the list to adjust the loading order.
- Items that do not comply with the screen type rules or have duplicate screen names are marked in red.

13.12.3 OPC UA Server Configuration

The monitoring application supports the OPC UA protocol, and the monitoring device acts as a server to expose data to external clients.

Configure the OPC UA server under "Runtime setting > OPC UA server".



Category	Item	Description
General settings	Access server	The OPC UA server is activated by default and cannot be modified.
	Server URL	The URL supports access using the server's PC name or IP address.
	Local discovery server	Whether to discover local or remote OPC UA servers.
Option	General mode	Not modifiable
	Subscription mode	Not modifiable The maximum number of monitored items per subscription is 0, meaning no limit on the item count.
Safety policy	No safety settings	Data transmission does not require encryption.
	Aes128Sha256RsaOaep - Sign	Select the corresponding options based on the certificate and safety mode; multiple selections are supported.
	Aes128Sha256RsaOaep - Sign and encryption	
	Aes256Sha256RsaPss - Sign	
	Aes256Sha256RsaPss - Sign and encryption	
	Basic256Sha256 - Sign	
	Basic256Sha256 - Sign and encryption	
Authentication	Anonymous login	No authentication is required for client access.
	Username and password authentication	Client access requires identity authentication.
	User Management	Add authentication credentials. Username: Consists of letters (case-sensitive), numbers, Chinese characters, and special characters. Maximum length: 128 characters. Password: Consists of letters (case-sensitive), numbers, and special characters. Length range: 8 to 32 characters.

13.12.4 VNC Server Configuration

In remote access to the HMI, the VNC (Virtual Network Computing) server is a software component running on the HMI device. It allows authorized external devices (such as computers and tablets) to access and control the HMI interface remotely via a network connection.

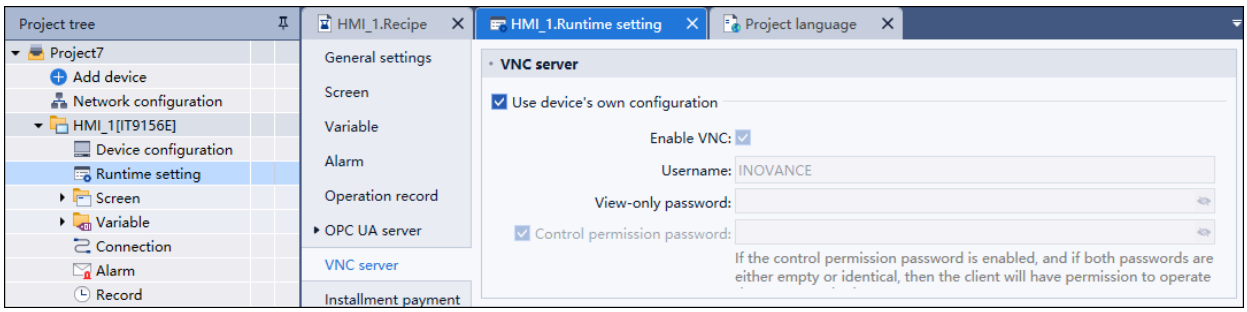
Access management configuration

Access management for the VNC server can be configured through the following three methods:

- Software interface: Configure directly in the project software.
- Hardware control panel: Configure on the HMI control panel, consistent with the software interface settings. For details, see 20.3.1 IT9000 Series Human Machine Interface User Guide.

- System functions in software configuration: Set VNC parameters by calling events. For details, see 19.12 VNC Operations.

This section focuses on configuring VNC server access permissions through the software interface.



Item	Description
Use device's own configuration	Determines whether the current configuration or the HMI control panel configuration take effect. Checked: HMI control panel configuration takes effect. Unchecked: Current software configuration takes effect.
Enable VNC	Enables or disables remote access to the HMI. Same as the function "19.12.1 Setting the VNC Enable". Checked: Remote access allowed. Unchecked: Remote access prohibited.
Username	Specifies the username for remote access to the HMI. Max length: 32 English characters. Same as the function 19.12.2 Setting the VNC Username.
View-only password	Sets the password for remote monitoring access. Password rules: • Maximum length is 32 characters. • Supports letters, numbers, special characters, and punctuation marks. Same as the function 19.12.3 Setting the VNC View-Only Password.
Control permission password	Sets the password for full remote operation access. Password rules: • Maximum length is 32 characters. • Supports letters, numbers, special characters, and punctuation marks. Its permission management is the same as the function 19.12.5 Setting the VNC Control Permission. Its password configuration is the same as the function 19.12.4 Setting the VNC Control Password.

> NOTE
>
> The storage of view-only passwords and control permission passwords uses AES-128 encryption, ensuring secure, reliable, and efficient protection of stored passwords.

13.12.5 VNC Client Configuration

The VNC client must be installed on the current operating device to connect to and control the remote device (HMI). To ensure compatibility and optimal performance, the following clients are recommended:

PC

- TigerVNC version 1.15.80 or later (https://tigervnc.org)
- RealVNC version 6.22 or later (https://www.realvnc.com)

Mobile

- AVNC version 2.7.0 or later (for Android systems)
  AVNC may not be available in some Android app stores. If not found, search for "AVNC APK" via a web browser.

Before use, ensure the following:

1. The VNC server on the HMI is enabled. For details, see the hardware device user guide.
2. The HMI is connected to the network and has obtained a corresponding IP address.

13.13 Compilation Configuration

- 13.13.1 Compiling/Clearing
- 13.13.2 Saving a Compiled File As

13.13.1 Compiling/Clearing

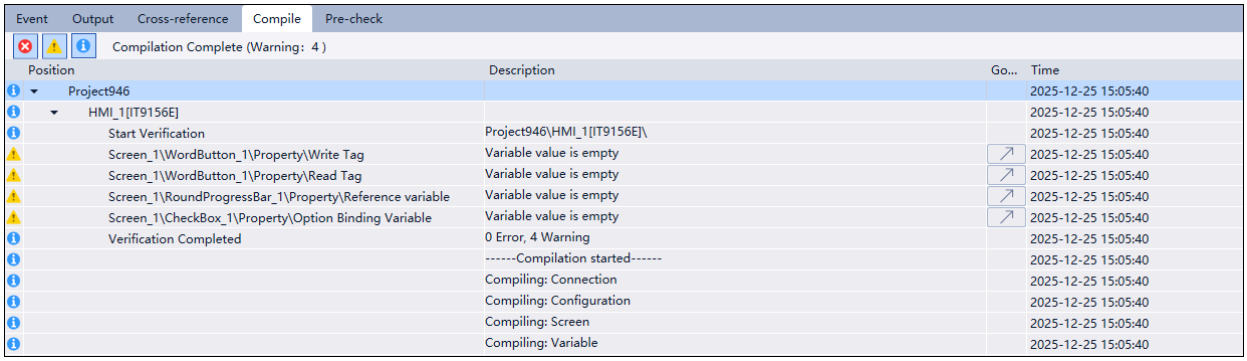
Compiling

After you have designed a project, you need to compile the project before running it.

To use the compile function, select "Compile (C) > Compile" from the menu bar. You can also click the corresponding icon on the toolbar, or use the shortcut key Ctrl + B to compile the configuration project.

During compilation, the correctness, boundary conditions, and associations of the configuration project data are verified.

After compilation, the results are displayed in the output view.



The icons in the upper-left corner of the figure below represent the three types of compilation output: fault, alarm, and info. Users can click the icons to hide or show the corresponding output.

If some control properties are set incorrectly in the configuration, alarms or faults will be reported during compilation. You can locate the specific issue by double-clicking the corresponding prompt in the output view. After modifying the configuration project to resolve alarms and faults, the project can be compiled and downloaded normally.

If the project files are locked, causing abnormal compilation file generation and preventing download, the clear function can be used. Select "Compile (C) > Clear/Clear all" from the menu bar, and a dialog box will pop up.

Clearing

After the user confirms in the dialog box, the compilation information will be cleared. Download can proceed after manually recompiling.

Clear: Clears the compilation information for the current device.

Clear all: Clears the compilation information for all devices in the current project, which may prevent devices from going online directly.

When the scope of project modifications is large, or when the user intends to perform a full compilation, the user can select "Compile (C) > Recompile". This automatically clears the previous compilation information and performs a new compilation. Recompile is a combined operation of clear and compile. Standard compilation is incremental, with only changed parts compiled. Recompilation compiles the entire project completely.

13.13.2 Saving a Compiled File As

Configuration files (project source code) and compiled files for HMI and PC projects can compressed. The compressed file can be copied to the monitoring device for running, replacing the standard project download process.

File packaging

Select "Compile > Save target file as" from menu bar to open the configuration window and specify the scope of the packaged files.

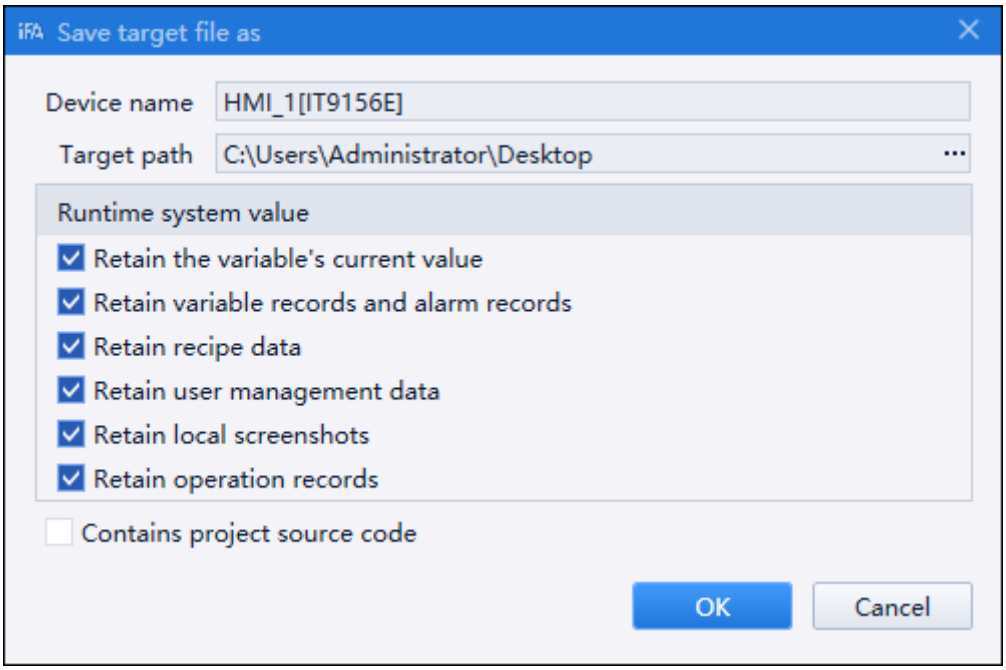


Figure 13-13-2-1 "Save target file as" window

- Device name: The device to be compiled is selected by default.
- Target path: The default path is "C:\Users\Username\Desktop". Users can modify the path.
- Runtime system value
  Refers to the data generated during runtime on the monitoring device due to online operations and variable value changes. Examples are recipe view data records added online and data records generated by variable value storage.
  When these project data are inconsistent with the offline configuration data, the user needs to determine the baseline data to be used when the project's compiled files are running on the monitoring device.
  - Retain the variable's current value: Sets the initial values for internal variables at runtime. Checked: use the last online values. Unchecked: use the initial values.
    This setting takes effect only when the internal variable's Retain property is set to .
  - Retain variable records and alarm records: Determines whether to clear historical variable data and alarm data. Checked: data is retained. Unchecked: data is cleared.

    > CAUTION
    >
    > If variable names or data types have changed, related historical data will still be cleared even if this option is checked.

  - Retain recipe data: Determines whether to retain online values and translated display names in multiple languages for recipe data records.

Object	Checked		Unchecked
Online values of data records	Retains current data and perform only incremental updates.	When an element variable is deleted, the data record corresponding to that variable will be cleared.	Clears all online values and run/display using offline configuration values.
		When an element variable is added or its name is changed, the data record for the new name will be displayed according to the element's default value.	
		When the data type of an element variable is changed, the data record for that variable will be displayed according to the element's default value.	
Multilingual translations of data record display names	Retains all current translations.	For the same recipe, data records with the same ID retain their original display names.	Uses new translations for display names.

◦ Retain user management data:

Object	Checked	Unchecked
13.3.5.4.5 User view control	Retains and applies all user account information and states that were added or edited online.	Does not retain any online changes; the control reverts to the user data defined during project configuration.
9.6 Account lockout information	Retains current failed login counts and remaining lock durations; the locked state remains active.	Resets the failed login count and lock durations; all accounts are unlocked.

◦ Retain local screenshots: Sets whether to retain screenshot files stored in the local disk directory (/**inodata/screenshot**). These files are captured online via the 19.11.1 screenshot command or the screenshot button in the trend view. Checked: retain files. Unchecked: clear files.

◦ Retain operation records: Sets whether to retain historical data of operation records. Checked: retain. Unchecked: clear.

- Include project source code
  Project source code refers to the project's offline configuration files. These files are downloaded with the project to the monitoring device as a backup for future reference and recovery.

File usage

The compressed package is named project.inovp.

It does not need to be renamed or decompressed, and can be loaded directly.

- PC device
  Copy the file to the runtime software's designated installation path: Installation directory:\Inovance View\View\Datas\ProjectData\HMI\Download. The runtime software loads the data automatically upon startup.

- HMI device
  Use the AutoRun application on the device to update the project package.
  After loading project.inovp via the HMI control panel (HMI upload/download > Download project), it is automatically copied to the project path.

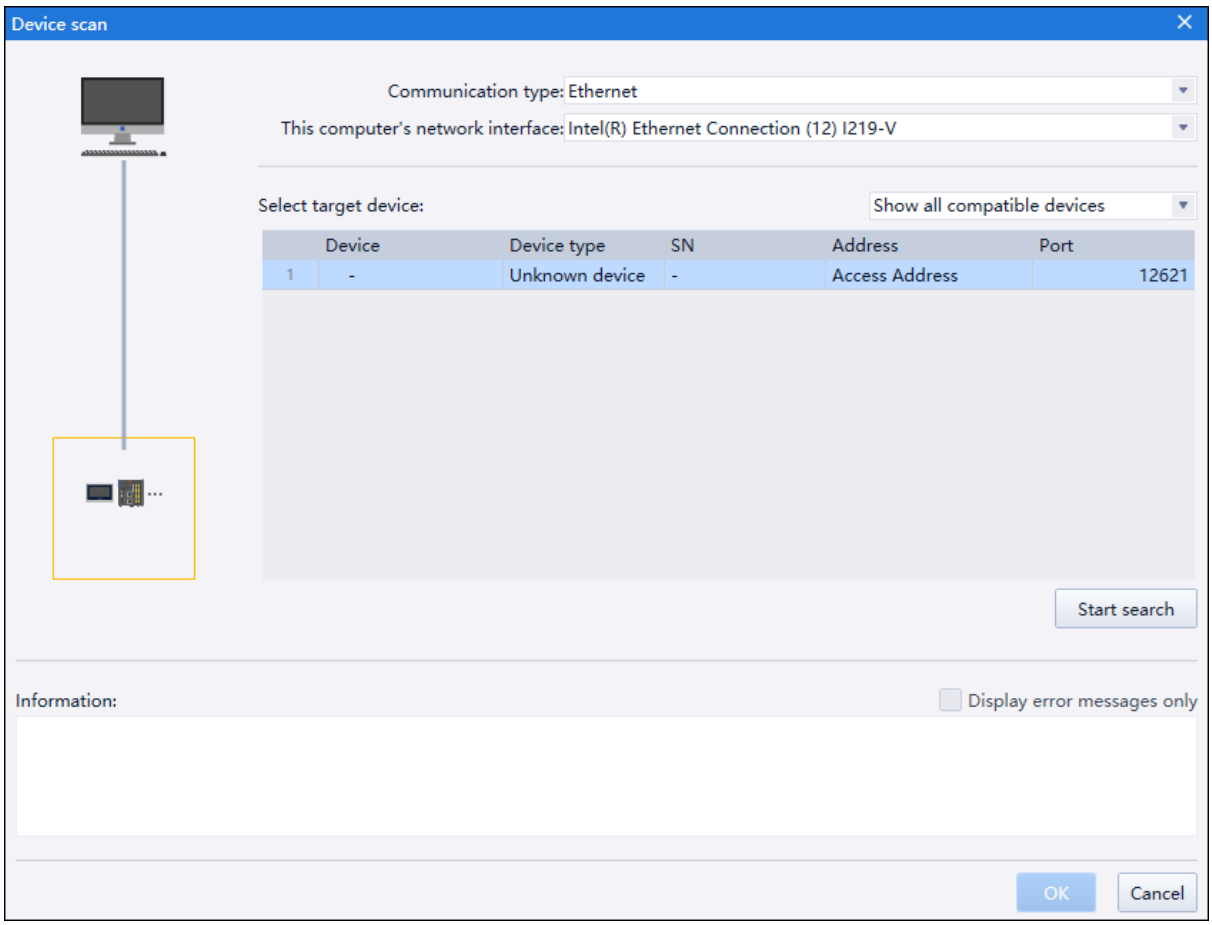
13.14 Project Download/Upload

- 13.14.1 Monitoring Device Connection
- 13.14.2 Project Download
- 13.14.3 Project Upload

13.14.1 Monitoring Device Connection

Before downloading to the monitoring device (HMI or PC), you need to establish a connection between the computer and the monitoring device, and set the IP address and other communication parameters. For HMI settings, see 7.3.1 HMI Ethernet Settings.

1. Select the "Download" command.
   The "Device scan" window pops up.



2. Set the communication parameters.
   Select the **Communication type** and the network interface of this computer, then click "Start search".
   Communication type supports both **Ethernet** and **USB** methods; choose based on the physical connection.

3. Check target device information.
   Check the target device's name, SN code, IP address, and port information, ensuring correct connection.

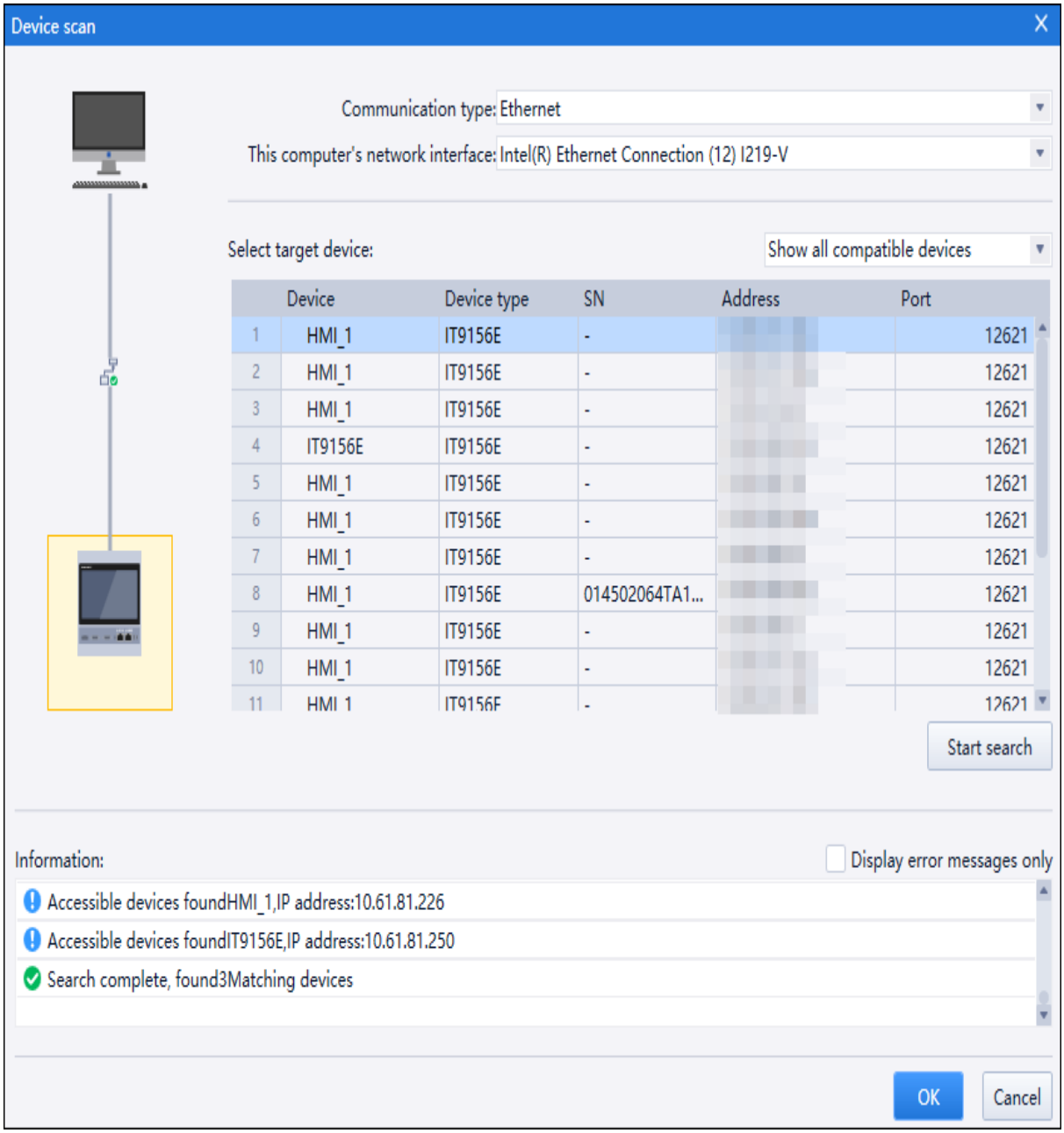


Figure 13-14-1-1 HMI search results

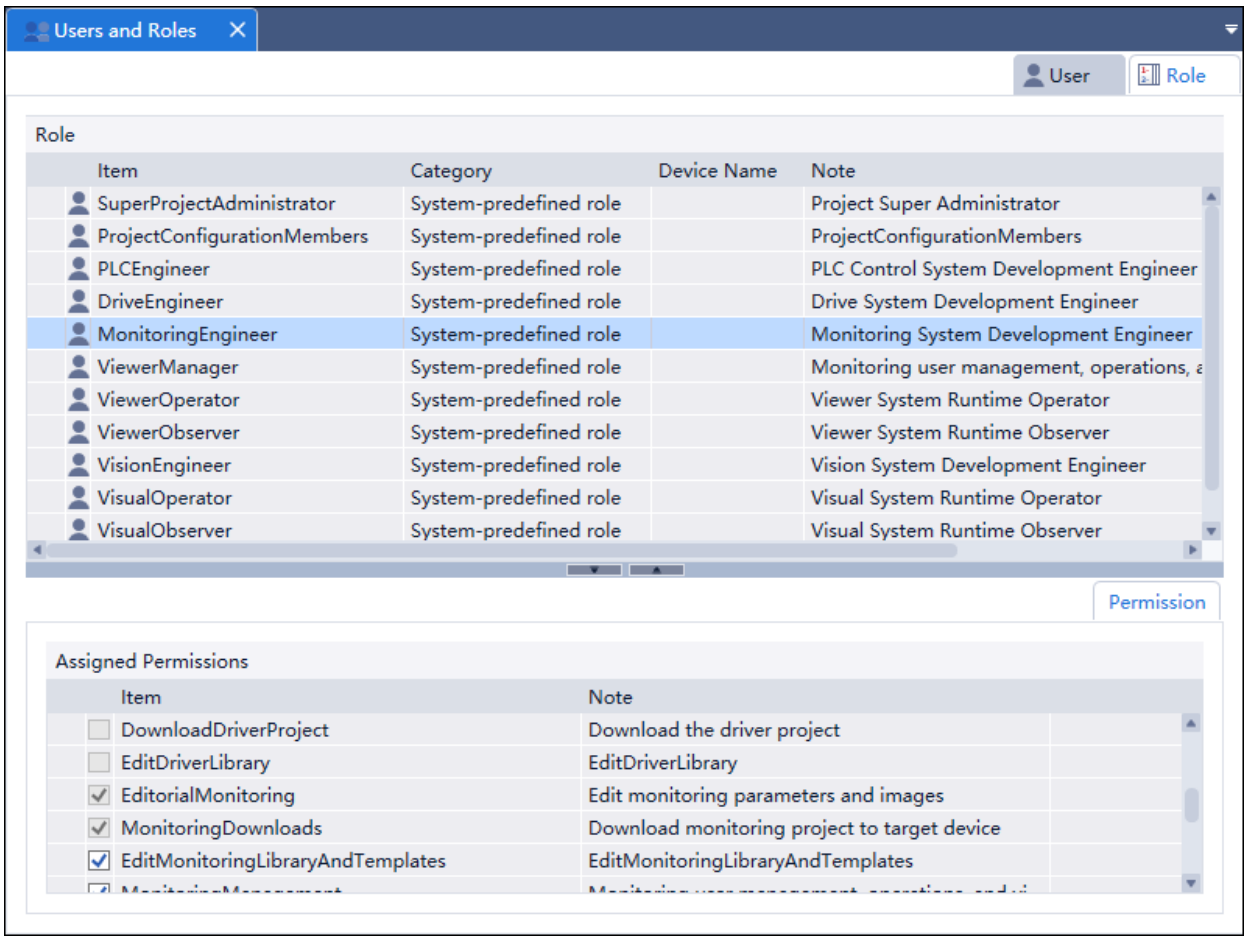
When scanning devices via USB, note that only one device can be identified at a time. Even if multiple devices are connected simultaneously, the system will randomly select one for scanning.

4. Confirm the connection object.
   Select the target device and click "OK" to establish the connection between the computer and the target device.

13.14.2 Project Download

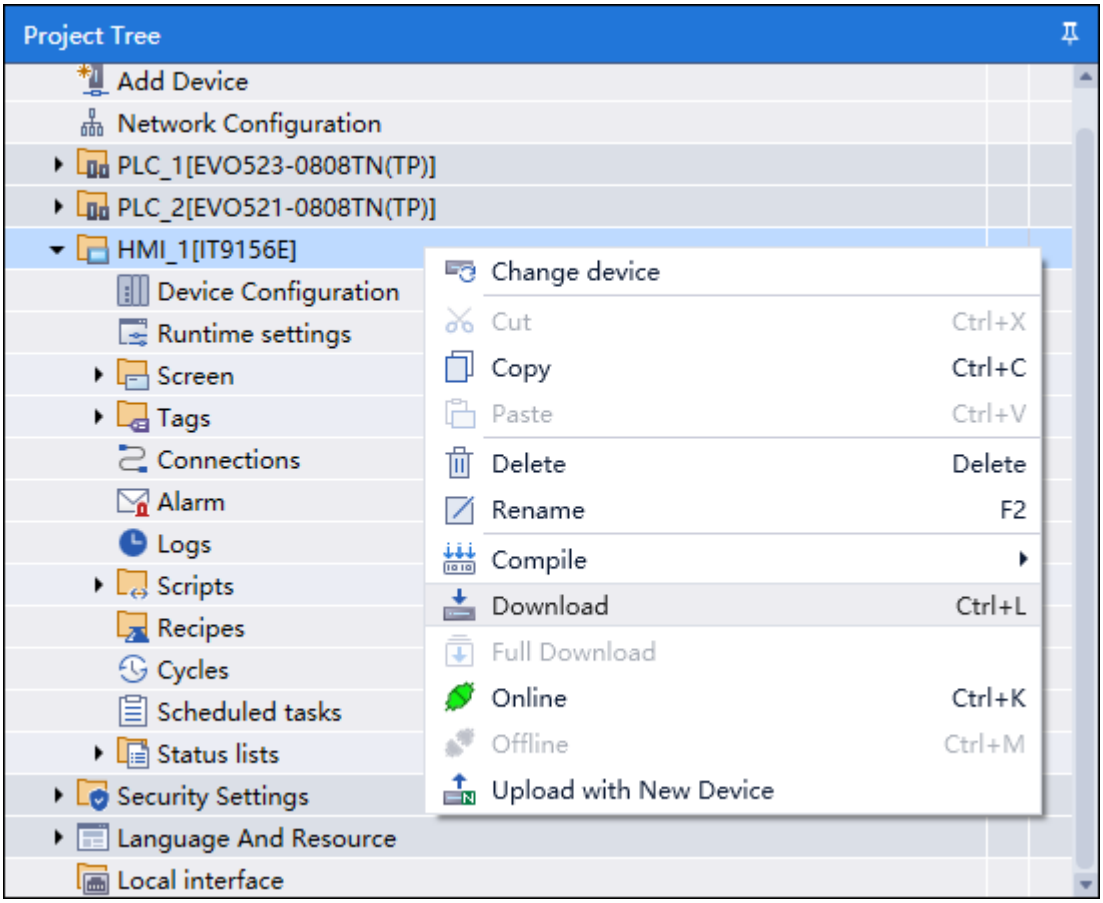
Before downloading the project to the monitoring device, ensure that the engineer has download permissions and knows the download password for the monitoring device.

Under the "User and role" node in the project tree, enable the **monitoring download** permission for the engineer. For details, see 9.7 Managing Users.



The download procedure is as follows:

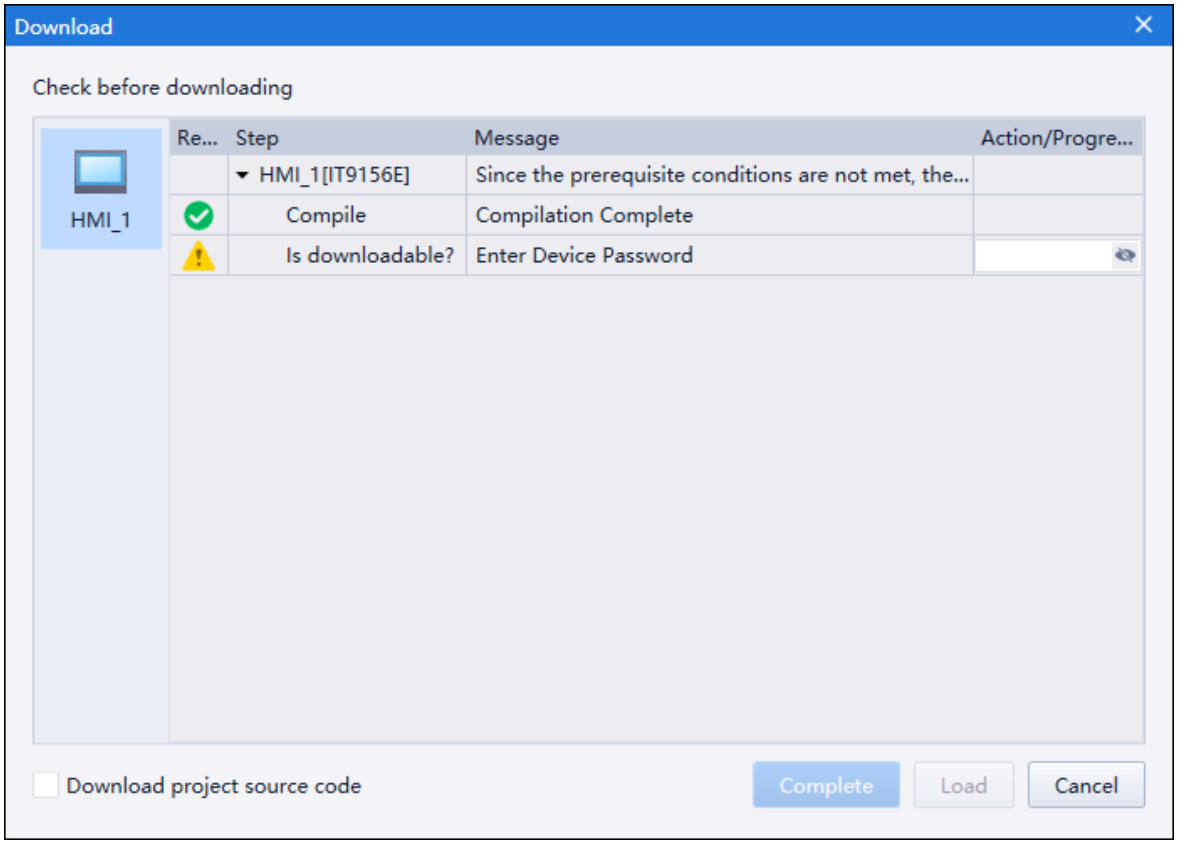
1. Select the "Download" command.
   The download command is available in three locations:
   a. Right-click the monitoring device in the project tree and select "Download".



b. Select "Online > Download" from the menu bar.

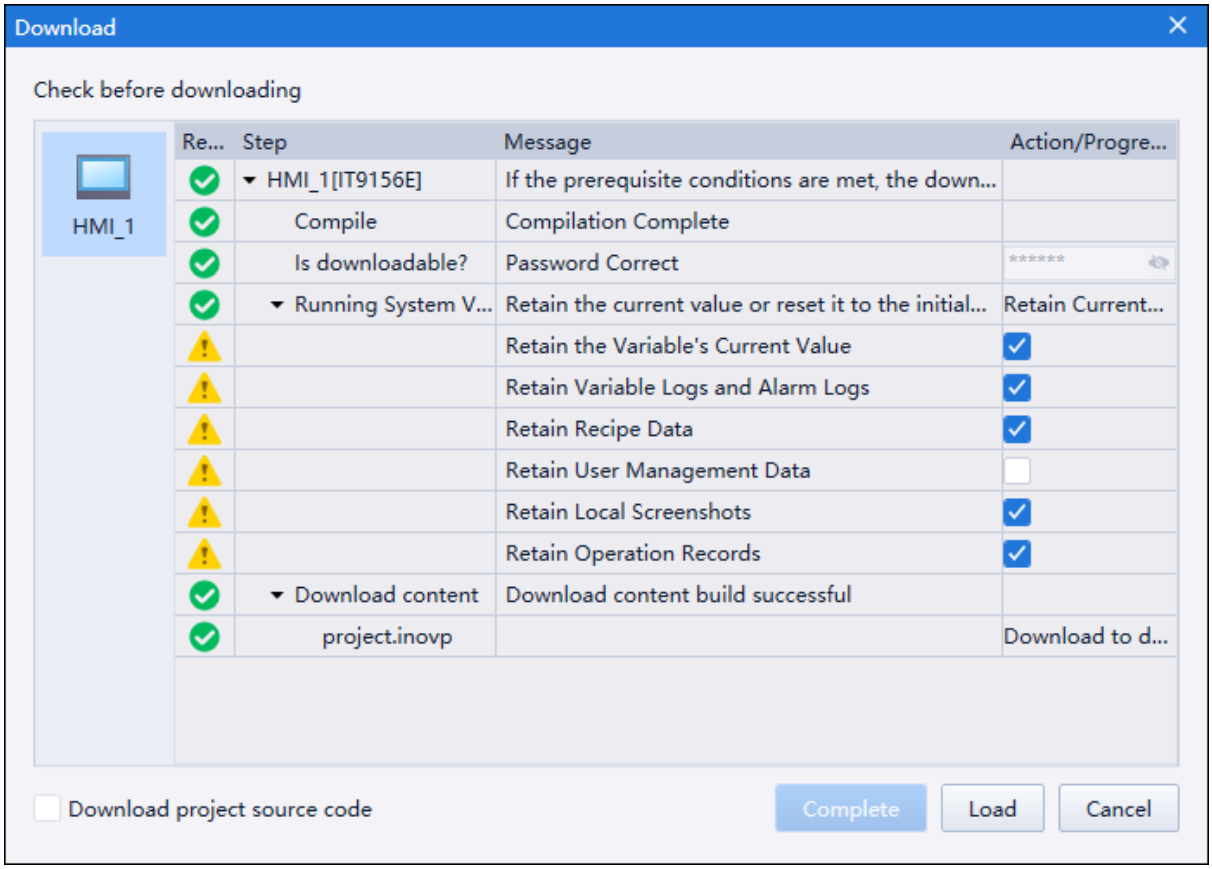
c. Select the download icon  from the toolbar.

After the project configuration is compiled, the "Download" window appears. The content of this window varies depending on whether the monitoring device has a download password set.



The download password is set in the device's general properties. For details, see 13.12.1 Screen Display Configuration.

2. Set the download scope.
   If a device password is required, enter the valid password and click anywhere in the window to complete password verification. Below is an example of a window after successful verification.



   The impact of downloading offline project content on the running data in the device, as well as whether the downloaded files include source code, need to be set in the "Download" window.

   Runtime system value

   If a project is already running on the monitoring device, the user needs to check the data items under **Runtime system value** to set whether to retain them. For descriptions of each item, see Runtime system value in 13.13.2 Saving a Compiled File As.

   Download project source code

   Project source code refers to the original configuration files of the currently downloaded project, including code, configuration files, and so on. It serves as a project backup and can be used to restore a specific version of the project during upload.

   Project source code is not mandatory for download. It is transmitted to the monitoring device as a backup only when this option is checked. These files are read and used only during project upload.

   Each download operation clears the previous download information from the monitoring device. If this option remains checked, the latest source code version is retained on the device.

   Click "Load" to start the project download.

3. Complete the download.
   When the download progress reaches 100%, the current project configuration has been successfully downloaded to the monitoring device.
   Click "OK" to end the download operation.

13.14.3 Project Upload

Project upload is a reverse-read function provided by iFA. It reads the complete project data (including screens, variables, communication parameters, scripts, alarms, recipes, and other configuration content) currently running on the HMI via a communication cable or network connection. It then synchronizes the data to the iFA project, and generates a viewable and editable project file, enabling backup and recovery of the on-site project.

Applicable Scenario

This function is suitable for scenarios such as loss of the local source project file, inconsistency between the on-site project and the local file, device replacement, and project archiving/backup. It can be used for quick retrieval of the complete program in use on-site.

Restrictions

The following conditions must be met to upload an HMI project:

- The HMI project source code must have been downloaded to the device with the "Download project source code" option checked. See 13.14.2 Project Download for details.
- If an upload password is set in the source project, please be aware of it.
- The device ID of the upload device must not duplicate any existing device ID in the current project (the software automatically detects and prompts accordingly).
- The software version running on the device must exactly match the iFA software version.
- The device model added in the iFA project must match the physical device model.

Upload scope

Uploaded data is limited to the device's own configuration content, including hardware parameter settings, scripts, internal/external variables, and screens.

For the following global functional modules, only the parts instantiated in the device will be uploaded:

- Template library
  Only symbols or custom type versions that are instantiated on the device are uploaded. For example, if a symbol has five versions but only two are used in the device, only these two versions are uploaded.
- Connections
  Only manually created PLC-HMI connections (with connection identifier ) are uploaded. Automatically created connections (with connection identifier ) are not uploaded.
- Device permissions
  If a permission configured in the source project does not exist under the same name in the current project, the permission configuration location is marked as abnormal.
- Image resources
  If local images are used in the project, they are uploaded along with the project data. If duplicate image names are encountered, the system automatically renames them.

Upload procedure

1. Open the target project in the iFA software.
2. Select the upload command.
   Select "Online > Upload with new device" from the menu.
3. Scan and select the target device.
   Scan devices via Ethernet or USB, then select the target device from the scan list.
   - USB connection: Scans and selects the connected device directly;
   - Ethernet connection: The PC and device IP addresses must be on the same subnet; otherwise, the device cannot be detected.
4. Verify the upload password.
   If an upload password was set during the download of the source project, enter the password. If no password was set, this step is skipped.
5. Execute project upload.
   Begin uploading the source project data. Monitor the prompts during the process. If a device ID conflict is detected, the upload operation will be terminated.
6. Verify and correct after upload.
   After the upload is complete, verify the result and manually adjust any anomalies.

13.15 Screen Operation

- 13.15.1 Real-time Variable Value Monitoring
- 13.15.2 Online Recipe View Operations
- 13.15.3 Online Historical Data Viewing
- 13.15.4 Online Alarm View Operations
- 13.15.6 Online Operation Record Operations
- 13.15.7 Online System Diagnostics
- 13.15.8 Online PLC Program Viewing

13.15.1 Real-time Variable Value Monitoring

When a user-designed process screen is running on the monitoring device, variable values on the screen are updated according to the following rules:

- System variables: Updated every 1s.
- User variables:
  - Updated according to the set 13.2.5.3 acquisition cycle if recording or alarm properties are configured.
  - If recording or alarm properties are not configured, variable values are updated according to the set 13.2.5.3 acquisition cycle only when the screen containing them is open. Updates stop when the screen is closed.

For objects configured on the screen, the display of numeric I/O field and character I/O field values differs under normal and abnormal communication conditions:

Control	Normal Communication		Communication Interrupted
/	Displayed in plain text	Displayed in cipher text	/
Numeric I/O field	56.30 ℃ 55 6688	***** ℃ **	0 ℃ 0 0
Character I/O field	INOVANCE	********	

CAUTION

When an IT9000 series device communicates with an EVO1000 series PLC, if the acquisition cycle of the communication variable is set to less than 500 ms, the new value may flicker after being written via the screen (symptom: the new value briefly jumps back to the old value before being written, and then returns to the new value). To avoid this phenomenon, it is recommended to set the acquisition cycle to 500 ms or longer.

13.15.2 Online Recipe View Operations

The recipe configuration data is monitored online through the 13.3.5.4.3 Recipe View.

The following shows the online operation interface of the recipe view after associating variables.

Figure 13-15-2-1 Online operation interface of the recipe view

The display effect in the recipe view can be configured on the "Property" tab on the right.

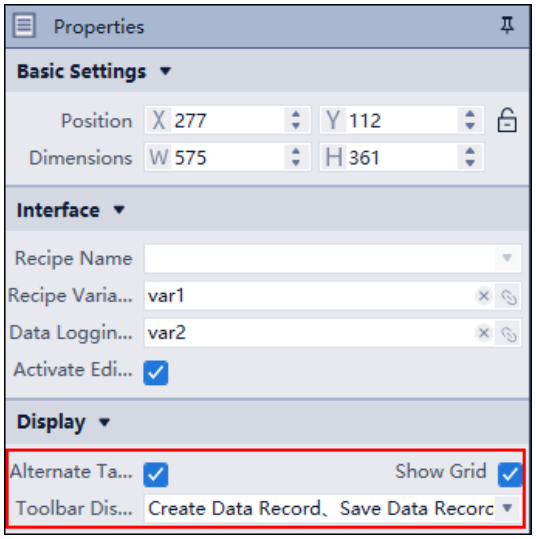


Figure 13-15-2-2 Online display properties of the recipe

The properties are described as follows:

- Recipe name: Selects the recipe to monitor by name.
- Recipe number: Selects the recipe to monitor by number.
- Data record name: Locates the corresponding row in the selected recipe by data record name and highlights it in blue, as shown in row 2 of the figure above.
- Data record number: Same function as data record name; locates the data record by ID.
- Create data record: Creates a new data record online. For naming requirements of the display name, see 13.6.2 Adding a Recipe.
  In multilingual environments, the name entered is used as the default name for all enabled languages.
- Rename data record: Modifies the display name of an existing data record. In multilingual environments, only the translation for the current UI language is changed; other language versions remain unchanged. To change other languages, switch to the corresponding language and rename again.
- Save data record: Saves the values of the selected data record to the monitoring device's data cache. When "Retain recipe data" is enabled in the download settings, saved values remain effective.
- Delete data record: Deletes the selected data record.
- Export recipe: Exports the current recipe information and data and saves it in CSV format.
  Connect an external storage device (e.g., USB drive, SD card), then select a folder and enter a file name.

  - Exported files cannot be saved in the root directory.
  - Recipe export only supports exporting the multilingual content of the current device. To export other languages, enable them in "Runtime setting > Language and font" for the device before exporting.

  After confirmation, the current recipe data is exported to the specified location in CSV format, and a confirmation message appears in the recipe view.
- Read from PLC: Reads real-time element values from the PLC and updates them into the selected data record.
- Write to PLC: Writes the values of the selected data record to the PLC to make them effective.

Operations supported by the recipe view during online operation, such as modifying data or adding/deleting data records, do not affect the offline configuration of the recipe.

Modifying data records

Modifying data records is allowed only when **Activate edit mode** is checked, as shown below.

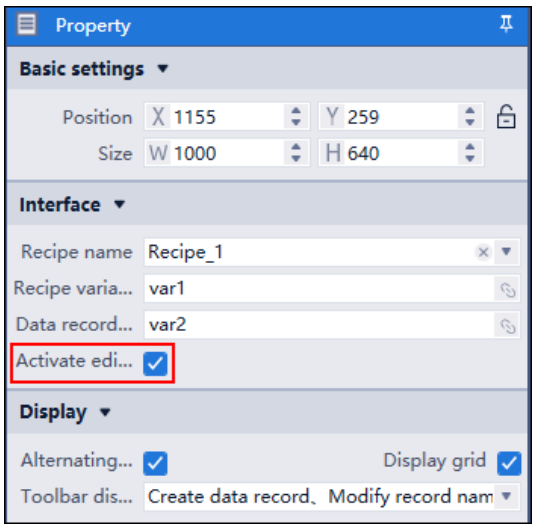
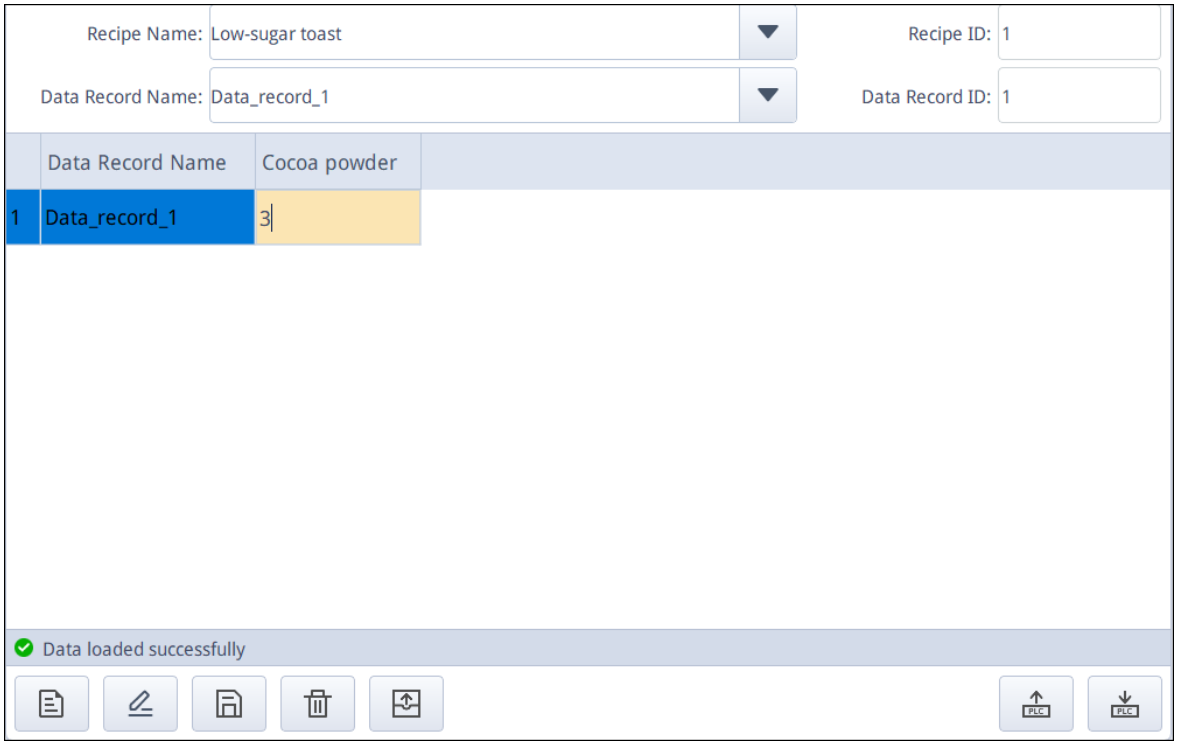


Figure 13-15-2-3 "Activate edit mode" property of a recipe

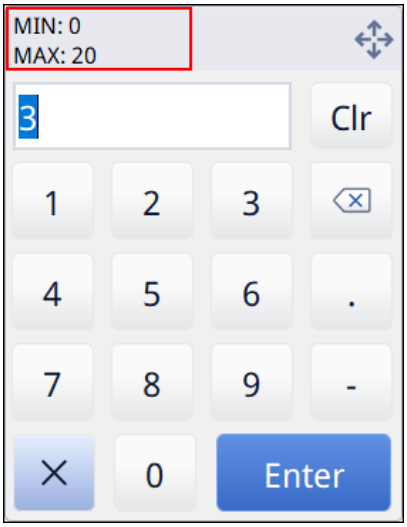
- Modifying element values

  1. Click the target cell and the cell background turns yellow, such as the **cocoa powder** content cell in the figure below.



  2. Write a new value.

     Enter the new value (within the allowed range) in the dialog box.



If the input value exceeds the limit, a red alarm is displayed.

When the input value is valid, the cell background turns green.

3. Click  to save data.
4. Click  to write the new value to the PLC.

Adding a data record

Click  to add a data record. A recipe supports up to 1000 data records.

Click  to save the added data records and click  to write them to the PLC to take effect.

Deleting a data record

Delete the selected data record.

Importing/Exporting recipe data

Recipe data export: Recipe data can be exported via the  icon or the script instruction (see 19.7.6 Exporting Recipe Data Records).

Ensure that element data has been saved. Values that have been modified but not yet saved (indicated by a green cell background) will be exported based on the pre-modification values.



Recipe export only supports exporting the multilingual content of the current device. To export other languages, enable them in "Runtime setting > Language and font" for the device before exporting.

Recipe data import: Imports the exported data to overwrite the current recipe content using the script instruction (see 19.7.7 Importing Recipe Data Records).

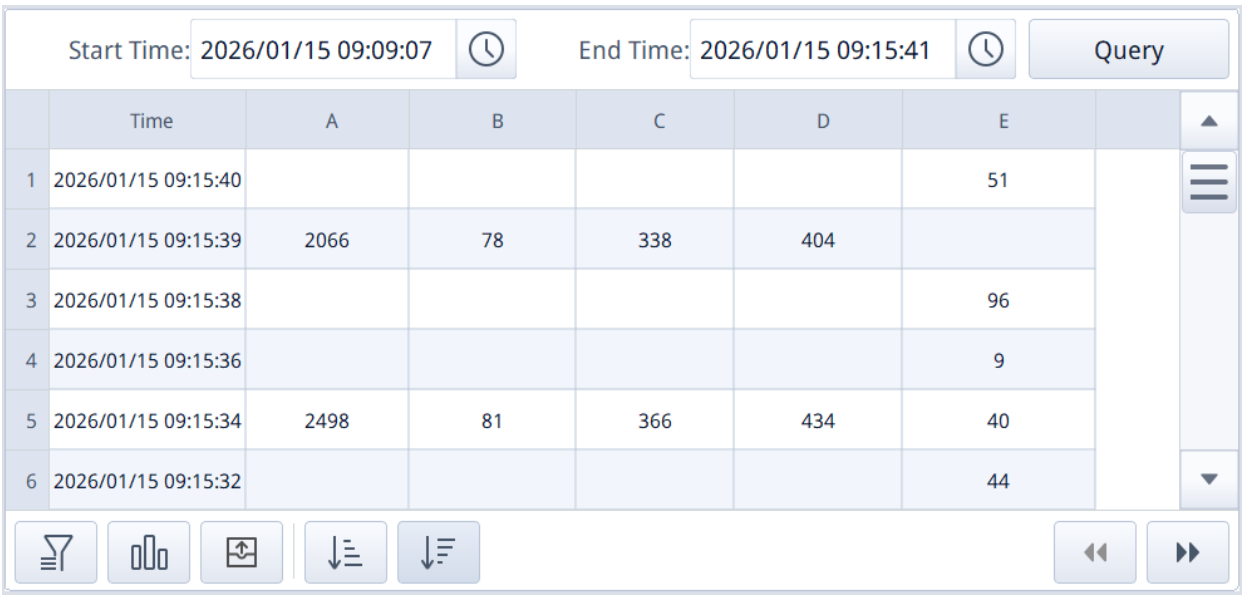
13.15.3 Online Historical Data Viewing

Users can view historical variable data online through controls 13.3.5.4.6 variable log view, 13.3.5.3.6 F(t) trend curve, and 13.3.5.3.7 F(x) trend curve. For details on their features and configuration, see the related sections. This section focuses on their online usage.

Variable log view

Variable log view displays detailed values of specified variables within a selected time period from the data record. For example, the figure below displays the values of variables A to E between 15:57 and 15:58 on November 22, 2024.

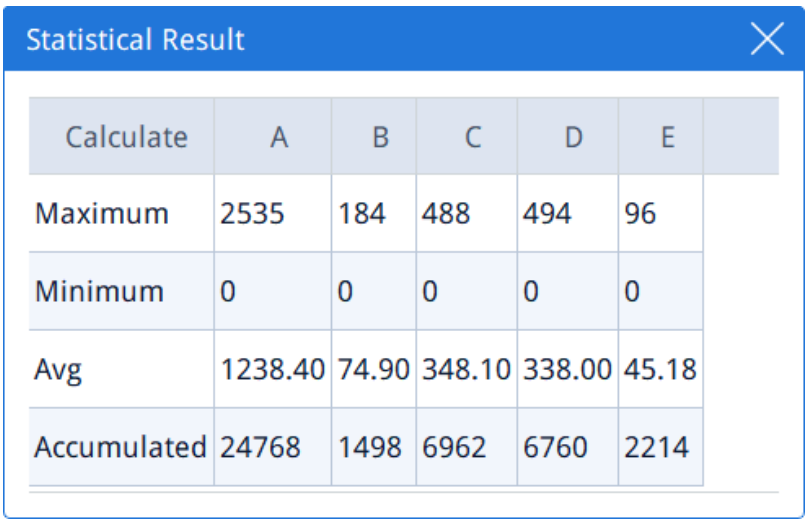
Variables A to D have a 5-second recording cycle, while E has a 2-second cycle. Since the table rows are organized by logging cycle, some cells may be blank, as shown in the figure below:



	Time	A	B	C	D	E
1	2026/01/15 09:15:40					51
2	2026/01/15 09:15:39	2066	78	338	404	
3	2026/01/15 09:15:38					96
4	2026/01/15 09:15:36					9
5	2026/01/15 09:15:34	2498	81	366	434	40
6	2026/01/15 09:15:32					44

: Filters the variables displayed in the view.

: Calculates the maximum/minimum, average, and cumulative values for integer and floating-point variables over the selected period (floating-point values default to two decimal places).



Calculate	A	B	C	D	E
Maximum	2535	184	488	494	96
Minimum	0	0	0	0	0
Avg	1238.40	74.90	348.10	338.00	45.18
Accumulated	24768	1498	6962	6760	2214

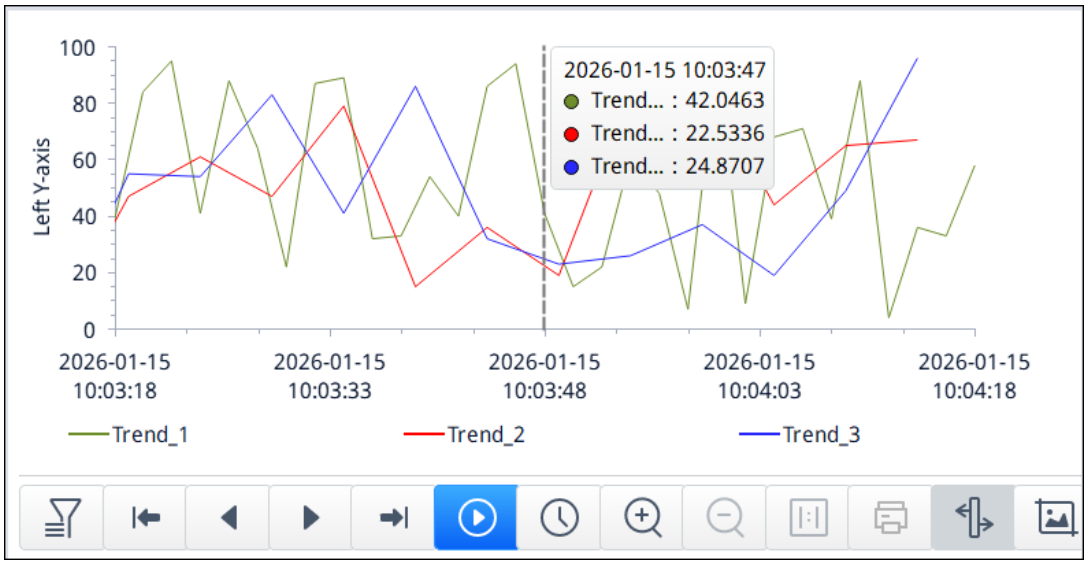
: Exports the current table data and statistics in CSV format. For related operations, refer to the data record file export in the recipe view.

: Sorts time from newest to oldest.

: Sorts time from oldest to newest.

F(t) trend curve

The F(t) trend curve displays the relationship between variable values and time. The figure below shows the trend of three variables between 17:11 and 17:12 on March 10, 2025.



F(t) trend curve example

: Filters the variables displayed; supports up to 16 trends.

: Moves the trend time to the first recorded value.

: Moves the trend time backward by one time period (set in view properties).

: Moves the trend time forward by one time period (set in view properties).

: Moves the trend time to the last recorded value.

: Activates real-time data display.

For example, click this icon and the view displays the real-time value of the variable. If the communication between the HMI and the PLC is interrupted, the real-time value acquisition fails, causing the X-axis time of the curve to stop refreshing. It resumes when communication and real-time value acquisition returns to normal.

: Sets the number of values or the time interval displayed in the view. Supports three configuration methods:

- Time range: Queries values within a specified duration from the start time. Range: 1 to 32767.
- Number of measuring points: Queries a specified number of data points from the start time. Range: 0 to 86400.
  For example, the figure below displays 20 measuring points backward from 17:11:41:

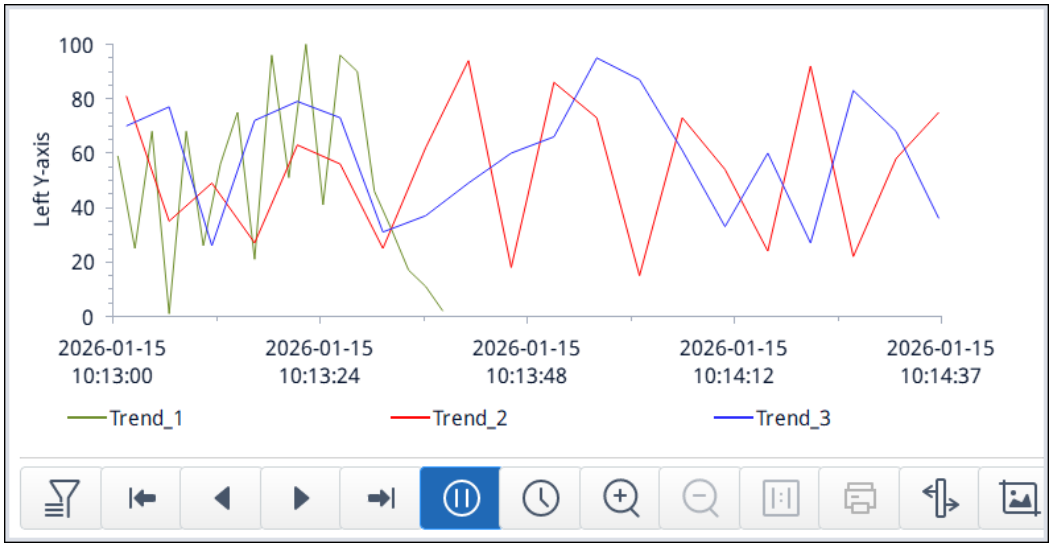


Figure 13-15-3-2 Displaying 20 measuring points

- Start and end time: Queries values within a custom time range.

: Zooms in the curve display area, with a maximum scale of 500%.

: Zooms out the curve display area, with a minimum scale of 100%.

: Returns the curve display area to normal.

: Visible only after offline configuration. Sets the visibility of scales in the view, such as the dashed line in the **F(t) trend curve example**.

The curve displays both real-time values and interpolated values. Interpolated values are calculated from adjacent two real-time values to ensure data continuity. For example, the interpolated value for Trend_3 lies on the line connecting the two points, as shown below.

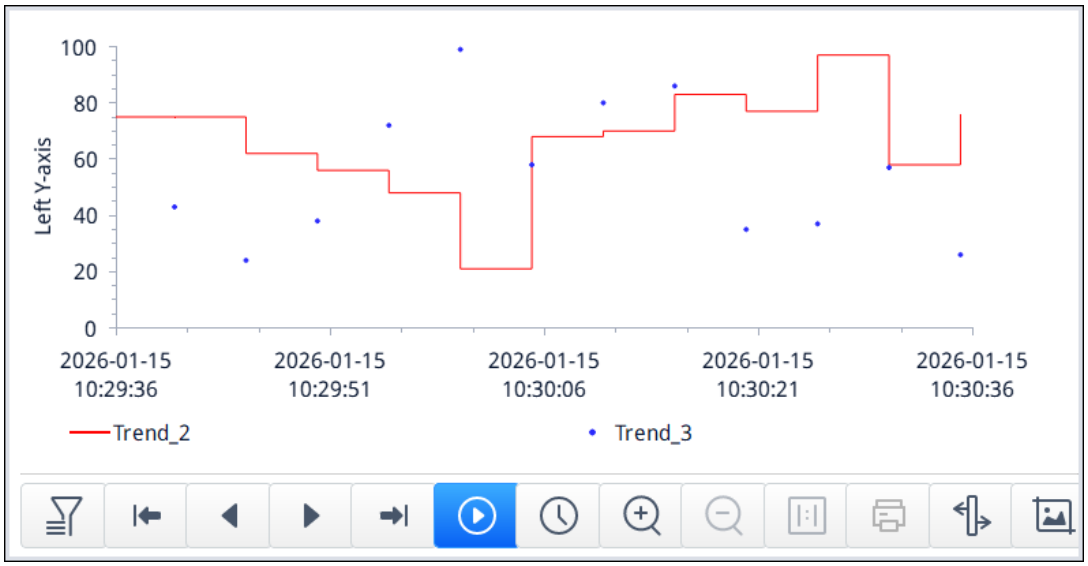


Figure 13-15-3-3 Interpolated value for Trend_3

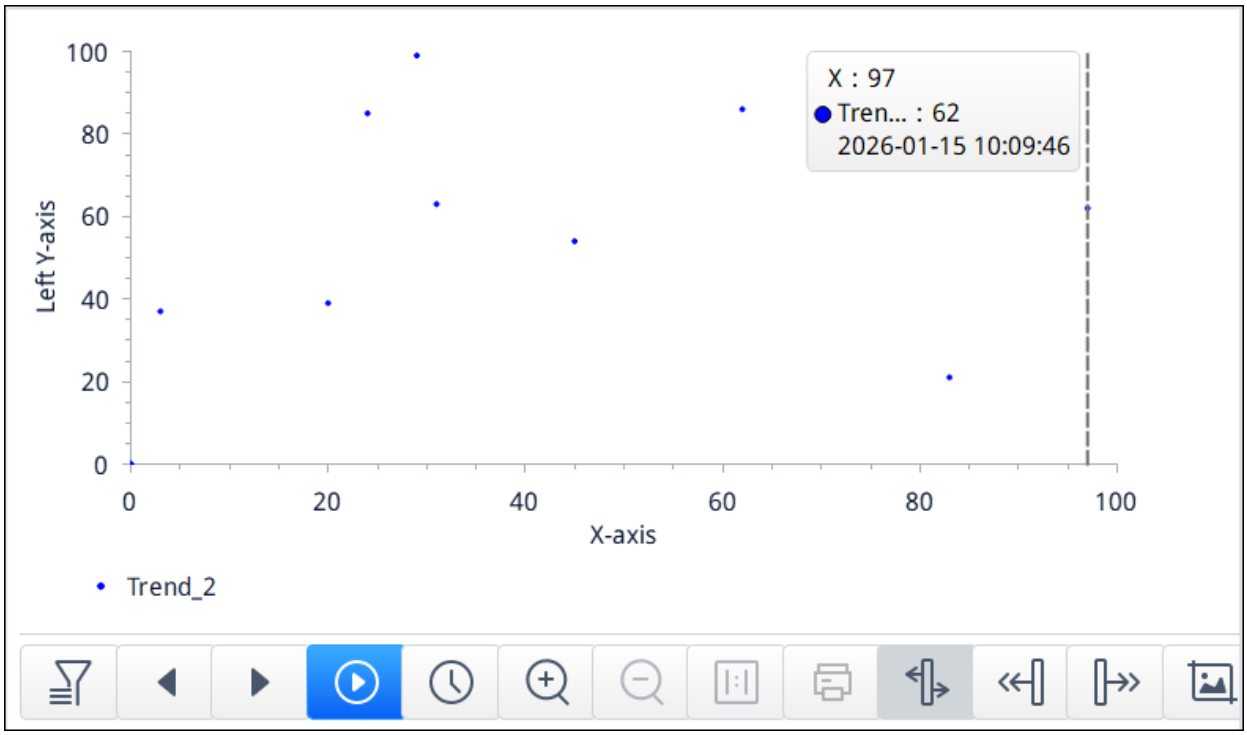
: Captures the current view and saves it as a PNG file named [screenshot_YYYY-MM-DD_HH-MM-SS-ms].png to the specified path. For detailed settings, see the **screenshot** property description of the 13.3.5.3.6 F(t) trend curve.

If stored on an SD card or USB device with full storage, the operation will fail.

F(x) trend curve

The F(x) trend curve displays data points on a Cartesian coordinate plane, showing the correlation between variables. Each point's position is determined by variable values.

The figure below shows the real-time trend of a group of variable values.



: Filters the trends displayed; supports up to 16 trends.

: Moves the trend time backward by one time period (set in view properties).

: Moves the trend time forward by one time period (set in view properties).

: Stops or starts updating data records.

: Sets the number of values or time intervals displayed in the view. For details, see above-mentioned descriptions of the **F(t) trend curve**.

: Zooms in the curve display area, with a maximum scale of 500%.

: Zooms out the curve display area, with a minimum scale of 100%.

: Returns the curve display area to normal.

: Shows or hides the ruler, such as the gray dashed line in the figure above. The ruler displays the value at the current position. For details, see above-mentioned descriptions of the **F(t) trend curve**.

: Moves the ruler to the left.

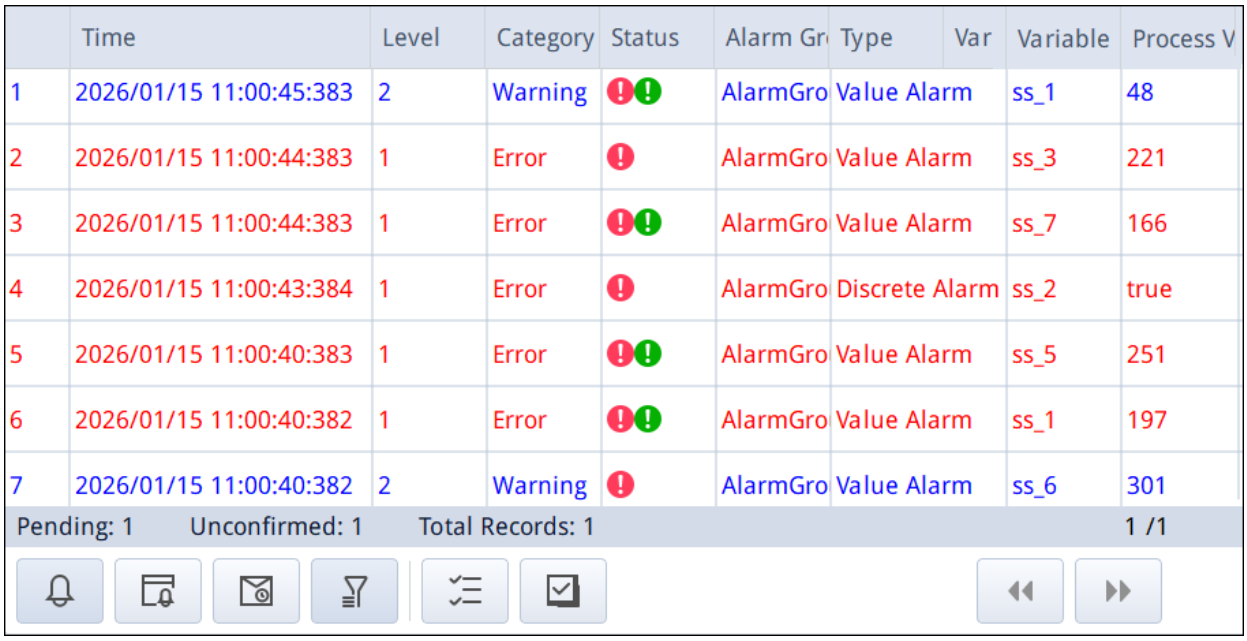
: Moves the ruler to the right.

: Captures the current view and saves it as a PNG file named [screenshot_YYYY-MM-DD_HH-MM-SS-ms].png to the specified path. For detailed settings, see the **screenshot** property description of the 13.3.5.3.6 F(t) Trend Curve.

If stored on an SD card or USB device with full storage, the operation will fail.

13.15.4 Online Alarm View Operations

The 13.3.5.4.1 alarm view configured on the screen is sent to the monitoring device via project download. The system displays generated alarms in the alarm view window, organizing them into different pages based on alarm time and category for easy access.



This section focuses on the online operations of the alarm view. For offline configuration, see 13.3.5.4.3 Recipe View.

The alarm view automatically displays the preset page during online operation. Preset values can be modified in the **"Data"** section on the **"Property"** tab.

The alarm view consists of three parts from top to bottom: alarm table, status bar, and buttons.

- Alarm table

  The alarm table is a collection of information that displays all generated alarms.

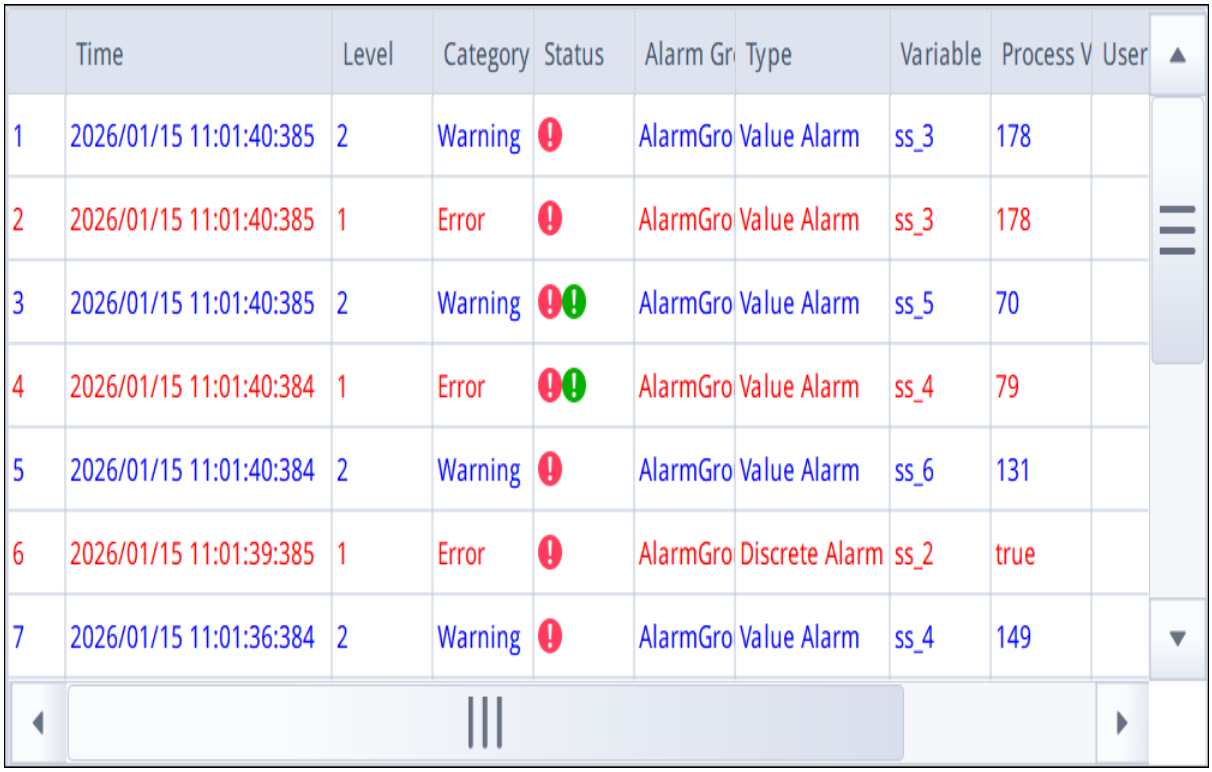


Figure 13-15-4-1 Alarm table

The alarm table header row can be configured in the "**Data**" section on the "**Property**" tab.
Time: Time when the alarm was triggered or the alarm event occurred. Events include alarm acknowledgment, alarm recovery, and user operations.
Level: Importance level of the alarm category. Lower numbers indicate higher priority. See 13.5.4 Alarm Category Configuration for offline settings.

Alarm category: There are four default categories, namely Error, Warning, User operation, and System. Users can rename category names and add types. See 13.5.4 Alarm Category Configuration for offline settings.

Alarm status: Status of Error and Alarm, including Active, Restored, and Confirmed.
Alarm group: Alarm group of the variable. For offline settings, see 13.2.5.3 Setting Variable Properties.
Type: Alarm type of the variable. For offline settings, see 13.5.1 Discrete Alarm Configuration and 13.5.2 Analog Alarm Configuration.
Variable: The variable that triggered the alarm.
Process value: The value at the moment of alarm triggering or event occurrence.
Username: Operator who triggered the "User operation" type alarm.
Text: Description of system alarms and user operations.

- Status bar
  Displays the current number of alarms and the total number of pages available.



- Buttons
  Buttons are displayed at the bottom of the alarm view window. A dark gray button indicates the corresponding operation is in progress.



: Real-time alarms. Displays errors and warnings triggered when variables meet preset alarm conditions. When a variable meets the preset conditions, an alarm of the corresponding category is triggered. The alarm remains in the real-time alarm page until it is both confirmed and restored. Supports filtering and acknowledgment.
: Real-time alarm events. Displays system alarms, user operations, and real-time alarm action events. The maximum display count is set in 13.5.8 Alarm Runtime Configuration.

Event recording rules:

- Each trigger of a system alarm or user operation generates a corresponding alarm event.
- Each status change of a real-time alarm generates a corresponding event.
- Each project download causes current alarms to be recorded again, even if previously recorded.

: Historical alarm events. Displays stored records for all alarm categories. Supports filtering. The maximum display count is set in 13.5.8 Alarm Runtime Configuration.
: Filters alarms by specified conditions for real-time alarms, real-time alarm events, and historical alarm events. For historical alarm events, the default filter time is set in 13.5.8 Alarm Runtime Configuration and can be customized in the filter dialog.

Once set, the filter only resets to default after a project download or system restart.
When the filter button is executed simultaneously with other buttons, it indicates that the displayed alarms are filtered results. For example, indicates that the content displayed on the

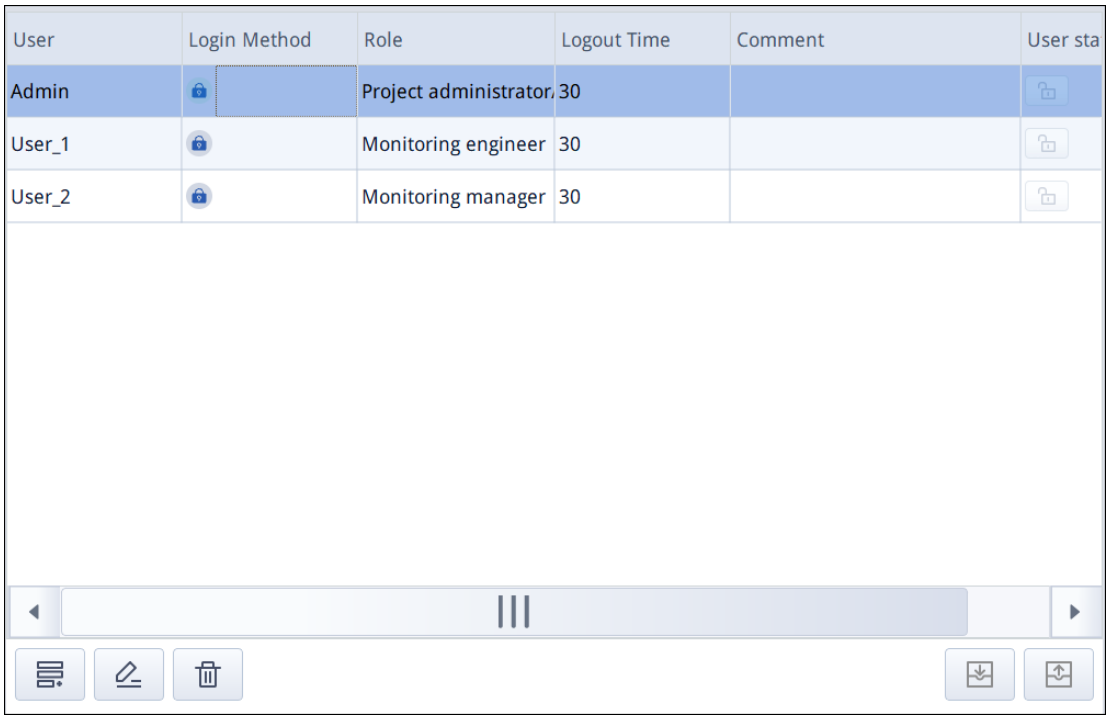
real-time alarm page is the result filtered according to the specified conditions.

: Confirms real-time alarm statuses one by one.

: Confirms all current real-time alarm statuses.

13.15.5 Online User View Operations

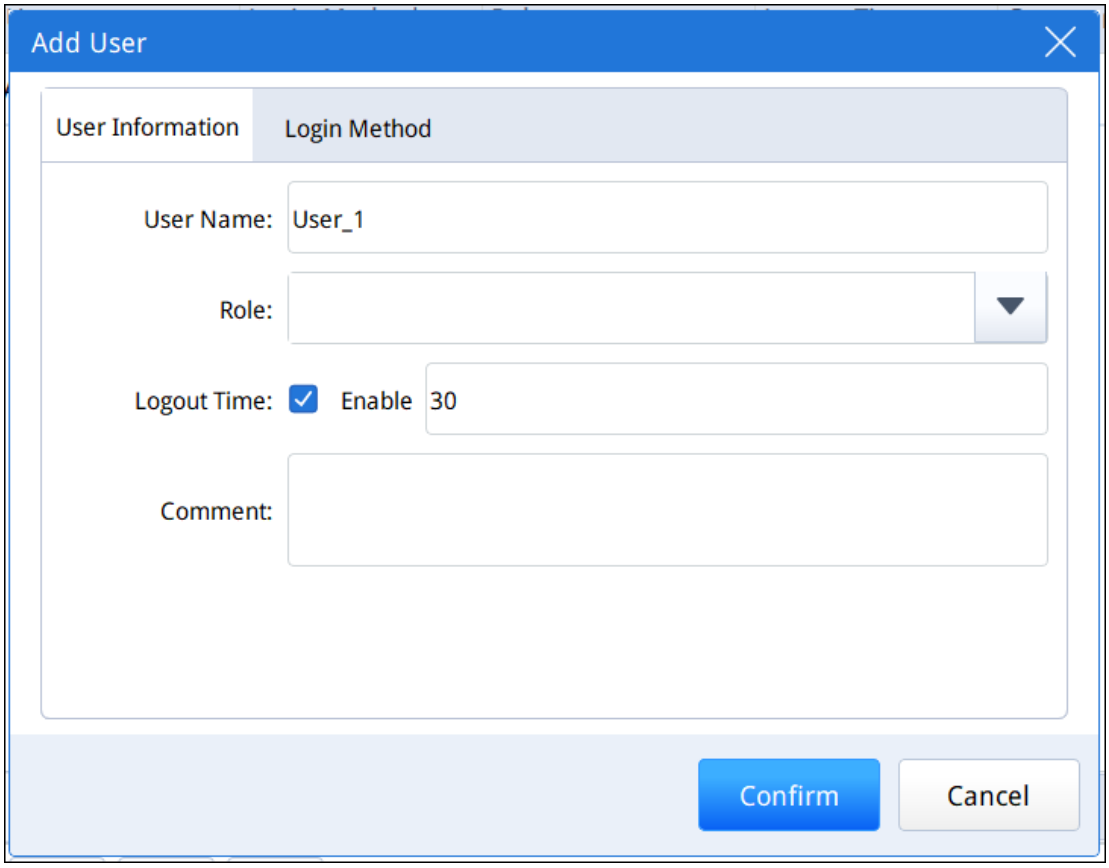
When the HMI device is running, administrators can add, delete, modify, and lock/unlock user accounts in the user view. For offline configuration, see 13.3.5.4.5 User View. It must be used in conjunction with the 13.3.5.4.4 User Login control to manage access permissions.



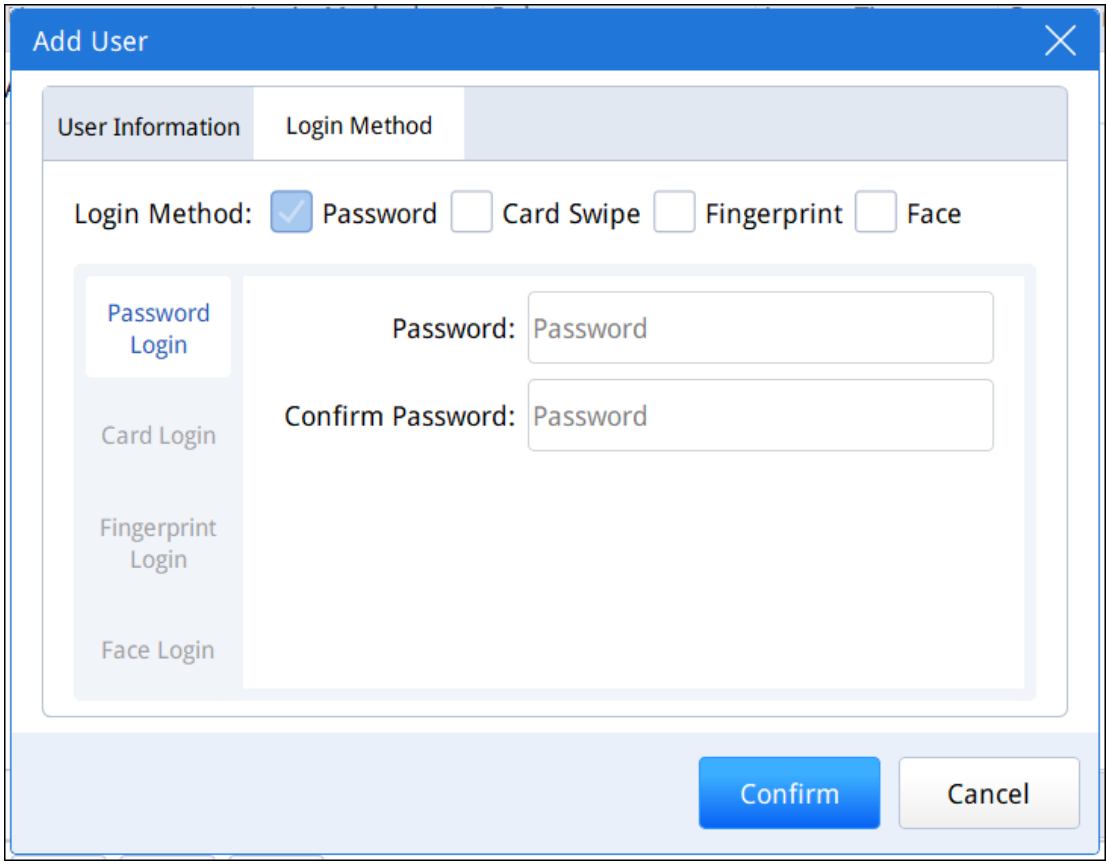
Adding a user

Administrators can create user accounts by clicking and configuring in the pop-up dialog.

- Configure user information.



  - Username: Displays the name of the new user. By default, it starts from User_1 and allows users to rename it. The maximum number of users per device is 50.
  - Role: Displays the role name assigned to the current device during offline configuration.
  - Logout time: Sets whether the logout function is enabled and sets the logout time. The default unit is minutes, with a default value of 30.
  - Comment: Displays user comment information.
- Configure login method.



- Password login: After selecting password login, the user password shall be set for login.
  Password storage uses AES-128 encryption, ensuring secure, reliable, and efficient protection of stored passwords.
- Card login: After selecting card login, swipe the card to input login information.
  To input card information, the card reader connection must be created, and the corresponding variable must be associated during offline configuration. Then, swipe the card to input the information.
- Fingerprint login: Not enabled.
- Face login: Not enabled.

Modifying a user

After logging in, administrators can edit all information of non-administrator users via . For administrator accounts, only limited fields such as comment and logout time can be edited.

Setting user status

Administrators can manually lock or unlock user accounts in the user view to control system access permissions.

- Operation scope
  - Administrators can lock or unlock non-administrator users.
  - Administrators can unlock each other but cannot lock each other.
- Lock duration setting

  Click the lock icon  in the user status column and specify the lock duration. Range: 0 to 3600s. 0s means permanently locked until manually unlocked by an administrator.

> **NOTE**
>
> To retain the lock status and remaining duration after a project download, ensure that "Retain user management data" is checked during download.

Deleting a user

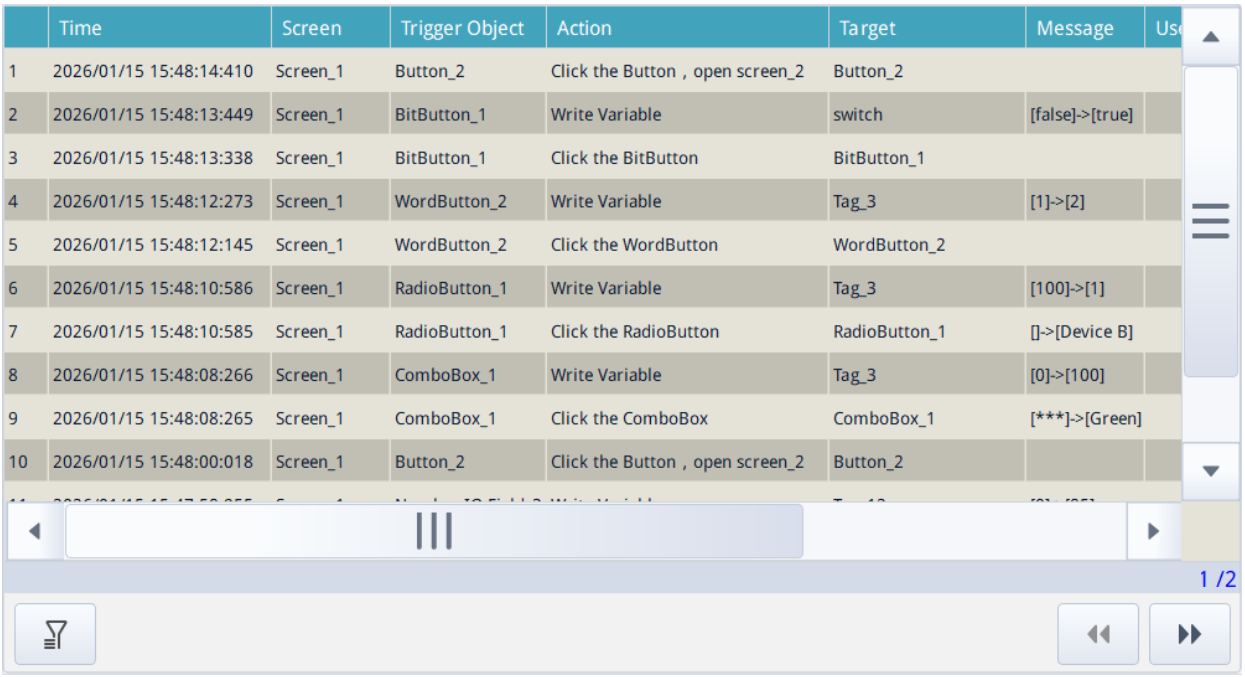
After logging in, administrators can delete non-administrator users by clicking .

13.15.6 Online Operation Record Operations

After the 13.3.5.4.8 Operation Records control is configured on the screen, recorded data is displayed in real time when specific operations occur during project runtime.

By default, recorded data is not automatically displayed when the control is online. It can be viewed by applying filters via  or by switching screens.

The control refreshes data every 2s by default. To change the refresh rate, adjust the "Refresh interval" parameter in the offline project.



	Time	Screen	Trigger Object	Action	Target	Message	Use
1	2026/01/15 15:48:14:410	Screen_1	Button_2	Click the Button , open screen_2	Button_2		
2	2026/01/15 15:48:13:449	Screen_1	BitButton_1	Write Variable	switch	[false]->[true]	
3	2026/01/15 15:48:13:338	Screen_1	BitButton_1	Click the BitButton	BitButton_1		
4	2026/01/15 15:48:12:273	Screen_1	WordButton_2	Write Variable	Tag_3	[1]->[2]	
5	2026/01/15 15:48:12:145	Screen_1	WordButton_2	Click the WordButton	WordButton_2		
6	2026/01/15 15:48:10:586	Screen_1	RadioButton_1	Write Variable	Tag_3	[100]->[1]	
7	2026/01/15 15:48:10:585	Screen_1	RadioButton_1	Click the RadioButton	RadioButton_1	[]->[Device B]	
8	2026/01/15 15:48:08:266	Screen_1	ComboBox_1	Write Variable	Tag_3	[0]->[100]	
9	2026/01/15 15:48:08:265	Screen_1	ComboBox_1	Click the ComboBox	ComboBox_1	[***]->[Green]	
10	2026/01/15 15:48:00:018	Screen_1	Button_2	Click the Button , open screen_2	Button_2		

1 /2

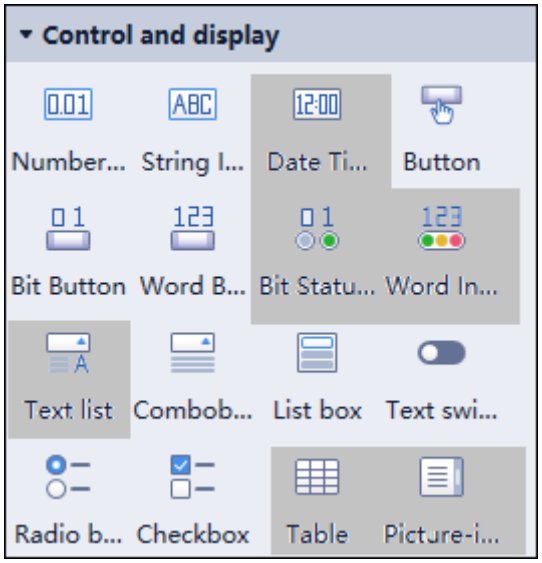
The number of columns and their titles can be customized. By default, the table contains 11 columns, as described below.

Time: Timestamp when the operation occurred. The default precision is millisecond level, which can be configured to second level.

Depending on the target object, the timestamp may refer to the time the control was operated, the function was executed, or the event was triggered.

Screen: Name of the screen containing the operated object.

Trigger object: Name of the operated object. Supported object types include task lists, scripts, controls (10 types shown in the figure below), records, and screens.



Action description: Description of the action, such as writing a variable, storage overflow, function execution, or exceeding a limit.

Action object: Target object of the action, such as a variable name or script name.

Information: Result of the action. For example, if a variable changes from 20 to 25, it is recorded as [20] -> [25]; if the setValue function changes from 1 to 3, it is recorded as setValue: [1] -> [3].

This field is recorded even if the value remains unchanged before and after the action.

Username: Name of the operator.

Device name: Name of the operating device.

Connection: When the action object is an external variable, displays the corresponding connection name.

Address: Storage address of the variable.

CAUTION

There is a 1s to 2s delay in saving operation records. Data is only saved after it appears in the control list. If the device is powered off before the data is displayed, the corresponding records will be lost.

13.15.7 Online System Diagnostics

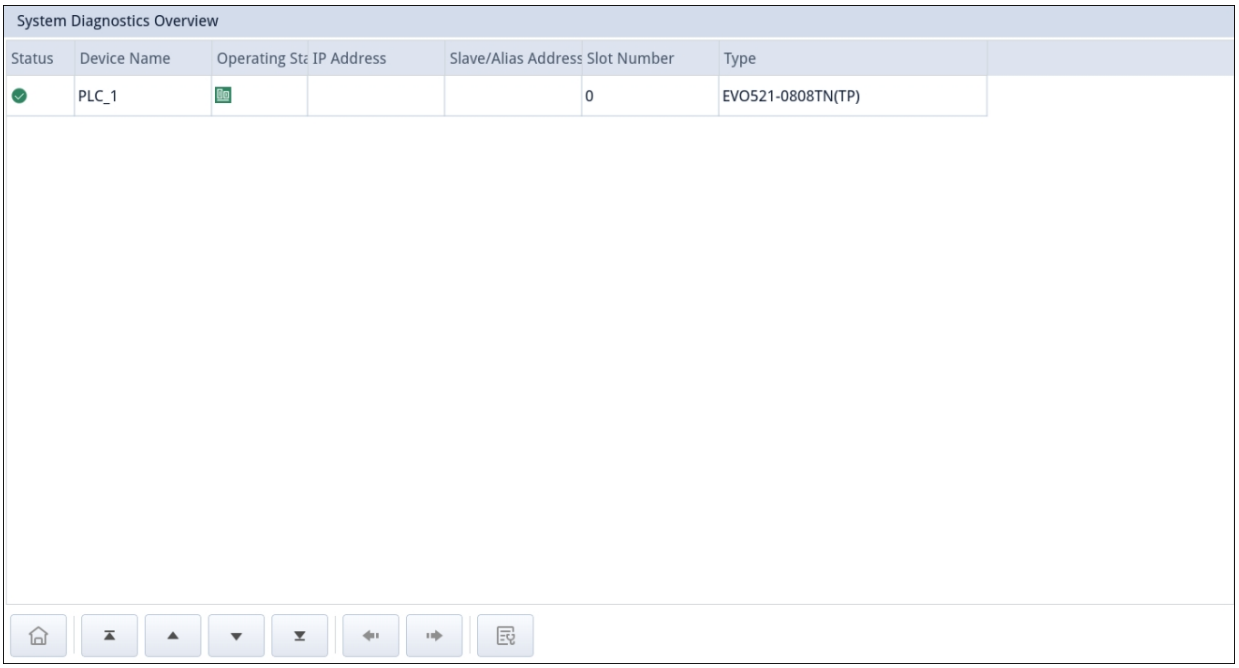
Prerequisites

- The PLC and HMI are in the same project in iFA software.
- "Inovance Technology, Ethernet/IP Tags protocol" is used for communication.
- The system diagnostics control can only run on physical devices and does not support simulation.

Online monitoring

When the screen containing the "System diagnostics" control is active, the control displays the system

diagnostic overview interface by default.

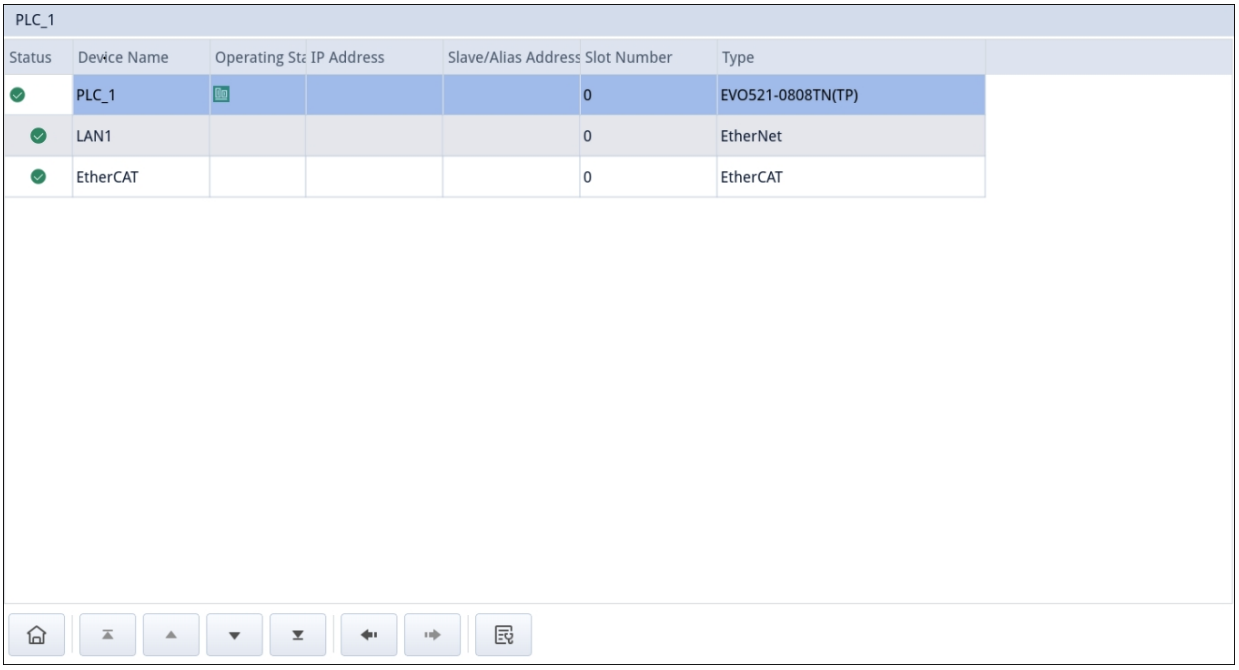


Item	Description
Control header	Displays the hierarchy of the device based on the current level and selected object. The hierarchy text is identical to the path name in the software.
Status	Displays the current device running status or the connection status between PLC and HMI. PLC statuses include: Normal, Fault, and Lower-level device fault.
Device name	Displays the name of the device as shown in the project tree.
Operation status	Displays the four PLC statuses: Operational: Stopped: Paused: Forced: This column is empty for other modules.
IP address	Displays the IP address or port connected between the PLC and the current HMI.
Slave/alias address	Displays the slave number or alias address of EtherCAT slave devices. Empty for non-EtherCAT slaves.
Slot number	Displays the slot number of the current module in the rack.
Type	Displays the device model.

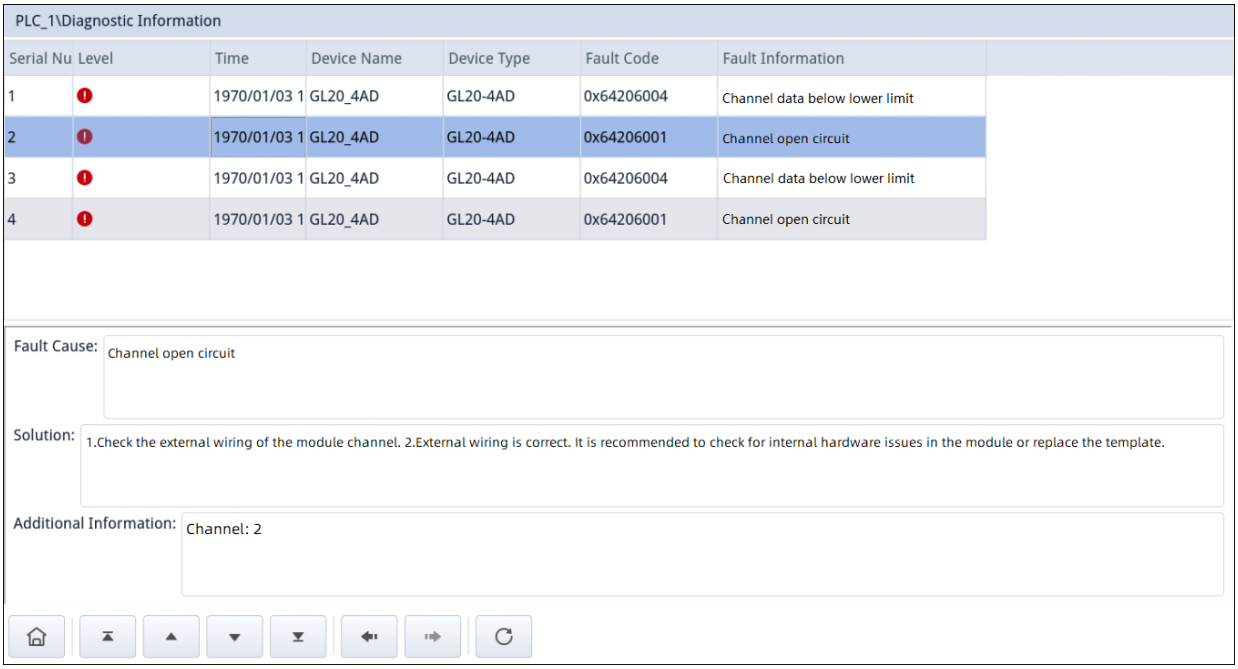
Button	Description
	Returns to the system diagnostics overview interface.

Button	Description
	Selects the first row.
	Selects the previous row.
	Selects the next row.
	Selects the last row.
	Returns to the previous interface.
	Moves to the next level interface. Available only when the PLC and HMI are well connected.
	Open PLC diagnostics

Select a PLC device row and click . View the connection status of the PLC device's LAN port and EtherCAT port.



Select a PLC device row and click . View the I/O channel diagnostic information of the PLC device.



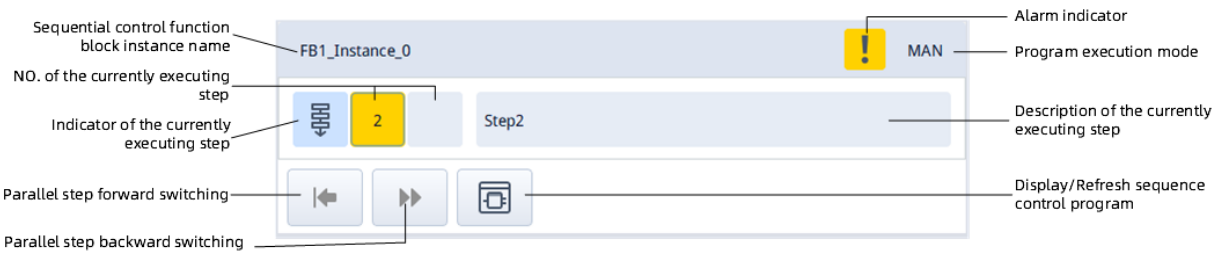
13.15.8 Online PLC Program Viewing

Sequential control programs can be displayed or hided on the interface, and the dynamic execution progress can be viewed in real time.

Prerequisites

- The SFC program has been compiled successfully.
- The 13.3.5.4.9 SFC view control and 13.3.5.4.10 code view control have been configured.

SFC view control interface

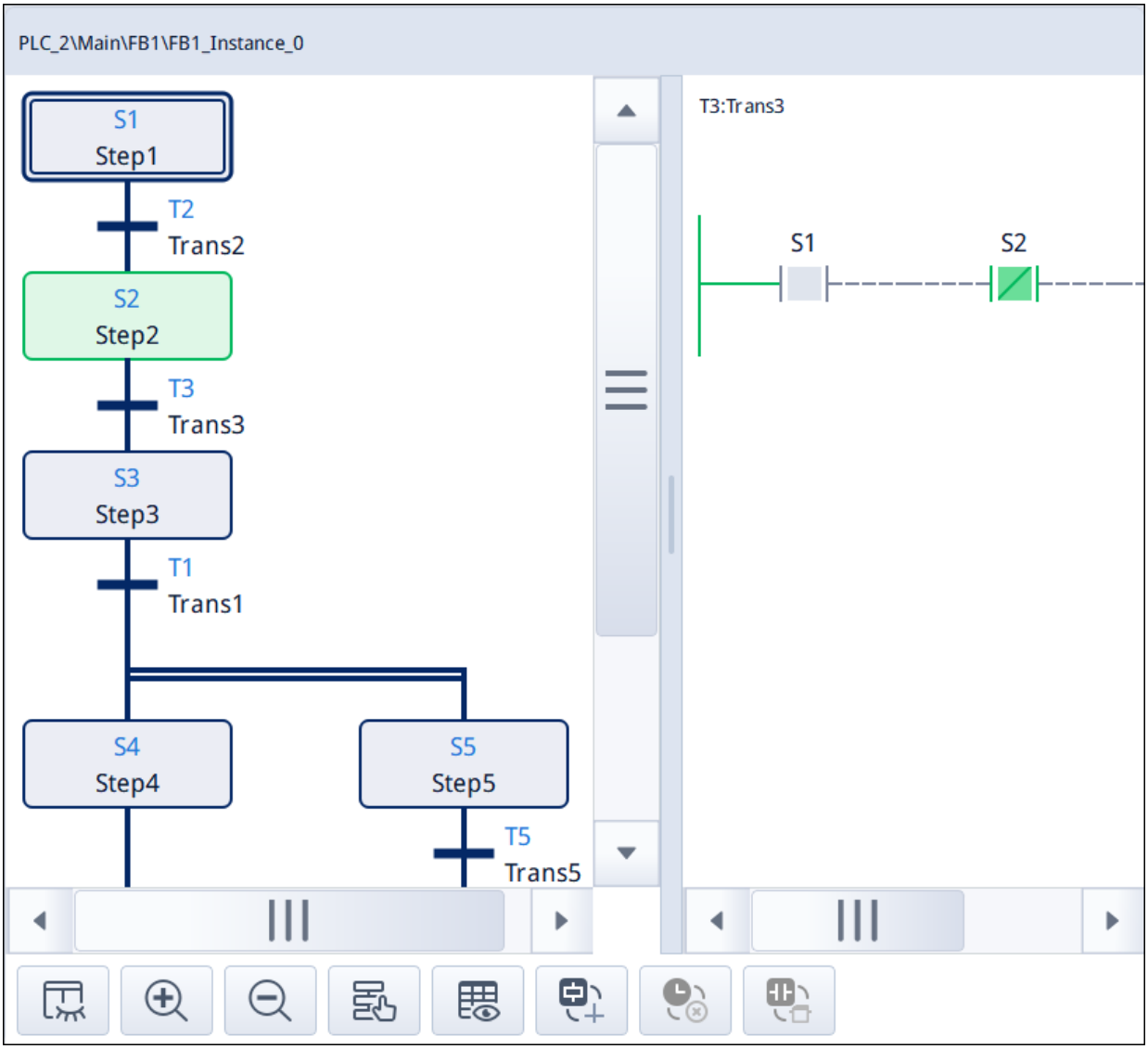


The interface is introduced below from top to bottom and left to right.

Item	Note
Sequential control function block instance name	An SFC view can display only one instance program.
Number of the currently executing step	Displays the number of the step being executed. If the interlock or monitor condition of the executing step is not satisfied, the number is highlighted in yellow. • In single-step mode, the corresponding step number is displayed in the second box. • In parallel branch mode, two step numbers are displayed in the second and third boxes. You can use the forward/backward icons to switch the displayed step numbers.

Item	Note
Indicator of the currently executing step	 : Initial step is being executed.  : Single process step is being executed.  : Parallel branch is being executed.
Parallel step forward/ backward switching	Switches the displayed step number during parallel branch execution.
Alarm indicator	Displayed when the interlock or monitor condition of the current step is satisfied.
Program execution mode	MAN: Steps are executed via the commissioning panel, independent of transition conditions. AUTO (default): Automatically moves to the next step when the transition condition is satisfied. TOP (Semi-auto - transition conditions ignored): Moves to the next step when the transition condition is satisfied or a rising edge of the T_PUSH parameter is triggered. TAP (Semi-auto - transition condition-based): Moves to the next step when the transition condition is satisfied and a rising edge of the T_PUSH parameter is triggered.
Description of the currently executing step	Description of the active step.
Display/Refresh sequence control program	Displays or refreshes the sequential program in the code view control.

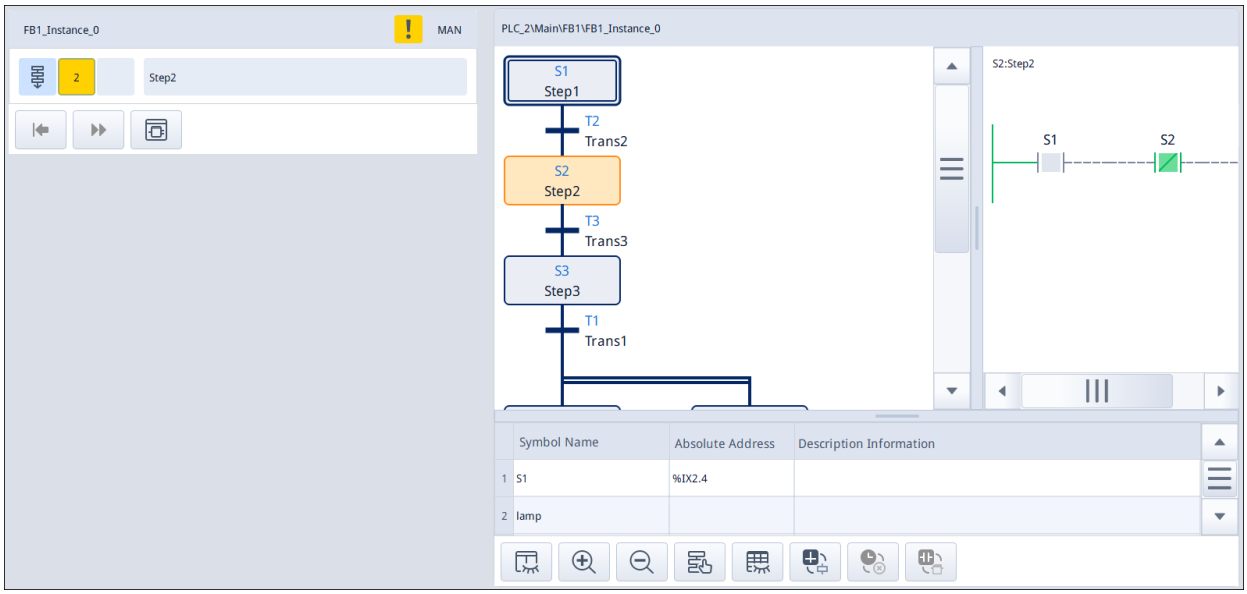
Code view control interface



Item	Description
	Opens the action of the current step or transition.
	Zooms in the program display scale.
	Zooms out the program display scale.
	Manual/Automatic switching: In manual mode, users can view the internal logic of each step/transition regardless of whether the step is active. This does not affect PLC operations.
	Displays or hides the symbol table. Click to display internal variable information of the currently executing transitions in a tabular format.
	In manual mode, the displayed content switches between steps and transitions.
	Reserved, not enabled.
	Reserved, not enabled.

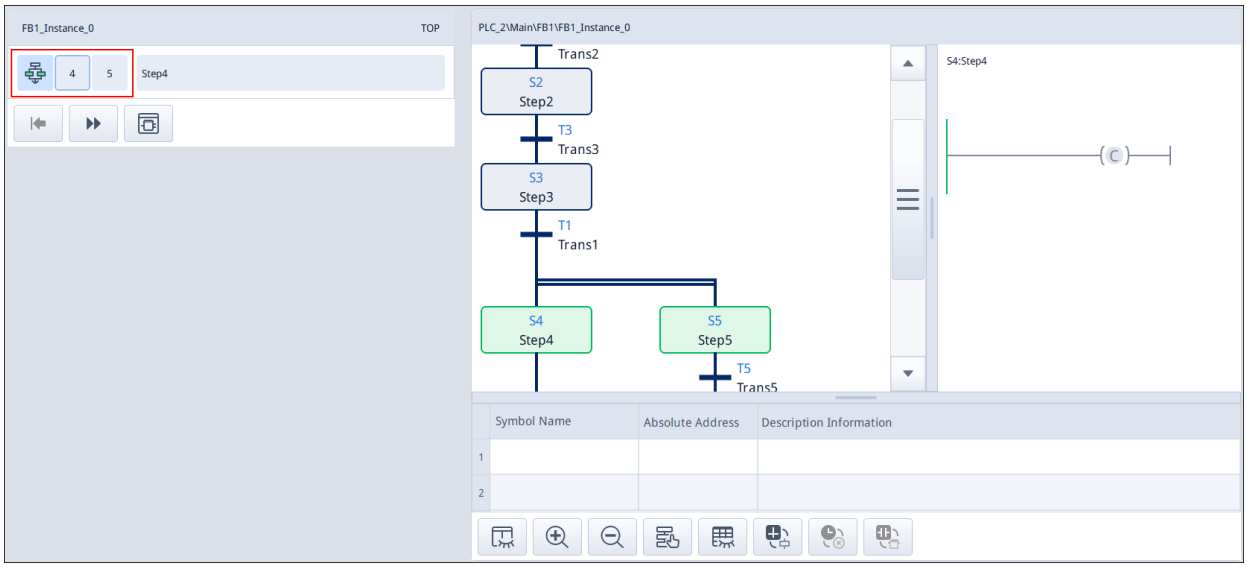
Combined usage of SFC view and code view

Click  in the SFC view to open the sequence control program in code view. The SFC view displays the overall execution status of the sequential program, while the code view displays its internal logics.



The SFC view connects to the PLC via an instance variable. Once connected, it displays the real-time execution status of the sequential program, including the number of currently executing step, step description, and commissioning mode. The code view is used to view the execution status of each step/transition and the program execution process, facilitating program commissioning and diagnosis.

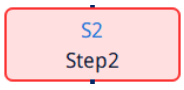
When executing parallel branches, the SFC view displays information for up to two parallel steps simultaneously. The effect is shown below:



Step status in the sequence control program:



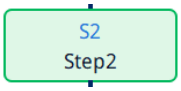
: Inactive status.



: Monitoring alarm status. An alarm is generated when the monitoring condition is met.



: Alarm status. The step enters an interlock alarm status when the interlock condition is not met, or when both interlock and monitoring alarms occur simultaneously.



: Active status.

13.16 Monitoring Simulation Commissioning

- 13.16.1 Simulation Data Storage
- 13.16.2 Simulating an HMI
- 13.16.3 Simulating a Variable
- 13.16.4 Co-simulation Between PLC and HMI

13.16.1 Simulation Data Storage

During simulation, certain scripts, controls (such as data import/export functions and dedicated controls), and external files required for control operation (such as video files) must be placed in the designated simulation resource folder. Only then can the simulator successfully recognize and call them, ensuring proper functionality of the related features.

Storage location

Root directory: The simulation resource folder named HMISimulation is stored in root directory of the iFA Evolution installation.

For example, if iFA Evolution is installed on the D drive, the simulation resource folder storage path is D:\HMISimulation.

HMISimulation folder structure

This folder contains four subfolders, each corresponding to a different storage location. The stored data objects are listed in the table below.

Subfolder	Storage	Script	Control	Data Record	Alarm Record	Operation Record
Local	Local disk	Functions for data import/ export only	File reading: such as multimedia and PDF Screenshots: F(t) trends and F(x) trends	Supported	Supported	Supported
SD	SD card					
USB1	USB equipment					
USB2						

13.16.2 Simulating an HMI

Single HMI simulation is suitable for initial HMI screen commissioning and testing. It focuses on verifying whether controls on the screen correctly display variable values and respond to interactions. It does not

require connection to a physical device, and simulation data is sourced from the 13.16.3 Simulating a Variable.

Limits

- Historical data and historical alarm-related functions are not supported in simulation.
- The maximum continuous runtime for each simulation is two hours.

Requirements

- Ensure that controls on the screen are correctly associated with variables.
- Ensure that the project compiles without errors.
- Each simulation supports only one device.

Procedure

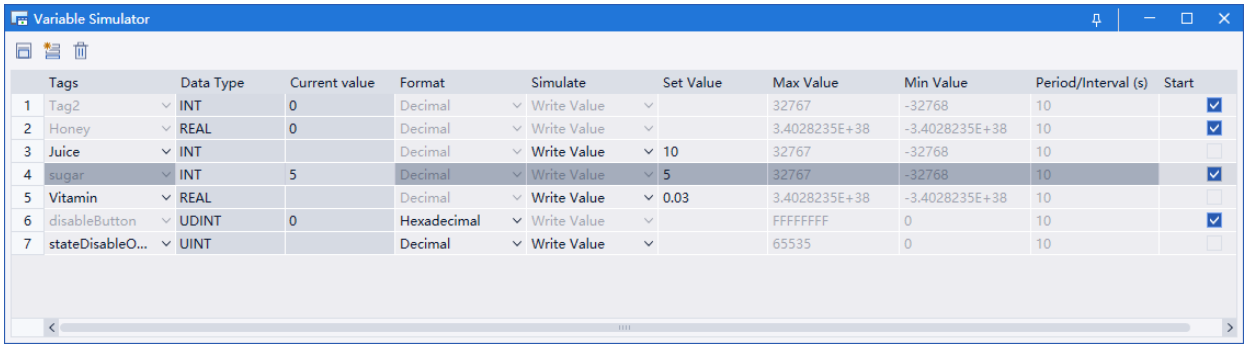
1. Start simulation.
   Click the simulation icon and it automatically changes to . In the meantime, the variable simulator and the graphical runtime system are opened. The window size of the graphical runtime system adjusts to the actual HMI screen size and displays the startup screen by default.
2. Simulate variables.
   Add the required variables in the variable simulator and set the simulation type and value range for each variable.
3. Adjust the screen.
   Check the values and interaction effects on the screen to verify normal functionality.
4. Exit simulation.
   Click to exit the current simulation. The configuration settings in the variable simulator can be saved for reuse in subsequent simulations.

13.16.3 Simulating a Variable

Use the variable simulator to simulate a variable. It is a tool for simulating the real-time values of internal and external variables. It synchronizes variable values to the screen in real time and retains runtime values when specific conditions are met.

Start

The variable simulator cannot be started independently; it must be launched via the simulation command .



- Variable: Simulation objects. A maximum of 500 variables can be added.
  Variables are displayed in a tiled layout, without using a tree structure to represent structures or arrays. Only data that exists in the variable table at the time the simulator is started can be selected; new data in the variable table is not synchronized after the simulator starts.
- Data type: The data type of the variable. This field cannot be edited.
- Current value: The current simulated value. This field cannot be edited.
- Format: The variable value is simulated in the specified format. Note:
  - BCD code (8421 BCD): A binary encoding used to represent decimal numbers.
  - Data types supporting BCD format: UINT, UDINT, ULINT, and USINT.
  - Data types for which the format cannot be modified: STRING, WSTRING, BOOL, REAL, LREAL, DATETIME, INT, DINT, LINT, and SINT.
- Simulate: Sets the simulation type for the value.
  - Write value: Manually sets the simulated value of the variable.
  - Random: Randomly changes the value between a defined minimum and maximum.
  - Increment: Automatically increases from the set minimum to the maximum; the step size can be configured.
  - Decrement: Automatically decreases from the set maximum to the minimum; the step size can be configured.
  - Shift: Applicable to unsigned integer data types. Shifts bit by bit upward from bit 0, cycling repeatedly.
  - Sine: Generates a periodic sine-wave variation between the maximum and minimum values.
- Set value: Sets the corresponding value based on the selected simulation type.
- Max./Min. value: Sets the value range for the variable.
- Period/Interval: Sets the period for value changes. The unit is seconds, with precision up to three decimal places.
- Start: Enables or disables the simulation for the current variable. Once enabled, the variable cannot be changed.

Simulation data processing

When simulating the same device (even after disconnecting), as long as the device has not been replaced, the system retains and handles data as follows:

- Data retention at power failure
  When the simulation restarts, the system retains the power failure data of variables defined in the previous simulation, ensuring continuity and consistency.

- Retention of data/alarm records
  All data/alarm records from previous simulations are retained.
- Recipe configuration data management
  If the recipe configuration data has not been modified, the data records from the previous simulation are retained.
  If the recipe configuration data has been modified, the system updates and displays the latest configuration data.
- User permission data management
  If the user permission data for the device has not been modified, the user permission data from the previous simulation is retained.
  If the user permission data has been modified, the system updates and displays the latest user permission information.

13.16.4 Co-simulation Between PLC and HMI

Co-simulation between PLC and HMI uses the PLCSIM tool and HMI simulation components for communication function commissioning between HMI and PLC without physical device connections. It focuses on verifying variable interaction and control logic between the two. The following are the steps to set the combined application simulation:

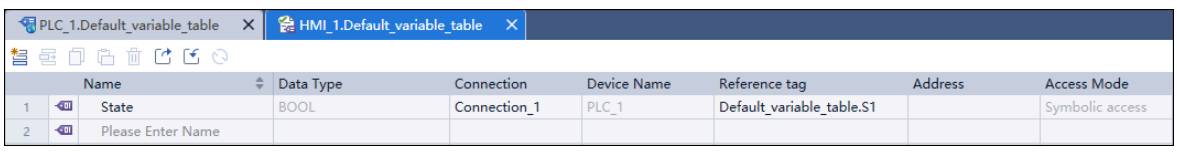
Requirements

- Ensure that the simulation commissioning component for HMI and PLC are installed in the iFA software.
- Configure the PLC and HMI devices within the same project.

Procedure

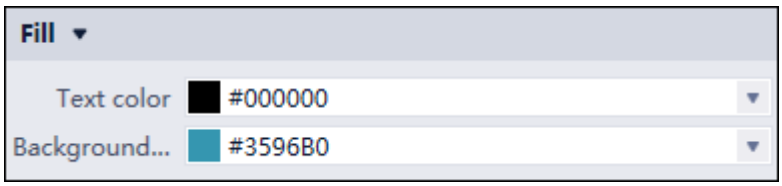
The following steps use one variable communicating and being written between the PLC and HMI to illustrate the simulation procedure:

1. Configure communication variables.
   a. Add a variable to the PLC device. For example, add S1 to the default variable table with a BOOL type.
   b. Set variable S1 as a public variable. For example, set the "Network public" property to "Read/Write."
   c. Add a variable to the HMI device. For example, add State to the default variable table with a BOOL type.
   d. Establish communication connection.
      Associate variable State with S1, and a connection between PLC and HMI is automatically established, named Connection_1.



   e. Configure HMI screen.
      Add controls to the HMI screen and associate them with variable State. For example, add a text switch.

      To facilitate identification, you can set a new style for the ON state, as shown in the following figure.

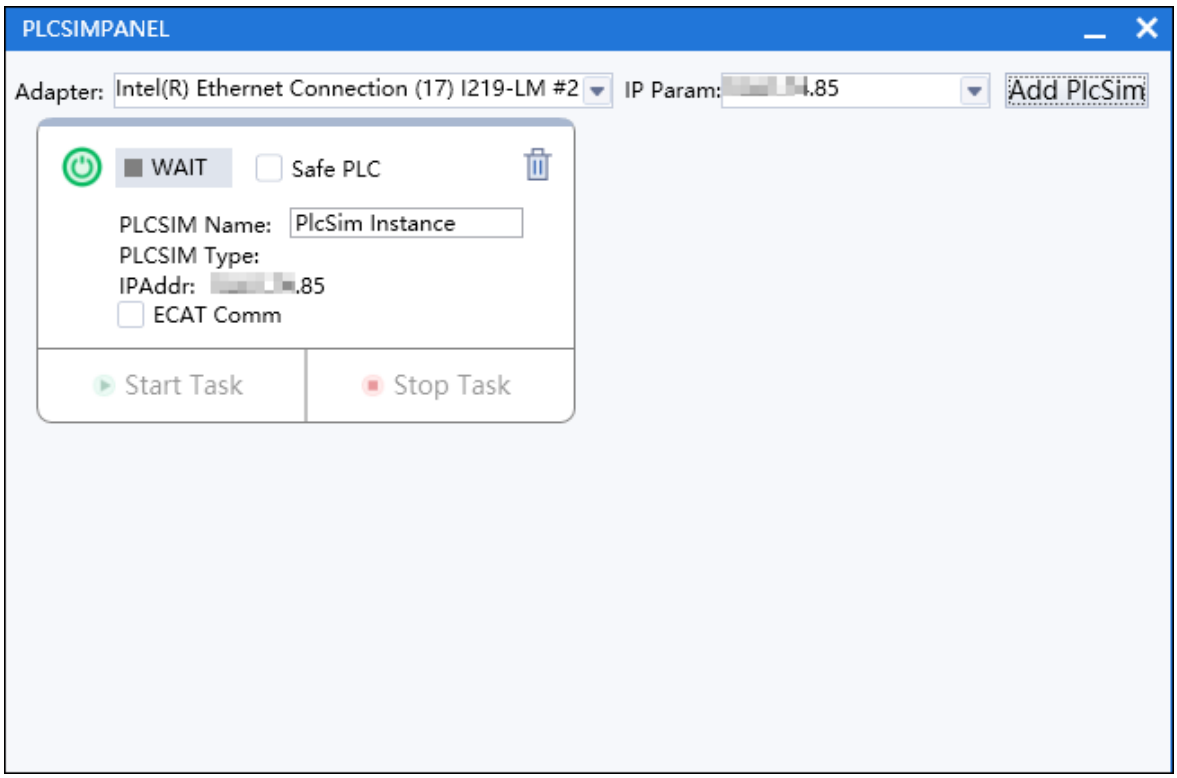


f. Compile the project.
Compile the PLC and the HMI project, and ensure there are no errors.

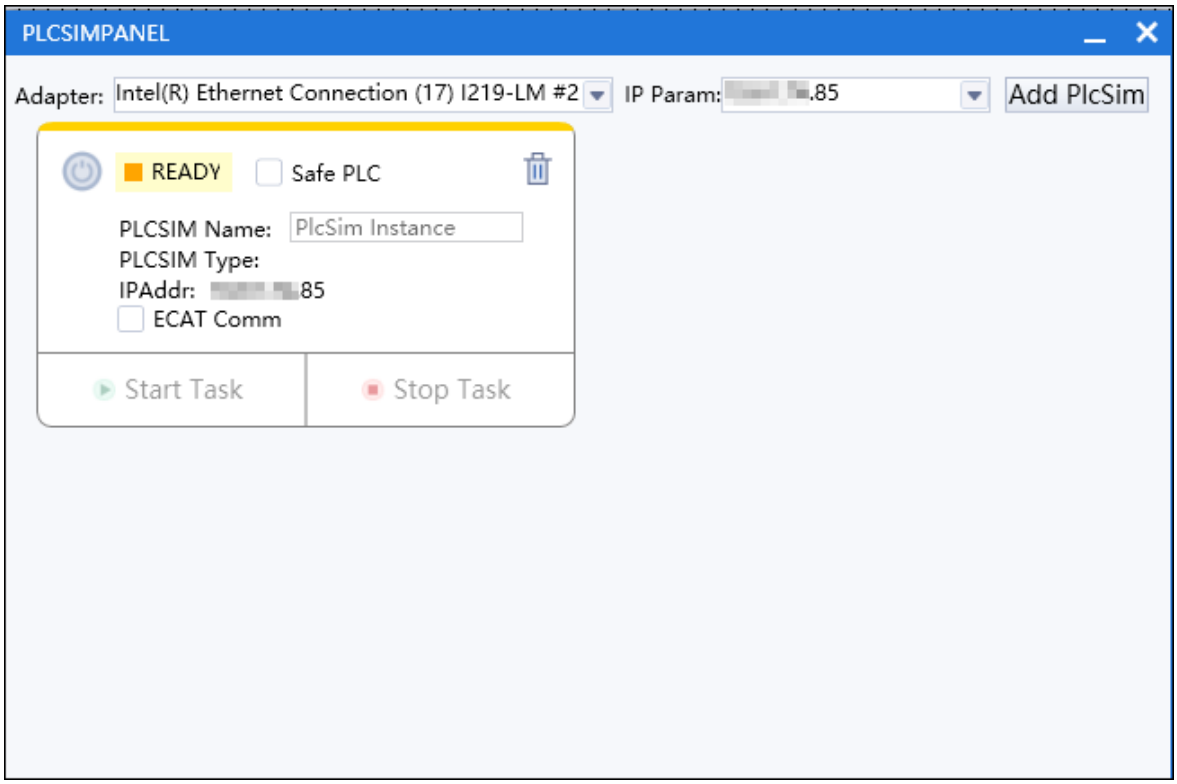
2. Start PLC device simulation.

a. Double-click the PLC device node in the project tree, then click . The button status is displayed as , and the PLCSIM control operation panel pops up.

b. Enter the current computer's IP address, select "Add Plcsimulation," and create a simulation instance for the PLC device.

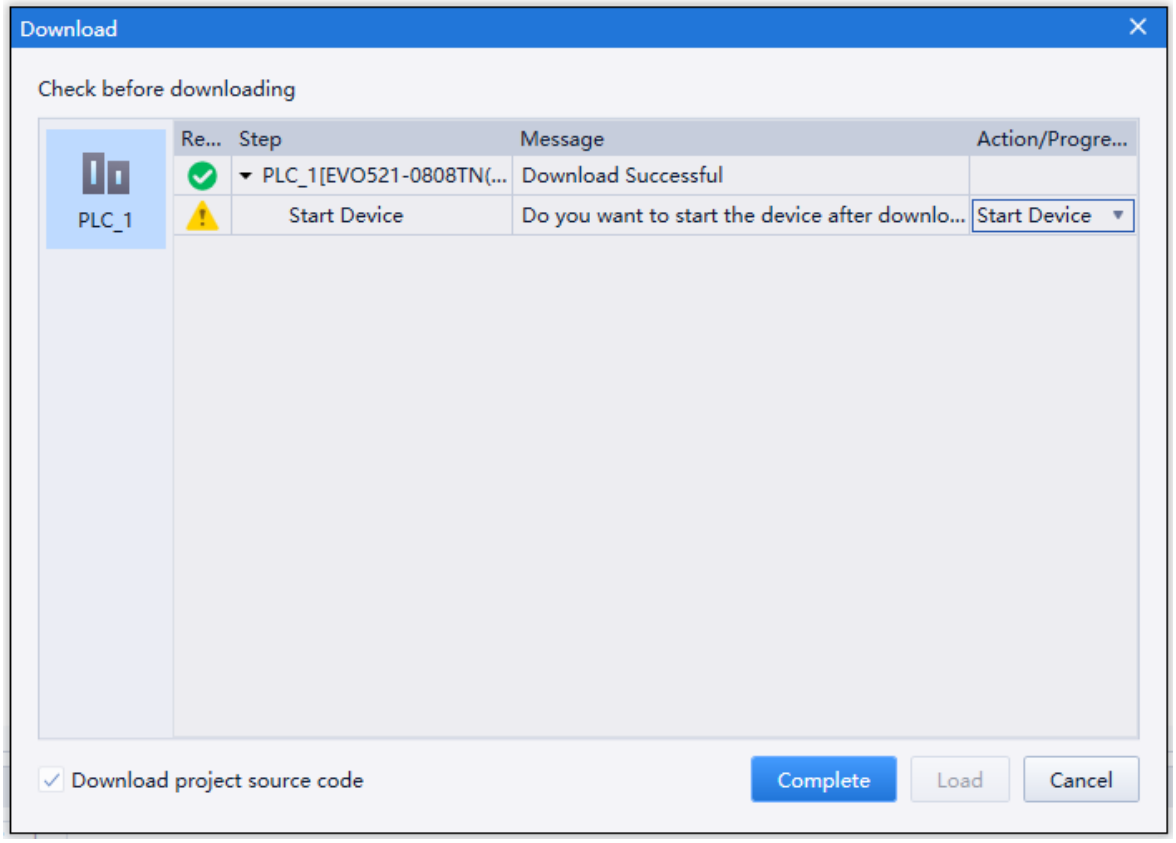


c. Click to activate the PLC simulation environment.



d. Click Online, and select the PLCSIM device with the same IP in the scan results.

e. Download and start the PLC device.

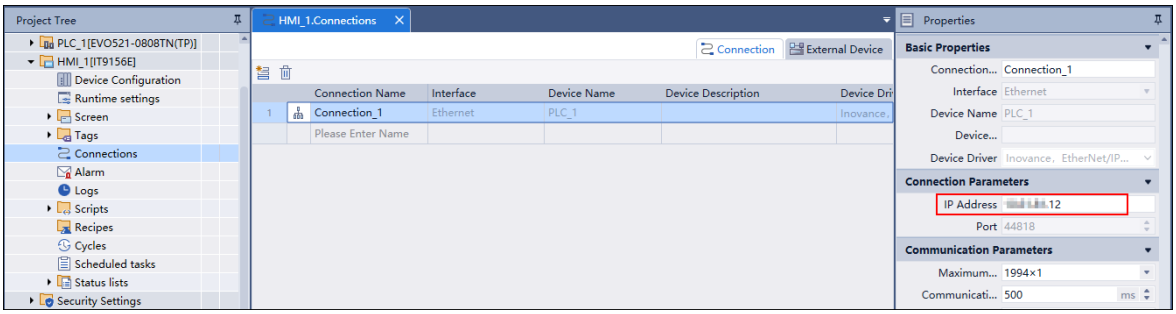


At this point, the PLC device enters simulation mode and starts running.

3. Start HMI device simulation.

a. Set the HMI communication connection address.

Double-click the HMI device's connection node in the project tree, and ensure the connection IP matches the IP in the PLCSIM control panel (the IP set in step 7).

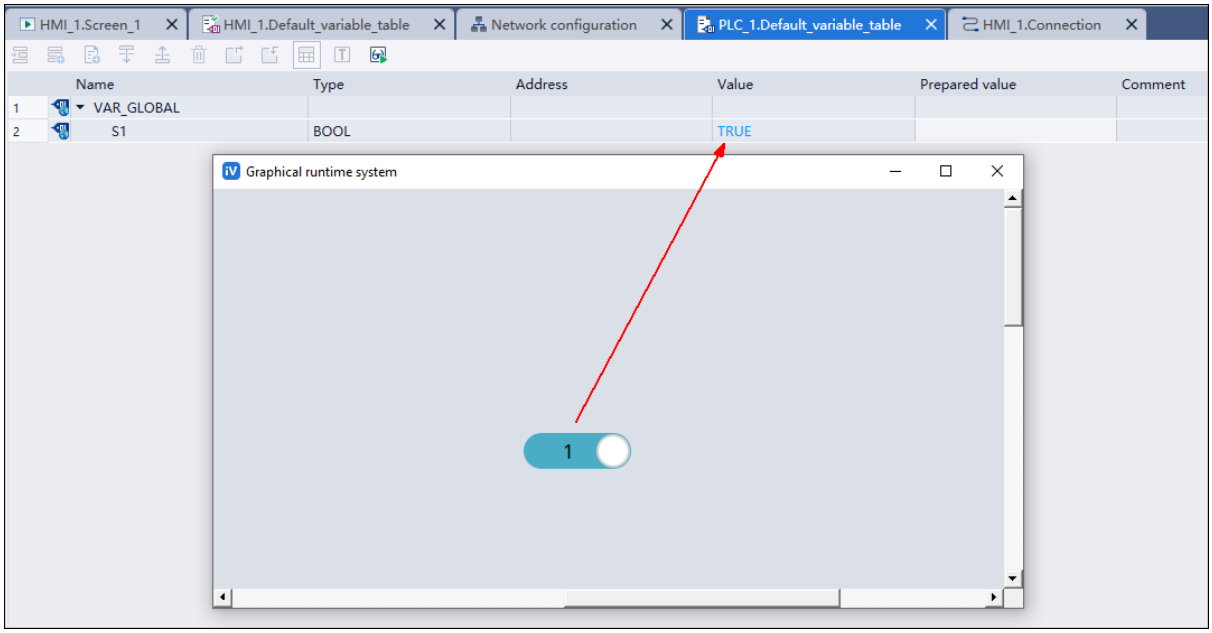


b. Start HMI simulation mode.

Click . The button is displayed as , and the "Graphical runtime system" window opens.

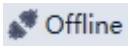
4. Perform co-simulation commissioning.

Change the switch status in the "Graphical runtime system" window and observe the value change of variable S1 in the PLC device.



5. Exit co-simulation mode.

   HMI device: Right-click the blank area of the "Graphical runtime system" window, select "Exit", and then click  to exit the simulation mode.

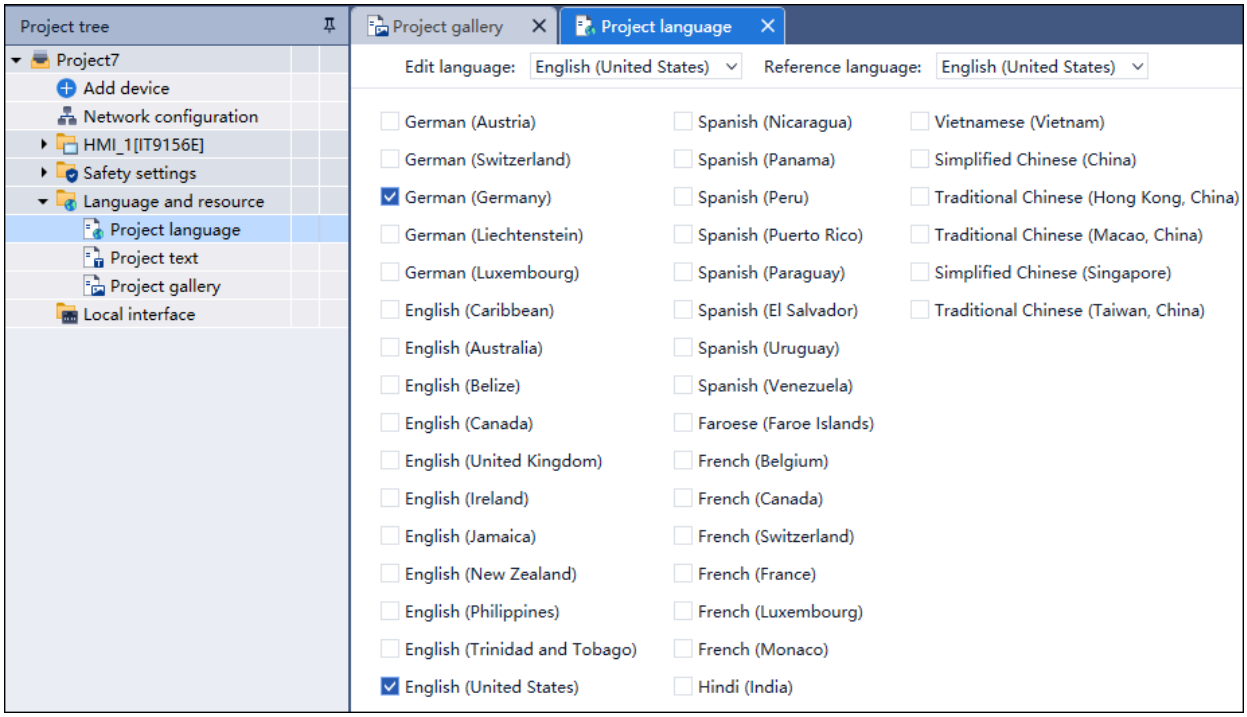
   PLC device: Select sequentially 

and  to exit the simulation mode.

13.17 Multilingual Settings

13.17.1 Multilingual Overview

Project language



- Edit language: Select the language used for the currently edited object. When entering text, it is always created in the edit language.

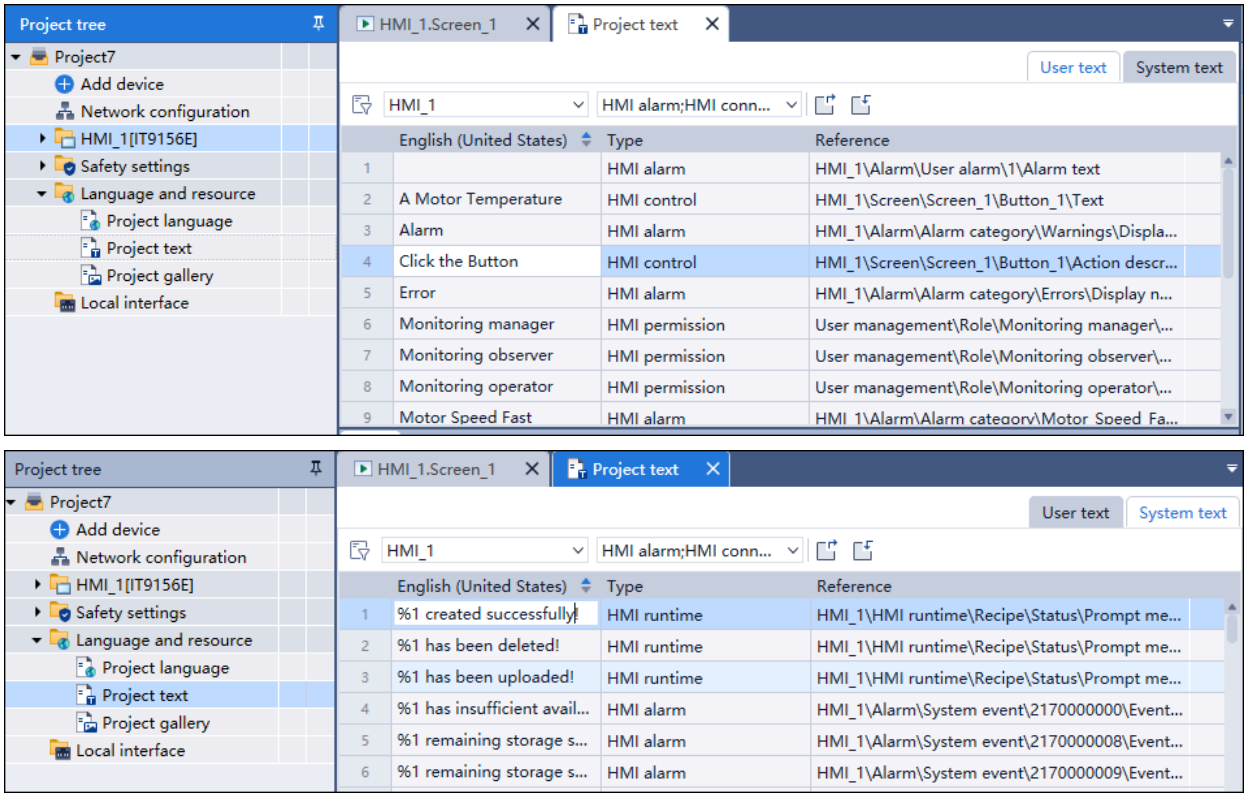
- Reference language: Select the base language used as a reference for translating other languages. Its default value is the default language.

Project text

After selecting a project language, the reference language serves as the base for translating other languages. Users can right-click an information text to jump to the corresponding text object, or locate it using the "Reference" column. Then edit the text objects in other languages.

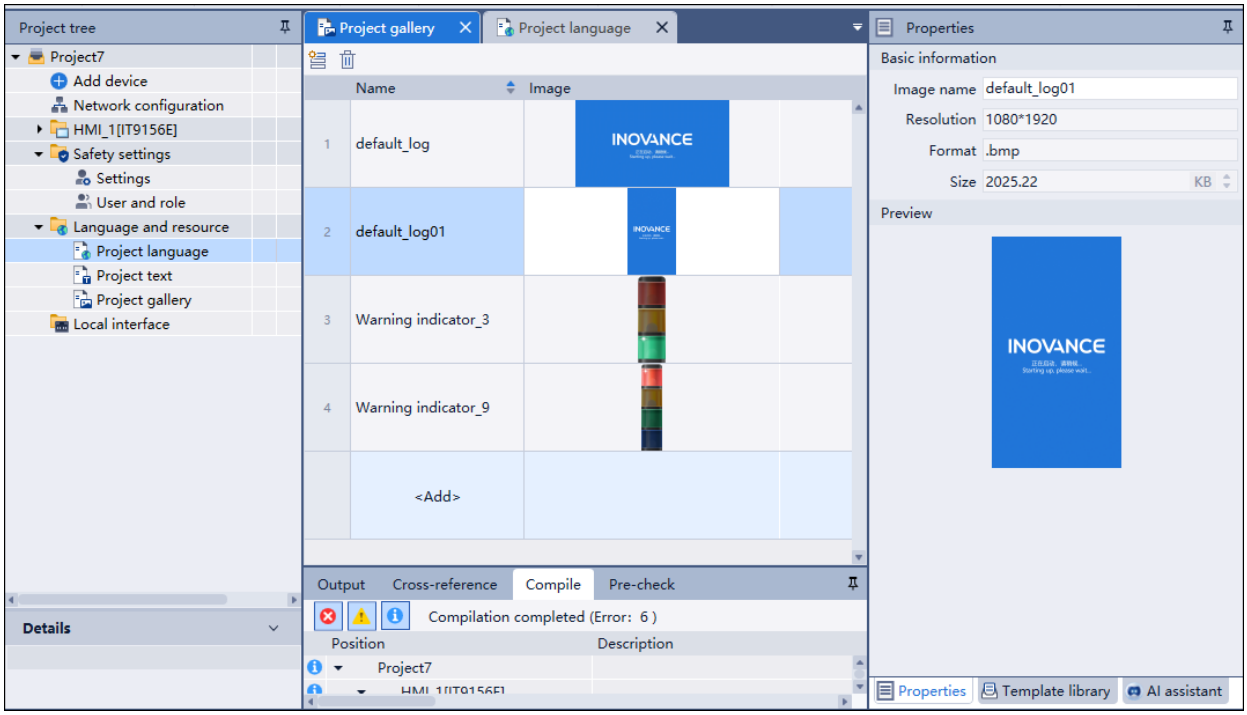
Project text is divided into user text and system text.

- User text: After user-defined information text objects are configured in 12 scenarios (such as HMI alarms, HMI variables, and HMI screens), they are automatically added to the user text list in the project texts.
- System text: Built-in information text objects of the system.



Project gallery

Graphics used within a project are automatically added to the project gallery for unified management. The project gallery can be used across all HMI devices in the current project.

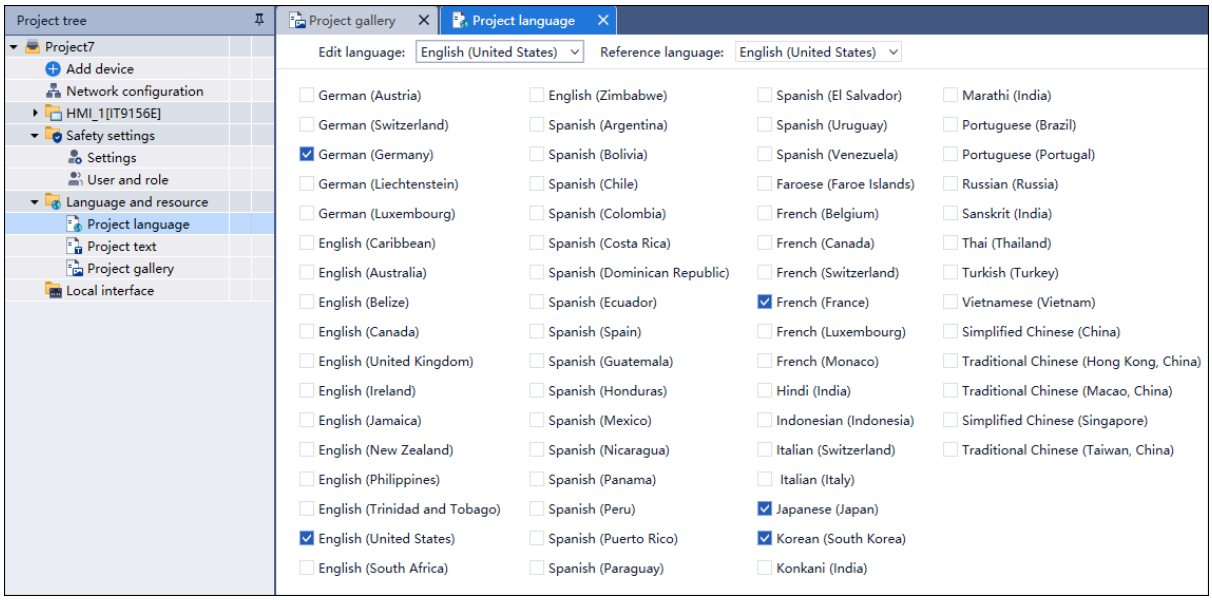


13.17.2 Multilingual Configuration and Modification

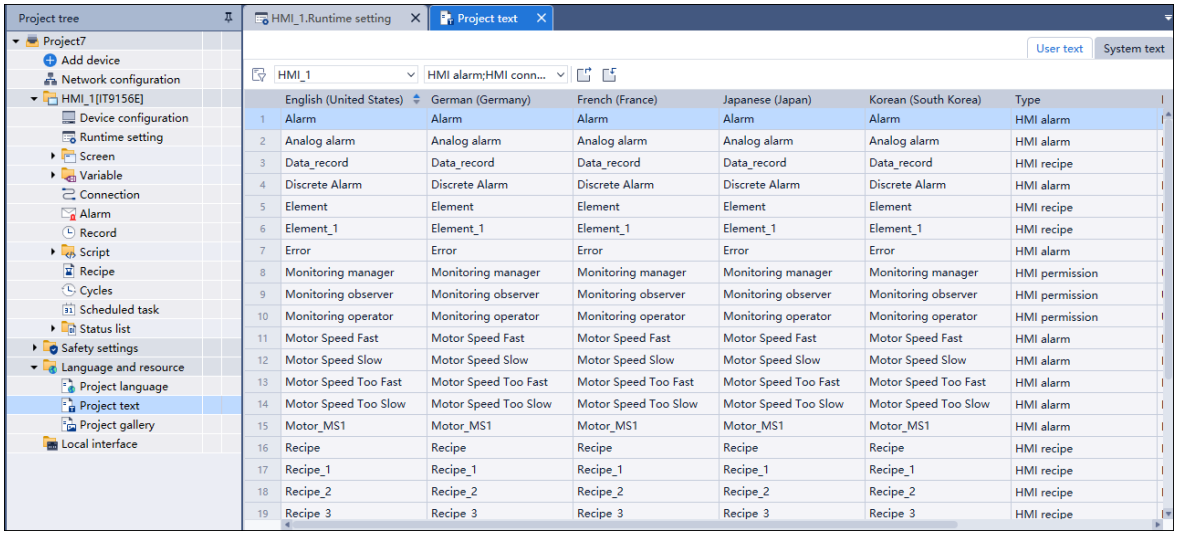
The project language page is used to specify the range of switchable languages. Only enabled languages can be dynamically switched on the HMI during operation. Disabled languages do not support switching.

Procedure

1. On the "Project language" page, check the required languages and select the edit language and reference language.



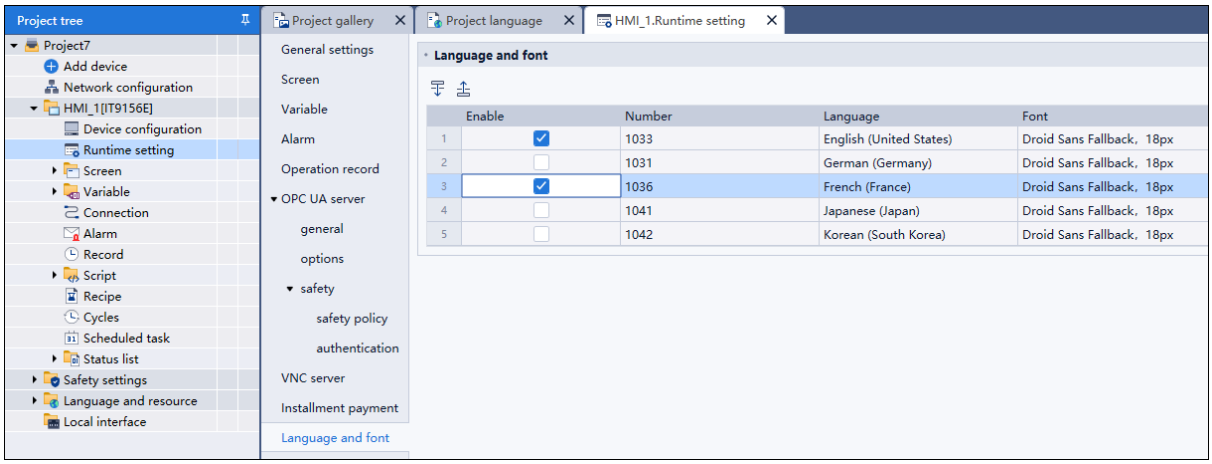
2. After adding an object that contains multilingual text information, the system simultaneously adds the corresponding project text information to the project text. Objects such as symbols, screens, alarms, and text lists support multilingual text information. For details, see 13.17.3 Multilingual Translation Scope.
   Example:
   a. Add a button control to the screen and edit the text information.
   b. In the project tree, double-click "Project text" under "Language and resource" to edit the text information for each language.



c. Switch the edit language of the screen; the language of the information text will change accordingly.

> **NOTE**
>
> - If an information text object is deleted, the corresponding information in the project texts will also be deleted.
> - When undoing or restoring objects containing multilingual text information, the system will synchronously undo or restore the project text information.
> - When copying objects containing multilingual text information within the device, across devices, or across projects, the system will synchronously copy the multilingual text information to the project texts. During pasting, all language texts that have been activated in the project for the selected content must be pasted together.

3. In the "Runtime setting" of the HMI device, activate the target language.



The order of languages can be adjusted.

Each language corresponds to a unique ID, which is used when configuring 19.9.12 language functions.

13.17.3 Multilingual Translation Scope

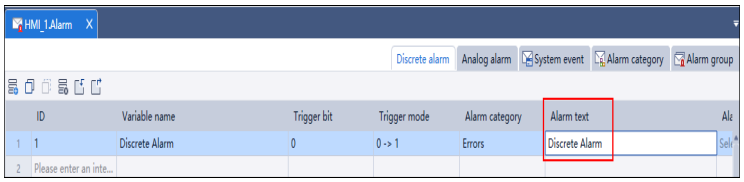
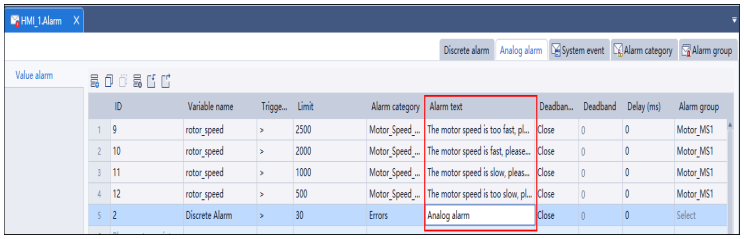
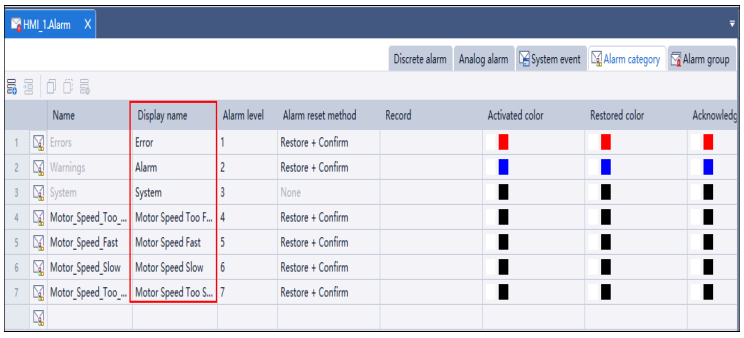
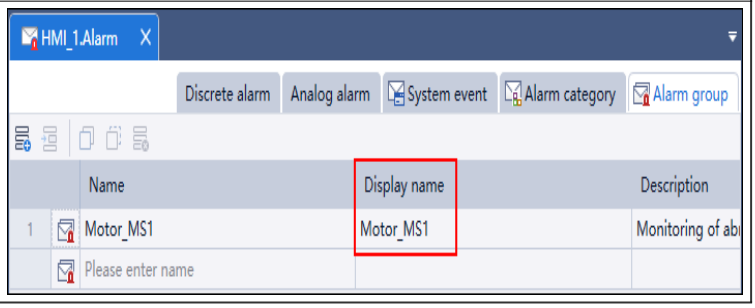
Multilingual translation scope

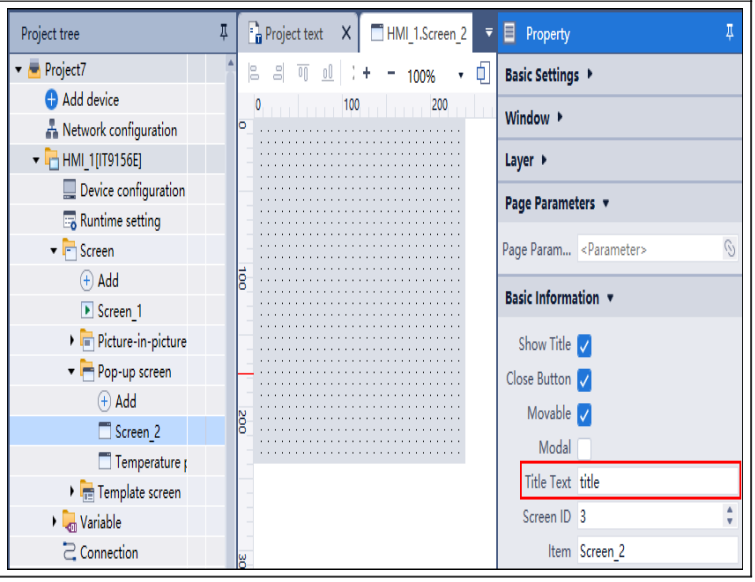
Project text is divided into user text and system text.

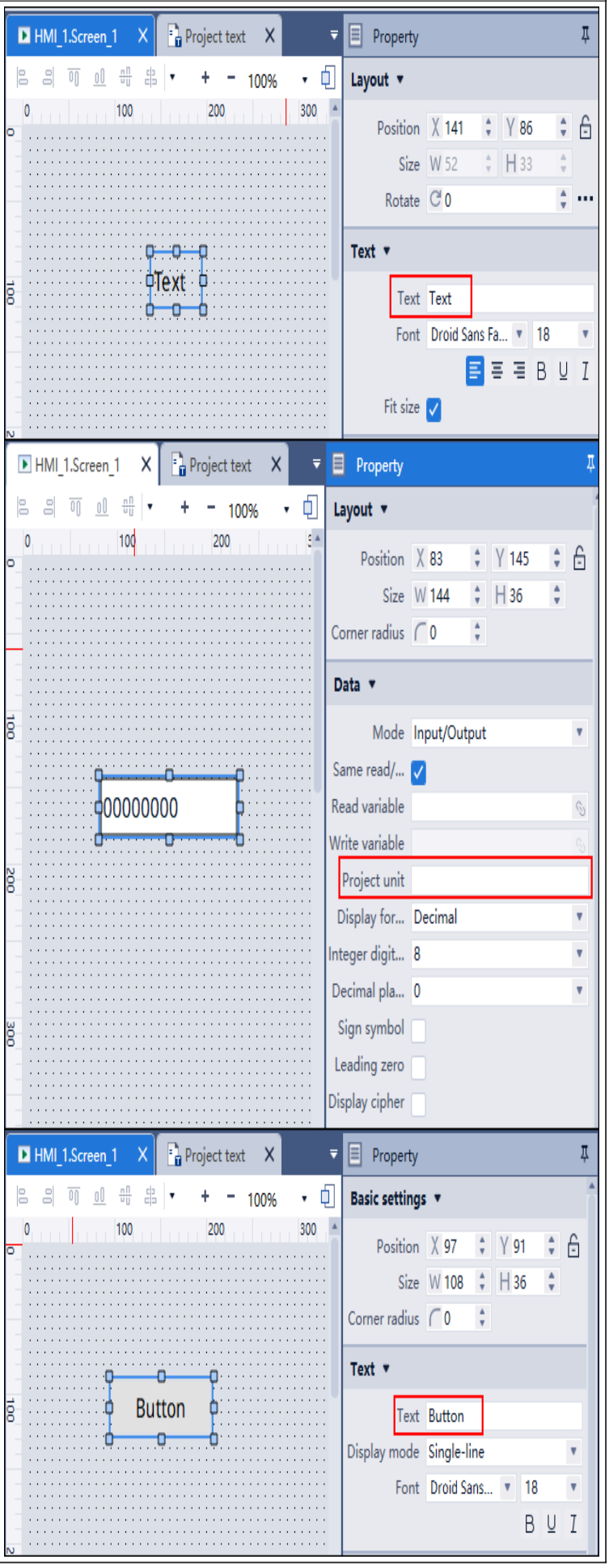
- User text refers to user-defined information text objects in 12 scenarios, including HMI alarm, HMI variable, HMI screen, etc.
- System text refers to built-in information text objects of the system.

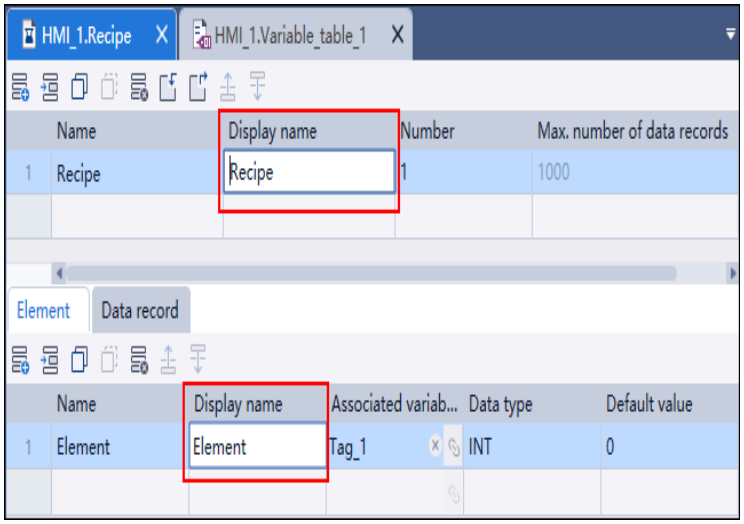
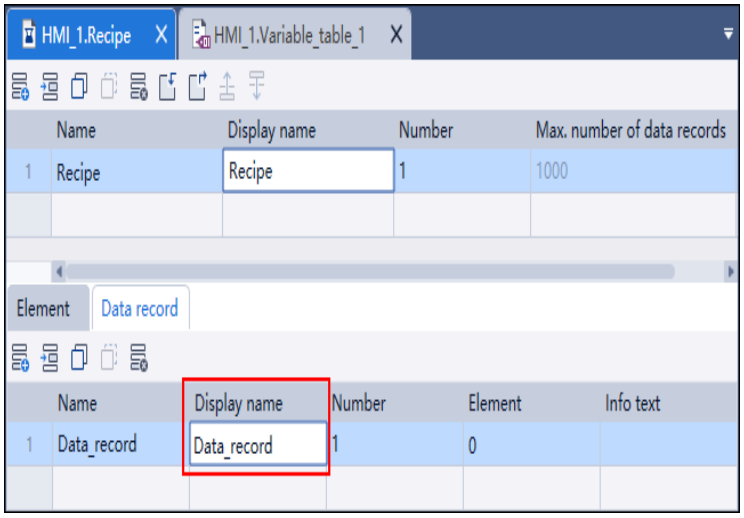
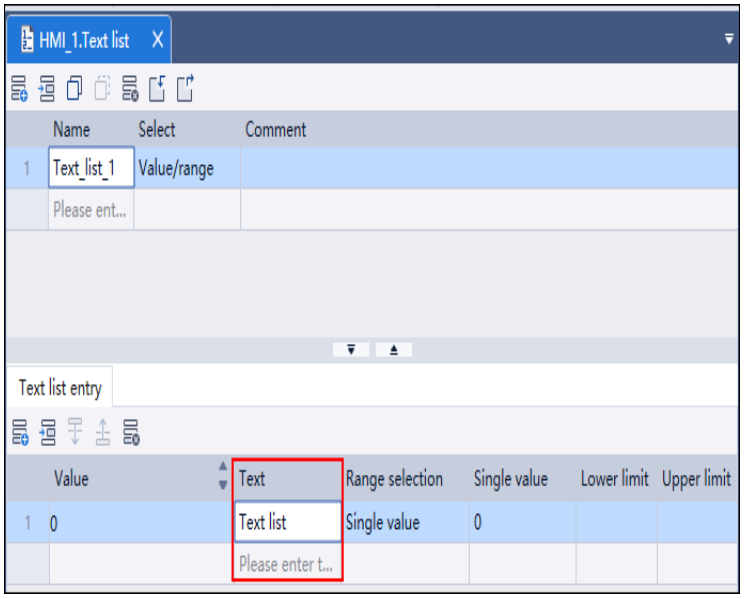
The table below lists the sources and scope of various types in user text, making it convenient for users to check and translate.

Users can right-click an information text on the project text page and select "Jump to text properties" to jump to the information text object. They can also locate the information text object based on the corresponding reference column, and edit the text object in other languages in both user text and system text.

Category	Scope	Editing Method
HMI alarm	"Alarm text" for discrete alarms.	
	"Alarm text" for analog alarms.	
	"Display name" for alarm categories.	
	"Display name" for alarm groups.	

Category	Scope	Editing Method
HMI screen	"Title text" for pop-up screens.	

Category	Scope	Editing Method
HMI control	Text supported for editing in controls, such as "Text content" and "Text" of the control, as illustrated below: • "Text content" of text controls. • "Project unit" of numeric I/O field controls. • "Text content" of button controls. Unsupported controls: SFC view and code view.	

Category	Scope	Editing Method
HMI recipe	"Display name" of recipes.	
	"Display name" of elements.	
	"Display name" of data records.	
HMI text list	"Text" of list items.	

13.18 HMI Program Reuse Across Projects

Cross-project HMI program reuse allows the process program of one HMI device to be reused in another device of the same model, reducing redundant configuration work and improving configuration efficiency.

Usage limits

Both of the following conditions must be met:

- Same device model: The source HMI and the target HMI must be of the same model.
- Same software version: The iFA software version on the PC must be consistent with the runtime program version running on the HMI device.

Reuse limits

Not all device information can be reused across projects via copy-paste or upload operations. The scope of reusable content via copy-paste is exactly the same as that supported by upload. For details, see Upload scope.

Program reuse procedure

- Method 1: Local source project exists – copy and paste directly within the software
  . Applicable scenario: The source project containing the device program already exists in iFA.
  1. Open the iFA software on the same computer;
  2. Open two windows and load the source project (containing the device program to be copied) and the target project;
  3. Select the HMI device program in the source project, then use Copy and Paste commands to complete cross-project reuse.

- Method 2: No local source project – upload the device program first, then copy and paste
  . Applicable scenario: The source project does not exist in iFA; the program is stored only on the physical HMI device.
  1. Connect the device and computer through one of the following:
     Ethernet connection: Configure the computer IP address to ensure it is on the same subnet as the HMI device.;
     USB connection: Connect the device and computer directly using a USB cable;
  2. Upload the program: Open iFA and perform the 13.14.3 upload operation to read the program from the device and generate a local source project;
  3. Cross-project copy: Follow the steps in Method 1 to copy the uploaded device program to other target projects.

13.19 Migrating InoTouchPad Project to iFA Evolution

- 13.19.1 InoTouchPad Project Migration
- 13.19.2 Communication Protocol Conversion During Project Migration

13.19.1 InoTouchPad Project Migration

InoTouchPad projects support migration to iFA Evolution projects.

Migration restrictions

Before migration, confirm that the software versions and device models meet the following requirements:

Item	InoTouchPad	iFA Evolution
Software version	V1R7C02SPC1Beta and above	iFA Evolution V1.4.4.5 and above
Device model	ITS7156E (1920*1080)	IT9156E (1920*1080)

Precautions before migration

Project backup: Make a complete backup of the InoTouchPad project and the iFA Evolution project.

Environment check: Ensure that the software versions on both sides meet the requirements and are running normally.

Function compatibility check: The current specified versions of InoTouchPad and iFA Evolution are not fully compatible. Please pre-check whether the InoTouchPad project involves the items listed in the following table and evaluate the impact.

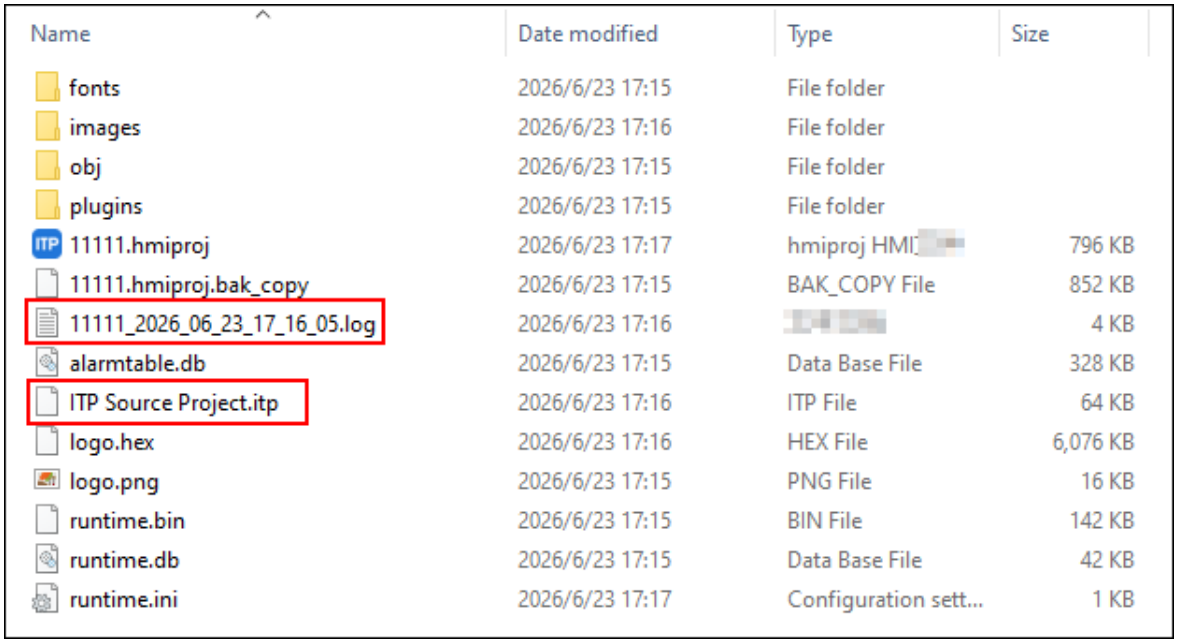
Module	Incompatible Item	Solution after Migration
Connection	Some communication protocols are supported by InoTouchPad but not by iFA Evolution.	Related connection data is not imported, and communication variables are automatically converted to internal variables. Equivalent protocols must be manually created and variables re-associated. For the correspondence of communication protocols, see 13.19.2 Communication Protocol Conversion During Project Migration.
Index tag	Index variables associated with absolute address protocols and internal connections.	Not imported.
	Index variables associated with non-array type variables under tag protocols.	Not imported.
	Different index logic.	index variable takes effect: Sub-index = index value
Style sheet	Some advanced fill style properties are not compatible, such as complex gradients and path brushes.	Only basic properties are imported; other appearance styles need to be manually adjusted.
Script	Function name and syntax are incompatible.	Refer to the error log and perform equivalent conversion item by item.
System variable	Some system variables are incompatible.	There is an alarm during import.
System function	Some functions are supported in InoTouchPad but not in iFA Evolution.	Related function data is not imported; please pay attention to the prompt information during import.

Module	Incompatible Item	Solution after Migration
Data record	When the variable recording mode is based on command, it is incompatible.	It needs to be manually configured to demand-based storage.
Event	Some events are supported in InoTouchPad but not in iFA Evolution.	Related event data is not imported; please pay attention to the prompt information during import.
Alarm	Some alarm properties are incompatible.	Alarm categories, groups, and properties are not imported; please pay attention to the prompt information during import.
Structure	Structure variables are incompatible.	Create the corresponding structure definition in the template library; naming rule: HMI instance name_structure name. Structure definitions with the same name are not imported repeatedly.
Timer control	Timer control is incompatible.	After migration, a timer control instance corresponds to the timer event on the screen. If a screen contains multiple timer controls, only the data of the control with the smallest ID is matched to the screen timer event after migration. Data of other controls is not imported.
Layer	Layer is incompatible.	After import, the contents of all visible layers in the InoTouchPad project (regardless of their original layer numbers) are merged into Layer 0. The contents of all invisible layers in the InoTouchPad project (regardless of their original layer numbers) are merged into Layer 1 after import.
Recipe	Recipe properties (synchronized variables and offline variables) are incompatible.	Properties are not imported. Element variables are automatically created.
Control	Alarm records, alarm bars, recipe views, user views, and operation records are incompatible.	The control's initial state is retained at its original position on the screen after import.
Animation	Some animation types are incompatible.	The flashing property of appearance change animations is converted to fill flashing animation. After import, pipe elements are automatically matched with pipe flow animation, and a Boolean variable "PipeFlow" with a default value of 1 is automatically created. The flow speed is matched to the closest available option based on the original value.

Module	Incompatible Item	Solution after Migration
Special character	Special characters in names (such as screens or variables) are incompatible.	Replaced with an underscore "_".
Name length	Name length is incompatible.	Names are truncated; exceeding parts are not imported. Prompts are output in the monitoring area.

Migration procedure

1. Export the InoTouchPad project.
   a. Open the source project to be migrated in InoTouchPad software.
   b. Export the project file from InoTouchPad.
   Select "Project > Export migration project" to export the project file.
   Two key files are exported: one is the migration project file (*.itp), the main file of the project data, which can be imported to iFA Evolution; the other is the export log file (*.log), which records the export process and content list in detail for verification and troubleshooting during migration. The file format is shown in the figure below.



2. Import the project to iFA Evolution.
   a. Open an existing project or create a project in iFA Evolution.
   b. Import the migration project file.
   Select "Project > Project migration" and select the target project file (with the suffix .itp).
   c. Start the migration.
   Click "Start", and the software automatically performs data conversion and device creation.
   Device names start from "HMI_1" and increment sequentially.

3. Check and adjust.
   After the migration, check and adjust each item according to the prompt information in the output area to ensure the integrity and accuracy of the data.

13.19.2 Communication Protocol Conversion During Project Migration

Ethernet

Category	InoTouchPad	iFA Evolution
Inovance	AM600 Modbus TCP protocol	Modbus TCP protocol (AM400/600 series)
	AM600 Qlink TCP protocol	Qlink TCP protocol (AM400/600 series)
	AM700 Qlink TCP protocol	Modbus TCP Protocol (AM300/500, AC700/800 series)
	AC810 Qlink TCP protocol	
	H5U TCP monitoring protocol	TCP monitoring protocol (H5U/Easy series)
	Easy TCP monitoring protocol	
	H5U Qlink TCP protocol	Qlink TCP protocol (H5U series)
	EthernetIP Tags protocol	Inovance Technology, Ethernet/IP Tags protocol
	AC800 Tags protocol	AC800 Tags protocol (AC/AI/AM/AP series)
Omron	Omron EthernetIP Tags monitoring protocol	Omron, EtherNet/IP (NJ/NX/NY series) - high speed
	Omron EthernetIP Tags protocol	Omron, EtherNet/IP (NJ/NX/NY series) - standard speed
Siemens	S7-1200\1500 (Ethernet)	S7-1200\1500
	S7-300 (Ethernet)	S7-300
Modbus IDA	Modbus TCP protocol	Modbus TCP client
	Modbus TCP slave	Modbus TCP server
OPC UA	OPC UA Tags protocol	OPC UA client
Mitsubishi	FX5U (Binary)	FX5U (Binary) protocol (FX5U series)
	MelSec-Q\L (Binary)	MelSec-Q\L (Binary) protocol (Q series)

Serial interface

Category	InoTouchPad	iFA Evolution
Inovance	AM600 Modbus protocol	Modbus RTU protocol (AM400/600 series)
	H5U Qlink protocol	Qlink RTU protocol (H5U series)
	MD series AC drive Modbus protocol	Modbus RTU protocol (MD series)
Modbus IDA	Modbus RTU protocol	Modbus RTU master
	Modbus slave	Modbus RTU slave

14 Maintenance Upgrade

14.1 PLC Firmware and Project Upgrade

14.1.1 Firmware Upgrade

14.1.1.1 Upgrading Firmware via SD Card

The PLC firmware upgrade package provides all required software data for the upgrade.

> NOTE
>
> Only EVO400, EVO500 and EVO800 series PLCs support firmware upgrade through SD card.

Prerequisites

- The device supports upgrade through SD card.
- A bootable SD card is ready (firmware can be downloaded from the Inovance website).

Procedure

1. Power off the device, and then insert the bootable SD card into the device.
2. Power on the device, and then the system begins the firmware upgrade. The device indicator flashes quickly for 3s.

   > NOTE
   >
   > The upgrade lasts about two minutes. Do not power off the device during the upgrade; otherwise, unrecoverable system faults may occur.

3. After the upgrade, power off the device and remove the SD card.
4. Power on the device to resume normal operation.

Device indicator states

Item	State
Firmware upgrade start	RUN + ERR: flashing quickly (3s)
Upgrade process	RUN + ERR: steady on
Upgrade success	RUN + ERR: flashing slowly
Upgrade failure	ERR: flashing slowly

14.1.1.2 Upgrading Firmware via Ethernet

The upgrade page is used for PLC firmware upgrade. The PLC firmware upgrade package contains all required software data for the upgrade.

Background

The EVO800 series firmware upgrade through Ethernet includes two methods: Direct connection between the PC and EVO800 series control module, and direct connection between the PC and EVO800-CM-EN Ethernet expansion module. The firmware upgrade description is shown in the table below.

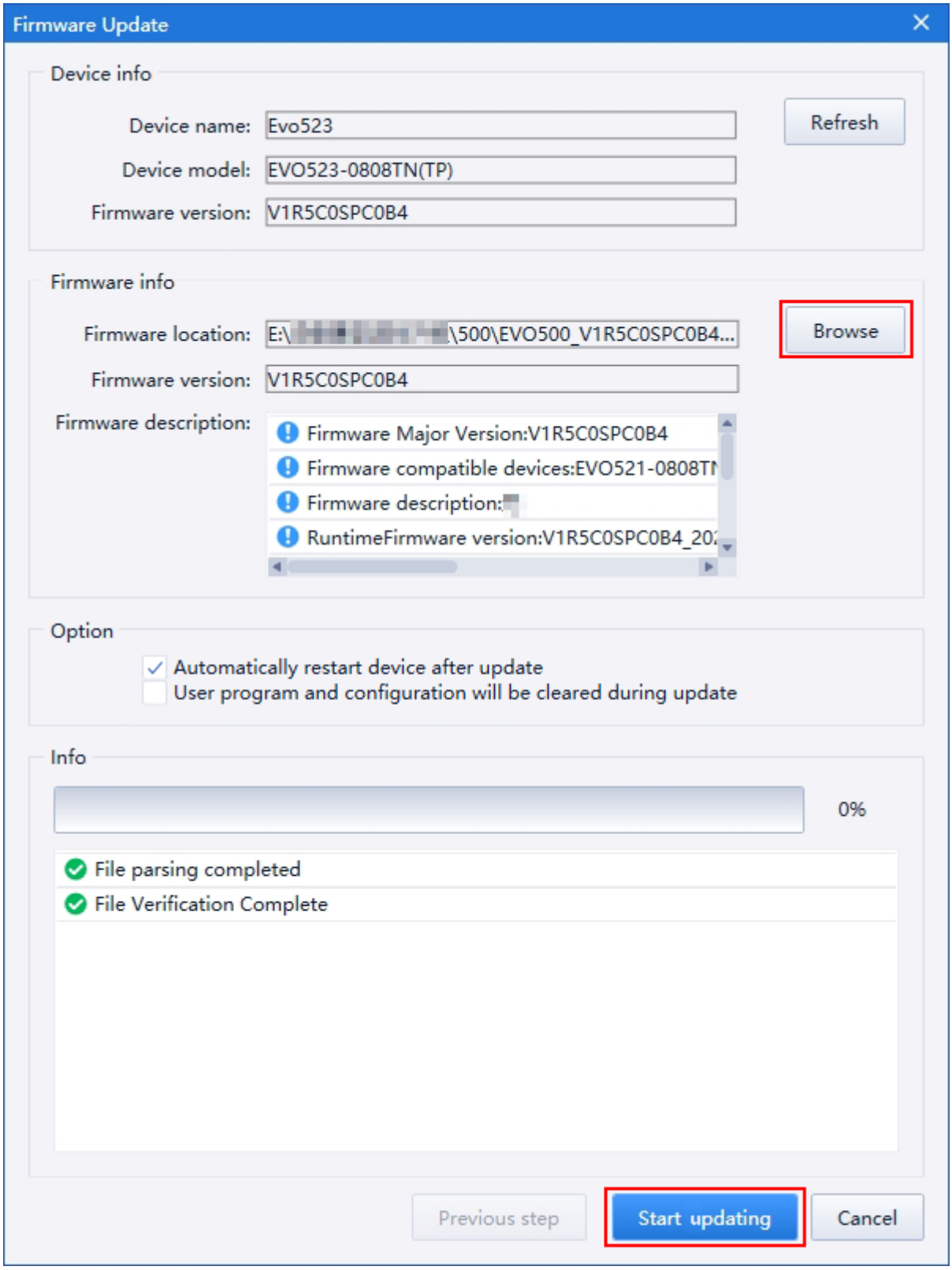
Ethernet connection method	Firmware upgrade description
Direct connection between the PC and EVO800 series control module	The following types of modules support firmware upgrades: • Directly connected control modules • EVO800-CM-EN Ethernet expansion module directly connected to the PLC • EVO800-CM-PN master expansion module directly connected to the PLC • EVO800-CM-DP master expansion module directly connected to the PLC
Direct connection between the PC and EVO800-CM-EN Ethernet expansion module	Only supports firmware upgrade for the directly connected EVO800-CM-EN Ethernet expansion module.

Prerequisites

The firmware file is obtained by downloading it from the Inovance website or contacting the Inovance service engineer.

Procedure

1. Choose "Online > Firmware update > PLC and HMI" from the menu bar.
2. On the opened "Device scanning" interface, select "Ethernet" for the "Communication type," and then click "Start search."
3. Select the target device, click "OK", and then the system automatically opens the "Firmware update" interface.
4. On the opened "Firmware update" interface, click "Browse" to select the firmware file to be upgraded, select the upgrade method under the "Options", and then click "Start updating".



Parameter		Description
Device information	Device name	Current device name.
	Device model	Current device model.
	Firmware version	Current firmware version of the device.
Firmware information	Firmware location	The location where the firmware is stored.
	Firmware version	The firmware version to be upgraded.
	Firmware description	Detailed information about the firmware upgrade package.

Parameter		Description
Options	Restart device after update	Automatically restart the current device after firmware update.
	Clear user program and configuration during the update	User projects, data, IP configuration, and other properties in the PLC will be cleared during the upgrade.

5. Click "OK" on the opened dialog box to start the firmware upgrade.

NOTE

- Do not power off the device during the upgrade; otherwise, unrecoverable system faults may occur.
- The upgrade lasts about two minutes. After the upgrade, the device automatically restarts.
- After the device restart (or the upgrade), the device's display shows "00" or changing numbers.

6. After the upgrade, check the device information and version information for consistency with the firmware version and details.

Device indicator states

Item	State
Firmware upgrade start	RUN + ERR: flashing quickly (3s)
Upgrade process	RUN + ERR: steady on
Upgrade success	RUN + ERR: flashing slowly
Upgrade failure	ERR: flashing slowly

14.1.1.3 Upgrading Firmware via Type C

The PLC firmware upgrade package provides all required software data for the upgrade.

NOTE

Only EVO400, EVO500, and EVO700 series PLCs support firmware updates through Type-C interface.

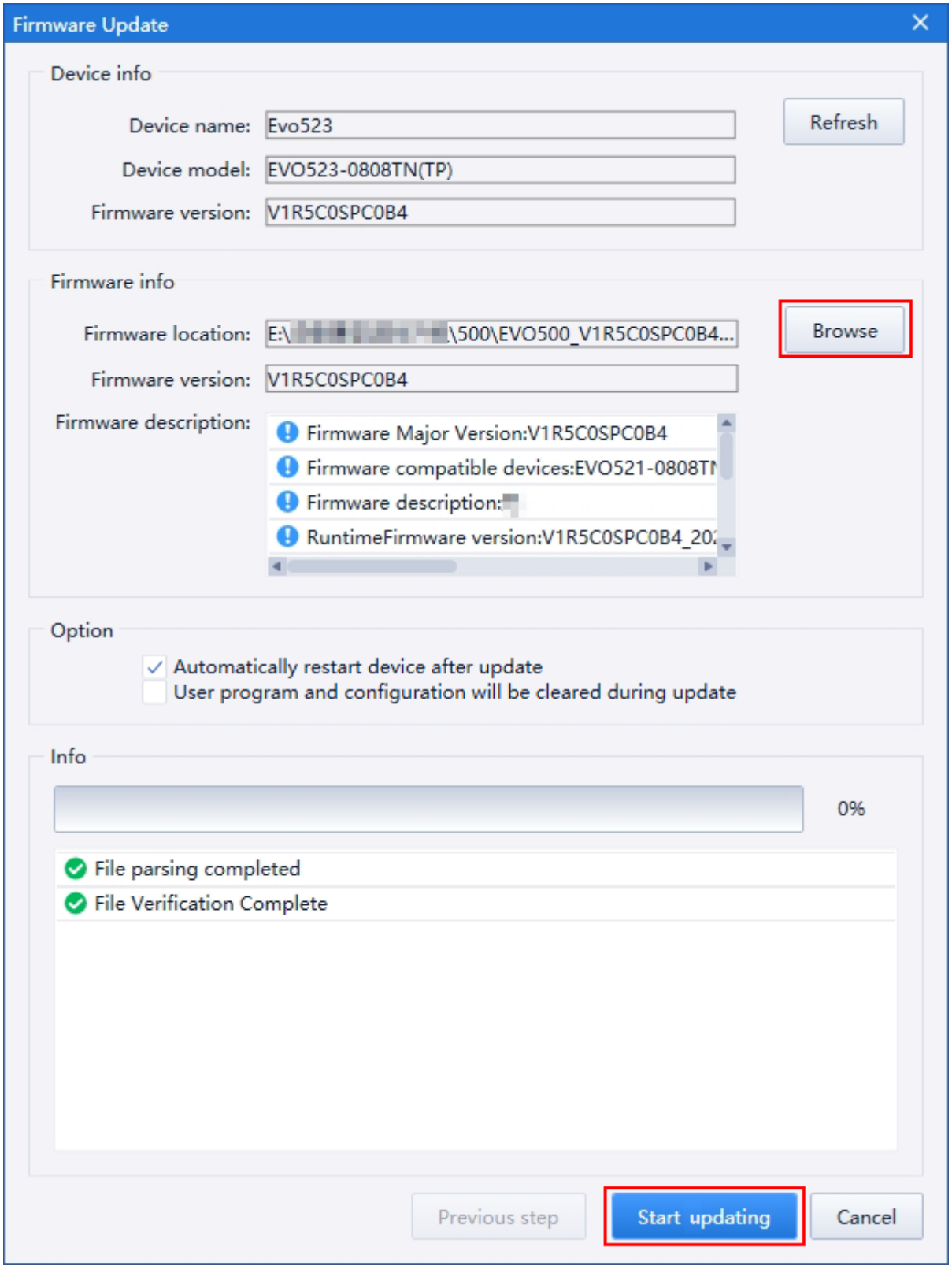
Prerequisites

- The PLC is connected to a Type-C interface.
- The firmware file is obtained by downloading it from the Inovance website or contacting the Inovance engineer.

Procedure

1. Choose "Online -> Firmware update -> PLC and HMI" from the menu bar.

2. On the opened "Device scanning" interface, select "USB" for the "Communication type," and then click "Start search."
3. Select the target device, click "OK", and then the system automatically opens the "Firmware update" interface.
4. On the opened "Firmware update" interface, click "Browse" to select the firmware file to be upgraded, select the upgrade method under the "Options", and then click "Start updating".



Parameter		Description
Device information	Device name	Current device name.

Parameter		Description
	Device model	Current device model.
	Firmware version	Current firmware version of the device.
Firmware information	Firmware location	The location where the firmware is stored.
	Firmware version	The firmware version to be upgraded.
	Firmware description	Detailed information about the firmware upgrade package.
Options	Restart device after update	Automatically restart the current device after firmware update.
	Clear user program and configuration during the update	User projects, data, IP configuration, and other properties in the device will be cleared during the upgrade.

5. Click "OK" on the opened dialog box to start the firmware upgrade.

NOTE

- Do not power off the device during the upgrade; otherwise, unrecoverable system faults may occur.
- The upgrade lasts about two minutes. After the upgrade, the device automatically restarts.
- After the device restart (or the upgrade), the device's display shows "00" or changing numbers.

6. After the upgrade, check the device information and version information for consistency with the firmware version and details.

Device indicator states

Item	State
Firmware upgrade start	RUN + ERR: flashing quickly (3s)
Upgrade process	RUN + ERR: steady on
Upgrade success	RUN + ERR: flashing slowly
Upgrade failure	ERR: flashing slowly

14.1.1.4 Upgrading Firmware via USB Flash Disk (EVO700)

EVO700 series PLCs support firmware upgrade through USB flash disk.

Prerequisites

- A USB flash disk with a capacity of not more than 32 G and using the FAT32 file system is ready.
- A bootable USB flash disk is created with the support of the Inovance engineer.

Procedure

1. Power off the device, and then insert the USB flash disk into the device's USB port.

2. Power on the device, and then the system begins the firmware upgrade.
3. During the upgrade, the segment display icon 888shows "UUU" and the icon ▬▬▬▬indicates the progress.
4. After the upgrade, the buzzer beeps three times, and the segment display icon 888shows "UUU" and flashes or shows "00".
5. Power off the device, remove the USB flash disk, and then power on the device to resume normal operation.

14.1.1.5 Upgrading Redundant System Firmware (EVO800)

14.1.1.5.1 Upgrading via SD Card

The EVO800 series PLC system supports dual-rack redundant mode with a primary-backup switchover in 20 ms. When upgrading the redundant system firmware, upgrade the primary and backup PLCs and redundant systems separately.

The methods for upgrading primary and backup devices are the same. The following describes the method for upgrading the primary device.

Prerequisites

- The dual-rack redundant system, consisting of primary and backup PLCs, power supply modules, racks, and redundant modules, has been installed and powered on, and the devices are operating normally.
- An SD card with a capacity of 8GB to 32GB is ready, and a bootable SD card is created with the support of the Inovance engineer.

Procedure

1. Power off the device, and then insert the bootable SD card into the SD card slot of the primary PLC.
2. Power on the device and wait for 30s (the power indicator being steady on indicates normal power supply). The system automatically starts the firmware upgrade. For the indicator states during the upgrade, see the "Device indicator states" table.
3. After the firmware upgrade, remove the SD card and power off the device.
4. Power on the device to resume normal operation.
5. Follow the preceding steps to upgrade the backup device firmware.

After the primary device firmware is upgraded, the firmware will not be automatically synchronized to the backup device. Therefore, you must manually upgrade the backup device using the same version of the firmware package to ensure the same firmware versions of the primary and backup devices.



The PLC firmware upgrade package includes the redundant module firmware package. As a result, the primary and backup redundant modules will be upgraded synchronously during the upgrade of the primary and backup PLCs.

Device indicator states

Item	State
Upgrade start	The SD card indicator on the display is lit (card inserted), and the digital display flashes slowly at 000 for 3s.
Upgrade process	The SD card indicator on the display flashes (data reading/writing in progress), and the digital display shows 1 to 100.
Upgrade success	The SD card indicator on the display is lit; the digital display shows 100, and the yellow exclamation mark flashes continuously, prompting the user to remove the card.
Upgrade failure	The red exclamation mark flashes, and the SD card indicator is lit; the digital display shows a fault code greater than 100.

14.1.1.5.2 Upgrading via Ethernet

The EVO800 series PLC system supports dual-rack redundant mode with a primary-backup switchover in 20 ms. When upgrading the redundant system firmware, upgrade the primary and backup PLCs and redundant systems separately.

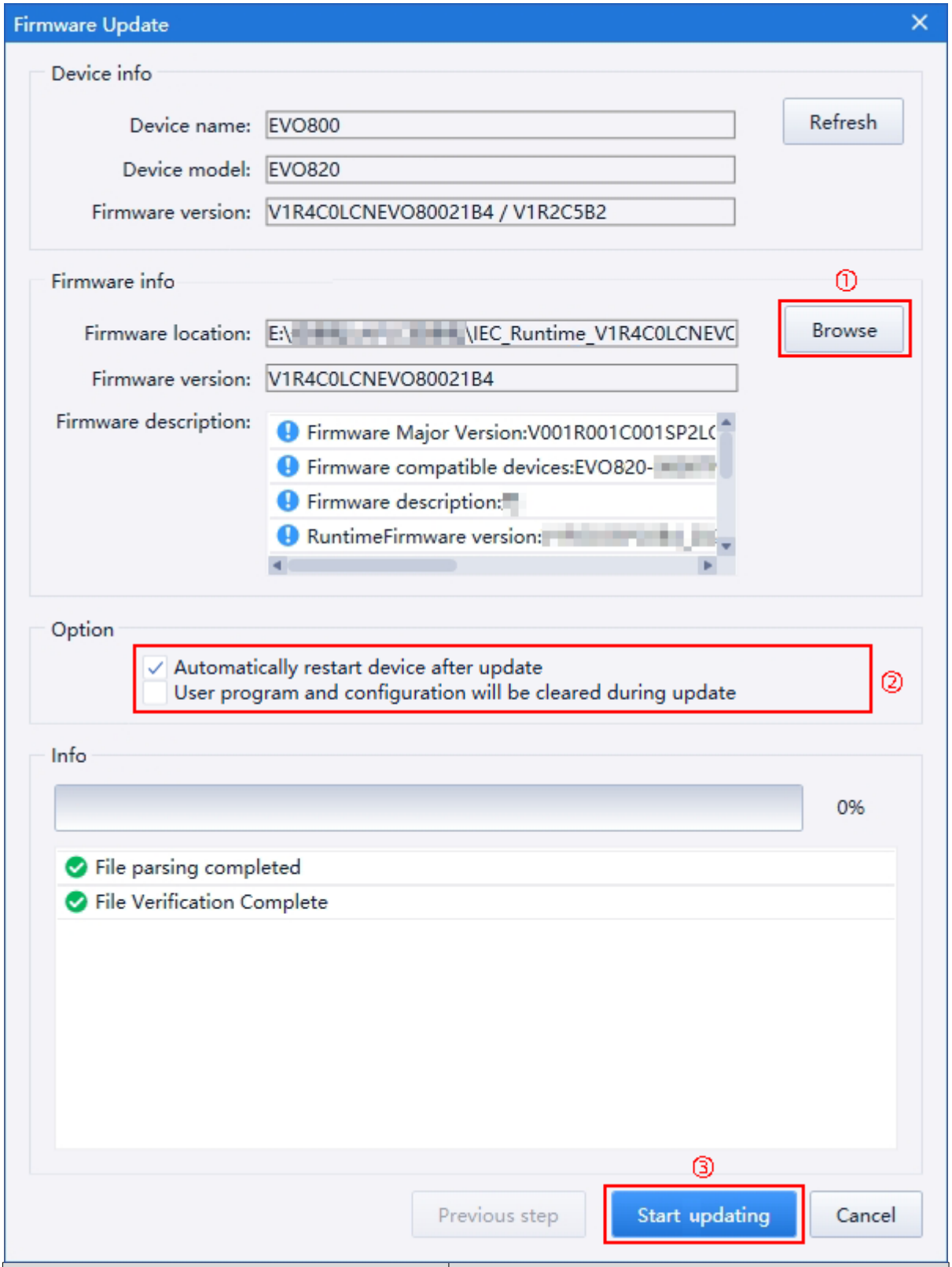
The methods for upgrading primary and backup devices are the same. The following describes the method for upgrading the primary device.

Prerequisites

- The dual-rack redundant system, consisting of primary and backup PLCs, power supply modules, racks, and redundant modules, has been installed and powered on, and the devices are operating normally.
- The primary and backup PLCs have been connected to the Ethernet via network cables.
- The optical fiber cable connecting the redundant racks has been disconnected; otherwise, the message "Firmware upgrade is prohibited in redundant mode" will appear.
- The PLC firmware upgrade package (such as EVO800-RM_V1R4C0LCNEVO80021B4_20250408_5053.zip) has been obtained by downloading it from the Inovance website or contacting the Inovance engineer.

Procedure

1. Choose "Online -> Firmware update -> PLC and HMI" from the menu bar.
2. On the opened "Device scan" interface, select "Ethernet" for the "Communication type," and then click "Start search."
3. Select the target device, click "OK", and then the system automatically opens the "Firmware Update" interface.
4. On the opened "Firmware update" interface, click "Browse" to select the firmware file to be upgraded, select the upgrade method under the "Options", and then click "Start updating".



Parameter		Description
Device information	Device name	Current device name, such as EVO800.
	Device model	Current device model, such as EVO820.
	Firmware Version	Current firmware version of the device, such as V1R4C0LCNEVO80021B4.
Firmware information	Firmware location	The location where the firmware is stored.
	Firmware version	The firmware version to be upgraded.

Parameter		Description
	Firmware description	Detailed information about the firmware upgrade package.
Options	Restart device after update	Automatically restart the current device after firmware update. It is recommended to check this option to ensure a successful firmware update.
	Clear user program and configuration during the update	User projects, data, IP configuration, and other properties in the device will be cleared during the upgrade.

5. When the upgrade progress under the "Information" bar shows 100% and the device begins to restart, the firmware is updated successfully. Wait for the device to restart.

NOTE

- Do not power off the device during the upgrade; otherwise, unrecoverable system faults may occur.
- The upgrade takes approximately two minutes, and the device automatically restarts after upgrade (if the option "Automatically restart the device after upgrade" is checked); otherwise, the device must be restarted manually.
- After the device restart (or the upgrade), the device's display shows "00" or changing numbers.

6. Follow the preceding steps to upgrade the firmware for the backup PLC and redundant module.
After the primary device firmware is upgraded, the firmware will not be automatically synchronized to the backup device. Therefore, you must manually upgrade the backup device using the same version of the firmware package to ensure the same firmware versions of the primary and backup devices.

NOTE

- The PLC firmware upgrade package includes the redundant module firmware package. As a result, the primary and backup redundant modules will be upgraded synchronously during the upgrade of the primary and backup PLCs.
- The redundant module cannot be automatically restarted. You must manually power off and then power on the redundant module to ensure a successful firmware upgrade.

14.1.1.6 Upgrading Safety PLC Firmware (EVO1000)

The EVO1000EF safety PLC module and the communication unit are combined into a firmware package. When a firmware update is performed, both devices are upgraded simultaneously.

- Downgrade: The version to be upgraded to is lower than the current firmware version.
- Same version: The version to be upgraded to is the same as the current firmware version.
- Upgrade: The version to be upgraded to is higher than the current firmware version.

> **NOTE**
>
> The firmware package must match the device model.

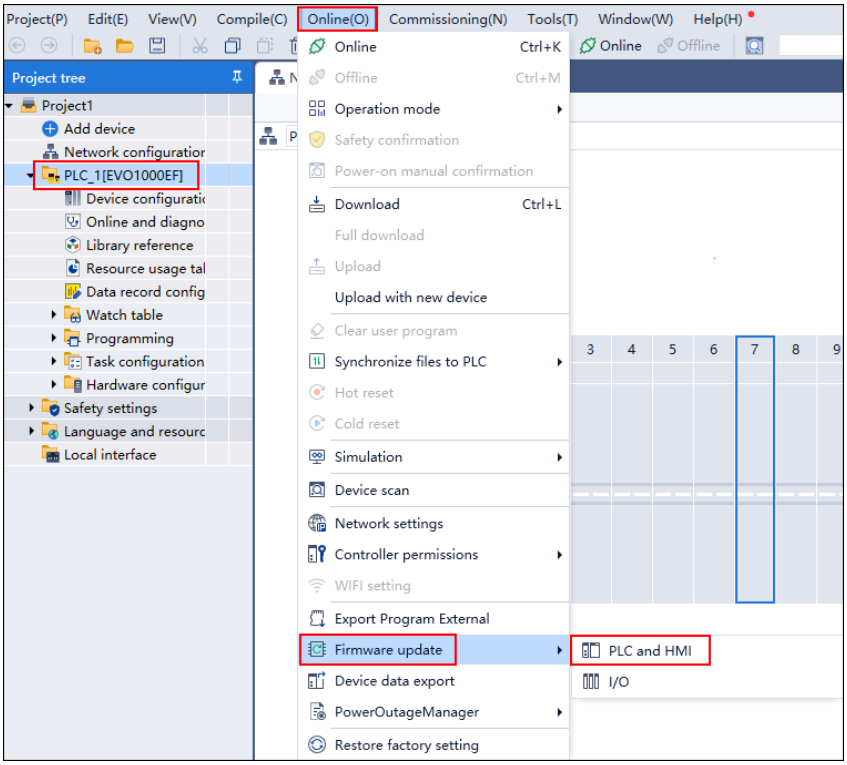
The following uses a firmware package created by combining the EVO1000EF safety PLC module firmware and the EVO1000EF EIP communication unit firmware as an example.

Prerequisites

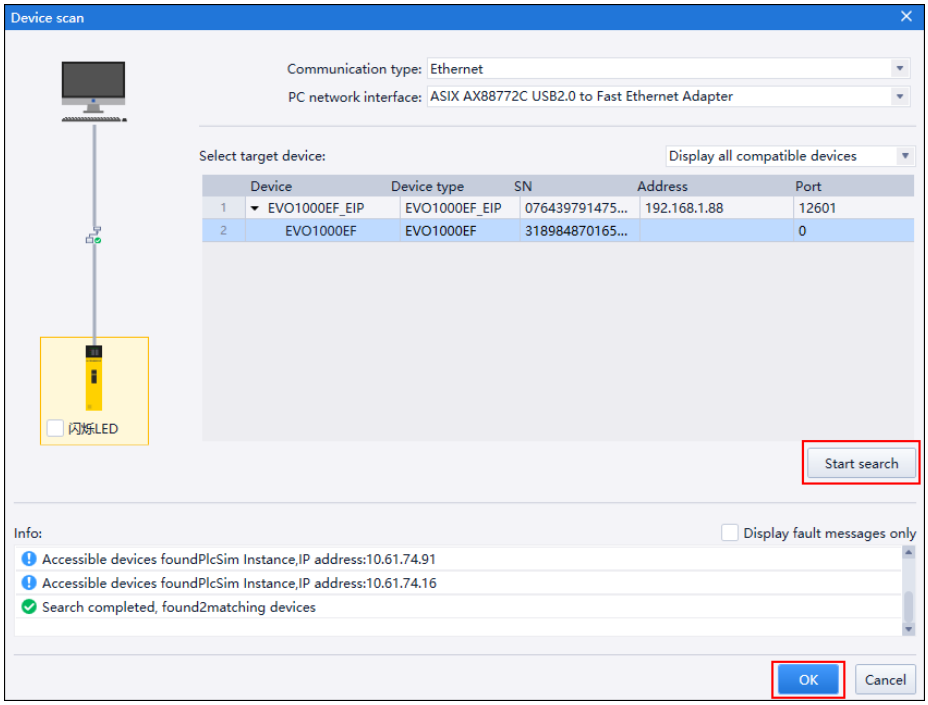
The firmware upgrade package has been obtained from the Inovance technical support.

Procedure

1. Open the iFA Evolution software, and select "Online > Firmware update > PLC and HMI" from the menu bar.

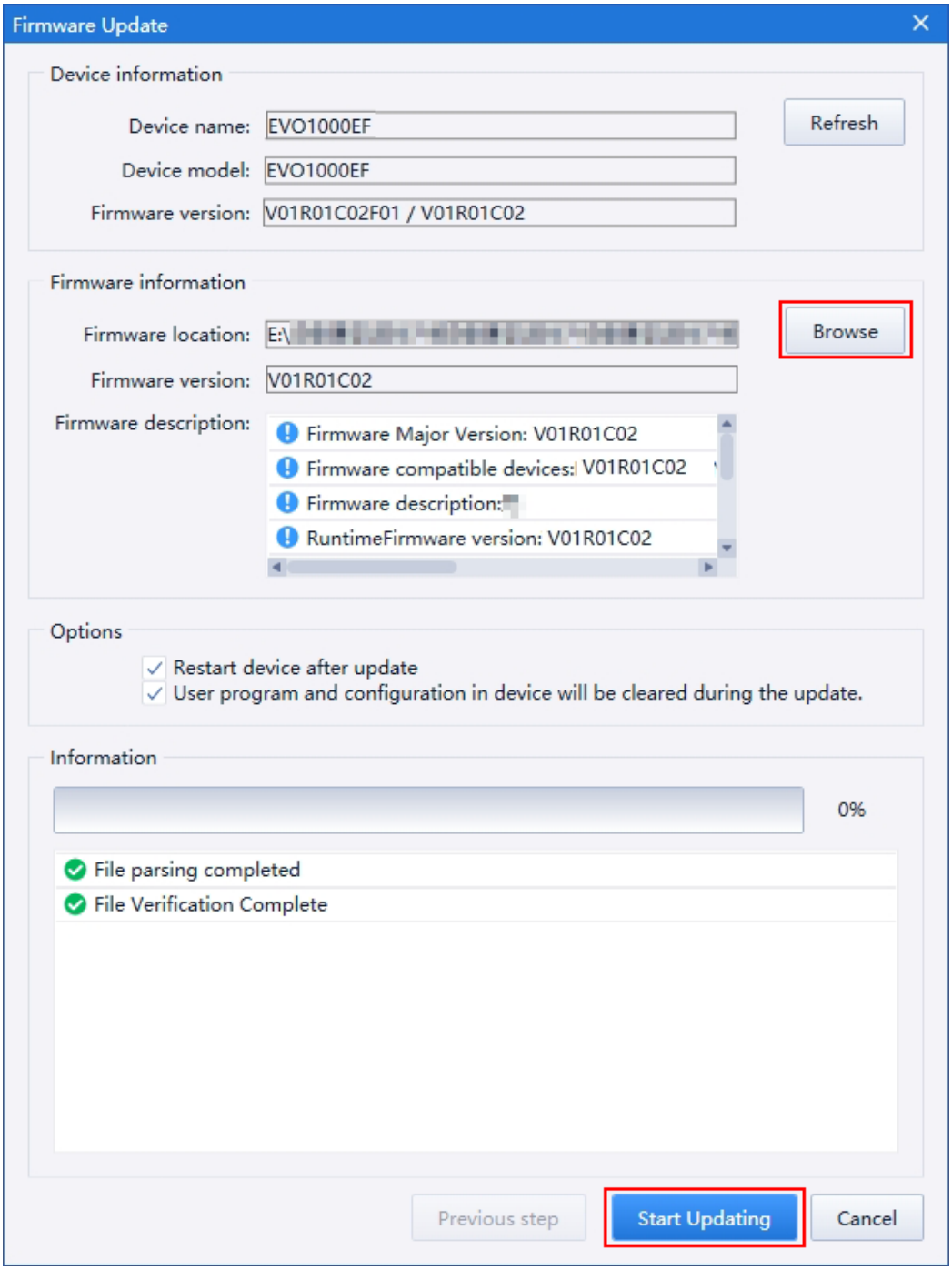


2. In the "Device scanning" window, click "Start search". Then, select the device to be upgraded and click "OK".



3. In the pop-up "Firmware update" window, click "Browse", locate the corresponding version of the firmware upgrade package on local PC, and click "OK".

4. After the firmware upgrade package is successfully opened, ensure the firmware version information is accurate, and click "Start update".



5. In the pop-up "Firmware update" window, click "OK" to start the firmware update.

   The "Firmware update" interface will display the firmware update progress. When the progress bar reaches 100%, it indicates that the firmware update is complete.

> **CAUTION**
>
> After the firmware update, the device will restart. The firmware update of the main partition during the device's boot phase will take several minutes. Wait until the CUS indicator on the safety PLC module becomes steady green before operating the device.

14.1.2 PLC User Program Upgrade

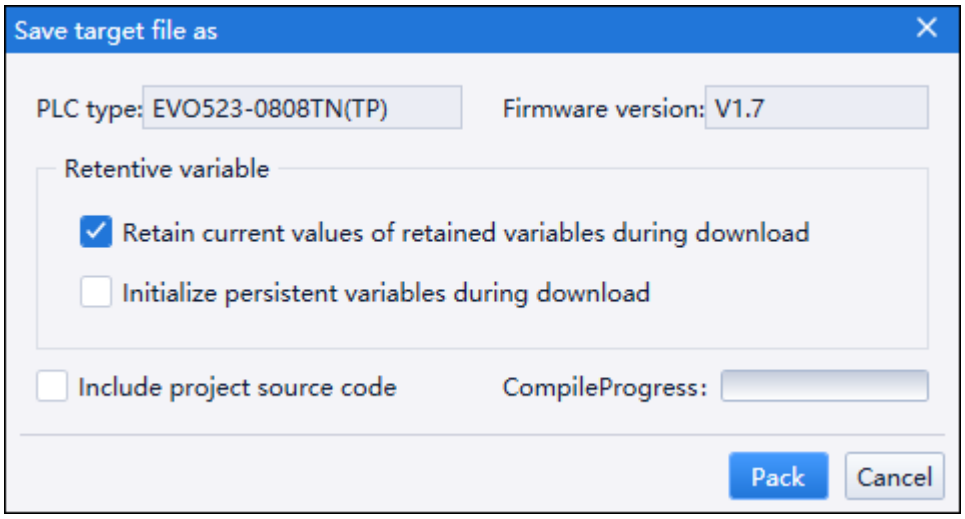
14.1.2.1 Saving a PLC User Program As

The software supports saving the successfully compiled user program as a target file, which can be used to upgrade the user program through SD card or USB flash disk.

> NOTE
>
> The saved target file contains only the device's general file: user project package information file (including device type, firmware version, whether to initialize at startup).

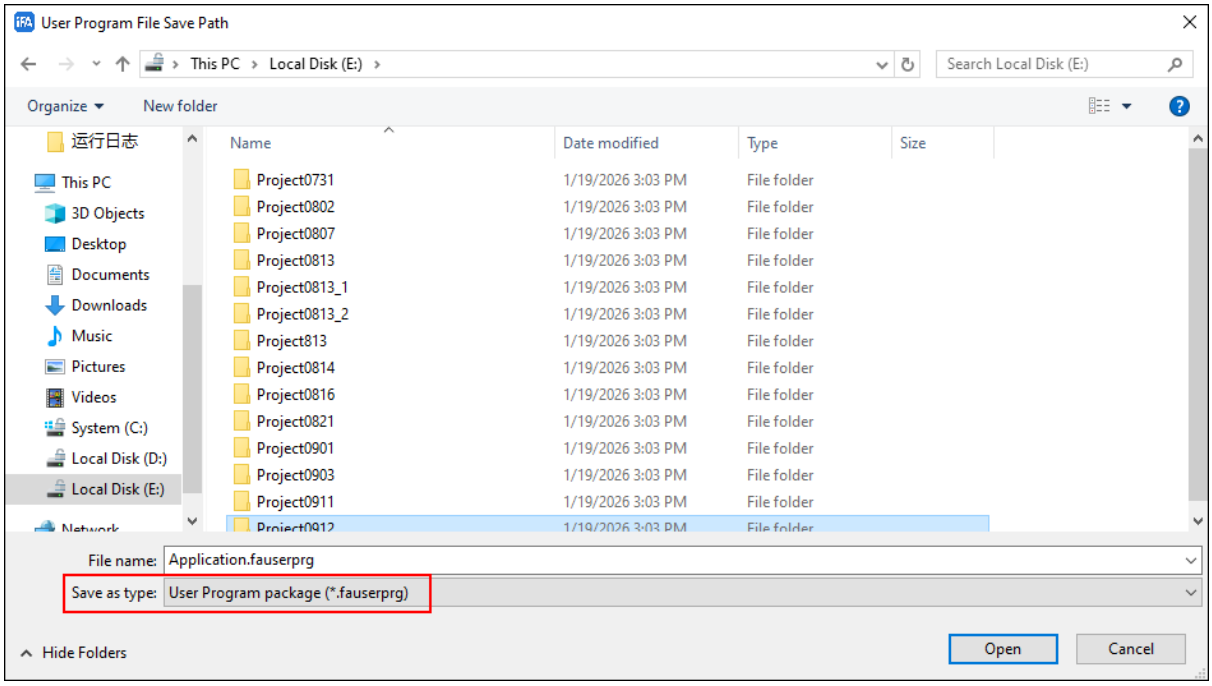
Procedure

1. Choose "Compile -> Save target file as" from the menu bar to access the "Save target file as" window.



   - PLC Type: Displays the current PLC type.
   - Firmware Version: Displays the current device's firmware version.
   - Retained Variables: Allows selecting whether to clear retentive variables during the download. The option "Retain current values of retained variables during download" is checked by default.
   - Include Project Source Code: Allows selecting whether to include the project source code in the package. This option is not checked by default. When checked, the project source code will be included in the package.
   - Compilation Progress: The interface displays the compilation progress by default. The packaging progress is displayed after the compilation is complete and the save path is selected.

2. Click "Package". A full compilation is performed by default. After successful compilation, a window for selecting the save path appears, allowing you to select a path to save the file. The save format is .fauserprg.



3. Click "Open" and follow the prompts to proceed.

> **CAUTION**
>
> Packaging is not supported in simulation mode or online mode.

14.1.2.2 Upgrading a User Program via SD Card/USB Flash Disk

EVO400, EVO500, and EVO800 series PLCs support user program upgrade through SD card, while EVO700 and EVO700C series PLCs support user program upgrade through USB flash disk.

Prerequisites

- A SD card or USB flash disk with a capacity of not more than 32 G and using the FAT32 file system is ready.
- A "PLCprogram" directory is created in the root directory of the SD card or USB flash disk, and the packaged user program is copied to this directory. For the steps to package the user program, see "14.1.2.1 Saving a PLC User Program As".

Procedure

The steps for upgrading user programs through SD card and USB flash disk are similar. The following describes the steps for upgrading user programs through SD card:

1. Disconnect both 5 V and 24 V power supplies from the PLC.
2. Insert the SD card containing the project file into the GE20-TF storage expansion card, and then insert the expansion card into the PLC.
3. Set the PLC's DIP switch to Stop state.

CAUTION

- If the controller is not in Stop state, the user program cannot be upgraded after re-power in step 4, resulting in a failed upgrade.
- After a failed upgrade, the original project in the controller will not be loaded. To restore and load it, power off the controller, remove the SD card, and then power on again.

4. Power on the PLC to automatically upgrade the user program.
   The device model, firmware version, and device permissions are verified during the upgrade. Any inconsistency in these items will cause the upgrade to fail.

   The indicator states during the upgrade are as follows:

Item	Indicator State
Upgrade process	RUN: flashing quickly (4Hz)
Upgrade success	RUN: off
Upgrade failure	ERR: flashing slowly
Program running	RUN: steady on

5. Set the DIP switch to Run state, load and run the user program, and then remove the SD card.

NOTE

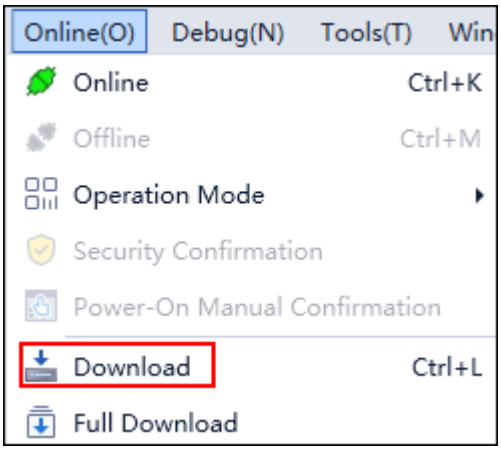
To upgrade the user program on the EVO760 series PLC using USB flash disk, insert the USB flash disk into the PLC's USB port, and then follow the steps described above.

14.1.2.3 Upgrading a User Program via Type-C/Ethernet

Only EVO400/EVO500 series PLCs support user program upgrade through Type-C and Ethernet.

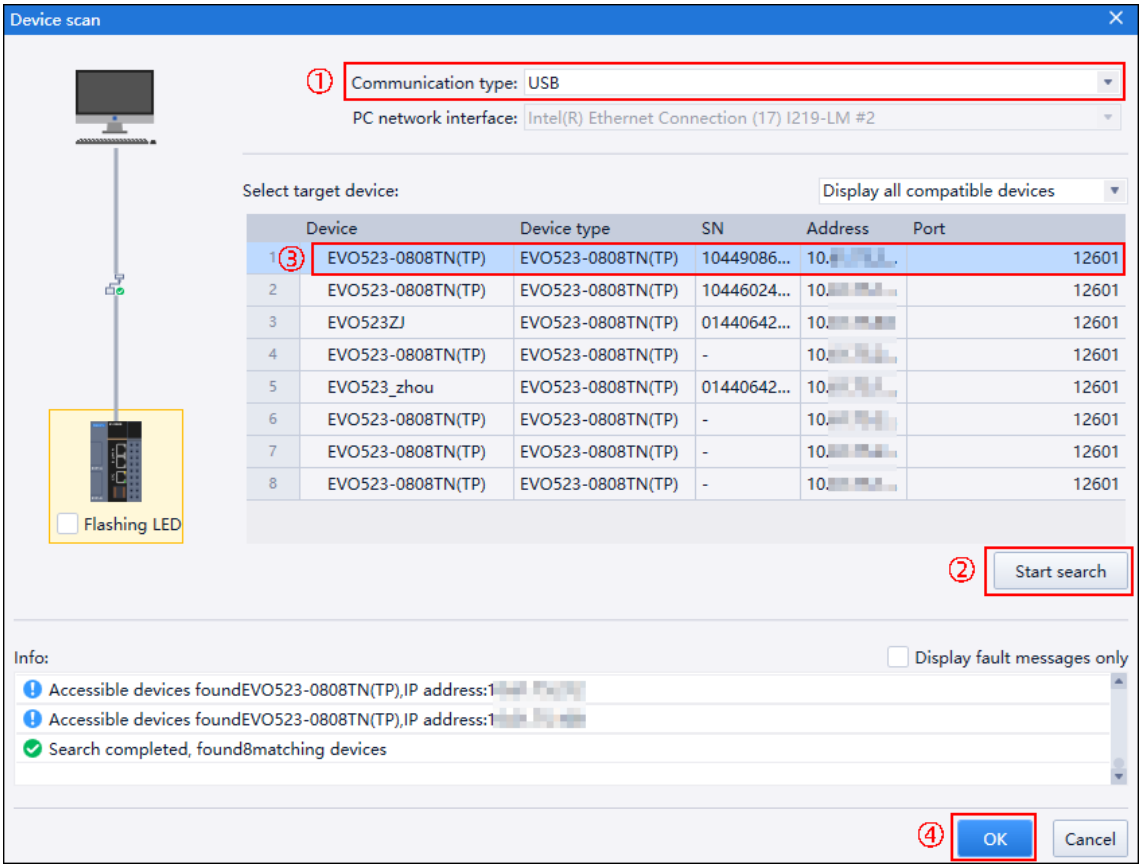
Procedure

1. Open a compiled project using the iFA Evolution.
2. Choose "Online -> Download" from the menu bar.



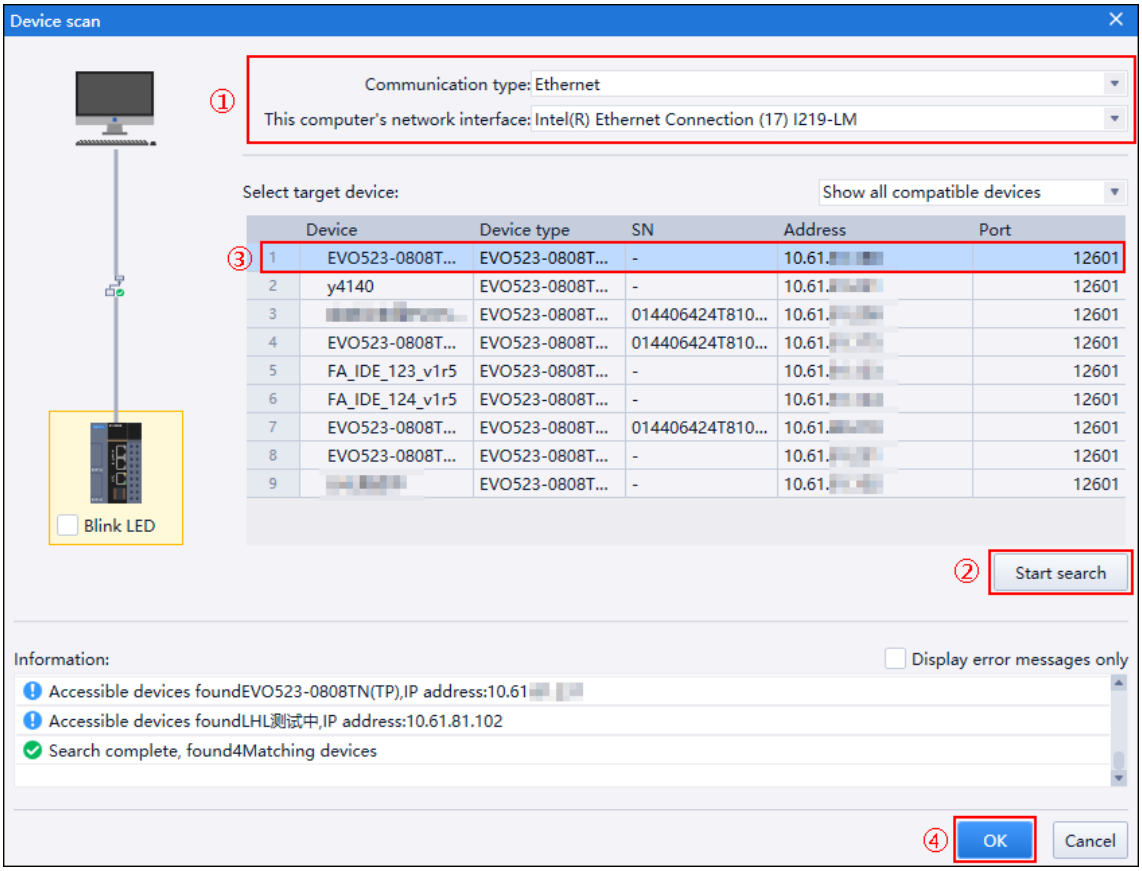
3. Scan for devices.
   - Through Type-C
     a. On the opened "Device scanning" interface, select "USB" for the "Communication type," and then click "Start search."

     b. Select the device to upgrade, and then click "OK".



   - Through Ethernet
     a. On the opened "Device scanning" interface, select "Ethernet" for the "Communication type" and the network interface connected to the device for the "This computer's network interface," and then click "Start search."

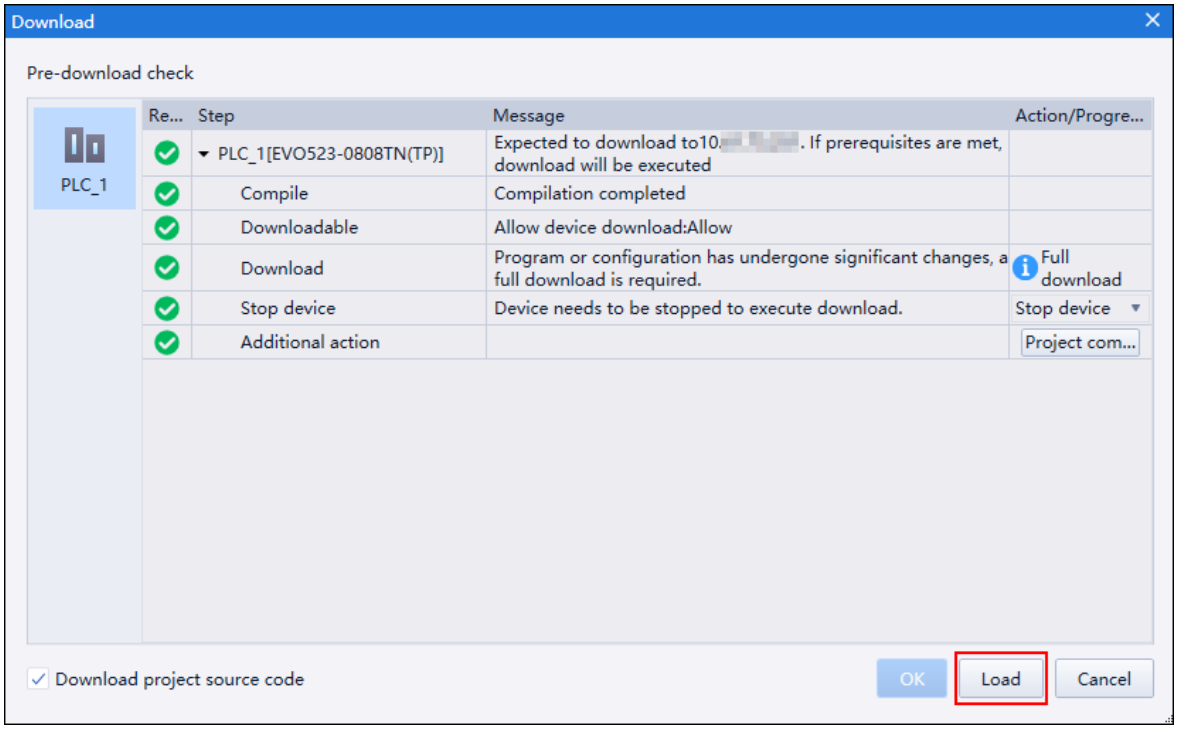
     b. Select the device to upgrade, and then click "OK".



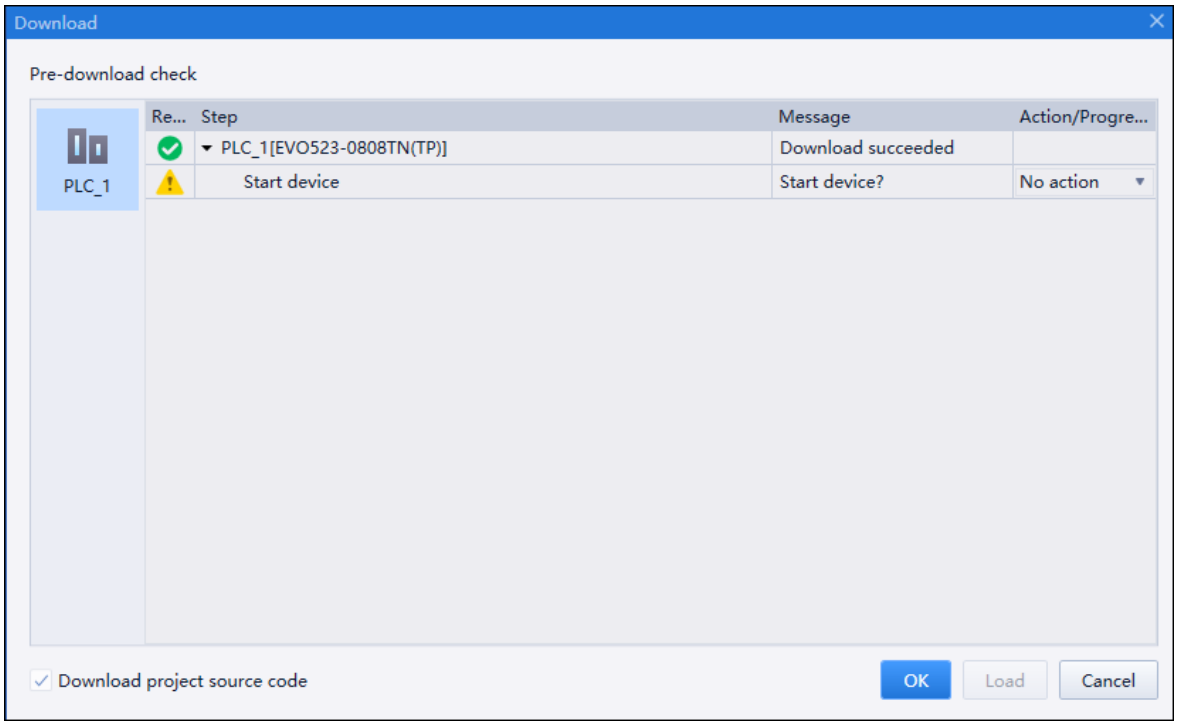
4. Download the project.

a. On the opened "Download" interface, click "Load" to download the project.

> **NOTE**
>
> The option "Download project source code" is checked by default, which means that the project source code will be downloaded to the device; if this option is not checked, the project source code will not be downloaded.



b. After successful download, select whether to start the device, and then click "OK."



14.1.2.4 Upgrading a User Program via SFTP

EVO500, EVO700, EVO760C, EVO780C, and APC series PLCs with firmware version V1.6 or later support user program upgrades via SFTP. Two upgrade methods are available; choose the appropriate one based on the actual situation:

- Immediate upgrade: The program takes effect immediately after the transfer is completed.
- Upgrade upon restart: After the program transfer is completed, manually restart the PLC for the changes to take effect.

Prerequisites

- The SFTP server function has been enabled. For specific operations, see 10.6.14.6 SFTP Server Configuration.
- The user program has been packaged. For specific operations, see 14.1.2.1 Saving a PLC User Program As.

Procedure

- Immediate upgrade

1. Upload the packaged user program to the /ssh_update directory on the PLC via SFTP.
2. After the file transfer is completed, the upgrade process is automatically triggered. The system updates the program and automatically restarts the PLC.

- Upgrade upon restart

1. Upload the packaged user program to the /ssh_update/manual_update directory on the PLC via SFTP.
2. After the file transfer is completed, manually restart the PLC. Upon power-up, the system automatically detects and completes the program upgrade.

Precautions

- The immediate upgrade automatically restarts the controller.

- For upgrade upon restart, it is recommended to wait one minute after the file transfer before restarting, to ensure the files are fully written to disk and to avoid upgrade failure.
- Regardless of the upgrade method, wait approximately one minute after the controller restarts to ensure system stability before attempting to reconnect.
- User files are not retained after either a successful or failed upgrade.
- The device status after the upgrade is determined by the DIP switch status.
- The indicator status during the upgrade is consistent with upgrading user programs via an SD card. For details, see 14.1.2.2 Upgrading a User Program via SD Card/USB Flash Disk.

14.2 HMI Firmware and Project Upgrade

14.2.1 Upgrading HMI Firmware

The firmware version of the HMI device is closely related to the version of the iFA software. The HMI firmware may need to be upgraded synchronously during software upgrade to ensure compatibility and functional integrity. The firmware upgrade requires a dedicated firmware package, so it is essential to obtain and use the correct version.

Requirements

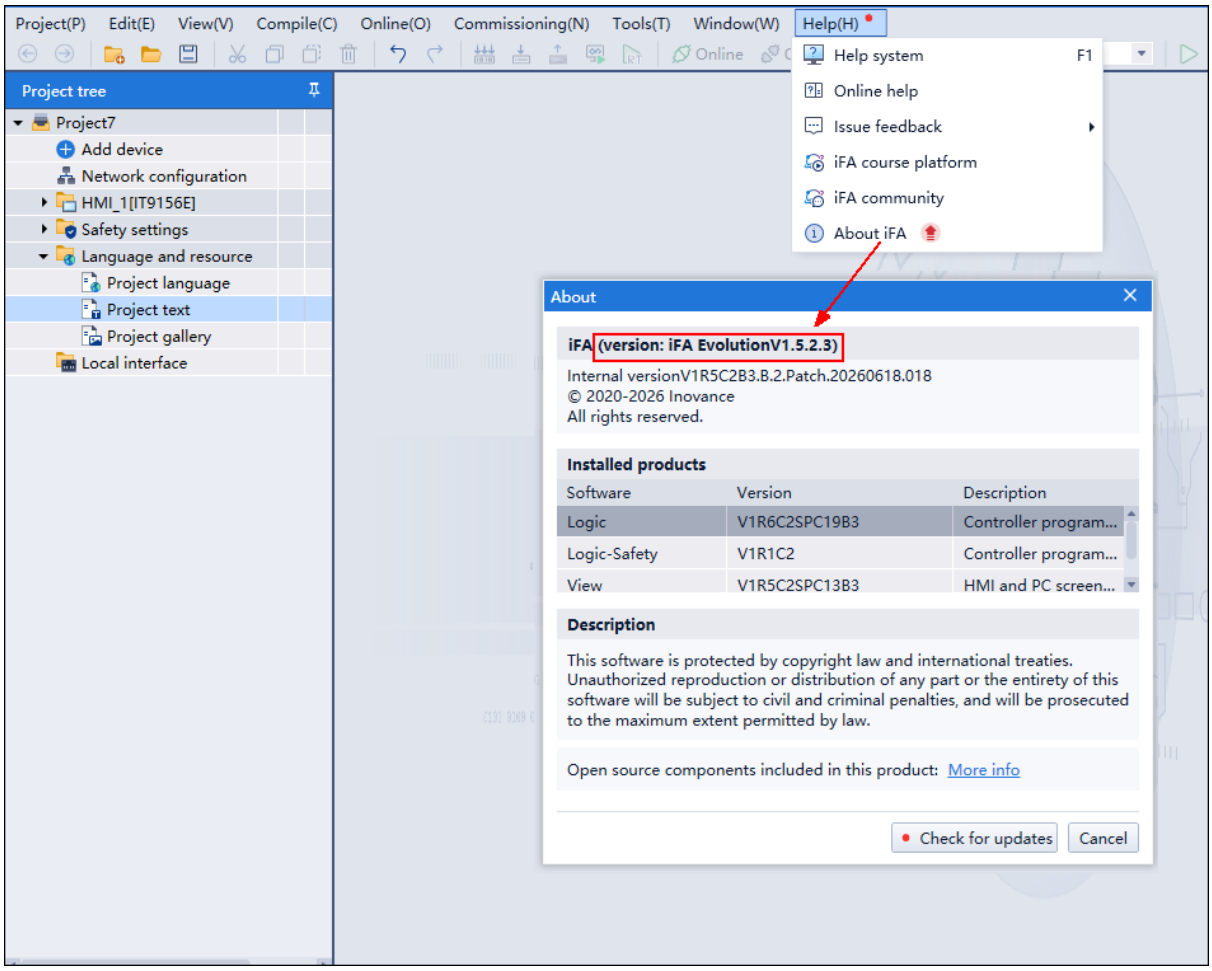
Back up the project data before upgrading the touchscreen firmware.

Obtaining the firmware package

- Firmware package format
  The firmware package format is changed as follows from iFA Evolution V1.4.4.5:
  - Before V1.4.4.5: The firmware packages are provided as .zip files.
  - V1.4.4.5 and later: The firmware packages are provided as .upgrade files.
- How to obtain

1. Check the iFA version number

Select "Help > About iFA".



2. Download the firmware package

   Access the website https://forum.inovance.com/forum?id=562, and then download the firmware package based on the iFA version number.

Installing the firmware package

1. Enable the HMI firmware upgrade command.

   Select "Online > Firmware update > PLC and HMI" from the menu bar.

2. Scan for devices.

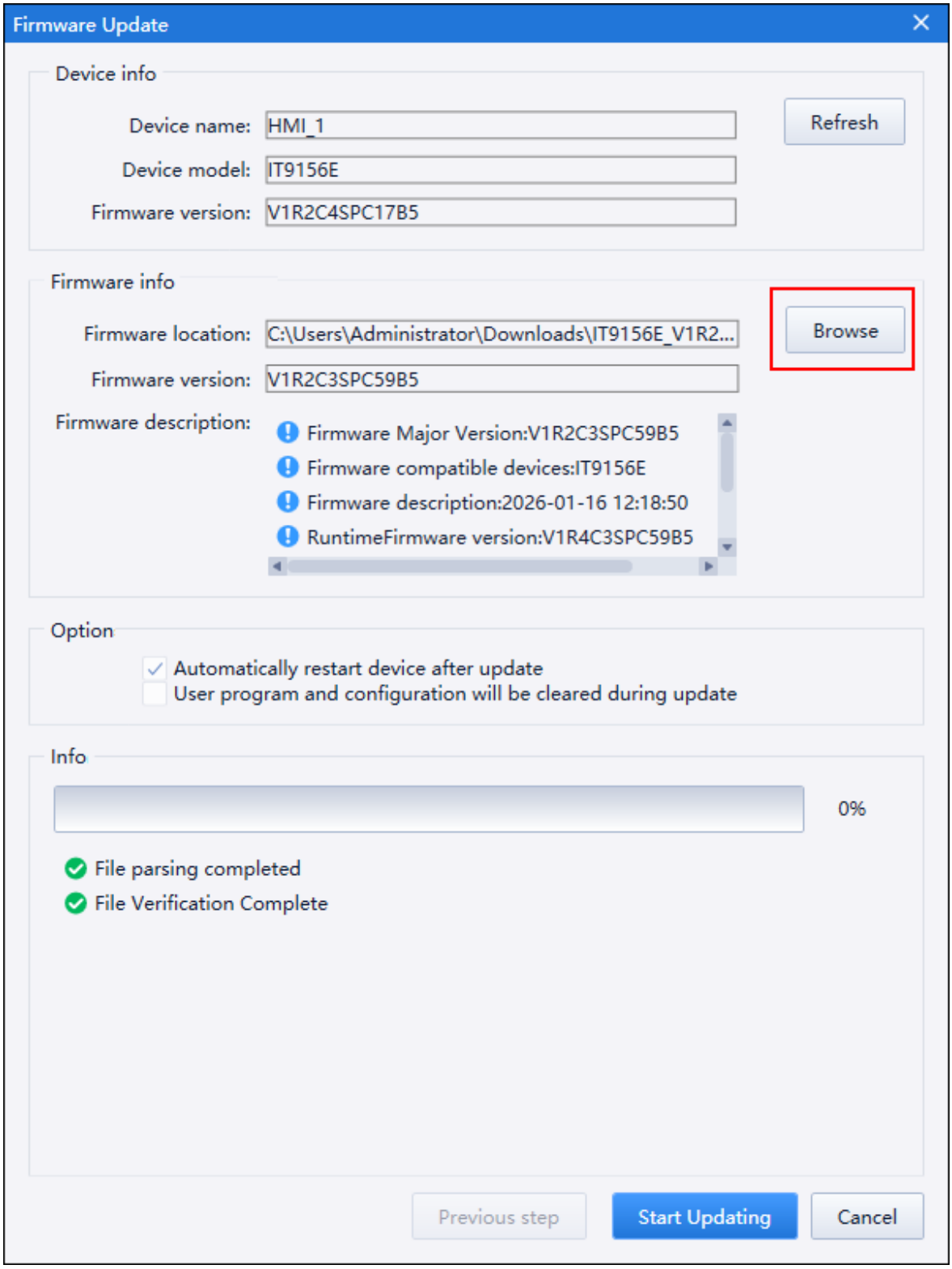
   On the opened "Device scanning" interface, specify the communication type and then complete the scan.

   - Ethernet: All devices on the network segment can be identified.
   - USB: Only one device can be identified at a time. Even if multiple devices are connected simultaneously, the system will randomly select one for scanning.

3. Select the target device, click "OK", and then the system automatically opens the "Firmware Update" interface.

4. Configure the upgrade object and method

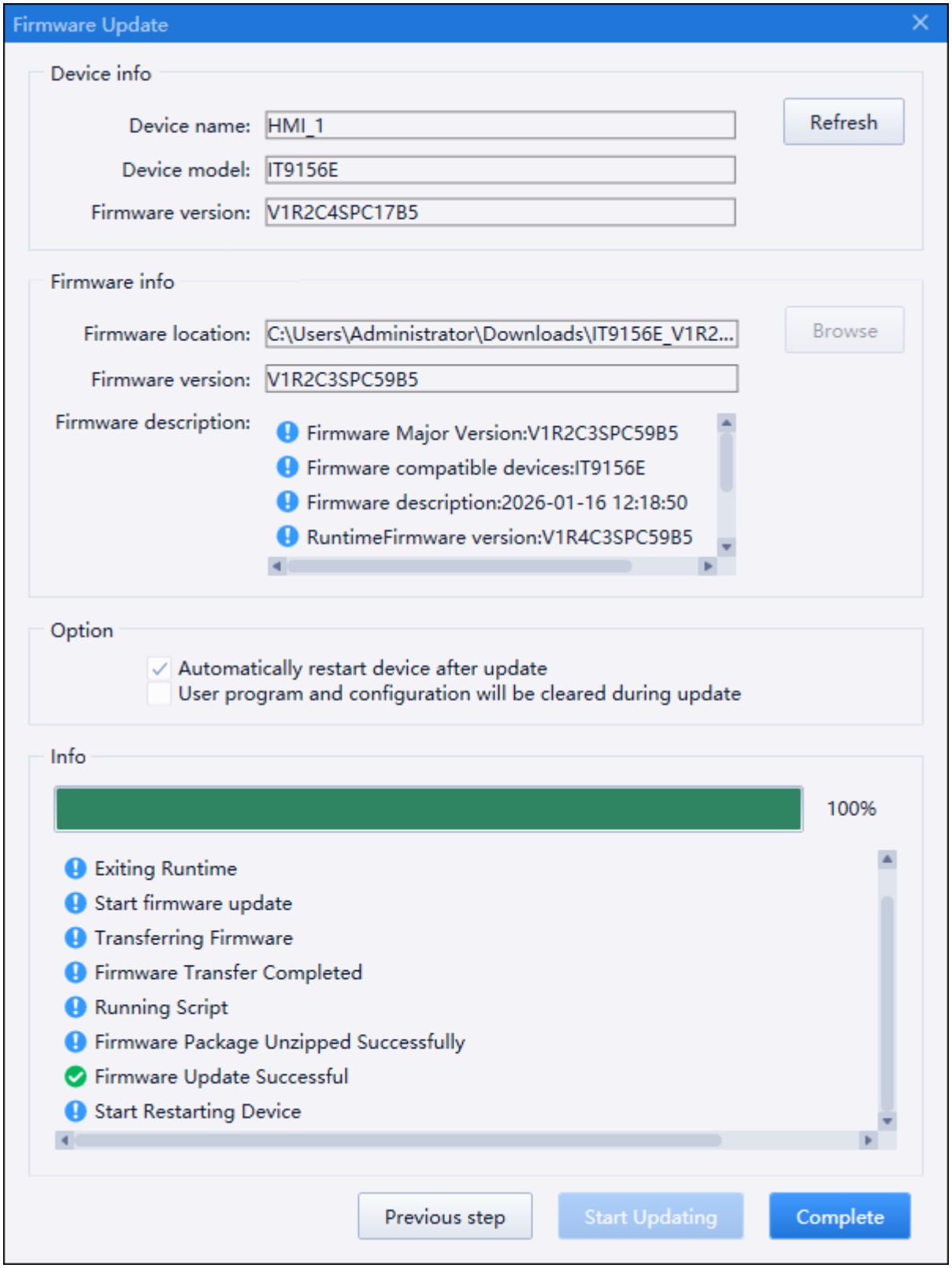
   a. Load the downloaded firmware package



b. Set the upgrade method

- Restart device after update: The system automatically restarts the device after successful update. This process does not affect the user project data.
- Clear user program and configuration during update: The user projects and data on the device are cleared during the update, and the control panel configuration is restored to factory settings.

5. Start updating



6. Check the version information

   Repeat Steps 1 to 3 to obtain the device information and check whether the firmware version is updated.

CAUTION

- Downgrading the firmware will erase user projects and settings on the device. Ensure all important information is backed up before the operation to prevent data loss.
- Ensure that the HMI device remains powered on during the firmware update and device restart.

Recovery after firmware update failure

In case of unexpected power failure or disconnected interface (USB/network port) during firmware update, the touchscreen may fail to start.

Follow these steps to recover the firmware update:

1. Press the RST button on the touchscreen to restore the touchscreen to factory settings.
2. After the HMI starts up normally, repeat the firmware upgrade process.

Risk warning: Restoring to factory settings will erase all projects and data on the touchscreen. To prevent data loss, back up data before the firmware upgrade.

14.3 I/O Firmware Upgrade

14.3.1 Upgrading Hi50 Series Safety I/O Module Firmware

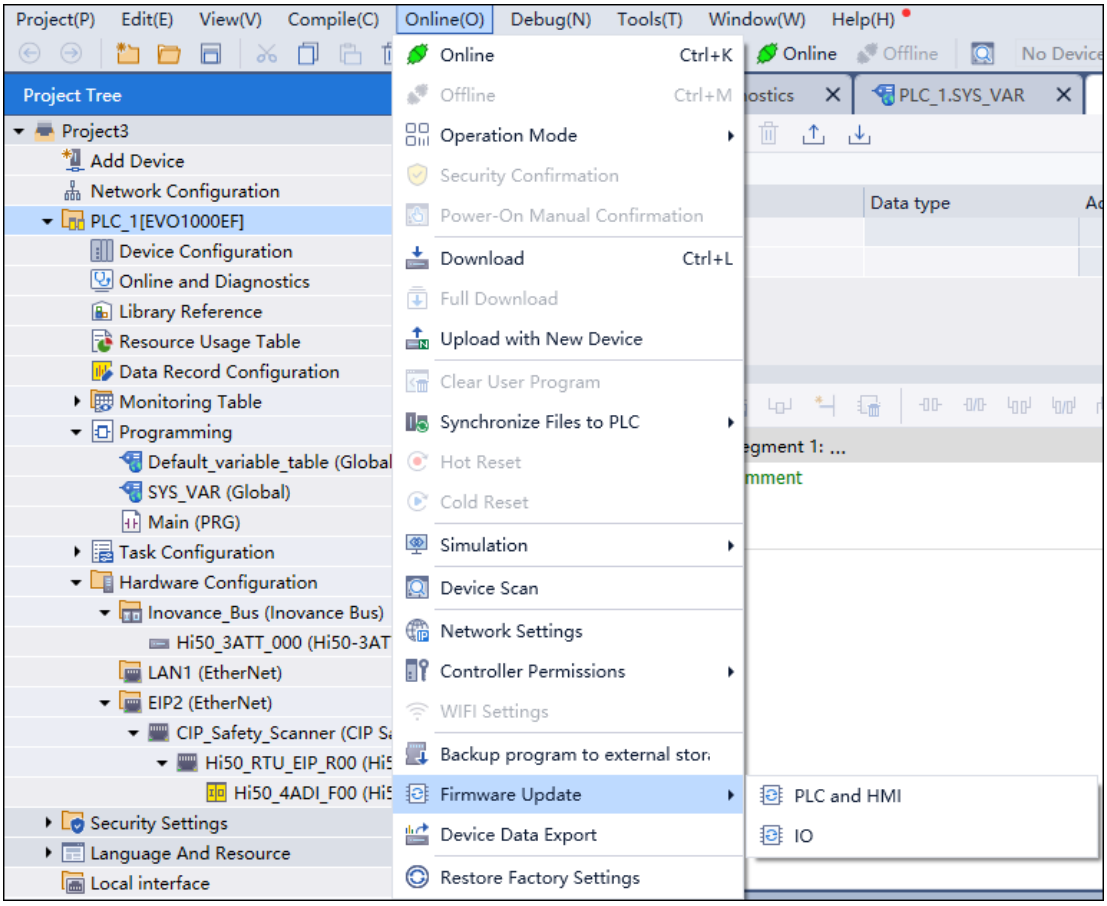
The combined application of the iFA Evolution software and the EVO1000EF series PLC system introduces the firmware upgrade procedure which is the same for the Hi50-RTU-ECT-R00 and Hi50-RTU-EIP-R00 modules. This section takes Hi50-RTU-ECT-R00 as an example.

Prerequisites

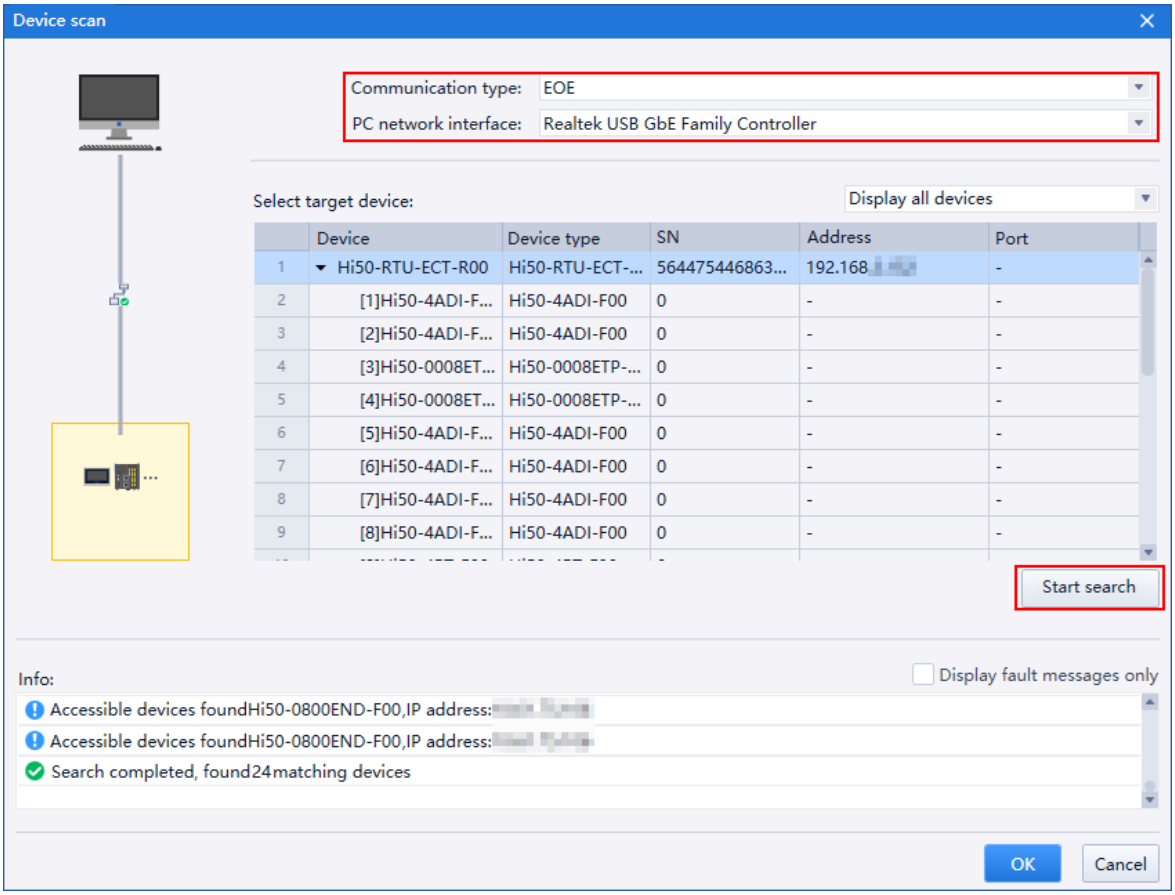
- The hardware installation and wiring in the configuration example is complete.
- All the devices in the configuration example are powered on and started.
- The PC installed with iFA programming software (V1R4C2B3.R.4 or later) is connected via network cable to the EIP port of the EVO1000EF.
- The IN port of the Hi50-RTU-ECT-R00 module is connected to the PC via a network cable.

Procedure

1. Choose "Online > Firmware Update > IO".

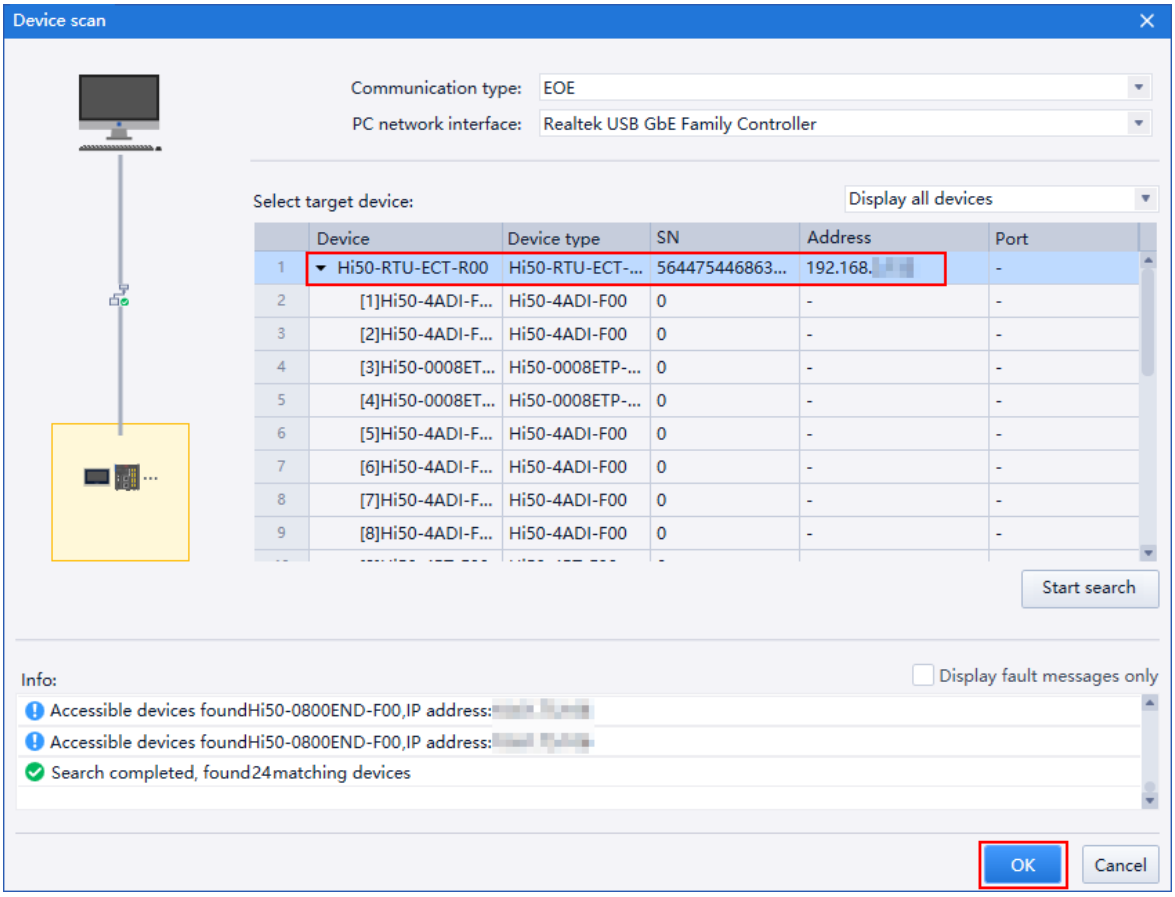


2. Select "Communication type" and "This computer's network interface", and then click "Start search".

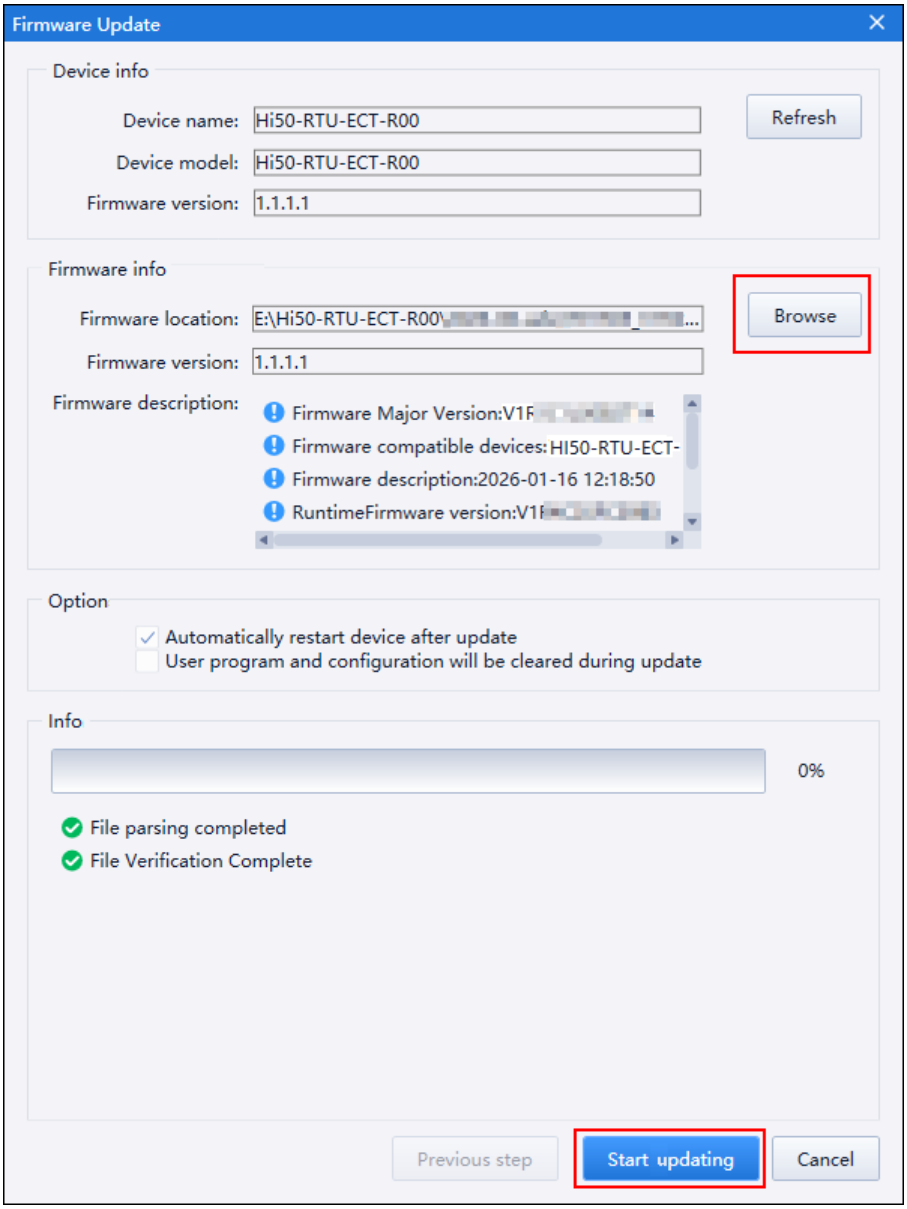


> **NOTE**
>
> For the Hi50-RTU-EIP-R00 module, set the "Communication type" to "Ethernet".

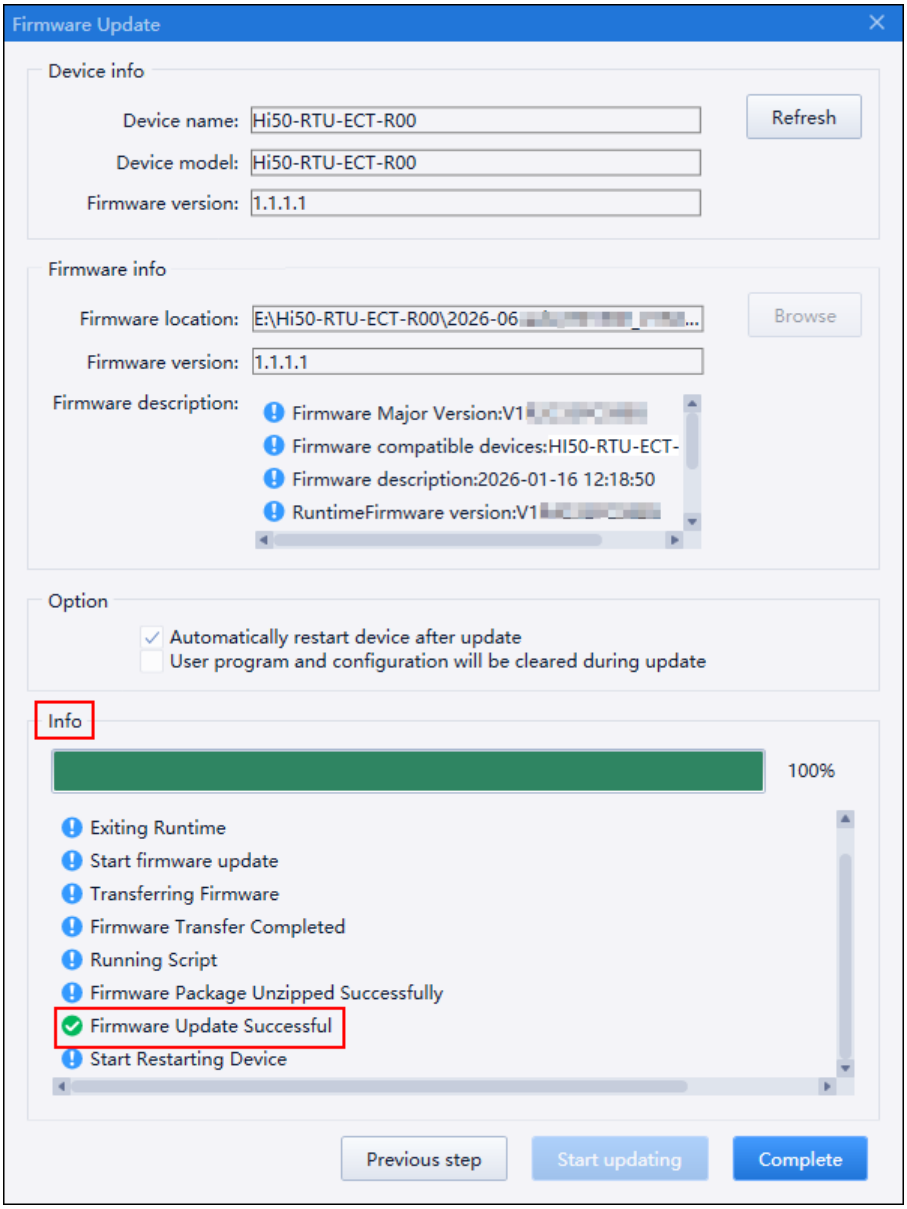
3. Select the module to be upgraded, and then click "OK."



4. Click the "Browse" button after the "Firmware location", select the firmware path for the module, and then click "Start updating".



5. When the firmware upgrade progress under "Information" shows 100% and the message "Firmware update successful" appears, click "Complete."



6. Power on the entire configuration system.

14.3.2 Upgrading Hi80 Series I/O Firmware

Introduction

The EIP communication interface module and the I/O module share the same firmware upgrade method. This section describes the firmware upgrade of the EIP communication interface module.

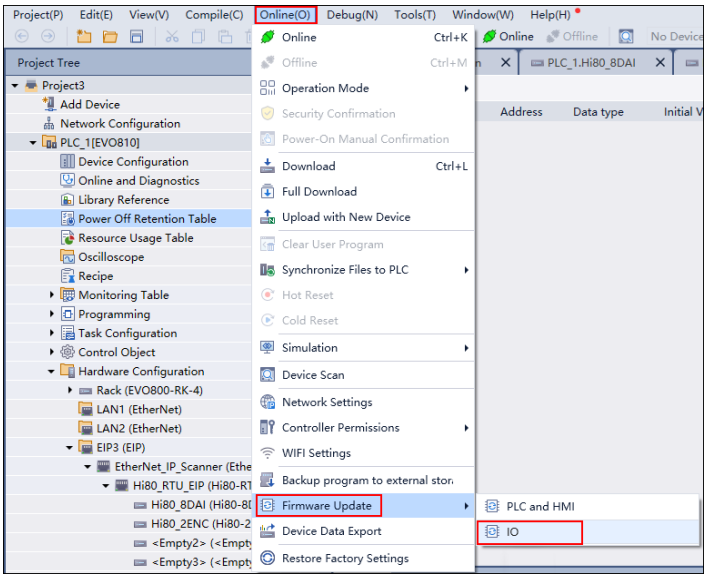
Prerequisites

- The firmware to be upgraded (Hi80-RTU-EIP_VxxRxxBxxTxx.upgrade or Hi80-RTU-EIP_VxxRxxBxxTxx.bin) has been obtained from the technical service personnel of Inovance.
- The PC installed with the iFA Evolution programming software is connected via network cable to the EIP communication interface module.
- The device is powered on and started.

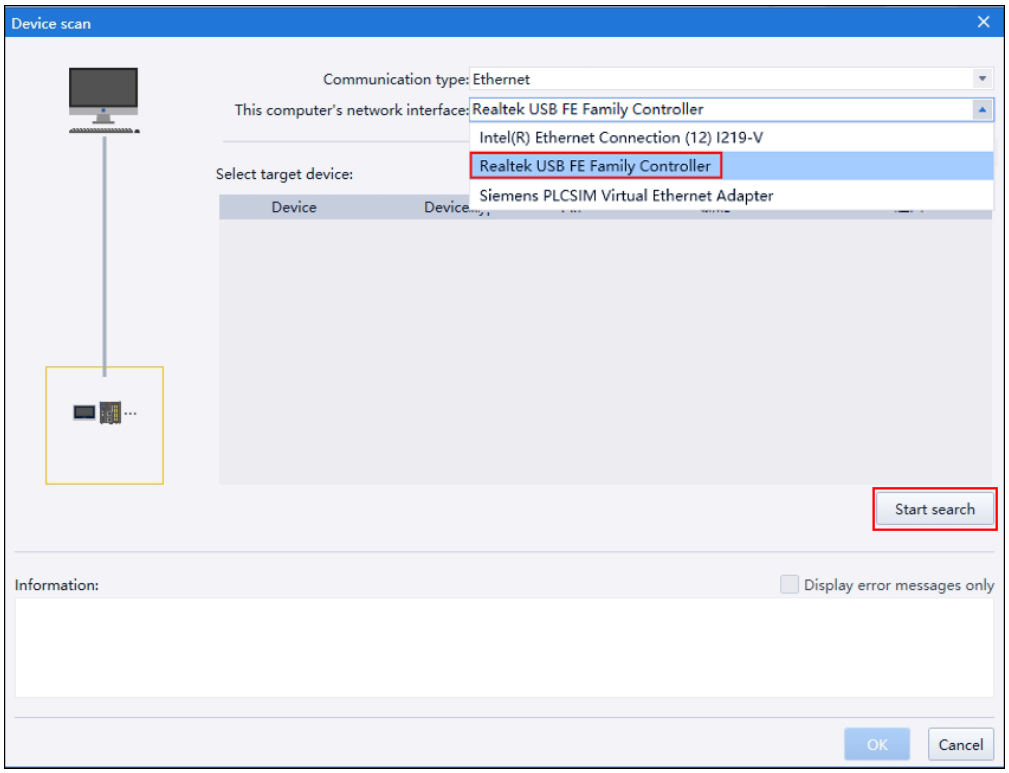
Procedure

1. Open the iFA Evolution programming software on the PC desktop and open the project.

2. Click "Online" and then choose "Firmware Update > IO".



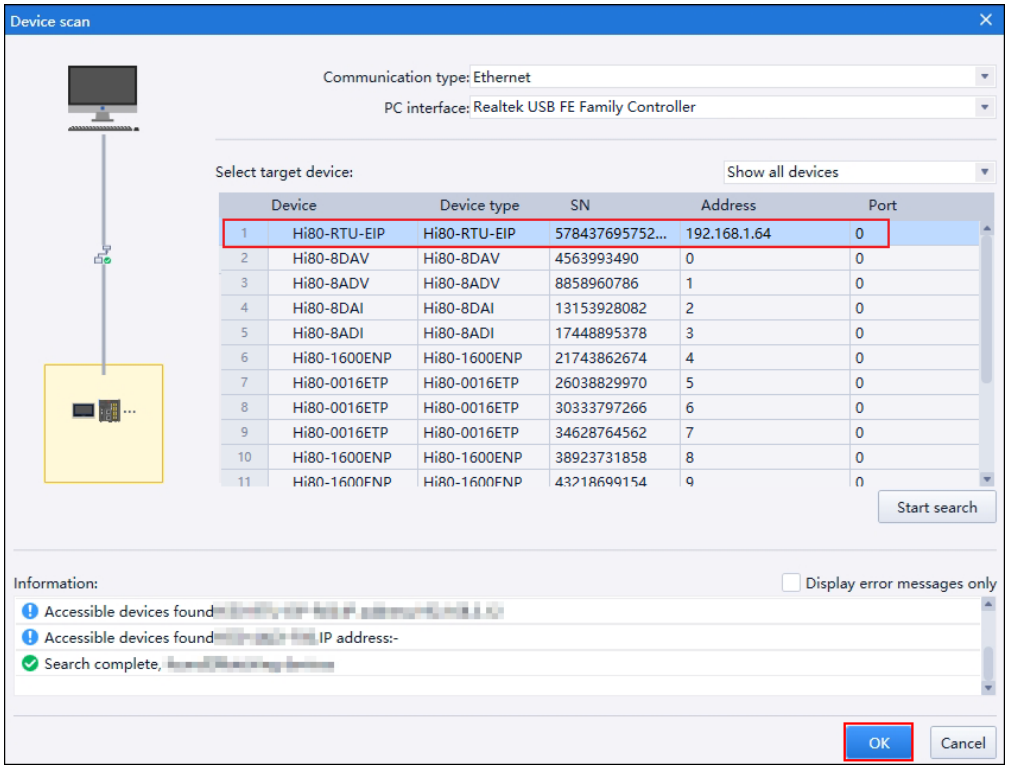
3. Select "This computer's network interface," and then click "Start search".



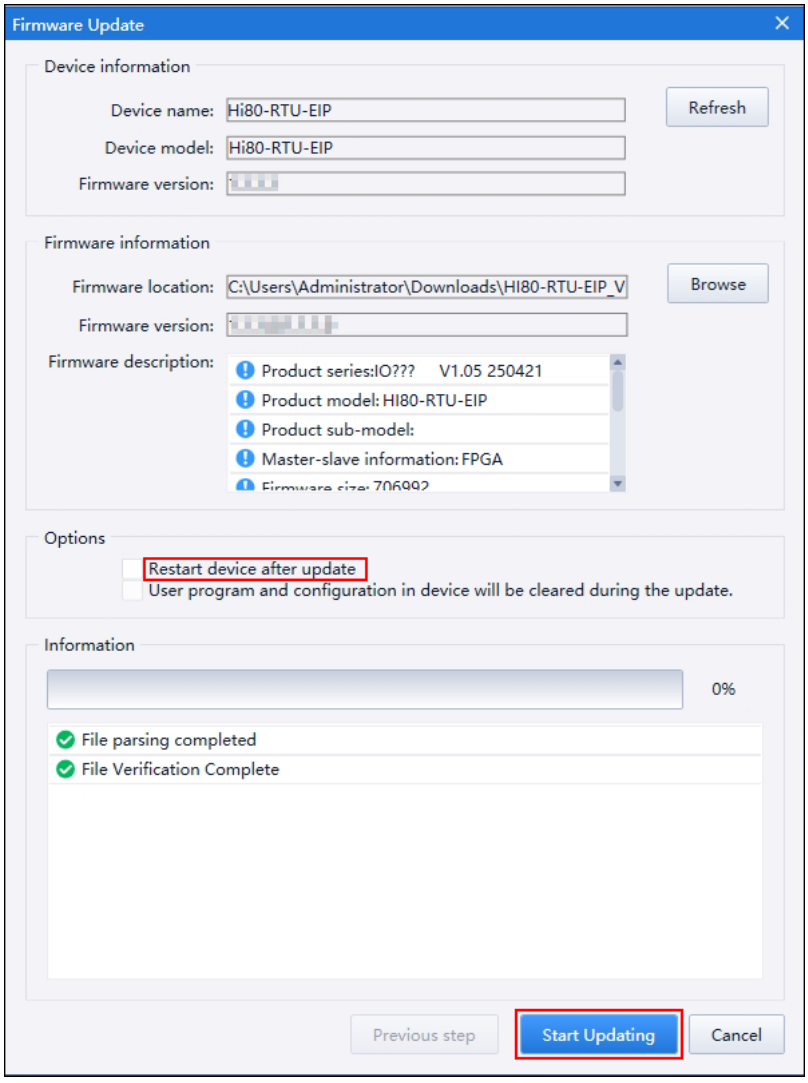
CAUTION

The network interface must be configured in the same network segment as the Hi80-RTU-EIP communication interface module.

4. Select the target device, and then click "OK."



5. Select the upgrade file, and then click "Start updating". If you want the upgrade to take effect immediately after completion, check the option "Restart device after update," as shown in the figure below.



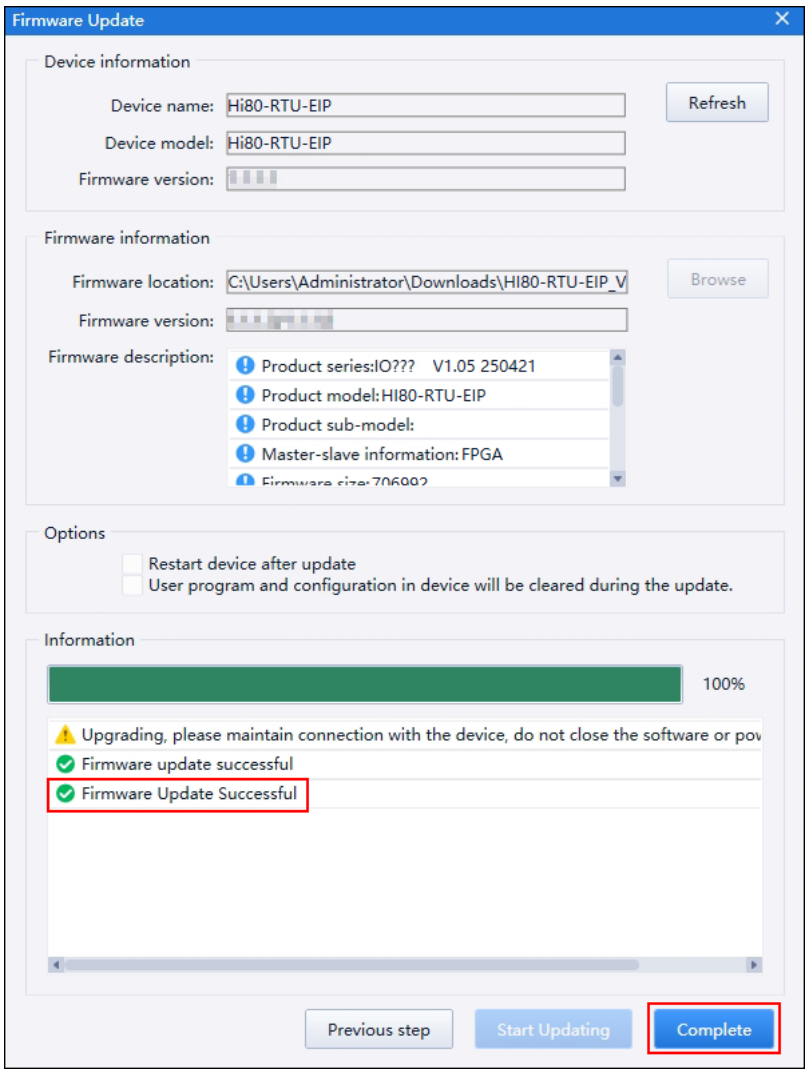
NOTE

- Firmware version: V1.1.25.0 and later
- If the option "Restart device after update" is not checked, the upgrade will take effect only after a power cycle on the system side.
- During firmware upgrade, the option "Clear user program and configuration during the update" cannot be checked.

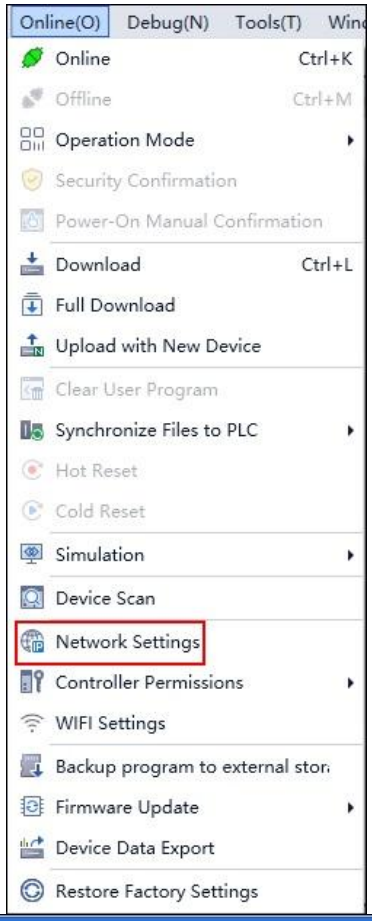
CAUTION

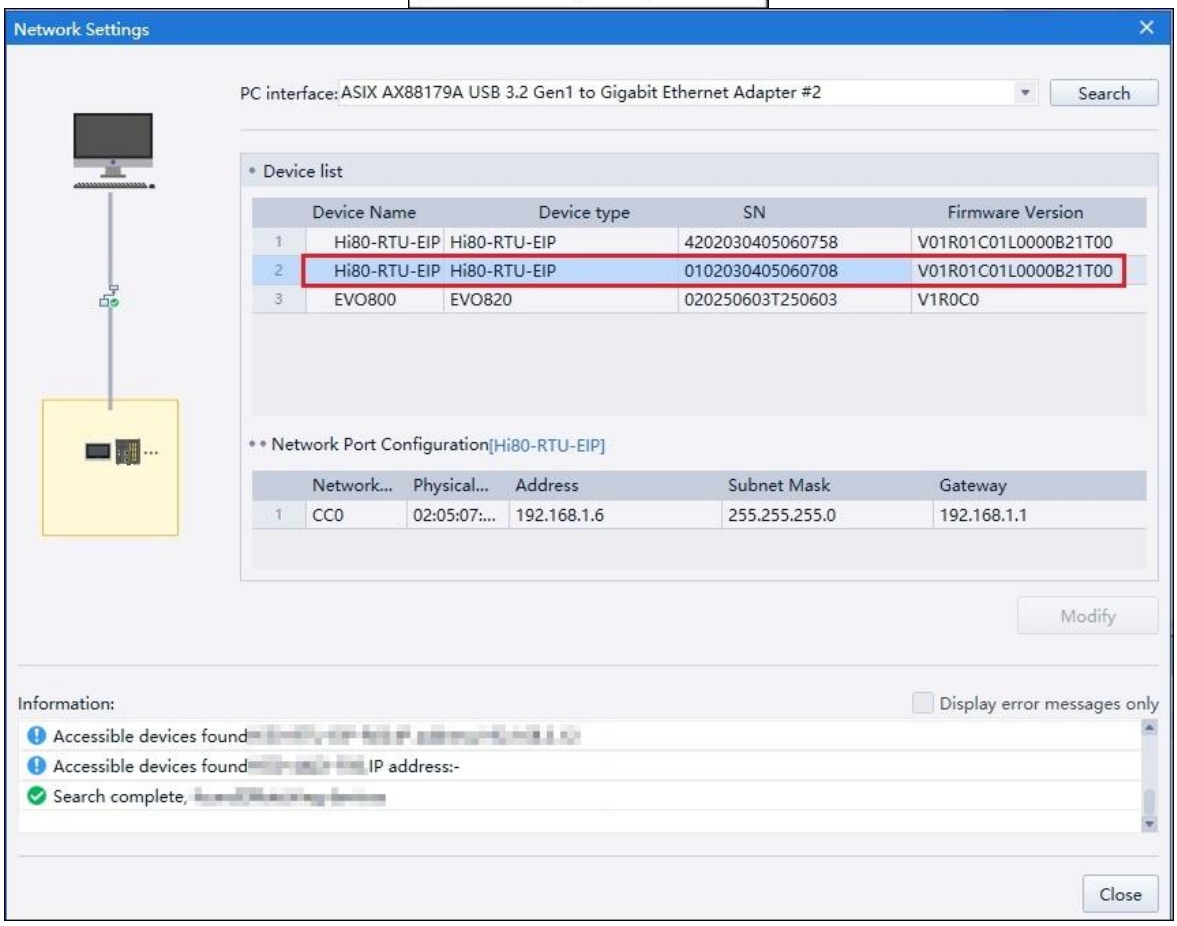
- LED indicator status during EIP communication interface module firmware upgrade: PWR - steady on; NET - flashing; MOD, SF, and A/B - steady off.
- LED indicator status during I/O module firmware upgrade: PWR - steady on; RUN - flashing; ERR - steady off.

6. Click "Complete" after "Firmware update successful" is displayed.



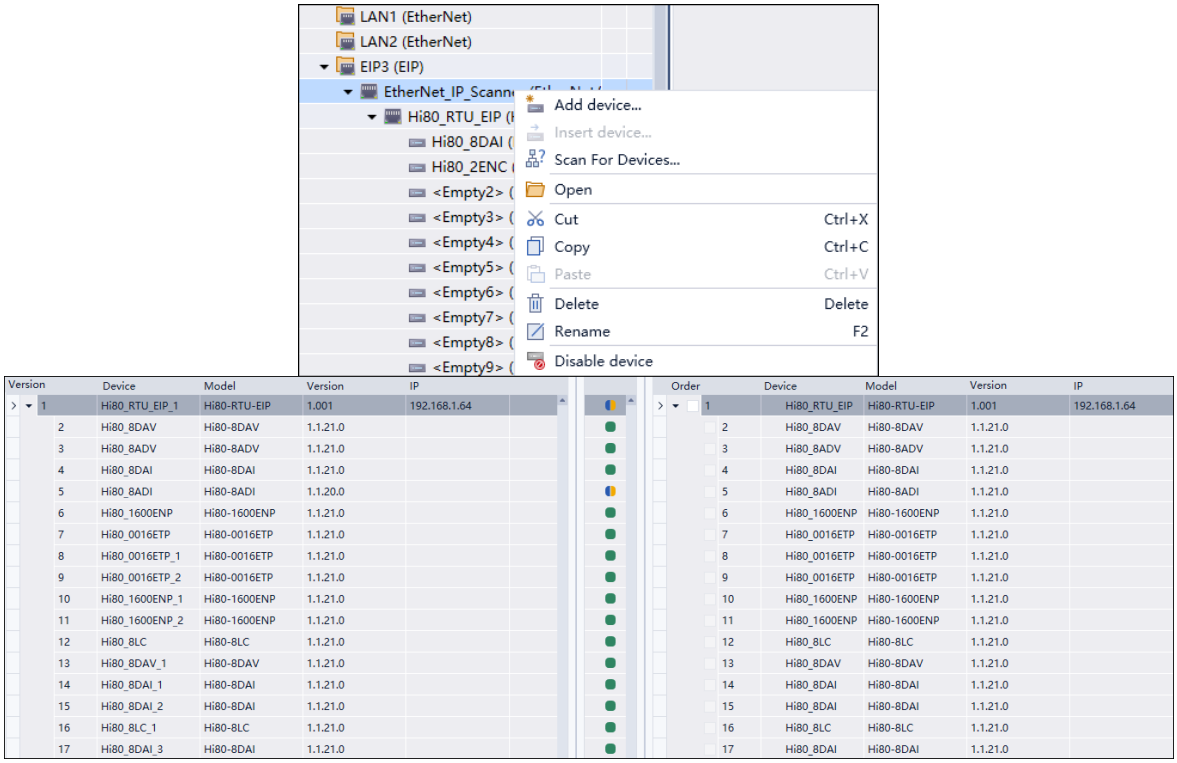
7. After the upgrade, check that the version information matches the expected version.
   - View the communication interface module version.
     Choose "Online (O) > Network setting".





◦ View the I/O module version.

Right-click "EtherNet_IP_Scanner" and select "Scan for devices" to view the version information.



Version	Device	Model	Version	IP
1	Hi80_RTU_EIP_1	Hi80-RTU-EIP	1.001	192.168.1.64
2	Hi80_8DAV	Hi80-8DAV	1.1.21.0	
3	Hi80_8ADV	Hi80-8ADV	1.1.21.0	
4	Hi80_8DAI	Hi80-8DAI	1.1.21.0	
5	Hi80_8ADI	Hi80-8ADI	1.1.20.0	
6	Hi80_1600ENP	Hi80-1600ENP	1.1.21.0	
7	Hi80_0016ETP	Hi80-0016ETP	1.1.21.0	
8	Hi80_0016ETP_1	Hi80-0016ETP	1.1.21.0	
9	Hi80_0016ETP_2	Hi80-0016ETP	1.1.21.0	
10	Hi80_1600ENP_1	Hi80-1600ENP	1.1.21.0	
11	Hi80_1600ENP_2	Hi80-1600ENP	1.1.21.0	
12	Hi80_8LC	Hi80-8LC	1.1.21.0	
13	Hi80_8DAV_1	Hi80-8DAV	1.1.21.0	
14	Hi80_8DAI_1	Hi80-8DAI	1.1.21.0	
15	Hi80_8DAI_2	Hi80-8DAI	1.1.21.0	
16	Hi80_8LC_1	Hi80-8LC	1.1.21.0	
17	Hi80_8DAI_3	Hi80-8DAI	1.1.21.0	

Order	Device	Model	Version	IP
1	Hi80_RTU_EIP	Hi80-RTU-EIP	1.001	192.168.1.64
2	Hi80_8DAV	Hi80-8DAV	1.1.21.0	
3	Hi80_8ADV	Hi80-8ADV	1.1.21.0	
4	Hi80_8DAI	Hi80-8DAI	1.1.21.0	
5	Hi80_8ADI	Hi80-8ADI	1.1.21.0	
6	Hi80_1600ENP	Hi80-1600ENP	1.1.21.0	
7	Hi80_0016ETP	Hi80-0016ETP	1.1.21.0	
8	Hi80_0016ETP	Hi80-0016ETP	1.1.21.0	
9	Hi80_0016ETP	Hi80-0016ETP	1.1.21.0	
10	Hi80_1600ENP	Hi80-1600ENP	1.1.21.0	
11	Hi80_1600ENP	Hi80-1600ENP	1.1.21.0	
12	Hi80_8LC	Hi80-8LC	1.1.21.0	
13	Hi80_8DAV	Hi80-8DAV	1.1.21.0	
14	Hi80_8DAI	Hi80-8DAI	1.1.21.0	
15	Hi80_8DAI	Hi80-8DAI	1.1.21.0	
16	Hi80_8LC	Hi80-8LC	1.1.21.0	
17	Hi80_8DAI	Hi80-8DAI	1.1.21.0	

15 Diagnostics

15.1 Overview

The diagnostics function is primarily used to quickly locate various errors that occur during the operation of PLCs and other devices. By parsing error information and equipment operating status, it helps users identify and implement corresponding solutions.

All PLC devices have built-in diagnostics functions to detect system faults in real time and assist in troubleshooting. This function can automatically identify anomalies that may occur during operation and provide detailed fault-related information.

Diagnostics features

- The programming system supports diagnostics for various communication devices, and can generate fault and other information based on the actual operating status of each device.
- The diagnostic range mainly involves CPU modules and other types of device modules.
- Fault causes and corresponding solutions are output in text format.
- Supports archiving and recording of alarm information.

15.2 Quick Guide

15.2.1 Diagnostics via Segment Code Screen

The segment code screen allows quick and direct reading of fault codes and other information. Only PLC devices with a segment code screen support this diagnostic method.

Prerequisites

- A project has been created.
- The project has been downloaded to the PLC device.

Procedure

Take the EVO800 series as an example, follow the steps below to determine the diagnostic information using the segment code screen:

1. Observe the status and fault code values displayed on the segment code screen.
2. Look up the corresponding fault code solutions in the software or documentation.



15.2.2 Diagnostics via iFA Software

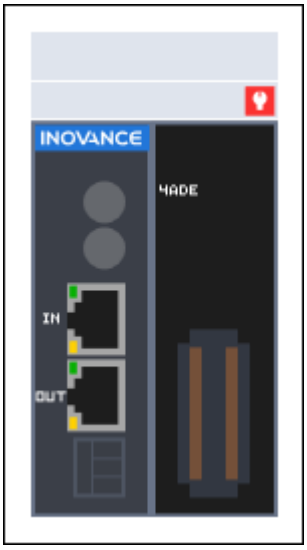
Detailed diagnostic information can be quickly accessed via iFA Evolution.

Prerequisites

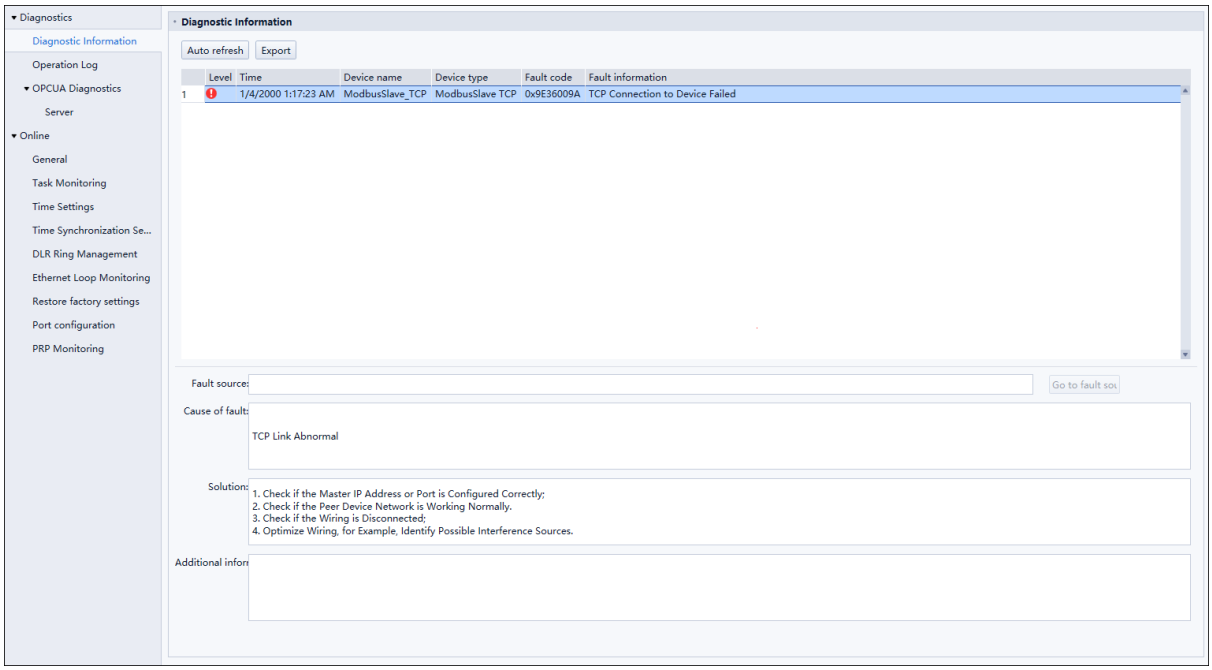
- A project has been created.
- The project has been downloaded to the PLC device.
- A fault has occurred.
- The programming device can be connected to the PLC device via the interface.
- The device has been switched to online.

Procedure

1. Double-click "Device configuration" in the left project tree and go to the "Network configuration" tab.
2. On the "Device configuration" page, view the device fault info.



3. In the "Diagnostic info" section, all diagnostic events are displayed in the order in which the events occurred.



15.3 Display of System Diagnostics

15.3.1 Viewing Diagnostic Information via Hardware Interface

15.3.1.1 Device Indicators

LED indicators on PLC devices and their expansion modules serve as critical diagnostic tools. They convey device operating status, fault information, and external communication conditions through different colors and flashing frequencies.

The meaning, combination, and fault resolution measures of LED indicators vary across different device models. The following describes the indicator meanings for some common models. For details on other devices or modules, see their respective device guides.

Indicators of the EVO500 series



Name	Label	Status	Note
Power supply indicator	Power	Steady green	Power supply normal
		Steady off	Power supply fault
Operation indicator	RUN	Steady green	The module operation is normal.
		Flashing	Firmware or SD card program upgrade
		Steady off	The module operation is faulty.
Fault indicator	ERR	Steady red	Module fault
		Steady off	The module is in normal state.
Network port indicator 1A	LAN1A	Steady green	Connection established successfully
		Flashing	Communicating
		Steady off	Connection not established

Name	Label	Status	Note
Network port indicator 1B	LAN1B	Steady green	Connection established successfully
		Flashing	Communicating
		Steady off	Connection not established

Indicators of the EVO800 series redundant module



Name	Label	Status	Note
Power supply indicator	Power	Steady green	Module power supply normal
		Steady off	The module power supply is faulty.
Operation indicator	RUN	Steady green	The module operation is normal.
		Steady off	The module operation is faulty.
Fault indicator	ERR	Steady red	Module fault
		Steady off	The module is in normal state.
Redundancy status indicator	OK	Steady green	Rack redundancy status normal
		Steady off	Rack redundancy status abnormal
Fiber 1 indicator	SFP1	Steady yellow	SFP1 connection normal
		Steady off	SFP1 connection abnormal
Fiber 2 indicator	SFP2	Steady yellow	SFP2 connection normal
		Steady off	SFP2 connection abnormal
Rack A indicator	A	Steady green	The rack setting switch is set to A, indicating that this is Rack A.
		Steady off	The rack setting switch is set to B.
Rack B indicator	B	Steady green	The rack setting switch is set to B, indicating that this is Rack B.
		Steady off	The rack setting switch is set to A.

15.3.1.2 Segment Code Screen

For devices equipped with a segment code display (such as the EVO700 and EVO800 series), users can directly read the device's operating status and fault codes from the screen. By referencing these fault codes, users can quickly look up the corresponding troubleshooting steps to efficiently identify and resolve issues.

The content displayed, icon meanings, and troubleshooting steps associated with fault codes on the segment code display vary by device model. See the specific device guide for accurate information. This section uses the EVO800 series as an example to illustrate the segment code display. For details on icon meanings and fault codes, see the device guide.



Figure 15-3-1-2-1 EVO800 series segment code display

15.3.2 Viewing Diagnostic Information via iFA Software

15.3.2.1 Diagnostic Symbol Description

Diagnostic symbols for devices and modules

When establishing an online connection to a device in the iFA software, users can view the diagnostic status of the device and its downstream devices, as well as the device's operating mode. The following table lists the related symbols and their meanings.

Operator	Description
	Connection to the module or device is being established.
	The device cannot be accessed via the configured address.
	The password dialog box was terminated without entering the correct password during the online connection to a protected device.
	No fault
	Fault/Error on downstream component
	Fault
	Diagnosis cannot be performed or can only be partially performed.
	The module or device (below the CPU) cannot be accessed from the CPU.
	The functionality of this module or submodule is no longer available (such as input and output data).
	Connection established, but module status is undetermined or unknown (common with third-party IO-Link devices or coupler modules).
	The configured module does not support displaying the diagnostic status.

Operator	Description
	The module or device is disabled.
	Hardware error on downstream component: at least one downstream hardware component is faulty.
	The library has been uninstalled.

Status comparison symbols

Diagnostic symbols can be used in combination with small symbols indicating online/offline comparison results. The following table lists the available comparison symbols and their meanings.

Operator	Description
	Online and offline programs are inconsistent.
	Show differences only
	Differences in subnodes
	Exists online only
	Exists offline only
	Online and offline programs are consistent.
	Show matches and differences
	Show all results
	Show online only

Symbols for device operation mode

To quickly identify the device's current operating status, the table below lists symbols related to device operating modes and their meanings.

Operator	Description
	Initial status
	Run
	Stop

15.3.2.2 Device View and Network View

After the device goes online, current status information of the device can be viewed in either device configuration or network configuration.

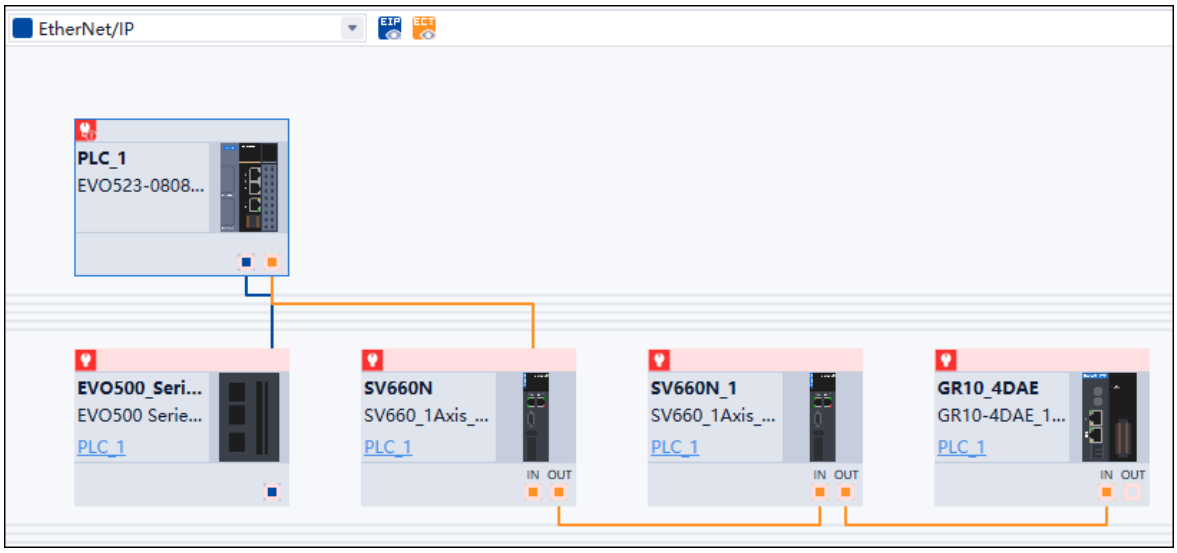
Prerequisites

The device has been switched to online.

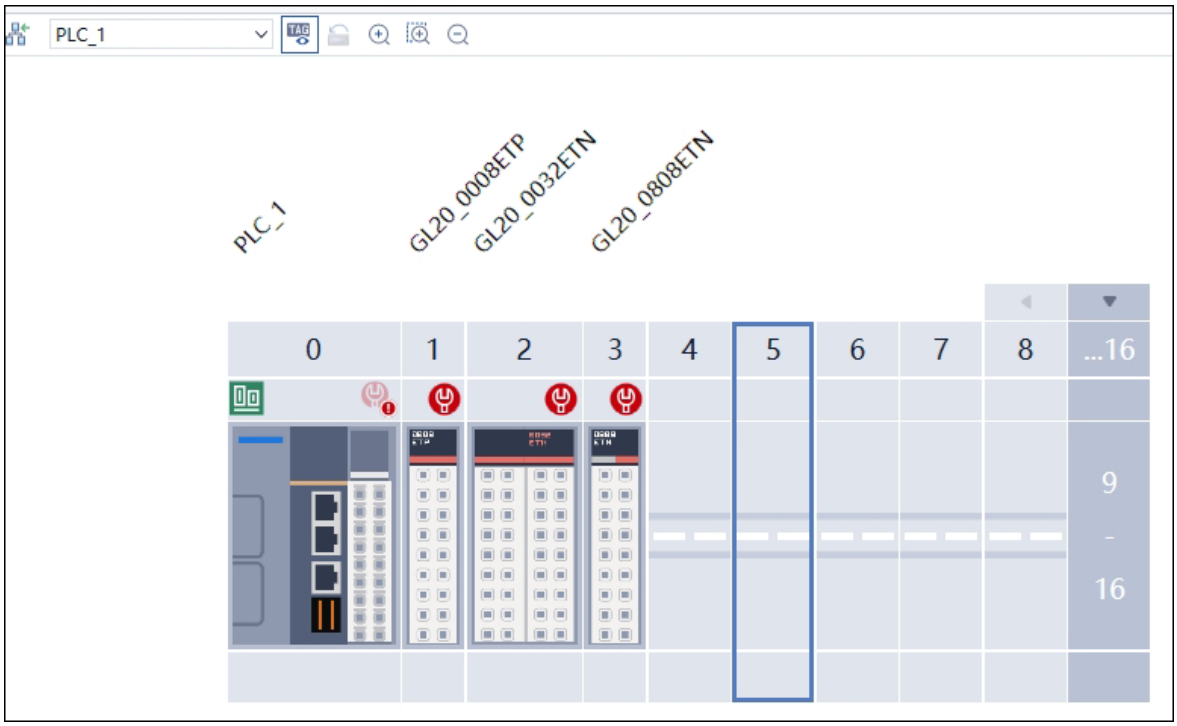
Procedure

1. Double-click "Network configuration" or "Device configuration" in the project tree to open the corresponding page.

2. View diagnostic information. For the meaning of each symbol, see 15.3.2.1 Diagnostic Symbol Description.
   - Network configuration: Displays diagnostic information for all connected devices. Double-click a device to navigate to its "Device configuration" view for its diagnostic information.



   - Device configuration: Displays diagnostic information only for the current device and its subordinate devices. Users can directly view the device's current status information.



NOTE

Through the network configuration or device configuration, users can view only the current status information of the device. To view detailed diagnostic information for each device, double-click the diagnostic symbol on the device directly. The system will automatically navigate to the "Online and diagnostics" page for more information.

15.3.2.3 Online and Diagnostics

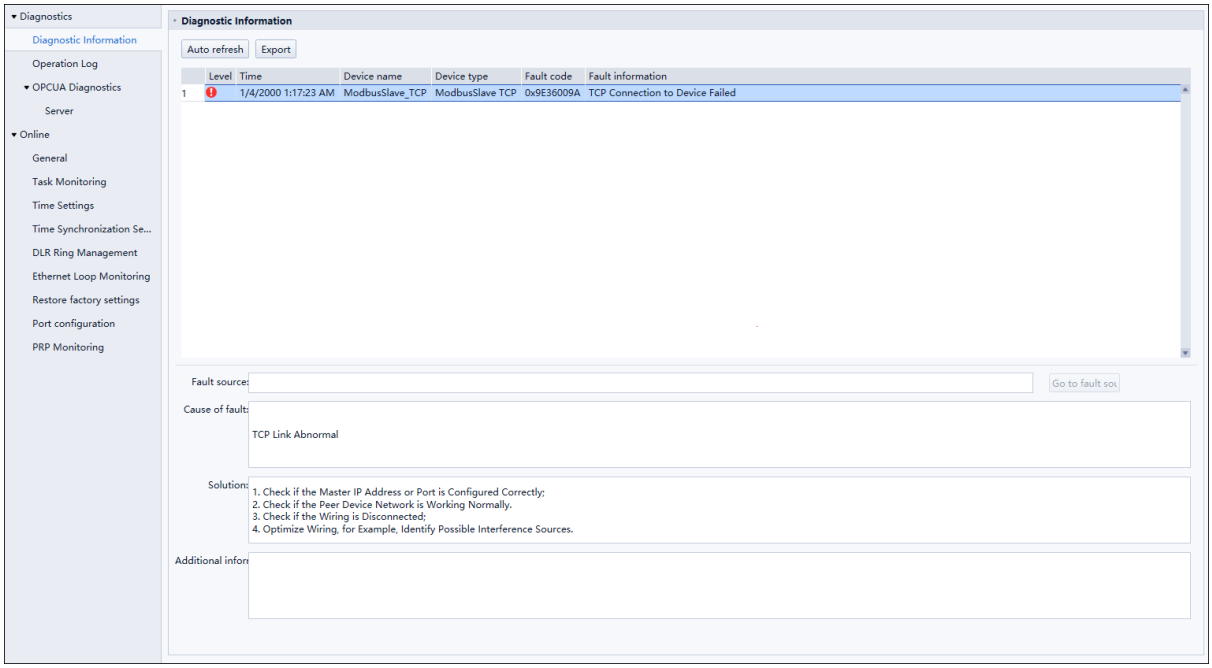
The "Online and diagnostics" interface provides a centralized view of device operating status, diagnostics information, and logs. It integrates status monitoring, fault localization, and data traceability to help users quickly assess device status, detect and address potential issues, and improve system reliability and maintenance efficiency.

Prerequisites

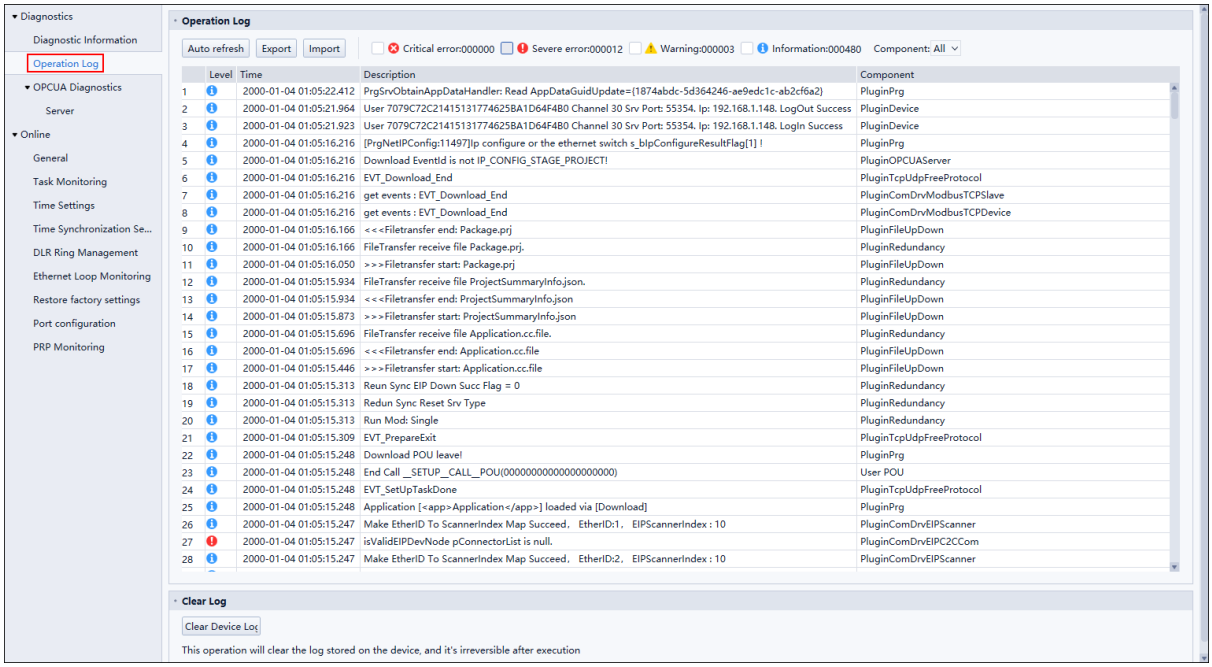
The device has been switched to online.

Procedure

1. Double-click "Online and diagnostics" in the project tree.
2. On "Diagnostic info" tab, view device diagnostic information, including fault level, time, device name, device type, fault code, and fault information. Select a fault entry to display the fault source, cause, and solution.



3. On "Operation log" tab, view relevant log information for the device during operation. Click "Freeze display" to filter by fault level, component, etc.



NOTE

For details about diagnostic information and operation logs, see 10.11.20 Diagnostic Information and 10.11.21 Operation Log.

15.3.3 Viewing Diagnostic Information via Web Server

After enabling the Web server, users can access module data, user program data, and diagnostic data from the device via an Internet browser on any terminal device (such as a PC or smartphone) without installing iFA.

Restrictions

The Web server supports the following browsers and versions:

Category	Browser and Version
PC	• Google Chrome 63.0 or later • Mozilla Firefox 67.0 or later • Microsoft Edge 18.0 or later • MacOS Safari 12.0 or later
Mobile	• iOS Safari 13.0 or later • Android Chrome 80.0 or later • Android Firefox 68.0 or later

Prerequisites

- The device has been switched to online.
- The Web server configuration is complete. For details, see 10.6.14.7 Web Server Configuration.

Logging in to the Web server

1. Enter "http://device_IP_address" in the browser, enter the username and password (configured during Web server setup), and click "Login".
2. On the "Web diagnostics and monitoring" page, users can view module data, user program data, and diagnostic data from the device.

Diagnostic items

- Main interface: Displays device system status, device information, and basic device details.

Item		Description
Product information		Specific model information of the product, and the current RTC time/time zone of the controller.
System State		Device operating status.
CPU operation		The device flashes based on online PLC. Click the "Device flashing" button once and the device will flash for 5s.
Device information (in EVO800 series redundant mode, the main/backup device status will be displayed; in single mode of EVO700 series and other devices, device information will not be displayed)		Displays the optical fiber status information of the main/backup devices.
Device image		Displays the device image.
Basic information	Project name	Name of the project created by the user.
	Controller version	Controller firmware version.
	Product model	Specific product model, such as EVO800.
	SN code	PLC hardware serial number.
	Main control module (Displayed in EVO800 redundant mode)	For redundant control modules, the model of the main control module is displayed, such as EVO820.
	Redundant synchronization module (Displayed in EVO800 redundant mode)	For redundant control modules, the model of redundant module is displayed, such as EVO800-RM.
	Last compilation time	The last project compilation time in the controller (including incremental compilation).
	Last download time	The last project download time in the controller (including online modifications).

- Online and diagnostics: Displays diagnostic information for the current device, including fault level, time, device name, fault code, and fault message.

NOTE

When the online diagnostic information section shows blank content, manually clear the browser cache and refresh the web server to retrieve the latest online diagnostic information.

Function	Description
Search	Enter a fault code in the search box to filter matching information.
Advanced filter	Select a fault level (Fatal, Critical, Alarm, Info, Unknown) to filter faults by level.
Sort	Sorting can be performed by level, time, or in ascending/descending order.
Freeze display	Click "Freeze display" to stop real-time refreshing of diagnostic information.
Export controller log	Click "Export controller log" to save device diagnostic information locally.
View fault details	Click the fault level to display detailed fault information, including the cause of the fault code and recommended solutions.

- Network communication: Displays network port communication information for the current device. All network information is read-only, not editable, and refreshes every second.

Parameter	Description
Port	The port refers to the actual physical port of the device.
Port IP	The IP address of the port.
Subnet mask	The subnet mask of the port.
Connected device information	The IP address of the connected device. If connected via a personal PC, the PC's IP address is displayed.
Communication type	The protocol type used by the device. Currently supports communication protocols such as EIP/OPCUA/WEB/Modbus TCP.
Connection status	Displays whether the device's connection status is normal or faulty. • For active connection, the slave status matches the project tree status, and the PLC acts as the master (such as EIP master and Modbus TCP/RTU master). • For passive connection, the connection status displays "Connected", and the PLC acts as a slave (such as EIP slave, Modbus TCP/RTU slave, WEB, OPCUA server, or HMI-CLASS3 connection).

NOTE

For details about the Web server, see 10.11.12 Web Diagnostics and Monitoring.

15.3.4 Viewing Diagnostic Information via HMI Device

The HMI provides a centralized diagnostics display function, allowing real-time monitoring of PLC operating status and viewing of diagnostic data from I/O modules and coupler modules. All HMI diagnostic data is synchronized in real time from the content under the PLC device's "Online and diagnostics" node.

Restrictions

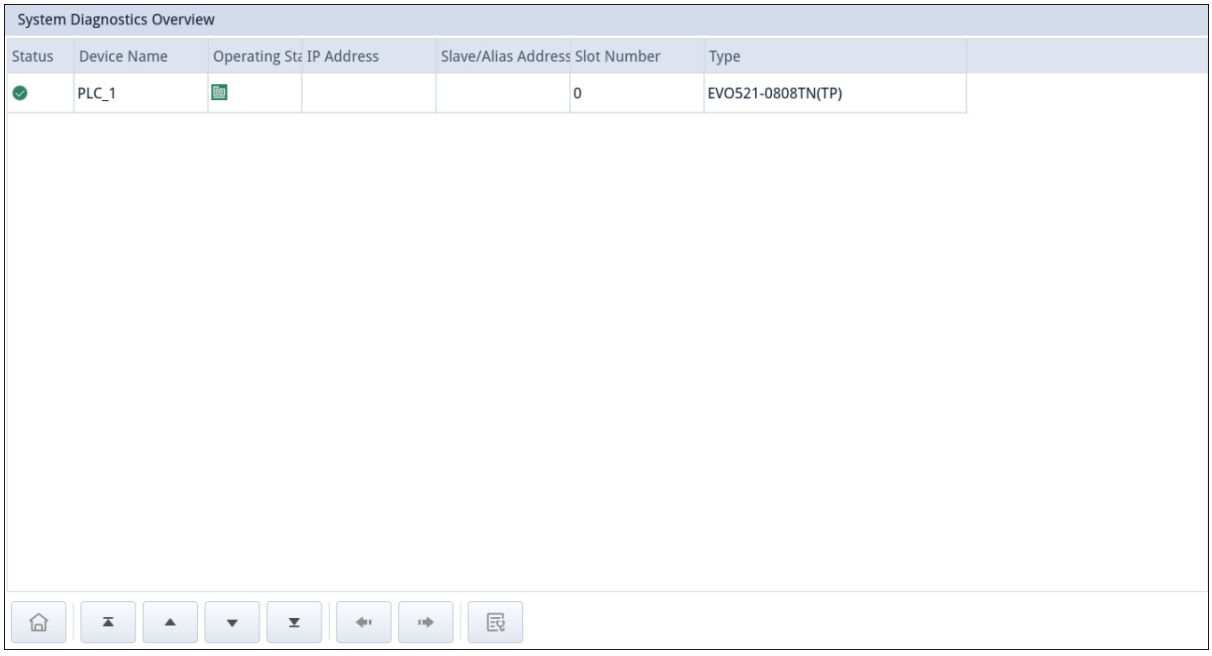
The HMI diagnostic display is implemented through the system diagnostic control. This control must run in a physical hardware environment and is not supported in a simulation environment.

Prerequisites

- The PLC and HMI are in the same project in iFA software.
- "Inovance Technology, Ethernet/IP Tags protocol" is used for communication.
- The HMI's system diagnostic control must run in a physical hardware environment and is not supported in a simulation environment.
- A screen has been added in the HMI.

Procedure

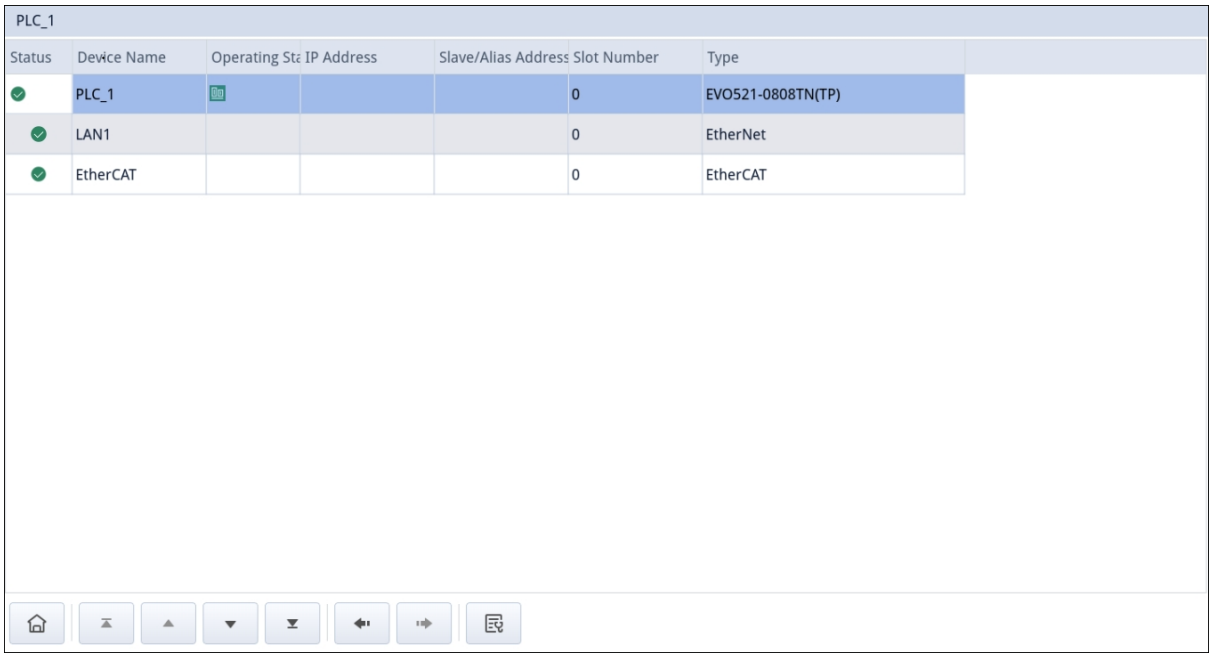
1. In the iFA project tree on the left, double-click the screen.
2. Select the "System diagnostics" object from the "Toolbox" on the right and drag it to the screen area.
3. On the "Property" tab on the right, configure the properties. For details, see System diagnostics control properties.
4. Download the project. For details, see 13.14 Project Download.
5. When the screen containing the "System diagnostics" control is active, the control displays the system diagnostic overview interface by default.



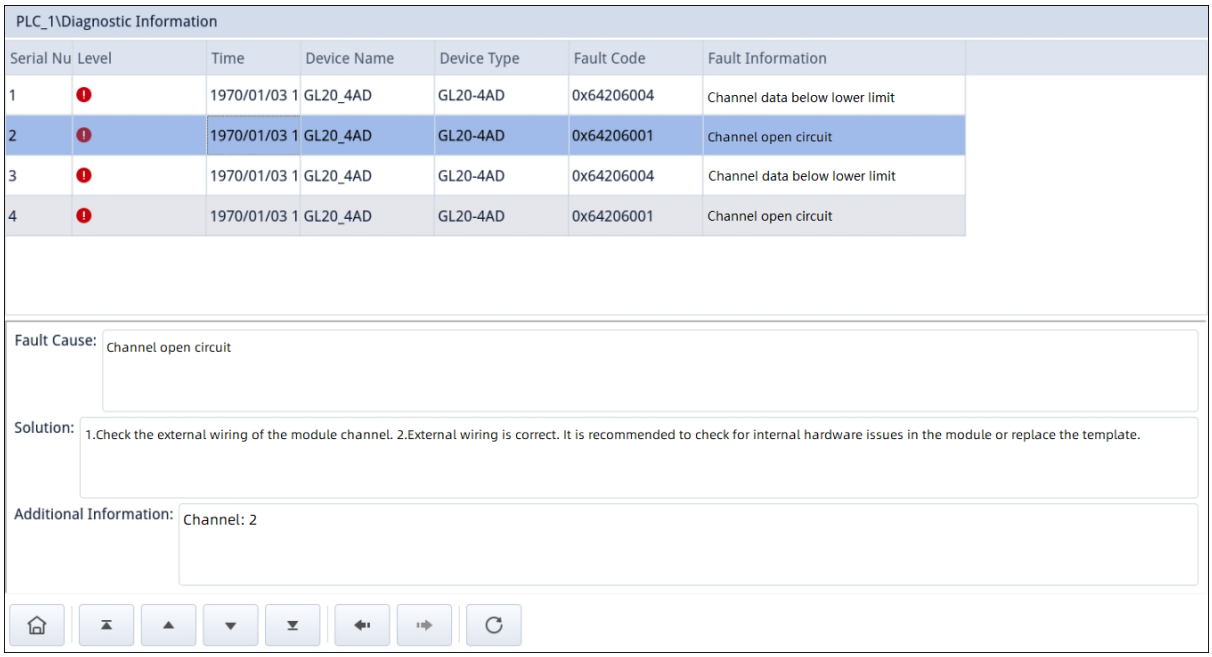
Item	Description
Control header	Displays the hierarchy of the device based on the current level and selected object. The hierarchy text is identical to the path name in the software.

Item	Description
Status	Displays the current device running status or the connection status between PLC and HMI. PLC statuses include: Normal, Fault, and Lower-level device fault.
Device name	Displays the name of the device as shown in the project tree.
Operation status	Displays the four PLC statuses: Operational: Stopped: Paused: Forced: This column is empty for other modules.
IP address	Displays the IP address or port connected between the PLC and the current HMI.
Slave/alias address	Displays the slave number or alias address of EtherCAT slave devices. Empty for non-EtherCAT slaves.
Slot number	Displays the slot number of the current module in the rack.
Type	Displays the device model.

6. Select a PLC device row and click . View the connection status of the PLC device's LAN port and EtherCAT port.



7. Select a PLC device row and click . View the I/O channel diagnostic information of the PLC device.



15.4 System Variable Diagnostics

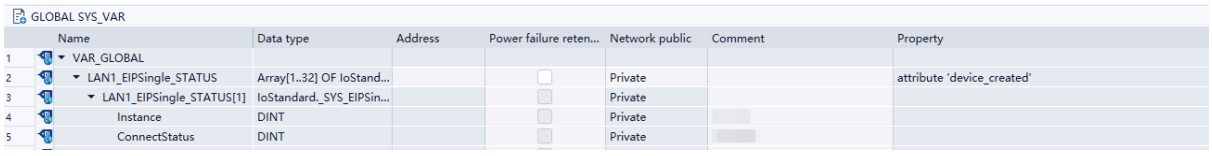
The platform provides a series of predefined system variables for real-time monitoring and diagnostics of system operating status. These system variables cover key aspects such as devices, fieldbuses, and remote communication. Users can quickly locate faults by directly reading these variables without writing complex diagnostic logic.

Prerequisites

- Devices and corresponding modules have been added, such as EtherCAT masters/slaves and EtherNet/IP adapters.
- The device has been switched to online.

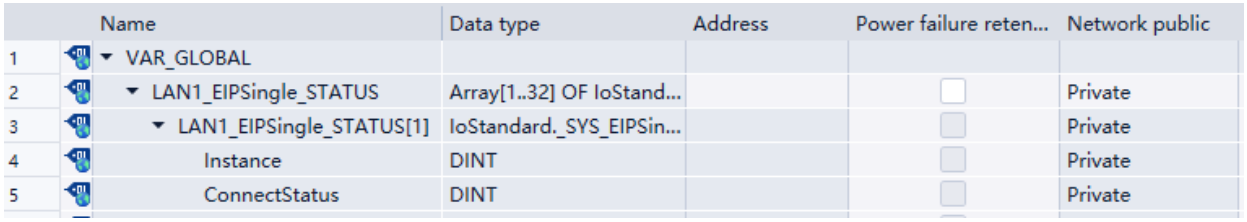
Procedure

1. Under the "Programming" node in the left project tree, double-click "SYS_VAR(Global)" to open the system variable table.
2. View diagnostic information in the list. For details about various faults, see System variable diagnostic categories.



System variable diagnostic categories

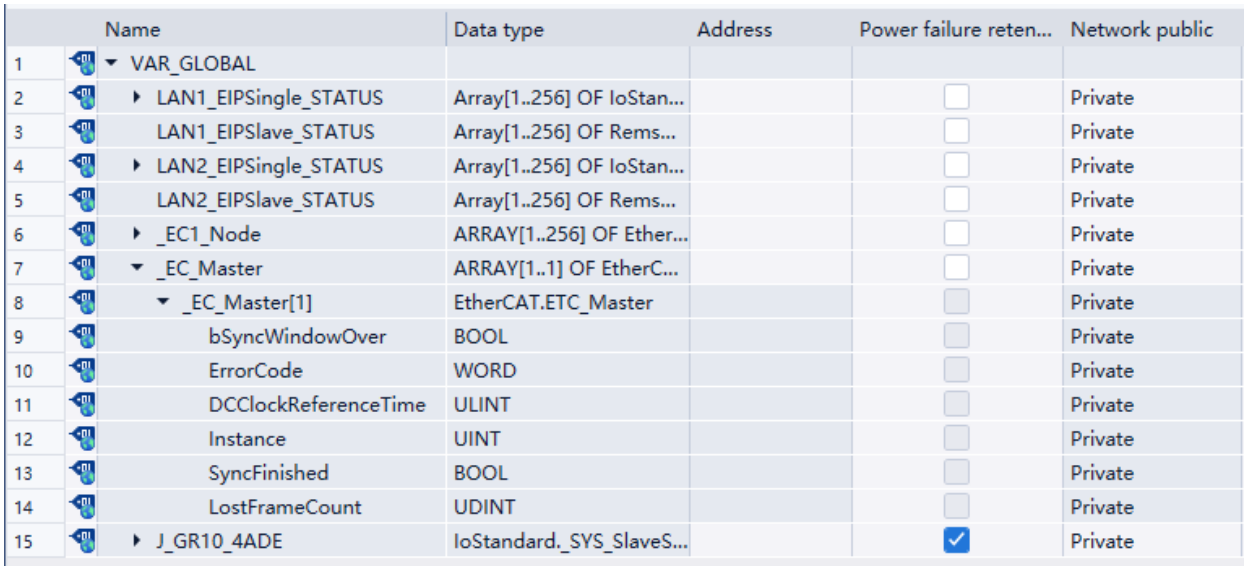
- Device diagnostic status variables Device diagnostic status variables are related to EtherNet/IP device configuration. When adding a PLC device, corresponding variables are automatically created in the system variable table. Variable name: Device network port number_EIPSingle_STATUS.
  These variables are used to display connection information of the master. The specific structure details are shown in the table below.



	Name	Data type	Address	Power failure reten...	Network public
1	VAR_GLOBAL				
2	LAN1_EIPSingle_STATUS	Array[1..32] OF IoStand...		☐	Private
3	LAN1_EIPSingle_STATUS[1]	IoStandard._SYS_EIPSin...		☐	Private
4	Instance	DINT		☐	Private
5	ConnectStatus	DINT		☐	Private

Variable	Definition	Function description
Instance	Instance number	-
ConnectStatus	Connection Status	0: Connection is closed. 1: Connection is abnormal. 3: Connection is normal.

- EtherCAT master system variables EtherCAT master system variables are related to EtherCAT device configuration and are automatically created in the system variable table when adding an EtherCAT device, with the variable name: _EC_Master.
  EtherCAT master system variables are used to display information about the EtherCAT master. The specific structure is described in the table below.

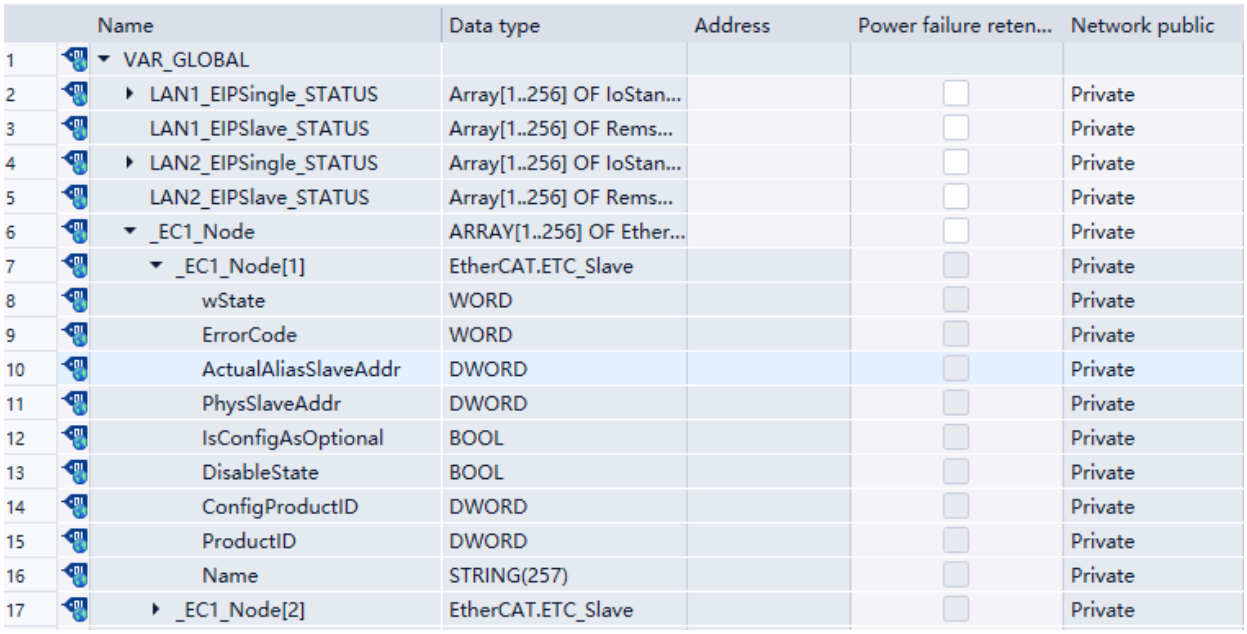


	Name	Data type	Address	Power failure reten...	Network public
1	VAR_GLOBAL				
2	LAN1_EIPSingle_STATUS	Array[1..256] OF IoStan...		☐	Private
3	LAN1_EIPSlave_STATUS	Array[1..256] OF Rems...		☐	Private
4	LAN2_EIPSingle_STATUS	Array[1..256] OF IoStan...		☐	Private
5	LAN2_EIPSlave_STATUS	Array[1..256] OF Rems...		☐	Private
6	_EC1_Node	ARRAY[1..256] OF Ether...		☐	Private
7	_EC_Master	ARRAY[1..1] OF EtherC...		☐	Private
8	_EC_Master[1]	EtherCAT.ETC_Master		☐	Private
9	bSyncWindowOver	BOOL		☐	Private
10	ErrorCode	WORD		☐	Private
11	DCClockReferenceTime	ULINT		☐	Private
12	Instance	UINT		☐	Private
13	SyncFinished	BOOL		☐	Private
14	LostFrameCount	UDINT		☐	Private
15	J_GR10_4ADE	IoStandard._SYS_SlaveS...		☑	Private

Variable	Definition	Function description
bSyncWindowOver	Synchronization window deviation	The default value is FALSE. bSyncWindowOver is set to TRUE when the "Sync window" value (μs) is greater than 0 and is no less than the current bus's "Max. sync window deviation".
ErrorCode	Master fault code	0x0001: Configured slave count < scanned slave count 0x0002: Slave alias conflict 0x0004: Slave fault detected 0x0008: Continuous frame loss 0x0010: Slave CRC fault 0x0020: Main network card not connected 0x0040: Redundant network card not connected 0x0080: Slave alias is 0 0x0100: Network card not connected
DCClockReferenceTime	DC reference clock	DC reference clock real-time value (read-only).

Variable	Definition	Function description
Instance	Master instance address	-
SyncFinished	Master synchronization status	SyncFinished is set to TRUE after master startup completes, regardless of whether slaves are in OP (Operational) status.
LostFrameCount	Lost frame count	Consistent with the "Lost frames" displayed on the "Online" configuration page of the EtherCAT master.

- EtherCAT slave system variables EtherCAT slave system variables are related to EtherCAT device configuration and are automatically created in the system variable table when adding an EtherCAT device, with the variable name: _EC1_Node.
  EtherCAT slave system variables are used to display information about the EtherCAT slaves. The specific structure is described in the table below.

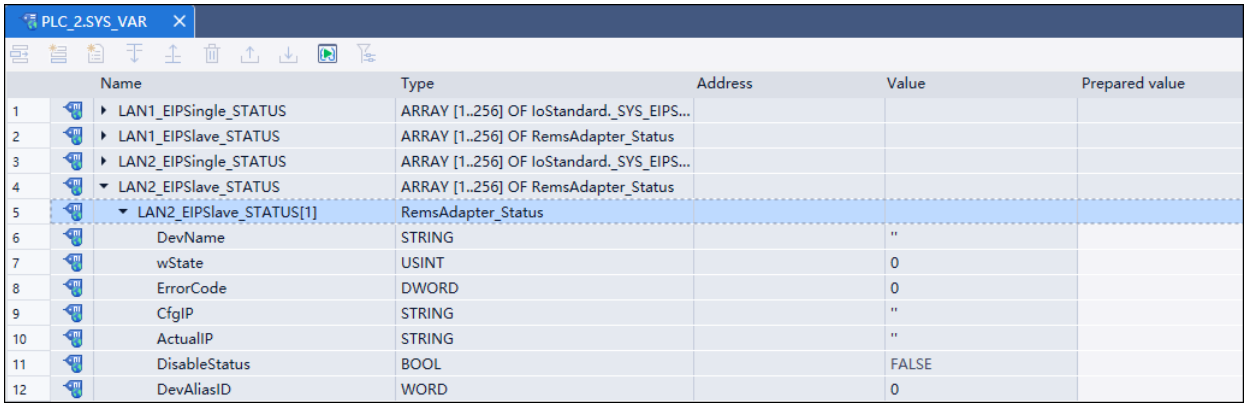


	Name	Data type	Address	Power failure reten...	Network public
1	VAR_GLOBAL				
2	LAN1_EIPSingle_STATUS	Array[1..256] OF IoStan...			Private
3	LAN1_EIPSlave_STATUS	Array[1..256] OF Rems...			Private
4	LAN2_EIPSingle_STATUS	Array[1..256] OF IoStan...			Private
5	LAN2_EIPSlave_STATUS	Array[1..256] OF Rems...			Private
6	_EC1_Node	ARRAY[1..256] OF Ether...			Private
7	_EC1_Node[1]	EtherCAT.ETC_Slave			Private
8	wState	WORD			Private
9	ErrorCode	WORD			Private
10	ActualAliasSlaveAddr	DWORD			Private
11	PhysSlaveAddr	DWORD			Private
12	IsConfigAsOptional	BOOL			Private
13	DisableState	BOOL			Private
14	ConfigProductID	DWORD			Private
15	ProductID	DWORD			Private
16	Name	STRING(257)			Private
17	_EC1_Node[2]	EtherCAT.ETC_Slave			Private

Variable	Definition	Function description
wState	Slave connection status	1: Initialized 2: Pre-operational 3: Boot-operational 4: Safe-operational 8: Run 17: Initialization fault 18: Pre-operational fault 19: Boot-operational fault 20: Safe-operational fault 24: Operational fault

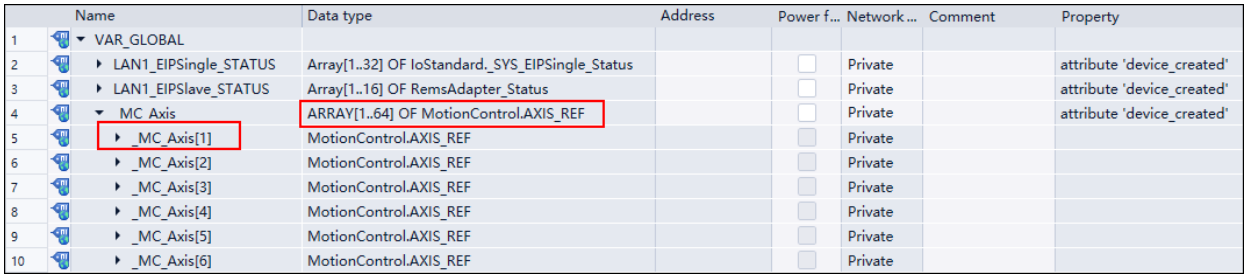
Variable	Definition	Function description
ErrorCode	Slave fault code	0x0001: Slave device offline 0x0002: Slave device vendor ID mismatch 0x0004: Slave device product code mismatch 0x0008: Slave device version number mismatch 0x0010: Slave device fault flag 0x0020: Slave device reverse connection 0x0040: Slave device port closed 0x0080: Slave device status fault 0x0100: Slave device SDO configuration failed 0x0200: Slave device status transition timeout 0x0400: Slave device SDO download timeout 0x0800: Slave device EEPROM fault Note: When multiple faults exist simultaneously, the slave fault codes will be displayed in an accumulated manner.
ActualAliasSlaveAddr	Slave alias address	-
PhysSlaveAddr	Slave physical address	Slave address 1001
IsConfigAsOptional	Whether the slave is configured with an alias	TRUE: The slave is configured with an alias. FALSE: The slave is not configured with an alias.
DisableState	Whether the slave is disabled	TRUE: The slave is disabled. FALSE: The slave is enabled.
ConfigProductID	Configured product ID	-
ProductID	Actual slave product ID	-
Name	Slave name	-

- Remote adapter status variables Remote adapter status variables are related to EtherNet/IP device configuration. When adding an EtherNet/IP device, corresponding variables are automatically created in the system variable table. Variable name: Device network port number_EIPSlave_STATUS. These variables are used to display the operation status of the remote adapter. The specific structure details are shown in the table below.



Variable	Definition	Function description
DevName	Remote adapter name	-
wState	Remote adapter connection status	0: Connection is closed. 1: Remote adapter is disabled. 2: Connection is abnormal. 3: Connection is normal.
ErrorCode	Remote adapter fault code	See 21.1 Appendix 1: List of Fault Codes.
CfgIP	Remote adapter configured IP	-
ActualIP	Remote adapter actual IP	-
DisableStatus	Remote adapter disabled status	TRUE: The slave is disabled. FALSE: The slave is enabled.
DevAliasID	Remote adapter alias ID	-

- Axis array variables Axis array variables are related to axis device configuration. When an axis device is added, the corresponding variables are automatically created in the system variable table. Variable name: _MC_Axis. These variables are used to manage all real axes and virtual axes under all EtherCAT masters in a centralized way, and store system variable information for these axes. For detailed structure description, refer to 18.3.3.2.2.1 AXIS_REF: Axis Structure.



Axis array variables	Description
Array name	_MC_Axis
Array length	Total number of EtherCAT axes supported by this PLC
Array mapping	Axis array subscript represents the axis number. For example, the array element corresponding to "Axis" is _MC_Axis[1].
Variables accessible in the array	Refer to 18.3.3.2.2.1 AXIS_REF: Axis Structure.

Axis array variables	Description
Axis array subscript corresponding to the axis	The subscript of the axis in the axis array is displayed under the project tree "PLC_X node > Control object > Axis". The subscript can be manually adjusted in the 16.1.3 Axis basic settings interface. Note: Axis numbers must not be duplicated and must not exceed the maximum number of axes supported by the PLC.



For more information on diagnostic system variables, see 10.1.1.9.6 System Variable Table.

15.5 Instruction Diagnostics

Instruction diagnostics enable real-time retrieval of device diagnostic information within user programs. By calling diagnostic instructions, users can query the current device operating status, fault information, and alarm details, enabling real-time monitoring of system status and rapid fault localization. The following lists all diagnostic instructions supported by the platform.

- System information

Instruction	Functions
SysHC_HWInfo	Obtain hardware Information
GetDeviceDiag	Obtain device diagnostic information
GetDeviceStatus	Obtain device state
SysHC_CPUInfo	Obtain CPU information such as memory, frequency, and temperature
SysHC_FaultRecord	Get fault data recording by calling instruction

- Redundancy information

Instruction	Functions
GetControllerModuleDiagnosisInformation	Obtain basic diagnostic information of the controller module
GetRedundancyModuleDiagnosisInformation	This instruction obtains the basic diagnostic information of the redundancy module.

- EtherNetIP

Instruction	Functions
GetEIPSystemstats	Retrieve EIP system diagnostic information
GetEIPConnectstats	Get diagnostic information of EIP connection
SysHC_EthernetIPDeviceDiagnose	Obtain the fault diagnostics of the EIP slave
SysHC_EthernetIPSlaveDiagnose	Obtain the fault diagnostics of the EIP master

- Local bus

Instruction	Functions
SysHC_CPU_IOMODULE_Diagnose	Get the diagnostic state of local IO module
LBus_GetDiagMailBoxData	Get diagnostic mailbox information of the specified module on the local bus
LBus_GetModuleInfo	Get all information about the local bus and local I/O
SysHC_GetAllGL20ModuleInfo	This instruction gets basic information about all modules on the local bus.
SysHC_GetGL20ModuleInfo	Get basic information about a specified module on the local bus

- ModbusRTU

Instruction	Functions
SysHC_ModbusRtuSlaveDiagnose	Get the diagnostic status from the Modbus-RTU master

- ModbusTCP

Instruction	Functions
SysHC_ModbusTcpSlaveDiagnose	Obatin Modbus-TCP master diagnostic status information
SysHC_ModbusTcpDeviceDiagnose	Get Modbus-TCP as slave diagnostic status information

- EtherCAT

Instruction	Functions
ETC_GetCouplerInfo	Read EtherCAT coupler information
ETC_GetMasterinfo	Get EtherCAT master information
ETC_GetSlaveInfo	Get EtherCAT slave information
ETC_MasterDiagnose	Get EtherCAT master diagnosis information

- CANBus

Instruction	Functions
CANBus_GetDevDiagInfo	Get device diagnostic information

- CIA405

Instruction	Functions
CIA405_GET_STATE	Obtain the status of the CANopen device

- Free TCP

Instruction	Functions
SktTCPGetStatus	This instruction gets TCP connection status.

- FINS communication

Instruction	Functions
FINS_ServerDiag	Get the diagnostic information of FINS server

- Melsec communication

Instruction	Functions
MELSEC_ServerDiag	Get the diagnostic information of Melsec server

NOTE

For detailed parameters, programming examples, and complete usage instructions for each instruction, see 18.4 Communications and Diagnostics in the "Common Instructions" chapter.

15.6 Axis Control Diagnostics

Axis control diagnostics are a critical component for ensuring stable operation of automation equipment and improving system reliability. They enable fast and accurate fault localization and allow users to trace the root cause of issues based on diagnostic information, so that corrective measures can be taken.

The platform provides two diagnostic approaches for viewing and analyzing axis-related diagnostics, helping users efficiently troubleshoot and resolve axis operation anomalies.

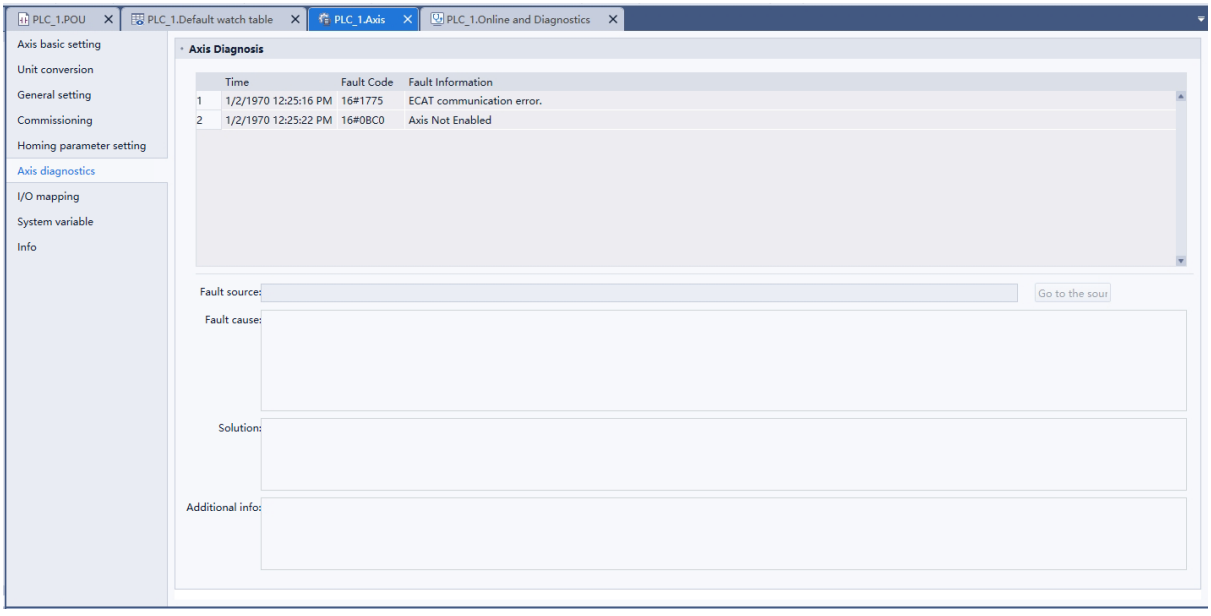
Diagnosing via the axis configuration interface

Prerequisites

- An axis has been added.
- The device has been switched to online.

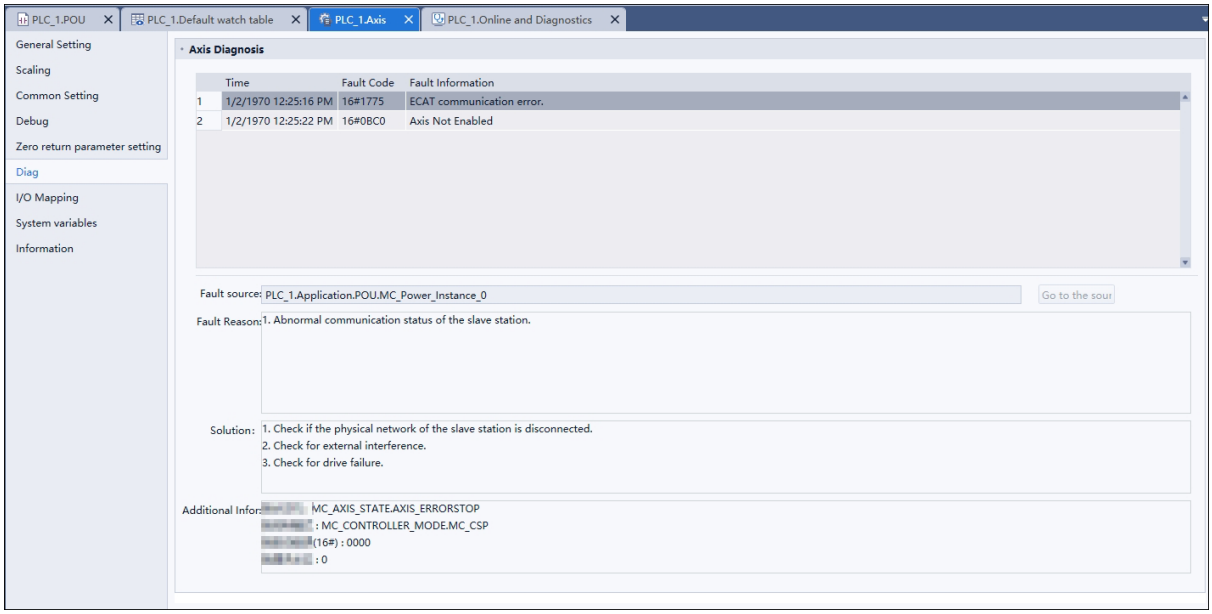
Procedure

1. In the project tree on the left, select "Control object > Axis". Double-click the added axis to open the axis configuration interface.
2. On the "Axis diagnostics" tab, view the occurrence time, fault code, and fault information of all faults in the list.



Parameter	Description
Time	The time when the fault occurred.
Fault code	Displayed in hexadecimal by default. For details, see 18.3.3.8 Fault Codes for all fault codes, corresponding information, causes, and solutions.
Fault information	Brief information about the fault.

3. In the axis diagnostics list, select a fault to view its source, cause, solution, and additional information.



Parameter	Description
Fault source	Displays the root cause of the fault.
Cause	The cause corresponding to the fault information.
Solution	The corrective action corresponding to the fault information.
Additional info	Additional details, including axis state machine, axis control mode, axis drive fault, and axis communication status.

Diagnosing via axis structure variables

The axis structure parameter fbeFBError records fault information for motion control instructions, including fault code, source, occurrence time, and communication status upon fault. When viewing diagnostic information through axis structure variables, the software provides the following two entries.

- System variable table
  The system variable table provides variables related to axis device configuration. When an axis device is added, the corresponding variable (variable name: _MC_Axis) is automatically created in the table. You can directly locate the axis structure variable Axis.fbeFBError and view diagnostic information here.
- Watch table
  The watch table supports real-time monitoring of specified variables. You can add the axis structure variable Axis.fbeFBError to the watch table to directly view diagnostic information.

NOTE

For details about the fbeFBError parameter, see FB fault message display and 18.3.3.2.2.4 MC_FBERROR_REF: Fault Record Structure.

16 Motion Control

16.1 Axis Configuration

16.1.1 Overview

In motion control systems, the object of motion control is referred to as an axis. The axis serves as a bridge connecting the drive and the PLC. When used as an output device, the axis can control EtherCAT bus drives compliant with the 402 protocol or local high-speed pulse outputs. When used as an input device, the axis can implement functions such as counting and frequency measurement for EtherCAT bus modules compliant with the 402 protocol or local high-speed pulse inputs.

16.1.2 Adding an Axis

Prerequisites

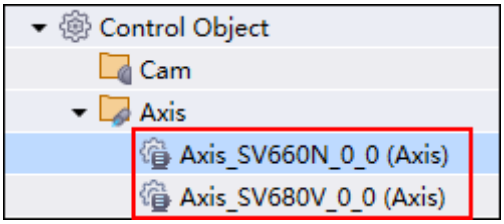
The selected controller supports the motion control function.

Procedure

1. Select "Control Object > Axis" in the left project tree.
2. Right-click the "Axis" node and select "Add axis".
3. In the pop-up dialogue box, enter the axis name and click "OK".

Result

The newly created axis is displayed under the "Axis" directory.

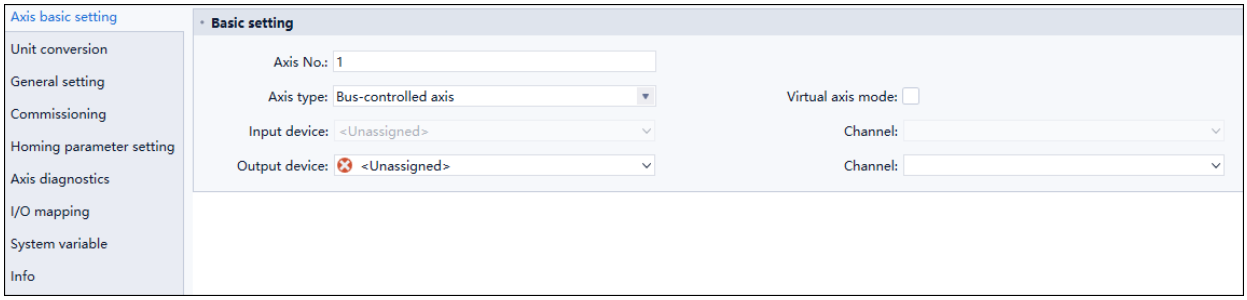


You can also assign axes in batches to added servo devices. For details, refer to 16.1.4 Batch Servo Assignment.

16.1.3 Basic Axis Settings

Double-click an added axis in the left project tree to open the axis configuration interface and set basic axis parameters.

Basic axis configuration includes "Axis number", "Axis type", "Input device", "Output device", and "Probe setting".



- Axis number: The axis identifier. When adding an axis device, a unique axis number is automatically assigned, which can be viewed and modified.
- Axis type: Options include bus-controlled axis, bus encoder axis, local encoder axis, and local pulse axis.

ID	Axis Type	Note	Category
①	Bus-controlled axis	Controls EtherCAT bus drives compliant with the CiA402 protocol. Supports basic control modes including torque, position, speed, and homing.	Real axis
②	Bus encoder axis	Controls EtherCAT bus expansion modules compliant with the CiA402 protocol. Combined with the module's input/output interfaces, it enables functions such as position preset, probe, gate control, and comparison output.	
③	Local Encoder Axis	Integrated with local high-speed I/O counters, it enables 4-axis 32-bit high-speed counting, frequency measurement, latch, and preset functions.	

ID	Axis Type	Note	Category
④	Local pulse axis	Controls pulse drives via local high-speed I/O. It supports up to four local pulse axes if configured as dual-channel mode. Supports basic control modes including position, speed, and homing. Torque mode is not supported.	

- Input device: EtherCAT-compliant expansion modules or high-speed counters.
- Output device: EtherCAT-compliant servo drives, expansion modules, or high-speed I/O terminals.
- Virtual axis mode: Effective only for bus-controlled axes and local pulse axes. After enabling virtual axis mode, no output device needs to be selected to use motion control instructions.
- Virtual axis probe setting: A controller supports two channels of virtual axis probe signals, triggered by local high-speed I/O terminals. It can be used with MC_TouchProbe to capture probe positions.

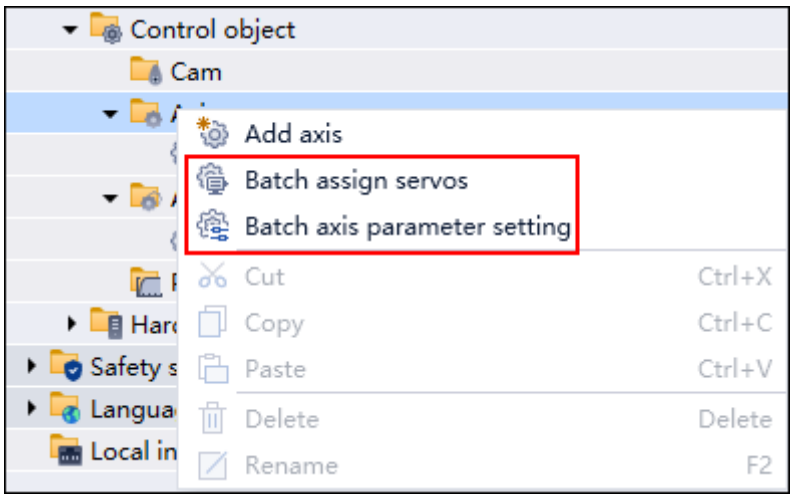
For multiple axes, axis parameters can be set in batches. For details, see 16.1.4 Batch Axis Parameter Settings.

16.1.4 Batch Servo Assignment and Axis Parameter Settings

When the device has multiple axes (such as servo motors and stepper motors), configuration parameters for different axes can be compared, copied, and pasted, reducing repetitive operations and improving configuration efficiency.

Procedure:

1. Right-click "Axis" under "Control object" in the left project tree.
2. Select "Batch assign servos" or "Batch axis parameter setting".



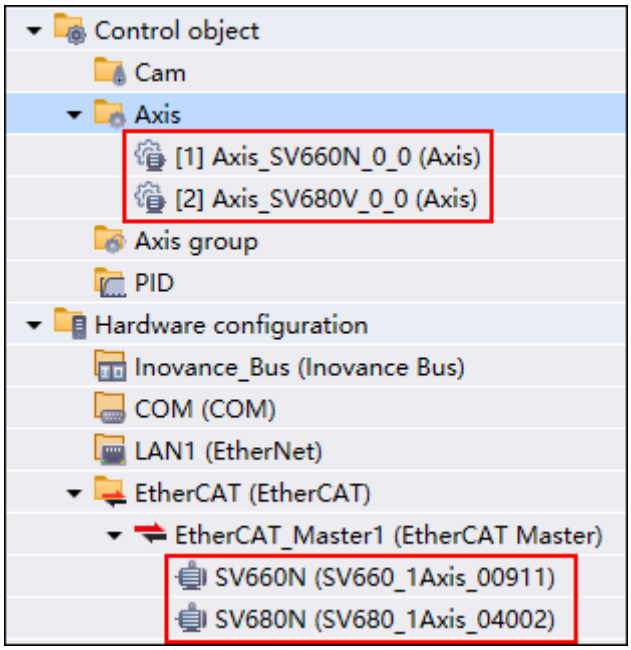
Batch servo assignment

The batch servo assignment supports the following operations:

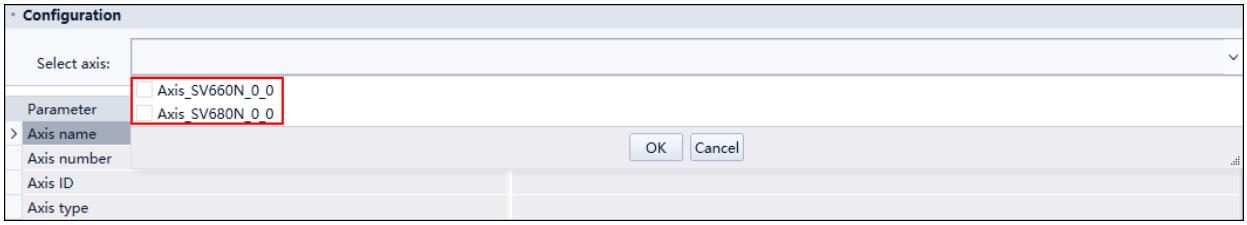
- Batch axis creation: Generates corresponding control axes for all servo drives without assigned axes.

- Auto mapping: Automatically completes the association settings between drives and axes.
- Multi-axis devices: If the drive supports multiple axes (such as a dual-axis servo), the system automatically creates the corresponding number of axes based on actual axis count.
- Third-party devices: Third-party devices are processed as single-axis drives by default, with only one axis created.

After adding servo devices, right-click "Axis" and select "Batch assign servos" to automatically create control axes corresponding to the servo devices, as shown in the figure below.

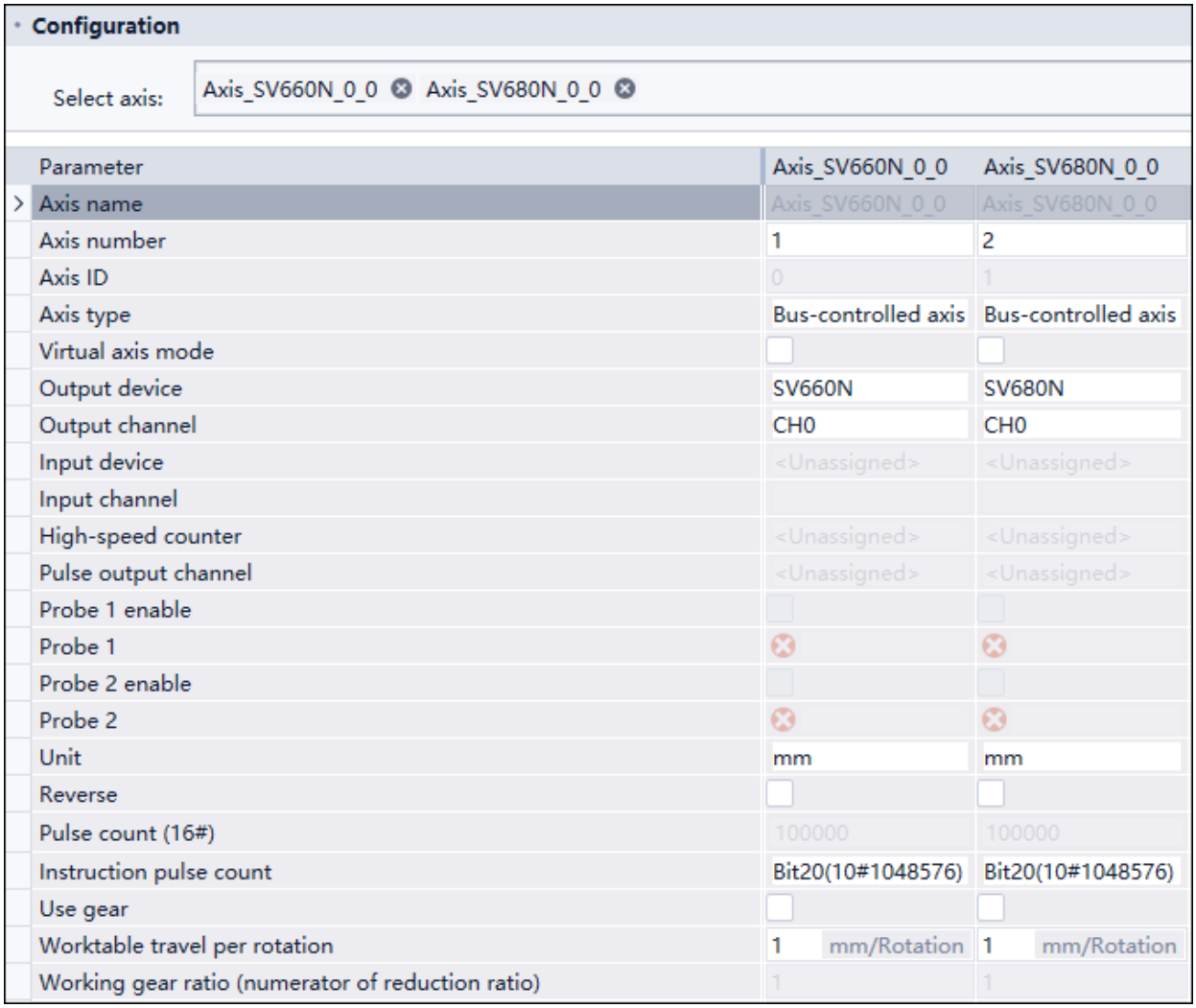


Batch axis parameter settings



After selecting the batch axis parameter settings function, you can select added axes for the following operations:

- Support viewing and modifying parameters of multiple axes simultaneously.
- Applicable to single-axis and IO modules of the same model.
- The parameters of up to eight axes can be compared simultaneously.



Multi-axis parameter comparison

- The parameters of multiple axes are displayed simultaneously for intuitive comparison.
- Different axes can be selected for parameter comparison (e.g., Axis 1 vs Axis 3 vs Axis 5).

Parameter modification method

- All parameters can be entered using drop-down options.
- Copying (Ctrl + C) and pasting (Ctrl + V) are supported for quick batch parameter modification.
- Parameter values can be dragged left or right to overwrite parameters on other axes.

Parameters available for batch settings

The following parameters support batch settings:

- Basic axis settings (e.g., axis type, control mode)
- Unit conversion (e.g., pulses/mm, speed unit)
- General settings (e.g., limit, alarm parameters)
- Homing parameters (e.g., homing method, speed)
- Bus cycle task (e.g., communication synchronization settings)

Use case 1: To adjust the homing speed of five servo motors, users can directly modify them in the table in batch, without setting each axis individually.

Use case 2: To copy the unit conversion parameters from axis 2 to axis 4 and axis 6, users only need to copy the parameter and paste it.

Use case 3: Users can compare the configuration differences between axis 1 and axis 3 to quickly identify inconsistent parameters, and adjust them.

16.1.5 Bus-Controlled Axis

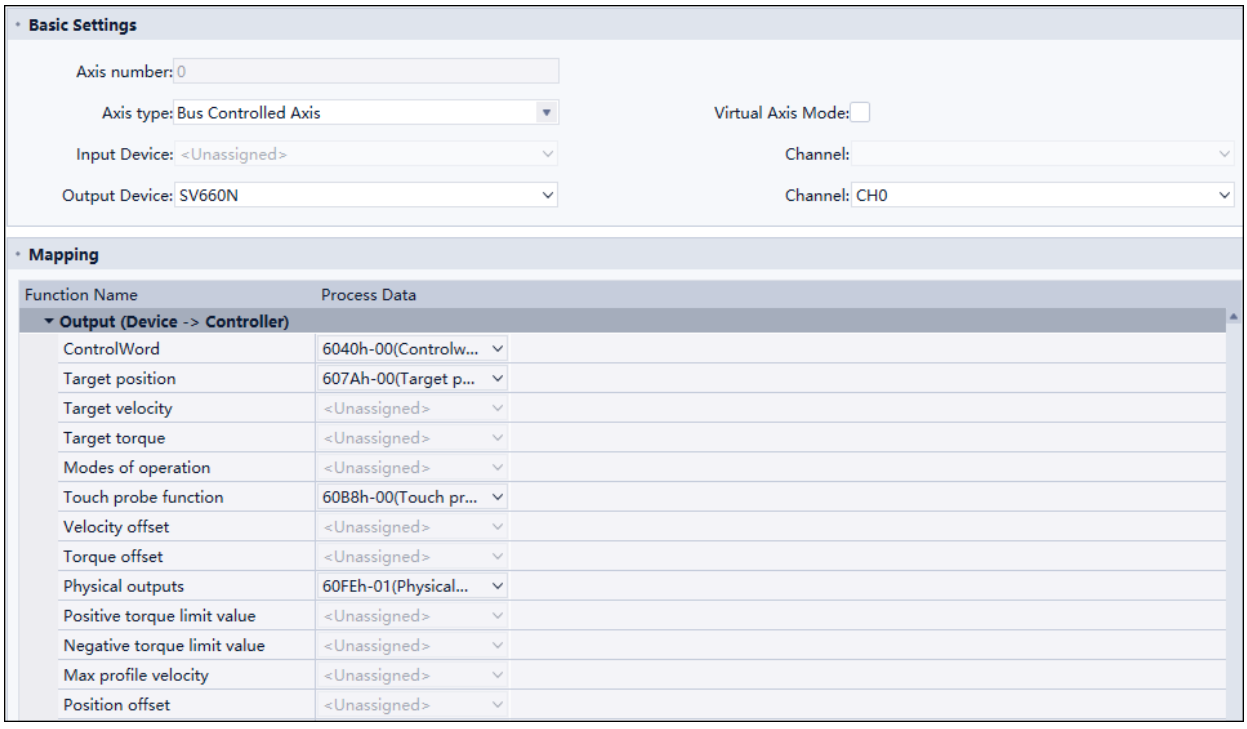
16.1.5.1 Basic Axis Settings

After selecting "Bus-controlled axis" as the axis type, the "Output device" option is activated.

- If device configuration has been completed in the device and network, you can select the corresponding device as the output.
- If the device and network configuration have not been performed, you can scan for devices under the EtherCAT_Master node.

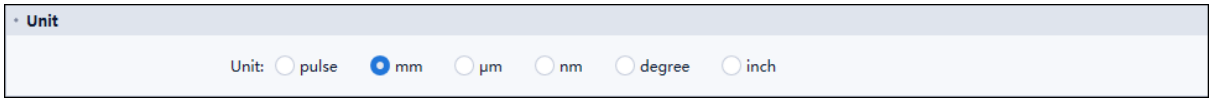


After completion, the PDOs selected on the servo side will be automatically mapped to the axis side:

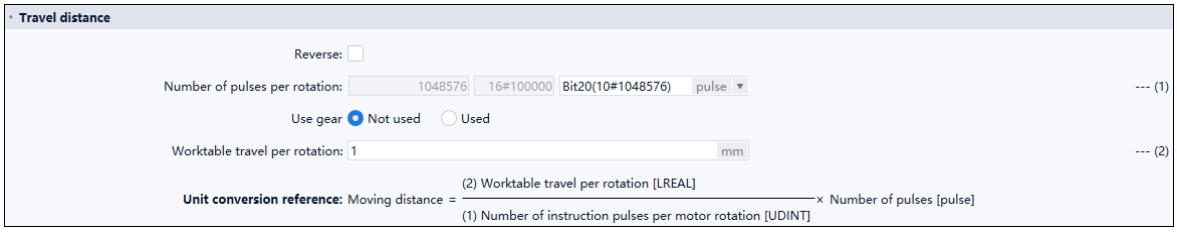


16.1.5.2 Unit Conversion

- Unit: Conversion between six units, including pulse, millimeter (default), micron, nanometer, degree, and inch. The unit can be configured as needed.



- Travel distance: Conversion between user unit and pulse unit.
  For bus-controlled axes or local pulse axes, the user unit (such as pulse, millimeter, or micron) is used on the instruction side, while the pulse unit is used for actual control of the physical motor. Gear ratio coefficient needs to be configured for unit conversion. The conversion modes are divided into the following scenarios:
  - Gears not used
    When gears are not used, the conversion formula from user unit to pulse unit is as follows:



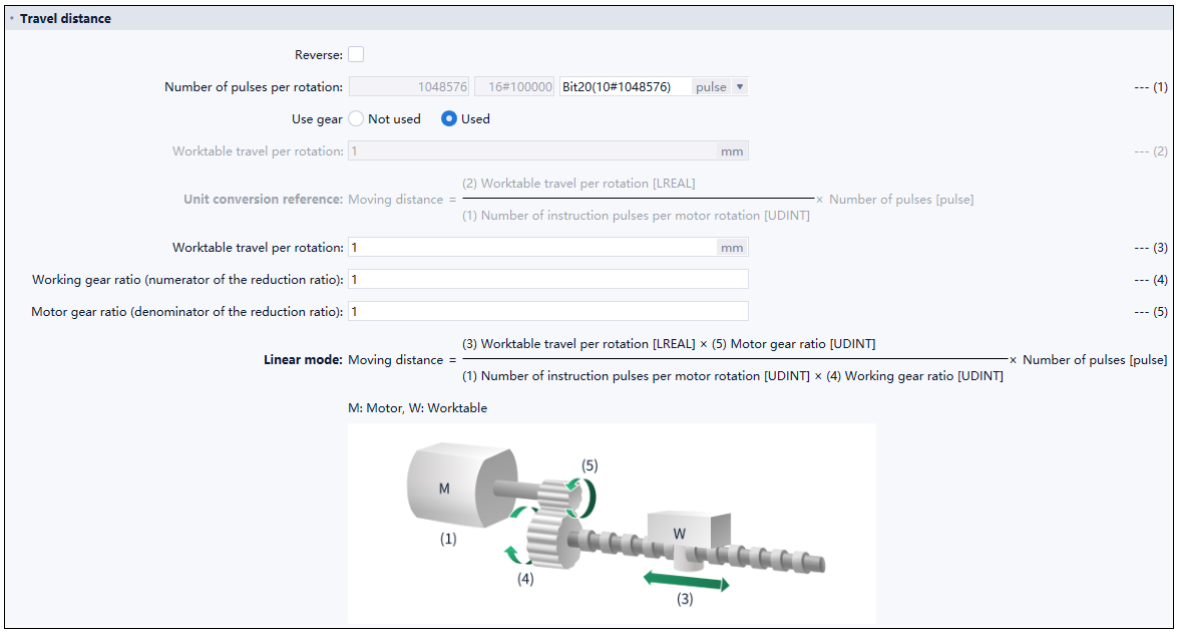
    Taking the Inovance 20-bit encoder as an example, set the parameters as follows:

    The number of pulses per motor revolution is default to 10#1048576. Number of bits for common encoders can be directly selected, which currently supports 14-bit, 18-bit, 20-bit, 23-bit, and 26-bit encoders. The number of pulses per motor revolution can also be customized, supporting both decimal and hexadecimal input formats. Hexadecimal input requires the "16#" prefix; otherwise, the system will automatically process the value as decimal.

    The working stroke per revolution of the worktable is 1.

    Based on the above parameters, if the target distance input on the relative motion instruction side is 100, the control axis will output an actual pulse count of 104857600, causing the motor to rotate 100 revolutions. Without a reducer, the worktable moves 100 user units.

  - Gears used
    The typical scenario for linear mode is shown in the following figure:

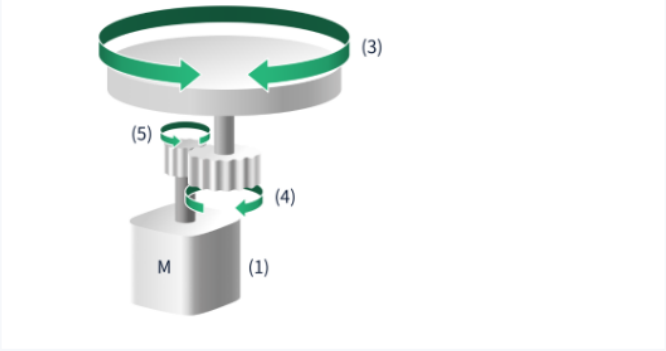


    (1) is the motor, (3) is the workpiece, (4) is the work gear ratio, and (5) is the motor gear ratio.

    The typical scenario for rotary mode is shown in the following figure:

Rotation mode: Moving distance = $\frac{\text{(3) Worktable travel per rotation [LREAL]} \times \text{(5) Motor gear ratio [UDINT]}}{\text{(1) Number of instruction pulses per motor rotation [UDINT]} \times \text{(4) Working gear ratio [UDINT]}} \times$ Number of pulses [pulse]

M: Motor, W: Worktable



(1) is the motor, (3) is the workpiece, (4) is the work gear ratio, and (5) is the motor gear ratio.

16.1.5.3 General Settings

- Basic settings



  - Axis type: Supports rotary mode and linear mode. The linear mode is suitable for motor scenarios involving reciprocating motion, such as lead screws and linear motors. The rotational mode is suitable for motor scenarios involving unidirectional rotation where cumulative position tracking is not required.
  - Encoder type: Supports incremental encoder and absolute encoder.
  - Ramp velocity: Supports trapezoidal and S-type motion planning. Trapezoidal (T-type) motion planning does not involve jerk parameter. S-type motion planning requires input of jerk parameter. $\sin^2$: The jerk parameter is only valid when the breakpoint acceleration drops to 0. If the initial acceleration is 0, the jerk parameter is invalid.
  - Data storage location: Applicable when the encoder type is set to absolute encoder; data can be saved on the servo side or the PLC side.
  - Position hysteresis management: Select "Disable" to turn off the excessive position deviation alarm. Select "Effective" to periodically check whether the deviation exceeds the set hysteresis limit.
- Period setting: Used in rotary mode to limit the rotation period.



- Software limit: Applicable in linear mode. When enabled, it limits the positive and negative positions of the axis.



- Software limit prediction parameter setting: Configures the axis limit protection by determining the risk between the current position and the limit position.



- Motion parameter limit: The figure below shows the upper limits for single-axis motion parameters. If the parameters in the motion instruction block exceed these limits, the function block will report a fault.



- Commissioning parameter limit: The figure below shows the limits for single-axis commissioning parameters. These limits only apply to MC_Jog and MC_Inch instructions, as well as the maximum speeds and acceleration/deceleration of the forward/reverse operations in the "Axis commissioning" function. If the set value exceeds the limit value, the system will operate at the limit value.



- Other settings



  - Positive torque maximum: The upper limit value of positive torque. The set parameter will be directly written to the 60E0 object via the SDO object.
  - Negative torque maximum: The upper limit value of negative torque. The set parameter will be directly written to the 60E1 object via the SDO object.
  - Maximum velocity: The upper limit speed. The set parameter will be multiplied by the gear ratio and then directly written to the 607F object via the SDO object.



- The maximum values of positive/negative torque and speed should be set appropriately. Excessively high settings will render the protection and restriction function ineffective.
- If the set maximum speed is lower than the homing speed, the homing speed of MC_Home will be executed according to the maximum speed set on the interface. When executing a torque control instruction, the speed parameter of the torque control instruction will be converted into pulse units and written to the 607F object internally via SDO.

- Monitor parameter setting



  - Positioning range: Used to set the positioning completion width, in reference units. The value is 0 or a real integer, with a default of 10.
  - Positioning check time: Used to set the positioning completion check time, in ms. The value range is 0 to 10000, with a default of 0. When set to 0, it indicates that the arrival check function is disabled.

- Option: If enabled, the axis status will not switch to ERRORSTOP upon limit hit or abnormal parameters. Use this option with caution.



16.1.5.4 Commissioning

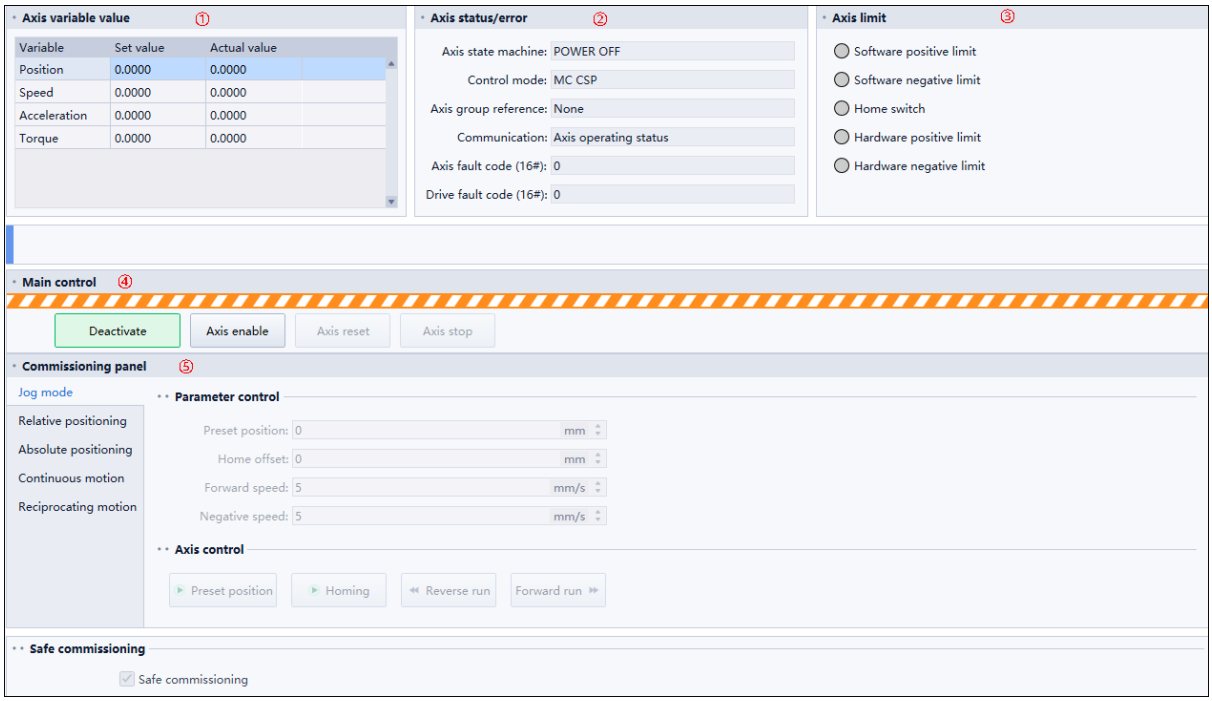
Basic functions such as enabling, positioning, reciprocation, homing, and stopping can be achieved through online commissioning. If the axis is connected to a bus stepper drive, the drive must support the relevant functions to perform homing or torque control. After confirming that basic motions are normal, complex motion control can be achieved using motion control function blocks.

NOTE

Axis commissioning and program motion control function blocks cannot be used simultaneously. Axis commissioning has the highest priority. After clicking "Activate commissioning", any execution of motion control function blocks will trigger a fault, and the motion control functions will be invalid. The functions can be used normally again after commissioning is disabled, and the function block is reset and triggered again.

Supported axis types: bus-controlled axes, local pulse axes, and virtual axes.

1. When the device is online and switched to the operational status, the "Activate commissioning" button can be clicked to activate axis commissioning. The interface will be as shown in the figure below.



No.	Description
①	Axis-end information including position, speed, acceleration, and so on.
②	Axis status and drive fault information
③	Axis-end limit terminal status
④	Buttons for activating/deactivating commissioning, enabling axis, resetting axis and stopping axis.
⑤	Jog or other motion mode options

2. Click the "Axis enable" button to enable the axis. You can then view the axis information including the current position, speed, status, and so on.
3. By setting different motion modes and motion parameters, the overall operational status of the servo can be viewed.
4. Safe commissioning: Enabled by default. When axis commissioning is activated, switching pages or closing the commissioning page will force the axis to stop.
   The axis must be stopped in the following cases:
   - The current commissioning page is switched.
   - The current commissioning page is closed.
   - The iFA software is closed.

   The axis will not be stopped in the following cases:
   - The axis commissioning page remains unchanged in multi-view mode.
   - The commissioning page is minimized.
5. Additional notes
   - The commissioning mode is used for axis commissioning. Motion control function blocks in this mode will report a fault.
   - In commissioning mode, the axis will be stopped automatically if the software is closed, the project is switched, or the PLC is re-downloaded and installed.
   - In commissioning mode, the oscilloscope can be used normally.
   - In commissioning mode, the I/O mapping data of all slave modules remain unaffected.
   - During single-axis commissioning, enabling servo commissioning will cause an enabling conflict. The servo commissioning must be disabled before using axis commissioning.
   - In commissioning mode, parameter anomalies and other faults will cause the axis to enter the ERRORSTOP status.
6. Common functions on this interface

Option	Note
Axis variable value	Supports monitoring of set and actual values of axis position, speed, acceleration, and torque.
Axis limit	Displays software positive limit, software negative limit, home switch, hardware positive limit, and hardware negative limit.
Activate commissioning	After the software is online, all parameters on the commissioning interface will be ineffective if this button is not clicked. Once clicked, the "Axis enable" button becomes operable. During axis operation, deactivating commissioning will switch the axis to the power-off status.
Enable axis	Once clicked, the axis status will be switched to StandStill; the jog mode parameters and other mode parameters on the commissioning interface become editable.
Axis reset	If a fault occurs after the axis is enabled, this button can be used to reset the axis.
Axis stop	This button is effective regardless of the motion mode of the axis.

Option	Note
Jog mode	◦ Sets the speed for forward and reverse jogging. The unit is converted according to the settings on the unit conversion interface. ◦ Supports preset value configuration. Input the preset value and click "Settings" to apply it. The unit is converted according to the settings on the unit conversion interface. ◦ Control the jogging direction of the motor by clicking "Jog backward"/"Job forward". ◦ The running parameter for jog mode serves as both the speed parameter and the acceleration parameter.
Other modes	Relative positioning: Parameters include target distance, speed, acceleration, deceleration, and jerk. Absolute positioning: Parameters include target distance, speed, acceleration, deceleration, and jerk. Continuous motion: Parameters include target speed, acceleration, deceleration, and jerk. Reciprocating motion: Parameters include two sets of target positions. Torque control: Parameters include target torque, torque ramp, and limited speed.

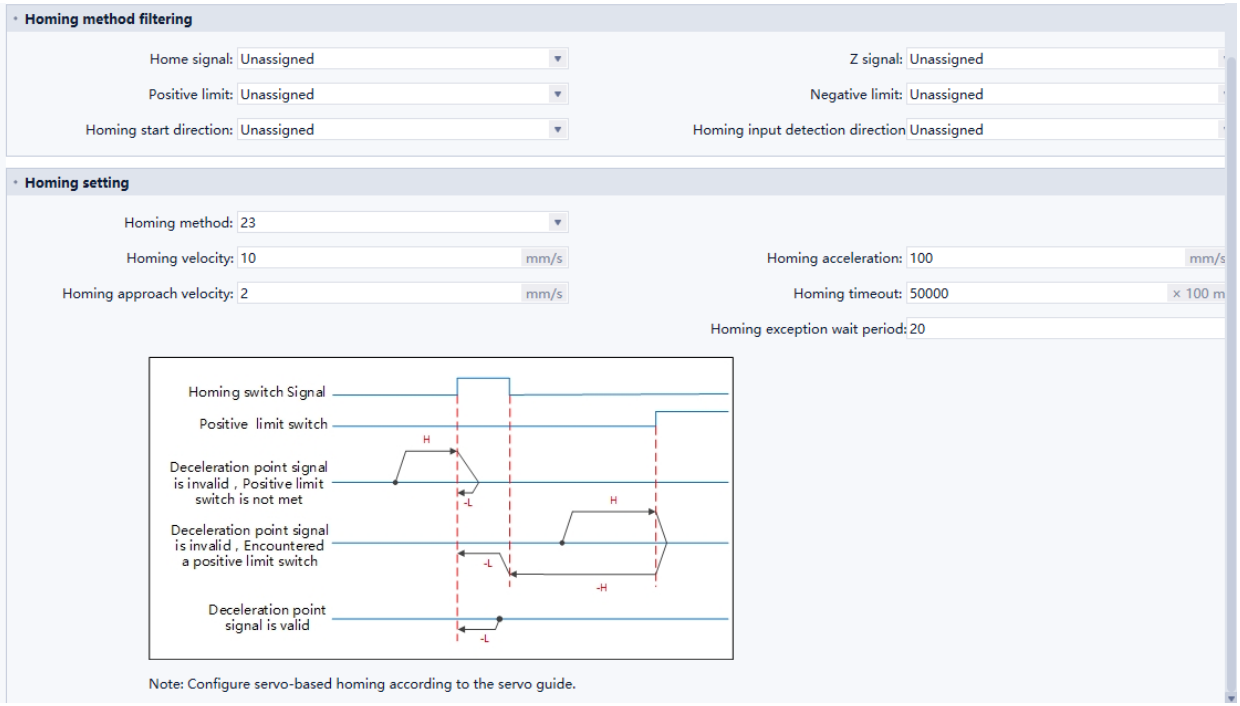
16.1.5.5 Homing Parameter Settings

On the homing parameter setting interface, two homing modes can be configured, namely servo-based homing and controller-based homing.



Servo-based homing

The parameters of servo-based homing involve homing method filtering and homing settings. Homing method filtering is used only to filter the options in the homing list and does not require PLC program download. Homing setting parameters are primarily used for graphical parameter configuration of axis-end servo homing. Graphical configuration guidelines are provided at the bottom of the interface. Follow the guidelines to select the homing method from the drop-down menu and complete the corresponding parameter configuration. It supports homing methods 1 to 35 and -1, -2, -3 as defined by the CiA402 protocol. For axes bound to bus expansion modules (e.g., 4PME), only homing methods 17 to 35 are supported.



Parameter		Description
Homing method filtering	Home signal	Used to select whether to use the home signal. • When "Unassigned" is selected, it is not used as a mandatory filter condition. • When "Used" is selected, homing methods that do not support the home signal will be excluded. • When "Not used" is selected, homing methods that require the home signal will be excluded.
	Z signal	Used to select whether to use the motor Z signal. • When "Unassigned" is selected, it is not used as a mandatory filter condition. • When "Used" is selected, homing methods that do not support the Z signal will be excluded. • When "Not used" is selected, homing methods that require the Z signal will be excluded.
	Positive limit	Used to select whether to use the hardware right limit signal. • When "Unassigned" is selected, it is not used as a mandatory filter condition. • When "Used" is selected, homing methods that do not support the positive limit signal will be excluded. • When "Not used" is selected, the homing modes that require the positive limit signal will be excluded.

Parameter		Description
	Negative limit	Used to select whether to use the hardware left limit signal. • When "Unassigned" is selected, it is not used as a mandatory filter condition. • When "Used" is selected, homing methods that do not support the negative limit signal will be excluded. • When "Not used" is selected, homing methods that require the negative limit signal will be excluded.
	Homing direction	Used to set the direction of motion at the beginning of homing. • Positive: The direction of motion is forward when the limit (home) signal input is inactive, otherwise it is the opposite. • Negative: The direction of motion is reverse when the limit (home) signal input is inactive, otherwise it is the opposite.
	Homing input detection direction	The direction of motion when the home signal is reached. • Positive: Stops at the edge of the home signal during forward motion. • Negative: Stops at the edge of the home signal during reverse motion.
Homing Settings	Homing method	The range is -3, -2, -1, and 1 to 35. It is written to object 0x6098 via startup parameters.
	Homing velocity	The homing velocity is written into the sub-index No. 1 of the object dictionary 0x6099 in the form of startup parameters after the user units are converted into pulse units.
	Approach velocity	Indicates the homing approach velocity. It is written into the sub-index No. 2 of the object dictionary 0x6099 in the form of startup parameters after the user units are converted into pulse units.
	Homing acceleration	The homing acceleration is written into the object dictionary 0x609A in the form of startup parameters after the user units are converted into pulse units.
	Homing timeout	This parameter is exclusive to Inovance drives.
	Homing exception wait time	Number of cycles to wait from a homing exception to the reporting of a homing error. Default: 20 cycles.



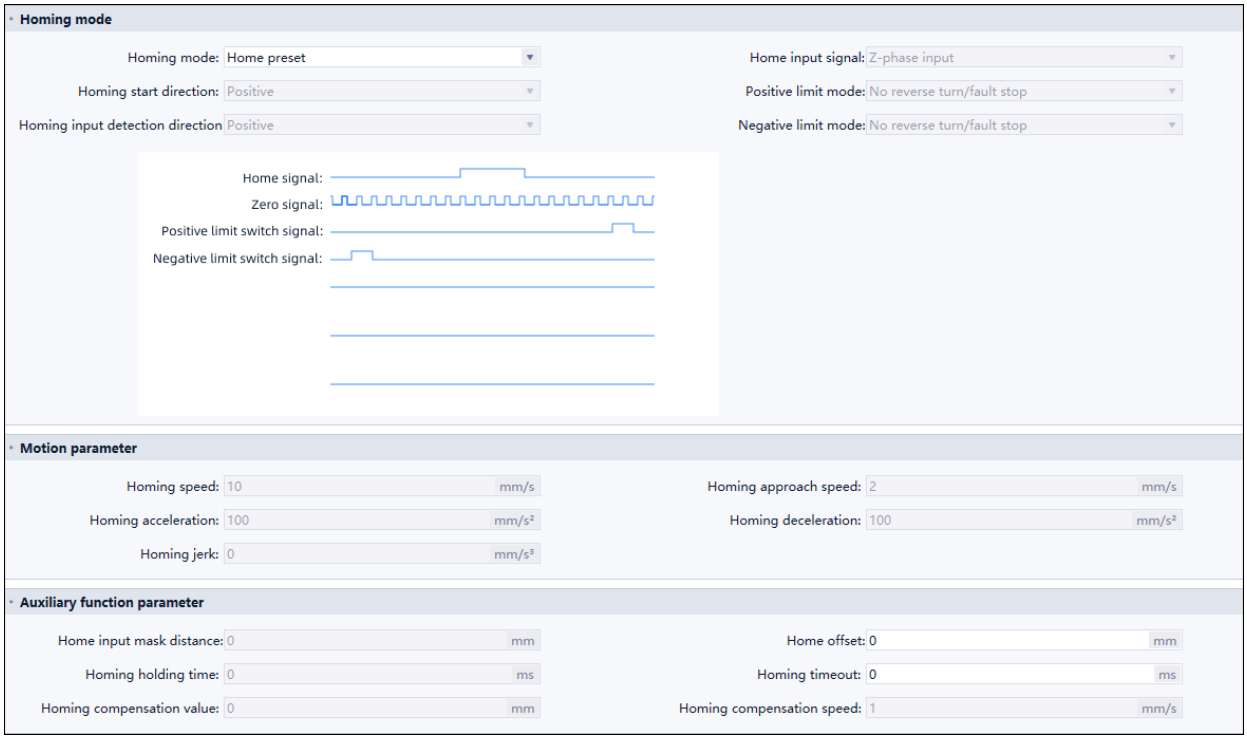
Parameter settings for servo-based homing are effective for the MC_Home instruction.

Associated parameter limits

Parameter	Affected Parameters
Positive limit	When using positive limit: • Negative limit: Not used. • Homing direction: Positive.
Negative limit	When using negative limit: • Positive limit: Used. • Homing direction: Negative.
Home input/output direction	When the home input/output direction is positive or negative, the home signal is used.

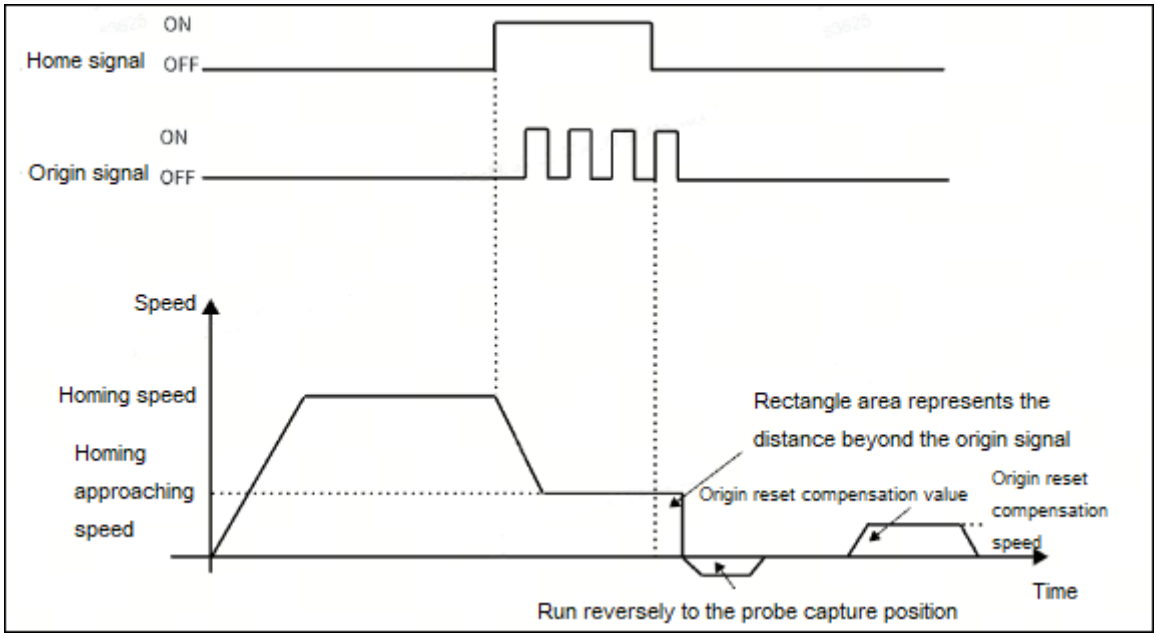
Controller-based homing

The following figure shows the controller-based homing interface. This function must be used with the MC_Homing instruction for homing. In this mode, the controller independently plans the homing trajectory and determines the final mechanical home using limit sensors, home switches, and probe signals. The controller-based homing interface mainly includes the settings of homing mode, motion parameters, and auxiliary function parameters.



- Homing mode: Includes selection of homing method, home input signal, homing direction, homing input detection direction, positive limit mode, negative limit mode, and homing trajectory preview.
- Motion parameters: Includes settings for homing speed, homing approach speed, homing acceleration, homing deceleration, and homing jerk.
- Auxiliary function parameters: Includes settings for home input mask distance, homing holding time, homing compensation value, homing compensation speed, homing offset, and homing timeout.

Introduction to controller-based homing:



After the homing instruction is initiated, the motor accelerates to the homing speed at the set homing acceleration, and runs at a constant speed to search for the home switch signal. Upon detecting the rising edge of the home switch signal, the motor decelerates at the set homing deceleration to the approach speed and continues running at this speed. When the rising edge of the first zero signal is detected after the home switch drops to a low level, the home is considered found. The motor then performs an emergency stop at the software maximum deceleration and latches the position of the rising edge of the current zero signal. After the motor stops, it moves in reverse to the position where the zero signal was latched. If a compensation value is set, the motor pauses for 500 ms and then moves the distance specified by the homing compensation value at the homing compensation speed before stopping. The entire homing process is then complete.

For detailed functionality, see 18.3.3.3.53 MC_Homing.

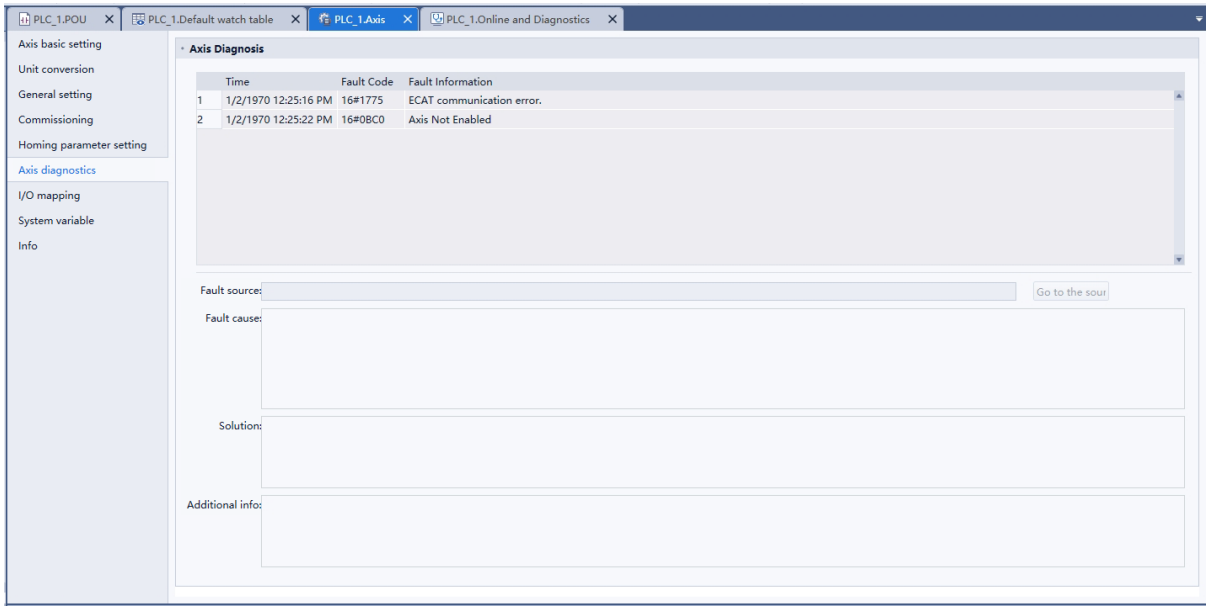
NOTE

Controller-based homing only modifies the position system of the controller and does not modify the position system of the servo drive.

16.1.5.6 Axis Diagnostics

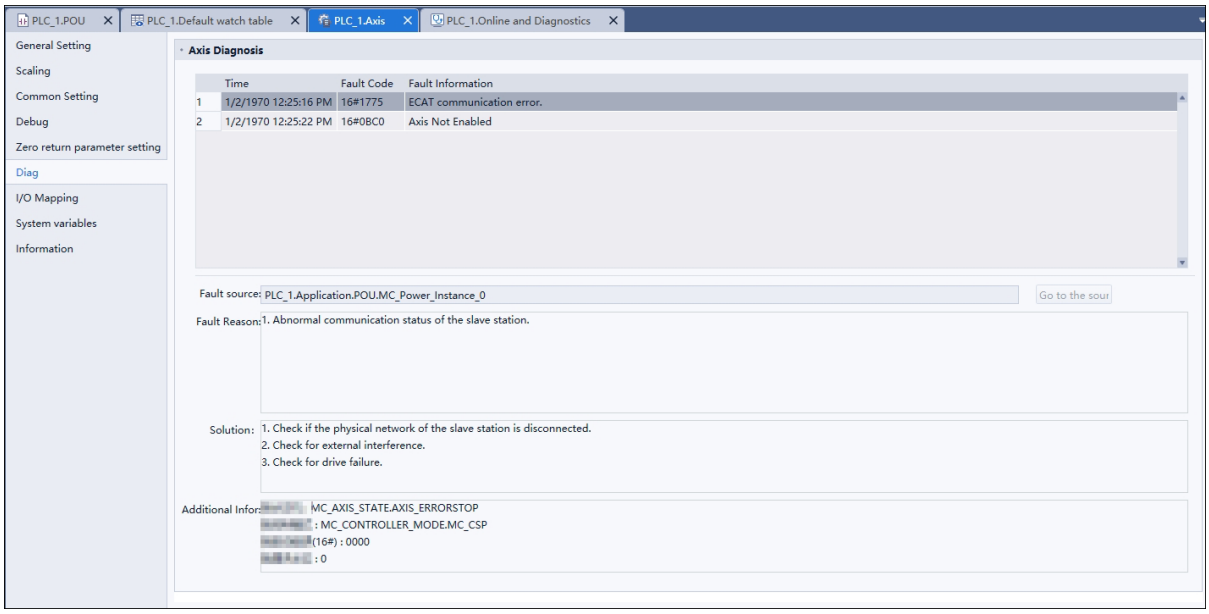
Axis diagnostics are a critical component for ensuring stable operation of automation equipment and improving overall system reliability. Through axis diagnostics, issues can be quickly and accurately identified and solved, reducing equipment downtime, increasing production efficiency, and ensuring continuous operation. Users can locate and track axis faults based on axis diagnostic information and resolve issues according to the provided solutions.

The axis diagnostic list is shown in the figure below.



- Time: The time when the fault occurred.
- Fault code: Displayed in hexadecimal by default. For details, see 18.3.3.8 Fault Codes for all fault codes, corresponding information, causes, and solutions.
- Fault message: A brief summary of the fault.

Click a fault entry in the axis diagnostic list to view the fault source, cause, solution, and additional information.



- Fault source: Displays the root cause of the fault.
- Fault cause: The cause corresponding to the fault information.
- Solution: The corrective action corresponding to the fault information.
- Additional information: Additional details, including axis state machine, axis control mode, axis drive fault, and axis communication status.

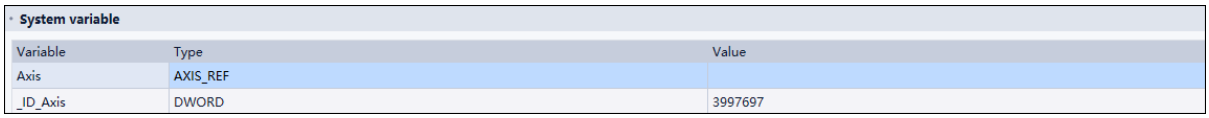
16.1.5.7 I/O Mapping

All task names and bus cycle settings will be displayed in the bus cycle options. For axis objects (including control axes and encoder axes), it is recommended to prioritize the EtherCATMaster_Task**.** Selecting other tasks may result in abnormal conditions such as planned speed jump.



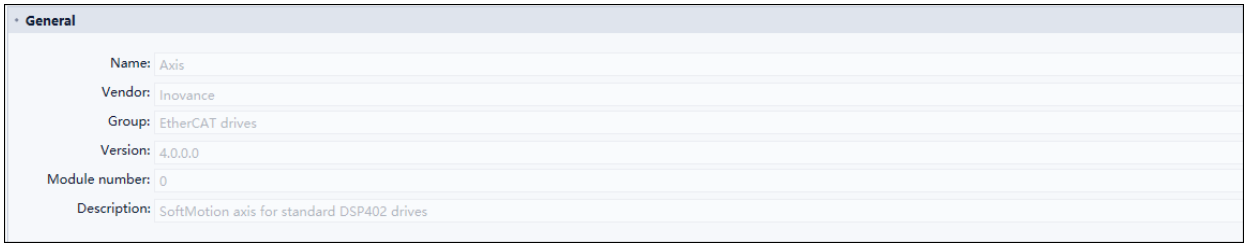
16.1.5.8 System Variables

The "System variable" tab displays the data structure and device ID of the axis.



16.1.5.9 Information

This interface displays basic information about the axis, including name, vendor, group, version, module number, and description.



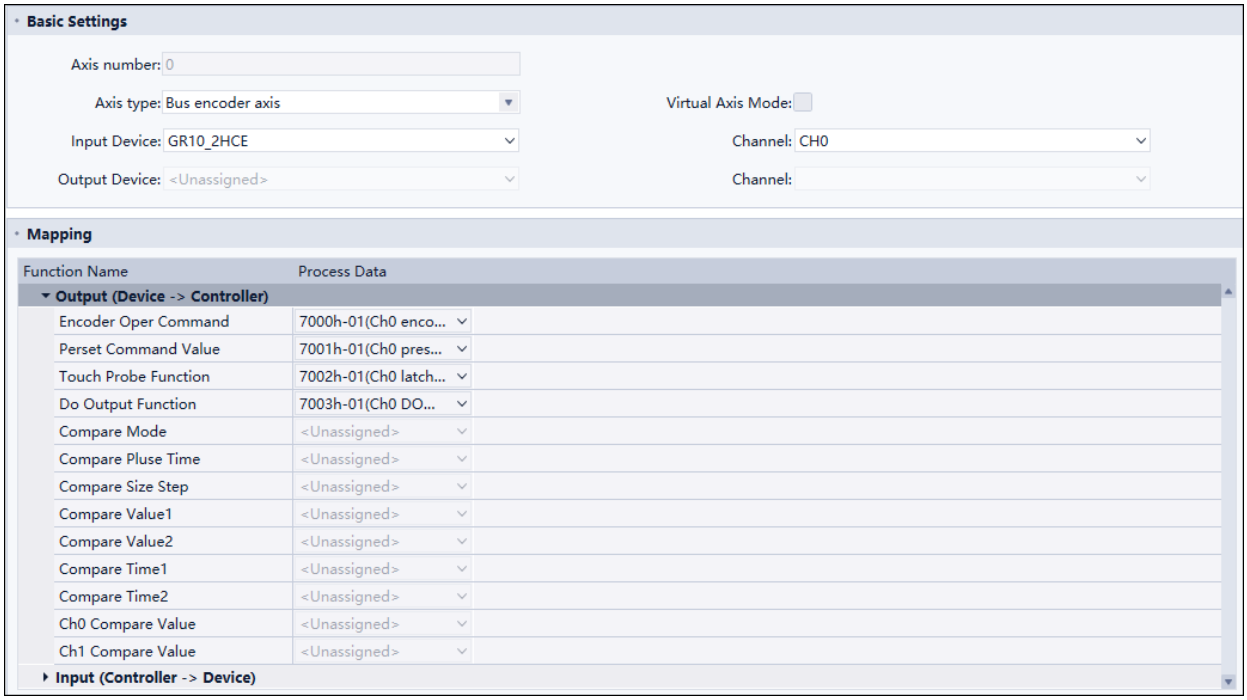
16.1.6 Bus Encoder Axis

16.1.6.1 Overview

The bus encoder axis supports a maximum input pulse frequency of 200 kHz. Taking the GR10-2HCE module as the input device for illustration, the bus encoder axis can count three forms of pulses: phase A/B, pulse+direction, and CW/CCW. One GR10-2HCE module has two channels, each of which can be assigned for one axis. Bus encoder axes can also work with the DI and DO terminals of the GR10-2HCE module to implement position presetting, probe, gating, and comparison output. The feedback position of the GR10-2HCE module in pulses is sent to the PLC through PDOs, and the feedback position within the PLC is finally provided to users in the form of LREAL data type through the conversion of gear ratios, which can also be used as the master axis of a cam or gear.

16.1.6.2 Basic Axis Settings

The EtherCAT bus expansion module must be configured in the device and network before selecting the bus encoder axis. After selecting the input device, the channel will be automatically configured by default.

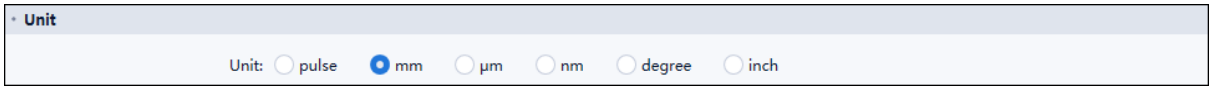


NOTE

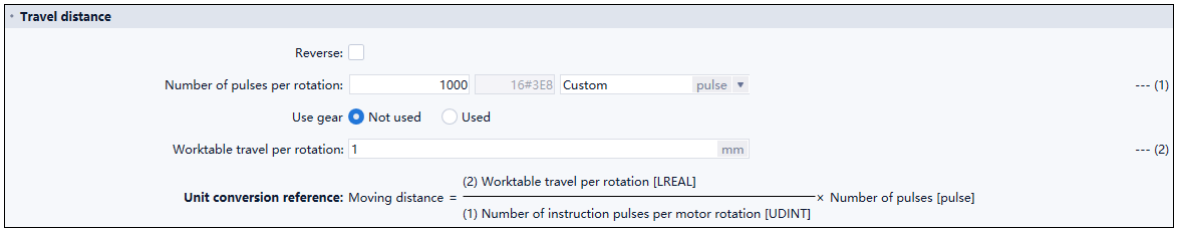
The input device for the bus encoder axis currently only supports Inovance's encoder modules.

16.1.6.3 Unit Conversion

- Unit: Conversion between six units, including pulse, millimeter (default), micron, nanometer, degree, and inch. The unit can be configured as needed.



- Travel distance: Conversion between user unit and pulse unit.
  For bus-controlled axes or local pulse axes, the user unit (such as pulse, millimeter, or micron) is used on the instruction side, while the pulse unit is used for actual control of the physical motor. Gear ratio coefficient needs to be configured for unit conversion. The conversion modes are divided into the following scenarios:
  - Gears not used
    When gears are not used, the conversion formula from user unit to pulse unit is as follows:



> **NOTE**
>
> The maximum input pulse frequency for the bus encoder axis GR10-2HCE is 200 kHz, so the default number of instruction pulses per motor revolution for the bus encoder axis is 16#3E8, which is 1000 in decimal.

Taking the Inovance 20-bit encoder as an example, set the parameters as follows:

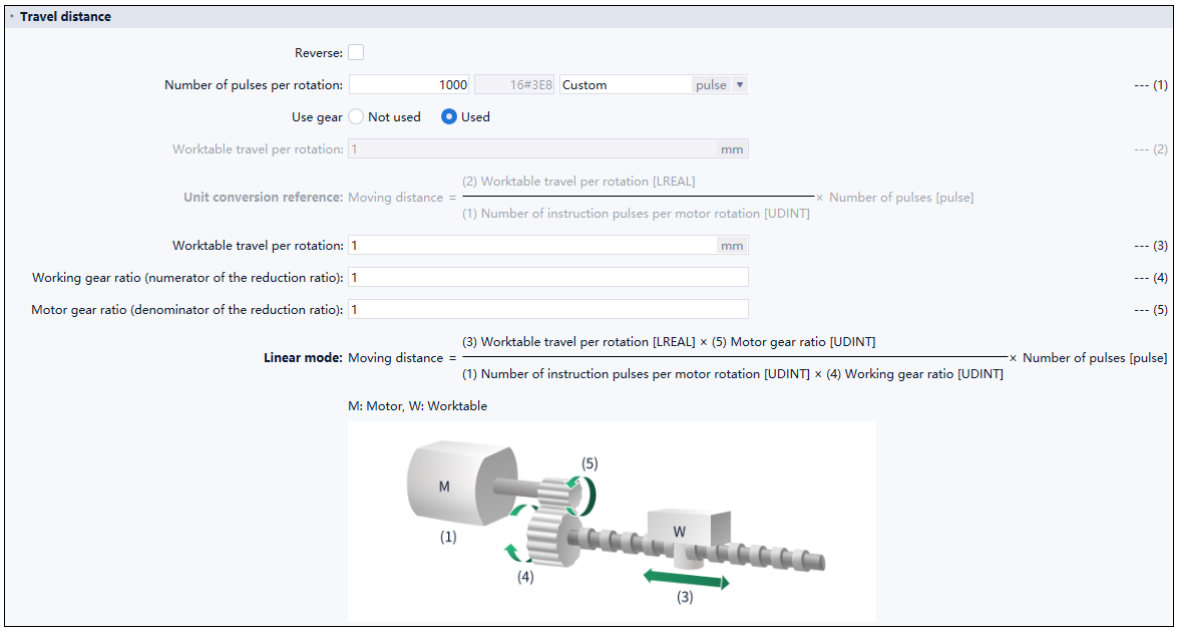
The number of pulses per motor revolution is default to 10#1048576. Number of bits for common encoders can be directly selected, which currently supports 14-bit, 18-bit, 20-bit, 23-bit, and 26-bit encoders. The number of pulses per motor revolution can also be customized, supporting both decimal and hexadecimal input formats. Hexadecimal input requires the "16#" prefix; otherwise, the system will automatically process the value as decimal.

The working stroke per revolution of the worktable is 1.

Based on the above parameters, if the target distance input on the relative motion instruction side is 100, the control axis will output an actual pulse count of 104857600, causing the motor to rotate 100 revolutions. Without a reducer, the worktable moves 100 user units.

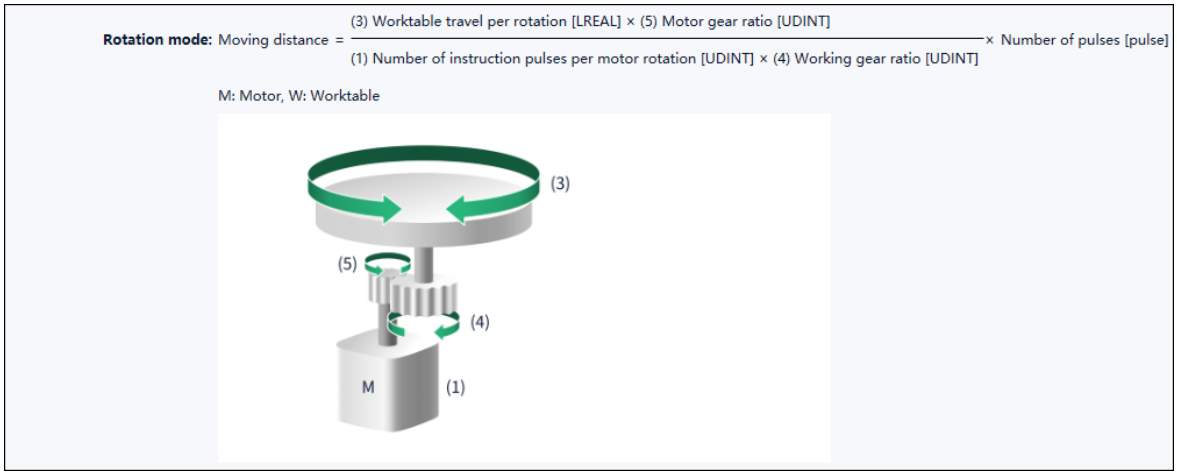
- Gears used

  The typical scenario for linear mode is shown in the following figure:



(1) is the motor, (3) is the workpiece, (4) is the work gear ratio, and (5) is the motor gear ratio.

The typical scenario for rotary mode is shown in the following figure:



(1) is the motor, (3) is the workpiece, (4) is the work gear ratio, and (5) is the motor gear ratio.

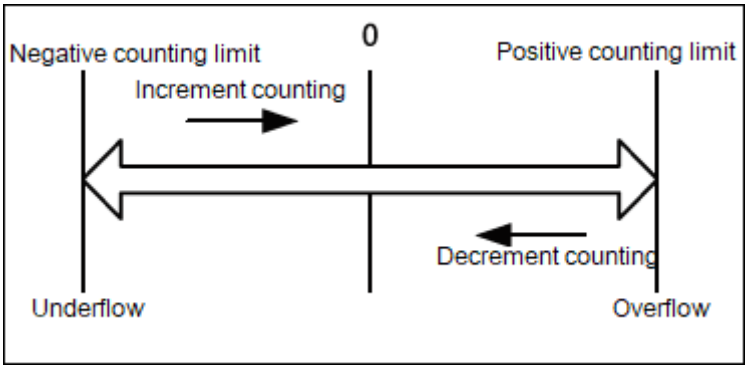
16.1.6.4 Counter Settings

1. Mode



- Linear mode

  In linear mode, the software limit can be enabled or disabled. After enabling the software limit, the positive limit value and negative limit value can be set. When the counter reaches the limit value, counting stops, and the PostiveLimit or NegativeLimit flag in the ENC_Counter instruction is set.



When software limit is disabled, the module's count value cycles within the range of -2147483648 to 2147483647. When forward counting reaches the maximum positive value, it directly jumps to the maximum negative value and continues counting. The reverse is the same.

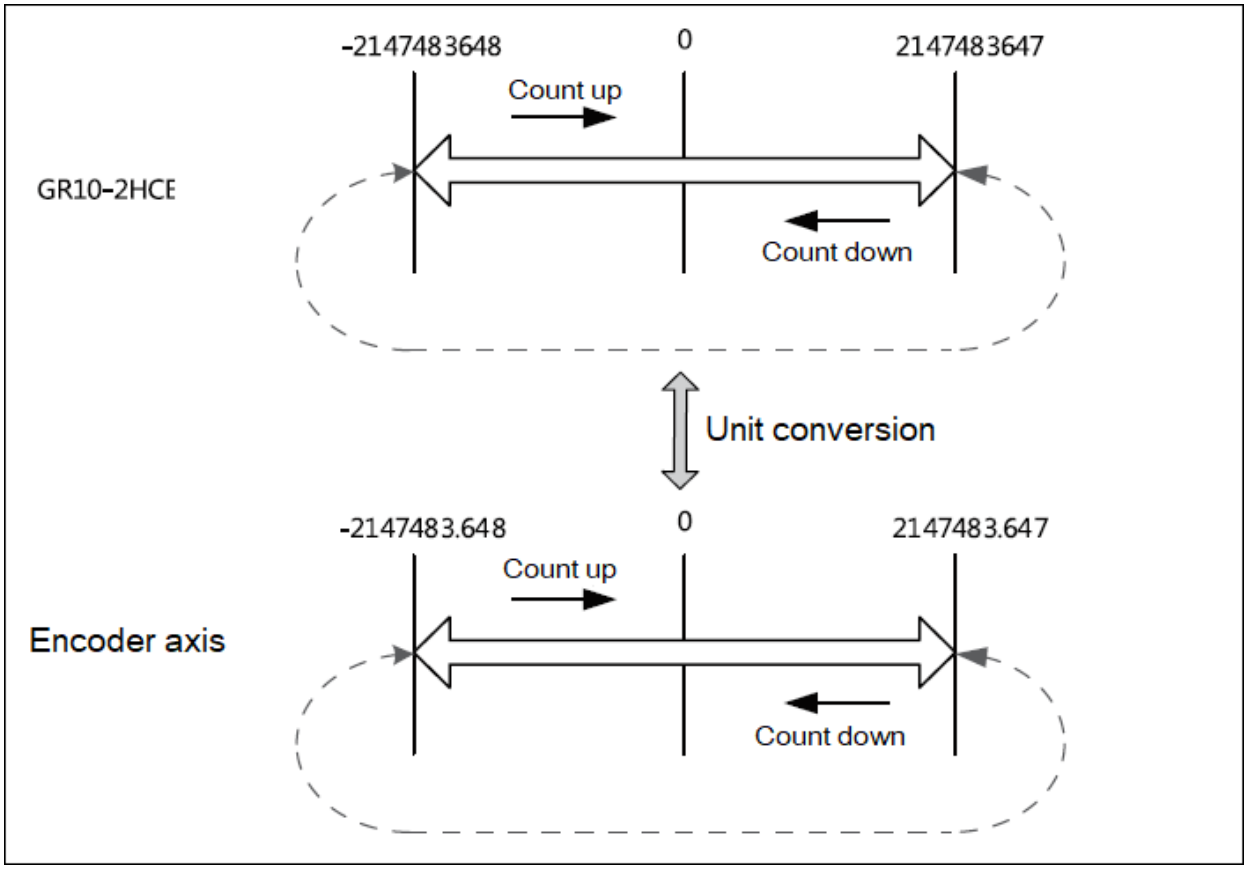
Example:

The actual position of the bus encoder axis is related to the gear ratio.

Number of pulses per motor revolution = 1000.

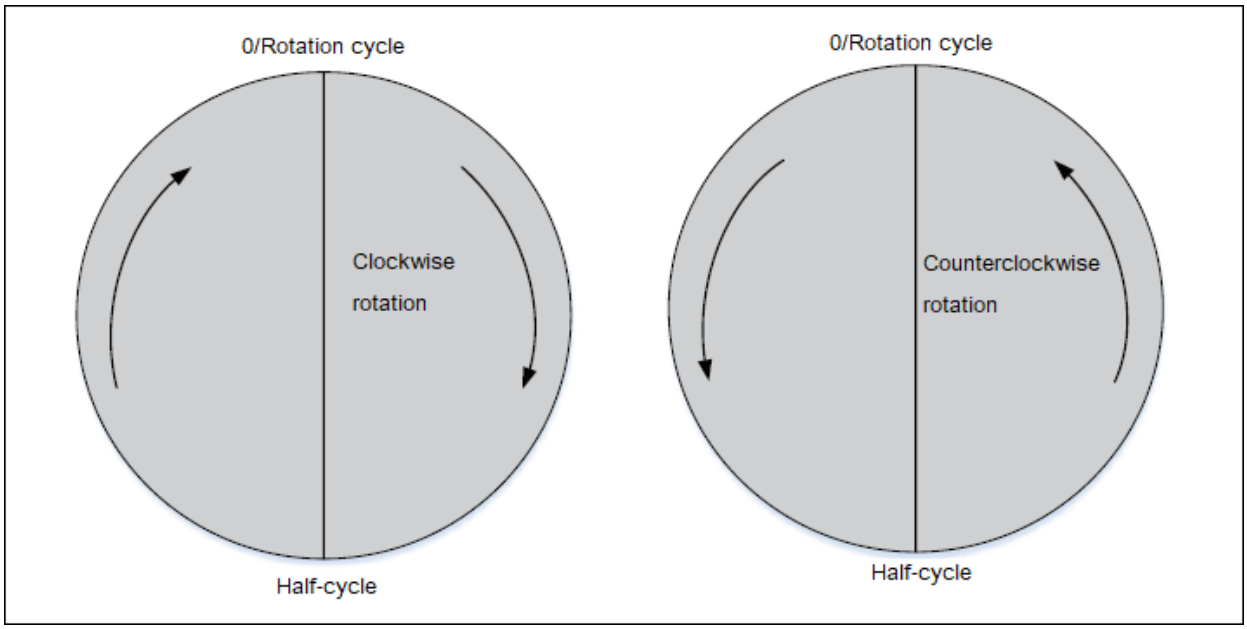
Working stroke per worktable revolution = 1.

All counting units in the module's PDO objects are in pulse units, while the counting units displayed at the axis end are in user units.



◦ Rotary mode

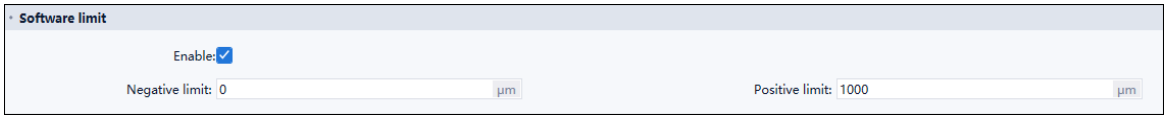
In rotary mode, the rotation period can be set. The counter will count reciprocally between 0 and the rotation period.



2. Period: Used in rotary mode to limit the rotation period.



3. Software limit: Used in linear mode. When enabled, it limits the positive and negative positions of the axis.



4. Counter parameters



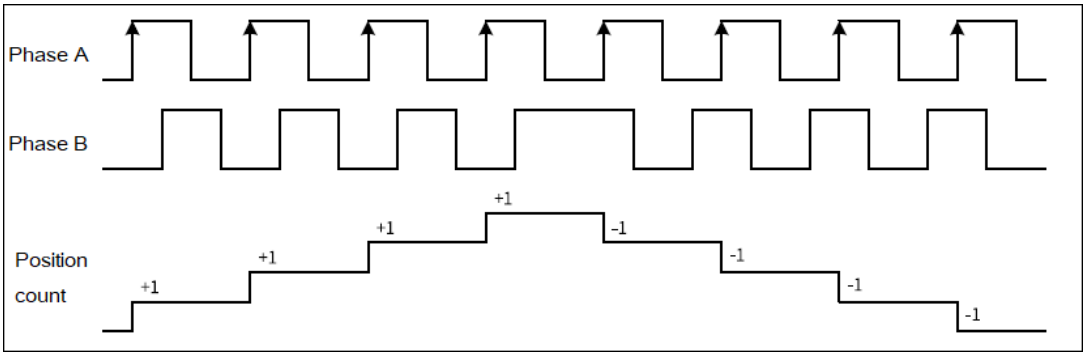
- Counting mode

  Five count modes are supported: phase A/B 1-frequency multiplication, phase A/B 2-frequency multiplication, phase A/B 4-frequency multiplication, pulse+direction, and CW/CCW.

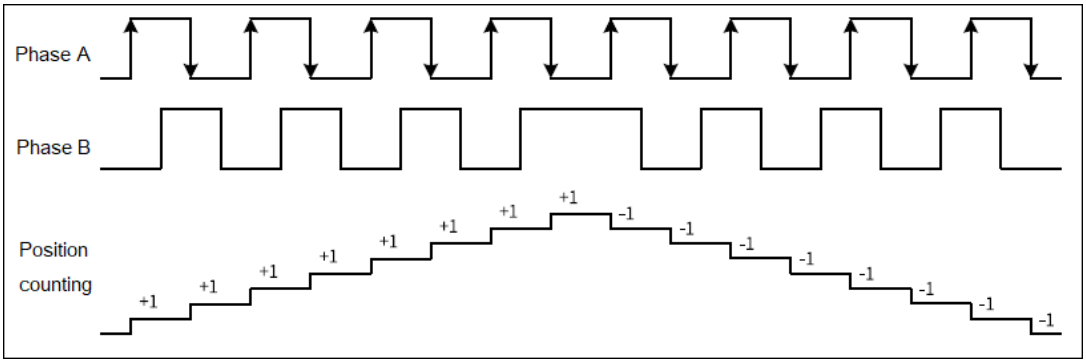
  - Phase A/B pulse

    In phase A/B mode, the encoder generates two orthogonal phase pulse signals with a phase difference of 90°, that is, phase A signal and phase B signal. When the phase A signal leads the phase B signal, the counter counts up; when the phase B signal leads the phase A signal, the counter counts down. The phase A/B pulse can be set to operate in frequency multiplication by 1, 2, or 4.

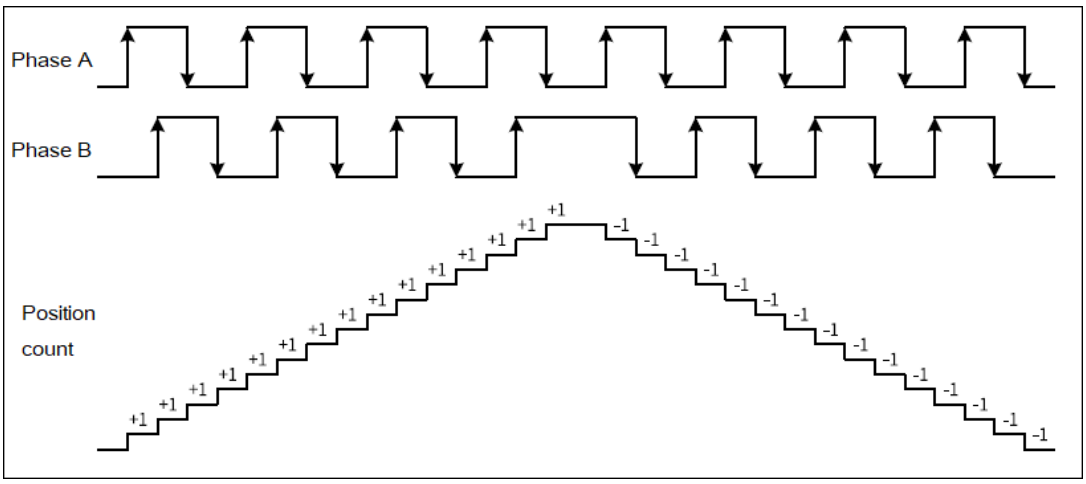
    When set to phase A/B 1-frequency multiplication, counting is performed only on the rising edge of the A-phase pulse, as shown in the figure below.



When set to A/B phase 2-frequency multiplication, counting is performed on both the rising and falling edges of the A-phase pulse, as shown in the figure below.

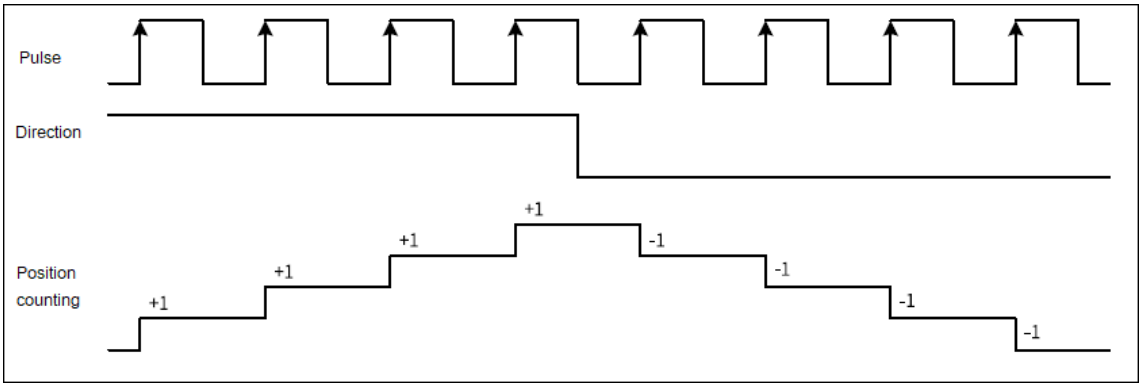


When set to A/B phase 4-frequency multiplication, counting is performed on the rising and falling edges of both the A-phase and B-phase pulses, as shown in the figure below.



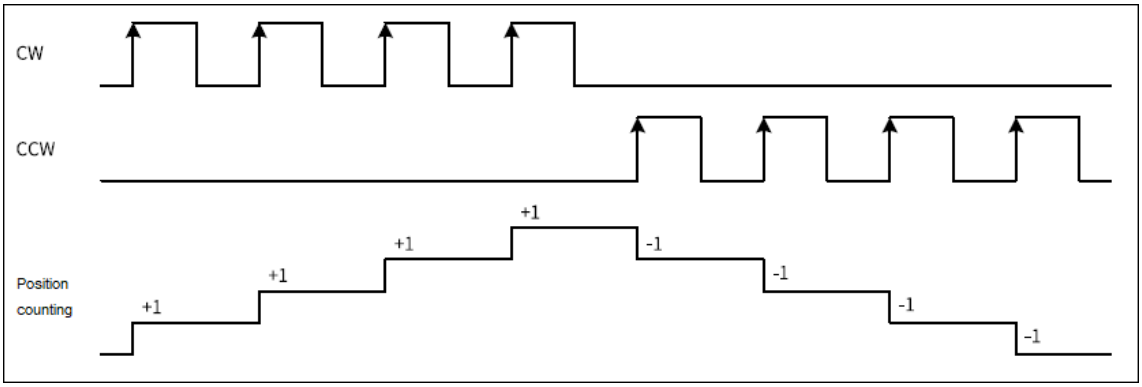
- Pulse + direction

  Phase A serves as the pulse input for the counter, and phase B serves as the direction control input for the counter. When the B-phase signal is at high level, the counter increments; when it is at low level, the counter decrements.



- CW/CCW mode

  Clock Wise (CW) is the forward pulse signal, and Counter Clock Wise (CCW) is the reverse pulse signal. When the encoder is forward running, CW pulse signals are output; when the encoder is reverse running, CCW pulse signals are output.



◦ Counting logic

Used to set the logic of counting direction as follows:

Table16-1-6-4-1 Counting logic description

Counting Mode	Positive Logic	Negative Logical
Phase A/B	Incremental count if phase A leads phase B. Decremental count if phase B leads phase A.	Decremental count if phase A leads phase B. Incremental count if phase B leads phase A.
Pulse + direction	Incremental count if phase B is at a high level. Decremental count if phase B is at a low level.	Decremental count if phase B is at a high level. Incremental count if phase B is at a low level.
CW/CCW	Incremental count if it is phase A pulse input. Decremental count if it is phase B pulse input.	Decremental count if it is phase A pulse input. Incremental count if it is phase B pulse input.

- Frequency sampling period
  Used to set the pulse frequency calculation period.
- Input filter time
  Used to set the filter time for input terminal signals. The default is 2 μs, indicating that waveforms below 2 μs are filtered out.

5. External input



Each counter channel can independently configure four digital input terminals. The functions of each terminal are as shown in the table below, where n represents the channel number.

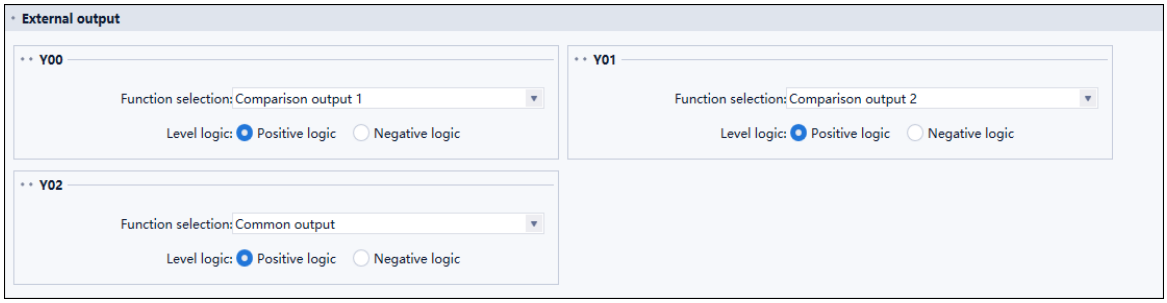
Table16-1-6-4-2 Input terminal function selection

Terminal	Optional Function	Default
Xn0	0: Common input 1: Probe function 1 3: Reset 4: Preset 5: Gating	Probe function 1

Terminal	Optional Function	Default
Xn1	0: Common input 2: Probe function 2 3: Reset 4: Preset 5: Gating	Probe function 2
Xn2	0: Common input 3: Reset 4: Preset 5: Gating	Common input
Xn3	0: Common input 3: Reset 4: Preset 5: Gating	Common input

- Common input
  When the terminal is configured as a common input, the terminal status can be obtained via module I/O mapping or the ENC_ReadStatus instruction.
- Probe function
  When the probe function is configured, the ENC_TouchProbe instruction must be called to latch the position. For details, see the ENC_TouchProbe instruction description.
- Reset function
  When the reset function is configured, the counter value will be cleared upon a valid input signal at the digital input terminal.
- Preset function
  When the preset function is configured, the ENC_Preset instruction must be called to preset the counter. When the input signal of the digital input terminal is active, the counter value will be set to the preset value specified by the Position parameter in the ENC_Preset instruction. If multiple terminals are configured as preset functions, only one terminal signal needs to be active for the preset function to be executed. Digital terminals can be set using the ENC_DigitalOutput instruction.
- Gating function
  When the gating function is configured, the counter starts only when the Enable pin of ENC_Counter is ON and the gating signal is active. If no terminal is configured for gating function, the counter starts as long as the Enable pin of ENC_Counter is set to ON. If multiple terminals configured for gating function, the counter starts only when all gating signals are active simultaneously.

6. External output



- Output terminal function selection

  Output terminal function description (n represents channel number)

Terminal	Optional Function	Default
Yn0	0: Common output 3: One-dimensional comparison output 4: Two-dimensional comparison output (currently not supported)	One-dimensional comparison output
Yn1	Common output	Common output
Yn2	Common output	Common output

- Common output

  When the terminal is configured as a common output, the status of the digital output terminal can be controlled via the ENC_DigitalOutput instruction. For specific usage, see 18.3.3.6.9 ENC_DigitalOutput.

- One-dimensional comparison output

  When one-dimensional comparison output is configured, the step comparison output or array comparison output functions can be performed. For specific usage, see 18.3.3.6.10 ENC_Compare, 18.3.3.6.7 ENC_StepCompare, and 18.3.3.6.6 ENC_ArrayCompare.

16.1.6.5 Axis Diagnostics

For details, see 16.1.5.6 Axis Diagnostics.

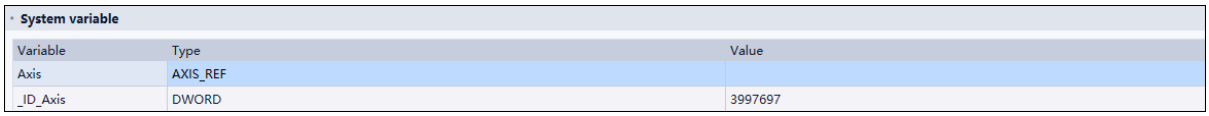
16.1.6.6 I/O Mapping

All task names and bus cycle settings will be displayed in the bus cycle options. For axis objects (including control axes and encoder axes), it is recommended to prioritize the EtherCATMaster_Task**.** Selecting other tasks may result in abnormal conditions such as planned speed jump.



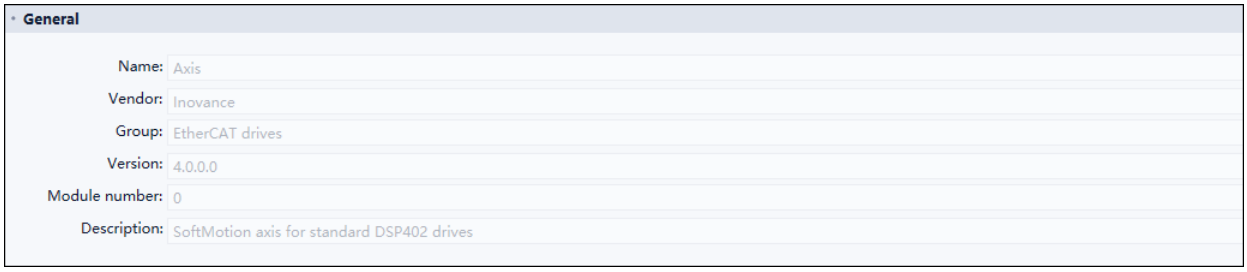
16.1.6.7 System Variables

The "System variable" tab displays the data structure and device ID of the axis.



16.1.6.8 Information

This interface displays basic information about the axis, including name, vendor, group, version, module number, and description.



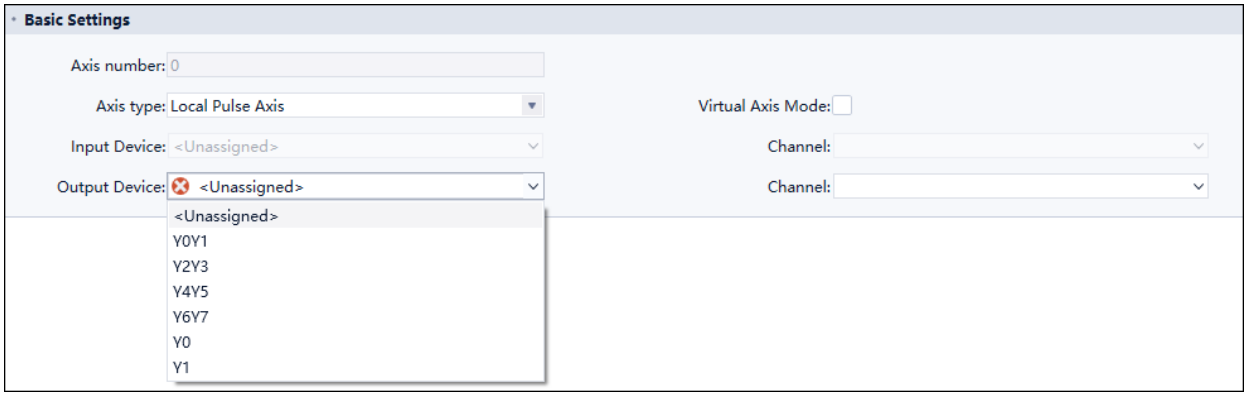
16.1.7 Local Pulse Axis

16.1.7.1 Overview

Local pulse axis is controlled using a pulse drive controlled by local high-speed I/O. For dual-end configuration, up to four local pulse axes can be configured: Y0/Y1, Y2/Y3, Y4/Y5, and Y6/Y7. Each pulse output channel can be configured as A/B phase (default to 4-frequency multiplication), pulse + direction, or CW/CCW. Up to two probe terminals can be set per pulse output channel. The local pulse axis supports control in several basic modes such as point, speed, and homing modes. Torque mode is not supported. The parameter configuration and instruction usage of local pulse axes are largely consistent with bus-controlled axes.

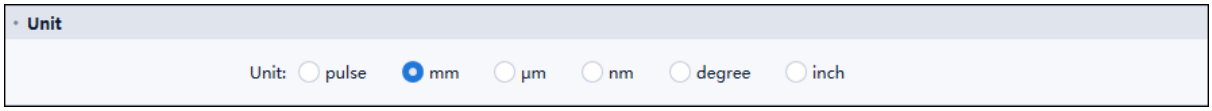
16.1.7.2 Basic Axis Settings

After selecting "Local pulse axis" as the axis type, the "Output device" option is activated. Corresponding high-speed I/O terminals can be selected for the output device. Y0 and Y1 can be used as single-ended outputs. The local pulse axis supports a maximum output frequency of 200 kHz. When setting the gear ratio and speed parameters, ensure that the calculated output frequency does not exceed this upper limit.

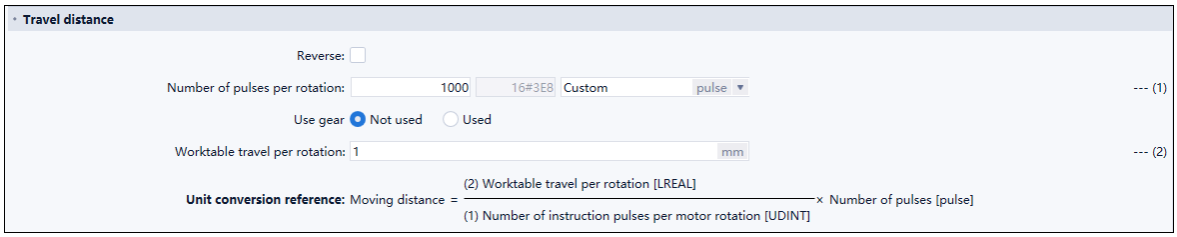


16.1.7.3 Unit Conversion

- Unit: Conversion between six units, including pulse, millimeter (default), micron, nanometer, degree, and inch. The unit can be configured as needed.



- Travel distance: Conversion between user unit and pulse unit.
  For bus-controlled axes or local pulse axes, the user unit (such as pulse, millimeter, or micron) is used on the instruction side, while the pulse unit is used for actual control of the physical motor. Gear ratio coefficient needs to be configured for unit conversion. The conversion modes are divided into the following scenarios:
  - Gears not used
    When gears are not used, the conversion formula from user unit to pulse unit is as follows:



    Taking the Inovance 20-bit encoder as an example, set the parameters as follows:

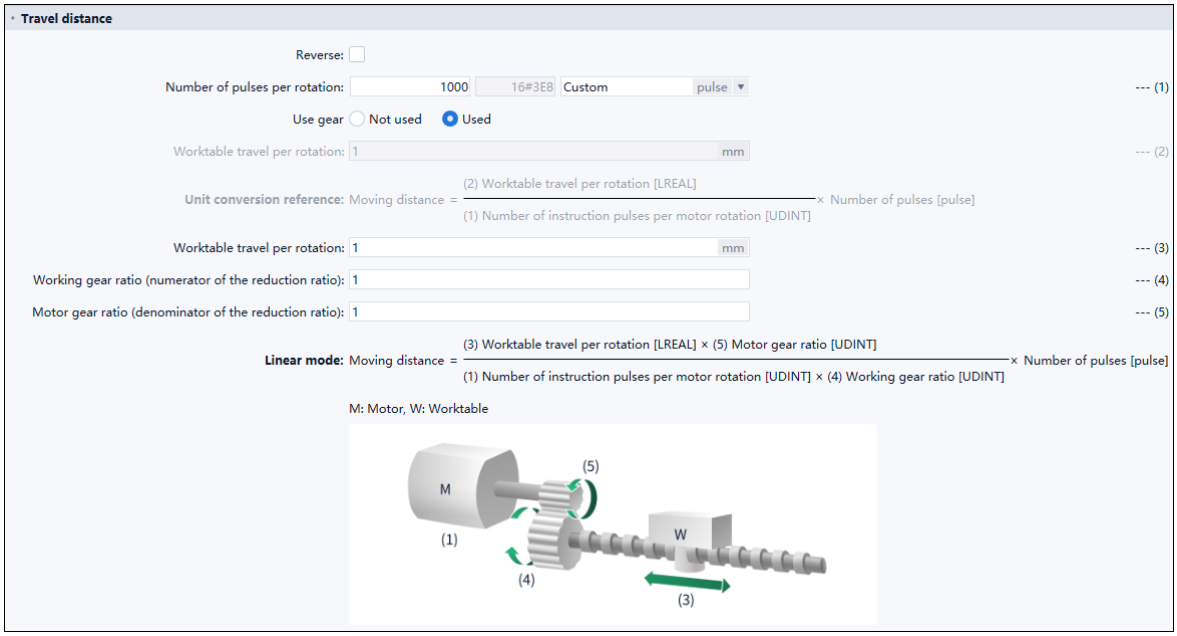
    The number of pulses per motor revolution is default to 10#1048576. Number of bits for common encoders can be directly selected, which currently supports 14-bit, 18-bit, 20-bit, 23-bit, and 26-bit encoders. The number of pulses per motor revolution can also be customized, supporting both decimal and hexadecimal input formats. Hexadecimal input requires the "16#" prefix; otherwise, the system will automatically process the value as decimal.

    The working stroke per revolution of the worktable is 1.

    Based on the above parameters, if the target distance input on the relative motion instruction side is 100, the control axis will output an actual pulse count of 104857600, causing the motor to rotate 100 revolutions. Without a reducer, the worktable moves 100 user units.

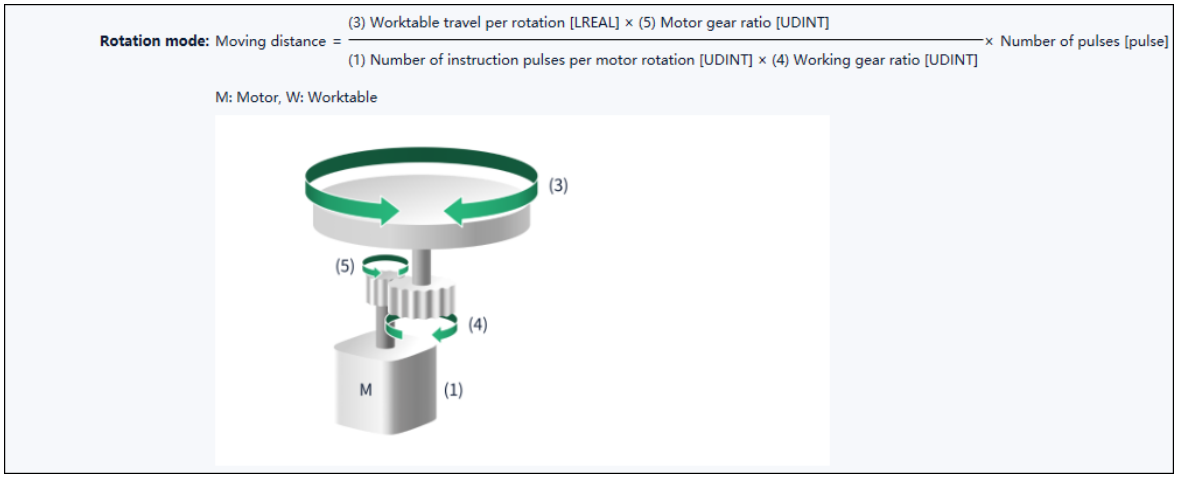
  - Gears used

    The typical scenario for linear mode is shown in the following figure:



    (1) is the motor, (3) is the workpiece, (4) is the work gear ratio, and (5) is the motor gear ratio.

    The typical scenario for rotary mode is shown in the following figure:



(1) is the motor, (3) is the workpiece, (4) is the work gear ratio, and (5) is the motor gear ratio.

16.1.7.4 Mode/Parameter Settings

1. Basic settings



   - Axis type: Supports rotary mode and linear mode. The linear mode is suitable for motor scenarios involving reciprocating motion, such as lead screws and linear motors. The rotational mode is suitable for motor scenarios involving unidirectional rotation where cumulative position tracking is not required.
   - Ramp velocity: Supports trapezoidal and S-type motion planning. Trapezoidal (T-type) motion planning does not involve jerk parameter. S-type motion planning requires input of jerk parameter.
   - Position hysteresis management: Select "Disable" to turn off the excessive position deviation alarm. Select "Effective" to periodically check whether the deviation exceeds the set hysteresis limit.

2. Period setting: Used in rotary mode to limit the rotation period.



3. Software limit: Applicable in linear mode. When enabled, it limits the positive and negative positions of the axis.



4. Software limit prediction parameter setting: Configures the axis limit protection by determining the risk between the current position and the limit position.



5. Motion parameter limit: The figure below shows the upper limits for single-axis motion parameters. If the parameters in the motion instruction block exceed these limits, the function block will report a fault.



6. Commissioning parameter limit: The figure below shows the limits for single-axis commissioning parameters. These limits only apply to MC_Jog and MC_Inch instructions, as well as the maximum speeds and acceleration/deceleration of the forward/reverse operations in the "Axis commissioning" function. If the set value exceeds the limit value, the system will operate at the limit value.

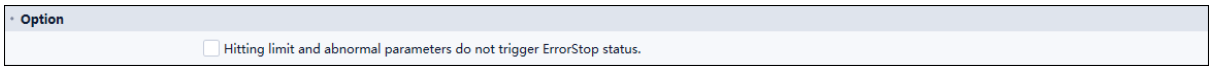


7. Monitor parameter setting



   - Positioning range: Used to set the positioning completion width, in reference units. The value is 0 or a real integer, with a default of 10.
   - Positioning check time: Used to set the positioning completion check time, in ms. The value range is 0 to 10000, with a default of 0. When set to 0, it indicates that the arrival check function is disabled.

8. Option: If enabled, the axis status will not switch to ERRORSTOP upon limit hit or abnormal parameters. Use this option with caution.



9. Probe setting: After enabling the probe function, select a digital input terminal to work with the MC_TouchProbe instruction for position latching. For detailed usage, see 18.3.3.3.10 MC_TouchProbe.

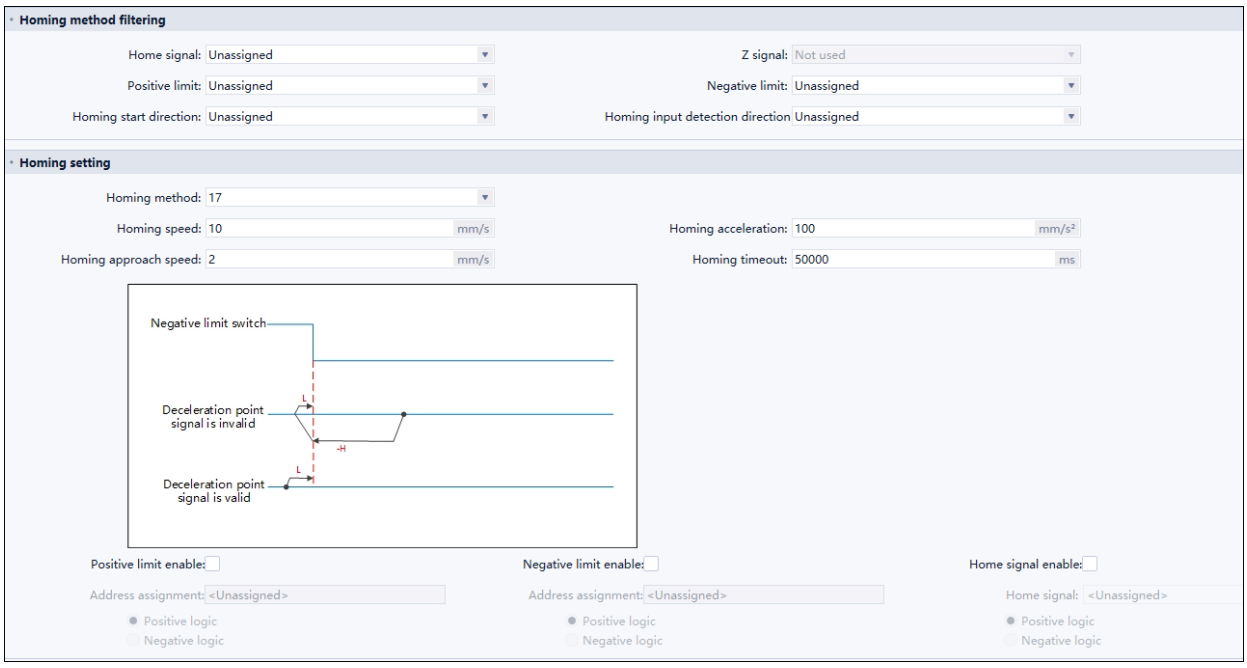


10. Output setting: Sets pulse output mode. The direction reversal delay time is ineffective.

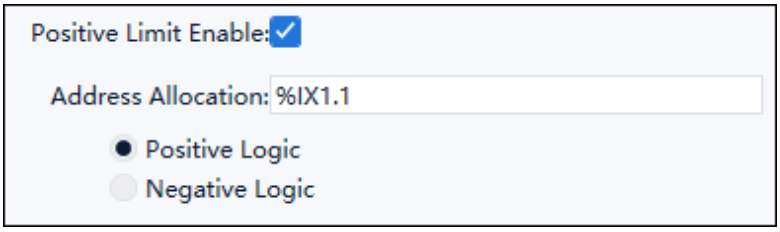


16.1.7.5 Homing Settings

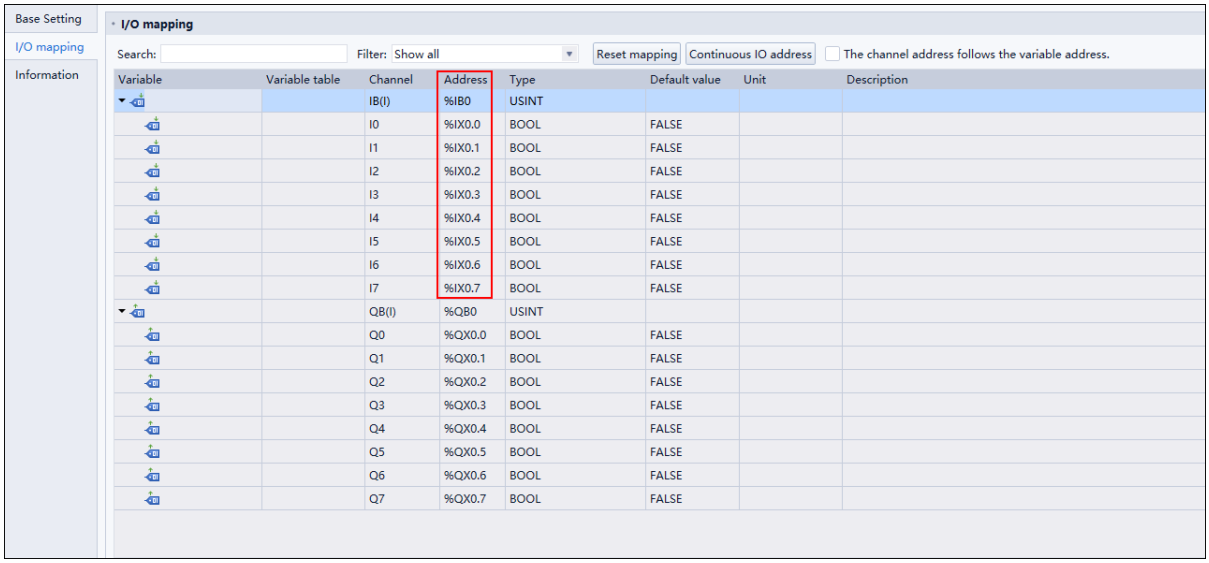
Homing settings are mainly used for graphical parameter configuration of homing for local pulse axes (as shown in the figure below). Follow the graphical configuration guidance to select the homing mode from the drop-down menu and complete the corresponding parameter configuration. The homing for local pulse axes adopts the method specified in the CiA402 protocol, and supports homing without Z-phase signal.



Since the input signals of local pulse axes are not assigned fixed functions, positive and negative limit signals and home signals need to be bound and enabled through configuration. The configuration is shown in the figure below.



Address allocation can be obtained through device I/O mapping in hardware configuration. As shown in the figure below, for local high-speed I/O, the mapped address of the input can be obtained according to the terminal to be connected.



Variable	Variable table	Channel	Address	Type	Default value	Unit	Description
		IB(I)	%IB0	USINT			
		I0	%IX0.0	BOOL	FALSE		
		I1	%IX0.1	BOOL	FALSE		
		I2	%IX0.2	BOOL	FALSE		
		I3	%IX0.3	BOOL	FALSE		
		I4	%IX0.4	BOOL	FALSE		
		I5	%IX0.5	BOOL	FALSE		
		I6	%IX0.6	BOOL	FALSE		
		I7	%IX0.7	BOOL	FALSE		
		QB(I)	%QB0	USINT			
		Q0	%QX0.0	BOOL	FALSE		
		Q1	%QX0.1	BOOL	FALSE		
		Q2	%QX0.2	BOOL	FALSE		
		Q3	%QX0.3	BOOL	FALSE		
		Q4	%QX0.4	BOOL	FALSE		
		Q5	%QX0.5	BOOL	FALSE		
		Q6	%QX0.6	BOOL	FALSE		
		Q7	%QX0.7	BOOL	FALSE		

NOTE

Controller-based homing is adopted for local pulse axes.

16.1.7.6 Commissioning

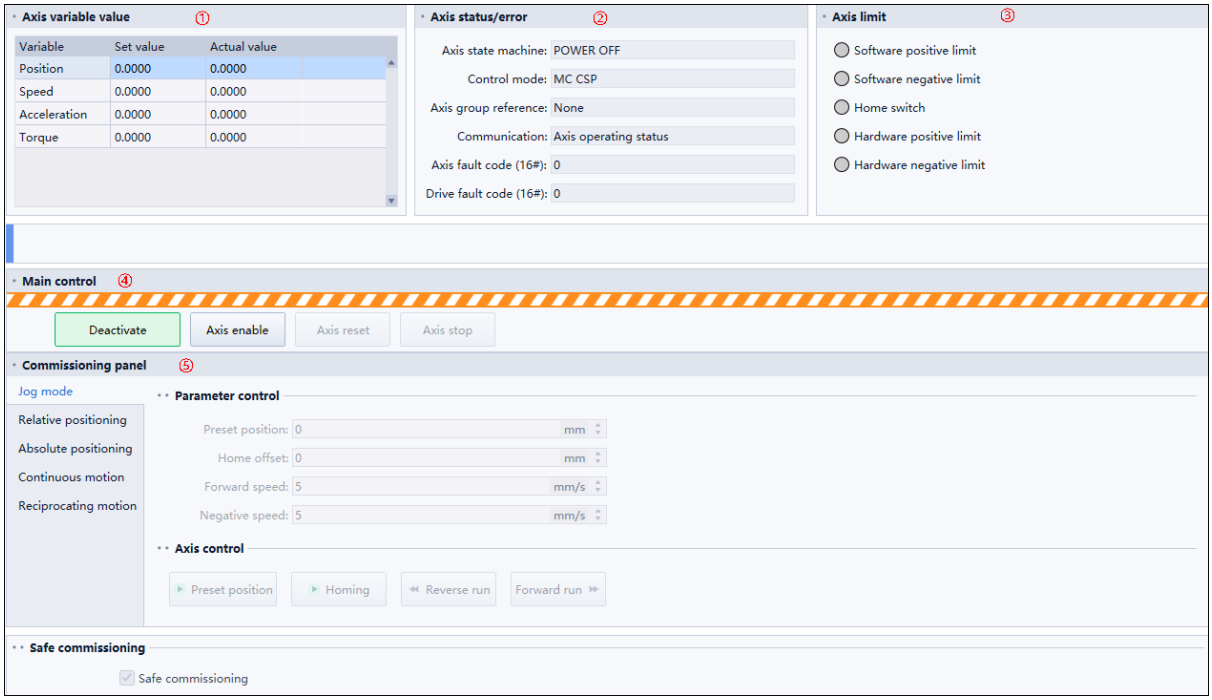
Basic functions such as enabling, positioning, reciprocation, homing, and stopping can be achieved through online commissioning. If the axis is connected to a bus stepper drive, the drive must support the relevant functions to perform homing or torque control. After confirming that basic motions are normal, complex motion control can be achieved using motion control function blocks.

NOTE

Axis commissioning and program motion control function blocks cannot be used simultaneously. Axis commissioning has the highest priority. After clicking "Activate commissioning", any execution of motion control function blocks will trigger a fault, and the motion control functions will be invalid. The functions can be used normally again after commissioning is disabled, and the function block is reset and triggered again.

Supported axis types: bus-controlled axes, local pulse axes, and virtual axes.

1. When the device is online and switched to the operational status, the "Activate commissioning" button can be clicked to activate axis commissioning. The interface will be as shown in the figure below.



No.	Description
①	Axis-end information including position, speed, acceleration, and so on.
②	Axis status and drive fault information
③	Axis-end limit terminal status
④	Buttons for activating/deactivating commissioning, enabling axis, resetting axis and stopping axis.
⑤	Jog or other motion mode options

2. Click the "Axis enable" button to enable the axis. You can then view the axis information including the current position, speed, status, and so on.
3. By setting different motion modes and motion parameters, the overall operational status of the servo can be viewed.
4. Safe commissioning: Enabled by default. When axis commissioning is activated, switching pages or closing the commissioning page will force the axis to stop.
   The axis must be stopped in the following cases:
   - The current commissioning page is switched.
   - The current commissioning page is closed.
   - The iFA software is closed.

   The axis will not be stopped in the following cases:
   - The axis commissioning page remains unchanged in multi-view mode.
   - The commissioning page is minimized.

5. Additional notes
   - The commissioning mode is used for axis commissioning. Motion control function blocks in this mode will report a fault.
   - In commissioning mode, the axis will be stopped automatically if the software is closed, the project is switched, or the PLC is re-downloaded and installed.
   - In commissioning mode, the oscilloscope can be used normally.
   - In commissioning mode, the I/O mapping data of all slave modules remain unaffected.
   - During single-axis commissioning, enabling servo commissioning will cause an enabling conflict. The servo commissioning must be disabled before using axis commissioning.
   - In commissioning mode, parameter anomalies and other faults will cause the axis to enter the ERRORSTOP status.

6. Common functions on this interface

Option	Note
Axis variable value	Supports monitoring of set and actual values of axis position, speed, acceleration, and torque.
Axis limit	Displays software positive limit, software negative limit, home switch, hardware positive limit, and hardware negative limit.
Activate commissioning	After the software is online, all parameters on the commissioning interface will be ineffective if this button is not clicked. Once clicked, the "Axis enable" button becomes operable. During axis operation, deactivating commissioning will switch the axis to the power-off status.
Enable axis	Once clicked, the axis status will be switched to StandStill; the jog mode parameters and other mode parameters on the commissioning interface become editable.
Axis reset	If a fault occurs after the axis is enabled, this button can be used to reset the axis.
Axis stop	This button is effective regardless of the motion mode of the axis.
Jog mode	◦ Sets the speed for forward and reverse jogging. The unit is converted according to the settings on the unit conversion interface. ◦ Supports preset value configuration. Input the preset value and click "Settings" to apply it. The unit is converted according to the settings on the unit conversion interface. ◦ Control the jogging direction of the motor by clicking "Jog backward"/"Job forward". ◦ The running parameter for jog mode serves as both the speed parameter and the acceleration parameter.

Option	Note
Other modes	Relative positioning: Parameters include target distance, speed, acceleration, deceleration, and jerk. Absolute positioning: Parameters include target distance, speed, acceleration, deceleration, and jerk. Continuous motion: Parameters include target speed, acceleration, deceleration, and jerk. Reciprocating motion: Parameters include two sets of target positions. Torque control: Parameters include target torque, torque ramp, and limited speed.

16.1.7.7 Axis Diagnostics

For details, see 16.1.5.6 Axis Diagnostics.

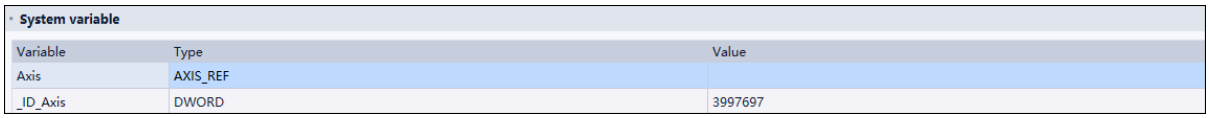
16.1.7.8 I/O Mapping

All task names and bus cycle settings will be displayed in the bus cycle options. For axis objects (including control axes and encoder axes), it is recommended to prioritize the EtherCATMaster_Task**.** Selecting other tasks may result in abnormal conditions such as planned speed jump.



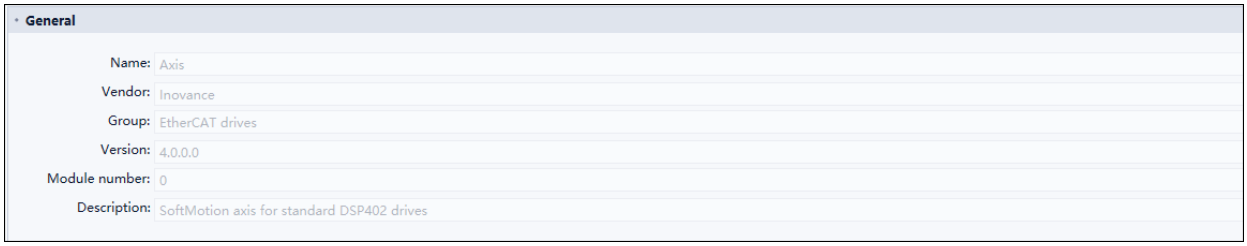
16.1.7.9 System Variables

The "System variable" tab displays the data structure and device ID of the axis.



16.1.7.10 Information

This interface displays basic information about the axis, including name, vendor, group, version, module number, and description.



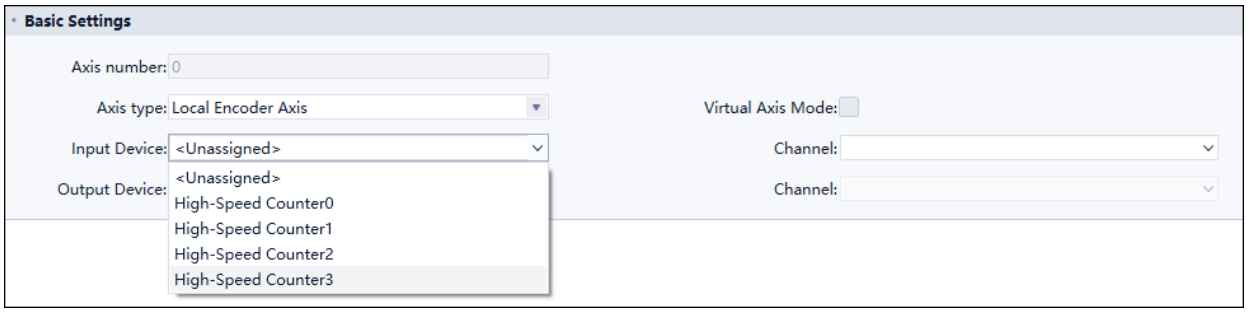
16.1.8 Local Encoder Axis

16.1.8.1 Overview

Axes associated with counters are referred to as counter axes. The system supports 4-axis 32-bit high-speed counters, which can realize phase A/B frequency multiplication by 1/2/4, CW/CCW, pulse + direction, and single-phase counting. Counting signal sources are external pulse input or internal 1 ms/1 μs clock counting, which can be used to preset and latch the counter in combination with other input signals.

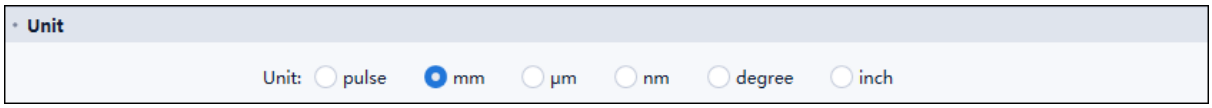
16.1.8.2 Basic Axis Settings

After selecting "Local encoder axis" as the axis type, the "Input device" option is activated. The input device can be selected as 4-channel high-speed counters. After selecting the high-speed counter, set "Signal source" to the I/O terminal corresponding to the A/B phase on the "Mode/Parameter setting" tab page.

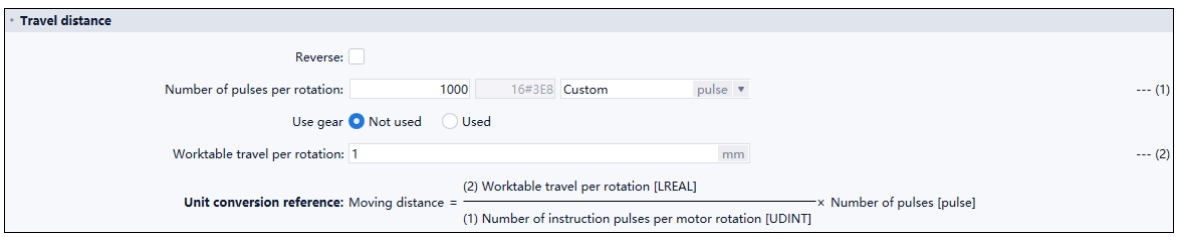


16.1.8.3 Unit Conversion

- Unit: Conversion between six units, including pulse, millimeter (default), micron, nanometer, degree, and inch. The unit can be configured as needed.



- Travel distance: Conversion between user unit and pulse unit.
  For bus-controlled axes or local pulse axes, the user unit (such as pulse, millimeter, or micron) is used on the instruction side, while the pulse unit is used for actual control of the physical motor. Gear ratio coefficient needs to be configured for unit conversion. The conversion modes are divided into the following scenarios:
  - Gears not used
    When gears are not used, the conversion formula from user unit to pulse unit is as follows:



    Taking the Inovance 20-bit encoder as an example, set the parameters as follows:

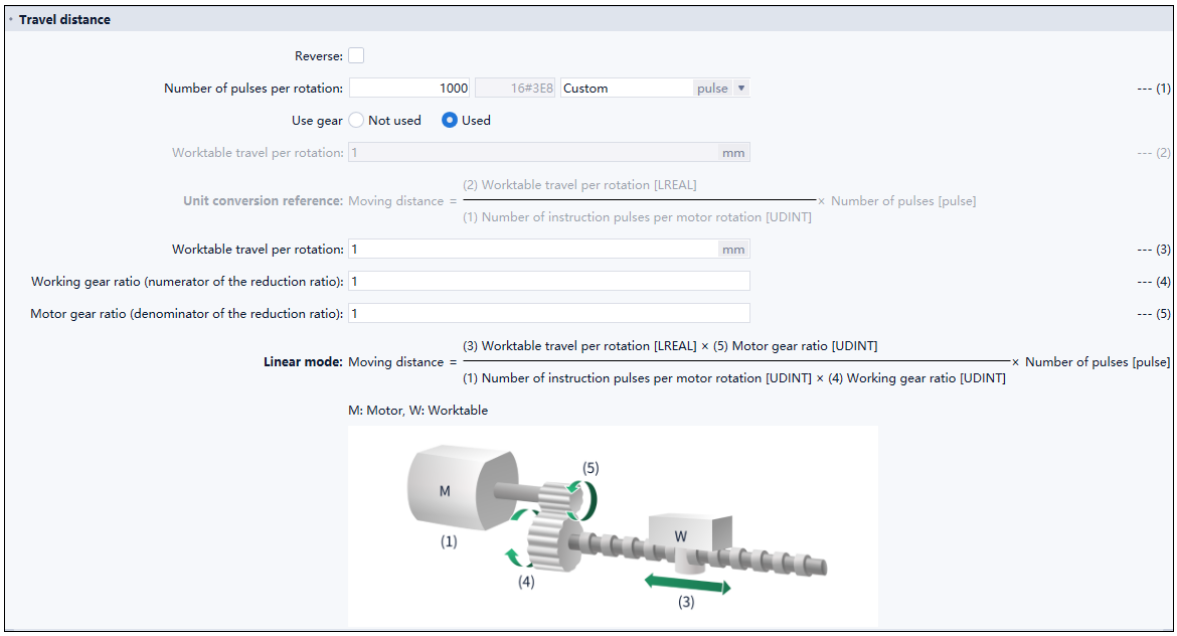
The number of pulses per motor revolution is default to 10#1048576. Number of bits for common encoders can be directly selected, which currently supports 14-bit, 18-bit, 20-bit, 23-bit, and 26-bit encoders. The number of pulses per motor revolution can also be customized, supporting both decimal and hexadecimal input formats. Hexadecimal input requires the "16#" prefix; otherwise, the system will automatically process the value as decimal.

The working stroke per revolution of the worktable is 1.

Based on the above parameters, if the target distance input on the relative motion instruction side is 100, the control axis will output an actual pulse count of 104857600, causing the motor to rotate 100 revolutions. Without a reducer, the worktable moves 100 user units.

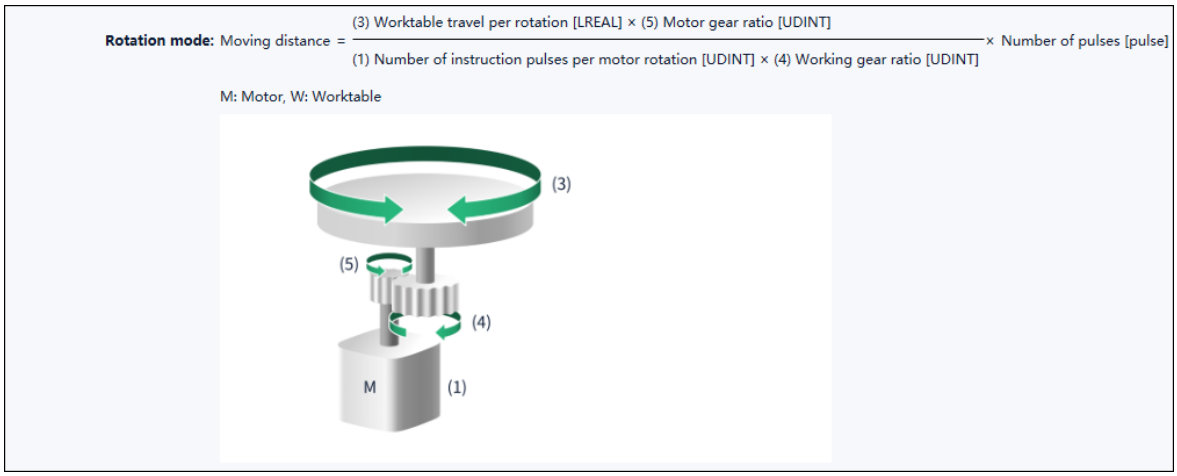
- Gears used

  The typical scenario for linear mode is shown in the following figure:



(1) is the motor, (3) is the workpiece, (4) is the work gear ratio, and (5) is the motor gear ratio.

The typical scenario for rotary mode is shown in the following figure:



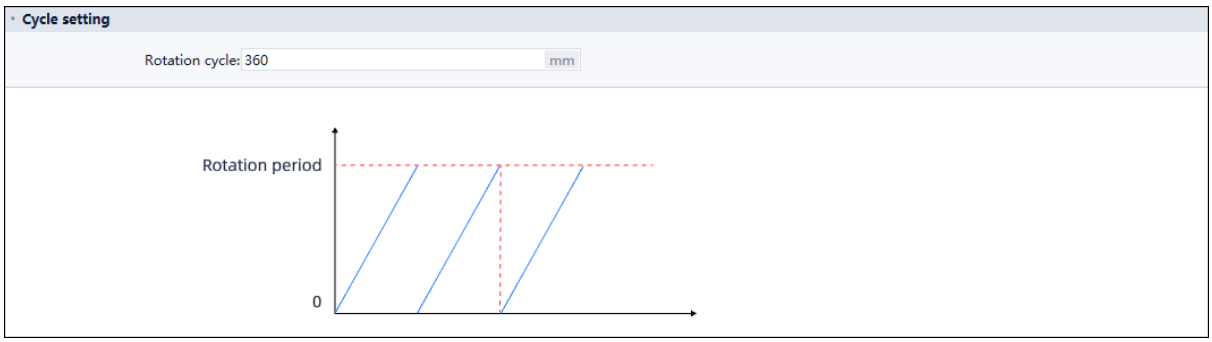
(1) is the motor, (3) is the workpiece, (4) is the work gear ratio, and (5) is the motor gear ratio.

16.1.8.4 Mode/Parameter Settings

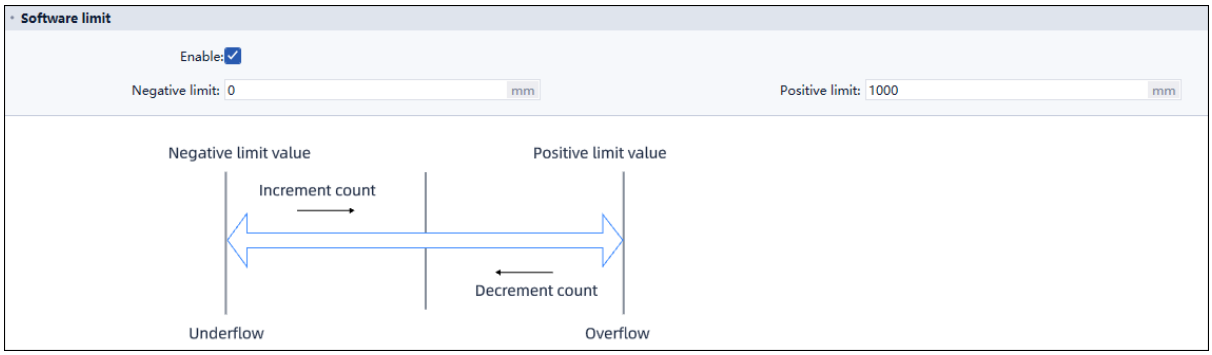
1. Mode setting



2. Period setting: Used in rotary mode to limit the rotation period. The counter counts reciprocally between 0 and the rotation period.



3. Software limit: Used to set whether to enable software limits in linear mode. After enabling the software limit, the positive limit value and negative limit value can be set. When the counter reaches the limit value, counting stops, and the PostiveLimit or NegativeLimit flag in the ENC_Counter instruction is set. When software limit is disabled, the module's count value cycles within the range of -2147483648 to 2147483647. When forward counting reaches the maximum positive value, it directly jumps to the maximum negative value and continues counting. The reverse is the same.



4. Counter mode setting



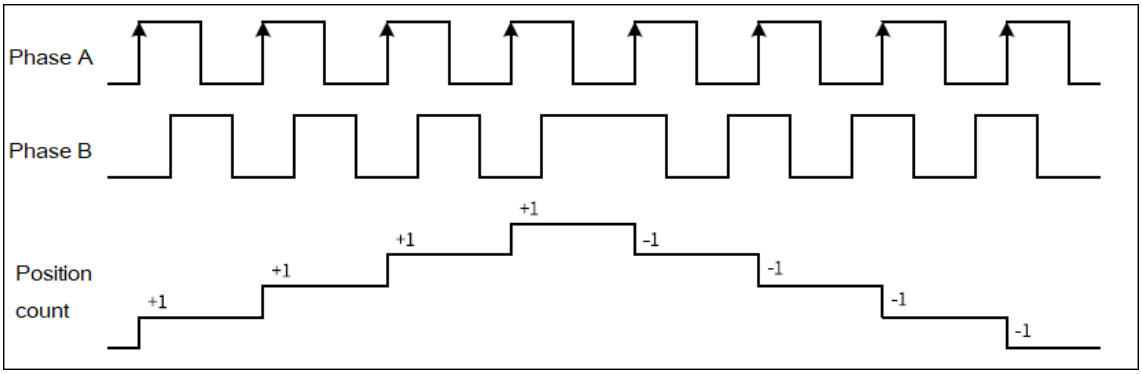
   - Counting mode

     Six count modes are supported: phase A/B 1-frequency multiplication, phase A/B 2-frequency multiplication, phase A/B 4-frequency multiplication, pulse+direction, CW/CCW, and single-phase counting.

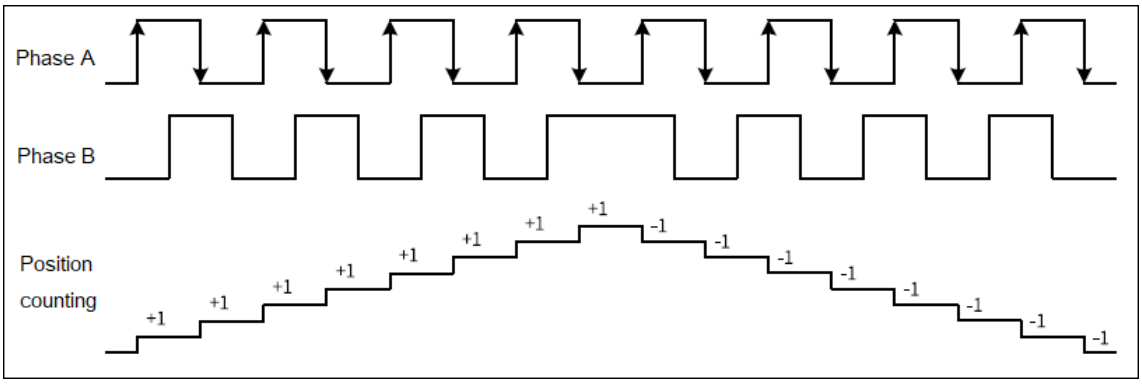
     - Phase A/B pulse

       In phase A/B mode, the encoder generates two orthogonal phase pulse signals with a phase difference of 90°, that is, phase A signal and phase B signal. When the phase A signal leads the phase B signal, the counter counts up; when the phase B signal leads the phase A signal, the counter counts down. The phase A/B pulse can be set to operate in frequency multiplication by 1, 2, or 4.

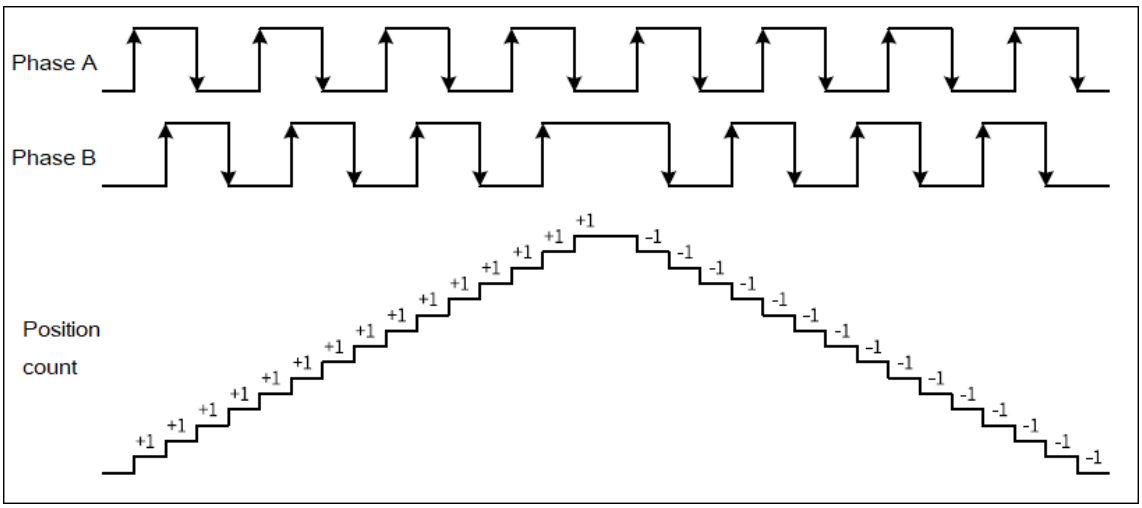
       In the mode of phase A/B frequency multiplication by 1, only the rising edge of phase A pulse is counted as shown in the following figure.



In the mode of phase A/B frequency multiplication by 2, the rising and falling edges of phase A pulse are counted as shown in the following figure.

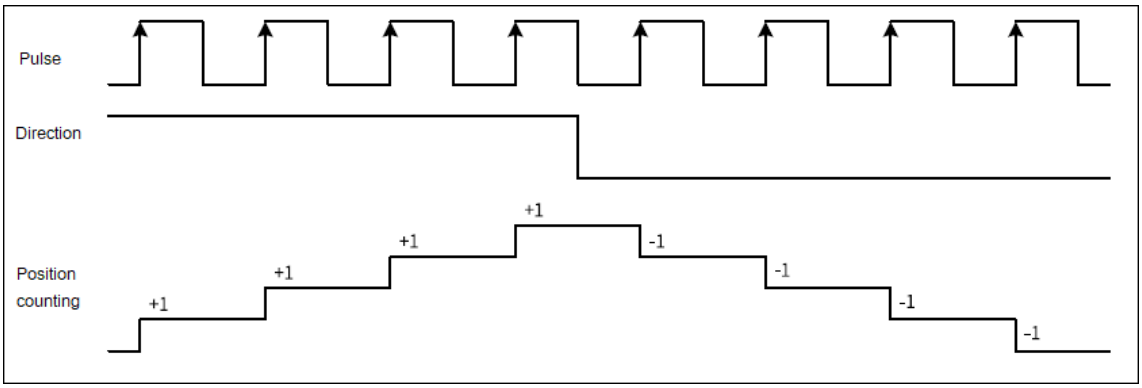


When set to A/B phase 4-frequency multiplication, counting is performed on the rising and falling edges of both the A-phase and B-phase pulses, as shown in the figure below.



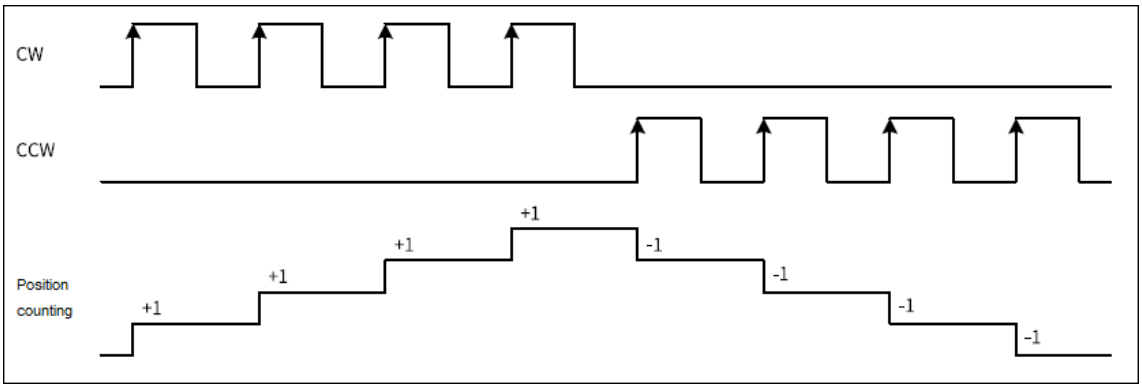
- Pulse + direction

  Phase A serves as the pulse input for the counter, and phase B serves as the direction control input for the counter. When the B-phase signal is at high level, the counter increments; when it is at low level, the counter decrements.



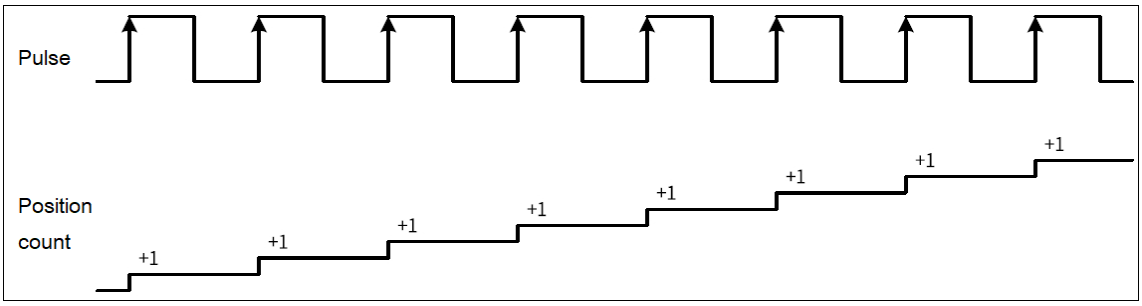
- CW/CCW mode

  Clock Wise (CW) is the forward pulse signal, and Counter Clock Wise (CCW) is the reverse pulse signal. When the encoder is forward running, CW pulse signals are output; when the encoder is reverse running, CCW pulse signals are output.



- Single-phase counting

  In this mode, the high-speed counter counts up the pulse signals. The position count is added with 1 when the rising edge of a pulse is input.



◦ Signal source

Different signal sources can be selected depending on the counting mode. The supported input/output terminals of signal sources in various counting modes are shown in the following table.

Table16-1-8-4-1 Signal source and counting mode

Counting Mode	X0	X1	X2	X3	X4	X5	X6	X7	Internal 1 ms	Internal 1 μs
Phase A/B	Phase A	Phase B	Phase A	Phase B	Phase A	Phase B	Phase A	Phase B	x	x
CW/CCW	CW	CCW	CW	CCW	CW	CCW	CW	CCW	x	x
Pulse + direction	Pulse	Direction	Pulse	Direction	Pulse	Direction	Pulse	Direction	x	x
Single-phase counting	Pulse	Pulse	Pulse	Pulse	Pulse	Pulse	Pulse	Pulse	Pulse	Pulse

5. Reset setting



When the reset function is configured, the counter value will be cleared upon a valid input signal at the digital input terminal.

6. Probe setting



For local encoder axes, each axis supports one channel of probe signal. It must be used in conjunction with the ENC_TouchProbe instruction for position latching.

7. Preset setting



When a terminal is allocated with the preset function, you need to call the instruction ENC_Preset to implement the counter preset function. When the input signal of the DI terminal is active, the value of the counter will be set to the preset value of parameter Position of the instruction ENC_Preset.

8. Compare output setting



The comparison output function provides high real-time performance with output response reaching the millisecond level. Output terminals Y0 to Y3 all support the comparison output function. Each encoder axis can be configured with one channel of comparison output function. The comparison output function can be used in conjunction with the ENC_Compare, ENC_ArrayCompare, or ENC_StepCompare instructions. The duration of the output terminal can be set in either pulse units or time units. When the time unit is set to ms, the configurable range is 0.1 to 6553.5 ms. When the time unit is set to Unit, the configurable range is 1 to 65535 Unit.

16.1.8.5 Axis Diagnostics

For details, see 16.1.5.6 Axis Diagnostics.

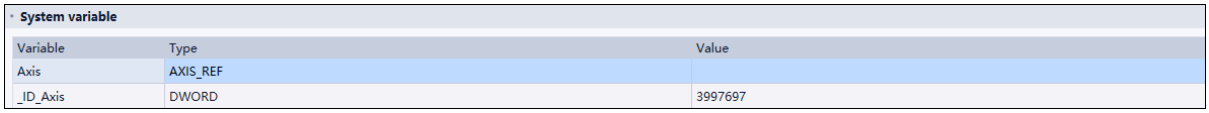
16.1.8.6 I/O Mapping

All task names and bus cycle settings will be displayed in the bus cycle options. For axis objects (including control axes and encoder axes), it is recommended to prioritize the EtherCATMaster_Task**.** Selecting other tasks may result in abnormal conditions such as planned speed jump.



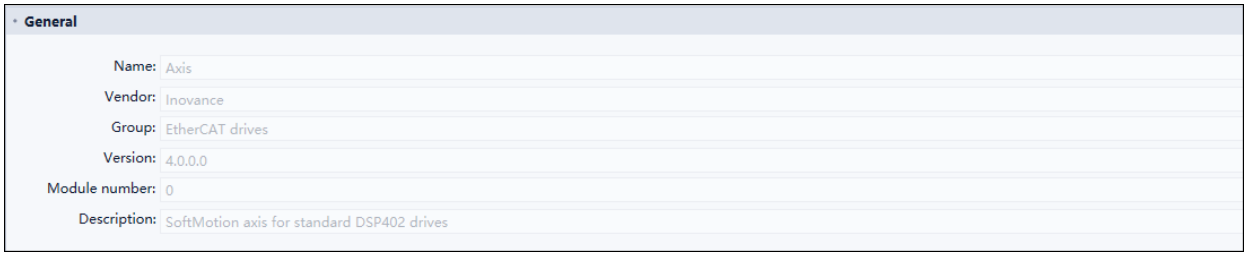
16.1.8.7 System Variables

The "System variable" tab displays the data structure and device ID of the axis.



16.1.8.8 Information

This interface displays basic information about the axis, including name, vendor, group, version, module number, and description.



16.2 Axis Group Configuration

16.2.1 Adding an Axis Group

Prerequisites

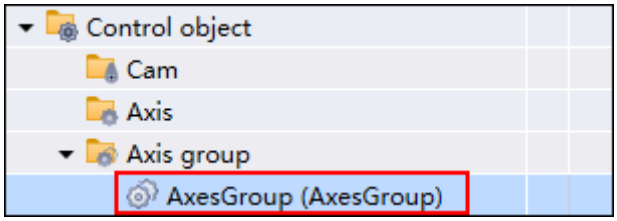
- Hardware configuration has been set up.
- Corresponding drives and axes have been added.

Procedure

1. Click "Control object" in the left project tree to expand the control object list.
2. Right-click "Axis group" and select "Add axis group". The "Add axis group" dialog box appears.
3. Enter the name of the axis group and click "OK".

Result

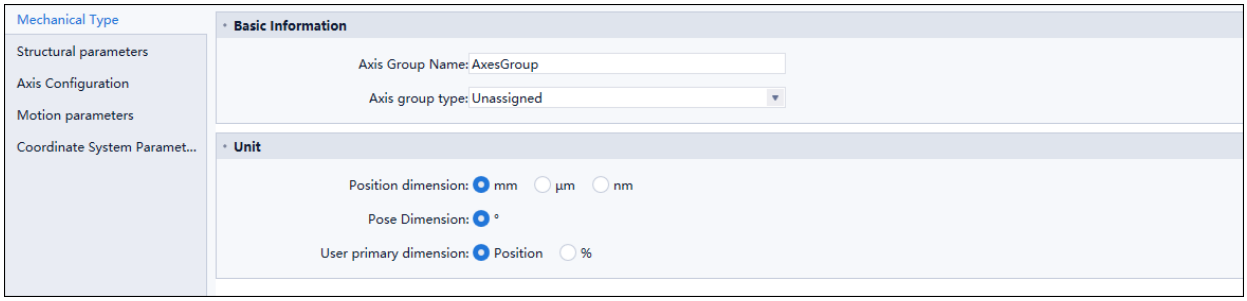
The newly created axis group can be seen under the axis group directory.



16.2.2 Axis Group Parameter Settings

16.2.2.1 Mechanical Types

Double-click the axis group added in the left project tree to open the configuration interface.



Axis group type description

Axis Group Type	Motion Position Parameter				Coordinate System Type (CoordSystem)	
	Absolute Position (ACS/PCS)	Relative Position (ACS/PCS)	Auxiliary Point (PCS)	End Point (PCS)	ACS	PCS
Planar XY	Position/ Velocity/ Acceleration[1] Position/ Velocity/ Acceleration[2]	Distance[1] Distance[2]	AuxPoint[1] AuxPoint[2]	EndPoint[1] EndPoint[2]	A1 A2	X Y
Cartesian XYZ	Position/ Velocity/ Acceleration[1] Position/ Velocity/ Acceleration[2] Position/ Velocity/ Acceleration[3]	Distance[1] Distance[2] Distance[3]	AuxPoint[1] AuxPoint[2] AuxPoint[3]	EndPoint[1] EndPoint[2] EndPoint[3]	A1 A2 A3	X Y Z

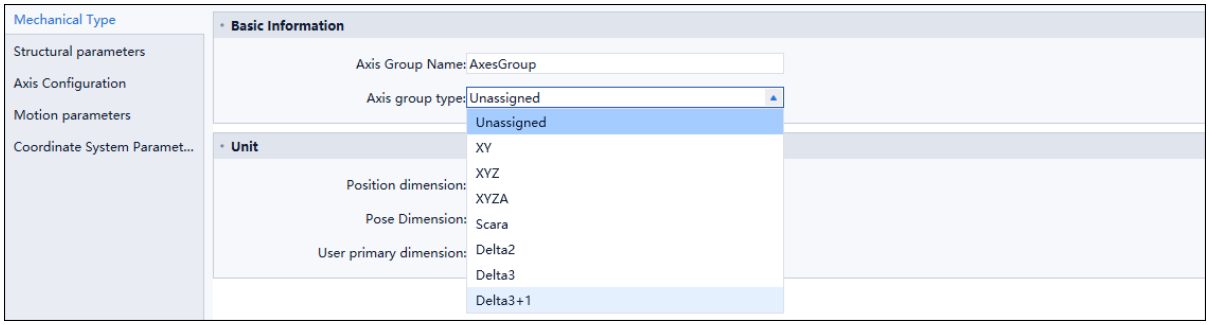
Cartesian XYZA	Position/ Velocity/ Acceleration[1] Position/ Velocity/ Acceleration[2] Position/ Velocity/ Acceleration[3] Position/ Velocity/ Acceleration[4]	Distance[1] Distance[2] Distance[3] Distance[4]	AuxPoint[1] AuxPoint[2] AuxPoint[3] AuxPoint[4]	EndPoint[1] EndPoint[2] EndPoint[3] EndPoint[4]	A1 A2 A3 A4	X Y Z A
SCARA	Position/ Velocity/ Acceleration[1] Position/ Velocity/ Acceleration[2] Position/ Velocity/ Acceleration[3] Position/ Velocity/ Acceleration[4]	Distance[1] Distance[2] Distance[3] Distance[4]	AuxPoint[1] AuxPoint[2] AuxPoint[3] AuxPoint[4]	EndPoint[1] EndPoint[2] EndPoint[3] EndPoint[4]	A1 A2 A3 A4	X Y Z A
Delta2	Position/ Velocity/ Acceleration[1] Position/ Velocity/ Acceleration[2]	Distance[1] Distance[2]	AuxPoint[1] AuxPoint[2]	EndPoint[1] EndPoint[2]	A1 A2	X Y

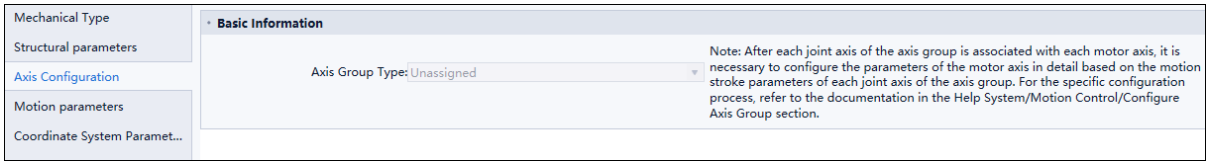
Delta3	Position/ Velocity/ Acceleration[1] Position/ Velocity/ Acceleration[2] Position/ Velocity/ Acceleration[3]	Distance[1] Distance[2] Distance[3]	AuxPoint[1] AuxPoint[2] AuxPoint[3]	EndPoint[1] EndPoint[2] EndPoint[3]	A1 A2 A3	X Y Z
Delta3+1	Position/ Velocity/ Acceleration[1] Position/ Velocity/ Acceleration[2] Position/ Velocity/ Acceleration[3] Position/ Velocity/ Acceleration[4]	Distance[1] Distance[2] Distance[3] Distance[4]	AuxPoint[1] AuxPoint[2] AuxPoint[3] AuxPoint[4]	EndPoint[1] EndPoint[2] EndPoint[3] EndPoint[4]	A1 A2 A3 A4	X Y Z A

NOTE

- The A-axis motion parameter values set for AuxPoint[4] do not participate in pose planning.
- The motion parameters of each joint axis set in the ACS space must be within the corresponding software limit range.
- The A-axis motion parameters set in the PCS space must be within ±180°.
- The HIC configuration is user-defined that supports axis groups consisting of 2 to 6 axes. It is recommended to use it with the "CmpHCInterpolator" library. Users can also call this axis group to control the motion of individual axes within the group based on the forward and inverse kinematics of the user-defined axis group.

Note: The "Axis group type" must be set; otherwise, the axes in the axis group cannot be configured. If the "Axis group type" is SCARA, Delta2, Delta3, or Delta3+1, the MC_SetPosition or MC_Home instruction must be used before executing the MC_GroupEnable instruction to set the position of each joint axis within the selected configuration range.

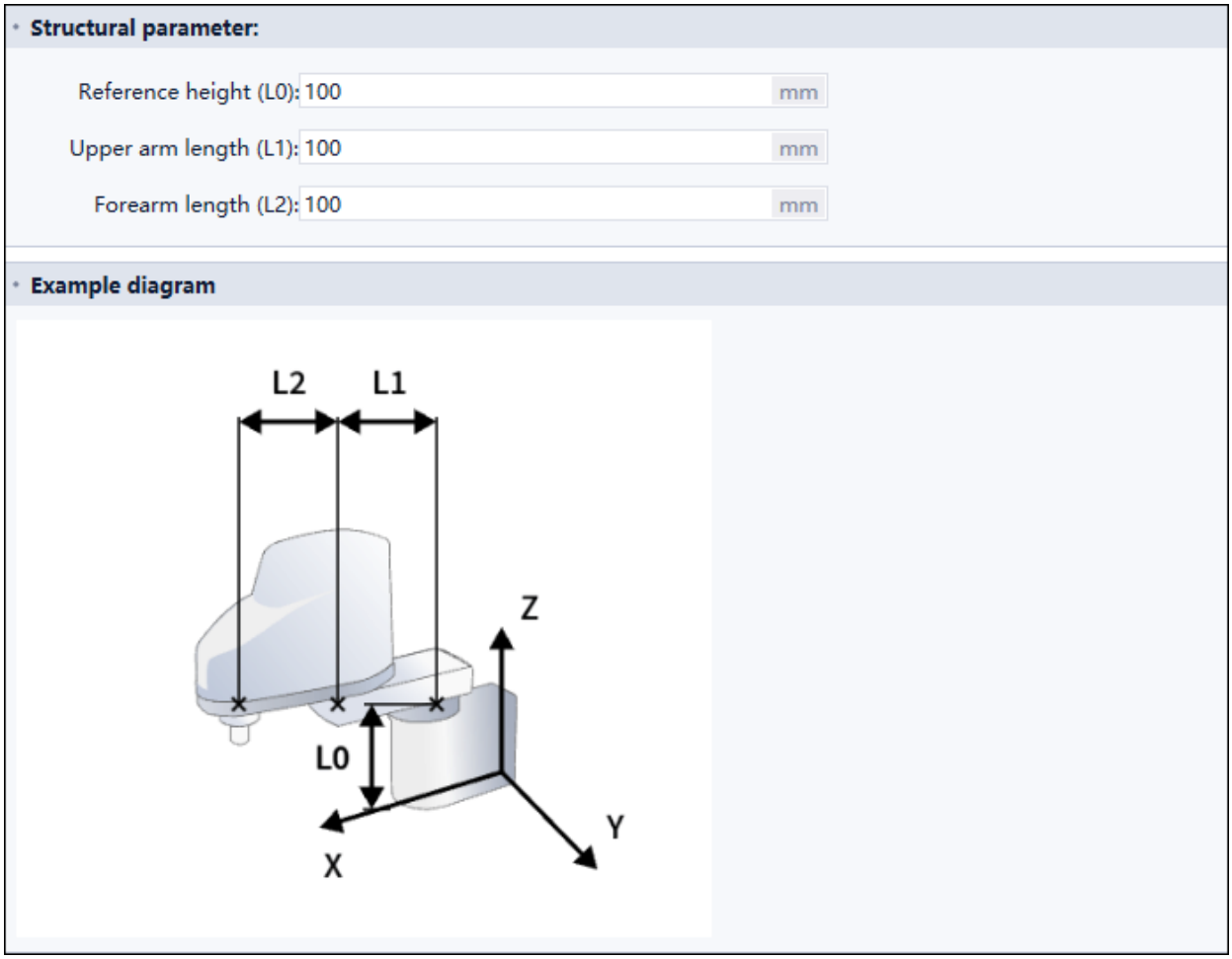




Option	Note
Axis group name	The set axis group name is displayed in the project tree.
Axis group type	Includes Planar XY, Cartesian XYZ, Cartesian XYZA, SCARA, Delta2, Delta3, and Delta3+1.
Unit	• Position dimension: Units include millimeters (default), micrometers, and nanometers. • Pose dimension: Unit is degrees. • User primary dimension: Units include position (default) and percentage. When set to "Position", the parameters (such as speed and acceleration) in the axis group motion instructions are input in the unit of position/ms. When set to "%", the parameters in the motion instructions are input as the product of the maximum parameter value set in the axis group and the input percentage.

16.2.2.2 Structural Parameters

- When the axis group type is set to SCARA, the following structural parameters shall be configured: reference height (L0), active arm length (L1), and driven arm length (L2).



- When the axis group type is set to Delta2, the following structural parameters shall be configured: static platform radius (R), moving platform radius (r), active arm length (L1), and driven arm length (L2). The structural parameters of Delta2 must satisfy the following constraints:
  - Structural parameters must be greater than 0.

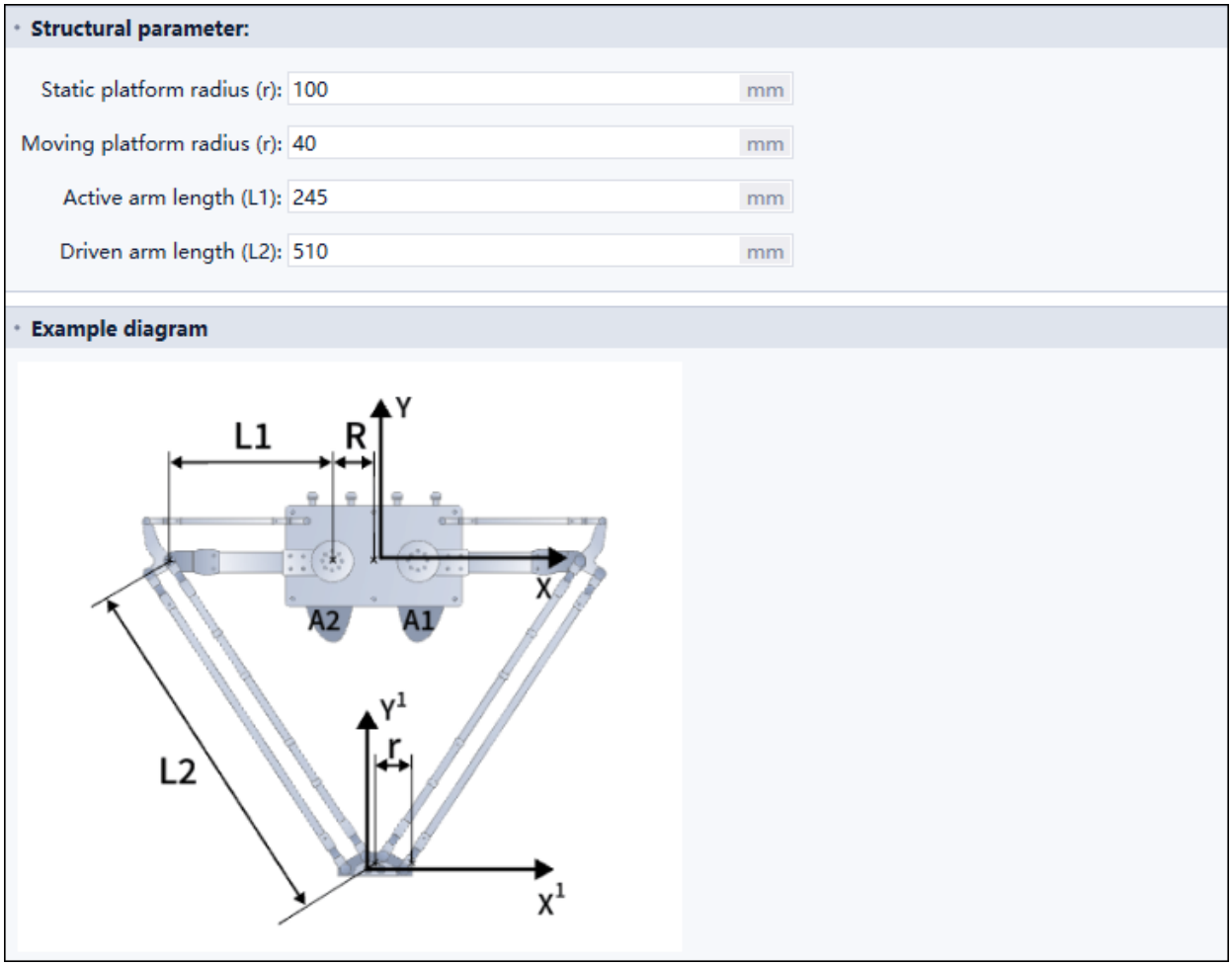
$$R>0, r>0, L_1>0, L_2>0$$

  - The static platform radius must be greater than the moving platform radius.

$$R>r$$

  - The difference between the driven arm length and the active arm length must be greater than the difference between the static platform radius and the moving platform radius.

$$L_2>(L_1+R-r)$$



- When the axis group type is set to Delta3, the following structural parameters shall be configured: static platform radius (R), moving platform radius (r), active arm length (L1), and driven arm length (L2). The structural parameters of Delta3 must satisfy the following constraints:
  - Structural parameters must be greater than 0.

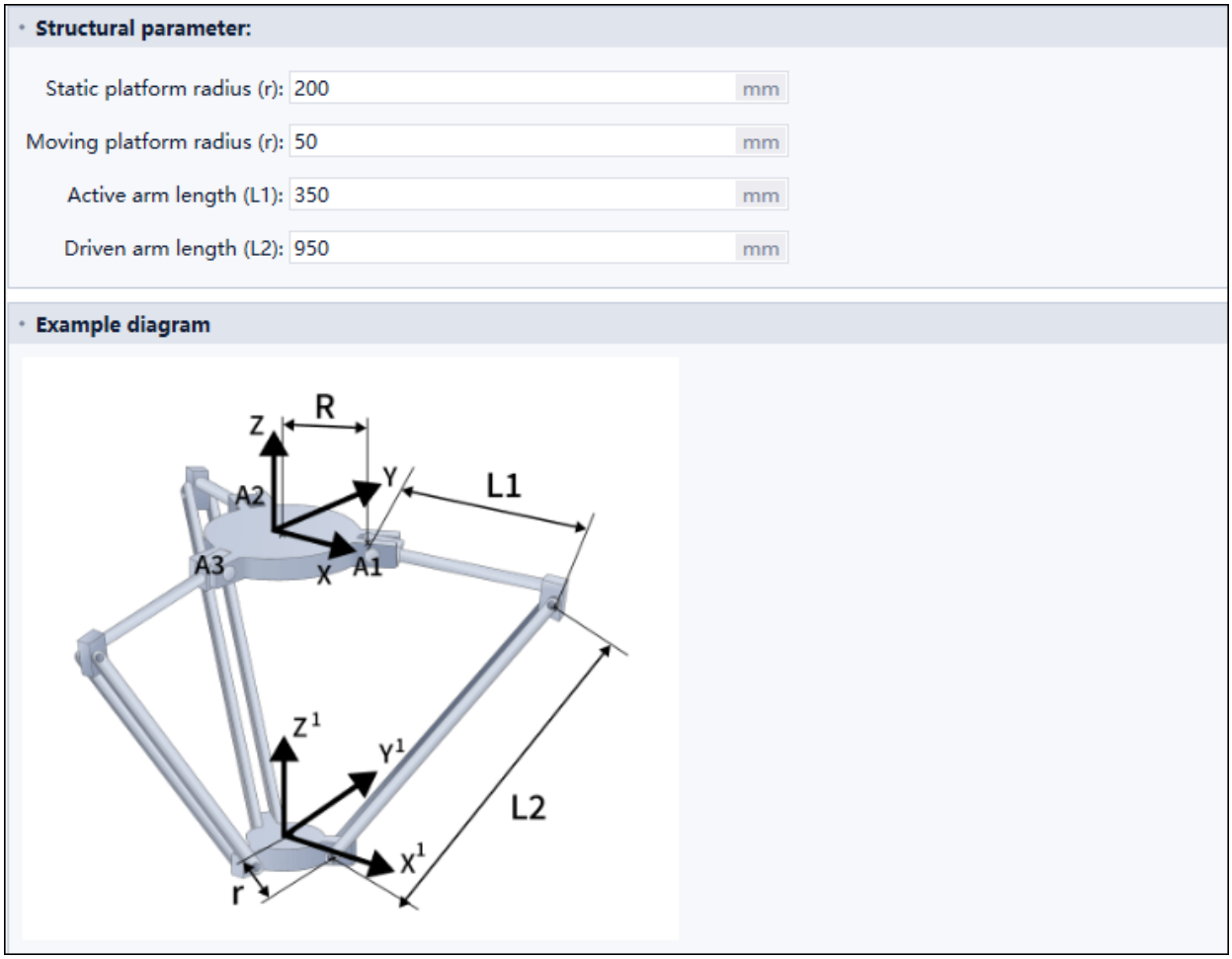
$$R>0, r>0, L_1>0, L_2>0$$

  - The static platform radius must be greater than the moving platform radius.

$$R>r$$

  - The difference between the driven arm length and the active arm length must be greater than the difference between the static platform radius and the moving platform radius.

$$L_2>(L_1+R-r)$$



- When the axis group type is set to Delta3+1, the following structural parameters shall be configured: static platform radius (R), moving platform radius (r), active arm length (L1), and driven arm length (L2). The structural parameters of Delta3+1 must satisfy the following constraints:
  - Structural parameters must be greater than 0.

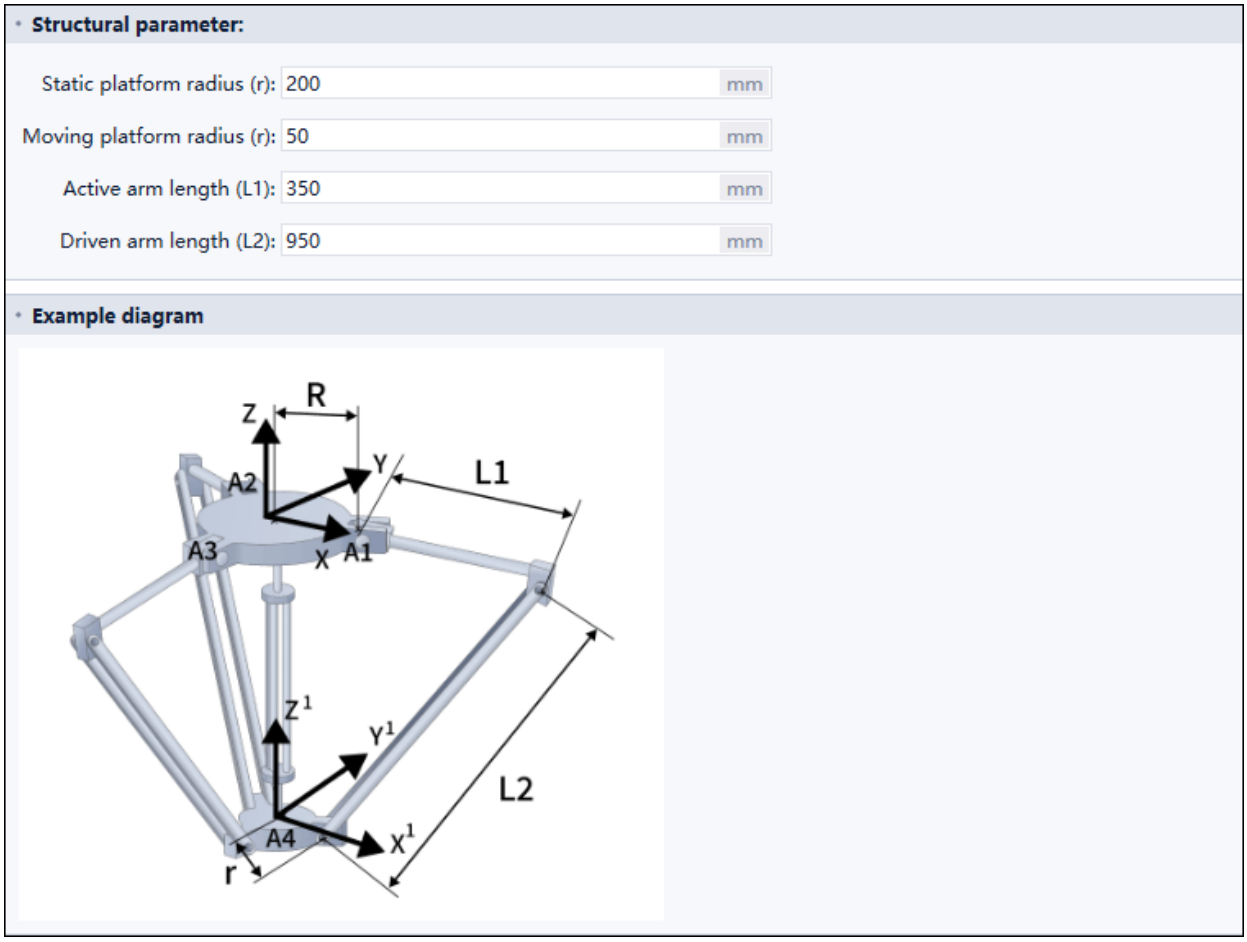
$$R>0, r>0, L_1>0, L_2>0$$

  - The static platform radius must be greater than the moving platform radius.

$$R>r$$

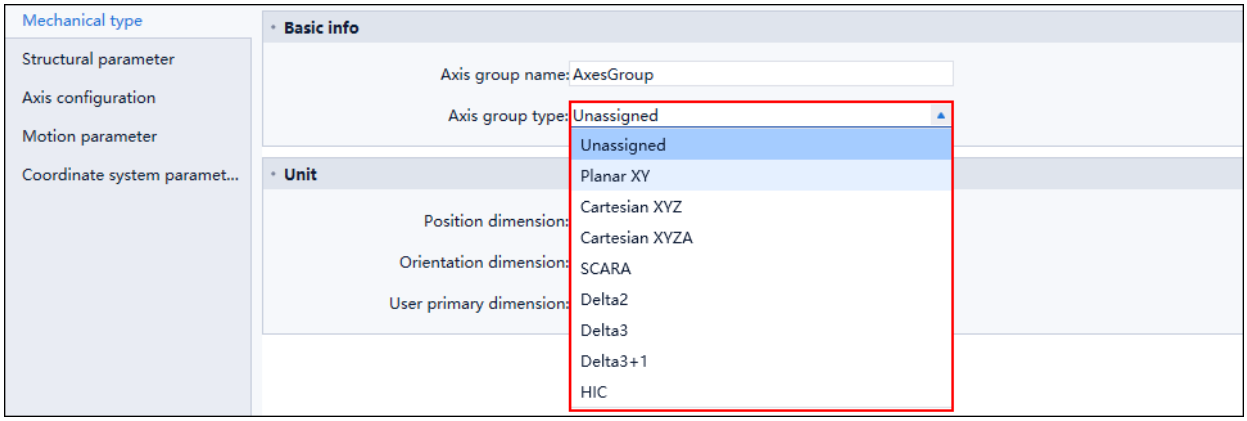
  - The difference between the driven arm length and the active arm length must be greater than the difference between the static platform radius and the moving platform radius.

$$L_2>(L_1+R-r)$$

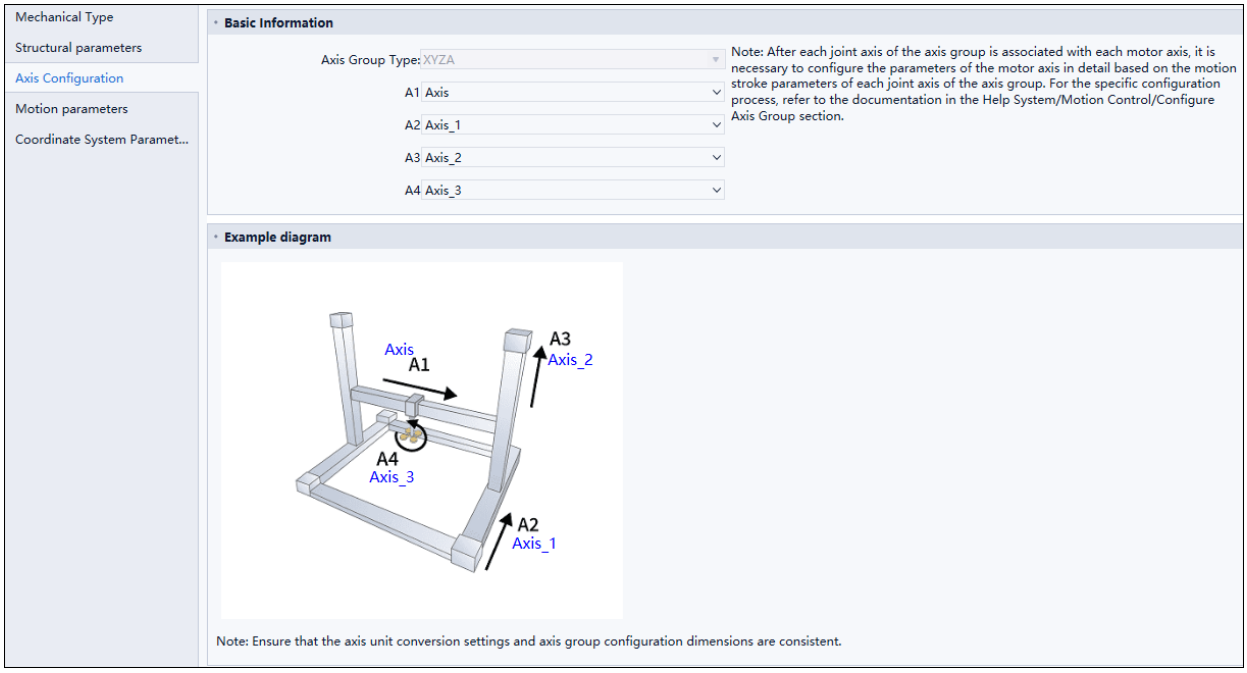


16.2.2.3 Axis Configuration

Before configuring the axes on the "Axis configuration" tab page, select the appropriate axis group type on the "Mechanical type" tab page.

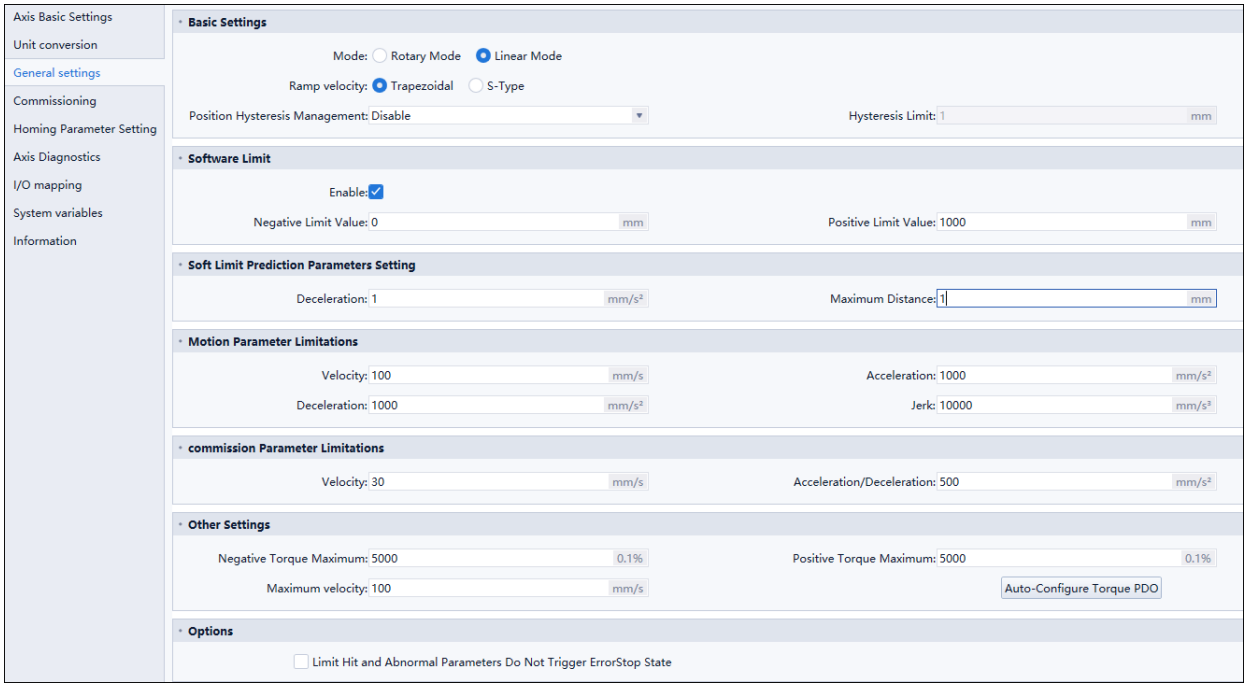


After selecting the axis group type, configure the axes within the axis group on the "Axis configuration" tab page. The axis group is associated with individual axes.



NOTE

- To limit the operating conditions of individual axes during axis group operation (such as motion parameter limits and software limits), configure the corresponding settings in the "General setting" interface of the respective axis as needed.
- If the axis group type is SCARA, Delta2, Delta3, or Delta3+1, you need to determine its motion range within the workspace and the motion range of each joint based on the structural parameters of the axis group. Set the software limits of individual axes based on the joint motion range to prevent the joint angle from exceeding the reachable limit during axis group motion; otherwise, the axis group motion will report a fault.
- The speed ramp mode, software limit prediction parameter settings, and commissioning parameter limits of individual axes do not take effect in the axis group.

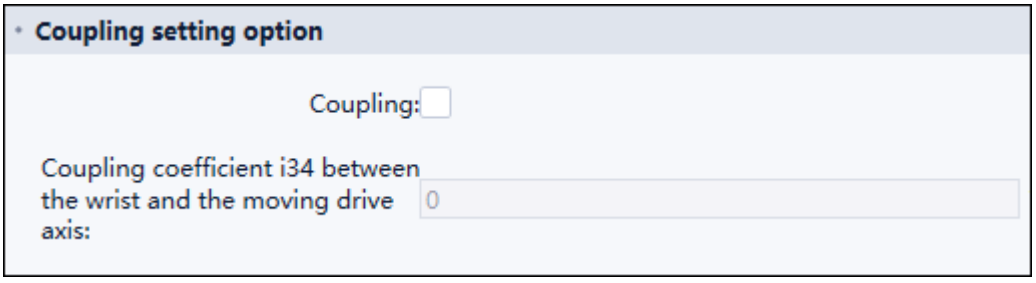


- Axes added in the left project tree must be associated consistently when configuring the axis group type.
  - The axis group type cannot be modified, and it must match the type selected in the "Mechanical type" tab page.
  - Configuration of bus-controlled axes, local pulse axes, and virtual axes is supported.
  - Only the linear mode is supported.
  - The example diagram shows the specific motion position and positive motion direction of each axis.

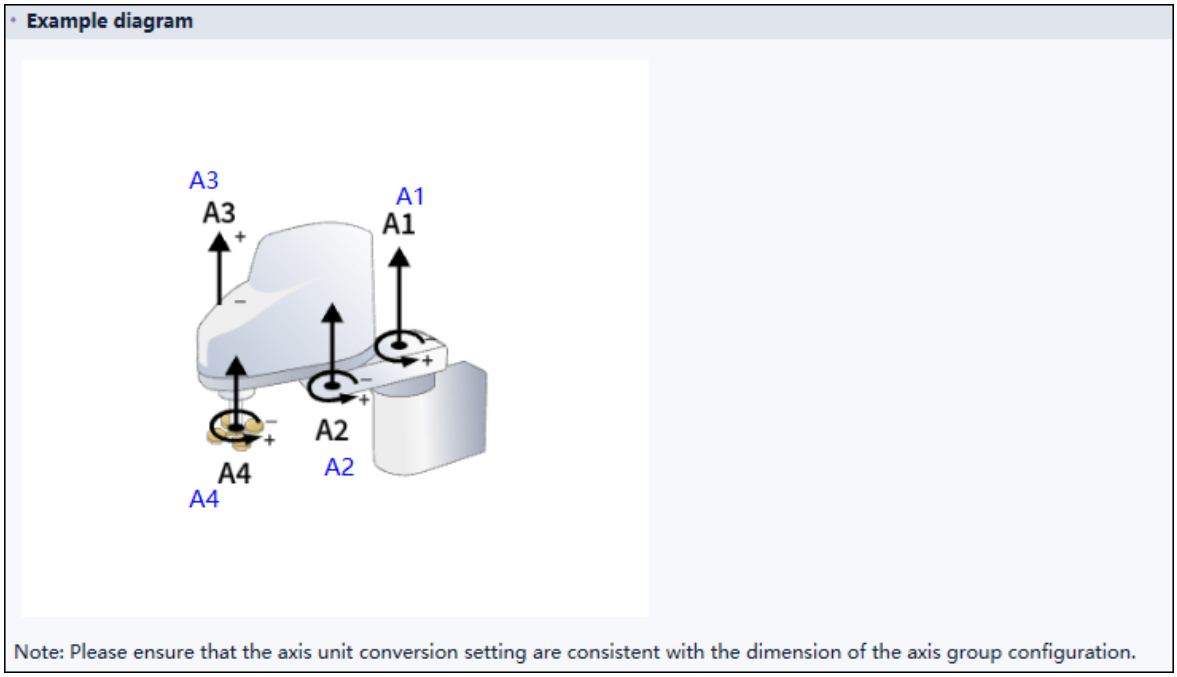
CAUTION

The motion units and transmission relationships of each joint must be correctly set in the interface of the bound axis.

- If the axis group type is set to the **Scara** mechanical model, the axis configuration interface will display the coupling setting options, as shown in the figure below.



  - The parameter will only take effect after checking "Coupling".
    Wrist and mobile drive shaft coupling coefficient i34: The rotation amount of A4 is added to A3 with a proportional coefficient. The unit used here is radians.
  - The example diagram of the SCARA model is as follows:



The meanings of A1 to A4 are as follows:

A1: Main arm drive axis.

A2: Forearm drive axis.

A3: Moving drive axis.

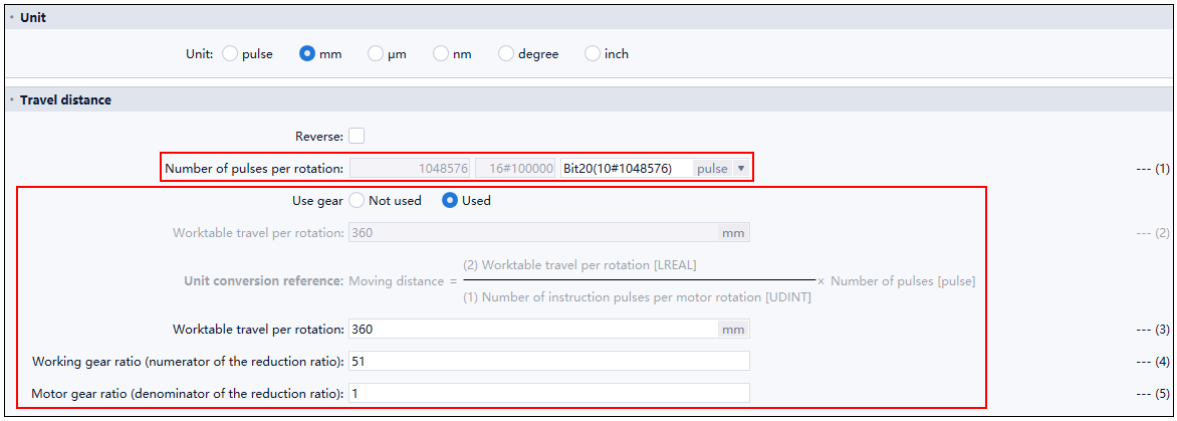
A4: Wrist drive axis.

- The position dimensions of **each joint are automatically synchronized with the corresponding settings of the axis.**
  If joints 1, 2, and 4 are rotary joints, set the axis unit to degrees, and settings shall be based on the reducer ratio and motor encoder bit count of each joint.

  If joint 3 is a linear joint, set the axis unit to mm, and settings shall be based on the mechanical transmission ratio of joint 3 and the pitch of the lead screw.

  Set parameters such as work gear ratio and motor gear ratio in the "Unit conversion" interface of each joint's bound axis.

  For a SCARA mechanical arm with the gear ratio parameter "dRatio":[51,51,2,12,0,0,0,0], if the joint motor encoder has 20 bits, set the **number of pulses per motor revolution** to 16#100000, i.e., 2^{20}.



Based on the gear ratio parameters, on the "**Unit conversion**" interface for Axis_1 corresponding to joint 2, select "Used" for the gear, set the "Worktable travel per revolution", and set the "Working gear ratio" to 51 and the "Motor gear ratio" to 1 according to the gear ratio parameters.

The settings for other joint axes are similar. Configure them individually based on the gear ratio parameters of each joint.

If joints 3 and 4 are coupled, the rotary motion of joint 4 causes displacement in joint 3, which can be offset by setting the **"Coupling coefficient i34 between the wrist and the moving drive axis"**.

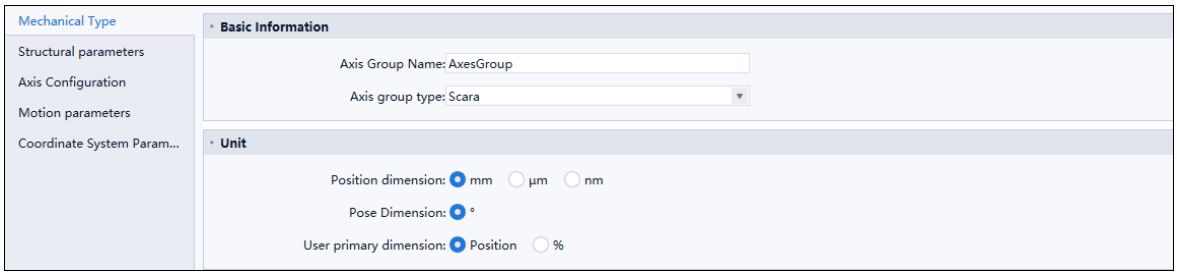
Calculation example of the "Coupling coefficient i34 between the wrist and the moving drive axis":

① Determine the relationship between the rotation angle and displacement: Manually rotate joint 4 by one revolution (2π, in radians) and measure the displacement of joint 3. Typically, it is the ratio of N-pitch displacement to one revolution radians, i.e., N × pitch / (2π).

② Determine the sign of the coupling coefficient: Manually rotate the rotary joint 4 and observe the motion direction of joint 3. If the direction of joint 3 is the same as the rotation direction of joint 4, the coupling coefficient is negative; otherwise, it is positive.

③ If the pitch of joint 3 is 16 mm, and joint 3 moves upward by one pitch when joint 4 completes one revolution, the coupling coefficient is: 16 / (2π) × (-1) = -2.55.

④ When calculating the coupling coefficient, the default "Unit" on the "Mechanical type" interface is mm. If the unit is changed to μm or nm, the calculated result must be scaled down proportionally based on step ③.

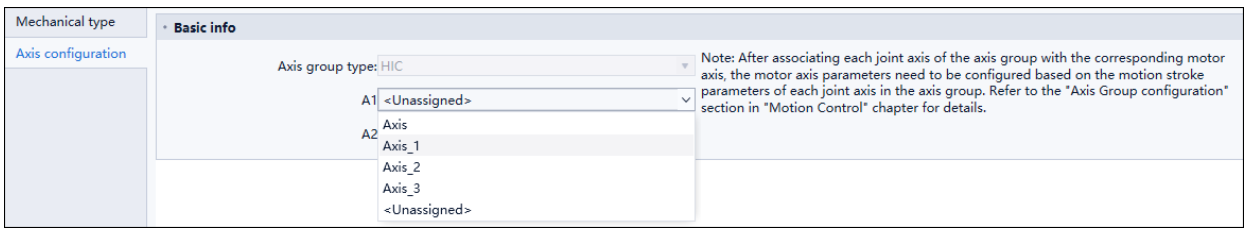


For example:

- If "Unit" is set to μm, the coupling coefficient calculated in step ③ must be scaled down by the conversion factor between mm and μm – since 1 mm = 1000 μm, the coefficient needs to be divided by 1000. Thus, -2.55/1000 equals -0.00255.
- If "Unit" is set to nm, the coupling coefficient calculated in step ③ must be scaled down by the conversion factor between mm and nm – since 1 mm = 1000000 nm, the coefficient needs to be divided by 1000000. Thus, -2.55/1000000 equals -0.00000255.

- If the axis group type is set to the "**HIC**" mechanical model, the configuration interface only supports setting the "Mechanical type" and "Axis configuration" panels, as shown in the figure below.



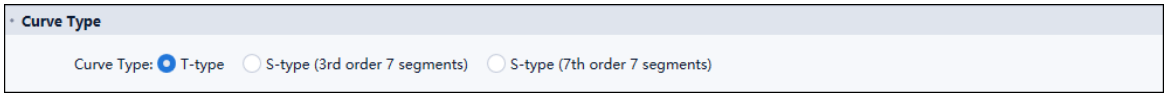


  - In the "Mechanical type" panel, you can set the number of axes in the custom axis group, with a range of [2,6].

- In the "Axis configuration" panel, you can set the axes corresponding to each joint of the custom axis group.

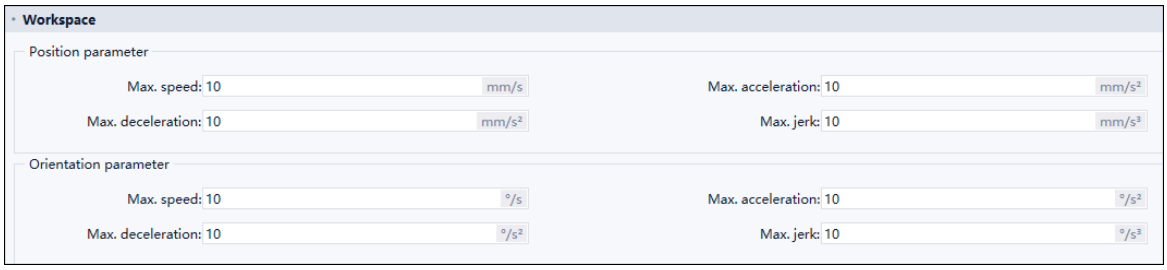
16.2.2.4 Motion Parameters

1. Curve type



Supports T-type (default), S-type (3rd order 7 segments), S-type (7th order 7 segments). This defines the planning method for the synthesized trajectory, which has no direct relation to single-axis planning methods.

2. Workspace

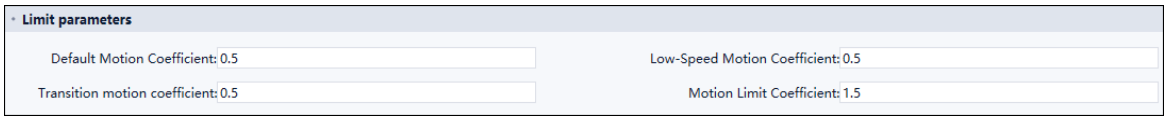


- The maximum values set for "Position parameter" apply to the position space, which limit the synthesized trajectory for position.
- The maximum values set for "Orientation parameter" apply to the pose space, which limit the synthesized trajectory within the pose space of axis groups including pose settings.

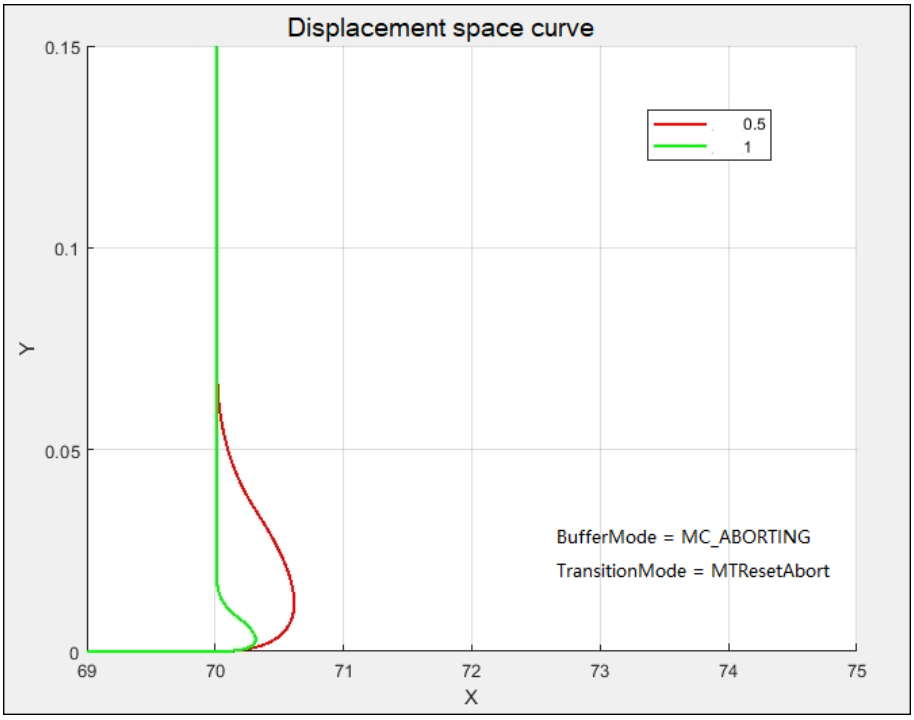
NOTE

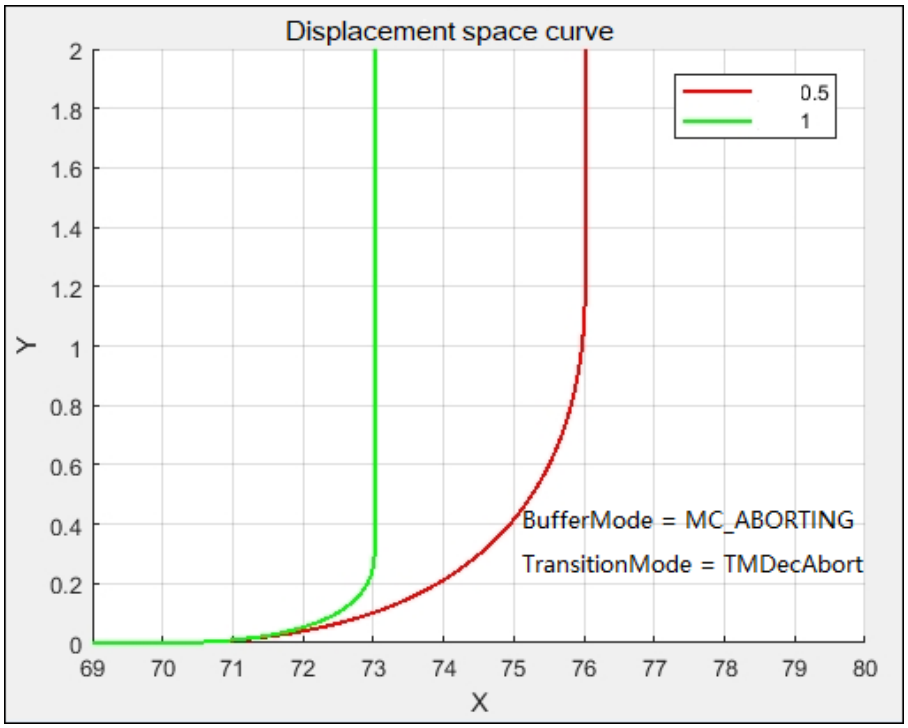
- The above are the maximum motion parameters in Cartesian space. The maximum motion parameters of individual joints shall be configured in the associated axis settings.
- When the axis group type is set to "SCARA" or "XYZA", the workspace support settings for "Orientation parameter"; for other axis group types, only "Position parameter" can be set.
- For EVO500 series PLC V1.2 version, "Orientation parameter" cannot be set for "XYZA" axis group type.

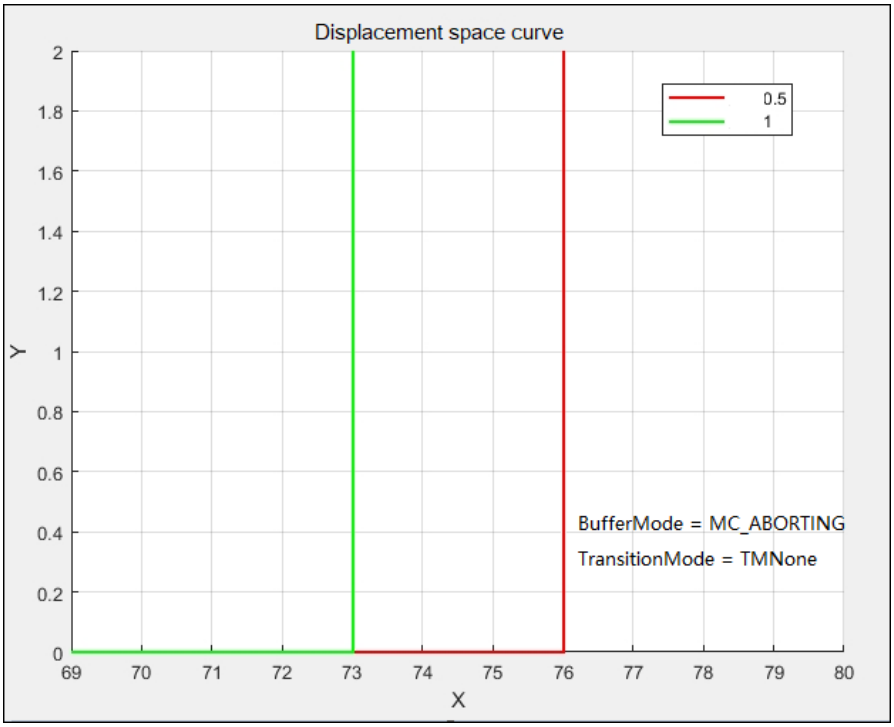
3. Limit parameters



- Default motion coefficient: Used to set the default speed for continuous trajectories. The continuous trajectory function is currently not available, so this parameter is ineffective.
- Low-speed motion coefficient: Used in the MC_GroupJog instruction for ratio settings. The MC_GroupJog instruction is currently not available, so this parameter is ineffective.
- Transition motion coefficient: Used when motion instructions are executed in Abort mode to construct transition trajectories. The higher the coefficient, the shorter the transition trajectory and the faster the transition is completed. The figure below shows the effect of different transition modes with different coefficients.

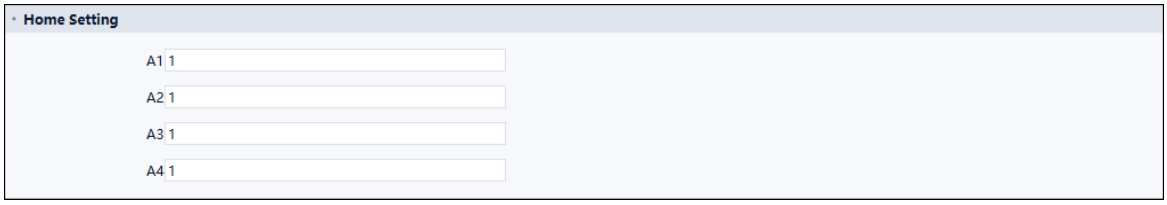

Displacement space curve
0.15
0.1
0.05
0
Y
69
70
71
72
73
74
75
X
0.5
1
BufferMode = MC_ABORTING
TransitionMode = MTResetAbort


Displacement space curve
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Y
69
70
71
72
73
74
75
76
77
78
79
80
X
0.5
1
BufferMode = MC_ABORTING
TransitionMode = TMDecAbort



◦ Motion limit coefficient: Used to limit virtual axis speed in ACS space (i.e., axis space). During interpolation, the speed of each virtual axis must not exceed the product of the motion limit coefficient and the single-axis speed limit; otherwise, a fault will be reported.

4. Home settings



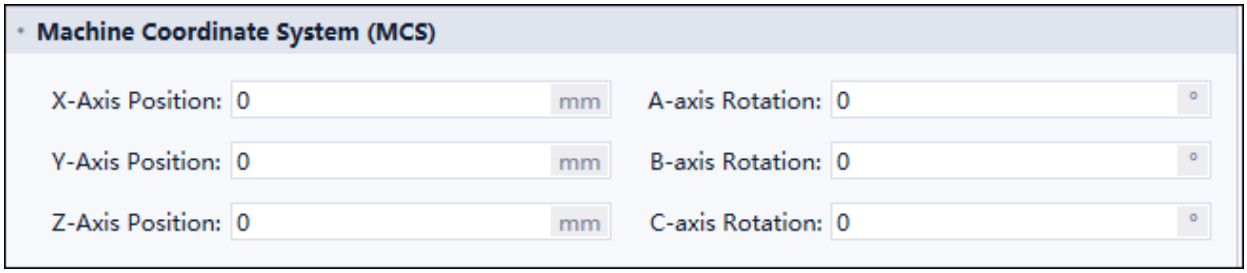
The defined homing sequence of single axes shall not contain descending order, and the homing sequence number of the first axis must be 1. Take the **Cartesian XYZA axis group type** as an example. Valid homing sequences include [1, 1, 1, 1;], [1, 2, 3, 4;], [1, 2, 2, 2;], [1, 2, 2, 3;], [1, 2, 3, 3;], etc. Axes assigned the same sequence number will perform homing simultaneously.

NOTE

The configuration of homing parameters must be completed in the bound associated joint axes.

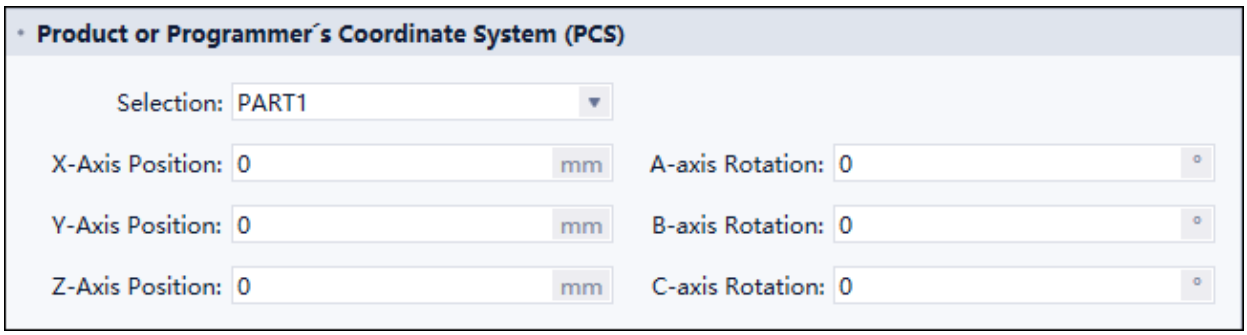
16.2.2.5 Coordinate System Parameters

1. Machine coordinate system (MCS)



Used to set the base coordinate system of the machine (MCS is currently not enabled, so default parameters can be kept).

2. Product or programmer's coordinate system (PCS)

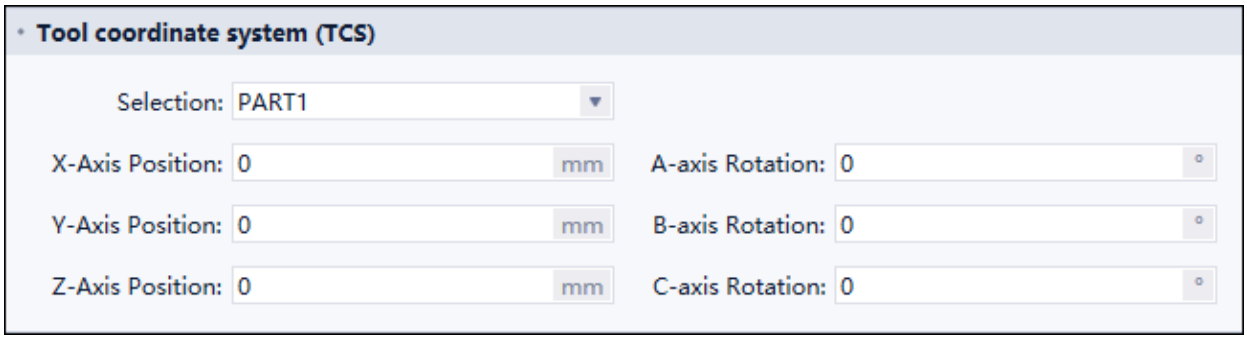


Used to store workpiece coordinate systems, a total of 10 sets can be stored.

NOTE

After the product coordinate system IDE is set, it is stored in the axis group object. To activate it, select the corresponding coordinate system for configuration via the MC_SetCartesianTransform instruction.

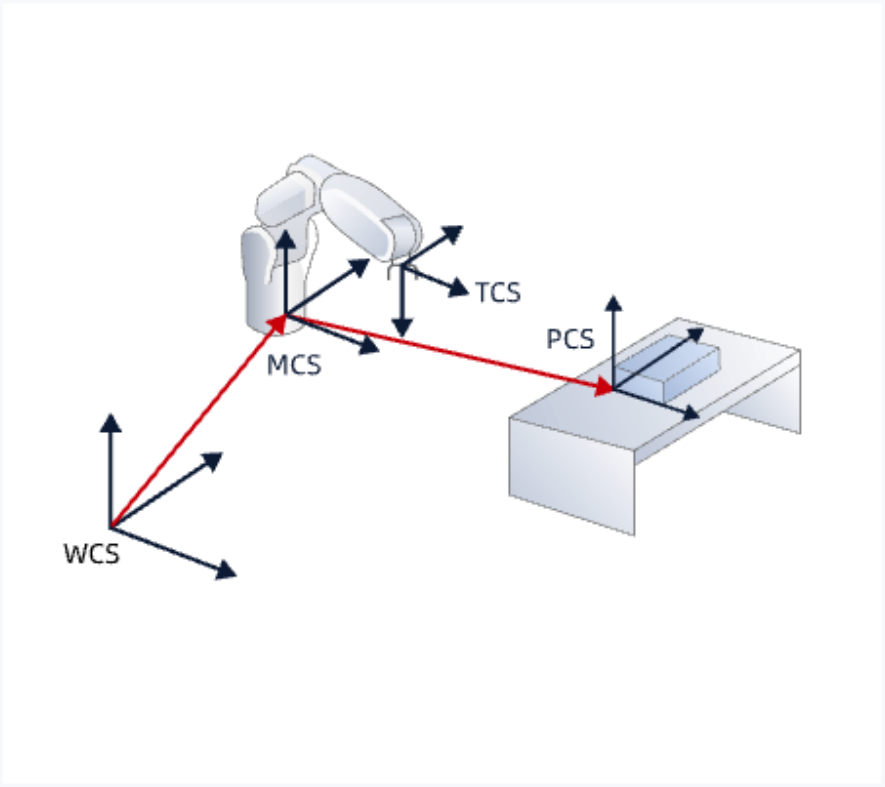
3. Tool coordinate system (TCS)



Used to store tool coordinate systems, a total of 10 sets can be stored (this function is currently not enabled).

4. Coordinate system diagram

Coordinate System Diagram



Note: The rotation angles of the coordinate system ABC correspond sequentially to the ZYX Euler angles.

16.2.2.6 HIC Configuration Example

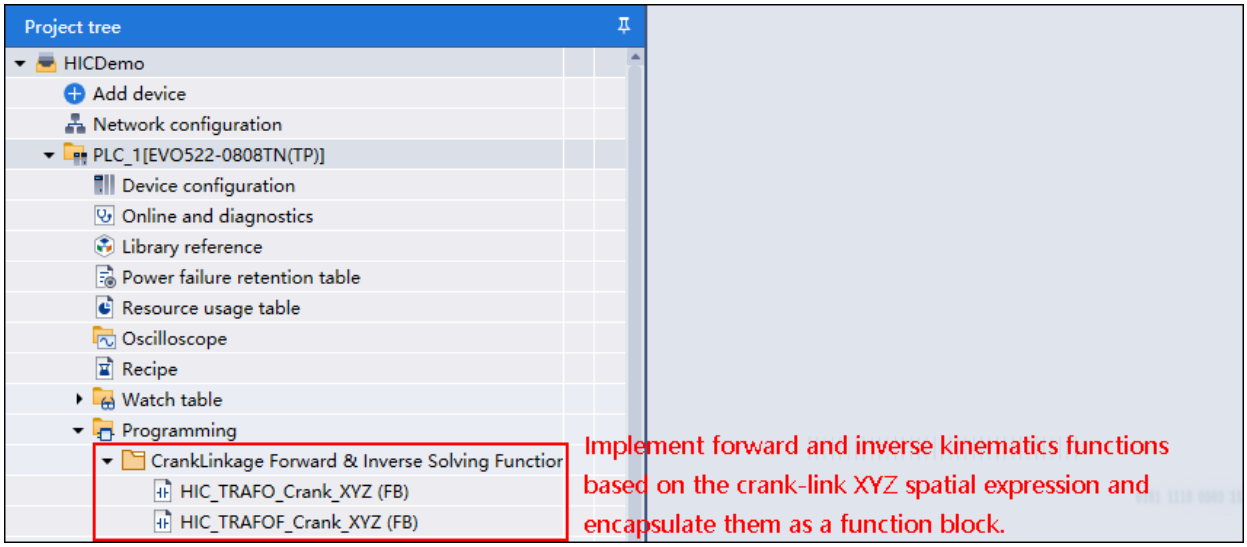
Taking a crank-link mechanism with three axes as an example: one rotational axis and two linear translational axes. Assuming its crank length is L_1 and the crank rotation angle is Θ_A, its end point in space can be expressed by the following formula.

$$\begin{cases} X = L_1 \cdot \cos\theta_A \\ Y = Y_{axis} \\ Z = Z_{axis} \end{cases}$$

Based on this formula, its forward kinematics function and inverse kinematics function can be derived. The example omits the specific implementation of the kinematics functions.

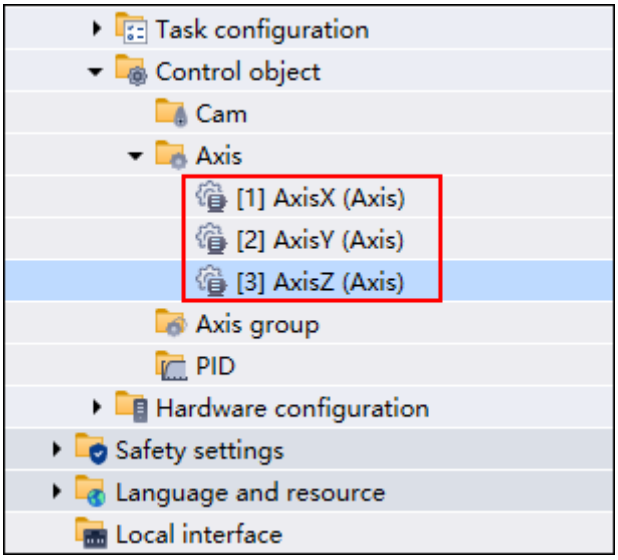
Kinematics function implementation

Encapsulate the implemented forward kinematics and inverse kinematics functions as function blocks, enabling forward and inverse kinematics calculations for the crank-link configuration.



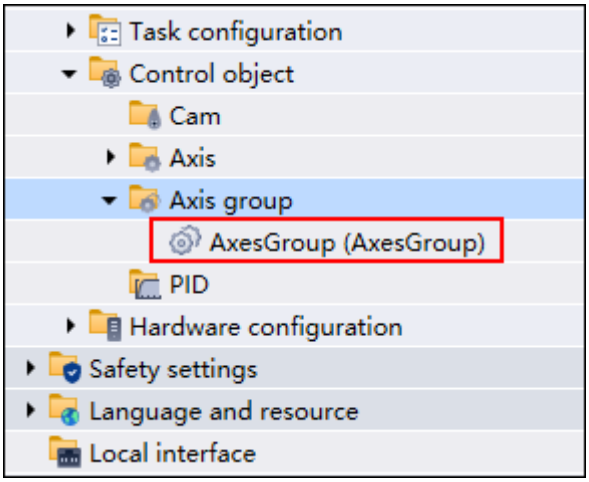
Axis addition

Right-click an axis under "Control object" in the left project tree, select "Add axis" to add the axes corresponding to the crank-link mechanism, and bind them to the servo. A virtual axis is used as an example here.

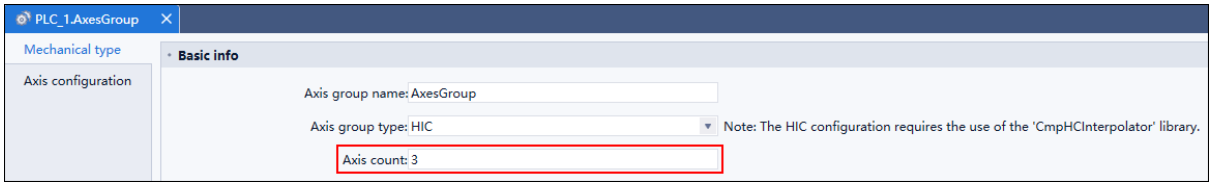


Axis group addition

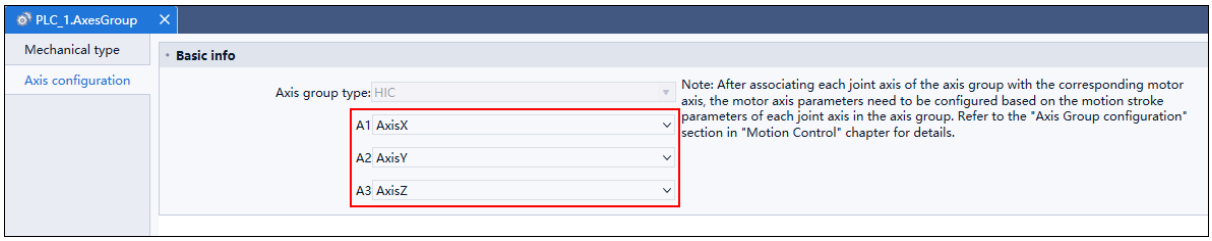
1. Right-click an axis under "Control object" in the left project tree, and select "Add axis group".



2. Set the "Axis count" according to the configuration (range: 2 to 6). In this example, it is set to 3.

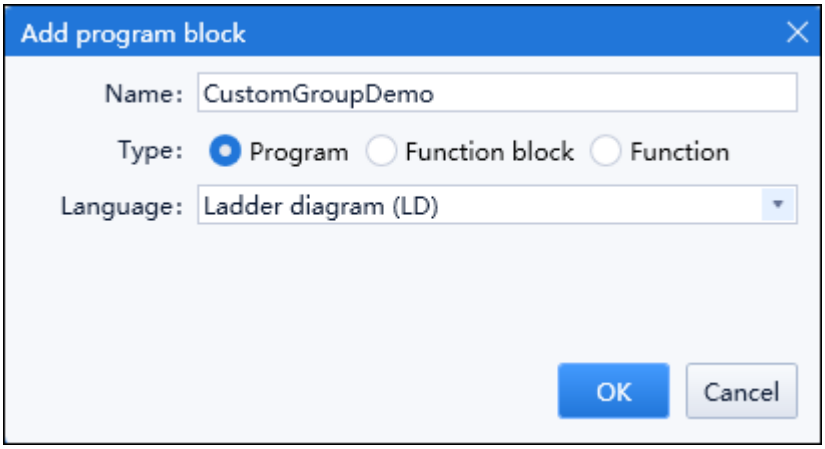


3. Assign axes to the axis group based on the joint-to-axis correspondence.



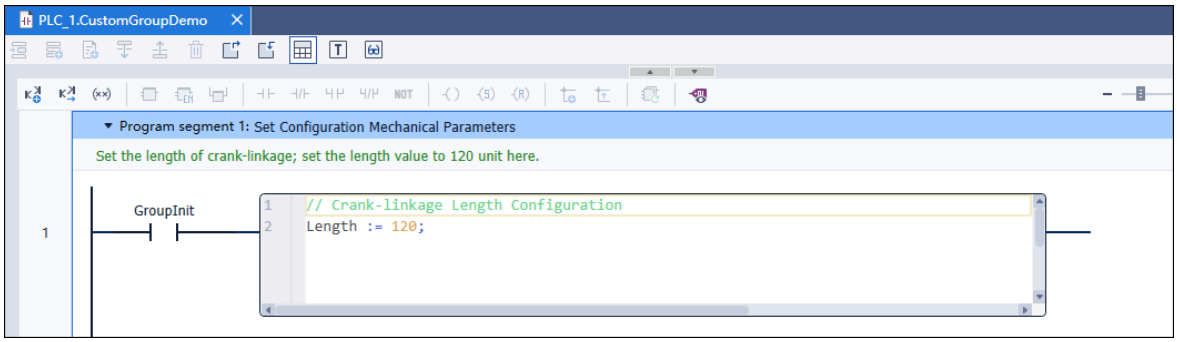
Program example

1. Right-click "Programming" in the project tree and select "Add program block".

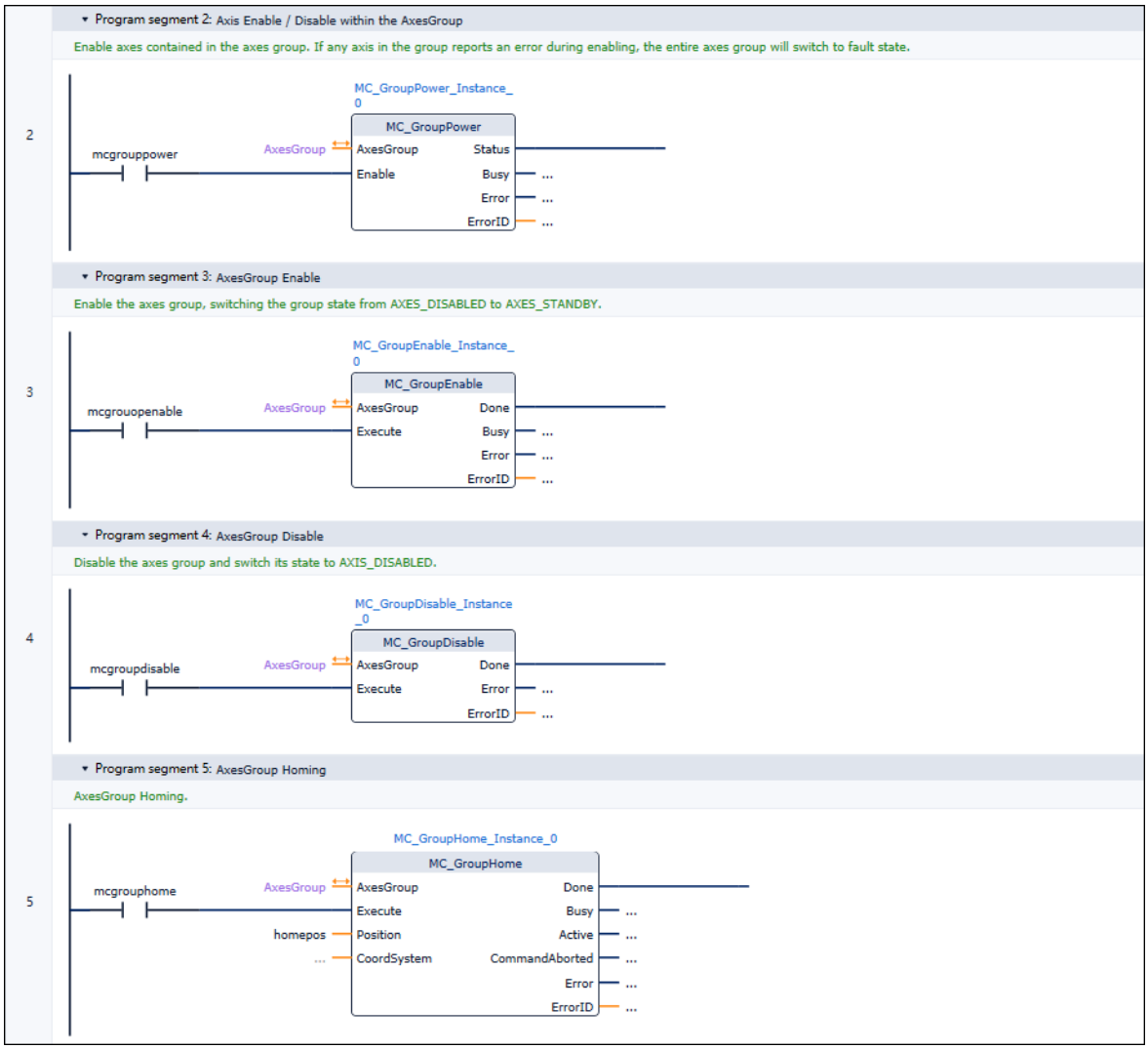


2. Follow the steps below to implement position control for the crank-link mechanism.

   a. Set the mechanical parameters of the configuration for forward/inverse kinematics functions.



b. Add axis group-related instructions.



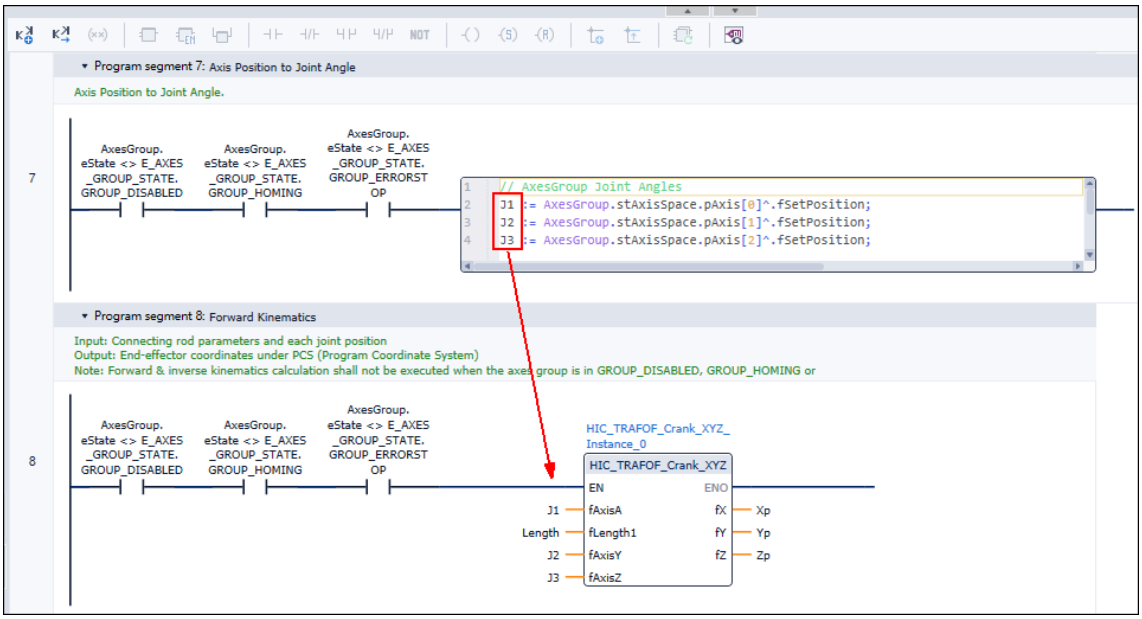
NOTE

The HIC configuration only supports instructions from the "MotionControl" library, such as MC_GroupPower, MC_GroupEnable, MC_GroupDisabled, MC_GroupHome, and MC_GroupReset. When calling other axis group motion instructions from the library (e.g., MC_MoveLinearAbsolute), the Done output parameter is set immediately after triggering, but no position output is generated on the axes in the group.

c. Place the implemented forward kinematics and inverse kinematics functions in the program. The placement order can be adjusted according to different scenario requirements. In this example, forward kinematics function is placed above inverse kinematics.

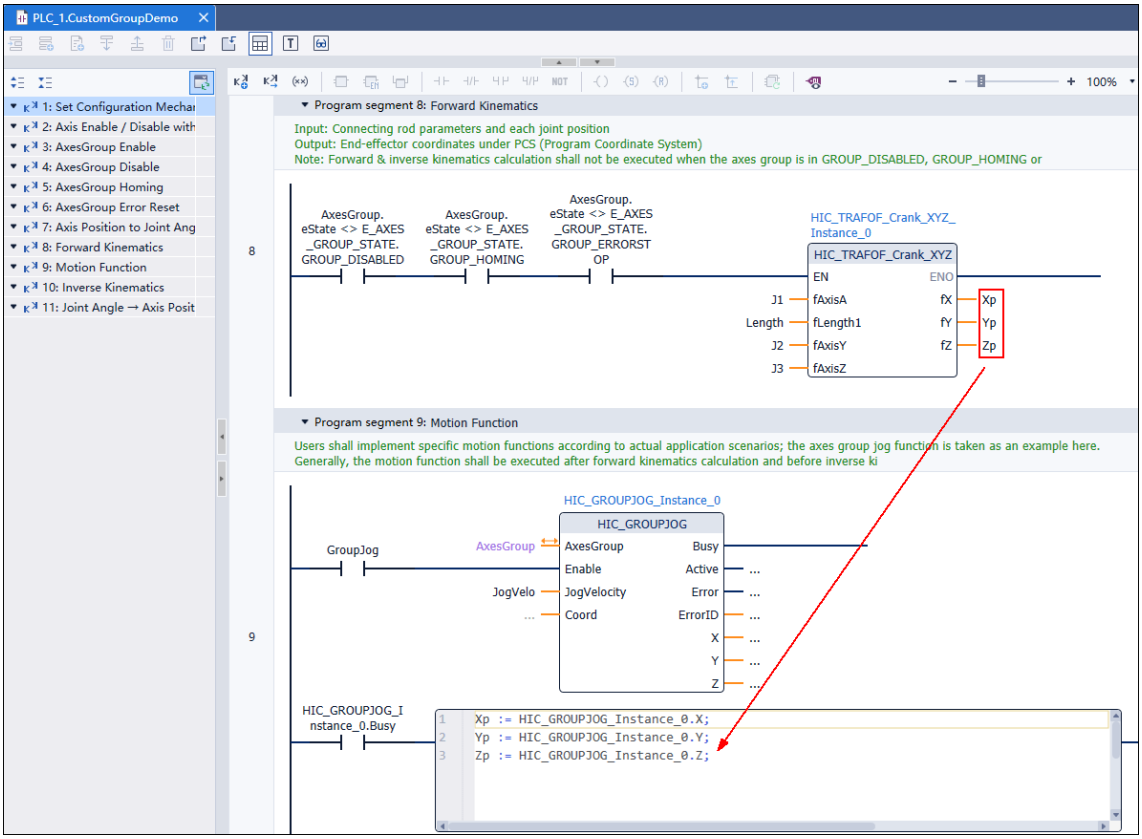
α. Forward kinematics (ACS -> PCS): Converts the joint angles of each axis in the axis group to spatial positions. Obtain each joint angle, combine with link parameters, and pass them to the forward kinematics function. In this example, kinematics solving is skipped when the axis group is in

GROUP_DISABLED, GROUP_HOMING, or GROUP_ERRORSTOP.



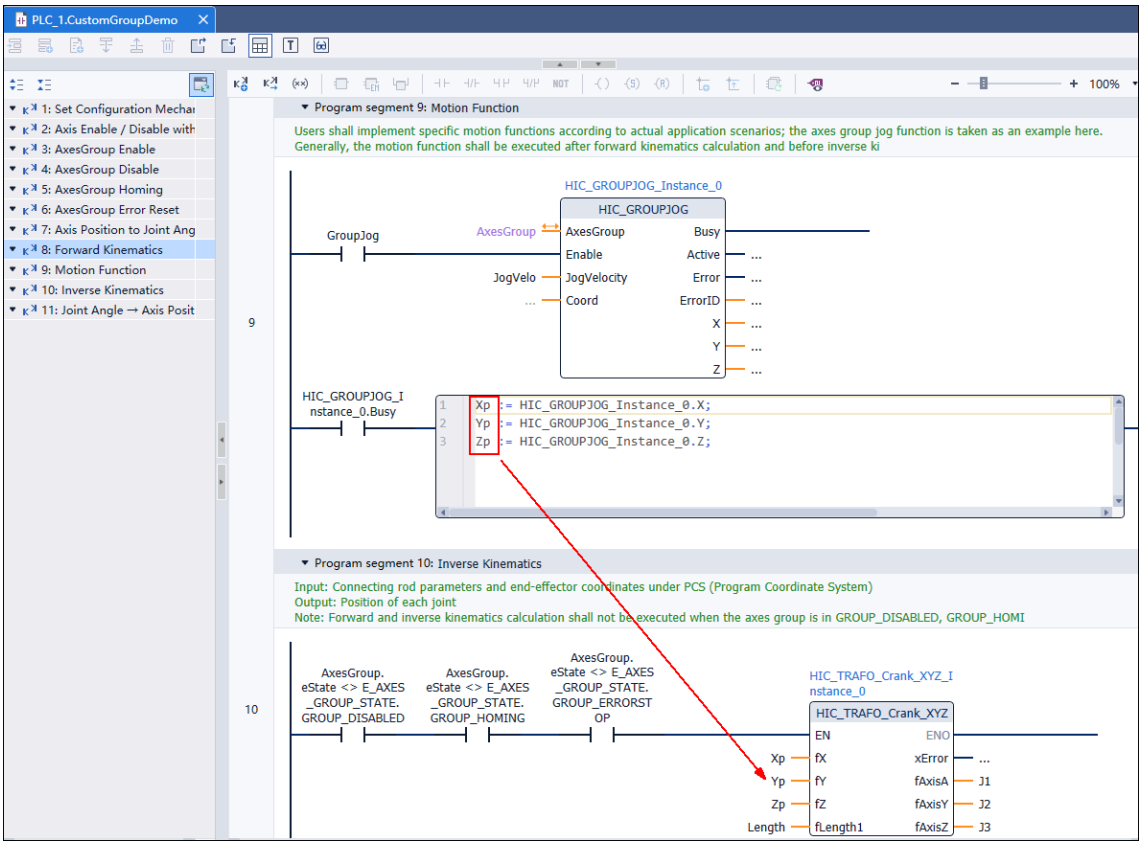
β. Motion function: The user implements specific motion functions based on application requirements. This example uses axis group jogging function. Generally, the motion function is placed after forward kinematics and before inverse kinematics.

γ.

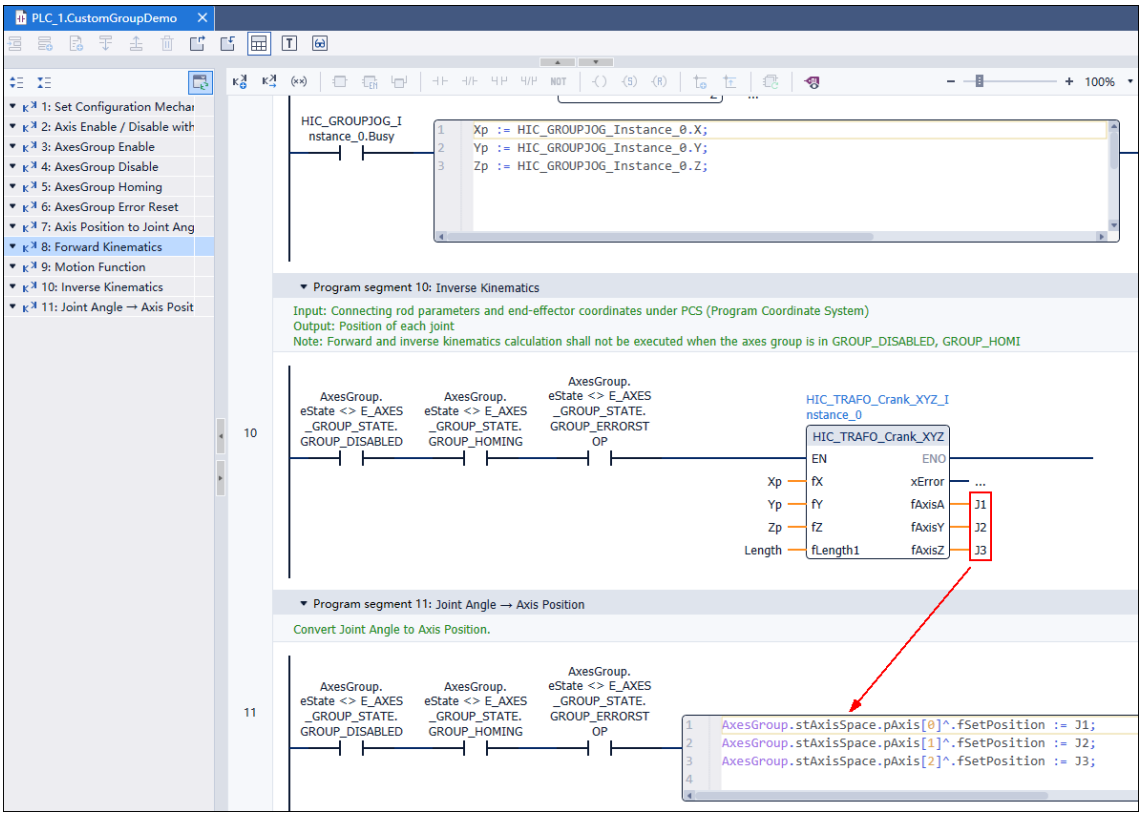


Inverse kinematics (PCS -> ACS.): Converts configuration spacial position to each axis joint angle. Pass the XYZ spatial positions output from the motion function, combined with link parameters, to the inverse kinematics function. In this example, kinematics solving is skipped when the axis group

is in GROUP_DISABLED, GROUP_HOMING, or GROUP_ERRORSTOP.

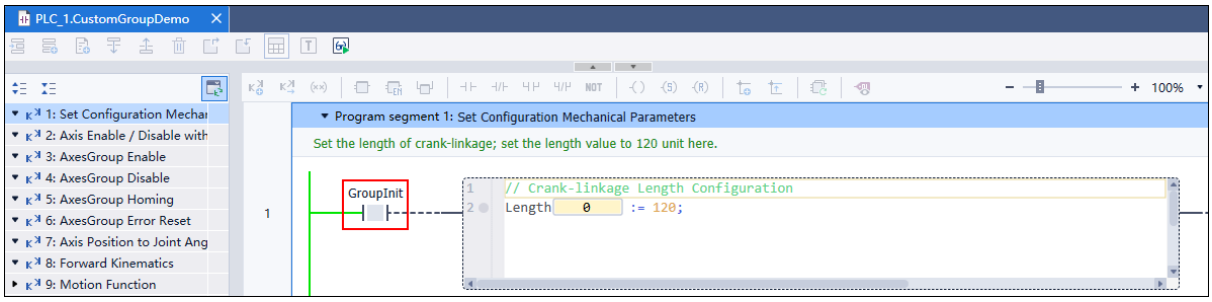


δ. The joint angles solved by the inverse kinematics function are assigned to the corresponding axes in the axis group to control its motion.

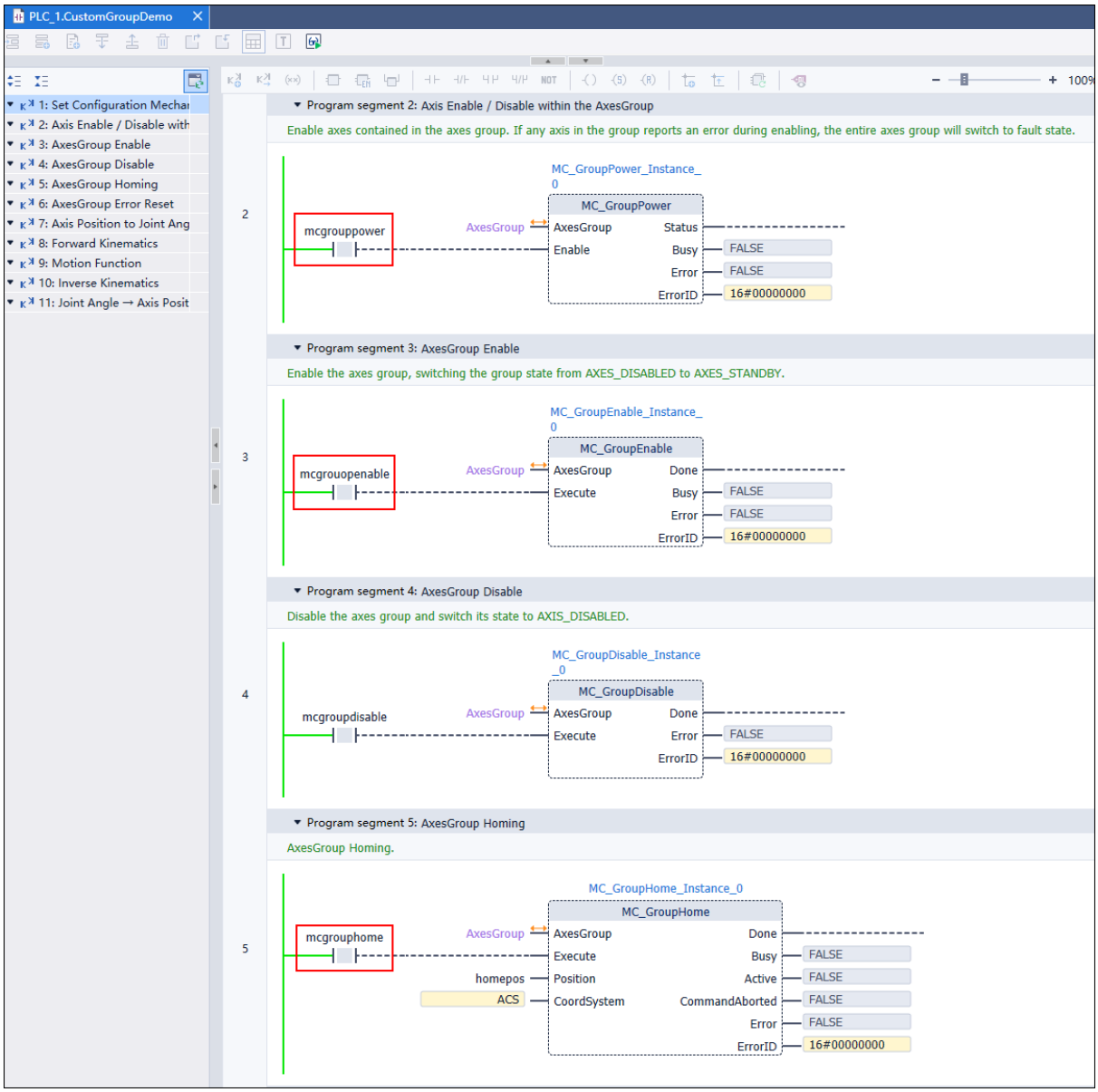


Examples

1. Set configuration parameters.

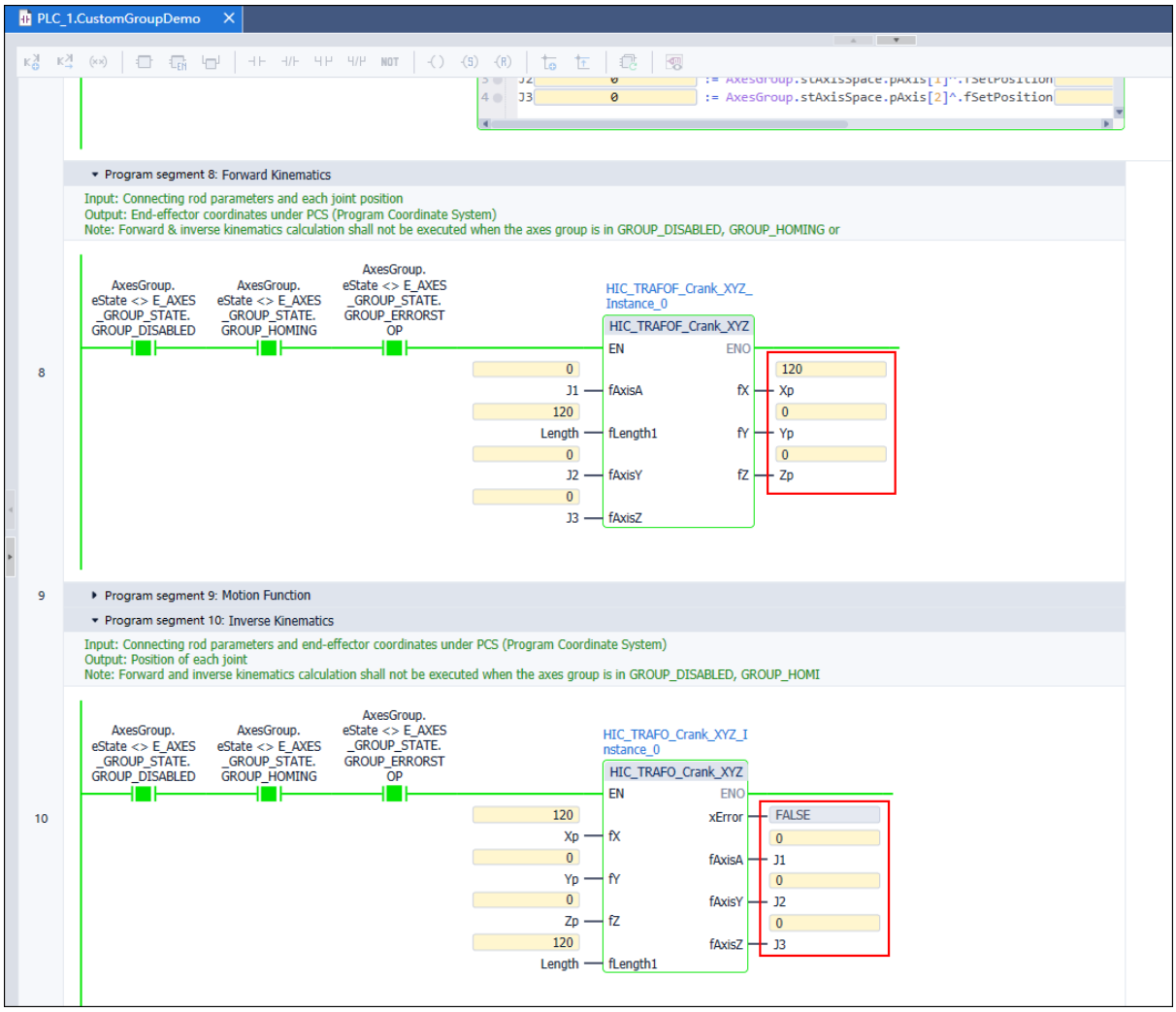


2. Execute MC_GroupPower (enable axes in the axis group), MC_GroupEnable (enable the axis group), and MC_GroupHome (home the axis group). If the axis group is in GROUP_ERRORSTOP or an axis in the axis group is in AXIS_ERRORSTOP, execute MC_GroupReset. If axis group motion is complete and no further control is needed, execute MC_GroupDisabled to disable the axis group.



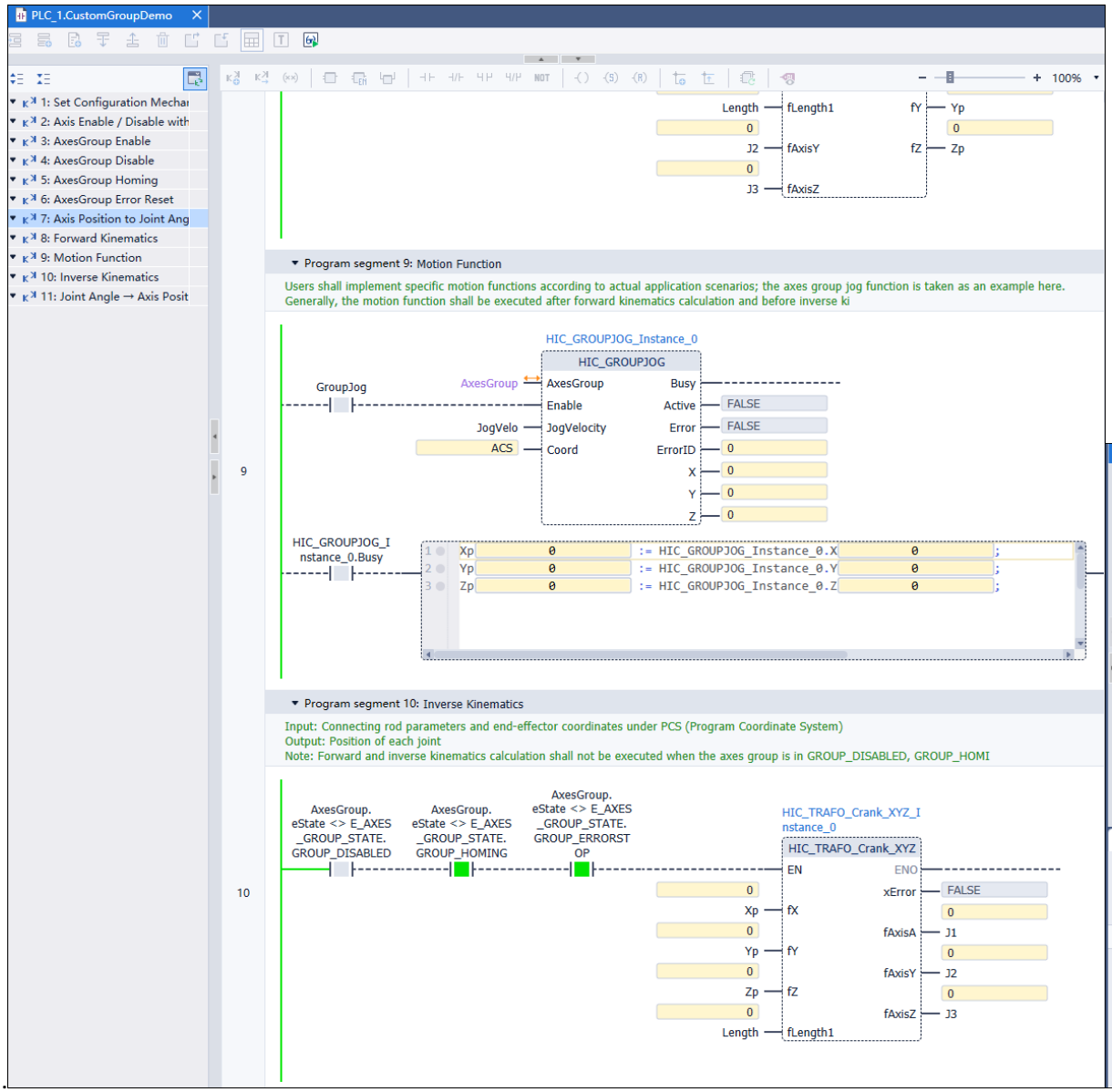
3. After the axis group completes MC_GroupHome, each axis joint is at 0. The forward kinematics function returns XYZ spatial position (120, 0, 0), which matches the configuration. The joint angles solved by the

inverse kinematics function also match the axis group joints.



4. Start the motion function HIC_GroupJog, which outputs XYZ spatial positions. The inverse kinematics function then calculates the corresponding joint angles and passes them to the joints in the axis group to

control the group's



motion.

16.3 Cam Configuration

16.3.1 Overview

- Overview of cam motion

  Cam motion adopts the relative motion characteristics between a mechanical cam and a tappet. It refers to a continuous synchronous motion where a slave servo axis follows a master axis according to a predefined nonlinear position relationship, to fulfill equipment motion requirements. It is widely used in applications requiring synchronization, such as fixed-length cutting, chasing shear control, flying shear control, and multi-color overprinting.

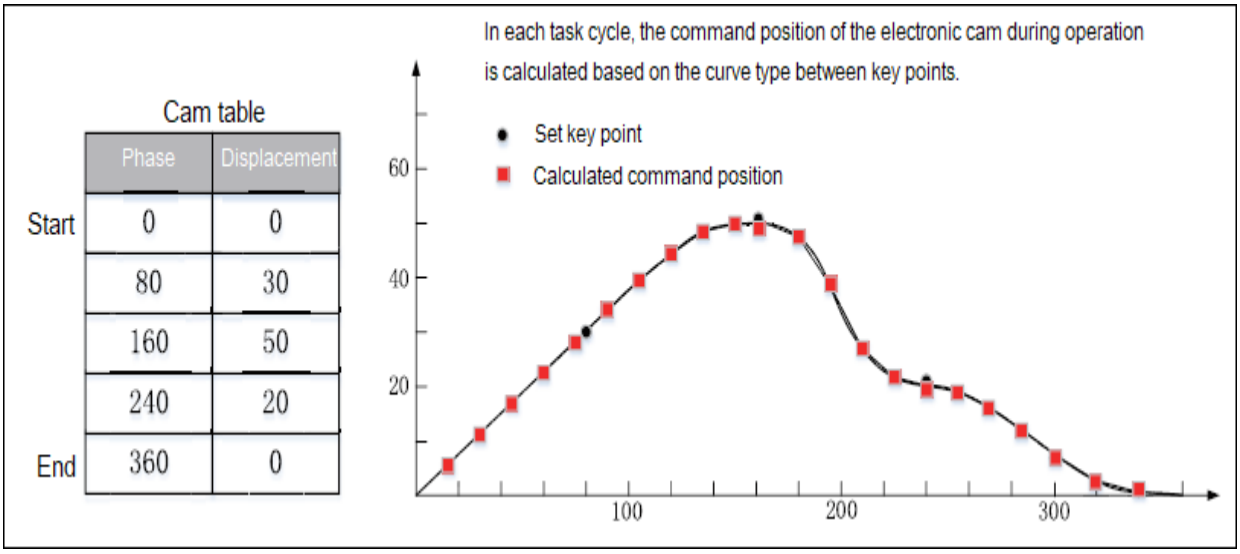
  The position relationship between the master and slave axes in an electronic cam curve is shown in the following figure, with the horizontal axis representing the master axis position and the vertical axis representing the slave axis position:



- The cam motion control function can be realized via software, which means using a software-based "cam table" to replace a mechanical cam. Therefore, this function is also called electronic cam control. Compared to mechanical cams, it has the following features:
  - Cam profile creation: Supports drawing various cam curves to create cam profiles;
  - Diverse cam profiles: Supports selecting multiple cam tables, which can be dynamically switched during operation through the MC_GenerateCamTable instruction;
  - Cam profile modification: Supports dynamic modification of key points or dense points in the cam table during operation;
  - Multiple cam slave axes: Supports multiple cam slave axes;
  - Cam clutch: Supports enabling or exiting cam motion during operation through the MC_CamIn or MC_CamOut instructions;
  - Unique functions of electronic cams: Supports virtual master axes, phase superposition, and other features.
- Technical terms
  - Master axis: The reference axis used for synchronous control;
  - Slave axis: The servo axis that follows the master axis position according to the required nonlinear characteristics;
  - Cam table: A data table or cam curve describing the relative position, range, and periodicity between the master and slave axes.
  - Phase: The relative value of the master axis from the starting position of the cam table.
  - Displacement: The relative value of the slave axis from the starting position of the master axis tracking.

16.3.2 Introduction to Cam Table

Cam data refers to a set of data composed of the master axis phase and the slave axis displacement. A cam table refers to a combination of multiple cam data sets. Phase and displacement values of cam data in a cam table are relative quantities expressed in relation to the start point "0.0". The phase values of the cam data in the cam table need to be arranged in ascending order.



There are two types of cam table data:

- "Key point" type: Each cam point of this type contains the positions, speed ratio, acceleration ratio, jerk ratio and curve type of master and slave axes. Currently supported curve types include: straight line, parabola, cubic curve, quintic curve, septic curve, cosine acceleration curve, sine acceleration curve, modified T-type acceleration curve. Due to the large amount of data contained in each key point, the maximum number of points supported per table is 20000.
- "Dense point" type: This type only contains the position of master and slave axes, and each cam point is linearly connected. This type of cam table can support up to 65535 points.

16.3.3 Cam Table Data Structure

The cam table data structure supports both key points and dense points. Each cam table is divided into two parts: One part describes the key information of the cam table, and the other describes the specific data stored in the table.

Number of cam table points: Up to 20000 key points and up to 65535 dense points are supported.

The MC_CAM_TABLE_REF data structure is defined as follows:

Table16-3-3-1 Cam table data structure

Variable Name	Data Type	Description
strCamTableName	STRING	Cam table name
iNodeType	INT	Node type: 1: Key node. 2: Dense node.
iNodeNumber	UINT	Number of points
pNodePtr	POINTER TO BYTE	Pointer to the cam table point array

In the cam table point array, key point or dense point arrays need to be bound, with the following data types:

The MC_CAM_KEY_NODE_REF data structure is defined as follows:

Table16-3-3-2 Key point definition

Variable Name	Data Type	Description
fPhase	LREAL	Phase (of the master axis) at the node.
fDistance	LREAL	Displacement (of the slave axis position) at the node.
fVel	LREAL	Velocity at the node.
fAcc	LREAL	Acceleration at the node.
fJerk	LREAL	Jerk at the point.
iCurveType	INT	Curve type: 1: Straight line. 2: Parabolic curve. 3: Cubic curve. 4: Quintic curve. 5: Septic curve. 6: Cosine acceleration curve. 7: Sine acceleration curve. 8: Modified T-type acceleration curve.

MC_CAM_FITTING_NODE_REF data structure definition:

Table16-3-3-3 Dense point definition

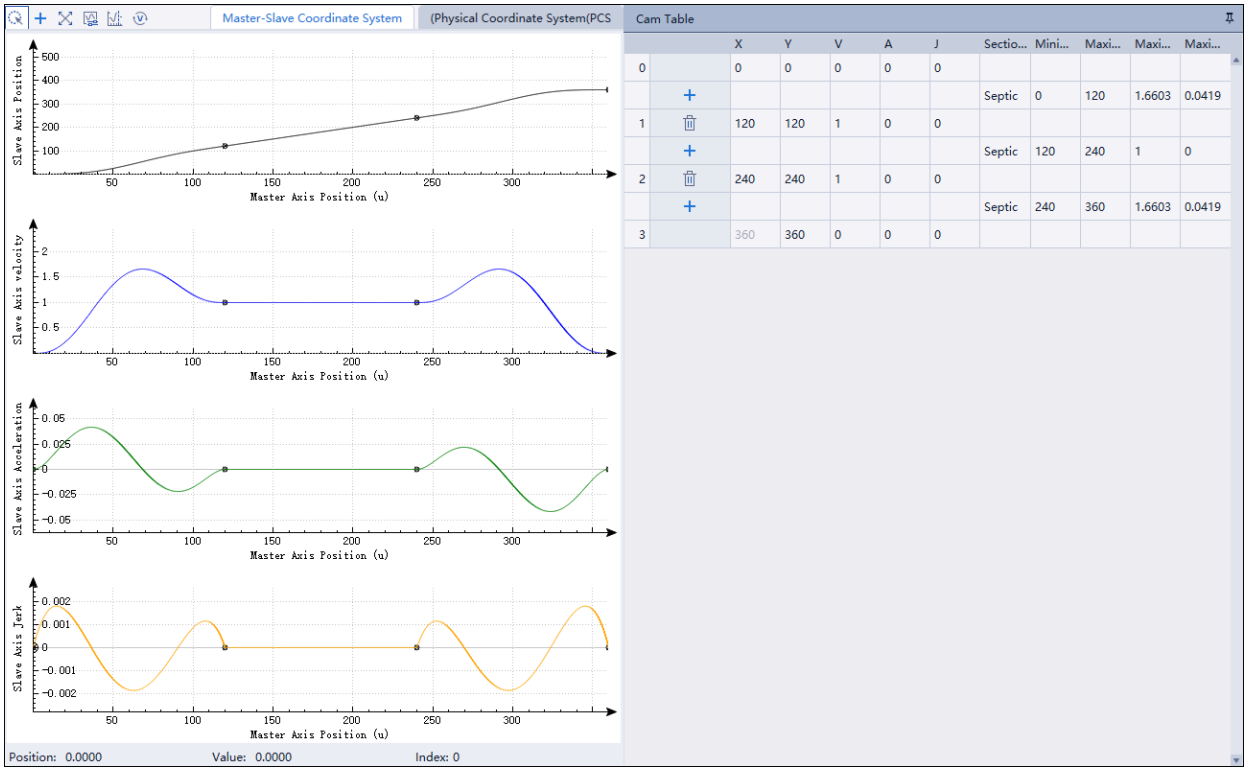
Variable Name	Data Type	Description
fPhase	LREAL	Phase (of the master axis) at the node.
fDistance	LREAL	Displacement (of the slave axis position) at the node.

16.3.4 Creating a Cam Table

1. Click the "Control object" in the left project tree to expand the control object list.
2. Right-click "Cam" and select "Add cam".
3. In the pop-up dialogue box, enter the cam name and click "OK".



By default, the added cam table is of "key point" type, which can be modified through cam table properties.



In general, a cam table of "dense point" type contains more cam points, which need to be added through the following steps:

Method 1:

1. Create a CSV file with the following content:

1	CamObjectName:	Cam_0	Cam table name
2	CAMModelTypeEnum:	2	Cam table type; 2 means dense point
3			
4	NumberOfCamPoints:	4	Number of cam points
5	X	Y	
6	0	0	
7	120	120	
8	240	240	Dense point data
9	360	360	

Note: The keywords "CamObjectName:", "CAMamModelTypeEnum:", "NumberOfCamPoints:", "X", and "Y" in the table must be exactly the same as those in the figure; otherwise, the CSV file cannot be imported.

The cam table name, number of cam points, and dense point data can be modified, but the cam table type (2) cannot be modified.

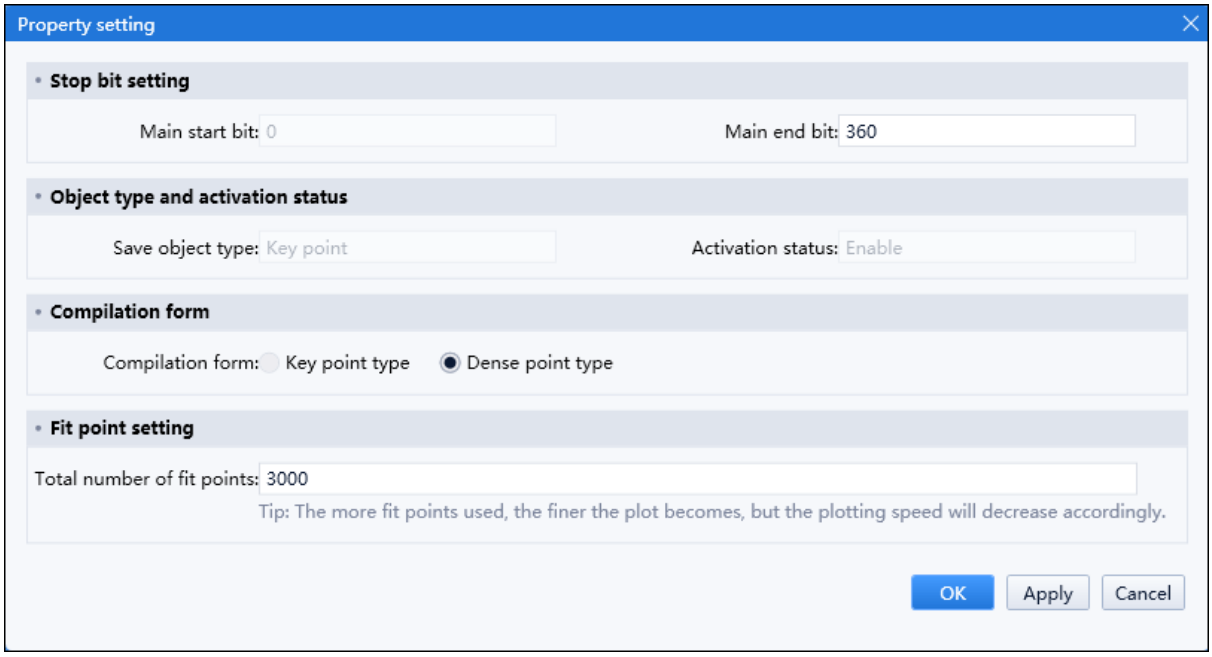
2. Based on specific process requirements, write the dense point data (X: phase, Y: displacement) into the CSV file.

NOTE

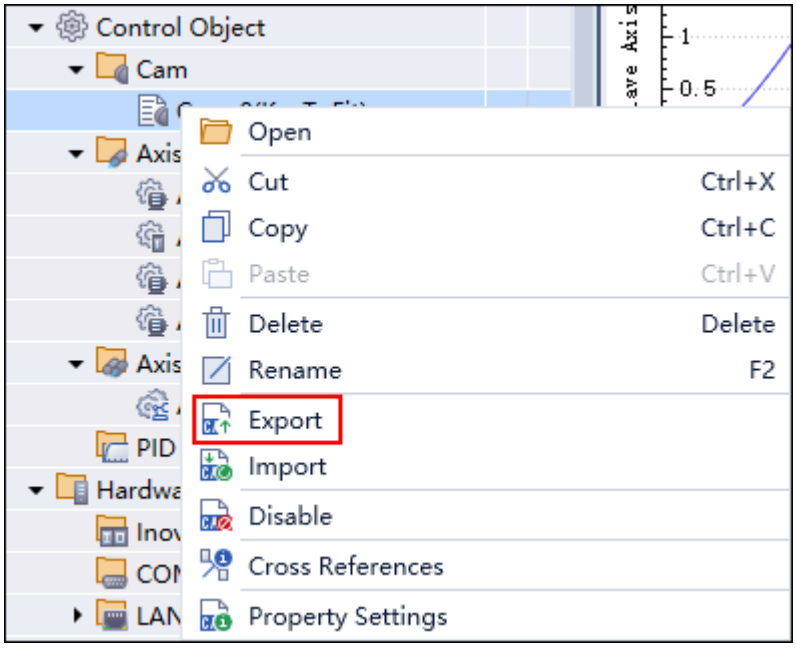
- The keywords "CamObjectName:", "CAMamModelTypeEnum:", "NumberOfCamPoints:", "X", and "Y" in the table must be exactly the same as those in the figure; otherwise, the CSV file cannot be imported.
- The cam table name, number of cam points, and dense point data can be modified, but the cam table type cannot be modified.
- The number of cam points can be less than the actual number of dense point data; otherwise, the CSV file cannot be imported.

Method 2:

1. Create a cam table in iFA, right-click the cam table and select "Property setting". In the pop-up window, modify the compilation form of the cam table to "Dense point type", as shown in the figure below:



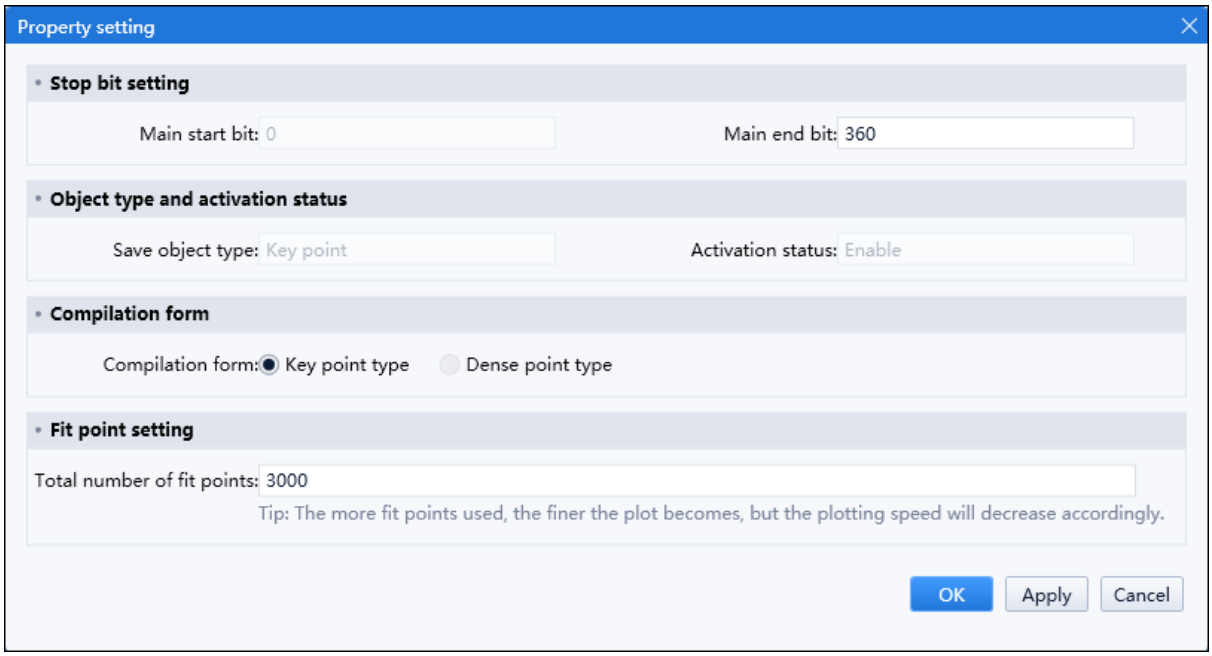
2. After modification, right-click the cam table and select "Export" to export the data to a CSV file, as shown in the figure below:



The exported file has the same format as the file created in method 1, and dense point data can be added in the table.

16.3.5 Cam Table Properties

1. Right-click the Cam_0 table and select "Property setting".
2. Set the corresponding properties in the pop-up window.

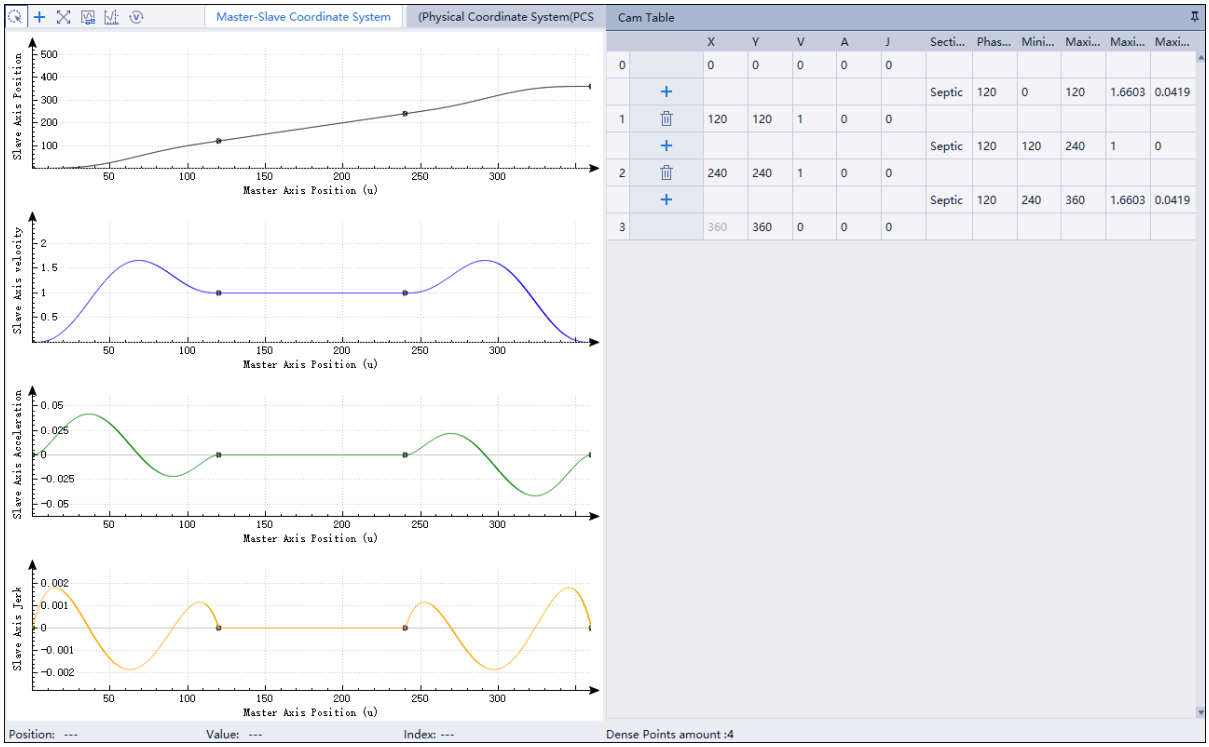


- The "Main start bit" is default to 0 and cannot be modified.
- The "Main end bit:" is the end position of the master axis and can be set according to the process scenario.
- The object type and activation status show the point type of the current cam table and whether it is disabled.
- The compilation form sets whether the point type of the current cam table is downloaded as key points or dense points, allowing switching between two types of data.
- The fit point setting adjusts the drawing precision; the more fit points, the more precise the drawing, but the drawing speed will decrease.

16.3.6 Cam Chart

16.3.6.1 Cam Table Configuration Interface

Double-click the added cam table in the project tree to open the cam table configuration interface. The interface is shown in the following figure.



16.3.6.2 Cam Diagram

The left view of the cam chart is the cam diagram view, which includes four commonly used diagrams and the following functions:

ID	Function	Description
①	Select point	This option is activated by default, allowing to select and drag any key point in a diagram.
②	Add point	Click the "Add point" icon and click a position in a diagram to add a key point at the position.
③	Adaptive	Based on the current screen size, the X and Y coordinates are automatically adjusted to fit the position curve, velocity curve, acceleration curve, and jerk curve into the screen.
④	Profile configuration	Used to set the content displayed in the diagrams, with all content displayed by default. You can set the point shape, point size, and curve color for each diagram.

ID	Function	Description
⑤	Add cursor	After adding a cursor and inputting the phase of the X master axis, click "Apply" to view the displacement and velocity/ acceleration data of the slave axis.
⑥	Switch coordinate	Used to switch between master-slave coordinate system and physical coordinate system.

Cam diagram description:

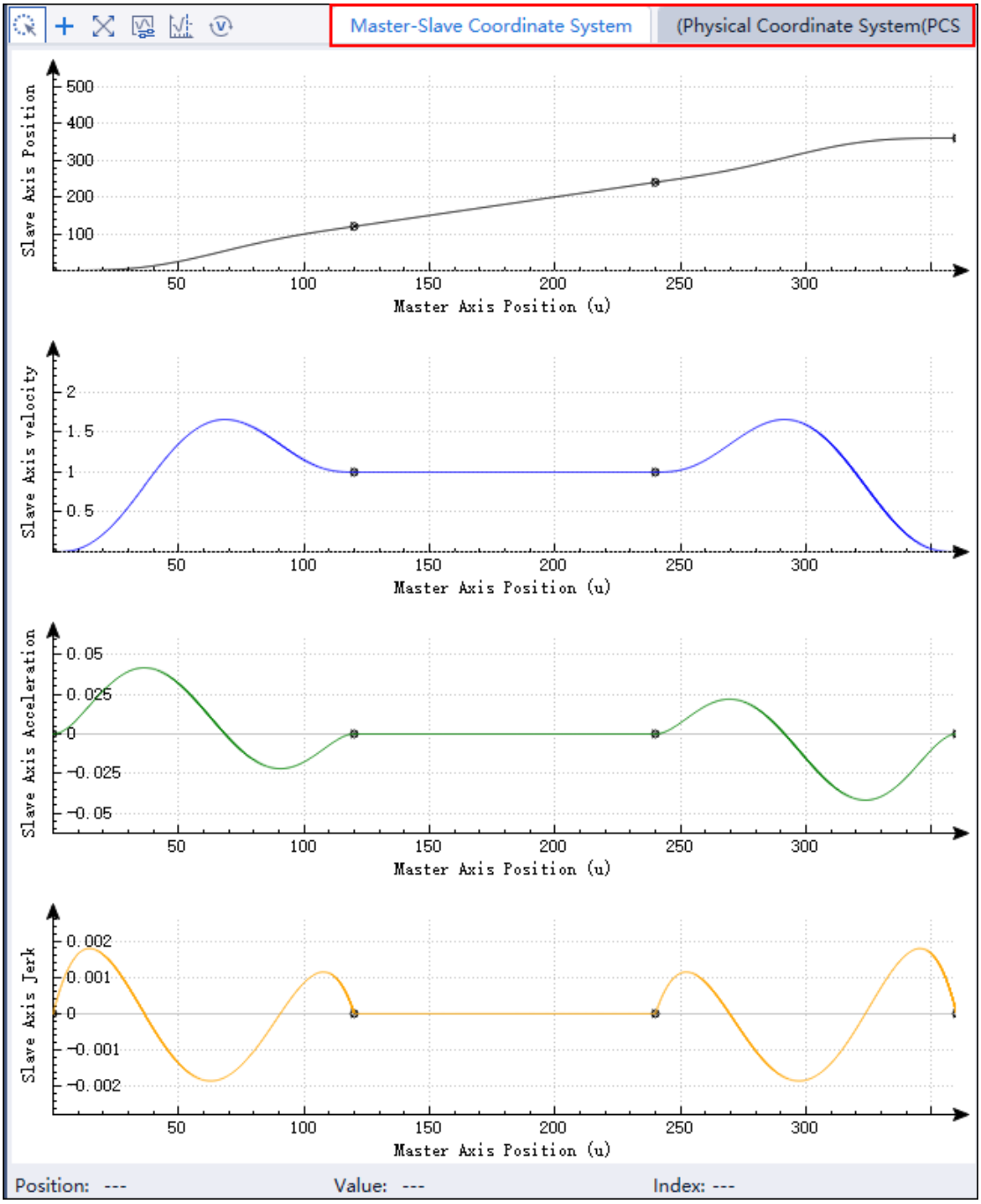
- The cam diagram supports inserting and deleting key points, editing coordinate, and selecting interpolation types.
- The cam diagram supports pitch editing, and the total number of cam interpolation points cannot exceed 65536.
- Whether the cam diagram is of dense point type or key point type, it can display position, velocity, acceleration, and jerk coordinates. Key points can be dragged and dense points cannot.
- The cam diagram supports zooming in on the selected area by dragging the mouse (zooming includes three modes: X-axis, Y-axis, and overall zoom). Adaptive zooming is also supported to automatically fit the entire cam curve to the diagram.
- When the mouse hovers over a key point on the cam diagram, the motion parameter data of the table will be displayed, and the selected key point will be marked with a bright spot. (The motion parameter data can be displayed in one table or multiple tables, and displaying in multiple tables requires selecting the curve).
- The cam diagram supports adding horizontal and vertical cursors, where the horizontal cursor can set the Y-axis position, and the vertical cursor can set the X-axis position. Precise values cannot be obtained by dragging the cursor.
- The cam diagram supports selection of graphical interfaces for velocity, acceleration, and jerk (multiple selections are supported; all interfaces are displayed by default). It also supports setting the point shape, point size, and curve color for each diagram.

Master-slave coordinate system and physical coordinate system

Overview

The master-slave coordinate system displays the relationship between the position, velocity, acceleration, and jerk of master axis and slave axis. The physical coordinate system displays the relationship between the cam cycle time and the slave axis position, velocity, and acceleration.

By default, the master-slave coordinate system is displayed. The coordinate system can be switched through the tabs in the upper right corner of the diagram view area.

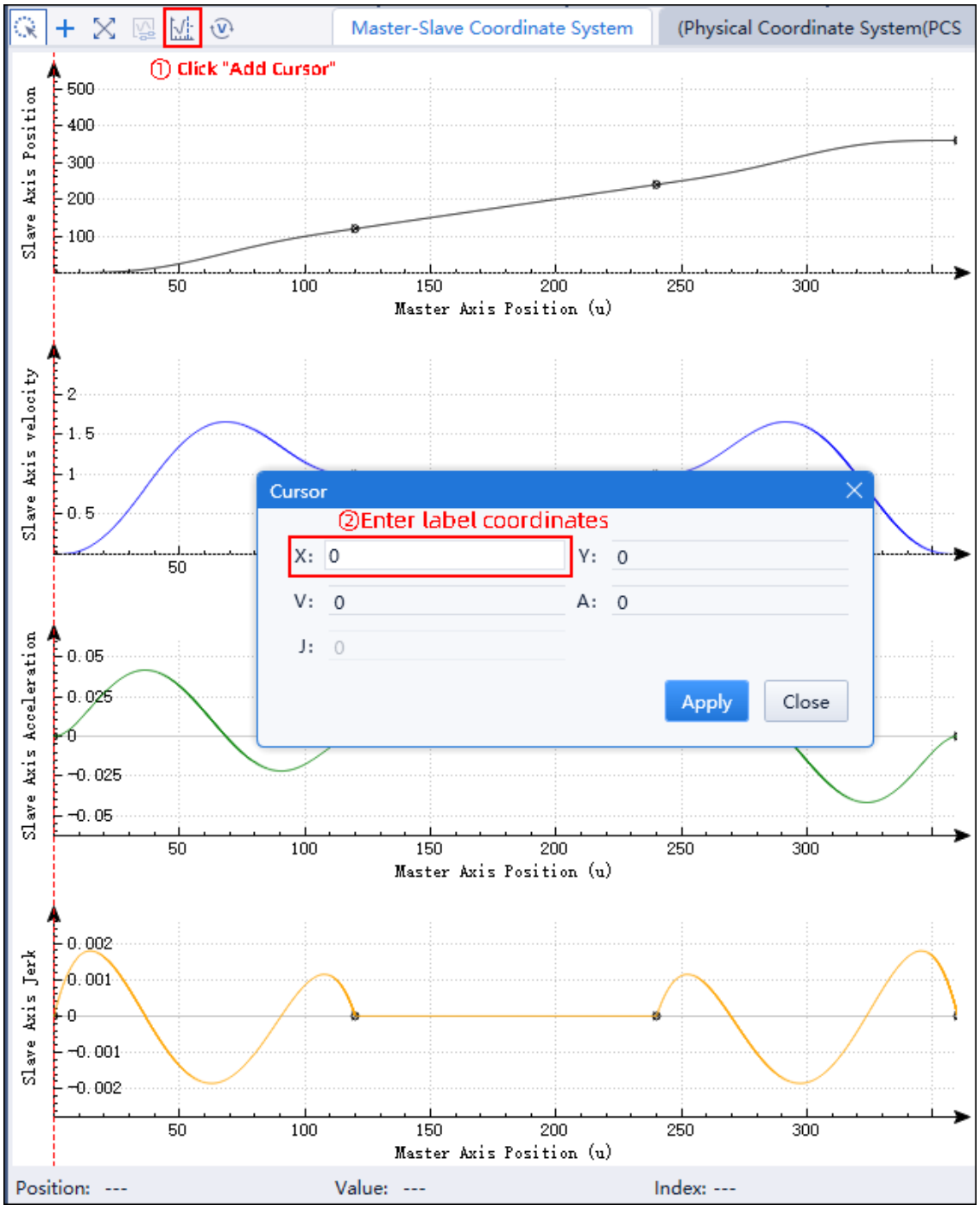


- In the physical coordinate system, click the "Set cam cycle time" icon, modify the cycle time of the cam in the pop-up window, and click "OK" to apply the changes and refresh the physical coordinate system.
- In the physical coordinate system, the jerk column in the cam table is not displayed.
- In the physical coordinate system, the jerk diagram is not displayed. You can choose to display the position, velocity, and acceleration diagrams as needed.
- In the physical coordinate system, points can only be added and moved on the master/slave axis position diagram.
- After a cursor is added, the "Graphic configuration" icon is unavailable in the master-slave coordinate system, and both the "Graphic configuration" icon and the "Set cam cycle time" icon are unavailable in the physical coordinate system.
- In the physical coordinate system, selecting or deselecting "Jerk curve setting" in the "Graphic configuration" window only affects the master-slave coordinate system.

- In the physical coordinate system, a cursor can be added to the position diagram, which can be dragged to display the X/Y/V/A data in the physical coordinate system in the pop-up "Cursor" window.
- In the physical coordinate system, key points in each diagram can be moved by mouse.
- The cam comparison function does not distinguish between coordinate system types and compares data in the master-slave coordinate system.

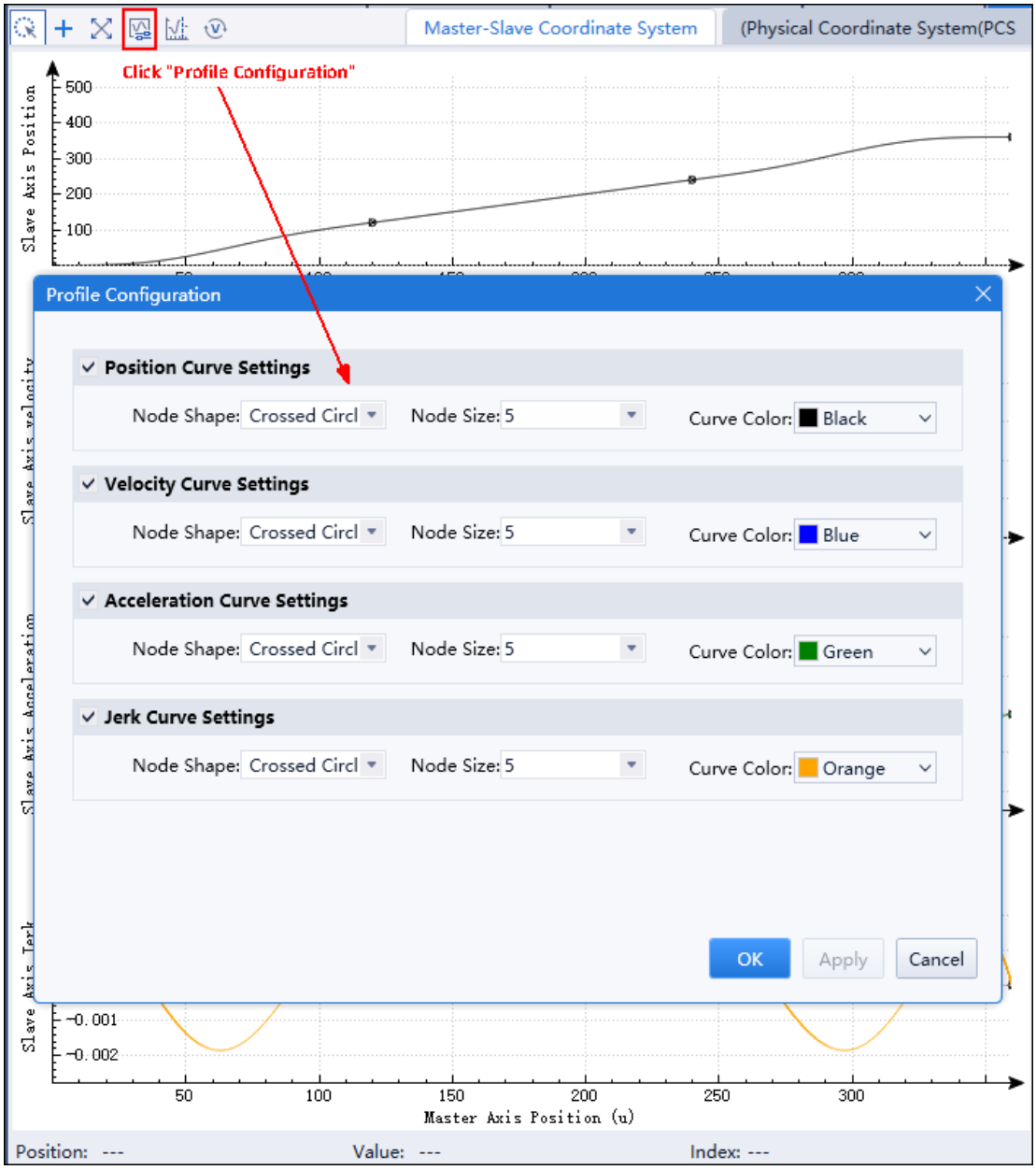
Cursor

The cursor can be displayed on the cam diagram for easier observation.



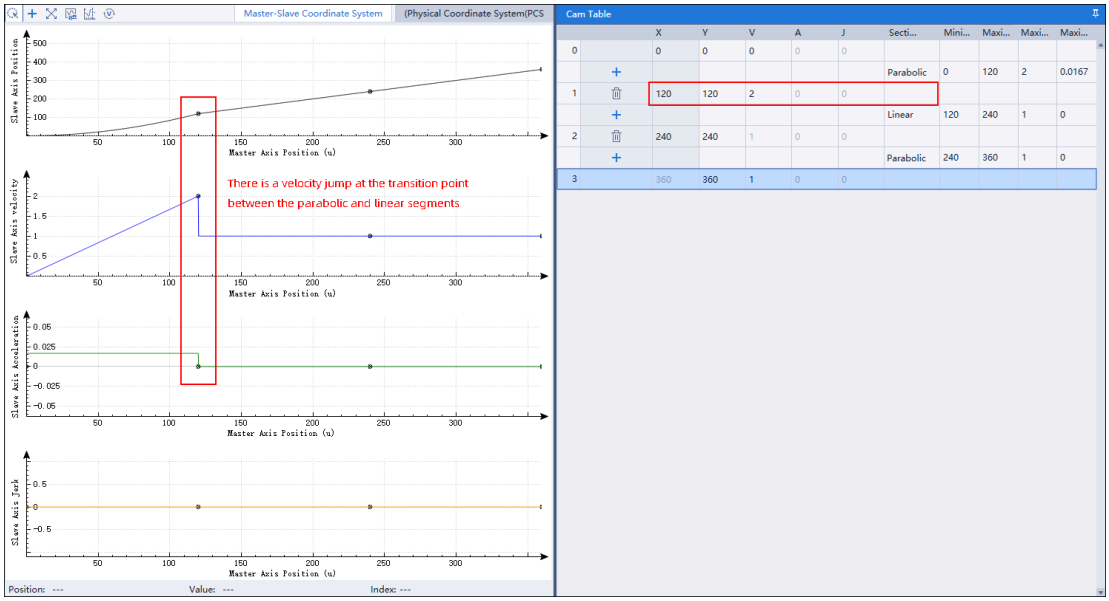
Profile configuration

Used to set the point shape, point size, and curve color for each diagram.

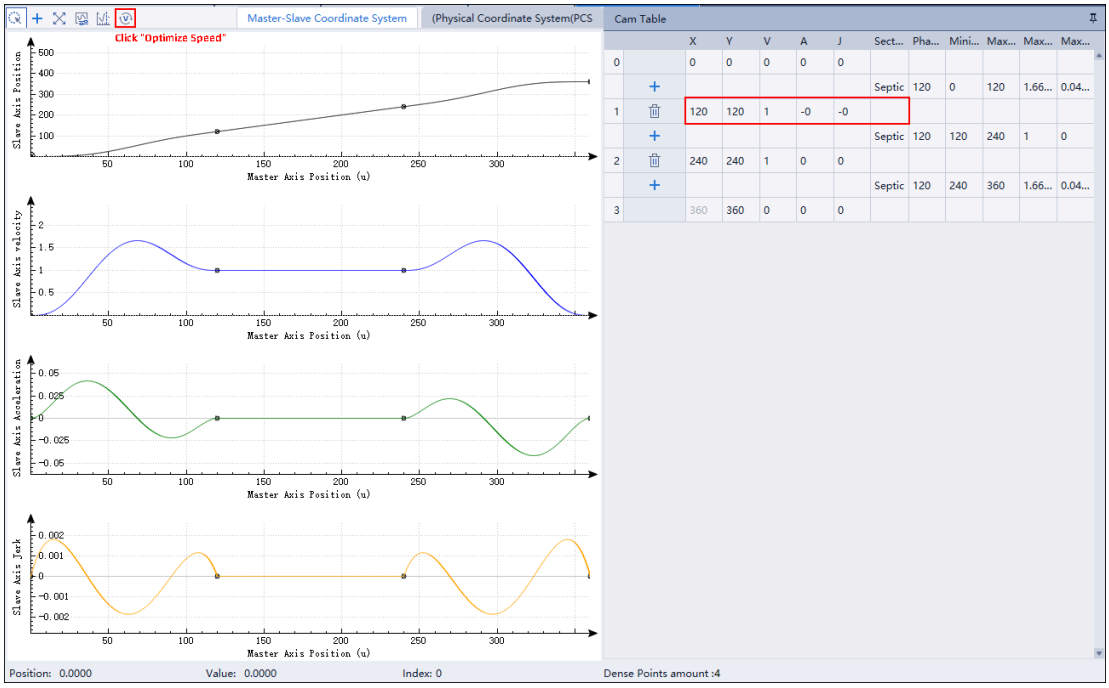


Optimize speed

Used to adjust the sudden changes in velocity during point transitions in the cam table.



If the system is sensitive to sudden changes in velocity, click the "Optimize speed" icon to optimize the motion parameters between points, such as velocity, acceleration, and jerk, eliminating sudden changes.



Currently, you can optimize sudden velocity changes due to the following two scenarios. The optimization is performed for the velocity of the entire cam table, rather than the velocity between two nodes.

- When a parabolic, cubic, quintic, or septic curve is followed by a linear curve, the velocity of the current node and subsequent nodes is adjusted to match the velocity of the linear curve.
- When a parabolic, cubic, quintic, or septic curve is followed by a cosine acceleration curve, a sine acceleration curve, or a modified trapezoidal acceleration curve, the velocity at the current node and the next node is set to 0.

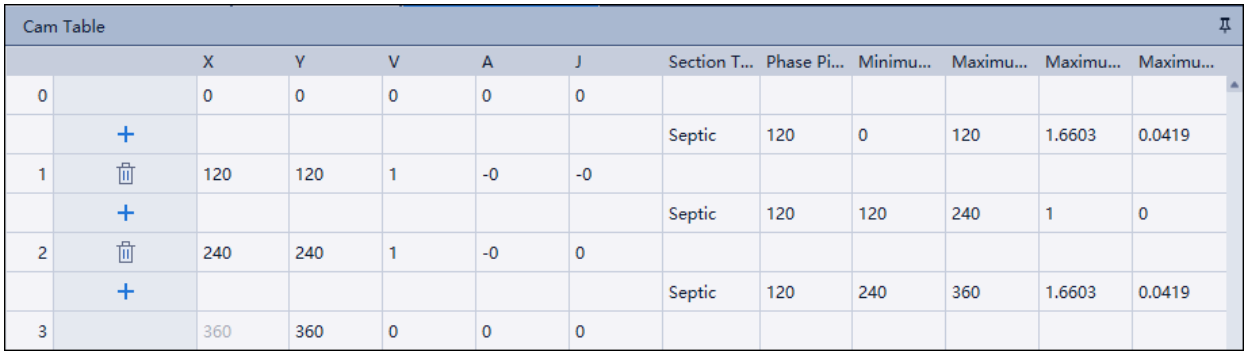
  The acceleration and jerk are adjusted based on the optimized velocity.

NOTE

The MC_GenerateCamTable instruction can also be used to optimize sudden velocity changes in user-created cam points. For details, see 18.3.3.4.10 MC_GenerateCamTable.

16.3.6.3 Cam Table

The right side of the cam chart is the cam table view, which contains more detailed information.



Cam table description:

- The starting point of the cam table is (0,0) and cannot be modified, but the velocity ratio at the starting point can be modified (for both the cam table and cam diagram).
- The velocity ratio of linear segments in the cam table cannot be modified. Additionally, for certain curve types, the speed ratio cannot be adjusted when they are connected, such as curve segments following cosine acceleration curves, modified T-type acceleration curves, and sine acceleration curves.
- The velocity ratios at the start and end points can be different (for both the cam table and cam diagram).
- The maximum length of the cam table is 65535, and the minimum pitch is 0.0001 (for both the cam table and cam diagram).
- The minimum precision of the cam table is to six decimal places.
- When changing the curve type of the cam table, parameters such as velocity, acceleration and jerk remain unchanged, unless the curve type is switched to linear.
- The supported curve types include linear, parabolic, cubic, quintic, septic, cosine acceleration, sine acceleration, and modified T-type curves.

ID	Curve Type	Note	Applicable Scenario
①	Linear curve	A linear acceleration curve typically refers to a trajectory where the following axis moves with constant acceleration along a straight line during the motion. This acceleration mode is characterized by a linear increase in velocity over time, hence its velocity-time diagram is a straight line with a positive slope.	Scenarios insensitive to impact
②	Parabolic curve	For motions with constant acceleration or constant deceleration, the displacement of the follower is proportional to the square of the cam rotation angle, and the displacement curve is a parabola, also known as the parabolic motion law. The velocity diagram of this motion law is continuous and will not cause rigid impact. However, there are sudden changes in acceleration at the start point, midpoint and end point, causing sudden changes in the inertial force of the push rod. These sudden changes are finite values and will cause flexible impacts in cam mechanisms.	Medium-speed and light-load scenarios

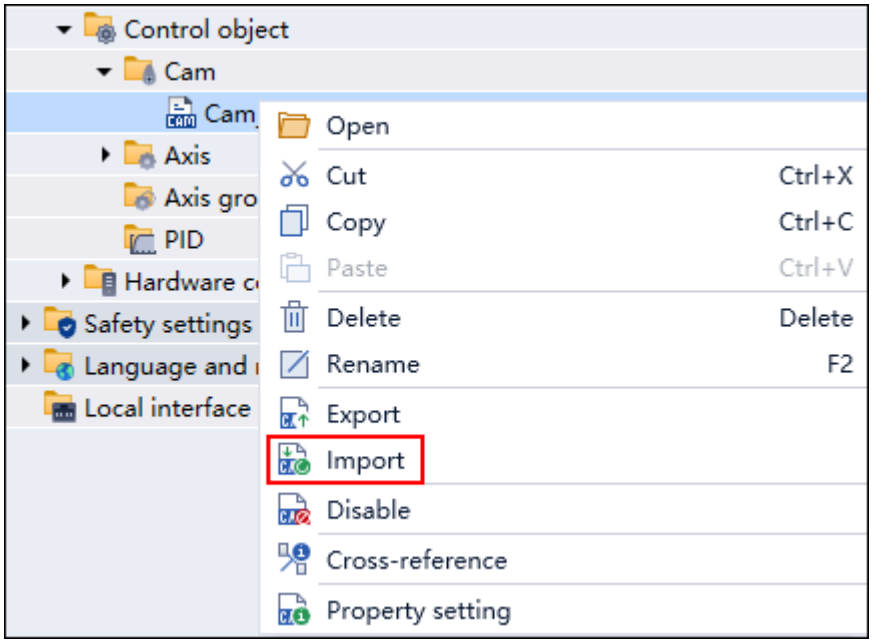
ID	Curve Type	Note	Applicable Scenario
③	Cubic curve	A cubic polynomial cam curve is a cam profile design method based on cubic polynomial functions, commonly used in mechanical processing and motion control fields. It constructs a smooth cam profile by fitting discrete data points, achieving precise motion control. In cubic curves, displacement and velocity are continuous, but acceleration is discontinuous.	High-speed and medium-load scenarios
④	Quintic curve	This motion law has neither rigid impact nor flexible impact, and features stable motions. This mode is applicable to high-speed scenarios.	High-speed and medium-load scenarios
⑤	Septic curve	The septic polynomial cam curve can meet the setting requirements of position, velocity, acceleration, and jerk at the start and end cam points. It achieves excellent motion performance in scenarios with strict requirements on rigid and flexible impacts.	High-speed and medium-load scenarios
⑥	Cosine acceleration curve	The cosine acceleration motion curve belongs to the trigonometric function-based motion law, also known as the simple harmonic cam curve. It has a flexible impact at the start and end of the motion. If the simple harmonic motion law is adopted for both the outstroke and return stroke periods, the non-impact condition can be satisfied The cosine acceleration curve can only meet the position requirements at the start and end cam points.	High-speed scenarios
⑦	Sine acceleration curve	The sine acceleration motion curve belongs to the trigonometric function-based motion law, also known as the cycloidal cam curve. It has no flexible impact at the start and end of the motion. The sine acceleration curve can only meet the position requirements at the start and end cam points.	High-speed scenarios
⑧	Modified T-type acceleration curve	The modified T-type acceleration cam curve has the advantages of smaller quasi-velocity and quasi-acceleration amplitude, and there is no flexible or rigid impact. This cam curve can only meet the position requirements at the start and end cam points.	Medium-high speed and heavy-load scenarios

- The cam table has "Add" and "Delete" icons. Click the "Add" icon to quickly insert an intermediate point between two points. Click the "Delete" icon to quickly delete a certain point.

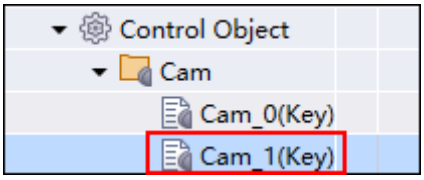
Importing a cam table

Normal import

1. Right-click "Cam" in the left project tree and select "Import cam table".

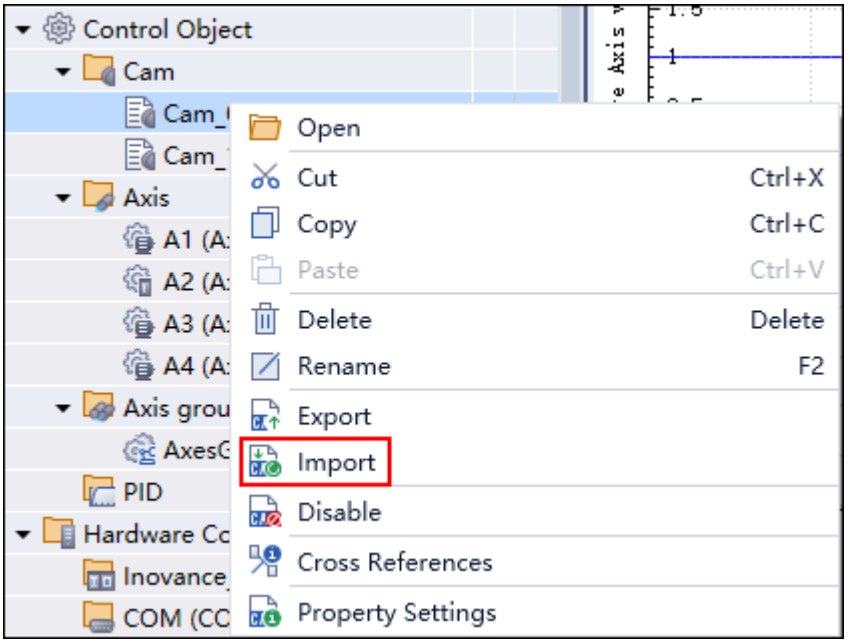


2. Select the file to import. After a successful import, a new cam table will be added under the "Cam" node.

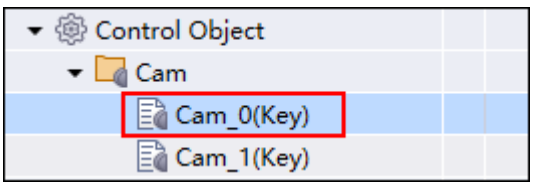


Overwrite import

1. Right-click an existing cam table in the left project tree and select "Import".



2. Select the file to import. After a successful import, the new cam table will overwrite the charts in the selected cam table.



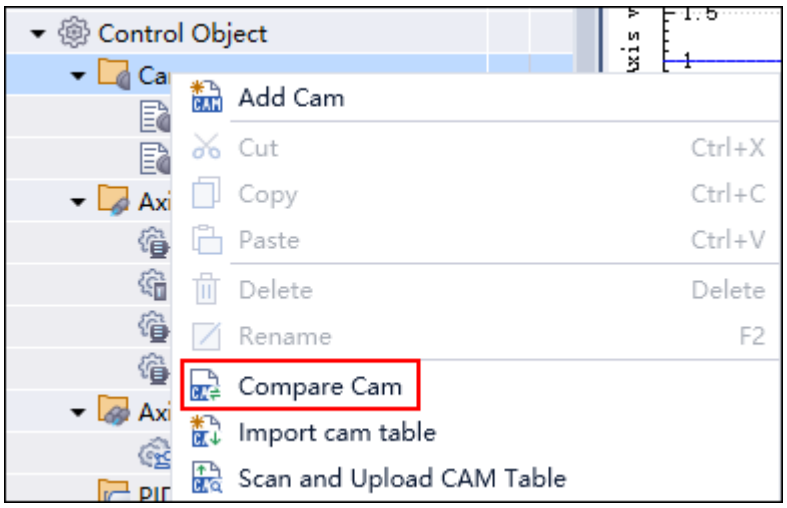
Comparing cam tables

Prerequisites

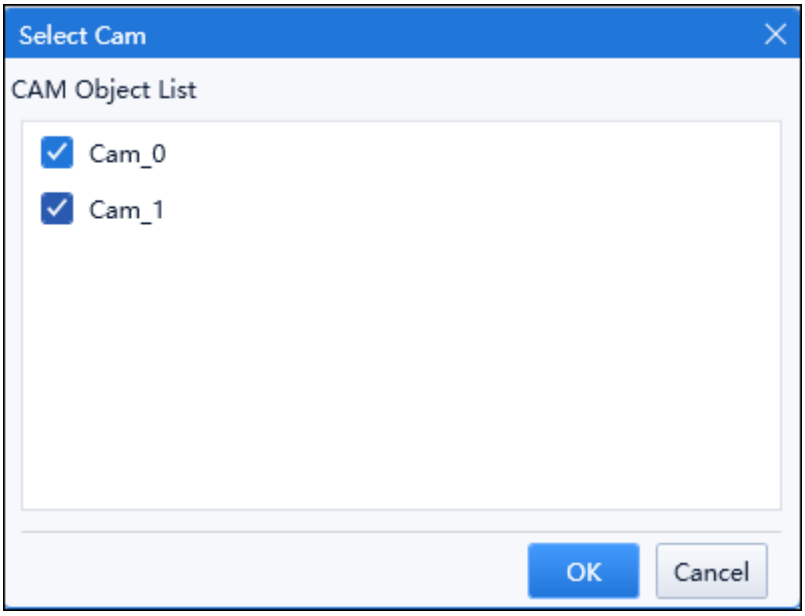
There are multiple cam tables.

Procedure

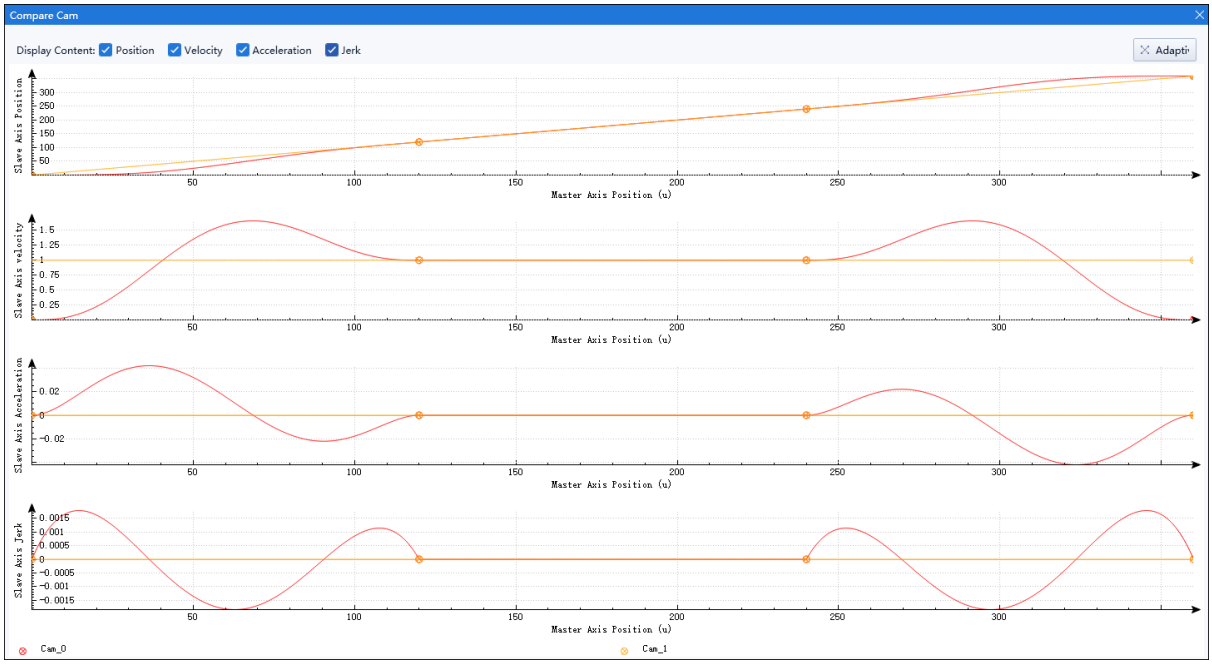
1. Right-click "Cam" in the left project tree and select "Compare cam".



2. In the pop-up "Select cam" window, check the cam tables to compare, and click "OK".



3. The comparison results of different cam tables are shown in the following figure.



Uploading a cam table

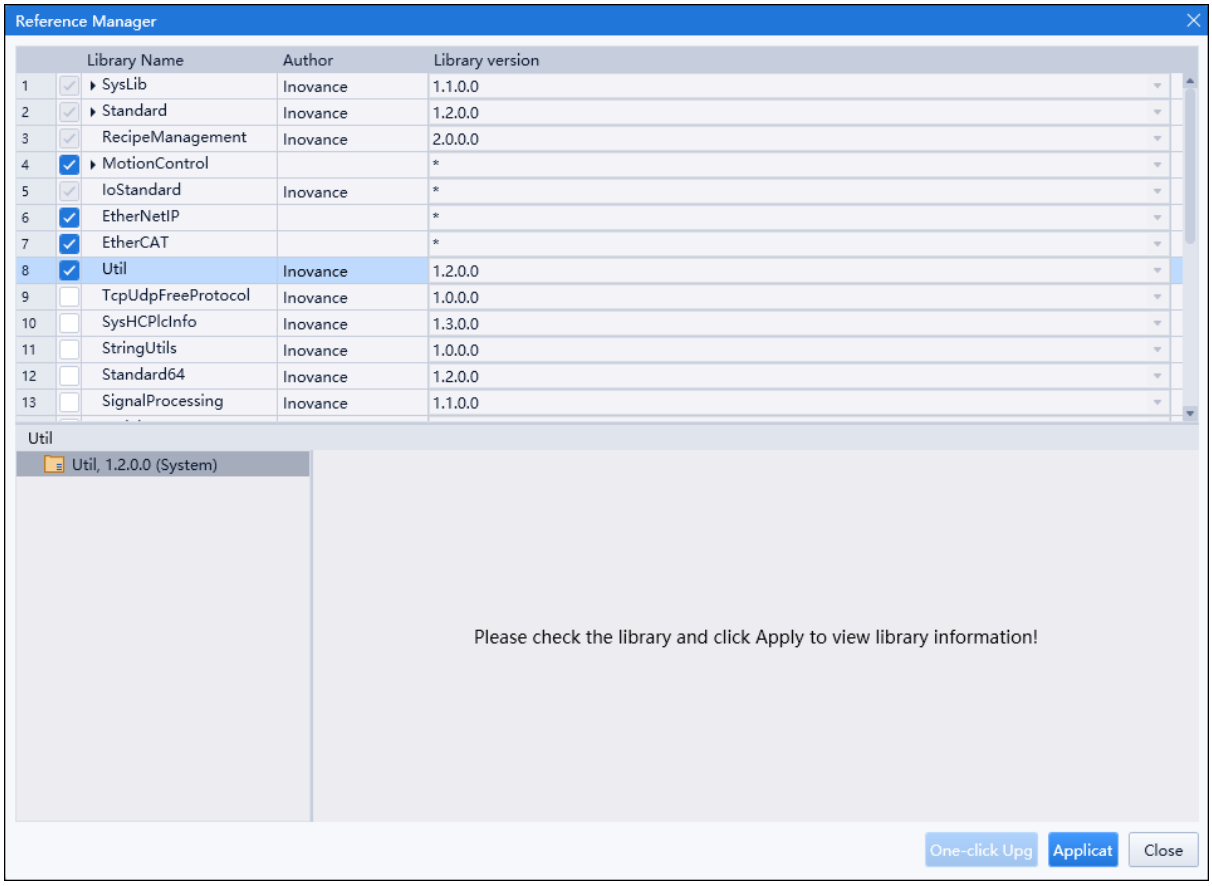
The MC_WriteCam instruction can be used to write custom cam tables or system-created cam tables into the controller. After the device goes offline, the cam tables in the controller can be uploaded to the "Cam" node.

Prerequisites

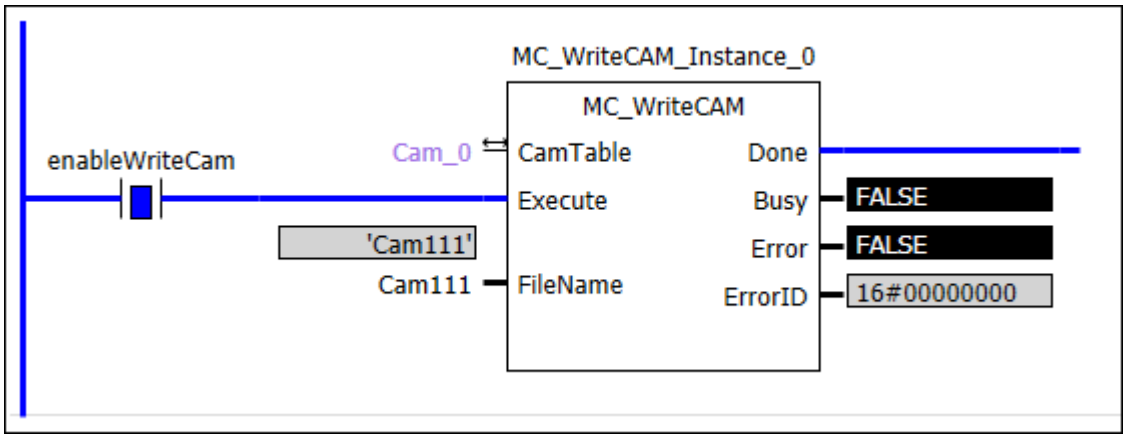
The relevant library references have been added.

Procedure

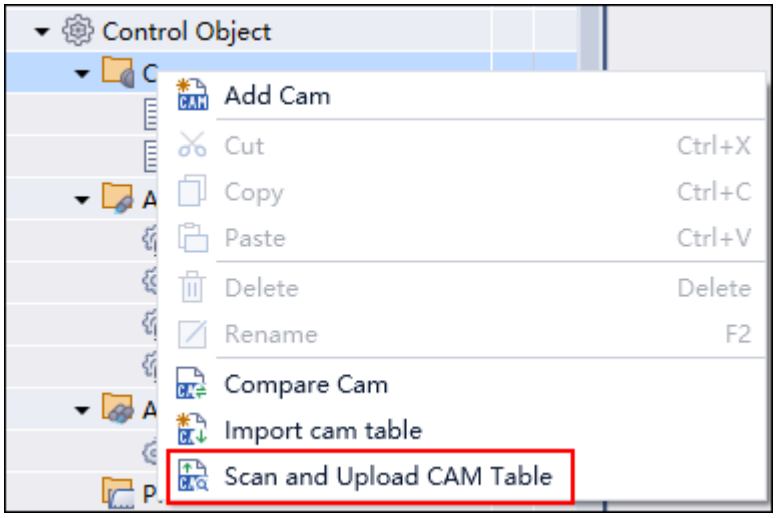
1. Double-click "Library reference" in the left project tree to open the "Reference manager" window.
2. Check the "MotionControl" library.



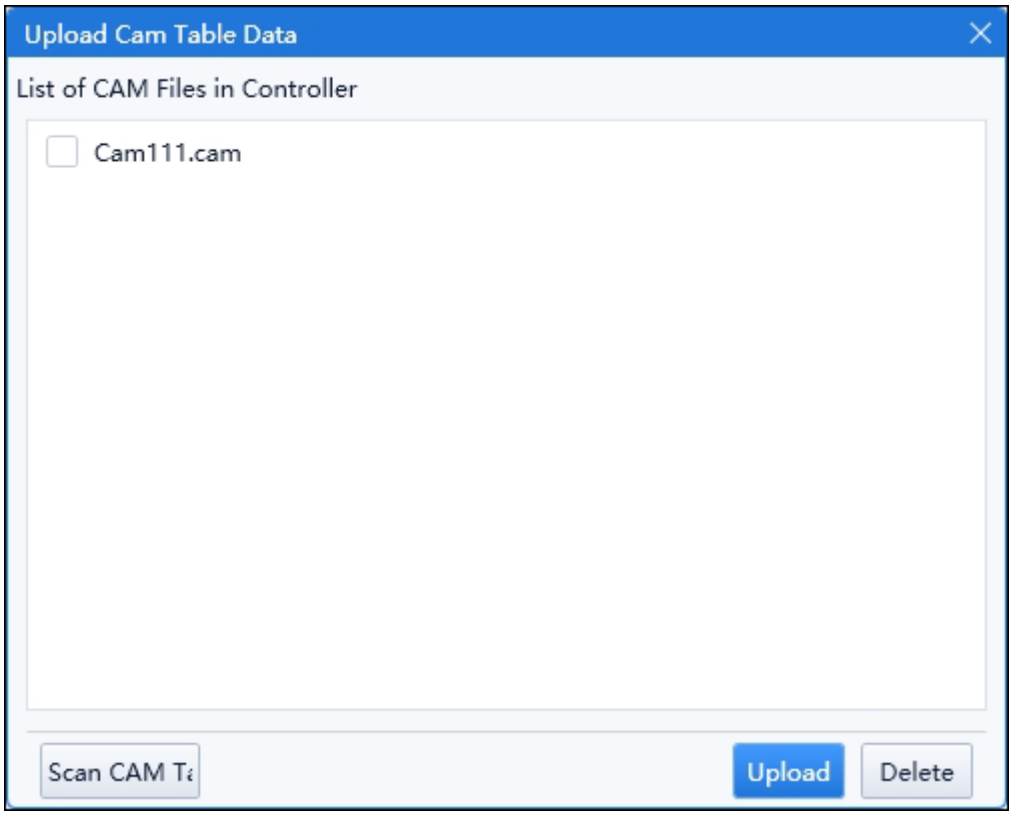
3. Select the upload and download functions for the cam table in the toolbox.



4. After a successful execution, click the "Offline" icon, right-click "Cam" in the project tree, and select "Scan and upload cam table".

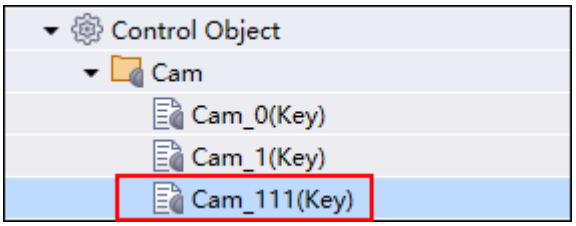


5. The cam files in the controller will be displayed in the pop-up window. Select a cam file and click "Upload".



6. A success prompt for the cam table upload will appear.

7. The uploaded cam table will be added under the "Cam" node in the project tree.



17 Drive Settings

17.1 EtherCAT Configuration

17.1.1 EOE Connection

Before commissioning the servo and performing operations, an EOE connection must be established first through the following steps:

1. Perform network settings and configure the computer IP address. For details, see 17.1.2 Network Settings.
2. Create a master station and add an EtherCAT master node. For details, see 17.1.3 Master Station Creation and Configuration.
3. Add the servo drive device. For details, see 17.1.4 Adding a Device.
4. Set the EOE parameters for the servo drive. For details, see 17.1.7.6 EoE Settings.
5. Switch to online mode.
6. Connect the servo drive. For details, see 17.1.8 Device Connection.

17.1.2 Network Settings

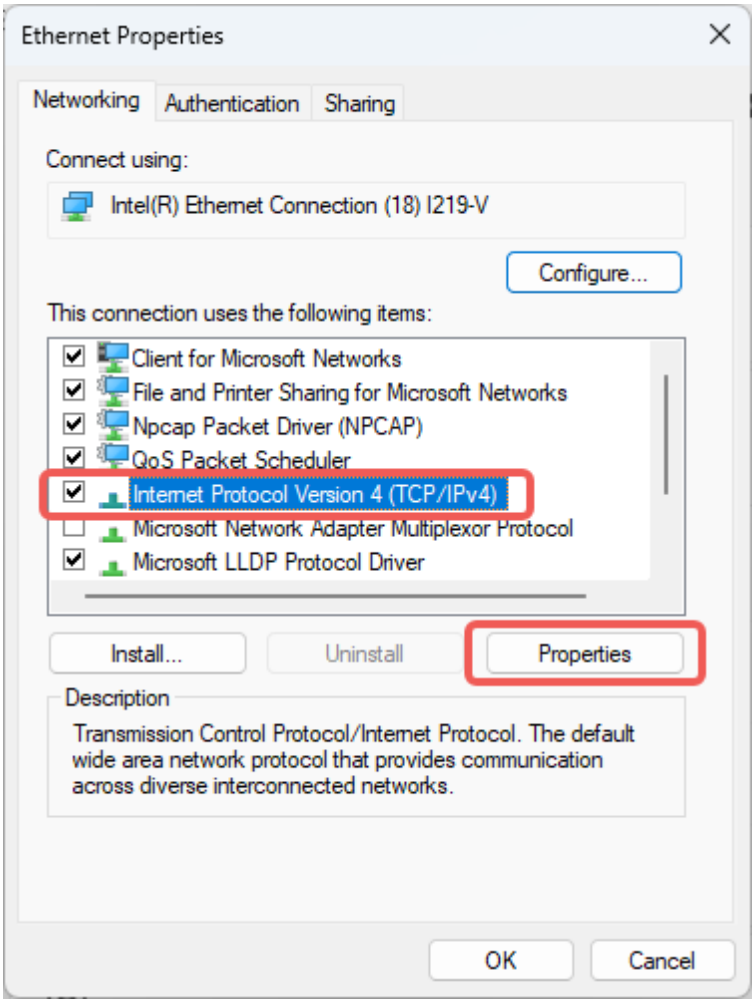
When communicating with the drive using the EOE method, the IP address of the local computer shall be modified to ensure that the computer and the device are in the same subnet.



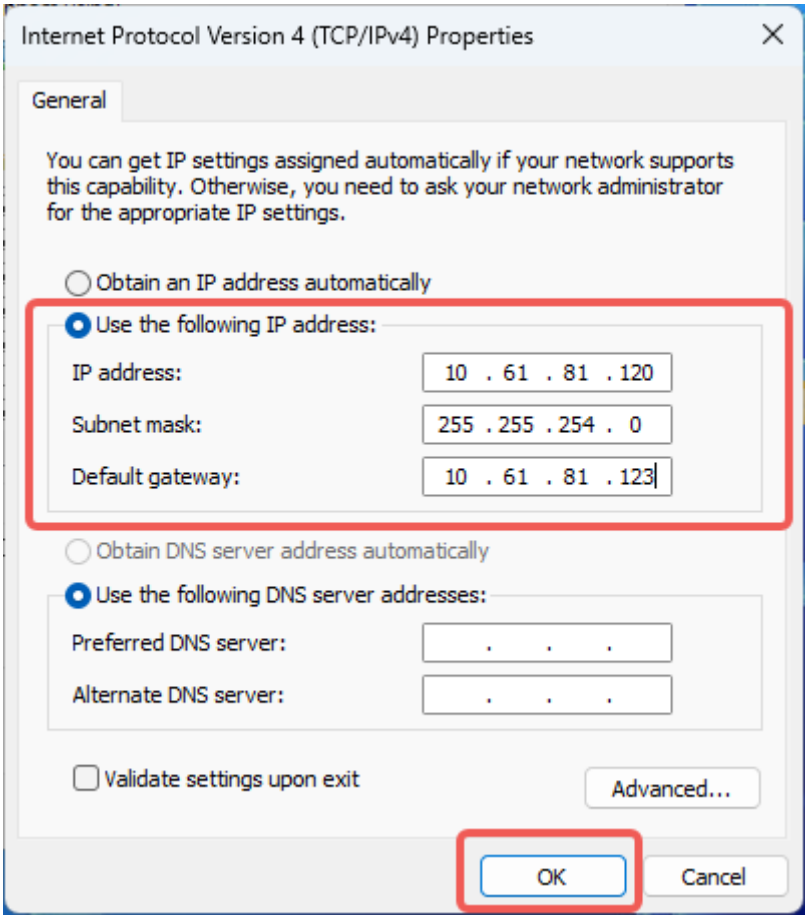
Manual network configuration is required for EVO500 series PLCs with a device version lower than V1.4 or a firmware version lower than 1.4.0.0. Other models or newer versions have achieved automatic adaptation and do not require additional operations.

The method for setting the computer's IP address is as follows:

1. In the "Ethernet properties -> Networking" window, select "Internet Protocol Version 4 (TCP/IPv4)" and click "Properties".



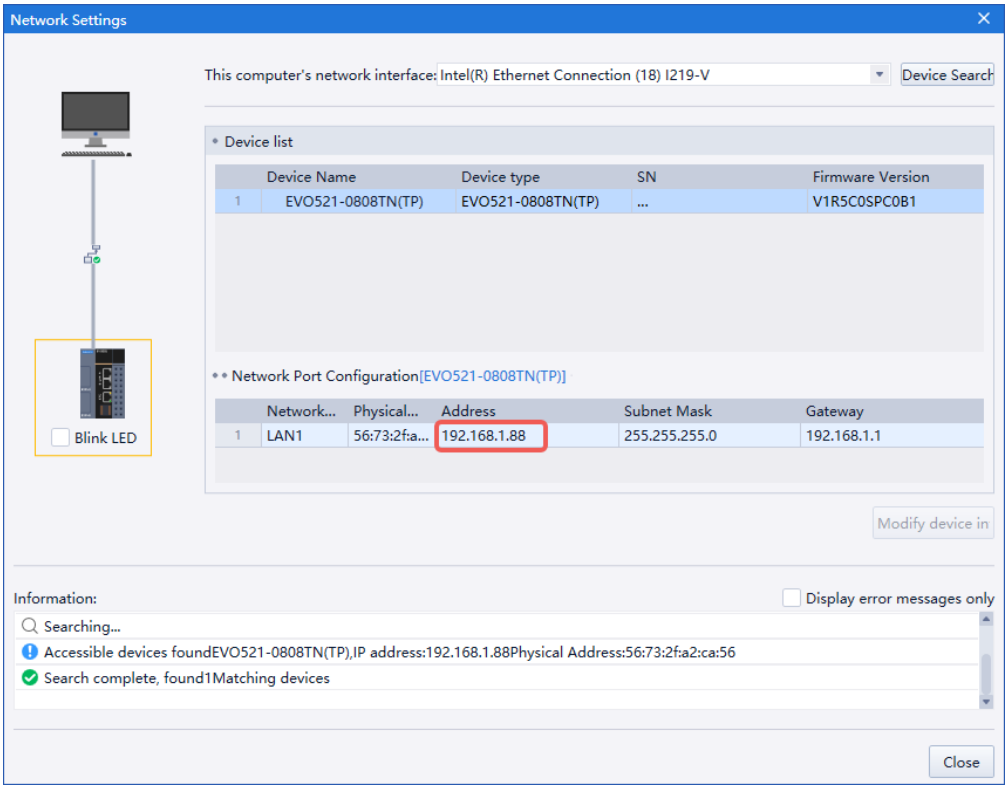
2. Select "Use the following IP address", then enter the correct IP address, subnet mask, and default gateway, and click "OK" to complete the settings.



NOTE

Set the default gateway to the IP address of the PLC, which can be obtained through software, as follows:

In the menu bar, select "Online > Network setting". In the pop-up window, click "Device search" to obtain the device IP.



17.1.3 Master Creation and Configuration

This section introduces the precautions for master station creation and configuration:

- Before creating the master, add a PLC device; otherwise, the drive cannot be added or configured.
- Before operating the device, switch the PLC to offline mode; otherwise, operations fail. When switching the device to offline mode, the "Online" icon in the toolbar becomes highlighted, and the "Offline" icon becomes grayed out.
- The EtherCAT configuration mainly involves device operations. For details, see 17.1.4 Adding a Device.
- For details about disabling/enabling devices, see 17.1.5 Disabling/Enabling a Device.

17.1.4 Adding a Device

During EtherCAT configuration, right-click the "EtherCAT (EtherCAT)" node under "Hardware configuration" of the PLC node to add an EtherCAT master; then add slaves (drive devices) under the master through the "Add device" and "Scan for devices" functions.

17.1.4.1 Adding a Device

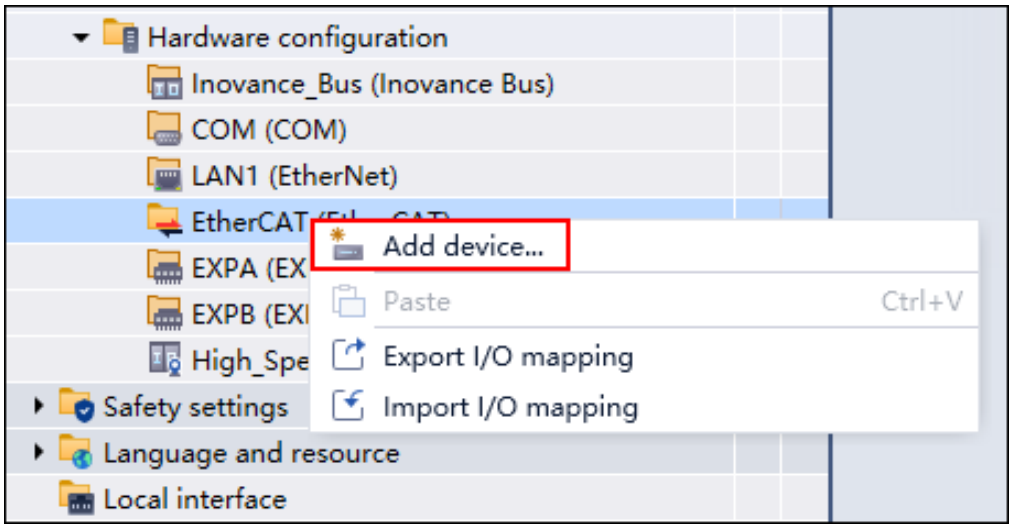
The "Add device" function can be used to add EtherCAT master and slaves.

Prerequisites

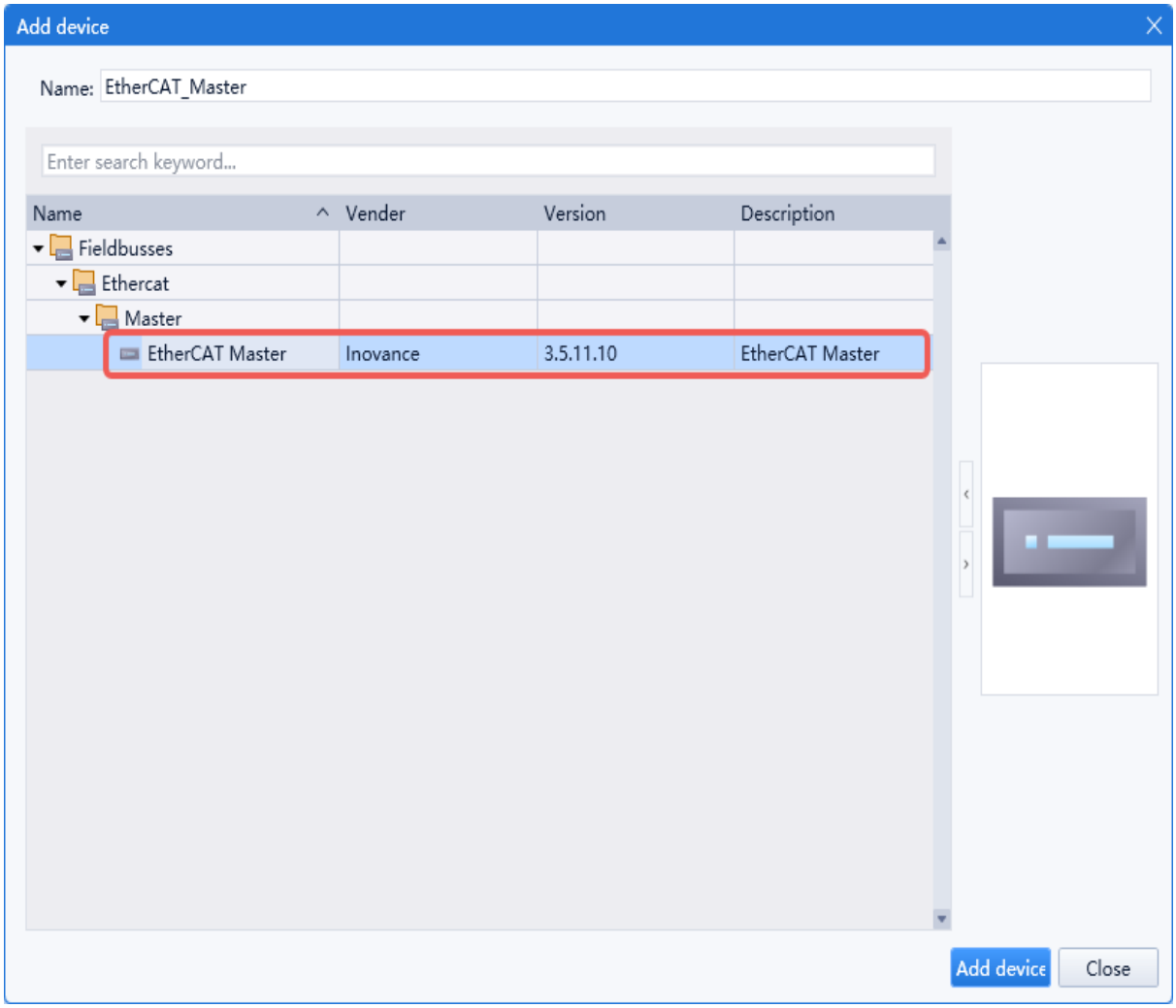
- A PLC device has been added.
- The drive device has been correctly connected.
- The PLC device is in an offline status.

Procedure

1. Right-click "Hardware configuration > EtherCAT (EtherCAT)" in the project tree, and select "Add device".

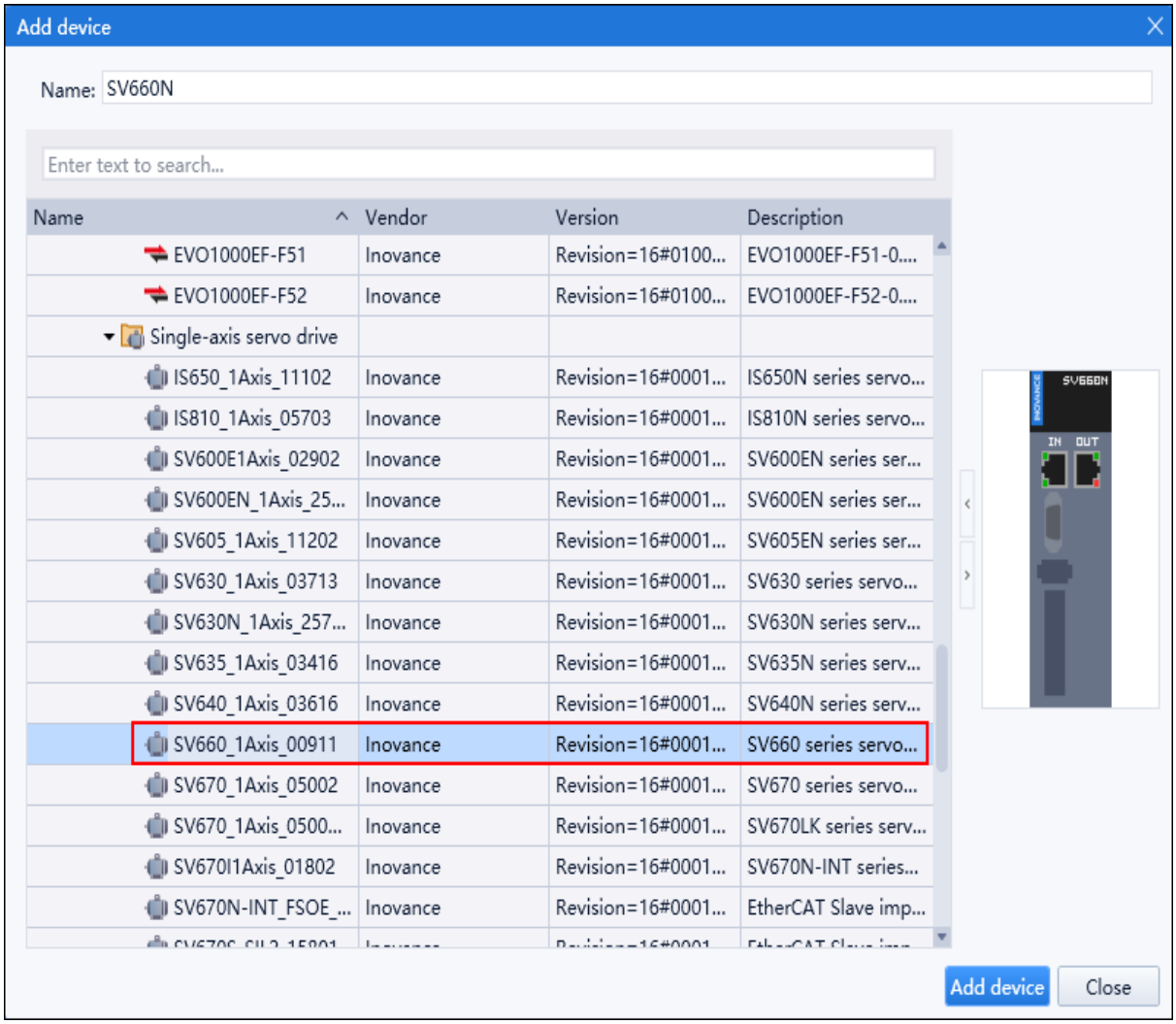


2. In the pop-up window, double-click "EtherCAT Master", or select it and click "Add device".



3. After adding, right-click "EtherCAT (EtherCAT) > EtherCAT_Master (EtherCAT Master)" in the project tree, and select "Add device".

4. In the pop-up "Add device" interface, select the actual drive model that is connected, then click "Add device" or double-click the device model to add the drive.



NOTE

Only one master can be added under the "EtherCAT (EtherCAT)" node; multiple servo drive devices can be added under "EtherCAT_Master (EtherCAT Master)".

17.1.4.2 Scanning for Devices

The "Scan for devices" function can be used to add servo drives.

Prerequisites

- The drive device has been correctly connected.
- The "EtherCAT_Master (EtherCAT Master)" node has been created.
- The PLC has been compiled and downloaded.

Procedure

1. Right-click the "EtherCAT_Master (EtherCAT Master)" node in the project tree and select "Scan for devices".
2. In the pop-up window, click "Device scanning" and wait for the scan results.
3. Select the scanned device, click "Copy all devices to project" to add the selected device.

17.1.5 Disabling/Enabling a Device

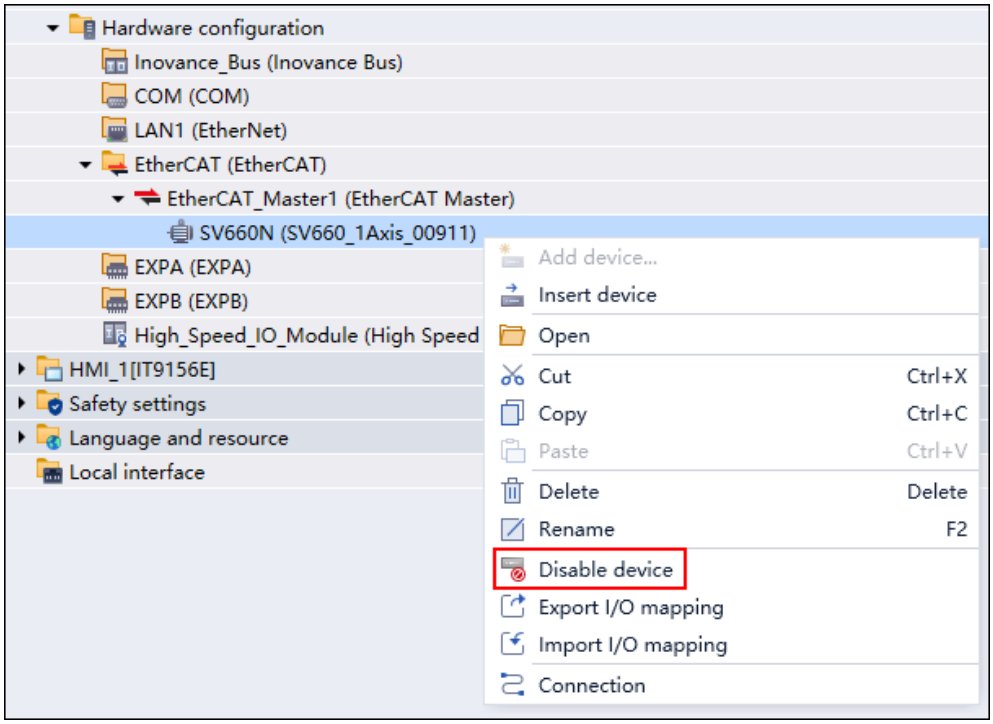
When "Virtual axis mode" is selected for a motion control axis, even if you disable the EtherCAT master (along with all the master and slave stations under this bus), you can still control the motion control axis through the program. When the EtherCAT master is disabled, all EtherCAT slaves are disabled and the bus servo axes associated with the slaves stop running. Similarly, when the EtherCAT master is enabled, all EtherCAT slaves will be enabled.

Prerequisites

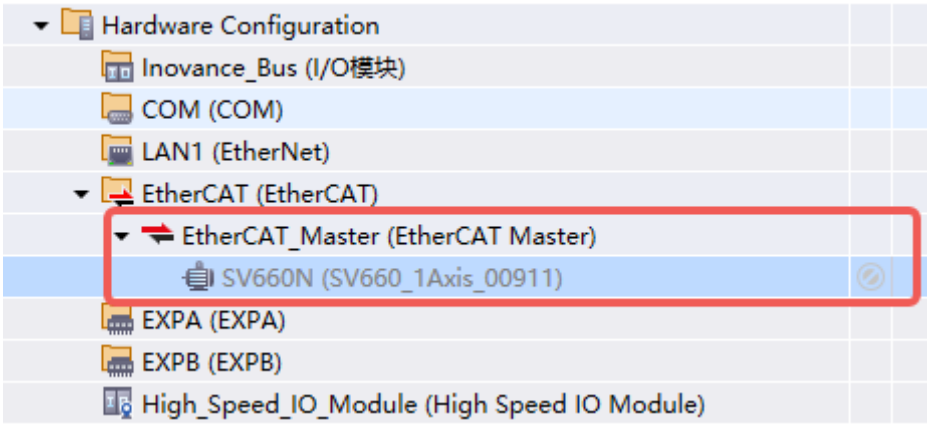
Devices have been added to the master.

Procedure

Right-click the device node in the project tree and select "Disable device" to disable the selected device and all devices under it (a single device can also be enabled or disabled using the same method).



For example, the effect of disabling "EtherCAT_Master" is shown in the following figure:



A disabled a device cannot be connected. To enable a disabled device, right-click the disabled device and select "Enable device".

17.1.6 Deleting a Device

The operation to delete a device is cascaded, meaning that the selected device and all its sub-devices will be deleted.

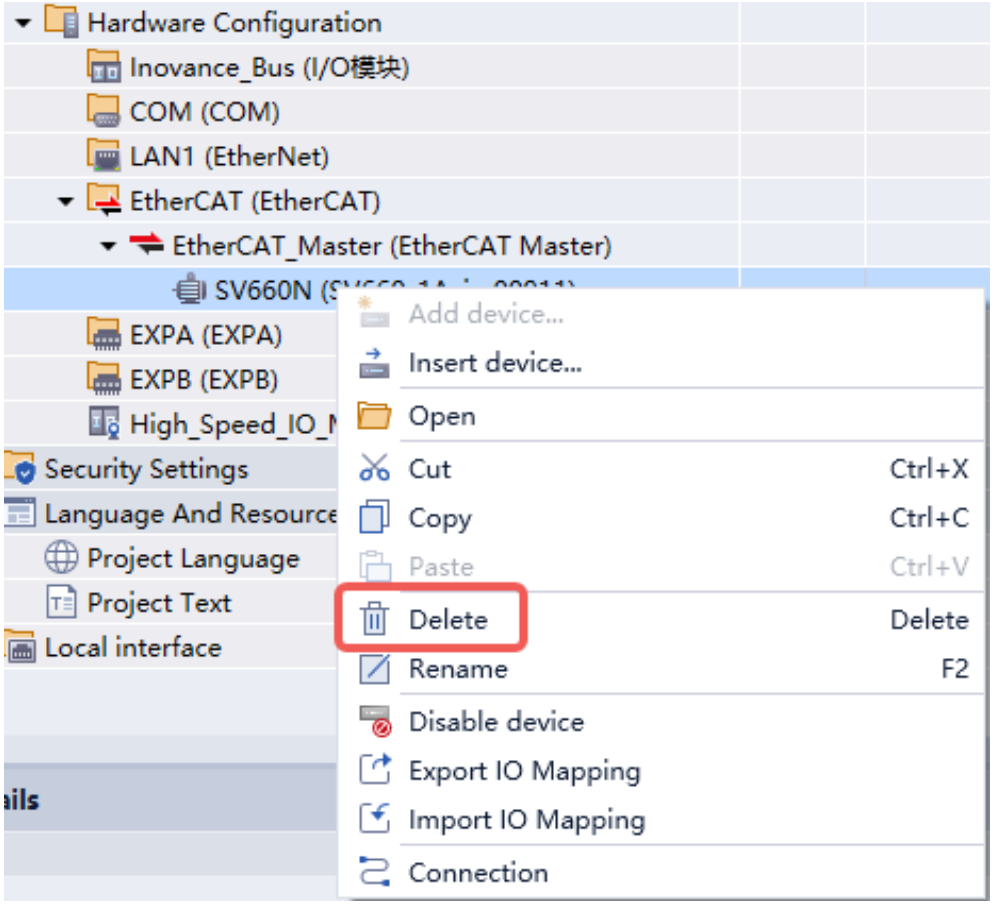
Prerequisites

The PLC has been switched to offline status.

Procedure

Right-click the added device node, select "Delete", and then click "OK" in the confirmation box.

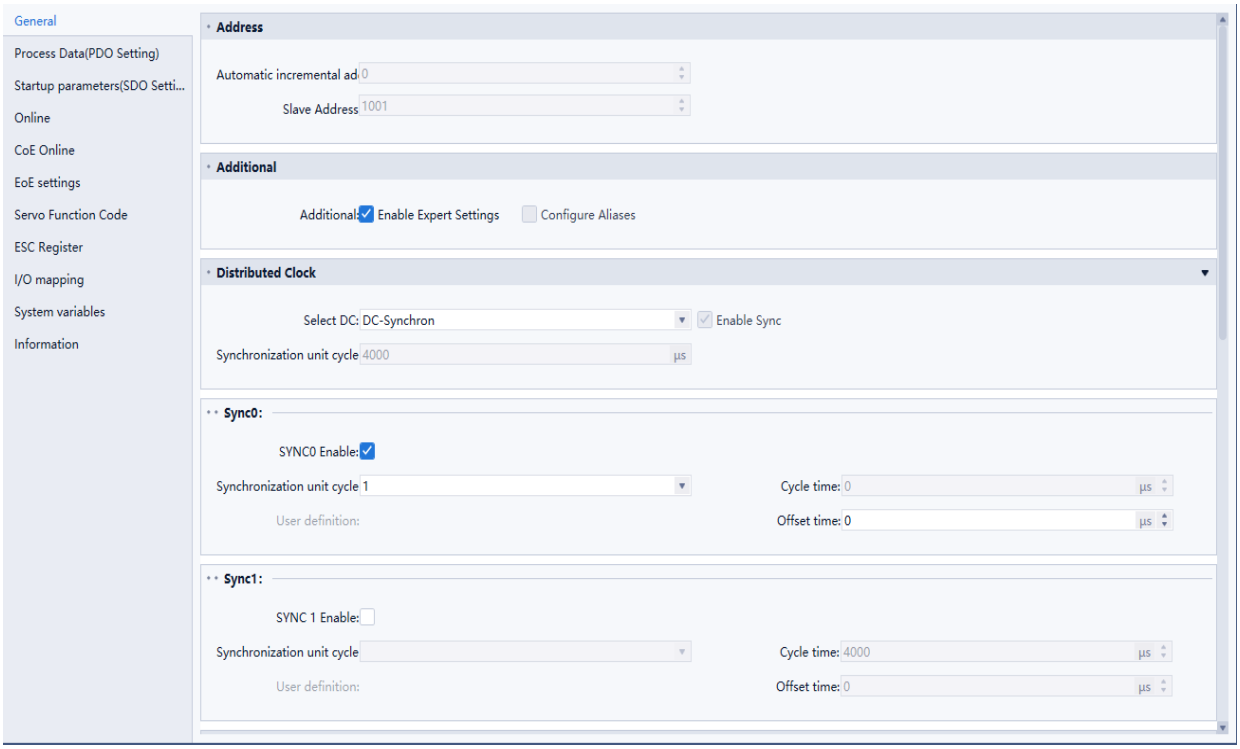
Servo drive devices, EtherCAT masters, or LC devices can all be deleted.



17.1.7 Device Configuration

17.1.7.1 General Settings

Double-click a slave node (servo device) to open the slave configuration window. The "General" interface of the EtherCAT slave provides basic configuration information for the slave, as shown in the figure below.



Address

If the "Auto-configure master/slave" option in the master device editor is not checked, the following options can be configured:

- Automatic incremental address: The automatic incremental address (16-bit) is determined by the physical topology position of the slave in the network. This address is only used when the EtherCAT master starts up. It assigns EtherCAT slave addresses to slaves at corresponding physical topology positions through sequential addressing.

  During sequential addressing, the auto-increment address of the slave is determined by its connection position in the physical topology network, represented by a negative number. Sequential addressing sends data frames containing sub-messages. As each data frame passes through a slave device, the auto-increment address in the sub-message automatically increments by 1. When a physical slave receives the data frame, it addresses its own message by checking whether the auto-increment address in the sub-message is 0. This mechanism of updating internal address variables in the slave through data frames is called "sequential addressing" or "auto-increment addressing".

- Slave address: The configuration address (nominal address) of the slave, assigned by the master during startup. This address is independent of the actual position in the network, and the address of the slave is unrelated to its connection order within the segment. After disabling a slave device in hardware configuration, the address of the slave device is set to 0, and it is reassigned according to the current specifications when the slave is re-enabled.

Additional settings

- Enable expert settings: If this option is checked, you can configure distributed clocks, startup checks, timeouts, DC cycle unit control, watchdogs, and other functions.
- Configure aliases: This reflects the status of the "Enable alias" option in the general settings of the EtherCAT master, and cannot be modified here.

Distributed clock

- Enable sync: If this option is checked, it indicates that the "Distributed clock" function is used.
- Synchronization unit cycle (μs): If the "Distributed clock" function is enabled, the synchronization unit cycle value will default to the same as the EtherCAT master's cycle time.
- Select DC: This drop-down menu provides all settings related to distributed clocks in the device description file, generally including free operation, SM event synchronization, DC synchronization, and so on. The setting items are described in the table below:

ID	Item	Description
1	Free operation mode	In this mode, the local control cycle is generated by a local timer interrupt. The cycle time can be set by the master. This is an optional function for the slave.
2	SM event synchronization	A local cycle is triggered when a data input or output event occurs. The master can write the transmission cycle of the process data frame to the slave. The slave can then check whether the cycle time is supported or perform local optimization of the cycle time. The slave can choose to support this function. Synchronization typically occurs with the data output event. If a slave has only input data, synchronization occurs with the data input event.
3	DC synchronization	A local cycle is triggered by the SYNC event. The master must complete the transmission of the data frame before the SYNC event. Therefore, the master clock must also be synchronized to the reference clock.
4	Sync0 and Sync1	Slave synchronization signals 0 and 1. The distributed clock control unit of the slave (an internal function of the EtherCAT-specific communication chip) can generate two synchronization signals, Sync0 and Sync1. These signals provide interrupt flags to the application layer of the slave or directly trigger the update of output data.
5	Sync0 enablement and Sync1 enablement	Check these options to enable the Sync0 and Sync1 synchronization signals. • Synchronization unit cycle: If this option is checked, slave synchronization cycle time = master cycle time × selected coefficient. • The "Cycle time (μs)" item indicates the current set cycle time of the slave.
6	User-defined DC mode	If this option is selected in the "Select DC" drop-down menu, user-defined synchronization cycle time in microseconds can be entered in the "Offset time" fields for Sync0 and Sync1.

Diagnostics

Visible only in online mode.

Current status: Displays the current status of the slave's communication state machine. Possible values include: Initialization, Pre-operational, Safe-operational, Operational, Boot (not supported by Inovance

servo). If the current status is operational, it indicates that the slave configuration has been correctly completed.

Option

Option: If a slave device is defined as "optional", no fault messages will be generated to prevent the device from not being present in the bus system. To enable this option, a station address must be stored in the slave device, so the "station alias" address must be defined and written into the EEPROM. This option is only valid when the "Auto-configure master/slave" option is checked in the EtherCAT master general settings and the EtherCAT slave supports this function.

Start check

By default, the system automatically checks the identity of the supplier and the product ID at startup based on the current configuration. You can also check the "Check version number" option to check the specified version. If a mismatch is detected, the bus will stop running and no further actions will be performed. This setting is used to prevent the download of incorrect configurations.

Timeout

By default, the following operations are not defined as timeouts. To observe whether these operations exceed a specific time, you can set them here.

- SDO access: Sets the timeout time for SDO (Service Data Object) access from the EtherCAT master protocol to the slave.
- I -> P: Sets the timeout time for the transition of the slave communication state machine from "Initialization" to "Pre-operational".
- P -> S / S-> O: Sets the timeout time for the transition of the slave communication state machine from "Pre-operational" to "Safe-operational", or from "Safe-operational" to "Operational".
- Detect port -> Open port: After the slave is connected, sets the timeout time for detecting and opening the port.

DC cycle unit control

This option is assigned to the local microprocessor to select the options defined by the distributed clock function. The control function is implemented in register 0x980 of the EtherCAT slave, with optional settings being: Periodic unit, Latch 0, and Latch 1.

Watchdog

- Watchdog multiplier: Sets the multiplier of the watchdog timer to determine its minimum timing increment. The default value is 2498. The minimum timing increment of the watchdog timer is 100 μs.
- PDI watchdog: After enabling the PDI watchdog, if the PDI (Physical Device Interface) communication time of the EtherCAT slave exceeds the set value, the watchdog will be triggered.
- PD watchdog: Sets the PD watchdog.

Slave alias

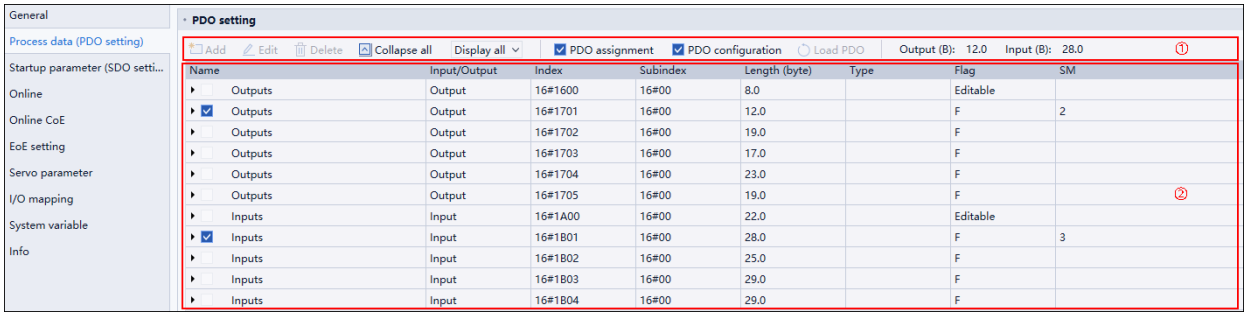
The settings of slave alias are only valid when "Option" is checked and the slave device supports alias addresses (defined by the device description file). If the corresponding alias address of the slave is configured, there is no need to change the configuration in the user program when adjusting its position in

the physical topology network, and the slave can still operate normally. Note: After modifying the alias address of the slave, the user program must be re-downloaded for the alias to take effect. Additionally, some slaves must be powered off and restarted for the alias to take effect. For details, see the user guides of relevant slave devices.

- Disable: If this option is checked, the slave will not detect the alias address.
- Configure site alias (ADO 0x0012): When the slave device supports alias addresses and "Option" is checked, writing the alias address is supported only in online operation mode.
- Write EEPROM: This command is visible only in online mode. It allows the defined address to be written to the EEPROM of the slave. If the slave does not support the written address, this command is invalid and the slave cannot operate in site alias form.
- Actual address: This field is visible only in online mode. The actual address of the slave is displayed, which can be used to check whether the "Write EEPROM" command is successful.
- Display device identification (ADO 0x0134): Reserved.
- Data word (2 bytes): Reserved.

17.1.7.2 Process Data (PDO Settings)

In actual automated control systems, there are normally two forms of data exchange between application programs: time-critical and non-time-critical. Time-critical indicates that specific actions must be completed within a designated time window; if not, it may lead to control failure. The process of periodically sending time-critical data is called periodic process data communication. Non-time-critical data can be sent non-periodically using MailBox data communication in EtherCAT.



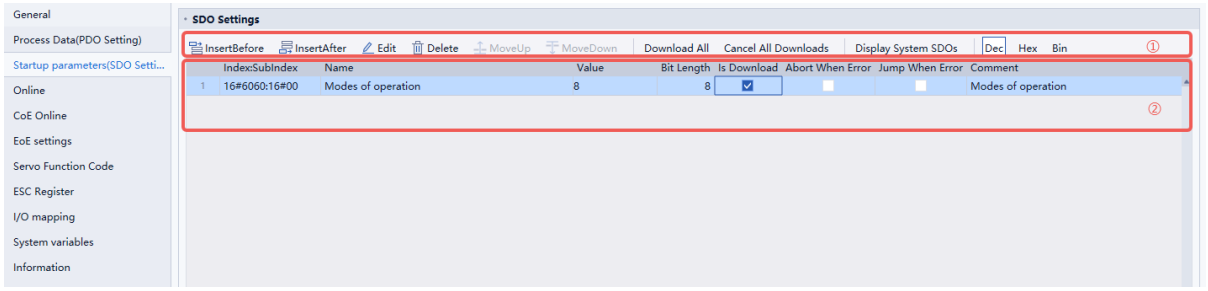
No.	Function	Description
①	Toolbar	Supports adding, editing, deleting, and filtering PDO entries.
②	PDO display area	Displays detailed information about PDO.

- Adding: Adds PDO items based on the properties of the current PDO group (editable properties only). Users can choose to add one set or multiple sets of data.
- Edit: Edits the selected PDO item based on the properties of the current PDO group (editable properties only).
- Delete: Deletes the selected PDO item based on the properties of the current PDO group (editable properties only).
- Collapse all: Collapses all entries in the PDO display area with one click.
- Filter box: Filters the PDO display area to display all, display output PDO entries, or display input PDO entries.

- PDO allocation/PDO configuration: In the configuration of the GR10_4ADE module, PDO allocation and configuration refer to assigning PDO objects to the corresponding input/output channels and setting PDO parameters to ensure correct communication between the module and the CANopen network.
- Load PDO: When the slave is in operation, the current PDO group data of the slave can be uploaded to the backend.

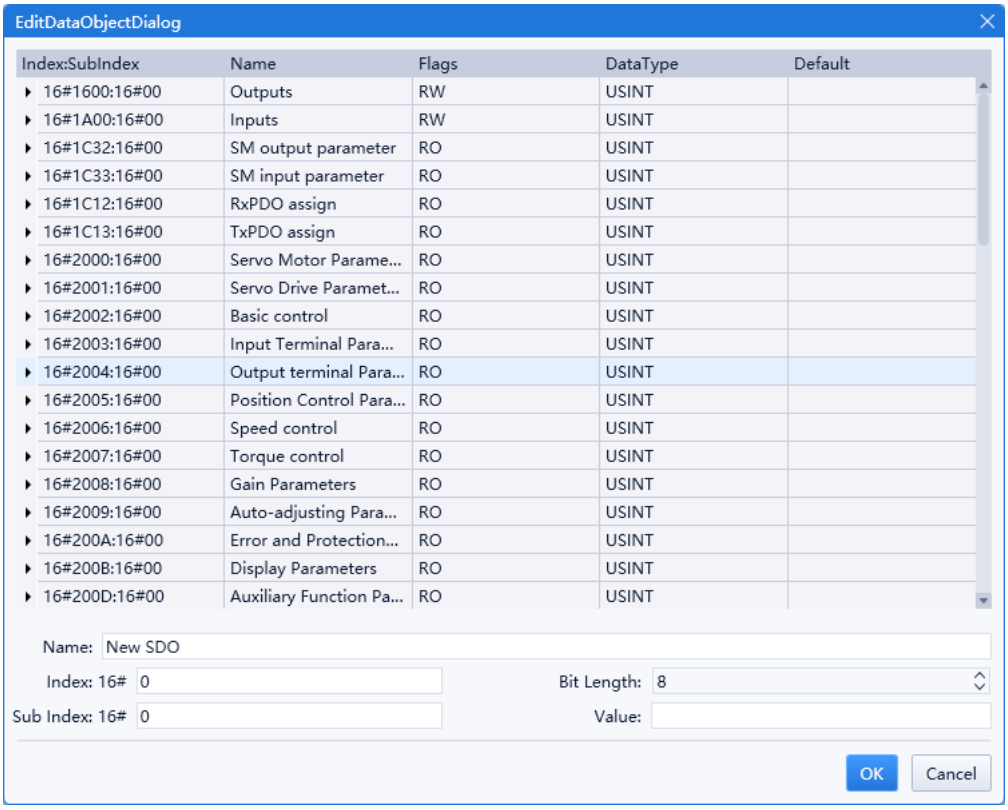
17.1.7.3 Startup Parameter (SDO Settings)

Startup parameters can be transmitted to the slave through SDO (Service Data Object) when the system starts. They include basic configuration parameters required for the slave startup, as shown in the following figure.



No.	Function	Description
①	Toolbar	Supports inserting, editing, deleting, downloading, canceling download, and setting display mode for SDO entries.
②	SDO entry display area	Displays detailed information of SDO entries.

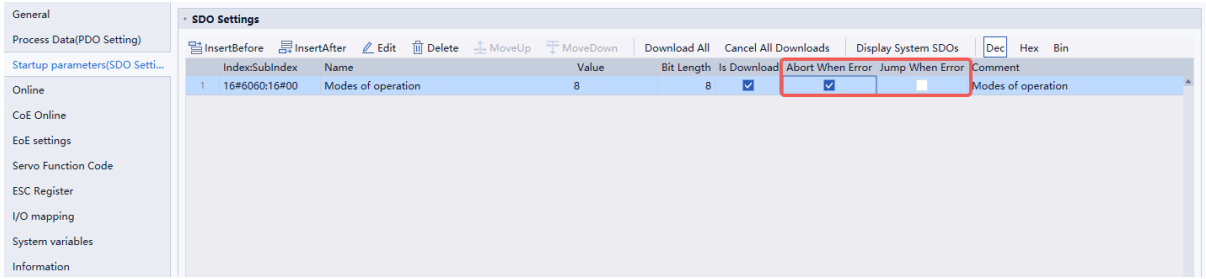
- Insert before: Adds an SDO entry to the startup parameter list before the currently selected entry. The object dictionary selection box that pops up is shown below. After selecting the SDO to add, you can modify its name, index, sub-index, bit length, and value in the edit bars below, then click "OK" to create a new startup parameter.



- Insert after: Adds an SDO entry to the startup parameter list after the currently selected entry.
- Edit: Edits the parameters of the currently selected entry. Read-only parameters cannot be edited, such as system parameters.
- Delete: Deletes the currently selected entry.
- Move up/Move down: The order of the SDO list (from top to bottom) represents the order in which startup parameters are transmitted to the module. The "Move up" and "Move down" buttons can change the transmission order.
- Download all (SDO)/Cancel all downloads (SDO): Click "Cancel all downloads" to deselect the download property of all entries. Click "Download all" to select the download property of all entries. You can also select the download property of some entries for partial download.
- Display system parameters: Displays the parameters added to the SDO after PDO allocation and PDO configuration are checked, for comparison only.
- Value display mode: The values in the "Value" column of the SDO entry display area can be displayed in decimal, hexadecimal, or binary.

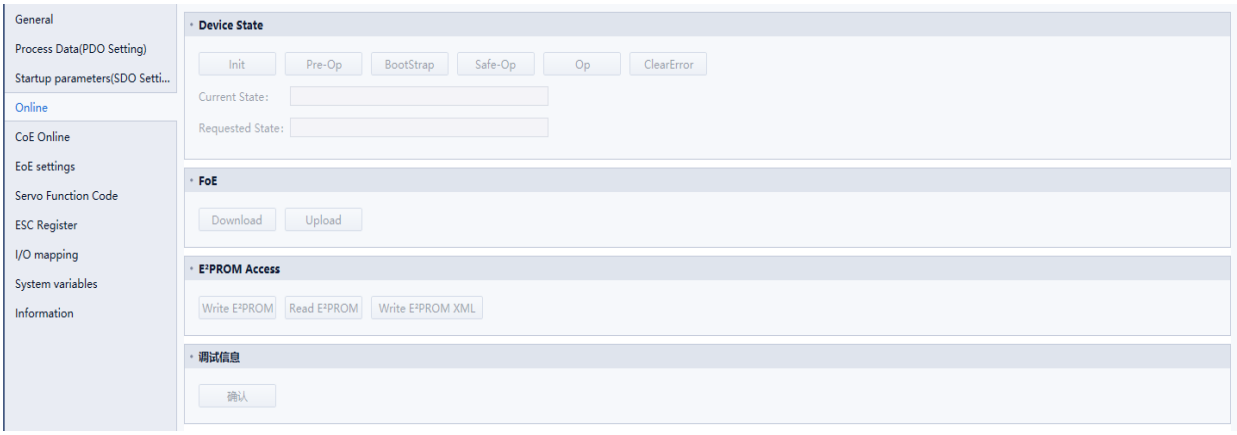
Note: When a fault occurs during the SDO transmission process, it can be resolved through the following options:

- Exit on error: If this option is selected, the SDO transmission will exit if an error is detected.
- Skip on error: If this option is selected, the SDO transmission will skip to the next line and continue if an error is detected.



17.1.7.4 Online

The slave online configuration editor can only be used after logging into the device. The related functions and interface are as follows.



- **Device state**

The EtherCAT state machine coordinates the status relationships between the master and slave applications during initialization and operation, as shown in the following figure. The EtherCAT operation transition sequence is: Initialization -> Pre-operational -> Safe-operational -> Operational. The following table introduces each status transition relationship.

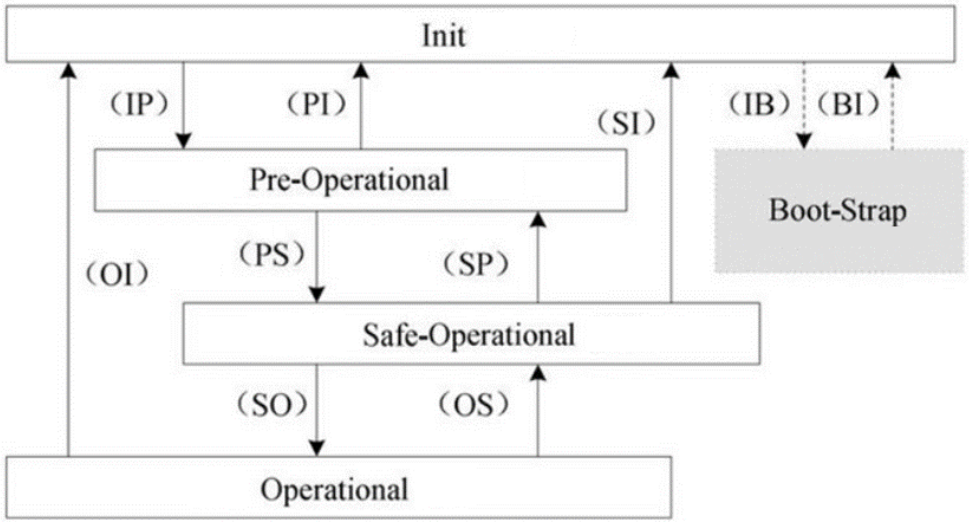


Figure 17-1-7-4-1 EtherCAT status transition relationship

No.	Status Transition	Local Management Service
①	IP	Starts mailbox communication.
②	PI	Stops mailbox communication.
③	PS	Starts input updating.
④	SP	Stops input updating.
⑤	SO	Starts output updating.
⑥	OS	Stops output updating.

No.	Status Transition	Local Management Service
⑦	OP	Stops output/input updating.
⑧	SI	Stops output updating and mailbox communication.
⑨	OI	Stops output/input updating and mailbox communication.
⑩	IB	Starts boot mode.
⑪	BI	Restart the device

  - Clear error: Click this button to clear the error in the current status.
  - Current status: Displays the current operating status of the device.
  - Request status: Displays the currently requested status. When selecting another status and the transition has not been completed, this bar will display the requested but uncompleted status.

- FoE

  Used to transmit firmware files to the slave for firmware upgrades.

  Click the "Boot status" button to transit the slave to boot mode. Click "Download"/"Upload" to download or upload firmware files and a dialog box pops up. Enter a string and password to execute the file transfer. This information is provided by the slave device and recorded in the slave's data table. Avoid power outages or status switches during firmware upgrades.

- ^{E2}PROM interface

  The configuration of the slave can be read from or written to the E2PROM.

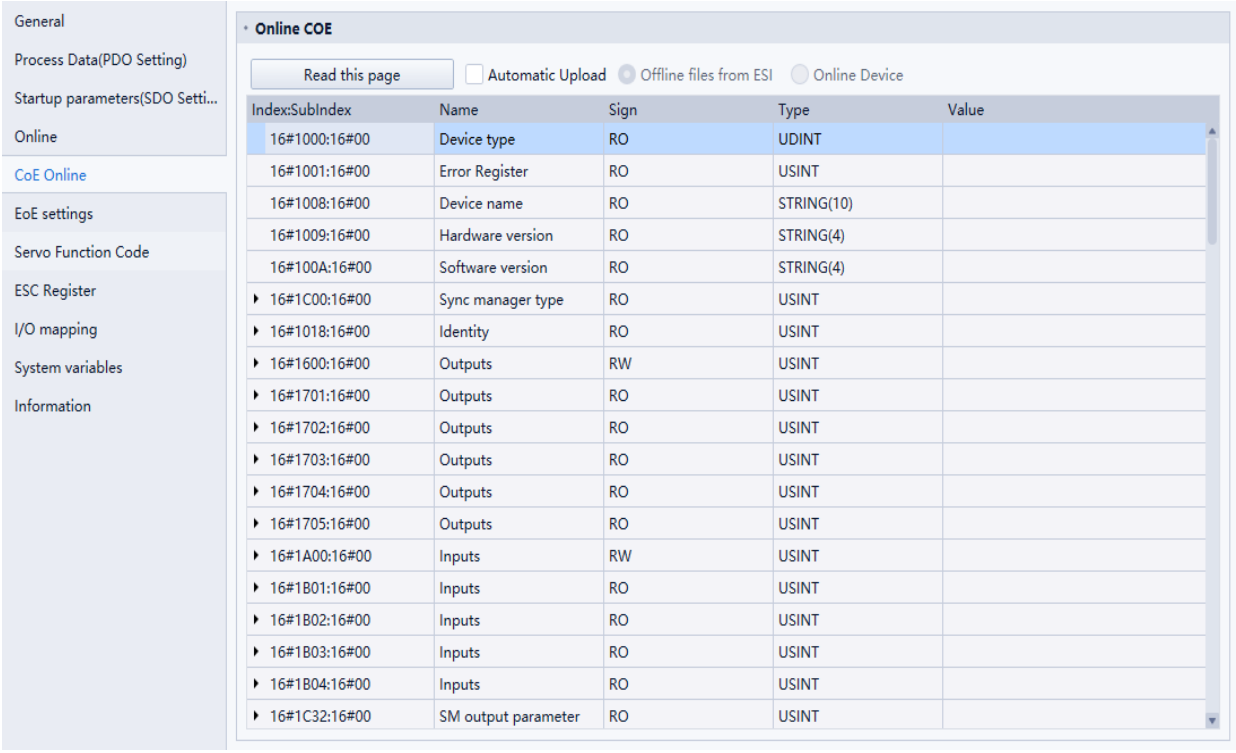
  Similar to firmware file transfer, this operation will also pop up a dialog box to open or save the file. Users can use the "Write E2PROMXML" command to directly write the slave configuration from an XML file to the device. This command is only valid if there is configuration data in the XML file (in the <configuration data> section).

- Commissioning

  Click "Clear" to clear all fault codes from the slave.

17.1.7.5 Online CoE

The values of online CoE can only be read when the bus is operating normally and the PLC is logged in, as shown in the following figure.



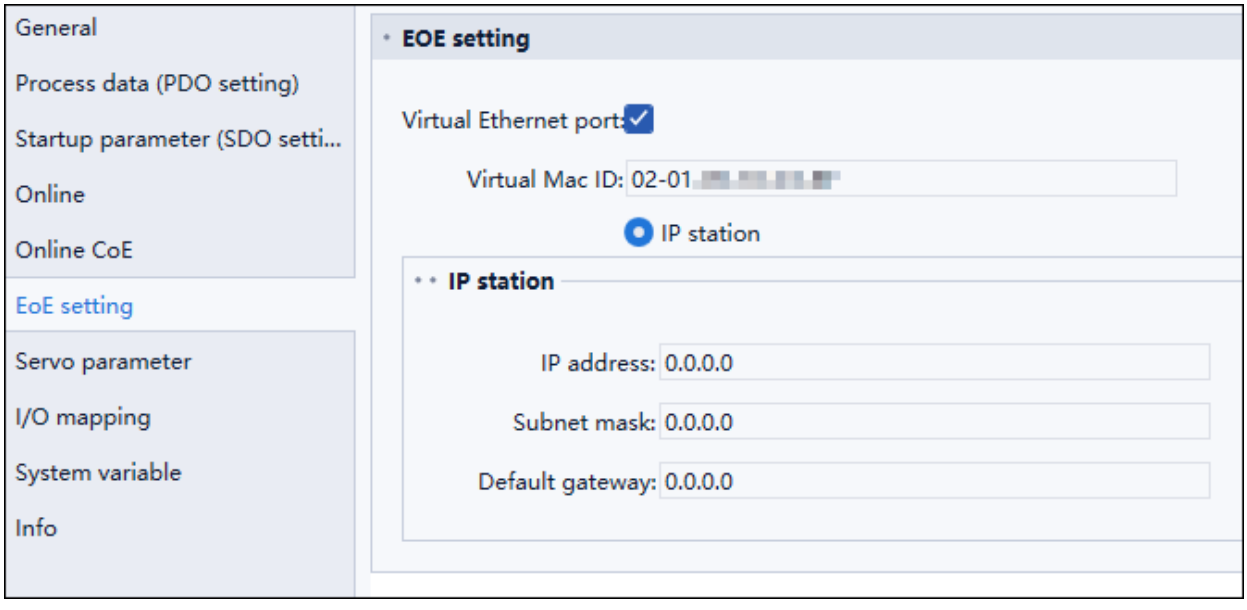
- The interface displays the index, item, flag, type, and value information of the data.
- The flags include RO (read-only) and RW (read-write).
- Click "Read this page" to read the values of the parameters on the current page.
- Click "Auto upload" to automatically read the values of the parameters from the device.
- Select "Offline files from ESI" or "Online device" to switch the source of the data in the list.

17.1.7.6 EoE Settings

This function is currently not supported for the MD800 and MD810 series products.

EtherNET over EtherCAT (EoE) allows any EtherNET device to be connected to EtherCAT via a conversion terminal without affecting the real-time performance of EtherCAT. Similar to well-known Internet protocols, such as TCP/IP, VPN, PPPoE (DSL), Ethernet frames are transmitted via the EtherCAT protocol.

- This setting allows standard network devices to be connected to end devices (such as printers, PCs) via a switch by enabling "Virtual Ethernet port" and set the "IP station".



For special slave devices that support EoE, communication can be set. The following dialog box will only pop up when the device supports EoE.

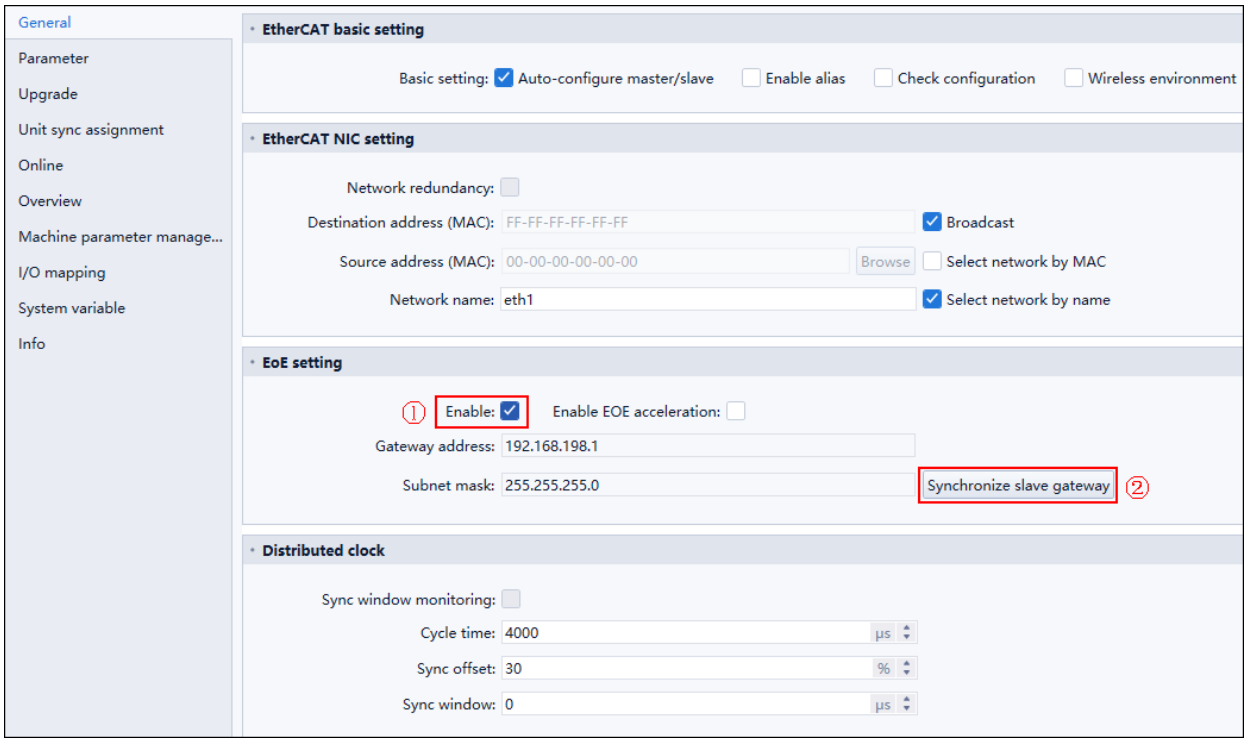
- Virtual Ethernet port: Enables the EoE function of the slave device. If this option is checked, a special "Virtual Mac ID" must be defined.
- Virtual MacID: Automatically assigned by the system, which can be manually modified by users. If a MacID duplication occurs, a compilation fault will occur.
- IP station: Configures the device as an IP interface. When this option is selected, the Ethernet communication parameters must be set.

  Ethernet communication parameters must be set based on the parameters of the virtual Ethernet adapter. The entries for IP address, subnet mask, and default gateway are each allocated four bytes to identify the slave in the network and support modifications to the default settings.

NOTE

The IP address must be in the same subnet as the virtual Ethernet adapter. For example, if the Ethernet adapter's address is 192.168.1.1 and the subnet mask is 255.255.255.0, then the IP address must be within the range of 192.168.1.2 to 192.168.1.254.

- After configuring the EoE settings of the slave, double-click the master node "EtherCAT_Master (EtherCAT Master)", check "Enable" in the "EoE settings" section of the general settings, and then click "Synchronize slave gateway" to complete the EtherCAT configuration, as shown in the figure below:



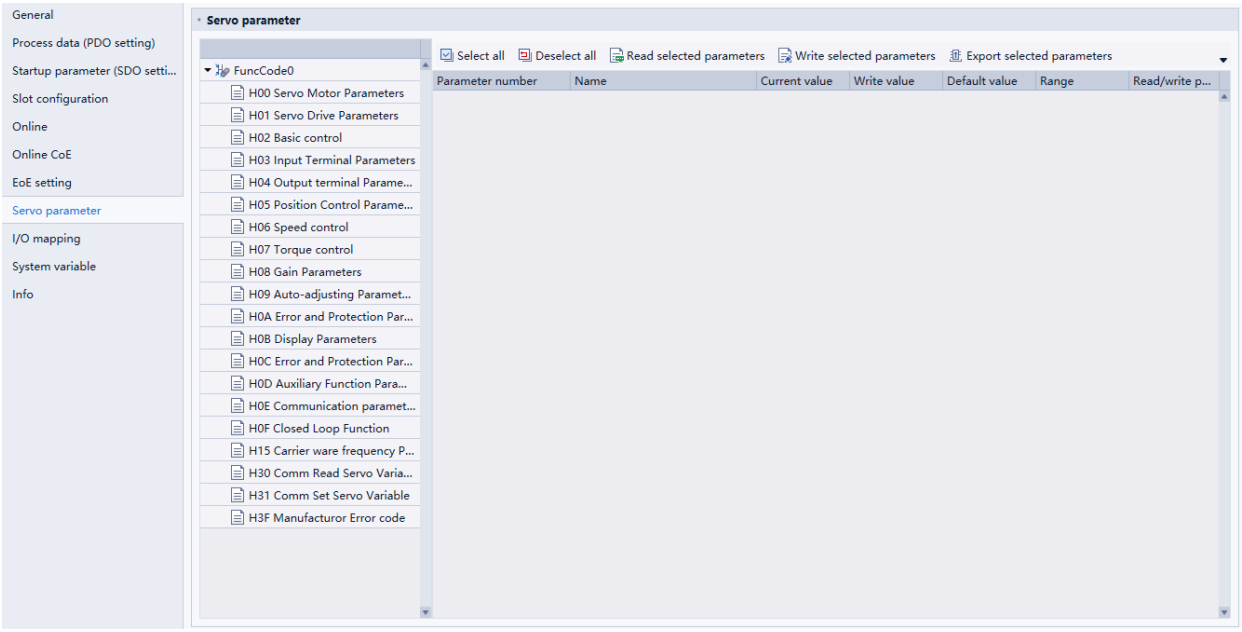
NOTE

The gateway address cannot be in the same subnet as the PLC's IP address. If they are in the same subnet, it is recommended to modify it manually.

Only devices that support the EoE function can be configured.

17.1.7.7 Servo Parameters

Servo function codes refer to the manufacturer parameters defined for Inovance servo products. Under the "Servo parameter" tab of slave configuration, the manufacturer parameters of multiple product can be read and written in batches, imported, and exported, facilitating commissioning and maintenance.

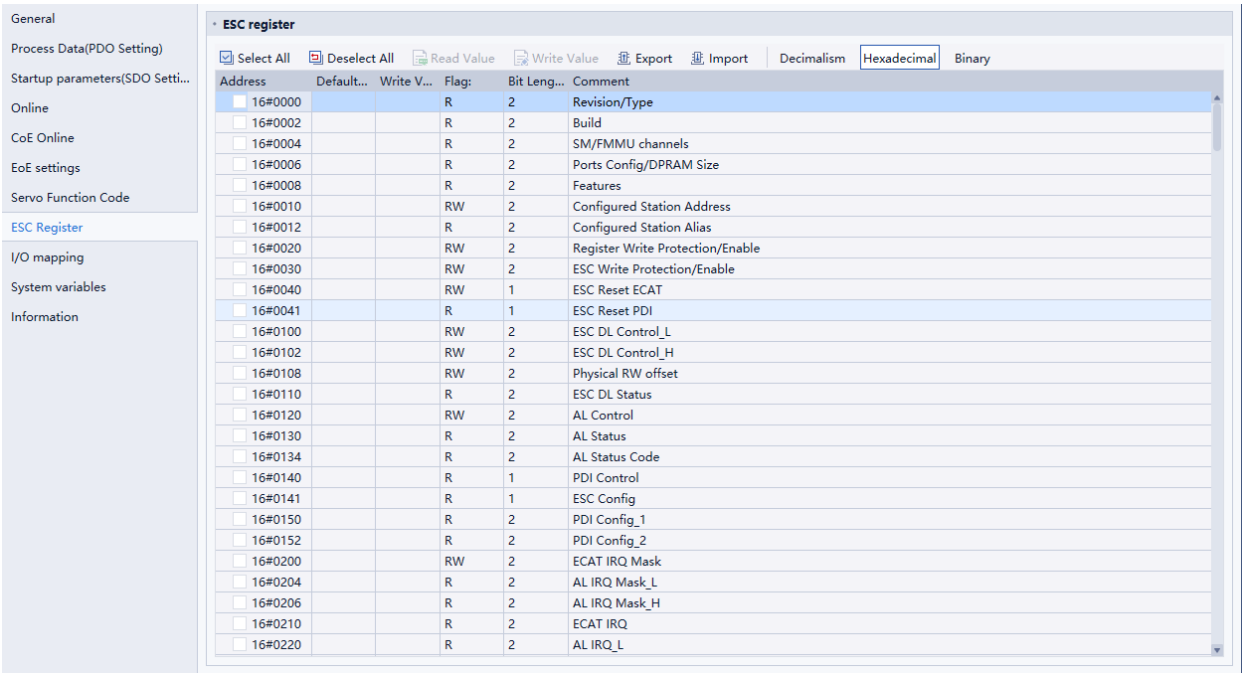


- Select all: Selects all slaves and axes, as well as the servo function codes under the axes.
- Deselect all: Cancels all selected items.
- Read selected parameters: In the servo operation status, reads the servo function codes with RO, RW, or R properties.
- Write selected parameters: In the servo operation status, writes the servo function codes with W or RW properties.
- Export selected parameters: Exports the selected servo parameters in an Excel file.
- Import parameters: Imports the parameters of all slaves from an external Excel file. If the configuration does not match the file, a fault will be reported.

17.1.7.8 ESC Register

The ESC register is used to read the values of the ESC chip register addresses in advanced commissioning, as shown in the figure below. For descriptions of each function and parameter, see the table below.

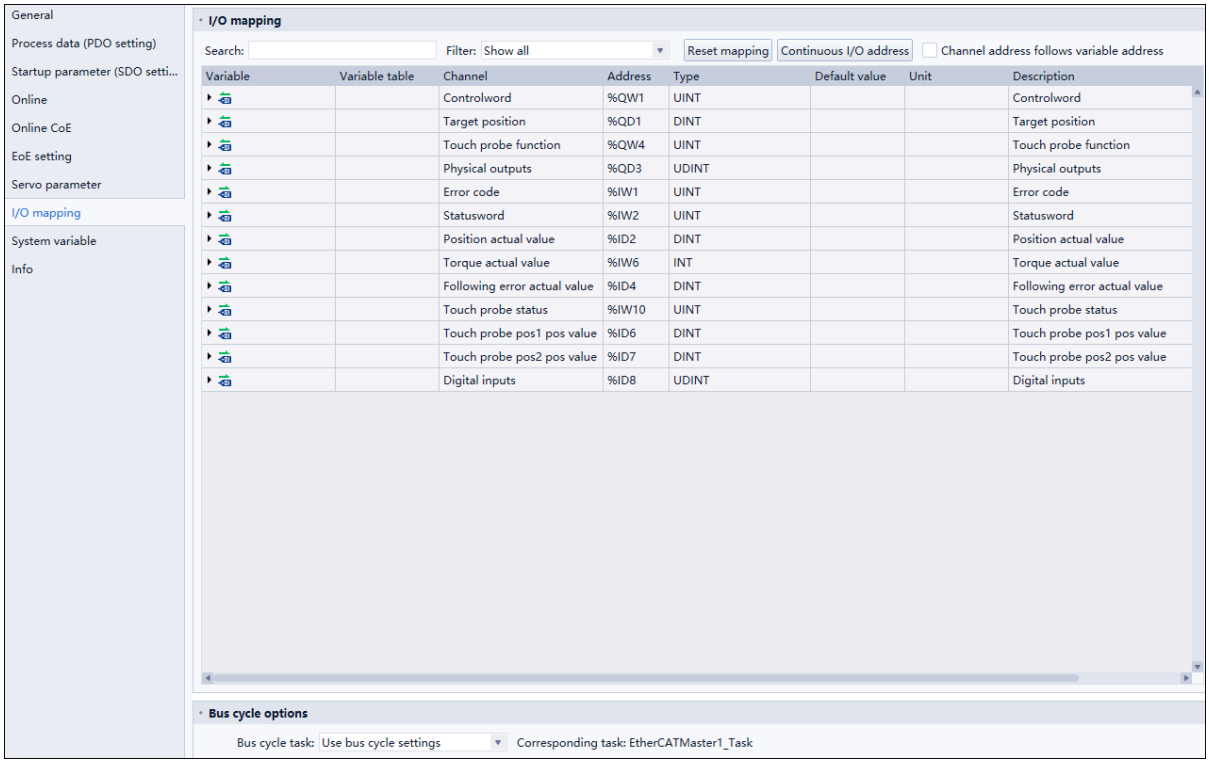
Currently, only the SV680N series device supports this feature.



Parameter	Description
Select all	Selects all parameters in the list.
Deselect all	Deselects all parameters in the list.
Read value	Reads the values of the selected items in operation status.
Writing a Value	Writes the values with write property in operation status.
Export	Exports the selected items as an .xml file and save it to the local disk.
Import	Imports an .xml file from the local disk that meets the format requirements.
Display format	Switches the display format to decimal, hexadecimal, or binary.

17.1.7.9 I/O Mapping

The "I/O mapping" interface is shown in the figure below, and detailed explanations of the parameters are provided in the table below.



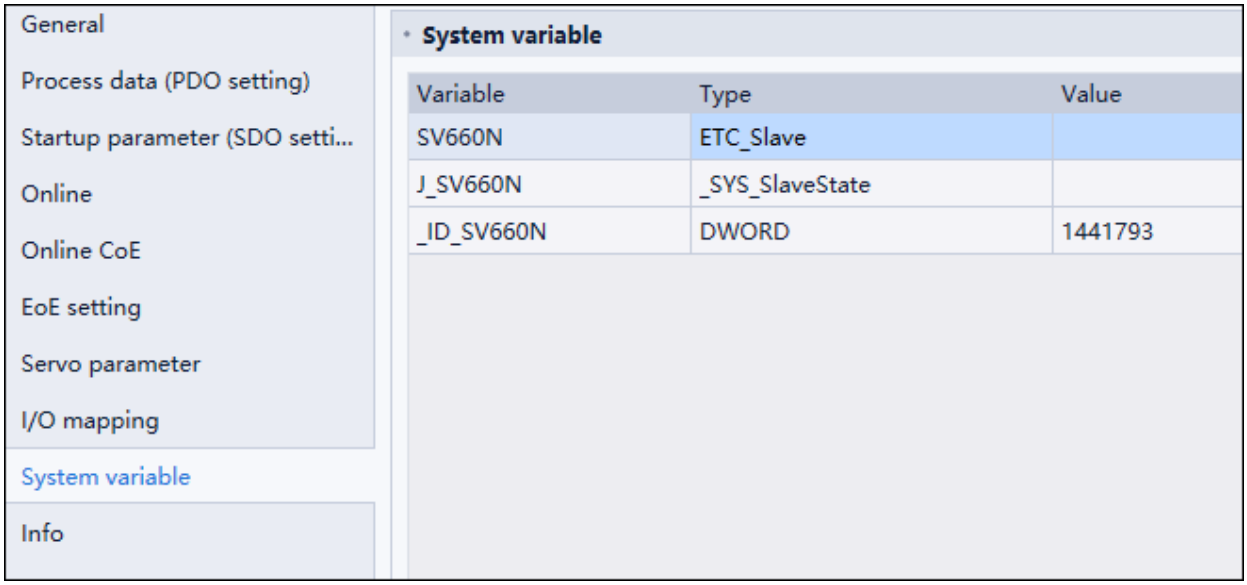
Parameter	Description
Search	Searches for the corresponding I/O mapping channel based on keywords.
Filter	Displays the corresponding variables based on the filtering option.
Reset mapping	Clears the mapped variable information.
Continuous I/O address	Clears the device's I/O address and reassign it to a continuous address.
Channel address follows variable address	This option is not checked by default. If "Channel address follows variable address" is not selected: The I/O mapping channel address is fixed, and the variable type must strictly match the channel type. This mode only supports binding standard data types. After the variable is bound to the channel, the channel address will be automatically assigned to the variable. If "Channel address follows variable address" is selected: Binding can be completed when the channel byte size matches the variable byte size. This mode supports binding standard data types, structures, unions, and enumerations as custom types. After the variable is bound to the channel, the I/O mapping channel address will be removed, and the channel address will automatically follow the variable address.

Parameter	Description
Variable	Configures the variables for I/O mapping, which can be performed in the following two ways: • Mapping the I/O mapping channel to an existing variable (standard data type) Double-click the cell in the "Variable" column, and click "...". Select the existing variable to be mapped in the pop-up "Input assistant" interface and click "OK". • Mapping the I/O mapping channel to an existing variable (struct/union/array) 1. Select "Channel address follows variable address". 2. Double-click the cell in the "Variable" column, and click "...". Select the user-defined instance to be mapped in the pop-up "Input assistant" interface and click "OK". 3. When binding a user-defined type to the first channel, select the binding method in the dialog box. ▪ Bind the structure directly to this channel: After selected, the variable is bound only to this channel. ▪ Bind the elements in the structure sequentially to all channels of this device: After selected, the variables in the struct are automatically bound to all channels of the device in order; when the struct variable has multiple levels of nesting, only the first-level variables are bound sequentially. • Mapping the I/O mapping channel to a new variable Double-click the cell in the "Variable" column, enter the name of the new variable, and press the Enter key. The new variable will be synchronized to the default variable table. The name of the new variable must comply with IEC programming standards, otherwise it will result in a compilation fault.
Variable table	Identifies the type of variable table for I/O mapping variables.
Channel	Indicates the bit number of the current I/O mapping channel.

Parameter	Description
Address	Indicates the address assigned to the current I/O mapping variable. To customize the I/O address, double-click the cell in the "Address" column to modify it.
Type	Indicates the data type of the current I/O mapping variable.
Default value	Indicates the default value of the current I/O mapping variable.
Unit	Indicates the unit of the current I/O mapping variable.
Description	Indicates the description information of the current I/O mapping variable.
Prepared value	Indicates the value pre-written or forced for the current I/O mapping variable. Double-clicking the cell in the "Prepared value" column for modification. After modification, use "Ctrl+F7" or "F7" to write the value or force it to be uploaded to the I/O mapping variable.
Bus cycle task	This function is applicable to virtual axes. The "Bus cycle task" must be bound to the called EtherCAT task (default is "Use bus cycle settings", which is applicable to a single master). It is prohibited to call the same virtual axis under multiple EtherCAT tasks.

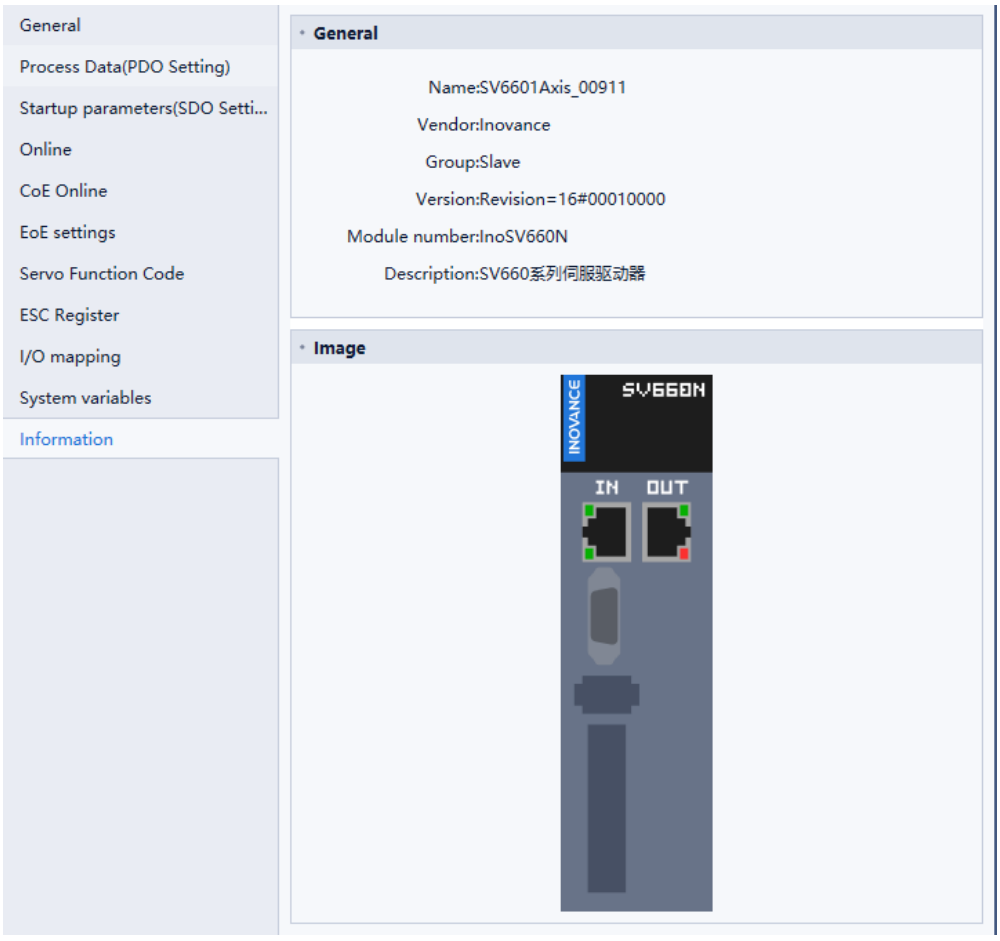
17.1.7.10 System Variables

In the "System variable" tab of the EtherCAT slave, instances (variables) of type ETC_Slave and _SYS_SlaveState are specified for servo slaves I/O, supporting control of devices connected to the EtherCAT slave via user programs, and displaying the device ID value of this slave.



17.1.7.11 Information

This interface displays the general information and image of the device, as shown in the following figure:



17.1.8 Drive Connection

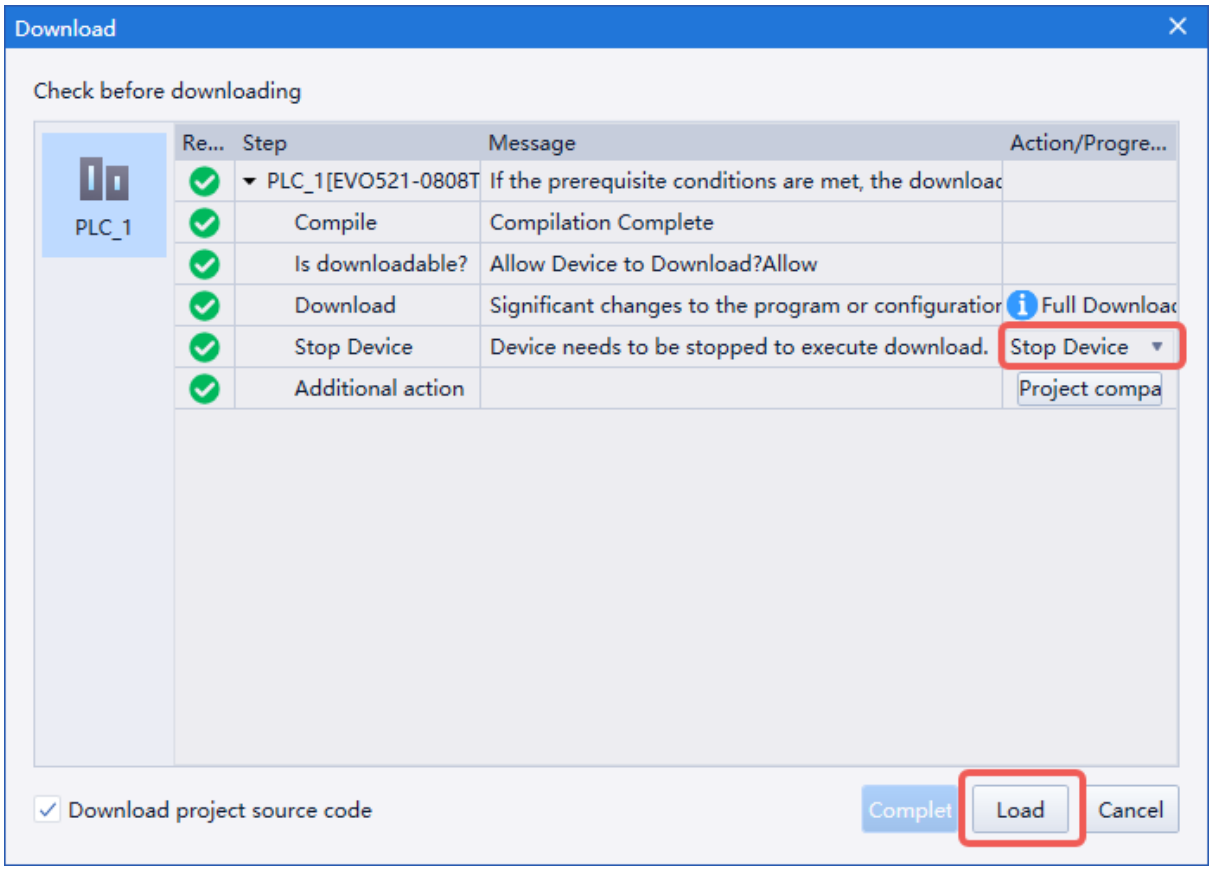
After the EtherCAT and drive device are configured, you can connect the drive.

Prerequisites

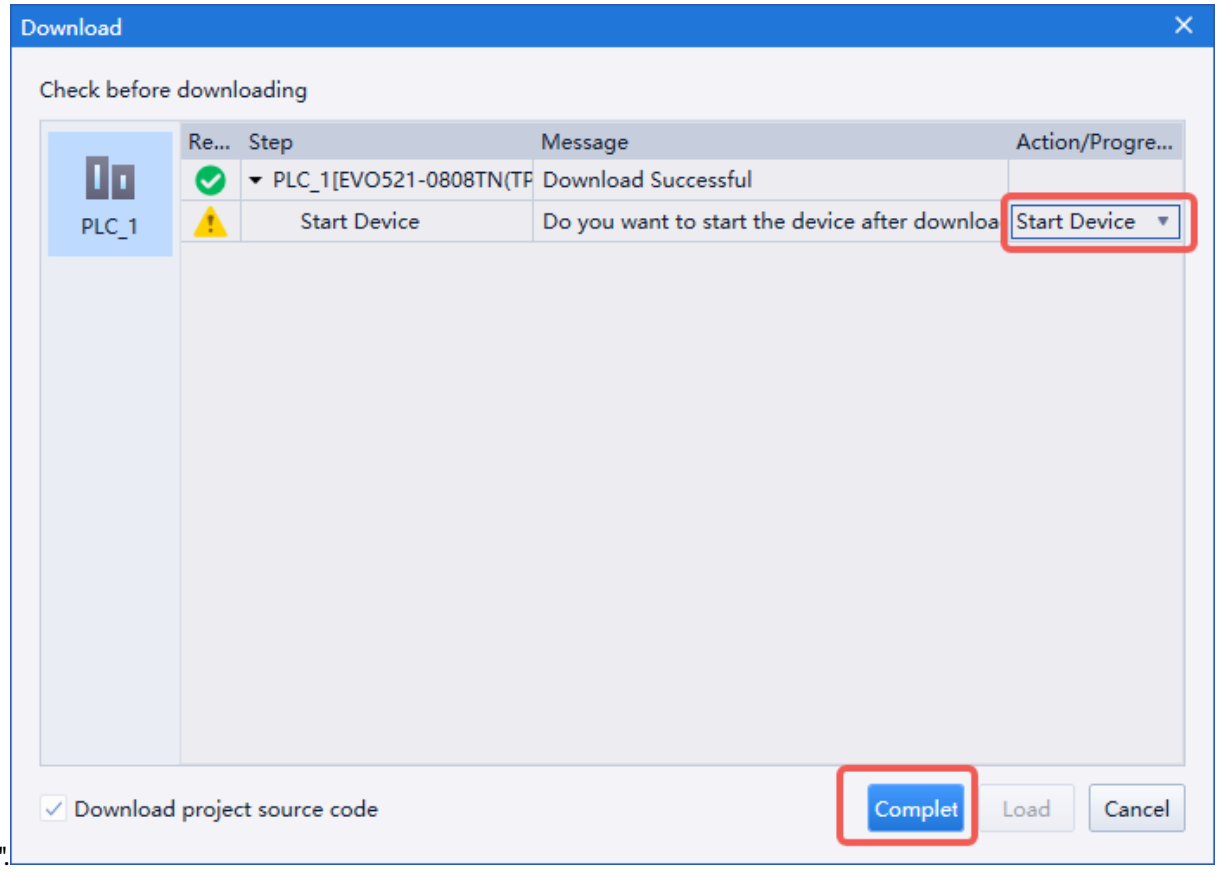
The PLC has been switched to online mode.

Procedure

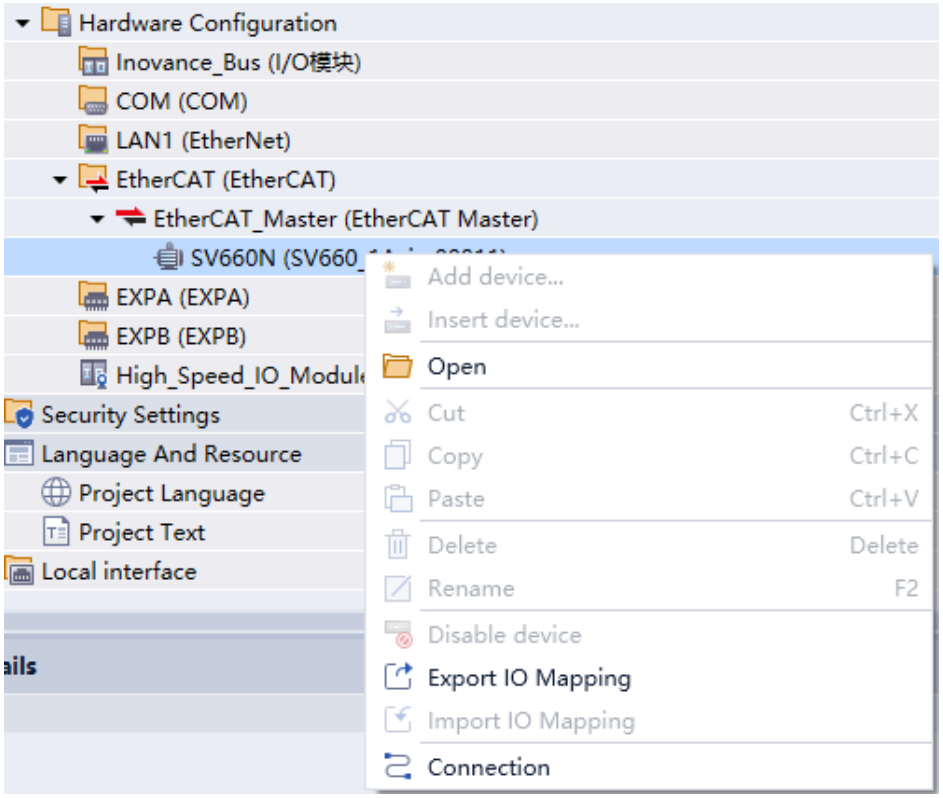
1. Click the "Online" icon on the toolbar. In the pop-up window, switch "No action" to "Stop device", and then click "Load".



2. After successful loading, switch "No action" to "Start device" in the pop-up window, then click "OK".



3. Right-click the servo drive node and select "Connect" to connect the drive.



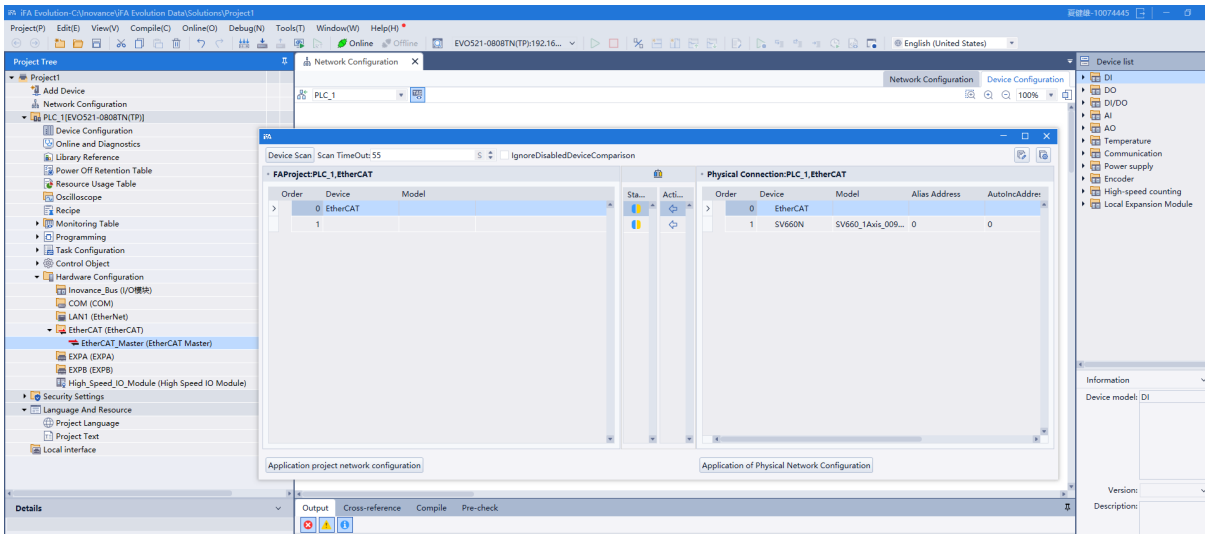
Note: If a device is faulty, a corresponding prompt will appear when connecting the device; if the PLC has not been switched to online mode, a corresponding prompt box will also appear.

17.1.9 Version Switching

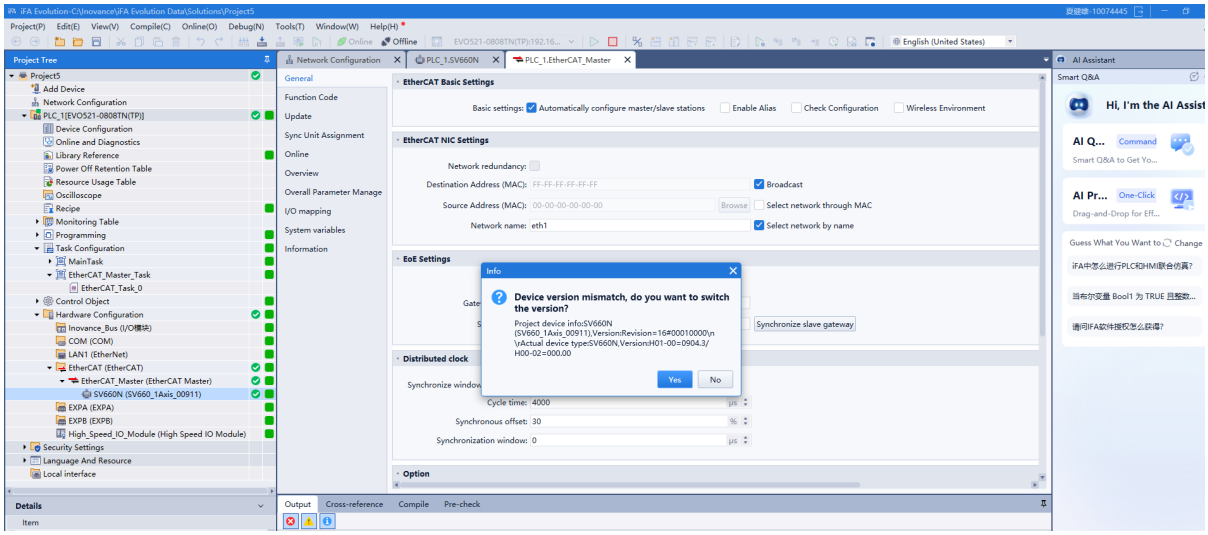
When the version of the device created in the project is inconsistent with that of the actual connected hardware device, version switching is required for device connection.

Procedure

1. After IEC scans the device, click "Connect" to check if the version added offline matches that of the actually connected device.
   Note: IEC scanning does not distinguish between device versions; for example, whether SV660NC or SV660NS is connected, the scanning result will all be SV660N.

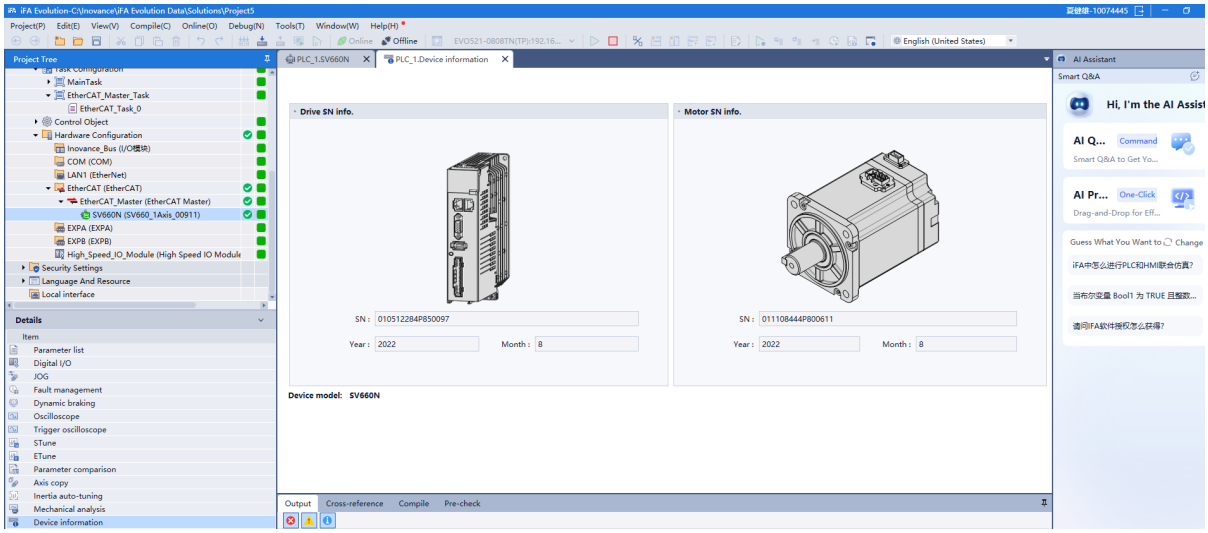


2. If the device versions are inconsistent, a version switching prompt will appear.



3. Version switching will delete the function nodes under the servo and their corresponding views, and refresh the commissioning function nodes in the detailed view.

4. After the version switching, the specific device type will be displayed in the device information interface.



Note:

- After reconnecting the device after disconnection, the version switching prompt will no longer appear.
- When opening a historical project and connecting the device, the version switching prompt will not appear.

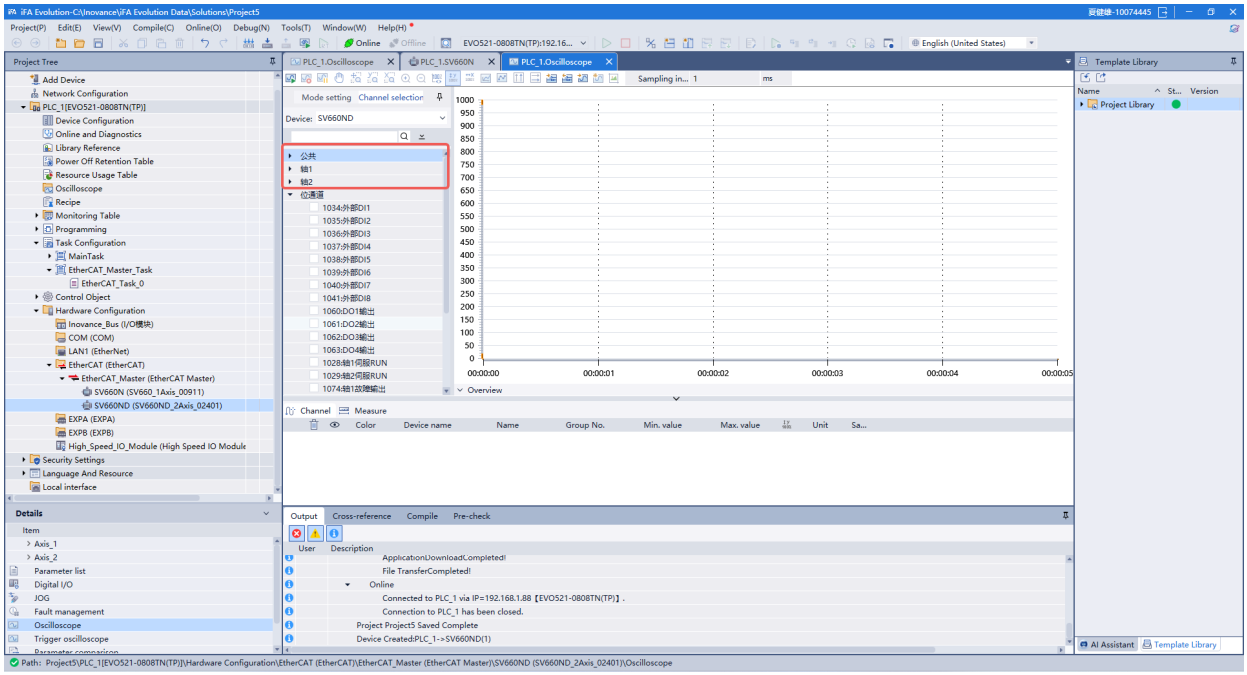
17.2 General Functions

17.2.1 Oscilloscope

17.2.1.1 Overview

This function is not supported by the STD60N series products.

Axis 1 and Axis 2 functions are available for dual-axis device.

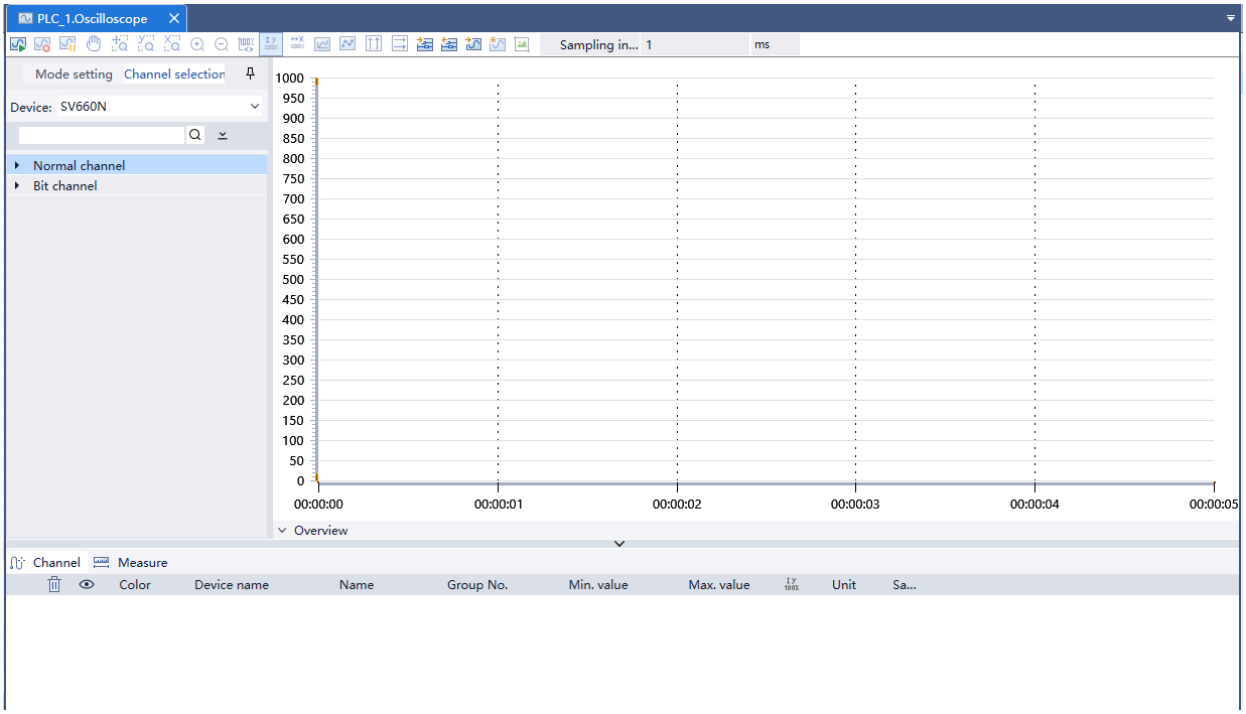


Oscilloscope mode is divided into continuous oscilloscope and trigger oscilloscope.

- Continuous oscilloscope: During on-site commissioning, the oscilloscope is typically used to monitor the operational status of device in real time. It continuously samples signals from selected parameter channels, and analyzes the device's operational status by observing waveform changes in the channel data.
- Trigger oscilloscope: The trigger conditions of the trigger object are configured. When the trigger conditions are met, save the channel values to be monitored to the cache of the drive. Then you can read the channel values which are displayed as a curve. This mode enables you to perform commissioning according to the field conditions and observe the change of variables after certain conditions are met.

The sampling modes of the oscilloscope are as follows:

Sampling Mode	Description
Continuous	Samples in a continuous and uninterrupted way.
Single trigger	Samples once when specific trigger conditions are met.



Note:

- Under the same project, any two consecutive oscilloscopes cannot run simultaneously.
- Servo hardware devices do not support multiple oscilloscopes (including trigger oscilloscopes and continuous oscilloscopes) collecting data simultaneously.
- The oscilloscope can acquire data from multiple servos under the same PLC.
- One drive does not allow simultaneous acquisition of multiple trigger oscilloscopes.

17.2.1.2 Functions

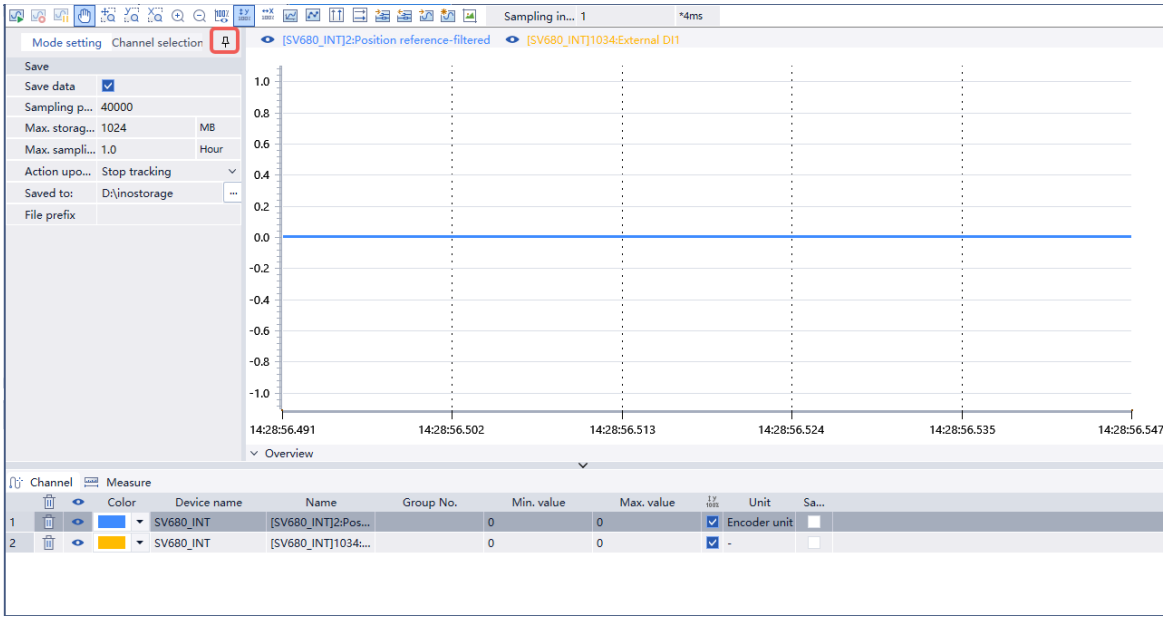
The functions of the oscilloscope are as follows:

- **Toolbar**

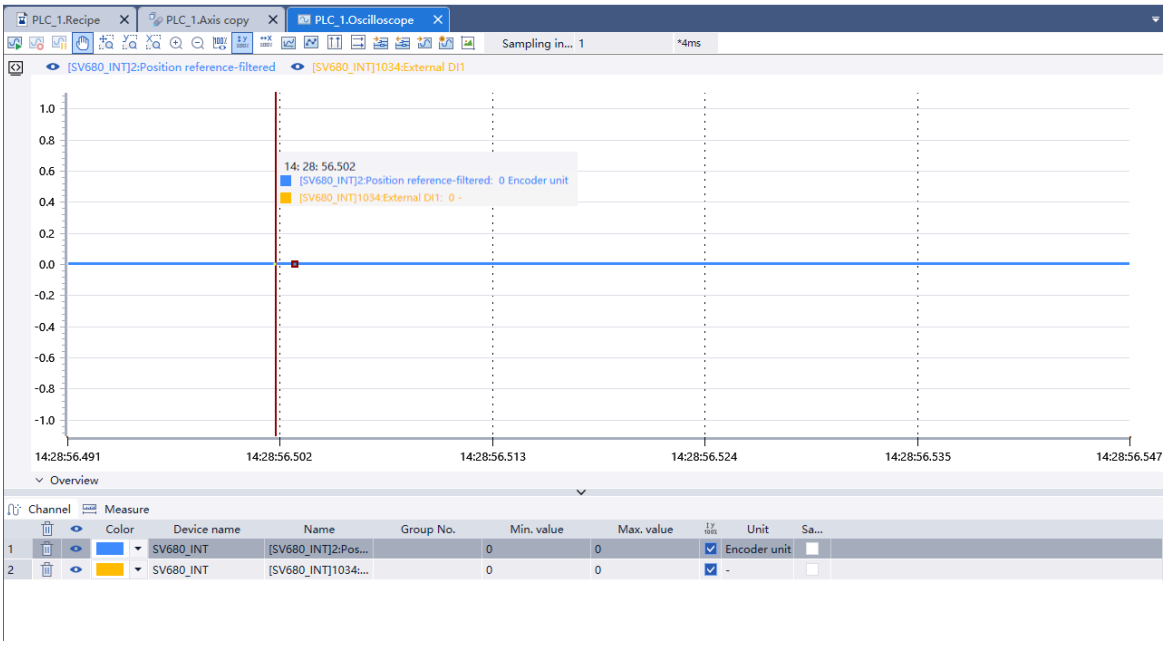
  For detailed descriptions of each function button in the auxiliary toolbar for operating the oscilloscope, see "Analysis Auxiliary Function".

- **Overview**

  The overview diagram includes a collapse button and area viewing function (viewing a specific area's waveform by dragging with the mouse).

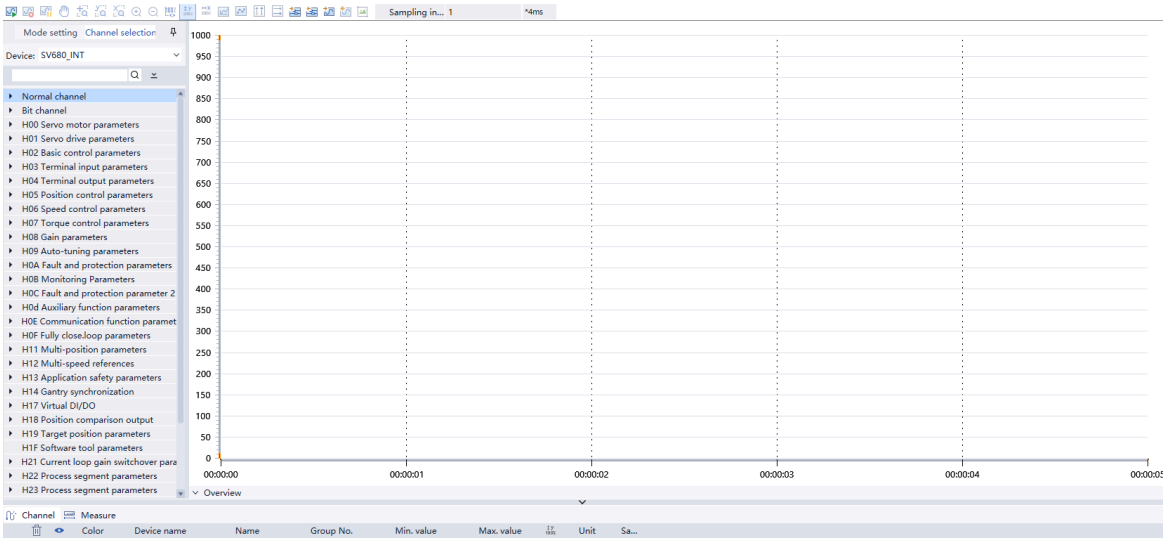


Click the collapse button to collapse the left-side menu bar.

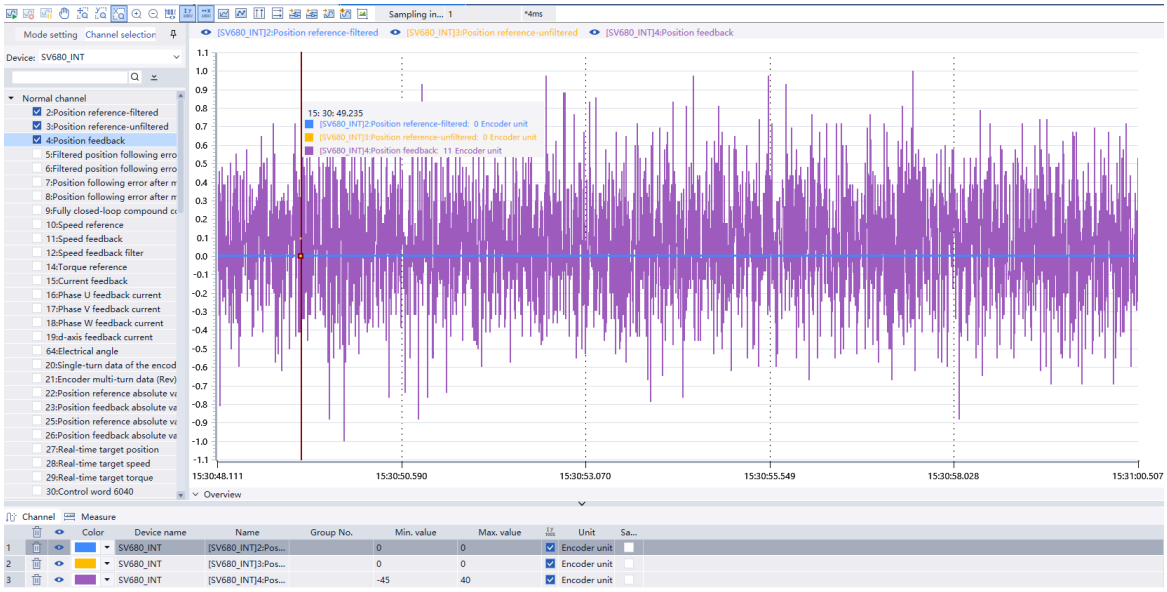


- **Detailed area**

  The detailed area is the main display area (waveform area).

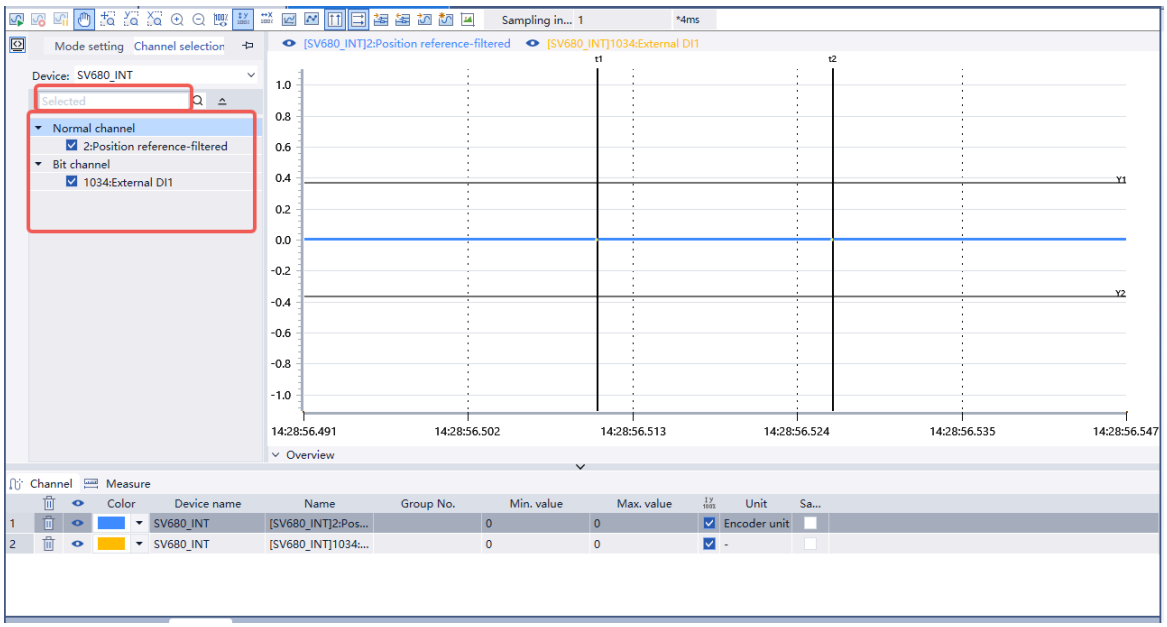


- **Channel**
  - The channel table mainly includes signal display/hide, color settings, and grouping functions.
    - Delete
      - Deletes the channel information of an entire row.
    - Show/Hide column: This column contains a toggle icon, which allows you to show or hide the signal in the detailed area.
      - The toggle icon in the column header applies to all signals.
      - The icon remains visible as long as at least one signal is visible. It only becomes hidden when all signals are invisible.
      - When hiding the currently viewed signal, the next visible signal in the current group becomes the current viewed signal; if no other visible signal exists in the current group, the next visible signal among all signals becomes the current viewed signal; if all signals are hidden, the button in the column header becomes hidden, and the detailed area reverts to default style.
      - When hiding a signal, if this signal is the last visible signal in the current group, the entire group is hidden in the detailed area.
    - Color setting column: The set signal color remains consistent in the Y-axis title, curve, and legend of the detailed chart.



- Group number column: Signals with the same group number are displayed in the same group.
  - Grouping is not supported in single-channel mode (group numbers do not take effect).
  - Signals without a group number form their own separate group.
  - When no visible signal exists in a group, the group is hidden.

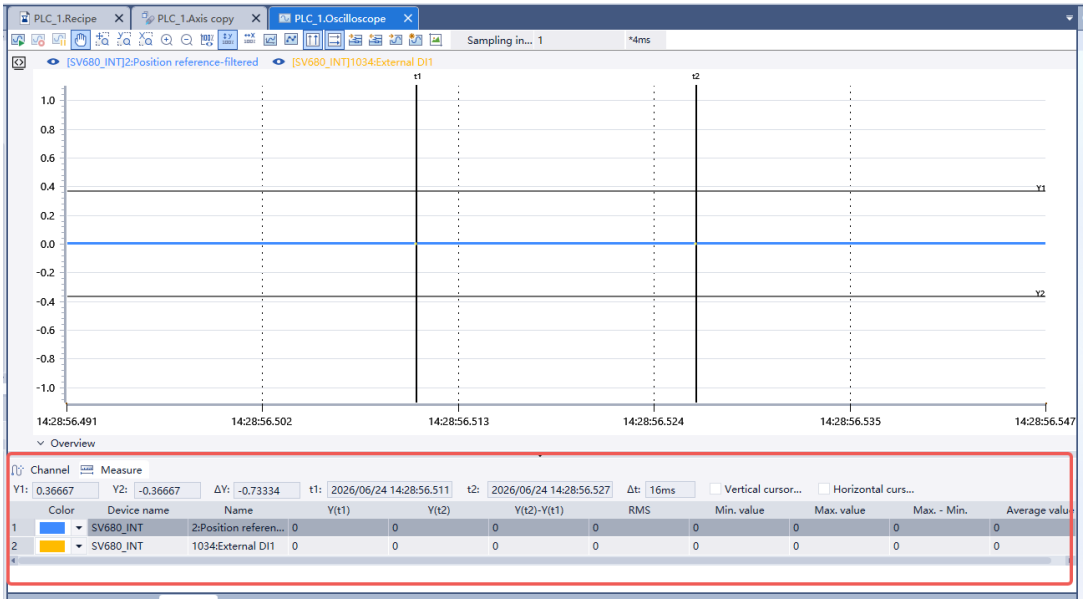
◦ Selected channels



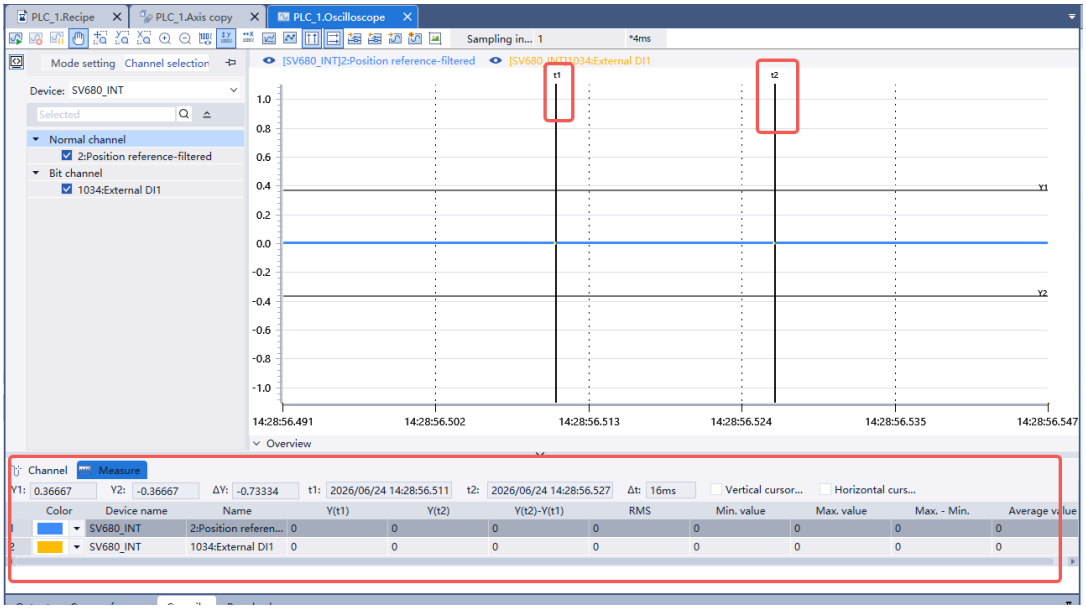
- **Measurement**

  X-axis and Y-axis cursor measurement.

  - Y-axis cursor: Click the "Horizontal cursor" icon button on the toolbar to display it.



- The measurement table is used in conjunction with the X-axis cursor to display statistical results within the X-axis cursor area; click the "Vertical cursor" icon button on the toolbar to display it.



17.2.1.3 Configuration

Prerequisites

- The project has been created.
- The servo device has been added.
- The servo device is online.

Steps

1. Click an added servo device in the project tree.
2. Double-click "Oscilloscope" in the detailed view to open the interface.
3. Se the sampling mode to the continuous trigger mode or single trigger mode, and set corresponding parameters.

4. Select channel parameters.

5. Start the oscilloscope.

17.2.1.4 Continuous Oscilloscope

Sampling settings

1. Set the sampling mode to "Continuous".

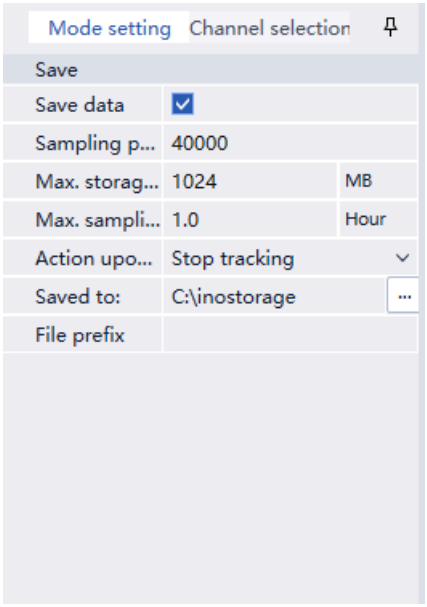
2. Set the sampling interval, which ranges from 1 to 200, with a default value of 1.

   After activating logging, the minimum sampling interval is calculated based on the selected signals. If the set value is smaller than the calculated minimum sampling interval, the set value will be overridden by the minimum value.

   The EoE network calculates the minimum sampling interval based on the controller's communication cycle and the number of channels, using the following formula:

Minimum sampling interval = Communication cycle * Transmission cycle * Bandwidth occupancy rate / (Oscilloscope data frame bytes / (Channel width * Number of channels))

The transmission cycle is 2, the oscilloscope bandwidth occupancy in normal mode is 25%, the number of data frame bytes is 120, and the channel width is default to 4.



Storage settings

The following table shows the functions and detailed descriptions of the storage settings:

Function	Description
Store continuous data	Not enabled by default. You can set the related parameters only when this option is checked.
Sampling per file	Default: 10000; range: 10000 to 30000
Max. storage data	Default: 1; range: 1 to 1000000; unit: MB. Provided that the storage capacity requirement is met, an error margin of 0 to 3 MB is permitted during actual data sampling.

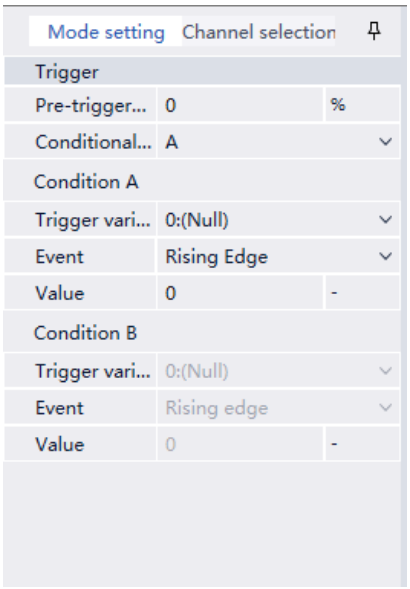
Function	Description
Max. sampling time	Default: 1 hour; range: 1.0 to 24000; unit: hour
Action upon max. data or time	When sampling data size or sampling time reaches the limit, you can choose to stop tracking (default) or delete old files.
Save to	Select the local directory for saving oscilloscope data.
File prefix	Set the prefix for oscilloscope data saving file. The default prefix is empty.

17.2.1.5 Trigger Oscilloscope

Currently, the trigger oscilloscope is not supported by the MD810 and MD800 series products.

Axis 1 and Axis 2 functions are available for dual-axis device.

Configure the trigger condition for the servo drive and poll the trigger results. If the trigger is successful, the data is read and displayed.



Sampling settings

The trigger mode only supports single device (current device) sampling. The triggered sampling mode supports single trigger.

The default unit for sampling interval is 62.5 μs (unit value is refreshed when the device is online). If the device is online, the unit value is read from the device. The sampling interval ranges from 1 to 1000×62.5 μs.

Triggering

1. Configure the data ratio set by pre-trigger. Unit: %; range: 0 to 100.
2. Configure the relationship between conditions. You can select the relationship between trigger condition A and trigger condition B.

Option	Description
A	Trigger the data recording when condition A is met.

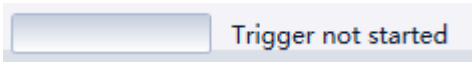
Option	Description
B	Trigger the data recording when condition B is met.
A and B	Trigger the data recording when conditions A and B are met.
A or B	Trigger the data recording when condition A or B is met.

3. Configure the trigger condition parameters.

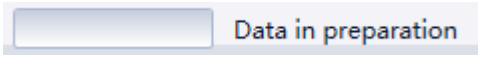
Function	Description
Trigger variable	Select the trigger variable in the drop-down box. The content is provided by the device file.
Event	Select the trigger event from the drop-down box. Options: rising edge, equal to, falling edge, and edge change
Value	Enter the trigger value in the text box. Limit the input range according to the value range.
Value range	Displays the range of the currently selected trigger variable, which cannot be edited.

Trigger process

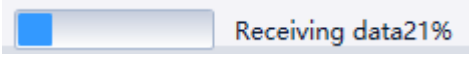
1. Indicates the trigger progress.
   - Initially, prompts "Trigger not started".

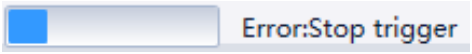
   - After you click "Start", it prompts "Data in preparation".

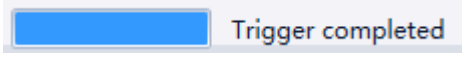
   - Displays the percentage of data reception progress.

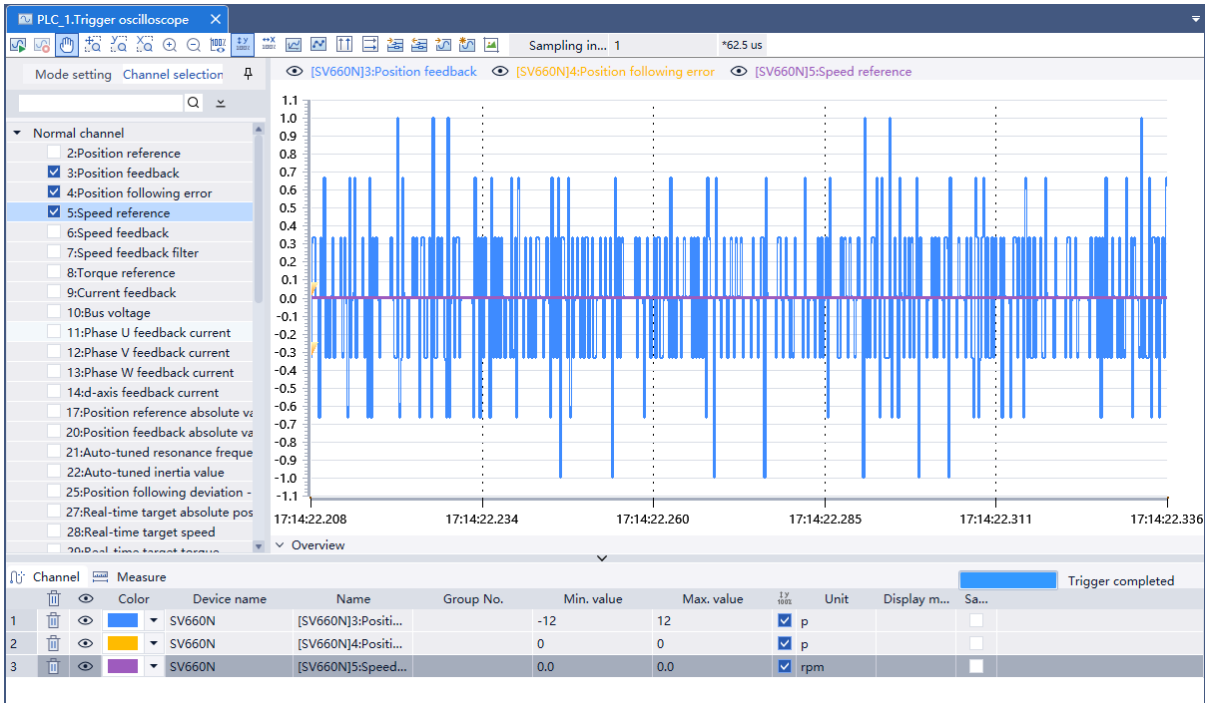
   - When the data reception is stopped, prompts "Error: stop trigger".

   - After the trigger data is received, it prompts "Trigger completed".

2. Trigger result
   - Single trigger



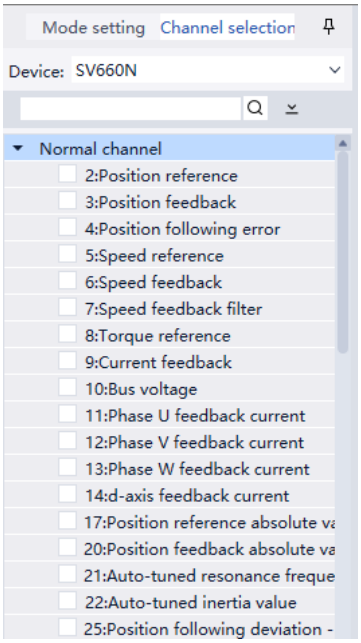
17.2.1.6 Channel Selection

Channel selection

1. Select a configured device.
2. Expand the list of normal channels or bit channels, and check one or more channels. The added channel is displayed in the lower information panel.
   - A corresponding signal is also added to the lower channel and measurement list, and a color is generated randomly.
   - If it is the first visible signal, it is set as the currently viewed signal.
   - In the channel table, after the newly added signal is selected, the Y-axis title of the corresponding color will be displayed in the detailed chart.
   - The corresponding signal will be displayed in the illustration and indicated in color.

   Channel selection limitations:

   - Up to 6 normal channels and 8 bit channels can be selected.
   - If the bit channel is selected, the quota of one normal channel will be occupied. For example, if you select 1 to 8 bit channels, you can only select 5 normal channels at most.



Signal removal

Unselect the selected a signal to remove it. At the same time, the contents related to the signal on the diagram interface are also removed.

- The signal is also removed from the channel and measurement list.
- The Y axis and curve of the signal are removed.
- The signal is removed from the illustration.

17.2.1.7 Waveform File Import/Export

Export

After the oscilloscope stops running, click the data export icon on the toolbar to export the oscilloscope data. The oscilloscope waveform file is named automatically in the following format: file prefix time stamp_serial number.csv. Click "Save" to export the file to the specified location.

Import

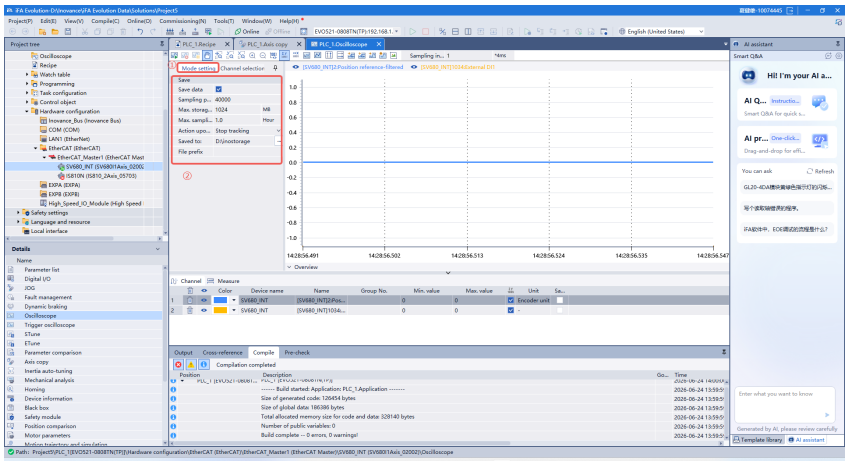
Click the data import icon on the toolbar to import oscilloscope data. Select a CSV data file to display the import result.

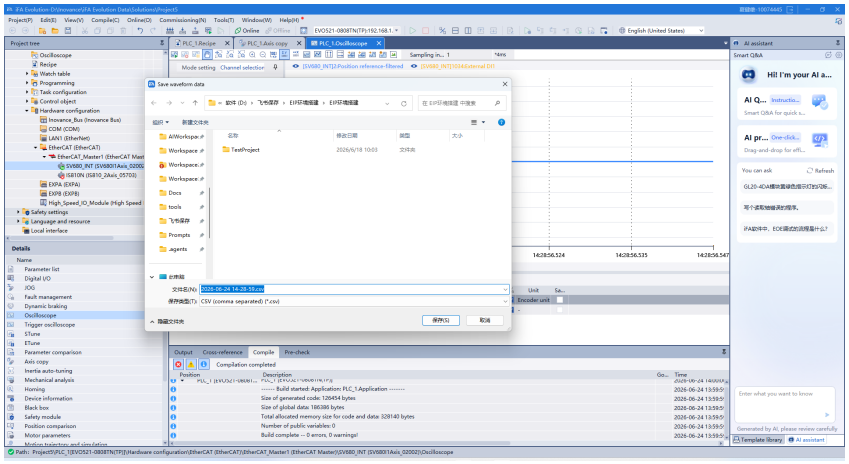
17.2.1.8 Analysis Auxiliary Function

See the following table for descriptions of the icons of the oscilloscope toolbar.



Function		Description
	Start	Starts the oscilloscope to acquire data and draw curves. Restrictions: • The drive must be online. • When a fault occurs on the parameter configuration interface, this process fails.
	Stop	Stops data acquisition.
	Halt	The oscilloscope waveform stops moving and pauses at the current position.
	Move view	When the function is enabled, you can drag the curve in the view, which is mutually exclusive with the zoom function. Restrictions: This function is disabled by default, selected and enabled at continuous acquisition, and disabled at triggered acquisition.
	Zoom	Zooms in or zooms out the curve. When the function is enabled, you can select a waveform segment from the X axis and Y axis for observation. It is mutually exclusive with the drag function. Drag the mouse from the left to right to zoom in the curve. Drag the mouse from right to left to zoom out. Restrictions: This function is disabled by default, unselected but enabled at continuous acquisition, and unselected and disabled at triggered acquisition.
	Vertical zoom	Zooms in or zooms out only on the Y axis Restrictions: This function is disabled by default, unselected but enabled at continuous acquisition, and unselected and disabled at triggered acquisition.
	Horizontal zoom	Zooms in or zooms out only on the X axis Restrictions: This function is disabled by default, unselected but enabled at continuous acquisition, and unselected and disabled at triggered acquisition.
	Zoom in	Zooms in on the entire curve. Restrictions: This function is disabled by default, enabled at continuous acquisition, and disabled at triggered acquisition.
	Zoom out	Zooms out on the entire curve. Restrictions: This function is disabled by default, enabled at continuous acquisition, and disabled at triggered acquisition.

Function		Description
	Show all	Displays 100% of the X and Y axes. When this function is enabled, the icons buttons for displaying all on Y-axis and X-axis are selected.
	Y axis display	Only the Y axis is 100% displayed.
	X axis display	Only the X axis is 100% displayed.
	Channel switchover	Switches between single-channel and multi-channel. Single-channel is displayed by default when the oscilloscope is turned on.
	Discrete diagram	Displays the data sampling points as small dots on the curve. Restrictions: This function is disabled by default and enabled after successful acquisition.
	Vertical measurement cursor	Displays a group of cursors (t1, t2) in horizontal direction, and the statistical data of the sampled data between cursor (t1, t2) in the measurement table. Restrictions: This function is disabled by default, and disabled upon acquisition and enabled after successful acquisition.
	Horizontal measurement cursor	Displays a group of cursors (Y1, Y2) in vertical direction, and displays the cursor values in the measurement window. Restrictions: • The Y axis cursor is displayed in the channel where the current viewing signal is located. Switching the current viewing signal also switches the Y axis cursor. • This function is disabled upon acquisition and enabled after successful acquisition.
	Mode setting	
	Import configuration	Imports the oscilloscope configuration file (JSON format). Restrictions: This function is disabled when acquisition starts.
	Import configuration	Exports the oscilloscope configuration file (JSON format). Restrictions: This function is disabled when acquisition starts.

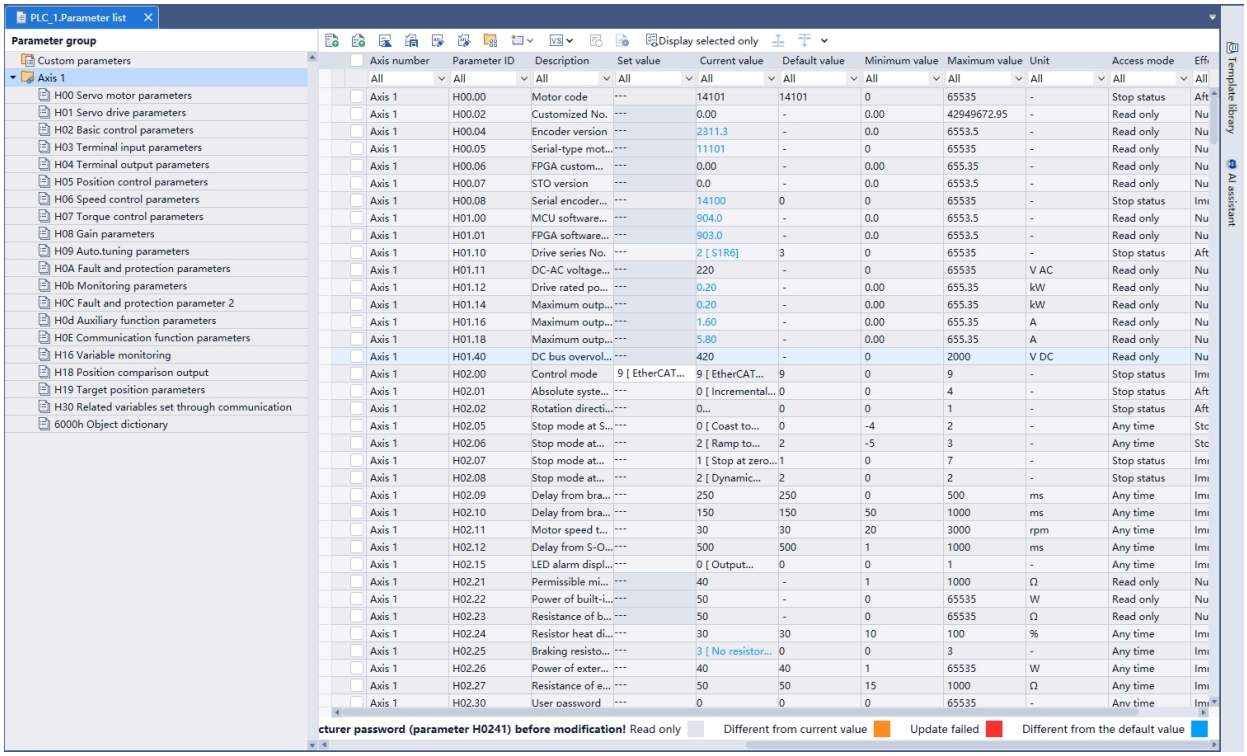
Function		Description
	Export data	Exports the oscilloscope data file (CSV format). Restrictions: This function is disabled when acquisition starts. Saves the current interface waveform. 
	Screenshot	Takes a screenshot of the waveform on the current interface.

The waveform area also includes the following functions:

- When the zoom function is enabled, drag the mouse from left to right for zooming. Dragging the mouse from right to left will display all the data in the overview area.
- Click to zoom:
  - X axis direction: Press and hold the Shift key. The horizontal zooming cursor appears. Click the left mouse button to zoom in, and the right mouse button to zoom out.
  - Y axis direction: Press and hold the Ctrl key. The vertical zooming cursor appears. Click the left mouse button to zoom out, and the right mouse button to zoom in.
- Scroll to zoom.

17.2.2 Parameter List

The parameter list displays and manages parameters, including information display, upload/download, comparison, property update, parameter customization and recipe management. The following figure shows the parameter list interface:

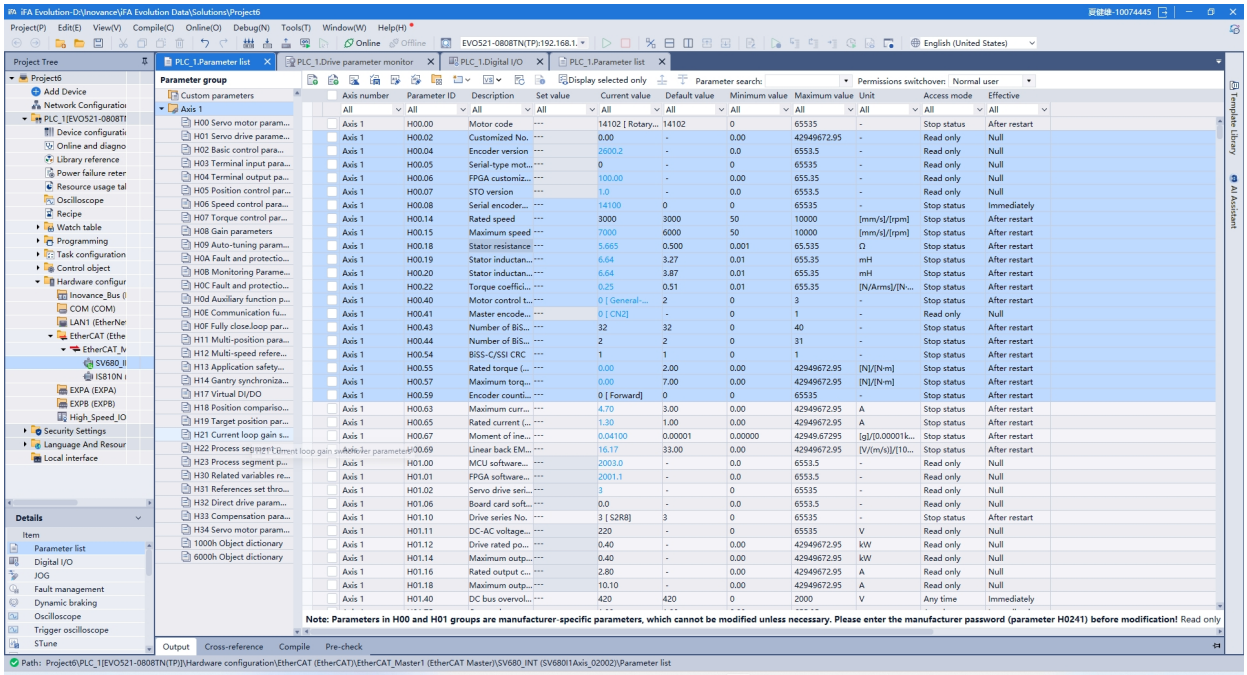


Note: The parameter list interface of the MD630 series differs from other servos and AC drives. For details, see 17.2.2.6 Parameter List (MD6X0 Series).

17.2.2.1 Parameter Viewing

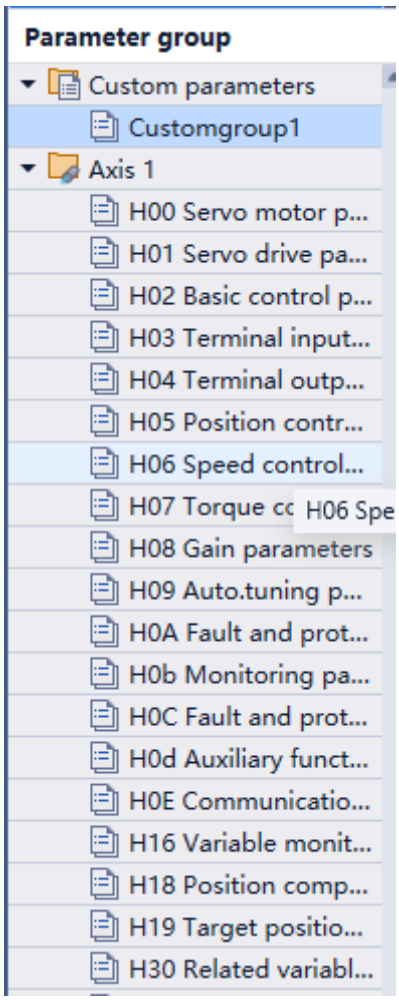
Select a servo drive in the project tree, and double-click "Parameter list" in the detailed view to open the interface.

- Axis 1 and Axis 2 functions are available for dual-axis device.



You can collapse or expand parameter groups via the right-click menu of the top-level node of the default group. For multi-axis devices, the top-level node of the default group is "All parameters"
; for single-axis devices, it is "Axis 1".

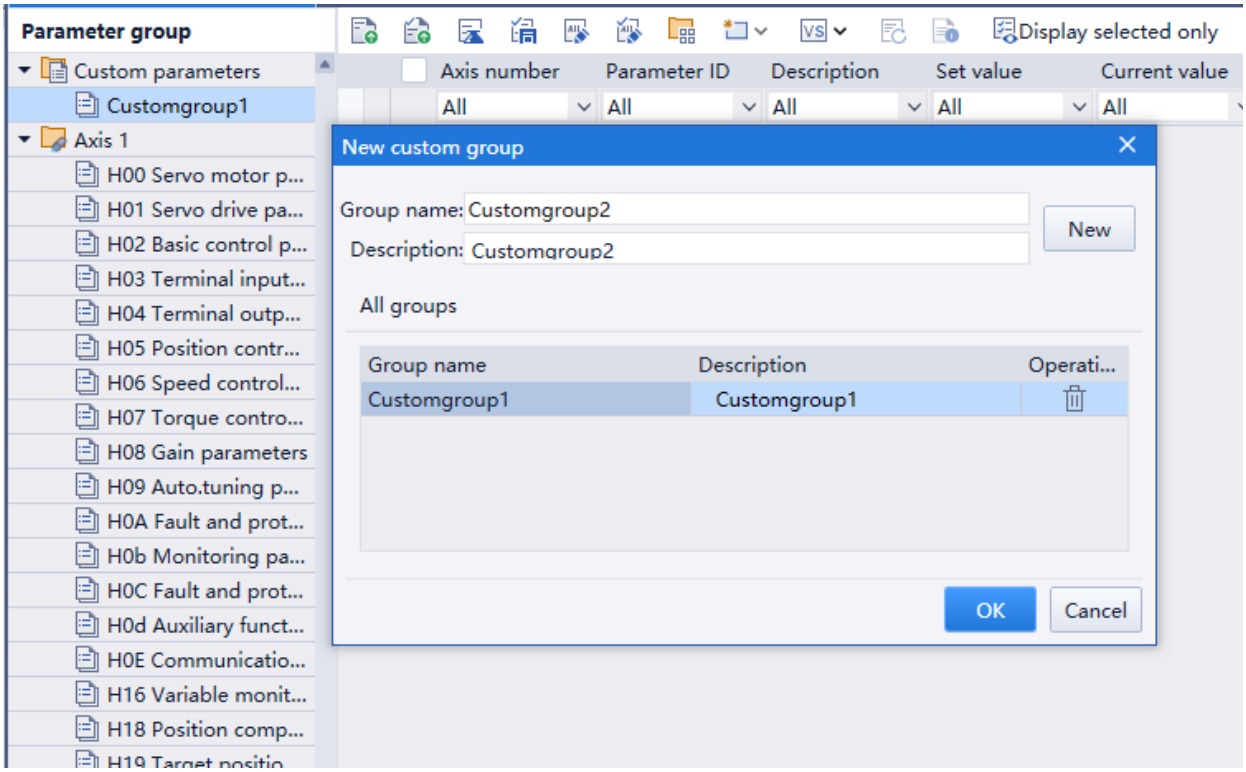
- Select "Custom parameters", and view the parameters by clicking the axis node in the custom parameter list or the default group node under the axis, as shown in the figure below:



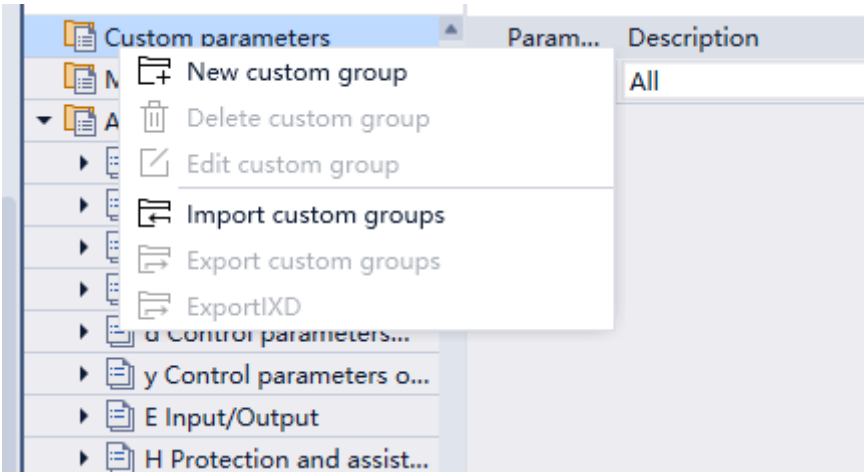
17.2.2.2 Custom Group

Currently, you can create up to 20 custom groups. The specific steps are as follows:

1. Right-click "Custom parameters" on the left side of the parameter list interface and select "New custom group".
2. Enter the group name and description, then click "New". The custom group will appear in the list below. Click "OK" to complete the creation. Multiple custom groups can be created in batch.



3. Right-click the new custom group to perform the following operations.



Function	Description
New custom group	Creates a custom group, with function identical to Step 2. Restrictions: • Up to 20 custom groups can be created. • Multi-level nesting is not supported. • The group name can contain up to 50 characters.
Delete custom group	Deletes the currently selected custom group.
Edit custom group	Modifies the name and description of an existing custom group.

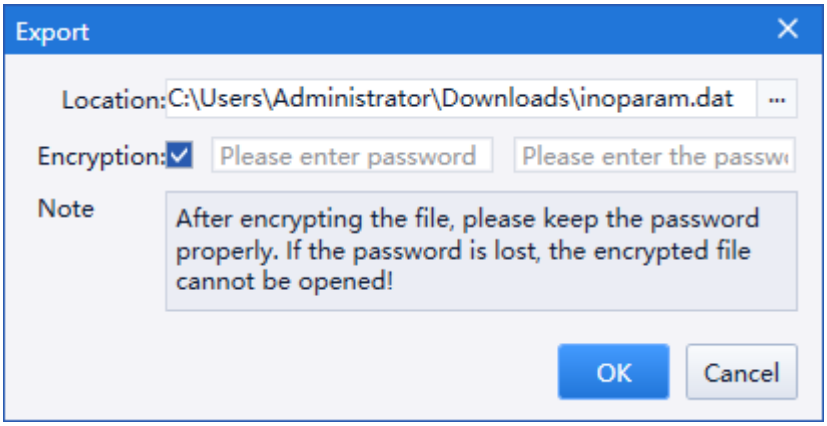
Function	Description
Import custom groups	Imports custom group from the local disk. Restrictions: • Only DAT format file is supported for import. • When the number of custom groups reaches 20, no further custom groups can be imported. • The imported custom group name is the parameter file name. If the file name exceeds 50 characters, the first 50 characters will be used as the group name. • The description of the imported custom group is blank by default.
Export custom groups	Exports the currently selected custom group to the local disk. Restrictions: Only DAT and CSV format files are supported for export.

17.2.2.3 Toolbar

The toolbar for the parameter list is shown in the figure below. For detailed descriptions of the functions, see the table below.



Description		Description
	Upload and save (current page)	Uploads the current values of all parameters on the current page and save them locally. • After the upload is complete, the upload result is displayed, including a list of parameters failed to be uploaded and the reasons for their failure. • You can retry uploading failed parameters.

Description		Description
	Upload and save all selected	Uploads the current values of all selected parameters and save them locally. • After the upload is complete, the upload result is displayed, including a list of parameters failed to be uploaded and the reasons for their failure. • You can retry uploading these parameters. • When exporting files, you can apply encryption settings as needed. 
	Import data	Opens a recipe saved on the local disk. After successfully opening a recipe, its variable values will be linked to the parameters of the current device, serving as the new setting values to be downloaded to the device. Restrictions: • Only DAT format recipe files are supported for opening. • A recipe can only be opened successfully if the device in the recipe matches the current device model being commissioned.
	Save set values of all selected	Saves the values of all selected parameters as data records in a file. The file contains basic information required for verification when using the recipe. Restrictions: Only DAT and CSV format files are supported for saving.
	Download all parameters except H00/H01 group	Writes the setting values of all parameters, excluding the H00 and H01 groups, into the device. • After downloading, the download result is displayed, and any parameters that failed to download, along with the reasons for failure, are shown in a list. • You can retry downloading failed parameters.

Description		Description
	Download checked options (current page)	Writes the setting values of all selected parameters on the current page into the device. • After downloading, the download result is displayed, and any parameters that failed to download, along with the reasons for failure, are shown in a list. • You can retry downloading failed parameters.
	Custom parameters	In the "Custom parameters" interface, you can add or remove parameters from custom groups. Parameters within custom groups are used in the same way as native device parameters. **Steps:** 1. In the drop-down list in the upper right corner, select the custom group where you want to add or remove parameters. 2. Check the required parameters in the left parameter list. 3. Click "Add to custom group" to add the selected parameters to the current group. 4. Click "OK" to complete the parameter addition. **Other functions:** • Right-click menu: ◦ The right-click menu on the left side includes: "Select all", "Deselect all", "Check selected", and "Uncheck selected". ◦ The right-click menu on the right side includes: "Select all", "Deselect all", "Check selected", "Uncheck selected", and "Add to". • New group: Opens the new custom group interface and returns to the custom parameter interface after the interface is closed. Any changes made in the "New custom group" interface will be synchronized to the custom parameters in the right-side parameter list when you click "OK". This synchronization applies to the custom group itself, not to the parameters within it. • Select all: Selects all parameter items in the list. • Delete: Deletes the selected parameters from the current group.
	Add to (selected)	Copies the currently selected parameters to a new or existing custom group. Restriction: Addition is not allowed if no parameters are currently selected.

Description		Description
	Parameter Comparison	Displays the changes of the parameter values. • You can compare the setpoint of a device parameter with its current value and display the comparison result. • You can compare the setpoint of a device parameter with its factory value and display the comparison result. • You can compare the current values with the set values of device parameters, and view the comparison result.
	Change property	Ensures that the parameter properties in the host controller match those in the physical device. Properties include default values, parameter lower/upper limits, and accuracy, and support both manual and automatic updates. Restrictions: • If the device supports automatic updates, its properties will be updated automatically upon device connection, and the update process cannot be canceled. • Cancellation is supported during manual attribute updates.
	Parameter description	Displays detailed information about the parameter.
	Display selected only	Displays only the selected items in the parameter list.
	Move up	Moves the parameter up in the custom parameter group.
	Move down	Moves the parameter down in the custom parameter group.
	Parameter search	Searches for parameters matching the keyword. Restrictions: • The search scope includes parameter ID, address, description, unit name, parameter upper limit, parameter lower limit, and access mode. • It offers a selection of previously searched keywords, allowing you to easily choose a historical keyword for your search.

Description		Description
	Permissions switchover	Limits the user's access range to parameters. The drop-down list supports two permission options: "Normal user" and "Super administrator". Restrictions: • When switching the permission from "Super administrator" to "Normal user", the permissions for all drive devices in the current project will be changed to "Normal user". • When "Super administrator" is selected, a pop-up window will prompt you to enter the password and ask whether to apply the permission setting to all drive devices in the project. If "Apply to all drive devices in the project" is selected and the password is correct, the permissions for all drive devices in the current project will be changed to "Super administrator". If not selected, only the current device will be set to "Super administrator". • The scope of permission includes viewing and operating parameters related to parameter list, custom parameters, parameter comparison, axis copy, and other functions.

17.2.2.4 Parameter Modification

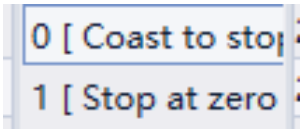
The access modes for editing parameters in the parameter list are classified as "Read only", "Stop status", and "Any time".

Parameters with access modes of "Stop status" and "Any time" can be edited by clicking the value in the "Set value" column. The editable types include the following:

1. General type: Simple types such as integers can be directly entered in the input box.



2. Drop-down box



3. Bit field

Index	Current value	Set value
bit0: Output DO active signal if current position changes from "les...	1	0
bit1: Output DO active signal if current position changes from "mo...	0	0

4. Bit field drop-down box

Index	Current value	Set value
bit0 to bit3: E234.0 fault level setting	0 [Default level]	0 [Default level]

5. 5-segment code drop-down box (exists in AC drive parameter list)

Index	Current value	Set value
Ones: 位置指令符号	0 [正]	0 [正]
Tens: 加减速时间选择	1 [位置控制加减速时间1]	1 [位置控制加减速时间1]

6. 5-segment code bit (not currently available)
7. 5-segment code bit drop-down box (not currently available)

Note: When editing a parameter, if the setting value is invalid, it will be reset to the previous data.

Right-clicking the parameter list editing area will display the following menu for parameter operations. For details, see the following table.

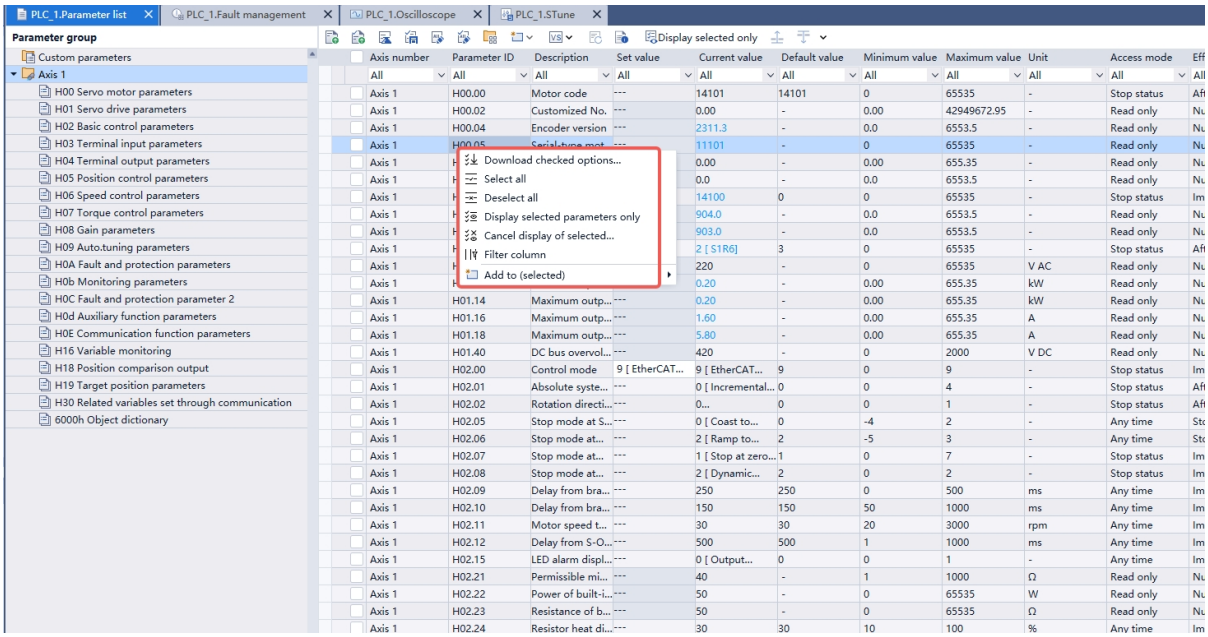


Figure 17-2-2-4-1 Default parameters

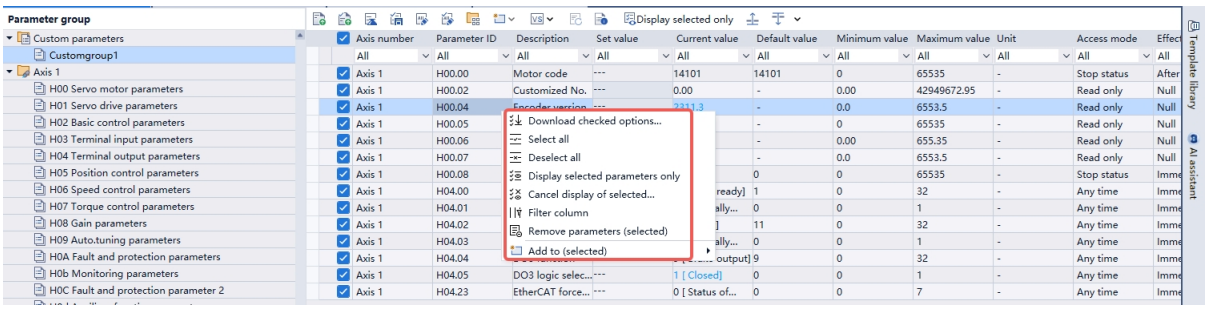


Figure 17-2-2-4-2 Custom parameters

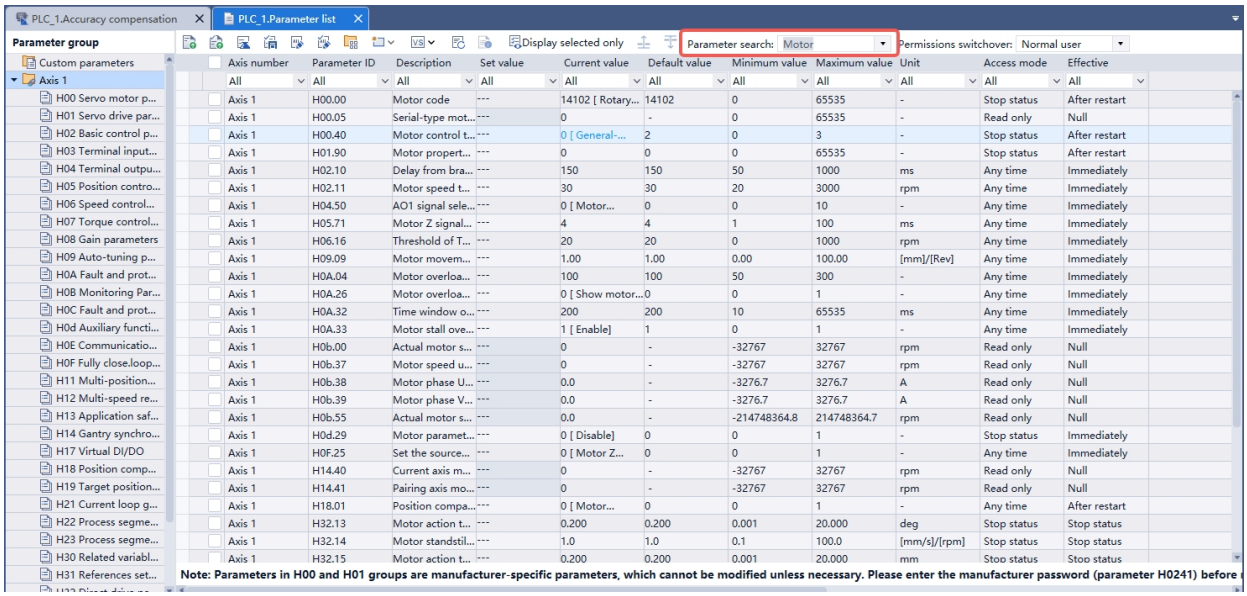
Function	Description
Download checked options (current page)	Writes the setting values of the selected parameters on the current page into the device. • After downloading, the download result is displayed, and any parameters that failed to download, along with the reasons for failure, are shown in a list. • You can retry downloading failed parameters.
Add the selected to the drive monitoring table (current page)	Adds the selected parameters on the current page to the drive monitoring table.
Select all	Selects all parameters on the current page.
Deselect all	Deselects all parameters on the current page.
Display selected parameters only	Displays only the selected parameters on the current page, hiding those that are not selected.
Cancel display of selected parameters only	Cancels the display of only the selected parameters on the current page, i.e., displays all parameters on the current page.

Function	Description
Filter column	Filters the columns displayed in the parameter list based on the filter conditions set in the pop-up "Filter column" dialog box.
Remove parameters (selected)	Removes the selected parameters from the current custom group. Restriction: This function is supported only in the parameter list of a custom group.
Add to (selected)	Copies the currently selected parameters to a new or existing custom group.

17.2.2.5 Parameter Search

The "Parameter search" function in the toolbar is used to search for target parameters. Note the following instructions:

- Except for the "Effective" parameter, you can search for other parameters by entering relevant keywords in the input box.
- The "Parameter search" drop-down box offers a selection of previously searched keywords for quick search.
- To cancel the filter after searching, select "All" from the drop-down list to revert to displaying all parameters.
- Keyword search uses fuzzy matching and is case-insensitive. For example, entering the keyword "Motor" will return the related parameters, as shown in the figure below.



- The search scope is limited to the currently selected node only. For example, if an axis node is selected, the search will cover all parameters within that axis. If a parameter group node is selected, the search will cover all parameters within that group.

17.2.2.6 Parameter List (MD6X0 Series)

The parameter list function of the MD630 series displays and manages parameters, which include information display, upload/download, property update, description, and custom parameter management.

Parameter viewing

Select a parameter node on the left to display its information on the right. The node contains the parameter information of all subnodes.

- Parameters under the "All parameters" node and its subnodes are system default parameters.
- Parameters under the "Modified parameters" node are parameters whose current values differ from the default values.
- Parameters under the "Custom parameters" node are parameters added by the user.

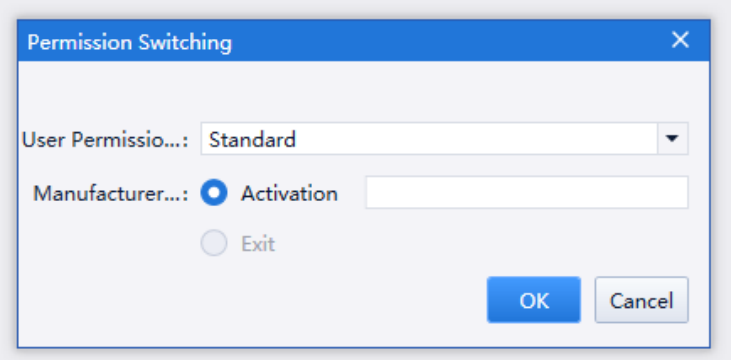
Custom parameters

See "Custom Group" for details.

Toolbar

- Displayed in Workspace

Description	Icon	Description
Upload and save (current page)		Upload and save all parameters included in the selected node and visible to all users. Retry is allowed upon a failure.
Upload and save (all parameters)		Upload and save all parameters included in the current device and visible to all users. Retry is allowed upon a failure.
Open and download parameter file		Open a .dat parameter file and download it to the device. Retry is allowed upon a failure.
Custom parameters		Used to edit parameter information in the custom parameter group.
Parameter Comparison		Used to check the change of the parameter values. You can compare the current values with the set values of device parameters, and view the comparison result.
Change property		Used to update the property of current device. If failed, check whether property update is already in progress.
Parameter description		Provides additional description for the selected parameter.
Move up/down		Move parameters in the customized group. You can select multiple parameters and then click "Move up" or "Move down".

Description	Icon	Description
Permissions switchover	Current permissions Standard	To limit the scope of users' access to parameters, you can set user permissions from the drop-down list, which provides three options: "Standard", "Extended", and "Expert". Restrictions: 1. When you switch "Standard" to "Expert", user permissions on all the drives in the current project will be switched to "Expert". The same rule applies to other user permission switchover cases. 2. Manufacturer permissions include "Activate" and "Exit".  3. General-purpose AC drives (such as MD810) have two permission levels: normal user and super administrator.
Parameter search	Parameter search:	Used to search for parameters in the list of parameter that can be accessed in the current node.

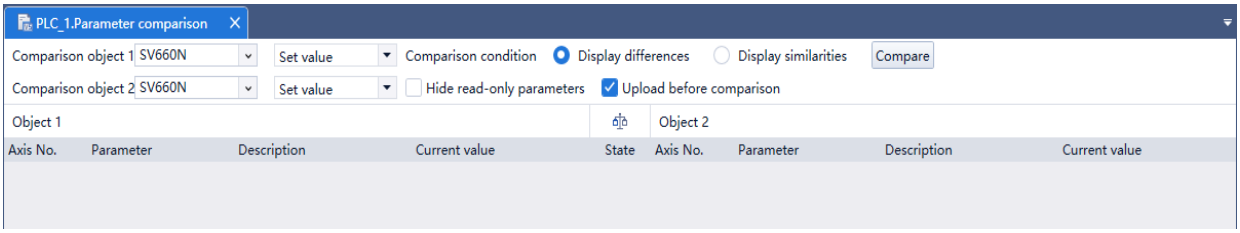
17.2.3 Parameter Comparison

This function is used to compare parameters. It only supports comparing devices of the same model under the same project. For devices with parameter tables that include set values, the comparison types include current values, default values, and set values. For devices with parameter tables that do not include set values, the comparison types include only current values and default values. This enables intuitive identification of differences between parameters, facilitating device commissioning.

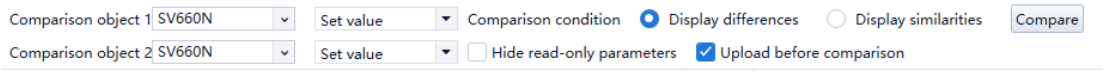
Select a device in the project tree and double-click "Parameter comparison" in the detailed view to open the parameter comparison interface.

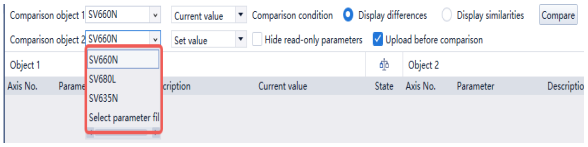
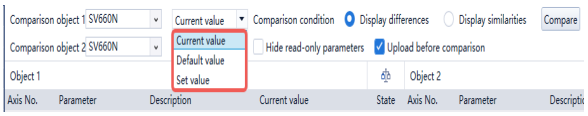
- **Function interface**

  The parameter comparison interface consists of two parts: the parameter configuration area and the comparison result area. The parameter configuration area allows you to set the parameters of the compared objects. The comparison result area displays the comparison result.



- **Parameter configuration**



Function	Description	Restriction
Comparison object	Select the object to be compared from the drop-down list. • The objects in the list are the root nodes of all devices in the current project (supporting the full series). • Select parameter file: Click "Select parameter file" to select a parameter file from the local disk as the comparison object. A parameter file is exported through software and stores corresponding device parameters. 	• When the information of the device root node in the project tree changes, both the text box and the drop-down list of the comparison object will be updated synchronously. Change types include: rename, delete, and add new devices. • There is no conflict between "Comparison object 1" and "Comparison object 2". Both can be compared with the same object. • Online devices can be compared with offline parameter files for parameters. • Only the parameter file in the DAT format is supported.
Comparison type	Select the parameter type to be compared in the drop-down list. You can select the current value, factory value, and set value. 	Only when the comparison object is set by "Select parameter file", the comparison type is forcibly set to "Set value" and the drop-down list is disabled.
Comparison condition	Select the display mode of the comparison result area. • Display same: The comparison result area displays only parameters without difference. • Display different: The comparison result area displays only parameters with difference.	When the comparison conditions are modified, the comparison results are updated immediately.

Function	Description	Restriction
Hide read-only parameters	Filter the comparison results. • When "Hide read-only parameters" is checked, the comparison result displays only the parameters that can be set. • When it is not checked, all comparison results will be displayed.	The checking and unchecking of "Hide read-only parameters" takes effect in real time and the comparison results are updated immediately.
Upload before comparison	Select whether to refresh the current value of the comparison object before comparison. • Check "Upload before comparison" to read the current value of the object from the drive before comparison. • Uncheck it to use the cached current value of the object for comparison.	It takes effect when at least one "Current value" is present in two comparison types.
Start comparing	Click the "Compare" button to start comparing. • After a successful comparison, a comparison completion window will pop up. • When comparison fails, you will be prompted that the device type does not match, and you need to select a comparison object again.	/

- **Result display**

  Parameter comparison results are displayed in a list. Data of comparison object 1 is displayed on the left, data of comparison object 2 is displayed on the right, and the comparison state is displayed in the middle. The data in the list does not support editing.

  - When the comparison finds no difference, the comparison status is green, and the data table color of comparison object 1 is the same as that of comparison object 2.
  - When the comparison finds any difference, the comparison status is a yellow exclamation mark, and the data table colors of comparison objects 1 and 2 are yellow and blue respectively.

17.2.4 Machine Parameter Management

17.2.4.1 Machine Recipe Management

For details, see Machine Recipe Management.

17.2.4.2 Machine Parameter Management

For details, see Machine Parameter Management.

17.2.4.3 Firmware Batch Update

For details, see 10.6.3.2.7.3 Batch Firmware Upgrade.

17.3 Common Servo Function

17.3.1 Overview

Before activating the servo commissioning function, an EOE connection needs to be established. For details, see 17.1.1 EOE Connection.

17.3.2 Fault Management

Faults and alarms are classified into three levels based on their severity.

- No. 1 non-resettable fault
- No. 1 resettable fault
- No. 2 resettable fault
- No. 3 resettable alarm

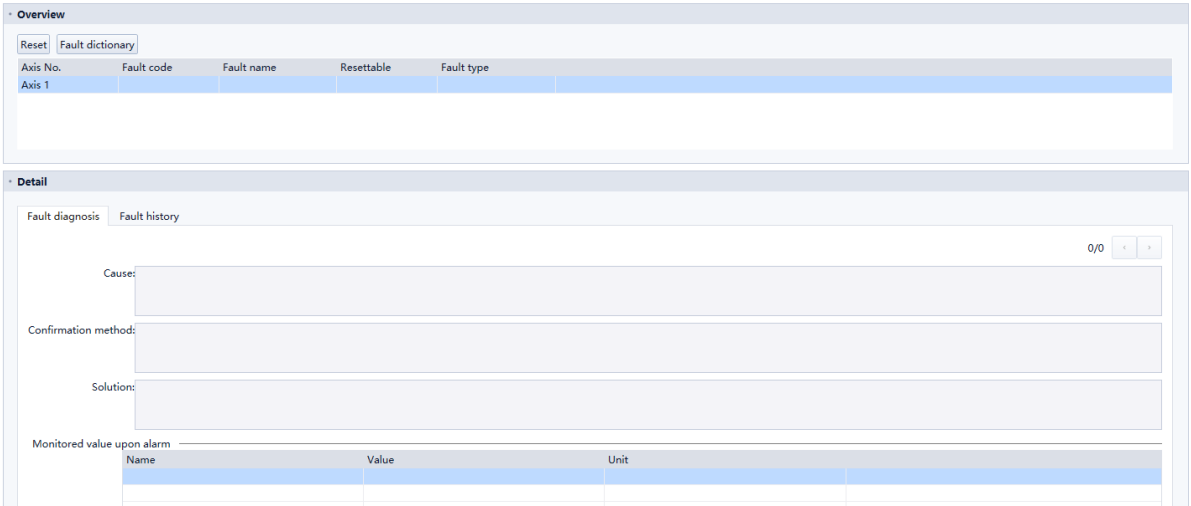
Note: Level of severity is No. 1 > No. 2 > No. 3

17.3.2.1 Servo Fault Management

Select a servo device in the project tree and double-click "Fault management" in the detailed view to display detailed information about the current device faults and their solutions.

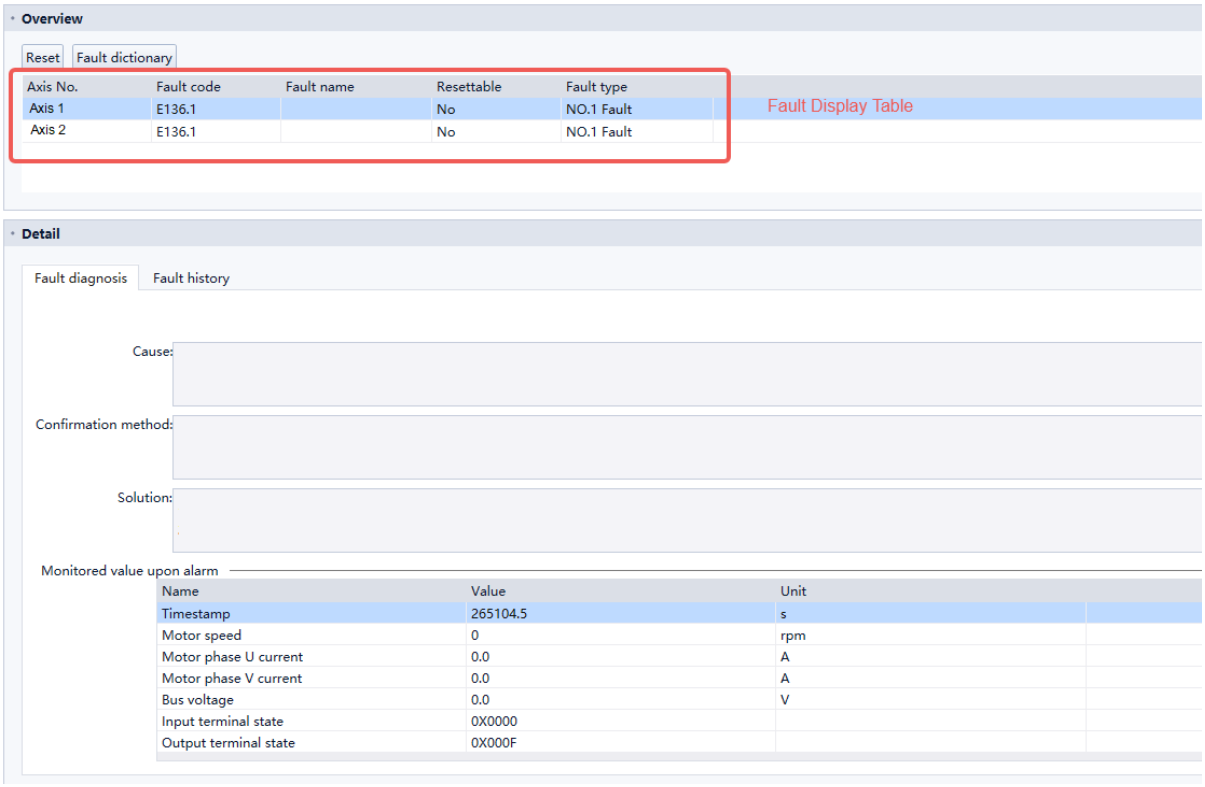
The servo fault management interface is shown in the figure below. It mainly consists of the following three parts:

- The fault overview area displays basic information about all current faults and allows "Reset" operation and access to the "Fault dictionary".
- The fault diagnosis area displays information such as the cause of the fault and its solutions.
- The fault history area displays historical fault information.



17.3.2.2 Fault Overview

The "Fault management" interface is shown in the following figure.

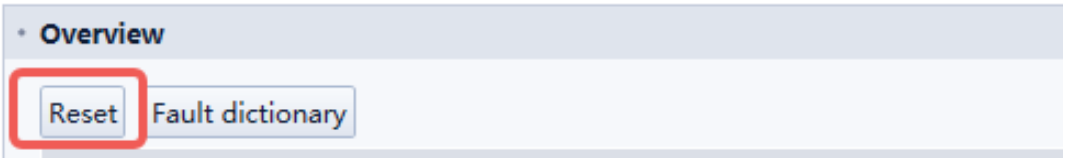


17.3.2.3 Troubleshooting

The main solution to faults is "Reset". Click "Reset" to reset the faults.

Restrictions:

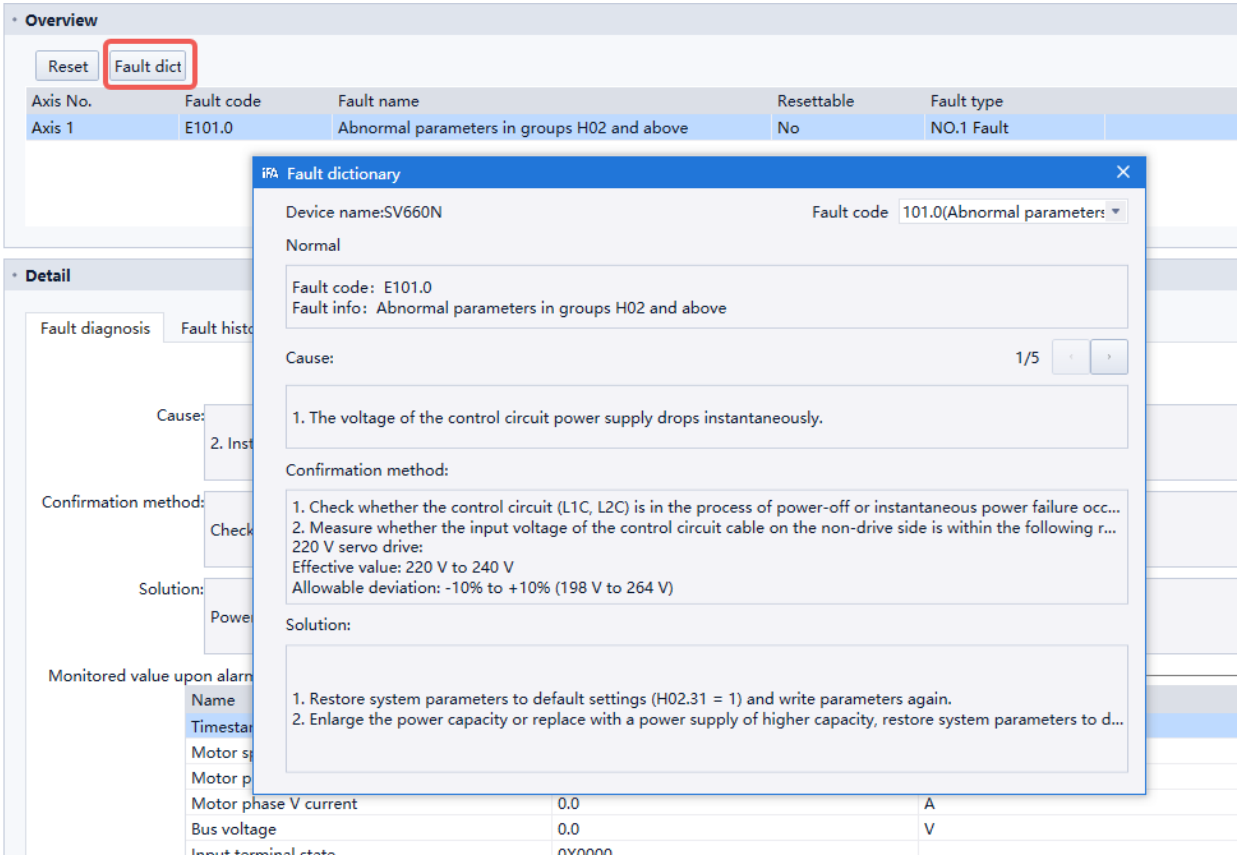
- Available only for resettable faults.
- Resettable faults can only be reset after the fault has been cleared.



17.3.2.4 Fault Dictionary

The fault dictionary displays detailed information about all faults related to the current device, including fault code, cause, confirmation method, and solution.

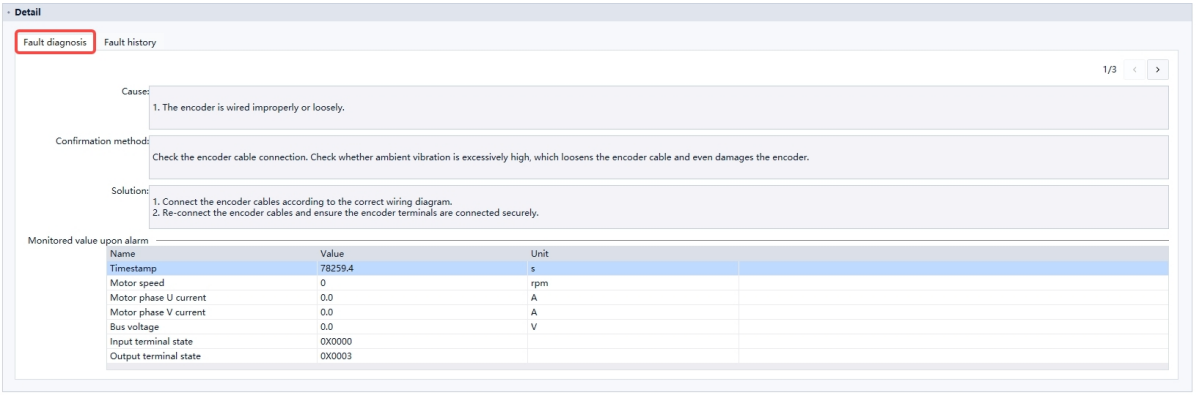
Clicking "Fault dictionary" will pop up the following interface. See the table below for detailed descriptions.



Description	Description
Fault code	The fault code to be viewed is selected from the drop-down box in the upper right corner of the interface. The options are product-specific. After selecting a fault code, the interface will display detailed information about the selected code.
Normal information	Displays the selected fault code and fault name.
Troubleshooting information	Displays the fault cause, confirmation method, and solution of the selected fault code.
Page turning buttons	If there are multiple causes for a fault, you can use these buttons to view different fault diagnosis information.

17.3.2.5 Fault Diagnostics

The fault diagnosis area displays information about the cause, confirmation method, solution of the current fault, and device parameters upon fault. The interface is shown in the figure below. For detailed function descriptions, see the table below.

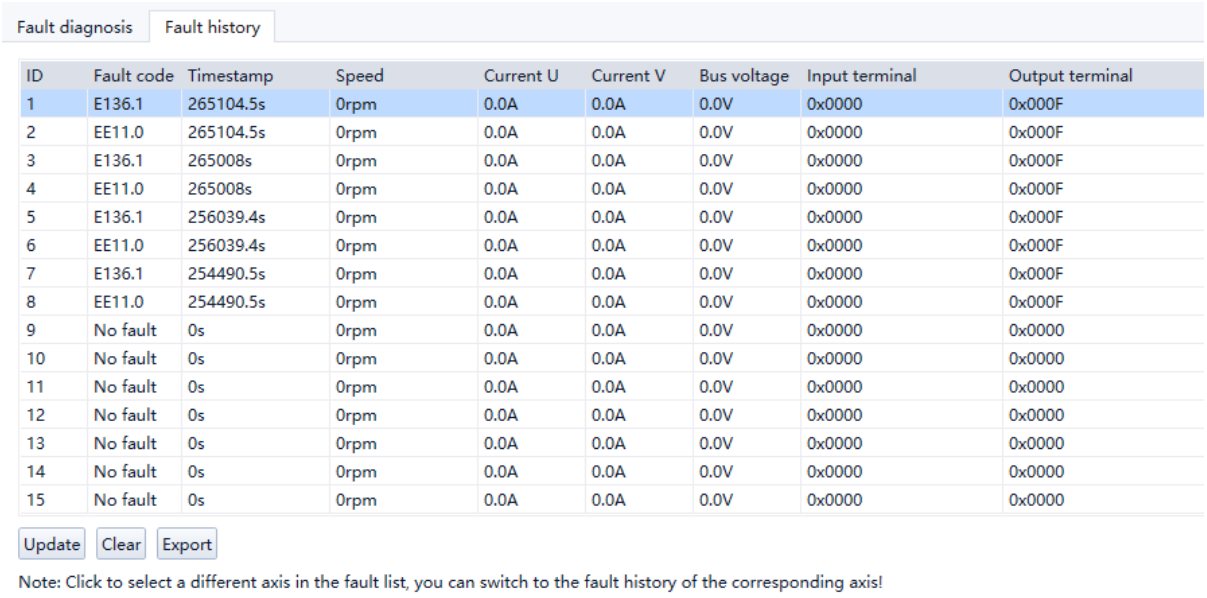


Function	Description
Cause	Displays the cause of the fault.
Confirmation method	Displays the confirmation method for the corresponding fault.
Solution	Displays the solution of the fault.
Monitored value upon alarm	Displays the name, value, and unit of the related parameters being monitored when the fault occurs.
Page turning buttons	If there are multiple causes for a fault, you can use these buttons to view different fault diagnosis information.

17.3.2.6 Fault History

The servo drive can record the latest faults and alarms, and the drive status parameters upon fault/alarm. The number of records varies among different devices.

Click "Fault history" in the "Fault management" interface to display the following interface. For detailed function descriptions, see the table below.



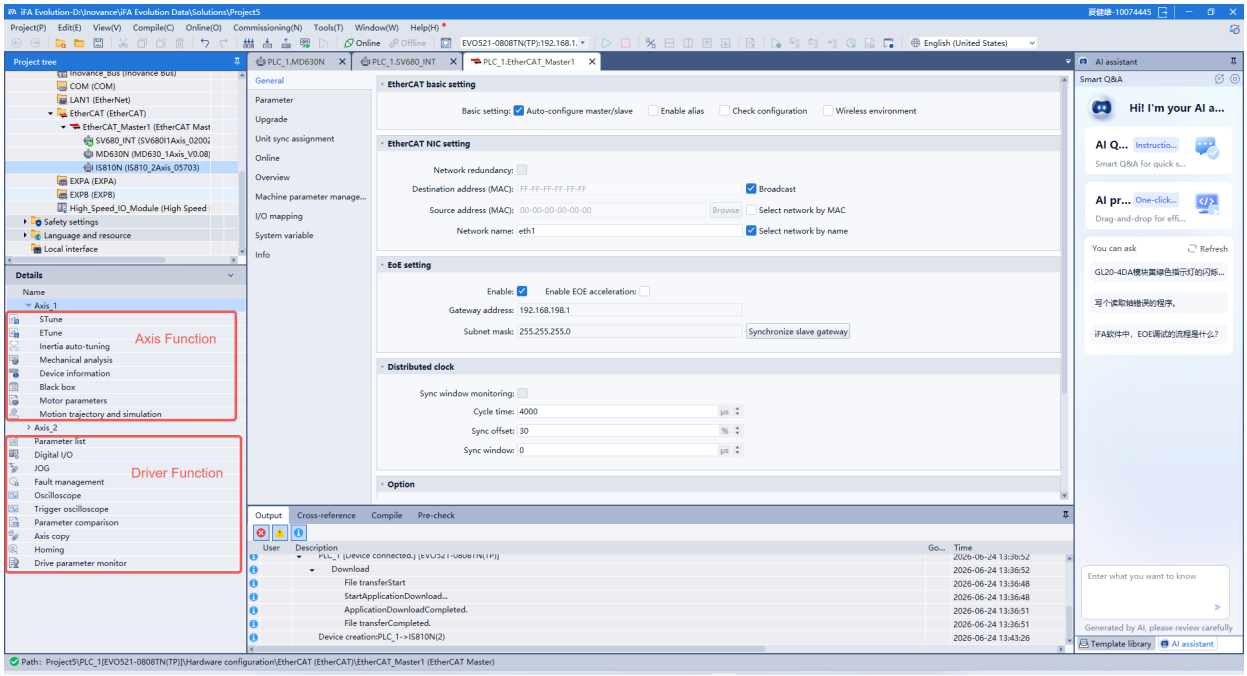
Function	Description
Update	Updates the fault history record of the current servo device.
Clear	Clears all current fault record information.
Export	Exports the current fault record information into a CSV file and save it to the selected local path.

17.3.3 Multi-Axis Device

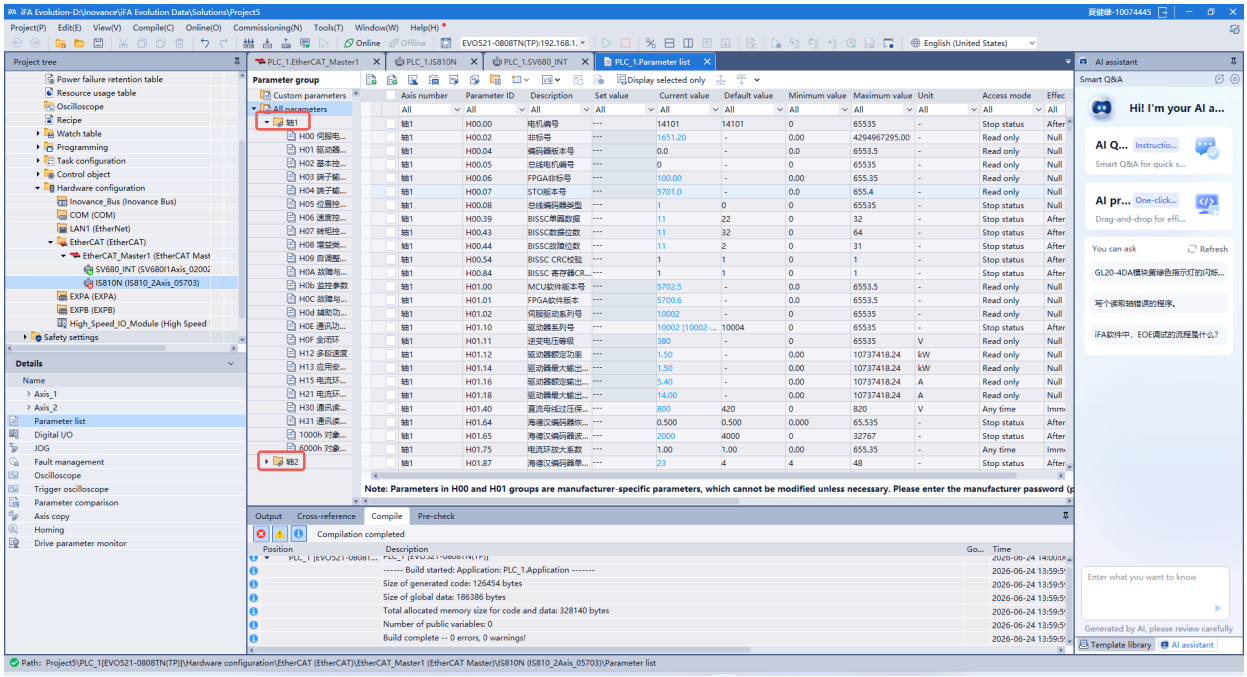
The currently integrated multi-axis devices include: SV660ND, SV630ND, and IS810N series products.

Display format for dual-axis devices

Click the device node in the project tree to display the axis function nodes and drive function nodes in the detailed view. Click the Axis_1/Axis_2 node to expand or collapse the specific axis function commissioning nodes.



Difference between axis functions and drive functions: in the opened commissioning view, drive functions allows simultaneous commissioning of multiple axes, while axis functions only allows commissioning of the corresponding single axis.



17.3.4 JOG Motion Control

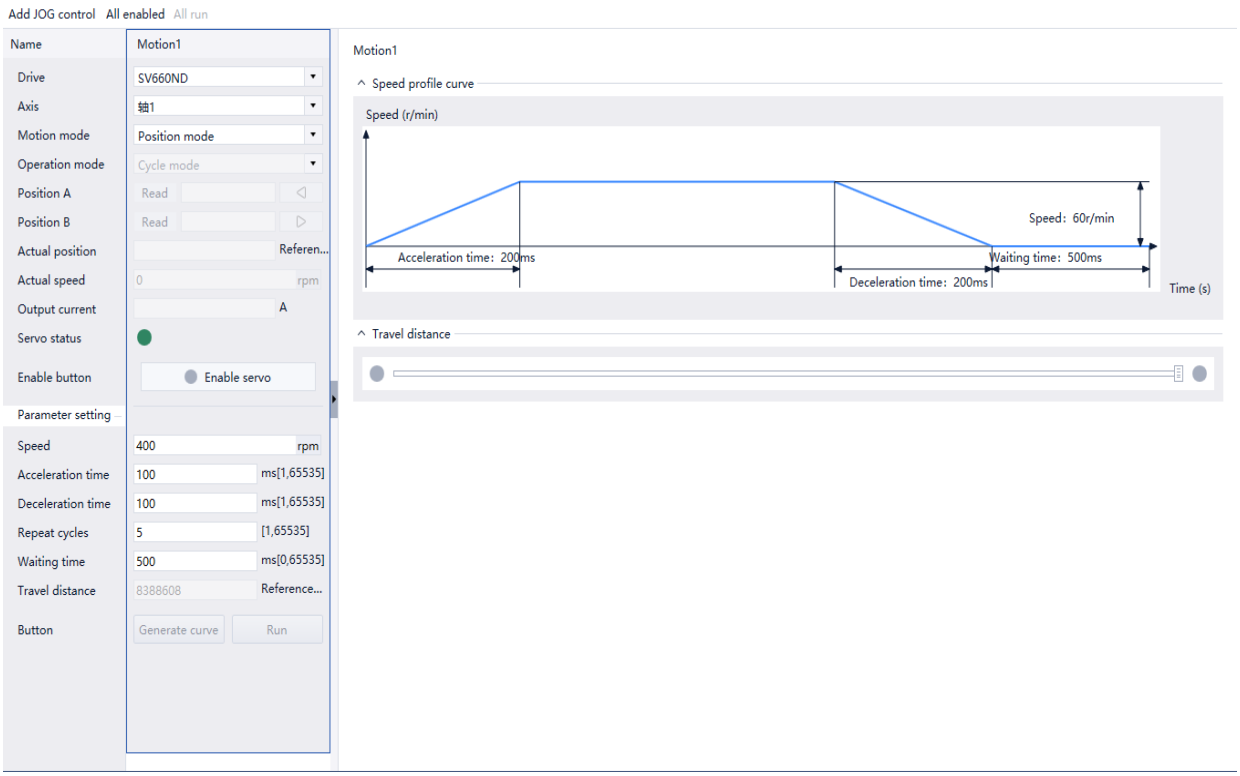
17.3.4.1 Overview

Select a servo device in the project tree and double-click "JOG" in the detailed view to open the following interface.

Jog motion control consists mainly of speed jog and position jog.

- Speed jog: It is mainly used to control the motor in the jog mode, which detects whether the motor can run and rotate properly.
- Position jog: It is mainly used to run the motor in the specified limit at the specified speed reciprocally or in a fixed distance.

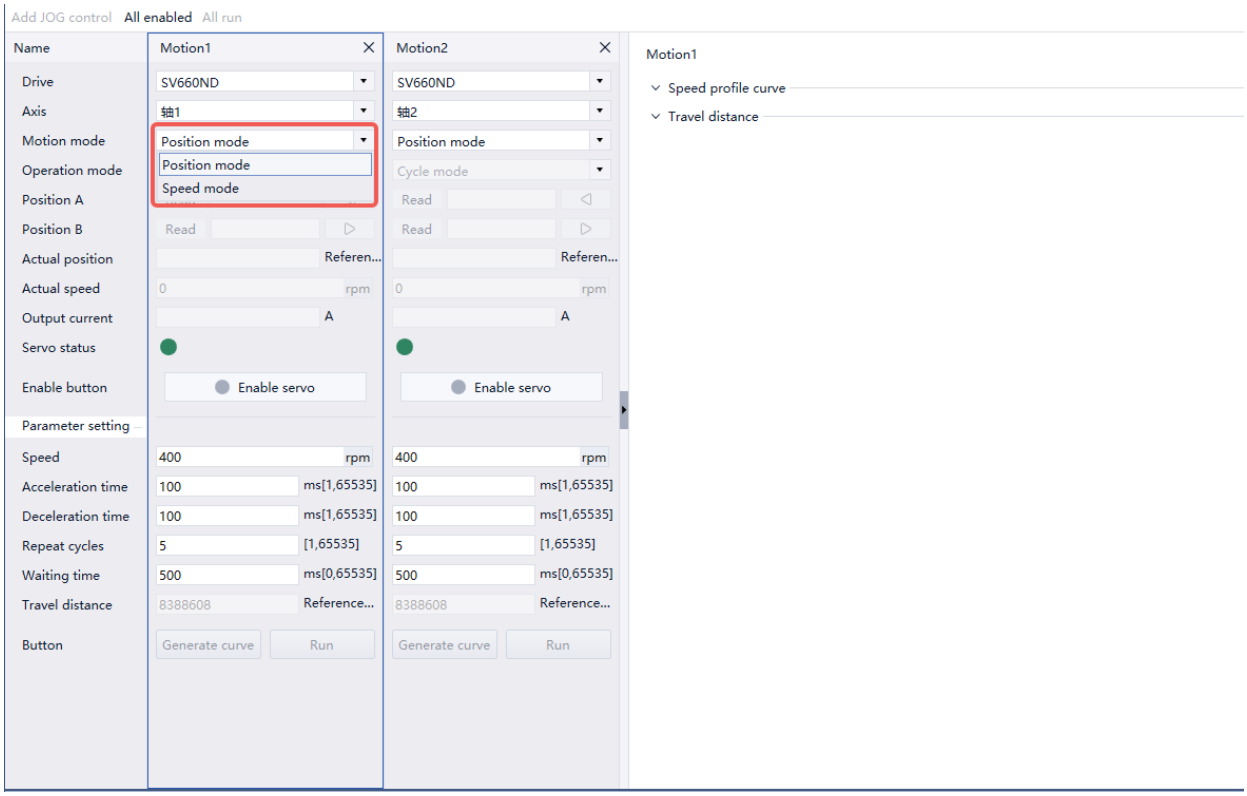
The "JOG" interface consists of the toolbar, parameter description, motion control, and chart.



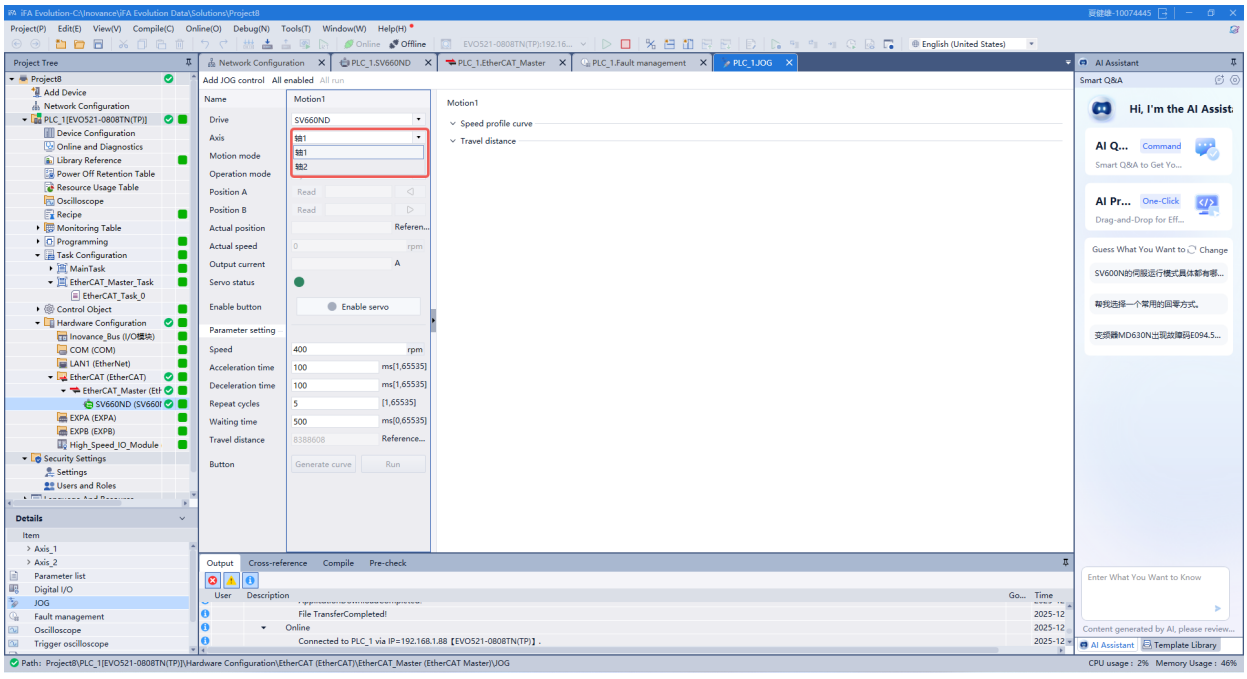
Function	Description
Add JOG control	Adds a jog panel. Restrictions: • If the number of current axes is n, up to n panels can be added. However, n cannot exceed 20. If the number of current axes is 0, no panel can be added. • The "Add JOG control" button is disabled when no jog panel can be added.
All enabled	Enables or disables all devices that have the jog control panel added on the current interface.
All run	Starts or stops all devices that have the jog control panel added on the current interface.

17.3.4.2 Mode Selection

The jog modes include position mode and speed mode.

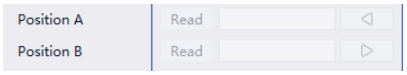


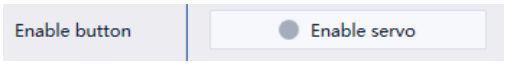
For dual-axis equipment, axis 1 or axis 2 can be selected.



17.3.4.2.1 Position JOG and Parameter Settings

This function is mainly used to run the motor in the specified limit at the specified speed reciprocally or in a fixed distance.

Function	Description
Drive	Selects the drive for position jog in the drop-down list. • The drive cannot be changed in the motion enabled status. • When the drive nodes change (added/deleted/edited) in the project tree, the drive information changes synchronously.
Axis	Selects the axis for position jog in the drop-down list. • The axis cannot be changed in the motion enabled status. • When the selected drive changes, the axis is simultaneously updated to the one currently mounted to the drive.
Motion mode	Selects the desired motion mode from the drop-down list. The motion mode cannot be changed in the motion enabled status.
Operation mode	Selects the required operation mode in the drop-down list. Options includes "Cycle mode" and "Specified distance". • In the cycle mode, the motor reciprocates. • In the specified distance mode, the motor runs at the specified distance. The operation mode cannot be changed in the motion enabled status.
Position A/Position B	 Limits the maximum limit position at both ends of the motor's reciprocating motion. • The "Read" button is used to read the current position as position A/B. • The text box displays the values of position A/B, which can be edited. • The forward/reverse running icons control the forward/reverse rotation of the motor. Restrictions: This function is available only when the motor is enabled and the operational status is not enabled.
Actual position	Displays the actual position of the current axis, which cannot be edited.
Actual speed	Displays the actual speed of the current axis, which cannot be edited.
Output current	Displays the output current, which cannot be edited.
Servo status	Displays the status of the servo drive. The icon is green when the drive is connected, and gray when the drive is disconnected.

Function	Description
Enable button	 Enables or disables the motor. The button is displayed as "Enable servo" in gray by default. Click "Enable serve". If the servo drive is enabled successfully, the button is displayed as "Disable servo" and the servo status icon becomes green. Click "Disable servo" to restore the button to the initial status. Restrictions: • When the servo drive is disconnected, the button is disabled. • When the servo drive is connected, the button is enabled. • When the interface is closed, a prompt indicating whether to stop all drive operations is displayed. If you select yes, the jog interface is closed and servo drives in jog operation will all be stopped. If you select no, the jog interface is closed but jog operations of the servo drives continue.
Speed	Defines the speed of an axis. It cannot be edited in the motion enabled status.
Acceleration time	Defines the acceleration time of an axis. It cannot be edited in the motion enabled status.
Deceleration time	Defines the deceleration time of an axis. It cannot be edited in the motion enabled status.
Repeat cycles	Defines the number of reciprocations for the motor. The parameter is available only when the operation mode is set to "Cycle mode".
Waiting time	Defines the waiting time of the motor. The parameter is available only when the operation mode is set to "Cycle mode".
Travel distance	Sets the motor to run at the specified distance. Range: -2147483648 to 2147483648. The parameter is available only when the operation mode is set to "Specified distance".
Generate curve	Generates a speed curve. Click "Generate curve" or "Run" to update the curve. When the current panel changes, the curve synchronously updates to the last updated data of the panel.
Run	Runs the motor.

Function	Description
Speed profile curve	 Displays a speed curve. • The speed curve is updated only when you click "Generate curve". • When multiple position mode panels are added, they share one speed curve chart. Click "Generate curve" of a specific axis to generate the speed curve for that axis.
Travel distance	 Tracks the motor position. • It is disabled by default. The left and right status indicators are both gray. • The interface will retain the last running status. • The left and right status indicators will be refreshed when operation starts. ◦ If the current position is larger than position A, the status indicator on the left becomes red. When the position point moves to the leftmost, the status indicator on the right becomes green. ◦ If the current position is smaller than position B, the status indicator on the right becomes red. When the position point moves to the rightmost, the status indicator on the left becomes green. ◦ If the current position is between position A and position B, both the left and right status indicators become green. The position point is displayed between position A and position B proportionally according to the current position. • Tracking is performed only when you click "Run". Tracking stops after the operation is disabled.

CAUTION

For specified distance operation in the position mode, the following conditions must be met:

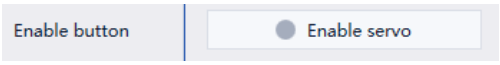
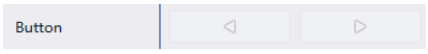
- During startup, the current position must be within the range of position B to position A.
- When you set the distance, the motion of the motor must be within the range of position B to position A. Otherwise, the motor cannot complete the desired operation.

17.3.4.2.2 Speed JOG and Parameter Settings

STD60N series does not support speed jog.

This function is mainly used to control the motor in the jog mode, which detects whether the motor can run and rotate properly.

Function	Description
Drive	Selects the drive for speed jog in the drop-down list. • The drive cannot be changed in the motion enabled status. • When the drive nodes change (added/deleted/edited) in the project, the drive information changes synchronously.
Axis	Selects the axis for speed jog in the drop-down list. • The axis cannot be changed in the motion enabled status. • When the selected drive changes, the axis is simultaneously updated to the one currently mounted to the drive.
Motion mode	Selects the desired motion mode from the drop-down list. The motion mode cannot be changed in the motion enabled status.
Actual position	Displays the actual position of the current axis, which cannot be edited.
Actual speed	Displays the actual speed of the current axis, which cannot be edited.
Output current	Displays the output current, which cannot be edited.
Servo status	Displays the status of the servo drive. The icon is green when the drive is connected, and gray when the drive is disconnected.

Function	Description
Enable button	 Enables or disables the motor. The button is displayed as "Enable servo" in gray by default. Click "Enable serve". If the servo drive is enabled successfully, the button is displayed as "Disable servo" and the servo status icon becomes green. Click "Disable servo" to restore the button to the initial status. Restrictions: • When the servo drive is disconnected, the button is disabled. • When the servo drive is connected, the button is enabled. • When the interface is closed, a prompt indicating whether to stop all drive operations is displayed. If you select yes, the jog interface is closed and servo drives in jog operation will all be stopped. If you select no, the jog interface is closed but jog operations of the servo drives continue.
Speed	Defines the speed of an axis. Modification is prohibited when the servo drive performs forward and reverse running.
Acceleration time	Defines the acceleration time of an axis. Modification is prohibited when the servo drive performs forward and reverse running.
Deceleration time	Defines the deceleration time of an axis. Modification is prohibited when the servo drive performs forward and reverse running.
Forward/ reverse running icons	 Controls forward/reverse running of an axis. Long press the icon to start forward/reverse running and release it to stop. When the servo drive is enabled, the forward/reverse icon is enabled. When the servo drive is disabled, the forward/reverse icon is disabled.

NOTE

The acceleration time of SV680N indicates the time required to accelerate from 0 rpm to 1000 rpm. The deceleration time indicates the time required to decelerate from 1000 rpm to 0 rpm.

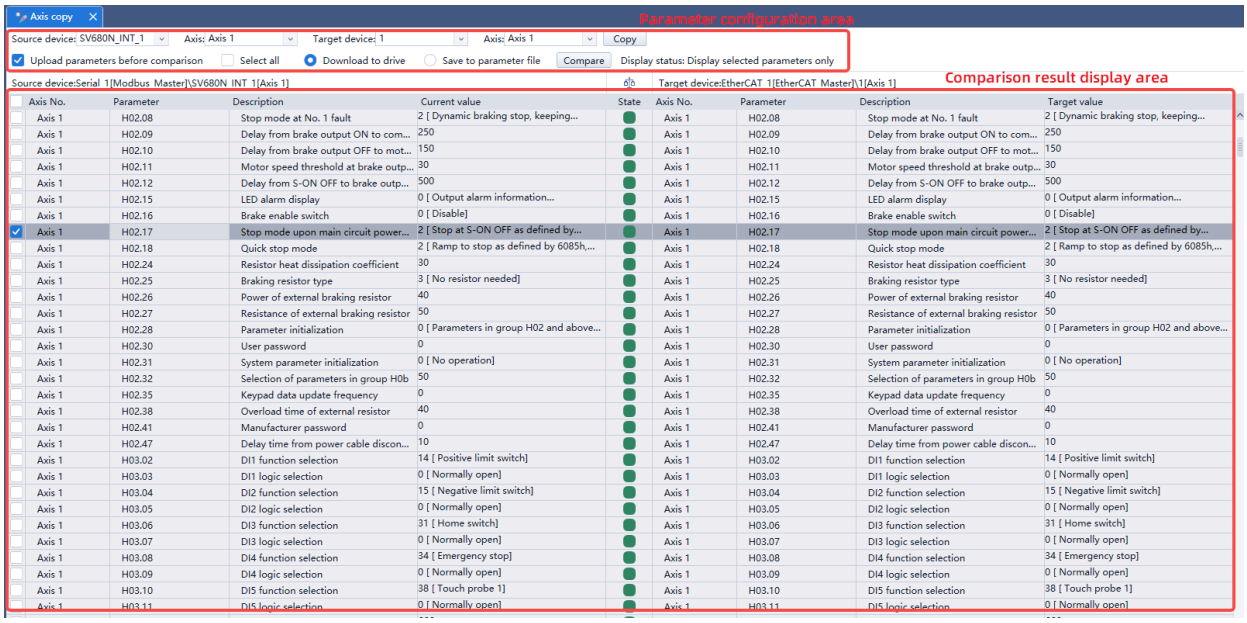
17.3.5 Axis Copy

This function is used to copy the current values of related axis parameters of the source device to the target device. Two devices must be of the same series. You can quickly copy the parameters of the source device to the target device through axis copying.

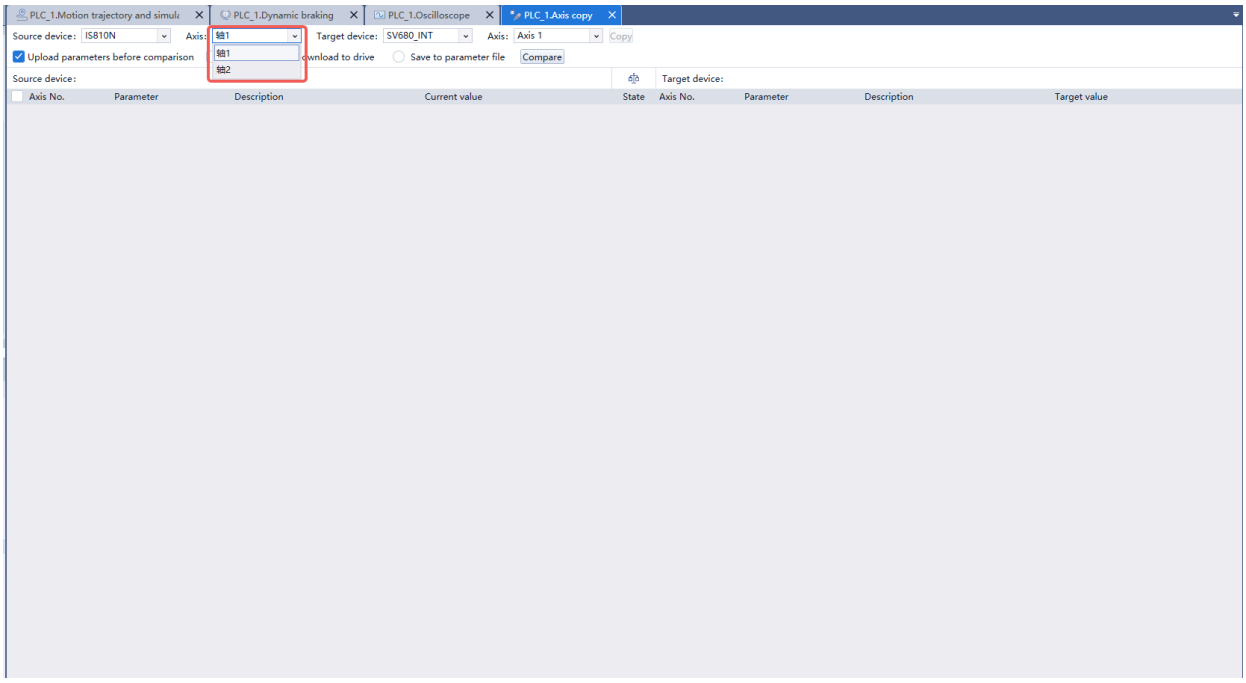
Select a servo drive in the project tree and double-click "Axis copy" in the detailed view to open the interface.

- **Function interface**

  The "Axis copy" interface consists of two parts: parameter configuration area and comparison result area. Before copying the axis parameters, compare the parameters of the two devices and copy the parameters of the source device to the target device based on the comparison results, as shown in the figure below.



Axis 1 and Axis 2 functions are available for dual-axis device.



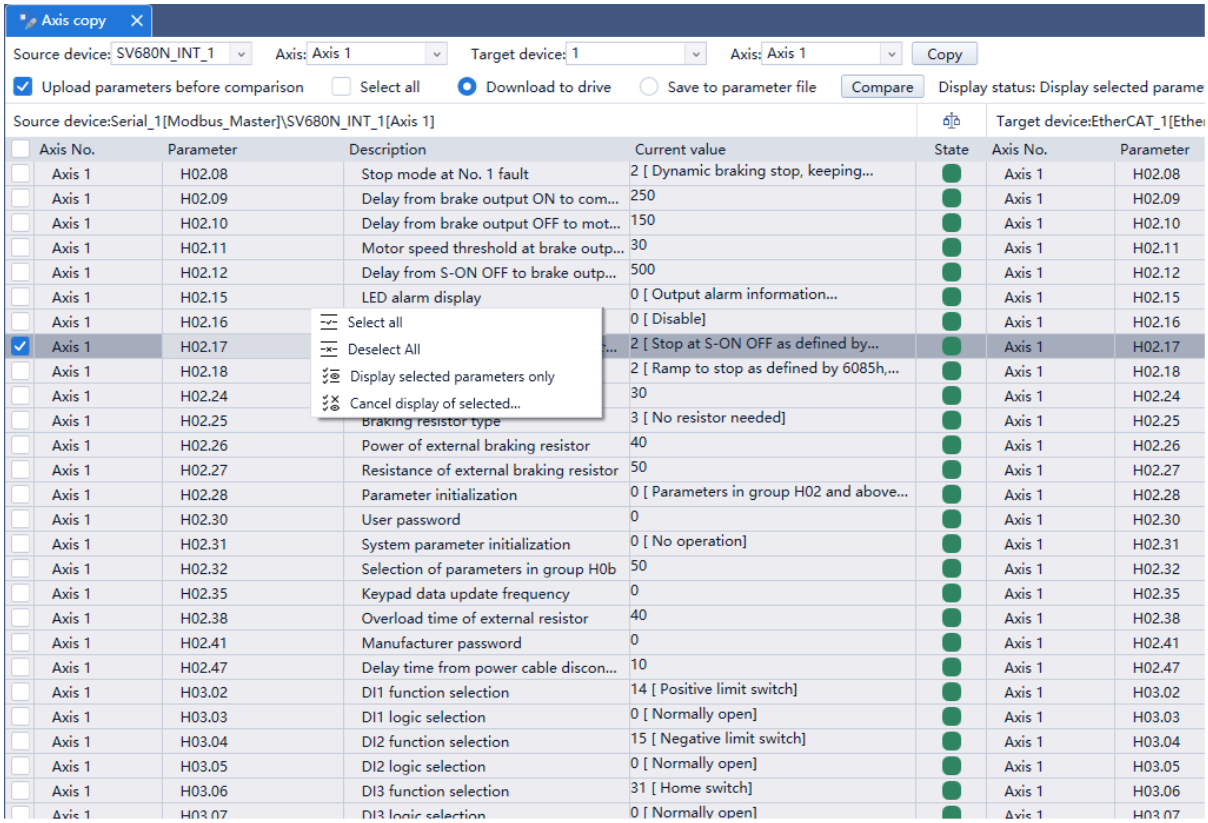
- **Parameter configuration**

Function	Description	Restriction
Source device	Selects the data source object to be copied in the drop-down list. All servo root nodes in the current project can be selected. It is only applicable to SV660N, SV680N, STD60N, SV630N, SV630NS, SV670N and IS680N-PLUS series products.	• When the information of the servo root node changes (renamed/deleted/added), both the text box and the drop-down list of the comparison object will be updated synchronously. • The "Source device" and "Target device" are independent. Different axes of the same type servo drive can be selected on both devices for comparison and copying.
Axis (source device)	Selects the data source object to be copied. Options include all axes under the source device.	When the "Source device" and "Target device" are the same servo drive and the same axis, comparison and copy cannot be performed.
Target device	Selects the target object to be copied in the drop-down list. All servo root nodes in the current project can be selected. It is only applicable to SV660N, SV680N, STD60N, SV630N, SV630NS, SV670N and IS680N-PLUS series products.	• When the information of the servo root node changes (renamed/deleted/added), both the text box and the drop-down list of the comparison object will be updated synchronously. • The "Source device" and "Target device" are independent. Different axes of the same type servo drive can be selected on both devices for comparison and copying.
Axis (target device)	Selects the target object to be copied. Options include all axes under the target device.	When the "Source device" and "Target device" are the same servo drive and the same axis, comparison and copy cannot be performed.

Function	Description	Restriction
Upload parameters before comparison	Check to refresh the current values of the comparison objects (the source and target devices) before comparison. • If it is checked, the current value of the object from the drive is read before comparison. • If it is unchecked, the cached current value of the object is used for comparison.	/
Compare	Click "Compare" to compare the current values of "Source device" and "Target device".	/
Select all	Selects or deselects all parameters in the comparison results.	/
Download to drive	When "Download to drive" is checked, click "Copy" to download the current values of the parameters checked on the source device directly to the target device.	/
Save to parameter file	When "Save to parameter file" is checked, click "Copy" to save the current values of the parameters checked on the source device to the local disk.	The file can only be in the DAT format.
Copy	After selecting the parameters to be copied in the comparison results, click the "Copy" button to execute the corresponding copy operation. • If "Download to drive" is checked, the current values are copy (downloaded) to the target device. • If "Save to parameter file" is checked, the current values are saved to the local drive.	Copy can only be performed when there are selected parameters in the comparison results.

- **Right-click menu**

  Right-click the axis copy result display area to perform the following operations to the list: selecting all, unselecting all, displaying selected parameters, and displaying all, as shown in the figure below.



- **Result display**

  The parameter comparison results are displayed in a list, with source device data on the left, target device data on the right, and the comparison status in the middle. The data in the list does not support editing.

  - When the comparison finds no difference, the comparison status is green, and the data table colors of the source and target devices are the same.
  - When the comparison finds any difference, the comparison status is a yellow exclamation mark, and the data table colors of source and target devices are yellow and blue respectively.

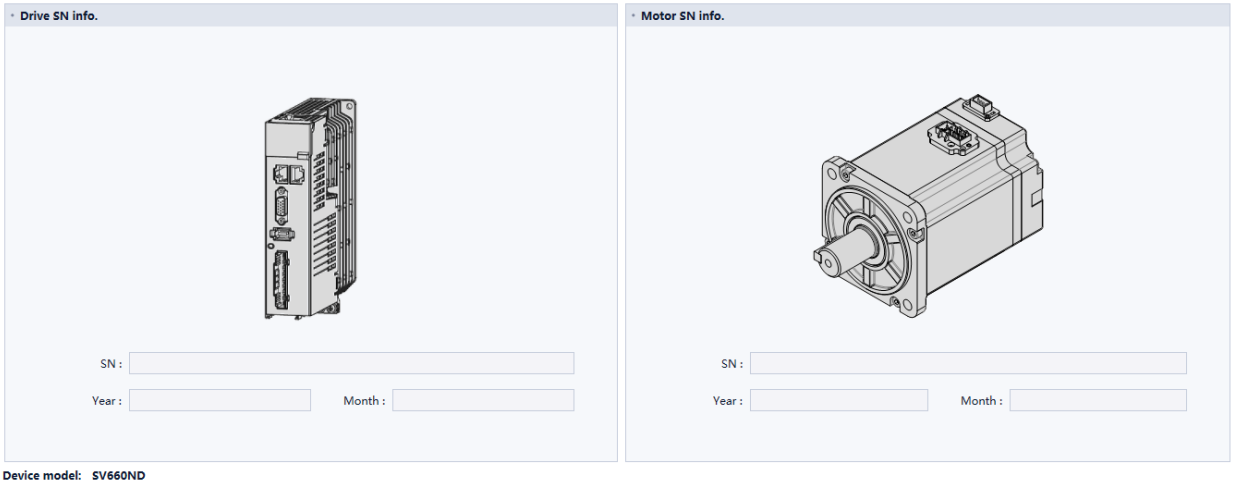
17.3.6 Device Information

This function is currently not supported for IS810N series products.

Select a servo drive in the project tree, and double-click "Device information" in the detailed view to open the interface.

The "Device information" interface consists of the following three parts:

- Axis selection: Selects the axis for the current device and refresh motor information.
- Drive SN info.: Displays the serial number and date of the current drive device.
- Motor SN info.: Displays the serial number and date of the motor connected to the current device.



- **Axis selection**

  If the current device has multiple axes, you can switch between them using the drop-down box.

  After the axis is switched, the motor device information will be synchronized and refreshed if a motor is connected.

- **Drive SN info.**

  Displays the drive illustration, serial number (SN code), and date information (year and month).

  If the current drive device is offline, the serial number and date information will be empty.

- **Motor SN info.**

  Displays the motor illustration, serial number (SN code), and date information (year and month).

  If the current drive device is offline or no motor device is connected, the serial number and date information will be empty.

17.3.7 Drive Parameter Monitoring

This function monitors the real-time values of axis parameters in the operation status of the servo drive. The monitored axis parameter objects are configured by the user.

Select a servo device in the project tree, and double-click "Drive parameter monitor" in the detailed view to open the interface, as shown in the following figure.

Restrictions:

- The drive monitoring table supports a maximum of 200 parameters, and the number of distributed servo slaves does not exceed 16.



	Device name	Axis name	Parameter	Parameter name	Value
1	SV660N_1	Axis1	H19.45	Target value of position comparison 16	
2	SV660N_1	Axis1	H19.48	Target value of position comparison 17	
3	SV660N_1	Axis1	H19.57	Target value of position comparison 20	
4	SV660N_1	Axis1	H19.59	Attribute value of position comparison 20	
5	SV660N_1	Axis1	H19.60	Target value of position comparison 21	
6	SV660N_1	Axis1	H19.62	Attribute value of position comparison 21	
7	SV660N_1	Axis1	H19.63	Target value of position comparison 22	
8	SV660N_1	Axis1	H19.65	Attribute value of position comparison 22	
9	SV660N_1	Axis1	H19.66	Target value of position comparison 23	
10	SV660N_1	Axis1	H19.68	Attribute value of position comparison 23	
11	SV660N_1	Axis1	H19.69	Target value of position comparison 24	
12	SV660N_1	Axis1	H19.71	Attribute value of position comparison 24	
13	SV660N_1	Axis1	H19.72	Target value of position comparison 25	
14	SV660N_1	Axis1	H19.74	Attribute value of position comparison 25	
15	SV660N_1	Axis1	H19.75	Target value of position comparison 26	
16	SV660N_1	Axis1	H19.77	Attribute value of position comparison 26	

Function	Description
	Moves down the selected row. Multiple rows can be moved simultaneously.
	Moves up the selected row. Multiple rows can be moved simultaneously.
	Deletes the selected row, and the row below will be shifted up.
	Enables monitoring. • Enabled: The PLC reads and displays the real-time values of the slave parameters.
	Disables monitoring. • Disabled: The PLC does not read real-time values.
Device name	The name of the servo slave in the project; if the name changes, it will automatically synchronize here.
Axis name	The axis number configured internally in the servo, independent of the motion control axis name.
Parameter	Parameter format: H00.00.
Parameter name	Description of the parameter.
Value	Indicates the real-time value of an axis parameter, which can only be monitored, but does not support writing. The refreshing period of the value is 1s. Displays the real-time value when the slave communicates with the PLC normally. If the communication is abnormal, it displays "Device not connected".

17.3.8 Mechanical Characteristic Analysis

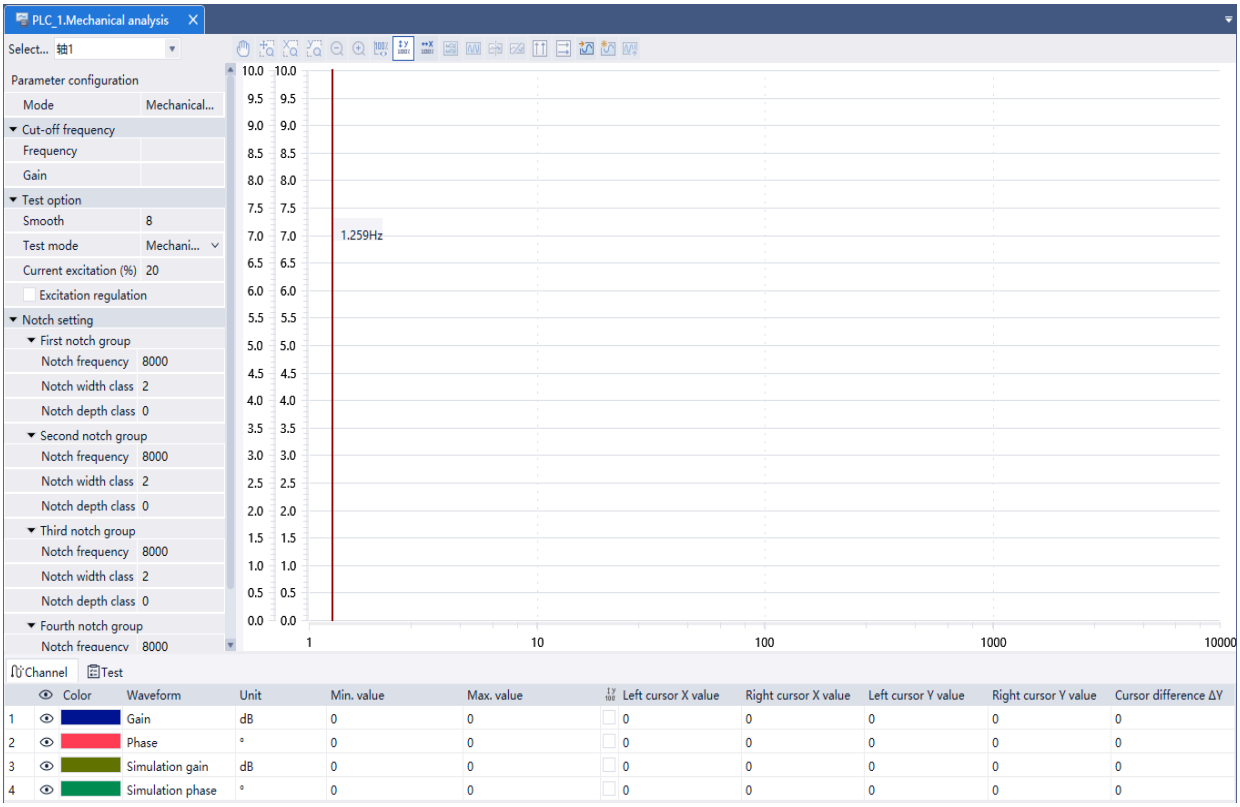
17.3.8.1 Mechanical Characteristics

This function determines the mechanical resonance point and system bandwidth. It supports three test modes: mechanical characteristics, speed open loop, and speed closed loop.

Select a servo device in the project tree, and double-click "Mechanical analysis" in the detailed view to open the interface. Select the "Mechanical characteristics" mode, as shown in the figure below.

- **Function interface**

  The mechanical analysis interface consists of three parts: parameter configuration area, Bode diagram display area, and channel/test area.

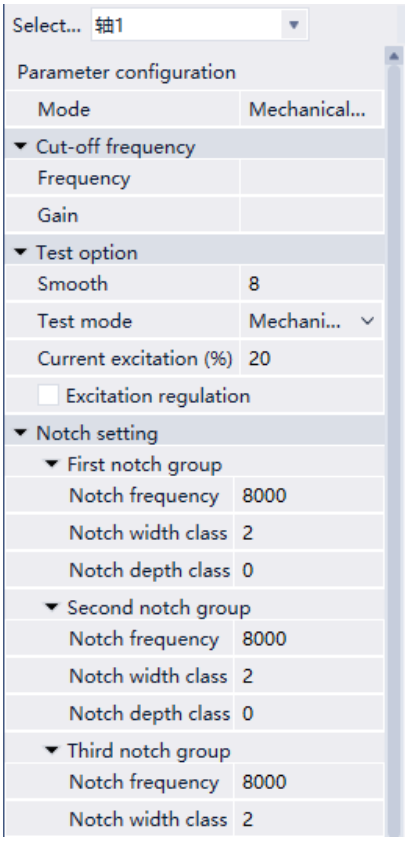


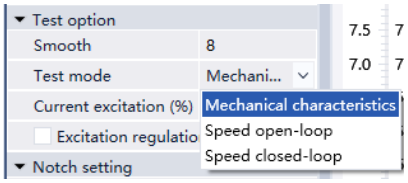
Note: Each servo drive node of the same project has a separate mechanical characteristic analysis view.

- **Basic operating process**

  Configure parameters in the "Parameter configuration" area, click "Start" on the "Test" tab, wait for the test to complete, and view the Bode diagram for analysis.

- **Parameter configuration**

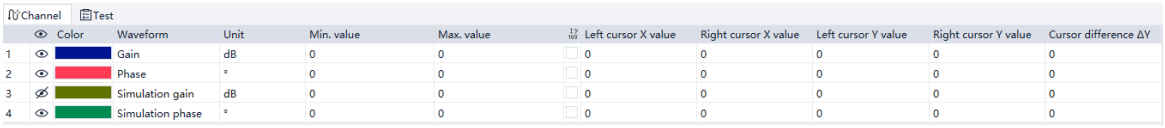


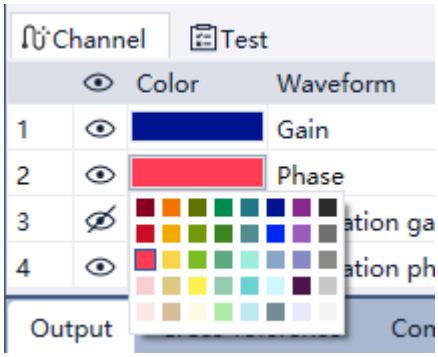
Function		Description	Restriction
Selected axis		Selects the axis number required for mechanical characteristic analysis.	/
Mode		Selects "Mechanical characteristics".	/
Cut-off frequency	Frequency	Indicates the cut-off frequency displayed after the mechanical characteristics analysis is done.	Read-only
	Gain	Indicates the gain corresponding to the cut-off frequency displayed after the mechanical characteristics analysis is done.	Read-only
Test option	Smooth	Sets the smoothness of the curve.	Value range: An integer within 0 to 25.
	Test mode	Defines the test mode for mechanical analysis. Options include "Mechanical characteristics", "Speed open-loop", and "Speed closed-loop". 	/
	Current excitation	Sets the current excitation value for mechanical characteristic analysis.	Value range: An integer within 0 to 150.
	Excitation regulation	Enables or disables the excitation regulation for mechanical characteristic analysis.	/

Function		Description	Restriction
Notch setting	Notch parameters	There are parameters of four notch groups, including the notch frequency, width class, and depth class.	• Notch frequency range: 50 to 8000 Hz. • Notch width class range: 0 to 20. • Notch depth class range: 0 to 99. The preceding parameters are all integer values.

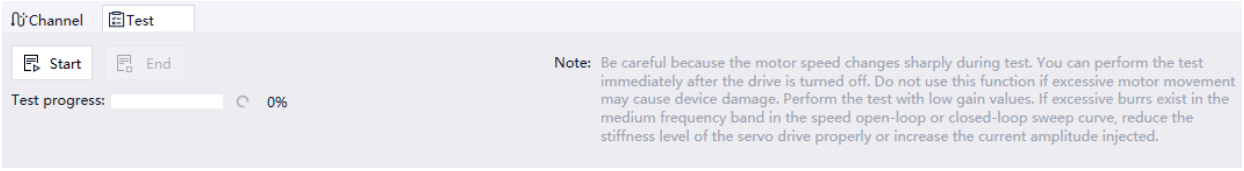
- **Channel tab**

The channel tab is used to set the curve display and data display styles in the Bode diagram.



Function	Description	Restriction
Display/hide	Displays or hides the current curve.	/
Color	Sets the color of the channel curve. 	/
Waveform	Displays the variable name of the current channel.	Read-only
Unit	Displays the unit of the current channel.	Read-only
Min. value	Displays the minimum value of the current channel.	Read-only
Max. value	Displays the maximum value of the current channel.	Read-only
Left cursor X/Y value	Displays the X/Y value corresponding to the current channel left cursor.	Read-only
Right cursor X/Y value	Displays the Y value corresponding to the current channel right cursor.	Read-only
Cursor difference ΔY	Displays the difference between the left and right cursor values.	Read-only

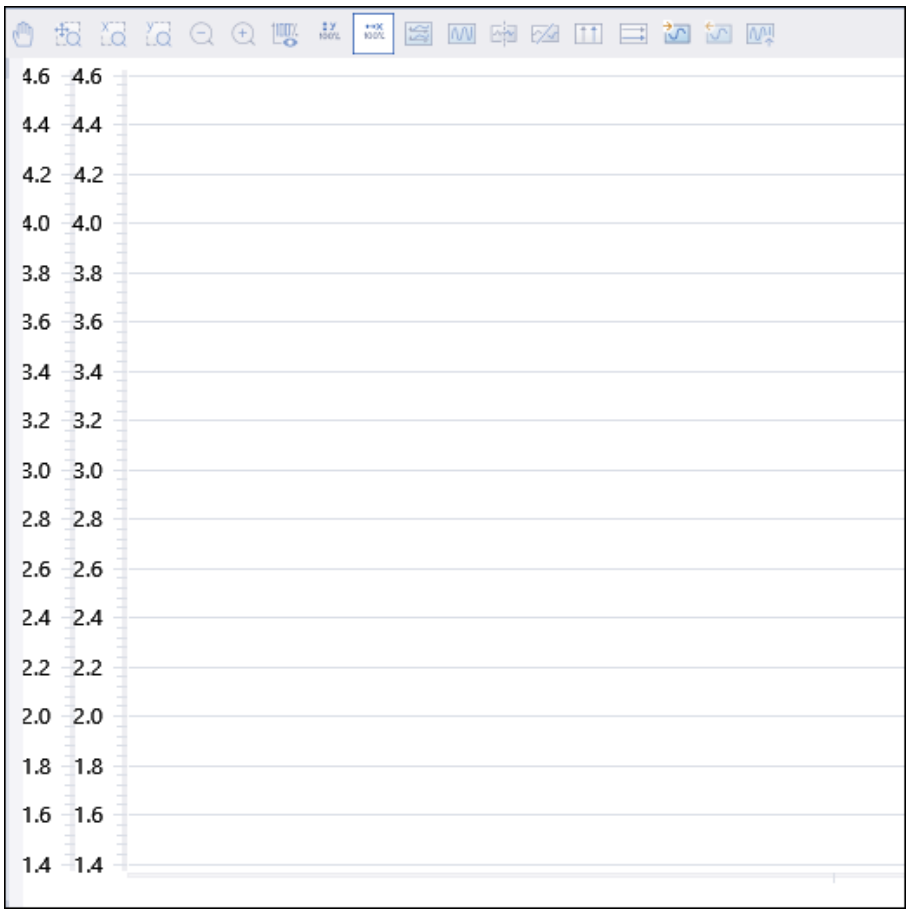
- **Test tab**



Function	Description	Restriction
Test mode	Displays the current mode of the drive after the test.	It is not displayed when "Mechanical characteristics" is selected.
Start	Starts the mechanical characteristics analysis and displays the Bode diagram.	/
End	Ends the test.	The button is available during the test and unavailable when the test is not started.
Test progress	Displays the data collection progress of mechanical characteristic analysis.	/
Note	Displays note information during the test.	/

- **Bode diagram**

The test results of mechanical characteristic analysis are displayed as a Bode diagram curve.



The functions of generating analog curve and exporting analog curve data are applicable only to the high-performance auto-tuning. For details, see 17.3.8.2 High-Performance Tuning.

Function		Description	Restriction
	Pan	When it is enabled, you can move and control the view conveniently with the mouse.	• It is mutually exclusive to the zoom function. • It is unavailable when there is no curve.
	Zoom	Zooms in/out the curve in the view. You can select a waveform segment on the X- and Y-axis for observation.	• It is mutually exclusive to the drag function. • It is unavailable when there is no curve.
	Horizontal zoom	Zooms the curve on the X axis.	• It is mutually exclusive to the drag function. • It is unavailable when there is no curve.
	Vertical zoom	Zooms the curve on the Y axis.	• It is mutually exclusive to the drag function. • It is unavailable when there is no curve.
	Zoom -	Zooms out on the entire curve.	It is unavailable when there is no curve.
	Zoom +	Zooms in on the entire curve.	It is unavailable when there is no curve.
	Show all	Displays 100% of the X and Y axes. When this function is enabled, the icons buttons for displaying all on Y-axis and X-axis are selected.	It is unavailable when there is no curve.
	Y-axis auto scale	Displays 100% of only the Y axis.	It is unavailable when there is no curve.
	X-axis auto scale	Displays 100% of only the X axis.	It is unavailable when there is no curve.
	Group display	Displays curves by group.	• This function can only be used when there are four curves. • It is unavailable when there is no curve.

Function		Description	Restriction
	Generate simulation curve	After modifying the configuration parameters for high-performance tuning, click the "Generate simulation curve" icon to generate two simulated curves.	• This is only available after the high-performance auto-tuning test is done. • This button is unavailable in the mechanical characteristic analysis mode.
	Compare waveform	Imports the CSV waveform file, and compares it with the waveform on the current interface.	• This function is only available when a waveform is present on the current interface. • High-performance tuning curves cannot be compared with mechanical characteristic curves. • It is unavailable when there is no curve.
	Cancel comparison	Cancels the waveform comparison and deletes the imported comparison waveform curves.	/
	Vertical cursor	Displays two vertical cursors (left and right); the intersection values of each vertical cursor with the curve can be read in the channel. 	It is unavailable when there is no curve.
	Horizontal cursor	Displays two horizontal cursors (upper and lower); the intersection values of each horizontal cursor with the curve can be read in the channel.	Horizontal cursor measurement is not currently supported.
	Import	Imports the CSV waveform file.	/
	Export	Exports the waveform curve to a CSV file and saves it to the local drive.	It is unavailable when there is no curve.
	Export simulation curve	Exports the simulation curve data to a CSV file and saves it to the local drive.	It cannot be used when there is no simulation curve.

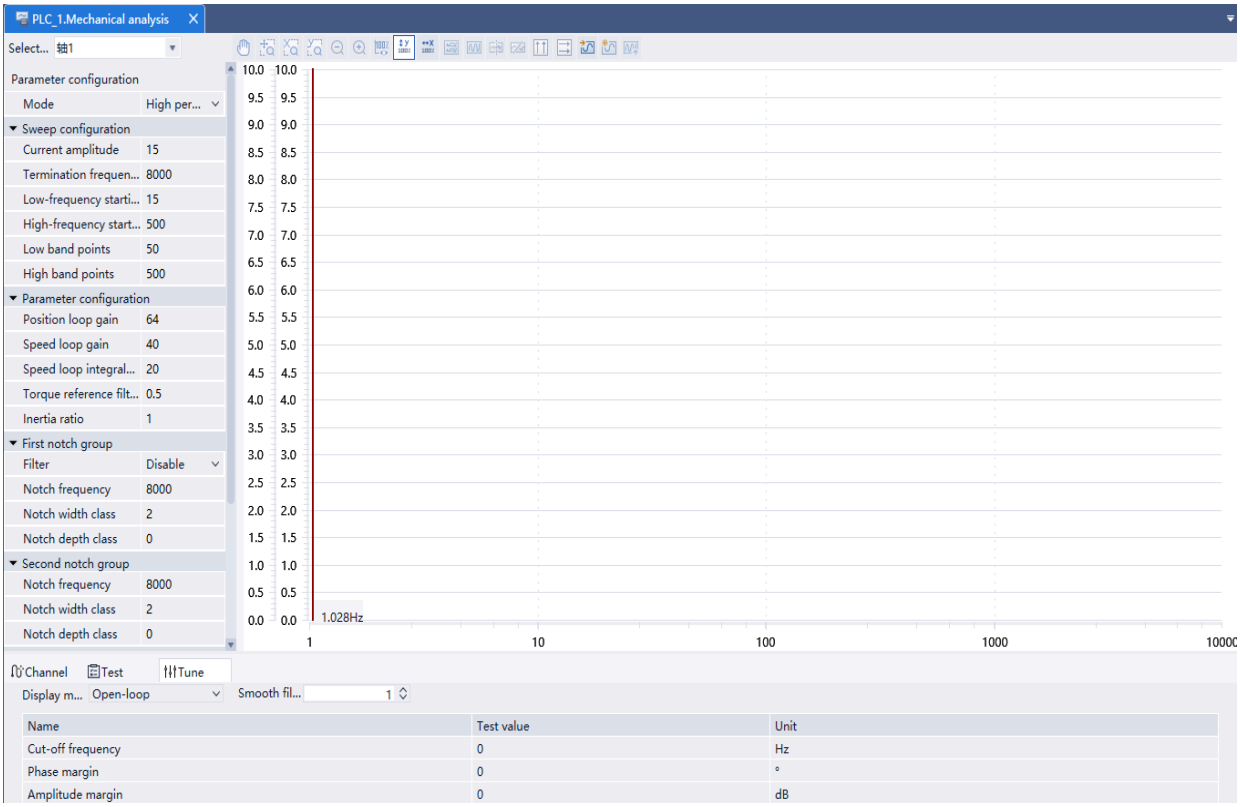
17.3.8.2 High-Performance Tuning

This feature currently supports only certain product series.

In high-performance auto-tuning mode, the characteristic analysis is automatically performed on a servo drive through built-in algorithms to obtain the relative optimal values of related parameters. This function eases the servo drive performance commissioning, precisely suppresses high-frequency resonance, and improves the servo drive stability and responsiveness.

- **Function interface**

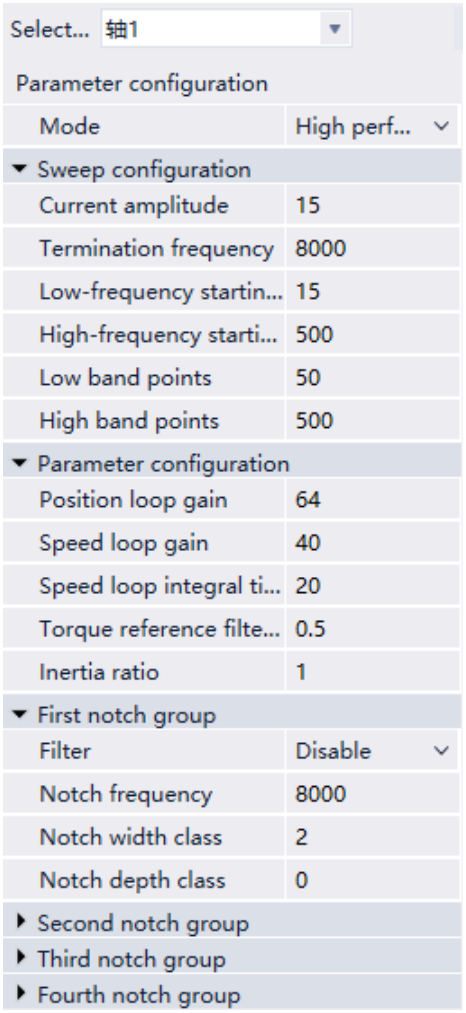
  Select a servo device in the project tree, and double-click "Mechanical analysis" in the detailed view to open the interface. Select the "High performance tuning" mode. The interface consists of three parts: parameter configuration area, Bode diagram display area, and channel/test/tuning tabs, as shown in the following figure.



- **Basic operating process**

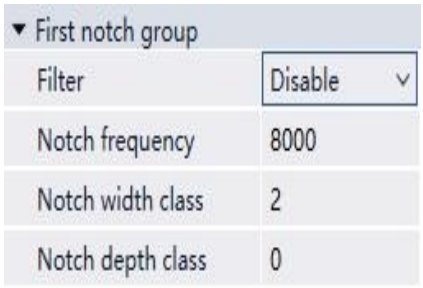
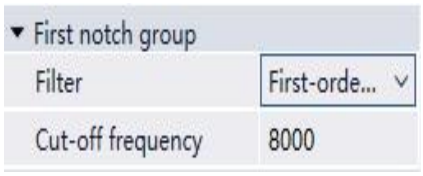
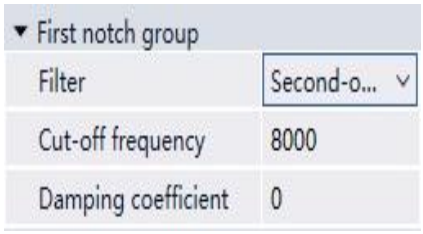
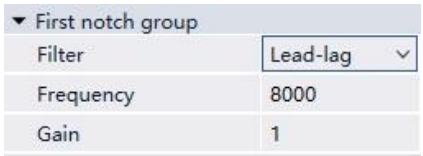
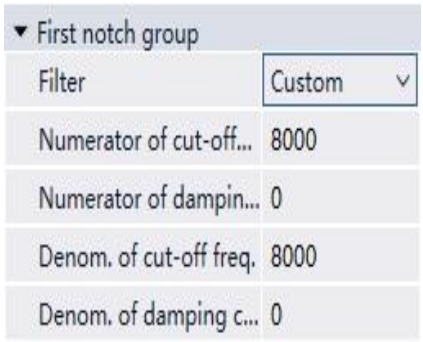
  Configure parameters in the "Parameter configuration" area, and click "Start" on the "Test" tab. Wait for the test to complete, click the "Start" button on the "Tune" tab, and view the Bode diagram for analysis.

- **Parameter configuration**



Function		Description	Restriction
Selected axis		Selects the axis No. to perform high-performance auto-tuning.	/
Mode		Selects "High performance tuning".	/
Sweep configuration	Current amplitude	Defines the current amplitude of the auto-tuned sweep frequency.	• A value with one decimal place • Range: 0.0 to 400.0
	Termination frequency	Sets the stop frequency for tuning.	• Integer • Range: 0 to 8000
	Low-frequency starting frequency	Defines the start frequency of the low-frequency range for auto-tuning.	• Integer • Range: 0 to 8000 • Must be below the high frequency start frequency.

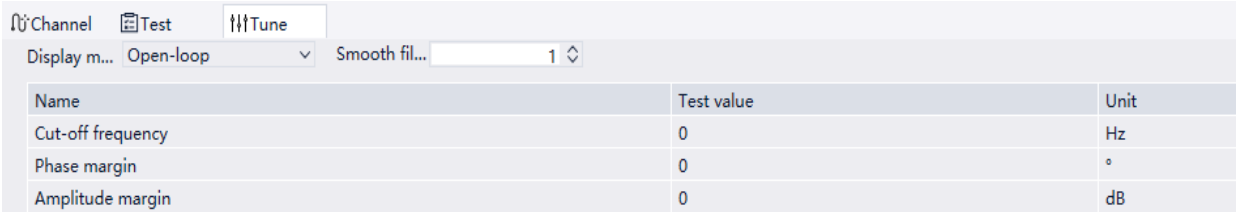
Function		Description	Restriction
	High-frequency starting frequency	Defines the start frequency of the high-frequency range for auto-tuning.	• Integer • Range: 0 to 8000 • Must be above the low frequency start frequency.
	Low band points	Defines the number of sweep frequency points of the low-frequency range for auto-tuning.	• Integer • Range: 0 to 500
	High band points	Defines the number of sweep frequency points of the high-frequency range for auto-tuning.	• Integer • Range: 0 to 1000
Parameter configuration	Position loop gain	Sets the position loop gain.	• A value with one decimal place • Range: 0.1 to 2000.0 Hz
	Speed loop gain	Sets the speed loop gain.	• A value with one decimal place • Range: 0.1 to 2000.0 Hz
	Speed loop integral time	Sets the speed loop integral time.	• A value with two decimal places • Range: 0.15 ms to 512.00 ms
	Torque reference filter time	Defines the torque reference filter time.	• A value with two decimal places • Range: 0.01 ms to 30.00 ms
	Inertia ratio	Sets the inertia ratio.	• A value with two decimal places • Range: 0.00 to 120.00
Notch	Notch parameters	Sets the parameters of the four notch groups. When different filter options are selected for the 1st notch, the configurable parameters of the 1st notch will also be changed. For details, see the following table.	• Notch frequency range: 50 to 8000 Hz. • Notch width class range: 0 to 20. • Notch depth class range: 0 to 99. All three parameters are integers.

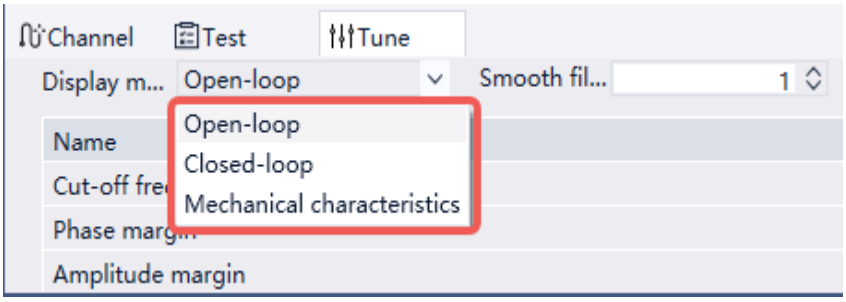
Filter	Configurable Parameter	Icon
Disabled	Notch frequency Notch width class Notch depth class	
First-order low-pass	Cut-off frequency	
Second-order low-pass	Cut-off frequency Damping coefficient: A value with three decimal places; range: 0.000 to 10.000.	
Lead-lag	Frequency: An integer; range: 0 to 16000. Gain: A value with one decimal place; range: -336.0 to 336.0.	
Custom	Numerator of cut-off freq.: An integer; range: 0 to 16000. Numerator of damping coef.: A value with three decimal places; range: 0.000 to 10.000. Denom. of cut-off freq.: An integer; range: 0 to 16000. Denom. of damping coef.: A value with three decimal places; range: 0.000 to 10.000.	

- **Channel/Test tab**

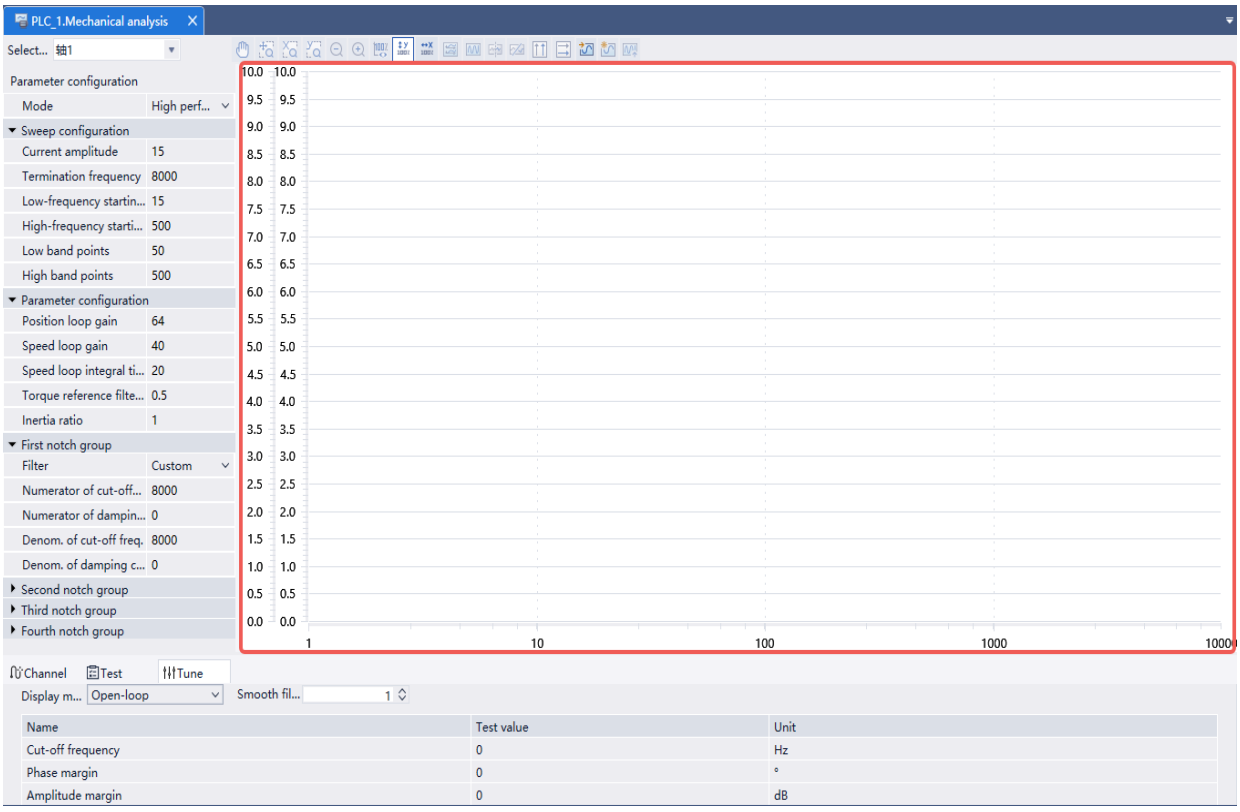
  The channel/test tab in the high-performance auto-tuning mode has the same functions as that in the mechanical characteristic analysis mode. For details, see "Mechanical Analysis".

- **Tune tab**



Function	Description
Display mode	The display modes include open-loop, closed-loop, and mechanical characteristics. When you select different display modes, the curve and the parameter list will change accordingly.
Smooth filtering	Defines the smoothness of the curve.
Filter table	Displays the corresponding parameter values after high performance auto-tuning is done.

- **Bode diagram**



This section only explains the two icons for generating simulation curve and exporting simulation curve. For details on other icons, see "Mechanical Analysis".

Icon	Function	Description	Restriction
	Generate simulation curve	After modifying the configuration parameters for high-performance tuning, click the "Generate simulation curve" icon to generate two simulated curves.	• This is only available after the high-performance auto-tuning test is done. • This is unavailable in the mechanical characteristic analysis mode.
	Export simulation curve	Exports the simulation curve data to a CSV file and saves it to the local drive.	It cannot be used when there is no simulation curve.

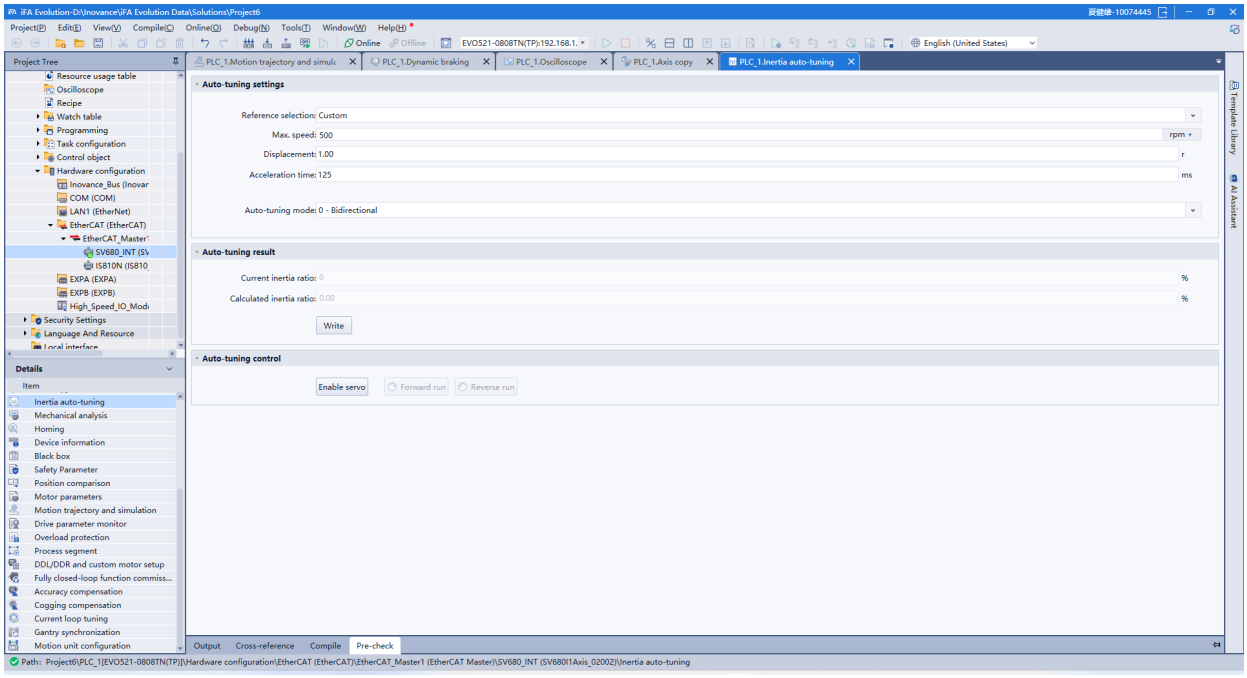
17.3.9 Inertia Auto-Tuning

This function enables automatic parameter settings for the servo drives and is an important part of servo gain auto-tuning. Correct load inertia ratio ensures quick commissioning. The load inertia ratio is an important parameter of the servo system, and its meaning is shown in the following figure.

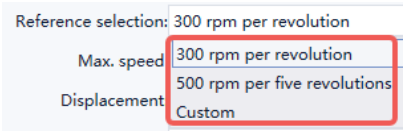
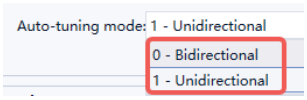
$$\text{Load inertia ratio} = \frac{\text{Total rotational inertia of mechanical load}}{\text{Motor rotor inertia}}$$

- **Function interface**

  In the project tree, select a servo device and double-click "Inertia auto-tuning" in the detailed view to open the interface. The interface mainly consists of three parts: "Auto-tuning settings", "Auto-tuning result", and "Auto-tuning control", as shown in the figure below.



- **Parameter setting**

Function	Description
Axis	Selects the axis number of the current device. Restrictions: Unable to configure when the drive is enabled.
Reference selection	Selects a motor reference. Restrictions: Unable to configure when the drive is enabled. 
Maximum velocity	Defines the maximum motor speed. Restrictions: • Editable only when the "Reference selection" is set to "Custom". • Integer; range: 100 to 1000; unit: rpm/min • It will be sent to the drive upon enabling.
Displacement	Defines the motor displacement. Restrictions: • Editable only when the "Reference selection" is set to "Custom". • A value with two decimal places; range: 0.00 to 100.00; unit: r (revolution). • It will be sent to the drive upon enabling.
Acceleration time	Defines the time for accelerating to the maximum speed. Restrictions: • It will be sent to the drive upon enabling. • Integer; range: 20 to 800; unit: ms • It cannot be edited after enabling.
Auto-tuning mode	Selects the inertia auto-tuning mode. Restrictions: • Unable to configure when the drive is enabled. • It will be sent to the drive upon enabling. 

- **Auto-tuning results**

Function	Description
Current inertia ratio	Displays the inertia ratio before presumption, which cannot be edited.
Calculated inertia ratio	Displays the presumed inertia ratio, which cannot be edited.
Write	Downloads the presumed inertia ratio to the drive. Download cannot be performed after enabling.

- **Auto-tuning control**

Function	Description
Enable servo	Enables/disables the drive.
Forward run	Controls the forward rotation of the motor. You can long press the button or click it once. Restrictions: It is available when the drive is enabled, and disabled after the drive is disabled.
Reverse Rotation	Controls the reverse rotation of the motor. You can long press the button or click it once. Restrictions: It is available when the drive is enabled, and disabled after the drive is disabled.

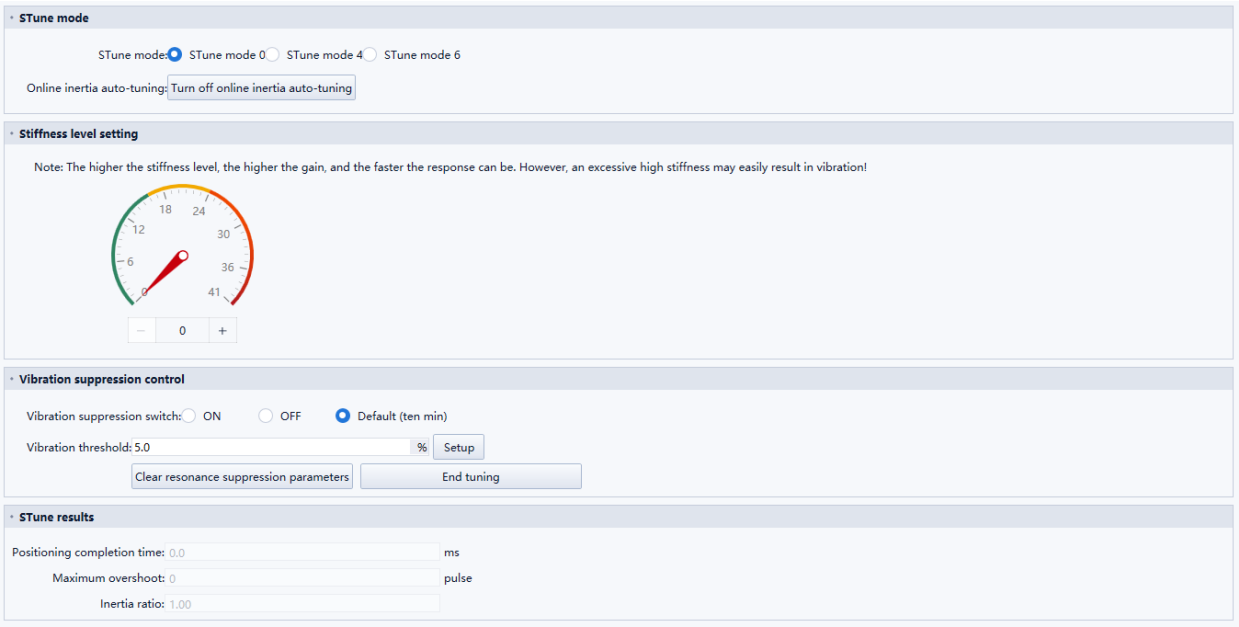
17.3.10 STune Adjustment

STune allows the servo drive to perform automatic parameter auto-tuning based on rigidity level settings for fast and stable gain adjustment.

It is applicable to scenarios with small load inertia change and does not support torque control.

- **Function interface**

  Select a servo drive in the project tree, then double-click "STune" in the detailed view to open the interface, as shown in the figure below.



- **ETune mode**
  - STune mode

> **NOTE**
>
> - In STune mode 0, you need to set the inertia ratio before adjusting the stiffness. For unknown inertia, perform inertia tuning manually. If vibration occurs on the machine, decrease the stiffness level before manual inertia tuning. In modes 4 and 6, the inertia ratio can be adjusted through the wizard-type interface, without the need for setting an inertia ratio.
> - Adjust the stiffness level gradually during load operation. The present stiffness level will be written to the drive automatically. Keep monitoring the operating waveform after increasing the stiffness level (increase by one level at a time) until desired performance is achieved.
> - The mode takes effect automatically after you select it.

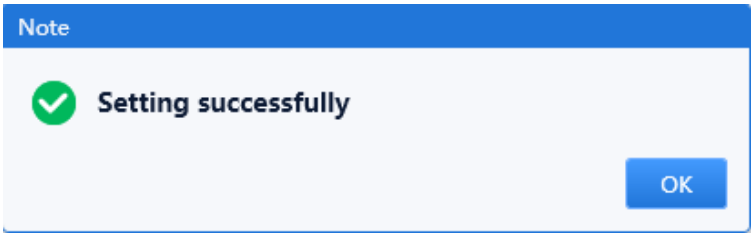
Mode	Name	Description
STune mode 0	Manual mode + Inertia auto-tuning	The gains need to be adjusted manually.
STune mode 4	Normal mode + Inertia auto-tuning	Automatic gain setting, inertia auto-tuning and vibration suppression based on the set stiffness.
STune mode 6	Quick positioning mode + Inertia auto-tuning	Automatic gain setting, inertia auto-tuning and vibration suppression based on the set stiffness. This mode is applicable to occasions requiring quick positioning.

  - Online inertia auto-tuning: Used to disable online inertia auto-tuning.

- **Stiffness level setting**

Description	Description
Diagram	Graphically displays the set stiffness level.
+	Increments the stiffness level. Restrictions: Click once to increase the stiffness level by 1, and long press is ineffective.
-	Decrements the stiffness level Restrictions: Click once to decrease the stiffness level by 1, and long press is ineffective.
Stiffness level	Displays the set stiffness level. Restrictions: Non-editable; range: 0 to 41.

- **Vibration suppression control**

Description	Description
Vibration suppression switch	Enables or disables the vibration suppression function. Options: ON, OFF, and default (10 minutes)
Vibration threshold	• Defines the percentage of the vibration threshold. Range: 0% to 100%. Default is 5.0%. • Write the input vibration threshold into the servo drive via the "Setup" button. If the setting is successful, the following dialog box appears: 
Clear resonance suppression parameters	Clears the resonance suppression parameters.
End tuning	Switches to STune mode 0.

- **STune results**

Description	Description
Positioning completion time	Displays the positioning completion time in real time. It cannot be edited.
Maximum overshoot	Displays the maximum overshoot in real time. It cannot be edited.

Description	Description
Inertia ratio	Displays the inertia ratio in real time. It cannot be edited.

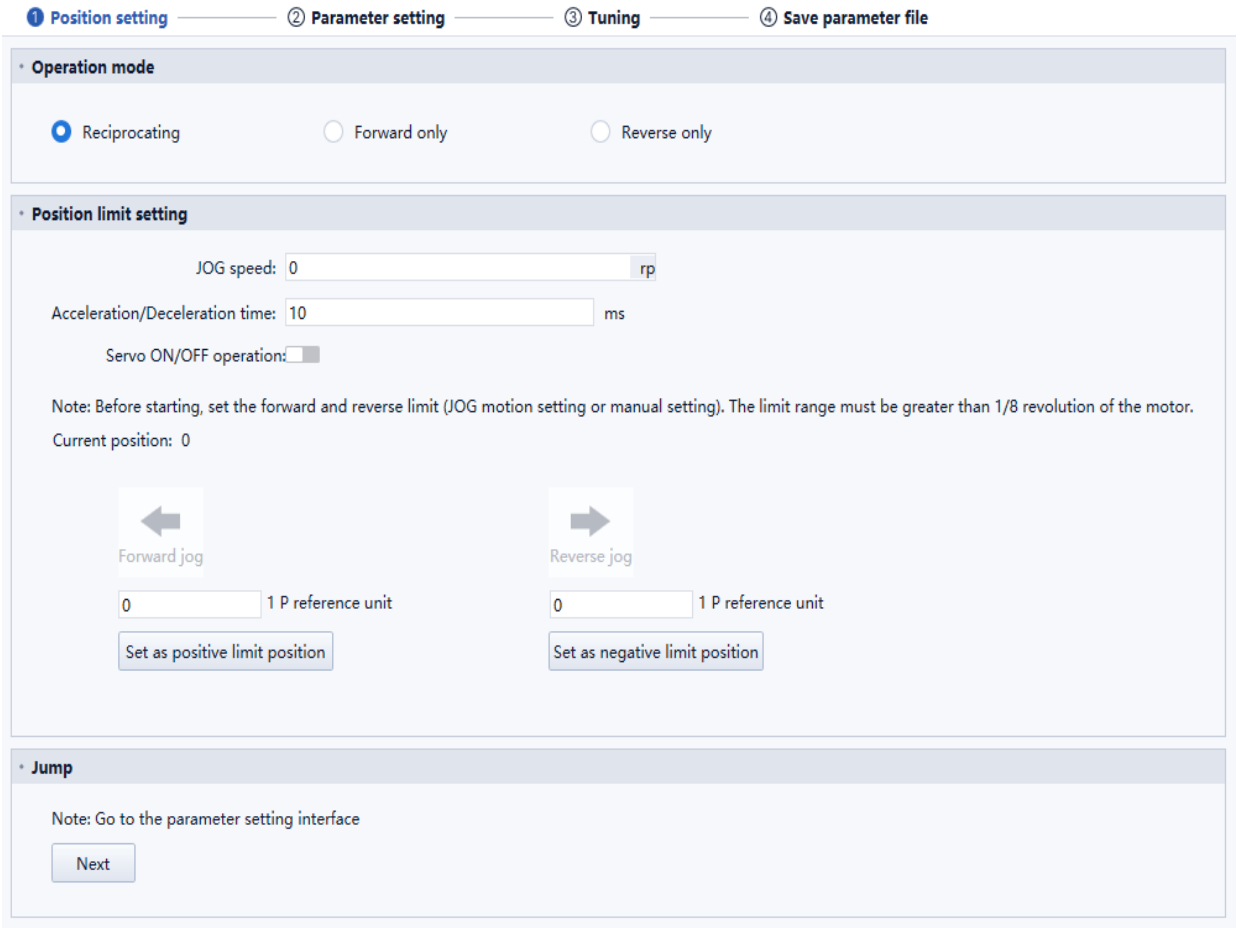
17.3.11 ETune Guided Auto-Tuning

17.3.11.1 Overview

ETune is a wizard-type auto-tuning function used to guide users to set corresponding curve trajectories and response parameters. After settings, the servo drive performs auto-tuning to generate the optimal gain parameters. The auto-tuned parameters can be saved and exported as a recipe for use in other devices of the same model. The ETune function is intended to be used in applications featuring small load inertia changes.

17.3.11.2 Position Settings

Select a servo device in the project tree, and double-click "ETune" in the detailed view to open the interface, as shown in the figure below.



Description of marks at the top of the interface

Mark	Description
	Set for the first time.
	Not set.
	Setting failed.
	Setting succeeded.
	Setting is ongoing, and the previous setting failed.
	Setting is ongoing, and the previous setting succeeded.

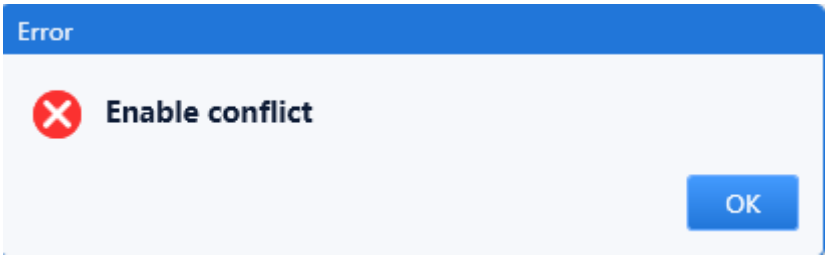
Function

Function	Sub-Function	Description
Run mode	Reciprocating	The motor reciprocates between positive and negative limit positions.
	Forward only	The motor takes the difference between the positive and negative position limits as the maximum distance per action and keeps running in the forward direction.
	Reverse only	The motor takes the difference between the positive and negative position limits as the maximum distance per action and keeps running in the reverse direction.
Position limit setting	JOG speed	Defines the jog speed. Range: 0 to 1000; unit: rpm
	Acceleration/ Deceleration time	Defines the acceleration/deceleration time during jog. Range: 0 to 65535; unit: ms
	Servo ON/OFF operation	Enables or disables the servo drive. It is displayed according to the enabling status of the servo drive.

Function	Sub-Function	Description
	Current position	Displays the actual feedback position of the motor.
	Forward jog	Controls the forward jog of the servo drive. Click or long press to start.
	Positive limit position (1 P reference unit)	Displays the positive limit position, which can be directly edited, or set by clicking the "Set as positive limit position" button. Restrictions: • Range: -2147483647 to 2147483647 • The positive limit position cannot be smaller than the negative limit position. • The positive limit position cannot be smaller than the current position. • The difference between the positive and negative limits cannot be smaller than 1/8 revolution of the motor (that is, 1/8 of the motor limit threshold).
	Set as positive limit position	Sets the current position to the positive limit position, which is the value of the previous electronic gear ratio. Click this button to set the current position to the input box of the positive limit position.
	Reverse jog	Controls the reverse jog of the servo drive. Click or long press to start.
	Negative limit position (1 P reference unit)	Displays the negative limit position, which can be directly edited, or set by clicking the "Set as negative limit position" button. Restrictions: • Range: -2147483647 to +2147483647 • The negative limit position cannot be greater than the positive limit position. • The negative limit position cannot be greater than the current position. • The difference between the positive and negative limits cannot be smaller than 1/8 revolution of the motor (that is, 1/8 of the motor limit threshold).
	Set as negative limit position	Sets the current position to the negative limit position, which is the value of the previous electronic gear ratio. Click this button to set the current position to the input box of the negative limit position.
Jump	Next	Jumps to the "Parameter setting" interface and writes all the currently configured parameters to the drive.

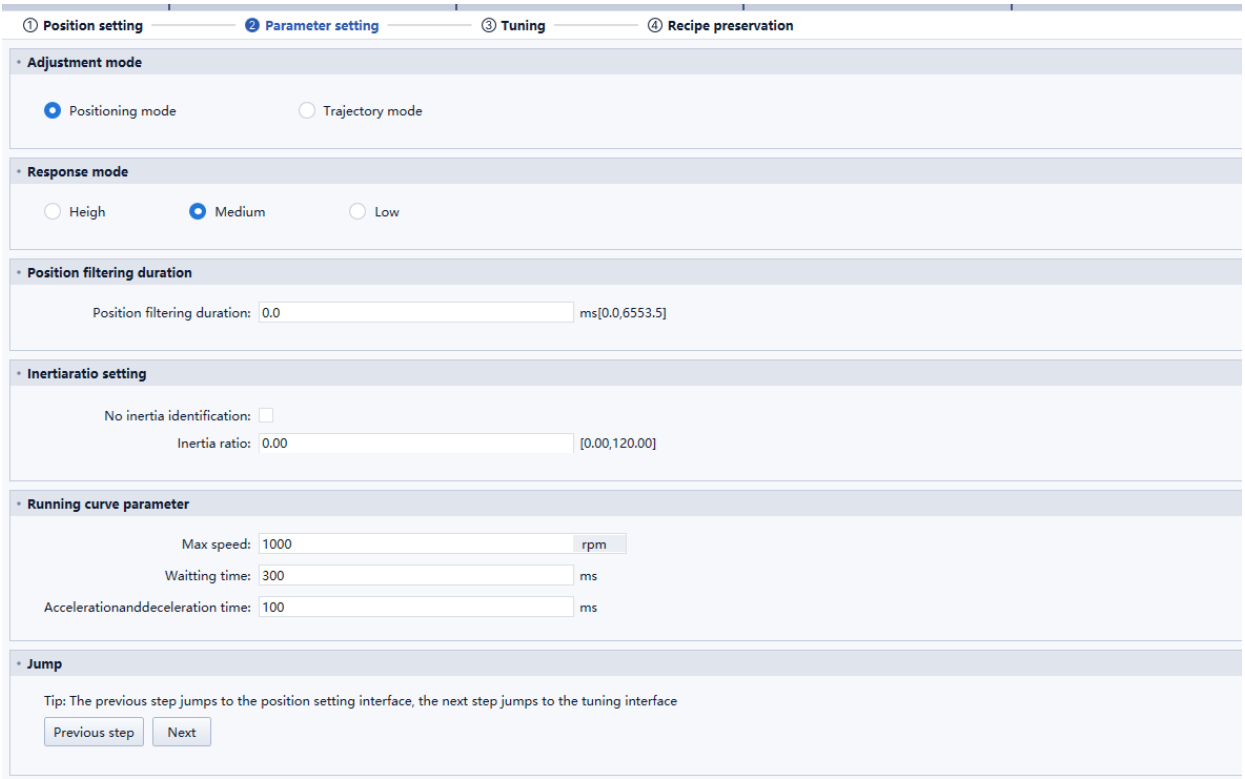
CAUTION

The enable operation in ETune and the servo enable operation in JOG control cannot be executed simultaneously; otherwise, an enable conflict will be prompted.



17.3.11.3 Parameter Settings

The parameter setting interface is shown in the following figure. See the following table for descriptions of functions and parameters.



Description		Description
ETune mode		Selects "Positioning mode" or "Trajectory mode".
Response mode		Selects the response mode from "High", "Medium", and "Low".
Position filtering time constant		Defines the position filter time constant. A value with one decimal place; range: 0.0 to 6553.5; unit: ms
Inertia ratio setting	No inertia auto-tuning	Sets whether inertia ratio is auto-tuned. Inertia auto-tuning is performed by default. Restrictions: When it is enabled, the "Inertia ratio" input box cannot be edited.

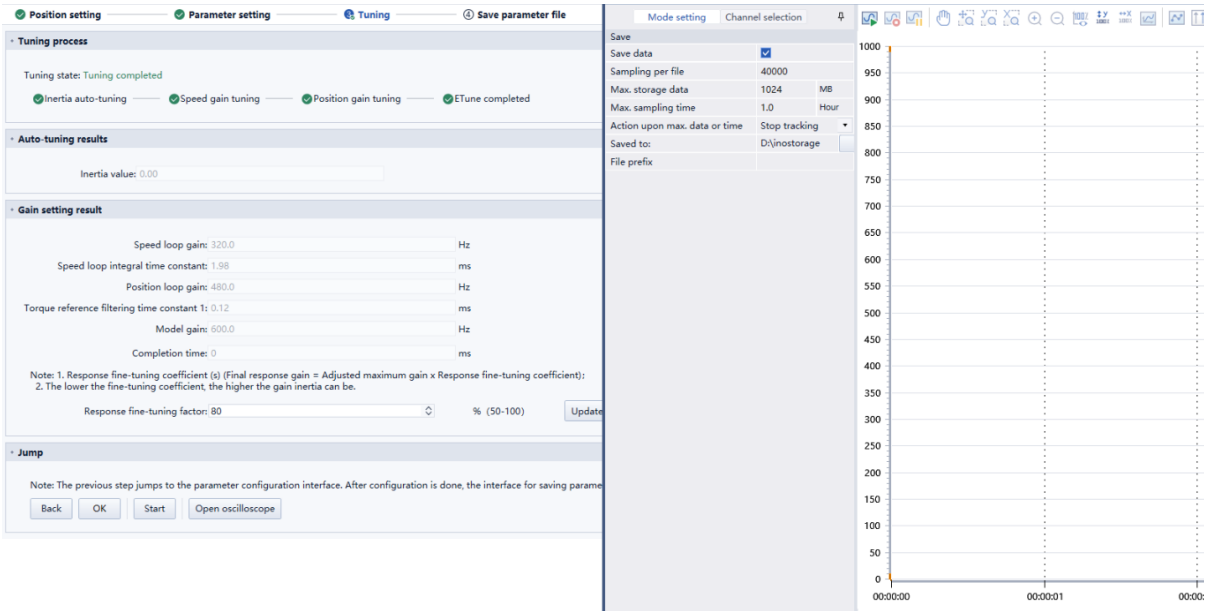
Description		Description
	Inertia ratio	Defines the inertia ratio. A value with two decimal places; range: 0.00 to 120.00
Motion curve parameter	Maximum speed	Defines the maximum auto-tuned speed. Integer; range: 0 to 1000; unit: rpm Restrictions: The maximum motor speed cannot be exceeded.
	Waiting time	Defines the waiting time between two operations. Integer; range: 0 to 65535; unit: ms
	Acceleration/ Deceleration time	Defines the maximum auto-tuned acceleration/deceleration time. Integer; range: 0 to 65535; unit: ms
Jump	Back	Jumps to the "Position setting" tab.
	Next	Jumps to the "Tuning" tab and writes the parameters set in this tab to the drive.

17.3.11.4 Tuning

If inertia auto-tuning is enabled (by default) in the "Parameter setting" tab, the following is displayed under the "Tuning state": ① Inertia auto-tuning, ② Speed gain tuning, ③ Position gain tuning, and ④ ETune completed. After auto-tuning is started, inertia auto-tuning is performed based on the set curve. After auto-tuning is done, the drive automatically enters the gain tuning stage.

If inertia auto-tuning is disabled in the "Parameter setting" tab, the following is displayed under the "Tuning state": ① Speed gain tuning, ② Position gain tuning, and ③ ETune completed. Gain is auto-tuned directly after auto-tuning is started.

The "Tuning" tab is shown in the following figure. See the following table for descriptions of functions and parameters.



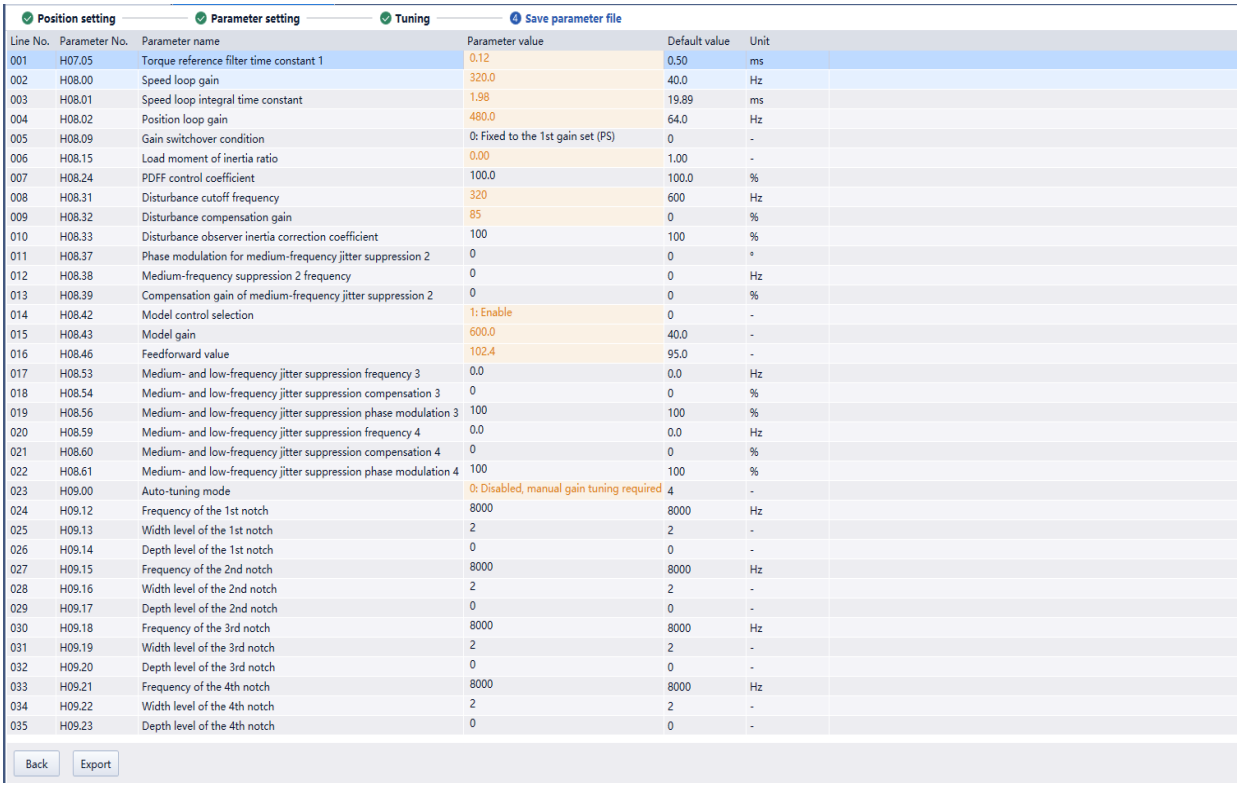
During gain auto-tuning, if you modify "Response fine-tuning coefficient" and click "Update", gain auto-tuning will be continued based on the set fine-tuning coefficient. After gain auto-tuning is done, click "OK" to save the parameters to EEPROM and export the parameters as a recipe file.

Function		Description
Tuning process	Tuning state	The number before each status is marked by different colors. • Blue indicates an ongoing process is going on. • No color indicates a process that has not been performed. • Green indicates a completed process with correct content • Red indicates a completed process with incorrect content If inertia auto-tuning is enabled (by default) in the "Parameter setting" tab, the following is displayed under the "Tuning state": ① Inertia auto-tuning, ② Speed gain tuning, ③ Position gain tuning, and ④ ETune completed. After auto-tuning is started, inertia auto-tuning is performed based on the set curve. After auto-tuning is done, the drive automatically enters the gain tuning stage. If inertia auto-tuning is disabled in the "Parameter setting" tab, the following is displayed under the "Tuning state": ① Speed gain tuning, ② Position gain tuning, and ③ ETune completed. Gain is auto-tuned directly after auto-tuning is started.
Auto-tuning results	Inertia value	Displays the inertia value after inertia auto-tuning. It cannot be edited.
Gain setting result	Speed loop gain	Displays the speed loop gain value after auto-gain. It cannot be edited.
	Speed loop integral time constant	Displays the speed loop integral time constant after auto-gain. It cannot be edited.
	Position loop gain	Displays the position loop gain value after auto-gain. It cannot be edited.
	Torque reference filtering time constant 1	Displays the torque reference filter time constant 1 after auto-gain. It cannot be edited.
	Model gain	Displays the model gain value after auto-gain. It cannot be edited.
	Completion Time	Displays the completion time after auto-gain. It cannot be edited.
	Response fine-tuning factor	Defines the response fine-tuning factor. Range: 50% to 100%
	Update	Writes the response fine-tuning coefficient to the servo drive.

Function		Description
Jump	Back	Jumps to the "Parameter setting" tab. If "Back" is clicked during auto-tuning, a dialog box will appear asking you whether to go back. If you confirm to go back, auto-tuning will be terminated. If "Back" is clicked without auto-tuning in progress, the system will directly go back to the previous step.
	Completion	After auto-tuning is completed, a dialog box will appear indicating that gain auto-tuning is completed. Clicking "OK" will jump to the "Save parameter file" interface and save the parameters to EEPROM. Restrictions: The "OK" button is unavailable before auto-tuning is done.
	Start/Stop	Starts or stops auto-tuning. Clicking "Stop" will stop auto-tuning, and restore the related recipe parameters to the status before auto-tuning.

17.3.11.5 Recipe Storage

After auto-tuning is done, the related parameters will be displayed in a list and exported as a recipe file for future reuse. The recipe storage interface is shown in the following figure. See the following table for descriptions of functions and parameters.



Line No.	Parameter No.	Parameter name	Parameter value	Default value	Unit
001	H07.05	Torque reference filter time constant 1	0.12	0.50	ms
002	H08.00	Speed loop gain	320.0	40.0	Hz
003	H08.01	Speed loop integral time constant	1.98	19.89	ms
004	H08.02	Position loop gain	480.0	64.0	Hz
005	H08.09	Gain switchover condition	0: Fixed to the 1st gain set (PS)	0	-
006	H08.15	Load moment of inertia ratio	0.00	1.00	-
007	H08.24	PDFF control coefficient	100.0	100.0	%
008	H08.31	Disturbance cutoff frequency	320	600	Hz
009	H08.32	Disturbance compensation gain	85	0	%
010	H08.33	Disturbance observer inertia correction coefficient	100	100	%
011	H08.37	Phase modulation for medium-frequency jitter suppression 2	0	0	°
012	H08.38	Medium-frequency suppression 2 frequency	0	0	Hz
013	H08.39	Compensation gain of medium-frequency jitter suppression 2	0	0	%
014	H08.42	Model control selection	1: Enable	0	-
015	H08.43	Model gain	600.0	40.0	-
016	H08.46	Feedforward value	102.4	95.0	-
017	H08.53	Medium- and low-frequency jitter suppression frequency 3	0.0	0.0	Hz
018	H08.54	Medium- and low-frequency jitter suppression compensation 3	0	0	%
019	H08.56	Medium- and low-frequency jitter suppression phase modulation 3	100	100	%
020	H08.59	Medium- and low-frequency jitter suppression frequency 4	0.0	0.0	Hz
021	H08.60	Medium- and low-frequency jitter suppression compensation 4	0	0	%
022	H08.61	Medium- and low-frequency jitter suppression phase modulation 4	100	100	%
023	H09.00	Auto-tuning mode	0: Disabled, manual gain tuning required	4	-
024	H09.12	Frequency of the 1st notch	8000	8000	Hz
025	H09.13	Width level of the 1st notch	2	2	-
026	H09.14	Depth level of the 1st notch	0	0	-
027	H09.15	Frequency of the 2nd notch	8000	8000	Hz
028	H09.16	Width level of the 2nd notch	2	2	-
029	H09.17	Depth level of the 2nd notch	0	0	-
030	H09.18	Frequency of the 3rd notch	8000	8000	Hz
031	H09.19	Width level of the 3rd notch	2	2	-
032	H09.20	Depth level of the 3rd notch	0	0	-
033	H09.21	Frequency of the 4th notch	8000	8000	Hz
034	H09.22	Width level of the 4th notch	2	2	-
035	H09.23	Depth level of the 4th notch	0	0	-

Function	Description
Parameter list	Displays information about the parameters. • The parameters are displayed according to the actual status of the servo drive. • If the parameter value is inconsistent with the default value, it will be highlighted. • The value of the offline parameter is highlighted in red.
Export	Exports the auto-tuning result to a recipe file in the CSV or DAT format for future use.
Back	Jumps to the "Tuning" tab.

17.3.12 Homing

This feature is currently not supported for certain product series.

The homing function is used to search for the mechanical home and determine the position relation between the mechanical home and mechanical zero.

- Mechanical home: A fixed position on the machine, which corresponds to a certain home switch or the motor Z signal.
- Mechanical zero: Absolute zero point on the machine. After homing is completed, the motor stops at the mechanical home. The relationship between the mechanical home and the mechanical zero can be set via 607Ch (Home offset): Mechanical home= Mechanical zero + Home offset (0x607C). When the offset is 0, the mechanical home is the same as the mechanical zero.

- **Function interface**

  Select a servo drive in the project tree, and double-click "Homing" in the detailed view to open the interface, as shown in the figure below.

- **Homing curve configuration**

Description	Description
Homing method	Sets the homing method for the servo home. The homing curve diagram corresponding to the selected homing method will be displayed on the right, and other parameters and their settings will also change.
Home signal	Sets the home signal. Corresponding settings will be displayed based on the selected homing method, and the corresponding homing curve diagram will be shown according to the different settings.

Description	Description
Starting direction	Sets the direction for starting the homing process. Corresponding settings will be displayed based on the selected homing method, and the corresponding homing curve diagram will be shown according to the different settings.
Motor action upon positive limit input	Sets the motor action when the positive limit input is triggered. Corresponding settings will be displayed based on the selected homing method, and the corresponding homing curve diagram will be shown according to the different settings.
Motor action upon negative limit input	Sets the motor action when the negative limit input is triggered. Corresponding settings will be displayed based on the selected homing method, and the corresponding homing curve diagram will be shown according to the different settings.
Motor stop position	Sets the position where the motor stops. Corresponding settings will be displayed based on the selected homing method, and the corresponding homing curve diagram will be shown according to the different settings.
Homing curve	Displays curve diagrams for various homing modes. When parameters in the homing curve configuration are changed, the homing curve interface will synchronously display the corresponding curve diagram.

- **Homing parameter configuration**

Description	Description
Speed in high-speed search for dec. home	Sets the speed in high-speed search for deceleration home.
Speed in low-speed search for dec. home	Sets the speed in low-speed search for deceleration home.
Homing time limit	Sets the time limit for home searching.
Homing acceleration time	Sets the homing acceleration time.

Description	Description
Home offset	Sets the home offset.
Mechanical home offset setting	Selects whether the mechanical home offset is the coordinate after homing or the relative offset after homing. The two modes cannot be selected simultaneously.

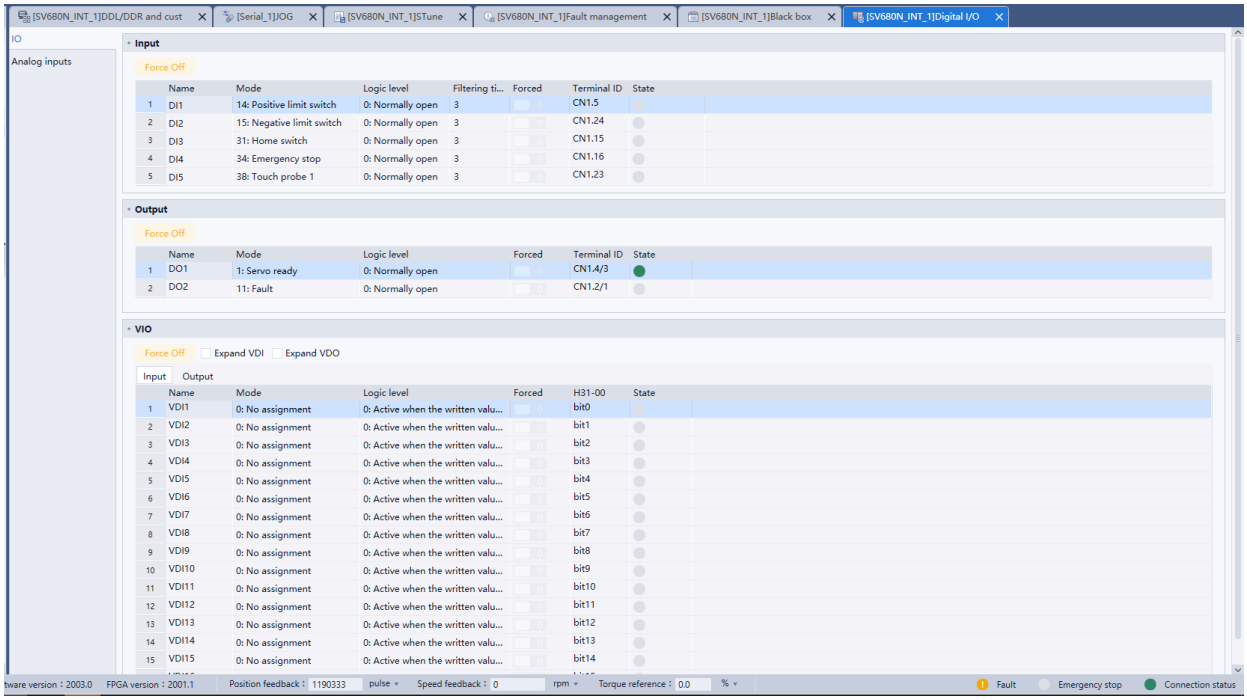
17.3.13 Digital I/O Settings

Digital I/O is used to set and monitor DI and DO functions, as well as virtual I/O functions.

17.3.13.1 I/O Settings and Monitoring

This function monitors and sets the DI/DO function and level polarity of the servo drive, and forcibly switches on or off the DI/DO status.

Select a servo drive in the project tree, and double-click "Digital I/O" in the detailed view to open the interface, as shown in the following figure. See the following table for the description of all functions and parameters.



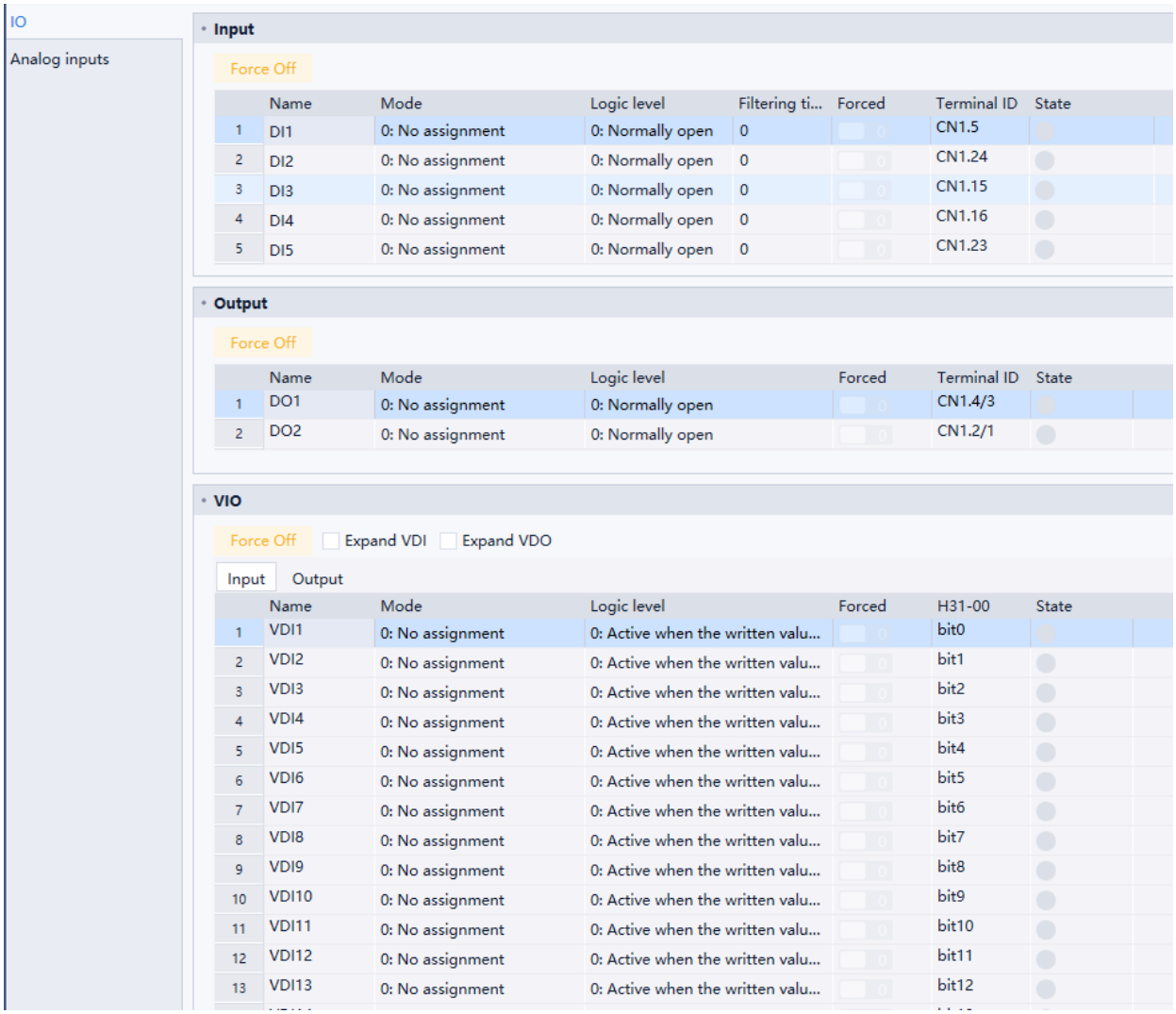
Description	Description
Name	I/O name and actual physical connector pin. It is displayed according to the pins and functions of the servo drive, and cannot be edited.

Description	Description
Mode	Mode setting and monitoring of the I/O terminal. Values can be switched through the drop-down list. The value is updated in real time according to the actual status of the servo drive.
Logic level	Level logic setting and monitoring of the I/O terminal. Values can be switched through the drop-down list. The value is updated in real time according to the actual status of the servo drive.
Filtering time (ms)	Filtering time of the DI terminal. The value is updated in real time according to the actual status of the servo drive and cannot be edited.
Forced	Used to forcibly change the current status of the I/O terminal. It can be set only when "Force on" is displayed. If the I/O interface is closed, the forced enable switch in the current interface will also be turned off.
Terminal ID	Connection sign of the I/O terminal, which cannot be edited.
Status	Used to monitor the actual status of the I/O terminal. The value is updated in real time according to the actual status of the servo drive. It can be manually set only when "Force on" is displayed.
Force off/Force on	Used to enable and disable the forced I/O enable function. The forced input function and the forced output function are mutually exclusive and cannot be enabled simultaneously.

17.3.13.2 Virtual I/O Settings and Monitoring

This function is used to monitor and set the virtual I/O function and logic level of the servo drive, and forcibly turn on or off the real-time VDI status. The forced function of the VDO is currently not supported.

The virtual I/O interface is shown in the following figure. See the following table for descriptions of functions and parameters.

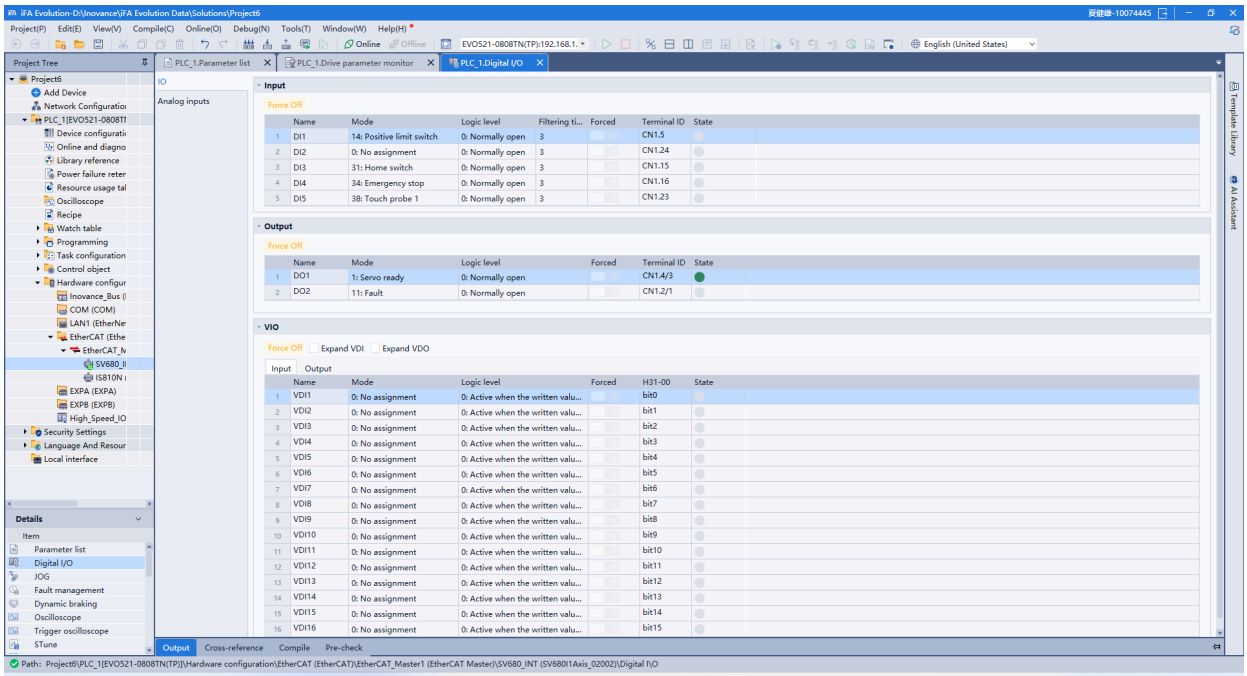


Description	Description
Expand VDI/VDO	Used to enable/disable the input/output communication function of the virtual I/O. Only specific servo drives have virtual I/O functions.
Force off/Force on	Used to enable and disable the forced virtual I/O enable function. Displays the current status of the forced virtual I/O enable function. This button is not displayed in the "Output" tab as the VDO does not support the forced enable function.
Name	Virtual I/O name and actual physical connector pin. It is displayed according to the pins and functions of the servo drive, and cannot be edited.
Mode	Mode setting and monitoring of the virtual I/O terminal. Values can be switched through the drop-down list. The value is updated in real time according to the actual status of the servo drive.

Description	Description
Logic level	Level logic setting and monitoring of the virtual I/O terminal. Values can be switched through the drop-down list. The value is updated in real time according to the actual status of the servo drive.
Forced	Used to forcibly change the current status of the virtual I/O terminal. It can be set only when "Force on" is displayed.
H31-00	Bit 0 to bit 15 of H31.00 represent VDI1 to VDI16, which cannot be edited.
State	Used to monitor the actual status of the virtual I/O terminal. The value is updated in real time according to the actual status of the servo drive. It can be manually set only when "Force on" is displayed.

17.3.13.3 Analog Input Settings

The AI function interface is shown in the following figure. See the following table for descriptions of functions and parameters.



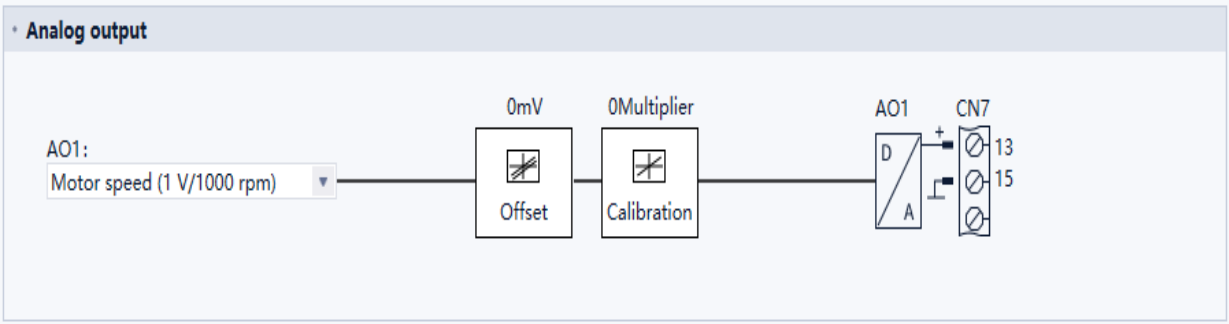
Function	Description
Input mode	Sets the input mode to "Speed mode" or "Torque mode".
Offset	Click "Offset" in the diagram and set the offset value in the pop-up window. Range: -5000 to 5000; unit: mV

Function	Description
Calibration	Click "Calibration" in the diagram and set the calibration value in the pop-up window. • Range in speed mode: 0 to 10000; unit: rpm • Range in torque mode: 0.00 to 8.00; unit: times
Dead zone	Click "Dead zone" in the diagram and set the dead zone value in the pop-up window. Range: 0.0 to 1000.0; unit: mV
Filter	Click "Filter" in the diagram and set the filter value in the pop-up window. Range: 0.00 to 655.35; unit: ms
Zero drift calibration	Click "Zero drift calibration" in the diagram to correct the zero drift.

17.3.13.4 Analog Output Settings

Models supporting analog output function: analog SV670N, SV680N, SV680P-INT, SV680N_INT, SV680N_1G, SV680LN, SV670P-INT, SV670N_INT, SV670N_1Axis_05002_LK.

The AO function interface is shown in the following figure. See the following table for descriptions of functions and parameters.



Function	Description
Signal selection	Sets the mode of the output signal.
Offset	Click "Offset" in the diagram and set the offset value in the pop-up window. Range: -10000 to 10000; unit: mV
Calibration	Click "Calibration" in the diagram and set the calibration value in the pop-up window. Range: -99.99 to 99.99; unit: times

17.3.14 Dynamic Braking Calculation

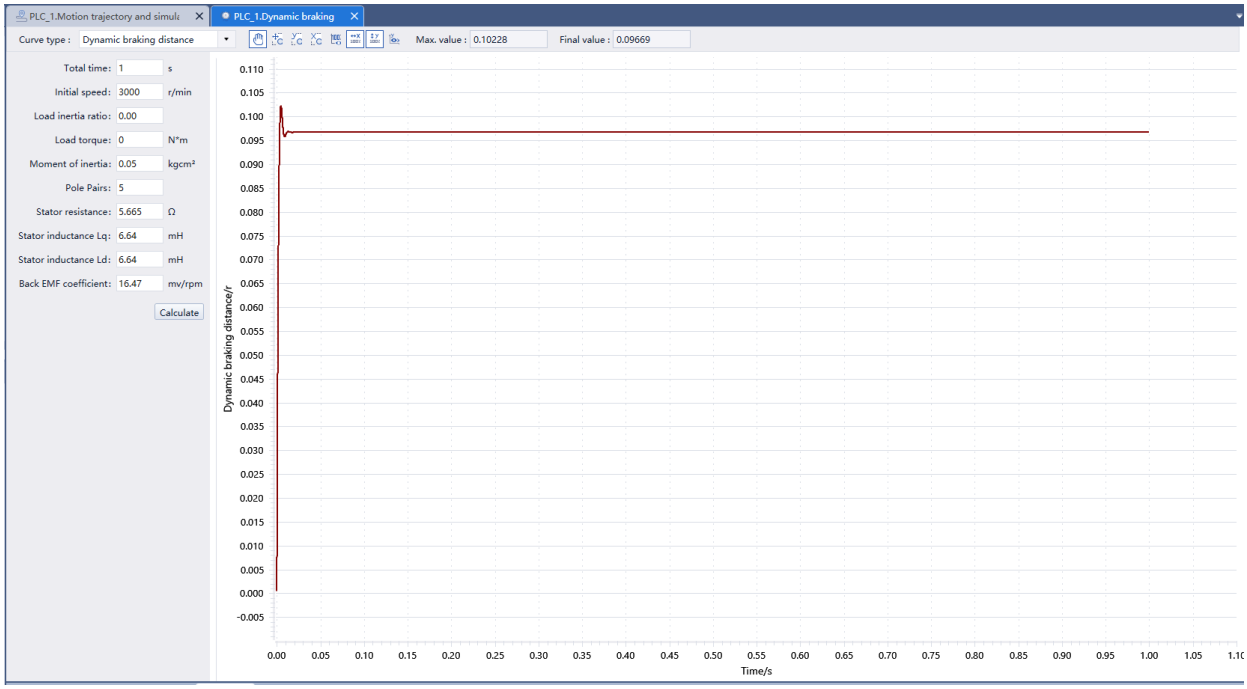
This function is not available for IS810N series products.

This function assesses and analyzes the performance of the dynamic braking system through calculation and simulation. It can be used to determine the correctness of dynamic braking parameters and the system design, and optimize the control policy. Accurate dynamic braking distance can be automatically measured by setting related parameters.

- **Parameter configuration**

  Select a servo device in the project tree, and double-click "Dynamic braking" in the detailed view to open the interface. After configuring the related parameters, click the "Calculate" button to generate the corresponding curve, as shown in the figure below. Refer to the table below for descriptions of each function and parameter.

  When the servo is online, the dynamic braking measurement function reads the corresponding parameters from the servo. If the servo is offline, the function uses automatically filled default parameters.



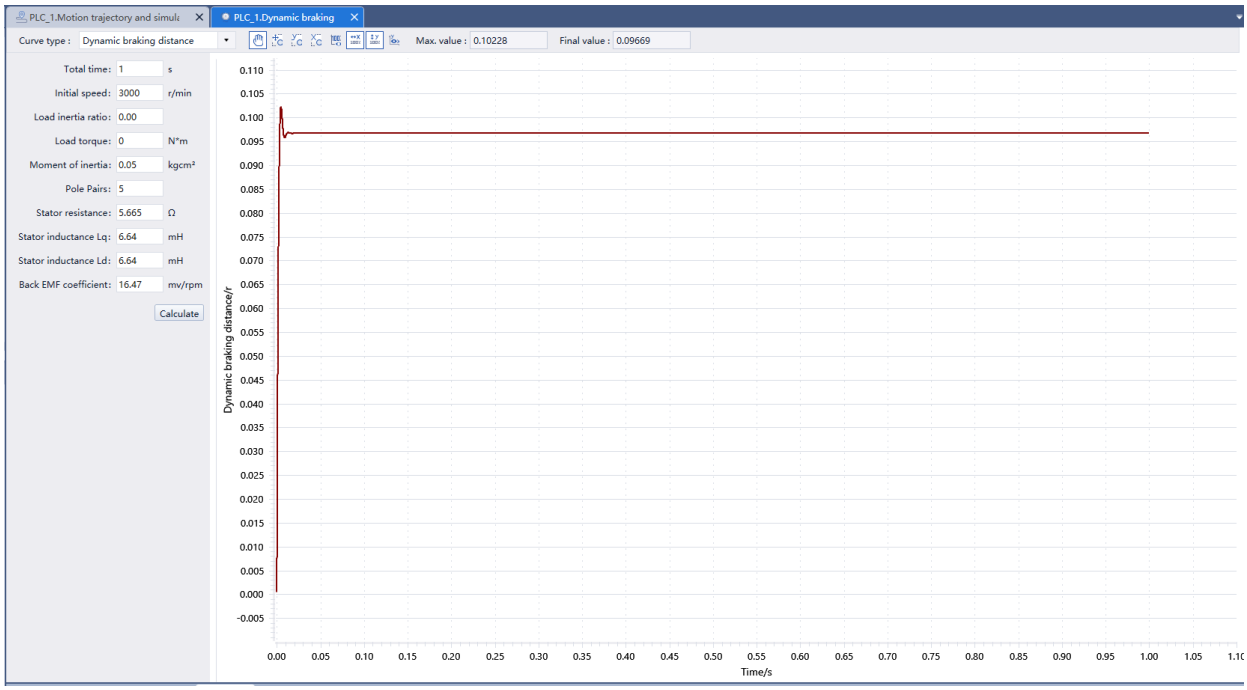
Function	Description
Total time	Defines the total time. The default value when the servo drive is not online is 1. Range: 0 to 100; unit: s
Initial speed	Defines the initial speed. The default value when the servo drive is not online is 3000. Range: -1000000.00 to 1000000.00; unit: r/min

Function	Description
Load inertia ratio	Defines the load moment of inertia ratio. If the servo drive is online, the parameter in the servo drive is read. The default value when the servo drive is not online is 20. Range: 0.00 to 120.00.
Load torque	Defines the load torque. The default value when the servo drive is not online is 0. Range: -1000000.00 to 1000000.00; unit: N*m
Moment of inertia	Defines the motor moment of inertia. If the servo drive is online, the parameter in the servo drive is read. The default value is 0.01 when the servo drive is not online. Range: 0.01 to 655.35, unit: $kgcm^2$.
Pole pairs	Defines the number of PMSM pole pairs. If the servo drive is online, the parameter in the servo drive is read. The default value when the servo drive is not online is 2. Range: 2 to 65535.
Stator resistance	Defines the stator resistance. If the servo drive is online, the parameter in the servo drive is read. The default value when the servo drive is not online is 0.001. Range: 0.001 to 65.535, unit: Ω.
Stator inductance Lq	Defines the stator inductance Lq. If the servo drive is online, the parameter in the servo drive is read. The default value is 0.01 when the servo drive is not online. Range: 0.01 to 655.35; unit: mH
Stator inductance Ld	Defines the stator inductance Ld. If the servo drive is online, the parameter in the servo drive is read. The default value is 0.01 when the servo drive is not online. Range: 0.01 to 655.35; unit: mH
Back EMF coefficient	Defines the linear back EMF coefficient. If the servo drive is online, the parameter in the servo drive is read. The default value is 0.01 when the servo drive is not online. Range: 0.01 to 655.35; unit: mV/rpm

Function	Description
Calculate (button)	Click the "Calculate" button to perform dynamic braking measurement based on the set parameters, then generate the corresponding curve on the right side.

- **Curve operation**

  After the curve is generated, you can perform operations on the curve through the icons above the curve. See the following table for descriptions of functions and parameters.



Function		Description
	Curve type	Selects a curve type that displays certain data, including dynamic braking distance, motor dynamic braking speed and three-phase current amplitude.
	Pan	When it is enabled, you can move and control the view conveniently with the mouse. Restrictions: • It is mutually exclusive with the zoom function. • It is disabled when there is no curve.
	Zoom	When it is enabled, you can select a waveform segment on the X and Y axes for observation. Restrictions: • It is mutually exclusive to the pan function. • It is disabled when there is no curve.

Function		Description
	Vertical zoom	Zooms only on the Y-axis. Restrictions: • It is mutually exclusive to the pan function. • It is disabled when there is no curve.
	Horizontal zoom	Zooms only on the X-axis. Restrictions: • It is mutually exclusive to the pan function. • It is disabled when there is no curve.
	Display whole	Displays 100% of the curve on the X and Y axes. Restrictions: It is disabled when there is no curve.
	X-axis auto scale	Displays 100% of the curve on the X axis. Restrictions: It is disabled when there is no curve.
	Y-axis auto scale	Displays 100% of the curve on the Y axis. Restrictions: It is disabled when there is no curve.
	Display value	Displays the coordinates in the curve. Restrictions: It is disabled when there is no curve.
	Max. value	Displays the maximum value of curve data. Accurate to 5 decimal places.
	Final value	Displays the final value of curve data. Accurate to 5 decimal places.

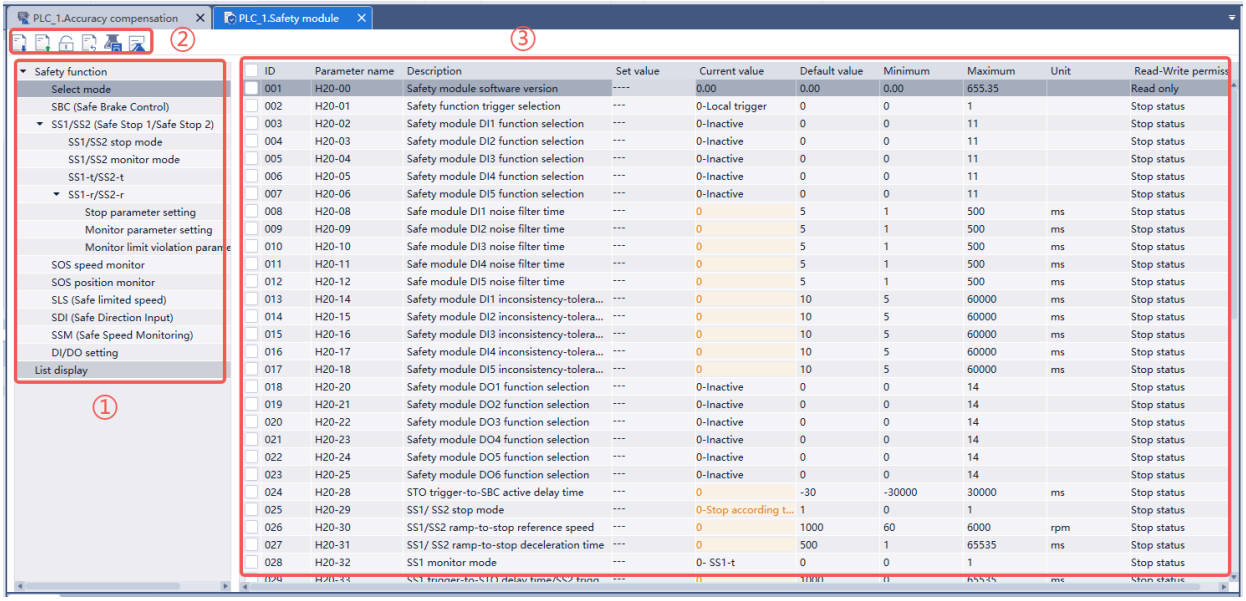
17.3.15 Safety Parameters



This function is currently supported only for the SV680 series function safety models.

This function is designed for production scenarios that require high safety levels and richer function configurations. It enables users to configure safety functions such as STO, SS1, or SDI through the safety module function. The safety module function allows reading and writing settings for servo safety module parameters through the parameter list, or configuration via graphical interfaces for each function.

Click and select the servo drive in the project tree, then double-click "Safety module" in the detailed view to open the interface, as shown in the figure below.



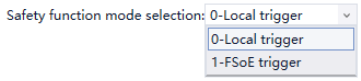
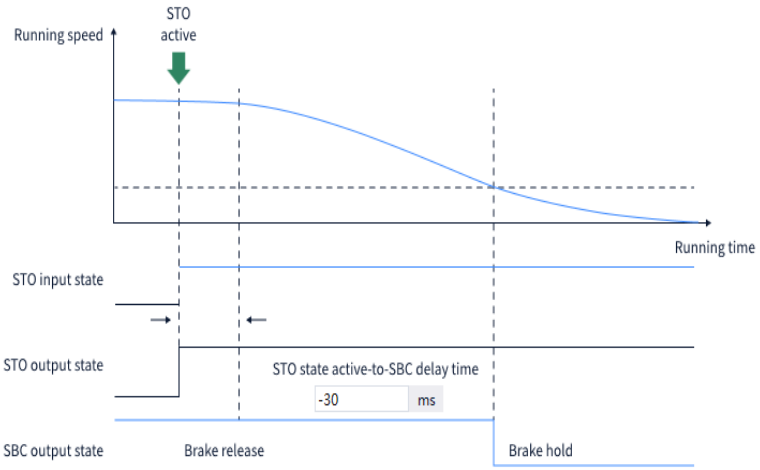
The function interface is divided into three parts: function selection area, toolbar, and parameter editing area.

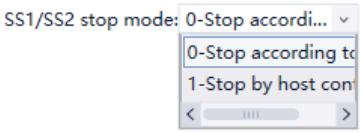
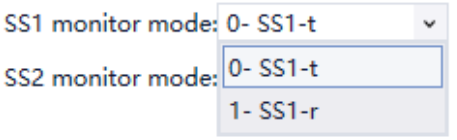
No.	Description	Description
①	Function selection	Selects the corresponding function configuration interface for the safety parameters.
②	Toolbar	Performs relevant operations on the safety parameter list.
③	Parameter editing area	Configures the various functions of the safety parameters, which can be used in conjunction with the toolbar.

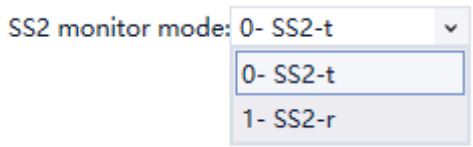
- **Function selection**

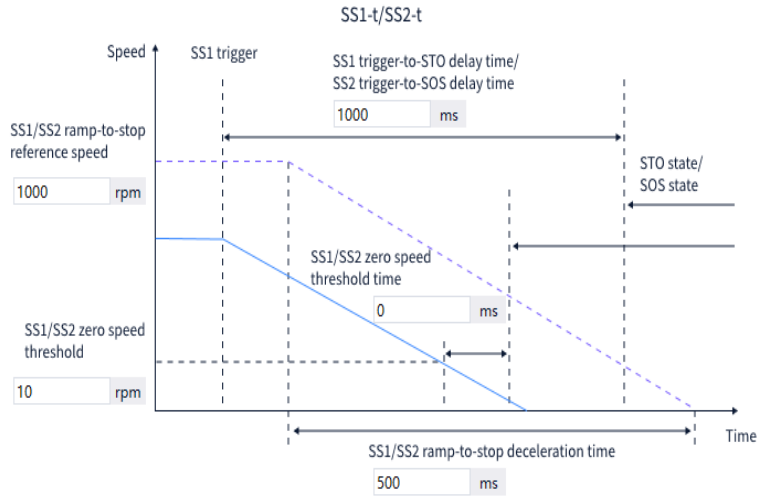
The function selection area displays all safety parameters in a tree structure. Clicking a function node will display its corresponding editing interface. Values set in the function editing interface are synchronized to the parameter editing area. The interface is shown in the figure below. For detailed descriptions of each safety function, see the table below.

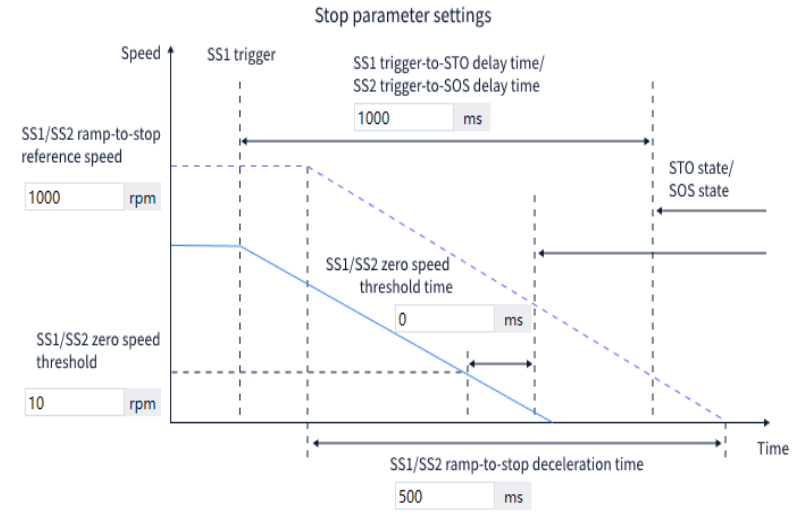
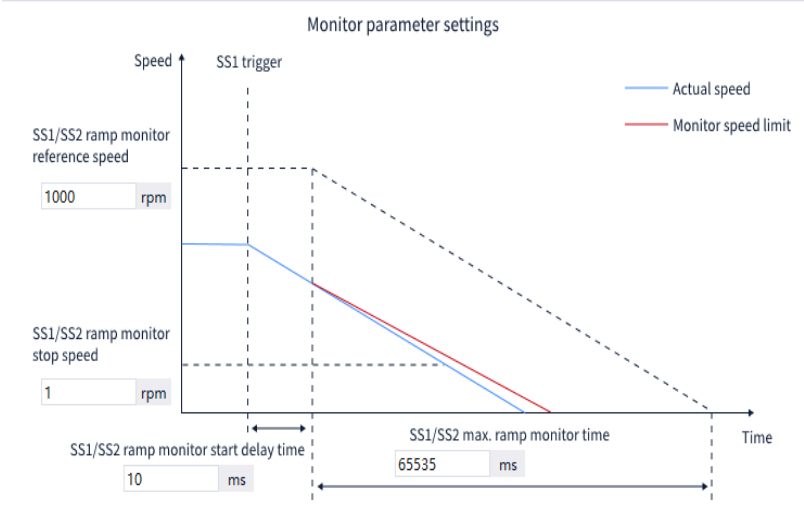


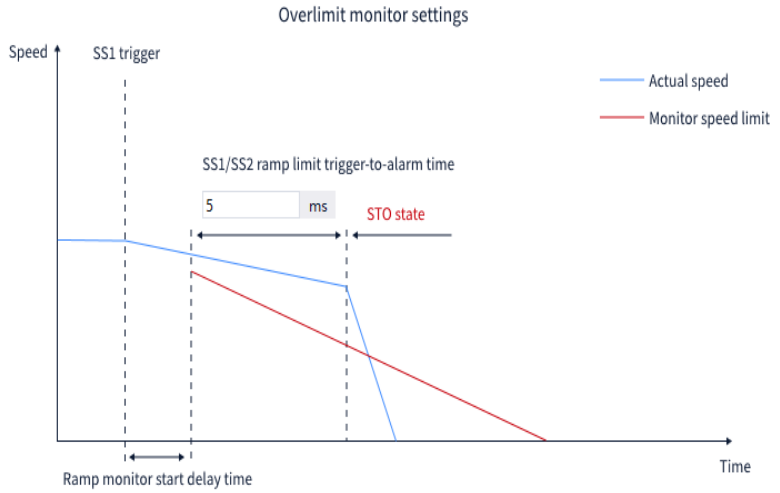
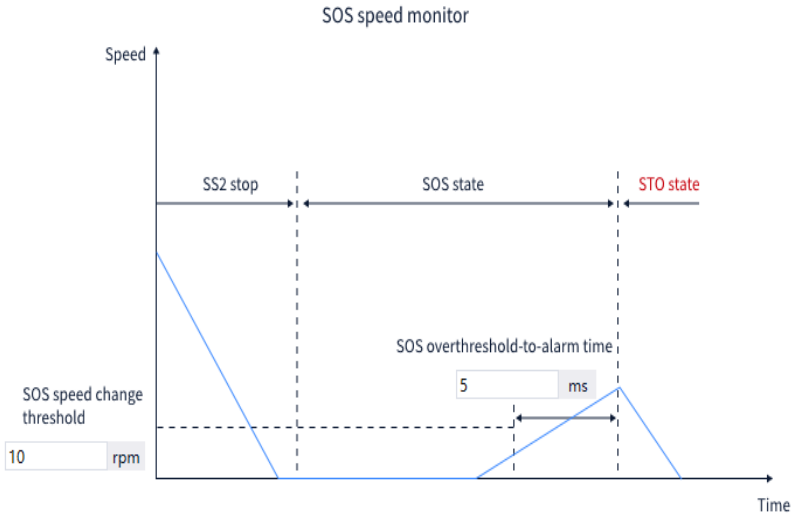
<table>
<tr><th colspan="2">Function</th><th>Configuration</th></tr>
<tr><td colspan="2">Mode selection</td><td>The selectable modes are shown in the figure below:

</td></tr>
<tr><td colspan="2">SBC (Safe brake control)</td><td>

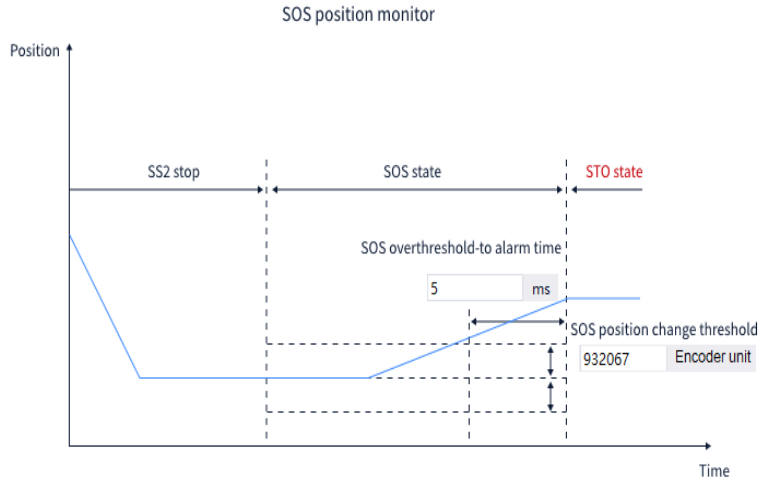
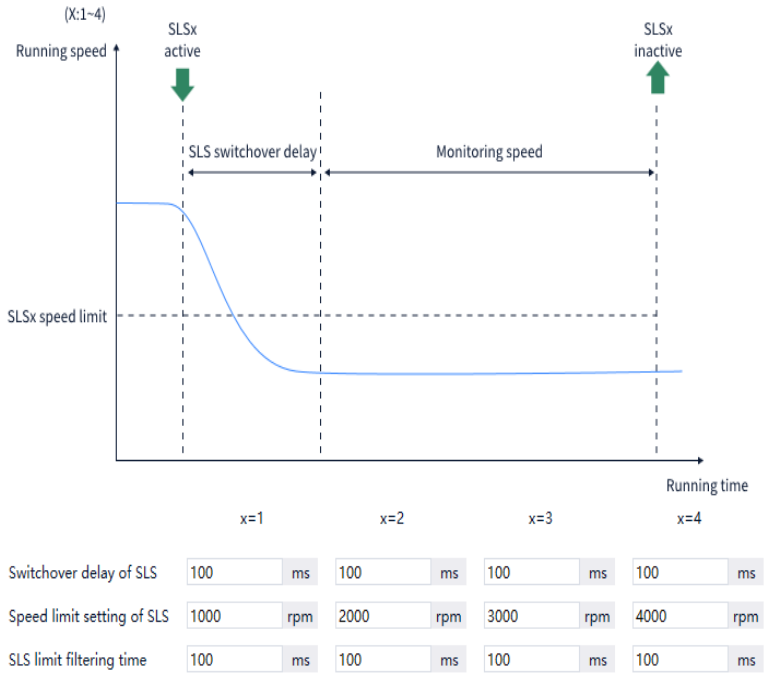
Configurable parameter:
STO state active-to-SBC delay time: Ranges from -30000 to 30000 ms.</td></tr>
<tr><td rowspan="2">SS1/SS2 (Safe stop 1/Safe stop 2)</td><td>SS1/SS2 stop mode</td><td>The selectable modes are shown in the figure below:

</td></tr>
<tr><td>SS1/SS2 monitor mode</td><td>The selectable modes are shown in the figure below:


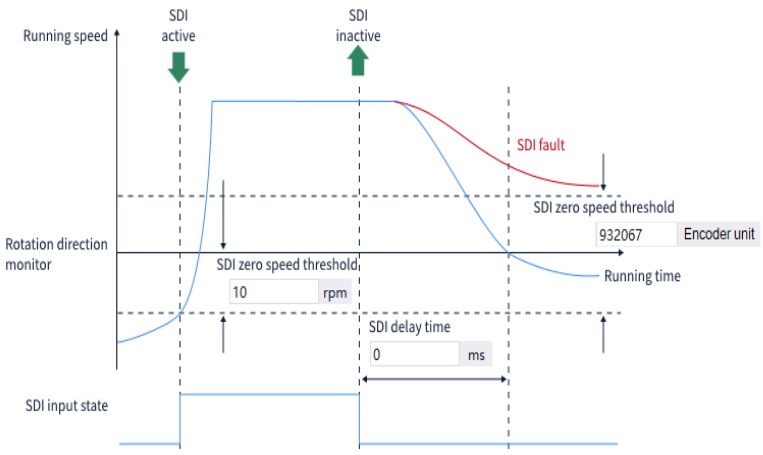
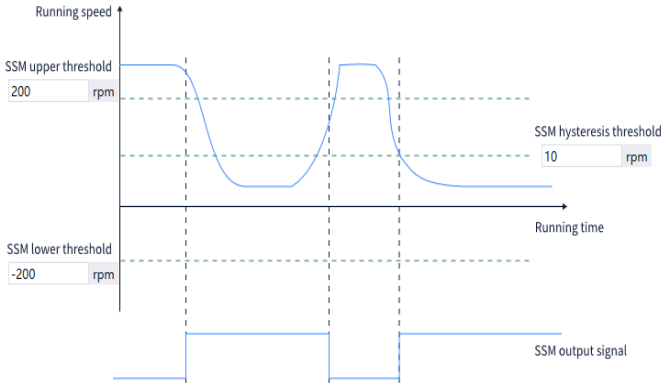

</td></tr>
</table>

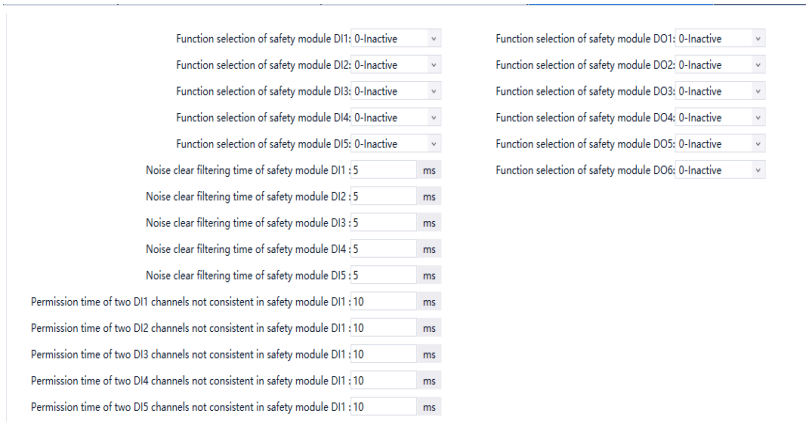
Function		Configuration
	SS1-t/SS2-t	 Configurable parameter: • SS1/SS2 ramp-to-stop reference speed: Ranges from 60 to 6000 rpm. • SS1/SS2 zero speed threshold: Ranges from 1 to 6000 rpm. • SS1 trigger-to-STO delay time/SS2 trigger-to-SOS delay time: Ranges from 0 to 65535 ms. • SS1/SS2 zero speed threshold time: Ranges from 0 to 65535 ms. • SS1/SS2 ramp-to-stop deceleration time: Ranges from 1 to 65535 ms.

Function			Configuration
	SS1-r/ SS2-r	Stop parameter setting	 Configurable parameter: • SS1/SS2 ramp-to-stop reference speed: Ranges from 60 to 6000 rpm. • SS1/SS2 zero speed threshold: Ranges from 1 to 6000 rpm. • SS1 trigger-to-STO delay time/SS2 trigger-to-SOS delay time: Ranges from 0 to 65535 ms. • SS1/SS2 zero speed threshold time: Ranges from 0 to 65535 ms. • SS1/SS2 ramp-to-stop deceleration time: Ranges from 1 to 65535 ms.
		Monitor parameter setting	 Configurable parameter: • SS1/SS2 ramp monitor reference speed: Ranges from 60 to 6000 rpm. • SS1/SS2 ramp monitor stop speed: Ranges from 0 to 6000 rpm. • SS1/SS2 ramp monitor start delay time: Ranges from 5 to 65535 ms. • SS1/SS2 max. ramp monitor time: Ranges from 1 to 65535 ms.

Function			Configuration
		Monitor limit violation parameter setting	 Configurable parameter: SS1/SS2 ramp-limit-trigger-to-alarm time, input range: 0 to 65535 ms
SOS speed monitor			 Configurable parameter: • SOS speed change threshold: Ranges from 1 to 6000 rpm. • SOS overthreshold-to-alarm time: Ranges from 0 to 65535 ms.

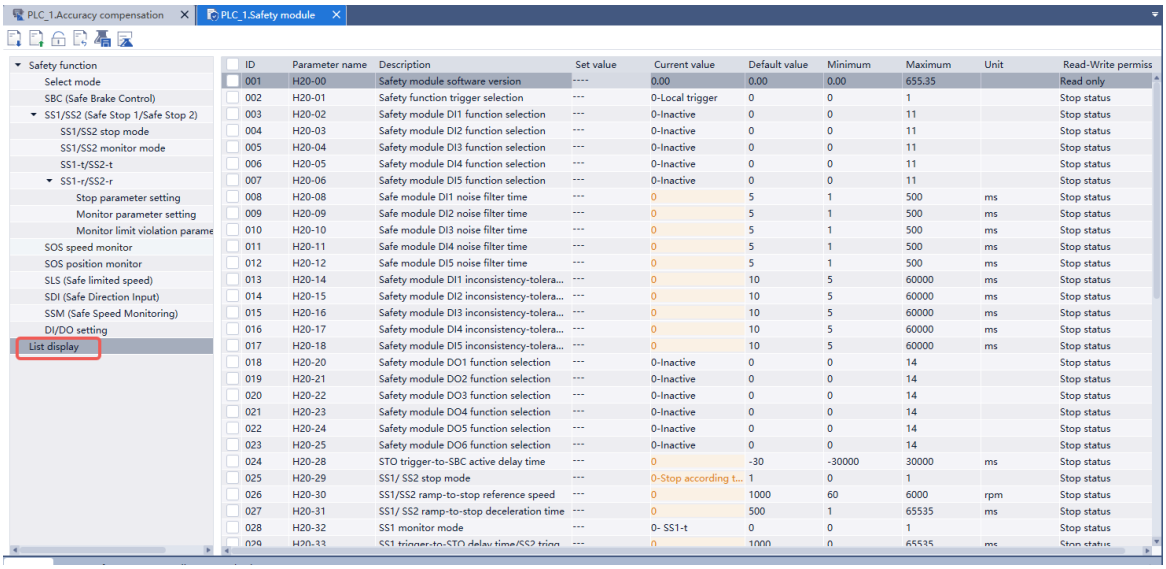
Function	Configuration
SOS position monitor	 Configurable parameter: • SOS position change threshold: Ranges from 1 to 536870912 pulses. • SOS overthreshold-to-alarm time: Ranges from 0 to 65535 ms.
SLS (Safe limited speed)	 x=1 x=2 x=3 x=4 Switchover delay of SLS 100 ms 100 ms 100 ms 100 ms Speed limit setting of SLS 1000 rpm 2000 rpm 3000 rpm 4000 rpm SLS limit filtering time 100 ms 100 ms 100 ms 100 ms Configurable parameter: • Switchover delay of SLS: Ranges from 0 to 65535 ms. • Speed limit setting of SLS: Ranges from 0 to 6000 rpm. • SLS limit filtering time: Ranges from 0 to 65535 ms.

Function	Configuration
SDI (Safe direction input)	 Configurable parameter: • SDI zero speed threshold: Ranges from 0 to 3000 rpm. • SDI zero position threshold: Ranges from 1 to 536870912 pulses. • SDI delay time: Ranges from 0 to 65535 ms.
SSM (Safe Speed Monitoring)	 Configurable parameter: • SSM upper threshold: Ranges from -6000 to 6000 rpm. • SSM lower threshold: Ranges from -6000 to 6000 rpm. • SSM hysteresis threshold: Ranges from 0 to 6000 rpm.

Function	Configuration
DI/DO setting	Select the functions of safety module DI/DO from the drop-down lists; set the noise clear filtering time of safety module DI and the permission time of two inconsistent DI channels via the input boxes. • Noise clear filtering time: Ranges from 1 to 500 ms. • Permission time of two inconsistent DI channels: Ranges from 5 to 60000 ms. 

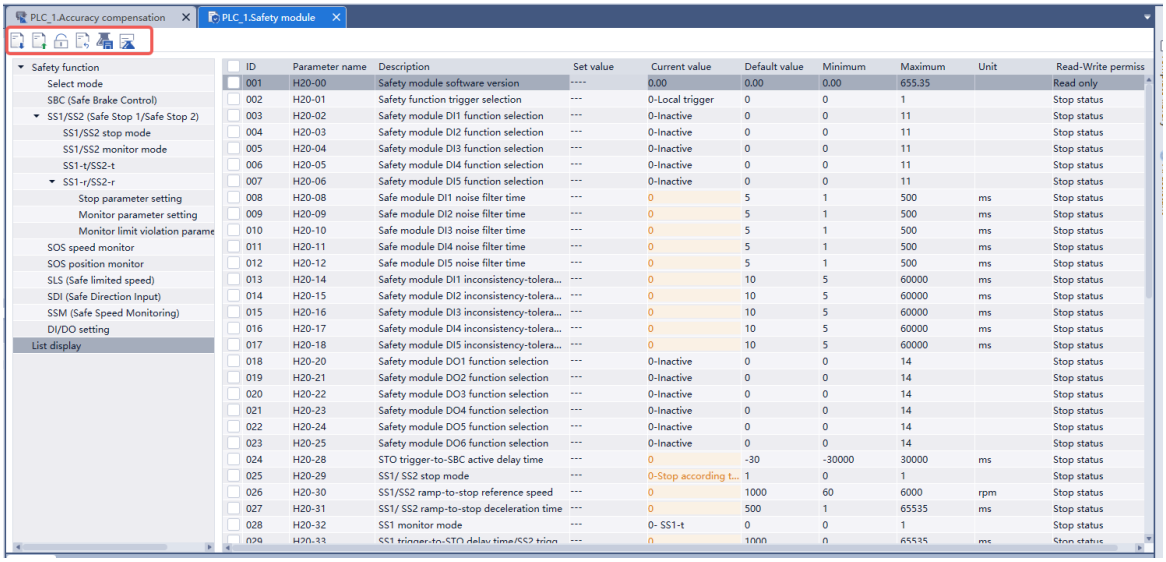
- **Parameter editing**

  The parameter editing area displays all safety function parameters in a list format. You can modify the set values of relevant parameters in the list and perform corresponding operations via the toolbar.



- **Toolbar**

  The toolbar contains the following function buttons as shown in the figure below. See the table below for detailed descriptions of each function button.



Function		Description
	Download	Downloads the selected safety parameters with set values from the parameter editing area to the drive. If password is required, the default password is "11111".
	Upload	Uploads the selected safety parameters from the drive and update them in the parameter editing area. If password is required, the default password is "11111".
	Password modification	Modifies the corresponding password. The default password is "11111".
	Restore default	Restores safety functions to factory settings.
	Export data	Saves the selected recipe parameters to the local disk (file extension: safetyparam).
	Import data	Imports the parameters from a local recipe file to the safety parameter module.

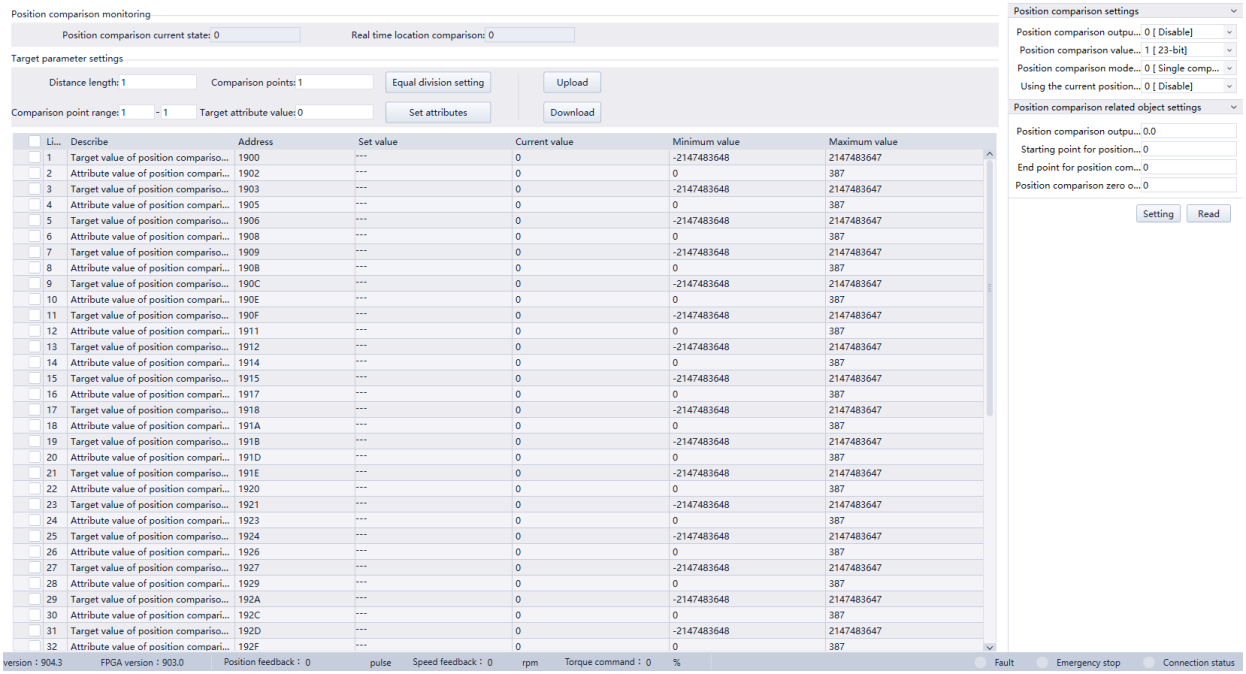
17.3.16 Comparison Output

This function is currently not supported for STD60N, IS650N-PLUS, and IS810N series products.

"Position comparison setting" and "Related object setting for position comparison" are general configurations for all models, while "ABZ output function setting" and "ABZ polarity selection" are supported only for certain products. Refer to the actual interface display for availability.

The position comparison function parameters are configured and monitored using the backend software. This function compares the instantaneous position data fed back from the drive with the data pre-stored in a target position array. When the comparison condition is met, it immediately outputs a DO pulse signal (with configurable DO number and pulse width) for subsequent motion control.

Select a servo drive in the project tree, and double-click "Position comparison" in the detailed view to open the interface, as shown in the figure below.



Steps

1. Configure position comparison settings.
   - Position comparison parameters are written immediately upon setting.
   - The following table shows the supported configuration items and their corresponding relationships for position comparison setting.

ID	Position comparison output enable	Position comparison output feedback source	Position comparison mode selection
1	1 [Enable (rising edge-triggered)]	0 [Motor encoder feedback]	0 [Individual comparison mode]
2	1 [Enable (rising edge-triggered)]	0 [Motor encoder feedback]	1 [Cyclic comparison mode]
3	1 [Enable (rising edge-triggered)]	0 [Motor encoder feedback]	2 [Fixed cyclic comparison mode]
4	1 [Enable (rising edge-triggered)]	1 [Fully closed-loop position feedback]	0 [Individual comparison mode]
5	1 [Enable (rising edge-triggered)]	1 [Fully closed-loop position feedback]	1 [Cyclic comparison mode]
6	1 [Enable (rising edge-triggered)]	1 [Fully closed-loop position feedback]	2 [Fixed cyclic comparison mode]
7	0 (disable)	0 [Motor encoder feedback]	0 [Individual comparison mode]
8	0 (disable)	0 [Motor encoder feedback]	1 [Cyclic comparison mode]
9	0 (disable)	0 [Motor encoder feedback]	2 [Fixed cyclic comparison mode]

ID	Position comparison output enable	Position comparison output feedback source	Position comparison mode selection
10	0 (disable)	1 [Fully closed-loop position feedback]	0 [Individual comparison mode]
11	0 (disable)	1 [Fully closed-loop position feedback]	1 [Cyclic comparison mode]
12	0 (disable)	1 [Fully closed-loop position feedback]	2 [Fixed cyclic comparison mode]

- "Position comparison output feedback source", "Resolution of position comparison value", and "Position comparison mode selection" are effective only when "Position comparison output enable" is set to "1 [Enable (rising edge-triggered)]". You can select the resolution of position comparison value according to actual needs.
- Enable "Current position as zero" only when the comparison status is invalid; otherwise, it may cause abnormal behavior in the comparison logic.

2. Configure "ABZ output function setting" and "ABZ polarity selection".
   - Click "Setting" to write the parameters set in "ABZ output function setting". (This function is not supported for SV660N series product)
   - The following table shows the supported configuration items and their corresponding relationships for ABZ output function settings.

No.	OCZ Port Function	Z Port Function	A/B Port Function	ABZ Output Function Setting
1	Frequency division output	Frequency division output	Frequency division output	0
2	Position comparison	Frequency division output	Frequency division output	1
3	Frequency division output	Position comparison	Frequency division output	2
4	Position comparison	Position comparison	Frequency division output	3
5	Frequency division output	Frequency division output	Position comparison	4
6	Position comparison	Frequency division output	Position comparison	5
7	Frequency division output	Position comparison	Position comparison	6
8	Position comparison	Position comparison	Position comparison	7

- Click "Setting" to write the parameters set in "ABZ polarity setting". (This function is not supported for SV660N series product)
- The following table shows the supported configuration items and their corresponding relationships for ABZ polarity selection.

ID	OCZ Port Output Logic	Z Port Output Logic	A/B Port Output Logic	ABZ Output Polarity Selection
1	Positive polarity, output high level when effective	Positive polarity, output high level when effective	Positive polarity, output high level when effective	0
2	Negative polarity, output low level when effective	Positive polarity, output high level when effective	Positive polarity, output high level when effective	1
3	Positive polarity, output high level when effective	Negative polarity, output low level when effective	Positive polarity, output high level when effective	2
4	Negative polarity, output low level when effective	Negative polarity, output low level when effective	Positive polarity, output high level when effective	3
5	Positive polarity, output high level when effective	Positive polarity, output high level when effective	Negative polarity, output low level when effective	4
6	Negative polarity, output low level when effective	Positive polarity, output high level when effective	Negative polarity, output low level when effective	5
7	Positive polarity, output high level when effective	Negative polarity, output low level when effective	Negative polarity, output low level when effective	6
8	Negative polarity, output low level when effective	Negative polarity, output low level when effective	Negative polarity, output low level when effective	7

3. Configure the parameters in "Related objects setting for position comparison". Click "Setting" to write the parameters set in "Related objects setting for position comparison".

Parameter	Description
Fixed number of cycles	Configures the number of cycles required for completion in the fixed cyclic comparison mode. Range: 1 to 65535.
Output width of position comparison	When the position comparison condition is met, the active level width of the DO can be set. Range: 0.1 to 204.7 ms.
Start point of position comparison	Position of the first comparison point. Range: 0 to 40 pulses.

Parameter	Description
End point of position comparison	Position of the last comparison point. Range: 0 to 40 pulses.
Zero offset of position comparison	Offset value after setting the current configured position as the zero point. Range: -2147483648 to 2147483647 pulses.

4. Read and set related parameters.
   - Click the "Setting" button to write related parameters for position comparison.
   - When the view becomes visible from a hidden status, click the "Read" button to read the related object parameters for position comparison from the online device.
5. Configure target parameters, and view position comparison monitoring information, target values, attribute values, etc.
   - Position comparison monitor area: Displays the current statuses and real-time position of the position comparison.
   - See the table below for the descriptions of target parameter setting.

Parameter	Description
Distance length	In individual comparison mode, it indicates the total travel distance of the servo drive; in cyclic or fixed comparison mode, it indicates the travel distance between two adjacent points. Range: -2147483648 to 2147483647 pulses.
Comparison points	In individual comparison mode, it indicates the total number of comparison points for the servo drive; in cyclic or fixed comparison mode, it indicates the number of points to be cyclically compared. Range: 1 to 40 points.
Comparison point range	Sets the number of points to be compared within the range of total comparison point count. Range: 1 to 40 points.
Target property value	Property value of the target position. Range: -2147483648 to 2147483647 pulses.
Equal division setting	Divides the target distance length equally according to the number of comparison points.
Upload	Reads the current values of the relevant position comparison parameters from the current axis of the connected drive and update them to the editing interface.
Download	Writes the set parameters from the current position comparison parameter editing interface to the target axis of the currently connected drive.

Parameter	Description
Setting property	Writes the target property-related parameters from the current target parameter editing interface to the target axis of the currently connected drive.

- List area: Reads and displays related target values and property values during the drive position comparison process.

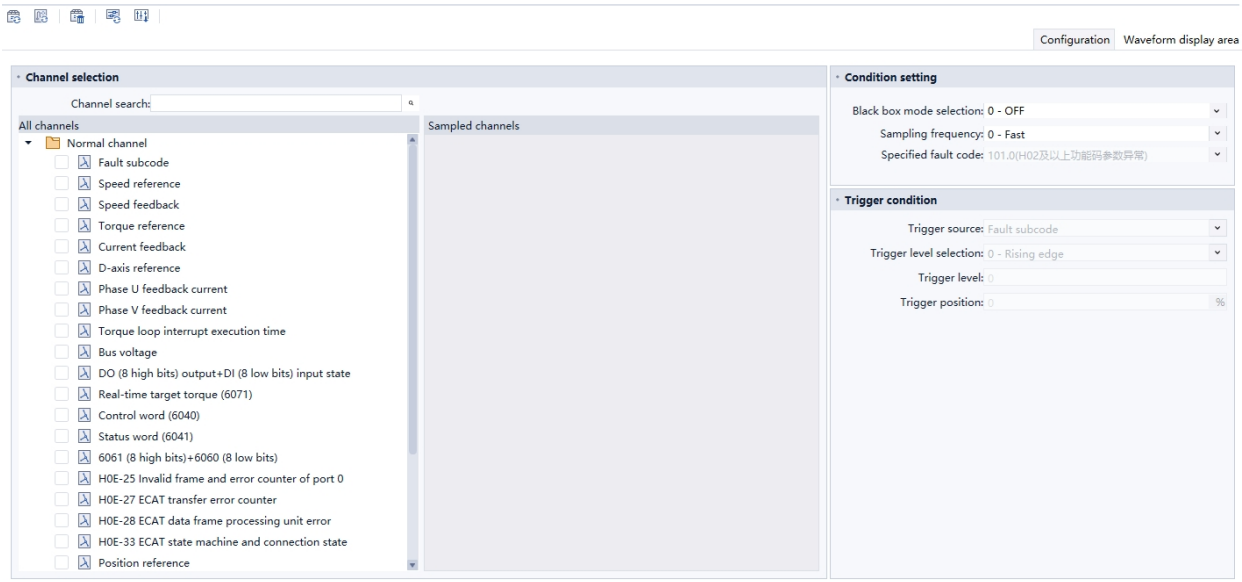
17.3.17 Black Box

17.3.17.1 Overview

This function analyzes the recorded data, operation statuses, and fault causes of the related devices to improve their maintainability.

Select a servo device in the project tree, and double-click "Black box" in the detailed view to open the interface, which contains two tabs: "Configuration" and "Waveform display area".

17.3.17.2 Configuration



- **Toolbar**

Function		Description
	Read black box	Reads the black box data, and then jumps to the waveform display area.
	Clear black box	Clears the black box data.

Function		Description
	Read configuration	Reads the configuration from the drive and displays it on the current page.
	Reading all channels	Reads all channel information from the black box. NOTE This function is not supported by the SV660N series products.
	Write configuration	Writes the configuration of the current page to the drive.

- **Axis selection**

  Selects the corresponding axis of the drive.

- **Channel selection**

  Selects the black box channel for data reading. You can search for the related channels quickly through the "Channel search" box.

- **Trigger condition**

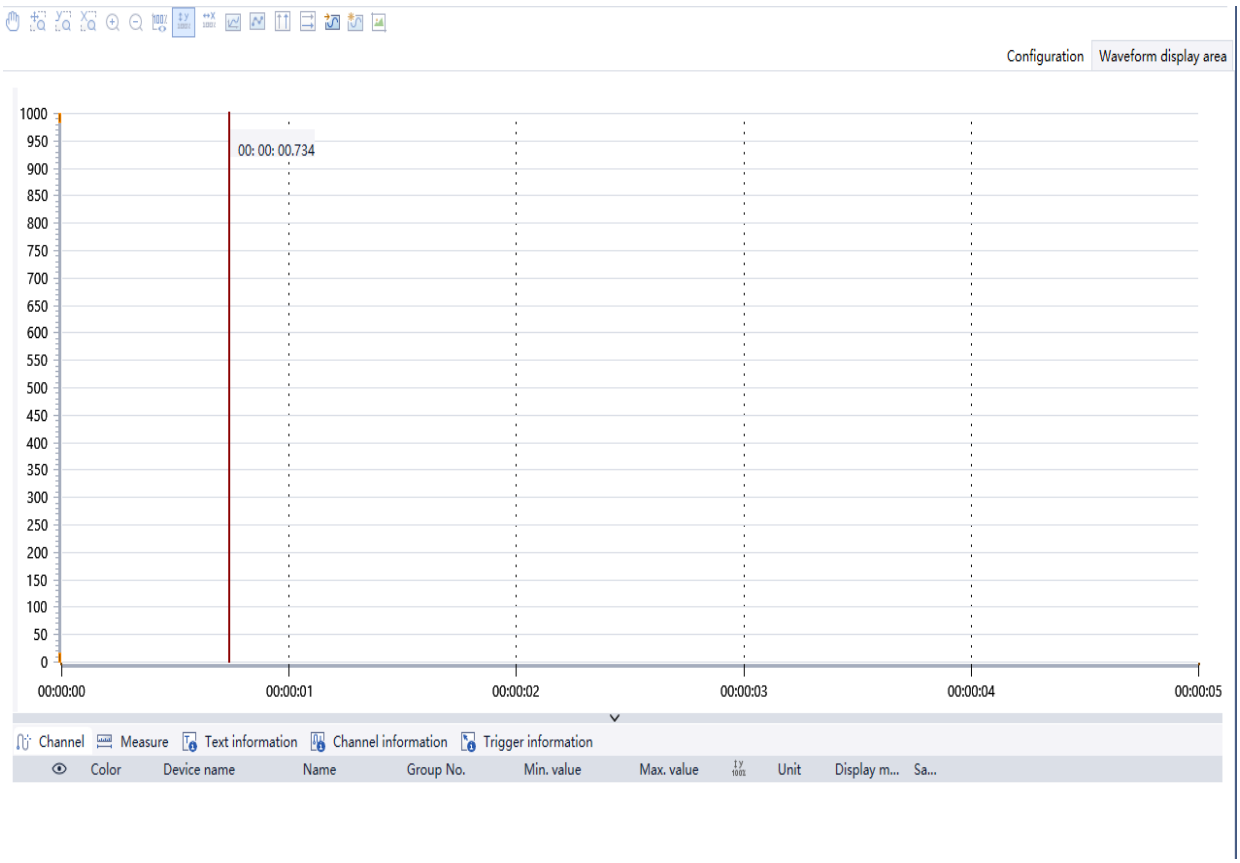
Parameter	Description
Trigger source	Defines the trigger source. It is available only when the black box mode is set to "3 - Triggered by specified condition".
Trigger level	Defines the trigger level. The value ranges from 0 to 65535 pulses. It is available only when the black box mode is set to "3 - Triggered by specified condition".
Trigger level selection	Defines the trigger level selection. It is available only when the black box mode is "3 - Triggered by specified condition".
Trigger position	Defines the trigger position. The value ranges from 0% to 100%.

- **Condition setting**

Parameter	Description
Sampling frequency	Defines the sampling frequency. Options are as follows: • 0 - Fast (16 kHz) • 1 - Normal (4 kHz) • 2 - Slow (1 kHz)

Parameter	Description
Black box mode selection	Selects the black box mode. Options are as follows: • 0 - OFF • 1 - Any fault • 2 - Specified fault • 3 - Triggered by specified condition
Specified fault code	Defines the fault code to trigger the black box function. It is available only when the black box mode is set to "2 - Specified fault".

17.3.17.3 Waveform Display



The toolbar on the "Waveform display area" tab is the same as that on the "Oscilloscope" tab. For details, see 17.2.1.8 Analysis Auxiliary Function. The waveform curve is displayed in the middle area and related information is displayed in the lower area. See the following table for details.

Function	Description
Channel	Displays channel information. You can also set the channel color and group number.
Measurement	Measures the waveform through the cursor, and displays the values of each curve in real time.

Function	Description
Channel information	Displays the sampling frequency, total sampling point count, and sampling point count before trigger of the speed and position references.
Trigger information	Displays the following two trigger messages: • Condition information: Displays the trigger condition based on the actual value. • Fault information: Displays the fault information generated upon triggering. NOTE The trigger information of the current servo drive is displayed as null.
Text information	Displays relevant information about the device.

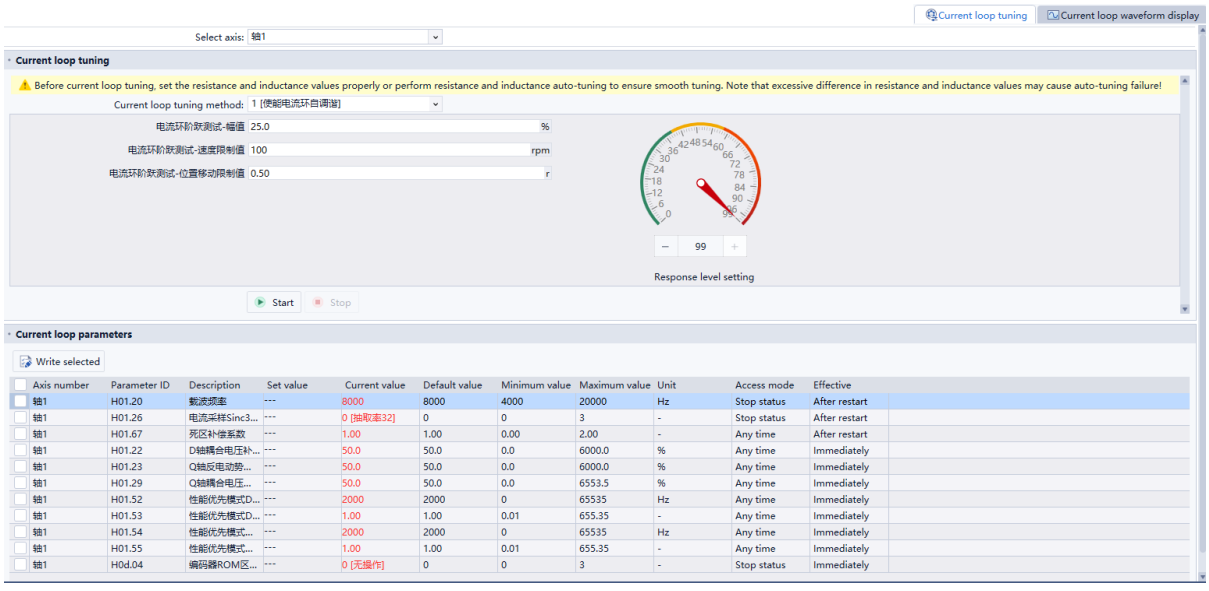
17.3.18 Current Loop Adjustment

This function is supported only by the IS650N-PLUS, SV680N-INT, SV670N-INT, and SV680LN series.

As the innermost loop of a servo control system, the response and stability of the current loop play a vital role in the performance of the system.

17.3.18.1 Function Interface

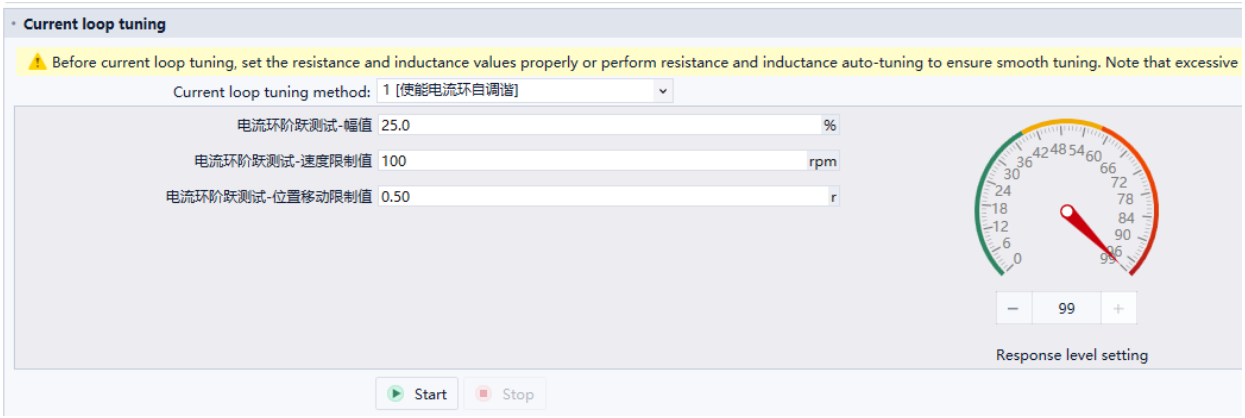
Select a servo device in the project tree, and double-click "Current loop tuning" in the detailed view to open the interface. The interface mainly consists of three parts: current loop tuning, current loop parameters, and current loop waveform display. The interface is shown in the following figure.



17.3.18.2 Current Loop Tuning

17.3.18.2.1 Current Loop Auto-Tuning

This mode provides a static auto-tuning method for the current loop. After this mode is enabled, the drive executes gain parameter tuning for the current loop automatically. You can set the response level. The higher the response level, the higher the responsiveness and bandwidth of the current loop.



Description	Description
Current loop step test - Amplitude	Sets the amplitude of the excitation signal for the auto-tuning process.
Current loop step test - Speed limit value	Sets the maximum speed at which the current excitation can remain stable.
Current loop step test - Displacement limit value	Sets the maximum distance at which the current excitation can remain stable.
Response level setting	Sets the response level for the current loop.

Description

Excessively high response level may induce current loop control noise. In actual use, users shall set the response level as needed to achieve a balance between current loop responsiveness and noise.

17.3.18.2.2 Static Step Test of Current Loop

This mode provides a static method for observing the current loop step response. Users can manually adjust the current loop gain parameters to observe and verify the current loop response.



Description	Description
Current loop step test - Amplitude	Sets the amplitude of the step excitation signal.
Current loop step test - Duration	Sets the duration of the step excitation signal.
Current loop step test - Speed limit value	Sets the maximum speed at which the current excitation can remain stable.
Current loop step test - Displacement limit value	Sets the maximum distance at which the current excitation can remain stable.

17.3.18.2.3 Step Test of Current Loop

This mode provides a dynamic method for observing the current loop step response. Its parameter settings are the same as static step test. Users can manually adjust the current loop gain parameters to observe and verify the current loop response.



17.3.18.3 Current Loop Parameters

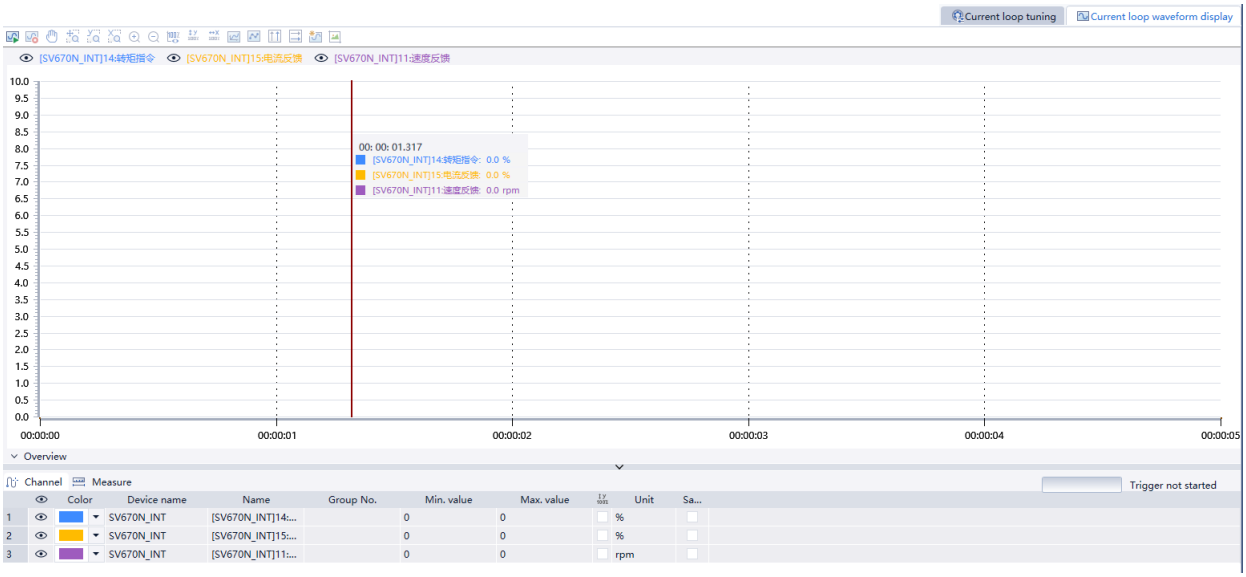
The current loop parameter interface is used to manually set and monitor current loop gain parameters. The parameter settings in this interface are consistent with those in the parameter list. For details, see 17.2.2 Parameter List.



Axis number	Parameter ID	Description	Set value	Current value	Default value	Minimum value	Maximum value	Unit	Access mode	Effective
轴1	H01.20	载波频率	---	8000	8000	4000	20000	Hz	Stop status	After restart
轴1	H01.26	电流采样Sinc3...	---	0 [抽取率32]	0	0	3	-	Stop status	After restart
轴1	H01.67	死区补偿系数	---	1.00	1.00	0.00	2.00	-	Any time	After restart
轴1	H01.22	D轴耦合电压补...	---	50.0	50.0	0.0	6000.0	%	Any time	Immediately
轴1	H01.23	Q轴反电动势...	---	50.0	50.0	0.0	6000.0	%	Any time	Immediately
轴1	H01.29	Q轴耦合电压...	---	50.0	50.0	0.0	6553.5	%	Any time	Immediately
轴1	H01.52	性能优先模式D...	---	2000	2000	0	65535	Hz	Any time	Immediately
轴1	H01.53	性能优先模式D...	---	1.00	1.00	0.01	655.35	-	Any time	Immediately
轴1	H01.54	性能优先模式...	---	2000	2000	0	65535	Hz	Any time	Immediately
轴1	H01.55	性能优先模式...	---	1.00	1.00	0.01	655.35	-	Any time	Immediately
轴1	H0d.04	编码器ROM区...	---	0 [无操作]	0	0	3	-	Stop status	Immediately

17.3.18.4 Current Loop Waveform Display

After completing the static step test and step test of the current loop, the system will automatically jump to the current loop waveform display area and trigger the data acquisition of torque reference, current feedback, and speed feedback. Users can observe and verify the current loop response through the waveform.



The toolbar in this interface has the same functions as that in the "Trigger oscilloscope" interface. For details, see 17.2.1.8 Trigger Oscilloscope.

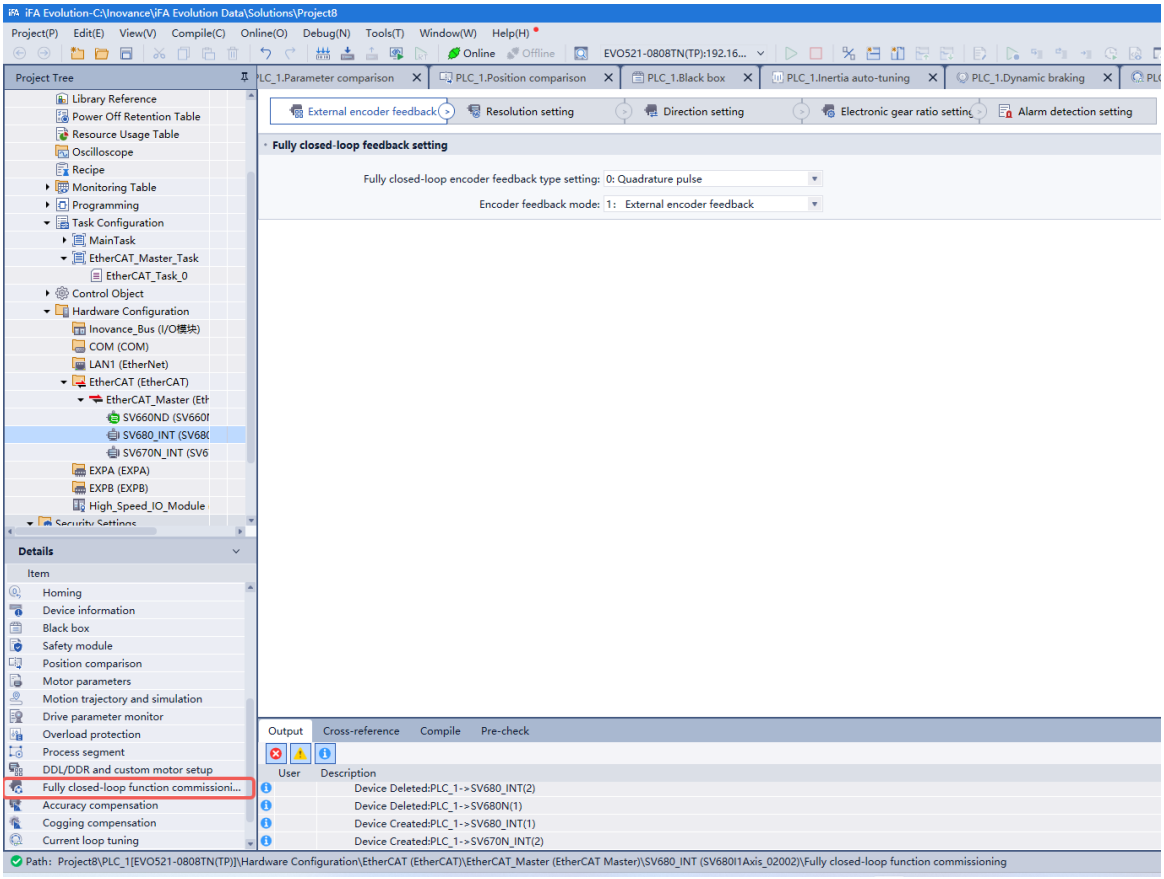
17.3.19 Fully Closed-Loop Functions

This function currently supports only the IS650N-PLUS, SV680-INT, and SV670N-INT product series.

The fully closed-loop control method installs position detection devices (such as linear encoders and encoders) on the moving components to provide real-time feedback on motion status and position information, eliminating the impact of temperature and mechanical errors on control performance and improving motion accuracy. This function is designed to guide users in quickly configuring the fully closed-loop function.

Function entry

Select a servo drive in the project tree and double-click "Fully closed-loop function commissioning" in the detailed view to open the interface.

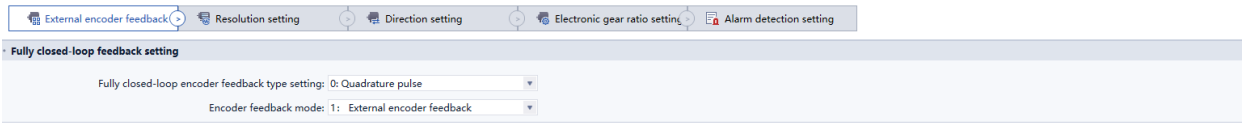


Function description

After setting basic gain parameters, check that the servo drive operates properly without overshoot and stops without generating noise. After the basic operating conditions are met, the fully closed-loop parameters need to be configured, which contains the following steps: external encoder feedback type setting, resolution setting, direction setting, electronic gear ratio setting, and alarm detection setting.

17.3.19.1 External Encoder Feedback Type Settings

The first step is to set the fully closed-loop encoder feedback type and encoder feedback mode. The interface is as follows.



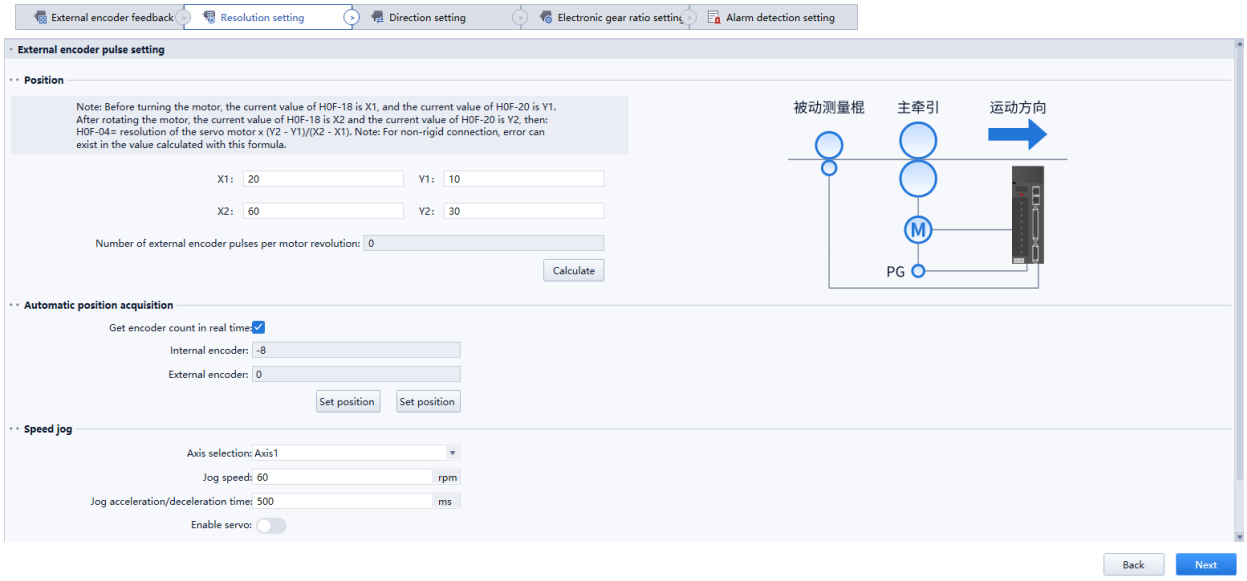
If the encoder is of a third-party type (SSI/ENDAT/BISSC), the system will prompt you to power off and on the device again after parameter configuration. When the encoder feedback mode is set to internal encoder mode, the next step (resolution setting) cannot be performed.

17.3.19.2 External Encoder Resolution Settings

This step mainly aims to calculate the number of pulses of the external encoder, which can be done in the following two ways:

1. Manually rotate the motor while observing the internal encoder pulse counter. After completing one full revolution, observe the value change in the pulse counter. The absolute value of this change indicates the number of pulses per revolution of the fully closed-loop encoder.
2. Before rotating the motor, the current value of the internal position pulse feedback is X1, and the current value of the fully closed-loop encoder position pulse feedback is Y1. After rotating the motor, X1 becomes X2, and Y1 becomes Y2. The number of pulses per revolution of the fully -closed-loop encoder is: Servo motor resolution × (Y2 - Y1) ÷ (X2 - X1).

You can choose whether to use jog mode for motor revolution. The jog operation interface is hidden or displayed based on your selection, which is shown below:



Click "Next" to jump to the "Direction setting" tab. Click "Back" to jump to the "External encoder feedback type setting" tab

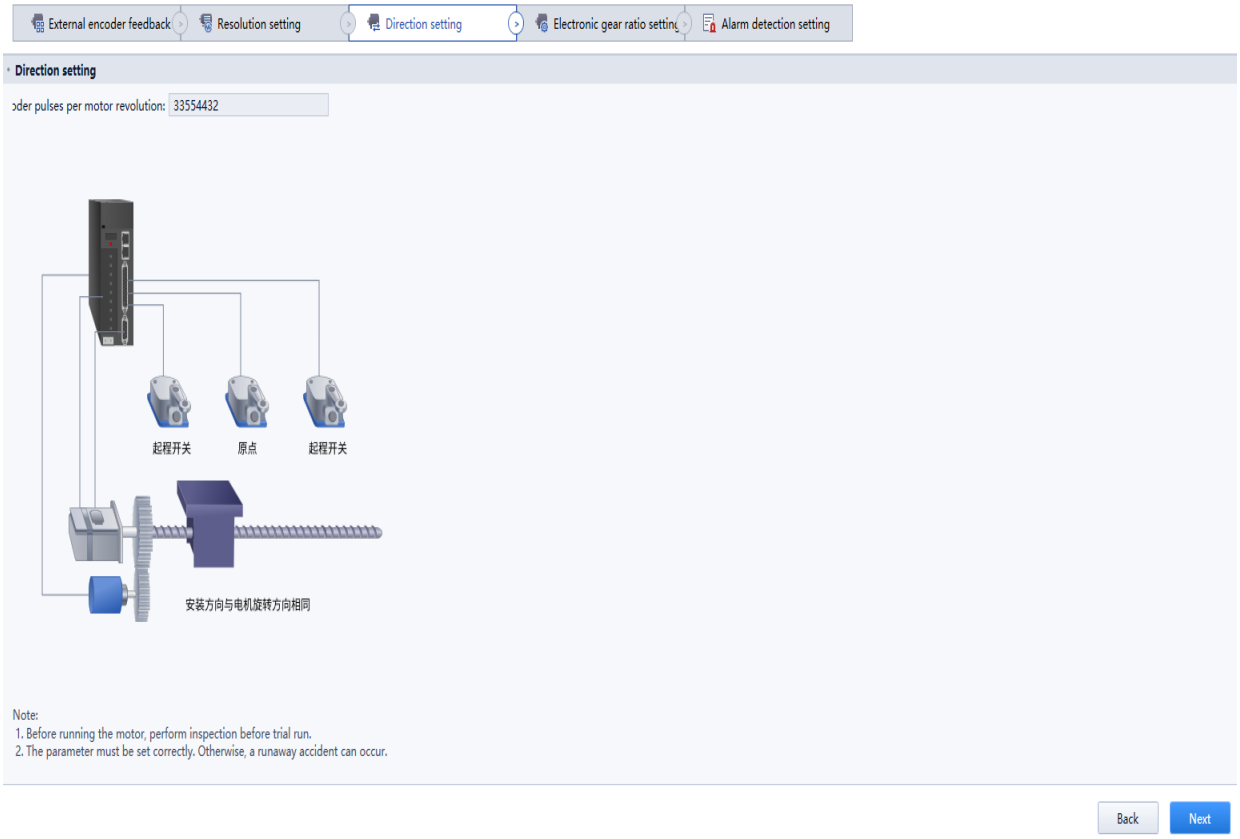
17.3.19.3 External Encoder Direction Settings

This step aims to determine the operation direction of the external encoder, and check whether this direction is consistent with that of the internal encoder. If not, positive feedback will be generated, leading to a runaway fault.

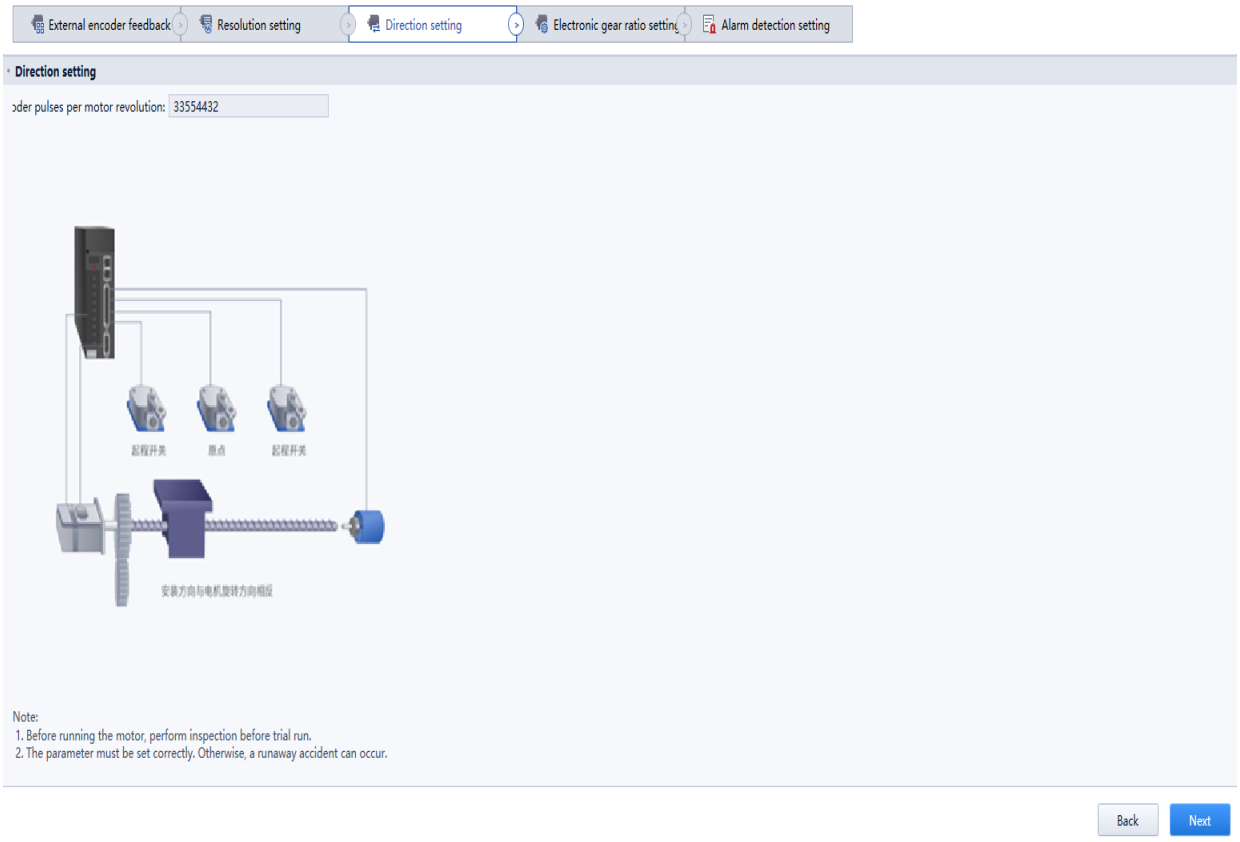
The operating direction can be determined based on the change in the number of pulses of the external encoder per motor revolution. If the change is negative, the direction is reverse; if the change is positive, the direction is forward. This interface does not involve parameter writing, but only determines whether the current condition matches the diagram. There are two main installation scenarios. Click "Next" to jump to the

"Electronic fear ratio setting" tab. Click "Back" to jump to the "Resolution setting" tab. The interface is as follows:

Installation direction same as motor revolution direction:



Installation direction opposite to motor revolution direction:



17.3.19.4 Electronic Gear Ratio Settings

This step sets the electronic gear ratio according to the encoder feedback mode. For the external encoder feedback mode, the first group of electronic gear ratios needs to be set on this tab. For the inter/outer loop switchover mode, both the first and second groups of electronic gear ratios need to be set. Click "Next" to jump to the "Alarm detection setting" tab. Click "Back" to jump to the "Direction setting" tab. The interface is as follows:



17.3.19.5 Alarm Detection Settings

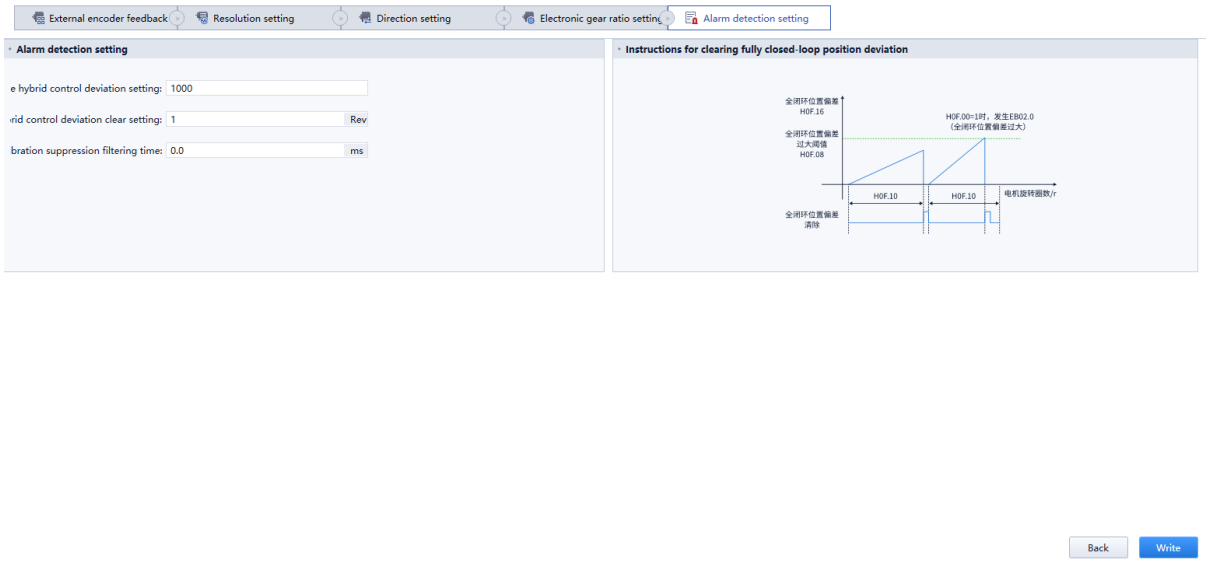
This step is the final step of this function. It is used to configure the deviation threshold, deviation clear setting, and first-order low-pass filter for deviation in compound control.

1. The deviation threshold sets the allowable positional tolerance between the motor and the external encoder. The unit is one reference unit (same as one external encoder unit). For example, if the deviation threshold is set to 1000, EB02.0 (Excessive position deviation in fully closed-loop mode) will be output when the deviation in compound control (displacement deviation between the motor-driven mechanism and the external encoder measurement) exceeds the displacement corresponding to 1000 external encoder pulses. The deviation threshold must be smaller than (fully closed-loop encoder pulse count per motor revolution) × (deviation reset value in compound control); otherwise, the fault code cannot be output.

2. The deviation clear value set the number of motor, upon the attainment of which the accumulated deviation in compound control will be cleared. For example, if the deviation clear value is set to 50, the system detects whether the accumulated deviation in compound control during 50 motor revolutions exceeds the deviation threshold. If so, a fault will be reported. If not, the accumulated deviation is cleared and the monitoring cycle is reinitialized after the motor completes 50 revolutions.

3. The first-order low-pass filter for deviation indicates the time constant of the first-order filter. It is used to filter vibrations in compound control deviation, enhancing the speed smoothness in the fully closed-loop mode.

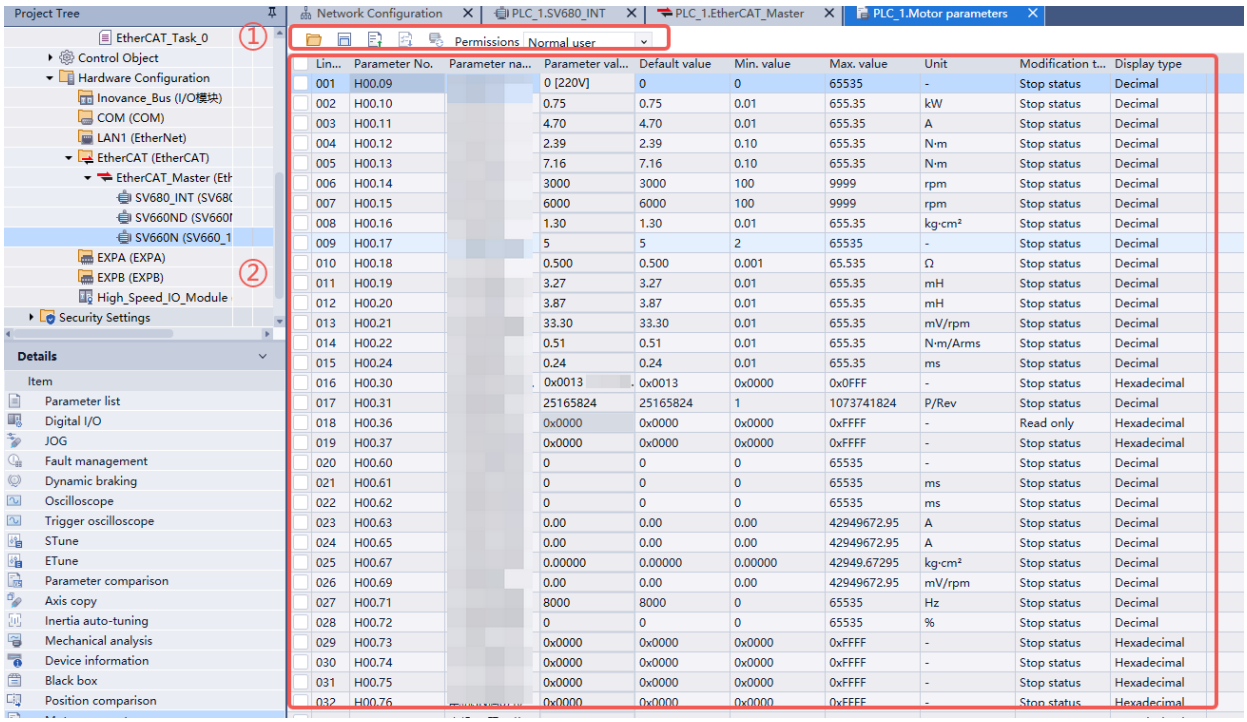
Click "Next" to write the configured parameters in this interface to the servo drive to complete the fully closed-loop commissioning. Click "Back" to jump to the "Electronic gear ratio setting" tab. The interface is shown in the figure below:



17.3.20 Bus Motor Parameters

- **Function interface**

Select a servo drive in the project tree, and double-click "Motor parameters" in the detailed view to open the interface, as shown in the figure below.



The function interface consists of two parts: the toolbar and the parameter editing area.

No.	Description	Description
①	Toolbar	Performs related operations on the motor bus parameter list.
②	Parameter editing area	Configures the various functions of the motor bus parameters, which can be used in conjunction with the toolbar.

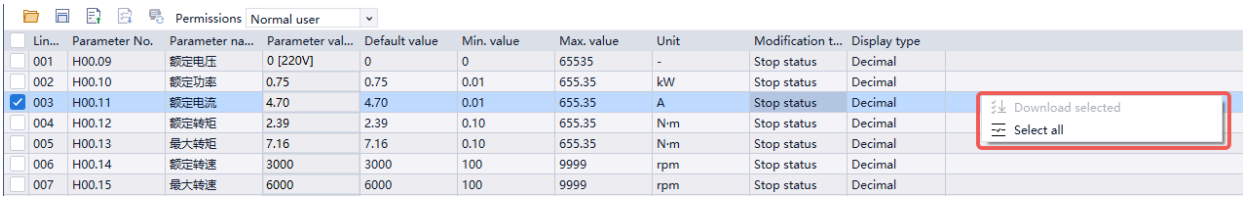
- **Toolbar**



Icon	Description	Configuration	Restriction
		1. Select an axis for the current device from the drop-down list. 2. Click "Parameter value" to modify the parameter through drop-down box selection or direct input.	/
	Open file	Opens the motor parameter file.	/
	Save file	Saves the motor parameter file.	/
	Upload all parameters	Uploads all motor parameters.	/
	Write selected parameters	Writes the selected parameter values to the device.	It is disabled under normal user permission and enabled for super administrators.
	Motor angle initialization	Initializes the motor angle.	Disabled under normal user permission, enabled under super administrator permission.
Permissions switchover		Switches permission of the current motor parameter table between "Normal user" and "Super administrator".	/

- **Download setting**

Right-click in the parameter editing area to pop up the download menu.



Function	Description	Restriction
Download selected	Writes the selected parameter values to the device.	It is disabled under normal user permission and enabled for super administrators.
Select all	Selects all parameters in the current table.	/
Deselect all	Deselects all parameters in the current table.	/

17.3.21 Process Segment

17.3.21.1 Overview

Function description

This feature is supported only for the STD60N series products.

The process segment wizard is a backend module designed to help users configure parameters related to process segments. The interface includes configuration of general parameters, process segment parameters, homing, process segments, event trigger, and commissioning.

The process segment function mainly consists of six functions: general parameter configuration, process segment parameter configuration, homing configuration, process segment configuration, event trigger configuration, and commissioning.

Function entry

Select a servo drive in the project tree and double-click "Process segment" in the detailed view to open the interface.

17.3.21.2 General Parameter Configuration

This tab involves configuration of general parameters, including axis selection, control mode selection, process segment deceleration time, electronic gear ratio settings, and positive/negative limit settings. The interface is as follows:

Click "Read" to read the corresponding parameter values from the drive and refresh them on the current page. Click "Download" to download all parameters on the current page.

17.3.21.3 Process Segment Parameter Configuration

This tab involves configuration of parameters related to the process segment, including acceleration/deceleration time, target speed, and waiting time.

Click "Read" to read the corresponding parameter values from the drive and refresh them in all tables. Click "Download" to download all parameters on the current page.

17.3.21.4 Homing Configuration

This tab involves configuration of homing method and homing process segment. The parameters on this tab are consistent with those in the "Homing" interface of the device.

Click "Read" to read the corresponding parameter values from the drive and refresh them on the current page. Click "Download" to download all parameters on the current page.

17.3.21.5 Process Segment Configuration

This function is supported only by the STD60N, SV680N-INT, SV670N-INT, SV680LN, and SVD60N series products.

There are 15 process segments, and each segment has the same function. Each process segment consists of its definition and data. There are 6 process segment operation modes by default:

0 - Undefined, 1 - Constant speed mode, 2 - Positioning mode: Stop when this segment is done running, 3 - Positioning mode: Load the next segment after this segment is done running, 7 - Jump to, 8 - Write parameters.

Interface introduction

When the operation mode is set to "0 - Undefined", there are no corresponding parameter settings, and this process segment performs no operation.

- **Constant speed mode**

This mode involves the configuration of process segment definition and data.

The process segment definition mainly includes configuration of whether to update immediately, whether to automatically update after DelayTime, speed unit, acceleration/deceleration time, and delay time.

The process segment data is used to set the target speed.

Positioning mode

This mode involves the configuration of process segment definition and data.

The process segment definition mainly includes configuration of whether to update immediately, whether to allow segment intersection, position reference type, acceleration/deceleration time, target speed, and delay time.

The process segment data is used to set the position reference.

After the settings of a certain process segment are completed, the next process segment will be automatically performed (for example, after the settings of process segment 1, process segment 2 will be automatically performed).

Jump mode

This mode involves the configuration of process segment definition and data.

The process segment definition mainly includes configuration of whether to update immediately and delay time.

The process segment data is used to set the process segment numbers for jumping.

- **Parameter-write mode**

This mode involves the configuration of process segment definition and data.

The process segment definition mainly includes configuration of whether to update immediately, whether to automatically update after DelayTime, whether to save to EEPROM, parameter group number, parameter number, and delay time.

The process segment data is used to set the parameter value to be written.

For STD60N series stepper devices, if H0d.99 parameter exists and its value is 1, the corresponding process segment will include an additional homing mode compared to the default settings.

- **Homing mode**

This mode only involves the configuration of process segment definition.

The process segment definition includes configuration of process segment numbers for jumping and delay time.

17.3.21.6 Event Trigger Configuration

This tab configures the rising/falling edge events used to trigger a specific process segment. Through the device's I/O functions, corresponding rising and falling edge events can be triggered. Currently, STD60N devices only support the trigger configuration of rising edge events.

17.3.21.7 Commissioning

After enabling the device, enter the corresponding process segment number in the text box. Click "Start" and the specified process segment starts running; click "Stop" and the process segment stops running.

Check "Display monitored process segment" and the text box below will display the running process segment number of the current device in real time.

The "Commissioning" tab includes configuration of servo enable switch, process segment run/stop, and process segment monitoring. The interface is as follows:

17.3.22 Overload Protection

This function is available only for the SV680N-INT, SV670N-INT, and SV680LN series.

Select a servo drive in the project tree, and double-click "Overload protection" in the detailed view to open the interface.

The overload protection interface mainly consists of overload solution selection, sub-interfaces of each solution, file loading, and file saving. The interface is as follows.

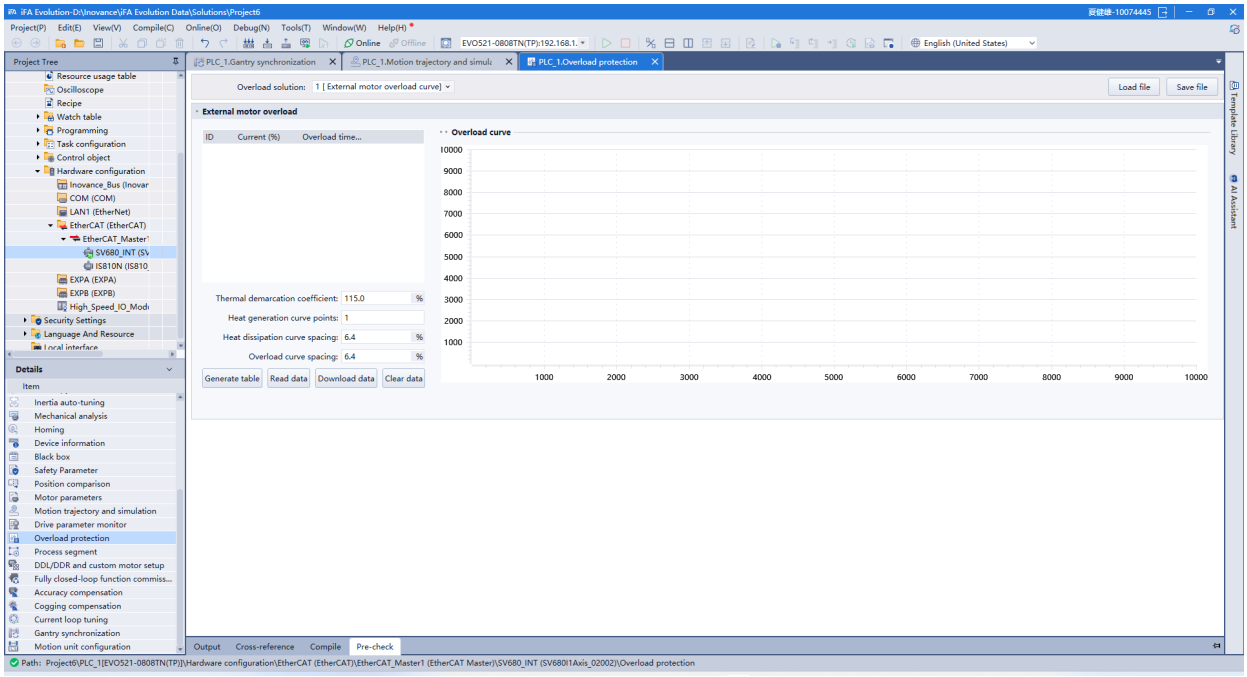
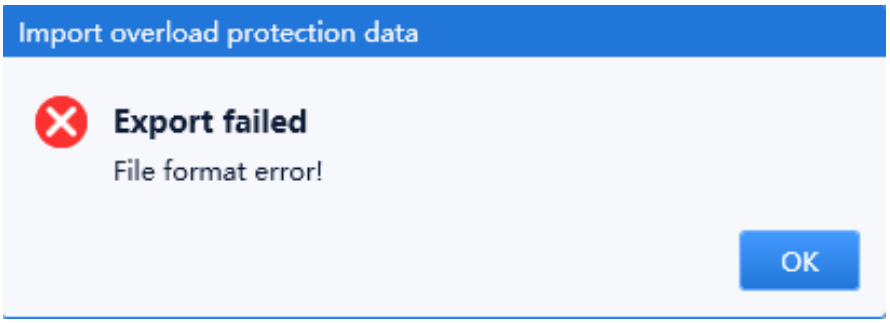
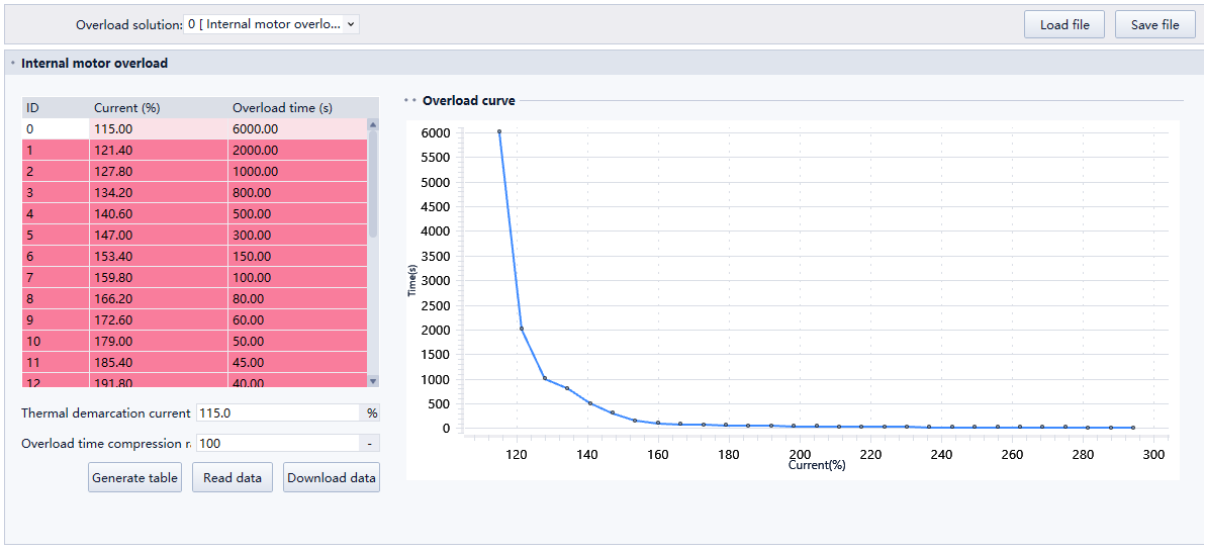


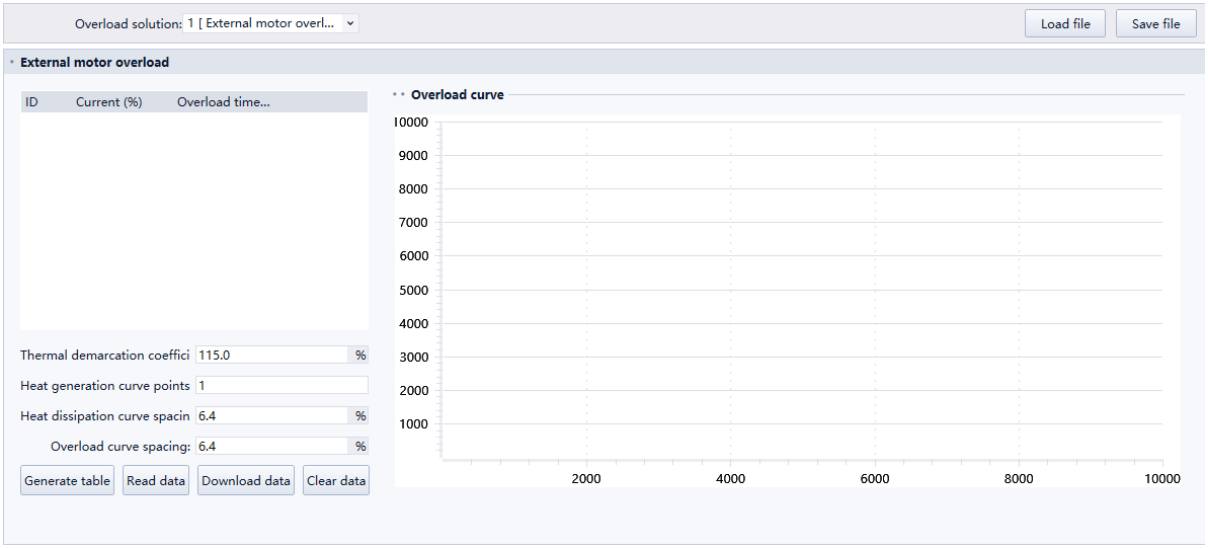
Table17-3-22-1

Description	Description
Overload solution	Downloads the solution type to the drive based on the selected solution. According to the selected solution, different solution views are switched.
Load file	Opens the solution saved on the local drive. Only files in the CSV format are supported. When importing overload protection data fails, the following dialog box is displayed. 
Save file	The saved content includes the overload solution type, overload protection data, and related parameters in turn. The file format is CSV.

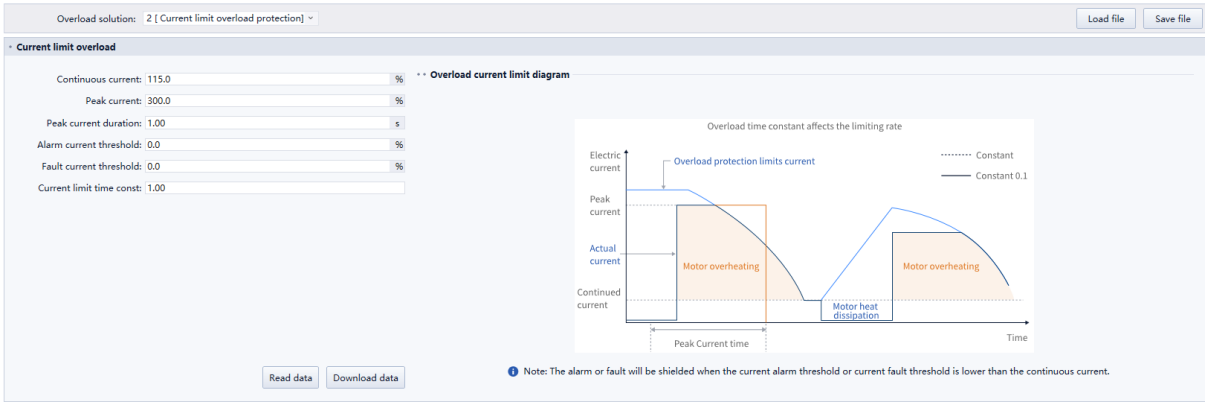
- The interface for "0 [Internal motor overload curve]" is as follows.



- The interface for "1 [External motor overload curve]" is as follows.



- The interface for "2 [Current limit overload protection]" is as follows.

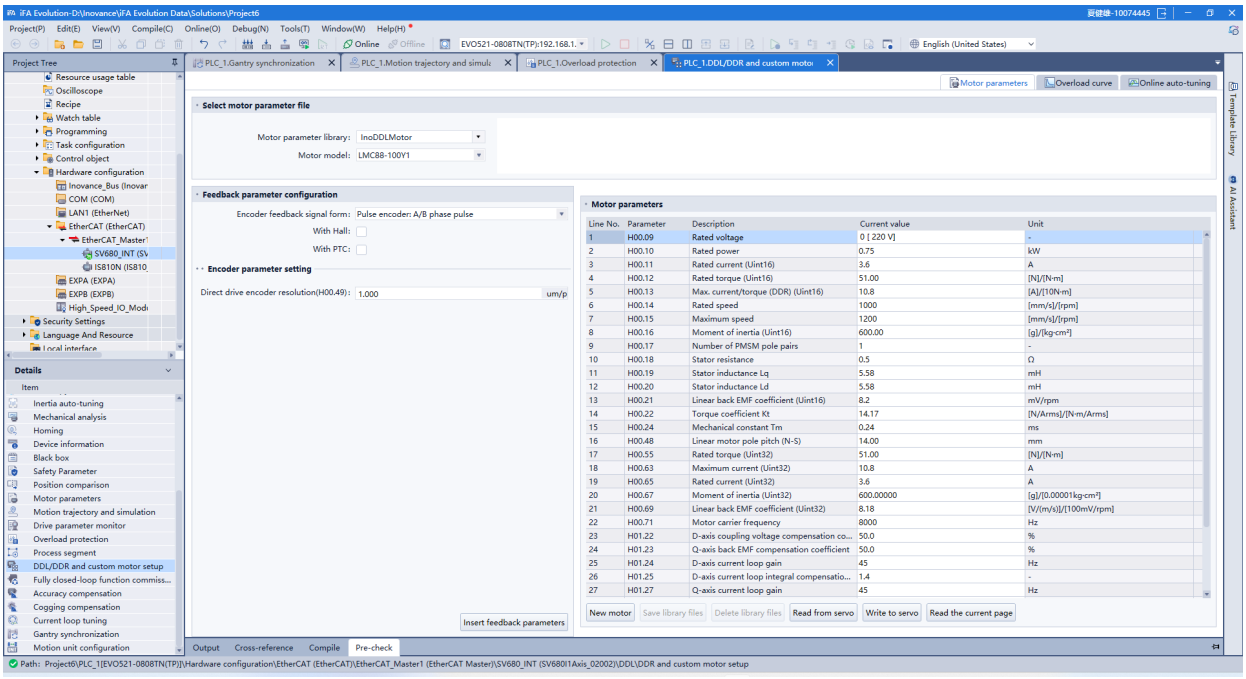


17.3.23 Motor Management Library

This function is available only to the SV680N-INT, SV670N-INT, and SV680LN series.

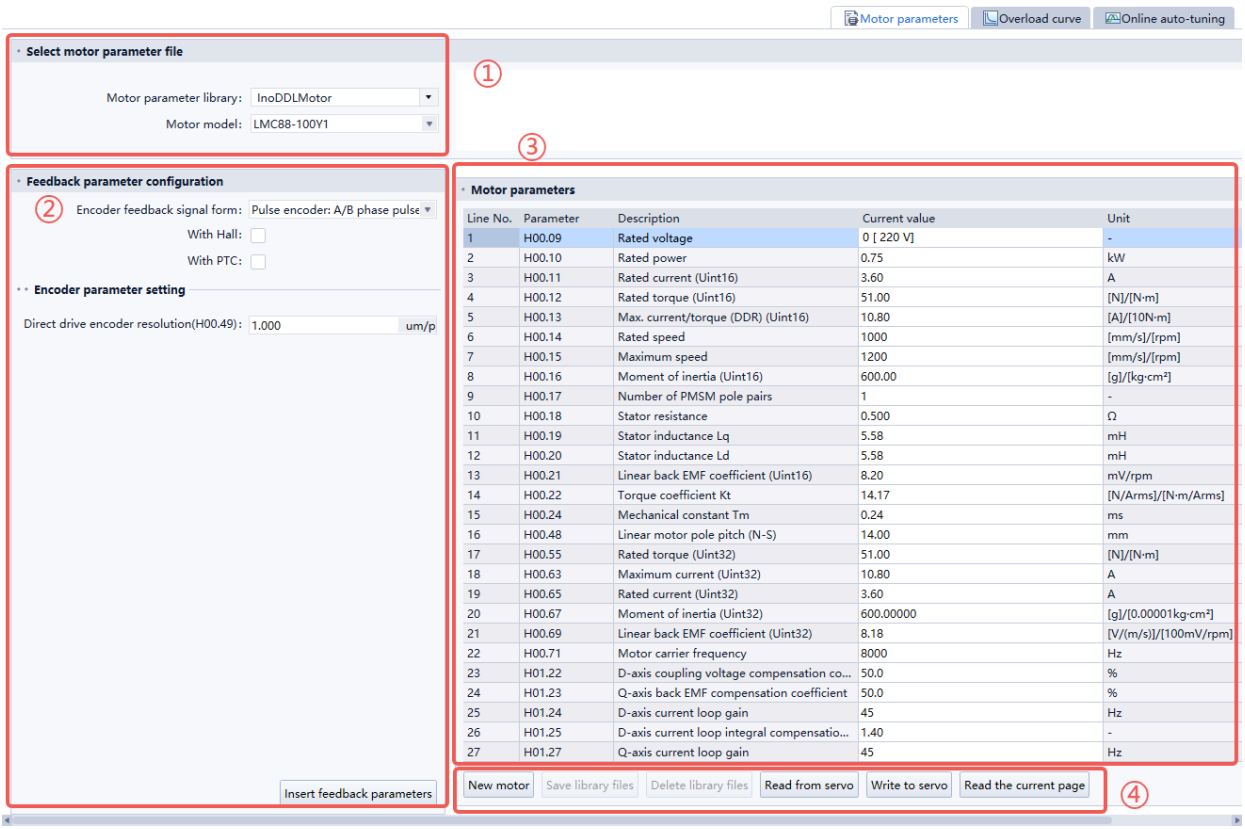
Select a servo drive in the project tree, and double-click "DDL/DDR and custom motor setup" in the detailed view to open the interface.

The interface is mainly composed of "Motor parameters", "Overload curve", and "Online auto-tuning". The interface is as follows.



- **Motor parameters**

On this tab, select the motor type. You can configure parameters of the corresponding motor type according to the wizard. The supported motor types vary with different devices.



No.	Description	Description
①	Select motor parameter file	Selects the motor parameter library and motor model.

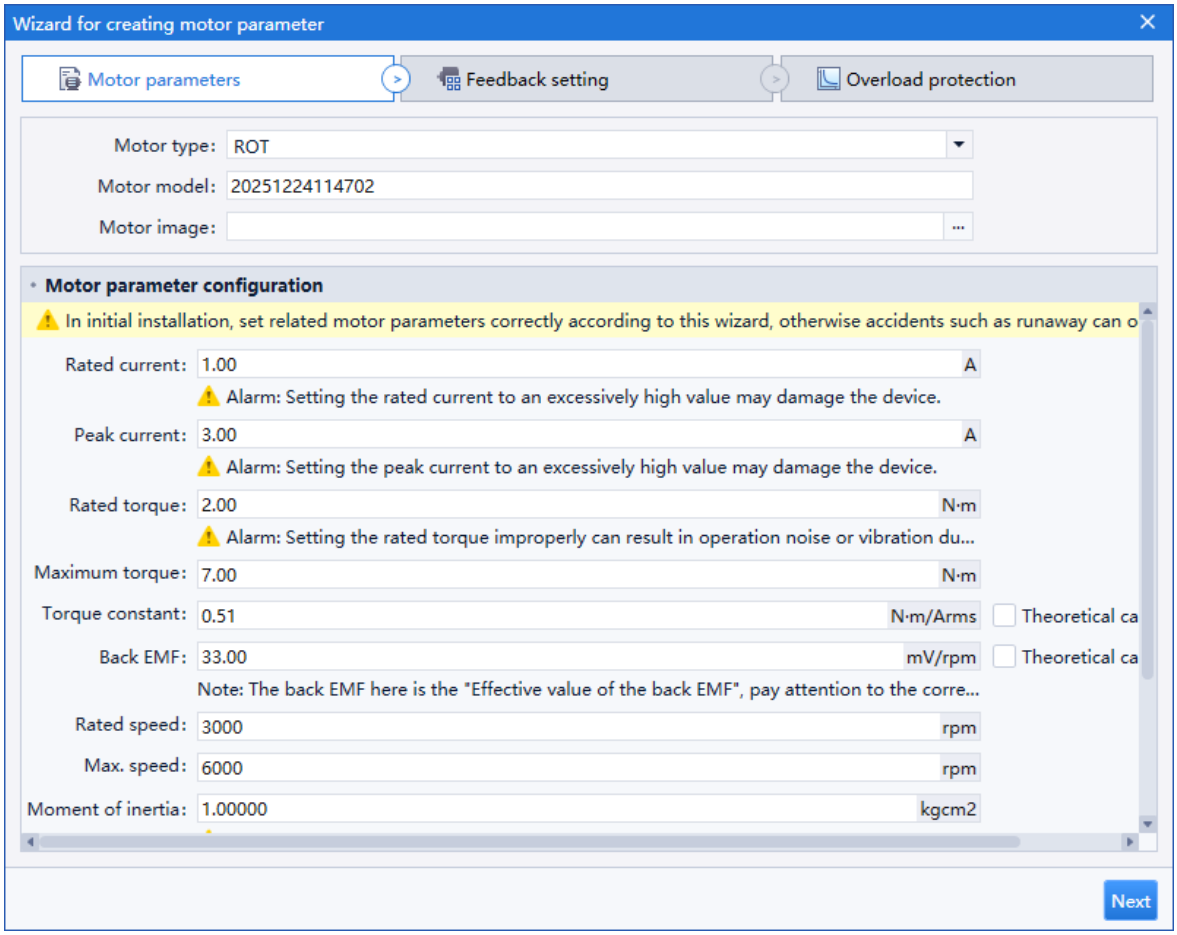
No.	Description		Description
②	Feedback parameter configuration		Include "Encoder feedback signal form" and "Encoder parameter setting". This feature is only available for certain motor models. Click the "Insert feedback parameters" button. The configuration parameters will be synchronized to the "Motor parameters" list on the right.
③	Motor parameters		Configure motor parameters.
④	Function area	New motor	See "New motor".
		Save library files	After selecting the motor parameter file in the main interface, you can adjust the parameters in the parameter list, and then click "Save library files" to add the adjusted motor parameter file to the corresponding motor parameter library. Restrictions: • The motor model name defaults to current motor model + copy and is modifiable. • If the current motor parameter library is UserDDLMotor, UserDDRMotor or UserROTMotor, and there are parameters in the motor parameter list, the button is enabled; otherwise the button is disabled. • Naming rules: ◦ It consists of Latin characters, Chinese characters (including spaces), and must not exceed 50 characters. ◦ It can be composed of letters, numbers, Chinese characters and ~, !, @, #, $, %, ^, &, ( ), _, -, {}. ◦ Single quotes, double quotes, consecutive underscores, ending with an underscore, and starting with double underscores are not allowed. ◦ It cannot start or end with a space. ◦ The following characters are not allowed: \, /, :, *, ?, <, >, |. ◦ Case-sensitive. Half-width and full-width characters are distinguished.
		Delete library files	Deletes the selected motor parameter file from the library. Restrictions: If the current motor parameter library is UserDDLMotor, UserDDRMotor or UserROTMotor, and the current motor model is not empty, the button is enabled; otherwise the button is disabled.

No.	Description		Description
		Read from servo	When the device is connected, click the "Read from servo" button to create a motor parameter file named "yyyyMMddHHmmss" (year, month, day, minute, second) according to the device type, and read the current values of relevant parameters in this motor parameter file from the device. Constraint: Motor parameters and feedback parameters are created based on the current motor type. Their values are read from the device.
		Write to servo	Writes the current parameter values in the current motor parameter file into the servo device. Restrictions: If there are parameters in the motor parameter list, the button is enabled; otherwise, the button is disabled.
		Read the current page	Reads parameter values of the current motor parameter file from the servo device. Restrictions: If there are parameters in the motor parameter list, the button is enabled; otherwise, the button is disabled.

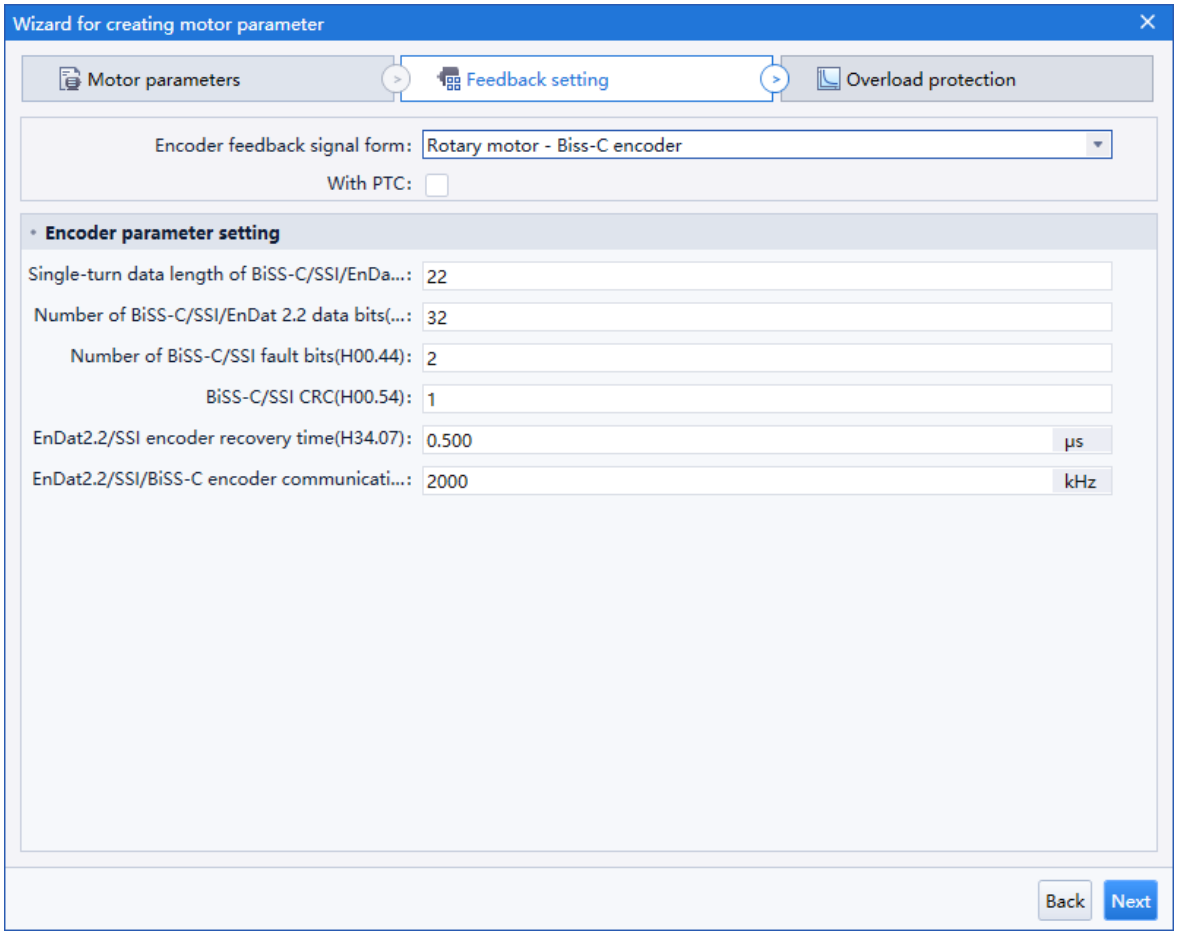
- **New motor**

  Click the "New motor" button in the function area. The "Wizard for creating motor parameter" dialog box pops up. Configure the relevant motor parameters according to the wizard, and save them to the motor parameter library.

1. Configure motor parameters. Set "Motor type", "Motor model", "Motor image", and configure parameters in the "Motor parameter configuration" area.
   In the motor parameter interface, when you select a motor type, the corresponding parameters of the motor type are displayed. Configure the parameters according to the wizard. The supported motor types may vary with different devices.

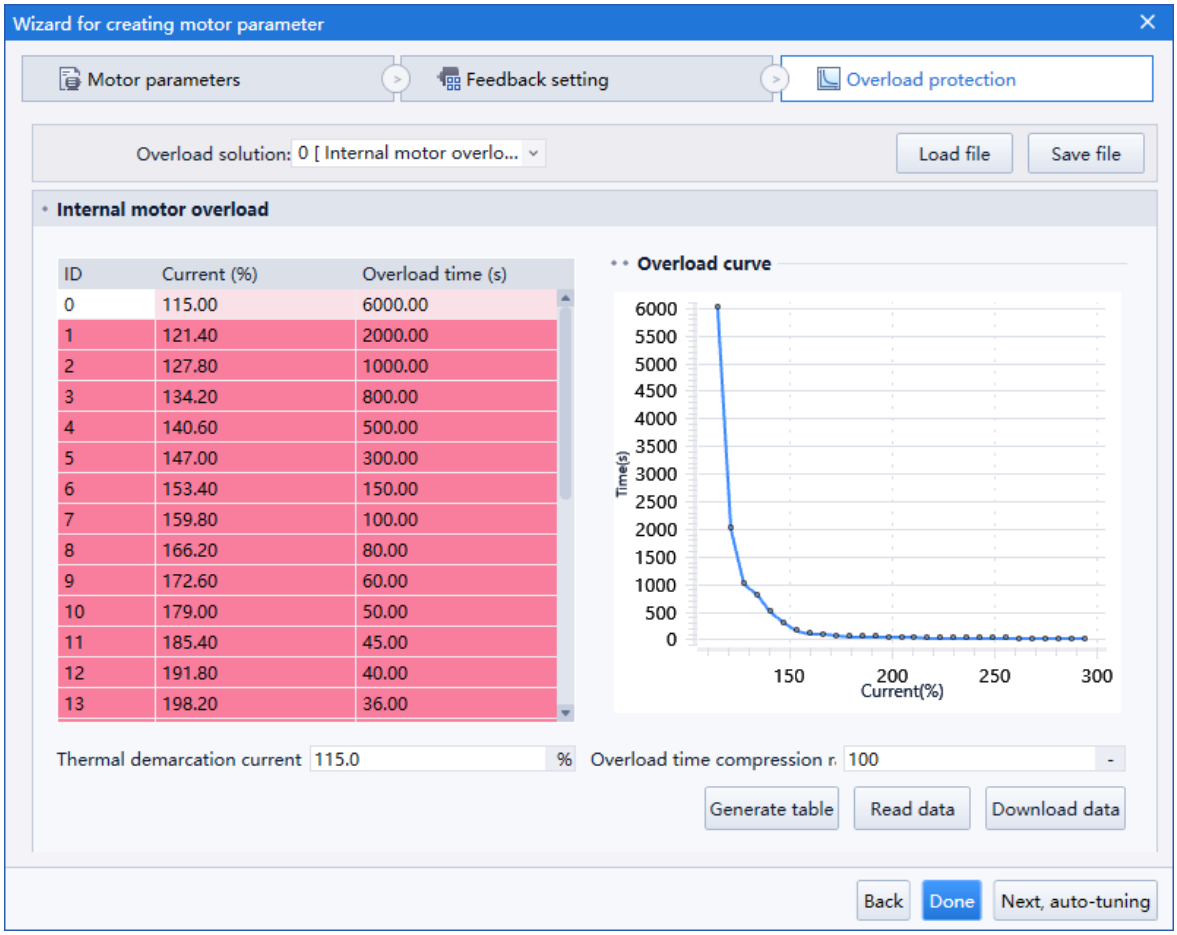


2. Click "Next" to enter the "Feedback setting" interface.
   In the "Feedback setting" interface, the parameter configuration of the encoder is displayed according to the selected encoder feedback signal form. You just need to configure the relevant parameters according to the wizard. The types of encoder feedback signal forms supported by different motor types may differ, and the types of parameters that need to be set among different encoder feedback signal forms also differ.

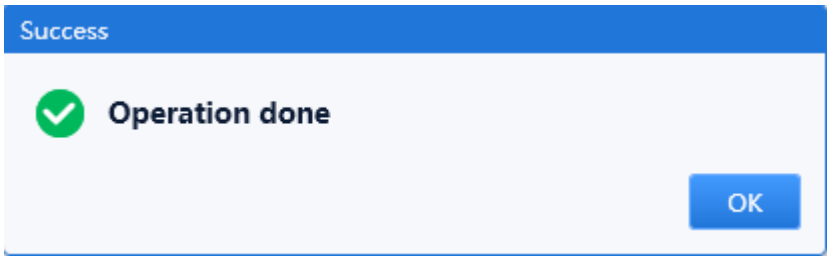


3. Click "Next" to enter the "Overload protection" interface.

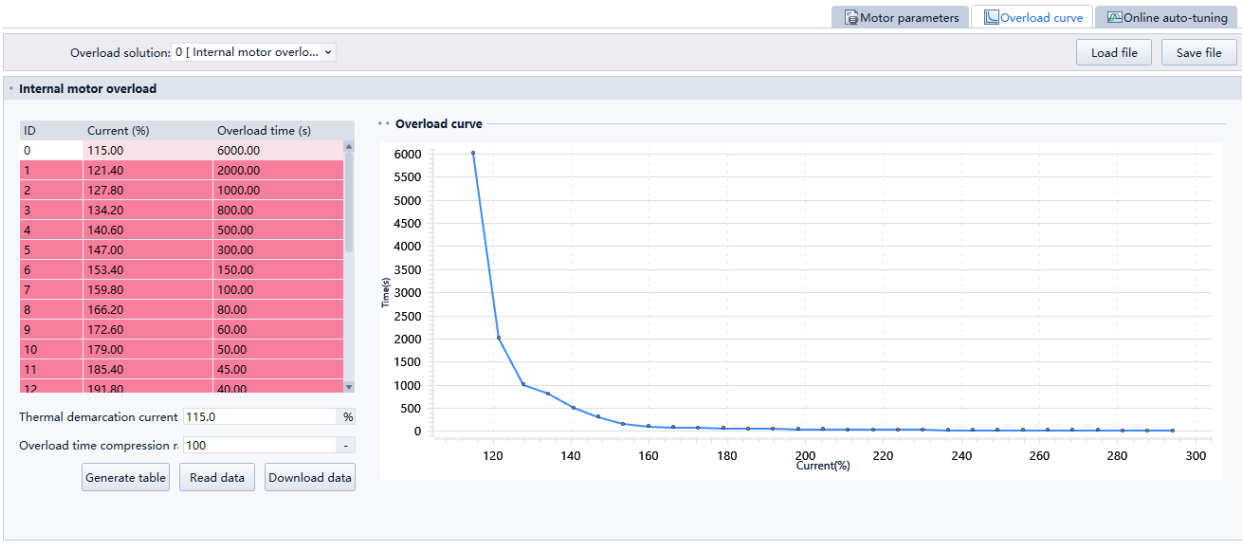
   In the "Overload protection" interface, according to the selected overload protection scheme, the parameter configuration of the corresponding overload protection scheme is displayed. You just need to configure the relevant parameters according to the wizard. After the parameter configuration is completed, you can click the "OK" button to save the created parameters to the motor parameter file library.



4. The "Operation done" dialog box is displayed. You can also click "Next, auto-tuning".



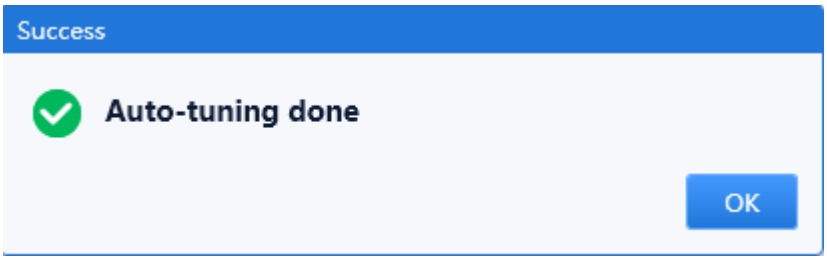
- **Overload curve**



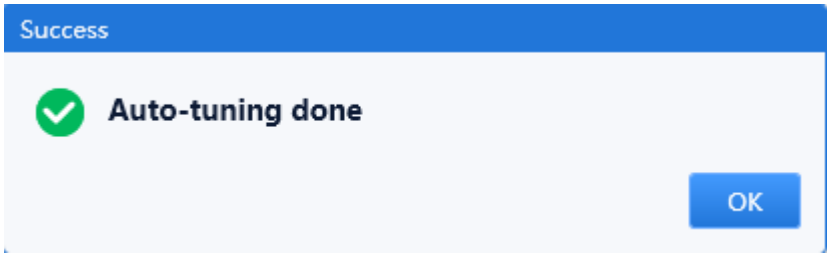
- **Online auto-tuning**

Note: Online auto-tuning is unavailable for the voice coil motor.

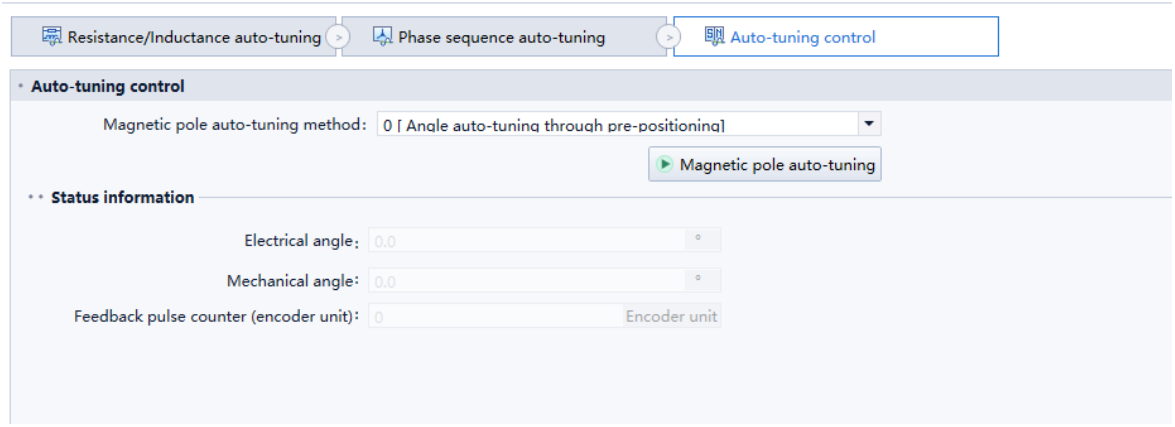
1. Click "Online auto-tuning" to enter the wizard interface. The "Resistance/Inductance auto-tuning" interface is displayed. Click the "Resistance/Inductance auto-tuning" button. After the motor auto-tuning is successful, a dialog box indicating successful auto-tuning appears.



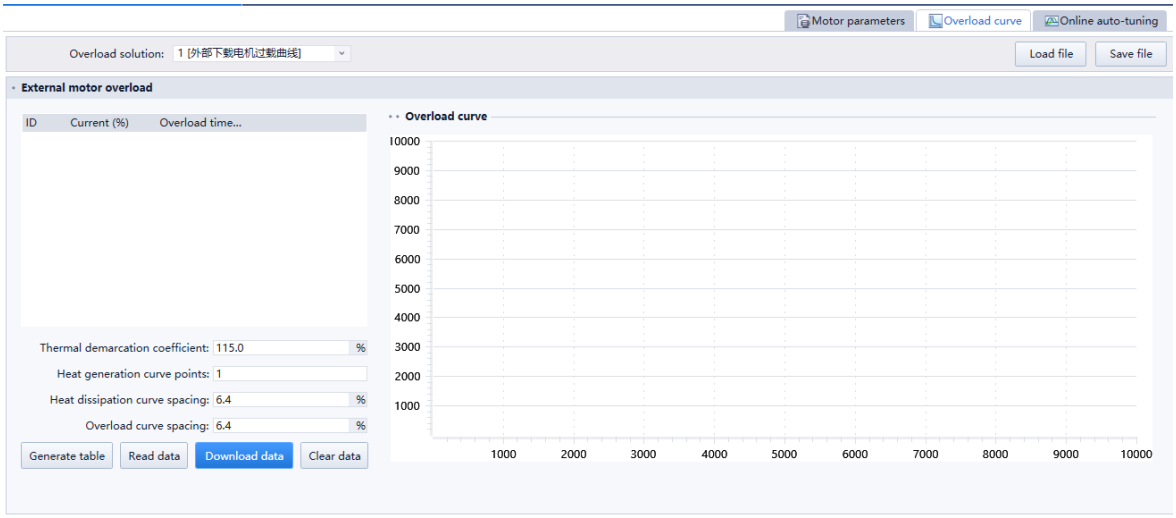
2. Click "Next" to enter the "Phase sequence auto-tuning" interface. Click the "UVW phase sequence auto-tuning" button. After the auto-tuning is successful, a dialog box indicating successful auto-tuning and effective after power-off appears.



3. Click "Next" to enter the "Auto-tuning control" interface.
   Note: Hall auto-tuning is required at initial installation if the device is equipped with one. The interface is as follows.



4. Click "Hall auto-tuning" to enter the interface. During the auto-tuning process, you can view that the status information data changes. If a dialog box indicating auto-tuning failure appears, click "OK" and retry until the auto-tuning is successful.



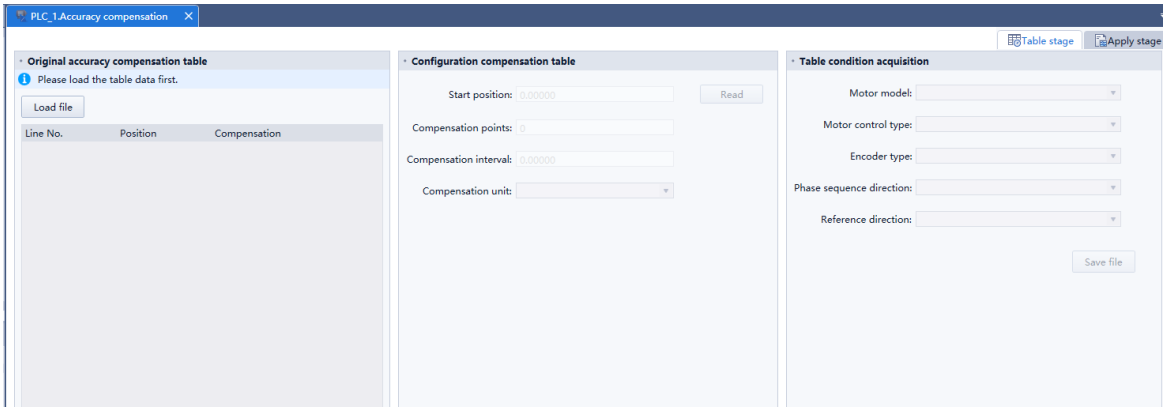
17.3.24 Accuracy Compensation

This function is available only for the SV680N-INT, SV670N-INT, and SV680LN series.

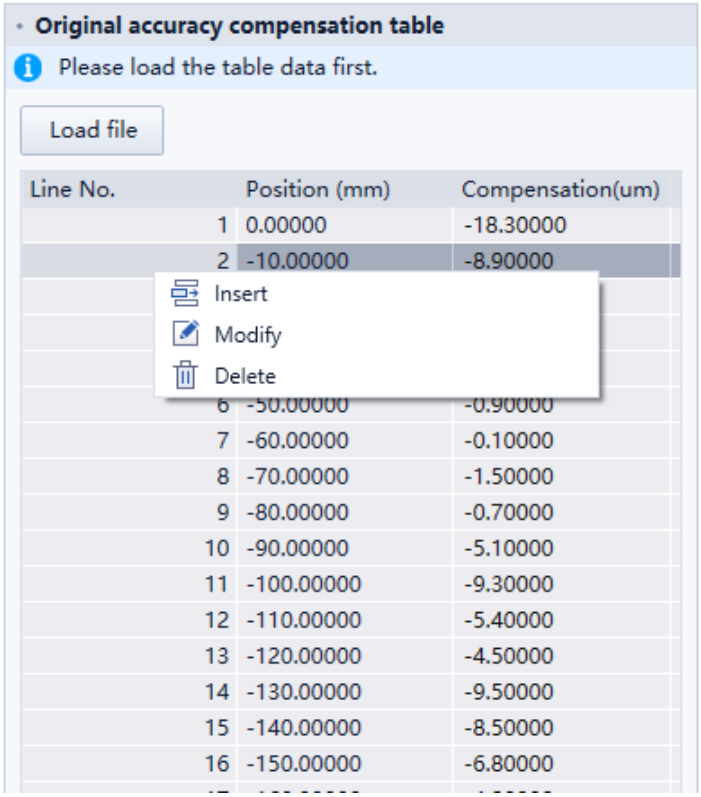
Select a servo device in the project tree, and double-click "Accuracy compensation" in the detailed view to open the interface. The interface contains the "Table stage" and "Apply stage" tabs.

- **Table stage**

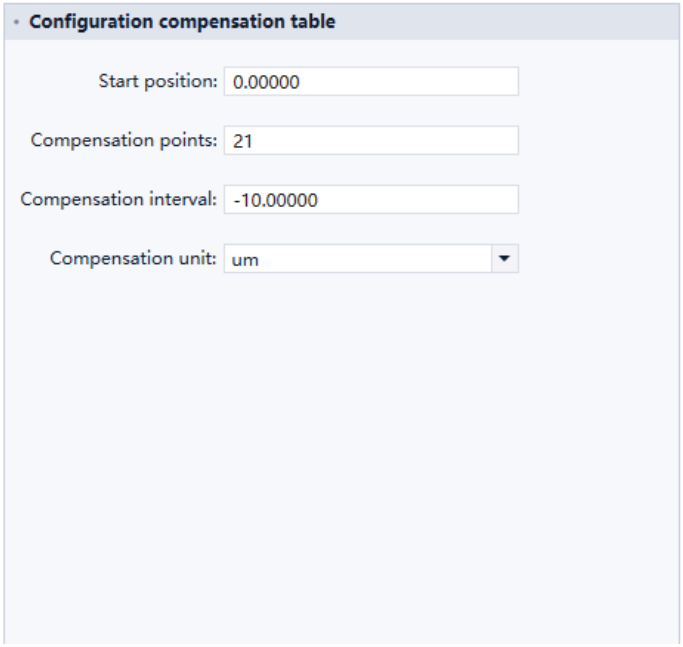
  The table stage of accuracy compensation mainly consists of three steps: loading the original accuracy compensation table, configuring the compensation table and obtaining the table conditions. The interface is as follows.



1. Load the original accuracy compensation table.
   Load the file. Only RTL and RTA files are currently supported. After selecting the corresponding file, all point information will be displayed in the table. You can right-click the data and select "Insert", "Modify", or "Delete" to edit the data.

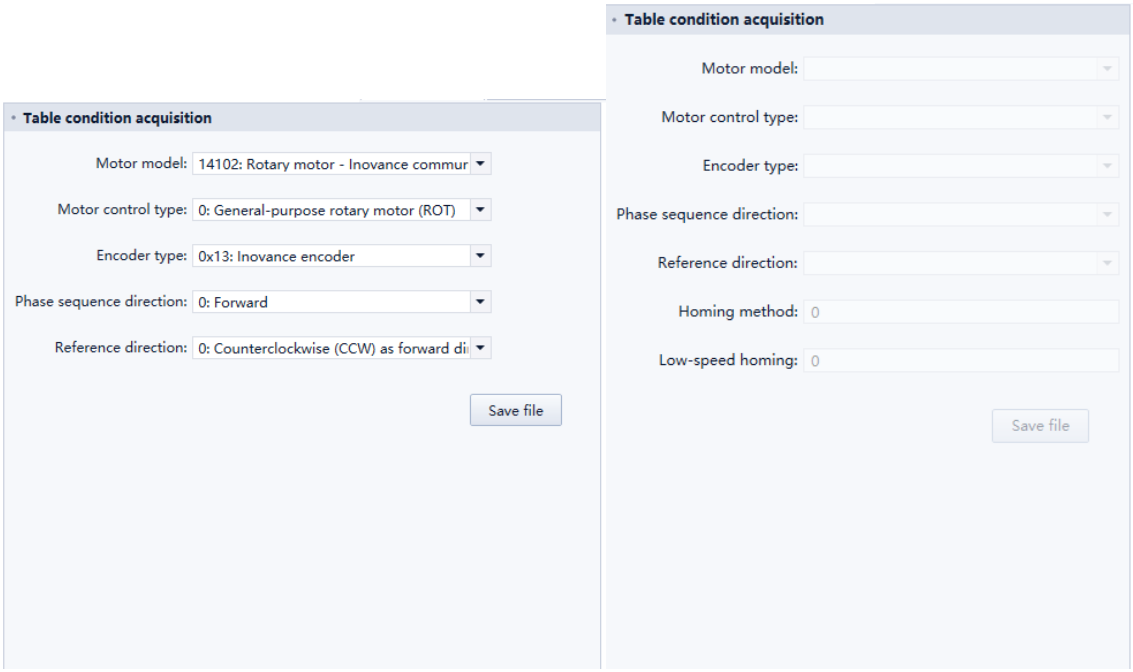


2. Configure the compensation table.
   After the data is successfully imported, relevant configuration information will be automatically generated based on the table data. It mainly includes start position, number of compensation points, compensation interval and compensation unit. The compensation unit is determined by the current motor control type (H00.40). When the value of H00.40 is 0 or 1, the unit is arcsec/mrad; otherwise, the unit is um/mm. When the value of encoder selection (H00.30) is 0x10, 0x12, 0x13, or 0x17, a "Read" button will be displayed below "Start position". Clicking the "Read" button will display the single turn position feedback value of the absolute encoder (H0B.71) in the text box.



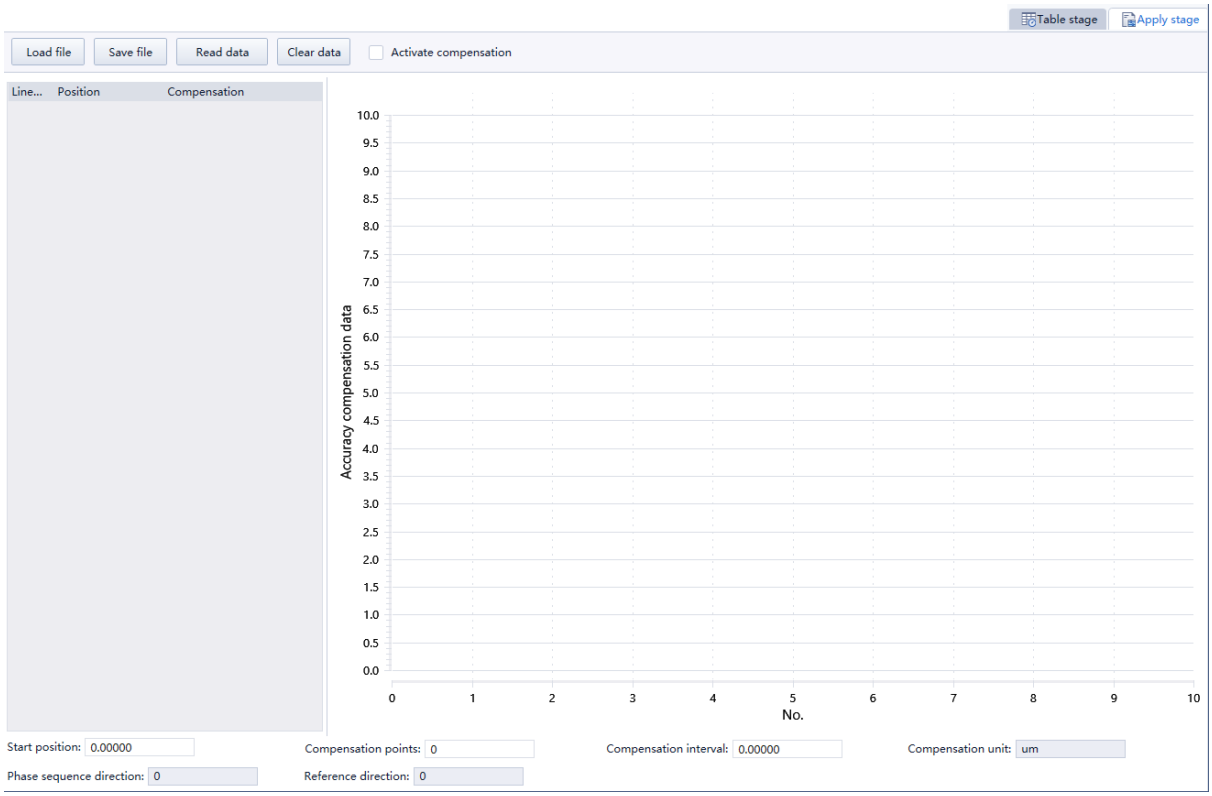
3. Obtain the table conditions. This step includes configurations of the following: motor model, motor control type, encoder type, phase sequence direction, and reference direction.
   After data is imported and the device is online, the relevant drive parameters are read and the parameter values in "Table condition acquisition" are updated. Default values are remains when the device is offline.

Read the value of the encoder selection (H00.30). If the value equals 0x10, 0x12, 0x13, or 9x17, "Homing method" and "Low-speed homing" are not displayed.
If the device is online, and the encoder type (H00.30) is a specific model, "Homing method" and "Low-speed homing" are displayed.



4. Click the "Save file" button. Save the preprocessed accuracy compensation data in a CSV file. The table stage is completed.

- **Apply stage**
  In this phase, the main functions include loading files, saving files, reading compensation data, clearing data, activating compensation, and forced homing. The function interface is as follows.



See the following table for function descriptions.

Description	Description
Load file	◦ Function: Imports the local accuracy compensation data file. After importing, the corresponding data will be displayed in the table, and the corresponding content will be displayed in the corresponding text box. ◦ Restrictions: The imported file must be of CSV format. If it is an absolute encoder, homing mode and homing low-speed text boxes will not be displayed.
Save file	◦ Function: Saves the data on the current interface locally. ◦ Restrictions: The data is saved to a CSV file.
Read data	◦ Function: Obtains the accuracy compensation data stored in the drive. The read data is updated into the table. ◦ Restrictions: The device must be online. When the device is of the absolute encoder type, the read start position is always 0.
Activate compensation	◦ Function: Downloads the current data to the drive and starts accuracy compensation. ◦ Restrictions: The device must be online. Data must be available on the current interface.
Clear data	◦ Function: Clears the accuracy compensation data saved in the drive. ◦ Restrictions: The device must be online.
Forced homing	◦ Function: Performs forced homing according to the homing configuration in the table stage. ◦ Restrictions: The device must be online.
Homing state	◦ Function: Displays the current homing status.
Table	◦ Function: Draws a line chart of the data in the table on the left.

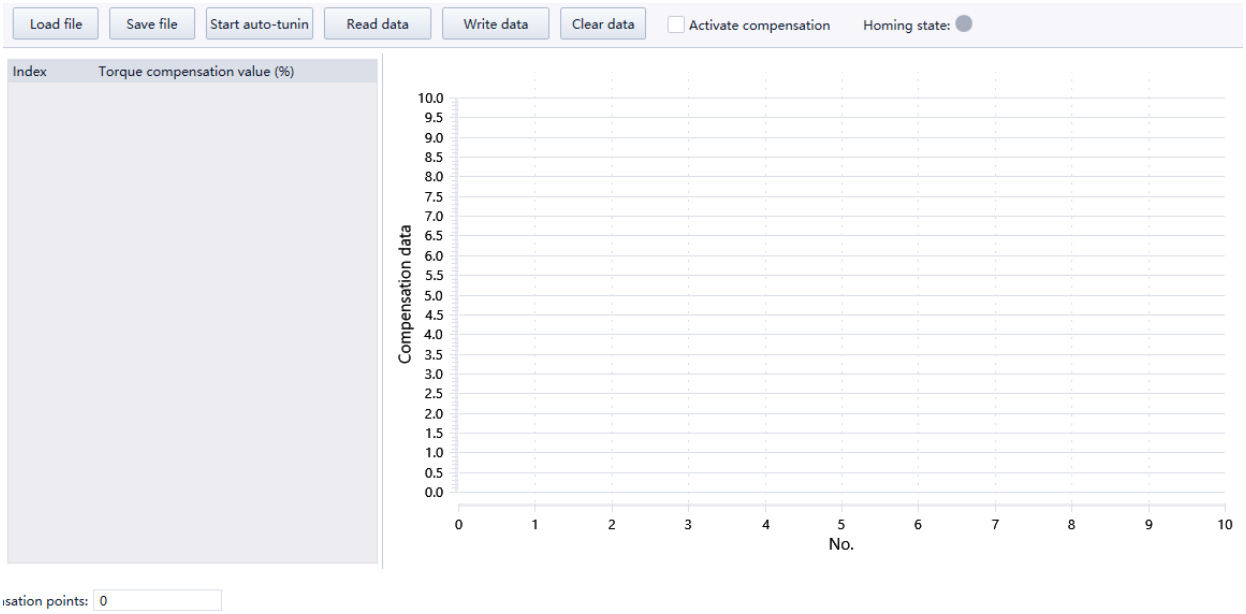
During file loading, if the file is incorrect, you will be prompted with the following message: The loaded file and the used motor do not match. Check the file first.

The motor model and encoder model are displayed according to the loaded file. When the device is offline and the file is loaded, the motor model and encoder model are also displayed according to the loaded file.

17.3.25 Cogging Compensation

This function is available only for the SV680N-INT, SV670N-INT, and SV680LN series.

Select a servo device in the project tree, and double-click "Cogging compensation" in the detailed view to open the interface. The main functions include loading file, saving file, starting cogging auto-tuning, reading compensation data, writing compensation data, activating compensation, and clearing data. The interface is as follows.



See the following table for function descriptions.

Description	Description
Load file	• Function: Imports the local cogging fluctuation compensation data file. After import, the data will be displayed in the table, and the content will be displayed in the corresponding text box. • Restrictions: The import file format is CSV.
Save file	• Function: Saves the data on the current interface locally. • Restrictions: The file is saved to the CSV format. If it is an absolute encoder, homing mode and homing low-speed text boxes will not be displayed.
Read data	• Function: Obtains the accuracy compensation data stored in the drive. The read data is updated into the table. • Restrictions: The device must be online.
Write data	• Function: Writes the compensation data on the current interface into the drive. • Restrictions: The device must be online. Data must be available on the current interface.
Activate compensation	• Function: Enables cogging fluctuation compensation. • Restrictions: The device must be online.
Clear Data	• Function: Clears the accuracy compensation data saved in the drive. • Restrictions: The device must be online.

Description	Description
Homing state	• Function: Displays the current homing status. • Restrictions: This control is displayed only when the current device is not an absolute encoder type.
Table	• Function: Draws a line chart of the data in the table on the left. You can insert, modify and delete data from the table with the context menu. After the data is changed, the curve on the right side is refreshed synchronously.
Start auto-tuning	• Function: Enables cogging torque fluctuation auto-tuning. After auto-tuning, the current drive compensation data is read automatically. • Restrictions: After cogging auto-tuning is enabled, the device will enter the auto-tuning state and be enabled automatically. At this time, all buttons on the interface are unavailable. When auto-tuning is completed or fails, all button states will become available.
Compensation data count	• Restrictions: For 16-bit data types, the maximum data count is 1024. For 32-bit data types, the maximum data count is 2048.

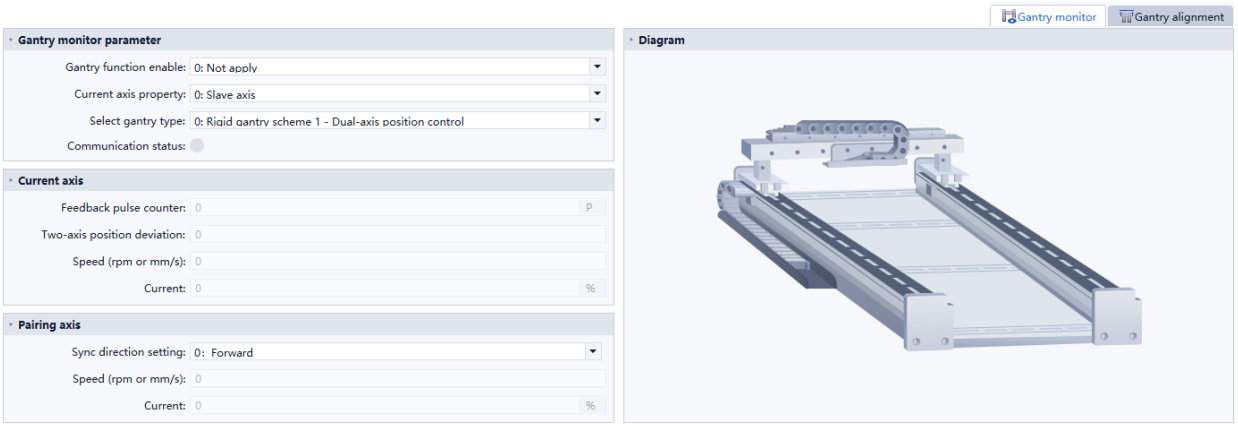
17.3.26 Gantry Synchronization

This function is available only for SV680N-INT and SV680LN series.

Select a servo device in the project, and double-click "Gantry synchronization" in the detailed view to open the interface. Gantry synchronization mainly includes gantry monitoring and gantry alignment.

- **Gantry monitoring**

The interface is as follows.



Description	Description	Restriction
Gantry function enable	1. The corresponding address is 0x1402. 2. Initializes the drop-down box and displays the current value. 3. Switches the option to write the value.	You cannot write values in running state.

Description	Description	Restriction
Current axis property	1. The corresponding address is 0x1403. 2. Initializes the drop-down box and displays the current value. 3. Switches the option to write the value.	You cannot write values in running state.
Select gantry type	1. The corresponding address is 0x1404. 2. Initializes the drop-down box and displays the current value. 3. Switches the option to write the value. 4. If a rigid gantry is selected, the gantry alignment tab cannot be configured.	You cannot write values in running state.
Feedback pulse counter	Address: 0x0B11 + axis offset	read-only
Two-axis position deviation	Address: 0x1423 + axis offset	read-only
Speed (current axis)	Address: 0x1428 + axis offset	read-only
Current (current axis)	Address: 0x1426 + axis offset	read-only
Sync direction setting	1. Address: 0x1405 + axis offset 2. Initializes the drop-down box and displays the current value. 3. Switches the option to write the value.	You cannot write values in running state.
Speed (paired axis)	Address: 0x1429 + axis offset	read-only
Current (paired axis)	Address: 0x1427 + axis offset	read-only
Diagram	Displays the scene of gantry synchronization in a three-dimensional diagram.	-
Communication status	1. The status mark is gray by default. 2. Address: 0x1400 + axis offset 3. Cyclic transmit and read. The interval is 300 ms in the early software. 4. If the difference between the value and 1 is 1e-6 (0.000001), the status mark is green; otherwise, it is gray.	-

- **Gantry alignment**

The interface is as follows.

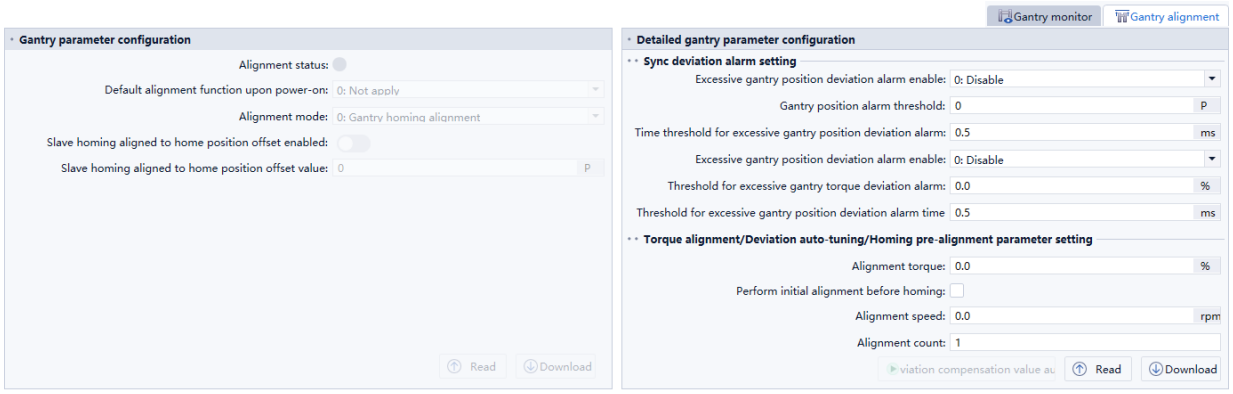


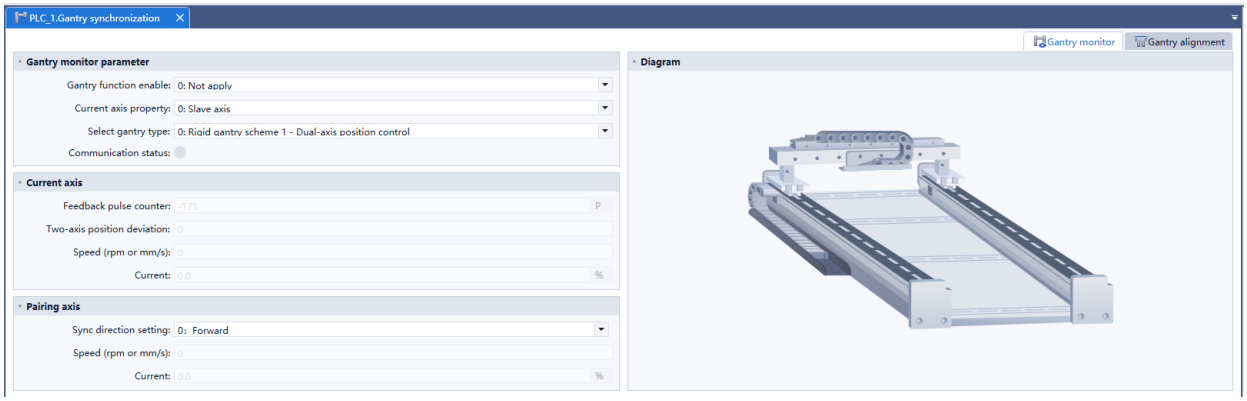
Table17-3-26-1

Description	Description	Restriction
Alignment status	Displays alignment status. Address: 0x1401 + axis offset; cyclic transmit and read; interval: 300 ms	-
Default alignment function upon power-on	You can select the alignment mode upon power-on from the drop-down list. The options include "0: No apply" and "1: Aligned by default upon initial power-on of two axes". The parameter address is 0x1414.	Restrictions: You cannot write values in running state.
Alignment mode	You can select the alignment mode from the drop-down list. Options include "0: Gantry homing alignment", "2: Gantry torque alignment", "3: Gantry enable alignment", and "4: Gantry DI alignment". The parameter address is 0x1415.	Restrictions: Value cannot be written while running
Slave homing aligned to home position offset enabled	Address: 0x1447 + axis offset	-
Slave homing aligned to home position offset value	Displays the value of address 0x1448 + axis offset.	-
Read	Click "Read" to read the parameter values corresponding to alignment status, power-on alignment, alignment mode, offset enable and offset value, and refresh the control display.	Restrictions: The function is unavailable when the device is offline.

Description	Description	Restriction
Download	Click "Download" to write parameter values corresponding to alignment state, power-on alignment, alignment mode, offset enable, and offset value.	Restrictions: The function is unavailable when the device is offline. You cannot write values in running state.

- **Pairing axis parameter configuration**

The axis pairing parameter configuration function is available only for SV680LN series.

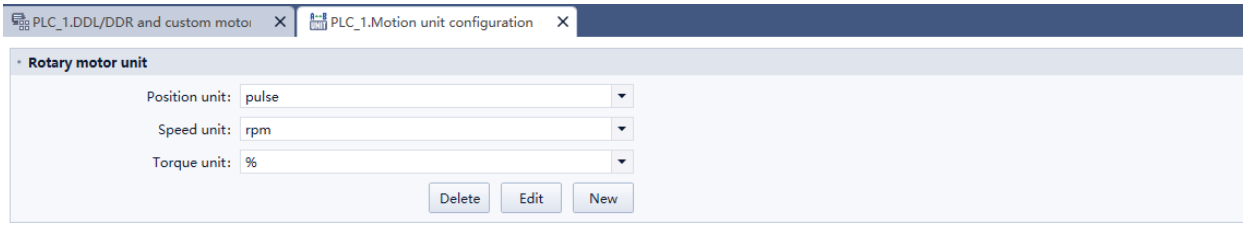


Function	Description
Pairing axis parameter address	• Address: 0x140F + axis offset • Initializes the drop-down box and displays the current value. • Switches the option to write the value.
Pairing axis parameter data	• Address: 0x1410 + axis offset • Reads the address value and initializes the control and range.
Used to enable pairing axis parameters.	• Address: 0x1411 + axis offset • Initializes the drop-down box and displays the current value.
Download	• Function: Writes the value of the axis pairing parameter address, the axis pairing parameter data, and the axis pairing parameter enable setting. • Restrictions: You cannot write values when the device is offline.

17.3.27 Motion Unit Configuration

This function is available only for the SV680N-INT, SV670N-INT, and SV680LN series.

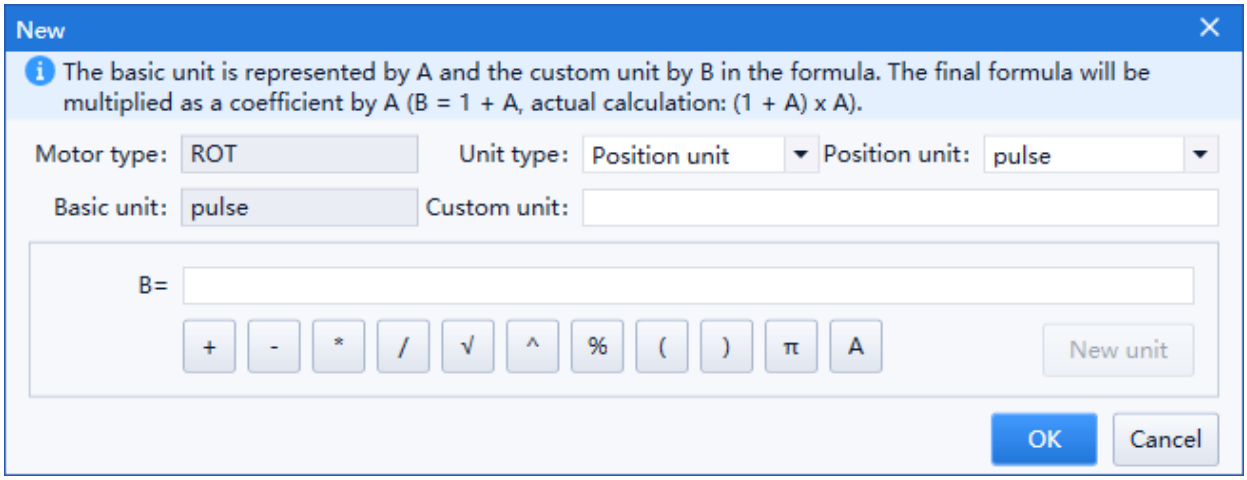
Select a servo drive in the project tree, and double-click "Motion unit configuration" in the detailed view to open the interface. The interface is as follows.



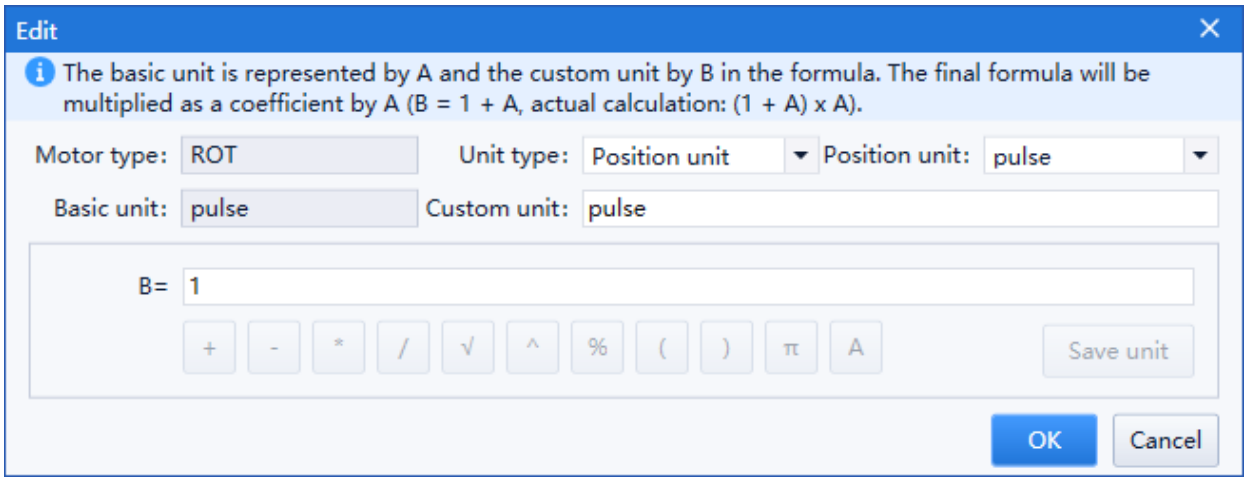
Description	Description
Position unit	• Function: You can select the position unit from the drop-down box. Options for rotary motors include "r", "rad", and "pulse". Options for linear motors include "mm", "um", "pulse", and "pitch". • Restrictions: ◦ If a linear motor (H0040 = 2 or 3) is used, the unit configuration of the linear motor is displayed. If a rotary motor is used (H0040 = 0 or 1), the unit configuration of the rotary motor is displayed. ◦ Linear motor switching coefficient: pulse -> mm = 2 * H0048/H0031, pulse -> um = 2000 * H0048/H0031, pulse -> pitch = 1/H0031. Rotary motor switching coefficient: pulse -> rad = 2 * pi/H0031, pulse -> r = 1/H0031. ◦ After switching units, each module takes effect synchronously.
Speed unit	• Function: You can select the position unit from the drop-down box. Options for rotary motors include "rpm", "r/s", and "rad/s". Options for linear motors include "mm/s" and "rad/s". • Restrictions: ◦ If a linear motor (H0040 = 2 or 3) is used, the unit configuration of the linear motor is displayed. If a rotary motor is used (H0040 = 0 or 1), the unit configuration of the rotary motor is displayed. ◦ Linear motor conversion coefficient: mm/s -> um/s = original value * 1000; mm/s -> pulse/s = original value * H0031 / (2 * H0048); mm/s -> rad/s = original value * pi / H0048. Rotary motor conversion coefficient: rpm -> r/s = 1/60; rpm -> rad/s = 2 * pi / 60. ◦ After switching units, each module takes effect synchronously.
Torque unit	• Function: Configure position units via a drop-down list. For rotary motors, the drop-down options include %; for linear motors, the drop-down options include %. • Restrictions: If a linear motor (H0040 = 2 or 3) is used, the unit configuration of the linear motor is displayed. If a rotary motor is used (H0040 = 0 or 1), the unit configuration of the rotary motor is displayed.
Custom unit	• Function: Supports creating, modifying, and deleting custom units and their corresponding conversion formulas. • Restrictions: /

Click the "New" or "Edit" button to enter the following editing interface.

- New



- Edit



17.3.28 Motion Trajectory and Simulation

This function is currently not supported for the IS650N, SVD630ND, SV660ND, SV680LN, STD60N, and SV670LN series products.

Select a servo drive in the project tree, and double-click "Motion trajectory and simulation" in the detailed view to open the interface.

The motion trajectory and simulation function mainly consists of two parts: "Motion trajectory" and "Simulation".

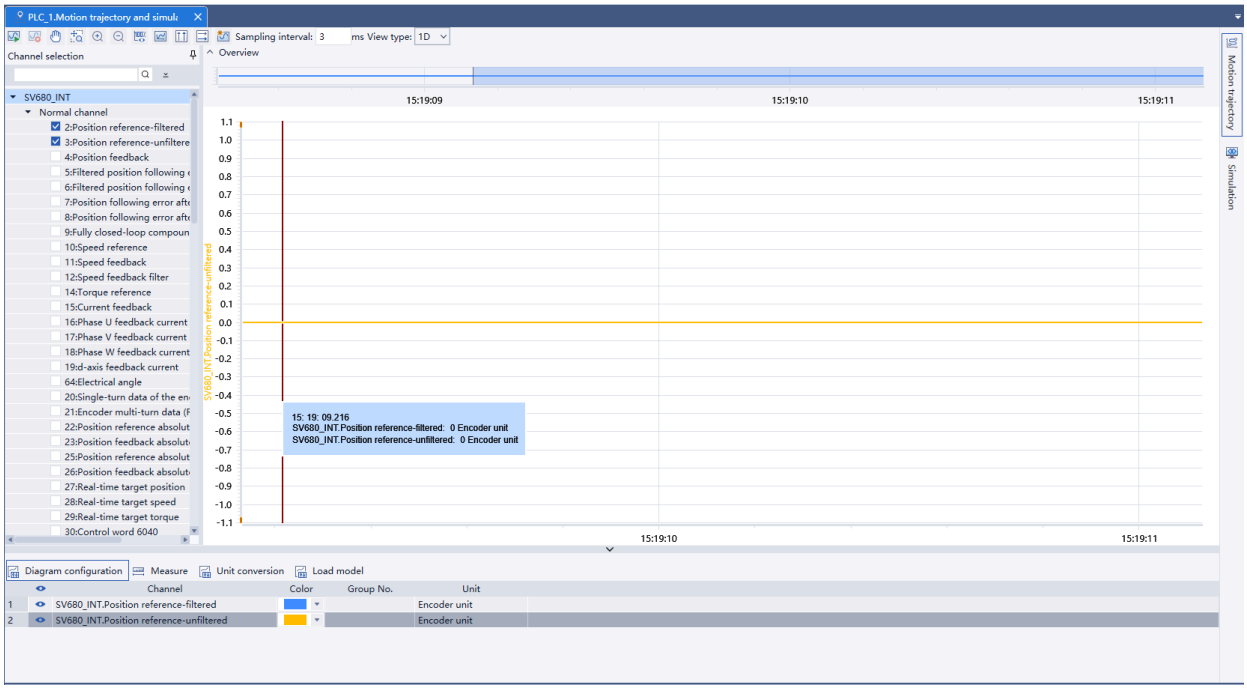
Motion trajectory: Obtains position references and position feedbacks from drive devices via EtherNet-PLC/EoE communication, and plots them as 1D, 2D, 3D, and circular curves. It also provides tools for curve analysis.

Simulation: Supports both file-based and reference-based data sources. It generates simulation curves by incorporating load parameters and drive parameters.

17.3.28.1 Motion Trajectory

- **Function interface**

  Select a servo device in the project tree, and double-click "Motion trajectory and simulation" in the detailed view to open the interface. On the right side, click the "Motion trajectory" tab. The tab consists of three parts: the toolbar, the curve display area, and the tabs for diagram configuration, unit conversion, load model, and measurement, as shown in the figure below.



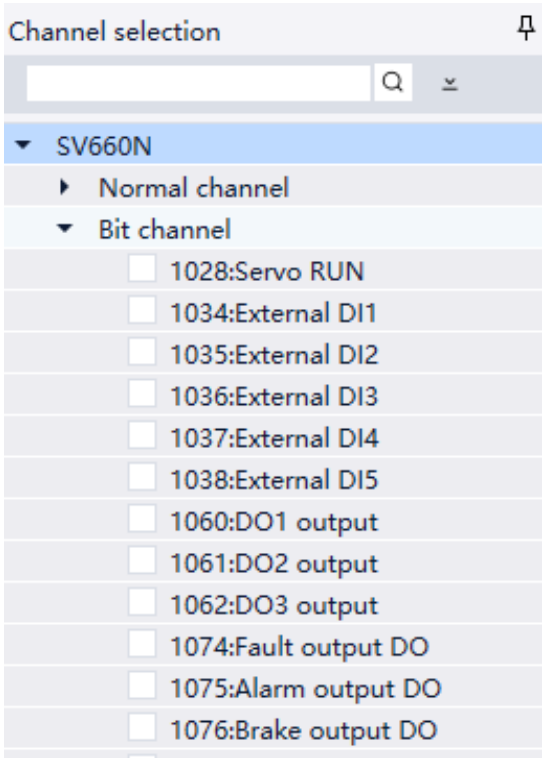
- **Basic operating process**

1. Select a channel from "Channel selection" and configure channel parameters.
2. Set "Sampling interval" and "View type" in the toolbar.
3. Click the "Start" icon in the toolbar to view the curve and perform analysis.

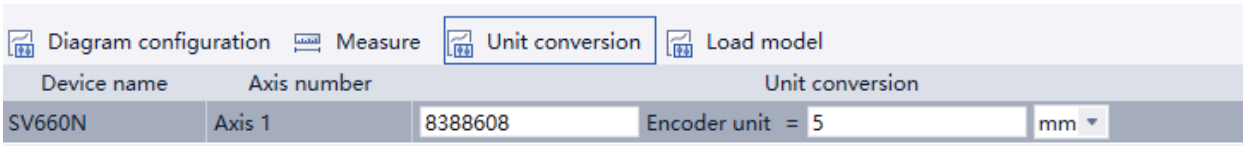
> NOTE
>
> The 1D curve is displayed by default. To view the 2D/3D curve, change the view type to 2D/3D.

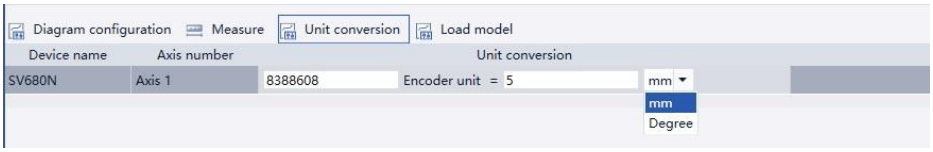
- **Channel selection**

Refer to the following figure for channel selection.

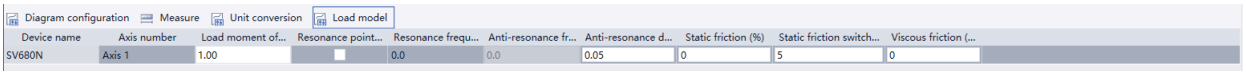


- **Unit conversion**



Function	Description	Restriction
Device name	Displays all device names under the current project.	read-only
Axis number	Displays the axis number of the current device.	read-only
Unit conversion	Used to convert between the encoder unit and the user unit. Options include mm and degree. 	/

- **Load model**



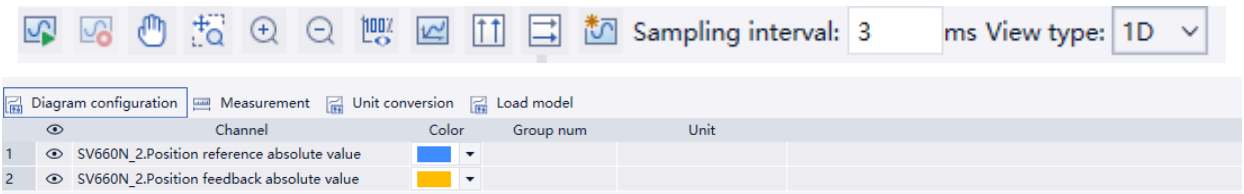
Function	Description	Restriction
Device name	Displays all device names under the current project.	read-only

Function	Description	Restriction
Axis Number	Displays the axis number of the current device.	read-only
Load moment of inertia ratio	It sets the ratio of the load inertia against the rotor (of the motor) inertia.	Range: 0.00 to 120.00
Resonance point exists	Sets whether there is any resonance point.	/
Resonance frequency (Hz)	Defines the model resonance frequency.	read-only
Anti-resonance frequency (Hz)	Defines the model anti-resonance frequency.	• This cell is editable only when the corresponding "Resonance point exists" is checked. • Range: 0.0 to 300.0
Anti-resonance damping coefficient	Defines the model anti-resonance damping.	Range: 0.00 to 1.00
Static friction (%)	Defines the static friction of the model.	Range: 0 to 100
Static friction switchover speed (rpm)	Defines the static friction switchover speed of the model.	Range: 0 to 3000
Viscous friction (%/100)	Defines the viscous friction of the model.	Range: 0 to 100

- **Diagram configuration**

  The view type of the "Diagram configuration" tab is set by the view type in the toolbar, with options 1D, 2D, and 3D. The 1D curve is automatically added after acquired by the channel.

Diagram configuration (1D)



Function	Description	Restriction
Display/hide	Displays the current curve.	/
	Hides the current curve.	/
Channel	Displays channel names.	read-only

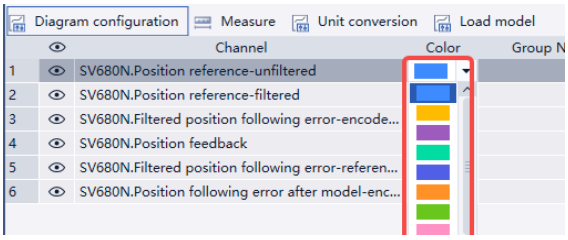
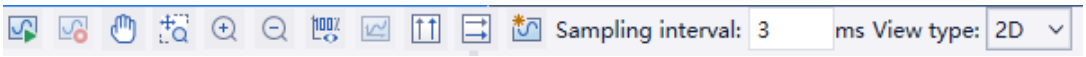
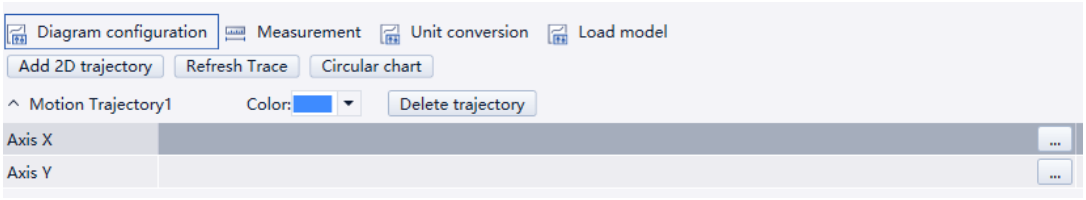
Function	Description	Restriction
Color	Sets the color of the channel curve. 	/
Group	Used to view channels in groups in multi-channel mode.	Channels without a group number are all in a separate group.
Unit	Displays the unit of the current channel.	read-only

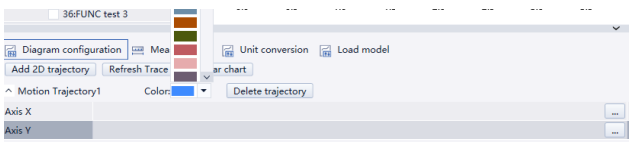
Diagram configuration (2D)

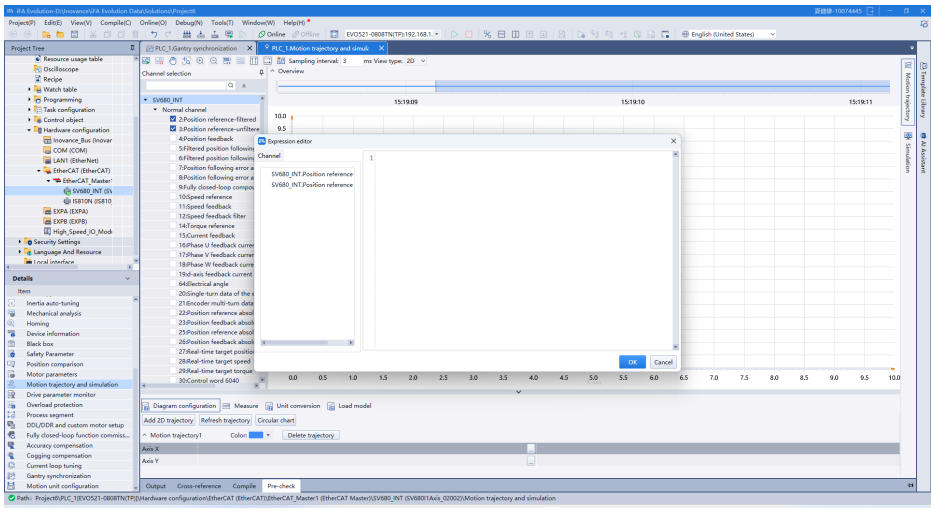


When the view type is 2D, you can view the 2D curve or circular curve on the diagram configuration tab.

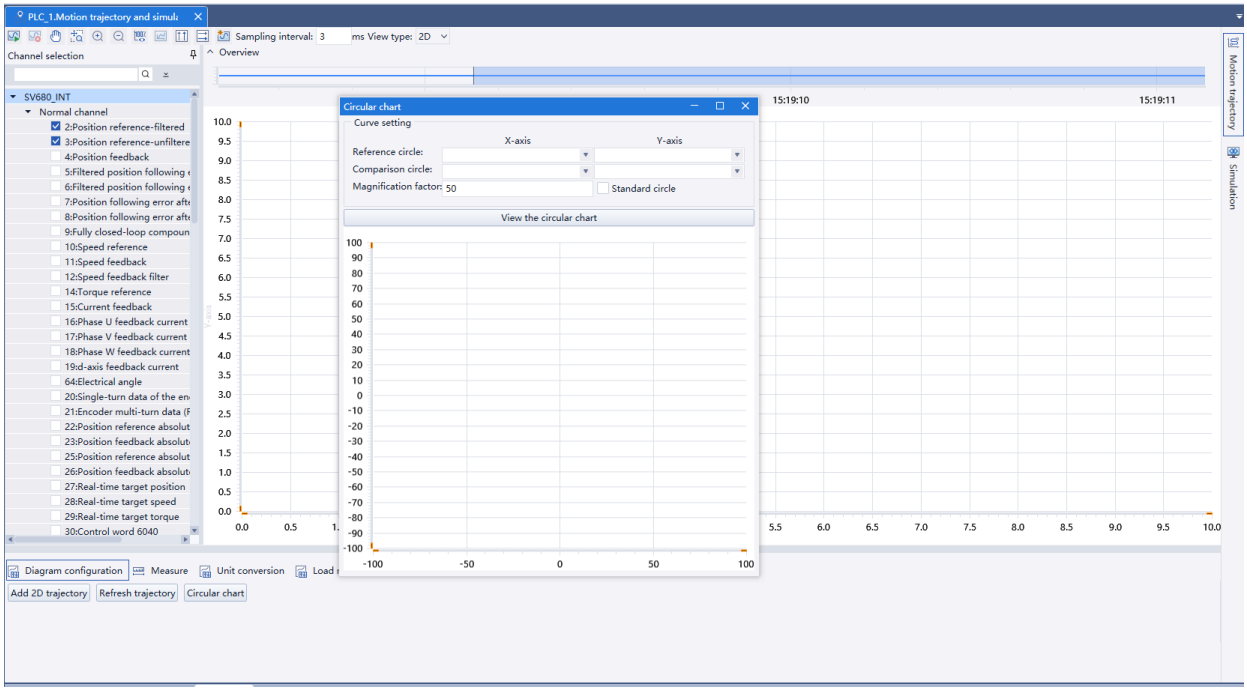
The setting of the 2D curve is as follows.



Function	Description	Restriction
Add 2D trajectory	Adds a new 2D trajectory.	/
Refresh trajectory	Refreshes the current trajectory.	/
Trajectory name	Name of the current trajectory. It is automatically generated and cannot be changed, such as motion trajectory 1.	read-only
Color	Sets the color of the channel curve. 	/
Delete trajectory	Deletes the current trajectory.	/

Function	Description	Restriction
Axis X	X-axis selection: Selects channels by dragging to edit the formula. 	/
Axis Y	Y-axis selection: Selects channels by dragging to edit the formula. For the schematic diagram, see the X-axis setting.	/

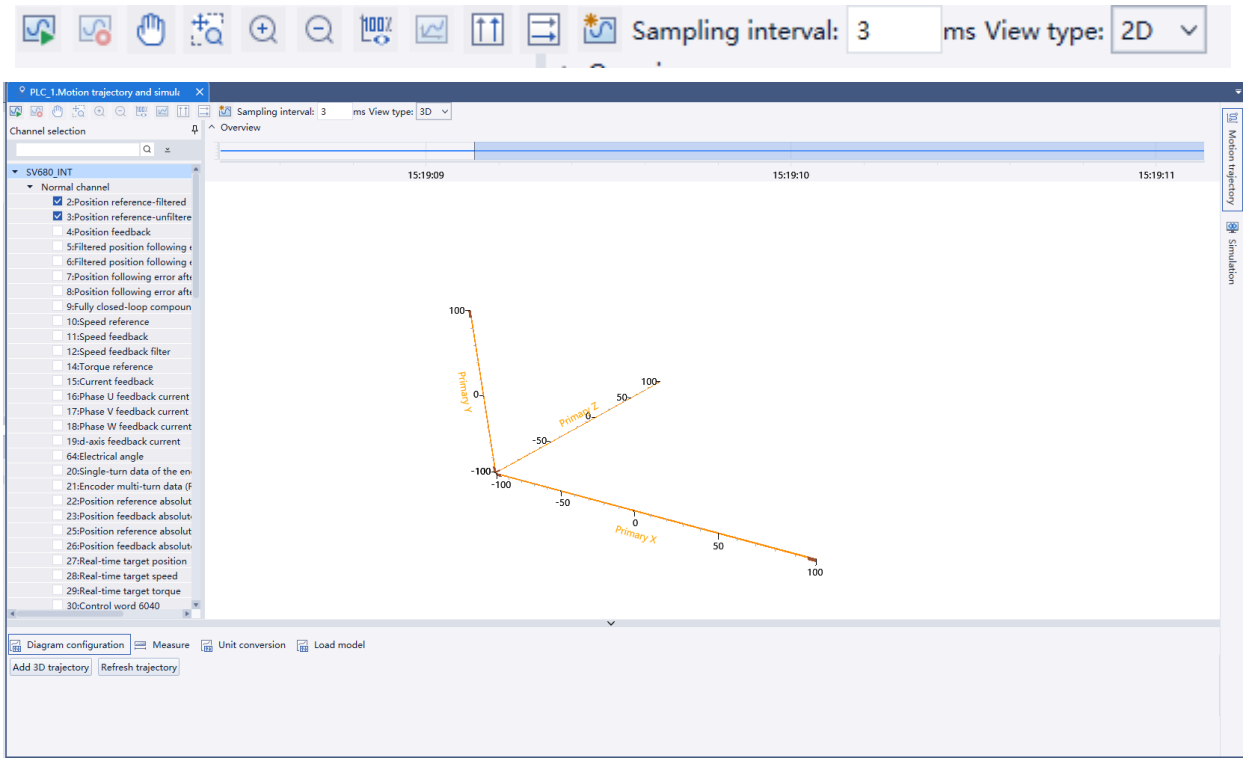
The setting of the circular chart is as follows.

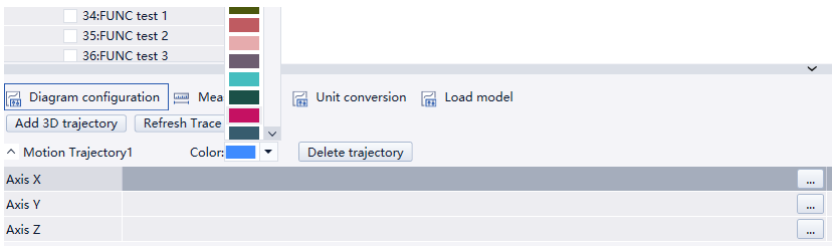


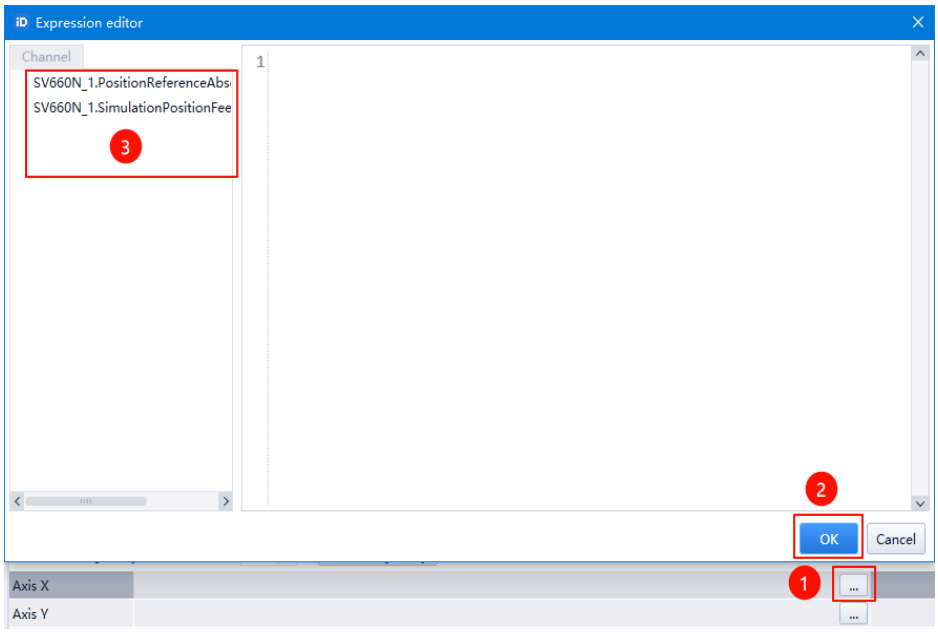
Function		Description	Restriction
Curve setting	Reference circle	Sets the reference circle.	/
	Comparison circle	Used to analyze the error relative to the reference circle.	/
	Magnification factor	Magnification of the error between the comparison circle and reference circle.	/

Function		Description	Restriction
	Standard circle	Sets whether the reference circle is used as the standard circle.	/
View the circular chart		Click to generate the circular chart.	/

Diagram configuration (3D)



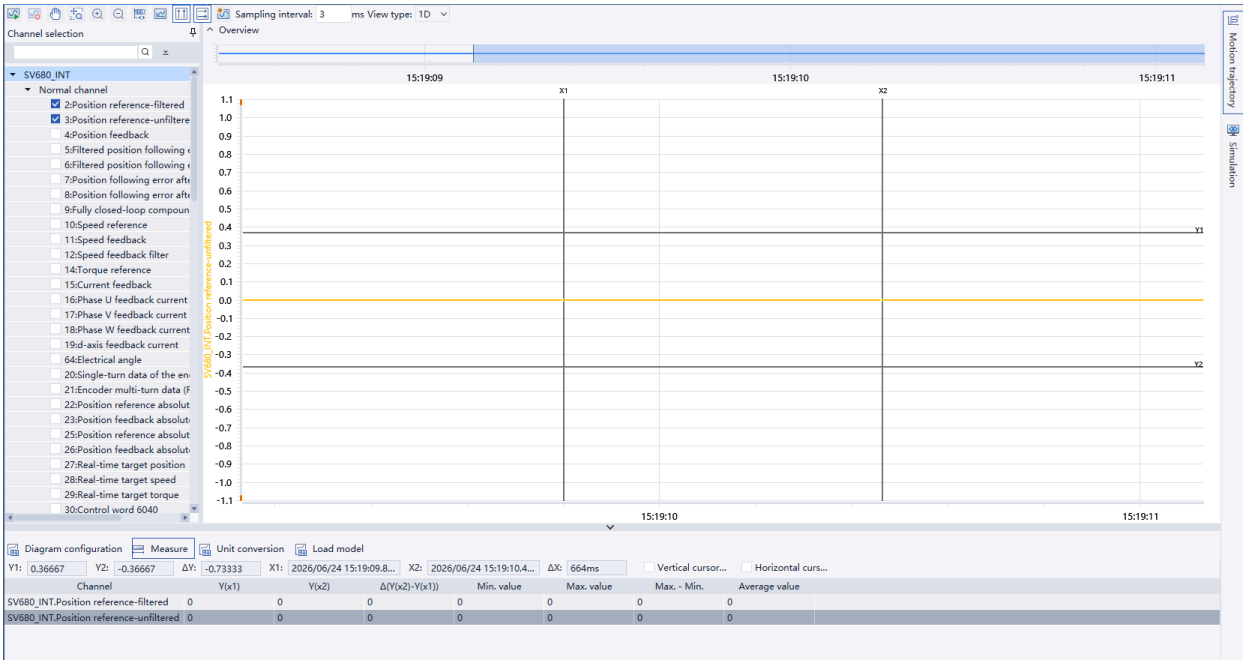
Function	Description	Restriction
Add 3D trajectory	Adds a new 3D trajectory.	/
Refresh trajectory	Refreshes the current trajectory.	/
Trajectory name	Name of the current trajectory. It is automatically generated and cannot be changed, such as motion trajectory 1.	read-only
Color	Sets the color of the channel curve. 	/
Delete trajectory	Deletes the current trajectory.	/

Function	Description	Restriction
Axis X	X-axis selection: Selects channels by dragging the mouse to edit the formula.  Figure 17-3-28-1-1 X-axis settings	/
Axis Y	Y-axis selection: Selects channels by dragging to edit the formula. For the schematic diagram, see the X-axis setting.	/
Axis Z	Z-axis selection: Selects channels by dragging to edit the formula. For the schematic diagram, see X-axis setting.	/

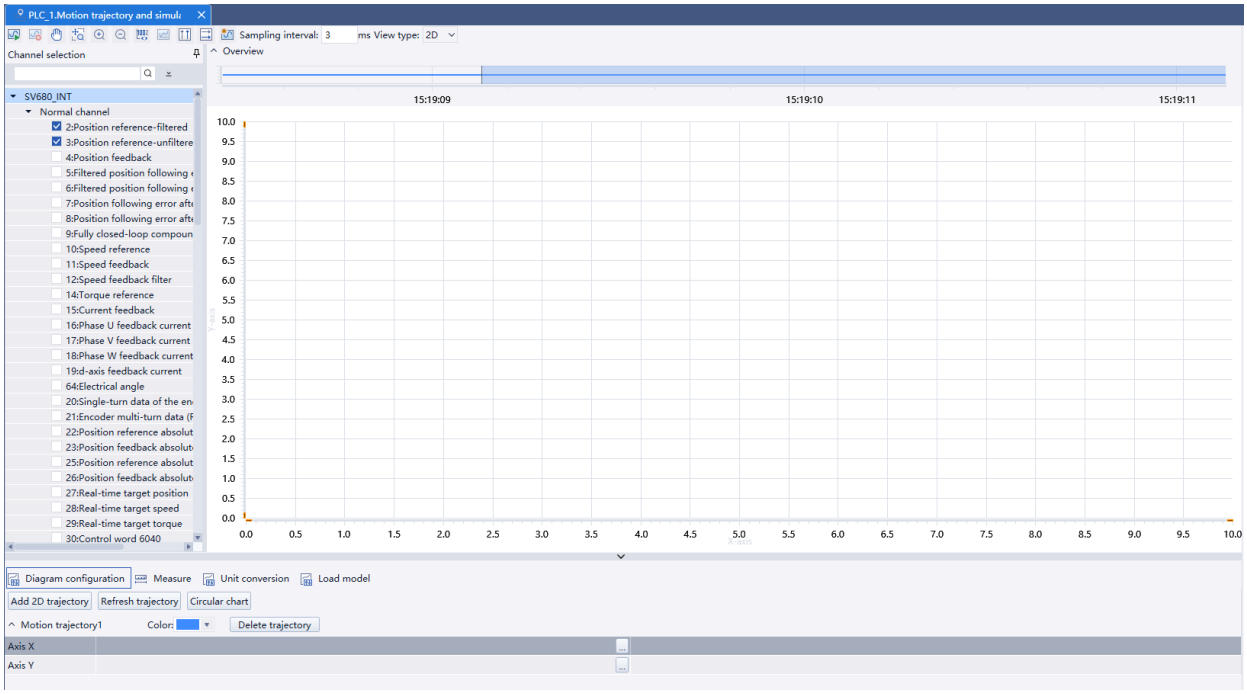
- Measurement

The view type of the "Measure" tab is set by "View type" in the toolbar, with options set to 1D, 2D, and 3D.

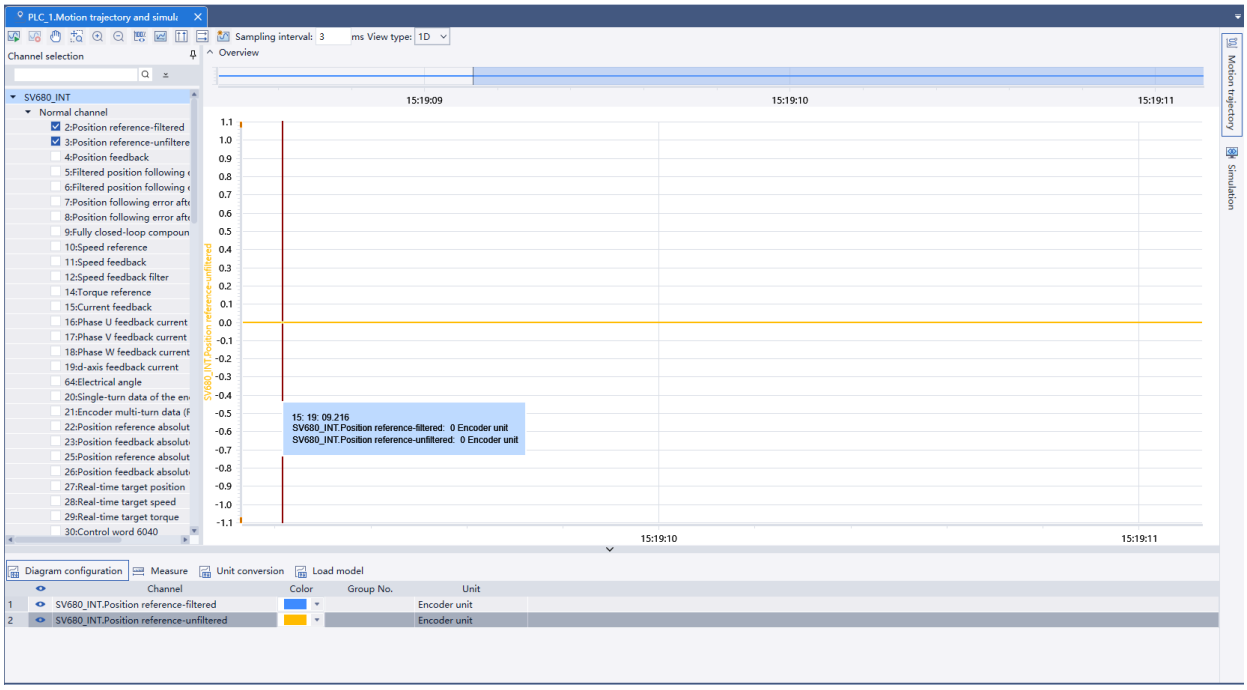
This tab includes X-axis cursor and Y-axis cursor measurement. For details, see the analysis auxiliary function in the oscilloscope section.



Measurement (1D)



Measurement (2D)

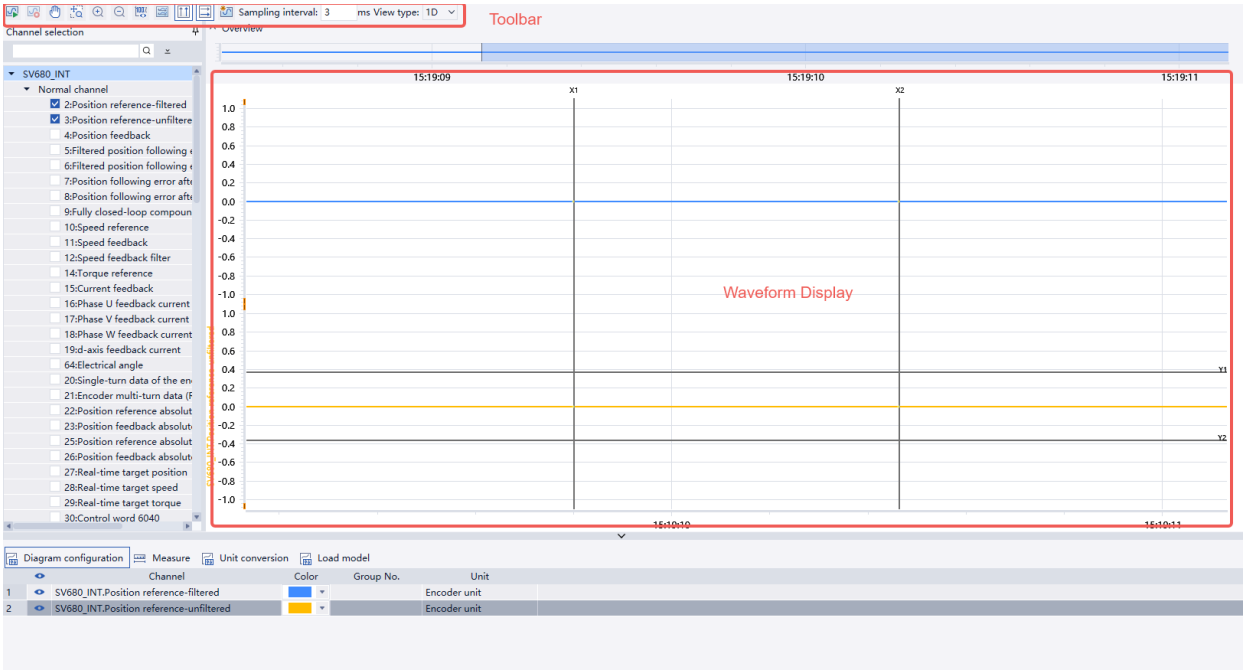


Measurement (3D)

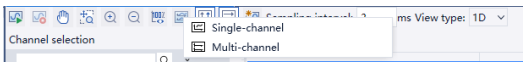
Function	Description	Restriction
Start/Stop measuring	After clicking "Start measuring," the system enters the measurement mode. In the 3D graphic, select two points as "Select point 1" and "Select point 2". After clicking "Stop measuring," the selected points and measurement results are cleared, and the view mode is displayed.	/
Reset	Clears the selected points and measurement results, but the system remains in the measurement mode. You can re-select points 1 and 2 for measurement.	/

- **Curve display**

Plots the sampling results as a curve.



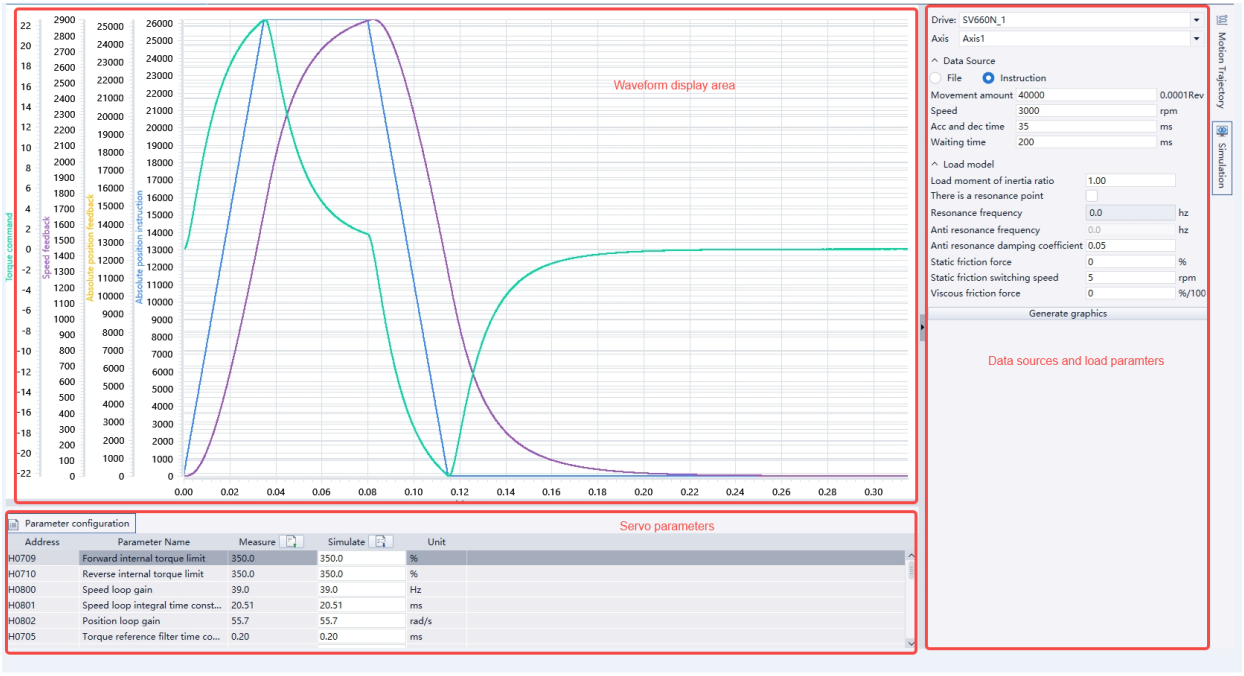
Function		Description	Restriction
	Start	Starts acquiring data and drawing the curve.	• The drive must be online. • When a fault occurs on the parameter configuration interface, this process fails.
	Immediate stop	Stops data acquisition.	/
	Pan	When enabled, you can drag the curve in the view, which is mutually exclusive with the zoom function.	It is disabled by default, selected and enabled at continuous acquisition, and disabled at triggered acquisition.
	Zoom	Zooms in or zooms out the curve. When the function is enabled, you can select a waveform segment from the X axis and Y axis for observation. It is mutually exclusive with the drag function. Drag the mouse from the left to right to zoom in the curve. Drag the mouse from right to left to zoom out.	• Disabled by default. • Unselected but enabled during continuous acquisition. • Unselected and disabled by default during trigger acquisition.
	Zoom +	Zooms in on the entire curve.	It is disabled by default, enabled at continuous acquisition, and disabled at triggered acquisition.
	Zoom -	Zooms out on the entire curve.	It is disabled by default, enabled at continuous acquisition, and disabled at triggered acquisition.

Function		Description	Restriction
	Show all	Displays 100% in the X and Y axis directions.	It is disabled by default, and enabled at continuous acquisition.
	Single-channel/ Multi-channel	Switches between single-channel and multi-channel. Single-channel display is the default type when motion profile is turned on. 	/
	Vertical cursor	Displays a group of cursors (t1, t2) in horizontal direction, and the statistical data of the sampled data between cursor (t1, t2) in the measurement table.	Restrictions: It is disabled upon acquisition and enabled after successful acquisition.
	Horizontal cursor	Displays a group of cursors (Y1, Y2) in vertical direction, and displays the cursor values in the measurement window.	• The Y axis cursor is displayed in the channel where the current viewing signal is located. Switching the current viewing signal also switches the Y axis cursor. • Restrictions: It is disabled upon acquisition and enabled after successful acquisition.
	Export data	Exports the oscilloscope data file (CSV format).	This function is disabled when acquisition starts.
Sampling interval		Sets the sampling interval.	/
View type		Sets the view type (1D/2D/3D).	/

17.3.28.2 Simulation

- **Function interface**

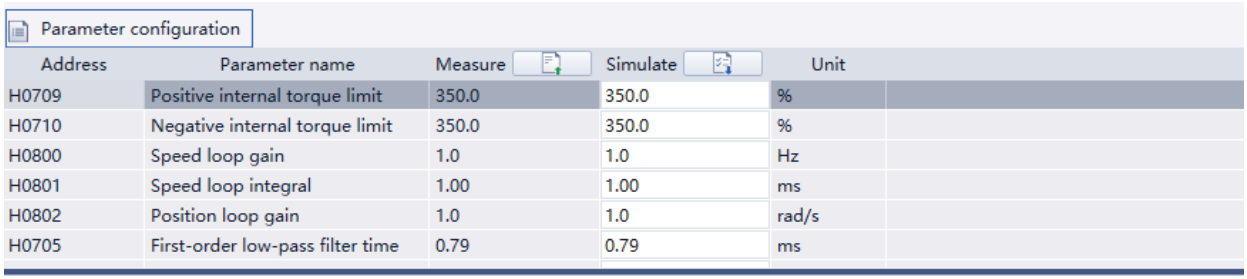
  Select a servo device in the project, and double-click "Motion trajectory and simulation" in the detailed view. Select the "Simulation" tab on the right side of the interface, which consists of the following parts: waveform display area, drive parameters, data source, and load model, as shown in the figure below.



- **Basic operating process**

  Configure servo drive parameters > data sources and load parameters > view waveform and analysis.

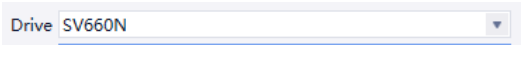
- **Parameter configuration**



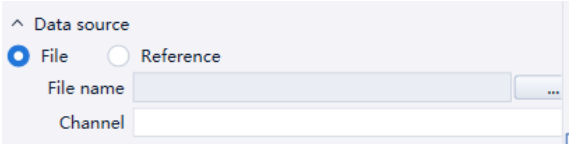
Function	Description	Restriction
Address	Parameter code	read-only
Parameter name	Description of the parameter	read-only
Measure	Click  to upload the drive parameters.	read-only
Simulate	Customized parameters are used for simulation. After simulation, click  to download.	/
Unit	Displays the unit of the current parameter.	read-only

- **Data source and load parameter configuration**

1. Select the drive for simulation through the drop-down box.

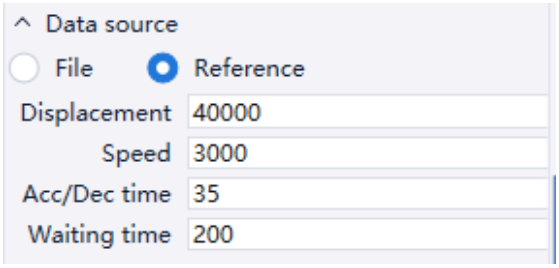


2. Select the data source, which includes "File" and "Reference".



Data source (file)

Function	Description	Restriction
File name	Click "..." to upload the data source file.	The file is exported by the "Motion trajectory and simulation" function.
Channel	Select the channel through the drop-down box.	/



Data source (reference)

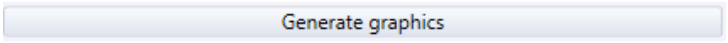
Function	Description	Restriction
Displacement	Defines the displacement of the reference.	Range: 0 to 65535
Speed	Sets the reference speed.	Range: 1 to 3000
Acc/Dec time	Sets the acceleration/deceleration time of the reference.	Range: 1 to 65535
Waiting time	Defines the waiting time of the reference.	Range: 0 to 65535

3. Sets the load model parameters.

Function	Description	Restriction
Load moment of inertia ratio	It sets the ratio of the load inertia against the rotor (of the motor) inertia.	Range: 0.00 to 120.00
Resonance point exists	Sets whether there is any resonance point.	/
Resonance frequency	Defines the model resonance frequency.	read-only
Anti-resonance frequency	Defines the model anti-resonance frequency.	◦ This is editable only when "Resonance point exists" is checked. ◦ Range: 0.0 to 300.0
Anti-resonance damping coefficient	Defines the model anti-resonance damping.	Range: 0.00 to 1.00
Static friction	Defines the static friction of the model.	Range: 0 to 100

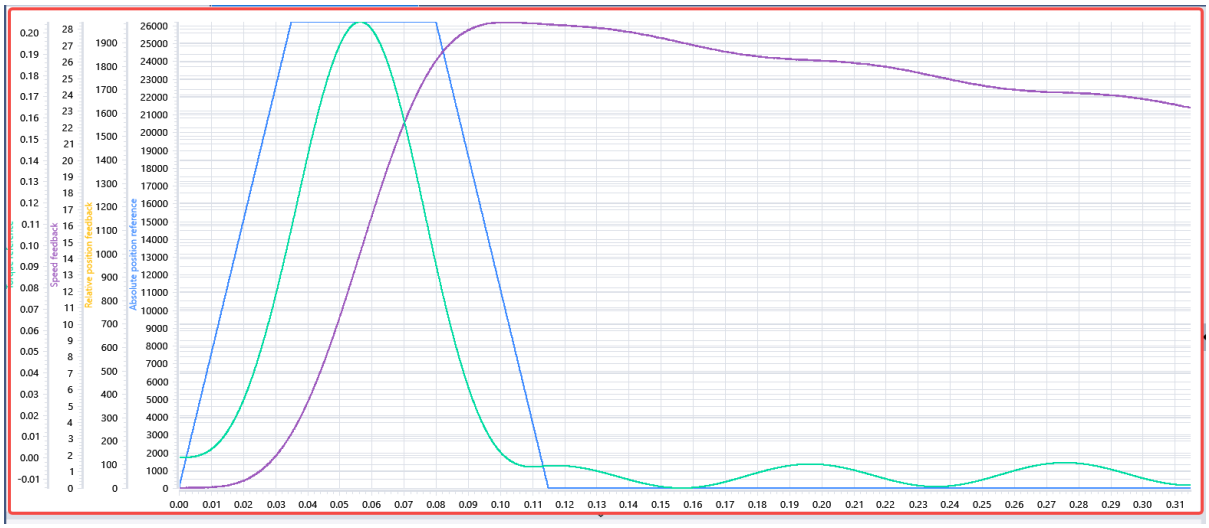
Function	Description	Restriction
Static friction switchover speed	Defines the static friction switchover speed of the model.	Range: 0 to 3000
Viscous friction	Defines the viscous friction of the model.	Range: 0 to 100

4. Click "Generate graphics" to generate the corresponding simulation curve.



- **Waveform display area**

  The waveform area displays the simulated curve, including torque references, speed feedback, relative position feedback and absolute position references.



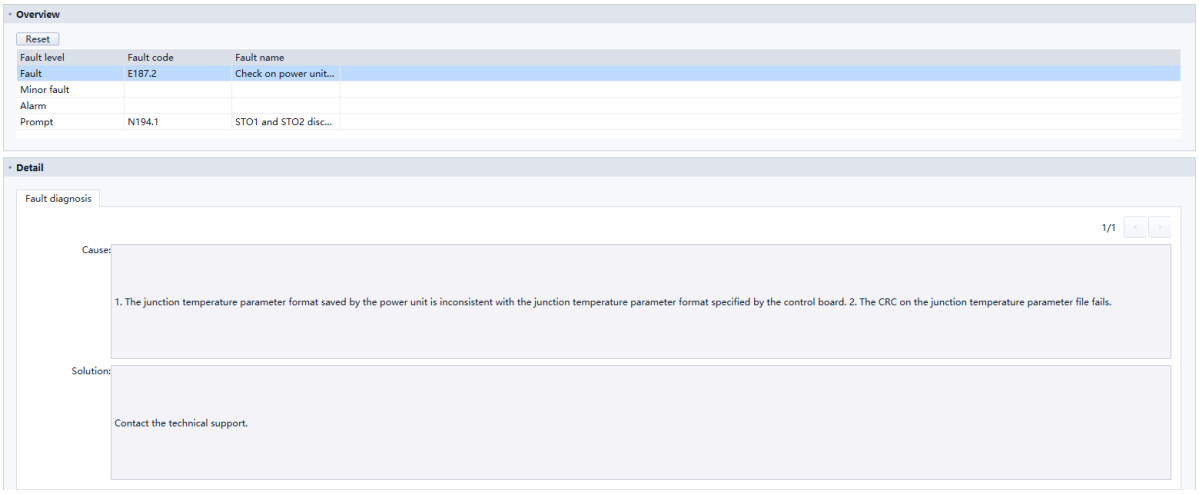
17.4 Common AC Drive Functions

17.4.1 Fault Management

17.4.1.1 AC Drive Fault Management

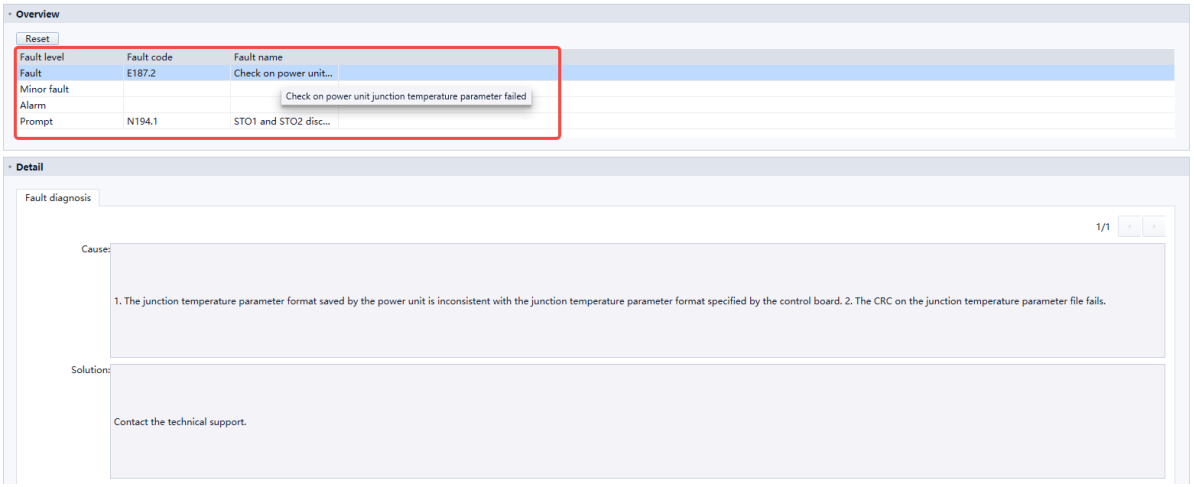
Select an AC drive in the project tree, double-click "Fault management" in the detailed view to open the interface. The interface displays detailed information about the current faults of the device and their solutions, helping users troubleshoot and resolve issues. The following figure shows the AC drive fault management interface, which mainly consists of the following parts:

- Fault overview area: Displays basic information of current faults and allows for a "Reset" operation.
- Fault diagnosis area: Displays information of the fault causes and solutions.



17.4.1.2 Fault Overview

After entering the fault management interface, the fault overview area is shown below.



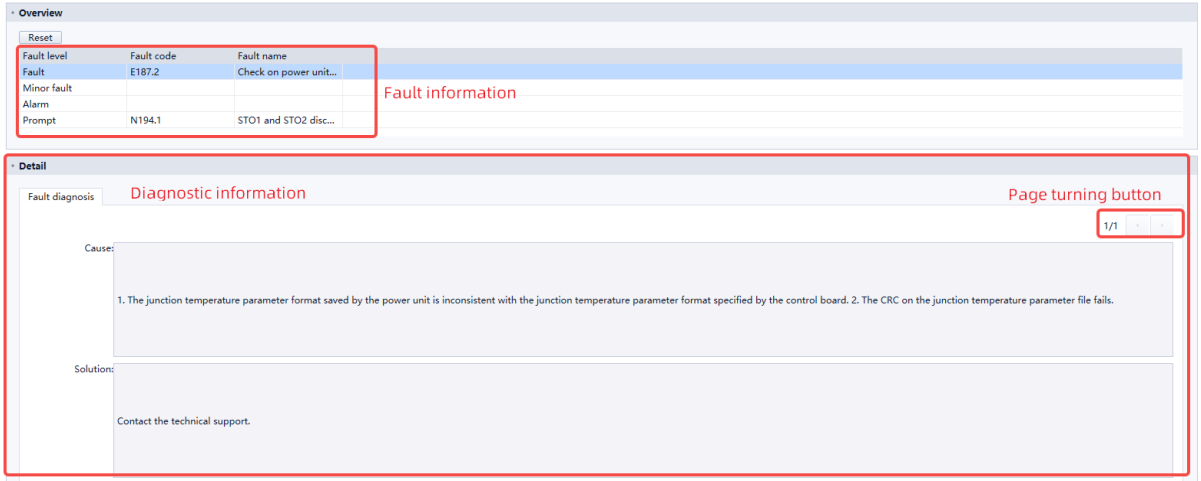
17.4.1.3 Troubleshooting

The main solution to faults is "Reset". Click "Reset" to reset the faults.

Restrictions:

The function is effective only on resettable faults.

The resettable fault can be reset only after the fault has been cleared.



Function	Description
Fault information	Displays the selected fault code and fault name.
Fault diagnosis information	Displays the fault cause, confirmation method, and solution of the selected fault code.
Page turning buttons	If there are multiple causes for a fault, you can use these buttons to view different fault diagnosis information.

17.4.1.4 Fault Diagnostics

The fault diagnosis area displays the causes and solutions for the current fault, as shown below. For detailed function descriptions, see the following table.



Description	Description
Cause	Displays the cause of the fault.
Solution	Displays the solution of the fault.
Page turning buttons	If there are multiple causes for a fault, you can use these buttons to view different fault diagnosis information.

17.4.2 Control Panel

The control panel is used to perform trial run on the AC drive and monitor actual values of drive status and critical variables.

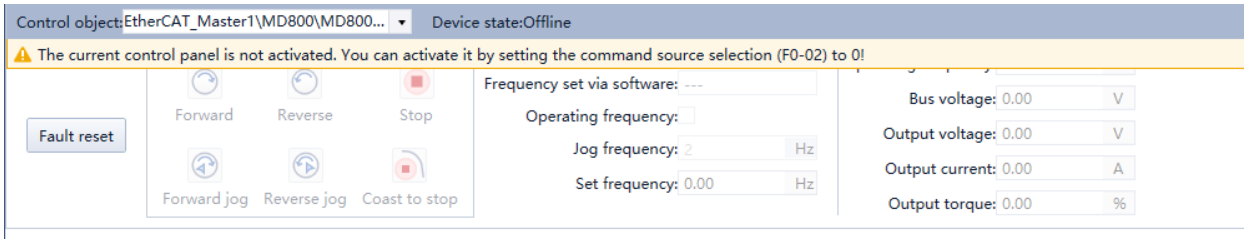
Function entry

After the device is added, double-click the "Control panel" node with the left mouse button to display the control panel interface in the workspace. All devices under the same project share a single control panel view. All devices on the same link are grouped into control objects for unified control. When entering the control panel from different links, the control object set changes to the device set of that link.

Path: Project tree > EtherCAT_Master > Device > Control panel.

Function interface

The control panel interface mainly consists of control objects, status monitoring, panel control, fault reset, motion control, frequency setting, and parameter monitoring.

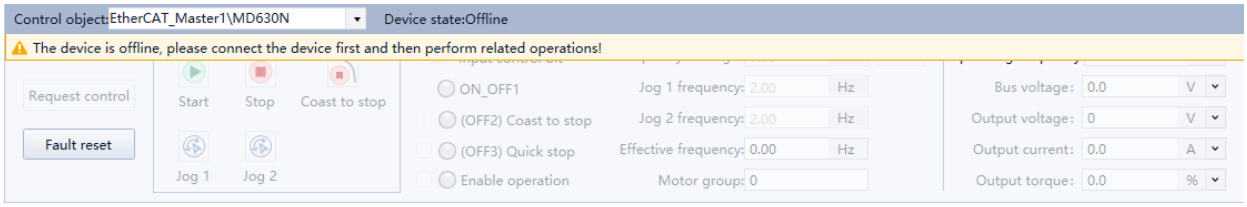


Control panel activation: When the interface is opened, it reads whether F0-02 (Command source selection) is 0. If so, the control panel is available; otherwise, it is disabled, and a prompt appears indicating that the parameter must be set to 0 for the control panel to be enabled.

Frequency setting validity: F0-03 (Selection of main frequency source X) is set to 0 or 1, and F0-07 (Frequency source superposition selection) is set to 2 or 3. When "Operating frequency" is checked, the main frequency source 0xF003 is set to 0, and the frequency superposition 0xF007 is set to 20. When "Operating frequency" is unchecked, the main frequency source 0xF003 is set to 5.

Note: The frequency input valid state cannot be switched during device operation.

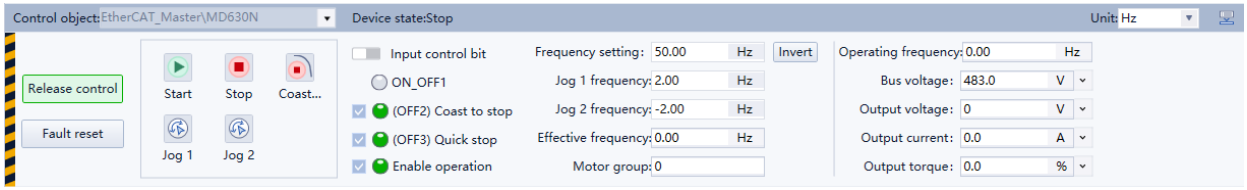
The control panel mainly consists of control objects, status monitoring, panel control, unit switching, control permission|fault reset, motion control, control bits, parameter setting, and parameter monitoring.



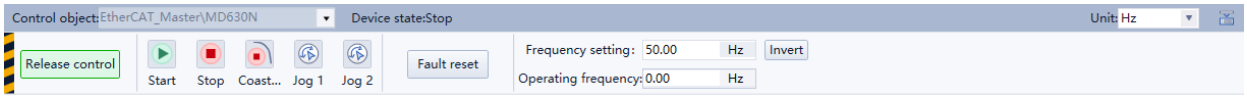
Note: The control panel can only be used after the disabled state is released.

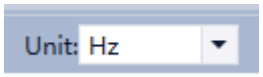
Taking the MD630/MD600 series AC drive as an example, the control panel interface is shown in the figure below:

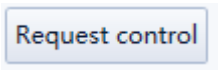
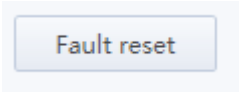
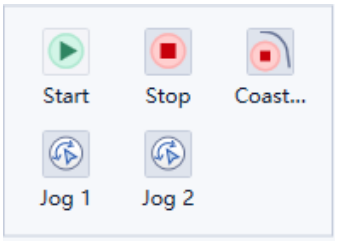
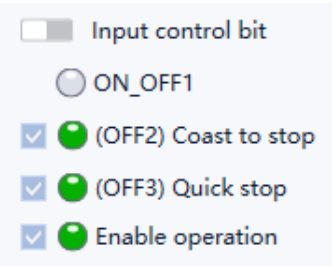
Complete panel:



Simplified panel:



Description	Icon	Description
Control object		Function: You can select the device to be controlled from the drop-down box. All devices in the network link will be controlled. For the multi-axis device, the drop-down box displays "device name + parent device name". For example, the inverter of MD800 is shown as Inverter_1[MD800_1]. The rectifier device does not support this function. Note: • The name and ToolTips change with the name of the project node. • It is selectable, but not editable. • The control object is automatically switched when the panel is accessed from different entry points. • If the current control panel is in a controlled state, it cannot be switched to control other devices.
Device state		Displays the current device state.
Fault information		Displays the current highest-priority fault information of the device.
Unit switchover		Used to switch units. After switching units, the corresponding descriptions and units in the parameter setting and monitoring areas will change synchronously, and related values will be converted accordingly. Note: This function is currently supported only for MD630N and MD630S series products.
Complete panel/ Simplified panel switchover		Switches the panel view between full and simplified modes. Click the icon once to switch to the simplified panel, and click again to return to the complete panel.
Floating panel/Fixed panel switchover		Switches the panel state. Click the icon once to switch to the floating panel, and click again to return to the fixed panel.
Close		Closes the control panel.

<table>
<tr><th>Description</th><th>Icon</th><th>Description</th></tr>
<tr><td>Request control/ Release control</td><td>
</td><td>Click the button once to request control for the current control object, and click again to release control.
Note:
The button is disabled if the control panel is not activated.</td></tr>
<tr><td>Fault reset</td><td>
</td><td>Used to reset the resettable faults for all devices.</td></tr>
<tr><td>Motion control</td><td>
</td><td>Used to control device motions.<table><tr><td>Start</td><td>Starts the device.</td></tr><tr><td>Immediate stop</td><td>Stops the device.</td></tr><tr><td>Coast to stop</td><td>Stops the device using the coast to stop mode.</td></tr><tr><td>Jog 1/Jog 2</td><td>Controls the device motion using jog mode. Press the button to start motion, and release to stop.</td></tr></table>Note:
The button is disabled if the control permission is not acquired.</td></tr>
<tr><td>Input control bit</td><td>
</td><td>Controls whether to enable the control function of control bit.<table><tr><td>ON_OFF1</td><td>Starts or stops the status indicator.</td></tr><tr><td>(OFF2) Coast to stop</td><td>Controls the stop mode.</td></tr><tr><td>(OFF3) Quick stop</td><td>Controls the stop mode.</td></tr><tr><td>Enable operation</td><td>Controls the operation enablement.</td></tr></table>Note:
The button is disabled if the control panel is not activated.</td></tr>
</table>

<table>
<tr><th>Description</th><th>Icon</th><th>Description</th></tr>
<tr><td>Parameter settings and detection</td><td>
</td><td>Reads and writes related parameters.
<table>
<tr><td>Frequency setting</td><td>Writes upon focus loss or through Enter key.</td></tr>
<tr><td>Inversion</td><td>Used to invert the frequency setting value and write it.</td></tr>
<tr><td>Jog 1 frequency/Jog 2 frequency</td><td>Writes upon focus loss or through Enter key.</td></tr>
<tr><td>Effective frequency</td><td>It cannot be edited.</td></tr>
<tr><td>Motor group</td><td>It cannot be edited.</td></tr>
<tr><td>Running frequency</td><td>It cannot be edited.</td></tr>
<tr><td>Bus voltage/Output voltage /Output current/Output torque</td><td>Monitored items can be selected from the drop-down list, but cannot be edited.</td></tr>
</table>
Note:
• The setting is written upon focus loss or through Enter key.
• Read upon focus loss.</td></tr>
</table>

Status monitoring

This function can be used to monitor drive status and actual value status:

Drive status:

- Operation status, fault code, and fault information.
- Operation status: Forward run, reverse run, stopped, tuning, and faulty.

Parameter setting and parameter monitoring may vary across different device series. Refer to the actual device for details.

All status monitoring items are non-editable.

General settings

- When all devices in the current project are deleted or the current project is closed, the control panel view is closed.
- Among control objects, motion operations interfere with each other, but monitoring does not. That is, only one device can be controlled for motion at a time.

17.4.3 Function Diagram

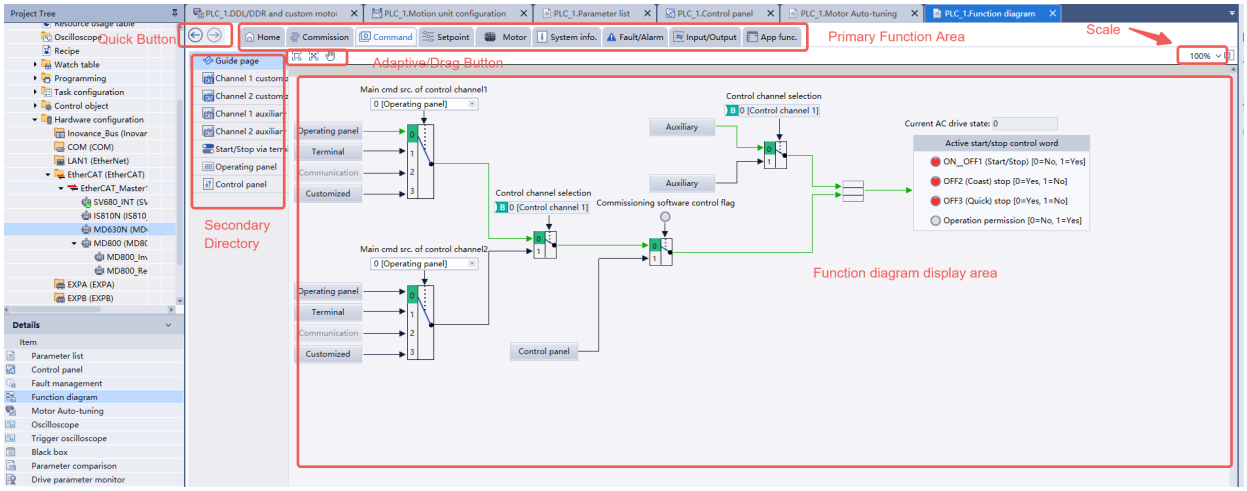
This function is currently supported only for MD630 series products.

The function diagram presents the relationships between parameters in a data flow format, which significantly lowers the learning cost for users to operate the AC drive. This function is specific to certain devices and may not be supported on all models.

The following table lists the models that support the function diagram.

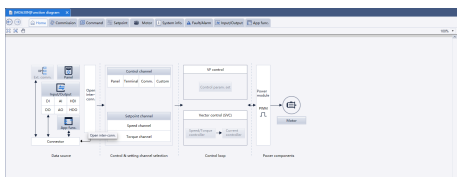
Series	Model
MD630	MD630N
	MD630S

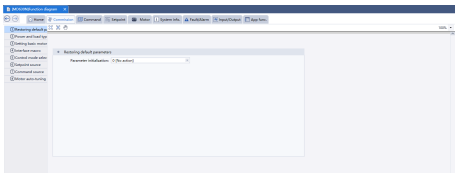
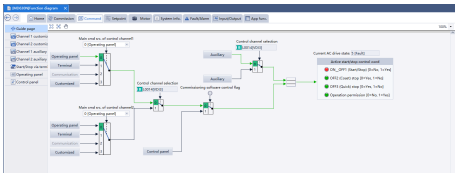
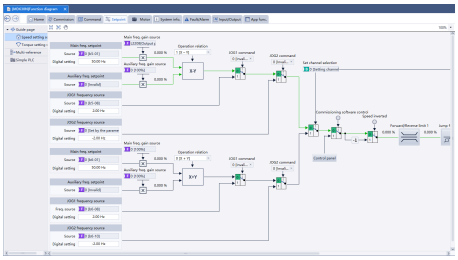
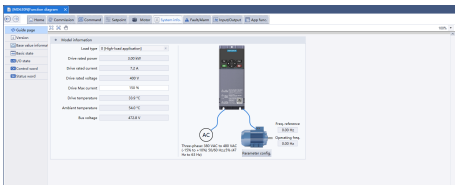
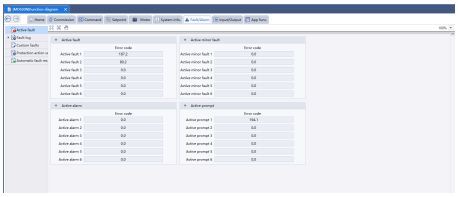
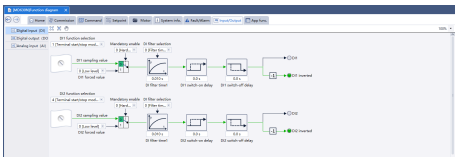
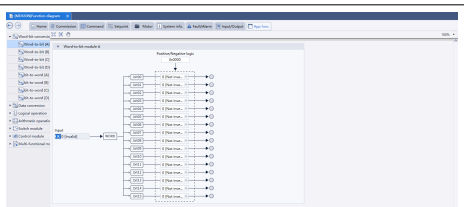
Taking the MD630 as an example, the structure tree of the function diagram is as follows; the home page provides an overview of the data flow in the function diagram. The function diagram includes 62 pages of mainstream functions, with each page containing different functions. The function diagram has 9 primary directories, each of which contains multiple secondary directories.



Description	Description
Shortcut icons	Back/Forward icons. Navigates to the previous/next page. If the current page is being accessed for the first time, the icon will be inactive.
Primary function area	Primary menu: Switches between primary directories.
Secondary directory	Secondary directory navigation tree: Switches between secondary directories.

Primary function area

Description	Icon	Description
Home page		The home page offers a comprehensive overview of the function diagram's functions and the logical relationships between modules. It also provides direct access points to each module, enabling you to easily navigate to the desired module for queries and settings.

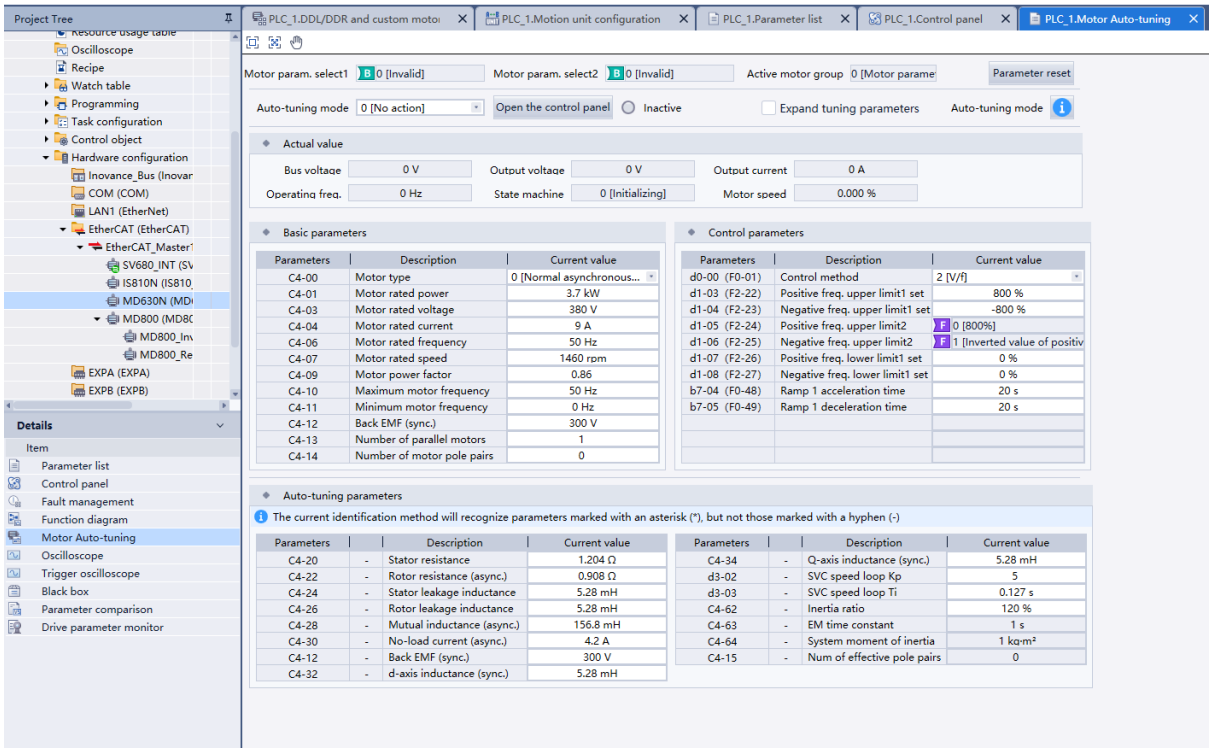
Description	Icon	Description
Quick commissioning		Used for basic parameter configuration and commissioning, helping you get started quickly.
Command source		Used to set system control commands from different sources to achieve motor start/stop, jog, forward/reverse operations.
Setpoint source		Used to set system frequency commands from different sources to achieve jog, normal operation, multi-reference, and simple PLC functions.
Motor parameters		Used to display motor-related parameters, making it easier for you to set the rated parameters for different motors and the current motor data group used by the AC drive.
System information		Used to query real-time system status information, software version information, base value information, etc.
Fault/ Alarm		Used to display fault-related parameters, fault protection, and auto-reset functions, helping you locate and analyze issues and providing early protection for the AC drive.
Input/ Output		Used to set terminal input/output functions and analog terminal input/output signals of the AC drive.
Application Function		Used to set parameters related to free function blocks, helping you freely select relevant modules to implement complex logic functions.

17.4.4 Motor Auto-Tuning

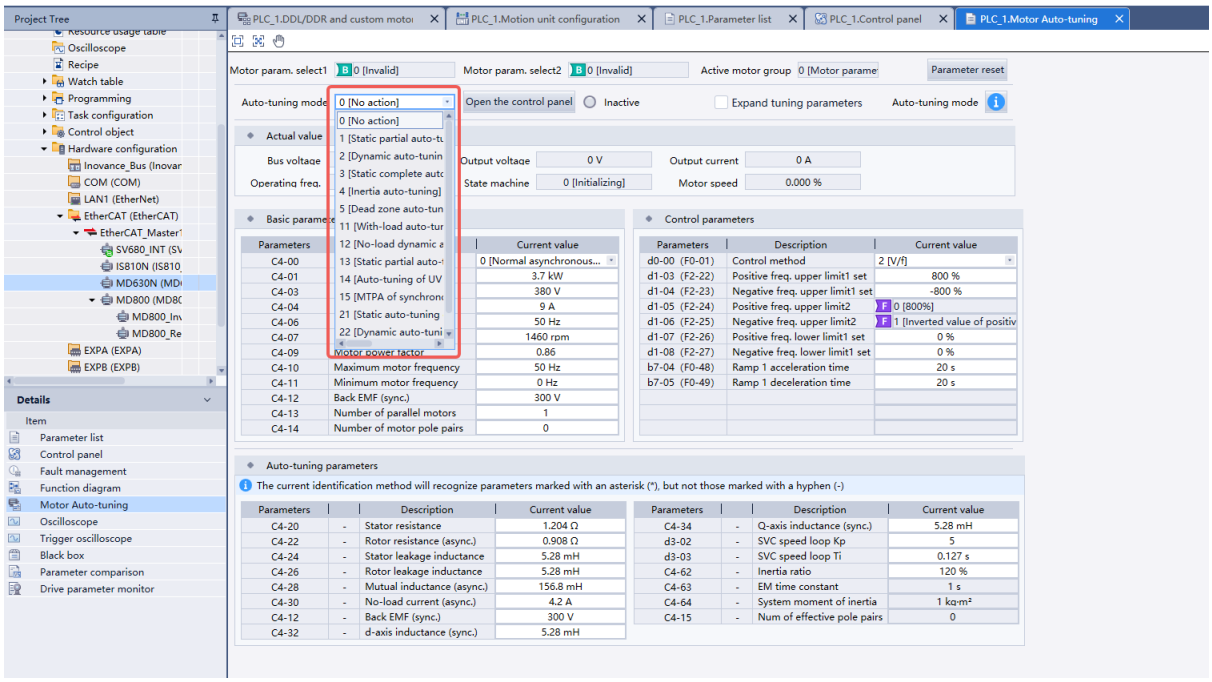
This function is currently supported only for MD630 series products.

Motor auto-tuning refers to the process where the AC drive autonomously learns the motor parameters.

1. Click "Motor auto-tuning" in the lower-left directory tree to enter the interface.



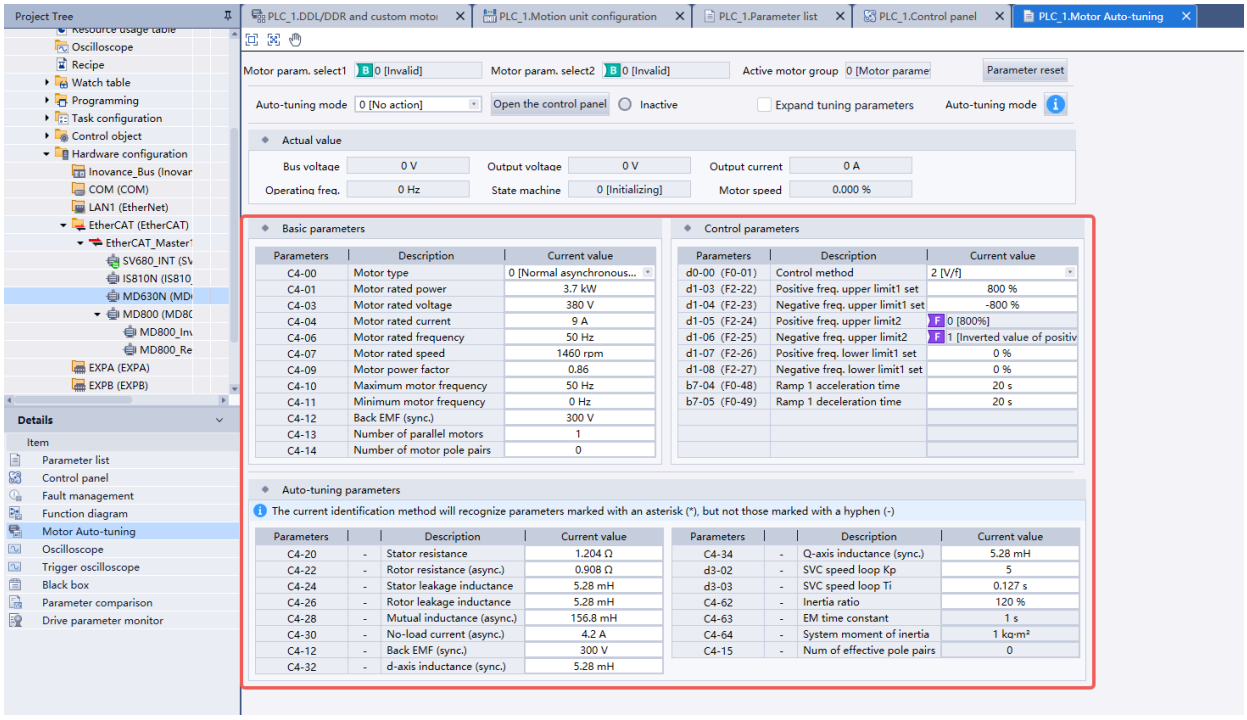
2. Set the auto-tuning mode.



3. Click "Open the control panel", "Request control", and then "Start" to start the motor.



4. Wait for the motor auto-tuning to complete, then check the related parameter changes.



17.4.5 Black Box

17.4.5.1 Overview

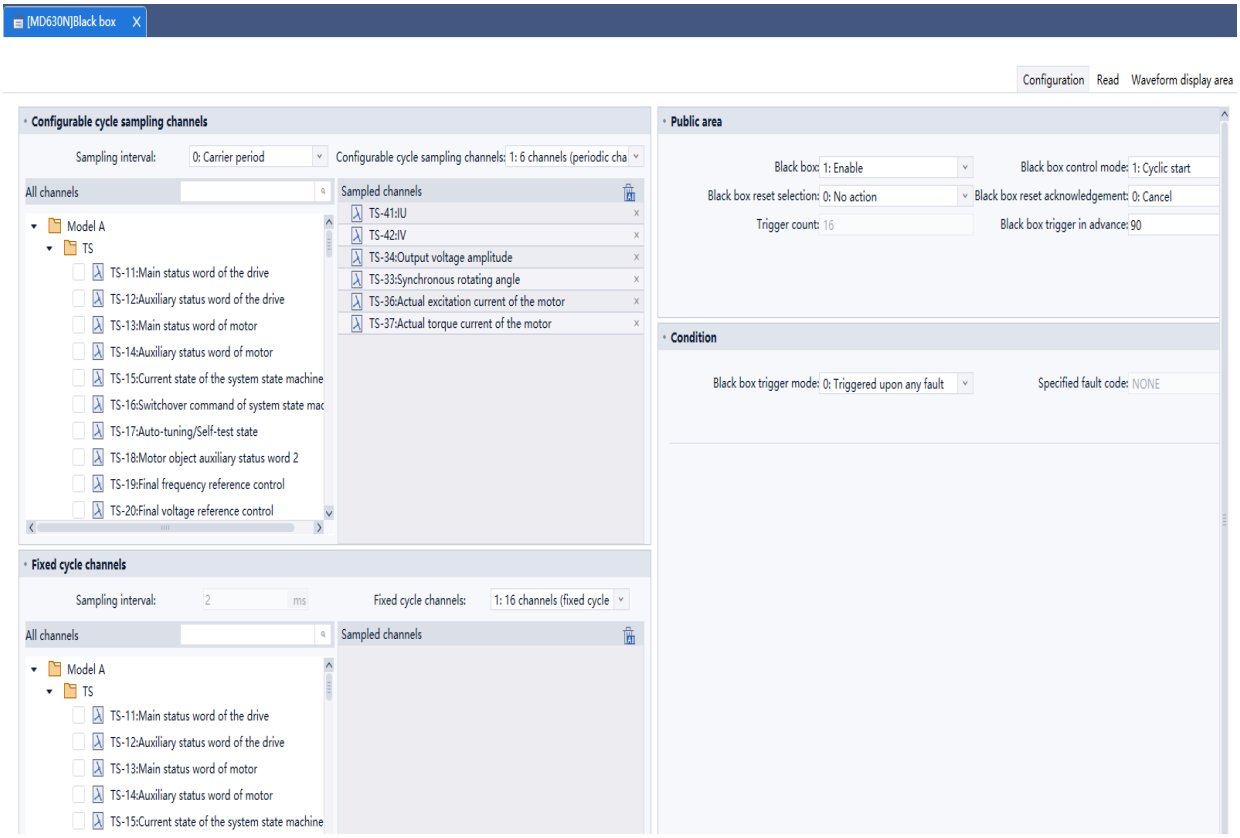
This function analyzes the recorded data, operation statuses, and fault causes of the related devices to improve their maintainability.

The following models currently support the black box feature:

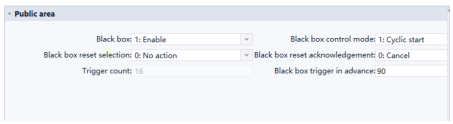
Series	Model
MD630	MD630N
	MD630S

Select an AC drive in the project tree, and double-click "Black box" in the detailed view to open the interface, which contains two tabs: "Configuration", "Read", and "Waveform display area".

17.4.5.2 Configuration

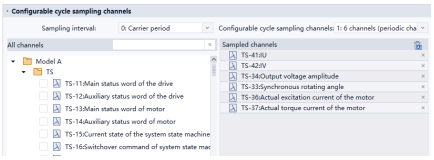


- **Public area:** Configuration area of black box common functions.

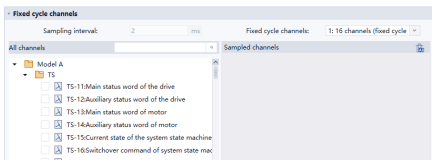
Description	Figure	Description
Enable black box		Enables or disables the black box function.
Black box control mode		Single start, stop after being triggered: Saves the record data after triggering, without repetitive startup. Cyclic start: After data is recorded after triggering, the black box enters the waiting status again.
Black box reset selection		No action: The original configuration is kept. Restore the black box configuration parameters: Restores the black box configuration parameters to default values.
Black box reset acknowledgement		Cancel: Cancels restoring black box parameters to default values. OK: Restores the black box configuration parameters to default values.
Trigger count		Number of triggering times of the black box.

Description	Figure	Description
Black box trigger in advance		Defines the pre-trigger setpoint of the black box (unit: %; range: 0 to 100).
RAM with history data		Checks whether there is historical data in the RAM.

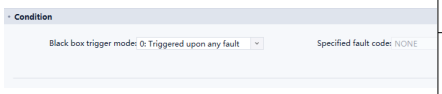
- **Configurable cycle sampling channel:** Configures the variable of the record of the channel with configurable sampling cycle.

Description	Figure	Description
Channel search		Enter the keyword in the "All channels" text box and click to display the channels matching the keyword.
Configurable cycle sampling channels		Number of channels to be recorded
All channels		Displays available options in all channels with configurable sampling cycle. The data source is the configuration file.
Sampled channels		Displays the selected channels. The number of channels is adjusted according to the number of fixed cyclic sampling channels (3 or 6 channels supported).

- **Fixed cycle channels:** Configures the variables of the record of the archived cycle sampling channel.

Description	Figure	Description
Channel search		Enter the keyword in the "All channels" text box and click to display the channels matching the keyword.
Fixed cycle channels		Number of channels to be recorded.
All channels		Displays available options in all channels with fixed sampling cycle. The data source is the configuration file.
Sampled channels		Displays the selected channels. The number of channels is adjusted according to the number of fixed cyclic sampling channels (8 or 16 channels supported).

- **Condition:** Black box triggering condition

Function	Figure	Description
Black box trigger mode		Options include "Triggered upon any fault", "Triggered upon designated fault", "Custom", "Custom or triggered upon any fault", and "Custom or triggered upon designated fault".
Specified fault code		If the trigger mode is set to "Triggered upon designated fault" or "Custom or triggered upon designated fault", you must set this parameter to any device fault. If the trigger mode is set to other values, the parameter is unavailable.

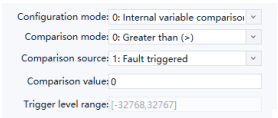
Trigger condition: Configures the black box trigger condition. When the trigger mode is set to a mode containing "Custom", you need to set the trigger condition. When the trigger mode is set to other values, the trigger condition parameters are unavailable.

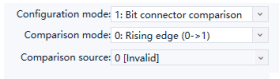
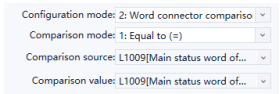
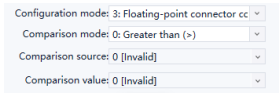
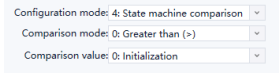
Trigger mode: user-defined condition A, user-defined condition B, user-defined condition A or B, and user-defined conditions A and B.

Configuration modes of conditions A and B include internal variable comparison, bit connector comparison, word connector comparison, float point connector comparison, and state machine comparison.

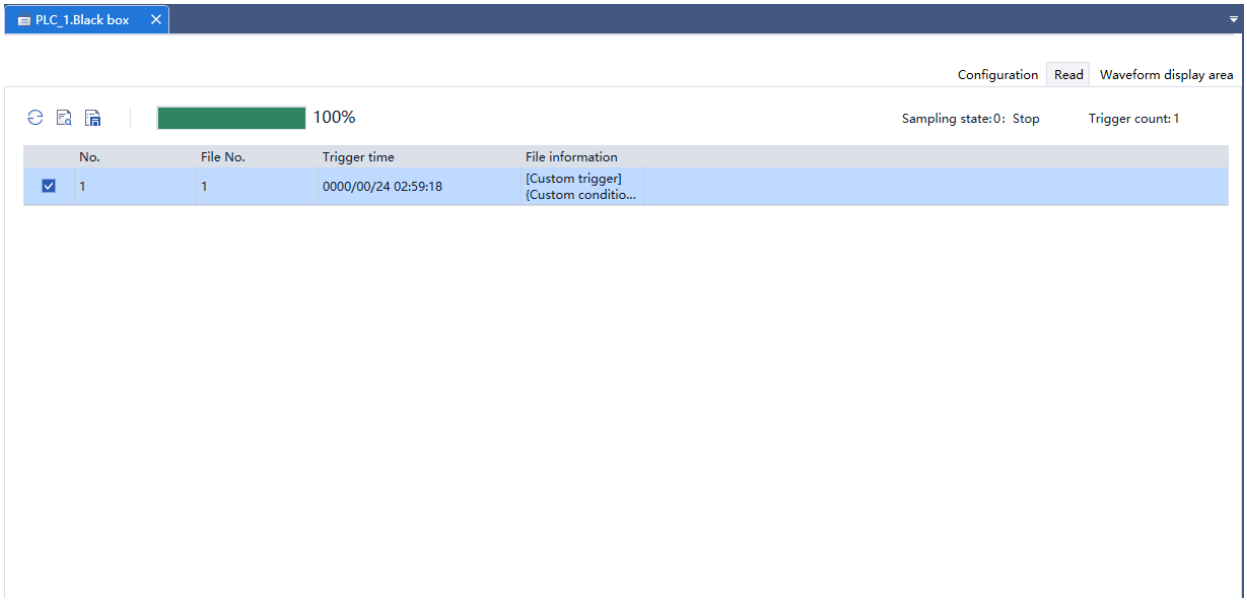
In different configuration modes, the comparison method, source, and value may differ. Refer to the table below for details.

Table17-4-5-2-1

Description	**Figure**	Comparison Mode	Comparison Source	Comparison Value	Trigger Range
Internal variable comparison		Greater than, equal to, less than, greater than or equal to, less than or equal to	Black box channel + unconditional save + fault trigger + save during operation	Number	Channel range

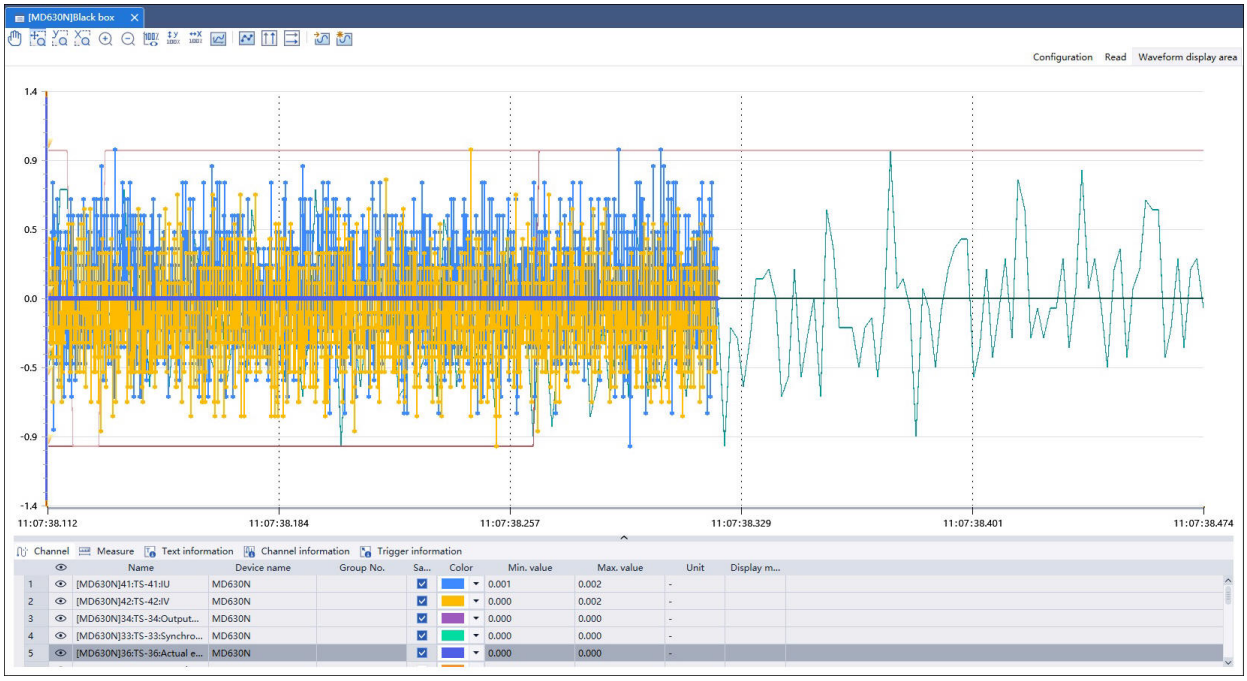
Description	Figure	Comparison Mode	Comparison Source	Comparison Value	Trigger Range
Bit connector comparison		Rising edge, falling edge, edge change, equal to logic 0, equal to logic 1	Invalid, connector B	/	/
Word connector comparison		Greater than, equal to, less than, greater than or equal to, less than or equal to	Invalid, K connector	Invalid, K connector	/
Float point connector comparison		Greater than, equal to, less than, greater than or equal to, less than or equal to	Invalid, connector F	Invalid, connector F	/
State machine comparison		Greater than, equal to, less than, greater than or equal to, less than or equal to	/	Initialization, switch-on disabled, switch-on enabled, operation permission, start, run, stop, self-test, auto-tuning	/

17.4.5.3 Read



Description	Icon	Description
Refresh directory		Refreshes the black box data directory recorded in the device.
Read file		Based on the selected file, obtains the configuration information and reads the data from the device. The displayed dialog box prompts you to save the read data. Select a proper directory and click "OK" to save the data to the specified directory. Then go to the waveform display area to view the waveform.
Save file		Saves the read file to local.
Progress bar	0%	Displays the operation progress bar.

17.4.5.4 Waveform Display



The toolbar on the "Waveform display area" is the same as that in the "Oscilloscope" interface. For details, see "Analysis Auxiliary Function". The waveform curve is displayed in the middle area and related information is displayed in the lower area. See the following table for details.

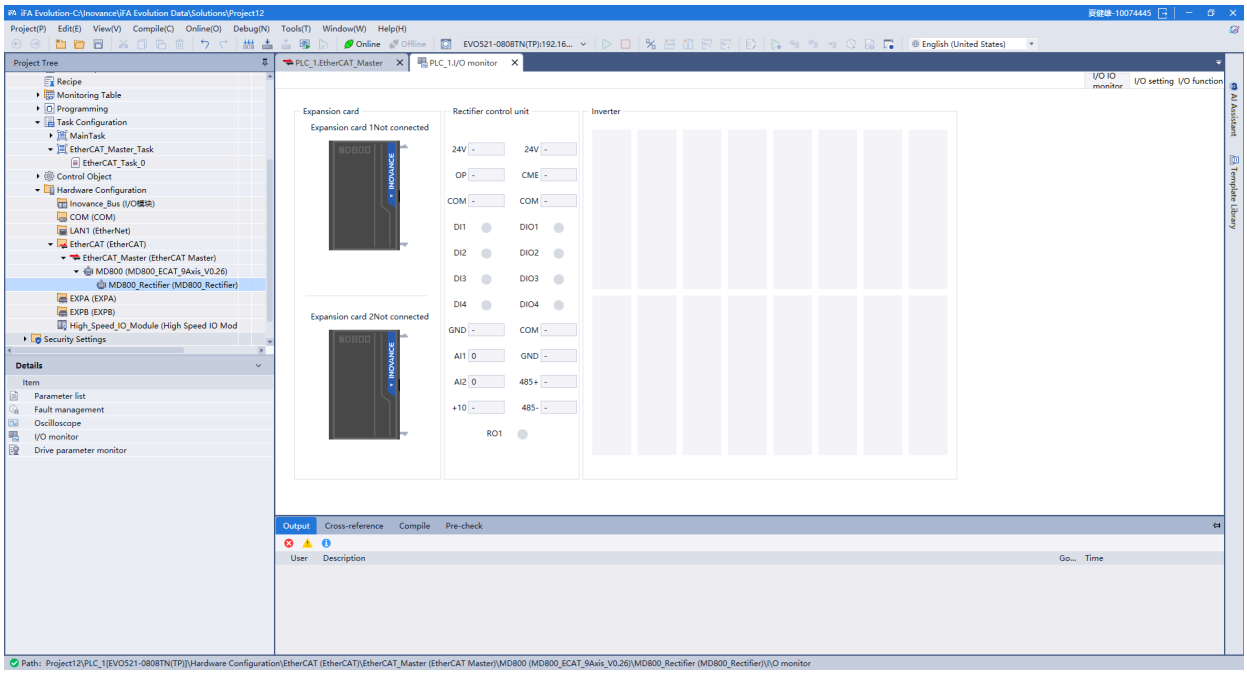
Function	Description
Channel	Displays channel information. You can also set the channel color and group number.
Measurement	Measures the waveform through the cursor, and displays the values of each curve in real time.
Channel information	Displays the sampling frequency, total sampling point count, and sampling point count before trigger of the speed and position references.
Trigger information	Displays the following two trigger messages: • Condition information: Displays the trigger condition based on the actual value. • Fault information: Displays the fault information generated upon triggering. Fault information is displayed dynamically; if there are no faults, nothing is displayed.
Text information	Displays relevant information about the device.

17.4.6 I/O monitoring

This feature is supported only for the MD800_Rectifier products.

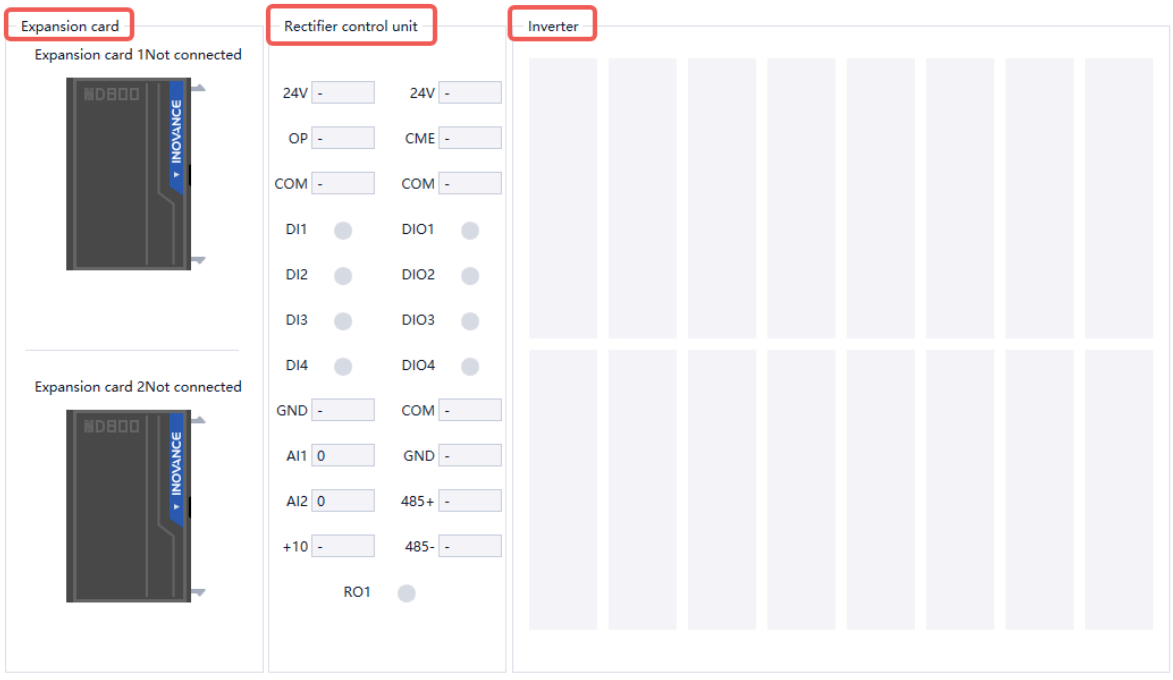
Select an AC drive in the project tree, and double-click "I/O monitor" in the detailed view to open the interface.

The interface mainly consists of three parts: I/O monitoring, I/O settings, and I/O functions, as shown below:

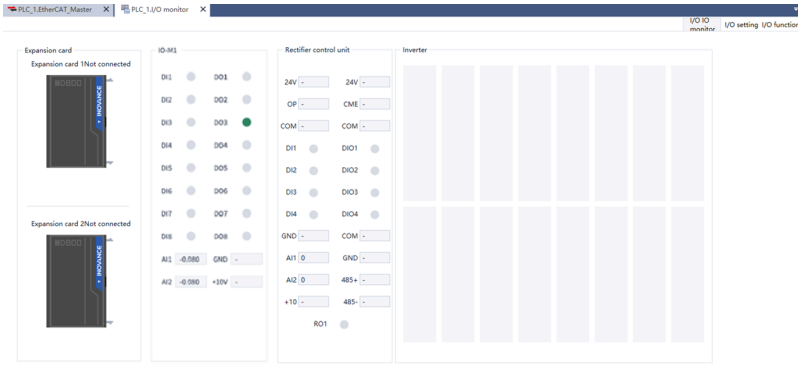
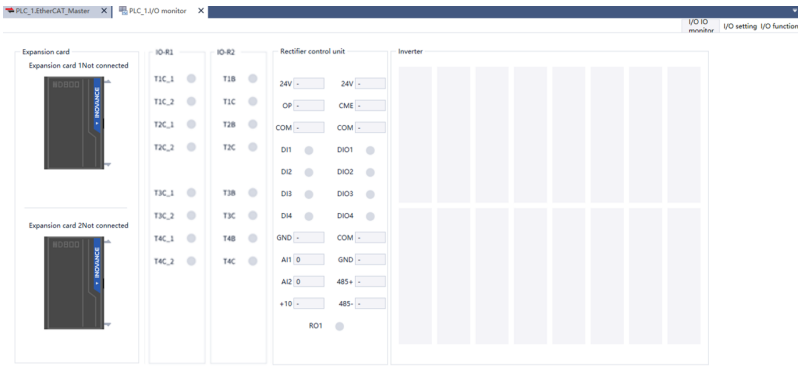


- ## I/O monitoring

This tab is used to monitor the status of expansion modules, rectifier control units, and inverter control units.

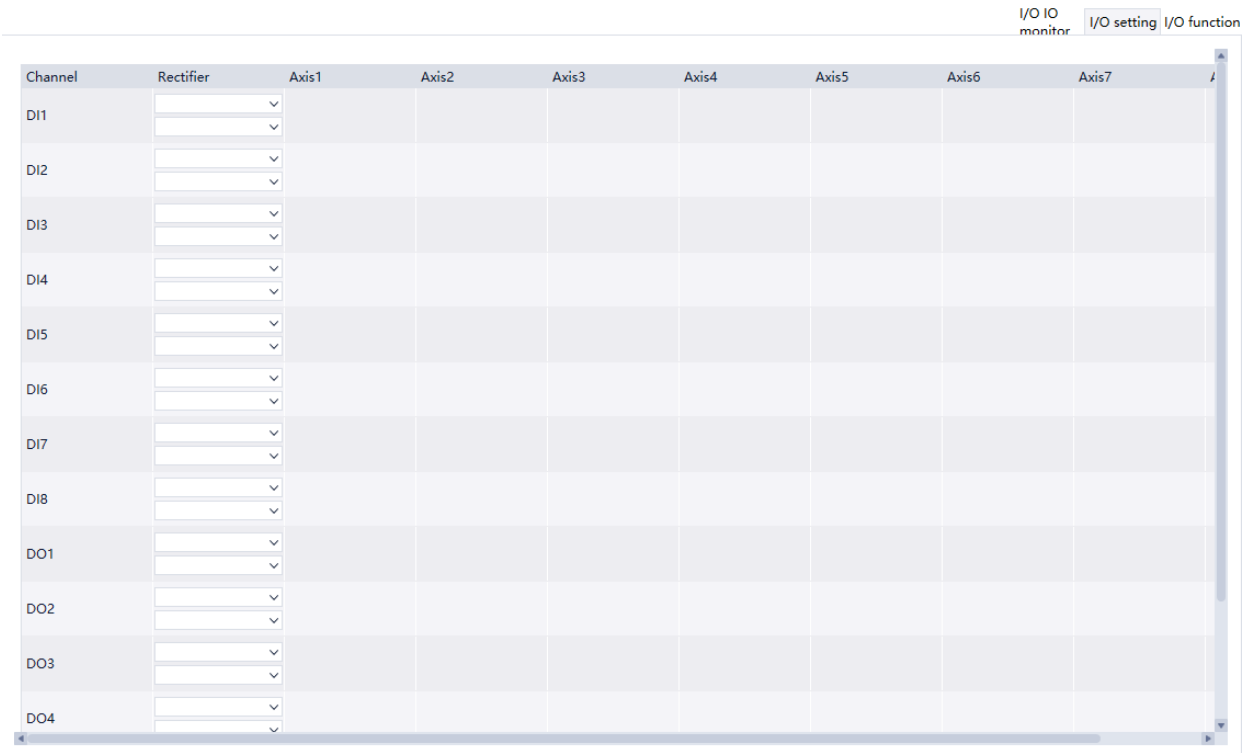


Function	Description
Expansion card	Supports insertion of IO-M1/IO-R1/IO-R2 type expansion cards.

Function	Description
	After an IO-M1 type expansion card is connected, the interface is as shown below. An "IO-M1" monitoring bar is generated between "Expansion card" and "Rectifier control unit".  After an IO-R1/IO-R2 type expansion card is connected, the interface is as shown below. "IO-R1"/"IO-R2" monitoring bars are generated between "Expansion card" and "Rectifier control unit". 
Rectifier control unit	Displays the DI/DO/AI status of the rectifier modules.
Inverter	Displays the status of each inverter expansion module under MD800. POW: Orange indicates the device is powered on. Gray indicates the device is powered off. RUN/STOP: Green with text "RUN" indicates the motor is running. Gray with text "STOP" indicates the motor is stopped. Fault code: If the device is faulty, the mark is displayed in red with the corresponding fault code text; otherwise, it is displayed in gray with no text.

- **I/O setting**

This tab is used to configure DI/DO/AI related functions for rectifier and inverter.



Function	Description
Read	Reads I/O configuration information from the device and updates it to the I/O settings page.
Write	Writes the user-configured I/O configuration information into the device.
Channel	I/O channel name (non-editable)
Rectifier	Physical channel selection and function configuration for the I/O channel corresponding to the rectifier
Axis 1 to Axis 8	Physical channel selection and function configuration for the I/O channel corresponding to the inverter
Restrictions	The hardware source for the same inverter cannot be the same. For example, if the hardware source selection for DI1 (F4-00) is set to 002, then the hardware source selection for DI2 (F4-02) cannot be set to 002.
	DIO function configuration: When a DIO hardware port is assigned to a DI function, this port cannot be assigned to a DO function. For example, when the hardware source selection for DI1 (F4-00) is set to 005 (rectifier DIO1), the rectifier DIO1 cannot be selected as the hardware source for the DO2/RO2 hardware source (F5-02) function, and the selection option for DO2/RO2 skips 001.
	When a DIO hardware port is assigned to a DO function, this port cannot be assigned to a DI function. For example, when the DO1/RO1 hardware source selection (F5-00) is set to 001, the rectifier DIO1 cannot be selected as the hardware source for the DI1 hardware source (F4-00) function, and the selection option for DI1 skips 005.

Function	Description
DO channel multiplexing	The DO hardware channels can be multiplexed across multiple axes, and the final DO status is determined by the logical OR of all associated axis statuses. For example, if DO1 is associated with the fault output function of axis 1 and axis 2, then DO1 becomes active whenever either axis 1 or axis 2 encounters a fault.

- **I/O functions**

This tab displays the device's I/O configuration information in a table. Unconfigured I/Os are displayed as blank, and modules without configured I/O functions are not displayed.



Function	Description
Read	Reads I/O configuration information from the device and updates it on the current page.
Card	I/O channel source (rectifier or I/O expansion card)
Channel	I/O channel name
Rectifier	I/O configuration function corresponding to the rectifier
Axis 1 to Axis 8	I/O function configured for the inverter module

17.5 Servo Parameter Description

See the guides in the table below for detailed parameter information of different servo drive series.

NOTE

The following guides require network access permission to open.

Product Series	Document
SV660N	SV660N Series Servo Drive Function Guide
	SV660N Series Servo Drive Troubleshooting Guide
	SV660N Series Servo Drive User Guide
SV680N	SV680N Series Servo Drive Function Guide
	SV680N Series Servo Drive Troubleshooting Guide
SV630N	SV630N Series Servo Drive User Guide
SV670N	SV670N Series Servo Drive User Guide
STD60N	STD60N Series Stepper Function Guide
	STD60N Series Stepper Troubleshooting Guide
IS650N-PLUS	IS650N-PLUS Series Servo Drive User Guide
SV680N-INT	SV680-INT Series Servo Drive User Guide
SV670N-INT	SV670-INT Series Servo Drive User Guide
SV670N_LK	SV670N-LK Series Servo Drive User Guide
SVD60N	SVD60N Series Servo Drive Guide Package
SV680LN	SV680LN Series Direct Drive Motor Servo Drive Guide Package
IS810N	IS810N Series Servo Drive (Multidrive System) User Guide

17.6 AC Drive Parameter Description

See the guides in the table below for detailed parameter information of different AC drive series.

NOTE

The following guides require network access permission to open.

Product Series	Document
MD800	MD800 Series AC Drive (Multidrive System) User Guide
MD810	MD810 Series AC Drive (Multidrive System) Function Guide
MD630	MD630 Series General-Purpose AC Drive User Guide

18 Common Instructions

Multiple libraries with different names may exist under the same functional category in the toolbox. To view instructions each library contains, you can obtain detailed information through the operations in the 12.3 library reference.

The following table lists the differences in software library supported by each product; general libraries are not mentioned as they are supported by all products.

Category		Library	EVO400	EVO500	EVO700/ EVO700C	EVO800 (Standalone)	EVO800 (Redundant)	APC520	APC760C
18.2 Extended Instructions	18.2.19 C Function Support	IOBlockHandle	√	√	√	×	×	√	√
	18.2.12 Special file processing	FileProc	√	√	√	×	×	√	√
18.3 Process Instructions	18.3.2 Advanced Process Control	PFC	×	×	×	×	×	√	×
	18.3.3 Motion Control	MotionControl	√	√	√	×	×	√	√
	18.3.4 Motor Control	ProcessControl	×	×	×	√	√	×	×
	18.3.5 High-Speed I/O Instructions	CmpHCHighSpeedIo	√	√	√	×	×	√	√
18.4 Communication and Diagnostics	18.4.2 Redundant Information	Redundancy	×	×	×	×	√	×	×
		Diagnosis	×	×	×	×	√	×	×
	18.4.4 Local Bus	ExtendIo	√	√	√	×	×	√	√
	18.4.7 Free Serial Port	SerialCom	√	√	√	×	×	√	√
	18.4.9 EtherCAT	EtherCAT	×	√	√	×	×	√	√
	18.4.10 CANBus	CANBus	√	√	√	×	×	√	√
	18.4.11 CIA405	CIA405	√	√	√	×	×	√	√

Category		Library	EVO400	EVO500	EVO7000/ EVO700C	EVO800 (Standalone)	EVO800 (Redundant)	APC520	APC760C
	18.4.15 FINS communication	HC_FINS	√	√	√	×	×	√	√
	18.4.16 Melsec communication	MelsecServer	√	√	√	×	×	√	√
	18.4.17 SFTP Communication	SftpClient	√	√	√	×	×	√	√
	18.4.18 SysPub	SysPub	×	×	×	×	√	×	×
	18.4.19 VarForce	VarForce	×	×	×	×	√	×	×

18.1 Basic Instructions

18.1.1 Bit Logic

18.1.1.1 Instruction List

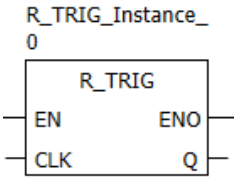
Instruction	FB/FC	Function	Library	Applicable Library Version
18.1.1.2 R_TRIG	FB	Rising edge detection	Standard	Library version 1.0.0.0 and later
18.1.1.3 F_TRIG	FB	Falling edge detection		Library version 1.0.0.0 and later
18.1.1.4 Please	FB	Rising edge output	Util	Library version 1.0.0.0 and later
18.1.1.5 PLF	FB	Falling edge output		Library version 1.0.0.0 and later
18.1.1.6 SR	FB	Prioritize the set value	Standard	Library version 1.0.0.0 and later
18.1.1.7 RS	FB	Prioritize the reset value		Library version 1.0.0.0 and later
18.1.1.8 ALT	FB	Alternate output	Util	Library version 1.0.0.0 and later
18.1.1.9 BOUT	FC	Bit data output		Library version 1.0.0.0 and later
18.1.1.10 BSET	FC	Bit data setting		Library version 1.0.0.0 and later
18.1.1.11 BRST	FC	Bit data reset		Library version 1.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.1.1.12 D_TRIG	FB	D flip-flop	Logic	Library version 1.0.0.0 and later

18.1.1.2 R_TRIG

The R_TRIG instruction is used to output a TRUE signal for only one task cycle when an input signal transitions from FALSE to TRUE (rising edge).

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
R_TRIG	Rising edge detection	FB		R_TRIG_Instance_0(CLK := , Q =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	CLK	Input signal	Follows data type	-	Valid when FALSE changes to TRUE.
Output	Q	Output signal	Follows data type	-	Rising edge detected.

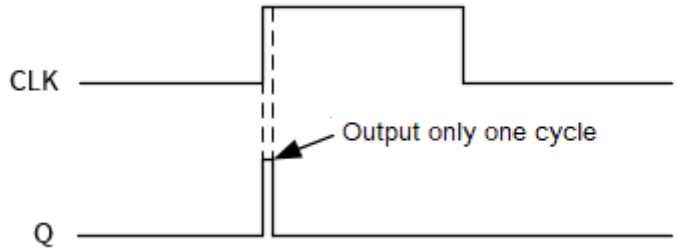
  Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
CLK	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function description**

  When the input signal "CLK" transitions from FALSE to TRUE, the output signal "Q" is set to TRUE for only one task cycle; otherwise, "Q" is FALSE.

- **Sequence diagram**



- **Program example**

  The "bFlag" variable is initially FALSE; when the output signal "Q" is set to TRUE, the "bFlag" variable becomes TRUE.

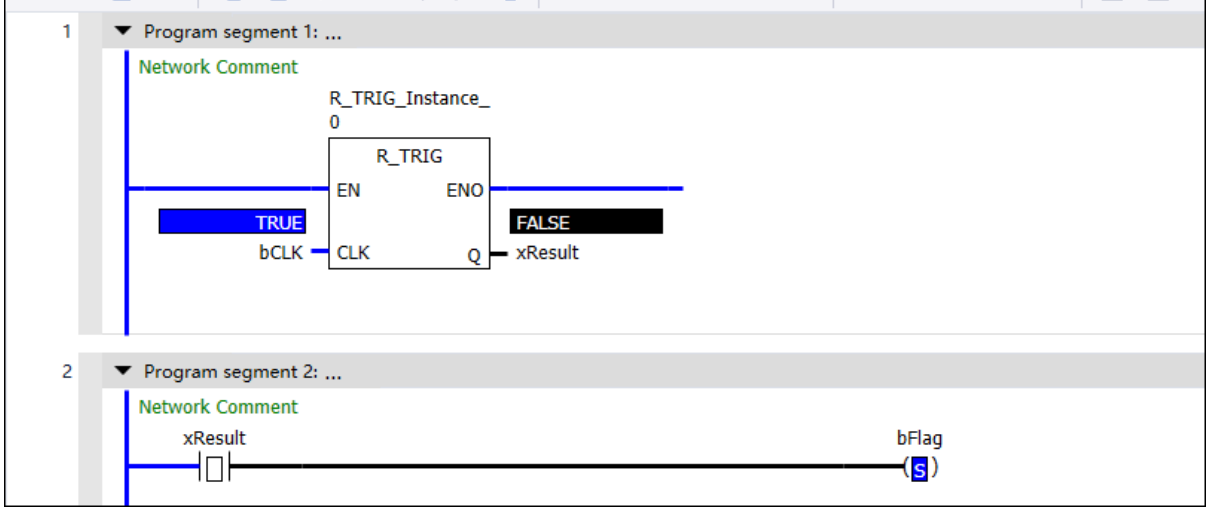
  ST:

	Name	Data type	Address	Value
1	R_TRIG_Instance_0	R_TRIG		
2	bCLK	BOOL		TRUE
3	xResult	BOOL		FALSE
4	bFlag	BOOL		TRUE

```
R_TRIG_Instance_0(
CLK  TRUE  := bCLK  TRUE ,
Q  FALSE  => xResult  FALSE );
IF xResult  FALSE  THEN
    bFlag  TRUE  := TRUE;
END_IFRETURN
```

  LD:

	Name	Data type	Address	Value
1	R_TRIG_Instance_0	R_TRIG		
2	bCLK	BOOL		TRUE
3	xResult	BOOL		FALSE
4	bFlag	BOOL		TRUE



18.1.1.3 F_TRIG

The F_TRIG instruction is used to output a TRUE signal for only one task cycle when an input signal transitions from TRUE to FALSE (falling edge).

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
F_TRIG	Falling edge detection	FB	F_TRIG_Instance_0 F_TRIG EN ENO CLK Q	F_TRIG_Instance_0(CLK := , Q =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	CLK	Input signal	Follows data type	-	Valid when TRUE changes to FALSE.
Output	Q	Output signal	Follows data type	-	Falling edge detected.

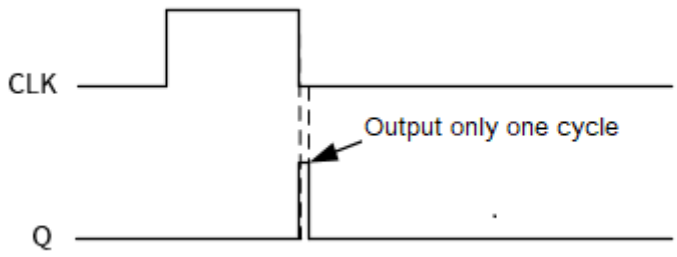
Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
CLK	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function description**

When the input signal "CLK" transitions from TRUE to FALSE, the output signal "Q" is set to TRUE for only one task cycle; otherwise, "Q" is FALSE.

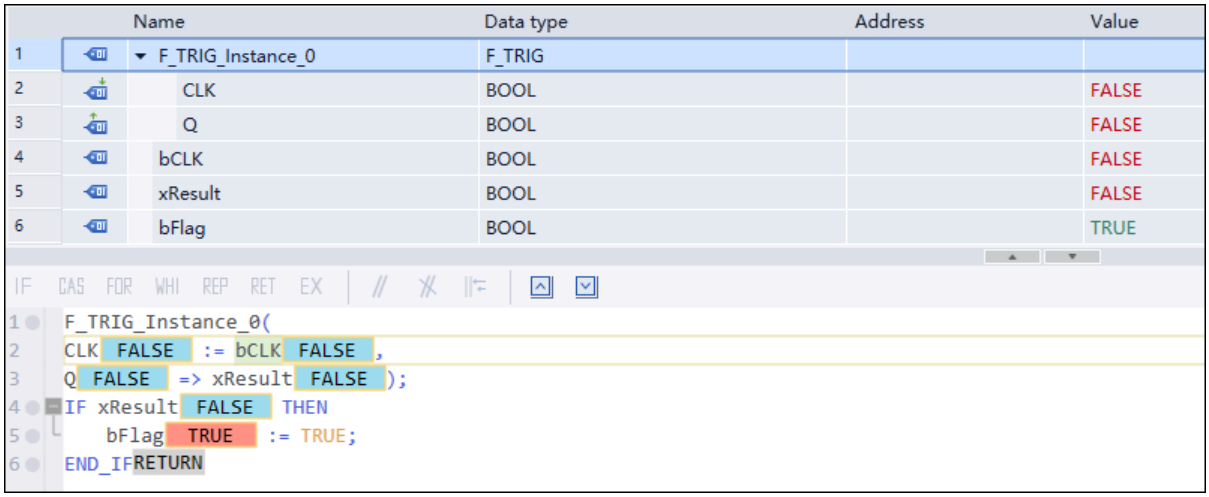
- **Sequence diagram**



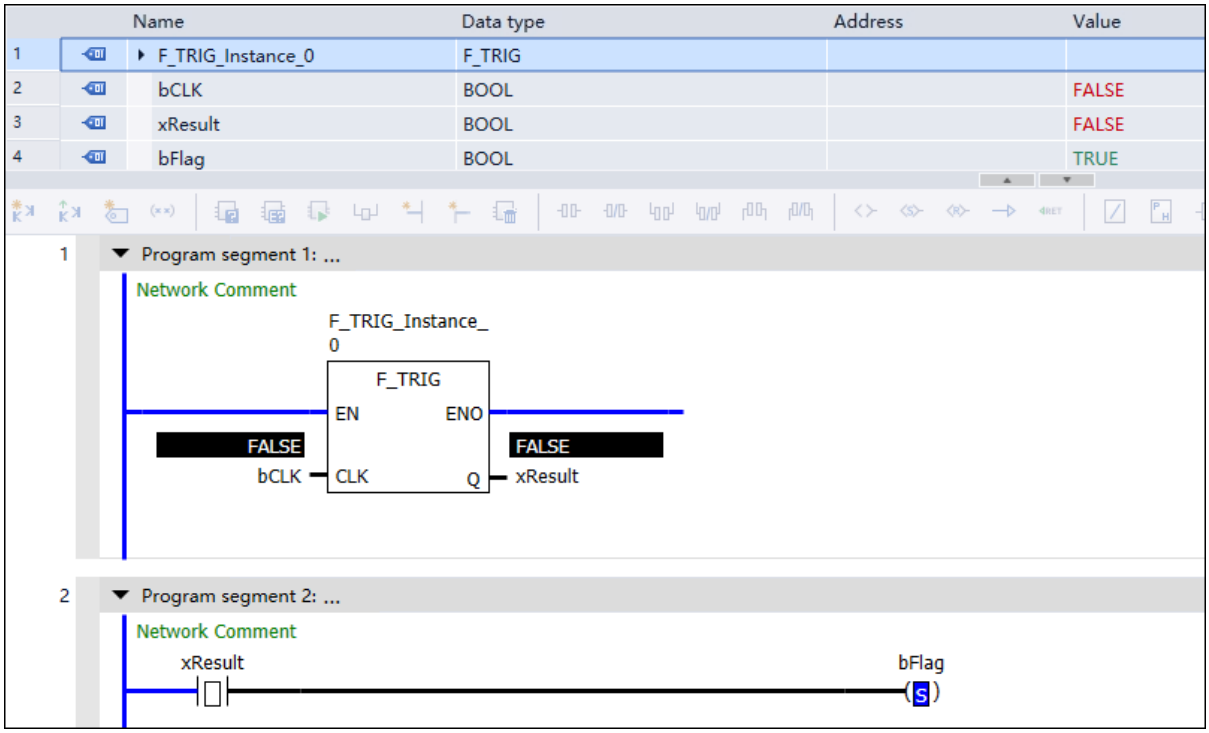
- **Program example**

The "bFlag" variable is initially FALSE; when the output signal "Q" is set to TRUE, the "bFlag" variable becomes TRUE.

ST:



LD:



18.1.1.4 PLS

When the PLS instruction detects a rising edge on the input variable xExecute, xOut will be set to TRUE for one scan cycle and then be set to FALSE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PLS	Rising edge output	FB	PLS_Instance_0 PLS EN ENO xExecute xOut	PLS_Instance_0(xExecute := , xOut =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	xExecute	Input signal	Follows data type.	FALSE	The BOOL value to be detected.
Output	xOut	Output signal	Follows data type.	FALSE	Set to ON if a rising edge on xExecute is detected.

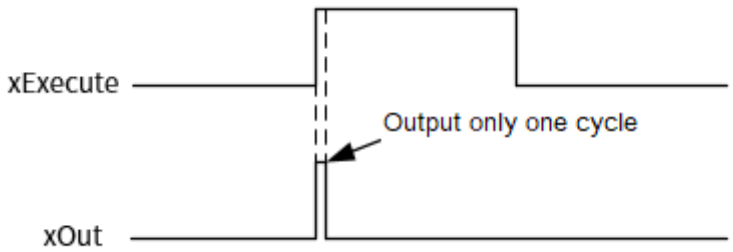
Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
xExecute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xOut	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function description**

  When the PLS instruction detects a rising edge on the input variable xExecute, xOut will be set to TRUE for one scan cycle and then be set to FALSE.

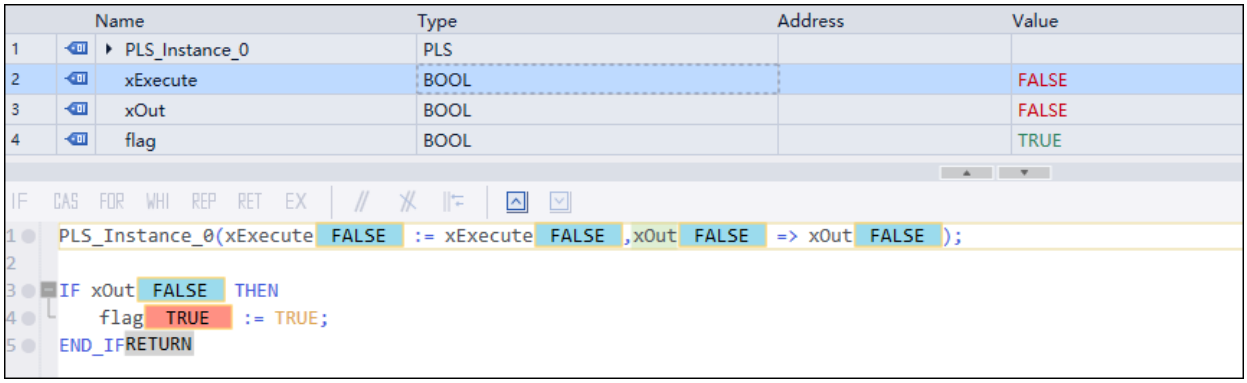
- **Sequence diagram**



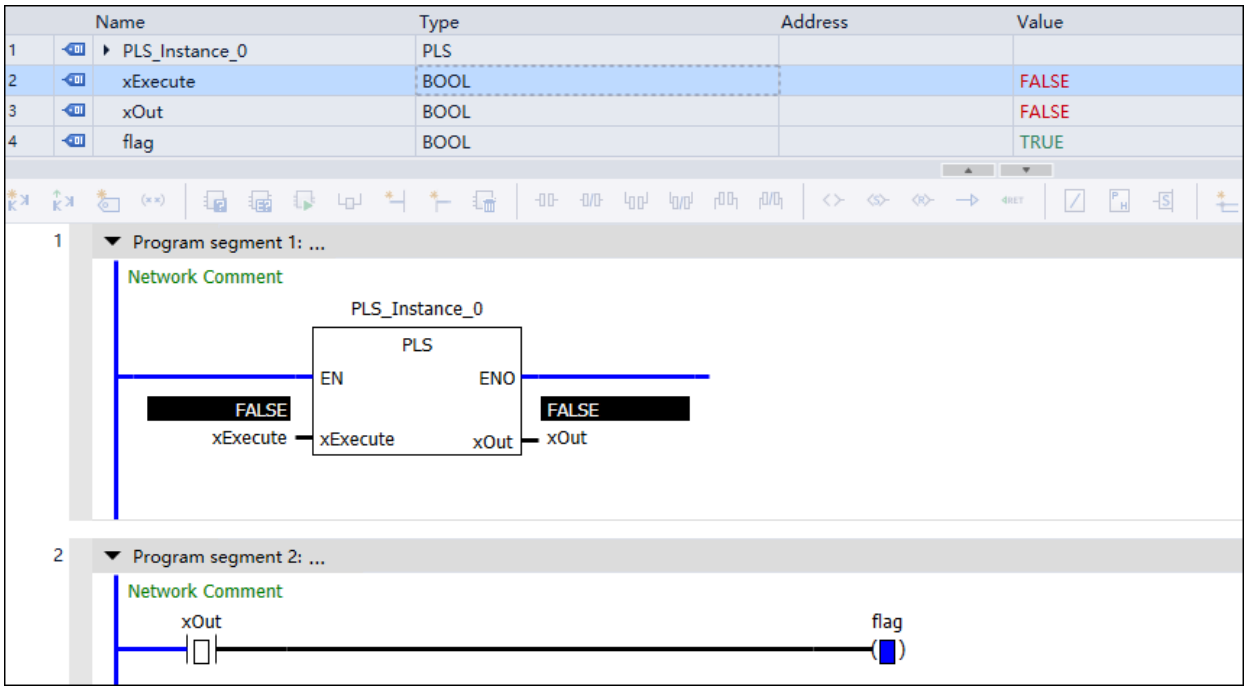
- **Program example**

  The initial value of input variable xExecute is set to FALSE. Manually set the value of xExecute to TRUE to generate a rising edge signal. Since the TRUE status of output variable xOut lasts only for one scan cycle, the change in output value cannot be observed directly; therefore, add a flag variable to assist in observing the status of output variable xOut.

ST:



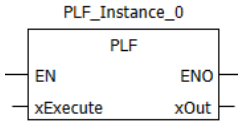
LD:



18.1.1.5 PLF

When the PLF instruction detects a falling edge on the input variable xExecute, xOut will be set to TRUE for one scan cycle and then be set to FALSE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PLF	Falling edge output	FB		PLF_Instance_0(xExecute := , xOut =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	xExecute	Input signal	Follows data type.	FALSE	The BOOL value to be detected.

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	xOut	Output signal	Follows data type.	FALSE	Set to ON if a falling edge on xExecute is detected.

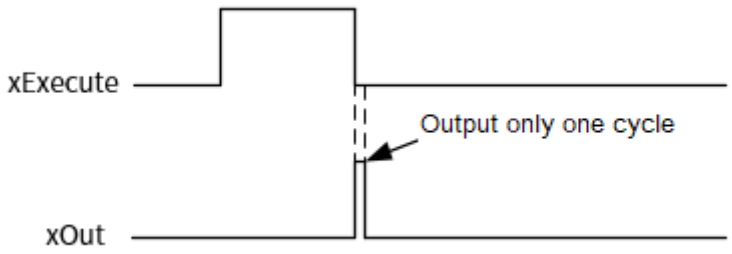
Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
xExecute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xOut	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function description**

  When the PLF instruction detects a falling edge on the input variable xExecute, xOut will be set to TRUE for one scan cycle and then be set to FALSE.

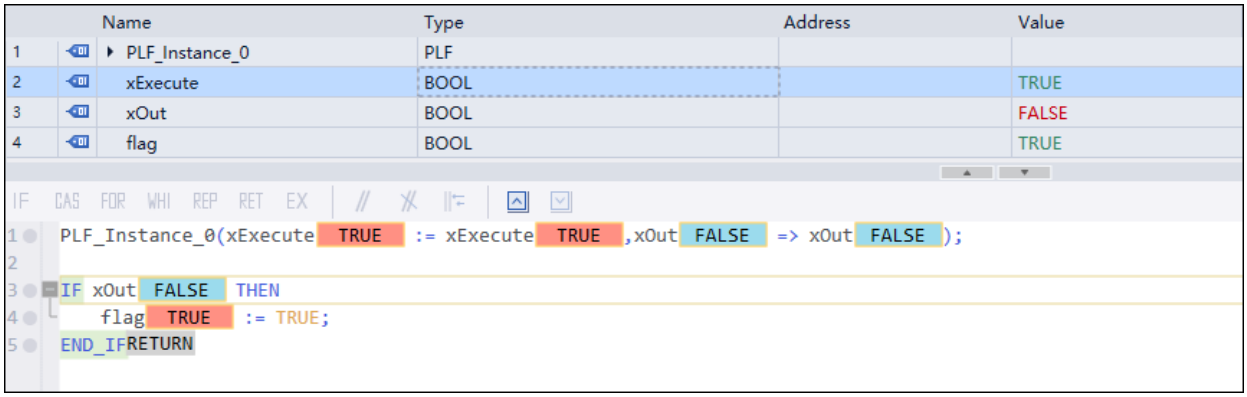
- **Sequence diagram**



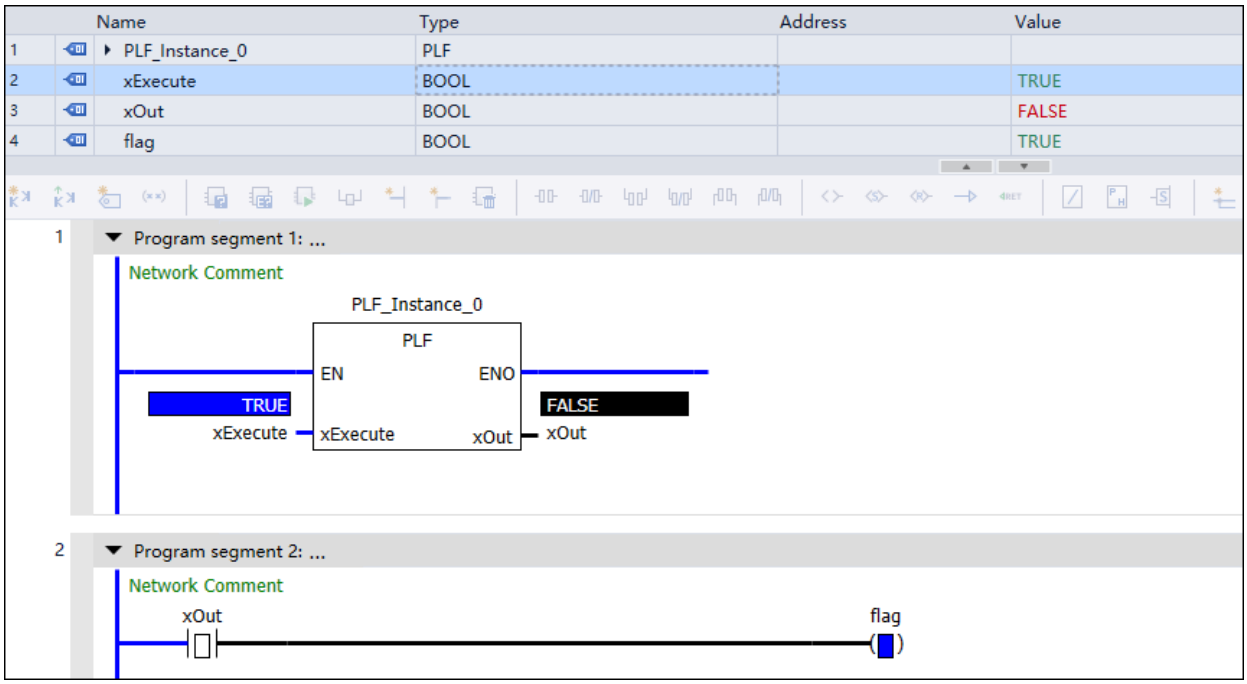
- **Program example**

  The initial value of input variable xExecute is set to TRUE. Manually set the value of xExecute to FALSE to generate a falling edge signal. Since the TRUE status of output variable xOut lasts only for one scan cycle, the change in output value cannot be observed directly; therefore, add a flag variable to assist in observing the status of output variable xOut.

  ST:



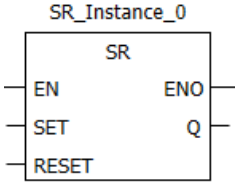
LD:



18.1.1.6 SR

When both the set input signal and the reset input signal are valid, the set input signal has priority.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SR	Prioritize the set value	FB		SR_Instance_0(SET := , RESET :=, Q =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	SET	Set signal	Follows data type	-	Set signal

Input/Output	Variable	Name	Valid Range	Default Value	Description
	RESET	Reset signal	Follows data type	-	Reset signal
Output	Q	Output signal	Follows data type	-	Set signal is prioritized to be output.

Variable type

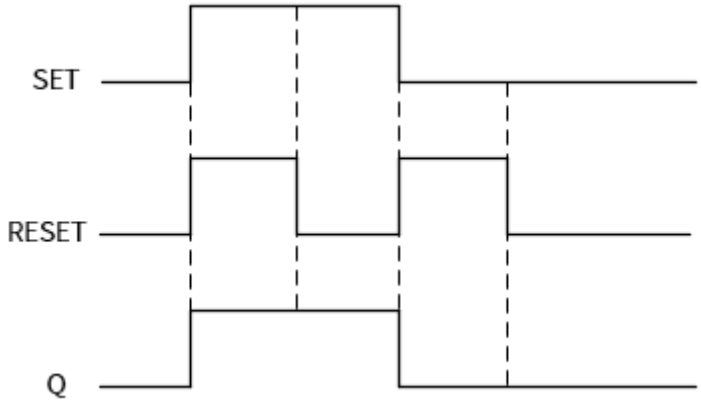
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
SET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function description**

  When both the set input signal SET and the reset input signal RESET are valid, the set input signal SET has priority and becomes the output Q.

SET	RESET	Q
TRUE	TRUE	TRUE
FALSE	TRUE	FALSE
TRUE	FALSE	TRUE
FALSE	FALSE	Remain

- **Sequence diagram**



- **Program example**

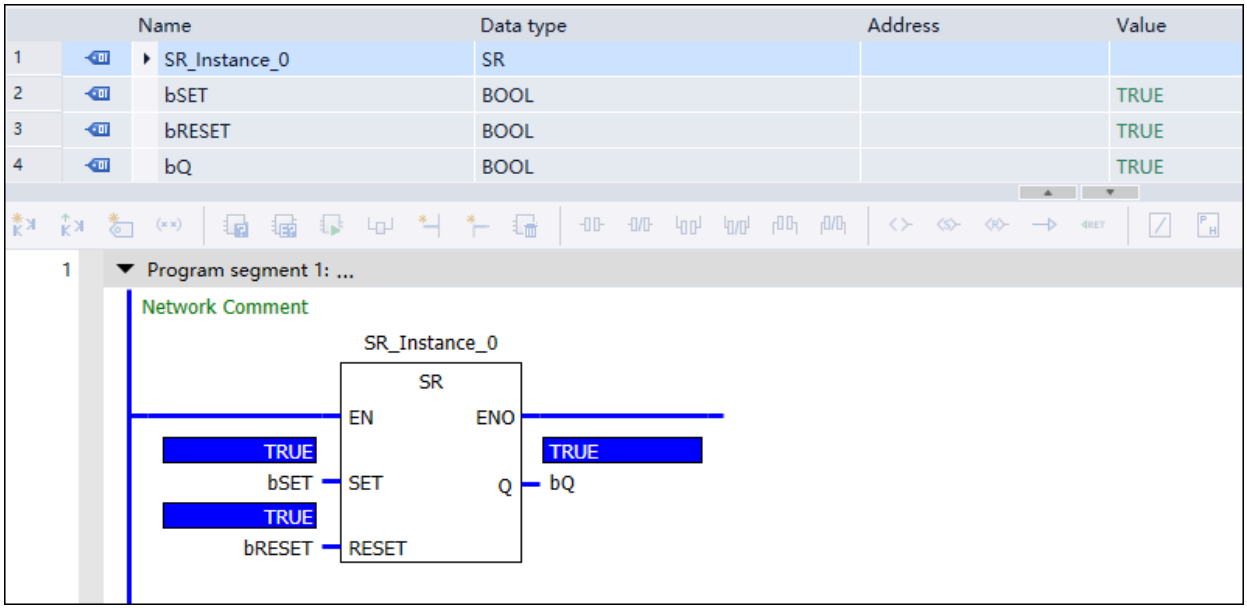
  ST:

		Name	Data type	Address	Value
1		SR_Instance_0	SR		
2		bSET	BOOL		TRUE
3		bRESET	BOOL		TRUE
4		bQ	BOOL		TRUE

```
SR_Instance_0(
SET TRUE := bSET TRUE,
RESET TRUE := bRESET TRUE,
Q TRUE => bQ TRUE);RETURN
```

  LD:

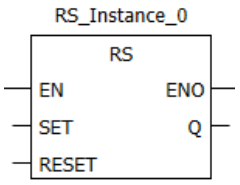
		Name	Data type	Address	Value
1		SR_Instance_0	SR		
2		bSET	BOOL		TRUE
3		bRESET	BOOL		TRUE
4		bQ	BOOL		TRUE



18.1.1.7 RS

When both the set input signal and the reset input signal are valid, the reset input signal has priority.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
RS	Prioritize the reset value	FB		RS_Instance_0(SET := , RESET := , Q =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	SET	Set signal	Follows data type	-	Set signal
	RESET	Reset signal	Follows data type	-	Reset signal
Output	Q	Output signal	Follows data type	-	Reset signal is prioritized to be output.

  Variable type

Variable Name	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
SET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Q1	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

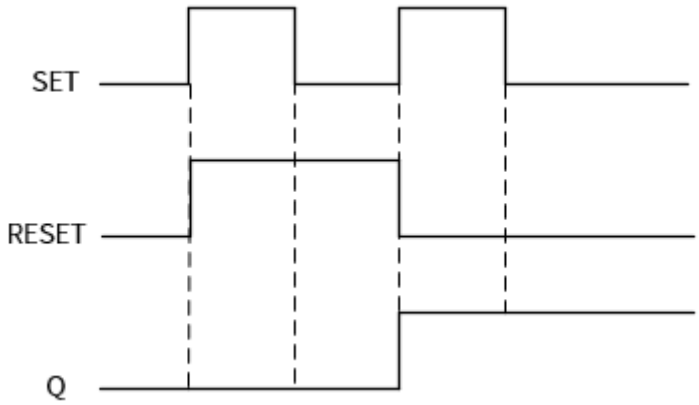
- **Function description**

  When both the set input signal SET and the reset input signal RESET are valid, the reset input signal RESET has priority and becomes the output Q.

SET	RESET	Q
TRUE	TRUE	FALSE
FALSE	TRUE	FALSE

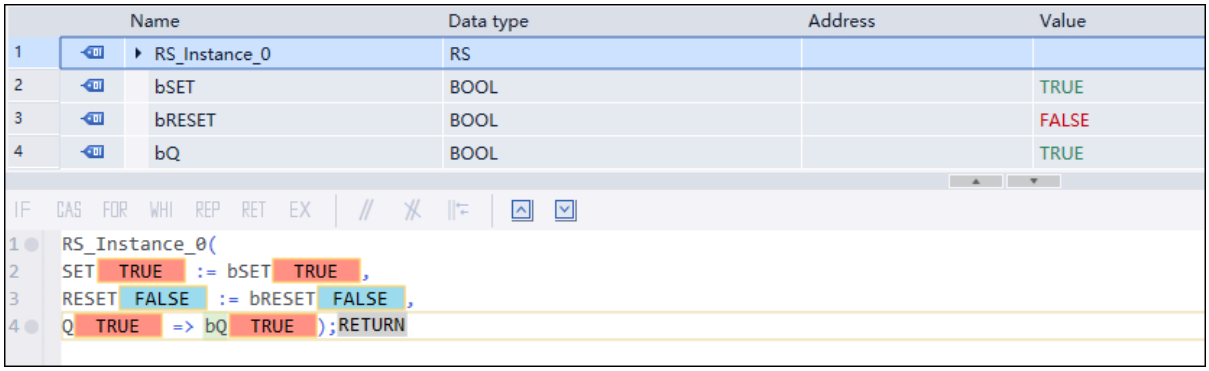
SET	RESET	Q
TRUE	FALSE	TRUE
FALSE	FALSE	Remain

- **Sequence diagram**

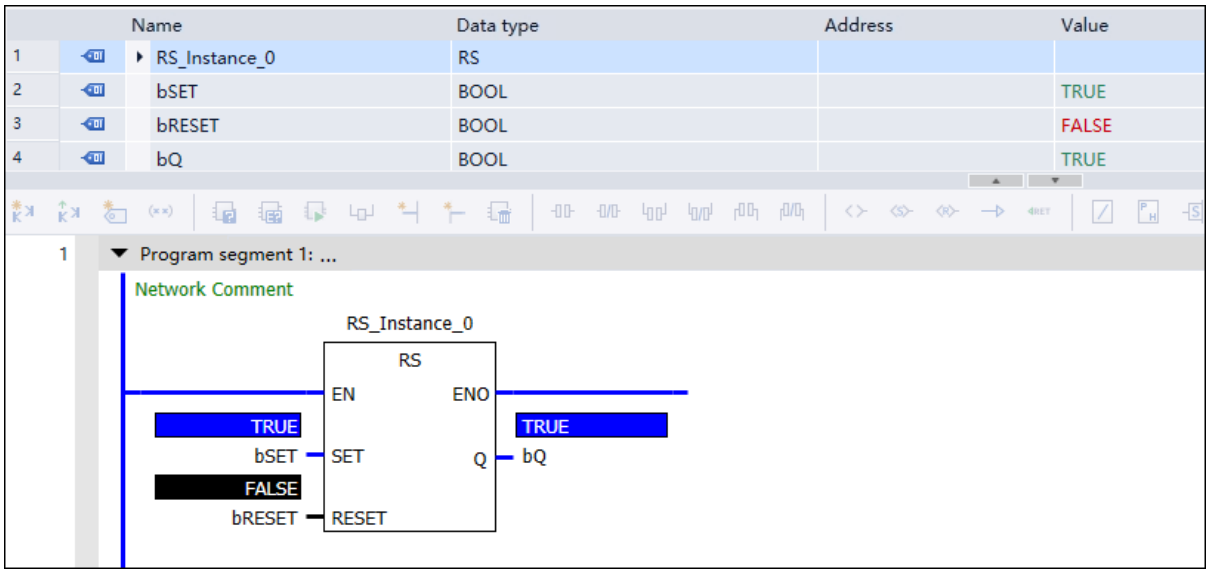


- **Program example**

ST:



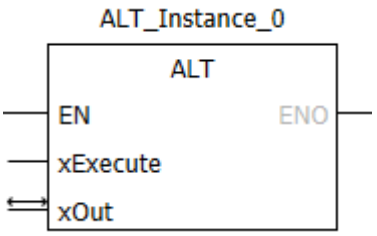
LD:



18.1.1.8 ALT

When the ALT instruction detects a rising edge on xExecute, the status of xOut will be reversed.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ALT	Alternate output	FB		ALT_Instance_0(xExecute := , xOut :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	xExecute	Input signal	Follows data type.	FALSE	The BOOL value to be detected.
Input/Output	xOut	Input/Output signal	Follows data type	FALSE	Reverse the status of xOut if a rising edge on xExecute is detected.

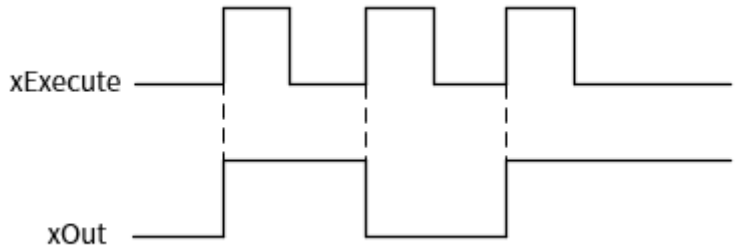
Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
xExecute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xOut	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function description**

When the ALT instruction detects a rising edge on xExecute, the status of xOut will be reversed.

- **Sequence diagram**



- **Program example**

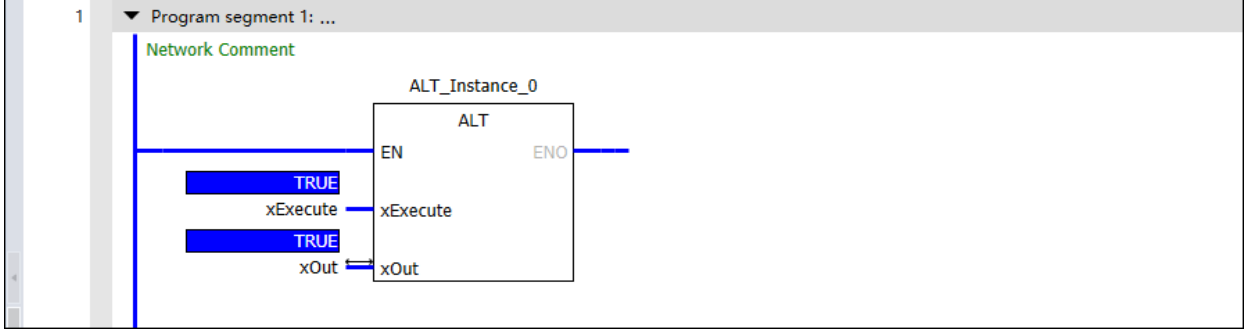
ST:

		Name	Type	Address	Value
1		ALT_Instance_0	ALT		
2		xExecute	BOOL		TRUE
3		xOut	BOOL		TRUE

```
1  ALT_Instance_0(xExecute := xExecute, xOut := xOut);
2  RETURN
```

LD:

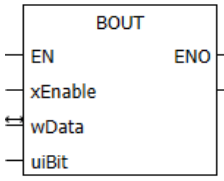
		Name	Type	Address	Value
1		ALT_Instance_0	ALT		
2		xExecute	BOOL		TRUE
3		xOut	BOOL		TRUE



18.1.1.9 BOUT

The BOUT instruction is used to set the status of the specified data bit. If the input variable xEnable is TRUE, the status of uiBit is set to TRUE; if the input variable xEnable is FALSE, the status of uiBit is set to FALSE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BOUT	Bit data output	FC		:= BOUT(xEnable := , wData := , uiBit :=);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	xEnable	Input signal	Follows data type.	FALSE	Level-triggered switch
	uiBit	Input signal	Follows data type	0	The bit to be operated on.
Input/ Output	wData	Input/ Output signal	Follows data type	0	Data to operate
Output	OUT	Output signal	Follows data type.	FALSE	Return value of the function.

  Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
xEnable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
wData	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
uiBit	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

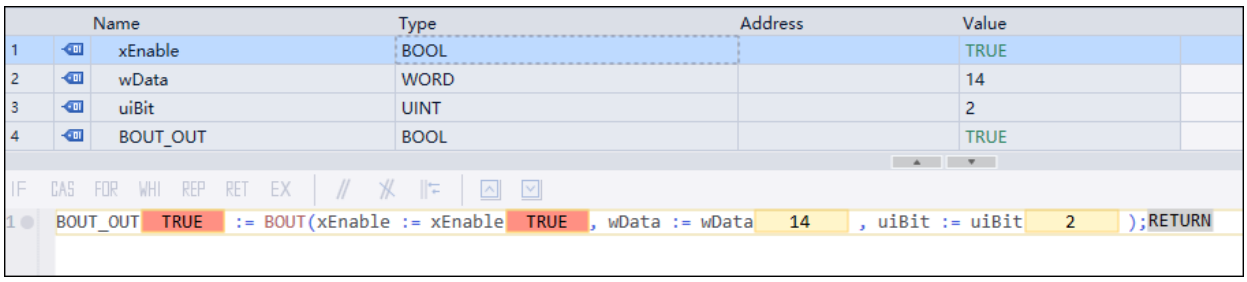
- **Function description**

  The BOUT instruction is used to set the status of the specified data bit. If the input variable xEnable is TRUE, the status of uiBit is set to TRUE; if the input variable xEnable is FALSE, the status of uiBit is set to FALSE.

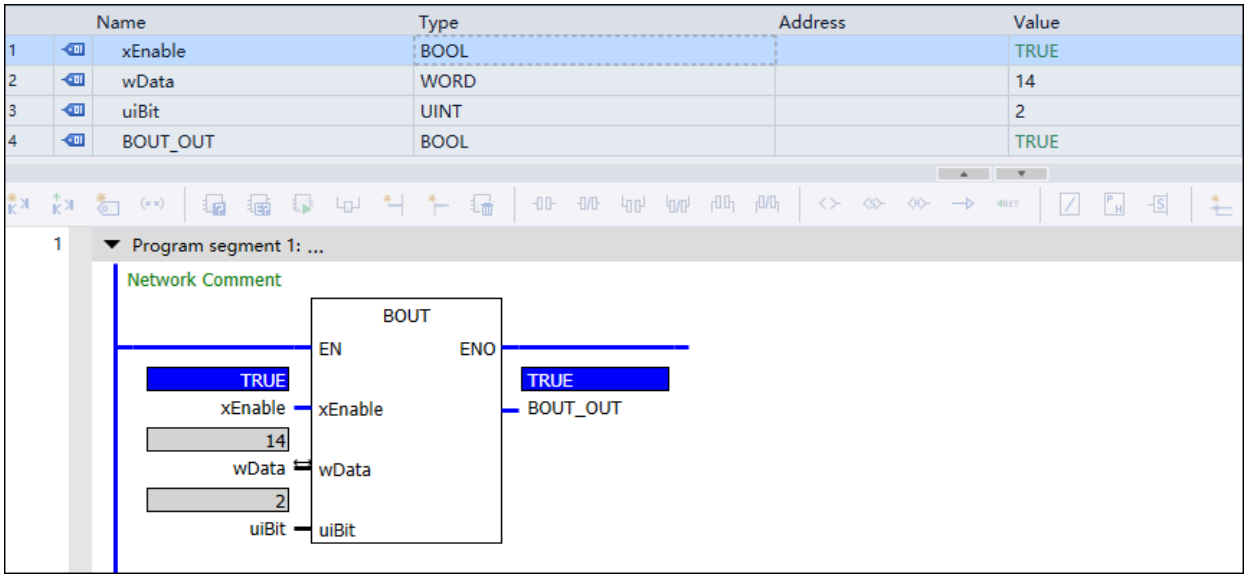
- **Program example**

Set the initial value of wData to 10 (#1010 in binary format). When uiBit = 2 and xEnable = TRUE, bit2 of wData is set, resulting in wData = 14 (2#1110 in binary format).

ST:



LD:

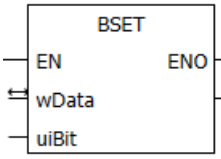


- **Precautions**
  - If uiBit is greater than 16, the instruction execution fails, and the return value is FALSE, with the output variable remaining at its initial value.
  - The instruction returns TRUE upon successful execution, otherwise returns FALSE.

18.1.1.10 BSET

Sets the status of the specified data bit to ON, and the set bit will remain ON. The ON status of the set bit will remain regardless of whether the BSET instruction is driven, and the bit can be reset to OFF using the BRST instruction.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BSET	Bit data setting	FC		:= BSET(wData := , uiBit :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input/Output	wData	Input/Output signal	Follows data type	0	Data to operate
Input	uiBit	Input signal	Follows data type	0	The bit to be operated on.
Output	OUT	Output signal	Follows data type.	FALSE	Return value of the function.

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
wData	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
uiBit	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

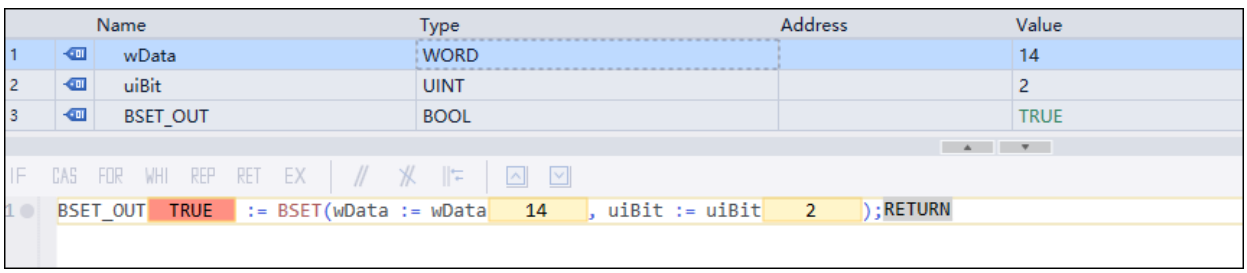
- **Function description**

  Sets the status of the specified data bit to ON, and the set bit will remain ON. The ON status of the set bit will remain regardless of whether the BSET instruction is driven, and the bit can be reset to OFF using the BRST instruction.

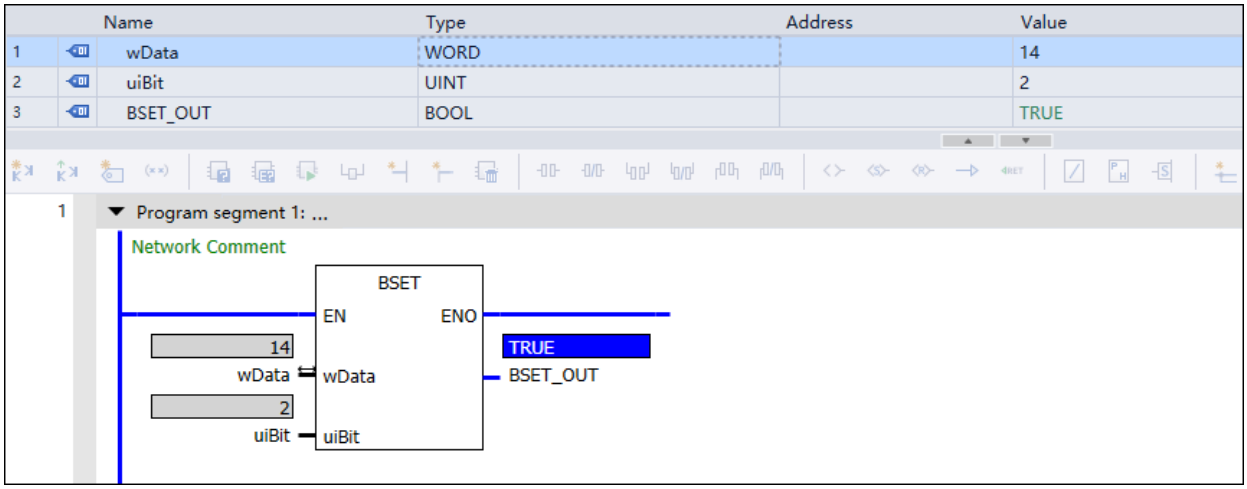
- **Program example**

  Set the initial value of wData to 10 (#1010 in binary format). When uiBit = 2, bit2 of wData is set, resulting in wData = 14 (2#1110 in binary format).

  ST:



  LD:

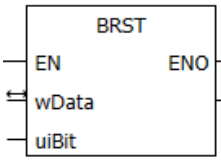


- **Precautions**
  - If uiBit is greater than 15, the instruction execution fails, and the return value is FALSE, with the output variable remaining at its initial value.
  - The instruction returns TRUE upon successful execution, otherwise returns FALSE.

18.1.1.11 BRST

The BRST instruction is used to reset the status of the specified data bit to OFF.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BRST	Bit data reset	FC		:= BRST(wData := , uiBit :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input/Output	wData	Input/Output signal	Follows data type	0	Data to operate
Input	uiBit	Input signal	Follows data type	0	The bit to be operated on.
Output	OUT	Output signal	Follows data type.	FALSE	Return value of the function.

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
wData	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
uiBit	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

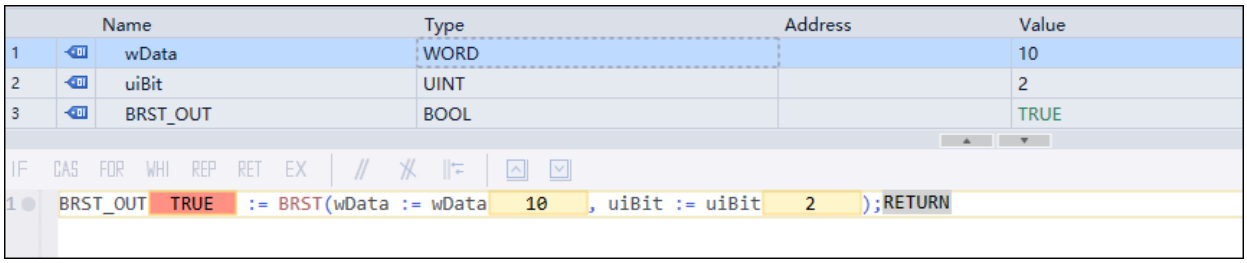
- **Function description**

  Resets the status of the specified data bit to OFF.

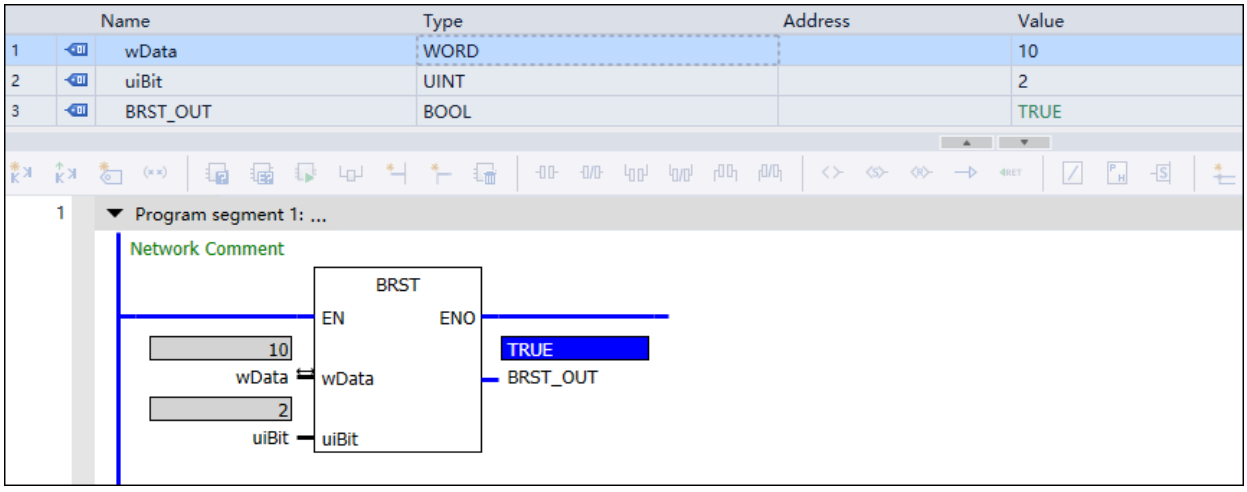
- **Program example**

  Set the initial value of wData to 14 (#1110 in binary format). When uiBit = 2, bit2 of wData is reset, resulting in wData = 10 (2#1010 in binary format).

  ST:



  LD:

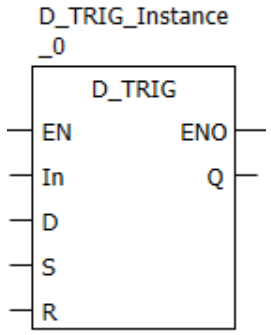


- **Precautions**
  - If uiBit is greater than 15, the instruction execution fails, and the return value is FALSE, with the output variable remaining at its initial value.
  - The instruction returns TRUE upon successful execution, otherwise returns FALSE.

18.1.1.12 D_TRIG

The D_TRIG instruction is used as a D flip-flop.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
D_TRIG	D flip-flop	FB		D_TRIG_Instance_0(In := , D := , S := , R := , Q =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input	Follows data type	-	Input
	D	Flip-flop	Follows data type	-	Flip-flop
	S	Set a bit	Follows data type	-	Sets the signal.
	R	Reset a bit	Follows data type	-	Reset
Output	Q	A flag indicating whether the input changes.	Follows data type	-	A flag indicating whether the input changes.

Data type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
D	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
S	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
R	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

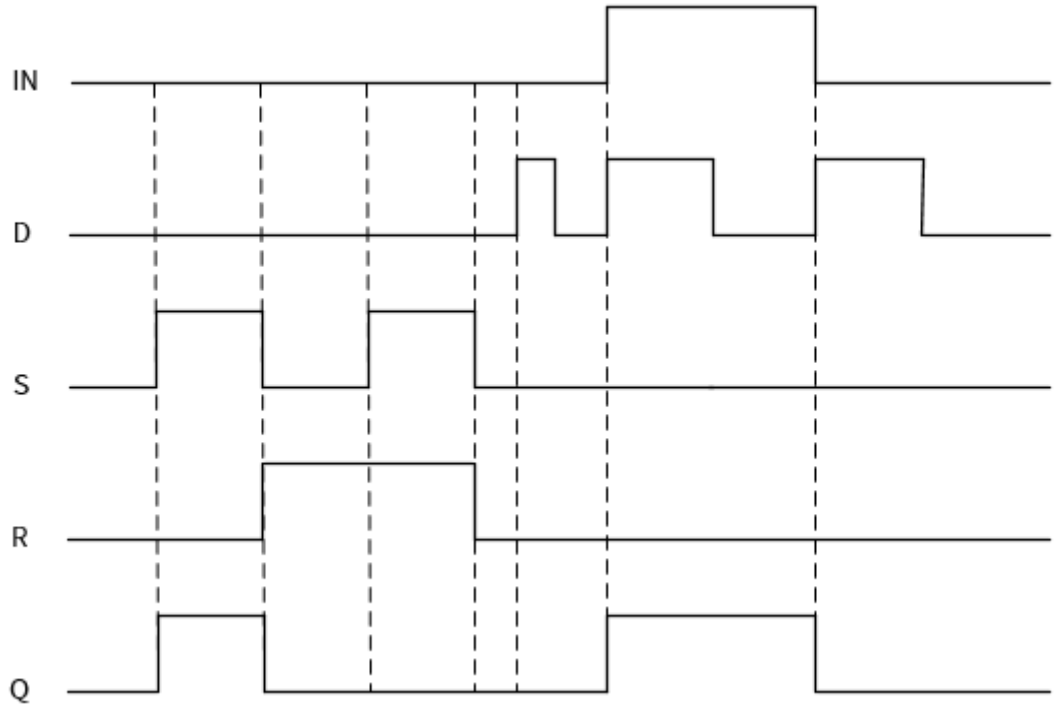
- **Function description**

  When input parameters are known, the output Q includes the following five cases as shown in the table:

S	R	D	In	Q
1	0	*	*	1
0	1	*	*	0
1	1	*	*	0
0	0	0 -> 1	In	In
0	0	Others	In	Q(n-1)

  Q(n-1) represents the output value Q from the previous cycle.

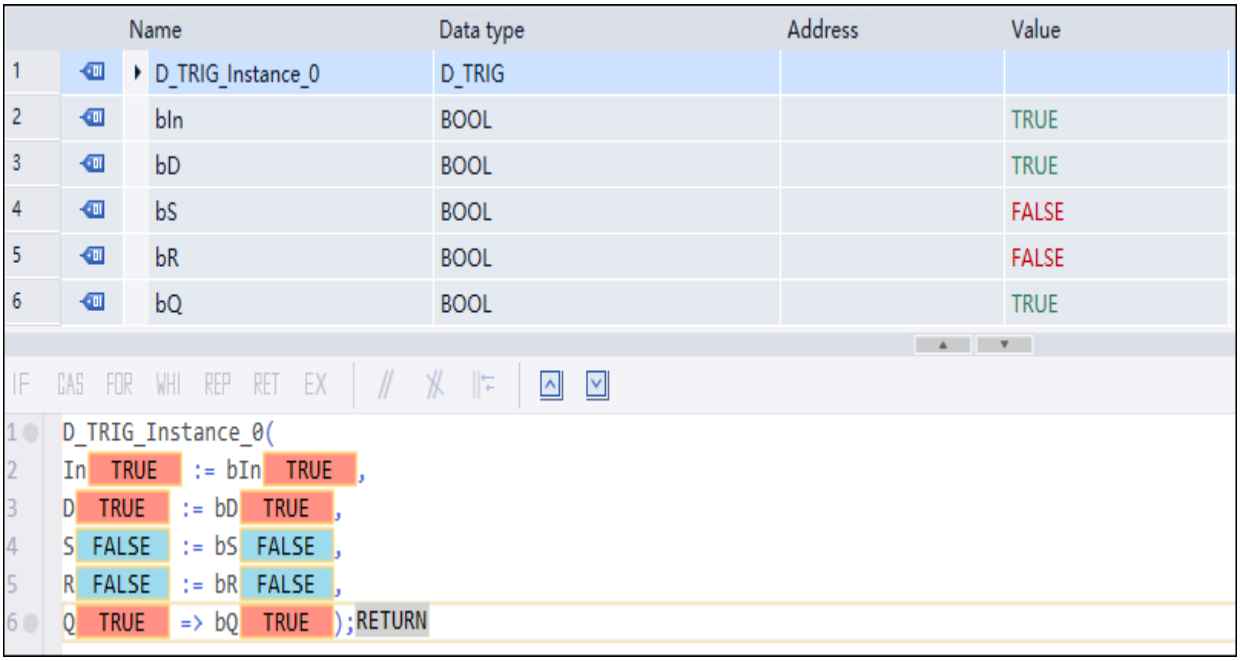
- **Sequence diagram**



- **Program example**

  When input variable In is TRUE, input variable S and R are FALSE, and input variable D changes from FALSE to TRUE, the output variable Q becomes TRUE.

  ST:



  LD:



18.1.2 Data Processing

18.1.2.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.1.2.2 BON	FC	Gets the status of the specified bit in the data.	Util	Library version 1.0.0.0 and later
18.1.2.3 SUM	FC	Total number of ON bits		Library version 1.0.0.0 and later
18.1.2.4 Swap	FC	Byte swap		Library version 1.0.0.0 and later
18.1.2.5 SWAP16	FC	Byte swap		Library version 1.0.0.0 and later
18.1.2.6 SWAP32	FC	Byte swap		Library version 1.0.0.0 and later
18.1.2.7 XCH	FC	Data exchange		Library version 1.0.0.0 and later
18.1.2.8 BitCnt	FC	Bit counter	OMUtils	Library version 1.0.0.0 and later
18.1.2.9 Clear	FC	Initialization		Library version 1.0.0.0 and later
18.1.2.10 Byte_To_HexString	FC	Byte-to-hexadecimal text string conversion	Util	Library version 1.2.0.0 and later
18.1.2.11 Word_To_HexString	FC	Word-to-hexadecimal text string conversion		Library version 1.2.0.0 and later
18.1.2.12 Dword_To_HexString	FC	Doubleword-to-hexadecimal text string conversion		Library version 1.2.0.0 and later
18.1.2.13 ColmToLine_BYTE	FC	Specified position combination output of array elements	OMUtils	Library version 1.0.0.0 and later
18.1.2.14 ColmToLine_WORD	FC	Specified position combination output of array elements		Library version 1.0.0.0 and later
18.1.2.15 ColmToLine_DWORD	FC	Specified position combination output of array elements		Library version 1.0.0.0 and later
18.1.2.16 ColmToLine_LWORD	FC	Specified position combination output of array elements		Library version 1.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.1.2.17 CopyRealToNum	FC	Bit pattern copy (real number to signed integer)		Library version 1.0.0.0 and later
18.1.2.18 CopyLrealToNum	FC	Bit pattern copy (long real number to signed integer)		Library version 1.0.0.0 and later
18.1.2.19 CopyByteToNum	FC	Bit pattern copy (BYTE-type bit string to signed integer)		Library version 1.0.0.0 and later
18.1.2.20 CopyWordToNum	FC	Bit pattern copy (WORD-type bit string to signed integer)		Library version 1.0.0.0 and later
18.1.2.21 CopyDwordToNum	FC	Bit pattern copy (DWORD-type bit string to signed integer)		Library version 1.0.0.0 and later
18.1.2.22 CopyLwordToNum	FC	Bit pattern copy (LWORD-type bit string to signed integer)		Library version 1.0.0.0 and later
18.1.2.23 CopyDwordToReal	FC	Bit pattern copy (DWORD-type bit string to real number)		Library version 1.0.0.0 and later
18.1.2.24 CopyLwordToLreal	FC	Bit pattern copy (LWORD-type bit string to long real number)		Library version 1.0.0.0 and later
18.1.2.25 CopyNumToReal	FC	Bit pattern copy (integer to real number)		Library version 1.0.0.0 and later
18.1.2.26 CopyNumToLreal	FC	Bit pattern copy (integer to long real number)		Library version 1.0.0.0 and later
18.1.2.27 CopyNumToByte	FC	Bit pattern copy (integer to BYTE-type bit string)		Library version 1.0.0.0 and later
18.1.2.28 CopyNumToWord	FC	Bit pattern copy (integer to WORD-type bit string)		Library version 1.0.0.0 and later
18.1.2.29 CopyNumToDword	FC	Bit pattern copy (integer to DWORD-type bit string)		Library version 1.0.0.0 and later
18.1.2.30 CopyNumToLword	FC	Bit pattern copy (integer to LWORD-type bit string)		Library version 1.0.0.0 and later
18.1.2.31 CopyRealToDword	FC	Bit pattern copy (real number to DWORD-type bit string)		Library version 1.0.0.0 and later
18.1.2.32 CopyLrealToLword	FC	Bit pattern copy (long real number to LWORD-type bit string)		Library version 1.0.0.0 and later
18.1.2.33 Dispart8Bit	FC	Data separation by byte		Library version 1.0.0.0 and later

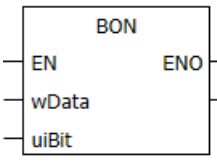
Instruction	FB/FC	Function	Library	Applicable Library Version
18.1.2.34 DispartDigit	FC	Data separation by digit		Library version 1.0.0.0 and later
18.1.2.35 DispartReal	FC	Separates mantissa and exponent		Library version 1.0.0.0 and later
18.1.2.36 Exchange	FC	Variable exchange		Library version 1.0.0.0 and later
18.1.2.37 FixNumToString	FC	Fixed-decimal number-to-string conversion		Library version 1.0.0.0 and later
18.1.2.38 GRAY	FC	Gray Code Conversion		Library version 1.0.0.0 and later
18.1.2.39 GrayToBin_BYTE	FC	Gray code-to-bit string conversion		Library version 1.0.0.0 and later
18.1.2.40 GrayToBin_WORD	FC	Gray code-to-bit string conversion		Library version 1.0.0.0 and later
18.1.2.41 GrayToBin_DWORD	FC	Gray code-to-bit string conversion		Library version 1.0.0.0 and later
18.1.2.42 GrayToBin_LWORD	FC	Gray code-to-bit string conversion		Library version 1.0.0.0 and later
18.1.2.43 LineToColm	FC	Line-to-column conversion		Library version 1.0.0.0 and later
18.1.2.44 MoveDigit	FC	Digit transfer		Library version 1.0.0.0 and later
18.1.2.45 ReadNbit_BYTE	FC	N-bit read group		Library version 1.0.0.0 and later
18.1.2.46 ReadNbit_WORD	FC	N-bit read group		Library version 1.0.0.0 and later
18.1.2.47 ReadNbit_DWORD	FC	N-bit read group		Library version 1.0.0.0 and later
18.1.2.48 ReadNbit_LWORD	FC	N-bit read group		Library version 1.0.0.0 and later
18.1.2.49 SetBlock	FC	Block setting		Library version 1.0.0.0 and later
18.1.2.50 StringToFixNum	FC	Conversion from a string to a fixed-decimal number		Library version 1.0.0.0 and later
18.1.2.51 TransBits	FC	Multiple bit transfer		Library version 1.0.0.0 and later
18.1.2.52 Unite8Bit_BYTE	FC	Byte data join group		Library version 1.0.0.0 and later
18.1.2.53 Unite8Bit_WORD	FC	Byte data join group		Library version 1.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.1.2.54 Unite8Bit_DWORD	FC	Byte data join group		Library version 1.0.0.0 and later
18.1.2.55 Unite8Bit_LWORD	FC	Byte data join group		Library version 1.0.0.0 and later
18.1.2.56 UniteDigit_BYTE	FC	4-bit join group		Library version 1.0.0.0 and later
18.1.2.57 UniteDigit_WORD	FC	4-bit join group		Library version 1.0.0.0 and later
18.1.2.58 UniteDigit_DWORD	FC	4-bit join group		Library version 1.0.0.0 and later
18.1.2.59 UniteDigit_LWORD	FC	4-bit join group		Library version 1.0.0.0 and later
18.1.2.60 UniteReal	FC	Combine mantissa and exponent		Library version 1.0.0.0 and later
18.1.2.61 WriteNbit_BYTE	FC	N-bit write group		Library version 1.0.0.0 and later
18.1.2.62 WriteNbit_WORD	FC	N-bit write group		Library version 1.0.0.0 and later
18.1.2.63 WriteNbit_DWORD	FC	N-bit write group		Library version 1.0.0.0 and later
18.1.2.64 WriteNbit_LWORD	FC	N-bit write group		Library version 1.0.0.0 and later
18.1.2.65 StringSum	FC	String sum calculation		Library version 1.0.0.0 and later

18.1.2.2 BON

Checks the status of the uiBit-th bit in data wData, stores the result in BON and outputs from the right.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BON	Gets the status of the specified bit in the data.	FC		:= BON(wData := , uiBit :=);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	wData	Data source	Follows data type	0	The data (in decimal format) to be checked.
	uiBit	Bit to check	Follows data type	0	The bit (in binary format) of the data to be checked.
Output	Out	Judgment result	Follows data type	FALSE	The status of the uiBit-th bit in wData.

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
wData	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
uiBit	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

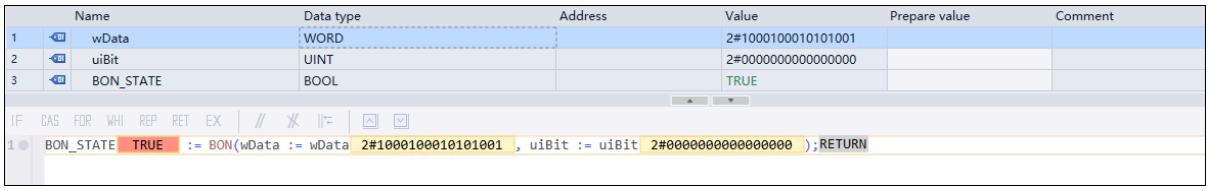
- **Function description**

  Checks the status of the uiBit-th bit in data wData, stores the result in BON and outputs from the right.

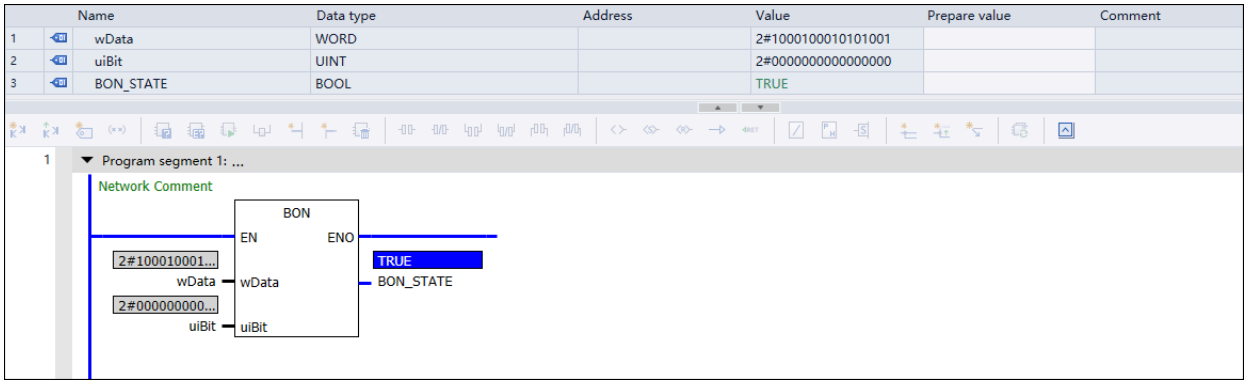
- **Program example**

  Input the data to be checked (wData), input the bit number to be checked (uiBit), and the corresponding bit status BON_STATE will be output. The initial value of wData is 2#1000100010101001.

  ST:



  LD:



- **Precautions**
  - If uiBit is greater than 16, the return value is FALSE, with the output variable remaining at its initial value.
  - The status of the uiBit-th bit in data wData is returned.

18.1.2.3 SUM

Counts the number of bits set to 1 in binary-format data stores the result in SUM and outputs it from the right.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SUM	Total number of ON bits	FC	SUM EN ENO wData	:= SUM(wData :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	wData	Data source	Follows data type	0	The data (in decimal format) to be checked.
Output	Out	Judgment result	Follows data type	0	Stores the number of bits set to 1 in wData (in binary format).

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
wData	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

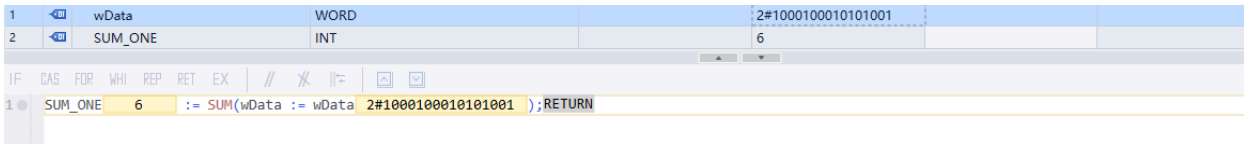
- **Function description**

  Counts the number of bits set to 1 in binary-format data stores the result in SUM and outputs it from the right.

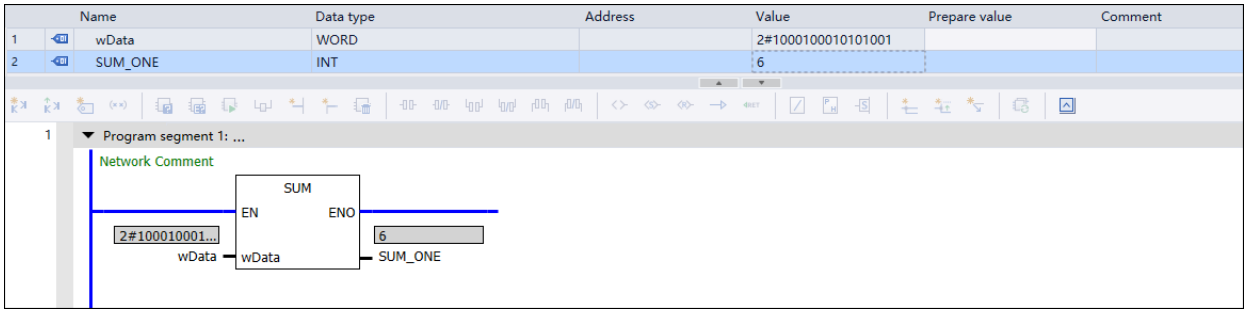
- **Program example**

  Input the data wData to be checked, count the number of bits set to 1 in binary-format data, and output the result as SUM_ONE. The initial value of wData is 2#1000100010101001.

  ST:



  LD:

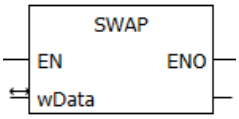


- **Precautions**
  - Returns the number of bits set to 1 in binary-format data wData.

18.1.2.4 SWAP

Swaps the high 8 bits and low 8 bits of a 16-bit data type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SWAP	High-low byte swap	FC		:= SWAP(wData :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input/Output	wData	Input data	Follows data type	0	Data to operate
Output	Out	Execution result	Follows data type	FALSE	The result of the instruction.

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
wData	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

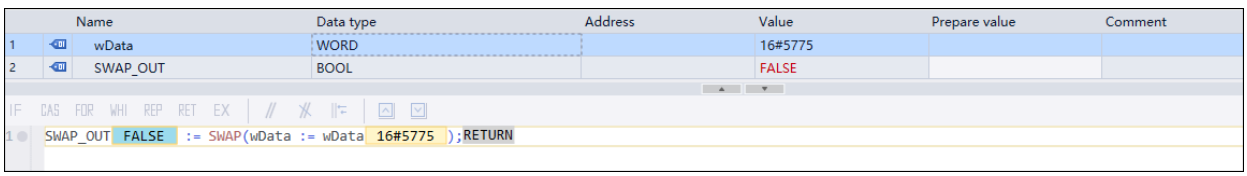
- **Function description**

  Swaps the high 8 bits and low 8 bits of a 16-bit data type, and returns the result of the instruction.

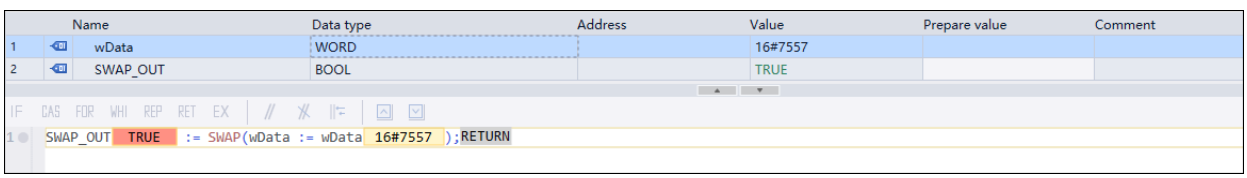
- **Program example**

  ST:

  Before execution:

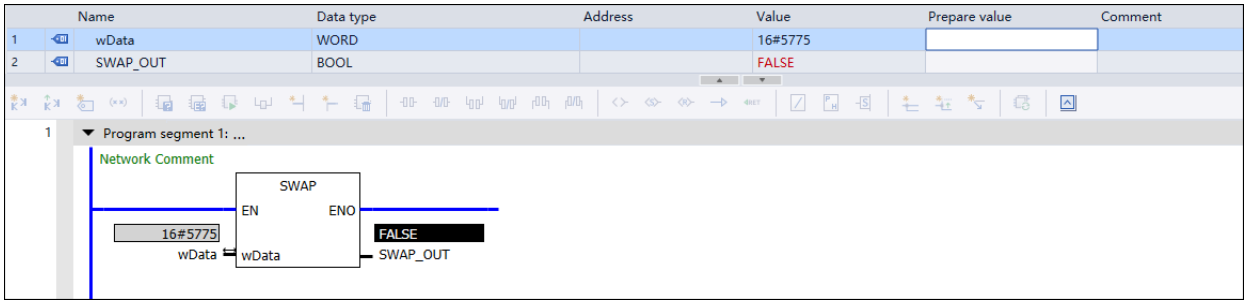


  After execution:

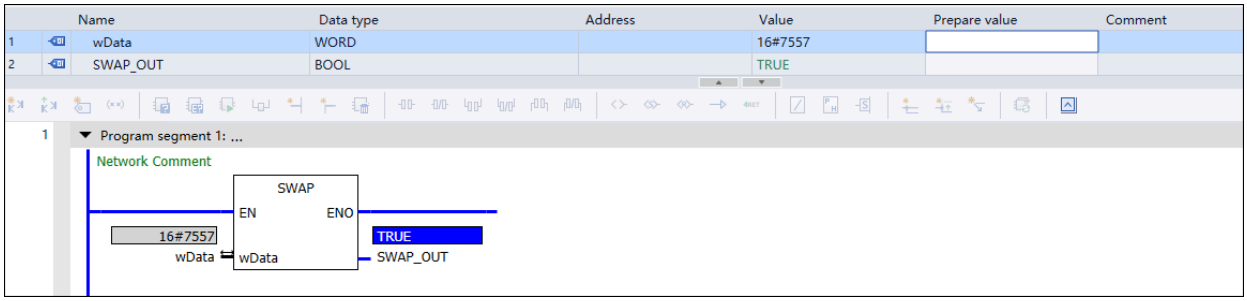


LD:

Before execution:



After execution:

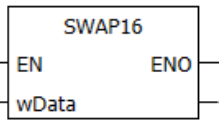


- **Precautions**
  - Returns TRUE if the instruction executes successfully, otherwise returns FALSE.

18.1.2.5 SWAP16

Swaps the high 8 bits and low 8 bits of a 16-bit data type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SWAP16	Byte swap	FC		:= SWAP16(wData :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	wData	Input data	Follows data type	0	Data to operate
Output	Out	Swap result	Follows data type	0	The swap result after instruction execution.

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
wData	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

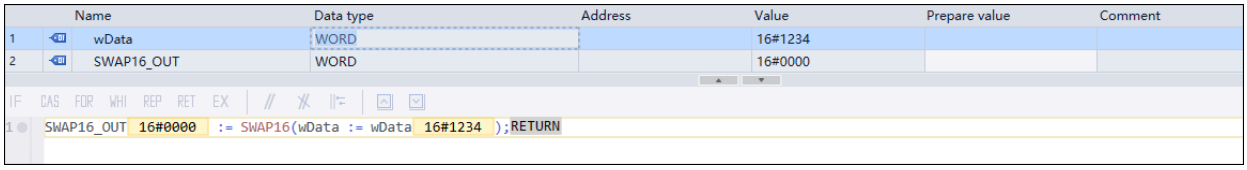
- **Function description**

  Swaps the high 8 bits and low 8 bits of a 16-bit data type, and returns the swap result.

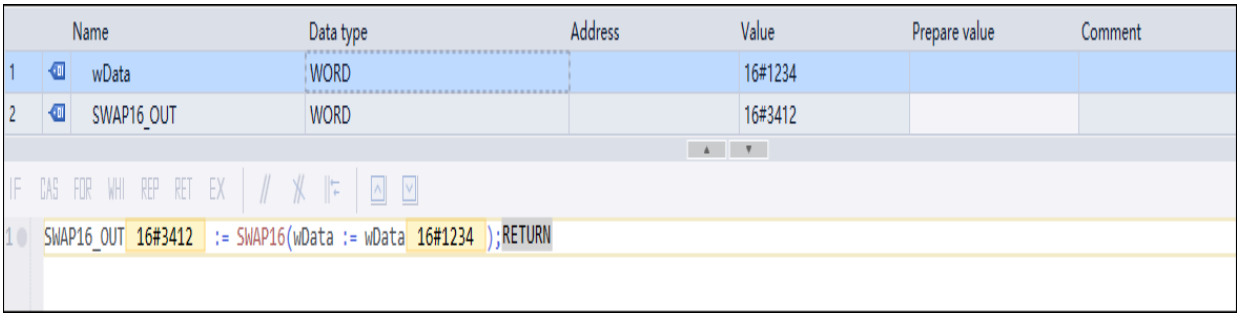
- **Program example**

  ST:

  Before execution:

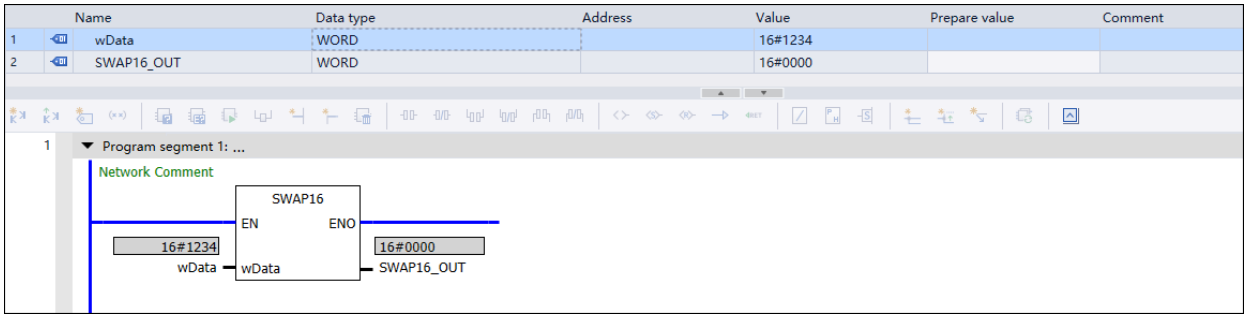


  After execution:

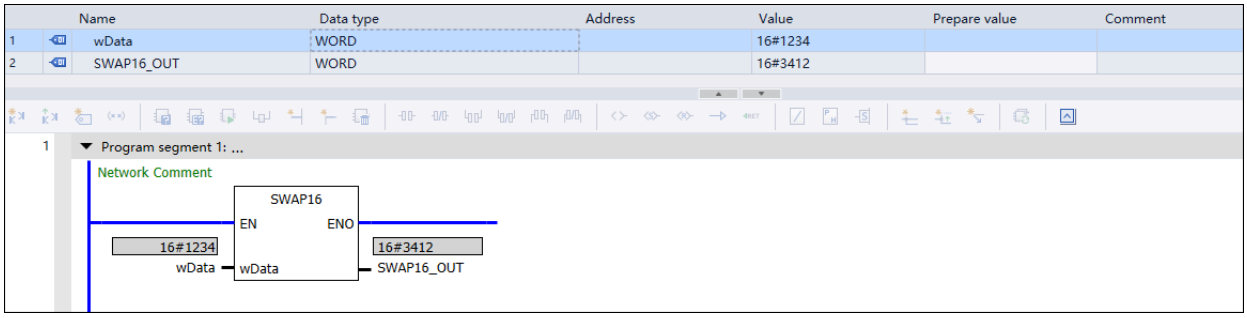


  LD:

  Before execution:



  After execution:



- **Precautions**
  - Returns the 16-bit data whose high 8 bits and low 8 bits are swapped.

18.1.2.6 SWAP32

Swaps the high 16 bits and low 16 bits of a 32-bit data type, and also swaps the high 8 bits and low 8 bits within each 16-bit segment.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SWAP32	Byte swap	FC	SWAP32 EN ENO wData	:= SWAP32(wData :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	wData	Input data	Follows data type	0	Data to operate
Output	Out	Swap result	Follows data type	0	The swap result after instruction execution.

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
wData	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

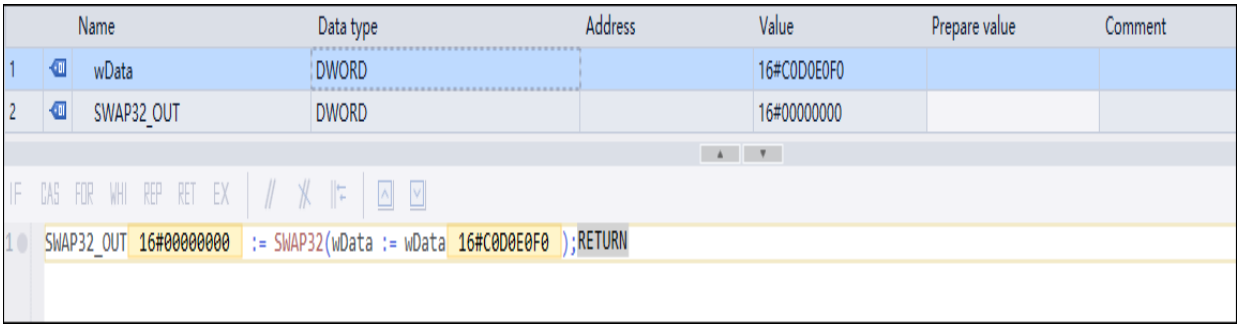
- **Function description**

Swaps the high 16 bits and low 16 bits of a 32-bit data type, and also swaps the high 8 bits and low 8 bits within each 16-bit segment. The swap result is returned.

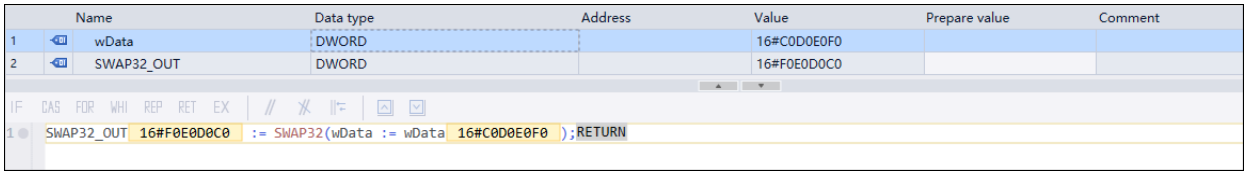
- **Program example**

  ST:

  Before execution:

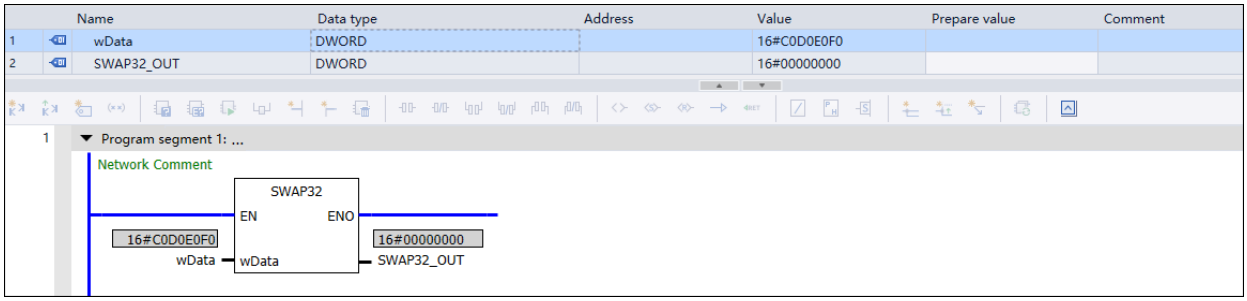


  After execution:

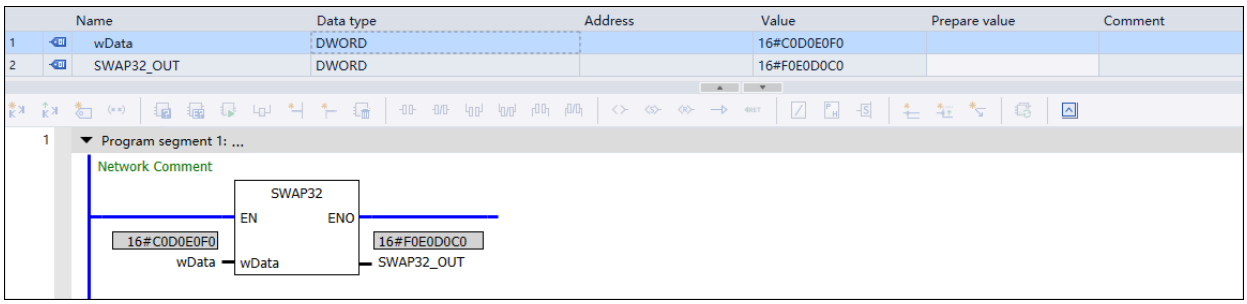


  LD:

  Before execution:



  After execution:

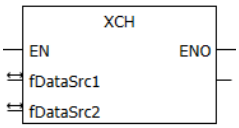


- **Precautions**
  - Returns the swap result.

18.1.2.7 XCH

Swaps two floating-point values.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
XCH	Data exchange	FC		:= XCH(fDataSrc1 := , fDataSrc2 :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input/Output	fDataSrc1	Data 1 to swap	Follows data type	0	The data to be swapped.
	fDataSrc2	Data 2 to swap	Follows data type	0	The data to be swapped.
Output	Out	Execution result	Follows data type	0	The result of the instruction execution.

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
fDataSrc1	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
fDataSrc2	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

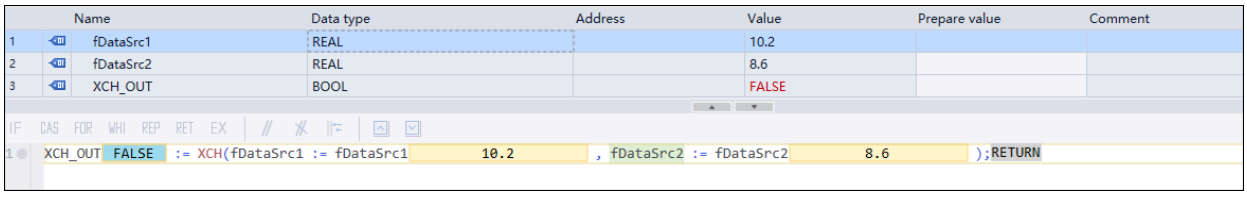
- **Function description**

  Swaps two floating-point values and returns the instruction execution result.

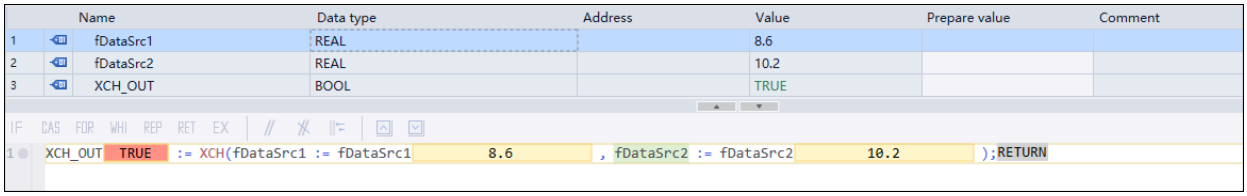
- **Program example**

  ST:

  Before execution:

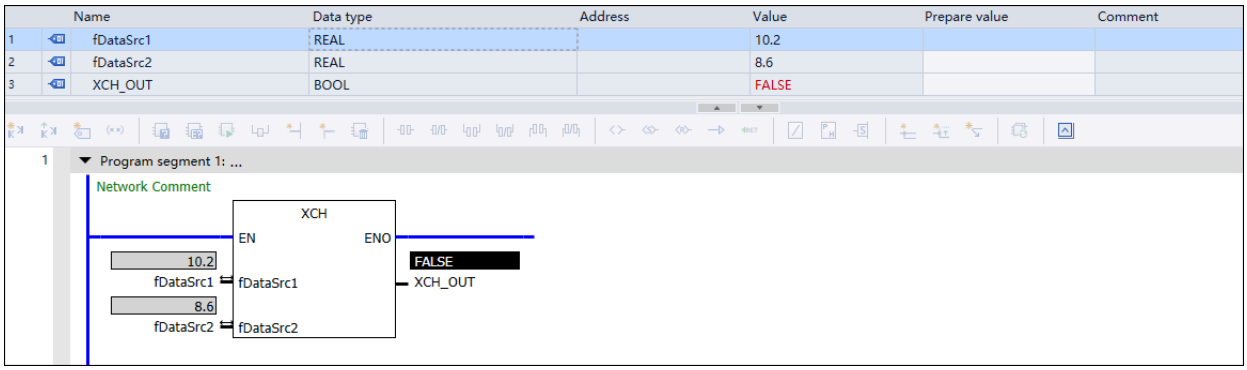


After execution:

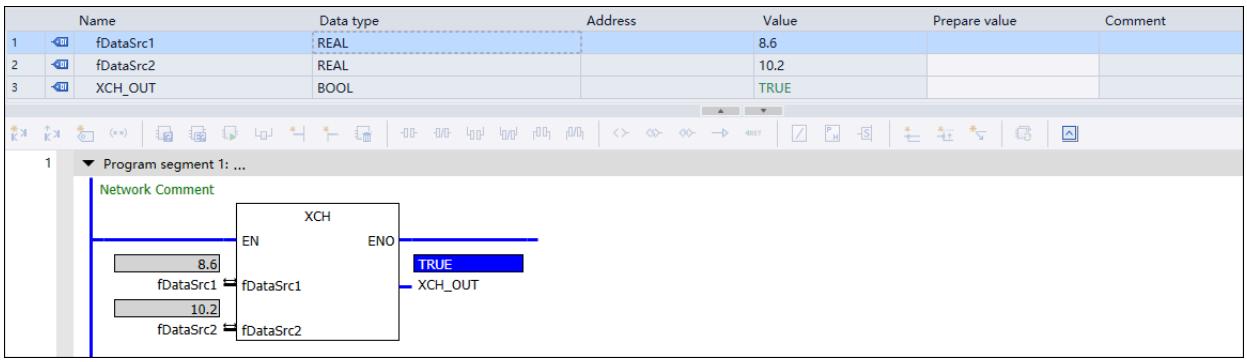


LD:

Before execution:



After execution:

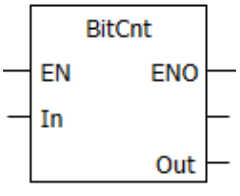


- **Precautions**
  - Returns TRUE if the instruction executes successfully, otherwise returns FALSE.

18.1.2.8 BitCnt

Counts the total number of bits with value TRUE in the bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BitCnt	Bit counter	FC		:= BitCnt(In := , Out =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Counted object	Follows data type	-	The object to be counted.
Output	Out	Count result	Follows data type	0	The number of bits with value TRUE in the counted object.

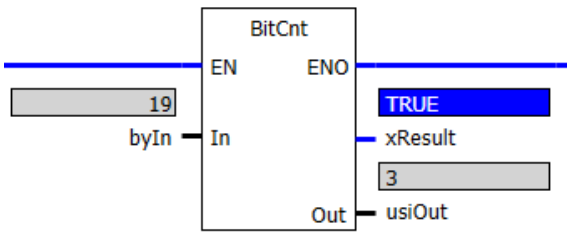
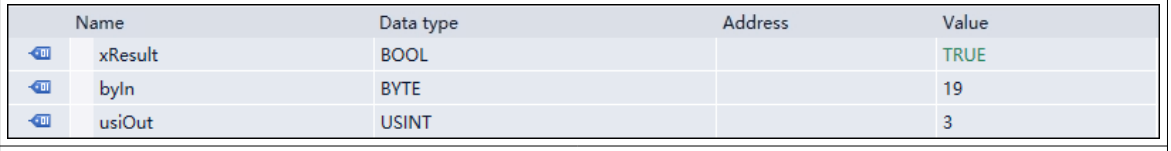
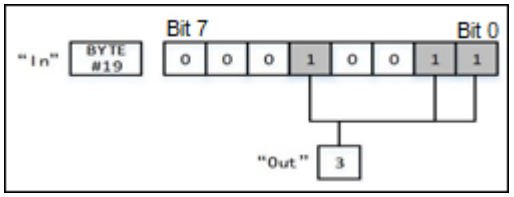
Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function description**

  Counts the total number of bits with value TRUE in the object In.

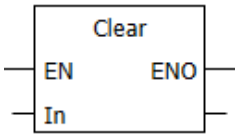
- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; byIn: BYTE := 19; usiOut: USINT; END_VAR	
Program		xResult TRUE := BitCnt(In := byIn 19, Out => usiOut 3);RETURN
Operation result		
Principle analysis		

18.1.2.9 Clear

Initializes the input variable.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
Clear	Initialization	FC		:= Clear(In :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Initialization object	Follows data type	-	The object to be initialized.

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration value, a structure, an array, or a single element of an array.																			

- **Function description**
  - Initializes the value of the object "In" to the initial value for each data type.
  - The initial value for each data type is shown in the table below.

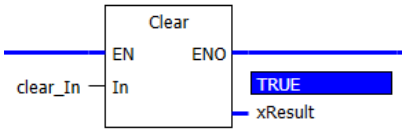
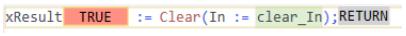
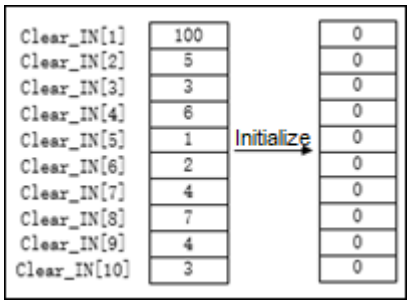
Data Type	Initial Value
BOOL	FALSE
BYTE, WORD, DWORD, LWORD	16#0
USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, LREAL	0
TIME	T#0ms
DATE	D#1970-1-1

Data Type	Initial Value
TOD	TOD#0:0:0
DT	DT#1970-1-1-0:0:0
STRING	''

When "In" is an array or a single element of an array, it is handled shown in the table below.

"In"	Processing Method
Array	Initializes all elements of the array
A single element of an array	Initializes only the element
Structure	Initializes all elements of the structure
A single element of a structure	Initializes only the element

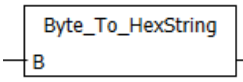
- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; clear_In: ARRAY[1..10] OF UINT := [100,5,3,6,1,2,4,7,4,3]; END_VAR	
Program		
Operation result		
Principle analysis		

18.1.2.10 Byte_To_HexString

This instruction converts bytes to hexadecimal text strings.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
Byte_To_HexString	Byte-to-hexadecimal text string conversion	FC		:= Byte_To_HexString(B :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	B	Byte to convert	0 to 255	0	Input data address
Output	Out	Return value	-	'00'	Hexadecimal string

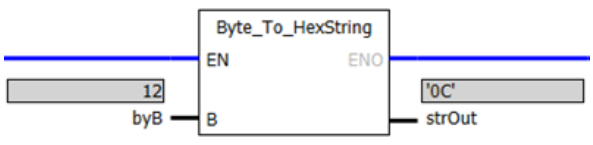
  Variable types

	Boolean	Bit String				Integer								Real Number		Time, duration, date, and text string				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
B	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**

  This instruction converts bytes to hexadecimal text strings.

- **Program example**

	LD	ST
Define variables	VAR byB: BYTE; strOut: STRING; END_VAR	
Program		
Operation result		

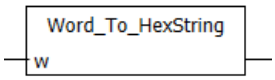
Name	Type	Address	Value
byB	BYTE		12
strOut	STRING		'0C'

- **Precautions**
  - The value range of B is 0 to 255. If the value is not within the range, an error occurs.
  - The output range is '00' to 'FF', that is, 00000000 to 11111111.

18.1.2.11 Word_To_HexString

This instruction converts words to hexadecimal text strings.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
Word_To_HexString	Word-to-hexadecimal text string conversion	FC		:= Word_To_HexString(w :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	w	Word to convert	0 to FFFF	0	Input data address
Output	Out	Return value	-	'0000'	Hexadecimal string

Variable types

	Boolean	Bit String				Integer								Real Number		Time, duration, date, and text string				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
w	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**

  This instruction converts words to hexadecimal text strings.

- **Program example**

	LD	ST
Define variables	VAR wInput: WORD; strOut: STRING; END_VAR	
Program		

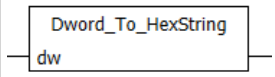
	LD	ST
Operation result		

- **Precautions**
  - The value range of B is 0 to 65535. If the value is not within the range, an error occurs.
  - The output range is '0000' to 'FFFF', that is, 0000 0000 0000 0000 to 1111 1111 1111 1111.

18.1.2.12 Dword_To_HexString

This instruction converts doublewords (DWORD) to hexadecimal text strings.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
Dword_To_HexString	Doubleword-to-hexadecimal text string conversion	FC		:= Dword_To_HexString(dw :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	dw	Doubleword to convert	0 to FFFF FFFF	0	Input data address
Output	Out	Return value	-	'0000 0000'	Hexadecimal string

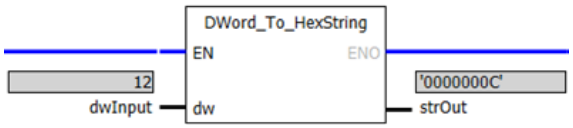
Variable types

	Boolean	Bit String				Integer								Real Number		Time, duration, date, and text string				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
dw	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**

  This instruction converts doublewords (DWORD) to hexadecimal text strings.

- **Program example**

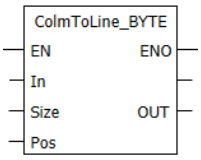
	LD	ST
Define variables	VAR dwInput: DWORD; strOut: STRING; END_VAR	
Program		
Operation result		

- **Precautions**
  - The value range of dw is 0 to 4294967295. If the value is not within the range, an error occurs.
  - The output range is '0000 0000' to 'FFFF_FFFF', that is, 0000 0000 0000 0000 0000 0000 0000 0000 to 1111 1111
    1111 1111 1111 1111 1111 1111.

18.1.2.13 ColmToLine_BYTE

Extracts the value at the specified position of each array element and outputs as a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ColmToLine_BYTE	Specified position combination output of array elements	FC		:= ColmToLine_BYTE(In := , Size := , Pos := , OUT =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In[ ] array	Array where the conversion object is located	Follows data type	-	Array of objects to be converted
	Size	Number of elements in the conversion object	Follows data type	1	Number of elements in In[ ] to be converted
	Pos	Bit position to convert	Follows data type	0	Bit position to be converted

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Conversion result	Follows data type	0	Conversion result

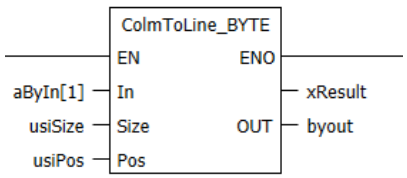
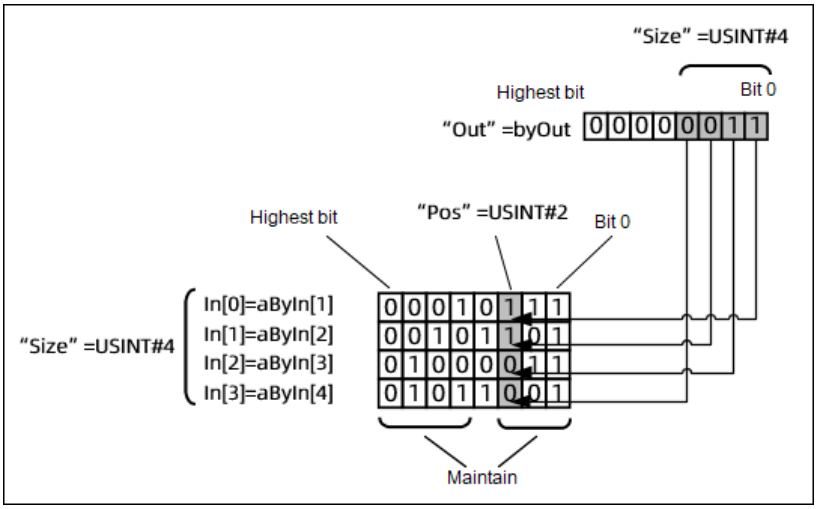
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Pos	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function description**
  - Extracts the value at the specified position of each array element and outputs as a bit string in an arranged way.
  - Extracts "Size" array elements from the input array In[ ] starting at In[0], and extracts only the "Pos"-th bit value from these array elements. Arranges them into a "Size"-bit string, and stores it at the least significant bit position of the conversion result "Out". All remaining bits of "Out" are set to FALSE.
- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	aByIn:ARRAY[0..4] OF byte := [1,23,45,67,89];
	usiSize: USINT := 4;
	usiPos: USINT := 2;
	byout:BYTE;
	END_VAR	

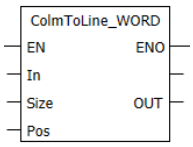
<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>
</td><td><pre>xResult := ColmToLine_BYTE(In := aByIn[1],
  Size := usiSize,
  Pos := usiPos,
  OUT => byout);</pre></td></tr>
<tr><td>Operation result</td><td colspan="2">
<table>
<tr><th></th><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td></td><td>xResult</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td></td><td>aByIn</td><td>ARRAY [0..4] OF BYTE</td><td></td><td></td></tr>
<tr><td></td><td>aByIn[0]</td><td>BYTE</td><td></td><td>2#00000001</td></tr>
<tr><td></td><td>aByIn[1]</td><td>BYTE</td><td></td><td>2#00010111</td></tr>
<tr><td></td><td>aByIn[2]</td><td>BYTE</td><td></td><td>2#00101101</td></tr>
<tr><td></td><td>aByIn[3]</td><td>BYTE</td><td></td><td>2#01000011</td></tr>
<tr><td></td><td>aByIn[4]</td><td>BYTE</td><td></td><td>2#01011001</td></tr>
<tr><td></td><td>usiSize</td><td>USINT</td><td></td><td>4</td></tr>
<tr><td></td><td>usiPos</td><td>USINT</td><td></td><td>2</td></tr>
<tr><td></td><td>byout</td><td>BYTE</td><td></td><td>2#00000011</td></tr>
</table>
</td></tr>
<tr><td>Principle analysis</td><td colspan="2">
</td></tr>
</table>

- **Precautions**
  - When the value of "Size" is 0, the return value is TRUE, and all bits of "Out" are FALSE.
  - When the value of "Size" exceeds the valid range, the return value is FALSE, and "Out" remains unchanged.
  - When the value of "Pos" exceeds the valid range, the return value is FALSE, and "Out" remains unchanged.
  - When the value of "Size" exceeds the array range of In[ ], the return value is FALSE, and "Out" remains unchanged.
  - When the input array dimension is greater than 3, the return value is FALSE, and "Out" remains unchanged.

18.1.2.14 ColmToLine_WORD

Extracts the value at the specified position of each array element and outputs as a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ColmToLine_WORD	Specified position combination output of array elements	FC		:= ColmToLine_WORD(In := , Size := , Pos := , OUT =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In[ ] array	Array where the conversion object is located	Follows data type	-	Array of objects to be converted
	Size	Number of elements in the conversion object	Follows data type	1	Number of elements in In[ ] to be converted
	Pos	Bit position to convert	Follows data type	0	Bit position to be converted
Output	Out	Conversion result	Follows data type	0	Conversion result

Variable types

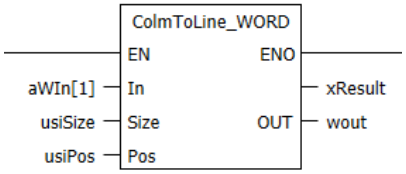
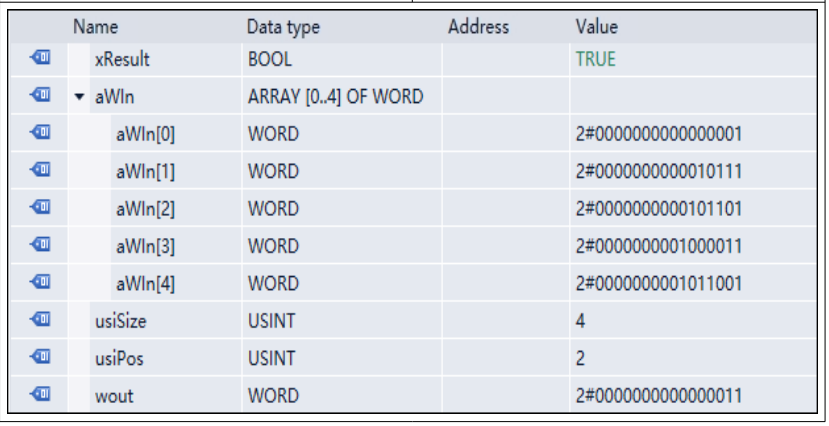
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Pos	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

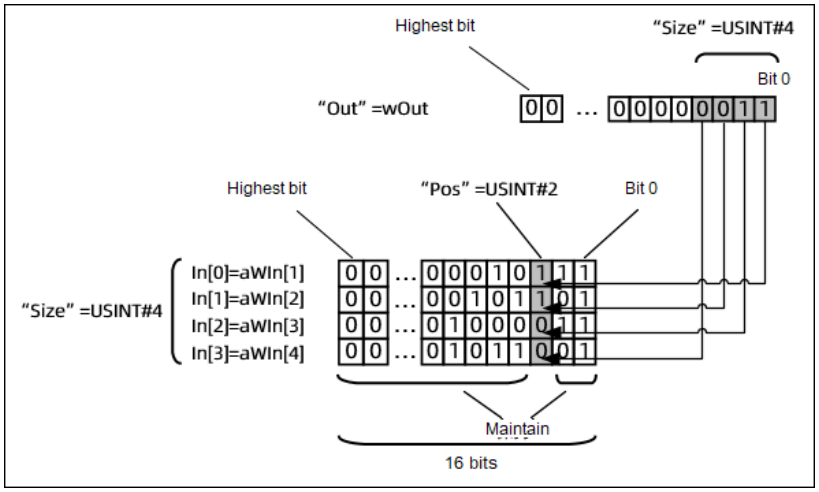
- **Function description**
  - Extracts the value at the specified position of each array element and outputs as a bit string in an arranged way.
  - Extracts "Size" array elements from the input array In[ ] starting at In[0], and extracts only the "Pos"-th bit value from these array elements. Arranges them into a "Size"-bit string, and stores it at the least significant bit position of the conversion result "Out". All remaining bits of "Out" are set to FALSE.
- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	aWIn: ARRAY[0..4] OF WORD := [1, 23, 45, 67, 89];
	usiSize: USINT := 4;
	usiPos: USINT := 2;
	wout: WORD;
	var1: BOOL;
	END_VAR	

Program		xResult := ColmToLine_WORD(In := aWIn[1], Size := usiSize, Pos := usiPos, OUT => wout);
Operation result		

Name	Data type	Address	Value
xResult	BOOL		TRUE
aWIn	ARRAY [0..4] OF WORD		
aWIn[0]	WORD		2#0000000000000001
aWIn[1]	WORD		2#0000000000010111
aWIn[2]	WORD		2#0000000000101101
aWIn[3]	WORD		2#0000000001000011
aWIn[4]	WORD		2#0000000001011001
usiSize	USINT		4
usiPos	USINT		2
wout	WORD		2#0000000000000011

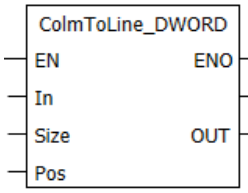
	LD	ST
Principle analysis		

- **Precautions**
  - When the value of "Size" is 0, the return value is TRUE, and all bits of "Out" are FALSE.
  - When the value of "Size" exceeds the valid range, the return value is FALSE, and "Out" remains unchanged.
  - When the value of "Pos" exceeds the valid range, the return value is FALSE, and "Out" remains unchanged.
  - When the value of "Size" exceeds the array range of In[ ], the return value is FALSE, and "Out" remains unchanged.
  - When the input array dimension is greater than 3, the return value is FALSE, and "Out" remains unchanged.

18.1.2.15 ColmToLine_DWORD

Extracts the value at the specified position of each array element and outputs as a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ColmToLine_DWORD	Specified position combination output of array elements	FC		:= ColmToLine_DWORD(In := , Size := , Pos := , OUT =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In[ ] array	Array where the conversion object is located	Follows data type	-	Array of objects to be converted

Input/Output	Variable	Name	Valid Range	Default Value	Description
	Size	Number of elements in the conversion object	Follows data type	1	Number of elements in In[ ] to be converted
	Pos	Bit position to convert	Follows data type	0	Bit position to be converted
Output	Out	Conversion result	Follows data type	0	Conversion result

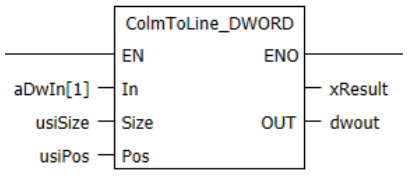
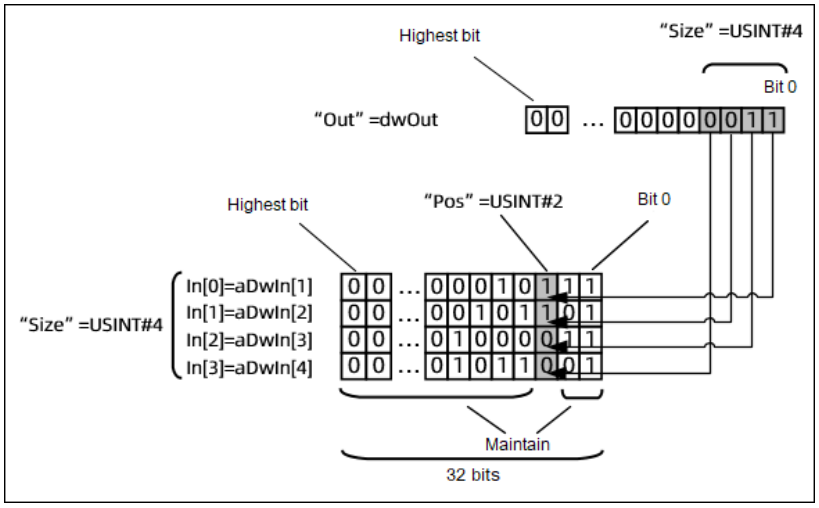
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Pos	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function description**
  - Extracts the value at the specified position of each array element and outputs as a bit string in an arranged way.
  - Extracts "Size" array elements from the input array In[ ] starting at In[0], and extracts only the "Pos"-th bit value from these array elements. Arranges them into a "Size"-bit string, and stores it at the least significant bit position of the conversion result "Out". All remaining bits of "Out" are set to FALSE.
- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	aDwIn: ARRAY[0..4] OF DWORD := [1, 23, 45, 67, 89];
	usiSize: USINT := 4;
	usiPos: USINT := 2;
	dwout: DWORD;
	END_VAR	

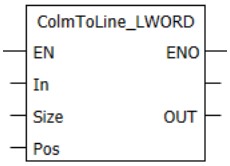
<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>
</td><td><pre>xResult := ColmToLine_DWORD(In := aDwIn[1],
  Size := usiSize,
   Pos := usiPos,
   OUT => dwout);</pre></td></tr>
<tr><td>Operation result</td><td colspan="2"><table>
<tr><td>xResult</td><td>BOOL</td><td>TRUE</td></tr>
<tr><td>aDwIn</td><td>ARRAY [0..4] OF DWORD</td><td></td></tr>
<tr><td>aDwIn[0]</td><td>DWORD</td><td>2#00000000000000000000000000000001</td></tr>
<tr><td>aDwIn[1]</td><td>DWORD</td><td>2#00000000000000000000000000010111</td></tr>
<tr><td>aDwIn[2]</td><td>DWORD</td><td>2#00000000000000000000000000101101</td></tr>
<tr><td>aDwIn[3]</td><td>DWORD</td><td>2#00000000000000000000000001000011</td></tr>
<tr><td>aDwIn[4]</td><td>DWORD</td><td>2#00000000000000000000000001011001</td></tr>
<tr><td>usiSize</td><td>USINT</td><td>4</td></tr>
<tr><td>usiPos</td><td>USINT</td><td>2</td></tr>
<tr><td>dwout</td><td>DWORD</td><td>2#00000000000000000000000000000011</td></tr>
</table></td></tr>
<tr><td>Principle analysis</td><td>
</td></tr>
</table>

- **Precautions**
  - When the value of "Size" is 0, the return value is TRUE, and all bits of "Out" are FALSE.
  - When the value of "Size" exceeds the valid range, the return value is FALSE, and "Out" remains unchanged.
  - When the value of "Pos" exceeds the valid range, the return value is FALSE, and "Out" remains unchanged.
  - When the value of "Size" exceeds the array range of In[ ], the return value is FALSE, and "Out" remains unchanged.
  - When the input array dimension is greater than 3, the return value is FALSE, and "Out" remains unchanged.

18.1.2.16 ColmToLine_LWORD

Extracts the value at the specified position of each array element and outputs as a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ColmToLine_LWORD	Specified position combination output of array elements	FC		:= ColmToLine_LWORD(In := , Size := , Pos := , OUT =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In[ ] array	Array where the conversion object is located	Follows data type	-	Array of objects to be converted
	Size	Number of elements in the conversion object	Follows data type	1	Number of elements in In[ ] to be converted
	Pos	Bit position to convert	Follows data type	0	Bit position to be converted
Output	Out	Conversion result	Follows data type	0	Conversion result

Variable types

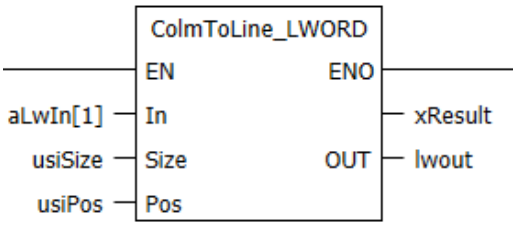
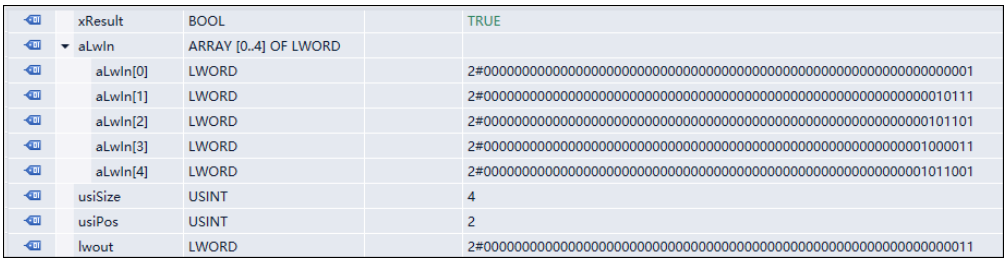
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Pos	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

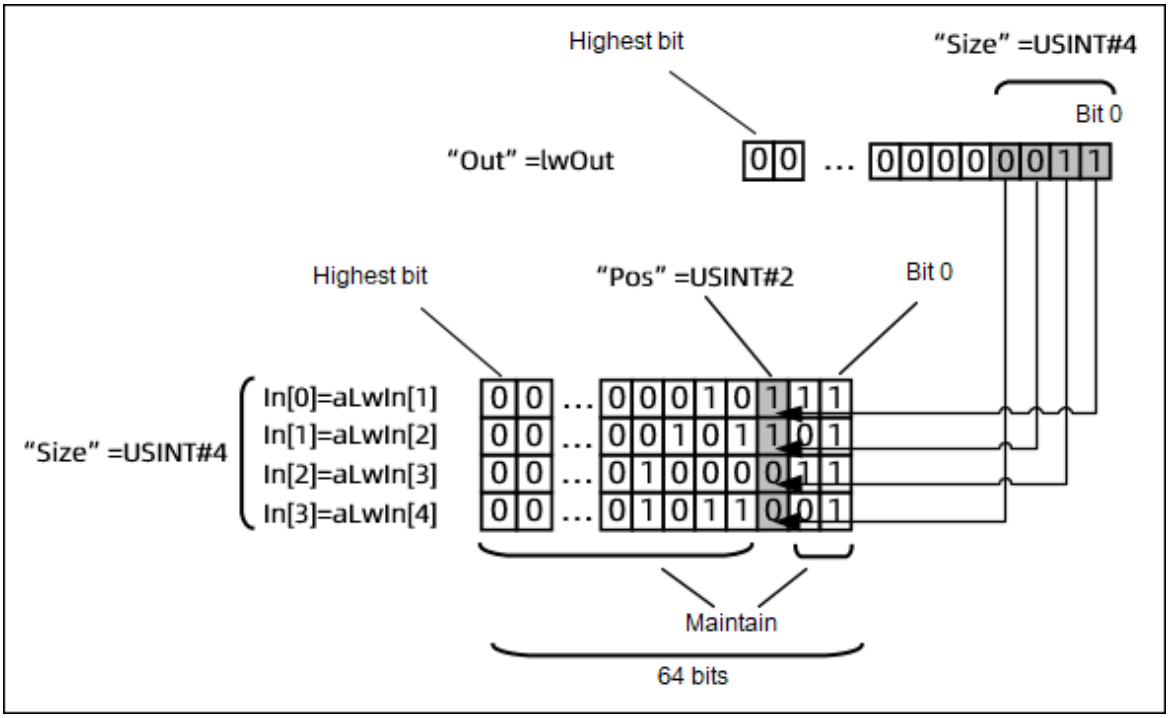
- **Function description**
  - Extracts the value at the specified position of each array element and outputs as a bit string in an arranged way.
  - Extracts "Size" array elements from the input array In[ ] starting at In[0], and extracts only the "Pos"-th bit value from these array elements. Arranges them into a "Size"-bit string, and stores it at the least significant bit position of the conversion result "Out". All remaining bits of "Out" are set to FALSE.
- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	aLwIn: ARRAY[0..4] OF LWORD := [1, 23, 45, 67, 89];
	usiSize: USINT := 4;
	usiPos: USINT := 2;
	lwout: LWORD;
	END_VAR	

Program		xResult := ColmToLine_LWORD(In := aLwIn[1], Size := usiSize, Pos := usiPos, OUT => lwout);
Operation result		

	xResult	BOOL	TRUE
	aLwIn	ARRAY [0..4] OF LWORD	
	aLwIn[0]	LWORD	2#0000000000000000000000000000000000000000000000000000000000000001
	aLwIn[1]	LWORD	2#0000000000000000000000000000000000000000000000000000000000010111
	aLwIn[2]	LWORD	2#0000000000000000000000000000000000000000000000000000000000101101
	aLwIn[3]	LWORD	2#0000000000000000000000000000000000000000000000000000000001000011
	aLwIn[4]	LWORD	2#0000000000000000000000000000000000000000000000000000000001011001
	usiSize	USINT	4
	usiPos	USINT	2
	lwout	LWORD	2#0000000000000000000000000000000000000000000000000000000000000011

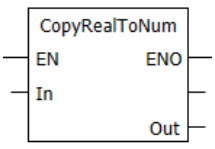
	LD	ST
Principle analysis		

- **Precautions**
  - When the value of "Size" is 0, the return value is TRUE, and all bits of "Out" are FALSE.
  - When the value of "Size" exceeds the valid range, the return value is FALSE, and "Out" remains unchanged.
  - When the value of "Pos" exceeds the valid range, the return value is FALSE, and "Out" remains unchanged.
  - When the value of "Size" exceeds the array range of In[ ], the return value is FALSE, and "Out" remains unchanged.
  - When the input array dimension is greater than 3, the return value is FALSE, and "Out" remains unchanged.

18.1.2.17 CopyRealToNum

Copies a real number directly to a signed integer.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CopyRealToNum	Bit pattern copy (real number to signed integer)	FC		:= CopyRealToNum(In := , Out =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Copy source	Follows data type	-	Source to be copied
Output	Out	Paste target	Follows data type	-	Target to paste to

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Out	-	-	-	-	-	-	-	-	-	-	-	√	–	-	-	-	-	-	-	-

- **Function description**

  The source content is copied directly to the target "Out" without any processing. There are 2 instruction names, depending on the combination of data types for "In" and "Out".

"In"	"Out"	Instruction
REAL	DINT	CopyRealToNum
LREAL	LINT	CopyLrealToNum

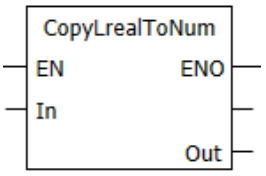
- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; reIn: REAL := 10.5; diOut: DINT; END_VAR	
Program	CopyRealToNum EN ENO reIn — In xResult Out — diOut	xResult := CopyRealToNum(In := reIn, Out => diOut);
Operation result	Name / Data type / Address / Value xResult / BOOL / / TRUE reIn / REAL / / 2#01000001001010000000000000000000 diOut / DINT / / 2#01000001001010000000000000000000	

18.1.2.18 CopyLrealToNum

Copies a real number directly to a signed integer.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CopyLrealToNum	Bit pattern copy (long real number to signed integer)	FC		:= CopyLrealToNum(In := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Copy source	Follows data type	-	Source to be copied
Output	Out	Paste target	Follows data type	-	Target to paste to

Variable type

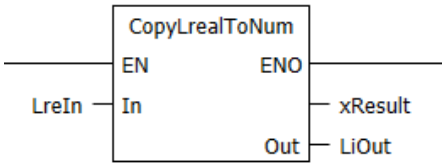
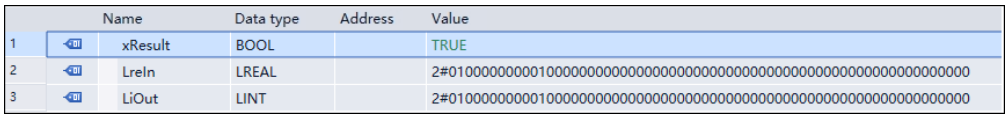
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Out	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-

- **Function description**

The source content is copied directly to the target "Out" without any processing. There are 2 instruction names, depending on the combination of data types for "In" and "Out".

"In"	"Out"	Instruction
REAL	DINT	CopyRealToNum
LREAL	LINT	CopyLrealToNum

- **Program example**

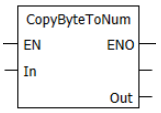
	LD	ST
Define variables	VAR xResult: BOOL; LreIn: LREAL := 4; LiOut: LINT; END_VAR	
Program		xResult := CopyLrealToNum(In := LreIn, Out => LiOut);
Operation result		

	Name	Data type	Address	Value
1	xResult	BOOL		TRUE
2	LreIn	LREAL		2#0100000000010000000000000000000000000000000000000000000000000000
3	LiOut	LINT		2#0100000000010000000000000000000000000000000000000000000000000000

18.1.2.19 CopyByteToNum

Copies a bit string directly to a signed integer.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CopyByteToNum	Bit pattern copy (BYTE-type bit string to signed integer)	FC		:= CopyByteToNum(In := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Copy source	Follows data type	0	Source to be copied
Output	Out	Paste target	Follows data type	0	Target to paste to

Variable type

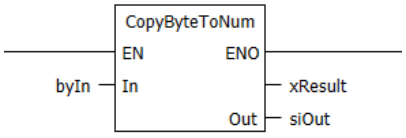
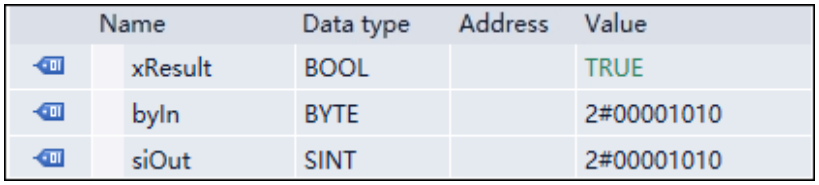
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	A signed integer with the same size as "In"																			

- **Function description**

  Copies a bit string directly to a signed integer. There are 4 instruction names, depending on the combination of data types for "In" and "Out".

"In"	"Out"	Instruction
BYTE	SINT	CopyByteToNum
WORD	INT	CopyWordToNum
DWORD	DINT	CopyDwordToNum
LWORD	LINT	CopyLwordToNum

- **Program example**

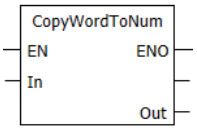
	LD	ST
Define variables	VAR xResult: BOOL; byIn: BYTE := 10; siOut: SINT; END_VAR	
Program		xResult := CopyByteToNum(In := byIn, Out => siOut);
Operation result		

Name	Data type	Address	Value
xResult	BOOL		TRUE
byIn	BYTE		2#00001010
siOut	SINT		2#00001010

18.1.2.20 CopyWordToNum

Copies a bit string directly to a signed integer.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CopyWordToNum	Bit pattern copy (WORD-type bit string to signed integer)	FC		:= CopyWordToNum(In := , Out =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Copy source	Follows data type	0	Source to be copied
Output	Out	Paste target	Follows data type	0	Target to paste to

  Variable type

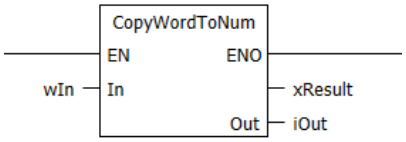
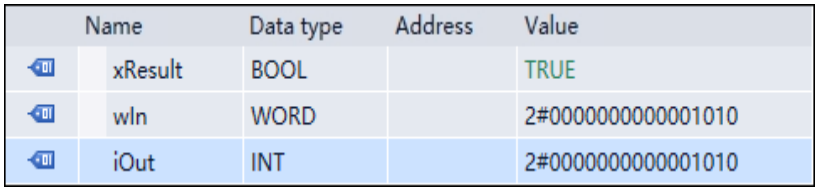
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	A signed integer with the same size as "In"																			

- **Function description**

  Copies a bit string directly to a signed integer. There are 4 instruction names, depending on the combination of data types for "In" and "Out".

"In"	"Out"	Instruction
BYTE	SINT	CopyByteToNum
WORD	INT	CopyWordToNum
DWORD	DINT	CopyDwordToNum
LWORD	LINT	CopyLwordToNum

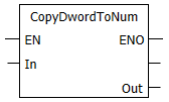
- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; wIn: WORD := 10; iOut: INT; END_VAR	
Program		xResult := CopyWordToNum(In := wIn, Out => iOut);
Operation result		

18.1.2.21 CopyDwordToNum

Copies a bit string directly to a signed integer.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CopyDwordToNum	Bit pattern copy (DWORD-type bit string to signed integer)	FC		:= CopyDwordToNum(In := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Copy source	Follows data type	0	Source to be copied
Output	Out	Paste target	Follows data type	0	Target to paste to

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	A signed integer with the same size as "In"																			

- **Function description**

  Copies a bit string directly to a signed integer. There are 4 instruction names, depending on the combination of data types for "In" and "Out".

"In"	"Out"	Instruction
BYTE	SINT	CopyByteToNum
WORD	INT	CopyWordToNum
DWORD	DINT	CopyDwordToNum
LWORD	LINT	CopyLwordToNum

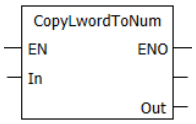
- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; dwIn: DWORD := 10; diOut: DINT; END_VAR	
Program	CopyDwordToNum EN ENO dwIn In xResult Out diOut	xResult := CopyDwordToNum(In := dwIn, Out => diOut);
Operation result	Name / Data type / Address / Value xResult / BOOL / / TRUE dwIn / DWORD / / 2#00000000000000000000000000001010 diOut / DINT / / 2#00000000000000000000000000001010	

18.1.2.22 CopyLwordToNum

Copies a bit string directly to a signed integer.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CopyLwordToNum	Bit pattern copy (LWORD-type bit string to signed integer)	FC		:= CopyLwordToNum(In := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Copy source	Follows data type	0	Source to be copied
Output	Out	Paste target	Follows data type	0	Target to paste to

Variable type

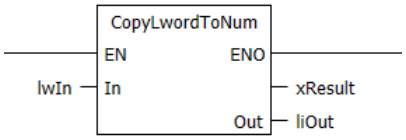
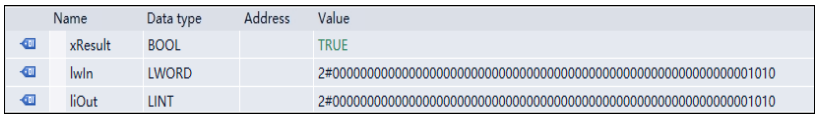
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Out	A signed integer with the same size as "In"																			

- **Function description**

Copies a bit string directly to a signed integer. There are 4 instruction names, depending on the combination of data types for "In" and "Out".

"In"	"Out"	Instruction
BYTE	SINT	CopyByteToNum
WORD	INT	CopyWordToNum
DWORD	DINT	CopyDwordToNum
LWORD	LINT	CopyLwordToNum

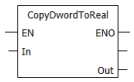
- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; lwIn: LWORD := 10; liOut: LINT; END_VAR	
Program		xResult := CopyLwordToNum(In := lwIn, Out => liOut);
Operation result		

18.1.2.23 CopyDwordToReal

Copies a bit string directly into a real number.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CopyDwordToReal	Bit pattern copy (DWORD-type bit string to real number)	FC		:= CopyDwordToReal(In := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Copy source	Follows data type	0	Source to be copied
Output	Out	Paste target	Follows data type	0	Target to paste to

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

- **Function description**

  Copies a bit string directly into a real number. There are 2 instruction names, depending on the combination of data types for "In" and "Out".

"In"	"Out"	Instruction
DWORD	REAL	CopyDwordToReal
LWORD	LREAL	CopyLwordToLreal

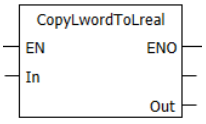
- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; dwIn: DWORD := 10; reOut: REAL; END_VAR	
Program	CopyDwordToReal EN ENO dwIn — In — xResult Out — reOut	xResult := CopyDwordToReal(In := dwIn, Out => reOut);
Operation result	Name / Data type / Address / Value xResult / BOOL / / TRUE dwIn / DWORD / / 2#00000000000000000000000000001010 reOut / REAL / / 2#00000000000000000000000000001010	

18.1.2.24 CopyLwordToLreal

Copies a bit string directly into a real number.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CopyLwordToLreal	Bit pattern copy (LWORD-type bit string to long real number)	FC		:= CopyLwordToLreal(In := , Out =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Copy source	Follows data type	0	Source to be copied
Output	Out	Paste target	Follows data type	0	Target to paste to

  Variable type

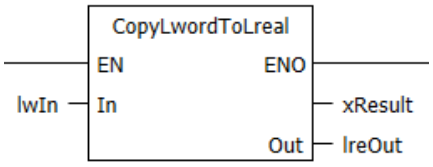
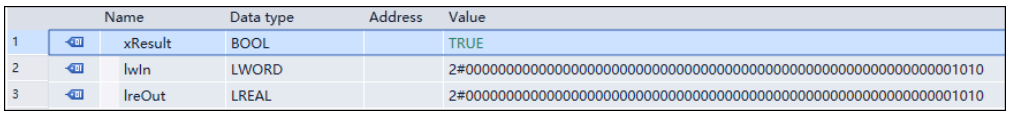
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-

- **Function description**

  Copies a bit string directly into a real number. There are 2 instruction names, depending on the combination of data types for "In" and "Out".

"In"	"Out"	Instruction
DWORD	REAL	CopyDwordToReal
LWORD	LREAL	CopyLwordToLreal

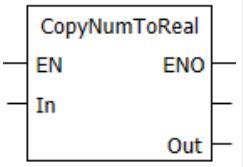
- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; lwIn: LWORD := 10; lreOut: LREAL; END_VAR	
Program		xResult := CopyLwordToLreal(In := lwIn, Out => lreOut);
Operation result		

18.1.2.25 CopyNumToReal

Copies a signed integer directly into a real number.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CopyNumToReal	Bit pattern copy (integer to real number)	FC		:= CopyNumToReal(In := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Copy source	Follows data type	0	Source to be copied
Output	Out	Paste target	Follows data type	0	Target to paste to

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

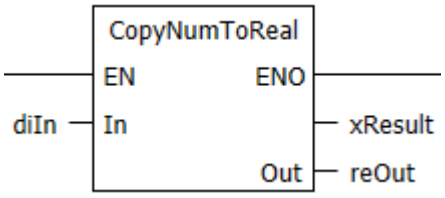
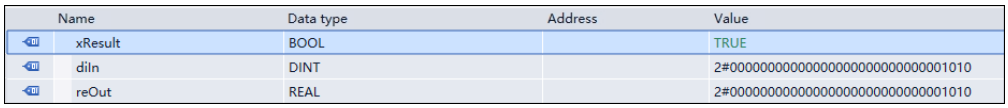
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-

- **Function**

  The source content is copied directly to the target "Out" without any processing. There are 2 instruction names, depending on the combination of data types for "In" and "Out".

"In"	"Out"	Instruction
DINT	REAL	CopyNumToReal
LINT	LREAL	CopyNumToLreal

- **Program example**

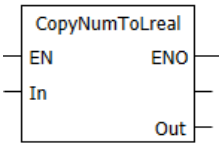
	LD	ST
Define variables	VAR xResult: BOOL; diIn: DINT:= 10; reOut: REAL; END_VAR	
Program		xResult := CopyNumToReal(In := diIn, Out => reOut);
Operation result		

Name	Data type	Address	Value
xResult	BOOL		TRUE
diIn	DINT		2#00000000000000000000000000001010
reOut	REAL		2#00000000000000000000000000001010

18.1.2.26 CopyNumToLreal

Copies a signed integer directly into a real number.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CopyNumToLreal	Bit pattern copy (integer to long real number)	FC		:= CopyNumToLreal(In := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Copy source	Follows data type	0	Source to be copied
Output	Out	Paste target	Follows data type	0	Target to paste to

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-

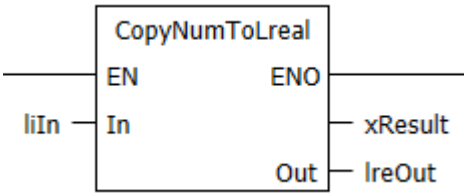
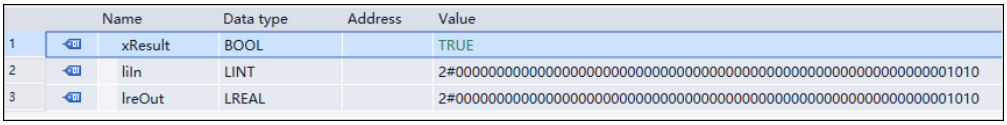
- **Function**

The source content is copied directly to the target "Out" without any processing. There are 2 instruction names, depending on the combination of data types for "In" and "Out".

"In"	"Out"	Instruction
DINT	REAL	CopyNumToReal
LINT	LREAL	CopyNumToLreal

- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; liIn: LINT := 10; lreOut: LREAL; END_VAR	

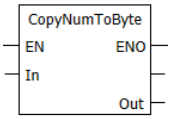
	LD	ST
Program		`xResult := CopyNumToLreal(In := liIn, Out => lreOut);`
Operation result		

	Name	Data type	Address	Value
1	xResult	BOOL		TRUE
2	liIn	LINT		2#0000000000000000000000000000000000000000000000000000000000001010
3	lreOut	LREAL		2#0000000000000000000000000000000000000000000000000000000000001010

18.1.2.27 CopyNumToByte

Copies a signed integer directly into a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CopyNumToByte	Bit pattern copy (integer to BYTE-type bit string)	FC		:= CopyNumToByte(In := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Copy source	Follows data type	0	Source to be copied
Output	Out	Paste target	Follows data type	0	Target to paste to

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–

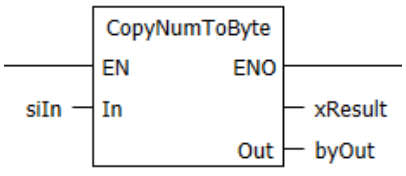
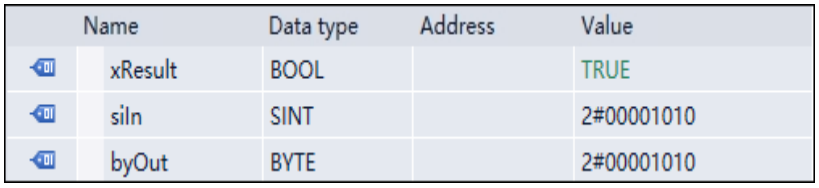
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  The source content is copied directly to the target "Out" without any processing. There are 4 instruction names, depending on the combination of data types for "In" and "Out".

"In"	"Out"	Instruction
SINT	BYTE	CopyNumToByte
INT	WORD	CopyNumToWord
DINT	DWORD	CopyNumToDword
LINT	LWORD	CopyNumToLword

- **Program example**

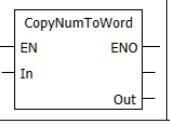
	LD	ST
Define variables	VAR xResult: BOOL; siIn: SINT := 10; byOut: BYTE; END_VAR	
Program		xResult:= CopyNumToByte(In := siIn, Out => byOut);
Operation result		

Name	Data type	Address	Value
xResult	BOOL		TRUE
siIn	SINT		2#00001010
byOut	BYTE		2#00001010

18.1.2.28 CopyNumToWord

Copies a signed integer directly into a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CopyNumToWord	Bit pattern copy (integer to WORD-type bit string)	FC		:= CopyNumToWord(In := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Copy source	Follows data type	0	Source to be copied
Output	Out	Paste target	Follows data type	0	Target to paste to

Variable type

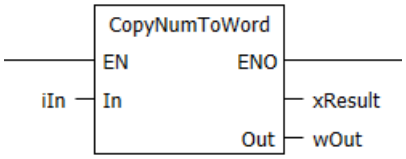
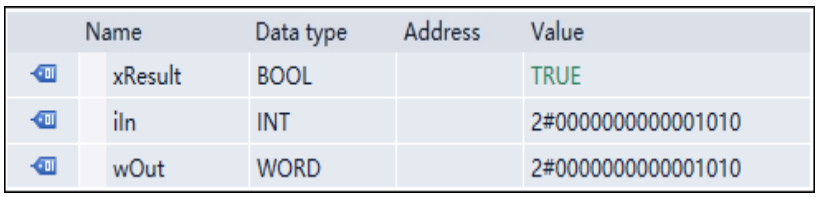
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
Out	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

The source content is copied directly to the target "Out" without any processing. There are 4 instruction names, depending on the combination of data types for "In" and "Out".

"In"	"Out"	Instruction
SINT	BYTE	CopyNumToByte
INT	WORD	CopyNumToWord
DINT	DWORD	CopyNumToDword
LINT	LWORD	CopyNumToLword

- **Program example**

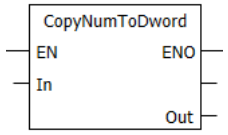
	LD	ST
Define variables	VAR xResult: BOOL; iIn: INT := 10; wOut: WORD; END_VAR	
Program		xResult := CopyNumToWord(In := iIn, Out => wOut);
Operation result		

	Name	Data type	Address	Value
	xResult	BOOL		TRUE
	iIn	INT		2#0000000000001010
	wOut	WORD		2#0000000000001010

18.1.2.29 CopyNumToDword

Copies a signed integer directly into a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CopyNumToDword	Bit pattern copy (integer to DWORD-type bit string)	FC		:= CopyNumToDword(In := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Copy source	Follows data type	0	Source to be copied
Output	Out	Paste target	Follows data type	0	Target to paste to

Variable type

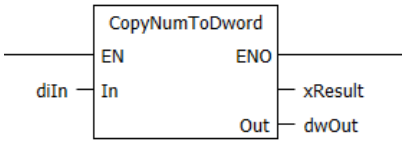
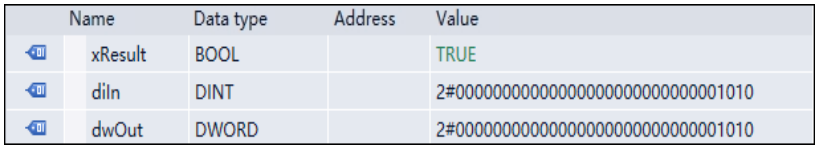
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Out	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  The source content is copied directly to the target "Out" without any processing. There are 4 instruction names, depending on the combination of data types for "In" and "Out".

"In"	"Out"	Instruction
SINT	BYTE	CopyNumToByte
INT	WORD	CopyNumToWord
DINT	DWORD	CopyNumToDword
LINT	LWORD	CopyNumToLword

- **Program example**

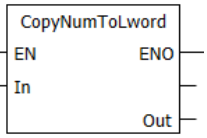
	LD	ST
Define variables	VAR xResult: BOOL; diIn: DINT := 10; dwOut: DWORD; END_VAR	
Program		xResult := CopyNumToDword(In := diIn, Out => dwOut);
Operation result		

Name	Data type	Address	Value
xResult	BOOL		TRUE
diIn	DINT		2#00000000000000000000000000001010
dwOut	DWORD		2#00000000000000000000000000001010

18.1.2.30 CopyNumToLword

Copies a signed integer directly into a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CopyNumToLword	Bit pattern copy (integer to LWORD-type bit string)	FC		:= CopyNumToLword(In := , Out =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Copy source	Follows data type	0	Source to be copied
Output	Out	Paste target	Follows data type	0	Target to paste to

  Variable type

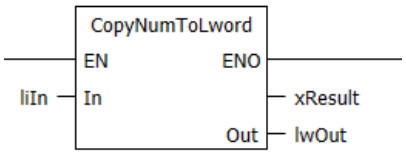
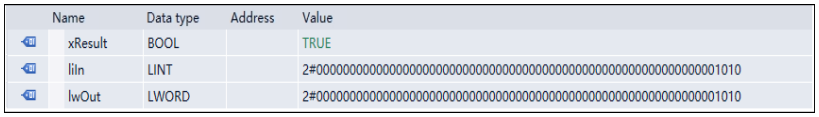
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–
Out	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  The source content is copied directly to the target "Out" without any processing. There are 4 instruction names, depending on the combination of data types for "In" and "Out".

"In"	"Out"	Instruction
SINT	BYTE	CopyNumToByte
INT	WORD	CopyNumToWord
DINT	DWORD	CopyNumToDword
LINT	LWORD	CopyNumToLword

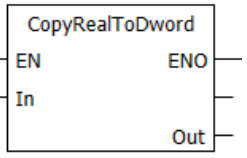
- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; liIn: LINT := 10; lwOut: LWORD; END_VAR	
Program		xResult := CopyNumToLword(In := liIn, Out => lwOut);
Operation result		

18.1.2.31 CopyRealToDword

Copies a real number directly into a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CopyRealToDword	Bit pattern copy (real number to DWORD-type bit string)	FC		:= CopyRealToDword(In := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Copy source	Follows data type	0	Source to be copied
Output	Out	Paste target	Follows data type	0	Target to paste to

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

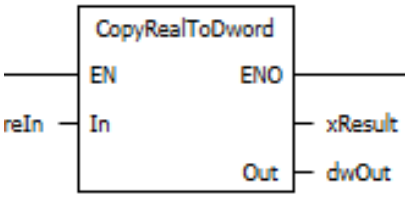
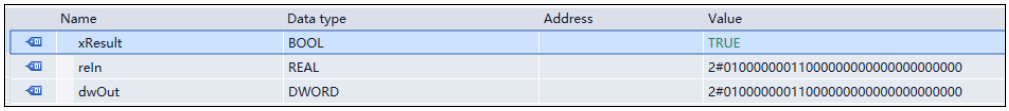
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**

  Copies a real number directly into a bit string. There are 2 instruction names, depending on the combination of data types for "In" and "Out".

"In"	"Out"	Instruction
REAL	DWORD	CopyRealToDword
LREAL	LWORD	CopyLrealToLword

- **Program example**

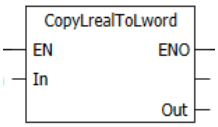
	LD	ST
Define variables	VAR xResult: BOOL; reIn: REAL := 3.5; dwOut: DWORD; END_VAR	
Program		xResult := CopyRealToDword(In := reIn, Out => dwOut);
Operation result		

Name	Data type	Address	Value
xResult	BOOL		TRUE
reIn	REAL		2#01000000011000000000000000000000
dwOut	DWORD		2#01000000011000000000000000000000

18.1.2.32 CopyLrealToLword

Copies a real number directly into a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CopyLrealToLword	Bit pattern copy (long real number to LWORD-type bit string)	FC		:= CopyLrealToLword(In := , Out =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Copy source	Follows data type	0	Source to be copied
Output	Out	Paste target	Follows data type	0	Target to paste to

  Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–		√	–	–	–	–	–
Out	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

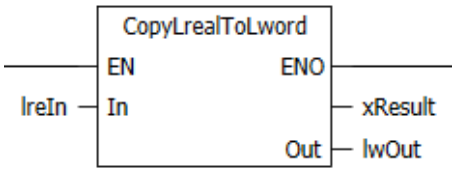
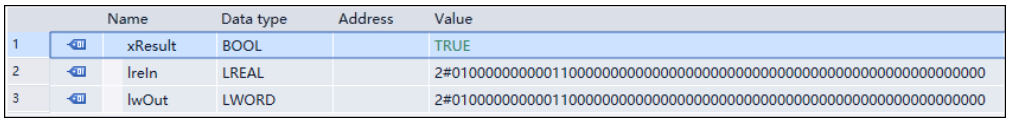
- **Function**

  Copies a real number directly into a bit string. There are 2 instruction names, depending on the combination of data types for "In" and "Out".

"In"	"Out"	Instruction
REAL	DWORD	CopyRealToDword
LREAL	LWORD	CopyLrealToLword

- **Program example**

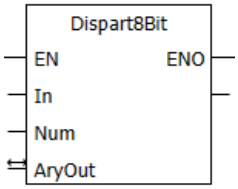
	LD	ST
Define variables	VAR xResult: BOOL; lreIn: LREAL := 3.5; lwOut: LWORD; END_VAR	

	LD	ST
Program		`xResult := CopyLrealToLword(In := lreIn, Out => lwOut);`
Operation result		

18.1.2.33 Dispart8Bit

Separates a bit string by 1 byte.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
Dispart8Bit	Data separation by byte	FC		:= Dispart8Bit(In := , Num := , AryOut :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Separation object	Follows data type	-	A bit string to be separated
	Num	Number of digits for separation	Follows data type	1	Number of bytes for separation
Input/Output	AryOut[ ] array	Separation result	Follows data type	-	An array indicating the separation result

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
AryOut[ ] array	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

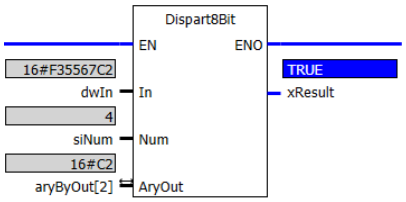
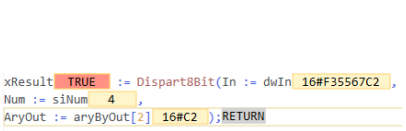
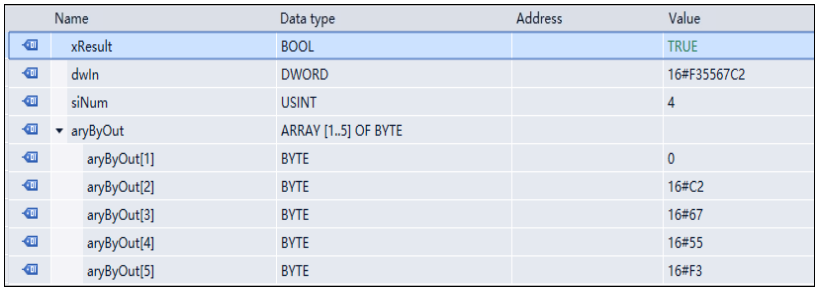
  Separates "In" by 1 byte and saves the result in the array "AryOut[ ]".

  The least significant byte is stored in AryOut[0], and the next byte is stored in AryOut[1]. The number of byte for separation "Num" is processed in the same way.

- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	dwIn: DWORD := 16#F35567C2;
	siNum: USINT := 4;
	aryByOut: ARRAY[1..5] OF byte;
	END_VAR	

Program		
Operation result		

Name	Data type	Address	Value
xResult	BOOL		TRUE
dwIn	DWORD		16#F35567C2
siNum	USINT		4
aryByOut	ARRAY [1..5] OF BYTE		
aryByOut[1]	BYTE		0
aryByOut[2]	BYTE		16#C2
aryByOut[3]	BYTE		16#67
aryByOut[4]	BYTE		16#55
aryByOut[5]	BYTE		16#F3

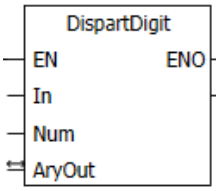
- **Precautions**
  - When the value of "Num" exceeds the valid range, the return value is FALSE, and "AryOut[ ]" remains unchanged.
  - If the value of "Num" is 0, the return value is TRUE, and "AryOut[ ]" remains unchanged.

- When the value of "Num" exceeds the valid range of "AryOut[ ]", the return value is FALSE, and "AryOut[ ]" remains unchanged.
- If the dimension of "AryOut[ ]" is greater than 3, the return value is FALSE, and "AryOut[ ]" remains unchanged.

18.1.2.34 DispartDigit

Separates a bit string by 1 digit (4 bits).

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DispartDigit	Data separation by digit	FC		:= DispartDigit(In := , Num := , AryOut :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Separation object	Follows data type	-	A bit string to be separated
	Num	Number of bits to separate	Follows data type	1	Number of bits to separate
Input/Output	AryOut[ ] array	Separation result	16#00 to 16#0F	-	An array indicating the separation result

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Ary Out[ ] array	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

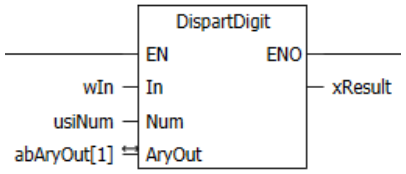
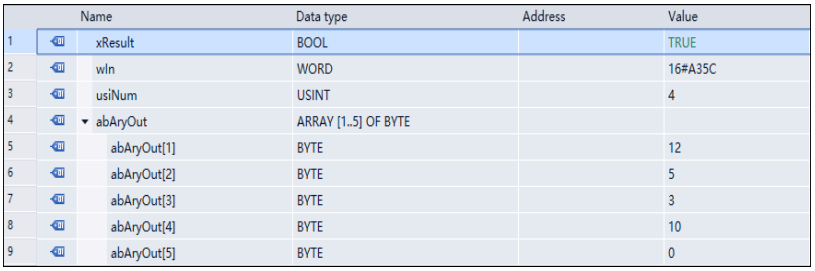
  Separates "In" by 1 digit (4 bits) and saves the result in the array "AryOut[ ]".

  The least significant digit is stored in "AryOut[0]". The 4th bit to 7th bit in the BYTE-type "AryOut[0]" are added with 16#0. The number of digits for separation "Num" is processed in the same way.

- **Program example**

	LD	ST
Define variables		

```
VAR
    xResult: BOOL;
    wIn: WORD := 16#a35c;
    usiNum : USINT := 4;
    abAryOut: ARRAY [1..5] OF BYTE;
END_VAR
```

Program		`xResult := DispartDigit(In := wIn, Num := usiNum, AryOut := abAryOut[1]);`
Operation result Operation result		

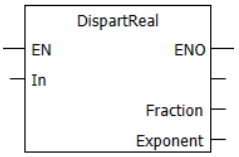
	Name	Data type	Address	Value
1	xResult	BOOL		TRUE
2	wIn	WORD		16#A35C
3	usiNum	USINT		4
4	abAryOut	ARRAY [1..5] OF BYTE		
5	abAryOut[1]	BYTE		12
6	abAryOut[2]	BYTE		5
7	abAryOut[3]	BYTE		3
8	abAryOut[4]	BYTE		10
9	abAryOut[5]	BYTE		0

- **Precautions**
  - If the value of "Num" is 0, the return value is TRUE, and "AryOut[ ]" remains unchanged.
  - When the value of "Num" exceeds the valid range, the return value is FALSE, and "AryOut[ ]" remains unchanged.
  - When the value of "Num" exceeds the valid range of "AryOut[ ]", the return value is FALSE, and "AryOut[ ]" remains unchanged.
  - If the dimension of "AryOut[ ]" is greater than 3, the return value is FALSE, and "AryOut[ ]" remains unchanged.

18.1.2.35 DispartReal

Separates a real number into the signed mantissa and the exponent.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DispartReal	Separates mantissa and exponent	FC		:= DispartReal(In := , Fraction => , Exponent =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Real Number	Follows data type	-	A real number to be separated
Output	Fraction	Signed mantissa part	Follows data type	0	Signed mantissa part
	Exponent	Exponent part	Follows data type	0	Exponent part

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–
Fraction	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–
Exponent	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

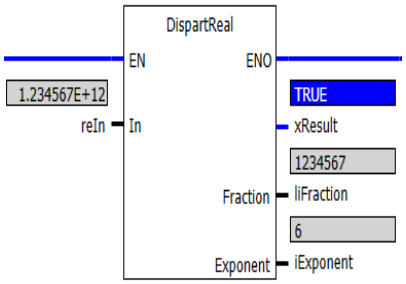
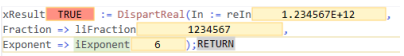
- **Function**

Separates the real number "In" into the signed mantissa part "Fraction" and the exponent part "Exponent".

When "In" is of REAL type, "Fraction" is a 7-digit integer; when "In" is of LREAL type, "Fraction" is a 15-digit integer. The valid range of "Fraction" differs depending on the data type of "In".

Data Type of "In"	Valid Range of "Fraction"
REAL	-9999999 to 9999999
LREAL	-999999999999999 to 999999999999999

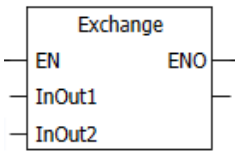
- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; reIn: REAL := 123.4567e+10; liFraction: LINT; iExponent: INT; END_VAR	
Program		
Operation result		

18.1.2.36 Exchange

Exchanges the values of two variables.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
Exchange	Variable exchange	FC		:= Exchange(InOut1 := , InOut2 :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	InOut1	Exchange object 1	Follows data type	-	Exchange object 1
	InOut2	Exchange object 2	Follows data type	-	Exchange object 2

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Return value	Follows data type	-	Return value.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
InOut1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	A specified member of an enumeration or a structure																			
InOut2	Same data type as "InOut1"																			

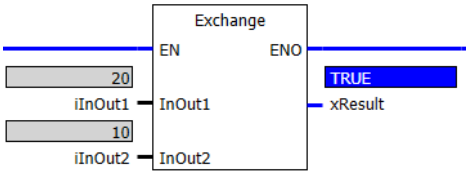
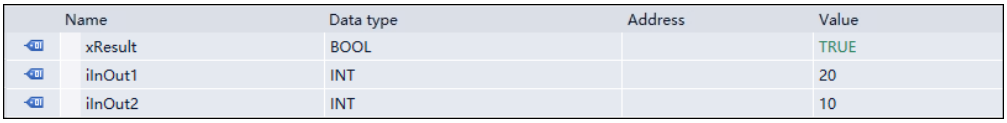
- **Function**

  Exchanges the values of "InOut1" and "InOut2".

- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	iInOut1: INT := 10;
	iInOut2: INT := 20;
	END_VAR	

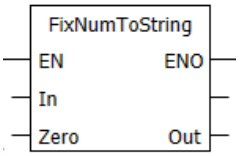
Program		xResult TRUE := Exchange(InOut1 := iInOut1 20, InOut2 := iInOut2 10);RETURN
Operation result		

- **Precautions**
  - The data types of "InOut1" and "InOut2" must be consistent; otherwise, an exception will occur during compilation.
  - If "InOut1" and "InOut2" are both of STRING type, and the string length of one is greater than the other, the function returns FALSE, and "InOut1" and "InOut2" remain unchanged.

18.1.2.37 FixNumToString

Converts a signed fixed-decimal number to a decimal string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FixNumToString	Fixed-decimal number-to-string conversion	FC		:= FixNumToString(In := , Zero := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Fixed-decimal number	Follows data type	0	A signed fixed-decimal number
	Zero	Zero supplement	Follows data type	0	Supplement of zeros if there are less than 3 decimal digits TRUE: Supplements with "0". FALSE: Do not supplement with "0".
Output	Out	Decimal string	Follows data type	''	A decimal string to be output

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Zero	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

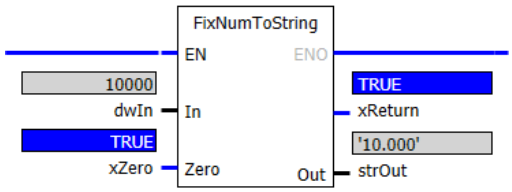
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**

  Converts the signed fixed-decimal number "In" to a decimal string "Out". The conversion steps are as follows:

  1. Converts "In" from hexadecimal format to decimal format.
  2. Divides this value by 1000.
  3. When there is less than 3 decimal places in "In", "Zero" specifies whether to supplement decimal places of "Out" with "0" to make the value with three decimal digits. If the value of "Zero" is TRUE, supplement "Out" with "0".
  4. "Out" ends with a NULL character.

- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; dwIn: DWORD := 10000; xZero : BOOL; strOut : STRING[12]; END_VAR	
Program		xReturn TRUE := FixNumToString(In := dwIn 10000 , Zero := xZero TRUE , Out => strOut '10.000');RETURN
Operation result		

		Name	Data type	Address	Value
1		dwIn	DWORD		10000
2		xZero	BOOL		TRUE
3		strOut	STRING(12)		'10.000'
4		xReturn	BOOL		TRUE

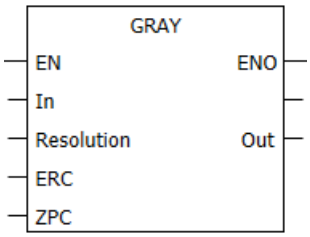
- **Precautions**

  If the conversion result exceeds the valid range of "Out", the function returns FALSE and "Out" remains unchanged.

18.1.2.38 GRAY

Converts a Gray code to an angle.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
Gray	Gray Code Conversion	FC		:= GRAY(In := , Resolution := , ERC := , ZPC := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	Follows data type	0	Gray code of the conversion object
	Resolution	Resolution	Follows data type	0	Resolution
	ERC	Encoder remainder correction value	Follows data type	_R256	Encoder remainder correction value
	ZPC	Home correction value	Follows data type	0	Home correction value
Output	Out	Conversion result	Follows data type	0	Conversion result

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Resolution	See function description for detailed functions of the enumeration type_eGRY_RESOLUTION.																			

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
ERC	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
ZPC	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–

- **Function**

  Converts the output value f a rotary encoder ("In") represented in Gray code into an angle value. The conversion result "Out" is in degrees. The data type of "Resolution" is enumeration type_eGRY_RESOLUTION. The meanings of the enumeration values are as follows in the table below.

Enumeration Value	Definition
_R256	256
_R1B	1-bit (2)
_R2B	2-bit (4)
_R3B	3-bit (8)
_R4B	4-bit (16)
_R5B	5-bit (32)
_R6B	6-bit (64)
_R7B	7-bit (128)
_R8B	8-bit (256)
_R9B	9-bit (512)
_R10B	10-bit (1024)
_R11B	11-bit (2048)
_R12B	12-bit (4096)
_R13B	13-bit (8192)
_R14B	14-bit (16384)
_R15B	15-bit (32768)

Enumeration Value	Definition
_R360	360
_R720	720
_R1024	1024

Gray code is a type of alternate binary code. Two adjacent values in Gray code differ by only one binary bit, such as 0 and 1, and 1 and 2.

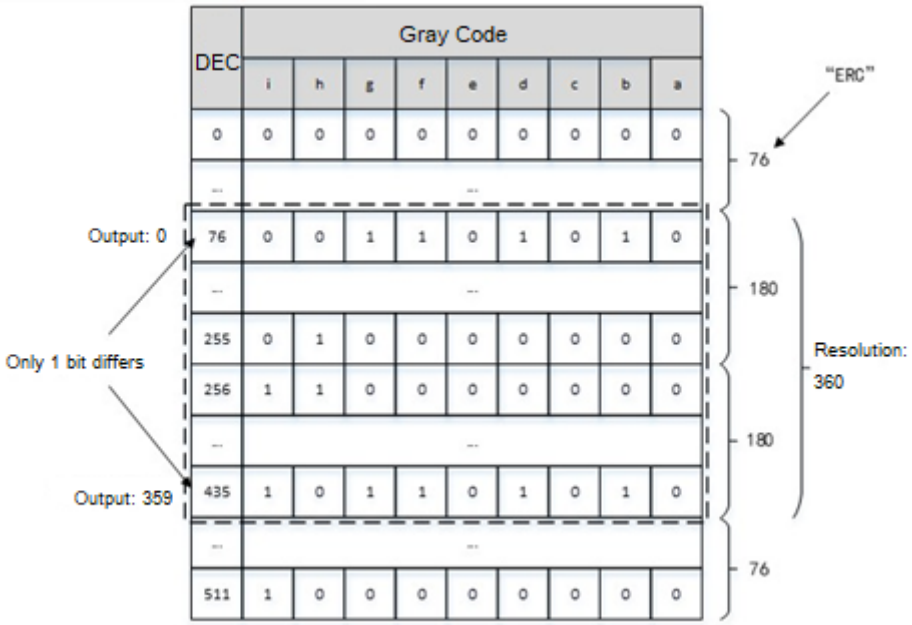
Gray code is commonly used in outputs of absolute encoders.

The following table shows the 4-bit binary code and Gray code.

Decimal Number	Binary Code				Gray Code			
	2^3	2^2	2^1	2^0	d	c	b	a
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

With the use of Gray code, when the encoder output value increases or decreases by 1, only one bit changes, thus preventing incorrect output values. The encoder remainder correction value "ERC" specifies the Gray code range when the encoder resolution is not a power of 2, ensuring that the maximum and minimum encoder output values in Gray code differ by only one bit.

Take an absolute encoder with a resolution of 360 as an example. The Gray code requires 9 bits, and the range of values representable by 9 bits is 0 to 511. In this case, Gray code is used within a range of 180 from the center of 0 to 511, i.e., from 76 to 435. Thus, the Gray code representing output value 0 is 001101010 (76 decimal), and the Gray code representing output value 359 is 101101010 (435 decimal), differing by only one bit. At this point, the encoder remainder correction value "ERC" is 76.



The home correction value "ZPC" is set to offset the initial angle of a rotary encoder. For example, to offset the initial angle of a rotary encoder with a resolution of 256 by 90 degrees, the value of "ZPC" is 256 × (90 / 360) = 64.

The following table shows an example when "In" = WORD#16#123, "Resolution" = _R9B, "ERC" = UINT#0, and "ZPC" = UINT#88.

Since the resolution is 9 bits, one unit of Gray code corresponds to 360° / 29 = 0.703125°.

The decimal value corresponding to Gray code 16#123 is 450. Therefore, the angle value before correction is 0.703125° × 450 = 316.40625°.

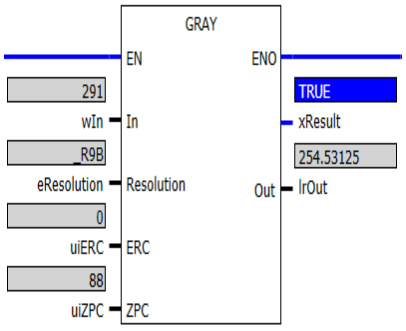
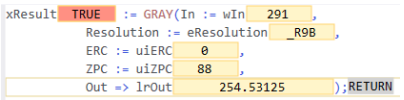
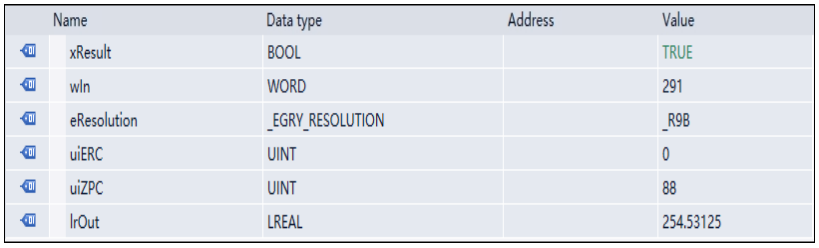
The value of "ERC" is 0, and the value of "ZPC" is 88. Therefore, the corrected angle value is 316.40625° - (0 + 88) × 0.703125° = 254.53125°.

The value of "Out" is LREAL#254.53125.

- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	wIn: WORD := 16#123;
	eResolution: _eGRY_RESOLUTION := _eGRY_RESOLUTION._R9B;
	uiERC:UINT := 0;
	uiZPC: UINT := 88;
	lrOut: LREAL;
	END_VAR	

	LD	ST
Program		
Operation result		

	Name	Data type	Address	Value
	xResult	BOOL		TRUE
	wIn	WORD		291
	eResolution	_EGRY_RESOLUTION		_R9B
	uiERC	UINT		0
	uiZPC	UINT		88
	lrOut	LREAL		254.53125

- **Precautions**

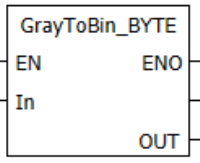
  An exception occurs under the following conditions. The function returns FALSE, and "Out" is initialized to 0.

  - The value of "ERC" is greater than "Resolution".
  - The value of "ZPC" is greater than "Resolution".
  - The value of "In" after converted to a bit string is less than "ERC".
  - The value of the bit string corrected with "ERC" is greater than "Resolution".

18.1.2.39 GrayToBin_BYTE

Converts a Gray code to a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GrayToBin_BYTE	Gray code-to-bit string conversion	FC		:= GrayToBin_BYTE(In := , OUT =>);

- **Related variables**

  Variable

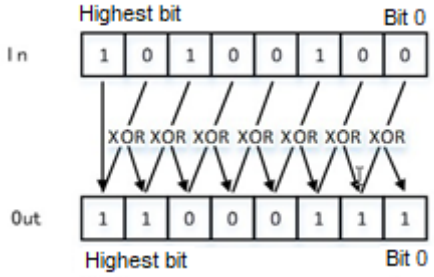
Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	Follows data type	0	Conversion object
Output	Out	Conversion result	Follows data type	-	Conversion result

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - Converts the Gray code object "In" to a bit string.
  - The conversion steps are described below.

1. The most significant bit (bit 7) of In is directly used as the most significant bit (bit 7) of Out.
2. The XOR operation result on bit 6 of "In" and bit 7 of "Out" is assigned to bit 6 of "Out".
3. The same method is used to determine the values through the least significant bit (bit 0) of "Out".



- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; byIn: BYTE := 2#10100100; byOut: BYTE; END_VAR	
Program	GrayToBin_BYTE EN ENO byIn — In xResult OUT — byOut	xResult:= GrayToBin_BYTE(In := byIn, OUT => byOut);
Operation result	Name / Data type / Address / Value xResult / BOOL / / TRUE byIn / BYTE / / 2#10100100 byOut / BYTE / / 2#11000111	

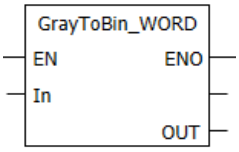
- **Precautions**

The data types of "In" and "OUT" must be consistent; otherwise, an exception will occur during compilation.

18.1.2.40 GrayToBin_WORD

Converts a Gray code to a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GrayToBin_WORD	Gray code-to-bit string conversion	FC		:= GrayToBin_WORD(In := , OUT =>);

- **Related variables**

Variable

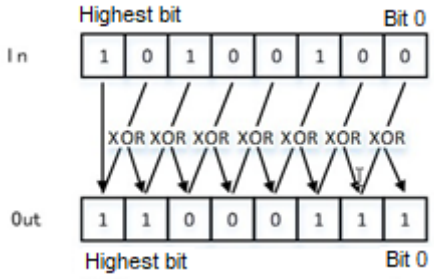
Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	Follows data type	0	Conversion object
Output	Out	Conversion result	Follows data type	-	Conversion result

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - Converts the Gray code object "In" to a bit string.
  - The conversion steps are described below.

1. The most significant bit (bit 7) of In is directly used as the most significant bit (bit 7) of Out.
2. The XOR operation result on bit 6 of "In" and bit 7 of "Out" is assigned to bit 6 of "Out".
3. The same method is used to determine the values through the least significant bit (bit 0) of "Out".



- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; wIn: WORD := 2#0000000010100100; wOut: WORD; END_VAR	
Program	GrayToBin_WORD EN ENO wIn In xResult OUT wOut	xResult := GrayToBin_WORD(In := wIn, OUT => wOut);
Operation result	Name / Data type / Address / Value xResult / BOOL / / TRUE wIn / WORD / / 2#0000000010100100 wOut / WORD / / 2#0000000011000111	

- **Precautions**

  The data types of "In" and "OUT" must be consistent; otherwise, an exception will occur during compilation.

18.1.2.41 GrayToBin_DWORD

Converts a Gray code to a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GrayToBin_DWORD	Gray code-to-bit string conversion	FC	GrayToBin_DWORD EN ENO In OUT	:= GrayToBin_DWORD(In := , OUT =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	Follows data type	0	Conversion object

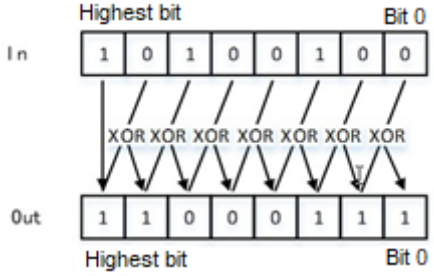
Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Conversion result	Follows data type	-	Conversion result

Variable types

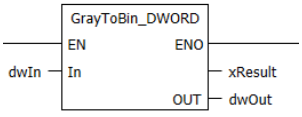
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

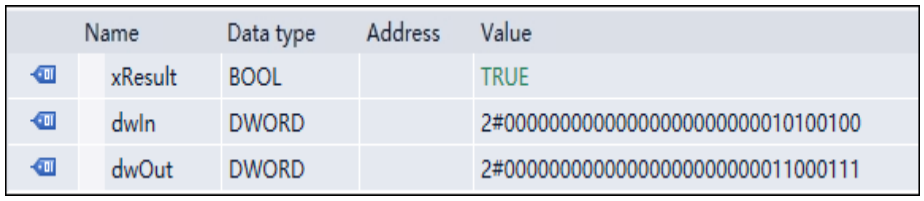
- **Function**
  - Converts the Gray code object "In" to a bit string.
  - The conversion steps are described below.

1. The most significant bit (bit 7) of In is directly used as the most significant bit (bit 7) of Out.
2. The XOR operation result on bit 6 of "In" and bit 7 of "Out" is assigned to bit 6 of "Out".
3. The same method is used to determine the values through the least significant bit (bit 0) of "Out".



- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; dwIn: DWORD := 2#00000000000000000000000010100100; dwOut: DWORD; END_VAR	
Program		xResult := GrayToBin_DWORD(In := dwIn, OUT => dwOut);

	LD	ST
Operation result		

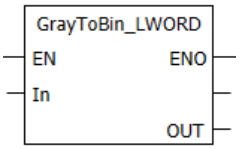
- **Precautions**

  The data types of "In" and "OUT" must be consistent; otherwise, an exception will occur during compilation.

18.1.2.42 GrayToBin_LWORD

Converts a Gray code to a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GrayToBin_LWORD	Gray code-to-bit string conversion	FC		:= GrayToBin_LWORD(In := , OUT =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	Follows data type	0	Conversion object
Output	Out	Conversion result	Follows data type	-	Conversion result

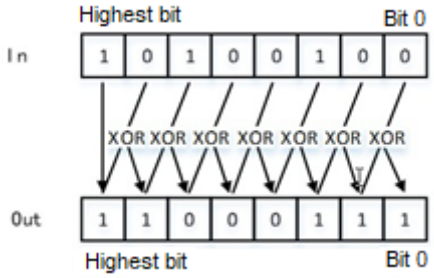
  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - Converts the Gray code object "In" to a bit string.

◦ The conversion steps are described below.

1. The most significant bit (bit 7) of In is directly used as the most significant bit (bit 7) of Out.
2. The XOR operation result on bit 6 of "In" and bit 7 of "Out" is assigned to bit 6 of "Out".
3. The same method is used to determine the values through the least significant bit (bit 0) of "Out".



- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; lwIn: LWORD := 2#0000000000000000000000000000000000000000000000000000000010100100; lwOut: LWORD; END_VAR	
Program	GrayToBin_LWORD EN ENO lwIn — In — xResult OUT — lwOut	xResult := GrayToBin_LWORD(In := lwIn, OUT => lwOut);
Operation result	Name / Data type / Address / Value xResult / BOOL / / TRUE lwIn / LWORD / / 2#0000000000000000000000000000000000000000000000000000000010100100 lwOut / LWORD / / 2#0000000000000000000000000000000000000000000000000000000011000111	

- **Precautions**

  The data types of "In" and "OUT" must be consistent; otherwise, an exception will occur during compilation.

18.1.2.43 LineToColm

Takes the bits from a bit string and outputs them to the specified bit position in array elements.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LineToColm	Line-to-column conversion	FC	LineToColm EN ENO In InOut Size Pos	:= LineToColm(In := , InOut := , Size := , Pos :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	Follows data type	-	Conversion object
	Size	Number of elements in the conversion result	Follows data type	1	Number of elements in the conversion result
	Pos	Bit position to convert	Follows data type	0	Bit position to convert
Input/Output	InOut[ ] array	Conversion result array	Follows data type	-	An array indicating the conversion result
Output	Out	Return value	Follows data type	-	Return value.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
InOut[ ] array	–	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Pos	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

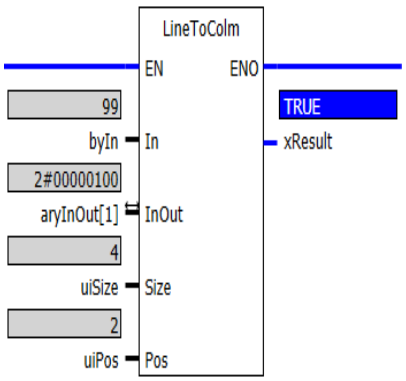
  Takes the bits from a bit string and outputs them to the specified bit position in array elements.

  - Extracts "Size" bits from the least significant bit of "In" and decomposes them into individual bits.
  - Each extracted bit is stored in the "Pos"-bit of each element in the result array InOut[ ], starting from InOut[0]. A total of "Size" array elements are saved.
  - For bits in the result array InOut[ ] that are not saved, their values remain unchanged.

- **Program example**

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    xResult: BOOL;
    byIn: BYTE := 99;
    aryInOut: ARRAY[0..4] OF BYTE;
    uiSize: UINT := 4;
    uiPos: UINT := 2;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>
</td><td>

```
xResult  TRUE  := LineToColm(In := byIn  99 ,
          InOut := aryInOut[1]  2#00000100 ,
          Size := uiSize  4 ,
          Pos := uiPos  2 );RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

Name	Data type	Address	Value
xResult	BOOL		TRUE
byIn	BYTE		99
aryInOut	ARRAY [0..4] OF BYTE		
aryInOut[0]	BYTE		2#00000000
aryInOut[1]	BYTE		2#00000100
aryInOut[2]	BYTE		2#00000100
aryInOut[3]	BYTE		2#00000000
aryInOut[4]	BYTE		2#00000000
uiSize	UINT		4
uiPos	UINT		2

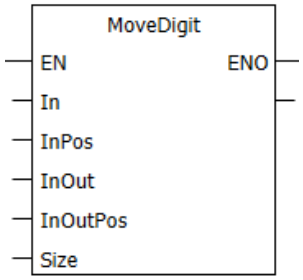
</td></tr>
</table>

- **Precautions**
  - When the value of "Size" is 0, the value of InOut[ ] remains unchanged, and the function returns TRUE.
  - When the value of "Size" exceeds the valid range, the value of InOut[ ] remains unchanged, and the function returns FALSE.
  - When the value of "Pos" exceeds the valid range, the value of InOut[ ] remains unchanged, and the function returns FALSE.
  - When the value of "Size" exceeds the array area of InOut[ ], the value of InOut[ ] remains unchanged, and the function returns FALSE.
  - If the array exceeds 3 dimensions, the value of InOut[ ] remains unchanged, and the function returns FALSE.

18.1.2.44 MoveDigit

Moves multiple digits (4 bits per digit) in a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
MoveDigit	Digit transfer	FC		:= MoveDigit(In := , InPos := , InOut := , InOutPos := , Size :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Move source	Follows data type	-	Move source
	InPos	Digit position of the move source	Follows data type	0	Digit position of the move source
	InOut	Move destination	Follows data type	-	Move destination
	InOutPos	Digit position of the move destination	Follows data type	0	Digit position of the move destination
	Size	Number of digits to move	Follows data type	1	Number of digits to move
Output	Out	Return value	Follows data type	-	Return value.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
InPos	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
InOut	–	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
InOutPos	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

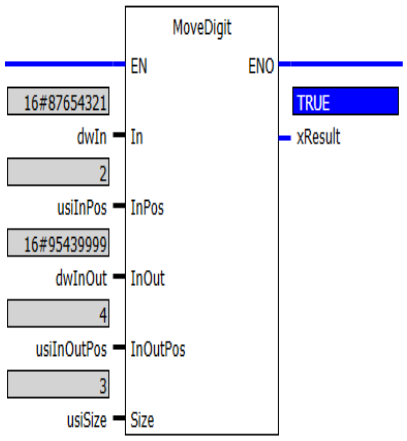
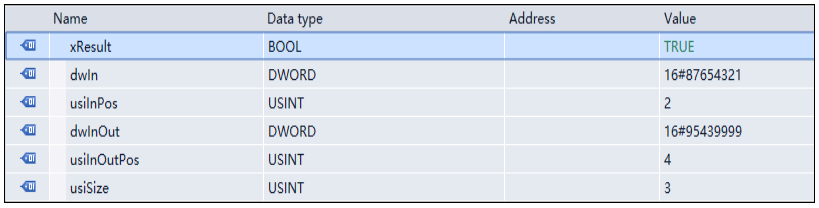
- **Function**

  Transfers "Size" number of digits starting from the digit position "InPos" in the source "In" to the digit position "InOutPos" in the destination "InOut".

- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	dwIn: DWORD := 16#87654321;
	usiInPos: USINT := 2;
	dwInOut: DWORD := 16#99999999;
	usiInOutPos: USINT := 4;
	usiSize: USINT := 3;
	END_VAR	

Program		xResult TRUE := MoveDigit(In := dwIn 16#87654321, InPos := usiInPos 2, InOut := dwInOut 16#95439999, InOutPos := usiInOutPos 4, Size := usiSize 3);RETURN
Operation result		

Name	Data type	Address	Value
xResult	BOOL		TRUE
dwIn	DWORD		16#87654321
usiInPos	USINT		2
dwInOut	DWORD		16#95439999
usiInOutPos	USINT		4
usiSize	USINT		3

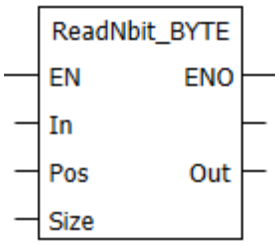
- **Precautions**
  - If the specified destination digit exceeds the most significant bit of "InOut", the remaining data will be stored in the least significant bit of "InOut".
  - If the specified source digit exceeds the most significant bit of "In", the remaining data will be moved to the least significant bit of "In".

- When the value of "Size" is 0, the function returns TRUE, and "InOut" remains unchanged.
- An exception occurs under the following conditions. The function returns FALSE, and "InOut" remains unchanged.
  - The value of "InPos" exceeds the valid range.
  - The value of "InOutPos" exceeds the valid range.
  - The value of "Size" exceeds the valid range.

18.1.2.45 ReadNbit_BYTE

Reads multiple bits from a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ReadNbit_BYTE	N-bit read group	FC		:= ReadNbit_BYTE(In := , Pos := , Size := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Read source	Follows data type	0	A bit string to be read and written to "Out"
	Pos	Read position	Follows data type	0	Bit position to be read
	Size	Read size	Follows data type	1	The number of bits to be read
Output	Out	Read result	Follows data type	-	Write result

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Pos	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

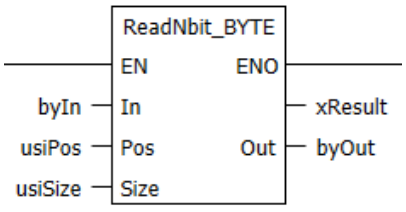
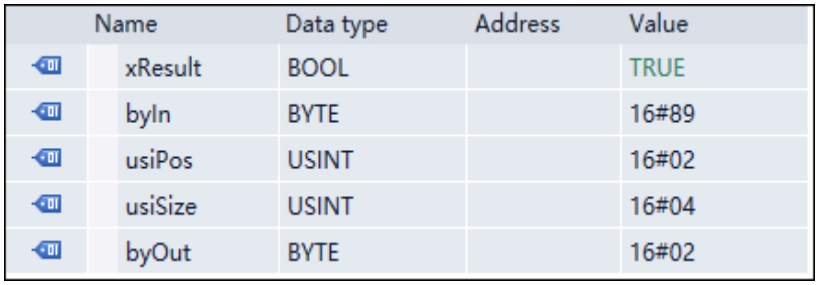
- **Function**

  Reads "Size" bits from the bit string "In" starting at read position "Pos", and stores the result in "Out".

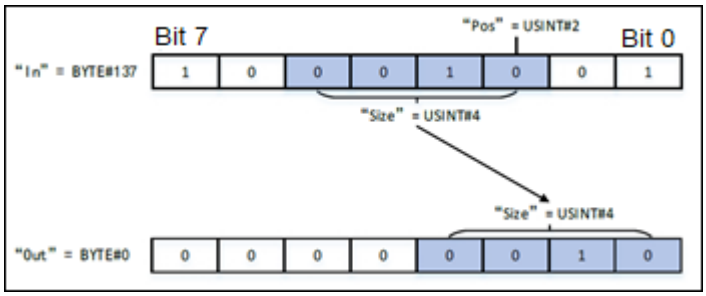
- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	byIn: BYTE := 137;
	usiPos: USINT := 2;
	usiSize: USINT := 4;
	byOut: BYTE;
	END_VAR	

Program		xResult := ReadNbit_BYTE(In := byIn, Pos := usiPos, Size := usiSize, Out => byOut);
Operation result		

Name	Data type	Address	Value
xResult	BOOL		TRUE
byIn	BYTE		16#89
usiPos	USINT		16#02
usiSize	USINT		16#04
byOut	BYTE		16#02

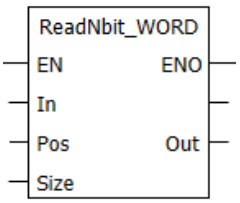
	LD	ST
Principle		

- **Precautions**
  - The data types of "In" and "Out" must be the same; otherwise, the function returns FALSE.
  - When the value of "Size" is 0, the value of "Out" is 16#0.
  - An exception occurs and the function returns FALSE, with "Out" set to 16#0, under the following conditions:
    - The value of "Pos" exceeds the valid range.
    - The value of "Size" exceeds the valid range.
    - The bits to be read specified by "Pos" and "Size" exceed the number of bits in "In".

18.1.2.46 ReadNbit_WORD

Reads multiple bits from a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ReadNbit_WORD	N-bit read group	FC		:= ReadNbit_WORD(In := , Pos := , Size := , Out =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Read source	Follows data type	0	A bit string to be read and written to "Out"
	Pos	Read position	Follows data type	0	Bit position to be read
	Size	Read size	Follows data type	1	The number of bits to be read
Output	Out	Read result	Follows data type	-	Write result

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Pos	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Reads "Size" bits from the bit string "In" starting at read position "Pos", and stores the result in "Out".

- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	wIn: WORD := 137;
	usiPos: USINT := 2;
	usiSize: USINT := 4;
	wOut: WORD;
	var1: BOOL;
	END_VAR	

Program	ReadNbit_WORD EN ENO wIn — In — xResult usiPos — Pos — Out — wOut usiSize — Size	xResult := ReadNbit_WORD(In := wIn, Pos := usiPos, Size := usiSize, Out => wOut);
Operation result	Name / Data type / Address / Value xResult / BOOL / / TRUE wIn / WORD / / 16#0089 usiPos / USINT / / 16#02 usiSize / USINT / / 16#04 wOut / WORD / / 16#0002	

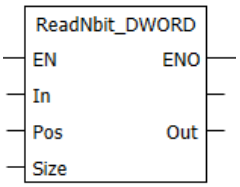
- **Precautions**
  - The data types of "In" and "Out" must be the same; otherwise, the function returns FALSE.

- When the value of "Size" is 0, the value of "Out" is 16#0.
- An exception occurs and the function returns FALSE, with "Out" set to 16#0, under the following conditions:
  - The value of "Pos" exceeds the valid range.
  - The value of "Size" exceeds the valid range.
  - The bits to be read specified by "Pos" and "Size" exceed the number of bits in "In".

18.1.2.47 ReadNbit_DWORD

Reads multiple bits from a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ReadNbit_DWORD	N-bit read group	FC		:= ReadNbit_DWORD(In := , Pos := , Size := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Read source	Follows data type	0	A bit string to be read and written to "Out"
	Pos	Read position	Follows data type	0	Bit position to be read
	Size	Read size	Follows data type	1	The number of bits to be read
Output	Out	Read result	Follows data type	-	Write result

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Pos	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

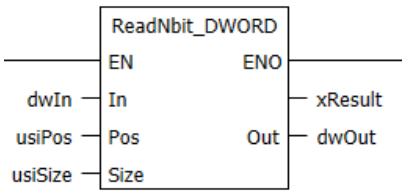
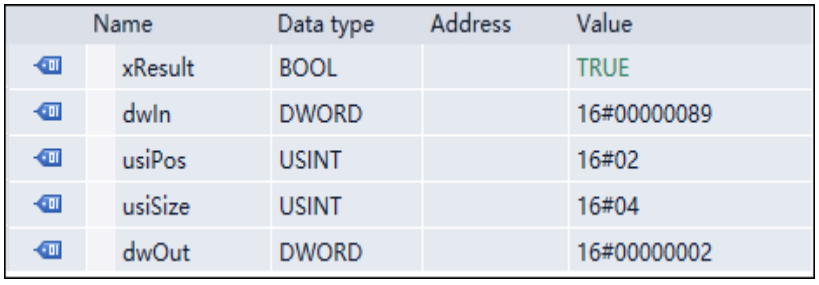
- **Function**

  Reads "Size" bits from the bit string "In" starting at read position "Pos", and stores the result in "Out".

- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	dwIn: DWORD := 137;
	usiPos: USINT := 2;
	usiSize: USINT := 4;
	dwOut: DWORD;
	END_VAR	

Program		xResult := ReadNbit_DWORD(In := dwIn, Pos := usiPos, Size := usiSize, Out => dwOut);
Operation result		

	Name	Data type	Address	Value
	xResult	BOOL		TRUE
	dwIn	DWORD		16#00000089
	usiPos	USINT		16#02
	usiSize	USINT		16#04
	dwOut	DWORD		16#00000002

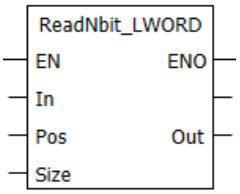
- **Precautions**
  - The data types of "In" and "Out" must be the same; otherwise, the function returns FALSE.
  - When the value of "Size" is 0, the value of "Out" is 16#0.

- An exception occurs and the function returns FALSE, with "Out" set to 16#0, under the following conditions:
  - The value of "Pos" exceeds the valid range.
  - The value of "Size" exceeds the valid range.
  - The bits to be read specified by "Pos" and "Size" exceed the number of bits in "In".

18.1.2.48 ReadNbit_LWORD

Reads multiple bits from a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ReadNbit_LWORD	N-bit read group	FC		:= ReadNbit_LWORD(In := , Pos := , Size := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Read source	Follows data type	0	A bit string to be read and written to "Out"
	Pos	Read position	Follows data type	0	Bit position to be read
	Size	Read size	Follows data type	1	The number of bits to be read
Output	Out	Read result	Follows data type	-	Write result

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Pos	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

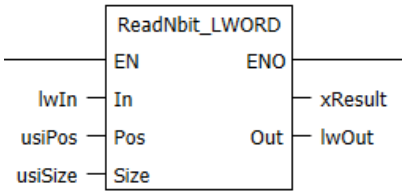
  Reads "Size" bits from the bit string "In" starting at read position "Pos", and stores the result in "Out".

- **Program example**

	LD	ST
Define variables		

```
VAR
    xResult: BOOL;
    lwIn: LWORD := 137;
    usiPos: USINT := 2;
    usiSize: USINT := 4;
    lwOut: LWORD;
END_VAR
```

Program




```
xResult := ReadNbit_LWORD(In := lwIn,
    Pos := usiPos,
    Size := usiSize,
    Out => lwOut);
```

Operation result

	Name	Data type	Address	Value
	xResult	BOOL		TRUE
	lwIn	LWORD		16#0000000000000089
	usiPos	USINT		16#02
	usiSize	USINT		16#04
	lwOut	LWORD		16#0000000000000002

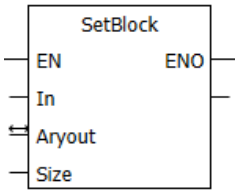
- **Precautions**
  - The data types of "In" and "Out" must be the same; otherwise, the function returns FALSE.
  - When the value of "Size" is 0, the value of "Out" is 16#0.

- An exception occurs and the function returns FALSE, with "Out" set to 16#0, under the following conditions:
  - The value of "Pos" exceeds the valid range.
  - The value of "Size" exceeds the valid range.
  - The bits to be read specified by "Pos" and "Size" exceed the number of bits in "In".

18.1.2.49 SetBlock

Moves the input value to multiple array elements.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SetBlock	Block setting	FC		:= SetBlock(In := , Aryout := , Size :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Move source	Follows data type	-	Move source
	Size	Number of elements to move	Follows data type	1	Number of array elements to move
Input/Output	AryOut[ ] array	Move destination array	Follows data type	-	Move destination array

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration, a structure, or a structure member can also be specified.																			
Size	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
AryOut[ ]	An array of elements with the same data type as "In"																			

- **Function**

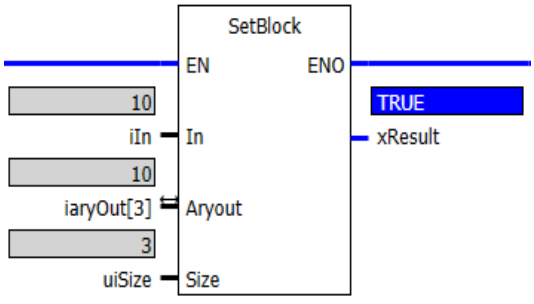
  Moves the value of "In" to "Size" elements in destination AryOut[ ] starting from AryOut[0].

- **Program example**

Define variables

```
VAR
    xResult: BOOL;
    iIn: INT := 10;
    iaryOut: ARRAY[0..9] OF INT := [0,1,2,3,4,5,6,7,8,9];
    uiSize: UINT := 3;
END_VAR
```

Program

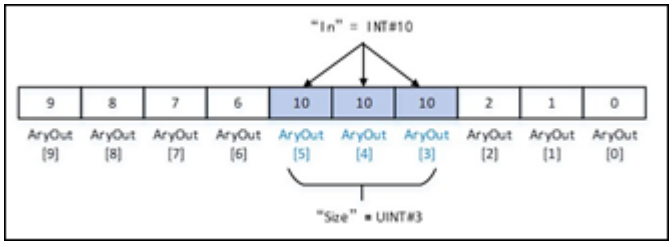



```
xResult TRUE := SetBlock(In := iIn 10,
          Aryout := iaryOut[3] 10,
          Size := uiSize 3 );RETURN
```

Operation result

	Name	Data type	Address	Value
	xResult	BOOL		TRUE
	iIn	INT		10
▾	iaryOut	ARRAY [0..9] OF INT		
	iaryOut[0]	INT		0
	iaryOut[1]	INT		1
	iaryOut[2]	INT		2
	iaryOut[3]	INT		10
	iaryOut[4]	INT		10
	iaryOut[5]	INT		10
	iaryOut[6]	INT		6
	iaryOut[7]	INT		7
	iaryOut[8]	INT		8
	iaryOut[9]	INT		9
	uiSize	UINT		3

Principle analysis



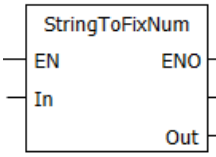
- **Precautions**
  - A compilation exception occurs when the data types of "In" and AryOut[ ] differ.

- When both "In" and AryOut[ ] are of STRING type, their sizes must be consistent. If the length of "In" exceeds the element length of AryOut[ ], the function returns FALSE, and AryOut[ ] remains unchanged.
- When the value of "Size" is 0, the function returns TRUE, and AryOut[ ] remains unchanged.
- When the value of "Size" exceeds the array area of AryOut[ ], the function returns FALSE, and AryOut[ ] remains unchanged.
- A compilation exception occurs when "In" and AryOut[ ] are of unsupported data types.
- If the array exceeds 3 dimensions, the function returns FALSE, and AryOut[ ] remains unchanged.

18.1.2.50 StringToFixNum

Converts a decimal string to a signed fixed-decimal number.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StringToFixNum	Conversion from a string to a fixed-decimal number	FC		:= StringToFixNum(In := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Decimal string	Follows data type	''	Decimal string
Output	Out	Fixed-decimal number	Follows data type	0	Fixed-decimal number

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
Out	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

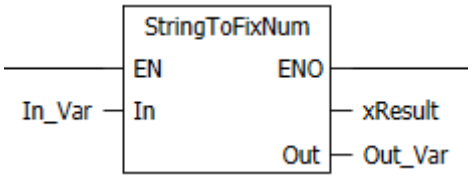
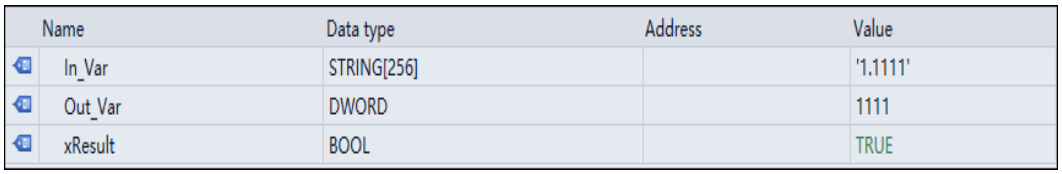
Converts the decimal string "In" to a fixed-decimal number. The conversion steps are as follows:

1. Multiplies the value represented by "In" by 1000.
2. Truncates the decimal places of this value.
3. Converts this value into a 32-bit hexadecimal number (DWORD type).

Name	Format
Operator	Consecutive spaces after the sign can be ignored.
Integer part	The integer part is the consecutive digits (0 to 9) after the sign and before the decimal point. The sign can be omitted. Consecutive spaces between the sign and the integer part can be ignored. When there is no decimal point or fractional part, the integer part is the characters up to the exponent. When there is no decimal point, fractional part, or exponent part, the integer part is the characters up to the end of the string. The integer part cannot be omitted. The maximum number of digits in the integer part is the maximum string length of 1985 minus the total number of bytes in the sign, decimal point, fractional part, exponent part, and spaces before and after the sign.
Decimal point	The decimal point is the single dot (".") following the integer part. When there is no fractional part, the decimal point must be omitted.
Fractional part	The fractional part is the consecutive digits (0 to 9) after the decimal point and before the exponent part. When there is no exponent part, the fractional part is the characters up to the end of the string. The fractional part can be omitted. When there is no decimal point, there is no fractional part. The maximum number of digits in the fractional part is 15.

Name	Format
Exponent part	The exponent part consists of a single "e" or "E" followed by a single "+" or "-" and then consecutive digits (0 to 9) to the end of the string. When there is no fractional part, the exponent part is the string after the decimal point. When there is no decimal point or fractional part, the exponent part is the string following the integer part. The exponent part can be omitted. The maximum number of digits in the exponent part is 3.

- **Program example**

	LD	ST
Define variables	VAR In_Var: STRING[256] := '1.1111'; Out_Var: DWORD; xResult: BOOL; END_VAR	
Program		xResult := StringToFixNum(In := In_Var, Out => Out_Var);
Operation result		

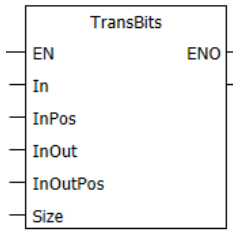
Name	Data type	Address	Value
In_Var	STRING[256]		'1.1111'
Out_Var	DWORD		1111
xResult	BOOL		TRUE

- **Precautions**
  - The digits after the fourth decimal in In digit are truncated.
  - The underscore (16#5F) in "In" is ignored.
  - If "In" is not NULL-terminated, the function returns FALSE.
  - If "In" contains characters that cannot be converted to numbers, the function returns FALSE.
  - If the conversion result exceeds the valid range of "Out", the function returns FALSE.

18.1.2.51 TransBits

Moves multiple bits in a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
TransBits	Multiple bit transfer	FC		:= TransBits(In := , InPos := , InOut := , InOutPos := , Size :=);

- **Related variables**

Variable

Input	Variable	Name	Valid Range	Default Value	Description
Input	In	Move source	Follows data type	-	Move source
	InPos	Bit position in the move source	0 to ("In" bit count - 1)	0	Bit position in "In" to transfer
	InOut	Move destination	Follows data type	-	Move destination
	InOutPos	Bit position in the move destination	0 to ("InOut" bit count - 1)	0	Bit position in "InOut" to receive
	Size	Number of bits to transfer	0 to the minimum bit count of "In" and "InOut"	1	Number of bits to transfer

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
InPos	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
InOut	–	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
InOutPos	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

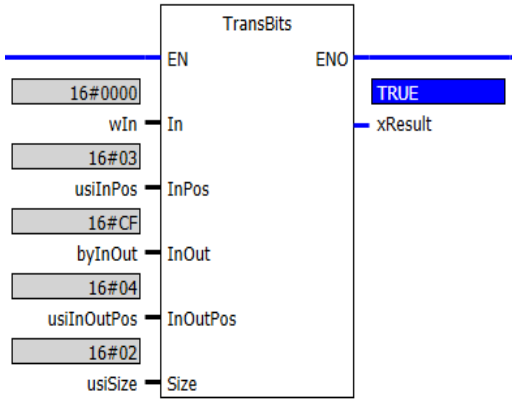
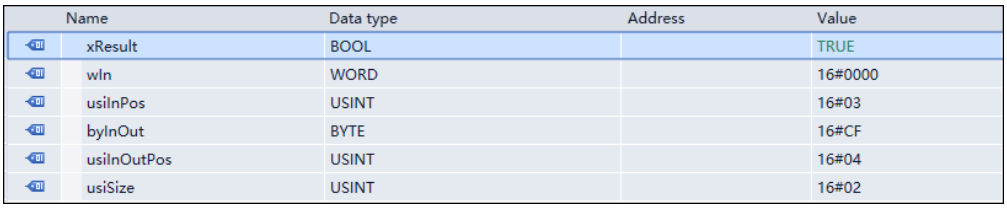
- **Function**

  Moves "Size" bits starting from position "InPos" in the source "In" to bit position "InOutPos" in the destination "InOut".

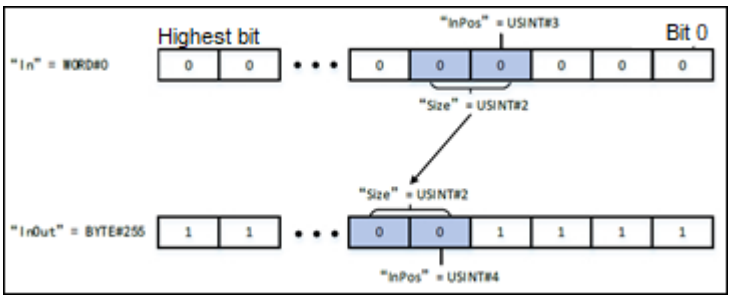
- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	wIn: WORD := 0;
	usiInPos: USINT := 3;
	byInOut: BYTE := 255;
	usiInOutPos: USINT := 4;
	usiSize: USINT := 2;
	END_VAR	

Program		xResult TRUE := TransBits(In := wIn 16#0000 , InPos := usiInPos 16#03 , InOut := byInOut 16#CF , InOutPos := usiInOutPos 16#04 , Size := usiSize 16#02);RETURN
Operation result		

Name	Data type	Address	Value
xResult	BOOL		TRUE
wIn	WORD		16#0000
usiInPos	USINT		16#03
byInOut	BYTE		16#CF
usiInOutPos	USINT		16#04
usiSize	USINT		16#02

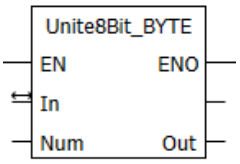
	LD	ST
Principle analysis		

- **Precautions**
  - When the value of "Size" is 0, no bit is moved.
  - The bits in "InOut" unrelated to the move operation remain unchanged.
  - An exception occurs under the following conditions. The function returns FALSE, and output variables remain unchanged.
    - The value of "InPos" exceeds the valid range.
    - The value of "InOutPos" exceeds the valid range.
    - The value of "Size" exceeds the valid range.
    - The value of "InPos" or "Size" exceeds the bit count of "In".
    - The value of "InOutPos" or "Size" exceeds the bit count of "InOut".

18.1.2.52 Unite8Bit_BYTE

Joins bytes of data into a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
Unite8Bit_BYTE	Byte data join group	FC		:= Unite8Bit_BYTE(In := , Num := , Out =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input/Output	In[ ] array	Array of objects to join	Follows data type	-	Array to join
Input	Num	Number of bytes to join	Follows data type	1	Number of bytes to join
Output	Out	Join result	Follows data type	0	A bit string indicating the join result

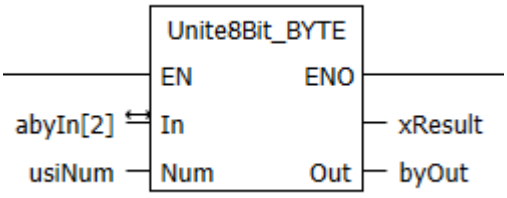
Variable type

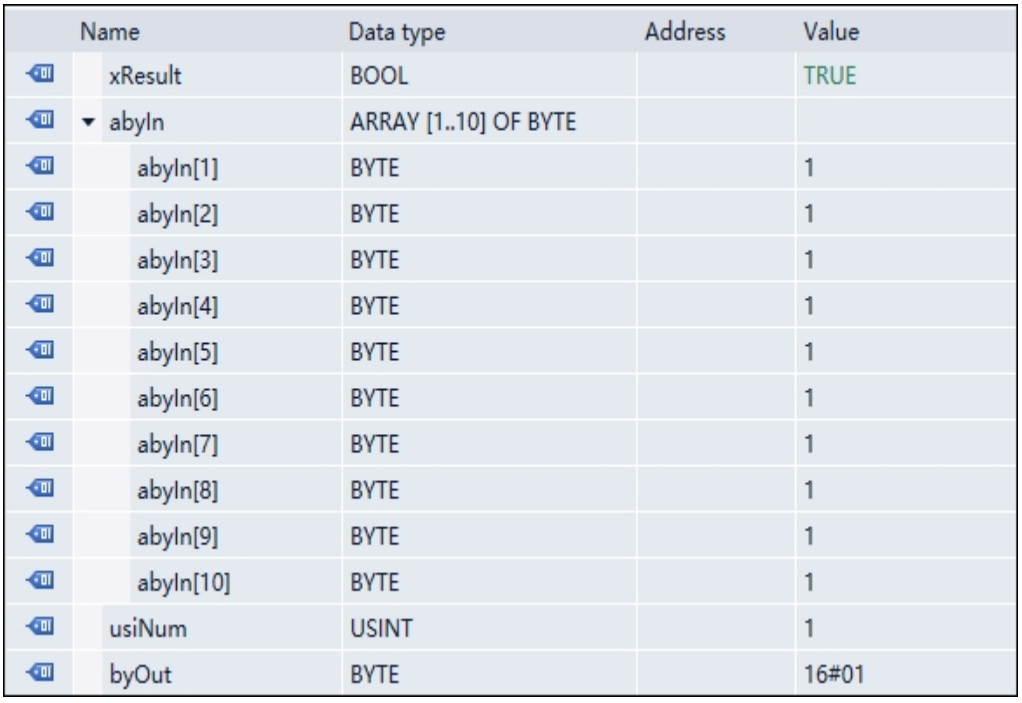
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - Joins the elements in the array In[ ] to generate the bit string "Out" as the join result. The number of array elements to be joined is specified by "Num".
  - Elements from In[0] to In["Num"-1] are joined to generate a bit string of "Num" bytes. If "Num" is less than the length of "Out", zeros are supplemented to the high bits of "Out".
- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	abyIn: ARRAY[1..10] OF BYTE := [10(1)];
	usiNum: USINT := 1;
	byOut: BYTE;
	END_VAR	

Program		xResult := Unite8Bit_BYTE(In := abyIn[2], Num := usiNum, Out => byOut);

	LD	ST
Operation result		

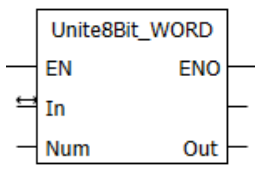
Name	Data type	Address	Value
xResult	BOOL		TRUE
abyIn	ARRAY [1..10] OF BYTE		
abyIn[1]	BYTE		1
abyIn[2]	BYTE		1
abyIn[3]	BYTE		1
abyIn[4]	BYTE		1
abyIn[5]	BYTE		1
abyIn[6]	BYTE		1
abyIn[7]	BYTE		1
abyIn[8]	BYTE		1
abyIn[9]	BYTE		1
abyIn[10]	BYTE		1
usiNum	USINT		1
byOut	BYTE		16#01

- **Precautions**
  - When the value of "Num" is 0, "Out" is initialized to 0, and the function returns TRUE.
  - When the value of "Num" exceeds the valid range, the function returns FALSE, and "Out" is initialized to 0.
  - When the value of "Num" exceeds the array area of In[ ], the function returns FALSE, and "Out" is initialized to 0.
  - If the array exceeds 3 dimensions, the function returns FALSE, and "Out" is initialized to 0.

18.1.2.53 Unite8Bit_WORD

Joins bytes of data into a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
Unite8Bit_WORD	Byte data join group	FC		:= Unite8Bit_WORD(In := , Num := , Out =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input/Output	In[ ] array	Array of objects to join	Follows data type	-	Array to join

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Num	Number of bytes to join	Follows data type	1	Number of bytes to join
Output	Out	Join result	Follows data type	0	A bit string indicating the join result

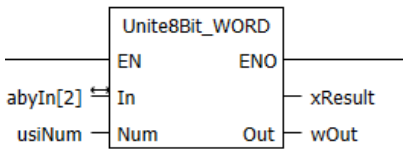
Variable type

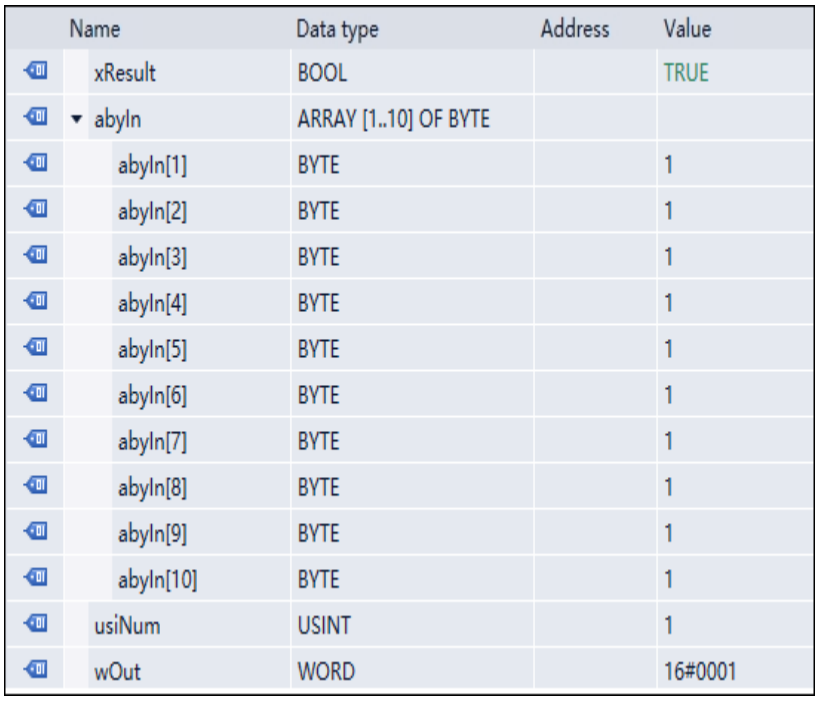
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - Joins the elements in the array In[ ] to generate the bit string "Out" as the join result. The number of array elements to be joined is specified by "Num".
  - Elements from In[0] to In["Num"-1] are joined to generate a bit string of "Num" bytes. If "Num" is less than the length of "Out", zeros are supplemented to the high bits of "Out".
- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	abyIn: ARRAY[1..10] OF BYTE := [10(1)];
	usiNum: USINT := 1;
	wOut: WORD;
	END_VAR	

Program		xResult := Unite8Bit_WORD(In := abyIn[2], Num := usiNum, Out => wOut);

	LD	ST
Operation result		

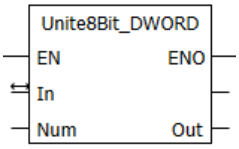
Name	Data type	Address	Value
xResult	BOOL		TRUE
abyIn	ARRAY [1..10] OF BYTE		
abyIn[1]	BYTE		1
abyIn[2]	BYTE		1
abyIn[3]	BYTE		1
abyIn[4]	BYTE		1
abyIn[5]	BYTE		1
abyIn[6]	BYTE		1
abyIn[7]	BYTE		1
abyIn[8]	BYTE		1
abyIn[9]	BYTE		1
abyIn[10]	BYTE		1
usiNum	USINT		1
wOut	WORD		16#0001

- **Precautions**
  - When the value of "Num" is 0, "Out" is initialized to 0, and the function returns TRUE.
  - When the value of "Num" exceeds the valid range, the function returns FALSE, and "Out" is initialized to 0.
  - When the value of "Num" exceeds the array area of In[ ], the function returns FALSE, and "Out" is initialized to 0.
  - If the array exceeds 3 dimensions, the function returns FALSE, and "Out" is initialized to 0.

18.1.2.54 Unite8Bit_DWORD

Joins bytes of data into a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
Unite8Bit_DWORD	Byte data join group	FC		:= Unite8Bit_DWORD(In := , Num := , Out =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input/Output	In[ ] array	Array of objects to join	Follows data type	-	Array to join

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Num	Number of bytes to join	Follows data type	1	Number of bytes to join
Output	Out	Join result	Follows data type	0	A bit string indicating the join result

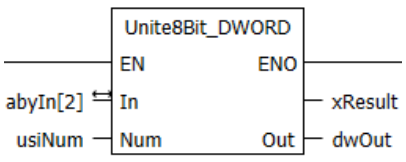
Variable type

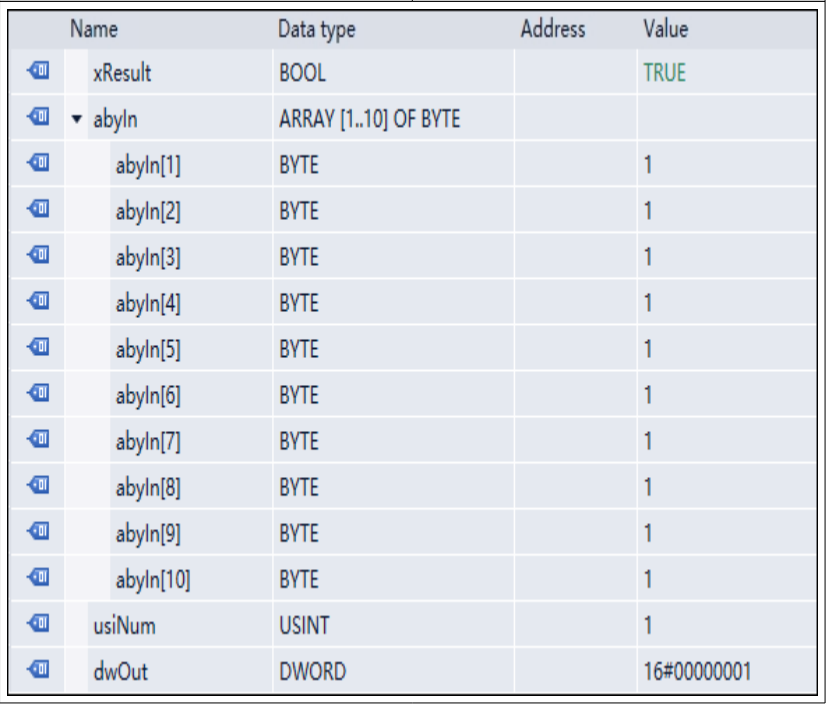
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - Joins the elements in the array In[ ] to generate the bit string "Out" as the join result. The number of array elements to be joined is specified by "Num".
  - Elements from In[0] to In["Num"-1] are joined to generate a bit string of "Num" bytes. If "Num" is less than the length of "Out", zeros are supplemented to the high bits of "Out".
- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	abyIn: ARRAY[1..10] OF BYTE := [10(1)];
	usiNum: USINT := 1;
	dwOut: DWORD;
	END_VAR	

Program		xResult := Unite8Bit_DWORD(In := abyIn[2], Num := usiNum, Out => dwOut);

	LD	ST
Operation result		

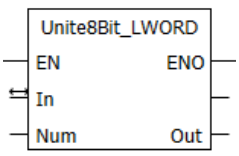
Name	Data type	Address	Value
xResult	BOOL		TRUE
abyIn	ARRAY [1..10] OF BYTE		
abyIn[1]	BYTE		1
abyIn[2]	BYTE		1
abyIn[3]	BYTE		1
abyIn[4]	BYTE		1
abyIn[5]	BYTE		1
abyIn[6]	BYTE		1
abyIn[7]	BYTE		1
abyIn[8]	BYTE		1
abyIn[9]	BYTE		1
abyIn[10]	BYTE		1
usiNum	USINT		1
dwOut	DWORD		16#00000001

- **Precautions**
  - When the value of "Num" is 0, "Out" is initialized to 0, and the function returns TRUE.
  - When the value of "Num" exceeds the valid range, the function returns FALSE, and "Out" is initialized to 0.
  - When the value of "Num" exceeds the array area of In[ ], the function returns FALSE, and "Out" is initialized to 0.
  - If the array exceeds 3 dimensions, the function returns FALSE, and "Out" is initialized to 0.

18.1.2.55 Unite8Bit_LWORD

Joins bytes of data into a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
Unite8Bit_LWORD	Byte data join group	FC		:= Unite8Bit_LWORD(In := , Num := , Out =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input/Output	In[ ] array	Array of objects to join	Follows data type	-	Array to join

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Num	Number of bytes to join	Follows data type	1	Number of bytes to join
Output	Out	Join result	Follows data type	0	A bit string indicating the join result

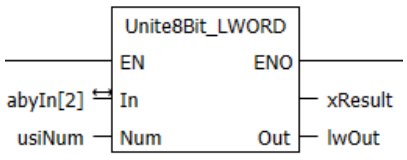
Variable type

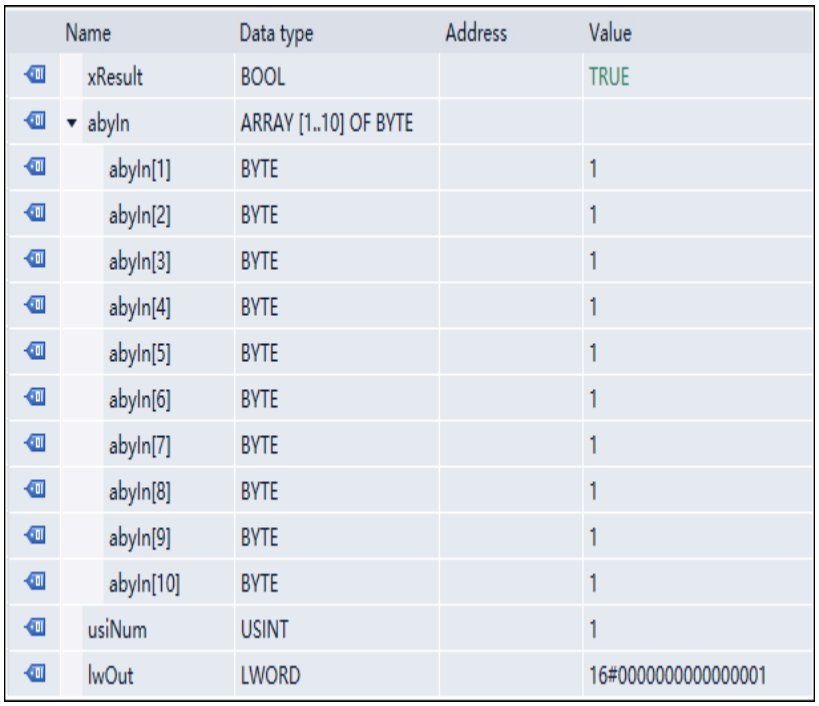
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - Joins the elements in the array In[ ] to generate the bit string "Out" as the join result. The number of array elements to be joined is specified by "Num".
  - Elements from In[0] to In["Num"-1] are joined to generate a bit string of "Num" bytes. If "Num" is less than the length of "Out", zeros are supplemented to the high bits of "Out".
- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	abyIn: ARRAY[1..10] OF BYTE := [10(1)];
	usiNum: USINT := 1;
	lwOut: LWORD;
	END_VAR	

Program		xResult := Unite8Bit_LWORD(In := abyIn[2], Num := usiNum, Out => lwOut);

	LD	ST
Operation result		

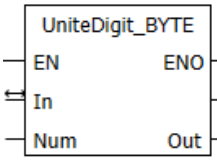
Name	Data type	Address	Value
xResult	BOOL		TRUE
abyIn	ARRAY [1..10] OF BYTE		
abyIn[1]	BYTE		1
abyIn[2]	BYTE		1
abyIn[3]	BYTE		1
abyIn[4]	BYTE		1
abyIn[5]	BYTE		1
abyIn[6]	BYTE		1
abyIn[7]	BYTE		1
abyIn[8]	BYTE		1
abyIn[9]	BYTE		1
abyIn[10]	BYTE		1
usiNum	USINT		1
lwOut	LWORD		16#0000000000000001

- **Precautions**
  - When the value of "Num" is 0, "Out" is initialized to 0, and the function returns TRUE.
  - When the value of "Num" exceeds the valid range, the function returns FALSE, and "Out" is initialized to 0.
  - When the value of "Num" exceeds the array area of In[ ], the function returns FALSE, and "Out" is initialized to 0.
  - If the array exceeds 3 dimensions, the function returns FALSE, and "Out" is initialized to 0.

18.1.2.56 UniteDigit_BYTE

Joins digits (4-bit unit) of data into a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
UniteDigit_BYTE	4-bit join group	FC		:= UniteDigit_BYTE(In := , Num := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input/Output	In[ ] array	Array of objects to join	Follows data type	-	Array to join

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Num	Number of digits to join	Follows data type	1	Number of digits to join
Output	Out	Join result	Follows data type	0	A bit string indicating the join result

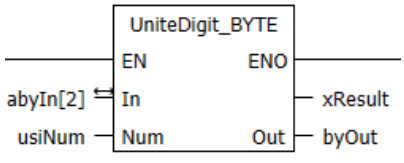
Variable type

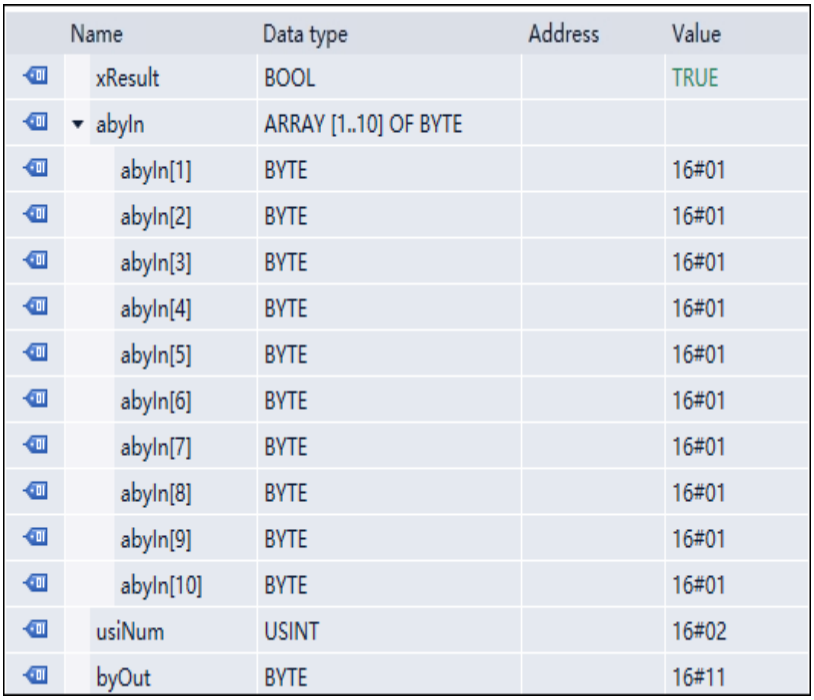
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - Joins the lower 4 bits (1 digit = 4 bits) of each element in the In[ ] array to generate the bit string "Out" as the join result. The number of array elements to be joined is specified by "Num".
  - The lower 4 bits of In[0] to In["Num" - 1] are joined to generate a bit string of "Num" digits. If "Num" is less than the length of "Out", 16#0 is supplemented to the high bits of "Out".
- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	abyIn: ARRAY[1..10] OF BYTE := [10(1)];
	usiNum: USINT := 2;
	byOut: BYTE;
	END_VAR	

Program		xResult := UniteDigit_BYTE(In := abyIn[2], Num := usiNum, Out => byOut);

	LD	ST
Operation result		

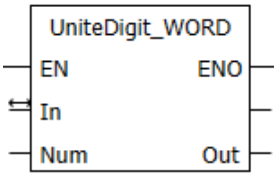
Name	Data type	Address	Value
xResult	BOOL		TRUE
abyIn	ARRAY [1..10] OF BYTE		
abyIn[1]	BYTE		16#01
abyIn[2]	BYTE		16#01
abyIn[3]	BYTE		16#01
abyIn[4]	BYTE		16#01
abyIn[5]	BYTE		16#01
abyIn[6]	BYTE		16#01
abyIn[7]	BYTE		16#01
abyIn[8]	BYTE		16#01
abyIn[9]	BYTE		16#01
abyIn[10]	BYTE		16#01
usiNum	USINT		16#02
byOut	BYTE		16#11

- **Precautions**
  - When the value of "Num" is 0, "Out" is 0, and the function returns TRUE.
  - When the value of "Num" exceeds the valid range, the function returns FALSE, and "Out" is 0.
  - When the value of "Num" exceeds the array area of In[ ], the function returns FALSE, and "Out" is 0.
  - If the array exceeds 3 dimensions, the function returns FALSE, and "Out" is initialized to 0.

18.1.2.57 UniteDigit_WORD

Joins digits (4-bit unit) of data into a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
UniteDigit_WORD	4-bit join group	FC		:= UniteDigit_WORD(In := , Num := , Out =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input/Output	In[ ] array	Array of objects to join	Follows data type	-	Array to join
Input	Num	Number of digits to join	Follows data type	1	Number of digits to join

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Join result	Follows data type	0	A bit string indicating the join result

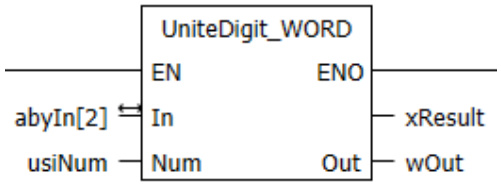
Variable type

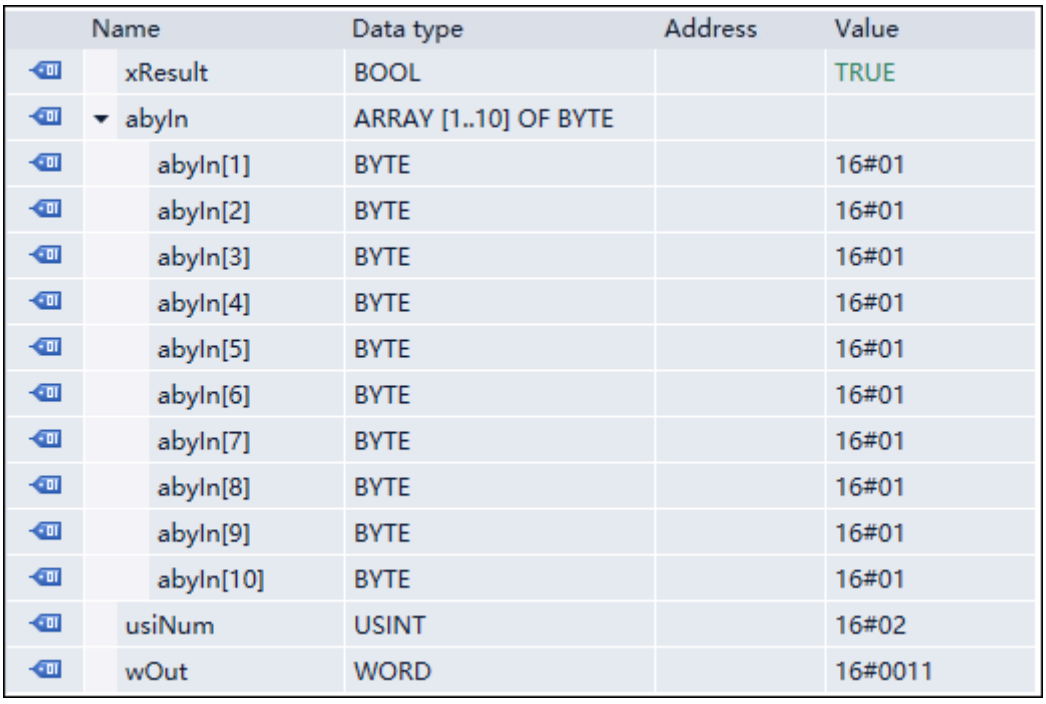
	Bool ean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - Joins the lower 4 bits (1 digit = 4 bits) of each element in the In[ ] array to generate the bit string "Out" as the join result. The number of array elements to be joined is specified by "Num".
  - The lower 4 bits of In[0] to In["Num" - 1] are joined to generate a bit string of "Num" digits. If "Num" is less than the length of "Out", 16#0 is supplemented to the high bits of "Out".
- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	abyIn: ARRAY[1..10] OF BYTE := [10(1)];
	usiNum: USINT := 2;
	wOut: WORD;
	END_VAR	

Program		xResult := UniteDigit_WORD(In := abyIn[2], Num := usiNum, Out => wOut);

	LD	ST
Operation result		

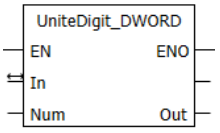
Name	Data type	Address	Value
xResult	BOOL		TRUE
abyIn	ARRAY [1..10] OF BYTE		
abyIn[1]	BYTE		16#01
abyIn[2]	BYTE		16#01
abyIn[3]	BYTE		16#01
abyIn[4]	BYTE		16#01
abyIn[5]	BYTE		16#01
abyIn[6]	BYTE		16#01
abyIn[7]	BYTE		16#01
abyIn[8]	BYTE		16#01
abyIn[9]	BYTE		16#01
abyIn[10]	BYTE		16#01
usiNum	USINT		16#02
wOut	WORD		16#0011

- **Precautions**
  - When the value of "Num" is 0, "Out" is 0, and the function returns TRUE.
  - When the value of "Num" exceeds the valid range, the function returns FALSE, and "Out" is 0.
  - When the value of "Num" exceeds the array area of In[ ], the function returns FALSE, and "Out" is 0.
  - If the array exceeds 3 dimensions, the function returns FALSE, and "Out" is initialized to 0.

18.1.2.58 UniteDigit_DWORD

Joins digits (4-bit unit) of data into a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
UniteDigit_DWORD	4-bit join group	FC		:= UniteDigit_DWORD(In := , Num := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input/Output	In[ ] array	Array of objects to join	Follows data type	-	Array to join
Input	Num	Number of digits to join	Follows data type	1	Number of digits to join

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Join result	Follows data type	0	A bit string indicating the join result

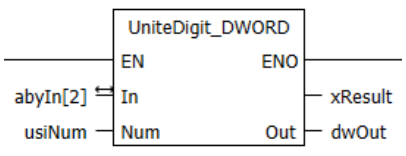
Variable type

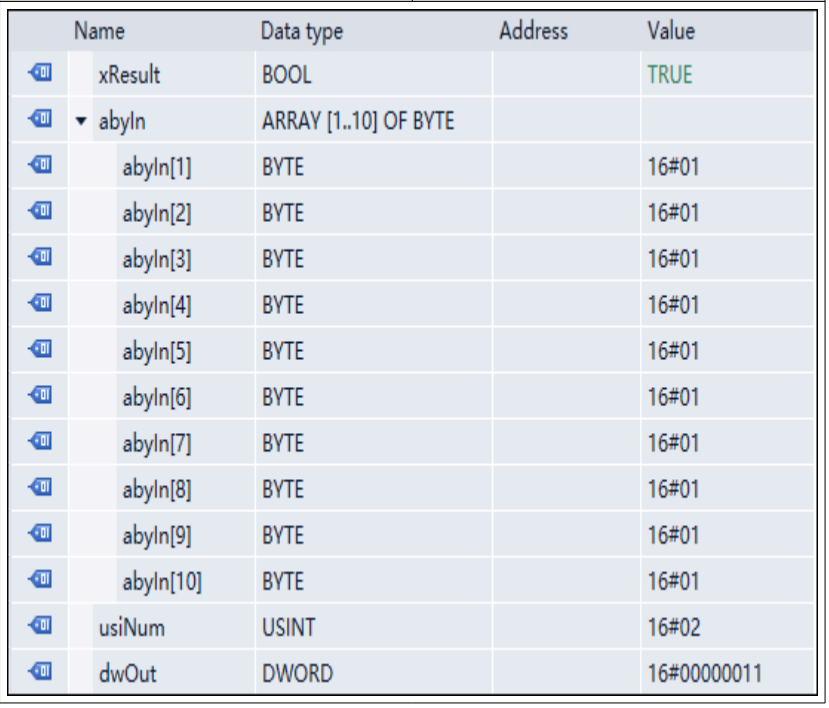
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - Joins the lower 4 bits (1 digit = 4 bits) of each element in the In[ ] array to generate the bit string "Out" as the join result. The number of array elements to be joined is specified by "Num".
  - The lower 4 bits of In[0] to In["Num" - 1] are joined to generate a bit string of "Num" digits. If "Num" is less than the length of "Out", 16#0 is supplemented to the high bits of "Out".
- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	abyIn: ARRAY[1..10] OF BYTE := [10(1)];
	usiNum: USINT := 2;
	dwOut: DWORD;
	END_VAR	

Program		xResult := UniteDigit_DWORD(In := abyIn[2], Num := usiNum, Out => dwOut);

	LD	ST
Operation result		

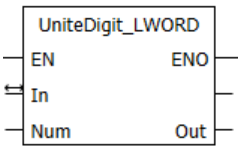
Name	Data type	Address	Value
xResult	BOOL		TRUE
abyIn	ARRAY [1..10] OF BYTE		
abyIn[1]	BYTE		16#01
abyIn[2]	BYTE		16#01
abyIn[3]	BYTE		16#01
abyIn[4]	BYTE		16#01
abyIn[5]	BYTE		16#01
abyIn[6]	BYTE		16#01
abyIn[7]	BYTE		16#01
abyIn[8]	BYTE		16#01
abyIn[9]	BYTE		16#01
abyIn[10]	BYTE		16#01
usiNum	USINT		16#02
dwOut	DWORD		16#00000011

- **Precautions**
  - When the value of "Num" is 0, "Out" is 0, and the function returns TRUE.
  - When the value of "Num" exceeds the valid range, the function returns FALSE, and "Out" is 0.
  - When the value of "Num" exceeds the array area of In[ ], the function returns FALSE, and "Out" is 0.
  - If the array exceeds 3 dimensions, the function returns FALSE, and "Out" is initialized to 0.

18.1.2.59 UniteDigit_LWORD

Joins digits (4-bit unit) of data into a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
UniteDigit_LWORD	4-bit join group	FC		:= UniteDigit_LWORD(In := , Num := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input/Output	In[ ] array	Array of objects to join	Follows data type	-	Array to join
Input	Num	Number of digits to join	Follows data type	1	Number of digits to join

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Join result	Follows data type	0	A bit string indicating the join result

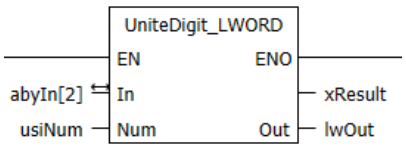
Variable type

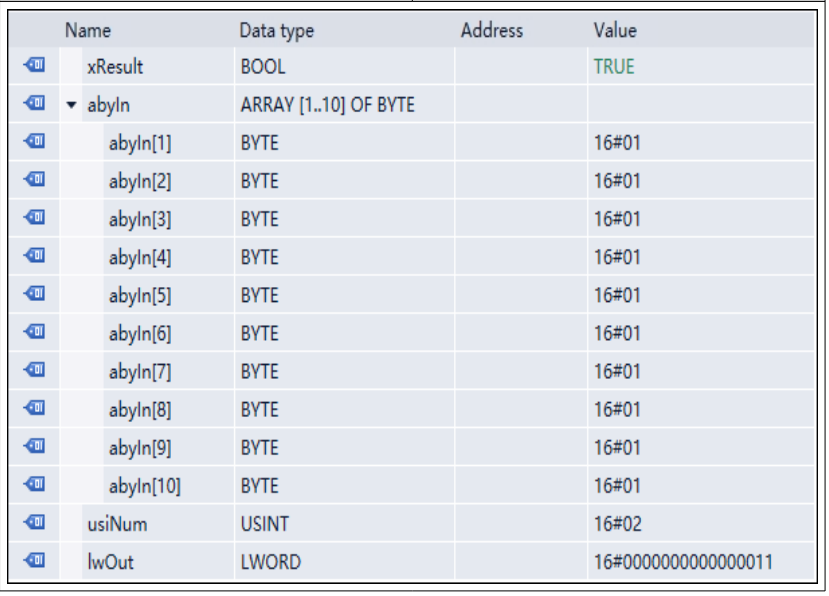
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[] array	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - Joins the lower 4 bits (1 digit = 4 bits) of each element in the In[ ] array to generate the bit string "Out" as the join result. The number of array elements to be joined is specified by "Num".
  - The lower 4 bits of In[0] to In["Num" - 1] are joined to generate a bit string of "Num" digits. If "Num" is less than the length of "Out", 16#0 is supplemented to the high bits of "Out".
- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	abyIn: ARRAY[1..10] OF BYTE := [10(1)];
	usiNum: USINT := 2;
	lwOut: LWORD;
	END_VAR	

Program		xResult := UniteDigit_LWORD(In := abyIn[2], Num := usiNum, Out => lwOut);

	LD	ST
Operation result		

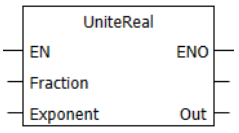
Name	Data type	Address	Value
xResult	BOOL		TRUE
abyIn	ARRAY [1..10] OF BYTE		
abyIn[1]	BYTE		16#01
abyIn[2]	BYTE		16#01
abyIn[3]	BYTE		16#01
abyIn[4]	BYTE		16#01
abyIn[5]	BYTE		16#01
abyIn[6]	BYTE		16#01
abyIn[7]	BYTE		16#01
abyIn[8]	BYTE		16#01
abyIn[9]	BYTE		16#01
abyIn[10]	BYTE		16#01
usiNum	USINT		16#02
lwOut	LWORD		16#0000000000000011

- **Precautions**
  - When the value of "Num" is 0, "Out" is 0, and the function returns TRUE.
  - When the value of "Num" exceeds the valid range, the function returns FALSE, and "Out" is 0.
  - When the value of "Num" exceeds the array area of In[ ], the function returns FALSE, and "Out" is 0.
  - If the array exceeds 3 dimensions, the function returns FALSE, and "Out" is initialized to 0.

18.1.2.60 UniteReal

Combines a signed mantissa and exponent to generate a real number.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
UniteReal	Combine mantissa and exponent	FC		:= UniteReal(Fraction := , Exponent := , Out =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Fraction	Signed mantissa part	Follows data type	-	Signed mantissa part
	Exponent	Exponent part	Follows data type	0	Exponent part

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Real Number	Follows data type	0	A real number indicating the combination result

Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Fraction	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–	–	–
Exponent	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–

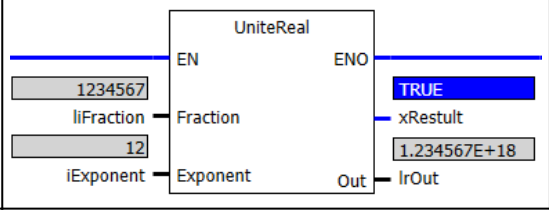
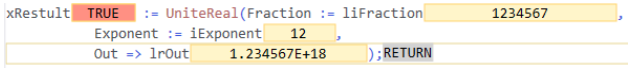
- **Function**

  Combines the signed mantissa "Fraction" and the exponent "Exponent" to obtain the real number "Out".

- **Program example**

	LD	ST

Define variables	VAR
	xRestult: BOOL;
	liFraction: LINT := 1234567;
	iExponent: INT := 12;
	lrOut: LREAL;
	END_VAR	

Program		

	LD	ST
Operation result		

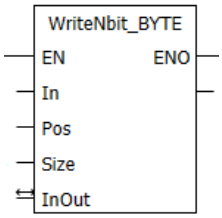
Name	Data type	Address	Value
xRestult	BOOL		TRUE
liFraction	LINT		1234567
iExponent	INT		12
lrOut	LREAL		1.234567E+18

- **Precautions**
  - Depending on the values of "Fraction" and "Exponent", converting integers to real numbers may introduce errors.
  - When the combined result exceeds the valid range of "Out", if "Exponent" is positive, the value of "Out" becomes infinity with the same sign as "Fraction". If "Exponent" is negative, the value of "Out" becomes 0.

18.1.2.61 WriteNbit_BYTE

Writes multiple bits into a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
WriteNbit_BYTE	N-bit write group	FC		:= WriteNbit_BYTE(In := , Pos := , Size := , InOut :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Read source	Follows data type	0	A bit string to be read and written to "InOut"
	Pos	Write position	Follows data type	0	Bit position to write to
	Size	Write size	Follows data type	0	The number of bits to be written
Input/Output	InOut	Write target	Follows data type	-	Write result

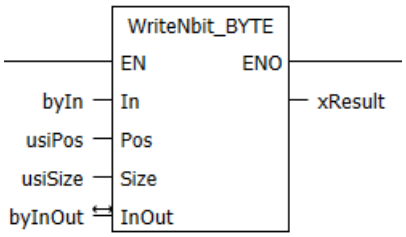
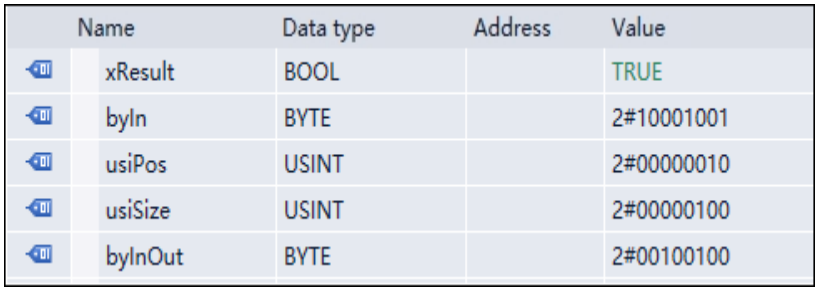
Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Pos	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
In Out	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

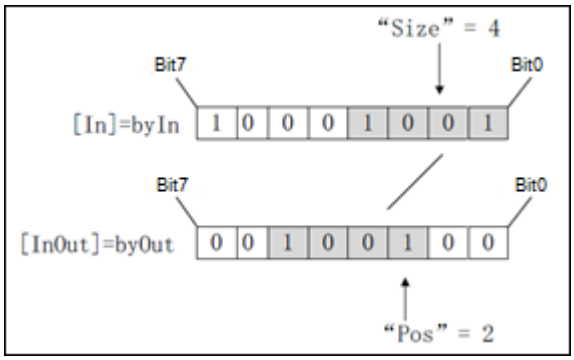
- **Function**
  - The function first reads the lower "Size" bits from the source "In".
  - Then, it writes the read value to the write position "Pos" of the target "InOut".
- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	byIn: BYTE := 16#89;
	usiPos: USINT := 2;
	usiSize: USINT := 4;
	byInOut: BYTE;
	END_VAR	

Program		xResult := WriteNbit_BYTE(In := byIn, Pos := usiPos, Size := usiSize, InOut := byInOut);
Operation result		

Name	Data type	Address	Value
xResult	BOOL		TRUE
byIn	BYTE		2#10001001
usiPos	USINT		2#00000010
usiSize	USINT		2#00000100
byInOut	BYTE		2#00100100

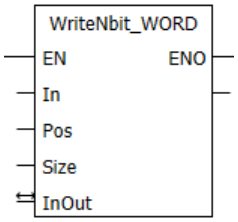
	LD	ST
Principle analysis		

- **Precautions**
  - A compilation exception occurs when the data types of "In" and "InOut" differ.
  - When the value of "Size" is 0, the value of "InOut" remains unchanged.
  - When "Size" or "Pos" exceeds the valid range, an exception occurs, the function returns FALSE, and the value of "InOut" remains unchanged.
  - When the bit string "InOut" does not contain enough bits for the number of bits specified by "Size" from the position specified by "Pos", an exception occurs, the function returns FALSE, and the value of "InOut" remains unchanged.

18.1.2.62 WriteNbit_WORD

Writes multiple bits into a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
WriteNbit_WORD	N-bit write group	FC		:= WriteNbit_WORD(In := , Pos := , Size := , InOut :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Read source	Follows data type	0	A bit string to be read and written to "InOut"
	Pos	Write position	Follows data type	0	Bit position to write to
	Size	Write size	Follows data type	0	The number of bits to be written

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input/Output	InOut	Write target	Follows data type	-	Write result

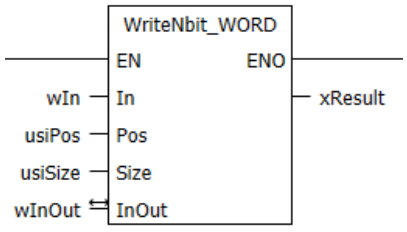
Variable type

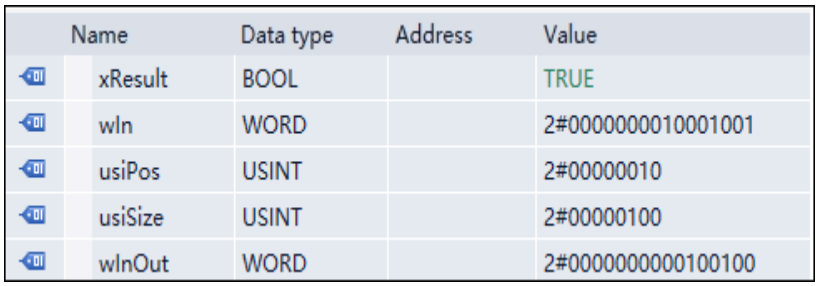
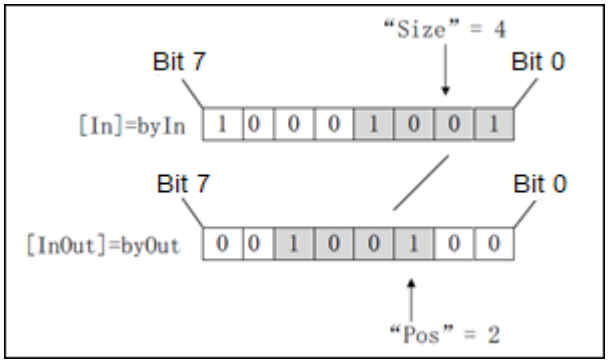
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Pos	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
InOut	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - The function first reads the lower "Size" bits from the source "In".
  - Then, it writes the read value to the write position "Pos" of the target "InOut".
- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	wIn: WORD := 16#89;
	usiPos: USINT := 2;
	usiSize: USINT := 4;
	wInOut: WORD;
	END_VAR	

Program		xResult := WriteNbit_WORD(In := wIn, Pos := usiPos, Size := usiSize, InOut := wInOut);

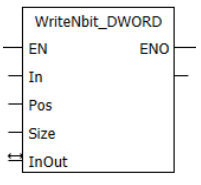
	LD	ST
Operation result		
Principle analysis		

- **Precautions**
  - A compilation exception occurs when the data types of "In" and "InOut" differ.
  - When the value of "Size" is 0, the value of "InOut" remains unchanged.
  - When "Size" or "Pos" exceeds the valid range, an exception occurs, the function returns FALSE, and the value of "InOut" remains unchanged.
  - When the bit string "InOut" does not contain enough bits for the number of bits specified by "Size" from the position specified by "Pos", an exception occurs, the function returns FALSE, and the value of "InOut" remains unchanged.

18.1.2.63 WriteNbit_DWORD

Writes multiple bits into a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
WriteNbit_DWORD	N-bit write group	FC		:= WriteNbit_DWORD(In := , Pos := , Size := , InOut :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Read source	Follows data type	0	A bit string to be read and written to "InOut"
	Pos	Write position	Follows data type	0	Bit position to write to
	Size	Write size	Follows data type	0	The number of bits to be written
Input/Output	InOut	Write target	Follows data type	-	Write result

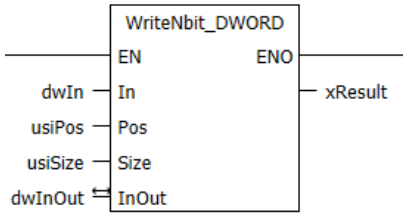
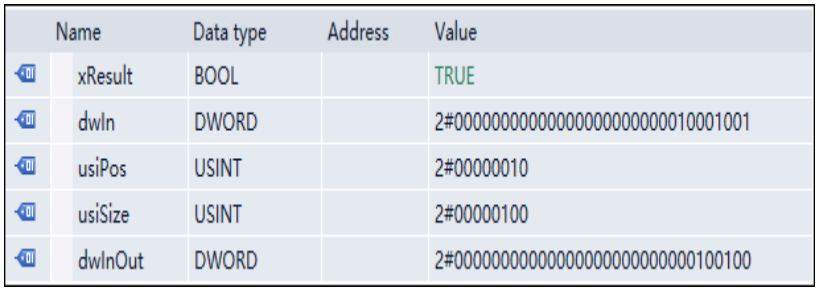
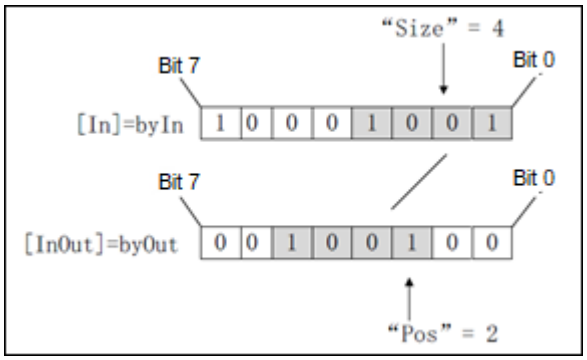
Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Pos	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
InOut	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - The function first reads the lower "Size" bits from the source "In".
  - Then, it writes the read value to the write position "Pos" of the target "InOut".
- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	dwIn: DWORD := 16#89;
	usiPos: USINT := 2;
	usiSize: USINT := 4;
	dwInOut: DWORD;
	END_VAR	

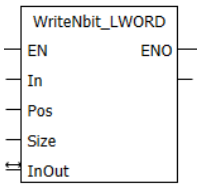
	LD	ST
Program		`xResult := WriteNbit_DWORD(In := dwIn, Pos := usiPos, Size := usiSize, InOut := dwInOut);`
Operation result		
Principle analysis		

- **Precautions**
  - A compilation exception occurs when the data types of "In" and "InOut" differ.
  - When the value of "Size" is 0, the value of "InOut" remains unchanged.
  - When "Size" or "Pos" exceeds the valid range, an exception occurs, the function returns FALSE, and the value of "InOut" remains unchanged.
  - When the bit string "InOut" does not contain enough bits for the number of bits specified by "Size" from the position specified by "Pos", an exception occurs, the function returns FALSE, and the value of "InOut" remains unchanged.

18.1.2.64 WriteNbit_LWORD

Writes multiple bits into a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
WriteNbit_LWORD	N-bit write group	FC		:= WriteNbit_LWORD(In := , Pos := , Size := , InOut :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Read source	Follows data type	-	A bit string to be read and written to "InOut"
	Pos	Write position	Follows data type	0	Bit position to write to
	Size	Write size	Follows data type	0	The number of bits to be written
Input/Output	InOut	Write target	Follows data type	-	Write result

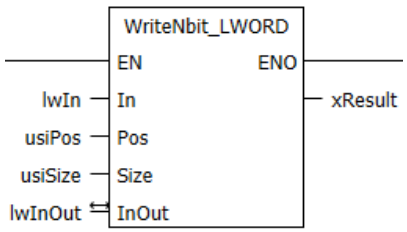
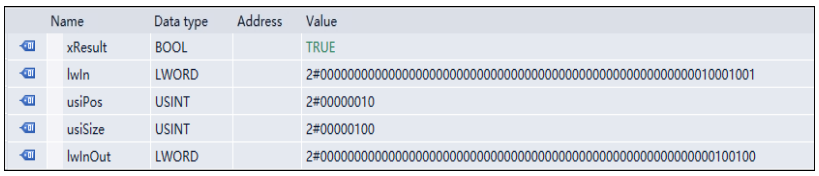
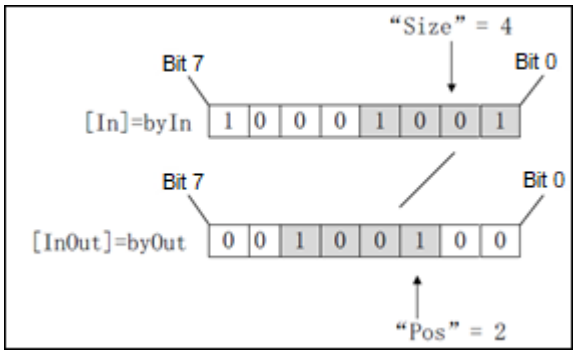
Variable type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Pos	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
InOut	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - The function first reads the lower "Size" bits from the source "In".
  - Then, it writes the read value to the write position "Pos" of the target "InOut".
- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	lwIn: LWORD := 16#89;
	usiPos: USINT := 2;
	usiSize: USINT := 4;
	lwInOut: LWORD;
	END_VAR	

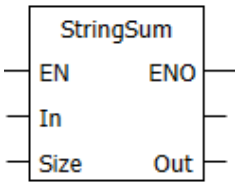
	LD	ST
Program		`xResult := WriteNbit_LWORD(In := lwIn, Pos := usiPos, Size := usiSize, InOut := lwInOut);`
Operation result		
Principle analysis		

- **Precautions**
  - A compilation exception occurs when the data types of "In" and "InOut" differ.
  - When the value of "Size" is 0, the value of "InOut" remains unchanged.
  - When "Size" or "Pos" exceeds the valid range, an exception occurs, the function returns FALSE, and the value of "InOut" remains unchanged.
  - When the bit string "InOut" does not contain enough bits for the number of bits specified by "Size" from the position specified by "Pos", an exception occurs, the function returns FALSE, and the value of "InOut" remains unchanged.

18.1.2.65 StringSum

Calculates the sum value (the sum of character codes of all characters) of a string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StringSum	String sum calculation	FC		:= StringSum(In := , Size := , Out =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Input string	Follows data type	-	Input string
	Size	Byte size	1 or 2	1	Byte size of the sum value
Output	Out	Sum value	Number of bytes specified by "Size"	-	Sum value

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
Size	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

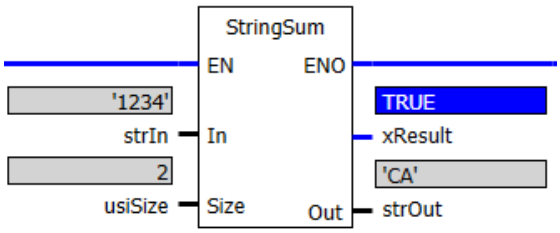
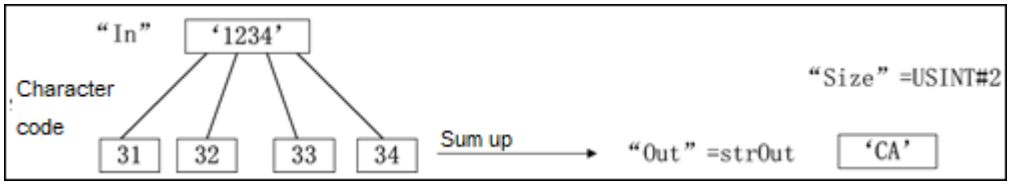
- **Function description**

  Calculates the sum value (the sum of character codes of all characters) of the input string "In". The sum value "Out" indicates the number of bytes specified by "Size". "Out" is represented as a hexadecimal string, with a NULL character stored at the end.

- **Program example**

  In the following example, the input parameter "In" is "1234" and "Size" is 2.

Item	LD	ST
Define variables	VAR strIn: STRING[256] := '1234'; usiSize: USINT := 2; xResult: BOOL; strOut: STRING[2]; END_VAR	

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>
</td><td>xResult TRUE := StringSum(In := strIn '1234',
Size := usiSize 2 ,
Out => strOut 'CA');RETURN</td></tr>
<tr><td>Operation result</td><td colspan="2"><table>
<tr><th></th><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td></td><td>strIn</td><td>STRING[256]</td><td></td><td>'1234'</td></tr>
<tr><td></td><td>usiSize</td><td>USINT</td><td></td><td>2</td></tr>
<tr><td></td><td>xResult</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td></td><td>strOut</td><td>STRING[2]</td><td></td><td>'CA'</td></tr>
</table></td></tr>
<tr><td>Principle analysis</td><td colspan="2">
</td></tr>
</table>

- **Precautions**
  - When the sum of the character codes in "In" exceeds the number of bits specified by "Size", the higher bits are discarded, and the function returns FALSE.
  - When the value of "Size" is out of the valid range, an exception occurs in the program, and the function returns FALSE.
  - When the byte count of "In" is 0 (NULL), an exception occurs in the program, and the function returns FALSE.
  - When "In" is not NULL-terminated, the result may become abnormal.

18.1.3 Comparison Operations

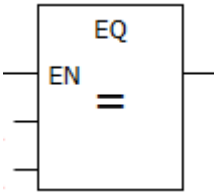
18.1.3.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.1.3.2 EQ	FC	Equal instruction	-	-
18.1.3.3 GE	FC	Greater than or equal instruction		
18.1.3.4 GT	FC	Greater than instruction		
18.1.3.5 LE	FC	Less than or equal instruction		
18.1.3.6 LT	FC	Less than instruction		
18.1.3.7 NE	FC	Not equal instruction		

18.1.3.2 EQ

EQ determines whether the two input values are equal. If the first input value equals the second, the output is TRUE; otherwise, the output is FALSE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
EQ	Equal instruction	FC		=

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Input 1	Follows data type	-	Data 1
	In2	Input 2	Follows data type	-	Data 2
Output	Out	Return value	Follows data type	FALSE	Comparison result

Data type

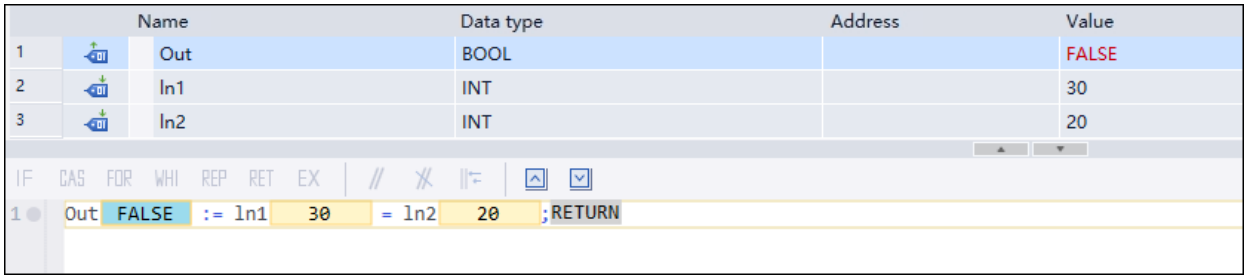
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

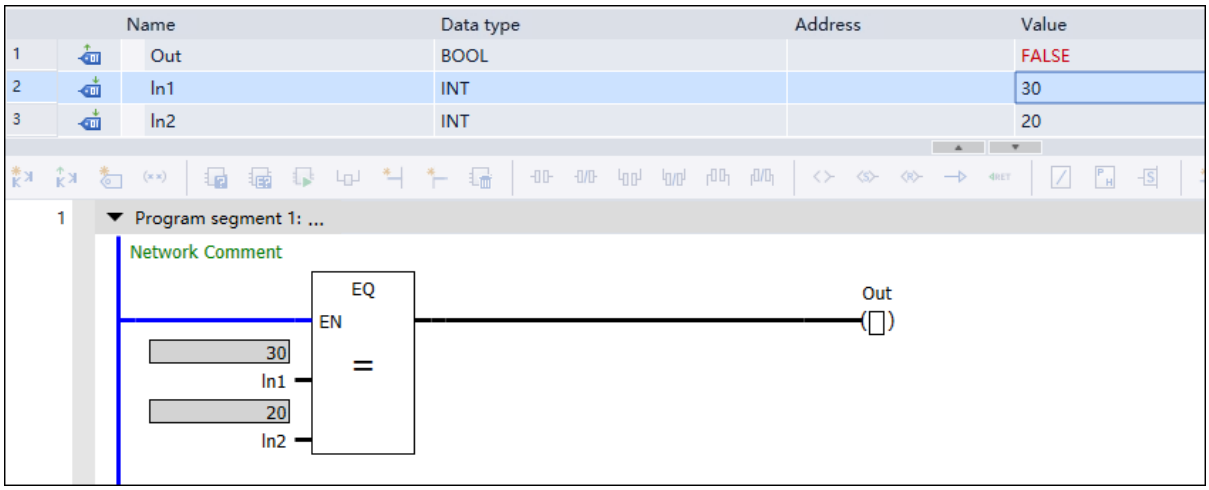
Outputs TRUE when In1 equals In2, otherwise outputs FALSE.

- **Program example**

ST:



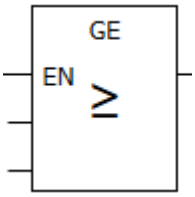
LD:



18.1.3.3 GE

GE compares the two input values. If the first input value is greater than or equal to the second, the output is TRUE; otherwise, the output is FALSE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GE	Greater than or equal instruction	FC		>=

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Input 1	Follows data type	-	Data 1
	In2	Input 2	Follows data type	-	Data 2
Output	Out	Return value	Follows data type	FALSE	Comparison result

Data type

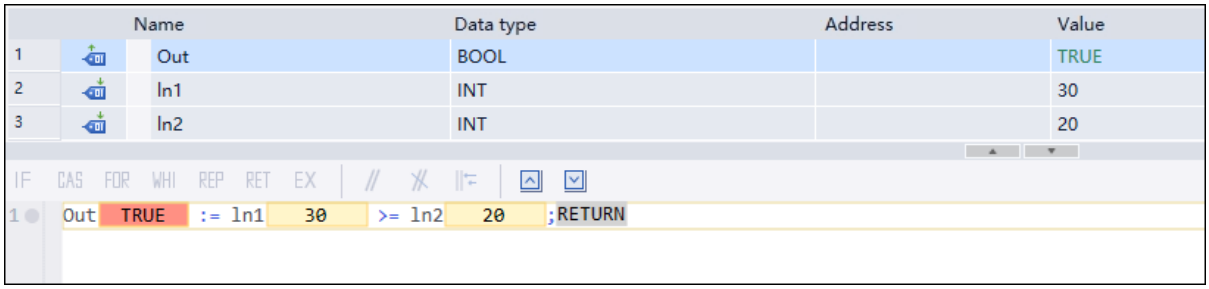
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

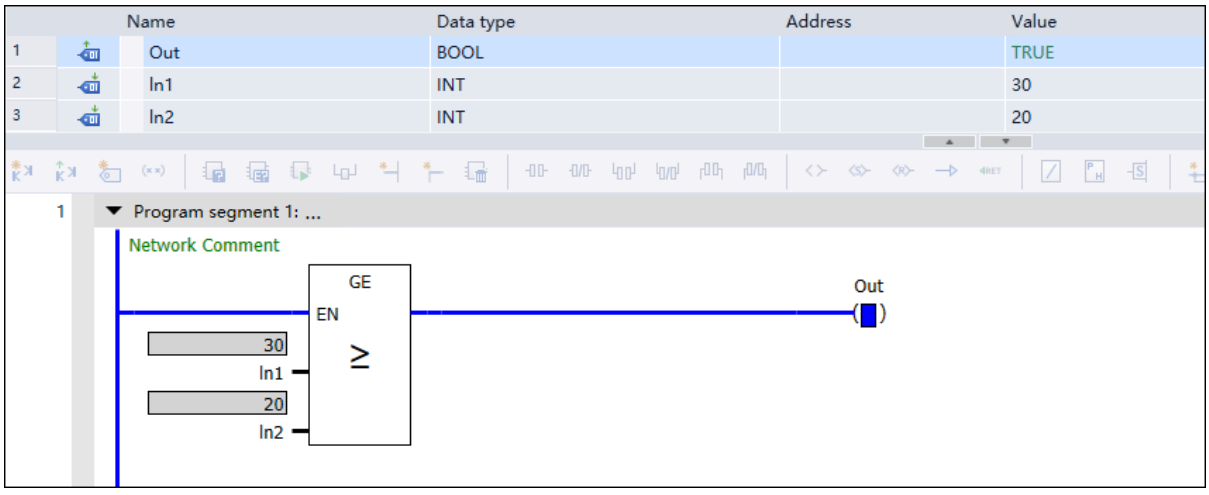
  Outputs TRUE when In1 is greater than or equal to In2, otherwise outputs FALSE.

- **Program example**

  ST:



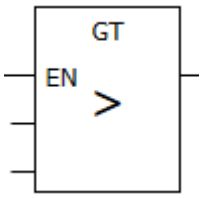
  LD:



18.1.3.4 GT

GE compares the two input values. If the first input value is greater than the second, the output is TRUE; otherwise, the output is FALSE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GT	Greater than instruction	FC		>

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Input 1	Follows data type	-	Data 1
	In2	Input 2	Follows data type	-	Data 2
Output	Out	Return value	Follows data type	FALSE	Comparison result

  Data type

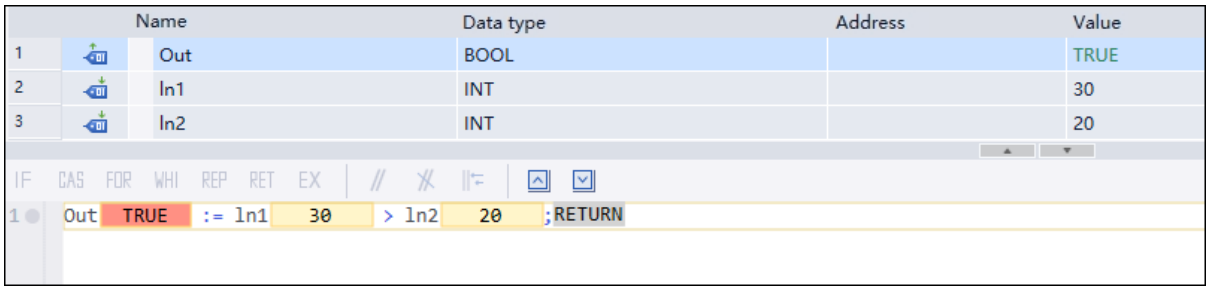
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Outputs TRUE when In1 is greater than In2, otherwise outputs FALSE.

- **Program example**

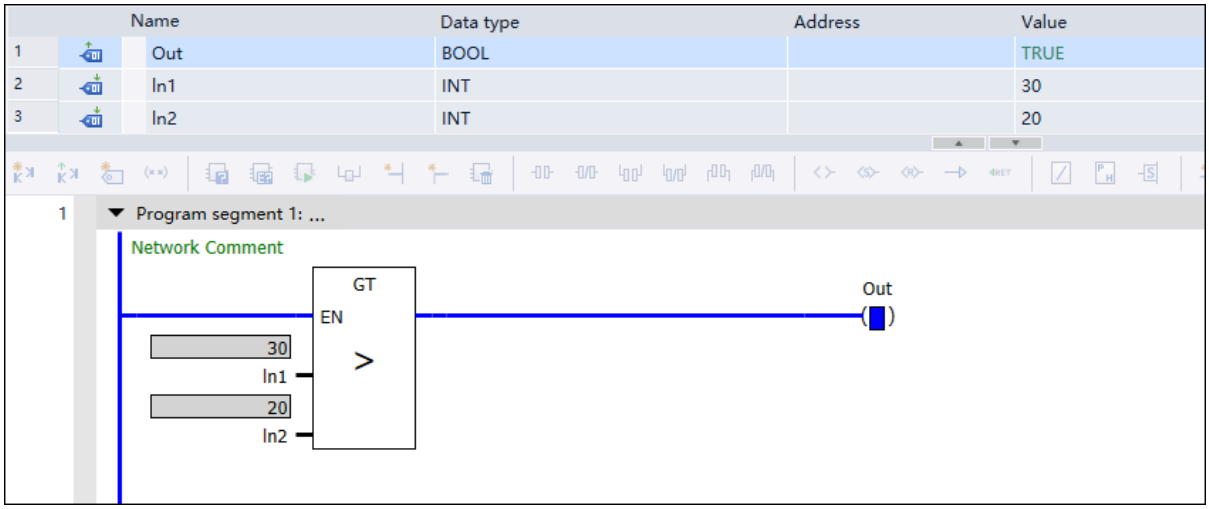
  ST:



	Name	Data type	Address	Value
1	Out	BOOL		TRUE
2	In1	INT		30
3	In2	INT		20

```
IF  CAS  FOR  WHI  REP  RET  EX
1   Out TRUE := In1 30 > In2 20 ;RETURN
```

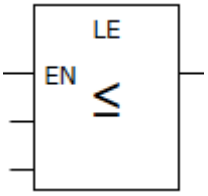
  LD:



18.1.3.5 LE

GE compares the two input values. If the first input value is smaller than or equal to the second, the output is TRUE; otherwise, the output is FALSE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LE	Less than or equal instruction	FC		<=

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Input 1	Follows data type	-	Data 1
	In2	Input 2	Follows data type	-	Data 2
Output	Out	Return value	Follows data type	FALSE	Comparison result

Data type

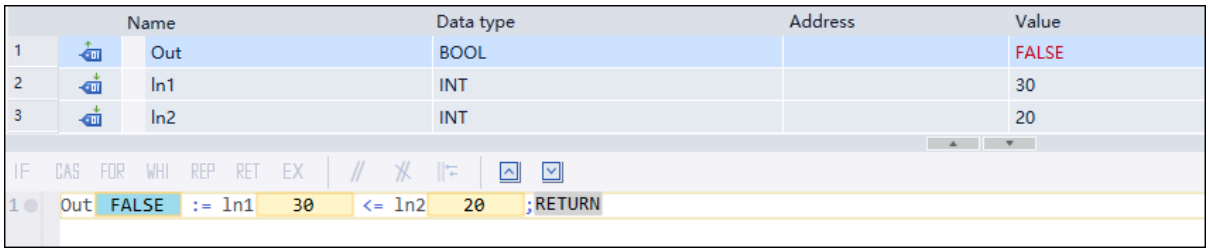
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

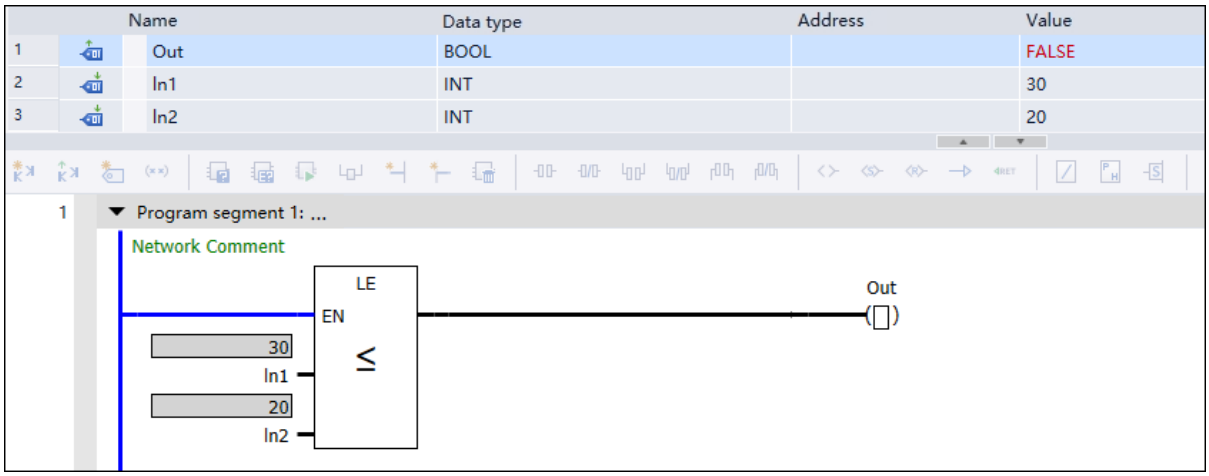
  Outputs TRUE when In1 is less than or equal to In2, otherwise outputs FALSE.

- **Program example**

  ST:



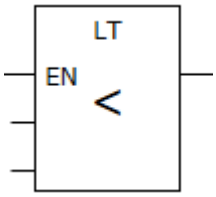
  LD:



18.1.3.6 LT

GE compares the two input values. If the first input value is smaller than the second, the output is TRUE; otherwise, the output is FALSE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LT	Less than instruction	FC		<

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Input 1	Follows data type	-	Data 1
	In2	Input 2	Follows data type	-	Data 2
Output	Out	Return value	Follows data type	FALSE	Comparison result

  Data type

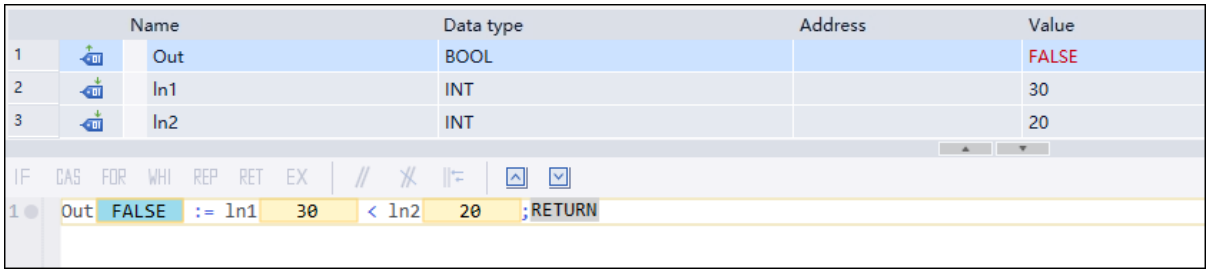
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

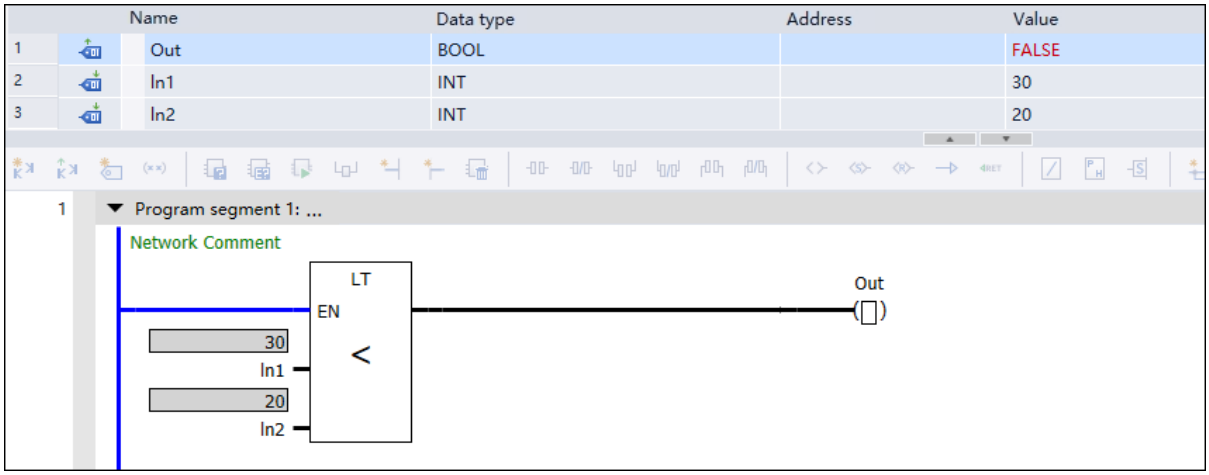
  Outputs TRUE when In1 is less than In2, otherwise outputs FALSE.

- **Program example**

  ST:



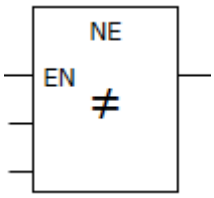
  LD:



18.1.3.7 NE

GE compares the two input values. If the first input value does not equal the second, the output is TRUE; otherwise, the output is FALSE.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
NE	Not equal instruction	FC		<>

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Input 1	Follows data type	-	Data 1
	In2	Input 2	Follows data type	-	Data 2
Output	Out	Return value	Follows data type	FALSE	Comparison result

Data type

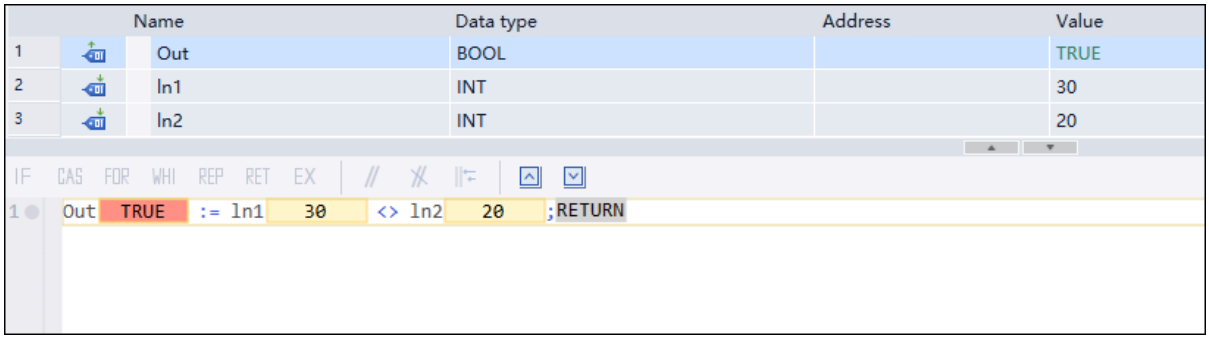
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

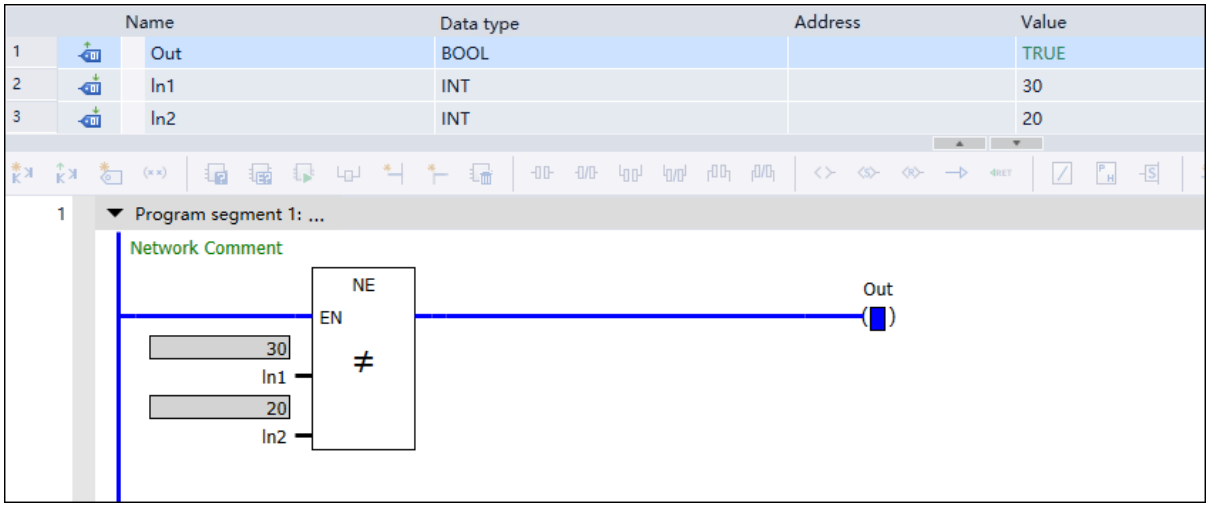
  Outputs TRUE when In1 does not equal In2, otherwise outputs FALSE.

- **Program example**

  ST:



  LD:



18.1.4 Logic Operations

18.1.4.1 Instruction List

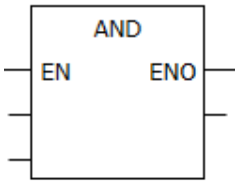
Instruction	FB/FC	Function	Library	Applicable Library Version
18.1.4.2 AND	FC	AND instruction	-	-

Instruction	FB/FC	Function	Library	Applicable Library Version
18.1.4.3 NOT	FC	NOT instruction		
18.1.4.4 OR	FC	OR instruction		
18.1.4.5 XOR	FC	XOR instruction		
18.1.4.6 TestABit	FC	Outputs the value of the specified bit	OMUtils	Library version 1.3.0.0 and later
18.1.4.7 TestABitN	FC	Outputs the inverted value of the specified bit		Library version 1.3.0.0 and later
18.1.4.8 SetBits	FC	Sets multiple bits consecutively		Library version 1.3.0.0 and later
18.1.4.9 ResetBits	FC	Resets multiple bits consecutively		Library version 1.3.0.0 and later
18.1.4.10 RESET_BF	FC	Batch reset		Applicable to library version 1.5.0.0 and later
18.1.4.11 SET_BF	FC	Batch setting		Applicable to library version 1.5.0.0 and later

18.1.4.2 AND

Bitwise AND: When the same bit in all inputs is 1, the corresponding output bit is 1; if not, the output bit is 0.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AND	AND instruction	FC		AND ;

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Input 1	Follows data type	-	Data 1
	In2	Input 2	Follows data type	-	Data 2
Output	Out	Return value	Follows data type	-	Comparison result

Data type

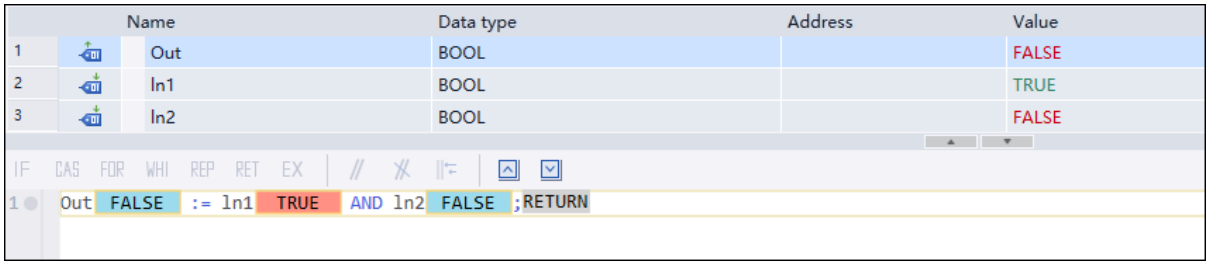
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–
Out	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–

- **Function**

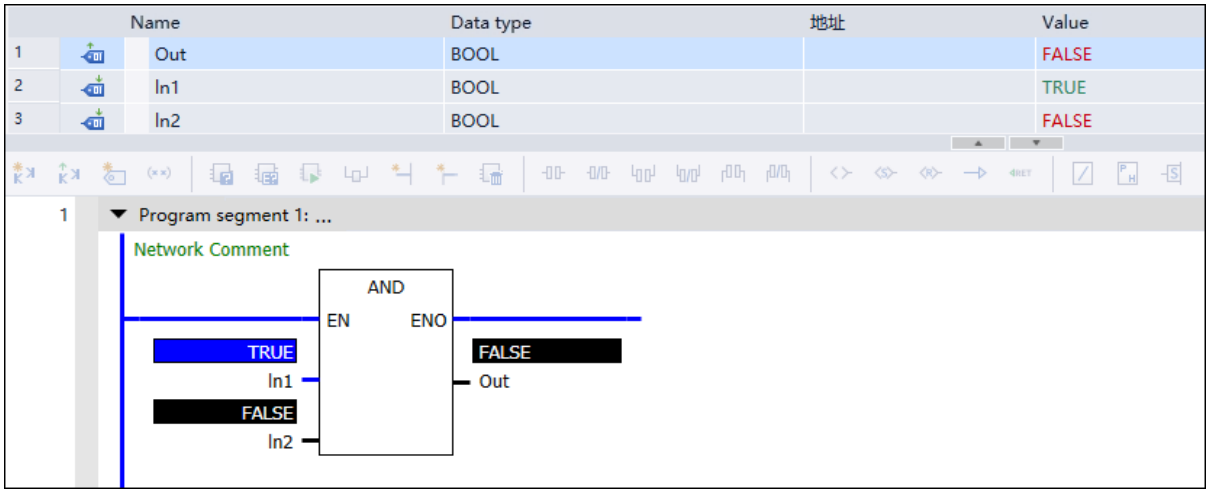
  When both inputs are 1, the output is also 1; otherwise, the output is 0.

- **Program example**

  ST:



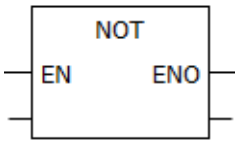
  LD:



18.1.4.3 NOT

Bitwise NOT: when the input is 0, the output is 1; when the input is 1, the output is 0.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
NOT	NOT instruction	FC		NOT( );

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Input 1	Follows data type	-	Data 1
Output	Out	Return value	Follows data type	-	Comparison result

  Data type

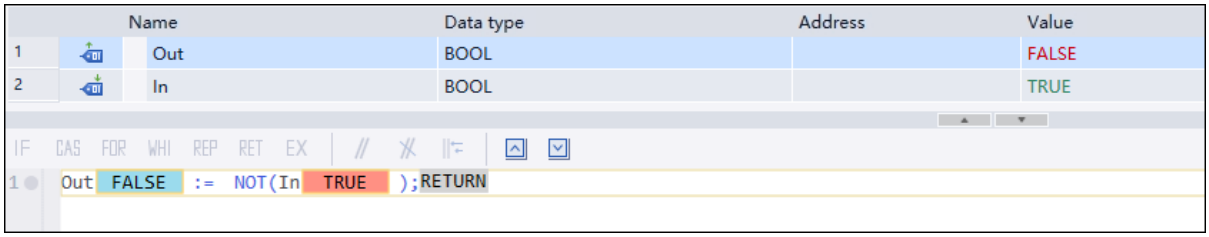
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–
Out	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–

- **Function**

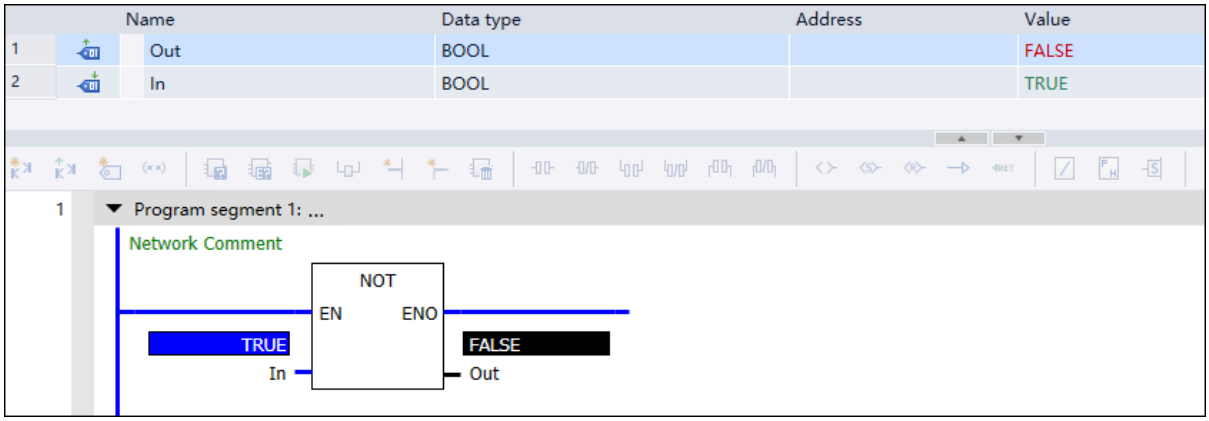
  Inverts each bit of the input parameter.

- **Program example**

  ST:



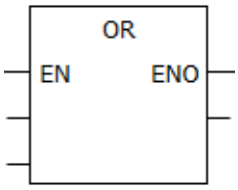
  LD:



18.1.4.4 OR

Bitwise OR: If at least one of the same bit of both inputs is non-zero, the corresponding output bit is 1; otherwise, it is 0.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
OR	OR instruction	FC		OR ;

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Input 1	Follows data type	-	Data 1
	In2	Input 2	Follows data type	-	Data 2
Output	Out	Return value	Follows data type	-	Comparison result

Data type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–

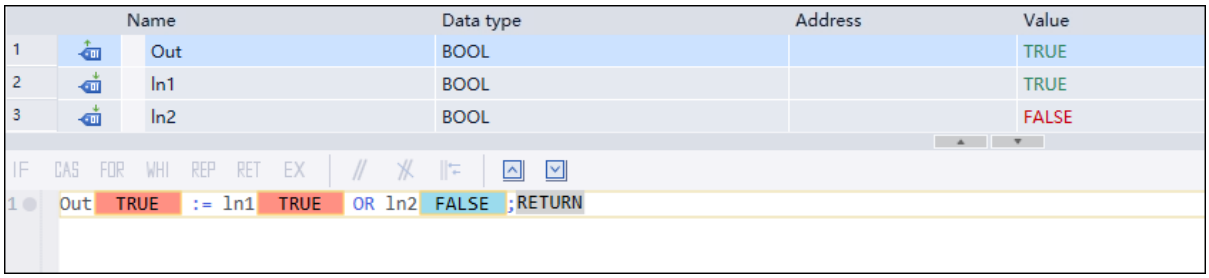
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–
Out	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–

- **Function**

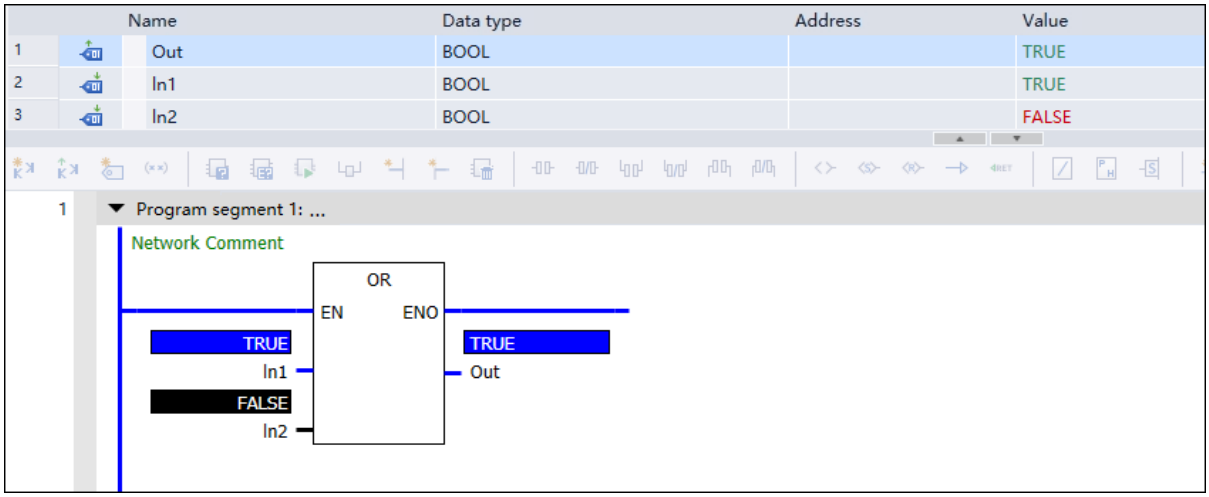
  Bitwise OR: If at least one of the same bit of both inputs is non-zero, the corresponding output bit is 1; otherwise, it is 0.

- **Program example**

  ST:



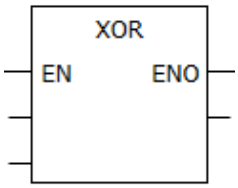
  LD:



18.1.4.5 XOR

Bitwise XOR: When one input is 1 and the other is 0 for the same bit of the two inputs, the corresponding output bit is 1; otherwise, when the two input values are the same for the same bit, the output bit is 0.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
XOR	XOR instruction	FC		XOR ;

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Input 1	Follows data type	-	Data 1
	In2	Input 2	Follows data type	-	Data 2
Output	Out	Return value	Follows data type	-	Comparison result

  Data type

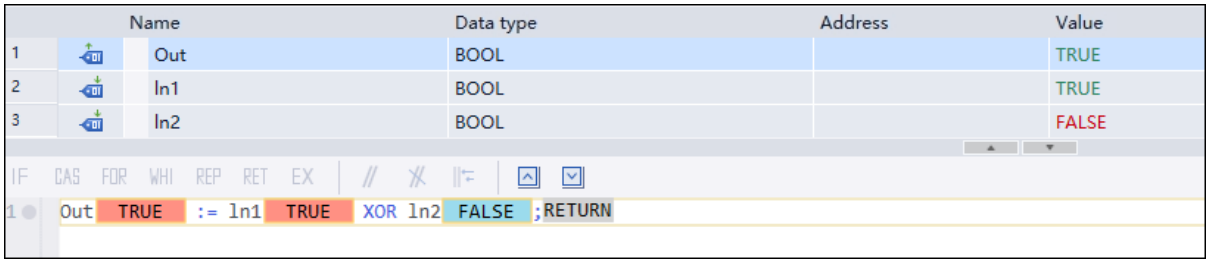
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–
Out	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–

- **Function**

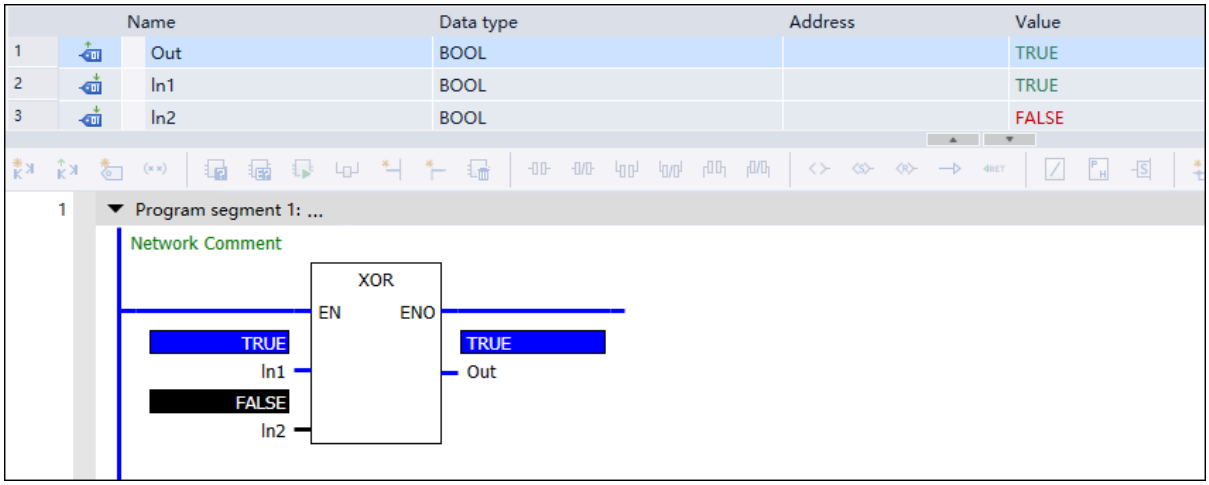
  Bitwise XOR: When one input is 1 and the other is 0 for the same bit of the two inputs, the corresponding output bit is 1; otherwise, when the two input values are the same for the same bit, the output bit is 0.

- **Program example**

  ST:



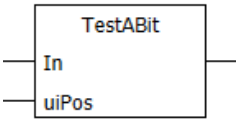
LD:



18.1.4.6 TestABit

TestABit outputs the value at the specified bit in the input.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
TestABit	Outputs the value of the specified bit	FC		:= TestABit(In := , uiPos :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Bit string	Follows data type	-	Bit string
	uiPos	Bit position	0 to ("In" bit count - 1)	0	The position of the specified bit
Output	TestABit	Bit value	Follows data type	-	The value of the specified bit

Data type

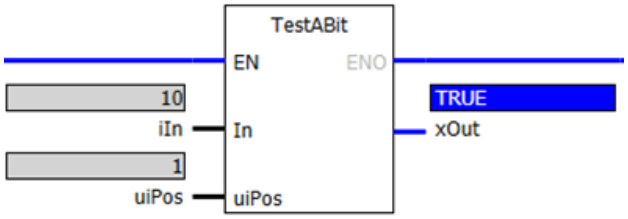
	Boolean	Bit String				Integer								Real Number		Time, duration, date, and text string				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time, duration, date, and text string				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
uiPos	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
TestABit	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function description**

  When EN is TRUE, the value of the bit at position "uiPos" in input "In" is output to the instruction return value "TestABit".

- **Program example**

	LD	ST
Define variables	VAR iIn: INT; uiPos: UINT; xOut: BOOL; END_VAR	
Program		xOut TRUE := TestABit(In := iIn 10 , uiPos := uiPos 1);
Operation result		

	Name	Type	Address	Value
1	iIn	INT		10
2	uiPos	UINT		1
3	xOut	BOOL		TRUE

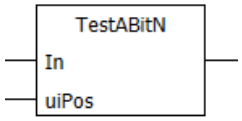
- **Note**

  When uiPos < 0 or uiPos > (number of bits in "In" - 1), "TestABit" returns FALSE.

18.1.4.7 TestABitN

TestABitN outputs the inverted value at the specified bit in the input.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
TestABitN	Outputs the inverted value of the specified bit	FC		:= TestABitN(In := , uiPos :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Bit string	Follows data type	-	Bit string
	uiPos	Bit position	0 to ("In" bit count - 1)	0	The position of the specified bit
Output	TestABitN	Bit value	Follows data type	-	The inverted value of the specified bit

Data type

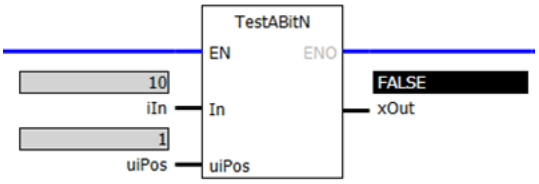
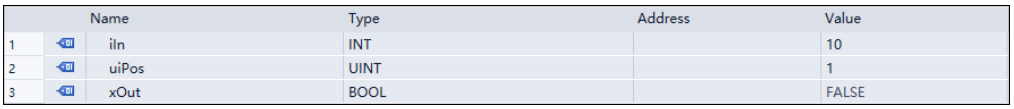
	Boolean	Bit String				Integer								Real Number		Time, duration, date, and text string				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–
uiPos	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
TestABitN	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function description**

  When EN is TRUE, the inverted value of the bit at position "uiPos" in input "In" is output to the instruction return value "TestABit".

- **Program example**

	LD	ST
Define variables	VAR iIn: INT; uiPos: UINT; xOut: BOOL; END_VAR	

	LD	ST
Program		xOut FALSE := TestABitN(In := iIn 10 , uiPos := uiPos 1);
Operation result		

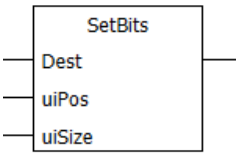
- **Note**

  When uiPos < 0 or uiPos > (number of bits in "In" - 1), "TestABitN" returns FALSE.

18.1.4.8 SetBits

Sets consecutive bits in a bit string.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SetBits	Sets multiple bits consecutively	FC		:= SetBits(Dest := , uiPos := , uiSize :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Dest	Bit string	Follows data type	-	Bit string
	uiPos	Bit position	0 to ("Dest" bit count - 1)	0	The position of the specified bit
	uiSize	Number of bits	0 to "Dest" bit count	0	Number of bits
Output	SetBits	Result	Follows data type	-	TRUE: Operation succeeded. FALSE: Operation failed.

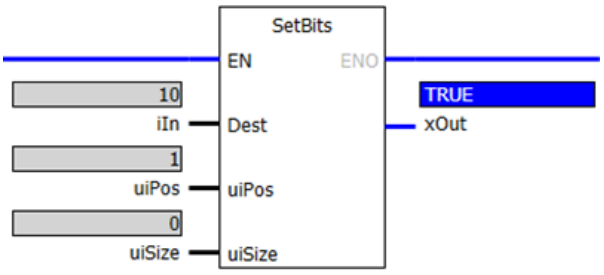
  Data type

	Boolean	Bit String				Integer								Real Number		Time, duration, date, and text string				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Dest	–	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–
uiPos	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
uiSize	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
SetBits	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function description**

  Sets consecutive bits in a bit string to TRUE.

- **Program example**

	LD	ST
Define variables	VAR iIn: INT; uiSize: UINT; xOut: BOOL; END_VAR	
Program		xOut TRUE := SetBits(Dest := iIn 10 , uiPos := uiPos 1 , uiSize := uiSize 0);
Operation result		

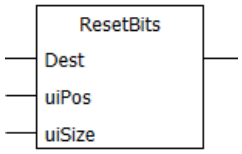
	Name	Type	Address	Value
1	iIn	INT		10
2	uiPos	UINT		1
3	uiSize	UINT		0
4	xOut	BOOL		TRUE

- **Precautions**
  - When (uiPos + uiSize) > bit count of "Dest", the instruction is not executed and returns FALSE.
  - When uiSize is 0, the instruction is executed normally and returns TRUE.

18.1.4.9 ResetBits

Resets consecutive bits in a bit string.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
ResetBits	Resets multiple bits consecutively	FC		:= ResetBits(Dest := , uiPos := , uiSize :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Dest	Bit string	Follows data type	-	Bit string
	uiPos	Bit position	0 to ("Dest" bit count - 1)	0	The position of the specified bit
	uiSize	Number of bits	0 to "Dest" bit count	0	Number of bits
Output	ResetBits	Result	Follows data type	-	TRUE: Operation succeeded. FALSE: Operation failed.

Data type

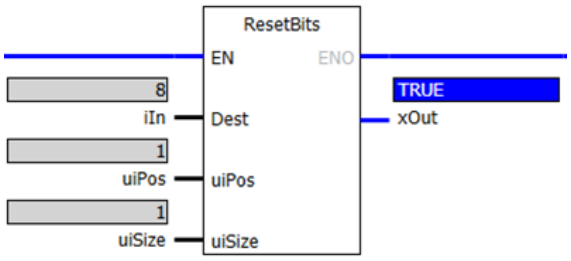
	Boolean	Bit String				Integer								Real Number		Time, duration, date, and text string				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Dest	–	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–
uiPos	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
uiSize	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time, duration, date, and text string				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
ResetBits	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function description**

  Sets consecutive bits in a bit string to FALSE.

- **Program example**

	LD	ST
Define variables	VAR iIn: INT; uiSize: UINT; xOut: BOOL; END_VAR	
Program		xOut TRUE := ResetBits(Dest := iIn 8 , uiPos := uiPos 1 , uiSize := uiSize 1);
Operation result		

Name	Type	Address	Value
iIn	INT		8
uiPos	UINT		1
uiSize	UINT		1
xOut	BOOL		TRUE

- **Precautions**
  - When (uiPos + uiSize) > bit count of "Dest", the instruction is not executed and returns FALSE.
  - When uiSize is 0, the instruction is executed normally and returns TRUE.

18.1.4.10 RESET_BF

This instruction batch resets a BOOL array.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
RESET_BF	Batch reset	FC	??? RESET_BF ???	:= RESET_BF(Dest := , uiSize :=);

- **Related variables**

Variable

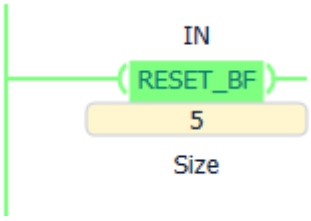
Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Dest	Variable to be reset	Follows data type	-	Variable to be reset.
	uiSize	Reset length	Follows data type	-	Reset length.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Dest	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**
  - Sets the specified number of bits (uiSize) in the Dest variable to FALSE, starting from the specified bit.
  - The Dest input must be a BOOL array with a valid array index; otherwise, a compilation fault will be reported.
  - The input for uiSize must be a constant.
- **Program example**

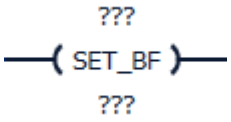
	LD	ST
Define variables	VAR IN: ARRAY[0..10] OF BOOL; Size: UINT; END_VAR	

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>
</td><td>
</td></tr>
<tr><td>Operation result</td><td>
<table>
<tr><th></th><th>Name</th><th>Type</th><th>Address</th><th>Value</th></tr>
<tr><td>1</td><td>VAR</td><td></td><td></td><td></td></tr>
<tr><td>2</td><td>IN</td><td>ARRAY [0..10] OF BOOL</td><td></td><td></td></tr>
<tr><td>3</td><td>IN[0]</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>4</td><td>IN[1]</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>5</td><td>IN[2]</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>6</td><td>IN[3]</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>7</td><td>IN[4]</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>8</td><td>IN[5]</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>9</td><td>IN[6]</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>10</td><td>IN[7]</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>11</td><td>IN[8]</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>12</td><td>IN[9]</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>13</td><td>IN[10]</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>14</td><td>Size</td><td>UINT</td><td></td><td>5</td></tr>
</table>
</td></tr>
</table>

18.1.4.11 SET_BF

This instruction batch sets a BOOL array.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SET_BF	Batch setting	FC		:= SET_BF(Dest := , uiSize :=);

- **Related variables**

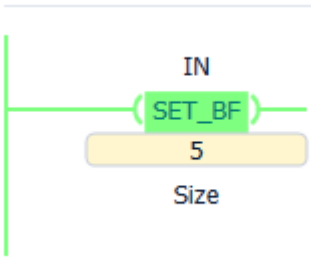
Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Dest	Variable to be set	Follows data type	-	Variable to be set.
	uiSize	Set length	Follows data type	-	Set length

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Dest	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
uiSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**
  - Sets the specified number of bits (uiSize) in the Dest variable to TRUE, starting from the specified bit.
  - The Dest input must be a BOOL array with a valid array index; otherwise, a compilation fault will be reported.
  - The input for uiSize must be a constant.
- **Program example**

	LD	ST
Define variables	VAR IN: ARRAY[0..10] OF BOOL; Size: UINT; END_VAR	
Program		SET_BF(Dest := IN, uiSize := Size 5);RETURN
Operation result		

	Name	Type	Address	Value
1	VAR			
2	IN	ARRAY [0..10] OF BOOL		
3	IN[0]	BOOL		TRUE
4	IN[1]	BOOL		TRUE
5	IN[2]	BOOL		TRUE
6	IN[3]	BOOL		TRUE
7	IN[4]	BOOL		TRUE
8	IN[5]	BOOL		FALSE
9	IN[6]	BOOL		FALSE
10	IN[7]	BOOL		FALSE
11	IN[8]	BOOL		FALSE
12	IN[9]	BOOL		FALSE
13	IN[10]	BOOL		FALSE
14	Size	UINT		5

18.1.5 Mathematical Operations

18.1.5.1 Instruction List

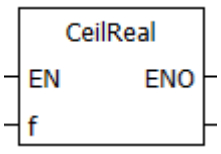
Instruction	FB/FC	Function	Library	Applicable Library Version
18.1.5.2 CeilReal	FC	Round up	Util	Library version 1.0.0.0 and later
18.1.5.3 CeilLReal	FC	Round up		Library version 1.0.0.0 and later
18.1.5.4 FloorReal	FC	Round down		Library version 1.0.0.0 and later
18.1.5.5 FloorLReal	FC	Round down		Library version 1.0.0.0 and later
18.1.5.6 TruncateF	FC	Truncation		Library version 1.0.0.0 and later
18.1.5.7 TruncateL	FC	Truncation		Library version 1.0.0.0 and later
18.1.5.8 fmodLR	FC	Floating-point modulo-division		Library version 1.0.0.0 and later
18.1.5.9 Rand	FB	Pseudorandom number generation	OMUtils	Library version 1.0.0.0 and later
18.1.5.10 CheckReal	FC	Real number check		Library version 1.0.0.0 and later
18.1.5.11 Inc	FC	Increment operation		Library version 1.0.0.0 and later
18.1.5.12 Dec	FC	Decrement operation		Library version 1.0.0.0 and later
18.1.5.13 ModReal	FC	Real number modulo-division		Library version 1.0.0.0 and later
18.1.5.14 ModReal_LR	FC	Long real number modulo-division		Library version 1.0.0.0 and later
18.1.5.15 ABS	FC	Absolute value	-	-
18.1.5.16 ACOS	FC	Arc cosine calculation		
18.1.5.17 ADD	FC	Addition instruction		
18.1.5.18 ASIN	FC	Arc sine calculation		
18.1.5.19 ATAN	FC	Arc tangent calculation		
18.1.5.20 COS	FC	Cosine calculation		
18.1.5.21 DIV	FC	Division instruction		

Instruction	FB/FC	Function	Library	Applicable Library Version
18.1.5.22 EXP	FC	Exponential calculation		
18.1.5.23 EXPT	FC	Exponentiation		
18.1.5.24 LN	FC	Natural logarithm		
18.1.5.25 LOG	FC	Base-10 logarithm		
18.1.5.26 MOD	FC	Modulo-division		
18.1.5.27 MOVE	FC	Value assignment		
18.1.5.28 MUL	FC	Multiplication instruction		
18.1.5.29 Sin	FC	Sine calculation		
18.1.5.30 SIZEOF	FC	Size calculation		
18.1.5.31 SQRT	FC	Square root calculation		
18.1.5.32 SUB	FC	Subtraction instruction		
18.1.5.33 TAN	FC	Tangent calculation		
18.1.5.34 RAD	FC	Degrees to radians	Util	Library version 1.2.0.0 and later
18.1.5.35 DEG	FC	Radians to degrees		Library version 1.2.0.0 and later
18.1.5.36 SCALE_X	FC	Scaling instruction	OMUtils	Library version 1.3.0.0 and later
18.1.5.37 NORM_X	FC	Normalization instruction		Library version 1.3.0.0 and later

18.1.5.2 CeilReal

Rounds up to get the smallest integer greater than or equal to the input value.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CeilReal	Round up	FC		:= CeilReal(f :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	f	The input floating-point value	Follows data type	-	The input long floating-point value
Output	Out	Return value	Follows data type	-	The rounded-up value

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
f	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–

- **Function**

  Rounds up to get the smallest integer greater than or equal to the input value.

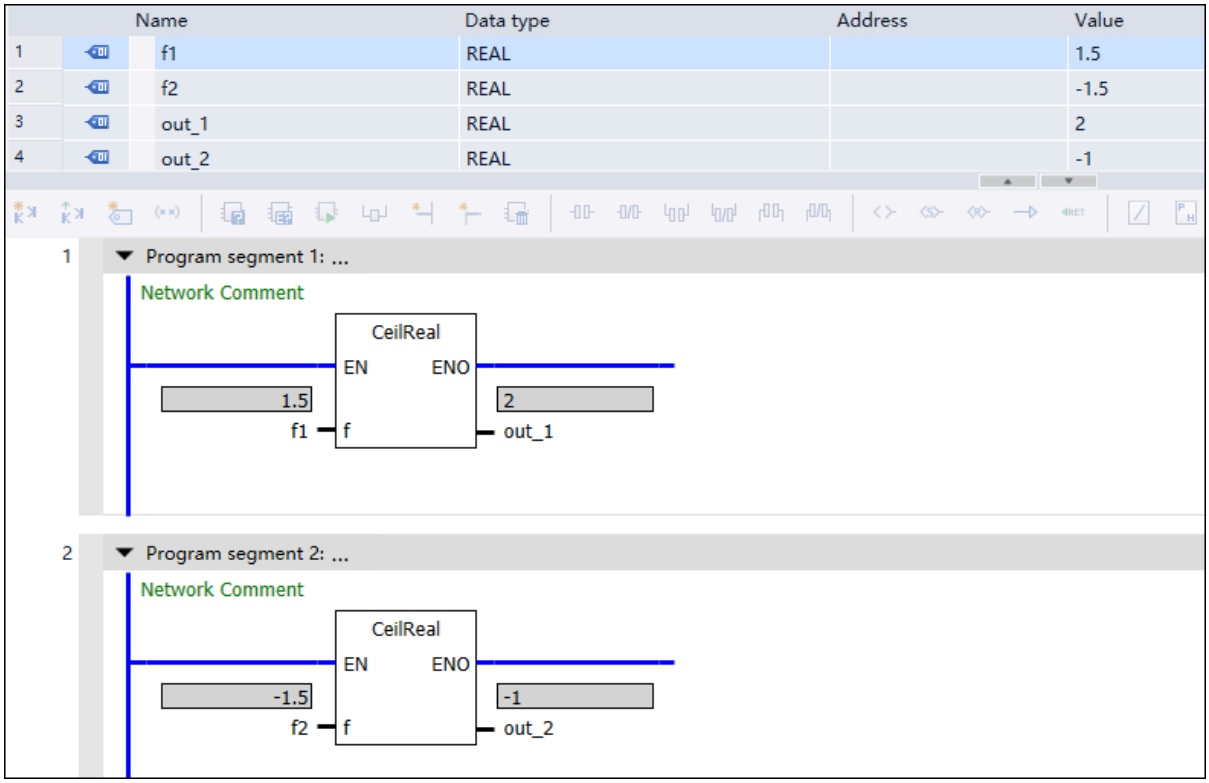
- **Program example**

  ST:

	Name	Data type	Address	Value
1	f1	REAL		1.5
2	f2	REAL		-1.5
3	out_1	REAL		2
4	out_2	REAL		-1

```
IF  CAS  FOR  WHI  REP  RET  EX
1  out_1  2  := CeilReal(f := f1  1.5  );
2  out_2  -1  := CeilReal(f := f2  -1.5  );RETURN
```

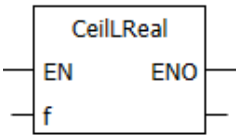
  LD:



18.1.5.3 CeilLReal

Rounds up to get the smallest integer greater than or equal to the input value.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CeilLReal	Round up	FC		:= CeilLReal(f :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	f	The input long floating-point value	Follows data type	-	The input long floating-point value
Output	Out	Return value	Follows data type	-	The rounded-up value

Variable types

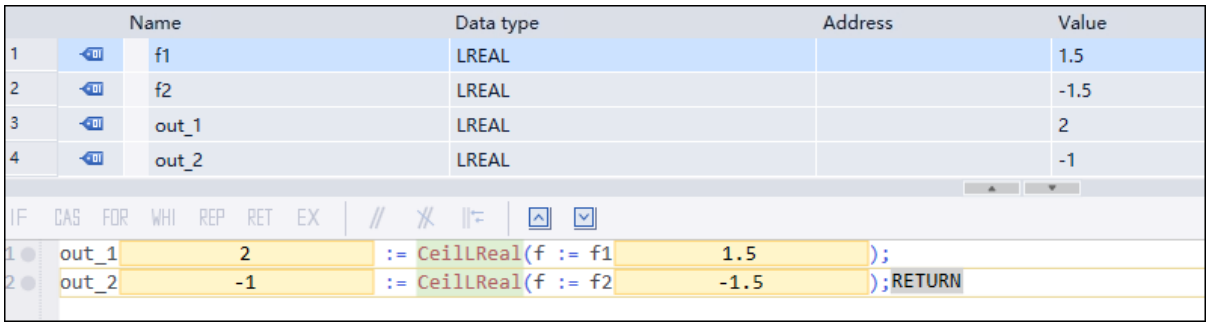
Variable Name	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
f	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–

- **Function**

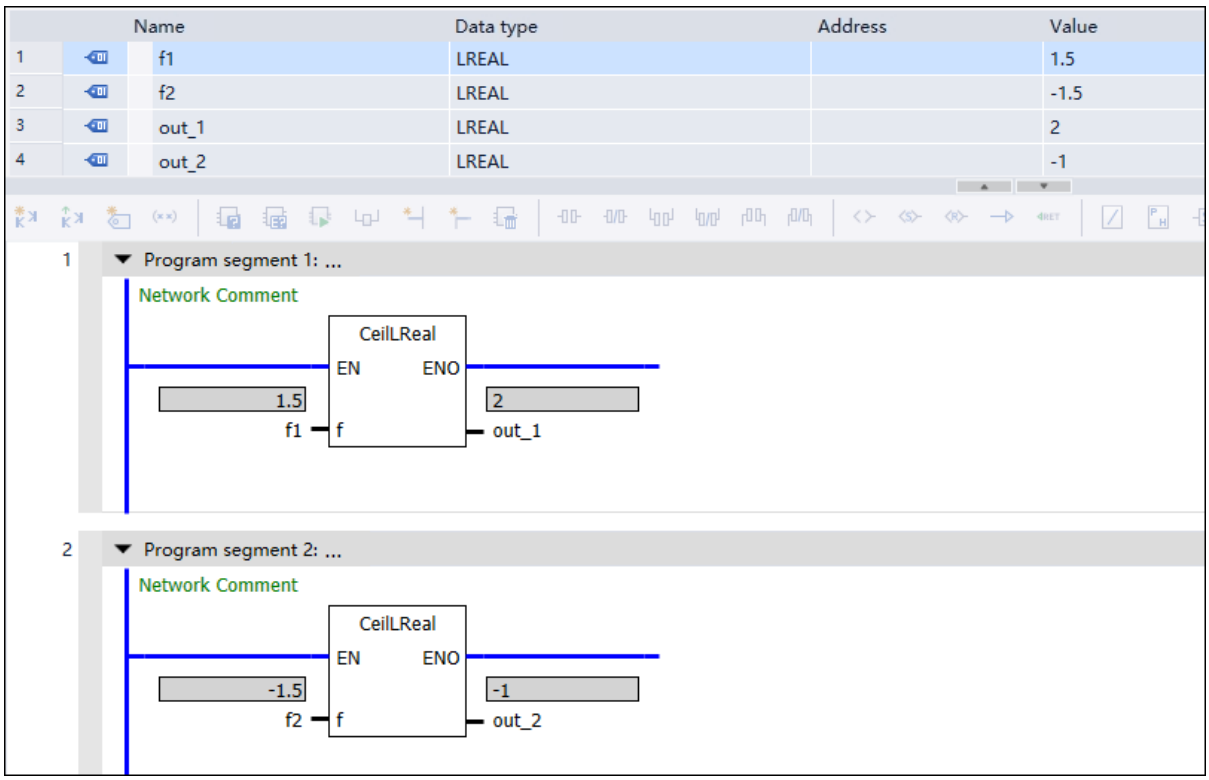
  Rounds up to get the smallest integer greater than or equal to the input value.

- **Program example**

  ST:



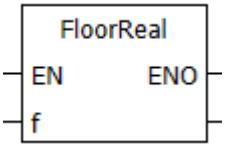
  LD:



18.1.5.4 FloorReal

Rounds down to get the largest integer less than or equal to the input value.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FloorReal	Round down	FC		:= FloorReal(f :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	f	The input floating-point value	Follows data type	-	The input long floating-point value
Output	Out	Return value	Follows data type	-	The rounded-down value

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
f	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–

- **Function**

  Rounds down to get the largest integer less than or equal to the input value.

- **Program example**

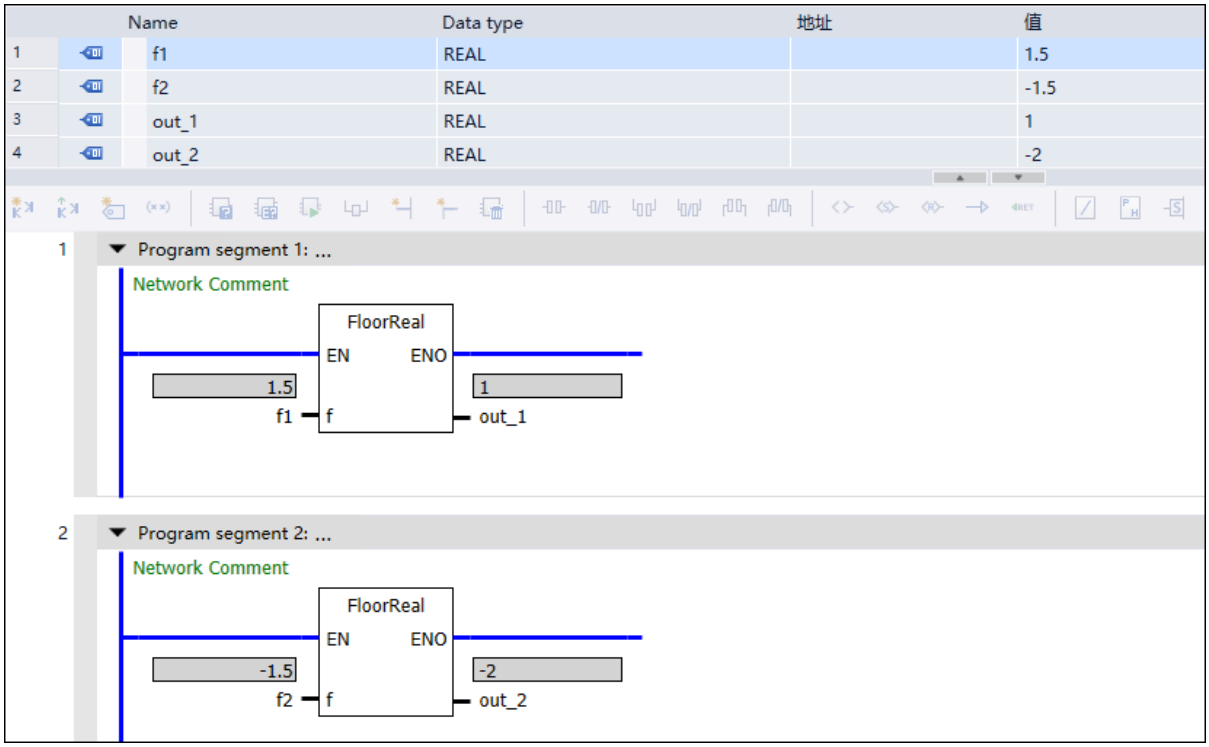
  ST:

	Name	Data type	Address	Value
1	f1	REAL		1.5
2	f2	REAL		-1.5
3	out_1	REAL		1
4	out_2	REAL		-2

```
1  out_1  1   := FloorReal(f := f1  1.5  );
2  out_2  -2  := FloorReal(f := f2  -1.5 );RETURN
```

LD:

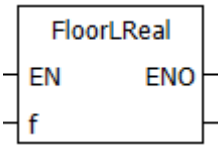
	Name	Data type	地址	值
1	f1	REAL		1.5
2	f2	REAL		-1.5
3	out_1	REAL		1
4	out_2	REAL		-2



18.1.5.5 FloorLReal

Rounds down to get the largest integer less than or equal to the input value.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FloorLReal	Round down	FC		:= FloorLReal(f :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	f	The input long floating-point value	Follows data type	-	The input long floating-point value
Output	Out	Return value	Follows data type	-	The rounded-down value

Variable types

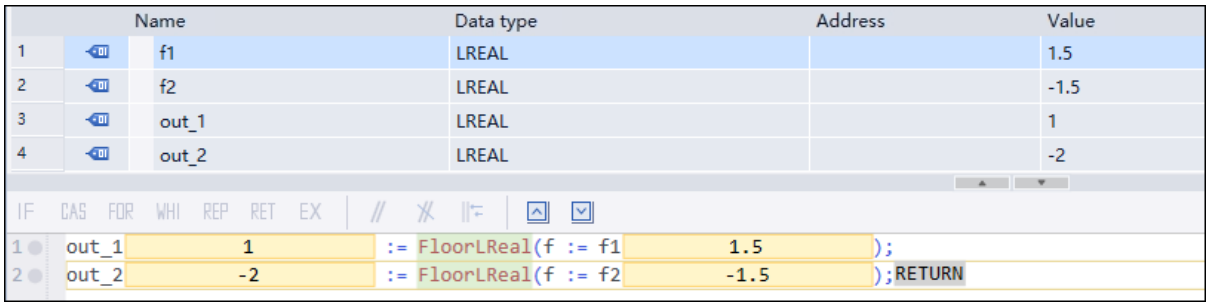
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
f	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–

- **Function**

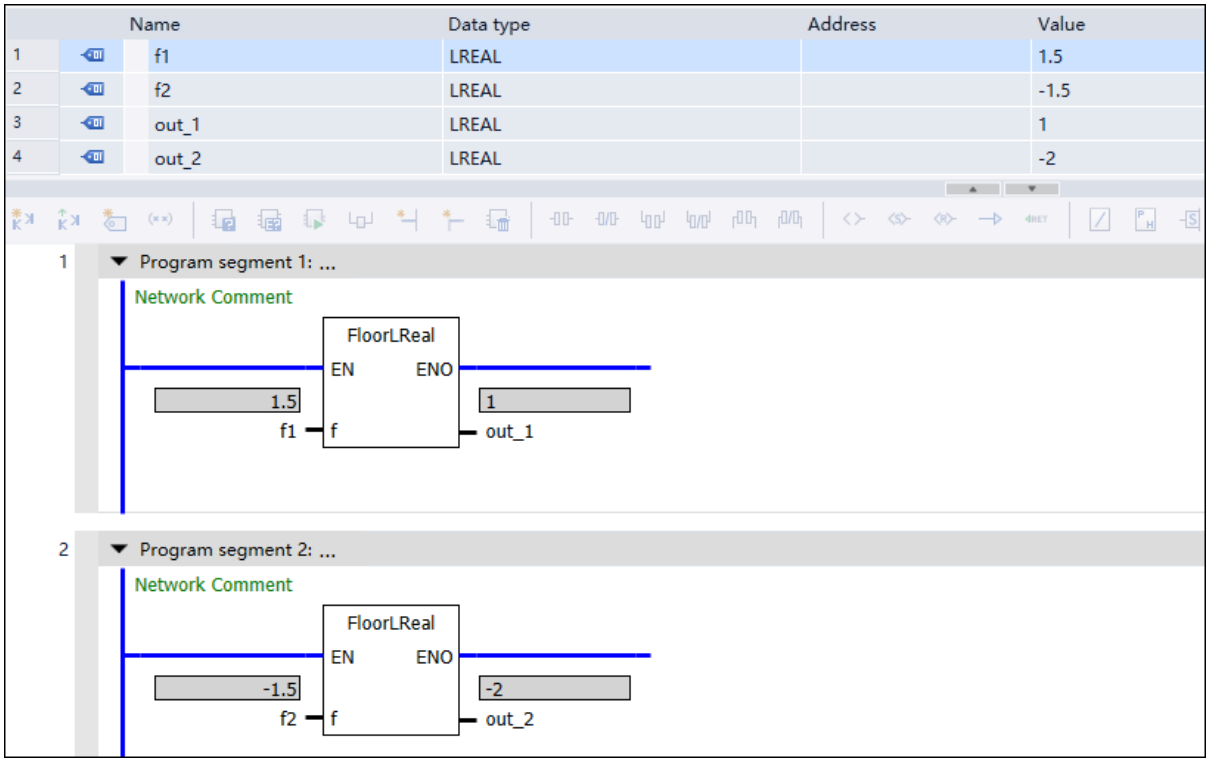
  Rounds down to get the largest integer less than or equal to the input value.

- **Program example**

  ST:



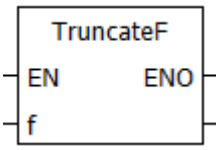
  LD:



18.1.5.6 TruncateF

Truncates a floating-point number to get its integer part.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
TruncateF	Truncation	FC		:= TruncateF(f :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	f	The input floating-point value	Follows data type	-	The input long floating-point value
Output	Out	Return value	Follows data type	-	The truncated value

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
f	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–

- **Function**

  Truncates a floating-point number to get its integer part.

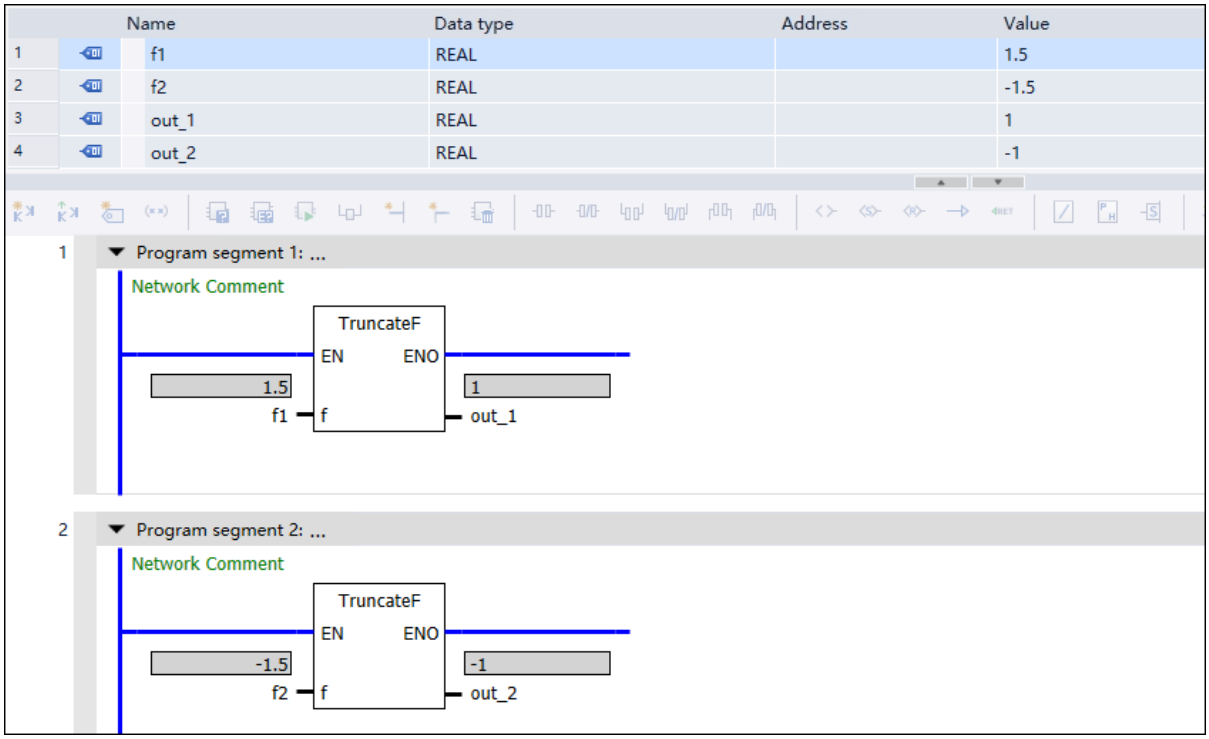
- **Program example**

  ST:

	Name	Data type	Address	Value
1	f1	REAL		1.5
2	f2	REAL		-1.5
3	out_1	REAL		1
4	out_2	REAL		-1

```
1  out_1  1    := TruncateF(f := f1  1.5  );
2  out_2  -1   := TruncateF(f := f2  -1.5 );RETURN
```

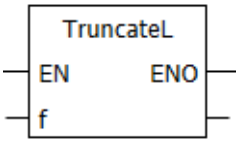
LD:



18.1.5.7 TruncateL

Truncates a long floating-point number to get its integer part.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
TruncateL	Truncation	FC		:= TruncateL(f :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	f	The input long floating-point value	Follows data type	-	The input long floating-point value
Output	Out	Return value	Follows data type	-	The truncated value

Variable types

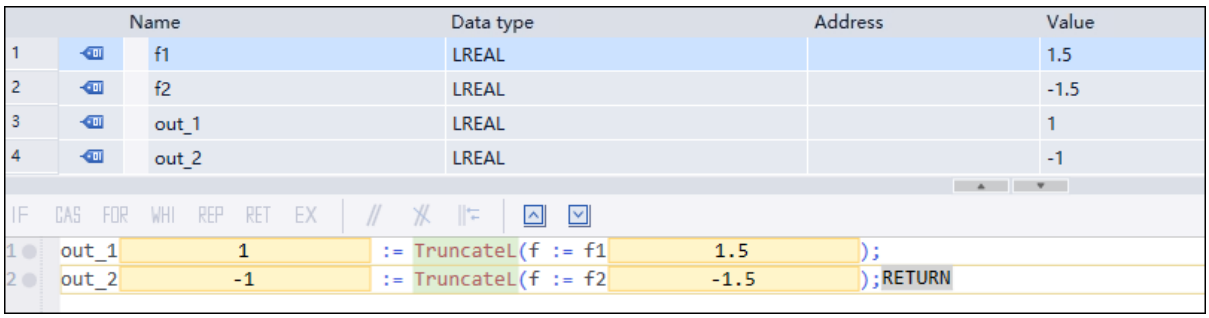
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
f	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–

- **Function**

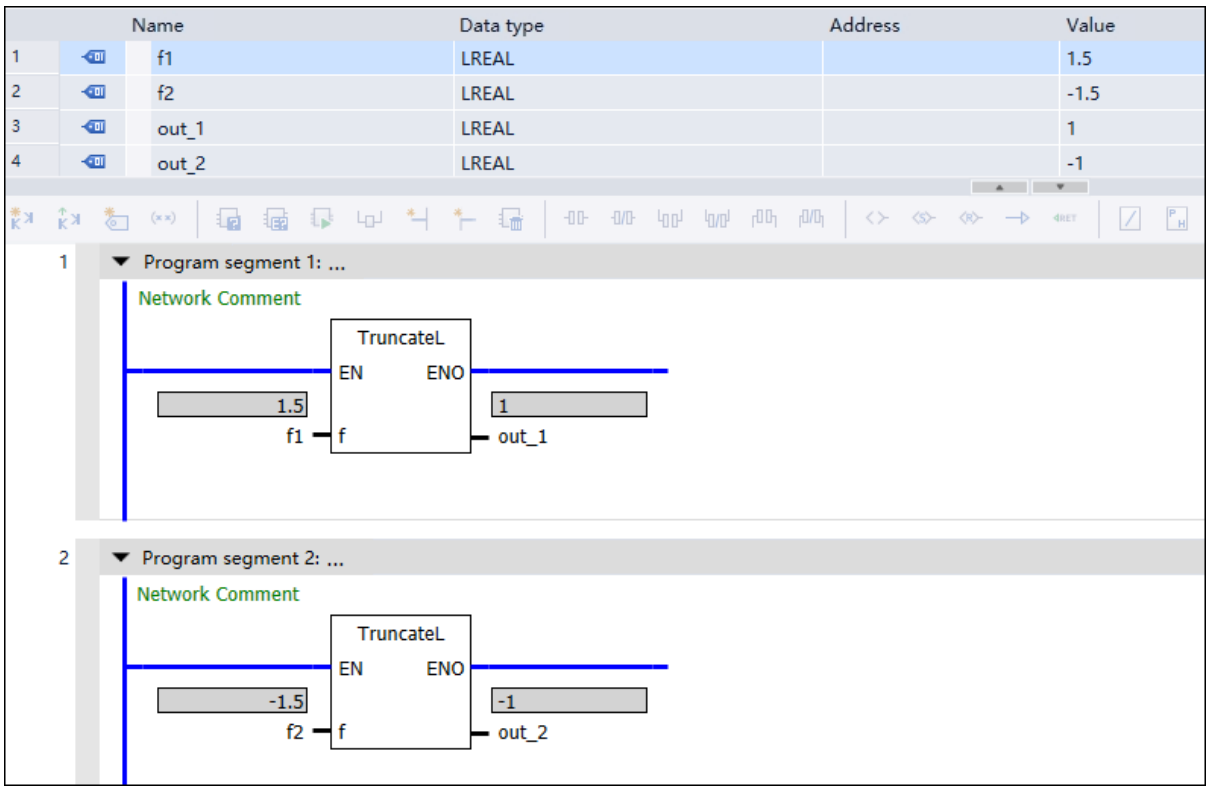
  Truncates a long floating-point number to get its integer part.

- **Program example**

  ST:



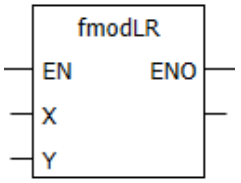
  LD:



18.1.5.8 fmodLR

Calculates the remainder of floating-point number division.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
fmodLR	Floating-point modulo-division	FC		:= fmodLR(X := , Y :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	X	Dividend	Follows data type	-	Dividend
	Y	Divider	Follows data type	-	Divider
Output	Out	Remainder	Follows data type	-	Remainder

Variable types

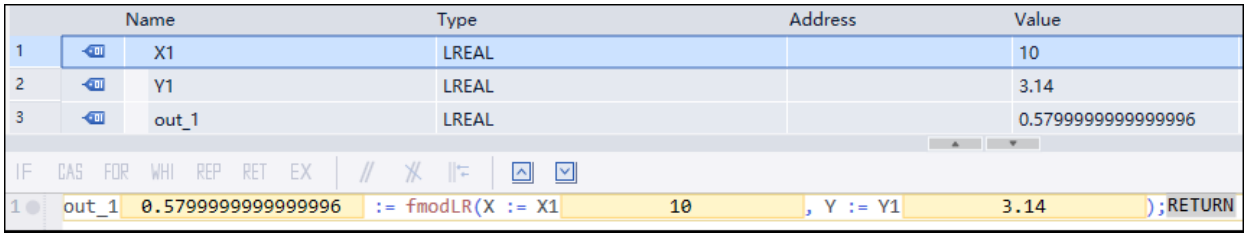
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
X	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Y	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–

- **Function**

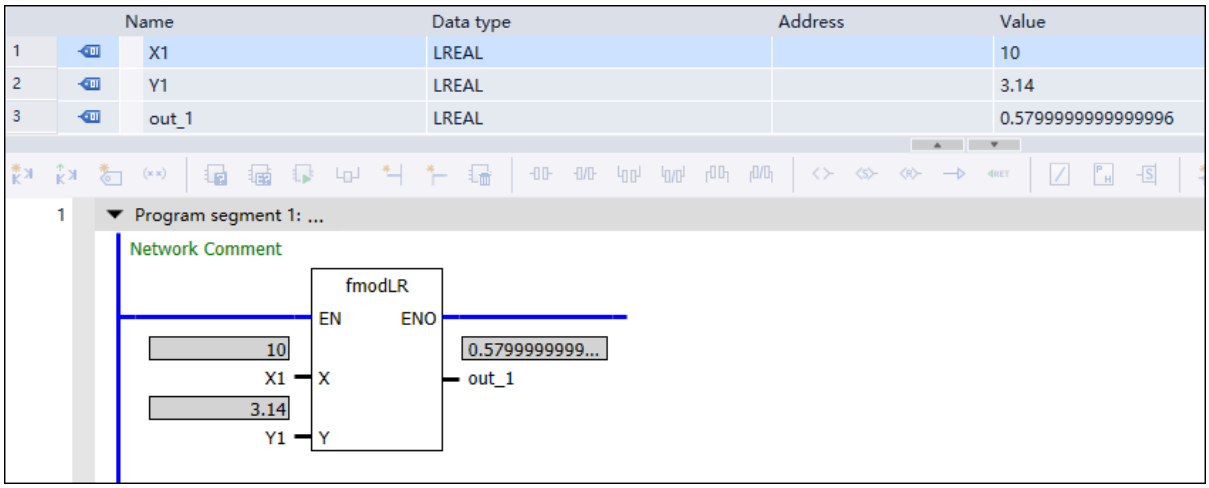
Calculates the remainder of floating-point number division.

- **Program example**

ST:



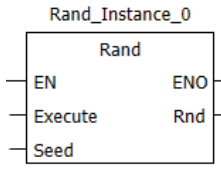
LD:



18.1.5.9 Rand

Generates pseudorandom numbers.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
Rand	Pseudorandom number generation	FB		Rand_Instance_0(Execute := , Seed := , Rnd =>);

- **Related variables**

Variable

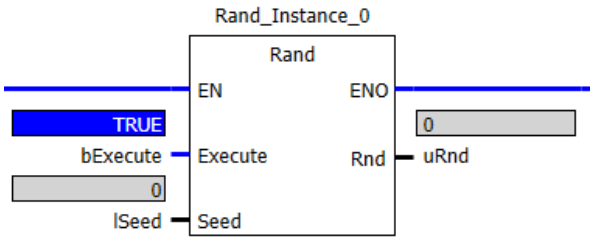
Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Triggering	Follows data type	-	A trigger for random number generation
	Seed	Random number mode	Follows data type	0	Random number sequence (0 means not specified).

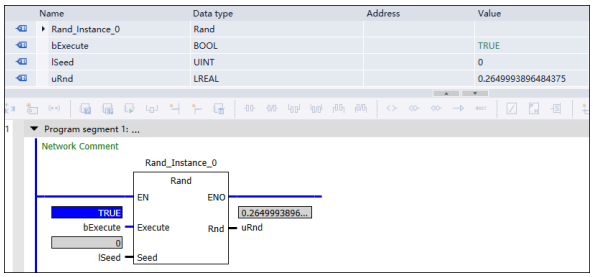
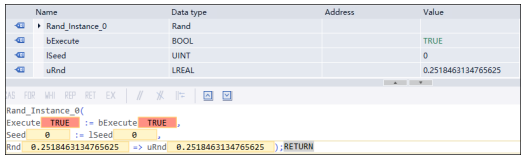
Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Rnd	Random number	0.000000000000000e+0 to 1.000000000000000e+0	-	Random number

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Seed	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Rnd	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–

- **Function**
  - Generates a random number "Rnd", whose value changes each time the instruction is executed.
  - The random number mode "Seed" is used to specify the random number sequence. If the value of "Seed" is the same, the same random number sequence is generated each time the program runs. Therefore, reproducible random numbers can be generated.
  - When the value of "Seed" is 0, irreproducible random numbers are generated. To avoid generating the same random number sequence each time the program runs, set the value of "Seed" to 0.
- **Program example**

	LD	ST
Define variables	VAR Rand_Instance_0: Rand; bExecute: BOOL := TRUE; lSeed: UINT; uRnd: LREAL; END_VAR	
Program		Rand_Instance_0(Execute FALSE := bExecute TRUE, Seed 0 := lSeed 0, Rnd 0 => uRnd 0);RETURN

	LD	ST
Operation result		

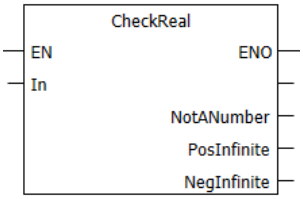
- **Precautions**

  The value of "Rand" is a real number between 0 and 1.

18.1.5.10 CheckReal

Checks whether a real number is infinite or non-numeric data.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CheckReal	Real number check	FC		:= CheckReal(In := , NotANumber => , PosInfinite => , NegInfinite =>);

- **Related variables**

  Input variables

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Real Number	Follows data type	-	Real number
Output	NotANumber	Non-numeric data check result	FALSE, TRUE	FALSE	TRUE: Non-numeric data FALSE: Not non-numeric data
	PosInfinite	Positive infinity check result	FALSE, TRUE	FALSE	TRUE: Positive infinity FALSE: Not positive infinity
	NegInfinite	Negative infinity check result	FALSE, TRUE	FALSE	TRUE: Negative infinity FALSE: Not negative infinity

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–
NotANumber	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PosInfinite	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
NegInfinite	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

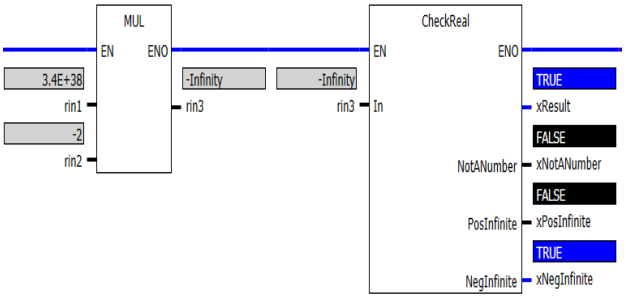
- **Function**

  Checks whether the real number "In" is non-numeric data, positive infinity, or negative infinity. The corresponding results are output to "NotANumber", "PosInfinite", and "NegInfinite", respectively.

- **Program example**

	LD	ST
Define variables		
Program		

```
VAR
    xResult: BOOL;
    rin1: REAL:=3.4E+38;
    rin2: REAL:=-2;
    rin3: REAL;
    xNotANumber: BOOL;
    xPosInfinite: BOOL;
    xNegInfinite: BOOL;
END_VAR
```




```
rin3 -Infinity :=rin1 3.4E+38 *rin2 -2 ;
xResult TRUE := CheckReal(In := rin3 -Infinity ,
    NotANumber => xNotANumber FALSE ,
    PosInfinite =>xPosInfinite FALSE ,
    NegInfinite =>xNegInfinite TRUE );RETURN
```

	LD	ST

Operation result

Name	Data type	Address	Value
xResult	BOOL		TRUE
rin1	REAL		3.4E+38
rin2	REAL		-2
rin3	REAL		-Infinity
xNotANumber	BOOL		FALSE
xPosInfinite	BOOL		FALSE
xNegInfinite	BOOL		TRUE

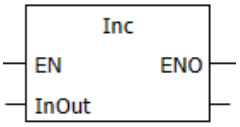
- **Precautions**

  In OMUtils library versions 1.2.0.1 and earlier, a compilation fault occurs if the parameter passed to "In" is not a real number.
  In OMUtils library versions 1.3.0.0 and later, no compilation fault occurs if the parameter passed to "In" is an integer.

18.1.5.11 Inc

Increments an integer value.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
Inc	Increment operation	FC		:= Inc(InOut :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	InOut	Input data	Follows data type	-	Input data

  Variable types

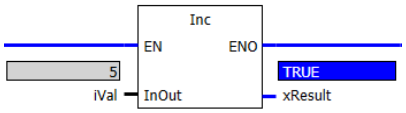
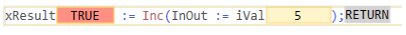
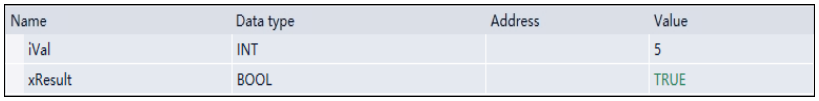
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
InOut	–	–	–	–	–	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–

- **Function**

Increments the input data "InOut". When the calculation result exceeds the maximum value of "InOut", it resets to the minimum value.

- **Program example**

  The following table shows examples of the Inc instruction. When the input value is INT#4, the value of "InOut" after instruction execution is INT#5.

	LD	ST
Define variables	VAR iVal: INT := 4; xResult: BOOL; END_VAR	
Program		
Operation result		

Name	Data type	Address	Value
iVal	INT		5
xResult	BOOL		TRUE

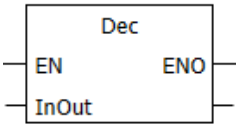
- **Precautions**

  The function returns TRUE.

18.1.5.12 Dec

Decrements an integer value.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
Dec	Decrement operation	FC		:= Dec(InOut :=);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	InOut	Input data	Follows data type	-	Input data

  Variable types

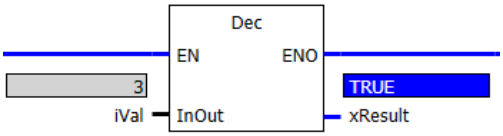
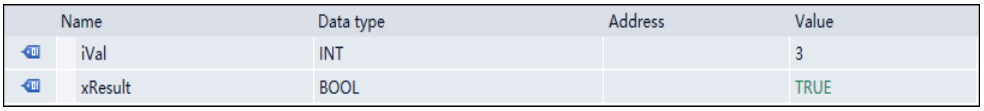
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In Out	–	–	–	–	–	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–

- **Function**

  Decrements the input data "InOut". When the calculation result is less than the minimum value of "InOut", it resets to the maximum value.

- **Program example**

  - The following table shows examples of the Dec instruction. When the input value is INT#4, the value of "InOut" after executing the instruction is INT#3.

	LD	ST
Define variables	VAR iVal :INT := 4; xResult:BOOL; END_VAR	
Program		xResult TRUE := Dec(InOut := iVal 3);RETURN
Operation result		

	Name	Data type	Address	Value
	iVal	INT		3
	xResult	BOOL		TRUE

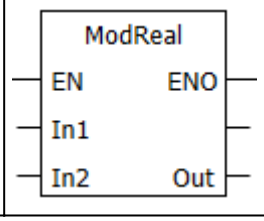
- **Precautions**

  The function returns TRUE.

18.1.5.13 ModReal

Calculates the remainder of real number division.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ModReal	Real number modulo-division	FC		:= ModReal(In1 := , In2 := , Out =>);

- **Related variable**

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Dividend	Follows data type	-	Dividend
	In2	Divider	Follows data type	-	Divider
Output	Out	Remainder	Follows data type	-	Remainder

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
In2	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

- **Function**

  Calculates the remainder obtained by dividing the dividend "In1" by the divisor "In2". The instruction is performed according to the following expression.

  Out = In1 - (In1 / In2) In2, where the division within parentheses truncates the decimal part. The following table shows example values for "In1", "In2", and "Out":

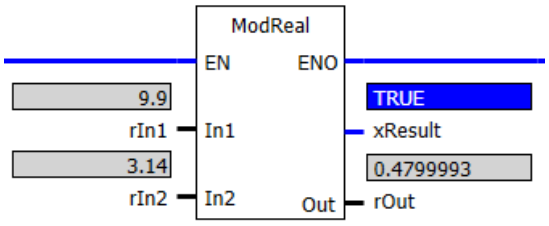
Value of In1	Value of In2	Value of Out
9.9	3.14	0.48
9.9	-3.14	0.48
-9.9	3.14	-0.48
-9.9	-3.14	-0.48

- **Program example**

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    rIn1: REAL := 9.9;
    rIn2: REAL := 3.14;
    rOut: REAL;
    xResult: BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
xResult TRUE := ModReal(In1 := rIn1 9.9 ,
        In2 := rIn2 3.14 ,
        Out => rOut 0.4799993 );RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

Name	Data type	Address	Value
rIn1	REAL		9.9
rIn2	REAL		3.14
rOut	REAL		0.4799993
xResult	BOOL		TRUE

</td></tr>
</table>

- **Precautions**

  When passing an integer parameter to "In1" or "In2", the data type is converted as follows before inputting.

Data Type of Parameter Passed to In1 or In2	Data Type of In1 or In2
USIN, UINT, SINT, INT	REAL
UDINT, DINT, ULINT, LINT	Data may be lost

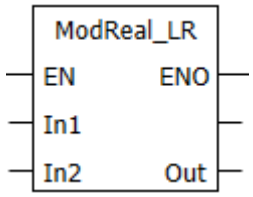
Depending on the values in "In1" and "In2", the value of "Out" is as shown in the table below.

0		In1				
		0	Number	+∞	-∞	Non-numeric
In2	0	Non-numeric	Non-numeric	Non-numeric	Non-numeric	Non-numeric
	Number	0	Remainder of "In1" divided by "In2"	Non-numeric	Non-numeric	Non-numeric
	+∞	0	Value of "In1"	Non-numeric	Non-numeric	Non-numeric
	-∞	0	Value of "In1"	Non-numeric	Non-numeric	Non-numeric
	Non-numeric	Non-numeric	Non-numeric	Non-numeric	Non-numeric	Non-numeric

18.1.5.14 ModReal_LR

Calculates the remainder of long real number division.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ModReal_LR	Long real number modulo-division	FC		:= ModReal_LR(In1 := , In2 := , Out =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Dividend	Follows data type	-	Dividend
	In2	Divider	Follows data type	-	Divider
Output	Out	Remainder	Follows data type	-	Remainder

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
In2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–

- **Function**

Calculates the remainder obtained by dividing the dividend "In1" by the divisor "In2". The instruction is performed according to the following expression.

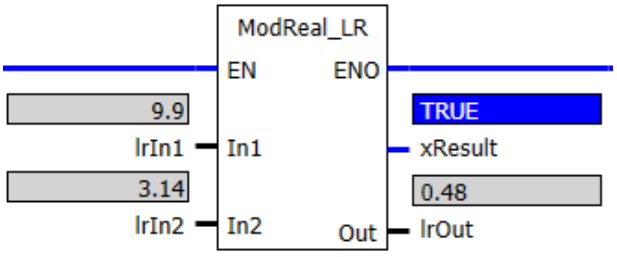
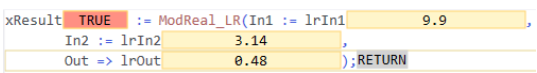
Out = In1 - (In1 / In2) In2, where the division within parentheses truncates the decimal part. The following table shows example values for "In1", "In2", and "Out":

Value of In1	Value of In2	Value of Out
9.9	3.14	0.48
9.9	-3.14	0.48
-9.9	3.14	-0.48
-9.9	-3.14	-0.48

- **Program example**

	LD	ST

Define variables	VAR
	lrIn1: LREAL := 9.9;
	lrIn2: LREAL := 3.14;
	lrOut: LREAL;
	xResult: BOOL;
	END_VAR	

Program		
Operation result		

	Name	Data type	Address	Value
	lrIn1	LREAL		9.9
	lrIn2	LREAL		3.14
	lrOut	LREAL		0.48
	xResult	BOOL		TRUE

- **Precautions**

When passing an integer parameter to "In1" or "In2", the data type is converted as follows before inputting.

Data Type of Parameter Passed to In1 or In2	Data Type of In1 or In2
USINT, UINT, SINT, INT, UDINT, DINT	LREAL
ULINT, LINT	Data may be lost

Depending on the values in "In1" and "In2", the value of "Out" is as shown in the table below.

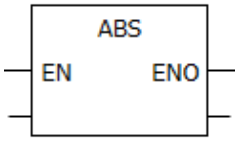
0		In1				
		0	Number	+∞	-∞	Non-numeric
In2	0	Non-numeric	Non-numeric	Non-numeric	Non-numeric	Non-numeric

0		In1				
		0	Number	+∞	-∞	Non-numeric
	Number	0	Remainder of "In1" divided by "In2"	Non-numeric	Non-numeric	Non-numeric
	+∞	0	Value of "In1"	Non-numeric	Non-numeric	Non-numeric
	-∞	0	Value of "In1"	Non-numeric	Non-numeric	Non-numeric
	Non-numeric	Non-numeric	Non-numeric	Non-numeric	Non-numeric	Non-numeric

18.1.5.15 ABS

Assigns the absolute value of the input data to the output variable.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ABS	Absolute value	FC		ABS()

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Input Variable	Follows data type	-	Variable to be calculated
Output	Out	Return value	Follows data type	-	Calculation result

Data type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–

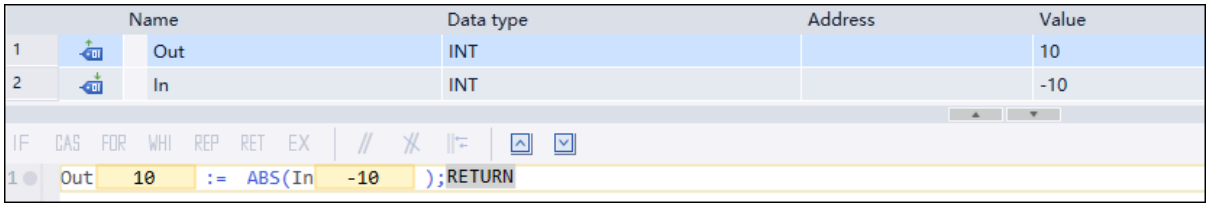
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–

- **Function**

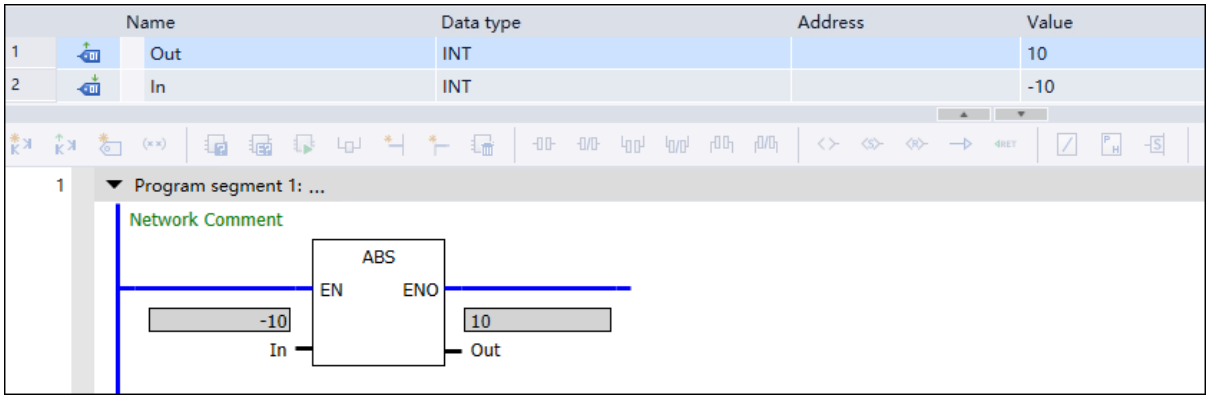
  Assigns the absolute value of the input data to the output variable.

- **Program example**

  ST:



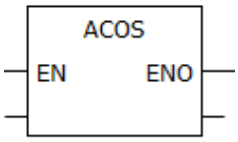
  LD:



18.1.5.16 ACOS

Calculates the arc cosine of the input value and outputs the result in radians.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ACOS	Arc cosine calculation	FC		ACOS()

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Input Variable	Follows data type	-	Variable to be calculated
Output	Out	Return value	Follows data type	-	Calculation result

Data type

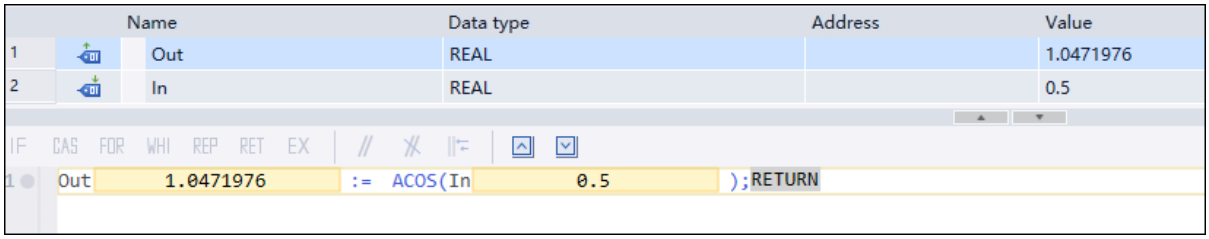
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–

- **Function**

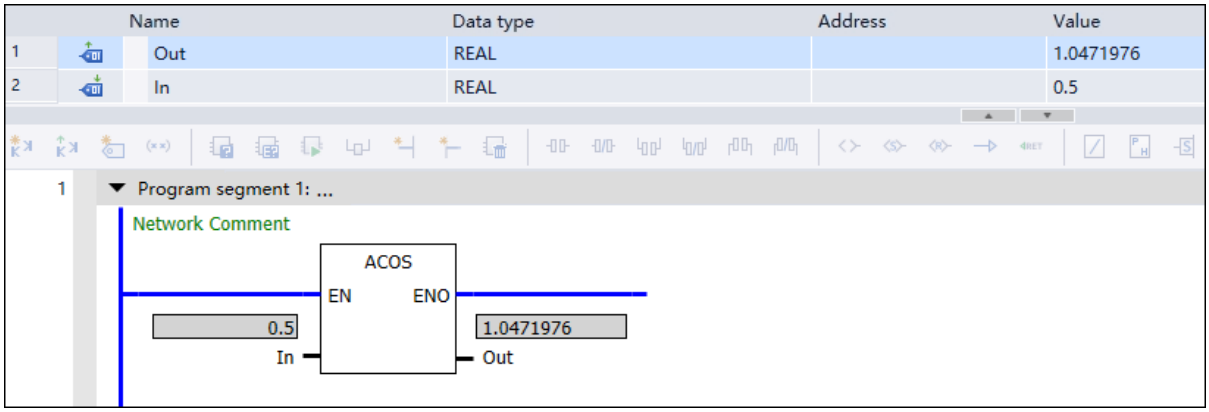
  Calculates the arc cosine of the input value and outputs the result in radians.

- **Program example**

  ST:



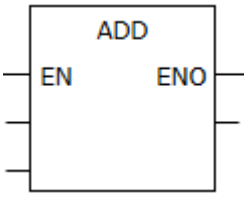
  LD:



18.1.5.17 ADD

Adds the two inputs on the left and outputs the addition result on the right.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ADD	Addition instruction	FC		+ ;

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Value to be added	Follows data type	-	Value 1 to be added
	In2	Value to be added	Follows data type	-	Value 2 to be added
Output	Out	Return value	Follows data type	-	Calculation result

  Data type

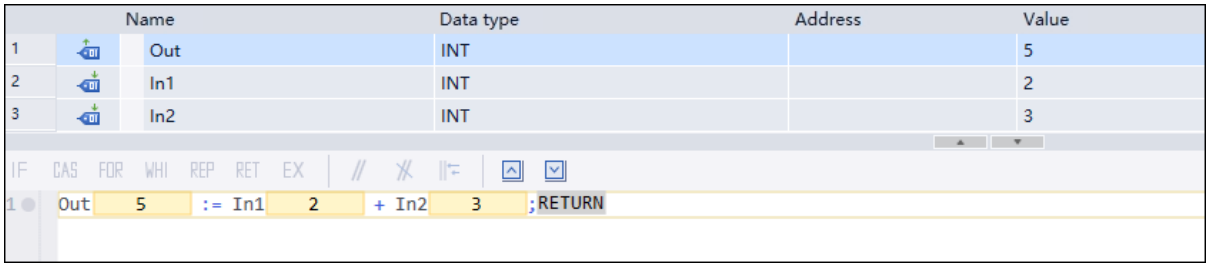
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–
In2	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–
Out	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–

- **Function**

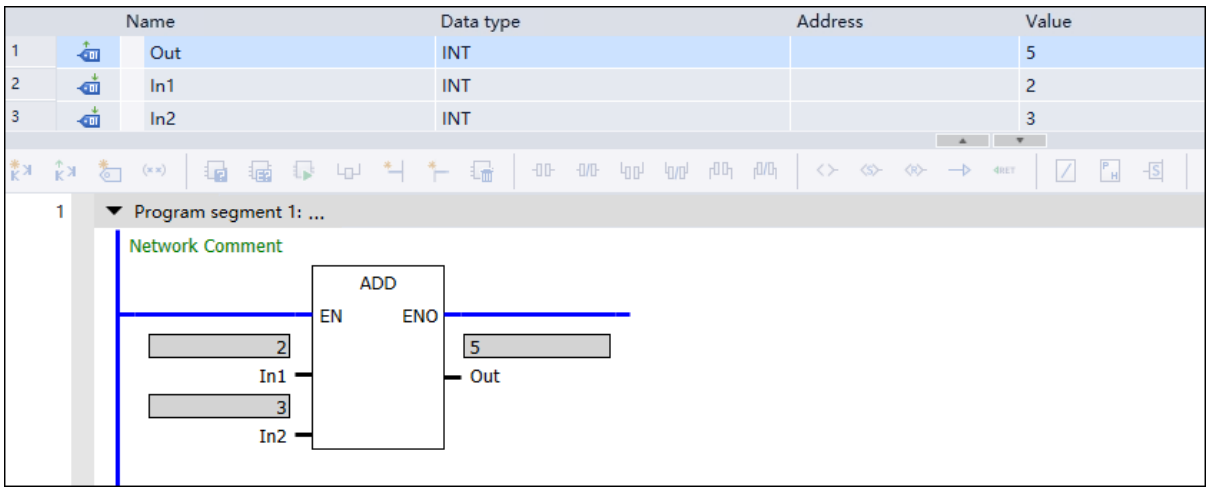
  Adds the two inputs on the left and outputs the addition result on the right.

- **Program example**

  ST:



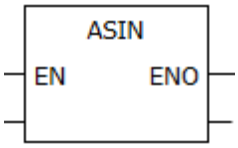
LD:



18.1.5.18 ASIN

Calculates the arc sine of the input value and outputs the result in radians.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ASIN	Arc sine calculation	FC		ASIN()

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Input Variable	Follows data type	-	Variable to be calculated
Output	Out	Return value	Follows data type	-	Calculation result

Data type

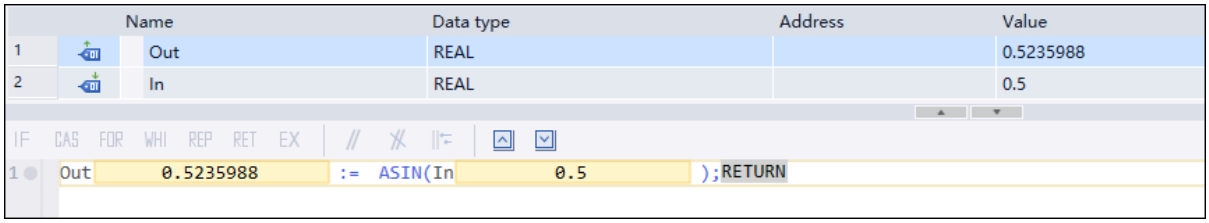
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–

- **Function**

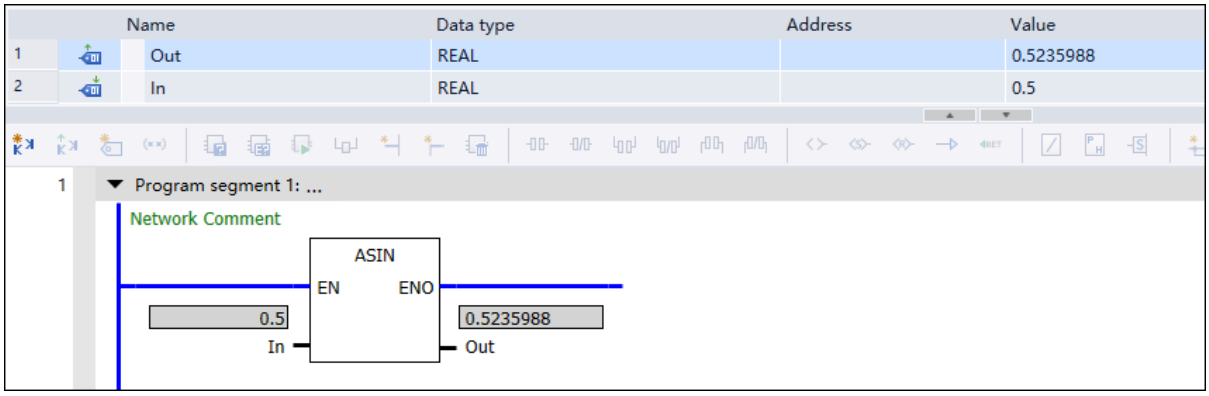
  Calculates the arc sine of the input value and outputs the result in radians.

- **Program example**

  ST:



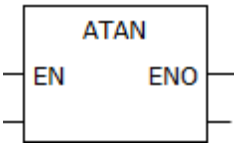
  LD:



18.1.5.19 ATAN

Calculates the arc tangent of the input value and outputs the result in radians.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ATAN	Arc tangent calculation	FC		ATAN()

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Input Variable	Follows data type	-	Variable to be calculated
Output	Out	Return value	Follows data type	-	Calculation result

Data type

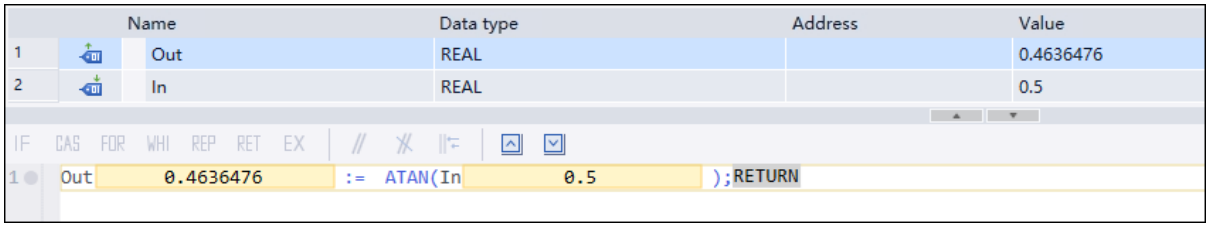
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–

- **Function**

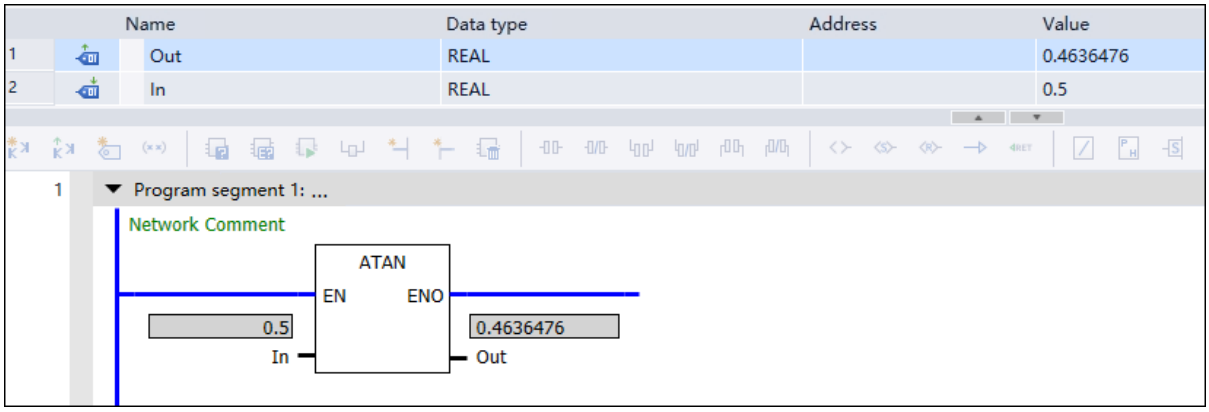
  Calculates the arc tangent of the input value and outputs the result in radians.

- **Program example**

  ST:



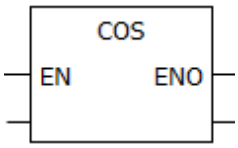
  LD:



18.1.5.20 COS

Calculates the cosine value of the input, and outputs the result.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
COS	Cosine calculation	FC		COS()

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Input Variable	Follows data type	-	Variable to be calculated
Output	Out	Return value	Follows data type	-	Calculation result

  Data type

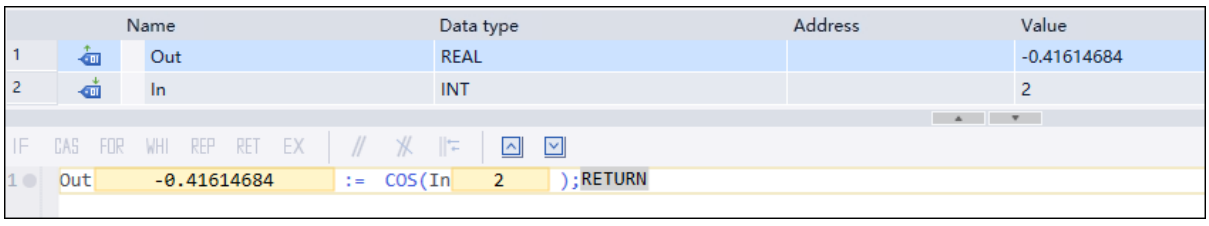
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–

- **Function**

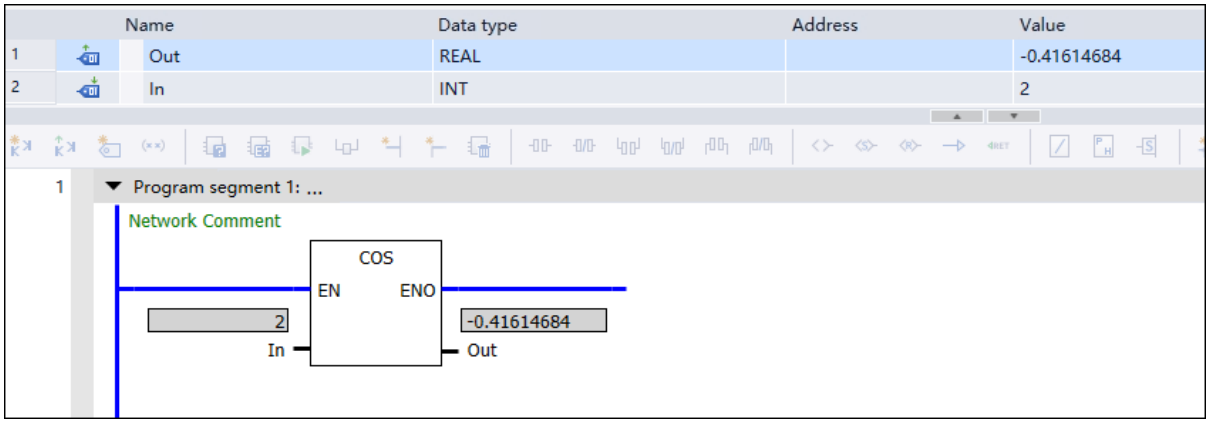
  Calculates the cosine value of the input, and outputs the result.

- **Program example**

  ST:



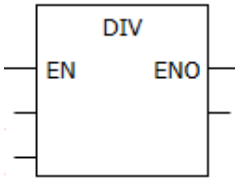
  LD:



18.1.5.21 DIV

Divides the two inputs on the left and outputs the division result on the right.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DIV	Division instruction	FC		/ ;

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Dividend	Follows data type	-	Dividend
	In2	Divisor	Follows data type	-	Divisor
Output	Out	Return value	Follows data type	-	Calculation result

Data type

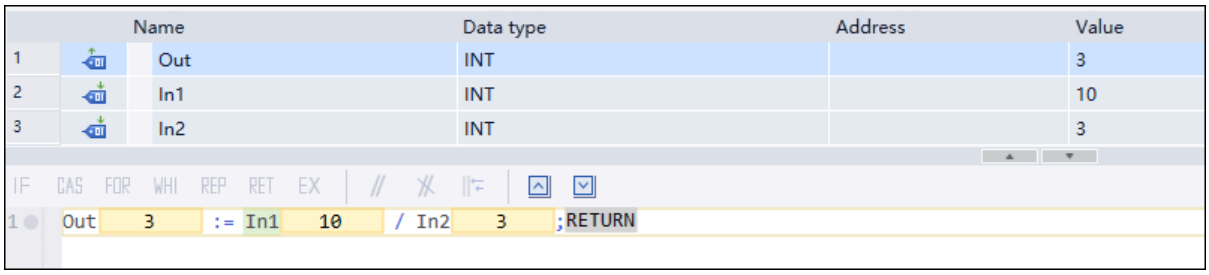
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–
In2	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Out	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–

- **Function**

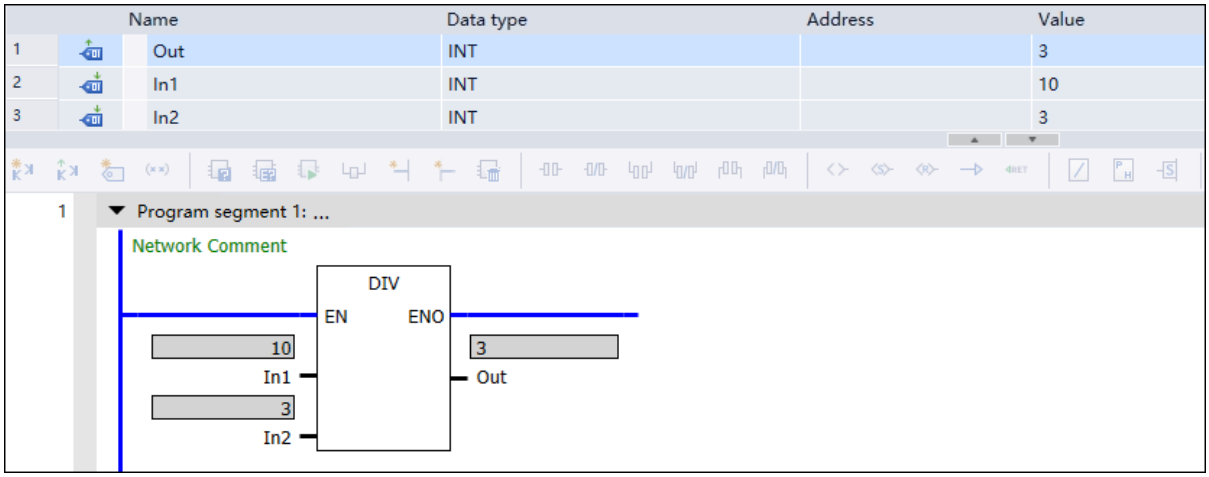
Divides the two inputs on the left and outputs the division result on the right.

- **Program example**

ST:



LD:



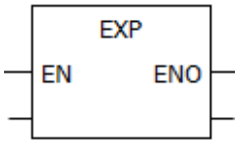
- **Precautions**

Dividing by zero produces different results depending on the target system.

18.1.5.22 EXP

Performs exponential function operation on the input value and outputs the function result.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
EXP	Exponential calculation	FC		EXP()

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Input Variable	Follows data type	-	Variable to be calculated
Output	Out	Return value	Follows data type	-	Calculation result

Data type

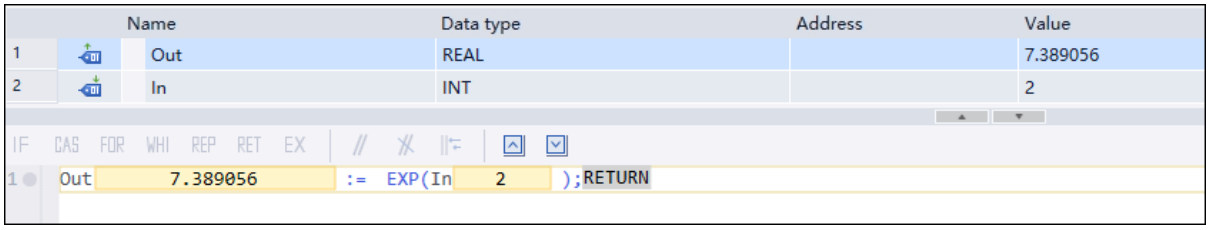
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–

- **Function**

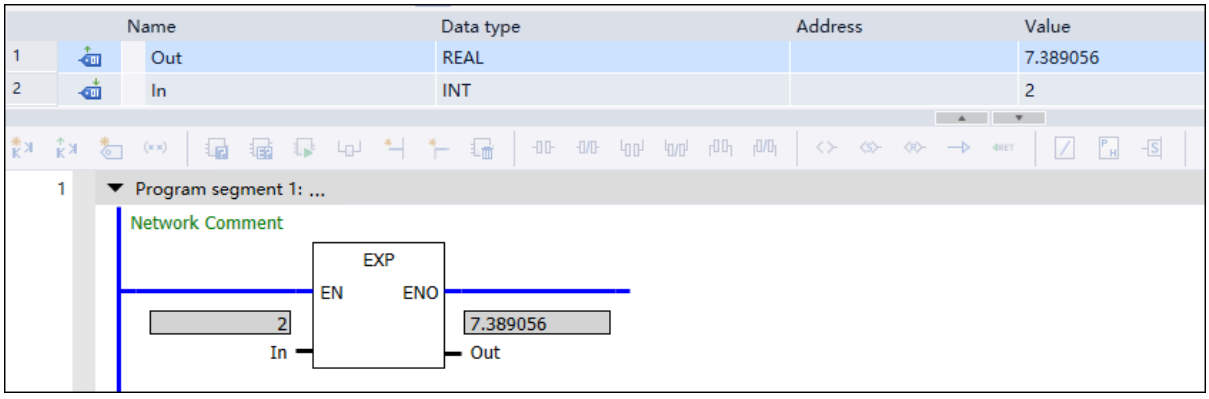
  Performs exponential function operation on the input value and outputs the function result.

- **Program example**

  ST:



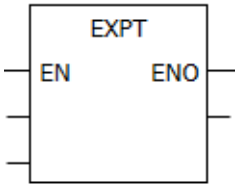
  LD:



18.1.5.23 EXPT

Outputs the power operation result, with the input variable 2 as the exponent and the input variable 1 as the base number.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
EXPT	Exponentiation	FC		EXPT(,)

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Base number	Follows data type	-	Base number
	In2	Exponent	Follows data type	-	Exponent
Output	Out	Return value	Follows data type	-	Calculation result

Data type

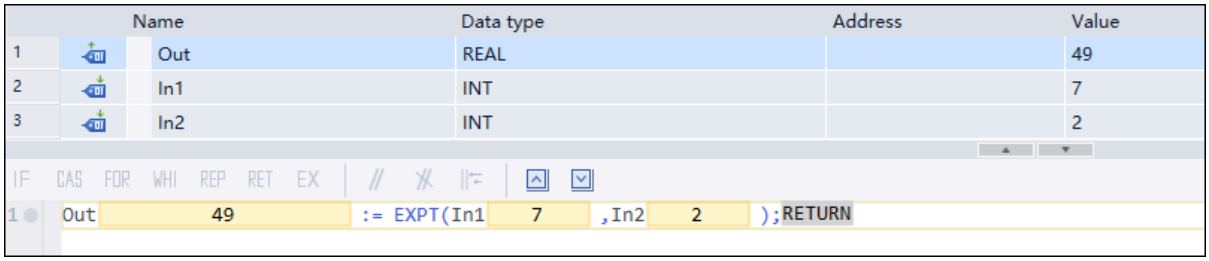
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
In2	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–

- **Function**

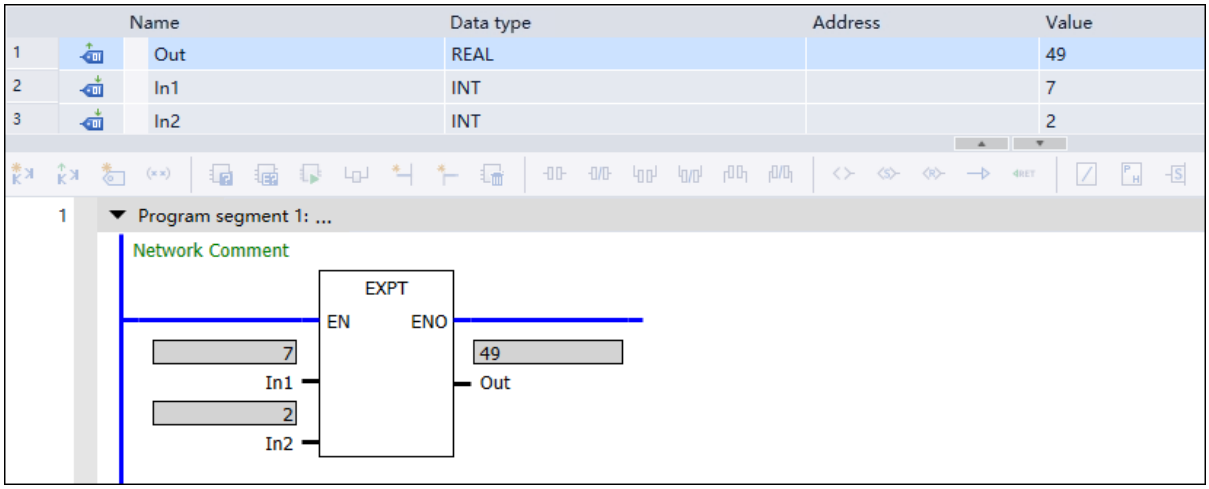
  Outputs the power operation result, with the input variable 2 as the exponent and the input variable 1 as the base number.

- **Program example**

  ST:



  LD:



- **Precautions**

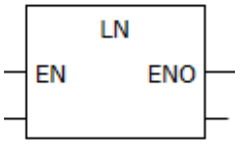
  The function result is undefined if any of the following conditions occur:

  1. The base number is negative.
  2. The base number is 0 and the exponent is less than or equal to 0.

18.1.5.24 LN

Calculates the natural logarithm of the input value, and outputs the result.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LN	Natural logarithm	FC		LN()

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Input Variable	Follows data type	-	Variable to be calculated
Output	Out	Return value	Follows data type	-	Calculation result

  Data type

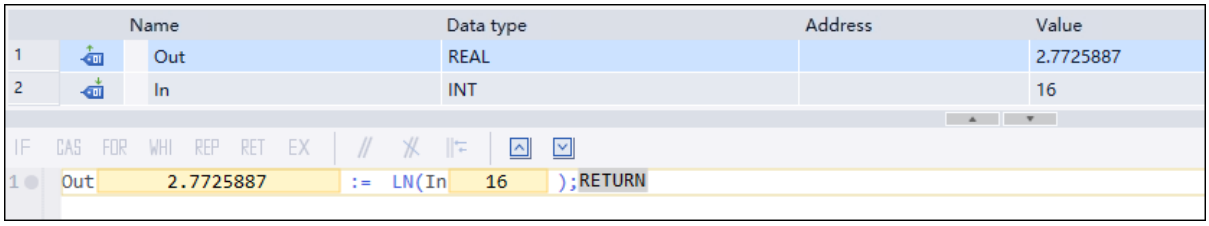
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–

- **Function**

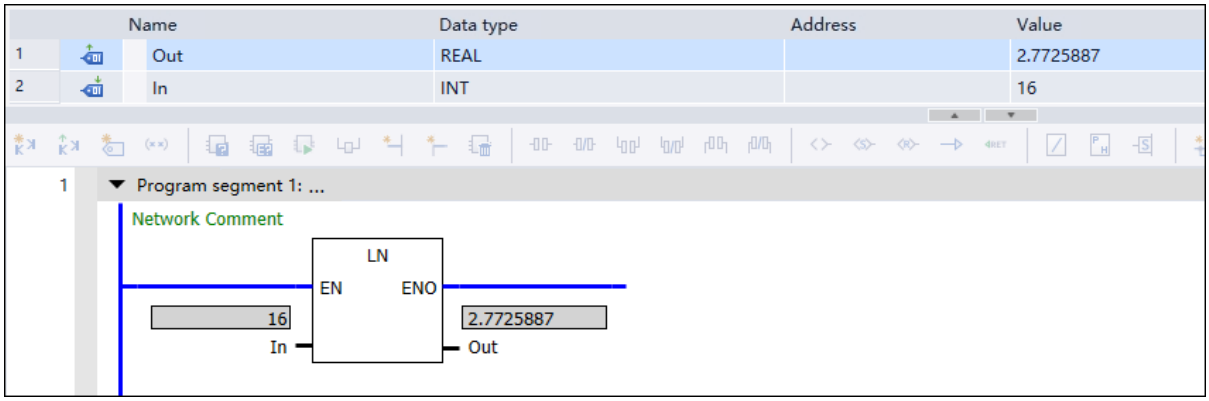
  Calculates the natural logarithm of the input value, and outputs the result.

- **Program example**

  ST:



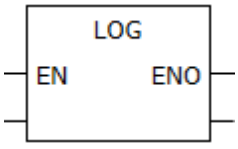
  LD:



18.1.5.25 LOG

Calculates the base-10 logarithm of the input value, and outputs the result.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LOG	Base-10 logarithm	FC		LOG()

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Input Variable	Follows data type	-	Variable to be calculated
Output	Out	Return value	Follows data type	-	Calculation result

Data type

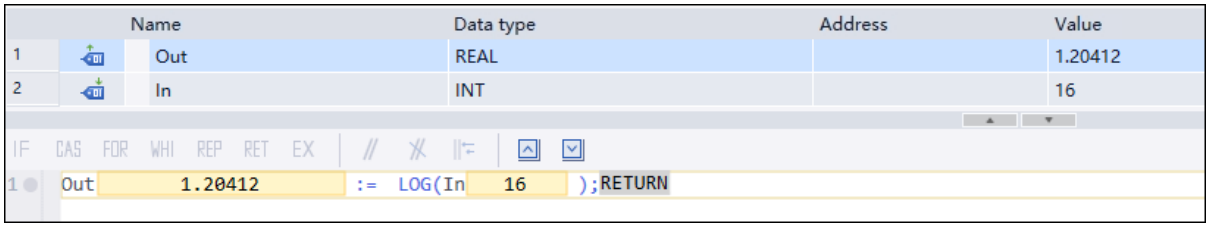
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–

- **Function**

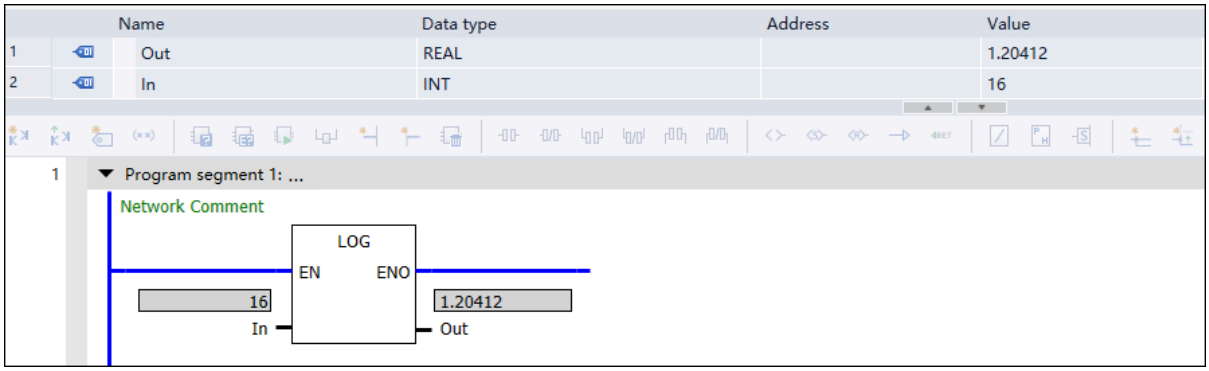
  Calculates the base-10 logarithm of the input value, and outputs the result.

- **Program example**

  ST:



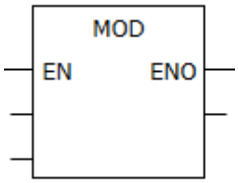
  LD:



18.1.5.26 MOD

Divides the two inputs on the left and outputs the non-negative integer remainder after division on the right.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
MOD	Modulo-division	FC		MOD ;

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Dividend	Follows data type	-	Dividend
	In2	Divisor	Follows data type	-	Divisor
Output	Out	Return value	Follows data type	-	Calculation result

  Data type

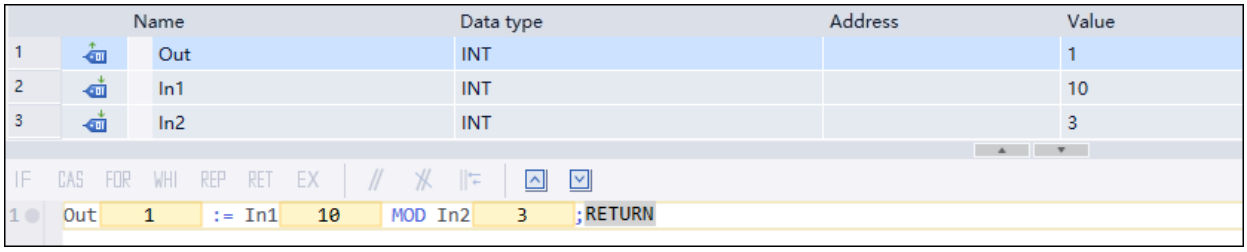
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	–	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–
In2	–	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–
Out	–	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–

- **Function**

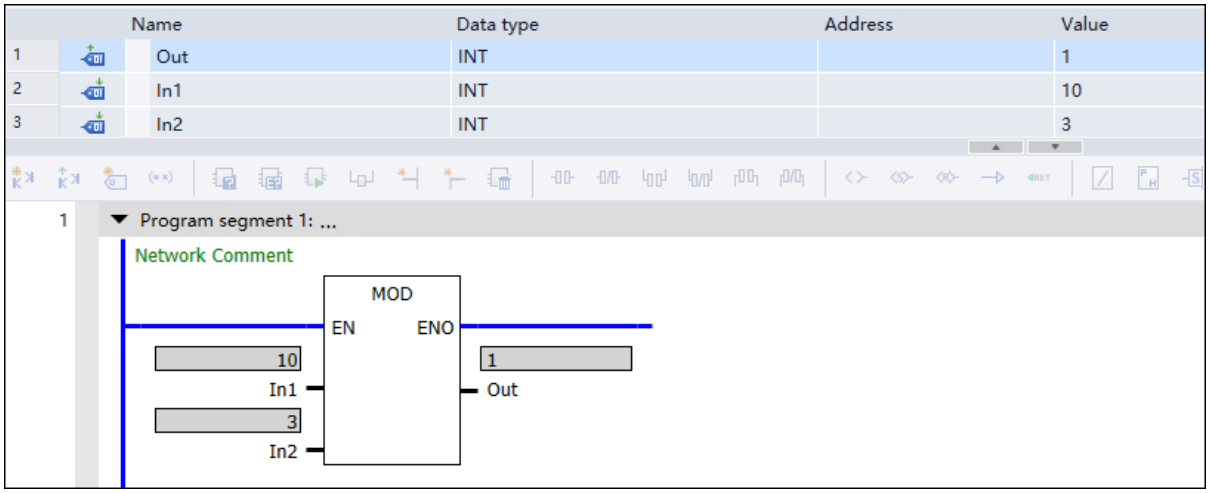
  Divides the two inputs on the left and outputs the integer remainder after division on the right.

- **Program example**

  ST:



  LD:



- **Precautions**

  Dividing by zero produces different results depending on the target system.

18.1.5.27 MOVE

Assigns the value of a variable to another variable of the same type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
MOVE	Value assignment	FC	MOVE EN ENO	:=

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Move source	Follows data type	-	Input data
Output	Out	Move destination	Follows data type	-	Output data.

  Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

- **Function description**

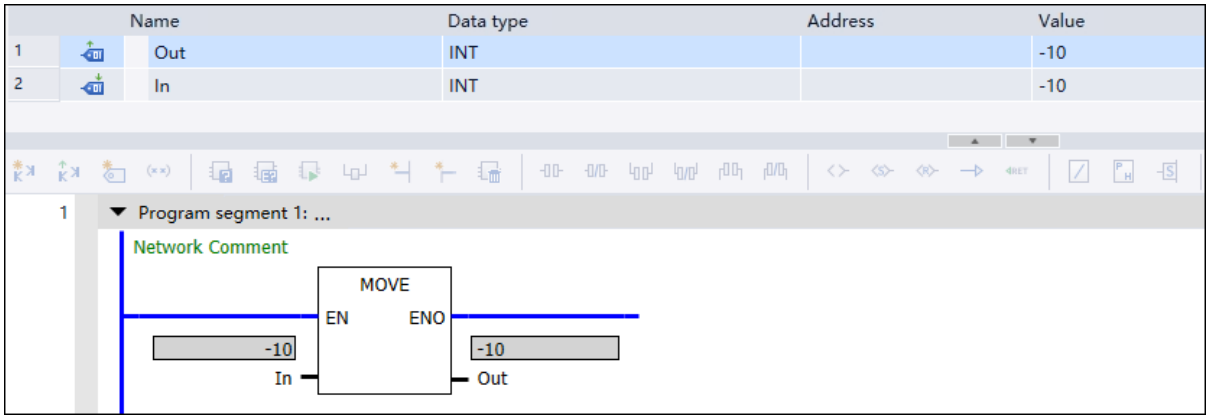
  Assigns the value of a variable to another variable of the same type.

- **Program example**

  ST:

```
Out := MOVE(In);
RETURN
```

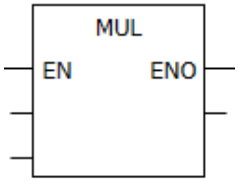
  LD:



18.1.5.28 MUL

Multiplies the two inputs on the left and outputs the multiplication result on the right.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
MUL	Multiplication instruction	FC		*

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Value to multiply	Follows data type	-	Input
	In2	Value to multiply	Follows data type	-	Input
Output	Out	Return value	Follows data type	-	Calculation result

Data type

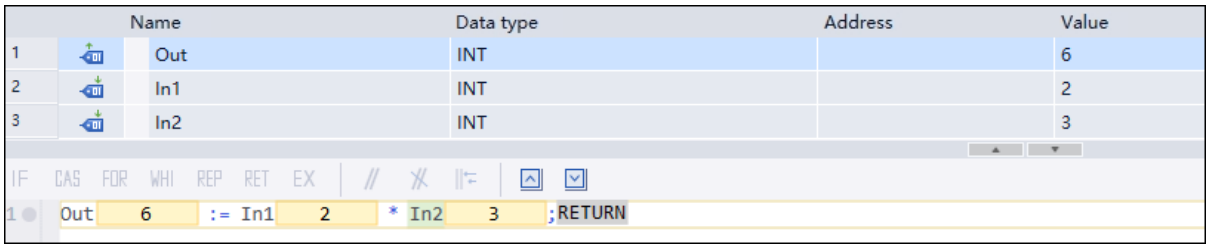
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–
In2	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Out	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–

- **Function**

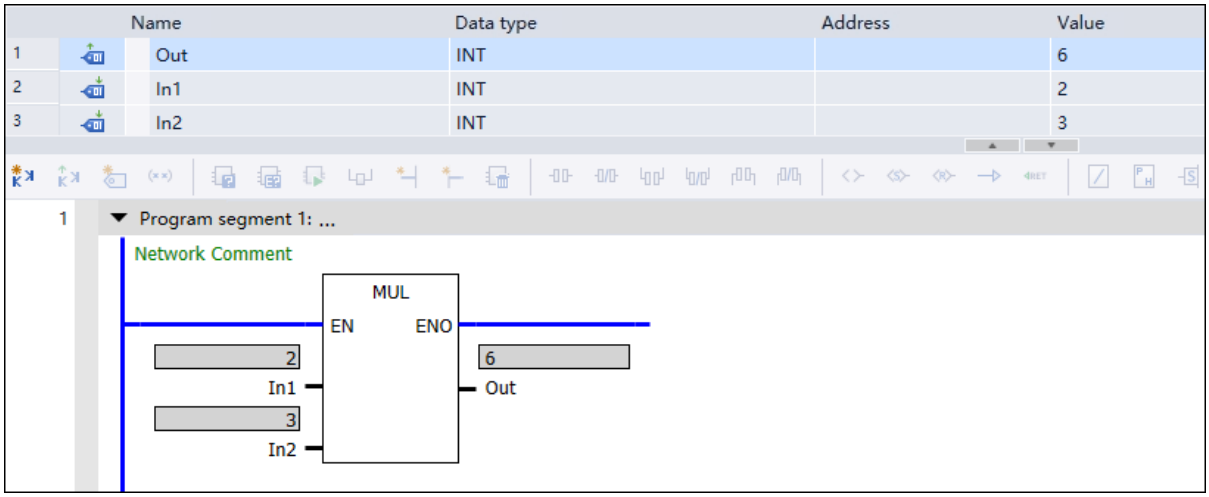
  Multiplies the two inputs on the left and outputs the multiplication result on the right.

- **Program example**

  ST:



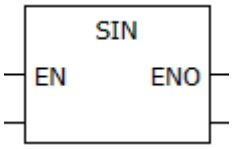
  LD:



18.1.5.29 SIN

Calculates the sine value of the input, and outputs the result.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SIN	Sine calculation	FC		SIN()

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Input Variable	Follows data type	-	Variable to be calculated
Output	Out	Return value	Follows data type	-	Calculation result

Data type

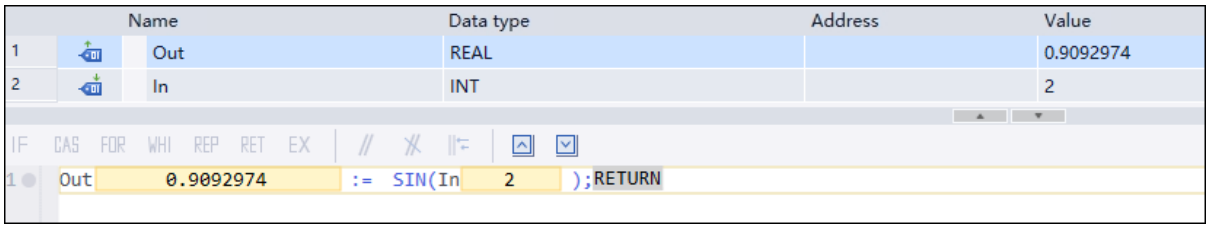
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–

- **Function**

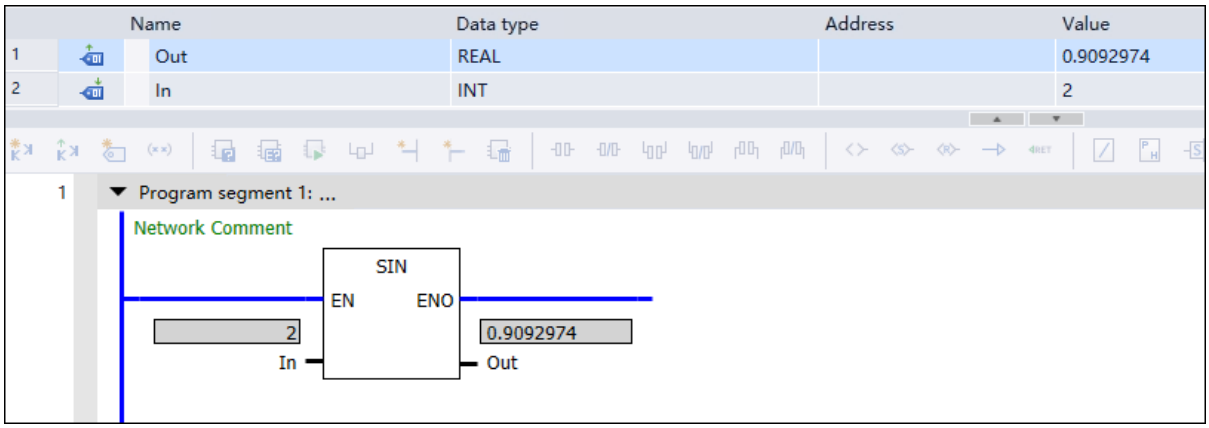
Calculates the sine value of the input, and outputs the result.

- **Program example**

ST:



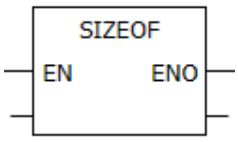
LD:



18.1.5.30 SIZEOF

Calculate the number of bytes for the input variable.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SIZEOF	Size calculation	FC		:= SIZEOF( );

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Source data	Follows data type	-	Target data
Output	Out	Data length	Follows data type	0	Data length

Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

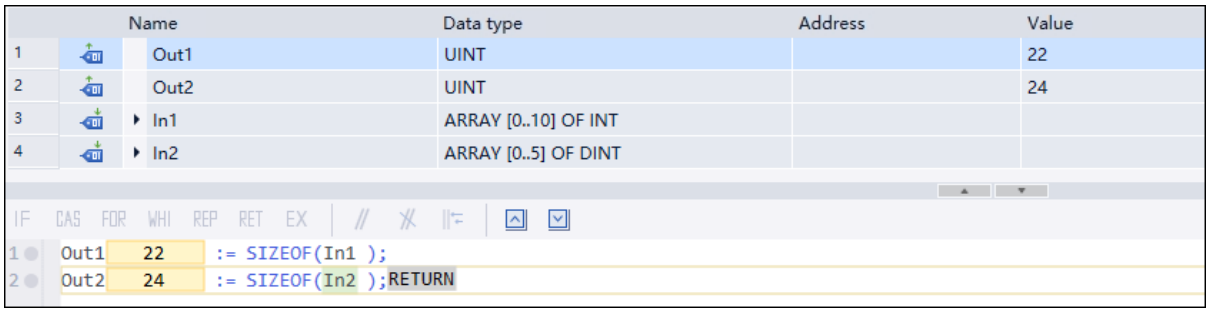
  Calculate the number of bytes for the input variable. The SIZEOF operator always produces an unsigned value. The type of the returned variable is adapted to the length detection of the input variable.

- **Program example**

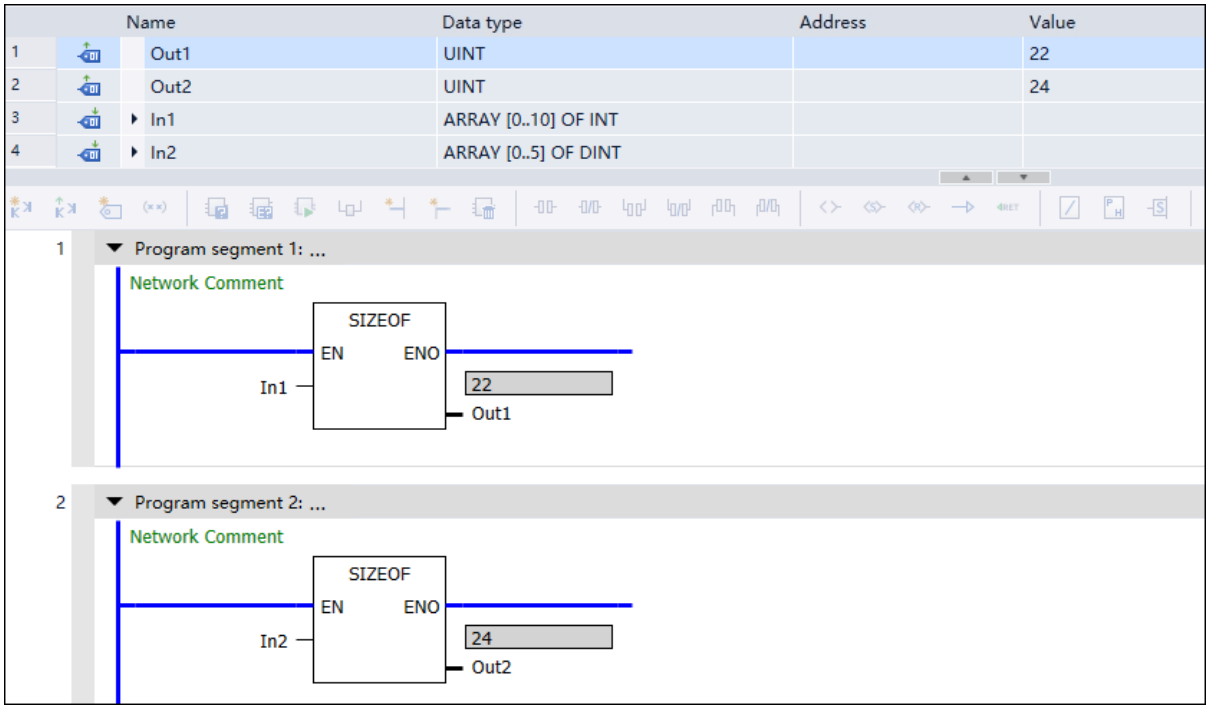
  Example 1 defines an INT array with 11 elements, resulting in a size of 22.

  Example 2 defines a DINT array with 6 elements, resulting in a size of 24.

  ST:



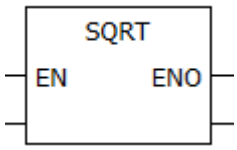
  LD:



18.1.5.31 SQRT

Calculates the square root of the input value, and outputs the result.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SQRT	Square root calculation	FC		SQRT()

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Input Variable	Follows data type	-	Input variable
Output	Out	Return value	Follows data type	-	Calculation result

Data type

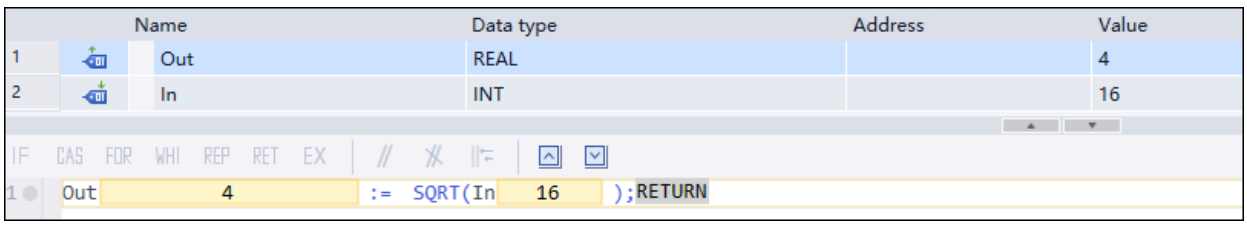
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–

- **Function**

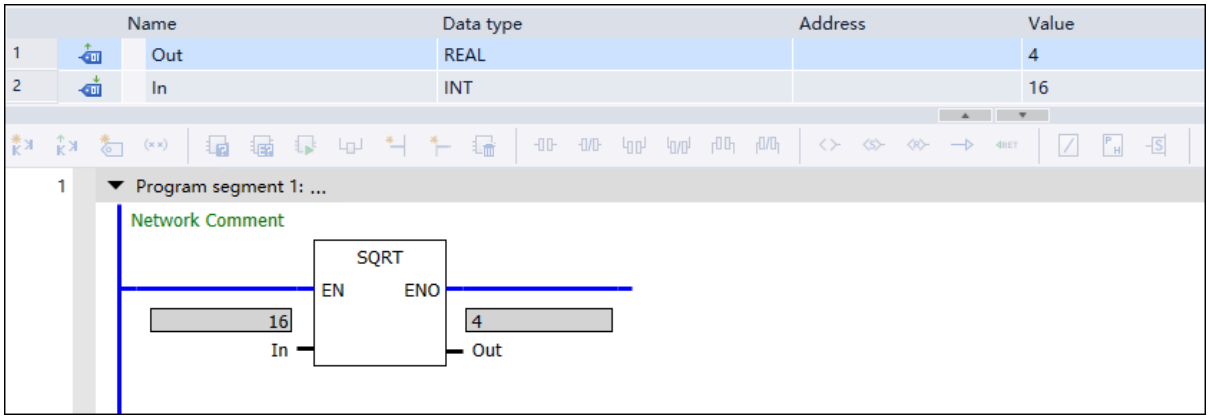
  Calculates the square root of the input value, and outputs the result.

- **Program example**

  ST:



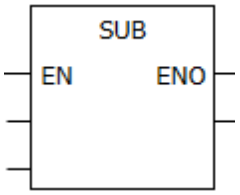
  LD:



18.1.5.32 SUB

Subtracts the two inputs on the left and outputs the subtraction result on the right.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SUB	Subtraction instruction	FC		- ;

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Minuend	Follows data type	-	Minuend
	In2	Subtrahend	Follows data type	-	Subtrahend
Output	Out	Return value	Follows data type	-	Calculation result

Data type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–
In2	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–

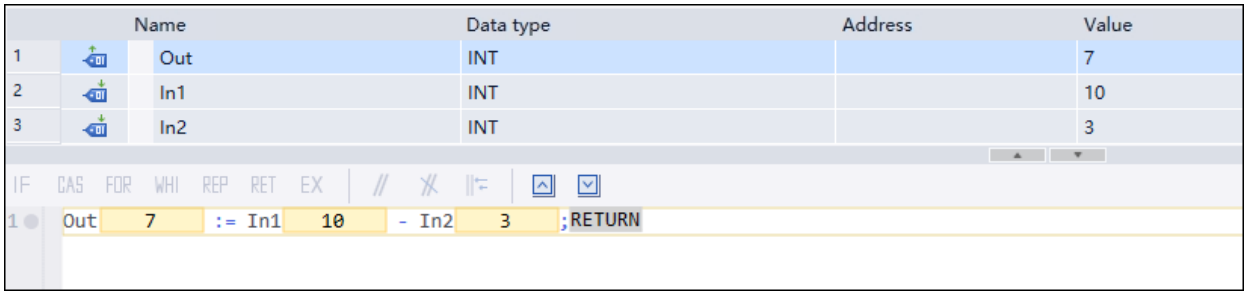
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–

- **Function**

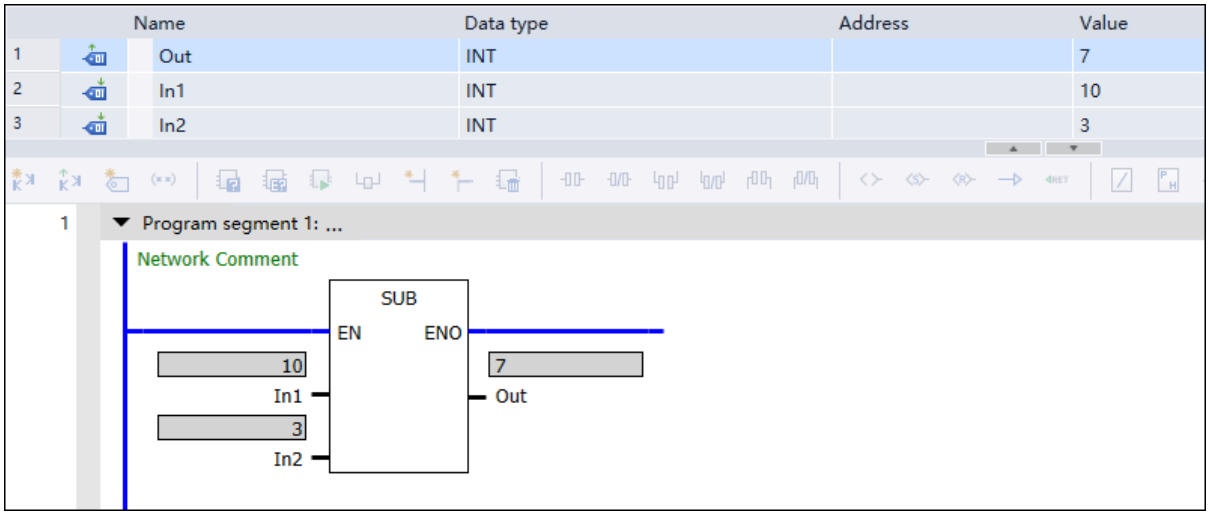
  Subtracts subtrahend "In2" from minuend "In1", and outputs the subtraction result on the right.

- **Program example**

  ST:



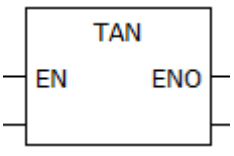
  LD:



18.1.5.33 TAN

Calculates the tangent value of the input, and outputs the result.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
TAN	Tangent calculation	FC		TAN()

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Input Variable	Follows data type	-	Variable to be calculated
Output	Out	Return value	Follows data type	-	Calculation result

Data type

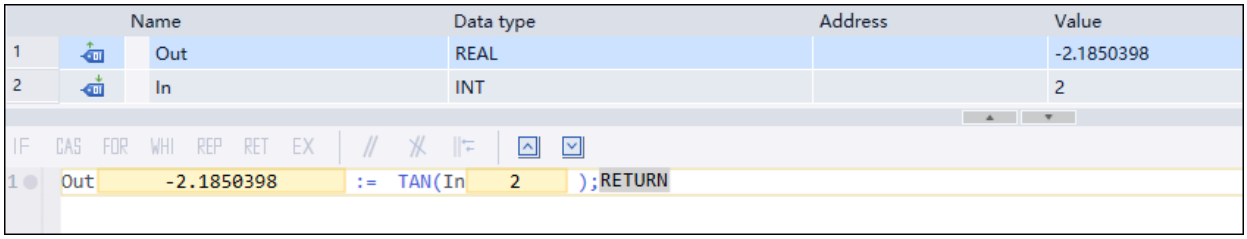
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–

- **Function**

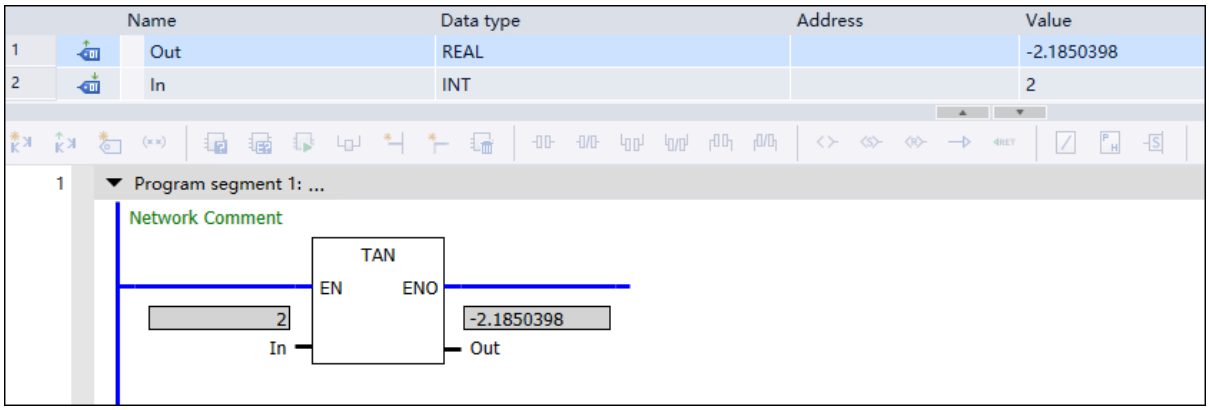
  Calculates the tangent value of the input, and outputs the result.

- **Program example**

  ST:



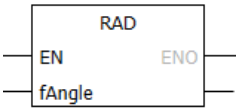
  LD:



18.1.5.34 RAD

Converts floating-point degrees to radians.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
RAD	Degrees to radians	FC		:= RAD(fAngle :=);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Angle	Input degree	Follows data type	0	Input degree
Output	Out	Output radian	Follows data type	0	Output radians

Variable types

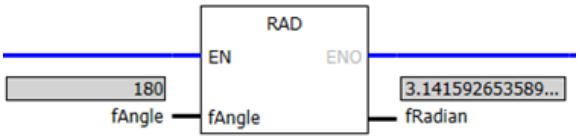
	Boolean	Bit String				Integer								Real Number		Time, duration, date, and text string				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Angle	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–

- **Function description**

The calculation formula for converting floating-point degrees to radians is: radians = degrees × π/180.

- **Program example**

	LD	ST
Define variables	VAR fAngle: LREAL; fRadian: LREAL; END_VAR	

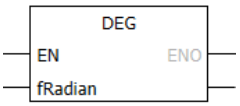
	LD	ST
Program		
Operation result		

Name	Type	Address	Value
fAngle	LREAL		180
fRadian	LREAL		3.141592653589793

18.1.5.35 DEG

Converts floating-point radians to degrees.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
DEG	Radians to degrees	FC		:= DEG(fRadian :=);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	fRadian	Input radian	Follows data type	0	Input radian
Output	Out	Output degree	Follows data type	0	Output degree

Variable types

	Boolean	Bit String				Integer								Real Number		Time, duration, date, and text string				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
fRadian	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–

- **Function description**

The calculation formula for converting floating-point radians to degrees is: degrees = radians × π/180.

- **Program example**

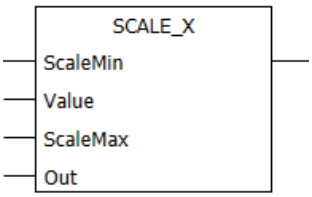
	LD	ST
Define variables	VAR fRadian: LREAL; fAngle: LREAL; END_VAR	
Program		
Operation result		

Name	Type	Address	Value
fRadian	LREAL		3.141592653589793
fAngle	LREAL		180

18.1.5.36 SCALE_X

SCALE_X is a linear scaling instruction.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SCALE_X	Scaling instruction	FC		:= SCALE_X(ScaleMin := , Value := , ScaleMax := , Out :=);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	ScaleMin	Lower limit of standard value	Follows data type	-	Lower limit of standard value
	Value	Input value to scale	Follows data type	-	Input value to scale
	ScaleMax	Upper limit of standard value	Follows data type	-	Upper limit of standard value
	Out	Scaled output value	Follows data type	-	Scaled output value

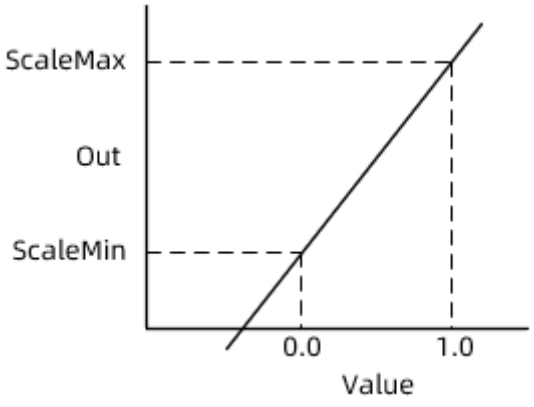
Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	SCALE_X	Result	Follows data type	-	TRUE: Operation succeeded. FALSE: Operation failed.

Variable types

	Boolean	Bit String				Integer								Real Number		Time, duration, date, and text string				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
ScaleMin	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Value	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–
ScaleMax	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Out	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
SCALE_X	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function description**

  The standardized real input "Value" (0.0 <= "Value" <= 1.0) is scaled according to the data type and range specified by parameters "ScaleMin" and "ScaleMax": Out := Value × (ScaleMax - ScaleMin) + ScaleMin.

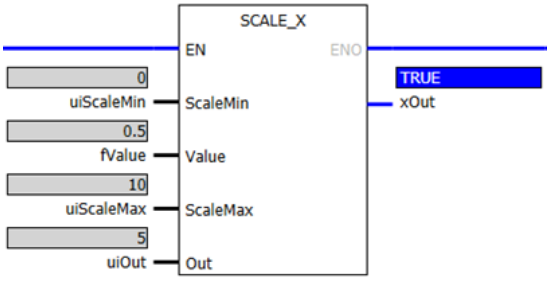


- **Program example**

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    uiScaleMin: UINT;
    fValue: LREAL;
    uiScaleMax: UINT;
    uiOut: UINT;
    xOut: BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
xOut TRUE := SCALE_X( ScaleMin := uiScaleMin 0 ,
                      Value := fValue 0.5 ,
                      ScaleMax := uiScaleMax 10 ,
                      Out := uiOut 5 );
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

Name	Type	Address	Value
uiScaleMin	UINT		0
fValue	LREAL		0.5
uiScaleMax	UINT		10
uiOut	UINT		5
xOut	BOOL		TRUE

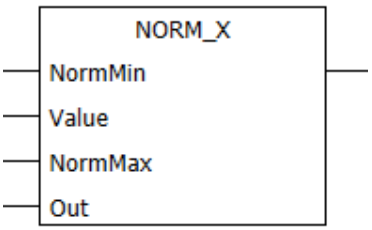
</td></tr>
</table>

- **Precautions**
  - "ScaleMin", "ScaleMax", and "Out" must have the same data type.
  - The range of "Value" is [0.0, 1.0]. If "Value" is less than 0.0 or greater than 1.0, the linear scaling instruction will generate "Out" values less than "ScaleMin" or greater than "ScaleMax". If these values are within the range of the "Out" data type, "SCALE_X" returns TRUE. If these values are beyond the allowable range, the "Out" parameter is set to an intermediate value equal to the least significant part of the scaled real number before its conversion to the "Out" data type. In such cases, "SCALE_X" returns FALSE.
  - Parameter "ScaleMin" must be less than "ScaleMax"; otherwise, "Out" is filled with the least significant part of the real number "Value", and "SCALE_X" returns FALSE.
  - If the parameter "Value" is +/-INF or +/-NaN, the input "Value" is written to "Out", and "SCALE_X" returns FALSE.

18.1.5.37 NORM_X

NORM_X is a normalization instruction.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
NORM_X	Normalization instruction	FC		:= NORM_X(NormMin := , Value := , NormMax := , Out :=);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	NormMin	Lower limit of standard value	Follows data type	-	Lower limit of standard value
	Value	Value to be normalized	Follows data type	-	Value to be normalized
	NormMax	Upper limit of standard value	Follows data type	-	Upper limit of standard value
	Out	Normalized value	Follows data type	-	Normalized value
Output	NORM_X	Result	Follows data type	-	TRUE: Operation succeeded. FALSE: Operation failed.

Variable types

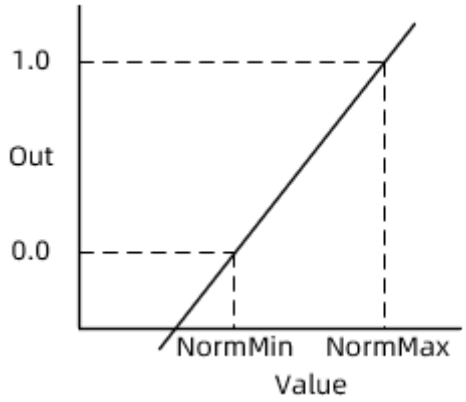
	Boolean	Bit String				Integer								Real Number		Time, duration, date, and text string				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
NormMin	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Value	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time, duration, date, and text string				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
NormMax	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–
NORM_X	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function description**

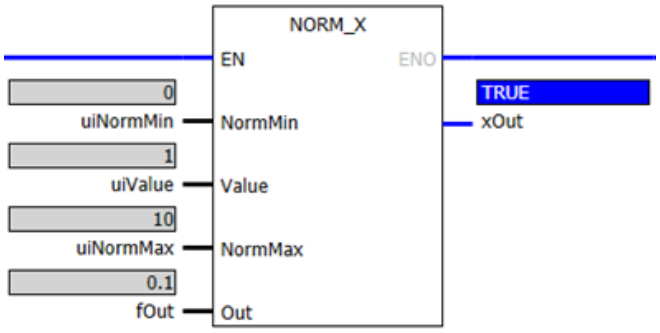
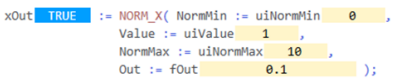
  The real input "Value" is normalized based on the data type and range specified by "NormMin" and "NormMax". "Out" equals (Value - NormMin) / (NormMax - NormMin), where (0.0 <= "Out" <= 1.0).

  The following figure shows how to normalize a value.



- **Program example**

	LD	ST
Define variables	VAR uiNormMin: UINT; uiValue: UINT; uiNormMax: UINT; fOut: LREAL; xOut: BOOL; END_VAR	

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>
</td><td>
</td></tr>
<tr><td>Operation result</td><td colspan="2">
</td></tr>
</table>

- **Precautions**
  - "NormMin", "NormMax", and "Value" must have the same data type.
  - The range of "Value" is ["NormMin", "NormMax"]. If "Value" is less than "NormMin" or greater than "NormMax", the linear scaling instruction will generate "Out" values less than 0.0 or greater than 1.0. If these values are beyond the allowable range of the "Out" data type, "Out" is set to an intermediate value equal to the least significant part of the normalized real number before its conversion to the "Out" data type. In such cases, "NORM_X" returns TRUE.
  - "NormMin" should be less than "NormMax", otherwise, "Value" is extended to fill double-word size, and "NORM_X" returns FALSE.
  - If the parameter "Value" is +/-INF or +/-NaN, the input "Value" is written to "Out", and "NORM_X" returns FALSE.

18.1.6 Selection Operations

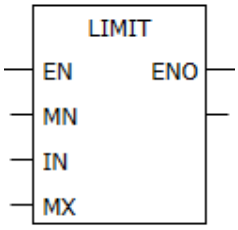
18.1.6.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.1.6.2 LIMIT	FC	Limit value instruction	-	-
18.1.6.3 MAX	FC	Maximum value instruction		
18.1.6.4 MIN	FC	Minimum value instruction		
18.1.6.5 MUX	FC	Select one from multiple		
18.1.6.6 SEL	FC	Binary selection		

18.1.6.2 LIMIT

Limits the input value between the specified maximum and minimum values, and outputs the limited value.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LIMIT	Limit value instruction	FC		LIMIT(, ,)

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	MN	Input signal	Follows data type	-	Minimum value
	IN	Input signal	Follows data type	-	The value to be limited
	MX	Input signal	Follows data type	-	Maximum value
Output	Out	Return value	Follows data type	-	Calculation result

Data type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
MN	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
IN	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
MX	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

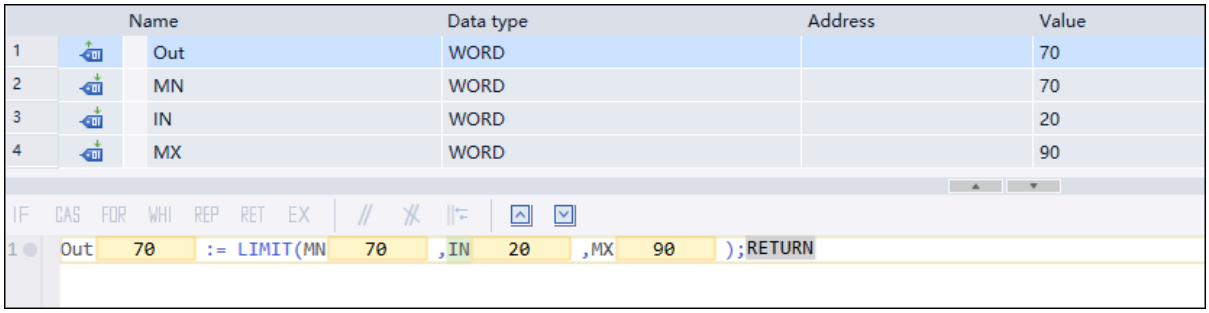
- **Function description**

"MX" is the upper limit of the result, and "MN" is the lower limit. If "IN" exceeds the upper limit "MX", the output result is "MX". If "IN" is less than the lower limit "MN", the output result is "MN". If "IN" is within the range of "MN" and "MX", the output value is "IN".

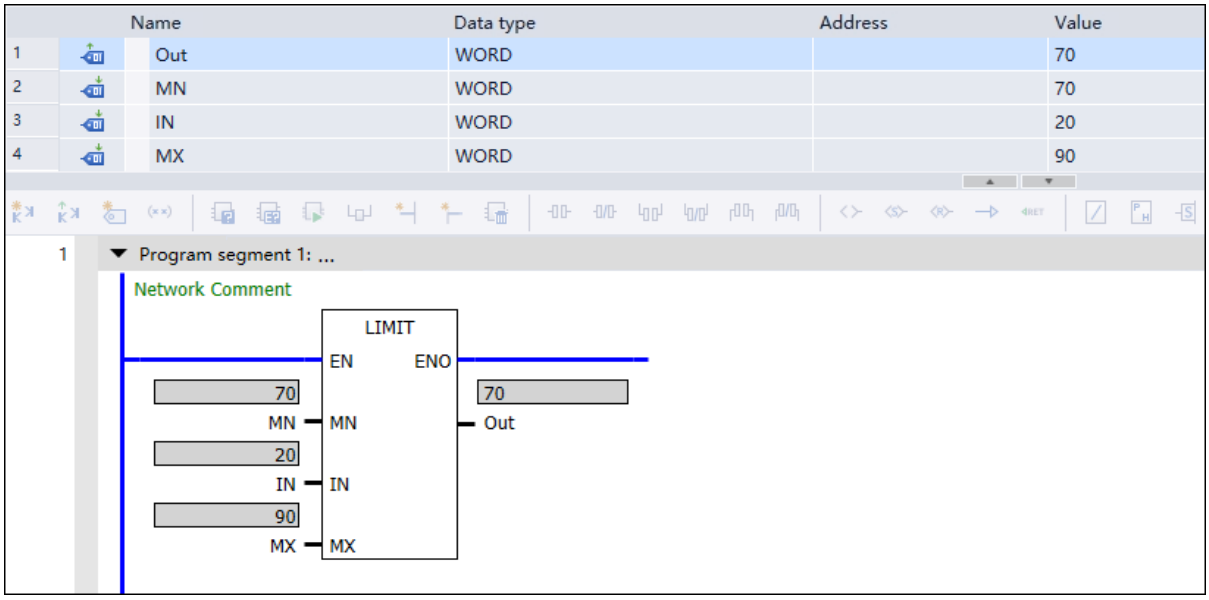
- **Program example**

  In the following example, "MN" is set to 70, "MX" to 90, and "IN" to 20; then the result "Out" is 70.

  ST:



  LD:



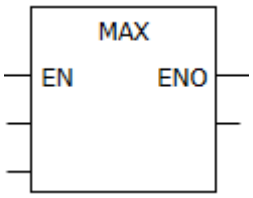
- **Precautions**

  The data types of "Out", "MX", "IN", and "MN" must be the same.

18.1.6.3 MAX

Outputs the maximum of the two input values on the right.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
MAX	Maximum value instruction	FC		MAX(,)

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Comparand	Follows data type	-	Comparand 1
	In2	Comparand	Follows data type	-	Comparand 2
Output	Out	Return value	Follows data type	-	The larger number of the two comparands

Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

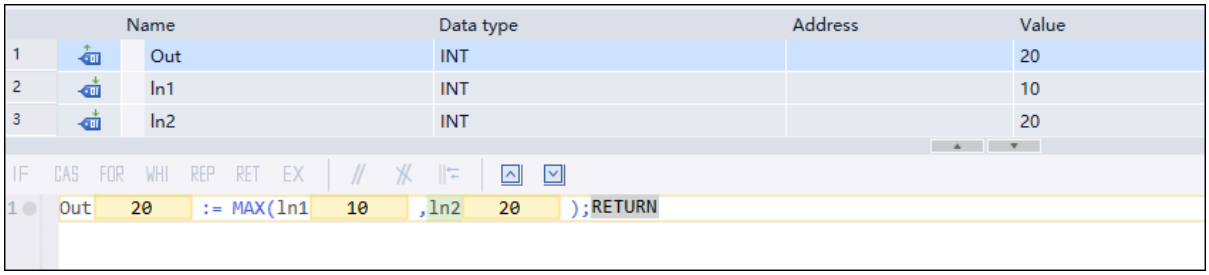
- **Function description**

  Outputs the maximum of the two input values on the right.

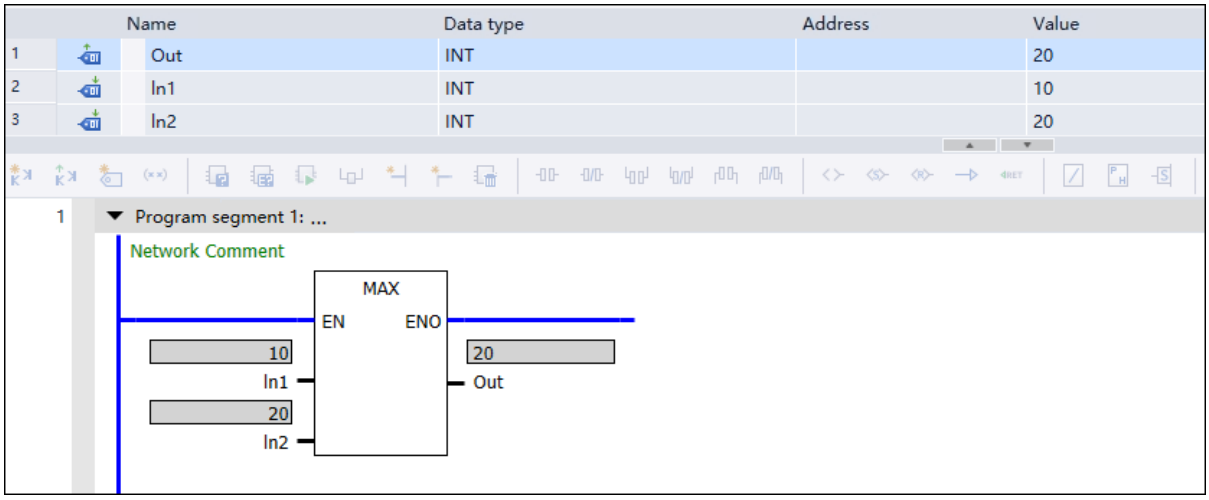
- **Program example**

  The following example outputs the maximum of the two input values.

  ST:



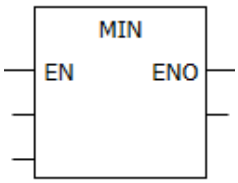
  LD:



18.1.6.4 MIN

Outputs the minimum of the two input values on the right.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
MIN	Minimum value instruction	FC		MIN(,)

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Comparand	Follows data type	-	Comparand 1
	In2	Comparand	Follows data type	-	Comparand 2
Output	Out	Return value	Follows data type	-	The smaller number of the two comparands

Data Type

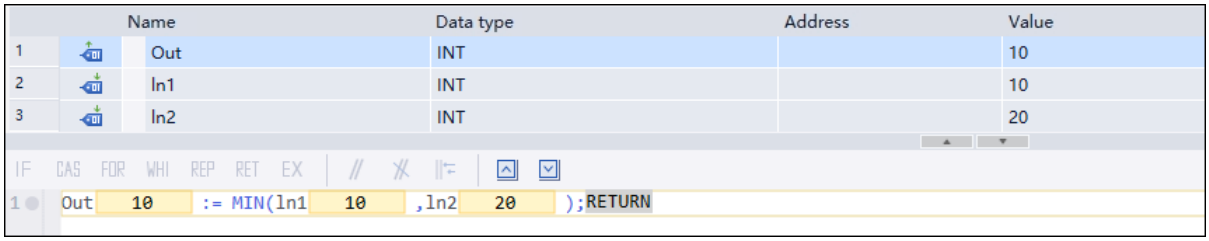
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
In2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

- **Function description**

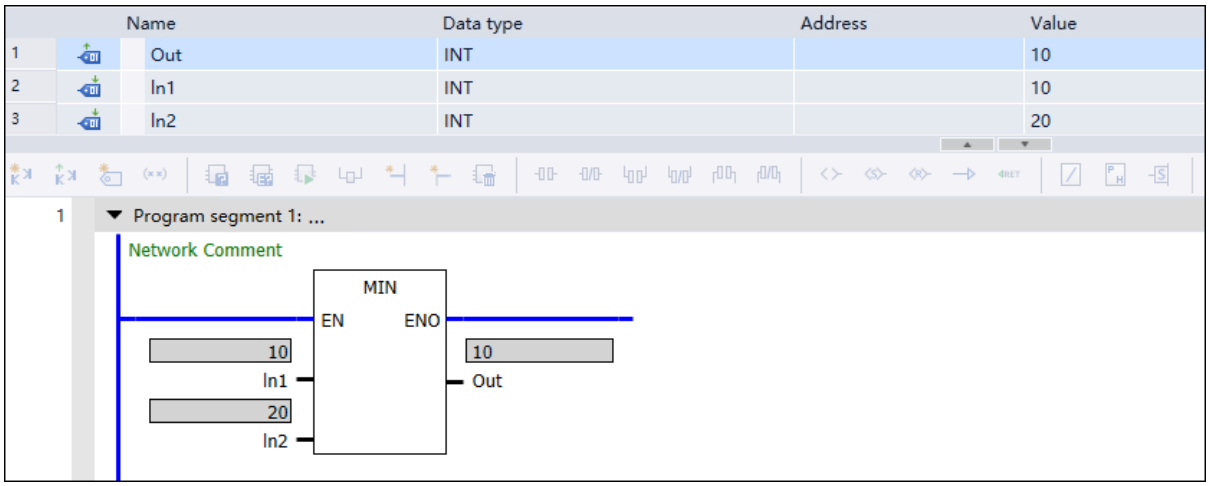
  Outputs the minimum of the two input values on the right.

- **Program example**

  ST:



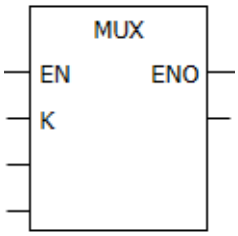
  LD:



18.1.6.5 MUX

Selects the K-th value from the input values, where K = 0 corresponds to the first value. If K is greater than the number of compared values or if K is negative, the last value (InN) is output.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
MUX	Select one from multiple	FC		MUX(, ,)

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	K	Selector bit	Follows data type	-	Selector bit
	In0 to InN	Comparand	Follows data type	-	Data source for selection
Output	Out	Return value	Follows data type	-	Selection result

  Data Type

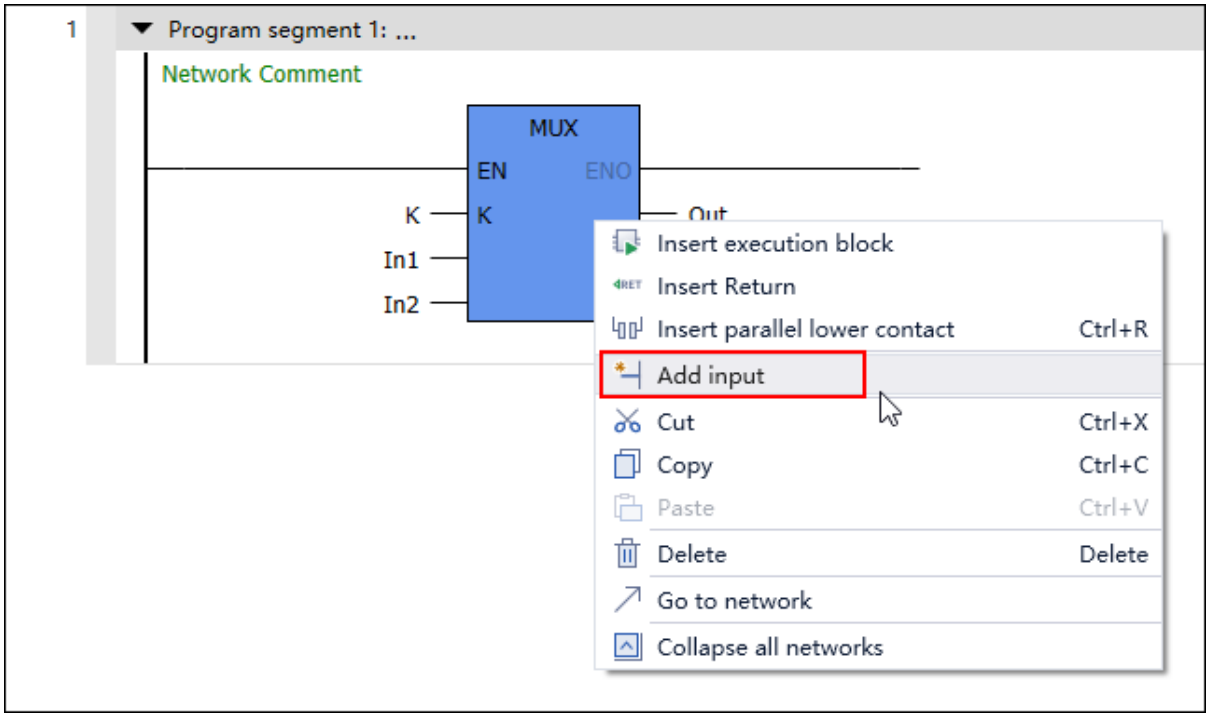
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
K	–	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–
In0 to InN	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

- **Function description**

  Selects the K-th value from the input values, where K = 0 corresponds to the first value. If K is greater than the number of compared values or if K is negative, the last value (InN) is output.

  The default number of compared values for the MUX instruction is 2; additional inputs are required if there are more than 2 compared values.

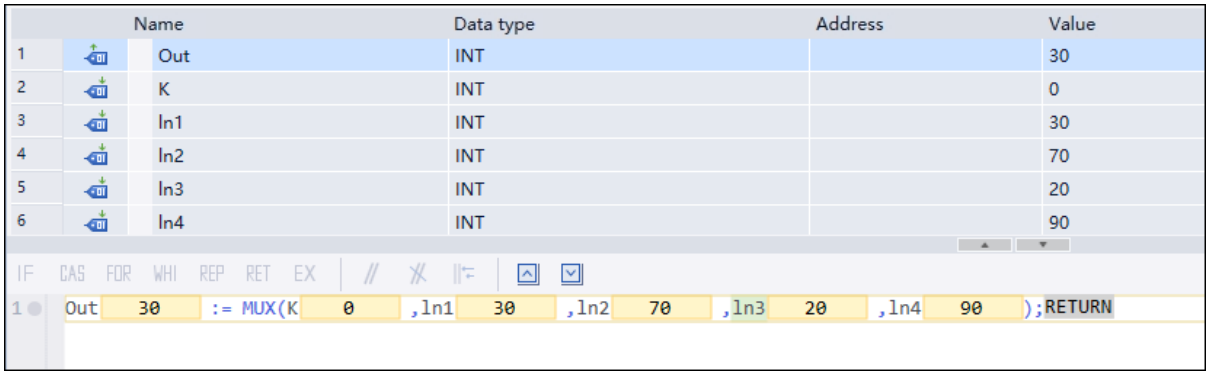
  The following figure shows how to add an input in LD programming.



- **Program example**

  Input a set of data: 30, 70, 20, and 90. When K is set to 0, the output is 30.

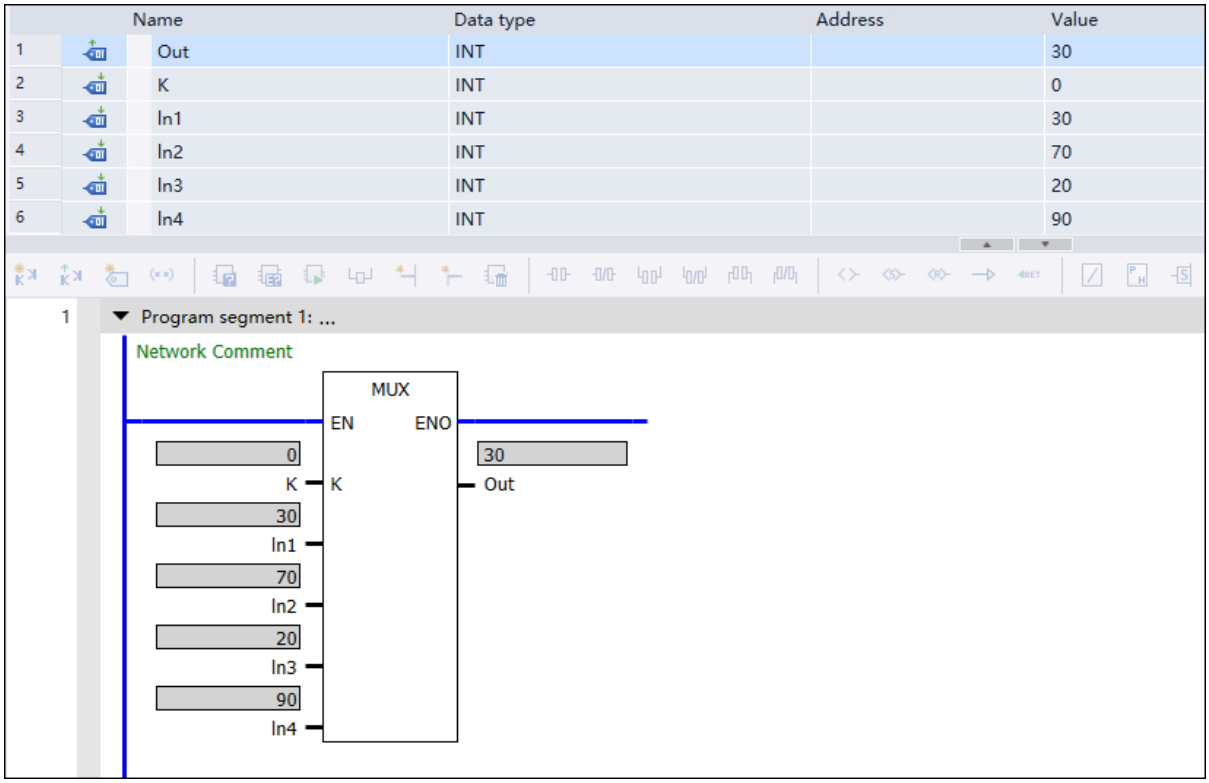
  ST:



	Name	Data type	Address	Value
1	Out	INT		30
2	K	INT		0
3	In1	INT		30
4	In2	INT		70
5	In3	INT		20
6	In4	INT		90

```
1  Out := MUX(K := 0, In1 := 30, In2 := 70, In3 := 20, In4 := 90);RETURN
```

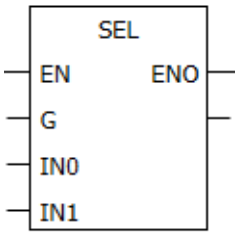
  LD:



18.1.6.6 SEL

Selects one of two input values based on "G".

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SEL	Binary selection	FC		SEL(, ,);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	G	Input signal	Follows data type	-	Outputs IN0 when G = FALSE; outputs IN1 when G = TRUE.
	IN0	Input signal	Follows data type	-	Output when G = FALSE
	IN1	Input signal	Follows data type	-	Output when G = TRUE

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Return value	Follows data type	-	Output result

Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
G	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IN0	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
IN1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

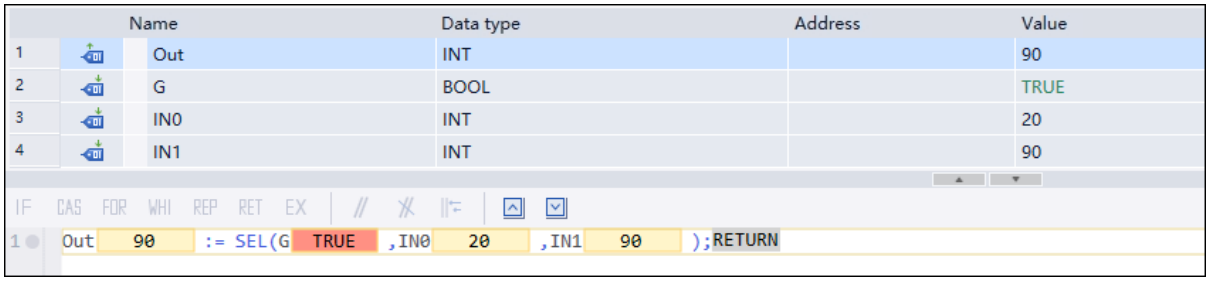
- **Function description**

  Outputs IN0 when G = FALSE; outputs IN1 when G = TRUE.

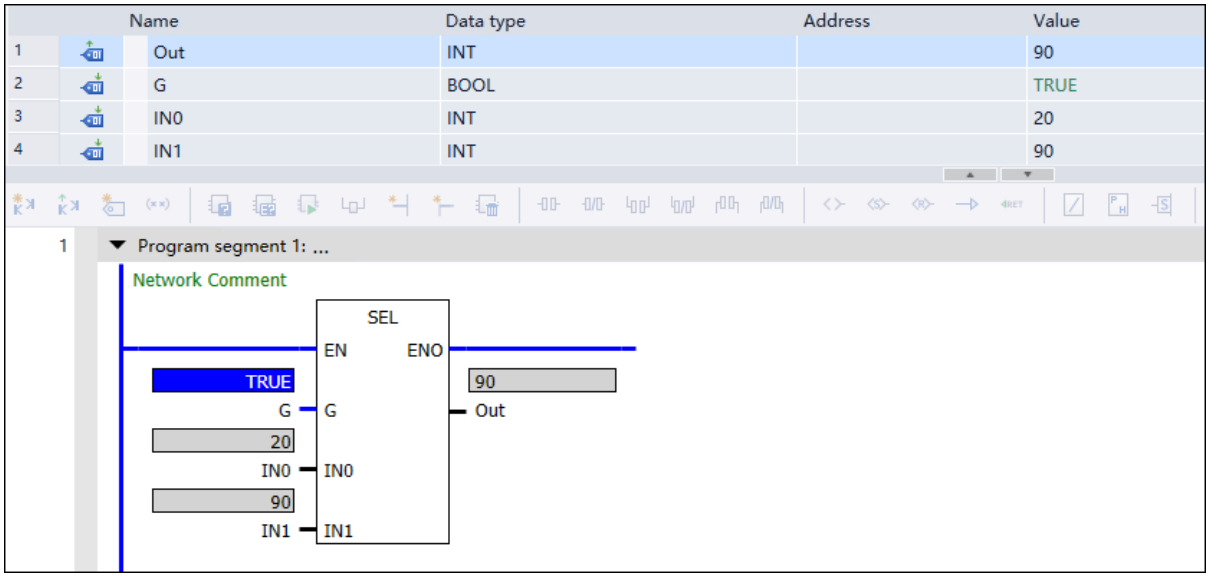
- **Program example**

  Set the input values IN0 and IN1 to 20 and 90, and set G = TRUE. Then, 90 (value of IN1) is output.

  ST:



  LD:



- **Precautions**

  When G = TRUE, the expression before IN0 will not be calculated. When G = FALSE, the expression before IN1 will not be calculated.

18.1.7 Shift Operations

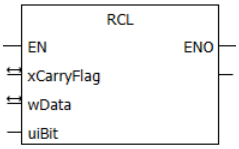
18.1.7.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.1.7.2 RCL	FC	Rotation left with carry	Util	Library version 1.0.0.0 and later
18.1.7.3 RCR	FC	Rotation right with carry		Library version 1.0.0.0 and later
18.1.7.4 ROLE	FC	Rotate left shift instruction	-	-
18.1.7.5 ROR	FC	Rotate right shift instruction		
18.1.7.6 SHL	FC	Left shift instruction		
18.1.7.7 SHR	FC	Right shift instruction		

18.1.7.2 RCL

Rotates the operand to the left. Bits that are shifted out from the left are directly appended to the rightmost bit. When a carry occurs, the carry flag is set to ON.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
RCL	Rotation left with carry	FC		:= RCL(xCarryFlag := , wData := , uiBit :=);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input/Output	xCarryFlag	Carry flag	Follows data type	-	Carry flag
	wData	Data Source	Follows data type	-	Data source
Input	uiBit	Number of bits to rotate	Follows data type	-	Number of bits to rotate
Output	OUT	Return value	Follows data type	-	Execution result

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
xCarryFlag	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
wData	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
uiBit	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

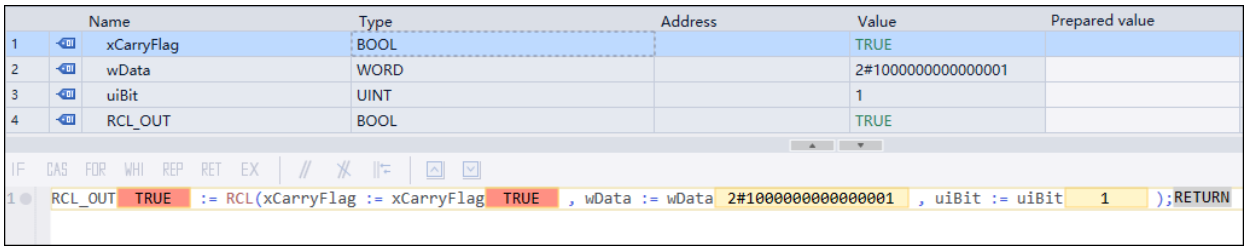
- **Function**

Rotates the operand to the left. Bits that are shifted out from the left are directly appended to the rightmost bit. When a carry occurs, the carry flag is set to ON.

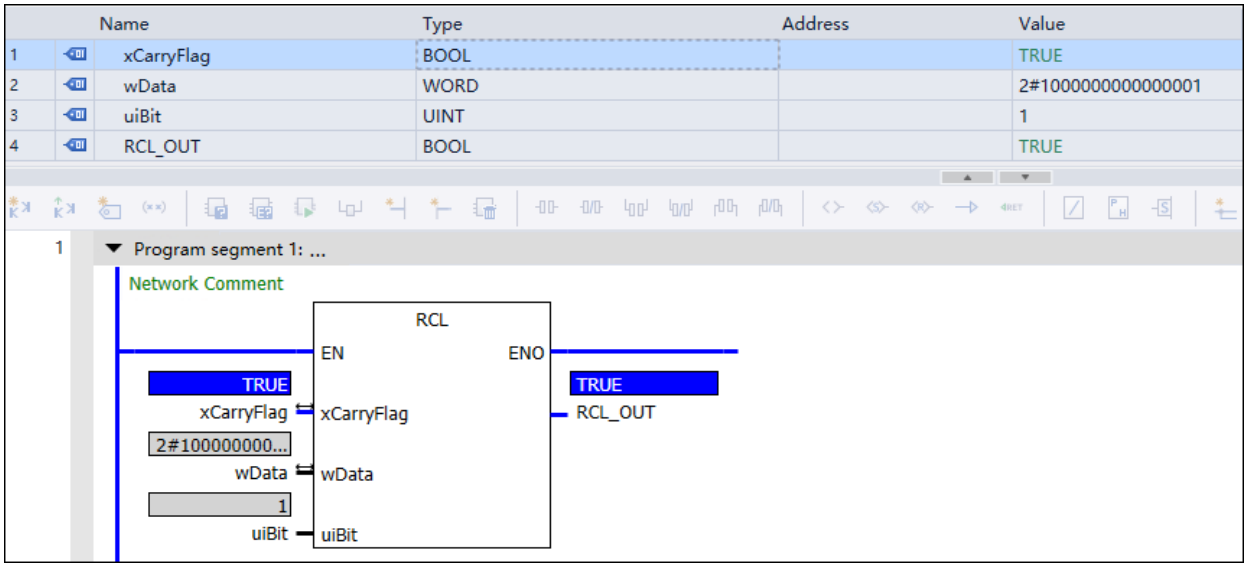
- **Program example**

  When the input source "wData" is set to 2#1100000000000000 and "uiBit" to 1, upon triggering the first time, the output signal "OUT" becomes 2#1000000000000001. The valid data shifts left by 1 bit, and the carry flag is set to TRUE.

  ST:



  LD:



- **Precautions**
  - If "uiBit" is less than 1 or greater than 16, the instruction execution fails, and the return value is FALSE, with the output variable remaining at its initial value.
  - The instruction returns TRUE upon successful execution, otherwise returns FALSE.

18.1.7.3 RCR

Rotates the operand to the right. Bits that are shifted out from the right are directly appended to the leftmost bit. When a carry occurs, the carry flag is set to ON.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
RCR	Rotation right with carry	FC	RCR EN ENO xCarryFlag wData uiBit	:= RCR(xCarryFlag := , wData := , uiBit :=);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input/Output	xCarryFlag	Carry flag	Follows data type	-	Carry flag
	wData	Data Source	Follows data type	-	Data source
Input	uiBit	Number of bits to rotate	Follows data type	-	Number of bits to rotate
Output	OUT	Return value	Follows data type	-	Execution result

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
xCarryFlag	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
wData	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
uiBit	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

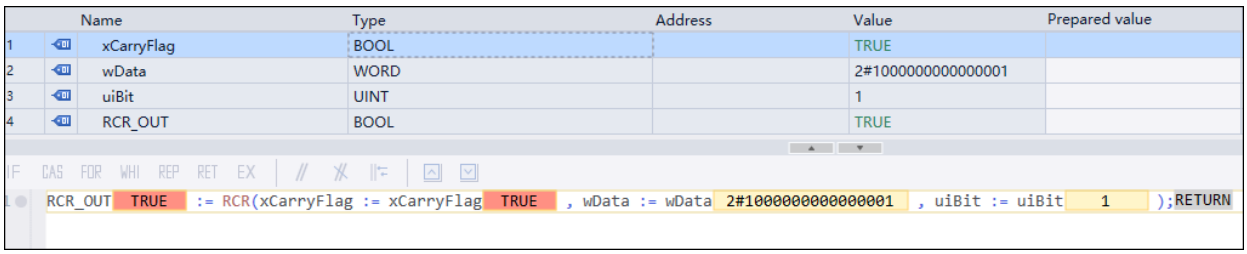
- **Function**

  Rotates the operand to the right. Bits that are shifted out from the right are directly appended to the leftmost bit. When a carry occurs, the carry flag is set to ON.

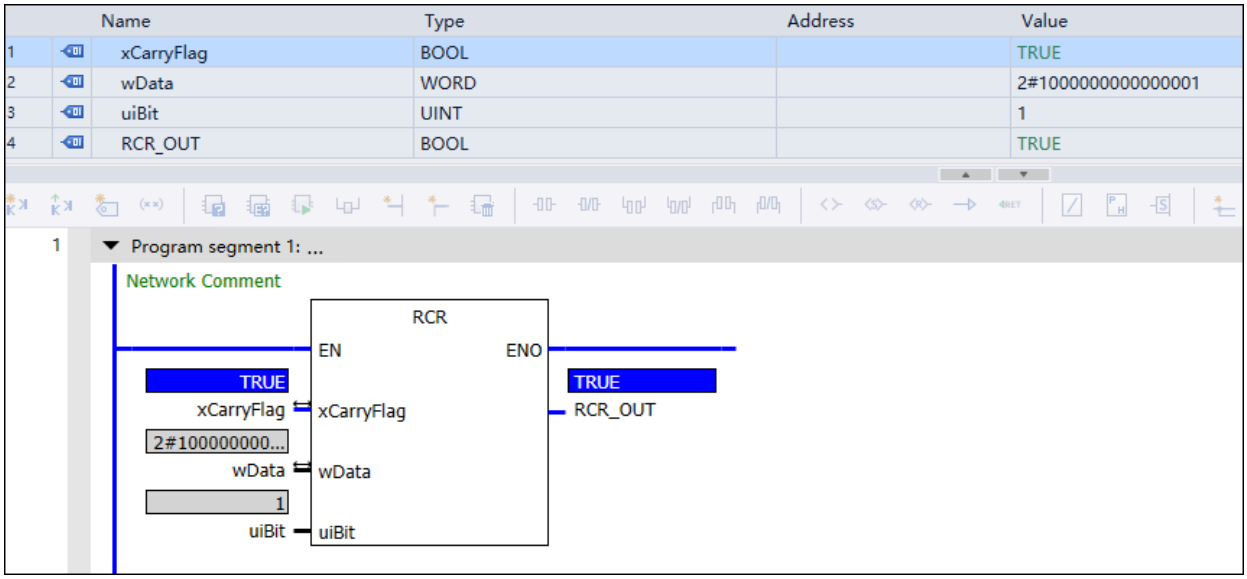
- **Program example**

  When the input source "wData" is set to 2#0000000000000011 and "uiBit" to 1, upon triggering the first time, the output signal "OUT" becomes 2#1000000000000001. The valid data shifts right by 1 bit, and the carry flag is set to TRUE.

  ST:



  LD:

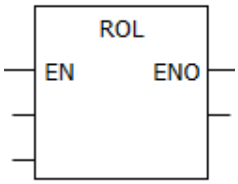


- **Precautions**
  - If "uiBit" is less than 1 or greater than 16, the instruction execution fails, and the return value is FALSE, with the output variable remaining at its initial value.
  - The instruction returns TRUE upon successful execution, otherwise returns FALSE.

18.1.7.4 ROL

Rotates the input value to the left, with the bits shifted out from the left directly appended to the rightmost bit.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ROL	Rotate left shift instruction	FC		ROL(,)

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Data Source	Follows data type	-	Data source
	In2	Number of shifted bits	Follows data type	-	Number of shifted bits
Output	Out	Return value	Follows data type	-	Shifted result

Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	–	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–
In2	–	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–
Out	–	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–

- **Function**

  Rotates the input value to the left, with the bits shifted out from the left directly appended to the rightmost bit.

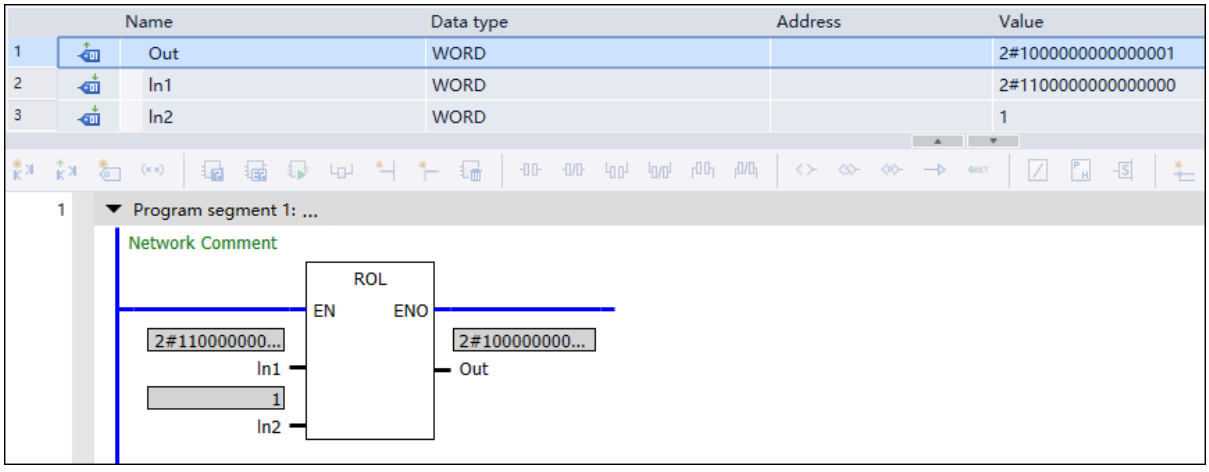
- **Program example**

  When the input source "In1" is set to 2#1100000000000000 and "In2" to 1, the output signal "Out" becomes 2#1000000000000001. The valid data shifts left by 1 bit.

  ST:



  LD:



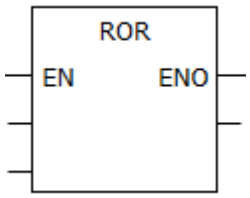
- **Precautions**

  The instruction supports integer data types.

18.1.7.5 ROR

Rotates the input value to the right, with the bits shifted out from the right directly appended to the leftmost bit.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ROR	Rotate right shift instruction	FC		ROR(,)

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Data Source	Follows data type	-	Data source
	In2	Number of shifted bits	Follows data type	-	Number of shifted bits
Output	Out	Return value	Follows data type	-	Shifted result

Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	–	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–
In2	–	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–
Out	–	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–

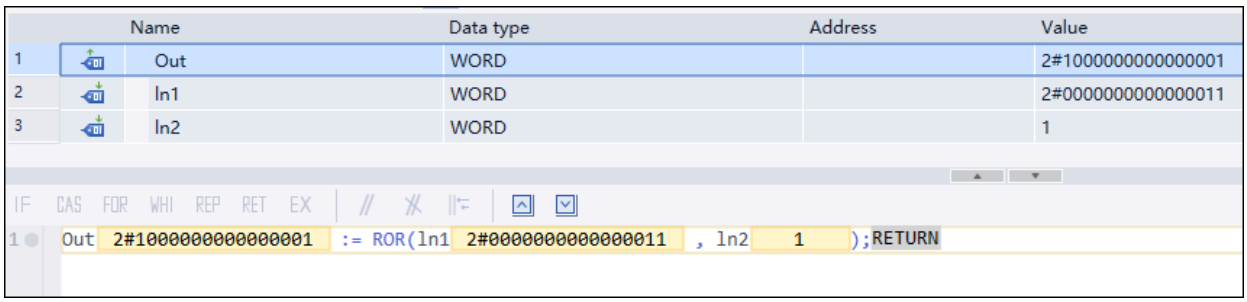
- **Function**

Rotates the input value to the right, with the bits shifted out from the right directly appended to the leftmost bit.

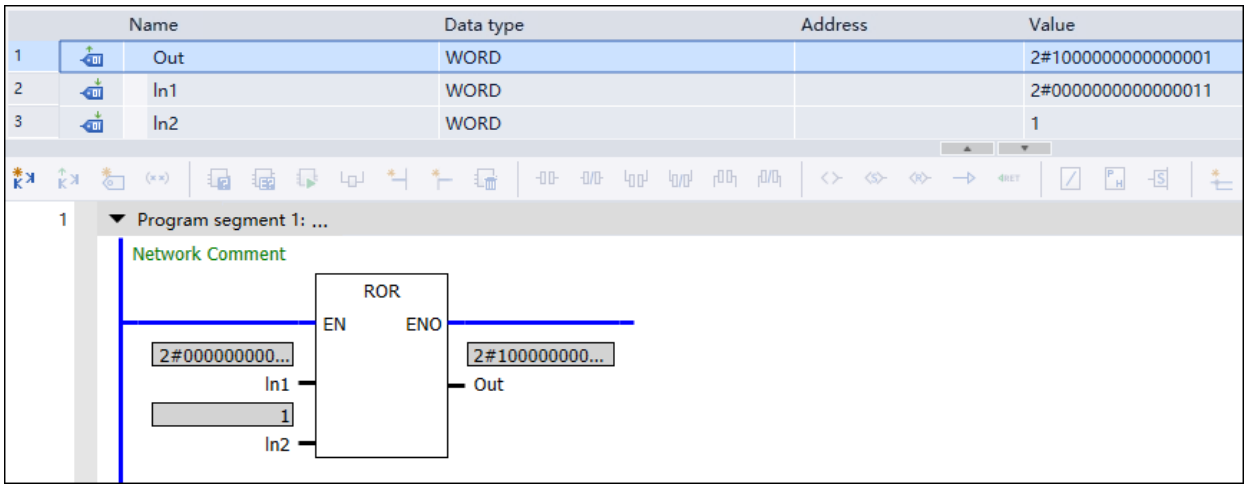
- **Program example**

When the input source "In1" is set to 2#0000000000000011 and "In2" to 1, the output signal "Out" becomes 2#1000000000000001. The valid data shifts right by 1 bit.

ST:



LD:



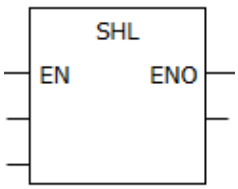
- **Precautions**

  The instruction supports integer data types.

18.1.7.6 SHL

Shifts the input value to the left. The bits shifted out from the left are ignored and zeros are automatically filled in on the right.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SHL	Left shift instruction	FC		SHL(,)

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Data Source	Follows data type	-	Data source

Input/Output	Variable	Name	Valid Range	Default Value	Description
	In2	Number of shifted bits	Follows data type	-	Number of shifted bits
Output	Out	Return value	Follows data type	-	Shifted result

Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	–	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–
In2	–	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–
Out	–	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–

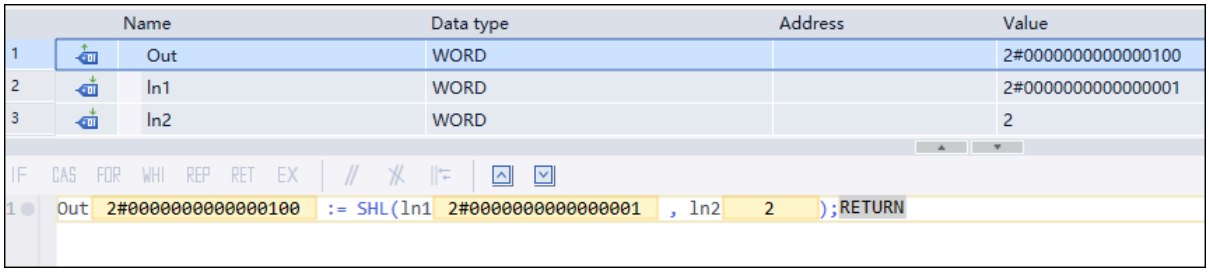
- **Function**

  Shifts the input value to the left. The bits shifted out from the left are ignored and zeros are automatically filled in on the right.

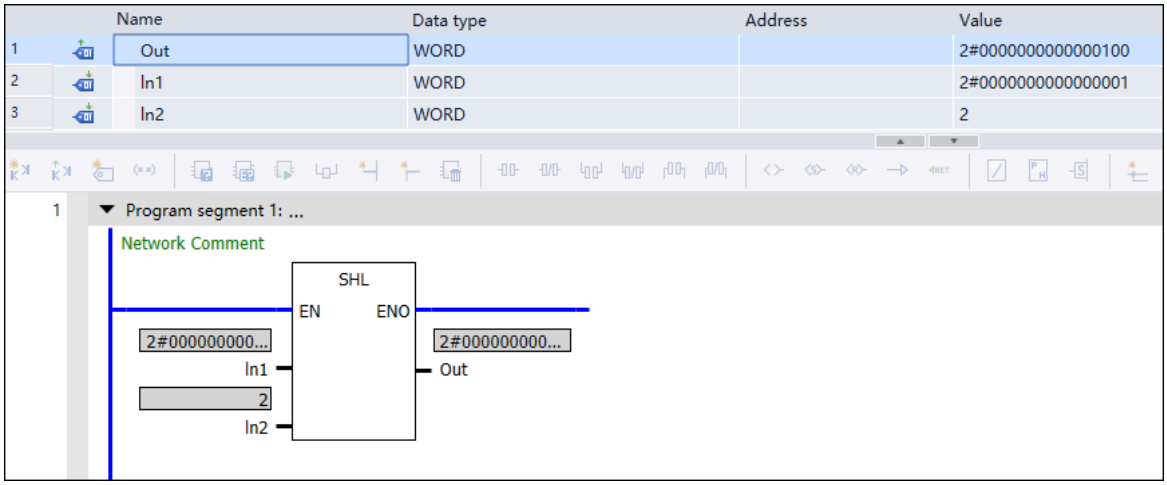
- **Program example**

  When the input source "In1" is set to 2#0000000000000001 and "In2" to 2, the output signal "Out" becomes 2#0000000000000100. The valid data shifts left by 2 bits.

  ST:



  LD:



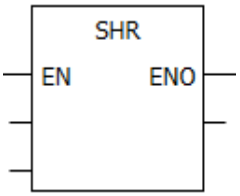
- **Precautions**

  The data source must be an integer; floating-point data triggers a fault.

18.1.7.7 SHR

Shifts the input value to the right. The bits shifted out from the right are ignored and zeros are automatically filled in on the left.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SHR	Right shift instruction	FC		SHR(,)

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Data source	Follows data type	-	Data source
	In2	Number of shifted bits	Follows data type	-	Number of shifted bits
Output	Out	Return value	Follows data type	-	Shifted result

  Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	–	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–
In2	–	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–
Out	–	√	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–

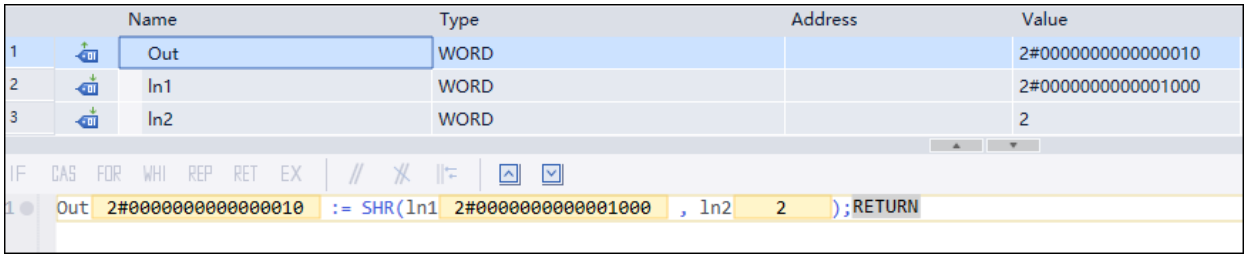
- **Function**

  Shifts the input value to the right. The bits shifted out from the right are ignored and zeros are automatically filled in on the left.

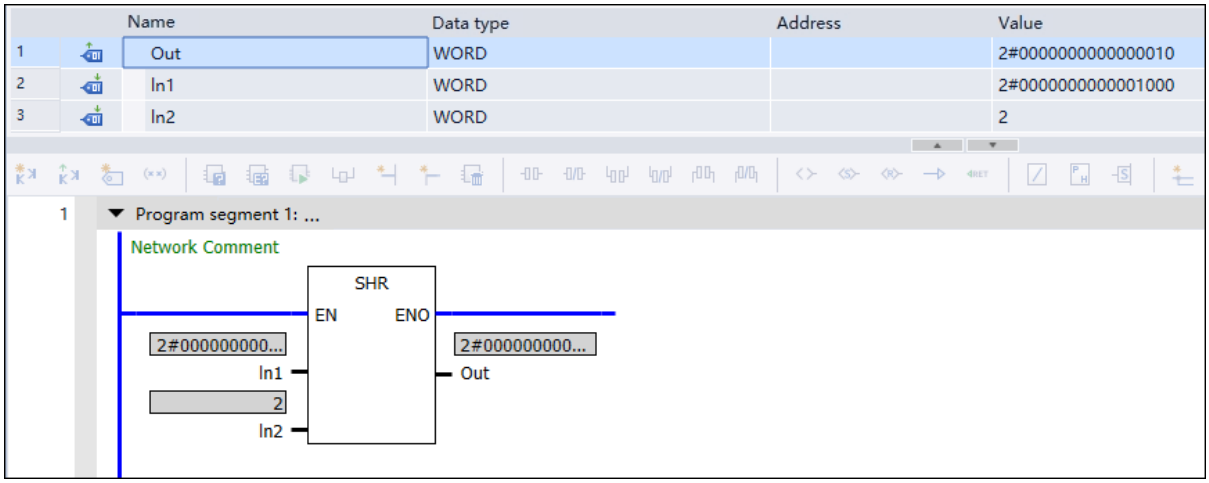
- **Program example**

  When the input source "In1" is set to 2#0000000000001000 and "In2" to 2, the output signal "Out" becomes 2#0000000000000010. The valid data shifts right by 2 bits.

  ST:



  LD:



- **Precautions**

  The data source must be an integer; floating-point data triggers a fault.

18.1.8 Counter

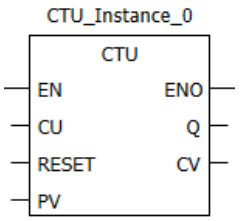
18.1.8.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.1.8.2 CTU	FB	Addition counter	Standard	Library version 1.0.0.0 and later
18.1.8.3 CTD	FB	Subtraction counter		Library version 1.0.0.0 and later
18.1.8.4 CTUD	FB	Addition/ subtraction counter		Library version 1.0.0.0 and later
18.1.8.5 CTU_DINT	FB	Addition counter	Standard64	Library version 1.0.0.0 and later
18.1.8.6 CTU_UDINT	FB	Addition counter		Library version 1.0.0.0 and later
18.1.8.7 CTU_LINT	FB	Addition counter		Library version 1.0.0.0 and later
18.1.8.8 CTU_ULINT	FB	Addition counter		Library version 1.0.0.0 and later
18.1.8.9 CTD_DINT	FB	Subtraction counter		Library version 1.0.0.0 and later
18.1.8.10 CTD_UDINT	FB	Subtraction counter		Library version 1.0.0.0 and later
18.1.8.11 CTD_LINT	FB	Subtraction counter		Library version 1.0.0.0 and later
18.1.8.12 CTD_ULINT	FB	Subtraction counter		Library version 1.0.0.0 and later
18.1.8.13 CTUD_DINT	FB	Reversible counter		Library version 1.0.0.0 and later
18.1.8.14 CTUD_UDINT	FB	Reversible counter		Library version 1.0.0.0 and later
18.1.8.15 CTUD_LINT	FB	Reversible counter		Library version 1.0.0.0 and later
18.1.8.16 CTUD_ULINT	FB	Reversible counter		Library version 1.0.0.0 and later

18.1.8.2 CTU

CTU is an addition counter.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CTU	Addition counter	FB		CTU_Instance_0(CU := , Reset := , PV := , Q => , CV =>);

- **Related variable**

Variable

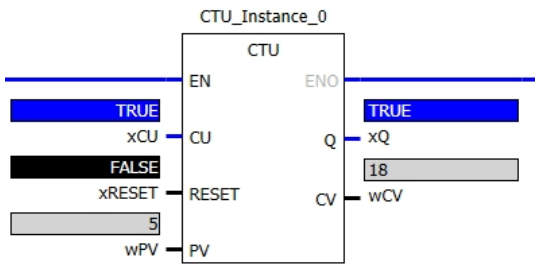
Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	CU	Counter input	Follows data type	-	Counter input
	RESET	Reset signal	Follows data type	-	TRUE: Resets "CV" to 0.
	PV	Preset value	Follows data type	-	Preset value of the counter
Output	Q	Counter output	Follows data type	-	TRUE: Counter output is ON. FALSE: Counter output is OFF.
	CV	Count value	Follows data type	-	The current value of the counter

Variable types

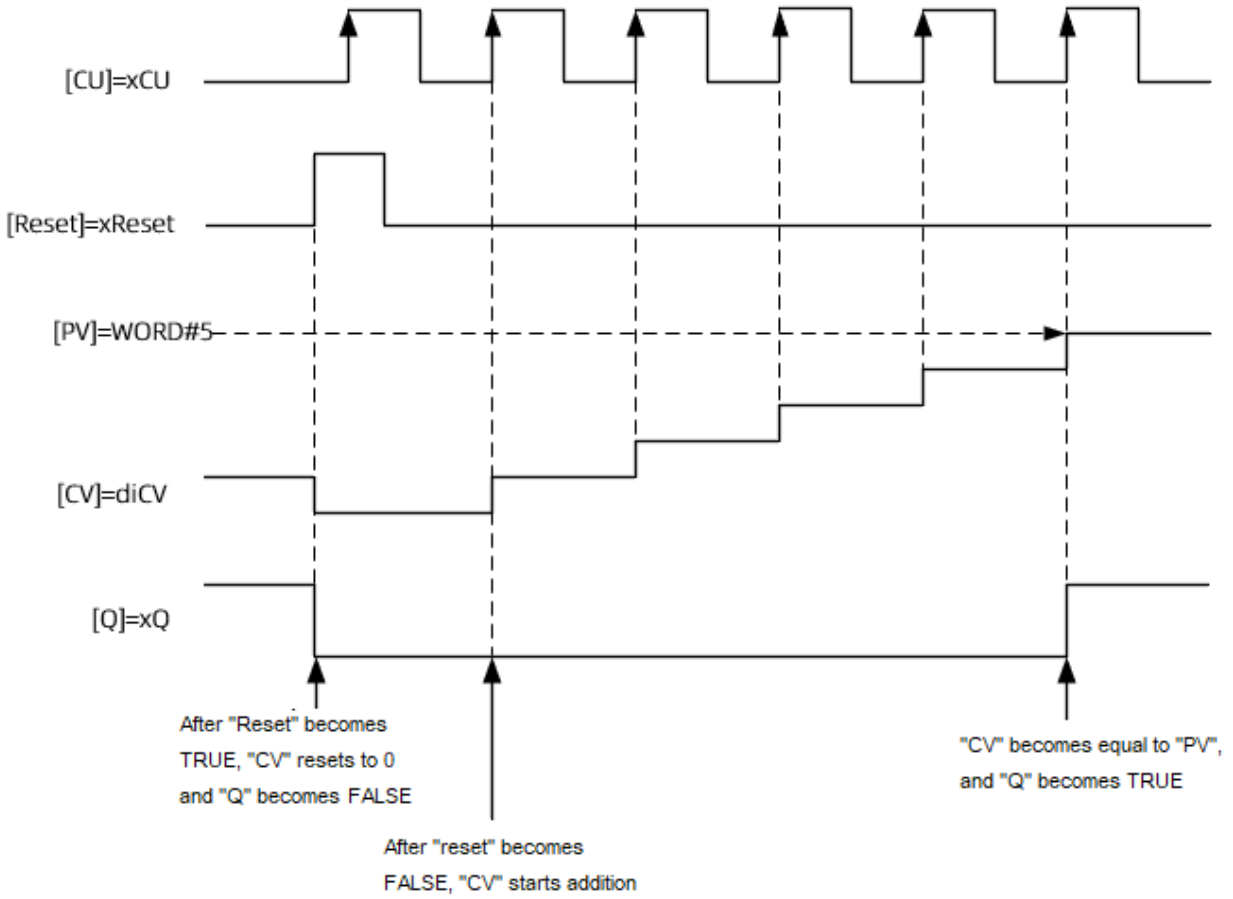
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
CU	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PV	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CV	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - If the reset signal "RESET" is set to TRUE, the counter value "CV" is reset to 0, the counter output "Q" becomes FALSE. The status of the counter input "CU" will not affect this instruction.
  - When "CU" changes from FALSE to TRUE (rising edge of the signal), this instruction is executed, and "CV" is incremented by "1". Upon each rising-edge signal of "CU", "CV" is incremented, until it reaches the maximum value of the specified data type (Max. 65535 for WORD type).
  - The counter output "Q" can be used to query the counting status. The status of "Q" is determined by "CV". If "CV" is greater than or equal to the preset value "PV", the status of "Q" is set to TRUE. In all other cases, the status of "Q" is FALSE.
- **Program example**
  - When "RESET" is set to TRUE, "CV" is reset to 0, and "Q" becomes FALSE. After "RESET" becomes FALSE, "CV" starts to be incremented. "Q" becomes TRUE when "CV" reaches the value of "PV".
  - The following example sets the preset value "PV" to 5.

	LD	ST
Define variables	VAR CTU_Instance_0: CTU; xCU: BOOL; xRESET: BOOL; wPV: WORD := 5; xQ: BOOL; wCV: WORD; END_VAR	
Program		CTU_Instance_0(CU TRUE := xCU TRUE , RESET FALSE := xRESET FALSE , PV 5 := wPV 5 , Q TRUE => xQ TRUE , CV 18 => wCV 18);RETURN

  - When the preset value "PV" is set to 5, the timing diagram is shown in the figure below.



18.1.8.3 CTD

CTD is a subtraction counter.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CTD	Subtraction counter	FB	CTD_Instance_0 CTD EN ENO CD Q LOAD CV PV	CTD_Instance_0(CD := , LOAD := , PV := , Q => , CV =>);

- **Related variable**

Variable

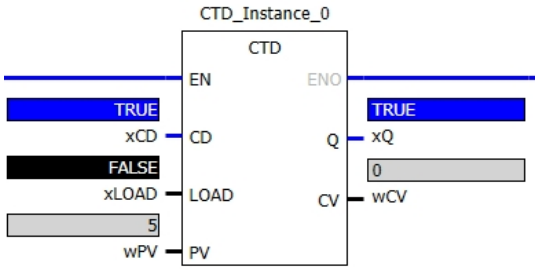
Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	CD	Counter input	Follows data type	-	Counter input
	LOAD	Load signal	Follows data type	-	TRUE: Loads "PV" into "CV".

Input/Output	Variable	Name	Valid Range	Default Value	Description
	PV	Preset value	Follows data type	-	Preset value of the counter
Output	Q	Counter output	Follows data type	-	TRUE: Counter output is ON. FALSE: Counter output is OFF.
	CV	Count value	Follows data type	-	The current value of the counter

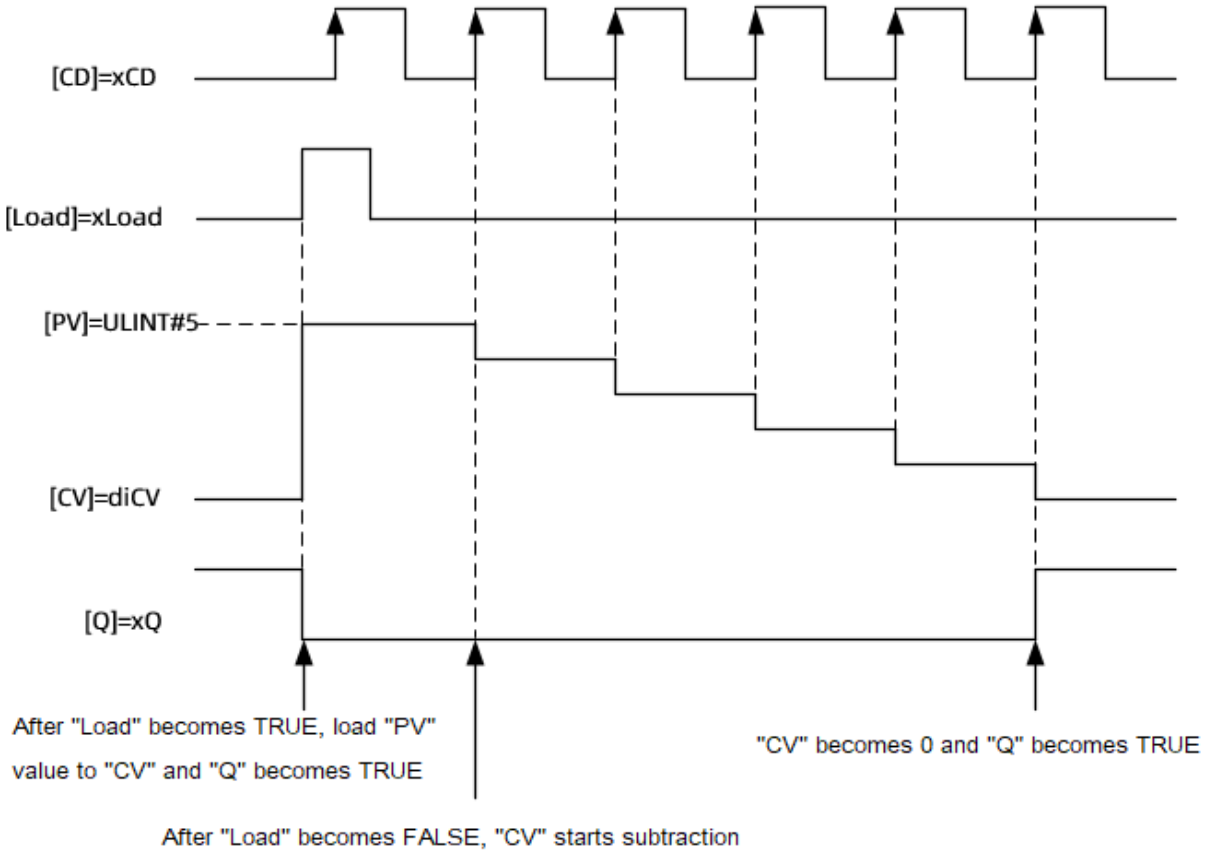
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
CD	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
LOAD	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PV	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CV	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - If the load signal "LOAD" is set to TRUE, the preset value "PV" is loaded into the count value "CV", and the counter output "Q" becomes FALSE.
  - When the counter input signal "CD" changes from FALSE to TRUE (rising edge), "CV" is decremented. When "CV" becomes less than 0, "Q" becomes TRUE, and even if "CD" is at a rising edge, no subtraction is performed on "CV".
- **Program example**
  - When "LOAD" is set to TRUE, "PV" is loaded into "CV", and "Q" becomes FALSE. After "LOAD" becomes FALSE, "CV" starts to be decremented, "CV" becomes 0, and "Q" becomes TRUE.
  - The following example sets the preset value "PV" to 5.

	LD	ST
Define variables	VAR CTD_Instance_0: CTD; xCD: BOOL; xLOAD: BOOL; wPV: WORD := 5; xQ: BOOL; wCV: WORD; END_VAR	
Program		CTD_Instance_0(CD TRUE := xCD TRUE, LOAD FALSE := xLOAD FALSE, PV 5 := wPV 5, Q TRUE => xQ TRUE, CV 0 => wCV 0);RETURN

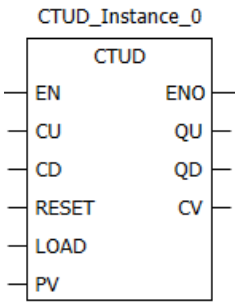
◦ When the preset value "PV" is set to 5, the timing diagram is shown in the figure below.



18.1.8.4 CTUD

CTUD is an addition/subtraction counter.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CTUD	Addition/ subtraction counter	FB		CTUD_Instance_0(CU := , CD := , RESET := , LOAD := , PV := , QU => , QD => , CV =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	CU	Addition counter input	Follows data type	-	Addition counter input
	CD	Subtraction counter input	Follows data type	-	Subtraction counter input
	RESET	Reset signal	Follows data type	-	TRUE: Resets "CV" to 0.
	LOAD	Load signal	Follows data type	-	TRUE: Loads "PV" into "CV".
	PV	Preset value	Follows data type	-	Preset value of the counter
Output	QU	Addition counter output	Follows data type	-	TRUE: Addition counter output is ON. FALSE: Addition counter output is OFF.

Input/Output	Variable	Name	Valid Range	Default Value	Description
	QD	Subtraction counter output	Follows data type	-	TRUE: Subtraction counter output is ON. FALSE: Subtraction counter output is OFF.
	CV	Count value	Follows data type	-	The current value of the counter

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
CU	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CD	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
LOAD	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PV	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
QU	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
QD	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CV	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  CTUD is a counter that performs addition and subtraction operations based on the input signals. It combines the functions of both the addition counter and the subtraction counter. Data type of preset value "PV" and counter value "CV" is WORD.

  ◦ Addition counter function

  When the reset signal "RESET" is set to TRUE, the counter value "CV" becomes 0, and the addition counter output "QU" becomes FALSE. When addition counter input signal "CU" changes from FALSE to

TRUE (rising edge), this instruction is executed, and "CV" is incremented by "1". Upon each rising-edge signal of "CU", "CV" is incremented, until it reaches the maximum value of the specified data type (Max. 65535 for WORD type).

◦ Subtraction counter function

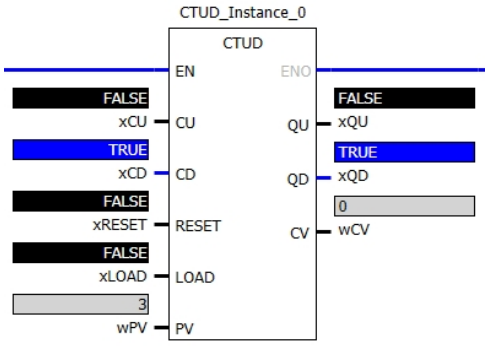
When the load signal "LOAD" is set to TRUE, the preset value "PV" is loaded into the count value "CV", and the subtraction counter output "QD" becomes FALSE. When the subtraction counter input signal "CD" changes from FALSE to TRUE (rising edge), "CV" is decremented. When the value of "CV" is less than 0, the value of "QD" becomes TRUE. When the value of "CV" is less than 0, even if a larger value of "CD" is input, "CV" remains unchanged.

◦ Common functions of the addition/subtraction counter

When "LOAD" or "RESET" is TRUE, the rising edges of "CU" and "CD" are ignored, and "CV" does not perform addition or subtraction operations. When "CU" and "CD" are both at the rising edge, "CV" does not change. When both "RESET" and "LOAD" are TRUE, "RESET" takes priority, and the value of "CV" becomes 0. When "RESET" is set to TRUE, the value of "CV" becomes 0, and the value of "QD" becomes TRUE. When "LOAD" is set to TRUE, the value of "CV" becomes equal to "PV", and the value of "QU" becomes TRUE.

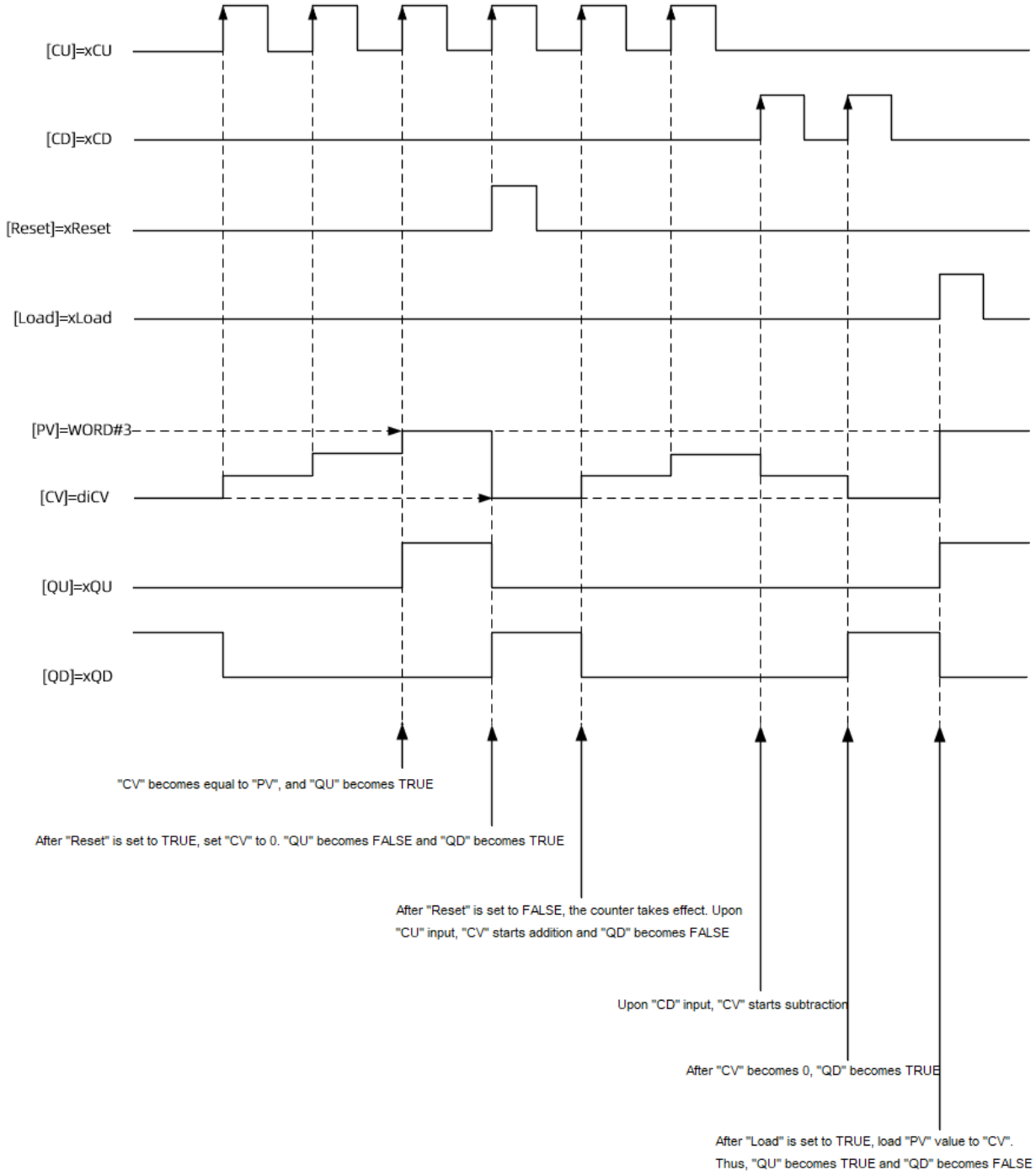
RESET	LOAD	CV	QU	QD	Action
FALSE	FALSE	Lower than 0	FALSE	TRUE	Acts only as an addition counter. When "CU" is at a rising edge, "CV" performs addition. When "CD" is at a rising edge, no subtraction is performed.
		Between 0 and "PV"	FALSE	FALSE	Acts as both addition/subtraction counter. When "CU" is at a rising edge, "CV" performs addition. When "CD" is at a rising edge, "CV" performs subtraction.
		Greater than "PV"	TRUE	FALSE	Acts only as a subtraction counter. When "CD" is at a rising edge, "CV" performs subtraction. When "CU" is at the rising edge, no addition operation is performed.
TRUE	FALSE	0	FALSE	TRUE	Reset status of the addition counter. The value of "CV" is set to 0.
FALSE	TRUE	"PV"	TRUE	FALSE	Reset status of the subtraction counter. The value of "CV" is set to "PV".
TRUE	TRUE	0	FALSE	TRUE	Reset status of the addition counter. "Reset" takes priority over "LOAD", and the value of "CV" is set to 0.

- **Program example**
  - The following example sets the preset value "PV" to 3.

	LD	ST
Define variables	VAR CTUD_Instance_0: CTUD; xCU: BOOL; xCD: BOOL; xREST: BOOL; xLOAD: BOOL; wPV: WORD := 3; xQU: BOOL; xQD: BOOL; wCV: WORD; END_VAR	
Program		CTUD_Instance_0(CU FALSE := xCU FALSE , CD TRUE := xCD TRUE , RESET FALSE := xRESET FALSE , LOAD FALSE := bLOAD FALSE , PV 3 := wPV 3 , QU FALSE => bQU FALSE , QD TRUE => bQD TRUE , CV 0 => wCV 0);RETURN

◦ When the preset value "PV" is set to 3, the timing diagram is shown in the figure below.

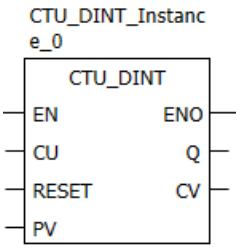
- When the input signal "CU" changes from FALSE to TRUE (rising edge), "CV" is incremented. When "CV" becomes equal to the preset value "PV", "QU" becomes TRUE.
- When "RESET" is set to TRUE, "CV" is reset to 0, "QU" becomes FALSE, and "QD" becomes TRUE.
- After "RESET" becomes FALSE, "CV" starts to be incremented and "QD" becomes TRUE. "QU" becomes TRUE when "CV" reaches the value of "PV".
- When the input signal "CD" changes from FALSE to TRUE (rising edge), "CV" is decremented. When the value of "CV" becomes 0, the value of "QD" becomes TRUE.
- When "LOAD" is set to TRUE, "PV" is loaded into "CV", "QD" becomes FALSE, and "QU" becomes TRUE.
- After "LOAD" becomes FALSE, "CV" starts to be decremented and "QU" becomes FALSE. When "CV" becomes 0, "QD" becomes TRUE.



18.1.8.5 CTU_DINT

A counter that performs addition on the count value each time the counter input signal is received. The data type of the preset value and count value is DINT.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
CTU_DINT	Addition counter	FB		CTU_DINT_Instance_0(CU := , RESET := , PV := , Q => , CV =>);

- **Related variable**

Variable

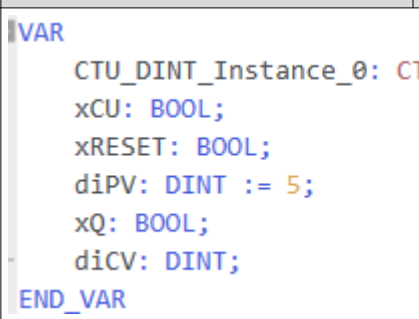
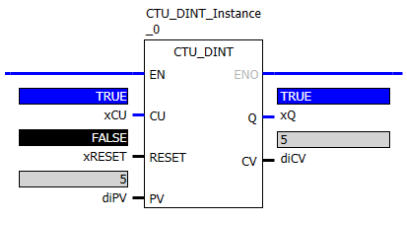
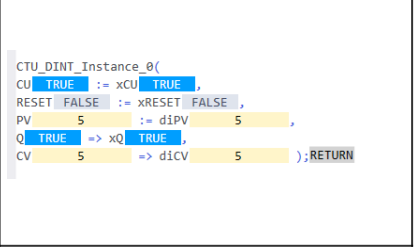
Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	CU	Counter input	Follows data type	-	Counter input
	RESET	Reset signal	Follows data type	-	TRUE: Resets "CV" to 0.
	PV	Preset value	Follows data type	-	Preset value of the counter
Output	Q	Counter output	Follows data type	-	TRUE: Counter output is ON. FALSE: Counter output is OFF.
	CV	Count value	Follows data type	-	The current value of the counter

Variable types

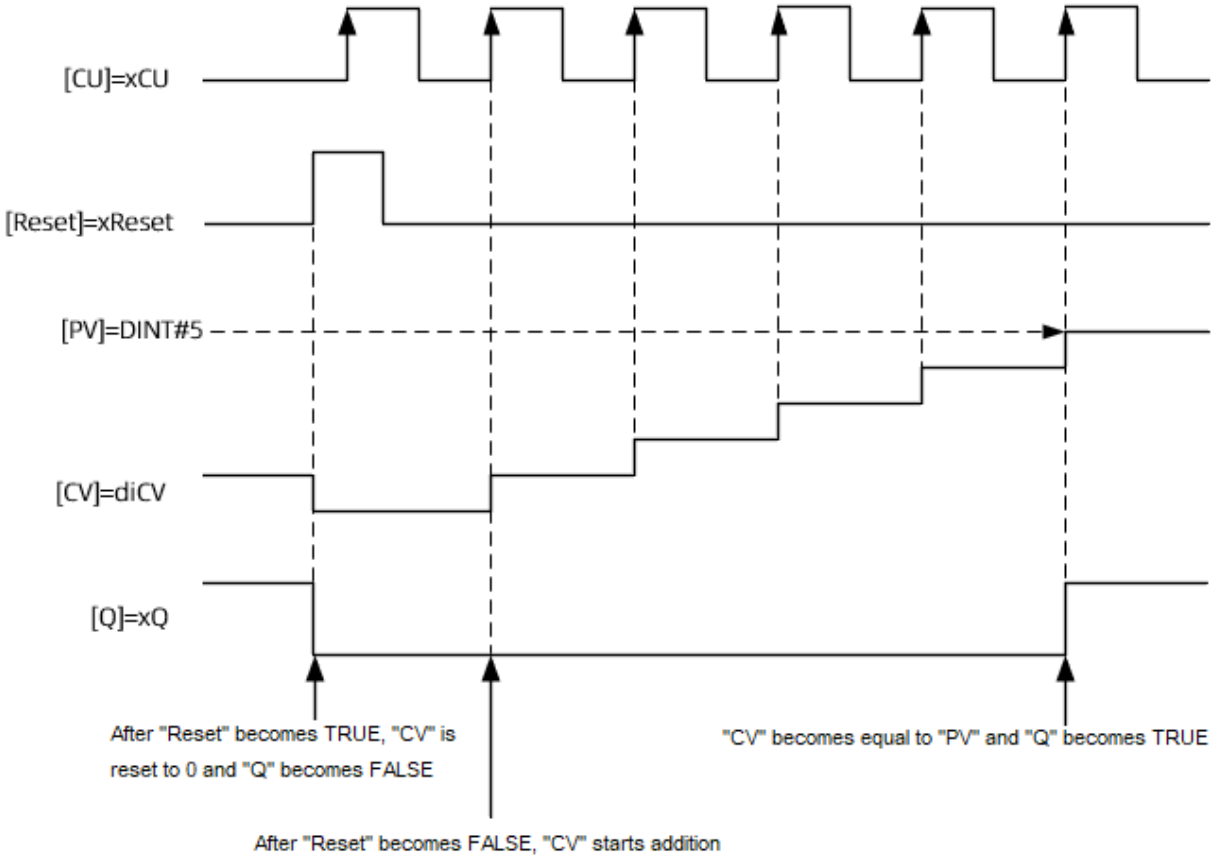
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
CU	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PV	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
CV	Same data type as "PV"																			

- **Function description**
  - If the reset signal "RESET" is set to TRUE, the count value "CV" becomes 0, and the counter output "Q" becomes FALSE. In this case, regardless of whether the counter input "CU" is at a rising edge, "CV" does not perform addition, so the status of "CU" does not affect this instruction.
  - When "CU" changes from FALSE to TRUE (rising edge of the signal), this instruction is executed, and "CV" is incremented by "1". Upon each rising-edge signal of "CU", "CV" is incremented, until it reaches the maximum value of the specified data type.
  - The counter output "Q" can be used to query the counting status. The status of "Q" is determined by "CV". If "CV" is greater than or equal to the preset value "PV", the status of "Q" is set to TRUE. In all other cases, the status of "Q" is FALSE.
- **Program example**
  - When "RESET" is set to TRUE, "CV" is reset to 0, and "Q" becomes FALSE. After "RESET" becomes FALSE, "CV" starts to be incremented. "Q" becomes TRUE when "CV" reaches the value of "PV".
  - The following example sets the preset value "PV" to 5.

	LD	ST
Define variables		
Program		

  - When the preset value "PV" is set to 5, the timing diagram is shown in the figure below.



18.1.8.6 CTU_UDINT

A counter that performs addition on the count value each time the counter input signal is received. The data type of the preset value and count value is UDINT.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
CTU_UDINT	Addition counter	FB	CTU_UDINT_Instance_0 CTU_UDINT EN ENO CU Q RESET CV PV	CTU_UDINT_Instance_0(CU := , RESET := , PV := , Q => , CV =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	CU	Counter input	Follows data type	-	Counter input

Input/Output	Variable	Name	Valid Range	Default Value	Description
	RESET	Reset signal	Follows data type	-	TRUE: Resets "CV" to 0.
	PV	Preset value	Follows data type	-	Preset value of the counter
Output	Q	Counter output	Follows data type	-	TRUE: Counter output is ON. FALSE: Counter output is OFF.
	CV	Count value	Follows data type	-	The current value of the counter

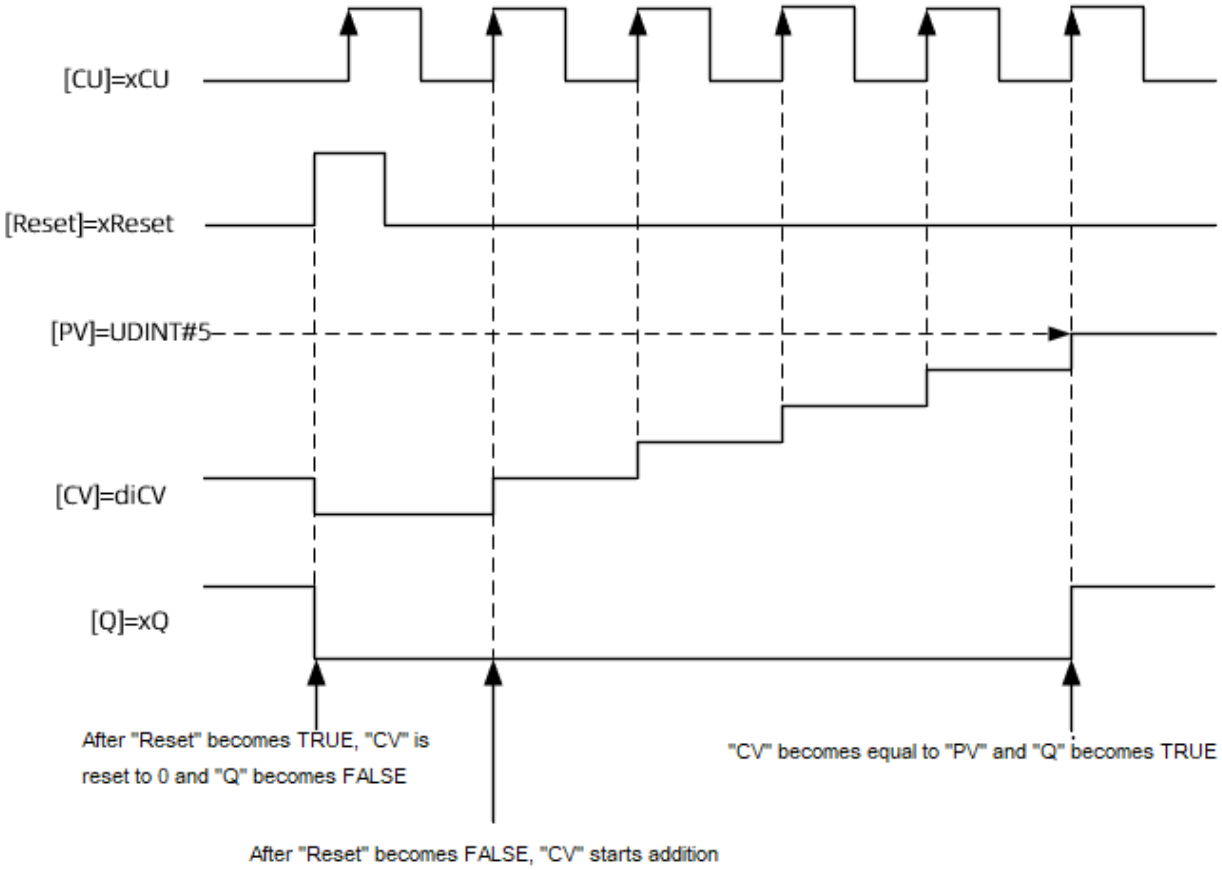
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
CU	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PV	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CV	Same data type as "PV"																			

- **Function description**
  - If the reset signal "RESET" is set to TRUE, the count value "CV" becomes 0, and the counter output "Q" becomes FALSE. In this case, regardless of whether the counter input "CU" is at a rising edge, "CV" does not perform addition, so the status of "CU" does not affect this instruction.
  - When "CU" changes from FALSE to TRUE (rising edge of the signal), this instruction is executed, and "CV" is incremented by "1". Upon each rising-edge signal of "CU", "CV" is incremented, until it reaches the maximum value of the specified data type.
  - The counter output "Q" can be used to query the counting status. The status of "Q" is determined by "CV". If "CV" is greater than or equal to the preset value "PV", the status of "Q" is set to TRUE. In all other cases, the status of "Q" is FALSE.
- **Program example**
  - When "RESET" is set to TRUE, "CV" is reset to 0, and "Q" becomes FALSE. After "RESET" becomes FALSE, "CV" starts to be incremented. "Q" becomes TRUE when "CV" reaches the value of "PV".
  - The following example sets the preset value "PV" to 5.

	LD	ST
Define variables	VAR CTU_UDINT_Instance_0: CTU_UDINT; xCU: BOOL; xRESET: BOOL; udiPV: UDINT := 5; xQ: BOOL; udiCV: UDINT; END_VAR	
Program	CTU_UDINT_Instance_0 CTU_UDINT EN ENO TRUE xCU CU Q xQ TRUE FALSE xRESET RESET CV udiCV 5 5 udiPV PV	CTU_UDINT_Instance_0(CU TRUE := xCU TRUE, RESET FALSE := xRESET FALSE, PV 5 := udiPV 5, Q TRUE => xQ TRUE, CV 5 => udiCV 5);RETURN

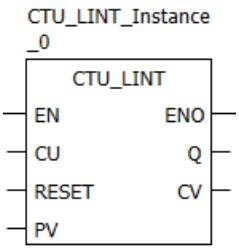
◦ When the preset value "PV" is set to 5, the timing diagram is shown in the figure below.



18.1.8.7 CTU_LINT

A counter that performs addition on the count value each time the counter input signal is received. The data type of the preset value and count value is LINT.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
CTU_LINT	Addition counter	FB		CTU_LINT_Instance_0(CU := , RESET := , PV := , Q => , CV =>);

- **Related variable**

Variable

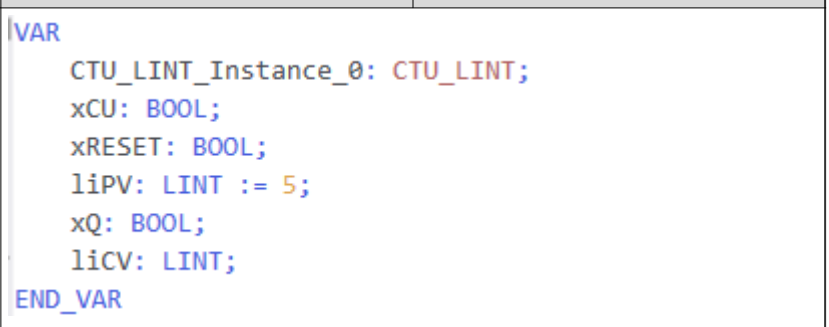
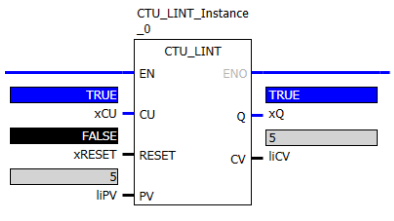
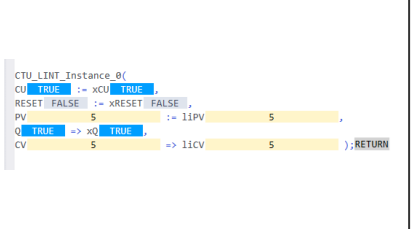
Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	CU	Counter input	Follows data type	-	Counter input
	RESET	Reset signal	Follows data type	-	TRUE: Resets "CV" to 0.
	PV	Preset value	Follows data type	-	Preset value of the counter
Output	Q	Counter output	Follows data type	-	TRUE: Counter output is ON. FALSE: Counter output is OFF.
	CV	Count value	Follows data type	-	The current value of the counter

Variable types

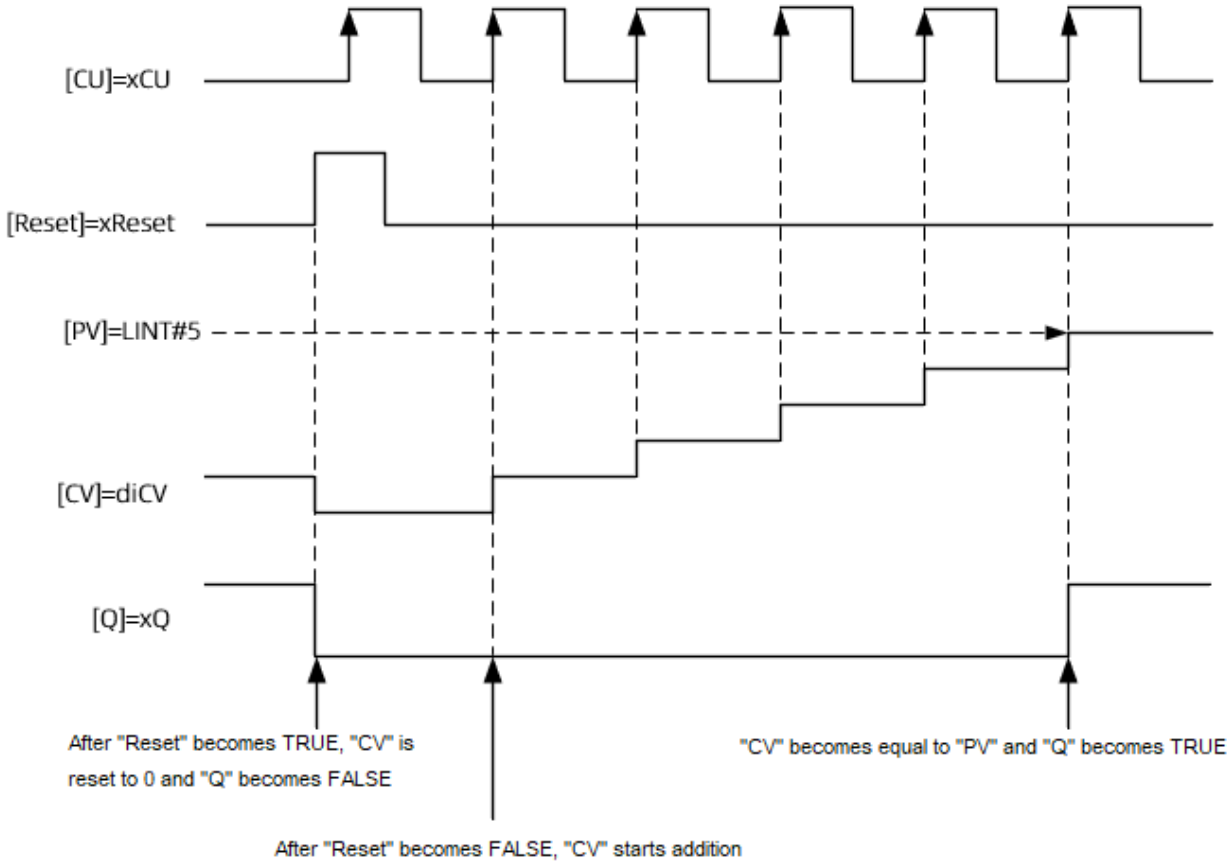
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
CU	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PV	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
CV	Same data type as "PV"																			

- **Function description**
  - If the reset signal "RESET" is set to TRUE, the count value "CV" becomes 0, and the counter output "Q" becomes FALSE. In this case, regardless of whether the counter input "CU" is at a rising edge, "CV" does not perform addition, so the status of "CU" does not affect this instruction.
  - When "CU" changes from FALSE to TRUE (rising edge of the signal), this instruction is executed, and "CV" is incremented by "1". Upon each rising-edge signal of "CU", "CV" is incremented, until it reaches the maximum value of the specified data type.
  - The counter output "Q" can be used to query the counting status. The status of "Q" is determined by "CV". If "CV" is greater than or equal to the preset value "PV", the status of "Q" is set to TRUE. In all other cases, the status of "Q" is FALSE.
- **Program example**
  - When "RESET" is set to TRUE, "CV" is reset to 0, and "Q" becomes FALSE. After "RESET" becomes FALSE, "CV" starts to be incremented. "Q" becomes TRUE when "CV" reaches the value of "PV".
  - The following example sets the preset value "PV" to 5.

	LD	ST
Define variables		
Program		

  - When the preset value "PV" is set to 5, the timing diagram is shown in the figure below.



18.1.8.8 CTU_ULINT

A counter that performs addition on the count value each time the counter input signal is received. The data type of the preset value and count value is ULINT.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
CTU_ULINT	Addition counter	FB	CTU_ULINT_Instance_0 CTU_ULINT EN ENO CU Q RESET CV PV	CTU_ULINT_Instance_0(CU := , RESET := , PV := , Q => , CV =>);

- **Related variable**

Variable

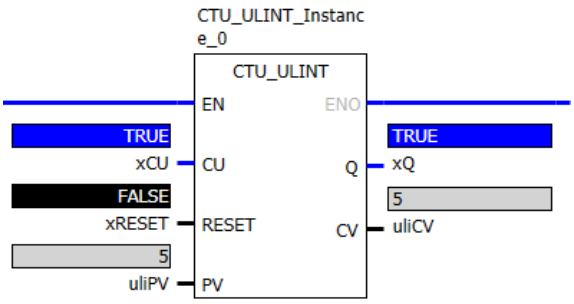
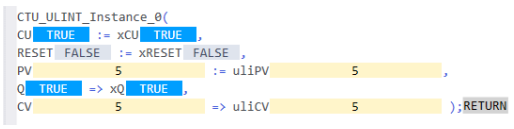
Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	CU	Counter input	Follows data type	-	Counter input

Input/Output	Variable	Name	Valid Range	Default Value	Description
	RESET	Reset signal	Follows data type	-	TRUE: Resets "CV" to 0.
	PV	Preset value	Follows data type	-	Preset value of the counter
Output	Q	Counter output	Follows data type	-	TRUE: Counter output is ON. FALSE: Counter output is OFF.
	CV	Count value	Follows data type	-	The current value of the counter

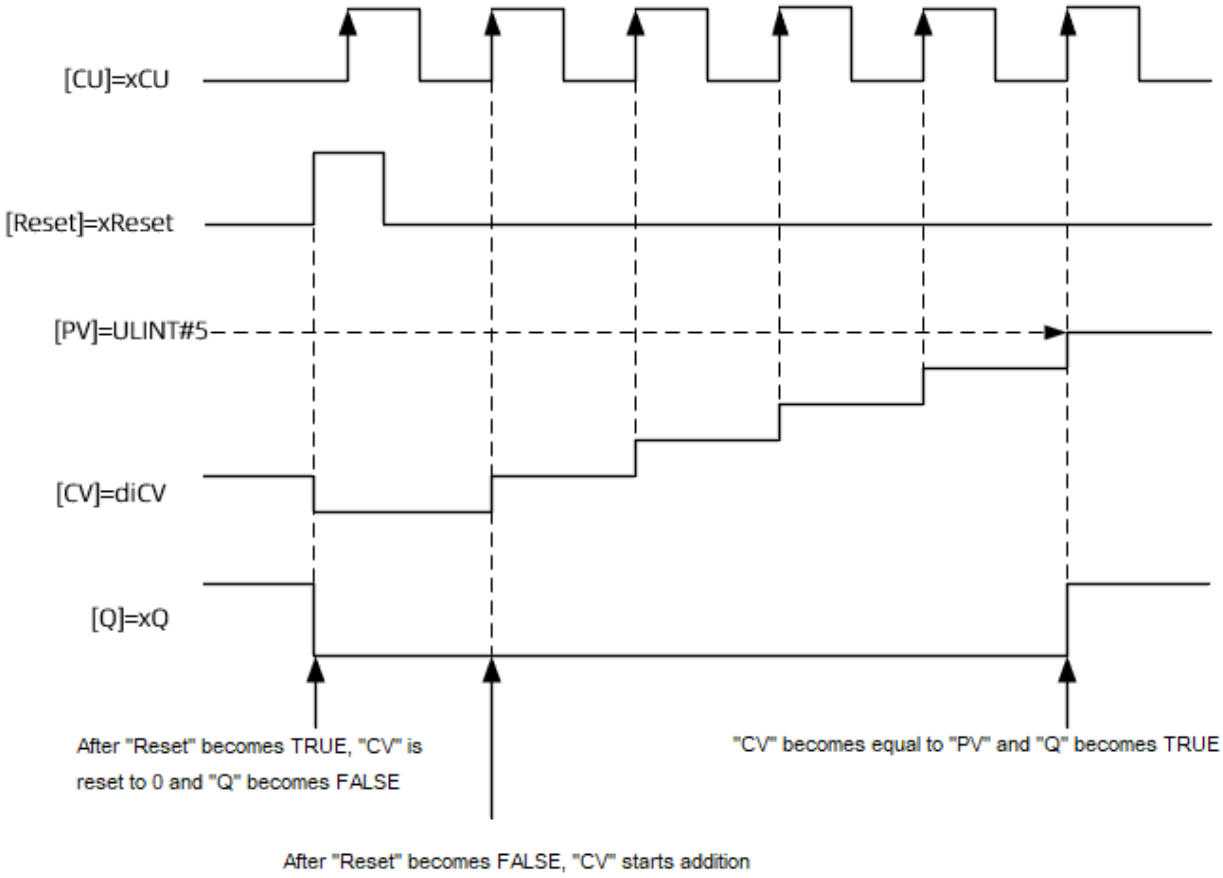
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
CU	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PV	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CV	Same data type as "PV"																			

- **Function description**
  - If the reset signal "RESET" is set to TRUE, the count value "CV" becomes 0, and the counter output "Q" becomes FALSE. In this case, regardless of whether the counter input "CU" is at a rising edge, "CV" does not perform addition, so the status of "CU" does not affect this instruction.
  - When "CU" changes from FALSE to TRUE (rising edge of the signal), this instruction is executed, and "CV" is incremented by "1". Upon each rising-edge signal of "CU", "CV" is incremented, until it reaches the maximum value of the specified data type.
  - The counter output "Q" can be used to query the counting status. The status of "Q" is determined by "CV". If "CV" is greater than or equal to the preset value "PV", the status of "Q" is set to TRUE. In all other cases, the status of "Q" is FALSE.
- **Program example**
  - When "RESET" is set to TRUE, "CV" is reset to 0, and "Q" becomes FALSE. After "RESET" becomes FALSE, "CV" starts to be incremented. "Q" becomes TRUE when "CV" reaches the value of "PV".
  - The following example sets the preset value "PV" to 5.

	LD	ST
Define variables	VAR CTU_ULINT_Instance_0: CTU_ULINT; xCU: BOOL; xRESET: BOOL; uliPV: ULINT := 5; xQ: BOOL; uliCV: ULINT; END_VAR	
Program		

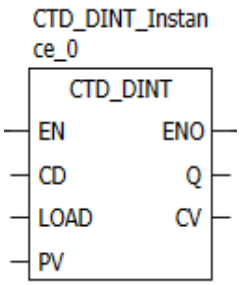
- When the preset value "PV" is set to 5, the timing diagram is shown in the figure below.



18.1.8.9 CTD_DINT

A counter that performs subtraction on the count value each time the counter input signal is received. The data type of the preset value and count value is DINT.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
CTD_DINT	Subtraction counter	FB		CTD_DINT_0(CD := , LOAD := , PV := , Q => , CV =>);

- **Related variable**

Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	CD	Counter input	Follows data type	-	Counter input
	LOAD	Load signal	Follows data type	-	TRUE: Loads "PV" into "CV".
	PV	Preset value	Follows data type	-	Preset value of the counter
Output	Q	Counter output	Follows data type	-	TRUE: Counter output is ON. FALSE: Counter output is OFF.
	CV	Count value	Follows data type	-	The current value of the counter

Variable types

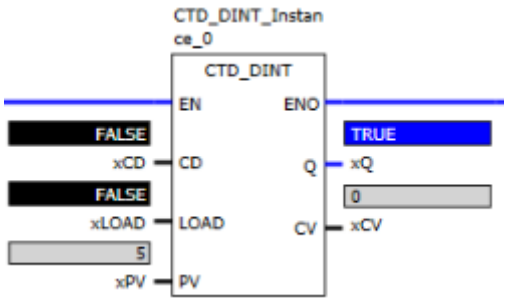
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
CU	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
LOAD	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PV	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
CV	Same data type as "PV"																			

- **Function description**
  - If the load signal "LOAD" is set to TRUE, the preset value "PV" is loaded into the count value "CV", and the counter output "Q" becomes FALSE. At this time, the rising edge of "CD" is ignored, and no subtraction operation is performed on "CV".
  - If the load signal "LOAD" is set to FALSE and the counter input signal "CD" changes from FALSE to TRUE (rising edge), a subtraction operation is performed on "CV". When the value of "CV" is less than 0, even if "CD" is at a rising edge, no subtraction operation is performed on "CV", and the value of "Q" becomes TRUE.
- **Program example**
  - When "LOAD" is set to TRUE, "PV" is loaded into "CV", and "Q" becomes FALSE. After "LOAD" becomes FALSE, "CV" starts to be decremented, "CV" becomes 0, and "Q" becomes TRUE.
  - The following example sets the preset value "PV" to 5.

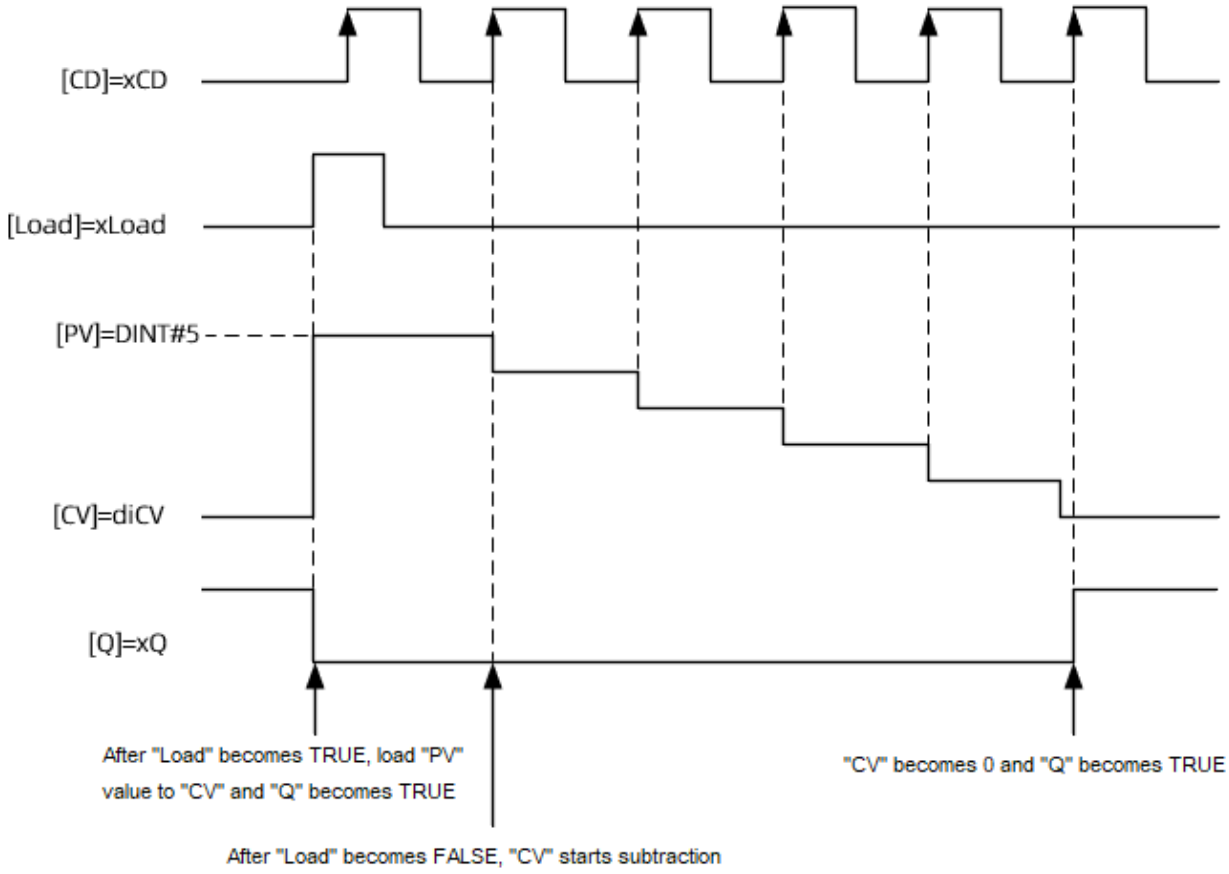
	LD	ST
Define variables		
Program		

```
VAR
    CTD_DINT_Instance_0: CTD_DINT;
    xCD: BOOL;
    xLOAD: BOOL;
    xPV: DINT := 5;
    xQ: BOOL;
    xCV: WORD;
END_VAR
```




```
CTD_DINT_Instance_0(
CD FALSE := xCD FALSE,
LOAD FALSE := xLOAD FALSE,
PV 5 := xPV 5,
Q TRUE => xQ TRUE,
CV 0 => xCV 0);RETURN
```

  - When the preset value "PV" is set to 5, the timing diagram is shown in the figure below.



18.1.8.10 CTD_UDINT

A counter that performs subtraction on the count value each time the counter input signal is received. The data type of the preset value and count value is UDINT.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
CTD_UDINT	Subtraction counter	FB	CTD_UDINT_Instance_0 CTD_UDINT EN ENO CD Q LOAD CV PV	CTD_UDINT_0(CD := , LOAD := , PV := , Q => , CV =>);

- **Related variable**

Variable

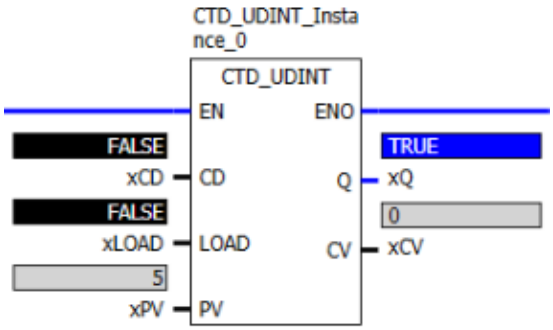
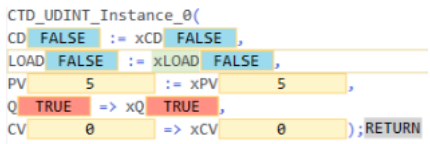
Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	CD	Counter input	Follows data type	-	Counter input

Input/Output	Variable	Name	Valid Range	Default Value	Description
	LOAD	Load signal	Follows data type	-	TRUE: Loads "PV" into "CV".
	PV	Preset value	Follows data type	-	Preset value of the counter
Output	Q	Counter output	Follows data type	-	TRUE: Counter output is ON. FALSE: Counter output is OFF.
	CV	Count value	Follows data type	-	The current value of the counter

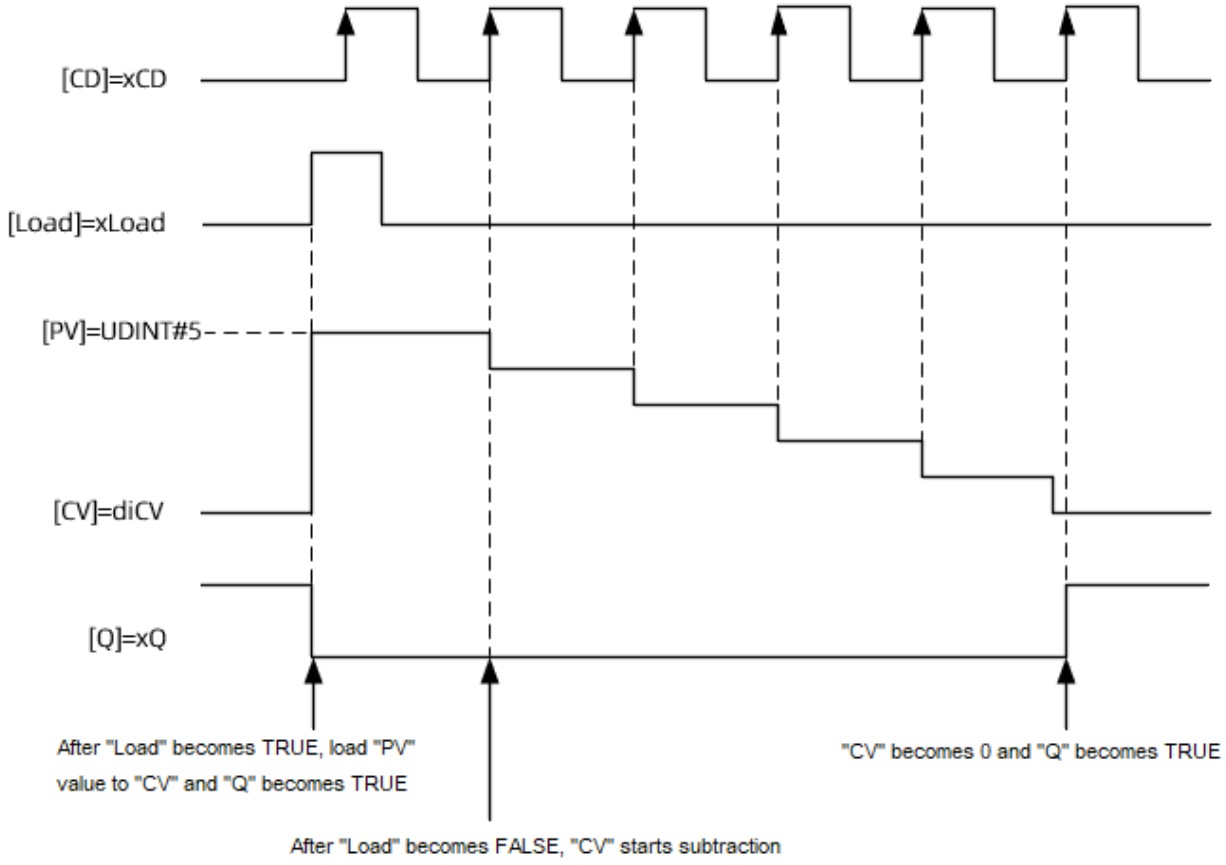
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
CU	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
LOAD	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PV	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CV	Same data type as "PV"																			

- **Function description**
  - If the load signal "LOAD" is set to TRUE, the preset value "PV" is loaded into the count value "CV", and the counter output "Q" becomes FALSE. At this time, the rising edge of "CD" is ignored, and no subtraction operation is performed on "CV".
  - If the load signal "LOAD" is set to FALSE and the counter input signal "CD" changes from FALSE to TRUE (rising edge), a subtraction operation is performed on "CV". When the value of "CV" is less than 0, even if "CD" is at a rising edge, no subtraction operation is performed on "CV", and the value of "Q" becomes TRUE.
- **Program example**
  - When "LOAD" is set to TRUE, "PV" is loaded into "CV", and "Q" becomes FALSE. After "LOAD" becomes FALSE, "CV" starts to be decremented, "CV" becomes 0, and "Q" becomes TRUE.
  - The following example sets the preset value "PV" to 5.

	LD	ST
Define variables	VAR CTD_UDINT_Instance_0: CTD_UDINT; xCD: BOOL; xLOAD: BOOL; xPV: UDINT := 5; xQ: BOOL; xCV: UDINT; END_VAR	
Program		

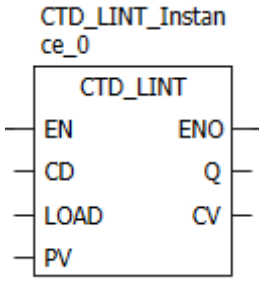
◦ When the preset value "PV" is set to 5, the timing diagram is shown in the figure below.



18.1.8.11 CTD_LINT

A counter that performs subtraction on the count value each time the counter input signal is received. The data type of the preset value and count value is LINT.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
CTD_LINT	Subtraction counter	FB		CTD_LINT_0(CD := , LOAD := , PV := , Q => , CV =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	CD	Counter input	Follows data type	-	Counter input
	LOAD	Load signal	Follows data type	-	TRUE: Loads "PV" into "CV".
	PV	Preset value	Follows data type	-	Preset value of the counter
Output	Q	Counter output	Follows data type	-	TRUE: Counter output is ON. FALSE: Counter output is OFF.
	CV	Count value	Follows data type	-	The current value of the counter

Variable types

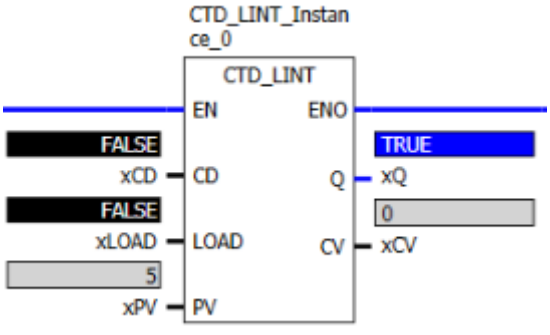
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
CU	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
LOAD	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PV	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
CV	Same data type as "PV"																			

- **Function description**
  - If the load signal "LOAD" is set to TRUE, the preset value "PV" is loaded into the count value "CV", and the counter output "Q" becomes FALSE. At this time, the rising edge of "CD" is ignored, and no subtraction operation is performed on "CV".
  - If the load signal "LOAD" is set to FALSE and the counter input signal "CD" changes from FALSE to TRUE (rising edge), a subtraction operation is performed on "CV". When the value of "CV" is less than 0, even if "CD" is at a rising edge, no subtraction operation is performed on "CV", and the value of "Q" becomes TRUE.
- **Program example**
  - When "LOAD" is set to TRUE, "PV" is loaded into "CV", and "Q" becomes FALSE. After "LOAD" becomes FALSE, "CV" starts to be decremented, "CV" becomes 0, and "Q" becomes TRUE.
  - The following example sets the preset value "PV" to 5.

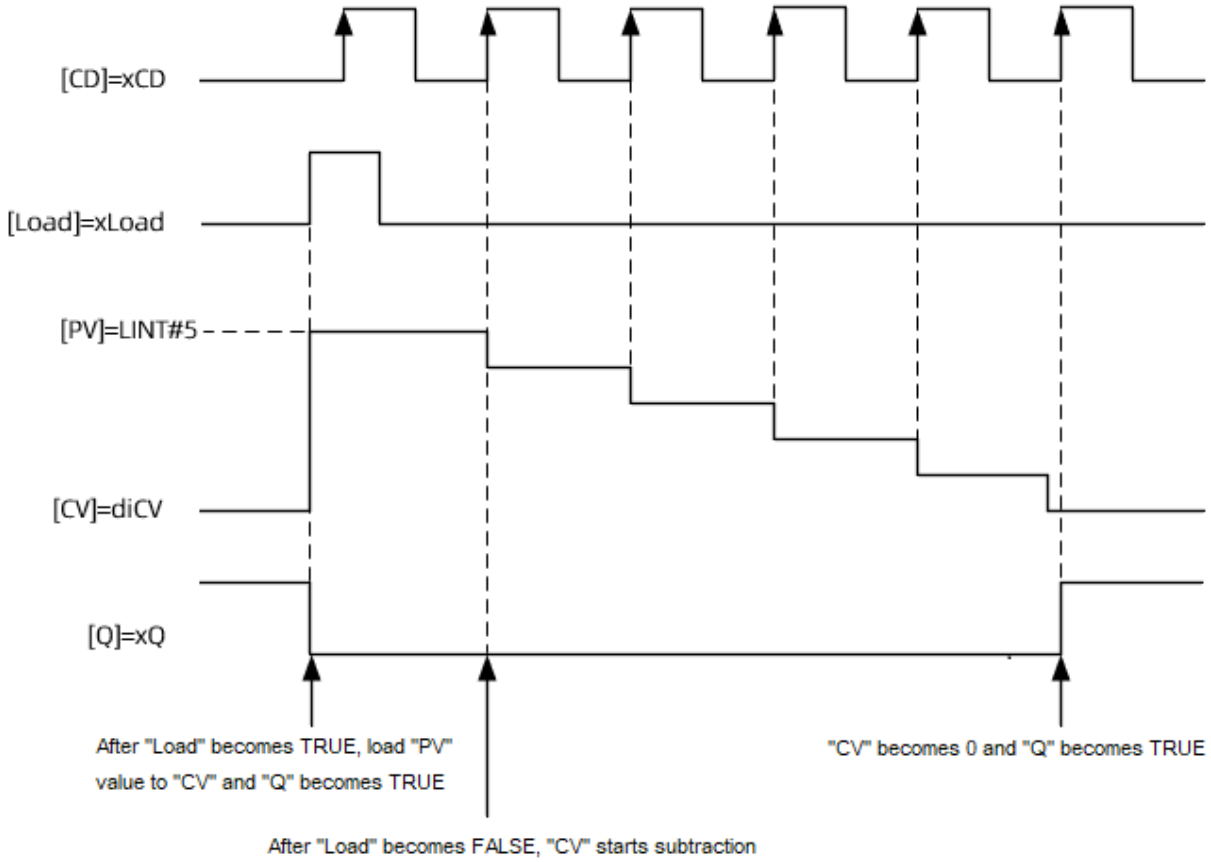
	LD	ST
Define variables		
Program		

```
VAR
    CTD_LINT_Instance_0: CTD_LINT;
    xCD: BOOL;
    xLOAD: BOOL;
    xPV: LINT := 5;
    xQ: BOOL;
    xCV: LINT;
END_VAR
```




```
CTD_LINT_Instance_0(
CD  FALSE  := xCD  FALSE ,
LOAD  FALSE  := xLOAD  FALSE ,
PV  5  := xPV  5 ,
Q  TRUE  => xQ  TRUE ,
CV  0  => xCV  0 );RETURN
```

  - When the preset value "PV" is set to 5, the timing diagram is shown in the figure below.



18.1.8.12 CTD_ULINT

A counter that performs subtraction on the count value each time the counter input signal is received. The data type of the preset value and count value is ULINT.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
CTD_ULINT	Subtraction counter	FB	CTD_ULINT_Instance_0 CTD_ULINT EN ENO CD Q LOAD CV PV	CTD_ULINT_0(CD := , LOAD := , PV := , Q => , CV =>);

- **Related variable**

Variable

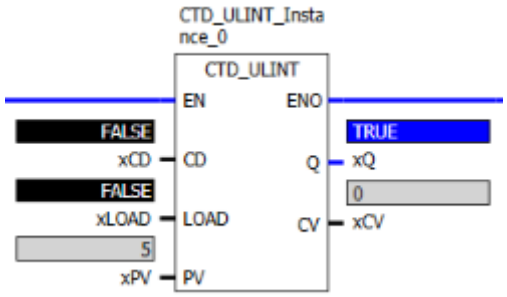
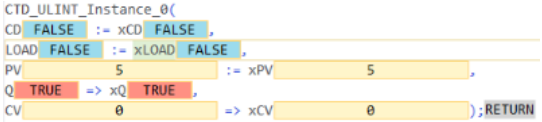
Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	CD	Counter input	Follows data type	-	Counter input

Input/Output	Variable	Name	Valid Range	Default Value	Description
	LOAD	Load signal	Follows data type	-	TRUE: Loads "PV" into "CV".
	PV	Preset value	Follows data type	-	Preset value of the counter
Output	Q	Counter output	Follows data type	-	TRUE: Counter output is ON. FALSE: Counter output is OFF.
	CV	Count value	Follows data type	-	The current value of the counter

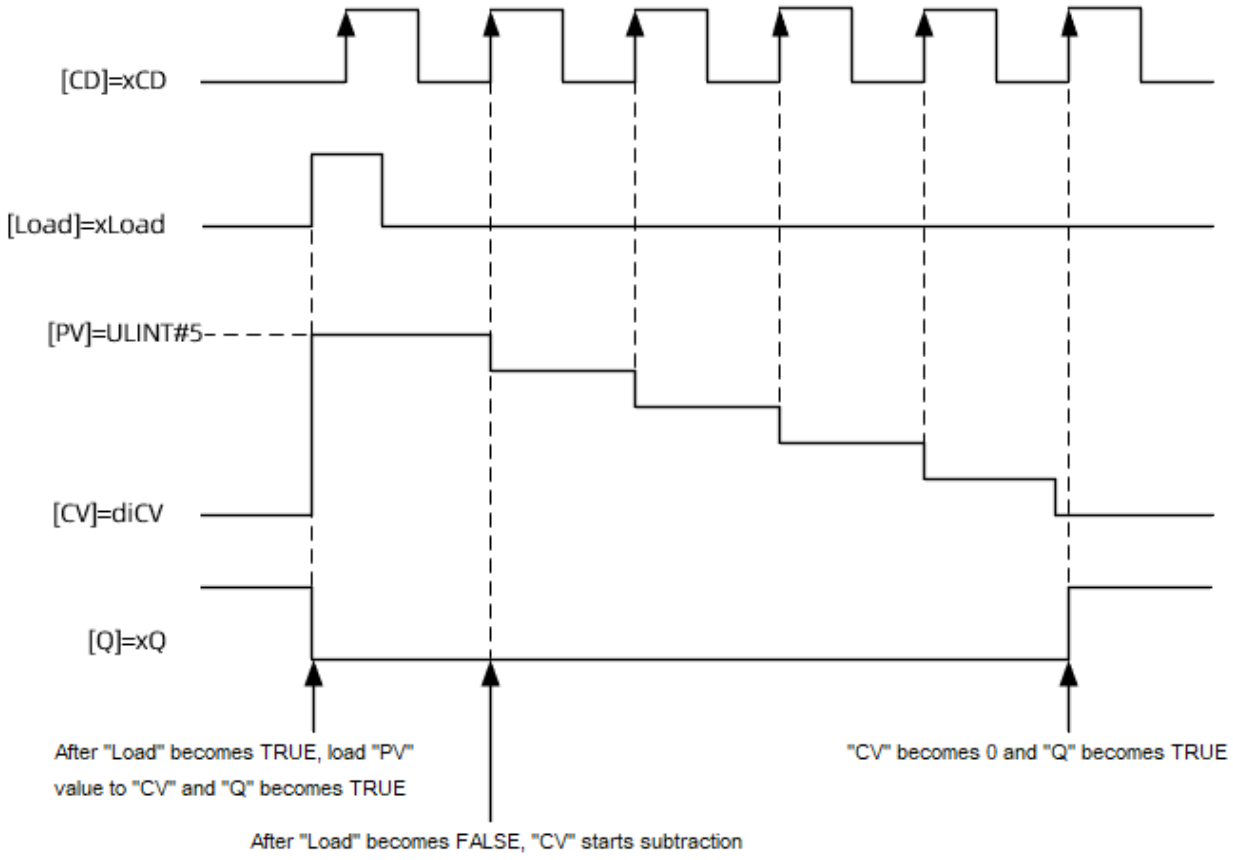
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
CU	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
LOAD	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PV	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CV	Same data type as "PV"																			

- **Function description**
  - If the load signal "LOAD" is set to TRUE, the preset value "PV" is loaded into the count value "CV", and the counter output "Q" becomes FALSE. At this time, the rising edge of "CD" is ignored, and no subtraction operation is performed on "CV".
  - If the load signal "LOAD" is set to FALSE and the counter input signal "CD" changes from FALSE to TRUE (rising edge), a subtraction operation is performed on "CV". When the value of "CV" is less than 0, even if "CD" is at a rising edge, no subtraction operation is performed on "CV", and the value of "Q" becomes TRUE.
- **Program example**
  - When "LOAD" is set to TRUE, "PV" is loaded into "CV", and "Q" becomes FALSE. After "LOAD" becomes FALSE, "CV" starts to be decremented, "CV" becomes 0, and "Q" becomes TRUE.
  - The following example sets the preset value "PV" to 5.

	LD	ST
Define variables	VAR CTD_ULINT_Instance_0: CTD_ULINT; xCD: BOOL; xLOAD: BOOL; xPV: ULINT := 5; xQ: BOOL; xCV: ULINT; END_VAR	
Program		

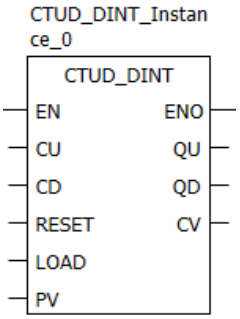
◦ When the preset value "PV" is set to 5, the timing diagram is shown in the figure below.



18.1.8.13 CTUD_DINT

A counter that performs addition and subtraction operations based on the addition/subtraction input signals. The data type of the preset value and count value is DINT.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
CTUD_DINT	Reversible counter	FB		CTUD_DINT_Instance_0(CU := , CD := , RESET := , LOAD := , PV := , QU => , QD => , CV =>);

- **Related variable**

Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	CU	Addition counter input	Follows data type	-	Addition counter input
	CD	Subtraction counter input	Follows data type	-	Subtraction counter input
	RESET	Reset signal	Follows data type	-	TRUE: Resets "CV" to 0.
	LOAD	Load signal	Follows data type	-	TRUE: Loads "PV" into "CV".
	PV	Preset value	Follows data type	-	Preset value of the counter
Output	QU	Addition counter output	Follows data type	-	TRUE: Addition counter output is ON. FALSE: Addition counter output is OFF.
	QD	Subtraction counter output	Follows data type	-	TRUE: Subtraction counter output is ON. FALSE: Subtraction counter output is OFF.
	CV	Count value	Follows data type	-	The current value of the counter

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
CU	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CD	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
LOAD	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PV	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
QU	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
QD	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CV	Same data type as "PV"																			

- **Function description**

  A counter that performs addition and subtraction operations based on the addition/subtraction input signals. It combines the functions of both the addition counter and the subtraction counter. The data type of the preset value and the counter value is DINT.

  ◦ Addition counter function

  When the reset signal "RESET" is set to TRUE, the counter value "CV" becomes 0, and the addition counter output "QU" becomes FALSE. When addition counter input signal "CU" changes from FALSE to TRUE (rising edge), this instruction is executed, and "CV" is incremented by "1". Upon each rising-edge signal of "CU", "CV" is incremented, until it reaches the maximum value of the specified data type.

  ◦ Subtraction counter function

  When the load signal "LOAD" is set to TRUE, the preset value "PV" is loaded into the count value "CV", and the subtraction counter output "QD" becomes FALSE. When the subtraction counter input signal "CD" changes from FALSE to TRUE (rising edge), "CV" is decremented. When the value of "CV" is less than 0, the value of "QD" becomes TRUE. When the value of "CV" is less than 0, even if a larger value of "CD" is input, "CV" remains unchanged.

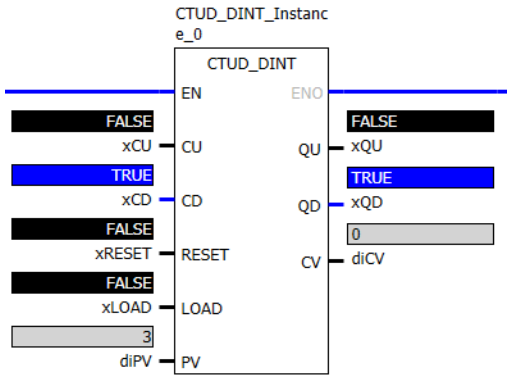
  ◦ Common functions of the addition/subtraction counter

  When "LOAD" or "RESET" is TRUE, "CU" and "CD" are ignored. No addition or subtraction operation is performed on "CV". When "CU" and "CD" are both at the rising edge, "CV" does not change. When both "RESET" and "LOAD" are TRUE, "RESET" takes priority, and the value of "CV" becomes 0. When "RESET" is set to TRUE, the value of "CV" becomes 0, and the value of "QD" becomes TRUE. When "LOAD" is set to TRUE, the value of "CV" becomes equal to "PV", and the value of "QU" becomes TRUE.

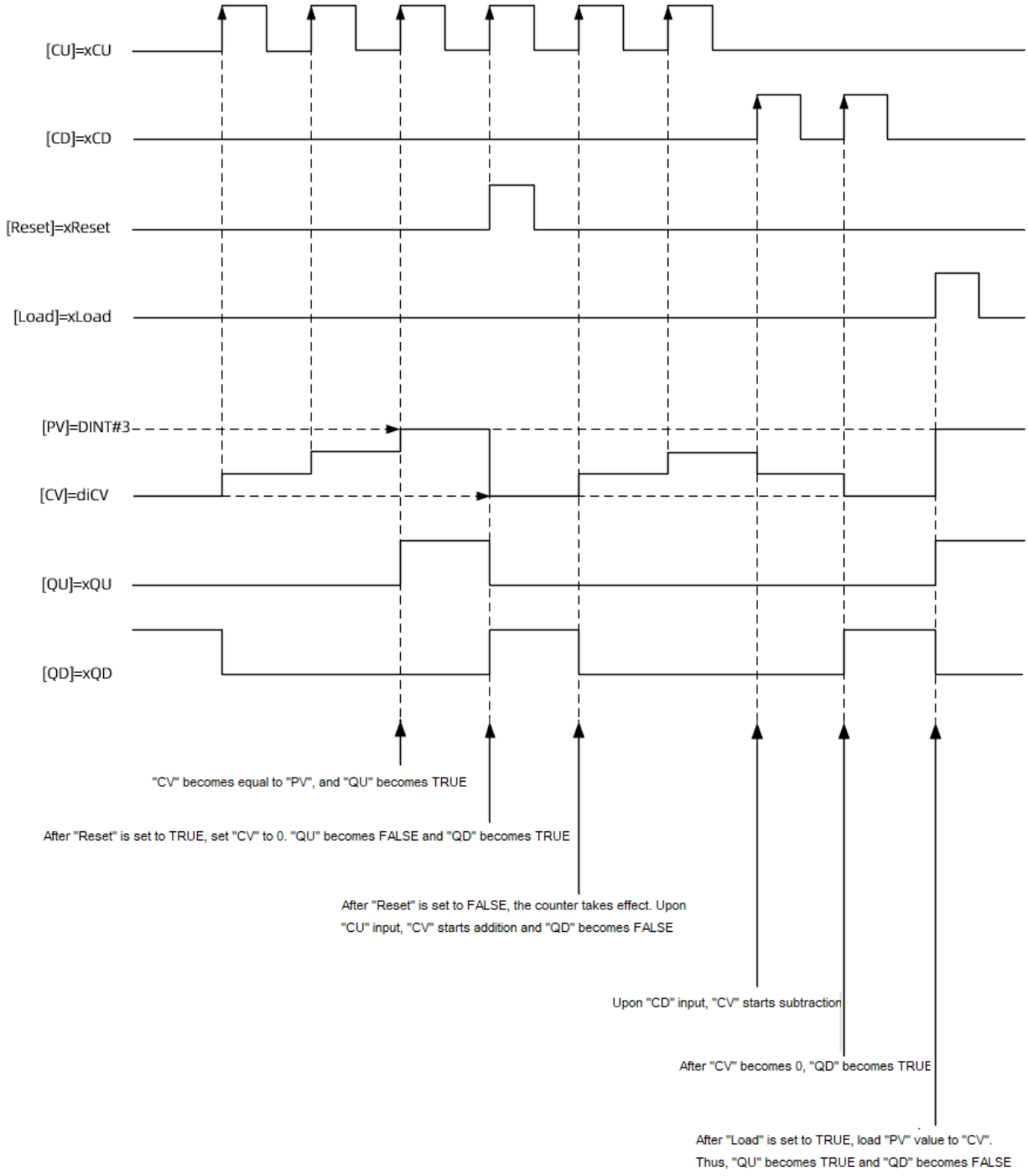
Reset	Load	CV	QU	QD	Action
FALSE	FALSE	Lower than 0	FALSE	TRUE	Acts only as an addition counter. When "CU" is at a rising edge, "CV" performs addition. When "CD" is at a rising edge, no subtraction is performed.
		Between 0 and "PV"	FALSE	FALSE	Acts as both addition/subtraction counter. When "CU" is at a rising edge, an addition operation is performed on "CV"; when "CD" is at a rising edge, a subtraction operation is performed on "CV".
		Greater than "PV"	TRUE	FALSE	Acts only as a subtraction counter. When "CD" is at a rising edge, "CV" performs subtraction. When "CU" is at the rising edge, no addition operation is performed.
TRUE	FALSE	0	FALSE	TRUE	Reset status of the addition counter. The value of "CV" is set to 0.
FALSE	TRUE	"PV"	TRUE	FALSE	Reset status of the subtraction counter. The value of "CV" is set to "PV".
TRUE	TRUE	0	FALSE	TRUE	Reset status of the addition counter. "RESET" takes priority over "LOAD", and the value of "CV" is set to 0.

- **Program example**
  - The following example sets the preset value "PV" to 3.

	LD	ST
Define variables	VAR CTUD_DINT_Instance_0: CTUD_DINT; xCU: BOOL; xCD: BOOL; xRESET: BOOL; xLOAD: BOOL; diPV: DINT := 3; xQU: BOOL; xQD: BOOL; diCV : DINT; END_VAR	

	LD	ST
Program		CTUD_DINT_Instance_0(CU FALSE := xCU FALSE , CD TRUE := xCD TRUE , RESET FALSE := xRESET FALSE , LOAD FALSE := xLOAD FALSE , PV 3 := diPV 3 , QU FALSE => xQU FALSE , QD TRUE => xQD TRUE , CV 0 => diCV 0);RETURN

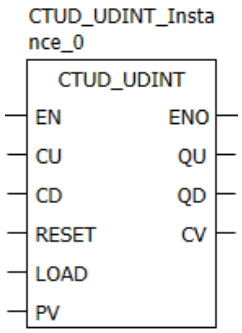
- When the preset value "PV" is set to 3, the timing diagram is shown in the figure below.
  - When the input signal "CU" changes from FALSE to TRUE (rising edge), "CV" is incremented. When "CV" becomes equal to the preset value "PV", "QU" becomes TRUE.
  - When "RESET" is set to TRUE, "CV" is reset to 0, "QU" becomes FALSE, and "QD" becomes TRUE.
  - After "RESET" becomes FALSE, "CV" starts to be incremented and "QD" becomes TRUE. "QU" becomes TRUE when "CV" reaches the value of "PV".
  - When the input signal "CD" changes from FALSE to TRUE (rising edge), "CV" is decremented. When the value of "CV" becomes 0, the value of "QD" becomes TRUE.
  - When "LOAD" is set to TRUE, "PV" is loaded into "CV", "QD" becomes FALSE, and "QU" becomes TRUE.
  - After "LOAD" becomes FALSE, "CV" starts to be decremented and "QU" becomes FALSE. When "CV" becomes 0, "QD" becomes TRUE.



18.1.8.14 CTUD_UDINT

A counter that performs addition and subtraction operations based on the addition/subtraction input signals. The data type of the preset value and count value is UDINT.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
CTUD_UDINT	Reversible counter	FB		CTUD_UDINT_Instance_0(CU := , CD := , RESET := , LOAD := , PV := , QU => , QD => , CV =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	CU	Addition counter input	Follows data type	-	Addition counter input
	CD	Subtraction counter input	Follows data type	-	Subtraction counter input
	RESET	Reset signal	Follows data type	-	TRUE: Resets "CV" to 0.
	LOAD	Load signal	Follows data type	-	TRUE: Loads "PV" into "CV".
	PV	Preset value	Follows data type	-	Preset value of the counter
Output	QU	Addition counter output	Follows data type	-	TRUE: Addition counter output is ON. FALSE: Addition counter output is OFF.
	QD	Subtraction counter output	Follows data type	-	TRUE: Subtraction counter output is ON. FALSE: Subtraction counter output is OFF.

Input/Output	Variable	Name	Valid Range	Default Value	Description
	CV	Count value	Follows data type	-	The current value of the counter

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
CU	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CD	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
LOAD	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PV	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–
QU	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
QD	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CV	Same data type as "PV"																			

- **Function description**

  A counter that performs addition and subtraction operations based on the addition/subtraction input signals. It combines the functions of both the addition counter and the subtraction counter. The data type of the preset value and the count value is UDINT.

  ◦ Addition counter function

  When the reset signal "RESET" is set to TRUE, the counter value "CV" becomes 0, and the addition counter output "QU" becomes FALSE. When addition counter input signal "CU" changes from FALSE to TRUE (rising edge), this instruction is executed, and "CV" is incremented by "1". Upon each rising-edge signal of "CU", "CV" is incremented, until it reaches the maximum value of the specified data type.

  ◦ Subtraction counter function

  When the load signal "LOAD" is set to TRUE, the preset value "PV" is loaded into the count value "CV", and the subtraction counter output "QD" becomes FALSE. When the subtraction counter input signal "CD" changes from FALSE to TRUE (rising edge), "CV" is decremented. When the value of "CV" is less than 0, the value of "QD" becomes TRUE. When the value of "CV" is less than 0, even if a larger value of "CD" is input, "CV" remains unchanged.

  ◦ Common functions of the addition/subtraction counter

  When "LOAD" or "RESET" is TRUE, "CU" and "CD" are ignored. No addition or subtraction operation is performed on "CV". When "CU" and "CD" are both at the rising edge, "CV" does not change. When both "RESET" and "LOAD" are TRUE, "RESET" takes priority, and the value of "CV" becomes 0. When "RESET" is

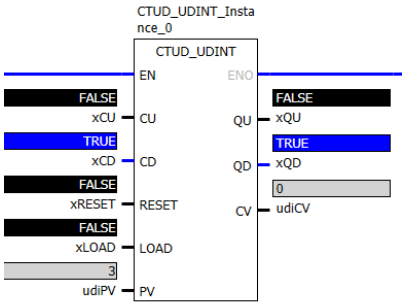
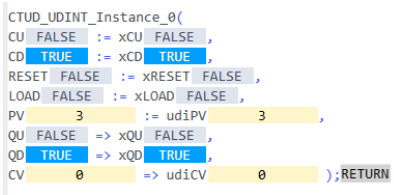
set to TRUE, the value of "CV" becomes 0, and the value of "QD" becomes TRUE. When "LOAD" is set to TRUE, the value of "CV" becomes equal to "PV", and the value of "QU" becomes TRUE.

Reset	Load	CV	QU	QD	Action
FALSE	FALSE	Lower than 0	FALSE	TRUE	Acts only as an addition counter. When "CU" is at a rising edge, "CV" performs addition. When "CD" is at a rising edge, no subtraction is performed.
		Between 0 and "PV"	FALSE	FALSE	Acts as both addition/ subtraction counter. When "CU" is at a rising edge, an addition operation is performed on "CV"; when "CD" is at a rising edge, a subtraction operation is performed on "CV".
		Greater than "PV"	TRUE	FALSE	Acts only as a subtraction counter. When "CD" is at a rising edge, "CV" performs subtraction. When "CU" is at the rising edge, no addition operation is performed.

Reset	Load	CV	QU	QD	Action
TRUE	FALSE	0	FALSE	TRUE	Reset status of the addition counter. The value of "CV" is set to 0.
FALSE	TRUE	"PV"	TRUE	FALSE	Reset status of the subtraction counter. The value of "CV" is set to "PV".
TRUE	TRUE	0	FALSE	TRUE	Reset status of the addition counter. "RESET" takes priority over "LOAD", and the value of "CV" is set to 0.

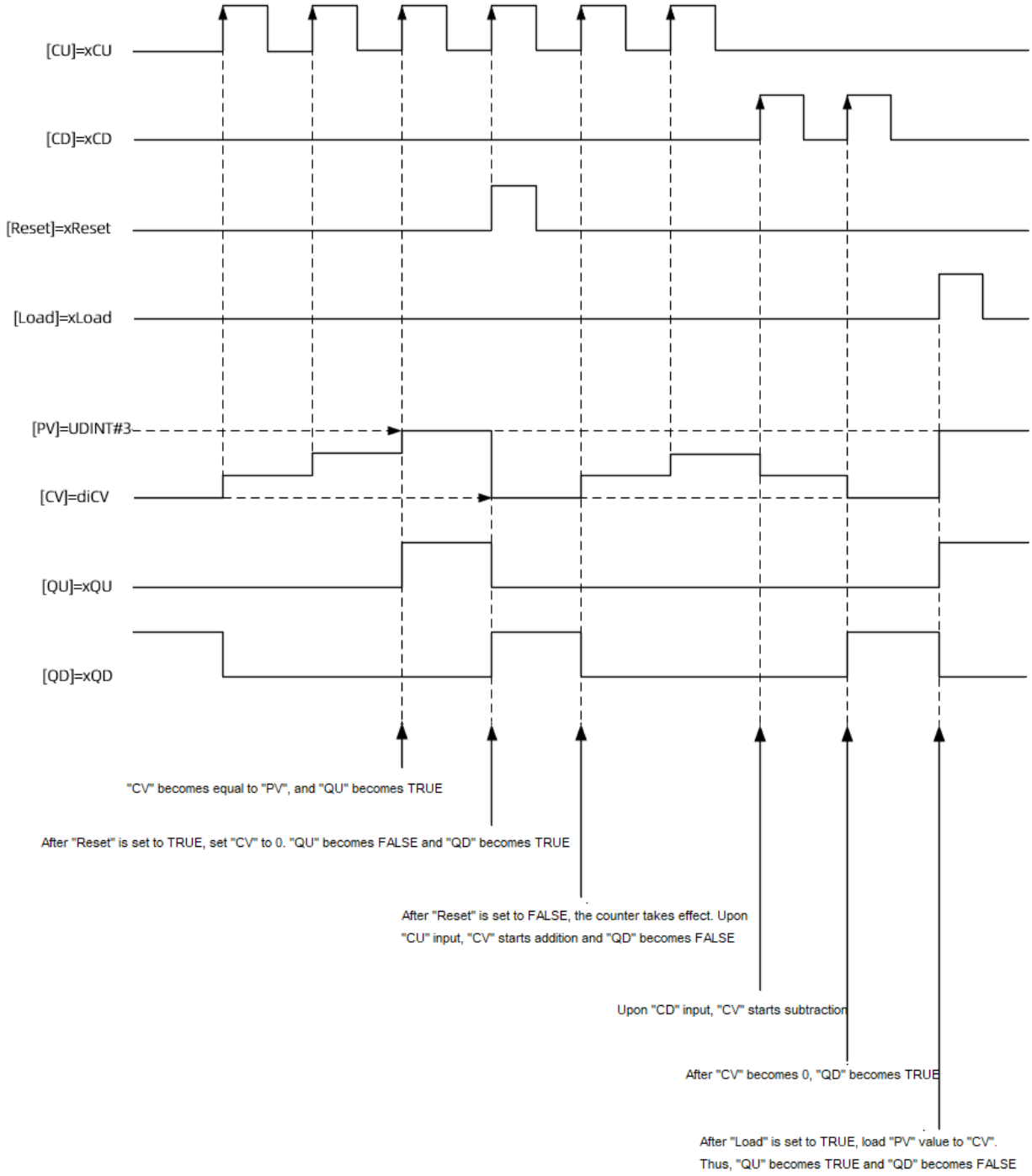
- **Program example**
  - The following example sets the preset value "PV" to 3.

	LD	ST
Define variables	VAR CTUD_UDINT_Instance_0: CTUD_UDINT; xCU: BOOL; xCD: BOOL; xRESET: BOOL; xLOAD: BOOL; udiPV: UDINT := 3; xQU: BOOL; xQD: BOOL; udiCV : UDINT; END_VAR	

	LD	ST
Program		

◦ When the preset value "PV" is set to 3, the timing diagram is shown in the figure below.

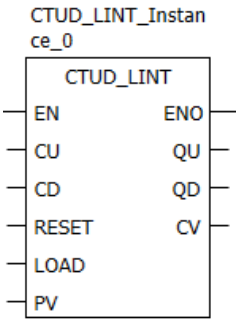
- When the input signal "CU" changes from FALSE to TRUE (rising edge), "CV" is incremented. When "CV" becomes equal to the preset value "PV", "QU" becomes TRUE.
- When "RESET" is set to TRUE, "CV" is reset to 0, "QU" becomes FALSE, and "QD" becomes TRUE.
- After "RESET" becomes FALSE, "CV" starts to be incremented and "QD" becomes TRUE. "QU" becomes TRUE when "CV" reaches the value of "PV".
- When the input signal "CD" changes from FALSE to TRUE (rising edge), "CV" is decremented. When the value of "CV" becomes 0, the value of "QD" becomes TRUE.
- When "LOAD" is set to TRUE, "PV" is loaded into "CV", "QD" becomes FALSE, and "QU" becomes TRUE.
- After "LOAD" becomes FALSE, "CV" starts to be decremented and "QU" becomes FALSE. When "CV" becomes 0, "QD" becomes TRUE.



18.1.8.15 CTUD_LINT

A counter that performs addition and subtraction operations based on the addition/subtraction input signals. The data type of the preset value and count value is LINT.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
CTUD_LINT	Reversible counter	FB		CTUD_LINT_Instance_0(CU := , CD := , RESET := , LOAD := , PV := , QU => , QD => , CV =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	CU	Addition counter input	Follows data type	-	Addition counter input
	CD	Subtraction counter input	Follows data type	-	Subtraction counter input
	RESET	Reset signal	Follows data type	-	TRUE: Resets "CV" to 0.
	LOAD	Load signal	Follows data type	-	TRUE: Loads "PV" into "CV".
	PV	Preset value	Follows data type	-	Preset value of the counter
Output	QU	Addition counter output	Follows data type	-	TRUE: Addition counter output is ON. FALSE: Addition counter output is OFF.
	QD	Subtraction counter output	Follows data type	-	TRUE: Subtraction counter output is ON. FALSE: Subtraction counter output is OFF.

Input/Output	Variable	Name	Valid Range	Default Value	Description
	CV	Count value	Follows data type	-	The current value of the counter

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
CU	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CD	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
LOAD	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PV	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–
QU	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
QD	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CV	Same data type as "PV"																			

- **Function description**

  A counter that performs addition and subtraction operations based on the addition/subtraction input signals. It combines the functions of both the addition counter and the subtraction counter. The data type of the preset value and counter value is LINT.

  ◦ Addition counter function

  When the reset signal "RESET" is set to TRUE, the counter value "CV" becomes 0, and the addition counter output "QU" becomes FALSE. When addition counter input signal "CU" changes from FALSE to TRUE (rising edge), this instruction is executed, and "CV" is incremented by "1". Upon each rising-edge signal of "CU", "CV" is incremented, until it reaches the maximum value of the specified data type.

  ◦ Subtraction counter function

  When the load signal "LOAD" is set to TRUE, the preset value "PV" is loaded into the count value "CV", and the subtraction counter output "QD" becomes FALSE. When the subtraction counter input signal "CD" changes from FALSE to TRUE (rising edge), "CV" is decremented. When the value of "CV" is less than 0, the value of "QD" becomes TRUE. When the value of "CV" is less than 0, even if a larger value of "CD" is input, "CV" remains unchanged.

  ◦ Common functions of the addition/subtraction counter

  When "LOAD" or "RESET" is TRUE, "CU" and "CD" are ignored. No addition or subtraction operation is performed on "CV". When "CU" and "CD" are both at the rising edge, "CV" does not change. When both

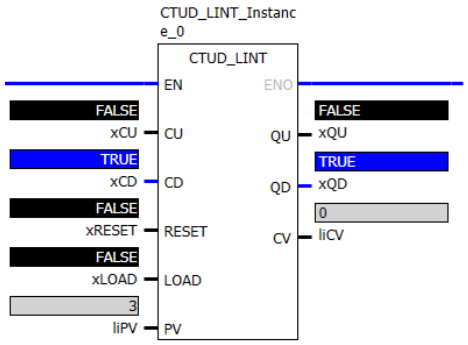
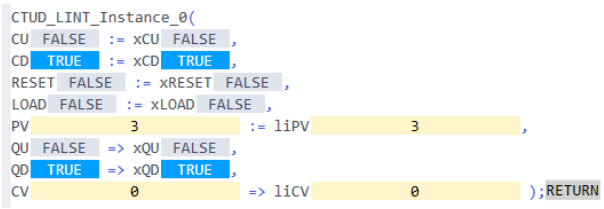
"RESET" and "LOAD" are TRUE, "RESET" takes priority, and the value of "CV" becomes 0. When "RESET" is set to TRUE, the value of "CV" becomes 0, and the value of "QD" becomes TRUE. When "LOAD" is set to TRUE, the value of "CV" becomes equal to "PV", and the value of "QU" becomes TRUE.

Reset	Load	CV	QU	QD	Action
FALSE	FALSE	Lower than 0	FALSE	TRUE	Acts only as an addition counter. When "CU" is at a rising edge, "CV" performs addition. When "CD" is at a rising edge, no subtraction is performed.
		Between 0 and "PV"	FALSE	FALSE	Acts as both addition/ subtraction counter. When "CU" is at a rising edge, an addition operation is performed on "CV"; when "CD" is at a rising edge, a subtraction operation is performed on "CV".
		Greater than "PV"	TRUE	FALSE	Acts only as a subtraction counter. When "CD" is at a rising edge, "CV" performs subtraction. When "CU" is at the rising edge, no addition operation is performed.

Reset	Load	CV	QU	QD	Action
TRUE	FALSE	0	FALSE	TRUE	Reset status of the addition counter. The value of "CV" is set to 0.
FALSE	TRUE	"PV"	TRUE	FALSE	Reset status of the subtraction counter. The value of "CV" is set to "PV".
TRUE	TRUE	0	FALSE	TRUE	Reset status of the addition counter. "RESET" takes priority over "LOAD", and the value of "CV" is set to 0.

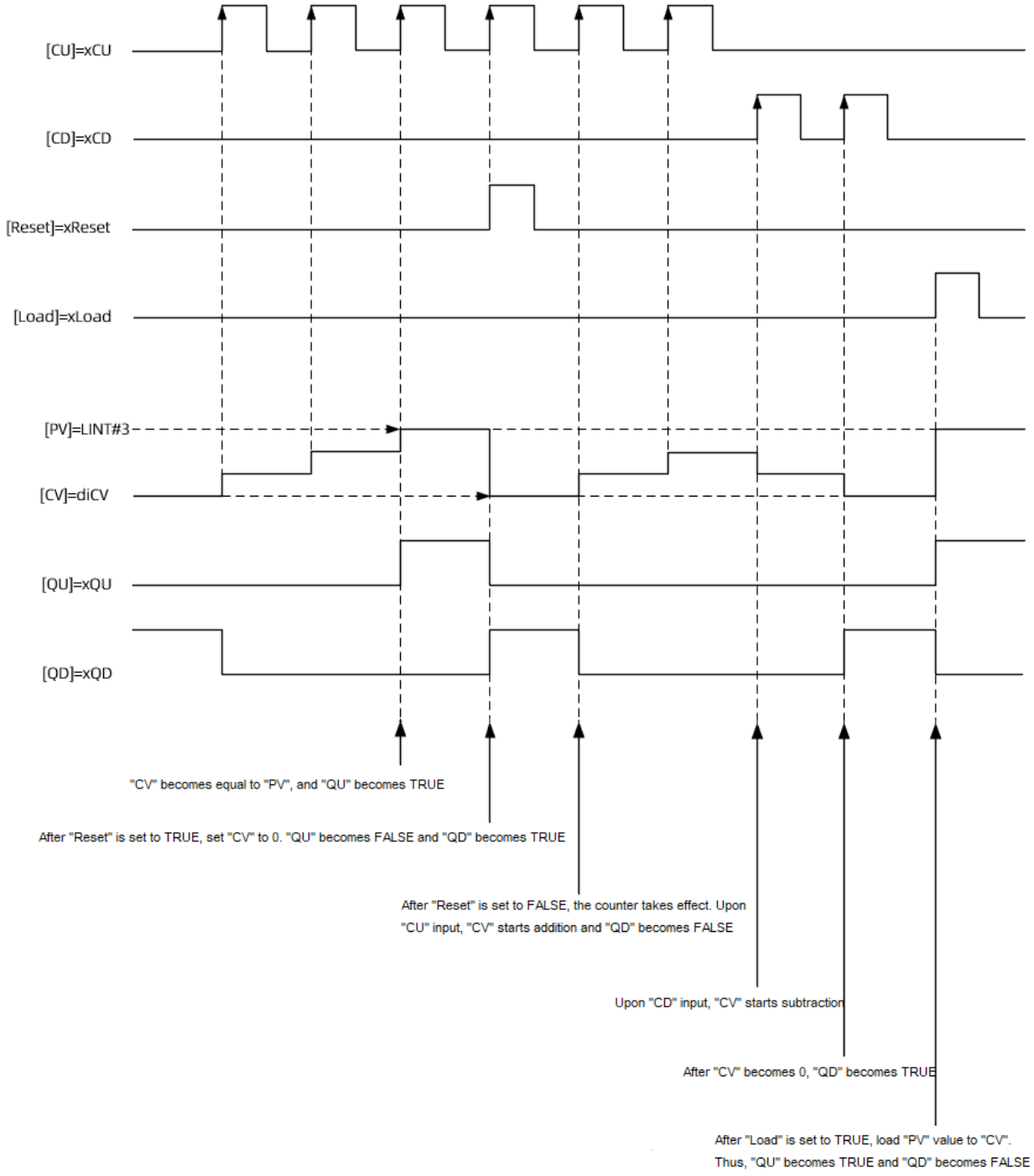
- **Program example**
  - The following example sets the preset value "PV" to 3.

	LD	ST
Define variables	VAR CTUD_LINT_Instance_0: CTUD_LINT; xCU: BOOL; xCD: BOOL; xRESET: BOOL; xLOAD: BOOL; liPV: LINT := 3; xQU: BOOL; xQD: BOOL; liCV : LINT; END_VAR	

	LD	ST
Program		

◦ When the preset value "PV" is set to 3, the timing diagram is shown in the figure below.

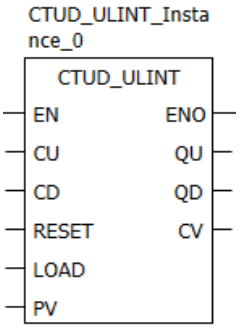
- When the input signal "CU" changes from FALSE to TRUE (rising edge), "CV" is incremented. When "CV" becomes equal to the preset value "PV", "QU" becomes TRUE.
- When "RESET" is set to TRUE, "CV" is reset to 0, "QU" becomes FALSE, and "QD" becomes TRUE.
- After "RESET" becomes FALSE, "CV" starts to be incremented and "QD" becomes TRUE. "QU" becomes TRUE when "CV" reaches the value of "PV".
- When the input signal "CD" changes from FALSE to TRUE (rising edge), "CV" is decremented. When the value of "CV" becomes 0, the value of "QD" becomes TRUE.
- When "LOAD" is set to TRUE, "PV" is loaded into "CV", "QD" becomes FALSE, and "QU" becomes TRUE.
- After "LOAD" becomes FALSE, "CV" starts to be decremented and "QU" becomes FALSE. When "CV" becomes 0, "QD" becomes TRUE.



18.1.8.16 CTUD_ULINT

A counter that performs addition and subtraction operations based on the addition/subtraction input signals. The data type of the preset value and count value is ULINT.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
CTUD_ULINT	Reversible counter	FB		CTUD_ULINT_Instance_0(CU := , CD := , RESET := , LOAD := , PV := , QU => , QD => , CV =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	CU	Addition counter input	Follows data type	-	Addition counter input
	CD	Subtraction counter input	Follows data type	-	Subtraction counter input
	RESET	Reset signal	Follows data type	-	TRUE: Resets "CV" to 0.
	LOAD	Load signal	Follows data type	-	TRUE: Loads "PV" into "CV".
	PV	Preset value	Follows data type	-	Preset value of the counter
Output	QU	Addition counter output	Follows data type	-	TRUE: Addition counter output is ON. FALSE: Addition counter output is OFF.
	QD	Subtraction counter output	Follows data type	-	TRUE: Subtraction counter output is ON. FALSE: Subtraction counter output is OFF.

Input/Output	Variable	Name	Valid Range	Default Value	Description
	CV	Count value	Follows data type	-	The current value of the counter

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
CU	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CD	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
LOAD	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PV	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–
QU	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
QD	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CV	Same data type as "PV"																			

- **Function description**

  A counter that performs addition and subtraction operations based on the addition/subtraction input signals. It combines the functions of both the addition counter and the subtraction counter. The data type of the preset value and counter value is ULINT.

  ◦ Addition counter function

  When the reset signal "RESET" is set to TRUE, the counter value "CV" becomes 0, and the addition counter output "QU" becomes FALSE. When addition counter input signal "CU" changes from FALSE to TRUE (rising edge), this instruction is executed, and "CV" is incremented by "1". Upon each rising-edge signal of "CU", "CV" is incremented, until it reaches the maximum value of the specified data type.

  ◦ Subtraction counter function

  When the load signal "LOAD" is set to TRUE, the preset value "PV" is loaded into the count value "CV", and the subtraction counter output "QD" becomes FALSE. When the subtraction counter input signal "CD" changes from FALSE to TRUE (rising edge), "CV" is decremented. When the value of "CV" is less than 0, the value of "QD" becomes TRUE. When the value of "CV" is less than 0, even if a larger value of "CD" is input, "CV" remains unchanged.

  ◦ Common functions of the addition/subtraction counter

  When "LOAD" or "RESET" is TRUE, "CU" and "CD" are ignored. No addition or subtraction operation is performed on "CV". When "CU" and "CD" are both at the rising edge, "CV" does not change. When both

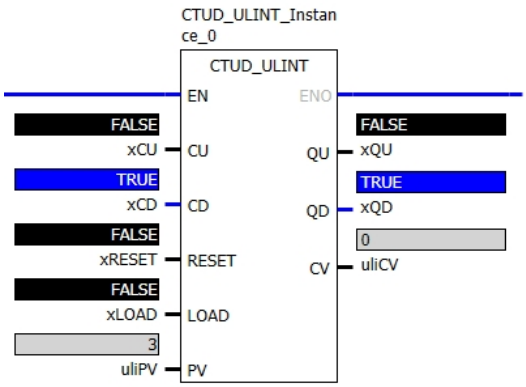
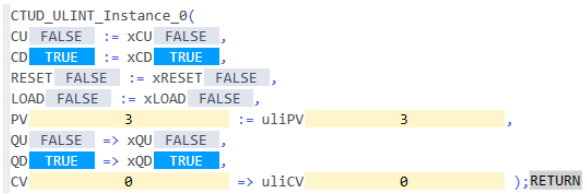
"RESET" and "LOAD" are TRUE, "RESET" takes priority, and the value of "CV" becomes 0. When "RESET" is set to TRUE, the value of "CV" becomes 0, and the value of "QD" becomes TRUE. When "LOAD" is set to TRUE, the value of "CV" becomes equal to "PV", and the value of "QU" becomes TRUE.

Reset	Load	CV	QU	QD	Action
FALSE	FALSE	Lower than 0	FALSE	TRUE	Acts only as an addition counter. When "CU" is at a rising edge, "CV" performs addition. When "CD" is at a rising edge, no subtraction is performed.
		Between 0 and "PV"	FALSE	FALSE	Acts as both addition/ subtraction counter. When "CU" is at a rising edge, an addition operation is performed on "CV"; when "CD" is at a rising edge, a subtraction operation is performed on "CV".
		Greater than "PV"	TRUE	FALSE	Acts only as a subtraction counter. When "CD" is at a rising edge, "CV" performs subtraction. When "CU" is at the rising edge, no addition operation is performed.

Reset	Load	CV	QU	QD	Action
TRUE	FALSE	0	FALSE	TRUE	Reset status of the addition counter. The value of "CV" is set to 0.
FALSE	TRUE	"PV"	TRUE	FALSE	Reset status of the subtraction counter. The value of "CV" is set to "PV".
TRUE	TRUE	0	FALSE	TRUE	Reset status of the addition counter. "RESET" takes priority over "LOAD", and the value of "CV" is set to 0.

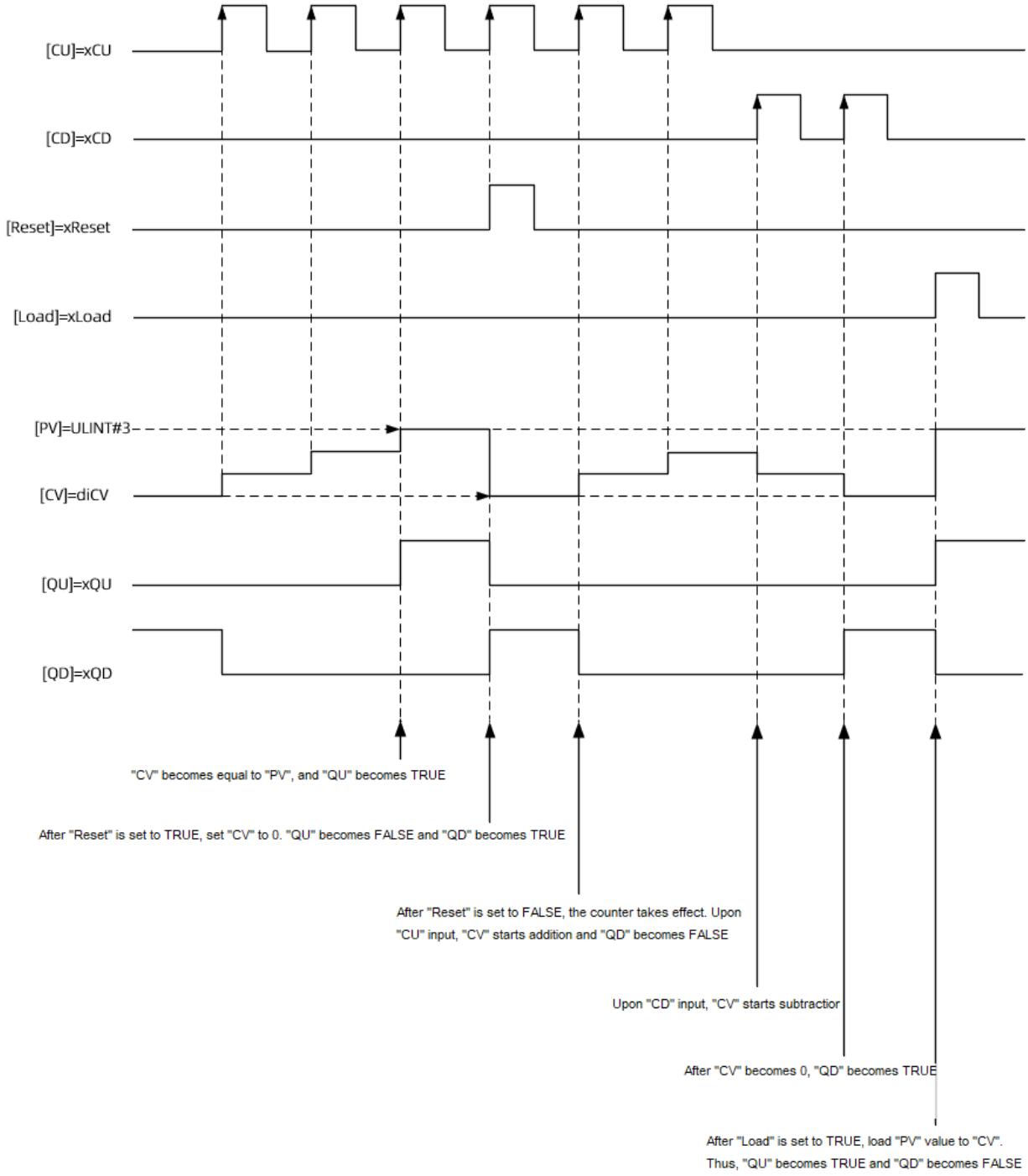
- **Program example**
  - The following example sets the preset value "PV" to 3.

	LD	ST
Define variables	VAR CTUD_ULINT_Instance_0: CTUD_ULINT; xCU: BOOL; xCD: BOOL; xRESET: BOOL; xLOAD: BOOL; uliPV: ULINT := 3; xQU: BOOL; xQD: BOOL; uliCV : ULINT; END_VAR	

	LD	ST
Program		

◦ When the preset value "PV" is set to 3, the timing diagram is shown in the figure below.

- When the input signal "CU" changes from FALSE to TRUE (rising edge), "CV" is incremented. When "CV" becomes equal to the preset value "PV", "QU" becomes TRUE.
- When "RESET" is set to TRUE, "CV" is reset to 0, "QU" becomes FALSE, and "QD" becomes TRUE.
- After "RESET" becomes FALSE, "CV" starts to be incremented and "QD" becomes TRUE. "QU" becomes TRUE when "CV" reaches the value of "PV".
- When the input signal "CD" changes from FALSE to TRUE (rising edge), "CV" is decremented. When the value of "CV" becomes 0, the value of "QD" becomes TRUE.
- When "LOAD" is set to TRUE, "PV" is loaded into "CV", "QD" becomes FALSE, and "QU" becomes TRUE.
- After "LOAD" becomes FALSE, "CV" starts to be decremented and "QU" becomes FALSE. When "CV" becomes 0, "QD" becomes TRUE.



18.1.9 Timer

18.1.9.1 Instruction List

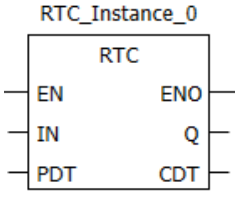
Instruction	FB/FC	Function	Library	Applicable Library Version
18.1.9.2 RTC	FB	Real-time clock	Standard	Library version 1.0.0.0 and later
18.1.9.3 TOF	FB	Power-off delay timer		Library version 1.0.0.0 and later
18.1.9.4 TON	FB	Power-on delay timer		Library version 1.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.1.9.5 TON2	FB	Power-on delay timer	Util	Library version 1.3.0.1 and later
18.1.9.6 TP	FB	Standard timer	Standard	Library version 1.0.0.0 and later
18.1.9.7 LTOF	FB	Power-off delay timer	Standard64	Library version 1.0.0.0 and later
18.1.9.8 LTON	FB	Power-on delay timer		Library version 1.0.0.0 and later
18.1.9.9 LTP	FB	Standard timer		Library version 1.0.0.0 and later
18.1.9.10 MTP	FB	Pulse timing output block	Logic	Library version 1.0.0.0 and later
18.1.9.11 TimeSwitch	FB	Timing control function block		Library version 1.0.0.0 and later
18.1.9.12 AccumulationTimer	FB	Accumulating timer	OMUtils	Library version 1.0.0.0 and later
18.1.9.13 AccumulationTimer2	FB	Accumulating timer	Util	Library version 1.3.0.1 and later
18.1.9.14 Timer	FC	100-ms timer	OMUtils	Library version 1.1.0.1 and earlier
18.1.9.15 Timer100ms	FC	100-ms timer		Library version 1.2.0.0 and later

18.1.9.2 RTC

A clock that starts timing from the set time.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
RTC	Real-time clock	FB		RTC_Instance_0(IN := , PDT := , Q => , CDT =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Timing start signal	Follows data type	-	Timing start signal
	PDT	Timing start time	Follows data type	-	Timing start time
Output	Q	Timing completion signal	Follows data type	-	When "Q" is TRUE, timing is effective.
	CDT	Current timing time	Follows data type	-	Current timing time

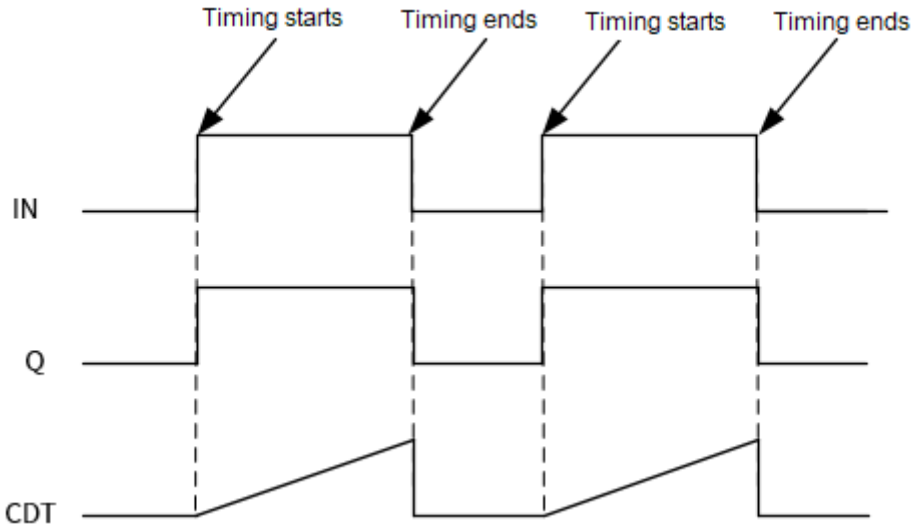
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
PDT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–
IN	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CDT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–

- **Function**

  When input "IN" is TRUE, timing starts from the set time PDT, output "Q" becomes TRUE, and output "CDT" displays the current timing time.

- **Sequence diagram**



- **Program example**

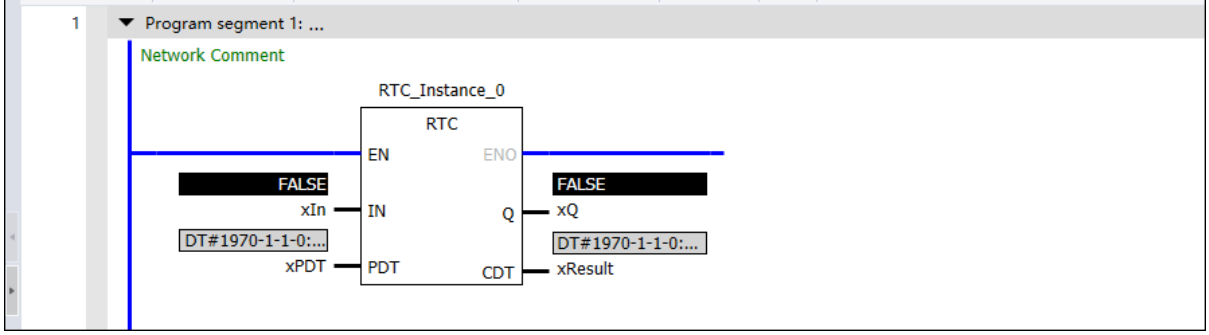
  ST:

		Name	Data type	Address	Value
1		RTC_Instance_0	RTC		
2		xIn	BOOL		TRUE
3		xPDT	DATE_AND_TIME		DT#1970-1-1-0:0:0
4		xQ	BOOL		TRUE
5		xResult	DATE_AND_TIME		DT#1970-1-1-0:12:58

```
1  RTC_Instance_0(
2  IN  TRUE  := xIn  TRUE ,
3  PDT          DT#1970-1-1-0:0:0          := xPDT          DT#1970-1-1-0:0:0          ,
4  Q  TRUE  => xQ  TRUE ,
5  CDT          DT#1970-1-1-0:12:58        => xResult       DT#1970-1-1-0:12:58        );
6  RETURN
```

  LD:

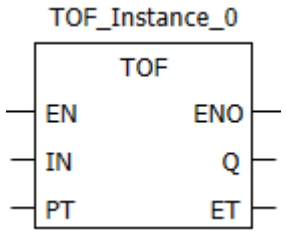
		Name	Data type	Address	Value
1		RTC_Instance_0	RTC		
2		xIn	BOOL		FALSE
3		xPDT	DATE_AND_TIME		DT#1970-1-1-0:0:0
4		xQ	BOOL		FALSE
5		xResult	DATE_AND_TIME		DT#1970-1-1-0:0:0



18.1.9.3 TOF

Outputs a timing completion signal after a delay following power-off.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
TOF	Power-off delay timer	FB		TOF_Instance_0(IN := , PT := , Q => , ET =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input signal	Follows data type	-	Starts timing when "IN" changes from TRUE to FALSE.
	PT	Set timing time	Follows data type	-	Set time
Output	Q	Timing completion signal	Follows data type	-	Timing completes when "Q" changes from TRUE to FALSE.
	ET	Current timing time	Follows data type	-	Current timing time

Variable types

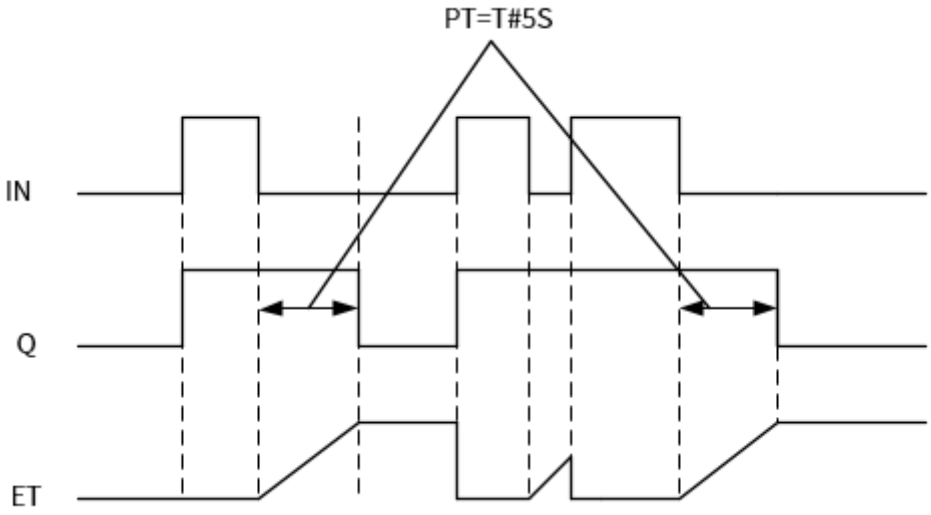
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ET	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–

- **Function**

  The input signal "PT" sets the timing duration. When the input signal "IN" changes from TRUE to FALSE, timing starts. After timing completes, "Q" is set to FALSE.

- **Sequence diagram**



- **Program example**

  ST:

  (1) Input signal "IN" is TRUE

	Name	Data type	Address	Value
1	TOF_Instance_0	TOF		
2	xIN	BOOL		TRUE
3	xPT	TIME		T#5s
4	xQ	BOOL		TRUE
5	xResult	TIME		T#0ms

```
TOF_Instance_0(
IN := xIN,
PT := xPT,
Q => xQ,
ET => xResult);RETURN
```

  (2) Input signal "IN" changes from TRUE to FALSE

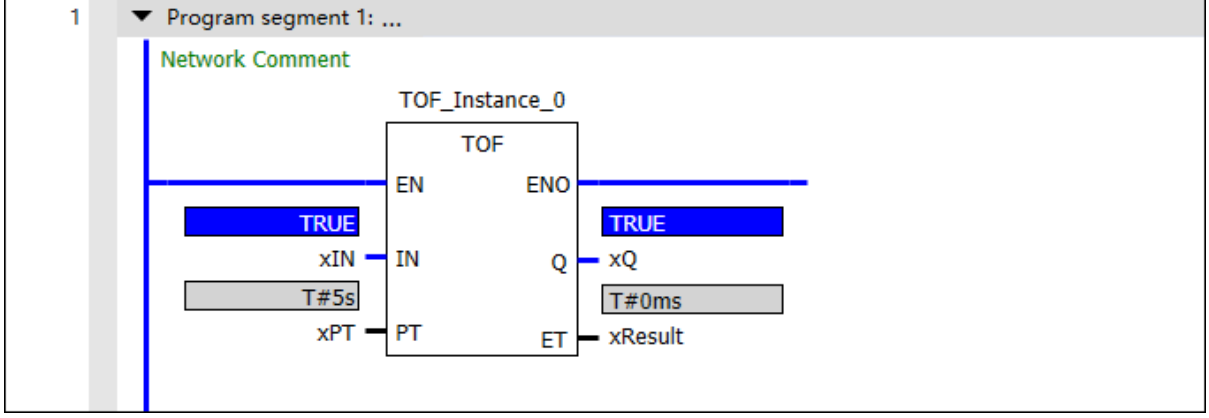
	Name	Data type	Address	Value
1	TOF_Instance_0	TOF		
2	xIN	BOOL		FALSE
3	xPT	TIME		T#5s
4	xQ	BOOL		FALSE
5	xResult	TIME		T#5s

```
TOF_Instance_0(
IN FALSE := xIN FALSE,
PT T#5s := xPT T#5s,
Q FALSE => xQ FALSE,
ET T#5s => xResult T#5s);RETURN
```

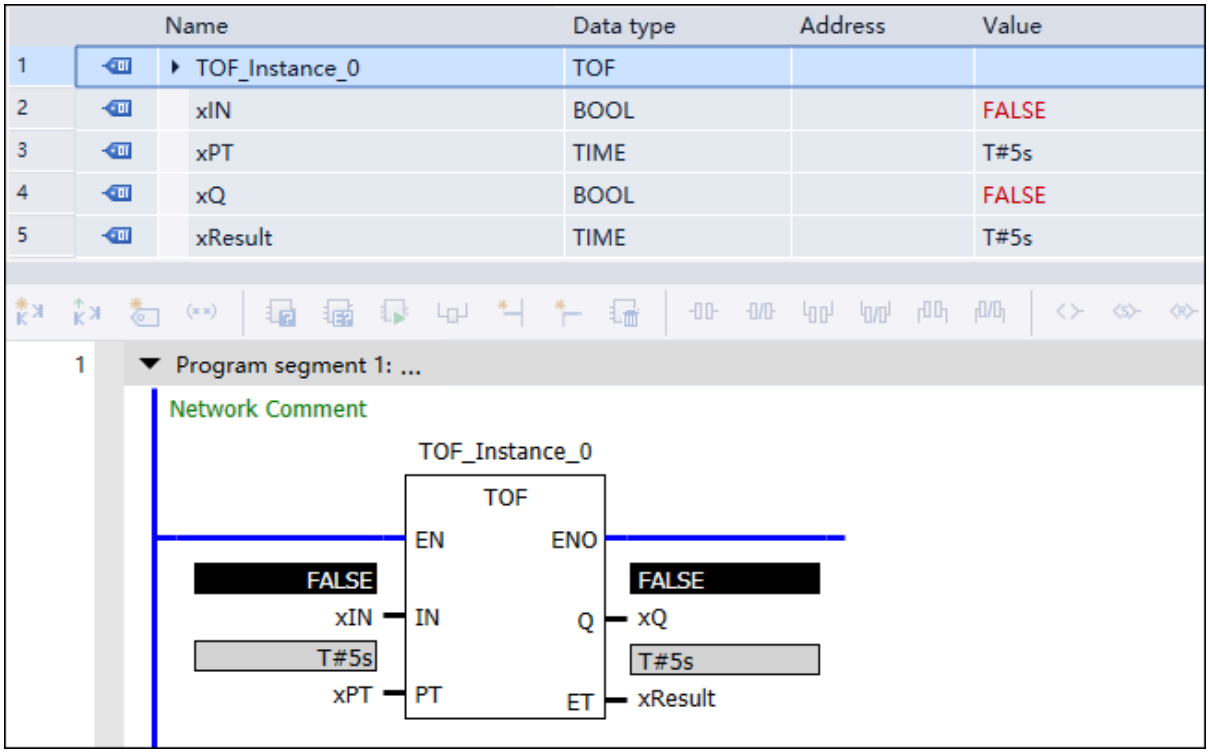
LD:

(1) Input signal "IN" is TRUE

	Name	Data type	Address	Value
1	TOF_Instance_0	TOF		
2	xIN	BOOL		TRUE
3	xPT	TIME		T#5s
4	xQ	BOOL		TRUE
5	xResult	TIME		T#0ms



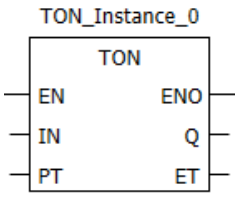
(2) Input signal "IN" changes from TRUE to FALSE



18.1.9.4 TON

Outputs a timing completion signal after a delay following power-on.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
TON	Power-on delay timer	FB		TON_Instance_0(IN := , PT := , Q => , ET =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input signal	Follows data type	-	Starts timing when "IN" changes from FALSE to TRUE.
	PT	Set timing time	Follows data type	-	Set time

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Q	Timing completion signal	Follows data type	-	Timing completes when "Q" changes from FALSE to TRUE.
	ET	Current timing time	Follows data type	-	Current timing time

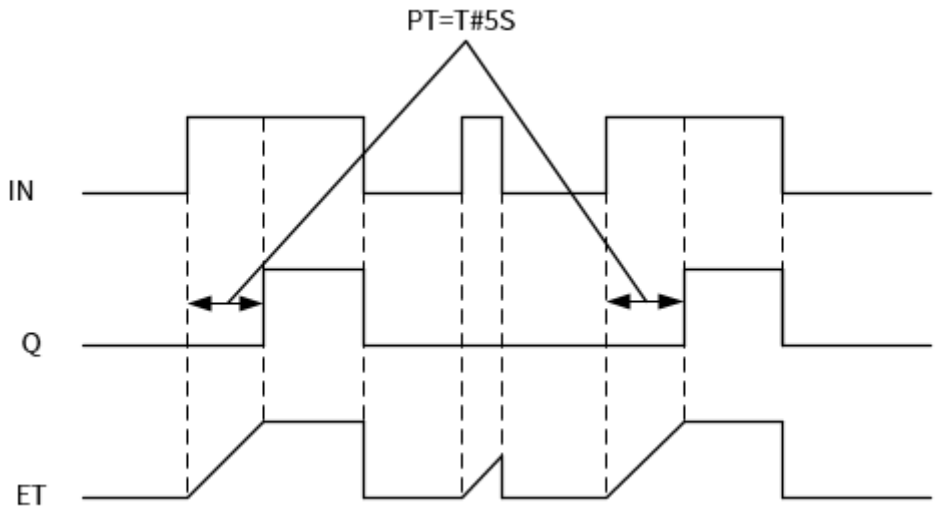
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ET	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–

- **Function**

  Input signal "PT" sets the timing duration. When input signal "IN" changes from FALSE to TRUE, timing starts. After timing completes, "Q" is set to TRUE.

- **Sequence diagram**



- **Program example**

ST:

(1) Input signal "IN" is FALSE

	Name	Data type	Address	Value
1	TON_Instance_0	TON		
2	xIN	BOOL		FALSE
3	xPT	TIME		T#5s
4	xQ	BOOL		FALSE
5	xResult	TIME		T#0ms

```
TON_Instance_0(
IN FALSE := xIN FALSE,
PT T#5s := xPT T#5s,
Q FALSE => xQ FALSE,
ET T#0ms => xResult T#0ms );RETURN
```

(2) When input signal "IN" changes from FALSE to TRUE, "Q" is set to TRUE after "ET" completes timing

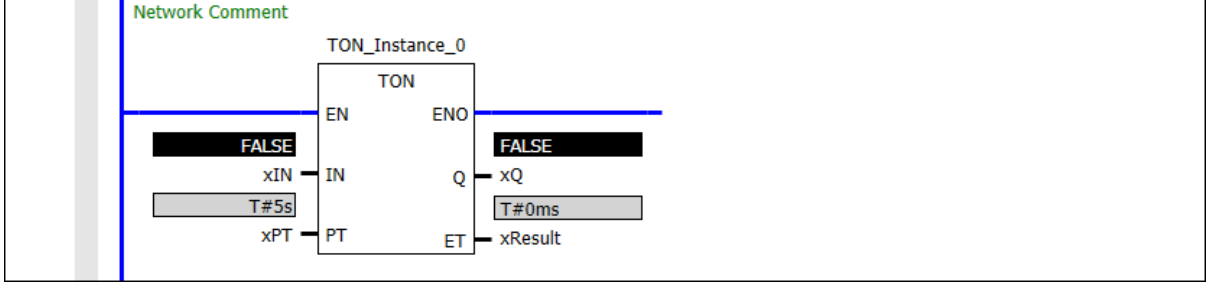
	Name	Data type	Address	Value
1	TON_Instance_0	TON		
2	xIN	BOOL		TRUE
3	xPT	TIME		T#5s
4	xQ	BOOL		TRUE
5	xResult	TIME		T#5s

```
TON_Instance_0(
IN TRUE := xIN TRUE,
PT T#5s := xPT T#5s,
Q TRUE => xQ TRUE,
ET T#5s => xResult T#5s );RETURN
```

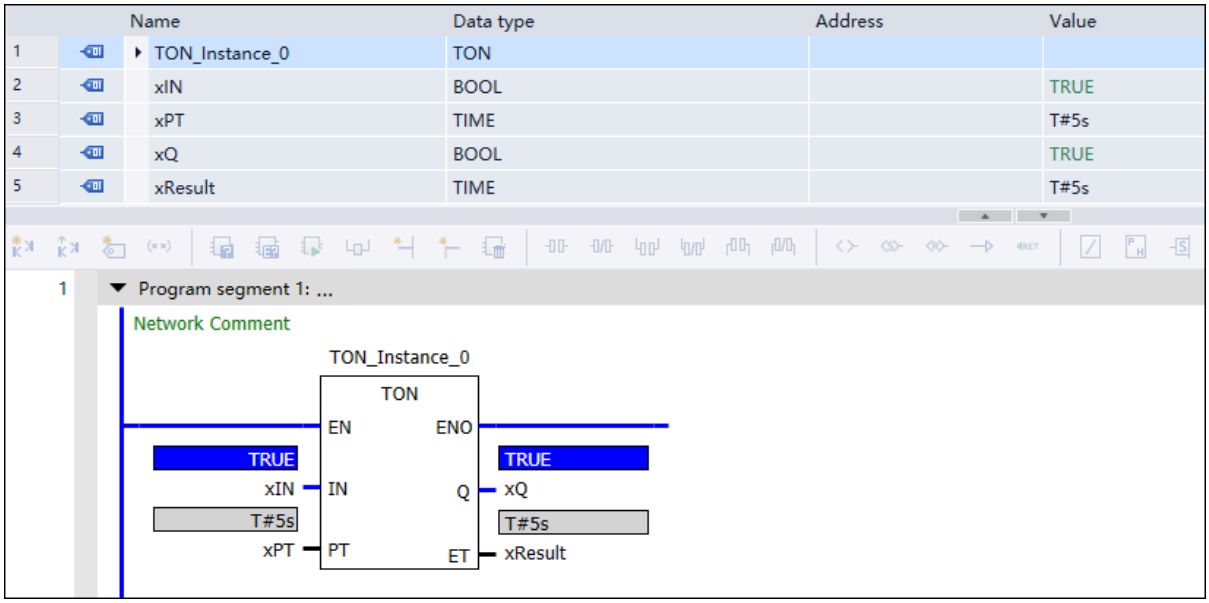
LD:

(1) Input signal "IN" is FALSE

	Name	Data type	Address	Value
1	TON_Instance_0	TON		
2	xIN	BOOL		FALSE
3	xPT	TIME		T#5s
4	xQ	BOOL		FALSE
5	xResult	TIME		T#0ms



(2) When input signal "IN" changes from FALSE to TRUE, "Q" is set to TRUE after "ET" completes timing



18.1.9.5 TON2

On-delay timer. The timer output can be modified during timing.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
TON2	Power-on delay timer	FB	TON2_Instance_0 TON2 IN Q PT TT ET	TON2_Instance_0(IN := , PT := , Q => , TT => , ET =>);

- **Related variables**

Variable

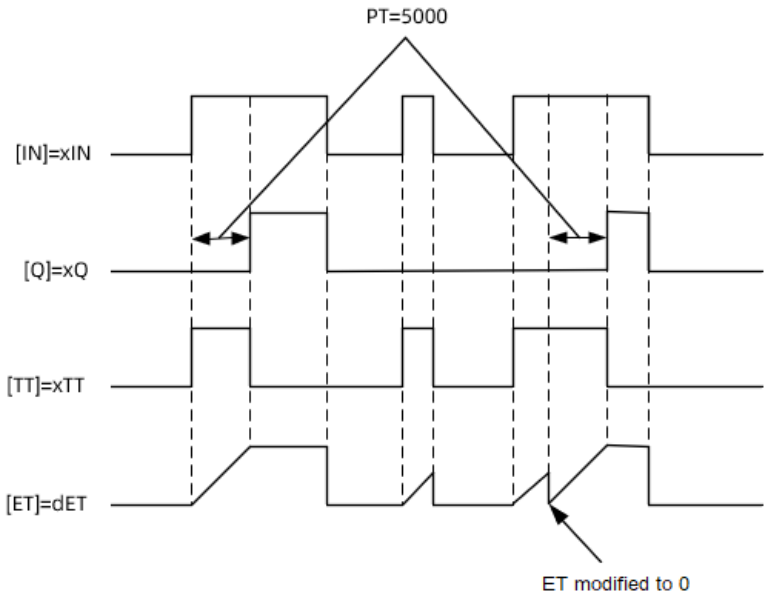
Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input signal	Follows data type.	FALSE	Input signal. Timing starts when FALSE transitions to TRUE.
	PT	Delay time	Follows data type	0	Delay time.
Output	Q	Timing completion signal	Follows data type	FALSE	Timing completion signal TRUE when "ET" is equal to or greater than PT.
	TT	Timing in progress	Follows data type	FALSE	TRUE during timing; FALSE when not timing or when timing is complete.
	ET	Current timing time	Follows data type	0	Timing output. Modifying "ET" during timing takes effect immediately.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PT	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Q	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TT	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ET	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **Function**
  - The input signal "PT" sets the delay time. Timing starts when the input signal "IN" transitions from FALSE to TRUE. During timing, "TT" is set to TRUE, and the accumulated time "ET" increases over time. When "ET" reaches the set time "PT", timing is complete, and "Q" is set to TRUE.
  - The timer resets when "IN" transitions from TRUE to FALSE. "ET" becomes 0, and "TT" and "Q" become FALSE.
  - If the timer is active and the input signal "IN" transitions to FALSE before "ET" reaches "PT", the timer is reset.
  - When modifying the "ET" parameter, the internal "ET" timing value must be forcibly written via the Set accessor of the ModifyET property.
- **Timing diagram**

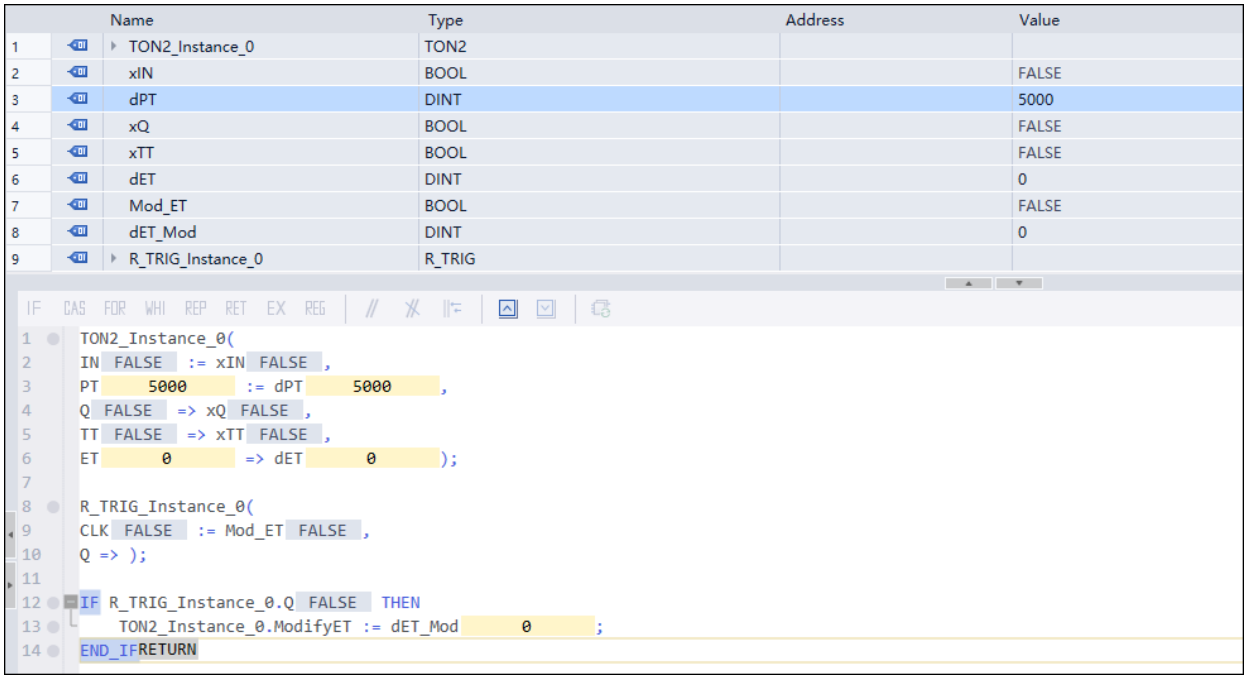
  Set "PT" to 5000. When the cumulative time for variable "IN" being TRUE reaches 5000 ms, "Q" becomes TRUE. If "IN" is set to FALSE before the cumulative time reaches "PT", "ET" and "TT" are reset. If "ET" is modified to 0 during timing, the elapsed time increases from the newly set value.



- **Program example**

  ST:

  - When input signal "IN" is FALSE.



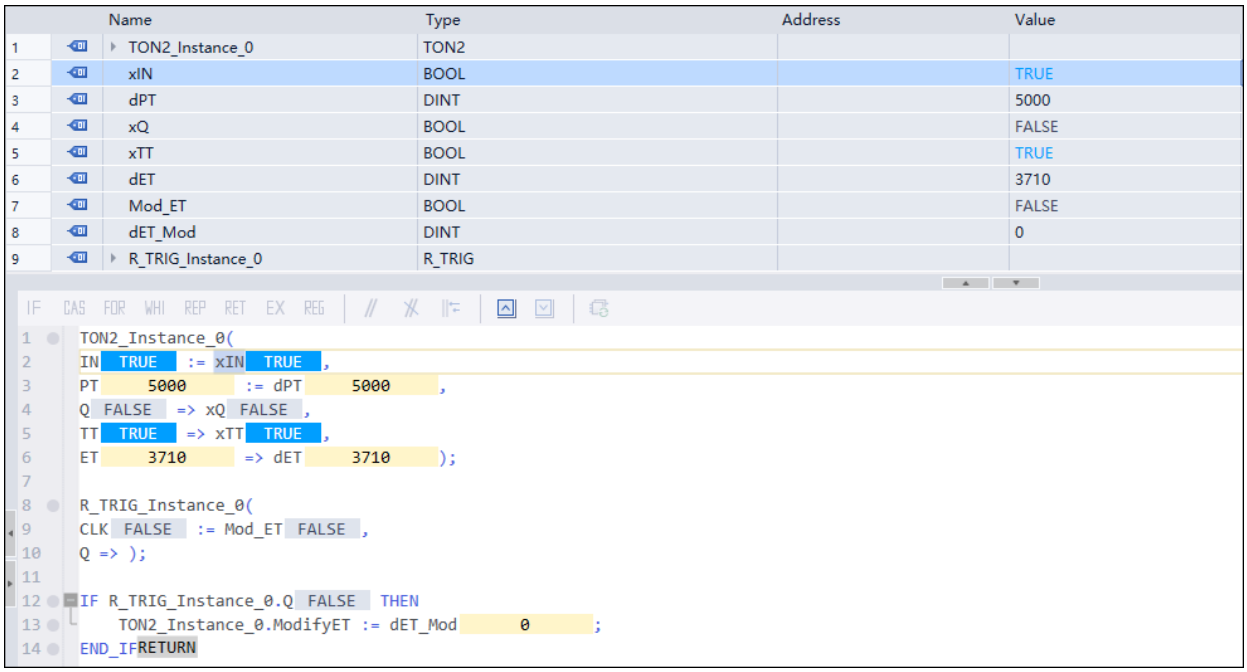
	Name	Type	Address	Value
1	TON2_Instance_0	TON2		
2	xIN	BOOL		FALSE
3	dPT	DINT		5000
4	xQ	BOOL		FALSE
5	xTT	BOOL		FALSE
6	dET	DINT		0
7	Mod_ET	BOOL		FALSE
8	dET_Mod	DINT		0
9	R_TRIG_Instance_0	R_TRIG		

```
TON2_Instance_0(
IN  FALSE  := xIN  FALSE ,
PT     5000      := dPT     5000     ,
Q  FALSE  => xQ  FALSE ,
TT  FALSE  => xTT  FALSE ,
ET     0         => dET     0        );

R_TRIG_Instance_0(
CLK  FALSE  := Mod_ET  FALSE ,
Q => );

IF R_TRIG_Instance_0.Q  FALSE  THEN
    TON2_Instance_0.ModifyET := dET_Mod     0       ;
END_IFRETURN
```

  - When input signal "IN" transitions from FALSE to TRUE, during "ET" timing.



	Name	Type	Address	Value
1	TON2_Instance_0	TON2		
2	xIN	BOOL		TRUE
3	dPT	DINT		5000
4	xQ	BOOL		FALSE
5	xTT	BOOL		TRUE
6	dET	DINT		3710
7	Mod_ET	BOOL		FALSE
8	dET_Mod	DINT		0
9	R_TRIG_Instance_0	R_TRIG		

```
TON2_Instance_0(
IN  TRUE  := xIN  TRUE ,
PT     5000      := dPT     5000     ,
Q  FALSE  => xQ  FALSE ,
TT  TRUE  => xTT  TRUE ,
ET     3710      => dET     3710     );

R_TRIG_Instance_0(
CLK  FALSE  := Mod_ET  FALSE ,
Q => );

IF R_TRIG_Instance_0.Q  FALSE  THEN
    TON2_Instance_0.ModifyET := dET_Mod     0       ;
END_IFRETURN
```

  - During "ET" timing, the rising edge of Mod_ET serves as the modification trigger signal. The set value dET_Mod (dET_Mod = 0) is force-written into the internal "ET" timing value of the timer through the Set accessor of the ModifyET property.

	Name	Type	Address	Value
1	TON2_Instance_0	TON2		
2	xIN	BOOL		TRUE
3	dPT	DINT		5000
4	xQ	BOOL		FALSE
5	xTT	BOOL		TRUE
6	dET	DINT		60
7	Mod_ET	BOOL		TRUE
8	dET_Mod	DINT		0
9	R_TRIG_Instance_0	R_TRIG		

```
TON2_Instance_0(
IN TRUE := xIN TRUE,
PT 5000 := dPT 5000,
Q FALSE => xQ FALSE,
TT TRUE => xTT TRUE,
ET 60 => dET 60);

R_TRIG_Instance_0(
CLK TRUE := Mod_ET TRUE,
Q => );

IF R_TRIG_Instance_0.Q FALSE THEN
    TON2_Instance_0.ModifyET := dET_Mod 0;
END_IFRETURN
```

◦ "ET" timing is complete.

	Name	Type	Address	Value
1	TON2_Instance_0	TON2		
2	xIN	BOOL		TRUE
3	dPT	DINT		5000
4	xQ	BOOL		TRUE
5	xTT	BOOL		FALSE
6	dET	DINT		5000
7	Mod_ET	BOOL		TRUE
8	dET_Mod	DINT		0
9	R_TRIG_Instance_0	R_TRIG		

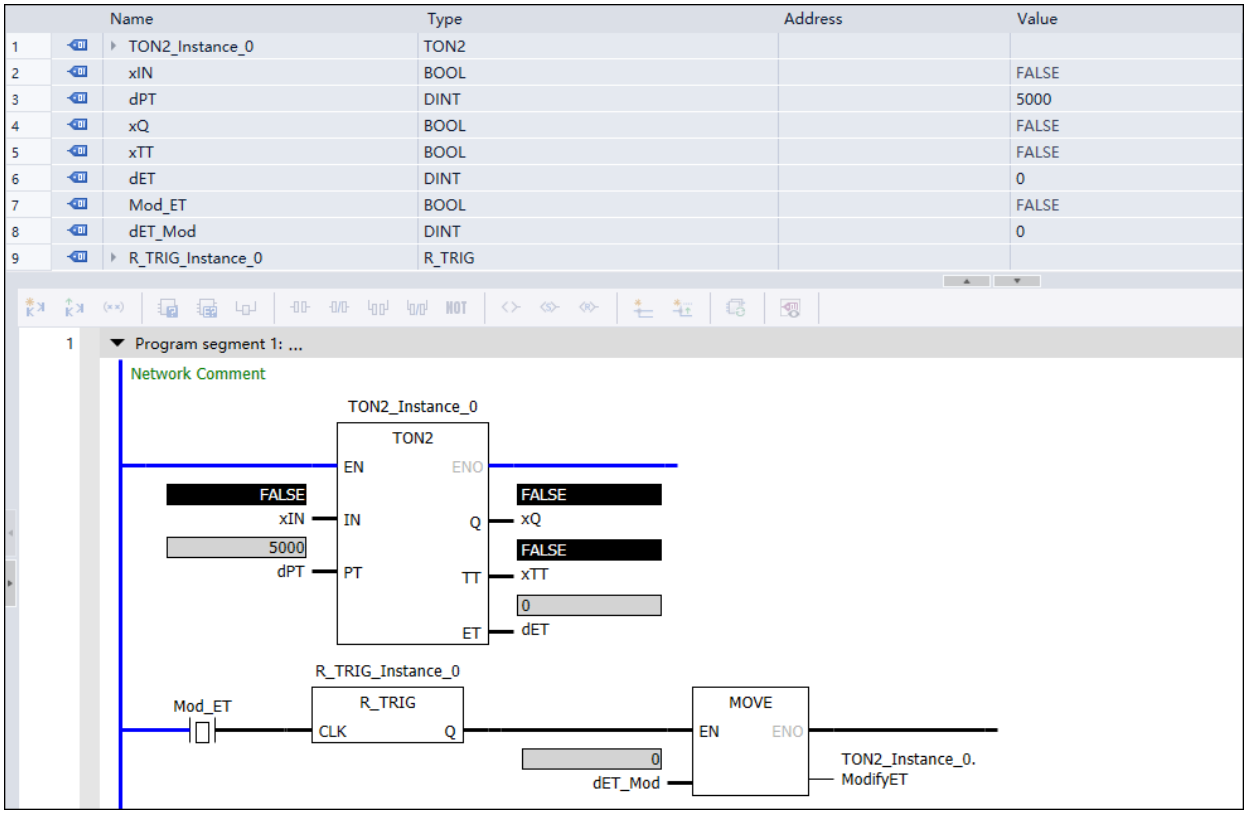
```
TON2_Instance_0(
IN TRUE := xIN TRUE,
PT 5000 := dPT 5000,
Q TRUE => xQ TRUE,
TT FALSE => xTT FALSE,
ET 5000 => dET 5000);

R_TRIG_Instance_0(
CLK TRUE := Mod_ET TRUE,
Q => );

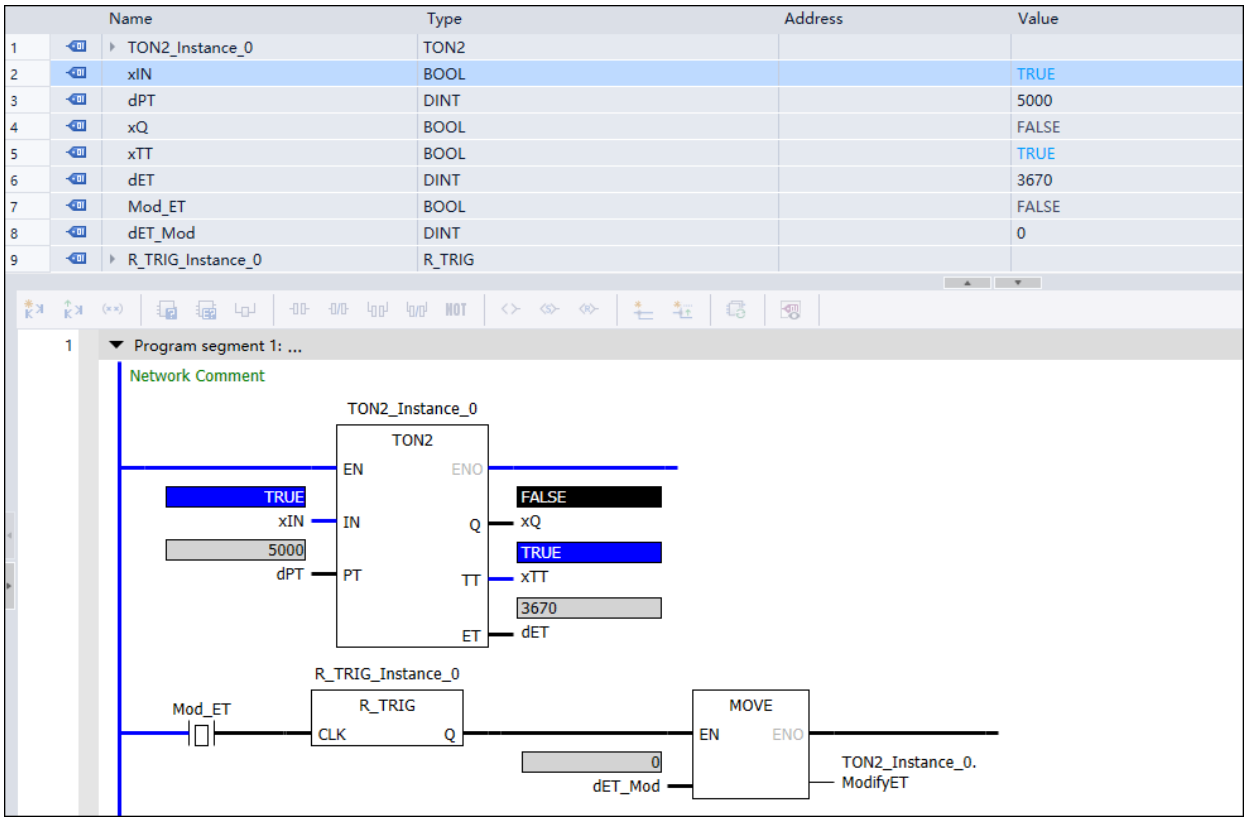
IF R_TRIG_Instance_0.Q FALSE THEN
    TON2_Instance_0.ModifyET := dET_Mod 0;
END_IFRETURN
```

LD:

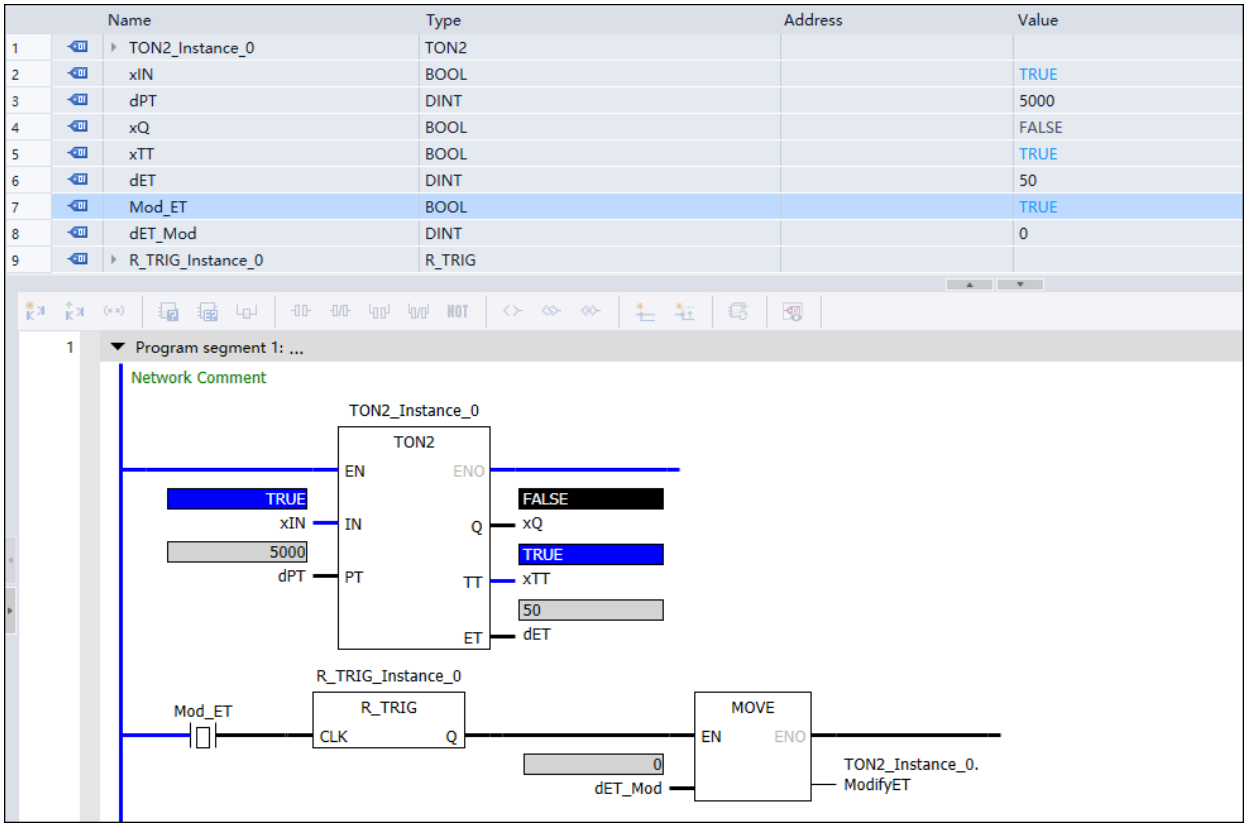
◦ When input signal "IN" is FALSE.



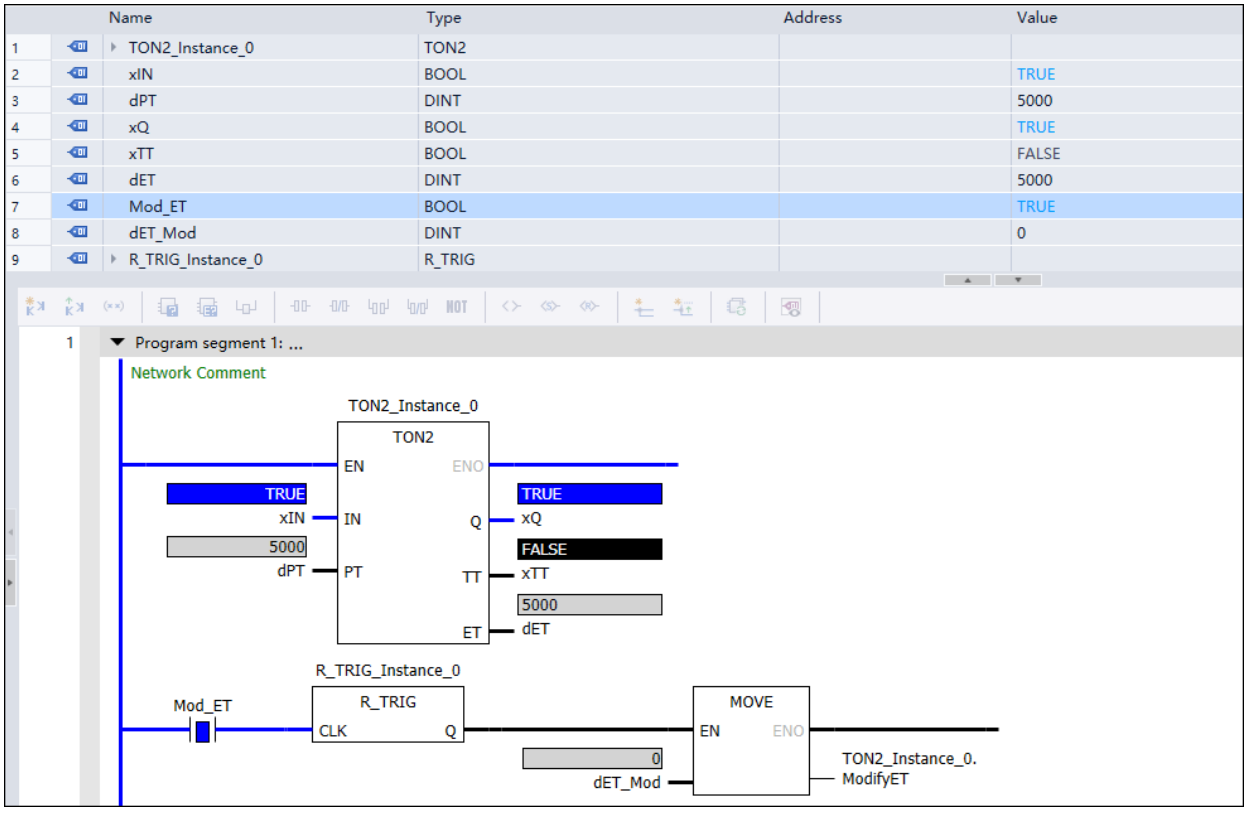
◦ When input signal "IN" transitions from FALSE to TRUE, during "ET" timing.



◦ During "ET" timing, the rising edge of Mod_ET serves as the modification trigger signal. The set value dET_Mod (dET_Mod = 0) is force-written into the internal "ET" timing value of the timer through the Set accessor of the ModifyET property.



◦ "ET" timing is complete.



- **Precautions**
  - PT and "ET" are set in ms, with a timing accuracy of 1 ms.
  - If "PT" or "ET" is input with a value less than 0, it is automatically corrected to 0 for operation.
  - Before "ET" reaches "PT", the value of "ET" can be changed. The following actions will occur.

Timer status	Value of Q	New Value of ET	Action
Before timing	FALSE	-	When the value of "IN" becomes TRUE, "ET" starts increasing from 0, and the modified value does not take effect.
After timing	TRUE	-	The value of "Q" remains TRUE. The value of "ET" changes (timer is not running).
During timing	FALSE	PT ≥ ET	Timing restarts based on the modified value of ET. When "ET" reaches the value of "PT", "Q" becomes TRUE, and "ET" stops increasing.
		PT < ET	The value of "Q" immediately becomes TRUE, and "ET" immediately stops increasing.

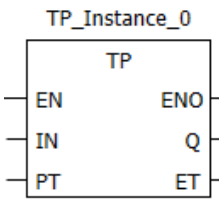
- Before the value of "ET" reaches the value of "PT", "PT" can be modified, and the following actions will occur.

Timer status	Value of Q	New Value of PT	Action
After timing	TRUE	-	The value of "Q" remains TRUE. The value of "ET" does not change and remains at the value of "PT" before it is modified.
During timing	FALSE	"PT" ≥ "ET"	When the value of "IN" becomes TRUE, timing continues. When the value of "ET" reaches the new value of "PT", the value of "Q" becomes TRUE, and "ET" stops increasing.
		"PT" < "ET"	When the value of "IN" becomes TRUE, the value of "Q" immediately becomes TRUE. "ET" also immediately stops increasing.

18.1.9.6 TP

Outputs the current timing time and timing status based on the input timing time.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
TP	Standard timer	FB		TP_Instance_0(IN := , PT := , Q => , ET =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input signal	Follows data type	-	Starts timing when "IN" changes from FALSE to TRUE.
	PT	Set timing time	Follows data type	-	Set time
Output	Q	Timing completion signal	Follows data type	-	Timing completes when "Q" changes from TRUE to FALSE.
	ET	Current timing time	Follows data type	-	Current timing time

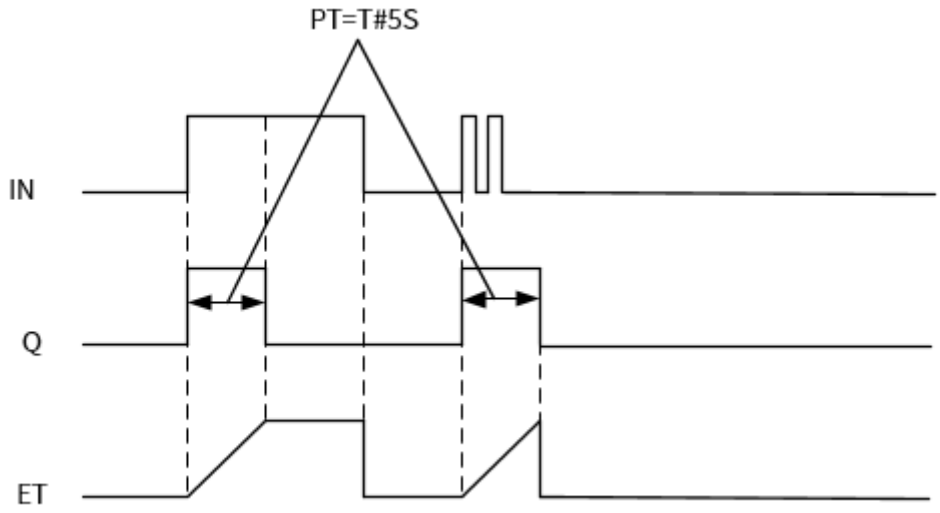
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ET	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–

- **Function**

  When "IN" is TRUE, "Q" outputs a high-level signal for the duration "PT". "ET" displays the current timing time; when "ET" = "PT", "Q" becomes FALSE.

- **Sequence diagram**



- **Program example**

  ST:

  (1) Input signal "IN" is TRUE

		Name	Data type	Address	Value	Prepar...	Comment
1		TP_Instance_0	TP				
2		IN	BOOL		TRUE		
3		PT	TIME		T#5s		
4		Q	BOOL		TRUE		
5		ET	TIME		T#578ms		

```
1  TP_Instance_0(
2  IN TRUE := xIN TRUE,
3  PT T#5s := xPT T#5s,
4  Q TRUE => xQ TRUE,
5  ET T#578ms => xResult T#578ms );RETURN
```

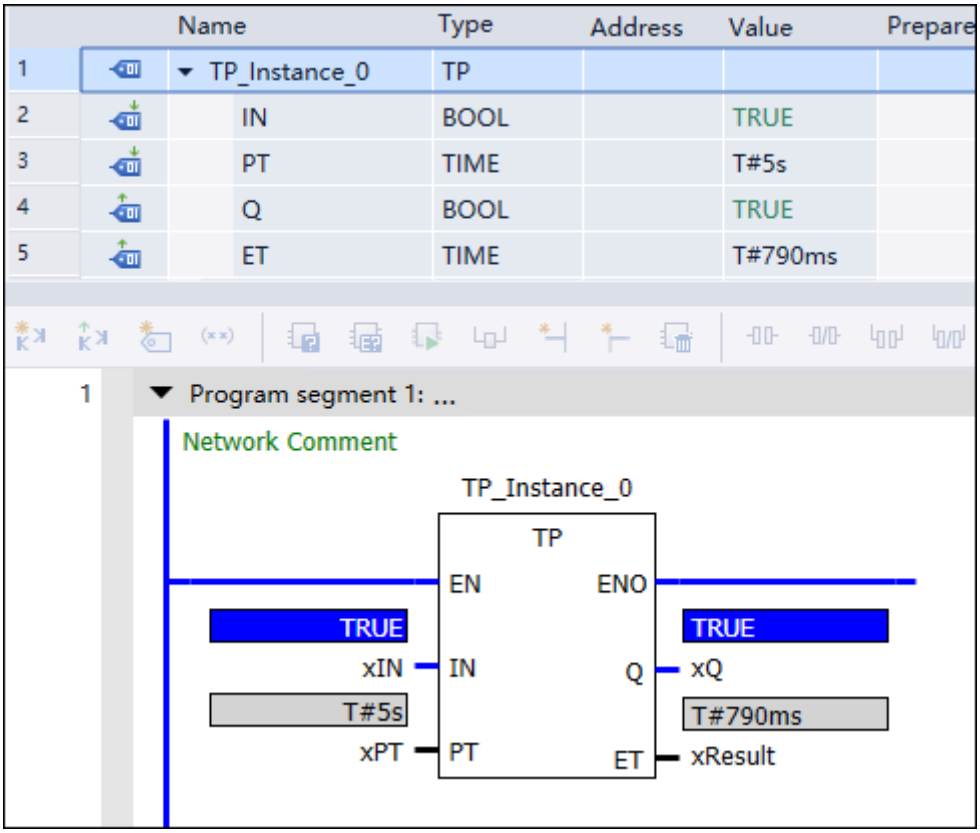
  (2) Current timing time (ET) equals set timing time (PT)

		Name	Data type	Address	Value	准备值	注释
1		TP_Instance_0	TP				
2		IN	BOOL		TRUE		输入信号
3		PT	TIME		T#5s		计时时间
4		Q	BOOL		FALSE		计时完成
5		ET	TIME		T#5s		计时输出

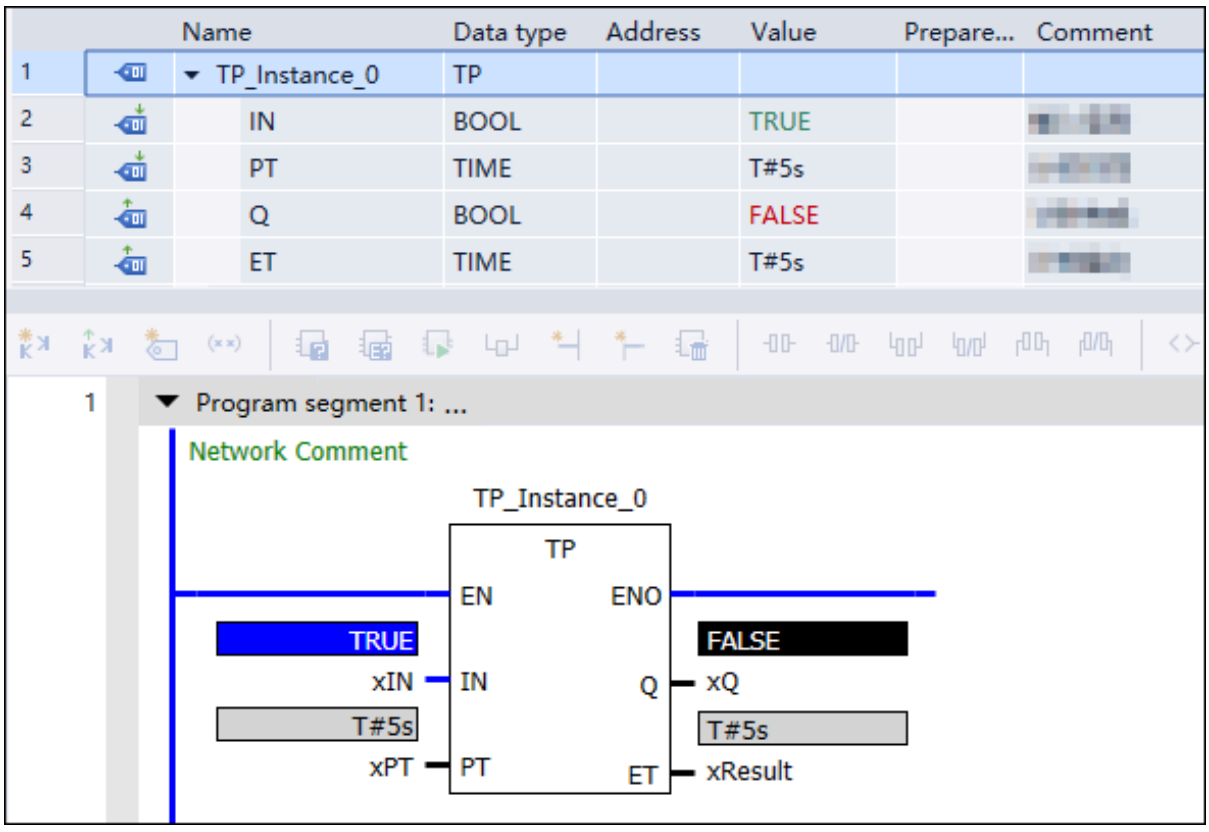
```
1  TP_Instance_0(
2  IN TRUE := xIN TRUE,
3  PT T#5s := xPT T#5s,
4  Q FALSE => xQ FALSE,
5  ET T#5s => xResult T#5s );RETURN
```

  LD:

  (1) Input signal "IN" is TRUE



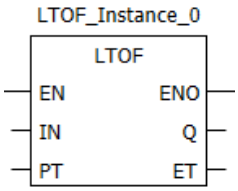
(2) Current timing time (ET) equals set timing time (PT)



18.1.9.7 LTOF

Outputs a timing completion signal after a delay following power-off.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LTOF	Power-off delay timer	FB		LTOF_Instance_0(IN := , PT := , Q => , ET =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input signal	Follows data type	-	Starts timing when "IN" changes from TRUE to FALSE.
	PT	Set timing time	Follows data type	-	Set time
Output	Q	Output	Follows data type	-	Timing completes when "Q" changes from TRUE to FALSE.
	ET	Current timing time	Follows data type	-	Elapsed time since the falling edge of "IN".

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
ET	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–

- **Function**

  The input signal "PT" sets the timing duration. When the input signal "IN" changes from TRUE to FALSE, timing starts. After timing completes, "Q" is set to FALSE.

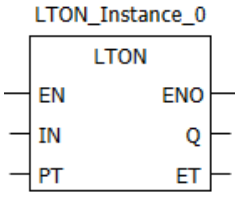
- **Program example**

  The program example is similar to 18.1.9.3 TOF in the standard library, except that the data types of "PT" and "ET" are changed from TIME to LTIME, thus providing higher precision.

18.1.9.8 LTON

Outputs a timing completion signal after a delay following power-on.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LTON	Power-on delay timer	FB		LTON_Instance_0(IN := , PT := , Q => , ET =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input signal	Follows data type	-	Starts timing when "IN" changes from FALSE to TRUE.
	PT	Set timing time	Follows data type	-	Set time

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Q	Output	Follows data type	-	Timing completes when "Q" changes from FALSE to TRUE.
	ET	Current timing time	Follows data type	-	Current timing time

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ET	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–

- **Function**

  Input signal "PT" sets the timing duration. When input signal "IN" changes from FALSE to TRUE, timing starts. After timing completes, "Q" is set to TRUE.

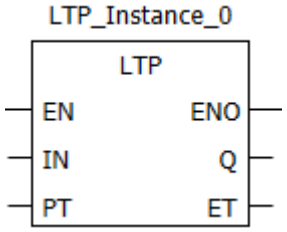
- **Program example**

  The program example is similar to 18.1.9.4 TON in the standard library, except that the data types of "PT" and "ET" are changed from TIME to LTIME, thus providing higher precision.

18.1.9.9 LTP

Outputs the current timing time and timing status based on the input timing time.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LTP	Standard timer	FB		LTP_Instance_0(IN := , PT := , Q => , ET =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input signal	Follows data type	-	Starts timing when "IN" changes from FALSE to TRUE.
	PT	Set timing time	Follows data type	-	Set time
Output	Q	Timer output	Follows data type	-	Timing completes when "Q" changes from TRUE to FALSE.
	ET	Timing time output	Follows data type	-	Elapsed time since the rising edge of "IN".

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	-	√	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
ET	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–

- **Function**

  When "IN" is TRUE, "Q" outputs a high-level signal for the duration "PT". "ET" displays the current timing time; when "ET" = "PT", "Q" becomes FALSE.

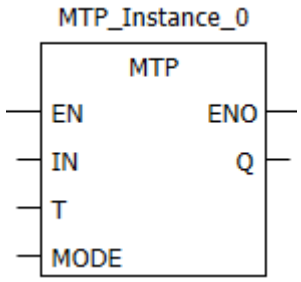
- **Program example**

  The program example is similar to 18.1.9.6 TP in the standard library, except that the data types of "PT" and "ET" are changed from TIME to LTIME, thus providing higher precision.

18.1.9.10 MTP

A pulse timing output block.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
MTP	Pulse timing output block	FB		MTP_Instance_0(IN := , T := , MODE := , Q =>);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input	Follows data type	-	Input
	T	Set timing time	Follows data type	TIME#0ms	Timing time.
	MODE	Input mode	Follows data type	-	FALSE for maximum pulse timing. TRUE for minimum pulse timing.

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Output	Q	Pulse output	Follows data type	-	Pulse output

Data Type

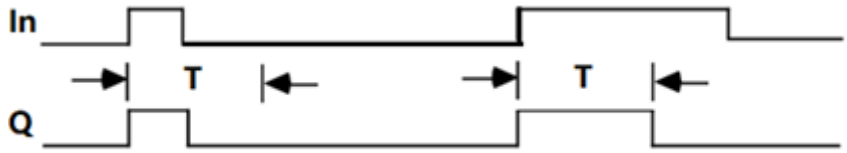
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
T	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
MODE	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Includes both maximum and minimum pulse timing functions.

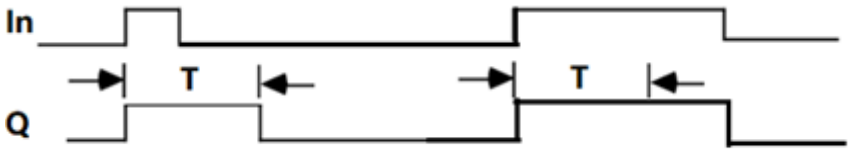
  ◦ Maximum pulse timing

  When the pulse input "IN" is on a rising edge, the output "Q" is 1. When the timing time reaches "T" or when "IN" is on a falling edge, the output "Q" is reset to 0.



  ◦ Minimum pulse timing output

  When the pulse input "IN" is on a rising edge, the output "Q" is 1. When the timing time reaches "T" or when "IN" is on a falling edge, the output "Q" is reset to 0.



- **Program example**

  ◦ If T is set to T#10s, MTP_Instance_0 will set MODE to FALSE for maximum pulse timing, and MTP_Instance_1 will set MODE to TRUE for minimum pulse timing.

  ▪ When TON_Instance_0 sets "PT" to T#5s, the high-level input time of "IN" is 5s. Under maximum pulse timing, the high-level output time of "Q" is 5s. Under minimum pulse timing, the high-level output time of "Q" is 10s.

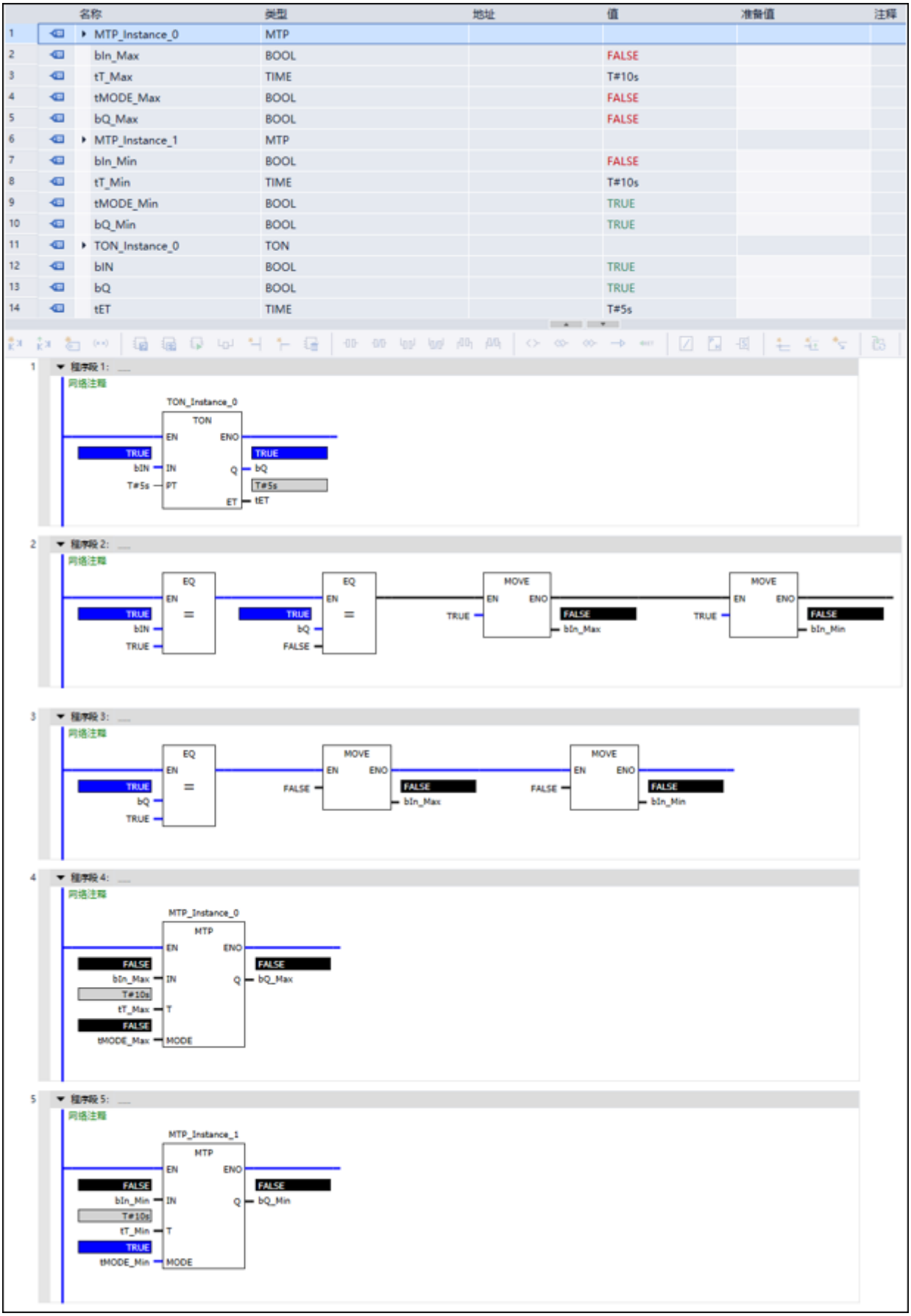
- When TON_Instance_0 sets "PT" to T#15s, the high-level input time of "IN" is 15s. Under maximum pulse timing, the high-level output time of "Q" is 10s. Under minimum pulse timing, the high-level output time of "Q" is 15s.

ST:

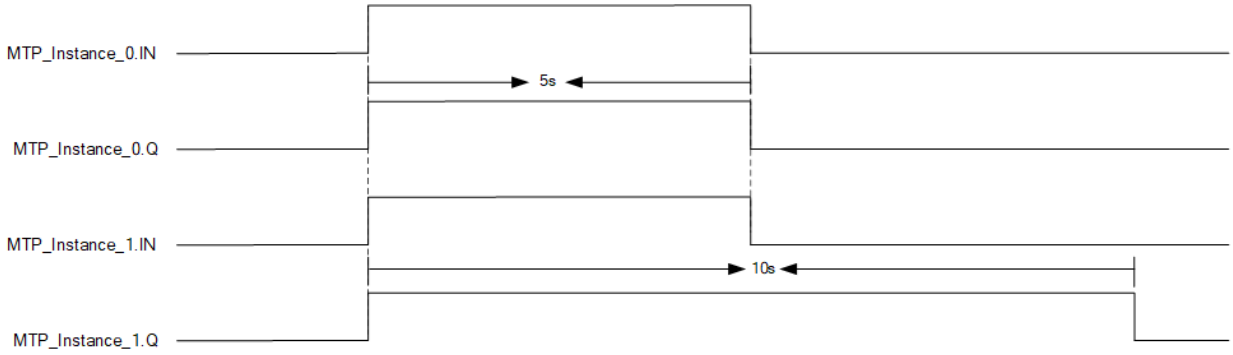
	Name	Data type	Address	Value
1	MTP_Instance_0	MTP		
2	bIn_Max	BOOL		FALSE
3	tT_Max	TIME		T#10s
4	tMODE_Max	BOOL		FALSE
5	bQ_Max	BOOL		FALSE
6	MTP_Instance_1	MTP		
7	bIn_Min	BOOL		FALSE
8	tT_Min	TIME		T#10s
9	tMODE_Min	BOOL		TRUE
10	bQ_Min	BOOL		FALSE
11	TON_Instance_0	TON		
12	bIN	BOOL		TRUE
13	bQ	BOOL		TRUE
14	tET	TIME		T#5s

```
TON_Instance_0(
IN := bIN,
PT := T#5s,
Q => bQ,
ET => tET);
IF (bIN AND (NOT bQ)) THEN
    bIn_Max := TRUE;
    bIn_Min := TRUE;
END_IF
IF bQ THEN
    bIn_Max := FALSE;
    bIn_Min := FALSE;
END_IF
MTP_Instance_0(
IN := bIn_Max,
T := T#10s := tT_Max := T#10s,
MODE := tMODE_Max,
Q => bQ_Max);
MTP_Instance_1(
IN := bIn_Min,
T := T#10s := tT_Min := T#10s,
MODE := tMODE_Min,
Q => bQ_Min);RETURN
```

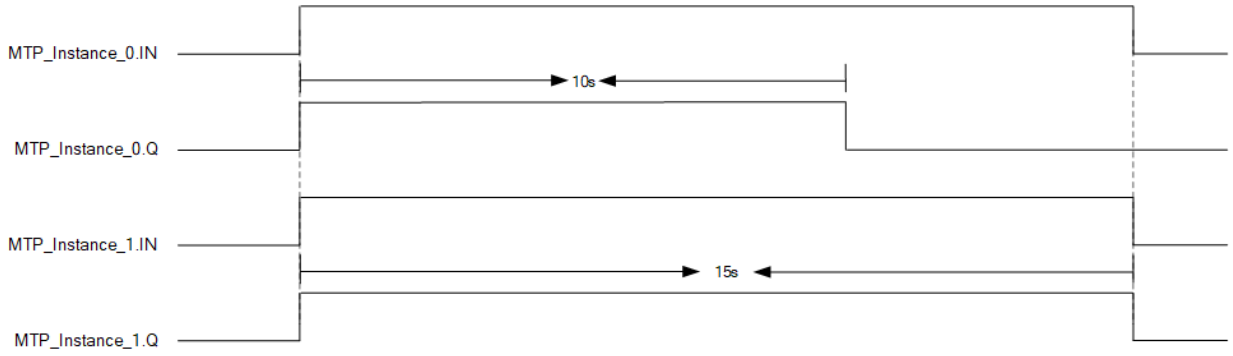
LD:



◦ The high-level input time of "IN" is 5s, the timing diagram is shown in the figure below.



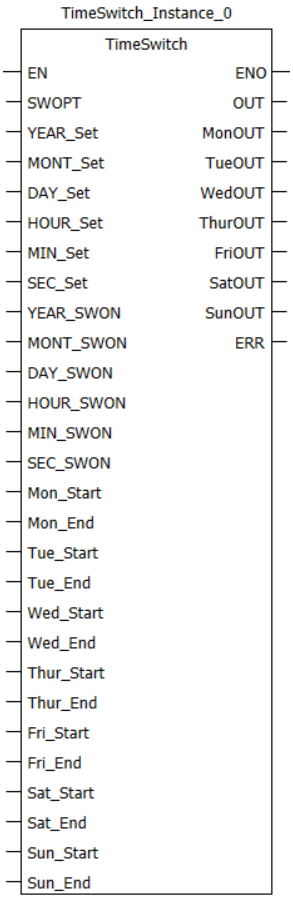
- The high-level input time of "IN" is 15s, the timing diagram is shown in the figure below.



18.1.9.11 TimeSwitch

Timing control function block.

- **Instruction format**

TimeSwitch	Timing control function block	FB		TimeSwitch_Instance_0(SWOPT := , YEAR_Set := , MONT_Set := , DAY_Set := , HOUR_Set := , MIN_Set := , SEC_Set := , YEAR_SWON := , MONT_SWON := , DAY_SWON := , HOUR_SWON := , MIN_SWON := , SEC_SWON := , Mon_Start := , Mon_End := , Tue_Start := , Tue_End := , Wed_Start := , Wed_End := , Thur_Start := , Thur_End := , Fri_Start := , Fri_End := , Sat_Start := , Sat_End := , Sun_Start := , Sun_End := , OUT => , MonOUT => , TueOUT => , WedOUT => , ThurOUT => , FriOUT => , SatOUT => , SunOUT => ,

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
				ERR =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	SWOPT	Time judgment method	Follows data type	FALSE	FALSE: Judges by time. TRUE: Judges by week.
	YEAR_Set	Sets judgment time: Year	Follows data type	1970	Sets judgment time: Year
	MONT_Set	Sets judgment time: Month	Follows data type	1	Sets judgment time: Month
	DAY_Set	Sets judgment time: Day	Follows data type	1	Sets judgment time: Day
	HOUR_Set	Sets judgment time: Hour	Follows data type	0	Sets judgment time: Hour
	MIN_Set	Sets judgment time: Minute	Follows data type	0	Sets judgment time: Minute
	SEC_Set	Sets judgment time: Second	Follows data type	0	Sets judgment time: Second
	YEAR_SWON	Year check switch	Follows data type	FALSE	Year check switch
	MONT_SWON	Month check switch	Follows data type	FALSE	Month check switch
	DAY_SWON	Day check switch	Follows data type	FALSE	Day check switch
	HOUR_SWON	Hour check switch	Follows data type	FALSE	Hour check switch
	MIN_SWON	Minute check switch	Follows data type	FALSE	Minute check switch
	SEC_SWON	Second check switch	Follows data type	FALSE	Second check switch

Input/Output	Variable	Name	Valid Range	Default Value	Description
	Mon_Start	Set start time for Monday	Follows data type	-	Set start time for Monday
	Mon_End	Set end time for Monday	Follows data type	-	Set end time for Monday
	Tue_Start	Set start time for Tuesday	Follows data type	-	Set start time for Tuesday
	Tue_End	Set end time for Tuesday	Follows data type	-	Set end time for Tuesday
	Wed_Start	Set start time for Wednesday	Follows data type	-	Set start time for Wednesday
	Wed_End	Set end time for Wednesday	Follows data type	-	Set end time for Wednesday
	Thur_Start	Set start time for Thursday	Follows data type	-	Set start time for Thursday
	Thur_End	Set end time for Thursday	Follows data type	-	Set end time for Thursday
	Fri_Start	Set start time for Friday	Follows data type	-	Set start time for Friday
	Fri_End	Set end time for Friday	Follows data type	-	Set end time for Friday
	Sat_Start	Set start time for Saturday	Follows data type	-	Set start time for Saturday
	Sat_End	Set end time for Saturday	Follows data type	-	Set end time for Saturday
	Sun_Start	Set start time for Sunday	Follows data type	-	Set start time for Sunday
	Sun_End	Set end time for Sunday	Follows data type	-	Set end time for Sunday
Output	OUT	Output	Follows data type	FALSE	Output
	MonOUT	Output for Monday	Follows data type	FALSE	Output for Monday
	TueOUT	Output for Tuesday	Follows data type	FALSE	Output for Tuesday

Input/Output	Variable	Name	Valid Range	Default Value	Description
	WedOUT	Output for Wednesday	Follows data type	FALSE	Output for Wednesday
	ThurOUT	Output for Thursday	Follows data type	FALSE	Output for Thursday
	FriOUT	Output for Friday	Follows data type	FALSE	Output for Friday
	SatOUT	Output for Saturday	Follows data type	FALSE	Output for Saturday
	SunOUT	Output for Sunday	Follows data type	FALSE	Output for Sunday
	ERR	Time setting fault	Follows data type	FALSE	Determines whether the time setting is faulty.

Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
SWOPT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
YEAR_Set	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
MONT_Set	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
DAY_Set	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
HOUR_Set	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
MIN_Set	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SEC_Set	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
YEAR_SWON	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
MONT_SWON	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
DAY_SWON	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
HOUR_SWON	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
MIN_SWON	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SEC_SWON	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Mon_Start	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–
Mon_End	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–
Tue_Start	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–
Tue_End	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–
Wed_Start	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–
Wed_End	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–
Thur_Start	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–
Thur_End	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–
Fri_Start	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–
Fri_End	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–
Sat_Start	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–
Sat_End	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–
Sun_Start	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–
Sun_End	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–
OUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
MonOUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
TueOUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
WedOUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ThurOUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FriOUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SatOUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SunOUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ERR	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  In industrial production, a common requirement is for equipment to start and stop periodically according to user-defined time setting to fulfill process needs. For example, equipment may need to perform scheduled cleaning and maintenance during weekdays, operating for certain durations every morning and evening from Monday to Friday. Another example is that equipment may need to operate periodically during specific or multiple time slots based on temporary work arrangements. In both cases, a flexibly configurable function block tailored to on-site processes is indispensable to ensure the smooth execution of production processes.

  This block functions similarly to a clock timer. The system time is compared with the set time, and an event output is generated when the condition is met. The following two timing modes are supported:

  - **Time-based setting**

    - Set times support year, month, day, hour, minute, second, and millisecond, and each can be individually enabled or disabled via a switch for timing function. When the set time is reached, the output "OUT" becomes TRUE.

TimeSwitch_Instance_0	TimeSwitch					不公开	
SWOPT	BOOL		FALSE			不公开	时间判定方式选择：0-按时间判断 1-按星期判断
YEAR_Set	WORD		1970			不公开	设定判断时间：年
MONT_Set	BYTE		1			不公开	设定判断时间：月
DAY_Set	BYTE		0			不公开	设定判断时间：日
HOUR_Set	BYTE		0			不公开	设定判断时间：时
MIN_Set	BYTE		0			不公开	设定判断时间：分
SEC_Set	BYTE		0			不公开	设定判断时间：秒
YEAR_SWON	BOOL		FALSE			不公开	年判定开启开关
MONT_SWON	BOOL		FALSE			不公开	月判定开启开关
DAY_SWON	BOOL		FALSE			不公开	日判定开启开关
HOUR_SWON	BOOL		FALSE			不公开	时判定开启开关
MIN_SWON	BOOL		FALSE			不公开	分判定开启开关
SEC_SWON	BOOL		FALSE			不公开	秒判定开启开关

    - If the equipment needs to start at 9:00 every day, it can be set as shown in the following figure:

TimeSwitch_Instance_0	TimeSwitch					
SWOPT	BOOL		FALSE			
YEAR_Set	WORD		1970			
MONT_Set	BYTE		1			
DAY_Set	BYTE		1			
HOUR_Set	BYTE		9			
MIN_Set	BYTE		0			
SEC_Set	BYTE		0			
YEAR_SWON	BOOL		FALSE			
MONT_SWON	BOOL		FALSE			
DAY_SWON	BOOL		FALSE			
HOUR_SWON	BOOL		TRUE			
MIN_SWON	BOOL		FALSE			
SEC_SWON	BOOL		FALSE			

- If the equipment needs to start on a specific time, such as September 24, 2024, at 12:30, it can be set as shown in the following figure:

SWOPT	BOOL		FALSE			不公开	时间判定方式选择：0-按时间判断 1-按星期判断
YEAR_Set	WORD		2024			不公开	设定判断时间：年
MONT_Set	BYTE		9			不公开	设定判断时间：月
DAY_Set	BYTE		24			不公开	设定判断时间：日
HOUR_Set	BYTE		12			不公开	设定判断时间：时
MIN_Set	BYTE		30			不公开	设定判断时间：分
SEC_Set	BYTE		0			不公开	设定判断时间：秒
YEAR_SWON	BOOL		TRUE			不公开	年判定开启开关
MONT_SWON	BOOL		TRUE			不公开	月判定开启开关
DAY_SWON	BOOL		TRUE			不公开	日判定开启开关
HOUR_SWON	BOOL		TRUE			不公开	时判定开启开关
MIN_SWON	BOOL		TRUE			不公开	分判定开启开关
SEC_SWON	BOOL		FALSE			不公开	秒判定开启开关

- The following parameter configuration faults will trigger "ERR" to become TRUE and output "OUT" to become FALSE.

1. The year check switch is enabled, and the set year is less than 1970.
2. The month check switch is enabled, and the set month is outside the range [1,12].
3. The day check switch is enabled, and the set day exceeds the number of days in the specified month of the year, such as setting 30 for February.
4. The hour check switch is enabled, and the set hour is outside the range [0, 23].
5. The minute check switch is enabled, and the set minute is outside the range [0, 59].
6. The second check switch is enabled, and the set second is outside the range [0, 59].

◦ **Weekday-based setting**

- Set times support weekdays.

SWOPT	BOOL		TRUE			

	Mon_Start	TIME_OF_DAY					不公开	星期一的起始设定时间
	Mon_End	TIME_OF_DAY					不公开	星期一的停止设定时间
	Tue_Start	TIME_OF_DAY					不公开	星期二的起始设定时间
	Tue_End	TIME_OF_DAY					不公开	星期二的停止设定时间
	Wed_Start	TIME_OF_DAY					不公开	星期三的起始设定时间
	Wed_End	TIME_OF_DAY					不公开	星期三的停止设定时间
	Thur_Start	TIME_OF_DAY					不公开	星期三的起始设定时间
	Thur_End	TIME_OF_DAY					不公开	星期四的停止设定时间
	Fri_Start	TIME_OF_DAY					不公开	星期五的起始设定时间
	Fri_End	TIME_OF_DAY					不公开	星期五的停止设定时间
	Sat_Start	TIME_OF_DAY					不公开	星期六的起始设定时间
	Sat_End	TIME_OF_DAY					不公开	星期六的停止设定时间
	Sun_Start	TIME_OF_DAY					不公开	星期日的起始设定时间
	Sun_End	TIME_OF_DAY					不公开	星期日的停止设定时间

- The function block can output 7 independent weekday events and an overall event. If any weekday event output is TRUE, the overall output "OUT" becomes TRUE. The time shall be configured in 24-hour format.

	OUT	BOOL		FALSE			不公开	输出
	MonOUT	BOOL		FALSE			不公开	星期一触发输出
	TueOUT	BOOL		FALSE			不公开	星期二触发输出
	WedOUT	BOOL		FALSE			不公开	星期三触发输出
	ThurOUT	BOOL		FALSE			不公开	星期四触发输出
	FriOUT	BOOL		FALSE			不公开	星期五触发输出
	SatOUT	BOOL		FALSE			不公开	星期六触发输出
	SunOUT	BOOL		FALSE			不公开	星期日触发输出
	ERR	BOOL		FALSE			不公开	判断时间设定错误

- If the equipment needs to run from 9:00 to 12:00 every Monday and from 15:00 to 17:30 every Wednesday, it can be configured as shown below. The corresponding outputs MonOUT and WedOUT can be used.

	Mon_Start	TIME_OF_DAY		TOD#9:0:0			不公开	星期一的起始设定时间
	Mon_End	TIME_OF_DAY		TOD#12:0:0			不公开	星期一的停止设定时间
	Tue_Start	TIME_OF_DAY		TOD#0:0:0			不公开	星期二的起始设定时间
	Tue_End	TIME_OF_DAY		TOD#0:0:0			不公开	星期二的停止设定时间
	Wed_Start	TIME_OF_DAY		TOD#15:0:0			不公开	星期三的起始设定时间
	Wed_End	TIME_OF_DAY		TOD#15:30:0			不公开	星期三的停止设定时间
	Thur_Start	TIME_OF_DAY		TOD#0:0:0			不公开	星期三的起始设定时间
	Thur_End	TIME_OF_DAY		TOD#0:0:0			不公开	星期四的停止设定时间
	Fri_Start	TIME_OF_DAY		TOD#0:0:0			不公开	星期五的起始设定时间
	Fri_End	TIME_OF_DAY		TOD#0:0:0			不公开	星期五的停止设定时间
	Sat_Start	TIME_OF_DAY		TOD#0:0:0			不公开	星期六的起始设定时间
	Sat_End	TIME_OF_DAY		TOD#0:0:0			不公开	星期六的停止设定时间
	Sun_Start	TIME_OF_DAY		TOD#0:0:0			不公开	星期日的起始设定时间
	Sun_End	TIME_OF_DAY		TOD#0:0:0			不公开	星期日的停止设定时间

- For example, in a sequential control process, the equipment may need to start at different times during the week and stop at other times. Additionally, two other equipment may need to operate simultaneously with the main motor at fixed times each week. All of these can be achieved through this function block.

- When the start time is set later than the end time, "ERR" becomes TRUE and "OUT" becomes FALSE.

- **Program example**

  Turn on the device at the fixed time 2024-9-24-3:13, and after reaching the set time, "OUT" becomes TRUE.

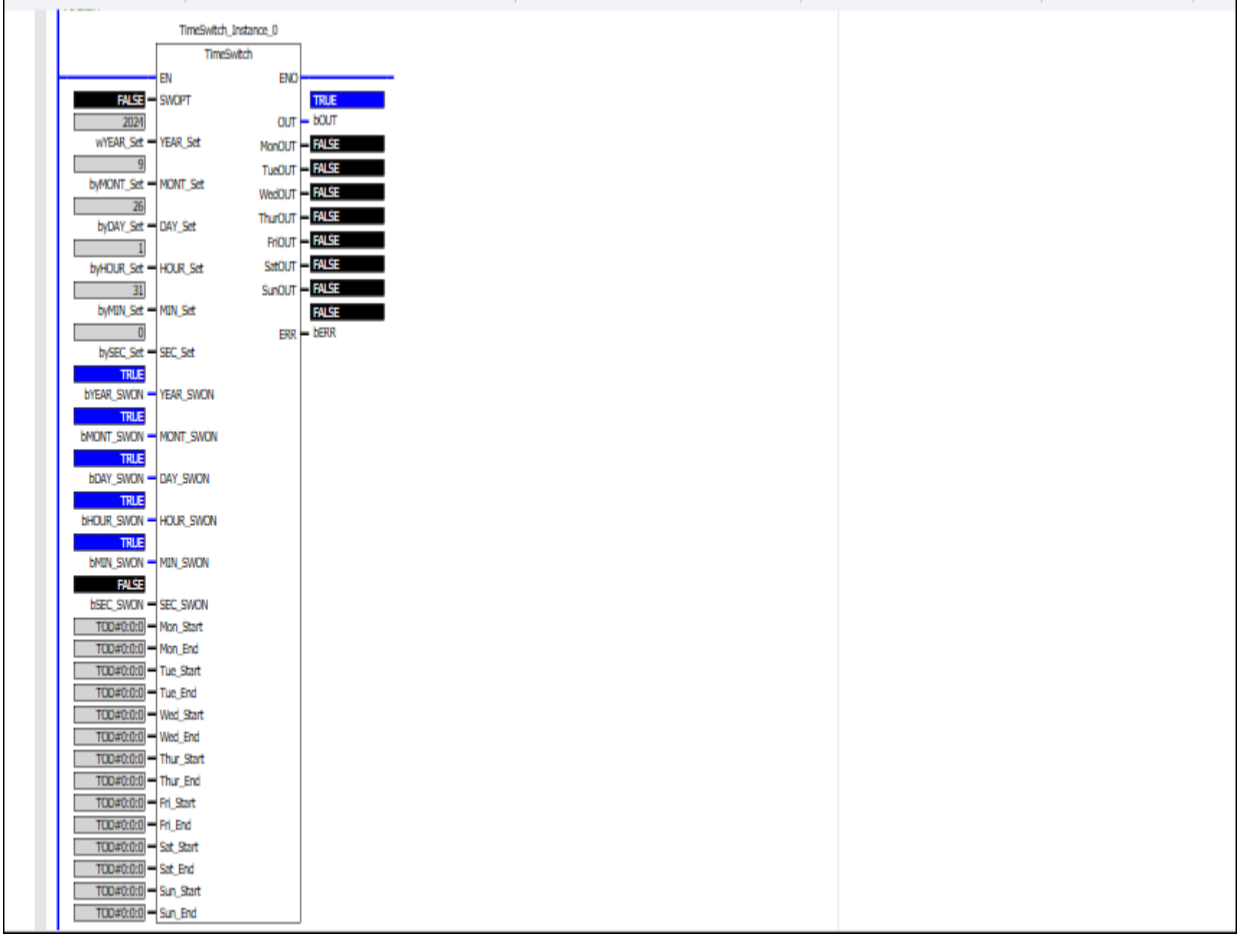
  ST:

	名称	类型	地址	值
1	TimeSwitch_Instance_0	TimeSwitch		
2	bSWOPT	BOOL		FALSE
3	wYEAR_Set	WORD		2024
4	byMONT_Set	BYTE		9
5	byDAY_Set	BYTE		26
6	byHOUR_Set	BYTE		1
7	byMIN_Set	BYTE		31
8	bySEC_Set	BYTE		0
9	bYEAR_SWON	BOOL		TRUE
10	bMONT_SWON	BOOL		TRUE
11	bDAY_SWON	BOOL		TRUE
12	bHOUR_SWON	BOOL		TRUE
13	bMIN_SWON	BOOL		TRUE
14	bSEC_SWON	BOOL		FALSE
15	bOUT	BOOL		TRUE
16	bERR	BOOL		FALSE
17	todMon_Start	TIME_OF_DAY		TOD#0:0:0
18	todMon_End	TIME_OF_DAY		TOD#0:0:0
19	todTue_Start	TIME_OF_DAY		TOD#0:0:0
20	todTue_End	TIME_OF_DAY		TOD#0:0:0
21	todWed_Start	TIME_OF_DAY		TOD#0:0:0
22	todWed_End	TIME_OF_DAY		TOD#0:0:0
23	todThur_Start	TIME_OF_DAY		TOD#0:0:0
24	todThur_End	TIME_OF_DAY		TOD#0:0:0
25	todFri_Start	TIME_OF_DAY		TOD#0:0:0
26	todFri_End	TIME_OF_DAY		TOD#0:0:0
27	todSat_Start	TIME_OF_DAY		TOD#0:0:0
28	todSat_End	TIME_OF_DAY		TOD#0:0:0
29	todSun_Start	TIME_OF_DAY		TOD#0:0:0
30	todSun_End	TIME_OF_DAY		TOD#0:0:0
31	bMonOUT	BOOL		FALSE
32	bTueOUT	BOOL		FALSE
33	bWedOUT	BOOL		FALSE
34	bThurOUT	BOOL		FALSE
35	bFriOUT	BOOL		FALSE
36	bSatOUT	BOOL		FALSE
37	bSunOUT	BOOL		FALSE

```
TimeSwitch_Instance_0(
SWOPT FALSE := bSWOPT FALSE,
YEAR_Set 2024 := wYEAR_Set 2024,
MONT_Set 9 := byMONT_Set 9,
DAY_Set 26 := byDAY_Set 26,
HOUR_Set 1 := byHOUR_Set 1,
MIN_Set 31 := byMIN_Set 31,
SEC_Set 0 := bySEC_Set 0,
YEAR_SWON TRUE := bYEAR_SWON TRUE,
MONT_SWON TRUE := bMONT_SWON TRUE,
DAY_SWON TRUE := bDAY_SWON TRUE,
HOUR_SWON TRUE := bHOUR_SWON TRUE,
MIN_SWON TRUE := bMIN_SWON TRUE,
SEC_SWON FALSE := bSEC_SWON FALSE,
Mon_Start TOD#0:0:0 := todMon_Start TOD#0:0:0,
Mon_End TOD#0:0:0 := todMon_End TOD#0:0:0,
Tue_Start TOD#0:0:0 := todTue_Start TOD#0:0:0,
Tue_End TOD#0:0:0 := todTue_End TOD#0:0:0,
Wed_Start TOD#0:0:0 := todWed_Start TOD#0:0:0,
Wed_End TOD#0:0:0 := todWed_End TOD#0:0:0,
Thur_Start TOD#0:0:0 := todThur_Start TOD#0:0:0,
Thur_End TOD#0:0:0 := todThur_End TOD#0:0:0,
Fri_Start TOD#0:0:0 := todFri_Start TOD#0:0:0,
Fri_End TOD#0:0:0 := todFri_End TOD#0:0:0,
Sat_Start TOD#0:0:0 := todSat_Start TOD#0:0:0,
Sat_End TOD#0:0:0 := todSat_End TOD#0:0:0,
Sun_Start TOD#0:0:0 := todSun_Start TOD#0:0:0,
Sun_End TOD#0:0:0 := todSun_End TOD#0:0:0,
OUT TRUE => bOUT TRUE,
MonOUT FALSE => bMonOUT FALSE,
TueOUT FALSE => bTueOUT FALSE,
WedOUT FALSE => bWedOUT FALSE,
ThurOUT FALSE => bThurOUT FALSE,
FriOUT FALSE => bFriOUT FALSE,
SatOUT FALSE => bSatOUT FALSE,
SunOUT FALSE => bSunOUT FALSE,
ERR FALSE => bERR FALSE);
```

  LD:

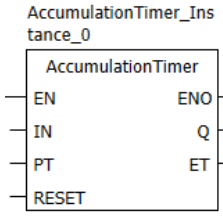
	Name	Data type	Address	Value
1	TimeSwitch_Instance_0	TimeSwitch		
2	wYEAR_Set	WORD		2024
3	byMONT_Set	BYTE		9
4	byDAY_Set	BYTE		26
5	byHOUR_Set	BYTE		1
6	byMIN_Set	BYTE		31
7	bySEC_Set	BYTE		0
8	bYEAR_SWON	BOOL		TRUE
9	bMONT_SWON	BOOL		TRUE
10	bDAY_SWON	BOOL		TRUE
11	bHOUR_SWON	BOOL		TRUE
12	bMIN_SWON	BOOL		TRUE
13	bSEC_SWON	BOOL		FALSE
14	bOUT	BOOL		TRUE



18.1.9.12 AccumulationTimer

Accumulating timer.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AccumulationTimer	Accumulating timer	FB		AccumulationTimer_Instance_0(IN := , PT := , RESET := , Q => , ET =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Timer input	Follows data type	FALSE	TRUE: Timer is active. FALSE: Timer is stopped.
	PT	Set time	Follows data type	TIME#100ms	Maximum value of timing
	RESET	Reset a bit	Follows data type	FALSE	TRUE: Timer is reset. FALSE: Timer is not reset.
Output	Q	Timer output	Follows data type	-	TRUE: "ET" reaches "PT". FALSE: "ET" has not reached "PT".
	ET	Accumulated time	Follows data type	-	Accumulated time

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
PT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ET	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–

- **Function**

  Accumulates the period of time during which the timer input "IN" is TRUE.

  When "RESET" is FALSE, the timer starts when "IN" changes from FALSE to TRUE, and the accumulated time "ET" increases simultaneously as time elapses.

  When "IN" changes from TRUE to FALSE, the timer stops. The value of "ET" at that moment is retained. When "IN" becomes TRUE again, the timer starts again.

  "ET" starts increasing from the retained value. When "ET" reaches the set time "PT", the timer output "Q" becomes TRUE. At this point, "ET" stops increasing.

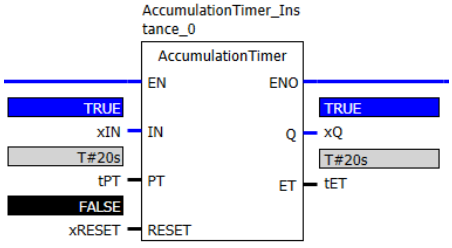
  When "RESET" becomes TRUE, the timer resets. The value of "ET" becomes 0, and "Q" becomes FALSE.

- **Program example**

  - The following example sets "PT" to T#20s.

	LD	ST
Variable definition		
Program		

```
VAR
    AccumulationTimer_Instance_0: AccumulationTimer;
    xIN: BOOL := FALSE;
    tPT: TIME := TIME#15ms;
    xRESET: BOOL := FALSE;
    xQ: BOOL;
    tET: TIME;
END_VAR
```

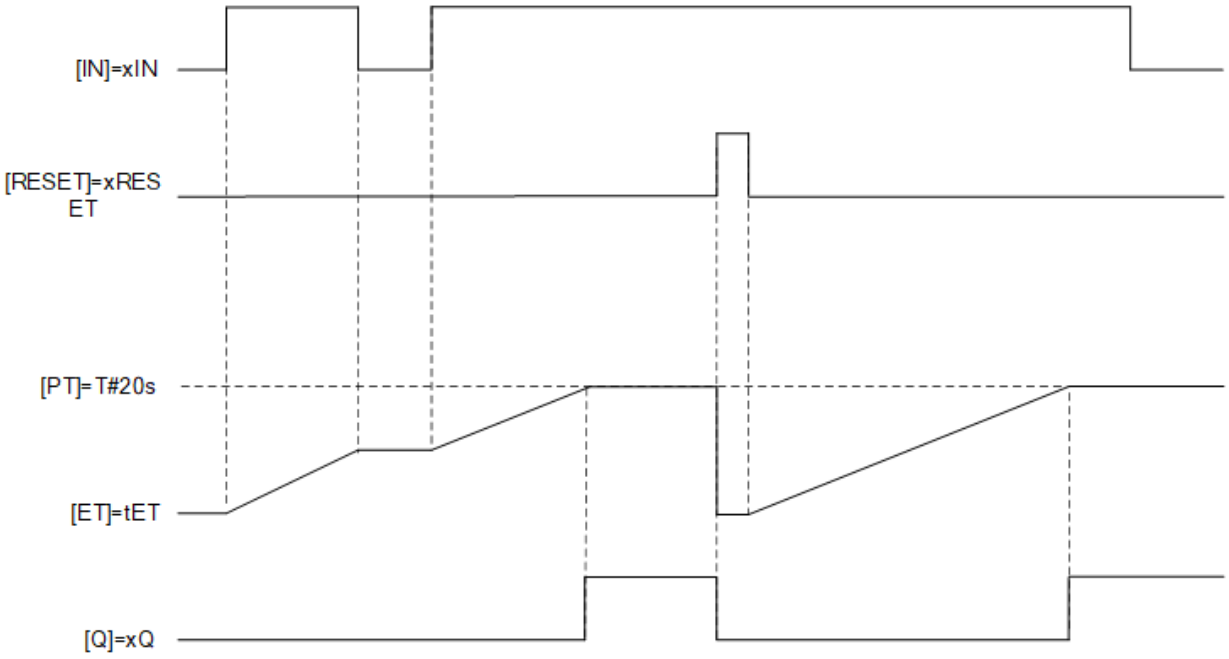



```
AccumulationTimer_Instance_0(
IN := xIN,
PT := tPT,
RESET := xRESET,
Q => xQ,
ET => tET);RETURN
```

	LD	ST
Operation result		

Name	Data type	Address	Value
AccumulationTimer_Instance_0	AccumulationTimer		
xIN	BOOL		TRUE
tPT	TIME		T#20s
xRESET	BOOL		FALSE
xQ	BOOL		TRUE
tET	TIME		T#20s

- The timing diagram when "PT" = T#20s is shown in the figure below. When "RESET" is FALSE and the accumulated time reaches 20s, the value of "Q" becomes TRUE. When "RESET" is set to TRUE, "ET" and "Q" are reset.



- **Precautions**
  - "ET" and "Q" are updated when this instruction is executed. Strictly speaking, the condition for "Q" to become TRUE is not the moment when the accumulated timer operation time reaches "PT". Instead, it is the moment when the accumulated timer operation time reaches "PT", and this instruction is executed for the first time after that. Therefore, a maximum delay of one task cycle may occur.
  - "PT" and "ET" are set in milliseconds, with a timing accuracy of 1 ms.
  - When both "IN" and "RESET" become TRUE, "RESET" takes priority. That is, the value of "ET" becomes 0, and "Q" becomes FALSE.
  - If the value of "IN" is already TRUE when execution starts, timing begins from that moment.
  - When "PT" is set to T#0ms, "Q" becomes TRUE immediately after the value of "IN" becomes TRUE.
  - If the MainTask cycle time of the system is greater than "PT", the value of "ET" equals the task cycle time. After one task cycle completes, the value of "Q" becomes TRUE.
  - Before the value of "ET" reaches the value of "PT", "PT" can be modified, and the following actions will occur.

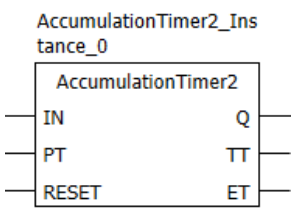
Timer Status	Value of "Q"	New Value of PT	Action
After timing	TRUE	-	The value of "Q" remains TRUE. The value of "ET" does not change and remains at the value of "PT" before it is modified.
During timing	FALSE	"PT" ≥ "ET"	When the value of "IN" becomes TRUE, timing continues. When the value of "ET" reaches the new value of "PT", the value of "Q" becomes TRUE, and "ET" stops increasing.
		"PT" < "ET"	When the value of "IN" becomes TRUE, the value of "Q" immediately becomes TRUE. "ET" also immediately stops increasing.

- After the operation is stopped via the user interface, "ET" and "Q" will continue their calculations.
- For "PT", the timing error which "Q" is TRUE ranges from -100 ns to (100 ns + 1 task cycle). Details of the above range are shown below.
  - ±100 ns is the timing error for "ET".
  - For each task cycle, "ET" is judged to check if it reaches "PT". If "ET" reaches "PT", there is a delay of one task cycle.

18.1.9.13 AccumulationTimer2

Accumulation timer. The timer output can be modified during timing.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AccumulationTimer2	Accumulating timer	FB		AccumulationTimer2_Instance_0(IN := , PT := , RESET := , Q => , TT => , ET =>);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input signal	Follows data type.	FALSE	TRUE: Timer is active. FALSE: Timer is stopped.
	PT	Set time	Follows data type	0	Set time
	RESET	Reset a bit	Follows data type	FALSE	TRUE: Timer is reset. FALSE: Timer is not reset.
Output	Q	Timing completion signal	Follows data type	FALSE	Timing completion signal TRUE when "ET" is equal to or greater than PT.
	TT	Timing in progress	Follows data type	FALSE	TRUE during timing; FALSE when not timing or when timing is complete.
	ET	Accumulated time	Follows data type	0	Accumulated time. Modifying "ET" during timing takes effect immediately.

Variable types

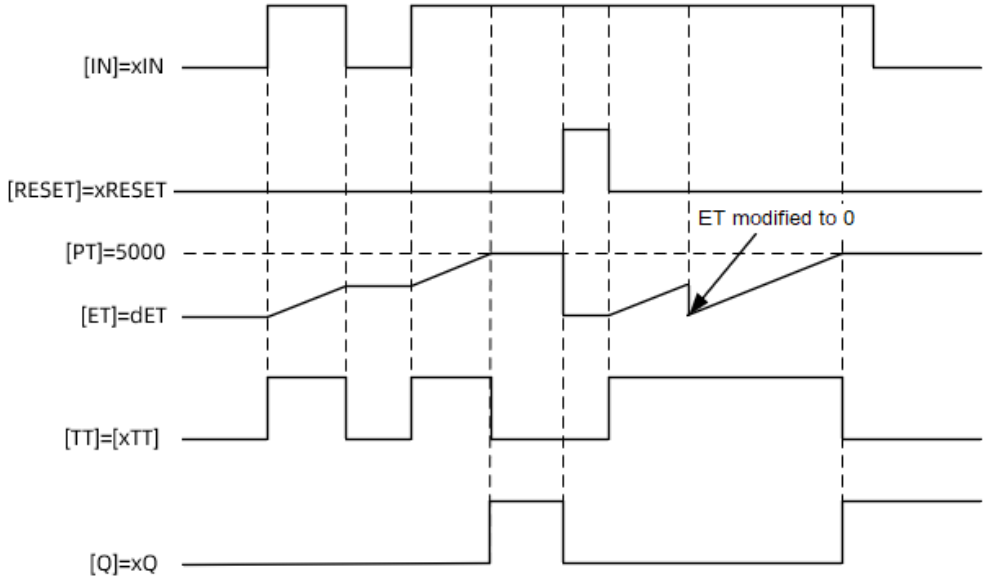
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PT	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
RESET	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Q	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
TT	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ET	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

- **Function**
  - The accumulating timer calculates the total duration for which the timer input signal "IN" is TRUE. "PT" and "ET" are set in ms.
  - When "RESET" is FALSE, the timer starts when "IN" changes from FALSE to TRUE. During timing, "TT" is set to TRUE, and the accumulated time "ET" increases over time.
  - When "IN" transitions from TRUE to FALSE, the timer stops but retains the "ET" value at that moment. When "IN" becomes TRUE again, the timer starts again.

- "ET" starts increasing from the retained value. When "ET" reaches the set time "PT", the timer output "Q" becomes TRUE. At this point, "ET" stops increasing.
- When "RESET" becomes TRUE, the timer resets. "ET" becomes 0, and "TT" and "Q" become FALSE.
- When modifying the "ET" parameter, the internal "ET" timing value must be forcibly written via the Set accessor of the ModifyET property.

- **Timing diagram**

  Set "PT" to 5000. When "RESET" is FALSE and the accumulated time for variable "IN" being TRUE reaches 5000 ms, variable "Q" becomes TRUE. When "RESET" is set to TRUE, "ET" and "Q" are reset. During timer operation, if "ET" is modified to 0, "ET" starts increasing from the set value.



- **Program example**

  ST:

  - When input signal "IN" is FALSE.

	Name	Type	Address	Value
1	AccumulationTimer2_Instance_0	AccumulationTimer2		
2	xIN	BOOL		FALSE
3	dPT	DINT		5000
4	xRESET	BOOL		FALSE
5	xQ	BOOL		FALSE
6	xTT	BOOL		FALSE
7	dET	DINT		0
8	Mod_ET	BOOL		FALSE
9	dET_Mod	DINT		0
10	R_TRIG_Instance_0	R_TRIG		

```
AccumulationTimer2_Instance_0(
IN FALSE := xIN FALSE ,
PT 5000 := dPT 5000 ,
RESET FALSE := xRESET FALSE ,
Q FALSE => xQ FALSE ,
TT FALSE => xTT FALSE ,
ET 0 => dET 0 );

R_TRIG_Instance_0(
CLK FALSE := Mod_ET FALSE ,
Q => );

IF R_TRIG_Instance_0.Q FALSE THEN
    AccumulationTimer2_Instance_0.ModifyET := dET_Mod 0 ;
END_IFRETURN
```

  - When input signal "IN" transitions from FALSE to TRUE, during "ET" timing.

	Name	Type	Address	Value
1	AccumulationTimer2_Instance_0	AccumulationTimer2		
2	xIN	BOOL		TRUE
3	dPT	DINT		5000
4	xRESET	BOOL		FALSE
5	xQ	BOOL		FALSE
6	xTT	BOOL		TRUE
7	dET	DINT		2210
8	Mod_ET	BOOL		FALSE
9	dET_Mod	DINT		0
10	R_TRIG_Instance_0	R_TRIG		

```
AccumulationTimer2_Instance_0(
IN TRUE := xIN TRUE ,
PT 5000 := dPT 5000 ,
RESET FALSE := xRESET FALSE ,
Q FALSE => xQ FALSE ,
TT TRUE => xTT TRUE ,
ET 2210 => dET 2210 );

R_TRIG_Instance_0(
CLK FALSE := Mod_ET FALSE ,
Q => );

IF R_TRIG_Instance_0.Q FALSE THEN
    AccumulationTimer2_Instance_0.ModifyET := dET_Mod 0 ;
END_IFRETURN
```

- During "ET" timing, the rising edge of Mod_ET serves as the modification trigger signal. The set value dET_Mod is forcibly written to the timer's internal "ET" timing value via the Set accessor of the ModifyET property.

	Name	Type	Address	Value
1	AccumulationTimer2_Instance_0	AccumulationTimer2		
2	xIN	BOOL		TRUE
3	dPT	DINT		5000
4	xRESET	BOOL		FALSE
5	xQ	BOOL		FALSE
6	xTT	BOOL		TRUE
7	dET	DINT		1170
8	Mod_ET	BOOL		TRUE
9	dET_Mod	DINT		0
10	R_TRIG_Instance_0	R_TRIG		

```
AccumulationTimer2_Instance_0(
IN TRUE := xIN TRUE ,
PT 5000 := dPT 5000 ,
RESET FALSE := xRESET FALSE ,
Q FALSE => xQ FALSE ,
TT TRUE => xTT TRUE ,
ET 1170 => dET 1170 );

R_TRIG_Instance_0(
CLK TRUE := Mod_ET TRUE ,
Q => );

IF R_TRIG_Instance_0.Q FALSE THEN
    AccumulationTimer2_Instance_0.ModifyET := dET_Mod 0 ;
END_IFRETURN
```

- "ET" timing is complete.

	Name	Type	Address	Value
1	AccumulationTimer2_Instance_0	AccumulationTimer2		
2	xIN	BOOL		TRUE
3	dPT	DINT		5000
4	xRESET	BOOL		FALSE
5	xQ	BOOL		TRUE
6	xTT	BOOL		FALSE
7	dET	DINT		5000
8	Mod_ET	BOOL		TRUE
9	dET_Mod	DINT		0
10	R_TRIG_Instance_0	R_TRIG		

```
AccumulationTimer2_Instance_0(
IN  TRUE   := xIN  TRUE  ,
PT      5000       := dPT      5000      ,
RESET  FALSE   := xRESET  FALSE  ,
Q  TRUE   => xQ  TRUE  ,
TT  FALSE   => xTT  FALSE  ,
ET      5000       => dET      5000      );

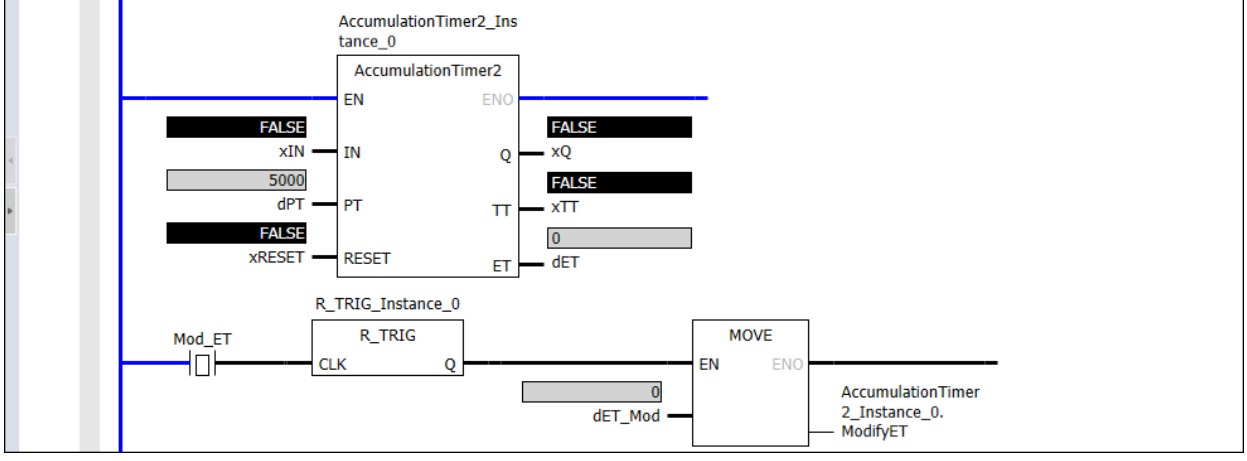
R_TRIG_Instance_0(
CLK  TRUE   := Mod_ET  TRUE  ,
Q => );

IF R_TRIG_Instance_0.Q  FALSE  THEN
    AccumulationTimer2_Instance_0.ModifyET := dET_Mod      0      ;
END_IFRETURN
```

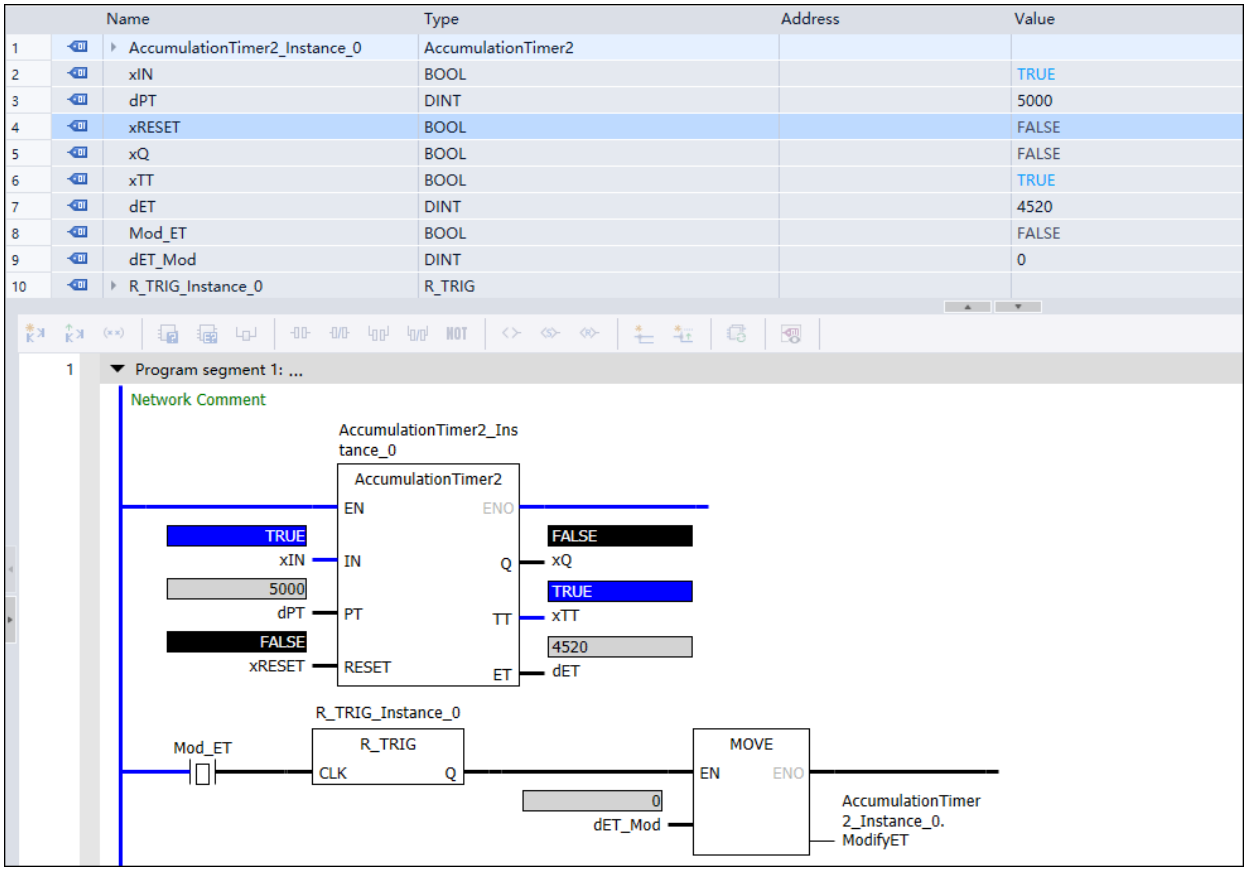
LD:

◦ When input signal "IN" is FALSE.

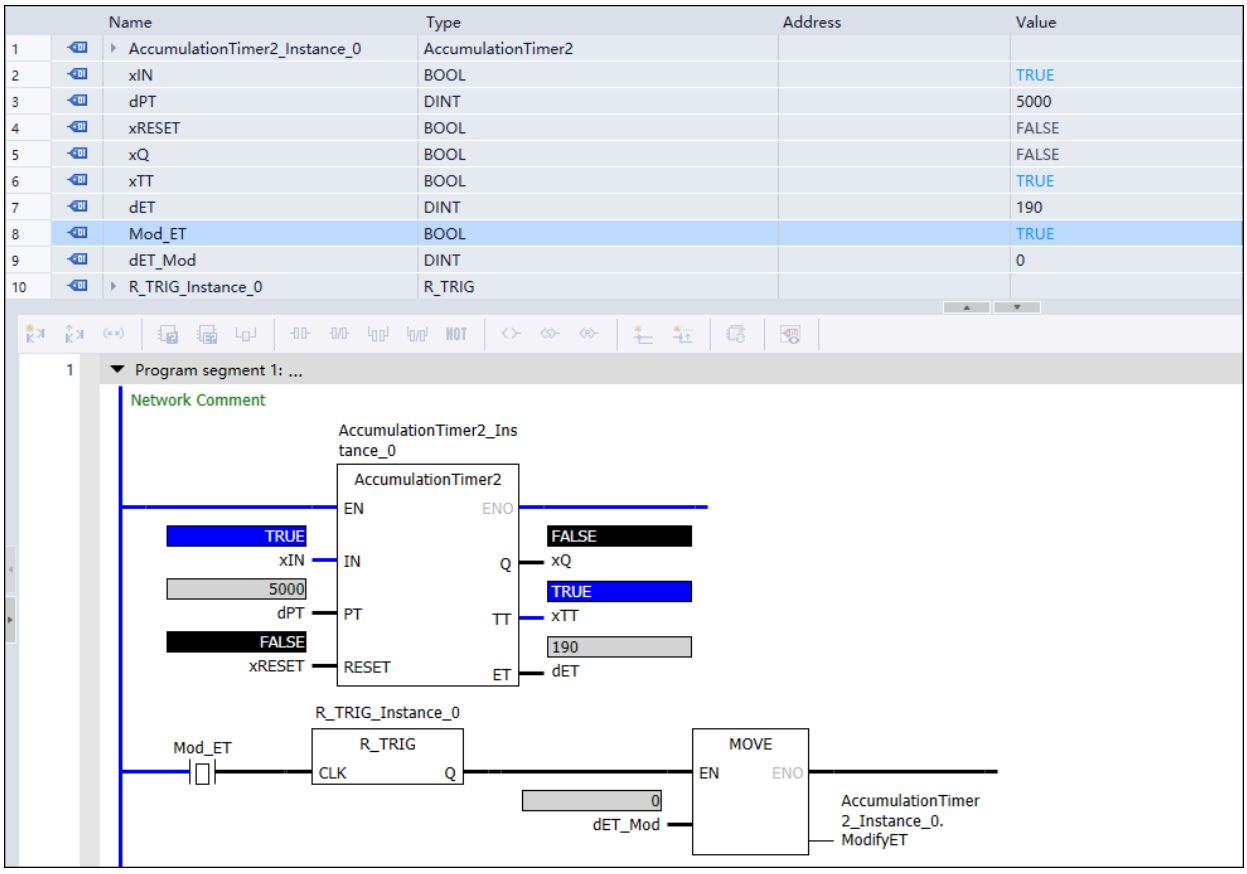
	Name	Type	Address	Value
1	AccumulationTimer2_Instance_0	AccumulationTimer2		
2	xIN	BOOL		FALSE
3	dPT	DINT		5000
4	xRESET	BOOL		FALSE
5	xQ	BOOL		FALSE
6	xTT	BOOL		FALSE
7	dET	DINT		0
8	Mod_ET	BOOL		FALSE
9	dET_Mod	DINT		0
10	R_TRIG_Instance_0	R_TRIG		



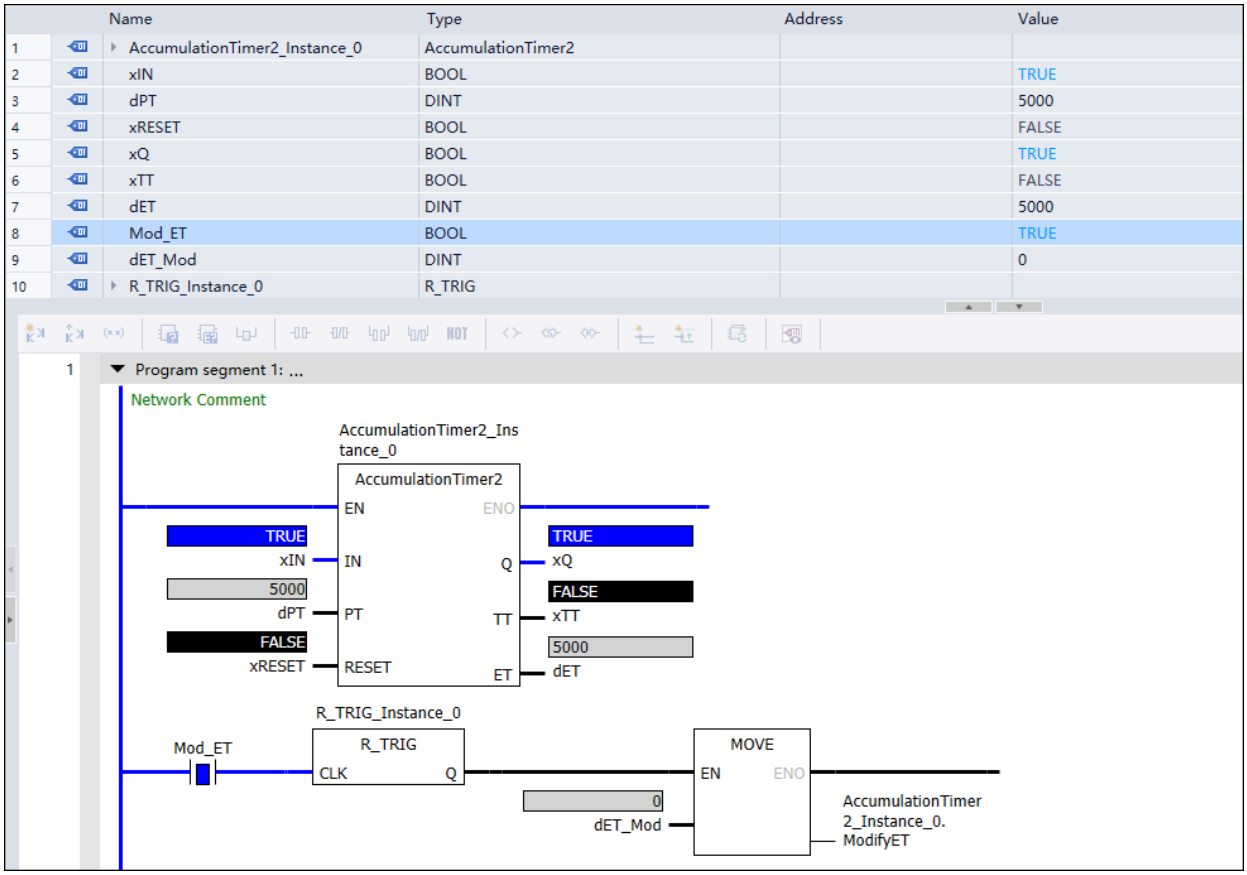
◦ When input signal "IN" transitions from FALSE to TRUE, during "ET" timing.



◦ During "ET" timing, the rising edge of Mod_ET serves as the modification trigger signal. The set value dET_Mod is forcibly written to the timer's internal "ET" timing value via the Set accessor of the ModifyET property.



◦ "ET" timing is complete.



- **Precautions**
  - When both "IN" and "RESET" become TRUE, "RESET" takes priority, meaning the value of "ET" becomes 0 and "Q" becomes FALSE.
  - If the value of "IN" is already TRUE when execution starts, timing begins from that moment.
  - When "PT" is set to 0, "Q" becomes TRUE immediately after the value of "IN" becomes TRUE.
  - If the system cycle time of the system is greater than "PT", the value of "ET" equals the task cycle time. After one task cycle completes, the value of "Q" becomes TRUE.
  - When the input for the delay time "PT" or the current timing time "ET" is less than 0, it is automatically corrected to 0 for operation.
  - Before "ET" reaches "PT", the value of "ET" can be changed. The following actions will occur.

Timer Status	Value of "Q"	New Value of PT	Action
Before timing	FALSE	-	When the value of "IN" becomes TRUE, "ET" starts increasing from 0, and the modified value does not take effect.
After timing	TRUE	-	The value of "Q" remains TRUE, and the value of "ET" changes (the timer does not run).
Timing in progress/ stopped	FALSE	"PT" ≥ "ET"	When the value of "IN" becomes TRUE, timing restarts based on the modified value of "ET". When the value of "ET" reaches the value of "PT", the value of "Q" becomes TRUE, and "ET" stops increasing.

Timer Status	Value of "Q"	New Value of PT	Action
		PT < ET	When the value of "IN" becomes TRUE, the value of "Q" immediately becomes TRUE, and "ET" immediately stops increasing.

- Before the value of "ET" reaches the value of "PT", "PT" can be modified, and the following actions will occur.

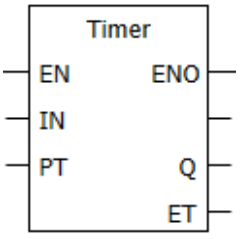
Timer Status	Value of "Q"	New Value of PT	Action
After timing	TRUE	-	The value of "Q" remains TRUE, and the value of "ET" also remains unchanged (retaining the "PT" value before the change).
Timing in progress/ stopped	FALSE	"PT" ≥ "ET"	When the value of "IN" becomes TRUE, timing continues. When the value of "ET" reaches the new value of "PT", the value of "Q" becomes TRUE, and "ET" stops increasing.
		"PT" < "ET"	When the value of "IN" becomes TRUE, the value of "Q" immediately becomes TRUE. "ET" also immediately stops increasing.

- When "Q" becomes TRUE, the timing error range for "PT" is -100 ns to (100 ns + one task cycle). This error range includes the following cases:
  - ±100 ns is the timing error for "ET".
  - For each task cycle, "ET" is judged to check if it reaches "PT". If "ET" reaches "PT", there is a delay of one task cycle.
- "ET" and "Q" are updated when this instruction is executed. Strictly speaking, the condition for "Q" becoming TRUE is not when the cumulative time of the timer equals "PT", but rather when the cumulative time reaches "PT" and the current instruction is executed for the first time. Therefore, a maximum delay of one task cycle may occur.

18.1.9.14 Timer

100-ms timer.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
Timer	100-ms timer	FC		:= Timer(IN := , PT := , Q => , ET =>);

- **Related variable**

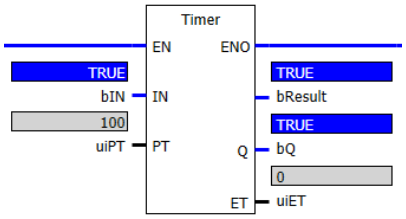
  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Timer input	Follows data type	-	TRUE: Timer is active. FALSE: Timer is stopped.
	PT	Set time	Follows data type	-	Maximum value for timing, in increments of 100 ms
Output	Q	Timer output	Follows data type	-	TRUE: "ET" reaches "PT". FALSE: "ET" has not reached "PT".
	ET	Remaining time	Follows data type	-	Remaining time

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PT	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ET	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

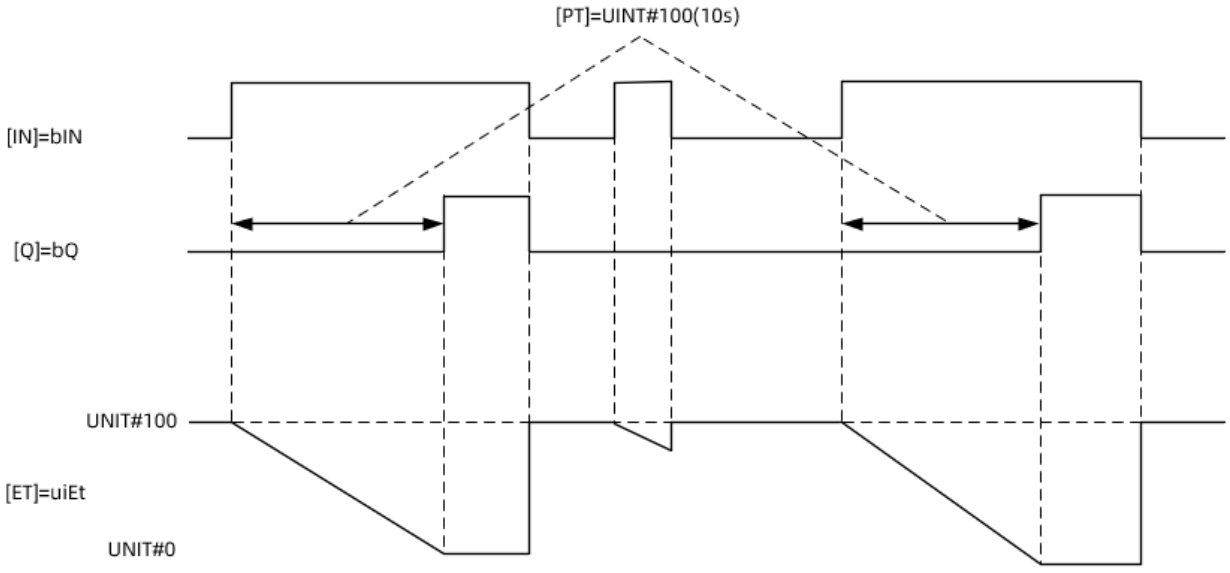
- **Function**
  - This instruction outputs TRUE when the set time elapses after the timer starts. The time is set in increments of 100 ms.
  - The timer is reset when timer input value of "IN" changes to FALSE. Remaining time "ET" is set to set time "PT", and timer output "Q" changes to FALSE. The timer starts when "IN" changes to TRUE. The value of "ET" is constantly decremented as time elapses. When "ET" reaches 0, timer output "Q" changes to TRUE. ET is not decremented after that. The timer is reset if "IN" changes to FALSE after the timer starts but before "ET" reaches 0.
- **Sequence diagram**
  - The following table shows an example when "PT" = UINT#100.

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">
<pre>
VAR
    bResult: BOOL;
    bIN: BOOL := TRUE;
    uiPT: UINT := 100;
    bQ: BOOL;
    uiET: UINT;
END_VAR
</pre>
</td></tr>
<tr><td>Program</td><td>


</td><td>
<pre>
bResult TRUE := Timer(IN := bIN TRUE,
        PT := uiPT 100,
        Q => bQ TRUE,
        ET => uiET 0);RETURN
</pre>
</td></tr>
<tr><td>Simulation results</td><td colspan="2">


</td></tr>
</table>

- The timing diagram when "PT" = UINT#100 is shown in the figure below. The value of "Q" becomes TRUE 10s after "IN" becomes TRUE, and the value of "ET" decreases to 0.

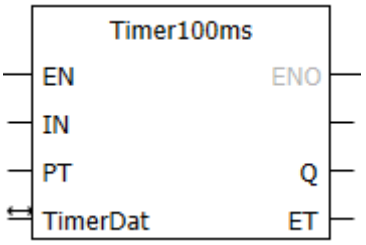


- **Precautions**
  - If multiple timers are called within the same POU program block, program faults may occur.
  - There is a one-task-cycle delay for "Q" to become TRUE after the timer reaches "PT". The timer checks at each task cycle whether "ET" has reached "PT". If so, "Q" is set to TRUE after a delay of one task cycle.
  - The timer starts as soon as operation starts if the value of "IN" has been TRUE.
  - If the value of "PT" changes, the new value needs to be used from the next time when the timer is reset. It does not take effect while timing is in progress.

18.1.9.15 Timer100ms

100-ms timer.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
Timer100ms	100-ms timer	FC		:= Timer100ms(IN := , PT := , TimerDat := , Q => , ET =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input/Output	TimerDat	Timer status	-	-	Current status of the timer
Input	IN	Timer input	TRUE, FALSE	-	TRUE: Timer is active. FALSE: Timer is stopped.
	PT	Set time	Follows data type	-	Elapsed time from timer start to "Q" being TRUE, measured in 100-ms units.
Output	Q	Timer output	TRUE, FALSE	-	TRUE: Timer execution completed. FALSE: Timer execution not completed.
	ET	Remaining time	Follows data type	-	"ET" = "PT" - elapsed time. "ET" is not less than 0, measured in 100-ms units.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
TimerDat	_sTimer																			
IN	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
PT	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ET	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

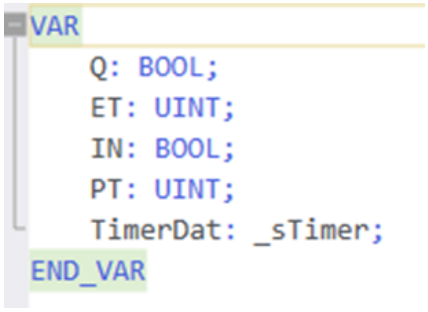
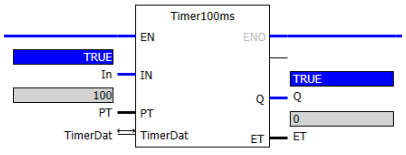
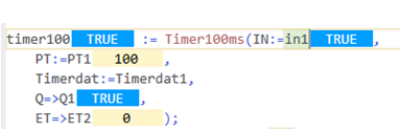
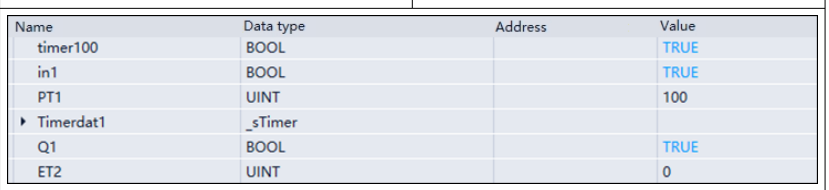
- **Function description**

  This instruction implements a timer function that outputs TRUE after a preset time from timer start. The specific principle is as follows:

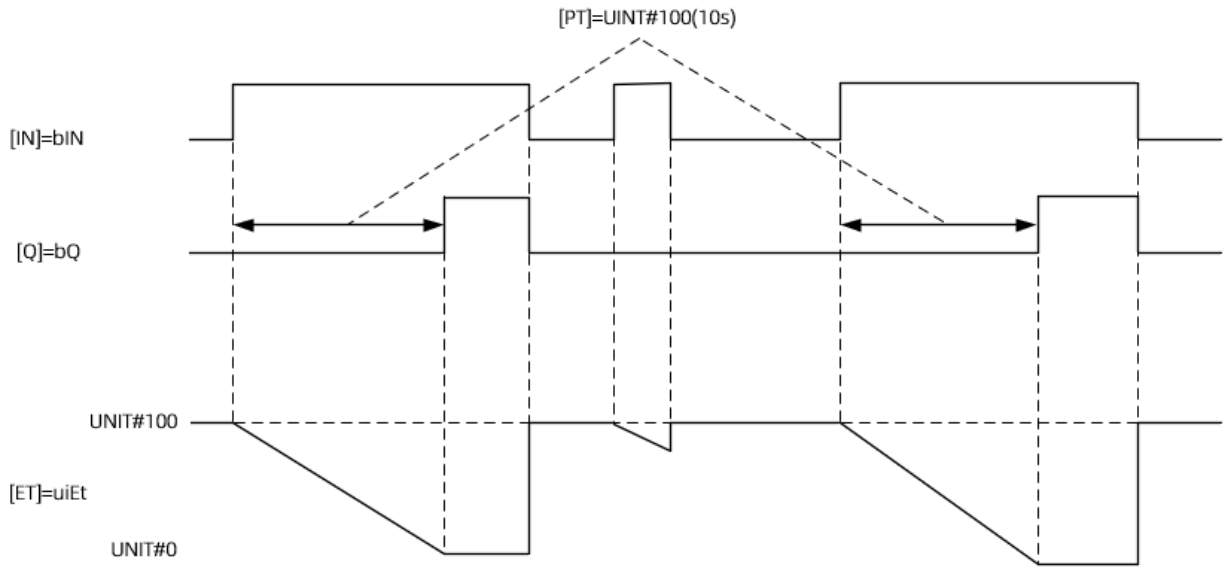
  - When "IN" is set to TRUE, the timer starts, and the remaining time (ET) begins to decrease continuously from the preset time (PT) until "ET" reaches 0, at which point the timer output (Q) becomes TRUE.
  - If "IN" is set to FALSE during timer operation, the timer resets, the output remaining time becomes "PT", and the output becomes FALSE; when "IN" is set to TRUE again, the timer restarts.

- **Sequence diagram**

  - The following table shows an example when "PT" = UINT#100.

	LD	ST
Define variables		
Program		
Operation result		

  - The timing diagram when "PT" = UINT#100 is shown in the figure below. The value of "Q" becomes TRUE 10s after "IN" becomes TRUE, and the value of "ET" decreases to 0.



- **Notes**
  - Multiple instances of this instruction can be called simultaneously.
  - The TimerDat parameter must be passed a variable that has not been initialized; it is prohibited to share the same TimerDat variable among multiple instructions.
  - When "IN" is set to TRUE, timing starts immediately. If the instruction is not executed due to program reasons, "ET" is not updated, but timing continues. When the instruction is executed, "ET" becomes the time elapsed since the start.
    For example: When the instruction (PT = 10s) is executed 5s after "IN" is set to TRUE, "ET" remains at 10s during this period and becomes 5s upon execution.
  - During the execution of this instruction, changes to "PT" are not effective; the modified "PT" will take effect the next time this instruction is executed.

18.1.10 String

18.1.10.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.1.10.2 CONCAT	FC	String concatenation	Standard	Library version 1.0.0.0 and later
18.1.10.3 DELETE	FC	String deletion		Library version 1.0.0.0 and later
18.1.10.4 FIND	FC	Get index of string in searched string		Library version 1.0.0.0 and later
18.1.10.5 INSERT	FC	String insertion		Library version 1.0.0.0 and later
18.1.10.6 LEFT	FC	Return string from left		Library version 1.0.0.0 and later
18.1.10.7 LEN	FC	Get string length		Library version 1.0.0.0 and later

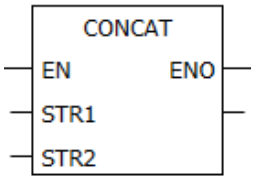
Instruction	FB/FC	Function	Library	Applicable Library Version
18.1.10.8 MID	FC	Return string from middle		Library version 1.0.0.0 and later
18.1.10.9 REPLACE	FC	This instruction replaces the string.		Library version 1.0.0.0 and later
18.1.10.10 RIGHT	FC	Return string from right		Library version 1.0.0.0 and later
18.1.10.11 WConcat	FC	Wide string concatenation	Standard64	Library version 1.0.0.0 and later
18.1.10.12 WDelete	FC	Wide string deletion		Library version 1.0.0.0 and later
18.1.10.13 WFind	FC	Wide string search		Library version 1.0.0.0 and later
18.1.10.14 WInsert	FC	Wide string insertion		Library version 1.0.0.0 and later
18.1.10.15 WLeft	FC	Return wide string from left		Library version 1.0.0.0 and later
18.1.10.16 WLen	FC	Get the length of the input wide string.		Library version 1.0.0.0 and later
18.1.10.17 WMid	FC	Return wide string from middle		Library version 1.0.0.0 and later
18.1.10.18 WReplace	FC	Wide string replacement		Library version 1.0.0.0 and later
18.1.10.19 WRight	FC	Return wide string from right		Library version 1.0.0.0 and later
18.1.10.20 AddDelimiter	FC	Write REAL-type array elements into a delimited string	OMUtils	Library version 1.0.0.0 and later
18.1.10.21 AddDelimiter_LR	FC	Write LREAL-type array elements into a delimited string		Library version 1.0.0.0 and later
18.1.10.22 ClearString	FC	Clear string		Library version 1.0.0.0 and later
18.1.10.23 GetByteLen	FC	String byte length calculation		Library version 1.0.0.0 and later
18.1.10.24 SubDelimiter	FC	Reads data from a delimited real number string		Library version 1.0.0.0 and later
18.1.10.25 SubDelimiter_LR	FC	Reads data from a delimited long real number string		Library version 1.0.0.0 and later
18.1.10.26 TodToString	FC	Time-to-string conversion		Library version 1.0.0.0 and later
18.1.10.27 ToLCase	FC	Convert to lowercase		Library version 1.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.1.10.28 ToUCase	FC	Convert to uppercase		Library version 1.0.0.0 and later
18.1.10.29 TrimL	FC	Trim string from the left		Library version 1.0.0.0 and later
18.1.10.30 TrimR	FC	Trim string from the right		Library version 1.0.0.0 and later

18.1.10.2 CONCAT

Concatenates the two input strings and outputs from the right.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CONCAT	String concatenation	FC		:= CONCAT(STR1 := , STR2 :=);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	STR1	Source data 1	Follows data type	-	Source data 1
	STR2	Source data 2	Follows data type	-	Source data 2
Output	OUT	Return value	Follows data type	-	Concatenated string

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
STR1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
STR2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

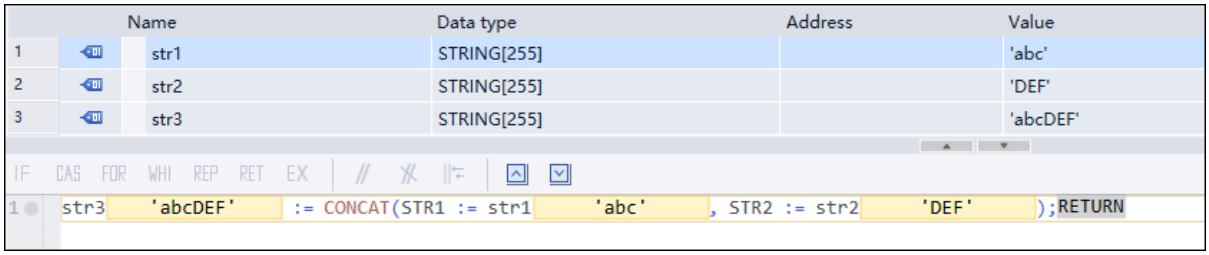
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**

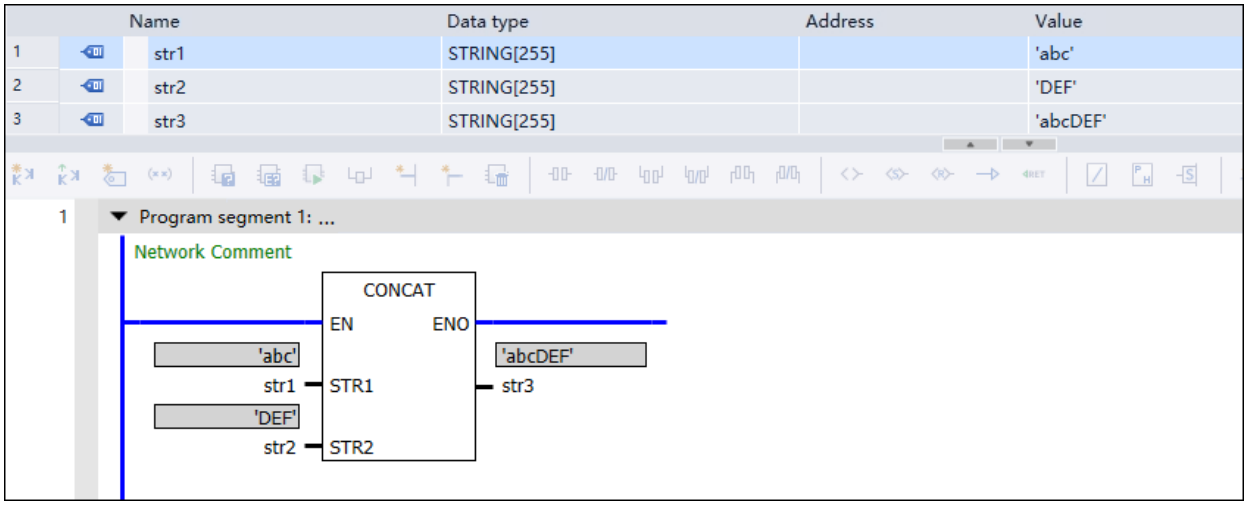
  Concatenates input strings STR1 and STR2, and outputs the result.

- **Program example**

  ST:



  LD:



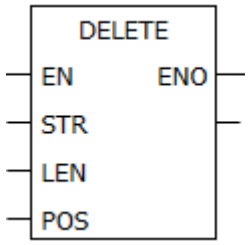
- **Precautions**

  If the length of the concatenated string exceeds 1985, the string will be truncated.

18.1.10.3 DELETE

Deletes a specified-length string at a specified position from the input string, and outputs the processed string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DELETE	String deletion	FC		:= DELETE(STR := , LEN := , POS :=);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	STR	Source data	Follows data type	-	Source data
	LEN	Data length	Follows data type	-	Data length
	POS	Data position	Follows data type	-	Data position
Output	OUT	Return value	Follows data type	-	Target string

Variable types

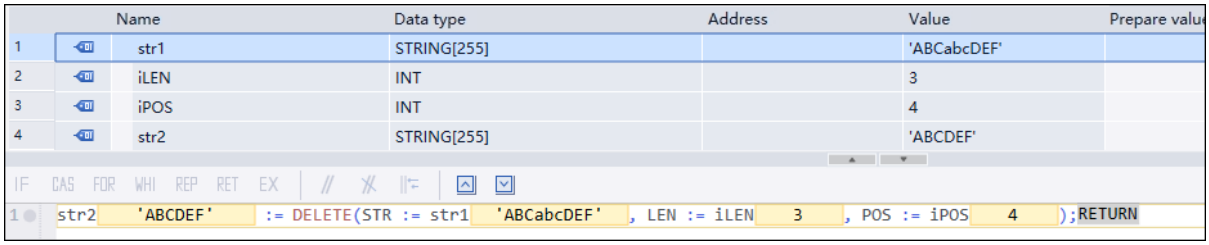
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
STR	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
LEN	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
POS	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**

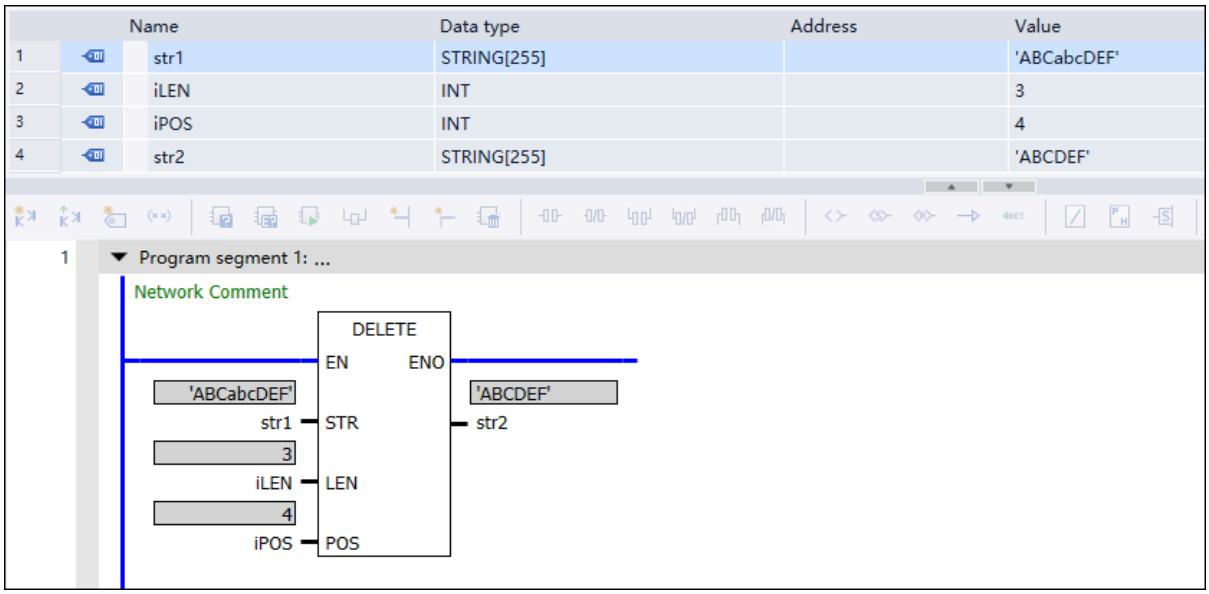
Deletes a string of length LEN from position POS in the input string STR, and generates a new string as the return value.

- **Program example**

ST:



LD:



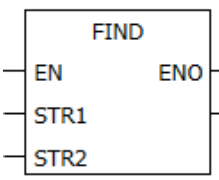
- **Precautions**

If POS = 0, deletion starts from the first character.

18.1.10.4 FIND

Searchs for the position of the target string within the input source string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FIND	Get index of string in searched string	FC		:= FIND(STR1 := , STR2 :=);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	STR1	Source data 1	Follows data type	-	Source data

Input/Output	Variable	Name	Valid Range	Default Value	Description
	STR2	Source data 2	Follows data type	-	Target string
Output	OUT	Return value	Follows data type	-	Position of the target string within the source data

Variable types

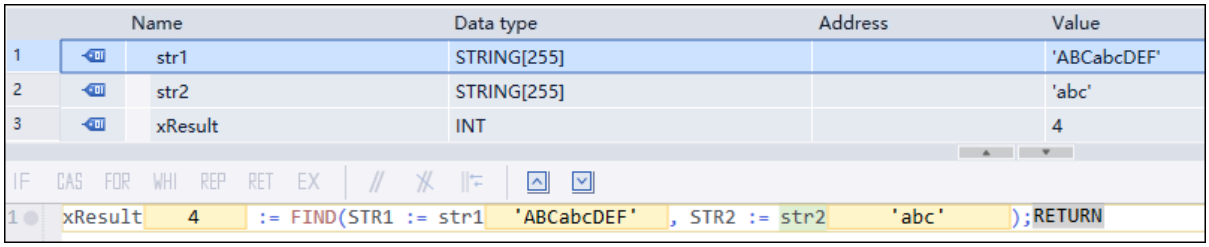
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
STR1	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	√
STR2	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	√
OUT	_	_	_	_	_	_	_	_	_	_	√	_	_	_	_	_	_	_	_	_

- **Function**

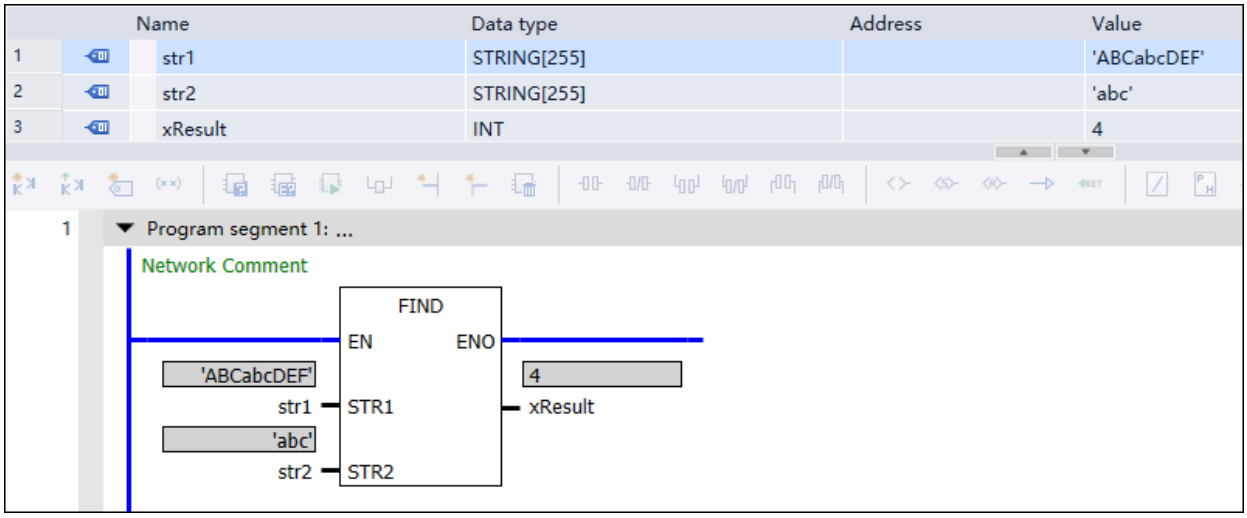
  Searchs for string STR2 within the input string STR1 and returns the position of the first character of STR2 in STR1.

- **Program example**

  ST:



  LD:

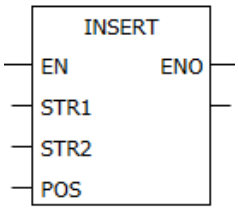


- **Precautions**
  - If the target string does not exist, the return value is 0.
  - The maximum supported string length is 1985.

18.1.10.5 INSERT

Inserts a string at the specified position in the input source string, and outputs the result string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
INSERT	String insertion	FC		:= INSERT(STR1 := , STR2 := , POS :=);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	STR1	Source data	Follows data type	-	Source data 1
	STR2	Source data	Follows data type	-	Source data 2
	POS	Insertion position	Follows data type	-	Data position
Output	OUT	Return value	Follows data type	-	Result string

Variable types

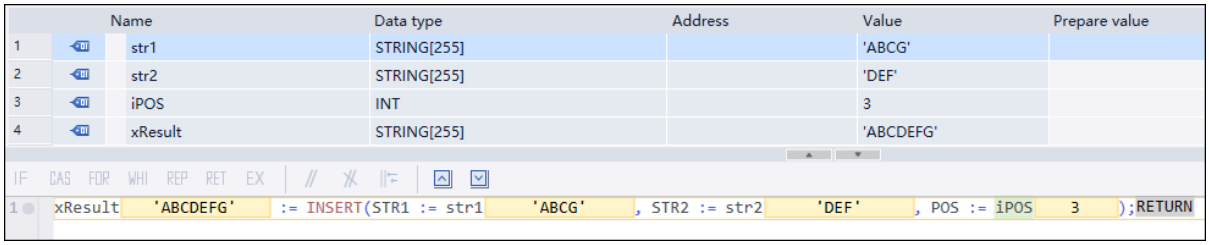
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
STR1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
STR2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
POS	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**

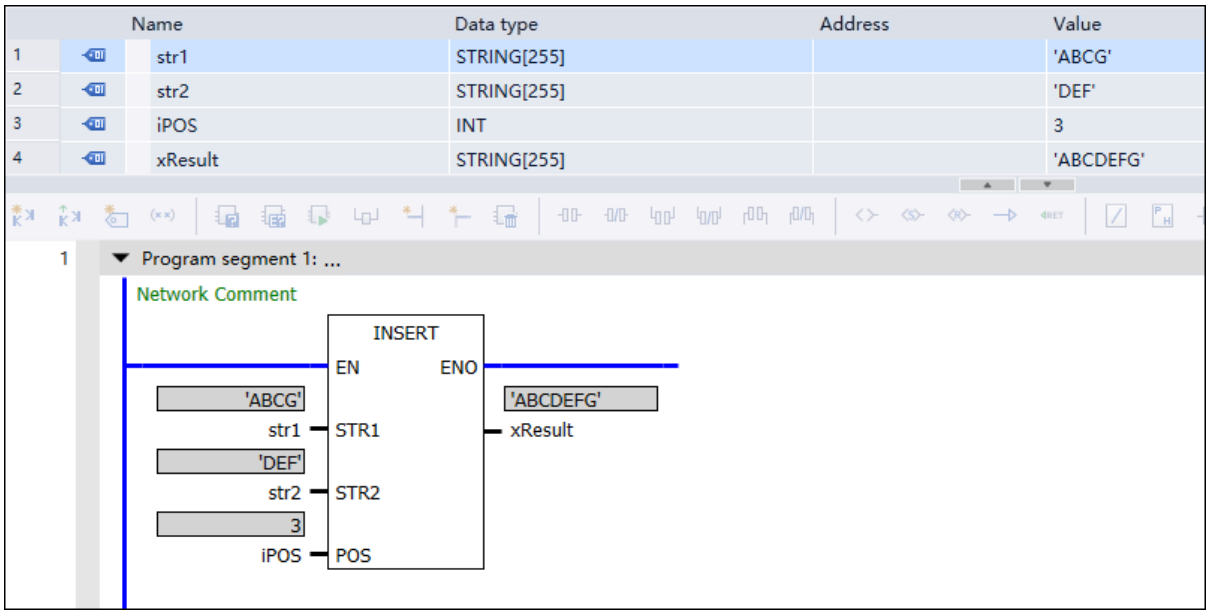
  Inserts STR2 after the POS-th character of input string STR1, and generates a new string as the output.

- **Program example**

  ST:



  LD:



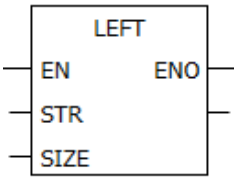
- **Precautions**
  - If POS equals 0, STR2 is inserted before STR1.

- The maximum supported string length is 1985.
- If POS is greater than the string length, the output result is the data corresponding to STR1.

18.1.10.6 LEFT

Returns the string of the specified length from the left of the source string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LEFT	Return string from left	FC		:= LEFT(STR := , SIZE :=);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	STR	Source data	Follows data type	-	Source data
	SIZE	Data length	Follows data type	-	Data length
Output	OUT	Return value	Follows data type	-	Target string

Variable types

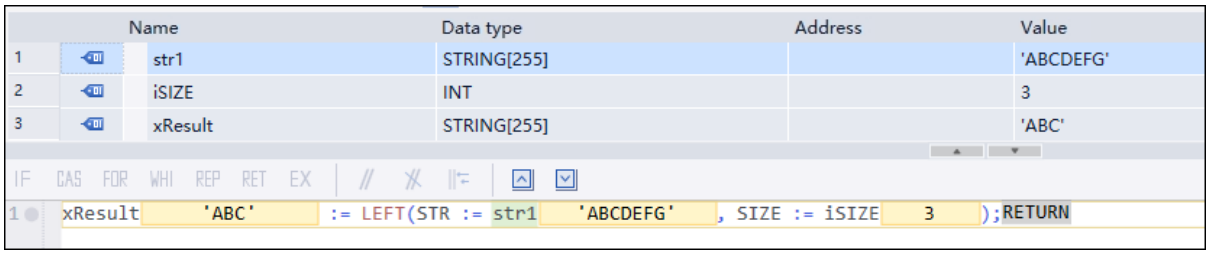
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
STR	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	√
SIZE	_	_	_	_	_	_	_	_	_	_	√	_	_	_	_	_	_	_	_	_
OUT	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	√

- **Function description**

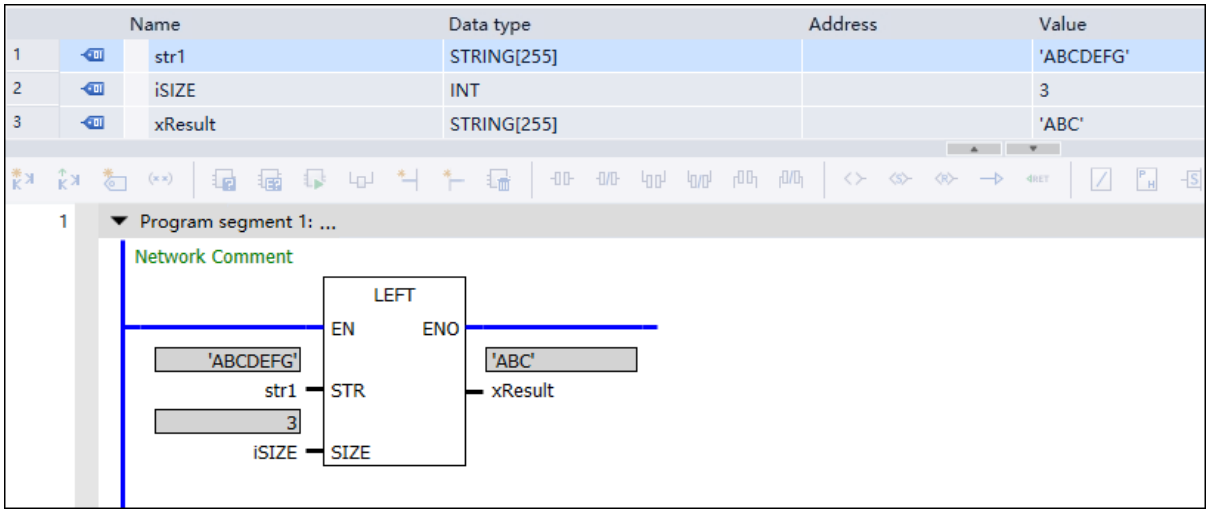
Takes a string from the left of STR and returns the string of the specified SIZE length.

- **Program example**

  ST:



  LD:

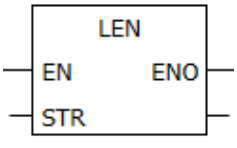


- **Precautions**
  - If the input SIZE is 0, the output string is empty.
  - The maximum supported string length is 1985.
  - If the input SIZE is greater than the length of the input string, the output string is the original input string.

18.1.10.7 LEN

Gets the number of characters in the target string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LEN	Get string length	FC		:= LEN(STR :=);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	STR	Source data	Follows data type	-	Target string

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	OUT	Return value	Follows data type	-	Number of characters in the target string

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
STR	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
OUT	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

- **Function**

  Gets the number of characters in the target string.

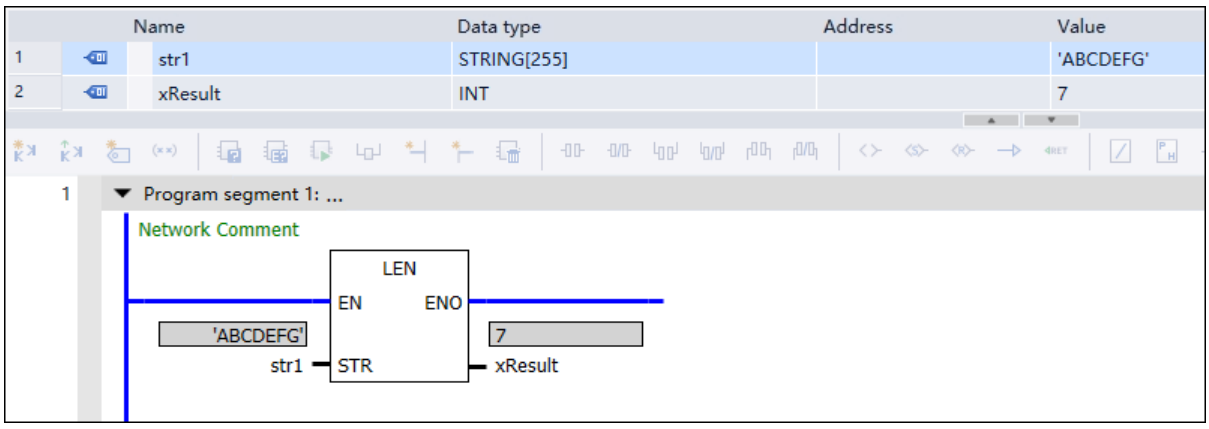
- **Program example**

  ST:

```
xResult := LEN(STR := str1);
RETURN
```

	Name	Data type	Address	Value
1	str1	STRING[255]		'ABCDEFG'
2	xResult	INT		7

  LD:



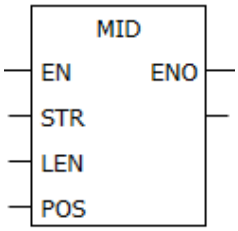
- **Precautions**
  - The maximum supported string length is 1985.
  - If the string size is not defined, the maximum string length is 80.

◦ Spaces within the string also occupy one character length.

18.1.10.8 MID

Gets a string of specified length from a specified position in the source string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
MID	Return string from middle	FC		:= MID(STR := , LEN := , POS :=);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	STR	Source data	Follows data type	-	Source data
	LEN	Data length	Follows data type	-	Data length
	POS	Target position	Follows data type	-	Position to extract string
Output	OUT	Return value	Follows data type	-	Target string

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
STR	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
LEN	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
POS	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
OUT	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	-	√

- **Function description**

  Extracts a string of specified length LEN from the specified position POS in the input string STR as the output string.

- **Program example**

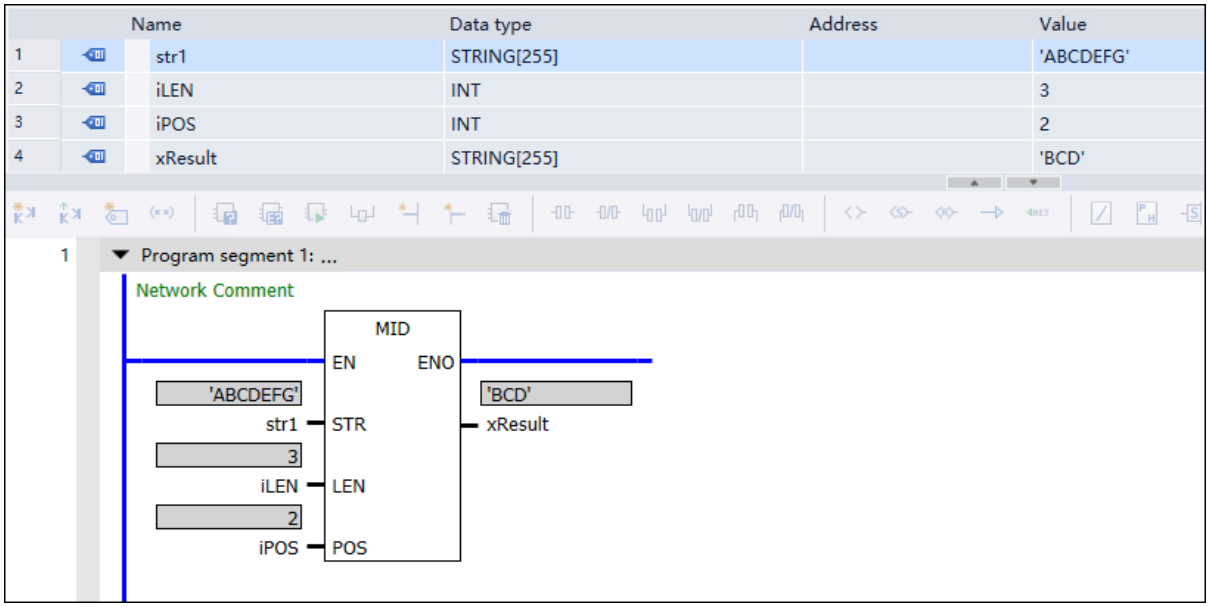
  ST:

	Name	Data type	Address	Value	Prepare va
1	str1	STRING[255]		'ABCDEFG'	
2	iLEN	INT		3	
3	iPOS	INT		2	
4	xResult	STRING[255]		'BCD'	

```
1  xResult := MID(STR := str1, LEN := iLEN, POS := iPOS);RETURN
```

  LD:

	Name	Data type	Address	Value
1	str1	STRING[255]		'ABCDEFG'
2	iLEN	INT		3
3	iPOS	INT		2
4	xResult	STRING[255]		'BCD'

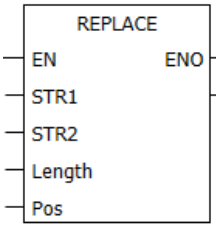


- **Precautions**
  - The maximum supported string length is 1985.
  - If the specified position POS exceeds the string length, no output will be produced.
  - If the specified length LEN exceeds the remaining length of the input string from the specified position POS, the function behaves the same as RIGHT.

18.1.10.9 REPLACE

Replaces a portion of the original string with the target string, and returns the new string generated after the replacement.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
REPLACE	String replacement	FC		:= REPLACE(STR1 := , STR2 := , Length := , Pos :=);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	STR1	Source data 1	Max length: 255	-	Source data 1
	STR2	Source data 2	Max length: 255	-	Source data 2
	Length	Data length	Follows data type	-	Data length
	Pos	Data position	Follows data type	-	Data position
Output	OUT	Return value	Max length: 255	-	Target string

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
STR1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
STR2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
Length	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

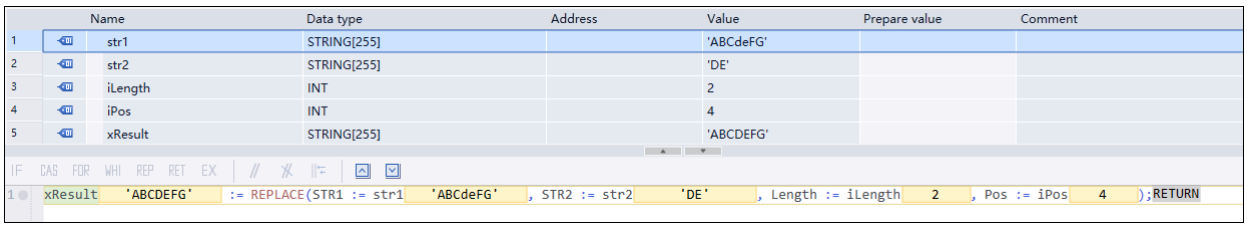
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Pos	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**

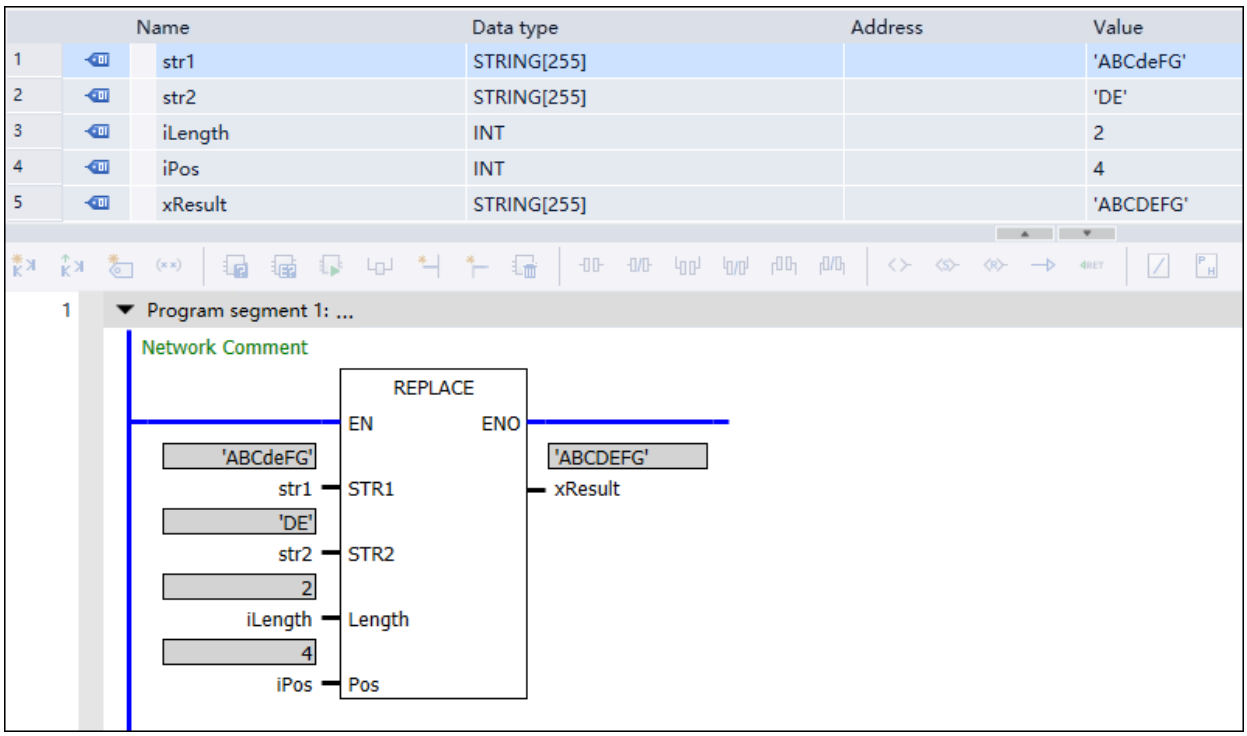
  Replaces Length characters in STR1 starting at position Pos with STR2, and returns the new string after replacement.

- **Program example**

  ST:



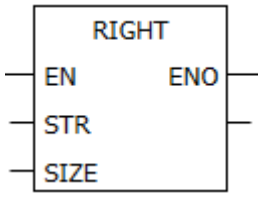
  LD:



18.1.10.10 RIGHT

Returns the string of the specified length from the right of the source string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
RIGHT	Return string from right	FC		:= RIGHT(STR := , SIZE :=);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	STR	Source data	Follows data type	-	Source data
	SIZE	Data length	0 to 255	-	Data length
Output	OUT	Return value	Follows data type	-	Target string

  Variable types

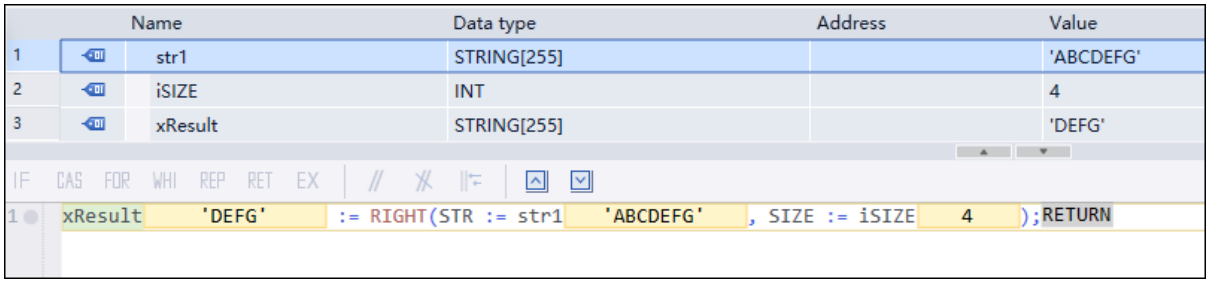
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
STR	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
SIZE	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**

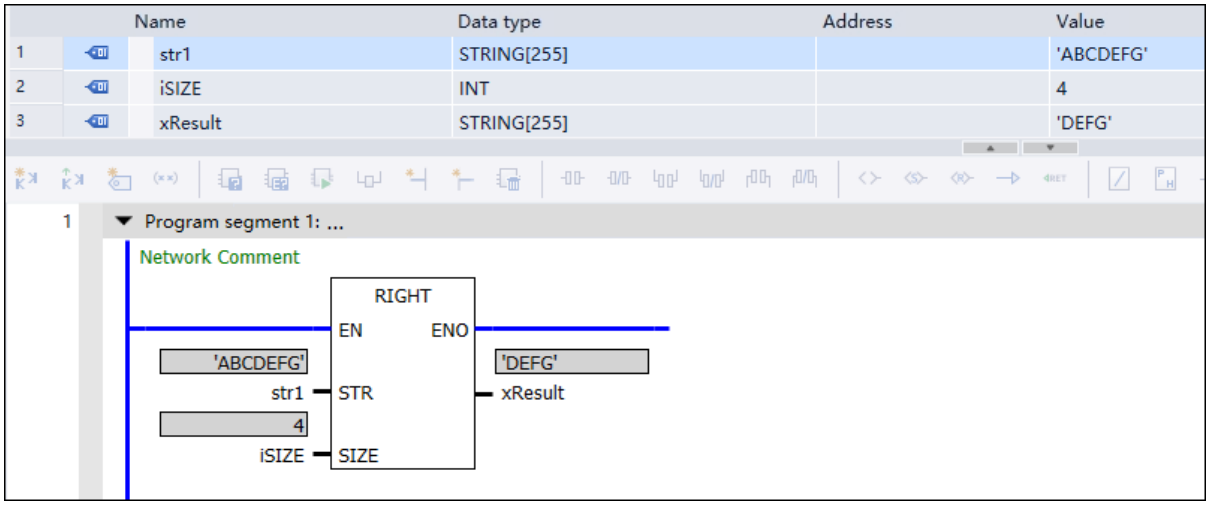
  Takes a string from the right of STR and returns the string of the specified SIZE length.

- **Program example**

  ST:



LD:

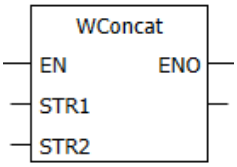


- **Precautions**
  - If the input SIZE is 0, the output string is empty.
  - The maximum supported string length is 1985.
  - If the input SIZE is greater than the length of the input string, the output string is the original input string.

18.1.10.11 WConcat

Concatenates two wide strings and outputs from the right.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
WConcat	Wide string concatenation	FC		:= WConcat(STR1 := , STR2 :=);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	STR1	Source data 1	Max length: 1985	-	Source data 1
	STR2	Source data 2	Max length: 1985	-	Source data 2
Output	OUT	Return value	Max length: 1985	-	Concatenated string

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
STR1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
STR2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**

  Concatenates input wide strings STR1 and STR2, and outputs the result.

- **Program example**

  WConcat is similar to 18.1.10.2 CONCAT in the standard library, except that STR1, STR2, and UT change from STRING type to WSTRING type, thus providing a larger representation range.

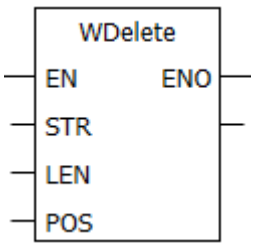
- **Precautions**

  If the length of the concatenated wide string exceeds 1985, the string will be truncated.

18.1.10.12 WDelete

Deletes a specified-length string at a specified position from the input wide string, and outputs the processed wide string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
WDelete	Wide string deletion	FC		:= WDelete(STR := , LEN := , POS :=);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	STR	Source data	Max length: 1985	-	Source data

Input/Output	Variable	Name	Valid Range	Default Value	Description
	LEN	Data length	Follows data type	-	Data length
	POS	Data position	Follows data type	-	Data position
Output	OUT	Return value	Max length: 1985	-	Target string

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
STR	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
LEN	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–
POS	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**

  Deletes a string of length LEN from position POS in the input wide string STR, and generates a new wide string as the return value.

- **Program example**

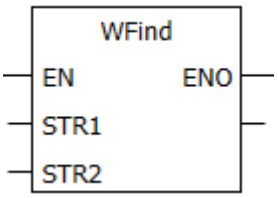
  WDelete is similar to 18.1.10.3 DELETE in the standard library, except that "STR" and "OUT" change from STRING type to WSTRING type, thus providing a larger representation range.

- **Precautions**
  - If POS = 0, deletion starts from the first character.
  - The maximum supported string length is 1985.

18.1.10.13 WFind

Searchs for the position of the target wide string within the input source wide string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
WFind	Wide string search	FC		:= WFind(STR1 := , STR2 :=);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	STR1	Source data 1	Max length: 1985	-	Source data
	STR2	Source data 2	Max length: 1985	-	Target string
Output	OUT	Return value	Follows data type	-	Position of the target string within the source data

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
STR1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
STR2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
OUT	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–

- **Function**

  Searchs for wide string STR2 within the input wide string STR1 and returns the position of the first character of STR2 in STR1.

- **Program example**

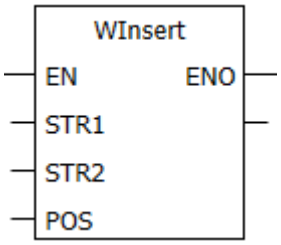
WFind is similar to 18.1.10.4 FIND in the standard library, except that STR1 and STR2 change from STRING type to WSTRING type, thus providing a larger representation range.

- **Precautions**
  - If the target string does not exist, the return value is 0.
  - The maximum supported string length is 1985.

18.1.10.14 WInsert

Inserts a string at the specified position in the input source wide string, and outputs the result string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
WInsert	Wide string insertion	FC		:= WInsert(STR1 := , STR2 := , POS :=);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	STR1	Source data 1	Max length: 1985	-	Source data 1
	STR2	Source data 2	Max length: 1985	-	Source data 2
	POS	Insertion position	Follows data type	-	Insertion position
Output	OUT	Return value	Max length: 1985	-	Result string

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
STR1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
STR2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
POS	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**

  Inserts STR2 after the POS-th character of the input wide string STR1, and generates a new wide string as the output.

- **Program example**

  WInsert is similar to 18.1.10.5 INSERT in the standard library, except that "STR1", "STR2", and "OUT" change from STRING type to WSTRING type, thus providing a larger representation range.

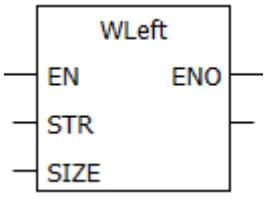
- **Precautions**

  - If POS = 0, then STR2 is inserted before STR1.
  - The maximum supported string length is 1985.
  - If POS is not less than the string length, the output result is the data corresponding to STR1.

18.1.10.15 WLeft

Returns the wide string of the specified length from the left of the source wide string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
WLeft	Return wide string from left	FC		:= WLeft(STR := , SIZE :=);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	STR	Source data	Max length: 1985	-	Source data
	SIZE	Data length	Follows data type	-	Target string
Output	OUT	Return value	Max length: 1985	-	Target string

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
STR	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
SIZE	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**

  Takes a wide string from the left of STR and returns the string of the specified SIZE length.

- **Program example**

  WLeft is similar to 18.1.10.6 LEFT in the standard library, except that "STR" and "OUT" change from STRING type to WSTRING type, thus providing a larger representation range.

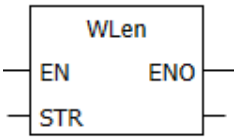
- **Precautions**

  - If the input SIZE is 0, the output string is empty.
  - The maximum supported string length is 1985.
  - If the input SIZE is greater than the length of the input string, the output string is the original input string.

18.1.10.16 WLen

Gets the number of characters in the target wide string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
WLen	Get the length of the input wide string.	FC		:= WLen(STR :=);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	STR	Source data	Max length: 1985	-	Target string
Output	OUT	Return value	Follows data type	-	Number of characters in the target string

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
STR	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
OUT	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–

- **Function**

  Gets the number of characters in the target wide string.

- **Program example**

  WLen is similar to 18.1.10.7 LEN in the standard library, except that STR changes from STRING type to WSTRING type, thus providing a larger representation range.

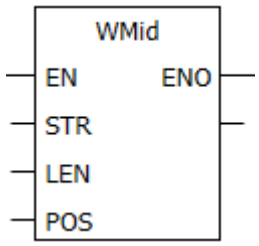
- **Precautions**

  - If the string size is not defined, the maximum string length is 80.
  - Spaces within the string also occupy one character length.

18.1.10.17 WMid

Returns a wide string of specified length from a specified position in the source wide string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
WMid	Return wide string from middle	FC		:= WMid(STR := , LEN := , POS :=);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	STR	Source data	Max length: 1985	-	Source data
	LEN	Data length	Follows data type	-	Data length
	POS	Target position	Follows data type	-	Position to extract string
Output	OUT	Return value	Max length: 1985	-	Target string

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
STR	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
LEN	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–
POS	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**

Extracts a wide string of specified length LEN from the specified position POS in the input wide string STR as the output string.

- **Program example**

  WMid is similar to 18.1.10.8 MID in the standard library, except that "STR" and "OUT" change from STRING type to WSTRING type, thus providing a larger representation range.

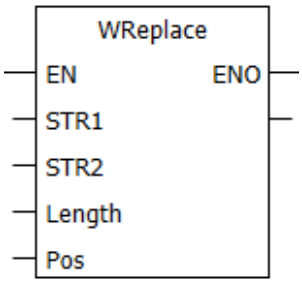
- **Precautions**

  - If the specified position POS exceeds the string length, no output will be produced.
  - If the specified length LEN exceeds the remaining length of the input string from the specified position POS, the function behaves the same as WRight.

18.1.10.18 WReplace

Replaces a portion of the original wide string with the target wide string, and returns the new wide string generated after the replacement.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
WReplace	Wide string replacement	FC		:= WReplace(STR1 := , STR2 := , Length := , Pos :=);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	STR1	Source data 1	Max length: 1985	-	Source data 1
	STR2	Source data 2	Max length: 1985	-	Source data 2
	Length	Data length	0 to 1985	-	Data length
	Pos	Data position	0 to 1985	-	Data position
Output	OUT	Return value	Max length: 1985	-	Target string

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
STR1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
STR2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
Length	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–
Pos	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**

  Replaces Length characters in STR1 starting at position Pos with STR2, and returns the new wide string after replacement.

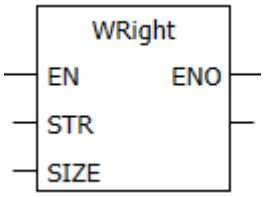
- **Program example**

  WReplace is similar to 18.1.10.9 REPLACE in the standard library, except that "STR1", "STR2", and "OUT" change from STRING type to WSTRING type, thus providing a larger representation range.

18.1.10.19 WRight

Returns the wide string of the specified length from the right of the source wide string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
WRight	Return wide string from right	FC		:= WRight(STR := , SIZE :=);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	STR	Source data	Max length: 1985	-	Source data
	SIZE	Data length	Follows data type	-	Data length
Output	OUT	Return value	Max length: 1985	-	Target string

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
STR	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
SIZE	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**

  Takes a wide string from the right of STR and returns the string of the specified SIZE length.

- **Program example**

  WRight is similar to 18.1.10.10 RIGHT in the standard library, except that "STR" and "OUT" change from STRING type to WSTRING type, thus providing a larger representation range.

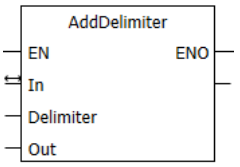
- **Precautions**

  - If the input SIZE is 0, the output string is empty.
  - If the input SIZE is greater than the length of the input string, the output string is the original input string.

18.1.10.20 AddDelimiter

Converts REAL-type array elements into a delimited string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AddDelimiter	Write REAL-type array elements into a delimited string	FC		:= AddDelimiter(In := , Delimiter := , Out :=);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Delimiter	Delimiter	Follows data type	_COMMA	Delimiter
	Out	Delimited string	Max length: 1985	-	Delimited string
Input/Output	In[ ] array	Input array	Follows data type	-	Array for conversion

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Delimiter	Enumeration type _eDELIMITER. For details, see function description for enumeration values.																			
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

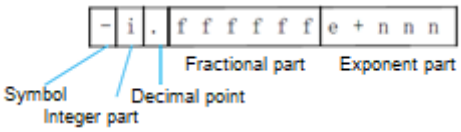
- **Function description**

  - Converts "Size" elements of the input array "In" starting from In[0] into strings sequentially, then concatenates them with the delimiter "Delimiter". Outputs the concatenated string to the return value "Out", with a NULL character at the end.
  - The data type of "Delimiter" is the enumeration type _eDELIMITER, whose meanings are as shown in the following table.

Enumerator	Definition
1 (_COMMA)	','
2 (_TAB)	'$T'
3 (_SEMICOLON)	';'

Enumerator	Definition
4 (_SPACE)	' '

◦ The return string after conversion is structured as shown below.



Item	Content
Symbol column	When the value of an array element is negative, "-" (minus sign) is added. When the value of an array element is positive, "+" (plus sign) is not added.
Integer part	Always represents with one digit.
Decimal point	Always represents the value of structural elements, even if not a decimal.
Fractional part	Represents with 6 digits.
Exponent	"e" represents the base "10" in scientific notation. The number of digits in nn is 3. When the absolute value of the array element value is greater than 1.0, the sign of nn is "+"; when the absolute value of the array element value is less than 1.0, the sign of nn is "-"; when the array element value is 0, the sign of nn is "+".

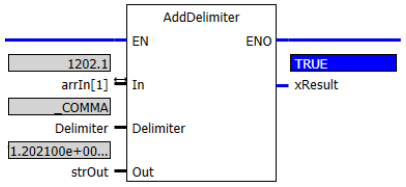
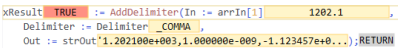
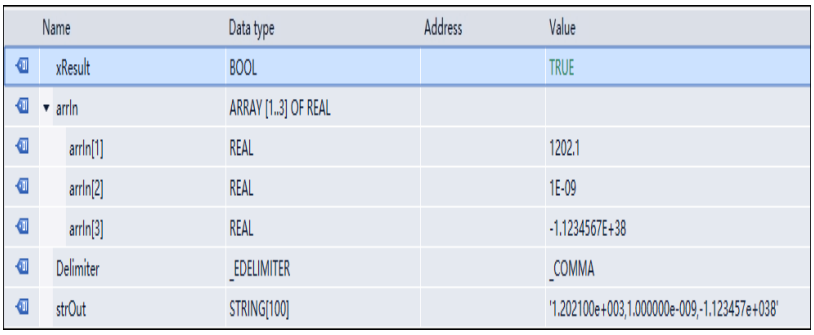
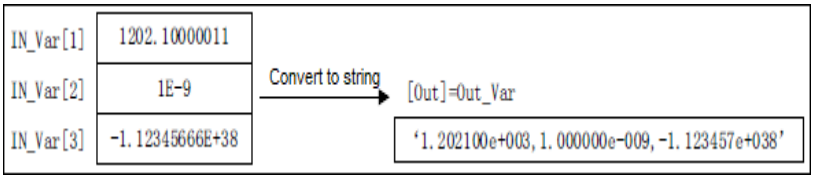
◦ When the array element value is infinity or non-numeric, the string is as shown in the table below.

Array Element Value	String
+∞	'infinity'
-∞	'-infinity'
Non-numeric	'nan'

◦ Examples are shown in the table below.

Array Element Value	Converted String
REAL#3.14e1	'3.140000e+001'
REAL#-123.4567	'-1.234567e+002'
REAL#0	'0.000000e+000'

- **Program example**

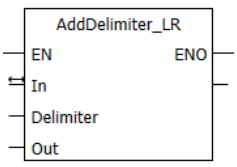
Item	LD	ST
Define variables	VAR xResult: BOOL; arrIn: ARRAY[1..3] OF REAL := [1202.100000000111,1E-9,-1.1234567E+38]; Delimiter: _eDELIMITER := _EDELIMITER._COMMA; strOut: STRING[100]; END_VAR	
Program		
Operation result		
Principle analysis		

- **Precautions**
  - When the byte length required for the conversion of the input "In" exceeds the defined byte length of output "Out", the actual output string consists of the maximum number of complete bytes that can be output within the defined byte length.
  - When the byte length required for the conversion of the input "In" exceeds 1985 (the maximum byte length defined for the output "Out"), the actual output string consists of the maximum number of complete bytes that can be output within a 1985-byte length.
  - The byte count of "Out" must not be less than 14; otherwise, the function returns FALSE, and the value of "Out" remains unchanged.

18.1.10.21 AddDelimiter_LR

Converts LREAL-type array elements into a delimited string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AddDelimiter_LR	Write LREAL-type array elements into a delimited string	FC		:= AddDelimiter_LR(In := , Delimiter := , Out :=);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Delimiter	Delimiter	Follows data type	_COMMA	Delimiter
	Out	Delimited string	Max length: 1985	-	Delimited string
Input/Output	In[ ] array	Input array	Follows data type	-	Array for conversion

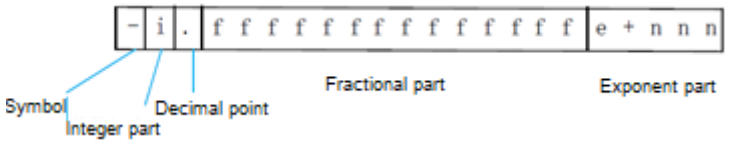
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	_	_	_	_	_	_	_	_	_	_	_	_	_	_	√	_	_	_	_	_
Delimiter	Enumeration type _eDELIMITER. For details, see function description for enumeration values.																			
Out	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	√

- **Function description**
  - Converts "Size" elements of the input array "In" starting from In[0] into strings sequentially, then concatenates them with the delimiter "Delimiter". Outputs the concatenated string to the return value "Out", with a NULL character at the end.
  - The data type of "Delimiter" is the enumeration type _eDELIMITER. The meaning of the enumeration is as shown in the following table.

Enumerator	Definition
1 (_COMMA)	','
2 (_TAB)	'$T'
3 (_SEMICOLON)	';'
4 (_SPACE)	' '

  - The return string after conversion is structured as shown below.



Item	Content
Symbol column	When the value of an array element is negative, "-" (minus sign) is added. When the value of an array element is positive, "+" (plus sign) is not added.
Integer part	Always represents with one digit.
Decimal point	Always represents the value of structural elements, even if not a decimal.
Fractional part	Represents with 14 digits.
Exponent	"e" represents the base "10" in scientific notation. The number of digits in nn is 3. When the absolute value of the array element value is greater than 1.0, the sign of nn is "+"; when the absolute value of the array element value is less than 1.0, the sign of nn is "-"; when the array element value is 0, the sign of nn is "+".

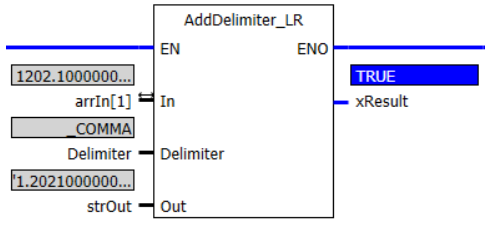
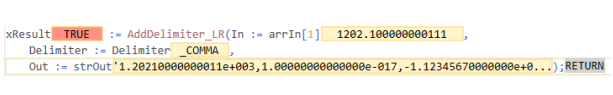
- When the array element value is infinity or non-numeric, the string is as shown in the table below.

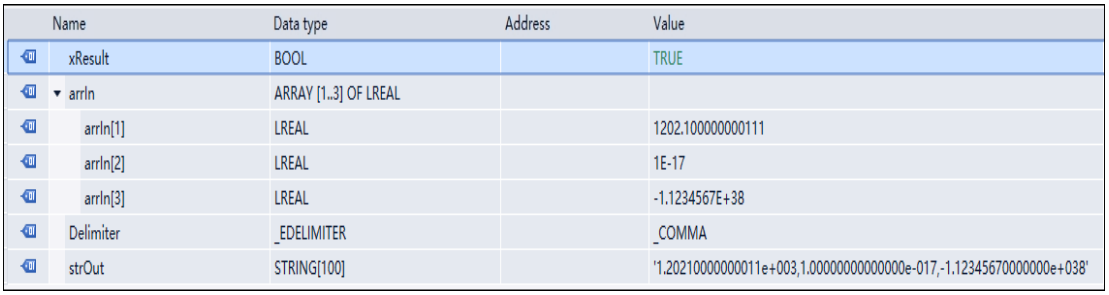
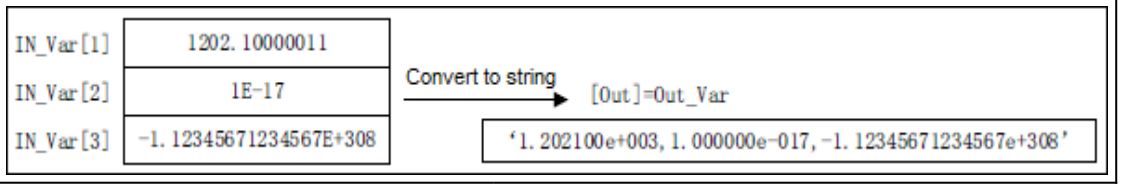
Array Element Value	String
+∞	'infinity'
-∞	'-infinity'
Non-numeric	'nan'

- Examples are shown in the table below.

Array Element Value	Converted String
LREAL#3.14e1	'3.14000000000000e+001'
LREAL#-123.4567	'-1.23456700000000e+002'
LREAL#0	'0.00000000000000e+000'

- **Program example**

Item	LD	ST
Define variables	VAR xResult: BOOL; arrIn: ARRAY[1..3] OF LREAL := [1202.100000000111,1E-17,-1.1234567E+38]; Delimiter: _eDELIMITER := _EDELIMITER._COMMA; strOut: STRING[100]; END_VAR	
Program		

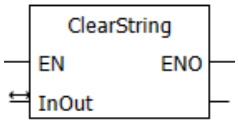
Item	LD	ST
Operation result		
Principle analysis		

- **Precautions**
  - When the byte length required for the conversion of the input "In" exceeds the defined byte length of output "Out", the actual output string consists of the maximum number of complete bytes that can be output within the defined byte length.
  - When the byte length required for the conversion of the input "In" exceeds 1985 (the maximum byte length defined for the output "Out"), the actual output string consists of the maximum number of complete bytes that can be output within a 1985-byte length.
  - The byte count of "Out" must not be less than 22; otherwise, the function returns FALSE, and the value of "Out" remains unchanged.

18.1.10.22 ClearString

Clears the input string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ClearString	Clear string	FC		:= ClearString(InOut :=);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input/Output	InOut	Clear string	Follows data type	-	The string to clear and output

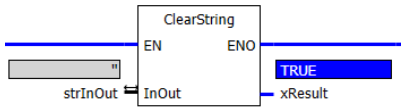
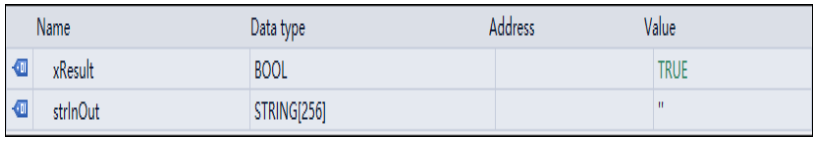
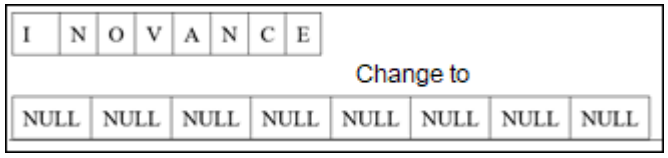
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**

  Clears the content of the string "InOut", and stores NULL characters in it.

- **Program example**

Item	LD	ST
Define variables	VAR xResult: BOOL; strInOut: STRING[256] := 'INOVANCE'; END_VAR	
Program		
Operation result		
Principle analysis		

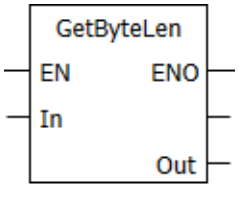
- **Precautions**

  The return value is always TRUE.

18.1.10.23 GetByteLen

Calculates the number of bytes in a string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetByteLen	String byte length calculation	FC		:= GetByteLen(In := , Out =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Counted object	Follows data type	-	The object string whose byte length is to be calculated
Output	Out	Number of bytes	0 to 1985	-	Number of bytes

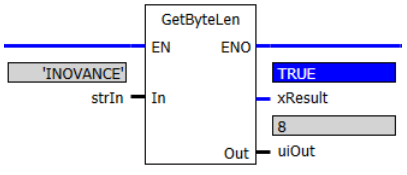
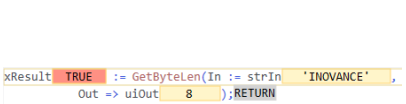
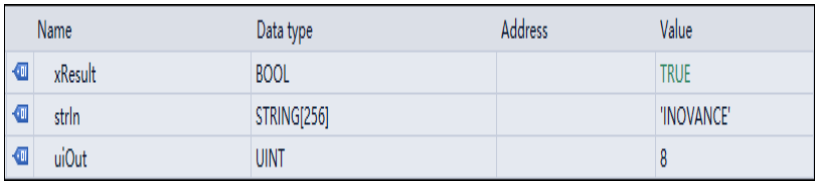
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
Out	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Calculates the number of bytes in the object string "In" from beginning to end; the NULL character at the end of the string is not included.

- **Program example**

Item	LD	ST
Define variables	VAR xResult: BOOL; strIn: STRING[256] := 'INOVANCE'; uiOut: UINT; END_VAR	
Program		
Operation result		
Principle analysis		

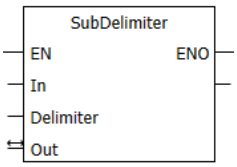
- **Precautions**

The length of the input string is limited by the data type STRING.

18.1.10.24 SubDelimiter

Reads data from a delimited string and outputs it as a real number array.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SubDelimiter	Reads data from a delimited real number string	FC		:= SubDelimiter(In := , Delimiter := , Out :=);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Input string	Maximum 1985 bytes	''	The delimited string to be read
	Delimiter	Delimiter	Follows data type	_COMMA	Delimiter
Input/Output	Out[ ] array	A real number array to store the return array	Follows data type	-	A real number array to store the results of data conversion

Variable types

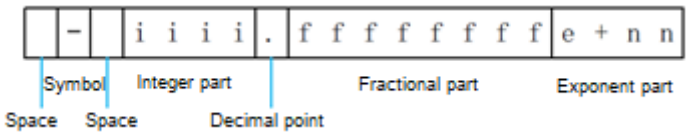
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Delimiter	Enumeration type _eDELIMITER. For details, see function description for enumeration values.																			
Out[ ] array	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
	Array of enumerations can also be specified.																			

- **Function**
  - Converts the strings separated by the delimiter "Delimiter" in the input string "In" into the elements of the real number array "Out", and saves them sequentially as values of each array element.
  - The data type of "Delimiter" is the enumeration type _eDELIMITER, and the meanings of the enumeration values are as follows.

Enumeration Value	Definition
_COMMA	','
_TAB	'$T'
_SEMICOLON	';'
_SPACE	' '

  - When the number of data items separated by delimiters in "In" exceeds the number of elements in "Out", the remaining data will be ignored.
  - When the number of data items separated by delimiters in "In" is less than the number of elements in "Out", the values of the missing elements remain unchanged.
  - The data in "In" separated by delimiters is of STRING type. The REAL-type string format is as follows.



Name	Format
Operator	With leading consecutive ' ' (spaces) ignored, a standalone '+' or '-' is treated as a sign. The '+' sign can be omitted, and consecutive spaces following the sign are ignored.
Integer part	The part from after the sign to before the decimal point is the integer part. Consecutive spaces after the sign are not included in the integer part, and the sign may be omitted. The integer part consists of numbers "0" to "9" and cannot be omitted. The maximum number of digits in the integer part is the maximum text string length of 1985 minus the total number of bytes in the following: the sign, decimal point, fractional part, exponent, and blank characters before and after the sign.

Name	Format
Decimal point	The decimal point is the single dot (".") following the integer part. When there is no fractional part, the decimal point must be omitted.
Fractional part	The part from after the decimal point to before the exponent part is the fractional part. When there is no exponent part, the entire string after the decimal point is the fractional part. The fractional part consists of numbers "0" to "9". The fractional part may be omitted. The maximum number of digits in the fractional part is 15. When there is no decimal point, the fractional part does not exist.
Exponent part	The exponent part is the part from the standalone 'e' following the fractional part and the standalone '+' or '-' after it, to the end of the string. When there is no fractional part, the string after the decimal point is considered the exponent part. When there is no decimal point or fractional part, the exponent part is the string following the integer part. The numeric part of the exponent consists of numbers "0" to "9". The exponent part may be omitted. The maximum number of digits in the exponent's numeric part is 3.

◦ The following table shows examples of conversion rules.

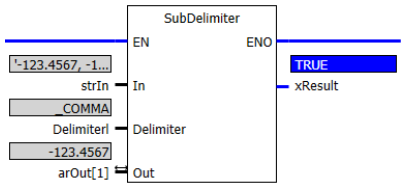
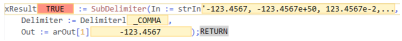
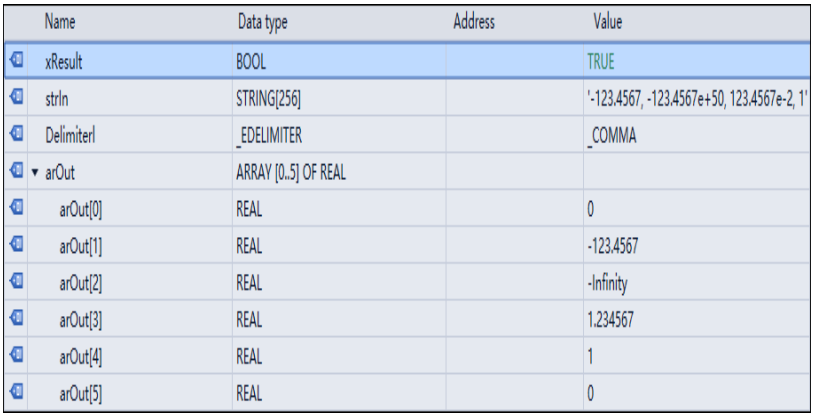
Format	STRING	REAL
Signed, with decimal point, with fractional part, no exponent part	'-123.4567'	-123.4567
Signed, with decimal point, with fractional part, with exponent part	'+123.4567e+02'	12345.67
Unsigned, with decimal point, with fractional part, with exponent part	'123.4567e-02'	1.234567
Unsigned, no decimal point, no fractional part, no exponent part	'1'	1

◦ Notes for conversion:

- When the value represented by the input string exceeds the valid range of the REAL data type, the conversion result is positive/negative infinity.
- When the input string does not represent a real number value, an exception occurs during conversion, and the function returns FALSE.
- When there are two or more consecutive '_' (underscore) characters at any intermediate position in the input string, conversion is successfully performed.
- When there is an '_' (underscore) between the '-' (minus sign) or '+' (plus sign) at the beginning of the input string, conversion is successfully performed.

- When an '_' (underscore) is at the beginning or end of the input string, conversion is successfully performed.
- When an '_' (underscore) is at any intermediate position in the input string, it is ignored.
- When the content of the input string exceeds the precision representable by the REAL data type, rounding is applied.
- When the content of the input string is closer to 0 than the minimum value representable by the REAL data type, the conversion result is 0.

- **Program example**

Item	LD	ST
Define variables	VAR xResult: BOOL; strIn: STRING[256] := '-123.4567, -123.4567e+50, 123.4567e-2, 1'; Delimiterl: _eDELIMITER := _eDELIMITER._COMMA; arOut: ARRAY[0..5] OF REAL; END_VAR	
Program		
Operation result		

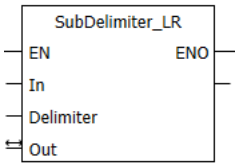
Name	Data type	Address	Value
xResult	BOOL		TRUE
strIn	STRING[256]		'-123.4567, -123.4567e+50, 123.4567e-2, 1'
Delimiterl	_EDELIMITER		_COMMA
▾ arOut	ARRAY [0..5] OF REAL		
arOut[0]	REAL		0
arOut[1]	REAL		-123.4567
arOut[2]	REAL		-Infinity
arOut[3]	REAL		1.234567
arOut[4]	REAL		1
arOut[5]	REAL		0

- **Precautions**
  - When the number of conversion results for "In" is large, set a larger task cycle; otherwise, it may lead to high CPU utilization and even PLC crash.
  - When there is more than one consecutive delimiter in "In", the delimited data does not exist. If nonexistent data is output as REAL#0 in the corresponding element of the Out[ ] array.
  - When "In" contains data that does not conform to the real number data format, no conversion is performed for the invalid data and subsequent data, the function returns FALSE, and the successfully converted values remain unchanged.
  - Do not use the delimiter in "In" for purposes other than separation. Even if used for other purposes, the instruction will still treat it as a delimiter.
  - When the conversion result exceeds the Out[ ] array area, the function returns FALSE, and the data within the range will be saved.

18.1.10.25 SubDelimiter_LR

Reads data from a delimited string and outputs it as a long real number array.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SubDelimiter_LR	Reads data from a delimited long real number string	FC		:= SubDelimiter_LR(In := , Delimiter := , Out :=);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Input string	Maximum 1985 bytes	''	The delimited string to be read
	Delimiter	Delimiter	Follows data type	_COMMA	Delimiter
Input/Output	Out[ ] array	A real number array to store the return array	Follows data type	-	A real number array to store the results of data conversion

Variable types

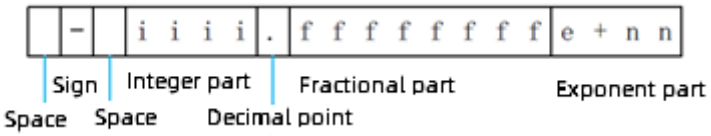
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
Delimiter	Enumeration type _eDELIMITER. For details, see function description for enumeration values.																			
Out[ ] array	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-

- **Function**
  - Converts the strings separated by the delimiter "Delimiter" in the input string "In" into the elements of the long real number array "Out", and saves them sequentially as values of each array element.

- The data type of "Delimiter" is the enumeration type _eDELIMITER, and the meanings of the enumeration values are as follows.

Enumeration Value	Definition
_COMMA	','
_TAB	'$T'
_SEMICOLON	';'
_SPACE	' '

- When the number of data items separated by delimiters in "In" exceeds the number of elements in "Out", the remaining data will be ignored.
- When the number of data items separated by delimiters in "In" is less than the number of elements in "Out", the values of the missing elements remain unchanged.
- The data in "In" separated by delimiters is of STRING type. The conversion rules for STRING-type data are as follows. The LREAL-type string format is as follows.



Name	Format
Operator	With leading consecutive ' ' (spaces) ignored, a standalone '+' or '-' is treated as a sign. The '+' sign can be omitted, and consecutive spaces following the sign are ignored.
Integer part	The part from after the sign to before the decimal point is the integer part. Consecutive spaces after the sign are not included in the integer part, and the sign may be omitted. When there is no decimal point or fractional part, the integer part is the characters up to the exponent. When there is no decimal point, fractional part, or exponent part, the integer part is whole string. The integer part consists of numbers "0" to "9" and cannot be omitted. The maximum number of digits in the integer part is the maximum text string length of 1985 minus the total number of bytes in the following: the sign, decimal point, fractional part, exponent, and blank characters before and after the sign.

Name	Format
Decimal point	The decimal point is the single dot (".") following the integer part. When there is no fractional part, the decimal point must be omitted.
Fractional part	The part from after the decimal point to before the exponent part is the fractional part. When there is no exponent part, the entire string after the decimal point is the fractional part. The fractional part consists of numbers "0" to "9". The fractional part may be omitted. The maximum number of digits in the fractional part is 15. When there is no decimal point, the fractional part does not exist.
Exponent part	The exponent part is the part from the standalone 'e' following the fractional part and the standalone '+' or '-' after it, to the end of the string. When there is no fractional part, the string after the decimal point is considered the exponent part. When there is no decimal point or fractional part, the exponent part is the string following the integer part. The numeric part of the exponent consists of numbers "0" to "9". The exponent part may be omitted. The maximum number of digits in the exponent's numeric part is 3.

◦ The following table shows examples of conversion rules.

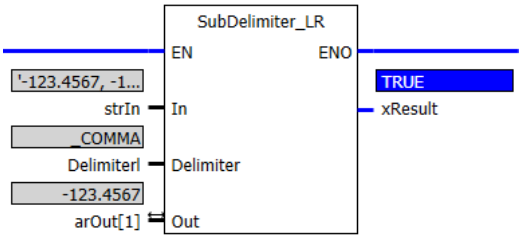
Format	STRING	LREAL
Signed, with decimal point, with fractional part, no exponent part	'-123.4567'	-123.45670318603516
Signed, with decimal point, with fractional part, with exponent part	'+123.4567e+02'	12345.669921875
Unsigned, with decimal point, with fractional part, with exponent part	'123.4567e-02'	1.2345670461654663

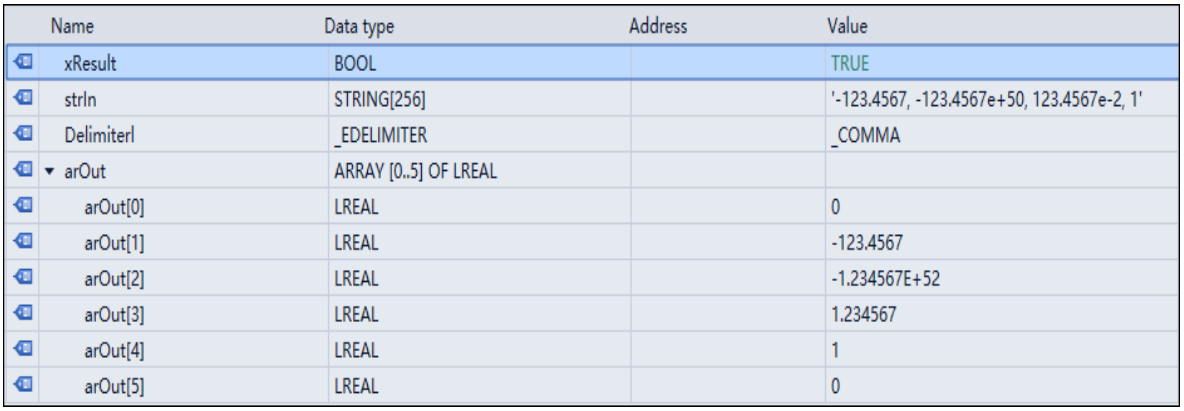
Format	STRING	LREAL
Unsigned, no decimal point, no fractional part, no exponent part	'1'	1

◦ Notes for conversion:

- When the value represented by the input string exceeds the valid range of the LREAL data type, the conversion result is positive/negative infinity.
- When the input string does not represent a long real number value, the conversion result is 0 and the function returns FALSE.
- When there are two or more consecutive '_' (underscore) characters at any intermediate position in the input string, conversion is successfully performed.
- When there is an '_' (underscore) between the '-' (minus sign) or '+' (plus sign) at the beginning of the input string, conversion is successfully performed.
- When an '_' (underscore) is at the beginning or end of the input string, a fault occurs.
- When an '_' (underscore) is at any intermediate position in the input string, it is ignored.
- When the content of the input string exceeds the precision representable by the LREAL data type, rounding is applied.
- When the content of the input string is closer to 0 than the minimum value representable by the LREAL data type, the conversion result is 0.

- **Program example**

Item	LD	ST
Define variables	VAR xResult: BOOL; strIn: STRING[256] := '-123.4567, -123.4567e+50, 123.4567e-2, 1'; Delimiterl: _eDELIMITER := _eDELIMITER._COMMA; arOut: ARRAY[0..5] OF LREAL; END_VAR	
Program		xResult TRUE := SubDelimiter_LR(In := strIn'-123.4567, -123.4567e+50, 123.4567e-2,..., Delimiter := Delimiterl _COMMA , Out := arOut[1] -123.4567);RETURN

Item	LD	ST
Operation result		

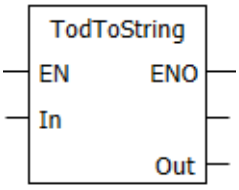
Name	Data type	Address	Value
xResult	BOOL		TRUE
strIn	STRING[256]		'-123.4567, -123.4567e+50, 123.4567e-2, 1'
DelimiterI	_EDELIMITER		_COMMA
arOut	ARRAY [0..5] OF LREAL		
arOut[0]	LREAL		0
arOut[1]	LREAL		-123.4567
arOut[2]	LREAL		-1.234567E+52
arOut[3]	LREAL		1.234567
arOut[4]	LREAL		1
arOut[5]	LREAL		0

- **Precautions**
  - When the number of conversion results for "In" is large, set a larger task cycle; otherwise, it may lead to high CPU utilization and even PLC crash.
  - When there is more than one consecutive delimiter in "In", the delimited data does not exist. If nonexistent data is output as LREAL#0 in the corresponding element of the Out[ ] array.
  - When "In" contains data that does not conform to the real number data format, no conversion is performed for the invalid data and subsequent data, the function returns FALSE, and the successfully converted values remain unchanged.
  - Do not use the delimiter in "In" for purposes other than separation. Even if used for other purposes, the instruction will still treat it as a delimiter.
  - When the conversion result exceeds the Out[ ] array area, the function returns FALSE, and the data within the range will be saved.

18.1.10.26 TodToString

Converts a time to a string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
TodToString	Time-to-string conversion	FC		:= TodToString(In := , Out =>);

- **Related variable**

  Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Time of day	Follows data type	TIME_OF_DAY#: 0:0:0	Hour, minute, second

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	String	13 bytes (12 half-width alphaumeric characters + final NULL character)	-	String

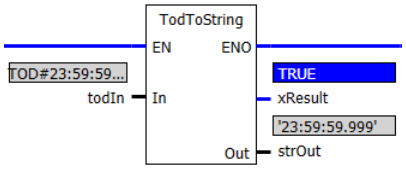
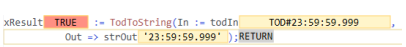
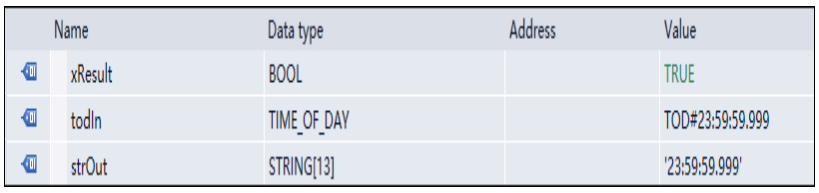
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**

  Converts the time "In" to the string "Out", with a NULL character appended to the end.

- **Program example**

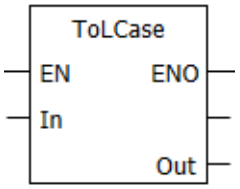
	LD	ST
Define variables	VAR xResult: BOOL; todIn: TIME_OF_DAY := TOD#23:59:59.999; strOut: STRING[13]; END_VAR	
Program		
Operation result		

- **Precautions**
  - "Out" is represented in hours, minutes, seconds, and milliseconds.
  - The return value is always TRUE.

18.1.10.27 ToLCase

Converts all half-width letters in a string to lowercase.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ToLCase	Convert to lowercase	FC		:= ToLCase(In := , Out =>);

- **Related variable**

Variable

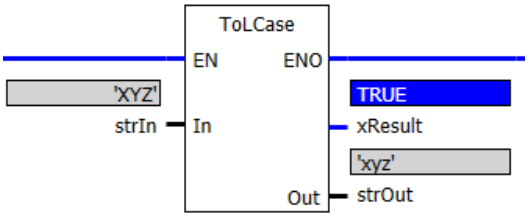
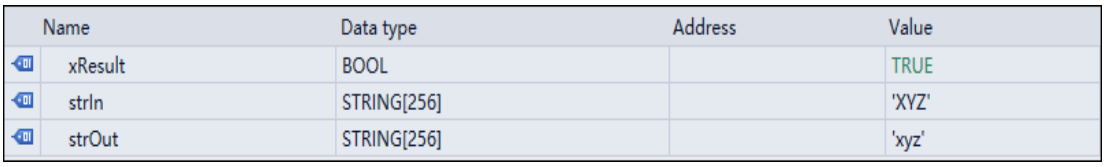
Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	Follows data type	-	The target string to be converted
Output	Out	Conversion result	Follows data type	-	Sum value

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**
  - ToLCase converts all half-width letters in the input string "In" to lowercase letters.
  - This instruction outputs a NULL character at the end of the string. Characters that are not half-width letters are not affected.
- **Program example**

The following table shows an example when "In" = 'XYZ'. The output value is 'xyz'.

Item	LD	ST
Define variables	VAR xResult: BOOL; strIn: STRING[256] := 'XYZ'; strOut: STRING[256]; END_VAR	
Program		xResult TRUE := ToLCase(In := strIn 'XYZ', Out => strOut 'xyz');RETURN
Operation result		

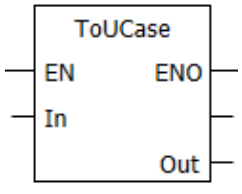
Name	Data type	Address	Value
xResult	BOOL		TRUE
strIn	STRING[256]		'XYZ'
strOut	STRING[256]		'xyz'

- **Precautions**
  - Full-width alphabetic and non-alphabetic characters are not subject to conversion.
  - An exception may occur if the character code of "In" is invalid, resulting in inaccurate "Out" values.
  - When "In" is not NULL-terminated, the result may become abnormal.
  - When the conversion result exceeds the size of "Out", only the portion up to the length of "Out" is displayed.
  - The return value is always TRUE.

18.1.10.28 ToUCase

Converts all half-width letters in a string to uppercase.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ToUCase	Convert to uppercase	FC		:= ToUCase(In := , Out =>);

- **Related variable**

Variable

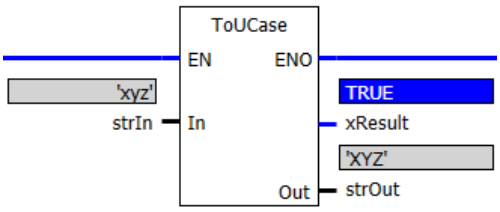
Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	Follows data type	-	The target string to be converted
Output	Out	Conversion result	Follows data type	-	Sum value

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**
  - ToLCase converts all half-width letters in the input string "In" to uppercase letters.
  - This instruction outputs a NULL character at the end of the string. Characters that are not half-width letters are not affected.
- **Program example**

  The following shows an example when "In" = 'xyz'. The output value is 'XYZ'.

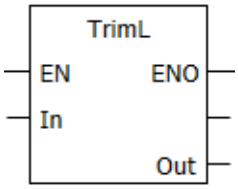
Item	LD	ST
Define variables	VAR xResult: BOOL; strIn: STRING[256] := 'xyz'; strOut: STRING[256]; END_VAR	
Program		xResult TRUE := ToUCase(In := strIn 'xyz', Out => strOut 'XYZ');RETURN
Operation result	Name / Data type / Address / Value xResult / BOOL / / TRUE strIn / STRING[256] / / 'xyz' strOut / STRING[256] / / 'XYZ'	

- **Precautions**
  - Full-width alphabetic and non-alphabetic characters are not subject to conversion.
  - An exception may occur if the character code of "In" is invalid, resulting in inaccurate "Out" values.
  - When "In" is not NULL-terminated, the result may become abnormal.
  - When the conversion result exceeds the size of "Out", only the portion up to the length of "Out" is displayed.
  - In OMUtils library versions 1.2.0.1 and earlier, the return value is always TRUE.
    In OMUtils library versions 1.3.0.0 and later, the return value is FALSE for empty strings.

18.1.10.29 TrimL

Trims leading spaces from a string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
TrimL	Trim string from the left	FC		:= TrimL(In := , Out =>);

- **Related variable**

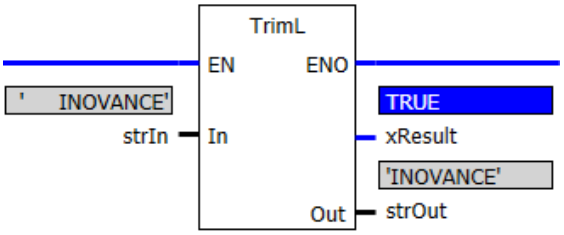
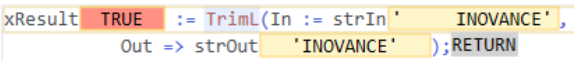
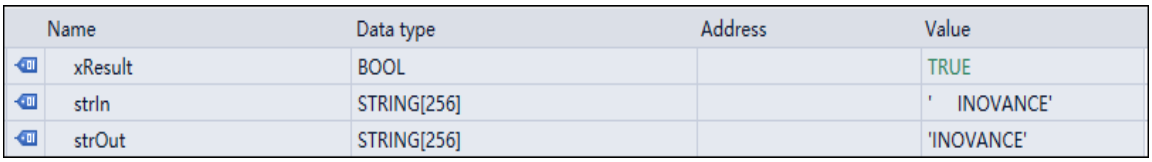
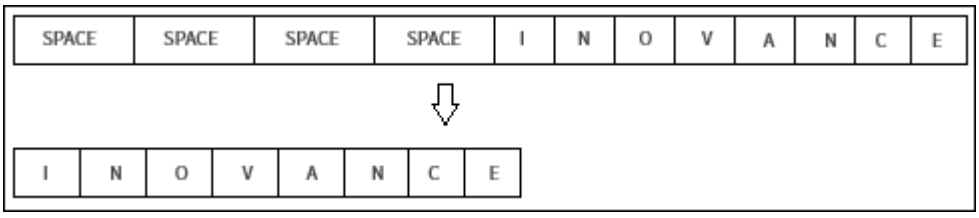
Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	String to trim	Follows data type	-	String to trim
Output	Out	Trimming result	Follows data type	-	The string after trimming

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**
  - Trims spaces from the beginning of the object string "In" up to the first occurrence of a character. When there are no spaces at the beginning, no processing is performed.
  - The instruction outputs the result including the NULL character at the end of the string.
- **Program example**

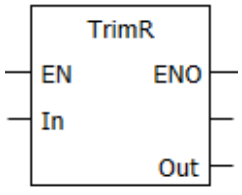
<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2"><pre>VAR
    xResult: BOOL;
    strIn: STRING[256] := '   INOVANCE';
    strOut: STRING[256];
END_VAR</pre></td></tr>
<tr><td>Program</td><td>
</td><td>
</td></tr>
<tr><td>Operation result</td><td colspan="2">
</td></tr>
<tr><td>Principle analysis</td><td colspan="2">
</td></tr>
</table>

- **Precautions**
  - Half-width and full-width spaces are both treated as spaces.
  - The instruction outputs the result including the NULL character at the end of the string.
  - The return value is always TRUE.

18.1.10.30 TrimR

Trims trailing spaces from a string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
TrimR	Trim string from the right	FC		:= TrimR(In := , Out =>);

- **Related variable**

Variable

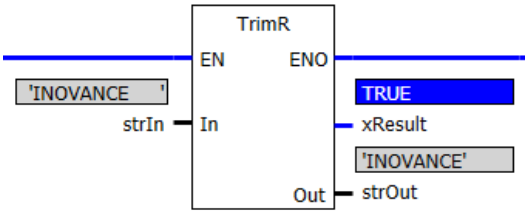
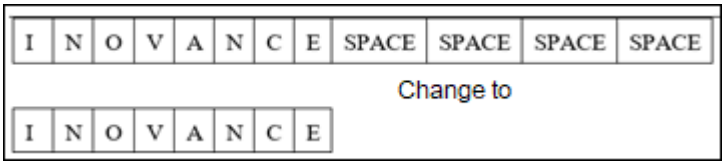
Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	String to trim	Follows data type	-	String to trim

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Trimming result	Follows data type	-	The string after trimming

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**
  - Trims spaces between the last character of the object string "In" and the end of the string. When there are no trailing spaces, no processing is performed.
  - The instruction outputs results including the NULL character at the beginning of the string.
- **Program example**

Item	LD	ST
Define variables	VAR xResult: BOOL; strIn: STRING[256] := 'INOVANCE '; strOut: STRING[256]; END_VAR	
Program		xResult TRUE := TrimR(In := strIn 'INOVANCE ', Out => strOut 'INOVANCE');RETURN
Operation result		
Principle analysis		

- **Precautions**
  - Half-width and full-width spaces are both treated as spaces.
  - The instruction outputs results including the NULL character at the beginning of the string.

◦ The return value is always TRUE.

18.1.11 Pointer

18.1.11.1 Instruction List

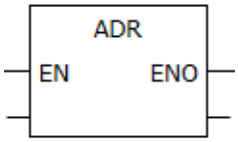
Instruction	FB/FC	Function	Library	Applicable Library Version
18.1.11.2 ADR	FC	Address retrieval instruction	-	-

18.1.11.2 ADR

ADR: Retrieves the memory address of the input variable.

^: Retrieves the content at the address of the input variable and assigns the result to the output variable.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ADR	Address retrieval instruction	FC		:= ADR( );
^	Address content retrieval instruction	FC		^

- **Related variable**

Variable

ADR

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Source data	Follows data type	-	Target data
Output	Out	Address of target data	Follows data type	-	Target data address

^

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Points to address content	Follows data type	-	Points to address content

Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Out	POINTER_TO_<TYPE>																			
Out	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

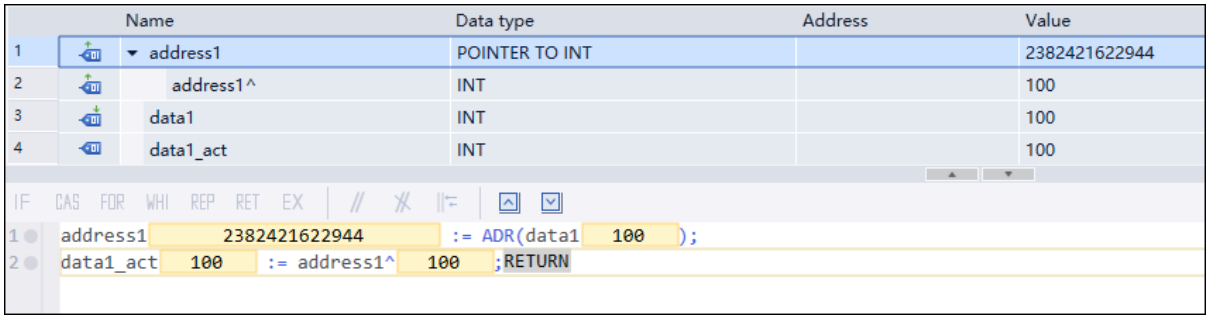
- **Program example**

  ADR/^

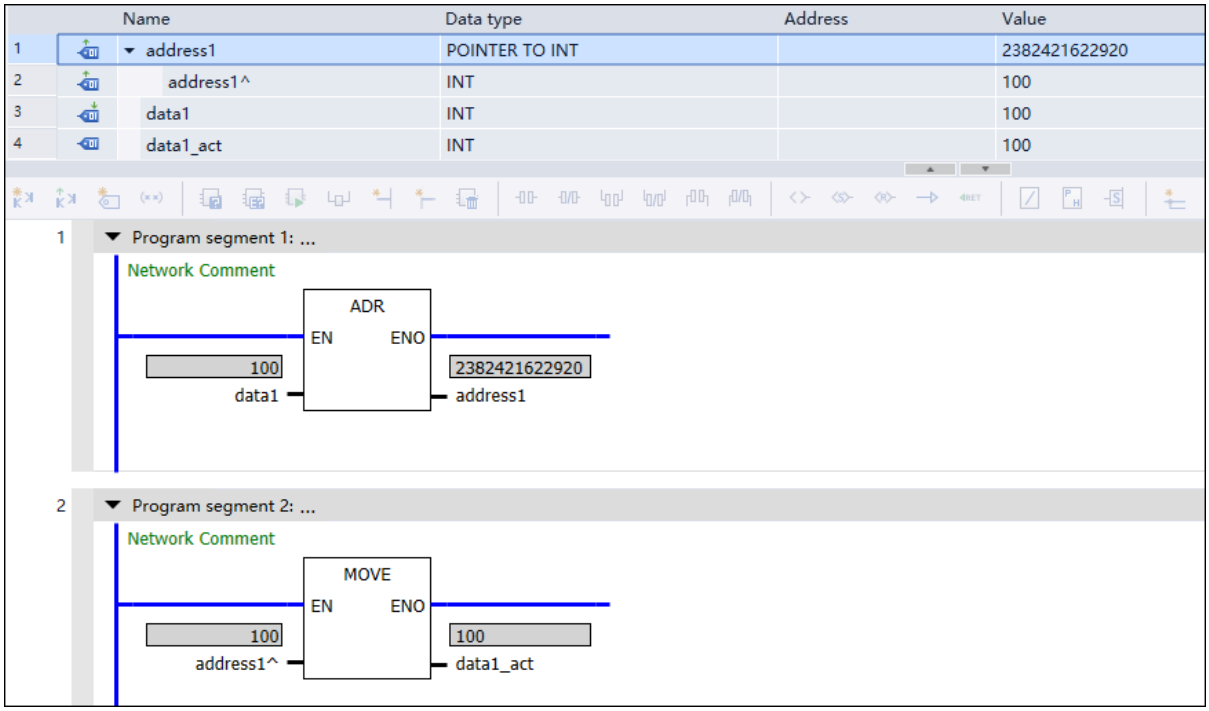
  ADR returns the address of a reference in DWORD format. This address can be used to modify the operation pointer of a function or for pointer operations within the project.

  After program execution, the pointer automatically generates a parameter address. Adding ^ after the pointer indicates the content at the address pointed to, for example, the value at the addressed location is 100.

  ST:



  LD:



- **Precautions**
  - For online modification, the address content will change, causing the POINTER_TO_<TYPE> variable to point to an unavailable memory area. To avoid this issue, ensure the pointer value is updated in each cycle.
  - Do not return the function's POINTER_TO_<TYPE> variable to the calling function or assign it to a global variable.

18.1.12 Conversion Operators

18.1.12.1 Instruction List

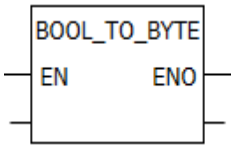
Instruction	FB/FC	Function	Library	Applicable Library Version
18.1.12.2 BOOL_TO_	FC	Boolean type conversion instruction	-	-
18.1.12.3 BYTE_TO_	FC	Byte type conversion instruction		
18.1.12.4 DATE_TO_	FC	Date type conversion instruction		
18.1.12.5 DINT_TO_	FC	Long integer conversion instruction		
18.1.12.6 DT_TO_	FC	Datetime type conversion instruction		
18.1.12.7 DWORD_TO_	FC	Double-word type conversion instruction		
18.1.12.8 INT_TO_	FC	Integer type conversion instruction		

Instruction	FB/FC	Function	Library	Applicable Library Version
18.1.12.9 LINT_TO_	FC	Long integer conversion instruction		
18.1.12.10 LREAL_TO_	FC	LREAL type conversion instruction		
18.1.12.11 LTIME_TO_	FC	Time type conversion instruction		
18.1.12.12 LWORD_TO_	FC	Byte type conversion instruction		
18.1.12.13 REAL_TO_	FC	Real type conversion instruction		
18.1.12.14 SINT_TO_	FC	Short integer conversion instruction		
18.1.12.15 STRING_TO_	FC	Character type conversion instruction		
18.1.12.16 TIME_TO_	FC	Time type conversion instruction		
18.1.12.17 TOD_TO_	FC	Time-of-day conversion instruction		
18.1.12.18 UDINT_TO_	FC	Unsigned long integer conversion instruction		
18.1.12.19 UINT_TO_	FC	Unsigned integer type conversion instruction		
18.1.12.20 ULINT_TO_	FC	Unsigned long integer type conversion instruction		
18.1.12.21 USINT_TO_	FC	Unsigned short integer type conversion instruction		
18.1.12.22 WORD_TO_	FC	Byte type conversion instruction		
18.1.12.23 WSTRING_TO_	FC	Character type conversion instruction		
18.1.12.24 UNICODE_TO_UTF8	FC	UNICODE-to-UTF8 conversion	OMUtils	Library version 1.2.0.0 and later

18.1.12.2 BOOL_TO_<TYPE>

Converts a Boolean variable to another type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BOOL_TO_<TYPE>	Boolean type conversion instruction	FC		BOOL_TO_BYTE

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Data Source	Follows data type	-	Data source
Output	OUT	Result data	Follows data type	-	Converted data

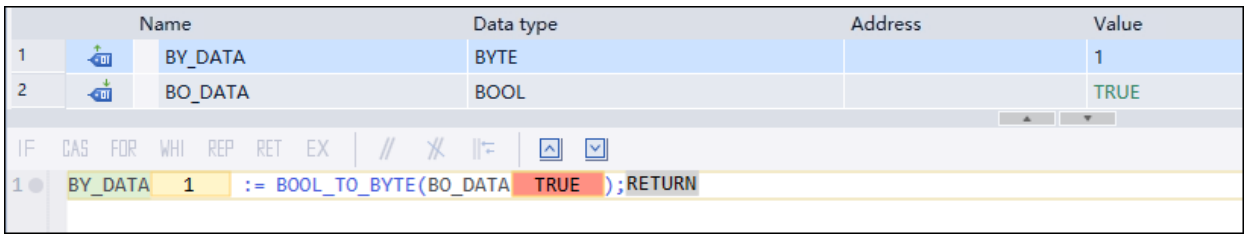
  Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

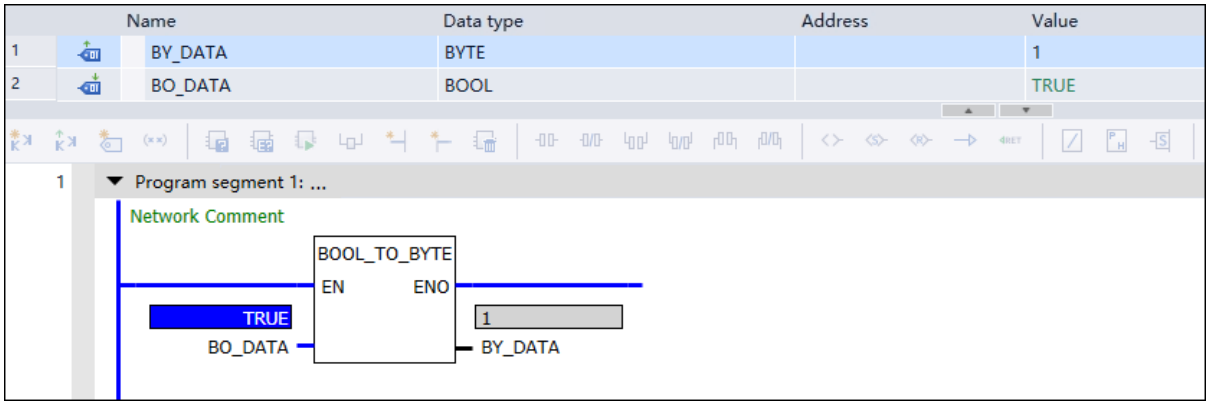
- **Program example**

  Take BOOL_TO_BYTE as an example.

  ST:



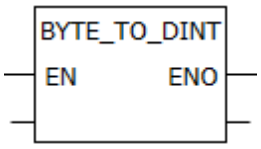
  LD:



18.1.12.3 BYTE_TO_<TYPE>

Converts a byte-type variable to another type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BYTE_TO_<TYPE>	Byte type conversion instruction	FC		BYTE_TO_DINT

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Data Source	Follows data type	-	Data source
Output	OUT	Result data	Follows data type	-	Converted data

Data Type

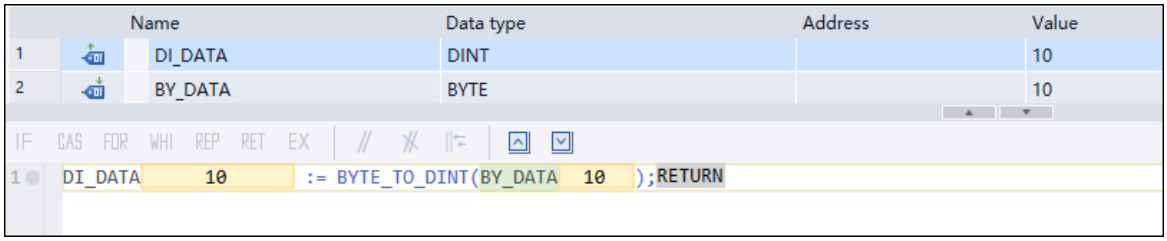
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
OUT	√	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

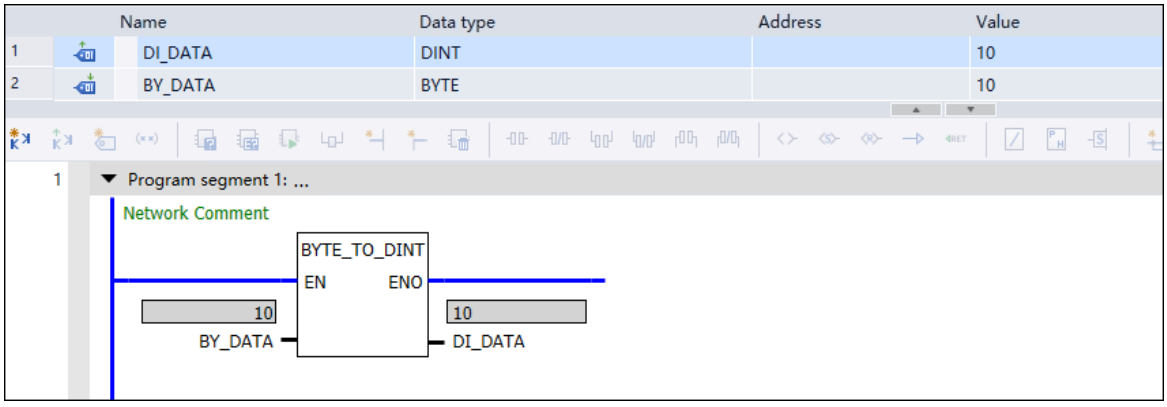
- **Program example**

  Take BYTE_TO_DINT as an example.

  ST:



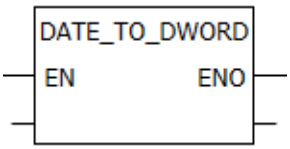
  LD:



18.1.12.4 DATE_TO_<TYPE>

Converts a date type variable to another type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DATE_TO_<TYPE>	Date type conversion instruction	FC		DATE_TO_DWORD

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Data Source	Follows data type	-	Data source
Output	OUT	Result data	Follows data type	-	Converted data

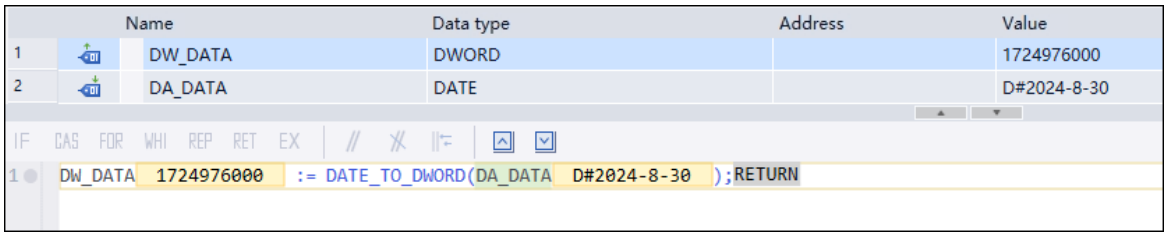
  Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	√	√	√	√

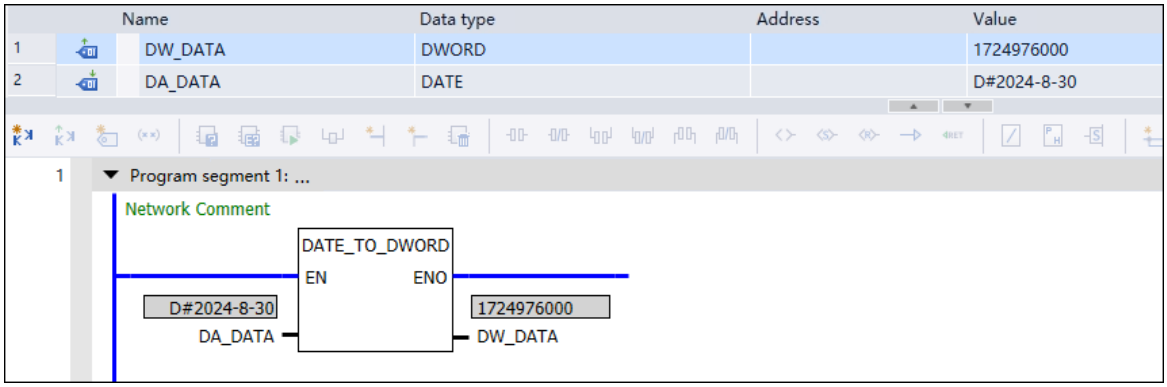
- **Program example**

  Take DATE_TO_DWORD as an example.

  ST:



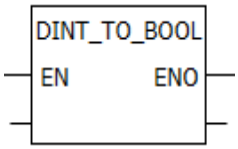
  LD:



18.1.12.5 DINT_TO_<TYPE>

Converts a long integer type variable to another type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DINT_TO_<TYPE>	Long integer conversion instruction	FC		DINT_TO_BOOL

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Data Source	Follows data type	-	Data source
Output	OUT	Result data	Follows data type	-	Converted data

Data Type

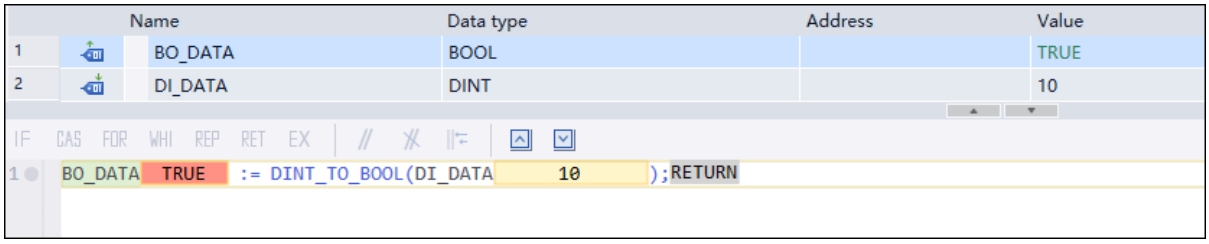
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String							
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
OUT	√	√	√	√	√	√	√	√	√	√	√	-	√	√	√	√	√	√	√	√	√	√

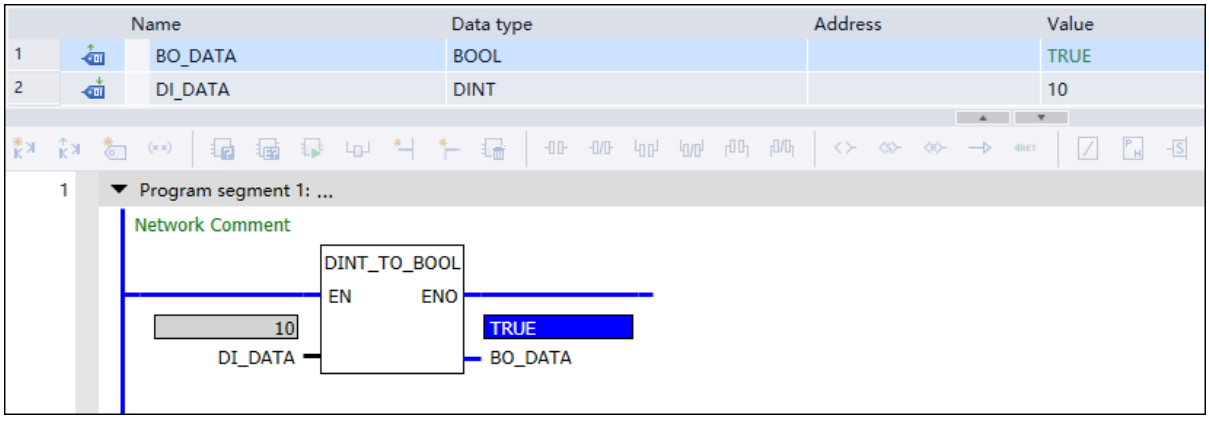
- **Program example**

  Take DINT_TO_BOOL as an example.

  ST:



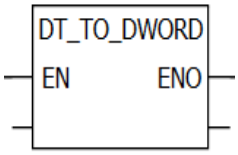
  LD:



18.1.12.6 DT_TO_<TYPE>

Converts a datetime-type variable into another type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DT_TO_<TYPE>	Datetime type conversion instruction	FC		DT_TO_DWORD

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Data Source	Follows data type	-	Data source
Output	OUT	Result data	Follows data type	-	Converted data

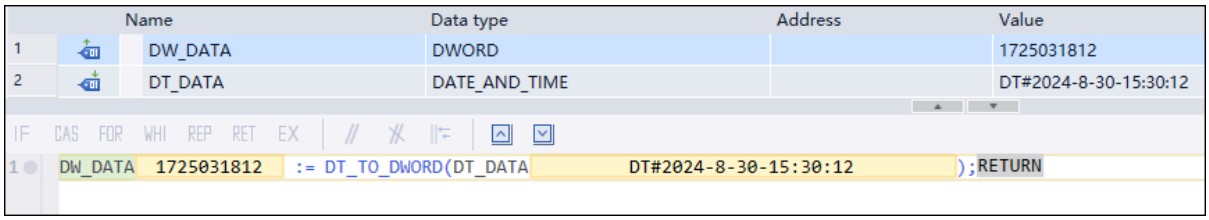
  Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	√	√

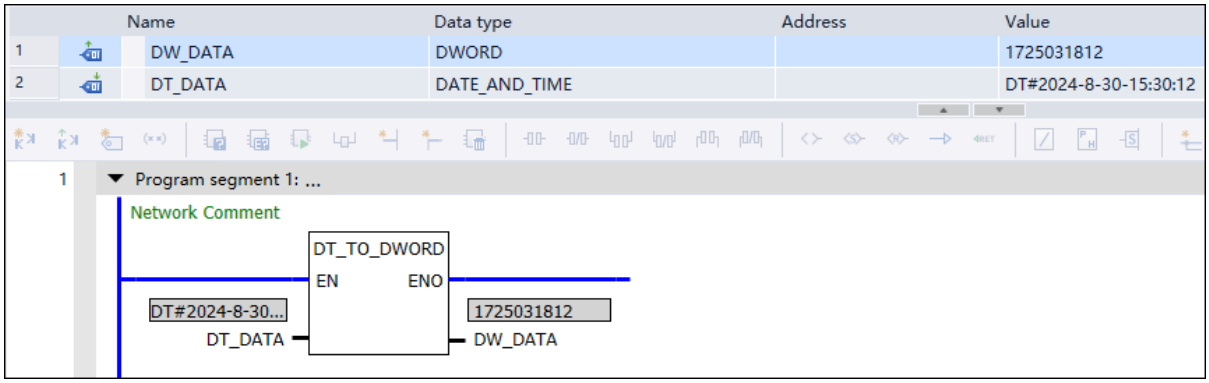
- **Program example**

  Take DT_TO_DWORD as an example. When the input is DT#2024-8-30-15:30:12, the output result is 1725031812, representing the number of seconds elapsed from 1970-1-1 00:00:00 to 2024-8-30 15:30:12.

  ST:



  LD:

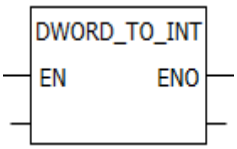


- **Precautions**
  - Take DT_TO_BOOL as an example, when the input is DT#1970-1-1-0:0:0, the output is FALSE; otherwise, the output is TRUE.
  - When the output result exceeds the value range of the target data type, the result will be taken modulo the maximum value representable by that data type.

18.1.12.7 DWORD_TO_<TYPE>

Converts a double-word type variable to another type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DWORD_TO_<TYPE>	Double-word type conversion instruction	FC		DWORD_TO_INT

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Data Source	Follows data type	-	Data source
Output	OUT	Result data	Follows data type	-	Converted data

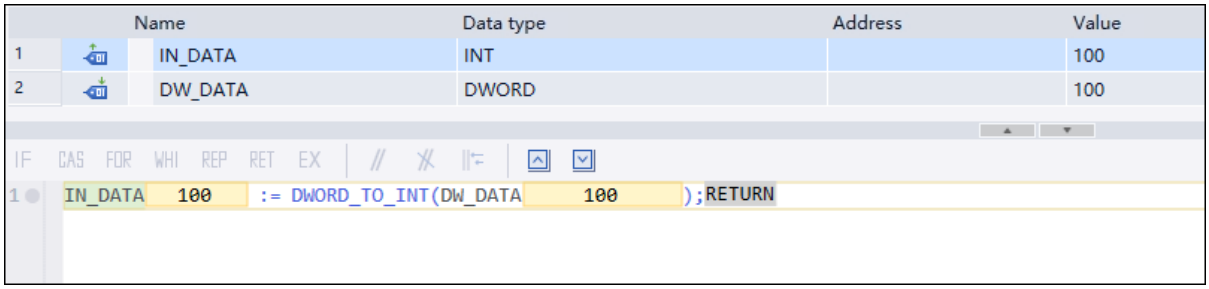
Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	√	√	√	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

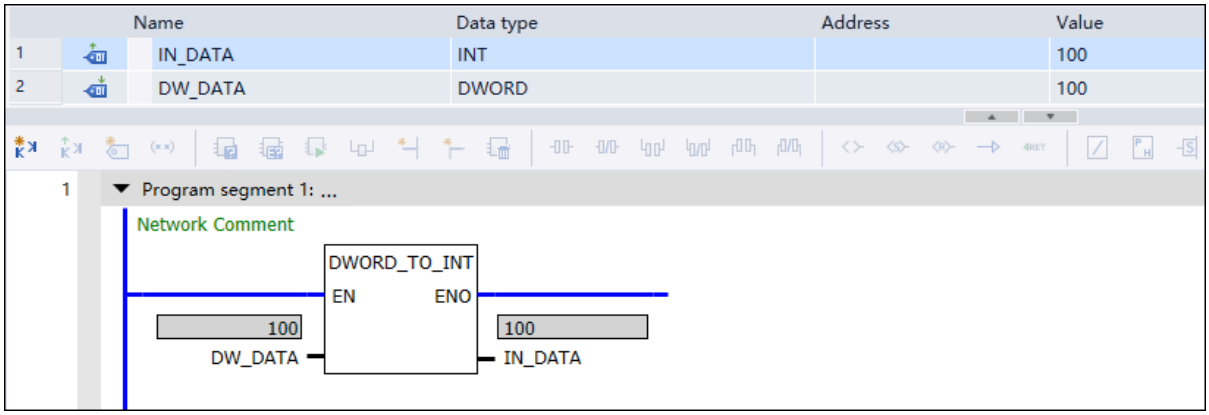
- **Program example**

  Take DWORD_TO_INT as an example.

  ST:



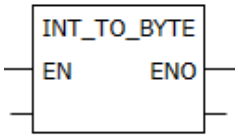
  LD:



18.1.12.8 INT_TO_<TYPE>

Converts an integer type variable to another type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
INT_TO_<TYPE>	Integer type conversion instruction	FC		INT_TO_BYTE

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Data Source	Follows data type	-	Data source
Output	OUT	Result data	Follows data type	-	Converted data

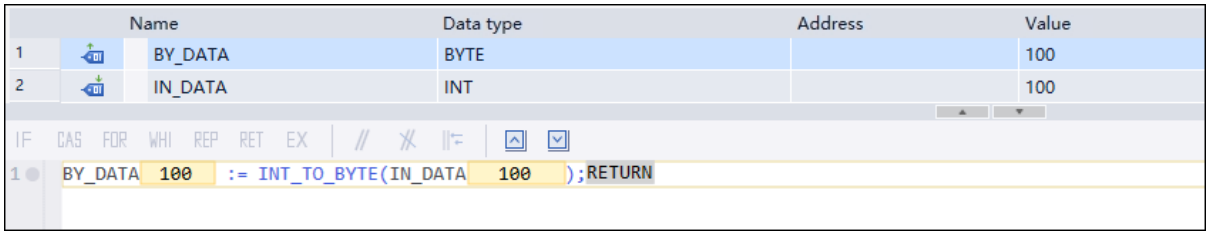
  Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-
OUT	√	√	√	√	√	√	√	√	√	√	-	√	√	√	√	√	√	√	√	√	√	√

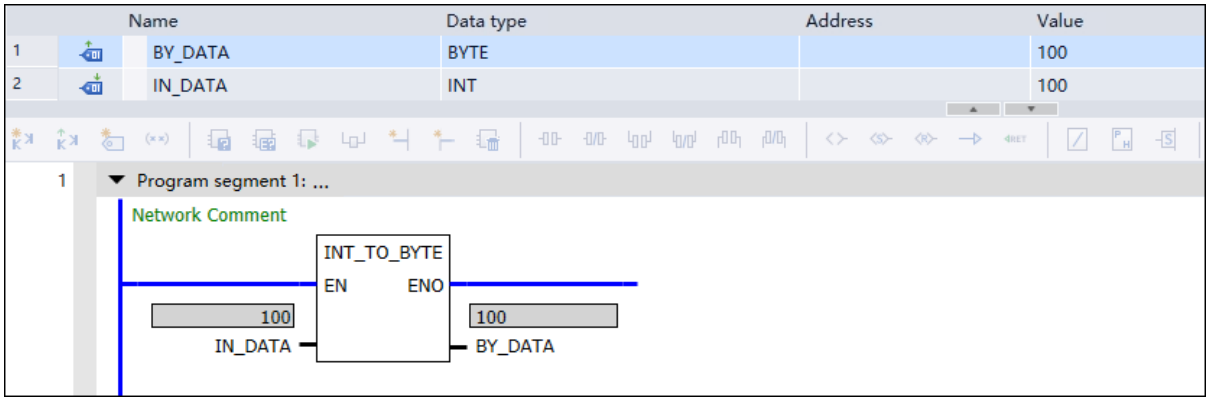
- **Program example**

  Take INT_TO_BYTE as an example.

  ST:



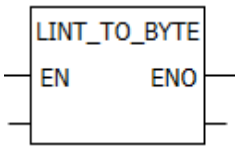
  LD:



18.1.12.9 LINT_TO_<TYPE>

Converts a long integer type variable to another type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LINT_TO_<TYPE>	Long integer conversion instruction	FC		LINT_TO_BYTE

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Data Source	Follows data type	-	Data source
Output	OUT	Result data	Follows data type	-	Converted data

Data Type

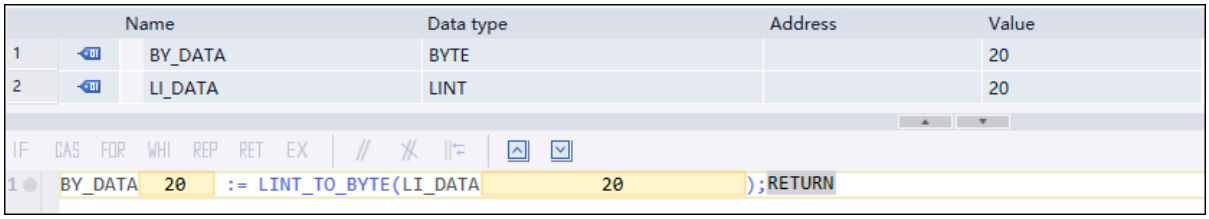
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
OUT	√	√	√	√	√	√	√	√	√	√	√	√	-	√	√	√	√	√	√	√	√	√

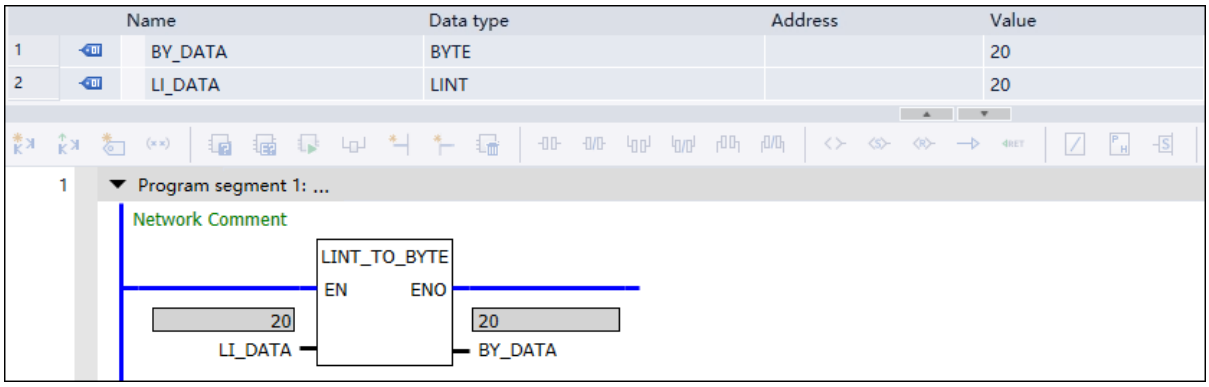
- **Program example**

  Take LINT_TO_BYTE as an example.

  ST:



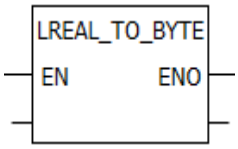
  LD:



18.1.12.10 LREAL_TO_<TYPE>

Converts a LREAL type variable to another type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LREAL_TO_<TYPE>	LREAL type conversion instruction	FC		LREAL_TO_BYTE

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Data Source	Follows data type	-	Data source
Output	OUT	Result data	Follows data type	-	Converted data

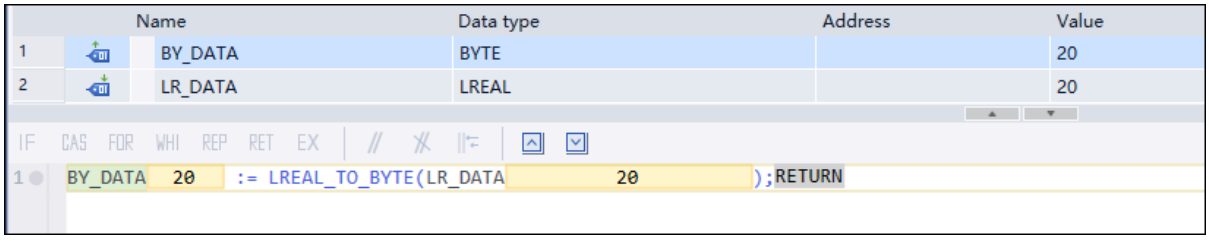
Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	√	√	√	√	√	√	√

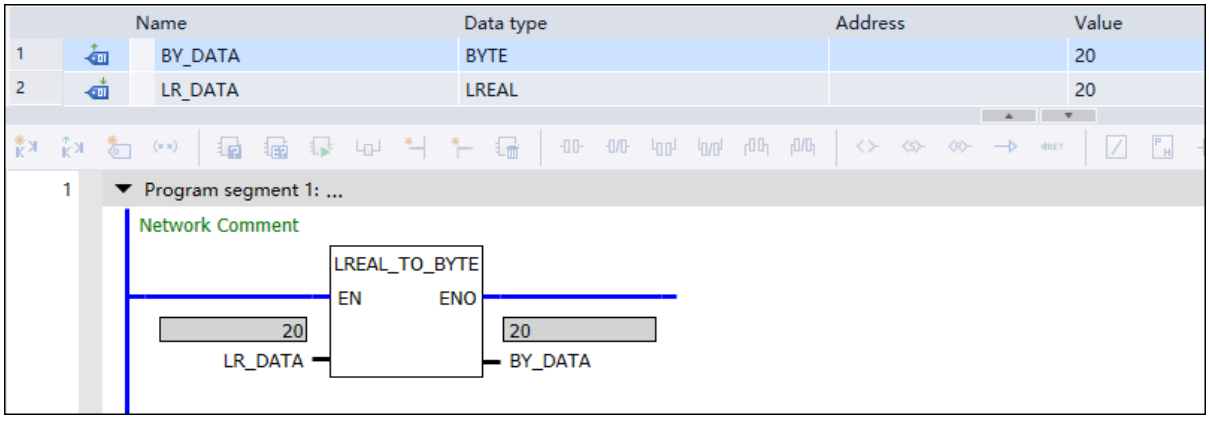
- **Program example**

  Take LREAL_TO_BYTE as an example.

  ST:



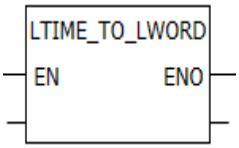
  LD:



18.1.12.11 LTIME_TO_<TYPE>

Convert a clock type variable to another type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LTIME_TO_<TYPE>	Time type conversion instruction	FC		LTIME_TO_LWORD

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Data Source	Follows data type	-	Data source
Output	OUT	Result data	Follows data type	-	Converted data

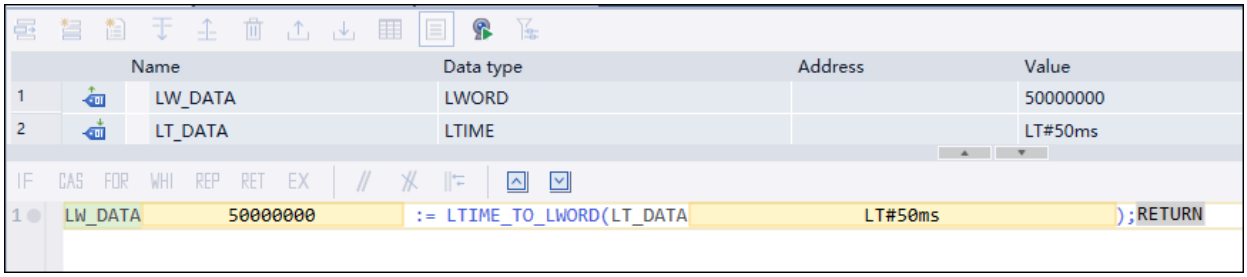
Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	√	√	√	√	√

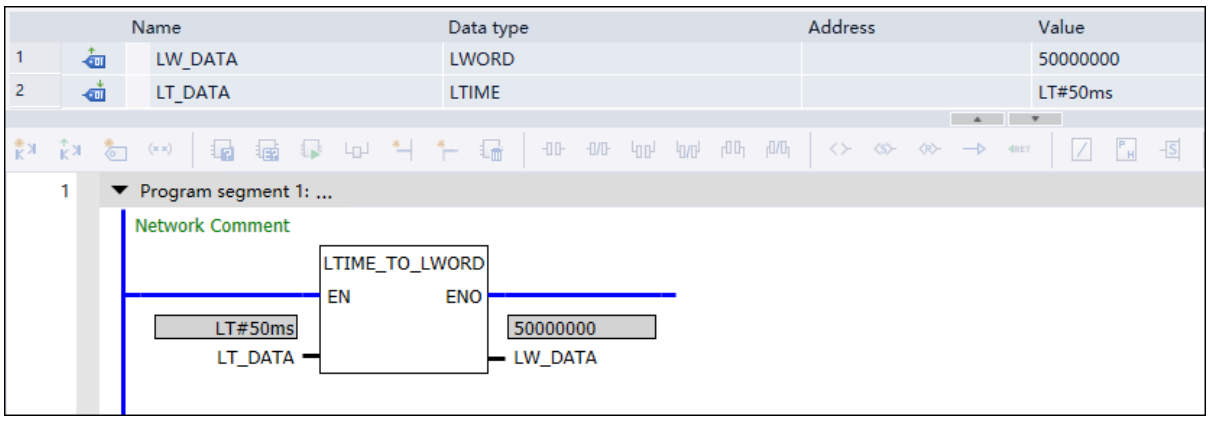
- **Program example**

Take LTIME_TO_LWORD as an example.

ST:



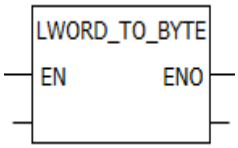
LD:



18.1.12.12 LWORD_TO_<TYPE>

Converts a byte-type variable to another type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LWORD_TO_<TYPE>	Byte type conversion instruction	FC		LWORD_TO_BYTE

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Data Source	Follows data type	-	Data source
Output	OUT	Result data	Follows data type	-	Converted data

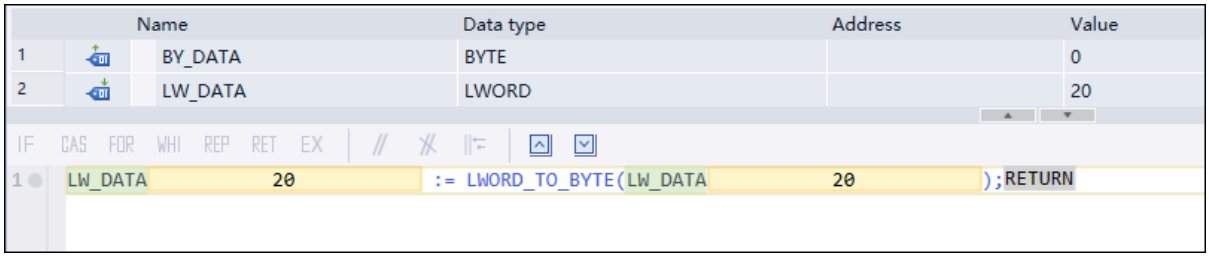
Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	√	√	√	√	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

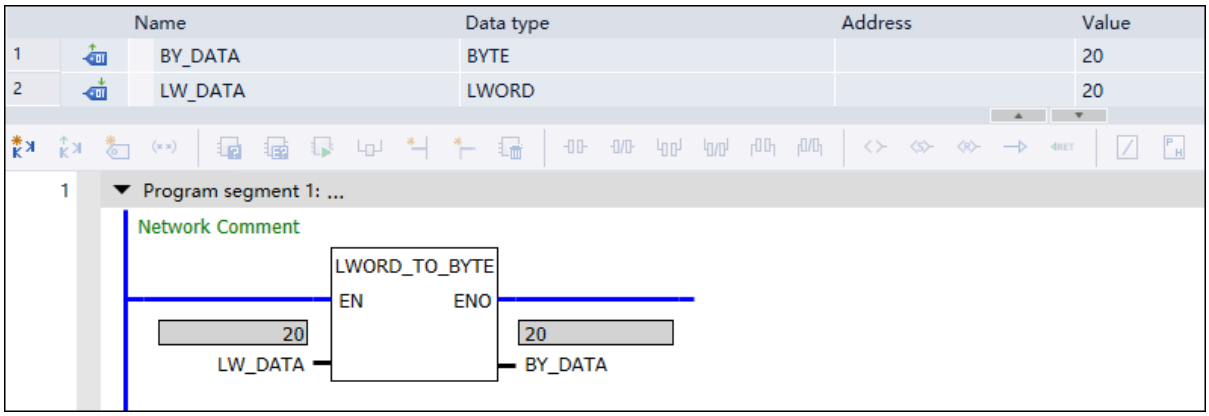
- **Program example**

  Take LWORD_TO_BYTE as an example.

  ST:



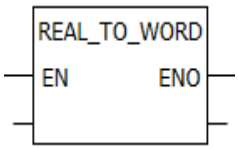
  LD:



18.1.12.13 REAL_TO_<TYPE>

Converts a real type variable to another type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
REAL_TO_<TYPE>	Real type conversion instruction	FC		REAL_TO_WORD

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Data Source	Follows data type	-	Data source
Output	OUT	Result data	Follows data type	-	Converted data

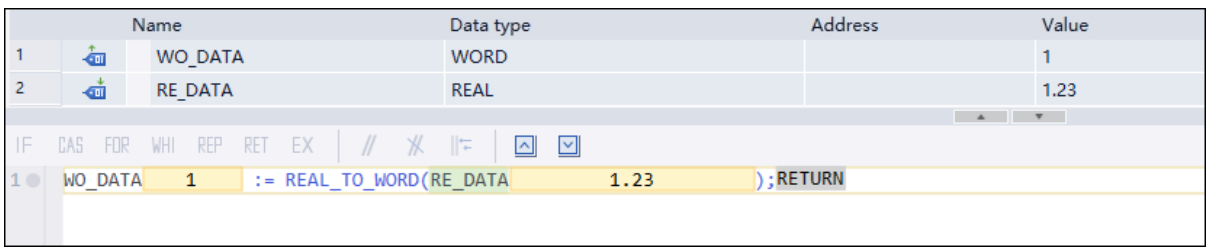
  Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	-	√	√	√	√	√	√	√	√

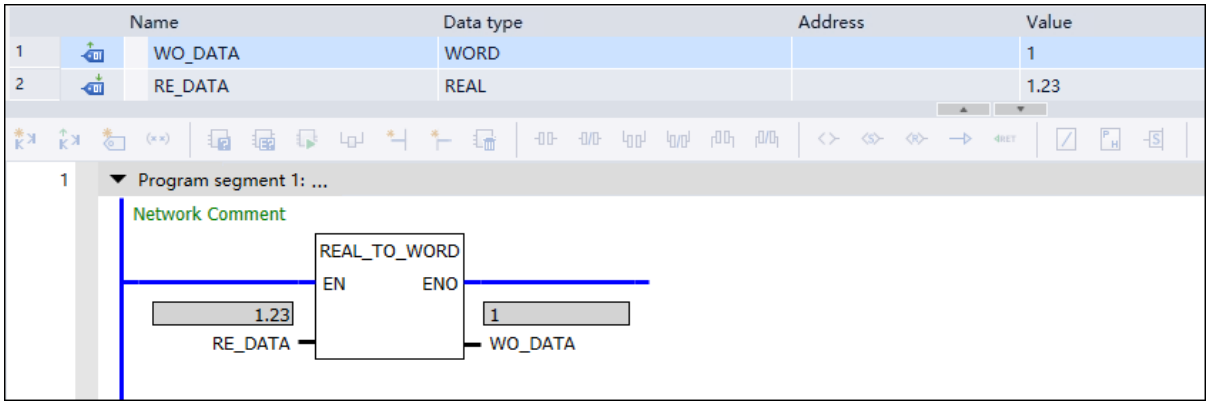
- **Program example**

  Take REAL_TO_WORD as an example.

  ST:



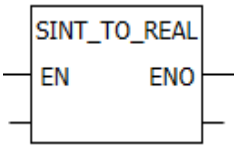
  LD:



18.1.12.14 SINT_TO_<TYPE>

Converts a short integer type variable to another type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SINT_TO_<TYPE>	Short integer conversion instruction	FC		SINT_TO_REAL

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Data Source	Follows data type	-	Data source
Output	OUT	Result data	Follows data type	-	Converted data

Data Type

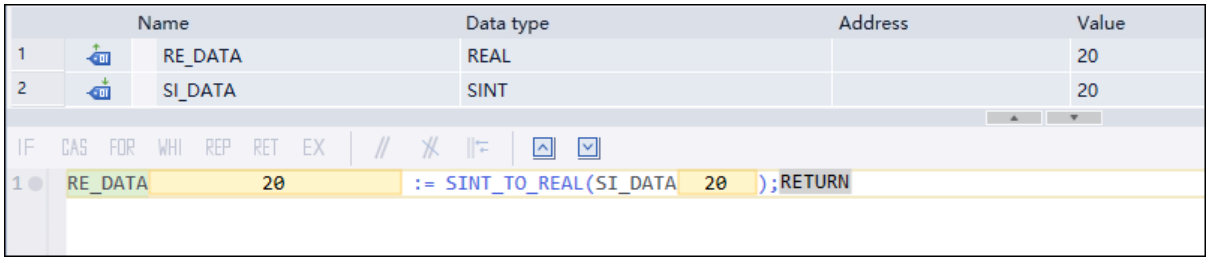
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
OUT	√	√	√	√	√	√	√	√	√	-	√	√	√	√	√	√	√	√	√	√	√	√

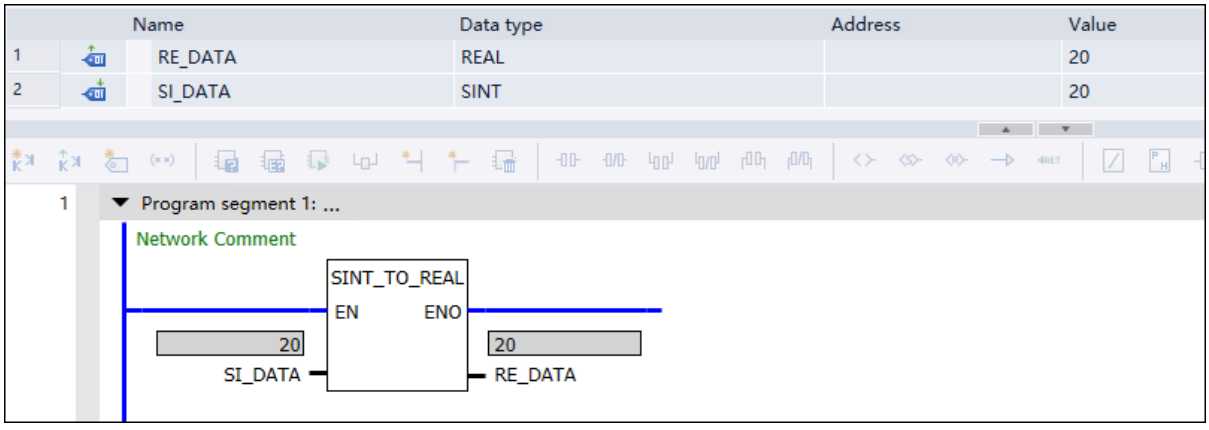
- **Program example**

  Take SINT_TO_REAL as an example.

  ST:



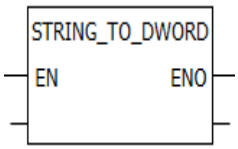
  LD:



18.1.12.15 STRING_TO_<TYPE>

Converts a character type variable to another type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
STRING_TO_<TYPE>	Character type conversion instruction	FC		STRING_TO_DWORD

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Data Source	Follows data type	-	Data source
Output	OUT	Result data	Follows data type	-	Converted data

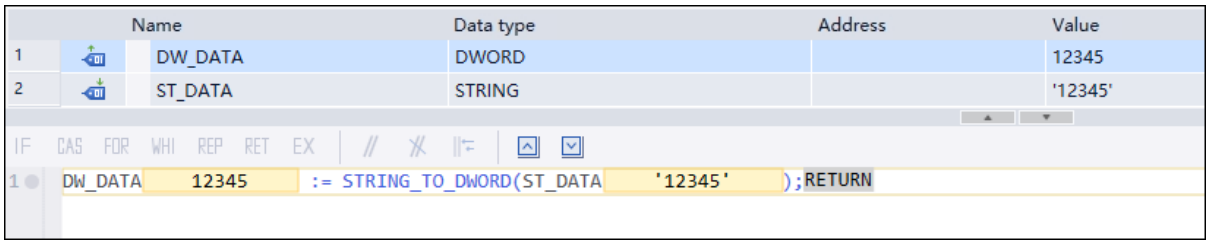
  Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	√

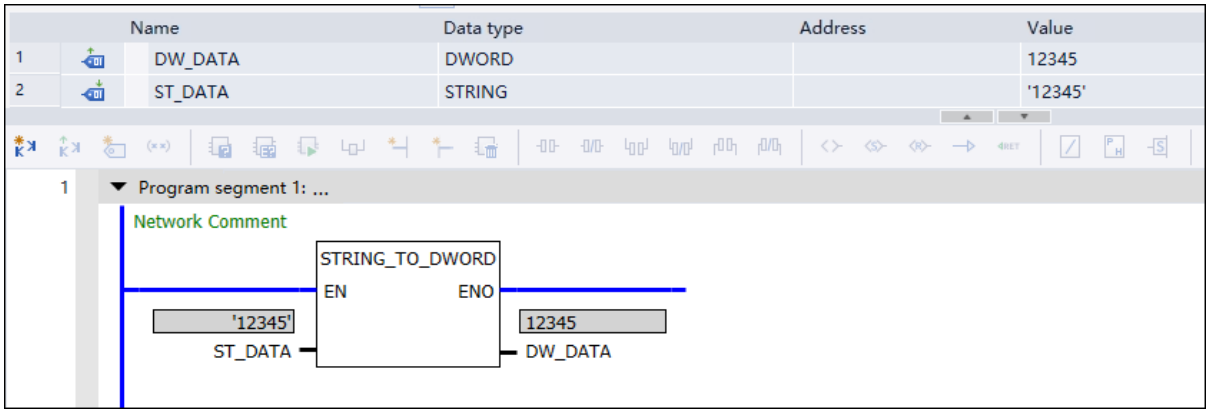
- **Program example**

  Take STRING_TO_DWORD as an example.

  ST:



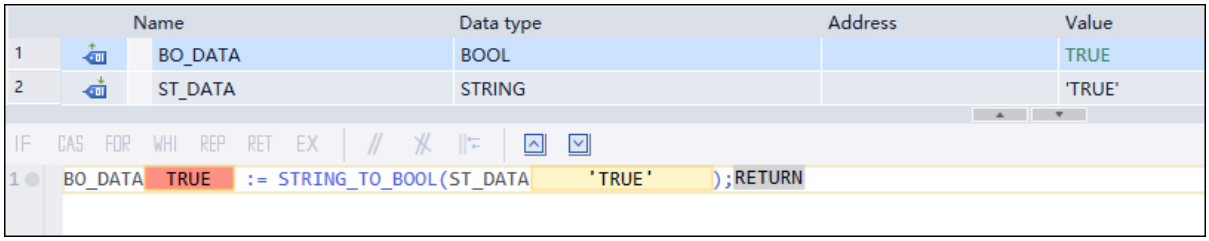
  LD:



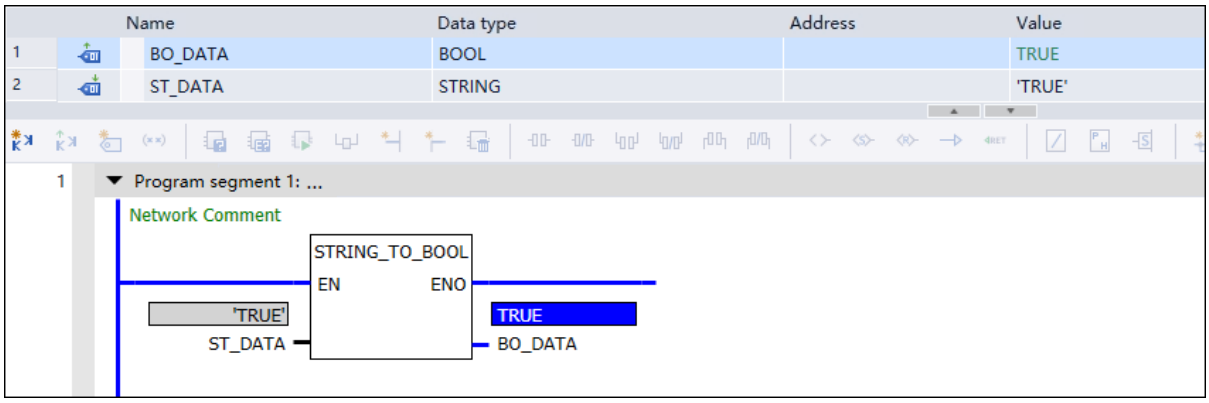
- **Special example**

  Take STRING_TO_BOOL as an example.

  ST:



  LD:



  The corresponding BOOL type variable will be set to TRUE only when the input STRING type data is "TRUE".

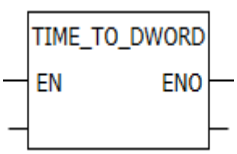
- **Note**

  : Take STRING_TO_WSTRING as an example, Chinese characters are not supported for conversion.

18.1.12.16 TIME_TO_<TYPE>

Converts a time type variable to another type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
TIME_TO_<TYPE>	Time type conversion instruction	FC		TIME_TO_DWORD

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Data Source	Follows data type	-	Data source
Output	OUT	Result data	Follows data type	-	Converted data

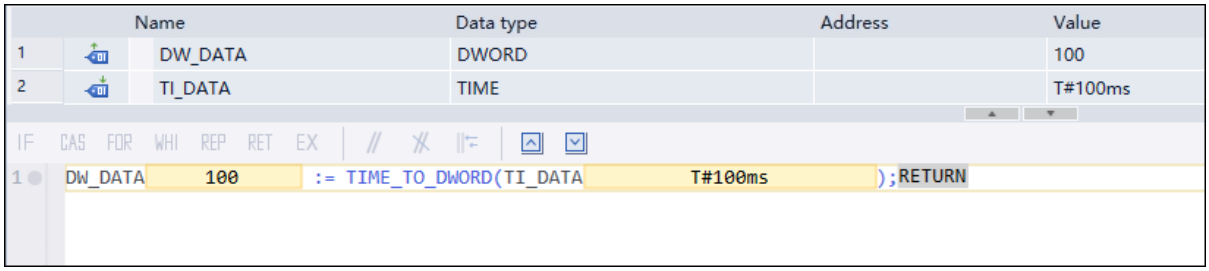
Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	√	√	√	√	√	√

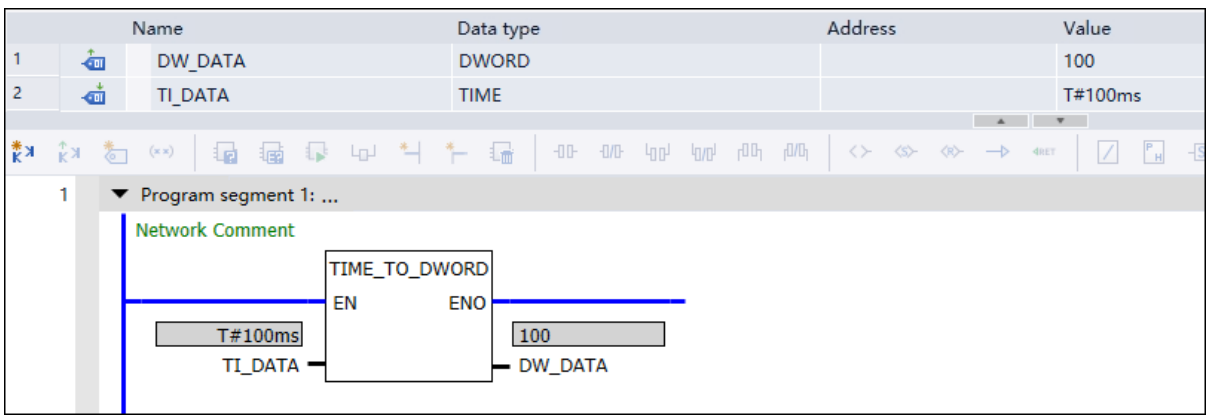
- **Program example**

  Take TIME_TO_DWORD as an example.

  ST:



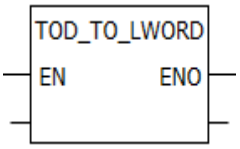
  LD:



18.1.12.17 TOD_TO_<TYPE>

Converts the time-of-day variable to another type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
TOD_TO_<TYPE>	Time-of-day conversion instruction	FC		TOD_TO_LWORD

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Data Source	Follows data type	-	Data source
Output	OUT	Result data	Follows data type	-	Converted data

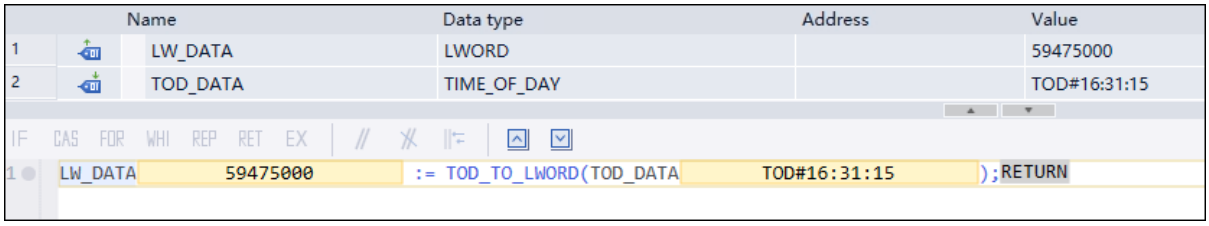
Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	√	√	√

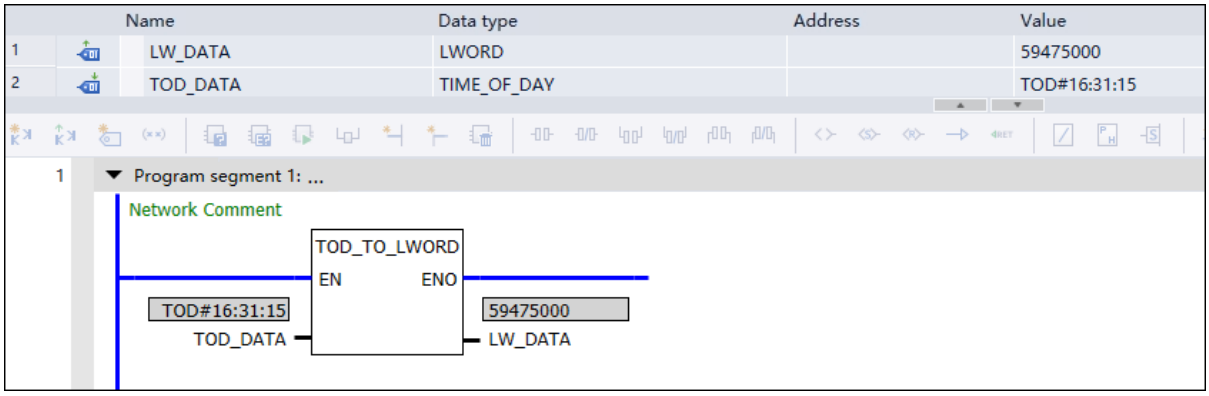
- **Program example**

Take TOD_TO_LWORD as an example.

ST:



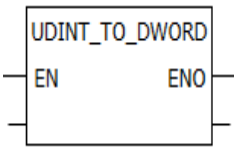
LD:



18.1.12.18 UDINT_TO_<TYPE>

Converts an unsigned long integer type variable to another type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
UDINT_TO_<TYPE>	Unsigned long integer conversion instruction	FC		UDINT_TO_DWORD

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Data Source	Follows data type	-	Data source
Output	OUT	Result data	Follows data type	-	Converted data

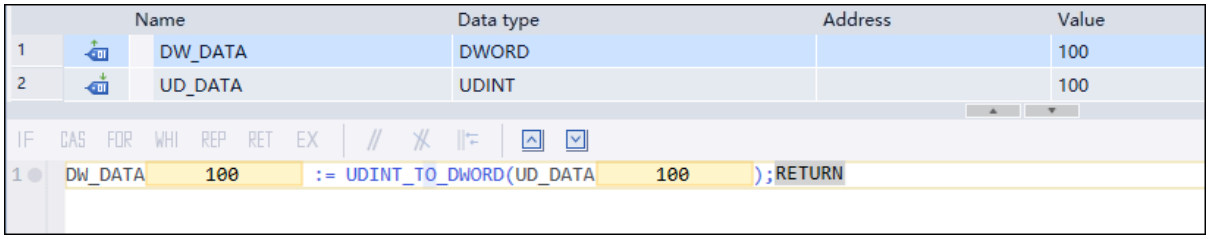
Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	√	√	√	√	√	√	√	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√

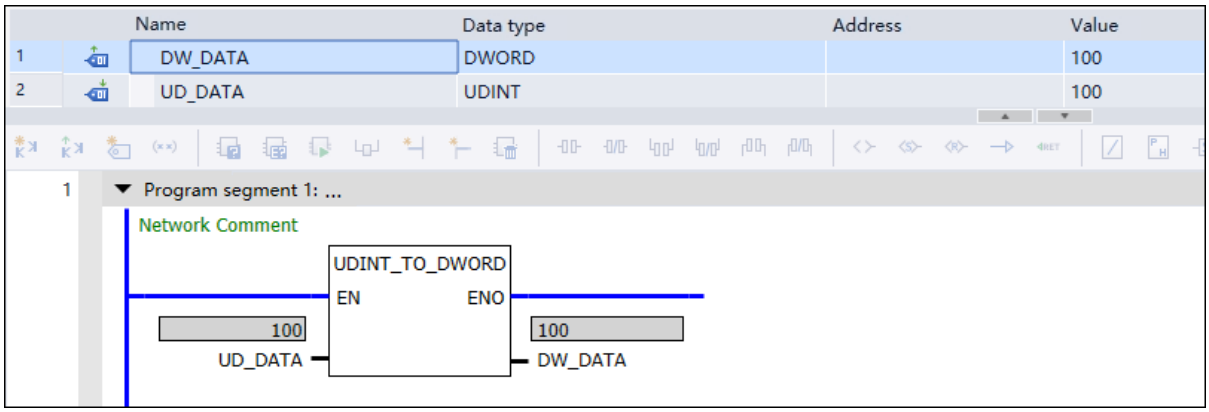
- **Program example**

  Take UDINT_TO_DWORD as an example.

  ST:



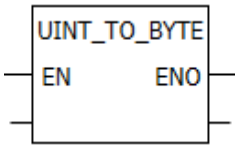
  LD:



18.1.12.19 UINT_TO_<TYPE>

Converts an unsigned integer type variable to another type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
UINT_TO_<TYPE>	Unsigned integer type conversion instruction	FC		UINT_TO_BYTE

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Data Source	Follows data type	-	Data source
Output	OUT	Result data	Follows data type	-	Converted data

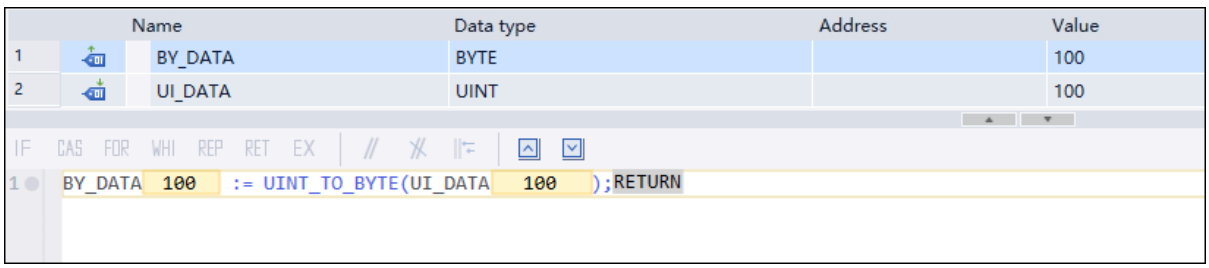
  Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	√	√	√	√	√	√	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

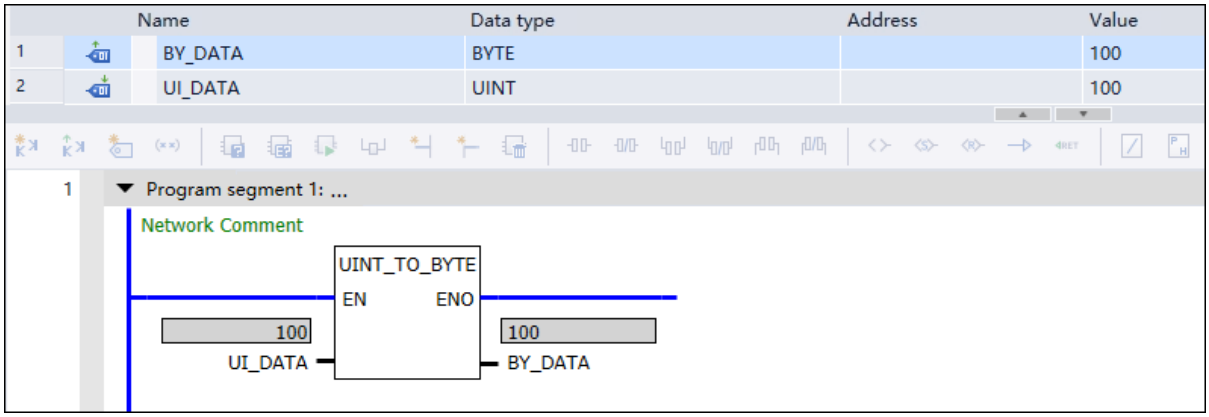
- **Program example**

  Take UINT_TO_BYTE as an example.

  ST:



  LD:



18.1.12.20 ULINT_TO_<TYPE>

Converts an unsigned long integer type variable to another type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ULINT_TO_<TYPE>	Unsigned long integer type conversion instruction	FC	ULINT_TO_BYTE EN ENO	ULINT_TO_BYTE

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Data Source	Follows data type	-	Data source
Output	OUT	Result data	Follows data type	-	Converted data

Data Type

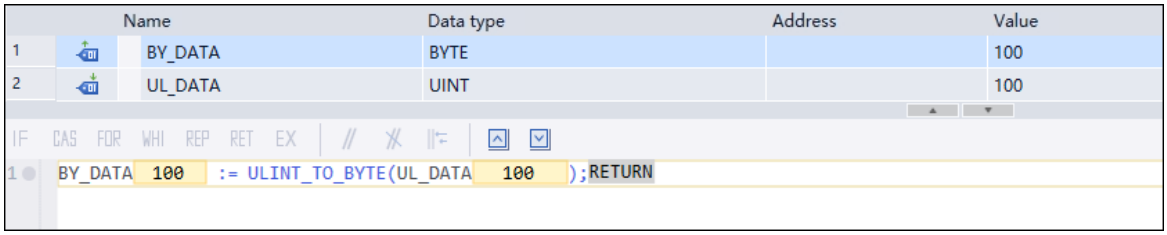
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
OUT	√	√	√	√	√	√	√	√	-	√	√	√	√	√	√	√	√	√	√	√	√	√

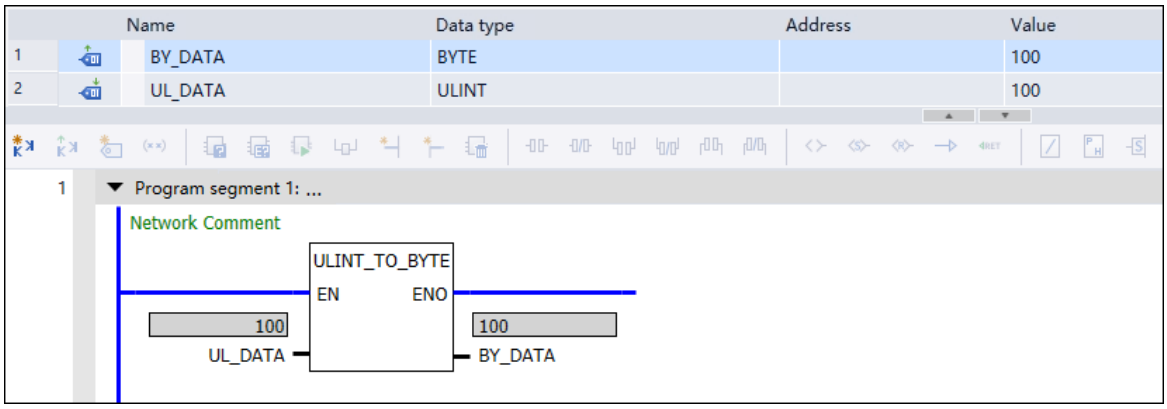
- **Program example**

  Take ULINT_TO_BYTE as an example.

  ST:



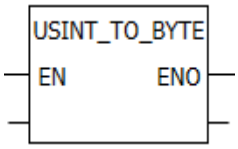
  LD:



18.1.12.21 USINT_TO_<TYPE>

Converts an unsigned short integer type variable to another type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
USINT_TO_<TYPE>	Unsigned short integer type conversion instruction	FC		USINT_TO_BYTE

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Data Source	Follows data type	-	Data source
Output	OUT	Result data	Follows data type	-	Converted data

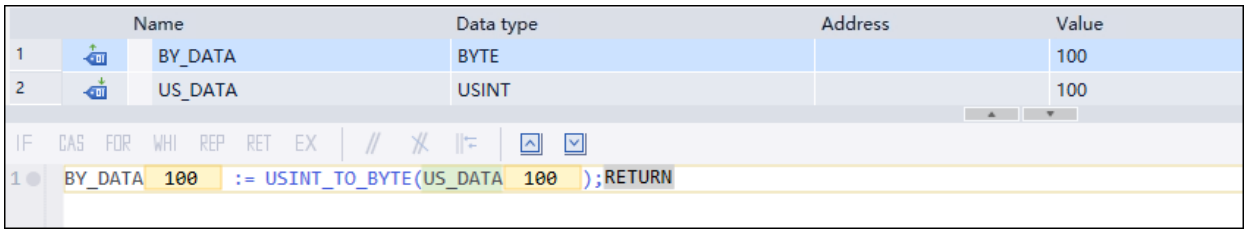
  Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
OUT	√	√	√	√	√	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

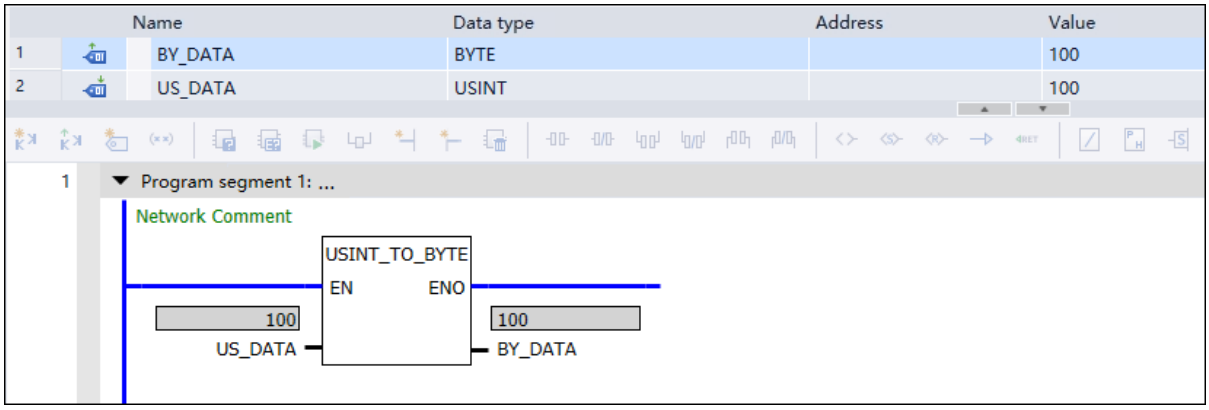
- **Program example**

  Take USINT_TO_BYTE as an example.

  ST:



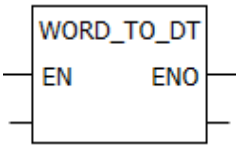
  LD:



18.1.12.22 WORD_TO_<TYPE>

Converts a byte-type variable to another type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
WORD_TO_<TYPE>	Byte type conversion instruction	FC		WORD_TO_DT

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Data Source	Follows data type	-	Data source
Output	OUT	Result data	Follows data type	-	Converted data

Data Type

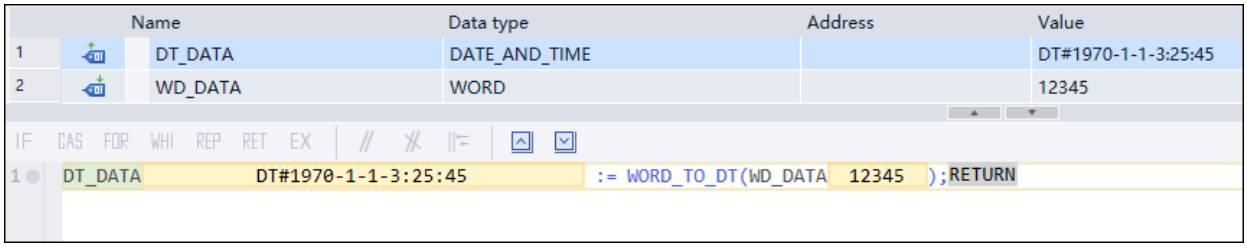
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
OUT	√	√	-	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√

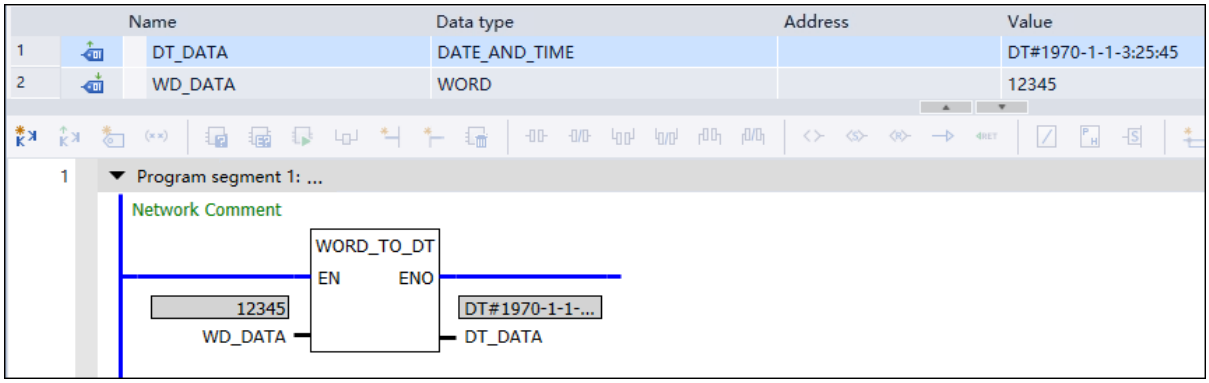
- **Program example**

  Take WORD_TO_DT as an example.

  ST:



  LD:



18.1.12.23 WSTRING_TO_<TYPE>

Converts a variable-length double-byte character type variable to another type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
WSTRING_TO_<TYPE>	Character type conversion instruction	FC	WSTRING_TO_DWORD EN ENO	WSTRING_TO_DWORD

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Data Source	Follows data type	-	Data source
Output	OUT	Result data	Follows data type	-	Converted data

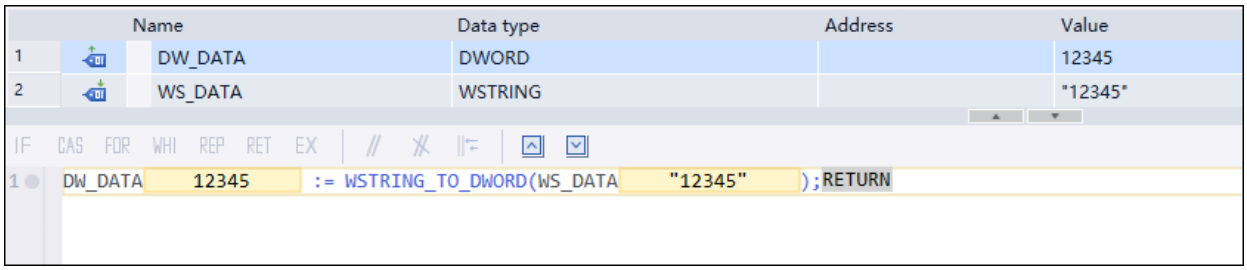
Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String						
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	LTIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
OUT	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-

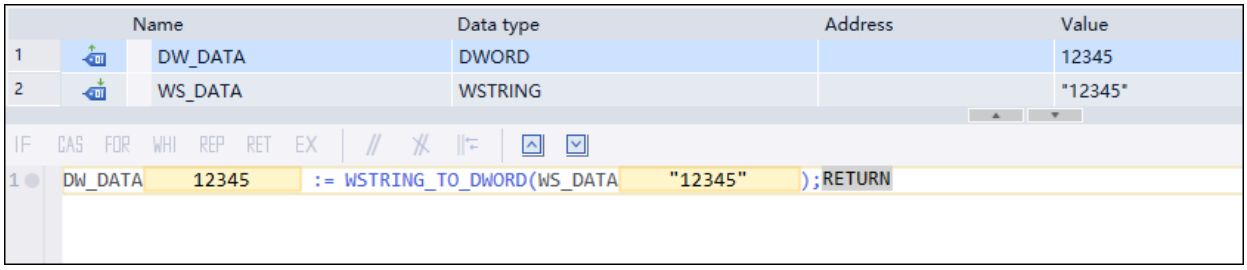
- **Program example**

  Take WSTRING_TO_DWORD as an example.

  ST:



  LD:



- **Note**

  : Take WSTRING_TO_STRING as an example, Chinese characters are not supported for conversion.

18.1.12.24 UNICODE_TO_UTF8

Converts the UNICODE encoding format to UTF8 encoding.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
UNICODE_TO_UTF8	UNICODE-to-UTF8 conversion	FC		:= UNICODE_TO_UTF8(In :=);

- **Related variable**

  Variable

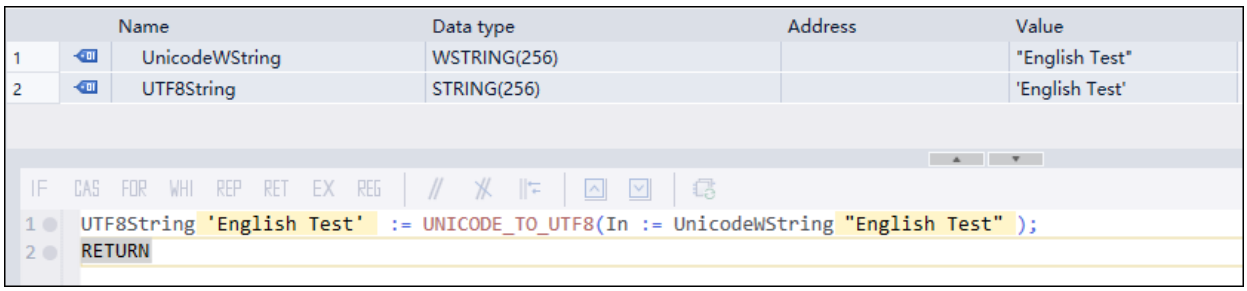
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	IN	UNICODE encoded character	WSTRING	Follows data type	-	The UNICODE encoded character to be converted
Output	UNICODE_TO_UTF8	Conversion result	STRING	Follows data type	-	The UTF8 encoded character resulting from the conversion

- **Function**

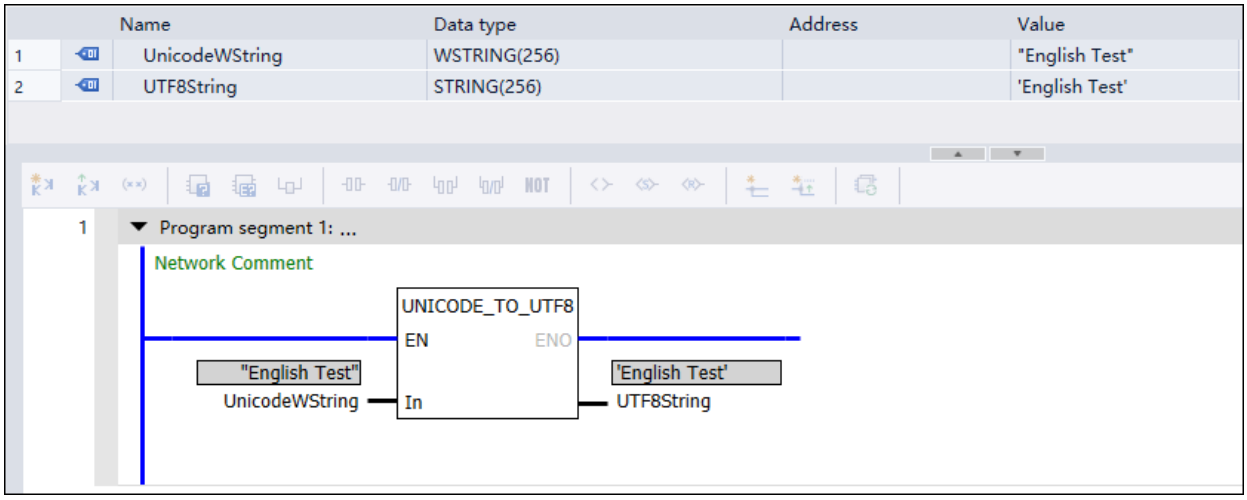
  Converts a wide string to a UTF-8 string. For data that cannot be converted to UTF8 encoding, the conversion proceeds, so abnormal data may result when illegal characters are present.

- **Program example**

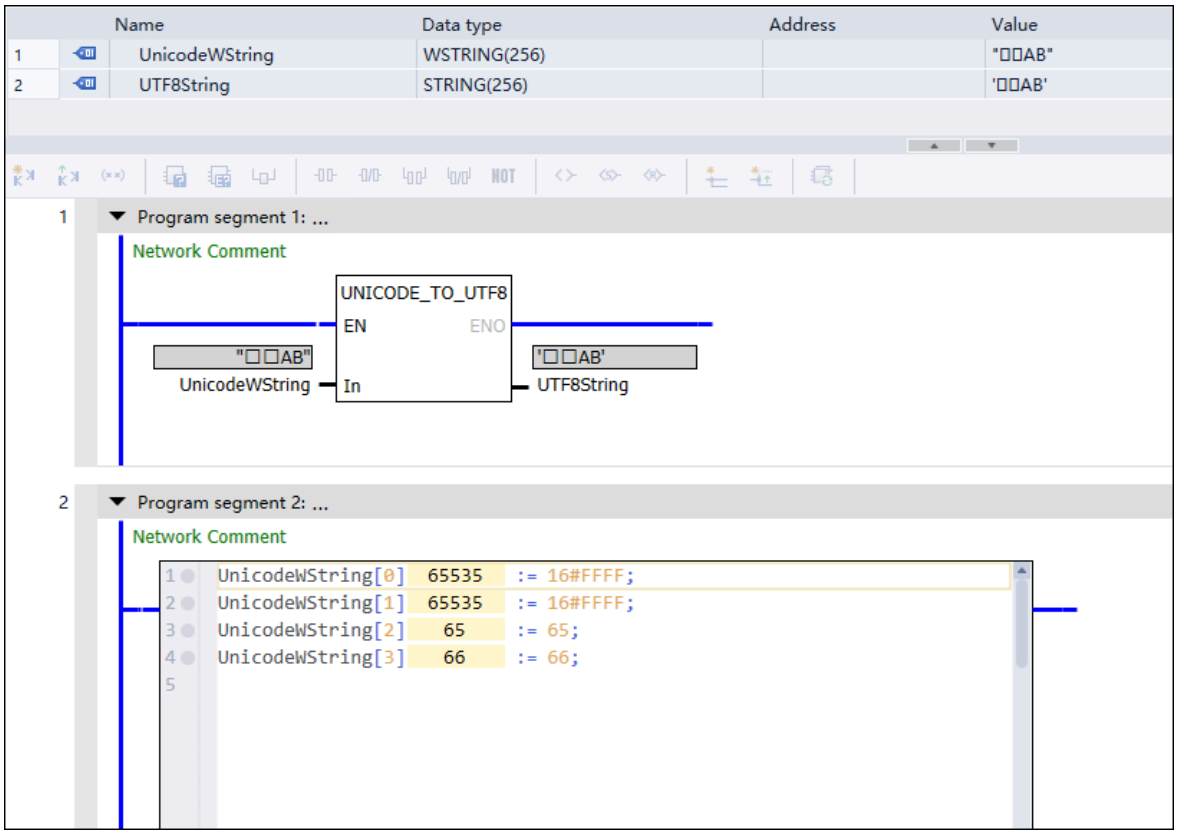
  ST:



  LD:

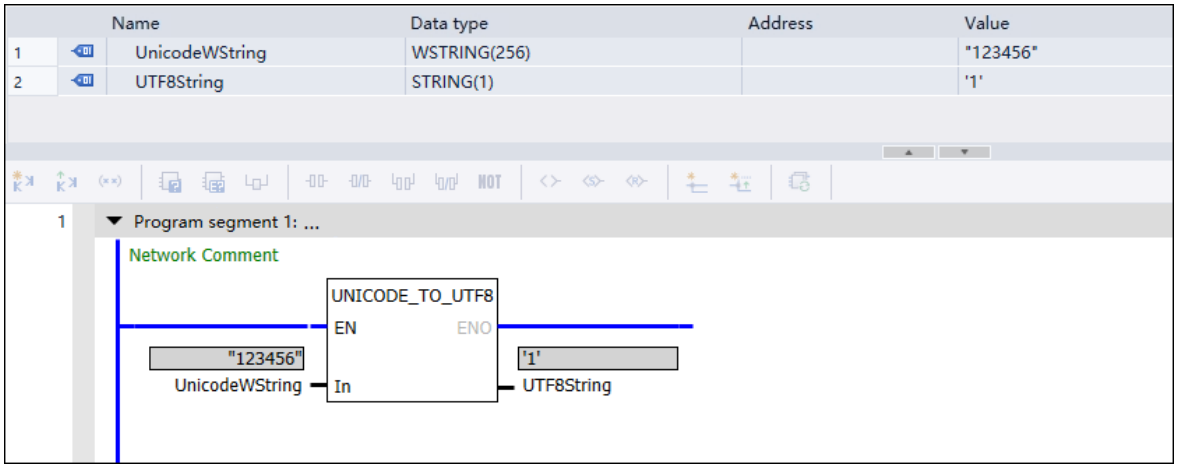


- **Precautions**
  - When the input UNICODE encoding is invalid, the encoding conversion proceeds, and illegal characters will be directly copied into the output string.

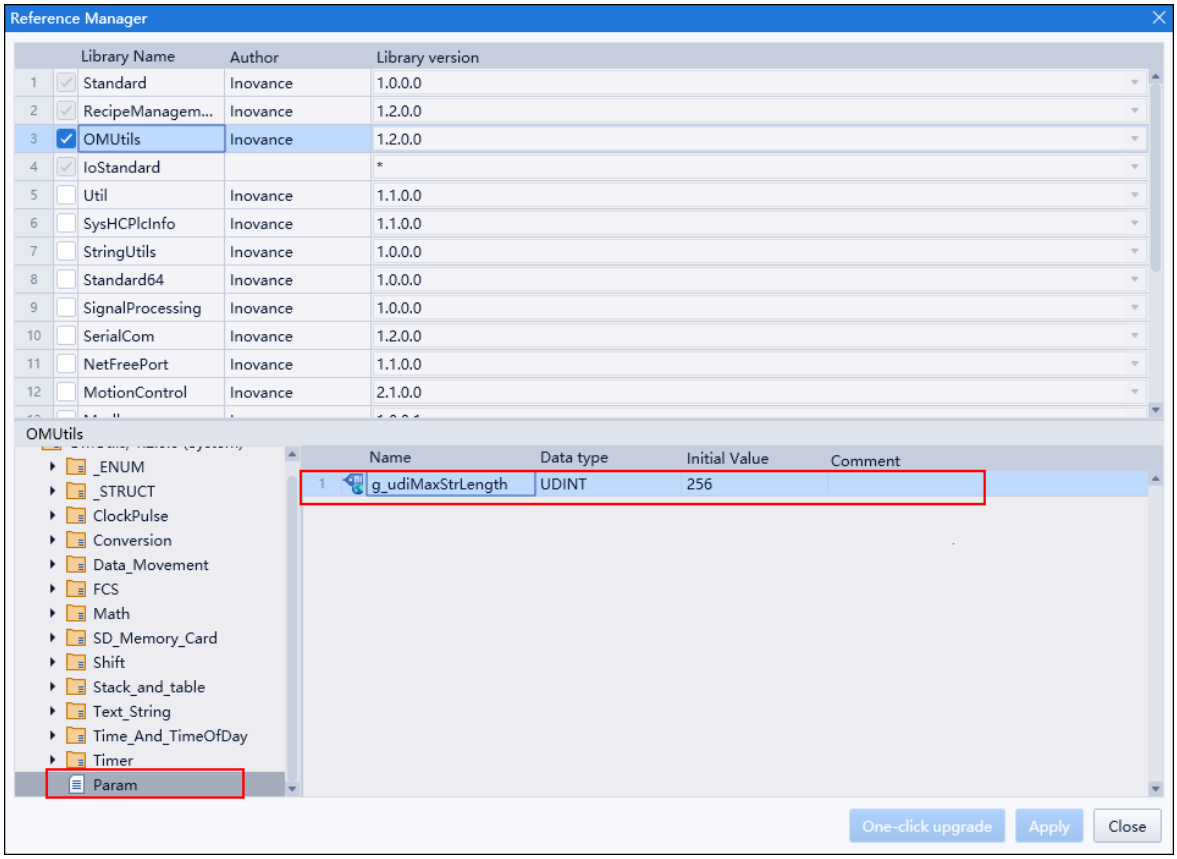


If the input UNICODE contains invalid encoding, the conversion proceeds, and valid encoding after the invalid encoding can be converted normally.

  - When the output string length is less than the length after conversion, the remaining part of the string will be truncated.



- The instruction defaults to supporting a character length of 256. The maximum supported length of the instruction can be modified by updating the global variable g_udiMaxStrLength in the OMUtil library's parameter list and then downloading the program. Parts exceeding the maximum length will be truncated.



18.2 Extended Instructions

18.2.1 System Time

18.2.1.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.1.2 GetLocalTime	FC	Gets the current time of the PLC system	Standard	Library version 1.0.0.0 and later

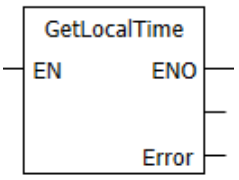
Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.1.3 GetSysTimeMs	FC	Gets the time elapsed since system startup to the current time		Library version 1.0.0.0 and later
18.2.1.4 GetSysTimeUs	FC	Gets the time elapsed since system startup to the current time		Library version 1.0.0.0 and later
18.2.1.5 GetSysTimeNs	FC	Gets the time elapsed since system startup to the current time		Library version 1.0.0.0 and later
18.2.1.6 GetSysTimeRTC	FC	Gets the real-time clock time of the current control system		Library version 1.0.0.0 and later
18.2.1.7 SetSysTimeRTC	FB	Sets the real-time clock time of the current control system		Library version 1.0.0.0 and later
18.2.1.8 GetSystemTime	FC	Gets the time elapsed since the system startup (in ms/us /ns)		Library version 1.2.0.0 and later
18.2.1.9 ADD_DT_TIME	FC	Adds time to date and time	OMUtils	Library version 1.0.0.0 and later
18.2.1.10 ADD_TOD_TIME	FC	Adds time to time of day		Library version 1.0.0.0 and later
18.2.1.11 CheckLeapYear	FC	Checks for leap year		Library version 1.0.0.0 and later
18.2.1.12 DateStructToDt	FC	Joins date and time		Library version 1.0.0.0 and later
18.2.1.13 DateToSec	FC	Converts date to seconds		Library version 1.0.0.0 and later
18.2.1.14 DaysToMonth	FC	Converts days to month		Library version 1.0.0.0 and later
18.2.1.15 DIVTIME	FC	Divides time		Library version 1.0.0.0 and later
18.2.1.16 DtToDateStruct	FC	Breaks down date and time		Library version 1.0.0.0 and later
18.2.1.17 Days to Seconds	FC	Converts date and time to seconds		Library version 1.0.0.0 and later
18.2.1.18 GetDayOfWeek	FC	Gets day of week		Library version 1.0.0.0 and later
18.2.1.19 Get Days of Month	FC	Gets the number of days in month		Library version 1.0.0.0 and later
18.2.1.20 GetSystemDate_sDt	FC	Gets the current system time		Library version 1.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.1.21 GetWeekOfYear	FC	Gets week number		Library version 1.0.0.0 and later
18.2.1.22 MULTIME	FC	Multiplies time		Library version 1.0.0.0 and later
18.2.1.23 NanoSecToTime	FC	Converts nanoseconds to time.		Library version 1.0.0.0 and later
18.2.1.24 SecToTime	FC	Converts seconds to time		Library version 1.0.0.0 and later
18.2.1.25 SecToDate	FC	Converts seconds to date		Library version 1.0.0.0 and later
18.2.1.26 SecToTod	FC	Converts seconds to time of day		Library version 1.0.0.0 and later
18.2.1.27 Second to Date	FC	Converts seconds to date and time		Library version 1.0.0.0 and later
18.2.1.28 SUB_DATE_DATE	FC	Subtracts date		Library version 1.0.0.0 and later
18.2.1.29 SUB_DT_DT	FC	Subtracts date and time		Library version 1.0.0.0 and later
18.2.1.30 SUB_DT_TIME	FC	Subtracts time from date and time		Library version 1.0.0.0 and later
18.2.1.31 SUB_TOD_TIME	FC	Subtracts time from time of day		Library version 1.0.0.0 and later
18.2.1.32 SUB_TOD_TOD	FC	Subtracts time of day		Library version 1.0.0.0 and later
18.2.1.33 TimeToNanoSec	FC	Converts time to nanoseconds		Library version 1.0.0.0 and later
18.2.1.34 TimeToSec	FC	Converts time to seconds		Library version 1.0.0.0 and later
18.2.1.35 TodToSec	FC	Converts time to seconds		Library version 1.0.0.0 and later
18.2.1.36 TruncDt	FC	Truncates date and time		Library version 1.0.0.0 and later
18.2.1.37 TruncTime	FC	Truncates time		Library version 1.0.0.0 and later
18.2.1.38 TruncTod	FC	Truncates time of day		Library version 1.0.0.0 and later

18.2.1.2 GetLocalTime

Gets the current time of the PLC system (including time zone information).

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetLocalTime	Gets the current time of the PLC system	FC		:= GetLocalTime(Error =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Return value	Follows data type	-	Returns the current time of the PLC.
	Error	Error flag	Follows data type	-	0: No fault. 1: Failed to get system time. 2: Failed to get hardware RTC time. 3: Failed to get both system time and hardware RTC time.

Variable types

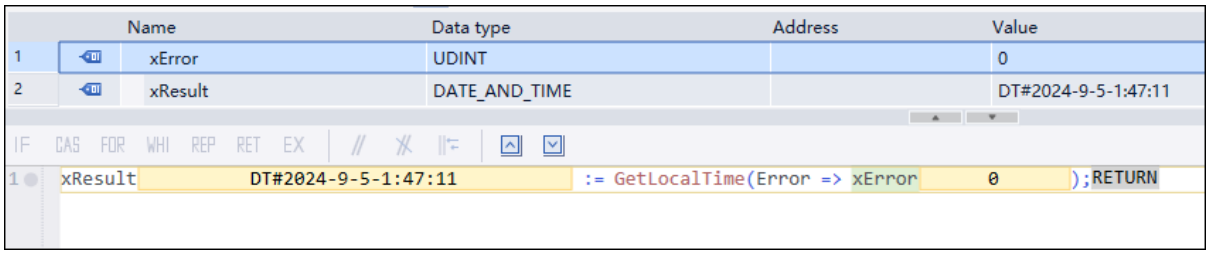
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–
Error	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

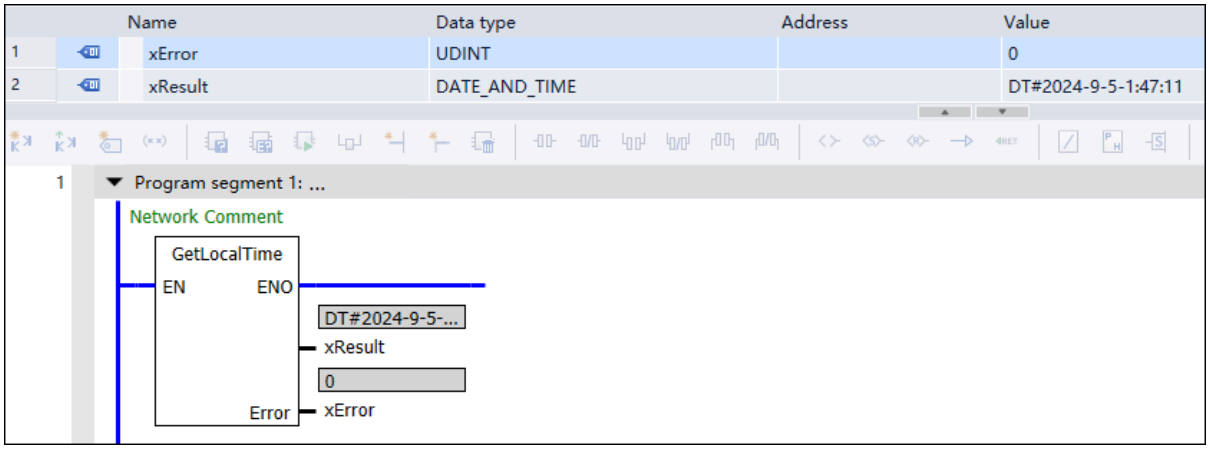
Gets the current time of the PLC system (including time zone information).

- **Program example**

ST:



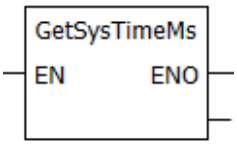
LD:



18.2.1.3 GetSysTimeMs

Gets the time elapsed since system startup in ms.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetSysTimeMs	Gets the time elapsed since system startup to the current time	FC		:= GetSysTimeMs();

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Return value	Follows data type	-	Returns the time elapsed since system startup to the current time in ms.

Variable types

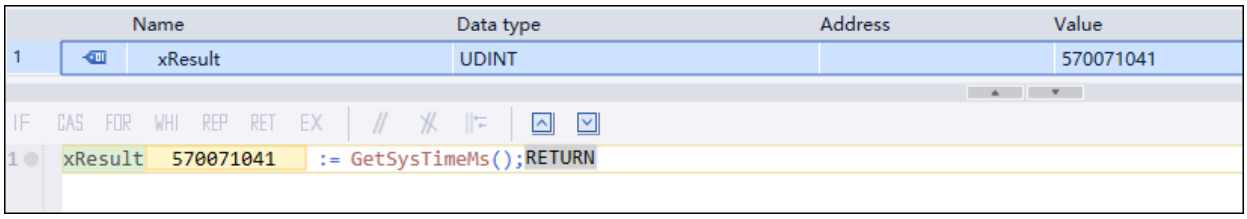
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

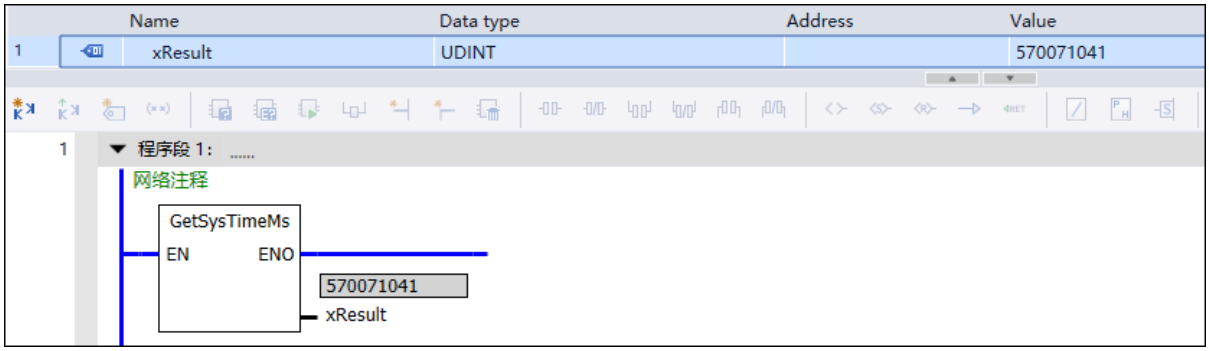
  Gets the time elapsed since system startup in ms.

- **Program example**

  ST:



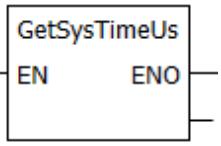
  LD:



18.2.1.4 GetSysTimeUs

Gets the time elapsed since system startup in us.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetSysTimeUs	Gets the time elapsed since system startup to the current time	FC		:= GetSysTimeUs();

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Return value	Follows data type	-	Returns the time elapsed since system startup to the current time in us.

Variable types

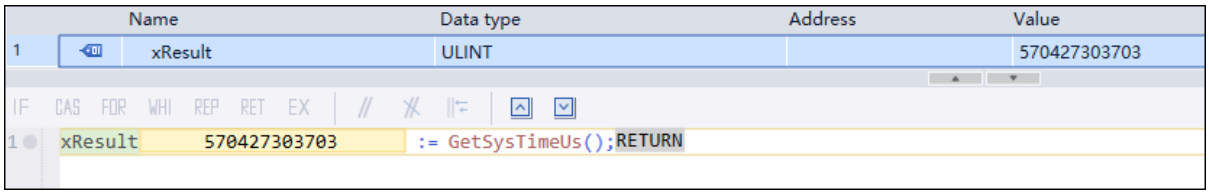
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–

- **Function**

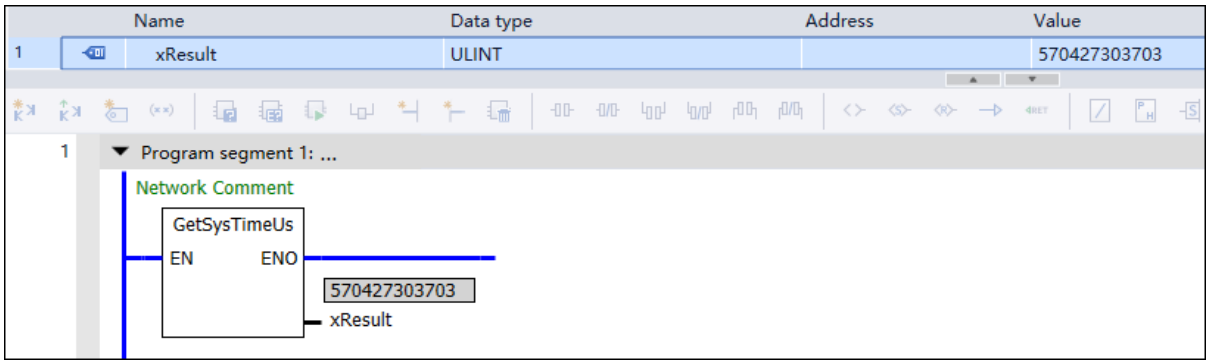
  Gets the time elapsed since system startup in us.

- **Program example**

  ST:



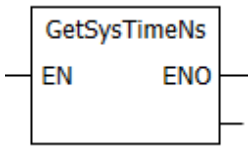
  LD:



18.2.1.5 GetSysTimeNs

Gets the time elapsed since system startup in ns.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetSysTimeNs	Gets the time elapsed since system startup to the current time	FC		:= GetSysTimeNs();

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Return value	Follows data type	-	Returns the time elapsed since system startup to the current time in ns.

  Variable types

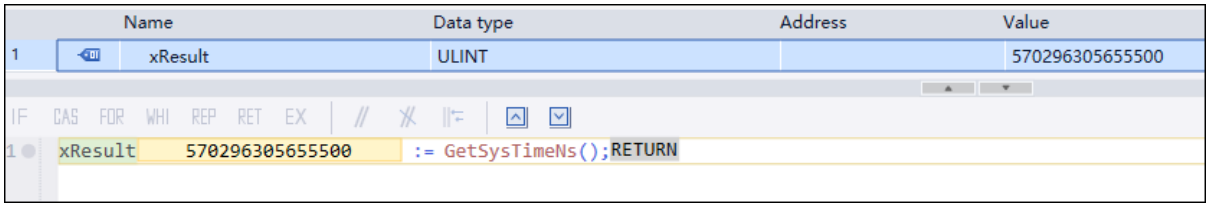
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–

- **Function**

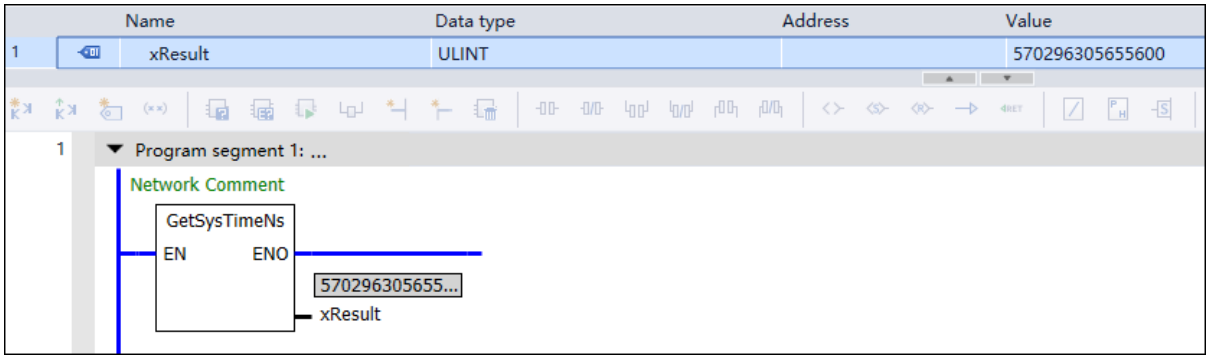
  Gets the time elapsed since system startup in ns.

- **Program example**

  ST:



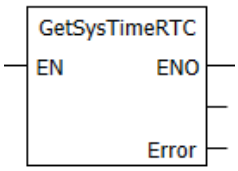
  LD:



18.2.1.6 GetSysTimeRTC

Gets the current real-time clock time of the control system, without timezone information (accurate to second).

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetSysTimeRTC	Gets the real-time clock time of the current control system	FC		:= GetSysTimeRTC(Error =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Return value	Follows data type	-	Gets the current real-time system time, displayed in DT format.
	Error	Success indicator	Follows data type	-	Indicates whether the time retrieval succeeds: 0: Retrieval successful. 1: System time retrieval failed. 2: Hardware RTC time retrieval failed.

Variable types

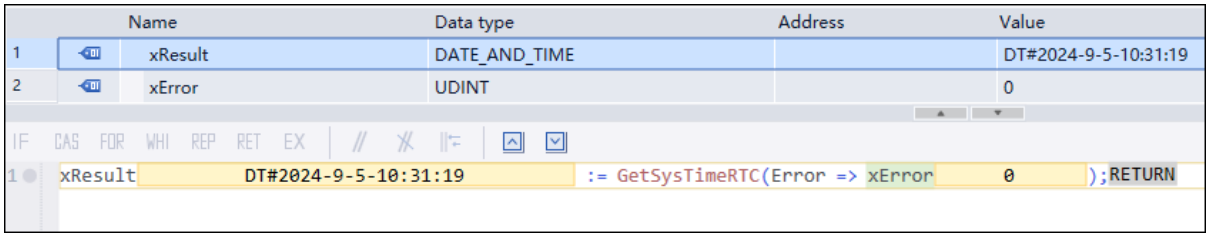
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–
Error	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

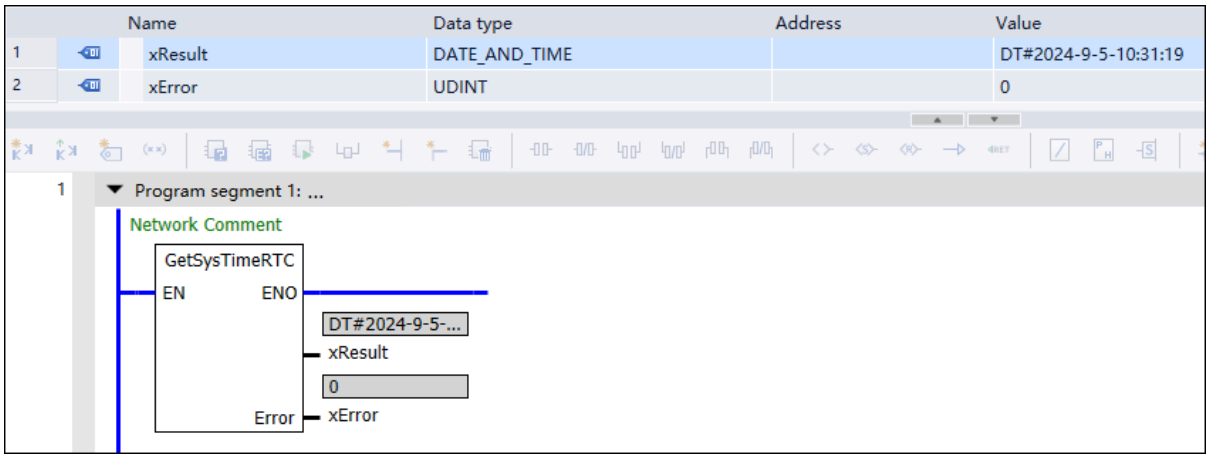
  Gets the current real-time clock time of the control system, without timezone information (accurate to second).

- **Program example**

  ST:



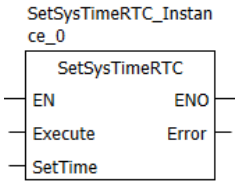
  LD:



18.2.1.7 SetSysTimeRTC

Sets the current real-time clock time of the control system, without timezone information (accurate to second).

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SetSysTimeRTC	Sets the real-time clock time of the current control system	FB		SetSysTimeRTC_Instance_0(Execute := , SetTime := Error=>);

- **Related variable**

Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Enable	Follows data type	FALSE	A rising edge-triggered switch for executing the function block
	SetTime	Set time	Follows data type	-	Sets the real-time clock time of the current system, accurate to second.
Output	Error	Success indicator	Follows data type	-	Indicates whether the time setting succeeds: 0: Setting successful. 1: System time setting failed. 2: Hardware RTC time setting failed. 3: System time can only be set up to January 19, 2038, 11:14:07 A.M. 24: This instruction is not supported in simulation mode.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SetTime	–	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Error	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Sets the current time of the control system, without timezone information (accurate to second).

- **Program example**

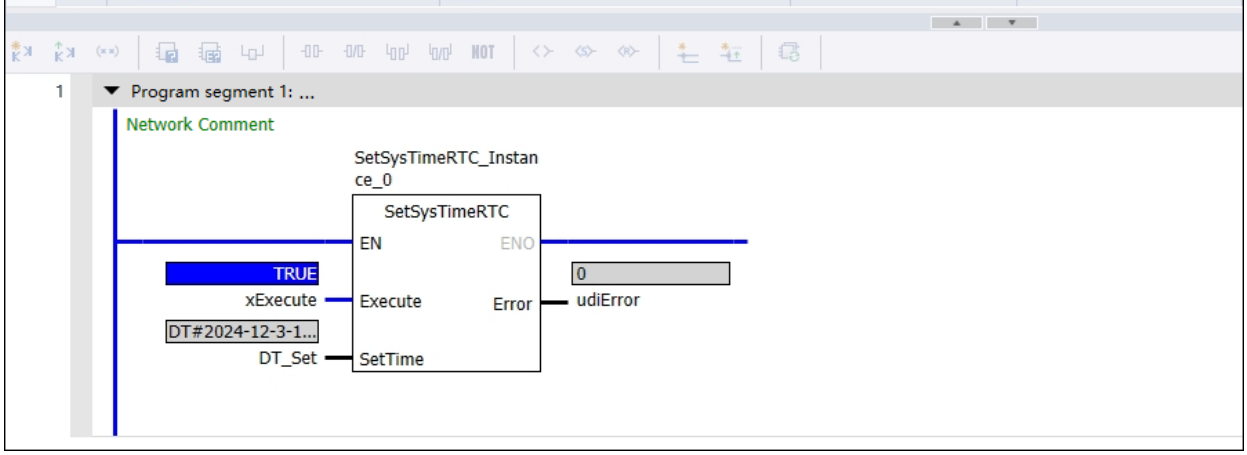
  ST:

	Name	Data type	Address	Value
1	SetSysTimeRTC_Instance_0	SetSysTimeRTC		
2	Execute	BOOL		TRUE
3	SetTime	DATE_AND_TIME		DT#2024-12-3-12:27:34
4	Error	UDINT		0
5	xExecute	BOOL		TRUE
6	DT_Set	DATE_AND_TIME		DT#2024-12-3-12:27:34
7	udiError	UDINT		0

```
1  SetSysTimeRTC_Instance_0(
2  Execute  TRUE  := xExecute  TRUE ,
3  SetTime  DT#2024-12-3-12:27:34  := DT_Set  DT#2024-12-3-12:27:34 ,
4  Error  0  => udiError  0 );
5  RETURN
```

  LD:

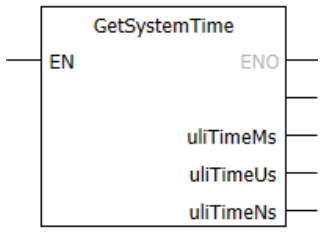
	Name	Data type	Address	Value
1	SetSysTimeRTC_Instance_0	SetSysTimeRTC		
2	Execute	BOOL		TRUE
3	SetTime	DATE_AND_TIME		DT#2024-12-3-12:27:34
4	Error	UDINT		0
5	xExecute	BOOL		TRUE
6	DT_Set	DATE_AND_TIME		DT#2024-12-3-12:27:34
7	udiError	UDINT		0



18.2.1.8 GetSystemTime

Gets the time elapsed since the system startup.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
GetSystemTime	Gets the time elapsed since the system startup (in ms/us/ns)	FC		:= GetSystemTime(uliTimeMs => , uliTimeUs => , uliTimeNs =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	uliTimeMs	Running time (ms)	Follows data type	-	Running time (ms)
	uliTimeUs	Running time (us)	Follows data type	-	Running time (us)
	uliTimeNs	Running time (ns)	Follows data type	-	Running time (ns)

Variable types

	Boolean	Bit String				Integer								Real Number		Time, duration, date, and text string				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
uliTimeMs	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–
uliTimeUs	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time, duration, date, and text string				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
uliTimeNs	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–

- **Function description**

  Gets the time elapsed since the system startup.

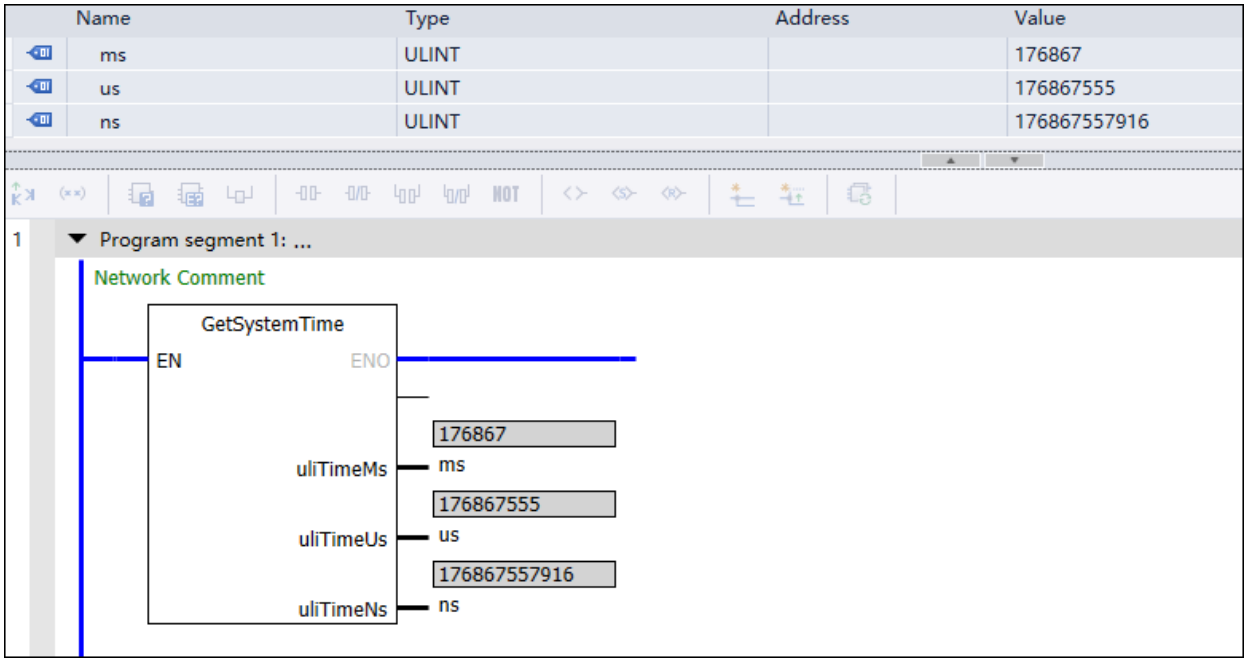
- **Program example**

  ST:

Name	Type	Address	Value	Prepared value	Comment
ms	ULINT		158127		
us	ULINT		158127556		
ns	ULINT		158127557782		

```
GetSystemTime(uliTimeMs => ms , uliTimeUs => us , uliTimeNs => ns );
```

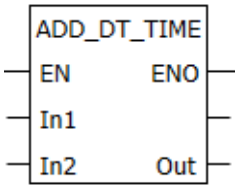
  LD:



18.2.1.9 ADD_DT_TIME

Adds "In2" (time) to "In1" (date and time), and returns "Out" (date and time).

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ADD_DT_TIME	Adds time to date and time	FC		:= ADD_DT_TIME(In1 := , In2 := , Out =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Date and time to be added	Follows data type	DATE_AND_TIME#1970-1-1-0:0:0	Date and time to be added
	In2	Time to add	Follows data type	TIME#0ms	Time to add
Output	Out	Addition result (date and time)	Follows data type	-	Addition result (date and time)

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–
In2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–

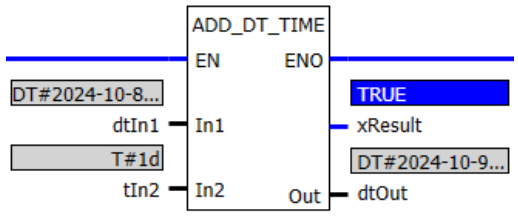
- **Function**

Adds "In2" (time) to "In1" (date and time), and returns "Out" (date and time).

- **Program example**

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    xResult: BOOL;
    dtIn1: DATE_AND_TIME := DT#2023-5-22-14:39:17;
    tIn2: TIME := T#1D;
    dtOut: DATE_AND_TIME;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>


</td><td>

```
xResult TRUE := ADD_DT_TIME(In1 := dtIn1 DT#2024-10-8-14:39:17,
    In2 := tIn2 T#1d,
    Out => dtOut DT#2024-10-9-14:39:17 );RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

Name	Data type	Address	Value
xResult	BOOL		TRUE
dtIn1	DATE_AND_TIME		DT#2024-10-8-14:39:17
tIn2	TIME		T#1d
dtOut	DATE_AND_TIME		DT#2024-10-9-14:39:17

</td></tr>
</table>

- **Precautions**

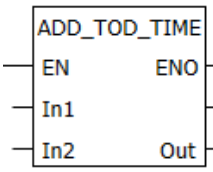
  When the result of addition exceeds the valid range of Out, no program fault occurs and the following operation is executed.

  DT#2106-2-7-6:28:15 + T#1s → DT#1970-1-1-0:0:0

18.2.1.10 ADD_TOD_TIME

Adds "In2" (time) to "In1" (time of day), and returns "Out" (time of day).

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ADD_TOD_TIME	Adds time to time of day	FC		:= ADD_TOD_TIME(In1 := , In2 := , Out =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Time of day to be added	Follows data type	TIME_OF_DAY#0:0:0	Time of day to be added
	In2	Time to add	Follows data type	TIME#0ms	Time to add

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Addition result (time of day)	Follows data type	-	Addition result (time of day)

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–
In2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–

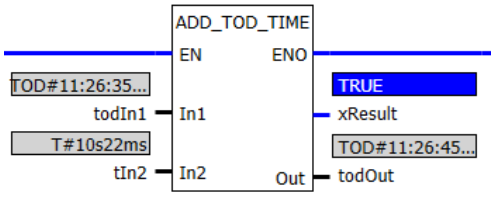
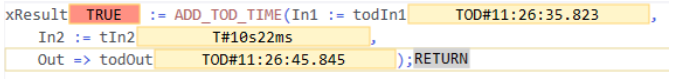
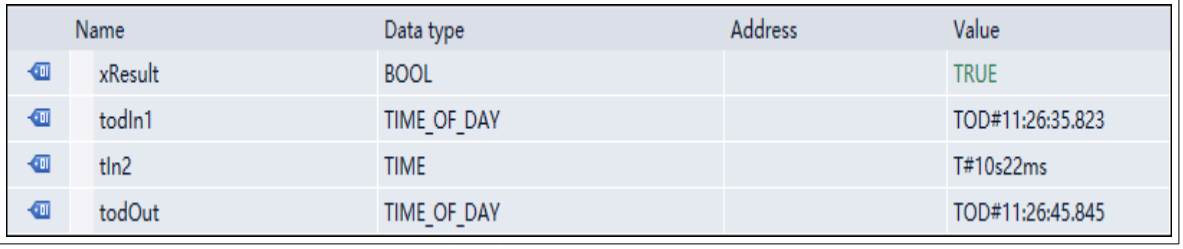
- **Function**

  Adds "In2" (time) to "In1" (time of day), and returns "Out" (time of day).

- **Program example**

Item	LD	ST

Define variables	VAR
	xResult: BOOL;
	todIn1: TIME_OF_DAY := TOD#11:26:35.823;
	tIn2: TIME := T#10s22ms;
	todOut: TIME_OF_DAY;
	END_VAR	

Program		
Operation result		

Name	Data type	Address	Value
xResult	BOOL		TRUE
todIn1	TIME_OF_DAY		TOD#11:26:35.823
tIn2	TIME		T#10s22ms
todOut	TIME_OF_DAY		TOD#11:26:45.845

- **Precautions**

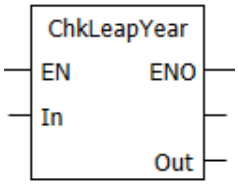
When the result of addition exceeds the valid range of Out, no program fault occurs and the following operation is executed.

TOD#23:59:59.999 + T#1s1ms → TOD#0:0:1

18.2.1.11 ChkLeapYear

Checks whether the input year is a leap year.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ChkLeapYear	Checks for leap year	FC		:= ChkLeapYear(In := , Out =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Year	1970 to 2106	1970	Year
Output	Out	Check result	Follows data type	-	TRUE: "In" is a leap year. FALSE: "In" is not a leap year.

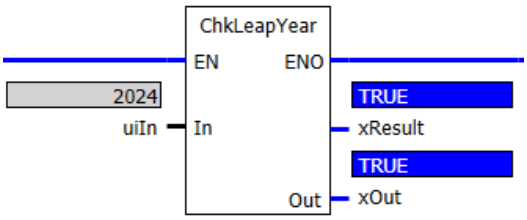
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

This instruction checks whether the year In is a leap year. If so, the result Out is TRUE; if not, Out is FALSE.

- **Program example**

Item	LD	ST
Define variables	VAR xResult: BOOL; uiIn: UINT := 2024; xOut: BOOL; END_VAR	
Program		xResult TRUE := ChkLeapYear(In := uiIn 2024 , Out => xOut TRUE);RETURN
Operation result		

Name	Data type	Address	Value
xResult	BOOL		TRUE
uiIn	UINT		2024
xOut	BOOL		TRUE

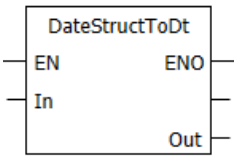
- **Precautions**

  When the value of In exceeds the valid range, a program fault occurs, the value of Out is FALSE, and the return value is FALSE.

18.2.1.12 DateStructToDt

Joins year, month, day, hour, minute, and second into date and time.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DateStructToDt	Joins date and time	FC		:= DateStructToDt(In := , Out =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Date and time	Follows the data type (_sDT)	-	Year, month, day, hour, minute, and second to be joined
Output	Out	Date and time	DT#1970-1-1-0:0:0 to DT#2106-2-7-6:28:15	-	Date and time after joining

  _sDT structure

Variable	Name	Content	Data Type	Valid Range	Initial Value	Unit
uiYear	Year	Year	UINT	1970 to 2106	1970	Year
uiMonth	Month	Month	UINT	1 to 12	1	Month
uiDay	Day	Day	UINT	1 to 31	1	Day
uiHour	Hour	Hour	UINT	0 to 23	0	Hour
uiMinute	Minute	Minute	UINT	0 to 59	0	Minute
uiSecond	Second	Second	UINT	0 to 59	0	Second
uiMillisecond	Millisecond	Millisecond	UINT	0 to 999	0	Millisecond
uiDayOfWeek	Day of the week	Day of the week	UINT	1 to 7	1	-

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	_sDT																			
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–

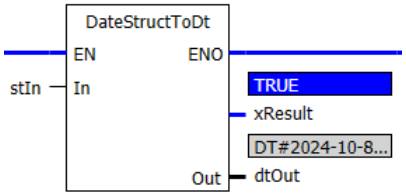
- **Function**

  Joins year, month, day, hour, minute, and second into date and time.

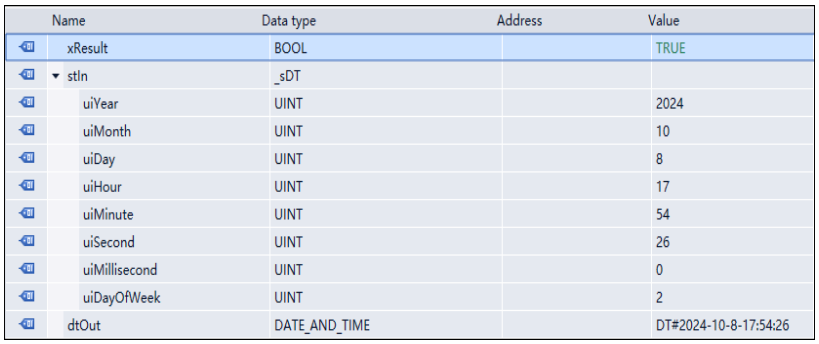
- **Program example**

Item	LD	ST
Define variables		
Program		

```
VAR
    xResult: BOOL;
    stIn: _sDT := (uiYear := 2024, uiMonth := 10, uiDay := 8, uiHour := 17, uiMinute := 54, uiSecond := 26, uiDayOfWeek := 2);
    dtOut: DATE_AND_TIME;
END_VAR
```




```
xResult := DateStructToDt(In := stIn,
Out => dtOut );RETURN
```

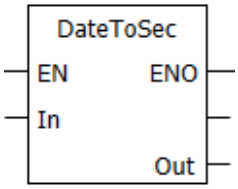
Item	LD	ST
Operation result		

- **Precautions**
  - The valid range of the output DT Out is DT#1970-1-1-0:0:0 to DT#2106-2-7-6:28:15.
  - If the value of Out exceeds the valid range, the return value is FALSE and the operation result is DT#1970-1-1-0:0:0.
  - If the value of In exceeds the valid range of each element in the _sDT structure, the return value is FALSE and the operation result is DT#1970-1-1-0:0:0.
  - The initial value of each element in the _sDT structure is the breakdown value of DT#1970-1-1-0:0:0.

18.2.1.13 DateToSec

Converts a date to the number of seconds from 00:00:00 on January 1, 1970.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DateToSec	Converts date to seconds	FC		:= DateToSec(In := , Out =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Date	Follows data type.	DATE#1970-1-1	Date
Output	Out	Conversion result	0 to 4294967295	0	Number of seconds since 00:00:00 on January 1, 1970

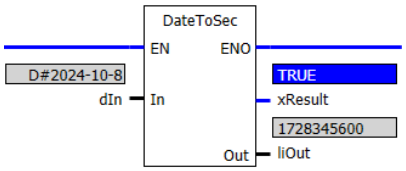
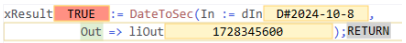
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–

- **Function**

  Returns the number of seconds from 00:00:00 on January 1, 1970 to 00:00:00 on the date specified by In. The unit is Out is second.

- **Program example**

Item	LD	ST
Define variables	VAR xResult: BOOL; dIn: DATE := D#2024-10-08; liOut: LINT; END_VAR	
Program		
Operation result		

Name	Data type	Address	Value
xResult	BOOL		TRUE
dIn	DATE		D#2024-10-8
liOut	LINT		1728345600

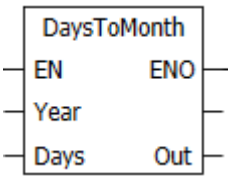
- **Precautions**

  To convert the number of seconds from 00:00:00 on January 1, 1970 to a date, use the 18.2.1.25 SecToDate instruction.

18.2.1.14 DaysToMonth

Calculates the month based on the number of days from January 1.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DaysToMonth	Converts days to month	FC		:= DaysToMonth(Year := , Days := , Out =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Year	Year	1970 to 2106	1970	Year
	Days	Number of days	Year ranges from 1 to 365 when it is a common year. Year ranges from 1 to 366 when it is a leap year.	1	Number of days from January 1
Output	Out	Month	1 to 12	0	Month

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Year	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Days	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–

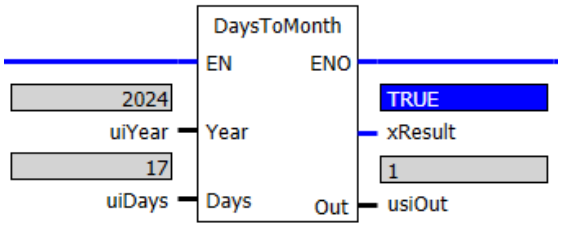
- **Function**

  Calculates the month based on Days (number of days from January 1 in Year).

- **Program example**

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    xResult: BOOL;
    uiYear: UINT := 2024;
    uiDays: UINT := 17;
    usiOut: USINT;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>
</td><td>

```
xResult := DaysToMonth(Year := uiYear ,
           Days := uiDays ,
           Out => usiOut );RETURN
```

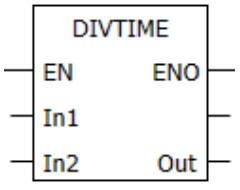
</td></tr>
<tr><td>Operation result</td><td colspan="2">
<table>
<tr><th></th><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td></td><td>xResult</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td></td><td>uiYear</td><td>UINT</td><td></td><td>2024</td></tr>
<tr><td></td><td>uiDays</td><td>UINT</td><td></td><td>17</td></tr>
<tr><td></td><td>usiOut</td><td>USINT</td><td></td><td>1</td></tr>
</table>
</td></tr>
</table>

- **Precautions**
  - When the value of Year exceeds the valid range, no fault occurs, the value of Out is 0, and the return value is FALSE.
  - When the value of Days exceeds the valid range, no fault occurs, the value of Out is 0, and the return value is FALSE.

18.2.1.15 DIVTIME

Divides a time by a specified number.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DIVTIME	Divides time	FC		:= DIVTIME(In1 := , In2 := , Out =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Dividend time	Follows data type	TIME#0ms	Dividend time
	In2	Divider	Follows data type	-	Divider

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Result time of division	Follows data type	-	Result time of division

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
In2	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–

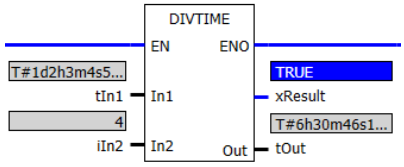
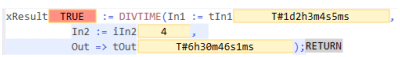
- **Function**

  This instruction divides a time In1 by a number In2. The result of division Out is a time.

- **Program example**

Item	LD	ST

Define variables	VAR
	xResult: BOOL;
	tIn1: TIME := T#1d2h3m4s5ms;
	iIn2: INT := 4;
	tOut: TIME;
	END_VAR	

Program		
Operation result		

Name	Data type	Address	Value
xResult	BOOL		TRUE
tIn1	TIME		T#1d2h3m4s5ms
iIn2	INT		4
tOut	TIME		T#6h30m46s1ms

- **Precautions**
  - When the division result exceeds the valid range of Out, the value of Out is T#0ms and the return value is FALSE.
  - When In2 is negative, the value of Out is T#0ms, and the return value is FALSE.

- When In2 is an integer with a value of 0, the value of Out is T#0ms, and the return value is FALSE.
- When In2 is a real number:
  - When the input value of In2 makes the value of Out exceed the upper limit of the TIME type, the value of Out is T#0ms and the return value is FALSE.
  - Division result less than 1 ms is rounded. When In2 is a real number, the division result may have an error of several milliseconds due to a higher number of significant digits.
  - When In2 is a valid value and is greater than 4294967295, the value of Out is T#0ms, and the function returns FALSE.
- When the value of In2 is 0, positive infinity, negative infinity, or non-numeric data, the corresponding value of Out is listed in the following table.

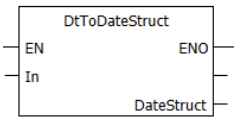
Value of In2	Value of Out
0	T#0ms
+∞	T#0ms
-∞	T#0ms
Non-numeric	T#0ms

Note: If In2 is a REAL value less than 1e-6, or an LREAL value less than 1e-15, it is treated as zero, resulting in a division-by-zero operation.

18.2.1.16 DtToDateStruct

Converts date and time to year, month, day, hour, minute, and second.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DtToDateStruct	Breaks down date and time	FC		:= DtToDateStruct(In := , DateStruct =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Date and time	Follows data type	DT#1970-1-1-0:0:0	Date and time to break down
Output	DateStruct	Date and time	Follows the data type (_sDT)	-	Date and time after breakdown

  _sDT structure

Variable	Name	Content	Data Type	Valid Range	Initial Value	Unit
uiYear	Year	Year	UINT	1970 to 2106	1970	Year
uiMonth	Month	Month	UINT	1 to 12	1	Month
uiDay	Day	Day	UINT	1 to 31	1	Day
uiHour	Hour	Hour	UINT	0 to 23	0	Hour
uiMinute	Minute	Minute	UINT	0~9	0	Minute
uiSecond	Second	Second	UINT	0 to 59	0	Second
uiMillisecond	Millisecond	Millisecond	UINT	0~999	0	Millisecond
uiDayOfWeek	Day of the week	Day of the week	UINT	1 to 7	1	-

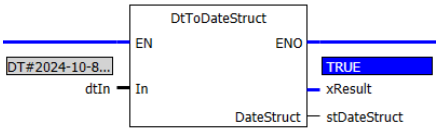
- **Variable types**

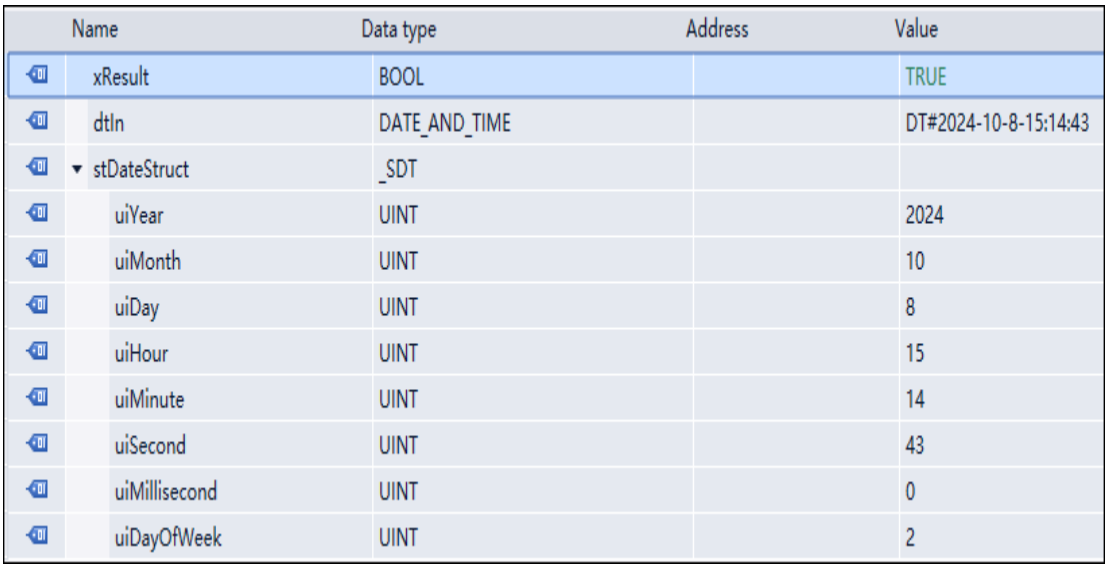
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–
DateStruct	_sDT																			

- **Function**

  Converts In (date and time) to year, month, day, hour, minute, and second in the _sDT structure format.

- **Program example**

Item	LD	ST
Define variables	VAR xResult: BOOL; dtIn: DATE_AND_TIME := DT#2024-10-08-15:14:43; stDateStruct: _SDT; END_VAR	
Program		xResult TRUE := DtToDateStruct(In := dtIn DT#2024-10-8-15:14:43 , DateStruct => stDateStruct);RETURN

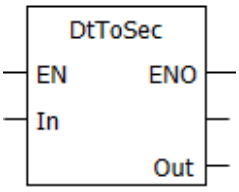
Item	LD	ST
Operation result		

- **Precautions**
  - The valid range of the input date and time In is DT#1970-1-1-0:0:0 to DT#2106-2-7-6:28:15.
  - If the value of In exceeds the valid range, the return value is FALSE and the operation result is the breakdown result of DT#1970-1-1-0:0:0.
  - The initial value of each element in the _sDT structure is the breakdown value of DT#1970-1-1-0:0:0.

18.2.1.17 DtToSec

Converts a date and time to the number of seconds from 00:00:00 on January 1, 1970.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DtToSec	Converts date and time to seconds	FC		:= DtToSec(In := , Out =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Date and time	Follows data type	DATE_AND_TIME#1970-1-1-0:0:0	Date and time
Output	Out	Conversion result	0 to 4294967295	0	Number of seconds since 00:00:00 on January 1, 1970

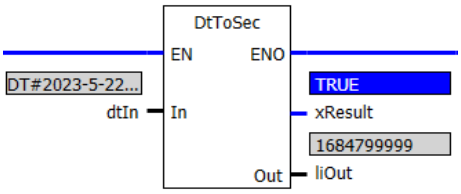
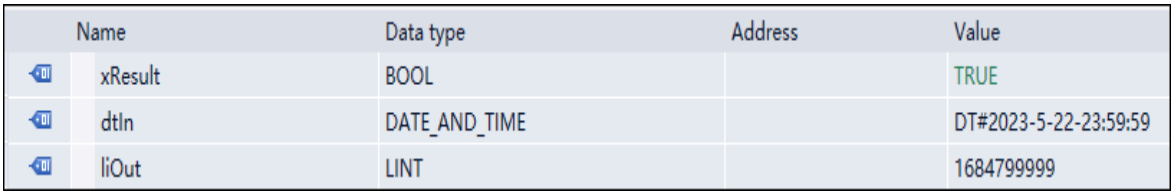
  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–

- **Function**

  Converts In (date and time) to the number of seconds from 00:00:00 on January 1, 1970. The converted value is in seconds, and values less than one second are truncated.

- **Program example**

Item	LD	ST
Define variables	VAR xResult: BOOL; dtIn: DATE_AND_TIME := DT#2023-5-22-23:59:59; liOut: LINT; END_VAR	
Program		xResult TRUE := DtToSec(In := dtIn DT#2023-5-22-23:59:59 , Out => liOut 1684799999);RETURN
Operation result		

	Name	Data type	Address	Value
	xResult	BOOL		TRUE
	dtIn	DATE_AND_TIME		DT#2023-5-22-23:59:59
	liOut	LINT		1684799999

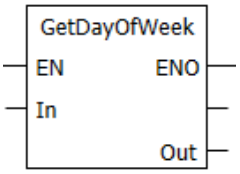
- **Precautions**

  To convert the number of seconds from 00:00:00 on January 1, 1970 to a date, use the 18.2.1.25 SecToDate instruction.

18.2.1.18 GetDayOfWeek

Gets the day of the week for the specified date or date and time.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetDayOfWeek	Gets day of week	FC		:= GetDayOfWeek(In := , Out =>);

- **Related variables**

  Input variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Date or date and time	Range for data type DATE: DATE#1970-1-1 to DATE#2106-2-7 Range for data type DT: DT#1970-1-1-0:0:0 to DT#2106-2-7-6:28:15	-	Data type DATE: Year-month-day Data type DT: Year-month-day-hour-minute-second
Output	Out	Week	Follows data type (_eDAYOFWEEK)	-	Day of week

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	√	–
Out	See "Function description" for the enumerators of enumeration type _eDAYOFWEEK.																			

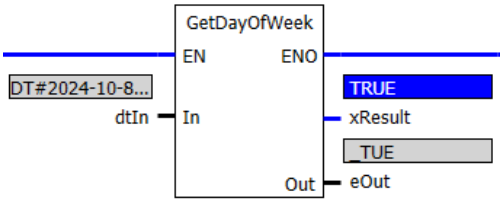
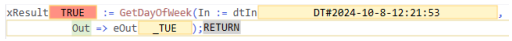
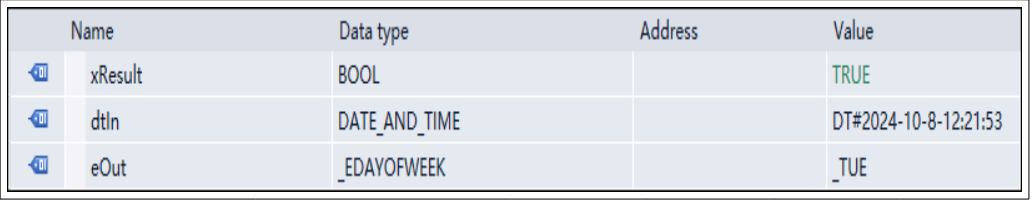
- **Function**

  Gets the day of the week for the specified date or date and time. The data type of Out is enumerated type _eDAYOFWEEK. The meanings of the enumerators are as follows.

Enumeration Value	Definition
_MON	Monday

Enumeration Value	Definition
_TUE	Tuesday
_WED	Wednesday
_THU	Thursday
_FRI	Friday
_SAT	Saturday
_SUN	Sunday

- **Program example**

Item	LD	ST
Define variables	VAR xResult: BOOL; dtIn: DATE_AND_TIME := DT#2024-10-08-12:21:53; eOut: _EDAYOFWEEK; END_VAR	
Program		
Operation result		

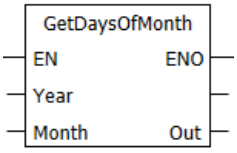
Name	Data type	Address	Value
xResult	BOOL		TRUE
dtIn	DATE_AND_TIME		DT#2024-10-8-12:21:53
eOut	_EDAYOFWEEK		_TUE

- **Precautions**
  - When the value of In exceeds the valid range, a program fault occurs, the value of Out is _MON, and the return value is FALSE.
  - The input parameter passed to In must be a variable; if a constant is passed, a compilation fault occurs.

18.2.1.19 GetDaysOfMonth

Gets the number of days in the specified month.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetDaysOfMonth	Gets the number of days in month	FC		:= GetDaysOfMonth(Year := , Month := , Out =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Year	Year	1970 to 2106	1970	Year
	Month	Month	1 to 12	1	Month
Output	Out	Number of days	28 to 31	-	Number of days

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Year	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Month	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–

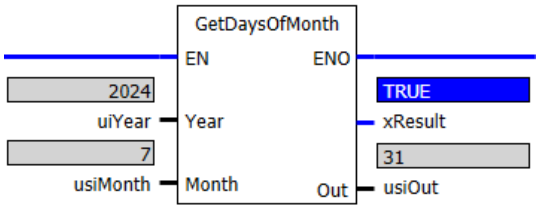
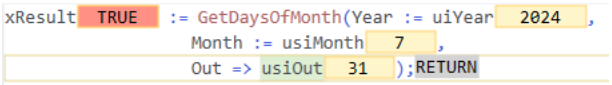
- **Function**

  Gets the number of days in the month specified by "Month" and "Year".

- **Program example**

Item	LD	ST

Define variables		VAR
		xResult: BOOL;
		uiYear: UINT := 2024;
		usiMonth: USINT := 7;
		usiOut: USINT;
		END_VAR

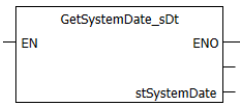
<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>
</td><td>
</td></tr>
<tr><td>Operation result</td><td colspan="2">
<table>
<tr><th></th><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td></td><td>xResult</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td></td><td>uiYear</td><td>UINT</td><td></td><td>2024</td></tr>
<tr><td></td><td>usiMonth</td><td>USINT</td><td></td><td>7</td></tr>
<tr><td></td><td>usiOut</td><td>USINT</td><td></td><td>31</td></tr>
</table>
</td></tr>
</table>

- **Precautions**
  - When the value of Year exceeds the valid range, the value of Out is 0, and the return value is FALSE.
  - When the value of Month exceeds the valid range, the value of Out is 0, and the return value is FALSE.

18.2.1.20 GetSystemDate_sDt

Outputs the current system time in _sDT structure format.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetSystemDate_sDt	Gets the current system time	FC		:= GetSystemDate_sDt(stSystemDate =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	stSystemDate	System time	Follows data type	-	Structure of system time

Variable types

	Boolean	Bit String					Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
stSystemDate	_sDT																			

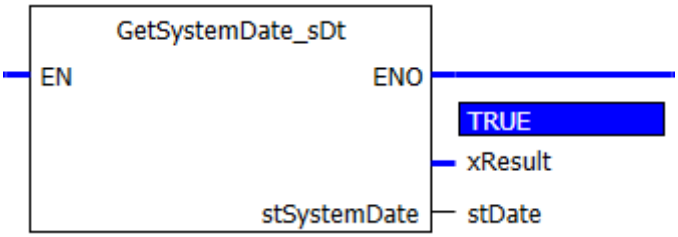
_sDT structure

Variable	Name	Content	Data Type	Valid Range	Initial Value	Unit
uiYear	Year	Year	UINT	1970 to 2106	0	Year
uiMonth	Month	Month	UINT	1 to 12	0	Month
uiDay	Day	Day	UINT	1 to 31	0	Day
uiHour	Hour	Hour	UINT	0 to 23	0	Hour
uiMinute	Minute	Minute	UINT	0 to 59	0	Minute
uiSecond	Second	Second	UINT	0 to 59	0	Second
uiMillisecond	Millisecond	Millisecond	UINT	0 to 999	0	Millisecond
uiDayOfWeek	Day of the week	Day of the week	UINT	1 to 7	0	-

- **Function**

  Outputs the current system time in _sDT structure format.

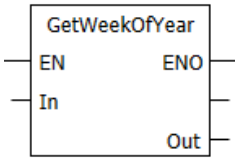
- **Program example**

Item	LD	ST
Define variables	VAR stDate: _SDT; xResult: BOOL; END_VAR	
Program		
Operation result		

18.2.1.21 GetWeekOfYear

Gets the week number for the specified date or date and time.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetWeekOfYear	Gets week number	FC		:= GetWeekOfYear(In := , Out =>);

- **Related variables**

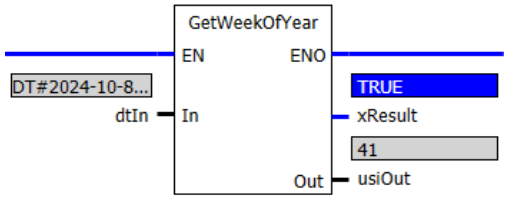
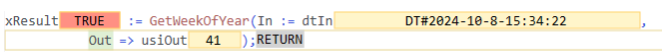
Input variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Date or date and time	Range for data type DATE: DATE#1970-1-1 to DATE#2106-2-7 Range for data type DATE_AND_TIME: DATE_AND_TIME#1970-1-1-0:0:0 to DT#2106-2-7-6:28:15	-	Data type DATE: Year-month-day Data type DATE_AND_TIME: Year-month-day-hour-minute-second
Output	Out	Week number	1 to 54	-	Week number

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	√	–
Out	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - Gets the week number for the date or date and time specified by In.
  - A week is considered from Monday to Sunday. The week count increments upon the transition from Sunday to Monday.
  - January 1 is always in week 1.
- **Program example**

Item	LD	ST
Define variables	VAR xResult: BOOL; dtIn: DATE_AND_TIME := DT#2024-10-08-15:34:22; usiOut: USINT; END_VAR	
Program		
Operation result		

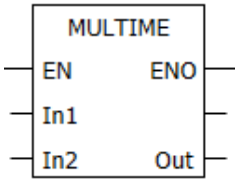
Name	Data type	Address	Value
xResult	BOOL		TRUE
dtIn	DATE_AND_TIME		DT#2024-10-8-15:34:22
usiOut	USINT		41

- **Precautions**
  - When the value of In exceeds the valid range, a program fault occurs, the value of Out is 0, and the return value is FALSE.
  - The input parameter transmitted to In must be a variable. If a constant is transmitted to In, a compilation error occurs.

18.2.1.22 MULTIME

Multiplies a time by a specified number.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
MULTIME	Multiplies time	FC		:= MULTIME(In1 := , In2 := , Out =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Multiplicand time	Follows data type	TIME#0ms	Multiplicand time
	In2	Multiplier	Follows data type	-	Multiplier
Output	Out	Product result time	Follows data type	-	Product result time

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
In2	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–

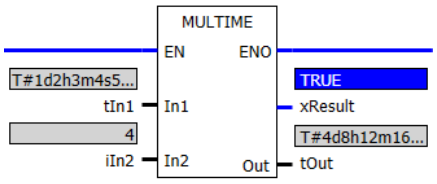
- **Function**

  This instruction multiplies a time In1 by multiplier In2. The result of multiplication in Out is a time.

- **Program example**

Item	LD	ST

Define variables	VAR
	xResult: BOOL;
	tIn1: TIME := T#1d2h3m4s5ms;
	iIn2: INT := 4;
	tOut: TIME;
	END_VAR	

Program		xResult TRUE := MULTIME(In1 := tIn1 T#1d2h3m4s5ms , In2 := iIn2 4 , Out => tOut T#4d8h12m16s20ms);RETURN
Operation result		

	Name	Data type	Address	Value
	xResult	BOOL		TRUE
	tIn1	TIME		T#1d2h3m4s5ms
	iIn2	INT		4
	tOut	TIME		T#4d8h12m16s20ms

- **Precautions**
  - When the multiplication result exceeds the valid range of Out, the value of Out is T#0ms and the return value is FALSE.
  - When In2 is negative, a program fault occurs, the value of Out is T#0ms, and the return value is FALSE.
  - When the input value of In2 makes the value of Out exceed the upper limit of the TIME type, the value of Out is T#0ms and the return value is FALSE.
  - When In2 is a real number, any fractional part of the product result that is less than 1 ms is truncated.

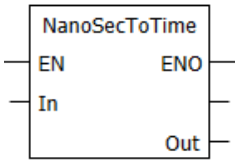
- When In2 is a real number, the product result containing more valid digits may cause an error of several milliseconds.
- When the value of In2 is positive infinity, negative infinity, or non-numeric data, the corresponding value of Out is listed in the following table.

"In2" Value	Value of Out
+∞	T#0ms
-∞	T#0ms
Non-numeric	T#0ms

18.2.1.23 NanoSecToTime

This instruction converts the number of nanoseconds to a time.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
NanoSecToTime	Converts nanoseconds to time.	FC		:= NanoSecToTime(In := , Out =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Number of nanoseconds	0 to 4294967295000000	0	Number of nanoseconds
Output	Out	Conversion result	Follows data type	TIME#0ms	Time

Variable types

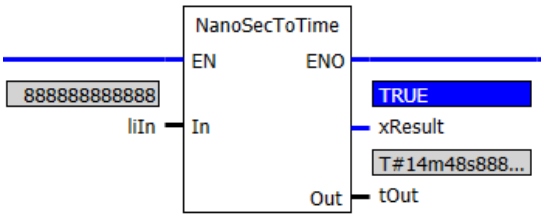
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–

- **Function**

  Converts nanoseconds "In" to time.

- **Program example**

Item	LD	ST
Define variables	VAR xResult: BOOL; liIn: LINT := 888888888888; tOut: TIME; END_VAR	
Program		xResult TRUE := NanoSecToTime(In := liIn 888888888888 , Out => tOut T#14m48s888ms);RETURN
Operation result		

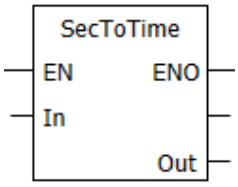
Name	Data type	Address	Value
xResult	BOOL		TRUE
liIn	LINT		888888888888
tOut	TIME		T#14m48s888ms

- **Precautions**
  - If In is beyond the input range, the function returns FALSE and outputs T#0ms.
  - To convert a time to the number of nanosecond, use the 18.2.1.33 TimeToNanoSec instruction.

18.2.1.24 SecToTime

Converts the number of seconds to a time.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SecToTime	Converts seconds to time	FC		:= SecToTime(In := , Out =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Number of seconds	0 to 4294967	0	Number of seconds
Output	Out	Conversion result	Follows data type	TIME#0ms	Time

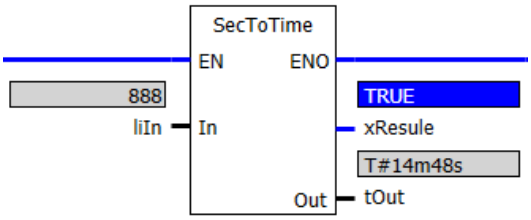
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–

- **Function**

Converts seconds "In" to time.

- **Program example**

Item	LD	ST
Define variables	VAR xResule: BOOL; liIn: LINT := 888; tOut: TIME; END_VAR	
Program		xResule TRUE := SecToTime(In := liIn 888, Out => tOut T#14m48s);RETURN

Item	LD	ST
Operation result		

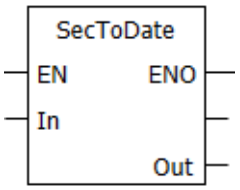
Name	Data type	Address	Value
xResule	BOOL		TRUE
liIn	LINT		888
tOut	TIME		T#14m48s

- **Precautions**
  - To convert a time to the number of seconds, use the 18.2.1.34 TimeToSec instruction.
  - The unit of In is seconds.
  - When the value of In exceeds the valid range, the return value is FALSE and Out is T#0s.

18.2.1.25 SecToDate

Converts the number of seconds since January 1, 1970, 00:00:00 to a date.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SecToDate	Converts seconds to date	FC		:= SecToDate(In := , Out =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Number of seconds	0 to 4294967295	0	Number of seconds since 00:00:00 on January 1, 1970
Output	Out	Conversion result	Follows data type	DATE#1970-1-1	Date

Variable types

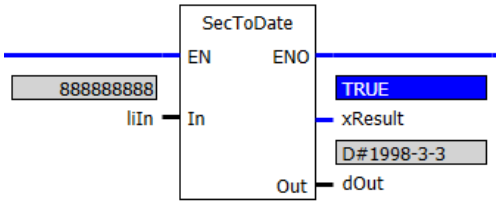
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–

- **Function**

  Converts the number of seconds "In" since January 1, 1970, 00:00:00 to a date. Values less than one day are truncated.

- **Program example**

Item	LD	ST
Define variables	VAR xResult: BOOL; liIn: LINT := 888888888; dOut: DATE; END_VAR	
Program		xResult TRUE := SecToDate(In := liIn 888888888, Out => dOut D#1998-3-3);RETURN
Operation result		

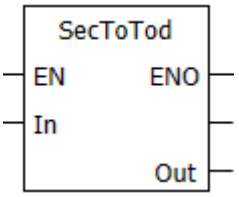
Name	Data type	Address	Value
xResult	BOOL		TRUE
liIn	LINT		888888888
dOut	DATE		D#1998-3-3

- **Precautions**
  - To convert a date to the number of seconds from 00:00:00 on January 1, 1970, use the 18.2.1.13 DateToSec instruction.
  - If the value of In exceeds the valid range, the return value changes to FALSE and the value of Out changes to D#1970-1-1.

18.2.1.26 SecToTod

Converts the number of seconds from 00:00:00 to time of day.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SecToTod	Converts seconds to time of day	FC		:= SecToTod(In := , Out =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Number of seconds	0 to 9223372036854775807	0	Number of seconds since 00:00:00 on January 1, 1970
Output	Out	Conversion result	Follows data type	TIME_OF_DAY#0:0:0	Time of day

  Variable types

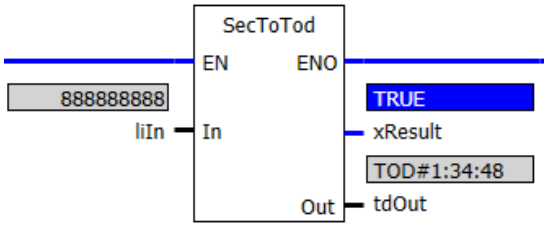
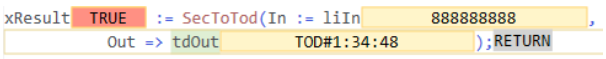
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–

- **Function**

  Converts the number of seconds from 00:00:00 (In) to time of day. When the value of In is greater than 24 hours (86400 seconds), divide the value of In by 24 hours and then convert the remainder into a TOD.

- **Program example**

Item	LD	ST
Define variables	VAR xResult: BOOL; liIn: LINT := 888888888; tdOut: TIME_OF_DAY; END_VAR	

Item	LD	ST
Program		
Operation result		

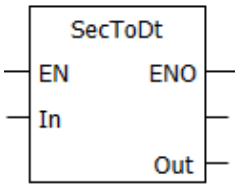
Name	Data type	Address	Value
xResult	BOOL		TRUE
liIn	LINT		888888888
tdOut	TIME_OF_DAY		TOD#1:34:48

- **Precautions**
  - To convert a TOD to the number of seconds from 00:00:00, use the 18.2.1.35 TodToSec instruction.
  - If the value of In exceeds the valid range, the return value changes to FALSE and the value of Out changes to TOD#0:0:0.

18.2.1.27 SecToDt

Converts the number of seconds since January 1, 1970, 00:00:00 to a date and time.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SecToDt	Converts seconds to date and time	FC		:= SecToDt(In := , Out =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Number of seconds	0 to 4294967295	0	Number of seconds since 00:00:00 on January 1, 1970
Output	Out	Conversion result	Follows data type	DATE_AND_TIME#1970-1-1-0:0:0	Date and time

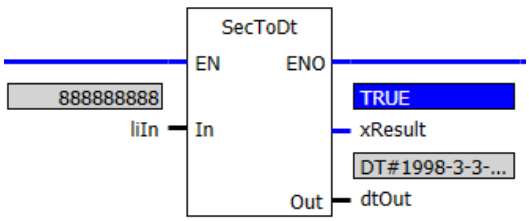
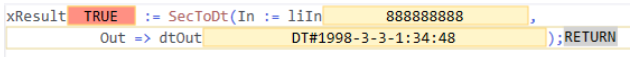
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–

- **Function**

  Converts the number of seconds since January 1, 1970, 00:00:00 (In) to a date and time.

- **Program example**

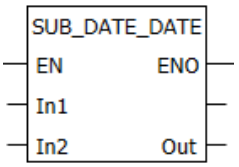
Item	LD	ST
Define variables	VAR xResult: BOOL; liIn: LINT := 888888888; dtOut: DATE_AND_TIME; END_VAR	
Program		
Operation result		

- **Precautions**
  - To convert a DT to the number of seconds from 00:00:00 on January 1, 1970, use the 18.2.1.17 DtToSec instruction.
  - If the value of In exceeds the valid range, the return value changes to FALSE and the value of Out changes to DT#1970-1-1-0:0:0.

18.2.1.28 SUB_DATE_DATE

Subtracts a date from another date.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SUB_DATE_DATE	Subtracts date	FC		:= SUB_DATE_DATE(In1 := , In2 := , Out =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Subtrahend date	DATE#1970-1-1 to DATE#2106-2-7	DATE#1970-1-1	Subtrahend date
	In2	Minuend date	DATE#1970-1-1 to DATE#2106-2-7	DATE#1970-1-1	Minuend date
Output	Out	Subtraction result	Follows data type	-	Subtraction result

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–
In2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–

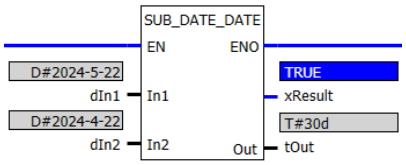
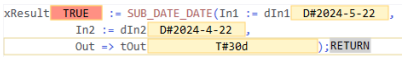
- **Function**

  Subtracts date In2 from date In1. The subtraction result Out is a time.

- **Program example**

Item	LD	ST

Define variables	VAR
	xResult: BOOL;
	dIn1: DATE := D#2024-5-22;
	dIn2: DATE := D#2024-4-22;
	tOut: TIME;
	END_VAR	

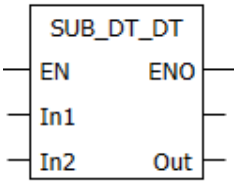
Program		
Operation result		

- **Precautions**
  - When the result of subtraction exceeds the valid range of Out, a program fault occurs, the value of Out is T#0ms, and the return value is FALSE.
  - Since the minimum unit for input DATE is s and for TIME is ms, the result must be less than 4294967 to remain within the valid range of Out.
  - When the value of In1 is less than In2, no program fault occurs and the following operation is executed.
    - In1 is incremented by T#4295030400s, which corresponds to the maximum value of DATE (DATE#2106-2-7) plus one additional day.
    - For example: DATE#1970-1-1 - DATE#2106-2-7 = (DATE#1970-1-1 + T#4295030400s) - DATE#2106-2-7 = T#1d

18.2.1.29 SUB_DT_DT

Subtracts date and time from date and time.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SUB_DT_DT	Subtracts date and time	FC		:= SUB_DT_DT(In1 := , In2 := , Out =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Subtrahend date and time	Follows data type	DATE_AND_TIME#1970-1-1-0:0:0	Subtrahend date and time
	In2	Minuend date and time	Follows data type	DATE_AND_TIME#1970-1-1-0:0:0	Minuend date and time
Output	Out	Subtraction result	Follows data type	-	Subtraction result

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–
In2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–

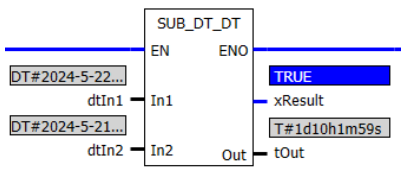
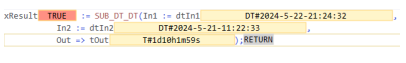
- **Function**

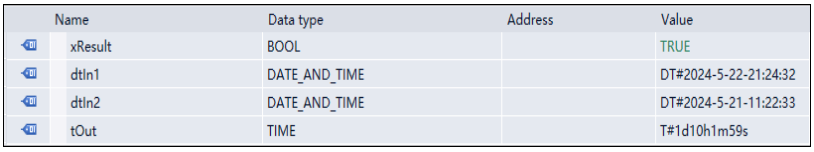
  Subtracts date and time In2 from date and time In1. The subtraction result is a time.

- **Program example**

Item	LD	ST

Define variables	VAR
	xResult: BOOL;
	dtIn1: DATE_AND_TIME := DT#2024-5-22-21:24:32;
	dtIn2: DATE_AND_TIME := DT#2024-5-21-11:22:33;
	tOut: TIME;
	END_VAR	

Program		

Item	LD	ST
Operation result		

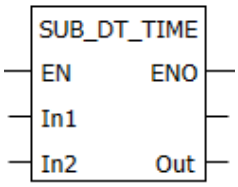
Name	Data type	Address	Value
xResult	BOOL		TRUE
dtIn1	DATE_AND_TIME		DT#2024-5-22-21:24:32
dtIn2	DATE_AND_TIME		DT#2024-5-21-11:22:33
tOut	TIME		T#1d10h1m59s

- **Precautions**
  - When the result of subtraction exceeds the valid range of Out, a program fault occurs, the value of Out is T#0ms, and the return value is FALSE.
  - Since the minimum unit for input DATE is s and for TIME is ms, the result must be less than 4294967 to remain within the valid range of Out.
  - When the value of In1 is less than In2, no program fault occurs and the following operation is executed.
    - In1 is incremented by T#4294967296s, which corresponds to the maximum value of DT.
    - For example: DT#1970-1-1-0:0:0 - DT#2106-2-6-6:28:15 = (DT#1970-1-1-0:0:0 + T#4294967296s) - DT#2106-2-6-6:28:15 = T#1d1s

18.2.1.30 SUB_DT_TIME

Subtracts a time from date and time.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SUB_DT_TIME	Subtracts time from date and time	FC		:= SUB_DT_TIME(In1 := , In2 := , Out =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Subtrahend date and time	Follows data type	DATE_AND_TIME#1970-1-1-0:0:0	Subtrahend date and time
	In2	Minuend time	Follows data type	TIME#0ms	Minuend time
Output	Out	Subtraction result	Follows data type	-	Subtraction result

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–
In2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–

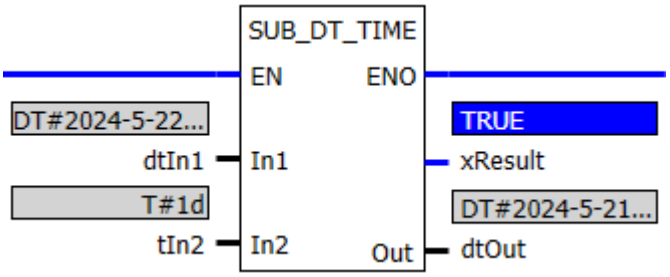
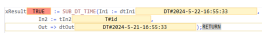
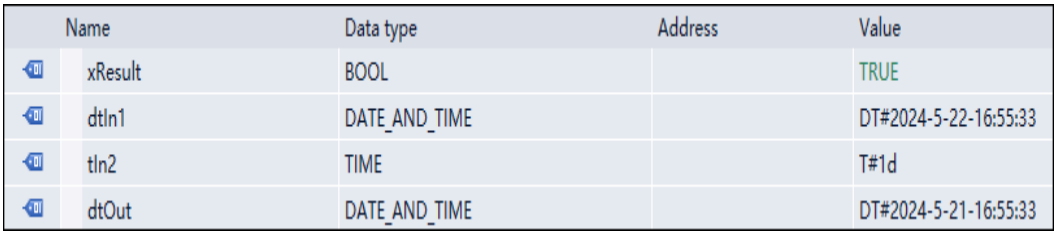
- **Function**

  Subtracts time In2 from date and time In1. The subtraction result Out is a date and time.

- **Program example**

Item	LD	ST

Define variables	VAR
	xResult: BOOL;
	dtIn1: DATE_AND_TIME := DT#2024-5-22-16:55:33;
	tIn2: TIME := T#1d;
	dtOut: DATE_AND_TIME;
	END_VAR	

Program		
Operation result		

Name	Data type	Address	Value
xResult	BOOL		TRUE
dtIn1	DATE_AND_TIME		DT#2024-5-22-16:55:33
tIn2	TIME		T#1d
dtOut	DATE_AND_TIME		DT#2024-5-21-16:55:33

- **Precautions**

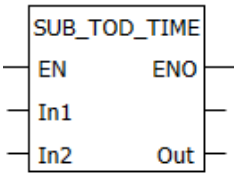
  When the result of subtraction exceeds the valid range of Out, no program fault occurs and the following operation is executed.

  - In1 is incremented by T#4294967296s, which corresponds to the maximum value of DT.
  - For example: DT#1970-1-1-0:0:0 - T#1d = (DT#1970-1-1-0:0:0 + T#4294967296s) - T#1d = DT#2106-2-6-6:28:15

18.2.1.31 SUB_TOD_TIME

Subtracts a time from time of day.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SUB_TOD_TIME	Subtracts time from time of day	FC		:= SUB_TOD_TIME(In1 := , In2 := , Out =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Subtrahend TOD	Follows data type	TIME_OF_DAY#0:0:0	Subtrahend TOD
	In2	Minuend time	Follows data type	TIME#0ms	Minuend time
Output	Out	Subtraction result	Follows data type	-	Subtraction result

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–
In2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–

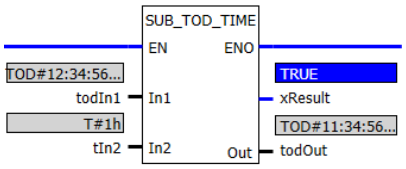
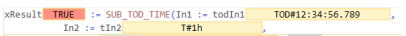
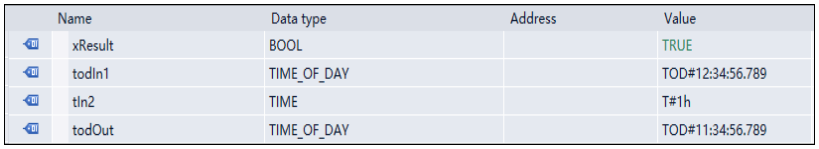
- **Function**

Subtracts time In2 from time of day In1. The subtraction result Out is a TOD.

- **Program example**

Item	LD	ST

Define variables	VAR
	xResult: BOOL;
	todIn1: TIME_OF_DAY := TOD#12:34:56.789;
	tIn2: TIME := T#1h;
	todOut: TIME_OF_DAY;
	END_VAR	

Program		
Operation result		

- **Precautions**

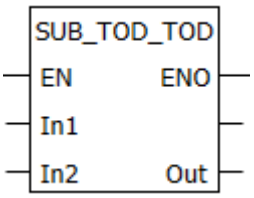
  When the result of subtraction exceeds the valid range of Out, no program fault occurs and the following operation is executed.

  - In1 is incremented by T#86400000S(24h).
  - For example: TOD#0:0:0 - T#1ms = (TOD#0:0:0 + T#86400000S) - T#1ms = TOD#23:59:59:999

18.2.1.32 SUB_TOD_TOD

Subtracts time of day from time of day.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SUB_TOD_TOD	Subtracts time of day	FC		:= SUB_TOD_TOD(In1 := , In2 := , Out =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1	Subtrahend TOD	Follows data type	TIME_OF_DAY#0:0:0	Subtrahend TOD
	In2	Minuend TOD	Follows data type	TIME_OF_DAY#0:0:0	Minuend TOD
Output	Out	Subtraction result	Follows data type	-	Subtraction result

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–
In2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–

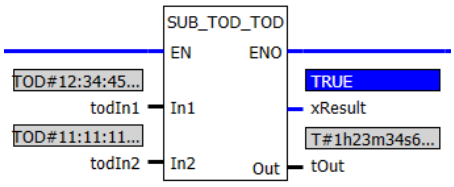
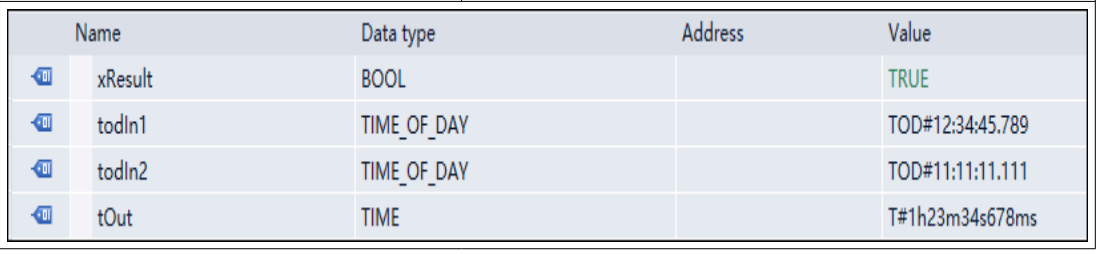
- **Function**

  Subtracts TOD In2 from TOD In1. The subtraction result Out is a time.

- **Program example**

Item	LD	ST

Define variables	VAR
	xResult: BOOL;
	todIn1: TIME_OF_DAY := TOD#12:34:45.789;
	todIn2: TIME_OF_DAY := TOD#11:11:11.111;
	tOut: TIME;
	END_VAR	

Program		xResult TRUE := SUB_TOD_TOD(In1 := todIn1 TOD#12:34:45.789 , In2 := todIn2 TOD#11:11:11.111 , Out => tOut T#1h23m34s678ms);RETURN
Operation result		

Name	Data type	Address	Value
xResult	BOOL		TRUE
todIn1	TIME_OF_DAY		TOD#12:34:45.789
todIn2	TIME_OF_DAY		TOD#11:11:11.111
tOut	TIME		T#1h23m34s678ms

- **Precautions**

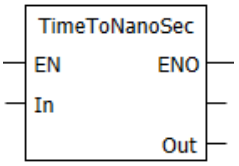
  When the value of In1 is less than In2, no program fault occurs and the following operation is executed.

  - In1 is incremented by T# 86400000S(24h).
  - For example: TOD#13:34:56.789 - TOD#13:35:56.789 = (TOD#13:34:56.789 + T# 86400000S) - TOD#13:35:56.789 = T#23h59m

18.2.1.33 TimeToNanoSec

Converts time to the number of nanoseconds.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
TimeToNanoSec	Converts time to nanoseconds	FC		:= TimeToNanoSec(In := , Out =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Time	Follows data type	TIME#0ms	Time
Output	Out	Conversion result	Follows data type	0	Number of nanoseconds

Variable types

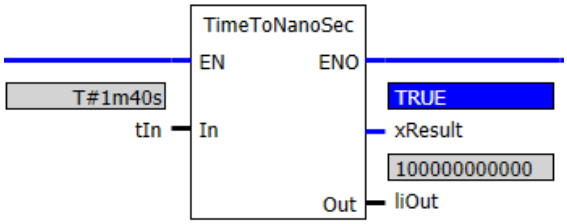
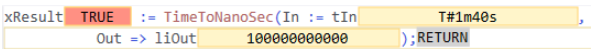
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–

- **Function**

Converts time In to number of nanoseconds.

- **Program example**

Item	LD	ST
Define variables	VAR xResult: BOOL; tIn: TIME := T#1m40s; liOut: LINT; END_VAR	

Item	LD	ST
Program		
Operation result		

	Name	Data type	Address	Value
	xResult	BOOL		TRUE
	tIn	TIME		T#1m40s
	liOut	LINT		100000000000

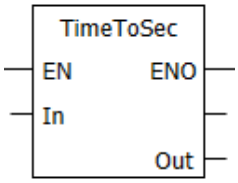
- **Precautions**

  To convert the number of nanoseconds to a time, use the 18.2.1.23 NanoSecToTime instruction.

18.2.1.34 TimeToSec

Converts a time to the number of seconds.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
TimeToSec	Converts time to seconds	FC		:= TimeToSec(In := , Out =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Time	Follows data type	TIME#0ms	Time
Output	Out	Conversion result	Follows data type	0	Number of seconds

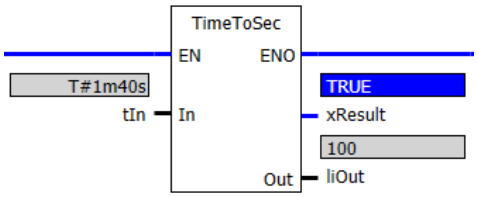
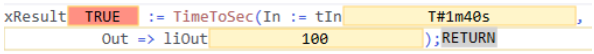
  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–

- **Function**

  Converts time In to number of seconds. Values less than one second will be truncated.

- **Program example**

Item	LD	ST
Define variables	VAR xResult: BOOL; tIn: TIME := T#1m40s; liOut: LINT; END_VAR	
Program		
Operation result		

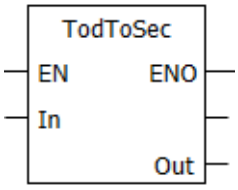
Name	Data type	Address	Value
xResult	BOOL		TRUE
tIn	TIME		T#1m40s
liOut	LINT		100

- **Precautions**
  - To convert the number of seconds to a time, use the 18.2.1.24 SecToTime instruction.
  - The unit of Out is seconds.

18.2.1.35 TodToSec

Converts time of day to the number of seconds from 00:00:00.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
TodToSec	Converts time to seconds	FC		:= TodToSec(In := , Out =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Time of day	Follows data type	TIME_OF_DAY#0:0:0	Time of day
Output	Out	Conversion result	0 to 86399	0	Number of seconds since 00:00:00

  Variable types

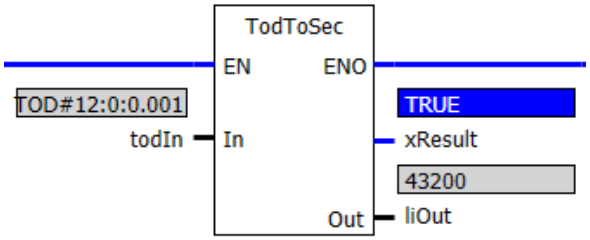
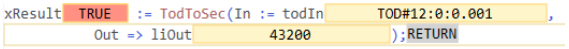
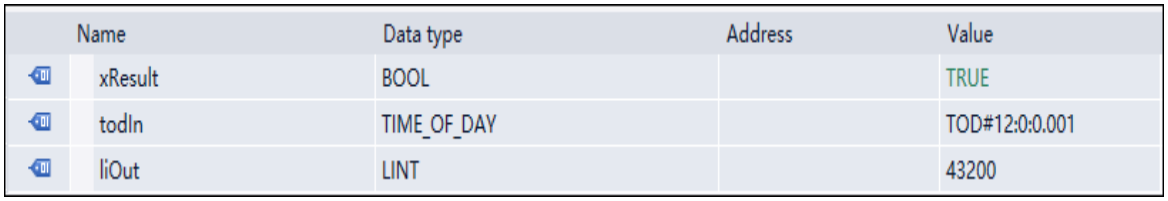
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–

- **Function**

  Converts time of day In to the number of seconds from 00:00:00. The converted value is in seconds. Values less than one second will be truncated.

- **Program example**

Item	LD	ST
Define variables	VAR xResult: BOOL; todIn: TIME_OF_DAY := TOD#12:00:00.001; liOut: LINT; END_VAR	

Item	LD	ST
Program		
Operation result		

	Name	Data type	Address	Value
	xResult	BOOL		TRUE
	todIn	TIME_OF_DAY		TOD#12:0:0.001
	liOut	LINT		43200

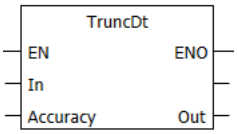
- **Precautions**

  To convert the number of seconds from 00:00:00 to a TOD, use the 18.2.1.26 SecToTod instruction.

18.2.1.36 TruncDt

Truncates all digits below the specified unit in a DT variable.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
TruncDt	Truncates date and time	FC		:= TruncDt(In := , Accuracy := , Out =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	DT to truncate	Follows data type	DATE_AND_TIME#1970-1-1-0:0:0	DT to truncate
	Accuracy	Smallest time unit to leave after truncation	Follows data type (_eSUBSEC)	_SEC	Smallest time unit to leave after truncation
Output	Out	DT after truncation	Follows data type	-	DT after truncation

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–
Accuracy	See "Function description" for the enumerators of enumeration type _eSUBSEC.																			
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–

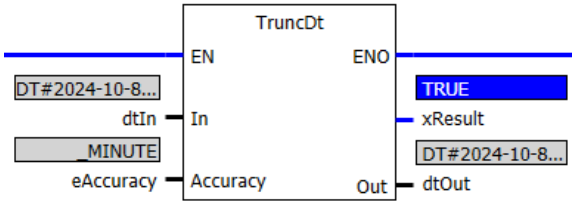
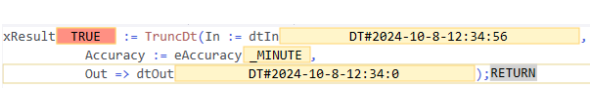
- **Function**
  - This instruction truncates all digits below the smallest unit specified by Accuracy from the date and time In. The result after truncation is stored in Out.
  - Accuracy is of enumeration type _eSUBSEC, and the meanings of enumeration values are as follows in the table.
  - _MILLISEC cannot be used for enumerators; otherwise, a program fault occurs and the return value is FALSE.

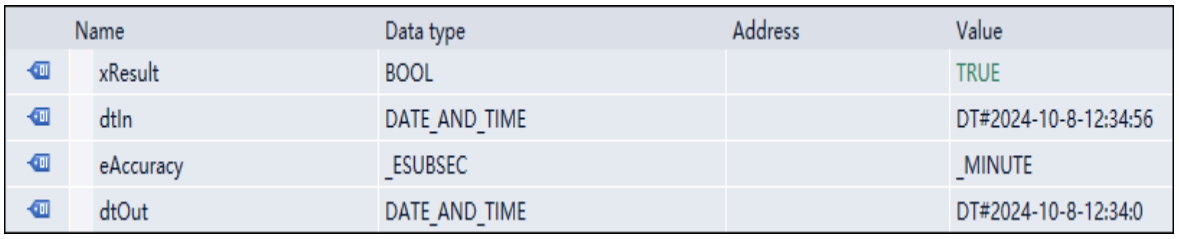
Enumeration Value	Definition
_MILLISEC	Millisecond
_SEC	Second
_MINUTE	Minute
_HOUR	Hour

- **Program example**

Item	LD	ST

Define variables	VAR
	xResult: BOOL;
	dtIn: DATE_AND_TIME := DT#2024-10-08-12:34:56;
	eAccuracy:_ESUBSEC := _ESUBSEC._MINUTE;
	dtOut: DATE_AND_TIME;
	END_VAR	

Program		

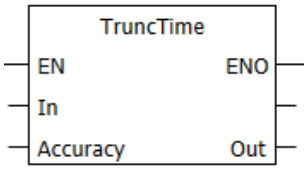
Item	LD	ST
Operation result		

Name	Data type	Address	Value
xResult	BOOL		TRUE
dtIn	DATE_AND_TIME		DT#2024-10-8-12:34:56
eAccuracy	_ESUBSEC		_MINUTE
dtOut	DATE_AND_TIME		DT#2024-10-8-12:34:0

18.2.1.37 TruncTime

Truncates all digits below the specified unit in a TIME variable.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
TruncTime	Truncates time	FC		:= TruncTime(In := , Accuracy := , Out =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Time to truncate	Follows data type	TIME#0ms	Time to truncate
	Accuracy	Smallest time unit to leave after truncation	Follows data type (_eSUBSEC)	_SEC	Smallest time unit to leave after truncation
Output	Out	Time after truncation	Follows data type	-	Time after truncation

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Accuracy	See "Function description" for the enumerators of enumeration type _eSUBSEC.																			

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–

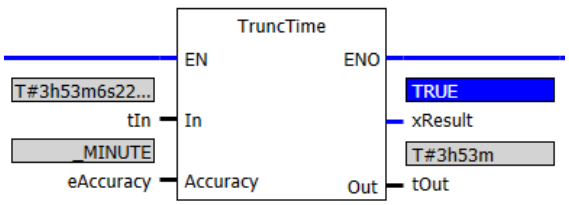
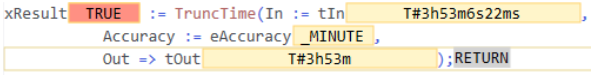
- **Function**
  - This instruction truncates all digits below the smallest unit specified by Accuracy from the time In. The result after truncation is stored in Out.
  - Accuracy is of enumeration type _eSUBSEC, and the meanings of enumeration values are as follows in the table.

Enumeration Value	Definition
_MILLISEC	Millisecond
_SEC	Second
_MINUTE	Minute
_HOUR	Hour

- **Program example**

Item	LD	ST

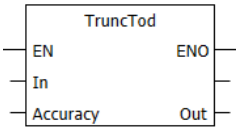
Define variables	VAR
	xResult: BOOL;
	tIn: TIME := T#3h53m6s22ms;
	eAccuracy: _ESUBSEC := _ESUBSEC._MINUTE;
	tOut: TIME;
	END_VAR	

Program		
Operation result		

18.2.1.38 TruncTod

Truncates all digits below the specified unit in a TOD variable.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
TruncTod	Truncates time of day	FC		:= TruncTod(In := , Accuracy := , Out =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	TOD to truncate	Follows data type	TIME_OF_DAY#0:0:0	TOD to truncate
	Accuracy	Smallest time unit to leave after truncation	Follows data type (_eSUBSEC)	_SEC	Smallest time unit to leave after truncation
Output	Out	TOD after truncation	Follows data type	-	TOD after truncation

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–
Accuracy	See "Function description" for the enumerators of enumeration type _eSUBSEC.																			
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–

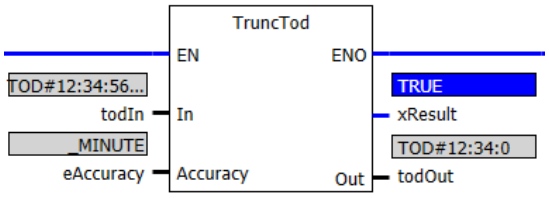
- **Function**
  - This instruction truncates all digits below the smallest unit specified by Accuracy from the time of day In. The result after truncation is stored in Out.
  - Accuracy is of enumeration type _eSUBSEC, and the meanings of enumeration values are as follows in the table.

Enumeration Value	Definition
_MILLISEC	Millisecond
_SEC	Second
_MINUTE	Minute
_HOUR	Hour

- **Program example**

Item	LD	ST

Define variables	VAR
	xResult: BOOL;
	todIn: TIME_OF_DAY := TOD#12:34:56.789;
	eAccuracy: _ESUBSEC := _ESUBSEC._MINUTE;
	todOut: TIME_OF_DAY;
	END_VAR	

Program		xResult TRUE := TruncTod(In := todIn TOD#12:34:56.789, Accuracy := eAccuracy _MINUTE, Out => todOut TOD#12:34:0);RETURN
Operation result	Name / Data type / Address / Value xResult / BOOL / / TRUE todIn / TIME_OF_DAY / / TOD#12:34:56.789 eAccuracy / _ESUBSEC / / _MINUTE todOut / TIME_OF_DAY / / TOD#12:34:0	

18.2.2 Data Calculation

18.2.2.1 Instruction List

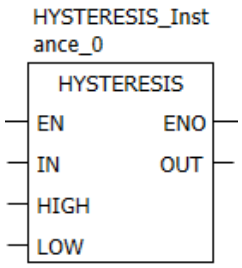
Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.2.2 HYSTERESIS	FB	Limit determination	Util	Library version 1.0.0.0 and later
18.2.2.3 LIMITALARM	FB	Upper and lower limit monitoring		Library version 1.0.0.0 and later
18.2.2.4 DERIVATIVE	FB	Derivative value calculation		Library version 1.0.0.0 and later
18.2.2.5 INTEGRAL	FB	Integral value calculation		Library version 1.0.0.0 and later
18.2.2.6 LIN_TRAFO	FB	Linear scaling		Library version 1.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.2.7 STATISTICS_INT	FB	Data statistics and analysis		Library version 1.0.0.0 and later
18.2.2.8 STATISTICS_REAL	FB	Data statistics and analysis		Library version 1.0.0.0 and later
18.2.2.9 VARIANCE	FB	Variance calculation		Library version 1.0.0.0 and later

18.2.2.2 HYSTERESIS

HYSTERESIS is used for limit determination.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
HYSTERESIS	Limit determination	FB		HYSTERESIS_Instance_0(IN := , HIGH := , LOW := , OUT =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input data	Follows data type	0	Input data
	HIGH	Upper limit	Follows data type	0	Upper limit
	LOW	Lower limit	Follows data type	0	Lower limit
Output	OUT	Output data	Follows data type	FALSE	Output data.

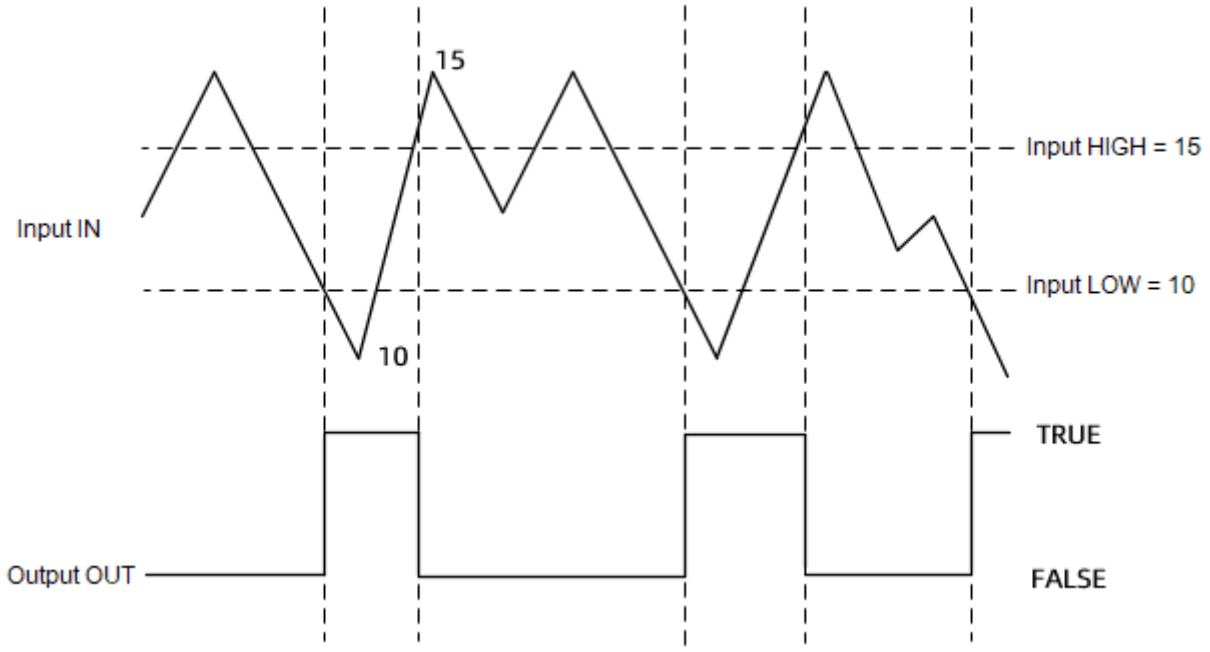
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
HIGH	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
LOW	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
OUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

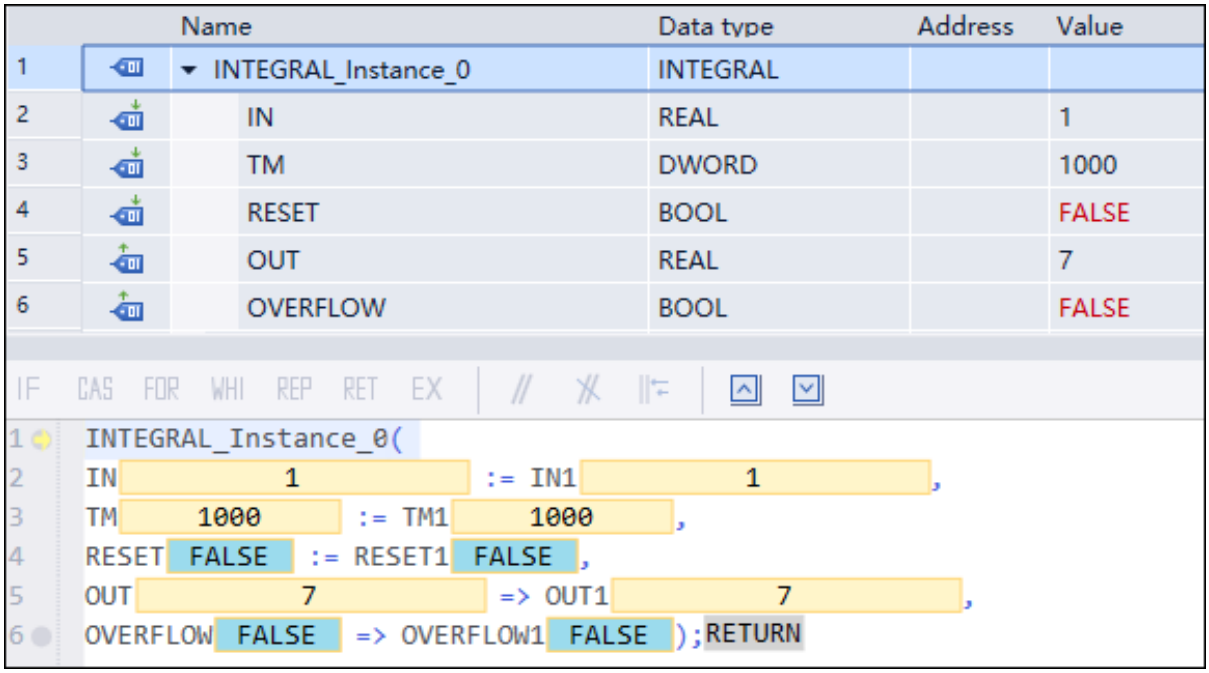
  If input "IN" is less than the lower limit "LOW", "OUT" is TRUE, and remains TRUE until "IN" is greater than the upper limit "HIGH". At this time, "OUT" changes to FALSE, and remains FALSE until "IN" is less than "LOW". OUT changes from FALSE to TRUE and remains until "IN" exceeds the upper limit HIGH, at which point "OUT" changes to FALSE, repeating this cycle.

- **Timing diagram**



- **Program example**

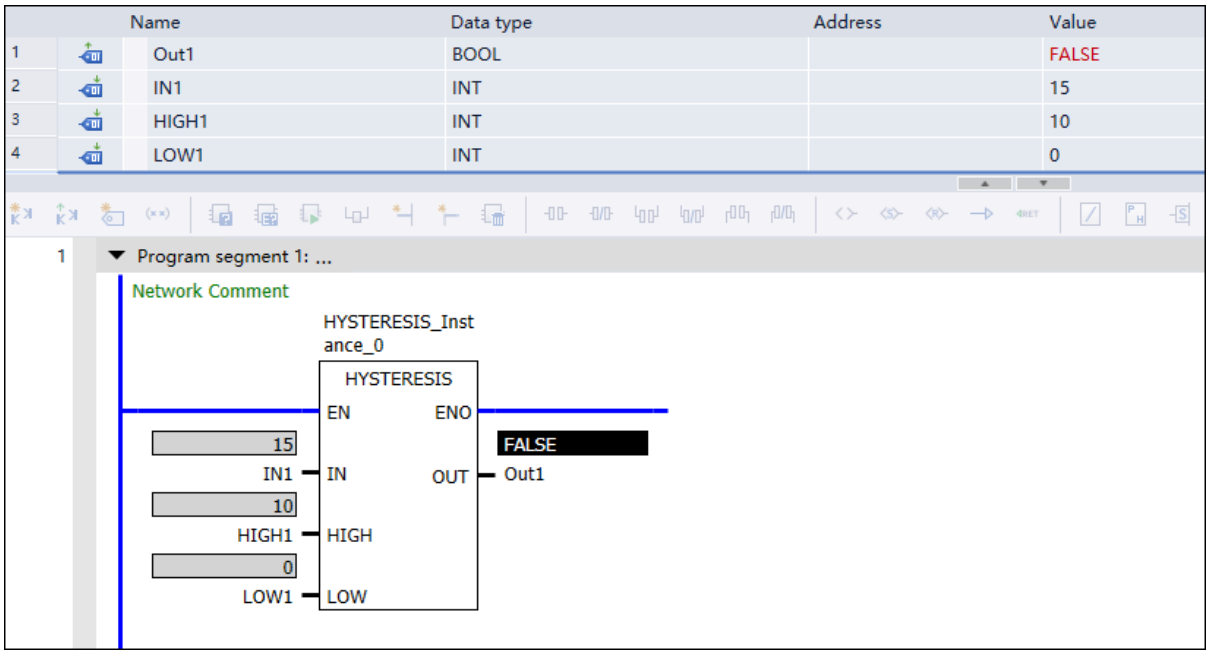
  ST:



	Name	Data type	Address	Value
1	INTEGRAL_Instance_0	INTEGRAL		
2	IN	REAL		1
3	TM	DWORD		1000
4	RESET	BOOL		FALSE
5	OUT	REAL		7
6	OVERFLOW	BOOL		FALSE

```
INTEGRAL_Instance_0(
IN := IN1,
TM := TM1,
RESET := RESET1,
OUT => OUT1,
OVERFLOW => OVERFLOW1);RETURN
```

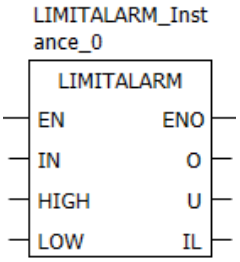
LD:



18.2.2.3 LIMITALARM

Monitors whether the input value is within the upper and lower limits.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LIMITALARM	Upper and lower limit monitoring	FB		LIMITALARM_Instance_0(IN := , HIGH := , LOW := , O => , U => , IL =>);

- **Related variable**

Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input data	Follows data type	0	Input data
	HIGH	Upper limit	Follows data type	0	Upper limit
	LOW	Lower limit	Follows data type	0	Lower limit
Output	O	Output data 1	Follows data type	-	Output data 1
	U	Output data 2	Follows data type	-	Output data 2
	IL	Output data 3	Follows data type	-	Output data 3

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
HIGH	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
LOW	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
O	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
U	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IL	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  When input "IN" is below the lower limit "LOW", output "U" is TRUE; when "IN" is above the upper limit "HIGH", output "O" is TRUE. When input is within the upper and lower limits, output IL is TRUE.

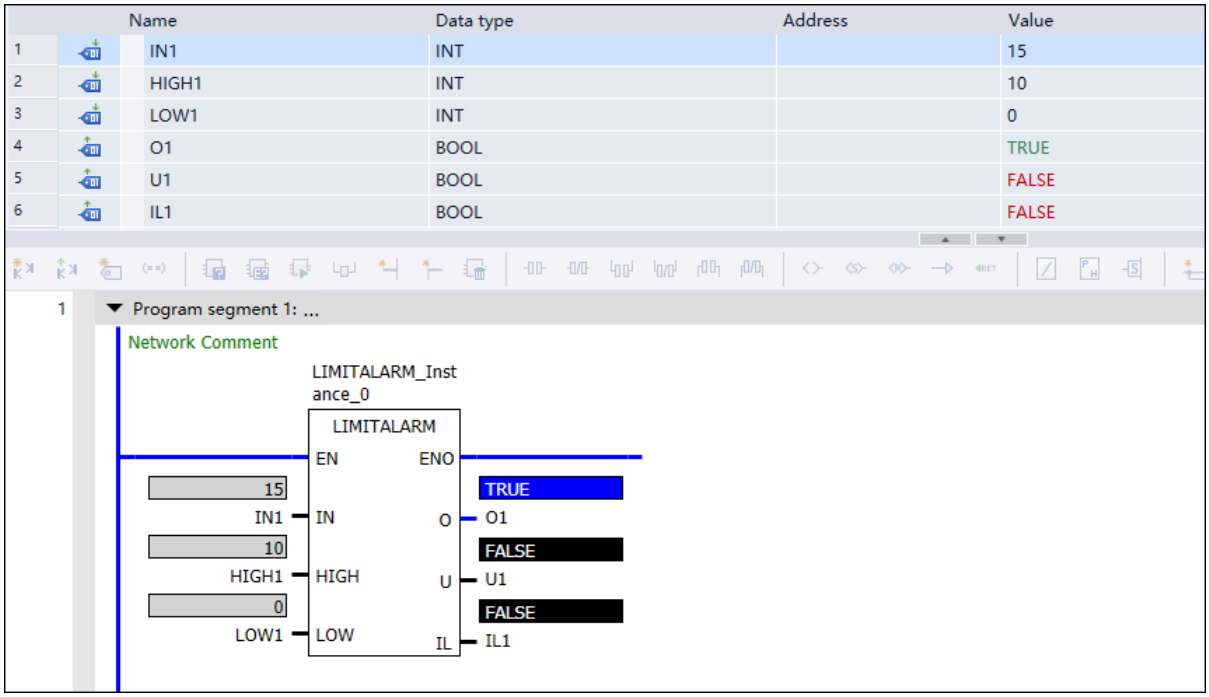
- **Program example**

  ST:

		Name	Data type	Address	Value
1		IN1	INT		15
2		HIGH1	INT		10
3		LOW1	INT		0
4		O1	BOOL		TRUE
5		U1	BOOL		FALSE
6		IL1	BOOL		FALSE

```
LIMITALARM_Instance_0(
IN 15 := IN1 15,
HIGH 10 := HIGH1 10,
LOW 0 := LOW1 0,
O TRUE => O1 TRUE,
U FALSE => U1 FALSE,
IL FALSE => IL1 FALSE );RETURN
```

  LD:



18.2.2.4 DERIVATIVE

DERIVATIVE is used for derivative calculation.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DERIVATIVE	Derivative value calculation	FB	DERIVATIVE_Instance_0 DERIVATIVE EN ENO IN OUT TM RESET	DERIVATIVE_Instance_0(IN := , TM := , RESET := , OUT =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input data	Follows data type	-	Input data
	TM	Derivative time	Follows data type	-	Derivative time (unit: ms)
	RESET	Reset	Follows data type	-	Reset
Output	OUT	Derivative value	Follows data type	-	Derivative value

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
TM	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

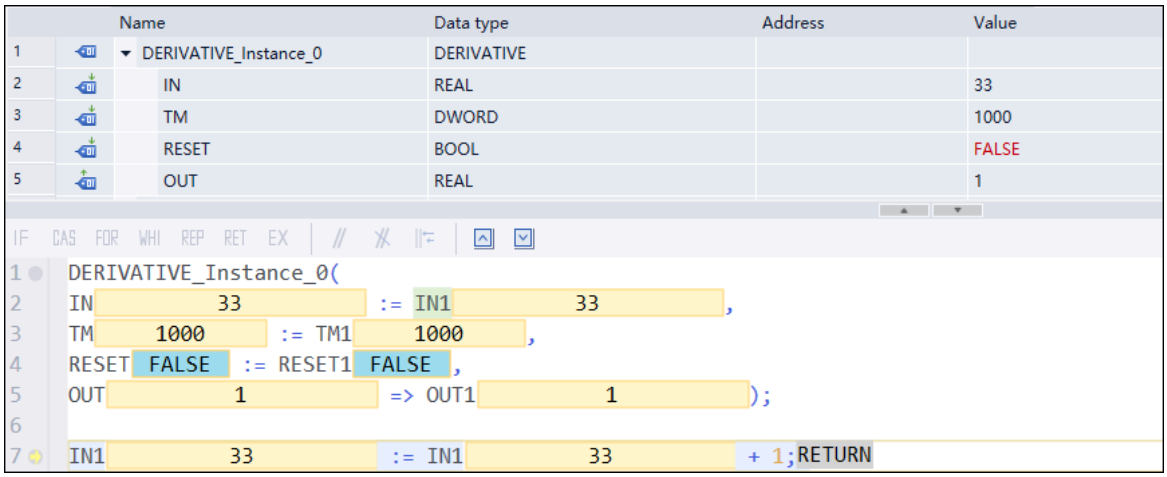
- **Function**

  Calculates the derivative of input "IN" within TM time and outputs "OUT". When "RESET" is TRUE, historical data is cleared and "OUT" is 0. It is recommended to set TM to the task scheduling cycle.

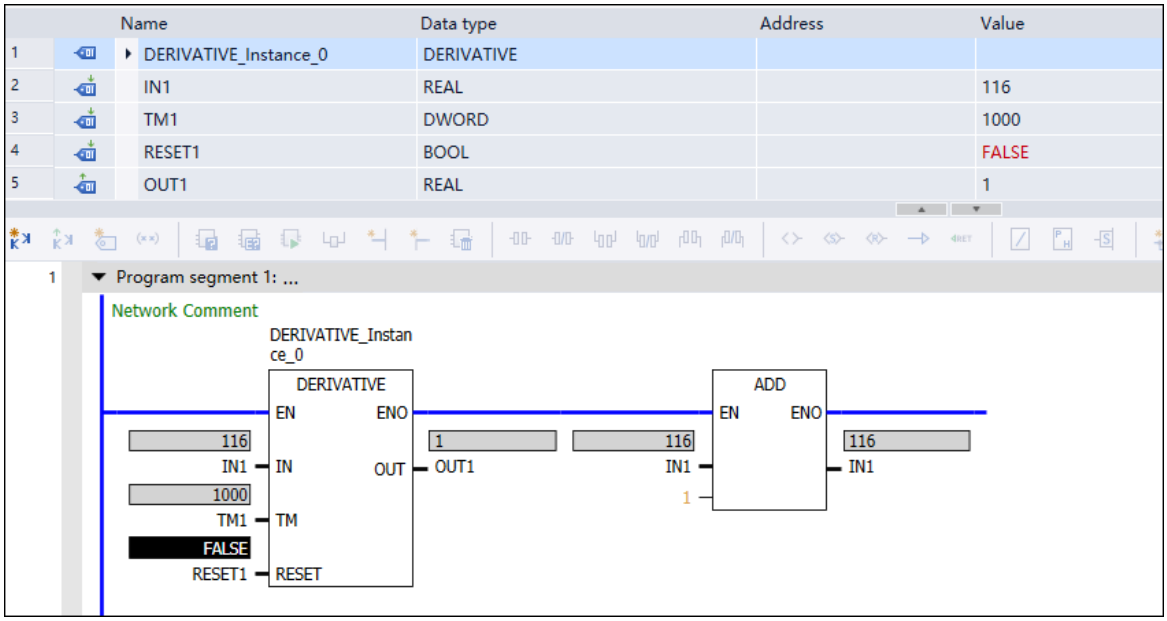
- **Program example**

  The following example sets the task scheduling cycle to 1000 ms. Each time the task executes, the input "IN" increments by 1. As a result, the derivative (rate of change) of "IN" is 1.

  ST:



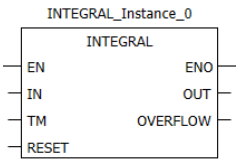
  LD:



18.2.2.5 INTEGRAL

INTEGRAL is used for integral calculation.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
INTEGRAL	Integral value calculation	FB		INTEGRAL_Instance_0(IN := , TM := , RESET := , OUT => , OVERFLOW => ;

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input data	Follows data type	-	Input data
	TM	Integral time	Follows data type	-	Integral time (unit: ms)
	RESET	Reset	Follows data type	-	Reset
Output	OUT	Integral value	Follows data type	-	Integral value
	OVERFLOW	Overflow flag	Follows data type	FALSE	Checks whether the output data exceeds the upper limit of the REAL type.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
TM	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
OVERFLOW	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Calculates the integral value of input "IN" within TM time and outputs "OUT". When "RESET" is TRUE, the historical data is cleared and "OUT" is 0.

  It is recommended to set "TM" to the task scheduling cycle. When the value of "OUT" exceeds the upper limit of the REAL type, "OVERFLOW" is set to TRUE.

- **Program example**

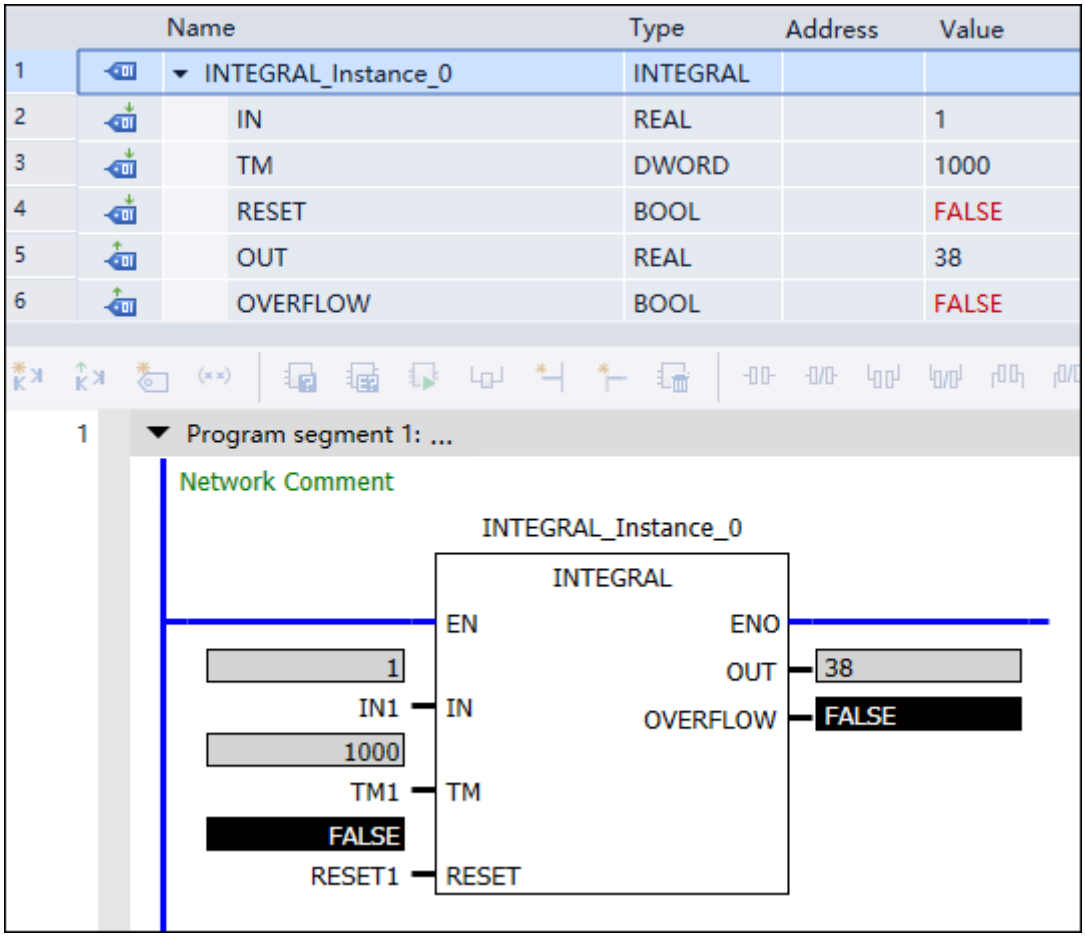
  The following example sets the task scheduling cycle to 1000 ms. Each time the task executes, the input "IN" increments by 1 and the integral value of "IN" increases over time. After 7 cycles, the output is 7. After 38 cycles, the output is 38.

  ST:

	Name	Type	Address	Value
1	INTEGRAL_Instance_0	INTEGRAL		
2	IN	REAL		1
3	TM	DWORD		1000
4	RESET	BOOL		FALSE
5	OUT	REAL		7
6	OVERFLOW	BOOL		FALSE

```
INTEGRAL_Instance_0(
IN := 1 := IN1 1,
TM 1000 := TM1 1000,
RESET FALSE := RESET1 FALSE,
OUT 7 => OUT1 7,
OVERFLOW FALSE => OVERFLOW1 FALSE );RETURN
```

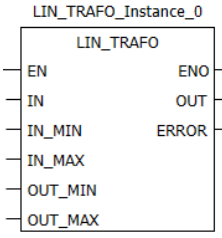
  LD:



18.2.2.6 LIN_TRAFO

LIN_TRAFO is used for linear scaling.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LIN_TRAFO	Linear scaling	FB		LIN_TRAFO_Instance_0(IN := , IN_MIN := , IN_MAX := , OUT_MIN := , OUT_MAX := , OUT => , ERROR =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input data	Follows data type	-	Input data
	IN_MIN	Input lower limit	Follows data type	-	Reset
	IN_MAX	Input upper limit	Follows data type	-	Input upper limit
	OUT_MIN	Output lower limit	Follows data type	-	Output lower limit
	OUT_MAX	Output upper limit	Follows data type	-	Output upper limit
Output	OUT	Output value	Follows data type	-	Output value
	ERROR	Fault flag	Follows data type	-	Fault flag

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
IN_MIN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
IN_MAX	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
OUT_MIN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
OUT_MAX	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
ERROR	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

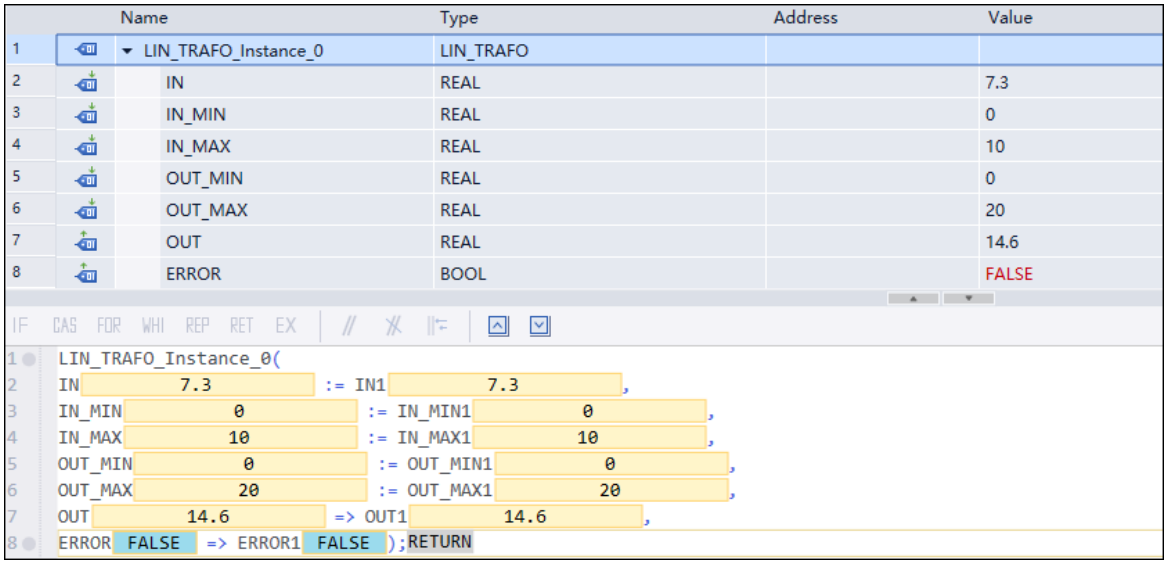
- **Function**

Performs linear scaling on input data based on the following rule:

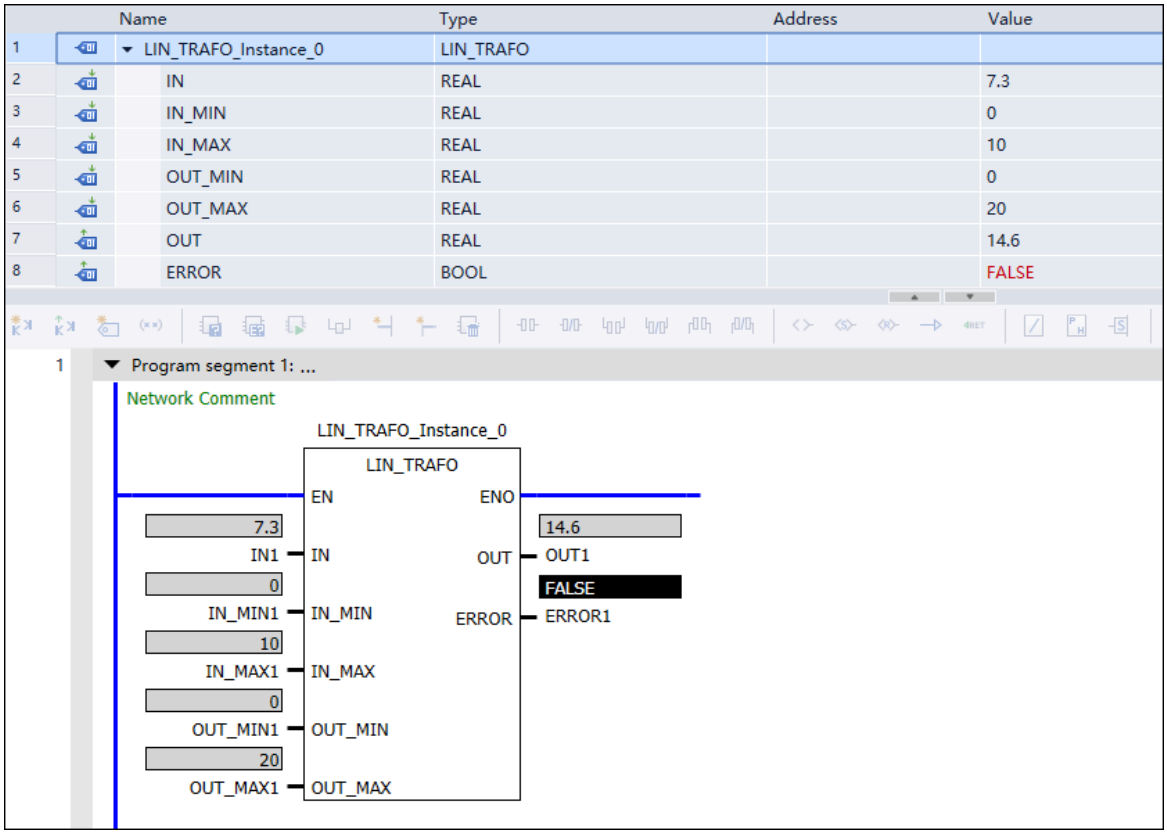
$$OUT = \frac{IN - IN_MIN}{IN_MAX - IN_MIN} * (OUT_MAX - OUT_MIN) + OUT_MIN$$

- **Program example**

  ST:



  LD:



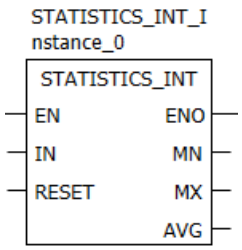
- **Precautions**

  When the input upper limit and input lower limit are equal, or when the input exceeds the limit range, ERROR is set to TRUE.

18.2.2.7 STATISTICS_INT

STATISTICS_INT is used for integer data statistics and analysis.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
STATISTICS_INT	Data statistics and analysis	FB		STATISTICS_INT_Instance_0(IN := , RESET := , MN => , MX => , AVG =>);

- **Related variable**

Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input data	Follows data type	-	Input data
	RESET	Reset a bit	Follows data type	-	When set to TRUE, AVG is set to 0, and "MN" and "MX" are set to the minimum and maximum values for that data type, respectively.
Output	MN	Minimum input value	Follows data type	-32768	Minimum input value
	MX	Maximum input value	Follows data type	32767	Maximum input value
	AVG	Average input value	Follows data type	-	Average input value

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
MN	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
MX	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
AVG	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

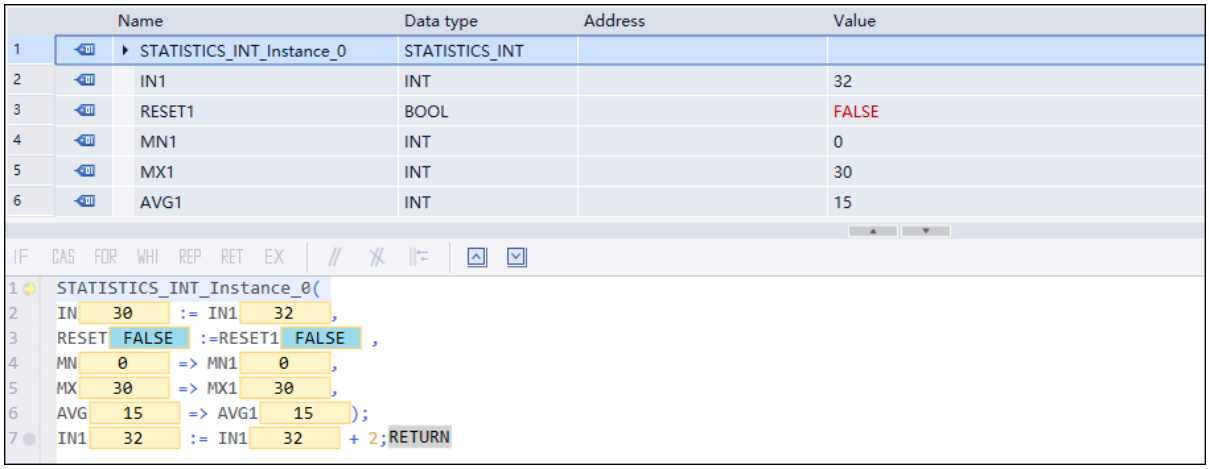
- **Function**

  Statically calculates the maximum, minimum, and average values of input data "IN". When "RESET" is set to 1, "AVG" is initialized to input "IN".

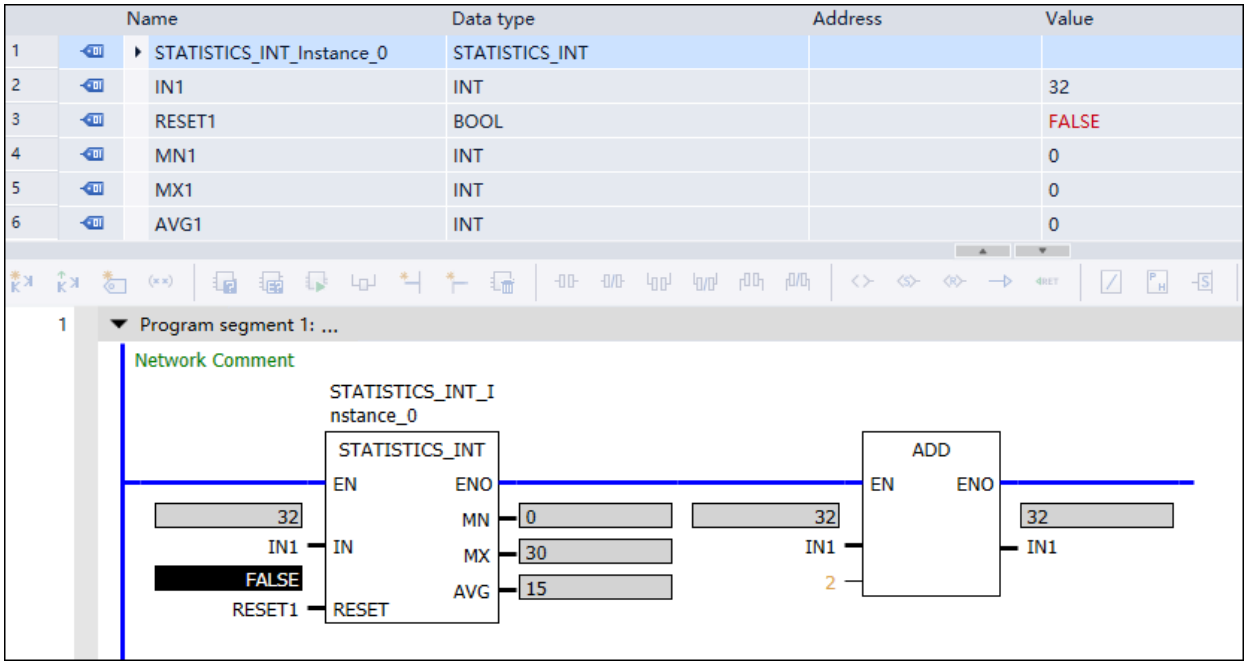
- **Program example**

  The following example sets "RESET" to 0, and "IN" to 0. "IN" increments by 2 per task scheduling cycle. The simulation results after 15 cycles are shown in the following figure.

  ST:



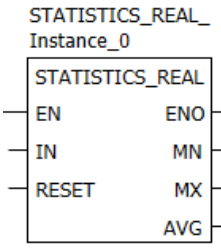
  LD:



18.2.2.8 STATISTICS_REAL

STATISTICS_INT is used for floating-point data statistics and analysis.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
STATISTICS_REAL	Data statistics and analysis	FB		STATISTICS_REAL_Instance_0(IN := , RESET := , MN => , MX => , AVG =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input data	Follows data type	-	Input data
	RESET	Reset a bit	Follows data type	-	When set to TRUE, AVG is set to 0, and "MN" and "MX" are set to the minimum and maximum values for that data type, respectively.
Output	MN	Minimum input value	Follows data type	-3.402823466E+38	Minimum input value

Input/Output	Variable	Name	Valid Range	Default Value	Description
	MX	Maximum input value	Follows data type	3.402823466E+38	Maximum input value
	AVG	Average input value	Follows data type	-	Average input value

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
MN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
MX	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
AVG	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

- **Function**

  Statically calculates the maximum, minimum, and average values of input data "IN". When "RESET" is set to 1, "AVG" is initialized to 0.

- **Program example**

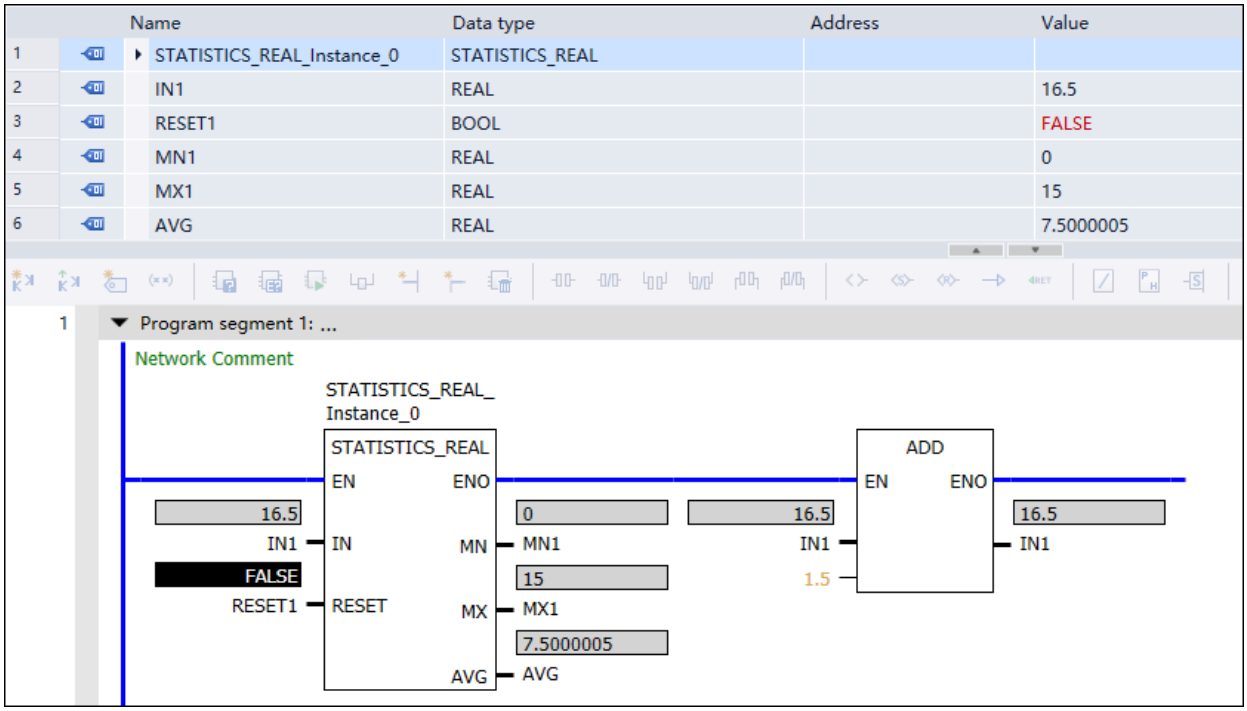
  The following example sets "RESET" to 0, and "IN" to 0. "IN" increments by 1.5 per task scheduling cycle. The simulation results after 10 cycles are shown in the following figure.

  ST:

	Name	Data type	Address	Value
1	STATISTICS_REAL_Instance_0	STATISTICS_REAL		
2	IN1	REAL		16.5
3	RESET1	BOOL		FALSE
4	MN1	REAL		0
5	MX1	REAL		15
6	AVG	REAL		7.5000005

```
STATISTICS_REAL_Instance_0(
IN := 15 := IN1 16.5,
RESET FALSE := RESET1 FALSE,
MN => MN1 0,
MX => MX1 15,
AVG 7.5000005 => AVG 7.5000005);
IN1 := IN1 + 1.5;RETURN
```

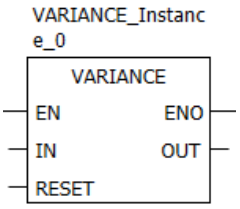
  LD:



18.2.2.9 VARIANCE

VARIANCE is used for variance calculation.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
VARIANCE	Variance calculation	FB		VARIANCE_Instance_0(IN := , RESET := , OUT =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input data	Follows data type	-	Input data
	RESET	Reset a bit	Follows data type	-	When set to TRUE, "OUT" is set to 0.
Output	OUT	Variance output	Follows data type	-	Minimum input value

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

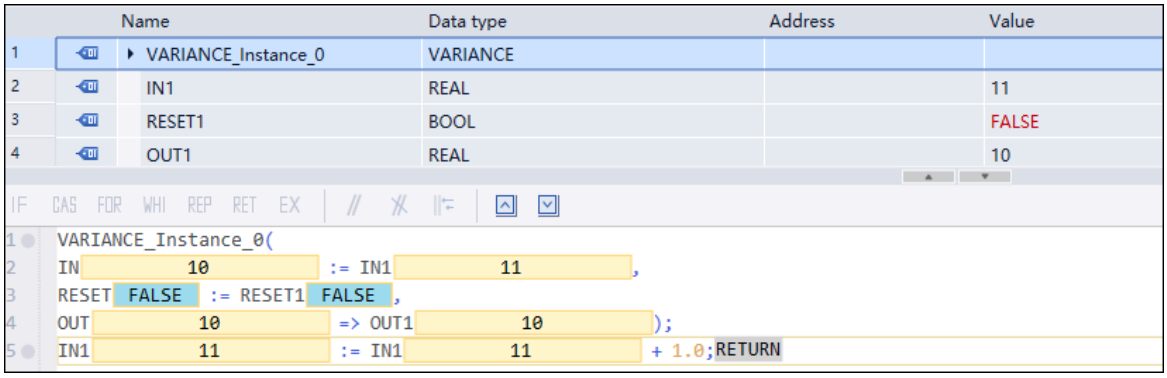
- **Function**

  Calculates the variance of input data "IN". When "RESET" is set to 1, output "OUT" is initialized to 0.

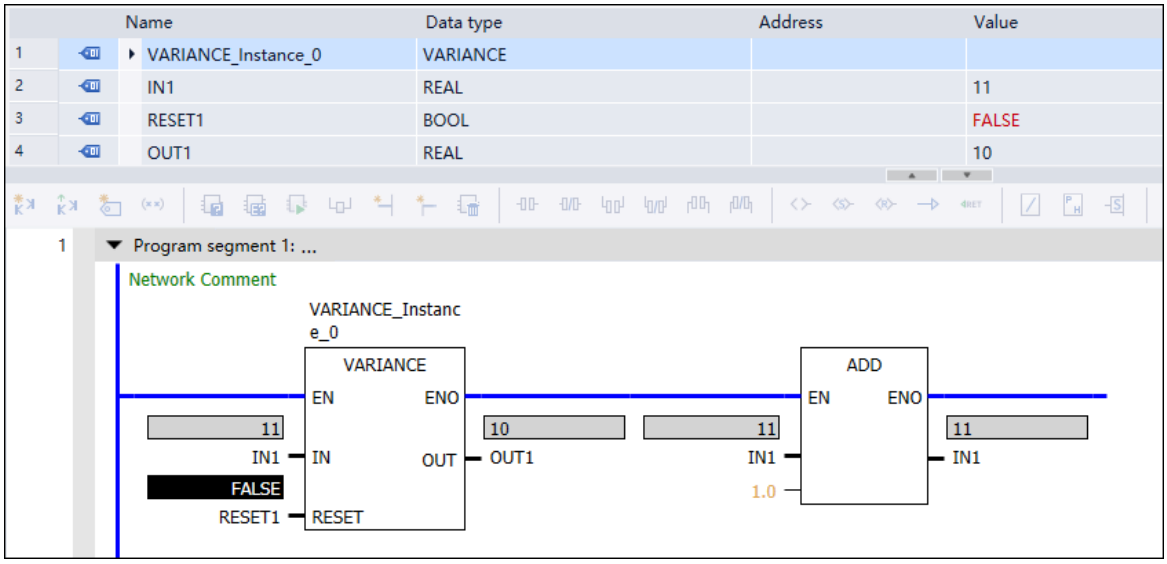
- **Program example**

  The following example sets "RESET" to 0, and "IN" to 0. "IN" increments by 1.0 per task scheduling cycle. The simulation results after 10 cycles are shown in the following figure.

  ST:



  LD:



18.2.3 BCD Code Conversion

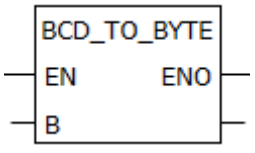
18.2.3.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.3.2 BCD_TO_BYTE	FC	BCD to byte	Util	Library version 1.0.0.0 and later
18.2.3.3 BCD_TO_DWORD	FC	BCD to double word		Library version 1.0.0.0 and later
18.2.3.4 BCD_TO_INT	FC	BCD to integer		Library version 1.0.0.0 and later
18.2.3.5 BCD_TO_WORD	FC	BCD to word		Library version 1.0.0.0 and later
18.2.3.6 BYTE_TO_BCD	FC	Byte to BCD		Library version 1.0.0.0 and later
18.2.3.7 DWORD_TO_BCD	FC	Double word to BCD		Library version 1.0.0.0 and later
18.2.3.8 INT_TO_BCD	FC	Integer to BCD		Library version 1.0.0.0 and later
18.2.3.9 WORD_TO_BCD	FC	Word to BCD		Library version 1.0.0.0 and later

18.2.3.2 BCD_TO_BYTE

Converts the BCD code of the source data into BYTE type data.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BCD_TO_BYTE	BCD to byte	FC		:= BCD_TO_BYTE(B :=);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	B	Input data	Follows data type	-	Input data
Output	OUT	Output data	Follows data type	-	Output data.

Variable types

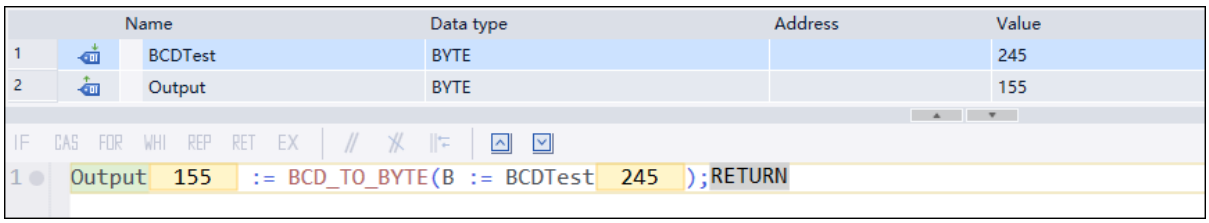
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
B	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

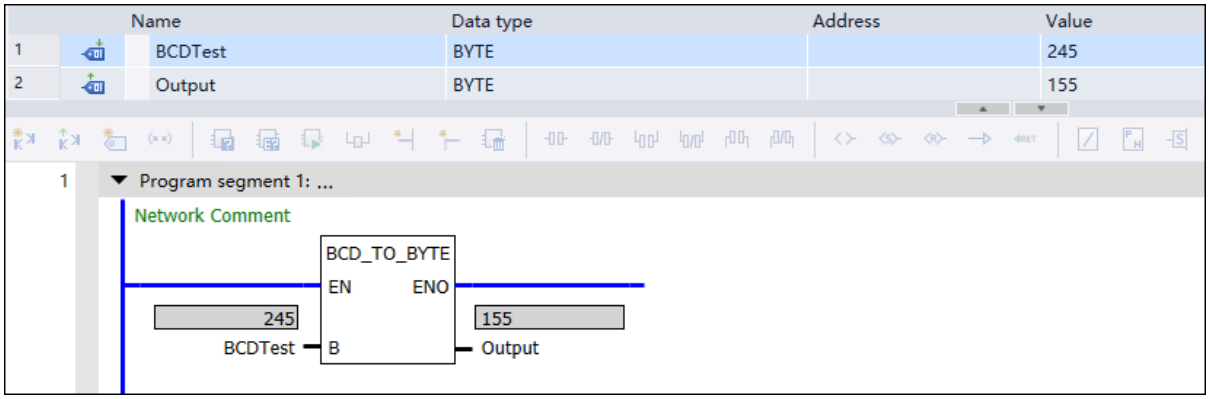
  This instruction converts BCD of the source data to bytes.

- **Program example**

  ST:



  LD:



- **Precautions**

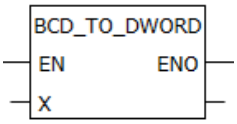
  There are no restrictions on the input of BCD_TO_BYTE. If the input is not in BCD format, it is processed with carry.

  For example: If BCDTest is 2#11110101, where the high 4 bits are 2#1111 (exceeding the BCD code format), 0110 (carry offset) will be added, resulting in 2#10101. This value is parsed as 15 in BCD format. The low-order four bits 2#0101 are parsed as 5 in BCD format. The final conversion result is 155.

18.2.3.3 BCD_TO_DWORD

Converts the BCD code of the source data into a DWORD type data.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BCD_TO_DWORD	BCD to double word	FC		:= BCD_TO_DWORD(X :=);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	X	Input data	Follows data type	-	Input data
Output	OUT	Output data	Follows data type	-	Output data.

  Variable types

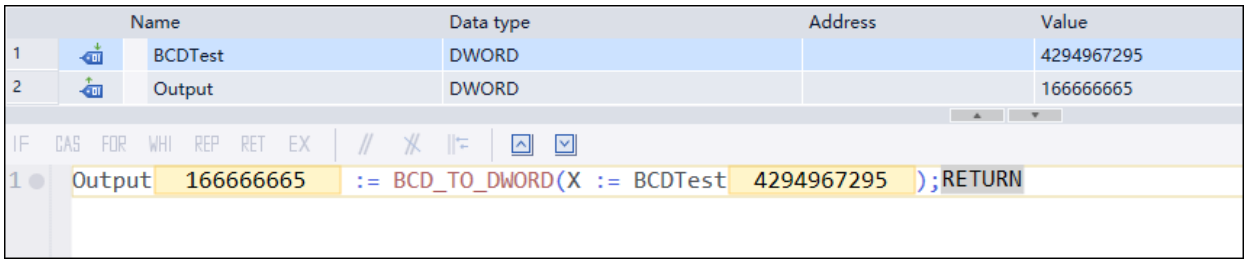
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
X	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

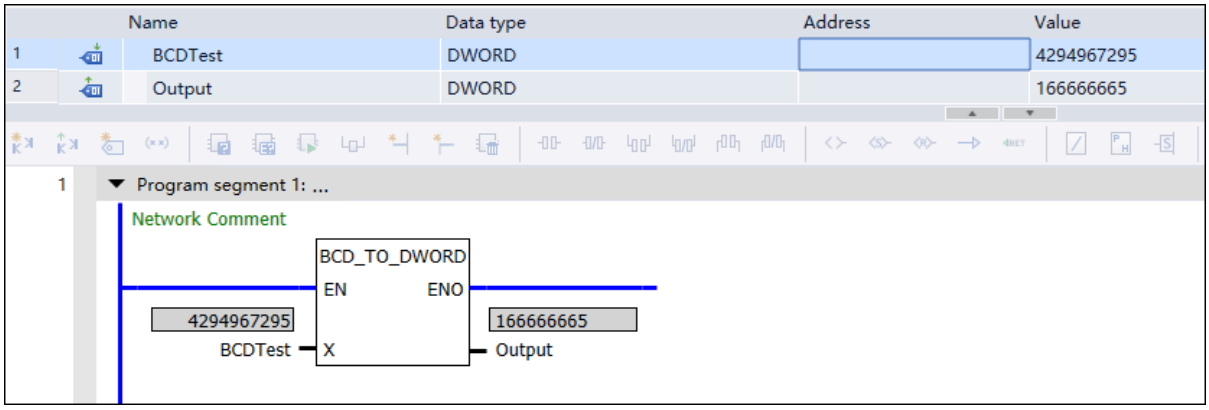
  This instruction converts BCD of the source data to doublewords.

- **Program example**

  ST:



  LD:



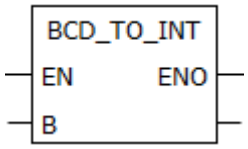
- **Precautions**

  The BCD_TO_DWORD instruction supports carry processing. For example, if BCDTest is set to 2#11111111111111111111111111111111, where the high-order 16 bits (2#1111111111111111) are parsed as decimal 16665 and the low-order 16 bits (2#1111111111111111) are parsed as decimal 16665. The final conversion result is166666665.

18.2.3.4 BCD_TO_INT

Converts the BCD code of the source data into an INT type data.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BCD_TO_INT	BCD to integer	FC		:= BCD_TO_INT(B :=);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	B	Input data	Follows data type	-	Input data
Output	OUT	Output data	Follows data type	-	Output data.

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
B	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

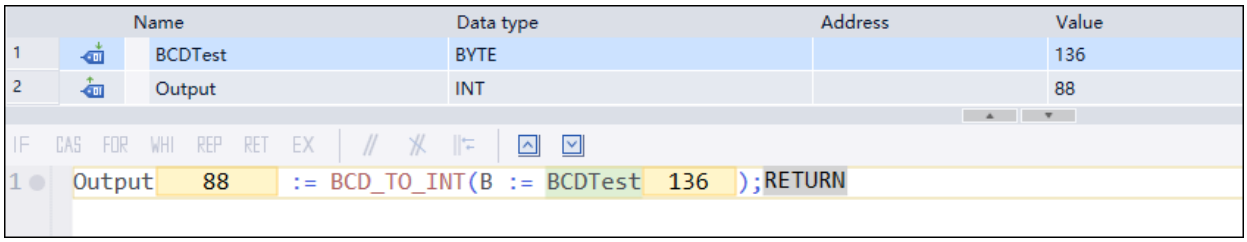
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
OUT	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

- **Function**

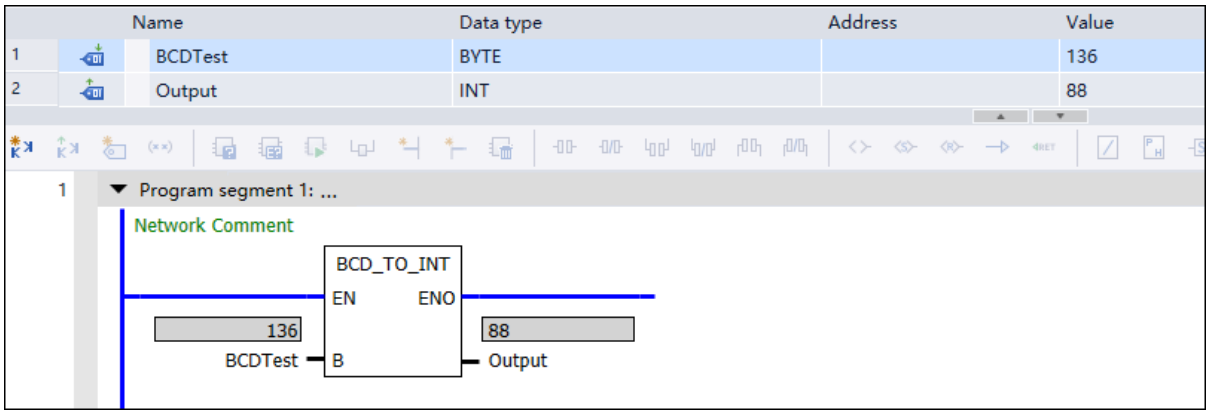
  This instruction converts BCD of the source data to integers.

- **Program example**

  ST:



  LD:



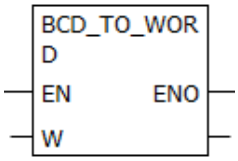
- **Precautions**

  This instruction can only convert data within the range of 0 to 99. The input must meet the BCD code format; otherwise, the output will be -1.

18.2.3.5 BCD_TO_WORD

Converts the BCD code of the source data into a WORD type data.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BCD_TO_WORD	BCD to word	FC		:= BCD_TO_WORD(W :=);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	W	Input data	Follows data type	-	Input data
Output	OUT	Output data	Follows data type	-	Output data.

  Variable types

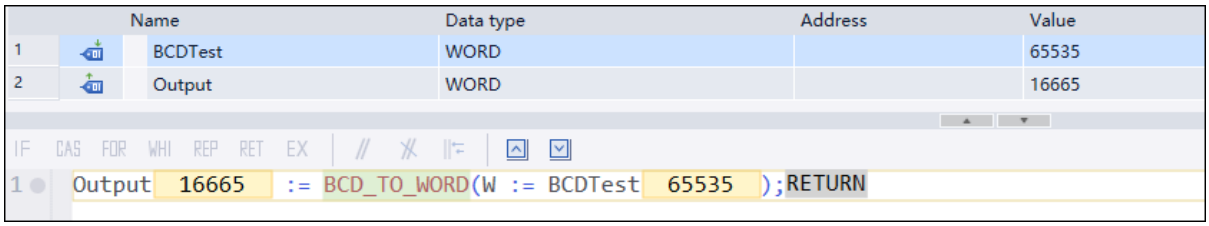
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
W	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

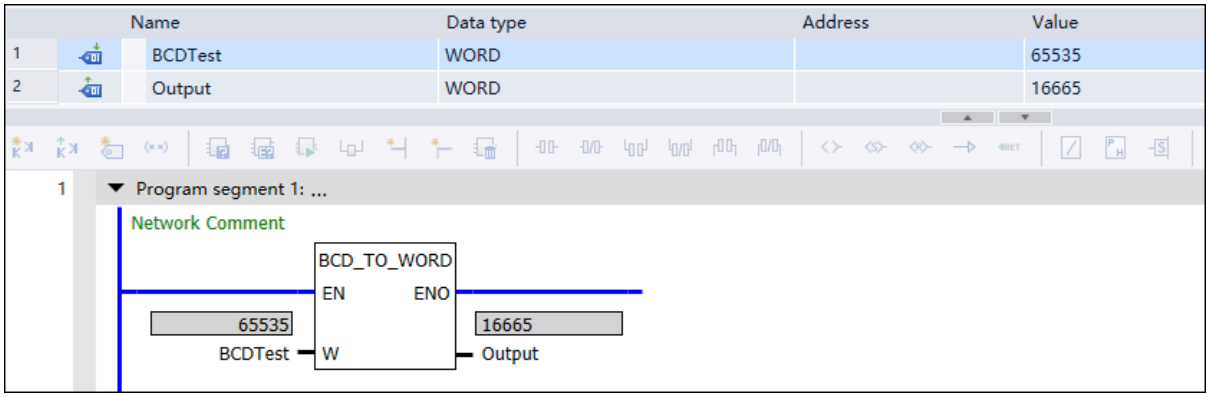
  This instruction converts BCD of the source data to words.

- **Program example**

  ST:



  LD:



- **Precautions**

  The BCD_TO_WORD instruction supports carry processing. For example, if BCDTest is set to 2#1111111111111111, where the high-order 8 bits (2#11111111) are parsed as decimal 165 and the low-order 8 bits (2#1111111) are parsed as decimal 165. The final conversion result is 16665.

18.2.3.6 BYTE_TO_BCD

Converts the source data of BYTE type into BCD code.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BYTE_TO_BCD	Byte to BCD	FC	BYTE_TO_BCD EN ENO B	:= BYTE_TO_BCD(B :=) ;

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	B	Input data	Follows data type	-	Input data
Output	OUT	Output data	Follows data type	-	Output data.

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
B	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

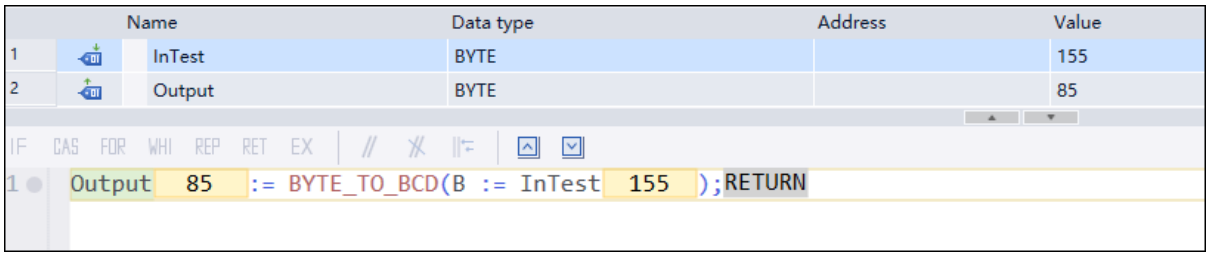
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
OUT	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

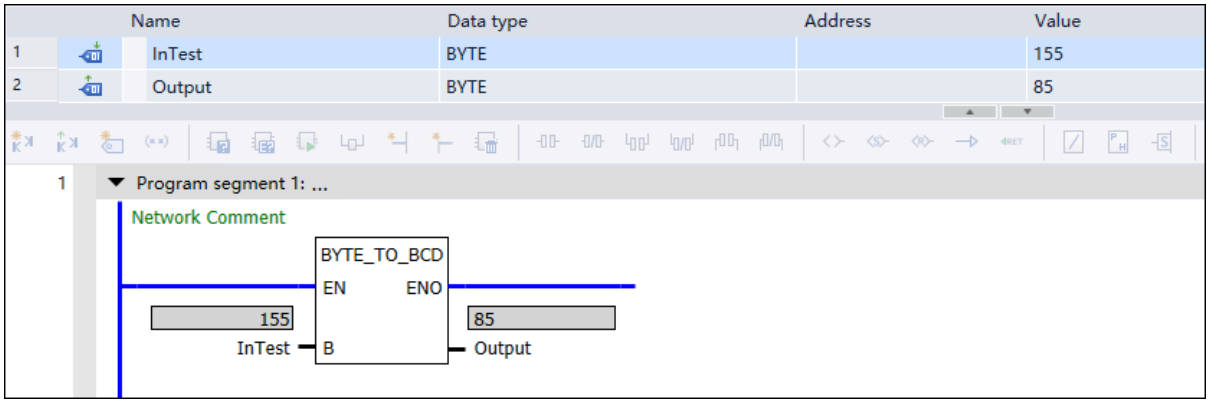
  This instruction converts bytes of the source data to BCD.

- **Program example**

  ST:



  LD:



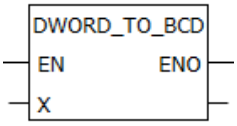
- **Precautions**

  The BYTE_TO_BCD instruction only converts the tens and ones digits.

18.2.3.7 DWORD_TO_BCD

Converts the source data of DWORD type into BCD code.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DWORD_TO_BCD	Double word to BCD	FC		:= DWORD_TO_BCD(X :=);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	X	Input data	Follows data type	-	Input data
Output	OUT	Output data	Follows data type	-	Output data.

  Variable types

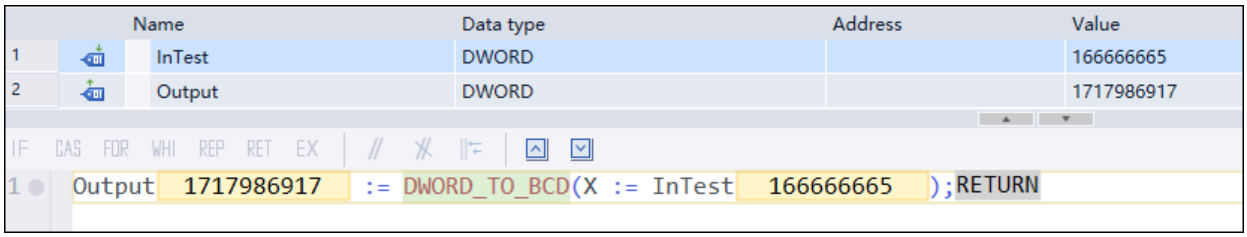
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
X	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

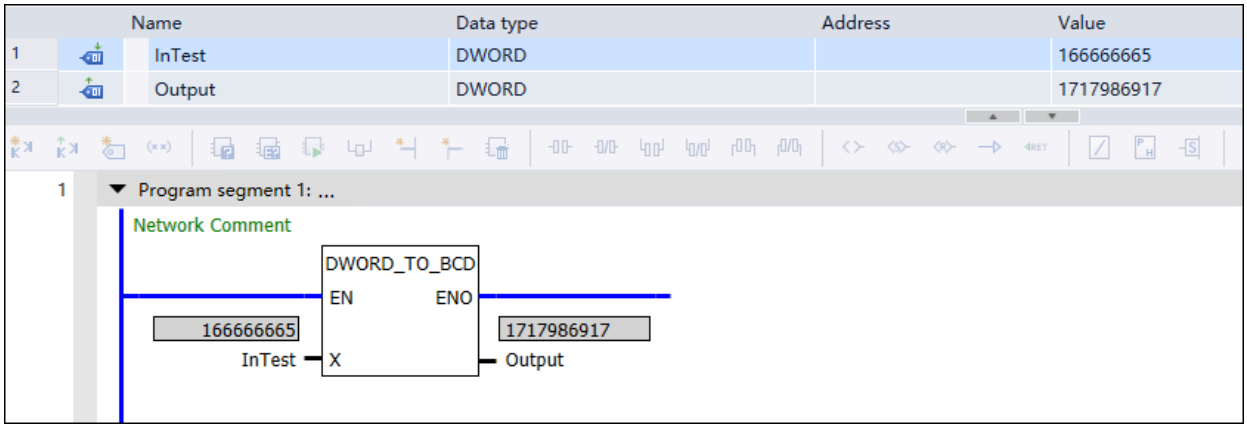
  This instruction converts doublewords of the source data to BCD.

- **Program example**

  ST:



  LD:



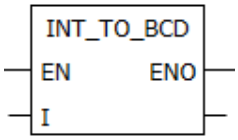
- **Precautions**

  The DWORD_TO_BCD instruction only converts to the thousands digit.

18.2.3.8 INT_TO_BCD

Converts the source data of INT type into BCD code.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
INT_TO_BCD	Integer to BCD	FC		:= INT_TO_BCD(I :=);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	I	Input data of INT type	Follows data type	-	If the integer is 39, 39 is input.
Output	OUT	BCD after conversion	Follows data type	-	If the integer 39 is converted to BCD 2#00101001, the output is 2#00101001 (10#57 or 16#39).

  Variable types

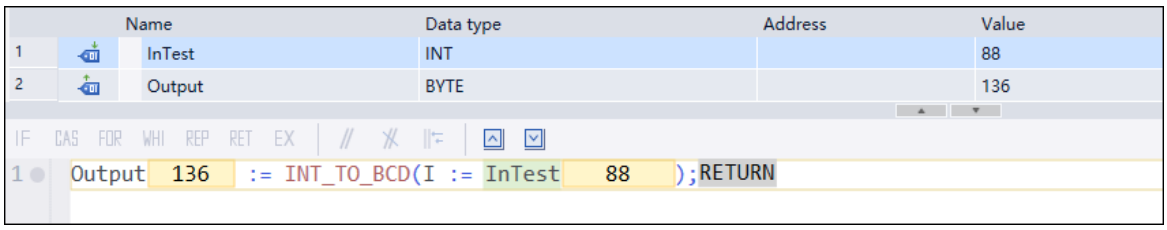
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
I	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
OUT	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

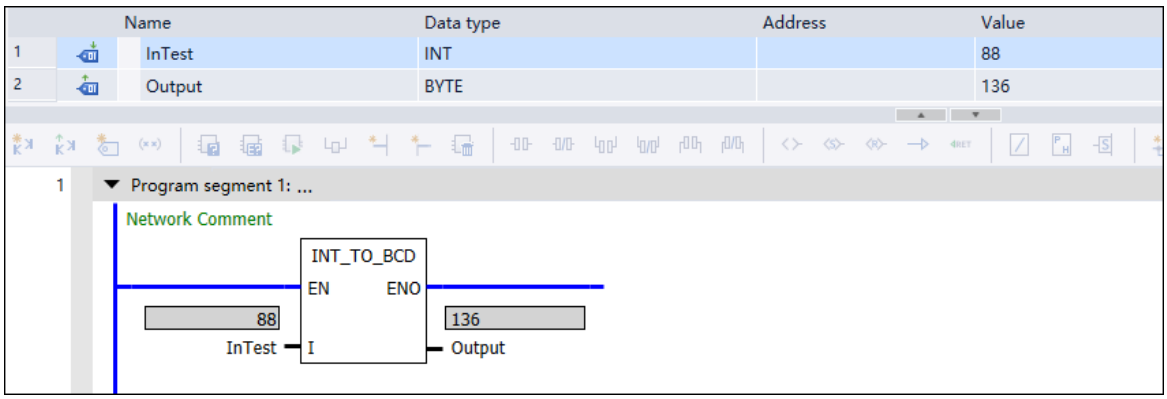
  This instruction converts integers of the source data to BCD.

- **Program example**

  ST:



  LD:



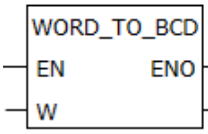
- **Precautions**

  This instruction can only convert data between 0 and 99. When the data to convert is not in the range, the conversion result is 255.

18.2.3.9 WORD_TO_BCD

Convert source data of WORD type into BCD code.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
WORD_TO_BCD	Word to BCD	FC		:= WORD_TO_BCD(W :=);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	W	Input data	Follows data type	-	Input data
Output	OUT	Output data	Follows data type	-	Output data.

  Variable types

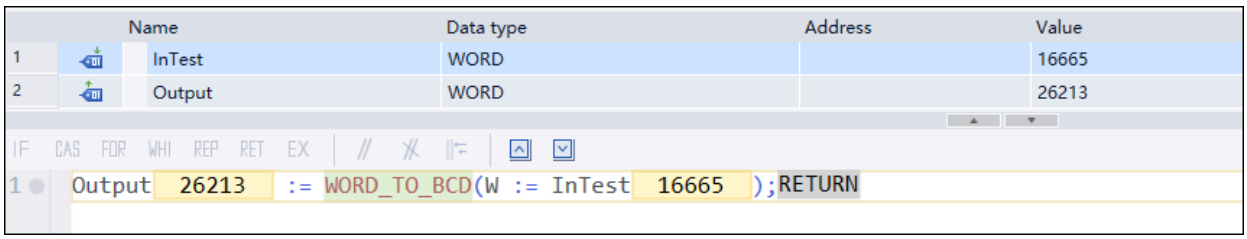
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
W	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

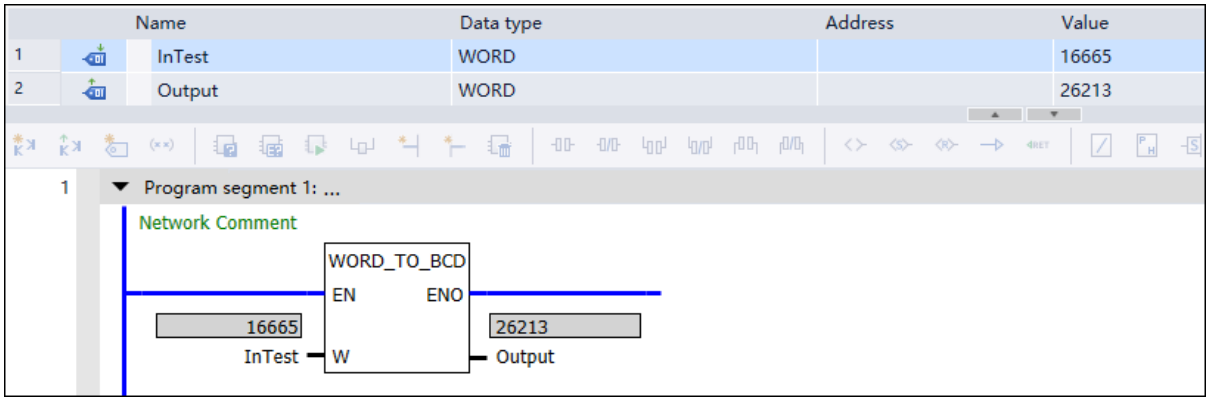
  This instruction converts words of the source data to BCD.

- **Program example**

  ST:



  LD:



- **Precautions**

  The WORD_TO_BCD instruction only converts the thousands, hundreds, tens, and ones digits.

18.2.4 Byte/Bit Operations

18.2.4.1 Instruction List

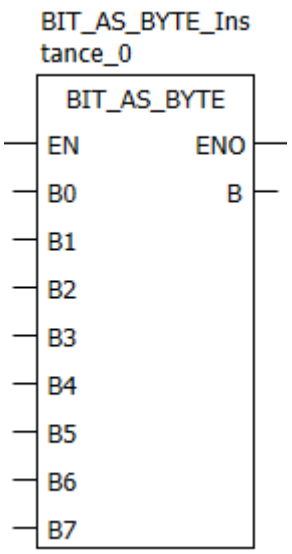
Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.4.2 BIT_AS_BYTE	FB	8 bits to byte	Util	Library version 1.0.0.0 and later
18.2.4.3 BIT_AS_WORD	FB	16 bits to word		Library version 1.0.0.0 and later
18.2.4.4 BIT_AS_DWORD	FB	32 bits to doubleword		Library version 1.0.0.0 and later
18.2.4.5 BYTE_AS_BIT	FB	Byte split		Library version 1.0.0.0 and later
18.2.4.6 WORD_AS_BIT	FB	Word split		Library version 1.0.0.0 and later
18.2.4.7 DWORD_AS_BIT	FB	Doubleword split		Library version 1.0.0.0 and later
18.2.4.8 EXTRACT	FC	Bit extract		Library version 1.0.0.0 and later
18.2.4.9 PACK	FC	Bit packing		Library version 1.0.0.0 and later
18.2.4.10 PUTBIT	FC	Bit assignment		Library version 1.0.0.0 and later
18.2.4.11 SWITCHBIT	FC	Toggle specific bit value		Library version 1.0.0.0 and later
18.2.4.12 UNPACK	FB	Bit split		Library version 1.0.0.0 and later
18.2.4.13 UnpackWord	FB	Word split		Library version 1.3.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.4.14 NSHLC	FC	Shift N-bits left with carry	OMUtils	Library version 1.0.0.0 and later
18.2.4.15 NSHRC	FC	Shift N-bits right with carry		Library version 1.0.0.0 and later

18.2.4.2 BIT_AS_BYTE

Converts eight input values of BOOL type into values of BYTE type, and outputs them from the right side.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BIT_AS_BYTE	8 bits to byte	FB		BIT_AS_BYTE_Instance_0(B0 := , B1 := , B2 := , B3 := , B4 := , B5 := , B6 := , B7 := , B =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	B0 to B7	BOOL values to be converted	Follows data type	-	BOOL values to be converted, corresponding to each bit
Output	OUT	Return value	Follows data type	-	Conversion result

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
B0 to B7	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
B	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Converts eight input values of BOOL data type to BYTE type and outputs from the right side, with B0 to B7 corresponding to bit 1 to bit 8 in the output byte.

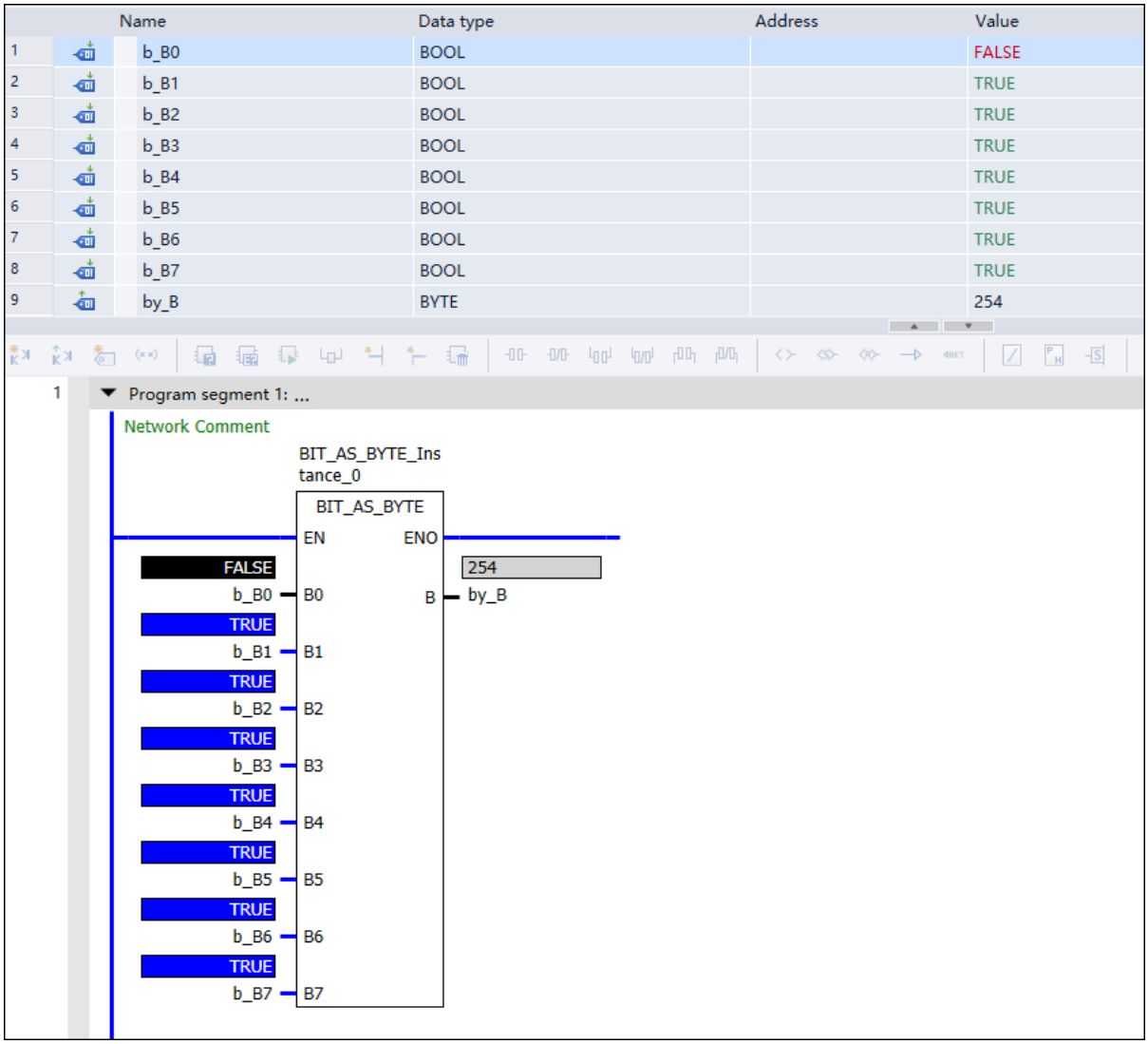
- **Program example**

  ST:

	Name	Data type	Address	Value
1	b_B0	BOOL		FALSE
2	b_B1	BOOL		TRUE
3	b_B2	BOOL		TRUE
4	b_B3	BOOL		TRUE
5	b_B4	BOOL		TRUE
6	b_B5	BOOL		TRUE
7	b_B6	BOOL		TRUE
8	b_B7	BOOL		TRUE
9	by_B	BYTE		254
10	BIT_AS_BYTE_Instance_0	BIT_AS_BYTE		

```
1  BIT_AS_BYTE_Instance_0(
2  B0 FALSE := b_B0 FALSE,
3  B1 TRUE  := b_B1 TRUE ,
4  B2 TRUE  := b_B2 TRUE ,
5  B3 TRUE  := b_B3 TRUE ,
6  B4 TRUE  := b_B4 TRUE ,
7  B5 TRUE  := b_B5 TRUE ,
8  B6 TRUE  := b_B6 TRUE ,
9  B7 TRUE  := b_B7 TRUE ,
10 B  254   => by_B  254  );RETURN
```

  LD:

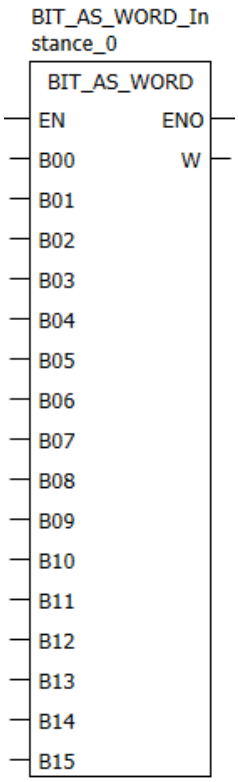


- **Precautions**
  - A fault occurs when the input is not of BOOL type.
  - The output range is 0 and 255.

18.2.4.3 BIT_AS_WORD

Packs 16 bits of the input data into a word.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BIT_AS_WORD	16 bits to word	FB		BIT_AS_WORD_Instance_0(B00 := , B01 := , B02 := , B03 := , B04 := , B05 := , B06 := , B07 := , B08 := , B09 := , B10 := , B11 := , B12 := , B13 := , B14 := , B15 := , W =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	B00 to B15	BOOL values to be converted	Follows data type	-	BOOL values to be converted, corresponding to each bit
Output	W	Return value	Follows data type	-	Conversion result

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
B00 to B15	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
W	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Converts 16 input values of BOOL type into WORD type and outputs it from the right side, with B0 to B15 corresponding to bit 1 to bit 16 in the WORD byte.

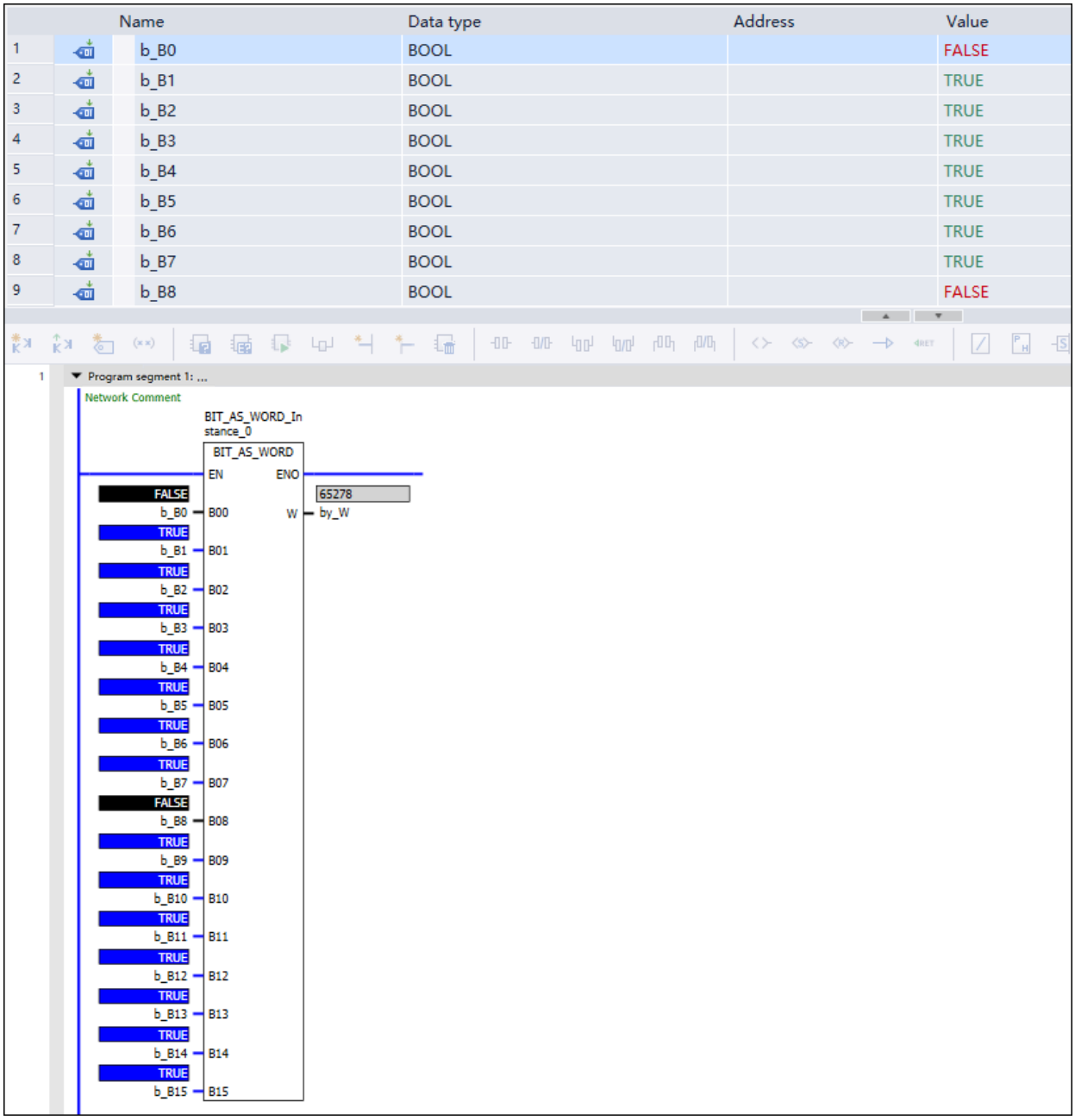
- **Program example**

  ST:

	Name	Data type	Address	Value
1	b_B0	BOOL		FALSE
2	b_B1	BOOL		TRUE
3	b_B2	BOOL		TRUE
4	b_B3	BOOL		TRUE
5	b_B4	BOOL		TRUE
6	b_B5	BOOL		TRUE
7	b_B6	BOOL		TRUE
8	b_B7	BOOL		TRUE
9	b_B8	BOOL		FALSE
10	b_B9	BOOL		TRUE
11	b_B10	BOOL		TRUE

```
IF  CAS  FOR  WHI  REP  RET  EX
1  BIT_AS_WORD_Instance_0(
2  B00 FALSE := b_B0 FALSE,
3  B01 TRUE := b_B1 TRUE,
4  B02 TRUE := b_B2 TRUE,
5  B03 TRUE := b_B3 TRUE,
6  B04 TRUE := b_B4 TRUE,
7  B05 TRUE := b_B5 TRUE,
8  B06 TRUE := b_B6 TRUE,
9  B07 TRUE := b_B7 TRUE,
10 B08 FALSE := b_B8 FALSE,
11 B09 TRUE := b_B9 TRUE,
12 B10 TRUE := b_B10 TRUE,
13 B11 TRUE := b_B11 TRUE,
14 B12 TRUE := b_B12 TRUE,
15 B13 TRUE := b_B13 TRUE,
16 B14 TRUE := b_B14 TRUE,
17 B15 TRUE := b_B15 TRUE,
18 W 65278 => by_W 65278 );RETURN
```

  LD:

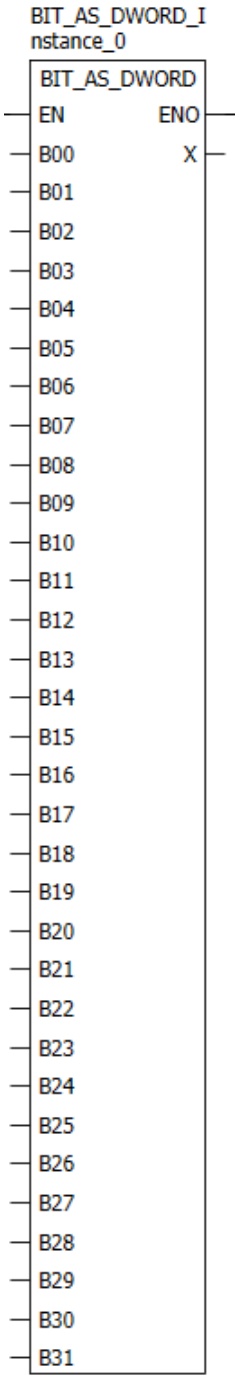


- **Precautions**
  - A fault occurs when the input is not of BOOL type.
  - The output range is 0 and 65535.

18.2.4.4 BIT_AS_DWORD

Converts 32 input values of BOOL type into values of the DWORD type, and outputs from the right side.

- **Instruction format**

BIT_AS_DWORD	32 bits to doubleword	FB		BIT_AS_DWORD_Instance_0(B00 := , B01 := , B02 := , B03 := , B04 := , B05 := , B06 := , B07 := , B08 := , B09 := , B10 := , B11 := , B12 := , B13 := , B14 := , B15 := , B16 := , B17 := , B18 := , B19 := , B20 := , B21 := , B22 := , B23 := , B24 := , B25 := , B26 := , B27 := , B28 := , B29 := , B30 := , B31 := , X =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	B00 to B31	BOOL values to be converted	Follows data type	-	BOOL values to be converted, corresponding to each bit
Output	X	Return value	Follows data type	-	Conversion result

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
B00 to B31	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
X	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Converts 32 input values of BOOL type into DWORD type and outputs it from the right side, with B0 to B31 corresponding to bit 1 to bit 32 in the DWORD byte.

- **Program example**

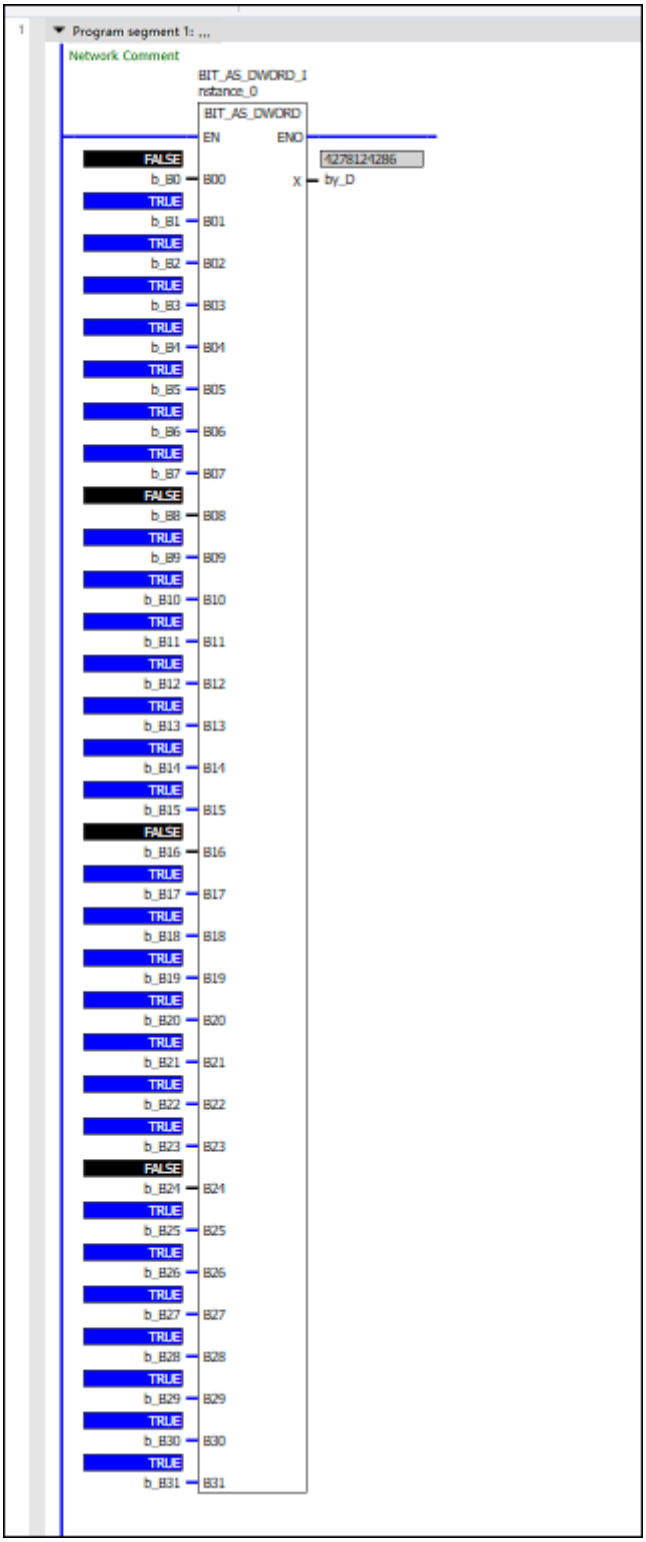
  ST:

	Name	Data type	Address	Value
1	b_B0	BOOL		FALSE
2	b_B1	BOOL		TRUE
3	b_B2	BOOL		TRUE
4	b_B3	BOOL		TRUE
5	b_B4	BOOL		TRUE
6	b_B5	BOOL		TRUE
7	b_B6	BOOL		TRUE
8	b_B7	BOOL		TRUE
9	b_B8	BOOL		FALSE
10	b_B9	BOOL		TRUE
11	b_B10	BOOL		TRUE
12	b_B11	BOOL		TRUE

IF CAS FOR WHI REP RET EX // //

```
13  B11 TRUE  := b_B11 TRUE ,
14  B12 TRUE  := b_B12 TRUE ,
15  B13 TRUE  := b_B13 TRUE ,
16  B14 TRUE  := b_B14 TRUE ,
17  B15 TRUE  := b_B15 TRUE ,
18  B16 FALSE := b_B16 FALSE ,
19  B17 TRUE  := b_B17 TRUE ,
20  B18 TRUE  := b_B18 TRUE ,
21  B19 TRUE  := b_B19 TRUE ,
22  B20 TRUE  := b_B20 TRUE ,
23  B21 TRUE  := b_B21 TRUE ,
24  B22 TRUE  := b_B22 TRUE ,
25  B23 TRUE  := b_B23 TRUE ,
26  B24 FALSE := b_B24 FALSE ,
27  B25 TRUE  := b_B25 TRUE ,
28  B26 TRUE  := b_B26 TRUE ,
29  B27 TRUE  := b_B27 TRUE ,
30  B28 TRUE  := b_B28 TRUE ,
31  B29 TRUE  := b_B29 TRUE ,
32  B30 TRUE  := b_B30 TRUE ,
33  B31 TRUE  := b_B31 TRUE ,
34  X 4278124286 => by_B 4278124286 );RETURN
```

LD:

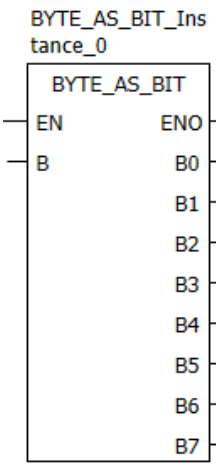


- **Precautions**
  - A fault occurs when the input is not of BOOL type.
  - The output range is 0 and 4294967295.

18.2.4.5 BYTE_AS_BIT

Converts the input value of BYTE type into eight values of BOOL type.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BYTE_AS_BIT	Byte split	FB		BYTE_AS_BIT_Instance_0(B := , B0 => , B1 => , B2 => , B3 => , B4 => , B5 => , B6 => , B7 =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	B	BYTE value to be converted	Follows data type	-	BYTE value to be converted
Output	B0 to B7	Return value	Follows data type	-	Conversion result

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
B	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
B0 to B7	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Converts the input value of BYTE type into eight values of BOOL type.

- **Program example**

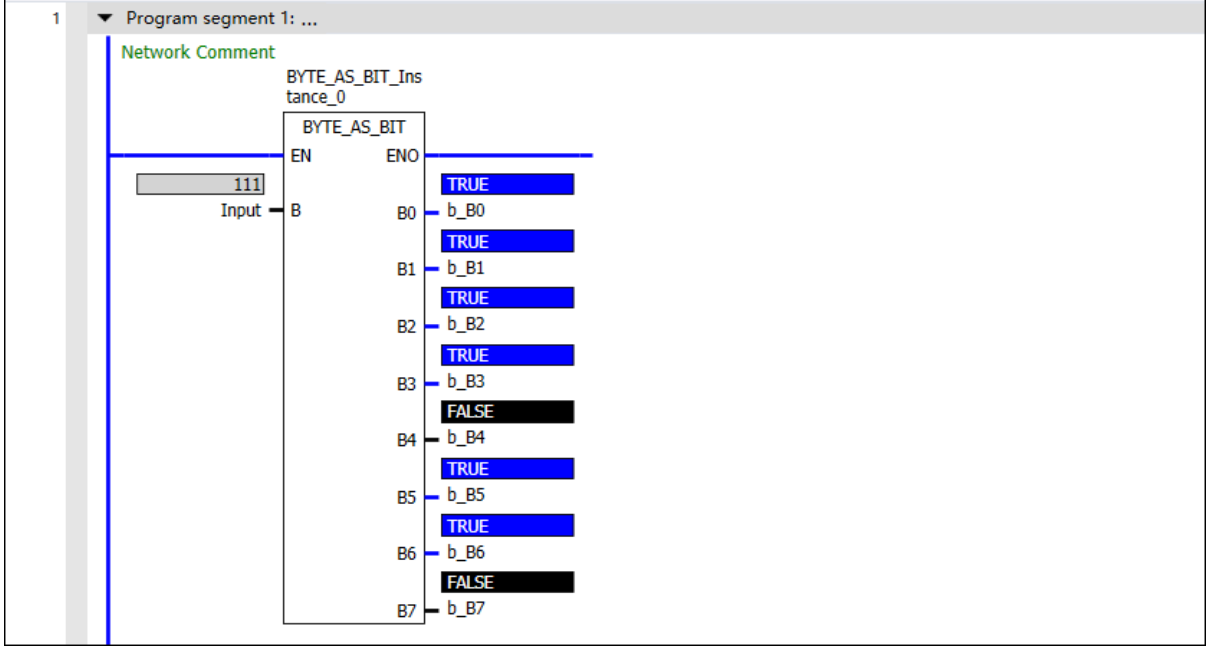
  ST:

	Name	Data type	Address	Value
1	Input	BYTE		111
2	b_B0	BOOL		TRUE
3	b_B1	BOOL		TRUE
4	b_B2	BOOL		TRUE
5	b_B3	BOOL		TRUE
6	b_B4	BOOL		FALSE
7	b_B5	BOOL		TRUE
8	b_B6	BOOL		TRUE
9	b_B7	BOOL		FALSE

```
BYTE_AS_BIT_Instance_0(
B  111   := Input  111  ,
B0 TRUE  => b_B0 TRUE ,
B1 TRUE  => b_B1 TRUE ,
B2 TRUE  => b_B2 TRUE ,
B3 TRUE  => b_B3 TRUE ,
B4 FALSE => b_B4 FALSE ,
B5 TRUE  => b_B5 TRUE ,
B6 TRUE  => b_B6 TRUE ,
B7 FALSE => b_B7 FALSE );RETURN
```

LD:

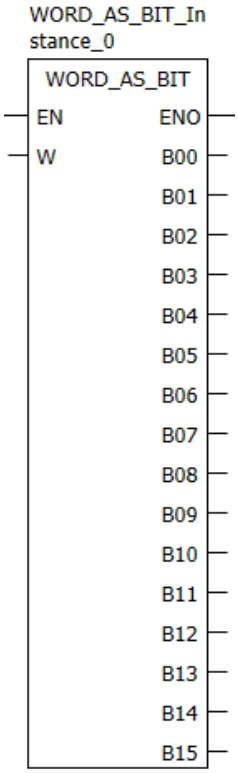
	Name	Data type	Address	Value
1	Input	BYTE		111
2	b_B0	BOOL		TRUE
3	b_B1	BOOL		TRUE
4	b_B2	BOOL		TRUE
5	b_B3	BOOL		TRUE
6	b_B4	BOOL		FALSE
7	b_B5	BOOL		TRUE
8	b_B6	BOOL		TRUE
9	b_B7	BOOL		FALSE



18.2.4.6 WORD_AS_BIT

Converts the word in W into bits and outputs them from the right side.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
WORD_AS_BIT	Word split	FB		WORD_AS_BIT_Instance_0(W := , B00 => , B01 => , B02 => , B03 => , B04 => , B05 => , B06 => , B07 => , B08 => , B09 => , B10 => , B11 => , B12 => , B13 => , B14 => , B15 =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	W	WORD value to be converted	Follows data type	-	WORD value to be converted
Output	B00 to B15	Return value	Follows data type	-	Conversion result

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
W	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
B00 to B15	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Converts the word in W into bits and outputs them from the right side.

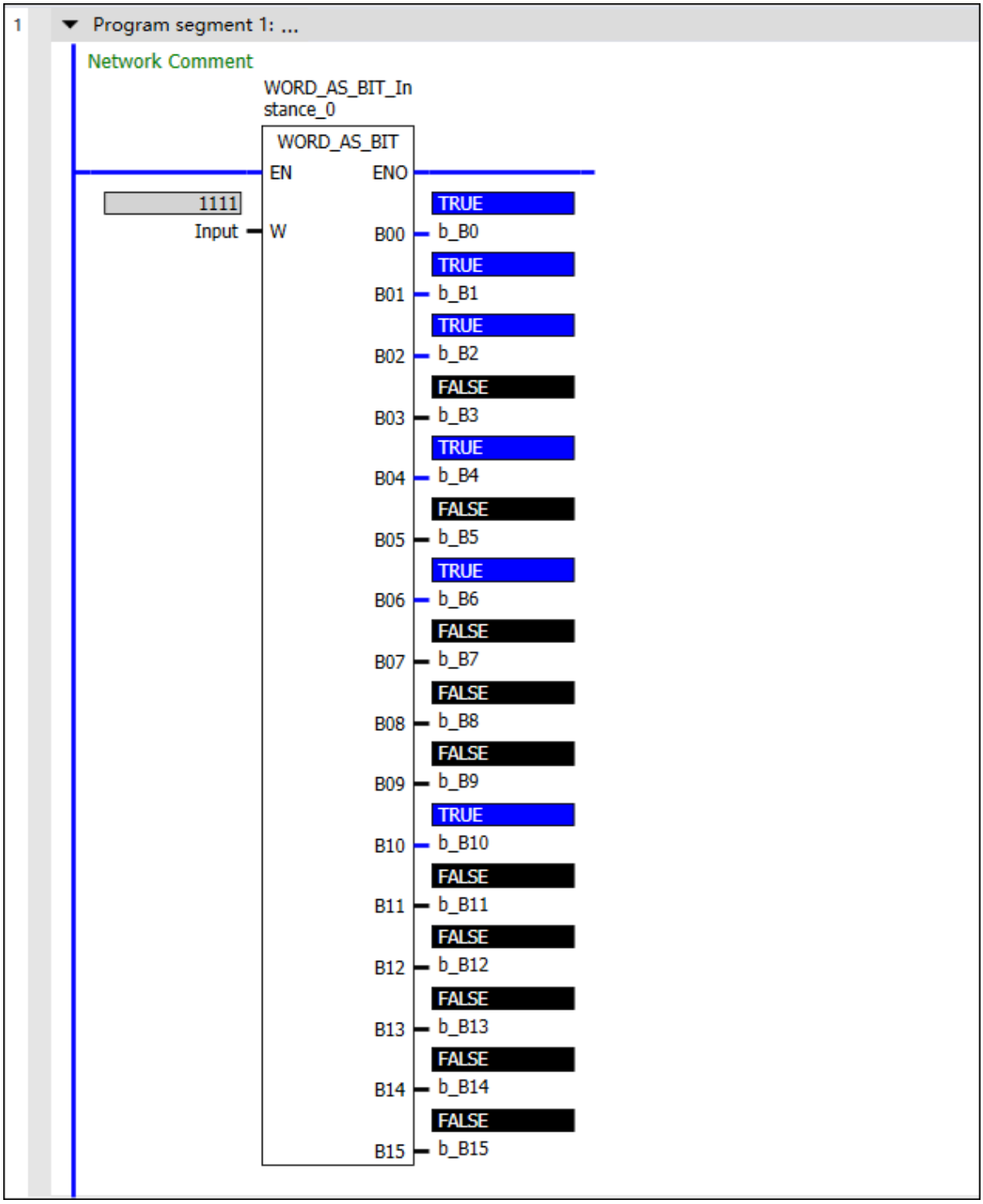
- **Program example**

  ST:

	Name	Data type	Address	Value	Prepare value	Comment
1	Input	WORD		1111		
2	b_B0	BOOL		TRUE		
3	b_B1	BOOL		TRUE		
4	b_B2	BOOL		TRUE		
5	b_B3	BOOL		FALSE		
6	b_B4	BOOL		TRUE		
7	b_B5	BOOL		FALSE		
8	b_B6	BOOL		TRUE		
9	b_B7	BOOL		FALSE		
10	b_B8	BOOL		FALSE		
11	b_B9	BOOL		FALSE		
12	b_B10	BOOL		TRUE		
13	b_B11	BOOL		FALSE		
14	b_B12	BOOL		FALSE		
15	b_B13	BOOL		FALSE		
16	b_B14	BOOL		FALSE		
17	b_B15	BOOL		FALSE		

```
IF  CAS  FOR  WHI  REP  RET  EX
1  WORD_AS_BIT_Instance_0(
2  W  1111  := Input  1111  ,B00  TRUE  => b_B0  TRUE  ,B01  TRUE  => b_B1  TRUE  ,B02  TRUE  => b_B2  TRUE  ,
3  B03  FALSE  => b_B3  FALSE  ,B04  TRUE  => b_B4  TRUE  ,B05  FALSE  => b_B5  FALSE  ,B06  TRUE  => b_B6  TRUE  ,
4  B07  FALSE  => b_B7  FALSE  ,B08  FALSE  => b_B8  FALSE  ,B09  FALSE  => b_B9  FALSE  ,B10  TRUE  => b_B10  TRUE  ,
5  B11  FALSE  => b_B11  FALSE  ,B12  FALSE  => b_B12  FALSE  ,B13  FALSE  => b_B13  FALSE  ,B14  FALSE  => b_B14  FALSE  ,
6  B15  FALSE  => b_B15  FALSE  );RETURN
```

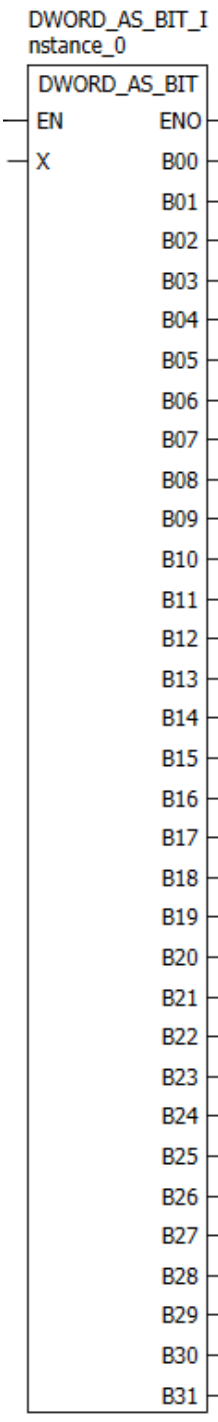
  LD:



18.2.4.7 DWORD_AS_BIT

Converts the specified doubleword into bits and outputs them from the right side.

- **Instruction format**

DWORD_AS_BIT	Doubleword split	FB		DWORD_AS_BIT_Instance_0(X := , B00 => , B01 => , B02 => , B03 => , B04 => , B05 => , B06 => , B07 => , B08 => , B09 => , B10 => , B11 => , B12 => , B13 => , B14 => , B15 => , B16 => , B17 => , B18 => , B19 => , B20 => , B21 => , B22 => , B23 => , B24 => , B25 => , B26 => , B27 => , B28 => , B29 => , B30 => , B31 =>);

- **Related variables**

Input variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	X	DWORD value to be converted	Follows data type	-	DWORD value to be converted.
Output	B00 to B31	Return value	Follows data type	-	Conversion result

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
X	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
B00 to B31	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Converts the specified doubleword into bits and outputs them from the right side.

- **Program example**

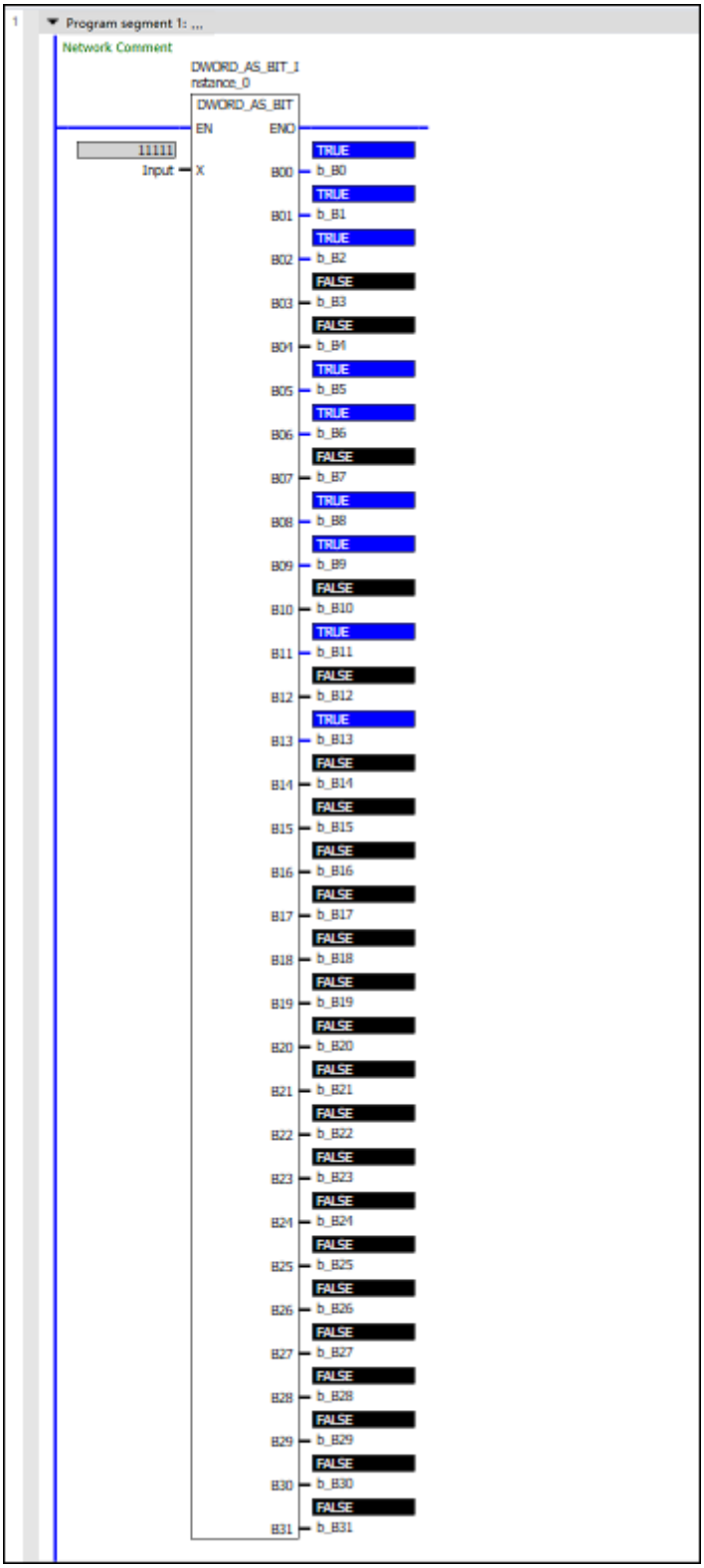
  ST:



	Name	Data type	Address	Value	Prepare value	Comment
1	Input	DWORD		11111		
2	b_B0	BOOL		TRUE		
3	b_B1	BOOL		TRUE		
4	b_B2	BOOL		TRUE		
5	b_B3	BOOL		FALSE		
6	b_B4	BOOL		FALSE		
7	b_B5	BOOL		TRUE		
8	b_B6	BOOL		TRUE		
9	b_B7	BOOL		FALSE		
10	b_B8	BOOL		TRUE		
11	b_B9	BOOL		TRUE		
12	b_B10	BOOL		FALSE		
13	b_B11	BOOL		TRUE		
14	b_B12	BOOL		FALSE		
15	b_B13	BOOL		TRUE		

```
DWORD_AS_BIT_Instance_0(
X := 11111 := Input := 11111,
B00 := TRUE => b_B0 := TRUE ,B01 := TRUE => b_B1 := TRUE ,B02 := TRUE => b_B2 := TRUE ,B03 := FALSE => b_B3 := FALSE ,
B04 := FALSE => b_B4 := FALSE ,B05 := TRUE => b_B5 := TRUE ,B06 := TRUE => b_B6 := TRUE ,B07 := FALSE => b_B7 := FALSE ,
B08 := TRUE => b_B8 := TRUE ,B09 := TRUE => b_B9 := TRUE ,B10 := FALSE => b_B10 := FALSE ,B11 := TRUE => b_B11 := TRUE ,
B12 := FALSE => b_B12 := FALSE ,B13 := TRUE => b_B13 := TRUE ,B14 := FALSE => b_B14 := FALSE ,B15 := FALSE => b_B15 := FALSE ,
B16 := FALSE => b_B16 := FALSE ,B17 := FALSE => b_B17 := FALSE ,B18 := FALSE => b_B18 := FALSE ,B19 := FALSE => b_B19 := FALSE ,
B20 := FALSE => b_B20 := FALSE ,B21 := FALSE => b_B21 := FALSE ,B22 := FALSE => b_B22 := FALSE ,B23 := FALSE => b_B23 := FALSE ,
B24 := FALSE => b_B24 := FALSE ,B25 := FALSE => b_B25 := FALSE ,B26 := FALSE => b_B26 := FALSE ,B27 := FALSE => b_B27 := FALSE ,
B28 := FALSE => b_B28 := FALSE ,B29 := FALSE => b_B29 := FALSE ,B30 := FALSE => b_B30 := FALSE ,B31 := FALSE => b_B31 := FALSE );RETURN
```

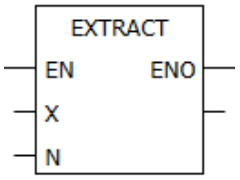
  LD:



18.2.4.8 EXTRACT

Reads the status of bit N in data source X and outputs from the right side.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
EXTRACT	Bit extract	FC		:= EXTRACT(X := , N :=);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	X	Data source	Follows data type	-	Data to operate
	N	Read bit	Follows data type	-	Bit to read
Output	OUT	Read bit status	Follows data type	-	Status of the read bit

  Variable types

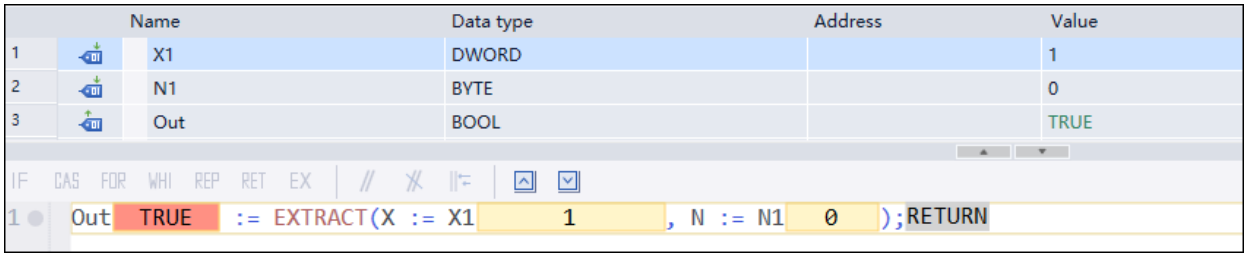
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
X	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
N	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

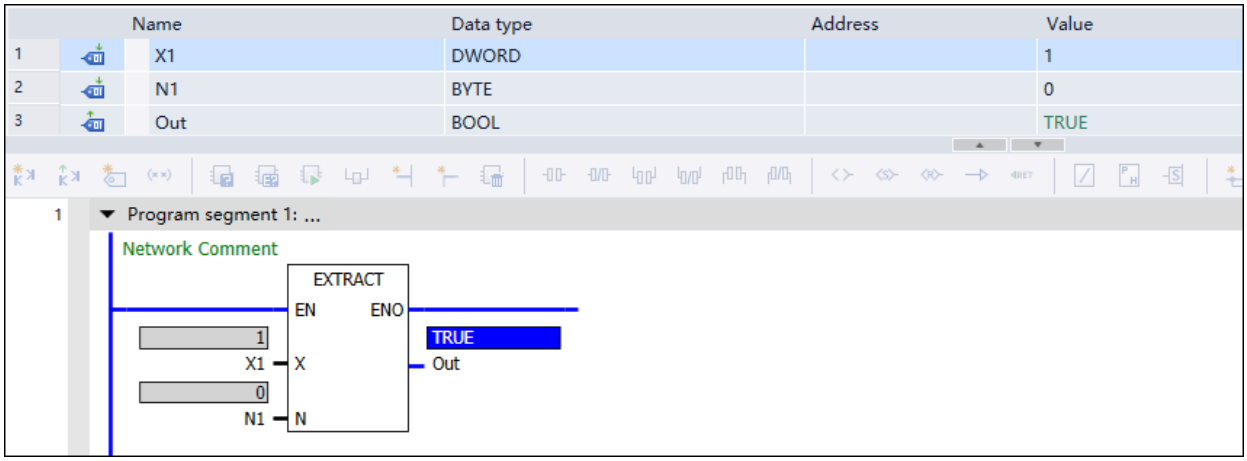
  This instruction reads the status of bit N in data source X and provides output from the right side.

- **Program example**

  ST:



LD:



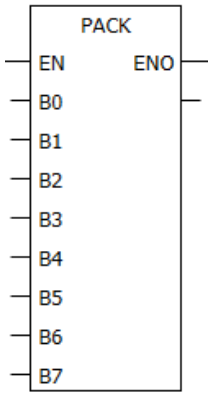
- **Precautions**

  If the bit to read exceeds the data boundary or falls outside the normal numerical operation range, the returned result may be abnormal.

18.2.4.9 PACK

Packs eight bits of the input data into a byte and outputs it from the right side.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PACK	Bit packing	FC		:= PACK(B0 := , B1 := , B2 := , B3 := , B4 := , B5 := , B6 := , B7 :=);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	B0 to B7	Data source	Follows data type	-	Bits to pack
Output	OUT	Output target	Follows data type	-	Conversion result

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
B0 to B7	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Packs eight bits of the input data into a byte and outputs it from the right side.

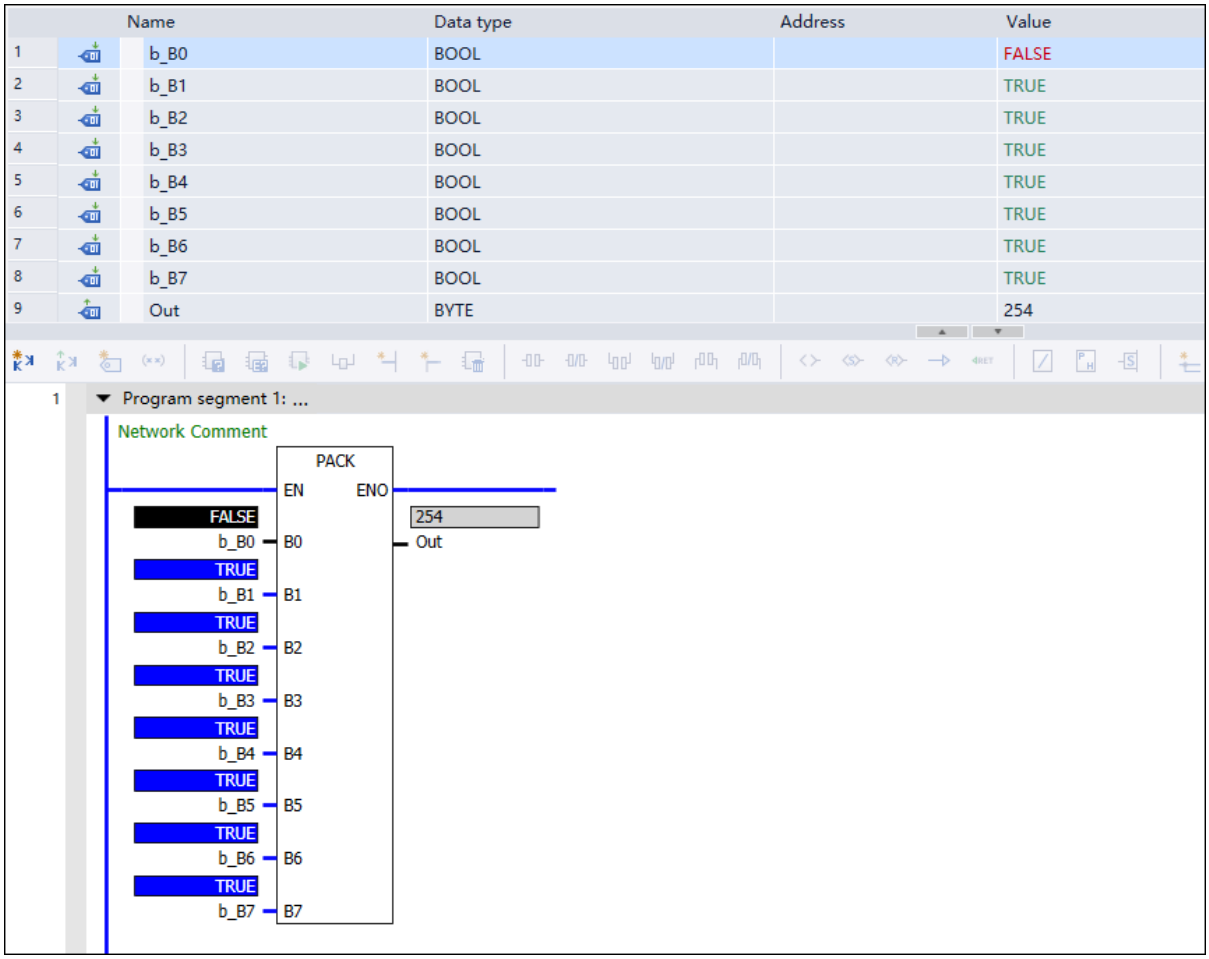
- **Program example**

  ST:

	Name	Data type	Address	Value
1	b_B0	BOOL		FALSE
2	b_B1	BOOL		TRUE
3	b_B2	BOOL		TRUE
4	b_B3	BOOL		TRUE
5	b_B4	BOOL		TRUE
6	b_B5	BOOL		TRUE
7	b_B6	BOOL		TRUE
8	b_B7	BOOL		TRUE
9	Out	BYTE		254

```
IF  CAS  FOR  WHI  REP  RET  EX
1  Out 254 := PACK(
2  B0 := b_B0 FALSE , B1 := b_B1 TRUE , B2 := b_B2 TRUE , B3 := b_B3 TRUE ,
3  B4 := b_B4 TRUE , B5 := b_B5 TRUE , B6 := b_B6 TRUE , B7 := b_B7 TRUE );RETURN
```

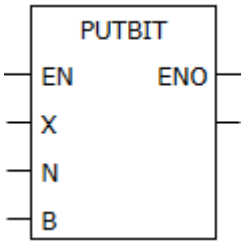
  LD:



18.2.4.10 PUTBIT

PUTBIT is used for bit assignment.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PUTBIT	Bit assignment	FC		:= PUTBIT(X := , N := , B :=);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	X	Data source	Follows data type	-	Data to operate
	N	Operation bit	Follows data type	-	Bit to operate

Input/Output	Variable	Name	Valid Range	Default Value	Description
	B	Operating switch	Follows data type	-	Operating switch of bit
Output	OUT	Target position	Follows data type	-	Data after operation

Variable types

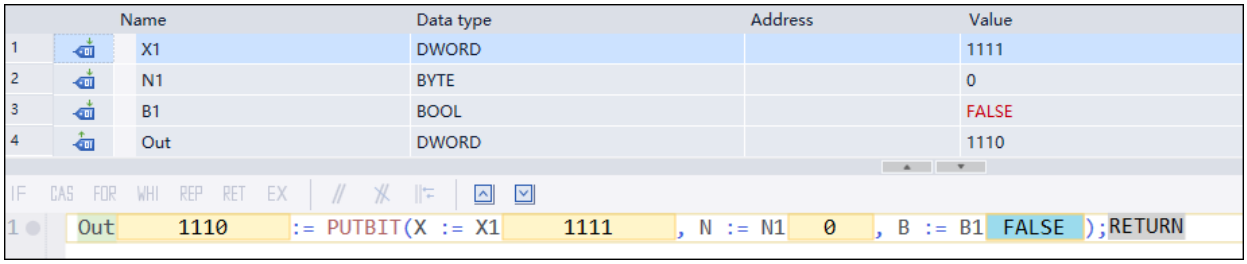
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
X	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
N	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
B	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

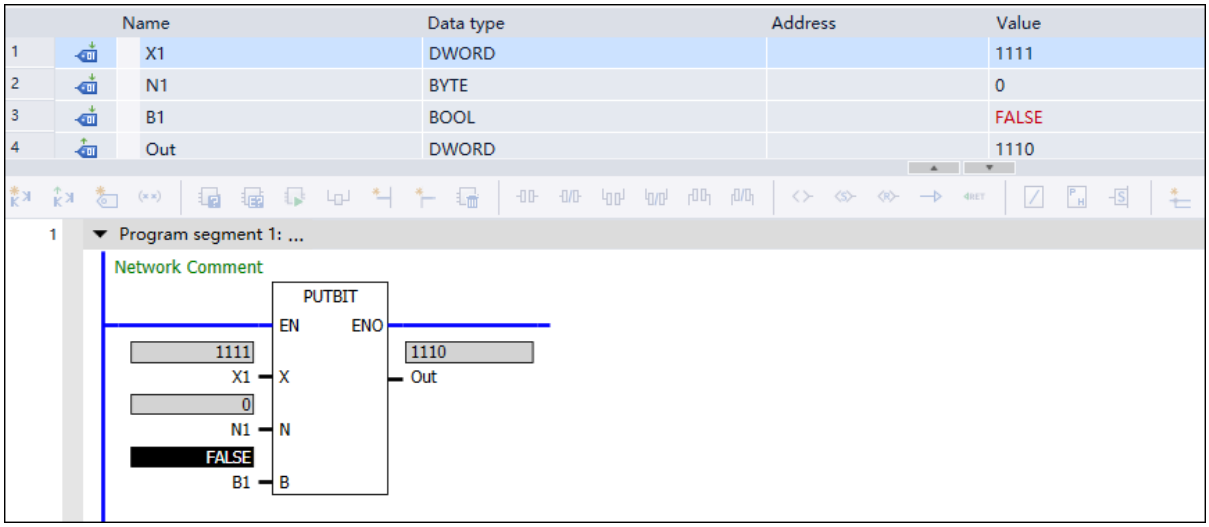
  Modifies the bit N of input X to the switch value B.

- **Program example**

  ST:



  LD:



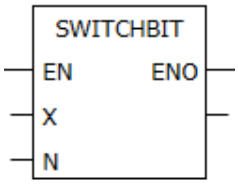
- **Precautions**

  If the bit to operate exceeds the data boundary or falls outside the normal numerical operation range, the returned result may be abnormal.

18.2.4.11 SWITCHBIT

Toggles specific bit value.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SWITCHBIT	Toggle specific bit value	FC		:= SWITCHBIT(X := , N :=);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	X	Data source	Follows data type	-	Data to operate
	N	Read bit	Follows data type	-	Bit to read
Output	OUT	Return value	Follows data type	-	Return value.

  Variable types

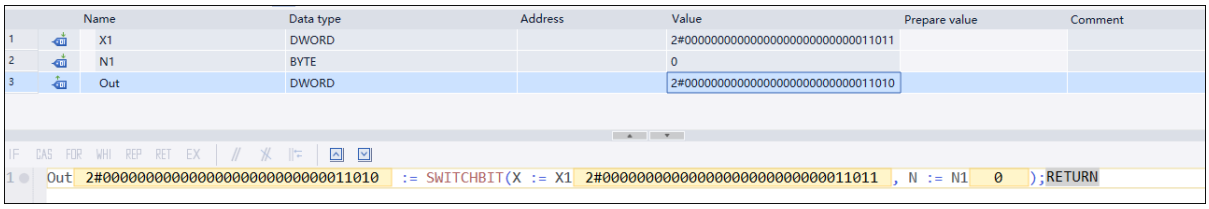
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
X	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
N	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

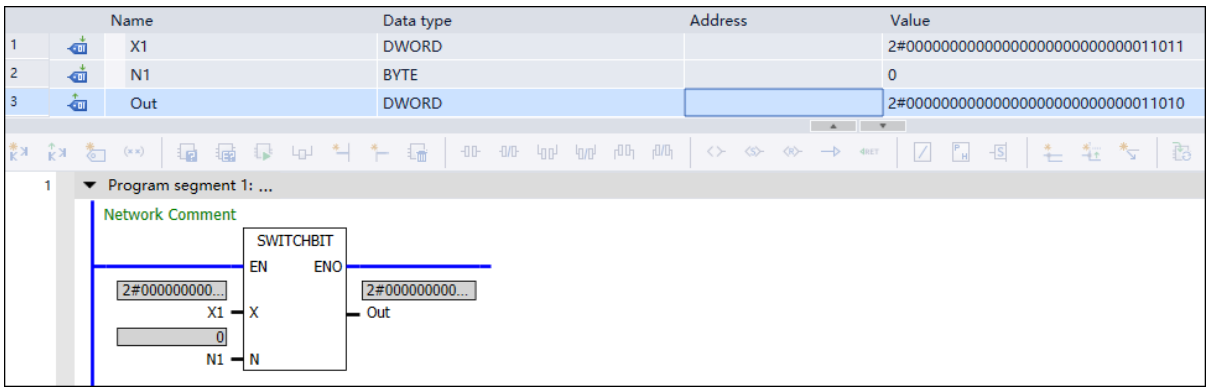
  Toggles the status of bit N in data source X, and outputs the result as X.

- **Program example**

  ST:



  LD:



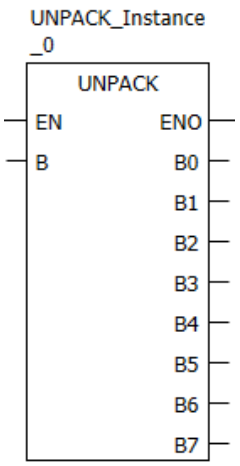
- **Precautions**

  If the bit to read exceeds the data boundary or falls outside the normal numerical operation range, the returned result may be abnormal.

18.2.4.12 UNPACK

Converts the byte in B into bits and outputs them from the right side.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
UNPACK	Bit split	FB		UNPACK_Instance_0(B := , B0 => , B1 => , B2 =>, B3 => , B4 => , B5 => , B6 => , B7 =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	B	Data source	Follows data type	-	Data to operate
Output	B0 to B7	Output target	Follows data type	-	Conversion result

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
B	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
B0 to B7	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Converts the byte in B into bits and outputs them from the right side.

- **Program example**

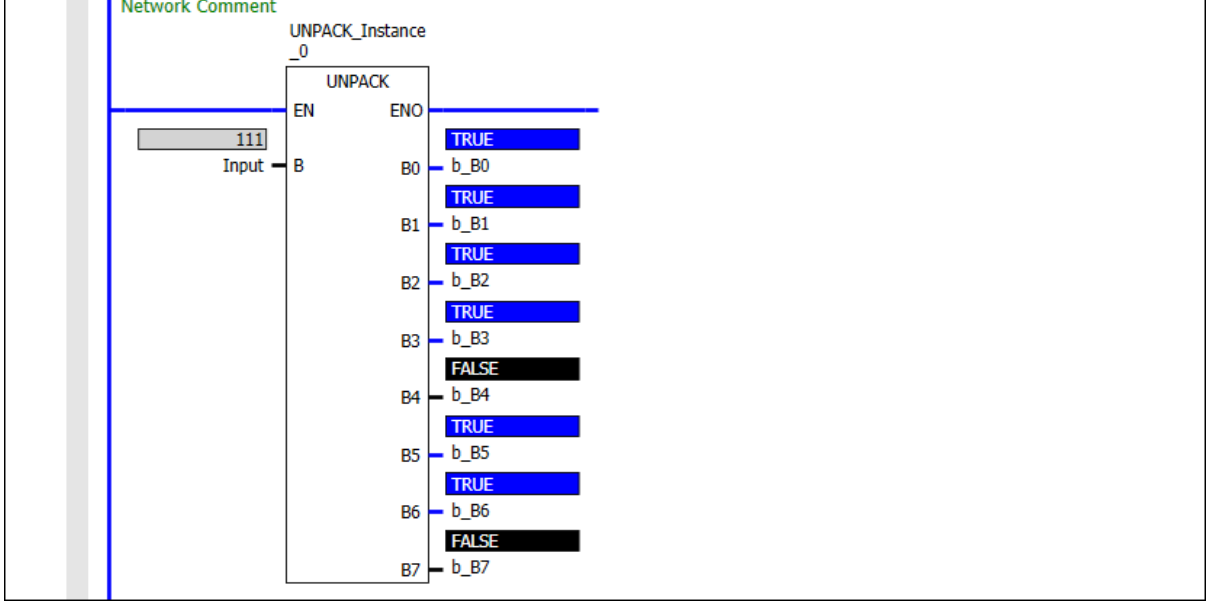
  ST:

	Name	Data type	Address	Value	Prepare value	Comment
1	Input	BYTE		2#01101111		
2	b_B0	BOOL		TRUE		
3	b_B1	BOOL		TRUE		
4	b_B2	BOOL		TRUE		
5	b_B3	BOOL		TRUE		
6	b_B4	BOOL		FALSE		
7	b_B5	BOOL		TRUE		
8	b_B6	BOOL		TRUE		
9	b_B7	BOOL		FALSE		

```
UNPACK_Instance_0(
B := Input ,B0 => b_B0 ,B1 => b_B1 ,B2 => b_B2 ,
B3 => b_B3 ,B4 => b_B4 ,B5 => b_B5 ,B6 => b_B6 ,
B7 => b_B7 );RETURN
```

LD:

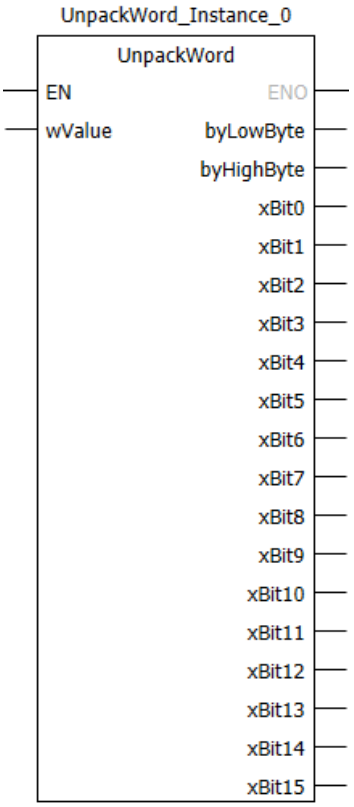
	Name	Data type	Address	Value
1	Input	BYTE		111
2	b_B0	BOOL		TRUE
3	b_B1	BOOL		TRUE
4	b_B2	BOOL		TRUE
5	b_B3	BOOL		TRUE
6	b_B4	BOOL		FALSE
7	b_B5	BOOL		TRUE
8	b_B6	BOOL		TRUE
9	b_B7	BOOL		FALSE



18.2.4.13 UnpackWord

Outputs the values of the lower eight bits, upper eight bits, and each bit of the WORD type input.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
UnpackWord	Word split	FB		UnpackWord_Instance_0(wValue := , byLowByte => , wValue xBit0 => , xBit1 => , xBit2 => , xBit3 => , xBit4 => , xBit5 => , xBit6 => , xBit7 => , xBit8 => , xBit9 => , xBit10 => , xBit11 => , xBit12 => , xBit13 => , xBit14 => , xBit15 =>);

- **Related variables**

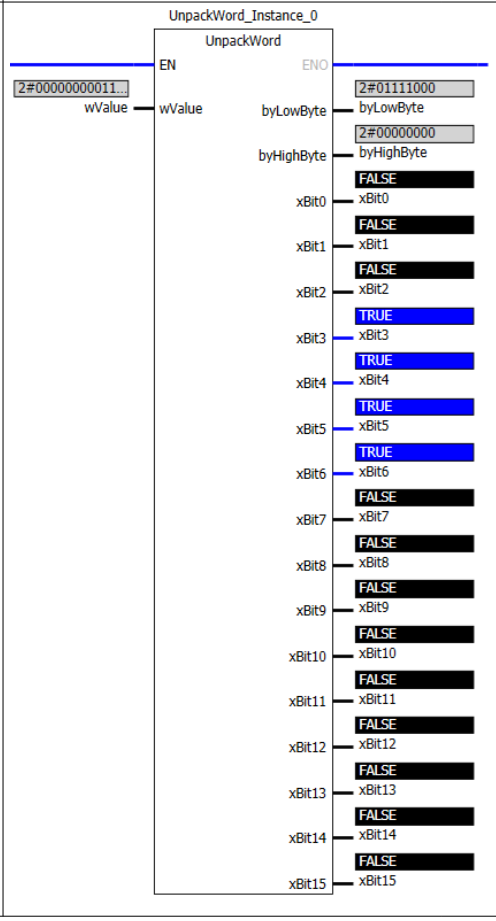
Variable

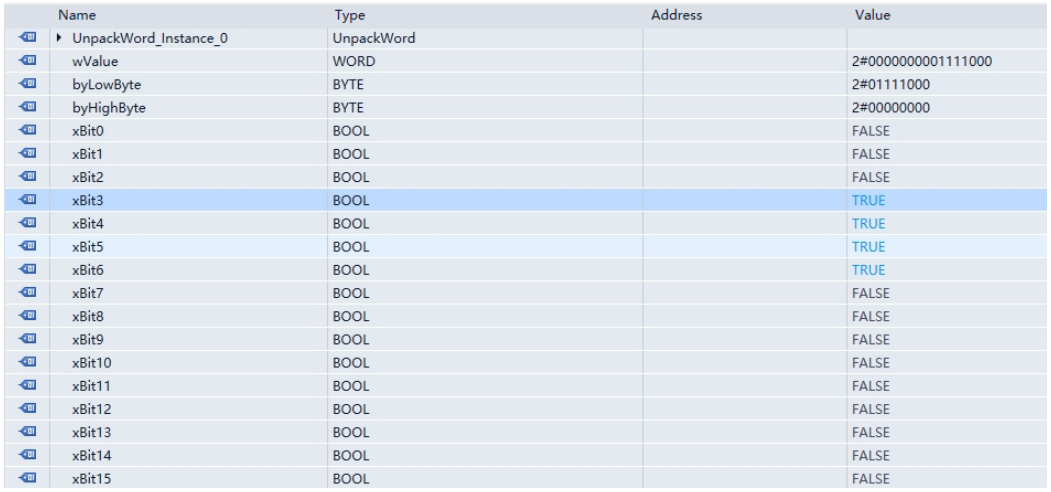
Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	wValue	WORD value to be converted	Follows data type	-	WORD value to be converted
Output	byLowByte	Lower eight bit values of the input	Follows data type	-	Lower eight bit values of the input.
	byHighByte	Upper eight bit values of the input	Follows data type	-	Upper eight bit values of the input.
	xBit0 to xBit15	Individual bit value of the input	Follows data type	-	Individual bit value of the input.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
wValue	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
byLowByte	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
byHighByte	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
xBit0 to xBit15	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**
  - Outputs the value of each bit of wValue, as well as the values of the upper and lower eight bits.
  - Only WORD type is supported for input; other types will be forcibly converted.
- **Program example**

	LD	ST
Define variables	VAR UnpackWord_Instance_0: UnpackWord; wValue: WORD; byLowByte: BYTE; byHighByte: BYTE; xBit0: BOOL; xBit1: BOOL; xBit2: BOOL; xBit3: BOOL; xBit4: BOOL; xBit5: BOOL; xBit6: BOOL; xBit7: BOOL; xBit8: BOOL; xBit9: BOOL; xBit10: BOOL; xBit11: BOOL; xBit12: BOOL; xBit13: BOOL; xBit14: BOOL; xBit15: BOOL; END VAR		
Program		UnpackWord_Instance_0(wValue 120 := wValue 2#0000000001111000 , byLowByte 120 => byLowByte 2#01111000 , byHighByte 0 => byHighByte 2#00000000 , xBit0 FALSE => xBit0 FALSE , xBit1 FALSE => xBit1 FALSE , xBit2 FALSE => xBit2 FALSE , xBit3 TRUE => xBit3 TRUE , xBit4 TRUE => xBit4 TRUE , xBit5 TRUE => xBit5 TRUE , xBit6 TRUE => xBit6 TRUE , xBit7 FALSE => xBit7 FALSE , xBit8 FALSE => xBit8 FALSE , xBit9 FALSE => xBit9 FALSE , xBit10 FALSE => xBit10 FALSE , xBit11 FALSE => xBit11 FALSE , xBit12 FALSE => xBit12 FALSE , xBit13 FALSE => xBit13 FALSE , xBit14 FALSE => xBit14 FALSE , xBit15 FALSE => xBit15 FALSE);RETURN

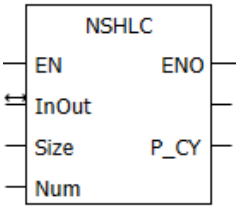
	LD	ST
Operation result		

Name	Type	Address	Value
UnpackWord_Instance_0	UnpackWord		
wValue	WORD		2#0000000001111000
byLowByte	BYTE		2#01111000
byHighByte	BYTE		2#00000000
xBit0	BOOL		FALSE
xBit1	BOOL		FALSE
xBit2	BOOL		FALSE
xBit3	BOOL		TRUE
xBit4	BOOL		TRUE
xBit5	BOOL		TRUE
xBit6	BOOL		TRUE
xBit7	BOOL		FALSE
xBit8	BOOL		FALSE
xBit9	BOOL		FALSE
xBit10	BOOL		FALSE
xBit11	BOOL		FALSE
xBit12	BOOL		FALSE
xBit13	BOOL		FALSE
xBit14	BOOL		FALSE
xBit15	BOOL		FALSE

18.2.4.14 NSHLC

Shifts the input array along with the carry flag (CY) to the left (toward higher bits) by a specified number of bits.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
NSHLC	Shift N-bits left with carry	FC		:= NSHLC(InOut := , Size := , Num := , P_CY =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Size	Number of bits in the shift object	Follows data type	1	Number of bits in the shift object
	Num	Number of bits to shift	Follows data type	1	Number of bits to shift
Input/Output	InOut[ ] array	Array where the shift object is located	Follows data type	-	Array where the shift object is located
Output	P_CY	Carry flag	Follows data type	-	Carry flag

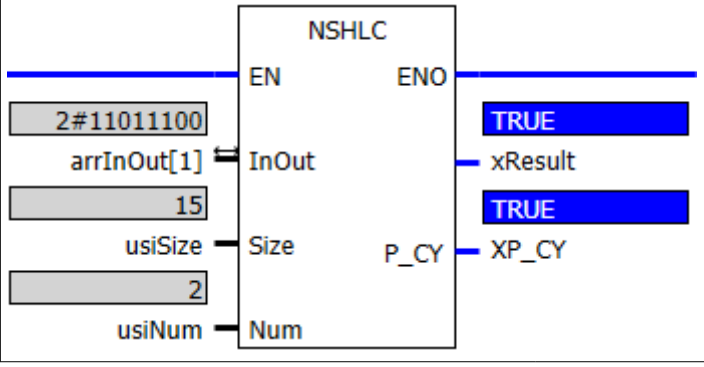
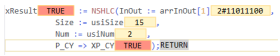
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
InOut[] array	√	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
P_CY	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - Shifts Size bits in InOut[0] starting from InOut[0] left by Num bits. The last overflow bit is output to the carry flag (CY) and the vacated bits on the opposite side are filled with 0.
  - The shift direction is left, from the least-significant bits to the most-significant bits.
- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	arrInOut: ARRAY[0..4] OF BYTE := [5(2#10110111)];
	usiSize: USINT := 15;
	usiNum: USINT := 2;
	XP_CY: BOOL;
	END_VAR	

Program		

	LD	ST

Operation result

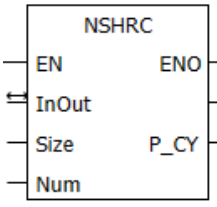
Name	Data type	Address	Value
xResult	BOOL		TRUE
arrInOut	ARRAY [0..4] OF BYTE		
arrInOut[0]	BYTE		2#10110111
arrInOut[1]	BYTE		2#11011100
arrInOut[2]	BYTE		2#11011110
arrInOut[3]	BYTE		2#10110111
arrInOut[4]	BYTE		2#10110111
usiSize	USINT		15
usiNum	USINT		2
XP_CY	BOOL		TRUE

- **Precautions**
  - When the value of Num or Size is 0, the shift operation is not performed. The function returns TRUE, InOut[] does not change, and the value of P_CY is FALSE.
  - When the value of Num is greater than or equal to Size, all Size bits (starting from bit 0 of InOut[0]) are set to FALSE. Additionally, the value of P_CY changes to FALSE, and the function returns TRUE.
  - When the value of Size exceeds the total number of bits in the InOut[] array, the function returns FALSE, InOut[] does not change, and the value of P_CY is FALSE.
  - When the input array has more than three dimensions, the function returns FALSE, InOut[ ] remains unchanged, and the value of P_CY is FALSE.

18.2.4.15 NSHRC

Shifts the input array along with the carry flag (CY) to the right (toward lower bits) by a specified number of bits.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
NSHRC	Shift N-bits right with carry	FC		:= NSHRC(InOut := , Size := , Num := , P_CY =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Size	Number of bits in the shift object	Follows data type	1	Number of bits in the shift object
	Num	Number of bits to shift	Follows data type	1	Number of bits to shift

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input/Output	InOut[ ] array	Array where the shift object is located	Follows data type	-	Array where the shift object is located
Output	P_CY	Carry flag	Follows data type	-	Carry flag

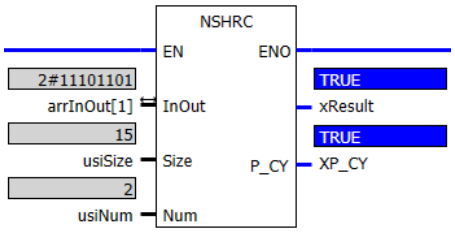
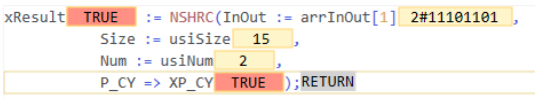
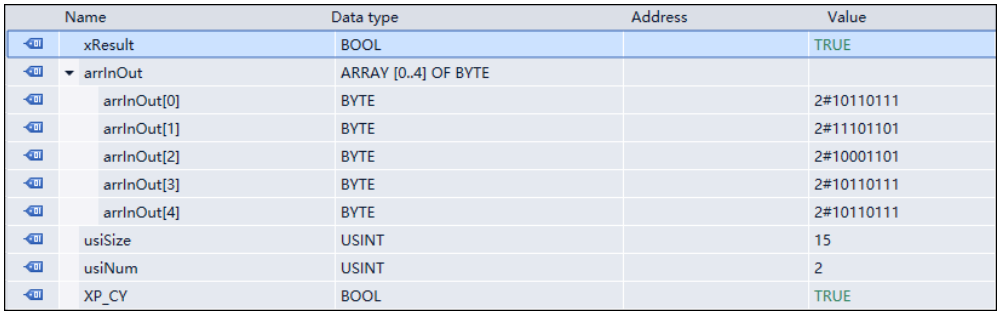
Variable types

	Bool ean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
InOut[ ] array	√	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
P_CY	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - Shifts Size bits in InOut[0] starting from InOut[0] left by Num bits. The last overflow bit is output to the carry flag (CY) and the vacated bits on the opposite side are filled with 0.
  - The shift direction is right, from the most-significant bits to the least-significant bits.
- **Program example**

	LD	ST

Define variables	VAR
	xResult: BOOL;
	arrInOut: ARRAY[0..4] OF BYTE := [5(2#10110111)];
	usiSize: USINT := 15;
	usiNum: USINT := 2;
	XP_CY: BOOL;
	END_VAR	

	LD	ST
Program		
Operation result		

Name	Data type	Address	Value
xResult	BOOL		TRUE
arrInOut	ARRAY [0..4] OF BYTE		
arrInOut[0]	BYTE		2#10110111
arrInOut[1]	BYTE		2#11101101
arrInOut[2]	BYTE		2#10001101
arrInOut[3]	BYTE		2#10110111
arrInOut[4]	BYTE		2#10110111
usiSize	USINT		15
usiNum	USINT		2
XP_CY	BOOL		TRUE

- **Precautions**
  - When the value of Num or Size is 0, the shift operation is not performed. The function returns TRUE, InOut[] does not change, and the value of P_CY is FALSE.
  - When the value of Num is greater than or equal to Size, all Size bits (starting from bit 0 of InOut[0]) are set to FALSE. Additionally, the value of P_CY changes to FALSE, and the function returns TRUE.
  - When the value of Size exceeds the total number of bits in the InOut[] array, the function returns FALSE, InOut[] does not change, and the value of P_CY is FALSE.
  - When the input array has more than three dimensions, the function returns FALSE, InOut[ ] remains unchanged, and the value of P_CY is FALSE.

18.2.5 String Conversion

18.2.5.1 Instruction List

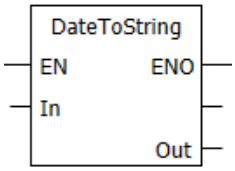
Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.5.2 DateToString	FC	Conversion from date to string	OMUtils	Library version 1.0.0.0 and later
18.2.5.3 DtToString	FC	Conversion from date and time to string		Library version 1.0.0.0 and later
18.2.5.4 Decoder	FC	Bit decoder		Library version 1.0.0.0 and later
18.2.5.5 Encoder	FC	Bit encoder		Library version 1.0.0.0 and later
18.2.5.6 HexStringToNum_DINT	FC	Conversion from hexadecimal string to integer		Library version 1.0.0.0 and later
18.2.5.7 HexStringToNum_INT	FC	Conversion from hexadecimal string to integer		Library version 1.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.5.8 HexStringToNum_LINT	FC	Conversion from hexadecimal string to long integer		Library version 1.0.0.0 and later
18.2.5.9 HexStringToNum_SINT	FC	Conversion from hexadecimal string to shorter integer		Library version 1.0.0.0 and later
18.2.5.10 HexStringToNum_UDINT	FC	Conversion from hexadecimal string to unsigned double integer		Library version 1.0.0.0 and later
18.2.5.11 HexStringToNum_UINT	FC	Conversion from hexadecimal string to unsigned integer		Library version 1.0.0.0 and later
18.2.5.12 HexStringToNum_ULINT	FC	Conversion from hexadecimal string to unsigned long integer		Library version 1.0.0.0 and later
18.2.5.13 HexStringToNum_USINT	FC	Conversion from hexadecimal string to unsigned shorter integer		Library version 1.0.0.0 and later
18.2.5.14 Moving Average	FC	Moving average calculation		Library version 1.0.0.0 and later
18.2.5.15 NumberToDecimalString	FC	Conversion from integer to fixed-length decimal string		Library version 1.0.0.0 and later
18.2.5.16 NumToHexString	FC	Conversion from integer to fixed-length hexadecimal string		Library version 1.0.0.0 and later
18.2.5.17 EnumToNum	FC	Conversion from enumeration to integer		Library version 1.0.0.0 and later
18.2.5.18 NumToEnum	FC	Conversion from integer to enumeration		Library version 1.0.0.0 and later
18.2.5.19 RoundUp	FC	Round up real number		Library version 1.0.0.0 and later
18.2.5.20 AnyBASE64	FC	BASE64 encoding	Util	Library version 1.3.0.0 and later
18.2.5.21 AnyNumFmt	FC	Numeric format conversion	OMUtils	Applicable to library version 1.5.0.0 and later

18.2.5.2 DateToString

Converts a date to a string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DateToString	Conversion from date to string	FC		:= DateToString(In := , Out =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Date	Follows data type	DATE#1970-1-1	Date
Output	Out	Conversion result	10 bytes	-	Conversion result

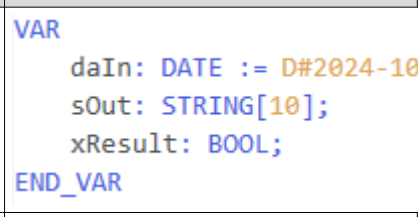
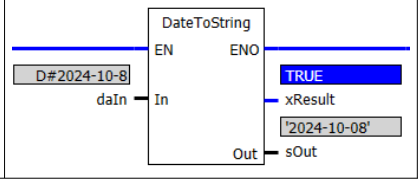
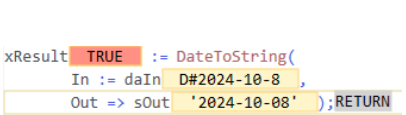
  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**

  Converts the date In to the string Out, appending NULL characters at the end of Out.

- **Program example**

	LD	ST
Define variables		
Program		

	LD	ST
Operation result		

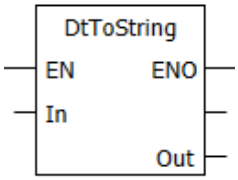
- **Precautions**

  The default type of the output variable Out is STRING[10], a string of length 10. If the defined length of the output string is less than 10, the output follows implicit conversion rules and may be truncated.

18.2.5.3 DtToString

Converts a date and time to a string.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
DtToString	Conversion from date and time to string	FC		:= DtToString(In := , Out =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Date and Time	Follows data type	DATE_AND_TIME#1970-1-1-0:0:0	Conversion object
Output	Out	Conversion result	19 bytes	-	Conversion result

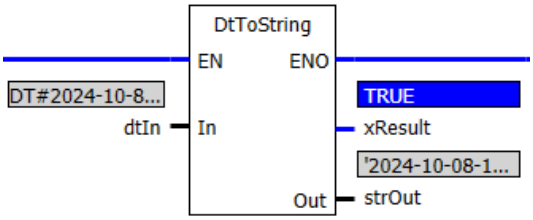
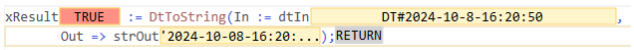
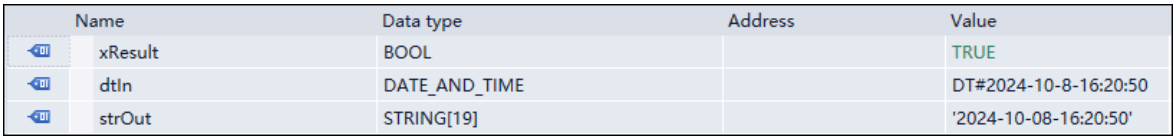
  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**

Converts the date and time In to the string Out. NULL characters are appended to the end of Out.

- **Program example**

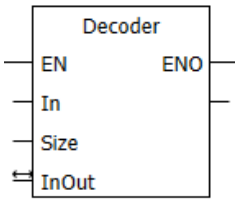
	LD	ST
Define variables	VAR xResult: BOOL; dtIn: DATE_AND_TIME:= DT#2024-10-08-16:20:50; strOut: STRING[19]; END_VAR	
Program		
Operation result		

Name	Data type	Address	Value
xResult	BOOL		TRUE
dtIn	DATE_AND_TIME		DT#2024-10-8-16:20:50
strOut	STRING[19]		'2024-10-08-16:20:50'

18.2.5.4 Decoder

Sets the specified bit to TRUE and the other bits to FALSE in an array element that consists of a maximum of 256 bits.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
Decoder	Bit decoder	FC		:= Decoder(In := , Size := , InOut :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Bit position to convert	Follows data type	0	Bit position to convert
	Size	Number of bits to convert	0 to 8	1	Number of bits to convert
Input/Output	InOut[ ] array	Array where the conversion object is located	Follows data type	-	Array where the conversion object is located

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
InOut[ ] array	√	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

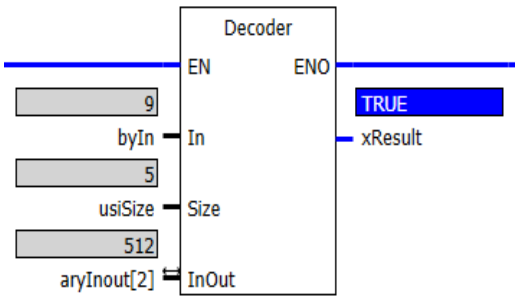
- **Function**

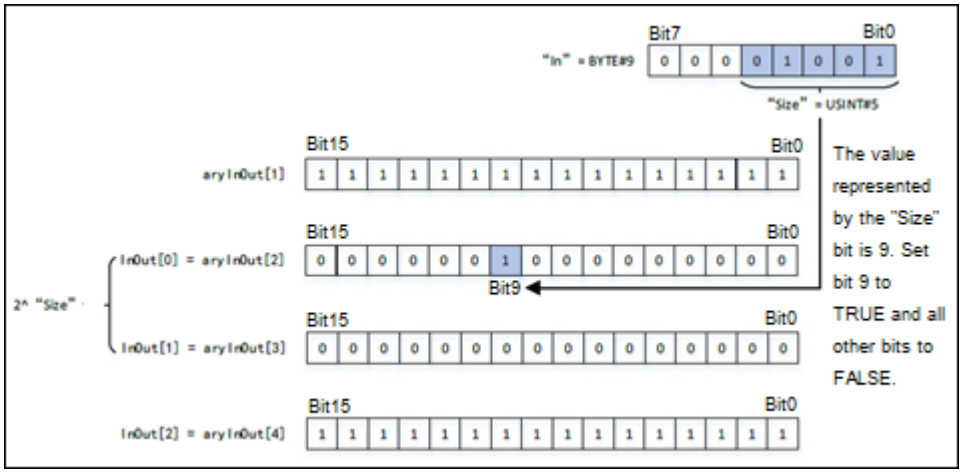
  This instruction converts array elements with 2^{Size} bits that start from InOut[0] in array InOut[]. It sets the specified bit to TRUE and other bits to FALSE. The position to be set to TRUE is specified by the conversion bit position In. Input parameters passed to InOut[ ] shall be specified along with the element index, such as Array[3].

- **Program example**

	LD	ST

Define variables	VAR
	byIn: BYTE := 9;
	usiSize: USINT := 5;
	aryInout: ARRAY [1..10] OF WORD := [10(65535)];
	xResult: BOOL;
	END_VAR	

Program		xResult TRUE := Decoder(In := byIn 9 , Size := usiSize 5 , InOut := aryInout[2] 512);RETURN

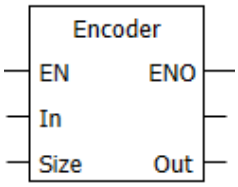
<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Operation result</td><td colspan="2">
<table>
<tr><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td>byIn</td><td>BYTE</td><td></td><td>9</td></tr>
<tr><td>usiSize</td><td>USINT</td><td></td><td>5</td></tr>
<tr><td>arylnout</td><td>ARRAY [1..10] OF WORD</td><td></td><td></td></tr>
<tr><td>arylnout[1]</td><td>WORD</td><td></td><td>65535</td></tr>
<tr><td>arylnout[2]</td><td>WORD</td><td></td><td>512</td></tr>
<tr><td>arylnout[3]</td><td>WORD</td><td></td><td>0</td></tr>
<tr><td>arylnout[4]</td><td>WORD</td><td></td><td>65535</td></tr>
<tr><td>arylnout[5]</td><td>WORD</td><td></td><td>65535</td></tr>
<tr><td>arylnout[6]</td><td>WORD</td><td></td><td>65535</td></tr>
<tr><td>arylnout[7]</td><td>WORD</td><td></td><td>65535</td></tr>
<tr><td>arylnout[8]</td><td>WORD</td><td></td><td>65535</td></tr>
<tr><td>arylnout[9]</td><td>WORD</td><td></td><td>65535</td></tr>
<tr><td>arylnout[10]</td><td>WORD</td><td></td><td>65535</td></tr>
<tr><td>xResult</td><td>BOOL</td><td></td><td>TRUE</td></tr>
</table>
</td></tr>
<tr><td>Principle analysis</td><td colspan="2">
</td></tr>
</table>

- **Precautions**
  - When the value of Size exceeds the number of bits in the array elements of InOut[ ], the return value is FALSE and InOut[ ] does not change.
  - When the value of Size exceeds the valid range, the return value is FALSE and InOut[ ] does not change.
  - When the value of Size is 0, all bits in InOut[0] change to FALSE.
  - When the number of bits in the InOut[ ] elements exceeds the number of bits represented by Size, the values of the remaining bits remain unchanged. For example, if Size = 2 and the InOut[ ] elements are of WORD type, the lower four bits of InOut[0] are processed, and the values of the remaining bits (bit 4 to bit 15) remain unchanged.
  - When the value of "IN" exceeds the number of bits represented by "Size", the actual bit position is determined by the remainder of "IN" divided by 2^{Size}. For example, if "IN" = 9 and Size = 2, the valid bit range is 4 bits and InOut[0] (9 mod 4 = 1) is set to 1.

18.2.5.5 Encoder

Calculates the position of the TRUE bit in array elements that consist of a maximum of 256 bits.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
Encoder	Bit encoder	FC		:= Encoder(In := , Size := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In[ ] array	Array where the conversion object is located	Follows data type	-	Array where the conversion object is located
	Size	Number of bits to convert	0 to 8	1	Number of bits to convert
Output	Out	Return position	Follows data type	-	Return position

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	√	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - Calculates the position of a TRUE bit in a range of bits in the array In[ ].
  - The range consists of 2^{Size} bits starting from In[0]. The position of the TRUE bit is stored in binary form in the lower Size bits of the output Out. The remaining higher bits of Out are set to FALSE.
  - If there are multiple TRUE bits in the specified range, the position of the highest TRUE bit is found.
  - Input parameters passed to In[ ] shall be specified along with the element index, such as Array[3].
- **Program example**

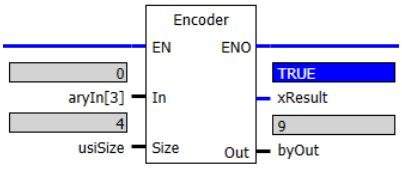
In the following example, Size is set to USINT#4 and the array In[ ] is of BYTE type. Since Size = USINT#4, the bit range for calculation consists of 16(2^4) bits starting from In[0].

In the figure below, the 9th bit in this range is TRUE. Therefore, the binary form of 9 (2#1001) is stored in the lower four bits of Out, while the upper four bits are set to FALSE.

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    aryIn: ARRAY [1..4] OF BYTE := [7,9,0,2];
    usiSize: USINT := 4;
    byOut: BYTE;
    xResult: BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



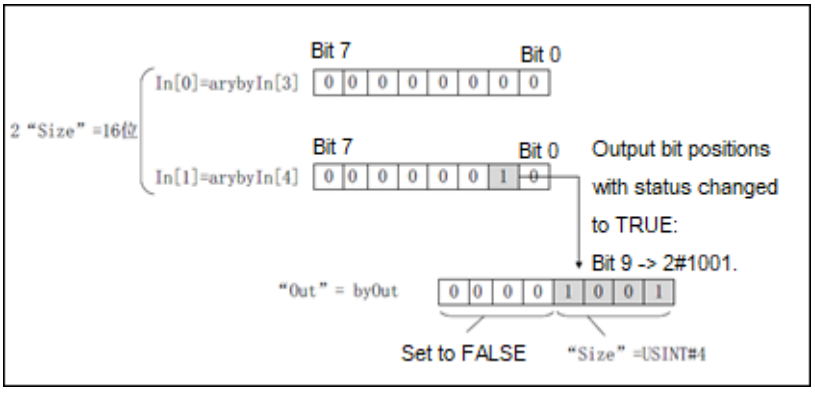
</td><td>

```
xResult TRUE := Encoder(In := aryIn[3] 0 ,
        Size := usiSize 4 ,
        Out => byOut 9 );RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

Name	Data type	Address	Value
aryIn	ARRAY [1..4] OF BYTE		
aryIn[1]	BYTE		7
aryIn[2]	BYTE		9
aryIn[3]	BYTE		0
aryIn[4]	BYTE		2
usiSize	USINT		4
byOut	BYTE		9
xResult	BOOL		TRUE

</td></tr>
<tr><td>Principle analysis</td><td colspan="2">



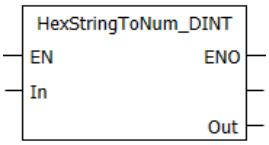
</td></tr>
</table>

- **Precautions**
  - When the value of Size is 0, the return value is TRUE, and all bits of Out are FALSE.
  - When the value of Size exceeds the valid range, a fault occurs, the return value is FALSE, and the value of Out is set to default.
  - When the value of 2^{Size} exceeds the number of bits in the In[ ] array elements, the return value is FALSE, and the value of Out is set to default.
  - When all bits in In[ ] that are specified by Size are FALSE, a fault occurs, the return value is FALSE, and the value of Out is set to default.

18.2.5.6 HexStringToNum_DINT

Converts a hexadecimal string to a double integer.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
HexStringToNum_DINT	Conversion from hexadecimal string to integer	FC		:= HexStringToNum_DINT(In := , Out =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Hexadecimal string	Follows data type	-	Hexadecimal string
Output	Out	Double integer	Follows data type	-	Double integer

  Variable types

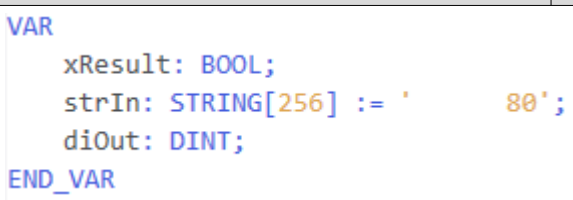
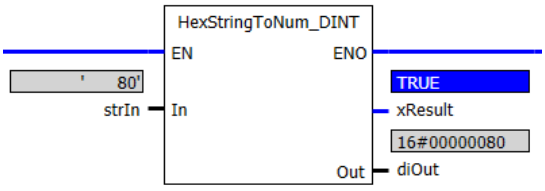
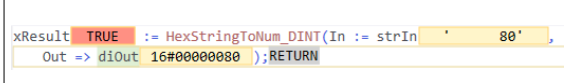
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
Out	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **Function**

  Converts the hexadecimal string In to the double integer Out.

  Half-width spaces (16#20) and '0' (16#30) in the upper digits, and underscores (16#5F) in the string are ignored.

- **Program example**

	LD	ST
Define variables		
Program		

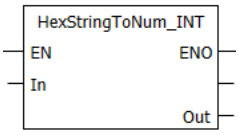
	LD	ST
Operation result		

- **Precautions**
  - If the conversion result exceeds the valid range of Out, no fault occurs, but the value of Out may not match expectations.
  - If In does not end with a NULL character, the conversion may fail.
  - If In includes characters that cannot be converted to numbers, a fault occurs, the return value is FALSE, and the value of Out remains unchanged.
  - If In contains only half-width spaces and underscores, the function returns FALSE, and Out remains unchanged.

18.2.5.7 HexStringToNum_INT

Converts a hexadecimal string to an integer.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
HexStringToNum_INT	Conversion from hexadecimal string to integer	FC		:= HexStringToNum_INT(In := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Hexadecimal string	Follows data type	-	Hexadecimal string
Output	Out	Integer	Follows data type	-	Integer

Variable types

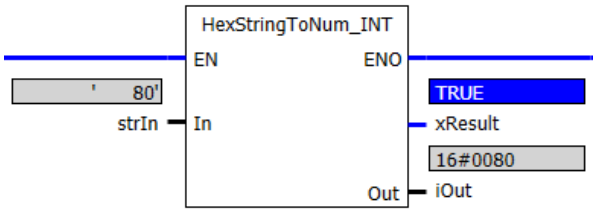
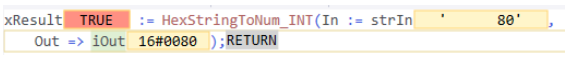
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
Out	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

- **Function**

  Converts the hexadecimal string In to the integer Out.

  Half-width spaces (16#20) and '0' (16#30) in the upper digits, and underscores (16#5F) in the string are ignored.

- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; strIn: STRING[256] := ' 80'; iOut: INT; END_VAR	
Program		
Operation result		

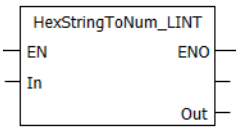
	Name	Data type	Address	Value
	xResult	BOOL		TRUE
	strIn	STRING[256]		' 80'
	iOut	INT		16#0080

- **Precautions**
  - If the conversion result exceeds the valid range of Out, no fault occurs, but the value of Out may not match expectations.
  - If In does not end with a NULL character, the conversion may fail.
  - If In includes characters that cannot be converted to numbers, a fault occurs, the return value is FALSE, and the value of Out remains unchanged.
  - If In contains only half-width spaces and underscores, the function returns FALSE, and Out remains unchanged.

18.2.5.8 HexStringToNum_LINT

Converts a hexadecimal string to a long integer.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
HexStringToNum_LINT	Conversion from hexadecimal string to long integer	FC		:= HexStringToNum_LINT(In := , Out =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Hexadecimal string	Follows data type	-	Hexadecimal string
Output	Out	Long integer	Follows data type	-	Long integer

  Variable types

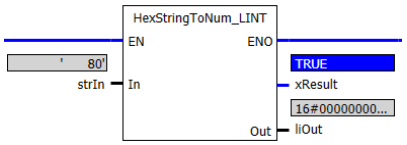
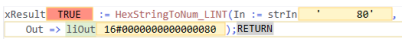
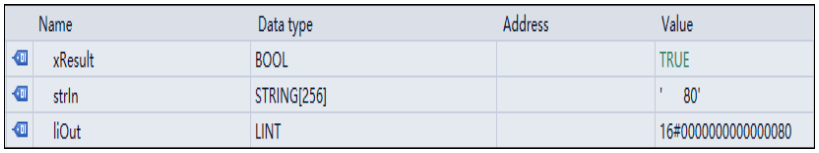
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
Out	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–

- **Function**

  Converts the hexadecimal format string In to the long integer Out.

  Half-width spaces (16#20) and '0' (16#30) in the upper digits, and underscores (16#5F) in the string are ignored.

- **Program example**

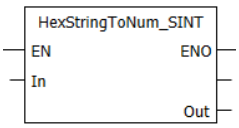
	LD	ST
Define variables	VAR xResult: BOOL; strIn: STRING[256] := ' 80'; liOut: LINT; END_VAR	
Program		
Operation result		

- **Precautions**
  - If the conversion result exceeds the valid range of Out, no fault occurs, but the value of Out may not match expectations.
  - If In does not end with a NULL character, the conversion may fail.
  - If In includes characters that cannot be converted to numbers, a fault occurs, the return value is FALSE, and the value of Out remains unchanged.
  - If In contains only half-width spaces and underscores, the function returns FALSE, and Out remains unchanged.

18.2.5.9 HexStringToNum_SINT

Converts a hexadecimal string to a short integer.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
HexStringToNum_SINT	Conversion from hexadecimal string to shorter integer	FC		:= HexStringToNum_SINT(In := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Hexadecimal string	Follows data type	-	Hexadecimal string
Output	Out	Short integer	Follows data type	-	Short integer

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
Out	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–

- **Function**

  Converts the hexadecimal format string In to the short integer Out.

  Half-width spaces (16#20) and '0' (16#30) in the upper digits, and underscores (16#5F) in the string are ignored.

- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; strIn: STRING[256] := ' 80'; siOut: SINT; END_VAR	
Program	HexStringToNum_SINT EN / ENO ' 80' → strIn In TRUE → xResult 16#80 → siOut Out	xResult TRUE := HexStringToNum_SINT(In := strIn ' 80', Out => siOut 16#80);RETURN
Operation result		

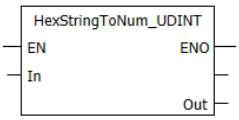
Name	Data type	Address	Value
xResult	BOOL		TRUE
strIn	STRING[256]		' 80'
siOut	SINT		16#80

- **Precautions**
  - If the conversion result exceeds the valid range of Out, no fault occurs, but the value of Out may not match expectations.
  - If In does not end with a NULL character, the conversion may fail.
  - If In includes characters that cannot be converted to numbers, a fault occurs, the return value is FALSE, and the value of Out remains unchanged.
  - If In contains only half-width spaces and underscores, the function returns FALSE, and Out remains unchanged.

18.2.5.10 HexStringToNum_UDINT

Converts a hexadecimal string into an unsigned double integer.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
HexStringToNum_UDINT	Conversion from hexadecimal string to unsigned double integer	FC		:= HexStringToNum_UDINT(In := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Hexadecimal string	Follows data type	-	Hexadecimal string
Output	Out	Unsigned double integer	Follows data type	-	Unsigned double integer

Variable types

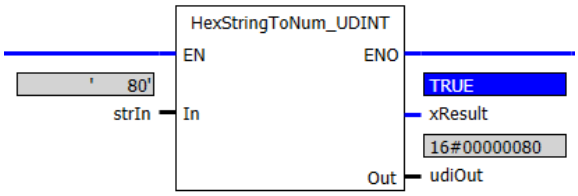
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
Out	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

Converts the hexadecimal string In to the unsigned double integer Out.

Half-width spaces (16#20) and '0' (16#30) in the upper digits, and underscores (16#5F) in the string are ignored.

- **Program example**

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">
<pre>
VAR
    xResult: BOOL;
    strIn: STRING[256] := '   80';
    udiOut: UDINT;
END_VAR
</pre>
</td></tr>
<tr><td>Program</td><td>


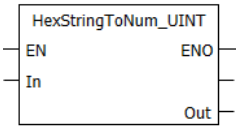
</td><td>
<pre>
xResult TRUE := HexStringToNum_UDINT(In := strIn ' 80',
    Out => udiOut 16#00000080);RETURN
</pre>
</td></tr>
<tr><td>Operation result</td><td colspan="2">
<table>
<tr><th></th><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td></td><td>xResult</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td></td><td>strIn</td><td>STRING[256]</td><td></td><td>' 80'</td></tr>
<tr><td></td><td>udiOut</td><td>UDINT</td><td></td><td>16#00000080</td></tr>
</table>
</td></tr>
</table>

- **Precautions**
  - If the conversion result exceeds the valid range of Out, no fault occurs, but the value of Out may not match expectations.
  - If In does not end with a NULL character, the conversion may fail.
  - If In includes characters that cannot be converted to numbers, a fault occurs, the return value is FALSE, and the value of Out remains unchanged.
  - If In contains only half-width spaces and underscores, the function returns FALSE, and Out remains unchanged.

18.2.5.11 HexStringToNum_UINT

Converts a hexadecimal string into an unsigned integer.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
HexStringToNum_UINT	Conversion from hexadecimal string to unsigned integer	FC		:= HexStringToNum_UINT(In := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Hexadecimal string	Follows data type	-	Hexadecimal string
Output	Out	Unsigned integer	Follows data type	-	Unsigned integer

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
Out	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Converts the hexadecimal string In to the unsigned integer Out.

  Half-width spaces (16#20) and '0' (16#30) in the upper digits, and underscores (16#5F) in the string are ignored.

- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; strIn: STRING[256] := ' 80'; uiOut: UINT; END_VAR	
Program	HexStringToNum_UINT EN / ENO ' 80' → strIn In TRUE → xResult 16#0080 → uiOut Out	xResult TRUE := HexStringToNum_UINT(In := strIn ' 80', Out => uiOut 16#0080);RETURN
Operation result		

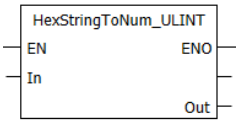
	Name	Data type	Address	Value
	xResult	BOOL		TRUE
	strIn	STRING[256]		' 80'
	uiOut	UINT		16#0080

- **Precautions**
  - If the conversion result exceeds the valid range of Out, no fault occurs, but the value of Out may not match expectations.
  - If In does not end with a NULL character, the conversion may fail.
  - If In includes characters that cannot be converted to numbers, a fault occurs, the return value is FALSE, and the value of Out remains unchanged.
  - If In contains only half-width spaces and underscores, the function returns FALSE, and Out remains unchanged.

18.2.5.12 HexStringToNum_ULINT

Converts a hexadecimal string into an unsigned long integer.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
HexStringToNum_ULINT	Conversion from hexadecimal string to unsigned long integer	FC		:= HexStringToNum_ULINT(In := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Hexadecimal string	Follows data type	-	Hexadecimal string
Output	Out	Unsigned long integer	Follows data type	-	Unsigned long integer

Variable types

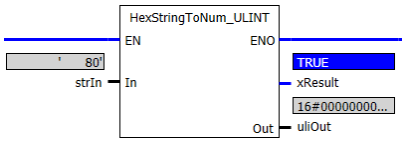
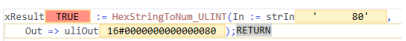
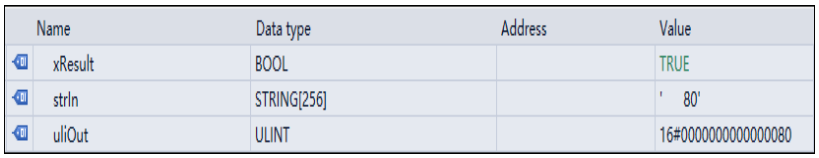
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
Out	–	–	–	–	–	–	–	–	√	_	–	–	–	–	–	–	–	–	–	–

- **Function**

Converts the hexadecimal string In to the unsigned long integer Out.

Half-width spaces (16#20) and '0' (16#30) in the upper digits, and underscores (16#5F) in the string are ignored.

- **Program example**

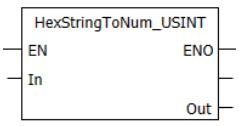
	LD	ST
Define variables	VAR xResult: BOOL; strIn: STRING[256] := ' 80'; uliOut: ULINT; END_VAR	
Program		
Operation result		

- **Precautions**
  - If the conversion result exceeds the valid range of Out, no fault occurs, but the value of Out may not match expectations.
  - If In does not end with a NULL character, the conversion may fail.
  - If In includes characters that cannot be converted to numbers, a fault occurs, the return value is FALSE, and the value of Out remains unchanged.
  - If In contains only half-width spaces and underscores, the function returns FALSE, and Out remains unchanged.

18.2.5.13 HexStringToNum_USINT

Converts a hexadecimal string into an unsigned short integer.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
HexStringToNum_USINT	Conversion from hexadecimal string to unsigned shorter integer	FC		:= HexStringToNum_USINT(In := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Hexadecimal string	Follows data type	-	Hexadecimal string
Output	Out	Unsigned short integer	Follows data type	-	Unsigned short integer

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
Out	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Converts the hexadecimal string In to the unsigned short integer Out.

  Half-width spaces (16#20) and '0' (16#30) in the upper digits, and underscores (16#5F) in the string are ignored.

- **Program example**

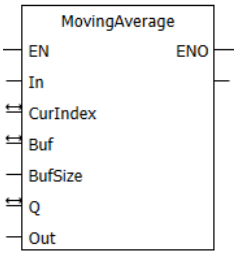
	LD	ST
Define variables	VAR xResult: BOOL; strIn: STRING[256] := ' 80'; usiOut: USINT; END_VAR	
Program	HexStringToNum_USINT EN / ENO ' 80' / strIn – In TRUE / xResult 16#80 / usiOut – Out	xResult TRUE := HexStringToNum_USINT(In := strIn ' 80', Out => usiOut 16#80);RETURN
Operation result	Name / Data type / Address / Value xResult / BOOL / / TRUE strIn / STRING[256] / / ' 80' usiOut / USINT / / 16#80	

- **Precautions**
  - If the conversion result exceeds the valid range of Out, no fault occurs, but the value of Out may not match expectations.
  - If In does not end with a NULL character, the conversion may fail.
  - If In includes characters that cannot be converted to numbers, a fault occurs, the return value is FALSE, and the value of Out remains unchanged.
  - If In contains only half-width spaces and underscores, the function returns FALSE, and Out remains unchanged.

18.2.5.14 MovingAverage

Calculates the moving average of the input.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
MovingAverage	Moving average calculation	FC		:= MovingAverage(In := , CurIndex := , Buf := , BufSize := , Q := , Out :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Input Value	Follow data type	-	Input value for average calculation
	BufSize	Maximum number of elements	Follows data type	1	Maximum number of elements for average calculation
	Out	Operation result	Follows data type	-	Operation result
Input/Output	CurInDex	Input value storage position	Follows data type	-	Position in Buf[ ] where In is stored
	Buf[ ] array	Input value storage array	Follows data type	-	Array where In is stored
	Q	Calculation completion flag	Follows data type	-	TRUE: The number of values stored in Buf[ ] ≥ BufSize FALSE: The number of values stored in Buf[ ] < BufSize

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
BufSize	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
CurInDex	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Buf[ ] array	Array with the same data type as elements of In																			
Q	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–

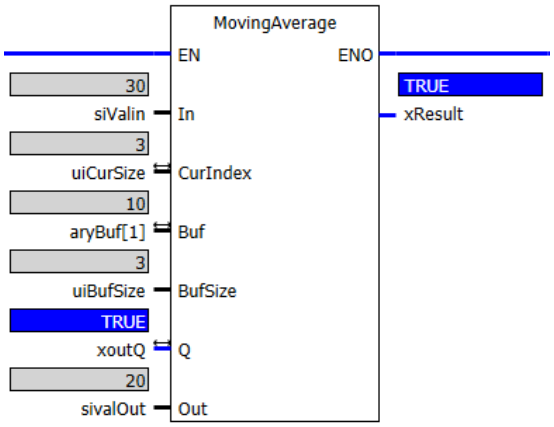
- **Function**

  Each time this instruction is executed, the input value In is stored in the array Buf[ ]. The average of the stored values is calculated and output to Out. The maximum number of elements used for the average calculation is specified by BufSize.

- **Program example**

  The following example sets BufSize to UINT#3. The instruction is executed three times, with input values In set to 10, 20, and 30 respectively.

	LD	ST
Define variables	VAR siValin: SINT; uiCurSize: UINT; aryBuf:ARRAY[1..4] OF SINT; uiBufSize: UINT := 3; xoutQ: BOOL; sivalOut: SINT; xResult: BOOL; END_VAR	

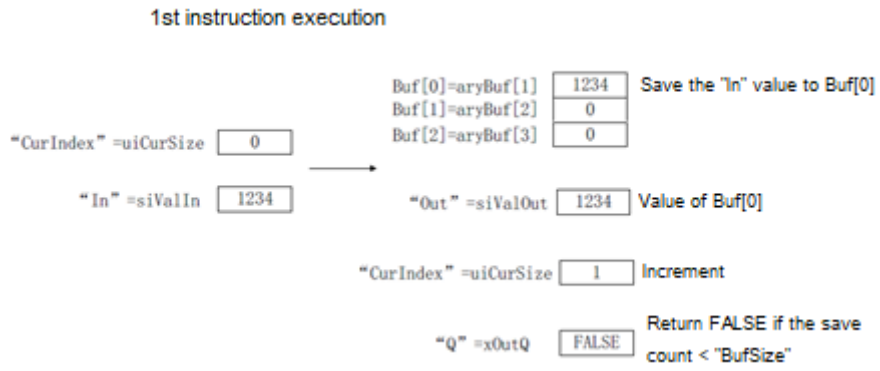
<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>
</td><td>xResult TRUE := MovingAverage(In := siValin 30 ,
CurIndex := uiCurSize 3 ,
Buf := aryBuf[1] 10 ,
BufSize := uiBufSize 3 ,
Q := xoutQ TRUE ,
Out := sivalOut 20);RETURN</td></tr>
<tr><td>Operation result</td><td colspan="2"><table>
<tr><th></th><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td></td><td>siValin</td><td>SINT</td><td></td><td>30</td></tr>
<tr><td></td><td>uiCurSize</td><td>UINT</td><td></td><td>3</td></tr>
<tr><td></td><td>aryBuf</td><td>ARRAY [1..4] OF SINT</td><td></td><td></td></tr>
<tr><td></td><td>aryBuf[1]</td><td>SINT</td><td></td><td>10</td></tr>
<tr><td></td><td>aryBuf[2]</td><td>SINT</td><td></td><td>20</td></tr>
<tr><td></td><td>aryBuf[3]</td><td>SINT</td><td></td><td>30</td></tr>
<tr><td></td><td>aryBuf[4]</td><td>SINT</td><td></td><td>0</td></tr>
<tr><td></td><td>uiBufSize</td><td>UINT</td><td></td><td>3</td></tr>
<tr><td></td><td>xoutQ</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td></td><td>sivalOut</td><td>SINT</td><td></td><td>20</td></tr>
<tr><td></td><td>xResult</td><td>BOOL</td><td></td><td>TRUE</td></tr>
</table></td></tr>
</table>

- Initial input value

  Set the input value storage position CurIndex to 0 and execute the instruction. After Buf[BufSize-1] is cleared, the input value In is stored in Buf[0].

  When the calculation completion flag "Q" is FALSE, the average is calculated based on the values from Buf[0] to Buf[CurIndex+1]. The result is stored in "Out", and "CurIndex" is incremented.

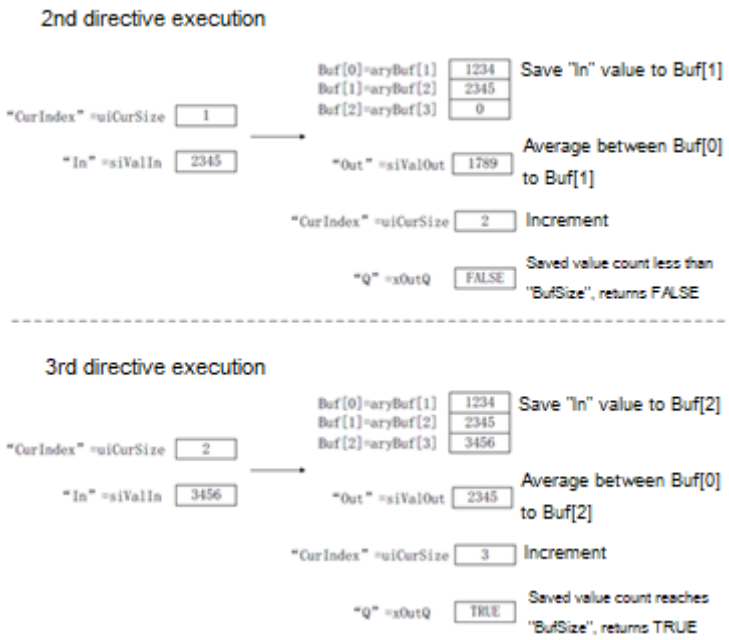
  If the number of values stored in the array Buf[ ] has not reached "BufSize", the value of "Q" remains FALSE.



- Value input up to the BufSize-th time

  Each time the instruction is executed, the input value In is stored in Buf[CurIndex]. The average is calculated based on the values from Buf[0] to Buf[CurIndex+1]. The result is stored in Out, and CurIndex is incremented.

  If the number of values stored in the array Buf[ ] reaches "BufSize", "Q" becomes TRUE.



- Value input after the BufSize-th time

  Each time the instruction is executed, Buf[0] to Buf[BufSize–1] are overwritten with the value of In cyclically. The average of Buf[0] to Buf[BufSize–1] is calculated and stored in Out.

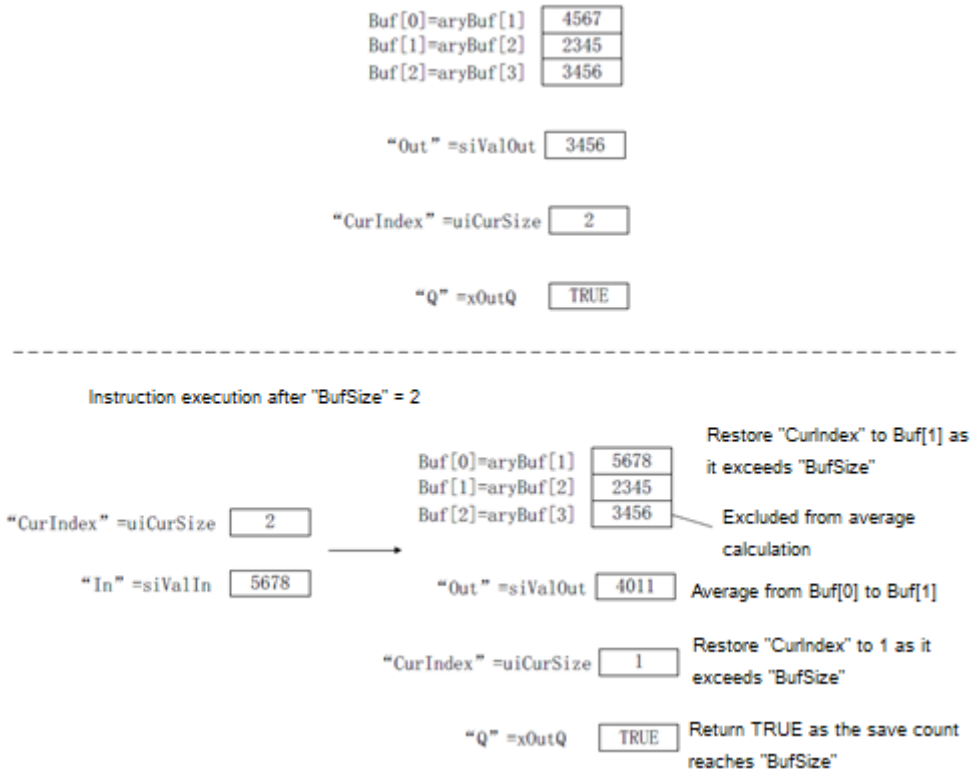
  If "CurIndex" reaches "BufSize", it resets to 1. It will increment again when new data is input, and "Q" remains TRUE.

- Initialization of stored values

  If the value of CurSize is set to 0 before the instruction is executed, values of Buf[0] to Buf[BufSize–1] change to 0. The current input value In is then stored in Buf[0]. "CurIndex" becomes 1, and "Q" becomes FALSE.

- Change of the BufSize value

  If the value of BufSize changes before the instruction is executed, the operation is performed with the new value of BufSize and the current value of CurIndex.



- **Precautions**
  - The data types of In, Out, and the elements of Buf[ ] must be consistent; otherwise, a compilation fault will occur.
  - The size of the array Buf[ ] shall be set to a value greater than BufSize. When the value of BufSize exceeds the size of Buf[ ], the return value is FALSE and Out outputs the default value.
  - When the calculation result exceeds the valid range of Out, the instruction does not trigger an exception, and an invalid value is stored in Out.
  - When the value of "BufSize" is 0, "Out" and "CurIndex" change to 0, and "Q" changes to TRUE.
  - When changing the value of BufSize, CurIndex must be set to 0 to initialize the stored values.

18.2.5.15 NumToDecString

Converts an integer to a decimal string with a fixed length.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
NumToDecString	Conversion from integer to fixed-length decimal string	FC	NumToDecString EN ENO In L Out Fill	:= NumToDecString(In := , L := , Fill := , Out =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Integer	Follows data type	-	Integer
	L	Number of characters	0 to 1985	1	String
	Fill	Character to fill	1 (_EFill_CHR._BLANK) 2 (_EFill_CHR._ZERO)	1 (_EFill_CHR._BLANK)	Character to fill
Input/Output	Out	String	Follows data type	-	String

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–
L	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Fill	See "Function description" for the enumerators for _EFill_CHR.																			
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**

  This instruction converts integer In to a decimal text string of UTF-8 alphanumeric characters. If In contains a negative value, a minus sign (–) is added at the beginning of the string.

  The instruction sets the number of characters in the output string Out to L. If the string length is insufficient, the high-order digits are filled with characters specified by Fill. If the number of characters in the conversion result exceeds L, L characters from the low-order digits of the conversion result are assigned to Out. The data type of Fill is the enumerated type _EFill_CHR.

  The meanings of the enumerators are as follows.

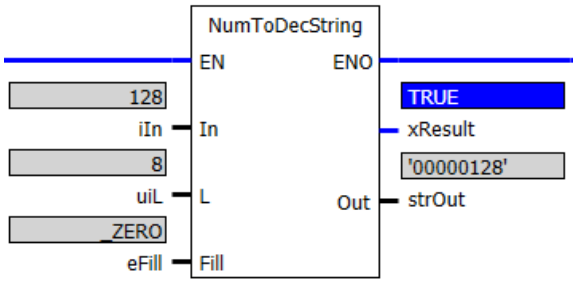
Enumerator	Definition
1 (_EFill_CHR._BLANK)	' ' (Half-width blank character)

Enumerator	Definition
2 (_EFill_CHR._ZERO)	'0'

- **Program example**

	LD	ST

Define variables	VAR
	iIn: INT := 128;
	uiL:UINT :=8;
	eFill:_eFILL_CHR := _EFILL_CHR._ZERO;
	strOut:STRING[256];
	xResult: BOOL;
	END_VAR	

Program		xResult TRUE := NumToDecString(In := iIn 128 , L := uiL 8 , Fill := eFill _ZERO , Out => strOut '00000128');RETURN
Operation result		

Name	Data type	Address	Value
iIn	INT		128
uiL	UINT		8
eFill	_EFILL_CHR		_ZERO
strOut	STRING[256]		'00000128'
xResult	BOOL		TRUE

- **Precautions**

  When the value of L is 0, the value of Out does not change. When the number of characters in the conversion result exceeds L, L characters from the low-order digits of the conversion result are stored in Out. The following is an example.

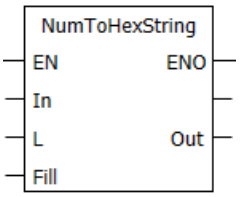
Value of In	Value of L	Value of Out
128	2	28

  - When the value of L exceeds the valid range, a fault occurs, the return value is FALSE, and the value of Out remains unchanged.
  - When the value of Fill exceeds the valid range, a compilation fault will occur.
  - When the conversion result exceeds the range of Out, the output value is incorrect.

18.2.5.16 NumToHexString

Converts an integer to a hexadecimal string with a fixed length.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
NumToHexString	Conversion from integer to fixed-length hexadecimal string	FC		:= NumToHexString(In := , L := , Fill := , Out =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Integer	Follows data type	-	Integer
	L	Number of characters	0 to 1985	1	String
	Fill	Character to fill	1 (_EFill_CHR._BLANK) 2 (_EFill_CHR._ZERO)	1 (_EFill_CHR._BLANK)	Character to fill
Input/Output	Out	String	Follows data type	-	String

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	√	√	√	√	√	√	√	√	–	–	–	–	–	–	–
L	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Fill	See "Function description" for the enumerators for _EFill_CHR.																			
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Function description**

  This instruction converts the integer In to a hexadecimal string of UTF-8 alphanumeric characters. If In is negative, it is expressed in its two's complement (bits inverted and then 1 added).

  The instruction sets the number of characters in the output string Out to L. If the string length is insufficient, the high-order digits are filled with characters specified by Fill. If the number of

characters in the conversion result exceeds L, L characters from the low-order digits of the conversion result are assigned to Out. The data type of Fill is the enumerated type _EFill_CHR.

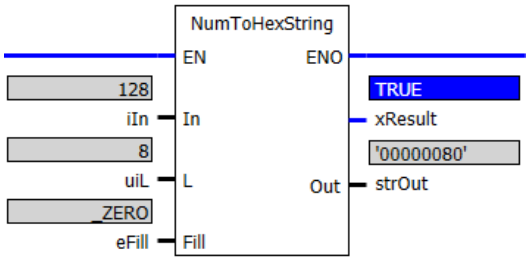
The meanings of the enumerators are as follows.

Enumerator	Definition
1 (_EFill_CHR._BLANK)	' ' (Half-width blank character)
2 (_EFill_CHR._ZERO)	'0'

- **Program example**

	LD	ST

Define variables	VAR
	iIn: INT := 128;
	uiL:UINT := 8;
	eFill:_eFILL_CHR := _EFILL_CHR._ZERO;
	strOut:STRING[256];
	xResult: BOOL;
	END_VAR	

Program		xResult TRUE := NumToHexString(In := iIn 128 , L := uiL 8 , Fill := eFill _ZERO , Out => strOut '00000080');RETURN
Operation result	Name / Data type / Address / Value iIn / INT / / 128 uiL / UINT / / 8 eFill / _EFILL_CHR / / _ZERO strOut / STRING[256] / / '00000080' xResult / BOOL / / TRUE	

- **Precautions**

When the value of L is 0, the value of Out does not change. When the number of characters in the conversion result exceeds the value of L, L characters from the lower characters of the conversion result are stored in Out. The following is an example.

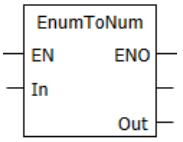
Value of In	Value of L	Value of Out
128	2	80

  - When the value of L exceeds the valid range, a fault occurs, the return value is FALSE, and the value of Out remains unchanged.
  - When the value of Fill exceeds the valid range, a compilation fault will occur.
  - When the conversion result exceeds the range of Out, the output value is incorrect.

18.2.5.17 EnumToNum

Converts the value of an enumeration type element to a double integer.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
EnumToNum	Conversion from enumeration to integer	FC		:= EnumToNum(In := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	Follows data type	-	Conversion object
Output	Out	Conversion result	Follows data type	-	Conversion result

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	Supports user-defined enumeration types. Take the enumeration elements of _eDAYOFWEEK as an example. Refer to the function description below for details.																			
Out	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **Function**

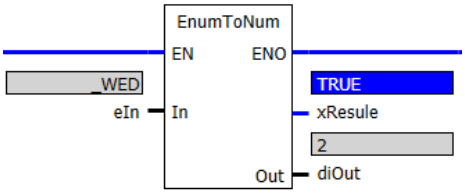
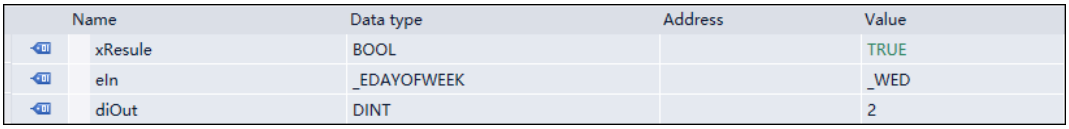
Converts the value of the enumeration type element In to a DINT type value and outputs the result to Out.

Enumeration type example: _EDAYOFWEEK

Enumerator	Definition
0 (_EDAYOFWEEK._MON)	Monday

Enumerator	Definition
1 (_EDAYOFWEEK._TUE)	Tuesday
2 (_EDAYOFWEEK._WED)	Wednesday
3 (_EDAYOFWEEK._THU)	Thursday
4 (_EDAYOFWEEK._FRI)	Friday
5 (_EDAYOFWEEK._SAT)	Saturday
6 (_EDAYOFWEEK._SUN)	Sunday

- **Program example**

	LD	ST
Define variables	VAR xResule: BOOL; eIn: _eDAYOFWEEK := _EDAYOFWEEK._WED; diOut: DINT; END_VAR	
Program		xResule TRUE := EnumToNum(In := eIn _WED , Out => diOut 2);RETURN
Operation result		

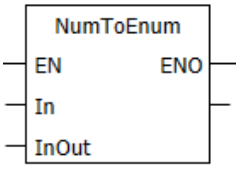
Name	Data type	Address	Value
xResule	BOOL		TRUE
eIn	_EDAYOFWEEK		_WED
diOut	DINT		2

- **Precautions**
  - When the value of In after conversion exceeds the DINT data range, a fault occurs. The return value is FALSE, and Out is initialized to 0.
  - Set the basic data type for In to BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, or LINT. Otherwise, the return value is FALSE and Out is initialized to 0.

18.2.5.18 NumToEnum

Converts a double integer to an enumerated type element.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
NumToEnum	Conversion from integer to enumeration	FC		:= NumToEnum(In := , InOut :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	Follows data type	-	Conversion object
Output	InOut	Conversion result	Follows data type	-	Conversion result

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
In Out	Supports user-defined enumeration types. Take the enumeration elements of _eDAYOFWEEK as an example. Refer to the function description below for details.																			

- **Function**

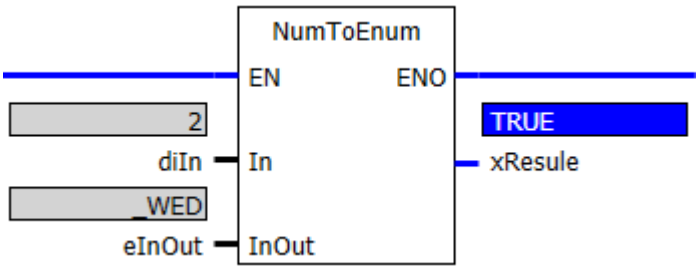
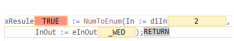
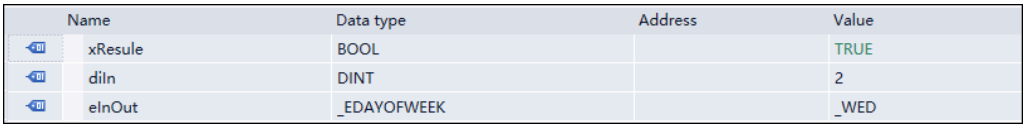
  Converts the double integer In to an enumeration value, and outputs the result to InOut.

  Enumeration type example: _EDAYOFWEEK

Enumerator	Definition
0 (_EDAYOFWEEK._MON)	Monday
1 (_EDAYOFWEEK._TUE)	Tuesday
2 (_EDAYOFWEEK._WED)	Wednesday
3 (_EDAYOFWEEK._THU)	Thursday
4 (_EDAYOFWEEK._FRI)	Friday
5 (_EDAYOFWEEK._SAT)	Saturday
6 (_EDAYOFWEEK._SUN)	Sunday

- **Program example**

	LD	ST
Define variables	VAR xResule: BOOL; diIn: DINT := 2; eInOut: _eDAYOFWEEK; END_VAR	

	LD	ST
Program		
Operation result		

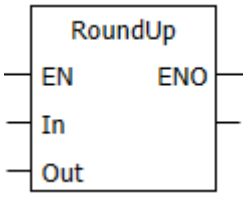
Name	Data type	Address	Value
xResule	BOOL		TRUE
diIn	DINT		2
eInOut	_EDAYOFWEEK		_WED

- **Precautions**
  - When the converted value of In exceeds the range of the enumeration InOut, a fault occurs. The return value is FALSE, and Out does not change.
  - Ensure that the converted value of In corresponds to one of the defined enumeration values of InOut; otherwise, unexpected results may occur. In such cases, the return value is TRUE, and Out is the numeric value converted from In.
  - Set the basic data type for the enumeration value of InOut to BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, or LINT. Otherwise, the return value is FALSE and Out does not change.

18.2.5.19 RoundUp

Converts a real number to an integer by rounding it up, away from zero.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
RoundUp	Round up real number	FC		:= RoundUp(In := , Out :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	Follows data type	-	Conversion object
	Out	Conversion result	Follows data type	-	Conversion result

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–	–	–

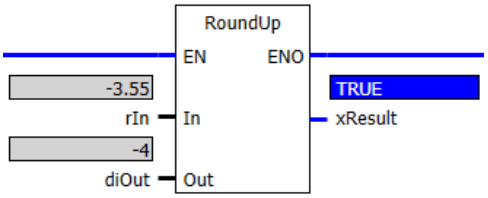
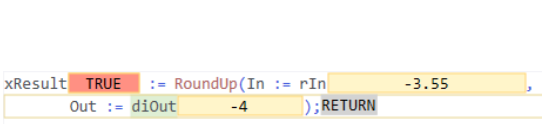
- **Function**

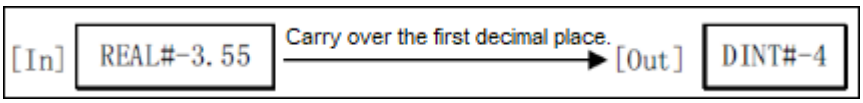
  Rounds the input real number In up to zero decimal places, and outputs the result to Out.

Input Value	Output Value
REAL#1.6	DINT#2
REAL#1.5	DINT#2
REAL#1.4	DINT#2
REAL#2.5	DINT#3
REAL#-1.6	DINT#-2
REAL#-1.5	DINT#-2
REAL#-1.4	DINT#-2
REAL#-2.5	DINT#-3

- **Program example**

  The following example sets In to REAL#-3.55, the output Out is DINT#-4, as shown below:

	LD	ST
Define variables	VAR xResult: BOOL; rIn: REAL := -3.55; diOut: DINT; END_VAR	
Program		

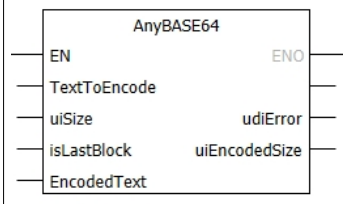
	LD	ST
Operation result		
Principle analysis		

- **Precautions**
  - When the conversion result exceeds the valid range of Out, the value of Out is incorrect and the function returns TRUE. When the value of In exceeds the floating-point precision range (6 valid digits for REAL and 15 valid digits for LREAL), precision loss will occur.
  - The data types of In and Out must match, otherwise, a program fault occurs and the function returns FALSE.
  - If the input or output is an element of an array with three or more dimensions, the function returns FALSE without performing the conversion.

18.2.5.20 AnyBASE64

AnyBASE64 is an encoding method for transmitting 8-bit byte code, which represents binary data based on 64 printable characters.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AnyBASE64	BASE64 encoding	FC		AnyBASE64(TextToEncode := , uiSize := , isLastBlock := , EncodedText := , udiError => , uiEncodedSize =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	TextToEncode	Data to be converted	Follows data type	-	Data to be processed. Supports arrays, array elements, and STRING types.
	uiSize	Conversion length	>= 0	0	Conversion length (number of elements).
	isLastBlock	Whether the current text is complete	TRUE, FALSE	FALSE	TRUE: The current text is complete. FALSE: The current text is only a part.

Input/ Output	Variable	Name	Valid Range	Default Value	Description
	EncodedText	Data after conversion	Follows data type	-	Data after conversion. Supports arrays, array elements, and STRING types.
Output	udiError	Fault Code	>=0	0	0: No fault 1: The text is too long. 2: The text character length is not an integer multiple of 3. 3: The output array length is less than 4/3 times the length of the input array to be converted. 4: The text pointer is null or the length uiSize = 0. 5: The conversion length exceeds the data length. 6: The input type for "TextToEncode" or "EncodedText" is an array with more than three dimensions.
	uiEncodedSize	Length of the converted text	>=0	0	Number of bytes in the converted text.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
TextToEncode	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
	You can also specify the entire array or a single element of the array.																				
uiSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-
isLastBlock	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
EncodedText	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
	You can also specify the entire array or a single element of the array.																				
udiError	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
uiEncodedSize	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function description**

  Specific encoding method:

Base64 requires converting every three 8-bit bytes into four 6-bit bytes (since 3 × 8 = 4 × 6 = 24). Then, two high-order zeros are padded to each 6-bit bytes to form four 8-bit bytes. Each byte corresponds to a number less than 64, which is the character index. The converted string will theoretically be one-third longer than the original.

Three English characters can be exactly converted into four Base64 characters. If the condition is not to meet, the remaining bytes of the source text are individually converted according to the encoding rules (1 byte → 2 Base64 chars, 2 bytes → 3 Base64 chars, missing bits are padded with zeros). The result is then filled to 4 bytes using equal signs (=), with at most two equal signs. If the remainder is 0, the original byte count is exactly a multiple of 3 (ideal case). If the remainder is 1, it is converted into two Base64 characters. To make the Base64 output length a multiple of 4, two equal signs are appended; similarly, if the remainder is 2, one equal sign is appended. If the length of uiSize is greater than the length of the original string, append the difference with character "A" to the converted result.

- When isLastBlock is TRUE and uiSize is 3, the encoding principle and results are shown in the table below:

Character	y	o	-
ASCII code	121	111	-
Binary bits	0 1 1 1 1 0 0 1	0 1 1 0 1 1 1 1	0 0 0 0 0 0 0 0
Encoding byte	00 011110	00 010110	00 111100	00 000000
Index number	30	22	60	0
Base64 encoding	e	W	8	=

- When isLastBlock is TRUE and uiSize is 1, the encoding principle and results are shown in the table below:

Character	y	-	-
ASCII code	121	-	-
Binary bits	0 1 1 1 1 0 0 1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
Encoding byte	00 011001	00 010000	00 000000	00 000000
Index number	25	16	0	0
Base64 encoding	Z	Q	=	=

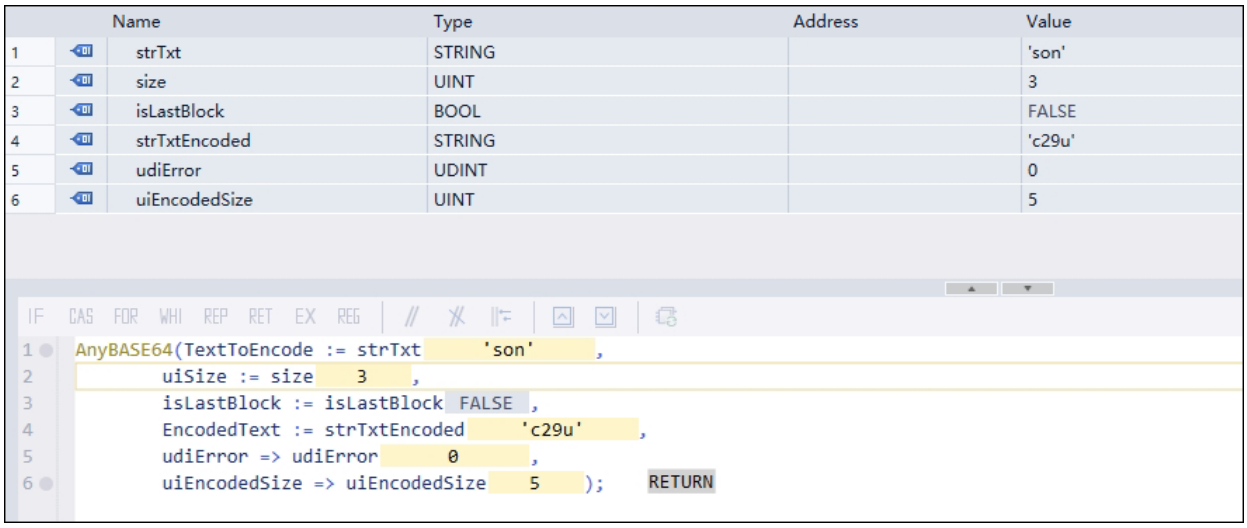
- When isLastBlock is TRUE and uiSize is 3, the encoding principle and results are shown in the table below:

Character	y	o	u
ASCII code	121	111	117
Binary bits	0 1 1 1 1 0 0 1	0 1 1 0 1 1 1 1	0 1 1 1 0 1 0 1
Encoding byte	00 011110	00 010110	00 111101	00 110101
Index number	30	22	61	53
Base64 encoding	e	W	9	1

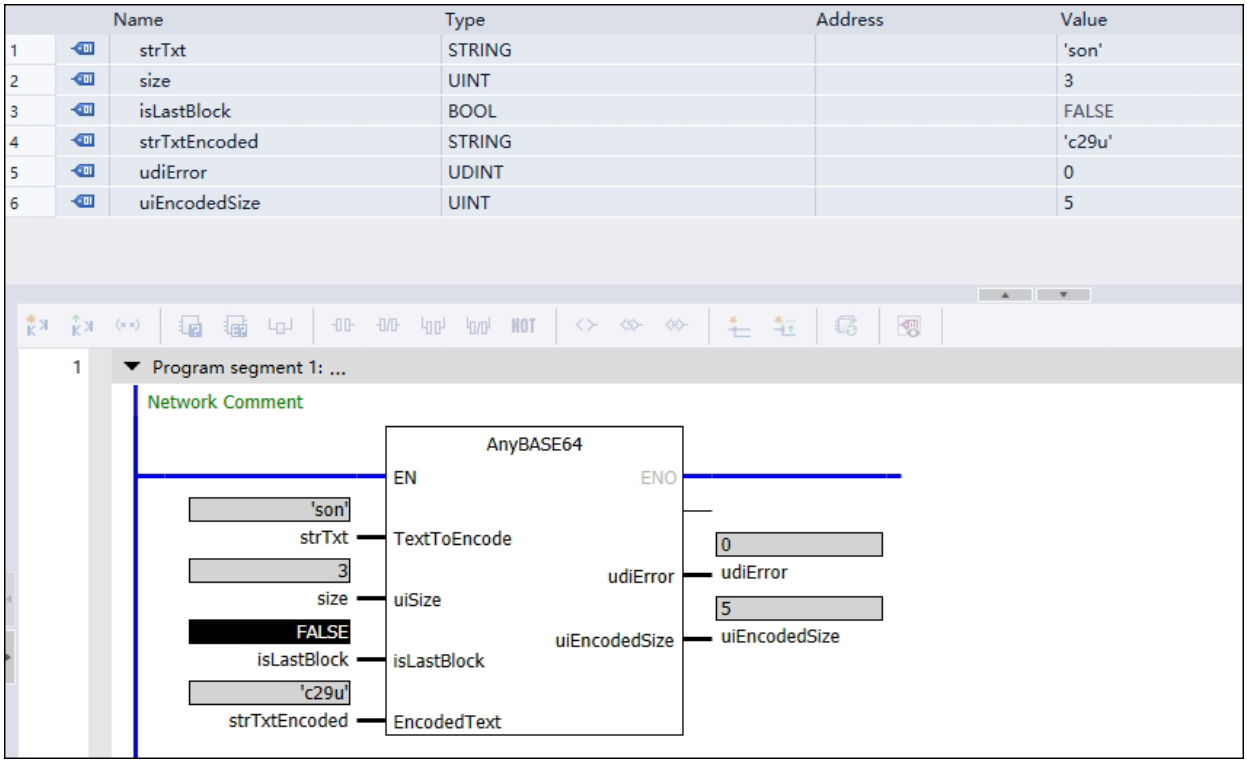
- **Program example**

  Set TextToEncode to 'son', and the encoded output data is 'c29u'.

  ST:



  LD:



◦ Encoding example 1: Input 'Son', uiSize = 3, isLastBlock = 1 → AnyBASE64 encoded result: 'U29u'.

ASCII character	S	o	n
Decimal	83	111	110
Binary	01010011	01101111	01101110

Each group consists of 6 bits	010100	110110	111101	101110
High-order padding with zeros	00010100	00110110	00111101	00101110
Corresponding Base64 index	20	54	61	46
Corresponding Base64 character	U	2	9	u

◦ Encoding example 2: Input 'Son', uiSize = 2, isLastBlock = 1 → AnyBASE64 encoded result: 'U28u'.

ASCII character	S	o	-
Decimal	83	111	-
Binary	01010011	01101111	-

Each group consists of 6 bits	010100	110110	111100	-
High-order padding with zeros	00010100	00110110	00111100	-

ASCII character	S	o	-
Corresponding Base64 index	20	54	60	-
Corresponding Base64 character	U	2	8	=

◦ Encoding example 3: Input 'Son', uiSize = 1, isLastBlock = 1 → AnyBASE64 encoded result: 'Uw==='.

ASCII character	S	-	-
Decimal	83	-	-
Binary	01010011	-	-
Each group consists of 6 bits	010100	110000	-	-
High-order padding with zeros	00010100	00110000	-	-
Corresponding Base64 index	20	48	-	-
Corresponding Base64 character	U	w	=	=

◦ Encoding example 4 (Chinese): For Base64 encoding of Chinese characters, use the String type under the UTF-8 format.
For example, input "你", uiSize = 3 → AnyBASE64 encoded result: '5L2g'.

ASCII character	你		
Decimal	228	189	160
Binary	11100100	10111101	10100000
Each group consists of 6 bits	111001	001011	110110	000000
High-order padding with zeros	00111001	00001011	00110110	00100000
Corresponding Base64 index	57	11	54	32
Corresponding Base64 character	5	L	2	g

◦ Fault description:
If any of the following errors occur, udiEncoded outputs 0 and the text result is incorrect. Normal usage should proceed when no ERROR is detected.

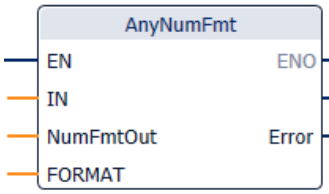
Condition	ERROR
TextToEncode is a null pointer.	4
uiSize = 0	4
uiSize > 32767 The length of input string is greater than 32767	1
isLastBlock = FALSE (only part of the text) and the length is not a multiple of 3.	2
The output array length is less than 4/3 times the length of the input array to be converted.	3
The conversion length exceeds the data length.	5
The input type for "TextToEncode" or "EncodedText" is an array with more than three dimensions.	6

- **Precautions**
  - The data length of EncodedText must be greater than 4/3 times the length of TextToEncode; otherwise, an error occurs with error code 3. If the conversion length uiSize exceeds the length of TextToEncode, an error occurs with error code 5.
  - When inputting array elements, conversion begins from the specified input array elements.

18.2.5.21 AnyNumFmt

Convert scientific notation to decimal or decimal to scientific notation.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AnyNumFmt	Numeric format conversion	FC		:= AnyNumFmt(IN := , NumFmtOut := , FORMAT := , Error =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Numeric string to be converted	Follows data type	-	Numeric string to be converted.
	NumFmtOut	Numeric string after conversion	Follows data type	-	Numeric string after conversion.

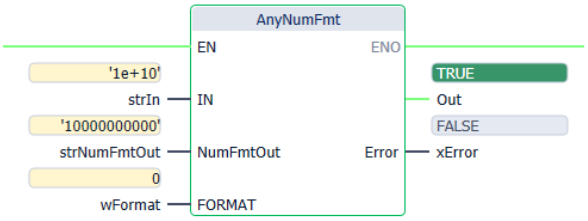
Input/Output	Variable	Name	Valid Range	Default Value	Description
	FORMAT	Conversion format	Follows data type	-	0: Converts scientific notation to decimal format. 1: Converts decimal format to scientific notation format.
Output	Error	Faulty status	Follows data type	-	Function execution status.

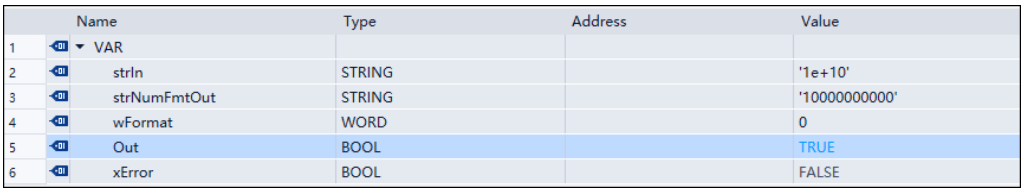
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
IN	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√
NumFmtOut	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√
FORMAT	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**
  - When FORMAT is 0, if "IN" is in decimal format, "NumFmtOut" remains consistent with "IN"; when "FORMAT" is 1, if "IN" is in scientific notation format, "NumFmtOut" remains consistent with "IN"; when "FORMAT" is any other value, the instruction does not execute.
  - If "IN" contains leading or trailing zeros, they will be removed in the converted string.
- **Program example**

Here is an example of converting scientific notation to decimal, as shown in the figure below:

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">
<pre>
VAR
    strIn: STRING := '1e+10';
    strNumFmtOut: STRING;
    wFormat: WORD;
    Out: BOOL;
    xError: BOOL;
END_VAR
</pre>
</td></tr>
<tr><td>Program</td><td>


</td><td>
<pre>
AnyNumFmt(IN := strIn   '1e+10'   ,
      NumFmtOut := strNumFmtOut '10000000000' ,
      FORMAT := wFormat   0   ,
      Error => xError FALSE);RETURN
</pre>
</td></tr>
<tr><td>Operation result</td><td colspan="2">


</td></tr>
</table>

- Precautions
  - Ensure that the data types of "IN" and NumFmtOut are consistent, otherwise a compilation error will occur.
  - The following situations will cause an error:
    - Output or output string length exceeds 255.
    - Input contains characters other than numbers, decimal points, and scientific notation indicators.
    - There are more than one decimal point, e|E, minus sign, or plus sign.
    - A decimal point, e|E, or plus sign appears at the beginning.
    - A minus sign appears not at the first position or immediately after e/E.
    - A plus sign appears not immediately after e/E.
    - The length of the converted string exceeds the size of NumFmtOut. (When converting scientific notation to a floating-point number, if the converted string size exceeds NumFmtOut due to leading zero padding, the system will not report an error, and NumFmtOut will output 0.)

18.2.6 Gray Code Conversion

18.2.6.1 Instruction List

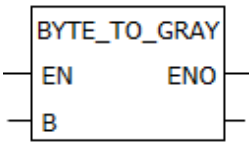
Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.6.2 BYTE_TO_GRAY	FC	Conversion from bit string to Gray code	Util	Library version 1.0.0.0 and later
18.2.6.3 WORD_TO_GRAY	FC	Conversion from bit string to Gray code		Library version 1.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.6.4 DWORD_TO_GRAY	FC	Conversion from bit string to Gray code		Library version 1.0.0.0 and later
18.2.6.5 GRAY_TO_BYTE	FC	Conversion from Gray code to bit string		Library version 1.0.0.0 and later
18.2.6.6 GRAY_TO_WORD	FC	Conversion from Gray code to bit string		Library version 1.0.0.0 and later
18.2.6.7 GRAY_TO_DWORD	FC	Conversion from Gray code to bit string		Library version 1.0.0.0 and later
18.2.6.8 BinToGray_BYTE	FC	Conversion from bit string to Gray code	OMUtils	Library version 1.0.0.0 and later
18.2.6.9 BinToGray_WORD	FC	Conversion from bit string to Gray code		Library version 1.0.0.0 and later
18.2.6.10 BinToGray_DWORD	FC	Conversion from bit string to Gray code		Library version 1.0.0.0 and later
18.2.6.11 BinToGray_LWORD	FC	Conversion from bit string to Gray code		Library version 1.0.0.0 and later

18.2.6.2 BYTE_TO_GRAY

Converts a bit string to a Gray code.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BYTE_TO_GRAY	Conversion from bit string to Gray code	FC		:= BYTE_TO_GRAY(B :=);

- **Related variables**

Variable

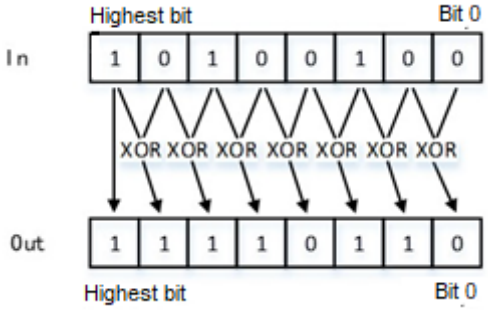
Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	Follows data type	-	Conversion object
Output	Out	Conversion result	Follows data type	-	Conversion result

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - Converts the bit string In to the Gray code Out.
  - The conversion steps are described below.

    1. The most significant bit (bit 7) of In is directly used as the most significant bit (bit 7) of Out.
    2. The XOR of the bit 7 and bit 6 of In is used as the bit 6 of Out.
    3. The same method is used to determine the values through the least significant bit (bit 0) of Out.



- **Program example**

	LD	ST
Define variables	VAR byIn: BYTE := 2#10100100; xResult: BYTE; END_VAR	
Program	BYTE_TO_GRAY EN ENO byIn — B — xResult	xResult := BYTE_TO_GRAY(B := byIn);
Operation result	Name / Data type / Address / Value byIn / BYTE / / 2#10100100 xResult / BYTE / / 2#11110110	

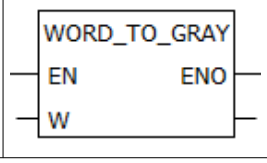
- **Precautions**

  The data types of "In" and "OUT" must be consistent; otherwise, an exception will occur during compilation.

18.2.6.3 WORD_TO_GRAY

Converts a bit string to a Gray code.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
WORD_TO_GRAY	Conversion from bit string to Gray code	FC		:= WORD_TO_GRAY(W :=);

- **Related variables**

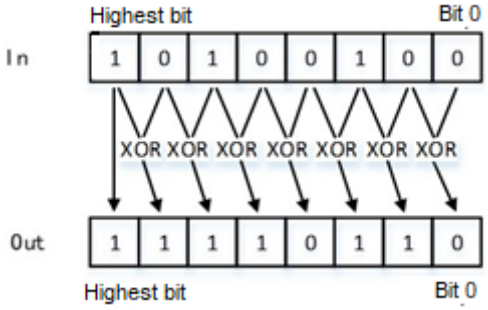
Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	Follows data type	-	Conversion object
Output	Out	Conversion result	Follows data type	-	Conversion result

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - Converts the bit string In to the Gray code Out.
  - The conversion steps are described below.
    1. The most significant bit (bit 7) of In is directly used as the most significant bit (bit 7) of Out.
    2. The XOR of the bit 7 and bit 6 of In is used as the bit 6 of Out.
    3. The same method is used to determine the values through the least significant bit (bit 0) of Out.



- **Program example**

	LD	ST
Define variables	VAR xResult: WORD; wIn: WORD := 2#0000000011110110; END_VAR	
Program	WORD_TO_GRAY EN ENO wIn — W — xResult	xResult := WORD_TO_GRAY(W := wIn);
Operation result	Name / Data type / Address / Value xResult / WORD / / 2#0000000010001101 wIn / WORD / / 2#0000000011110110	

- **Precautions**

  The data types of "In" and "OUT" must be consistent; otherwise, an exception will occur during compilation.

18.2.6.4 DWORD_TO_GRAY

Converts a bit string to a Gray code.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DWORD_TO_GRAY	Conversion from bit string to Gray code	FC	DWORD_TO_GRAY EN ENO X	:= DWORD_TO_GRAY(X :=);

- **Related variables**

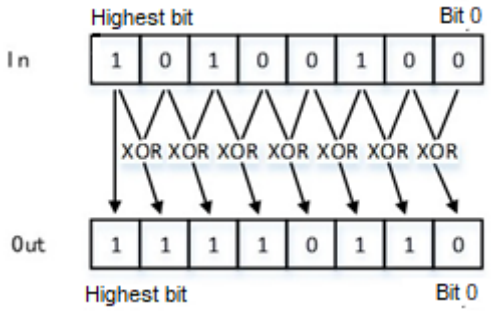
  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	Follows data type	-	Conversion object
Output	Out	Conversion result	Follows data type	-	Conversion result

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - Converts the bit string In to the Gray code Out.
  - The conversion steps are described below.
    1. The most significant bit (bit 7) of In is directly used as the most significant bit (bit 7) of Out.
    2. The XOR of the bit 7 and bit 6 of In is used as the bit 6 of Out.
    3. The same method is used to determine the values through the least significant bit (bit 0) of Out.



- **Program example**

	LD	ST
Define variables	VAR dwIn: DWORD := 2#00000000000000000000000010100100; xResult: DWORD; END_VAR	
Program	DWORD_TO_GRAY EN ENO dwIn – X – xResult	xResult := DWORD_TO_GRAY(X := dwIn);
Operation result	Name / Data type / Address / Value dwIn / DWORD / / 2#00000000000000000000000010100100 xResult / DWORD / / 2#00000000000000000000000011110110	

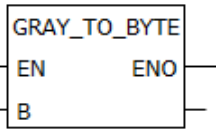
- **Precautions**

The data types of "In" and "OUT" must be consistent; otherwise, an exception will occur during compilation.

18.2.6.5 GRAY_TO_BYTE

Converts a Gray code to a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GRAY_TO_BYTE	Conversion from Gray code to bit string	FC		:= GRAY_TO_BYTE(B :=);

- **Related variables**

  Variable

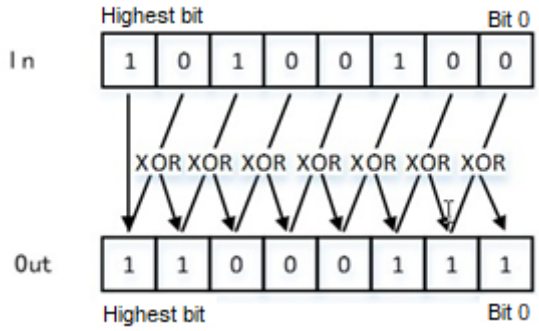
Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	Follows data type	-	Conversion object
Output	Out	Conversion result	Follows data type	-	Conversion result

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - Converts a Gray code to a bit string.
  - The conversion steps are described below.

    1. The most significant bit (bit 7) of In is directly used as the most significant bit (bit 7) of Out.
    2. The XOR of bit 6 of In and bit 7 of Out is used as the bit 6 of Out.
    3. The same method is used to determine the values through the least significant bit (bit 0) of Out.



- **Program example**

	LD	ST
Define variables	VAR xResult: BYTE; byIn: BYTE := 2#10100100; END_VAR	
Program	GRAY_TO_BYTE EN ENO byIn — B — xResult	xResult:= GRAY_TO_BYTE(B := byIn);
Operation result	Name / Data type / Address / Value xResult / BYTE / / 2#11000111 byIn / BYTE / / 2#10100100	

- **Precautions**

  The data types of "In" and "OUT" must be consistent; otherwise, an exception will occur during compilation.

18.2.6.6 GRAY_TO_WORD

Converts a Gray code to a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GRAY_TO_WORD	Conversion from Gray code to bit string	FC	GRAY_TO_WORD EN ENO W	:= GRAY_TO_WORD(W :=);

- **Related variables**

  Variable

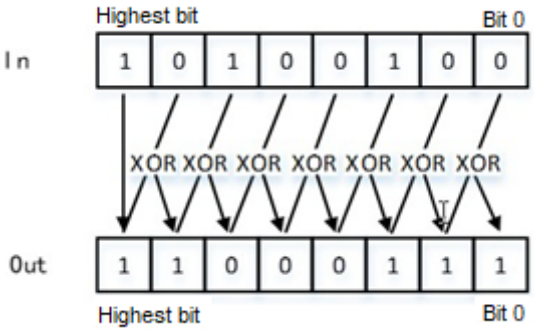
Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	Follows data type	-	Conversion object

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Conversion result	Follows data type	-	Conversion result

Variable types

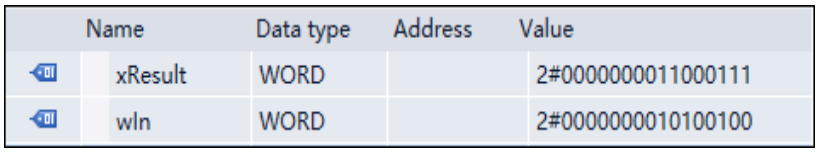
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - Converts a Gray code to a bit string.
  - The conversion steps are described below.
    1. The most significant bit (bit 7) of In is directly used as the most significant bit (bit 7) of Out.
    2. The XOR of bit 6 of In and bit 7 of Out is used as the bit 6 of Out.
    3. The same method is used to determine the values through the least significant bit (bit 0) of Out.



- **Program example**

	LD	ST
Define variables	VAR xResult: WORD; wIn: WORD := 2#0000000010100100; END_VAR	
Program	GRAY_TO_WORD EN ENO wIn W xResult	xResult := GRAY_TO_WORD(W := wIn);

	LD	ST
Operation result		

	Name	Data type	Address	Value
	xResult	WORD		2#0000000011000111
	wIn	WORD		2#0000000010100100

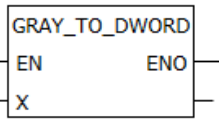
- **Precautions**

  The data types of "In" and "OUT" must be consistent; otherwise, an exception will occur during compilation.

18.2.6.7 GRAY_TO_DWORD

Converts a Gray code to a bit string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GRAY_TO_DWORD	Conversion from Gray code to bit string	FC		:= GRAY_TO_DWORD(X :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	Follows data type	-	Conversion object
Output	Out	Conversion result	Follows data type	-	Conversion result

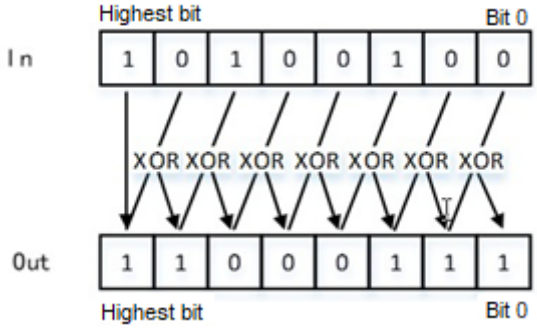
  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - Converts a Gray code to a bit string.

◦ The conversion steps are described below.

1. The most significant bit (bit 7) of In is directly used as the most significant bit (bit 7) of Out.
2. The XOR of bit 6 of In and bit 7 of Out is used as the bit 6 of Out.
3. The same method is used to determine the values through the least significant bit (bit 0) of Out.



- **Program example**

	LD	ST
Define variables	VAR xResult: DWORD; dwIn: DWORD := 2#00000000000000000000000010100100; END_VAR	
Program	GRAY_TO_DWORD EN ENO dwIn — X — xResult	xResult := GRAY_TO_DWORD(X := dwIn);
Operation result	Name / Data type / Address / Value xResult / DWORD / / 2#00000000000000000000000011000111 dwIn / DWORD / / 2#00000000000000000000000010100100	

- **Precautions**

  The data types of "In" and "OUT" must be consistent; otherwise, an exception will occur during compilation.

18.2.6.8 BinToGray_BYTE

Converts a bit string to a Gray code.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BinToGray_BYTE	Conversion from bit string to Gray code	FC	BinToGray_BYTE EN ENO In OUT	:= BinToGray_BYTE(In := , OUT =>);

- **Related variables**

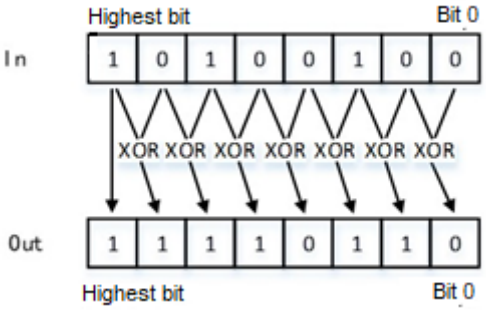
  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	Follows data type	-	Conversion object
Output	OUT	Conversion result	Follows data type	-	Conversion result

Variable types

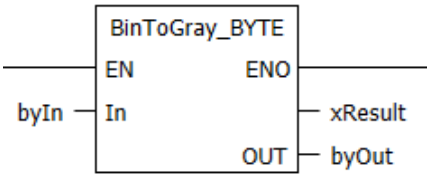
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction converts the bit string in the data to convert In to a gray code.
  - The instruction name is determined by the data types of "In" and "OUT".
  - The conversion steps are described below.
    1. The most significant bit (bit 7) of "In" is directly used as the most significant bit (bit 7) of "OUT".
    2. The XOR of the bit 7 and bit 6 of "In" is used as the bit 6 of "OUT".
    3. The same method is used to determine the values through the least significant bit (bit 0) of "OUT".



- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; byIn: BYTE:=2#10100100; byOut: BYTE; END_VAR	

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>
</td><td><pre>xResult := BinToGray_BYTE(In := byIn,
    OUT => byOut);</pre></td></tr>
<tr><td>Operation result</td><td colspan="2">
<table>
<tr><th></th><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td></td><td>xResult</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td></td><td>byIn</td><td>BYTE</td><td></td><td>2#10100100</td></tr>
<tr><td></td><td>byOut</td><td>BYTE</td><td></td><td>2#11110110</td></tr>
</table>
</td></tr>
</table>

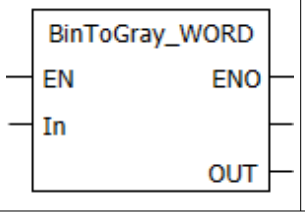
- **Precautions**

  The data types of "In" and "OUT" must be consistent; otherwise, an exception will occur during compilation.

18.2.6.9 BinToGray_WORD

Converts a bit string to a Gray code.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BinToGray_WORD	Conversion from bit string to Gray code	FC		:= BinToGray_WORD(In := , OUT =>);

- **Related variables**

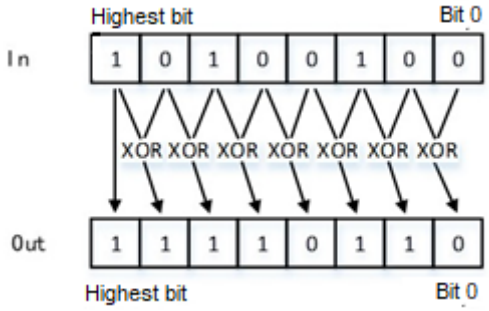
  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	Follows data type	-	Conversion object
Output	OUT	Conversion result	Follows data type	-	Conversion result

  Variable types

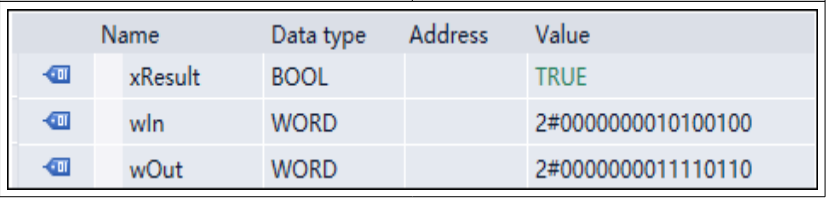
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	√	–		–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	√	–		–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction converts the bit string in the data to convert In to a gray code.
  - The instruction name is determined by the data types of "In" and "OUT".
  - The conversion steps are described below.
    1. The most significant bit (bit 7) of "In" is directly used as the most significant bit (bit 7) of "OUT".
    2. The XOR of the bit 7 and bit 6 of "In" is used as the bit 6 of "OUT".
    3. The same method is used to determine the values through the least significant bit (bit 0) of "OUT".



- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; wIn: WORD := 2#0000000010100100; wOut: WORD; END_VAR	
Program	BinToGray_WORD EN ENO wIn – In xResult OUT – wOut	xResult := BinToGray_WORD(In := wIn, OUT => wOut);

	LD	ST
Operation result		

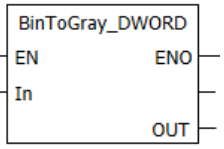
- **Precautions**

  The data types of "In" and "OUT" must be consistent; otherwise, an exception will occur during compilation.

18.2.6.10 BinToGray_DWORD

Converts a bit string to a Gray code.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BinToGray_DWORD	Conversion from bit string to Gray code	FC		:= BinToGray_DWORD(In := , OUT =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	Follows data type	-	Conversion object
Output	Out	Conversion result	Follows data type	-	Conversion result

  Variable types

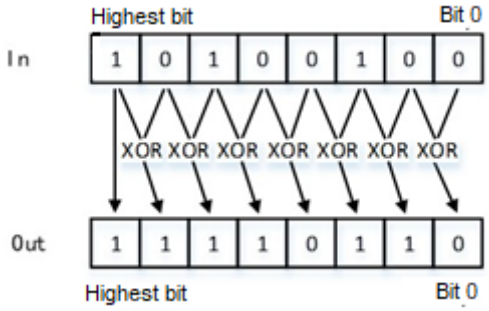
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction converts the bit string in the data to convert In to a gray code.

◦ The instruction name is determined by the data types of In and Out.

◦ The conversion steps are described below.

1. The most significant bit (bit 7) of In is directly used as the most significant bit (bit 7) of Out.
2. The XOR of the bit 7 and bit 6 of In is used as the bit 6 of Out.
3. The same method is used to determine the values through the least significant bit (bit 0) of Out.



- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; dwIn: DWORD := 2#00000000000000000000000010100100; dwOut: DWORD; END_VAR	
Program	BinToGray_DWORD EN ENO dwIn — In — xResult OUT — dwOut	xResult := BinToGray_DWORD(In := dwIn, OUT => dwOut);
Operation result	Name / Data type / Address / Value xResult / BOOL / / TRUE dwIn / DWORD / / 2#00000000000000000000000010100100 dwOut / DWORD / / 2#00000000000000000000000011110110	

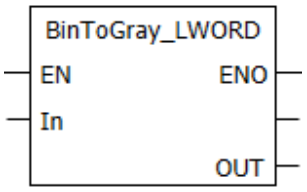
- **Precautions**

The data types of "In" and "OUT" must be consistent; otherwise, an exception will occur during compilation.

18.2.6.11 BinToGray_LWORD

Converts a bit string to a Gray code.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BinToGray_LWORD	Conversion from bit string to Gray code	FC		:= BinToGray_LWORD(In := , OUT =>);

- **Related variables**

  Variable

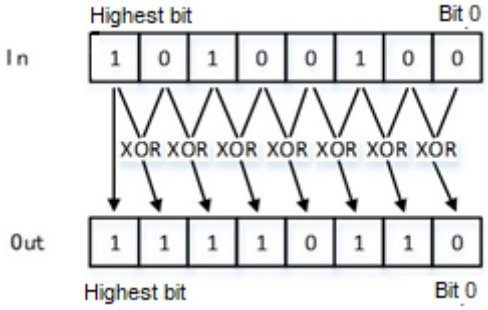
Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	Follows data type	-	Conversion object
Output	Out	Conversion result	Follows data type	-	Conversion result

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction converts the bit string in the data to convert In to a gray code.
  - The instruction name is determined by the data types of In and Out.
  - The conversion steps are described below.

    1. The most significant bit (bit 7) of In is directly used as the most significant bit (bit 7) of Out.
    2. The XOR of the bit 7 and bit 6 of In is used as the bit 6 of Out.
    3. The same method is used to determine the values through the least significant bit (bit 0) of Out.



- **Program example**

	LD	ST
Define variables	VAR xResult: BOOL; lwIn: LWORD := 2#0000000000000000000000000000000000000000000000000000000010100100; lwOut: LWORD; END_VAR	
Program	BinToGray_LWORD EN ENO lwIn — In — xResult OUT — lwOut	xResult := BinToGray_LWORD(In := lwIn, OUT => lwOut);
Operation result	Name / Data type / Address / Value xResult / BOOL / / TRUE lwIn / LWORD / / 2#0000000000000000000000000000000000000000000000000000000010100100 lwOut / LWORD / / 2#0000000000000000000000000000000000000000000000000000000011110110	

- **Precautions**

  The data types of "In" and "OUT" must be consistent; otherwise, an exception will occur during compilation.

18.2.7 Signal Processing

18.2.7.1 Instruction List

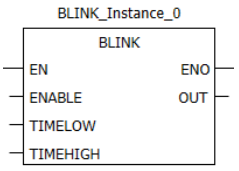
Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.7.2 BLINK	FB	Pulse signal generator	Util	Library version 1.0.0.0 and later
18.2.7.3 GEN	FB	Cyclic signal generator		Library version 1.0.0.0 and later
18.2.7.4 FREQ_MEASURE	FB	Frequency measurement		Library version 1.0.0.0 and later
18.2.7.5 GetClk_1ms	FC	1-ms timing pulse	OMUtils	Library version 1.2.0.0 and later
18.2.7.6 GetClk_10ms	FC	10-ms timing pulse		Library version 1.2.0.0 and later
18.2.7.7 GetClk_20ms	FC	20-ms timing pulse		Library version 1.2.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.7.8 GetClk_100ms	FC	100-ms timing pulse		Library version 1.2.0.0 and later
18.2.7.9 GetClk_1s	FC	1-s timing pulse		Library version 1.2.0.0 and later
18.2.7.10 GetClk_1min	FC	1-min timing pulse		Library version 1.2.0.0 and later

18.2.7.2 BLINK

BLINK is used as a pulse signal generator.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BLINK	Pulse signal generator	FB		BLINK_Instance_0(ENABLE := , TIMELOW := , TIMEHIGH := , OUT =>);

- **Related variable**

Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	ENABLE	Instruction enable	Follows data type	-	TRUE: The instruction starts working.
	TIMELOW	Low-level duration	Follows data type	-	Duration for remaining the low level for pulse signals
	TIMEHIGH	High-level duration	Follows data type	-	Duration for remaining the high level for pulse signals
Output	OUT	Pulse signal output	Follows data type	-	Output value of a pulse signal

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
ENABLE	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
TIMELOW	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
TIMEHIGH	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
OUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  If ENABLE is set to TRUE, this instruction sets the output to TRUE within the period specified by TIMEHIGH and then sets the output to FALSE in the period specified by TIMELOW.

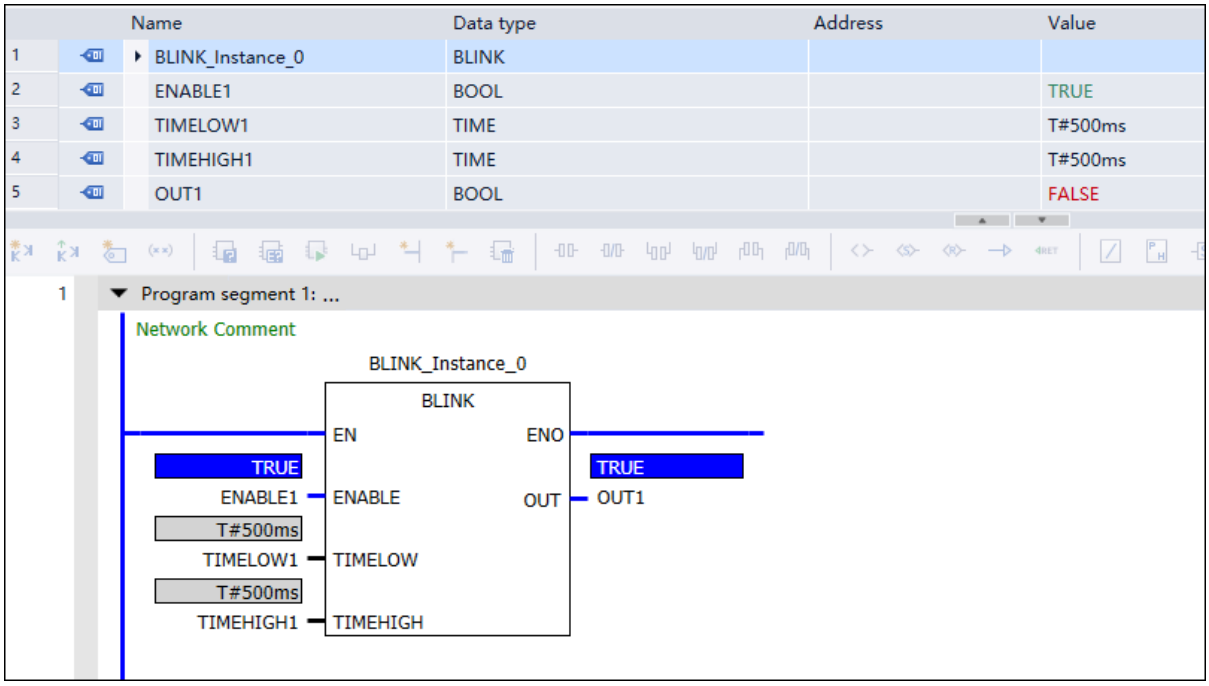
- **Program example**

  ST:

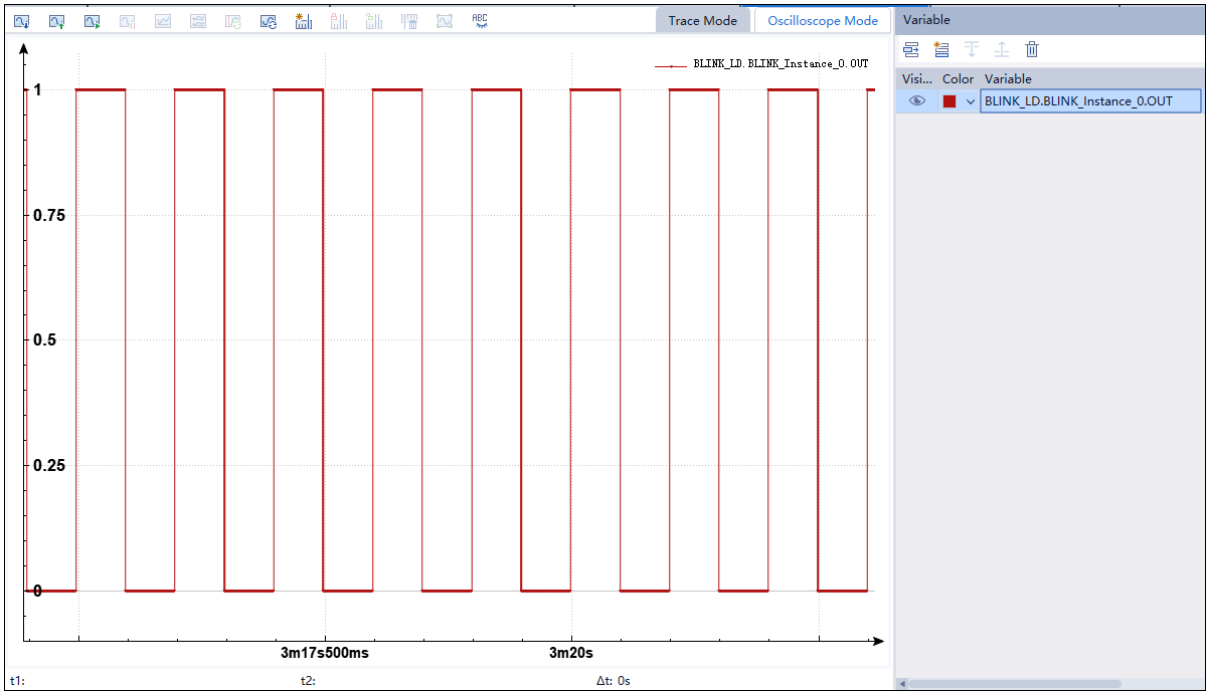
	Name	Data type	Address	Value
1	BLINK_Instance_0	BLINK		
2	ENABLE1	BOOL		TRUE
3	TIMELOW1	TIME		T#500ms
4	TIMEHIGH1	TIME		T#500ms
5	OUT1	BOOL		TRUE

```
BLINK_Instance_0(
ENABLE  TRUE  := ENABLE1  TRUE ,
TIMELOW  T#500ms  := TIMELOW1  T#500ms ,
TIMEHIGH  T#500ms  :=TIMEHIGH1  T#500ms ,
OUT  TRUE  => OUT1  TRUE );RETURN
```

  LD:



TRACE:



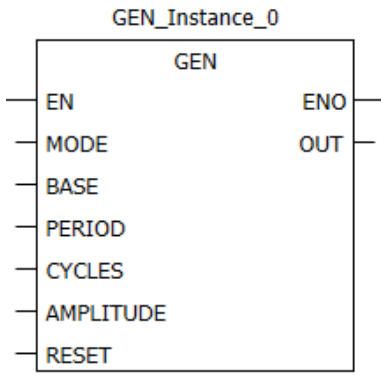
- **Precautions**

  When "ENABLE" is set to FALSE, the output "OUT" retains its last value.

18.2.7.3 GEN

Generates specified function waveforms based on the input parameters.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GEN	Cyclic signal generator	FB		GEN_Instance_0(MODE := , BASE := , PERIOD := , CYCLES := , AMPLITUDE := , RESET := , OUT =>);

- **Related variable**

  Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	MODE	Signal type	GEN_MODE	TRIANGLE	MODE is an enumerator of the GEN_MODE type, with the following enumeration values: 0: TRIANGLE, indicating triangular waves with the output range from -AMPLITUDE to +AMPLITUDE 1: TRIANGLE_POS, indicating triangular waves with the output range from 0 to +AMPLITUDE 2: SAWTOOTH_RISE, indicating rising sawtooth waves with the output range from -AMPLITUDE to +AMPLITUDE 3: SAWTOOTH_FALL, indicating falling sawtooth waves with the output range from +AMPLITUDE to -AMPLITUDE 4: RECTABGLE, indicating square waves with the output range from -AMPLITUDE to +AMPLITUDE 7: SINE, indicating sine waves 8: COSINE, indicating cosine waves
	BASE	Circulating mode	Follows data type	FALSE	TRUE: Outputs signals by the cycle period specified by PERIOD FALSE: Outputs signals by the cycle times specified by CYCLES

Input/ Output	Variable	Name	Valid Range	Default Value	Description
	PERIOD	Cycle period	Follows data type	TIME#1s 0ms	This variable is valid when BASE is TRUE, specifying the cycle period for output signals.
	CYCLES	Loop count	Follows data type	1000	This variable is valid when BASE is FALSE, specifying the cycle times for output signals.
	AMPLITUDE	Signal vibration amplitude	Follows data type	0	Amplitude of the function, determining the oscillation range of the curve
	RESET	Reset a bit	Follows data type	FALSE	TRUE: Resets the output signal to 0. FALSE: Outputs the signal normally.
Output	OUT	Signal output value	Follows data type	0	Output value of a periodic signal

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
MODE	GEN_MODE																			
BASE	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PERIOD	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
CYCLES	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
AMPLITUDE	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

- **Function**

  Generates specified function waveforms based on the input parameters.

- **Program example**

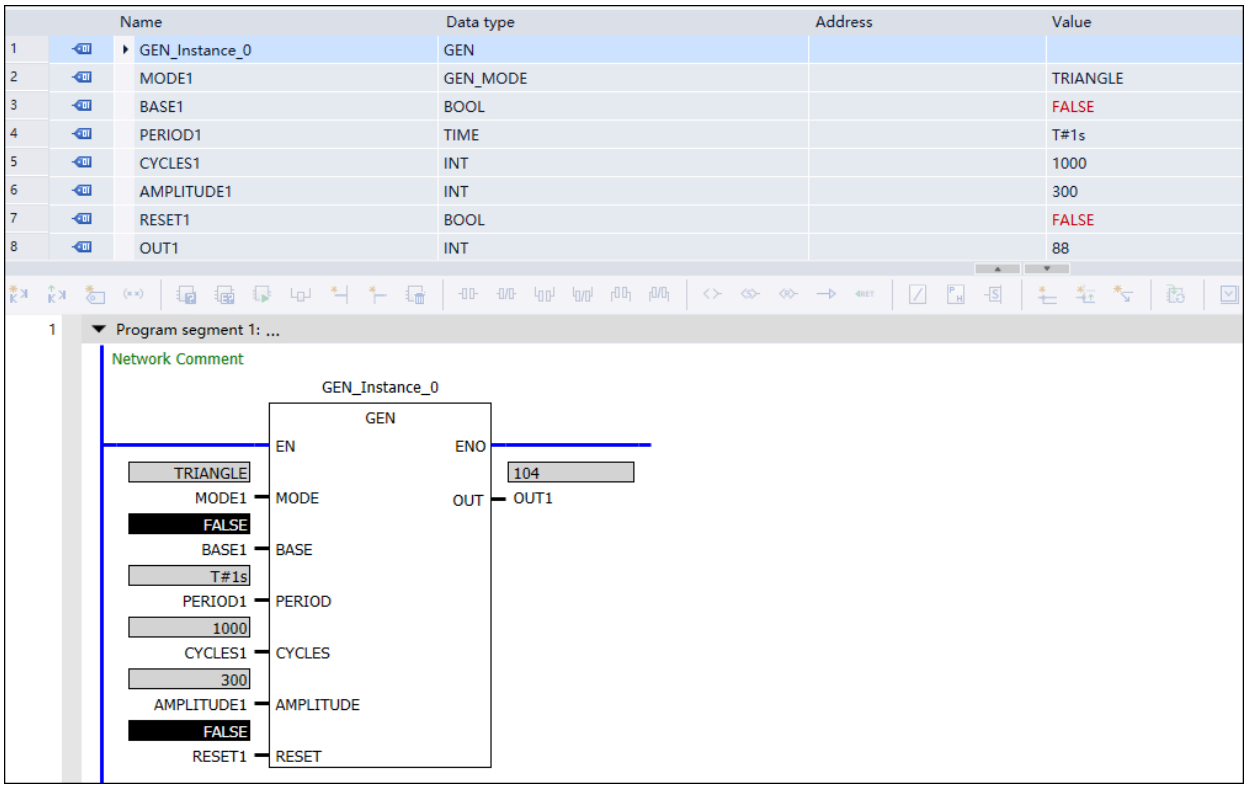
ST:

	Name	Data type	Address	Value
1	GEN_Instance_0	GEN		
2	MODE1	GEN_MODE		TRIANGLE
3	BASE1	BOOL		FALSE
4	PERIOD1	TIME		T#1s
5	CYCLES1	INT		1000
6	AMPLITUDE1	INT		300
7	RESET1	BOOL		FALSE
8	OUT1	INT		228

```
IF  CAS  FOR  WHI  REP  RET  EX
1  GEN_Instance_0(
2  MODE TRIANGLE  := MODE1 TRIANGLE ,
3  BASE  FALSE  := BASE1  FALSE ,
4  PERIOD  T#1s  := PERIOD1  T#1s ,
5  CYCLES  1000  := CYCLES1  1000 ,
6  AMPLITUDE  300  :=AMPLITUDE1  300 ,
7  RESET  FALSE  := RESET1  FALSE ,
8  OUT  228  => OUT1  228 );RETURN
```

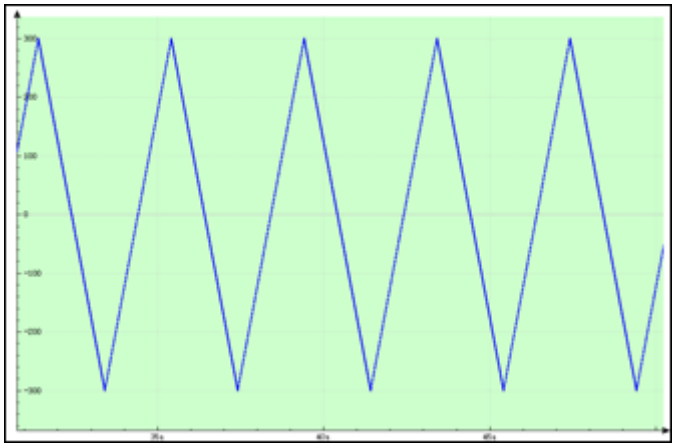
LD:

	Name	Data type	Address	Value
1	GEN_Instance_0	GEN		
2	MODE1	GEN_MODE		TRIANGLE
3	BASE1	BOOL		FALSE
4	PERIOD1	TIME		T#1s
5	CYCLES1	INT		1000
6	AMPLITUDE1	INT		300
7	RESET1	BOOL		FALSE
8	OUT1	INT		88

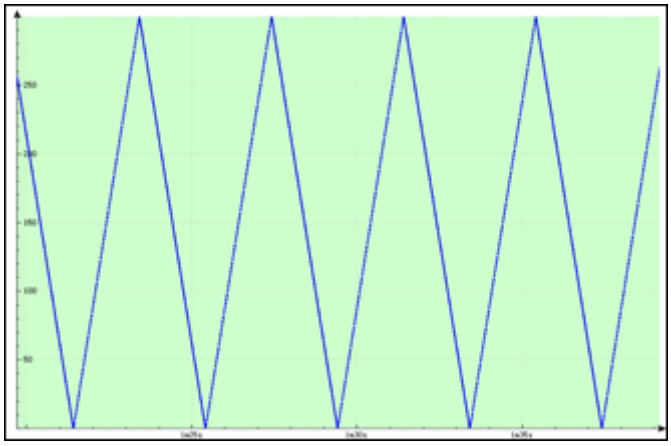


TRACE:

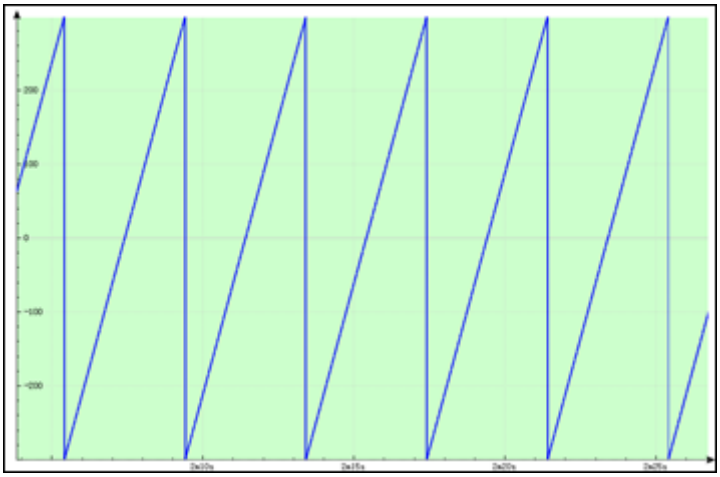
MODE = TRIANGLE



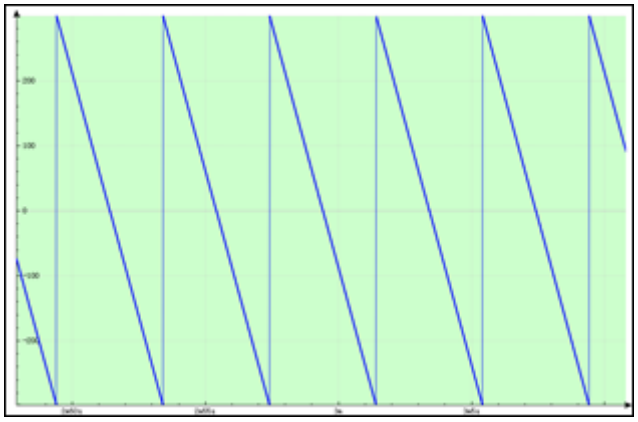
MODE = TRIANGLE_POS



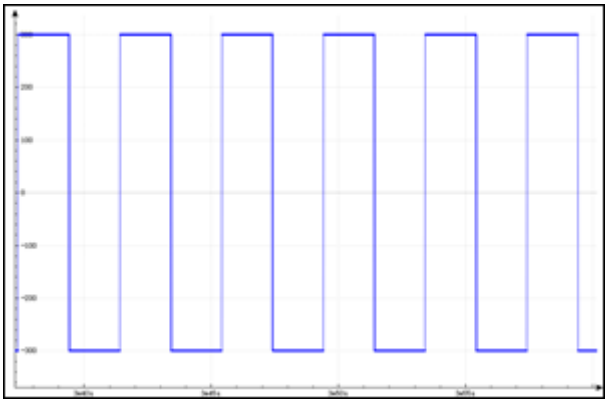
MODE = SAWTOOTH_RISE



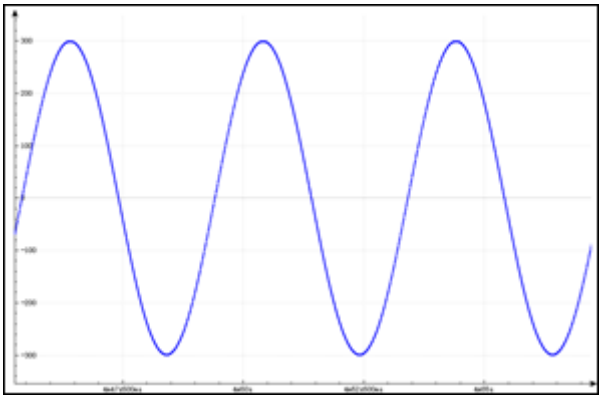
MODE = SAWTOOTH_FALL



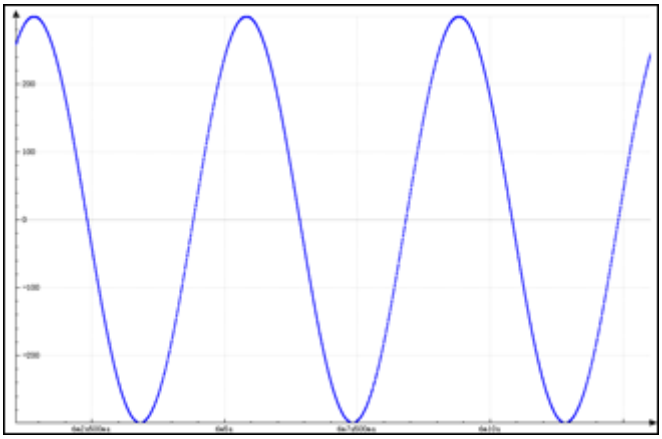
MODE = RECTANGLE



MODE = SINE



MODE = COSINE



18.2.7.4 FREQ_MEASURE

Measures the (average) frequency (Hz) of the input Boolean signal.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FREQ_MEASURE	Frequency measurement	FB	FREQ_MEASURE_Instance_0 FREQ_MEASURE EN ENO IN OUT PERIODS VALID RESET	FREQ_MEASURE_Instance_0(IN := , PERIODS := , RESET := , OUT => , VALID =>);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input signal	Follows data type	-	TRUE: The instruction starts working.
	PERIODS	Measurement period	Follows data type	1	Ranges from 1 (default) to 10. The time interval between two rising edges of the input signal is one period. By averaging the frequency values over N periods, we can obtain the frequency of the input signal. This variable specifies the number of periods used for the averaging operation.
	RESET	Reset a bit	Follows data type	-	When the value is TRUE, the frequency is re-measured. During normal execution, the value should be FALSE.
Output	OUT	Frequency output	Follows data type	-	Frequency of the input signal, in the unit of Hz
	VALID	Operation flag	Follows data type	-	The value is TRUE when the first operation is completed or the operation fails. The value is FALSE for other time.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PERIODS	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
VALID	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Program example**

ST:

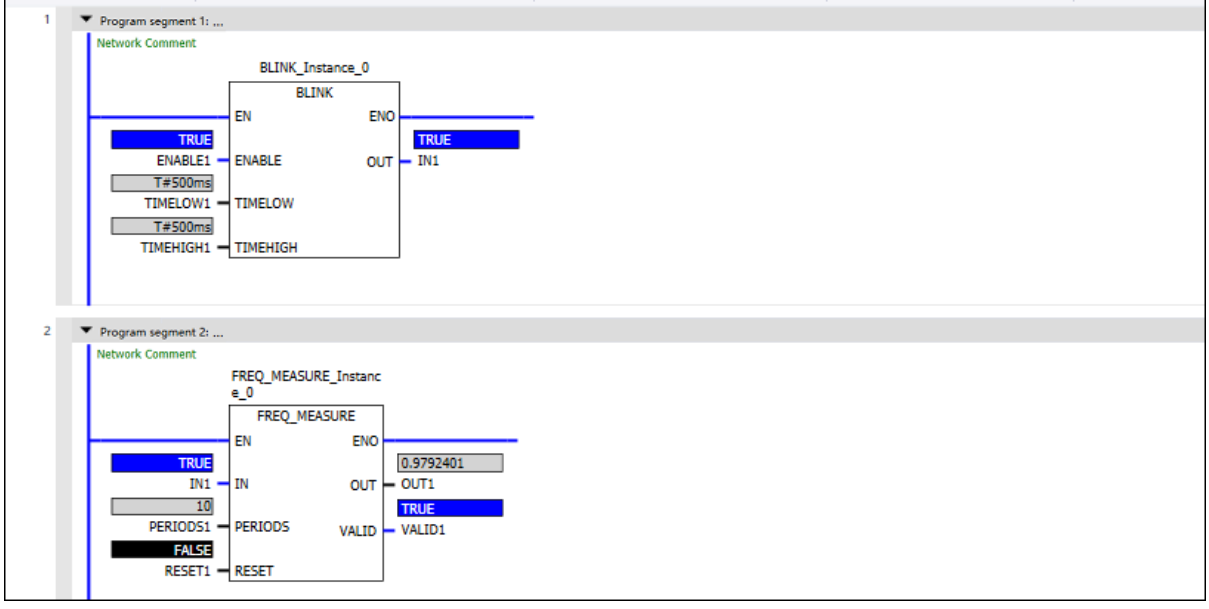
	Name	Data type	Address	Value
1	BLINK_Instance_0	BLINK		
2	FREQ_MEASURE_Instance_0	FREQ_MEASURE		
3	ENABLE1	BOOL		TRUE
4	TIMELOW1	TIME		T#500ms
5	TIMEHIGH1	TIME		T#500ms
6	OUT1	REAL		0.9792401
7	IN1	BOOL		TRUE
8	PERIODS1	INT (1..10)		10
9	RESET1	BOOL		FALSE
10	VALID1	BOOL		TRUE

```
BLINK_Instance_0(
ENABLE TRUE := ENABLE1 TRUE,
TIMELOW T#500ms := TIMELOW1 T#500ms,
TIMEHIGH T#500ms := TIMEHIGH1 T#500ms,
OUT TRUE => IN1 TRUE);

FREQ_MEASURE_Instance_0(
IN TRUE := IN1 TRUE,
PERIODS 10 := PERIODS1 10,
RESET FALSE := RESET1 FALSE,
OUT 0.9792401 => OUT1 0.9792401,
VALID TRUE => VALID1 TRUE);RETURN
```

LD:

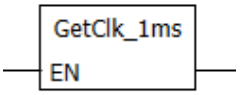
	Name	Data type	Address	Value
1	BLINK_Instance_0	BLINK		
2	FREQ_MEASURE_Instance_0	FREQ_MEASURE		
3	ENABLE1	BOOL		TRUE
4	TIMELOW1	TIME		T#500ms
5	TIMEHIGH1	TIME		T#500ms
6	OUT1	REAL		0.9792401
7	IN1	BOOL		TRUE
8	PERIODS1	INT (1..10)		10
9	RESET1	BOOL		FALSE
10	VALID1	BOOL		TRUE



18.2.7.5 GetClk_1ms

Outputs a timing pulse of 1 ms.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetClk_1ms	1-ms timing pulse	FC		:= GetClk_1ms();

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Return value	Follows data type	-	Pulse output

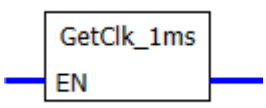
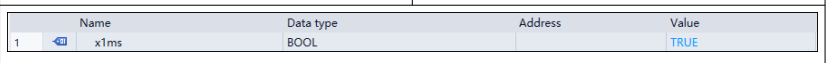
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

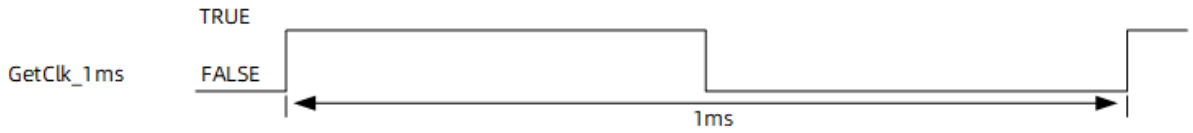
- **Function**

Outputs a timing pulse of 1 ms.

- **Program example**

Name	LD	ST
Define variables	VAR x1ms: BOOL; END_VAR	
Program		x1ms FALSE :=GetClk_1ms();
Operation result		

- **Timing diagram**



- **Precautions**

  When using this instruction in a ladder diagram program, if the EN input is FALSE, the output becomes FALSE. The period of the timing pulse is not affected by instruction calls, task type, or task cycle. It is an absolute pulse generated by the system:

  - The pulse waveform is independent of calling location, task cycle, and task type.
  - The value obtained through the instruction is the instantaneous value of the pulse at the current moment. Therefore, multiple calls within the same cycle may yield different results.
  - After the PLC switches to RUN mode, the result of the first cycle is indeterminate and may be TRUE or FALSE.

18.2.7.6 GetClk_10ms

Outputs a timing pulse of 10 ms.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetClk_10ms	10-ms timing pulse	FC	GetClk_10ms EN	:= GetClk_10ms();

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Return value	Follows data type	-	Pulse output

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

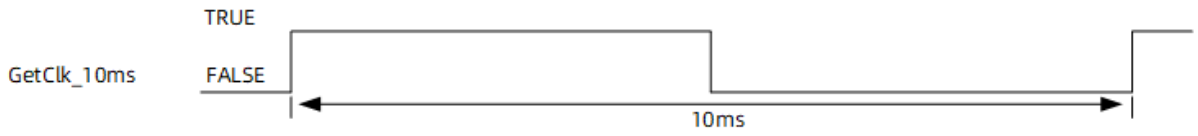
- **Function**

  Outputs a timing pulse of 10 ms.

- **Program example**

Name	LD	ST
Define variables	VAR x10ms: BOOL; END_VAR	
Program	GetClk_10ms EN	x10ms FALSE :=GetClk_10ms();
Operation result	Name / Data type / Address / Value 1 x10ms / BOOL / / TRUE	

- **Timing diagram**



- **Precautions**

  When using this instruction in a ladder diagram program, if the EN input is FALSE, the output becomes FALSE. The period of the timing pulse is not affected by instruction calls, task type, or task cycle. It is an absolute pulse generated by the system:

  - The pulse waveform is independent of calling location, task cycle, and task type.
  - The value obtained through the instruction is the instantaneous value of the pulse at the current moment. Therefore, multiple calls within the same cycle may yield different results.
  - After the PLC switches to RUN mode, the result of the first cycle is indeterminate and may be TRUE or FALSE.

18.2.7.7 GetClk_20ms

Outputs a timing pulse of 20 ms.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetClk_20ms	20-ms timing pulse	FC	GetClk_20ms EN	:= GetClk_20ms();

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Return value	Follows data type	-	Pulse output

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

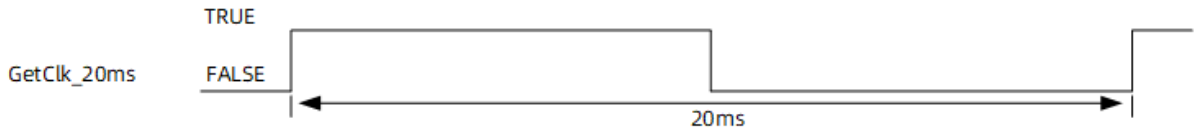
- **Function**

  Outputs a timing pulse of 20 ms.

- **Program example**

Name	LD	ST
Define variables	VAR x20ms: BOOL; END_VAR	
Program	GetClk_20ms EN	x20ms TRUE :=GetClk_20ms();
Operation result	Name / Data type / Address / Value 1 x20ms / BOOL / / TRUE	

- **Timing diagram**



- **Precautions**

  When using this instruction in a ladder diagram program, if the EN input is FALSE, the output becomes FALSE. The period of the timing pulse is not affected by instruction calls, task type, or task cycle. It is an absolute pulse generated by the system:

  - The pulse waveform is independent of calling location, task cycle, and task type.
  - The value obtained through the instruction is the instantaneous value of the pulse at the current moment. Therefore, multiple calls within the same cycle may yield different results.
  - After the PLC switches to RUN mode, the result of the first cycle is indeterminate and may be TRUE or FALSE.

18.2.7.8 GetClk_100ms

Outputs a timing pulse of 100 ms.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetClk_100ms	100-ms timing pulse	FC	GetClk_100ms EN	:= GetClk_100ms();

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Return value	Follows data type	-	Pulse output

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

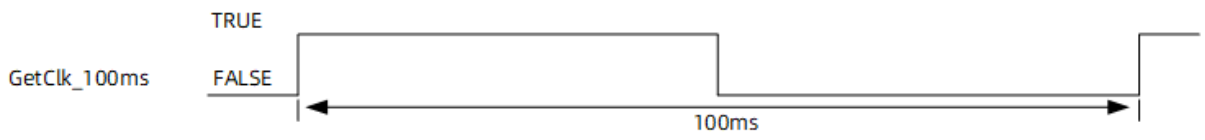
- **Function**

  Outputs a timing pulse of 100 ms.

- **Program example**

Name	LD	ST
Define variables	VAR x100ms: BOOL; END_VAR	
Program	GetClk_100ms EN	x100ms TRUE :=GetClk_100ms();
Operation result	Name / Data type / Address / Value 1 x100ms / BOOL / / TRUE	

- **Timing diagram**



- **Precautions**

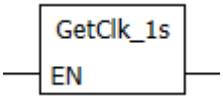
When using this instruction in a ladder diagram program, if the EN input is FALSE, the output becomes FALSE. The period of the timing pulse is not affected by instruction calls, task type, or task cycle. It is an absolute pulse generated by the system:

- The pulse waveform is independent of calling location, task cycle, and task type.
- The value obtained through the instruction is the instantaneous value of the pulse at the current moment. Therefore, multiple calls within the same cycle may yield different results.
- After the PLC switches to RUN mode, the result of the first cycle is indeterminate and may be TRUE or FALSE.

18.2.7.9 GetClk_1s

Outputs a timing pulse of 1s.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetClk_1s	1-s timing pulse	FC		:= GetClk_1s();

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Return value	Follows data type	-	Pulse output

  Variable types

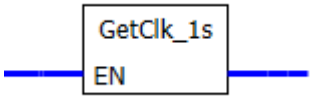
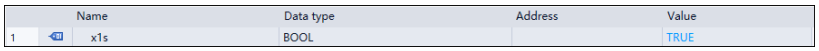
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

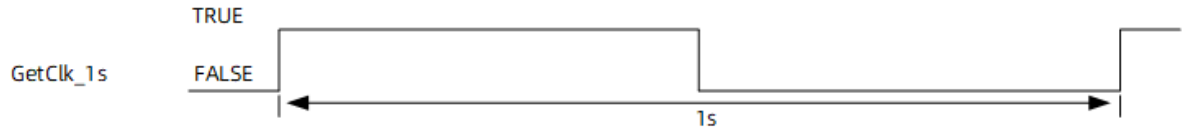
  Outputs a timing pulse of 1s.

- **Program example**

Name	LD	ST
Define variables	VAR x1s: BOOL; END_VAR	

Name	LD	ST
Program		x1s FALSE :=GetClk_1s();
Operation result		

- **Timing diagram**



- **Precautions**

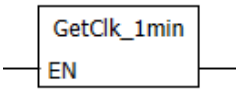
  When using this instruction in a ladder diagram program, if the EN input is FALSE, the output becomes FALSE. The period of the timing pulse is not affected by instruction calls, task type, or task cycle. It is an absolute pulse generated by the system:

  - The pulse waveform is independent of calling location, task cycle, and task type.
  - The value obtained through the instruction is the instantaneous value of the pulse at the current moment. Therefore, multiple calls within the same cycle may yield different results.
  - After the PLC switches to RUN mode, the result of the first cycle is indeterminate and may be TRUE or FALSE.

18.2.7.10 GetClk_1min

Outputs a timing pulse of 1 min.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetClk_1min	1-min timing pulse	FC		:= GetClk_1min();

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Return value	Follows data type	-	Pulse output

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

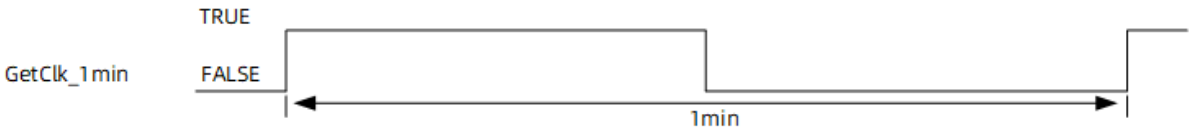
- **Function**

  Outputs a timing pulse of 1 min.

- **Program example**

Name	LD	ST
Define variables	VAR x1min: BOOL; END_VAR	
Program	GetClk_1min EN	x1min TRUE :=GetClk_1min();
Operation result	Name / Data type / Address / Value 1 x1min / BOOL / / FALSE	

- **Timing diagram**



- **Precautions**

  When using this instruction in a ladder diagram program, if the EN input is FALSE, the output becomes FALSE. The period of the timing pulse is not affected by instruction calls, task type, or task cycle. It is an absolute pulse generated by the system:

  - The pulse waveform is independent of calling location, task cycle, and task type.
  - The value obtained through the instruction is the instantaneous value of the pulse at the current moment. Therefore, multiple calls within the same cycle may yield different results.
  - After the PLC switches to RUN mode, the result of the first cycle is indeterminate and may be TRUE or FALSE.

18.2.8 Filtering Algorithm

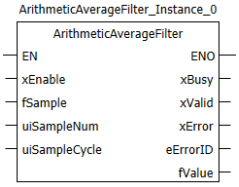
18.2.8.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.8.2 ArithmeticAverageFilter	FB	Arithmetic average filter	Util	Library version 1.0.0.0 and later
18.2.8.3 DebounceFilter	FB	Debounce filter		Library version 1.0.0.0 and later
18.2.8.4 First-Order Lag Filter	FB	First-order lag filter		Library version 1.0.0.0 and later
18.2.8.5 LimitingAverageFilter	FB	Limiting average filter		Library version 1.0.0.0 and later
18.2.8.6 Limiting Debounce Filter	FB	Limiting debounce filter		Library version 1.0.0.0 and later
18.2.8.7 LimitingFilter	FB	Limiting filter		Library version 1.0.0.0 and later
18.2.8.8 Median Average Filter	FB	Median average filter		Library version 1.0.0.0 and later
18.2.8.9 MedianFilter	FB	Median filter		Library version 1.0.0.0 and later
18.2.8.10 Recursive Average Filter	FB	Recursive average filter		Library version 1.0.0.0 and later

18.2.8.2 ArithmeticAverageFilter

Calculates the arithmetic average filter.

- Instruction format

Instruction	Name	FB/FC	LD Representation	ST Representation
ArithmeticAverage Filter	Arithmetic average filter	FB		ArithmeticAverage Filter_Instance_0(xEnable := , fSample := , uiSampleNum := , uiSampleCycle := , xBusy => , xValid => , xError => , eErrorID => , fValue =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	xEnable	Filter enable	Follows data type	FALSE	FALSE: Disabled. TRUE: Enabled.
	fSample	Input sampling value	Follows data type	0	Input sampling value
	uiSampleNum	Input sampling count	1 to 1000	1	Input sampling count
	uiSampleCycle	Input sampling period	1 to 1000	1	Input sampling period
Output	xBusy	Instruction executing	Follows data type	FALSE	TRUE: The function block is executing.
	xValid	Output active	Follows data type	FALSE	TRUE: Instruction execution is valid.
	xError	Fault	Follows data type	FALSE	TRUE: Instruction execution is abnormal.
	eErrorID	Fault ID	Follows data type	0	Refer to 18.2.8.11 Fault Code.
	fValue	Filter output value	Follows data type	0	Filter output value

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
xEnable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
fSample	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
uiSampleNum	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
uiSampleCycle	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
xValid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eErrorID	UTIL_ERROR. For details, see 18.2.8.11 Fault Code.																			
fValue	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

- **Function**

  When xEnable is set to TRUE, the sampling variable fValue is input based on the given scan period count uiSampleCycle and sampling count uiSampleNum. Sample data for consecutively uiSampleNum times and average the sampled data.

  The larger uiSampleNum is, the higher the signal smoothness, but the lower the sensitivity.

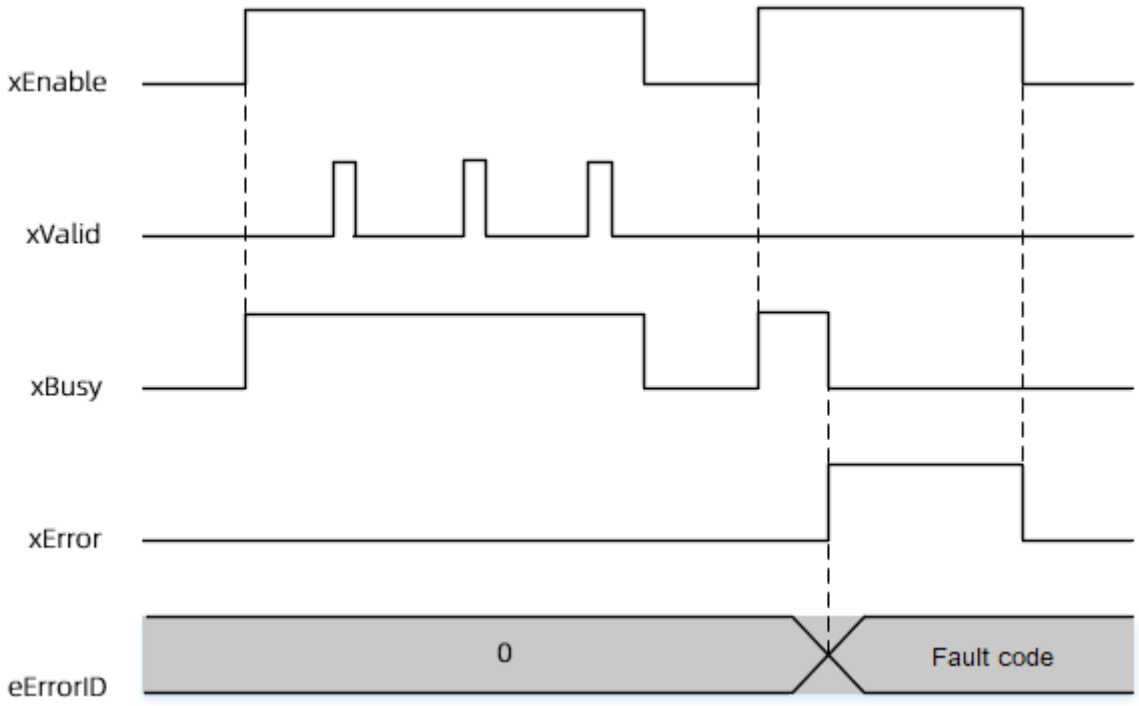
  The smaller uiSampleNum is, the lower the signal smoothness, but the higher the sensitivity.

  For general flow calculation, set uiSampleNum to 12. For pressure calculation, set uiSampleNum to 4.

  Advantages: Suitable for filtering signals with random interference that fluctuate around a certain average value.

  Disadvantages: Not suitable for real-time control with a slow measurement speed or requiring a fast data calculation speed, and it consumes a significant amount of RAM.

- **Sequence diagram**



- **Program example**

  The sampling type is consecutive sampling. The raw data for sampling is sourced from the consecutively updated program variables.

  The number of samples for the sampling calculation is 3 (unsigned integer ranging from 1 to 3000).

  The sampling period is 1 (data sampling is executed once every given number of running periods, with the default value of 1).

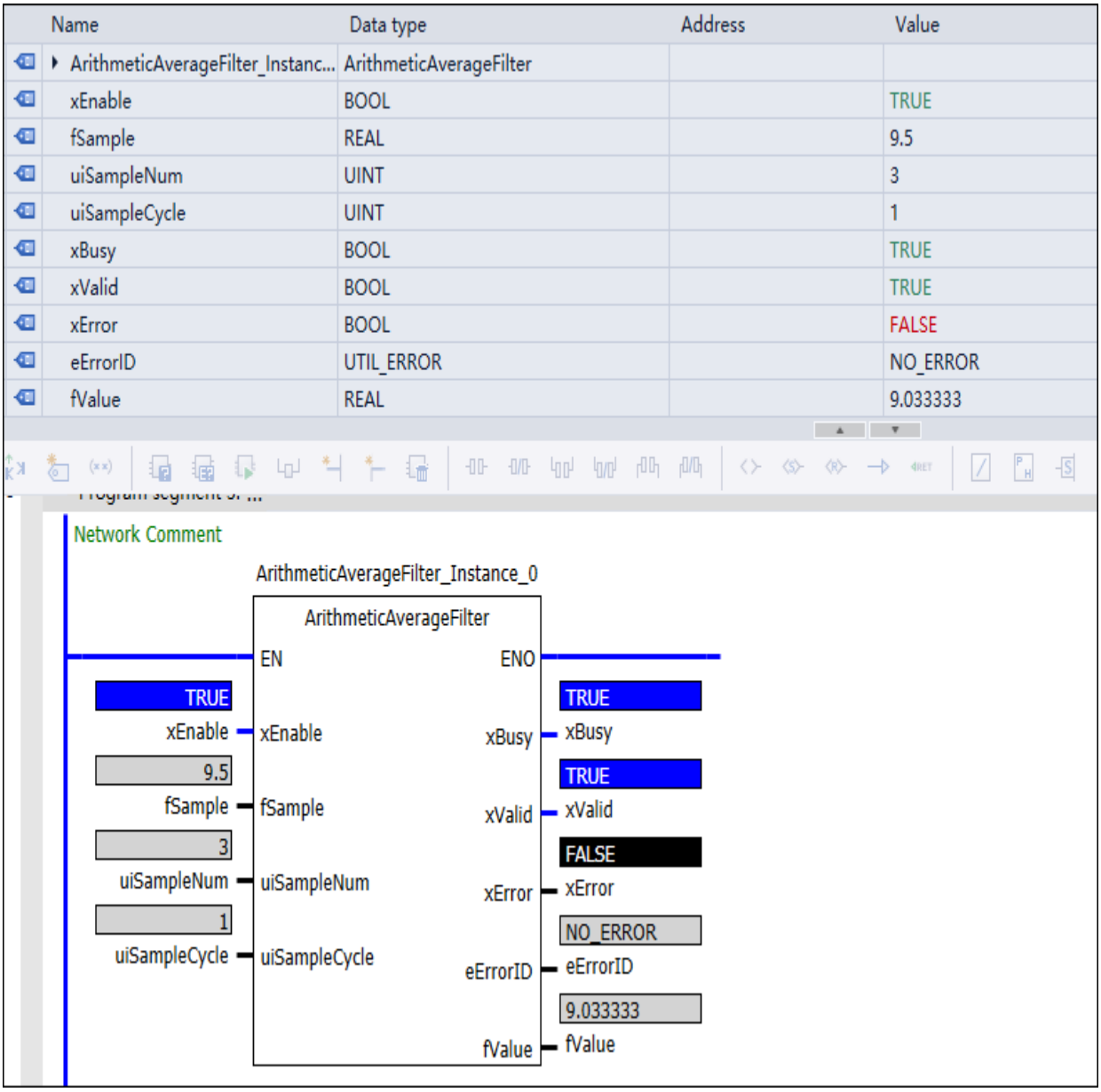
  When the input xFilter is TRUE, the function block is enabled and begins sampling and filtering. With the number of samples set to 3, filter calculation is performed once for every three samples collected. The arithmetic average of the 3 sample values is calculated as the filter output value, and the filter execution flag xValid is set to TRUE.

  ST:

Name	Data type	Address	Value
ArithmeticAverageFilter_Instanc...	ArithmeticAverageFilter		
xEnable	BOOL		TRUE
fSample	REAL		9.5
uiSampleNum	UINT		3
uiSampleCycle	UINT		1
xBusy	BOOL		TRUE
xValid	BOOL		TRUE
xError	BOOL		FALSE
eErrorID	UTIL_ERROR		NO_ERROR
fValue	REAL		9.033333

```
ArithmeticAverageFilter_Instance_0(
xEnable TRUE := xEnable TRUE,
fSample 9.5 := fSample 9.5,
uiSampleNum 3 := uiSampleNum 3,
uiSampleCycle 1 := uiSampleCycle 1,
xBusy TRUE => xBusy TRUE,
xValid TRUE => xValid TRUE,
xError FALSE => xError FALSE,
eErrorID NO_ERROR => eErrorID NO_ERROR,
fValue 9.033333 => fValue 9.033333 );RETURN
```

  LD:



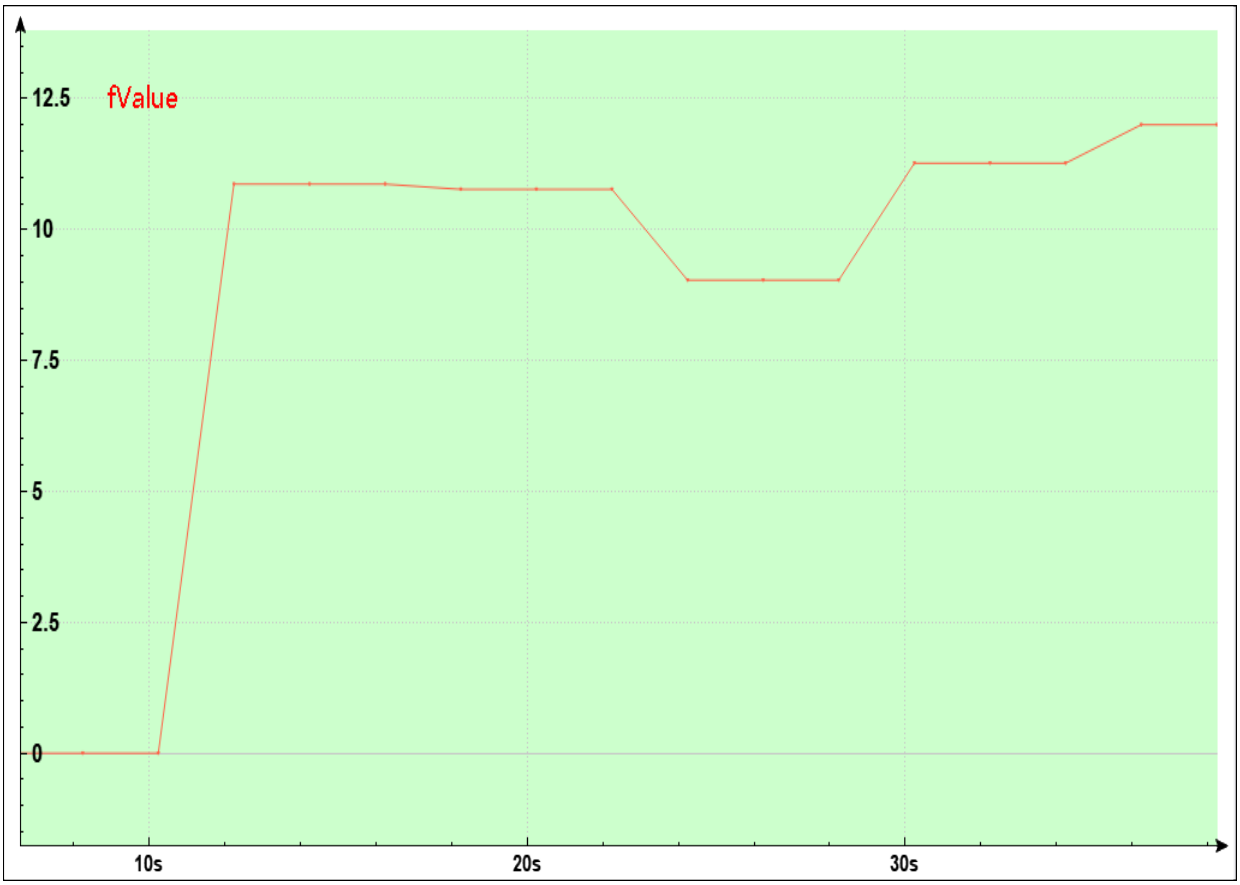
The sampling values of 12 consecutive periods are respectively 9.8, 9.8, 13.0, 12.5, 11.4, 8.4, 8.7, 8.9, 9.5, 11.1, 10.7, and 12.0.

After filtering, the values are respectively 0, 0, 10.87, 10.87, 10.87, 10.77, 10.77, 10.77, 9.03, 9.03, 9.03, and 11.27.

The following figure shows the curve before filtering.



The following figure shows the curve after filtering.



- **Precautions**
  - In the filter function block, the input parameter uiSampleCycle indicates the sampling period. When the value of uiSampleCycle is 1, sampling is performed every period. When the value of uiSampleCycle is 3, sampling is performed once every three periods. The function block execution flag xValid outputs

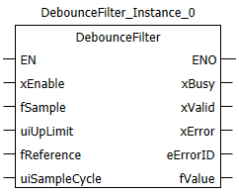
TRUE, indicating that the filtering is valid. The filter output value of fValue is updated when the filter output is valid.

- If the sampled data is stored in an array, the data needs to be put in the filter function block one by one and handled by the user.
- If the value of the input parameter uiSampleNum is not within the valid range, the xError flag is set to TRUE, and the function block generates a fault code eErrorID, indicating the message: "SAMPLENUM_PARAMETER_SET_ERROR". To handle the fault, you need to reset the uiSampleNum parameter within the valid range and re-trigger a flow to reset the fault and the xError signal.
- For details about faults, see UTIL_ERROR.

18.2.8.3 DebounceFilter

Calculates the debounce filter.

- Instruction format

Instruction	Name	FB/FC	LD Representation	ST Representation
DebounceFilter	Debounce filter	FB		DebounceFilter_Instance_0(xEnable := , fSample := , uiUpLimit := , fReference := , uiSampleCycle := , xBusy => , xValid => , xError => , eErrorID => , fValue =>);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	xEnable	Filter enable	Follows data type	FALSE	FALSE: Disabled. TRUE: Enabled.
	fSample	Input sampling value	Follows data type	0	Input sampling value
	uiUpLimit	Filter counter upper limit	1-65535	1	Filter counter upper limit

Input/ Output	Variable	Name	Valid Range	Default Value	Description
	Reference	Input reference value	Follows data type	0	Input reference value
	uiSampleCycle	Input sampling period	1-1000	1	Input sampling period
Output	xBusy	Instruction executing	Follows data type	FALSE	TRUE: The function block is executing.
	xValid	Output active	Follows data type	FALSE	TRUE: Instruction execution is valid.
	xError	Fault	Follows data type	FALSE	TRUE: Instruction execution is abnormal.
	eErrorID	Fault ID	Follows data type	0	Refer to 18.2.8.11 Fault Code.
	fValue	Filter output value	Follows data type	0	Filter output value

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
xEnable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
fSample	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
uiUpLimit	_	_	_	_	_	_	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Reference	_	_	_	_	_	_	–	–	–	–	–	–	–	√	–	–	–	–	–	–
uiSampleCycle	_	_	_	_	_	_	√	–	–	–	–	–	–	–	–	–	–	–	–	–
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xValid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eErrorID	UTIL_ERROR. For details, see 18.2.8.11 Fault Code.																			

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
fValue	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

- **Function**

  When xEnable is set to TRUE, the sampling variable fSample is input based on the given scan period count uiSampleCycle, sampling reference value fReference, and filter upper limit uiUpLimit.

  Compare each sampled value with the current effective value:

  If the sampling value is equal to the current effective value, set the current count value to zero.

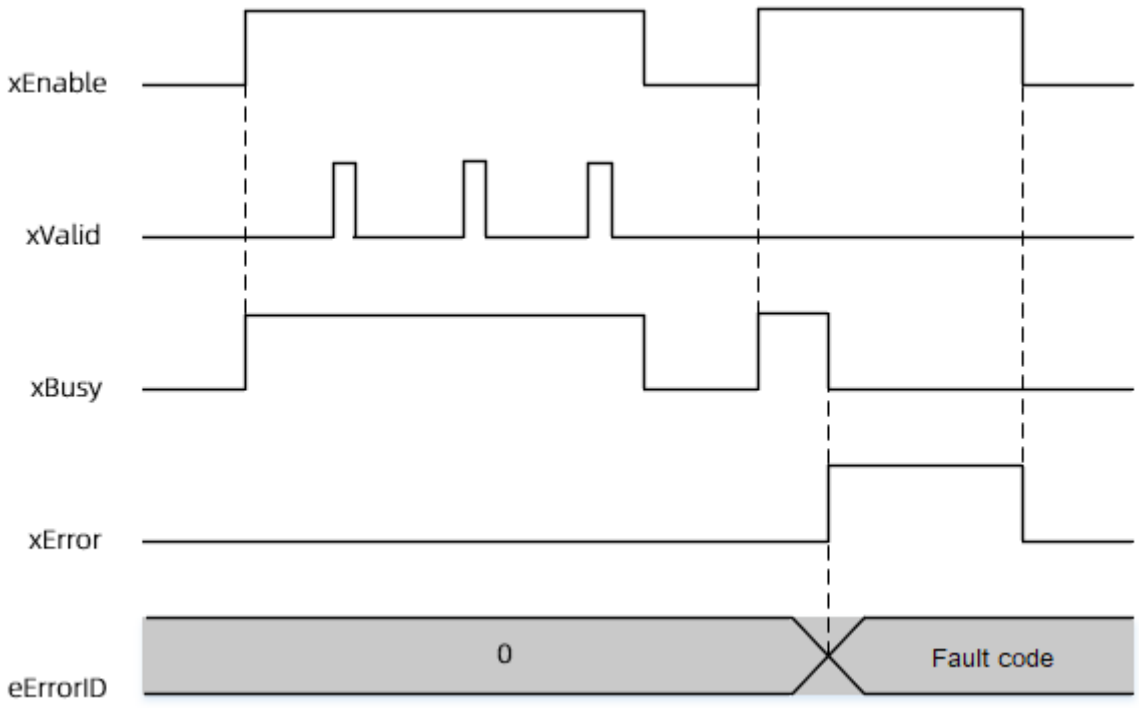
  If the sampling value is not equal to the current effective value, increment the count value by 1 and check whether the count value exceeds the upper limit uiUpLimit.

  If the count value exceeds the upper limit, replace the current filter value with the current sampling value, output the value, and reset the count value to zero.

  Advantages: Suitable for slowly changing parameters; it prevents repeated on/off switchovers on the controller or numerical jitters on the display near critical values.

  Disadvantages: Not suitable for rapidly changing parameters; if the value sampled at the time of counter overflow happens to be an interference value, it may be mistakenly introduced into the system as an effective value.

- **Sequence diagram**



- **Program example**

The sampling type is consecutive sampling. The raw data for sampling is sourced from the consecutively updated program variables.

The reference effective value of the sample is 10.0 (user-defined reference value of the floating-point type).

The maximum number of jitters for the sample value on the counter is 3 (unsigned integer).

The sampling period is 1 (data sampling is executed once every given number of running periods, with the default value of 1).

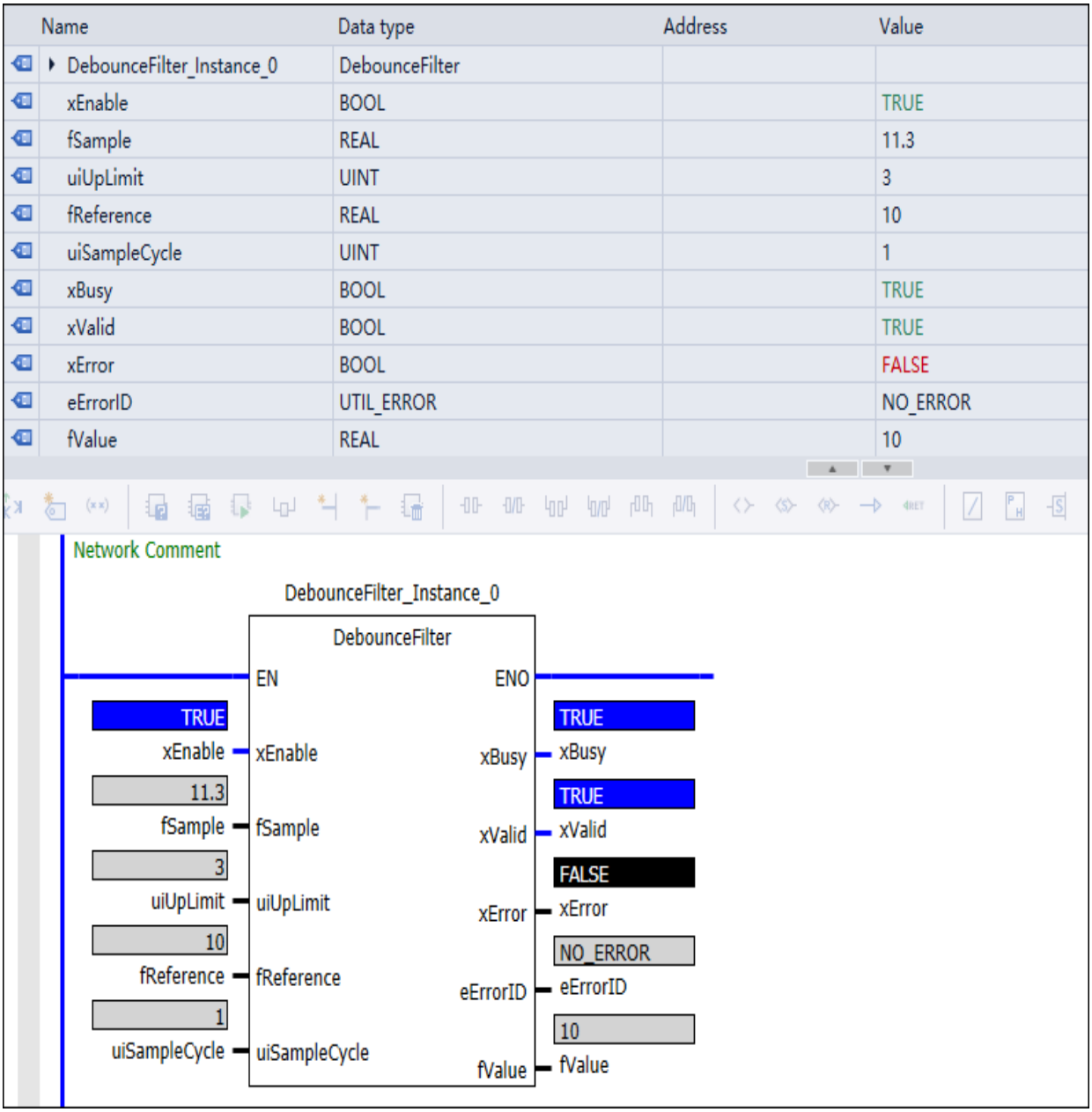
When the input xEnable is set to TRUE, the function block starts sampling and filtering with energy flow effective. Set the reference value to 10.0 as the effective value. Compare the sampling value with the current effective value. If they are the same, reset the jitter counter value to zero. If they are different, increase the counter value by 1. When the counter value exceeds the counter value upper limit, replace the current effective value with the current sampling value as the filter effective value, output the effective value, and set the filter execution valid flag xValid to TRUE.

ST:

Name	Data type	Address	Value
DebounceFilter_Instance_0	DebounceFilter		
xEnable	BOOL		TRUE
fSample	REAL		11.3
uiUpLimit	UINT		3
fReference	REAL		10
uiSampleCycle	UINT		1
xBusy	BOOL		TRUE
xValid	BOOL		TRUE
xError	BOOL		FALSE
eErrorID	UTIL_ERROR		NO_ERROR
fValue	REAL		10

```
DebounceFilter_Instance_0(
xEnable TRUE := xEnable TRUE,
fSample 11.3 := fSample 11.3,
uiUpLimit 3 := uiUpLimit 3,
fReference 10 := fReference 10,
uiSampleCycle 1 := uiSampleCycle 1,
xBusy TRUE => xBusy TRUE,
xValid TRUE => xValid TRUE,
xError FALSE => xError FALSE,
eErrorID NO_ERROR => eErrorID NO_ERROR,
fValue 10 => fValue 10);RETURN
```

LD:



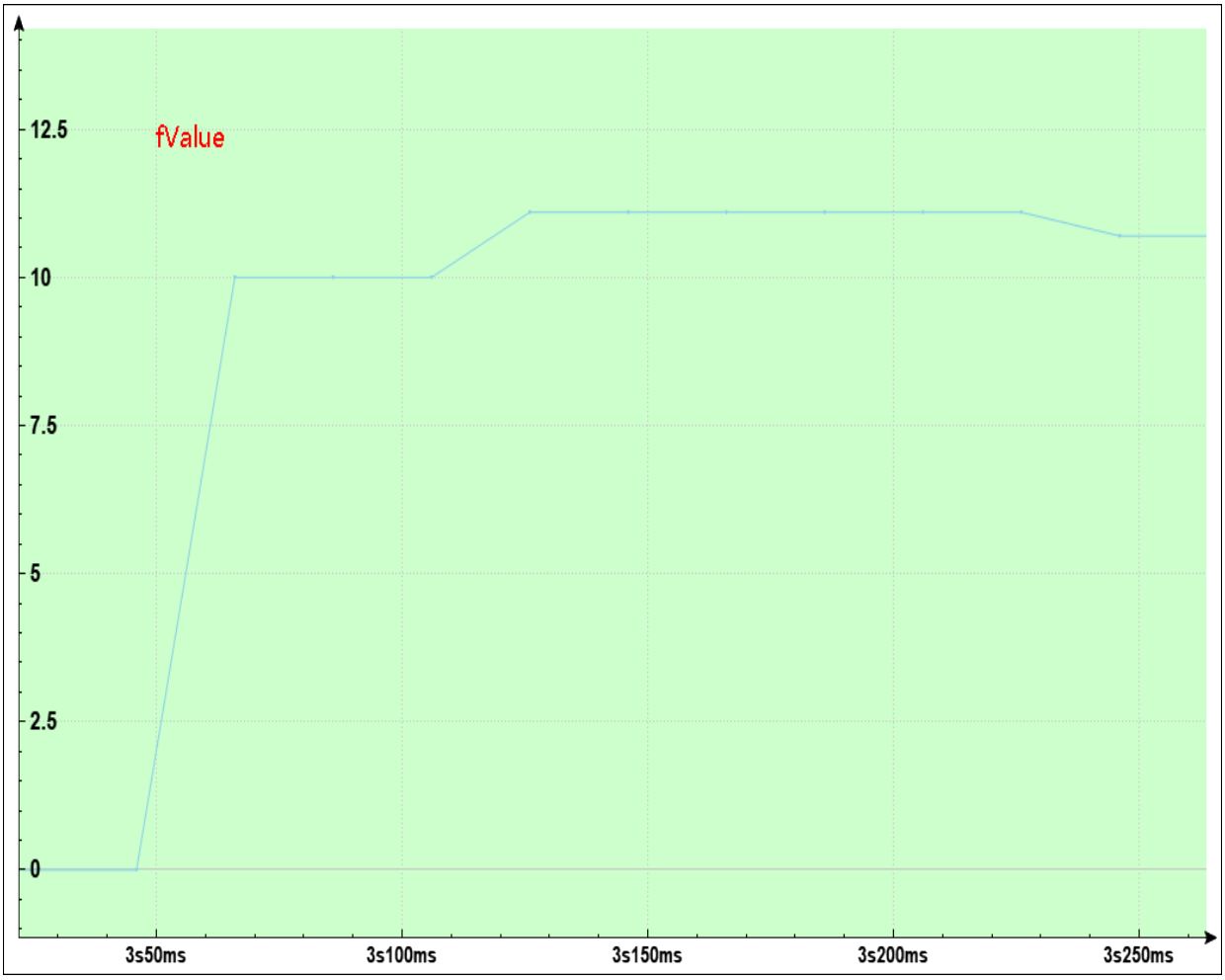
The sampling values of 10 consecutive periods are respectively 9.8, 10.8, 11.3, 11.1, 11.4, 11.1, 8.7, 8.9, 9.5, and 10.7.

After filtering, the values are respectively 10.0, 10.0, 10.0, 11.1, 11.1, 11.1, 11.1, 11.1, 11.1, and 10.7.

The following figure shows the curve before filtering.



The following figure shows the curve after filtering.

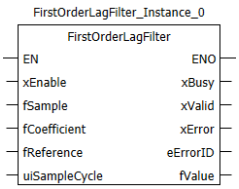


- **Precautions**
  - In the filter function block, the input parameter uiSampleCycle indicates the sampling period. When the value of uiSampleCycle is 1, sampling is performed every period. When the value of uiSampleCycle is 3, sampling is performed once every three periods. If the output of the function block flag DebounceFilter.xValid is TRUE, the filter is valid. The filter output value of fValue is updated when the filter output is valid.
  - If the sampled data is stored in an array, the data needs to be put in the filter function block one by one and handled by the user.
  - For details about errors, see UTIL_ERROR.

18.2.8.4 FirstOrderLagFilter

Calculates the first-order lag filter.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
First-Order Lag Filter	First-order lag filter	FB		FirstOrderLagFilter_Instance_0(xEnable := , fSample := , fCoefficient := , fReference := , uiSampleCycle := , xBusy => , xValid => , xError => , eErrorID => , fValue =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	xEnable	Filter enable	Follows data type	FALSE	FALSE: Disabled. TRUE: Enabled.
	fSample	Input sampling value	Follows data type	0	Input sampling value
	fCoefficient	Input first-order low-pass filter coefficient	0-1	0	Input first-order low-pass filter coefficient (a = 0 to 1)
	Reference	Input effective value	Follows data type	0	Input effective value
	uiSampleCycle	Input sampling period	1-1000	1	Input sampling period
Output	xBusy	Instruction executing	Follows data type	FALSE	TRUE: The function block is executing.
	xValid	Output active	Follows data type	FALSE	TRUE: Instruction execution is valid.

Input/Output	Variable	Name	Valid Range	Default Value	Description
	xError	Fault	Follows data type	FALSE	TRUE: Instruction execution is abnormal.
	eErrorID	Fault ID	Follows data type	0	Refer to 18.2.8. 11 Fault Code.
	fValue	Filter output value	Follows data type	0	Filter output value

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
xEnable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
fSample	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
fCoefficient	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Reference	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
uiSampleCycle	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xValid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eErrorID	UTIL_ERROR. For details, see 18.2.8.11 Fault Code.																			
fValue	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

- **Function**

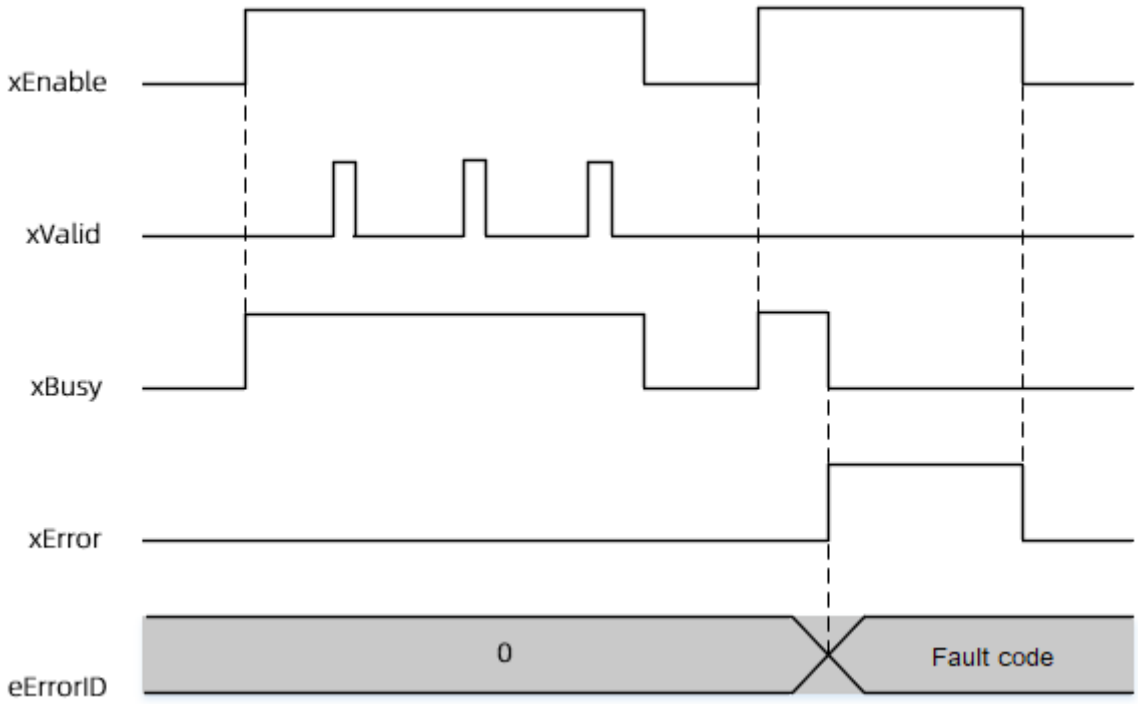
  When xEnable is set to TRUE, the sampling variable fSample is input based on the given scan period count uiSampleCycle, sampling reference value fReference, and input first-order low-pass filter coefficient fCoefficient (0 to 1).

  Each time a new value is sampled: Current filter result = fCoefficient x Current sampling value + (1 - fCoefficient) x Previous filter result

  Advantages: It suppresses periodic interference well, and is suitable for systems with high-frequency oscillations.

  Disadvantages: It has phase lag and low sensitivity. The lagging program depends on the value of the fCoefficient. It cannot eliminate interference signals with the filter frequency higher than half of the sampling frequency.

- **Sequence diagram**



- **Program example**

  The sampling type is consecutive sampling. The raw data for sampling is sourced from the consecutively updated program variables.

  The first-order low-pass filter coefficient is 0.01 (floating-point type).

  The filter reference value is 10.0 and the sampling period is 1 (i.e., sample once per cycle).

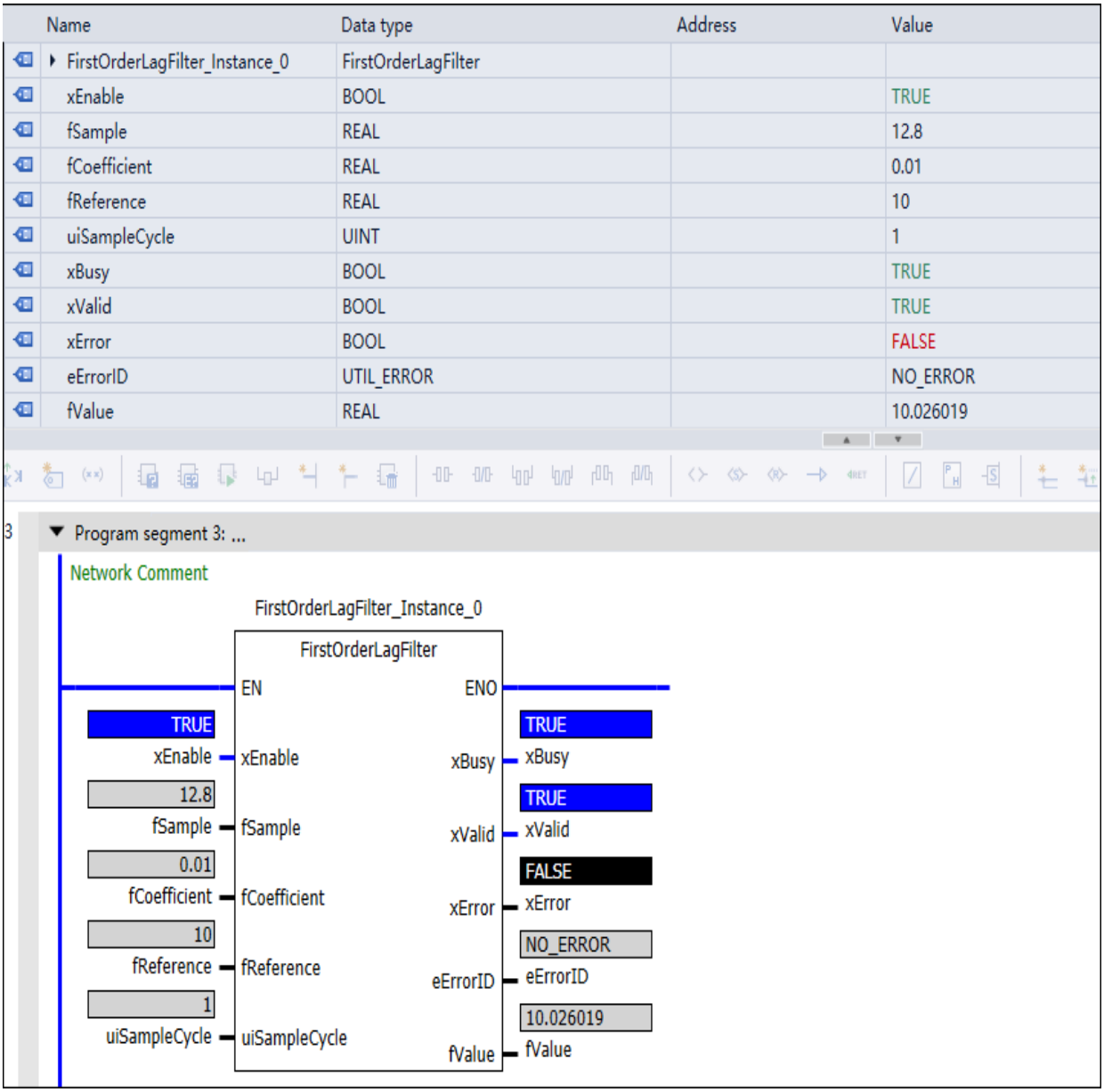
  When the input xEnable is set to TRUE, the function block starts sampling and filtering with energy flow effective. Set the initial reference value to 10.0 as the effective value and the first-order low-pass filter coefficient to 0.01. Calculate the filter value as follows: Current filter result = fCoefficient x Current sampling value + (1 - fCoefficient) x Previous filter result. Output the filter value, and set the filter execution valid flag xValid to TRUE.

  ST:

Name	Data type	Address	Value
FirstOrderLagFilter_Instance_0	FirstOrderLagFilter		
xEnable	BOOL		TRUE
fSample	REAL		12.8
fCoefficient	REAL		0.01
fReference	REAL		10
uiSampleCycle	UINT		1
xBusy	BOOL		TRUE
xValid	BOOL		TRUE
xError	BOOL		FALSE
eErrorID	UTIL_ERROR		NO_ERROR
fValue	REAL		10.026019

```
FirstOrderLagFilter_Instance_0(
xEnable  TRUE  := xEnable  TRUE ,
fSample  12.8  := fSample  12.8 ,
fCoefficient  0.01  := fCoefficient  0.01 ,
fReference  10  := fReference  10 ,
uiSampleCycle  1  := uiSampleCycle  1 ,
xBusy  TRUE  => xBusy  TRUE ,
xValid  TRUE  => xValid  TRUE ,
xError  FALSE  => xError  FALSE ,
eErrorID NO_ERROR  => eErrorID NO_ERROR ,
fValue  10.026019  => fValue  10.026019 );
```

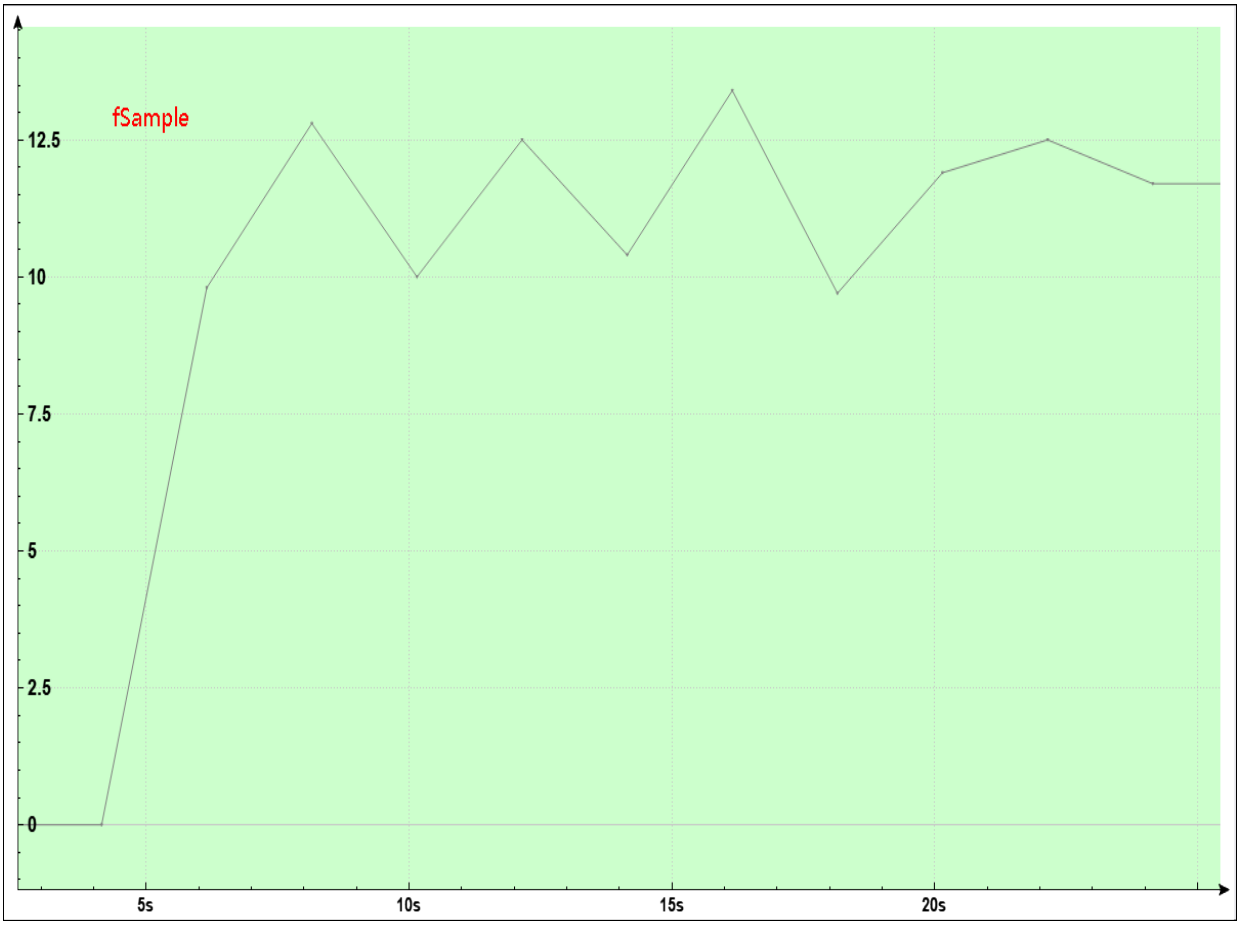
LD:



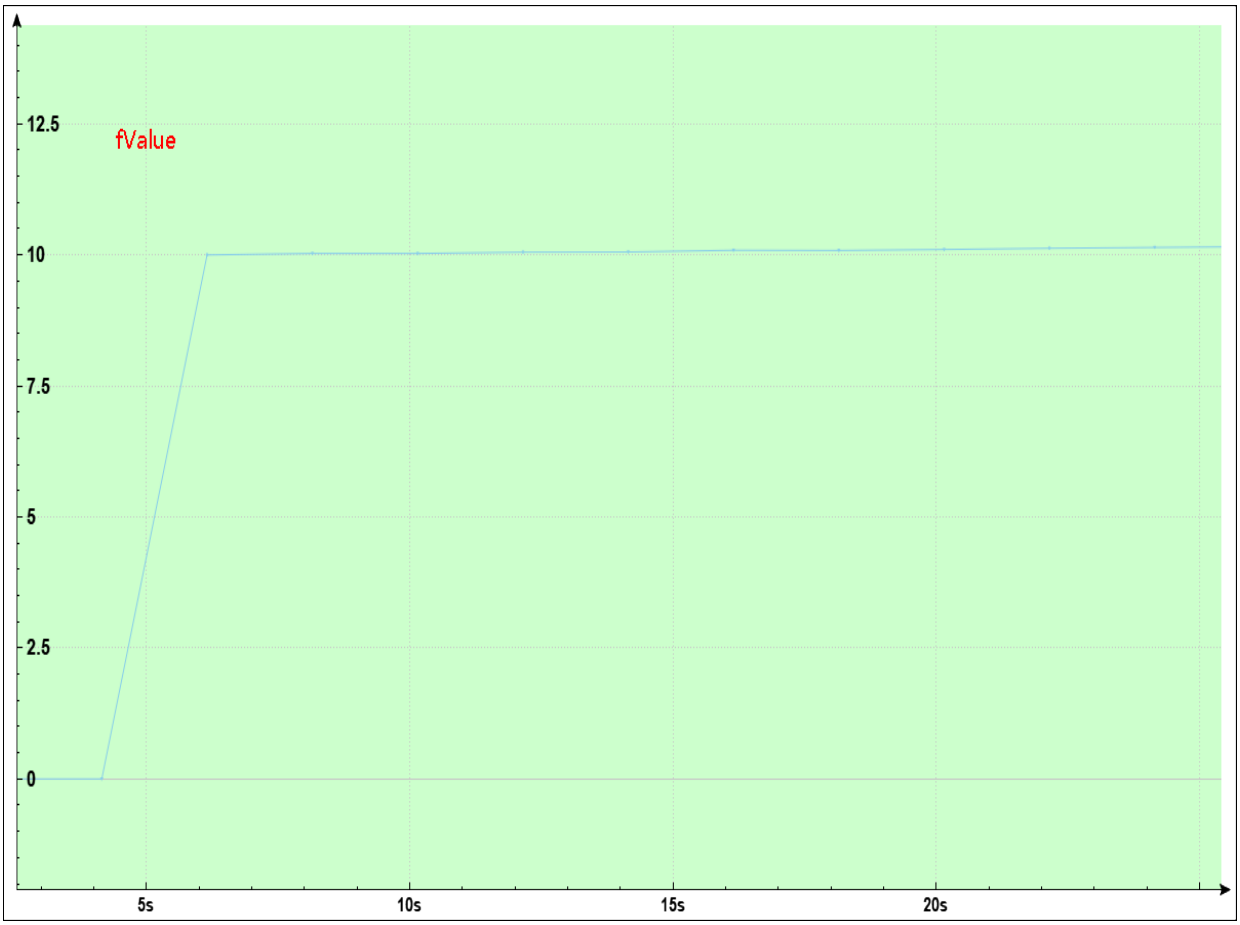
The sampling values of 10 consecutive periods are respectively 9.8, 12.8, 10.0, 12.5, 10.4, 13.4, 9.7, 11.9, 12.5, and 11.7.

After filtering, the values are respectively 9.998, 10.03, 10.03, 10.05, 10.05, 10.09, 10.08, 10.10, 10.13, and 10.14.

The following figure shows the curve before filtering.



The following figure shows the curve after filtering.



- **Precautions**
  - In the filter function block, the input parameter uiSampleCycle indicates the sampling period. When the value of uiSampleCycle is 1, sampling is performed every period. When the value of uiSampleCycle

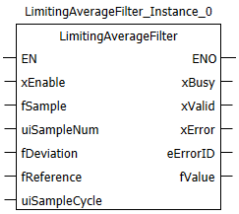
is 3, sampling is performed once every three periods. If the output of the function block flag FirstOrderLagFilter.xValid is TRUE, the filter is valid. The filter output value of fValue is updated when the filter output is valid.

- If the sampled data is stored in an array, the data needs to be put in the filter function block one by one and handled by the user.
- For details about errors, see UTIL_ERROR.

18.2.8.5 LimitingAverageFilter

Calculates limiting average filter.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LimitingAverageFilter	Limiting average filter	FB		LimitingAverageFilter_Instance_0(xEnable := , fSample := , uiSampleNum := , fDeviation := , fReference := , uiSampleCycle := , xBusy => , xValid => , xError => , eErrorID => , fValue =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	xEnable	Filter enable	Follows data type	FALSE	FALSE: Disabled. TRUE: Enabled.
	fSample	Input sampling value	Follows data type	0	Input sampling value
	uiSampleNum	Input sampling count	1-1000	1	Input sampling count

Input/Output	Variable	Name	Valid Range	Default Value	Description
	fDeviation	Maximum allowable deviation for two consecutive samples	Follows data type	0	Maximum allowable deviation for two consecutive samples
	Reference	Input effective value	Follows data type	0	Input effective value
	uiSampleCycle	Input sampling period	1-1000	1	Input sampling period
Output	xBusy	Instruction executing	Follows data type	FALSE	TRUE: The function block is executing.
	xValid	Output active	Follows data type	FALSE	TRUE: Instruction execution is valid.
	xError	Fault	Follows data type	FALSE	TRUE: Instruction execution is abnormal.
	eErrorID	Fault ID	Follows data type	0	Refer to 18.2.8.11 Fault Code.
	fValue	Filter output value	Follows data type	0	Filter output value

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
xEnable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
fSample	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
uiSampleNum	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
fDeviation	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Reference	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
uiSampleCycle	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xValid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eErrorID	UTIL_ERROR. For details, see 18.2.8.11 Fault Code.																			
fValue	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

- **Function**

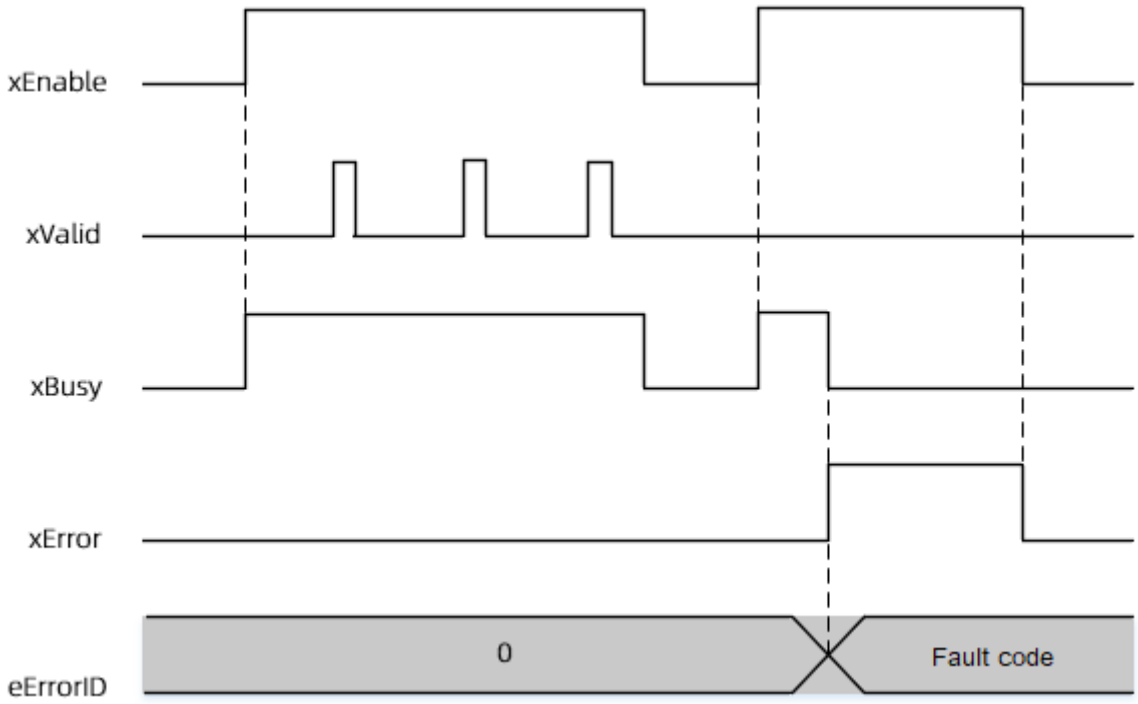
  When xEnable is set to TRUE, the sampling variable fSample is input based on the given scan period count uiSampleCycle, sampling count uiSampleNum, sampling reference effective value fReference, and maximum allowable deviation fDeviation for two consecutive samples.

  Both limiting filter and recursive average filter are used. Limit each new sampled data and then add it to the queue for recursive average filter processing.

  Advantages: Combines the advantages of both limiting filter and recursive average filter. It can eliminate sampling deviations caused by occasional pulse interference.

  Disadvantages: Consumes a significant amount of RAM.

- **Sequence diagram**



- **Program example**

  The sampling type is consecutive sampling. The raw data for sampling is sourced from the consecutively updated program variables.

  The maximum allowable deviation in amplitude is 1.5 (deviation range of the floating-point type).

  The amplitude reference value is 10.0 (user-defined reference value of the floating-point type).

  The number of samples for the sampling calculation is 3 (unsigned integer ranging from 1 to 3000).

  The sampling period is 1 (data sampling is executed once every given number of running periods, with the default value of 1).

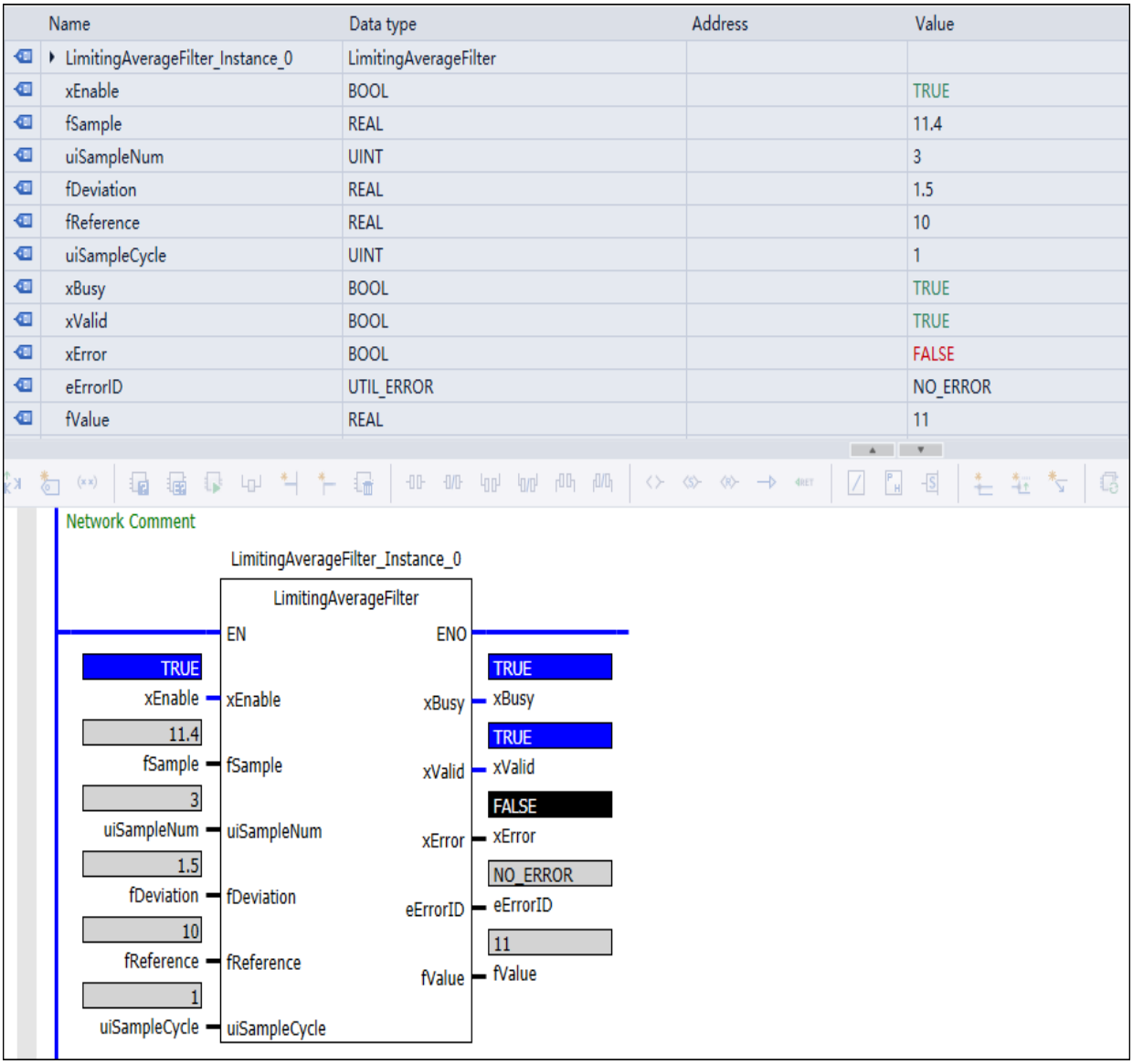
  When the input xEnable is set to TRUE, the function block starts sampling and filtering with energy flow effective. Set the initial reference value to 10.0 as the effective value. Compare the sampling value with the previous effective value. If the absolute difference between them is greater than the set maximum deviation value of 1.5, use the previous effective value as the current effective value. If it is less than or equal to the maximum deviation value, use the current value as the effective value. Then output the effective value as the filter value, and set the filter execution valid flag xValid to TRUE.

  ST:

Name	Data type	Address	Value
LimitingAverageFilter_Instance_0	LimitingAverageFilter		
xEnable	BOOL		TRUE
fSample	REAL		11.4
uiSampleNum	UINT		3
fDeviation	REAL		1.5
fReference	REAL		10
uiSampleCycle	UINT		1
xBusy	BOOL		TRUE
xValid	BOOL		TRUE
xError	BOOL		FALSE
eErrorID	UTIL_ERROR		NO_ERROR
fValue	REAL		11

```
LimitingAverageFilter_Instance_0(
xEnable  TRUE  := xEnable  TRUE  ,
fSample  11.4  := fSample  11.4  ,
uiSampleNum  3  := uiSampleNum  3  ,
fDeviation  1.5  := fDeviation  1.5  ,
fReference  10  := fReference  10  ,
uiSampleCycle  1  := uiSampleCycle  1  ,
xBusy  TRUE  => xBusy  TRUE  ,
xValid  TRUE  => xValid  TRUE  ,
xError  FALSE  => xError  FALSE  ,
eErrorID NO_ERROR  => eErrorID NO_ERROR ,
fValue  11  => fValue  11  );
```

LD:



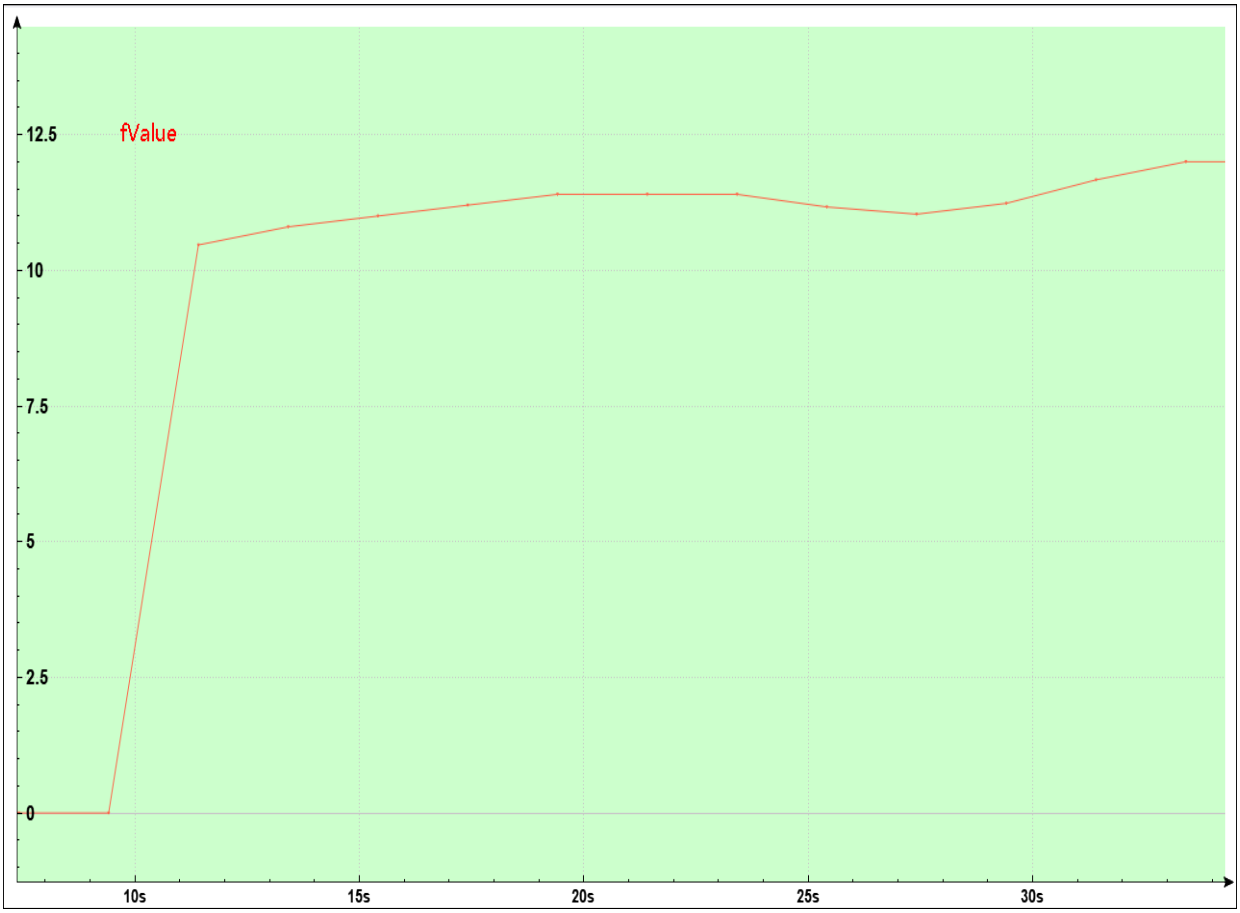
The sampling values of 12 consecutive periods are respectively 9.8, 10.8, 13.0, 12.5, 11.4, 8.4, 8.7, 8.9, 9.5, 10.7, 11.0, and 12.0.

After filtering, the values are respectively 0, 0, 10.47, 10.8, 11, 11.2, 11.4, 11.4, 11.4, 11.17, 11.03, and 11.23.

The following figure shows the curve before filtering.



The following figure shows the curve after filtering.

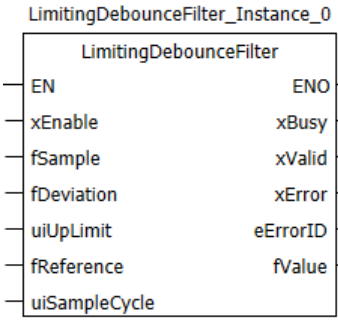


- **Precautions**
  - In the filter function block, the input parameter uiSampleCycle indicates the sampling period. When the value of uiSampleCycle is 1, sampling is performed every period. When the value of uiSampleCycle is 3, sampling is performed once every three periods. If the output of the function block flag LimitingAverageFilter.xValid is TRUE, the filter is valid. The filter output value of fValue is updated when the filter output is valid.
  - If the sampled data is stored in an array, the data needs to be put in the filter function block one by one and handled by the user.
  - For details about errors, see UTIL_ERROR.

18.2.8.6 LimitingDebounceFilter

Calculates the limiting debounce filter.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
Limiting Debounce Filter	Limiting debounce filter	FB		LimitingDebounceFilter_Instance_0(xEnable := , fSample := , fDeviation := , uiUpLimit := , fReference := , uiSampleCycle := , xBusy => , xValid => , xError => , eErrorID => , fValue =>);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	xEnable	Filter enable	Follows data type	FALSE	FALSE: Disabled. TRUE: Enabled.
	fSample	Input sampling value	Follows data type	0	Input sampling value
	fDeviation	Maximum allowable deviation for two consecutive samples	Follows data type	0	Maximum allowable deviation for two consecutive samples
	uiUpLimit	Filter counter upper limit	1-65535	1	Filter counter upper limit
	Reference	Input reference value	Follows data type	0	Input reference value
	uiSampleCycle	Input sampling period	1-1000	1	Input sampling period
Output	xBusy	Instruction executing	Follows data type	FALSE	TRUE: The function block is executing.
	xValid	Output active	Follows data type	FALSE	TRUE: Instruction execution is valid.
	xError	Fault	Follows data type	FALSE	TRUE: Instruction execution is abnormal.

Input/ Output	Variable	Name	Valid Range	Default Value	Description
	eErrorID	Fault ID	Follows data type	0	Refer to 18.2.8.11 Fault Code.
	fValue	Filter output value	Follows data type	0	Filter output value

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
xEnable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
fSample	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
fDeviation	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
uiUpLimit	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Reference	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
uiSampleCycle	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xValid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eErrorID	UTIL_ERROR. For details, see 18.2.8.11 Fault Code.																			
fValue	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

- **Function**

  When xEnable is set to TRUE, the sampling variable fSample is input based on the given scan period count uiSampleCycle, filter counter upper limit uiUpLimit, reference value fReference, and maximum allowable deviation fDeviation for two consecutive samples.

  Both limiting filter and debounce filter are used. Limiting filter is used before debounce filter.

  If the difference between the current value and the previous value is less than or equal to the value of fDeviation, the current value is valid and is used as the sampling value for debounce filter.

If the difference between the current value and the previous value is greater than the value of fDeviation, the current value is invalid. In this case, the current value is discarded, and the previous value is used as the sampling value for debounce filter.

Use the output of limiting filter as the sampling value for debounce filter, and compare it with the current effective value:

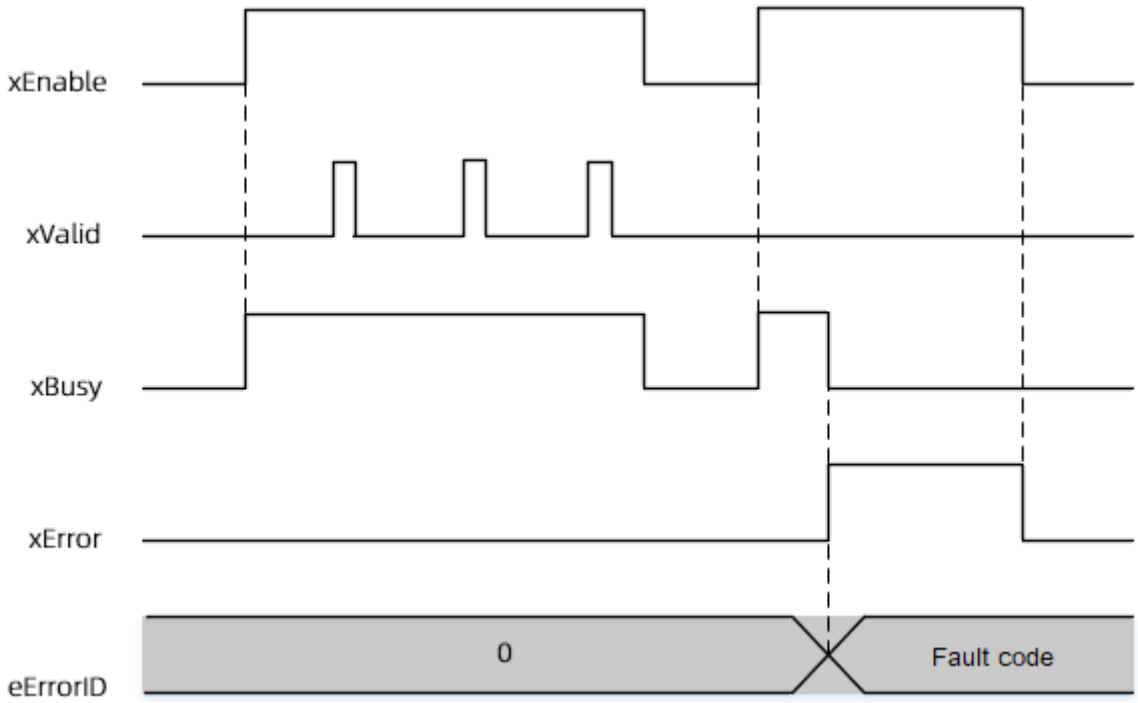
If the sampling value is equal to the current effective value, set the current count value to zero.

If the sampling value is not equal to the current effective value, increment the count value by 1 and check whether the count value exceeds the upper limit uiUpLimit; if so, replace the current filter value with the current sampling value, output the value, and reset the count value to zero.

Advantages: Combines the advantages of both limiting filter and debounce filter. It overcomes certain defects of debounce filter, preventing introduction of interference values into the system.

Disadvantages: Not suitable for rapidly changing parameters.

- **Sequence diagram**



- **Program example**

The sampling type is consecutive sampling. The raw data for sampling is sourced from the consecutively updated program variables.

The reference effective value of the sample is 10.0 (user-defined reference value of the floating-point type).

The maximum number of jitters for the sample value on the counter is 3 (unsigned integer).

The sampling period is 1 (data sampling is executed once every given number of running periods, with the default value of 1).

When the input xEnable is set to TRUE, the function block starts sampling and filtering with energy flow effective. Set the reference value to 10.0 as the effective value. Compare the sampling value with the current effective value. If they are the same, reset the jitter counter value to zero. If they are different, increase the counter value by 1. When the counter value exceeds the counter value upper

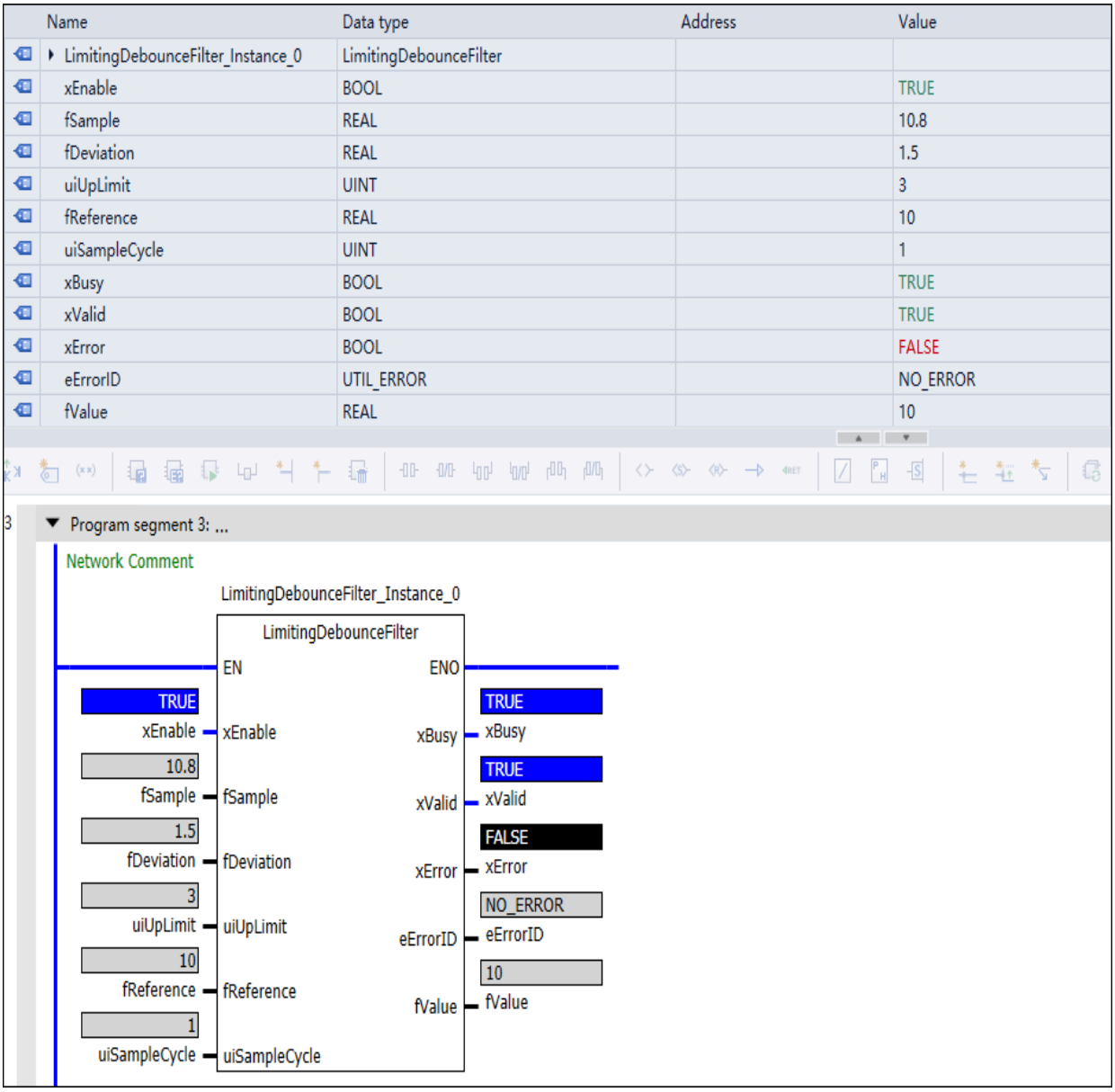
limit, replace the current effective value with the current sampling value as the filter effective value, output the effective value, and set the filter execution valid flag xValid to TRUE.

ST:

Name	Data type	Address	Value
LimitingDebounceFilter_Instance_0	LimitingDebounceFilter		
xEnable	BOOL		TRUE
fSample	REAL		10.8
fDeviation	REAL		1.5
uiUpLimit	UINT		3
fReference	REAL		10
uiSampleCycle	UINT		1
xBusy	BOOL		TRUE
xValid	BOOL		TRUE
xError	BOOL		FALSE
eErrorID	UTIL_ERROR		NO_ERROR
fValue	REAL		10

```
LimitingDebounceFilter_Instance_0(
xEnable  TRUE  := xEnable  TRUE  ,
fSample  10.8  := fSample  10.8  ,
fDeviation  1.5  := fDeviation  1.5  ,
uiUpLimit  3  := uiUpLimit  3  ,
fReference  10  := fReference  10  ,
uiSampleCycle  1  := uiSampleCycle  1  ,
xBusy  TRUE  => xBusy  TRUE  ,
xValid  TRUE  => xValid  TRUE  ,
xError  FALSE  => xError  FALSE  ,
eErrorID NO_ERROR  => eErrorID NO_ERROR ,
fValue  10  => fValue  10  );
```

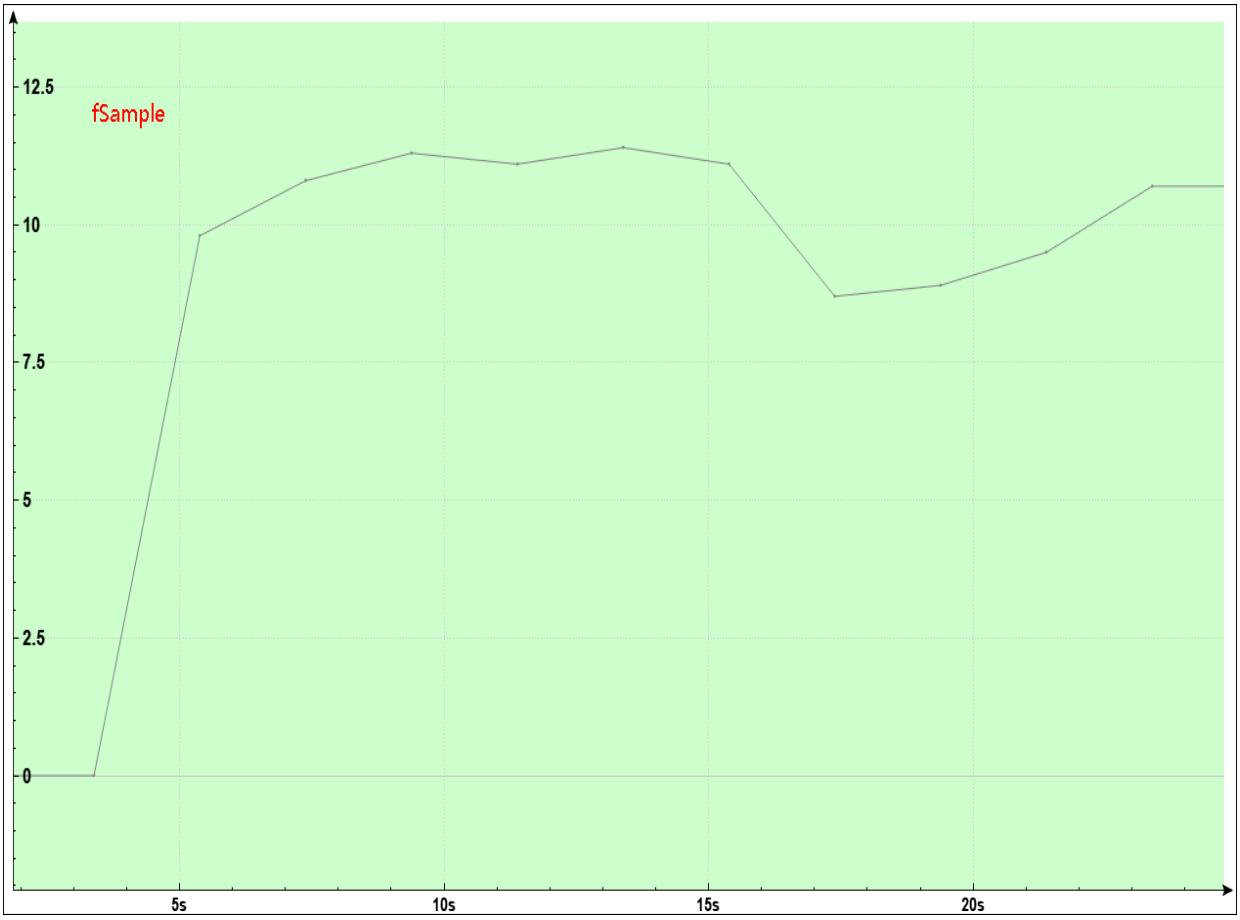
LD:



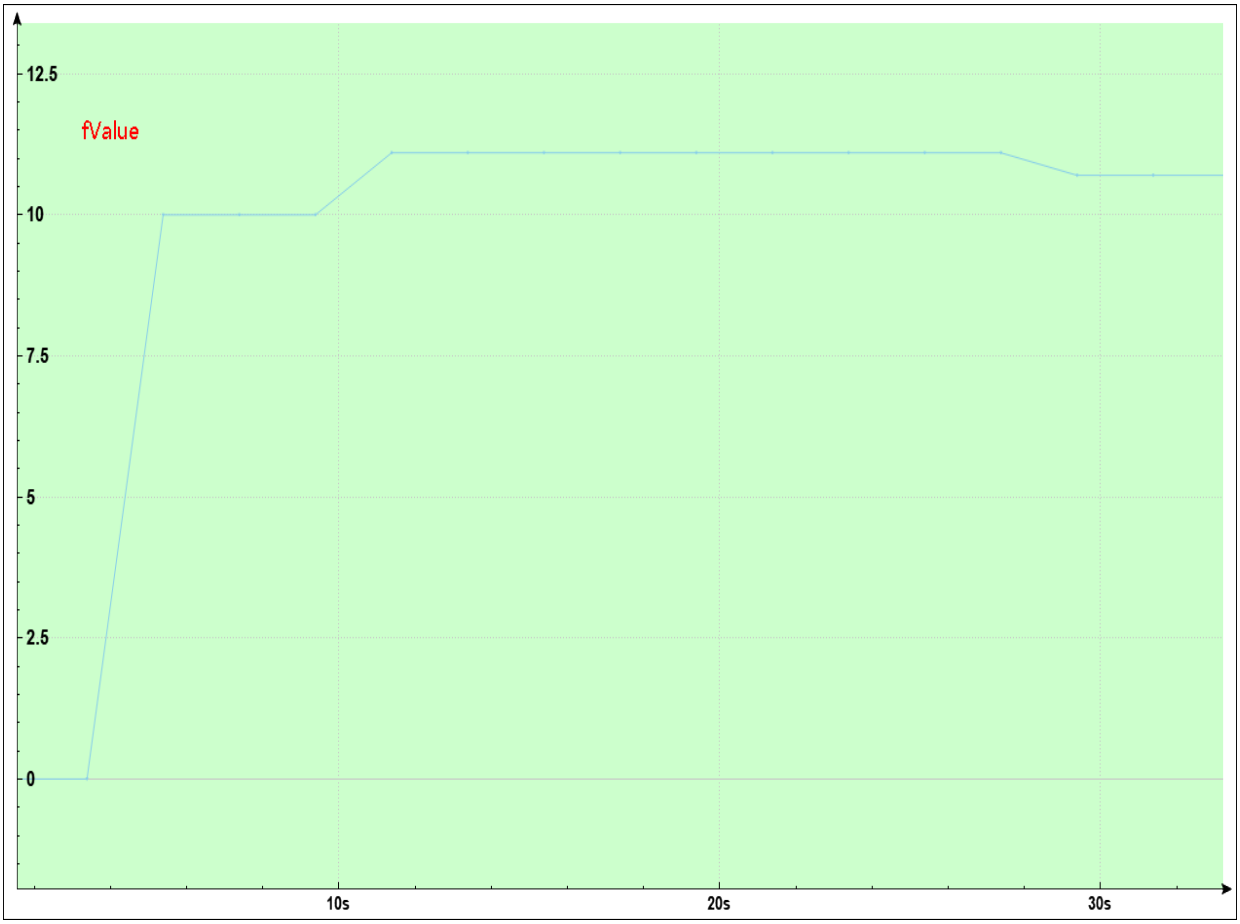
The sampling values of 10 consecutive periods are respectively 9.8, 10.8, 11.3, 11.1, 11.4, 11.1, 8.7, 8.9, 9.5, and 10.7.

After filtering, the values are respectively 10.0, 10.0, 10.0, 11.1, 11.1, 11.1, 11.1, 11.1, 11.1, and 11.1.

The following figure shows the curve before filtering.



The following figure shows the curve after filtering.



- **Precautions**
  - In the filter function block, the input parameter uiSampleCycle indicates the sampling period. When the value of uiSampleCycle is 1, sampling is performed every period. When the value of uiSampleCycle

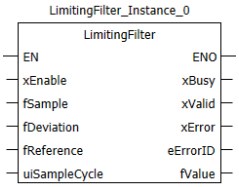
is 3, sampling is performed once every three periods. If the output of the function block flag LimitingDebounceFilter.xValid is TRUE, the filter is valid. The filter output value of fValue is updated when the filter output is valid.

- If the sampled data is stored in an array, the data needs to be put in the filter function block one by one and handled by the user.
- For details about faults, see UTIL_ERROR.

18.2.8.7 LimitingFilter

Calculates the limiting filter.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LimitingFilter	Limiting filter	FB		LimitingFilter_Instance_0(xEnable := , fSample := , fDeviation := , fReference := , uiSampleCycle := , xBusy => , xValid => , xError => , eErrorID => , fValue =>);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	xEnable	Filter enable	Follows data type	FALSE	FALSE: Disabled. TRUE: Enabled.
	fSample	Input sampling value	Follows data type	0	Input sampling value
	fDeviation	Maximum allowable deviation for two consecutive samples	Follows data type	0	Maximum allowable deviation for two consecutive samples
	Reference	Input reference value	Follows data type	0	Input reference value

Input/ Output	Variable	Name	Valid Range	Default Value	Description
	uiSampleCycle	Input sampling period	1-1000	1	Input sampling period
Output	xBusy	Instruction executing	Follows data type	FALSE	TRUE: The function block is executing.
	xValid	Output active	Follows data type	FALSE	TRUE: Instruction execution is valid.
	xError	Fault	Follows data type	FALSE	TRUE: Instruction execution is abnormal.
	eErrorID	Fault ID	Follows data type	0	Refer to 18.2.8.11 Fault Code.
	fValue	Filter output value	Follows data type	0	Filter output value

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
xEnable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
fSample	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
fDeviation	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Reference	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
uiSampleCycle	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xValid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eErrorID	UTIL_ERROR. For details, see 18.2.8.11 Fault Code.																			
fValue	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

- **Function**

  When xEnable is set to TRUE, the sampling variable fSample is input based on the given scan period count uiSampleCycle, reference value fReference, and maximum allowable deviation fDeviation for two consecutive samples.

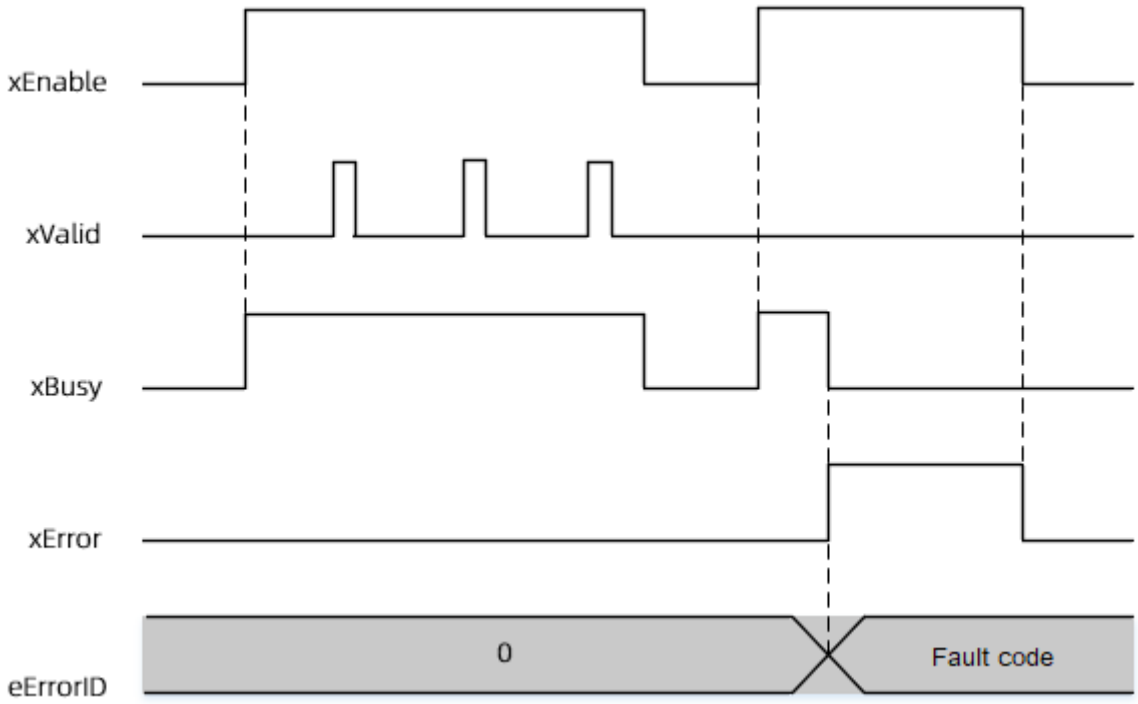
  Each time a new sample is obtained, if the difference between the current value and the previous value is less than or equal to the value of fDeviation, the current value is valid and is output as the filter value.

  If the difference between the current value and the previous value is greater than the value of fDeviation, the current value is invalid. In this case, the current value is discarded, and the previous value is output as the filter value.

  Advantage: It effectively suppresses pulse interference caused by occasional factors.

  Disadvantage: It cannot suppress periodic interference, causing poor smoothness.

- **Sequence diagram**



- **Program example**

  The sampling type is consecutive sampling. The raw data for sampling is sourced from the consecutively updated program variables.

  The amplitude reference value is 10.0 (user-defined reference value of the floating-point type). The maximum allowable deviation in amplitude is 1.5 (deviation range of the floating-point type).

  The filter calculation period is 1 (the sampling filter is executed once every given number of running period, with the default value of 1).

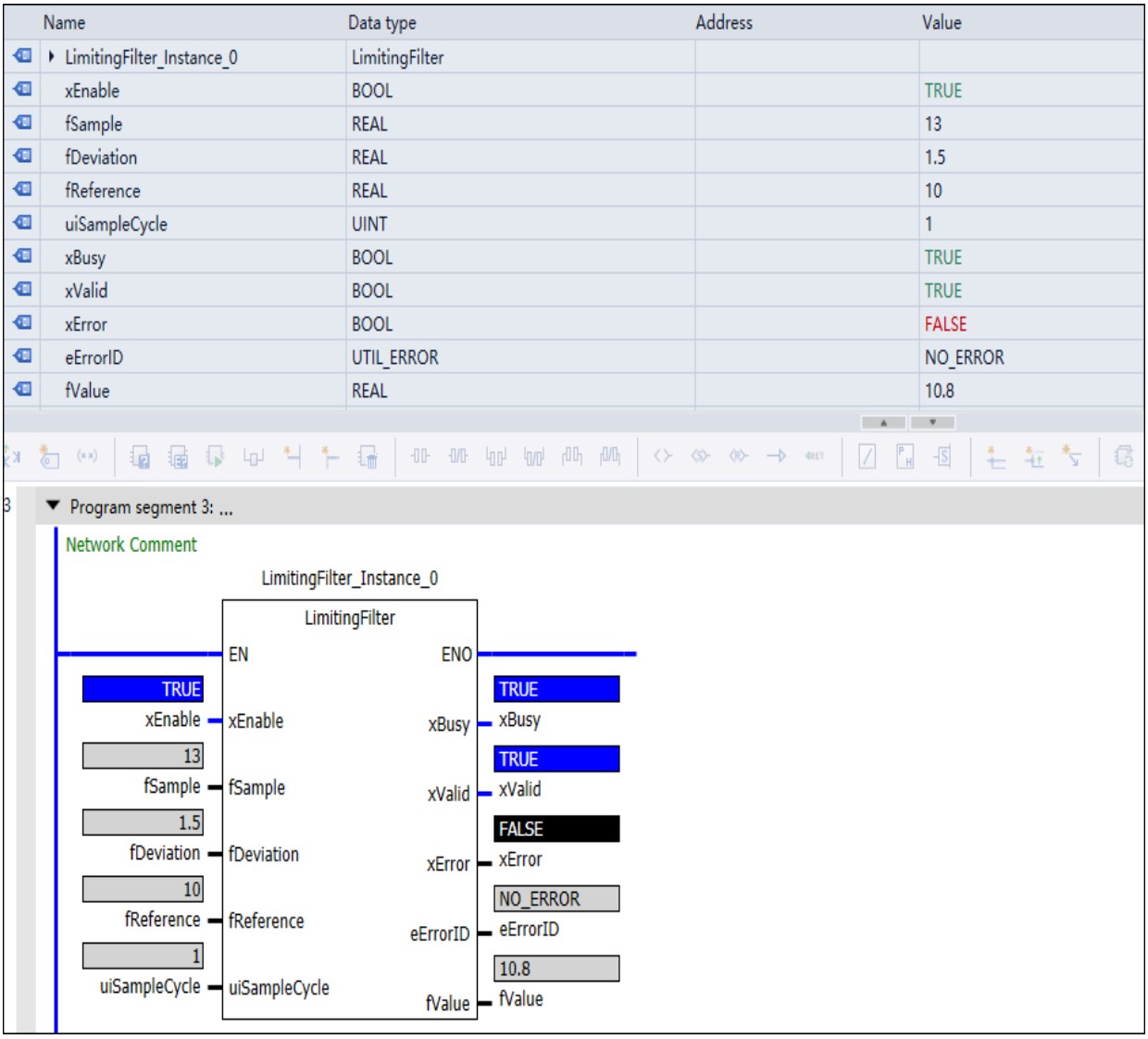
  When the input xEnable is set as energy flow effective, the limiting filter function block starts the sampling filter. Set the initial reference value to 10.0 as the effective value. Compare the sampling value with the previous effective value. If the absolute difference between them is greater than the set maximum deviation value of 1.5, use the previous effective value as the current effective value. If the absolute difference is less than or equal to the maximum deviation value, use the current value as the effective value. Then output the effective value as the filter value, and set the filter execution valid flag xValid to TRUE.

  ST:

Name	Data type	Address	Value
LimitingFilter_Instance_0	LimitingFilter		
xEnable	BOOL		TRUE
fSample	REAL		13
fDeviation	REAL		1.5
fReference	REAL		10
uiSampleCycle	UINT		1
xBusy	BOOL		TRUE
xValid	BOOL		TRUE
xError	BOOL		FALSE
eErrorID	UTIL_ERROR		NO_ERROR
fValue	REAL		10.8

```
LimitingFilter_Instance_0(
xEnable TRUE := xEnable TRUE,
fSample 13 := fSample 13,
fDeviation 1.5 := fDeviation 1.5,
fReference 10 := fReference 10,
uiSampleCycle 1 := uiSampleCycle 1,
xBusy TRUE => xBusy TRUE,
xValid TRUE => xValid TRUE,
xError FALSE =>xError FALSE ,
eErrorID NO_ERROR =>eErrorID NO_ERROR ,
fValue 10.8 =>fValue 10.8 );
```

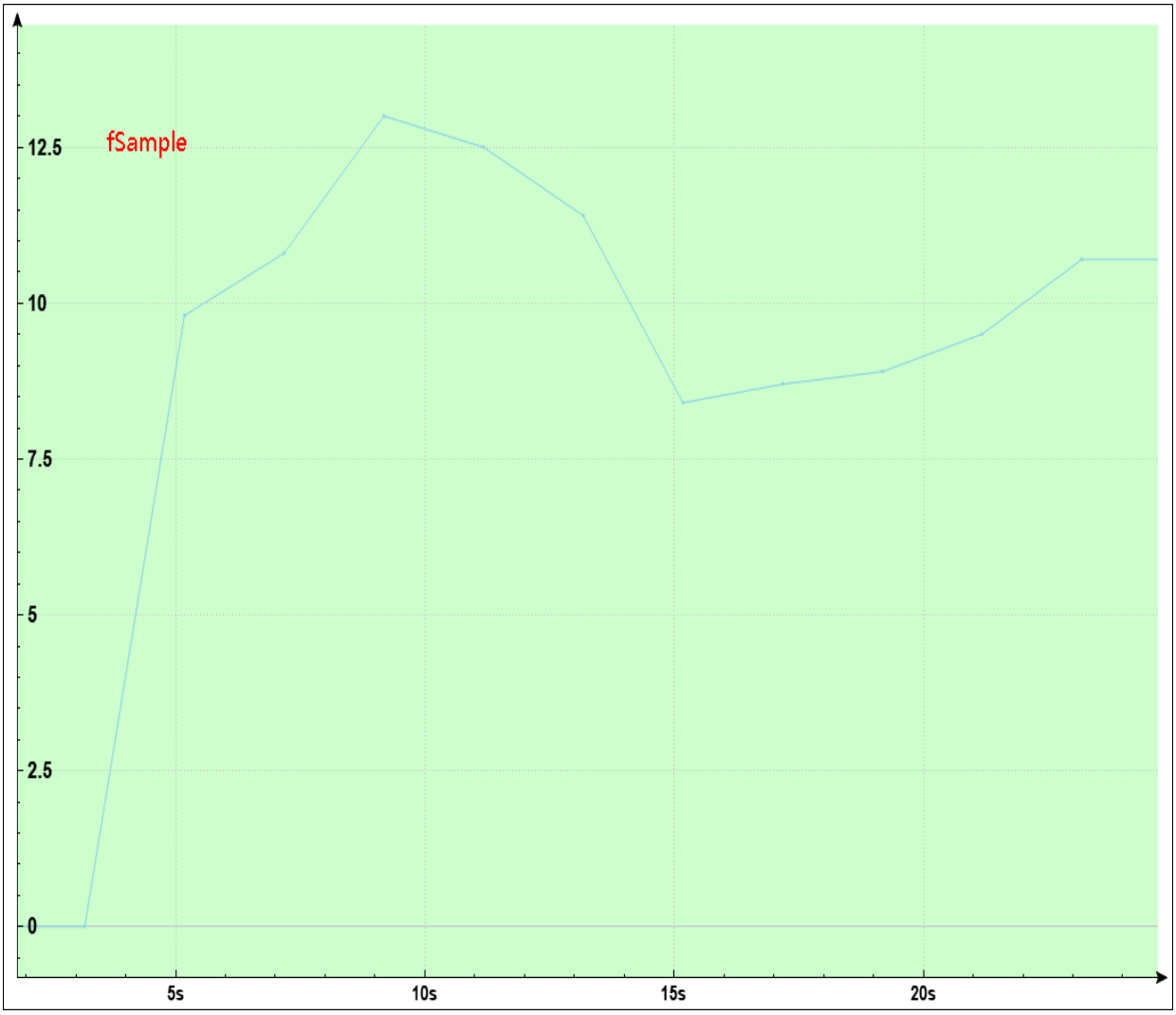
LD:



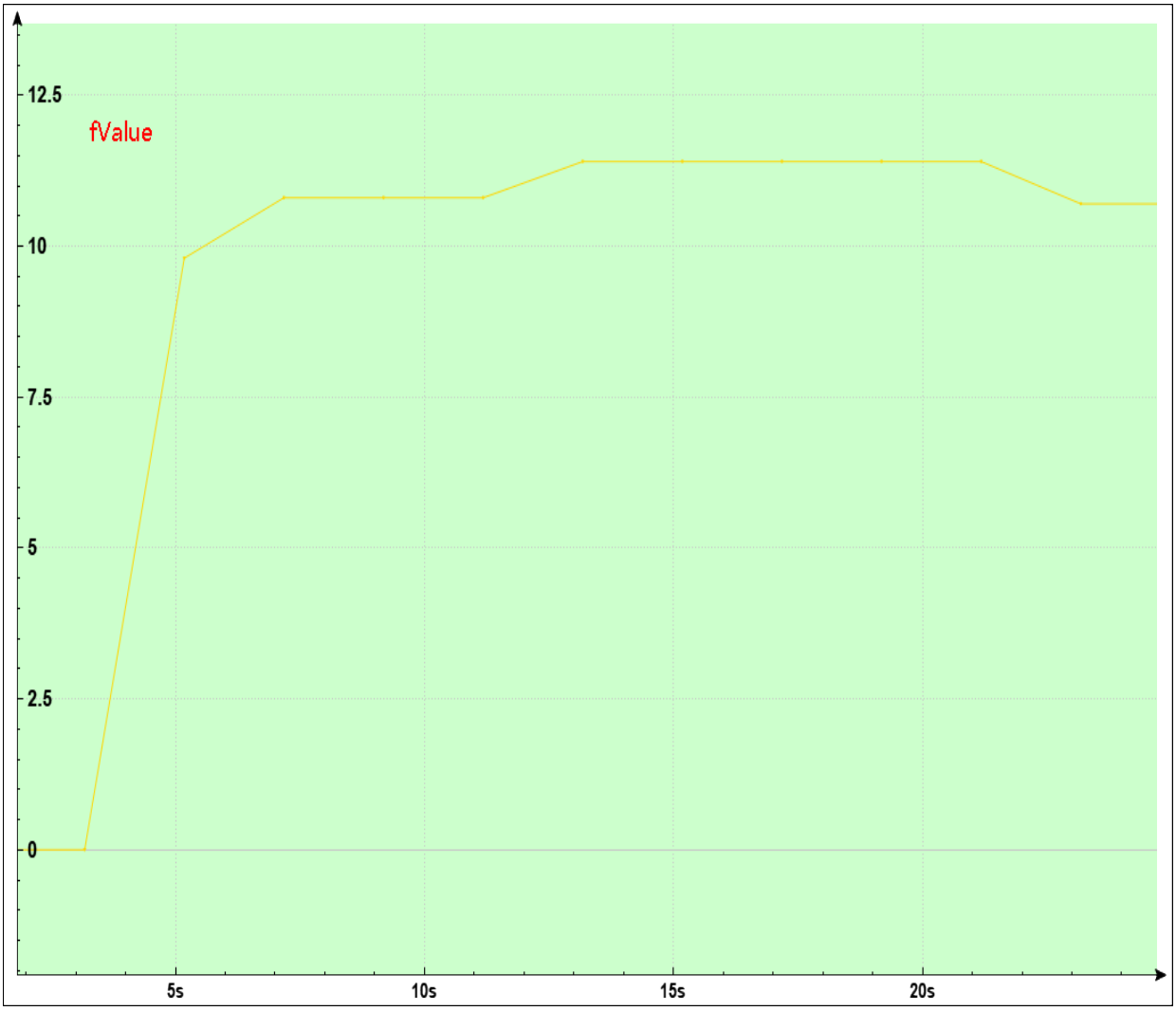
The sampling values of 10 consecutive periods are respectively 9.8, 10.8, 13.0, 12.5, 11.4, 8.4, 8.7, 8.9, 9.5, and 10.7.

After filtering, the values are respectively 9.8, 10.8, 10.8, 10.8, 11.4, 11.4, 11.4, 11.4, 11.4, and 10.7.

The following figure shows the curve before filtering.



The following figure shows the curve after filtering.

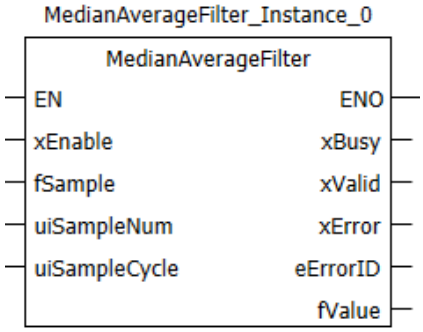


- **Precautions**
  - In the filter function block, the input parameter uiSampleCycle indicates the sampling period. When the value of uiSampleCycle is 1, sampling is performed every period. When the value of uiSampleCycle is 3, sampling is performed once every three periods. If the output of the function block flag MedianFilter.xValid is TRUE, the filter is valid. The filter output value of fValue is updated when the filter output is valid.
  - If the sampled data is stored in an array, the data needs to be put in the filter function block one by one and handled by the user.
  - For details about faults, see UTIL_ERROR.

18.2.8.8 MedianAverageFilter

Calculates the median average filter.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
Median Average Filter	Median average filter	FB		MedianAverageFilter_Instance_0(xEnable := , fSample := , uiSampleNum := , uiSampleCycle := , xBusy => , xValid => , xError => , eErrorID => , fValue =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	xEnable	Filter enable	Follows data type	FALSE	FALSE: Disabled. TRUE: Enabled.
	fSample	Input sampling value	Follows data type	0	Input sampling value
	uiSampleNum	Input sampling count	3 to 3000	3	Input sampling count
	uiSampleCycle	Input sampling period	1-1000	1	Input sampling period
Output	xBusy	Instruction executing	Follows data type	FALSE	TRUE: The function block is executing.
	xValid	Output active	Follows data type	FALSE	TRUE: Instruction execution is valid.
	xError	Fault	Follows data type	FALSE	TRUE: Instruction execution is abnormal.
	eErrorID	Fault ID	Follows data type	0	Refer to 18.2.8.11 Fault Code.

Input/Output	Variable	Name	Valid Range	Default Value	Description
	fValue	Filter output value	Follows data type	0	Filter output value

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
xEnable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
fSample	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
uiSampleNum	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
uiSampleCycle	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xValid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eErrorID	UTIL_ERROR. For details, see 18.2.8.11 Fault Code.																			
fValue	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

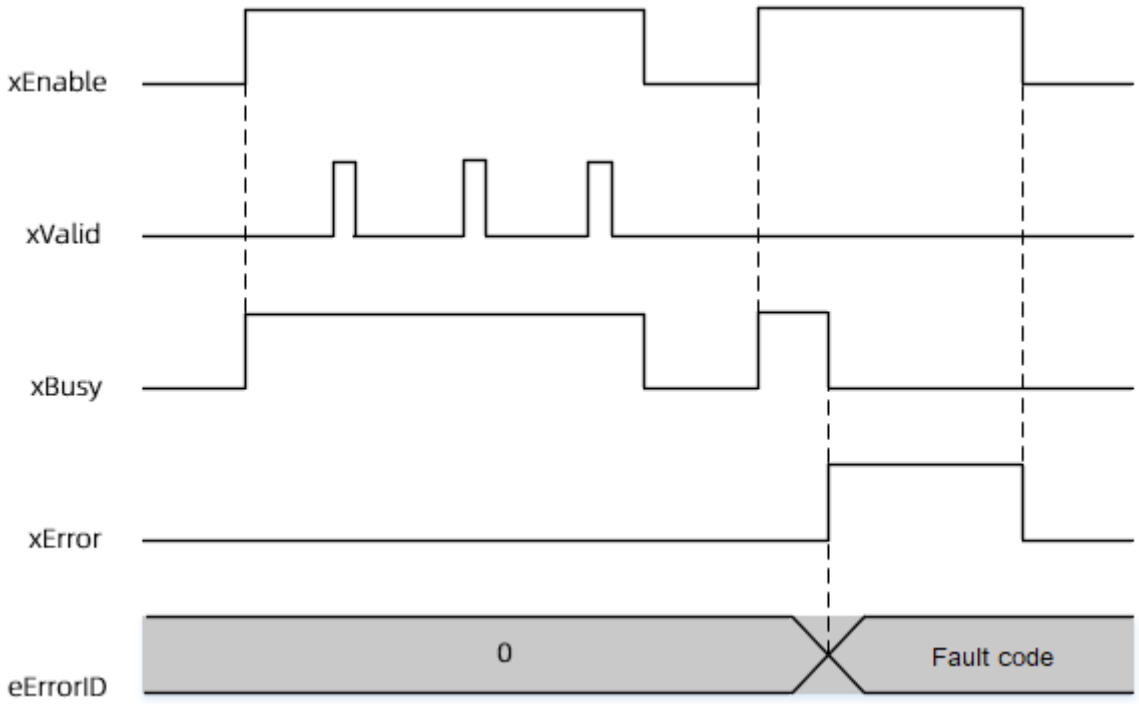
- **Function**

  When xEnable is set to TRUE, the sampling variable fSample is input based on the given scan period count uiSampleCycle and sampling count uiSampleNum. Sample data for consecutively uiSampleNum times, sort the sampled data, exclude the maximum value and minimum value, and average the rest values. This means using both median value filtering and one-dimensional arithmetic mean filtering. In general, uiSampleNum is set to 3 to 14.

  Advantages: Combines the advantages of both median value filtering and arithmetic average filtering. It can eliminate sampling deviations caused by occasional pulse interference, suppress periodic interferences, and provide high smoothness. It is suitable for systems with high-frequency oscillations.

  Disadvantages: It provides a slower calculation speed, similar to arithmetic average filtering, and it consumes a significant amount of RAM.

- **Sequence diagram**



- **Program example**

  The sampling type is consecutive sampling. The raw data for sampling is sourced from the consecutively updated program variables.

  The number of samples for the sampling calculation is 4 (unsigned integer ranging from 3 to 3000).

  The sampling period is 1 (data sampling is executed once every given number of running periods, with the default value of 1).

  When the input value of xEnable is TRUE, the function block starts sampling and filtering with energy flow effective. Set the number of samples for the sampling calculation to 4, indicating that filter calculation is performed once for every four sampling values. Sort the four sampling values by numerical value, exclude the maximum value and minimum value, and average the rest two values. Output the average value as the filter value, and set the filter execution valid flag xValid to TRUE.

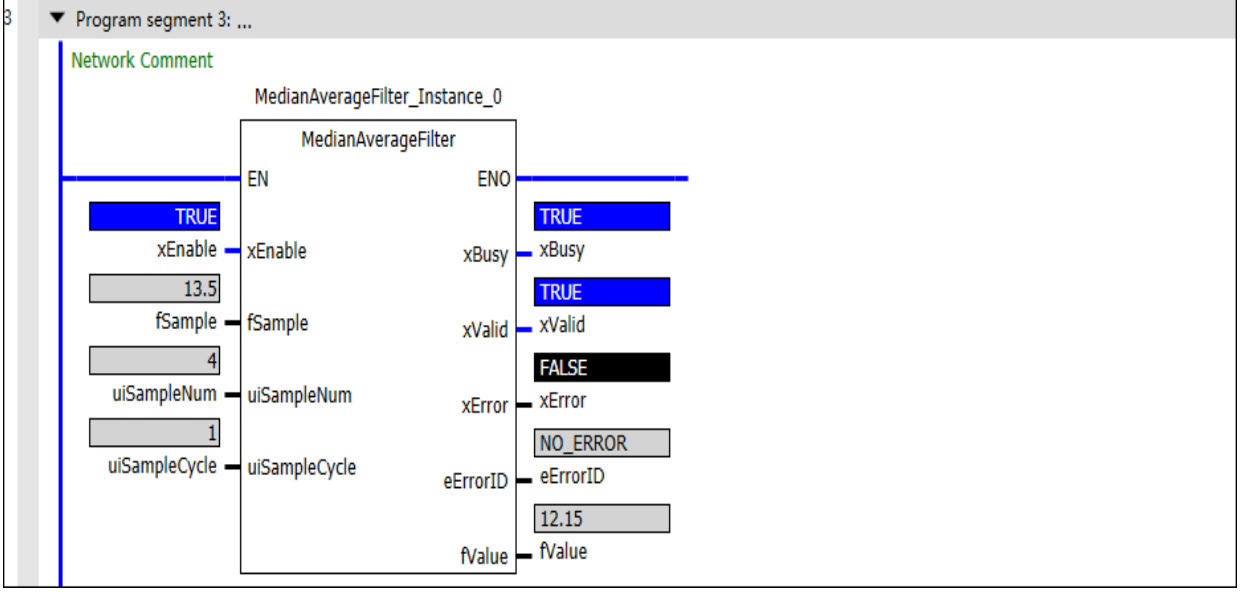
  ST:

	Name	Data type	Address	Value
	MedianAverageFilter_Instance_0	MedianAverageFilter		
	xEnable	BOOL		TRUE
	fSample	REAL		13.5
	uiSampleNum	UINT		4
	uiSampleCycle	UINT		1
	xBusy	BOOL		TRUE
	xValid	BOOL		TRUE
	xError	BOOL		FALSE
	eErrorID	UTIL_ERROR		NO_ERROR
	fValue	REAL		12.15

```
MedianAverageFilter_Instance_0(
xEnable  := xEnable  ,
fSample  := fSample  ,
uiSampleNum  := uiSampleNum  ,
uiSampleCycle  := uiSampleCycle  ,
xBusy  => xBusy  ,
xValid  => xValid  ,
xError  => xError  ,
eErrorID  => eErrorID  ,
fValue  => fValue  );
```

LD:

	Name	Data type	Address	Value
	MedianAverageFilter_Instance_0	MedianAverageFilter		
	xEnable	BOOL		TRUE
	fSample	REAL		13.5
	uiSampleNum	UINT		4
	uiSampleCycle	UINT		1
	xBusy	BOOL		TRUE
	xValid	BOOL		TRUE
	xError	BOOL		FALSE
	eErrorID	UTIL_ERROR		NO_ERROR
	fValue	REAL		12.15



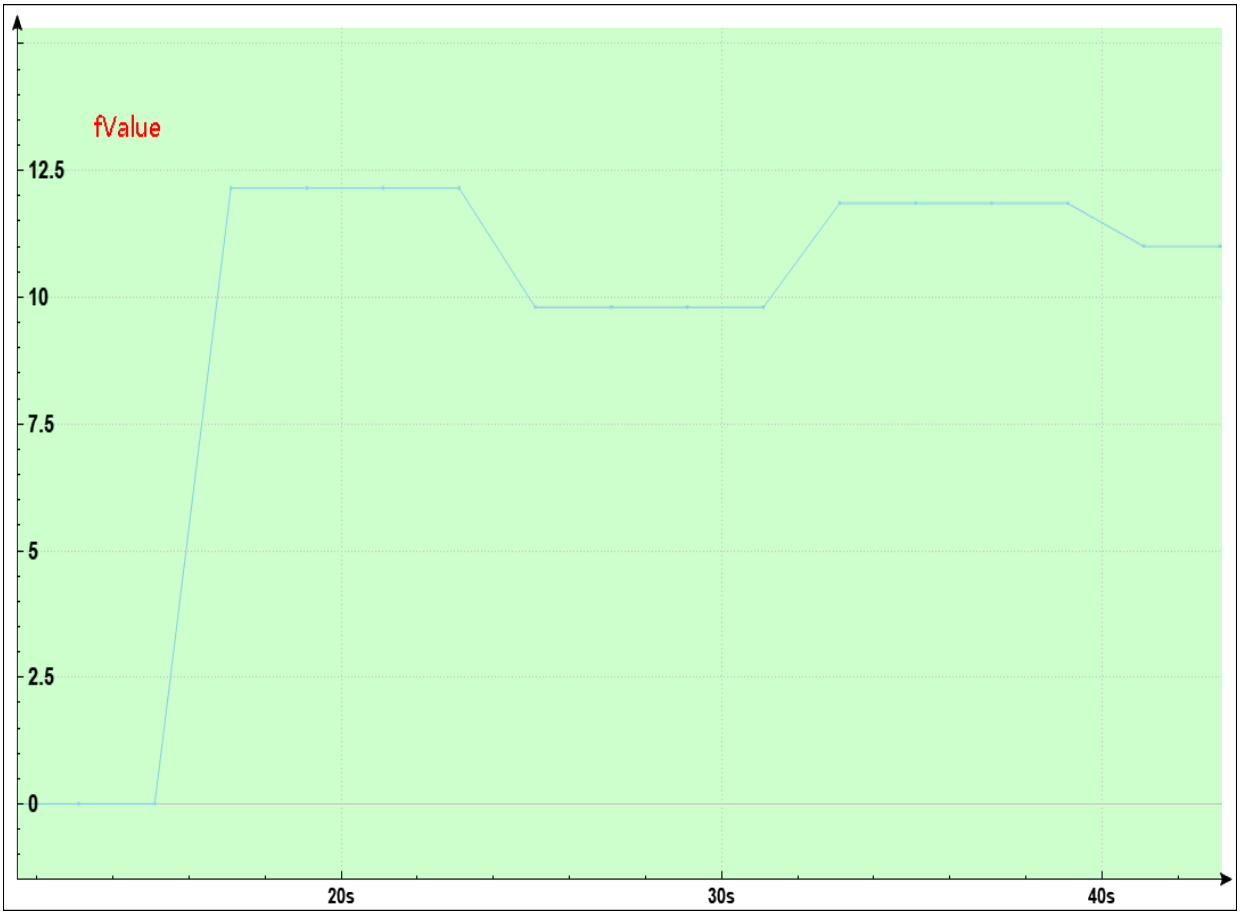
The sampling values of 12 consecutive periods are respectively 9.8, 19.8, 10.8, 13.5, 12.4, 4.4, 9.7, 9.9, 10.5, 17.1, 12.7, and 11.0.

After filtering, the values are respectively 0, 0, 0, 12.15, 12.15, 12.15, 12.15, 9.80, 9.80, 9.80, 9.80, and 11.85.

The following figure shows the curve before filtering.



The following figure shows the curve after filtering.

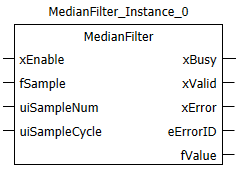


- **Precautions**
  - In the filter function block, the input parameter uiSampleCycle indicates the sampling period. When the value of uiSampleCycle is 1, sampling is performed every period. When the value of uiSampleCycle is 4, sampling is performed once every three periods. If the output of the function block flag MedianAverageFilter.xValid is TRUE, the filter is valid. The filter output value of fValue is updated when the filter output is valid.
  - If the sampled data is stored in an array, the data needs to be put in the filter function block one by one and handled by the user.
  - For details about faults, see UTIL_ERROR.

18.2.8.9 MedianFilter

Calculates the median filter.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
MedianFilter	Median filter	FB		MedianFilter_Instance_0(xEnable := , fSample := , uiSampleNum := , uiSampleCycle := , xBusy => , xValid => , xError => , eErrorID => , fValue =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	xEnable	Filter enable	Follows data type	FALSE	FALSE: Disabled. TRUE: Enabled.
	fSample	Input sampling value	Follows data type	0	Input sampling value
	uiSampleNum	Input sampling count	1-1000	1	Input sampling count
	uiSampleCycle	Input sampling period	1-1000	1	Input sampling period
Output	xBusy	Instruction executing	Follows data type	FALSE	TRUE: The function block is executing.
	xValid	Output active	Follows data type	FALSE	TRUE: Instruction execution is valid.
	xError	Fault	Follows data type	FALSE	TRUE: Instruction execution is abnormal.
	eErrorID	Fault ID	Follows data type	0	Refer to 18.2.8.11 Fault Code.

Input/Output	Variable	Name	Valid Range	Default Value	Description
	fValue	Filter output value	Follows data type	0	Filter output value

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
xEnable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
fSample	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
uiSampleNum	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
uiSampleCycle	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xValid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eErrorID	UTIL_ERROR. For details, see 18.2.8.11 Fault Code.																			

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
fValue	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

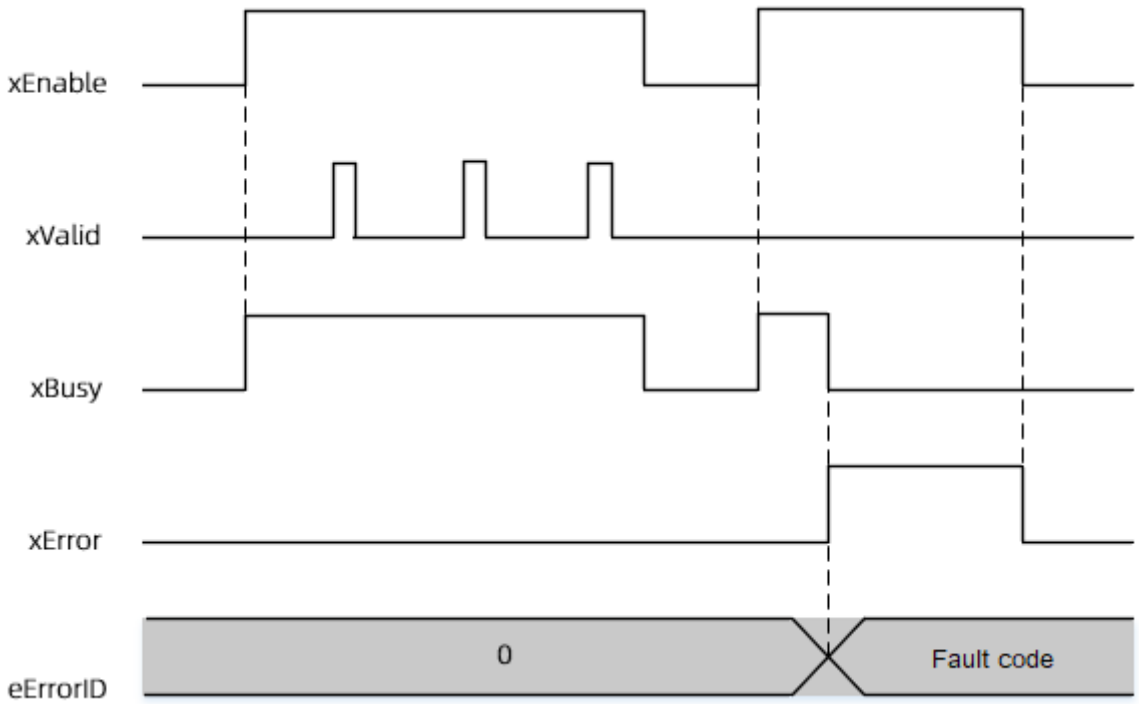
- **Function**

  When xEnable is set to TRUE, the sampling variable fValue is input based on the given scan period count uiSampleCycle and sampling count uiSampleNum. Sample data for consecutively uiSampleNum times, sort the sampled data by size, and take the middle value as the filter value for this period.

  Advantages: It effectively suppresses pulse interference caused by occasional factors, and provides good filtering effect for slowly changing parameters such as the temperature and liquid level.

  Disadvantages: Not suitable for rapidly changing parameters such as flow rate and speed.

- **Timing diagram**



- **Program example**

  The sampling type is consecutive sampling. The raw data for sampling is sourced from the consecutively updated program variables.

  The number of samples for the sampling calculation is 3 (unsigned integer ranging from 1 to 3000).

The sampling period is 1 (data sampling is executed once every given number of running periods, with the default value of 1).

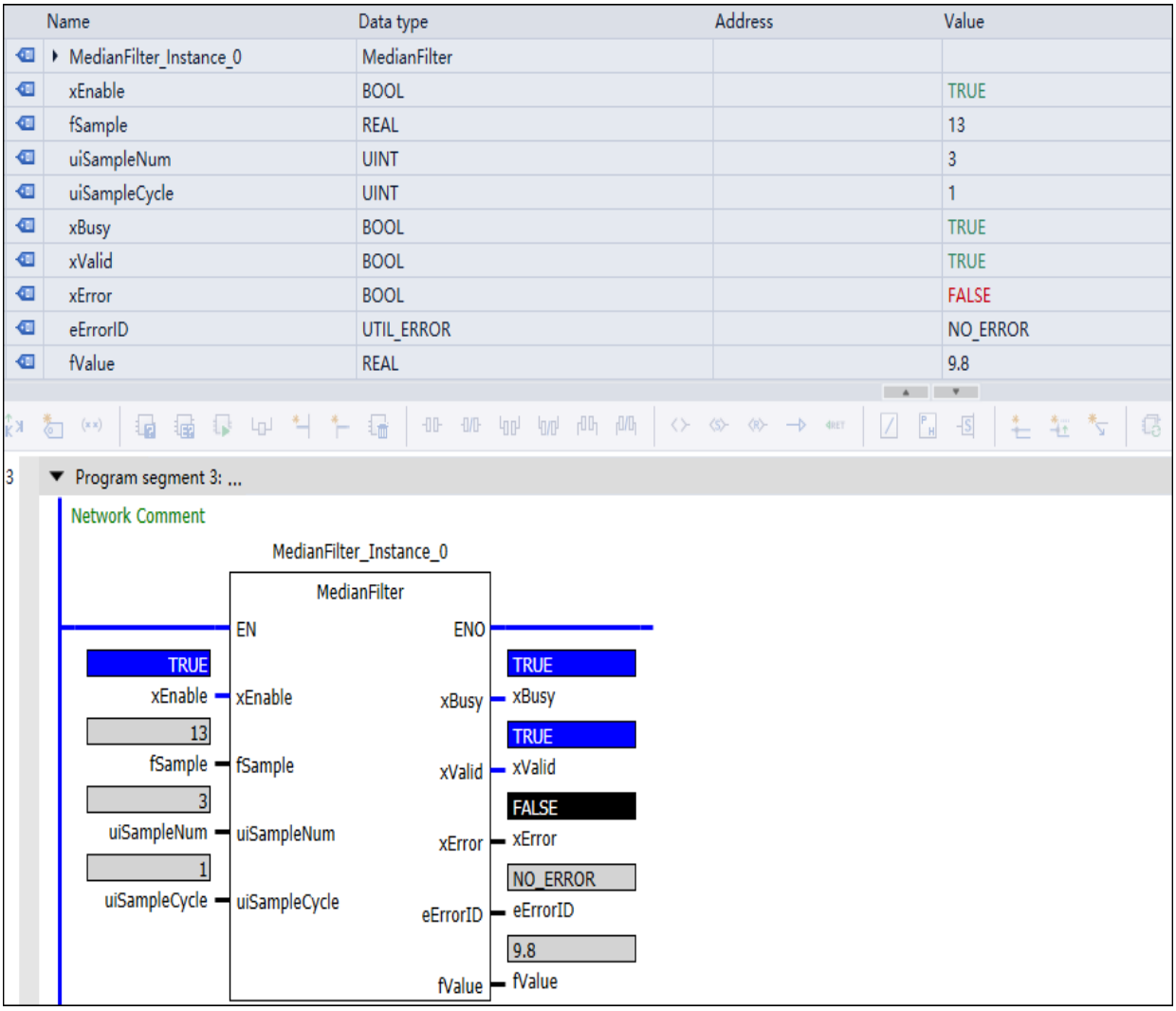
When the input xFilter is TRUE, the function block is enabled and begins sampling and filtering. With the number of samples set to 3, filter calculation is performed once for every three samples collected. Sort the three sampling values by numerical value, output the middle value as the filter value, and set the filter execution valid flag xValid to TRUE.

ST:

Name	Data type	Address	Value
MedianFilter_Instance_0	MedianFilter		
xEnable	BOOL		TRUE
fSample	REAL		13
uiSampleNum	UINT		3
uiSampleCycle	UINT		1
xBusy	BOOL		TRUE
xValid	BOOL		TRUE
xError	BOOL		FALSE
eErrorID	UTIL_ERROR		NO_ERROR
fValue	REAL		9.8

```
MedianFilter_Instance_0(
xEnable  TRUE  := xEnable  TRUE ,
fSample  13  := fSample  13 ,
uiSampleNum  3  := uiSampleNum  3 ,
uiSampleCycle  1  := uiSampleCycle  1 ,
xBusy  TRUE  => xBusy  TRUE ,
xValid  TRUE  => xValid  TRUE ,
xError  FALSE  => xError  FALSE ,
eErrorID NO_ERROR  => eErrorID NO_ERROR ,
fValue  9.8  => fValue  9.8 );
```

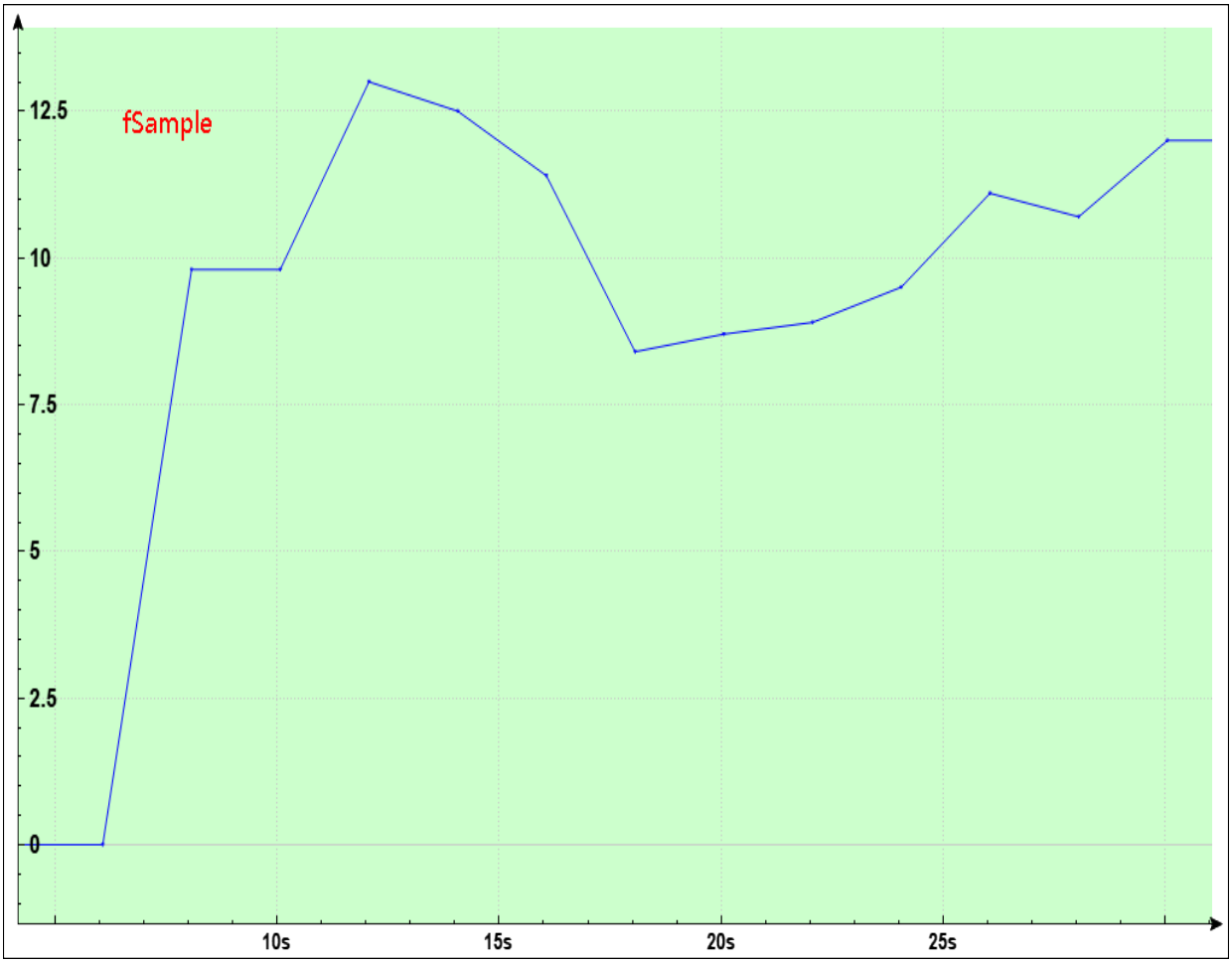
LD:



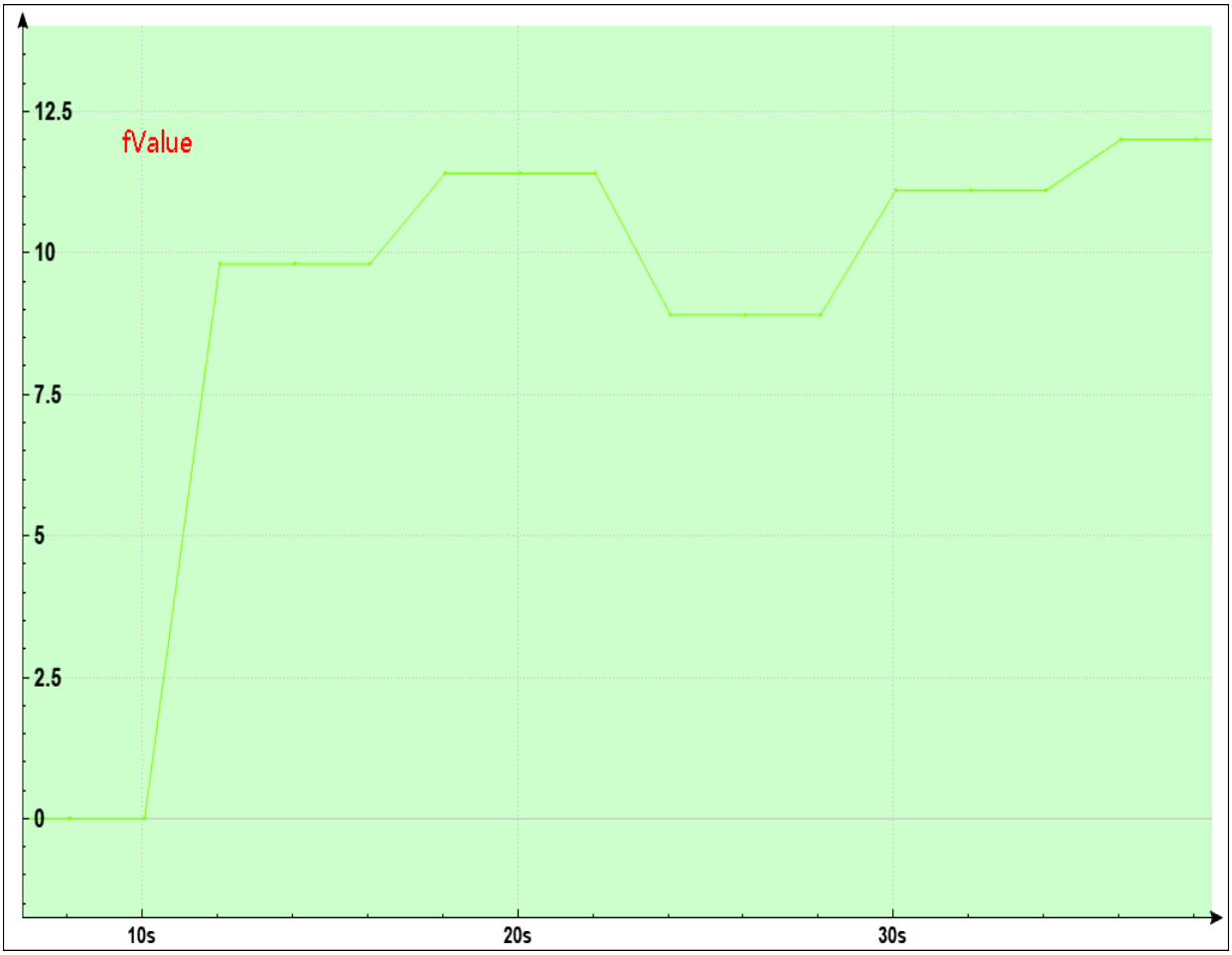
The sampling values of 12 consecutive periods are respectively 9.8, 9.8, 13.0, 12.5, 11.4, 8.4, 8.7, 8.9, 9.5, 11.1, 10.7, and 12.0.

After filtering, the values are respectively 0, 0, 9.8, 9.8, 9.8, 11.4, 11.4, 11.4, 8.9, 8.9, 8.9, and 11.1.

The following figure shows the curve before filtering.



The following figure shows the curve after filtering.

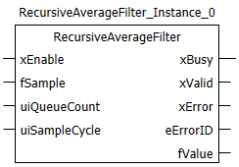


- **Precautions**
  - In the filter function block, the input parameter uiSampleCycle indicates the sampling period. When the value of uiSampleCycle is 1, sampling is performed every period. When the value of uiSampleCycle is 3, sampling is performed once every three periods. If the output of the function block flag MedianFilter.xValid is TRUE, the filter is valid. The filter output value of fValue is updated when the filter output is valid.
  - If the sampled data is stored in an array, the data needs to be put in the filter function block one by one and handled by the user.
  - For details about faults, see UTIL_ERROR.

18.2.8.10 RecursiveAverageFilter

Calculates the recursive average filter.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
Recursive Average Filter	Recursive average filter	FB		RecursiveAverageFilter_Instance_0(xEnable := , fSample := , uiQueueCount := , uiSampleCycle := , xBusy => , xValid => , xError => , eErrorID => , fValue =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	xEnable	Filter enable	Follows data type	FALSE	FALSE: Disabled. TRUE: Enabled.
	fSample	Input sampling value	Follows data type	0	Input sampling value
	uiQueueCount	Input queue length	1 to 3000	1	Input queue length
	uiSampleCycle	Input sampling period	1-1000	1	Input sampling period
Output	xBusy	Instruction executing	Follows data type	FALSE	TRUE: The function block is executing.
	xValid	Output active	Follows data type	FALSE	TRUE: Instruction execution is valid.
	xError	Fault	Follows data type	FALSE	TRUE: Instruction execution is abnormal.
	eErrorID	Fault ID	Follows data type	0	Refer to 18.2.8.11 Fault Code.

Input/Output	Variable	Name	Valid Range	Default Value	Description
	fValue	Filter output value	Follows data type	0	Filter output value

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
xEnable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
fSample	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
uiQueueCount	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
uiSampleCycle	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xValid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eErrorID	UTIL_ERROR. For details, see 18.2.8.11 Fault Code.																			

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
fValue	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

- **Function**

  When xEnable is set to TRUE, the sampling variable fSample is input based on the given scan period count uiSampleCycle and queue length uiQueueCount. Consider consecutively uiQueueCount samples as a queue with the fixed length of uiQueueCount. Place each newly sampled piece of data to the end of the queue, and discard the first piece of data in the queue, following the FIFO principle. Afterward, average N data samples in the queue for a new filter result.

  The larger uiQueueCount is, the higher the signal smoothness, but the lower the sensitivity.

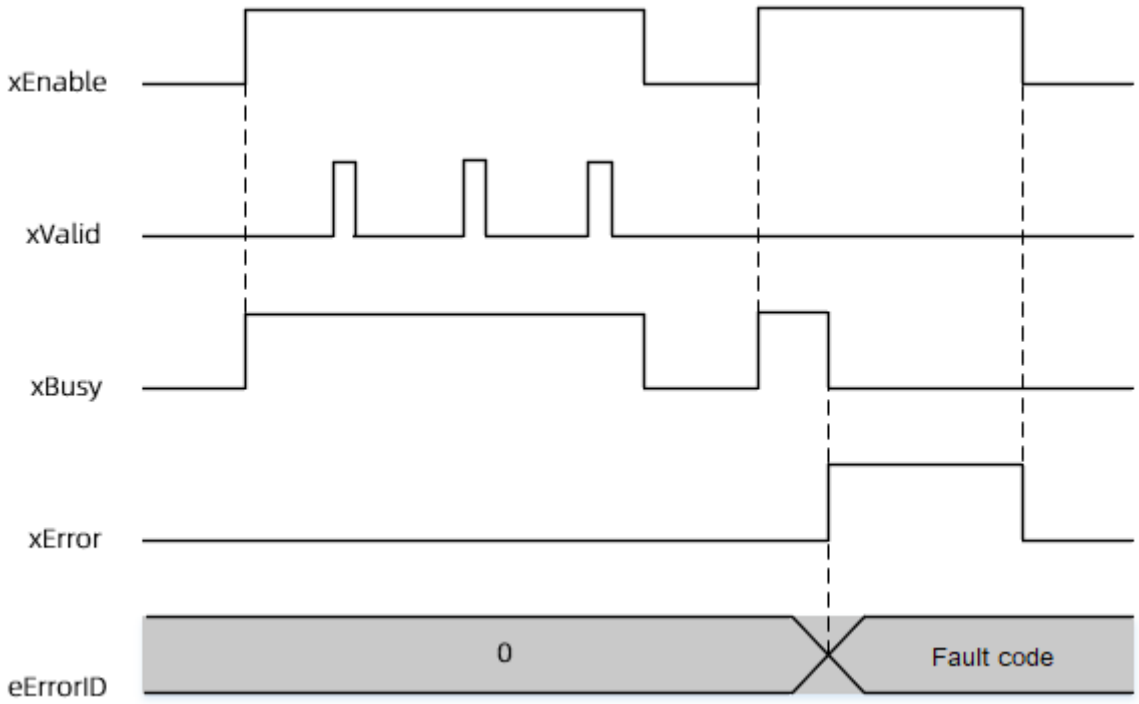
  The smaller uiQueueCount is, the lower the signal smoothness, but the higher the sensitivity.

  For general flow calculation, set uiQueueCount to 12. For pressure calculation, set uiQueueCount to 4. For liquid level calculation, set uiQueueCount to 4 to 12. For temperature calculation, set uiQueueCount to 4.

  Advantages: Suppresses periodic interference well and provides high smoothness. It is suitable for systems with high-frequency oscillations.

  Disadvantages: It has low sensitivity and provides poor suppression for occasional pulse interference. It is not effective in eliminating sampling deviations caused by pulse interference. It is not suitable for situations with significant pulse interference, and may consume a significant amount of RAM

- **Sequence diagram**



- **Program example**

  The sampling type is consecutive sampling. The raw data for sampling is sourced from the consecutively updated program variables.

  The number of samples for the sampling calculation is 3 (unsigned integer ranging from 1 to 3000).

  The sampling period is 1 (data sampling is executed once every given number of running periods, with the default value of 1).

  When the input value of xEnable is TRUE, the function block starts sampling and filtering with the energy flow effective. Set the number of sample queues for the sampling calculation to 3. Place each newly sampled piece of data to the end of the queue, and discard the first piece of data in the queue, following the FIFO principle. Average the three sampling values in the queue, output the result as the filter value, and set the filter execution valid flag xValid to TRUE.

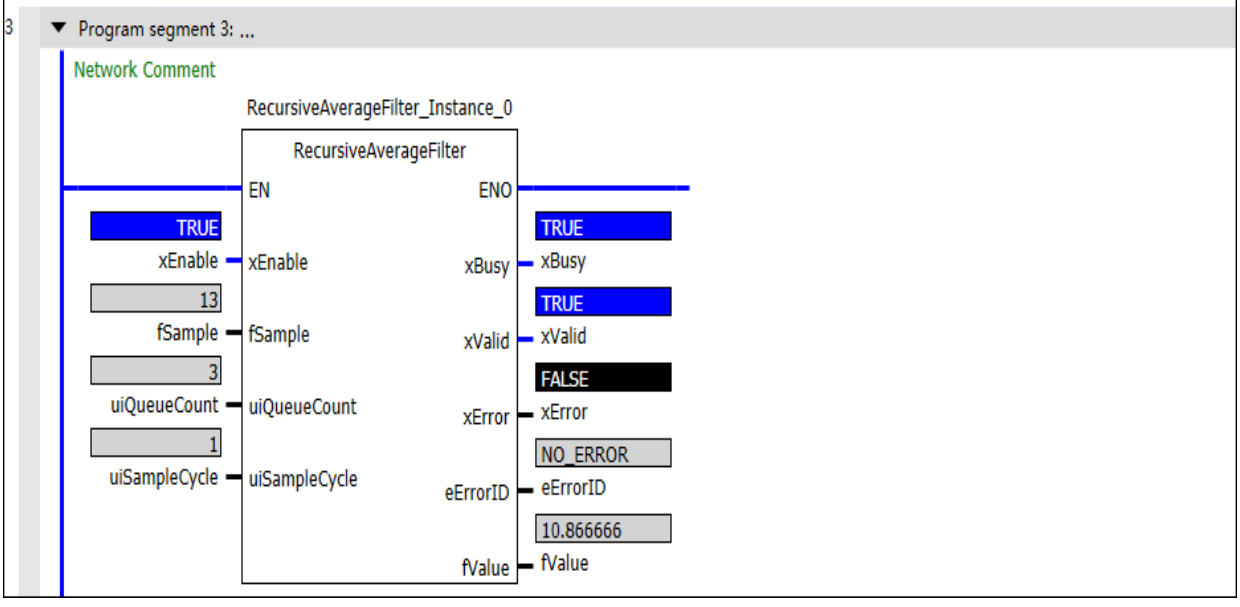
  ST:

Name	Data type	Address	Value
RecursiveAverageFilter_Instance_0	RecursiveAverageFilter		
xEnable	BOOL		TRUE
fSample	REAL		13
uiQueueCount	UINT		3
uiSampleCycle	UINT		1
xBusy	BOOL		TRUE
xValid	BOOL		TRUE
xError	BOOL		FALSE
eErrorID	UTIL_ERROR		NO_ERROR
fValue	REAL		10.866666

```
RecursiveAverageFilter_Instance_0(
xEnable  TRUE   := xEnable  TRUE  ,
fSample  13     := fSample  13    ,
uiQueueCount  3 := uiQueueCount  3 ,
uiSampleCycle  1 := uiSampleCycle  1 ,
xBusy  TRUE  => xBusy  TRUE  ,
xValid  TRUE  => xValid  TRUE  ,
xError  FALSE  => xError  FALSE  ,
eErrorID NO_ERROR  => eErrorID NO_ERROR ,
fValue  10.866666  => fValue  10.866666 );
```

LD:

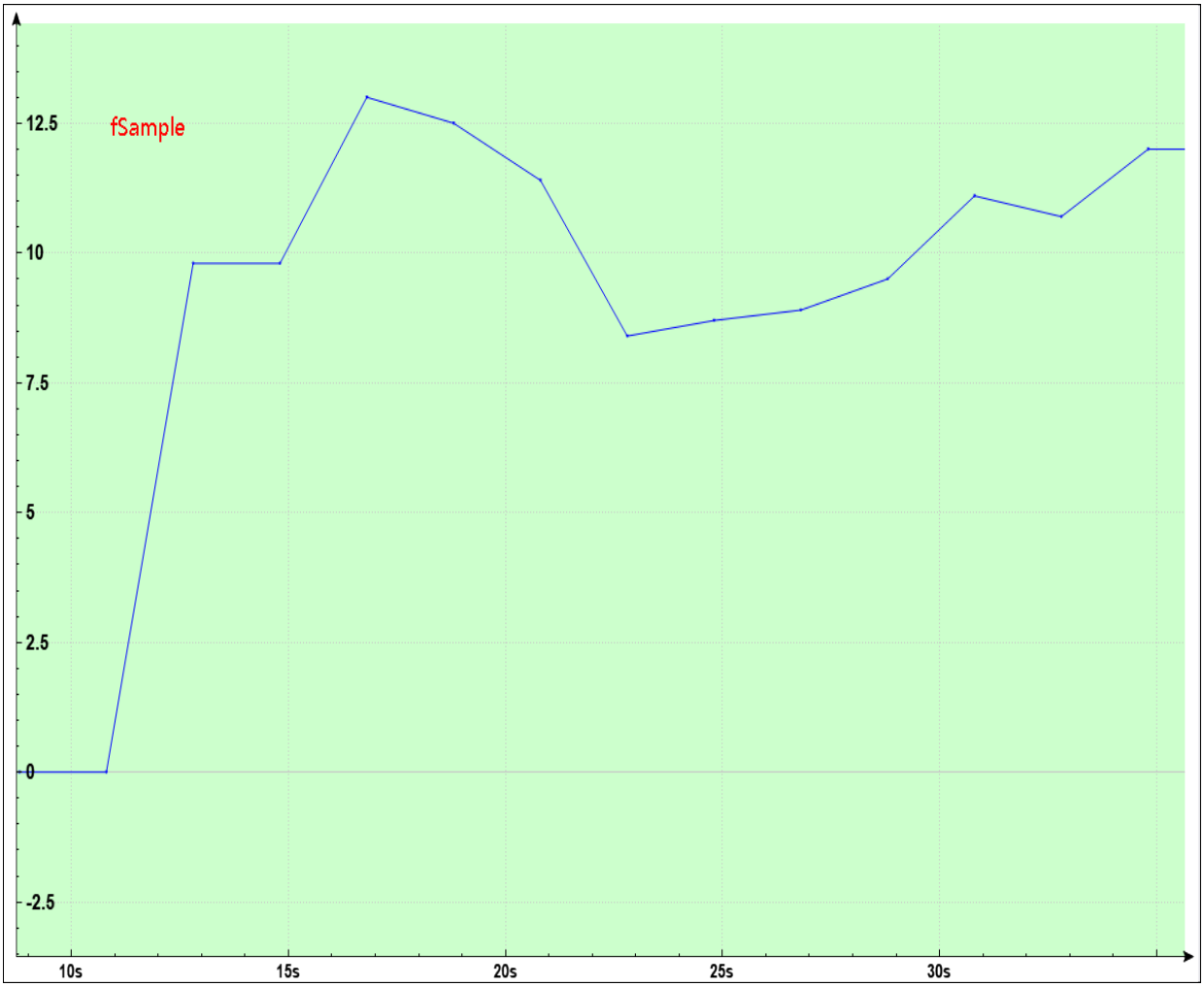
Name	Data type	Address	Value
RecursiveAverageFilter_Instance_0	RecursiveAverageFilter		
xEnable	BOOL		TRUE
fSample	REAL		13
uiQueueCount	UINT		3
uiSampleCycle	UINT		1
xBusy	BOOL		TRUE
xValid	BOOL		TRUE
xError	BOOL		FALSE
eErrorID	UTIL_ERROR		NO_ERROR
fValue	REAL		10.866666



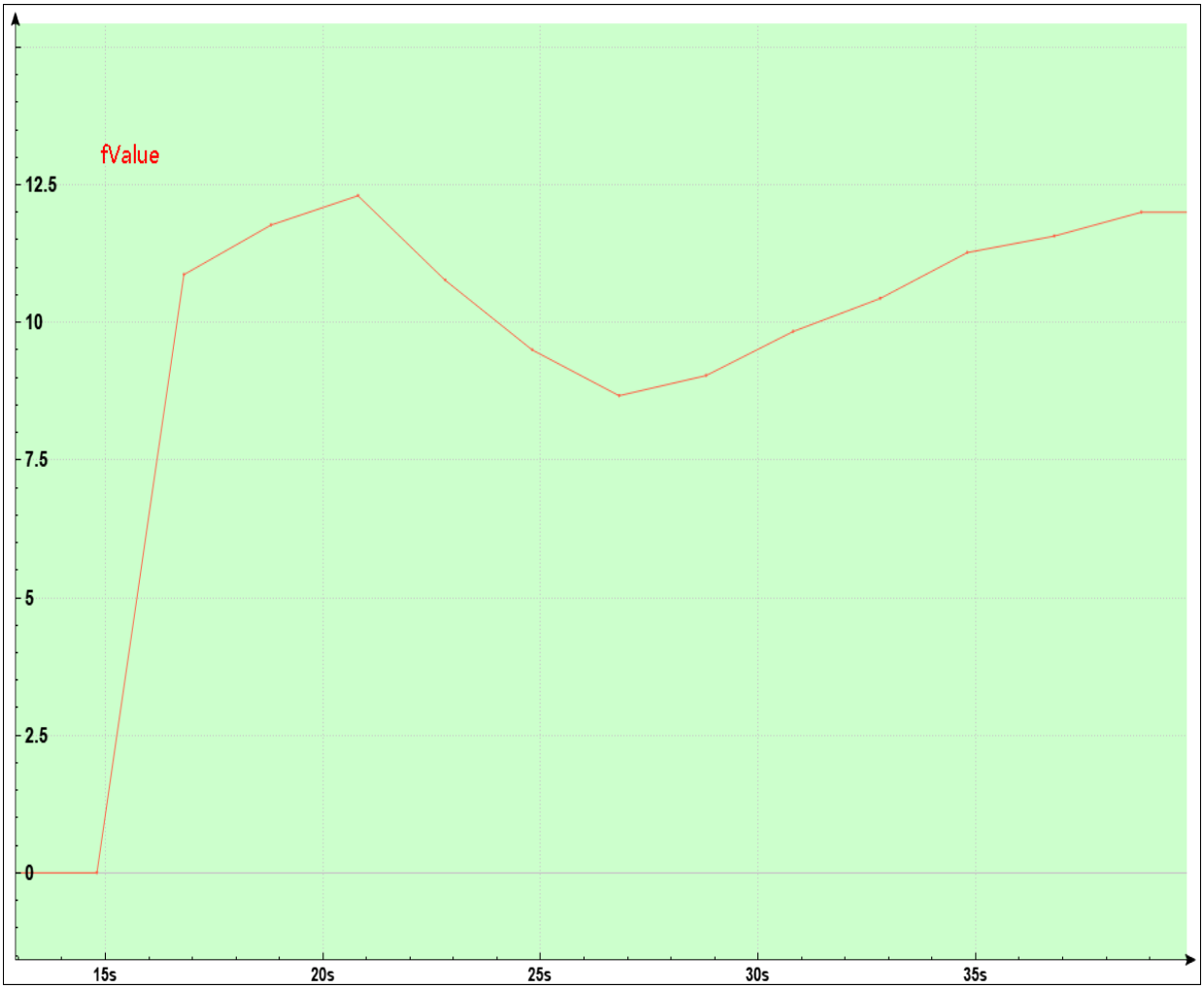
The sampling values of 12 consecutive periods are respectively 9.8, 9.8, 13.0, 12.5, 11.4, 8.4, 8.7, 8.9, 9.5, 11.1, 10.7, and 12.0.

After filtering, the values are respectively 0, 0, 10.87, 11.77, 12.30, 10.77, 9.50, 8.67, 9.03, 9.83, 10.43, and 11.27.

The following figure shows the curve before filtering.



The following figure shows the curve after filtering.



- **Precautions**
  - In the filter function block, the input parameter uiSampleCycle indicates the sampling period. When the value of uiSampleCycle is 1, sampling is performed every period. When the value of uiSampleCycle is 3, sampling is performed once every three periods. If the output of RecursiveAverageFilter.xValid is TRUE, the filter is valid. The filter output value of fValue is updated when the filter output is valid.
  - If the sampled data is stored in an array, the data needs to be put in the filter function block one by one and handled by the user.
  - For details about faults, see UTIL_ERROR.

18.2.8.11 Fault Codes

See the table below for the fault code ID of instruction.

Fault code	Value	Description
NO_ERROR	0	No fault.
DEVIATION_PARAMETER_SET_ERROR	1	Incorrect absolute value for the maximum deviation of two consecutive samples, which should be greater than 0
UPLIMIT_PARAMETER_SET_ERROR	2	Incorrect upper limit of the jitter count value

Fault code	Value	Description
MAF_SAMPLENUM_PARAMETER_SET_ERROR	3	Incorrect consecutive sampling number of median average filtering, which should be 3 to 3000
SAMPLENUM_PARAMETER_SET_ERROR	4	Incorrect consecutive sampling number, which should be 1 to 1000
QUEUECOUNT_PARAMETER_SET_ERROR	5	Incorrect length of the queue for storing sampling values
COEFFICIENT_PARAMETER_SET_ERROR	6	Incorrect first-order low-pass filter coefficient
WEIGHTED_PARAMETER_SET_ERROR	7	Incorrect weighting coefficients, which cannot be all 0

18.2.9 ASCII Code Conversion

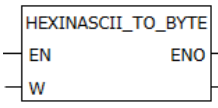
18.2.9.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.9.2 HEXinASCII_TO_BYTE	FC	Conversion from HEXinASCII to byte	Util	Library version 1.0.0.0 and later
18.2.9.3 BYTE_TO_HEX in ASCII	FC	Conversion from byte to HEXinASCII		Library version 1.0.0.0 and later
18.2.9.4 WORD_AS_STRING	FC	Conversion from word to string		Library version 1.0.0.0 and later

18.2.9.2 HEXinASCII_TO_BYTE

Converts the source data from the HEXinASCII format to bytes.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
HEXinASCII_TO_BYTE	Conversion from HEXinASCII to byte	FC		:= HEXinASCII_TO_BYTE(W :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	W	Input data	Follows data type	-	Input data

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	OUT	Output data	Follows data type	-	Output data.

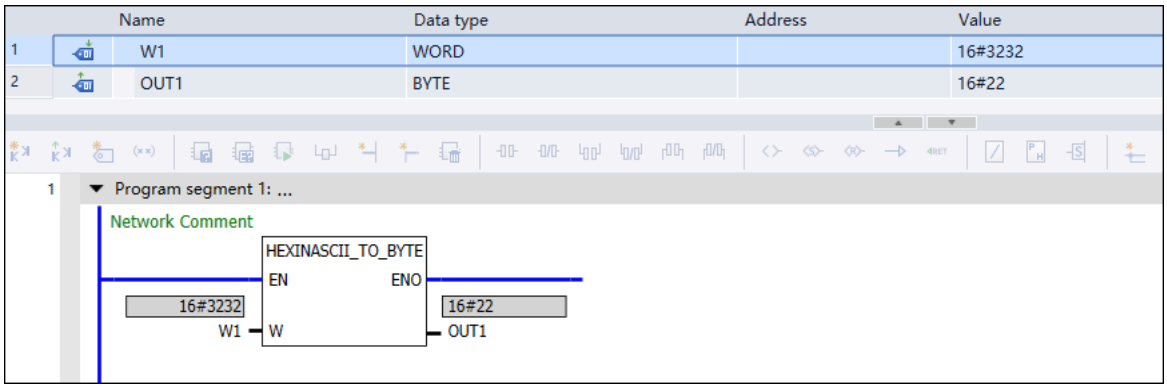
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
W	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

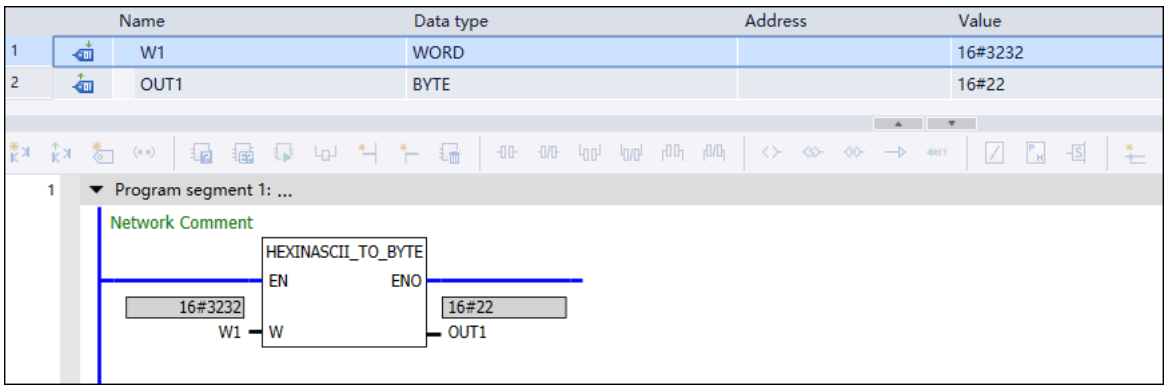
- **Program example**

  This instruction converts the source data from the HEXinASCII format to bytes.

  ST:



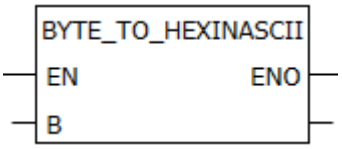
  LD:



18.2.9.3 BYTE_TO_HEXinASCII

Converts the source data from bytes to the HEXinASCII format.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BYTE_TO_HEXinASCII	Conversion from byte to HEXinASCII	FC		:= BYTE_TO_HEXinASCII(B :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	B	Input data	Follows data type	-	Input data
Output	OUT	Output data	Follows data type	-	Output data.

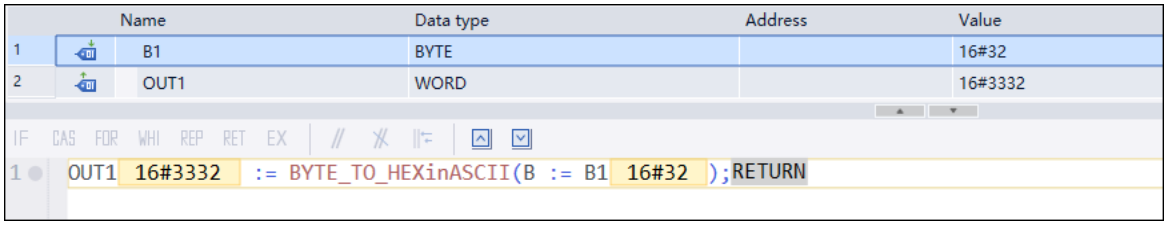
  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
B	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

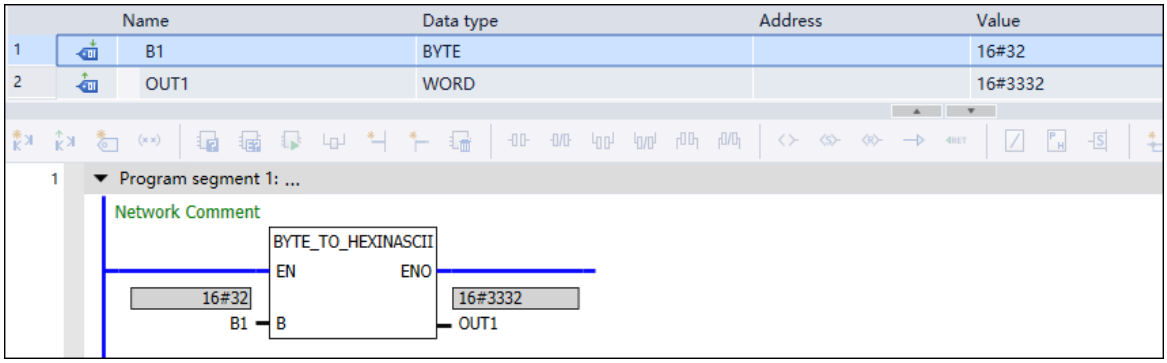
- **Program example**

  This instruction converts the source data from bytes to the HEXinASCII format.

  ST:



  LD:



18.2.9.4 WORD_AS_STRING

Converts the source data from words to strings.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
WORD_AS_STRING	Conversion from word to string	FC	WORD_AS_STRING EN ENO W ORDER	:= WORD_AS_STRING(W := , ORDER :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	W	Input data	Follows data type	-	Input data
	ORDER	High/Low-order bit conversion	Follows data type	-	Determines whether to perform high/low-order bit conversion
Output	OUT	Conversion result	Follows data type	-	Conversion result

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
W	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

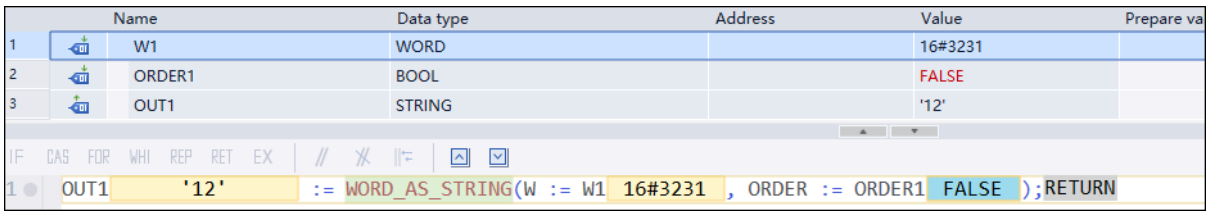
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
ORDER	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

- **Program example**

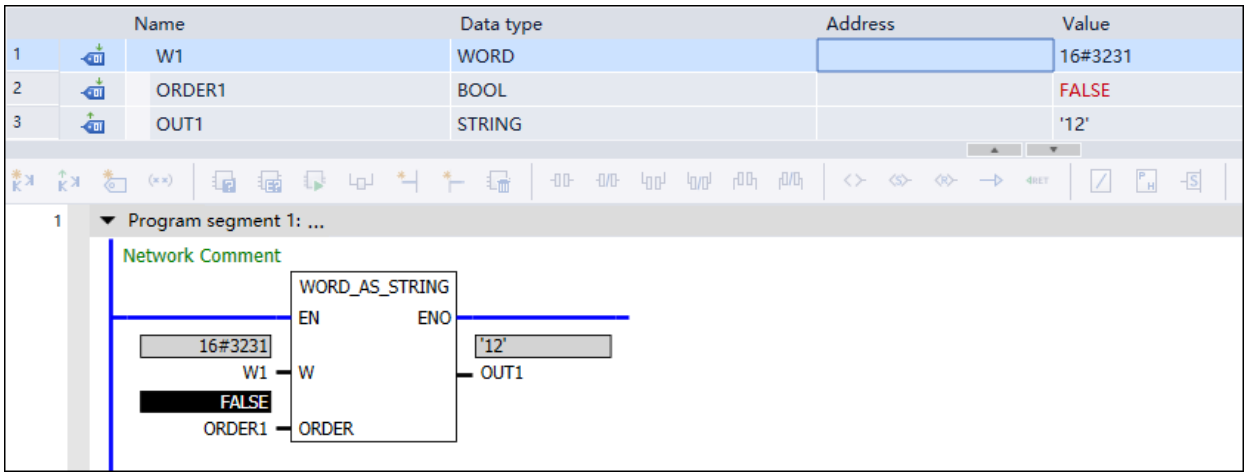
  This instruction converts the source data from words to strings. The ORDER pin determines whether to perform high/low-order bit conversion.

  When ORDER = FALSE:

  ST:

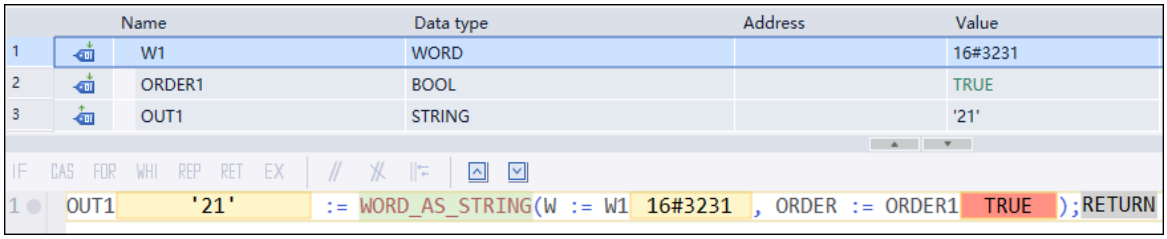


  LD:

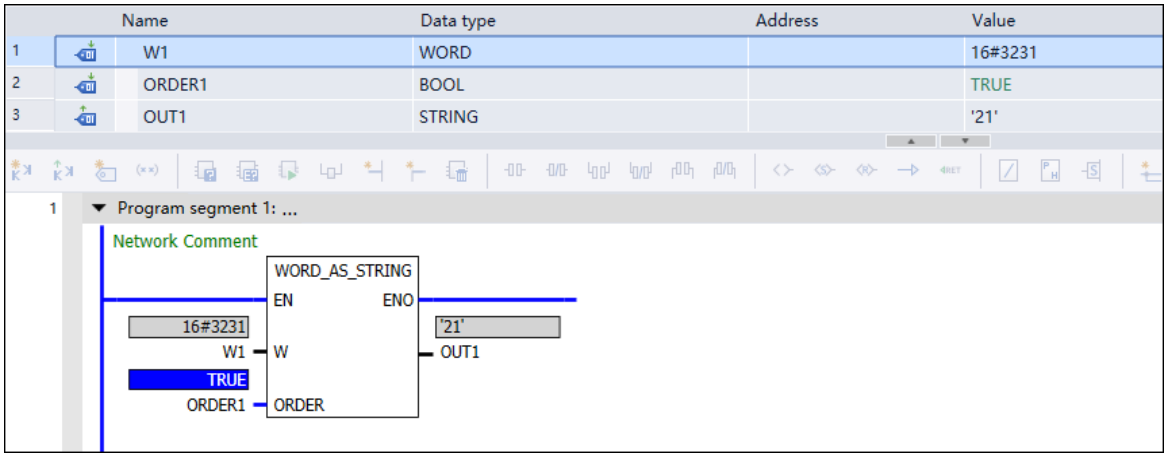


  When ORDER = TRUE:

  ST:



  LD:



- **Precautions**

  When the ORDER pin is FALSE, this instruction only converts WORD-type ASCII code data to STRING type data, without modifying the numerical value. When the ORDER pin is TRUE, the high/low-order bits are switched.

18.2.10 Table Range

18.2.10.1 Instruction List

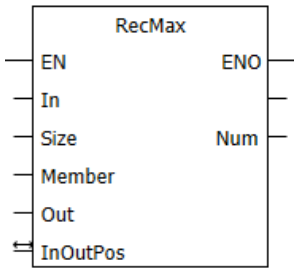
Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.10.2 RecMax	FC	Get the maximum record	OMUtils	Library version 1.0.0.0 and later
18.2.10.3 RecMin	FC	Get the minimum record		Library version 1.0.0.0 and later
18.2.10.4 RecNum	FC	Get number of records		Library version 1.0.0.0 and later
18.2.10.5 Range Search	FC	Get records within specified range		Library version 1.0.0.0 and later
18.2.10.6 RecSearch	FC	Get specified record		Library version 1.0.0.0 and later
18.2.10.7 RecSort	FB	Sort records		Library version 1.0.0.0 and later
18.2.10.8 StackDel	FC	Delete a value from a specified position in a stack		Library version 1.0.0.0 and later
18.2.10.9 StackFIFO	FC	Retrieve the bottom value from a stack		Library version 1.0.0.0 and later
18.2.10.10 StackIns	FC	Insert into stack		Library version 1.0.0.0 and later
18.2.10.11 StackLIFO	FC	Retrieve the top value from a stack		Library version 1.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.10.12 StackPush	FC	Push a value into a stack		Library version 1.0.0.0 and later
18.2.10.13 TableCmp	FC	Table comparison		Library version 1.0.0.0 and later
18.2.10.14 ZoneCmp	FC	Zone comparison		Library version 1.0.0.0 and later
18.2.10.15 RAMP_INT	FB	Ramp function (integer type)	Util	Library version 1.0.0.0 and later
18.2.10.16 RAMP_REAL	FB	Ramp function (real number type)		Library version 1.0.0.0 and later
18.2.10.17 CHARCURVE	FB	Characteristic curve function		Library version 1.0.0.0 and later
18.2.10.18 BZAND_TAB	FC	Dead zone control		Library version 1.0.0.0 and later
18.2.10.19 RAMP_TAB	FB	Ramp instruction		Library version 1.0.0.0 and later
18.2.10.20 ZONE_TAB	FC	Zone control		Library version 1.0.0.0 and later

18.2.10.2 RecMax

Gets the maximum value of a specified member from a structure array.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
RecMax	Get the maximum record	FC		:= RecMax(In := , Size := , Member := , Out := , InOutPos := , Num =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In[ ] array	Structure array to search	Follows data type	-	Array of structures to search

Input/Output	Variable	Name	Valid Range	Default Value	Description
	Size	Number of elements to search	Follows data type	1	Number of array elements to search
	Member	Member to search	Follows data type	-	Member of In[ ] structure array to search
	Out	Search result	Follows data type	-	Search result
Input/Output	InOutPos[ ] array	Index array of search results	Follows data type	-	Array storing the indices of the searched elements
Output	Num	Number of searched elements	Follows data type	-	Number of searched elements

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	Specify an array of structures.																			
Size	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Member	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Data with the same data type as the search member of In[ ]																			
Out	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
InOutPos[ ] array	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction searches the Size elements of the structure array In[ ] (from In[0] to In[Size-1]) to get the value of the specified structure member Member. The member to search from any element of In[ ] is

passed to the Member parameter. The indices of the searched elements are assigned to InOutPos[0] and the number of searched elements is assigned to Num. If there are more than two searched elements, the index of the lowest-order searched element is stored in InOutPos[0].

- Input parameters passed to In[ ] shall be specified along with the element index, such as array[3]. The relationships between values with data types that are not integers or real numbers are determined as follows.

Data Type	Relationship
TIME	The numerically larger value is considered to be larger.
DATA, TOD, DT	Later date or TOD are considered to be larger.
STRING	Character encoding is used for size comparison. The comparison steps are as follows: 1. Compare the character encodings of the first character of each string. If the encodings are different, the size of string is determined by the corresponding encoding values. 2. If the encodings are the same, compare the encodings of the second, third, and subsequent characters until a difference is found. 3. If the lengths of the strings are different, NULL characters (16#00) are added to the shorter string to complete the comparison.

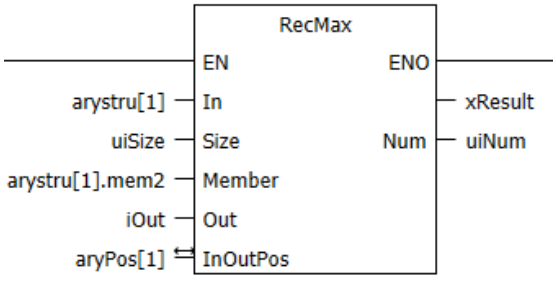
RecMax retrieves the maximum value. The maximum value of the member to search is assigned to search result Out.

- **Program example**

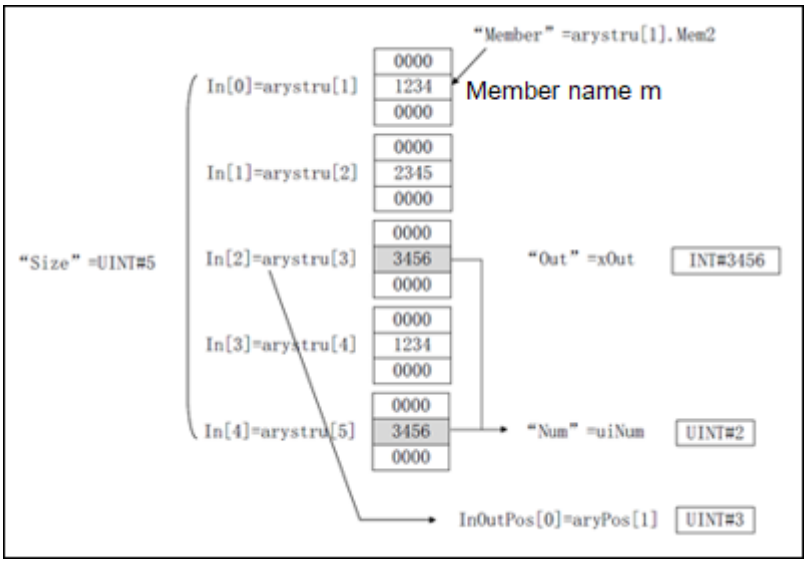
  The following example sets Size to UINT#5.

Item	LD	ST
Define variables		
Program		

```
VAR
    xResult: BOOL;
    arystru: ARRAY [1..5] OF STRUCT_TEST := [(mem2 := 1234), (mem2 := 2345), (mem2 := 3456), (mem2 := 1234), (mem2 := 3456)];
    uiSize: UINT := 5;
    iOut: INT;
    aryPos: ARRAY [1..5] OF UINT;
    uiNum: UINT;
END_VAR
```




```
xResult := RecMax(In := arystru[1],
Size := uiSize,
Member := arystru[1].mem2,
Out := iOut,
InOutPos := aryPos[1],
Num => uiNum);
```

<table>
<tr><th>Item</th><th colspan="2">LD</th><th>ST</th></tr>
<tr><td>Operation result</td><td colspan="3">
<table>
<tr><th></th><th></th><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td>1</td><td></td><td>xResult</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>2</td><td></td><td>arystru</td><td>ARRAY [1..5] OF STRUCT_TEST</td><td></td><td></td></tr>
<tr><td>3</td><td></td><td>arystru[1]</td><td>STRUCT_TEST</td><td></td><td></td></tr>
<tr><td>4</td><td></td><td>arystru[2]</td><td>STRUCT_TEST</td><td></td><td></td></tr>
<tr><td>5</td><td></td><td>arystru[3]</td><td>STRUCT_TEST</td><td></td><td></td></tr>
<tr><td>6</td><td></td><td>arystru[4]</td><td>STRUCT_TEST</td><td></td><td></td></tr>
<tr><td>7</td><td></td><td>arystru[5]</td><td>STRUCT_TEST</td><td></td><td></td></tr>
<tr><td>8</td><td></td><td>uiSize</td><td>UINT</td><td></td><td>5</td></tr>
<tr><td>9</td><td></td><td>iOut</td><td>INT</td><td></td><td>3456</td></tr>
<tr><td>10</td><td></td><td>aryPos</td><td>ARRAY [1..5] OF UINT</td><td></td><td></td></tr>
<tr><td>11</td><td></td><td>aryPos[1]</td><td>UINT</td><td></td><td>3</td></tr>
<tr><td>12</td><td></td><td>aryPos[2]</td><td>UINT</td><td></td><td>0</td></tr>
<tr><td>13</td><td></td><td>aryPos[3]</td><td>UINT</td><td></td><td>0</td></tr>
<tr><td>14</td><td></td><td>aryPos[4]</td><td>UINT</td><td></td><td>0</td></tr>
<tr><td>15</td><td></td><td>aryPos[5]</td><td>UINT</td><td></td><td>0</td></tr>
<tr><td>16</td><td></td><td>uiNum</td><td>UINT</td><td></td><td>2</td></tr>
</table>
</td></tr>
<tr><td>Principle analysis</td><td colspan="3">
</td></tr>
</table>

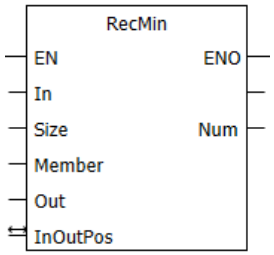
- **Precautions**
  - In[ ] can be a higher-level structure member, for example: In[0] = str0.str1[0].
  - The data types of Member and Out must be consistent; otherwise, a compilation fault will occur.
  - When Member is a real number, different values may introduce errors and may not produce the expected results.
  - If an element from an array is passed to In[ ], all elements from that element onward will be processed.
  - In OMUtils library versions 1.2.0.1 and earlier, when Size is 0, Out and Num are 0, InOutPos[ ] remains unchanged, and the return value is TRUE.
    In OMUtils library versions 1.3.0.0 and later, when Size is 0, Out and Num are 0, InOutPos[ ] remains unchanged, and the return value is FALSE.
  - The dimension of In[ ] must be less than or equal to 3; otherwise, the function returns FALSE, Out and InOutPos[ ] remain unchanged, and Num is 0.
  - InOutPos[ ] must be a one-dimensional array; otherwise, a compilation fault will occur.
  - If Size exceeds the array region of In[ ], the function returns FALSE, Out and InOutPos[ ] remain unchanged, and Num is 0.
  - If Member is not a member of the element specified in In[ ], the function returns FALSE, Out and InOutPos[ ] remain unchanged, and Num is 0.

- If Member is of STRING type and is not NULL-terminated, the result may be abnormal.
- In[ ] can also be an array with two or more dimensions. For a 2D array, the first-dimension index of the matching element is stored in InOutPos[0], and the second-dimension index is stored in InOutPos[1].
- For a 3D array, the first-dimension index is stored in InOutPos[0], the second in InOutPos[1], and the third in InOutPos[2].
- When In[ ] is a multi-dimensional array, the number of elements in InOutPos[ ] must not be less than the dimension of In[ ]; otherwise, the function returns FALSE, Out and InOutPos[ ] remain unchanged, and Num is 0.

18.2.10.3 RecMin

Gets the minimum value of a specified member from a structure array.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
RecMin	Get the minimum record	FC		:= RecMin(In := , Size := , Member := , Out := , InOutPos := , Num =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In[ ] array	Structure array to search	Follows data type	-	Array of structures to search
	Size	Number of elements to search	Follows data type	1	Number of array elements to search
	Member	Member to search	Follows data type	-	Member of In[ ] structure array to search
	Out	Search result	Follows data type	-	Search result
Input/Output	InOutPos[ ] array	Index array of search results	Follows data type	-	Array storing the indices of the searched elements

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Num	Number of searched elements	Follows data type	-	Number of searched elements

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	Specify an array of structures.																			
Size	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Member	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Data with the same data type as the search member of In[ ]																			
Out	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
InOutPos[ ] array	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction searches the Size elements of the structure array In[ ] (from In[0] to In[Size-1]) to get the value of the specified structure member Member. The member to search from any element of In[ ] is passed to the Member parameter. The indices of the searched elements are assigned to InOutPos[0] and the number of searched elements is assigned to Num. If there are more than two searched elements, the index of the lowest-order searched element is stored in InOutPos[0].
  - Input parameters passed to In[ ] shall be specified along with the element index, such as array[3]. The relationships between values with data types that are not integers or real numbers are determined as follows.

Data Type	Relationship
TIME	The numerically smaller value is considered to be smaller.
DATA, TOD, DT	Earlier dates or TOD are considered to be larger.

Data Type	Relationship
STRING	Character encoding is used for size comparison. Then there are the following possibilities: 1. Compare the character encodings of the first character of each string. If the encodings are different, the size of string is determined by the corresponding encoding values. 2. If the encodings are the same, compare the encodings of the second, third, and subsequent characters until a difference is found. 3. If the lengths of the strings are different, NULL characters (16#00) are added to the shorter string to complete the comparison.

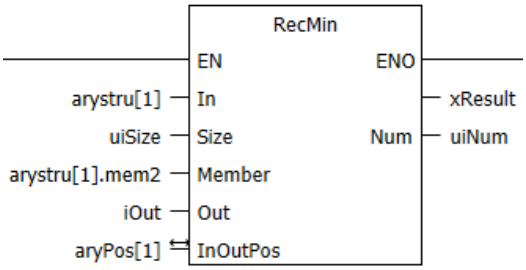
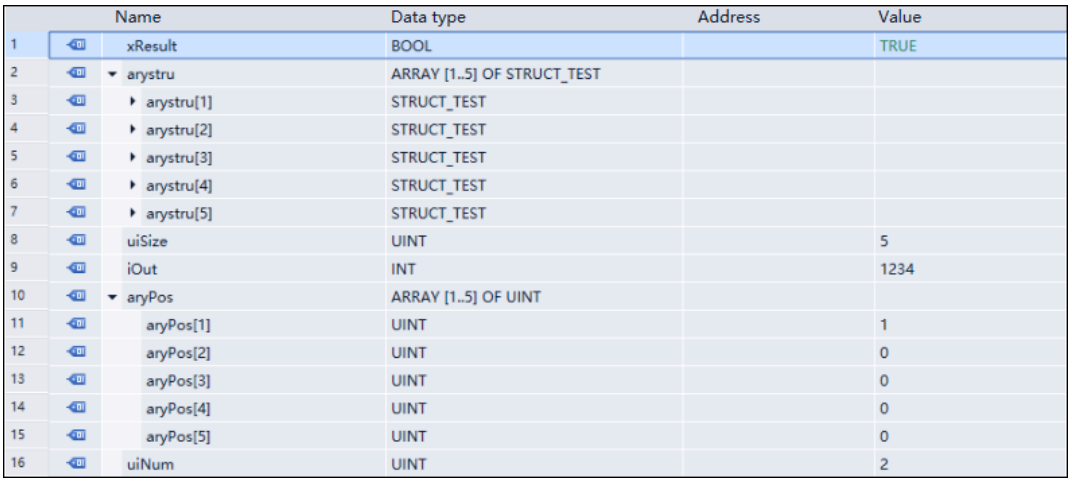
RecMin retrieves the minimum value. The minimum value of the member to search is assigned to search result Out.

- **Program example**

  The following example sets Size to UINT#5.

Item	LD	ST

Define variables	VAR
	xResult: BOOL;
	arystru: ARRAY [1..5] OF STRUCT_TEST := [(mem2 := 1234), (mem2 := 2345), (mem2 := 3456), (mem2 := 1234), (mem2 := 3456)];
	uiSize: UINT := 5;
	iOut: INT;
	aryPos: ARRAY [1..5] OF UINT;
	uiNum: UINT;
	END_VAR	

Program		xResult := RecMin(In := arystru[1], Size := uiSize, Member := arystru[1].mem2, Out := iOut, InOutPos := aryPos[1], Num => uiNum);
Operation result		

	Name	Data type	Address	Value
1	xResult	BOOL		TRUE
2	arystru	ARRAY [1..5] OF STRUCT_TEST		
3	arystru[1]	STRUCT_TEST		
4	arystru[2]	STRUCT_TEST		
5	arystru[3]	STRUCT_TEST		
6	arystru[4]	STRUCT_TEST		
7	arystru[5]	STRUCT_TEST		
8	uiSize	UINT		5
9	iOut	INT		1234
10	aryPos	ARRAY [1..5] OF UINT		
11	aryPos[1]	UINT		1
12	aryPos[2]	UINT		0
13	aryPos[3]	UINT		0
14	aryPos[4]	UINT		0
15	aryPos[5]	UINT		0
16	uiNum	UINT		2

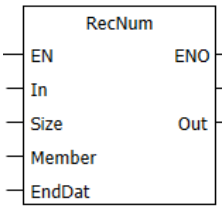
- **Precautions**
  - In[ ] can be a higher-level structure member, for example: In[0] = str0.str1[0].
  - The data types of Member and Out must be consistent; otherwise, a compilation fault will occur.

- When Member is a real number, different values may introduce errors and may not produce the expected results.
- If an element from an array is passed to In[ ], all elements from that element onward will be processed.
- In OMUtils library versions 1.2.0.1 and earlier, when Size is 0, Out and Num are 0, InOutPos[ ] remains unchanged, and the return value is TRUE.
  In OMUtils library versions 1.3.0.0 and later, when Size is 0, Out and Num are 0, InOutPos[ ] remains unchanged, and the return value is FALSE.
- The dimension of In[ ] must be less than or equal to 3; otherwise, the function returns FALSE, Out and InOutPos[ ] remain unchanged, and Num is 0.
- InOutPos[ ] must be a one-dimensional array; otherwise, a compilation fault will occur.
- If Size exceeds the array region of In[ ], the function returns FALSE, Out and InOutPos[ ] remain unchanged, and Num is 0.
- If Member is not a member of the element specified in In[ ], the function returns FALSE, Out and InOutPos[ ] remain unchanged, and Num is 0.
- If Member is of STRING type and is not NULL-terminated, the result may be abnormal.
- In[ ] can also be an array with two or more dimensions. For a 2D array, the first-dimension index of the matching element is stored in InOutPos[0], and the second-dimension index is stored in InOutPos[1].
- For a 3D array, the first-dimension index is stored in InOutPos[0], the second in InOutPos[1], and the third in InOutPos[2].
- When In[ ] is a multi-dimensional array, the number of elements in InOutPos[ ] must not be less than the dimension of In[ ]; otherwise, the function returns FALSE, Out and InOutPos[ ] remain unchanged, and Num is 0.

18.2.10.4 RecNum

Gets the number records in an array of structures to the end data.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
RecNum	Get number of records	FC		:= RecNum(In := , Size := , Member := , EndDat := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In[ ] array	Structure array to search	Follows data type	-	Array of structures to search

Input/Output	Variable	Name	Valid Range	Default Value	Description
	Size	Number of elements to search	Follows data type	1	Number of array elements to search
	Member	Member to search	Follows data type	-	Member of In[ ] structure array to search
	EndDat	End data	Follows data type	-	End data
Output	Out	Number of records	Follows data type	-	Number of records

Variable types

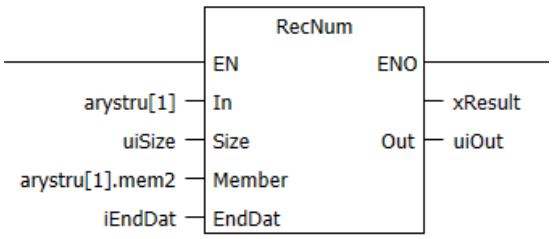
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	Specify an array of structures.																			
Size	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Member	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Data with the same data type as the search member of In[ ]																			
End Dat	Array with the same data type as the elements of Member																			
Out	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction searches an array of structures In[ ] for Size elements ranging from In[0] to In[Size–1] where the value of Member matches the end data EndDat. Then the number of elements (records) matching EndDat is assigned to Out.
  - The member to search from any element of In[ ] is passed to the Member parameter.
  - Input parameters passed to In[ ] shall be specified along with the element index, such as array[3].
- **Program example**

  The following example sets EndDat to INT#9999.

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

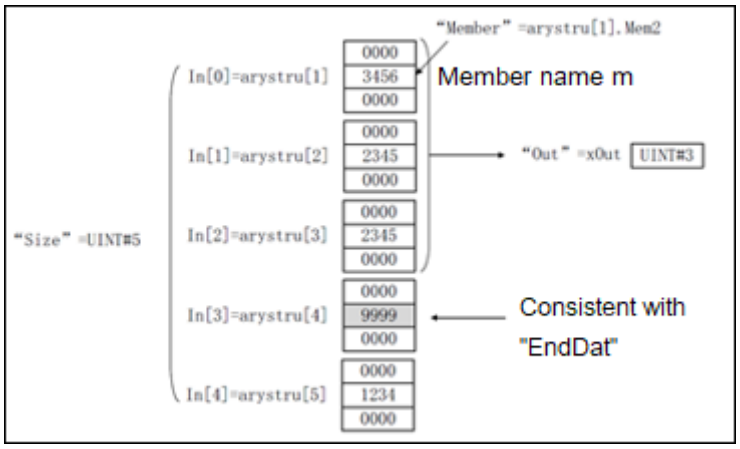
```
VAR
    xResult: BOOL;
    arystru: ARRAY [1..5] OF STRUCT_TEST := [(mem2 := 3456), 2((mem2 := 2345)), (mem2 := 9999), (mem2 := 1234)];
    uiSize: UINT := 5;
    iEndDat: INT := 9999;
    uiOut: UINT;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>
</td><td>

```
xResult := RecNum(In := arystru[1],
                  Size := uiSize,
                  Member := arystru[1].mem2,
                  EndDat := iEndDat,
                  Out => uiOut);
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

	Name	Data type	Address	Value
1	xResult	BOOL		TRUE
2	arystru	ARRAY [1..5] OF STRUCT_TEST		
3	arystru[1]	STRUCT_TEST		
4	arystru[2]	STRUCT_TEST		
5	arystru[3]	STRUCT_TEST		
6	arystru[4]	STRUCT_TEST		
7	arystru[5]	STRUCT_TEST		
8	uiSize	UINT		5
9	iEndDat	INT		9999
10	uiNum	UINT		3

</td></tr>
<tr><td>Principle analysis</td><td colspan="2">
</td></tr>
</table>

- **Precautions**
  - In[ ] must be an array of structure members, otherwise, a fault occurs during compilation.
  - The data types of Member and EndDat must be consistent; otherwise, an exception will occur during compilation.
  - If no element in Size members of In[ ] matches EndDat, the value of Out is Size.
  - When Member is a real number, different values may introduce errors and may not produce the expected results.
  - When EndDat is a real number, do not specify a non-numeric value for EndDat.
  - If an element from an array is passed to In[ ], all elements from that element onward will be processed.
  - If In[ ] exceeds 3 dimensions, the function returns FALSE, and Out is initialized to 0.
  - In OMUtils library versions 1.2.0.1 and earlier, when the value of Size is 0, Out is initialized to 0, and the function returns TRUE.

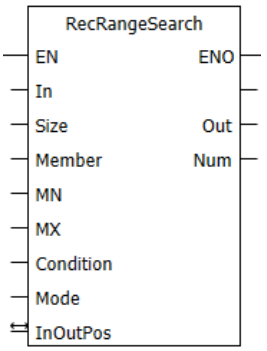
In OMUtils library versions 1.3.0.0 and later, when the value of Size is 0, Out is initialized to 0, and the function returns FALSE.

- When the value of Size exceeds the In[ ] array, the return value is FALSE and the value of Out is initialized to 0.
- When the value of Member is not a member of In[ ], the return value is FALSE and the value of Out is initialized to 0.
- If Member is of STRING type and is not NULL-terminated, the result may be abnormal.

18.2.10.5 RecRangeSearch

Searches an array of structures for elements that match the search condition range with the specified method.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
RecRange Search	Get records within specified range	FC		:= RecRangeSearch(In := , Size := , Member := , MN := , MX := , Condition := , Mode := , InOutPos := , Out => , Num =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In[ ] array	Array to search	Follows data type	-	Array of structures to search
	Size	Number of elements to search	Follows data type	0	Number of array elements to search
	Member	Member to search	Follows data type	-	Member of In[ ] structure array to search
	MN	Search condition lower limit	Follows data type	-	Search condition lower limit
	MX	Search condition upper limit	Follows data type	-	Search condition upper limit

Input/Output	Variable	Name	Valid Range	Default Value	Description
	Condition	Search condition	_eSEARCH_CONDITION	1(_EQ_BOTH)	Search condition
	Mode	Search method	_eSEARCH_MODE	1(_LINEAR)	Search method
Input/Output	InOutPos[ ] array	Element indices of matching elements	Follows data type	-	Element indices of matching elements
Output	Out	Search result	Follows data type	-	Search result
	Num	Number of matching elements	Follows data type	-	Number of matching elements

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	Specify an array of structures.																			
Size	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Member	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Data with the same data type as the search member of In[ ]																			
MN	Array with the same data type as the elements of Member																			
MX	Array with the same data type as the elements of Member																			
Condition	Refer to "Function description" for the enumerators of _eSEARCH_CONDITION.																			
Mode	Refer to "Function description" for enumerators of _eSEARCH_MODE.																			

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
InOutPos[ ] array	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction searches an array of structures In[ ] for Size elements ranging from In[0] to In[Size–1] where the value of Member matches the search conditions.
  - Condition specifies the search condition. Mode specifies the search method. The member to search from any element of In[ ] is passed to the Member parameter. If any elements matching the conditions are found, the value of search result Out changes to TRUE. The indices of the matching elements are assigned to InOutPos[0] and the number of matching elements is assigned to Num. If there are more than two matching elements, the index of the lowest-order matching element is stored in InOutPos[0].
  - If there are no matching elements, the value of Out is FALSE and InOutPos[0] and Num are 0. Input parameters passed to In[ ] shall be specified along with the element index, such as array[3].
  - The data type of the search condition Condition is enumerated type _eSEARCH_CONDITION. The meanings of the enumerators are as follows:

Enumerator	Definition
1 (EQ_BOTH)	"MN" ≤ "Member" ≤ "MX"
2 (_EQ_MIN)	"MN" ≤ "Member" < "MX"
3 (_EQ_MAX)	"MN" < "Member" ≤ "MX"
4 (_NE_BOTH)	"MN" < "Member" < "MX"

  - The data type of search method Mode is enumerated type _eSEARCH_MODE. The meanings of the enumerators are as follows:

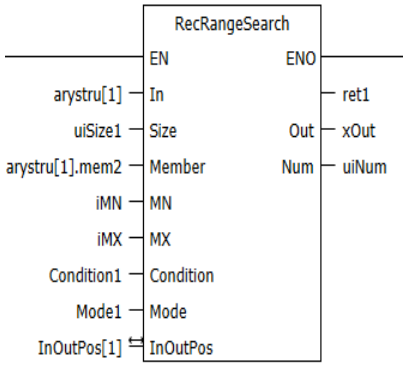
Enumerator	Definition
1 (_LINEAR)	Linear search
2 (_BIN_ASC)	Ascending binary search
3 (_BIN_DESC)	Descending binary search

- **Program example**

In the following example, "Size" is UINT#5, "MN" is INT#1000, "MX" is INT#2000, "Condition" is _EQ_BOTH, and "Mode" is _LINEAR.

<table>
<tr><th>Item</th><th colspan="2">LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="3">

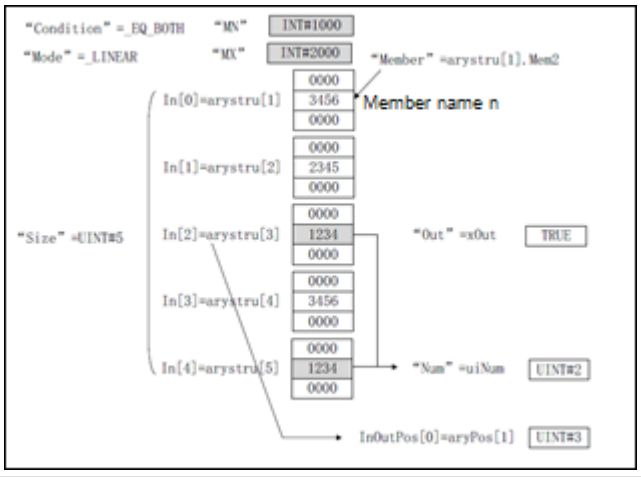
```
VAR
    ret1: BOOL;
    arystru: ARRAY [1..5] OF STRUCT_TEST := [(mem2 := 3456), (mem2 := 2345), (mem2 := 1234), (mem2 := 3456), (mem2 := 1234)];
    uiSize1: UINT := 5;
    iMN: INT := 1000;
    iMX: INT := 2000;
    Condition1: _eSEARCH_CONDITION := _ESEARCH_CONDITION._EQ_BOTH;
    Mode1: _eSEARCH_MODE := _ESEARCH_MODE._LINEAR;
    InOutPos: ARRAY [1..3] OF UINT;
    xOut: BOOL;
    uiNum: UINT;
END_VAR
```

</td></tr>
<tr><td>Program</td><td colspan="2">
</td><td>

```
ret1 := RecRangeSearch(In := arystru[1],
Size := uiSize1,
Member := arystru[1].mem2,
MN := iMN,
MX := iMX,
Condition := Condition1,
Mode := Mode1,
InOutPos := InOutPos[1],
Out => xOut,
Num => uiNum);
```

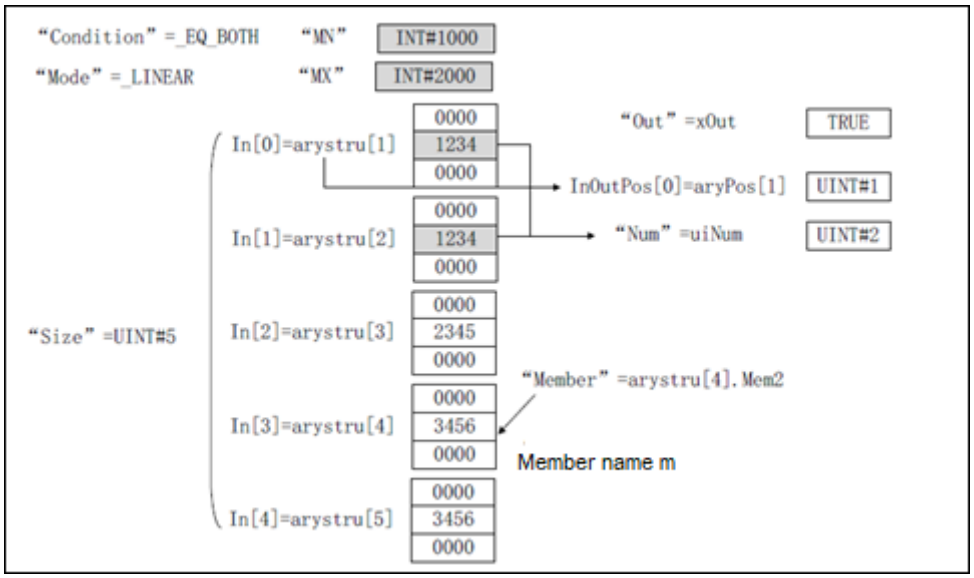
</td></tr>
<tr><td>Operation result</td><td colspan="3">

Name	Data type	Address	Value
ret1	BOOL		TRUE
arystru	ARRAY [1..5] OF STRUCT_TEST		
arystru[1]	STRUCT_TEST		
arystru[2]	STRUCT_TEST		
arystru[3]	STRUCT_TEST		
arystru[4]	STRUCT_TEST		
arystru[5]	STRUCT_TEST		
uiSize1	UINT		5
iMN	INT		1000
iMX	INT		2000
Condition1	_ESEARCH_CONDITION		_EQ_BOTH
Mode1	_ESEARCH_MODE		_LINEAR
InOutPos	ARRAY [1..3] OF UINT		
InOutPos[1]	UINT		3
InOutPos[2]	UINT		0
InOutPos[3]	UINT		0
xOut	BOOL		TRUE
uiNum	UINT		2

</td></tr>
<tr><td>Principle analysis</td><td colspan="3">
</td></tr>
</table>

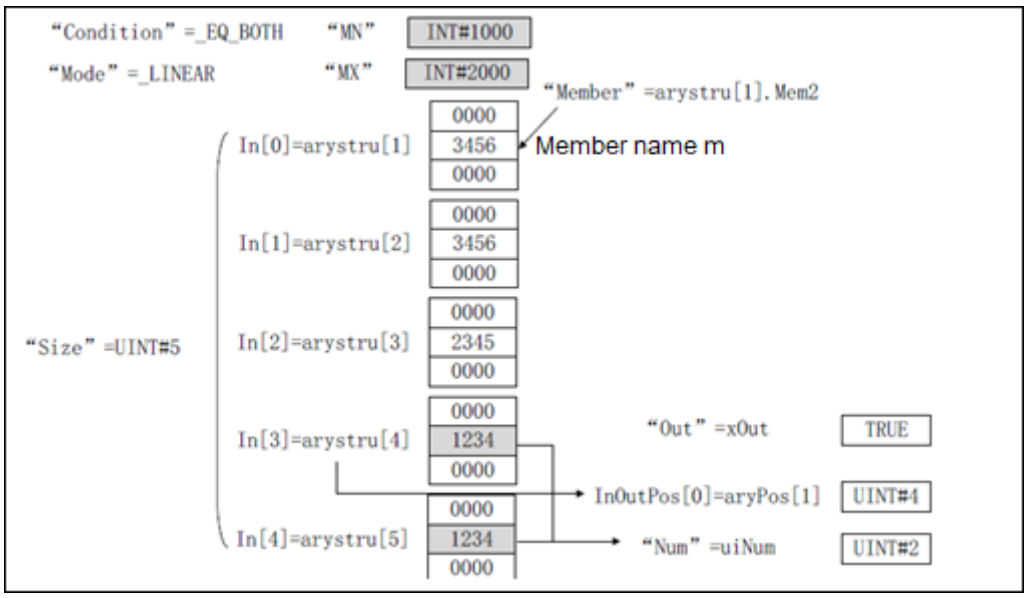
When performing an ascending binary search, the array elements passed to In[ ] must be converted into an ascending array. Then execute this instruction and perform a binary search.

The figure below shows order of the array elements and the processing results for the an ascending binary search based on the above example.



When performing a descending binary search, the array elements passed to In[ ] must be converted into a descending array. Then execute this instruction and perform a binary search.

The figure below shows order of the array elements and the processing results for the a descending binary search based on the above example.



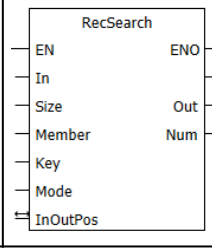
- **Precautions**
  - In[ ] can be a higher-level structure member, for example: In[0] = str0.str1[0].
  - In[ ] must be an array of structure members, otherwise, a fault occurs during compilation.
  - The data types of "Member", "MN", and "MX" shall be the same as that of the members searched in In[ ], otherwise, a fault occurs during compilation.
  - If an element from an array is passed to In[ ], all elements from that element onward will be processed.
  - When Member is a real number, different values may introduce errors and may not produce the expected results.

- When "MN" or "MX" is a real number, do not specify a non-numeric value.
- When the value of Size is 0, Out changes to FALSE. When the value of Num is 0, InOutPos[ ] remains unchanged.
- When the value of Mode is _BIN_ASC or _BIN_DESC and the elements of In[ ] are not in ascending or descending order, the correct result is not obtained. Sort the elements in ascending or descending order before executing this instruction.
- If "MN" > "MX", a fault occurs in the program, the function returns FALSE, "Out" is FALSE, "Num" is 0, and InOutPos[ ] remains unchanged.
- When the value of Condition exceeds the valid range, a fault occurs during compilation.
- When the value of Mode exceeds the valid range, an error occurs during compilation.
- When the value of "Size" exceeds the array range of In[ ] or when the input array dimension is greater than 3, the function returns FALSE, and InOutPos[ ] remains unchanged.
- When Member is not a member of In[ ], the return value is FALSE and InOutPos[ ] remains unchanged.
- InOutPos[ ] must be an one-dimensional array.
- In[ ] can also be an array with two or more dimensions. For a 2D array, the first-dimension index of the matching element is stored in InOutPos[0], and the second-dimension index is stored in InOutPos[1].
- For a 3D array, the first-dimension index is stored in InOutPos[0], the second in InOutPos[1], and the third in InOutPos[2].
- When In[ ] is a multi-dimensional array, the number of elements in InOutPos[ ] must not be less than the dimension of In[ ]; otherwise, the function returns FALSE, Out is FALSE, Num is 0, and InOutPos[ ] remains unchanged.

18.2.10.6 RecSearch

Searches an array of structures for elements that match the search keyword with the specified method.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
RecSearch	Get specified record	FC		:= RecSearch(In := , Size := , Member := , Key := , Mode := , InOutPos := , Out => , Num =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In[ ] array	Array to search	Follows data type	-	Array of structures to search

Input/Output	Variable	Name	Valid Range	Default Value	Description
	Size	Number of elements to search	Follows data type	0	Number of array elements to search
	Member	Member to search	Follows data type	-	Member of In[ ] structure array to search
	Key	Search keyword	Follows data type	-	Search keywords.
	Mode	Search method	_eSEARCH_MODE	1 (_LINEAR)	Search method
Input/Output	InOutPos[ ] array	Element indices of matching elements	Follows data type	-	Element indices of matching elements
Output	Out	Search result	Follows data type	-	TRUE: Matching elements exist. FALSE: No matching elements exist.
	Num	Number of matching elements	Follows data type	-	Number of matching elements

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	Specify an array of structures.																			
Size	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Member	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Data with the same data type as the search member of In[ ]																			
Key	Array with the same data type as the elements of Member																			

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Mode	Refer to "Function description" for enumerators of _eSEARCH_MODE.																			
InOutPos[ ] array	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction searches an array of structures In[ ] for Size elements ranging from In[0] to In[Size–1] where the value of Member matches Key.
  - The member to search from any element of In[ ] is passed to the Member parameter. If any matching elements are found, the value of search result Out changes to TRUE. The indices of the matching elements are assigned to InOutPos[0] and the number of matching elements is assigned to Num. If there are more than two matching elements, the index of the lowest-order matching element is stored in InOutPos[0].
  - If there are no matching elements, the value of Out is FALSE and InOutPos[0] and Num are 0. Input parameters passed to In[ ] shall be specified along with the element index, such as array[3].
  - The data type of search method Mode is enumerated type _eSEARCH_MODE. The meanings of the enumerators are as follows:

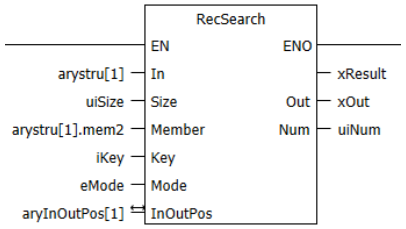
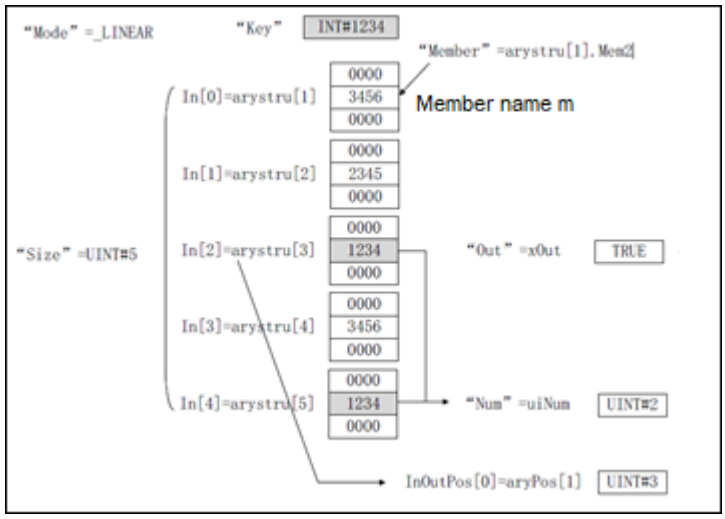
Enumerator	Definition
1 (_LINEAR)	Linear search
2 (_BIN_ASC)	Ascending binary search
3 (_BIN_DESC)	Descending binary search

- **Program example**

  For a linear search, the search is performed from the first element of In[ ]. In the following example, Size is UINT#5, Key is INT#1234, and Mode is _LINEAR.

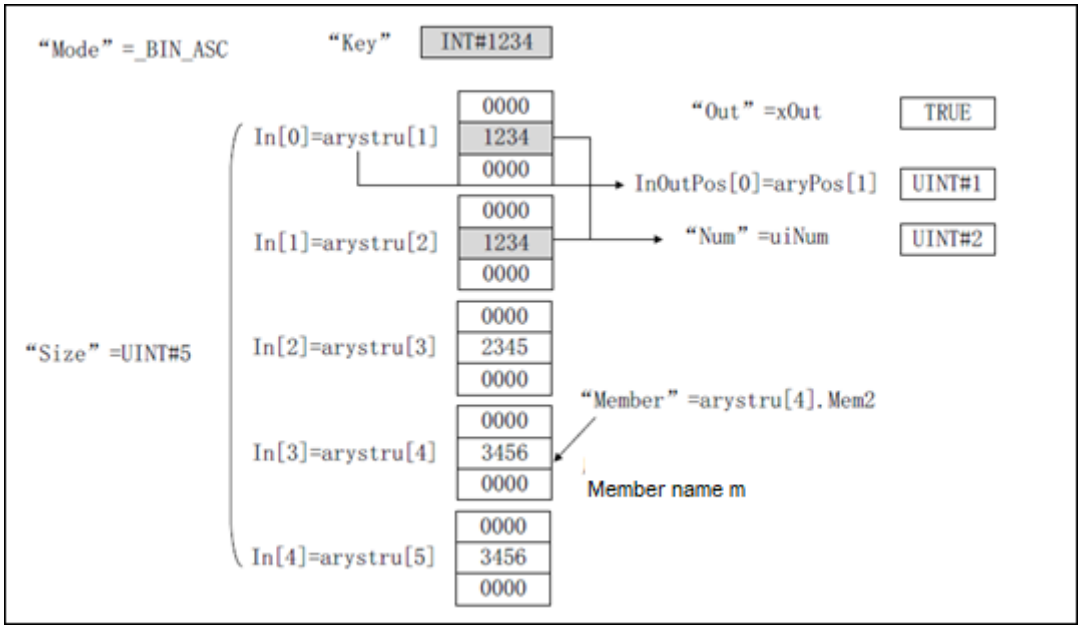
Item	LD	ST
Define variables		

```
VAR
    xResult: BOOL;
    arystru: ARRAY [1..5] OF STRUCT_TEST := [(mem2 := 3456), (mem2 := 2345), (mem2 := 1234), (mem2 := 3456), (mem2 := 1234)];
    uiSize: UINT := 5;
    iKey: INT := 1234;
    eMode: _eSEARCH_MODE := _eSEARCH_MODE._LINEAR;
    aryInOutPos: ARRAY [1..5] OF UINT;
    xOut: BOOL;
    uiNum: UINT;
END_VAR
```

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>
</td><td><pre>
xResult := RecSearch(In := arystru[1],
Size := uiSize,
Member := arystru[1].mem2,
Key := iKey,
Mode := eMode,
InOutPos := aryInOutPos[1],
Out => xOut,
Num => uiNum);
</pre></td></tr>
<tr><td>Operation result</td><td colspan="2">
<table>
<tr><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td>xResult</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>arystru</td><td>ARRAY [1..5] OF STRUCT_TEST</td><td></td><td></td></tr>
<tr><td>arystru[1]</td><td>STRUCT_TEST</td><td></td><td></td></tr>
<tr><td>arystru[2]</td><td>STRUCT_TEST</td><td></td><td></td></tr>
<tr><td>arystru[3]</td><td>STRUCT_TEST</td><td></td><td></td></tr>
<tr><td>arystru[4]</td><td>STRUCT_TEST</td><td></td><td></td></tr>
<tr><td>arystru[5]</td><td>STRUCT_TEST</td><td></td><td></td></tr>
<tr><td>uiSize</td><td>UINT</td><td></td><td>5</td></tr>
<tr><td>iKey</td><td>INT</td><td></td><td>1234</td></tr>
<tr><td>eMode</td><td>_ESEARCH_MODE</td><td></td><td>_LINEAR</td></tr>
<tr><td>aryInOutPos</td><td>ARRAY [1..5] OF UINT</td><td></td><td></td></tr>
<tr><td>aryInOutPos[1]</td><td>UINT</td><td></td><td>3</td></tr>
<tr><td>aryInOutPos[2]</td><td>UINT</td><td></td><td>0</td></tr>
<tr><td>aryInOutPos[3]</td><td>UINT</td><td></td><td>0</td></tr>
<tr><td>aryInOutPos[4]</td><td>UINT</td><td></td><td>0</td></tr>
<tr><td>aryInOutPos[5]</td><td>UINT</td><td></td><td>0</td></tr>
<tr><td>xOut</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>uiNum</td><td>UINT</td><td></td><td>2</td></tr>
</table>
</td></tr>
<tr><td>Cause Analysis</td><td colspan="2">
</td></tr>
</table>

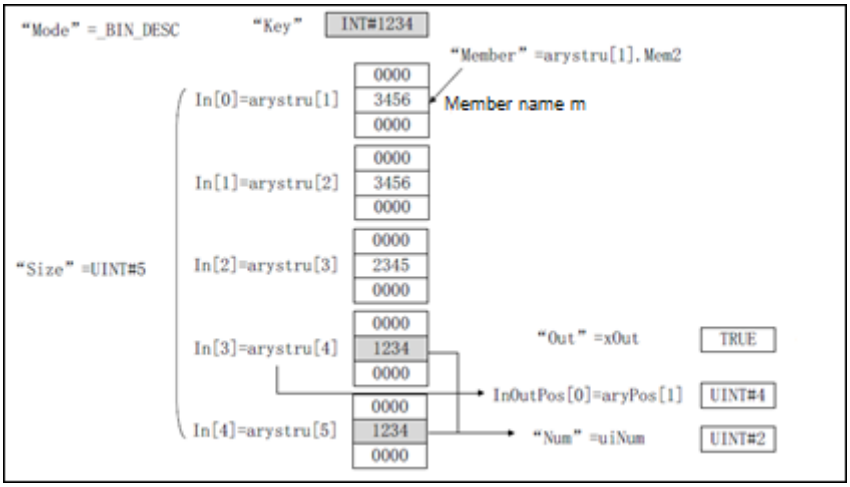
When performing an ascending binary search, the array elements passed to In[ ] must be converted into an ascending array. Then execute this instruction and perform a binary search.

The figure below shows order of the array elements and the processing results for the an ascending binary search based on the above example.



When performing a descending binary search, the array elements passed to In[ ] must be converted into a descending array. Then execute this instruction and perform a binary search.

The figure below shows order of the array elements and the processing results for the a descending binary search based on the above example.



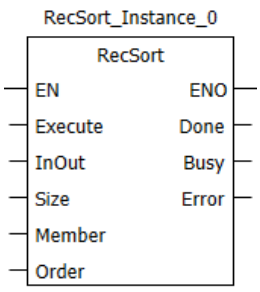
- **Precautions**
  - In[ ] must be an array of structure members, otherwise, a fault occurs during compilation.
  - In[ ] can be a higher-level structure member, for example: In[0] = str0.str1[0].
  - If an element from an array is passed to In[ ], all elements from that element onward will be processed.
  - When Member is a real number, different values may introduce errors and may not produce the expected results.
  - When Key is a real number, do not specify a non-numeric value for Key, otherwise, the program may be faulty.
  - Use the same data type for Key and Member; otherwise, a fault occurs during compilation.
  - When the value of Size is 0, Out is 0 and the return value is TRUE. When the value of Num is 0, InOutPos[ ] remains unchanged.
  - When the value of Mode is _BIN_ASC or _BIN_DESC and the elements of In[ ] are not in ascending or descending order, the correct result is not obtained. Sort the elements in ascending or descending order before executing this instruction.

- When the value of Mode exceeds the valid range, an error occurs during compilation.
- When the value of Size exceeds the In[ ] array, the return value is FALSE and InOutPos[ ] remains unchanged.
- When Member is not a member of In[ ], the return value is FALSE and InOutPos[ ] remains unchanged.
- If Member is of STRING type and is not NULL-terminated, the result may be abnormal.
- InOutPos[ ] must be a one-dimensional array.
- In[ ] can also be an array with two or more dimensions. For a 2D array, the first-dimension index of the matching element is stored in InOutPos[0], and the second-dimension index is stored in InOutPos[1].
- For a 3D array, the first-dimension index is stored in InOutPos[0], the second in InOutPos[1], and the third in InOutPos[2].
- When In[ ] is a multi-dimensional array, the number of elements in InOutPos[ ] must not be less than the dimension of In[ ]; otherwise, the function returns FALSE, Out is FALSE, Num is 0, and InOutPos[ ] remains unchanged.

18.2.10.7 RecSort

Sorts the elements of an array of structures.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
RecSort	Sort records	FB		RecSort_Instance_0(Execute := , InOut := , Size := , Member := , Order := , Done => , Busy => , Error =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Start	Follows data type	FALSE	FALSE: The instruction is not enabled. TRUE: The instruction is enabled.

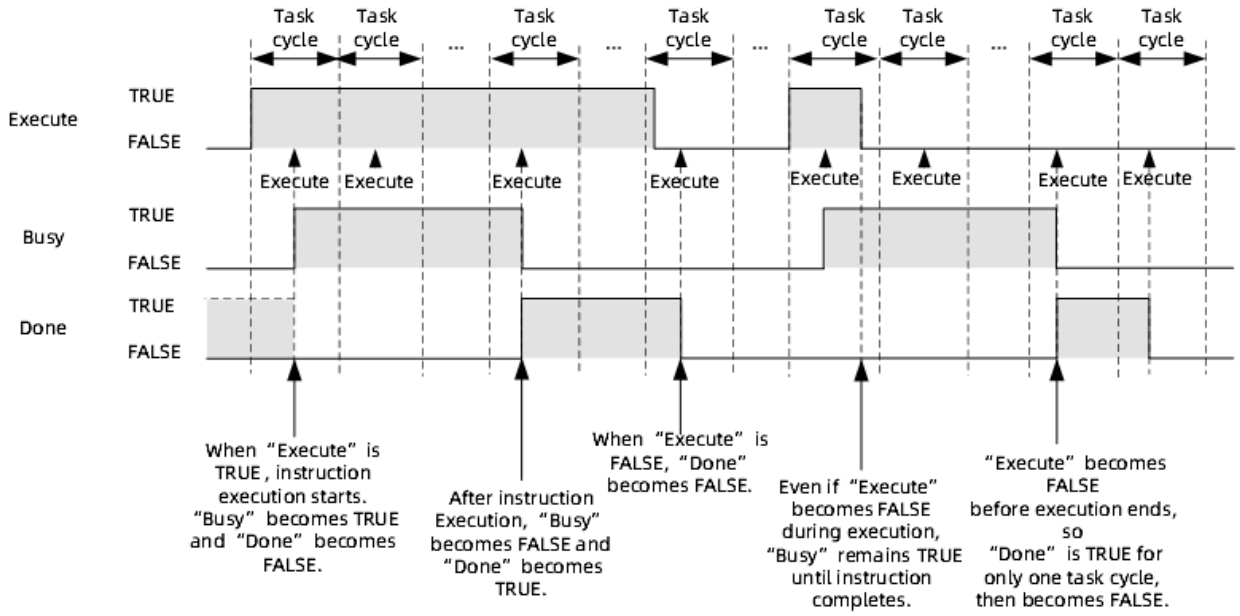
Input/Output	Variable	Name	Valid Range	Default Value	Description
	Size	Number of elements to search	Follows data type	1	Number of array elements to search
	Member	Member to search	Follows data type	-	Member of InOut[ ] structure to sort
	Order	Array order	1(_ASC), 2(_DESC)	1 (_ASC)	1: Ascending order 2: Descending order
	InOut[ ] array	Array to sort	Follows data type	-	Array of structures to sort
Output	Done	Completion flag	Follows data type	FALSE	FALSE: Abnormal termination, execution in progress, or execution condition not met TRUE: Normal termination
	Busy	Operation flag	Follows data type	FALSE	FALSE: Not running TRUE: Running
	Error	Error flag	Follows data type	FALSE	FALSE: No error TRUE: An error exists.

Variable types

	Boolean	Bit String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	
Execute	√	–	–	–	–	–	–

	Boolean	Bit String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	
Size	–	–	–	–	–	–	√
Member	–	–	–	–	–	√	√
	Data with the same data type as the search member of InOut[ ]						
Order	Refer to Function for the enumerators of the enumerated type _eSORT_ORDER.						
InOut[ ] array	Specify an array of structures.						
Done	√	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–
Error	√	–	–	–	–	–	–

- **Function**
  - When Execute is TRUE, Size elements of InOut[ ] are sorted (from InOut[0] to InOut[Size–1]) based on the value of Member for the structure. Order specifies the sort order.
  - The member to search from any element of InOut[ ] is passed to the Member parameter. Input/Output parameters passed to InOut[ ] shall be specified along with the element index, such as array[3].
  - The sequence diagram for startup, execution, and termination is shown in the figure below.



◦ The data for the sort order Order is enumerated type _eSORT_ORDER. The meanings of the enumerators are as shown in the table below.

Enumerator	Definition
1 (_ASC)	Ascending
2 (_DESC)	Descending

◦ The relationships between values with data types that are not integers or real numbers are determined as follows.

Data Type	Relationship
TIME	The numerically larger value is considered to be larger.
DATA, TOD, DT	Later dates or TOD are considered to be larger.
STRING	Character encoding is used for size comparison. The comparison steps are as follows: 1. Compare the character encodings of the first character of each string. If the encodings are different, the size of string is determined by the corresponding encoding values. 2. If the encodings are the same, compare the encodings of the second, third, and subsequent characters until a difference is found. 3. If the lengths of the strings are different, NULL characters (16#00) are added to the shorter string to complete the comparison.

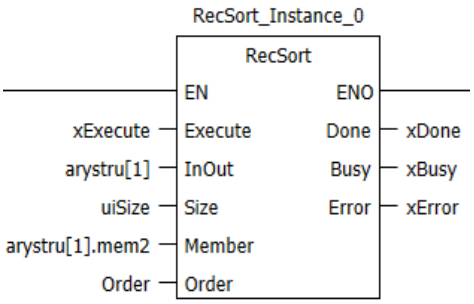
- **Program example**

  The following example sets Size to UINT#5 and Order to _ASC.

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    RecSort_Instance_0: RecSort;
    xExecute: BOOL;
    arystru: ARRAY [1..5] OF STRUCT_TEST := [(mem2 := 3456), (mem2 := 2345), (mem2 := 1234), (mem2 := 3456), (mem2 := 1234)];
    uiSize: UINT := 5;
    Order: _eSORT_ORDER := _ESORT_ORDER._ASC;
    xDone: BOOL;
    xBusy: BOOL;
    xError: BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



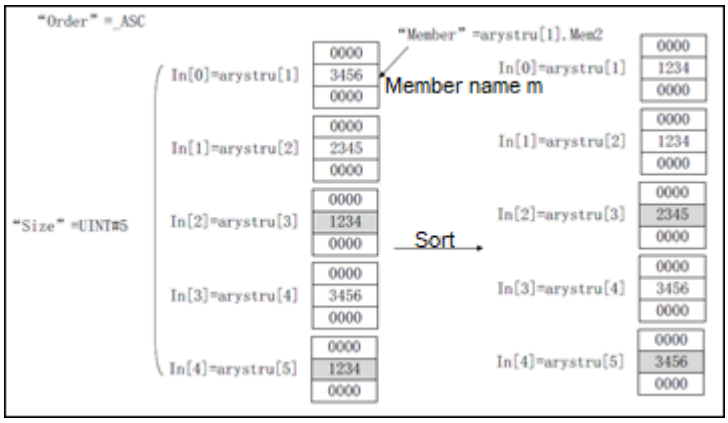
</td><td>

```
RecSort_Instance_0(
Execute := xExecute,
InOut := arystru[1],
Size := uiSize,
Member := arystru[1].mem2,
Order := Order,
Done => xDone,
Busy => xBusy,
Error => xError);
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

Name	Data type	Address	Value
RecSort_Instance_0	RecSort		
xExecute	BOOL		TRUE
arystru	ARRAY [1..5] OF STRUCT_TEST		
arystru[1]	STRUCT_TEST		
mem1	INT		0
mem2	INT		1234
mem3	INT		0
arystru[2]	STRUCT_TEST		
mem1	INT		0
mem2	INT		1234
mem3	INT		0
arystru[3]	STRUCT_TEST		
mem1	INT		0
mem2	INT		2345
mem3	INT		0
arystru[4]	STRUCT_TEST		
mem1	INT		0
mem2	INT		3456
mem3	INT		0
arystru[5]	STRUCT_TEST		
mem1	INT		0
mem2	INT		3456
mem3	INT		0
uiSize	UINT		5
Order	_ESORT_ORDER		_ASC
xDone	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE

</td></tr>
</table>

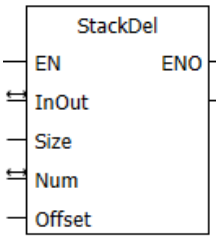
Item	LD	ST
Principle analysis		

- **Precautions**
  - InOut[ ] must be an array of structure members, otherwise, a fault occurs during compilation.
  - Execution of this instruction is continued until processing is completed even if the value of Execute changes to FALSE or the execution time exceeds the task period. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.
  - When Member is a real number, different values may introduce errors and may not produce the expected results.
  - If an element from an array is passed to InOut[ ], all elements from that element onward will be processed.
  - The dimension of InOut[ ] cannot exceed 3; otherwise, InOut[ ] remains unchanged, and the value of Error is TRUE.
  - When the value of Size is 0, the value of Done is TRUE and InOut[ ] remains unchanged.
  - When the value of Order exceeds the valid range, a fault occurs during compilation.
  - When the value of Size exceeds the InOut[ ] array range, Error is TRUE and InOut[ ] remains unchanged.
  - When Member is not a member of InOut[ ], the value of Error is TRUE, and InOut[ ] remains unchanged.
  - If Member is of STRING type and is not NULL-terminated, the result may be abnormal.

18.2.10.8 StackDel

Deletes a value from any position in a stack.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StackDel	Delete a value from a specified position in a stack	FC		:= StackDel(InOut := , Size := , Num := , Offset :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Size	Number of elements in stack	Follows data type	1	Number of array elements in stack
	Offset	Offset	Follows data type	0	Offset of an element to delete from stack
Input/Output	InOut[ ] array	Stack array	Follows data type	-	Array that functions as stack
	Num	Number of elements stored in stack	Follows data type	-	Number of elements stored in stack

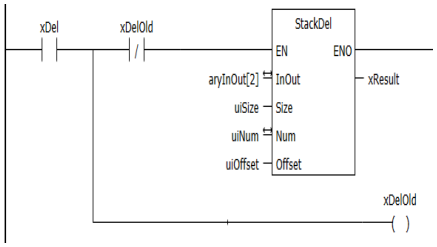
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
InOut[ ] array	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Arrays of enumerations or structures can also be specified.																			
Size	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Offset	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction assumes that the InOut[ ] array containing Num elements is the stack array. It deletes the value at the position specified by InOut[Offset].
  - All higher elements (InOut[Offset + 1] to InOut[Num - 1]) are moved to the next lower element in the stack array.
  - If the value of (Offset + 1) is greater than (Num - 1), InOut[ ] remains unchanged.
  - Size specifies the number of elements to be used as a stack in InOut[ ].
- **Program example**

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    xResult: BOOL;
    aryInOut: ARRAY [1..10] OF UINT := [1, 2, 3, 4, 5, 5(0)];
    uiSize: UINT:= 5;
    uiNum: UINT:= 4;
    uiOffset: UINT:= 2;
END_VAR
```

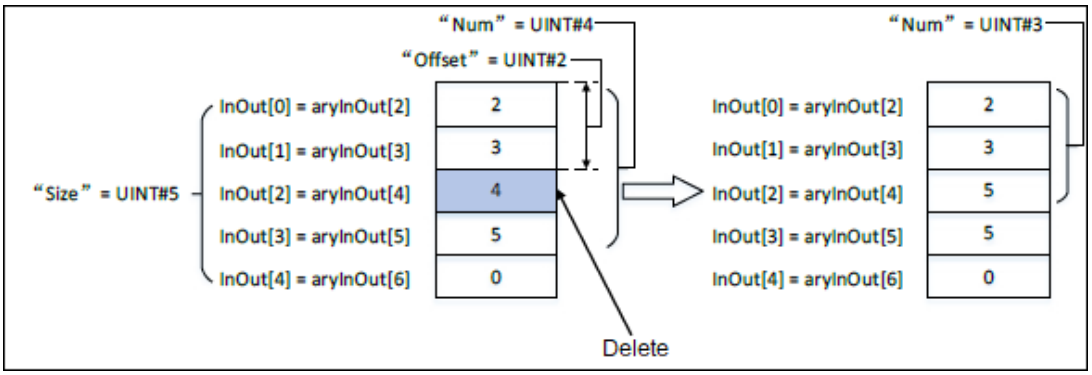
</td></tr>
<tr><td>Program</td><td>
</td><td>

```
IF xDel AND NOT xDelOld THEN
   xResult := StackDel(InOut := aryInOut[2], Size := uiSize, Num := uiNum, Offset := uiOffset);
END_IF

xDelOld:= xDel;
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

Name	Data type	Address	Value
xResult	BOOL		TRUE
aryInOut	ARRAY [1..10] OF INT		
aryInOut[1]	INT		1
aryInOut[2]	INT		2
aryInOut[3]	INT		3
aryInOut[4]	INT		5
aryInOut[5]	INT		5
aryInOut[6]	INT		0
aryInOut[7]	INT		0
aryInOut[8]	INT		0
aryInOut[9]	INT		0
aryInOut[10]	INT		0
uiSize	UINT		5
uiNum	UINT		3
uiOffset	UINT		2
xDel	BOOL		TRUE
xDelOld	BOOL		TRUE

</td></tr>
<tr><td>Principle analysis</td><td colspan="2">
</td></tr>
</table>

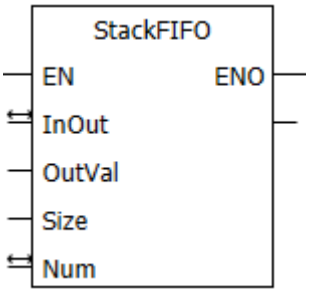
- **Precautions**
  - In OMUtils library versions 1.2.0.1 and earlier, when the value of Size or Num is 0, the function returns TRUE, and InOut[ ] and Num remain unchanged.
    In OMUtils library versions 1.3.0.0 and later, when the value of Size is 0, the function returns TRUE; when the value of Num is 0, the function returns FALSE, and InOut[ ] and Num remain unchanged.
  - If the value of Size exceeds the InOut[ ] array range or when the InOut[ ] dimension exceeds 3, the function returns FALSE, and InOut[ ] and Num remain unchanged.
  - If Size and Num are both non-zero and the condition Size ≥ Num > Offset is not satisfied, the function returns FALSE, and InOut[ ] and Num remain unchanged.

◦ If an element from an array is passed to InOut[ ], all elements from that element onward will be processed.

18.2.10.9 StackFIFO

Retrieves the bottom value from a stack.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StackFIFO	Retrieve the bottom value from a stack	FC		:= StackFIFO(InOut := , OutVal := , Size := , Num :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	OutVal	Output value	Follows data type	-	Value output from stack
	Size	Number of elements in stack	Follows data type	1	Number of array elements in stack
Input/Output	InOut[ ] array	Stack array	Follows data type	-	Array that functions as stack
	Num	Number of elements stored in stack	Follows data type	-	Number of elements stored in stack

Variable types

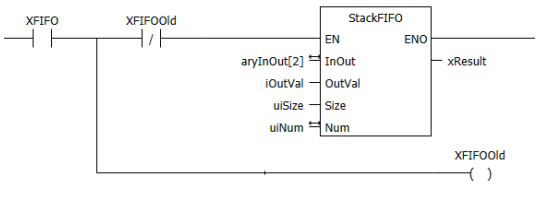
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
InOut[ ] array	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumerator, a structure, or a structure member can also be specified.																			

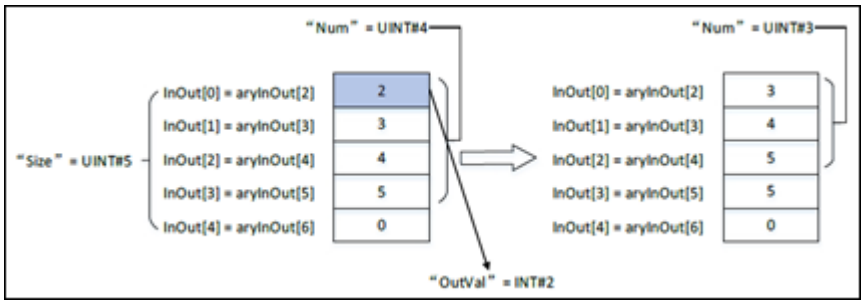
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
OutVal	Array with the same data type as the elements of InOut[ ]																			
Size	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction assumes that the InOut[ ] array containing Num elements is the stack array. It retrieves the value of the least-significant bit InOut[0] from the stack and assigns it to OutVal.
  - Then, all (Num - 1) elements that begin with InOut[1] are shifted to the next lower element in the stack array.
  - Size specifies the number of elements to be used as a stack in InOut[ ].
- **Program example**

Item	LD	ST

Define variables	VAR
	xResult: BOOL;
	aryInOut: ARRAY [1..10] OF INT := [1, 2, 3, 4, 5, 5(0)];
	iOutVal: INT;
	uiSize: UINT:= 5;
	uiNum: UINT:= 4;
	XFIFO: BOOL;
	XFIFOOld: BOOL;
	END_VAR	

Program		IF XFIFO AND NOT(XFIFOOld) THEN xResult := StackFIFO(InOut := aryInOut[2], OutVal := iOutVal, Size := uiSize, Num := uiNum); END_IF XFIFOOld := XFIFO;

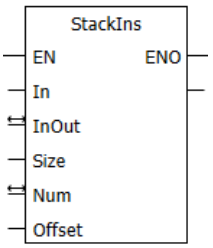
<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Operation result</td><td colspan="2">
<table>
<tr><th></th><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td></td><td>xResult</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td></td><td>aryInOut</td><td>ARRAY [1..10] OF INT</td><td></td><td></td></tr>
<tr><td></td><td>aryInOut[1]</td><td>INT</td><td></td><td>1</td></tr>
<tr><td></td><td>aryInOut[2]</td><td>INT</td><td></td><td>3</td></tr>
<tr><td></td><td>aryInOut[3]</td><td>INT</td><td></td><td>4</td></tr>
<tr><td></td><td>aryInOut[4]</td><td>INT</td><td></td><td>5</td></tr>
<tr><td></td><td>aryInOut[5]</td><td>INT</td><td></td><td>5</td></tr>
<tr><td></td><td>aryInOut[6]</td><td>INT</td><td></td><td>0</td></tr>
<tr><td></td><td>aryInOut[7]</td><td>INT</td><td></td><td>0</td></tr>
<tr><td></td><td>aryInOut[8]</td><td>INT</td><td></td><td>0</td></tr>
<tr><td></td><td>aryInOut[9]</td><td>INT</td><td></td><td>0</td></tr>
<tr><td></td><td>aryInOut[10]</td><td>INT</td><td></td><td>0</td></tr>
<tr><td></td><td>iOutVal</td><td>INT</td><td></td><td>2</td></tr>
<tr><td></td><td>uiSize</td><td>UINT</td><td></td><td>5</td></tr>
<tr><td></td><td>uiNum</td><td>UINT</td><td></td><td>3</td></tr>
<tr><td></td><td>xFIFO</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td></td><td>xFIFOOld</td><td>BOOL</td><td></td><td>TRUE</td></tr>
</table>
</td></tr>
<tr><td>Principle analysis</td><td colspan="2">
</td></tr>
</table>

- **Precautions**
  - If the value of Size exceeds the InOut[ ] array range or when the InOut[ ] dimension exceeds 3, the function returns FALSE, and the values of InOut[ ], Num, and OutVal remain unchanged.
  - When OutVal and InOut[ ] are of STRING type and the number of bytes in InOut[ ] exceeds the value of OutVal, the function returns FALSE, and the values of InOut[ ], Num, and OutVal remain unchanged.
  - The data type of OutVal shall be the same as that of the InOut[ ] elements; otherwise, a compilation fault will occur.
  - In OMUtils library versions 1.2.0.1 and earlier, when the value of Size or Num is 0, the function returns TRUE, and the values of InOut[ ], Num, and OutVal remain unchanged.
    In OMUtils library versions 1.3.0.0 and later, when the value of Size is 0, the function returns TRUE; when the value of Num is 0, the function returns FALSE, the values of InOut[ ], Num, and OutVal remain unchanged.
  - When the value of Size and Num are non-zero and Num is greater than Size, the function returns FALSE, the values of InOut[ ], Num, and OutVal remain unchanged
  - If an element from an array is passed to InOut[ ], all elements from that element onward will be processed.

18.2.10.10 StackIns

Inserts a value at any position in a stack.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StackIns	Insert into stack	FC		:= StackIns(In := , InOut := , Size := , Num := , Offset :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Input	Follows data type	-	Value to input to the stack
	Size	Number of elements in stack	Follows data type	1	Number of array elements in stack
	Offset	Offset	Follows data type	0	Offset position in stack at which to insert In
Input/Output	InOut[ ] array	Stack array	Follows data type	-	Array that functions as stack
	Num	Number of elements stored in stack	Follows data type	-	Number of elements stored in stack

Variable types

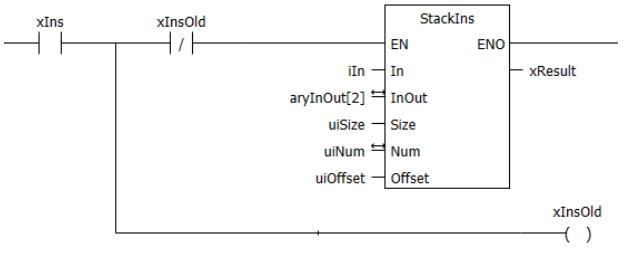
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumerator, a structure, or a structure member can also be specified.																			
InOut[ ] array	Array with the same data type as the elements of InOut[ ]																			

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Size	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Offset	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction assumes that the InOut[ ] array containing Num elements is the stack array. The input value In is inserted at the position specified by InOut[Offset] in the stack.
  - All higher elements (InOut[Offset] to InOut[Num - 1]) are moved to the next higher element in the stack array.
  - Size specifies the number of elements to be used as a stack in InOut[ ].
- **Program example**

Item	LD	ST

Define variables	VAR
	xResult: BOOL;
	iIn: INT := 10;
	aryInOut: ARRAY [1..10] OF INT := [1, 2, 3, 4, 5, 5(0)];
	uiSize: UINT := 5;
	uiNum: UINT := 4;
	uiOffset: UINT := 2;
	xIns: BOOL;
	xInsOld: BOOL;
	END_VAR	

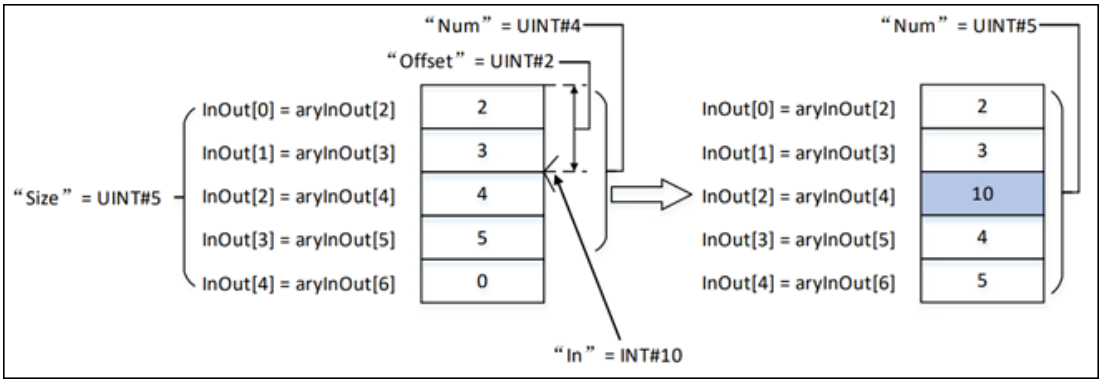
Program		IF xIns AND NOT(xInsOld) THEN
		xResult := StackIns(In := iIn,
		InOut := aryInOut[2],
		Size := uiSize,
		Num := uiNum,
		Offset := uiOffset);
		END_IF
		xInsOld := xInsOld;

Item	LD	ST

Operation result

Name	Data type	Address	Value
xResult	BOOL		TRUE
iIn	INT		10
aryInOut	ARRAY [1..10] OF INT		
aryInOut[1]	INT		1
aryInOut[2]	INT		2
aryInOut[3]	INT		3
aryInOut[4]	INT		10
aryInOut[5]	INT		4
aryInOut[6]	INT		5
aryInOut[7]	INT		0
aryInOut[8]	INT		0
aryInOut[9]	INT		0
aryInOut[10]	INT		0
uiSize	UINT		5
uiNum	UINT		5
uiOffset	UINT		2
xIns	BOOL		TRUE
xInsOld	BOOL		TRUE

Principle analysis

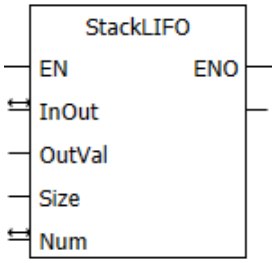


- **Precautions**
  - If the value of Size exceeds the InOut[ ] array range or when the InOut[ ] dimension exceeds 3, the function returns FALSE, and InOut[ ] and Num remain unchanged.
  - If In and InOut[ ] are of STRING type and the number of bytes in In exceeds the value of InOut[ ], the function returns FALSE, and InOut[ ] and Num remain unchanged.
  - A compilation exception occurs when the data types of In and InOut differ.
  - When the value of Size is 0, the function returns TRUE, and InOut[ ] and Num remain unchanged.
  - When the value of Size is not 0 and the condition Size > Num ≥ Offset is not satisfied, the program fails to run, the function returns FALSE, and InOut[ ] and Num remain unchanged.
  - If an element from an array is passed to InOut[ ], all elements from that element onward will be processed.
  - The input parameter transmitted to In must be a variable. If a constant is transmitted, a fault occurs during compilation.
  - When In is an enumeration, no enumerator can be directly transmitted. If an enumerator is directly transmitted, a fault occurs during compilation.
  - When the values of Num and Offset are both 0, the input value is passed to InOut[Offset], and the value of Num is incremented by 1.

18.2.10.11 StackLIFO

Retrieves the top value from a stack.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StackLIFO	Retrieve the top value from a stack	FC		:= StackLIFO(InOut := , OutVal := , Size := , Num :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	OutVal	Output value	Follows data type	-	Value output from stack
	Size	Number of elements in stack	Follows data type	1	Number of array elements in stack
Input/Output	InOut[ ] array	Stack array	Follows data type	-	Array that functions as stack
	Num	Number of elements stored in stack	Follows data type	-	Number of elements stored in stack

Variable types

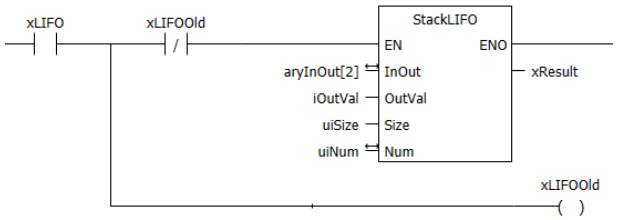
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
InOut[ ] array	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumerator, a structure, or a structure member can also be specified.																			
OutVal	Array with the same data type as the elements of InOut[ ]																			

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Size	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction assumes that the InOut[ ] array containing Num elements is the stack array. It retrieves the value of the most-significant bit InOut[Num - 1] from the stack and assigns it to OutVal.
  - Num is decremented by 1.
  - Size specifies the number of elements to be used as a stack in InOut[ ].
- **Program example**

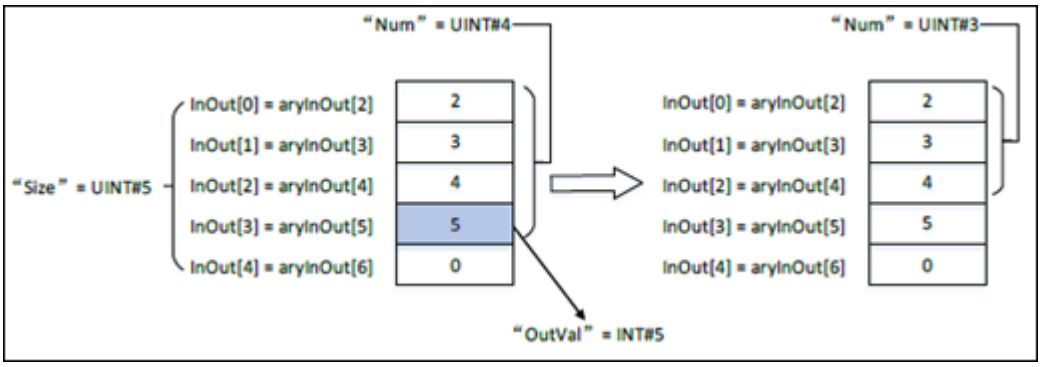
Item	LD	ST
Define variables		
Program		
Operation result		

```
VAR
    xResult: BOOL;
    aryInOut: ARRAY [1..10] OF INT := [1, 2, 3, 4, 5, 5(0)];
    iOutVal: INT;
    uiSize: UINT := 5;
    uiNum: UINT := 4;
    xLIFO: BOOL;
    xLIFOOld: BOOL;
END_VAR
```




```
IF xLIFO AND NOT(xLIFOOld) THEN
    xResult := StackLIFO(InOut := aryInOut[2],
            OutVal := iOutVal,
            Size := uiSize,
            Num := uiNum);
END_IF
xLIFOOld := xLIFO;
```

	Name	Data type	Address	Value
	xResult	BOOL		TRUE
	aryInOut	ARRAY [1..10] OF INT		
	aryInOut[1]	INT		1
	aryInOut[2]	INT		2
	aryInOut[3]	INT		3
	aryInOut[4]	INT		4
	aryInOut[5]	INT		5
	aryInOut[6]	INT		0
	aryInOut[7]	INT		0
	aryInOut[8]	INT		0
	aryInOut[9]	INT		0
	aryInOut[10]	INT		0
	iOutVal	INT		5
	uiSize	UINT		5
	uiNum	UINT		3
	xLIFO	BOOL		TRUE
	xLIFOOld	BOOL		TRUE

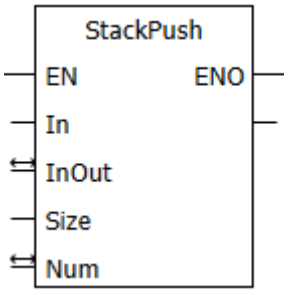
Item	LD	ST
Principle analysis		

- **Precautions**
  - If the value of Size exceeds the InOut[ ] array range or when the InOut[ ] dimension exceeds 3, the function returns FALSE, and the values of InOut[ ], Num, and OutVal remain unchanged.
  - When OutVal and InOut[ ] are of STRING type and the number of bytes in InOut[ ] exceeds the value of OutVal, the function returns FALSE, and the values of InOut[ ], Num, and OutVal remain unchanged.
  - The data type of OutVal shall be the same as that of the InOut[ ] elements; otherwise, a compilation fault will occur.
  - In OMUtils library versions 1.2.0.1 and earlier, when the value of Size or Num is 0, the function returns TRUE, and the values of InOut[ ], Num, and OutVal remain unchanged.
    In OMUtils library versions 1.3.0.0 and later, when the value of Size is 0, the function returns TRUE; when the value of Num is 0, the function returns FALSE, the values of InOut[ ], Num, and OutVal remain unchanged.
  - When the value of Size and Num are non-zero and Num is greater than Size, the function returns FALSE, the values of InOut[ ], Num, and OutVal remain unchanged
  - If an element from an array is passed to InOut[ ], all elements from that element onward will be processed.

18.2.10.12 StackPush

Stores a value in a stack.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StackPush	Push a value into a stack	FC		:= StackPush(In := , InOut := , Size := , Num :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Input	Follows data type	-	Value to input to the stack
	Size	Number of elements in stack	Follows data type	1	Number of array elements in stack
Input/Output	InOut[ ] array	Stack array	Follows data type	-	Array that functions as stack
	Num	Number of elements stored in stack	Follows data type	-	Number of elements stored in stack

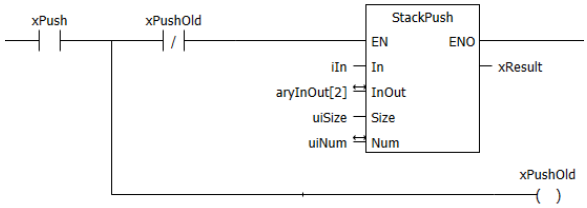
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumerator, a structure, or a structure member can also be specified.																			
InOut[ ] array	Array with the same data type as elements of In																			
Size	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction assumes that the InOut[ ] array containing Num elements is the stack array.
  - The input value In overwrites the next element InOut[Num]. Num is incremented by 1.
  - Size specifies the number of elements to be used as a stack in InOut[ ].
- **Program example**

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

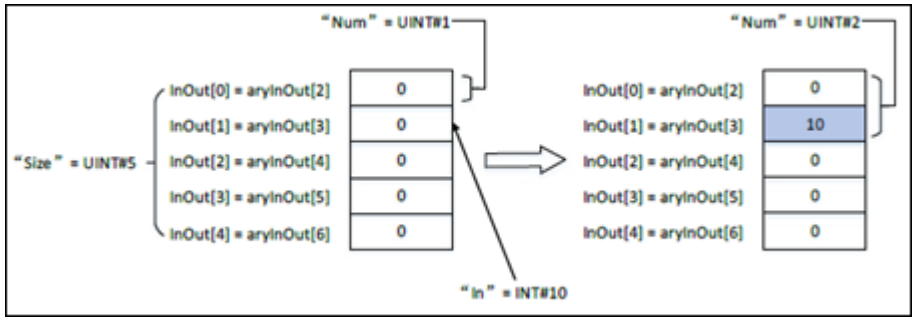
```
VAR
    xResult: BOOL;
    iIn: INT := 10;
    aryInOut: ARRAY [1..10] OF INT;
    uiSize: UINT := 5;
    uiNum: UINT := 1;
    xPush: BOOL;
    xPushOld: BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>
</td><td>

```
IF xPush AND NOT(xPushOld) THEN
    xResult := StackPush(In := iIn,
            InOut := aryInOut[2],
            Size := uiSize,
            Num := uiNum);
END_IF
xPushOld := xPush;
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

	Name	Data type	Address	Value
	xResult	BOOL		TRUE
	iIn	INT		10
	aryInOut	ARRAY [1..10] OF INT		
	aryInOut[1]	INT		0
	aryInOut[2]	INT		0
	aryInOut[3]	INT		10
	aryInOut[4]	INT		0
	aryInOut[5]	INT		0
	aryInOut[6]	INT		0
	aryInOut[7]	INT		0
	aryInOut[8]	INT		0
	aryInOut[9]	INT		0
	aryInOut[10]	INT		0
	uiSize	UINT		5
	uiNum	UINT		2
	xPush	BOOL		TRUE
	xPushOld	BOOL		TRUE

</td></tr>
<tr><td>Principle analysis</td><td colspan="2">
</td></tr>
</table>

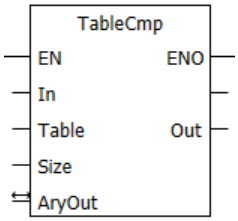
- **Precautions**
  - If the value of Size exceeds the InOut[ ] array range or when the InOut[ ] dimension exceeds 3, the function returns FALSE, and InOut[ ] remains unchanged.
  - If In and InOut[ ] are of STRING type and the number of bytes in In exceeds the value of InOut[ ], the function returns FALSE, and InOut[ ] remains unchanged.
  - A compilation exception occurs when the data types of In and InOut differ.
  - When the value of Size is 0, InOut[ ] and Num remain unchanged, and the function returns TRUE.
  - When the value of Size is 0 and Num is greater than or equal to Size, the program cannot run and the function returns FALSE.

- If the value of Num is 0, the input value is assigned to InOut[Num] and the value of Num is incremented by 1.
- If an element from an array is passed to InOut[ ], all elements from that element onward will be processed.
- The input parameter transmitted to In must be a variable. If a constant is transmitted, a fault occurs during compilation.
- When In is an enumeration, no enumerator can be directly transmitted. If an enumerator is directly transmitted, a fault occurs during compilation.

18.2.10.13 TableCmp

Compares the comparison data with multiple defined ranges in a comparison table.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
TableCmp	Table comparison	FC		:= TableCmp(In := , Table := , Size := , AryOut := , Out =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Comparison data	Follows data type	-	Value to compare
	Table[ ] (two-dimensional array)	Comparison table	Follows data type	-	Two-dimensional array that contains the elements for the defined ranges
	Size	Number of elements to compare	Follows data type	1	Number of elements in Table[ ] to be compared with In
Input/ Output	AryOut[ ] array	Individual comparison result array	Follows data type	-	Comparison results for each element in Table[ ]: TRUE: Consistent. FALSE: Inconsistent.
Output	Out	Comparison result	Follows data type	FALSE	TRUE if all elements are consistent, otherwise FALSE.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Table[ ] (two-dimensional array)	A two-dimensional array with elements that have the same data type as In																			
Size	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
AryOut[ ] array	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

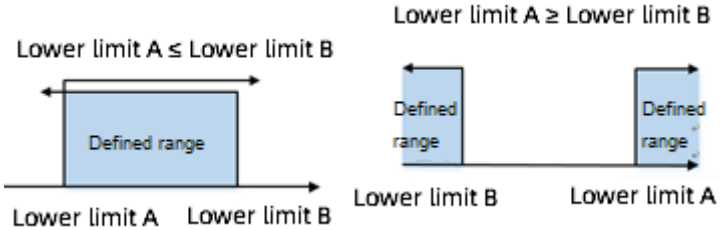
- **Function**

  This instruction compares the comparison data In with defined ranges of Size in the comparison table Table[ ].

  Table[ ] is a two-dimensional array. The first dimension contains the indices of the defined ranges. In the second dimension, element 0 is the minimum value A of the defined range, and element 1 is the maximum value B of the range.

Defined Range	Minimum Value A	Maximum Value B
Range 0	Table[0,0]	Table[0,1]
Range 1	Table[1,0]	Table[1,1]
…	…	…
Interval Size - 1	Table[Size - 1, 0]	Table[Size - 1, 1]

  Use the minimum value A and maximum value B to specify the defined range. Both values are included in the defined range.



The results are stored in the individual comparison result array AryOut[ ]. If In is within the defined range for element i, AryOut[i] is TRUE. If it is not within the range, AryOut[i] is FALSE. If all Size elements of AryOut[ ] are TRUE, the comparison result Out is TRUE; otherwise, it is FALSE.

- **Program example**

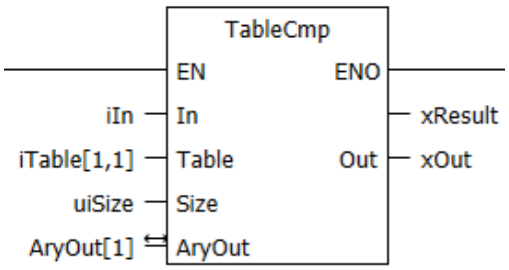
<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    xResult: BOOL;
    iIn: INT := 5;

    iTable: ARRAY [1..3,1..2] OF INT := [0,2,0,10,10,2];

    uiSize: UINT := 3;
    AryOut: ARRAY [1..3] OF BOOL;
    xOut: BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
xResult := TableCmp(In := iIn,
Table := iTable[1,1],
Size := uiSize,
AryOut := AryOut[1],
Out => xOut);
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

Name	Data type	Address	Value
xResult	BOOL		TRUE
iIn	INT		5
iTable	ARRAY [1..3, 1..2] OF INT		
iTable[1, 1]	INT		0
iTable[1, 2]	INT		2
iTable[2, 1]	INT		0
iTable[2, 2]	INT		10
iTable[3, 1]	INT		10
iTable[3, 2]	INT		2
uiSize	UINT		3
AryOut	ARRAY [1..3] OF BOOL		
AryOut[1]	BOOL		FALSE
AryOut[2]	BOOL		TRUE
AryOut[3]	BOOL		FALSE
xOut	BOOL		FALSE

</td></tr>
</table>

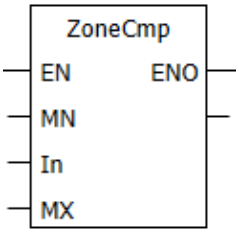
- **Precautions**
  - The data types of In and Table[ ] elements must be consistent; otherwise, a compilation fault will occur.
  - Table[ ] must be a 2D array; otherwise, a compilation fault will occur. Additionally, the size of the second dimension array must be 2; otherwise, the output result may not match expectations.
  - If the size of AryOut[ ] is greater than the value of Size, the comparison results are stored in AryOut[0] to AryOut[Size - 1]. Other array elements remain unchanged.
  - If real numbers that contain non-terminating decimal numbers are compared, a rounding error may occur, causing unexpected processing results.
  - When the value of Size is 0, the function returns TRUE, the value of Out is FALSE, and AryOut[ ] remains unchanged.
  - When the value of Size exceeds the size of AryOut[ ] or exceeds the size of the first dimension of Table[ ], Out is FALSE, and AryOut[ ] remains unchanged.

◦ When the size of the second dimension of Table[ ] is less than 2, the function returns FALSE, Out is FALSE, and AryOut[ ] remains unchanged.

18.2.10.14 ZoneCmp

Determines whether the comparison data is within the specified range.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ZoneCmp	Zone comparison	FC		:= ZoneCmp(MN := , In := , MX :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	MN	Lower limit	Follows data type	-	Lower limit
	In	Comparison data	Follows data type	-	Value to compare
	Mexico	Upper limit	Follows data type	-	Upper limit
Output	Out	Return value	Follows data type	-	Comparison result.

Variable types

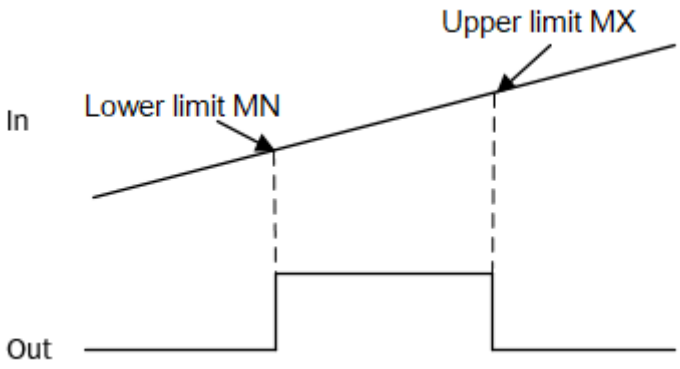
	Bool ean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
MN	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	–
In	Same data type as MN																			

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Mexico	Same data type as MN																			
Out	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction determines whether the comparison data "In" is between maximum value "MX" and minimum value "MN".
  - If "MX" ≥ "In" ≥ "MN", "Out" is TRUE; otherwise, "Out" is FALSE.
  - The relationships between values with data types that are not integers or real numbers are determined as follows.

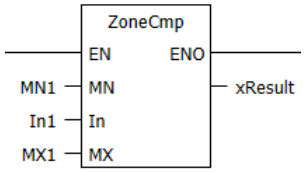
Data Type	Relationship
TIME	The numerically larger value is considered to be larger.
DATA, TOD, DT	Later dates or TOD are considered to be larger.

- **Sequence diagram**



- **Program example**

The following example sets "MN" to INT#10, "In" to INT#20, and "MX" to INT#30.

	LD	ST
Define variables	VAR xResult: BOOL; MN1: INT := 10; In1: INT := 20; MX1: INT := 30; END_VAR	
Program		xResult := ZoneCmp(MN := MN1, In := In1, MX := MX1);
Operation result		

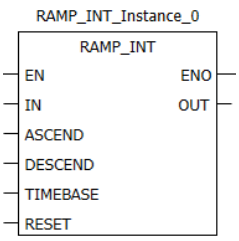
Name	Data type	Address	Value
xResult	BOOL		TRUE
MN1	INT		10
In1	INT		20
MX1	INT		30

- **Precautions**
  - When the data types of In, "MX", and "MN" are different, a fault occurs during compilation.
  - If "In", "MX", and "MN" are real numbers that contain non-terminating decimal numbers, a rounding error may occur, causing unexpected processing results.
  - Two values that are positive infinity or negative infinity are considered equivalent.
  - When the value of In is non-numeric, the value of Out is FALSE.
  - When the value of "MN" is greater than "MX", a fault occurs and the return value is FALSE.
  - If either "MX" or "MN" is non-numeric, a fault occurs and the return value is FALSE.

18.2.10.15 RAMP_INT

Ramps the output to the input setpoint at a defined rate.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
RAMP_INT	Ramp function (integer type)	FB		RAMP_INT_Instance_0(IN := , ASCEND := , DESCEND := , TIMEBASE := , RESET := , OUT =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input	Follows data type	-	Input
	ASCEND	Ascending limit	Follows data type	-	Ascending limit within the specified time. For example, if TIMEBASE = 1s and ASCEND = 20, the output ascends by at most 20 every 1s.
	DESCEND	Descending limit	Follows data type	-	Descending limit within the specified time. For example, if TIMEBASE = 500 ms and DESCEND = 10, the output descends by at most 10 every 500 ms.
	TIMEBASE	Set time	Follows data type	-	Set time
	RESET	Reset a bit	Follows data type	-	Reset
Output	OUT	Output	Follows data type	-	Output

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer							Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
ASCEND	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
DESCEND	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
TIMEBASE	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

- **Function**

  When the input undergoes a step change, the output gradually ramps to the new value within the specified time.

- **Program example**

  Example: Set IN = 0, ASCEND = 4, DESCEND = 8, TIMEBASE = T#2s, assuming the cycle operation period CYC is 1s.

Input	Judgment	Output
IN = 50	$\text{IN change} > \frac{CYC * ASCEND}{TIMEBASE}$ $50 > \frac{1 * 4}{2}$	$\text{OUT} = \text{Last cycle output} + \frac{CYC * ASCEND}{TIMEBASE}$ $\text{OUT} = \text{Last cycle output} + \frac{1 * 4}{2}$ The output will gradually ascend to 50.
IN = 20	$\text{IN Variation} < -\frac{CYC * \text{DESCEND}}{TIMEBASE}$ $-30 < \frac{1 * 8}{2}$	$\text{OUT} = \text{Rising-edge output} - \frac{CYC * \text{DESCEND}}{TIMEBASE}$ $\text{OUT} = \text{Rising-edge output} - \frac{1 * 8}{2}$ The output will gradually descend to 20.

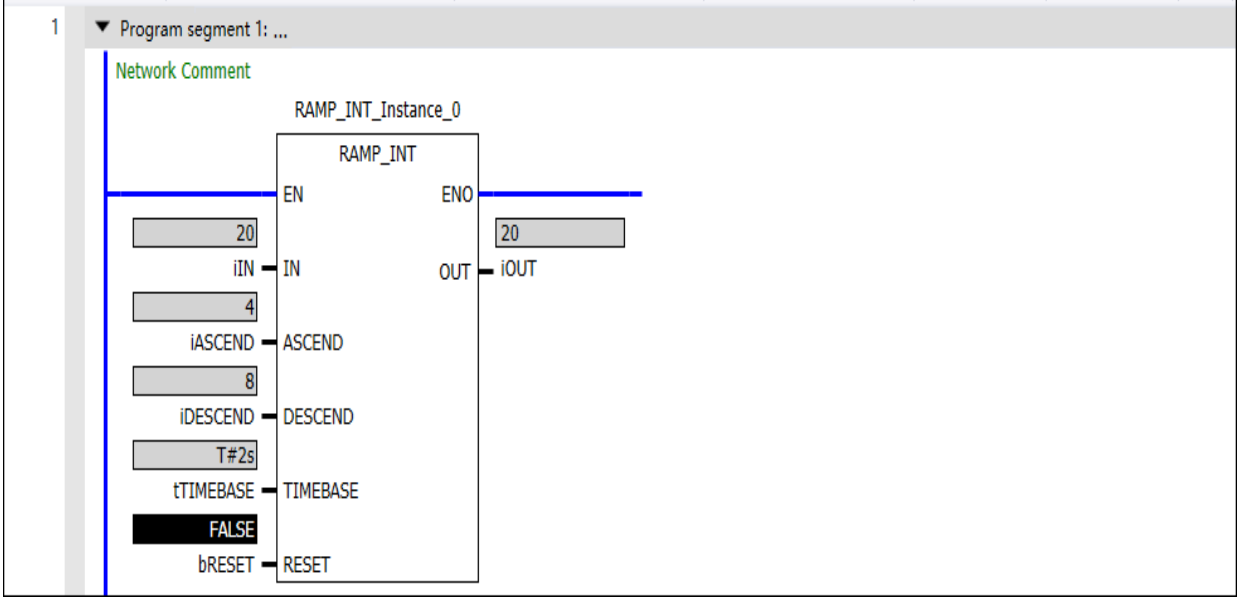
ST:

	Name	Data type	Address	Value
1	RAMP_INT_Instance_0	RAMP_INT		
2	iIN	INT		20
3	iASCEND	INT		4
4	iDESCEND	INT		8
5	tTIMEBASE	TIME		T#2s
6	bRESET	BOOL		FALSE
7	iOUT	INT		20

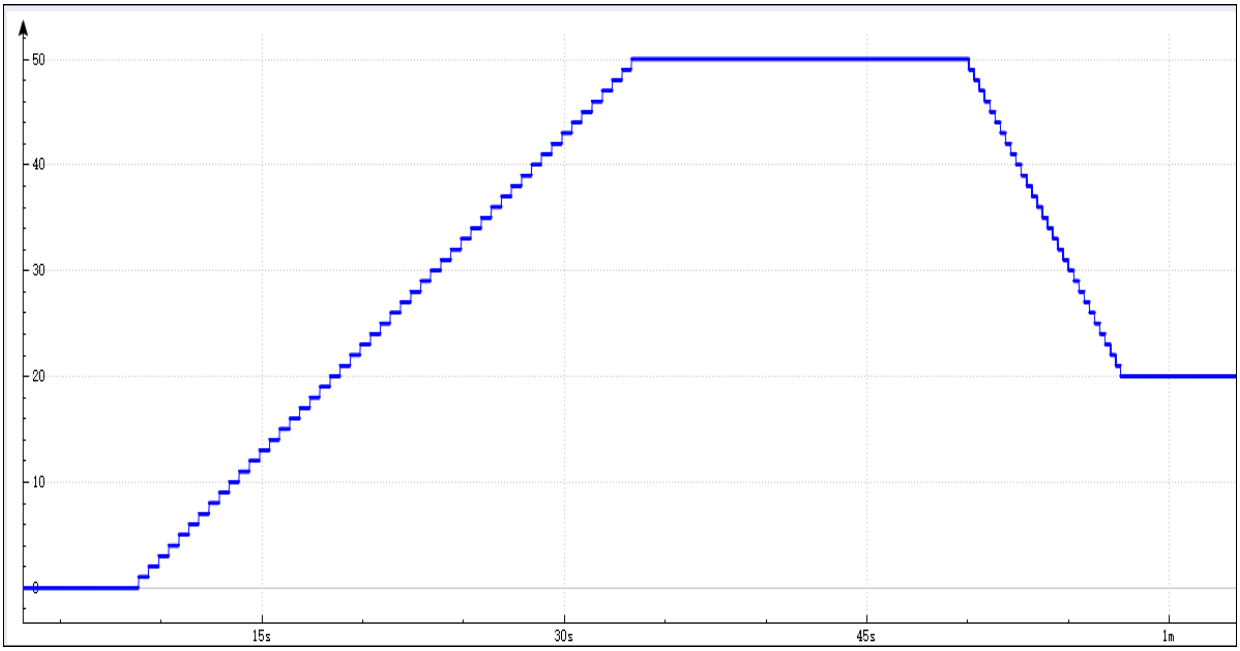
```
RAMP_INT_Instance_0(
IN    20    := iIN    20    ,
ASCEND    4    := iASCEND    4    ,
DESCEND    8    := iDESCEND    8    ,
TIMEBASE    T#2s    := tTIMEBASE    T#2s    ,
RESET    FALSE    := bRESET    FALSE    ,
OUT    20    => iOUT    20    );RETURN
```

LD:

	Name	Data type	Address	Value
1	RAMP_INT_Instance_0	RAMP_INT		
2	iIN	INT		20
3	iASCEND	INT		4
4	iDESCEND	INT		8
5	tTIMEBASE	TIME		T#2s
6	bRESET	BOOL		FALSE
7	iOUT	INT		20



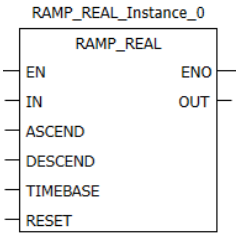
When the input "IN" undergoes a step change from 0 to 50 and then to 20, the value of the output variable "OUT" is shown below:



18.2.10.16 RAMP_REAL

Ramps the output to the input setpoint at a defined rate.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
RAMP_REAL	Ramp function (real number type)	FB		RAMP_REAL_Instance_0(IN:= ASCEND:= DESCEND := TIMEBASE := RESET:= OUT =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input	Follows data type	-	Input

Input/Output	Variable	Name	Valid Range	Default Value	Description
	ASCEND	Ascending limit	Follows data type	-	Ascending limit within the specified time. For example, if TIMEBASE = 1s and ASCEND = 20, the output ascends by at most 20 every 1s.
	DESCEND	Descending limit	Follows data type	-	Descending limit within the specified time. For example, if TIMEBASE = 500 ms and DESCEND = 10, the output descends by at most 10 every 500 ms.
	TIMEBASE	Set time	Follows data type	-	Set time
	RESET	Reset a bit	Follows data type	-	Reset
Output	OUT	Output	Follows data type	-	Output

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
ASCEND	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
DESCEND	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
TIMEBASE	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

- **Function**

  When the input undergoes a step change, the output gradually ramps to the new value within the specified time.

- **Program example**

  Example: Set IN = 0, ASCEND = 4, DESCEND = 8, TIMEBASE = T#2s, assuming the cycle operation period CYC is 1s.

Input	Judgment	Output
IN = 50	$\text{IN change} > \frac{CYC * ASCEND}{TIMEBASE}$ $50 > \frac{1 * 4}{2}$	$\text{OUT} = \text{Last cycle output} + \frac{CYC * ASCEND}{TIMEBASE}$ $\text{OUT} = \text{Last cycle output} + \frac{1 * 4}{2}$ The output will gradually ascend to 50.

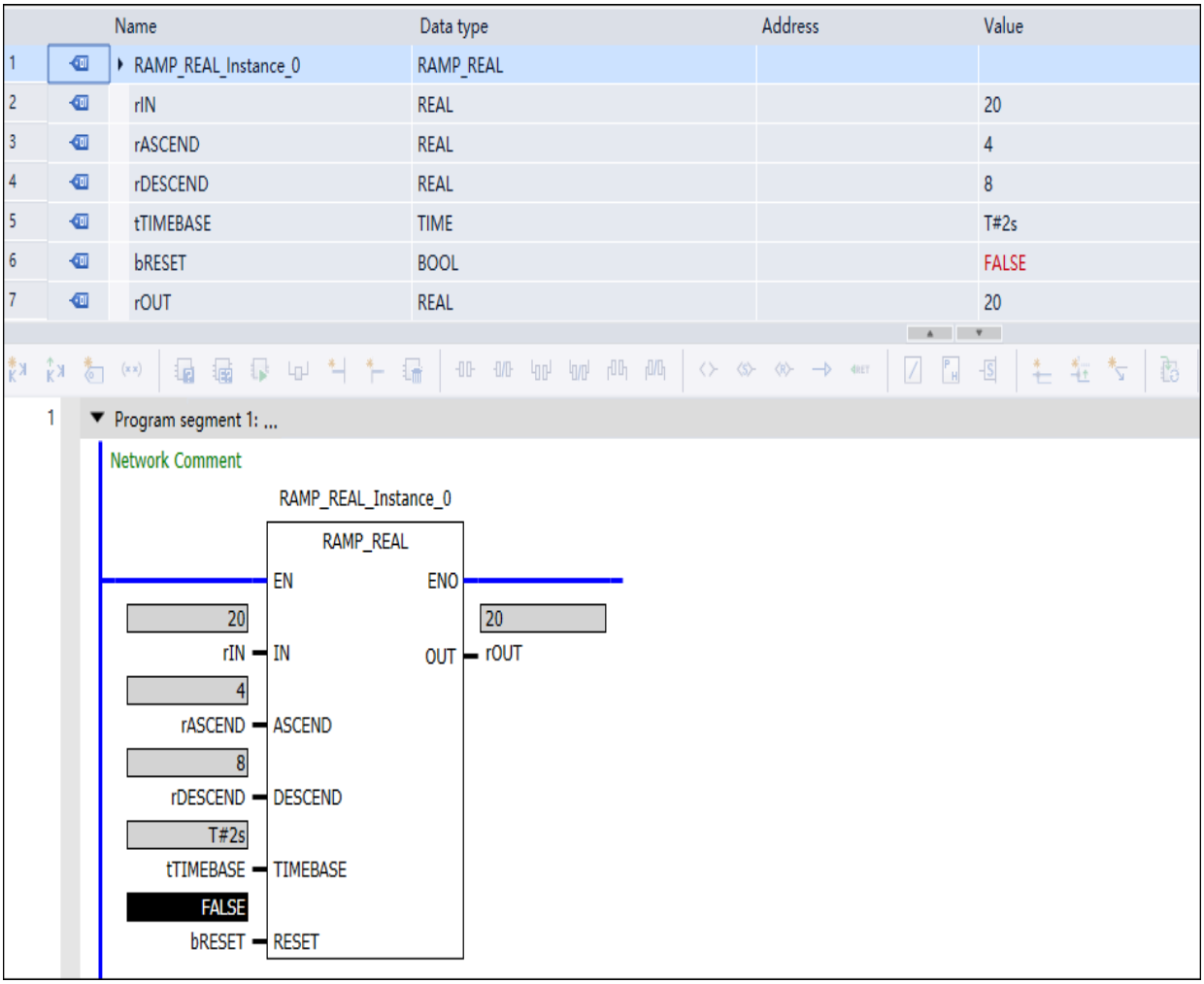
Input	Judgment	Output
IN = 20	$\text{IN Variation} < -\frac{CYC * \text{DESCEND}}{TIMEBASE}$ $-30 < \frac{1 * 8}{2}$	$\text{OUT} = \underset{\text{output}}{\text{Rising-edge}} - \frac{CYC * \text{DESCEND}}{TIMEBASE}$ $\text{OUT} = \underset{\text{output}}{\text{Rising-edge}} - \frac{1 * 8}{2}$ The output will gradually descend to 20.

ST:

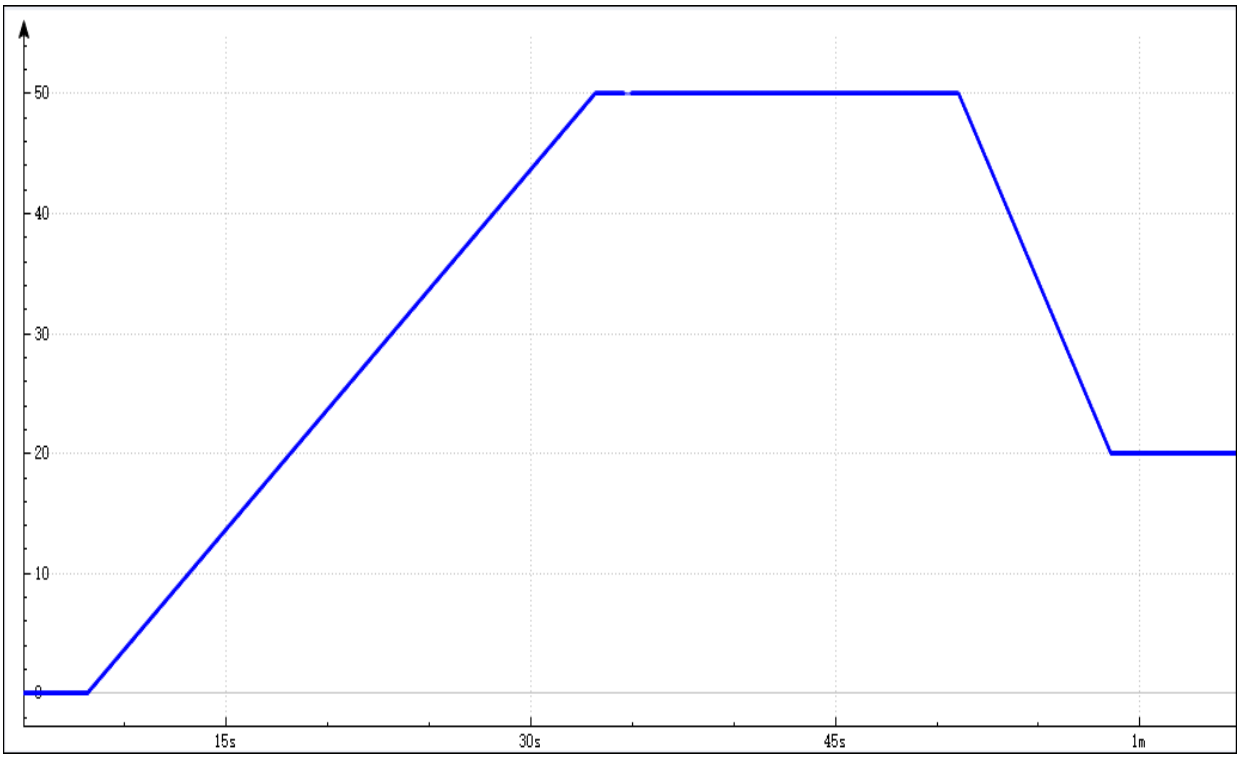
	Name	Data type	Address	Value
1	RAMP_REAL_Instance_0	RAMP_REAL		
2	rIN	REAL		20
3	rASCEND	REAL		4
4	rDESCEND	REAL		8
5	tTIMEBASE	TIME		T#2s
6	bRESET	BOOL		FALSE
7	rOUT	REAL		20

```
RAMP_REAL_Instance_0(
IN := rIN,
ASCEND := rASCEND,
DESCEND := rDESCEND,
TIMEBASE := tTIMEBASE,
RESET := bRESET,
OUT => rOUT);RETURN
```

LD:



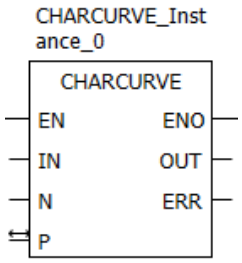
When the input "IN" undergoes a step change from 0 to 50 and then to 20, the value of the output variable "OUT" is shown below:



18.2.10.17 CHARCURVE

Maps the input values onto a characteristic curve.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CHARCURVE	Characteristic curve function	FB		CHARCURVE_Instance_0(IN :=, N :=, P :=, OUT =>, ERR =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input	Follows data type	-	-
	N	Number of valid points	2 ≤ N ≤ 11	-	Number of points defining the characteristic curve
Input/Output	P	Array of points	P ≤ 10	-	POINT: X: DINT (X value) Y: DINT (Y value)
Output	OUT	Output value	Follows data type	-	-
	ERR	Fault message	0, 1, 2, 4	-	0: No fault. 1: Points in P are not in ascending order. 2: Input is outside the limits of P. 4: Number of valid points N is out of range.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
IN	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
N	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
P	ARRAY [0..10] OF POINT																			
OUT	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
ERR	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Maps the input "IN" (X-value) to the output "OUT" (Y-value) based on the characteristic curve defined by 10 points specified by the array P.

- **Program example**

  Set "IN" to 8, the number of valid points N to 10, and the array of points P is shown in the table below.

Array Index	X	Y
0	-6	-1
1	-4	1
2	-2	2
3	1	4
4	5	5
5	9	10
6	14	12
7	16	15
8	20	20
9	22	24

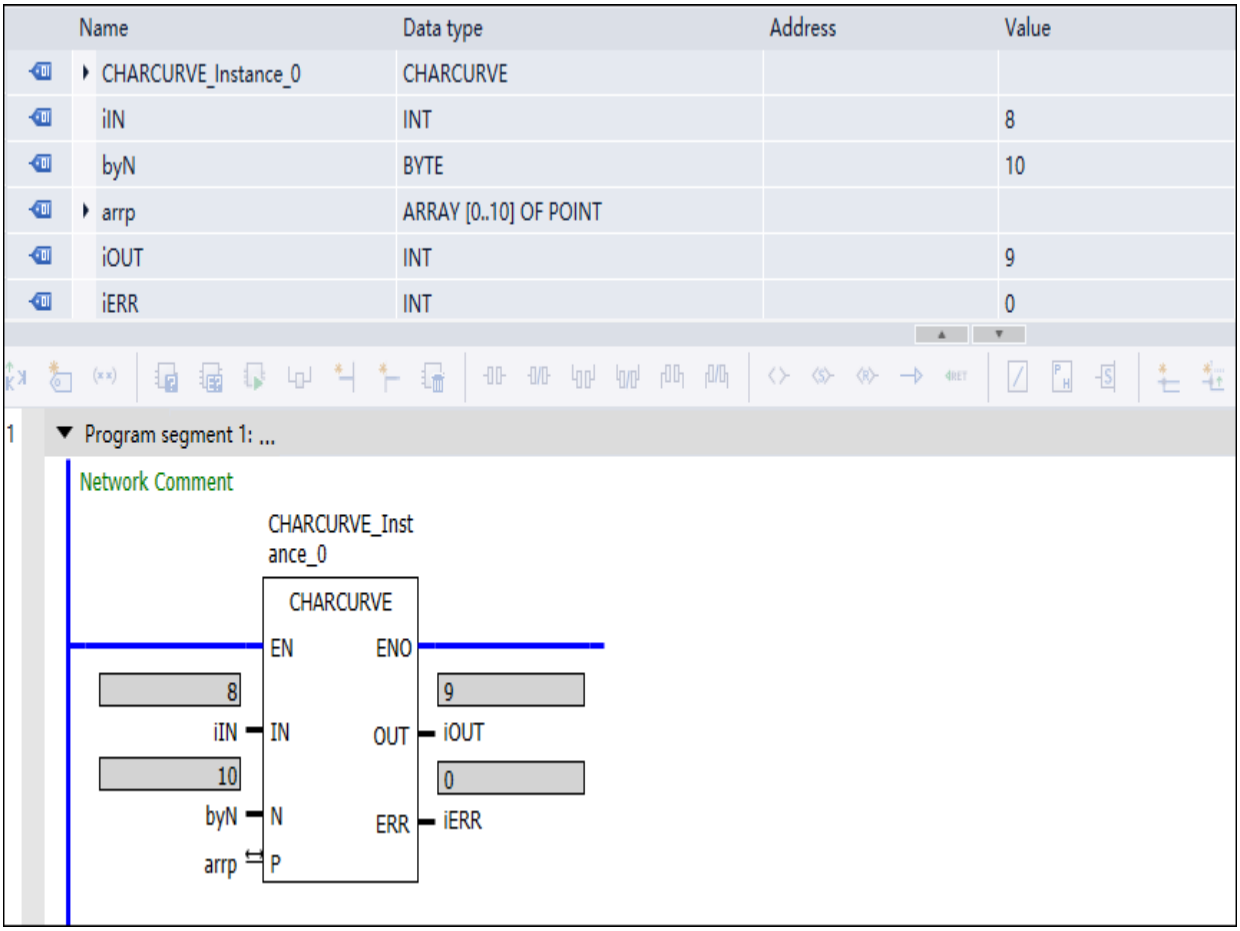
ST:

	Name	Data type	Address	Value
	CHARCURVE_Instance_0	CHARCURVE		
	iIN	INT		8
	byN	BYTE		10
	arrp	ARRAY [0..10] OF POINT		
	iOUT	INT		9
	iERR	INT		0

```
CHARCURVE_Instance_0(
IN  8   := iIN  8  ,
N  10  := byN  10  ,
P := arrp,
OUT  9   => iOUT  9  ,
ERR  0   => iERR  0  );RETURN
```

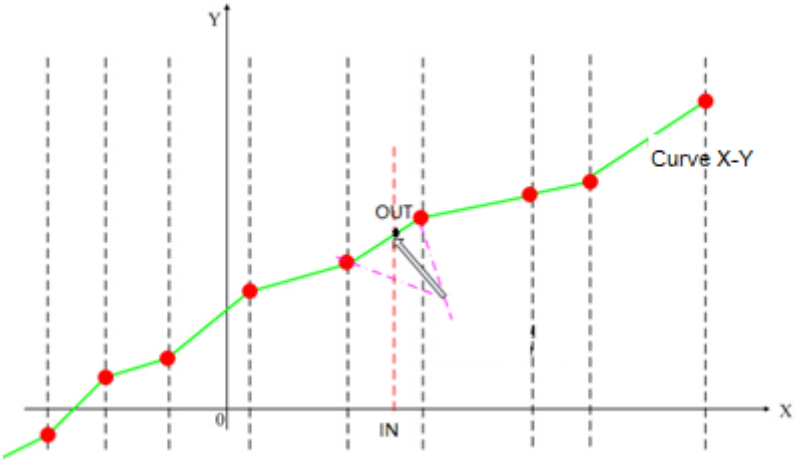
LD:

	Name	Data type	Address	Value
	CHARCURVE_Instance_0	CHARCURVE		
	iIN	INT		8
	byN	BYTE		10
	arrp	ARRAY [0..10] OF POINT		
	iOUT	INT		9
	iERR	INT		0



- **Precautions**
  - The X coordinates of the points in array P must be in ascending order for correct operation.
  - When "IN" > P[N].X, the output is outside the point boundary, "OUT" outputs P[N].Y, and "ERR" is 2.
  - When "IN" < P[0].X, the output is outside the point boundary, "OUT" outputs P[0].Y, and "ERR" is 2.
  - When "IN" is between the X coordinates of two points, the corresponding "OUT" is calculated by linear interpolation between the two points.

    The diagram below shows an example with N = 9:



18.2.10.18 BZAND_TAB

Controls an output value based on whether the input value is within the specified dead zone range. The dead zone range is defined by fLimitLow and fLimitHigh so that the input value fDataSrc is output to fDataOut outside the dead zone range.

- Instruction format

Instruction	Name	FB/FC	LD Representation	ST Representation
BZAND_TAB	Dead zone control	FC	BZAND_TAB EN ENO fLimitHigh fLimitLow fDataOut fDataSrc	:= BZAND_TAB(fLimitHigh := , fLimitLow := , fDataSrc := , fDataOut =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	fLimitHigh	Dead zone upper limit	Follows data type	0	Dead zone upper limit
	fLimitLow	Dead zone lower limit	Follows data type	0	Dead zone lower limit
	fDataSrc	Source data	Follows data type	0	Input value subject to dead zone control
Output	fDataOut	Source data	Follows data type	-	Output value from the dead zone

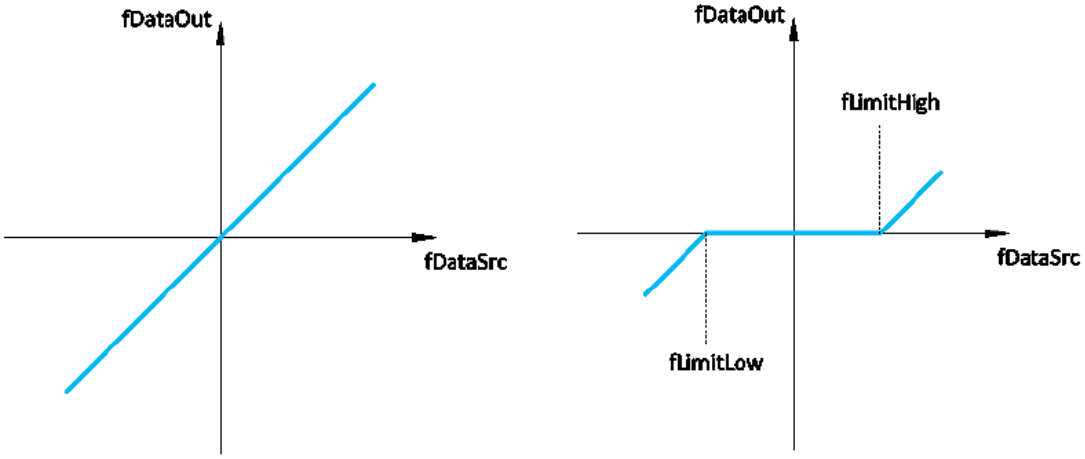
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
fLimitHigh	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
fLimitLow	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
fDataSrc	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
fDataOut	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

- **Function**

  Controls an output value based on whether the input value is within the specified dead zone range. The dead zone range is defined by fLimitLow and fLimitHigh so that the input value fDataSrc is output to fDataOut outside the dead zone range.

  When fLimitLow > fDataSrc, fDataSrc - fLimitLow => fDataOut;
  when fLimitHigh < fDataSrc, fDataSrc - fLimitHigh => fDataOut;
  when fLimitLow ≤ fDataSrc ≤ fLimitHigh, 0 => fDataOut.



- **Program example**

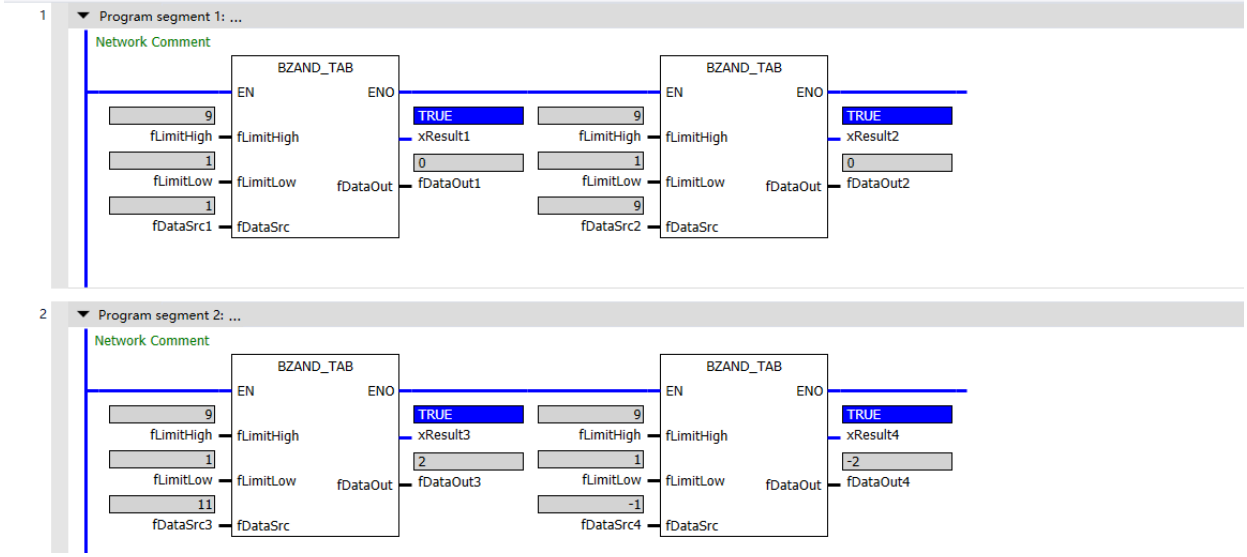
  ST:

	Name	Data type	Address	Value	Prepare value	Comment
1	fLimitHigh	REAL		9		
2	fLimitLow	REAL		1		
3	fDataSrc1	REAL		1		
4	fDataSrc2	REAL		9		
5	fDataSrc3	REAL		11		
6	fDataSrc4	REAL		-1		
7	fDataOut1	REAL		0		
8	fDataOut2	REAL		0		
9	fDataOut3	REAL		2		
10	fDataOut4	REAL		-2		
11	xResult1	BOOL		TRUE		
12	xResult2	BOOL		TRUE		
13	xResult3	BOOL		TRUE		
14	xResult4	BOOL		TRUE		

```
xResult1 TRUE := BZAND_TAB(fLimitHigh := fLimitHigh 9 , fLimitLow := fLimitLow 1 ,
fDataSrc := fDataSrc1 1 , fDataOut => fDataOut1 0 );
xResult2 TRUE := BZAND_TAB(fLimitHigh := fLimitHigh 9 , fLimitLow := fLimitLow 1 ,
fDataSrc := fDataSrc2 9 , fDataOut => fDataOut2 0 );
xResult3 TRUE := BZAND_TAB(fLimitHigh := fLimitHigh 9 , fLimitLow := fLimitLow 1 ,
fDataSrc := fDataSrc3 11 , fDataOut => fDataOut3 2 );
xResult4 TRUE := BZAND_TAB(fLimitHigh := fLimitHigh 9 , fLimitLow := fLimitLow 1 ,
fDataSrc := fDataSrc4 -1 , fDataOut => fDataOut4 -2 );RETURN
```

LD:

	Name	Data type	Address	Value	Prepare value	Comment
1	fLimitHigh	REAL		9		
2	fLimitLow	REAL		1		
3	fDataSrc1	REAL		1		
4	fDataSrc2	REAL		9		
5	fDataSrc3	REAL		11		
6	fDataSrc4	REAL		-1		
7	fDataOut1	REAL		0		
8	fDataOut2	REAL		0		
9	fDataOut3	REAL		2		
10	fDataOut4	REAL		-2		
11	xResult1	BOOL		TRUE		
12	xResult2	BOOL		TRUE		

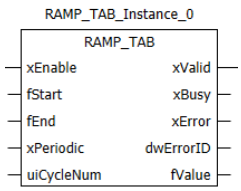


- **Precautions**
  - If fLimitHigh is less than fLimitLow, the instruction execution fails and returns FALSE, and the output variable remains at its initial value. .
  - The instruction returns TRUE upon successful execution, otherwise returns FALSE.

18.2.10.19 RAMP_TAB

Changes the output value from the minimum to maximum at a specified interval, and vice versa. This instruction aims to perform linear interpolation between two given data entries at a specified time interval, to output intermediate values sequentially based on the scanning time, until the end value is reached.

- Instruction format

Instruction	Name	FB/FC	LD Representation	ST Representation
RAMP_TAB	Ramp instruction	FB		RAMP_TAB_Instance_0(xEnable := , fStart := , fEnd := , xPeriodic := , uiCycleNum := , xValid => , xBusy => , xError => , dwErrorID => , fValue =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	xEnable	Execute	Follows data type	FALSE	Input high level to enable the function block
	Start	Start value	Follows data type	0	Start value
	fEnd	End value	Follows data type	0	End value
	xPeriodic	Cyclic output	Follows data type	FALSE	FALSE: Execute once; TRUE: Cyclic execution.
	uiCycleNum	Number of completed periods	Follows data type	0	Number of completed periods
Output	xValid	Completed	Follows data type	FALSE	FALSE: Function execution failed. TRUE: Function execution succeeded.
	xBusy	In progress	Follows data type	FALSE	FALSE: Function execution failed. TRUE: Function execution succeeded.
	xError	Fault	Follows data type	FALSE	FALSE: Function execution failed. TRUE: Function execution succeeded.

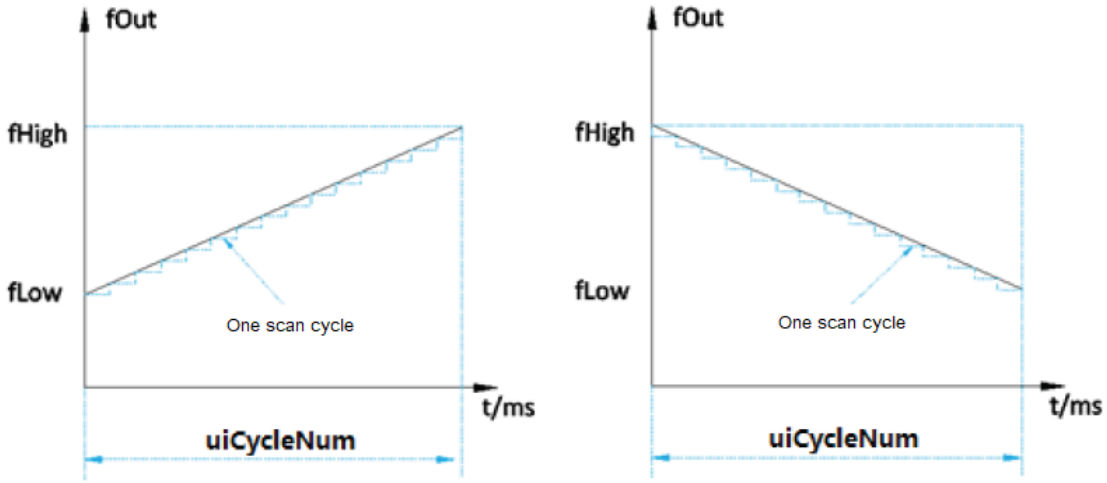
Input/Output	Variable	Name	Valid Range	Default Value	Description
	dwErrorID	Error code	Follows data type	0	Fault ID.
	fValue	Current value	Follows data type	0	Current value

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
xEnable	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Start	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
fEnd	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
xPeriodic	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
uiCycleNum	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
xValid	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
dwErrorID	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
fValue	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

- **Function**

  Changes the output value from the minimum to maximum at a specified interval, and vice versa. This instruction aims to perform linear interpolation between two given data entries at a specified time interval, to output intermediate values sequentially based on the scanning time, until the end value is reached.



- **Timing diagram**

- **Program example**

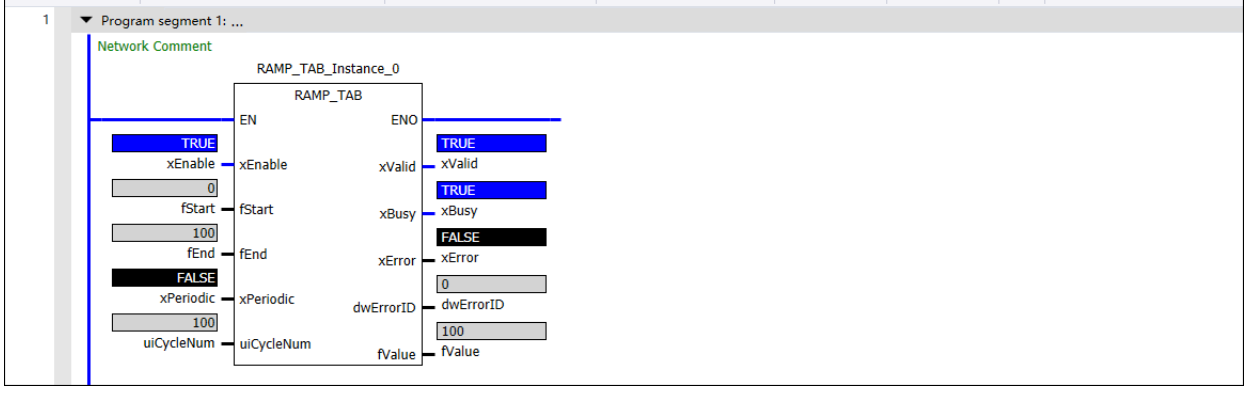
ST:

	Name	Data type	Address	Value	Prepare value	Comment
1	RAMP_TAB_Instance_0	RAMP_TAB				
2	xEnable	BOOL		TRUE		
3	fStart	REAL		0		
4	fEnd	REAL		100		
5	xPeriodic	BOOL		FALSE		
6	uiCycleNum	UINT		100		
7	xValid	BOOL		TRUE		
8	xBusy	BOOL		TRUE		
9	xError	BOOL		FALSE		
10	dwErrorID	DWORD		0		
11	fValue	REAL		100		

```
RAMP_TAB_Instance_0(
xEnable TRUE := xEnable TRUE,
fStart 0 := fStart 0,
fEnd 100 := fEnd 100,
xPeriodic FALSE := xPeriodic FALSE,
uiCycleNum 100 := uiCycleNum 100,
xValid TRUE => xValid TRUE,
xBusy TRUE => xBusy TRUE,
xError FALSE => xError FALSE,
dwErrorID 0 => dwErrorID 0,
fValue 100 => fValue 100);
RETURN
```

LD:

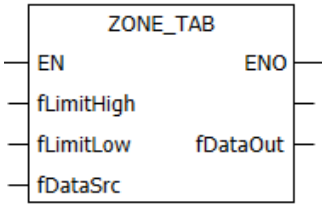
	Name	Data type	Address	Value	Prepare value	Comment
1	RAMP_TAB_Instance_0	RAMP_TAB				
2	xEnable	BOOL		TRUE		
3	fStart	REAL		0		
4	fEnd	REAL		100		
5	xPeriodic	BOOL		FALSE		
6	uiCycleNum	UINT		100		
7	xValid	BOOL		TRUE		
8	xBusy	BOOL		TRUE		
9	xError	BOOL		FALSE		
10	dwErrorID	DWORD		0		
11	fValue	REAL		100		



18.2.10.20 ZONE_TAB

Controls the output value using the specified deviation based on whether the input value is positive or negative (adding the deviation to the input value). The instruction adds fLimitHigh or fLimitLow to fDataSrc depending on its sign, and outputs the result to fDataOut.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ZONE_TAB	Zone control	FC		:= ZONE_TAB(fLimitHigh := , fLimitLow := , fDataSrc := , fDataOut =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	fLimitHigh	Positive deviation	Follows data type	0	Positive deviation, with the deviation value of the input value
	fLimitLow	Negative deviation	Follows data type	0	Negative deviation, with the deviation value of the input value
	fDataSrc	Source data	Follows data type	0	Input value.
Output	fDataOut	Source data	Follows data type	-	Output value

Variable types

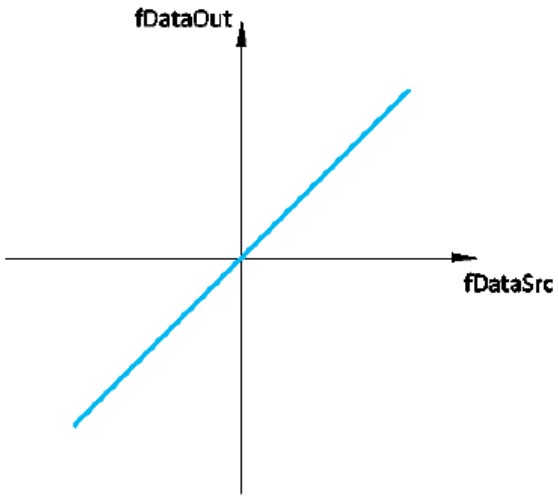
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
fLimitHigh	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

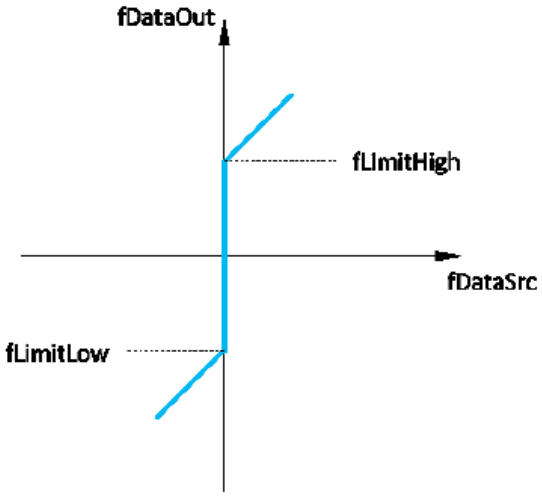
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
fLimitLow	_	_	_	_	_	_	_	_	_	_	_	_	_	√	_	_	_	_	_	_
fDataSrc	_	_	_	_	_	_	_	_	_	_	_	_	_	√	_	_	_	_	_	_
fDataOut	_	_	_	_	_	_	_	_	_	_	_	_	_	√	_	_	_	_	_	_

- **Function**

  Controls the output value using the specified deviation based on whether the input value is positive or negative (adding the deviation to the input value). The instruction adds fLimitHigh or fLimitLow to fDataSrc depending on its sign, and outputs the result to fDataOut.

  When fDataSrc < 0, fDataSrc + fLimitLow => fDataOut;
  when fDataSrc > 0, fDataSrc + fLimitHigh => fDataOut;
  when fDataSrc = 0, 0 => fDataOut.





- **Program example**

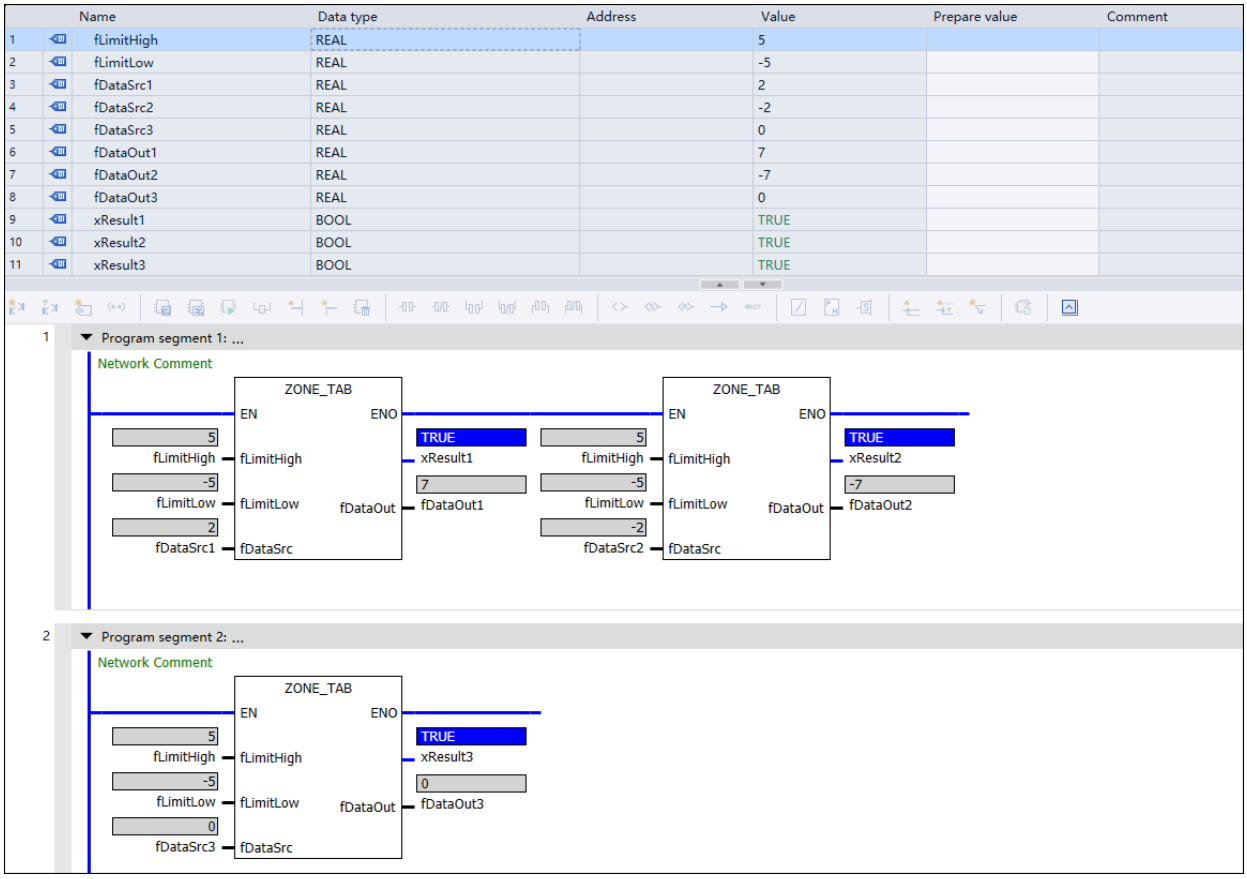
  ST:

	Name	Data type	Address	Value	Prepare value	Comment
1	fLimitHigh	REAL		5		
2	fLimitLow	REAL		-5		
3	fDataSrc1	REAL		2		
4	fDataSrc2	REAL		-2		
5	fDataSrc3	REAL		0		
6	fDataOut1	REAL		7		
7	fDataOut2	REAL		-7		
8	fDataOut3	REAL		0		
9	xResult1	BOOL		TRUE		
10	xResult2	BOOL		TRUE		
11	xResult3	BOOL		TRUE		

```
xResult1 := ZONE_TAB(fLimitHigh := fLimitHigh, fLimitLow := fLimitLow,
fDataSrc := fDataSrc1, fDataOut => fDataOut1);
xResult2 := ZONE_TAB(fLimitHigh := fLimitHigh, fLimitLow := fLimitLow,
fDataSrc := fDataSrc2, fDataOut => fDataOut2);
xResult3 := ZONE_TAB(fLimitHigh := fLimitHigh, fLimitLow := fLimitLow,
fDataSrc := fDataSrc3, fDataOut => fDataOut3);
RETURN
```

LD:

	Name	Data type	Address	Value	Prepare value	Comment
1	fLimitHigh	REAL		5		
2	fLimitLow	REAL		-5		
3	fDataSrc1	REAL		2		
4	fDataSrc2	REAL		-2		
5	fDataSrc3	REAL		0		
6	fDataOut1	REAL		7		
7	fDataOut2	REAL		-7		
8	fDataOut3	REAL		0		
9	xResult1	BOOL		TRUE		
10	xResult2	BOOL		TRUE		
11	xResult3	BOOL		TRUE		



18.2.11 File Operations

18.2.11.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.11.3 FileOpen	FB	Open file	OMUtils	Library version 1.0.0.0 and later
18.2.11.4 FileClose	FB	Close file		Library version 1.0.0.0 and later
18.2.11.5 FileSeek	FB	Seek file		Library version 1.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.11.6 FileTell	FB	Acquire position of file indicator		Library version 1.0.0.0 and later
18.2.11.7 FileFindList	FB	Query all files		Library version 1.0.0.0 and later
18.2.11.8 FileCopy	FB	Copy file		Library version 1.0.0.0 and later
18.2.11.9 Delete File	FB	Delete file		Library version 1.0.0.0 and later
18.2.11.10 FileRename	FB	Change file name		Library version 1.0.0.0 and later
18.2.11.11 DirCreate	FB	Create directory		Library version 1.0.0.0 and later
18.2.11.12 DirRemove	FB	Delete directory		Library version 1.0.0.0 and later
18.2.11.13.1 FileGets	FB	Get text string		Library version 1.0.0.0 and later
18.2.11.13.2 FilePuts	FB	Put text string		Library version 1.0.0.0 and later
18.2.11.13.3 FileRead	FB	Read file		Library version 1.0.0.0 and later
18.2.11.13.4 FileWrite	FB	Write file		Library version 1.0.0.0 and later
18.2.11.13.5 FileReadVar	FB	File read variable		Library version 1.0.0.0 and later
18.2.11.13.6 FileWriteVar	FB	File write variable		Library version 1.0.0.0 and later
18.2.11.14.1 FileGetsWstr	FB	Get text string		Library version 1.1.0.0 and later
18.2.11.14.2 FilePutsWstr	FB	Put text string		Library version 1.1.0.0 and later
18.2.11.14.3 FileReadWstr	FB	Read file		Library version 1.1.0.0 and later
18.2.11.14.4 FileWriteWstr	FB	Write file		Library version 1.1.0.0 and later
18.2.11.14.5 FileReadVarWstr	FB	File read variable		Library version 1.1.0.0 and later
18.2.11.14.6 FileWriteVarWstr	FB	File write variable		Library version 1.1.0.0 and later

18.2.11.2 Overview

- File operations support the PLC user directory and the SD card, with the following rules:
  - The absolute path of the SD card can be obtained via the SysHC_UDiskPath instruction. When this absolute path is input, the SD card will be operated; otherwise, the user directory of the PLC will be operated by default.
  - "/PLCprogram/" in the SD card is a system directory and is prohibited for user operations. Any attempt to access it will return error code 255.
  - OMUtils library version 1.3.0.0 and above supports PLC path operations. The path in version 1.3.0.0 is not retained after power failure. Therefore, it is recommended to use file operation instructions from OMUtils library version 1.4.0.0 and above.
- Currently, SD-related instructions are triggered by rising edge and executed according to the task cycle schedule. Therefore, to use file read/write, copy, rename, and other operations, it is necessary to set an appropriate task cycle based on the current controller model and file read/write speed; otherwise, the PLC may be abnormal.
- The current SD card does not support hot-swapping. Ensure the SD card is properly inserted and mounted before performing operations.
- If Execute becomes FALSE or the instruction execution time exceeds the task cycle, the instruction continues executing until completion. The value of Done changes to TRUE when processing is completed. Use this to confirm normal completion of processing.
- The PLC program runs under the Linux system; therefore, illegal characters follow Linux rules, which differ from Windows rules.
- All filenames mentioned in this section include absolute paths and are subject to total length restrictions.
- In SD card file operation instructions, file contents are stored as binary streams. To view or modify them on other platforms, use a binary-aware editor (e.g., UltraEdit, hex editor) to prevent unintended content alteration.
- The PLC built-in user directory capacity is 15 MB for EVO400/EVO500 and 1 GB for EVO700.
- The maximum number of opened files is 300. Exceeding this limit when executing the FileOpen instruction returns fault code 1536.
- In simulation mode, please enter the absolute path (case-sensitive): c:\mnt\sdcard\filename.

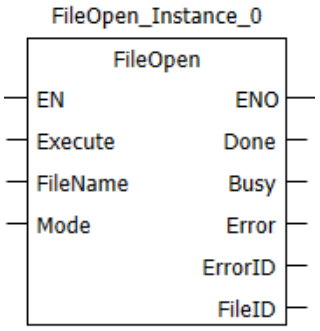


File operations on the PLC's user directory are supported only when the OMUtils library version is 1.3.0.0 or later.

18.2.11.3 FileOpen

Opens the specified file.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FileOpen	Open file	FB		FileOpen_Instance_0(Execute := , FileName :=, Mode := , Done => , Busy =>, Error =>, ErrorID => , FileID =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Enable	Follows data type	FALSE	(Triggered by edges) Execution of the function block switch
	FileName	Specified file name	Max. 200 characters	''	Name of the file to open
	Mode	Open mode	Follows data type	AM_READ	Mode in which to open file
Output	Done	Function block completed state	Follows data type	-	Function block completed state
	Busy	Function block execution flag	Follows data type	-	Function block execution flag
	Error	Function block fault flag	Follows data type	-	Function block fault flag
	ErrorID	Function block fault ID	Follows data type	-	Function block fault ID
	FileID	File ID	Follows data type	-	ID of the opened file

Variable types

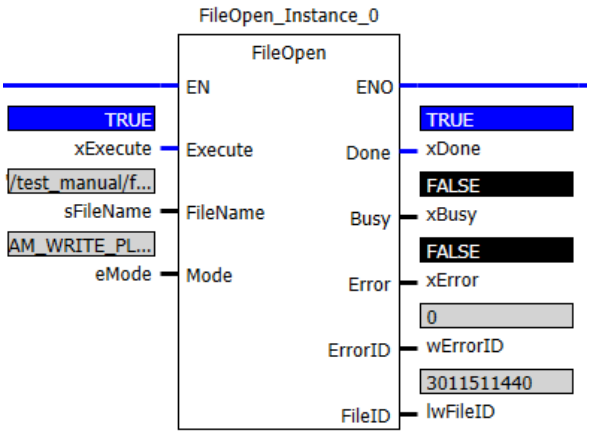
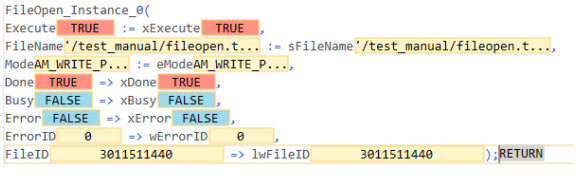
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FileName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
Mode	See "Function description" for the enumerators for ACCESS_MODE.																			
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FileID	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - Opens the file specified by FileName in the mode specified by Mode.
  - The file ID (FileID) is output, which can be used in the FileRead and FileWrite instructions to specify the file.
  - The data type of Mode is enumerated type _eSEARCH_MODE. The meanings of the enumerators are as follows:

Enumeration Value	Definition
AM_READ	If the file exists in the target path, the file is opened in read-only mode. If the file does not exist, the fault ID 16 is returned.
AM_WRITE	If the file exists in the target path, the file is opened in re-write mode. If the file does not exist, a new file with the write permission is automatically created.
AM_APPEND	If the file exists in the target path, open the file in write-only mode and the content is appended to the end of the file. If the file does not exist, the fault ID 16 is returned.

Enumeration Value	Definition
AM_READ_PLUS	If the file exists in the target path, the file is opened in read-write mode. If the file does not exist, the fault ID 16 is returned.
AM_WRITE_PLUS	If the file exists in the target path, the file is opened in re-read-write mode. If the file does not exist, a new file with the read/write permission is automatically created.
AM_APPEND_PLUS	If the file exists in the target path, the file is opened in read-write mode and the content is appended to the end of the file. If the file does not exist, a new file with the read/write permission is automatically created.

- **Program example**

Item	LD	ST
Define variables	VAR FileOpen_Instance_0: FileOpen; xExecute: BOOL; sFileName: STRING[200] := '/test_manual/fileopen.txt'; eMode: ACCESS_MODE := ACCESS_MODE.AM_WRITE_PLUS; xDone: BOOL; xBusy: BOOL; xError: BOOL; wErrorID: WORD; lwFileID: LWORD; END_VAR	
Program		

Item	LD	ST

Operation result

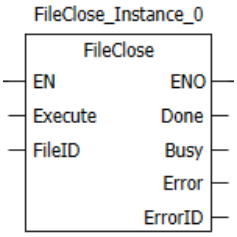
Name	Data type	Address	Value
FileOpen_Instance_0	FileOpen		
xExecute	BOOL		TRUE
sFileName	STRING[200]		'/test_manual/fileopen.txt'
eMode	ACCESS_MODE		AM_WRITE_PLUS
xDone	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
wErrorID	WORD		0
lwFileID	LWORD		3011511440

- **Precautions**
  - Before using this instruction, please read the 18.2.11.2 Overview to avoid improper operations.
  - To use instructions such as FileSeek, FileRead, FileWrite, FileGets, and FilePuts that require a FileID input, execute this instruction first to obtain the corresponding FileID.
  - After using the file opened by this instruction, execute the FileClose instruction to close the file.
  - After this instruction is completed, the file ID is output to FileID. The value of Done changes to TRUE.
  - If the SD card is removed while a file is open, the file remains in an open state. However, when the SD card is reinserted, the file cannot be read from or written to. To read and write the file, re-open the file.
  - An exception occurs and Error becomes TRUE in the following cases:
    - If the SD card is write-protected and the file is opened in a mode other than read-only (AM_READ), ErrorID is 16.
    - If the value of Mode is AM_READ, AM_APPEND, or AM_READ_PLUS, and the file specified by FileName does not exist, ErrorID is 16.
    - If FileName is invalid and the file is opened in a mode other than read-only (AM_READ), ErrorID is 16.
    - If the file is not closed, and the instruction is executed again to open a file with the same name, ErrorID is 1.
    - If the SD card is not successfully mounted, data stored on the card cannot be accessed, and ErrorID is 2.
    - If multiple threads execute the FileOpen instruction simultaneously, ErrorID is 1539.

18.2.11.4 FileClose

Closes the specified file.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FileClose	Close file	FB		FileClose_Instance_0(Execute := , FileID := , Done => , Busy =>, Error =>, ErrorID =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Enable	Follows data type	FALSE	(Triggered by edges) Execution of the function block switch
	FileID	File ID	Follows data type	0	ID of the file to close
Output	Done	Function block completed state	Follows data type	-	Function block completed state
	Busy	Function block execution flag	Follows data type	-	Function block execution flag
	Error	Function block fault flag	Follows data type	-	Function block fault flag
	ErrorID	Function block fault ID	Follows data type	-	Function block fault ID

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

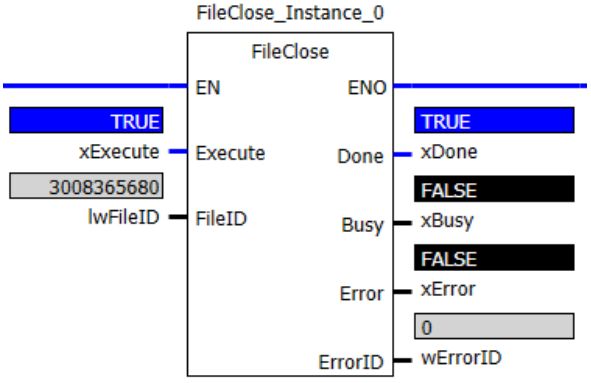
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
FileID	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Closes the file specified by FileID

- **Program example**

  Note: The ID corresponding to the variable lwFileID is generated by the FileOpen instruction.

Item	LD	ST
Define variables	VAR FileOpen_Instance_0: FileOpen; xFileName: STRING[242] := '/test_manual/fileopen.txt'; eMode: ACCESS_MODE := ACCESS_MODE.AM_WRITE_PLUS; FileClose_Instance_0: FileClose; xExecute: BOOL; lwFileID: LWORD; xDone: BOOL; xBusy: BOOL; xError: BOOL; wErrorID: WORD; END_VAR	
Program		FileClose_Instance_0(Execute TRUE := xExecute TRUE, FileID 3008365680 := lwFileID 3008365680, Done TRUE => xDone TRUE, Busy FALSE => xBusy FALSE, Error FALSE => xError FALSE, ErrorID 0 => wErrorID 0);RETURN

Item	LD	ST

Operation result

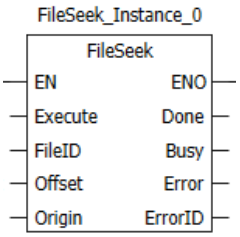
	Name	Data type	Address	Value
1	FileOpen_Instance_0	FileOpen		
2	xFileName	STRING[242]		'/test_manual/fileopen.txt'
3	eMode	ACCESS_MODE		AM_WRITE_PLUS
4	FileClose_Instance_0	FileClose		
5	xExecute	BOOL		TRUE
6	lwFileID	LWORD		3008365680
7	xDone	BOOL		TRUE
8	xBusy	BOOL		FALSE
9	xError	BOOL		FALSE
10	wErrorID	WORD		0

- **Precautions**
  - Before using this instruction, please read the 18.2.11.2 Overview to avoid improper operations.
  - Do not execute this instruction on a file for multiple consecutive times; otherwise, the PLC may fail to run and even break down.
  - Execute the FileOpen instruction in advance to obtain the value for FileID.
  - After using the file opened by the FileOpen instruction, execute this instruction to close the file, otherwise, the result will be unexpected. If the SD card is removed while a file is open, the file remains in an open state. However, when the SD card is reinserted, the file cannot be read from or written to. To read and write the file, re-open the file.
  - An exception occurs and Error becomes TRUE in the following cases:
    - If the file specified by FileID does not exist, ErrorID is 1.
    - If a file opened by the FileOpen instruction is closed again by FileClose after already being closed, ErrorID is 1538.
    - If the input FileID is invalid (less than 0 or greater than 300), ErrorID is 1537.
    - If multiple threads execute the FileClose instruction simultaneously, ErrorID is 1539.

18.2.11.5 FileSeek

Sets a file position indicator in the specified file.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FileSeek	Seek file	FB		FileSeek_Instance_0(Execute := , FileID := , Offset := , Origin := , Done => , Busy =>, Error =>, ErrorID =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Enable	Follows data type	FALSE	(Triggered by edges) Execution of the function block switch
	FileID	File ID	Follows data type	0	ID of the file in which to set the file position indicator
	Offset	Offset	Follows data type	0	Offset of the file position indicator from Origin
	Origin	Reference position	Follows data type	_SEEK_SET	Reference position for file position indicator
Output	Done	Function block completed state	Follows data type	-	Function block completed state
	Busy	Function block execution flag	Follows data type	-	Function block execution flag
	Error	Function block fault flag	Follows data type	-	Function block fault flag

Input/Output	Variable	Name	Valid Range	Default Value	Description
	ErrorID	Function block fault ID	Follows data type	-	Function block fault ID

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FileID	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Offset	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Origin	See "Function description" for the enumerators for _eFSEEK_ORIGIN.																			
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction sets a file position indicator in the file specified by FileID. A file position indicator is the position in a file at which to start reading or writing when an instruction such as the FileRead or FileWrite instruction is executed. For example, to read from the beginning of a file, set a file position indicator at the beginning of the file with the FileSeek instruction, and then execute the FileRead instruction.
  - The file position indicator is set at offset Offset from reference position Origin. The data type of Origin is enumerated type _eFSEEK_ORIGIN. The meanings of the enumeration values are as follows in the table below.

Enumeration Value	Definition
_SEEK_SET	Beginning of the file
_SEEK_CUR	Current position of the file position indicator

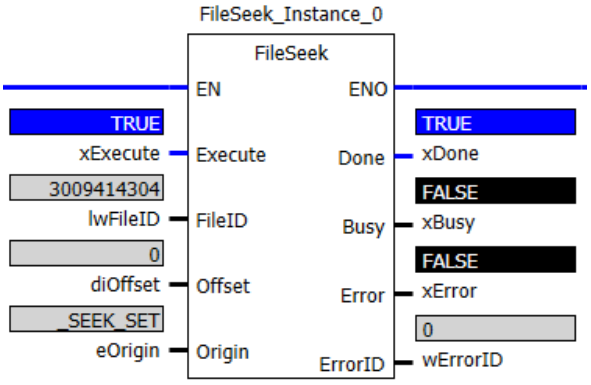
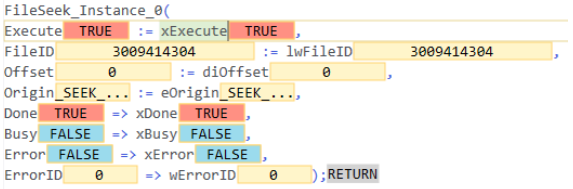
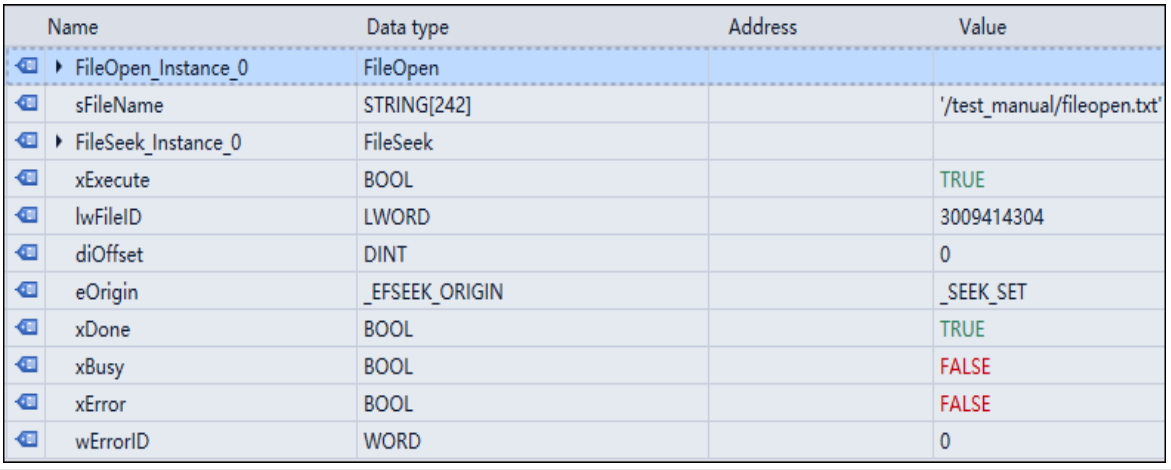
Enumeration Value	Definition
_SEEK_END	End of the file

- **Program example**

  Note: The ID corresponding to the variable lwFileID is generated by the FileOpen instruction.

Item	LD	ST

Define variables	VAR
	FileOpen_Instance_0: FileOpen;
	sFileName: STRING[242] := '/test_manual/fileopen.txt';
	FileSeek_Instance_0: FileSeek;
	xExecute: BOOL;
	lwFileID: LWORD;
	diOffset: DINT:= 0;
	eOrigin: _eFSEEK_ORIGIN := _EFSEEK_ORIGIN._SEEK_SET;
	xDone: BOOL;
	xBusy: BOOL;
	xError: BOOL;
	wErrorID: WORD;
	END_VAR	

Program		
Operation result		

Name	Data type	Address	Value
FileOpen_Instance_0	FileOpen		
sFileName	STRING[242]		'/test_manual/fileopen.txt'
FileSeek_Instance_0	FileSeek		
xExecute	BOOL		TRUE
lwFileID	LWORD		3009414304
diOffset	DINT		0
eOrigin	_EFSEEK_ORIGIN		_SEEK_SET
xDone	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
wErrorID	WORD		0

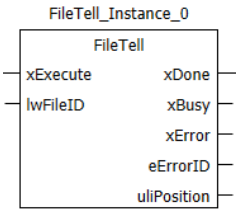
- **Precautions**
  - Before using this instruction, please read the 18.2.11.2 Overview to avoid improper operations.
  - Execute the FileOpen instruction in advance to obtain the value for FileID.
  - An exception occurs and Error becomes TRUE in the following cases:
    - If the file specified by FileID does not exist, ErrorID is 1.

- If the file position indicator specified by Origin and Offset is located before the start of the file, ErrorID is 1.
- If the input FileID is invalid (less than 0 or greater than 300), ErrorID is 1537.
- If a file opened by the FileOpen instruction is used by FileSeek after already being closed by FileClose, ErrorID is 1538.

18.2.11.6 FileTell

Gets the file position indicator in the specified file.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FileTell	Acquire position of file indicator	FB		FileTell_Instance_0(xExecute := , lwFileID := , xDone => , xBusy => , xError => , eErrorID => , uliPosition =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	xExecute	Enable	Follows data type	FALSE	Enable the instruction
	lwFileID	File ID	Follows data type	0	File ID
Output	xDone	Function block completed state	Follows data type	-	Function block completed state
	xBusy	Function block execution flag	Follows data type	-	Function block execution flag
	xError	Function block fault flag	Follows data type	-	Function block fault flag
	eErrorID	Function block fault ID	Follows data type	-	Function block fault ID
	uliPosition	Position in file	Follows data type	-	Position in file

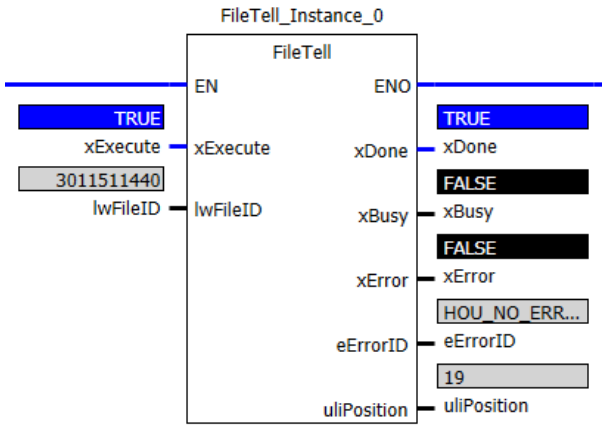
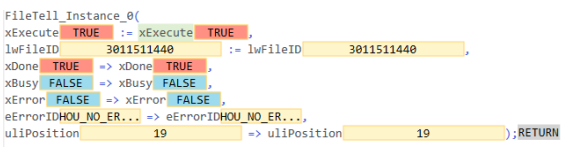
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
xExecute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
lwFileID	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eErrorID	See "Function description" for the enumerators for HOU_ERROR.																			
uliPosition	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction gets the position of the file position indicator in the file specified by lwFileID. A file position indicator is the position in a file at which to start reading or writing when an instruction such as the FileRead or FileWrite instruction is executed.
  - The file ID lwFileID is obtained via the FileOpen instruction. The file position indicator is set in the specified file using the FileSeek instruction, and then obtained using FileTell. Then, use FileClose instruction to close the file.
- **Program example**

Note: The ID corresponding to the variable lwFileID is generated by the FileOpen instruction.

Item	LD	ST
Define variables	VAR FileOpen_Instance_0: FileOpen; sFileName: STRING[200] := '/test_manual/filetell.txt'; eMode: ACCESS_MODE := ACCESS_MODE.AM_WRITE_PLUS; FileTell_Instance_0: FileTell; xExecute: BOOL; lwFileID: LWORD; xDone: BOOL; xBusy: BOOL; xError: BOOL; eErrorID: HOU_ERROR; uliPosition: ULINT; END_VAR	

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>
</td><td>
</td></tr>
<tr><td>Operation result</td><td colspan="2">
<table>
<tr><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td>FileOpen_Instance_0</td><td>FileOpen</td><td></td><td></td></tr>
<tr><td>sFileName</td><td>STRING[200]</td><td></td><td>'/test_manual/filetell.txt'</td></tr>
<tr><td>eMode</td><td>ACCESS_MODE</td><td></td><td>AM_WRITE_PLUS</td></tr>
<tr><td>FileTell_Instance_0</td><td>FileTell</td><td></td><td></td></tr>
<tr><td>xExecute</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>lwFileID</td><td>LWORD</td><td></td><td>3011511440</td></tr>
<tr><td>xDone</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>xBusy</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>xError</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>eErrorID</td><td>HOU_ERROR</td><td></td><td>HOU_NO_ERROR</td></tr>
<tr><td>uliPosition</td><td>ULINT</td><td></td><td>19</td></tr>
</table>
</td></tr>
</table>

- **Precautions**
  - Execution of this instruction is continued until completion even if the value of xExecute changes to FALSE or the execution time exceeds the task cycle. Use the value of xDone to determine whether the instruction is completed, which becomes TRUE upon completion.
  - Execute the FileOpen instruction in advance to obtain the value for lwFileID.
  - An exception occurs under the following conditions. The value of xError is TRUE.
    - The SD memory card is unavailable.
    - The file specified by lwFileID is being accessed.
    - The file specified by lwFileID does not exist.
    - An error that prevents access occurs during SD memory card access.
    - If the input FileID is invalid (less than 0 or greater than 300), ErrorID is 1537.
    - If a file opened by the FileOpen instruction is used by FileTell after already being closed by FileClose, ErrorID is 1538.
- **Fault ID**

eErrorID	Value	Note
HOU_NO_ERROR	0	No fault.
HOU_INTERNAL_ERROR	1	General internal fault
HOU_INVALIDPARAMETER_ERROR	2	Invalid instruction parameters

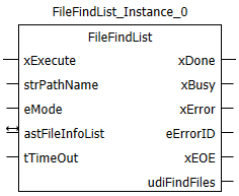
eErrorID	Value	Note
HOU_NOTINITIALIZED_ERROR	3	Component not initialized, failure to execute the instruction
HOU_VERSION_ERROR	4	Version conflict
HOU_TIMEOUT_ERROR	5	Instruction timeout
HOU_NOBUFFER_ERROR	6	Insufficient memory, failure to execute the instruction
HOU_PENDING_ERROR	10	Instruction calling not completed
HOU_NUMPENDING_ERROR	11	Excessive instruction calls
HOU_NOTIMPLEMENTED_ERROR	12	Instruction not implemented
HOU_INVALIDID_ERROR	13	Invalid ID input
HOU_OVERFLOW_ERROR	14	Integer overflow
HOU_BUFFER_SIZE_ERROR	15	Buffer too small or invalid
HOU_NOOBJECT_ERROR	16	Input named object not existing
HOU_NOMEMORY_ERROR	17	No heap memory available
HOU_DUPLICATE_ERROR	18	Another object with the same name existing
HOU_MEMORYOVERWRITE_ERROR	19	Heap memory write out of bounds, memory overwrite error
HOU_INVALIDHANDLE_ERROR	20	Invalid object handle
HOU_ENDOFOBJECT_ERROR	21	Object end reached
HOU_NOCHANGE_ERROR	22	No change
HOU_INVALIDINTERFACE_ERROR	23	Invalid or unknown interface
HOU_NOTSUPPORTED_ERROR	24	Function not supported
HOU_NOACCESSRIGHTS_ERROR	25	No access permission
HOU_OUTOFLIMITS_ERROR	26	Resource limit exceeded
HOU_ENTRIESREMAINING_ERROR	27	Buffer limit causing transmission incompletion
HOU_INVALIDSESSIONID_ERROR	28	Invalid online session ID
HOU_EXCEPTION_ERROR	29	Exception

eErrorID	Value	Note
HOU_INVALIDPATHNAME_ERROR	33	Path name invalid
HOU_INVALIDFILENAME_ERROR	34	File name invalid
HOU_INVALIDLISTMODE_ERROR	35	Invalid control command
HOU_FILEOVERWRITE_ERROR	50	File overwrite fault
HOU_OVERBUFFERSIZE_ERROR	1030	Data area too small
HOU_SAMEFILENAME_ERROR	1403	File with the same name existing

18.2.11.7 FileFindList

Queries all files or subdirectories with specified names in a directory.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FileFindList	Query all files	FB		FileFindList_Instance_0(xExecute := , strPathName := , eMode := , astFileInfoList := , tTimeOut := , xDone => , xBusy => , xError => , eErrorID => , xEOE => , udiFindFiles =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	xExecute	Enable	Follows data type	FALSE	Enable the instruction
	strPathName	Directory name	Follows data type	-	Directory name

Input/Output	Variable	Name	Valid Range	Default Value	Description
	eMode	Query control command	Follows data type	-	Query of the control command in the directory: LISTCMD_FIRST, LISTCMD_NEXT, LISTCMD_ABORT
	TimeOut	Timeout period	Follows data type	T#5s	Timeout period
Input/Output	astFileInfoList	File information array	Follows data type	-	File information array
Output	xDone	Function block completed state	Follows data type	-	Function block completed state
	xBusy	Function block execution flag	Follows data type	-	Function block execution flag
	xError	Function block fault flag	Follows data type	-	Function block fault flag
	eErrorID	Function block fault ID	Follows data type	-	Function block fault ID
	xEOE	End of directory enumeration	Follows data type	-	End of directory enumeration
	udiFindFiles	Directory enumeration count	Follows data type	-	Directory enumeration count

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
xExecute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
strPathName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
eMode	See "Function description" for the enumerators for FILELIST_MODE.																			
astFileInfoList	See "Function description" for details about the structure array Array [*] of FILEINFO.																			

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
TimeOut	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
xDone	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xBusy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
xError	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
eErrorID	See "Function description" for the enumerators for HOU_ERROR.																			
xEOE	√	–	–	–	–	–	–	–	_	–	–	–	–	–	–	–	–	–	–	–
udiFindFiles	–	–	–	–	–	–	–	√	_	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction queries the files or subdirectories with names similar to the specified names in the directory.
  - The input parameter strPathName specifies the target directory and the name of the entry to be searched for. The file name to be searched for supports two types of wildcards (* and ?).
  - The input parameter eMode, whose data type is the FILELIST_MODE enumerator, specifies whether the search is conducted for the first time or in sequential mode. It is used to navigate in the entry list. The meanings of the enumerators are as follows.

Enumeration Value	Definition
LISTCMD_FIRST	It means to search for the first time. The number of files found is determined by the size of the astFileInfoList array and the number of similar files in the directory.
LISTCMD_NEXT	It means to search in sequential mode. After the first search is completed, it is called multiple times to find all similar files. The number of files found is determined by the size of the astFileInfoList array and the number of similar files in the directory.
LISTCMD ABORT	It means to abort the search and close the related directory handle.

  - The input/output parameter astFileInfoList (array of structures) stores the information of the files found. The meanings of the FILEINFO structure are as follows.

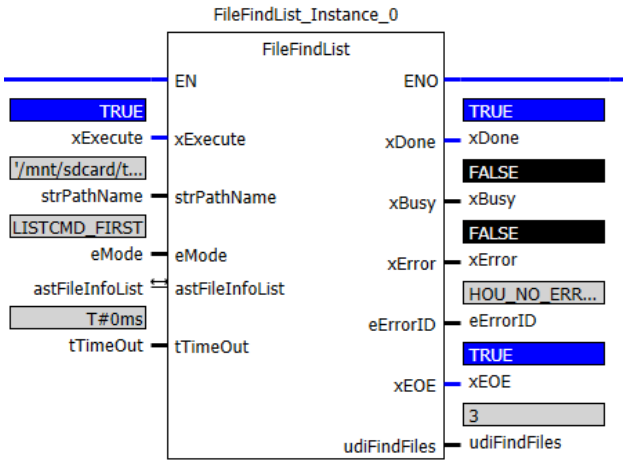
Structure Member	Data Type	Description
strFileName	STRING[255]	File or directory name
xDirectory	BOOL	File or directory

Structure Member	Data Type	Description
File size	ULINT	File size, in bytes
dtCreationTime	DATE_AND_TIME	Creation time of the file or directory
dtLastAccessTime	DATE_AND_TIME	Last access time of the file or directory
dtLastModificationTime	DATE_AND_TIME	Last write time of the file or directory

- **Program example**

Item	LD	ST
Define variables		
Program		

```
VAR
    FileFindList_Instance_0: FileFindList;
    xExecute: BOOL;
    strFileName: STRING[255] := '/mnt/sdcard/test_manual/*txt';
    eMode: FILELIST_MODE := FILELIST_MODE.LISTCMD_FIRST;
    astFileInfoList: ARRAY[1..5] OF FileInfo;
    tTimeOut: TIME;
    xDone: BOOL;
    xBusy: BOOL;
    xError: BOOL;
    eErrorID: HOU_ERROR;
    xEOE: BOOL;
    udiFindFiles: UDINT;
END_VAR
```




```
FileFindList_Instance_0(
xExecute TRUE := xExecute TRUE,
strPathName'/mnt/sdcard/test_manual/*t... := strPathName'/mnt/sdcard/test_manual/*t...,
eModeLISTCMD_FI... := eModeLISTCMD_FI...,
astFileInfoList 3012568152 := astFileInfoList,
tTimeOut T#0ms := tTimeOut T#0ms,
xDone TRUE => xDone TRUE,
xBusy FALSE => xBusy FALSE,
xError FALSE => xError FALSE,
eErrorIDHOU_NO_ER... => eErrorIDHOU_NO_ER...,
xEOE TRUE => xEOE TRUE,
udiFindFiles 3 => udiFindFiles 3 );RETURN
```

Item	LD	ST

Operation result

Name	Data type	Address	Value
FileFindList_Instance_0	FileFindList		
xExecute	BOOL		TRUE
strPathName	STRING[255]		'/mnt/sdcard/test_manual/*txt'
eMode	FILELIST_MODE		LISTCMD_FIRST
astFileInfoList	ARRAY [1..5] OF FileInfo		
astFileInfoList[1]	FileInfo		
strFileName	STRING[255]		'test3.txt'
xDirectory	BOOL		FALSE
uliFileSize	ULINT		0
dtCreationTime	DATE_AND_TIME		DT#1970-1-2-2:32:50
dtLastAccessTime	DATE_AND_TIME		DT#1970-1-2-2:32:50
dtLastModificationTime	DATE_AND_TIME		DT#1970-1-2-2:32:50
astFileInfoList[2]	FileInfo		
strFileName	STRING[255]		'test2.txt'
xDirectory	BOOL		FALSE
uliFileSize	ULINT		0
dtCreationTime	DATE_AND_TIME		DT#1970-1-2-2:32:46
dtLastAccessTime	DATE_AND_TIME		DT#1970-1-2-2:32:46
dtLastModificationTime	DATE_AND_TIME		DT#1970-1-2-2:32:46
astFileInfoList[3]	FileInfo		
strFileName	STRING[255]		'test1.txt'
xDirectory	BOOL		FALSE
uliFileSize	ULINT		0
dtCreationTime	DATE_AND_TIME		DT#1970-1-2-2:32:45
dtLastAccessTime	DATE_AND_TIME		DT#1970-1-2-2:32:45
dtLastModificationTime	DATE_AND_TIME		DT#1970-1-2-2:32:45
astFileInfoList[4]	FileInfo		
astFileInfoList[5]	FileInfo		
tTimeOut	TIME		T#0ms
xDone	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
eErrorID	HOU_ERROR		HOU_NO_ERROR
xEOE	BOOL		TRUE
udiFindFiles	UDINT		3

- **Precautions**
  - Execution of this instruction is continued until completion even if the value of xExecute changes to FALSE or the execution time exceeds the task cycle. Use the value of xDone to determine whether the instruction is completed, which becomes TRUE upon completion.
  - To search for the directory specified by strPathName, the user must have the access to the directory and its subdirectories.
  - The directory name strPathName must be in English and case-sensitive. Any character except wildcards (* and ?) is treated as valid data. Wildcards are only supported at the beginning or end of the filename, such as A*, *A, or *A*.

- When the found file name specified by strPathName is invalid, the value of xError is TRUE.
- The file name specified by strPathName contains the path separators (/), and each directory contains two entries "." and "..", respectively indicating the current directory and the parent directory.
- xEOE indicates whether all similar files in the specified directory are found. The function block may need to be triggered multiple times to find all similar files in the specified folder. The search is completed only when xEOE is TRUE or when the search is prematurely terminated by setting eMode to LISTCMD_ABORT. If the required file or directory is found in the program, xEOE is FALSE. In this case, the value of the input parameter eCmd must be changed to LISTCMD_ABORT and the function block must be triggered again to release all internal resources.
- bEOE turns to TRUE when the first attempt is made to read a non-existent entry.
- tTimeOut specifies the timeout period for execution of the function block. When it is set to T#0S, the timeout function is not enabled.
- A new search can begin only after the search completion signal xDone is set.

- **Fault ID**

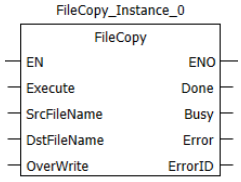
eErrorID	Value	Note
HOU_NO_ERROR	0	No fault.
HOU_INTERNAL_ERROR	1	General internal fault
HOU_INVALIDPARAMETER_ERROR	2	Invalid instruction parameters
HOU_NOTINITIALIZED_ERROR	3	Component not initialized, failure to execute the instruction
HOU_VERSION_ERROR	4	Version conflict
HOU_TIMEOUT_ERROR	5	Instruction timeout
HOU_NOBUFFER_ERROR	6	Insufficient memory, failure to execute the instruction
HOU_PENDING_ERROR	10	Instruction calling not completed
HOU_NUMPENDING_ERROR	11	Excessive instruction calls
HOU_NOTIMPLEMENTED_ERROR	12	Instruction not implemented
HOU_INVALIDID_ERROR	13	Invalid ID input
HOU_OVERFLOW_ERROR	14	Integer overflow
HOU_BUFFER_SIZE_ERROR	15	Buffer too small or invalid
HOU_NOOBJECT_ERROR	16	Input named object not existing
HOU_NOMEMORY_ERROR	17	No heap memory available
HOU_DUPLICATE_ERROR	18	Another object with the same name existing

eErrorID	Value	Note
HOU_MEMORYOVERWRITE_ERROR	19	Heap memory write out of bounds, memory overwrite error
HOU_INVALIDHANDLE_ERROR	20	Invalid object handle
HOU_ENDOFOBJECT_ERROR	21	Object end reached
HOU_NOCHANGE_ERROR	22	No change
HOU_INVALIDINTERFACE_ERROR	23	Invalid or unknown interface
HOU_NOTSUPPORTED_ERROR	24	Function not supported
HOU_NOACCESSRIGHTS_ERROR	25	No access permission
HOU_OUTOFLIMITS_ERROR	26	Resource limit exceeded
HOU_ENTRIESREMAINING_ERROR	27	Buffer limit causing transmission incompletion
HOU_INVALIDSESSIONID_ERROR	28	Invalid online session ID
HOU_EXCEPTION_ERROR	29	Exception
HOU_INVALIDPATHNAME_ERROR	33	Path name invalid
HOU_INVALIDFILENAME_ERROR	34	File name invalid
HOU_INVALIDLISTMODE_ERROR	35	Invalid control command
HOU_FILEOVERWRITE_ERROR	50	File overwrite fault
HOU_OVERBUFFERSIZE_ERROR	1030	Data area too small
HOU_SAMEFILENAME_ERROR	1403	File with the same name existing

18.2.11.8 FileCopy

Copies the specified file.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FileCopy	Copy file	FB		FileCopy_Instance_0(Execute := , SrcFileName := , DstFileName := , OverWrite := , Done => , Busy =>, Error =>, ErrorID =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Enable	Follows data type	FALSE	Edge-triggered switch to execute the function block
	SrcFileName	Name of source file to copy	Max. 200 characters	''	Name of source file to copy
	DstFileName	Name of destination file to paste	Max. 200 characters	''	Name of destination file to paste
	OverWrite	Overwrite enable	Follows data type	FALSE	TRUE: Enable overwrite FALSE: Prohibit overwrite
Output	Done	Function block completed state	Follows data type	-	Function block completed state
	Busy	Function block execution flag	Follows data type	-	Function block execution flag
	Error	Function block fault flag	Follows data type	-	Function block fault flag
	ErrorID	Function block fault ID	Follows data type	-	Function block fault ID

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SrcFile Name	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
DstFile Name	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
OverWrite	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

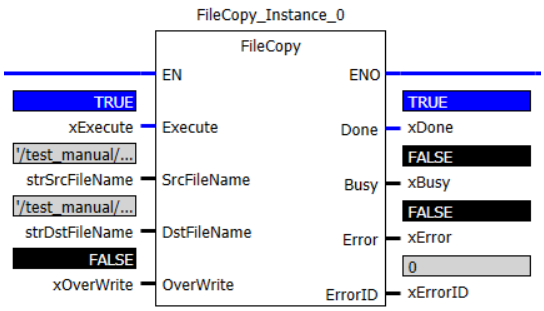
- **Function**
  - This instruction copies the source file specified by SrcFileName to DstFileName.
  - If there is no file with the same name as DstFileName, a new file is created.
  - If a file with the same name as DstFileName already exists, the following is performed depending on the value of OverWrite, as shown below.

Value of OverWrite	Operation
TRUE	The existing file is overwritten
FALSE	The existing file is not overwritten and a fault error occurs

- **Program example**

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    FileCopy_Instance_0: FileCopy;
    xExecute: BOOL;
    strSrcFileName: STRING[242] := '/test_manual/test1.txt';
    strDstFileName: STRING[242] := '/test_manual/test1_copy.txt';
    xOverWrite: BOOL := FALSE;
    xDone: BOOL;
    xBusy: BOOL;
    xError: BOOL;
    xErrorID: WORD;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>


</td><td>

```
FileCopy_Instance_0(
Execute := xExecute,
SrcFileName := strSrcFileName,
DstFileName := strDstFileName,
OverWrite := xOverWrite,
Done => xDone,
Busy => xBusy,
Error => xError,
ErrorID => xErrorID);RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

	Name	Data type	Address	Value
1	FileCopy_Instance_0	FileCopy		
2	xExecute	BOOL		TRUE
3	strSrcFileName	STRING[242]		'/test_manual/test1.txt'
4	strDstFileName	STRING[242]		'/test_manual/test1_copy.txt'
5	xOverWrite	BOOL		FALSE
6	xDone	BOOL		TRUE
7	xBusy	BOOL		FALSE
8	xError	BOOL		FALSE
9	xErrorID	WORD		0

</td></tr>
</table>

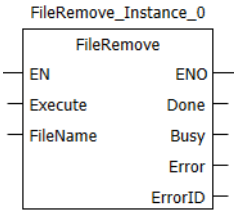
- **Precautions**
  - Before using this instruction, please read the 18.2.11.2 Overview to avoid improper operations.
  - If the SD card is removed while a file is open, the file remains in an open state. However, when the SD card is reinserted, the file cannot be read from or written to. To read and write the file, re-open the file.
  - If there is insufficient space available on the SD card, a new file can be created, but the data in the copied file will be lost, compared to the original.
  - An exception occurs and Error becomes TRUE in the following cases:
    - If the SD card is write-protected, ErrorID is 1.
    - If DstFileName is empty and OverWrite is FALSE, ErrorID is 1403; if DstFileName is empty and OverWrite is TRUE, ErrorID is 1.
    - If SrcFileName is empty, ErrorID is 1.
    - If the file specified by SrcFileName does not exist, ErrorID is 1403.
    - If SrcFileName is an invalid file name, ErrorID is 1.

- If DstFileName is an invalid file name and OverWrite is FALSE, ErrorID is 1403; if DstFileName is an invalid file name and OverWrite is TRUE, ErrorID is 1.
- If a file with the name specified by DstFileName exits and OverWrite is FALSE, ErrorID is 1403.
- If the SD card is not successfully mounted, data stored on the card cannot be accessed, and ErrorID is 2.
- If multiple threads execute the FileCopy instruction simultaneously, execution is queued (threads themselves are not blocked). If the operation does not complete within 10 task cycles, ErrorID is 1540.
- If the FileCopy instruction is executed after user directory size initialization fails, ErrorID is 1543.

18.2.11.9 FileRemove

Deletes the specified file.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FileRemove	Delete file	FB		FileRemove_Instance_0(Execute := , FileName :=, Done =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Enable	Follows data type	FALSE	(Triggered by edges) Execution of the function block switch
	FileName	Specified file name	Max. 200 characters	''	Name of the file to delete
Output	Done	Function block completed state	Follows data type	-	Function block completed state
	Busy	Function block execution flag	Follows data type	-	Function block execution flag

Input/Output	Variable	Name	Valid Range	Default Value	Description
	Error	Function block fault flag	Follows data type	-	Function block fault flag
	ErrorID	Function block fault ID	Follows data type	-	Function block fault ID

Variable types

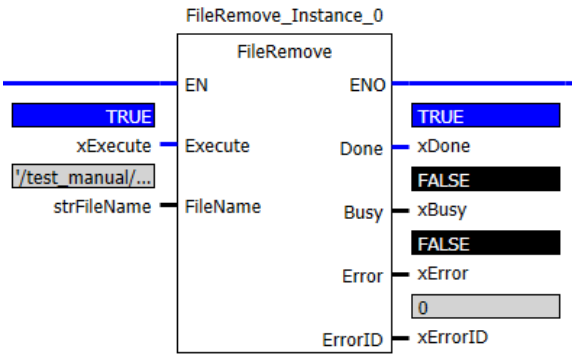
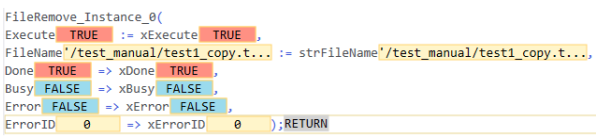
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FileName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction deletes the file specified by FileName.

- **Program example**

Item	LD	ST
Define variables	VAR FileRemove_Instance_0: FileRemove; xExecute: BOOL; strFileName: STRING[242] := '/test_manual/test1_copy.txt'; xDone: BOOL; xBusy: BOOL; xError: BOOL; xErrorID: WORD; END_VAR	

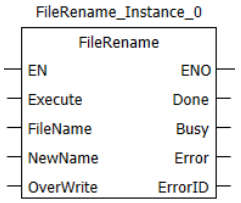
<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>
</td><td>
</td></tr>
<tr><td>Operation result</td><td colspan="2">
<table>
<tr><th></th><th></th><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td>1</td><td></td><td>FileRemove_Instance_0</td><td>FileRemove</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td>xExecute</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>3</td><td></td><td>strFileName</td><td>STRING[242]</td><td></td><td>'/test_manual/test1_copy.txt'</td></tr>
<tr><td>4</td><td></td><td>xDone</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>5</td><td></td><td>xBusy</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>6</td><td></td><td>xError</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>7</td><td></td><td>xErrorID</td><td>WORD</td><td></td><td>0</td></tr>
</table>
</td></tr>
</table>

- **Precautions**
  - Before using this instruction, please read the 18.2.11.2 Overview to avoid improper operations.
  - If the SD card is removed while a file is open, the file remains in an open state. However, when the SD card is reinserted, the file cannot be read from or written to. To read and write the file, re-open the file.
  - An exception occurs and Error becomes TRUE in the following cases:
    - If the SD card is write-protected, ErrorID is 1.
    - If FileName is an empty name, ErrorID is 1.
    - If the file specified by FileName does not exist, ErrorID is 16.
    - If the SD card is not successfully mounted, data stored on the card cannot be accessed, and ErrorID is 2.

18.2.11.10 FileRename

Renames of the specified file or directory.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FileRename	Change file name	FB		FileRename_Instance_0(Execute := , FileName :=, NewName := , OverWrite := , Done => , Busy =>, Error =>, ErrorID =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Enable	Follows data type	FALSE	(Triggered by edges) Execution of the function block switch
	FileName	Original file name	Max. 200 characters	''	Original file name
	NewName	New file name	Max. 200 characters	''	New file name
	OverWrite	Overwrite enable	Follows data type	FALSE	TRUE: Enable overwrite FALSE: Prohibit overwrite
Output	Done	Function block completed state	Follows data type	-	Function block completed state
	Busy	Function block execution flag	Follows data type	-	Function block execution flag
	Error	Function block fault flag	Follows data type	-	Function block fault flag
	ErrorID	Function block fault ID	Follows data type	-	Function block fault ID

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FileName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
NewName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
OverWrite	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

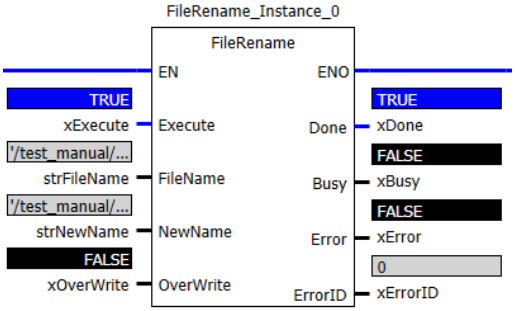
- **Function**
  - This instruction changes the original name of the file or directory specified by FileName to new file name specified by NewName.
  - If there is no file with the same name as NewName, a new file is created.
  - If a file with the same name as NewName already exists, the following is performed depending on the value of OverWrite, as shown below.

Value of OverWrite	Operation
TRUE	The file or directory is overwritten.
FALSE	The existing file is not overwritten and a fault error occurs.

- **Program example**

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    FileRename_Instance_0: FileRename;
    xExecute: BOOL;
    strFileName: STRING[242] := '/test_manual/test1.txt';
    strNewName: STRING[242] := '/test_manual/test1_rename.txt';
    xOverWrite: BOOL;
    xDone: BOOL;
    xBusy: BOOL;
    xError: BOOL;
    xErrorID: WORD;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>
</td><td>

```
FileRename_Instance_0(
Execute TRUE := xExecute TRUE,
FileName'/test_manual/test1.t... := strFileName'/test_manual/test1.t...,
NewName'/test_manual/test1_rename.t... := strNewName'/test_manual/test1_rename.t...,
OverWrite FALSE := xOverWrite FALSE,
Done TRUE => xDone TRUE,
Busy FALSE => xBusy FALSE,
Error FALSE => xError FALSE,
ErrorID 0 => xErrorID 0 );RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

	Name	Data type	Address	Value
1	FileRename_Instance_0	FileRename		
2	xExecute	BOOL		TRUE
3	strFileName	STRING[242]		'/test_manual/test1.txt'
4	strNewName	STRING[242]		'/test_manual/test1_rename.txt'
5	xOverWrite	BOOL		FALSE
6	xDone	BOOL		TRUE
7	xBusy	BOOL		FALSE
8	xError	BOOL		FALSE
9	xErrorID	WORD		0

</td></tr>
</table>

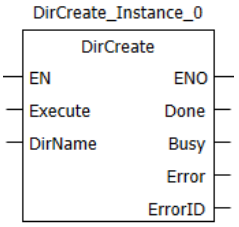
- **Precautions**
  - Before using this instruction, please read the 18.2.11.2 Overview to avoid improper operations.
  - If the directories are different for FileName and NewName, the file is moved to the directory specified by NewName. If the SD card is removed while a file is open, the file remains in an open state. However, when the SD card is reinserted, the file cannot be read from or written to. To read and write the file, re-open the file.
  - If a read-only file with the name NewName already exists and OverWrite is TRUE, it may cause a PLC exception.
  - An exception occurs and Error becomes TRUE in the following cases:
    - If the SD card is write-protected, ErrorID is 1.
    - If NewName is empty, ErrorID is 1403.
    - If FileName is empty or an invalid filename, ErrorID is 1.
    - If the file specified by FileName does not exist, ErrorID is 1403.

- If NewName is an invalid filename, ErrorID is 1403, which may cause a PLC exception.
- If a file with the name NewName already exits and OverWrite is FALSE, ErrorID is 1403.
- If NewName is the same as FileName and OverWrite is TRUE, ErrorID is 1403.
- If the directory or file specified by FileName and NewName both exist but are of different types, ErrorID is 1.
- If NewName contains both a different folder and a file compared to FileName, and the target folder does not exist, ErrorID is 1. For example, when FileName is "folder1/folder2/file01" and NewName is "folder1/folder3/file01_rename", but folder3 does not exist, the instruction fails to create a new folder and a new file simultaneously.
- If the SD card is not successfully mounted, data stored on the card cannot be accessed, and ErrorID is 2.

18.2.11.11 DirCreate

Creates a directory with the specified name.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DirCreate	Create directory	FB		DirCreate_Instance_0(Execute := , DirName := , Done => , Busy =>, Error =>, ErrorID =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Enable	Follows data type	FALSE	(Triggered by edges) Execution of the function block switch
	DirName	Specified directory name	Max. 200 characters	''	Name of the directory to create
Output	Done	Function block completed state	Follows data type	-	Function block completed state

Input/Output	Variable	Name	Valid Range	Default Value	Description
	Busy	Function block execution flag	Follows data type	-	Function block execution flag
	Error	Function block fault flag	Follows data type	-	Function block fault flag
	ErrorID	Function block fault ID	Follows data type	-	Function block fault ID

Variable types

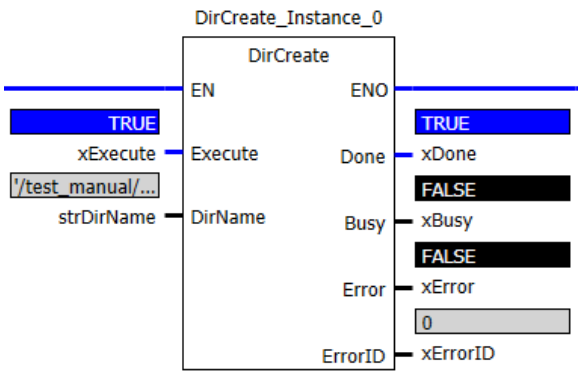
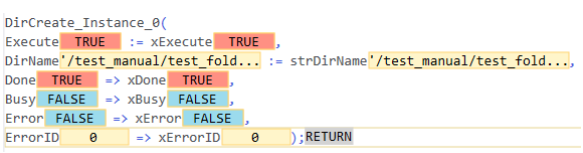
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
DirName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Creates a directory with the name specified by DirName.

- **Program example**

Item	LD	ST
Define variables		VAR DirCreate_Instance_0: DirCreate; xExecute: BOOL; strDirName: STRING[242] := '/test_manual/test_folder'; xDone: BOOL; xBusy: BOOL; xError: BOOL; xErrorID: WORD; END_VAR

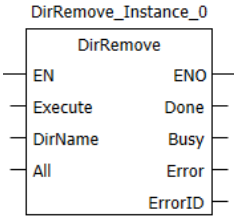
<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>
</td><td>
</td></tr>
<tr><td>Operation result</td><td colspan="2">
<table>
<tr><th></th><th></th><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td>1</td><td></td><td>DirCreate_Instance_0</td><td>DirCreate</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td>xExecute</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>3</td><td></td><td>strDirName</td><td>STRING[242]</td><td></td><td>'/test_manual/test_folder'</td></tr>
<tr><td>4</td><td></td><td>xDone</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>5</td><td></td><td>xBusy</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>6</td><td></td><td>xError</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>7</td><td></td><td>xErrorID</td><td>WORD</td><td></td><td>0</td></tr>
</table>
</td></tr>
</table>

- **Precautions**
  - Before using this instruction, please read the 18.2.11.2 Overview to avoid improper operations.
  - If the SD card is removed while a file is open, the file remains in an open state. However, when the SD card is reinserted, the file cannot be read from or written to. To read and write the file, re-open the file.
  - An exception occurs and Error becomes TRUE in the following cases:
    - If the SD card is write-protected, ErrorID is 1.
    - If there is insufficient space available on the SD card, ErrorID is 1.
    - If the directory specified by DirName already exists, ErrorID is 18.
    - If DirName is an invalid directory name, ErrorID is 18.
    - If the SD card is not successfully mounted, data stored on the card cannot be accessed, and ErrorID is 2.

18.2.11.12 DirRemove

Deletes the specified directory.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DirRemove	Delete directory	FB		DirRemove_Instance_0(Execute := , DirName := , All := , Done => , Busy =>, Error =>, ErrorID =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Enable	Follows data type	FALSE	(Triggered by edges) Execution of the function block switch
	DirName	Directory to delete	Max. 200 characters	''	Directory to delete
	All	Select all	Follows data type	FALSE	Whether to delete a directory that contains files and subdirectories TRUE: Delete. FALSE: Do not delete.
Output	Done	Function block completed state	Follows data type	-	Function block completed state
	Busy	Function block execution flag	Follows data type	-	Function block execution flag
	Error	Function block fault flag	Follows data type	-	Function block fault flag
	ErrorID	Function block fault ID	Follows data type	-	Function block fault ID

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
DirName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
All	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

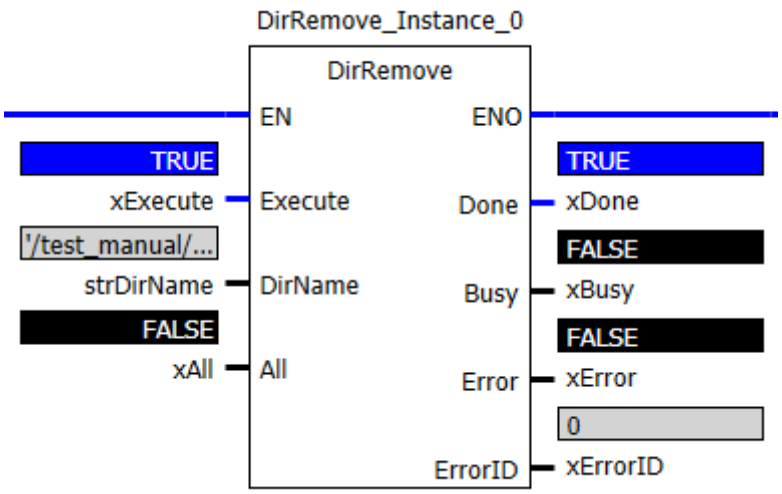
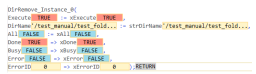
- **Function**

  This instruction deletes the directory specified by DirName. If there are files or subdirectories in the specified directory, the following is performed depending on the value of All.

Value of All	Operation
TRUE	The specified directory is deleted, including all files and subdirectories in it.
FALSE	The specified directory is not deleted and a fault occurs.

- **Program example**

Item	LD	ST
Define variables	VAR DirRemove_Instance_0: DirRemove; xExecute: BOOL; strDirName: STRING[242] := '/test_manual/test_folder'; xAll: BOOL := FALSE; xDone: BOOL; xBusy: BOOL; xError: BOOL; xErrorID: WORD; END_VAR	

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>
</td><td></td></tr>
<tr><td>Operation result</td><td>
<table>
<tr><th></th><th></th><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td>1</td><td></td><td>DirRemove_Instance_0</td><td>DirRemove</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td>xExecute</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>3</td><td></td><td>strDirName</td><td>STRING[242]</td><td></td><td>'/test_manual/test_folder'</td></tr>
<tr><td>4</td><td></td><td>xAll</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>5</td><td></td><td>xDone</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>6</td><td></td><td>xBusy</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>7</td><td></td><td>xError</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>8</td><td></td><td>xErrorID</td><td>WORD</td><td></td><td>0</td></tr>
</table>
</td></tr>
</table>

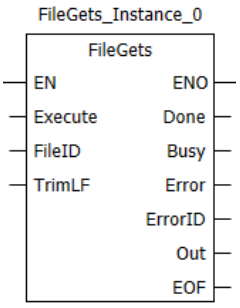
- **Precautions**
  - Before using this instruction, please read the 18.2.11.2 Overview to avoid improper operations.
  - If the SD card is removed while a file is open, the file remains in an open state. However, when the SD card is reinserted, the file cannot be read from or written to. To read and write the file, re-open the file.
  - An exception occurs and Error becomes TRUE in the following cases:
    - If the SD card is write-protected, ErrorID is 1.
    - If the value of All is FALSE and the directory specified by DirName contains a file or subdirectory, ErrorID is 16.
    - If the file specified by DirName does not exist or is an invalid file name, ErrorID is 16.
    - If DirName is an empty name, ErrorID is 16.
    - If the SD card is not successfully mounted, data stored on the card cannot be accessed, and ErrorID is 2.

18.2.11.13 File Read/Write (String Supported)

18.2.11.13.1 FileGets

Reads a string of one line from the specified file.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FileGets	Get text string	FB		FileGets_Instance_0(Execute := , FileID := , TrimLF := , Done => , Busy =>, Error =>, ErrorID => , Out => , EOF =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Enable	Follows data type	FALSE	(Triggered by edges) Execution of the function block switch
	FileID	File ID	Follows data type	0	ID of the file to write
	TrimLF	Line feed designation	Follows data type	FALSE	Handling of the line feed code of the read text string TRUE: Delete. FALSE: Do not delete.
Output	Done	Function block completed state	Follows data type	-	Function block completed state
	Busy	Function block execution flag	Follows data type	-	Function block execution flag
	Error	Function block fault flag	Follows data type	-	Function block fault flag
	ErrorID	Function block fault ID	Follows data type	-	Function block fault ID

Input/Output	Variable	Name	Valid Range	Default Value	Description
	Out	Read text string	Maximum 1985 bytes	-	Read text string
	EOF	End of file	Follows data type	-	Whether the end of the file is reached TRUE: Reached. FALSE: Not reached.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FileID	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
TrimLF	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
EOF	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction reads a string of one line from the file position indicator in the file specified by FileID.
  - The file position indicator is set at the desired location in advance with the FileSeek instruction.
  - Line endings are determined by a line feed code. The text string that is read is written to read the text string Out.

- Line endings are automatically identified from three types: CR (\r), LF (\n), and CR + LF (\r\n). If line feed designation TrimLF is TRUE, the line feed code is deleted from the text string before it is written to Out.
- If data is read to the end of the file, EOF (end of file) changes to TRUE.

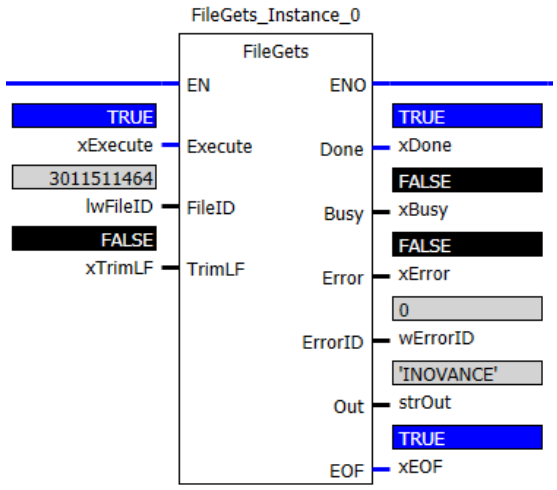
- **Program example**

  Note: The ID corresponding to the variable lwFileID is generated by the FileOpen instruction.

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    FileOpen_Instance_0: FileOpen;
    sFileName: STRING[242] := '/test_manual/fileopen.txt';
    FileGets_Instance_0: FileGets;
    xExecute: BOOL;
    lwFileID: LWORD;
    xTrimLF: BOOL;
    xDone: BOOL;
    xBusy: BOOL;
    xError: BOOL;
    wErrorID: WORD;
    strOut: STRING[255];
    xEOF: BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
FileGets_Instance_0(
Execute TRUE := xExecute TRUE ,
FileID 3011511464 := lwFileID 3011511464 ,
TrimLF FALSE := xTrimLF FALSE ,
Done TRUE => xDone TRUE ,
Busy FALSE => xBusy FALSE ,
Error FALSE => xError FALSE ,
ErrorID 0 => wErrorID 0 ,
Out 'INOVANCE' => strOut 'INOVANCE' ,
EOF TRUE => xEOF TRUE );RETURN
```

</td></tr>
</table>

Item	LD	ST

Operation result

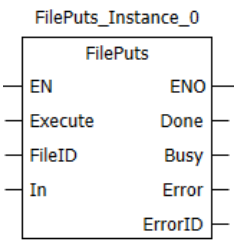
Name	Data type	Address	Value
FileOpen_Instance_0	FileOpen		
sFileName	STRING[242]		'/test_manual/fileopen.txt'
FileGets_Instance_0	FileGets		
xExecute	BOOL		TRUE
lwFileID	LWORD		3011511464
xTrimLF	BOOL		FALSE
xDone	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
wErrorID	WORD		0
strOut	STRING[255]		'INOVANCE'
xEOF	BOOL		TRUE

- **Precautions**
  - Before using this instruction, please read the 18.2.11.2 Overview to avoid improper operations.
  - If the length of the one-line text string exceeds 1985 bytes, the first 1985 bytes of the text string and the NULL text string at the end are stored in Out. Execute the FileOpen instruction in advance to obtain the value for FileID.
  - An exception occurs under the following conditions. The value of Error is TRUE.
    - If the file specified by FileID does not exist, ErrorID is 1.
    - If the file specified by FileID is not opened in a reading mode, ErrorID is 1.
    - If the input FileID is invalid (less than 0 or greater than 300), ErrorID is 1537.
    - If a file opened by the FileOpen instruction is used by FileGets after already being closed by FileClose, ErrorID is 1538.

18.2.11.13.2 FilePuts

Writes a string to the specified file.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FilePuts	Put text string	FB		FilePuts_Instance_0(Execute := , FileID := , In := , Done => , Busy =>, Error =>, ErrorID =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Enable	Follows data type	FALSE	(Triggered by edges) Execution of the function block switch
	FileID	File ID	Follows data type	0	ID of the file to write
	In	Text string to write	Max. 1985 characters	''	Text string to write
Output	Done	Function block completed state	Follows data type	-	Function block completed state
	Busy	Function block execution flag	Follows data type	-	Function block execution flag
	Error	Function block fault flag	Follows data type	-	Function block fault flag
	ErrorID	Function block fault ID	Follows data type	-	Function block fault ID

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FileID	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

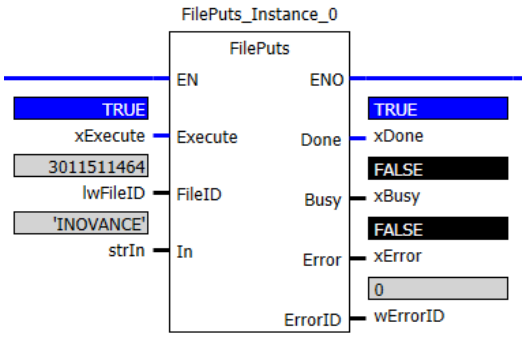
This instruction writes the text string In to the position of the file position indicator in the file specified by FileID. The file position indicator is set at the desired location in advance with the FileSeek instruction.

- **Program example**

  Note: The ID corresponding to the variable lwFileID is generated by the FileOpen instruction.

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    FileOpen_Instance_0: FileOpen;
    sFileName: STRING[242] := '/test_manual/fileopen.txt';
    FilePuts_Instance_0: FilePuts;
    xExecute: BOOL;
    lwFileID: LWORD;
    strIn: STRING[255] := 'INOVANCE';
    xDone: BOOL;
    xBusy: BOOL;
    xError: BOOL;
    wErrorID: WORD;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>
</td><td>

```
FilePuts_Instance_0(
Execute  TRUE   := xExecute  TRUE  ,
FileID      3011511464      := lwFileID      3011511464      ,
In    'INOVANCE'    := strIn    'INOVANCE'    ,
Done  TRUE   => xDone  TRUE  ,
Busy  FALSE  => xBusy  FALSE ,
Error FALSE  => xError FALSE ,
ErrorID   0     => wErrorID   0     );RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

Name	Data type	Address	Value
FileOpen_Instance_0	FileOpen		
sFileName	STRING[242]		'/test_manual/fileopen.txt'
FilePuts_Instance_0	FilePuts		
xExecute	BOOL		TRUE
lwFileID	LWORD		3011511464
strIn	STRING[255]		'INOVANCE'
xDone	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
wErrorID	WORD		0

</td></tr>
</table>

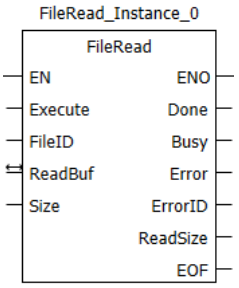
- **Precautions**
  - Before using this instruction, please read the 18.2.11.2 Overview to avoid improper operations.
  - Execute the FileOpen instruction in advance to obtain the value for FileID.
  - An exception occurs and Error becomes TRUE in the following cases:
    - If there is insufficient space available on the SD card, ErrorID is 41.
    - If the file specified by FileID does not exist, ErrorID is 1.
    - If In is empty, ErrorID is 2.
    - If the file specified by FileID is not opened in a writing mode, ErrorID is 1.

- If the input FileID is invalid (less than 0 or greater than 300), ErrorID is 1537.
- If a file opened by the FileOpen instruction is used by FilePuts after already being closed by FileClose, ErrorID is 1538.
- If multiple threads execute the DirRemove instruction simultaneously, execution is queued (threads themselves are not blocked). If the operation does not complete within 10 task cycles, ErrorID is 1540.
- If the FilePuts instruction is executed after user directory size initialization fails, ErrorID is 1543.

18.2.11.13.3 FileRead

Reads data from the specified file.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FileRead	Read file	FB		FileRead_Instance_0(Execute := , FileID := , ReadBuf := , Size := , Done => , Busy =>, Error =>, ErrorID => , ReadSize => , EOF =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Enable	Follows data type	FALSE	(Triggered by edges) Execution of the function block switch
	FileID	File ID	Follows data type	0	ID of the file to read
	Size	Number of elements to read	Follows data type	1	Number of elements to read

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input/Output	ReadBuf[ ] array	Read buffer	Follows data type	-	Buffer in which to write data that was read
Output	Done	Function block completed state	Follows data type	-	Function block completed state
	Busy	Function block execution flag	Follows data type	-	Function block execution flag
	Error	Function block fault flag	Follows data type	-	Function block fault flag
	ErrorID	Function block fault ID	Follows data type	-	Function block fault ID
	ReadSize	Number of elements actually read	Follows data type	-	Number of elements actually read
	EOF	End of file	Follows data type	-	Whether the end of the file is reached TRUE: Reached. FALSE: Not reached.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FileID	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ReadBuf[ ] array	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Array of enumerations or structures can also be specified.																			
Size	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

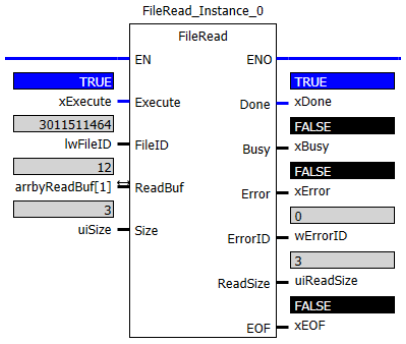
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Read Size	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
EOF	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction reads data from the file position indicator in the file specified by FileID, and then stores the data in read buffer ReadBuf[ ].
  - The file position indicator is set at the desired location in advance with the FileSeek instruction.
  - The amount of data that is read is the size of the data type of ReadBuf[ ] times Size.
  - The number of elements actually read is stored in ReadSize. In general, values of Size and ReadSize are the same. If the amount of data from the file position indicator to the end of the file is less than the value of Size, no error occurs and the data to the end of the file is stored in ReadBuf[ ]. If that occurs, the value of ReadSize is less than the value of Size.
  - When reading to the end of the file, EOF becomes TRUE; if not reading to the end, EOF is FALSE.
- **Program example**

  Note: The ID corresponding to the variable lwFileID is generated by the FileOpen instruction.

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    FileOpen_Instance_0: FileOpen;
    sFileName: STRING[242] := '/test_manual/fileopen.txt';
    FileRead_Instance_0: FileRead;
    xExecute: BOOL;
    lwFileID: lWORD;
    arrbyReadBuf: ARRAY[1..5] OF BYTE;
    uiSize: UINT := 3;
    xDone: BOOL;
    xBusy: BOOL;
    xError: BOOL;
    wErrorID:WORD;
    uiReadSize: UINT;
    xEOF: BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>
</td><td>

```
FileRead_Instance_0(
Execute  TRUE  := xExecute  TRUE ,
FileID  3011511464  := lwFileID  3011511464 ,
ReadBuf := arrbyReadBuf[1]  12 ,
Size  3  := uiSize  3 ,
Done  TRUE  => xDone  TRUE ,
Busy  FALSE  => xBusy  FALSE ,
Error  FALSE  => xError  FALSE ,
ErrorID  0  => wErrorID  0 ,
ReadSize  3  => uiReadSize  3 ,
EOF  FALSE  => xEOF  FALSE );RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

Name	Data type	Address	Value
FileOpen_Instance_0	FileOpen		
sFileName	STRING[242]		'/test_manual/fileopen.txt'
FileRead_Instance_0	FileRead		
xExecute	BOOL		TRUE
lwFileID	LWORD		3011511464
arrbyReadBuf	ARRAY [1..5] OF BYTE		
arrbyReadBuf[1]	BYTE		12
arrbyReadBuf[2]	BYTE		34
arrbyReadBuf[3]	BYTE		5
arrbyReadBuf[4]	BYTE		0
arrbyReadBuf[5]	BYTE		0
uiSize	UINT		3
xDone	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
wErrorID	WORD		0
uiReadSize	UINT		3
xEOF	BOOL		FALSE

</td></tr>
</table>

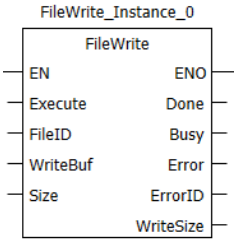
- **Precautions**
  - Before using this instruction, please read the 18.2.11.2 Overview to avoid improper operations.
  - If the data is read to the end of the file and the size of the data is not evenly divisible by the size of the data type of ReadBuf[ ], the data that is insufficient for the data size of ReadBuf[ ] is discarded. The file position indicator advances to the end of the file, and the value of EOF changes to TRUE.
  - If the value of Size exceeds the ReadBuf[ ] array, a program fault occurs, Error becomes TRUE, and ErrorID is 1030. The elements in ReadBuf[ ] retain their values from before executing this instruction.
  - Execute the FileOpen instruction in advance to obtain the value for FileID.

- If ReadBuf[ ] is an array of structures, adjustment areas between members may be inserted depending on the composition.
- An exception occurs and Error becomes TRUE in the following cases:
  - If the file specified by FileID does not exist, ErrorID is 1.
  - If Size is not 0 and the file specified by FileID is not opened in a reading mode, ErrorID is 1.
  - If the input FileID is invalid (less than 0 or greater than 300), ErrorID is 1537.
  - If a file opened by the FileOpen instruction is used by FileRead after already being closed by FileClose, ErrorID is 1538.

18.2.11.13.4 FileWrite

Writes the data to the specified file.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FileWrite	Write file	FB		FileWrite_Instance_0(Execute := , FileID := , WriteBuf := , Size := , Done => , Busy =>, Error =>, ErrorID => , WriteSize =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Enable	Follows data type	FALSE	(Triggered by edges) Execution of the function block switch
	FileID	File ID	Follows data type	0	ID of the file to write
	WriteBuf[ ] array	Write buffer	Follows data type	-	Data to write

Input/Output	Variable	Name	Valid Range	Default Value	Description
	Size	Number of elements to write	Follows data type	1	Number of elements to write
Output	Done	Function block completed state	Follows data type	-	Function block completed state
	Busy	Function block execution flag	Follows data type	-	Function block execution flag
	Error	Function block fault flag	Follows data type	-	Function block fault flag
	ErrorID	Function block fault ID	Follows data type	-	Function block fault ID
	WriteSize	Number of elements actually written	Follows data type	-	Number of elements actually written

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FileID	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
WriteBuf[ ] array	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Array of enumerations or structures can also be specified.																			
Size	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

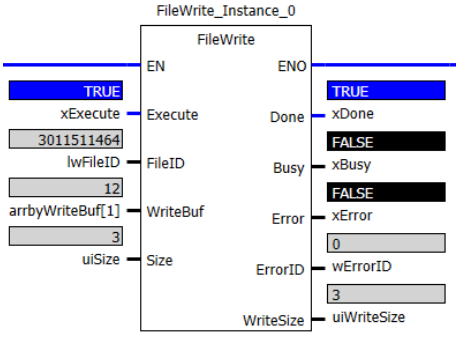
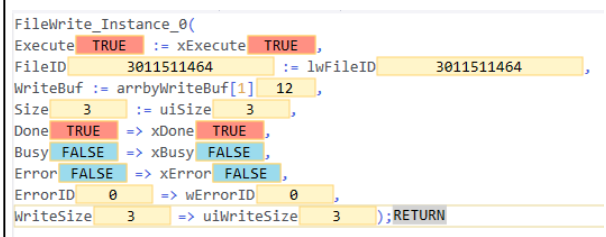
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Write Size	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

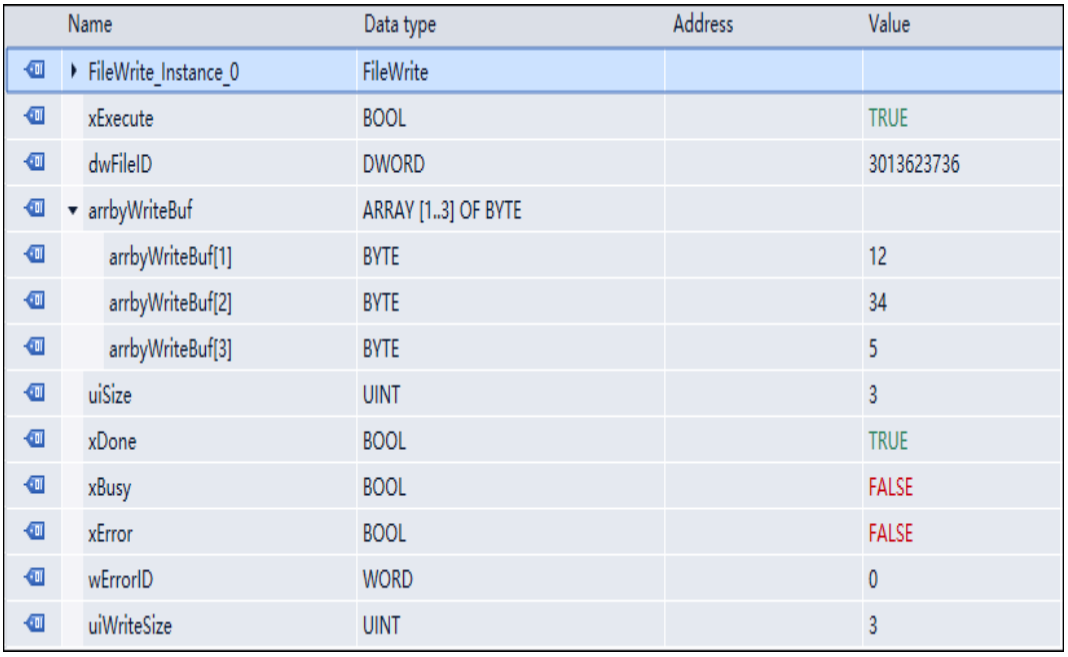
- **Function**
  - This instruction writes the data written to WriteBuf[ ] to the file position indicator in the file specified by FileID.
  - The file position indicator is set at the desired location in advance with the FileSeek instruction.
  - The amount of data that is written is the size of the data type of WriteBuf[ ] times Size.
  - The number of elements actually written is saved to WriteSize. In general, values of Size and ReadSize are the same.
- **Program example**

  Note: The ID corresponding to the variable lwFileID is generated by the FileOpen instruction.

Item	LD	ST

Define variables	VAR
	FileOpen_Instance_0: FileOpen;
	sFileName: STRING[242] := '/test_manual/fileopen.txt';
	FileWrite_Instance_0: FileWrite;
	xExecute: BOOL;
	lwFileID: lWORD;
	arrbyWriteBuf: ARRAY[1..3] OF BYTE := [12, 34, 5];
	uiSize: UINT:= 3;
	xDone: BOOL;
	xBusy: BOOL;
	xError: BOOL;
	wErrorID: WORD;
	uiWriteSize: UINT;
	END_VAR	

Program		

Item	LD	ST
Operation result		

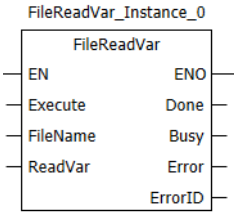
Name	Data type	Address	Value
FileWrite_Instance_0	FileWrite		
xExecute	BOOL		TRUE
dwFileID	DWORD		3013623736
arrbyWriteBuf	ARRAY [1..3] OF BYTE		
arrbyWriteBuf[1]	BYTE		12
arrbyWriteBuf[2]	BYTE		34
arrbyWriteBuf[3]	BYTE		5
uiSize	UINT		3
xDone	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
wErrorID	WORD		0
uiWriteSize	UINT		3

- **Precautions**
  - Before using this instruction, please read the 18.2.11.2 Overview to avoid improper operations.
  - Execute the FileOpen instruction in advance to obtain the value for FileID.
  - If WriteBuf[ ] is an array of structures, adjustment areas between members may be inserted depending on the composition.
  - An exception occurs under the following conditions. The value of Error is TRUE.
    - If the file specified by FileID does not exist, ErrorID is 1.
    - If the file specified by FileID is not opened in a writing mode, ErrorID is 1.
    - If there is insufficient space available on the SD card, ErrorID is 41.
    - If the value of Size exceeds the WriteBuf[ ] array, a program fault occurs, Error becomes TRUE, and ErrorID is 1030.
    - If the input variable WriteBuf[ ] is not passed a parameter or if an array with more than three dimensions is passed, ErrorID is 1030.
    - If the input FileID is invalid (less than 0 or greater than 300), ErrorID is 1537.
    - If a file opened by the FileOpen instruction is used by FileWrite after already being closed by FileClose, ErrorID is 1538.
    - If multiple threads execute the FileWrite instruction simultaneously, execution is queued (threads themselves are not blocked). If the operation does not complete within 10 task cycles, ErrorID is 1540.
    - If the FileWrite instruction is executed after user directory size initialization fails, ErrorID is 1543.

18.2.11.13.5 FileReadVar

Reads the values of the specified file as binary data and writes the data to a variable.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FileReadVar	File read variable	FB		FileReadVar_Instance_0(Execute := , FileName :=, ReadVar := , Done => , Busy =>, Error =>, ErrorID =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Enable	Follows data type	FALSE	(Triggered by edges) Execution of the function block switch
	FileName	Specified file name	Max. 200 characters	''	Name of the file to which to write variable data
	ReadVar	Position variable to read	Follows data type	-	Position variable from which to read variable data
Output	Done	Function block completed state	Follows data type	-	Function block completed state
	Busy	Function block execution flag	Follows data type	-	Function block execution flag
	Error	Function block fault flag	Follows data type	-	Function block fault flag
	ErrorID	Function block fault ID	Follows data type	-	Function block fault ID

Variable types

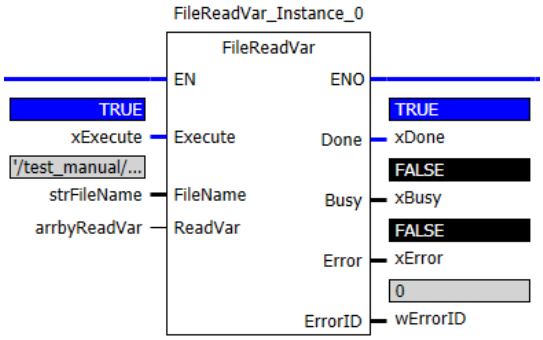
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FileName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
ReadVar	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration, an array, an array element, a structure, or a structure member can also be specified.																			
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

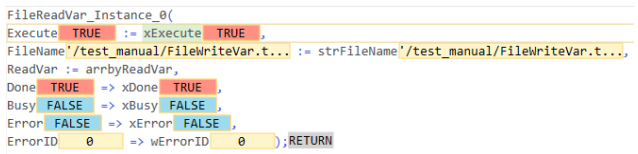
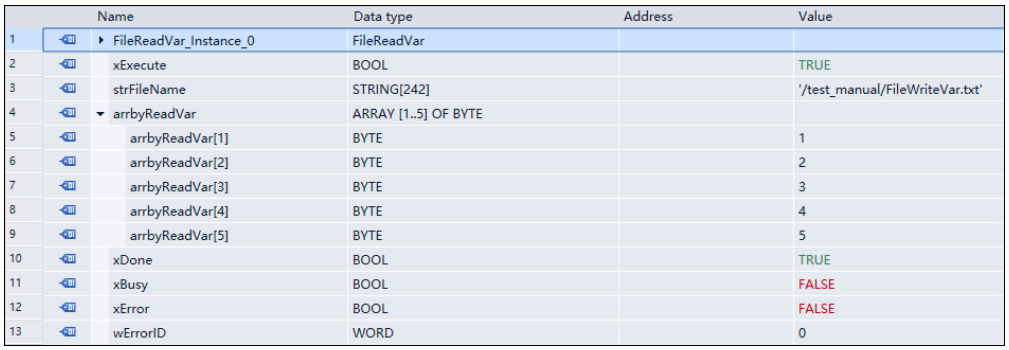
- **Function**

  This instruction reads variable data from the file specified by FileName in binary format and assigns the read data to the position variable specified by ReadVar.

- **Program example**

Item	LD	ST
Define variables	VAR FileReadVar_Instance_0: FileReadVar; xExecute: BOOL; strFileName: STRING[242] := '/test_manual/FileWriteVar.txt'; arrbyReadVar: ARRAY[1..5] OF BYTE; xDone: BOOL; xBusy: BOOL; xError: BOOL; wErrorID: WORD; END_VAR	

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>

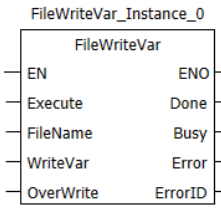

</td></tr>
<tr><td>Operation result</td><td colspan="2">
</td></tr>
</table>

- **Precautions**
  - Before using this instruction, please read the 18.2.11.2 Overview to avoid improper operations.
  - If the size of the specified file is greater than that of ReadVar, no error occurs and only the data of the size of ReadVar is read. If the size of the specified file is less than that of ReadVar, no error occurs and only the data of the size of the specified file is read. The remaining area in ReadVar retains the values used before execution of this instruction. If ReadVar is a structure, adjustment areas between members may be inserted depending on the composition.
  - An exception occurs and Error becomes TRUE in the following cases:
    - If FileName is empty or an invalid filename, ErrorID is 1.
    - If the file specified by FileName does not exist, ErrorID is 16.
    - If the SD card is not successfully mounted, data stored on the card cannot be accessed, and ErrorID is 2.
    - If multiple threads execute the FileWriteVar instruction simultaneously, execution is queued (threads themselves are not blocked). If the operation does not complete within 10 task cycles, ErrorID is 1540.
    - If the FileWriteVar instruction is executed after user directory size initialization fails, ErrorID is 1543.

18.2.11.13.6 FileWriteVar

Writes the value of a variable to the specified file in binary format.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FileWriteVar	File write variable	FB		FileWriteVar_Instance_0(Execute := , FileName :=, WriteVar := , OverWrite := , Done => , Busy =>, Error =>, ErrorID =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Enable	Follows data type	FALSE	(Triggered by edges) Execution of the function block switch
	FileName	Specified file name	Max. 200 characters	''	Name of the file to which to write variable data
	WriteVar	Specified variable	Follows data type	-	Variable to write
	OverWrite	Overwrite enable	Follows data type	FALSE	TRUE: Enable overwrite FALSE: Prohibit overwrite
Output	Done	Function block completed state	Follows data type	-	Function block completed state
	Busy	Function block execution flag	Follows data type	-	Function block execution flag
	Error	Function block fault flag	Follows data type	-	Function block fault flag
	ErrorID	Function block fault ID	Follows data type	-	Function block fault ID

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FileName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
WriteVar	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration, an array, an array element, a structure, or a structure member can also be specified.																			
OverWrite	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

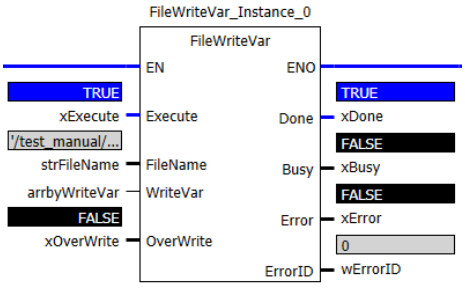
- **Function**
  - This instruction writes the value of variable WriteVar to the file specified by FileName in binary format.
  - If there is no file with the same name as FileName, a new file is created.
  - If a file with the same name as FileName already exists, the following is performed depending on the value of OverWrite, as shown below.

Value of OverWrite	Operation
TRUE	The file is overwritten.
FALSE	The file is not overwritten and an error occurs.

- **Program example**

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    FileWriteVar_Instance_0: FileWriteVar;
    xExecute: BOOL;
    strFileName: STRING[242] := '/test_manual/FileWriteVar.txt';
    arrbyWriteVar: ARRAY[1..5] OF BYTE := [1, 2, 3, 4, 5];
    xOverWrite: BOOL := FALSE;
    xDone: BOOL;
    xBusy: BOOL;
    xError: BOOL;
    wErrorID: WORD;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>


</td><td>

```
FileWriteVar_Instance_0(
Execute := xExecute,
FileName := strFileName,
WriteVar := arrbyWriteVar,
OverWrite := xOverWrite,
Done => xDone,
Busy => xBbusy,
Error => xError,
ErrorID => wErrorID);RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

	Name	Data type	Address	Value
1	FileWriteVar_Instance_0	FileWriteVar		
2	xExecute	BOOL		TRUE
3	strFileName	STRING[242]		'/test_manual/FileWriteVar.txt'
4	arrbyWriteVar	ARRAY [1..5] OF BYTE		
5	arrbyWriteVar[1]	BYTE		1
6	arrbyWriteVar[2]	BYTE		2
7	arrbyWriteVar[3]	BYTE		3
8	arrbyWriteVar[4]	BYTE		4
9	arrbyWriteVar[5]	BYTE		5
10	xOverWrite	BOOL		FALSE
11	xDone	BOOL		TRUE
12	xBusy	BOOL		FALSE
13	xError	BOOL		FALSE
14	wErrorID	WORD		0

</td></tr>
</table>

- **Notes**
  - Before using this instruction, please read the 18.2.11.2 Overview to avoid improper operations.
  - If WriteVar is a structure, adjustment areas between members may be inserted depending on the composition.
  - An exception occurs under the following conditions. The value of Error is TRUE.
    - If the SD card is write-protected, ErrorID is 1.
    - If there is insufficient space available on the SD card, ErrorID is 41.
    - If FileName is empty and OverWrite is FALSE, ErrorID is 50; if FileName is empty and OverWrite is TRUE, ErrorID is 1.
    - If FileName is an invalid file name and OverWrite is FALSE, ErrorID is 50; if FileName is an invalid file name and OverWrite is TRUE, ErrorID is 1.
    - If a file with the name specified by FileName exits and OverWrite is FALSE, ErrorID is 50.

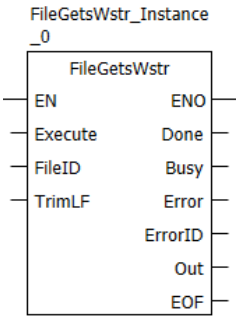
- If a file with the name FileName already exists and the file is read-only, ErrorID is 50.
- If the SD card is not successfully mounted, data stored on the card cannot be accessed, and ErrorID is 2.
- If multiple threads execute the FileWriteVar instruction simultaneously, execution is queued (threads themselves are not blocked). If the operation does not complete within 10 task cycles, ErrorID is 1540.
- If the FileWriteVar instruction is executed after user directory size initialization fails, ErrorID is 1543.

18.2.11.14 File Read/Write (Wstring Supported)

18.2.11.14.1 FileGetsWstr

Reads a string of one line from the specified file.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FileGetsWstr	Get text string	FB		FileGetsWstr_Instance_0(Execute := , FileID := , TrimLF := , Done => , Busy =>, Error =>, ErrorID => , Out => , EOF =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Enable	Follows data type	FALSE	(Triggered by edges) Execution of the function block switch
	FileID	File ID	Follows data type	0	ID of the file to write

Input/Output	Variable	Name	Valid Range	Default Value	Description
	TrimLF	Line feed designation	Follows data type	FALSE	Handling of the line feed code of the read text string TRUE: Delete. FALSE: Do not delete.
Output	Done	Function block completed state	Follows data type	-	Function block completed state
	Busy	Function block execution flag	Follows data type	-	Function block execution flag
	Error	Function block fault flag	Follows data type	-	Function block fault flag
	ErrorID	Function block fault ID	Follows data type	-	Function block fault ID
	Out	Read text string	Maximum 1985 bytes	-	Read text string
	EOF	End of file	Follows data type	-	Whether the end of the file is reached TRUE: Reached. FALSE: Not reached.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	WSTRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

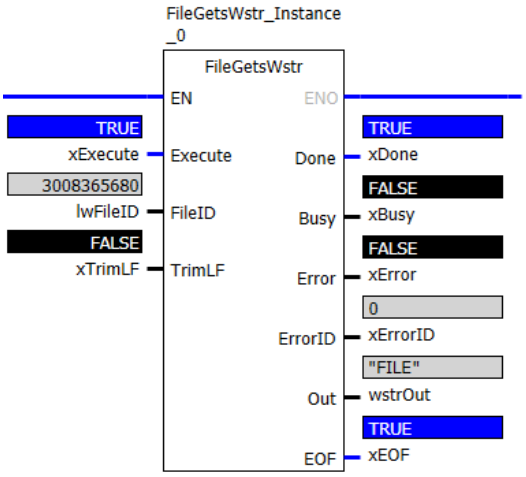
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	WSTRING
FileID	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
TrimLF	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
EOF	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction reads a string of one line from the file position indicator in the file specified by FileID.
  - The file position indicator is set at the desired location in advance with the FileSeek instruction.
  - Line endings are determined by a line feed code. The text string that is read is written to read the text string Out.
  - Line endings are automatically identified from three types: CR (\r), LF (\n), and CR + LF (\r\n). If line feed designation TrimLF is TRUE, the line feed code is deleted from the text string before it is written to Out.
  - If data is read to the end of the file, EOF (end of file) changes to TRUE.
- **Program example**

  Note: The ID corresponding to the variable lwFileID is generated by the FileOpen instruction.

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    FileOpen_Instance_0: FileOpen;
    FilePutsWstr_Instance_0: FilePutsWstr;
    FileSeek_Instance_0: FileSeek;
    FileGetsWstr_Instance_0: FileGetsWstr;
    FileClose_Instance_0: FileClose;
    xExecute: BOOL;
    lwFileID: LWORD;
    xTrimLF: BOOL;
    xDone: BOOL;
    xBusy: BOOL;
    xError: BOOL;
    xErrorID: WORD;
    wstrOut: WSTRING(255);
    xEOF: BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>


</td><td>

```
FileGetsWstr_Instance_0(
Execute  TRUE  := xExecute  TRUE ,
FileID  3008365680  := lwFileID  3008365680 ,
TrimLF  FALSE  := xTrimLF  FALSE ,
Done  TRUE  => xDone  TRUE ,
Busy  FALSE  => xBusy  FALSE ,
Error  FALSE  => xError  FALSE ,
ErrorID  0  => xErrorID  0 ,
Out  "FILE"  => wstrOut  "FILE" ,
EOF  TRUE  => xEOF  TRUE );
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

		Name	Type	Address	Value
1		FileOpen_Instance_0	FileOpen		
2		FilePutsWstr_Instance_0	FilePutsWstr		
3		FileSeek_Instance_0	FileSeek		
4		FileGetsWstr_Instance_0	FileGetsWstr		
5		FileClose_Instance_0	FileClose		
6		xExecute	BOOL		TRUE
7		lwFileID	LWORD		3008365680
8		xTrimLF	BOOL		FALSE
9		xDone	BOOL		TRUE
10		xBusy	BOOL		FALSE
11		xError	BOOL		FALSE
12		xErrorID	WORD		0
13		wstrOut	WSTRING(255)		"FILE"
14		xEOF	BOOL		TRUE

</td></tr>
</table>

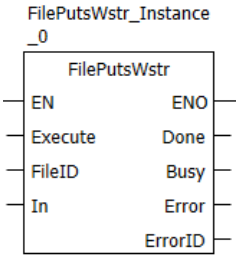
- **Precautions**
  - Before using this instruction, please read the 18.2.11.2 Overview to avoid improper operations.
  - If the length of the one-line text string exceeds 1985 bytes, the first 1985 bytes of the text string and the NULL text string at the end are stored in Out. Execute the FileOpen instruction in advance to obtain the value for FileID.
  - An exception occurs under the following conditions. The value of Error is TRUE.
    - If the file specified by FileID does not exist, ErrorID is 1.
    - If the file specified by FileID is not opened in a reading mode, ErrorID is 1.

- If the input FileID is invalid (less than 0 or greater than 300), ErrorID is 1537.
- If a file opened by the FileOpen instruction is used by FileGetsWstr after already being closed by FileClose, ErrorID is 1538.

18.2.11.14.2 FilePutsWstr

Writes a string to the specified file.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FilePutsWstr	Input text string	FB		FilePutsWstr_Instance_0(Execute := , FileID := , In := , Done => , Busy =>, Error =>, ErrorID =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Enable	Follows data type	FALSE	(Triggered by edges) Execution of the function block switch
	FileID	File ID	Follows data type	0	ID of the file to write
	In	Text string to write	Max. 1985 characters	''	Text string to write
Output	Done	Function block completed state	Follows data type	-	Function block completed state
	Busy	Function block execution flag	Follows data type	-	Function block execution flag
	Error	Function block fault flag	Follows data type	-	Function block fault flag

Input/Output	Variable	Name	Valid Range	Default Value	Description
	ErrorID	Function block fault ID	Follows data type	-	Function block fault ID

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	WSTRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FileID	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

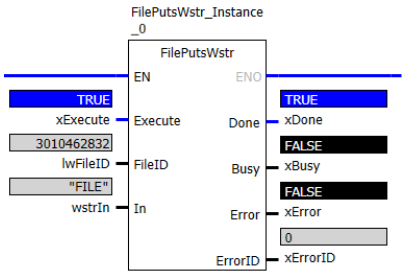
  This instruction writes the text string In to the position of the file position indicator in the file specified by FileID. The file position indicator is set at the desired location in advance with the FileSeek instruction.

- **Program example**

  Note: The ID corresponding to the variable lwFileID is generated by the FileOpen instruction.

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    FileOpen_Instance_0: FileOpen;
    FilePutsWstr_Instance_0: FilePutsWstr;
    FileSeek_Instance_0: FileSeek;
    FileGetsWstr_Instance_0: FileGetsWstr;
    FileClose_Instance_0: FileClose;
    xExecute: BOOL;
    lwFileID: LWORD;
    wstrIn: WSTRING(255) := "FILE";
    xDone: BOOL;
    xBusy: BOOL;
    xError: BOOL;
    xErrorID: WORD;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>
</td><td>

```
FilePutsWstr_Instance_0(
Execute  TRUE  := xExecute  TRUE ,
FileID  3010462832  := lwFileID  3010462832 ,
In  "FILE"  := wstrIn  "FILE" ,
Done  TRUE  => xDone  TRUE ,
Busy  FALSE  => xBusy  FALSE ,
Error  FALSE  => xError  FALSE ,
ErrorID  0  => xErrorID  0 );
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

		Name	Type	Address	Value
1		FileOpen_Instance_0	FileOpen		
2		FilePutsWstr_Instance_0	FilePutsWstr		
3		FileSeek_Instance_0	FileSeek		
4		FileGetsWstr_Instance_0	FileGetsWstr		
5		FileClose_Instance_0	FileClose		
6		xExecute	BOOL		TRUE
7		lwFileID	LWORD		3010462832
8		wstrIn	WSTRING(255)		"FILE"
9		xDone	BOOL		TRUE
10		xBusy	BOOL		FALSE
11		xError	BOOL		FALSE
12		xErrorID	WORD		0

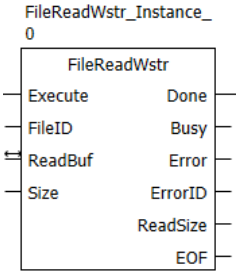
</td></tr>
</table>

- **Precautions**
  - Before using this instruction, please read the 18.2.11.2 Overview to avoid improper operations.
  - Execute the FileOpen instruction in advance to obtain the value for FileID.
  - An exception occurs and Error becomes TRUE in the following cases:
    - If there is insufficient space available on the SD card, ErrorID is 41.
    - If the file specified by FileID does not exist, ErrorID is 1.
    - If In is empty, ErrorID is 2.
    - If the file specified by FileID is not opened in a writing mode, ErrorID is 1.
    - If the input FileID is invalid (less than 0 or greater than 300), ErrorID is 1537.
    - If a file opened by the FileOpen instruction is used by FilePutsWstr after already being closed by FileClose, ErrorID is 1538.
    - If multiple threads execute the FilePutsWstr instruction simultaneously, execution is queued (threads themselves are not blocked). If the operation does not complete within 10 task cycles, ErrorID is 1540.
    - If the FilePutsWstr instruction is executed after user directory size initialization fails, ErrorID is 1543.

18.2.11.14.3 FileReadWstr

Reads data from the specified file.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FileReadWstr	Read file	FB		FileReadWstr_Instance_0(Execute := , FileID := , ReadBuf := , Size := , Done => , Busy =>, Error =>, ErrorID => , ReadSize => , EOF =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Enable	Follows data type	FALSE	(Triggered by edges) Execution of the function block switch
	FileID	File ID	Follows data type	0	ID of the file to read
	Size	Number of elements to read	Follows data type	1	Number of elements to read
Input/Output	ReadBuf[ ] array	Read buffer	Follows data type	-	Buffer in which to write data that was read
Output	Done	Function block completed state	Follows data type	-	Function block completed state

Input/Output	Variable	Name	Valid Range	Default Value	Description
	Busy	Function block execution flag	Follows data type	-	Function block execution flag
	Error	Function block fault flag	Follows data type	-	Function block fault flag
	ErrorID	Function block fault ID	Follows data type	-	Function block fault ID
	ReadSize	Number of elements actually read	Follows data type	-	Number of elements actually read
	EOF	End of file	Follows data type	-	Whether the end of the file is reached TRUE: Reached. FALSE: Not reached.

Variable types

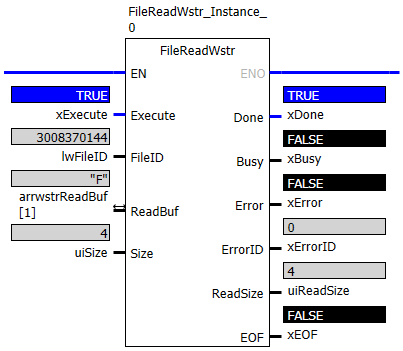
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FileID	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ReadBuf [ ] array	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Array of enumerations or structures can also be specified.																				
Size	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

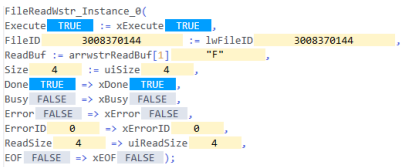
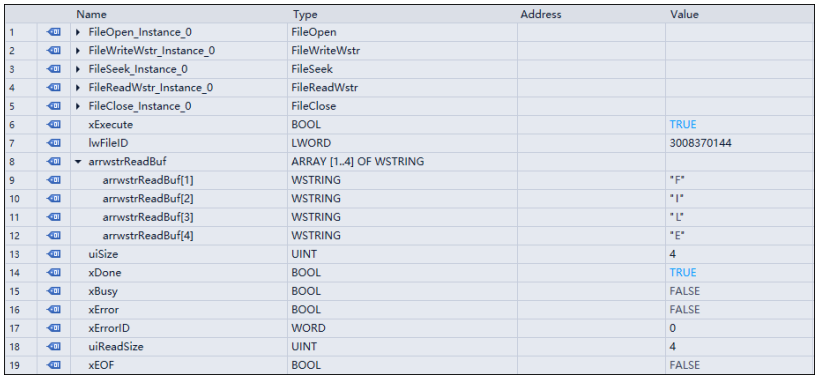
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
ReadSize	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
EOF	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction reads data from the file position indicator in the file specified by FileID, and then stores the data in read buffer ReadBuf[ ].
  - The file position indicator is set at the desired location in advance with the FileSeek instruction.
  - The amount of data that is read is the size of the data type of ReadBuf[ ] times Size.
  - The number of elements actually read is stored in ReadSize. In general, values of Size and ReadSize are the same. If the amount of data from the file position indicator to the end of the file is less than the value of Size, no error occurs and the data to the end of the file is stored in ReadBuf[ ]. If that occurs, the value of ReadSize is less than the value of Size.
  - When reading to the end of the file, EOF becomes TRUE; if not reading to the end, EOF is FALSE.
- **Program example**

  Note: The ID corresponding to the variable lwFileID is generated by the FileOpen instruction.

Item	LD	ST
Define variables	VAR FileOpen_Instance_0: FileOpen; FileWriteWstr_Instance_0: FileWriteWstr; FileSeek_Instance_0: FileSeek; FileReadWstr_Instance_0: FileReadWstr; FileClose_Instance_0: FileClose; xExecute: BOOL; lwFileID: LWORD; arrwstrReadBuf: ARRAY [1..4] OF WSTRING; uiSize: UINT := 4; xDone: BOOL; xBusy: BOOL; xError: BOOL; xErrorID: WORD; uiReadSize: UINT; xEOF: BOOL; END_VAR	

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>

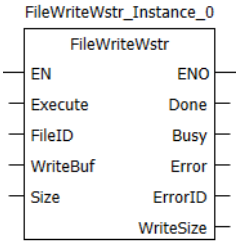
</td><td>Operation result</td><td>
</td></tr>
</table>

- **Precautions**
  - Before using this instruction, please read the 18.2.11.2 Overview to avoid improper operations.
  - If the data is read to the end of the file and the size of the data is not evenly divisible by the size of the data type of ReadBuf[ ], the data that is insufficient for the data size of ReadBuf[ ] is discarded. The file position indicator advances to the end of the file, and the value of EOF changes to TRUE.
  - If the value of Size exceeds the ReadBuf[ ] array, a program fault occurs, Error becomes TRUE, and ErrorID is 1030. The elements in ReadBuf[ ] retain their values from before executing this instruction.
  - Execute the FileOpen instruction in advance to obtain the value for FileID.
  - If ReadBuf[ ] is an array of structures, adjustment areas between members may be inserted depending on the composition.
  - An exception occurs and Error becomes TRUE in the following cases:
    - If the file specified by FileID does not exist, ErrorID is 1.
    - If Size is not 0 and the file specified by FileID is not opened in a reading mode, ErrorID is 1.
    - If the input FileID is invalid (less than 0 or greater than 300), ErrorID is 1537.
    - If a file opened by the FileOpen instruction is used by FileReadWstr after already being closed by FileClose, ErrorID is 1538.

18.2.11.14.4 FileWriteWstr

Writes data to the specified file.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FileWriteWstr	Write file	FB		FileWriteWstr_Instance_0(Execute := , FileID := , WriteBuf := , Size := , Done => , Busy =>, Error =>, ErrorID => , WriteSize =>);

- **Related variables**

Input Variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Enable	Follows data type	FALSE	(Triggered by edges) Execution of the function block switch
	FileID	File ID	Follows data type	0	ID of the file to write
	WriteBuf[ ] array	Write buffer	Follows data type	-	Data to write
	Size	Number of elements to write	Follows data type	1	Number of elements to write
Output	Done	Function block completed state	Follows data type	-	Function block completed state
	Busy	Function block execution flag	Follows data type	-	Function block execution flag
	Error	Function block fault flag	Follows data type	-	Function block fault flag
	ErrorID	Function block fault ID	Follows data type	-	Function block fault ID

Input/Output	Variable	Name	Valid Range	Default Value	Description
	WriteSize	Number of elements actually written	Follows data type	-	Number of elements actually written

Variable types

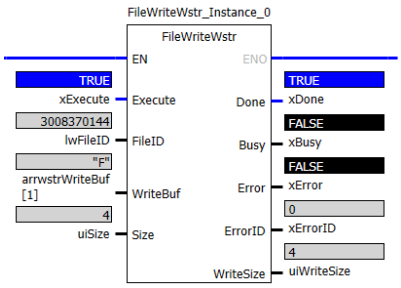
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FileID	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
WriteBuf[ ] array	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Array of enumerations or structures can also be specified.																				
Size	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
WriteSize	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction writes the data written to WriteBuf[ ] to the file position indicator in the file specified by FileID.
  - The file position indicator is set at the desired location in advance with the FileSeek instruction.
  - The amount of data that is written is the size of the data type of WriteBuf[ ] times Size.
  - The number of elements actually written is saved to WriteSize. In general, values of Size and ReadSize are the same.
- **Program example**

  Note: The ID corresponding to the variable lwFileID is generated by the FileOpen instruction.

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    FileOpen_Instance_0: FileOpen;
    FileWriteWstr_Instance_0: FileWriteWstr;
    FileSeek_Instance_0: FileSeek;
    FileReadWstr_Instance_0: FileReadWstr;
    FileClose_Instance_0: FileClose;
    xExecute: BOOL;
    lwFileID: LWORD;
    arrwstrWriteBuf: ARRAY [1..4] OF WSTRING := ["F","I","L","E"];
    uiSize: UINT := 4;
    xDone: BOOL;
    xBusy: BOOL;
    xError: BOOL;
    xErrorID: WORD;
    uiWriteSize: UINT;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>


</td><td>

```
FileWriteWstr_Instance_0(
Execute  TRUE  := xExecute  TRUE ,
FileID  3008370144  := lwFileID  3008370144 ,
WriteBuf := arrwstrWriteBuf[1]  "F" ,
Size  4  := uiSize  4 ,
Done  TRUE  => xDone  TRUE ,
Busy  FALSE  => xBusy  FALSE ,
Error  FALSE  => xError  FALSE ,
ErrorID  0  => xErrorID  0 ,
WriteSize  4  => uiWriteSize  4 );
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

		Name	Type	Address	Value
1		FileOpen_Instance_0	FileOpen		
2		FileWriteWstr_Instance_0	FileWriteWstr		
3		FileSeek_Instance_0	FileSeek		
4		FileReadWstr_Instance_0	FileReadWstr		
5		FileClose_Instance_0	FileClose		
6		xExecute	BOOL		TRUE
7		lwFileID	LWORD		3008370144
8		arrwstrWriteBuf	ARRAY [1..4] OF WSTRING		
9		arrwstrWriteBuf[1]	WSTRING		"F"
10		arrwstrWriteBuf[2]	WSTRING		"I"
11		arrwstrWriteBuf[3]	WSTRING		"L"
12		arrwstrWriteBuf[4]	WSTRING		"E"
13		uiSize	UINT		4
14		xDone	BOOL		TRUE
15		xBusy	BOOL		FALSE
16		xError	BOOL		FALSE
17		xErrorID	WORD		0
18		uiWriteSize	UINT		4

</td></tr>
</table>

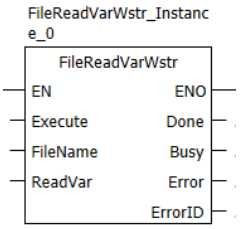
- **Precautions**
  - Before using this instruction, please read the 18.2.11.2 Overview to avoid improper operations.
  - If the value of Size exceeds the WriteBuf[ ] array, a program fault occurs, Error becomes TRUE, and ErrorID is 1030.
  - Execute the FileOpen instruction in advance to obtain the value for FileID.
  - If WriteBuf[ ] is an array of structures, adjustment areas between members may be inserted depending on the composition.
  - An exception occurs under the following conditions. The value of Error is TRUE.
    - If the file specified by FileID does not exist, ErrorID is 1.
    - If the file specified by FileID is not opened in a writing mode, ErrorID is 1.
    - If there is insufficient space available on the SD card, ErrorID is 41.
    - If the input FileID is invalid (less than 0 or greater than 300), ErrorID is 1537.
    - If a file opened by the FileOpen instruction is used by FileWriteWstr after already being closed by FileClose, ErrorID is 1538.

- If multiple threads execute the FileWriteWstr instruction simultaneously, execution is queued (threads themselves are not blocked). If the operation does not complete within 10 task cycles, ErrorID is 1540.
- If the FileWriteWstr instruction is executed after user directory size initialization fails, ErrorID is 1543.

18.2.11.14.5 FileReadVarWstr

Reads the values of the specified file as binary data and writes the data to a variable.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FileReadVarWstr	File read variable	FB		FileReadVarWstr_Instance_0(Execute := , FileName :=, ReadVar := , Done => , Busy =>, Error =>, ErrorID =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Enable	Follows data type	FALSE	(Triggered by edges) Execution of the function block switch
	FileName	Specified file name	Max. 200 characters	''	Name of the file to which to write variable data
	ReadVar	Position variable to read	Follows data type	-	Position variable from which to read variable data
Output	Done	Function block completed state	Follows data type	-	Function block completed state
	Busy	Function block execution flag	Follows data type	-	Function block execution flag

Input/Output	Variable	Name	Valid Range	Default Value	Description
	Error	Function block fault flag	Follows data type	-	Function block fault flag
	ErrorID	Function block fault ID	Follows data type	-	Function block fault ID

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FileName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–
ReadVar	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration, an array, an array element, a structure, or a structure member can also be specified.																				
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

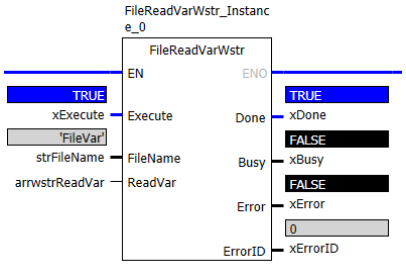
- **Function**

  This instruction reads variable data from the file specified by FileName in binary format and assigns the read data to the position variable specified by ReadVar.

- **Program example**

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    FileWriteVarWstr_Instance_0: FileWriteVarWstr;
    FileReadVarWstr_Instance_0: FileReadVarWstr;
    xExecute: BOOL;
    strFileName: STRING;
    arrwstrReadVar: ARRAY [1..4] OF WSTRING;
    xOverWrite: BOOL;
    xDone: BOOL;
    xBusy: BOOL;
    xError: BOOL;
    xErrorID: WORD;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>
</td><td>

```
FileReadVarWstr_Instance_0(
Execute  TRUE  := xExecute  TRUE ,
FileName   'FileVar'   := strFileName   'FileVar' ,
ReadVar := arrwstrReadVar,
Done  TRUE  => xDone  TRUE ,
Busy  FALSE  => xBusy  FALSE ,
Error  FALSE  => xError  FALSE ,
ErrorID   0   => xErrorID   0 );
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

		Name	Type	Address	Value
1		FileWriteVarWstr_Instance_0	FileWriteVarWstr		
2		FileReadVarWstr_Instance_0	FileReadVarWstr		
3		xExecute	BOOL		TRUE
4		strFileName	STRING		'FileVar'
5		arrwstrReadVar	ARRAY [1..4] OF WSTRING		
6		arrwstrReadVar[1]	WSTRING		"F"
7		arrwstrReadVar[2]	WSTRING		"I"
8		arrwstrReadVar[3]	WSTRING		"L"
9		arrwstrReadVar[4]	WSTRING		"E"
10		xOverWrite	BOOL		FALSE
11		xDone	BOOL		TRUE
12		xBusy	BOOL		FALSE
13		xError	BOOL		FALSE
14		xErrorID	WORD		0

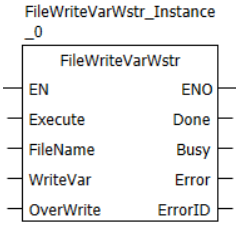
</td></tr>
</table>

- **Precautions**
  - Before using this instruction, please read the 18.2.11.2 Overview to avoid improper operations.
  - If the size of the specified file is greater than that of ReadVar, no error occurs and only the data of the size of ReadVar is read. If the size of the specified file is less than that of ReadVar, no error occurs and only the data of the size of the specified file is read. The remaining area in ReadVar retains the values used before execution of this instruction. If ReadVar is a structure, adjustment areas between members may be inserted depending on the composition.
  - An exception occurs and Error becomes TRUE in the following cases:
    - If FileName is empty or an invalid filename, ErrorID is 1.
    - If the file specified by FileName does not exist, ErrorID is 16.

18.2.11.14.6 FileWriteVarWstr

Writes the value of a variable to the specified file in binary format.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FileWriteVarWstr	File write variable	FB		FileWriteVarWstr_Instance_0(Execute := , FileName :=, WriteVar := , OverWrite := , Done => , Busy =>, Error =>, ErrorID =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Enable	Follows data type	FALSE	(Triggered by edges) Execution of the function block switch
	FileName	Specified file name	Max. 200 characters	''	Name of the file to which to write variable data
	WriteVar	Specified variable	Follows data type	-	Variable to write
	OverWrite	Overwrite enable	Follows data type	FALSE	TRUE: Enable overwrite FALSE: Prohibit overwrite
Output	Done	Function block completed state	Follows data type	-	Function block completed state
	Busy	Function block execution flag	Follows data type	-	Function block execution flag
	Error	Function block fault flag	Follows data type	-	Function block fault flag
	ErrorID	Function block fault ID	Follows data type	-	Function block fault ID

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
Execute	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FileName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–
WriteVar	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration, an array, an array element, a structure, or a structure member can also be specified.																				
OverWrite	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Done	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Busy	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Error	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorID	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

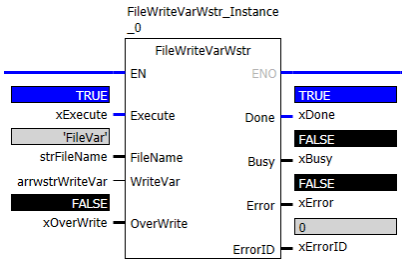
- **Function**
  - This instruction writes the value of variable WriteVar to the file specified by FileName in binary format.
  - If there is no file with the same name as FileName, a new file is created.
  - If a file with the same name as FileName already exists, the following is performed depending on the value of OverWrite, as shown below.

Value of OverWrite	Operation
TRUE	The file is overwritten.
FALSE	The file is not overwritten and an error occurs.

- **Program example**

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    FileWriteVarWstr_Instance_0: FileWriteVarWstr;
    FileReadVarWstr_Instance_0: FileReadVarWstr;
    xExecute: BOOL;
    strFileName: STRING;
    arrwstrWriteVar: ARRAY [1..4] OF WSTRING := ["F","I","L","E"];
    xOverWrite: BOOL;
    xDone: BOOL;
    xBusy: BOOL;
    xError: BOOL;
    xErrorID: WORD;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>


</td><td>

```
FileWriteVarWstr_Instance_0(
Execute  TRUE  := xExecute  TRUE ,
FileName  'FileVar'  := strFileName  'FileVar' ,
WriteVar := arrwstrWriteVar,
OverWrite  FALSE  := xOverWrite  FALSE ,
Done  TRUE  => xDone  TRUE ,
Busy  FALSE  => xBusy  FALSE ,
Error  FALSE  => xError  FALSE ,
ErrorID  0  => xErrorID  0 );
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

		Name	Type	Address	Value
1		FileWriteVarWstr_Instance_0	FileWriteVarWstr		
2		FileReadVarWstr_Instance_0	FileReadVarWstr		
3		xExecute	BOOL		TRUE
4		strFileName	STRING		'FileVar'
5		arrwstrWriteVar	ARRAY [1..4] OF WSTRING		
6		arrwstrWriteVar[1]	WSTRING		"F"
7		arrwstrWriteVar[2]	WSTRING		"I"
8		arrwstrWriteVar[3]	WSTRING		"L"
9		arrwstrWriteVar[4]	WSTRING		"E"
10		xOverWrite	BOOL		FALSE
11		xDone	BOOL		TRUE
12		xBusy	BOOL		FALSE
13		xError	BOOL		FALSE
14		xErrorID	WORD		0

</td></tr>
</table>

- **Notes**
  - Before using this instruction, please read the 18.2.11.2 Overview to avoid improper operations.
  - If WriteVar is a structure, adjustment areas between members may be inserted depending on the composition.
  - An exception occurs under the following conditions. The value of Error is TRUE.
    - If the SD card is write-protected, ErrorID is 1.
    - If there is insufficient space available on the SD card, ErrorID is 41.
    - If FileName is empty and OverWrite is FALSE, ErrorID is 50; if FileName is empty and OverWrite is TRUE, ErrorID is 1.
    - If FileName is an invalid file name and OverWrite is FALSE, ErrorID is 50; if FileName is an invalid file name and OverWrite is TRUE, ErrorID is 1.
    - If a file with the name specified by FileName exits and OverWrite is FALSE, ErrorID is 50.
    - If a file with the name FileName already exists and the file is read-only, ErrorID is 50.
    - If multiple threads execute the FileWriteVarWstr instruction simultaneously, execution is queued (threads themselves are not blocked). If the operation does not complete within 10 task cycles, ErrorID is 1540.
    - If the FileWriteVarWstr instruction is executed after user directory size initialization fails, ErrorID is 1543.

18.2.12 Special File Processing

18.2.12.1 XML File Processing

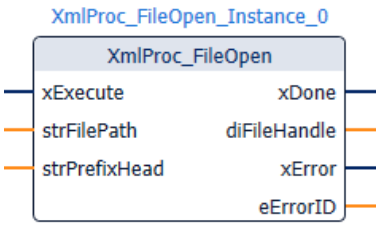
18.2.12.1.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.12.1.2 XmlProc_FileOpen	FB	Open the specified XML file	FileProc	Library version 1.0.0.0 and later
18.2.12.1.3 XmlProc_FileClose	FB	Close the specified XML file		Library version 1.0.0.0 and later
18.2.12.1.4 XmlProc_FileDelete	FB	Delete the specified XML file		Library version 1.0.0.0 and later
18.2.12.1.5 XmlProc_GetNameSpaceInfo	FB	Get namespace		Library version 1.0.0.0 and later
18.2.12.1.6 XmlProc_NodeEdit	FB	Edit nodes in the specified XML file		Library version 1.0.0.0 and later
18.2.12.1.7 XmlProc_AddJmfInfo	FB	Add information to a JMF file		Library version 1.0.0.0 and later

18.2.12.1.2 XmlProc_FileOpen

Opens the specified XML file and returns a file handle.

- **Instruction format**

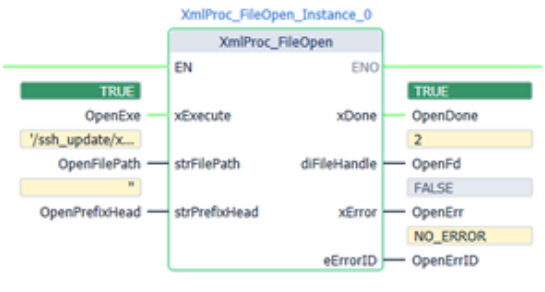
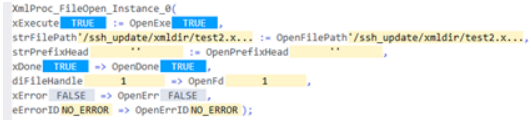
Instruction	Name	FB/FC	LD Representation	ST Representation
XmlProc_FileOpen	Open the specified XML file	FB		XmlProc_FileOpen_Instance_0(xExecute := , strFilePath := , strPrefixHead := , xDone => , diFileHandle => , xError => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Trigger enable bit	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	strFilePath	File path	STRING[128]	-	-	Path of the XML file.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	strPrefixHead	Namespace prefix	STRING[8]	-	ns	Prefix of the namespace of the XML file. Default is ns.
Output	xDone	Completion flag	BOOL	TRUE, FALSE	FALSE	Completion flag.
	diFileHandle	File handle	DINT	[-1, 1000]	0	File handle returned after the file is opened.
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Error flag for file open failure.
	eErrorID	Fault ID	XML_PROC_ERROR_ID	-	NO_ERROR	Error code for file open failure. See 18.2.12.1.8 Fault Codes for details.

- **Function**
  - Before instruction execution, input the file path "strFilePath". This is a relative path, and the root directory is the PLC SFTP server path.
  - Before instruction execution, input the prefix of the namespace. If this item is empty, the default is ns. Taking ns as an example, the default namespaces of the XML file are ns0, ns1, etc. For details on XML file namespaces, use the 18.2.12.1.5 XmlProc_GetNameSpaceInfo instruction.
  - After instruction execution, the file handle "diFileHandle" is returned. A "diFileHandle" value of 0 or greater indicates that the file is opened successfully, allowing file operations via this handle.
- **Program example**

Item	LD	ST
Define variables	XmlProc_FileOpen_Instance_0: XmlProc_FileOpen; OpenExe : BOOL := FALSE; OpenFilePath : STRING[128] := '/ssh_update/xmldir/test1.xml'; OpenPrefixHead : STRING[8]; OpenDone : BOOL := FALSE; OpenFd : DINT := 0; OpenErr : BOOL := FALSE; OpenErrID : XML_PROC_ERROR_ID := XML_PROC_ERROR_ID.NO_ERROR;	
Program		

Item	LD	ST
Operation result	XmlProc_FileOpen_Instance_0 : XmlProc_FileOpen OpenExe : BOOL = TRUE OpenFilePath : STRING[128] = '/ssh_update/xmldir/test2.xml' OpenPrefixHead : STRING[8] = '' OpenDone : BOOL = TRUE OpenFd : DINT = 1 OpenErr : BOOL = FALSE OpenErrID : XML_PROC_ERROR_ID = NO_ERROR	

- **Precautions**
  - Open the XML file before use and close it after file operations are completed. Avoid frequent open/close actions during use.
  - After the file is opened successfully, "xDone" is set and "diFileHandle" is 0 or greater.
  - When opening a file, the system checks the number of XML files currently open. Supports up to 100 XML files opened simultaneously; otherwise, an error is reported indicating that the number of opened files exceeds the limit "OPENED_FILE_NUM_OVER_LIMIT".
  - When opening a file, the system checks the size of the XML file currently being opened. The file size must not exceed 1 MB; otherwise, an error is reported indicating that the opened file size exceeds the limit "FILE_SIZE_OVER_LIMIT".
  - When opening a file, the system checks the total size of all currently open files. The total size of all open XML files must not exceed 5% of the total memory of the PLC; otherwise, an error is reported indicating that the cumulative size of open files exceeds the limit "OPENED_FILE_SIZE_SUM_OVER_LIMIT".
  - The "diFileHandle" used here differs from file handles in other libraries and should not be mixed.

18.2.12.1.3 XmlProc_FileClose

Closes the XML file and determines how to save the file based on the close mode.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
XmlProc_FileClose	Close the specified XML file	FB	XmlProc_FileClose_Instance_0 XmlProc_FileClose xExecute diFileHandle eCloseMode strSaveAsPath xDone xError eErrorID	XmlProc_FileClose_Instance_0(xExecute := , diFileHandle := , eCloseMode := , strSaveAsPath := , xDone => , xError => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Trigger enable bit	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	diFileHandle	File handle	DINT	[-1, 1000]	0	Handle of the file.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	eCloseMode	Close mode	XML_CLOSE_MODE	-	-	Close mode of the file.
	strSaveAsPath	Save Path	STRING[128]	-	-	Path for saving the file as a new file.
Output	xDone	Completion flag	BOOL	TRUE, FALSE	FALSE	Completion flag.
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Error flag.
	eErrorID	Fault code	XML_PROC_ERROR_ID	-	NO_ERROR	See 18.2.12.1.8 Fault Codes.

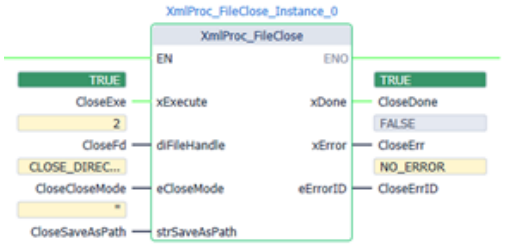
- **Function**
  - Before instruction execution, input the file handle "diFileHandle" and the file close mode "eCloseMode". The data type of "eCloseMode" is XML_CLOSE_MODE. The enumeration values are defined as follows:

Variable	Value	Description
CLOSE_DIRECTLY	16#0000	Close directly; changes will not be written back to the file.
CLOSE_AND_SAVE	16#0001	Close and save; changes will be written back to the original file.
CLOSE_AND_SAVE_AS	16#0002	Close and save the content to the specified file.

  - If the close mode is CLOSE_AND_SAVE_AS, the parameter "strSaveAsPath" must also be set.
  - After the instruction execution, the result is returned. If "xDone" is TRUE, the operation is successful.
- **Program example**

Item	LD	ST
Define variables		
Program		

```
XmlProc_FileClose_Instance_0: XmlProc_FileClose;
CloseExe        : BOOL := FALSE;
CloseFd         : DINT := 0;
CloseCloseMode  : XML_CLOSE_MODE;
CloseSaveAsPath : STRING[128];
CloseDone       : BOOL := FALSE;
CloseErr        : BOOL := FALSE;
CloseErrID      : XML_PROC_ERROR_ID := XML_PROC_ERROR_ID.NO_ERROR;
```




```
XmlProc_FileClose_Instance_0(
xExecute TRUE := CloseExe TRUE ,
diFileHandle 1 := CloseFd 1 ,
eCloseModeCLOSE_DIREC... := CloseCloseModeCLOSE_DIREC...,
strSaveAsPath '' := CloseSaveAsPath '' ,
xDone TRUE => CloseDone TRUE ,
xError FALSE => CloseErr FALSE ,
eErrorID NO_ERROR => CloseErrID NO_ERROR );
```

Item	LD	ST
Operation result	XmlProc_FileClose_Instance_0 / XmlProc_FileClose CloseExe / BOOL / TRUE CloseFd / DINT / 1 CloseCloseMode / XML_CLOSE_MODE / CLOSE_DIRECTLY CloseSaveAsPath / STRING[128] / '' CloseDone / BOOL / TRUE CloseErr / BOOL / FALSE CloseErrID / XML_PROC_ERROR_ID / NO_ERROR	

- **Precautions**
  - "strSaveAsPath" is valid only when the close mode is CLOSE_AND_SAVE_AS. In other modes, this parameter is empty.
  - When the close mode is CLOSE_AND_SAVE_AS, if the path specified by "strSaveAsPath" does not exist, the function block automatically creates the corresponding path.

18.2.12.1.4 XmlProc_FileDelete

Deletes the file at the specified path.

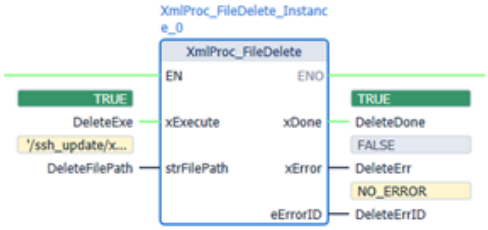
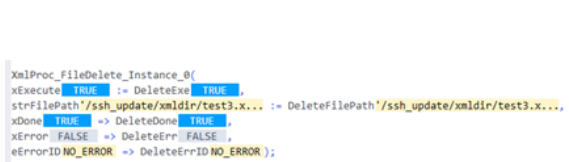
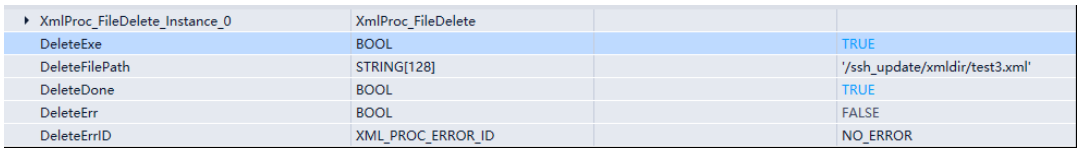
- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
XmlProc_FileDelete	Delete the specified XML file	FB	XmlProc_FileDelete_Instance_0 XmlProc_FileDelete xExecute / xDone strFilePath / xError eErrorID	XmlProc_FileDelete_Instance_0(xExecute := , strFilePath := , xDone => , xError => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Trigger enable bit	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	strFilePath	File path	STRING[128]	-	-	File path.
Output	xDone	Completion flag	BOOL	TRUE, FALSE	FALSE	Completion flag.
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Error flag.
	eErrorID	Fault code	XML_PROC_ERROR_ID	-	NO_ERROR	See 18.2.12.1.8 Fault Codes.

- **Function**
  - Before instruction execution, input the file path "strFilePath". This path is relative to the SFTP server root directory.
  - After instruction execution, the corresponding file at the specified path will be deleted.
- **Program example**

Item	LD	ST
Define variables	XmlProc_FileDelete_Instance_0: XmlProc_FileDelete; DeleteExe : BOOL := FALSE; DeleteFilePath : STRING[128]:= '/ssh_update/xmldir/test3.xml'; DeleteDone : BOOL := FALSE; DeleteErr : BOOL := FALSE; DeleteErrID : XML_PROC_ERROR_ID := XML_PROC_ERROR_ID.NO_ERROR;	
Program		
Operation result		

XmlProc_FileDelete_Instance_0	XmlProc_FileDelete		
DeleteExe	BOOL		TRUE
DeleteFilePath	STRING[128]		'/ssh_update/xmldir/test3.xml'
DeleteDone	BOOL		TRUE
DeleteErr	BOOL		FALSE
DeleteErrID	XML_PROC_ERROR_ID		NO_ERROR

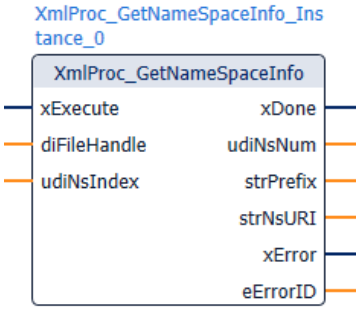
- **Precautions**

  This instruction deletes the file from the disk. Deletion is irreversible and cannot be recovered.

18.2.12.1.5 XmlProc_GetNameSpaceInfo

Gets the namespace information of the XML file.

- **Instruction format**

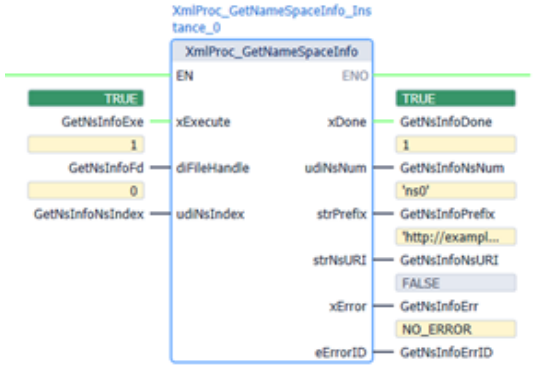
Instruction	Name	FB/FC	LD Representation	ST Representation
XmlProc_GetNameSpaceInfo	Get namespace	FB		XmlProc_GetNameSpaceInfo_Instance_0(xExecute := , diFileHandle := , udiNsIndex := , xDone => , udiNsNum => , strPrefix => , strNsURI => , xError => , eErrorID =>);

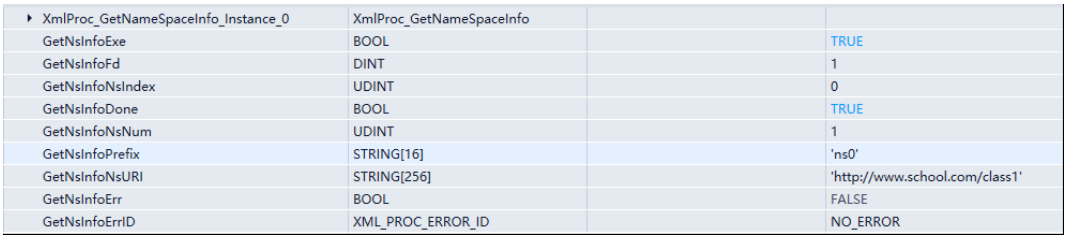
- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Trigger enable bit	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	diFileHandle	File handle	DINT	[-1, 1000]	0	Handle of the file.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	udiNsIndex	Namespace index	UDINT	[0, udiNsNum-1]	0	Index of the namespace, starting from 0.
Output	xDone	Completion flag	BOOL	TRUE, FALSE	FALSE	Completion flag.
	udiNsNum	Number of namespaces	UDINT	-	0	Number of namespaces.
	strPrefix	Namespace prefix	STRING[16]	-	-	Namespace prefix.
	strNsURI	Namespace URI	STRING[256]	-	-	Namespace URI.
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Error flag.
	eErrorID	Fault code	XML_PROC_ERROR_ID	-	NO_ERROR	See 18.2.12.1.8 Fault Codes.

- **Function**
  - Before instruction execution, input the file handle "diFileHandle" and the namespace index "udiNsIndex".
  - After execution, the instruction returns the namespace prefix and namespace URI corresponding to the given index, as well as the total number of namespaces in the file referenced by "diFileHandle".
- **Program example**

Item	LD	ST
Define variables	XmlProc_GetNameSpaceInfo_Instance_0: XmlProc_GetNameSpaceInfo; GetNsInfoExe : BOOL := FALSE; GetNsInfoFd : DINT := 0; GetNsInfoNsIndex : UDINT := 0; GetNsInfoDone : BOOL := FALSE; GetNsInfoNsNum : UDINT := 0; GetNsInfoPrefix : STRING[16]; GetNsInfoNsURI : STRING[256]; GetNsInfoErr : BOOL := FALSE; GetNsInfoErrID : XML_PROC_ERROR_ID := XML_PROC_ERROR_ID.NO_ERROR;	
Program		

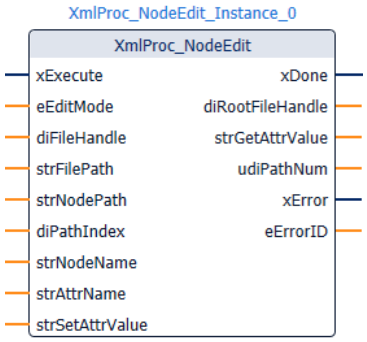
Item	LD	ST
Operation result		

- **Precautions**
  - The valid range for the namespace index "udiNsIndex" is [0, udiNsNum-1].
  - If a variable in the XML file belongs to a certain namespace, when accessing that variable, the corresponding namespace information should be added to the variable path.

18.2.12.1.6 XmlProc_NodeEdit

Edits nodes in specified XML file.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
XmlProc_NodeEdit	Edit nodes in the specified XML file	FB		XmlProc_NodeEdit_Instance_0(xExecute := , eEditMode := , diFileHandle := , strFilePath := , strNodePath := , diPathIndex := , strNodeName := , strAttrName := , strSetAttrValue := , xDone => , diRootFileHandle => , strGetAttrValue => , udiPathNum => , xError => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Trigger enable bit	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	eEditMode	Edit mode	XML_NODE_EDIT_ACTION	-	NO_ACTION	Edit mode of the node.
	diFileHandle	File handle	DINT	[-1, 1000]	0	File handle.
	strFilePath	File path	STRING[128]	-	-	File path.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	strNodePath	Node path	STRING[256]	-	-	Node path.
	diPathIndex	Path index	DINT	[0, udiPathNum -1]	0	Index of the path, starting from 0.
	strNodeName	Node name	STRING[64]	-	-	Node name.
	strAttrName	Property	STRING[64]	-	-	Property name.
	strSetAttrValue	Set property value	STRING[256]	-	-	Set property value.
Output	xDone	Completion flag	BOOL	TRUE, FALSE	FALSE	Completion flag.
	diRootFileHandle	File handle of the root node	DINT	-	0	File handle of the root node.
	strGetAttrValue	Retrieved property value	STRING[256]	-	-	Retrieved property value.
	udiPathNum	Number of matching node paths	UDINT	-	0	Number of paths.
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Error flag.
	eErrorID	Fault code	XML_PROC_ERROR_ID	-	NO_ERROR	See 18.2.12.1.8 Fault Codes.

- **Function**
  - Before instruction execution, input the edit mode "eEditMode". The input/output parameters under different edit modes are as follows:

Key Parameter	Add Root Node	Delete Root Node	Add Child Node	Delete Child Node	Add Property	Delete Property	Read Property	Write Property
xExecute	Valid	Valid	Valid	Valid	Valid	Valid	Valid	Valid
eEditMode	1	2	3	4	5	6	7	8
diFileHandle	-	Valid	Valid	Valid	Valid	Valid	Valid	Valid
strFilePath	Valid	-	-	-	-	-	-	-
strNodePath	-	-	Valid	Valid	Valid	Valid	Valid	Valid
diPathIndex	-	-	Valid	Valid	Valid	Valid	Valid	Valid
strNodeName	Valid	-	Valid	Valid	-	-	-	-

Key Parameter	Add Root Node	Delete Root Node	Add Child Node	Delete Child Node	Add Property	Delete Property	Read Property	Write Property
strAttrName	-	-	-	-	Valid	Valid	Valid	Valid
strSetAttrValue	-	-	-	-	Valid	-	-	Valid
xDone	Valid	Valid	Valid	Valid	Valid	Valid	Valid	Valid
diRootFileHandle	Valid	-	-	-	-	-	-	-
strGetAttrValue	-	-	-	-	-	-	Valid	-
udiPathNum	-	-	Valid	Valid	Valid	Valid	Valid	Valid
xError	Valid	Valid	Valid	Valid	Valid	Valid	Valid	Valid
eErrorID	Valid	Valid	Valid	Valid	Valid	Valid	Valid	Valid

◦ After instruction execution, check "xDone", "xError", and "eErrorID" to view the result of the edit action.

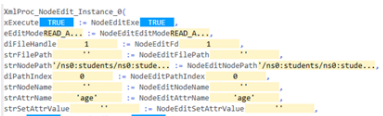
◦ The data type of "eEditMode" is XML_NODE_EDIT_ACTION. The enumeration values are defined as follows:

Variable	Value	Description
NO_ACTION	16#0000	No action.
ADD_ROOT_NODE	16#0001	Add a root node.
DELETE_ROOT_NODE	16#0002	Delete a root node.
ADD_CHILD_NODE	16#0003	Add a child node.
DELETE_CHILD_NODE	16#0004	Delete a child node.
ADD_ATTR	16#0005	Add a property within the node.
DELETE_ATTR	16#0006	Delete a property within the node.
READ_ATTR	16#0007	Read a property within the node.
WRITE_ATTR	16#0008	Write a property within the node.

- **Program example**

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
XmlProc_NodeEdit_Instance_0: XmlProc_NodeEdit;
NodeEditExe          : BOOL := FALSE;
NodeEditEditMode     : XML_NODE_EDIT_ACTION;
NodeEditFd           : DINT := 0;
NodeEditFilePath     : STRING[128];
NodeEditNodePath     : STRING[256]:= '/ns0:students/ns0:student';;
NodeEditPathIndex    : DINT := 0;
NodeEditNodeName     : STRING[64];
NodeEditAttrName     : STRING[64];
NodeEditSetAttrValue: STRING[256];
NodeEditDone         : BOOL := FALSE;
NodeEditRootFileHandle: DINT := 0;
NodeEditGetAttrValue: STRING[256];
NodeEditPathNum      : UDINT := 0;
NodeEditErr          : BOOL := FALSE;
NodeEditErrID        : XML_PROC_ERROR_ID := XML_PROC_ERROR_ID.NO_ERROR;
```

</td></tr>
<tr><td>Program</td><td>
</td><td></td></tr>
</table>

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Operation result</td><td colspan="2">

The XML file does not contain a namespace.

File example

test1.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<students>
    <student id="001" name="zhangsan" age="18" gender="male" ></student>
    <student id="002" name="lisi" age="17" gender="famale" ></student>
    <student id="003" name="lilei" age="18" gender="male" ></student>
</students>
```


Read property values

XmlProc_NodeEdit_Instance_0	XmlProc_NodeEdit		
NodeEditExe	BOOL		TRUE
NodeEditEditMode	XML_NODE_EDIT_ACTION		READ_ATTR
NodeEditFd	DINT		1
NodeEditFilePath	STRING[128]		''
NodeEditNodePath	STRING[256]		'/students/student'
NodeEditPathIndex	DINT		0
NodeEditNodeName	STRING[64]		''
NodeEditAttrName	STRING[64]		'age'
NodeEditSetAttrValue	STRING[256]		''
NodeEditDone	BOOL		TRUE
NodeEditRootFileHandle	DINT		-1
NodeEditGetAttrValue	STRING[256]		'18'
NodeEditPathNum	UDINT		3
NodeEditErr	BOOL		FALSE
NodeEditErrID	XML_PROC_ERROR_ID		NO_ERROR

The XML file contains a namespace.

File example

test2.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<students xmlns="http://www.school.com/class1">
    <student id="001" name="zhangsan" age="18" gender="male" ></student>
    <student id="002" name="lisi" age="17" gender="famale" ></student>
    <student id="003" name="lilei" age="18" gender="male"></student>
</students>
```


Read property values

XmlProc_NodeEdit_Instance_0	XmlProc_NodeEdit		
NodeEditExe	BOOL		TRUE
NodeEditEditMode	XML_NODE_EDIT_ACTION		READ_ATTR
NodeEditFd	DINT		1
NodeEditFilePath	STRING[128]		''
NodeEditNodePath	STRING[256]		'/ns0:students/ns0:student'
NodeEditPathIndex	DINT		0
NodeEditNodeName	STRING[64]		''
NodeEditAttrName	STRING[64]		'age'
NodeEditSetAttrValue	STRING[256]		''
NodeEditDone	BOOL		TRUE
NodeEditRootFileHandle	DINT		-1
NodeEditGetAttrValue	STRING[256]		'18'
NodeEditPathNum	UDINT		3
NodeEditErr	BOOL		FALSE
NodeEditErrID	XML_PROC_ERROR_ID		NO_ERROR

</td></tr>
</table>

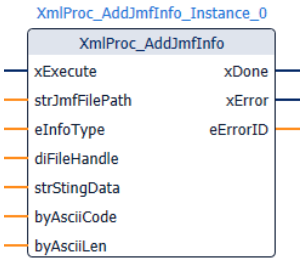
- **Precautions**
  - Different edit modes have different valid input and output parameters. Invalid parameters have no effect and may be set to 0 or empty.
  - If a node in the XML file belongs to a namespace, the node path should include the corresponding namespace information.

- In an XML file, "strNodePath" may match multiple paths. Different paths are distinguished by "diPathIndex".
- The value range of "diPathIndex" is [0, udiPathNum-1].

18.2.12.1.7 XmlProc_AddJmfInfo

Adds information to a JMF file.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
XmlProc_AddJmfInfo	Add information to a JMF file	FB		XmlProc_AddJmfInfo_Instance_0(xExecute := , strJmfFilePath := , eInfoType := , diFileHandle := , strStingData := , byAsciiCode := , byAsciiLen := , xDone => , xError => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Trigger enable bit	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	strJmfFilePath	JMF file path	STRING[128]	-	-	Path to the JMF file.
	eInfoType	Information type to add	XML_ADD_INFO_TYPE	-	-	Information type to add.
	diFileHandle	File handle	DINT	-	0	Handle of the content to add.
	strStingData	String information to add	STRING[128]	-	-	String content to add.
	byAsciiCode	ASCII code value	ARRAY [0..15] OF BYTE	-	0	ASCII code value. For example, the ASCII code for a line feed is 10.
	byAsciiLen	Valid data length of ASCII code	BYTE	[0, 255]	0	Valid data length of ASCII code.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	xDone	Completion flag	BOOL	TRUE, FALSE	FALSE	Completion flag.
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Error flag.
	eErrorID	Fault code	XML_PROC_ERROR_ID	-	NO_ERROR	See 18.2.12.1.8 Fault Codes.

- **Function**
  - Before instruction execution, input the JMF file path "strJmfFilePath" and the information type "eInfoType". The valid input parameters for different information type modes are shown in the following table:

Key Parameter	Handle Content to Add	String to Add	ASCII Code to Add
eInfoType	0	1	2
strJmfFilePath	Valid	Valid	Valid
diFileHandle	Valid	-	-
strStingData	-	Valid	-
byAsciiCode	-	-	Valid
byAsciiLen	-	-	Valid

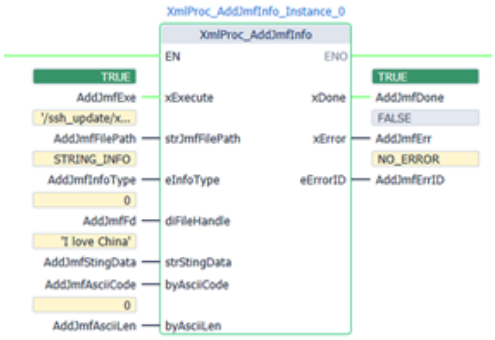
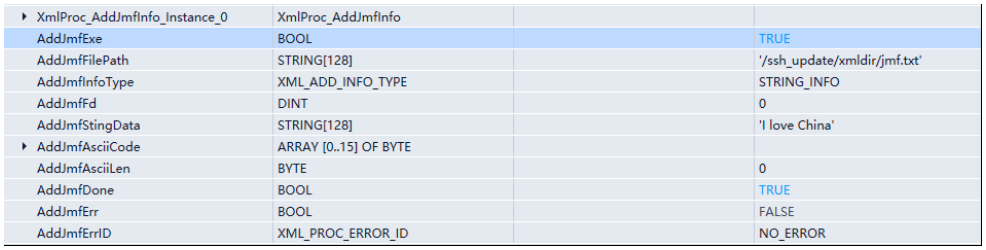
  - After the instruction execution, the result is returned. If "xDone" is TRUE, the operation is successful.
  - The data type of "eInfoType" is XML_ADD_INFO_TYPE. The enumeration values are defined as follows:

Variable	Value	Description
HANDLE_INFO	16#0000	Add the content in the handle to the JMF file.
STRING_INFO	16#0001	Add the string information directly to the JMF file.
ASCII_INFO	16#0002	Add the character corresponding to the ASCII code to the JMF file.

- **Program example**

Item	LD	ST
Define variables		

```
XmlProc_AddJmfInfo_Instance_0: XmlProc_AddJmfInfo;
AddJmfExe           : BOOL := FALSE;
AddJmfFilePath      : STRING[128] := '/ssh_update/xmldir/jmf.txt';;
AddJmfInfoType      : XML_ADD_INFO_TYPE;
AddJmfFd            : DINT := 0;
AddJmfStingData     : STRING[128];
AddJmfAsciiCode     : ARRAY[0..15] OF BYTE;
AddJmfAsciiLen      : BYTE;
AddJmfDone          : BOOL := FALSE;
AddJmfErr           : BOOL := FALSE;
AddJmfErrID         : XML_PROC_ERROR_ID := XML_PROC_ERROR_ID.NO_ERROR;
```

Item	LD	ST
Program		
Operation result		

XmlProc_AddJmfInfo_Instance_0	XmlProc_AddJmfInfo		
AddJmfExe	BOOL		TRUE
AddJmfFilePath	STRING[128]		'/ssh_update/xmldir/jmf.txt'
AddJmfInfoType	XML_ADD_INFO_TYPE		STRING_INFO
AddJmfFd	DINT		0
AddJmfStingData	STRING[128]		'I love China'
AddJmfAsciiCode	ARRAY [0..15] OF BYTE		
AddJmfAsciiLen	BYTE		0
AddJmfDone	BOOL		TRUE
AddJmfErr	BOOL		FALSE
AddJmfErrID	XML_PROC_ERROR_ID		NO_ERROR

- **Precautions**
  - Different edit modes have different valid input and output parameters. Invalid parameters have no effect and may be set to 0 or empty.
  - When "eInfoType" is HANDLE_INFO, the content of the file corresponding to "diFileHandle" is added to the file at the path "strJmfFilePath".
  - When "eInfoType" is STRING_INFO, the specified string can be added to the file.
  - When "eInfoType" is ASCII_INFO, specific control characters can be added to the file at the path "strJmfFilePath".

18.2.12.1.8 Fault Codes

Variable	Value	Description
NO_ERROR	16#0000	No fault
FILE_NOT_EXIST	16#0001	File does not exist.
FILE_NAME_FORMAT_ERROR	16#0002	Filename format error.
NO_OPERATION_PERMISSION	16#0003	No operation permission for the file.
FILE_PARSE_FAILURE	16#0004	File parsing failed.
OPENED_FILE_NUM_OVER_LIMIT	16#0005	The number of open files exceeds the limit.
FILE_HANDLE_NOT_EXIST	16#0006	File handle does not exist.
FILE_CREATE_FAILURE	16#0007	File creation failed.
FILE_DELETE_FAILURE	16#0008	File deletion failed.
PARAMETER_FORMAT_ERROR	16#0009	Input parameter format error.
PARAMETER_OVER_LIMIT	16#000A	Input parameter exceeds the limit.
PARAMETER_ERROR	16#000B	Input parameter error.

Variable	Value	Description
OPENED_FILE_SIZE_SUM_OVER_LIMIT	16#000C	The total size of the cumulatively opened files exceeds the limit.
FILE_SIZE_EXCEEDS_LIMIT	16#000D	File size exceeds the limit.
PATH_INDEX_ERROR	16#0100	Path index error.
NODE_NO_FOUND	16#0101	The corresponding node not found in the file.
ATTR_NO_FOUND	16#0102	The corresponding property not found in the node.
PATH_INDEX_OVER_LIMIT	16#0103	Path index exceeds the limit.
VALUE_LEN_OVER_LIMIT	16#0104	Data length exceeds the limit.
ATTR_NAME_CONFLICT	16#0105	Property name conflict.
NAMESPACE_NO_FOUND	16#0106	Namespace not found in the file.
UNKNOWN	16#FFFF	Unknown error

18.2.13 Analog Signal

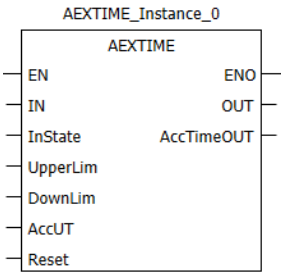
18.2.13.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.13.2 AEXTIME	FB	Analog over-limit time accumulation	SignalProcessing	Library version 1.0.0.0 and later
18.2.13.3 CHDETECT	FB	Value change detection		Library version 1.0.0.0 and later
18.2.13.4 CHDETECT_T	FB	Time-limited value change detection		Library version 1.0.0.0 and later
18.2.13.5 FILTER	FB	Analog filtering		Library version 1.0.0.0 and later
18.2.13.6 FLOWACC	FB	Flow accumulation		Library version 1.0.0.0 and later
18.2.13.7 INCRSUM	FB	Incremental summator		Library version 1.0.0.0 and later
18.2.13.8 SWND	FB	Bumpless switchover		Library version 1.0.0.0 and later
18.2.13.9 THRSEL	FB	Three-analog value selection		Library version 1.0.0.0 and later
18.2.13.10 TWOSEL	FB	Two-analog value selection		Library version 1.0.0.0 and later

18.2.13.2 AEXTIME

AEXTIME is used for analog over-limit time accumulation.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AEXTIME	Analog over-limit time accumulation	FB		AEXTIME_Instance_0(IN := , InState := , UpperLim := , DownLim := , AccUT := , Reset := , OUT => , AccTimeOUT =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input	Follows data type	-	Input
	InState	Input state	Follows data type	TRUE	TRUR: Input is of good quality. FALSE: Input is of bad quality.
	UpperLim	Input upper limit	Follows data type	100.0	Input upper limit
	DownLim	Input lower limit	Follows data type	0.0	Input lower limit
	AccUT	Accumulation unit	Follows data type	-	0: Second 1: Minute 2: Hour 3: Day
	Reset	Reset a bit	Follows data type	-	Reset

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	OUT	Accumulated time output (in specified unit)	Follows data type	-	Accumulated time output (in specified unit)
	AccTimeOUT	Total accumulated time output (in second)	Follows data type	-	Total accumulated time output (in second)

Variable types

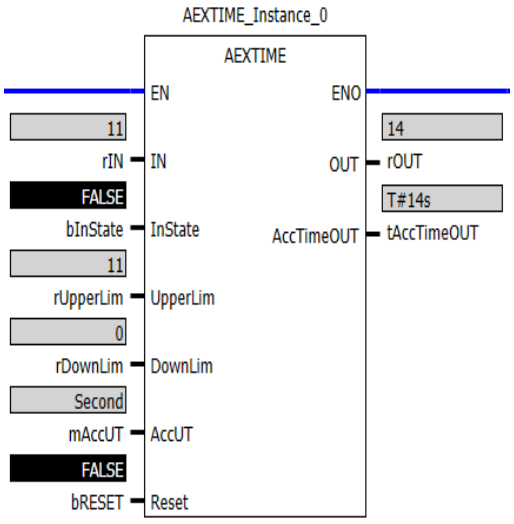
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
InState	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
UpperLim	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
DownLim	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
AccUT	Enumeration type: Acc_UT (0: Second; 1: Minute; 2: Hour; 3: Day)																			
Reset	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
AccTimeOUT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–

- **Function**
  - Accumulates the time during which the analog input exceeds the upper or lower limit, or when the input signal is of bad quality.
  - When the input "IN" is of bad quality or exceeds upper/lower limits, time accumulation begins, and the result is output to "OUT" (in the unit specified by AccUT) and "AccTimeOUT" (in seconds).
- **Program example**

  The following example sets "IN" and UpperLim to REAL#11.0, the results after 14 executions are shown in the table below.

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

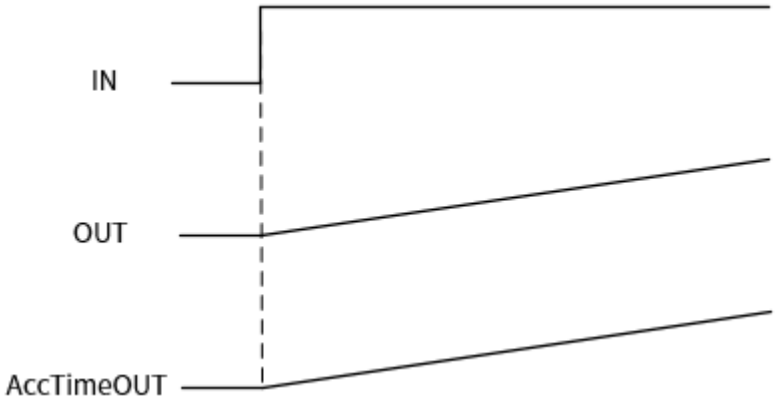
```
VAR
    AEXTIME_Instance_0: AEXTIME;
    rIN: REAL := 11;
    bInState: BOOL := FALSE;
    rUpperLim: REAL := 11;
    rDownLim: REAL := 0;
    mAccUT: ACC_UT := ACC_UT.Second;
    bRESET: BOOL := FALSE;
    rOUT: REAL;
    tAccTimeOUT: TIME;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>
</td><td>

```
AEXTIME_Instance_0(
IN := 11 := rIN 11,
InState FALSE := bInState FALSE,
UpperLim 11 := rUpperLim 11,
DownLim 0 := rDownLim 0,
AccUT Second := mAccUT Second,
Reset FALSE := bReset FALSE,
OUT 14 => rOUT 14,
AccTimeOUT T#14s => tAccTimeOUT T#14s);
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

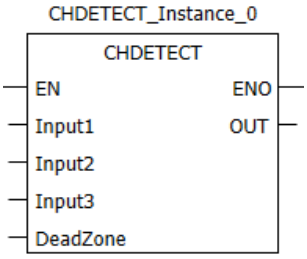
		Name	Data type	Address	Value
1		AEXTIME_Instance_0	AEXTIME		
2		rIN	REAL		11
3		bInState	BOOL		FALSE
4		rUpperLim	REAL		11
5		rDownLim	REAL		0
6		mAccUT	ACC_UT		Second
7		bRESET	BOOL		FALSE
8		rOUT	REAL		14
9		tAccTimeOUT	TIME		T#14s

</td></tr>
<tr><td>Operation result</td><td colspan="2">
</td></tr>
</table>

18.2.13.3 CHDETECT

Detects changes in analog input values.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CHDETECT	Value change detection	FB		CHDETECT_Instance_0(Input1 := , Input2 := , Input3 := , DeadZone := , OUT =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Input1	Input 1	Follows data type	0.0	Input 1
	Input2	Input 2	Follows data type	0.0	Input 2
	Input3	Input 3	Follows data type	0.0	Input 3
	DeadZone	Input dead zone width	Follows data type	0.0	Changes within the dead zone are not detected; DeadZone > 0.
Output	OUT	A flag indicating whether the input changes.	Follows data type	FALSE	A flag indicating whether the input changes.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Input1	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Input2	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Input3	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
DeadZone	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
OUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Detects whether any of the three floating-point input values have changed. A dead zone can be specified, values within which are considered unchanged.

  Check method: $OUT = (|Input1 - lastInput1| \geq DeadZone)\ OR\ (|Input2 - lastInput2| \geq DeadZone)\ OR\ (|Input3 - lastInput3| \geq DeadZone)$

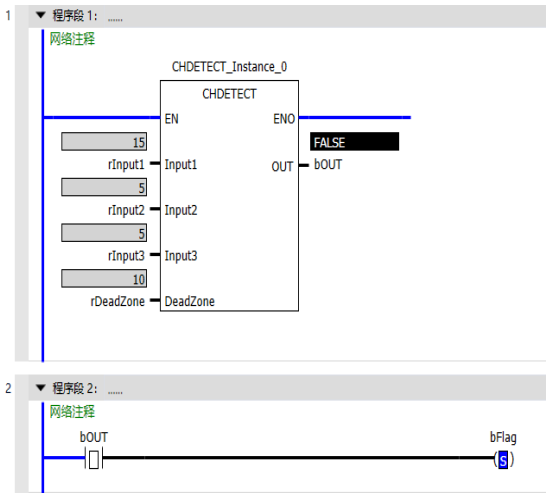
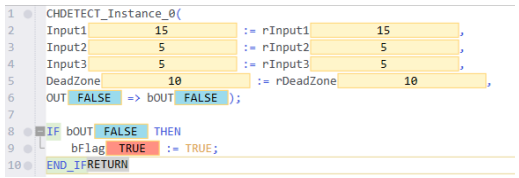
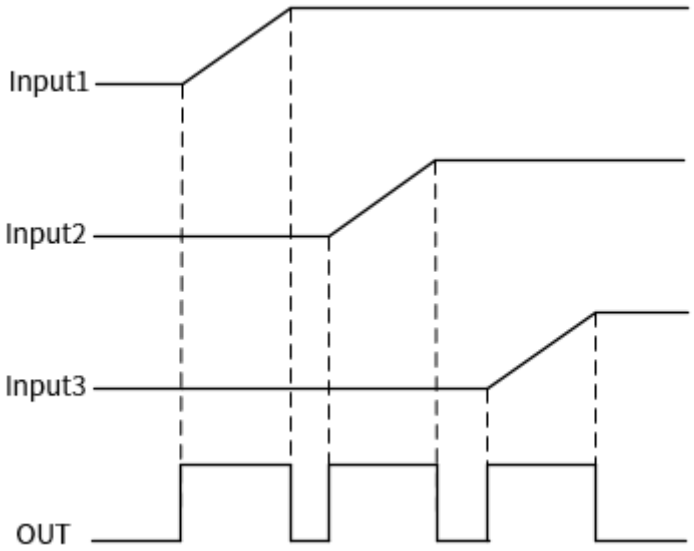
  Input - LastInput is the value from the previous cycle.

- **Program example**

  The following example sets the deadband width "DeadZone" to 10. When "Input1" increases from 5 to 15, the output "OUT" changes from FALSE to TRUE. Use variable "bFlag" to detect the change in "OUT".

Item	LD	ST

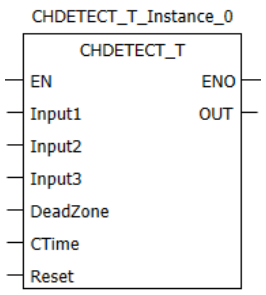
Define variables	VAR
	CHDETECT_Instance_0: CHDETECT;
	rInput1: REAL := 5;
	rInput2: REAL := 5;
	rInput3: REAL := 5;
	rDeadZone: REAL := 10;
	bOUT: BOOL;
	bFlag: BOOL := FALSE;
	END_VAR	

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>
</td><td>
</td></tr>
<tr><td>Operation result</td><td colspan="2"><table>
<tr><th></th><th></th><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td>1</td><td></td><td>CHDETECT_Instance_0</td><td>CHDETECT</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td>rInput1</td><td>REAL</td><td></td><td>15</td></tr>
<tr><td>3</td><td></td><td>rInput2</td><td>REAL</td><td></td><td>5</td></tr>
<tr><td>4</td><td></td><td>rInput3</td><td>REAL</td><td></td><td>5</td></tr>
<tr><td>5</td><td></td><td>rDeadZone</td><td>REAL</td><td></td><td>10</td></tr>
<tr><td>6</td><td></td><td>bOUT</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>7</td><td></td><td>bFlag</td><td>BOOL</td><td></td><td>TRUE</td></tr>
</table></td></tr>
<tr><td>Operation result</td><td colspan="2">
</td></tr>
</table>

18.2.13.4 CHDETECT_T

Detects whether the duration of analog value change exceeds the limit.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CHDETECT_T	Time-limited value change detection	FB		CHDETECT_T_Instance_0(Input1 := , Input2 := , Input3 := , DeadZone := , CTime := , Reset := , OUT =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Input1	Input 1	Follows data type	0.0	Input 1
	Input2	Input 2	Follows data type	0.0	Input 2
	Input3	Input 3	Follows data type	0.0	Input 3
	DeadZone	Input dead zone width	Follows data type	0.0	Changes within the dead zone are not detected; DeadZone > 0.
	CTime	Change duration limit	Follows data type	TIME#0ms	Change duration limit
	Reset	Timer reset	Follows data type	FALSE	Timer reset
Output	OUT	A flag indicating whether the input changes.	Follows data type	FALSE	A flag indicating whether the input changes.

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Input1	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Input2	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Input3	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
DeadZone	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
CTime	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
Reset	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Detects whether the duration of change in any of the three inputs exceeds the set time; if so, the change flag is triggered. A dead zone can also be specified, values within which are considered unchanged.

  Supports timer reset function.

  Check method: $(|Input1 - lastInput1| \geq DeadZone)\ OR\ (|Input2 - lastInput2| \geq DeadZone)\ OR\ (|Input3 - lastInput3| \geq DeadZone)$

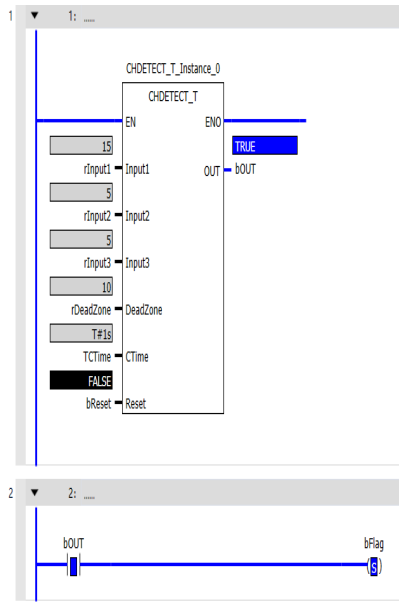
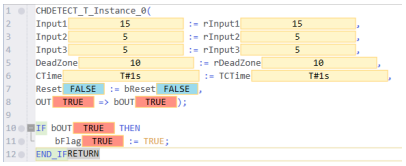
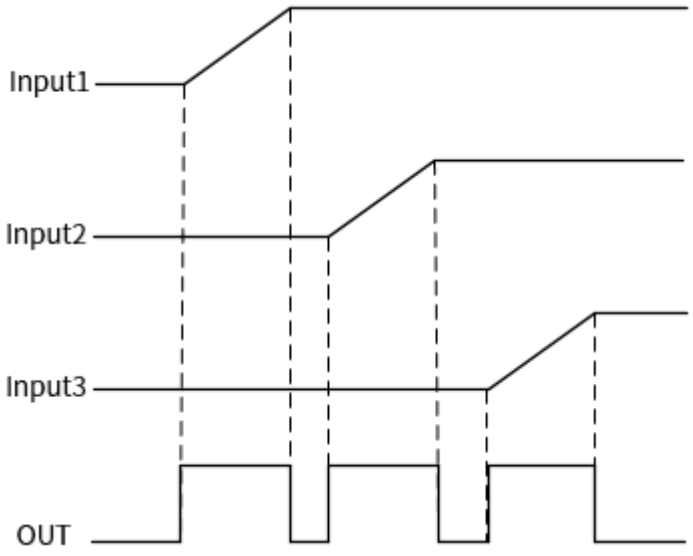
  If the duration of change in any input is greater than or equal to "CTime", output "OUT" becomes TRUE. Input - LastInput is the value from the previous cycle.

- **Program example**

  The following example sets the deadband width "DeadZone" to 10. When "Input1" increases from 5 to 15, the output "OUT" changes from FALSE to TRUE. Use variable "bFlag" to detect the change in "OUT".

Item	LD	ST

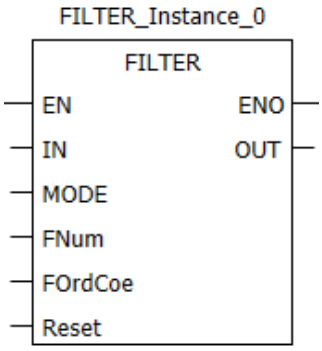
Define variables	VAR
	CHDETECT_T_Instance_0: CHDETECT_T;
	rInput1: REAL := 5;
	rInput2: REAL := 5;
	rInput3: REAL := 5;
	rDeadZone: REAL := 10;
	TCTime: TIME := T#1s;
	bReset: BOOL;
	bOUT: BOOL;
	bFlag: BOOL := FALSE;
	END_VAR	

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>
</td><td>
</td></tr>
<tr><td>Operation result</td><td colspan="2"><table>
<tr><th></th><th></th><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td>1</td><td></td><td>CHDETECT_T_Instance_0</td><td>CHDETECT_T</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td>rInput1</td><td>REAL</td><td></td><td>15</td></tr>
<tr><td>3</td><td></td><td>rInput2</td><td>REAL</td><td></td><td>5</td></tr>
<tr><td>4</td><td></td><td>rInput3</td><td>REAL</td><td></td><td>5</td></tr>
<tr><td>5</td><td></td><td>rDeadZone</td><td>REAL</td><td></td><td>10</td></tr>
<tr><td>6</td><td></td><td>TCTime</td><td>TIME</td><td></td><td>T#1s</td></tr>
<tr><td>7</td><td></td><td>bReset</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>8</td><td></td><td>bOUT</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>9</td><td></td><td>bFlag</td><td>BOOL</td><td></td><td>TRUE</td></tr>
</table></td></tr>
<tr><td>Operation result</td><td colspan="2">
</td></tr>
</table>

18.2.13.5 FILTER

FILTER is used for analog value filtering.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FILTER	Analog filtering	FB		FILTER_Instance_0(IN := , MODE := , FNum := , FOrdCoe := , Reset := , OUT =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Analog input	Follows data type	-	Analog input
	MODE	Filter mode	Follows data type	-	• average (0): Average filtering. • median (1): Median filtering. • first_order (2): First-order linear filtering.
	FNum	Filter cycle count	20 >= FNum >= 2	2	Filter cycle count
	FOrdCoe	Sampled value weight	Follows data type	1	Effective when MODE is set to first_order (2); 1 >= FOrdCoe >= 0; the closer to 1, the greater the weight of the new sampled value.
	Reset	Reset a bit	Follows data type	-	Reset
Output	OUT	Filter output	Follows data type	-	Filter output

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
MODE	Enumeration type FILTER_MODE																			
FNum	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
FOrdCoe	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Reset	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

- **Function**

  Provides the following three filtering functions, which can be set via MODE.

  - **Average filtering (MODE = 0)**
    - This algorithm calculates the average of the sampled values over FNum cycles; the algorithm is relatively simple. When FNum is large, the filtered signal is smoother, but with poorer sensitivity; conversely, when FNum is small, the filtering is less smooth, but with greater sensitivity.
    - Due to software memory limitations, 20 cycles are recorded by default.
  - **Median filtering (MODE = 1)**
    - This algorithm collects data over FNum cycles, sorts the values, removes the top 1/3 and bottom 1/3 of the values, and averages the remaining data. This mode is suitable for systems where occasional abnormal values may occur.
    - The selected value range depends on FNum as follows:

      3n: Selected range is n-1 to 2n-1 (average of n values)

      3n+1: Selected range is n to 2n (average of n+1 values)

      3n+2: Selected range is n to 2n+1 (average of n+2 values)
  - **First-order filtering (MODE = 2)**

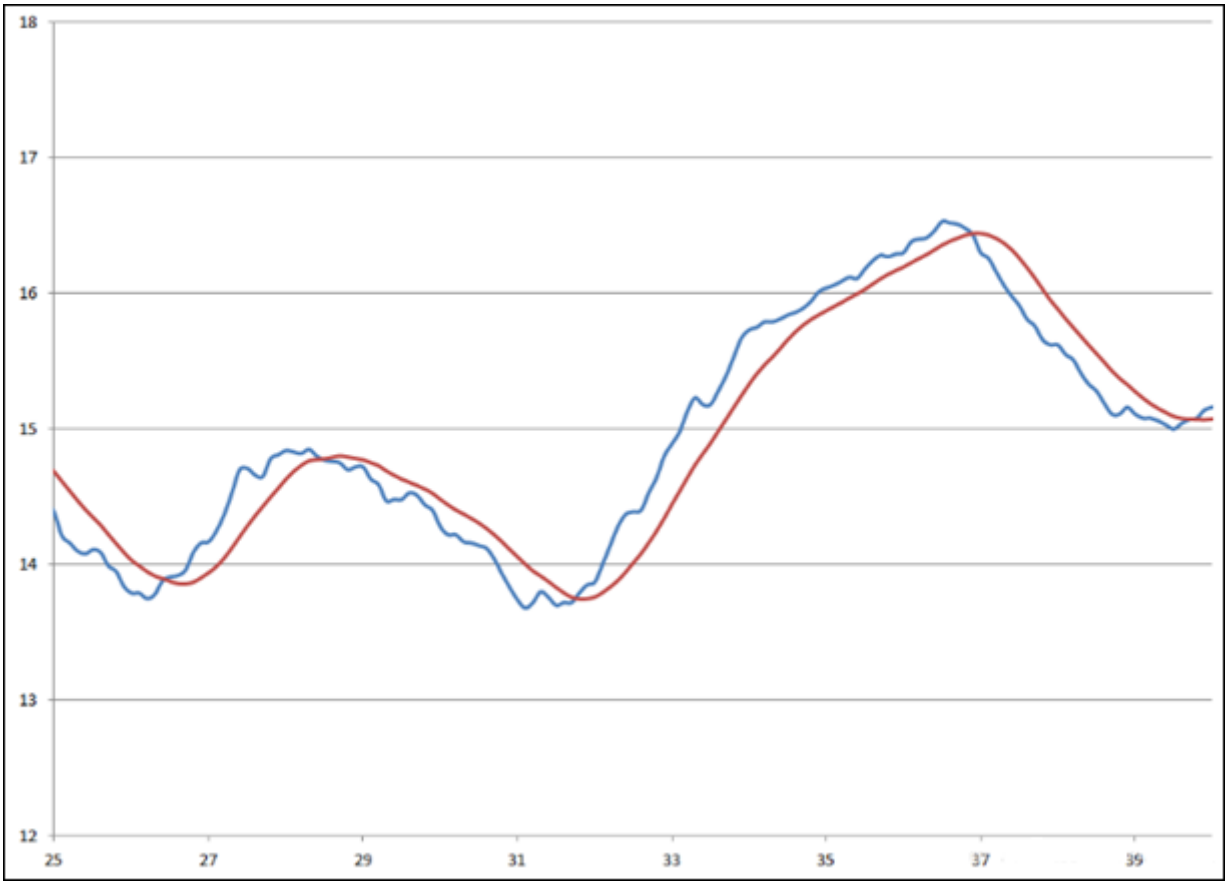
$$Y(n) = FOrdCoe * X(n) + (1 - FOrdCoe) * Y(n-1)$$

(Filter coefficient FOrdCoe ranges from 0 to 1)

- The first-order filter calculates a weighted average between the new sample and the previous filter output. The value of FOrdCoe determines the sensitivity of the algorithm. The larger the FOrdCoe, the greater the weight of the new sample, resulting in higher sensitivity but poorer smoothness; conversely, the smaller the FOrdCoe, the smaller the weight of the new sample, resulting in lower sensitivity but better smoothness.
- The filter also supports a reset function to restart filtering. Reset is level-triggered: when Reset transitions from TRUE to FALSE, the filter restarts.

- Performance requirements: Performance specifications and requirements associated for this function.

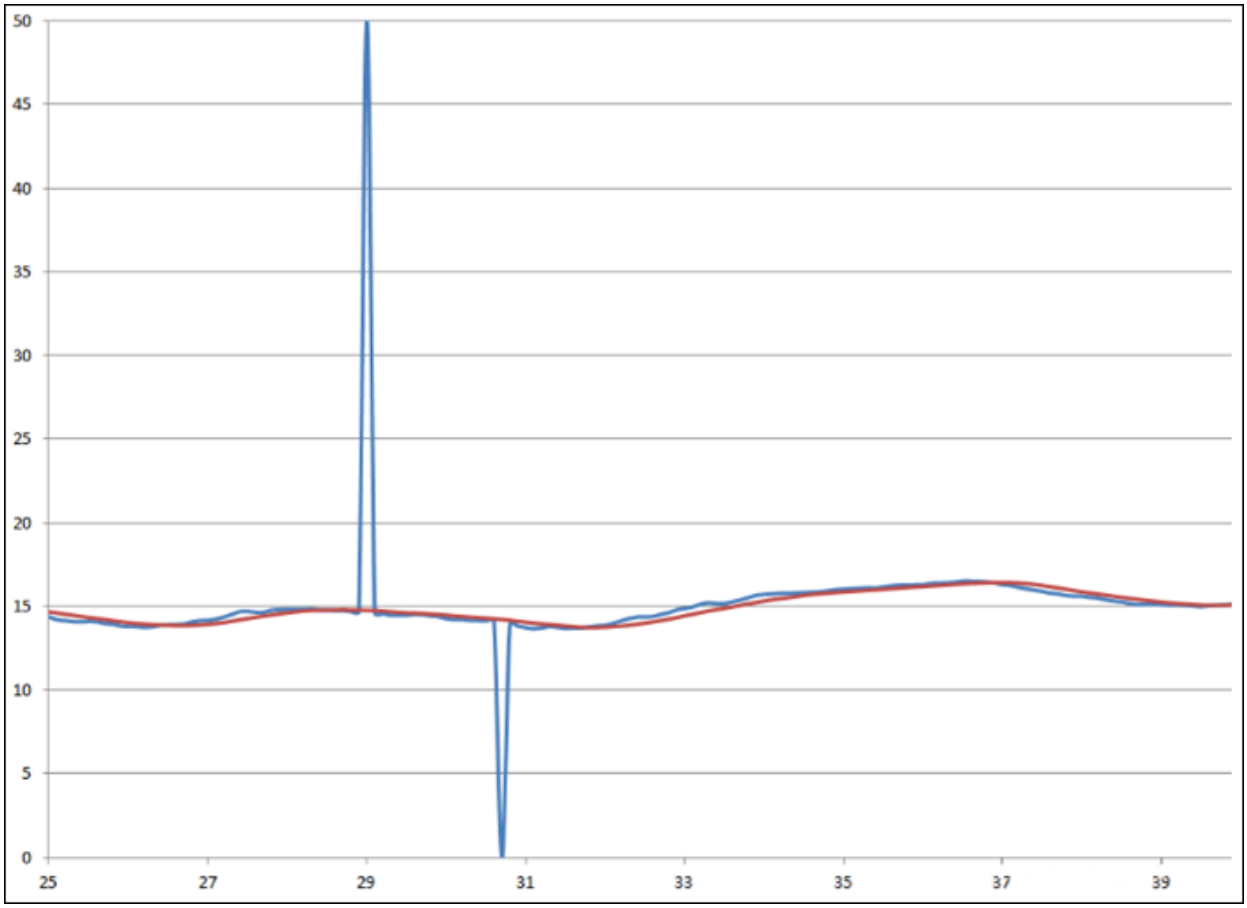
◦ The average filter greatly smooths the data, but introduces data lag.



Blue - before filtering

Red - after filtering

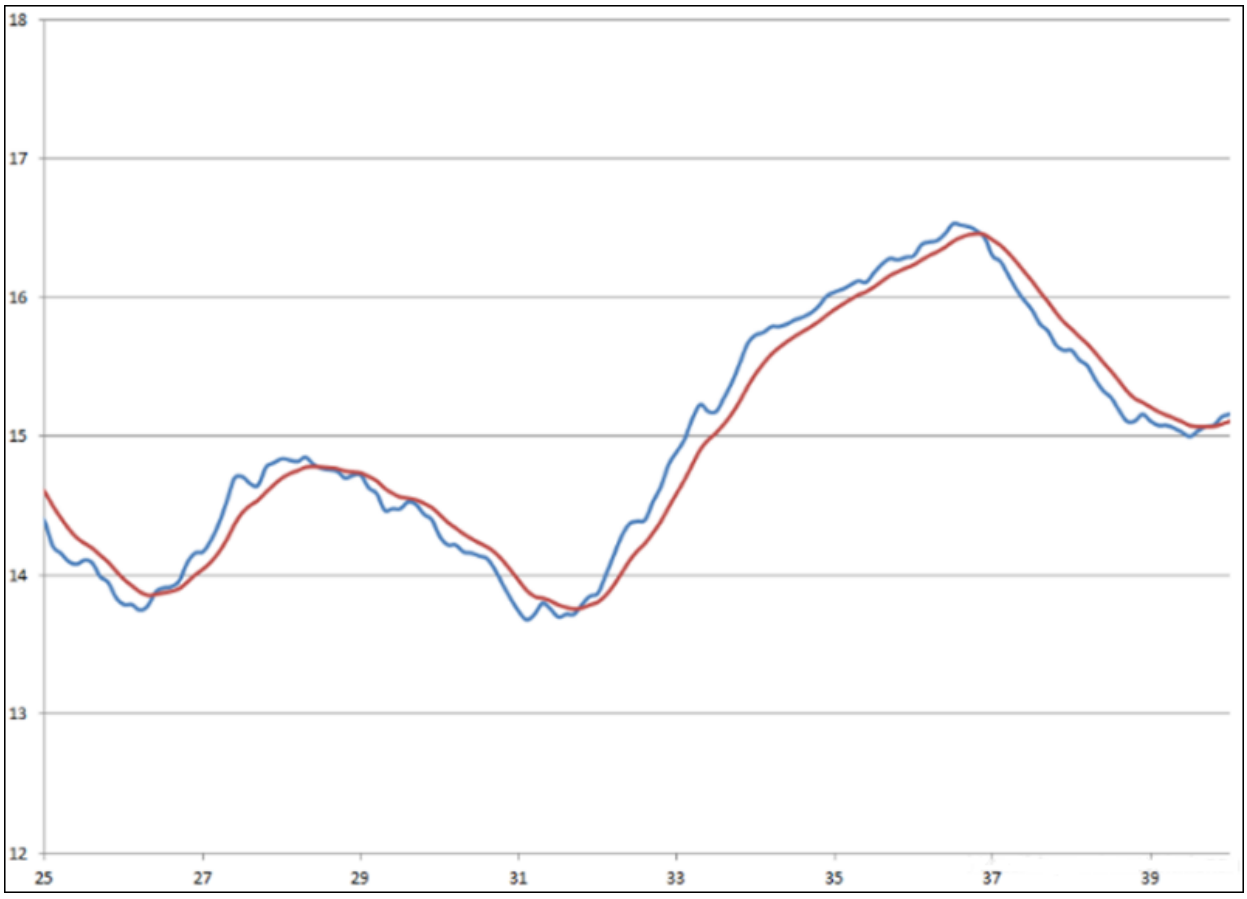
◦ When the original data contains two abnormal values, they may be caused by data interference during sampling or abnormal data processing. Median filtering can be used to effectively eliminate the impact caused by such abnormal values.



Blue - before filtering

Red - after filtering

◦ The first-order filtering smooths the original signal, but introduces lag.



Blue - before filtering

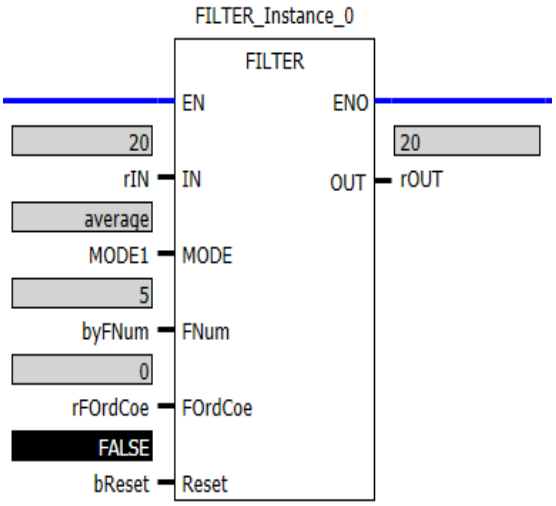
Red - after filtering

- **Program example**

  Taking average filtering as an example, the input signal rIN increases by 1 each scheduling, with the filtering period set to 5. The results are shown in the table below. It can be seen that the output signal of the average filtering exhibits a certain degree of lag.

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

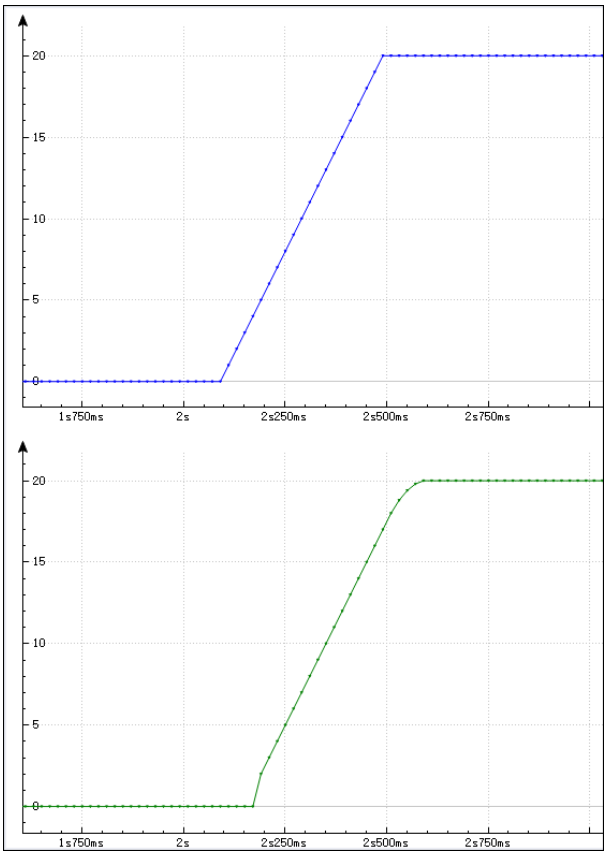
```
VAR
    FILTER_Instance_0: FILTER;
    rIN: REAL := 0;
    MODE1: FILTER_MODE := FILTER_MODE.average;
    byFNum: BYTE := 5;
    rFOrdCoe: REAL;
    bReset: BOOL;
    rOUT: REAL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>


</td><td>

```
FILTER_Instance_0(
IN := 20 := rIN 20,
MODE average := MODE1 average,
FNum 5 := byFNum 5,
FOrdCoe 0 := rFOrdCoe 0,
Reset FALSE := bReset FALSE,
OUT 20 => rOUT 20);
```

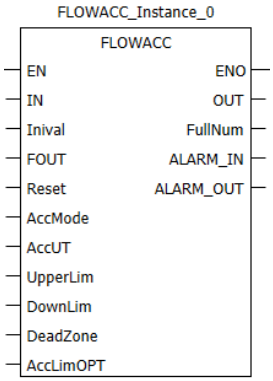
</td></tr>
</table>

Item	LD	ST
Operation result		

18.2.13.6 FLOWACC

FLOWACC is used for flow accumulation.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FLOWACC	Flow accumulation	FB		FLOWACC_Instance_0(IN := , Inival := , FOUT := , Reset := , AccMode := , AccUT := , UpperLim := , DownLim := , DeadZone := , AccLimOPT := , OUT => , FullNum => , ALARM_IN => , ALARM_OUT =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Flow input	Follows data type	-	Flow input
	Inival	Initial accumulated value	Follows data type	0.0	Initial accumulated value
	FOUT	Accumulation completion value	Follows data type	3.4E+38	Accumulation completion value
	Reset	Reset a bit	Follows data type	FALSE	Reset
	AccMode	Accumulation method	Follows data type	-	0: Full accumulation 1: Positive accumulation

Input/Output	Variable	Name	Valid Range	Default Value	Description
	AccUT	Accumulation unit	Follows data type	-	0: Second 1: Minute 2: Hour 3: Day
	UpperLim	Input upper limit	Follows data type	10000	Input upper limit
	DownLim	Input lower limit	Follows data type	0	Input lower limit
	DeadZone	Input dead zone width	Follows data type	0	Changes within the dead zone are not detected; DeadZone > 0.
	AccLimOPT	Accumulation limit mode	Follows data type	2: Continuous accumulation	0: Limited accumulation 1: Auto-reset accumulation 2: Continuous accumulation
Output	OUT	Accumulated output	Follows data type	-	Accumulated output
	FullNum	Number of times accumulation reaches FOUT	Follows data type	-	Number of times accumulation reaches FOUT
	ALARM_IN	Input over-limit alarm	Follows data type	-	1: Upper limit exceeded 2: Lower limit exceeded
	ALARM_OUT	Output over-limit alarm	Follows data type	-	1: Accumulation limit exceeded 2: Output exceeds maximum floating-point limit

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Inival	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
FOUT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Reset	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
AccMode	Enumeration type: FLOWACC_ACCMODE 0: All_ACC (Full accumulation) 1: Positive_ACC (Positive accumulation)																			
AccUT	Enumeration type: Acc_UT 0: Second 1: Minute 2: Hour 3: Day																			
UpperLim	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
DownLim	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
DeadZone	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
AccLimOPT	Custom type: FLOWACC_ACCLIMOPT 0: Acc_LIMIT (Accumulation limit) 1: Acc_AutoReset (Auto-reset accumulation) 2: Acc_Continue (Continuous accumulation)																			
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
FullNum	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
ALARM_IN	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ALARM_OUT	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  In many fields involving liquids or gases, it is necessary to obtain the instantaneous flow rate within the pipeline, and accurately measure the total flow volume over a specific period. This requirement can be met using the flow accumulation instruction.

  The specific functions are as follows:

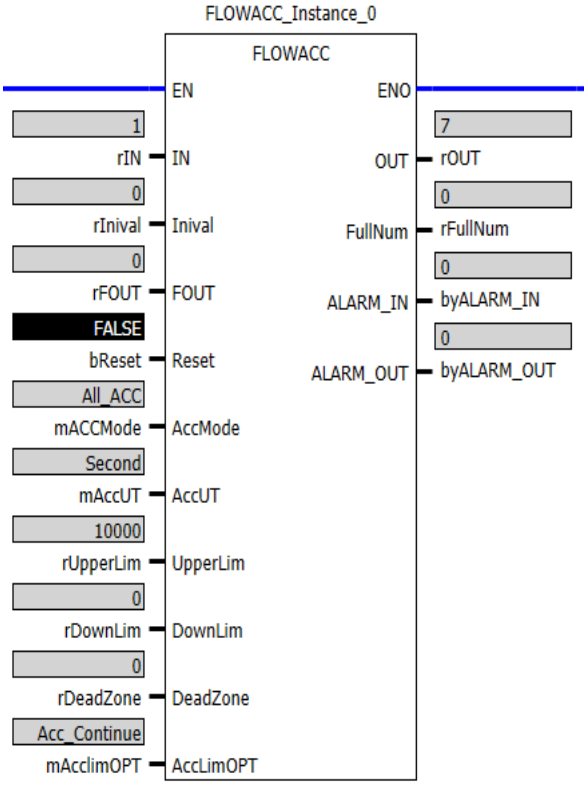
  - The initial value of flow accumulation can be set. After reset, accumulation will start again from the set initial value.
  - Input limit alarm can be set. When the input exceeds the upper limit, ALARM_IN outputs 1, and the current output remains unchanged. When the input exceeds the lower limit, ALARM_IN outputs 2, and the current output remains unchanged.
  - Flow accumulation mode and accumulation time unit can be set:
    - AccMode = 0: Full accumulation. Accumulation occurs regardless of input value.
    - AccMode = 1: Positive accumulation. Accumulation occurs only when the input is positive.
  - Flow accumulation dead zone can be set. Inputs within the dead zone are not accumulated.
  - The integral limit function can be set via the mode parameter (AccLimOPT):
    - When AccLimOPT = 0, the system is in integral limit mode. When the output (OUT) reaches the accumulation completion value (FOUT), it is limited to FOUT, and ALARM_OUT becomes 1.
    - When AccLimOPT = 1, the system is in automatic reset mode. When the output (OUT) reaches the accumulation completion value (FOUT), it automatically resets. FullNum is incremented by 1 upon each completed accumulation.
    - When AccLimOPT = 2 (default mode), the system is in continuous accumulation mode. After the output (OUT) reaches the accumulation completion value (FOUT), accumulation continues without stopping until manually reset. If the output exceeds the floating-point limit, ALARM_OUT becomes 2.

- **Program example**

  When the accumulation mode is set to continuous accumulation, and the flow input is 1, the system output after seven task scheduling cycles is shown in the table below.

Item	LD	ST

Define variables	VAR
	FLOWACC_Instance_0: FLOWACC;
	rIN: REAL := 1;
	rInival: REAL := 0;
	rFOUT: REAL := 0;
	bReset: BOOL := FALSE;
	mAccMode: FLOWACC_ACCMODE := FLOWACC_ACCMODE.All_ACC;
	mAccUT: Acc_UT := ACC_UT.Second;
	rUpperLim: REAL := 10000;
	rDownLim: REAL := 0;
	rDeadZone: REAL := 0;
	mAccLimOPT: FLOWACC_ACCLIMOPT := FLOWACC_ACCLIMOPT.Acc_Continue;
	rOUT: REAL;
	rFullNum: REAL;
	byALARM_IN: BYTE;
	byALARM_OUT: BYTE;
	END_VAR		

Program		FLOWACC_Instance_0(IN := rIN, Inival := rInival, FOUT := rFOUT, Reset := bReset, AccMode := mACCMode, AccUT := mAccUT, UpperLim := rUpperLim, DownLim := rDownLim, DeadZone := rDeadZone, AccLimOPT := mAcclimOPT, OUT => rOUT, FullNum => rFullNum, ALARM_IN => byALARM_IN, ALARM_OUT => byALARM_OUT);

Item	LD	ST

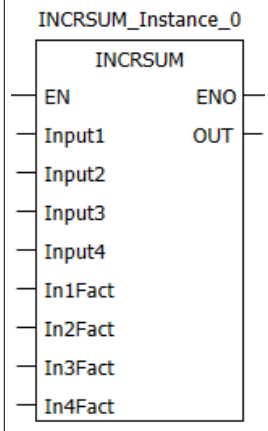
Operation result

	Name	Data type	Address	Value
1	FLOWACC_Instance_0	FLOWACC		
2	rIN	REAL		1
3	rInival	REAL		0
4	rFOUT	REAL		0
5	bReset	BOOL		FALSE
6	mACCMode	FLOWACC_ACCMODE		All_ACC
7	mAccUT	ACC_UT		Second
8	rUpperLim	REAL		10000
9	rDownLim	REAL		0
10	rDeadZone	REAL		0
11	mAcclimOPT	FLOWACC_ACCLIMOPT		Acc_Continue
12	rOUT	REAL		7
13	rFullNum	REAL		0
14	byALARM_IN	BYTE		0
15	byALARM_OUT	BYTE		0

18.2.13.7 INCRSUM

INCRSUM is used for incremental summator.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
INCRSUM	Incremental summator	FB		INCRSUM_Instance_0(Input1 := , Input2 := , Input3 := , Input4 := , In1Fact := , In2Fact := , In3Fact := , In4Fact := , OUT =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Input1	Input 1	Follows data type	0.0	Input 1
	Input2	Input 2	Follows data type	0.0	Input 2
	Input3	Input 3	Follows data type	0.0	Input 3

Input/Output	Variable	Name	Valid Range	Default Value	Description
	Input4	Input 4	Follows data type	0.0	Input 4
	In1Fact	Input 1 proportional coefficient	Follows data type	1.0	Input 1 proportional coefficient
	In2Fact	Input 2 proportional coefficient	Follows data type	1.0	Input 2 proportional coefficient
	In3Fact	Input 3 proportional coefficient	Follows data type	1.0	Input 3 proportional coefficient
	In4Fact	Input 4 proportional coefficient	Follows data type	1.0	Input 4 proportional coefficient
Output	OUT	Output	Follows data type	0.0	Output

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Input1	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Input2	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Input3	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Input4	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
In1Fact	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
In2Fact	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
In3Fact	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In4Fact	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

- **Function**

  Calculates the incremental change of up to four input values. Each input's change is multiplied by its respective proportional coefficient, then the sum of all input changes is calculated.

$$OUT = \sum \text{Inputx Change} * \text{InxFact}$$

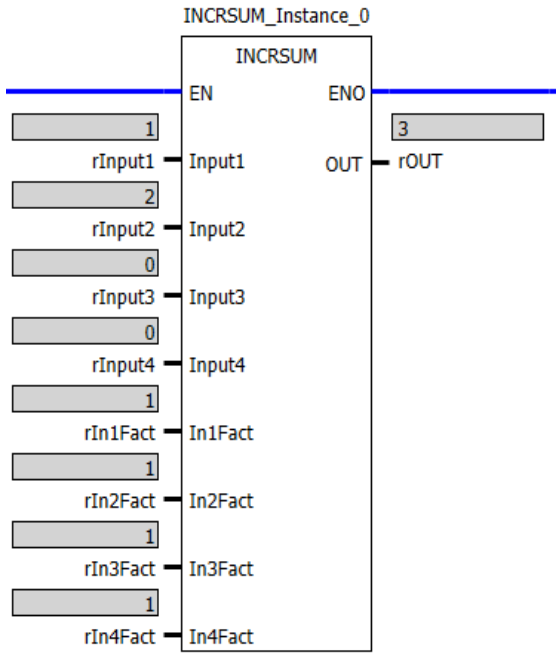
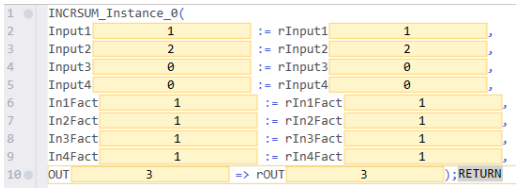
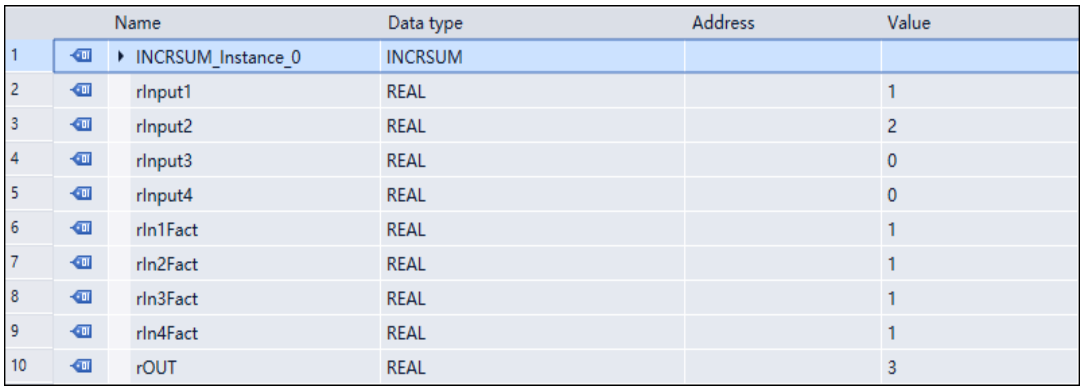
Inputx: Input 1 to 4

- **Program example**

  Set all proportional coefficients to 1. When input signal 1 and input signal 2 change from the default value (0) to 1 and 2 respectively, the system output is as shown in the table below.

Item	LD	ST

Define variables	VAR
	INCRSUM_Instance_0: INCRSUM;
	rInput1 : REAL := 0;
	rInput2 : REAL := 0;
	rInput3 : REAL := 0;
	rInput4 : REAL := 0;
	rIn1Fact : REAL := 1;
	rIn2Fact : REAL := 1;
	rIn3Fact : REAL := 1;
	rIn4Fact : REAL := 1;
	rOUT : REAL;
	END_VAR	

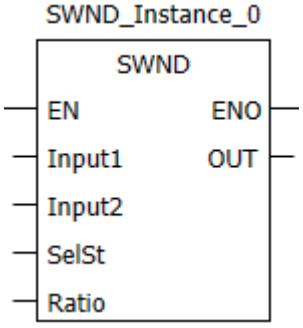
Item	LD	ST
Program		
Operation result		

	Name	Data type	Address	Value
1	INCRSUM_Instance_0	INCRSUM		
2	rInput1	REAL		1
3	rInput2	REAL		2
4	rInput3	REAL		0
5	rInput4	REAL		0
6	rIn1Fact	REAL		1
7	rIn2Fact	REAL		1
8	rIn3Fact	REAL		1
9	rIn4Fact	REAL		1
10	rOUT	REAL		3

18.2.13.8 SWND

Performs bumpless switchover between two analog signals.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SWND	Bumpless switchover	FB		SWND_Instance_0(Input1 := , Input2 := , SelSt := , Ratio := , OUT =>);

- **Related variable**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Input1	Input 1	Follows data type	0.0	Input 1
	Input2	Input 2	Follows data type	0.0	Input 2
	SelSt	Target input selection	Follows data type	FALSE	FALSE: Input 1 TRUE: Input 2
	Ratio	Percentage of the difference between Input 1 and Input 2	Follows data type	100	Range: 0% to 100%. 0% and 100% have the same effect, both being direct response.
Output	OUT	Output	Follows data type	0.0	Output

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Input1	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Input2	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
SelSt	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Ratio	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

- **Function**

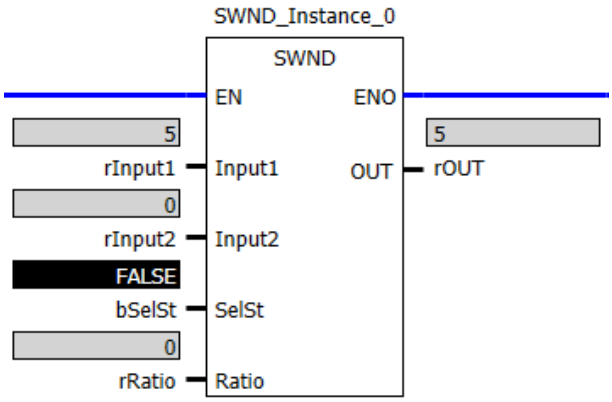
  This instruction uses the switch SelSt to select one analog output; during switching, the output tracks the target input at the set rate Ratio.

- **Program example**

  When SelSt is FALSE, the system tracks Input 1; when Ratio is 0, the system output is as shown in the table below.

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    SWND_Instance_0: SWND;
    rInput1: REAL := 5;
    rInput2: REAL := 0;
    bSelSt: BOOL := FALSE;
    rRatio: REAL := 0;
    rOUT: REAL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>
</td><td>

```
SWND_Instance_0(
Input1    5    := rInput1    5,
Input2    0    := rInput2    0,
SelSt  FALSE  := bSelSt  FALSE,
Ratio     0    := rRatio     0,
OUT       5    => rOUT       5 );RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

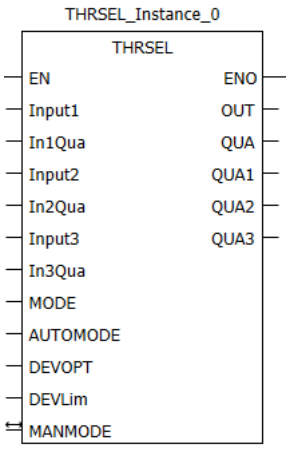
		Name	Data type	Address	Value
1		SWND_Instance_0	SWND		
2		rInput1	REAL		5
3		rInput2	REAL		0
4		bSelSt	BOOL		FALSE
5		rRatio	REAL		0
6		rOUT	REAL		5

</td></tr>
</table>

18.2.13.9 THRSEL

Selects output based on three analog input values, their quality status, and the operating mode (average, median, high select, low select, channel 1, channel 2, or channel 3).

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
THRSEL	Three-analog value selection	FB		THRSEL_Instance_0(Input1 := , In1Qua := , Input2 := , In2Qua := , Input3 := , In3Qua := , MODE := , AUTOMODE := , DEVOPT := , DEVLim := , MANMODE := , OUT => , QUA => , QUA1 => , QUA2 => , QUA3 =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Input1	Input 1	Follows data type	0.0	Input 1
	In1Qua	Quality of input 1	Follows data type	TRUE	Quality of input 1, default is good.
	Input2	Input 2	Follows data type	0.0	Input 2
	In2Qua	Quality of input 2	Follows data type	TRUE	Quality of input 2, default is good.
	Input3	Input 3	Follows data type	0.0	Input 3
	In3Qua	Quality of input 3	Follows data type	TRUE	Quality of input 3, default is good.

Input/Output	Variable	Name	Valid Range	Default Value	Description
	MODE	Manual/Auto mode	Follows data type	-	Default is manual.
	AUTOMODE	Automatic mode	Follows data type	-	Default is average.
	DEVOPT	Whether to perform deviation check	Follows data type	TRUE	Whether to perform deviation check
	DEVLim	Deviation value	Follows data type	30	Deviation value
Input/Output	MANMODE	Manual mode	Follows data type	-	Default is channel 1.
Output	OUT	Output	Follows data type	-	Output
	QUA	Total output quality	Follows data type	-	Total output quality
	QUA1	Quality of output 1	Follows data type	-	Quality of output 1
	QUA2	Quality of output 2	Follows data type	-	Quality of output 2
	QUA3	Quality of output 3	Follows data type	-	Quality of output 3

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Input1	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
In1Qua	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input2	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
In2Qua	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input3	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
In3Qua	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
MODE	Enumeration type: MA_MODE Name: MAN, Enumeration Value: 0, Comment: Manual mode Name: AUTO, Enumeration Value: 1, Comment: Automatic mode																			
AUTOMODE	Enumeration type: THRSEL_AUTOMODE Name: Average, Enumeration Value: 0, Comment: Average Name: Median, Enumeration Value: 1, Comment: Median Name: Higher, Enumeration Value: 2, Comment: High select Name: Lower, Enumeration Value: 3, Comment: Low select Name: First, Enumeration Value: 4, Comment: Channel 1 Name: Second, Enumeration Value: 5, Comment: Channel 2 Name: Third, Enumeration Value: 6, Comment: Channel 3																			
DEVOPT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
DEVLim	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
MANMODE	Enumeration type: THRSEL_MANMODE Name: First, Enumeration Value: 0, Comment: Channel 1 Name: Second, Enumeration Value: 1, Comment: Channel 2 Name: Third, Enumeration Value: 2, Comment: Channel 3																			
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
QUA	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
QUA1	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
QUA2	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
QUA3	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Calculates output based on three analog input values, their quality status, and the operating mode (average, median, high select, low select, channel 1, channel 2, or channel 3).

  Supports manual and automatic modes.

  - In manual mode, output of channel 1/2/3 can be selected, and the output quality corresponds to the input quality.
  - In automatic mode, the following modes are supported: average, median, high select, low select, and channel 1/2/3. The deviation check function can be enabled or disabled.
    - Average: Outputs the average of the inputs, based on quality and deviation check.
    - Median: Outputs the median value when there are three inputs; Outputs the average value when there are two inputs.
  - When all three inputs are of good quality
    - Deviation check
      1. If all three input groups exceed the deviation limit, the output holds its previous value, and the output quality is set to bad.
      2. If two input groups exceed the deviation limit, the output is based on the configured mode using only the input group that is within the limit. The overall output quality is good, and the quality of the inputs used in the calculation is also good. The remaining input is marked as bad quality.
      3. If only one input group exceeds the deviation limit, the output is the median value – specifically, the input that is within the deviation limit relative to the other two. The overall output quality is good, and the two inputs that are within the limit are marked as good quality. The input that exceeded the limit is marked as bad quality.
      4. If no input exceeds the deviation limit, the output is determined by the configured mode.
    - No deviation check: The output is determined by the configured mode.
  - When two inputs are of good quality
    - Deviation check:
      1. If the two inputs exceed the deviation limit, the output holds its previous value, and the output quality is set to bad.
      2. If no input exceeds the deviation limit, the output is determined by the configured mode.
    - No deviation check: The output is determined by the configured mode.
  - When one input is of good quality

Regardless of deviation check, the input of good quality is selected. The overall output quality is also good.

◦ When all three inputs are of bad quality

Channel 1/2/3 is selected: if the selected path's input quality is good, that channel is directly output; if the selected channel's quality is bad, the output holds its previous value.

For example:

When all three inputs are of good quality (In1Qua = In2Qua = In3Qua = TRUE), and DEVOPT = TRUE, the deviation check is performed as follows:

◦ If all deviations are within limits, the output is determined by the selection mode; the overall output quality is good (QUA = TRUE), and the three inputs are also of good quality (QUA1 = QUA2 = QUA3 = TRUE).

◦ When one input group exceeds deviation limit

If In2 and In3 exceed limits: outputs the median value (In1). QUA = TRUE, QUA1 = TRUE, QUA2 = QUA3 = FALSE.

If In1 and In3 exceed limits: outputs the median value (In2). QUA = TRUE, QUA2 = TRUE, QUA1 = QUA3 = FALSE.

If In1 and In2 exceed limits: outputs the median value (In3). QUA = TRUE, QUA3 = TRUE, QUA1 = QUA2 = FALSE.

◦ When two input groups exceed deviation limit, the output is based on the configured mode using the input group that is within the limit.

If (In1, In2) and (In1, In3) exceed limits: the output is based on the group within the limit (In2, In3). QUA = TRUE, QUA2 = QUA3 = TRUE, QUA1 = FALSE.

If (In1, In2) and (In2, In3) exceed limits: the output is based on the group within the limit (In1, In3). QUA = TRUE, QUA1 = QUA3 = TRUE, QUA2 = FALSE.

If (In1, In3) and (In2, In3) exceed limits: the output is based on the group within the limit (In1, In2). QUA = TRUE, QUA1 = QUA2 = TRUE, QUA3 = FALSE.

◦ When all three input groups exceed deviation limit, the output holds its previous value.

When two inputs are of good quality (In1Qua = In2Qua = TRUE; In3Qua = FALSE), and DEVOPT = TRUE, the deviation check is performed as follows:

Deviation within limits: The output follows the selected mode. For average or median mode, the output is the average of the two good inputs. QUA = TRUE, QUA1 = QUA2 = TRUE, QUA3 = FALSE.

Deviation exceeds limit: The output holds its previous value. All input qualities are bad and the overall output quality is also bad. QUA1 = QUA2 = QUA3 = FALSE, QUA = FALSE.

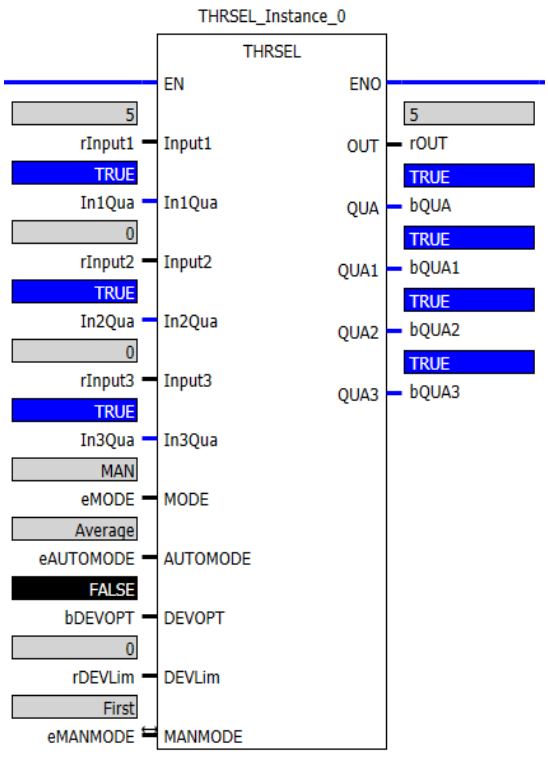
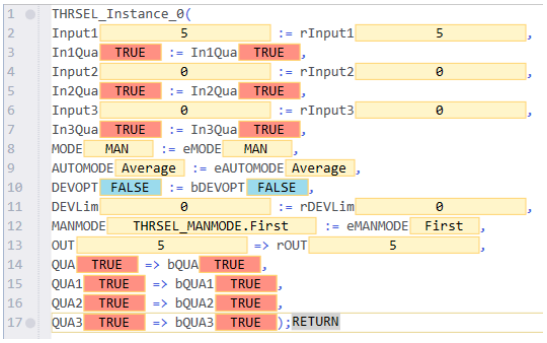
Bumpless switchover from automatic mode to manual mode

In automatic mode, manual mode tracks the automatic selection result to ensure bumpless switchover from automatic to manual mode. In automatic mode, if the input to track cannot be determined (e.g., in the case of averaging), the channel whose value is closest to the auto output is tracked.

- **Program example**

Item	LD	ST

Define variables	VAR
	THRSEL_Instance_0: THRSEL;
	rInput1: REAL := 5;
	In1Qua: BOOL := TRUE;
	rInput2: REAL := 0;
	In2Qua: BOOL := TRUE;
	rInput3: REAL := 0;
	In3Qua: BOOL := TRUE;
	eMODE: MA_MODE := MA_MODE.MAN;
	eAUTOMODE: THRSEL_AUTOMODE := THRSEL_AUTOMODE.Average;
	bDEVOPT: BOOL := FALSE;
	rDEVLim: REAL := 0;
	eMANMODE: THRSEL_MANMODE;
	rOUT: REAL;
	bQUA: BOOL;
	bQUA1: BOOL;
	bQUA2: BOOL;
	bQUA3: BOOL;
	END_VAR		

Program		

Item	LD	ST

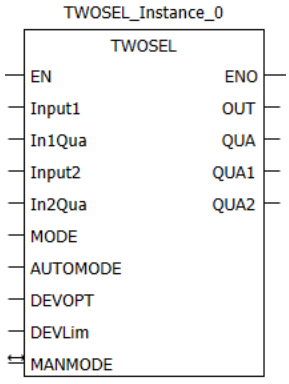
Operation result

	Name	Data type	Address	Value
1	THRSEL_Instance_0	THRSEL		
2	rInput1	REAL		5
3	In1Qua	BOOL		TRUE
4	rInput2	REAL		0
5	In2Qua	BOOL		TRUE
6	rInput3	REAL		0
7	In3Qua	BOOL		TRUE
8	eMODE	MA_MODE		MAN
9	eAUTOMODE	THRSEL_AUTOMODE		Average
10	bDEVOPT	BOOL		FALSE
11	rDEVLim	REAL		0
12	eMANMODE	THRSEL_MANMODE		First
13	rOUT	REAL		5
14	bQUA	BOOL		TRUE
15	bQUA1	BOOL		TRUE
16	bQUA2	BOOL		TRUE
17	bQUA3	BOOL		TRUE

18.2.13.10 TWOSEL

TWOSEL is used for two-analog value selection.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
TWOSEL	Two-analog value selection	FB		TWOSEL_Instance_0(Input1 := , In1Qua := , Input2 := , In2Qua := , MODE := , AUTOMODE := , DEVOPT := , DEVLim := , MANMODE := , OUT => , QUA => , QUA1 => , QUA2 =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Input1	Input 1	Follows data type	0.0	Input 1

Input/Output	Variable	Name	Valid Range	Default Value	Description
	In1Qua	Quality of input 1	Follows data type	TRUE	Quality of input 1
	Input2	Input 2	Follows data type	0.0	Input 2
	In2Qua	Quality of input 2	Follows data type	TRUE	Quality of input 2
	MODE	Manual/Auto mode	Follows data type	-	Default is manual.
	AUTOMODE	Automatic mode	Follows data type	-	Default is average.
	DEVOPT	Whether to perform deviation check	Follows data type	TRUE	Whether to perform deviation check
	DEVLim	Deviation value	Follows data type	30	Deviation value
Input/Output	MANMODE	Manual mode	Follows data type	-	Default is channel 1.
Output	OUT	Output	Follows data type	-	Output
	QUA	Total output quality	Follows data type	-	Total output quality
	QUA1	Quality of output 1	Follows data type	-	Quality of output 1
	QUA2	Quality of output 2	Follows data type	-	Quality of output 2

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Input1	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
In1Qua	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Input2	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
In2Qua	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
MODE	Enumeration type: MA_MODE Name: MAN, Enumeration Value: 0, Comment: Manual mode Name: AUTO, Enumeration Value: 1, Comment: Automatic mode																			
AUTOMODE	Enumeration type: TWOSEL_AUTOMODE Name: Higher, Enumeration Value: 0, Comment: High select Name: Lower, Enumeration Value: 1, Comment: Low select Name: Redundant, Enumeration Value: 2, Comment: Redundant Name: First, Enumeration Value: 3, Comment: Channel 1 Name: Second, Enumeration Value: 4, Comment: Channel 2																			
DEVOPT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
DEVLim	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
MANMODE	Enumeration type: TWOSEL_MANMODE Name: First, Enumeration Value: 0, Comment: Channel 1 Name: Second, Enumeration Value: 1, Comment: Channel 2																			
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
QUA	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
QUA1	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
QUA2	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

Calculates output based on the two analog input values, their quality status, and the operating mode (high select, low select, redundant, channel 1, and channel 2).

- Supports manual and automatic modes.
  - In manual mode, output of channel 1/2 can be selected, and the output quality corresponds to the input quality.
  - In automatic mode, the following modes are supported: high select, low select, redundant, channel 1, and channel 2. The deviation check function can be enabled or disabled.

High select or low select:

- When both inputs are of good quality
  - Deviation check:
    1. If the deviation between the two inputs exceeds the limit, the output holds its previous value. The total output quality and the qualities of the both outputs are set to bad.
    2. If the deviation is within the limit, the output follows the selected mode. The total output quality and the qualities of the both outputs are set to good.
  - Deviation check disabled: The output follows the selected mode. The total output quality and the qualities of the both outputs are set to good.
- When only one input is of good quality:

  This input is selected regardless of deviation check.

Redundancy mode:

- When only one input is of good quality, this input is selected. When both inputs are of good quality, the output holds its previous value.
- When both inputs are of bad quality, the output holds its previous value, and the total output quality and the qualities of the both outputs are set to bad.

Channel 1 or channel 2:

If the selected channel has good input quality, it is directly output; otherwise, the output holds its previous value, and the total output quality and the qualities of the both outputs are set to bad.

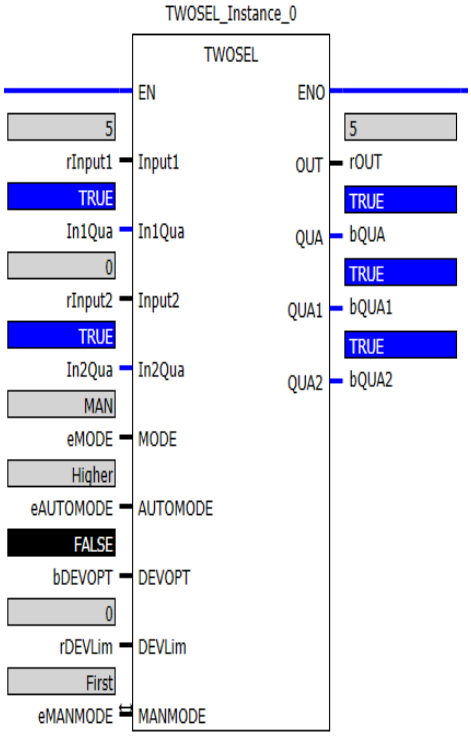
Bumpless switchover from automatic mode to manual mode

In automatic mode, manual mode tracks the automatic selection result to ensure bumpless switchover from automatic to manual mode.

- **Program example**

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    TWOSEL_Instance_0: TWOSEL;
    rInput1: REAL := 5;
    In1Qua: BOOL := TRUE;
    rInput2: REAL := 0;
    In2Qua: BOOL := TRUE;
    eMODE: MA_MODE := MA_MODE.MAN;
    eAUTOMODE: TWOSEL_AUTOMODE := TWOSEL_AUTOMODE.Higher;
    bDEVOPT: BOOL := FALSE;
    rDEVLim: REAL := 0;
    eMANMODE: TWOSEL_MANMODE := TWOSEL_MANMODE.First;
    rOUT: REAL;
    bQUA: BOOL;
    bQUA1: BOOL;
    bQUA2: BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>
</td><td>

```
TWOSEL_Instance_0(
Input1 := 5 := rInput1 := 5,
In1Qua := TRUE := In1Qua := TRUE,
Input2 := 0 := rInput2 := 0,
In2Qua := TRUE := In2Qua := TRUE,
MODE := MAN := eMODE := MAN,
AUTOMODE := Higher := eAUTOMODE := Higher,
DEVOPT := FALSE := bDEVOPT := FALSE,
DEVLim := 0 := rDEVLim := 0,
MANMODE := TWOSEL_MANMODE.First := eMANMODE := First,
OUT := 5 => rOUT := 5,
QUA := TRUE => bQUA := TRUE,
QUA1 := TRUE => bQUA1 := TRUE,
QUA2 := TRUE => bQUA2 := TRUE);RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

	Name	Data type	Address	Value
1	TWOSEL_Instance_0	TWOSEL		
2	rInput1	REAL		5
3	In1Qua	BOOL		TRUE
4	rInput2	REAL		0
5	In2Qua	BOOL		TRUE
6	eMODE	MA_MODE		MAN
7	eAUTOMODE	TWOSEL_AUTOMODE		Higher
8	bDEVOPT	BOOL		FALSE
9	rDEVLim	REAL		0
10	eMANMODE	TWOSEL_MANMODE		First
11	rOUT	REAL		5
12	bQUA	BOOL		TRUE
13	bQUA1	BOOL		TRUE
14	bQUA2	BOOL		TRUE

</td></tr>
</table>

18.2.14 Digital Signal

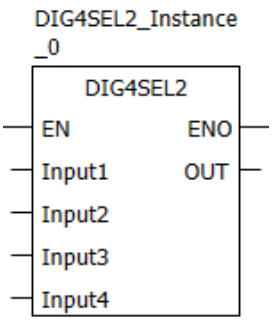
18.2.14.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.14.2 DIG4SEL2	FB	2-out-of-4 digital output selection	SignalProcessing	Library version 1.0.0.0 and later
18.2.14.3 DIG4SEL3	FB	3-out-of-4 digital output selection		Library version 1.0.0.0 and later
18.2.14.4 DIGCNUM	FB	Digital input transition count		Library version 1.0.0.0 and later
18.2.14.5 DIGONUM	FB	Digital input TRUE count comparison		Library version 1.0.0.0 and later
18.2.14.6 DIGSTA	FB	Digital input status time accumulation		Library version 1.0.0.0 and later
18.2.14.7 DIGXOr04	FB	XOR logic operation		Library version 1.0.0.0 and later
18.2.14.8 FSTDEF	FB	First transition memory		Library version 1.0.0.0 and later
18.2.14.9 Inlock4	FB	Interlock block		Library version 1.0.0.0 and later

18.2.14.2 DIG4SEL2

DIG4SEL2 is used for 2-out-of-4 digital output selection.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DIG4SEL2	2-out-of-4 digital output selection	FB		DIG4SEL2_Instance_0(Input1 := , Input2 := , Input3 := , Input4 := , OUT =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Input1	Input 1	Follows data type	-	Input 1

Input/Output	Variable	Name	Valid Range	Default Value	Description
	Input2	Input 2	Follows data type	-	Input 2
	Input3	Input 3	Follows data type	-	Input 3
	Input4	Input 4	Follows data type	-	Input 4
Output	OUT	Output	Follows data type	-	Output

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Input1	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input2	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input3	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input4	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

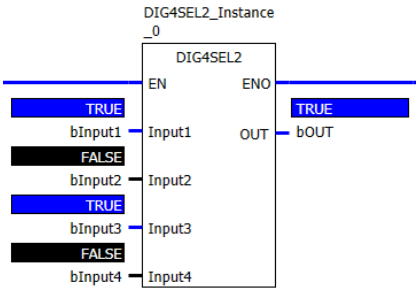
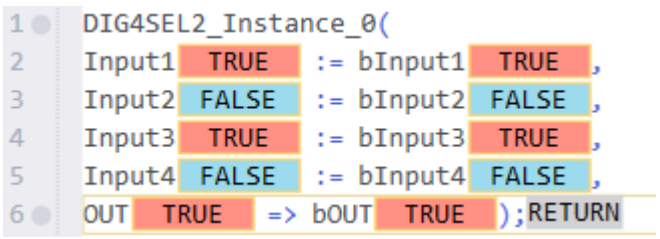
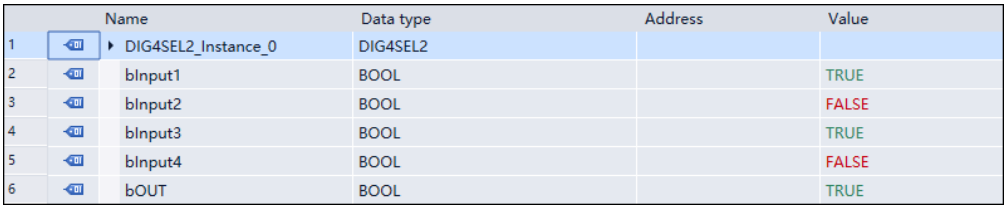
- **Function**

  If at least 2 out of 4 digital inputs are ON, the output is ON.

- **Program example**

Item	LD	ST

Define variables	VAR
	DIG4SEL2_Instance_0: DIG4SEL2;
	bInput1 : BOOL := TRUE;
	bInput2 : BOOL := FALSE;
	bInput3 : BOOL := TRUE;
	bInput4 : BOOL := FALSE;
	bOUT : BOOL;
	END_VAR	

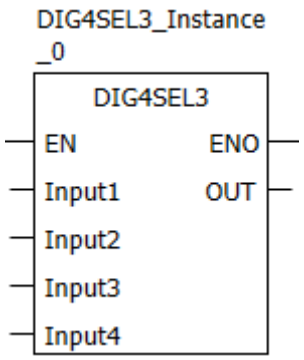
Item	LD	ST
Program		
Operation result		

	Name	Data type	Address	Value
1	DIG4SEL2_Instance_0	DIG4SEL2		
2	bInput1	BOOL		TRUE
3	bInput2	BOOL		FALSE
4	bInput3	BOOL		TRUE
5	bInput4	BOOL		FALSE
6	bOUT	BOOL		TRUE

18.2.14.3 DIG4SEL3

DIG4SEL3 is used for 3-out-of-4 digital output selection.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DIG4SEL3	3-out-of-4 digital output selection	FB		DIG4SEL3_Instance_0(Input1 := , Input2 := , Input3 := , Input4 := , OUT =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Input1	Input 1	Follows data type	-	Input 1
	Input2	Input 2	Follows data type	-	Input 2
	Input3	Input 3	Follows data type	-	Input 3
	Input4	Input 4	Follows data type	-	Input 4
Output	OUT	Output	Follows data type	-	Output

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Input1	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input2	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input3	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input4	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

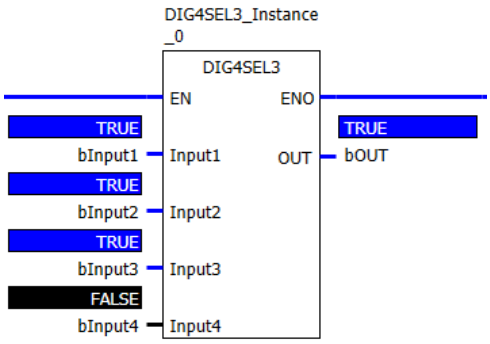
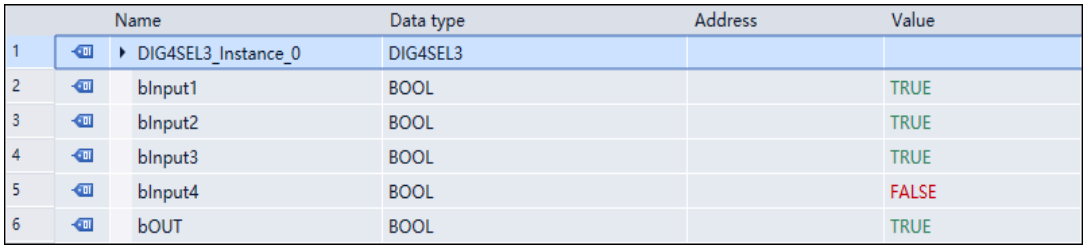
- **Function**

  If at least 3 out of 4 digital inputs are ON, the output is ON.

- **Program example**

Item	LD	ST

Define variables	VAR
	DIG4SEL3_Instance_0: DIG4SEL3;
	bInput1 : BOOL := TRUE;
	bInput2 : BOOL := TRUE;
	bInput3 : BOOL := TRUE;
	bInput4 : BOOL := FALSE;
	bOUT : BOOL;
	END_VAR	

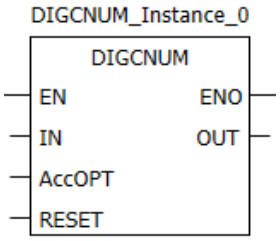
Program		DIG4SEL3_Instance_0(Input1 TRUE := bInput1 TRUE, Input2 TRUE := bInput2 TRUE, Input3 TRUE := bInput3 TRUE, Input4 FALSE := bInput4 FALSE, OUT TRUE => bOUT TRUE);RETURN
Operation result		

	Name	Data type	Address	Value
1	DIG4SEL3_Instance_0	DIG4SEL3		
2	bInput1	BOOL		TRUE
3	bInput2	BOOL		TRUE
4	bInput3	BOOL		TRUE
5	bInput4	BOOL		FALSE
6	bOUT	BOOL		TRUE

18.2.14.4 DIGCNUM

Calculates the number of digital input signal transitions.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DIGCNUM	Digital input transition count	FB		DIGCNUM_Instance_0(IN :=, AccOPT :=, RESET :=, OUT =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input	Follows data type	-	Input
	AccOPT	Selected status for accumulation	Follows data type	-	0: Rising edge transition 1: Falling edge transition 2: Rising and falling edge transition
	RESET	Reset a bit	Follows data type	-	Reset
Output	OUT	Output	Follows data type	-	Output

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
AccOPT	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

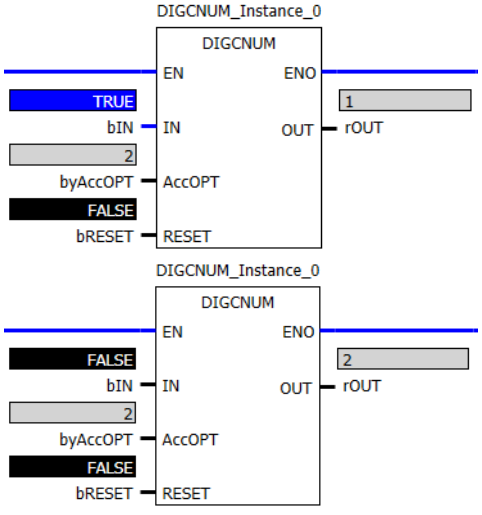
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

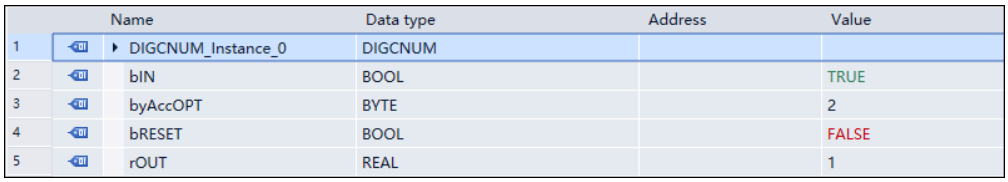
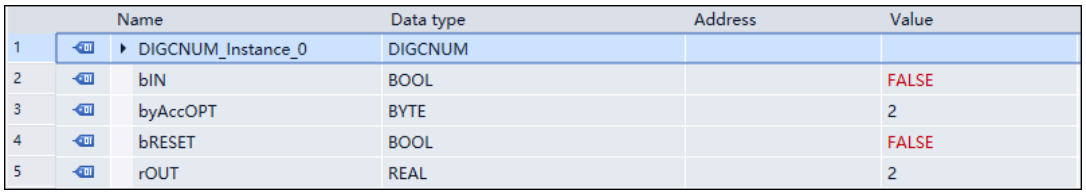
- **Function**
  - Counts the number of rising edge transitions, falling edge transitions, or both rising and falling edge transitions for digital input signals. This instruction applies to scenarios requiring counting or accumulating the number of state transitions of equipment. It is commonly applied for valves, motors, and similar equipment.
  - The counted transition types include:
    - 0: Rising edge transition
    - 1: Falling edge transition
    - 2: Rising and falling edge transition
  - The statistics result can be reset via the "RESET" parameter.
- **Program example**

  The following example counts the number of both rising and falling edge transitions. Set input "IN" from FALSE to TRUE, and then from TRUE to FALSE; the output is as shown in the table below.

Item	LD	ST

Define variables	VAR
	DIGCNUM_Instance_0: DIGCNUM;
	bIN : BOOL := TRUE;
	byAccOPT : BYTE := 2;
	bRESET : BOOL := FALSE;
	rOUT : REAL;
	END_VAR	

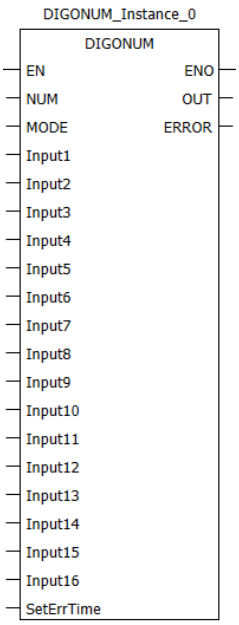
Program		1 DIGCNUM_Instance_0(2 IN TRUE := bIN TRUE, 3 AccOPT 2 := byAccOPT 2, 4 RESET FALSE := bRESET FALSE, 5 OUT 1 => rOUT 1);RETURN 1 DIGCNUM_Instance_0(2 IN FALSE := bIN FALSE, 3 AccOPT 2 := byAccOPT 2, 4 RESET FALSE := bRESET FALSE, 5 OUT 2 => rOUT 2);RETURN

Item	LD	ST
Operation result	 	

18.2.14.5 DIGONUM

Calculates the number of TRUE digital inputs and compares the result with the set number.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DIGONUM	Digital input TRUE count comparison	FB		DIGONUM_Instance_0(NUM := , MODE := , Input1 := , Input2 := , Input3 := , Input4 := , Input5 := , Input6 := , Input7 := , Input8 := , Input9 := , Input10 := , Input11 := , Input12 := , Input13 := , Input14 := , Input15 := , Input16 := , SetErrTime := , OUT => , ERROR =>);

- **Related variable**

  Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	NUM	Preset number of TRUE digital inputs	Follows data type	-	Preset number of TRUE digital inputs

Input/ Output	Variable	Name	Valid Range	Default Value	Description
	MODE	Comparison mode	Follows data type	-	0: Greater than or equal to 1: Greater than 2: Equal to 3: Less than or equal to 4: Less than
	Input1	Input 1	Follows data type	-	Input 1
	Input2	Input 2	Follows data type	-	Input 2
	Input3	Input 3	Follows data type	-	Input 3
	Input4	Input 4	Follows data type	-	Input 4
	Input5	Input 5	Follows data type	-	Input 5
	Input6	Input 6	Follows data type	-	Input 6
	Input7	Input 7	Follows data type	-	Input 7
	Input8	Input 8	Follows data type	-	Input 8
	Input9	Input 9	Follows data type	-	Input 9
	Input10	Input 10	Follows data type	-	Input 10
	Input11	Input 11	Follows data type	-	Input 11
	Input12	Input 12	Follows data type	-	Input 12
	Input13	Input 13	Follows data type	-	Input 13

Input/ Output	Variable	Name	Valid Range	Default Value	Description
	Input14	Input 14	Follows data type	-	Input 14
	Input15	Input 15	Follows data type	-	Input 15
	Input16	Input 16	Follows data type	-	Input 16
	SetErrTime	Time limit for fault detection	Follows data type	TIME#4294967295ms	Time limit for fault detection
Output	OUT	Output	Follows data type	-	Output
	ERROR	Abnormal output timeout	Follows data type	-	Abnormal output timeout

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
NUM	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
MODE	Custom type: DIGONUM_MODE 0: eGE (Greater than or equal to) 1: eGT (Greater than) 2: eEQ (Equal to) 3: eLE (Less than or equal to) 4: eLT (Less than)																			
Input1	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input2	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input3	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input4	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input5	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input6	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Input7	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input8	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input9	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input10	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input11	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input12	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input13	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input14	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input15	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input16	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SetErrTime	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–
OUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ERROR	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Counts the number of digital inputs (up to 16 inputs supported) that are TRUE, compares the result with the set number NUM, and outputs the comparison result. In field applications, it is necessary to count how many devices are in TRUE (1) or FALSE (0) status. When the number of devices in certain status is greater than or equal to the set number, the output of BOOL type becomes ON; otherwise, it becomes OFF.

  ◦ Condition check

  Different comparison methods can be applied between the count result and the preset value. When the comparison condition is met, "OUT" outputs TRUE; otherwise, it outputs FALSE. Comparison methods include: greater than or equal, less than or equal, equal to, greater than, and less than.

  By setting the preset value NUM and comparison method MODE, the voting function for digital signals can be achieved. For example, to implement 2-out-of-3 voting for air-operated digital signals, set the comparison method MODE to 0 (Greater than or equal to) with a preset value of 2, set the comparison value to 2, and connect the three signals to be voted to the three inputs of the function block.

  ◦ Fault alarm

When the comparison output changes from TRUE to FALSE, the fault timer is started, counting the cumulative time during which the output remains FALSE. When the timer reaches the preset time limit for fault detection "SetErrTime", "ERR" outputs TRUE; otherwise, "ERR" outputs FALSE.

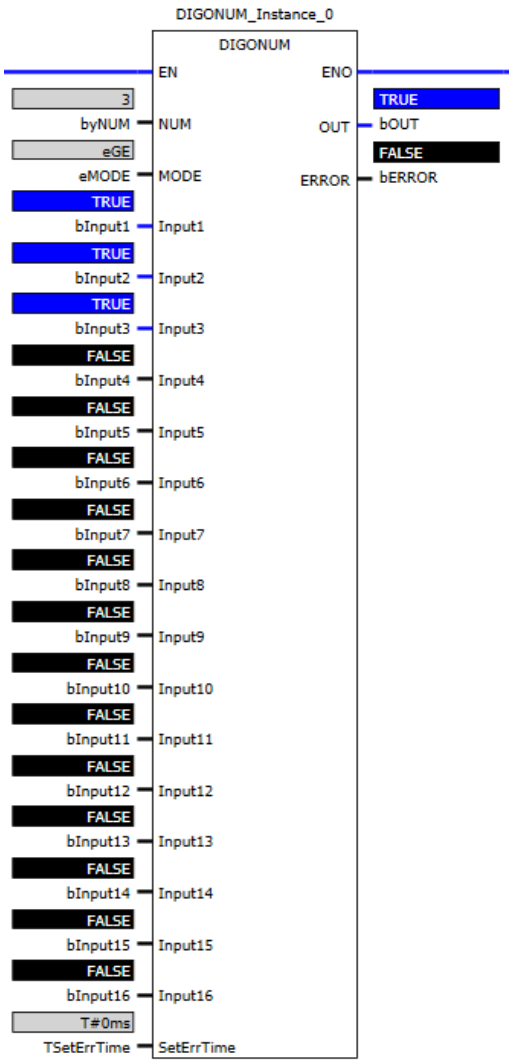
This function can be used to determine the duration of invalid field conditions. If this duration exceeds SetErrTime, the output ERROR is set to TRUE. If the fault alarm is not required, set SetErrTime to 0.

- **Program example**

  The following example sets the input NUM to 3. When three input signals are TRUE, the output result is as shown in the table below.

Item	LD	ST

Define variables	VAR
	DIGONUM_Instance_0: DIGONUM;
	byNUM: BYTE := 3;
	eMODE: DIGONUM_MODE := DIGONUM_MODE.eGE;
	bInput1: BOOL := TRUE;
	bInput2: BOOL := TRUE;
	bInput3: BOOL := TRUE;
	bInput4: BOOL := FALSE;
	bInput5: BOOL := FALSE;
	bInput6: BOOL := FALSE;
	bInput7: BOOL := FALSE;
	bInput8: BOOL := FALSE;
	bInput9: BOOL := FALSE;
	bInput10: BOOL := FALSE;
	bInput11: BOOL := FALSE;
	bInput12: BOOL := FALSE;
	bInput13: BOOL := FALSE;
	bInput14: BOOL := FALSE;
	bInput15: BOOL := FALSE;
	bInput16: BOOL := FALSE;
	TSetErrTime: TIME := T#0ms;
	bOUT: BOOL;
	bERROR: BOOL;
	END_VAR	

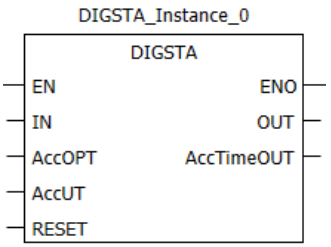
<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>

</td><td>
<pre>
DIGONUM_Instance_0(
NUM 3 := byNUM 3,
MODE eGE := eMODE eGE,
Input1 TRUE := bInput1 TRUE,
Input2 TRUE := bInput2 TRUE,
Input3 TRUE := bInput3 TRUE,
Input4 FALSE := bInput4 FALSE,
Input5 FALSE := bInput5 FALSE,
Input6 FALSE := bInput6 FALSE,
Input7 FALSE := bInput7 FALSE,
Input8 FALSE := bInput8 FALSE,
Input9 FALSE := bInput9 FALSE,
Input10 FALSE := bInput10 FALSE,
Input11 FALSE := bInput11 FALSE,
Input12 FALSE := bInput12 FALSE,
Input13 FALSE := bInput13 FALSE,
Input14 FALSE := bInput14 FALSE,
Input15 FALSE := bInput15 FALSE,
Input16 FALSE := bInput16 FALSE,
SetErrTime T#0ms := TSetErrTime T#0ms,
OUT TRUE => bOUT TRUE,
ERROR FALSE => bERROR FALSE);RETURN
</pre>
</td></tr>
<tr><td>Operation result</td><td colspan="2">
<table>
<tr><th></th><th></th><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td>1</td><td></td><td>DIGONUM_Instance_0</td><td>DIGONUM</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td>byNUM</td><td>BYTE</td><td></td><td>3</td></tr>
<tr><td>3</td><td></td><td>eMODE</td><td>DIGONUM_MODE</td><td></td><td>eGE</td></tr>
<tr><td>4</td><td></td><td>bInput1</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>5</td><td></td><td>bInput2</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>6</td><td></td><td>bInput3</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>7</td><td></td><td>bInput4</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>8</td><td></td><td>bInput5</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>9</td><td></td><td>bInput6</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>10</td><td></td><td>bInput7</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>11</td><td></td><td>bInput8</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>12</td><td></td><td>bInput9</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>13</td><td></td><td>bInput10</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>14</td><td></td><td>bInput11</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>15</td><td></td><td>bInput12</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>16</td><td></td><td>bInput13</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>17</td><td></td><td>bInput14</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>18</td><td></td><td>bInput15</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>19</td><td></td><td>bInput16</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>20</td><td></td><td>TSetErrTime</td><td>TIME</td><td></td><td>T#0ms</td></tr>
<tr><td>21</td><td></td><td>bOUT</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>22</td><td></td><td>bERROR</td><td>BOOL</td><td></td><td>FALSE</td></tr>
</table>
</td></tr>
</table>

18.2.14.6 DIGSTA

Outputs the cumulative time for a specified status of the digital inputs.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DIGSTA	Digital input status time accumulation	FB		DIGSTA_Instance_0(IN := , AccOPT := , AccUT := , RESET := , OUT => , AccTimeOUT =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input	Follows data type	-	Input
	AccOPT	Selected status for accumulation	Follows data type	-	0: OFF status (0) 1: ON status (1)
	AccUT	Accumulation unit	Follows data type	ACC_UT.Second	0: Second 1: Minute 2: Hour 3: Day
	RESET	Reset a bit	Follows data type	-	Reset
Output	OUT	Accumulated time output (in specified unit)	Follows data type	-	Accumulated time output (in specified unit)
	AccTimeOUT	Total accumulated time output (in second)	Follows data type	-	Total accumulated time output (in second)

Variable types

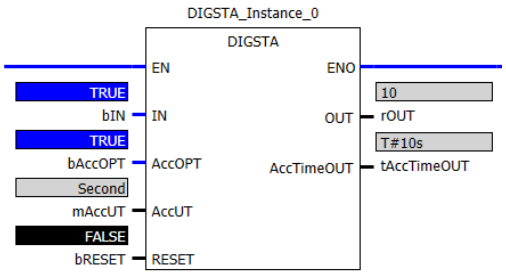
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
AccOPT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
AccUT	Custom type: ACC_UT 0: Second 1: Minute 2: Hour 3: Day																			
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
AccTime OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–

- **Function**
  - This instruction accumulates the duration of the 0 (OFF) and 1 (ON) statuses of a digital input signal separately, and outputs the accumulated time for the selected status in the specified unit. It can be used to calculate the cumulative time of equipment in a specific status, such as pump operation or feeding timing.
  - The time units support seconds, minutes, hours, and days.
  - The selected status for accumulation AccOPT can be 0 or 1:
    - AccOPT = 0: accumulates only the time when the input signal is 0.
    - AccOPT = 1: accumulates only the time when the input signal is 1.
  - The accumulated time can be reset using the "RESET" parameter.
- **Program example**

  The following example calculates the accumulated time (in seconds) for input signal "IN". The calculation results for 10 scheduled cycles are shown in the table below.

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    DIGSTA_Instance_0: DIGSTA;
    bIN: BOOL := TRUE;
    bAccOPT: BOOL := TRUE;
    mAccUT: ACC_UT := ACC_UT.Second;
    bRESET: BOOL;
    rOUT: REAL;
    tAccTimeOUT: TIME;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>
</td><td>

```
DIGSTA_Instance_0(
IN TRUE := bIN TRUE,
AccOPT TRUE := bAccOPT TRUE,
AccUT Second := mAccUT Second,
RESET FALSE := bRESET FALSE,
OUT 10 => rOUT 10,
AccTimeOUT T#10s => tAccTimeOUT T#10s);
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

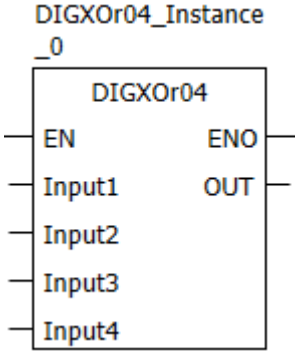
		Name	Data type	Address	Value
1		DIGSTA_Instance_0	DIGSTA		
2		bIN	BOOL		TRUE
3		bAccOPT	BOOL		TRUE
4		mAccUT	ACC_UT		Second
5		bRESET	BOOL		FALSE
6		rOUT	REAL		10
7		tAccTimeOUT	TIME		T#10s

</td></tr>
</table>

18.2.14.7 DIGXOr04

DIGXOr04 is used for XOR logic operation.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DIGXOr04	XOR logic operation	FB		DIGXOr04_Instance_0(Input1 := , Input2 := , Input3 := , Input4 := , OUT =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Input1	Input 1	Follows data type	-	Input 1
	Input2	Input 2	Follows data type	-	Input 2

Input/Output	Variable	Name	Valid Range	Default Value	Description
	Input3	Input 3	Follows data type	-	Input 3
	Input4	Input 4	Follows data type	-	Input 4
Output	OUT	Output	Follows data type	-	Output

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Input1	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input2	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input3	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input4	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Performs XOR logic operations on up to four digital inputs.

  This block can perform logical operations on up to four digital input parameters. If an odd number of inputs are 1 and the others are 0, the output "OUT" is 1.

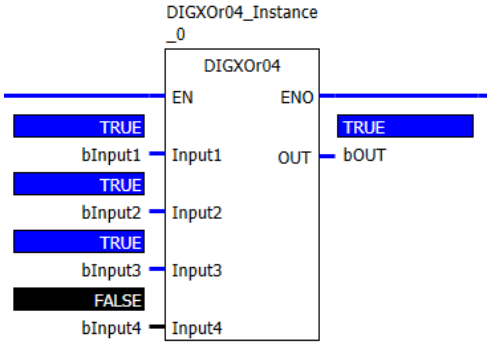
No.	In1	In2	In3	In4	Out
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0

No.	In1	In2	In3	In4	Out
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	0

- **Program example**

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    DIGXOr04_Instance_0: DIGXOr04;
    bInput1: BOOL := TRUE;
    bInput2: BOOL := TRUE;
    bInput3: BOOL := TRUE;
    bInput4: BOOL := FALSE;
    bOUT: BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>
</td><td>

```
DIGXOr04_Instance_0(
Input1 TRUE := bInput1 TRUE ,
Input2 TRUE := bInput2 TRUE ,
Input3 TRUE := bInput3 TRUE ,
Input4 FALSE := bInput4 FALSE ,
OUT TRUE => bOUT TRUE );RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

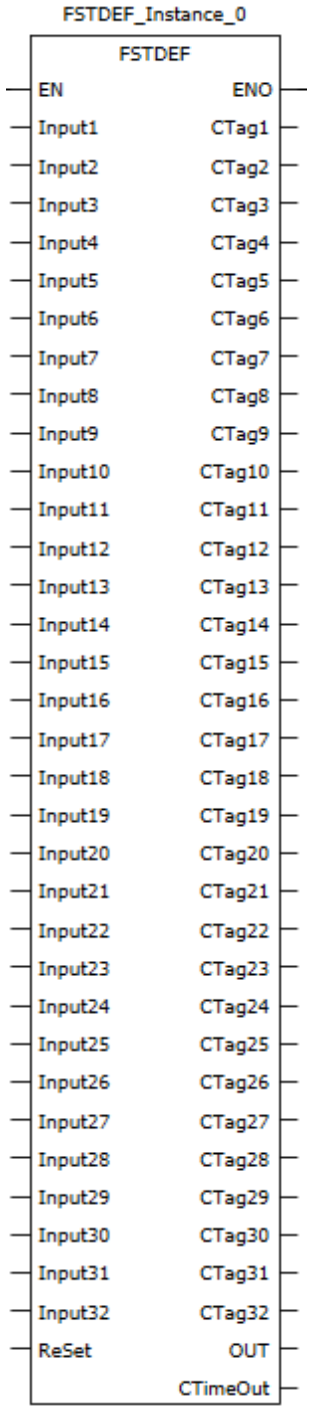
	Name	Data type	Address	Value
1	DIGXOr04_Instance_0	DIGXOr04		
2	bInput1	BOOL		TRUE
3	bInput2	BOOL		TRUE
4	bInput3	BOOL		TRUE
5	bInput4	BOOL		FALSE
6	bOUT	BOOL		TRUE

</td></tr>
</table>

18.2.14.8 FSTDEF

Monitors the digital inputs and sets the first digital input that transitions (rising edge) to TRUE.

- **Instruction format**

FSTDEF	First transition memory	FB		FSTDEF_Instance_0(Input1 := , Input2 := , Input3 := , Input4 := , Input5 := , Input6 := , Input7 := , Input8 := , Input9 := , Input10 := , Input11 := , Input12 := , Input13 := , Input14 := , Input15 := , Input16 := , Input17 := , Input18 := , Input19 := , Input20 := , Input21 := , Input22 := , Input23 := , Input24 := , Input25 := , Input26 := , Input27 := , Input28 := , Input29 := , Input30 := , Input31 := , Input32 := , ReSet := , CTag1 => , CTag2 => ,

Instruction	Name	FB/FC	LD Representation	ST Representation
				CTag3 => , CTag4 => , CTag5 => , CTag6 => , CTag7 => , CTag8 => , CTag9 => , CTag10 => , CTag11 => , CTag12 => , CTag13 => , CTag14 => , CTag15 => , CTag16 => , CTag17 => , CTag18 => , CTag19 => , CTag20 => , CTag21 => , CTag22 => , CTag23 => , CTag24 => , CTag25 => , CTag26 => , CTag27 => , CTag28 => , CTag29 => , CTag30 => , CTag31 => , CTag32 => , OUT => , CTimeOut =>);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	Input1 to Input32	Input 1 to Input 32	Follows data type	-	Input 1 to Input 32.
	ReSet	Reset a bit	Follows data type	-	Reset
Output	CTag1 to CTag32	Transition mark 1 to Transition mark 32	Follows data type	-	Transition mark 1 to Transition mark 32
	OUT	Transition mark	Follows data type	-	Transition mark
	CTimeOut	Transition time	Follows data type	-	Transition time

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Input1 to Input32	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CTag1 to CTag32	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CTimeOut	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–

- **Function**

  This instruction monitors up to 32 digital signals for status transitions (rising edge-triggered). The transition mark corresponding to the first signal that transitions becomes TRUE until reset. It can be used for on-site accident analysis.

  - First transition function
    - If any one of the 32 input signals transitions, the total transition mark "OUT" becomes TRUE, and the transition mark of the corresponding input also becomes TRUE until reset.

- When multiple input signals transition in sequence, only the one occurs first is recorded; subsequent transitions are not recorded. This status is maintained until reset.
- When multiple input signals transition simultaneously, the function block defaults to recording the lowest-indexed input as the first transition.

◦ Time recording

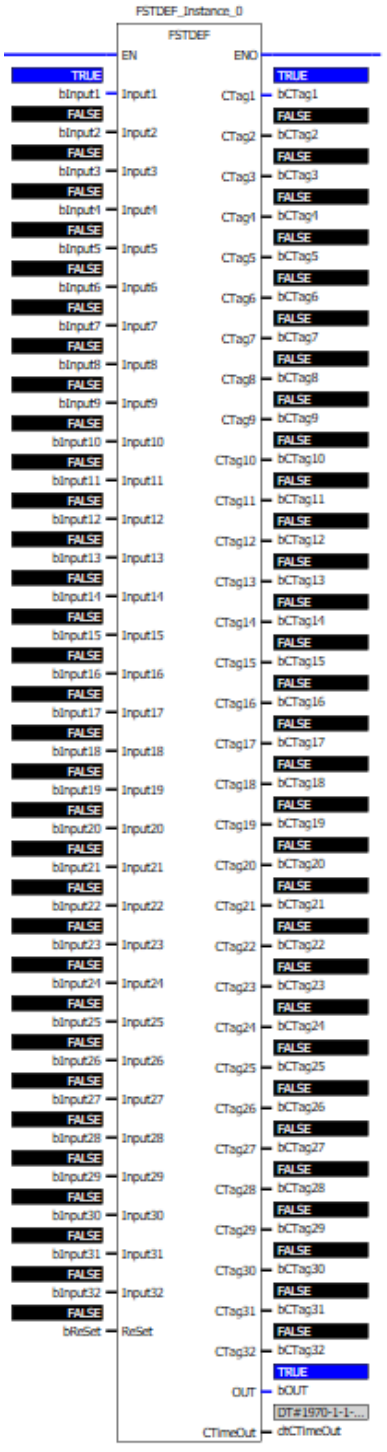
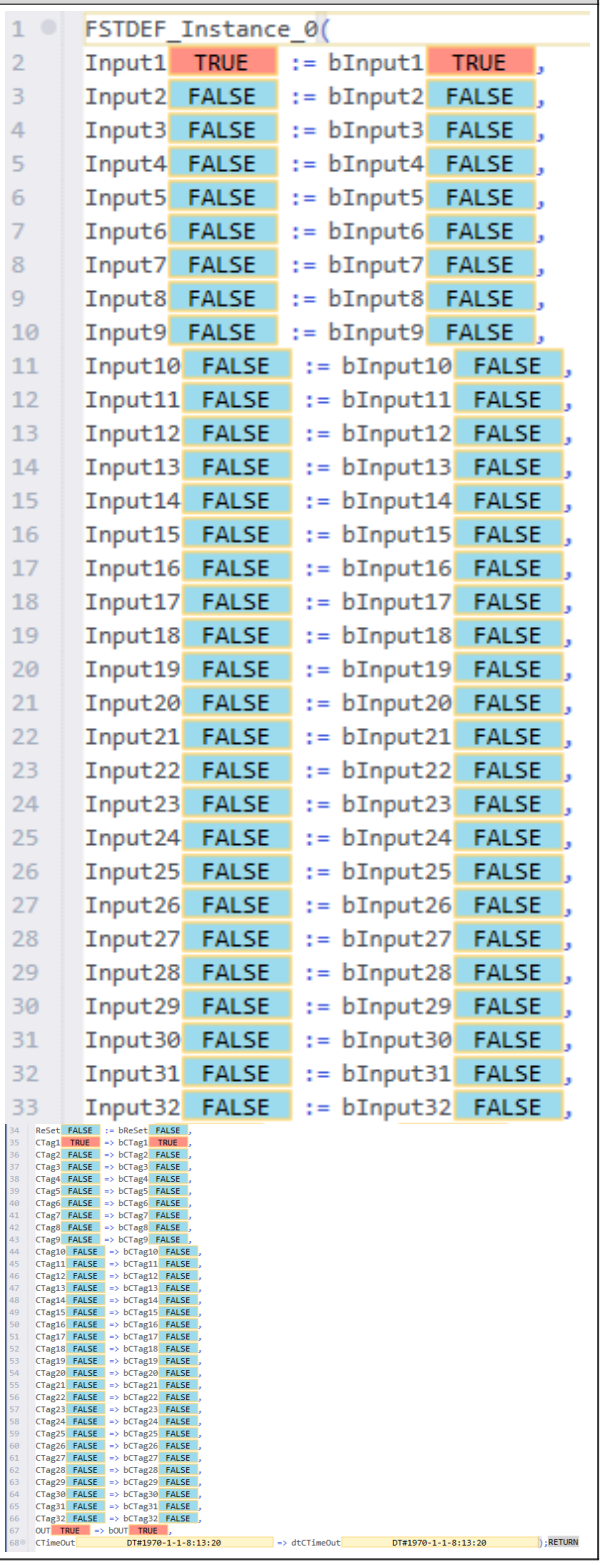
- The moment the total transition mark "OUT" changes from FALSE to TRUE (i.e., when the first transition occurs) is output via the "CTimeOut" parameter.
- When no transition is triggered, CTimeOut continues to update to the current system time.

- **Program example**

  The following example sets "Input1" to TRUE and all other parameters to default. In this case, "CTag1" becomes TRUE, "OUT" becomes TRUE, and "CTimeOut" outputs the moment when "Input1" transitions from 0 to 1.

Item	LD	ST

Define variables	VAR
	FSTDEF_Instance_0: FSTDEF;
	bInput1: BOOL := TRUE; bInput2: BOOL := FALSE; bInput3: BOOL := FALSE; bInput4: BOOL := FALSE;
	bInput5: BOOL := FALSE; bInput6: BOOL := FALSE; bInput7: BOOL := FALSE; bInput8: BOOL := FALSE;
	bInput9: BOOL := FALSE; bInput10: BOOL := FALSE; bInput11: BOOL := FALSE; bInput12: BOOL := FALSE;
	bInput13: BOOL := FALSE; bInput14: BOOL := FALSE; bInput15: BOOL := FALSE; bInput16: BOOL := FALSE;
	bInput17: BOOL := FALSE; bInput18: BOOL := FALSE; bInput19: BOOL := FALSE; bInput20: BOOL := FALSE;
	bInput21: BOOL := FALSE; bInput22: BOOL := FALSE; bInput23: BOOL := FALSE; bInput24: BOOL := FALSE;
	bInput25: BOOL := FALSE; bInput26: BOOL := FALSE; bInput27: BOOL := FALSE; bInput28: BOOL := FALSE;
	bInput29: BOOL := FALSE; bInput30: BOOL := FALSE; bInput31: BOOL := FALSE; bInput32: BOOL := FALSE;
	bReSet: BOOL := FALSE;
	bCTag1: BOOL; bCTag2: BOOL; bCTag3: BOOL; bCTag4: BOOL; bCTag5: BOOL; bCTag6: BOOL;
	bCTag7: BOOL; bCTag8: BOOL; bCTag9: BOOL; bCTag10: BOOL; bCTag11: BOOL; bCTag12: BOOL;
	bCTag13: BOOL; bCTag14: BOOL; bCTag15: BOOL; bCTag16: BOOL; bCTag17: BOOL; bCTag18: BOOL;
	bCTag19: BOOL; bCTag20: BOOL; bCTag21: BOOL; bCTag22: BOOL; bCTag23: BOOL; bCTag24: BOOL;
	bCTag25: BOOL; bCTag26: BOOL; bCTag27: BOOL; bCTag28: BOOL; bCTag29: BOOL; bCTag30: BOOL;
	bCTag31: BOOL; bCTag32: BOOL; bOUT: BOOL;
	dtCTimeOut: DATE_AND_TIME;
	END_VAR	

Item	LD	ST
Program		


```
FSTDEF_Instance_0(
Input1  TRUE  := bInput1  TRUE  ,
Input2  FALSE := bInput2  FALSE ,
Input3  FALSE := bInput3  FALSE ,
Input4  FALSE := bInput4  FALSE ,
Input5  FALSE := bInput5  FALSE ,
Input6  FALSE := bInput6  FALSE ,
Input7  FALSE := bInput7  FALSE ,
Input8  FALSE := bInput8  FALSE ,
Input9  FALSE := bInput9  FALSE ,
Input10 FALSE := bInput10 FALSE ,
Input11 FALSE := bInput11 FALSE ,
Input12 FALSE := bInput12 FALSE ,
Input13 FALSE := bInput13 FALSE ,
Input14 FALSE := bInput14 FALSE ,
Input15 FALSE := bInput15 FALSE ,
Input16 FALSE := bInput16 FALSE ,
Input17 FALSE := bInput17 FALSE ,
Input18 FALSE := bInput18 FALSE ,
Input19 FALSE := bInput19 FALSE ,
Input20 FALSE := bInput20 FALSE ,
Input21 FALSE := bInput21 FALSE ,
Input22 FALSE := bInput22 FALSE ,
Input23 FALSE := bInput23 FALSE ,
Input24 FALSE := bInput24 FALSE ,
Input25 FALSE := bInput25 FALSE ,
Input26 FALSE := bInput26 FALSE ,
Input27 FALSE := bInput27 FALSE ,
Input28 FALSE := bInput28 FALSE ,
Input29 FALSE := bInput29 FALSE ,
Input30 FALSE := bInput30 FALSE ,
Input31 FALSE := bInput31 FALSE ,
Input32 FALSE := bInput32 FALSE ,
ReSet   FALSE := bReSet   FALSE ,
CTag1   TRUE  => bCTag1   TRUE  ,
CTag2   FALSE => bCTag2   FALSE ,
CTag3   FALSE => bCTag3   FALSE ,
CTag4   FALSE => bCTag4   FALSE ,
CTag5   FALSE => bCTag5   FALSE ,
CTag6   FALSE => bCTag6   FALSE ,
CTag7   FALSE => bCTag7   FALSE ,
CTag8   FALSE => bCTag8   FALSE ,
CTag9   FALSE => bCTag9   FALSE ,
CTag10  FALSE => bCTag10  FALSE ,
CTag11  FALSE => bCTag11  FALSE ,
CTag12  FALSE => bCTag12  FALSE ,
CTag13  FALSE => bCTag13  FALSE ,
CTag14  FALSE => bCTag14  FALSE ,
CTag15  FALSE => bCTag15  FALSE ,
CTag16  FALSE => bCTag16  FALSE ,
CTag17  FALSE => bCTag17  FALSE ,
CTag18  FALSE => bCTag18  FALSE ,
CTag19  FALSE => bCTag19  FALSE ,
CTag20  FALSE => bCTag20  FALSE ,
CTag21  FALSE => bCTag21  FALSE ,
CTag22  FALSE => bCTag22  FALSE ,
CTag23  FALSE => bCTag23  FALSE ,
CTag24  FALSE => bCTag24  FALSE ,
CTag25  FALSE => bCTag25  FALSE ,
CTag26  FALSE => bCTag26  FALSE ,
CTag27  FALSE => bCTag27  FALSE ,
CTag28  FALSE => bCTag28  FALSE ,
CTag29  FALSE => bCTag29  FALSE ,
CTag30  FALSE => bCTag30  FALSE ,
CTag31  FALSE => bCTag31  FALSE ,
CTag32  FALSE => bCTag32  FALSE ,
OUT     TRUE  => bOUT     TRUE  ,
CTimeOut        DT#1970-1-1-8:13:20 => dtCTimeOut        DT#1970-1-1-8:13:20 );RETURN
```

Item	LD	ST

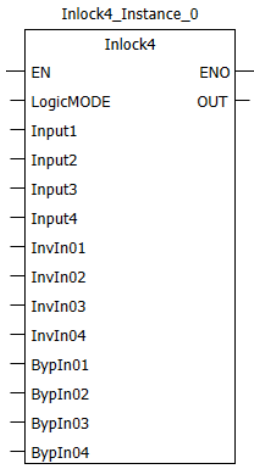
Operation result

	Name	Data type	Address	Value
1	FSTDEF_Instance_0	FSTDEF		
2	bInput1	BOOL		TRUE
3	bInput2	BOOL		FALSE
4	bInput3	BOOL		FALSE
5	bInput4	BOOL		FALSE
6	bInput5	BOOL		FALSE
7	bInput6	BOOL		FALSE
8	bInput7	BOOL		FALSE
9	bInput8	BOOL		FALSE
10	bInput9	BOOL		FALSE
11	bInput10	BOOL		FALSE
12	bInput11	BOOL		FALSE
13	bInput12	BOOL		FALSE
14	bInput13	BOOL		FALSE
15	bInput14	BOOL		FALSE
16	bInput15	BOOL		FALSE
17	bInput16	BOOL		FALSE
18	bInput17	BOOL		FALSE
19	bInput18	BOOL		FALSE
20	bInput19	BOOL		FALSE
21	bInput20	BOOL		FALSE
22	bInput21	BOOL		FALSE
23	bInput22	BOOL		FALSE
24	bInput23	BOOL		FALSE
25	bInput24	BOOL		FALSE
26	bInput25	BOOL		FALSE
27	bInput26	BOOL		FALSE
28	bInput27	BOOL		FALSE
29	bInput28	BOOL		FALSE
30	bInput29	BOOL		FALSE
31	bInput30	BOOL		FALSE
32	bInput31	BOOL		FALSE
33	bInput32	BOOL		FALSE
34	bReSet	BOOL		FALSE
35	bCTag1	BOOL		TRUE
36	bCTag2	BOOL		FALSE
37	bCTag3	BOOL		FALSE
38	bCTag4	BOOL		FALSE
39	bCTag5	BOOL		FALSE
40	bCTag6	BOOL		FALSE
41	bCTag7	BOOL		FALSE
42	bCTag8	BOOL		FALSE
43	bCTag9	BOOL		FALSE
44	bCTag10	BOOL		FALSE
45	bCTag11	BOOL		FALSE
46	bCTag12	BOOL		FALSE
47	bCTag13	BOOL		FALSE
48	bCTag14	BOOL		FALSE
49	bCTag15	BOOL		FALSE
50	bCTag16	BOOL		FALSE
51	bCTag17	BOOL		FALSE
52	bCTag18	BOOL		FALSE
53	bCTag19	BOOL		FALSE
54	bCTag20	BOOL		FALSE
55	bCTag21	BOOL		FALSE
56	bCTag22	BOOL		FALSE
57	bCTag23	BOOL		FALSE
58	bCTag24	BOOL		FALSE
59	bCTag25	BOOL		FALSE
60	bCTag26	BOOL		FALSE
61	bCTag27	BOOL		FALSE
62	bCTag28	BOOL		FALSE
63	bCTag29	BOOL		FALSE
64	bCTag30	BOOL		FALSE
65	bCTag31	BOOL		FALSE
66	bCTag32	BOOL		FALSE
67	bOUT	BOOL		TRUE
68	dtCTimeOut	DATE_AND_TIME		DT#1970-1-1-8:13:20

18.2.14.9 Inlock4

Inlock4 is used as an interlock block.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
Inlock4	Interlock block	FB		Inlock4_Instance_0(LogicMODE := , Input1 := , Input2 := , Input3 := , Input4 := , InvIn01 := , InvIn02 := , InvIn03 := , InvIn04 := , BypIn01 := , BypIn02 := , BypIn03 := , BypIn04 := , OUT =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	LogicMODE	Logic operation mode	Follows data type	-	Optional operation mode: 0: OR 1: AND
	Input1 to Input4	Input 1 to Input 4	Follows data type	-	Input 1 to Input 4
	InvIn01 to InvIn04	Input 1 inverted to Input 4 inverted	Follows data type	-	Input 1 inverted to Input 4 inverted

Input/Output	Variable	Name	Valid Range	Default Value	Description
	BypIn01 to BypIn04	Input 1 bypassed to Input 4 bypassed	Follows data type	-	Input 1 bypassed to Input 4 bypassed
Output	OUT	Output	Follows data type	-	Output

Variable types

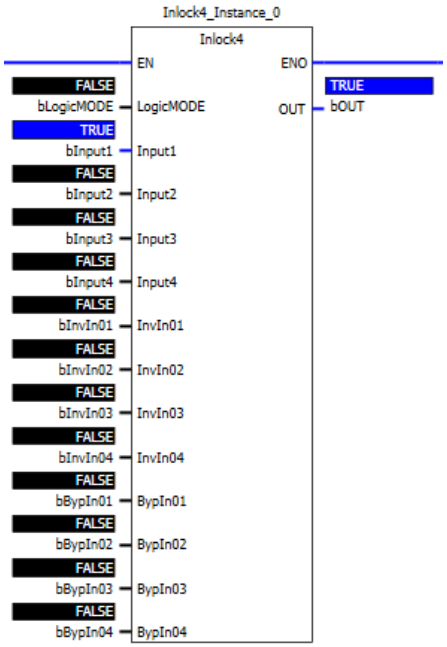
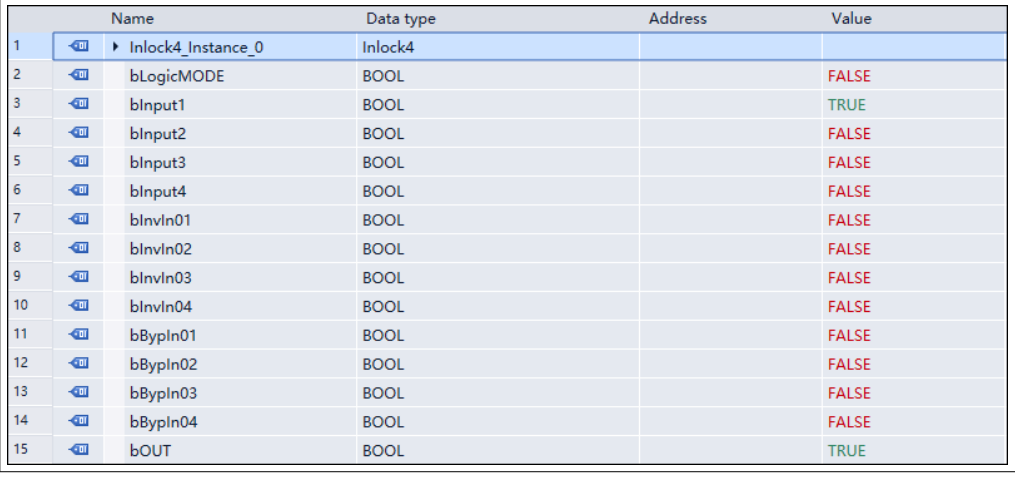
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
LogicMODE	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Input1 to Input4	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
InvIn01 to InvIn04	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
BypIn01 to BypIn04	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Performs interlock binary logic operation (AND/OR) on up to four input signals. Each signal includes independent inversion and bypass function (bypassed signal is excluded from the operation).

Item	LD	ST

Define variables	VAR
	Inlock4_Instance_0: Inlock4;
	bLogicMODE : BOOL := FALSE;
	bInput1 : BOOL := TRUE;
	bInput2 : BOOL := FALSE;
	bInput3 : BOOL := FALSE;
	bInput4 : BOOL := FALSE;
	bInvIn01 : BOOL := FALSE;
	bInvIn02 : BOOL := FALSE;
	bInvIn03 : BOOL := FALSE;
	bInvIn04 : BOOL := FALSE;
	bBypIn01 : BOOL := FALSE;
	bBypIn02 : BOOL := FALSE;
	bBypIn03 : BOOL := FALSE;
	bBypIn04 : BOOL := FALSE;
	bOUT : BOOL;
	END_VAR		

Program		Inlock4_Instance_0(LogicMODE := , Input1 TRUE := bInput1 TRUE , Input2 FALSE := bInput2 FALSE , Input3 FALSE := bInput3 FALSE , Input4 FALSE := bInput4 FALSE , InvIn01 FALSE := bInvIn01 FALSE , InvIn02 FALSE := bInvIn02 FALSE , InvIn03 FALSE := bInvIn03 FALSE , InvIn04 FALSE := bInvIn04 FALSE , BypIn01 FALSE := bBypIn01 FALSE , BypIn02 FALSE := bBypIn02 FALSE , BypIn03 FALSE := bBypIn03 FALSE , BypIn04 FALSE := bBypIn04 FALSE , OUT TRUE => bOUT TRUE);RETURN
Operation result		

	Name	Data type	Address	Value
1	Inlock4_Instance_0	Inlock4		
2	bLogicMODE	BOOL		FALSE
3	bInput1	BOOL		TRUE
4	bInput2	BOOL		FALSE
5	bInput3	BOOL		FALSE
6	bInput4	BOOL		FALSE
7	bInvIn01	BOOL		FALSE
8	bInvIn02	BOOL		FALSE
9	bInvIn03	BOOL		FALSE
10	bInvIn04	BOOL		FALSE
11	bBypIn01	BOOL		FALSE
12	bBypIn02	BOOL		FALSE
13	bBypIn03	BOOL		FALSE
14	bBypIn04	BOOL		FALSE
15	bOUT	BOOL		TRUE

18.2.15 Array Comparison

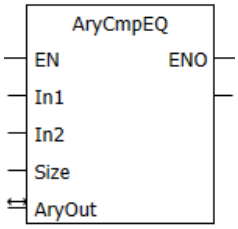
18.2.15.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.15.2 AryCmpEQ	FC	Array equality comparison	ArrayUtils	Library version 1.0.0.0 and later
18.2.15.3 AryCmpNE	FC	Array inequality comparison		Library version 1.0.0.0 and later
18.2.15.4 AryCmpEQV	FC	Array and value equality comparison		Library version 1.0.0.0 and later
18.2.15.5 AryCmpNEV	FC	Array and value inequality comparison		Library version 1.0.0.0 and later
18.2.15.6 AryCmpLT	FC	Array comparison (less than)		Library version 1.0.0.0 and later
18.2.15.7 AryCmpLE	FC	Array comparison (less than or equal to)		Library version 1.0.0.0 and later
18.2.15.8 AryCmpGT	FC	Array comparison (greater than)		Library version 1.0.0.0 and later
18.2.15.9 AryCmpGE	FC	Array comparison (greater than or equal to)		Library version 1.0.0.0 and later
18.2.15.10 AryCmpLTV	FC	Array and value comparison (less than)		Library version 1.0.0.0 and later
18.2.15.11 AryCmpLEV	FC	Array and value comparison (less than or equal to)		Library version 1.0.0.0 and later
18.2.15.12 AryCmpGTV	FC	Array and value comparison (greater than)		Library version 1.0.0.0 and later
18.2.15.13 AryCmpGEV	FC	Array and value comparison (greater than or equal to)		Library version 1.0.0.0 and later

18.2.15.2 AryCmpEQ

Compares whether the elements of two arrays are equal.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryCmpEQ	Array equality comparison	FC		:= AryCmpEQ(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In1[ ] array	Comparison array 1	Follows data type	-	Array 1 to compare
	In2[ ] array	Comparison array 2	Follows data type	-	Array 2 to compare
	Size	Number of elements to compare	Follows data type	1	Number of elements to compare
Input/ Output	AryOut[ ] array	Comparison result array	Follows data type	-	Comparison result array
Output	AryCmpEQ	Function return value	Follows data type	-	TRUE: Normal execution completed. FALSE: The instruction was not executed due to an error.

Variable types

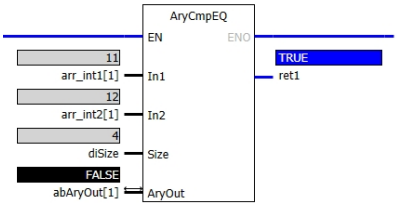
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Array In1[ ]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[ ] array	Array with the same data type as In1[ ]																			
Size	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

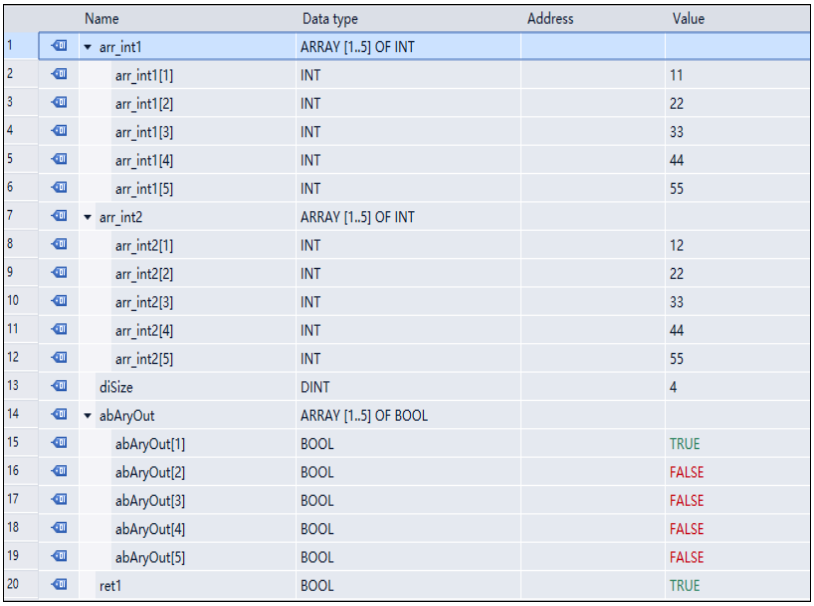
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
AryOut[ ] array	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
AryCmpEQ	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**
  - This instruction compares the elements with the same indices in two arrays (In1[0] to In1[Size-1] and In2[0] to In2[Size-1]) to determine whether they are equal. The comparison results are stored in elements with the corresponding indices in the array (AryOut[0] to AryOut[Size–1]).
  - When In1[i] = In2[i], AryOut[i] is TRUE; otherwise, it is FALSE.
- **Program example**

	LD	ST

Define variables	VAR
	ret1: BOOL;
	arr_int1: ARRAY[1..5] OF INT := [11,22,33,44,55];
	arr_int2: ARRAY[1..5] OF INT := [12,22,33,45,55];
	disize: DINT := 4;
	abArrout: ARRAY[1..5] OF BOOL;
	END_VAR	

Program		ret1 TRUE := AryCmpEQ(In1 := arr_int1[1] 11 , In2 := arr_int2[1] 12 , Size := diSize 4 , AryOut := abAryOut[1] FALSE);RETURN

	LD	ST
Operation result		

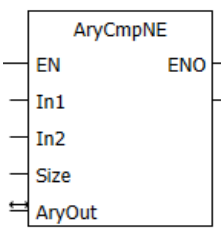
	Name	Data type	Address	Value
1	arr_int1	ARRAY [1..5] OF INT		
2	arr_int1[1]	INT		11
3	arr_int1[2]	INT		22
4	arr_int1[3]	INT		33
5	arr_int1[4]	INT		44
6	arr_int1[5]	INT		55
7	arr_int2	ARRAY [1..5] OF INT		
8	arr_int2[1]	INT		12
9	arr_int2[2]	INT		22
10	arr_int2[3]	INT		33
11	arr_int2[4]	INT		44
12	arr_int2[5]	INT		55
13	diSize	DINT		4
14	abAryOut	ARRAY [1..5] OF BOOL		
15	abAryOut[1]	BOOL		TRUE
16	abAryOut[2]	BOOL		FALSE
17	abAryOut[3]	BOOL		FALSE
18	abAryOut[4]	BOOL		FALSE
19	abAryOut[5]	BOOL		FALSE
20	ret1	BOOL		TRUE

- **Precautions**
  - Use the same data type for In1[ ] and In2[ ]; otherwise, a compilation fault will occur.
  - If In1[ ] and In2[ ] are real numbers that contain non-terminating decimal numbers, the processing results may be unexpected due to rounding errors.
  - When the value of Size is 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - When the value of Size exceeds the In1[ ], In2[ ], or AryOut[ ] array, the return value is FALSE and AryOut[ ] remains unchanged.
  - When the value of Size is less than 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.15.3 AryCmpNE

Compares whether the elements of two arrays are unequal.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryCmpNE	Array inequality comparison	FC		:= AryCmpNE(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In1[ ] array	Comparison array 1	Follows data type	-	Array 1 to compare
	In2[ ] array	Comparison array 2	Follows data type	-	Array 2 to compare
	Size	Number of elements to compare	Follows data type	1	Number of elements to compare
Input/ Output	AryOut[ ] array	Comparison result array	Follows data type	-	Comparison result array
Output	AryCmpNE	Function return value	Follows data type	-	TRUE: Normal execution completed. FALSE: The instruction was not executed due to an error.

Variable types

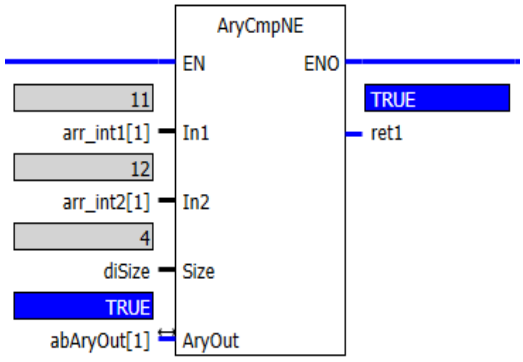
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Array In1[ ]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[ ] array	Array with the same data type as In1[ ]																			
Size	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
AryOut[ ] array	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
AryCmpNE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**
  - This instruction compares the elements with the same indices in two arrays (In1[0] to In1[Size-1] and In2[0] to In2[Size-1]) to determine whether they are unequal. The comparison results are stored in elements with the corresponding indices in the array (AryOut[0] to AryOut[Size–1]).
  - When In1[i] is not equal to In2[i], AryOut[i] is TRUE; otherwise, it is FALSE.
- **Program example**

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    ret1: BOOL;
    arr_int1: ARRAY[1..5] OF INT := [11,22,33,44,55];
    arr_int2: ARRAY[1..5] OF INT := [12,22,33,45,55];
    disize: DINT := 4;
    abArrout: ARRAY[1..5] OF BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
ret1 TRUE := AryCmpNE(
        In1 := arr_int1[1] 11 ,
        In2 := arr_int2[1] 12 ,
        Size := diSize 4 ,
        AryOut := abAryOut[1] TRUE );RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

	Name	Data type	Address	Value
1	arr_int1	ARRAY [1..5] OF INT		
2	arr_int1[1]	INT		11
3	arr_int1[2]	INT		22
4	arr_int1[3]	INT		33
5	arr_int1[4]	INT		44
6	arr_int1[5]	INT		55
7	arr_int2	ARRAY [1..5] OF INT		
8	arr_int2[1]	INT		12
9	arr_int2[2]	INT		22
10	arr_int2[3]	INT		33
11	arr_int2[4]	INT		44
12	arr_int2[5]	INT		55
13	diSize	DINT		4
14	abAryOut	ARRAY [1..5] OF BOOL		
15	abAryOut[1]	BOOL		TRUE
16	abAryOut[2]	BOOL		FALSE
17	abAryOut[3]	BOOL		FALSE
18	abAryOut[4]	BOOL		FALSE
19	abAryOut[5]	BOOL		FALSE
20	ret1	BOOL		TRUE

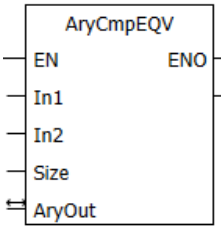
</td></tr>
</table>

- **Precautions**
  - Use the same data type for In1[ ] and In2[ ]; otherwise, a compilation fault will occur.
  - If In1[ ] and In2[ ] are real numbers that contain non-terminating decimal numbers, the processing results may be unexpected due to rounding errors.
  - When the value of Size is 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - When the value of Size exceeds the In1[ ], In2[ ], or AryOut[ ] array, the return value is FALSE and AryOut[ ] remains unchanged.
  - When the value of Size is less than 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.15.4 AryCmpEQV

Compares whether each element of the array is equal to the value of the comparison variable.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryCmpEQV	Array and value equality comparison	FC		:= AryCmpEQV(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In1[ ] array	Comparison array 1	Follows data type	-	Array 1 to compare
	In2	Comparison variable	Follows data type	-	Variable to compare
	Size	Number of elements to compare	Follows data type	1	Number of elements to compare
Input/ Output	AryOut[ ] array	Comparison result array	Follows data type	-	Comparison result array
Output	AryCmpEQV	Function return value	Follows data type	-	TRUE: Normal execution completed. FALSE: The instruction was not executed due to an error.

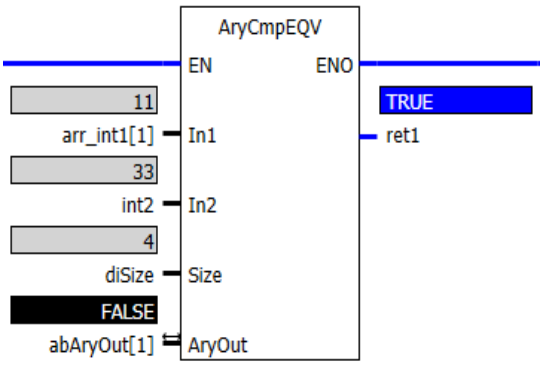
Variable types

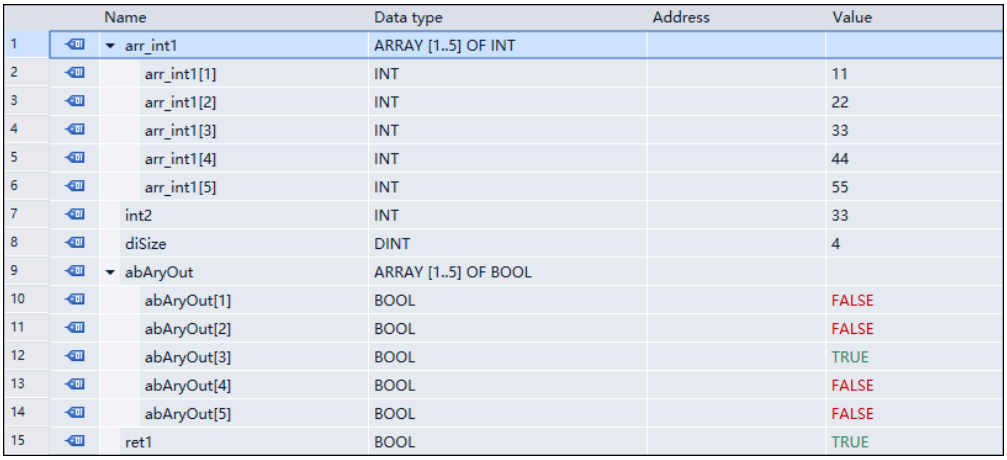
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Array In1[ ]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2	Same data type as the elements of In1[ ]																			
Size	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
AryOut[ ] array	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
AryCmpEQV	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**
  - This instruction compares the value of In2 with the array (In1[0] to In1[Size-1]) to determine whether they are equal. The comparison results are stored in elements with the corresponding indices in the array (AryOut[0] to AryOut[Size−1]).
  - When In1[i] is equal to In2, AryOut[i] is TRUE; otherwise, it is FALSE.
- **Program example**

	LD	ST

Define variables	VAR
	ret1:BOOL;
	arr_int1:ARRAY[1..5] OF INT := [11,22,33,44,55];
	int2 :INT := 33;
	disize :DINT := 4;
	abArrout:Array[1..5] OF BOOL;
	END_VAR	

Program		ret1 TRUE := AryCmpEQV(In1 := arr_int1[1] 11 , In2 := int2 33 , Size := diSize 4 , AryOut := abAryOut[1] FALSE);RETURN

	LD	ST
Operation result		

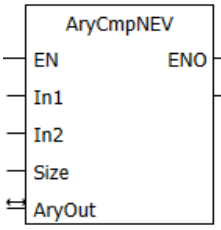
	Name	Data type	Address	Value
1	arr_int1	ARRAY [1..5] OF INT		
2	arr_int1[1]	INT		11
3	arr_int1[2]	INT		22
4	arr_int1[3]	INT		33
5	arr_int1[4]	INT		44
6	arr_int1[5]	INT		55
7	int2	INT		33
8	diSize	DINT		4
9	abAryOut	ARRAY [1..5] OF BOOL		
10	abAryOut[1]	BOOL		FALSE
11	abAryOut[2]	BOOL		FALSE
12	abAryOut[3]	BOOL		TRUE
13	abAryOut[4]	BOOL		FALSE
14	abAryOut[5]	BOOL		FALSE
15	ret1	BOOL		TRUE

- **Precautions**
  - Use the same data type for In1[ ] and In2[ ]; otherwise, a compilation fault will occur.
  - If In1[ ] and In2[ ] are real numbers that contain non-terminating decimal numbers, the processing results may be unexpected due to rounding errors.
  - When the value of Size is 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - When the value of Size exceeds the In1[ ] or AryOut[ ] array, the return value is FALSE and AryOut[ ] remains unchanged.
  - When the value of Size is less than 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.15.5 AryCmpNEV

Compares whether each element of the array is unequal to the value of the comparison variable.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryCmpNEV	Array and value inequality comparison	FC		:= AryCmpNEV(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

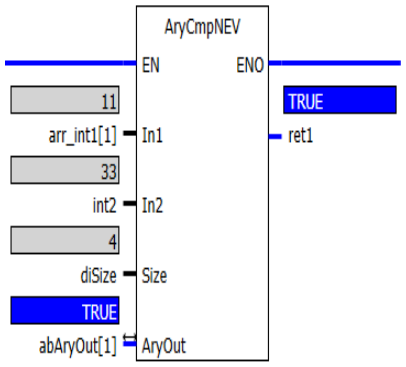
  Variable

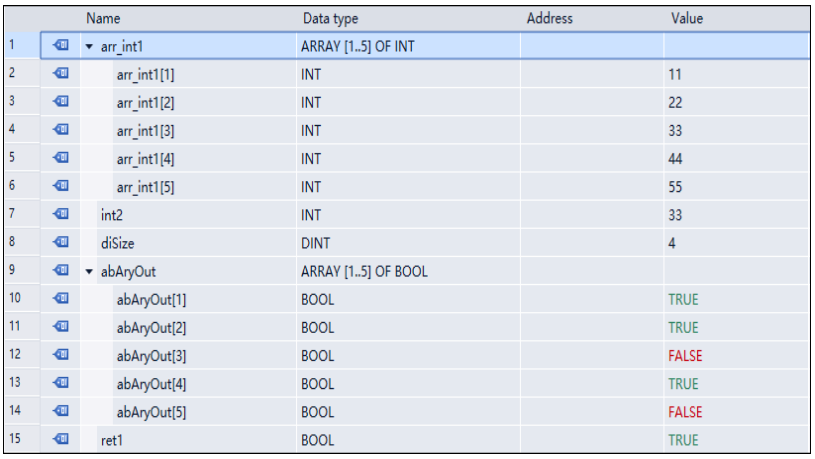
Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In1[ ] array	Comparison array 1	Follows data type	-	Array 1 to compare
	In2	Comparison variable	Follows data type	-	Variable to compare
	Size	Number of elements to compare	Follows data type	1	Number of elements to compare
Input/ Output	AryOut[ ] array	Comparison result array	Follows data type	-	Comparison result array
Output	AryCmpNEV	Function return value	Follows data type	-	TRUE: Normal execution completed. FALSE: The instruction was not executed due to an error.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Array In1[ ]	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2	Same data type as the elements of In1[ ]																			
Size	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
AryOut[ ] array	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
AryCmpNEV	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**
  - This instruction compares the value of In2 with the array (In1[0] to In1[Size-1]) to determine whether they are unequal. The comparison results are stored in elements with the corresponding indices in the array (AryOut[0] to AryOut[Size–1]).
  - When In1[i] is not equal to In2, AryOut[i] is TRUE; otherwise, it is FALSE.
- **Program example**

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">
<pre>
VAR
    ret1:BOOL;
    arr_int1:ARRAY[1..5] OF INT := [11,22,33,44,55];
    int2 :INT := 33;
    disize :DINT := 4;
    abArrout:Array[1..5] OF BOOL;
END_VAR
</pre>
</td></tr>
<tr><td>Program</td><td>


</td><td>
<pre>
ret1 TRUE := AryCmpNEV(
    In1 := arr_int1[1] 11 ,
    In2 := int2 33 ,
    Size := diSize 4 ,
    AryOut := abAryOut[1] TRUE);RETURN
</pre>
</td></tr>
<tr><td>Operation result</td><td colspan="2">


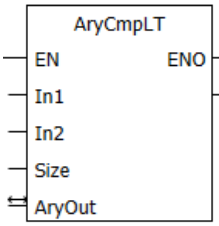
</td></tr>
</table>

- **Precautions**
  - Use the same data type for In1[ ] and In2[ ]; otherwise, a compilation fault will occur.
  - If In1[ ] and In2[ ] are real numbers that contain non-terminating decimal numbers, the processing results may be unexpected due to rounding errors.
  - When the value of Size is 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - When the value of Size exceeds the In1[ ] or AryOut[ ] array, the return value is FALSE and AryOut[ ] remains unchanged.
  - When the value of Size is less than 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.15.6 AryCmpLT

Performs a less-than comparison between the values of elements in two arrays.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryCmpLT	Array comparison (less than)	FC		:= AryCmpLT(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In1[ ] array	Comparison array 1	Follows data type	-	Array 1 to compare
	In2[ ] array	Comparison array 2	Follows data type	-	Array 2 to compare
	Size	Number of elements to compare	Follows data type	1	Number of elements to compare
Input/ Output	AryOut[ ] array	Comparison result array	Follows data type	-	Comparison result array
Output	AryCmpLT	Function return value	Follows data type	-	TRUE: Normal execution completed. FALSE: The instruction was not executed due to an error.

Variable types

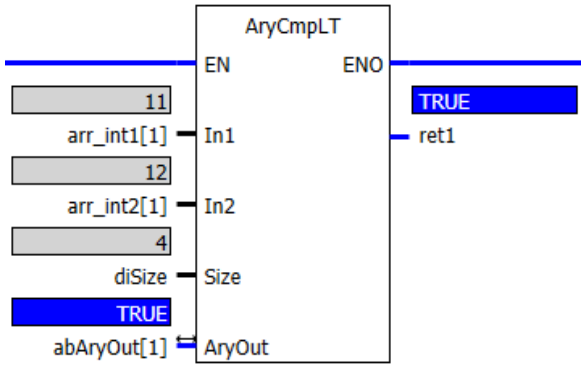
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1[ ] array	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[ ] array	Array with the same data type as In1[ ]																			
Size	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-

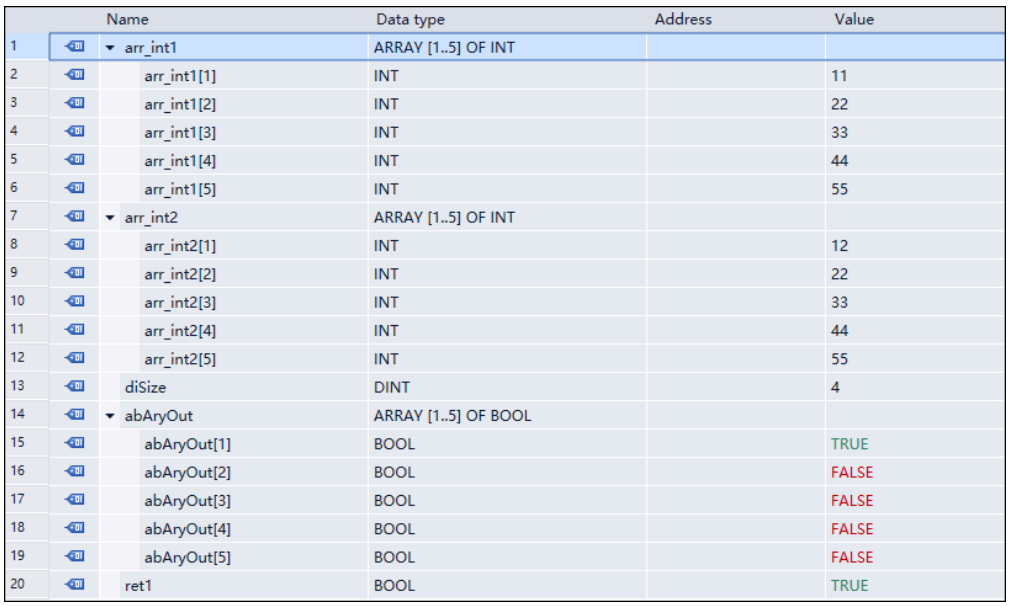
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
AryOut[ ] array	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
AryCmpLT	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**
  - This instruction performs a less-than comparison between the elements with the same indices in two arrays (In1[0] to In1[Size-1] and In2[0] to In2[Size-1]). The comparison results are stored in elements with the corresponding indices in the array (AryOut[0] to AryOut[Size–1]).
  - When In1[i] < In2[i], AryOut[i] is TRUE; otherwise, it is FALSE.
- **Program example**

	LD	ST

Define variables	VAR
	ret1: BOOL;
	arr_int1: ARRAY[1..5] OF INT := [11,22,33,44,55];
	arr_int2: ARRAY[1..5] OF INT := [12,22,33,45,55];
	disize: DINT := 4;
	abArrout: ARRAY[1..5] OF BOOL;
	END_VAR	

Program		ret1 TRUE := AryCmpLT(In1 := arr_int1[1] 11 , In2 := arr_int2[1] 12 , Size := diSize 4 , AryOut := abAryOut[1] TRUE);RETURN

	LD	ST
Operation result		

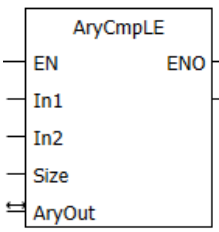
	Name	Data type	Address	Value
1	arr_int1	ARRAY [1..5] OF INT		
2	arr_int1[1]	INT		11
3	arr_int1[2]	INT		22
4	arr_int1[3]	INT		33
5	arr_int1[4]	INT		44
6	arr_int1[5]	INT		55
7	arr_int2	ARRAY [1..5] OF INT		
8	arr_int2[1]	INT		12
9	arr_int2[2]	INT		22
10	arr_int2[3]	INT		33
11	arr_int2[4]	INT		44
12	arr_int2[5]	INT		55
13	diSize	DINT		4
14	abAryOut	ARRAY [1..5] OF BOOL		
15	abAryOut[1]	BOOL		TRUE
16	abAryOut[2]	BOOL		FALSE
17	abAryOut[3]	BOOL		FALSE
18	abAryOut[4]	BOOL		FALSE
19	abAryOut[5]	BOOL		FALSE
20	ret1	BOOL		TRUE

- **Precautions**
  - Use the same data type for In1[ ] and In2[ ]; otherwise, a compilation fault will occur.
  - If In1[ ] and In2[ ] are real numbers that contain non-terminating decimal numbers, the processing results may be unexpected due to rounding errors.
  - When the value of Size is 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - When the value of Size exceeds the In1[ ], In2[ ], or AryOut[ ] array, the return value is FALSE and AryOut[ ] remains unchanged.
  - When the value of Size is less than 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.15.7 AryCmpLE

Performs a less-than or equal-to comparison between the values of elements in two arrays.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryCmpLE	Array comparison (less than or equal to)	FC		:= AryCmpLE(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In1[ ] array	Comparison array 1	Follows data type	-	Array 1 to compare
	In2[ ] array	Comparison array 2	Follows data type	-	Array 2 to compare
	Size	Number of elements to compare	Follows data type	1	Number of elements to compare
Input/ Output	AryOut[ ] array	Comparison result array	Follows data type	-	Comparison result array
Output	AryCmpLE	Function return value	Follows data type	-	TRUE: Normal execution completed. FALSE: The instruction was not executed due to an error.

Variable types

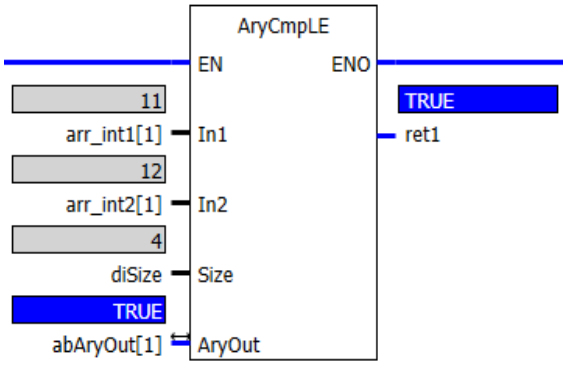
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1[ ] array	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[ ] array	Array with the same data type as In1[ ]																			
Size	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
AryOut[ ] array	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
AryCmpLE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**
  - This instruction performs a less-than or equal-to comparison between the elements with the same indices in two arrays (In1[0] to In1[Size-1] and In2[0] to In2[Size-1]). The comparison results are stored in elements with the corresponding indices in the array (AryOut[0] to AryOut[Size–1]).
  - When In1[i] <= In2[i], AryOut[i] is TRUE; otherwise, it is FALSE.
- **Program example**

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    ret1: BOOL;
    arr_int1: ARRAY[1..5] OF INT := [11,22,33,44,55];
    arr_int2: ARRAY[1..5] OF INT := [12,22,33,45,55];
    disize: DINT := 4;
    abArrout: ARRAY[1..5] OF BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
ret1 TRUE := AryCmpLE(
        In1 := arr_int1[1] 11 ,
        In2 := arr_int2[1] 12 ,
        Size := diSize 4 ,
        AryOut := abAryOut[1] TRUE );RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

	Name	Data type	Address	Value
1	arr_int1	ARRAY [1..5] OF INT		
2	arr_int1[1]	INT		11
3	arr_int1[2]	INT		22
4	arr_int1[3]	INT		33
5	arr_int1[4]	INT		44
6	arr_int1[5]	INT		55
7	arr_int2	ARRAY [1..5] OF INT		
8	arr_int2[1]	INT		12
9	arr_int2[2]	INT		22
10	arr_int2[3]	INT		33
11	arr_int2[4]	INT		44
12	arr_int2[5]	INT		55
13	diSize	DINT		4
14	abAryOut	ARRAY [1..5] OF BOOL		
15	abAryOut[1]	BOOL		TRUE
16	abAryOut[2]	BOOL		TRUE
17	abAryOut[3]	BOOL		TRUE
18	abAryOut[4]	BOOL		TRUE
19	abAryOut[5]	BOOL		FALSE
20	ret1	BOOL		TRUE

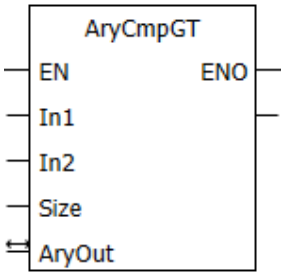
</td></tr>
</table>

- **Precautions**
  - Use the same data type for In1[ ] and In2[ ]; otherwise, a compilation fault will occur.
  - If In1[ ] and In2[ ] are real numbers that contain non-terminating decimal numbers, the processing results may be unexpected due to rounding errors.
  - When the value of Size is 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - When the value of Size exceeds the In1[ ], In2[ ], or AryOut[ ] array, the return value is FALSE and AryOut[ ] remains unchanged.
  - When the value of Size is less than 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.15.8 AryCmpGT

Performs a greater-than comparison between the values of elements in two arrays.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryCmpGT	Array comparison (greater than)	FC		:= AryCmpGT(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In1[ ] array	Comparison array 1	Follows data type	-	Array 1 to compare
	In2[ ] array	Comparison array 2	Follows data type	-	Array 2 to compare
	Size	Number of elements to compare	Follows data type	1	Number of elements to compare
Input/ Output	AryOut[ ] array	Comparison result array	Follows data type	-	Comparison result array
Output	AryCmpGT	Function return value	Follows data type	-	TRUE: Normal execution completed. FALSE: The instruction was not executed due to an error.

Variable types

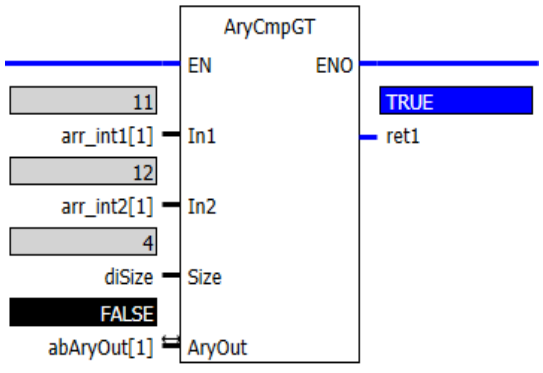
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1[ ] array	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-

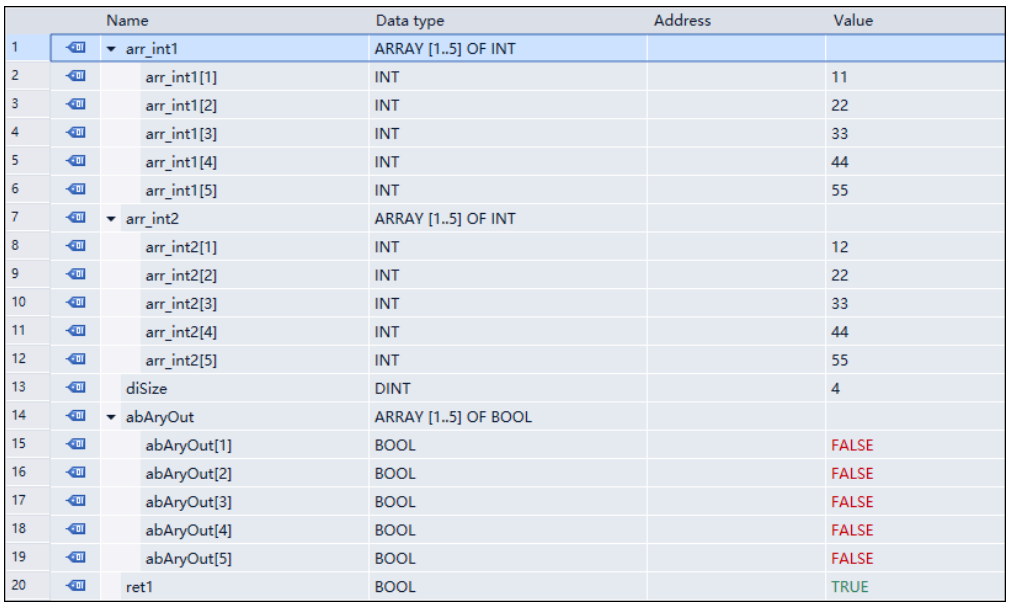
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In2[ ] array	Array with the same data type as In1[ ]																			
Size	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
AryOut [ ] array	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
AryCmpGT	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**
  - This instruction performs a greater-than comparison between the elements with the same indices in two arrays (In1[0] to In1[Size-1] and In2[0] to In2[Size-1]). The comparison results are stored in elements with the corresponding indices in the array (AryOut[0] to AryOut[Size–1]).
  - When In1[i] > In2[i], AryOut[i] is TRUE; otherwise, it is FALSE.
- **Program example**

	LD	ST

Define variables	VAR
	ret1: BOOL;
	arr_int1: ARRAY[1..5] OF INT := [11,22,33,44,55];
	arr_int2: ARRAY[1..5] OF INT := [12,22,33,45,55];
	disize: DINT := 4;
	abArrout: ARRAY[1..5] OF BOOL;
	END_VAR	

Program		ret1 TRUE := AryCmpGT(In1 := arr_int1[1] 11 , In2 := arr_int2[1] 12 , Size := diSize 4 , AryOut := abAryOut[1] FALSE);RETURN

	LD	ST
Operation result		

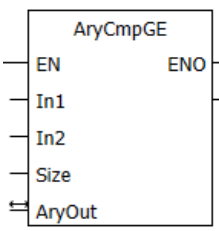
	Name	Data type	Address	Value
1	arr_int1	ARRAY [1..5] OF INT		
2	arr_int1[1]	INT		11
3	arr_int1[2]	INT		22
4	arr_int1[3]	INT		33
5	arr_int1[4]	INT		44
6	arr_int1[5]	INT		55
7	arr_int2	ARRAY [1..5] OF INT		
8	arr_int2[1]	INT		12
9	arr_int2[2]	INT		22
10	arr_int2[3]	INT		33
11	arr_int2[4]	INT		44
12	arr_int2[5]	INT		55
13	diSize	DINT		4
14	abAryOut	ARRAY [1..5] OF BOOL		
15	abAryOut[1]	BOOL		FALSE
16	abAryOut[2]	BOOL		FALSE
17	abAryOut[3]	BOOL		FALSE
18	abAryOut[4]	BOOL		FALSE
19	abAryOut[5]	BOOL		FALSE
20	ret1	BOOL		TRUE

- **Precautions**
  - Use the same data type for In1[ ] and In2[ ]; otherwise, a compilation fault will occur.
  - If In1[ ] and In2[ ] are real numbers that contain non-terminating decimal numbers, the processing results may be unexpected due to rounding errors.
  - When the value of Size is 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - When the value of Size exceeds the In1[ ], In2[ ], or AryOut[ ] array, the return value is FALSE and AryOut[ ] remains unchanged.
  - When the value of Size is less than 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.15.9 AryCmpGE

Performs a greater-than or equal-to comparison between the values of elements in two arrays.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryCmpGE	Array comparison (greater than or equal to)	FC		:= AryCmpGE(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In1[ ] array	Comparison array 1	Follows data type	-	Array 1 to compare
	In2[ ] array	Comparison array 2	Follows data type	-	Array 2 to compare
	Size	Number of elements to compare	Follows data type	1	Number of elements to compare
Input/ Output	AryOut[ ] array	Comparison result array	Follows data type	-	Comparison result array
Output	AryCmpGE	Function return value	Follows data type	-	TRUE: Normal execution completed. FALSE: The instruction was not executed due to an error.

Variable types

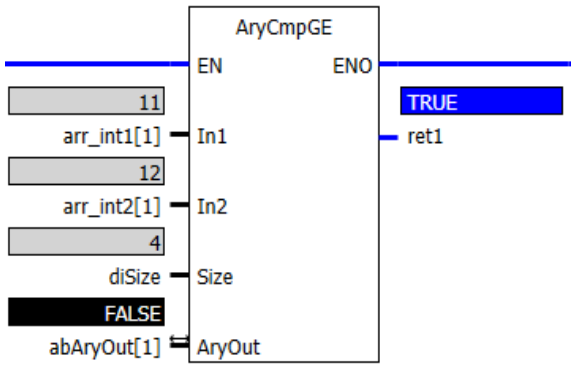
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1[ ] array	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2[ ] array	Array with the same data type as In1[ ]																			
Size	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
AryOut[ ] array	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
AryCmpGE	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**
  - This instruction performs a greater-than or equal-to comparison between the elements with the same indices in two arrays (In1[0] to In1[Size-1] and In2[0] to In2[Size-1]). The comparison results are stored in elements with the corresponding indices in the array (AryOut[0] to AryOut[Size–1]).
  - When In1[i] >= In2[i], AryOut[i] is TRUE; otherwise, it is FALSE.
- **Program example**

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    ret1: BOOL;
    arr_int1: ARRAY[1..5] OF INT := [11,22,33,44,55];
    arr_int2: ARRAY[1..5] OF INT := [12,22,33,45,55];
    disize: DINT := 4;
    abArrout: ARRAY[1..5] OF BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
ret1 TRUE := AryCmpGE(
    In1 := arr_int1[1] 11 ,
    In2 := arr_int2[1] 12 ,
    Size := diSize 4 ,
    AryOut := abAryOut[1] FALSE );RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

	Name	Data type	Address	Value
1	arr_int1	ARRAY [1..5] OF INT		
2	arr_int1[1]	INT		11
3	arr_int1[2]	INT		22
4	arr_int1[3]	INT		33
5	arr_int1[4]	INT		44
6	arr_int1[5]	INT		55
7	arr_int2	ARRAY [1..5] OF INT		
8	arr_int2[1]	INT		12
9	arr_int2[2]	INT		22
10	arr_int2[3]	INT		33
11	arr_int2[4]	INT		44
12	arr_int2[5]	INT		55
13	diSize	DINT		4
14	abAryOut	ARRAY [1..5] OF BOOL		
15	abAryOut[1]	BOOL		FALSE
16	abAryOut[2]	BOOL		TRUE
17	abAryOut[3]	BOOL		TRUE
18	abAryOut[4]	BOOL		TRUE
19	abAryOut[5]	BOOL		FALSE
20	ret1	BOOL		TRUE

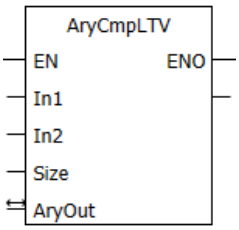
</td></tr>
</table>

- **Precautions**
  - Use the same data type for In1[ ] and In2[ ]; otherwise, a compilation fault will occur.
  - If In1[ ] and In2[ ] are real numbers that contain non-terminating decimal numbers, the processing results may be unexpected due to rounding errors.
  - When the value of Size is 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - When the value of Size exceeds the In1[ ], In2[ ], or AryOut[ ] array, the return value is FALSE and AryOut[ ] remains unchanged.
  - When the value of Size is less than 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.15.10 AryCmpLTV

Performs a less-than comparison between the values of elements in an array and the value of a variable.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryCmpLTV	Array and value comparison (less than)	FC		:= AryCmpLTV(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In1[ ] array	Comparison array	Follows data type	-	Array to compare
	In2	Comparison variable	Follows data type	-	Variable to compare
	Size	Number of elements to compare	Follows data type	1	Number of elements to compare
Input/ Output	AryOut[ ] array	Comparison result array	Follows data type	-	Comparison result array
Output	AryCmpLTV	Function return value	Follows data type	-	TRUE: Normal execution completed. FALSE: The instruction was not executed due to an error.

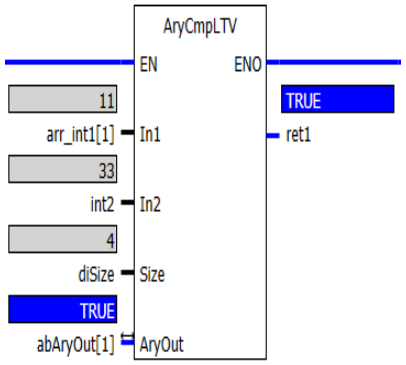
Variable types

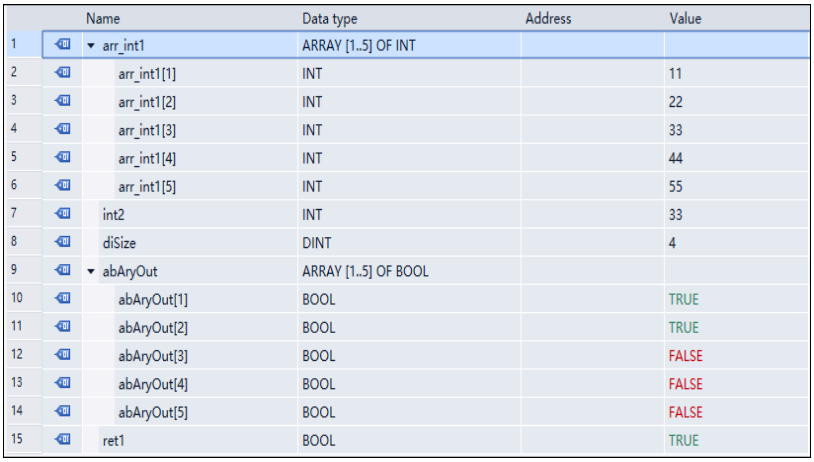
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1[ ] array	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2	Same data type as the elements of In1[ ]																			
Size	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
AryOut[ ] array	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
AryCmpLTV	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**
  - This instruction performs a less-than comparison between the values of elements in the array (In1[0] to In1[Size-1]) and the value of In2. The comparison results are stored in elements with the corresponding indices in the array (AryOut[0] to AryOut[Size−1]).
  - When In1[i] < In2, AryOut[i] is TRUE; otherwise, it is FALSE.
- **Program example**

	LD	ST

Define variables	VAR
	ret1:BOOL;
	arr_int1:ARRAY[1..5] OF INT := [11,22,33,44,55];
	int2 :INT := 33;
	disize :DINT := 4;
	abArrout:Array[1..5] OF BOOL;
	END_VAR	

Program		ret1 TRUE := AryCmpLTV(In1 := arr_int1[1] 11 , In2 := int2 33 , Size := diSize 4 , AryOut := abAryOut[1] TRUE);RETURN

	LD	ST
Operation result		

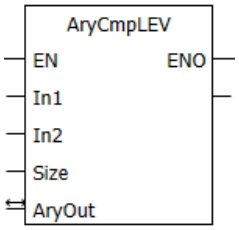
	Name	Data type	Address	Value
1	arr_int1	ARRAY [1..5] OF INT		
2	arr_int1[1]	INT		11
3	arr_int1[2]	INT		22
4	arr_int1[3]	INT		33
5	arr_int1[4]	INT		44
6	arr_int1[5]	INT		55
7	int2	INT		33
8	diSize	DINT		4
9	abAryOut	ARRAY [1..5] OF BOOL		
10	abAryOut[1]	BOOL		TRUE
11	abAryOut[2]	BOOL		TRUE
12	abAryOut[3]	BOOL		FALSE
13	abAryOut[4]	BOOL		FALSE
14	abAryOut[5]	BOOL		FALSE
15	ret1	BOOL		TRUE

- **Precautions**
  - Use the same data type for In1[ ] and In2[ ]; otherwise, a compilation fault will occur.
  - If In1[ ] and In2[ ] are real numbers that contain non-terminating decimal numbers, the processing results may be unexpected due to rounding errors.
  - When the value of Size is 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - When the value of Size exceeds the In1[ ] or AryOut[ ] array, the return value is FALSE and AryOut[ ] remains unchanged.
  - When the value of Size is less than 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.15.11 AryCmpLEV

Performs a less-than or equal-to comparison between the values of elements in an array and the value of a variable.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryCmpLEV	Array and value comparison (less than or equal to)	FC		:= AryCmpLEV(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In1[ ] array	Comparison array	Follows data type	-	Array to compare
	In2	Comparison variable	Follows data type	-	Variable to compare
	Size	Number of elements to compare	Follows data type	1	Number of elements to compare
Input/ Output	AryOut[ ] array	Comparison result array	Follows data type	-	Comparison result array
Output	AryCmpLEV	Function return value	Follows data type	-	TRUE: Normal execution completed. FALSE: The instruction was not executed due to an error.

Variable types

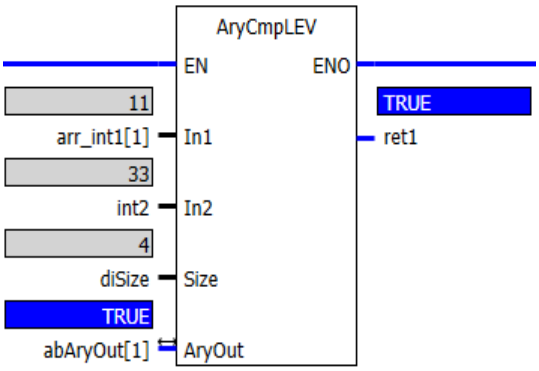
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1[ ] array	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2	Same data type as the elements of In1[ ]																			
Size	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
AryOut[ ] array	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
AryCmpLEV	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**
  - This instruction performs a less-than or equal-to comparison between the values of elements in the array (In1[0] to In1[Size-1]) and the value of In2. The comparison results are stored in elements with the corresponding indices in the array (AryOut[0] to AryOut[Size−1]).
  - When In1[i] <= In2, AryOut[i] is TRUE; otherwise, it is FALSE.
- **Program example**

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    ret1:BOOL;
    arr_int1:ARRAY[1..5] OF INT := [11,22,33,44,55];
    int2 :INT := 33;
    disize :DINT := 4;
    abArrout:Array[1..5] OF BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
ret1 TRUE := AryCmpLEV(
        In1 := arr_int1[1] 11 ,
        In2 := int2 33 ,
        Size := diSize 4 ,
        AryOut := abAryOut[1] TRUE );RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

	Name	Data type	Address	Value
1	arr_int1	ARRAY [1..5] OF INT		
2	arr_int1[1]	INT		11
3	arr_int1[2]	INT		22
4	arr_int1[3]	INT		33
5	arr_int1[4]	INT		44
6	arr_int1[5]	INT		55
7	int2	INT		33
8	diSize	DINT		4
9	abAryOut	ARRAY [1..5] OF BOOL		
10	abAryOut[1]	BOOL		TRUE
11	abAryOut[2]	BOOL		TRUE
12	abAryOut[3]	BOOL		TRUE
13	abAryOut[4]	BOOL		FALSE
14	abAryOut[5]	BOOL		FALSE
15	ret1	BOOL		TRUE

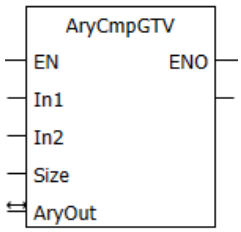
</td></tr>
</table>

- **Precautions**
  - Use the same data type for In1[ ] and In2[ ]; otherwise, a compilation fault will occur.
  - If In1[ ] and In2[ ] are real numbers that contain non-terminating decimal numbers, the processing results may be unexpected due to rounding errors.
  - When the value of Size is 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - When the value of Size exceeds the In1[ ] or AryOut[ ] array, the return value is FALSE and AryOut[ ] remains unchanged.
  - When the value of Size is less than 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.15.12 AryCmpGTV

Performs a greater-than comparison between the values of elements in an array and the value of a variable.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryCmpGTV	Array and value comparison (greater than)	FC		:= AryCmpGTV(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In1[ ] array	Comparison array	Follows data type	-	Array to compare
	In2	Comparison variable	Follows data type	-	Variable to compare
	Size	Number of elements to compare	Follows data type	1	Number of elements to compare
Input/ Output	AryOut[ ] array	Comparison result array	Follows data type	-	Comparison result array
Output	AryCmpGTV	Function return value	Follows data type	-	TRUE: Normal execution completed. FALSE: The instruction was not executed due to an error.

Variable types

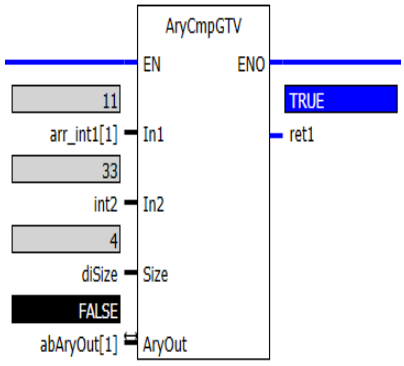
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1[ ] array	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2	Same data type as the elements of In1[ ]																			
Size	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
AryOut[ ] array	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

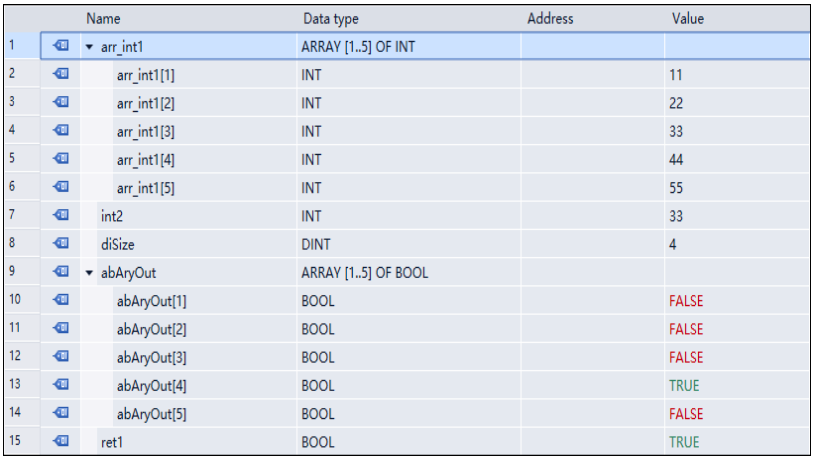
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
AryCmpGTV	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**
  - This instruction performs a greater-than comparison between the values of elements in the array (In1[0] to In1[Size-1]) and the value of In2. The comparison results are stored in elements with the corresponding indices in the array (AryOut[0] to AryOut[Size−1]).
  - When In1[i] > In2, AryOut[i] is TRUE; otherwise, it is FALSE.
- **Program example**

	LD	ST

Define variables	VAR
	ret1:BOOL;
	arr_int1:ARRAY[1..5] OF INT := [11,22,33,44,55];
	int2 :INT := 33;
	disize :DINT := 4;
	abArrout:Array[1..5] OF BOOL;
	END_VAR	

Program		ret1 TRUE := AryCmpGTV(In1 := arr_int1[1] 11 , In2 := int2 33 , Size := diSize 4 , AryOut := abAryOut[1] FALSE);RETURN

	LD	ST
Operation result		

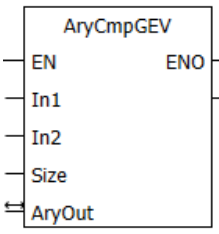
	Name	Data type	Address	Value
1	arr_int1	ARRAY [1..5] OF INT		
2	arr_int1[1]	INT		11
3	arr_int1[2]	INT		22
4	arr_int1[3]	INT		33
5	arr_int1[4]	INT		44
6	arr_int1[5]	INT		55
7	int2	INT		33
8	diSize	DINT		4
9	abAryOut	ARRAY [1..5] OF BOOL		
10	abAryOut[1]	BOOL		FALSE
11	abAryOut[2]	BOOL		FALSE
12	abAryOut[3]	BOOL		FALSE
13	abAryOut[4]	BOOL		TRUE
14	abAryOut[5]	BOOL		FALSE
15	ret1	BOOL		TRUE

- **Precautions**
  - Use the same data type for In1[ ] and In2[ ]; otherwise, a compilation fault will occur.
  - If In1[ ] and In2[ ] are real numbers that contain non-terminating decimal numbers, the processing results may be unexpected due to rounding errors.
  - When the value of Size is 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - When the value of Size exceeds the In1[ ] or AryOut[ ] array, the return value is FALSE and AryOut[ ] remains unchanged.
  - When the value of Size is less than 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.15.13 AryCmpGEV

Performs a greater-than or equal-to comparison between the values of elements in an array and the value of a variable.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryCmpGEV	Array and value comparison (greater than or equal to)	FC		:= AryCmpGEV(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In1[ ] array	Comparison array	Follows data type	-	Array to compare
	In2	Comparison variable	Follows data type	-	Variable to compare
	Size	Number of elements to compare	Follows data type	1	Number of elements to compare
Input/ Output	AryOut[ ] array	Comparison result array	Follows data type	-	Comparison result array
Output	AryCmpGEV	Function return value	Follows data type	-	TRUE: Normal execution completed. FALSE: The instruction was not executed due to an error.

Variable types

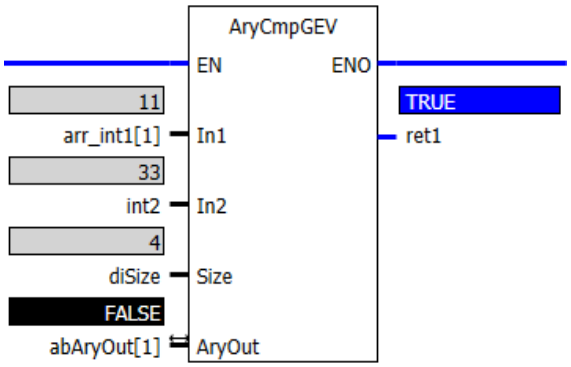
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1[ ] array	-	-	-	-	-	√	√	√	√	√	√	√	√	√	√	-	-	-	-	-
In2	Same data type as the elements of In1[ ]																			
Size	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
AryOut[ ] array	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
AryCmpGEV	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**
  - This instruction performs a greater-than or equal-to comparison between the values of elements in the array (In1[0] to In1[Size-1]) and the value of In2. The comparison results are stored in elements with the corresponding indices in the array (AryOut[0] to AryOut[Size−1]).
  - When In1[i] >= In2, AryOut[i] is TRUE; otherwise, it is FALSE.
- **Program example**

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    ret1:BOOL;
    arr_int1:ARRAY[1..5] OF INT := [11,22,33,44,55];
    int2 :INT := 33;
    disize :DINT := 4;
    abArrout:Array[1..5] OF BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
ret1 TRUE := AryCmpGEV(
        In1 := arr_int1[1] 11 ,
        In2 := int2 33 ,
        Size := diSize 4 ,
        AryOut := abAryOut[1] FALSE );RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

	Name	Data type	Address	Value
1	arr_int1	ARRAY [1..5] OF INT		
2	arr_int1[1]	INT		11
3	arr_int1[2]	INT		22
4	arr_int1[3]	INT		33
5	arr_int1[4]	INT		44
6	arr_int1[5]	INT		55
7	int2	INT		33
8	diSize	DINT		4
9	abAryOut	ARRAY [1..5] OF BOOL		
10	abAryOut[1]	BOOL		FALSE
11	abAryOut[2]	BOOL		FALSE
12	abAryOut[3]	BOOL		TRUE
13	abAryOut[4]	BOOL		TRUE
14	abAryOut[5]	BOOL		FALSE
15	ret1	BOOL		TRUE

</td></tr>
</table>

- **Precautions**
  - Use the same data type for In1[ ] and In2[ ]; otherwise, a compilation fault will occur.
  - If In1[ ] and In2[ ] are real numbers that contain non-terminating decimal numbers, the processing results may be unexpected due to rounding errors.
  - When the value of Size is 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - When the value of Size exceeds the In1[ ] or AryOut[ ] array, the return value is FALSE and AryOut[ ] remains unchanged.
  - When the value of Size is less than 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16 Any Array

18.2.16.1 Instruction List

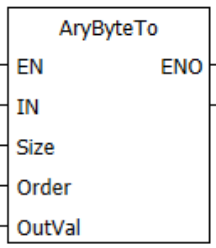
Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.16.2 AryByteTo	FC	Conversion from byte array	ArrayUtils	Library version 1.0.0.0 and later
18.2.16.3 AryExchange	FC	Array element exchange		Library version 1.0.0.0 and later
18.2.16.4 AryMove	FC	Array move		Library version 1.0.0.0 and later
18.2.16.5 AryToBCD	FC	Array-to-BCD conversion		Library version 1.0.0.0 and later
18.2.16.6 AryToBin	FC	Array-to-bit string conversion		Library version 1.0.0.0 and later
18.2.16.7 AryToString	FC	Array-to-string conversion		Library version 1.0.0.0 and later
18.2.16.8 SizeOfAry	FC	Get number of array elements		Library version 1.0.0.0 and later
18.2.16.9 StringToAry	FC	String-to-array conversion		Library version 1.0.0.0 and later
18.2.16.10 ToAryByte	FC	Conversion to byte array		Library version 1.0.0.0 and later
18.2.16.11 AryAnd	FC	Array logic AND		Library version 1.0.0.0 and later
18.2.16.12 AryOr	FC	Array logic OR		Library version 1.0.0.0 and later
18.2.16.13 AryXor	FC	Array logic XOR		Library version 1.0.0.0 and later
18.2.16.14 AryXorN	FC	Array logic XNOR		Library version 1.0.0.0 and later
18.2.16.15 AryAdd	FC	Array addition		Library version 1.0.0.0 and later
18.2.16.16 ArySub	FC	Array subtraction		Library version 1.0.0.0 and later
18.2.16.17 AryMul	FC	Array multiplication		Library version 1.0.0.0 and later
18.2.16.18 AryDiv	FC	Array division		Library version 1.0.0.0 and later
18.2.16.19 AryAddV	FC	Array and value addition		Library version 1.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.16.20 ArySubV	FC	Array and value subtraction		Library version 1.0.0.0 and later
18.2.16.21 AryMulV	FC	Array and value multiplication		Library version 1.0.0.0 and later
18.2.16.22 AryDivV	FC	Array and value division		Library version 1.0.0.0 and later
18.2.16.23 AryMean	FC	Array mean calculation		Library version 1.0.0.0 and later
18.2.16.24 arySD	FC	Array standard deviation calculation		Library version 1.0.0.0 and later
18.2.16.25 AryMax	FC	Maximum array value retrieval		Library version 1.0.0.0 and later
18.2.16.26 AryMin	FC	Minimum array value retrieval		Library version 1.0.0.0 and later
18.2.16.27 arySearch	FC	Array search		Library version 1.0.0.0 and later
18.2.16.28 arySHL	FC	Shift N-bits left with carry		Library version 1.0.0.0 and later
18.2.16.29 arySHR	FC	Shift N-bits right with carry		Library version 1.0.0.0 and later
18.2.16.30 aryShiftReg	FB	Shift register		Library version 1.0.0.0 and later
18.2.16.31 aryShiftRegLR	FB	Reversible shift register		Library version 1.0.0.0 and later

18.2.16.2 AryByteTo

Joins BYTE-type array elements and saves the result in a variable.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryByteTo	Conversion from byte array	FC		:= AryByteTo(IN := , Size := , Order := , OutVal :=);

- **Related variables**

  Variable

Input Variable	Variable	Name	Valid Range	Default Value	Description
Input	In[ ] array	Array to convert	Follows data type	-	Array to convert
	Size	Number of elements to convert	Follows data type	1	Number of elements in In[ ] to convert
	Order	Conversion order	Follows data type	-	Conversion order
	OutVal	Conversion result	Follows data type	-	Conversion result

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Order	See "Function description" for the enumerators for the enumerated type _eBYTE_ORDER.																			
OutVal	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration, an array, an array element, a structure, or a structure member can also be specified.																			

- **Function**
  - This instruction joins the first Size elements in In[ ] to match the size of the data type of the conversion result OutVal, and then saves the result to OutVal.
  - The order to join the elements of In[ ] is specified by Order. The data type of Order is enumerated type _eBYTE_ORDER. The meanings of the enumeration values are as follows.

Enumeration Value	Definition
Low-High	Lower byte first, higher byte last

Enumeration Value	Definition
_HIGH_LOW	Higher byte first, lower byte last

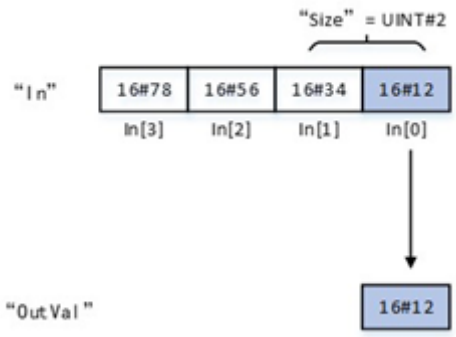
- The following describes the functions of AryByteTo in two situations.
  - If the data type of OutVal is one byte, In[ ] is stored in OutVal by one byte. The following data types have one byte.

Category	Data Type
Bit string	BYTE
Integer	USINT, SINT
Real number	/
Time, duration, date, and text string	/
Others	An enumeration with a size of one byte; an array for which the total for all elements is one byte; an array element that is one byte; a structure for which the total for all members is one byte; a structure member that is one byte

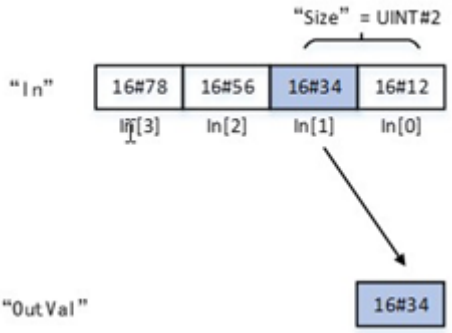
The storage method is as follows:

Value of Order	Data Type of OutVal	Storage Method
Low-High	BOOL	Bit 0 of In[0] is stored in OutVal.
	Non-BOOL	The value of In[0] is stored in OutVal.
_HIGH_LOW	BOOL	Bit 0 of In[1] is stored in OutVal.
	Non-BOOL	The value of In[1] is stored in OutVal.

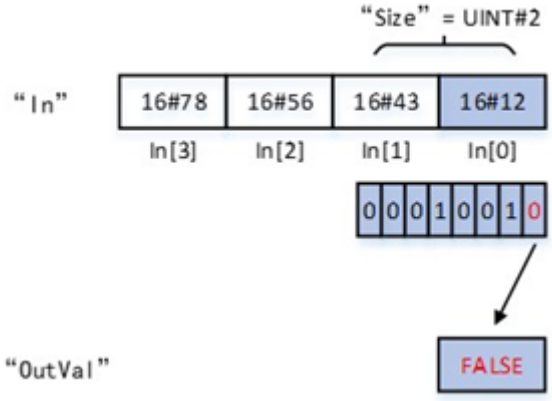
The following example sets OutVal to a SINT variable, Size to UINT#2, and Order to _LOW_HIGH, as shown below.



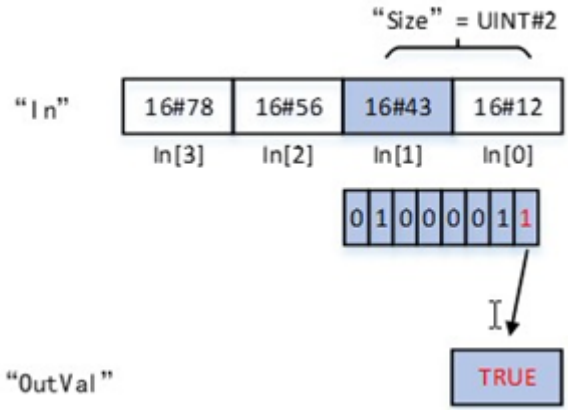
The following example sets OutVal to a SINT variable, Size to UINT#2, and Order to _HIGH_LOW, as shown below.



The following example sets OutVal to a BOOL variable, Size to UINT#2, and Order to _LOW_HIGH, as shown below.



The following example sets OutVal to a BOOL variable, Size to UINT#2, and Order to _HIGH_LOW, as shown below.



- If the data type of OutVal is two bytes or more, elements from In[ ] are joined to be equivalent to the data size of OutVal. The result is stored in OutVal. The following data types have two bytes or more.

Category	Data Type
Bit string	WORD, DWORD, LWORD
Integer	UINT, UDINT, ULINT, INT, DINT, LINT
Real number	REAL, LREAL
Time, duration, date, and text string	TIME, DATE, TOD, DT, and STRING types of two bytes or more

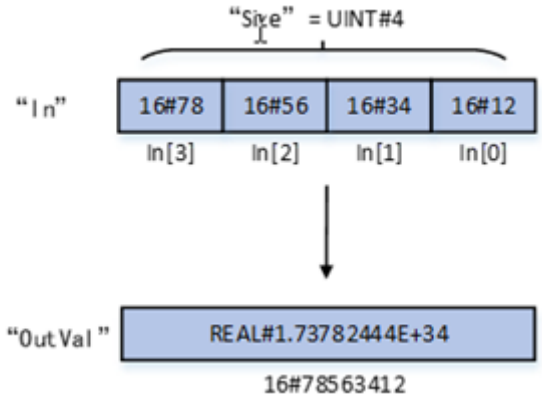
Category	Data Type
Others	An enumeration with two or more bytes; an array for which the total for all elements is two or more bytes; an array element that is two or more bytes; a structure for which the total for all members is two or more bytes; a structure member that is two or more bytes

The storage method is as follows:

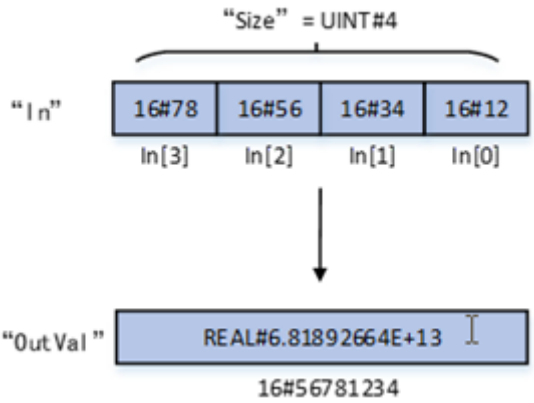
In[0] and In[1] are joined according to the value of Order to create one word (two bytes) of data. If Order is _LOW_HIGH, the higher byte is stored in In[1] and the lower byte is stored in In[0]. If Order is _HIGH_LOW, the higher byte is stored in In[0] and the lower byte is stored in In[1]. In the same way elements that start from In[2] and In[3] are joined to make more words of data. The words of data are joined to match the size of the data type of OutVal.

For example, if OutVal is DWORD data, four individual words of data are joined. If OutVal is an array, In[i] (i as an even number) is stored in OutVal[i+1]. In[i] (i as an odd number) is stored in OutVal[i-1]. If the value of "Size" is odd, data is stored till OutVal[Size], and 16#00 is stored in OutVal[Size-1]. The result data is stored in "OutVal".

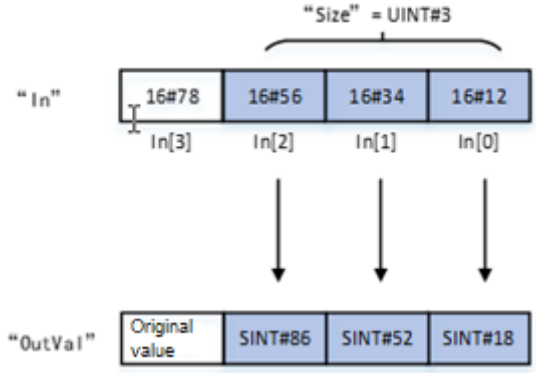
The following example sets OutVal to a REAL variable, Size to UINT#4, and Order to _LOW_HIGH, as shown below.



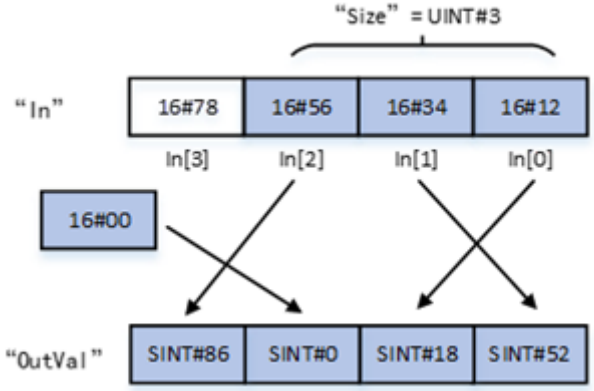
The following example sets OutVal to a REAL variable, Size to UINT#4, and Order to _HIGH_LOW, as shown below.



The following example sets OutVal to a SINT array with four elements, Size to UINT#3, and Order to _LOW_HIGH, as shown below.



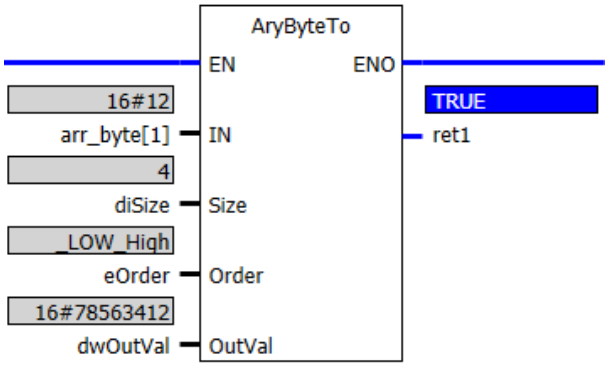
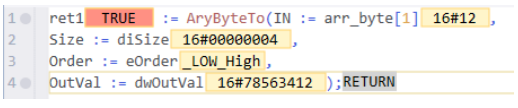
The following example sets OutVal to a SINT array with four elements, Size to UINT#3, and Order to _HIGH_LOW, as shown below.

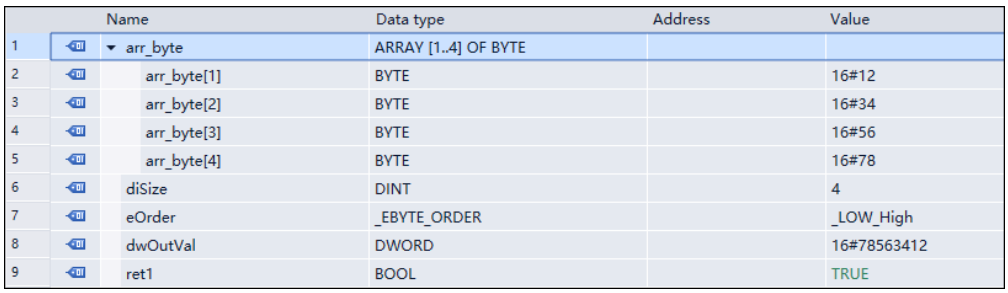


- **Program example**

	LD	ST

Define variables	VAR
	arr_byte: ARRAY[1..4] OF BYTE := [16#12,16#34,16#56,16#78];
	diSize: DINT := 4;
	eOrder: _eBYTE_ORDER := _EBYTE_ORDER._LOW_High;
	dwOutVal: DWORD;
	ret1: BOOL;
	END_VAR	

Program		

	LD	ST
Operation result		

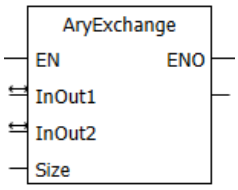
	Name	Data type	Address	Value
1	arr_byte	ARRAY [1..4] OF BYTE		
2	arr_byte[1]	BYTE		16#12
3	arr_byte[2]	BYTE		16#34
4	arr_byte[3]	BYTE		16#56
5	arr_byte[4]	BYTE		16#78
6	diSize	DINT		4
7	eOrder	_EBYTE_ORDER		_LOW_High
8	dwOutVal	DWORD		16#78563412
9	ret1	BOOL		TRUE

- **Precautions**
  - When Size exceeds the array range of In[ ] or when Order exceeds the valid range of the enumeration _eBYTE_ORDER, the function returns FALSE, and OutVal remains unchanged.
  - OutVal cannot be a BOOL array; otherwise, the program operation fails.
  - When OutVal is a structure, some of the values of In[ ] may be inserted in areas between members depending on the composition.
  - When the value of Size is 0, the value of Out is TRUE and OutVal remains unchanged.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.
  - In[ ] must be an indexed array; otherwise, a compilation fault will occur.

18.2.16.3 AryExchange

Exchanges elements between two arrays.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryExchange	Array element exchange	FC		:= AryExchange(InOut1 := , InOut2 := , Size :=);

- **Related variables**

Input variables

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Size	Number of elements to exchange	Follows data type	1	Number of elements to exchange
Input/Output	InOut1[ ] array	Array 1 to exchange	Follows data type	-	Array 1 to exchange

Input/Output	Variable	Name	Valid Range	Default Value	Description
	InOut2[ ] array	Array 2 to exchange	Follows data type	-	Array 2 to exchange

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
InOut1[ ] array	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Arrays of enumerations or structures can also be specified.																			
InOut2[ ] array	Same data type as the elements of InOut1[ ].																			

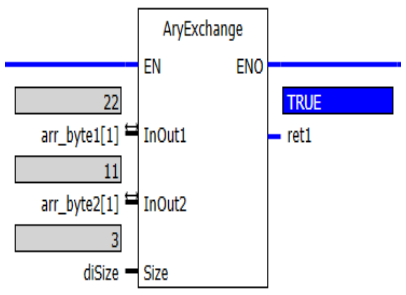
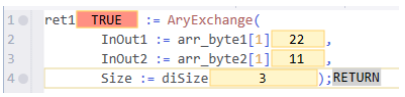
- **Function**

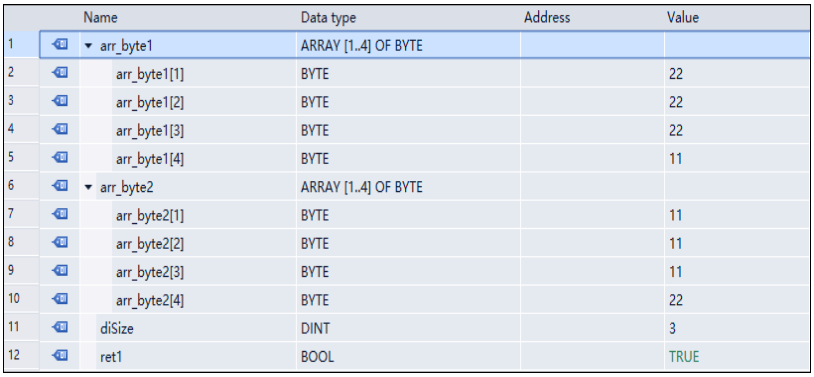
  This instruction exchanges the values of InOut1[ ] and InOut2[ ]. This instruction exchanges Size elements from the specified position between arrays InOut1[ ] and InOut2[ ].

- **Program example**

	LD	ST

Define variables	VAR
	arr_byte1: ARRAY[1..4] OF BYTE := [4(11)];
	arr_byte2: ARRAY[1..4] OF BYTE := [4(22)];
	diSize: DINT := 3;
	ret1: BOOL;
	END_VAR	

Program		

	LD	ST
Operation result		

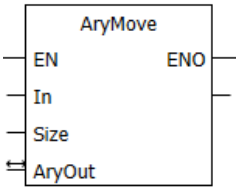
	Name	Data type	Address	Value
1	arr_byte1	ARRAY [1..4] OF BYTE		
2	arr_byte1[1]	BYTE		22
3	arr_byte1[2]	BYTE		22
4	arr_byte1[3]	BYTE		22
5	arr_byte1[4]	BYTE		11
6	arr_byte2	ARRAY [1..4] OF BYTE		
7	arr_byte2[1]	BYTE		11
8	arr_byte2[2]	BYTE		11
9	arr_byte2[3]	BYTE		11
10	arr_byte2[4]	BYTE		22
11	diSize	DINT		3
12	ret1	BOOL		TRUE

- **Precautions**
  - Use the same data type for InOut1[ ] and InOut2[ ]; otherwise, a compilation fault will occur.
  - When the value of Size is 0, the return value is TRUE, and InOut1[ ] and InOut2[ ] remain unchanged.
  - When both InOut1[ ] and InOut2[ ] are STRING-type arrays, and the length of a string element in one array exceeds that of the corresponding element in the other array, the return value is FALSE, and InOut1[ ] and InOut2[ ] remain unchanged.
  - When the value of Size exceeds the range of InOut1[ ] or InOut2[ ], the return value is FALSE, and InOut1[ ] and InOut2[ ] remain unchanged.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and InOut1[ ] and InOut2[ ] remain unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.4 AryMove

Moves one or more array elements.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryMove	Array move	FC		:= AryMove(In := , Size := , AryOut :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In[ ] array	Move source array	Follows data type	-	Array to move
	Size	Number of elements to move	Follows data type	1	Number of elements to move

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input/Output	AryOut[ ] array	Move result array	Follows data type	-	Move result array

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Arrays of enumerations or structures can also be specified.																			
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
AryOut[ ] array	Same data type as the elements of In[ ]																			

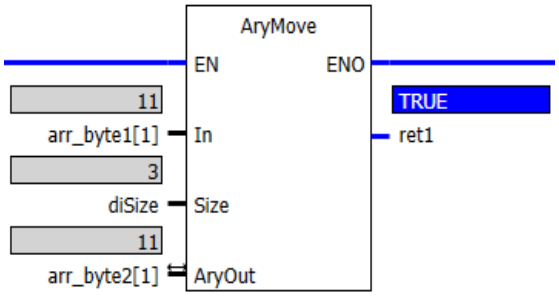
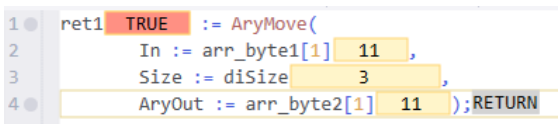
- **Function**

  This instruction moves Size elements of In[ ] starting from In[0] to AryOut[ ] starting from AryOut[0].

- **Program example**

	LD	ST

Define variables	VAR
	arr_byte1 : ARRAY[1..4] OF BYTE := [4(11)];
	arr_byte2 : ARRAY[1..4] OF BYTE := [4(22)];
	diSize : DINT := 3;
	ret1 : BOOL;
	END_VAR	

Program		

	LD	ST

Operation result

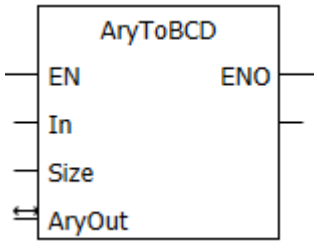
	Name	Data type	Address	Value
1	arr_byte1	ARRAY [1..4] OF BYTE		
2	arr_byte1[1]	BYTE		11
3	arr_byte1[2]	BYTE		11
4	arr_byte1[3]	BYTE		11
5	arr_byte1[4]	BYTE		11
6	arr_byte2	ARRAY [1..4] OF BYTE		
7	arr_byte2[1]	BYTE		11
8	arr_byte2[2]	BYTE		11
9	arr_byte2[3]	BYTE		11
10	arr_byte2[4]	BYTE		22
11	diSize	DINT		3
12	ret1	BOOL		TRUE

- **Precautions**
  - Use the same data type for In[ ] and AryOut[ ]; otherwise, a compilation fault will occur.
  - When the value of Size is 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - When both In[ ] and AryOut[ ] are STRING-type arrays and the length of a string in In[ ] exceeds the memory size of AryOut[ ], the return value is FALSE and AryOut[ ] remains unchanged.
  - When the value of Size exceeds the range of In[ ] or AryOut[ ], the return value is FALSE and AryOut[ ] remains unchanged.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.5 AryToBCD

Converts the elements of an unsigned integer array to BCD bit strings.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryToBCD	Array-to-BCD conversion	FC		:= AryToBCD(In := , Size := , AryOut :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In[ ] array	Unsigned integer array	Follows data type	-	Unsigned integer array

Input/Output	Variable	Name	Valid Range	Default Value	Description
	Size	Number of elements to convert	Follows data type	1	Number of elements in In[ ] to convert
Input/Output	AryOut[ ] array	BCD array	Follows data type	-	BCD array

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	–	–	–	–	–	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
AryOut[ ] array	Same data type as the elements of In[ ]																			

- **Function**

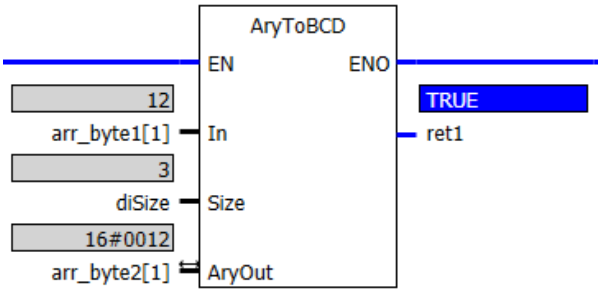
  This instruction converts Size elements of the unsigned integer array In[ ] starting from In[0] to a BCD bit string, and outputs the result to the BCD array AryOut[ ]. The valid ranges for In[ ] and AryOut[ ] vary depending on the data types of their elements, as shown below.

Data Type of In[ ] Element	Valid Range of In[ ]	Valid Range of AryOut[ ]
USINT	0 to 99	16#00 to 16#99 (BCD)
UINT	0 to 9999	16#0000 to 16#9999 (BCD)
UDINT	0 to 99999999	16#0000_0000 to 16#9999_9999 (BCD)
ULINT	0 to 9999999999999999	16#0000_0000_0000_0000 to 16#9999_9999_9999_9999 (BCD)

- **Program example**

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    arr_byte1: ARRAY[1..4] OF UINT := [12,34,567,89];
    arr_byte2: ARRAY[1..4] OF UINT;
    diSize: DINT := 3;
    ret1: BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>
</td><td>

```
ret1 TRUE := AryToBCD(
        In := arr_byte1[1] 12 ,
        Size := diSize 3 ,
        AryOut := arr_byte2[1] 16#0012 );RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

	Name	Data type	Address	Value
1	arr_byte1	ARRAY [1..4] OF UINT		
2	arr_byte1[1]	UINT		12
3	arr_byte1[2]	UINT		34
4	arr_byte1[3]	UINT		567
5	arr_byte1[4]	UINT		89
6	arr_byte2	ARRAY [1..4] OF UINT		
7	arr_byte2[1]	UINT		16#0012
8	arr_byte2[2]	UINT		16#0034
9	arr_byte2[3]	UINT		16#0567
10	arr_byte2[4]	UINT		16#0000
11	diSize	DINT		3
12	ret1	BOOL		TRUE

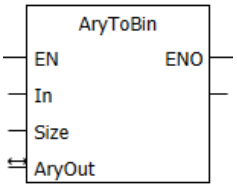
</td></tr>
</table>

- **Precautions**
  - This instruction does not support conversion to signed BCD. Therefore, the data type of In[ ] must be unsigned integer (USINT, UINT, UDINT, or ULINT).
  - When the value of Size is 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - When the value of In[ ] exceeds the valid range, the return value is FALSE and AryOut[ ] remains unchanged.
  - When the value of Size exceeds the range of In[ ] or AryOut[ ], the return value is FALSE and AryOut[ ] remains unchanged.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.6 AryToBin

Converts the elements of a BCD array into bit strings.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryToBin	Array-to-bit string conversion	FC		:= AryToBin(In := , Size := , AryOut :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In[ ] array	BCD array	Follows data type	-	BCD array
	Size	Number of elements to convert	Follows data type	1	Number of elements in In[ ] to convert
Input/Output	AryOut[ ] array	Bit string array	Follows data type	-	Bit string after conversion

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	–	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
AryOut[ ] array	Same data type as the elements of In[ ]																			

- **Function**

  This instruction converts Size elements of the BCD array In[ ] starting from In[0] to bit strings, and outputs the result to the BCD array AryOut[ ]. The valid ranges for In[ ] and AryOut[ ] vary depending on the data types of their elements, as shown below.

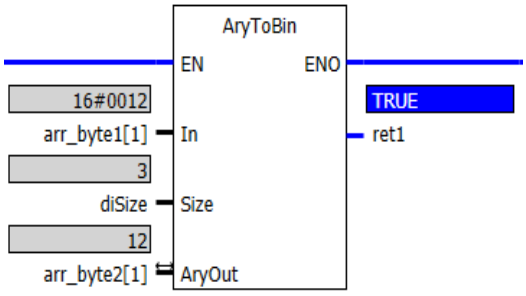
Data Type of In[ ] Element	Valid Range of In[ ]	Valid Range of AryOut[ ]
BYTE	16#00 to 16#99 (BCD)	0 to 99
WORD	16#0000 to 16#9999 (BCD)	0 to 9999

Data Type of In[ ] Element	Valid Range of In[ ]	Valid Range of AryOut[ ]
DWORD	16#0000_0000 to 16#9999_9999 (BCD)	0 to 99999999
LWORD	16#0000_0000_0000_0000 to 16#9999_9999_9999_9999 (BCD)	0 to 9999999999999999

- **Program example**

	LD	ST

Define variables	VAR
	arr_byte1 : ARRAY[1..4] OF WORD := [16#12,16#34,16#567,16#89];
	arr_byte2 : ARRAY[1..4] OF WORD;
	diSize : DINT := 3;
	ret1 : BOOL;
	END_VAR	

Program		ret1 TRUE := AryToBin(In := arr_byte1[1] 16#0012 , Size := diSize 3 , AryOut := arr_byte2[1] 12);RETURN
Operation result		

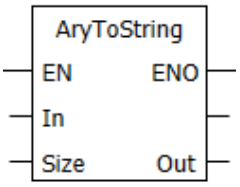
	Name	Data type	Address	Value
1	arr_byte1	ARRAY [1..4] OF WORD		
2	arr_byte1[1]	WORD		16#0012
3	arr_byte1[2]	WORD		16#0034
4	arr_byte1[3]	WORD		16#0567
5	arr_byte1[4]	WORD		16#0089
6	arr_byte2	ARRAY [1..4] OF WORD		
7	arr_byte2[1]	WORD		12
8	arr_byte2[2]	WORD		34
9	arr_byte2[3]	WORD		567
10	arr_byte2[4]	WORD		0
11	diSize	DINT		3
12	ret1	BOOL		TRUE

- **Precautions**
  - This instruction does not support conversion to signed BCD. Therefore, use a bit string (BYTE, WORD, DWORD, or LWORD) as the data type of AryOut[ ].
  - When the value of Size is 0, the return value is TRUE and AryOut[ ] remains unchanged.
  - When the value of In[ ] exceeds the valid range, the return value is FALSE and AryOut[ ] remains unchanged.
  - When the value of Size exceeds the range of In[ ] or AryOut[ ], the return value is FALSE and AryOut[ ] remains unchanged.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.7 AryToString

Converts a BYTE-type array to a string.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
AryToString	Array-to-string conversion	FC		:= AryToString(In := , Size := , Out =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In[ ] array	Byte array	Follows data type	-	Byte array
	Size	Number of elements to convert	Follows data type	1	Number of elements in In[ ] to convert
Output	Out	String	Follows data type	-	String

Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

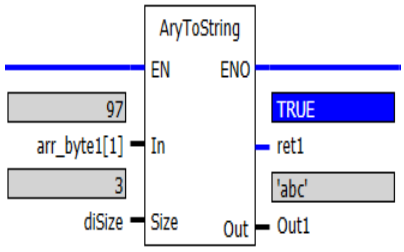
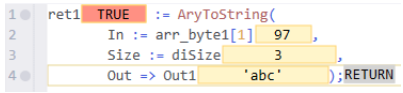
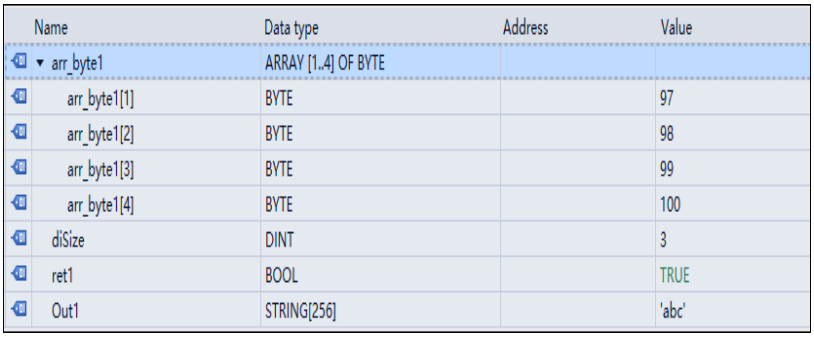
- **Function description**
  - This instruction processes the elements in the BYTE-type array In[ ] starting from In[0] as character codes, converts them into a string, and stores the result in Out. NULL is added to the end of Out.

- Size specifies the number of elements in In[ ] to convert.
- If there is a NULL character between In[0] and In[Size–1], only the content before NULL is stored in Out.

- **Program example**

	LD	ST

Define variables	VAR
	arr_byte1 : ARRAY[1..4] OF BYTE := [97,98,99,100];
	diSize : DINT := 3;
	ret1 : BOOL;
	Out1 : STRING[256];
	END_VAR	

Program		
Operation result		

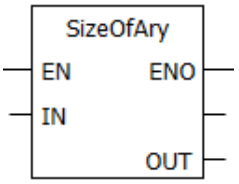
Name	Data type	Address	Value
arr_byte1	ARRAY [1..4] OF BYTE		
arr_byte1[1]	BYTE		97
arr_byte1[2]	BYTE		98
arr_byte1[3]	BYTE		99
arr_byte1[4]	BYTE		100
diSize	DINT		3
ret1	BOOL		TRUE
Out1	STRING[256]		'abc'

- **Precautions**
  - When Size exceeds the array range of In[ ] or exceeds the maximum string length of 1985, the function returns FALSE, and Out remains unchanged.
  - In[ ] must be of BYTE type; otherwise, a compilation fault will occur.
  - When the value of Size is 0, Out is a string containing only NULL characters.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and "Out" remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.
  - If the size of the input elements exceeds the ASCII index range (0 to 127), the output result will be abnormal.

18.2.16.8 SizeOfAry

Gets the number of elements in an array.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SizeOfAry	Get number of array elements	FC		:= SizeOfAry(IN := , OUT =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In[ ] array	Array	Follows data type	-	Array
Output	Out	Number of elements	Follows data type	-	Number of elements

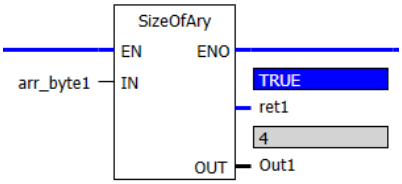
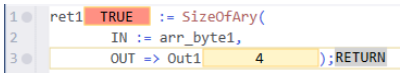
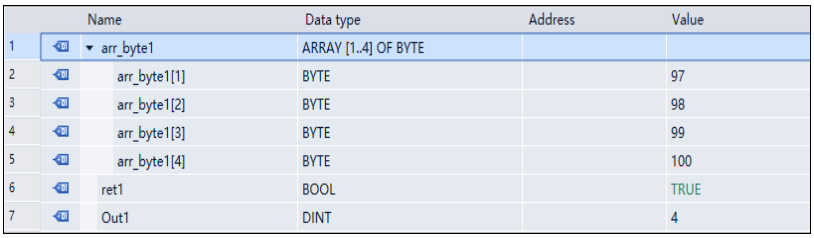
Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Arrays of enumerations or structures can also be specified.																			
Out	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **Function**

  This instruction gets the number of elements in array In[ ]. The input parameter can be an array or an array element.

- **Program example**

	LD	ST
Define variables	VAR arr_byte1 : ARRAY[1..4] OF BYTE := [97,98,99,100]; ret1 : BOOL; Out1 : DINT; END_VAR	
Program		
Operation result		

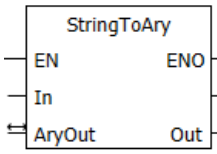
	Name	Data type	Address	Value
1	arr_byte1	ARRAY [1..4] OF BYTE		
2	arr_byte1[1]	BYTE		97
3	arr_byte1[2]	BYTE		98
4	arr_byte1[3]	BYTE		99
5	arr_byte1[4]	BYTE		100
6	ret1	BOOL		TRUE
7	Out1	DINT		4

- **Precautions**
  - If the dimension of the input array exceeds 3, the return value is FALSE, and Out remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.9 StringToAry

Converts a string to a BYTE-type array.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StringToAry	String-to-array conversion	FC		:= StringToAry(In := , AryOut := , Out =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	String	Max. 1985 characters	-	String
Input/Output	AryOut[ ] array	Byte array	Follows data type	-	Byte array

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	Out	Number of bytes after conversion	0 to 1985	-	Number of bytes after conversion

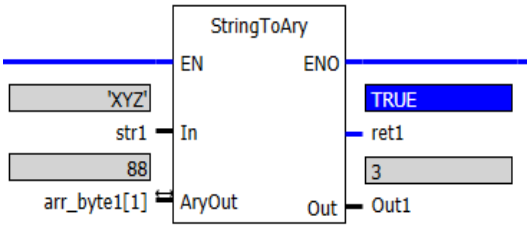
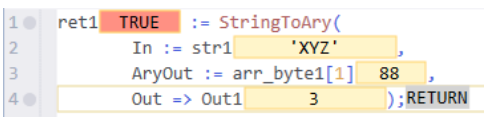
Variable types

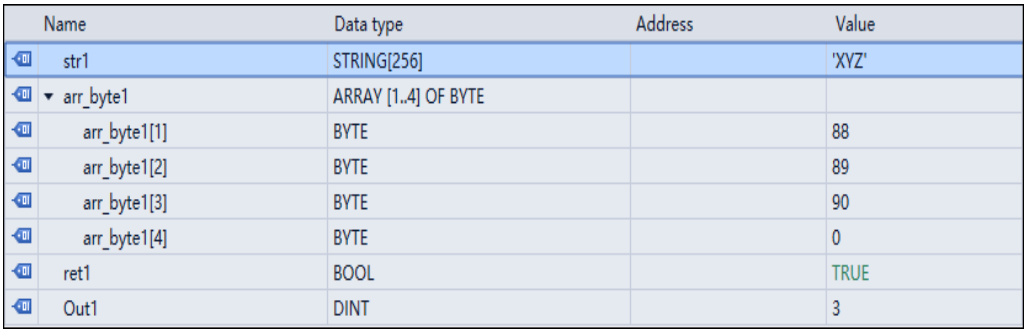
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
AryOut[ ] array	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **Function**

  This instruction processes the character codes in text string In as numeric values and stores each value to the corresponding element of the byte array AryOut[ ]. The number of bytes after conversion is stored in Out.

- **Program example**

	LD	ST
Define variables	VAR str1 : STRING[256] := 'XYZ'; arr_byte1 : ARRAY[1..4] OF BYTE; ret1 : BOOL; Out1 : DINT; END_VAR	
Program		

	LD	ST
Operation result		

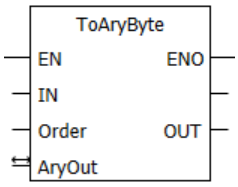
Name	Data type	Address	Value
str1	STRING[256]		'XYZ'
arr_byte1	ARRAY [1..4] OF BYTE		
arr_byte1[1]	BYTE		88
arr_byte1[2]	BYTE		89
arr_byte1[3]	BYTE		90
arr_byte1[4]	BYTE		0
ret1	BOOL		TRUE
Out1	DINT		3

- **Precautions**
  - The NULL character at the end of In is not stored in AryOut[ ].
  - When In contains only NULL characters, the value of Out is 0 and AryOut[ ] remains unchanged.
  - When the number of bytes in In exceeds the number of elements in AryOut[ ], the return value is FALSE and AryOut[ ] remains unchanged.
  - When the number of bytes in In exceeds the upper limit 1985, excessive characters are truncated and only 1985 characters are converted.
  - If the dimension of the output array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.10 ToAryByte

Separates a variable into bytes and stores the result in a BYTE-type array.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ToAryByte	Conversion to byte array	FC		:= ToAryByte(IN := , Order := , AryOut := , OUT =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In	Conversion object	Follows data type	-	Conversion object
	Order	Conversion order	Follows data type	_LOW_HIGH	Conversion order

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input/Output	AryOut[ ] array	Conversion result array	Follows data type	-	An array indicating the conversion result
Output	Out	Number of elements in the conversion result	Follows data type	-	Number of elements in the conversion result

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration, an array, an array element, a structure, or a structure member can also be specified.																			
Order	See "Function description" for the enumerators for the enumerated type _eBYTE_ORDER.																			
AryOut[ ] array	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **Function**

  This instruction separates the input variable In into bytes and stores them in order in AryOut[ ] starting from AryOut[0]. The number of elements in AryOut[ ] is stored in Out. The conversion order Order specifies the order in which to convert the value of In to bytes. The data type of Order is enumerated type _eBYTE_ORDER. The meanings of the enumerators are as follows.

Enumerator	Definition
_LOW_HIGH	Lower byte first, higher byte last
_HIGH_LOW	Higher byte first, lower byte last

In is separated into bytes and stored in AryOut[ ]. The following data types have two bytes or more.

Category	Data Type
Bit string	WORD, DWORD, and LWORD

Category	Data Type
Integer	UINT, UDINT, ULINT, INT, DINT, LINT
Real number	REAL, LREAL
Time, duration, date, and text string	TIME, DATE, TOD, DT, and STRING types of two bytes or more
Others	An enumeration An array for which the total for all elements is two bytes or more An array element of two bytes or more A structure for which the total for all members is two bytes or more A structure member of two bytes or more

Processing steps:

1. Split the value of In by word (two bytes).
2. The lowest word is separated into bytes.
3. If the value of Order is _LOW_HIGH, save the lower byte into AryOut[0] and the higher byte into AryOut[1]; if the value of Order is _HIGH_LOW, save the higher byte into AryOut[0] and the lower byte into AryOut[1].
4. The next word is separated into bytes and stored in AryOut[2] and AryOut[3] in the same way.
5. This process is repeated to the end of the value of In, and the result is stored in the corresponding position. If In is an array, the same process is repeated to the last element of In.

When the data type of In is one byte, In is stored in AryOut[ ] as one byte. The following data types have one byte.

Category	Data Type
Bit string	BYTE
Integer	USINT, SINT
Real number	/
Time, duration, date, and text string	String of one byte
Others	An array for which the total for all elements is one byte; An array element that is one byte; A structure for which the total for all members is one byte; A structure member that is one byte

The storage method is as follows:

Value of Order	In (Array or Not)	Storage Method
_LOW_HIGH	Not an integer	The value of In is stored in AryOut[0].
	Integer	The value of In[i] is stored in AryOut[i].
_HIGH_LOW	Not an array	The value of In is stored in AryOut[1]. 16#00 is stored in AryOut[0].
	Array	In[i] (where i is even) is stored in AryOut[i+1]. In[i] (where i is odd) is stored in AryOut[i-1]. If the number of elements in In[ ] is odd, 16#00 is stored in the last element of AryOut[n–1].

Data is stored in AryOut[ ] as described below.

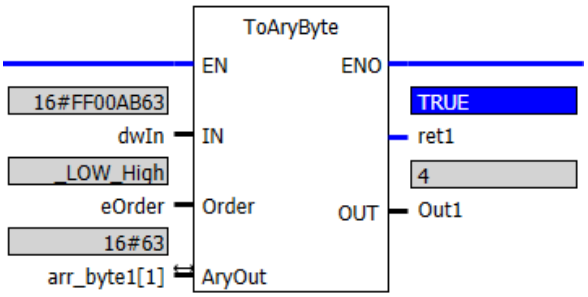
Value of Order	In (Array or Not)	Storage Method
_LOW_HIGH	Not an array	The logical AND of the value of In and 16#00 is stored in AryOut[0].
	Array	The value of In[0] to In[7] is stored in AryOut[0]. The value of In[8] to In[15] is stored in AryOut[1]. The same process is repeated to store the rest of the data. When In[ ] contains fewer than eight elements, add FALSE to the higher bits. The value of Out is always even. Any insufficient bits are filled with FALSE.
_HIGH_LOW	Not an array	The logical AND of the value of In and 16#00 is stored in AryOut[1]. 16#00 is stored in AryOut[0].
	Array	The value of In[0] to In[7] is stored in AryOut[1]. The value of In[8] to In[15] is stored in AryOut[0]. The same process is repeated to store the rest of the data. The value of Out is always even. Any insufficient bits are filled with FALSE.

- **Program example**

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    dwIn: DWORD := 16#FF00AB63;
    eOrder: _eBYTE_ORDER := _EBYTE_ORDER._LOW_High;
    arr_byte1: ARRAY[1..4] OF BYTE;
    ret1: BOOL;
    Out1: DINT;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
ret1 := ToAryByte(
        IN := dwIn ,
        Order := eOrder ,
        AryOut := arr_byte1[1] ,
        OUT => Out1 );RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

	Name	Data type	Address	Value
1	dwIn	DWORD		16#FF00AB63
2	eOrder	_EBYTE_ORDER		_LOW_High
3	arr_byte1	ARRAY [1..4] OF BYTE		
4	arr_byte1[1]	BYTE		16#63
5	arr_byte1[2]	BYTE		16#AB
6	arr_byte1[3]	BYTE		16#00
7	arr_byte1[4]	BYTE		16#FF
8	ret1	BOOL		TRUE
9	Out1	DINT		4

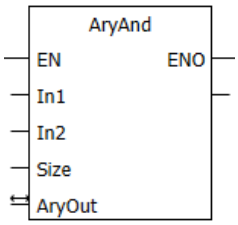
</td></tr>
</table>

- **Precautions**
  - The input parameter In must be a variable. If it is a constant, a compilation fault will occur.
  - When In is an enumeration, you cannot directly pass an enumerator. Otherwise, a compilation fault will occur.
  - When In is of STRING type, the ASCII value of each character is converted into a byte array.
  - When In is a structure, adjustment areas between members may be inserted into AryOut[ ] depending on the composition.
  - When the value of Order is _HIGH_LOW and the total number of bytes in In is an odd number, 16#00 is added to the end of In to make an even number of bytes before the conversion is started.
  - When the value of Order exceeds the valid range, the return value is FALSE, and Out and AryOut[ ] remain unchanged.
  - When the conversion result exceeds the AryOut[ ] array, the return value is FALSE, and Out and AryOut[ ] remain unchanged.
  - If the dimension of the output array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.11 AryAnd

Performs a bit-wise logical AND operation on each Boolean or bit-string element in two arrays.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryAnd	Array logic AND	FC		:= AryAnd(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Array In1[ ]	Array 1 to process	Follows data type	-	Array 1 to process
	Array In2[ ]	Array 2 to process	Follows data type	-	Array 2 to process
	Size	Number of elements to process	Follows data type	1	Number of elements to process
Input/Output	AryOut[ ] array	Processing result array	Follows data type	-	Processing result array

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Array In1[ ]	√	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Array In2[ ]	Same data type as In1[ ]																			
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
AryOut[ ] array	Same data type as In1[ ]																			

- **Function**
  - This instruction performs a bit-wise logical AND operation on Size elements in arrays In1[ ] and In2[ ]. The operation result is stored in the corresponding element of AryOut[ ]. Therefore, In1[ ], In2[ ], and AryOut[ ] must be of the same data type. The relationships between input and output variables are described in the following tables.
  - When both bits are TRUE, the processing result is TRUE. Otherwise, the processing result is FALSE.

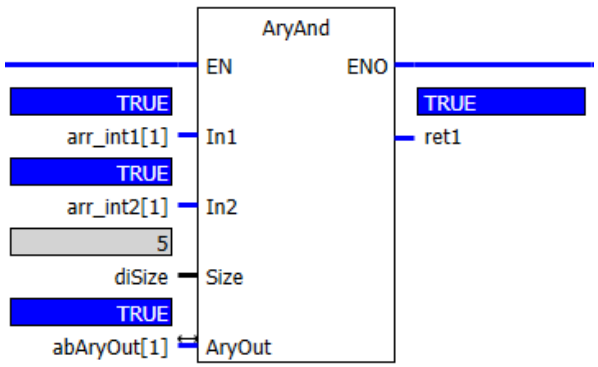
Bit of Element in In1[ ]	Bit of Element in In2[ ]	Bit of AryOut[ ]
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

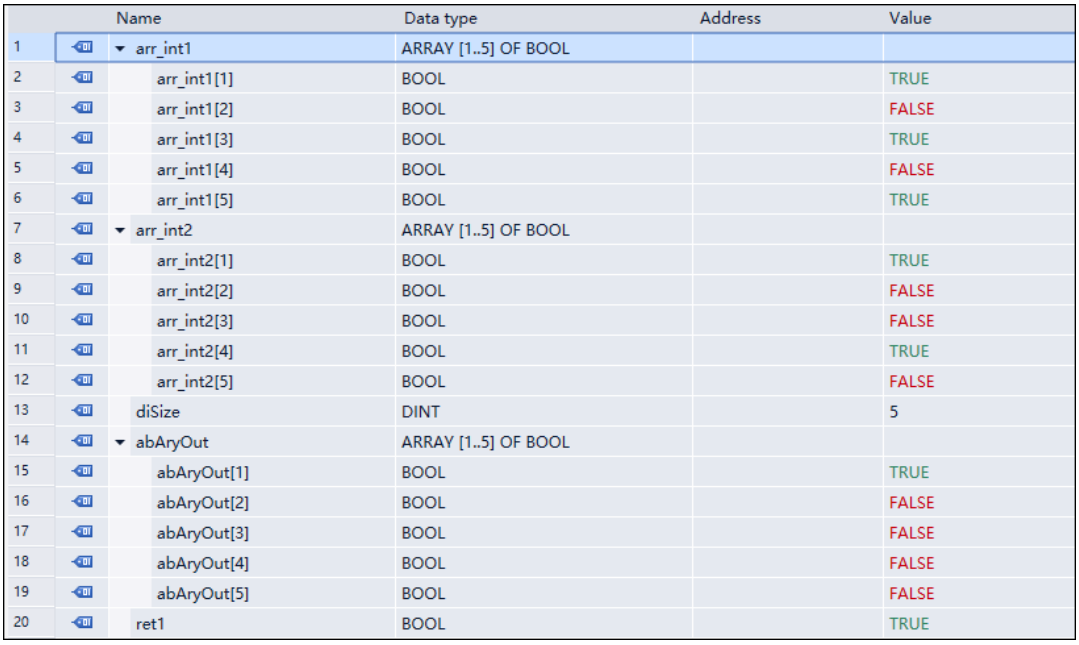
- **Program example**

  The following example sets Size to DINT#5:

	LD	ST

Define variables	VAR
	arr_int1: ARRAY[1..5] OF BOOL := [TRUE,FALSE,TRUE,FALSE,TRUE];
	arr_int2: ARRAY[1..5] OF BOOL := [TRUE,FALSE,FALSE,TRUE,FALSE];
	diSize: DINT := 5;
	abAryOut: ARRAY[1..5] OF BOOL;
	ret1: BOOL;
	END_VAR	

Program		ret1 TRUE := AryAnd(In1 := arr_int1[1] TRUE , In2 := arr_int2[1] TRUE , Size := diSize 5 , AryOut := abAryOut[1] TRUE);RETURN

	LD	ST
Operation result		

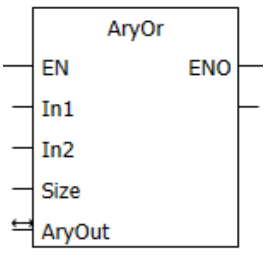
	Name	Data type	Address	Value
1	arr_int1	ARRAY [1..5] OF BOOL		
2	arr_int1[1]	BOOL		TRUE
3	arr_int1[2]	BOOL		FALSE
4	arr_int1[3]	BOOL		TRUE
5	arr_int1[4]	BOOL		FALSE
6	arr_int1[5]	BOOL		TRUE
7	arr_int2	ARRAY [1..5] OF BOOL		
8	arr_int2[1]	BOOL		TRUE
9	arr_int2[2]	BOOL		FALSE
10	arr_int2[3]	BOOL		FALSE
11	arr_int2[4]	BOOL		TRUE
12	arr_int2[5]	BOOL		FALSE
13	diSize	DINT		5
14	abAryOut	ARRAY [1..5] OF BOOL		
15	abAryOut[1]	BOOL		TRUE
16	abAryOut[2]	BOOL		FALSE
17	abAryOut[3]	BOOL		FALSE
18	abAryOut[4]	BOOL		FALSE
19	abAryOut[5]	BOOL		FALSE
20	ret1	BOOL		TRUE

- **Precautions**
  - In1[ ], In2[ ], and AryOut[ ] must be of the same data type. Otherwise, a compilation fault will occur.
  - When the value of Size exceeds the In1[ ], In2[ ], or AryOut[ ] array, the return value is FALSE and AryOut[ ] remains unchanged.
  - When the value of Size is 0, AryOut[ ] remains unchanged and the return value is TRUE.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.12 AryOr

Performs a bit-wise logical OR operation on each Boolean or bit-string element in two arrays.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryOr	Array logic OR	FC		:= AryOr(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Array In1[ ]	Array 1 to process	Follows data type	-	Array 1 to process
	Array In2[ ]	Array 2 to process	Follows data type	-	Array 2 to process
	Size	Number of elements to process	Follows data type	1	Number of elements to process
Input/Output	AryOut[ ] array	Processing result array	Follows data type	-	Processing result array

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Array In1[ ]	√	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Array In2[ ]	Same data type as In1[ ]																			
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
AryOut[ ] array	Same data type as In1[ ]																			

- **Function**
  - This instruction performs a bit-wise logical XNOR operation on Size elements in arrays In1[ ] and In2[ ]. The operation result is stored in the corresponding element of AryOut[ ]. Therefore, In1[ ], In2[ ], and AryOut[ ] must be of the same data type. The relationships between input and output variables are described in the following tables.
  - When both bits are FALSE, the processing result is FALSE. Otherwise, the processing result is TRUE.

Bit of Element in In1[ ]	Bit of Element in In2[ ]	Bit of AryOut[ ]
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE

Bit of Element in In1[ ]	Bit of Element in In2[ ]	Bit of AryOut[ ]
TRUE	TRUE	TRUE

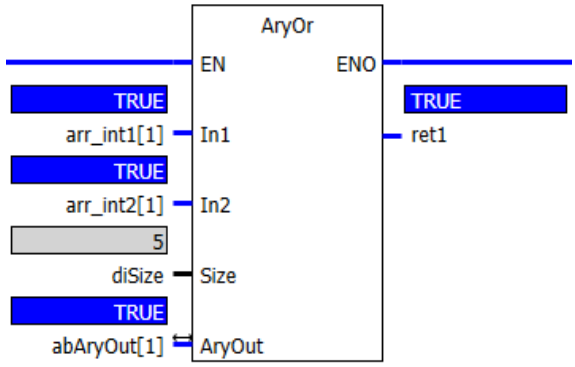
- **Program example**

  The following example sets Size to DINT#4:

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    arr_int1: ARRAY[1..5] OF BOOL := [TRUE,FALSE,TRUE,FALSE,TRUE];
    arr_int2: ARRAY[1..5] OF BOOL := [TRUE,FALSE,FALSE,TRUE,FALSE];
    diSize: DINT := 5;
    abAryOut: ARRAY[1..5] OF BOOL;
    ret1: BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
ret1 TRUE := AryOr(
        In1 := arr_int1[1] TRUE ,
        In2 := arr_int2[1] TRUE ,
        Size := diSize 5 ,
        AryOut := abAryOut[1] TRUE );RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

	Name	Data type	Address	Value
1	arr_int1	ARRAY [1..5] OF BOOL		
2	arr_int1[1]	BOOL		TRUE
3	arr_int1[2]	BOOL		FALSE
4	arr_int1[3]	BOOL		TRUE
5	arr_int1[4]	BOOL		FALSE
6	arr_int1[5]	BOOL		TRUE
7	arr_int2	ARRAY [1..5] OF BOOL		
8	arr_int2[1]	BOOL		TRUE
9	arr_int2[2]	BOOL		FALSE
10	arr_int2[3]	BOOL		FALSE
11	arr_int2[4]	BOOL		TRUE
12	arr_int2[5]	BOOL		FALSE
13	diSize	DINT		5
14	abAryOut	ARRAY [1..5] OF BOOL		
15	abAryOut[1]	BOOL		TRUE
16	abAryOut[2]	BOOL		FALSE
17	abAryOut[3]	BOOL		TRUE
18	abAryOut[4]	BOOL		TRUE
19	abAryOut[5]	BOOL		TRUE
20	ret1	BOOL		TRUE

</td></tr>
</table>

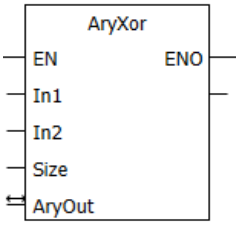
- **Precautions**
  - In1[ ], In2[ ], and AryOut[ ] must be of the same data type. Otherwise, a compilation fault will occur.
  - When the value of Size exceeds the In1[ ], In2[ ], or AryOut[ ] array, the return value is FALSE and AryOut[ ] remains unchanged.
  - When the value of Size is 0, AryOut[ ] remains unchanged and the return value is TRUE.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.

◦ Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.13 AryXor

Performs a bit-wise logical XOR operation on each Boolean or bit-string element in two arrays.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryXor	Array logic XOR	FC		:= AryXor(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Array In1[ ]	Array 1 to process	Follows data type	-	Array 1 to process
	Array In2[ ]	Array 2 to process	Follows data type	-	Array 2 to process
	Size	Number of elements to process	Follows data type	1	Number of elements to process
Input/Output	AryOut[ ] array	Processing result array	Follows data type	-	Processing result array

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Array In1[ ]	√	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Array In2[ ]	Same data type as In1[ ]																			
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
AryOut[ ] array	Same data type as In1[ ]																			

- **Function**
  - This instruction performs a bit-wise logical XNOR operation on Size elements in arrays In1[ ] and In2[ ]. The operation result is stored in the corresponding element of AryOut[ ]. Therefore, In1[ ], In2[ ], and AryOut[ ] must be of the same data type. The relationships between input and output variables are described in the following tables.
  - When both bits are the same, the processing result is FALSE. The operation result is TRUE when they are different.

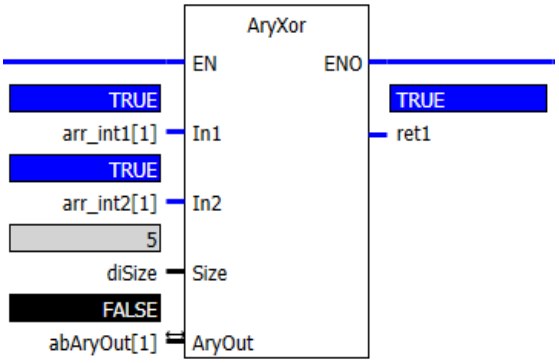
Bit of Element in In1[ ]	Bit of Element in In2[ ]	Bit of AryOut[ ]
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	FALSE

- **Program example**

  The following example sets Size to DINT#5:

	LD	ST

Define variables	VAR
	arr_int1: ARRAY[1..5] OF BOOL := [TRUE,FALSE,TRUE,FALSE,TRUE];
	arr_int2: ARRAY[1..5] OF BOOL := [TRUE,FALSE,FALSE,TRUE,FALSE];
	diSize: DINT := 5;
	abAryOut: ARRAY[1..5] OF BOOL;
	ret1: BOOL;
	END_VAR	

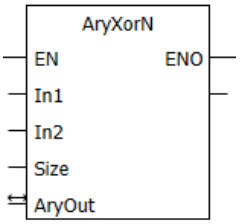
<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>
</td><td>ret1 TRUE := AryXor(
In1 := arr_int1[1] TRUE ,
In2 := arr_int2[1] TRUE ,
Size := diSize 5 ,
AryOut := abAryOut[1] FALSE);RETURN</td></tr>
<tr><td>Operation result</td><td colspan="2">
<table>
<tr><th></th><th></th><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td>1</td><td></td><td>arr_int1</td><td>ARRAY [1..5] OF BOOL</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td>arr_int1[1]</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>3</td><td></td><td>arr_int1[2]</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>4</td><td></td><td>arr_int1[3]</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>5</td><td></td><td>arr_int1[4]</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>6</td><td></td><td>arr_int1[5]</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>7</td><td></td><td>arr_int2</td><td>ARRAY [1..5] OF BOOL</td><td></td><td></td></tr>
<tr><td>8</td><td></td><td>arr_int2[1]</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>9</td><td></td><td>arr_int2[2]</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>10</td><td></td><td>arr_int2[3]</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>11</td><td></td><td>arr_int2[4]</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>12</td><td></td><td>arr_int2[5]</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>13</td><td></td><td>diSize</td><td>DINT</td><td></td><td>5</td></tr>
<tr><td>14</td><td></td><td>abAryOut</td><td>ARRAY [1..5] OF BOOL</td><td></td><td></td></tr>
<tr><td>15</td><td></td><td>abAryOut[1]</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>16</td><td></td><td>abAryOut[2]</td><td>BOOL</td><td></td><td>FALSE</td></tr>
<tr><td>17</td><td></td><td>abAryOut[3]</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>18</td><td></td><td>abAryOut[4]</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>19</td><td></td><td>abAryOut[5]</td><td>BOOL</td><td></td><td>TRUE</td></tr>
<tr><td>20</td><td></td><td>ret1</td><td>BOOL</td><td></td><td>TRUE</td></tr>
</table>
</td></tr>
</table>

- **Precautions**
  - In1[ ], In2[ ], and AryOut[ ] must be of the same data type. Otherwise, a compilation fault will occur.
  - When the value of Size exceeds the In1[ ], In2[ ], or AryOut[ ] array, the return value is FALSE and AryOut[ ] remains unchanged.
  - When the value of Size is 0, AryOut[ ] remains unchanged and the return value is TRUE.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.14 AryXorN

Performs a bit-wise logical XNOR operation on each Boolean or bit-string element in two arrays.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryXorN	Array logic XNOR	FC		:= AryXorN(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Array In1[ ]	Array 1 to process	Follows data type	-	Array 1 to process
	Array In2[ ]	Array 2 to process	Follows data type	-	Array 2 to process
	Size	Number of elements to process	Follows data type	1	Number of elements to process
Input/Output	AryOut[ ] array	Processing result array	Follows data type	-	Processing result array

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Array In1[ ]	√	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Array In2[ ]	Same data type as In1[ ]																			
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
AryOut[ ] array	Same data type as In1[ ]																			

- **Function**
  - This instruction performs a bit-wise logical XNOR operation on Size elements in arrays In1[ ] and In2[ ]. The operation result is stored in the corresponding element of AryOut[ ]. Therefore, In1[ ], In2[ ], and

AryOut[ ] must be of the same data type. The relationships between input and output variables are described in the following tables.

- When both bits are the same, the processing result is TRUE. The operation result is FALSE when they are different.

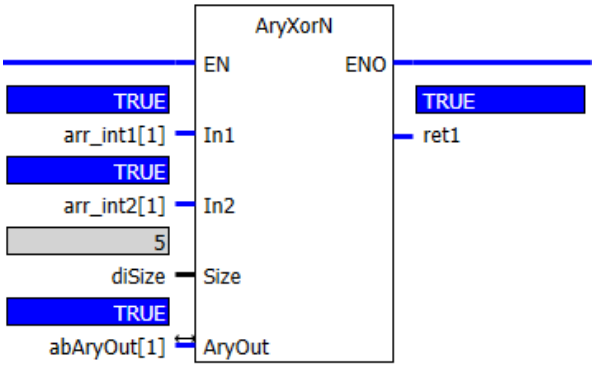
Bit of Element in In1[ ]	Bit of Element in In2[ ]	Bit of AryOut[ ]
FALSE	FALSE	TRUE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

- **Program example**

The following example sets Size to DINT#5:

	LD	ST

Define variables	VAR
	arr_int1: ARRAY[1..5] OF BOOL := [TRUE,FALSE,TRUE,FALSE,TRUE];
	arr_int2: ARRAY[1..5] OF BOOL := [TRUE,FALSE,FALSE,TRUE,FALSE];
	diSize: DINT := 5;
	abAryOut: ARRAY[1..5] OF BOOL;
	ret1: BOOL;
	END_VAR	

Program		ret1 TRUE := AryXorN(In1 := arr_int1[1] TRUE , In2 := arr_int2[1] TRUE , Size := diSize 5 , AryOut := abAryOut[1] TRUE);RETURN

	LD	ST

Operation result

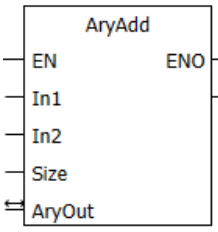
	Name	Data type	Address	Value
1	arr_int1	ARRAY [1..5] OF BOOL		
2	arr_int1[1]	BOOL		TRUE
3	arr_int1[2]	BOOL		FALSE
4	arr_int1[3]	BOOL		TRUE
5	arr_int1[4]	BOOL		FALSE
6	arr_int1[5]	BOOL		TRUE
7	arr_int2	ARRAY [1..5] OF BOOL		
8	arr_int2[1]	BOOL		TRUE
9	arr_int2[2]	BOOL		FALSE
10	arr_int2[3]	BOOL		FALSE
11	arr_int2[4]	BOOL		TRUE
12	arr_int2[5]	BOOL		FALSE
13	diSize	DINT		5
14	abAryOut	ARRAY [1..5] OF BOOL		
15	abAryOut[1]	BOOL		TRUE
16	abAryOut[2]	BOOL		TRUE
17	abAryOut[3]	BOOL		FALSE
18	abAryOut[4]	BOOL		FALSE
19	abAryOut[5]	BOOL		FALSE
20	ret1	BOOL		TRUE

- **Precautions**
  - In1[ ], In2[ ], and AryOut[ ] must be of the same data type. Otherwise, a compilation fault will occur.
  - When the value of Size exceeds the In1[ ], In2[ ], or AryOut[ ] array, the return value is FALSE and AryOut[ ] remains unchanged.
  - When the value of Size is 0, AryOut[ ] remains unchanged and the return value is TRUE.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.15 AryAdd

Adds corresponding elements of two arrays.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryAdd	Array Addition	FC		:= AryAdd(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Array In1[ ]	Array 1 to process	Follows data type	-	Array 1 to process
	Array In2[ ]	Array 2 to process	Follows data type	-	Array 2 to process
	Size	Number of elements to process	Follows data type	1	Number of elements to process
Input/Output	AryOut[ ] array	Processing result array	Follows data type	-	Processing result array

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1[ ] array	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Array In2[ ]	Same data type as elements in In1[ ]																			
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
AryOut [ ] array	Same data type as elements in In1[ ]																			

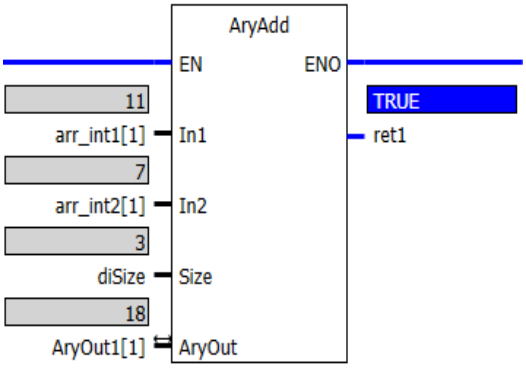
- **Function**

  This instruction adds Size elements starting from the calculation index of array In1[ ] to the corresponding elements of In2[ ]. The result is stored in AryOut[ ].

- **Program example**

	LD	ST
Define variables		

```
VAR
    arr_int1: ARRAY[1..3] OF INT := [11, 23, 35];
    arr_int2: ARRAY[1..3] OF INT := [7, 9, 18];
    diSize: DINT := 3;
    AryOut1: ARRAY[1..3] OF INT;
    ret1: BOOL;
END_VAR
```

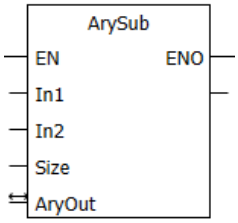
<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>
</td><td>ret1 TRUE := AryAdd(
In1 := arr_int1[1] 11 ,
In2 := arr_int2[1] 7 ,
Size := diSize 3 ,
AryOut := AryOut1[1] 18);RETURN</td></tr>
<tr><td>Operation result</td><td colspan="2">
<table>
<tr><th></th><th></th><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td>1</td><td></td><td>arr_int1</td><td>ARRAY [1..3] OF INT</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td>arr_int1[1]</td><td>INT</td><td></td><td>11</td></tr>
<tr><td>3</td><td></td><td>arr_int1[2]</td><td>INT</td><td></td><td>23</td></tr>
<tr><td>4</td><td></td><td>arr_int1[3]</td><td>INT</td><td></td><td>35</td></tr>
<tr><td>5</td><td></td><td>arr_int2</td><td>ARRAY [1..3] OF INT</td><td></td><td></td></tr>
<tr><td>6</td><td></td><td>arr_int2[1]</td><td>INT</td><td></td><td>7</td></tr>
<tr><td>7</td><td></td><td>arr_int2[2]</td><td>INT</td><td></td><td>9</td></tr>
<tr><td>8</td><td></td><td>arr_int2[3]</td><td>INT</td><td></td><td>18</td></tr>
<tr><td>9</td><td></td><td>diSize</td><td>DINT</td><td></td><td>3</td></tr>
<tr><td>10</td><td></td><td>AryOut1</td><td>ARRAY [1..3] OF INT</td><td></td><td></td></tr>
<tr><td>11</td><td></td><td>AryOut1[1]</td><td>INT</td><td></td><td>18</td></tr>
<tr><td>12</td><td></td><td>AryOut1[2]</td><td>INT</td><td></td><td>32</td></tr>
<tr><td>13</td><td></td><td>AryOut1[3]</td><td>INT</td><td></td><td>53</td></tr>
<tr><td>14</td><td></td><td>ret1</td><td>BOOL</td><td></td><td>TRUE</td></tr>
</table>
</td></tr>
<tr><td>Principle analysis</td><td colspan="2">Add arr_int1[1] and arr_int2[1], and save the result in ArrOut1[1]. The same operation is repeated for Size elements.</td></tr>
</table>

- **Precautions**
  - When the value of Size exceeds the range of In[ ] or AryOut[ ], the return value is FALSE and AryOut[ ] remains unchanged.
  - Elements in In1[ ], In2[ ], and AryOut[ ] must be of the same data type. Otherwise, a compilation fault will occur.
  - If the results exceed the valid range of AryOut[ ], AryOut[ ] will contain elements of illegal values. In this case, no fault will occur. Data in the memory area adjacent to those elements will not be corrupted.
  - When the value of Size is 0, AryOut[ ] remains unchanged and the return value is TRUE.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.16 ArySub

Subtracts corresponding elements of two arrays.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ArySub	Array subtraction	FC		:= ArySub(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Array In1[ ]	Array 1 to process	Follows data type	-	Array 1 to process
	Array In2[ ]	Array 2 to process	Follows data type	-	Array 2 to process
	Size	Number of elements to process	Follows data type	1	Number of elements to process
Input/Output	AryOut[ ] array	Processing result array	Follows data type	-	Processing result array

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1[ ] array	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Array In2[ ]	Same data type as elements in In1[ ]																			
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
AryOut[ ] array	Same data type as elements in In1[ ]																			

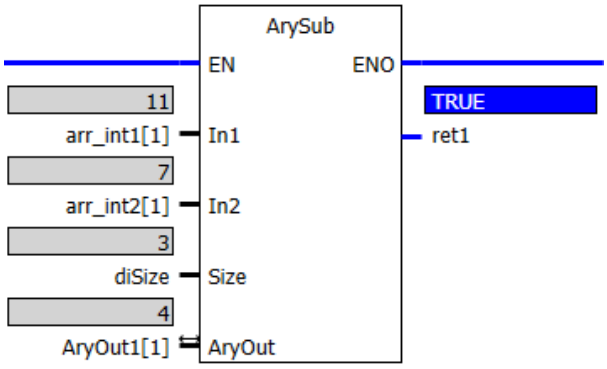
- **Function**

This instruction subtracts Size elements starting from the calculation index of subtrahend array In2[ ] from the corresponding elements of minuend array In1[ ]. The result is stored in AryOut[ ].

- **Program example**

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    arr_int1: ARRAY[1..3] OF INT := [11, 23, 35];
    arr_int2: ARRAY[1..3] OF INT := [7, 9, 18];
    diSize: DINT := 3;
    AryOut1: ARRAY[1..3] OF INT;
    ret1: BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>
</td><td>

```
ret1 TRUE := ArySub(
        In1 := arr_int1[1]  11 ,
        In2 := arr_int2[1]  7 ,
        Size := diSize  3 ,
        AryOut := AryOut1[1]  4 );RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

		Name	Data type	Address	Value
1		arr_int1	ARRAY [1..3] OF INT		
2		arr_int1[1]	INT		11
3		arr_int1[2]	INT		23
4		arr_int1[3]	INT		35
5		arr_int2	ARRAY [1..3] OF INT		
6		arr_int2[1]	INT		7
7		arr_int2[2]	INT		9
8		arr_int2[3]	INT		18
9		diSize	DINT		3
10		AryOut1	ARRAY [1..3] OF INT		
11		AryOut1[1]	INT		4
12		AryOut1[2]	INT		14
13		AryOut1[3]	INT		17
14		ret1	BOOL		TRUE

</td></tr>
<tr><td>Principle analysis</td><td colspan="2">Subtract arr_int2[1] from arr_int1[1], and save the result in ArrOut1[1]. The same operation is repeated for Size elements.</td></tr>
</table>

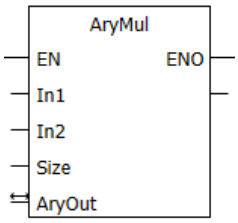
- **Precautions**
  - When the value of Size exceeds the range of In[ ] or AryOut[ ], the return value is FALSE and AryOut[ ] remains unchanged.
  - Elements in In1[ ], In2[ ], and AryOut[ ] must be of the same data type. Otherwise, a compilation fault will occur.
  - If the results exceed the valid range of AryOut[ ], AryOut[ ] will contain elements of illegal values. In this case, no fault will occur. Data in the memory area adjacent to those elements will not be corrupted.
  - When the value of Size is 0, AryOut[ ] remains unchanged and the return value is TRUE.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.

◦ Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.17 AryMul

Multiplies corresponding elements of two arrays.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryMul	Array multiplication	FC		:= AryMul(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Array In1[ ]	Array 1 to process	Follows data type	-	Array 1 to process
	Array In2[ ]	Array 2 to process	Follows data type	-	Array 2 to process
	Size	Number of elements to process	Follows data type	1	Number of elements to process
Input/Output	AryOut[ ] array	Processing result array	Follows data type	-	Processing result array

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1[ ] array	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Array In2[ ]	Same data type as elements in In1[ ]																			
Size	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–
AryOut[ ] array	Same data type as elements in In1[ ]																			

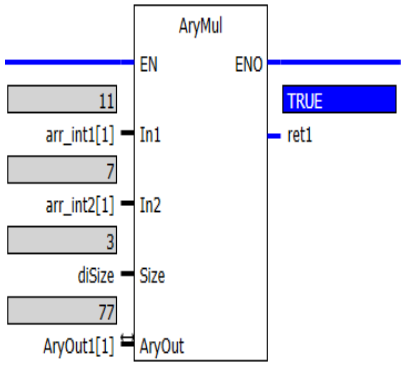
- **Function**

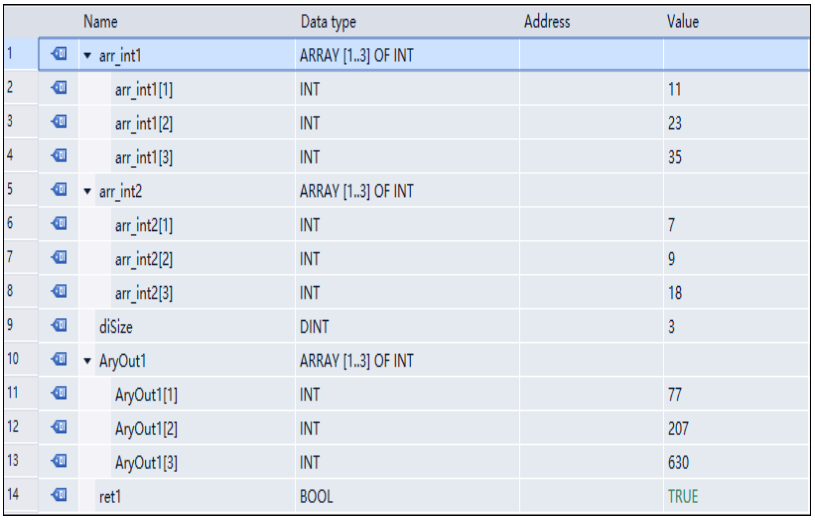
  This instruction multiplies Size elements starting from the calculation index of the array In1[ ] by the corresponding elements of In2[ ]. The result is stored in AryOut[ ].

- **Program example**

	LD	ST

Define variables	VAR
	arr_int1: ARRAY[1..3] OF INT := [11, 23, 35];
	arr_int2: ARRAY[1..3] OF INT := [7, 9, 18];
	diSize: DINT := 3;
	AryOut1: ARRAY[1..3] OF INT;
	ret1: BOOL;
	END_VAR	

Program		ret1 TRUE := AryMul(In1 := arr_int1[1] 11 , In2 := arr_int2[1] 7 , Size := diSize 3 , AryOut := AryOut1[1] 77);RETURN

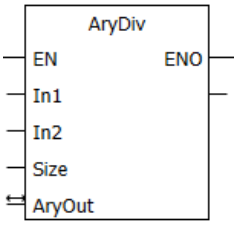
	LD	ST
Operation result		
Principle analysis	Multiply arr_int1[1] by arr_int2[1], and save the result in ArrOut1[1]. The same operation is repeated for Size elements.	

- **Precautions**
  - When the value of Size exceeds the range of In[ ] or AryOut[ ], the return value is FALSE and AryOut[ ] remains unchanged.
  - Elements in In1[ ], In2[ ], and AryOut[ ] must be of the same data type. Otherwise, a compilation fault will occur.
  - If the results exceed the valid range of AryOut[ ], AryOut[ ] will contain elements of illegal values. In this case, no fault will occur. Data in the memory area adjacent to those elements will not be corrupted.
  - When the value of Size is 0, AryOut[ ] remains unchanged and the return value is TRUE.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.18 AryDiv

Divides corresponding elements of two arrays.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryDiv	Array division	FC		:= AryDiv(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Array In1[ ]	Array 1 to process	Follows data type	-	Array 1 to process
	Array In2[ ]	Array 2 to process	Follows data type	-	Array 2 to process
	Size	Number of elements to process	Follows data type	1	Number of elements to process
Input/Output	AryOut[ ] array	Processing result array	Follows data type	-	Processing result array

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1[ ] array	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
Array In2[ ]	Same data type as elements in In1[ ]																			
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
AryOut[ ] array	Same data type as elements in In1[ ]																			

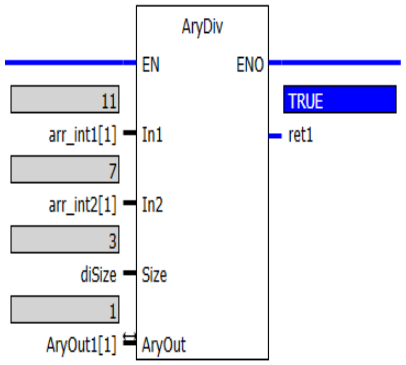
- **Function**

  This instruction divides Size elements starting from the calculation index of the divisor array In2[ ] into the corresponding elements of the dividend array In1[ ]. The result is stored in AryOut[ ].

- **Program example**

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    arr_int1: ARRAY[1..3] OF INT := [11, 23, 35];
    arr_int2: ARRAY[1..3] OF INT := [7, 9, 18];
    diSize: DINT := 3;
    AryOut1: ARRAY[1..3] OF INT;
    ret1: BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>
</td><td>

```
ret1 TRUE := AryDiv(
    In1 := arr_int1[1] 11,
    In2 := arr_int2[1] 7,
    Size := diSize 3,
    AryOut := AryOut1[1] 1 );RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

	Name	Data type	Address	Value
1	arr_int1	ARRAY [1..3] OF INT		
2	arr_int1[1]	INT		11
3	arr_int1[2]	INT		23
4	arr_int1[3]	INT		35
5	arr_int2	ARRAY [1..3] OF INT		
6	arr_int2[1]	INT		7
7	arr_int2[2]	INT		9
8	arr_int2[3]	INT		18
9	diSize	DINT		3
10	AryOut1	ARRAY [1..3] OF INT		
11	AryOut1[1]	INT		1
12	AryOut1[2]	INT		2
13	AryOut1[3]	INT		1
14	ret1	BOOL		TRUE

</td></tr>
<tr><td>Principle analysis</td><td colspan="2">Divide arr_int2[1] into arr_int1[1], and save the result in ArrOut1[1]. The same operation is repeated for Size elements.</td></tr>
</table>

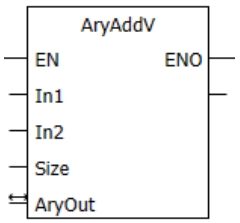
- **Precautions**
  - When the value of Size exceeds the range of In[ ] or AryOut[ ], the return value is FALSE and AryOut[ ] remains unchanged.
  - Elements in In1[ ], In2[ ], and AryOut[ ] must be of the same data type. Otherwise, a compilation fault will occur.
  - If the results exceed the valid range of AryOut[ ], AryOut[ ] will contain elements of illegal values. In this case, no fault will occur. Data in the memory area adjacent to those elements will not be corrupted.
  - When the value of Size is 0, AryOut[ ] remains unchanged and the return value is TRUE.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

◦ When the dividend is 0, the instruction follows the system's zero-division protection rule and returns FALSE.

18.2.16.19 AryAddV

Adds the same value to each element of an array.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryAddV	Array and value addition	FC		:= AryAddV(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1[ ] array	Array to process	Follows data type	-	Array to process
	In2	Divisor variable	Follows data type	-	Divisor variable
	Size	Number of elements to process	Follows data type	1	Number of elements to process
Input/Output	AryOut[ ] array	Processing result array	Follows data type	-	Processing result array

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1[ ] array	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
In2	Same data type as the elements of In1[ ]																			
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
AryOut[ ] array	Same data type as elements in In1[ ]																			

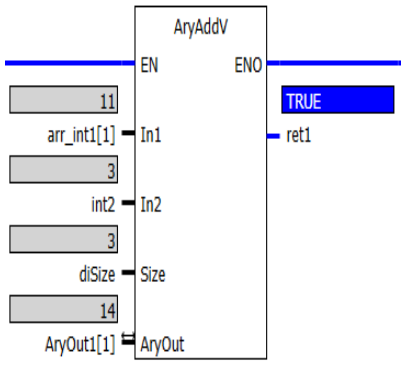
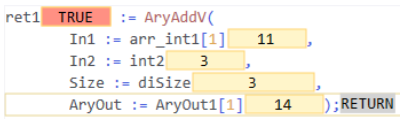
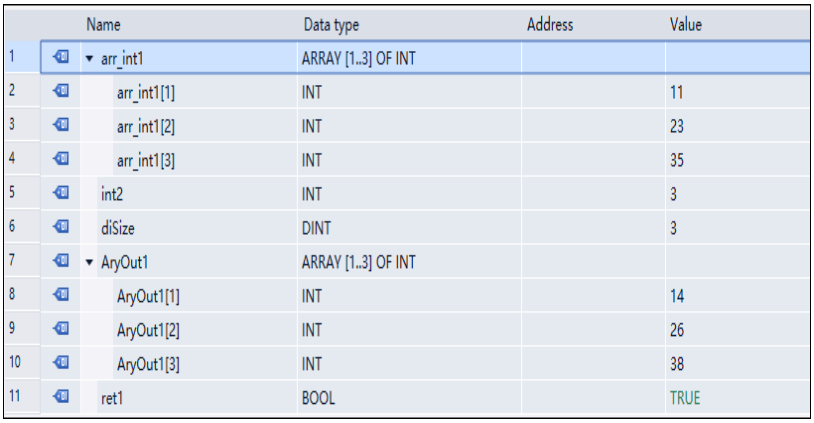
- **Function**

  This instruction adds the value In2 to each of the Size elements of the array In1[ ] starting from In1[0]. The result is stored in AryOut[ ].

- **Program example**

	LD	ST

Define variables	VAR
	arr_int1: ARRAY[1..3] OF INT := [11, 23, 35];
	int2: INT := 3;
	diSize: DINT := 3;
	AryOut1: ARRAY[1..3] OF INT;
	ret1: BOOL;
	END_VAR	

Program		
Operation result		

	Name	Data type	Address	Value
1	arr_int1	ARRAY [1..3] OF INT		
2	arr_int1[1]	INT		11
3	arr_int1[2]	INT		23
4	arr_int1[3]	INT		35
5	int2	INT		3
6	diSize	DINT		3
7	AryOut1	ARRAY [1..3] OF INT		
8	AryOut1[1]	INT		14
9	AryOut1[2]	INT		26
10	AryOut1[3]	INT		38
11	ret1	BOOL		TRUE

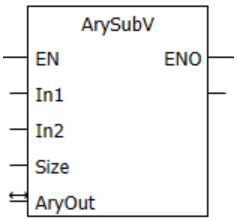
	LD	ST
Principle analysis	Add int2 to arr_int1[1 to 3] respectively, and store the results in ArrOut1[1 to 3].	

- **Precautions**
  - When the value of Size exceeds the range of In[ ] or AryOut[ ], the return value is FALSE and AryOut[ ] remains unchanged.
  - In1[ ], In2[ ], and AryOut[ ] must be of the same data type. Otherwise, a compilation fault will occur.
  - If the results exceed the valid range of AryOut[ ], AryOut[ ] will contain elements of illegal values. In this case, no fault will occur. Data in the memory area adjacent to those elements will not be corrupted.
  - When the value of Size is 0, AryOut[ ] remains unchanged and the return value is TRUE.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.20 ArySubV

Subtracts the same value from each element of an array.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ArySubV	Array and value subtraction	FC		:= ArySubV(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1[ ] array	Array to process	Follows data type	-	Array to process
	In2	Divisor variable	Follows data type	-	Divisor variable
	Size	Number of elements to process	Follows data type	1	Number of elements to process
Input/Output	AryOut[ ] array	Processing result array	Follows data type	-	Processing result array

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1[ ] array	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
In2	Same data type as the elements of In1[ ]																			
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
AryOut[ ] array	Same data type as elements in In1[ ]																			

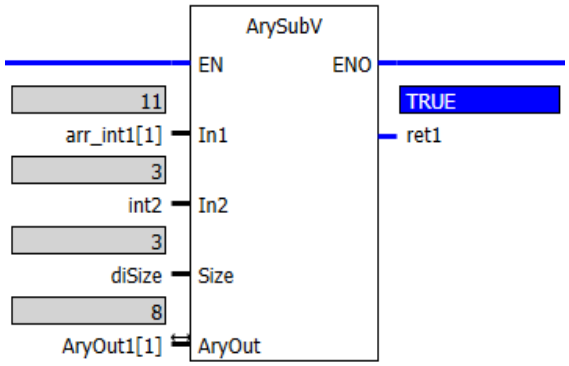
- **Function**

  This instruction subtracts In2 from each of the Size elements in the array In1[ ] starting from In1[0]. The result is stored in AryOut[ ].

- **Program example**

	LD	ST

Define variables	VAR
	arr_int1: ARRAY[1..3] OF INT := [11, 23, 35];
	int2: INT := 3;
	diSize: DINT := 3;
	AryOut1: ARRAY[1..3] OF INT;
	ret1: BOOL;
	END_VAR	

Program		ret1 TRUE := ArySubV(In1 := arr_int1[1] 11 , In2 := int2 3 , Size := diSize 3 , AryOut := AryOut1[1] 8);RETURN

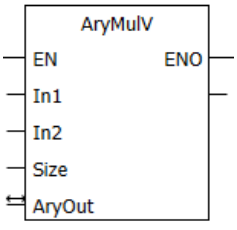
<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Operation result</td><td colspan="2">
<table>
<tr><th></th><th></th><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td>1</td><td></td><td>arr_int1</td><td>ARRAY [1..3] OF INT</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td>arr_int1[1]</td><td>INT</td><td></td><td>11</td></tr>
<tr><td>3</td><td></td><td>arr_int1[2]</td><td>INT</td><td></td><td>23</td></tr>
<tr><td>4</td><td></td><td>arr_int1[3]</td><td>INT</td><td></td><td>35</td></tr>
<tr><td>5</td><td></td><td>int2</td><td>INT</td><td></td><td>3</td></tr>
<tr><td>6</td><td></td><td>diSize</td><td>DINT</td><td></td><td>3</td></tr>
<tr><td>7</td><td></td><td>AryOut1</td><td>ARRAY [1..3] OF INT</td><td></td><td></td></tr>
<tr><td>8</td><td></td><td>AryOut1[1]</td><td>INT</td><td></td><td>8</td></tr>
<tr><td>9</td><td></td><td>AryOut1[2]</td><td>INT</td><td></td><td>20</td></tr>
<tr><td>10</td><td></td><td>AryOut1[3]</td><td>INT</td><td></td><td>32</td></tr>
<tr><td>11</td><td></td><td>ret1</td><td>BOOL</td><td></td><td>TRUE</td></tr>
</table>
</td></tr>
<tr><td>Principle analysis</td><td colspan="2">Subtract int2 from arr_int1[1 to 3] respectively, and store the results in ArrOut1[1 to 3].</td></tr>
</table>

- **Precautions**
  - When the value of Size exceeds the range of In[ ] or AryOut[ ], the return value is FALSE and AryOut[ ] remains unchanged.
  - In1[ ], In2[ ], and AryOut[ ] must be of the same data type. Otherwise, a compilation fault will occur.
  - If the results exceed the valid range of AryOut[ ], AryOut[ ] will contain elements of illegal values. In this case, no fault will occur. Data in the memory area adjacent to those elements will not be corrupted.
  - When the value of Size is 0, AryOut[ ] remains unchanged and the return value is TRUE.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.21 AryMulV

Multiplies each element of an array by the same value.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryMulV	Array and value multiplication	FC		:= AryMulV(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1[ ] array	Array to process	Follows data type	-	Array to process

Input/Output	Variable	Name	Valid Range	Default Value	Description
	In2	Divisor variable	Follows data type	-	Divisor variable
	Size	Number of elements to process	Follows data type	1	Number of elements to process
Input/Output	AryOut[ ] array	Processing result array	Follows data type	-	Processing result array

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1[ ] array	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
In2	Same data type as the elements of In1[ ]																			
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
AryOut[ ] array	Same data type as elements in In1[ ]																			

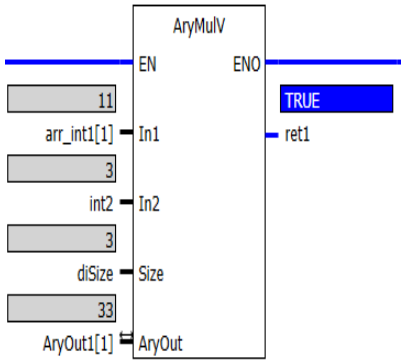
- **Function**

  This instruction multiplies each of the Size elements in the array In1[ ] starting from In1[0] by In2. The result is stored in AryOut[ ].

- **Program example**

	LD	ST

Define variables	VAR
	arr_int1: ARRAY[1..3] OF INT := [11, 23, 35];
	int2: INT := 3;
	diSize: DINT := 3;
	AryOut1: ARRAY[1..3] OF INT;
	ret1: BOOL;
	END_VAR	

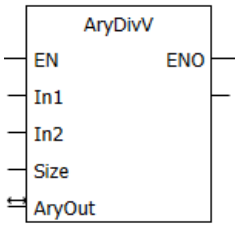
<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>
</td><td>ret1 TRUE := AryMulV(
In1 := arr_int1[1] 11 ,
In2 := int2 3 ,
Size := diSize 3 ,
AryOut := AryOut1[1] 33);RETURN</td></tr>
<tr><td>Operation result</td><td colspan="2">
<table>
<tr><th></th><th></th><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td>1</td><td></td><td>arr_int1</td><td>ARRAY [1..3] OF INT</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td>arr_int1[1]</td><td>INT</td><td></td><td>11</td></tr>
<tr><td>3</td><td></td><td>arr_int1[2]</td><td>INT</td><td></td><td>23</td></tr>
<tr><td>4</td><td></td><td>arr_int1[3]</td><td>INT</td><td></td><td>35</td></tr>
<tr><td>5</td><td></td><td>int2</td><td>INT</td><td></td><td>3</td></tr>
<tr><td>6</td><td></td><td>diSize</td><td>DINT</td><td></td><td>3</td></tr>
<tr><td>7</td><td></td><td>AryOut1</td><td>ARRAY [1..3] OF INT</td><td></td><td></td></tr>
<tr><td>8</td><td></td><td>AryOut1[1]</td><td>INT</td><td></td><td>33</td></tr>
<tr><td>9</td><td></td><td>AryOut1[2]</td><td>INT</td><td></td><td>69</td></tr>
<tr><td>10</td><td></td><td>AryOut1[3]</td><td>INT</td><td></td><td>105</td></tr>
<tr><td>11</td><td></td><td>ret1</td><td>BOOL</td><td></td><td>TRUE</td></tr>
</table>
</td></tr>
<tr><td>Principle analysis</td><td colspan="2">Multiply arr_int1[1 to 3] by int2 respectively, and store the results in ArrOut1[1 to 3].</td></tr>
</table>

- **Precautions**
  - When the value of Size exceeds the range of In[ ] or AryOut[ ], the return value is FALSE and AryOut[ ] remains unchanged.
  - In1[ ], In2[ ], and AryOut[ ] must be of the same data type. Otherwise, a compilation fault will occur.
  - If the results exceed the valid range of AryOut[ ], AryOut[ ] will contain elements of illegal values. In this case, no fault will occur. Data in the memory area adjacent to those elements will not be corrupted.
  - When the value of Size is 0, AryOut[ ] remains unchanged and the return value is TRUE.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.22 AryDivV

Divides each element of an array by the same value.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryDivV	Array and value division	FC		:= AryDivV(In1 := , In2 := , Size := , AryOut :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In1[ ] array	Array to process	Follows data type	-	Array to process
	In2	Divisor variable	Follows data type	-	Divisor variable
	Size	Number of elements to process	Follows data type	1	Number of elements to process
Input/Output	AryOut[ ] array	Processing result array	Follows data type	-	Processing result array

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In1[ ] array	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	–	–	–	–	–
In2	Same data type as the elements of In1[ ]																			
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
AryOut[ ] array	Same data type as elements in In1[ ]																			

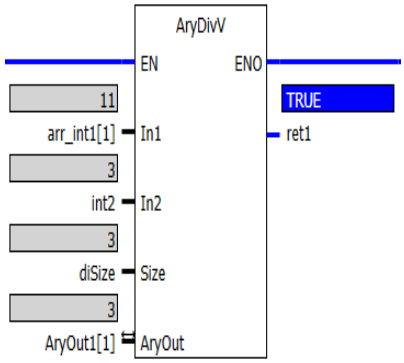
- **Function**

This instruction divides In2 into each of the Size elements in the dividend array In1[ ] starting from In1[0]. The result is stored in AryOut[ ].

- **Program example**

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    arr_int1: ARRAY[1..3] OF INT := [11, 23, 35];
    int2: INT := 3;
    diSize: DINT := 3;
    AryOut1: ARRAY[1..3] OF INT;
    ret1: BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>
</td><td>

```
ret1  TRUE  := AryDivV(
      In1 := arr_int1[1]   11  ,
      In2 := int2    3  ,
      Size := diSize    3  ,
      AryOut := AryOut1[1]   3  );RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

	Name	Data type	Address	Value
1	arr_int1	ARRAY [1..3] OF INT		
2	arr_int1[1]	INT		11
3	arr_int1[2]	INT		23
4	arr_int1[3]	INT		35
5	int2	INT		3
6	diSize	DINT		3
7	AryOut1	ARRAY [1..3] OF INT		
8	AryOut1[1]	INT		3
9	AryOut1[2]	INT		7
10	AryOut1[3]	INT		11
11	ret1	BOOL		TRUE

</td></tr>
<tr><td>Principle analysis</td><td colspan="2">Divide int2 into arr_int1[1 to 3] respectively, and store the results in ArrOut1[1 to 3].</td></tr>
</table>

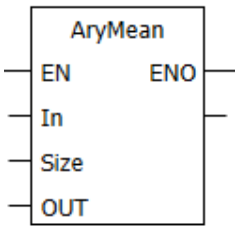
- **Precautions**
  - When the value of Size exceeds the range of In[ ] or AryOut[ ], the return value is FALSE and AryOut[ ] remains unchanged.
  - In1[ ], In2[ ], and AryOut[ ] must be of the same data type. Otherwise, a compilation fault will occur.
  - If the results exceed the valid range of AryOut[ ], AryOut[ ] will contain elements of illegal values. In this case, no fault will occur. Data in the memory area adjacent to those elements will not be corrupted.
  - When the value of Size is 0, AryOut[ ] remains unchanged and the return value is TRUE.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

◦ When the dividend is 0, the instruction follows the system's zero-division protection rule and returns FALSE.

18.2.16.23 AryMean

Calculates the mean of the elements of an array.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryMean	Array mean calculation	FC		:= AryMean(In := , Size := , OUT :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In[ ] array	Array to process	Follows data type	-	Array to process
	Size	Number of elements to process	Follows data type	1	Number of elements to process
	Out	Operation result	Follows data type	-	Mean value

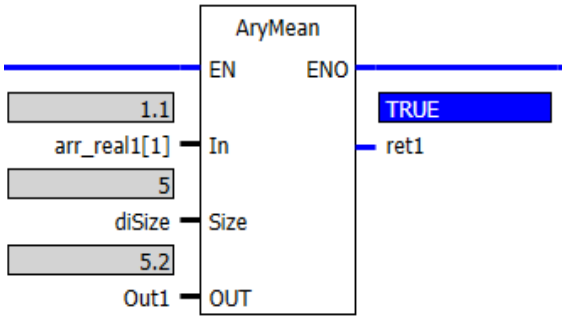
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	√	–	–	–	–
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Out	Same data type as elements in In1[ ]																			

- **Function**

This instruction calculates the mean of elements of the array In[ ].

- **Program example**

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">
<pre>
VAR
    arr_real1: ARRAY[1..5] OF REAL := [1.1, 3.4, 5.5, 7, 9];
    diSize: DINT := 5;
    Out1: REAL;
    ret1: BOOL;
END_VAR
</pre>
</td></tr>
<tr><td>Program</td><td>


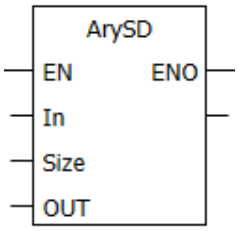
</td><td>
<pre>
ret1 := AryMean(
    In := arr_real1[1],
    Size := diSize,
    OUT := Out1);RETURN
</pre>
</td></tr>
<tr><td>Operation result</td><td colspan="2">
<table>
<tr><th></th><th></th><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td>1</td><td></td><td>arr_real1</td><td>ARRAY [1..5] OF REAL</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td>arr_real1[1]</td><td>REAL</td><td></td><td>1.1</td></tr>
<tr><td>3</td><td></td><td>arr_real1[2]</td><td>REAL</td><td></td><td>3.4</td></tr>
<tr><td>4</td><td></td><td>arr_real1[3]</td><td>REAL</td><td></td><td>5.5</td></tr>
<tr><td>5</td><td></td><td>arr_real1[4]</td><td>REAL</td><td></td><td>7</td></tr>
<tr><td>6</td><td></td><td>arr_real1[5]</td><td>REAL</td><td></td><td>9</td></tr>
<tr><td>7</td><td></td><td>diSize</td><td>DINT</td><td></td><td>5</td></tr>
<tr><td>8</td><td></td><td>Out1</td><td>REAL</td><td></td><td>5.2</td></tr>
<tr><td>9</td><td></td><td>ret1</td><td>BOOL</td><td></td><td>TRUE</td></tr>
</table>
</td></tr>
<tr><td>Principle analysis</td><td colspan="2">Sum arr_real1[1] to arr_real1[5] and calculate the mean value of the sum.</td></tr>
</table>

- **Precautions**
  - Arithmetic operations are not supported when the elements of In[ ] are positive infinity, negative infinity, or non-numeric data.
  - When the data types of In[ ] and Out are different, a fault occurs during compilation.
  - When the value of Size exceeds the array range of In[ ], the return value is FALSE, and Out remains unchanged.
  - When In[ ] or Out is an integer, the decimal portion of the mean is truncated.
  - When the value of Size is 0, Out is 0 and the return value is TRUE.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.24 ArySD

Calculates the standard deviation of the elements in an array.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ArySD	Array standard deviation calculation	FC		:= ArySD(In := , Size := , OUT :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In[ ] array	Array to process	Follows data type	-	Array to process
	Size	Number of elements to process	Follows data type	2	Number of elements to process
	Out	Operation result	Follows data type	-	Standard deviation

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Out	–	–	–	–	–	–	–	–	–	–	–	–	–	√	√	–	–	–	–	–

- **Function**

This instruction calculates the standard deviation of Size elements of the array In[ ] starting from In[0].

Standard deviation calculation formula:

$$\frac{\sqrt{\sum_i (In[i]-InM)^2}}{Size-1}$$

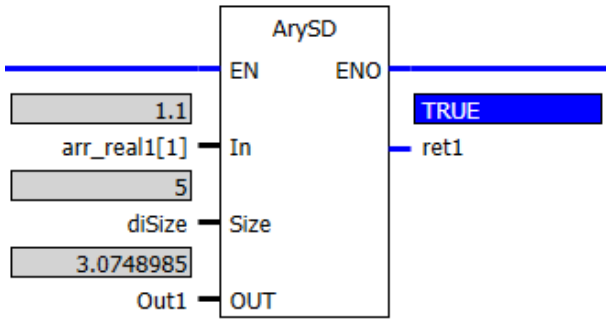
i: Suffix of In[ ] (0 to Size-1).

InM: Mean value of In[0] to In[Size-1].

- **Program example**

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    arr_real1: ARRAY[1..5] OF REAL := [1.1, 3.4, 5.5, 7, 9];
    diSize: DINT := 5;
    Out1: REAL;
    ret1: BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>


</td><td>

```
ret1 := ArySD(
    In := arr_real1[1],
    Size := diSize,
    OUT := Out1);RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

	Name	Data type	Address	Value
1	arr_real1	ARRAY [1..5] OF REAL		
2	arr_real1[1]	REAL		1.1
3	arr_real1[2]	REAL		3.4
4	arr_real1[3]	REAL		5.5
5	arr_real1[4]	REAL		7
6	arr_real1[5]	REAL		9
7	diSize	DINT		5
8	Out1	REAL		3.0748985
9	ret1	BOOL		TRUE

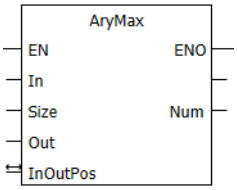
</td></tr>
<tr><td>Principle analysis</td><td colspan="2">Calculate the mean value of the elements to process, and then calculate the standard deviation based on the above formula.</td></tr>
</table>

- **Precautions**
  - In ArrayUtils library versions 1.2.0.1 and earlier, when the value of Size is 0 or 1, Out is set to 0 and TRUE is returned.
    In ArrayUtils library versions 1.3.0.0 and later, when the value of Size is less than 2, Out is set to 0 and FALSE is returned.
  - Elements in the input array In[ ] and Out must be of the same data type.
  - When an intermediate value in the calculation process exceeds the valid range of In[ ], the value of Out is an illegal value. In this case, no fault will occur.
  - When the value of Size exceeds the array range of In[ ], the return value is FALSE, and Out remains unchanged.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.25 AryMax

Gets the maximum value in a one-dimensional array.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryMax	Maximum array value retrieval	FC		:= AryMax(In := , Size := , Out := , InOutPos := , Num =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In[ ] array	Array to search	Follows data type	-	Array to search
	Size	Number of elements to search	Follows data type	-	Number of elements in In[ ] to search
	Out	Search result	Follows data type	1	Search result
Input/Output	InOutPos	Index of the retrieved element	Follows data type	-	Index of the retrieved element
Output	Num	Number of times the key value is retrieved	Follows data type	-	Number of times the key value is retrieved

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
InOutPos	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Out	Data with the same data type as elements in In[ ]																			
Num	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **Function**

  This instruction gets the minimum value of the Size elements in the array In[ ] starting from In[0]. The retrieval result is assigned to Out, the index of the retrieved element is assigned to InOutPos, and the number of times that the minimum value is retrieved is assigned to Num. If Num is greater than 1, the lowest index of the retrieved element is assigned to InOutPos. The relationship between values with data types that are not integers or real numbers is determined as given in the following table.

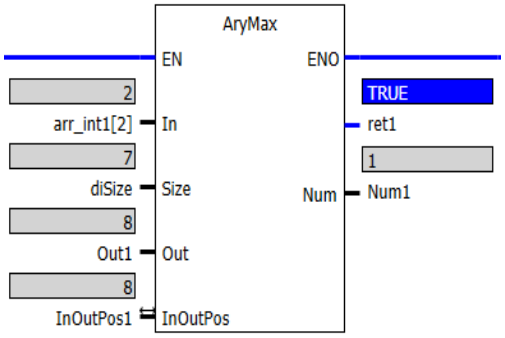
Data Type	Relationship
TIME	The numerically larger value is considered to be larger.
DATE, TOD, DT	The later date or time is considered to be larger.
STRING	First, the first character encodings in all strings are compared. If the encodings are different, the size of string is determined by the corresponding encoding values. If the encodings are the same, compare the encodings of the second, third, and subsequent characters until a difference is found. If the lengths of the strings are different, NULL characters (16#00) are added to the shorter string to complete the comparison.

- **Program example**

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    arr_int1: ARRAY[0..9] OF INT := [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];
    diSize: DINT := 7;
    Out1: INT;
    InOutPos1: DINT;
    Num1: DINT;
    ret1: BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



</td><td>

```
ret1 TRUE := AryMax(
    In := arr_int1[2] 2 ,
    Size := diSize 7 ,
    Out := Out1 8 ,
    InOutPos := InOutPos1 8 ,
    Num => Num1 1 );RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

	Name	Data type	Address	Value
1	arr_int1	ARRAY [0..9] OF INT		
2	arr_int1[0]	INT		0
3	arr_int1[1]	INT		1
4	arr_int1[2]	INT		2
5	arr_int1[3]	INT		3
6	arr_int1[4]	INT		4
7	arr_int1[5]	INT		5
8	arr_int1[6]	INT		6
9	arr_int1[7]	INT		7
10	arr_int1[8]	INT		8
11	arr_int1[9]	INT		9
12	diSize	DINT		7
13	Out1	INT		8
14	InOutPos1	DINT		8
15	Num1	DINT		1
16	ret1	BOOL		TRUE

</td></tr>
</table>

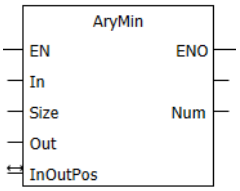
- **Precautions**
  - When the value of Size exceeds the array range of In[ ], the return value is FALSE, and Out remains unchanged.
  - When In[ ] is not a one-dimensional array, a fault occurs during compilation.
  - Ensure that the data type of Out is the same as that of the In[ ] array; otherwise, a fault occurs during compilation.
  - The index of the retrieved element is assigned to InOutPos.
  - When In[ ] is a real number, the result may be unexpected due to operation errors.
  - When the value of Size is 0, the value of Num changes to 0, the values of InOutPos and Out remain unchanged, and the return value is TRUE.
  - When the data type of In[ ] is not supported, a fault occurs during compilation.
  - When the data type of In[ ] and Out is STRING, the size of In[ ] shall not exceed that of Out; otherwise, the return value is FALSE and Out remains unchanged.

- When the data type of In[ ] and "Out" is STRING, the value range supports a maximum of 256 characters. Characters beyond the range will be truncated.
- Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.26 AryMin

Gets the minimum value in a one-dimensional array.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryMin	Minimum array value retrieval	FC		:= AryMin(In := , Size := , Out := , InOutPos := , Num =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	In[ ] array	Array to search	Follows data type	-	Array to search
	Size	Number of elements to search	Follows data type	-	Number of elements in In[ ] to search
	Out	Search result	Follows data type	1	Search result
Input/Output	InOutPos	Index of the retrieved element	Follows data type	-	Index of the retrieved element
Output	Num	Number of times the key value is retrieved	Follows data type	-	Number of times the key value is retrieved

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	–	–	–	–	–	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
InOutPos	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Out	Data with the same data type as elements in In[ ]																			
Num	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **Function**

  This instruction gets the minimum value of the Size elements in the array In[ ] starting from In[0]. The retrieval result is assigned to Out, the index of the retrieved element is assigned to InOutPos, and the number of times that the minimum value is retrieved is assigned to Num. If Num is greater than 1, the lowest index of the retrieved element is assigned to InOutPos. The relationship between values with data types that are not integers or real numbers is determined as given in the following table.

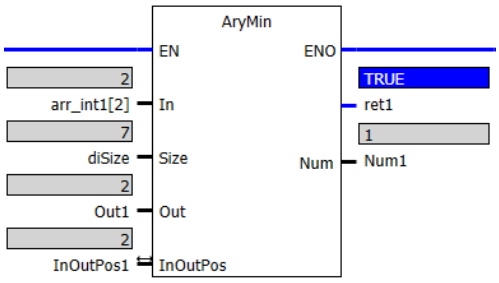
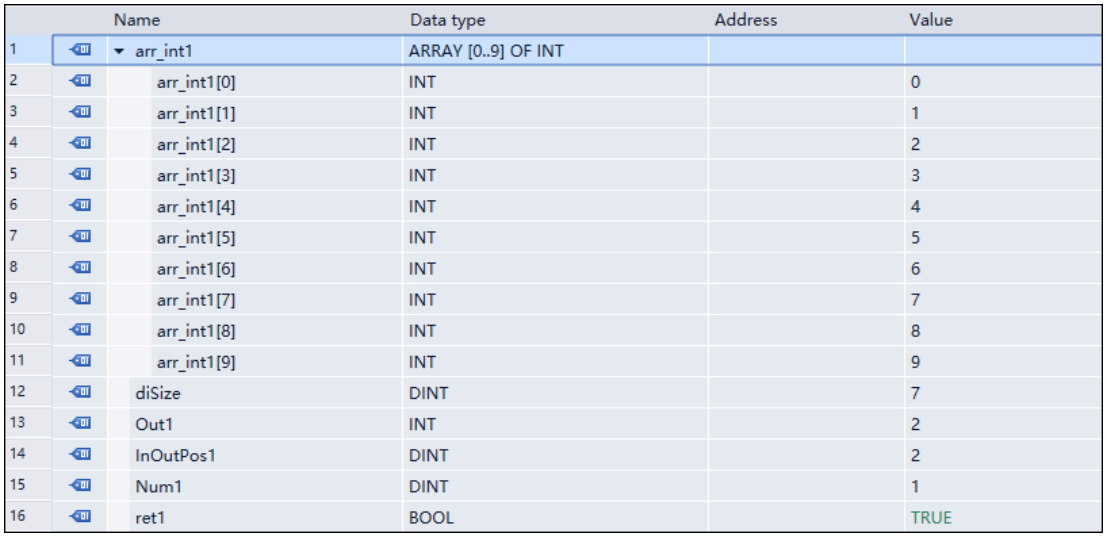
Data Type	Relationship
TIME	The numerically larger value is considered to be larger.
DATE, TOD, DT	The later date or time is considered to be larger.

Data Type	Relationship
STRING	Compare the character encodings of the first character of each string. If the encodings are different, the size of string is determined by the corresponding encoding values. If the encodings are the same, compare the encodings of the second, third, and subsequent characters until a difference is found. If the lengths of the strings are different, NULL characters (16#00) are added to the shorter string to complete the comparison.

- **Program example**

	LD	ST

Define variables		VAR
		arr_int1: ARRAY[0..9] OF INT := [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];
		diSize: DINT := 7;
		Out1: INT;
		InOutPos1: DINT;
		Num1: DINT;
		ret1: BOOL;
		END_VAR

Program		ret1 TRUE := AryMin(In := arr_int1[2] 2 , Size := diSize 7 , Out := Out1 2 , InOutPos := InOutPos1 2 , Num => Num1 1);RETURN
Operation result		

	Name	Data type	Address	Value
1	arr_int1	ARRAY [0..9] OF INT		
2	arr_int1[0]	INT		0
3	arr_int1[1]	INT		1
4	arr_int1[2]	INT		2
5	arr_int1[3]	INT		3
6	arr_int1[4]	INT		4
7	arr_int1[5]	INT		5
8	arr_int1[6]	INT		6
9	arr_int1[7]	INT		7
10	arr_int1[8]	INT		8
11	arr_int1[9]	INT		9
12	diSize	DINT		7
13	Out1	INT		2
14	InOutPos1	DINT		2
15	Num1	DINT		1
16	ret1	BOOL		TRUE

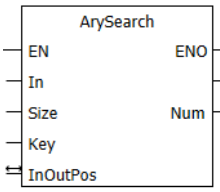
- **Precautions**
  - When the value of Size exceeds the array range of In[ ], the return value is FALSE, and Out remains unchanged.

- When In[ ] is not a one-dimensional array, a fault occurs during compilation.
- Ensure that the data type of Out is the same as that of the In[ ] array; otherwise, a fault occurs during compilation.
- The index of the retrieved element is assigned to InOutPos.
- When In[ ] is a real number, the result may be unexpected due to operation errors.
- When the value of Size is 0, the value of Num changes to 0, the values of InOutPos and Out remain unchanged, and the return value is TRUE.
- When the data type of In[ ] is not supported, a fault occurs during compilation.
- When the data type of In[ ] and Out is STRING, the size of In[ ] shall not exceed that of Out; otherwise, the return value is FALSE and Out remains unchanged.
- When the data type of In[ ] and "Out" is STRING, the value range supports a maximum of 256 characters. Characters beyond the range will be truncated.
- Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.27 ArySearch

Gets the specified value in a one-dimensional array.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ArySearch	Array search	FC		:= ArySearch(In := , Size := , Key := , InOutPos := , Num =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	In[ ] array	Array to search	Follows data type	-	Array to search
	Size	Number of elements to search	Follows data type	1	Number of elements in In[ ] to search
	Key	Search keyword	Follows data type	-	Search keyword
Input/ Output	InOutPos	Index of the retrieved element	Follows data type	-	Index of the retrieved element in the target array In[ ]

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Output	Num	Number of times the key value is retrieved	Follows data type	-	Number of times the key value is retrieved

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
In[ ] array	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Arrays of enumerations can also be specified.																			
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Key	Data with the same data type as elements in In[ ]																			
InOutPos	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **Function**

  This instruction searches Size elements in the one-dimensional array In[ ] starting from In[0] for elements with the same value as the search keyword Key.

  The values of the element index InOutPos and the number of times that the key value is retrieved Num are as shown in the table below.

Element with Same Value As Key	"InOutPos"	"Num"
Exists	Index of the lowest element among those with the same value as Key	Number of elements with the same value as Key
Does not exist	No change	0

The relationship between values with data types that are not integers or real numbers is determined as given in the following table.

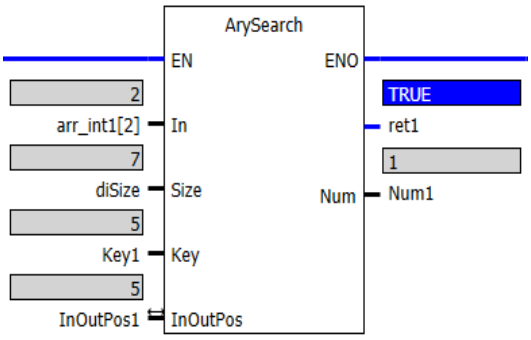
Data Type	Relationship
TIME	The numerically larger value is considered to be larger.
DATE	The later date is considered to be larger.
STRING	First, the first character encodings in all strings are compared. If the encodings are different, the size of string is determined by the corresponding encoding values. If the encodings are the same, compare the encodings of the second, third, and subsequent characters until a difference is found. If the lengths of the strings are different, NULL characters (16#00) are added to the shorter string to complete the comparison.

- **Program example**

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    arr_int1: ARRAY[0..9] OF INT := [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];
    diSize: DINT := 7;
    Key1: INT := 5;
    InOutPos1: DINT;
    Num1: DINT;
    ret1: BOOL;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>



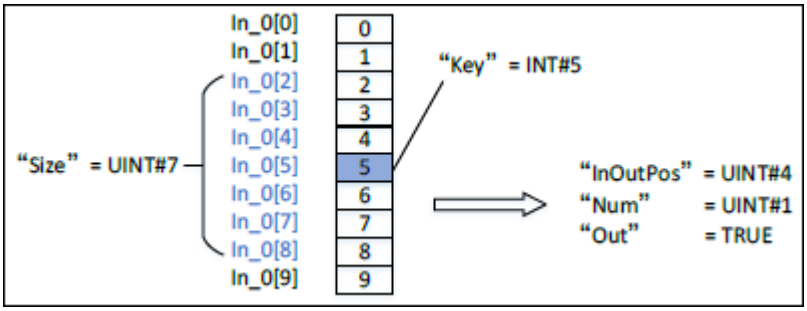
</td><td>

```
ret1 TRUE := ArySearch(
        In := arr_int1[2] 2 ,
        Size := diSize 7 ,
        Key := Key1 5 ,
        InOutPos := InOutPos1 5 ,
        Num => Num1 1 );RETURN
```

</td></tr>
<tr><td>Operation result</td><td colspan="2">

	Name	Data type	Address	Value
1	arr_int1	ARRAY [0..9] OF INT		
2	arr_int1[0]	INT		0
3	arr_int1[1]	INT		1
4	arr_int1[2]	INT		2
5	arr_int1[3]	INT		3
6	arr_int1[4]	INT		4
7	arr_int1[5]	INT		5
8	arr_int1[6]	INT		6
9	arr_int1[7]	INT		7
10	arr_int1[8]	INT		8
11	arr_int1[9]	INT		9
12	diSize	DINT		7
13	Key1	INT		5
14	InOutPos1	DINT		5
15	Num1	DINT		1
16	ret1	BOOL		TRUE

</td></tr>
</table>

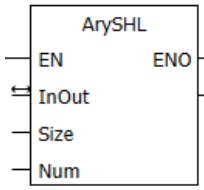
	LD	ST
Principle analysis		

- **Precautions**
  - When the value of Size exceeds the AryOut[ ] array, the return value is FALSE and the value of Num is initialized to 0.
  - When In[ ] is not a one-dimensional array, a fault occurs during compilation.
  - Ensure that the data type of Key is the same as that of the In[ ] array; otherwise, a fault occurs during compilation.
  - The index of the retrieved element is assigned to InOutPos.
  - When the value of Size is 0, the value of Num is initialized to 0, InOutPos remains unchanged, and the return value is TRUE.
  - When the data type of In[ ] and Key is STRING, the value range supports a maximum of 256 characters. Characters beyond the range will be truncated. When the value exceeds the value range, the return value is FALSE and the value of Num is initialized to 0.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.28 ArySHL

Shifts multiple array elements to the left (toward the higher elements).

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ArySHL	Shift N-bits left with carry	FC		:= ArySHL(InOut := , Size := , Num :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Size	Number of elements in the shift register array	Follows data type	1	Number of elements in the shift register array

Input/Output	Variable	Name	Valid Range	Default Value	Description
	Num	Number of bits to shift	Follows data type	1	Number of bits to shift
Input/Output	InOut[ ] array	Array where the shift object is located	Follows data type	-	Array where the shift object is located

Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
InOut[ ] array	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Arrays of structures can also be specified.																			
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **Function**

  This instruction shifts the lower Size elements in shift register array InOut[ ] to the left by Num elements.

  Elements that are shifted out of the array are discarded.

  The empty elements in InOut[ ] after shifting are assigned with the initial values of the corresponding data types.

  The following table describes default values for various data types.

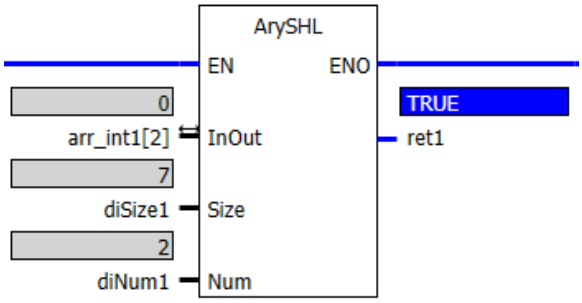
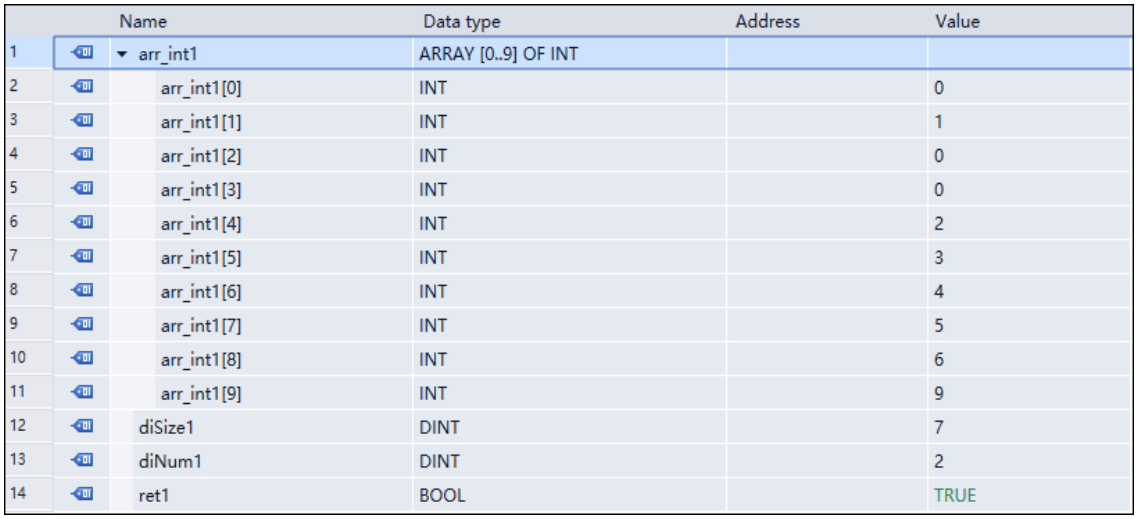
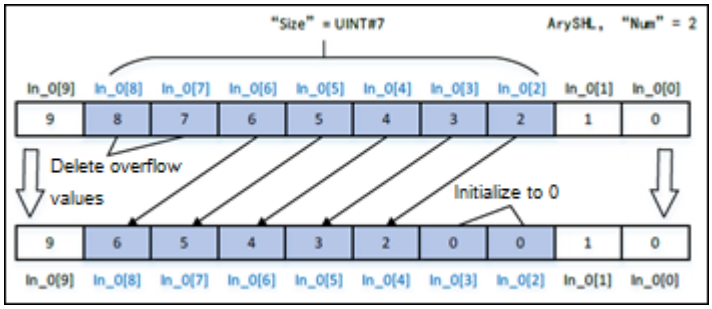
Data Type	Initial Value
BOOL	FALSE
BYTE, WORD, DWORD, LWORD	16#0
USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, LREAL	0
TIME	T#0ms
DATE	D#1970-1-1

Data Type	Initial Value
TOD	TOD#0:0:0
DT	DT#1970-1-1-0:0:0
STRING	''

- **Program example**

	LD	ST

Define variables	VAR
	arr_int1: ARRAY[0..9] OF INT := [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];
	diSize1: DINT := 7;
	diNum1: DINT := 2;
	ret1: BOOL;
	END_VAR	

Program		ret1 TRUE := ArySHL(InOut := arr_int1[2] 0 , Size := diSize1 7 , Num := diNum1 2);RETURN
Operation result		
Principle analysis		

	Name	Data type	Address	Value
1	arr_int1	ARRAY [0..9] OF INT		
2	arr_int1[0]	INT		0
3	arr_int1[1]	INT		1
4	arr_int1[2]	INT		0
5	arr_int1[3]	INT		0
6	arr_int1[4]	INT		2
7	arr_int1[5]	INT		3
8	arr_int1[6]	INT		4
9	arr_int1[7]	INT		5
10	arr_int1[8]	INT		6
11	arr_int1[9]	INT		9
12	diSize1	DINT		7
13	diNum1	DINT		2
14	ret1	BOOL		TRUE

- **Precautions**
  - In ArrayUtils library versions 1.2.0.0 and earlier, when the value of Num is 0, no shifting is performed, and TRUE is returned (this condition has the highest priority).
    In ArrayUtils library versions 1.2.0.1 and later, when the value of Num is 0, no shifting is performed; in

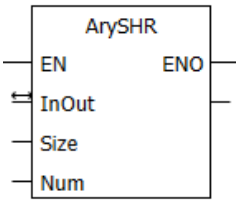
this case, if the value of Size exceeds the array area of InOut[ ], FALSE is returned; otherwise, TRUE is returned.

- When the value of Size is 0, no shifting is performed, and TRUE is returned.
- When the value of Num is greater than Size, all values from InOut[0] to InOut[Size–1] are initialized to 0.
- If the value of "Size" exceeds the InOut[ ] array, the return value is FALSE and InOut[ ] does not change.
- If the dimension of the input array exceeds 3, the return value is FALSE, and InOut[ ] remains unchanged.
- Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.29 ArySHR

Shifts multiple array elements to the right (toward the lower elements).

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ArySHR	Shift N-bits right with carry	FC		:= ArySHR(InOut := , Size := , Num :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Size	Number of elements in the shift register array	Follows data type	1	Number of elements in the shift register array
	Num	Number of bits to shift	Follows data type	1	Number of bits to shift
Input/Output	InOut[ ] array	Array where the shift object is located	Follows data type	-	Array where the shift object is located

Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
InOut[ ] array	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	Arrays of structures can also be specified.																			
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Num	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **Function**

  This instruction shifts the higher Size elements in shift register array InOut[ ] to the right by Num elements. Elements that are shifted out of the array are discarded. The empty elements in InOut[ ] after shifting are assigned with the initial values of the corresponding data types.

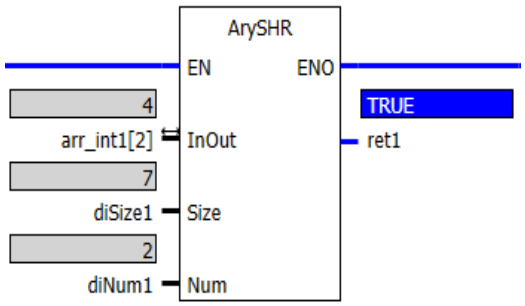
  The following table describes default values for various data types.

Data Type	Initial Value
BOOL	FALSE
BYTE, WORD, DWORD, LWORD	16#0
USINT, UINT, UDINT, ULINT, SINT, INT, DINT, LINT, REAL, LREAL	0
TIME	T#0ms
DATE	D#1970-1-1
TOD	TOD#0:0:0
DT	DT#1970-1-1-0:0:0
STRING	''

- **Program example**

	LD	ST

Define variables	VAR
	arr_int1: ARRAY[0..9] OF INT := [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];
	diSize1: DINT := 7;
	diNum1: DINT := 2;
	ret1: BOOL;
	END_VAR	

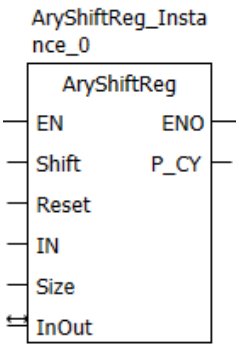
<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>
</td><td>ret1 TRUE := ArySHR(
InOut := arr_int1[2] 4 ,
Size := diSize1 7 ,
Num := diNum1 2);RETURN</td></tr>
<tr><td>Operation result</td><td colspan="2">
<table>
<tr><th></th><th></th><th>Name</th><th>Data type</th><th>Address</th><th>Value</th></tr>
<tr><td>1</td><td></td><td>arr_int1</td><td>ARRAY [0..9] OF INT</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td>arr_int1[0]</td><td>INT</td><td></td><td>0</td></tr>
<tr><td>3</td><td></td><td>arr_int1[1]</td><td>INT</td><td></td><td>1</td></tr>
<tr><td>4</td><td></td><td>arr_int1[2]</td><td>INT</td><td></td><td>4</td></tr>
<tr><td>5</td><td></td><td>arr_int1[3]</td><td>INT</td><td></td><td>5</td></tr>
<tr><td>6</td><td></td><td>arr_int1[4]</td><td>INT</td><td></td><td>6</td></tr>
<tr><td>7</td><td></td><td>arr_int1[5]</td><td>INT</td><td></td><td>7</td></tr>
<tr><td>8</td><td></td><td>arr_int1[6]</td><td>INT</td><td></td><td>8</td></tr>
<tr><td>9</td><td></td><td>arr_int1[7]</td><td>INT</td><td></td><td>0</td></tr>
<tr><td>10</td><td></td><td>arr_int1[8]</td><td>INT</td><td></td><td>0</td></tr>
<tr><td>11</td><td></td><td>arr_int1[9]</td><td>INT</td><td></td><td>9</td></tr>
<tr><td>12</td><td></td><td>diSize1</td><td>DINT</td><td></td><td>7</td></tr>
<tr><td>13</td><td></td><td>diNum1</td><td>DINT</td><td></td><td>2</td></tr>
<tr><td>14</td><td></td><td>ret1</td><td>BOOL</td><td></td><td>TRUE</td></tr>
</table>
</td></tr>
</table>

- **Precautions**
  - In ArrayUtils library versions 1.2.0.0 and earlier, when the value of Num is 0, no shifting is performed, and TRUE is returned (this condition has the highest priority).
    In ArrayUtils library versions 1.2.0.1 and later, when the value of Num is 0, no shifting is performed; in this case, if the value of Size exceeds the array area of InOut[ ], FALSE is returned; otherwise, TRUE is returned.
  - When the value of Size is 0, no shifting is performed, and TRUE is returned.
  - When the value of Num is greater than Size, all values from InOut[0] to InOut[Size–1] are initialized to 0.
  - If the value of "Size" exceeds the InOut[ ] array, the return value is FALSE and InOut[ ] does not change.
  - If the dimension of the input array exceeds 3, the return value is FALSE, and InOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.30 AryShiftReg

Shifts an array to the left by one bit and inserts the input value to the least-significant bit.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryShiftReg	Shift register	FB		AryShiftReg_Instance_0(Shift := , Reset := , IN := , Size := , InOut := , P_CY =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Shift	Shift	Follows data type	FALSE	Shift when the signal changes to TRUE
	Reset	Reset a bit	Follows data type	FALSE	Reset when the signal changes to TRUE
	In	Input Value	Follows data type	FALSE	Value to insert to the least-significant bit of InOut[ ]
	Size	Number of elements to process	Follows data type	1	Number of elements to use as a shift register in InOut[ ]
Input/Output	InOut[ ] array	Array where the shift object is located	Follows data type	-	Array where the shift object is located
Output	P_CY	Carry flag	Follows data type	-	Value stored in the carry flag

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Shift	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
In	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
InOut[ ] array	√	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
P_CY	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

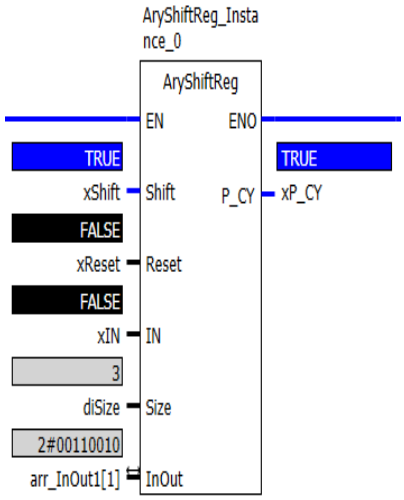
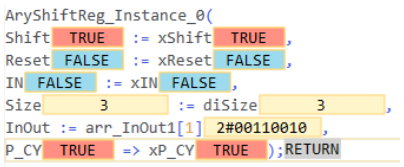
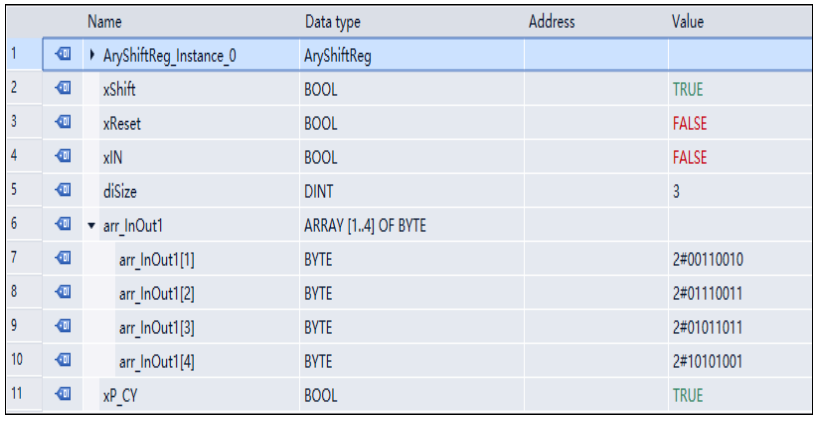
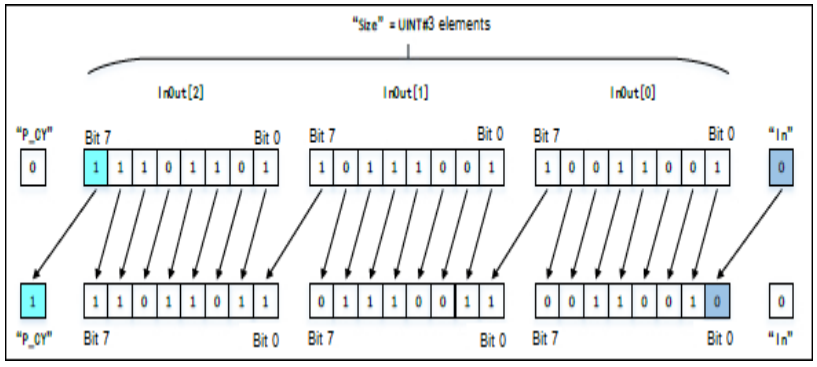
  This instruction shifts Size element in InOut[ ] starting from InOut[0] to the left (toward the most-significant bit) by one bit when Shift changes from FALSE to TRUE. The input value In is inserted into the least-significant bit. The most-significant bit shifted out of the array is output to the carry (CY) flag P_CY.

  When Reset is TRUE, CY and all bits of the Size elements starting from InOut[0] are set to FALSE.

- **Program example**

	LD	ST

Define variables	VAR
	AryShiftReg_0: AryShiftReg;
	xshift: BOOL;
	xReset: BOOL;
	disize1: DINT := 3;
	xIn: BOOL;
	xP_CY: BOOL;
	inout1:ARRAY[1..4] OF BYTE :=[2#10011001,2#10111001,2#11101101,2#10101101];
	END_VAR	

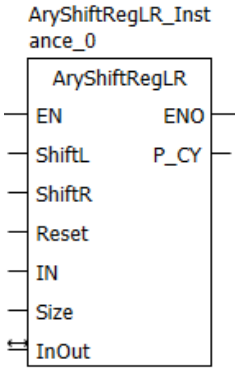
<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>
</td><td>
</td></tr>
<tr><td>Operation result</td><td>
</td><td></td></tr>
<tr><td>Principle analysis</td><td>
</td><td></td></tr>
</table>

- **Precautions**
  - When Reset is TRUE, no shift is performed even if Shift changes from FALSE to TRUE.
  - When the value of Size is 0, InOut[ ] remains unchanged.
  - When the value of Size exceeds the array area of InOut[ ], InOut[ ] remains unchanged.
  - If the dimension of the input array exceeds 3, AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.16.31 AryShiftRegLR

Shifts an array to the left or right by one bit and inserts the input value to the least-significant bit or most-significant bit.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AryShiftRegLR	Reversible shift register	FB		AryShiftRegLR_Instance_0(ShiftL := , ShiftR := , Reset := , IN := , Size := , InOut := , P_CY =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	ShiftL	Left shift	Follows data type	FALSE	Shift left when the signal changes to TRUE
	ShiftR	Right shift	Follows data type	FALSE	Shift right when the signal changes to TRUE
	Reset	Reset a bit	Follows data type	FALSE	Reset when the signal changes to TRUE
	In	Input Value	Follows data type	FALSE	Value to insert to the least-significant bit of InOut[ ]
	Size	Number of elements to process	Follows data type	1	Number of elements to use as a shift register in InOut[ ]
Input/Output	InOut[ ] array	Array where the shift object is located	Follows data type	-	Array where the shift object is located
Output	P_CY	Carry flag	Follows data type	-	Value stored in the carry flag

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
ShiftL	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ShiftR	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Reset	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
In	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
InOut[ ] array	√	√	√	√	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Size	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
P_CY	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction shifts Size element in InOut[ ] starting from InOut[0] to the left (toward the most-significant bit) by one bit when ShiftL changes from FALSE to TRUE. The input value In is inserted into the least-significant bit. The most-significant bit shifted out of the array is output to the carry (CY) flag P_CY.

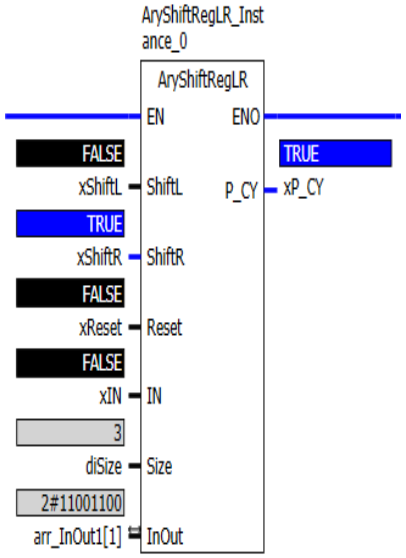
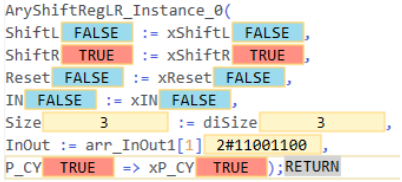
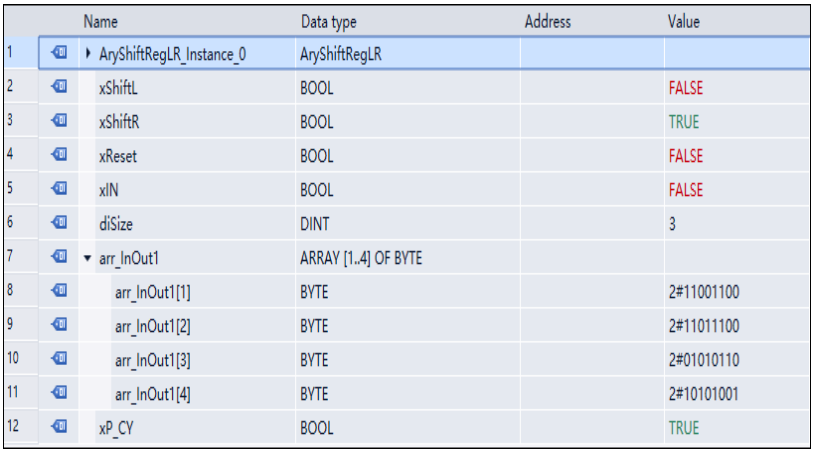
  This instruction shifts Size element in InOut[ ] starting from InOut[0] to the right (toward the least-significant bit) by one bit when ShiftR changes from FALSE to TRUE. The input value In is inserted into the most-significant bit. The least-significant bit shifted out of the array is output to the carry (CY) flag P_CY.

  When Reset is TRUE, CY and all bits of the Size elements starting from InOut[0] are set to FALSE.

- **Program example**

	LD	ST

Define variables	VAR
	AryShiftRegLR_0: AryShiftRegLR;
	xShiftL: BOOL;
	xShiftR: BOOL := 1;
	xReset: BOOL;
	xIn: BOOL;
	xP_CY: BOOL;
	disizel: DINT := 3;
	inoutl:ARRAY[1..4] OF BYTE :=[2#10011001,2#10111001,2#11101101,2#10101101];
	END_VAR	

<table>
<tr><th></th><th>LD</th><th>ST</th></tr>
<tr><td>Program</td><td>
</td><td>
</td></tr>
<tr><td>Operation result</td><td colspan="2">
</td></tr>
</table>

- **Precautions**
  - When Reset is TRUE, no shift is performed even if ShiftL/ShiftR changes from FALSE to TRUE.
  - No shift is performed when both ShiftL and ShiftR change from FALSE to TRUE simultaneously.
  - When the value of Size is 0, InOut[ ] remains unchanged.
  - When the value of Size exceeds the array area of InOut[ ], InOut[ ] remains unchanged.
  - If the dimension of the input array exceeds 3, AryOut[ ] remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.17 Memory Copy

18.2.17.1 Instruction List

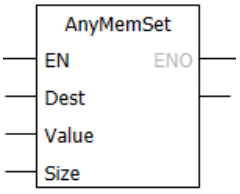
Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.17.2 AnyMemSet	FC	Memory fill	Util	Library version 1.1.0.0 and later
18.2.17.3 AnyMemCpy	FC	Memory copy		Library version 1.1.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.17.4 AnyMemMove	FC	Memory move		Library version 1.1.0.0 and later
18.2.17.5 AnyMemCmp	FC	Memory comparison		Library version 1.2.0.0 and later

18.2.17.2 AnyMemSet

Fills the specified memory.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AnyMemSet	Memory fill	FC		:= AnyMemSet(Dest := , Value := , Size :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Dest	Target variable	Follows data type	-	Target object for memory fill
	Value	Fill value	Follows data type	-	Value to be filled into the memory
	Size	Fill length	Follows data type	16#FFFFFFFF	Number of bytes to fill
Output	AnyMemSet	Execution result	Follows data type	FALSE	The result of the instruction.

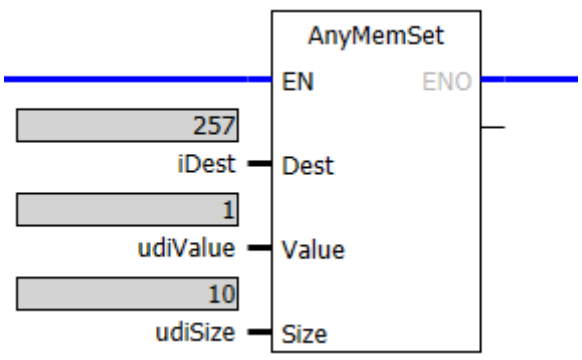
Variable types

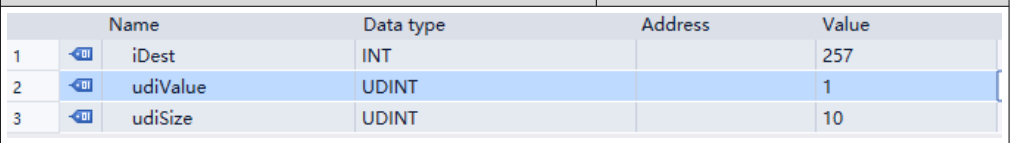
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
Dest	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration, an array, an array element, a structure, or a structure member can also be specified.																				
Value	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
Size	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AnyMemSet	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**

  This instruction fills data into Dest in bytes, with the number of bytes filled being the minimum of the byte size of Dest and the Size parameter.

- **Program example**

Item	LD	ST
Define variables	PROGRAM POU VAR iDest: INT; udiValue: UDINT; udiSize: UDINT; END_VAR	
Program		AnyMemSet(Dest := iDest 257 , Value := udivalue 1 , Size := udiSize 10);

Item	LD	ST
Operation result		

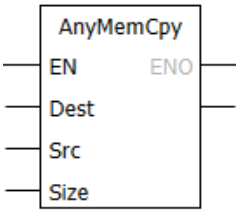
	Name	Data type	Address	Value
1	iDest	INT		257
2	udiValue	UDINT		1
3	udiSize	UDINT		10

- **Precautions**
  - When Dest is an array element, the fill calculates the byte length starting from that element to the end of the array, compares it with Size, and fills the smaller of the two.
  - Dest supports arrays with a maximum dimension of three. If the dimension exceeds three, the function immediately returns TRUE, and Dest remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.17.3 AnyMemCpy

Copies the specified memory.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AnyMemCpy	Memory copy	FC		:= AnyMemCpy(Dest := , Src := , Size :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Dest	Target variable	Follows data type	-	Target object for memory copy
	Src	Source variable	Follows data type	-	Source object to be copied
	Size	Copy length	Follows data type	16#FFFFFFFF	Number of bytes to copy
Output	AnyMemCpy	Execution result	Follows data type	FALSE	The result of the instruction.

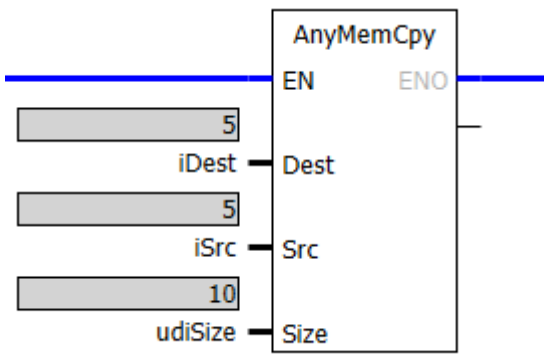
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
Dest	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration, an array, an array element, a structure, or a structure member can also be specified.																				
Src	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration, an array, an array element, a structure, or a structure member can also be specified.																				
Size	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AnyMemCpy	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**

  This instruction copies data from the starting position of Src to Dest in bytes. The number of bytes copied is the minimum of the byte count of Src, the byte count of Dest, and Size.

- **Program example**

Item	LD	ST
Define variables	VAR iDest: INT; iSrc: INT; udiSize: UDINT; END_VAR	
Program		AnyMemCpy(Dest := iDest 5 , Src := iSrc 5 , Size := udiSize 10);

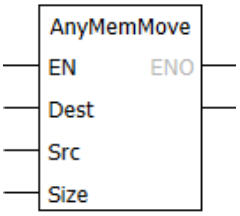
Item	LD	ST
Operation result		

- **Precautions**
  - When Dest or Src is an array element, this instruction calculates the byte length starting from Dest or Src to the end of the array, compares it with Size, and the smaller value is used for copying.
  - The data types of Dest and Src can be different, but the copy result may be incorrect. For example, if Dest is INT and Src is DINT with a value of 65536, Dest will be assigned 0.
  - Dest or Src supports arrays with a maximum dimension of three. If the dimension exceeds three, the function immediately returns TRUE, and Dest remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.17.4 AnyMemMove

Moves the specified memory.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AnyMemMove	Memory move	FC		:= AnyMemMove(Dest := , Src := , Size :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Dest	Target variable	Follows data type	-	Target object for memory move
	Src	Source variable	Follows data type	-	Source object for memory move
	Size	Move length	Follows data type	16#FFFFFFFF	Number of bytes to move
Output	AnyMemMove	Execution result	Follows data type	FALSE	The result of the instruction.

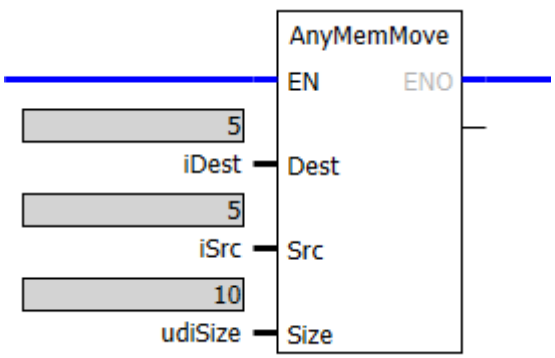
Variable types

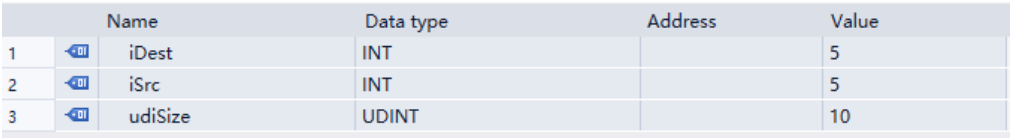
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
Dest	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration, an array, an array element, a structure, or a structure member can also be specified.																				
Src	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration, an array, an array element, a structure, or a structure member can also be specified.																				
Size	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-
AnyMemMove	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**

  This instruction moves data from the starting position of Src to Dest in bytes. The number of bytes moved is the minimum of the byte count of Src, the byte count of Dest, and Size.

- **Program example**

Item	LD	ST
Define variables	VAR iDest: INT; iSrc: INT; udiSize: UDINT; END_VAR	
Program		AnyMemMove(Dest := iDest 5 , Src := iSrc 5 , Size := udiSize 10);

Item	LD	ST
Operation result		

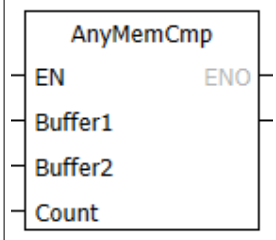
	Name	Data type	Address	Value
1	iDest	INT		5
2	iSrc	INT		5
3	udiSize	UDINT		10

- **Precautions**
  - When Dest or Src is an array element, this instruction calculates the byte length starting from Dest or Src to the end of the array, compares it with Size, and the smaller value is used for moving.
  - The data types of Dest and Src can be different, but the result may be incorrect. For example, if Dest is INT and Src is DINT with a value of 65536, Dest will be assigned 0.
  - Dest or Src supports arrays with a maximum dimension of three. If the dimension exceeds three, the function immediately returns TRUE, and Dest remains unchanged.
  - Size is of fixed type DINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.17.5 AnyMemCmp

Compares the memory of two specified variables.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
AnyMemCmp	Memory comparison	FC		:= AnyMemCmp(Buffer1 := , Buffer2 := , Count :=);

- **Related variables**

Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	Buffer1	Comparison buffer 1	Follows data type	-	Variable to be compared
	Buffer2	Comparison buffer 2	Follows data type	-	Variable to be compared
	Count	Comparison length	Follows data type	0	Number of bytes to compare

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Output	AnyMemCmp	Comparison result	Follows data type	0	Comparison result. • A negative return value indicates that the first unequal byte in the two buffers has a smaller value in Buffer1 than in Buffer2. • A return value of 0 indicates that the values in the two buffers are completely equal within the comparison range. • A positive return value indicates that the first unequal byte in the two buffers has a greater value in Buffer1 than in Buffer2. Note: Only the sign of the return value is meaningful; the specific return value is meaningless.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
Buffer1	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration, an array, an array element, a structure, or a structure member can also be specified.																				
Buffer2	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√	√
	An enumeration, an array, an array element, a structure, or a structure member can also be specified.																				
Count	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
AnyMemCmp	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

- **Function**

  This instruction compares whether the memory of Count bytes starting from the addresses of Buffer1 and Buffer2 are identical.

- **Program example**

Item	LD	ST

Define variables	VAR
	aiBuf1 : ARRAY[0..3] OF INT := [1,2,3,4];
	aiBuf2 : ARRAY[0..3] OF INT := [1,2,3,5];
	Count : UDINT := 8;
	Ret1 : DINT;
	END_VAR	

Program	AnyMemCmp EN ENO aiBuf1 Buffer1 aiBuf2 Buffer2 8 Count -1 Ret1	Ret1 -1 := AnyMemCmp(Buffer1 := aiBuf1, Buffer2 := aiBuf2, Count := Count 8); RETURN
Operation result	Name / Type / Address / Value aiBuf1 / ARRAY [0..3] OF INT aiBuf2 / ARRAY [0..3] OF INT Count / UDINT / 8 Ret1 / DINT / -1	

- **Precautions**
  - When Buffer1 or Buffer2 is an array element, this instruction calculates the byte length starting from Buffer1 or Buffer2 to the end of the array, compares it with Size, and the smaller value is used for comparison.
  - The data types of Buffer1 and Buffer2 can be different. The comparison is performed byte-by-byte and is independent of data types.
  - Buffer1 or Buffer2 supports arrays with a maximum dimension of three. If the dimension exceeds three, the function performs a 0-byte comparison and returns 0 directly.
  - Count is of fixed type UDINT. If another data type is passed to the function block, it will be handled according to implicit type conversion rules.

18.2.18 Recipe Library

18.2.18.1 RecipeManager

18.2.18.1.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.18.1.3 CreateRecipe	Method	Create a recipe	RecipeManagement	Library version 1.0.0.0 and later
18.2.18.1.4 CreateRecipeNoSave	Method	Create a recipe without saving files		Library version 1.0.0.0 and later
18.2.18.1.5 DeleteAllRecipe	Method	Delete all recipes		Library version 1.2.0.0 and later
18.2.18.1.6 DeleteRecipe	Method	Delete a recipe		Library version 1.0.0.0 and later
18.2.18.1.7 DeleteRecipeFile	Method	Delete a recipe file		Library version 1.0.0.0 and later
18.2.18.1.8 LoadAllRecipes	Method	Load all recipe files		Library version 1.0.0.0 and later
18.2.18.1.9 LoadAndWriteRecipe	Method	Load a recipe file and write it into a variable		Library version 1.0.0.0 and later
18.2.18.1.10 LoadFromAndWriteRecipe	Method	Load a recipe file from the specified file and write it into a variable		Library version 1.0.0.0 and later
18.2.18.1.11 LoadRecipe	Method	Load a recipe file		Library version 1.0.0.0 and later
18.2.18.1.12 ReadAndSaveAs	Method	Read PLC variables and save as a recipe file		Library version 1.0.0.0 and later
18.2.18.1.13 ReadAndSaveRecipe	Method	Read PLC variables into a recipe and save the recipe file		Library version 1.0.0.0 and later
18.2.18.1.14 ReadAndSaveRecipeAs	Method	Read PLC variables into a recipe and save as a recipe file		Library version 1.0.0.0 and later
18.2.18.1.15 SaveRecipe	Method	Save a recipe file		Library version 1.0.0.0 and later
18.2.18.1.16 ReadRecipe	Method	Read PLC variables into a recipe		Library version 1.0.0.0 and later
18.2.18.1.17 WriteRecipe	Method	Write a recipe		Library version 1.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.18.1.18 ReloadRecipes	Method	Reload recipes		Library version 1.0.0.0 and later
18.2.18.1.19 GetRecipeCount	Method	Get the number of recipes associated to a specified recipe		Library version 1.0.0.0 and later
18.2.18.1.20 GetRecipeNames	Method	Get the recipe names associated to a specified recipe		Library version 1.0.0.0 and later
18.2.18.1.21 GetRecipeValues	Method	Get the variable values of a specified recipe		Library version 1.0.0.0 and later
18.2.18.1.22 GetRecipeVariableNames	Method	Get the variable names of a specified recipe		Library version 1.0.0.0 and later
18.2.18.1.23 SetRecipeValues	Method	Set the variable values of a specified recipe		Library version 1.0.0.0 and later
18.2.18.1.24 GetRecipeNamesV2	Method	Get the recipe names associated to a specified recipe		Library version 1.1.0.0 and later
18.2.18.1.25 GetRecipeValuesV2	Method	Get the variable values of a specified recipe		Library version 1.1.0.0 and later
18.2.18.1.26 GetRecipeVariableNames V2	Method	Get the variable names of a specified recipe		Library version 1.1.0.0 and later
18.2.18.1.27 SetRecipeValuesV2	Method	Set the variable values of a specified recipe		Library version 1.1.0.0 and later
18.2.18.1.28 GetLastError	Method	Get the latest fault code		Library version 1.0.0.0 and later
18.2.18.1.29 ResetLastError	Method	Reset fault codes		Library version 1.0.0.0 and later
18.2.18.1.30 SetStoragePath	Method	Set storage path		Library version 1.0.0.0 and later
18.2.18.1.31 CreateRecipeV3	FB	Create a recipe		Library version 2.0.0.0 and later
18.2.18.1.32 DeleteALLRecipeV3	FB	Deletes all recipes under a recipe definition		Library version 2.0.0.0 and later
18.2.18.1.33 DeleteRecipeV3	FB	Delete a specified recipe under a recipe definition		Library version 2.0.0.0 and later
18.2.18.1.34 GetRecipeDefinitionInfoV3	FB	Get recipe definition information		Library version 2.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.18.1.35 GetRecipeInfoV3	FB	Get recipe information		Library version 2.0.0.0 and later
18.2.18.1.36 LoadRecipeV3	FB	Load a recipe		Library version 2.0.0.0 and later
18.2.18.1.37 ReadRecipeV3	FB	Read PLC values into a recipe		Library version 2.0.0.0 and later
18.2.18.1.38 RenameRecipeV3	FB	Rename a recipe		Library version 2.0.0.0 and later
18.2.18.1.39 SaveRecipeV3	FB	Save a recipe file		Library version 2.0.0.0 and later
18.2.18.1.40 SetRecipeValuesV3	FB	Set the variable values of a specified recipe		Library version 2.0.0.0 and later
18.2.18.1.41 WriteRecipeV3	FB	Write the recipe into the PLC		Library version 2.0.0.0 and later
18.2.18.1.42 SetStoragePathV3	FB	Set the path to store or load the recipe file		Library version 2.0.0.0 and later

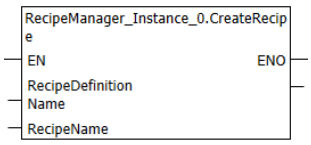
18.2.18.1.2 Overview

1. The basic configuration of the recipe can be defined under the "Recipe" node in the project tree. For details, see the 10.3 Recipe chapter.
2. The storage location and format of the recipe can be defined in the RecipeManager object, under which one or more recipe definitions can be added. Each recipe definition includes multiple recipe names corresponding to all variables within it, with each recipe name containing specific variable values.
3. All instructions in the recipe management library are methods of the RecipeManager function bloc and must be used within the same RecipeManager instance to manage recipes programmatically.
4. It is recommended to use edge-triggered signals for the enable input EN. This can be achieved by inserting an edge detection contact at the connection point or using the 18.1.1.4 PLS or 18.1.1.2 R_TRIG rising edge output instruction in combination.

18.2.18.1.3 CreateRecipe

Creates a new recipe based on the specified recipe definition. The current PLC variable values are read into the new recipe and saved as a recipe file with the default name. The default recipe file name format is <recipe definition>.<recipe>.txtrecipe.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CreateRecipe	Create a recipe	Method		:= RecipeManager_Instance_0.CreateRecipe(RecipeDefinitionName := , RecipeName :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name
	RecipeName	Recipe name	Follows data type	-	Recipe name
Output	CreateRecipe	Return value	Follows data type	-	Refer to Fault Code.

  Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
RecipeName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
CreateRecipe	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction creates a new recipe based on the specified recipe definition. The current PLC variable values are read into the new recipe and saved as a recipe file with the default name.

- **Program example**

  ST:

	Name	Data type	Address	Value	Prepare value	Comment
1	RecipeManager_Instance_0	RecipeManager				
2	R_TRIG_Instance_0	R_TRIG				
3	EN_Manual0	BOOL		TRUE		
4	RecipeDefinition	STRING		'Recipe_1'		
5	RecipeName	STRING		'test1'		
6	result0	DWORD		0		

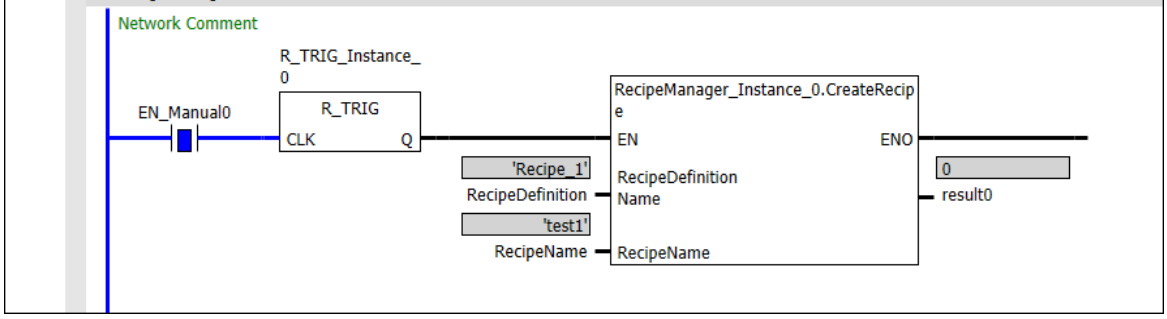
```
R_TRIG_Instance_0(
CLK := EN_Manual0,
Q => );

IF R_TRIG_Instance_0.Q THEN
    result0 := RecipeManager_Instance_0.CreateRecipe(
    RecipeDefinitionName := RecipeDefinition 'Recipe_1',
    RecipeName := RecipeName 'test1');
END_IF

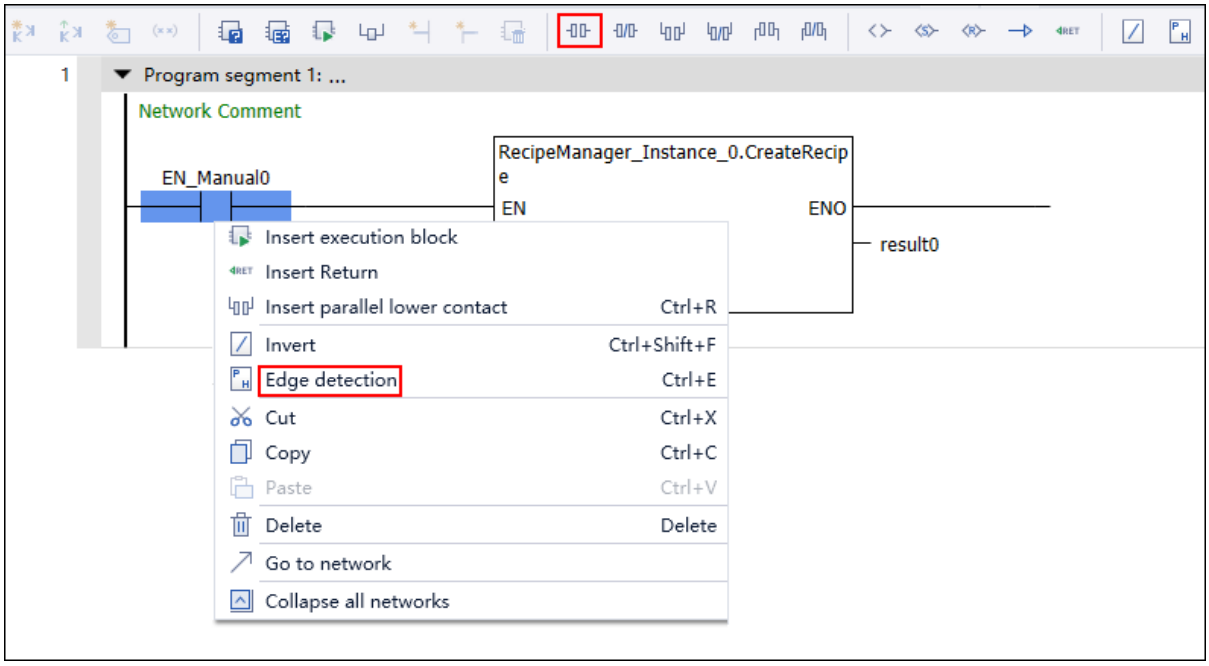
RETURN
```

LD:

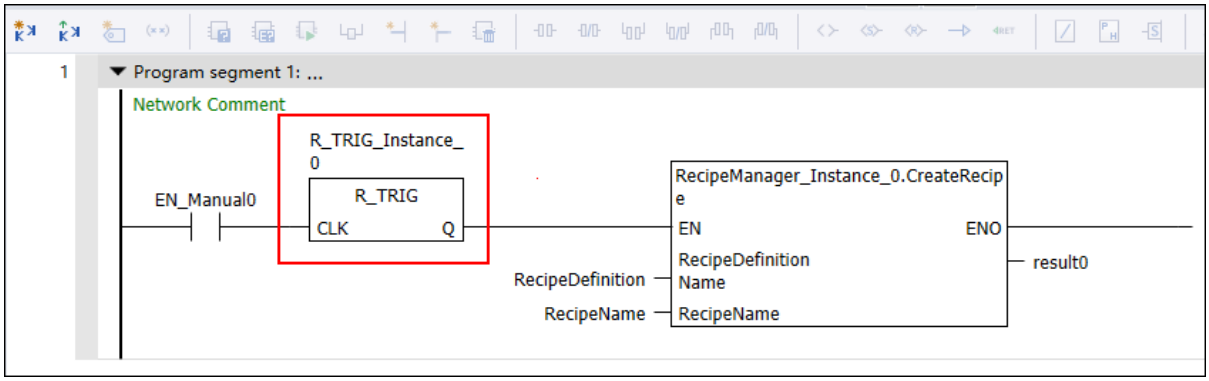
	Name	Data type	Address	Value	Prepare value	Comment
1	RecipeManager_Instance_0	RecipeManager				
2	R_TRIG_Instance_0	R_TRIG				
3	EN_Manual0	BOOL		TRUE		
4	RecipeDefinition	STRING		'Recipe_1'		
5	RecipeName	STRING		'test1'		
6	result0	DWORD		0		



- **Precautions**
  - It is recommended to use edge-triggered signals for the enable input EN; otherwise, this method will return a fault code.
  - Take the following ladder logic (LD) diagram as an example:
    - Insert a contact at the connection point, and select "Edge detection" in the right-click menu.



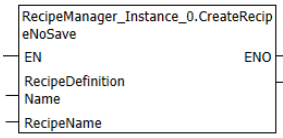
- Use the 18.1.1.2 R_TRIGrising edge output instruction in combination.



18.2.18.1.4 CreateRecipeNoSave

Creates a new recipe based on the specified recipe definition, and reads the current PLC variable values into the new recipe.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CreateRecipeNoSave	Create a recipe without saving files	Method		:= RecipeManager_Instance_0.CreateRecipeNoSave(RecipeDefinitionName := , RecipeName :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name
	RecipeName	Recipe name	Follows data type	-	Recipe name
Output	CreateRecipeNoSave	Return value	Follows data type	-	Refer to Fault Code.

Data Type

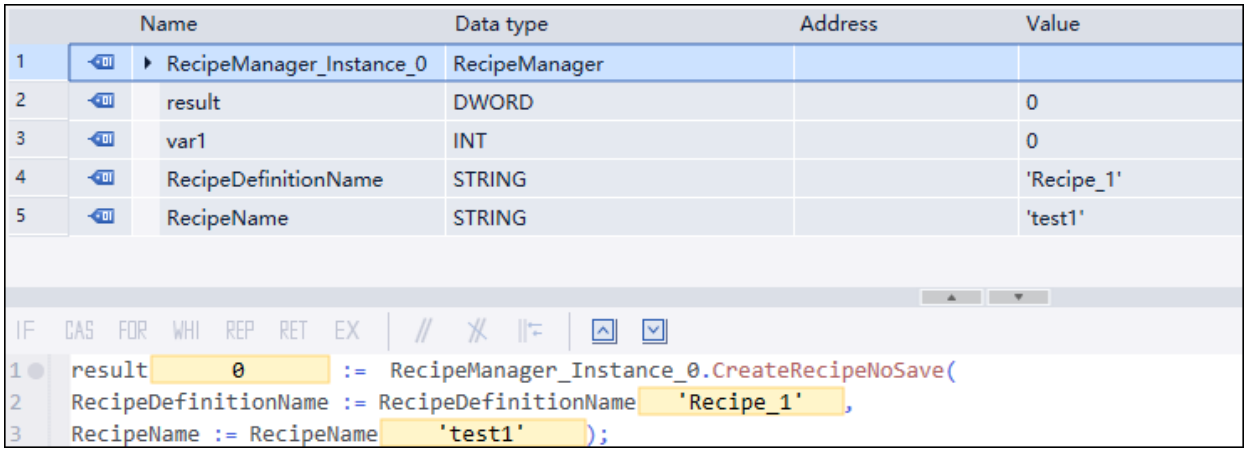
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
RecipeName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
CreateRecipeNoSave	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

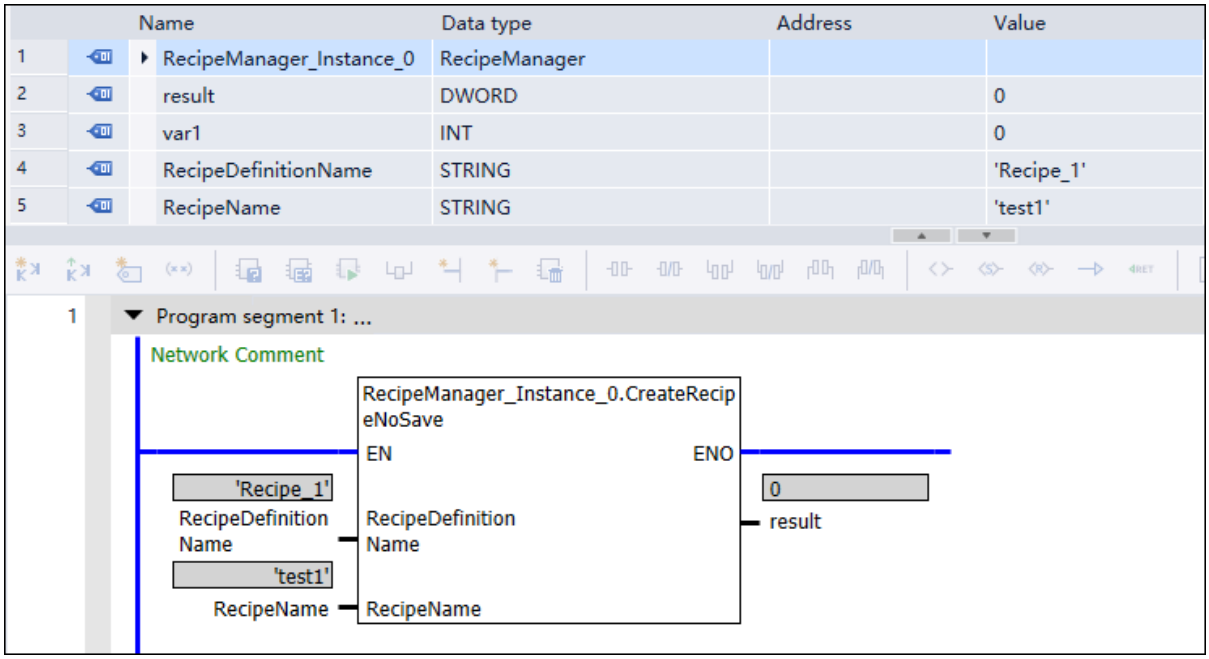
  This instruction creates a new recipe based on the specified recipe definition, and reads the current PLC variable values into the new recipe.

- **Program example**

  ST:



  LD:



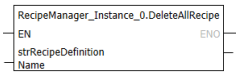
- **Precautions**

  It is recommended to use edge-triggered signals for the enable input EN; otherwise, this method will return a fault code.

18.2.18.1.5 DeleteAllRecipe

Deletes all recipes from the specified recipe definition.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DeleteAllRecipe	Delete all recipes	Method		:= RecipeManager_Instance_0.DeleteAllRecipe(strRecipeDefinitionName :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	strRecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name
Output	DeleteAllRecipe	Return value	Follows data type	-	Refer to Fault Code.

  Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
strRecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
DeleteAllRecipe	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction deletes all recipes from the specified recipe definition.

- **Program example**

  Call the CreateRecipe instruction to create a new recipe. Then, call the GetRecipeCount and GetRecipeNamesV2 instructions to get the recipe count and recipe names. Finally, execute the DeleteAllRecipe instruction and compare the recipe count and recipe names before and after deletion.

  ST:

  Before executing the DeleteAllRecipe instruction, the recipe count is 3, and the recipe names associated with the recipe definition are 'recipe1', 'recipe2', and 'recipe3'.

	Name	Data type	Address	Value
1	RecipeManager_Instance_0	RecipeManager		
2	RecipeDefinitionName	STRING		'Recipe_1'
3	RecipeName	STRING		'recipe3'
4	result0	DWORD		0
5	result1	DWORD		0
6	result2	DWORD		3
7	result3	DWORD		0
8	EN_Manual0	BOOL		TRUE
9	EN_Manual1	BOOL		FALSE
10	EN_Manual2	BOOL		TRUE
11	R_TRIG_Instance_0	R_TRIG		
12	R_TRIG_Instance_1	R_TRIG		
13	aString	ARRAY [0..5] OF STRING		
14	aString[0]	STRING		'recipe1'
15	aString[1]	STRING		'recipe3'
16	aString[2]	STRING		'recipe2'
17	aString[3]	STRING		''
18	aString[4]	STRING		''
19	aString[5]	STRING		''

```
R_TRIG_Instance_0(
CLK TRUE := EN_Manual0 TRUE,
Q => );

IF R_TRIG_Instance_0.Q FALSE THEN
    result0 0 := RecipeManager_Instance_0.CreateRecipe(
    RecipeDefinitionName := RecipeDefinitionName 'Recipe_1' ,
    RecipeName := RecipeName 'recipe3' );
END_IF

R_TRIG_Instance_1(
CLK FALSE := EN_Manual1 FALSE,
Q => );

IF R_TRIG_Instance_1.Q FALSE THEN
    result1 0 := RecipeManager_Instance_0.DeleteAllRecipe(
    strRecipeDefinitionName := RecipeDefinitionName 'Recipe_1' );
END_IF

IF EN_Manual2 TRUE THEN
    result2 3 := RecipeManager_Instance_0.GetRecipeCount(
    RecipeDefinitionName := RecipeDefinitionName 'Recipe_1' );

    result3 0 := RecipeManager_Instance_0.GetRecipeNamesV2(
    RecipeDefinitionName := RecipeDefinitionName 'Recipe_1' ,
    aStrings := aString[0] 'recipe1' ,
    iSize := 3,
    iStartIndex := 0);
END_IFRETURN
```

After executing the DeleteAllRecipe instruction, the recipe count is 0, and the recipe names associated with the recipe definition are empty.

	Name	Data type	Address	Value
1	RecipeManager_Instance_0	RecipeManager		
2	RecipeDefinitionName	STRING		'Recipe_1'
3	RecipeName	STRING		'recipe3'
4	result0	DWORD		0
5	result1	DWORD		0
6	result2	DWORD		0
7	result3	DWORD		0
8	EN_Manual0	BOOL		TRUE
9	EN_Manual1	BOOL		TRUE
10	EN_Manual2	BOOL		TRUE
11	R_TRIG_Instance_0	R_TRIG		
12	R_TRIG_Instance_1	R_TRIG		
13	aString	ARRAY [0..5] OF STRING		
14	aString[0]	STRING		''
15	aString[1]	STRING		''
16	aString[2]	STRING		''
17	aString[3]	STRING		''
18	aString[4]	STRING		''
19	aString[5]	STRING		''

```
R_TRIG_Instance_0(
CLK  TRUE  := EN_Manual0  TRUE ,
Q => );

IF R_TRIG_Instance_0.Q  FALSE  THEN
    result0      0       :=  RecipeManager_Instance_0.CreateRecipe(
    RecipeDefinitionName := RecipeDefinitionName    'Recipe_1'    ,
    RecipeName := RecipeName    'recipe3'    );
END_IF

R_TRIG_Instance_1(
CLK  TRUE  := EN_Manual1  TRUE ,
Q => );

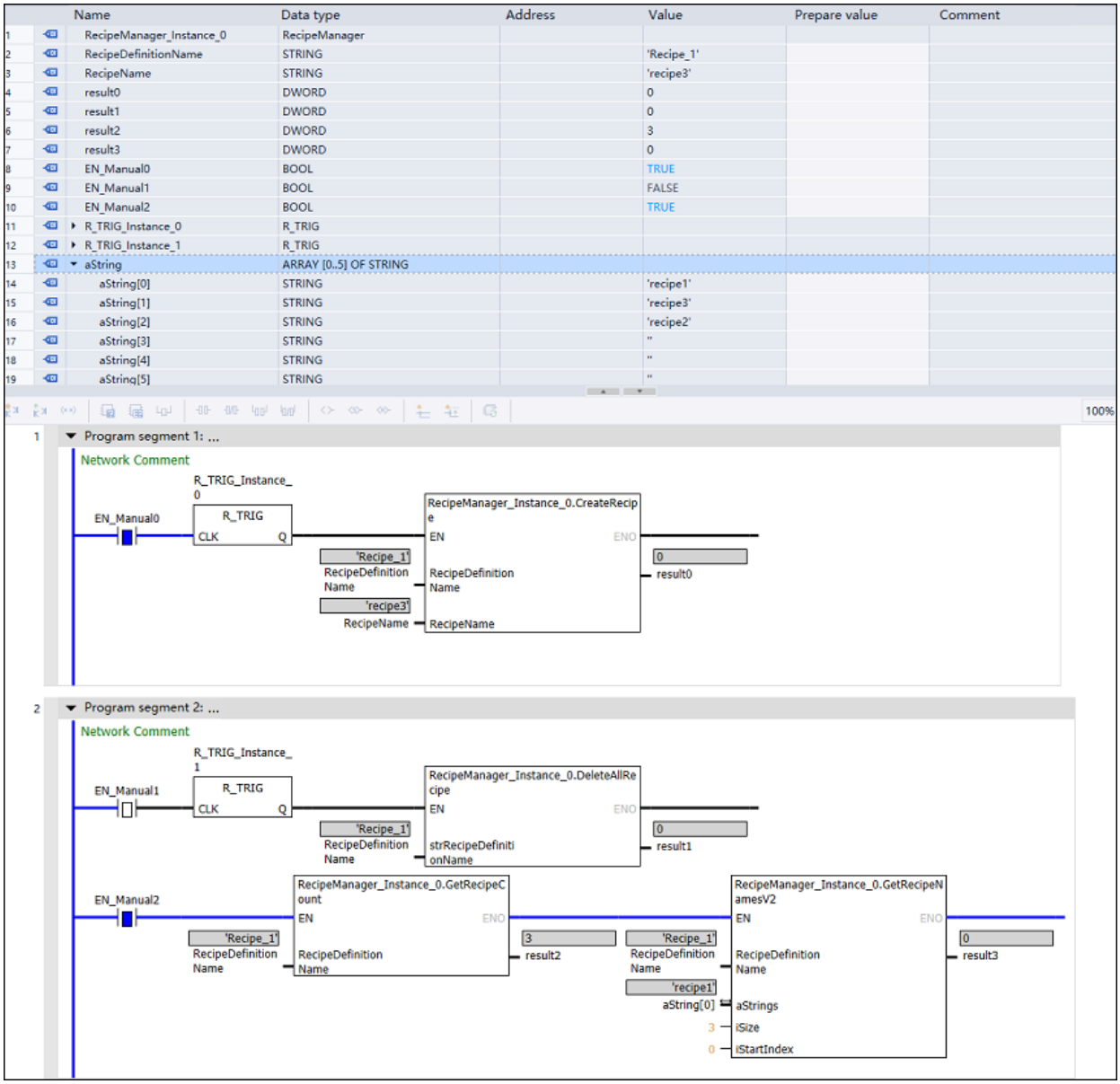
IF R_TRIG_Instance_1.Q  FALSE  THEN
    result1      0       :=  RecipeManager_Instance_0.DeleteAllRecipe(
    strRecipeDefinitionName := RecipeDefinitionName    'Recipe_1'    );
END_IF

IF EN_Manual2  TRUE  THEN
    result2      0       :=  RecipeManager_Instance_0.GetRecipeCount(
    RecipeDefinitionName := RecipeDefinitionName    'Recipe_1'    );

    result3      0       :=  RecipeManager_Instance_0.GetRecipeNamesV2(
    RecipeDefinitionName := RecipeDefinitionName    'Recipe_1'    ,
    aStrings := aString[0]       ''       ,
    iSize := 3,
    iStartIndex := 0);
END_IFRETURN
```

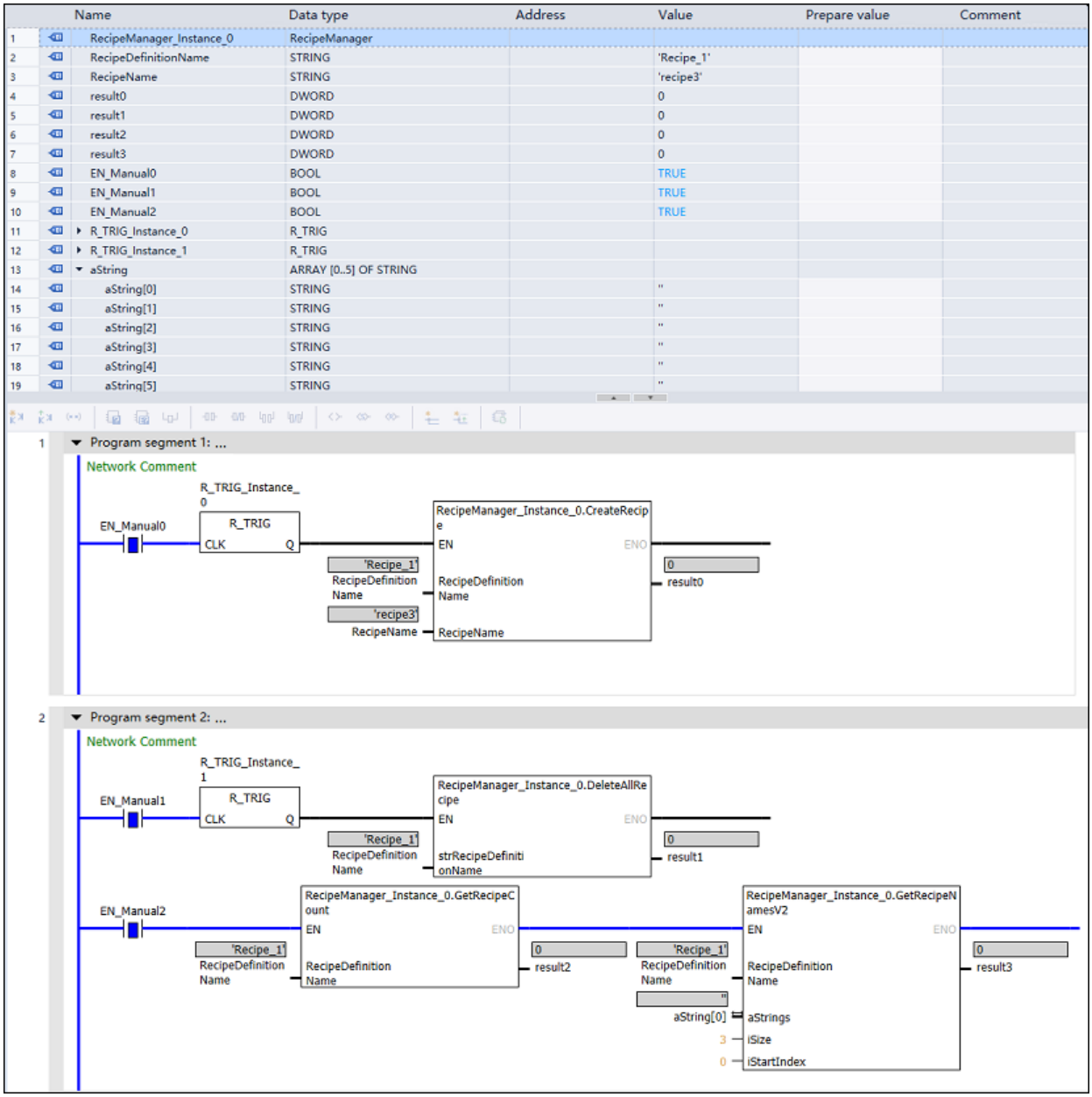
LD:

Before executing the DeleteAllRecipe instruction, the recipe count is 3, and the recipe names associated with the recipe definition are 'recipe1', 'recipe2', and 'recipe3'.



	Name	Data type	Address	Value	Prepare value	Comment
1	RecipeManager_Instance_0	RecipeManager				
2	RecipeDefinitionName	STRING		'Recipe_1'		
3	RecipeName	STRING		'recipe3'		
4	result0	DWORD		0		
5	result1	DWORD		0		
6	result2	DWORD		3		
7	result3	DWORD		0		
8	EN_Manual0	BOOL		TRUE		
9	EN_Manual1	BOOL		FALSE		
10	EN_Manual2	BOOL		TRUE		
11	R_TRIG_Instance_0	R_TRIG				
12	R_TRIG_Instance_1	R_TRIG				
13	aString	ARRAY [0..5] OF STRING				
14	aString[0]	STRING		'recipe1'		
15	aString[1]	STRING		'recipe3'		
16	aString[2]	STRING		'recipe2'		
17	aString[3]	STRING		''		
18	aString[4]	STRING		''		
19	aString[5]	STRING		''		

After executing the DeleteAllRecipe instruction, the recipe count is 0, and the recipe names associated with the recipe definition are empty.



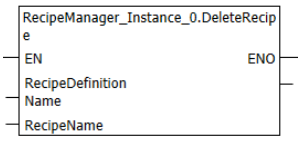
- **Precautions**

  It is recommended to use edge-triggered signals for the enable input EN; otherwise, this method will return a fault code.

18.2.18.1.6 DeleteRecipe

Deletes a recipe from the specified recipe definition.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DeleteRecipe	Delete a recipe	Method		:= RecipeManager_Instance_0.DeleteRecipe(RecipeDefinitionName := , RecipeName :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name
	RecipeName	Recipe name	Follows data type	-	Recipe name
Output	DeleteRecipe	Return value	Follows data type	-	Refer to Fault Code.

Data Type

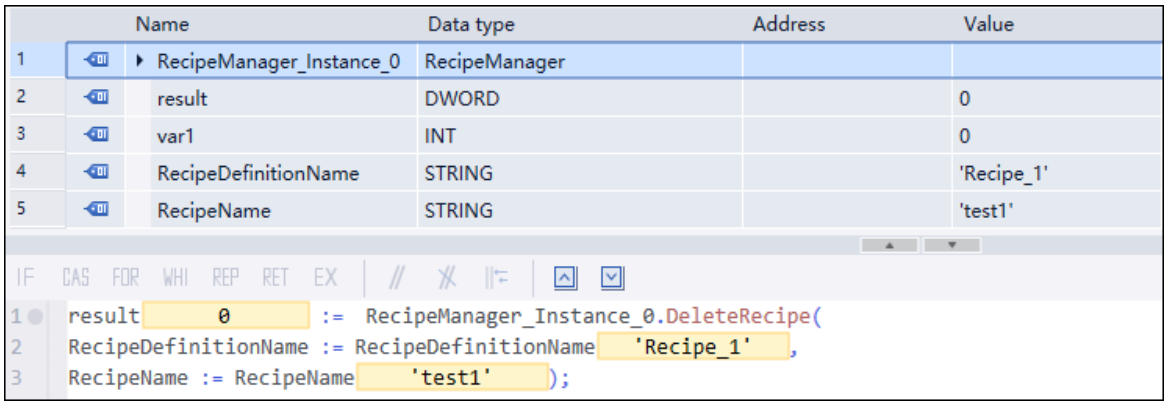
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
RecipeName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
DeleteRecipe	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

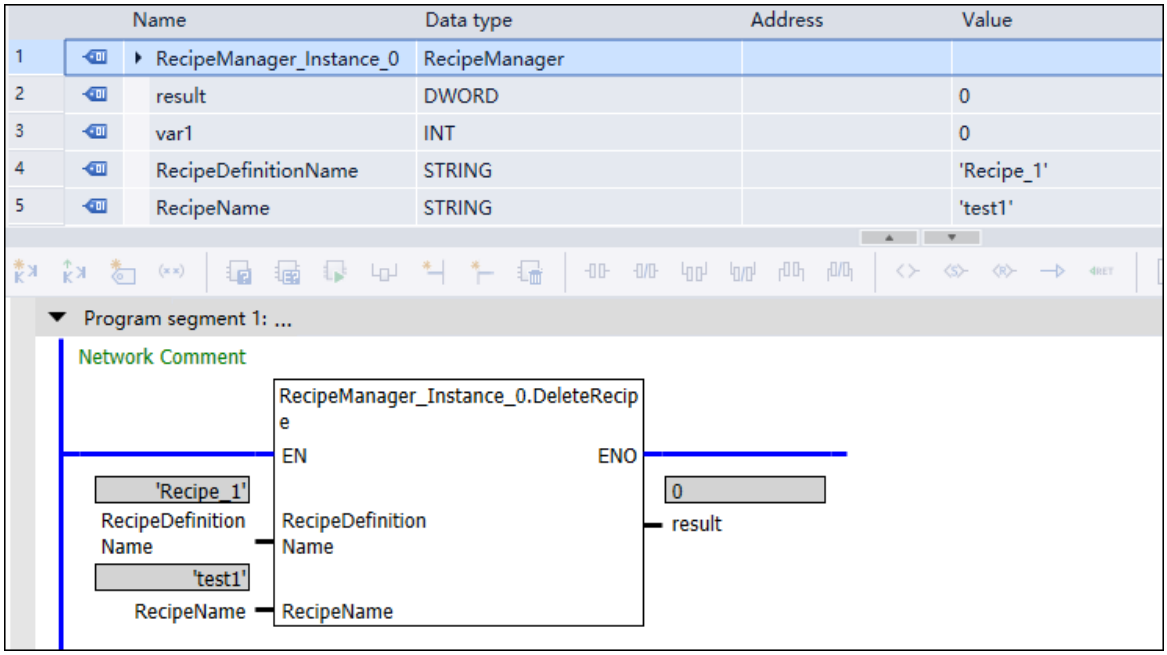
  This instruction deletes a recipe from the specified recipe definition.

- **Program example**

  ST:



  LD:



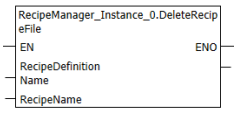
- **Precautions**

  It is recommended to use edge-triggered signals for the enable input EN; otherwise, this method will return a fault code.

18.2.18.1.7 DeleteRecipeFile

Deletes the recipe file corresponding to a recipe. The recipe file must be saved with the default filename format <recipe definition>.<recipe>.txtrecipe.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DeleteRecipeFile	Delete a recipe file	Method		:= RecipeManager_Instance_0.DeleteRecipeFile(RecipeDefinitionName := , RecipeName :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name
	RecipeName	Recipe name	Follows data type	-	Recipe name

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	DeleteRecipeFile	Return value	Follows data type	-	Refer to Fault Code.

Data Type

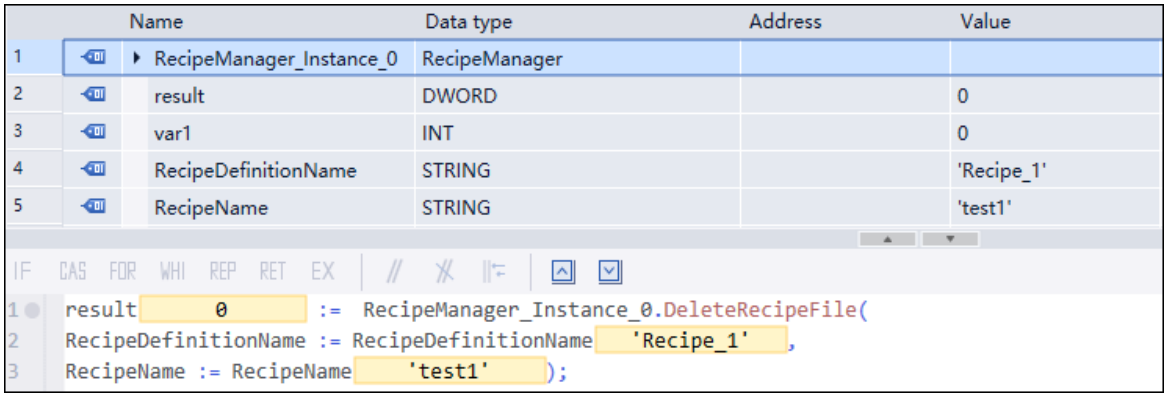
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
RecipeName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
DeleteRecipeFile	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

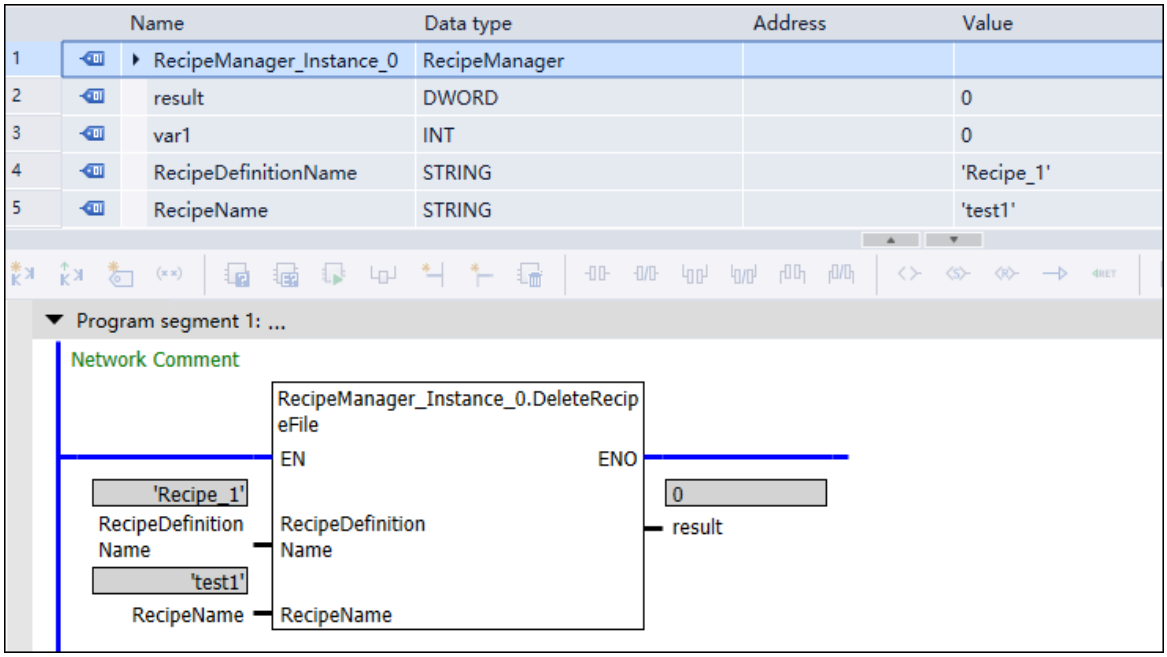
  This instruction deletes the recipe file corresponding to a recipe, which must be saved with the default filename format.

- **Program example**

  ST:



  LD:



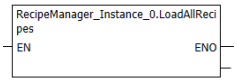
- **Precautions**

  It is recommended to use edge-triggered signals for the enable input EN; otherwise, this method will return a fault code.

18.2.18.1.8 LoadAllRecipes

Loads the recipe files into a recipe. The recipe files must be saved with the default filename format <recipe definition>.<recipe>.txtrecipe.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LoadAllRecipes	Load all recipe files	Method		:= RecipeManager_Instance_0.LoadAllRecipes();

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	LoadAllRecipes	Return value	Follows data type	-	Refer to Fault Code.

  Data Type

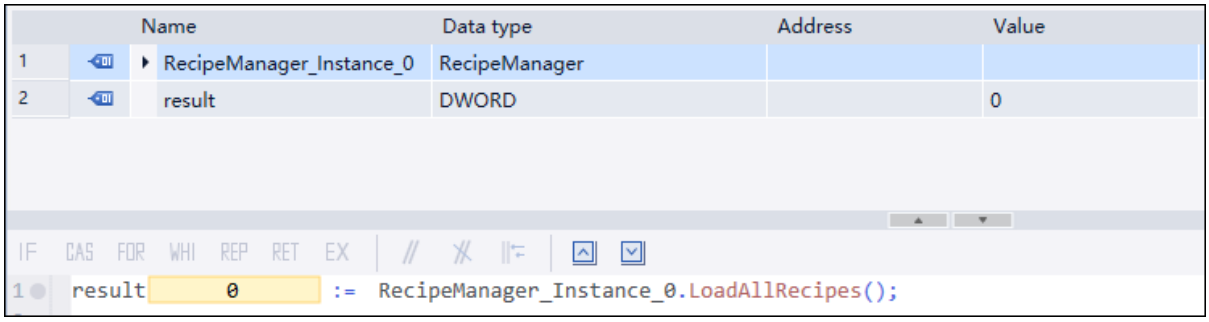
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
LoadAll Recipes	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

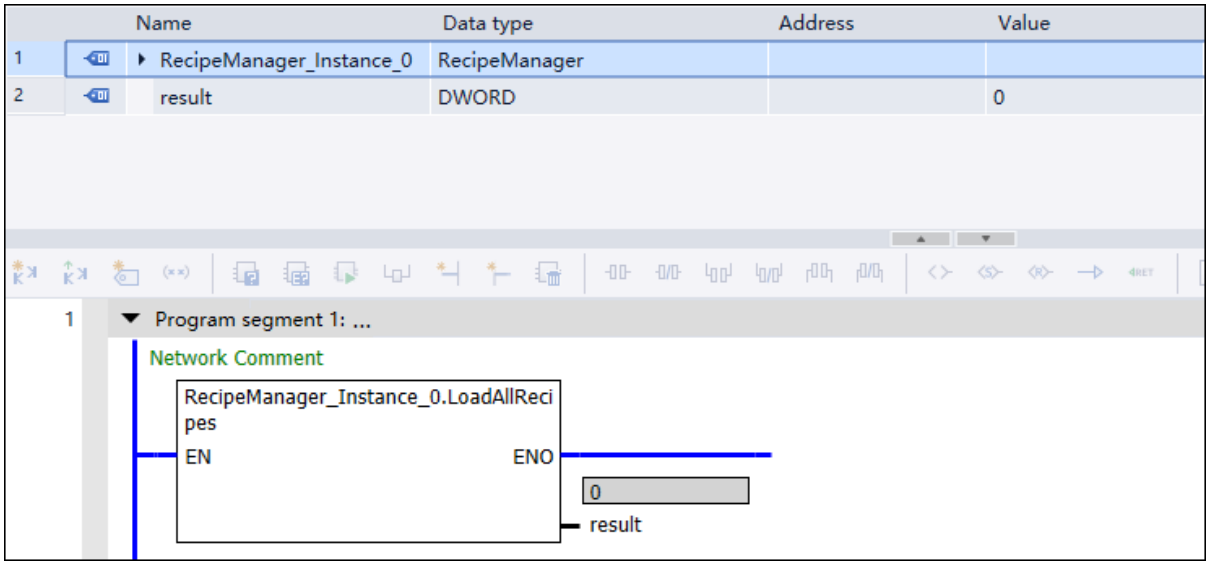
  This instruction loads all recipe files into a recipe.

- **Program example**

  ST:



  LD:



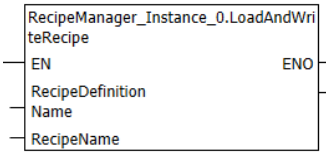
- **Precautions**

  It is recommended to use edge-triggered signals for the enable input EN.

18.2.18.1.9 LoadAndWriteRecipe

Loads a recipe file based on the specified recipe definition and recipe name, and writes the recipe values into the current PLC variables. The recipe file must be saved with the default filename format <recipe definition>.<recipe>.txtrecipe.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LoadAndWriteRecipe	Load a recipe file and write it into a variable	Method		:= RecipeManager_Instance_0.LoadAndWriteRecipe(RecipeDefinitionName := , RecipeName :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name
	RecipeName	Recipe name	Follows data type	-	Recipe name
Output	LoadAndWriteRecipe	Return value	Follows data type	-	Refer to Fault Code.

  Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
RecipeName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
LoadAndWriteRecipe	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction loads the corresponding recipe file based on the specified recipe definition and recipe name.

- **Precautions**

  If entries in the recipe file are not included in the corresponding recipe, these entries will not be loaded and written to the current PLC variables.

- **Program example**

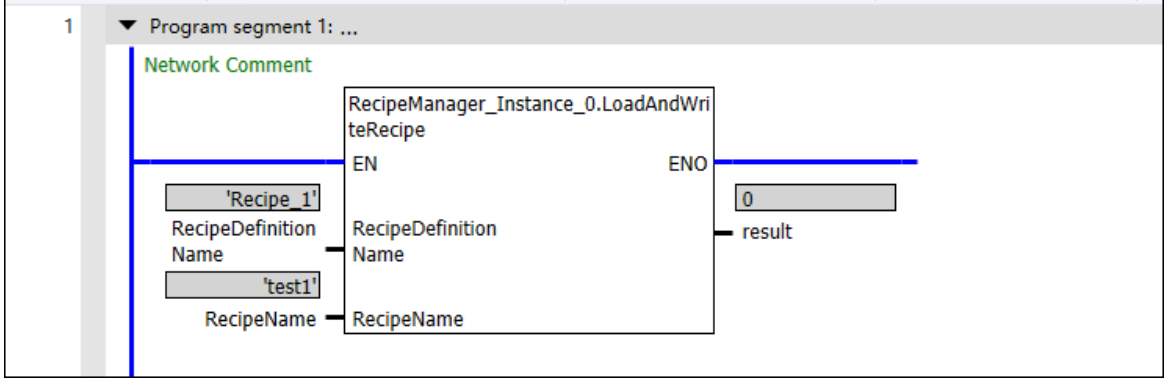
  ST:

	Name	Data type	Address	Value
1	RecipeManager_Instance_0	RecipeManager		
2	result	DWORD		0
3	var1	INT		0
4	RecipeDefinitionName	STRING		'Recipe_1'
5	RecipeName	STRING		'test1'

```
result := RecipeManager_Instance_0.LoadAndWriteRecipe(
RecipeDefinitionName := RecipeDefinitionName,
RecipeName := RecipeName);
```

LD:

	Name	Data type	Address	Value
1	RecipeManager_Instance_0	RecipeManager		
2	result	DWORD		0
3	var1	INT		0
4	RecipeDefinitionName	STRING		'Recipe_1'
5	RecipeName	STRING		'test1'



- **Precautions**

  It is recommended to use edge-triggered signals for the enable input EN.

18.2.18.1.10 LoadFromAndWriteRecipe

Loads a recipe file to the specified recipe and writes the recipe values to the current PLC variables. The default recipe file format is <recipe definition>.<recipe>.txtrecipe.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LoadFromAndWriteRecipe	Load a recipe file from the specified file and write it into a variable	Method	RecipeManager_Instance_0.LoadFromAndWriteRecipe EN ENO RecipeDefinitionName RecipeName FileName	:= RecipeManager_Instance_0.LoadFromAndWriteRecipe(RecipeDefinitionName := , RecipeName := , FileName :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name
	RecipeName	Recipe name	Follows data type	-	Recipe name
	FileName	File name	Follows data type	-	File name
Output	LoadFromAndWriteRecipe	Return value	Follows data type	-	Refer to Fault Code.

Data Type

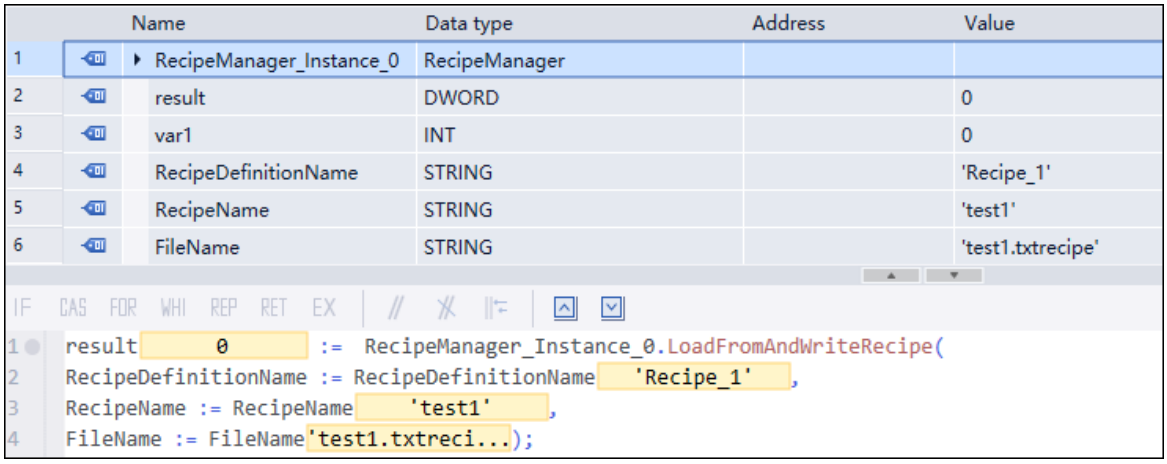
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
RecipeName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
FileName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
LoadFromAndWriteRecipe	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

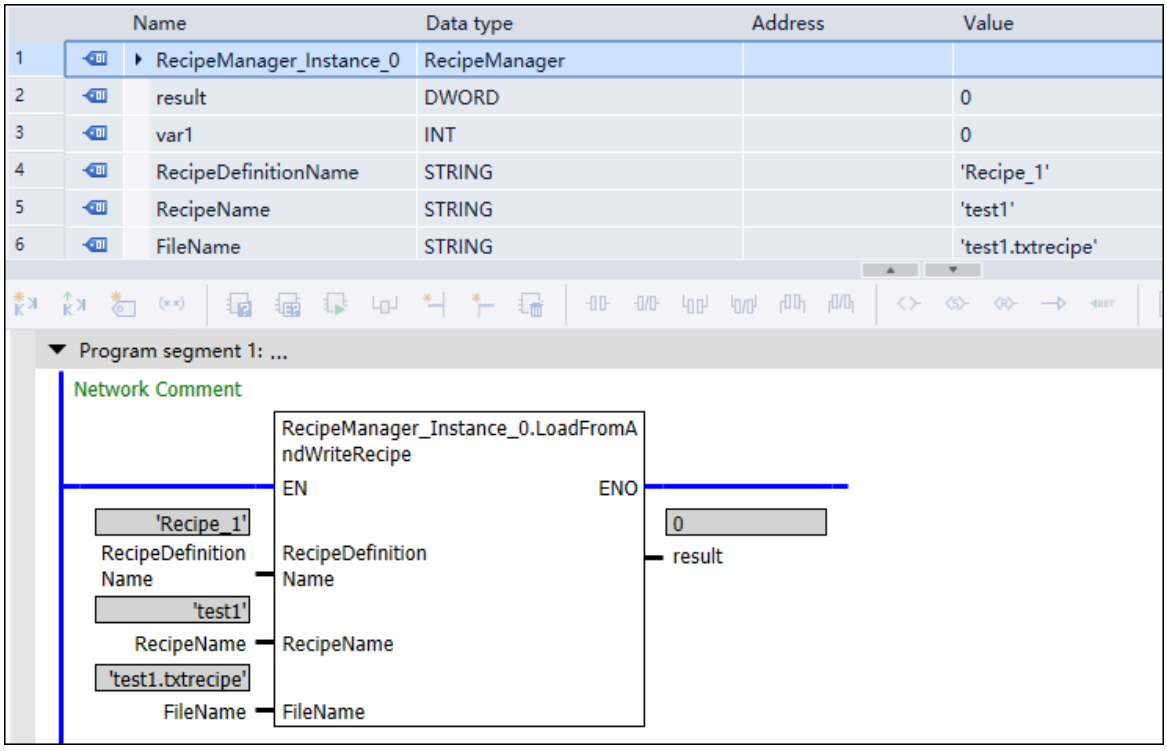
  This instruction loads a recipe file to the specified recipe and writes the recipe values to the current PLC variables.

- **Program example**

  ST:



LD:



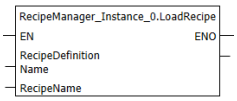
- **Precautions**

  It is recommended to use edge-triggered signals for the enable input EN.

18.2.18.1.11 LoadRecipe

Loads a recipe from the specified recipe file. The recipe file must be saved with the default filename format <recipe definition>.<recipe>.txtrecipe.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LoadRecipe	Load a recipe file	Method		:= RecipeManager_Instance_0.LoadRecipe(RecipeDefinitionName := , RecipeName :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name
	RecipeName	Recipe name	Follows data type	-	Recipe name
Output	LoadRecipe	Return value	Follows data type	-	Refer to Fault Code.

  Data Type

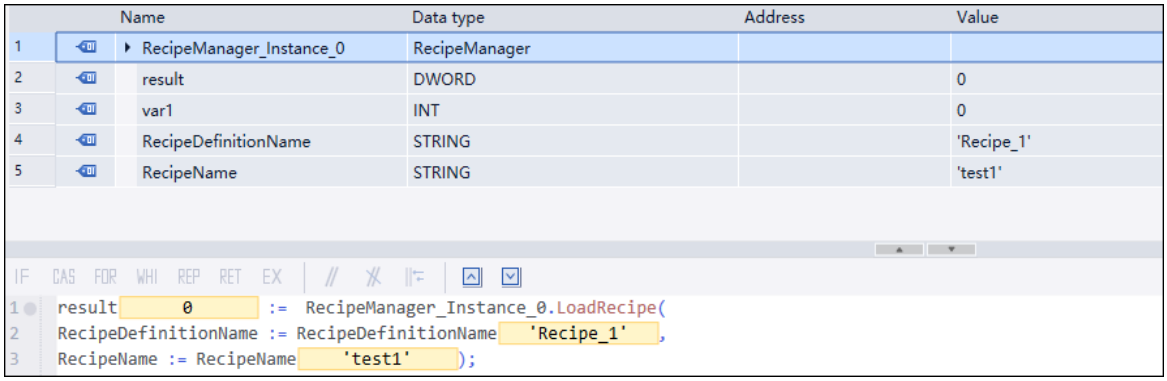
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
RecipeName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
LoadRecipe	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

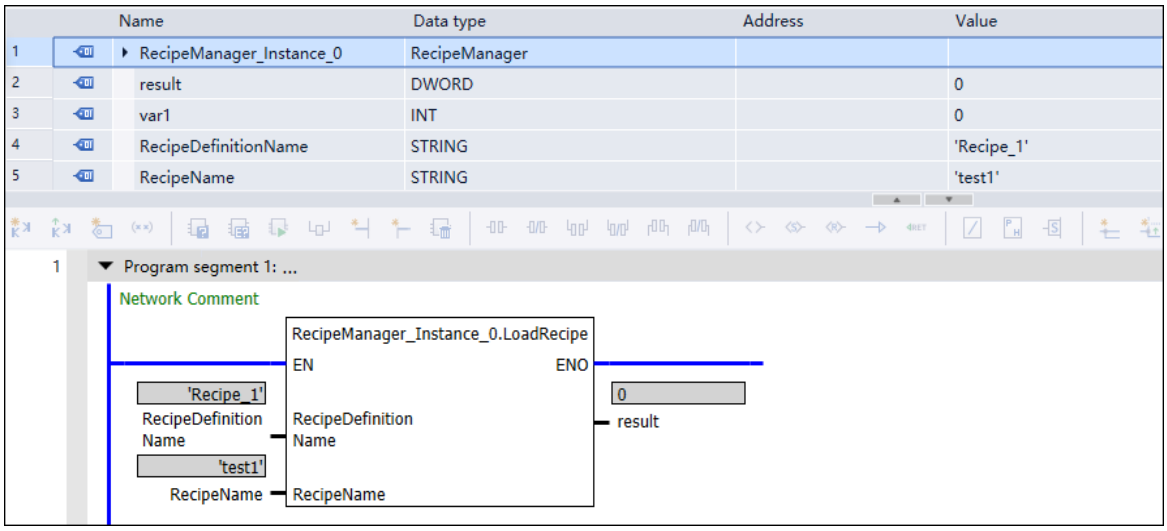
  This instruction loads a recipe from the specified recipe file.

- **Program example**

  ST:



LD:



- **Precautions**

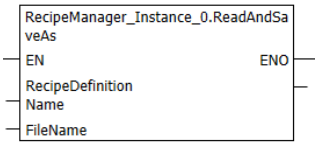
  It is recommended to use edge-triggered signals for the enable input EN.

18.2.18.1.12 ReadAndSaveAs

Reads the current PLC values corresponding to the variables in the recipe definition, and saves these data sets as a recipe file.

The existing standard recipe file <recipe definition>.<recipe>.txtrecipe will not be modified.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ReadAndSaveAs	Read PLC variables and save as a recipe file	Method		:= RecipeManager_Instance_0.ReadAndSaveAs(RecipeDefinitionName := , FileName :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name

Input/Output	Variable	Name	Valid Range	Default Value	Description
	FileName	Recipe name	Follows data type	-	Recipe name
Output	ReadAndSaveAs	Return value	Follows data type	-	Refer to Fault Code.

Data Type

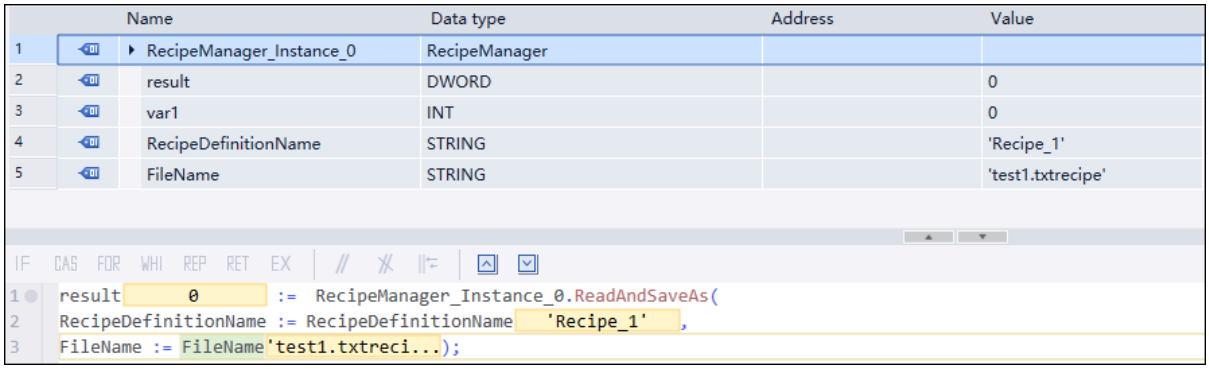
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
RecipeName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
ReadAndSaveAs	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

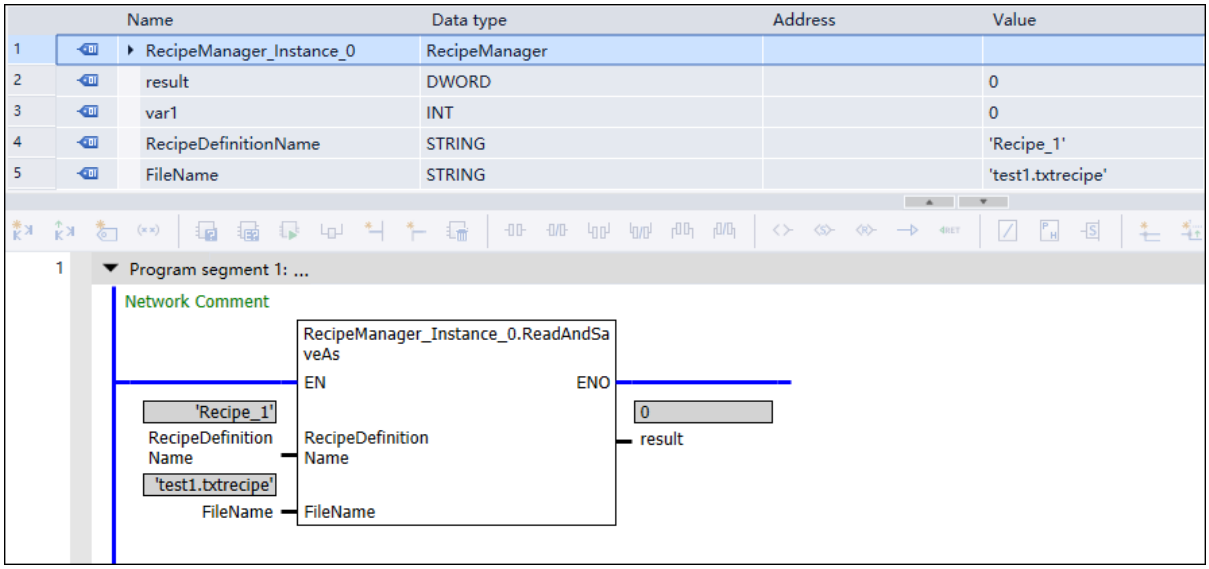
  This instruction reads the current values of the variables in the recipe definition and saves them as a recipe file.

- **Program example**

  ST:



LD:



- **Precautions**

  It is recommended to use edge-triggered signals for the enable input EN.

18.2.18.1.13 ReadAndSaveRecipe

Reads the current PLC variable values into the recipe, then stores the recipe into a recipe file. The default recipe file name format is <recipe definition>.<recipe>.txtrecipe.

If the target recipe file already exists, it will be overwritten.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ReadAndSaveRecipe	Read PLC variables into a recipe and save the recipe file	Method	RecipeManager_Instance_0.ReadAndSaveRecipe EN ENO RecipeDefinition Name RecipeName	:= RecipeManager_Instance_0.ReadAndSaveRecipe(RecipeDefinitionName := , RecipeName :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name
	RecipeName	Recipe name	Follows data type	-	Recipe name
Output	ReadAndSaveRecipe	Return value	Follows data type	-	Refer to Fault Code.

Data Type

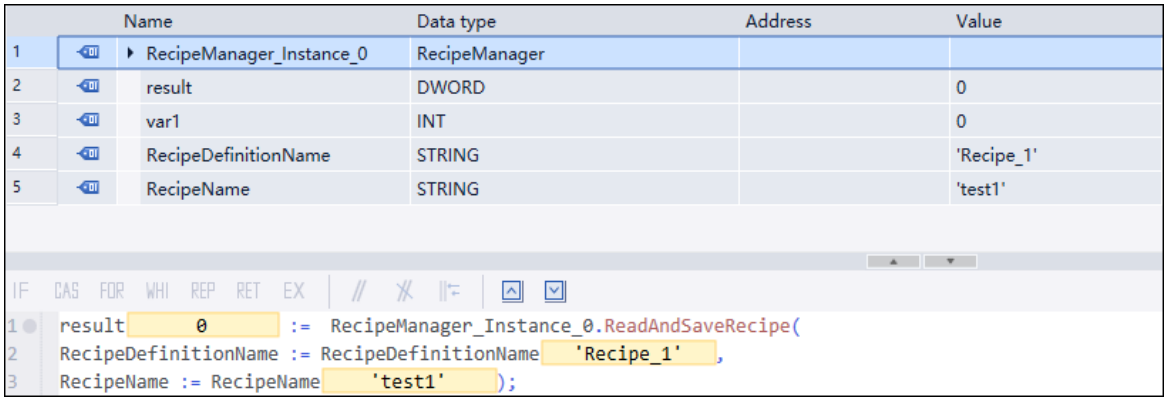
	Bool ean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
RecipeName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
ReadAndSaveRecipe	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

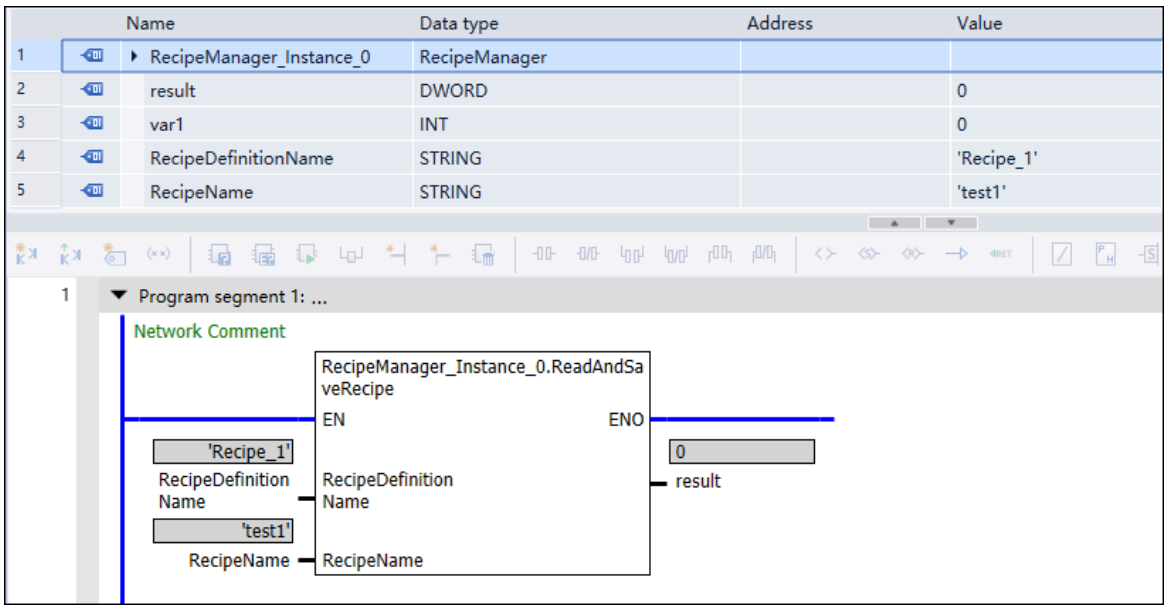
  This instruction reads the current PLC variable values into the recipe, then stores the recipe into a recipe file.

- **Program example**

ST:



LD:



- **Precautions**

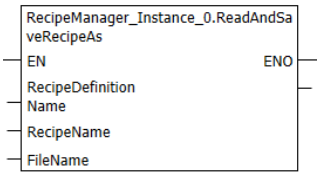
  It is recommended to use edge-triggered signals for the enable input EN.

18.2.18.1.14 ReadAndSaveRecipeAs

Reads the current PLC variable values into the recipe, then saves the recipe as a recipe file.

If a file with the same name exists, the existing file will be overwritten.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ReadAndSaveRecipeAs	Read PLC variables into a recipe and save as a recipe file	Method		:= RecipeManager_Instance_0.ReadAndSaveRecipeAs(RecipeDefinitionName := , RecipeName := , FileName :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name
	RecipeName	Recipe name	Follows data type	-	Recipe name
	FileName	File name	Follows data type	-	File name
Output	ReadAndSaveRecipeAs	Return value	Follows data type	-	Refer to Fault Code.

Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
RecipeName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
FileName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

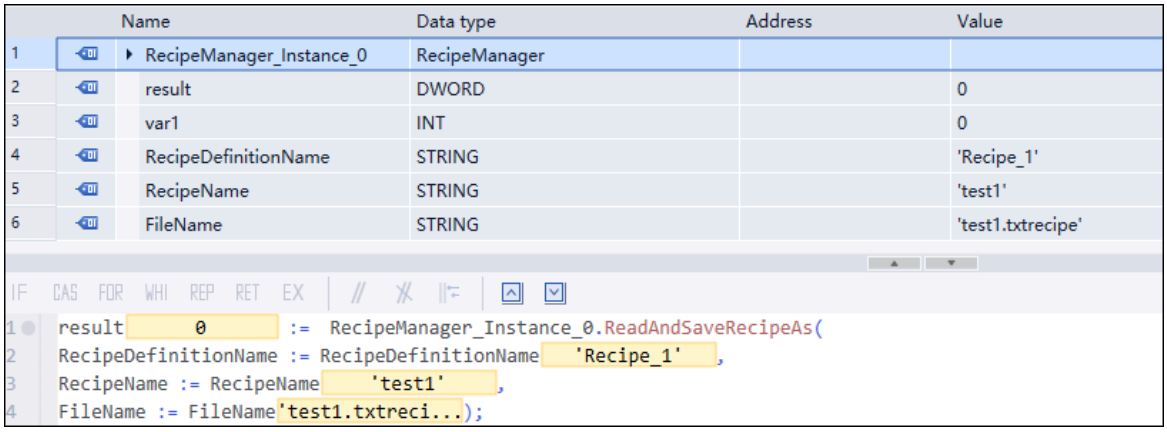
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ReadAndSaveRecipeAs	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

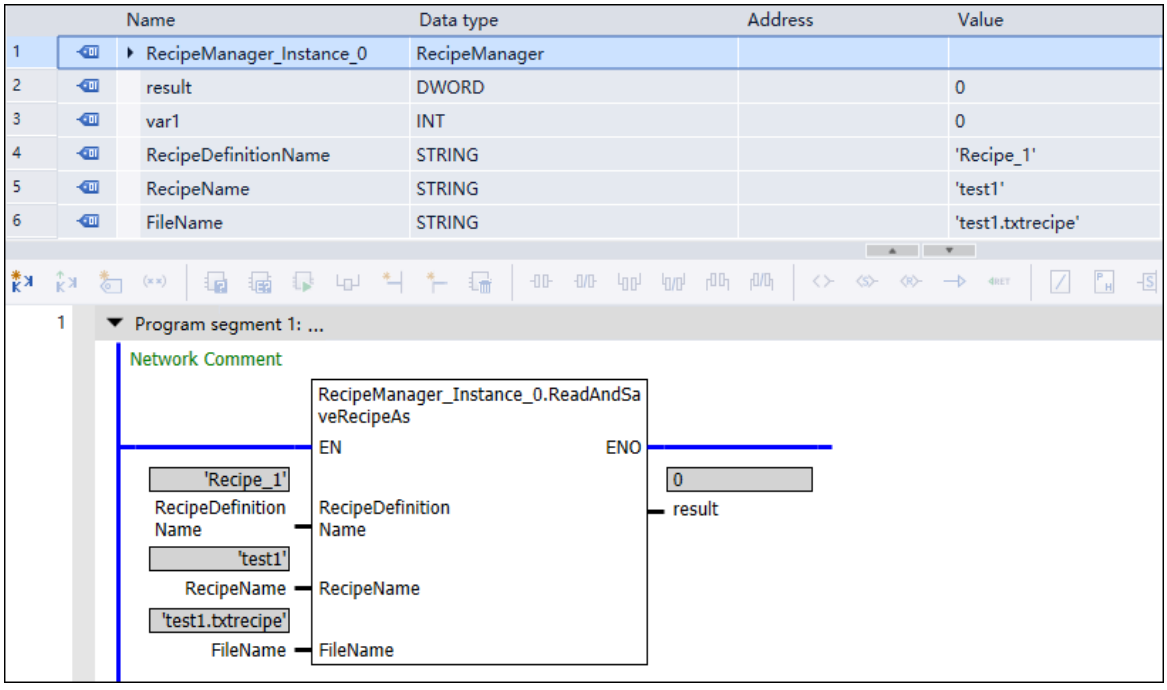
  This instruction reads the current PLC variable values into the recipe, then saves the recipe as a recipe file.

- **Program example**

  ST:



  LD:



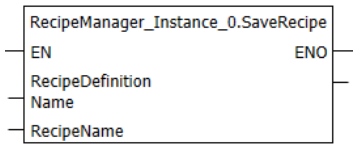
- **Precautions**

  It is recommended to use edge-triggered signals for the enable input EN.

18.2.18.1.15 SaveRecipe

Saves the specified recipe as a recipe file. If a file with the same name exists, the existing file will be overwritten. The default format for recipe files is <recipe definition>.<recipe>.txtrecipe.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SaveRecipe	Save a recipe file	Method		:= RecipeManager_Instance_0.SaveRecipe(RecipeDefinitionName := , RecipeName :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name
	RecipeName	Recipe name	Follows data type	-	Recipe name
Output	SaveRecipe	Return value	Follows data type	-	Refer to Fault Code.

Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
RecipeName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
SaveRecipe	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

This instruction saves the specified recipe as a recipe file.

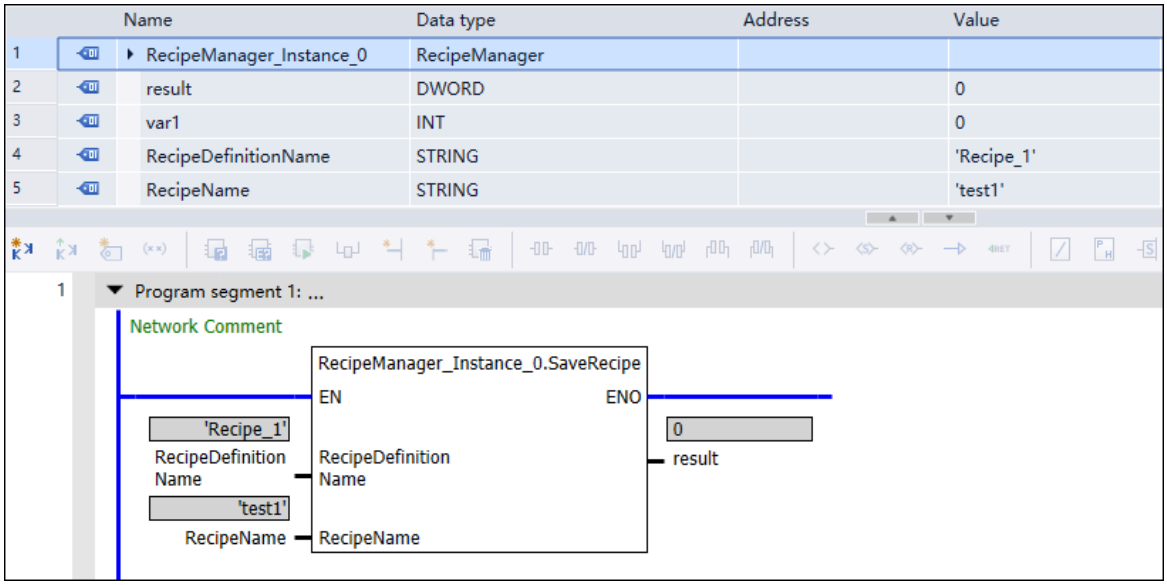
- **Program example**

ST:

	Name	Data type	Address	Value
1	RecipeManager_Instance_0	RecipeManager		
2	result	DWORD		0
3	var1	INT		0
4	RecipeDefinitionName	STRING		'Recipe_1'
5	RecipeName	STRING		'test1'

```
result := RecipeManager_Instance_0.SaveRecipe(
RecipeDefinitionName := RecipeDefinitionName 'Recipe_1',
RecipeName := RecipeName 'test1');
```

LD:



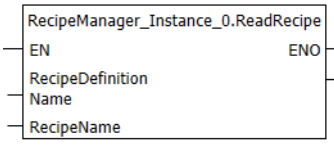
- **Precautions**

  It is recommended to use edge-triggered signals for the enable input EN.

18.2.18.1.16 ReadRecipe

Reads the current PLC variable values and saves it to the recipe.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ReadRecipe	Read PLC variables into a recipe	Method		:= RecipeManager_Instance_0.ReadRecipe(RecipeDefinitionName := , RecipeName :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name

Input/Output	Variable	Name	Valid Range	Default Value	Description
	RecipeName	Recipe name	Follows data type	-	Recipe name
Output	ReadRecipe	Return value	Follows data type	-	Refer to Fault Code.

Data Type

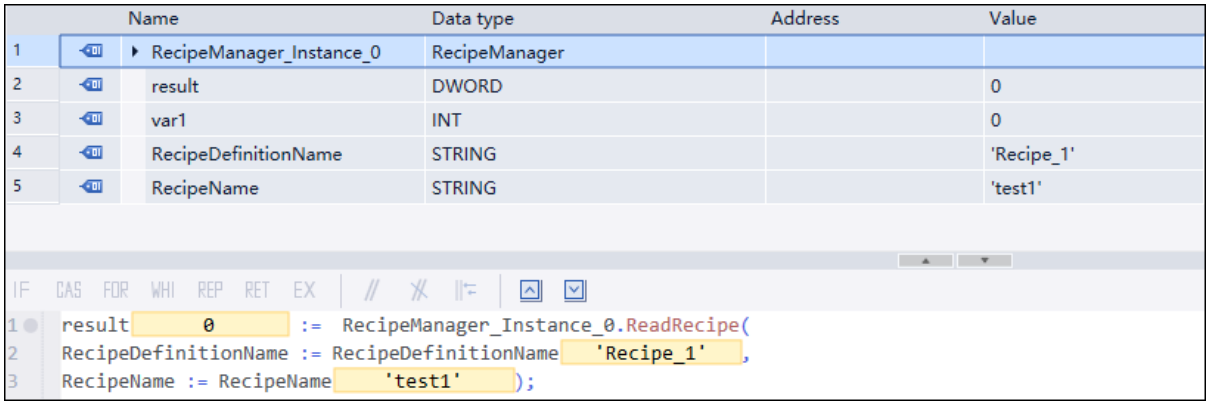
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
RecipeName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
ReadRecipe	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

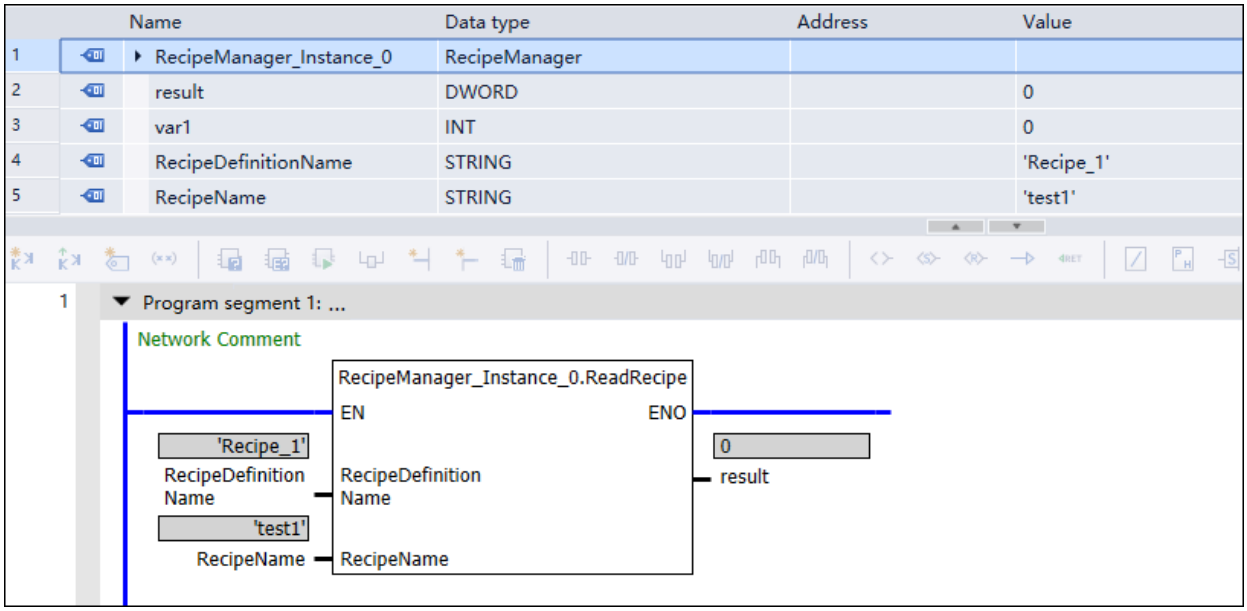
  This instruction reads the current PLC variable values and saves it to the recipe.

- **Program example**

  ST:



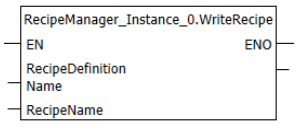
LD:



18.2.18.1.17 WriteRecipe

Writes the variable values from a recipe into the current PLC variables.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
WriteRecipe	Write a recipe	Method		:= RecipeManager_Instance_0.WriteRecipe(RecipeDefinitionName := , RecipeName :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name
	RecipeName	Recipe name	Follows data type	-	Recipe name

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	WriteRecipe	Return value	Follows data type	-	Refer to Fault Code.

Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
RecipeName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
WriteRecipe	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction writes the variable values from a recipe into the current PLC variables.

- **Program example**

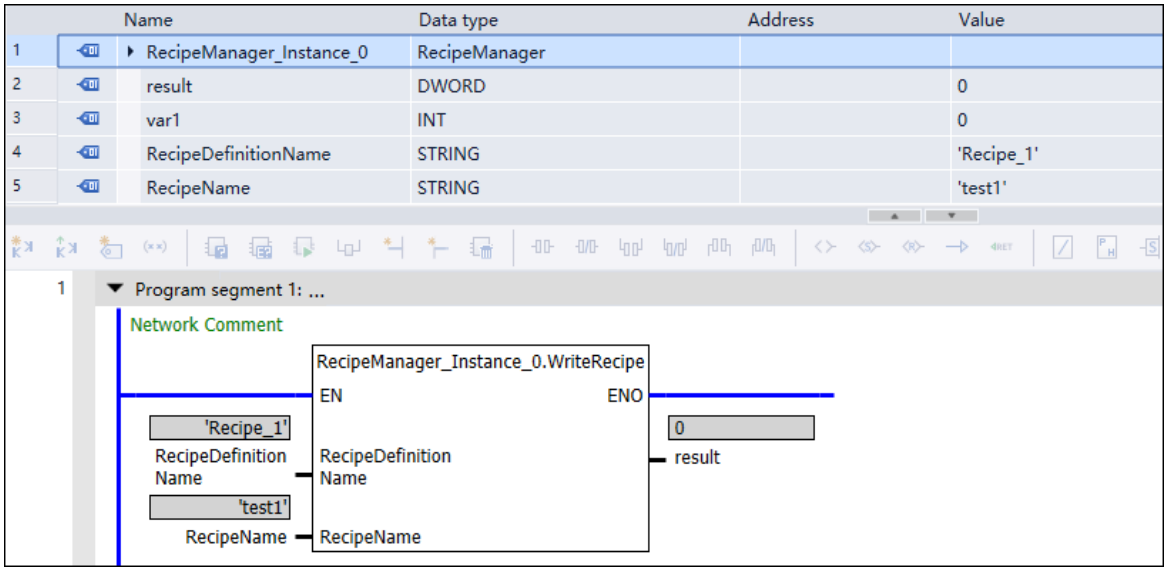
  ST:

	Name	Data type	Address	Value
1	RecipeManager_Instance_0	RecipeManager		
2	result	DWORD		0
3	var1	INT		0
4	RecipeDefinitionName	STRING		'Recipe_1'
5	RecipeName	STRING		'test1'

IF CAS FOR WHI REP RET EX //

```
result := RecipeManager_Instance_0.WriteRecipe(
RecipeDefinitionName := RecipeDefinitionName,
RecipeName := RecipeName);
```

LD:



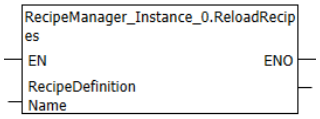
- **Precautions**

  It is recommended to use edge-triggered signals for the enable input EN.

18.2.18.1.18 ReloadRecipes

Loads all recipe files under the storage path into the recipes, respectively. The recipe files must be saved with the default filename format <recipe definition>.<recipe>.txtrecipe.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ReloadRecipes	Reload recipes	Method		:= RecipeManager_Instance_0.ReloadRecipes(RecipeDefinitionName :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name
Output	ReloadRecipes	Return value	Follows data type	-	Refer to Fault Code.

  Data Type

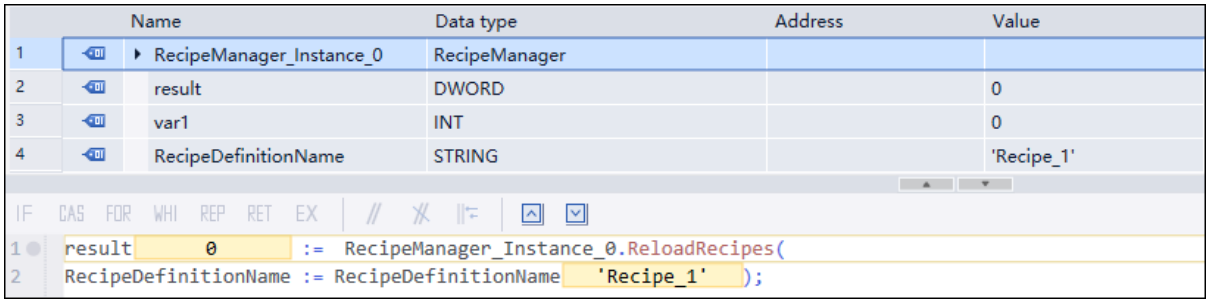
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
ReloadRecipes	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

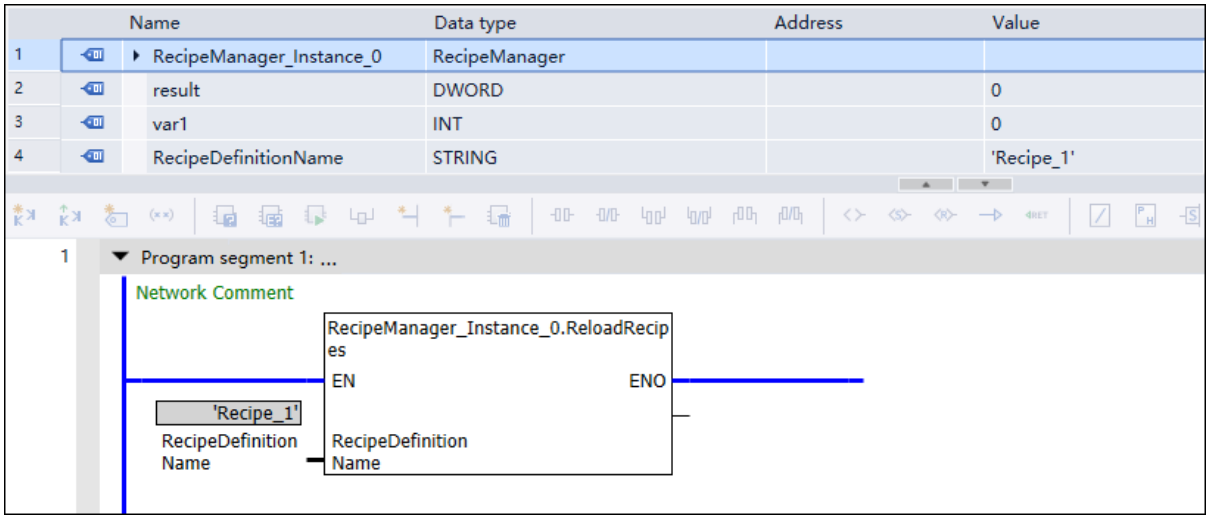
  This instruction loads all recipe files under the storage path into the recipes, respectively.

- **Program example**

  ST:



  LD:



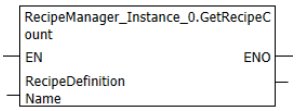
- **Precautions**

  It is recommended to use edge-triggered signals for the enable input EN.

18.2.18.1.19 GetRecipeCount

Gets the number of recipes associated with the specified recipe definition.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetRecipeCount	Get the number of recipes associated to a specified recipe	Method		:= RecipeManager_Instance_0.GetRecipeCount(RecipeDefinitionName :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name
Output	GetRecipeCount	Recipe count	Follows data type	-	Recipe count

  Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
GetRecipeCount	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction gets the number of recipes associated with the specified recipe definition.

- **Program example**

  Call the CreateRecipe instruction to create a new recipe. Then, call the GetRecipeCount instruction to get the number of recipes associated with the specified recipe definition.

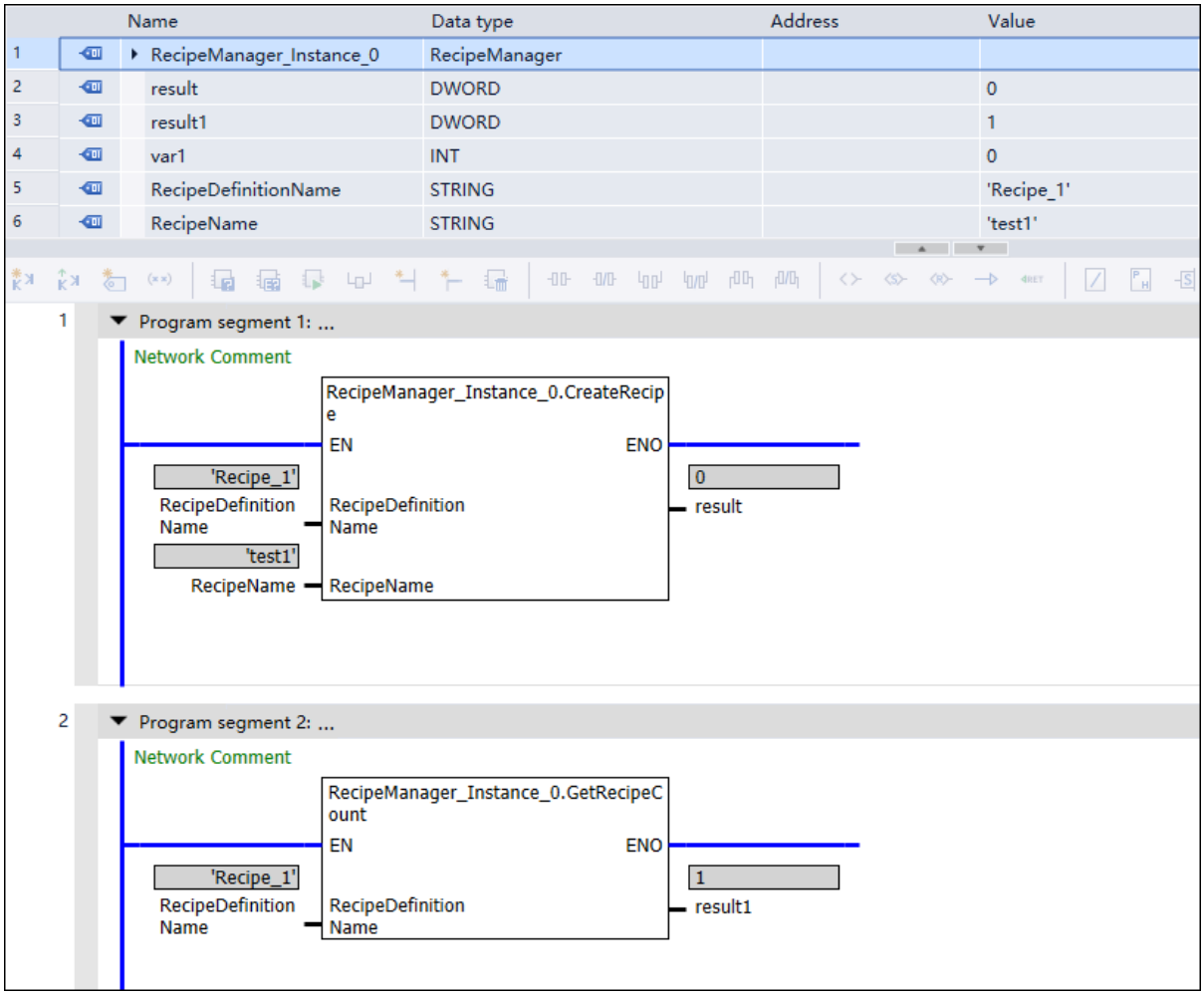
  ST:

	Name	Data type	Address	Value
1	RecipeManager_Instance_0	RecipeManager		
2	result	DWORD		0
3	result1	DWORD		1
4	var1	INT		0
5	RecipeDefinitionName	STRING		'Recipe_1'
6	RecipeName	STRING		'test1'

```
1  result 0 := RecipeManager_Instance_0.CreateRecipe(
2  RecipeDefinitionName := RecipeDefinitionName 'Recipe_1',
3  RecipeName := RecipeName 'test1');
4
5  result1 1 := RecipeManager_Instance_0.GetRecipeCount(
6  RecipeDefinitionName := RecipeDefinitionName 'Recipe_1');
```

LD:

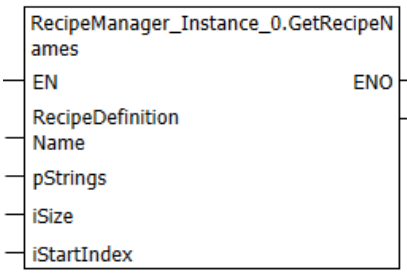
	Name	Data type	Address	Value
1	RecipeManager_Instance_0	RecipeManager		
2	result	DWORD		0
3	result1	DWORD		1
4	var1	INT		0
5	RecipeDefinitionName	STRING		'Recipe_1'
6	RecipeName	STRING		'test1'



18.2.18.1.20 GetRecipeNames

Gets the names of recipes associated with the specified recipe definition.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetRecipeNames	Get the recipe names associated to a specified recipe	Method		:= RecipeManager_Instance_0.GetRecipeNames(RecipeDefinitionName := , pStrings := , iSize := , iStartIndex :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name
	pStrings	Pointer to the string array that stores recipe names	Follows data type	-	Pointer to the string array that stores recipe names
	iSize	Size of the string array	Follows data type	-	Size of the string array
	iStartIndex	Start index	Follows data type	-	Start index
Output	RecipeDefinitionName	Return value	Follows data type	-	Refer to Fault Code.

Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
iSize	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
pStrings	POINTER TO ARRAY [ ] OF STRING																			
iStartIndex	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
RecipeDefinitionName	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction gets the names of recipes associated with the specified recipe definition.

- **Program example**

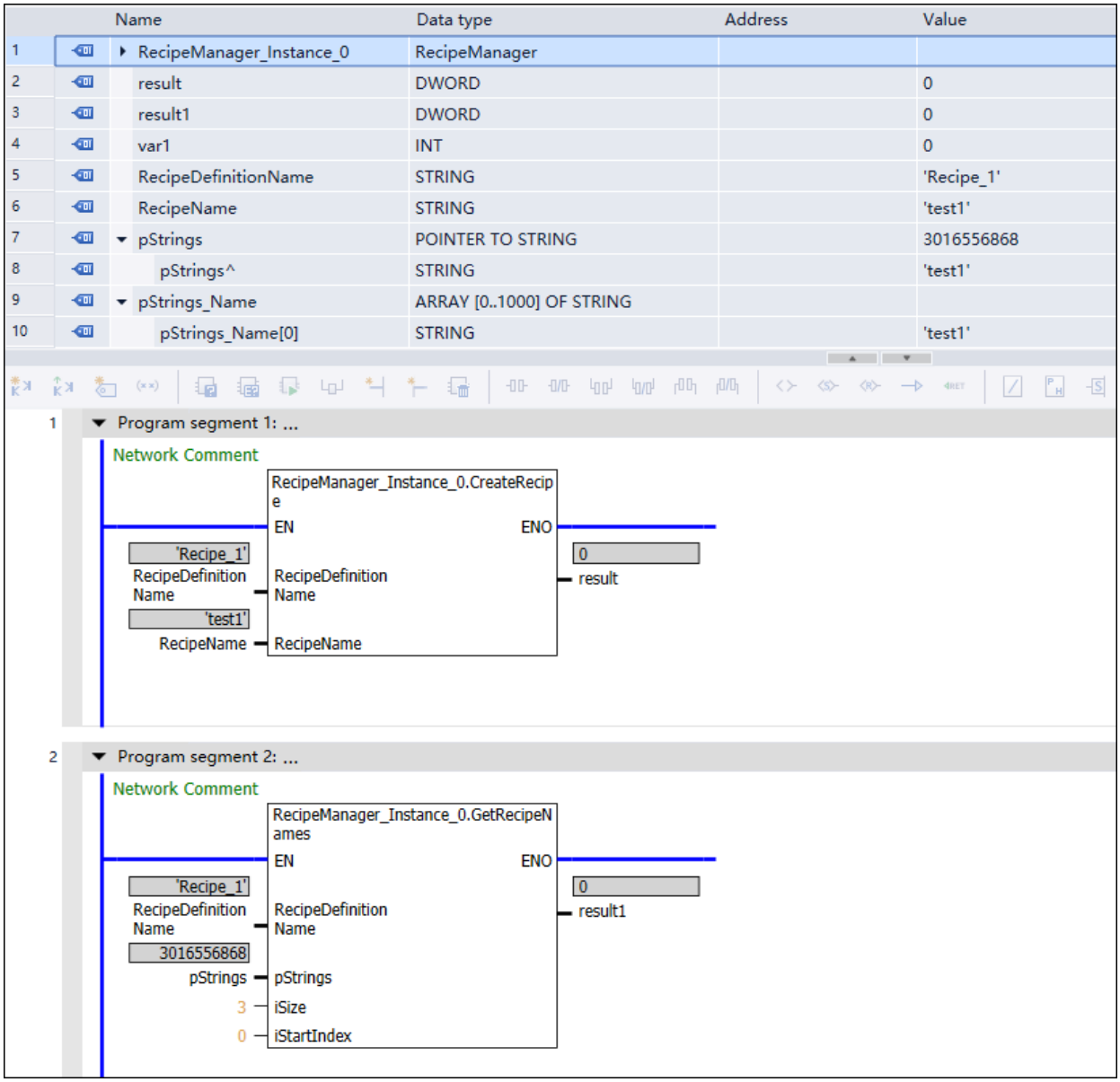
  Call the CreateRecipe instruction to create a new recipe. Then, call the GetRecipeNames instruction to get the names of recipes associated with the specified recipe definition.

  ST:

	Name	Data type	Address	Value
1	RecipeManager_Instance_0	RecipeManager		
2	result	DWORD		0
3	result1	DWORD		0
4	var1	INT		10
5	var2	INT		20
6	var3	INT		30
7	RecipeDefinitionName	STRING		'Recipe_1'
8	RecipeName	STRING		'test1'
9	pStrings	POINTER TO STRING		3016475484
10	pStrings^	STRING		'test1'
11	pStrings_Name	ARRAY [0..1000] OF STRING		
12	pStrings_Name[0]	STRING		'test1'

```
IF  CAS  FOR  WHI  REP  RET  EX
1  result 0 := RecipeManager_Instance_0.CreateRecipe(
2  RecipeDefinitionName := RecipeDefinitionName 'Recipe_1',
3  RecipeName := RecipeName 'test1');
4
5  result1 0 := RecipeManager_Instance_0.GetRecipeNames(
6  RecipeDefinitionName := RecipeDefinitionName 'Recipe_1',
7  pStrings := pStrings 3016475484,
8  iSize := 3,
9  iStartIndex := 0);
```

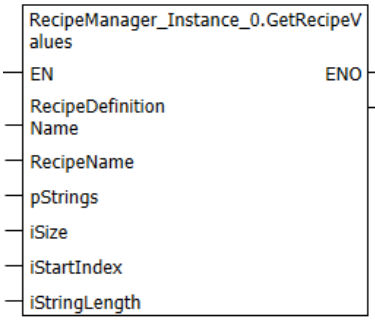
  LD:



18.2.18.1.21 GetRecipeValues

Gets the recipe variable values in the specified recipe.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetRecipeValues	Get the variable values of a specified recipe	Method		:= RecipeManager_Instance_0.GetRecipeValues(RecipeDefinitionName := , RecipeName := , pStrings := , iSize := , iStartIndex := , iStringLength :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name
	RecipeName	Recipe name	Follows data type	-	Recipe name
	pStrings	Pointer to the string array that stores recipe values	Follows data type	-	String array of recipe values
	iSize	Size of the string array	Follows data type	-	Size of the string array
	iStartIndex	Array start index	Follows data type	-	Start index
	iStringLength	String length	Follows data type	-	String length
Output	GetRecipeValues	Return value	Follows data type	-	Refer to Fault Code.

  Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
RecipeName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
pStrings	POINTER TO ARRAY [ ] OF STRING																			
iSize	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
iStartIndex	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
iStringLength	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
GetRecipeValues	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction gets the recipe variable values in the specified recipe.

- **Program example**

  Call the CreateRecipe instruction to generate create a new recipe. Then, call the SetRecipeValues instruction to set the values of recipe variables in the specified recipe. Finally, call the GetRecipeValues instruction to get the recipe variable values in the specified recipe.

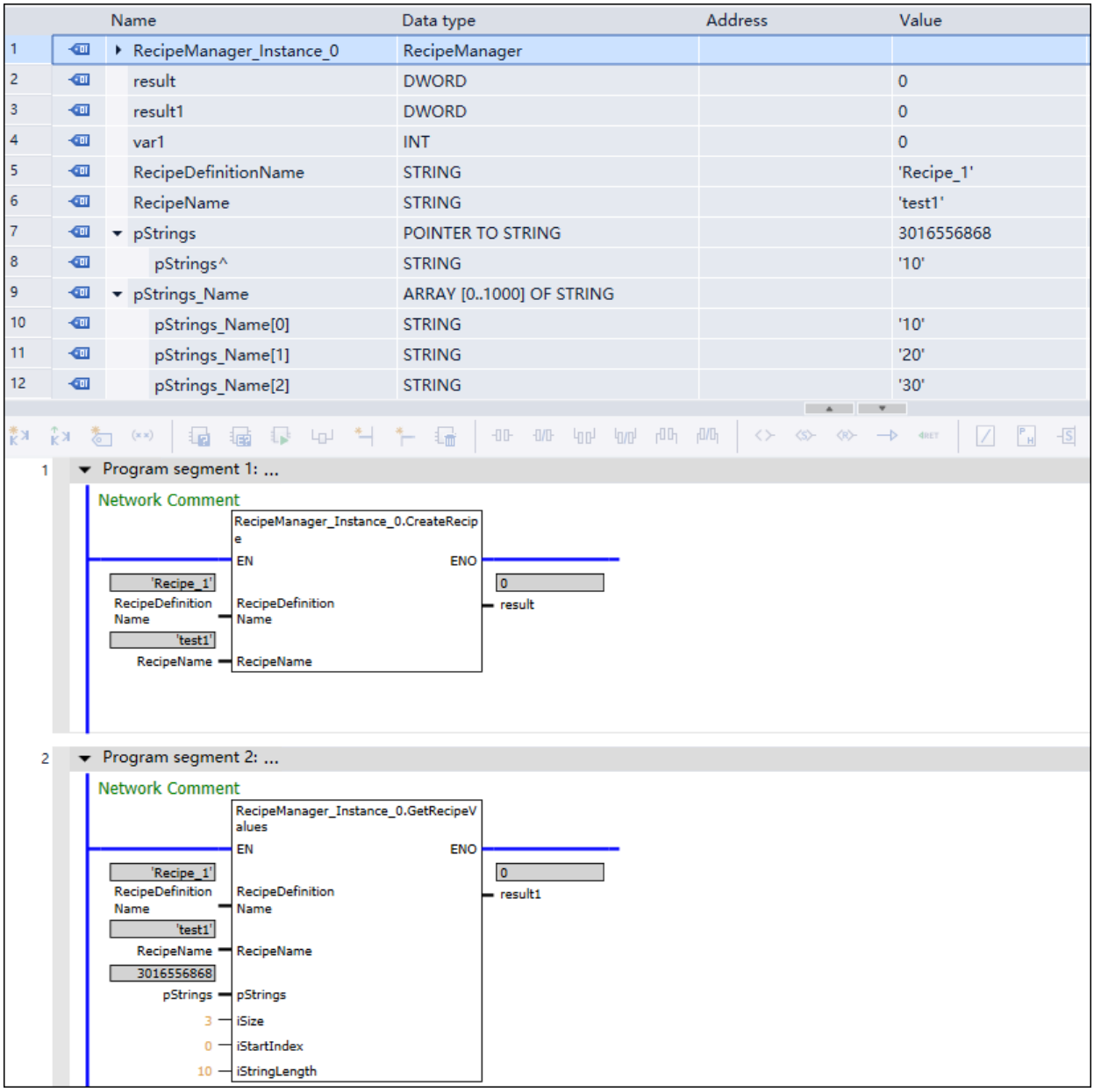
  ST:

	Name	Data type	Address	Value
1	RecipeManager_Instance_0	RecipeManager		
2	result	DWORD		0
3	result1	DWORD		0
4	var1	INT		10
5	var2	INT		20
6	var3	INT		30
7	RecipeDefinitionName	STRING		'Recipe_1'
8	RecipeName	STRING		'test1'
9	pStrings	POINTER TO STRING		3016475484
10	pStrings^	STRING		'10'
11	pStrings_Name	ARRAY [0..1000] OF STRING		
12	pStrings_Name[0]	STRING		'10'
13	pStrings_Name[1]	STRING		'20'
14	pStrings_Name[2]	STRING		'30'

```
result := RecipeManager_Instance_0.CreateRecipe(
RecipeDefinitionName := RecipeDefinitionName := 'Recipe_1',
RecipeName := RecipeName := 'test1');

result := RecipeManager_Instance_0.GetRecipeValues(
RecipeDefinitionName := RecipeDefinitionName := 'Recipe_1',
RecipeName := RecipeName := 'test1',
pStrings := pStrings := 3016475484,
iSize := 3,
iStartIndex := 0,
iStringLength := 10);
```

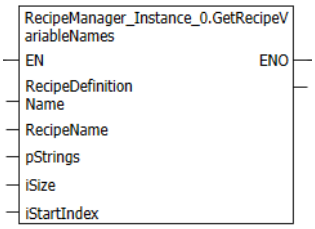
LD:



18.2.18.1.22 GetRecipeVariableNames

Gets the recipe variable names in the specified recipe.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetRecipeVariableNames	Get the variable names of a specified recipe	Method		:= RecipeManager_Instance_0.GetRecipeVariableNames(RecipeDefinitionName := , RecipeName := , pStrings := , iSize := , iStartIndex :=);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name
	RecipeName	Recipe name	Follows data type	-	Recipe name
	pStrings	Pointer to the string array that stores recipe variable names	Follows data type	-	Returned data
	iSize	Size of the string array	Follows data type	-	Size of the string array
	iStartIndex	Start index	Follows data type	-	Start index
Output	GetRecipeVariableNames	Return value	Follows data type	-	Refer to Fault Code.

  Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
RecipeName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
pStrings	POINTER TO ARRAY [ ] OF STRING																			
iSize	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
iStartIndex	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
GetRecipeVariableNames	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction gets the recipe variable names in the specified recipe.

- **Program example**

  Call the CreateRecipe instruction to create a new recipe. Then, call the GetRecipeVariableNames instruction to get the recipe variable names in the specified recipe.

  ST:

	Name	Data type	Address	Value
1	RecipeManager_Instance_0	RecipeManager		
2	result	DWORD		0
3	result1	DWORD		0
4	var1	INT		10
5	var2	INT		20
6	var3	INT		30
7	RecipeDefinitionName	STRING		'Recipe_1'
8	RecipeName	STRING		'test1'
9	pStrings	POINTER TO STRING		3016475484
10	pStrings^	STRING		'POU.var1'
11	pStrings_Name	ARRAY [0..1000] OF STRING		
12	pStrings_Name[0]	STRING		'POU.var1'
13	pStrings_Name[1]	STRING		'POU.var2'
14	pStrings_Name[2]	STRING		'POU.var3'

```
result := RecipeManager_Instance_0.CreateRecipe(
RecipeDefinitionName := RecipeDefinitionName := 'Recipe_1',
RecipeName := RecipeName := 'test1');

result1 := RecipeManager_Instance_0.GetRecipeVariableNames(
RecipeDefinitionName := RecipeDefinitionName := 'Recipe_1',
RecipeName := RecipeName := 'test1',
pStrings := pStrings := 3016475484,
iSize := 3,
iStartIndex := 0);
```

LD:

	Name	Data type	Address	Value
1	RecipeManager_Instance_0	RecipeManager		
2	result	DWORD		0
3	result1	DWORD		0
4	var1	INT		10
5	var2	INT		20
6	var3	INT		30
7	RecipeDefinitionName	STRING		'Recipe_1'
8	RecipeName	STRING		'test1'
9	pStrings	POINTER TO STRING		3016475484
10	pStrings^	STRING		'POU.var1'
11	pStrings_Name	ARRAY [0..1000] OF STRING		
12	pStrings_Name[0]	STRING		'POU.var1'
13	pStrings_Name[1]	STRING		'POU.var2'
14	pStrings_Name[2]	STRING		'POU.var3'

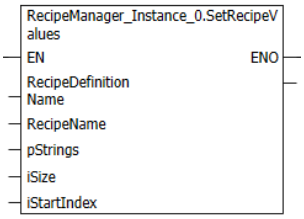
```
result := RecipeManager_Instance_0.CreateRecipe(
RecipeDefinitionName := RecipeDefinitionName := 'Recipe_1',
RecipeName := RecipeName := 'test1');

result1 := RecipeManager_Instance_0.GetRecipeVariableNames(
RecipeDefinitionName := RecipeDefinitionName := 'Recipe_1',
RecipeName := RecipeName := 'test1',
pStrings := pStrings := 3016475484,
iSize := 3,
iStartIndex := 0);
```

18.2.18.1.23 SetRecipeValues

Sets the recipe variable values in the specified recipe.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SetRecipeValues	Set the variable values of a specified recipe	Method		:= RecipeManager_Instance_0.SetRecipeValues(RecipeDefinitionName := , RecipeName := , pStrings := , iSize := , iStartIndex :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name
	RecipeName	Recipe name	Follows data type	-	Recipe name
	pStrings	Pointer to the string array that stores recipe values	Follows data type	-	Returned data
	iSize	Size of the string array	Follows data type	-	Size of the string array
	iStartIndex	Start index	Follows data type	-	Start index
Output	SetRecipeValues	Return value	Follows data type	-	Refer to Fault Code.

Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
pStrings	POINTER TO ARRAY [ ] OF STRING																			
iSize	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
iStartIndex	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
SetRecipeValues	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction sets the recipe variable values in the specified recipe.

- **Program example**

  Call the CreateRecipe instruction to generate create a new recipe. Then, call the SetRecipeValues instruction to set the values of recipe variables in the specified recipe. Finally, call the GetRecipeValues instruction to get the recipe variable values in the specified recipe.

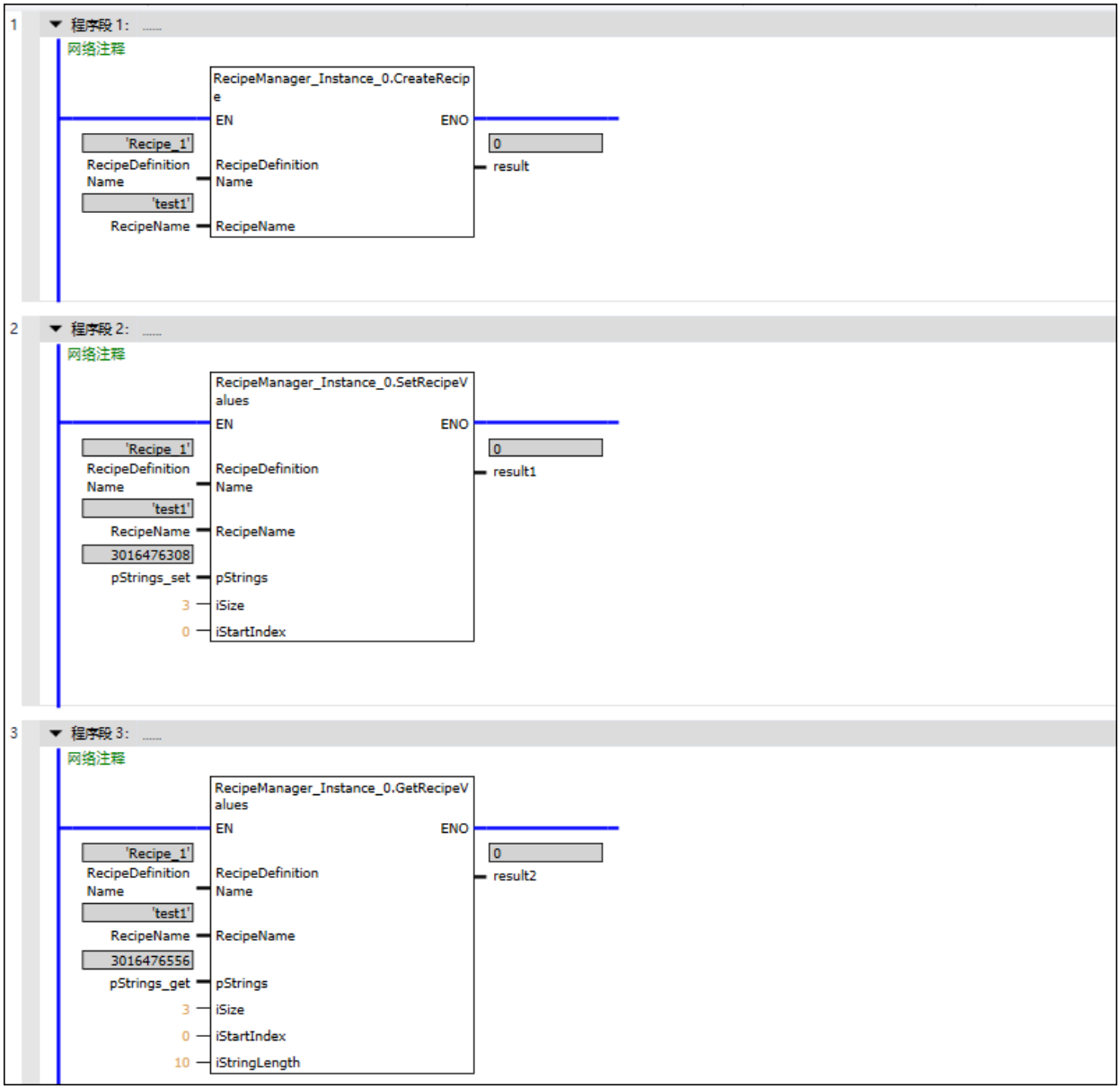
  ST:

	Name	Data type	Address	Value
1	RecipeManager_Instance_0	RecipeManager		
2	result	DWORD		0
3	result1	DWORD		0
4	result2	DWORD		0
5	var1	INT		0
6	var2	INT		0
7	var3	INT		0
8	RecipeDefinitionName	STRING		'Recipe_1'
9	RecipeName	STRING		'test1'
10	pStrings_set	POINTER TO STRING		3016475492
11	pStrings_Name_set	ARRAY [0..2] OF STRING		
12	pStrings_Name_set[0]	STRING		'10'
13	pStrings_Name_set[1]	STRING		'20'
14	pStrings_Name_set[2]	STRING		'30'
15	pStrings_get	POINTER TO STRING		3016475740
16	pStrings_Name_get	ARRAY [0..2] OF STRING		
17	pStrings_Name_get[0]	STRING		'10'
18	pStrings_Name_get[1]	STRING		'20'
19	pStrings_Name_get[2]	STRING		'30'

```
result := RecipeManager_Instance_0.CreateRecipe(
RecipeDefinitionName := RecipeDefinitionName 'Recipe_1',
RecipeName := RecipeName 'test1');
result1 := RecipeManager_Instance_0.SetRecipeValues(
RecipeDefinitionName := RecipeDefinitionName 'Recipe_1',
RecipeName := RecipeName 'test1',
pStrings := pStrings_set 3016475492,
iSize := 3,
iStartIndex := 0);
result2 := RecipeManager_Instance_0.GetRecipeValues(
RecipeDefinitionName := RecipeDefinitionName 'Recipe_1',
RecipeName := RecipeName 'test1',
pStrings := pStrings_get 3016475740,
iSize := 3,
iStartIndex := 0,
iStringLength := 10);
```

LD:

	Name	Data type	Address	Value
1	RecipeManager_Instance_0	RecipeManager		
2	result	DWORD		0
3	result1	DWORD		0
4	result2	DWORD		0
5	var1	INT		0
6	var2	INT		0
7	var3	INT		0
8	RecipeDefinitionName	STRING		'Recipe_1'
9	RecipeName	STRING		'test1'
10	pStrings_set	POINTER TO STRING		3016476308
11	pStrings_Name_set	ARRAY [0..2] OF STRING		
12	pStrings_Name_set[0]	STRING		'10'
13	pStrings_Name_set[1]	STRING		'20'
14	pStrings_Name_set[2]	STRING		'30'
15	pStrings_get	POINTER TO STRING		3016476556
16	pStrings_Name_get	ARRAY [0..2] OF STRING		
17	pStrings_Name_get[0]	STRING		'10'
18	pStrings_Name_get[1]	STRING		'20'
19	pStrings_Name_get[2]	STRING		'30'



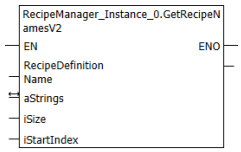
- **Precautions**

  It is recommended to use edge-triggered signals for the enable input EN.

18.2.18.1.24 GetRecipeNamesV2

Gets the names of recipes associated with the specified recipe definition.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetRecipeNamesV2	Get the recipe names associated to a specified recipe	Method		:= RecipeManager_Instance_0.GetRecipeNamesV2(RecipeDefinitionName := , aStrings := , iSize := , iStartIndex :=);

- **Related variables**

Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name
	iSize	Size of the string array	Follows data type	-	Size of the string array
	iStartIndex	Start index	Follows data type	-	Start index
Input/ Output	aStrings	String array that stores recipe names	Follows data type	-	String array that stores recipe names
Output	GetRecipeNamesV2	Return value	Follows data type	-	Refer to Fault Code.

Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
iSize	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
iStartIndex	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
aStrings	String array ARRAY [*] OF STRING																			

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
GetRecipeNamesV2	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction gets the names of recipes associated with the specified recipe definition.

- **Program example**

  Call the CreateRecipe instruction to create a new recipe. Then, call the GetRecipeNamesV2 instruction to get the names of recipes associated with the specified recipe definition.

  ST:

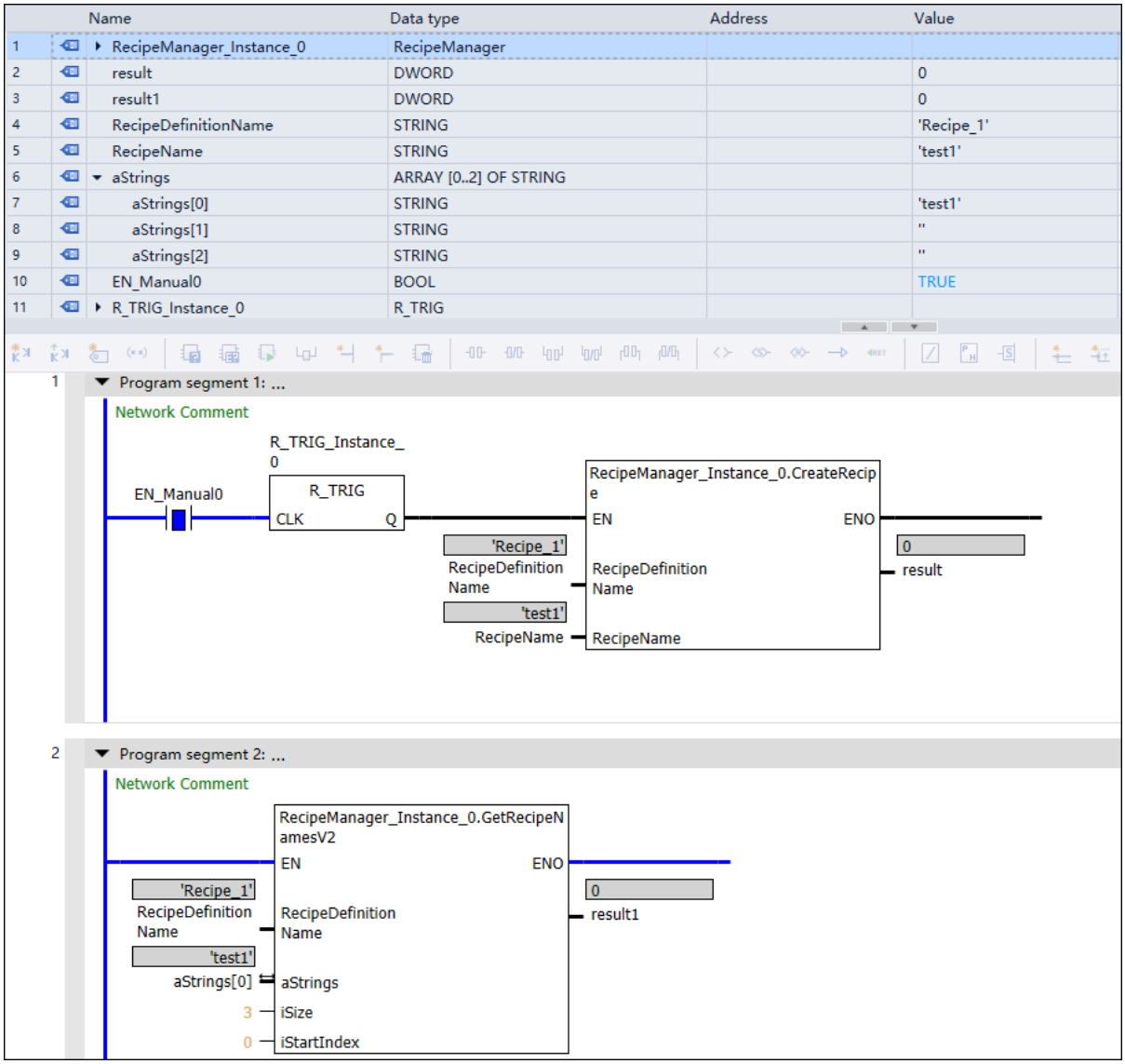
	Name	Data type	Address	Value
1	RecipeManager_Instance_0	RecipeManager		
2	result	DWORD		0
3	result1	DWORD		0
4	RecipeDefinitionName	STRING		'Recipe_1'
5	RecipeName	STRING		'test1'
6	aStrings	ARRAY [0..2] OF STRING		
7	aStrings[0]	STRING		'test1'
8	aStrings[1]	STRING		''
9	aStrings[2]	STRING		''
10	EN_Manual0	BOOL		TRUE
11	R_TRIG_Instance_0	R_TRIG		

```
R_TRIG_Instance_0(
CLK := EN_Manual0,
Q => );

IF R_TRIG_Instance_0.Q THEN
    result := RecipeManager_Instance_0.CreateRecipe(
    RecipeDefinitionName := RecipeDefinitionName 'Recipe_1' ,
    RecipeName := RecipeName 'test1' );
END_IF

result1 := RecipeManager_Instance_0.GetRecipeNamesV2(
RecipeDefinitionName := RecipeDefinitionName 'Recipe_1' ,
aStrings := aStrings[0] 'test1' ,
iSize := 3,
iStartIndex := 0);RETURN
```

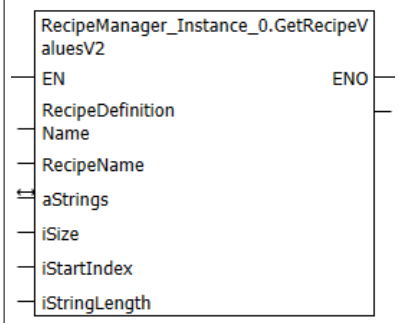
  LD:



18.2.18.1.25 GetRecipeValuesV2

Gets the recipe variable values in the specified recipe.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetRecipeValuesV2	Get the variable values of a specified recipe	Method		:= RecipeManager_Instance_0.GetRecipeValuesV2(RecipeDefinitionName := , RecipeName := , aStrings := , iSize := , iStartIndex := , iStringLength :=);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name
	RecipeName	Recipe name	Follows data type	-	Recipe name
	iSize	Size of the string array	Follows data type	-	Size of the string array
	iStartIndex	Array start index	Follows data type	-	Start index
	iStringLength	String length	Follows data type	-	String length
Input/ Output	aStrings	String array to store recipe values	Follows data type	-	String array to store recipe values
Output	GetRecipeValuesV2	Return value	Follows data type	-	Refer to Fault Code.

Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
RecipeName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
iSize	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
iStartIndex	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
iStringLength	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
aStrings	String array ARRAY [*] OF STRING																			
GetRecipeValuesV2	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction gets the recipe variable values in the specified recipe.

- **Program example**

Call the CreateRecipe instruction to generate create a new recipe. Then, call the SetRecipeValuesV2 instruction to set the values of recipe variables in the specified recipe. Finally, call the GetRecipeValuesV2 instruction to get the recipe variable values in the specified recipe.

ST:

	Name	Data type	Address	Value
1	RecipeManager_Instance_0	RecipeManager		
2	result	DWORD		0
3	result1	DWORD		0
4	result2	DWORD		0
5	RecipeDefinitionName	STRING		'Recipe_1'
6	RecipeName	STRING		'test1'
7	aStrings_set	ARRAY [0..2] OF STRING		
8	aStrings_set[0]	STRING		'TRUE'
9	aStrings_set[1]	STRING		'DT#2025-5-15-11:11:11'
10	aStrings_set[2]	STRING		'abcde'
11	aStrings_get	ARRAY [0..2] OF STRING		
12	aStrings_get[0]	STRING		'TRUE'
13	aStrings_get[1]	STRING		'DT#2025-5-15-11:11:11'
14	aStrings_get[2]	STRING		'abcde'
15	EN_Manual0	BOOL		TRUE
16	EN_Manual1	BOOL		TRUE
17	EN_Manual2	BOOL		TRUE
18	R_TRIG_Instance_0	R_TRIG		
19	R_TRIG_Instance_1	R_TRIG		

```
R_TRIG_Instance_0(
CLK TRUE := EN_Manual0 TRUE,
Q => );

IF R_TRIG_Instance_0.Q FALSE THEN
    result 0 := RecipeManager_Instance_0.CreateRecipe(
    RecipeDefinitionName := RecipeDefinitionName 'Recipe_1',
    RecipeName := RecipeName 'test1');
END_IF

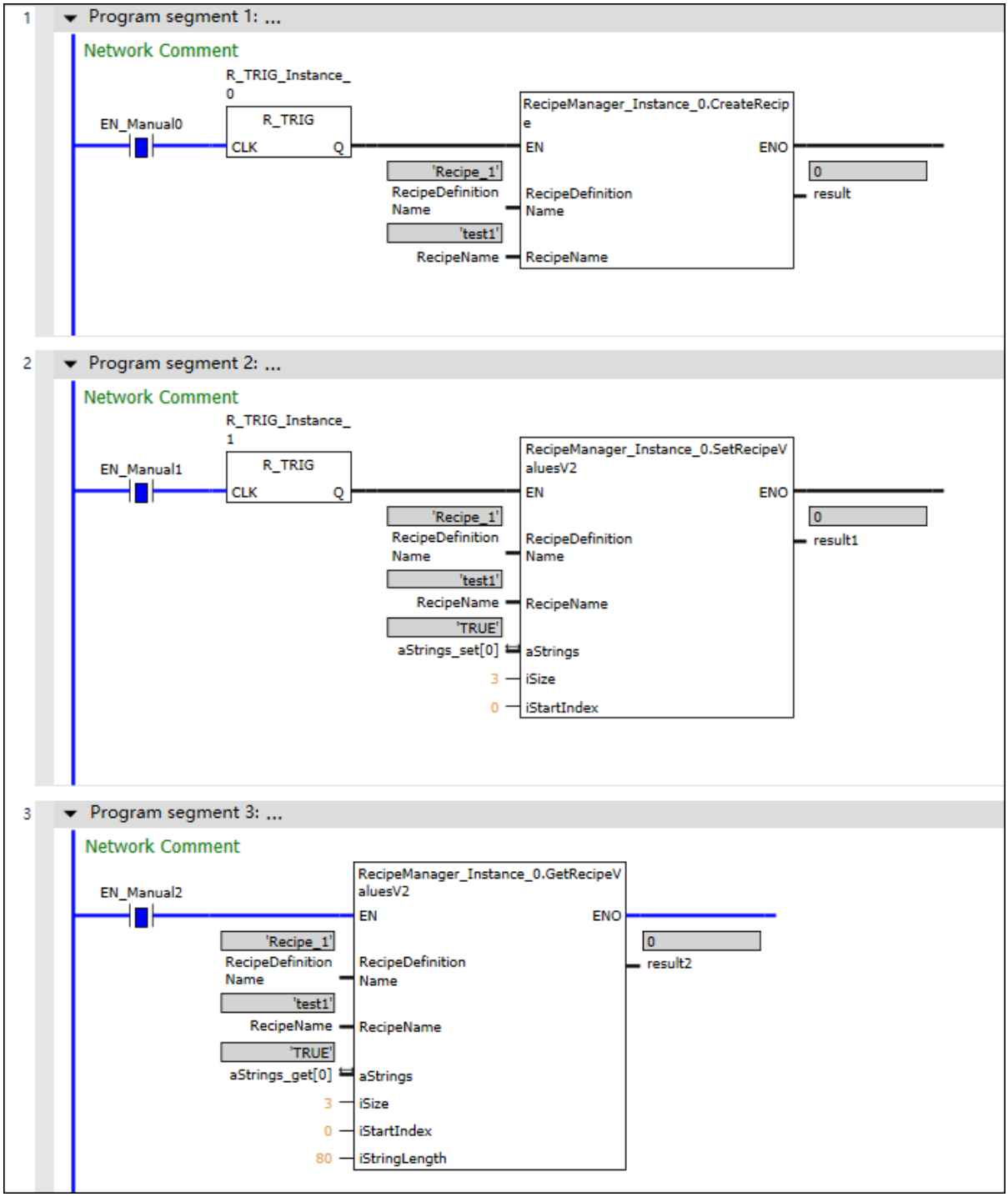
R_TRIG_Instance_1(
CLK TRUE := EN_Manual1 TRUE,
Q => );

IF R_TRIG_Instance_1.Q FALSE THEN
    result1 0 := RecipeManager_Instance_0.SetRecipeValuesV2(
    RecipeDefinitionName := RecipeDefinitionName 'Recipe_1',
    RecipeName := RecipeName 'test1',
    aStrings := aStrings_set[0] 'TRUE',
    iSize := 3,
    iStartIndex := 0);
END_IF

IF EN_Manual2 TRUE THEN
    result2 0 := RecipeManager_Instance_0.GetRecipeValuesV2(
    RecipeDefinitionName := RecipeDefinitionName 'Recipe_1',
    RecipeName := RecipeName 'test1',
    aStrings := aStrings_get[0] 'TRUE',
    iSize := 3,
    iStartIndex := 0,
    iStringLength := 80);
END_IFRETURN
```

LD:

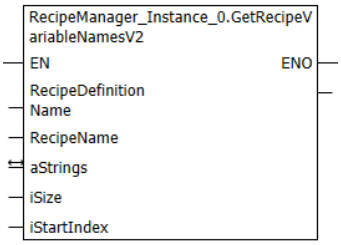
	Name	Data type	Address	Value
1	RecipeManager_Instance_0	RecipeManager		
2	result	DWORD		0
3	result1	DWORD		0
4	result2	DWORD		0
5	RecipeDefinitionName	STRING		'Recipe_1'
6	RecipeName	STRING		'test1'
7	aStrings_set	ARRAY [0..2] OF STRING		
8	aStrings_set[0]	STRING		'TRUE'
9	aStrings_set[1]	STRING		'DT#2025-5-15-11:11:11'
10	aStrings_set[2]	STRING		'abcde'
11	aStrings_get	ARRAY [0..2] OF STRING		
12	aStrings_get[0]	STRING		'TRUE'
13	aStrings_get[1]	STRING		'DT#2025-5-15-11:11:11'
14	aStrings_get[2]	STRING		'abcde'
15	EN_Manual0	BOOL		TRUE
16	EN_Manual1	BOOL		TRUE
17	EN_Manual2	BOOL		TRUE
18	R_TRIG_Instance_0	R_TRIG		
19	R_TRIG_Instance_1	R_TRIG		



18.2.18.1.26 GetRecipeVariableNamesV2

Gets the recipe variable names in the specified recipe.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetRecipeVariableNamesV2	Get the variable names of a specified recipe	Method		:= RecipeManager_Instance_0.GetRecipeVariableNamesV2(RecipeDefinitionName := , RecipeName := , aStrings := , iSize := , iStartIndex :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name
	RecipeName	Recipe name	Follows data type	-	Recipe name
	iSize	Size of the string array	Follows data type	-	Size of the string array
	iStartIndex	Start index	Follows data type	-	Start index
Input/Output	aStrings	String array of recipe variable names	Follows data type	-	Returned data
Output	GetRecipeVariableNamesV2	Return value	Follows data type	-	Refer to Fault Code.

Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
RecipeName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
iSize	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
iStartIndex	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
aStrings	String array ARRAY [*] OF STRING																			
GetRecipeVariableNamesV2	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction gets the recipe variable names in the specified recipe.

- **Program example**

  Call the CreateRecipe instruction to create a new recipe. Then, call the GetRecipeVariableNamesV2 instruction to get the recipe variable names in the specified recipe.

  ST:

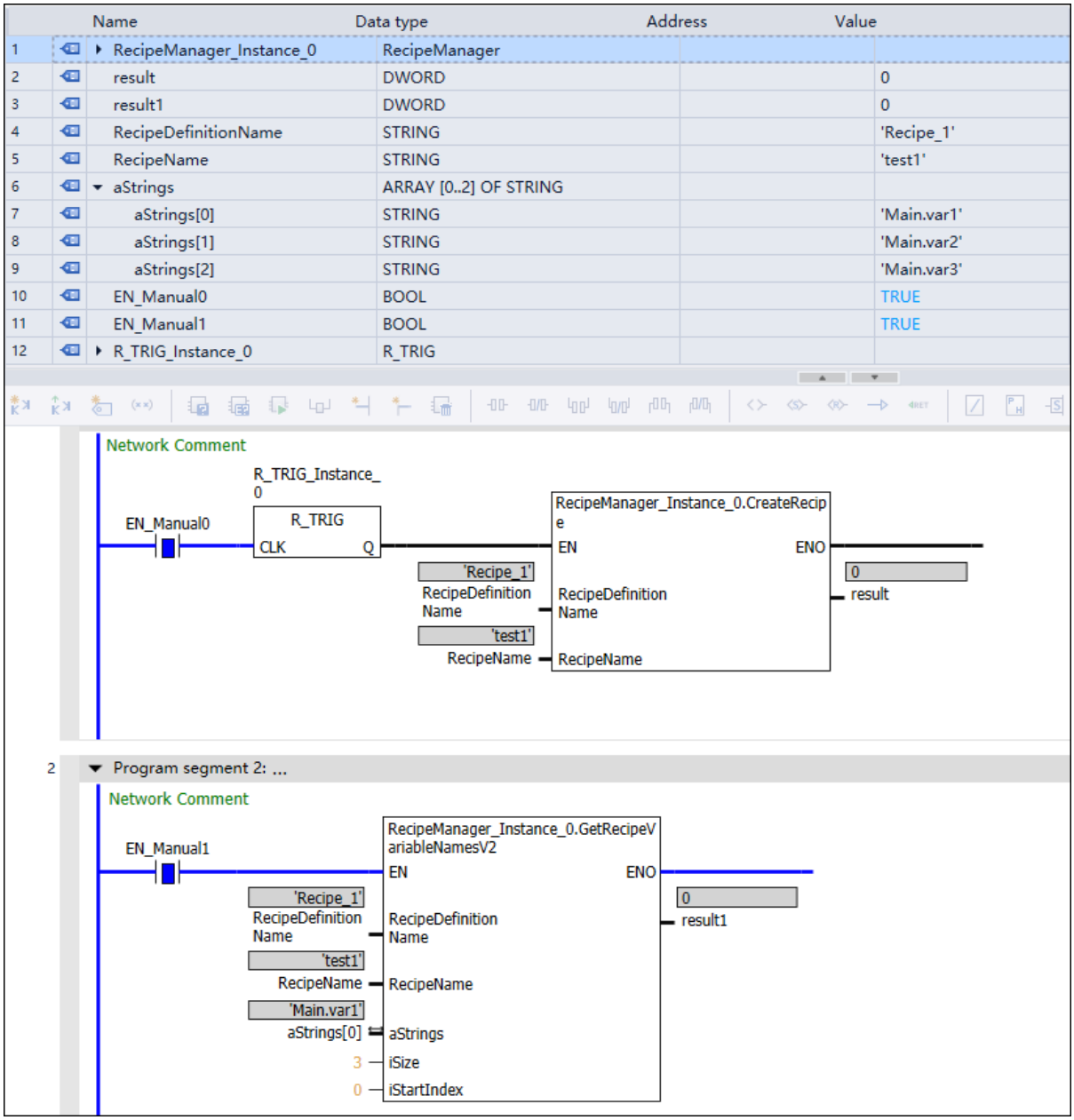
	Name	Data type	Address	Value
1	RecipeManager_Instance_0	RecipeManager		
2	result	DWORD		0
3	result1	DWORD		0
4	RecipeDefinitionName	STRING		'Recipe_1'
5	RecipeName	STRING		'test1'
6	aStrings	ARRAY [0..2] OF STRING		
7	aStrings[0]	STRING		'Main.var1'
8	aStrings[1]	STRING		'Main.var2'
9	aStrings[2]	STRING		'Main.var3'
10	EN_Manual0	BOOL		TRUE
11	EN_Manual1	BOOL		TRUE
12	R_TRIG_Instance_0	R_TRIG		

```
R_TRIG_Instance_0(
CLK  TRUE  := EN_Manual0  TRUE ,
Q => );

IF R_TRIG_Instance_0.Q  FALSE  THEN
    result      0       :=  RecipeManager_Instance_0.CreateRecipe(
    RecipeDefinitionName := RecipeDefinitionName    'Recipe_1'    ,
    RecipeName := RecipeName     'test1'     );
END_IF

IF EN_Manual1  TRUE  THEN
    result1      0       :=  RecipeManager_Instance_0.GetRecipeVariableNamesV2(
    RecipeDefinitionName := RecipeDefinitionName    'Recipe_1'    ,
    RecipeName := RecipeName     'test1'     ,
    aStrings := aStrings[0]   'Main.var1'    ,
    iSize := 3,
    iStartIndex := 0);
END_IFRETURN
```

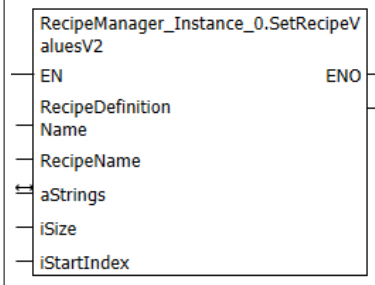
  LD:



18.2.18.1.27 SetRecipeValuesV2

Sets the recipe variable values in the specified recipe.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SetRecipeValuesV2	Set the variable values of a specified recipe	Method		:= RecipeManager_Instance_0.SetRecipeValuesV2(RecipeDefinitionName := , RecipeName := , aStrings := , iSize := , iStartIndex :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	RecipeDefinitionName	Recipe definition name	Follows data type	-	Recipe definition name
	RecipeName	Recipe name	Follows data type	-	Recipe name
	iSize	Size of the string array	Follows data type	-	Size of the string array
	iStartIndex	Start index	Follows data type	-	Start index
Input/Output	aStrings	String array of recipe values	Follows data type	-	Returned data
Output	SetRecipeValuesV2	Return value	Follows data type	-	Refer to Fault Code.

Data Type

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
RecipeDefinitionName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√
RecipeName	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	√

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
iSize	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
iStartIndex	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
aStrings	String array ARRAY [*] OF STRING																			
SetRecipeValuesV2	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction sets the recipe variable values in the specified recipe.

- **Program example**

  Call the CreateRecipe instruction to generate create a new recipe. Then, call the SetRecipeValuesV2 instruction to set the values of recipe variables in the specified recipe. Finally, call the GetRecipeValuesV2 instruction to get the recipe variable values in the specified recipe.

  ST:

	Name	Data type	Address	Value
1	RecipeManager_Instance_0	RecipeManager		
2	result	DWORD		0
3	result1	DWORD		0
4	result2	DWORD		0
5	RecipeDefinitionName	STRING		'Recipe_1'
6	RecipeName	STRING		'test1'
7	aStrings_set	ARRAY [0..2] OF STRING		
8	aStrings_set[0]	STRING		'TRUE'
9	aStrings_set[1]	STRING		'DT#2025-5-15-11:11:11'
10	aStrings_set[2]	STRING		'abcde'
11	aStrings_get	ARRAY [0..2] OF STRING		
12	aStrings_get[0]	STRING		'TRUE'
13	aStrings_get[1]	STRING		'DT#2025-5-15-11:11:11'
14	aStrings_get[2]	STRING		'abcde'
15	EN_Manual0	BOOL		TRUE
16	EN_Manual1	BOOL		TRUE
17	EN_Manual2	BOOL		TRUE
18	R_TRIG_Instance_0	R_TRIG		
19	R_TRIG_Instance_1	R_TRIG		

```
R_TRIG_Instance_0(
CLK  TRUE  := EN_Manual0  TRUE ,
Q => );

IF R_TRIG_Instance_0.Q  FALSE  THEN
    result      0       :=  RecipeManager_Instance_0.CreateRecipe(
    RecipeDefinitionName := RecipeDefinitionName   'Recipe_1'   ,
    RecipeName := RecipeName     'test1'     );
END_IF

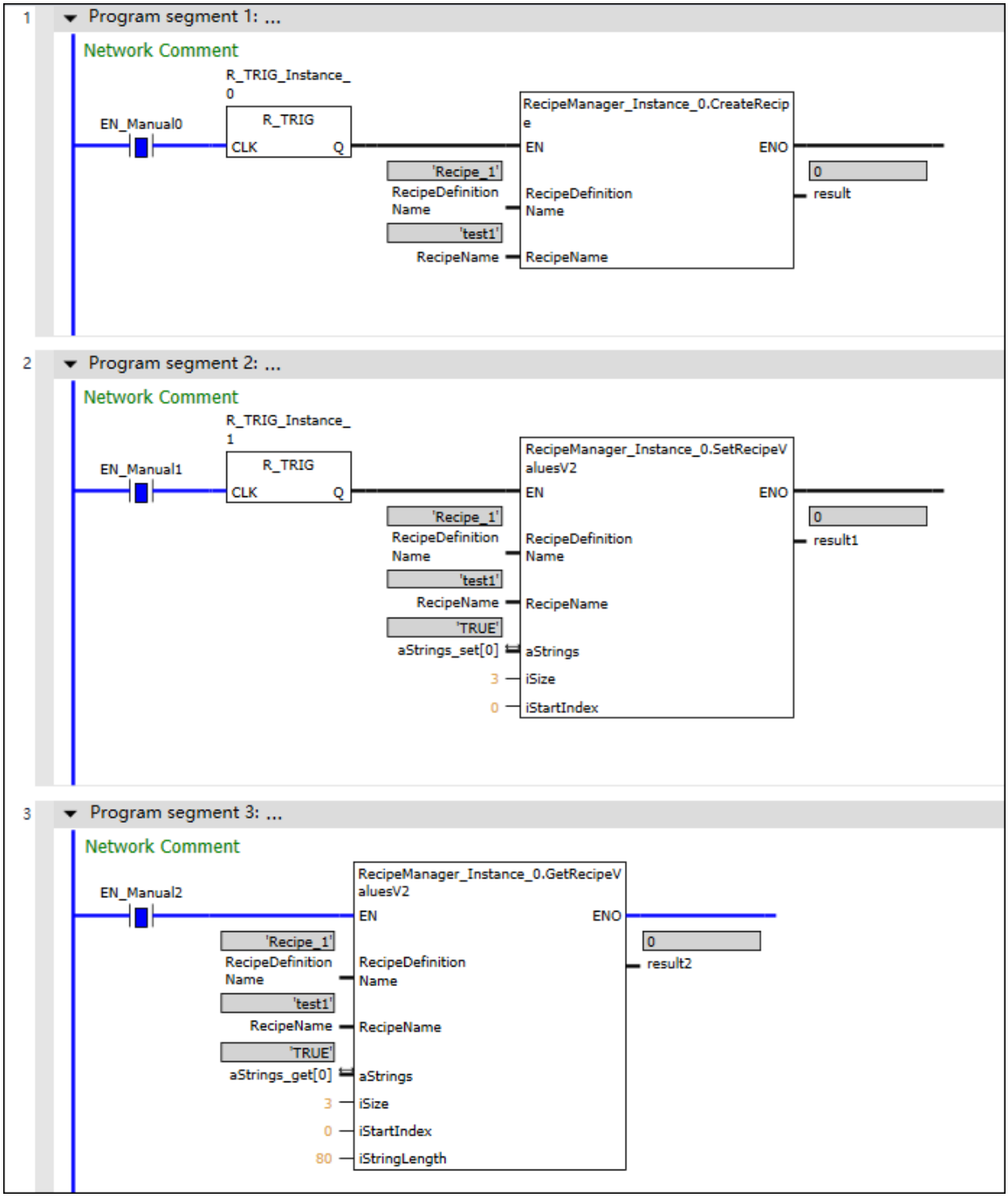
R_TRIG_Instance_1(
CLK  TRUE  := EN_Manual1  TRUE ,
Q => );

IF R_TRIG_Instance_1.Q  FALSE  THEN
    result1      0       :=  RecipeManager_Instance_0.SetRecipeValuesV2(
    RecipeDefinitionName := RecipeDefinitionName   'Recipe_1'   ,
    RecipeName := RecipeName     'test1'     ,
    aStrings := aStrings_set[0]     'TRUE'      ,
    iSize := 3,
    iStartIndex := 0);
END_IF

IF EN_Manual2  TRUE  THEN
    result2      0       :=  RecipeManager_Instance_0.GetRecipeValuesV2(
    RecipeDefinitionName := RecipeDefinitionName   'Recipe_1'   ,
    RecipeName := RecipeName     'test1'     ,
    aStrings := aStrings_get[0]     'TRUE'      ,
    iSize := 3,
    iStartIndex := 0,
    iStringLength := 80);
END_IFRETURN
```

LD:

		Name	Data type	Address	Value
1		▸ RecipeManager_Instance_0	RecipeManager		
2		result	DWORD		0
3		result1	DWORD		0
4		result2	DWORD		0
5		RecipeDefinitionName	STRING		'Recipe_1'
6		RecipeName	STRING		'test1'
7		▾ aStrings_set	ARRAY [0..2] OF STRING		
8		aStrings_set[0]	STRING		'TRUE'
9		aStrings_set[1]	STRING		'DT#2025-5-15-11:11:11'
10		aStrings_set[2]	STRING		'abcde'
11		▾ aStrings_get	ARRAY [0..2] OF STRING		
12		aStrings_get[0]	STRING		'TRUE'
13		aStrings_get[1]	STRING		'DT#2025-5-15-11:11:11'
14		aStrings_get[2]	STRING		'abcde'
15		EN_Manual0	BOOL		TRUE
16		EN_Manual1	BOOL		TRUE
17		EN_Manual2	BOOL		TRUE
18		▸ R_TRIG_Instance_0	R_TRIG		
19		▸ R_TRIG_Instance_1	R_TRIG		



- **Precautions**

  It is recommended to use edge-triggered signals for the enable input EN.

18.2.18.1.28 GetLastError

Gets the latest fault code after the last operation.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetLastError	Get the latest fault code	Method	RecipeManager_Instance_0.GetLastError EN ENO	:= RecipeManager_Instance_0.GetLastError();

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	GetLastError	Return value	Follows data type	-	Refer to Fault Code.

  Data Type

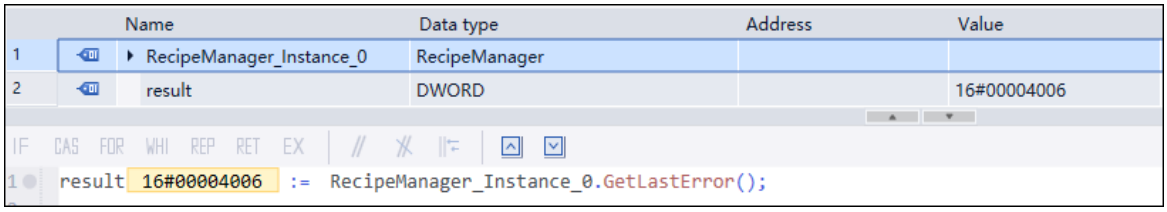
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
GetLastError	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

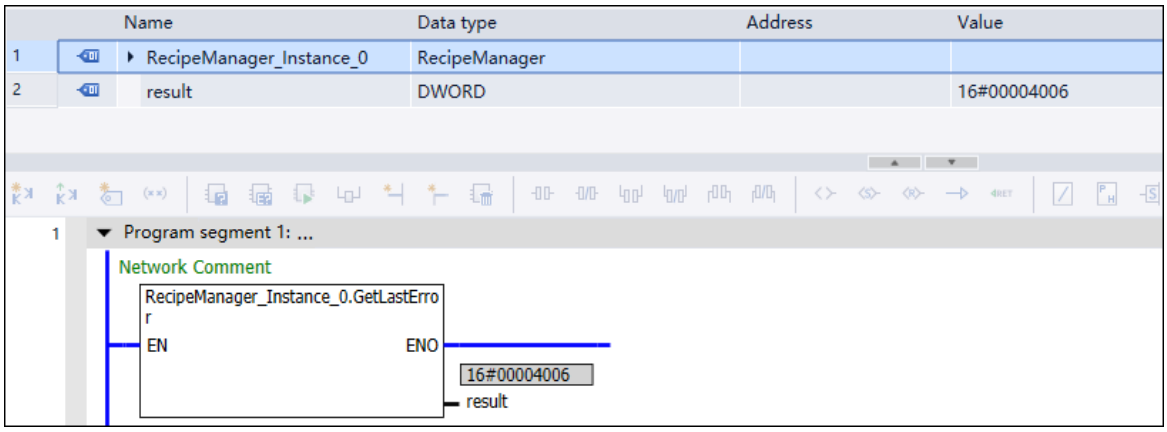
  This instruction gets the latest fault code after the last operation.

- **Program example**

  ST:



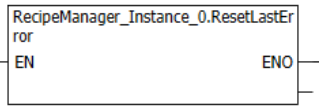
  LD:



18.2.18.1.29 ResetLastError

Resets the fault codes of the recipes.

- Instruction format

Instruction	Name	FB/FC	LD Representation	ST Representation
ResetLastError	Reset fault codes	Method		:= RecipeManager_Instance_0.ResetLastError();

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	ResetLastError	Return value	Follows data type	-	Refer to Fault Code.

  Data Type

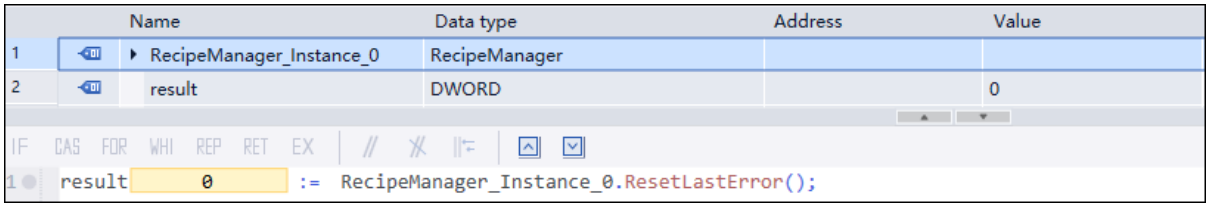
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
ResetLastError	_	_	√	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_

- **Function**

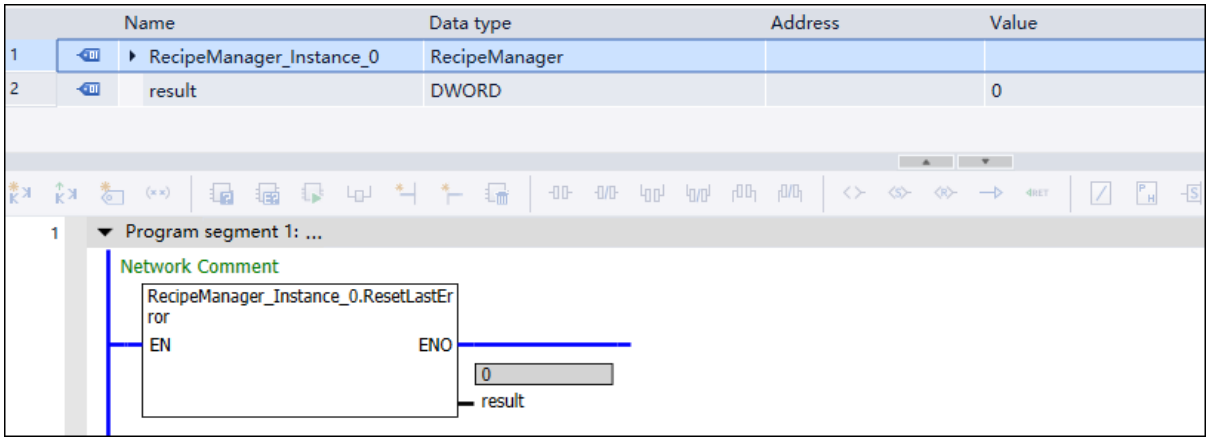
  This instruction resets the fault codes of the recipes.

- **Program example**

  ST:



  LD:



- **Precautions**

  It is recommended to use edge-triggered signals for the enable input EN.

18.2.18.1.30 SetStoragePath

Sets the storage path of the recipe file on the controller.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SetStoragePath	Set storage path	Method	RecipeManager_Instance_0.SetStoragePath EN ENO stPath	:= RecipeManager_Instance_0.SetStoragePath(stPath :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	stPath	Recipe storage path	Follows data type	-	Recipe storage path
Output	SetStoragePath	Return value	TRUE, FALSE	FALSE	Indicates whether the storage path is set successfully.

  Data Type

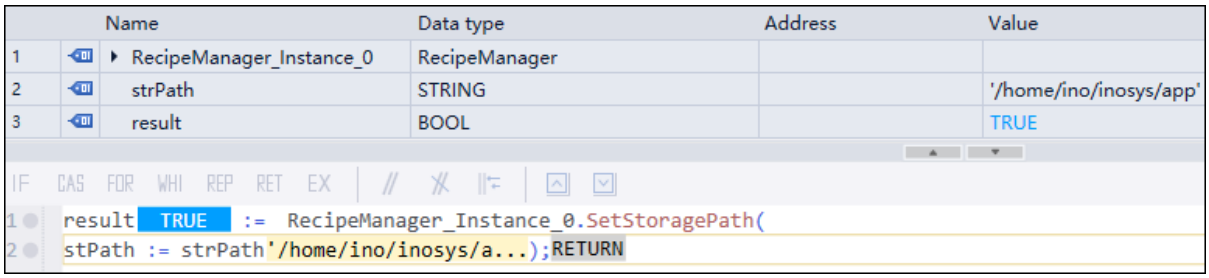
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
stPath	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
SetStoragePath	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**

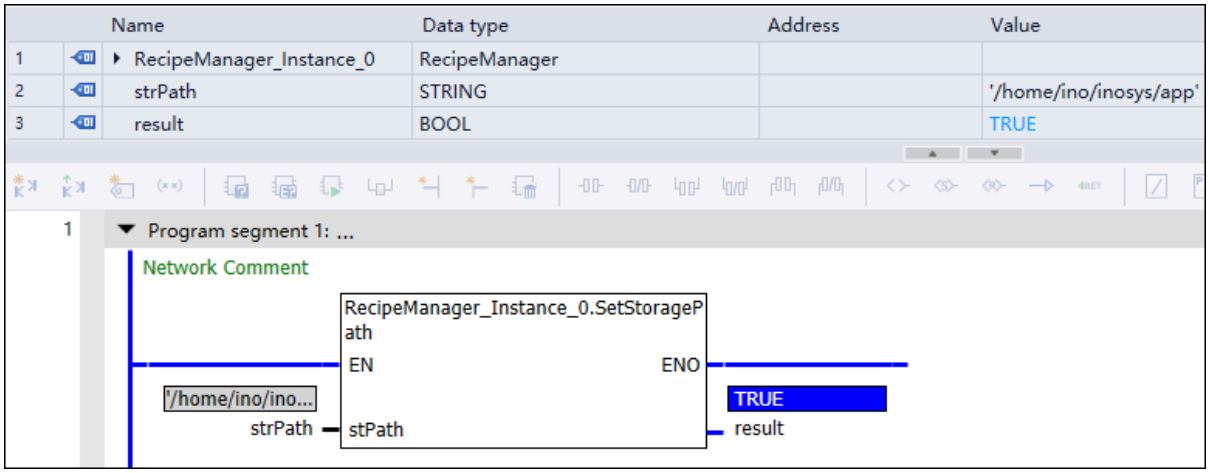
This instruction sets the storage path of the recipe file on the controller.

- **Program example**

ST:



LD:



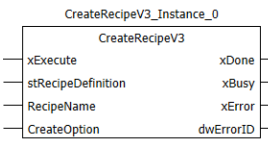
- **Precautions**

It is recommended to use edge-triggered signals for the enable input EN.

18.2.18.1.31 CreateRecipeV3

Creates recipe data and a file with the specified name based on the specified recipe definition.

- **Instruction format**

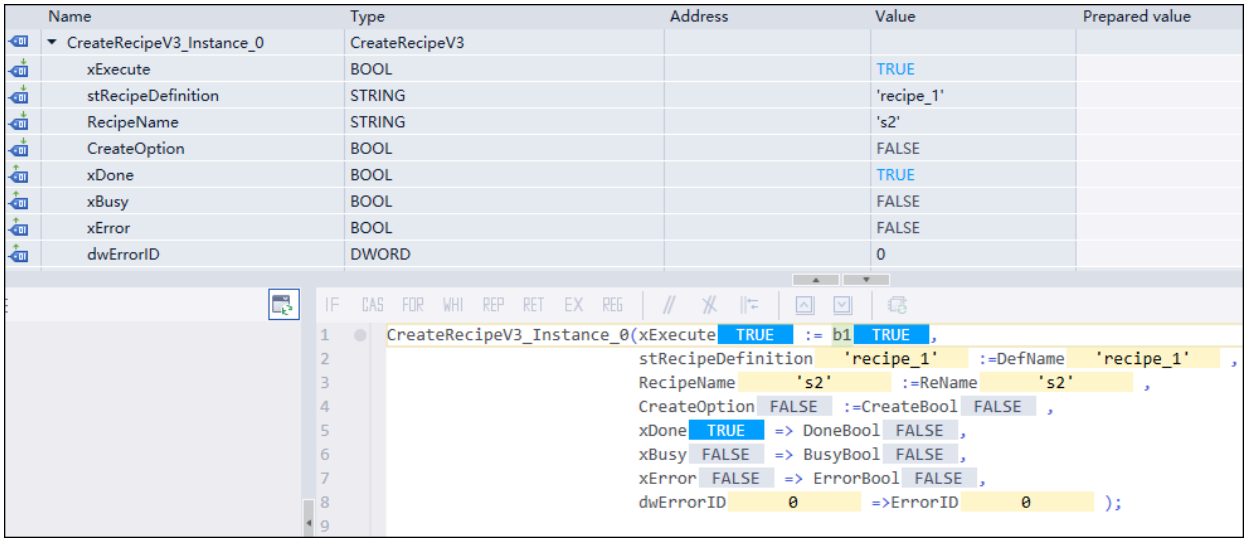
Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
CreateRecipeV3	Create a recipe	FB		CreateRecipeV3_Instance_0(xExecute := , stRecipeDefinition := , RecipeName := , CreateOption := , xDone => , xBusy => , xError => , dwErrorID =>);

- **Related variables**

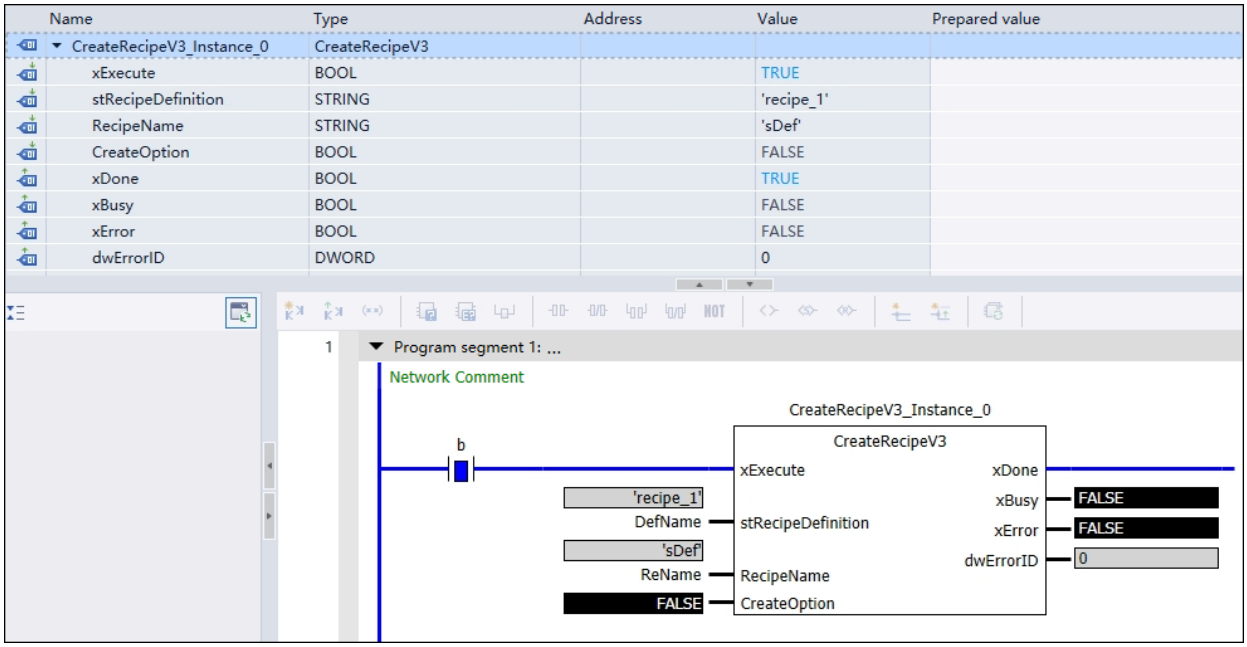
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Enable	BOOL	TRUE, FALSE	FALSE	This instruction is triggered by the rising edge of xExecute.
	stRecipeDefinition	Recipe definition name	STRING	Follows data type	0	Specifies the recipe definition
	RecipeName	Recipe name	STRING	Follows data type	-	Specifies the recipe
	CreateOption	Creation mode	BOOL	TRUE, FALSE	FALSE	0: Creates recipe and file 1: Creates only temporary recipe
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is completed.
	xBusy	Instruction in use	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is currently being executed.
	xError	Instruction fault flag	BOOL	TRUE, FALSE	FALSE	Indicates whether a fault occurred during instruction execution.
	dwErrorID	Instruction fault ID	DWORD	Depends on data type	0	Indicates the cause of the instruction faults. For details, see Fault Codes.

- **Function description**
  When the input parameter xExecute changes from FALSE to TRUE, the instruction is enabled and can be used to create a recipe after certain conditions are met.
- **Program example**
  Input the name of an existing recipe definition and the name of the recipe to be created, and change the value of CreateOption to determine whether to create a recipe file.

  ST:



LD:

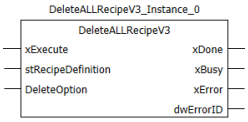


- **Precautions**
  - If a temporary recipe with the same name already exists, it cannot be created again. The instruction will return a fault message indicating the recipe already exists.
  - A recipe cannot be created under a recipe definition that does not exist. The instruction will return a fault message indicating the recipe definition not found.
  - The created recipe file is saved to the path specified by the SetStoragePathV3 instruction. If an old recipe file with the same recipe definition and recipe name already exists at that path, the existing file will be overwritten.

18.2.18.1.32 DeleteALLRecipeV3

Deletes all recipes under the specified recipe definition.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
DeleteALLRecipeV3	Deletes all recipes under a recipe definition	FB		DeleteALLRecipeV3_Instance_0(xExecute := , stRecipeDefinition := , DeleteOption := , xDone => , xBusy => , xError => , dwErrorID =>);

- **Related variables**

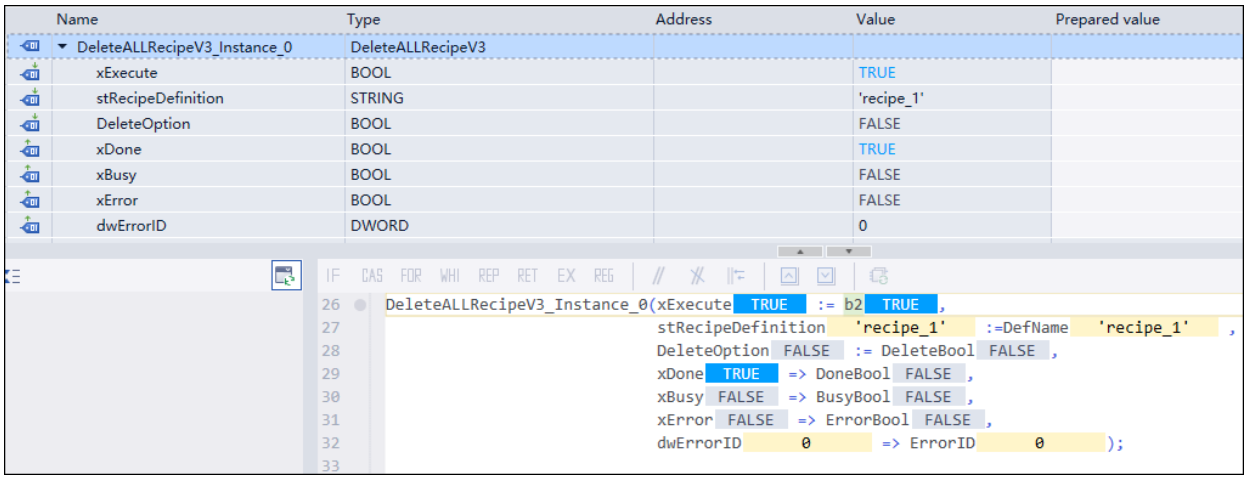
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Enable	BOOL	TRUE, FALSE	FALSE	This instruction is triggered by the rising edge of xExecute.
	stRecipeDefinition	Recipe definition name	STRING	Follows data type	0	Specifies the recipe definition
	DeleteOption	Deletion mode	BOOL	TRUE, FALSE	FALSE	0: Deletes recipe and file. 1: Deletes only temporary recipe in memory.
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is completed.
	xBusy	Instruction in use	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is currently being executed.
	xError	Instruction fault flag	BOOL	TRUE, FALSE	FALSE	Indicates whether a fault occurred during instruction execution.
	dwErrorID	Instruction fault ID	DWORD	Depends on data type	0	Indicates the cause of the instruction faults. For details, see Fault Codes.

- **Function description**

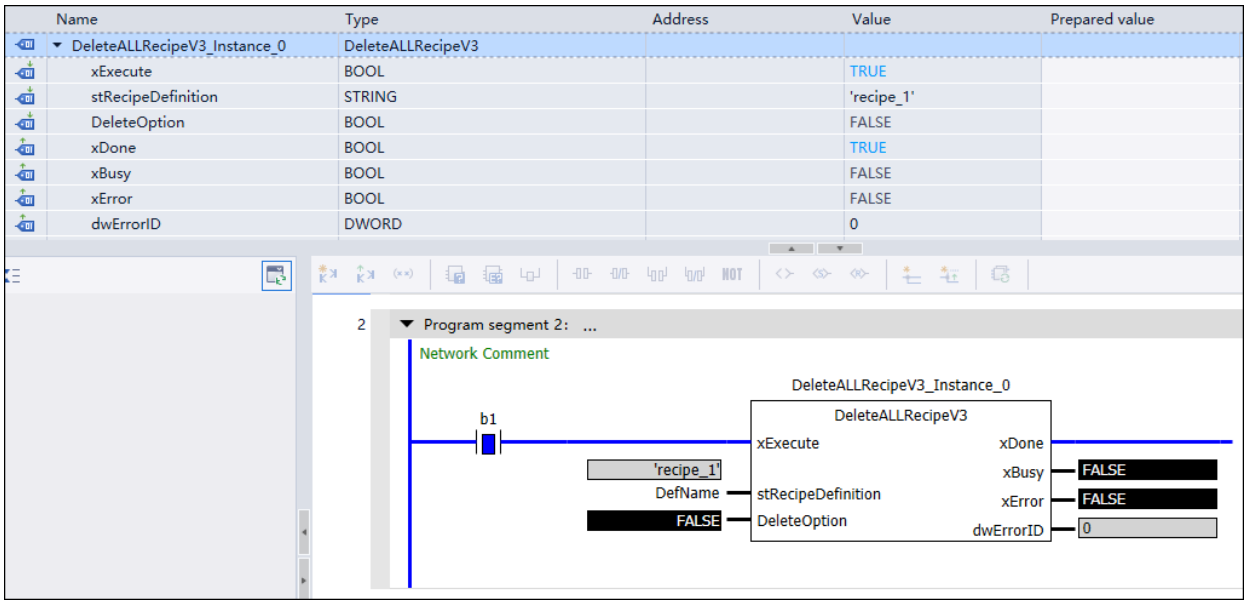
  When the input parameter xExecute changes from FALSE to TRUE, the instruction is enabled and can be used to delete recipes after certain conditions are met.

- **Program example**
  Input the name of an existing recipe definition, and change the value of DeleteOption to determine whether to delete recipe files.

  ST:



  LD:

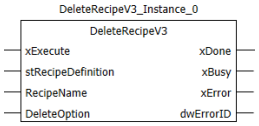


- **Precautions**
  To delete a file, the recipe file must be loaded as a temporary recipe. After the instruction is executed, the system searches for the recipe file under the path specified by the SetStoragePathV3 instruction. If the file is found, both the temporary recipe and the recipe file are deleted. If the file is not found, only the temporary recipe is deleted.

18.2.18.1.33 DeleteRecipeV3

Deletes recipe data and a file with the specified name based on the specified recipe definition.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
DeleteRecipeV3	Delete a specified recipe under a recipe definition	FB		DeleteRecipeV3_Instance_0(xExecute := , stRecipeDefinition := , RecipeName := , DeleteOption := , xDone => , xBusy => , xError => , dwErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Enable	BOOL	TRUE, FALSE	FALSE	This instruction is triggered by the rising edge of xExecute.
	stRecipeDefinition	Recipe definition name	STRING	Follows data type	0	Specifies the recipe definition
	RecipeName	Recipe name	STRING	Follows data type	0	Specifies the recipe
	DeleteOption	Deletion mode	BOOL	TRUE, FALSE	FALSE	0: Deletes recipe and file. 1: Deletes only temporary recipe in memory.
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is completed.
	xBusy	Instruction in use	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is currently being executed.
	xError	Instruction fault flag	BOOL	TRUE, FALSE	FALSE	Indicates whether a fault occurred during instruction execution.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	dwErrorID	Instruction fault ID	DWORD	Depends on data type	0	Indicates the cause of the instruction faults. For details, see Fault Codes.

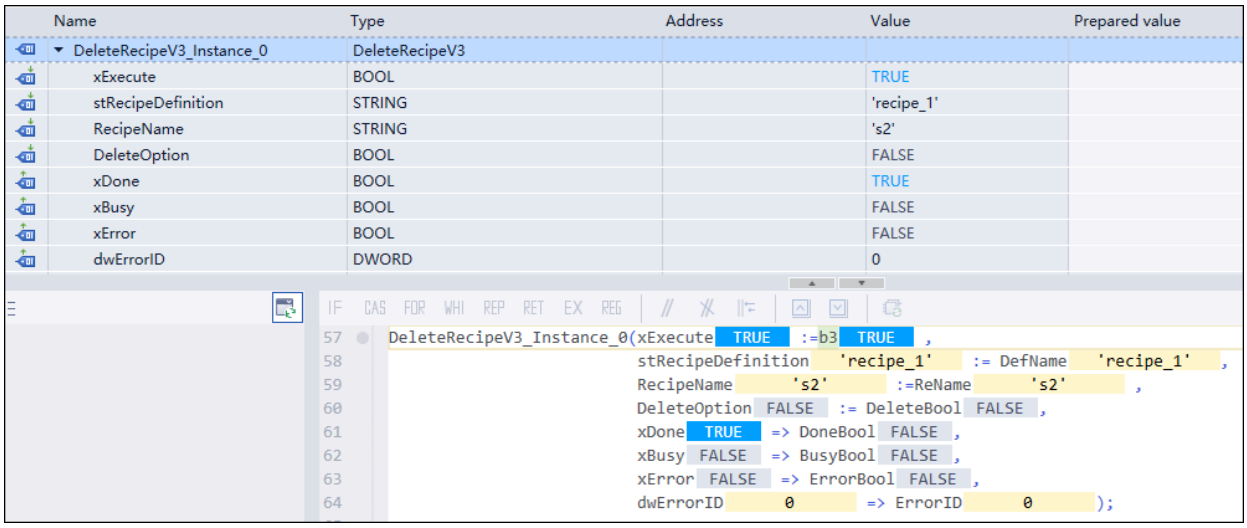
- **Function description**

  When the input parameter xExecute changes from FALSE to TRUE, the instruction is enabled and can be used to delete a recipe after certain conditions are met.

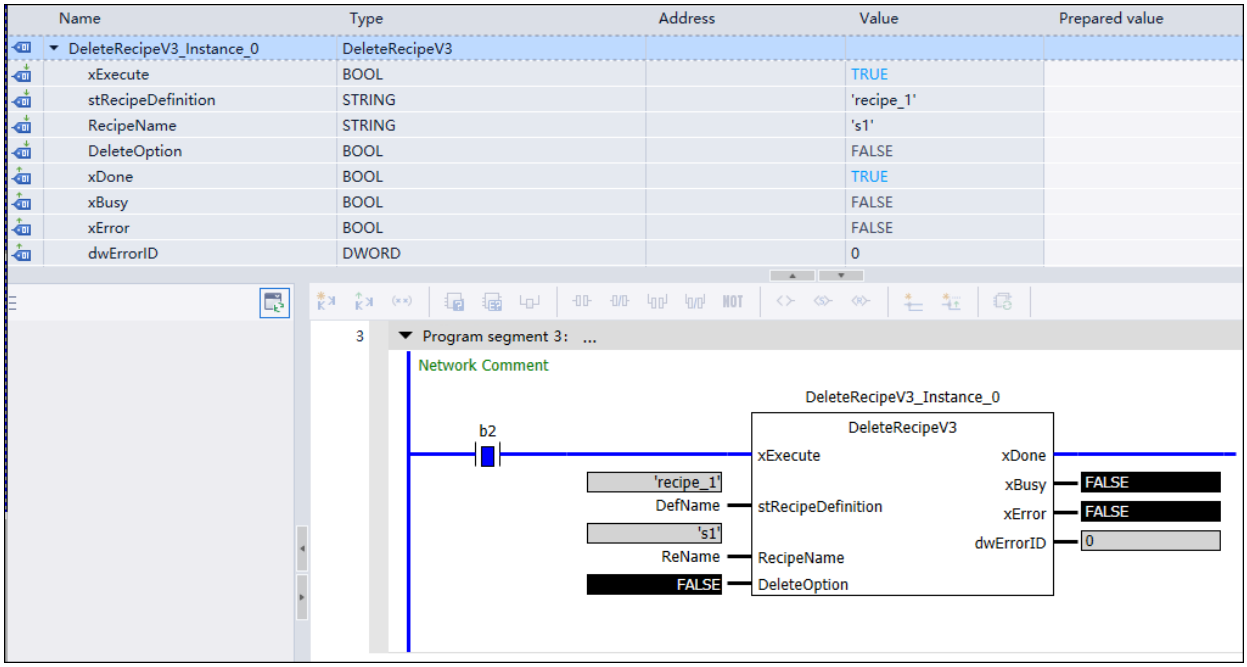
- **Program example**

  Input the name of an existing recipe definition and the name of a recipe that has been created under that recipe definition. Change the value of CreateOption to determine whether to delete the specified recipe file.

  ST:



  LD:



- **Precautions**

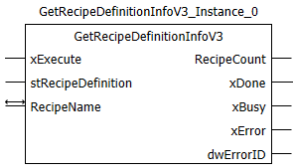
  If you determine to delete recipe and file, the system will search for recipe files under the path specified

by the SetStoragePathV3 instruction. If the recipe file is detected, both the temporary recipe and the recipe file will be deleted; if the file is not detected, only the temporary recipe will be deleted.

18.2.18.1.34 GetRecipeDefinitionInfoV3

Gets the information about the specified recipe definition, including the number of recipes and recipe names.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
GetRecipeDefinitionInfoV3	Get recipe definition information	FB		GetRecipeDefinitionInfoV3_Instance_0(xExecute := , stRecipeDefinition := , RecipeCount => , xDone => , xBusy => , xError => , dwErrorID => , RecipeName :=);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Enable	BOOL	TRUE, FALSE	FALSE	This instruction is triggered by the rising edge of xExecute.
	stRecipeDefinition	Recipe definition name	STRING	Follows data type	0	Specifies the recipe definition
Output	RecipeCount	Recipe count	DWORD	Depends on data type	0	Indicates the number of recipes retrieved
	RecipeName	Recipe name	ARRAY [*] OF STRING	Follows data type	0	An array containing the names of all retrieved recipes
	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is completed.
	xBusy	Instruction in use	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is currently being executed.

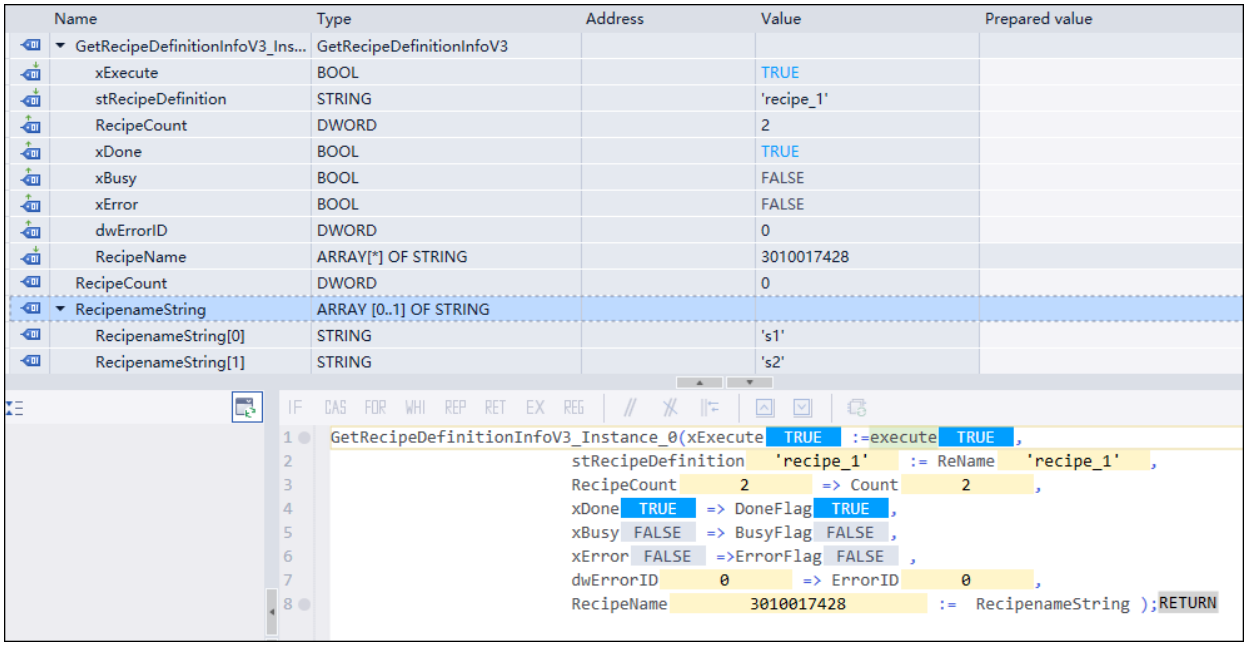
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	xError	Instruction fault flag	BOOL	TRUE, FALSE	FALSE	Indicates whether a fault occurred during instruction execution.
	dwErrorID	Instruction fault ID	DWORD	Depends on data type	0	Indicates the cause of the instruction faults. For details, see Fault Codes.

- **Function description**

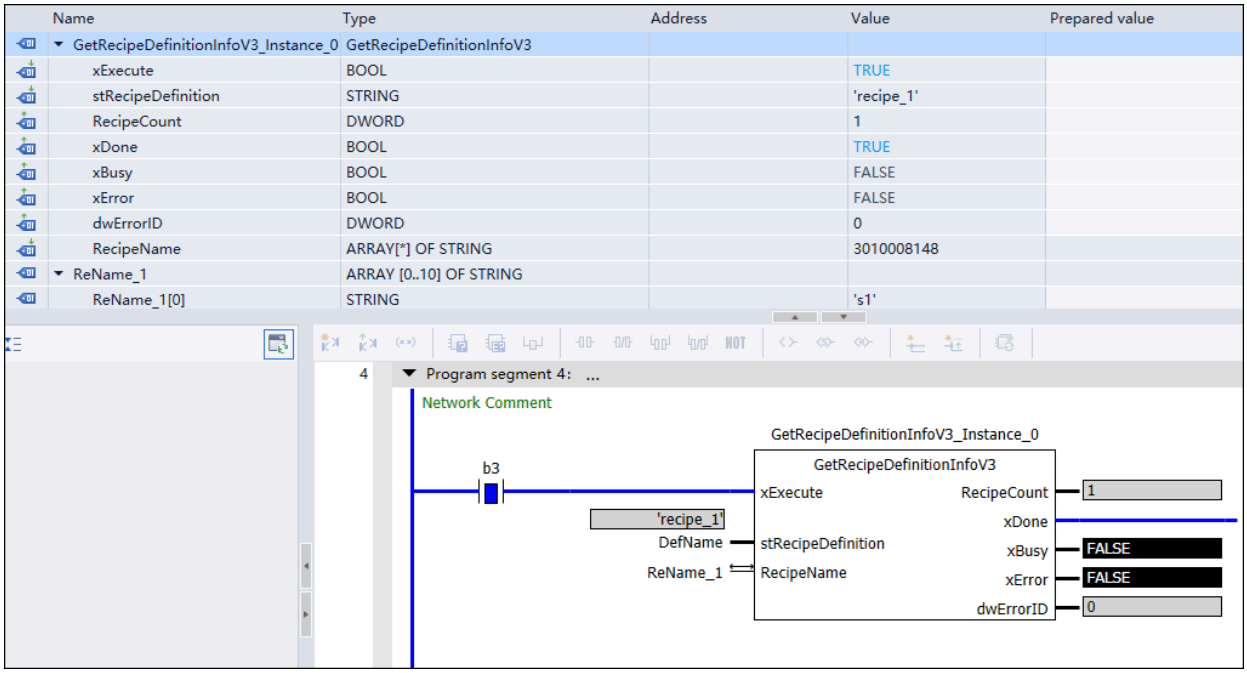
  When the input parameter xExecute changes from FALSE to TRUE, the instruction is enabled and gets the information about the specified recipe definition after certain conditions are met, including the number of recipes under this recipe definition and their names.

- **Program example**

  ST:



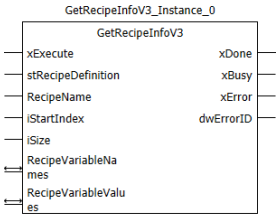
  LD:



18.2.18.1.35 GetRecipeInfoV3

Gets the variable names and values of the specified recipe.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
GetRecipeInfoV3	Get recipe information	FB		GetRecipeInfoV3_Instance_0(xExecute := , stRecipeDefinition := , RecipeName := , iStartIndex := , iSize := , xDone => , xBusy => , xError => , dwErrorID => , RecipeVariableNames := , RecipeVariableValues :=);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Enable	BOOL	TRUE, FALSE	FALSE	This instruction is triggered by the rising edge of xExecute.
	stRecipeDefinition	Recipe definition name	STRING	Follows data type	0	Specifies the recipe definition

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	RecipeName	Recipe name	STRING	Follows data type	0	Specifies the recipe
	iSize	Size of the string array	UINT	Follows data type	0	Size of the string array that stores recipe variable names and values
	iStartIndex	Start index	UINT	Follows data type	0	Start index for retrieving recipe variables
Output	RecipeVariableNames	Recipe variable name	ARRAY [*] OF STRING	Follows data type	0	Outputs recipe variable names of the specified length
	RecipeVariableValues	Recipe variable value	ARRAY [*] OF STRING	Follows data type	0	Outputs recipe variable values of the specified length
	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is completed.
	xBusy	Instruction in use	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is currently being executed.
	xError	Instruction fault flag	BOOL	TRUE, FALSE	FALSE	Indicates whether a fault occurred during instruction execution.
	dwErrorID	Instruction fault ID	DWORD	Depends on data type	0	Indicates the cause of the instruction faults. For details, see Fault Codes.

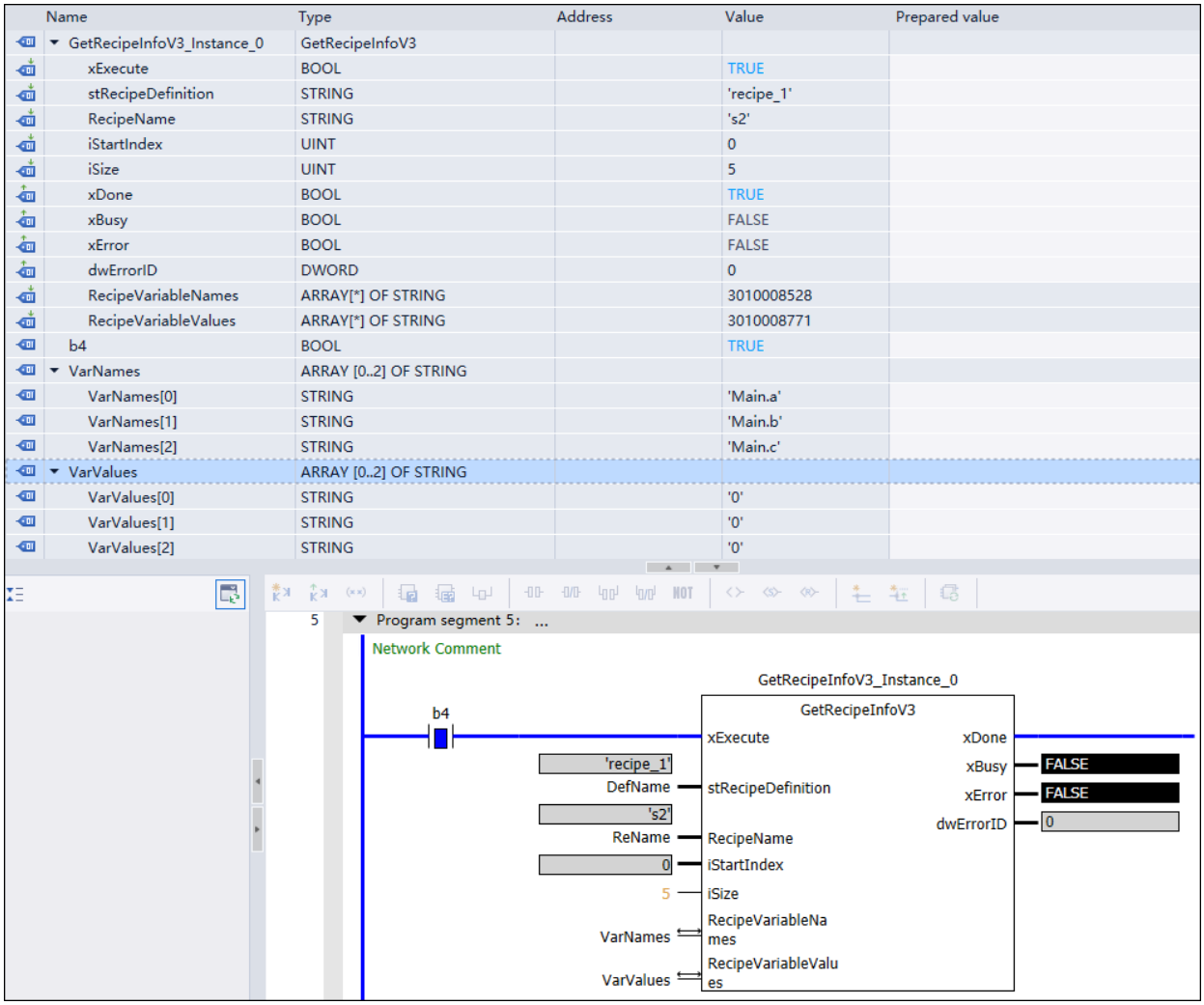
- **Function description**
  When the input parameter xExecute changes from FALSE to TRUE, the instruction is enabled. After certain conditions are met, the instruction gets the variable names and values for the specified recipe, based on the specified length (iSize) and start index (iStartIndex).
- **Program example**

  ST:

Name	Type	Address	Value	Prepared value
GetRecipeInfoV3_Instance_0	GetRecipeInfoV3			
xExecute	BOOL		TRUE	
stRecipeDefinition	STRING		'recipe_1'	
RecipeName	STRING		's1'	
iStartIndex	UINT		0	
iSize	UINT		3	
xDone	BOOL		TRUE	
xBusy	BOOL		FALSE	
xError	BOOL		FALSE	
dwErrorID	DWORD		0	
RecipeVariableNames	ARRAY[*] OF STRING		3010016560	
RecipeVariableValues	ARRAY[*] OF STRING		3010016803	
VarNameString	ARRAY [0..2] OF STRING			
VarNameString[0]	STRING		'Main.a'	
VarNameString[1]	STRING		'Main.b'	
VarNameString[2]	STRING		'Main.c'	
VarValueString	ARRAY [0..2] OF STRING			
VarValueString[0]	STRING		'0'	
VarValueString[1]	STRING		'0'	
VarValueString[2]	STRING		'0'	

```
1  GetRecipeInfoV3_Instance_0(xExecute  TRUE  := execute  TRUE ,
2                             stRecipeDefinition  'recipe_1'  :=ReDefName  'recipe_1'  ,
3                             RecipeName  's1'  := ReName  's1'  ,
4                             iStartIndex  0  := Index  0  ,
5                             iSize  3  :=Size  3  ,
6                             xDone  TRUE  => DoneFlag  TRUE ,
7                             xBusy  FALSE  => BusyFlag  FALSE ,
8                             xError  FALSE  => ErrorFlag  FALSE ,
9                             dwErrorID  0  => ErrorID  0  ,
10                            RecipeVariableNames  3010016560  := VarNameString,
11                            RecipeVariableValues  3010016803  := VarValueString);RETURN
```

LD:

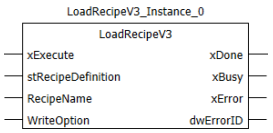


- **Precautions**
  - The input index value specifies the starting position to read from the recipe. If the index exceeds the maximum number of recipe variables, the instruction returns a fault.
  - If the sum of the index value and the array size exceeds the maximum number of recipe variables, the instruction returns only the information about the variables from the start index to the last variable in the recipe.
  - Variable indices within a recipe start from 0.

18.2.18.1.36 LoadRecipeV3

Loads the specified recipe into PLC memory and writes it to the PLC.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
LoadRecipeV3	Load a recipe	FB		LoadRecipeV3_Instance_0(xExecute := , stRecipeDefinition := , RecipeName := , WriteOption := , xDone => , xBusy => , xError => , dwErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Enable	BOOL	TRUE, FALSE	FALSE	This instruction is triggered by the rising edge of xExecute.
	stRecipeDefinition	Recipe definition name	STRING	Follows data type	0	Specifies the recipe definition
	RecipeName	Recipe name	STRING	Follows data type	0	Specifies the recipe
	WriteOption	Load mode	BOOL	TRUE, FALSE	FALSE	0: Loads without writing to PLC. 1: Loads and writes to PLC.
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is completed.
	xBusy	Instruction in use	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is currently being executed.
	xError	Instruction fault flag	BOOL	TRUE, FALSE	FALSE	Indicates whether a fault occurred during instruction execution.

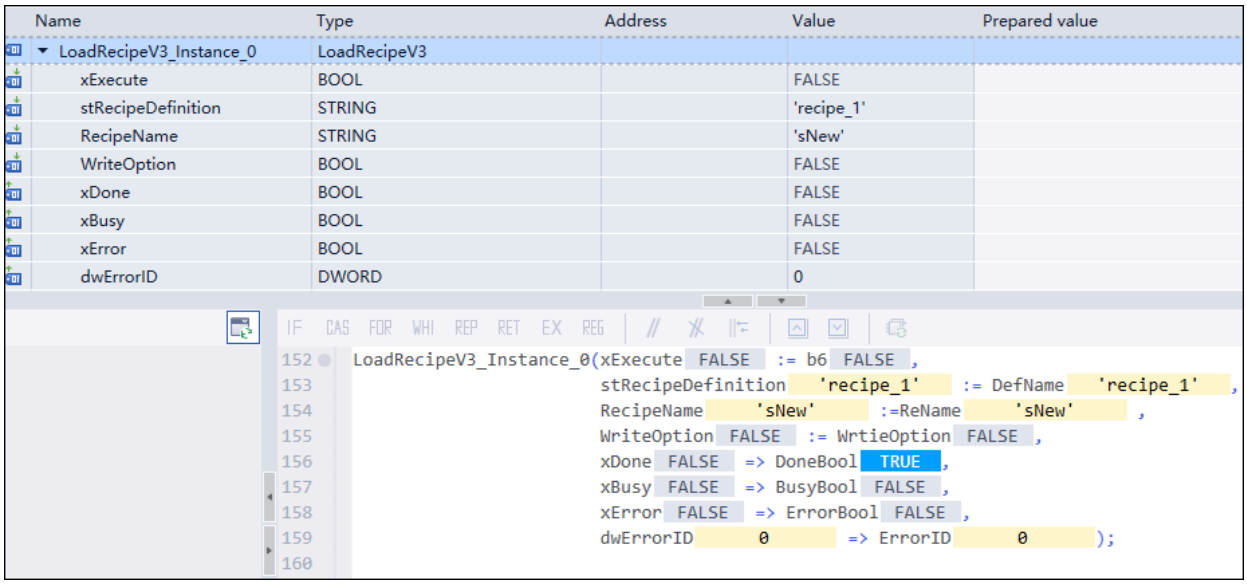
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	dwErrorID	Instruction fault ID	DWORD	Depends on data type	0	Indicates the cause of the instruction faults. For details, see Fault Codes.

- **Function description**

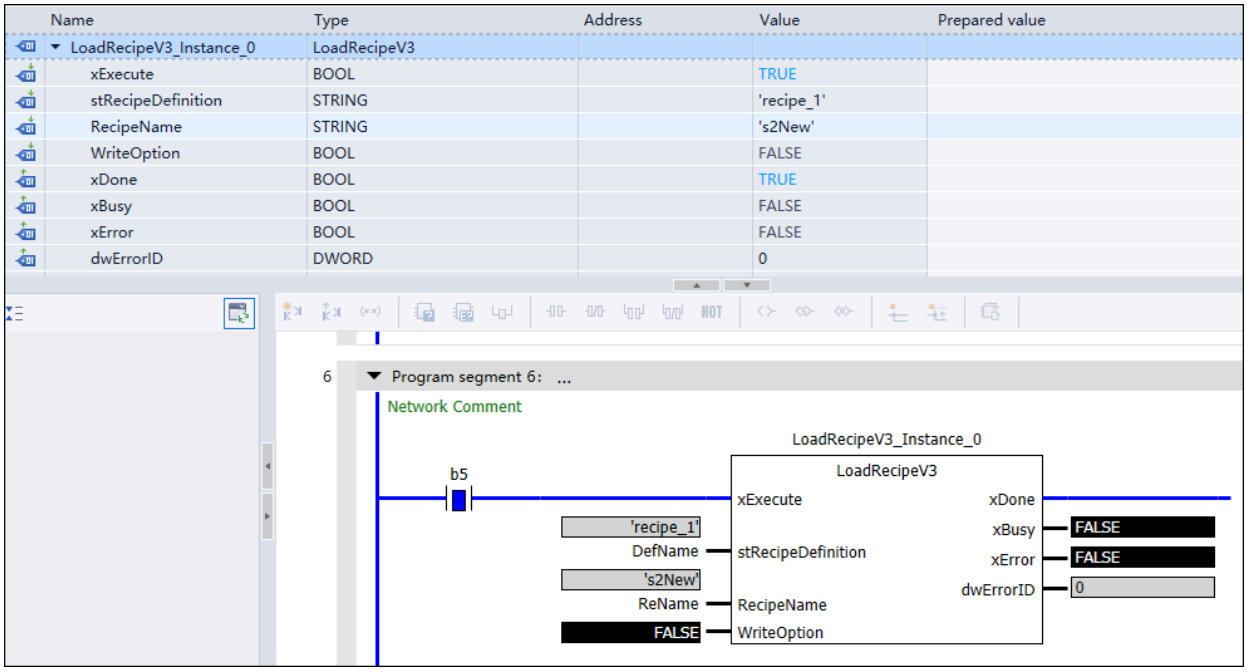
  When the input parameter xExecute changes from FALSE to TRUE, the instruction is enabled. If WriteOption is set to 0 (load only), a temporary recipe with the specified recipe file name is created; if WriteOption is set to 1 (load and write), a temporary recipe is created, and the recipe variable values are written to PLC.

- **Program example**

  ST:



  LD:



- **Precautions**

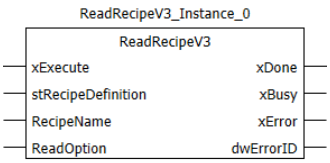
  If a temporary recipe with the specified recipe name already exists, this instruction returns a fault

message indicating the recipe already exists. You can delete the existing temporary recipe and run the LoadRecipeV3 instruction to load again.

18.2.18.1.37 ReadRecipeV3

Writes the PLC variable values into the specified recipe or recipe file.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
ReadRecipeV3	Read PLC values into a recipe	FB		ReadRecipeV3_Instance_0(xExecute := , stRecipeDefinition := , RecipeName := , ReadOption := , xDone => , xBusy => , xError => , dwErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Enable	BOOL	TRUE, FALSE	FALSE	This instruction is triggered by the rising edge of xExecute.
	stRecipeDefinition	Recipe definition name	STRING	Follows data type	0	Specifies the recipe definition
	RecipeName	Recipe name	STRING	Follows data type	0	Specifies the recipe
	ReadOption	Write mode	BYTE	0 to 2	0	0: Writes only the recipe. 1: Writes both the recipe and the recipe file. 2: Writes only the recipe file.
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is completed.
	xBusy	Instruction in use	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is currently being executed.

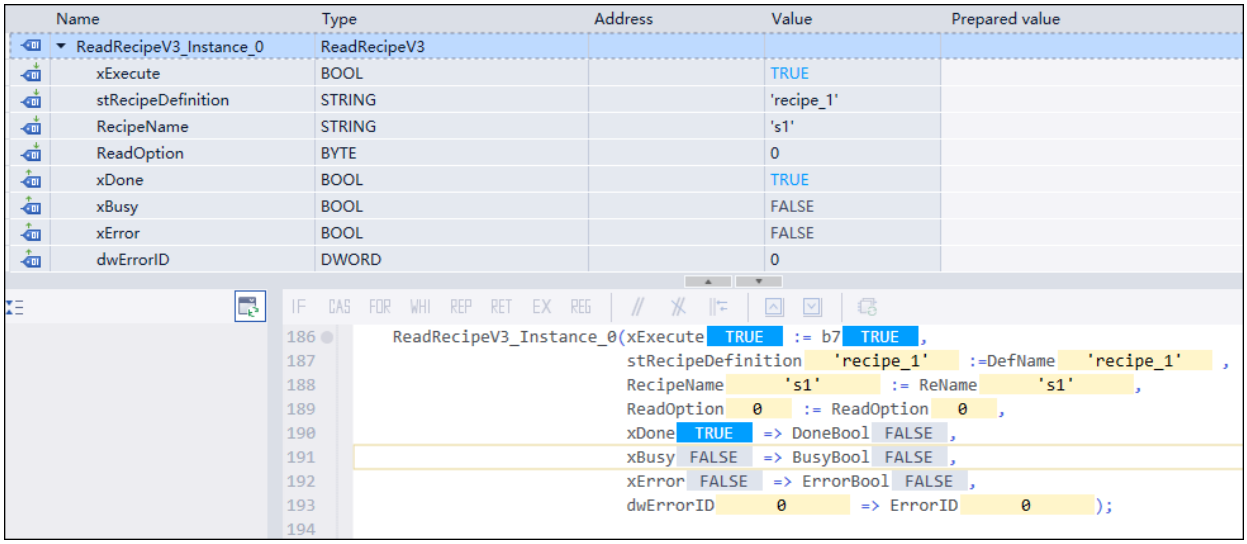
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	xError	Instruction fault flag	BOOL	TRUE, FALSE	FALSE	Indicates whether a fault occurred during instruction execution.
	dwErrorID	Instruction fault ID	DWORD	Depends on data type	0	Indicates the cause of the instruction faults. For details, see Fault Codes.

- **Function description**

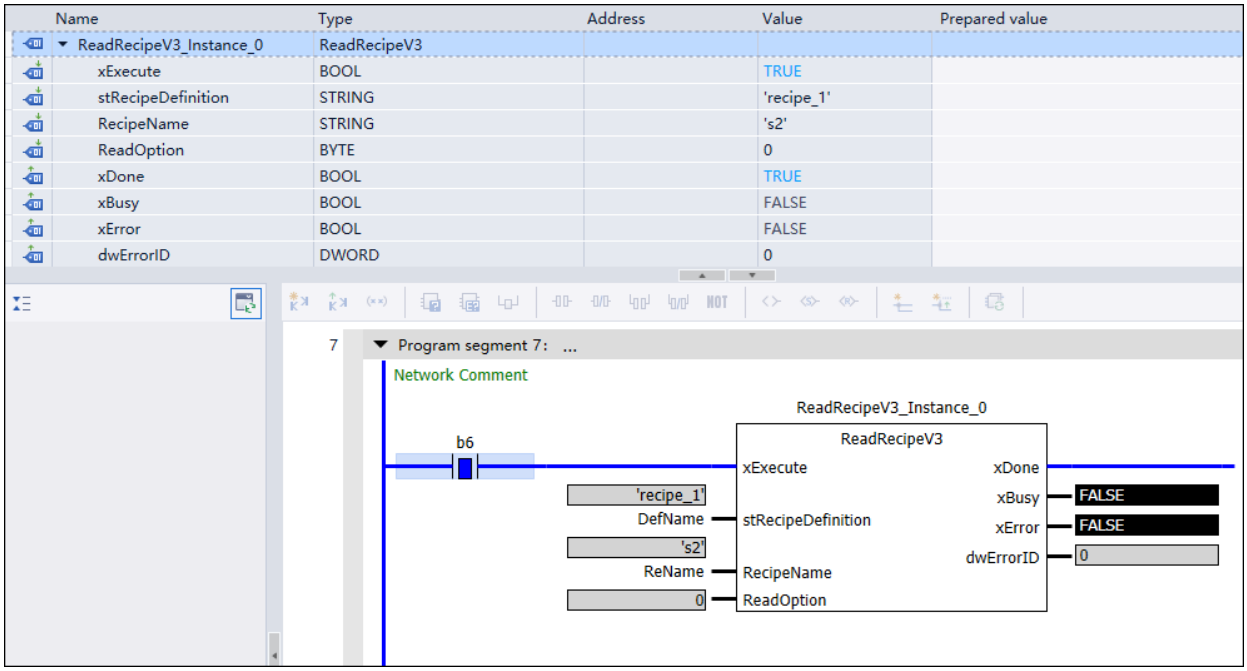
  When the input parameter xExecute changes from FALSE to TRUE, the instruction is enabled. After certain conditions are met, the specified PLC variable values can be written into the recipe based on the write mode specified by ReadOption (write only recipe, write only recipe file, or write both).

- **Program example**

  ST:



  LD:



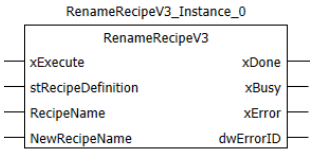
- **Precautions**

  If writing to a recipe file is selected, the instruction checks whether the file exists under the path set by the SetStoragePathV3 instruction. If the file does not exist, a fault message indicating file not found is returned. If the file exists, the recipe file is written normally.

18.2.18.1.38 RenameRecipeV3

Renames the specified recipe and recipe file.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
RenameRecipeV3	Rename a recipe	FB		RenameRecipeV3_Instance_0(xExecute := , stRecipeDefinition := , RecipeName := , NewRecipeName := , xDone => , xBusy => , xError => , dwErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Enable	BOOL	TRUE, FALSE	FALSE	This instruction is triggered by the rising edge of xExecute.

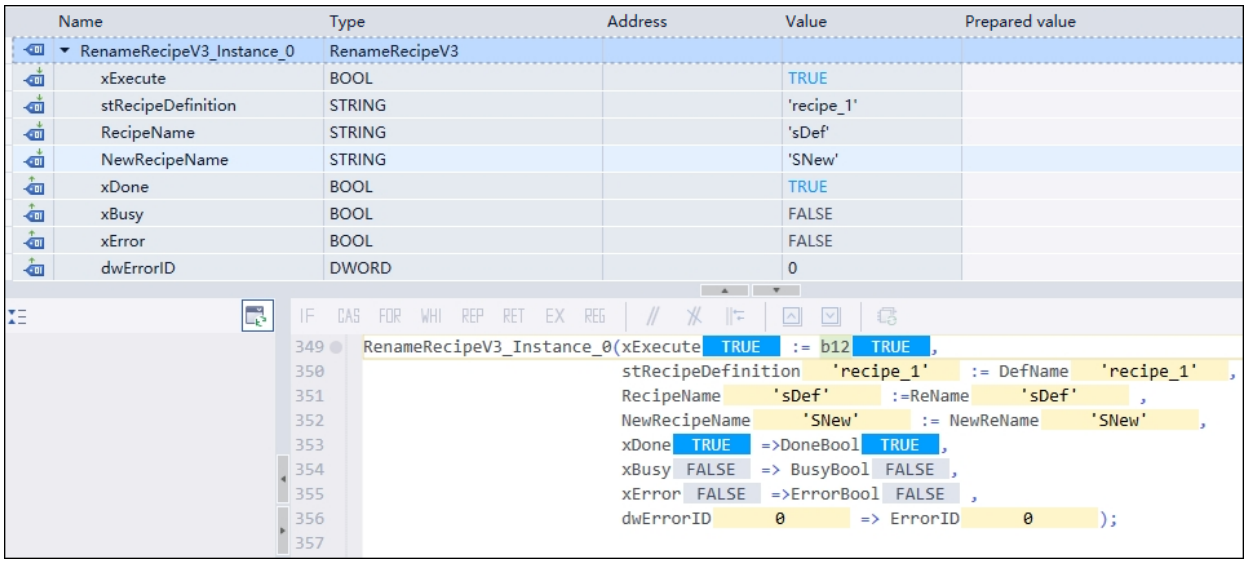
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	stRecipeDefinition	Recipe definition name	STRING	Follows data type	0	Specifies the recipe definition
	RecipeName	Recipe name	STRING	Follows data type	0	Specifies the recipe
	NewRecipeName	New recipe name	STRING	Follows data type	0	The new recipe name to be modified
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is completed.
	xBusy	Instruction in use	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is currently being executed.
	xError	Instruction fault flag	BOOL	TRUE, FALSE	FALSE	Indicates whether a fault occurred during instruction execution.
	dwErrorID	Instruction fault ID	DWORD	Depends on data type	0	Indicates the cause of the instruction faults. For details, see Fault Codes.

- **Function description**

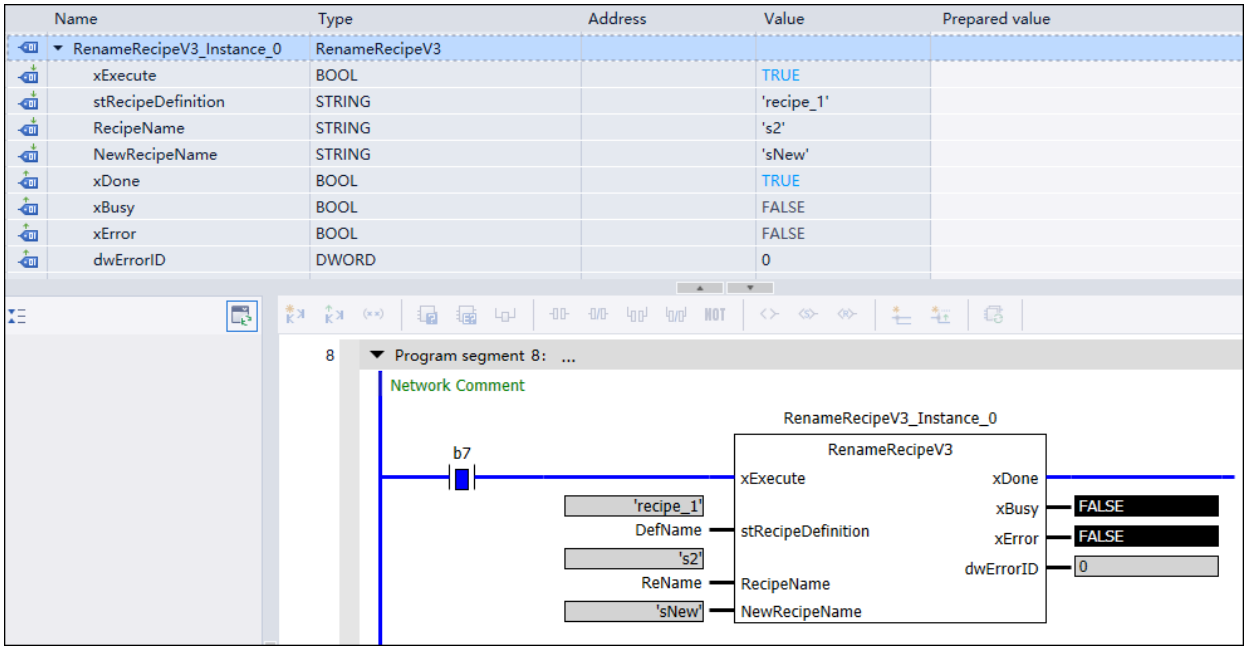
  When the input parameter xExecute changes from FALSE to TRUE, the instruction is enabled and can be used to rename a recipe and recipe file after certain conditions are met.

- **Program example**

  ST:



  LD:

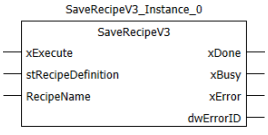


- **Precautions**
  - This instruction simultaneously renames both the temporary recipe and the recipe file. If the recipe file does not exist under the path set by SetStoragePathV3, the instruction returns a fault message indicating the file not found.
  - If a recipe file created by another project exists at the same path with the same recipe definition name and recipe name, the instruction returns a fault indicating the recipe and recipe file do not match.

18.2.18.1.39 SaveRecipeV3

Saves the recipe in memory as a recipe file.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SaveRecipeV3	Save a recipe file	FB		SaveRecipeV3_Instance_0(xExecute := , stRecipeDefinition := , RecipeName := , xDone => , xBusy => , xError => , dwErrorID =>);

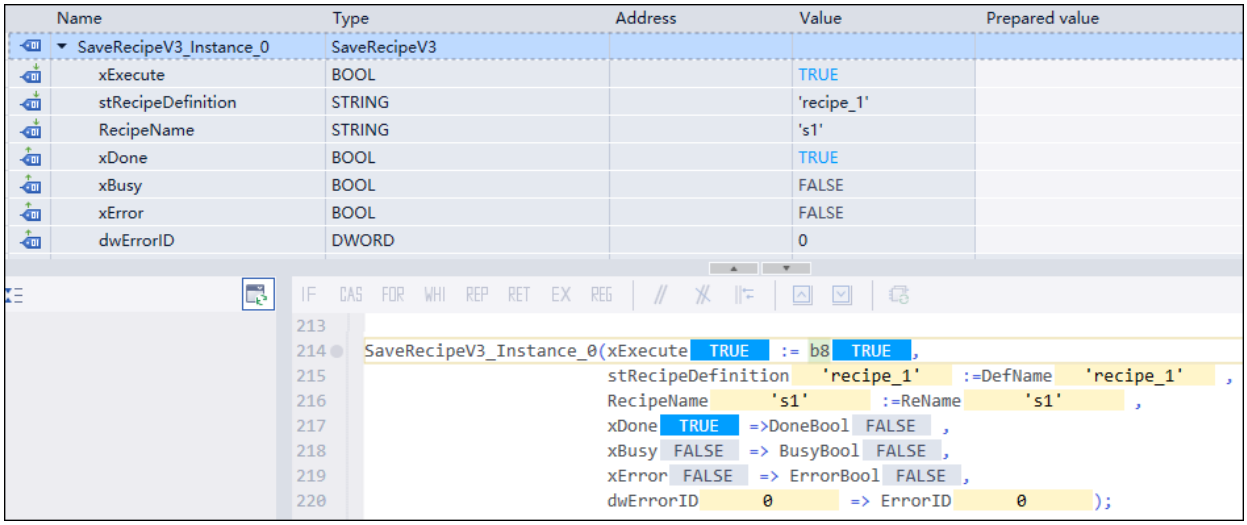
- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Enable	BOOL	TRUE, FALSE	FALSE	This instruction is triggered by the rising edge of xExecute.

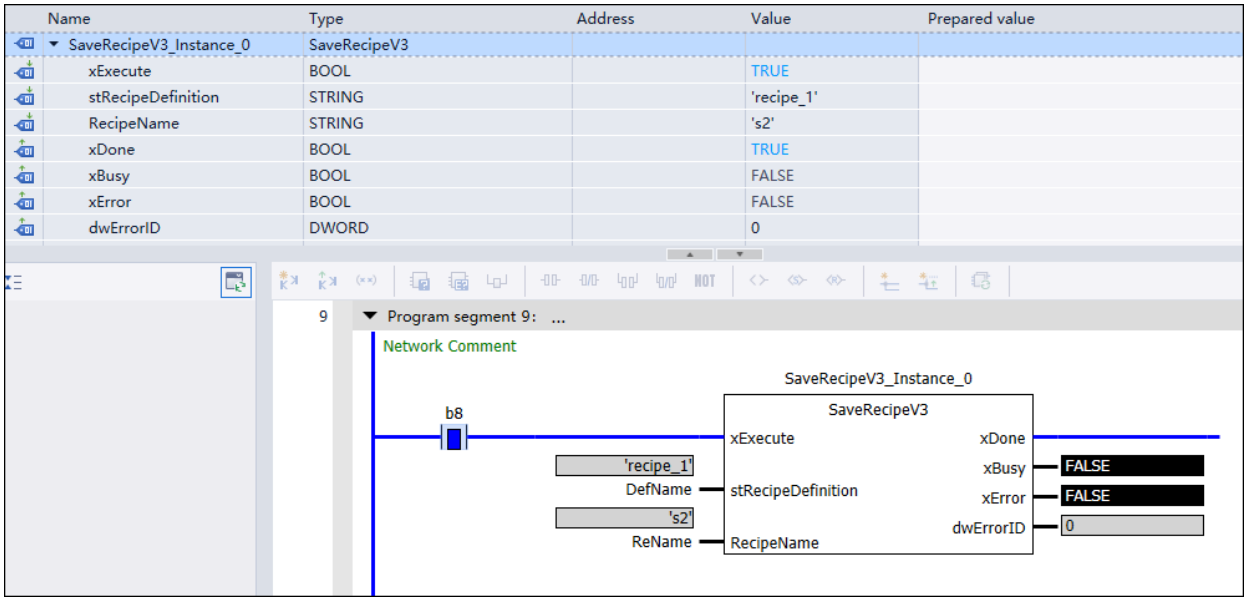
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	stRecipeDefinition	Recipe definition name	STRING	Follows data type	0	Specifies the recipe definition
	RecipeName	Recipe name	STRING	Follows data type	0	Specifies the recipe
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is completed.
	xBusy	Instruction in use	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is currently being executed.
	xError	Instruction fault flag	BOOL	TRUE, FALSE	FALSE	Indicates whether a fault occurred during instruction execution.
	dwErrorID	Instruction fault ID	DWORD	Depends on data type	0	Indicates the cause of the instruction faults. For details, see Fault Codes.

- **Function description**
  When the input parameter xExecute changes from FALSE to TRUE, the instruction is enabled and can be used to save the specified temporary recipe as a recipe file after certain conditions are met.
- **Program example**

  ST:



  LD:



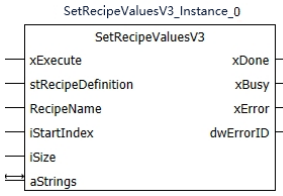
- **Precautions**

  This instruction saves the recipe file to the path set by SetStoragePathV3. If a recipe file with the same name already exists at this path, the instruction will execute normally and overwrite the existing recipe file.

18.2.18.1.40 SetRecipeValuesV3

Sets the recipe data in a recipe.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SetRecipeValuesV3	Set the variable values of a specified recipe	FB		SetRecipeValuesV3_Instance_0(xExecute := , stRecipeDefinition := , RecipeName := , iStartIndex := , iSize := , xDone => , xBusy => , xError => , dwErrorID => , aStrings :=);

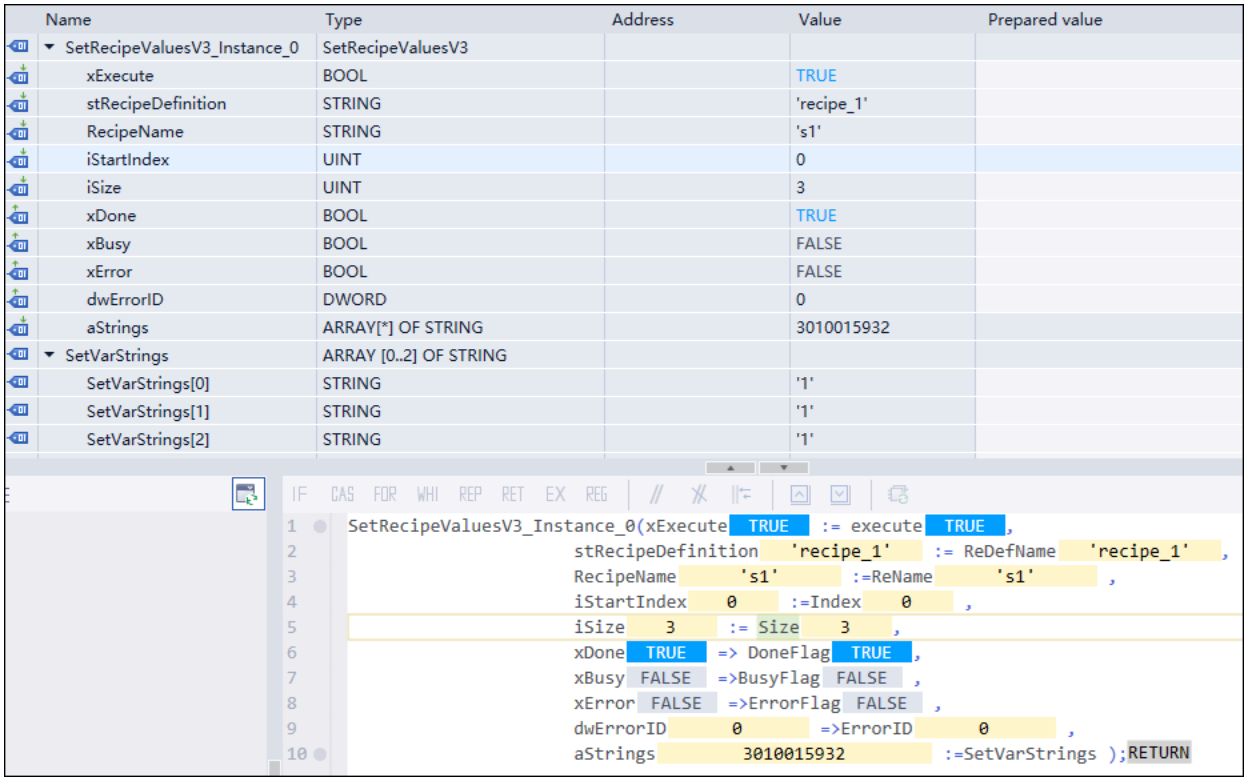
- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Enable	BOOL	TRUE, FALSE	FALSE	This instruction is triggered by the rising edge of xExecute.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	stRecipeDefinition	Recipe definition name	STRING	Follows data type	0	Specifies the recipe definition
	RecipeName	Recipe name	STRING	Follows data type	0	Specifies the recipe
	iSize	Size of the string array	UINT	Follow data type	-	Number of recipe variables to write
	iStartIndex	Start index	UINT	Follow data type	-	Start index for writing recipe variables
	aStrings	String array of recipe values	ARRAY [*] OF STRING	Follows data type	-	Array of variable values to write into the recipe
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is completed.
	xBusy	Instruction in use	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is currently being executed.
	xError	Instruction fault flag	BOOL	TRUE, FALSE	FALSE	Indicates whether a fault occurred during instruction execution.
	dwErrorID	Instruction fault ID	DWORD	Depends on data type	0	Indicates the cause of the instruction faults. For details, see Fault Codes.

- **Function description**
  When the input parameter xExecute changes from FALSE to TRUE, the instruction is enabled and can be used to write the variable values in the array aStrings to the recipe after certain conditions are met.
- **Program example**

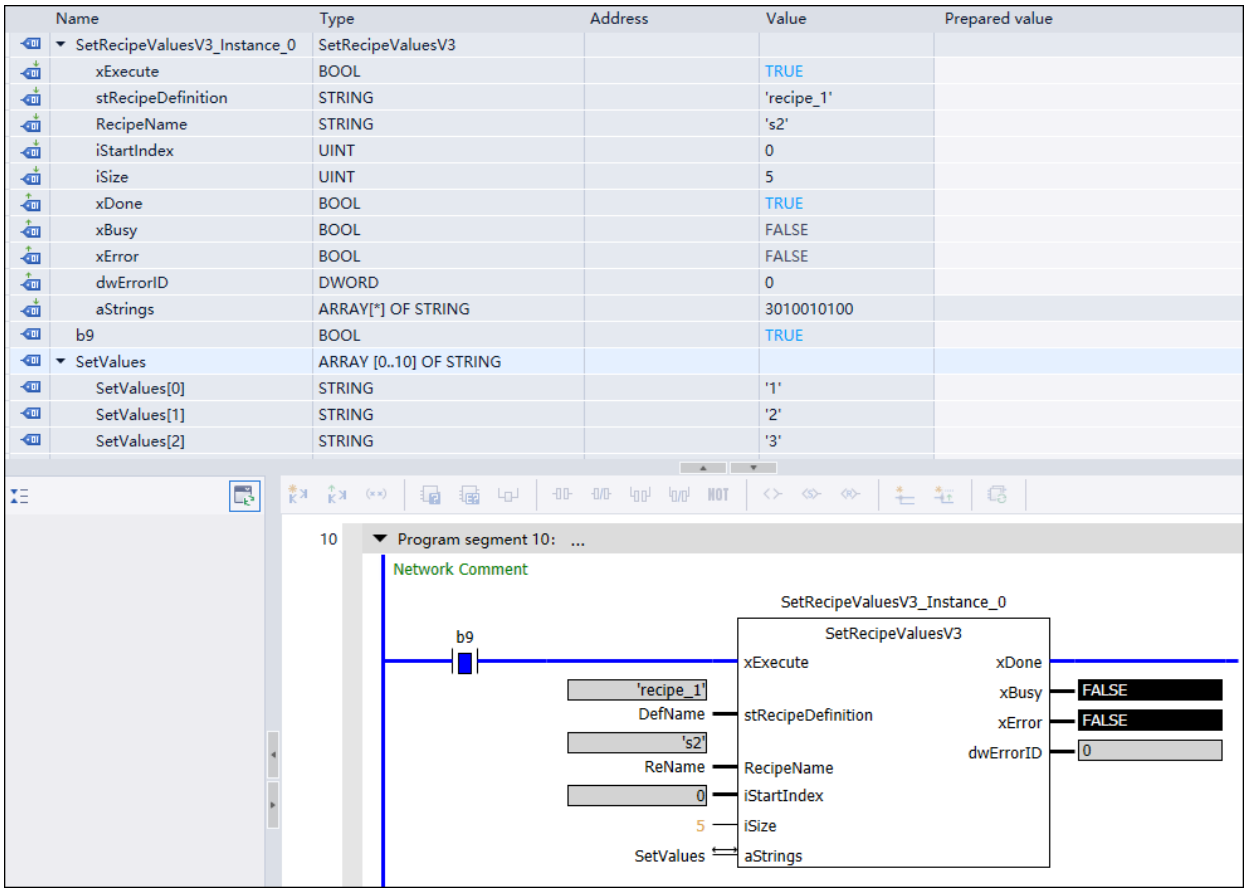
  ST:



Name	Type	Address	Value	Prepared value
SetRecipeValuesV3_Instance_0	SetRecipeValuesV3			
xExecute	BOOL		TRUE	
stRecipeDefinition	STRING		'recipe_1'	
RecipeName	STRING		's1'	
iStartIndex	UINT		0	
iSize	UINT		3	
xDone	BOOL		TRUE	
xBusy	BOOL		FALSE	
xError	BOOL		FALSE	
dwErrorID	DWORD		0	
aStrings	ARRAY[*] OF STRING		3010015932	
SetVarStrings	ARRAY [0..2] OF STRING			
SetVarStrings[0]	STRING		'1'	
SetVarStrings[1]	STRING		'1'	
SetVarStrings[2]	STRING		'1'	

```
1  SetRecipeValuesV3_Instance_0(xExecute TRUE := execute TRUE ,
2                               stRecipeDefinition 'recipe_1' := ReDefName 'recipe_1' ,
3                               RecipeName 's1' :=ReName 's1' ,
4                               iStartIndex 0 :=Index 0 ,
5                               iSize 3 := Size 3 ,
6                               xDone TRUE => DoneFlag TRUE ,
7                               xBusy FALSE =>BusyFlag FALSE ,
8                               xError FALSE =>ErrorFlag FALSE ,
9                               dwErrorID 0 =>ErrorID 0 ,
10                              aStrings 3010015932 :=SetVarStrings );RETURN
```

LD:



Name	Type	Address	Value	Prepared value
SetRecipeValuesV3_Instance_0	SetRecipeValuesV3			
xExecute	BOOL		TRUE	
stRecipeDefinition	STRING		'recipe_1'	
RecipeName	STRING		's2'	
iStartIndex	UINT		0	
iSize	UINT		5	
xDone	BOOL		TRUE	
xBusy	BOOL		FALSE	
xError	BOOL		FALSE	
dwErrorID	DWORD		0	
aStrings	ARRAY[*] OF STRING		3010010100	
b9	BOOL		TRUE	
SetValues	ARRAY [0..10] OF STRING			
SetValues[0]	STRING		'1'	
SetValues[1]	STRING		'2'	
SetValues[2]	STRING		'3'	

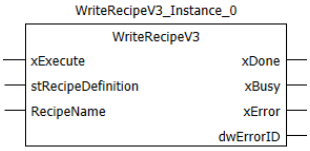
- **Precautions**
  - iStartIndex specifies which variable in the recipe to start writing from. Special attention shall be paid to this index to avoid execution faults. Variable indices within a recipe start from 0.
  - If the sum of the index value and the number of variables to write exceeds the maximum number of recipe variables, the instruction writes sequentially from the start index to the last variable in the recipe.

◦ If the index value is greater than the maximum number of recipe variables, the instruction returns an input fault.

18.2.18.1.41 WriteRecipeV3

Writes the specified recipe data to the PLC.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
WriteRecipeV3	Write the recipe into the PLC	FB		WriteRecipeV3_Instance_0(xExecute := , stRecipeDefinition := , RecipeName := , xDone => , xBusy => , xError => , dwErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Enable	BOOL	TRUE, FALSE	FALSE	This instruction is triggered by the rising edge of xExecute.
	stRecipeDefinition	Recipe definition name	STRING	Follows data type	0	Specifies the recipe definition
	RecipeName	Recipe name	STRING	Follows data type	0	Specifies the recipe
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is completed.
	xBusy	Instruction in use	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is currently being executed.
	xError	Instruction fault flag	BOOL	TRUE, FALSE	FALSE	Indicates whether a fault occurred during instruction execution.
	dwErrorID	Instruction fault ID	DWORD	Depends on data type	0	Indicates the cause of the instruction faults. For details, see Fault Codes.

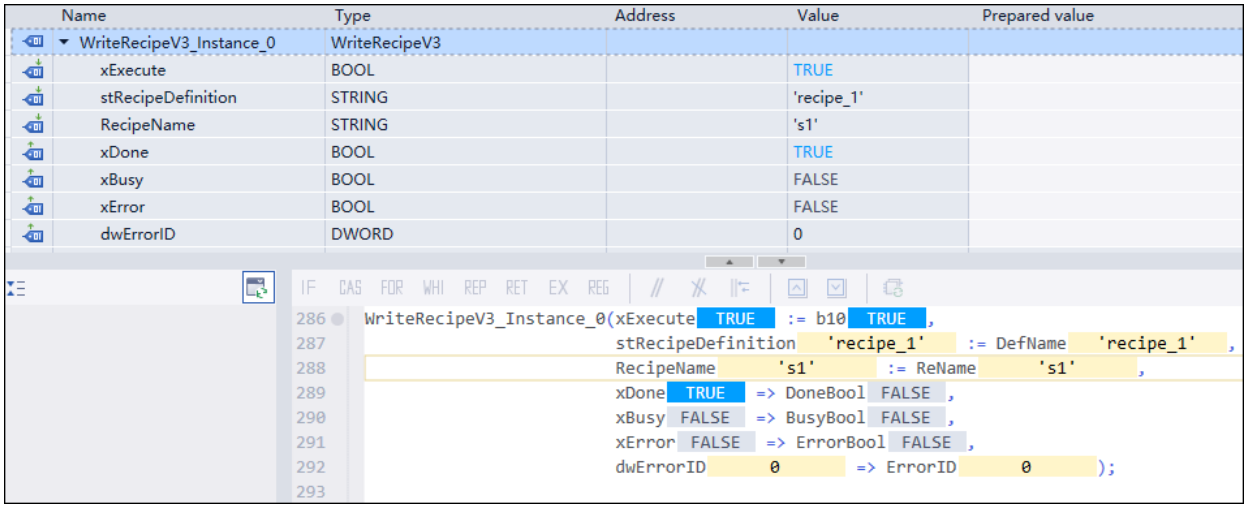
- **Function description**

  When the input parameter xExecute changes from FALSE to TRUE, the instruction is enabled and can be

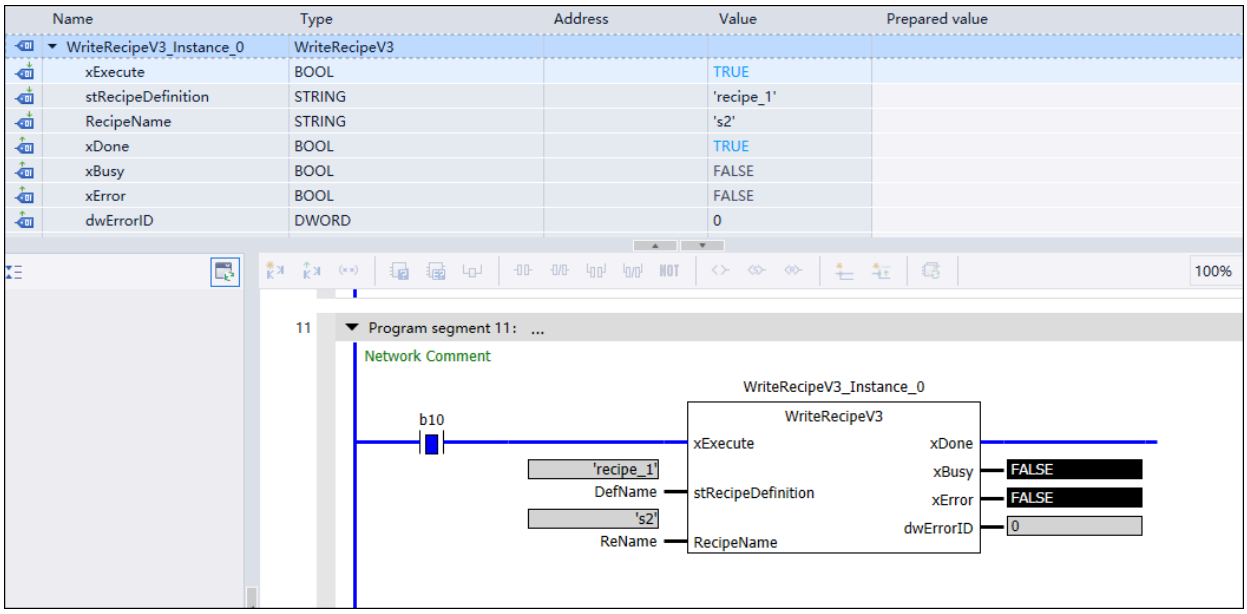
used to write the variables values in the specified temporary recipe into the PLC after certain conditions are met.

- **Program example**

  ST:



  LD:



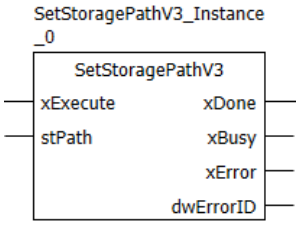
- **Precautions**

  The temporary recipe to be written must already be created and exist in the running PLC.

18.2.18.1.42 SetStoragePathV3

Modifies the recipe file storage path.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SetStoragePathV3	Set the path to store or load the recipe file	FB		SetStoragePathV3_Instance_0(xExecute := , stPath := , xDone => , xBusy => , xError => , dwErrorID =>);

- **Related variables**

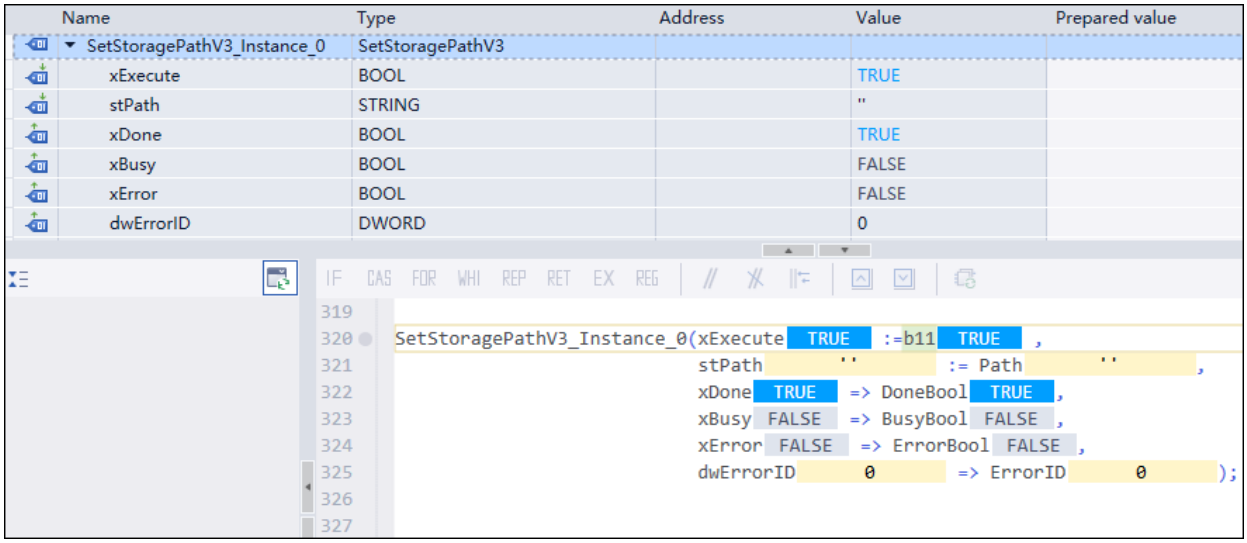
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Enable	BOOL	TRUE, FALSE	FALSE	This instruction is triggered by the rising edge of xExecute.
	stPath	Recipe file operation path	STRING	Follows data type	0	Recipe file operation path If empty, the default is the fixed PLC path.
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is completed.
	xBusy	Instruction in use	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is currently being executed.
	xError	Instruction fault flag	BOOL	TRUE, FALSE	FALSE	Indicates whether a fault occurred during instruction execution.
	dwErrorID	Instruction fault ID	DWORD	Depends on data type	0	Indicates the cause of the instruction faults. For details, see Fault Codes.

- **Function description**

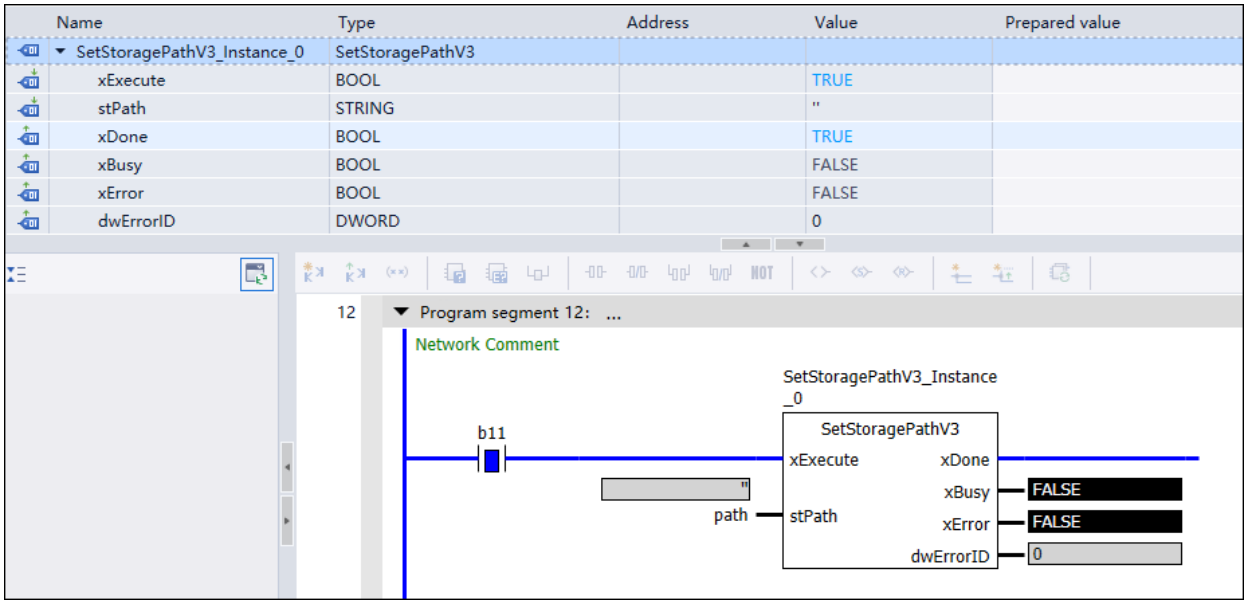
  When the input parameter xExecute changes from FALSE to TRUE, the instruction is enabled. This instruction affects the recipe file operation path of CreateRecipe, DeleteALLRecipe, DeleteRecipe, SaveRecipe, ReadRecipe, LoadRecipe, and RenameRecipe instructions.

- **Program example**

  ST:



LD:



- **Precautions**
  - The input path must start with "/". For example, "/home" indicates that the recipe files will be stored under /home. If only "home" is entered, the instruction will return a fault message indicating an invalid storage path.
  - The input path must be valid, such as "/home/ino/inodata/userspace". If an unexpected path separator such as "\" is entered, the recipe file may fail to be stored correctly.
  - This instruction affects the recipe file operation path of CreateRecipe, DeleteALLRecipe, DeleteRecipe, SaveRecipe, ReadRecipe, LoadRecipe, and RenameRecipe instructions.
  - If stPath is empty or the instruction is not triggered, the storage path is set to the default PLC path under the app directory.

18.2.18.1.43 Fault Codes

- Fault codes for library version 1.0.0.0/1.1.0.0/1.2.0.0

Name	Fault Code (Decimal)	Fault Code (Hexadecimal)	Description
ERR_OK	0	0	Operation succeeded.
ERR_FAILED	1	1	Operation failed.
ERR_PARAMETER	2	2	Parameter fault.
ERR_NOMEMORY	17	11	Insufficient memory.
ERR_RECIPE_FILE_NOT_FOUND	16384	4000	Recipe file not found.
ERR_RECIPE_MISMATCH	16385	4001	The recipe file does not match the current recipe.
ERR_RECIPE_SAVE_ERR	16386	4002	Failed to save the recipe file.
ERR_RECIPE_NOT_FOUND	16387	4003	Recipe not found.
ERR_RECIPE_DEFINITION_NOT_FOUND	16388	4004	Recipe definition not found.
ERR_RECIPE_ALREADY_EXIST	16389	4005	Recipe already exists.
ERR_NO_RECIPE_MANAGER_SET	16390	4006	Recipe manager not initialized.
ERR_RECIPE_NOT_ALL_VARIABLES_WERE_LOADED	16391	4007	The recipe definition contains more variables than the recipe file. Message: The recipe file will be loaded and written.
ERR_RELOAD_RECIPES_ALL_FAILED	16392	4008	Batch load of recipe files failed for all files.
ERR_RELOAD_RECIPES_PARTIAL_SUCCESS	16393	4009	Batch load of recipe files partially succeeded.

- Fault codes for library version 2.0.0.0

Name	Fault Code (decimal)	Fault Code (Hexadecimal)	Description
ERR_OK	0	0	Execution succeeded.
ERR_FAILED	1	1	Device fault (such as insufficient device memory or incorrect variable address in device).
OUT OF RANGE	7	7	Input index exceeds the maximum number of variables.

Name	Fault Code (decimal)	Fault Code (Hexadecimal)	Description
INPUT ERROR	8	8	Input value out of range (for example, the value of ReadOption in the ReadRecipe instruction is not from 0 to 2).
INPUT PATH ERROR	9	9	Input storage path is invalid.
ERR_RECIPE_FILE_NOT_FOUND	16384	4000	Recipe file not found.
RECIPE MISMATCH	16385	4001	Recipe does not match the recipe file.
ERR_RECIPE_SAVE_ERR	16386	4002	Device file read/write fault.
ERR_RECIPE_NOT_FOUND	16387	4003	Recipe not found.
ERR_RECIPE_DEFINITION_NOT_FOUND	16388	4004	Recipe definition not found.
ERR_RECIPE_ALREADY_EXIST	16389	4005	Recipe already exists.

18.2.19 C Function Support

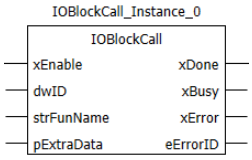
18.2.19.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.2.19.2 IOBlockCall	FB	Call a C/C++ function	IOBlockHandle	Library version 1.2.0.0 and later
18.2.19.3 IOBlockProcess	FB	Call a C/C++ function with user-defined input and output parameters		Library version 1.2.0.0 and later
18.2.19.4 IOBlockLibCall	FB	Call a C/C++ function from a user-defined library		Library version 1.3.0.0 and later
18.2.19.5 IOBlockLibProcess	FB	Call a C/C++ function from a user-defined library with custom input and output parameters		Library version 1.3.0.0 and later

18.2.19.2 IOBlockCall

Calls a function with a specified format in the C/C++ dynamic library and passes IEC variables into the C/C++ program via pointers for processing.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
IOBlockCall	Call a C/C++ function	FB		IOBlockCall_Instance_0(xEnable := , dwID := , strFunName := , pExtraData := , xDone => , xBusy => , xError => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Enable instruction	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled; FALSE: Disabled
	dwID	An ID specified by the user	DWORD	0 to 65535	0	An ID specified by the user to identify the call in C/C++ code
	strFunName	Function name in C/C++	STRING	Follows data type	Empty string	Function name in C/C++ (case-sensitive)
	pExtraData	Additional parameters	POINTER	Follows data type	NULL	Pointer to additional data to be passed to the C/C++ function
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is completed.
	xBusy	Running status flag	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is running.
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is faulty.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	eErrorID	Fault code	ERROR	Follow data type	NO_ERROR	Fault code during C/C++ function call: NO_ERROR: No fault LIB_NOT_FOUND: Dynamic library not found FUN_NAME_ERROR: Incorrect function name FUN_NOT_FOUND: Function not found FUN_CALL_ERROR: Function call failed In addition to the above standard fault codes, eErrorID can also output a user-defined value returned by the C/C++ function.

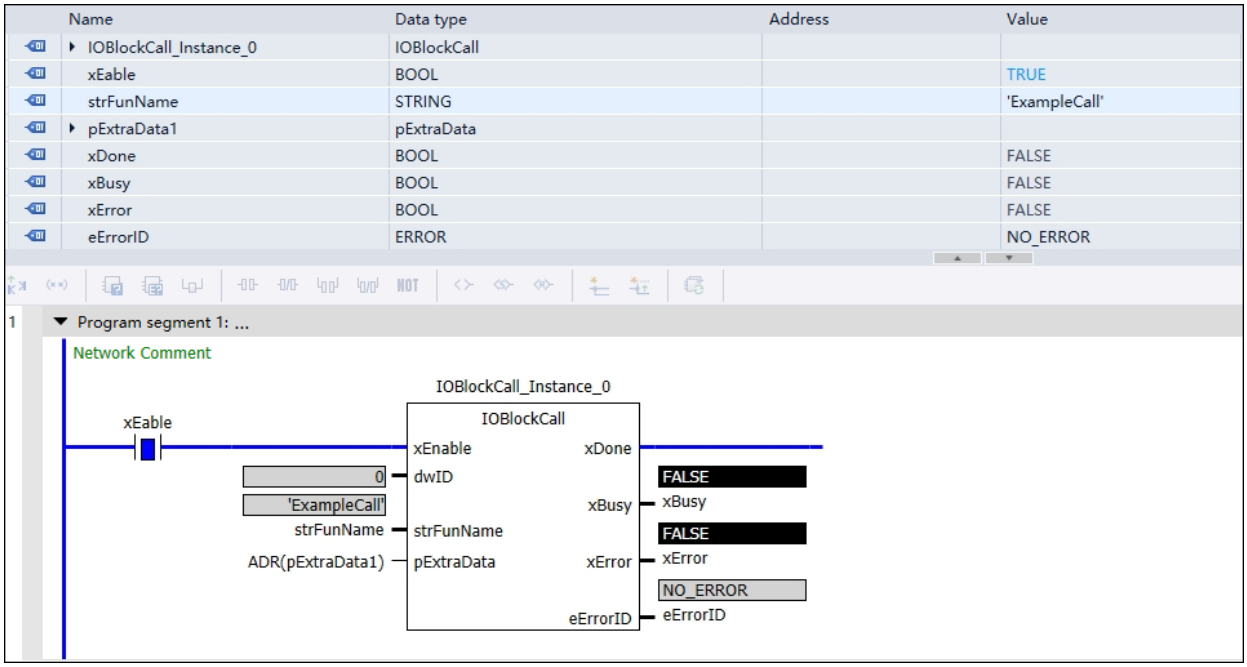
- **Function**
  - When the input parameter xEnable is TRUE, the instruction calls the user-written C/C++ dynamic library. After certain conditions are met, the pointer specified by pExtraData is passed into the C/C++ function context, allowing the user to process the data pointed to. When the input parameter xEnable is FALSE, the call to the C/C++ function is stopped.
  - If the user's dynamic library is not found, the user-defined function is not found, or a fault occurs during the call, the xError variable will be TRUE, and the relevant fault code will be displayed in eErrorID.
- **Program example**

  ST:

	Name	Data type	Address	Value
1	IOBlockCall_Instance_0	IOBlockCall		
2	xEable	BOOL		TRUE
3	dwID	DWORD		0
4	strFunName	STRING		'ExampleCall'
5	pExtraData1	pExtraData		
6	xDone	BOOL		TRUE
7	xBusy	BOOL		FALSE
8	xError	BOOL		FALSE
9	eErrorID	ERROR		NO_ERROR

```
IOBlockCall_Instance_0(
xEnable TRUE :=xEable TRUE ,
dwID 0 :=dwID 0 ,
strFunName 'ExampleCall' := strFunName 'ExampleCall' ,
pExtraData 3023960504 := ADR(pExtraData1),
xDone TRUE =>xDone TRUE ,
xBusy FALSE => xBusy FALSE ,
xError FALSE =>xError FALSE ,
eErrorID NO_ERROR =>eErrorID NO_ERROR );
```

LD:



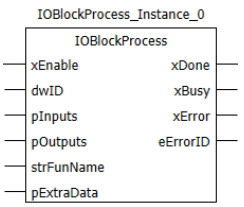
- **Precautions**

  When replacing the dynamic library, it is recommended to place the PLC application in STOP mode. The C/C++ dynamic library will be reloaded when the PLC enters RUN mode, ensuring that the dynamic library in memory is the latest version.

18.2.19.3 IOBlockProcess

Calls a function with a specified format in the C/C++ dynamic library and passes IEC variables into the C/C++ program via pointers for processing.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
IOBlockProcess	Call a C/C++ function with user-defined input and output parameters	FB		IOBlockProcess_Instance_0(xEnable := , dwID := , pInputs := , pOutputs := , strFunName := , pExtraData := , xDone => , xBusy => , xError => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Enable instruction	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled; FALSE: Disabled

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	dwID	An ID specified by the user	DWORD	0 to 65535	0	An ID specified by the user to identify the call in C/C++ code
	pInputs	POU of the input	POINTER	Follows data type	NULL	Typically a pointer to the mapped input structure
	pOutputs	POU of the output	POINTER	Follows data type	NULL	Typically a pointer to the mapped output structure
	strFunName	Function name in C/C++	STRING	Follows data type	Empty string	Function name in C/C++ (case-sensitive)
	pExtraData	Additional parameters	POINTER	Follows data type	NULL	Pointer to additional data to be passed to the C/C++ function
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is completed.
	xBusy	Running status flag	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is running.
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is faulty.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	eErrorID	Fault code	ERROR	Follow data type	NO_ERROR	Fault code during C/C++ function call: NO_ERROR: No fault LIB_NOT_FOUND: Dynamic library not found FUN_NAME_ERROR: Incorrect function name FUN_NOT_FOUND: Function not found FUN_CALL_ERROR: Function call failed In addition to the above standard fault codes, eErrorID can also output a user-defined value returned by the C/C++ function.

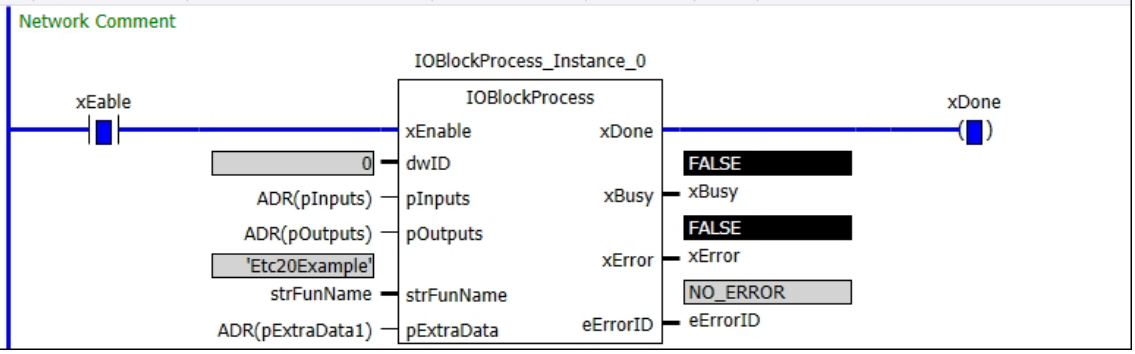
- **Function**
  - When the input parameter xEnable is TRUE, the instruction calls the user-written C/C++ dynamic library. After certain conditions are met, the pointer specified by pInput, pOutput, and pExtraData is passed into the C/C++ function context, allowing the user to process the data pointed to. When the input parameter xEnable is FALSE, the call to the C/C++ function is stopped.
  - If the user's dynamic library is not found, the user-defined function is not found, or a fault occurs during the call, the xError variable will be TRUE, and the relevant fault code will be displayed in eErrorID.
- **Program example**
  ST:

	Name	Data type	Address	Value
1	IOBlockProcess_Instance_0	IOBlockProcess		
2	xEable	BOOL		TRUE
3	dwID	DWORD		0
4	pInputs	pInputs		
5	pOutputs	pOutputs		
6	strFunName	STRING		'Etc20Example'
7	pExtraData1	pExtraData		
8	xDone	BOOL		TRUE
9	xBusy	BOOL		FALSE
10	xError	BOOL		FALSE
11	eErrorID	ERROR		NO_ERROR

```
IOBlockProcess_Instance_0(
xEnable TRUE := xEable TRUE ,
dwID 0 := dwID 0 ,
pInputs 3023960156 :=ADR(pInputs) ,
pOutputs 3023960288 := ADR(pOutputs),
strFunName 'Etc20Example' :=strFunName 'Etc20Example' ,
pExtraData 3023960504 :=ADR(pExtraData1) ,
xDone TRUE => xDone TRUE ,
xBusy FALSE => xBusy FALSE ,
xError FALSE => xError FALSE ,
eErrorID NO_ERROR => eErrorID NO_ERROR );
```

LD:

	Name	Data type	Address	Value
1	IOBlockProcess_Instance_0	IOBlockProcess		
2	xEable	BOOL		TRUE
3	pInputs	pInputs		
4	pOutputs	pOutputs		
5	strFunName	STRING		'Etc20Example'
6	pExtraData1	pExtraData		
7	xDone	BOOL		TRUE
8	xBusy	BOOL		FALSE
9	xError	BOOL		FALSE
10	eErrorID	ERROR		NO_ERROR



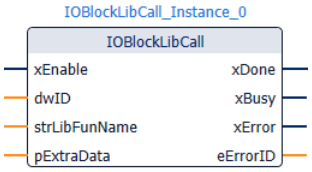
- **Precautions**

  When replacing the dynamic library, it is recommended to place the PLC application in STOP mode. The C/C++ dynamic library will be reloaded when the PLC enters RUN mode, ensuring that the dynamic library in memory is the latest version.

18.2.19.4 IOBlockLibCall

Calls a function with a specified format in the C/C++ dynamic library from a user-defined library and passes IEC variables into the C/C++ program via pointers for processing.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
IOBlockLibCall	Call a C/C++ function from a user-defined library	FB		IOBlockLibCall_Instance_0(xEnable := , dwID := , strLibFunName := , pExtraData := , xDone => , xBusy => , xError => , eErrorID =>);

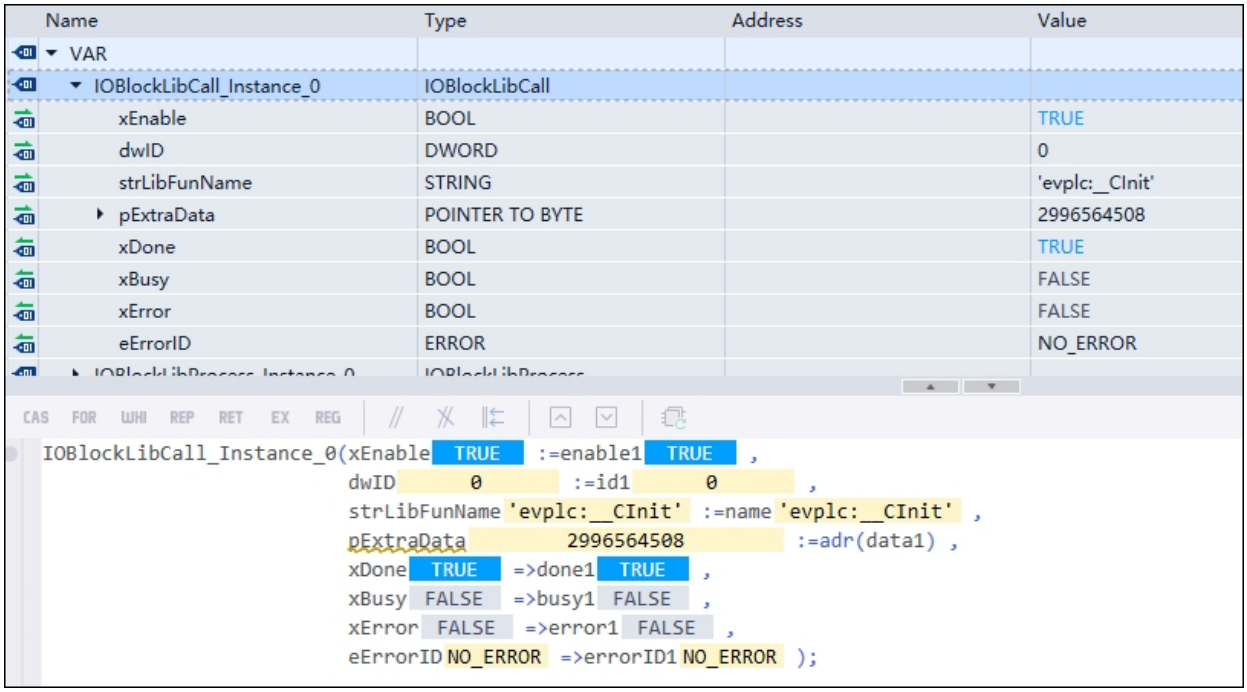
- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Enable instruction	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled; FALSE: Disabled
	dwID	An ID specified by the user	DWORD	0 to 65535	0	An ID specified by the user to identify the call in C/C++ code
	strLibFunName	Name of the library and function in C/C++	STRING	Follows data type	Empty string	Name of the library and function to be called, in the format "LibraryName:FunctionName". For example, if the library is libmylib.so and the function name is func, then this parameter is mylib:func.
	pExtraData	Additional parameters	POINTER	Follows data type	NULL	Pointer to additional data to be passed to the C/C++ function
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is completed.
	xBusy	Running status flag	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is running.
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is faulty.

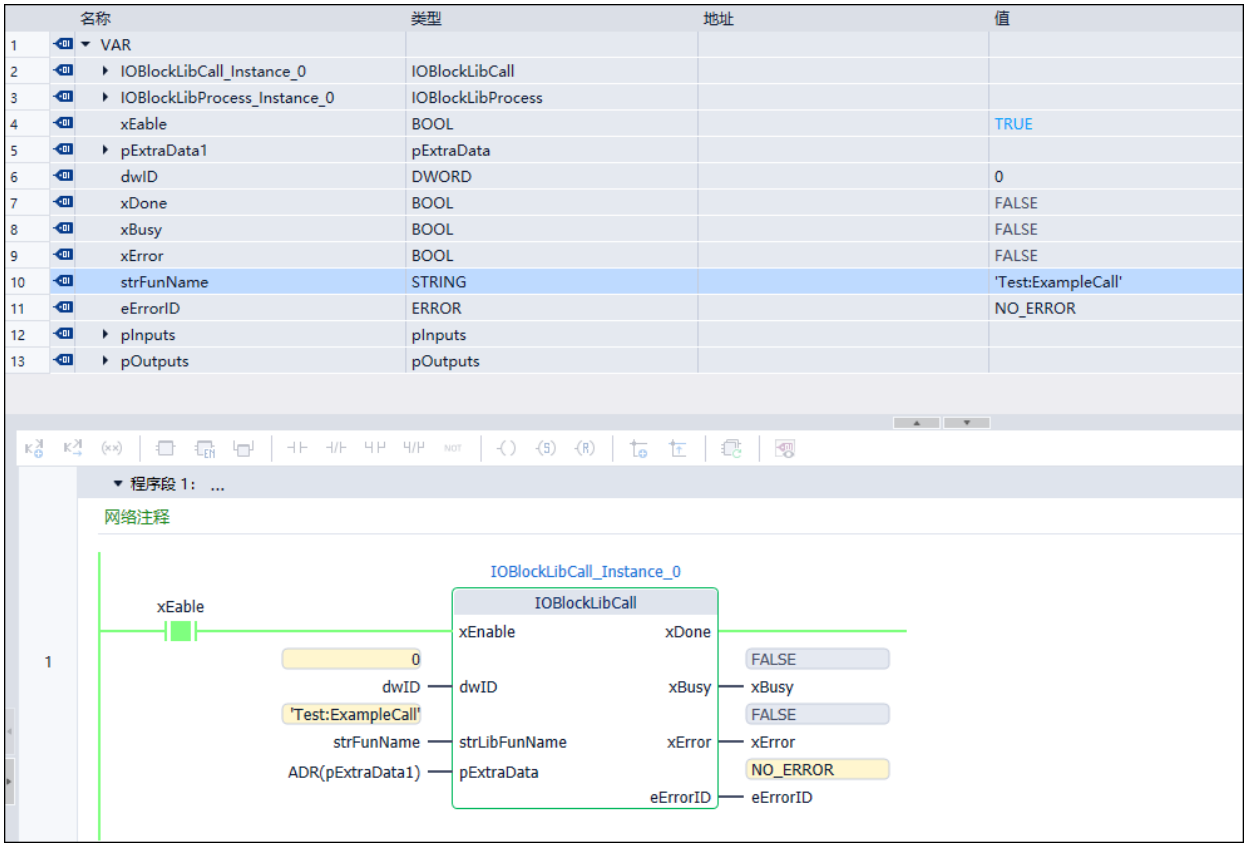
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	eErrorID	Fault code	ERROR	Follow data type	NO_ERROR	Fault code during C/C++ function call: NO_ERROR: No error; LIB_NOT_FOUND: Dynamic library not found; FUN_NAME_ERROR: Function name error; FUN_NOT_FOUND: Function not found; FUN_CALL_ERROR: Function call failed In addition to the above standard fault codes, eErrorID can also output a user-defined value returned by the C/C++ function.

- **Function**
  - When the input parameter xEnable is TRUE, the instruction calls the user-defined C/C++ dynamic library. After certain conditions are met, the pointer specified by pExtraData is passed into the C/C++ function context, allowing the user to process the data pointed to. When the input parameter xEnable is FALSE, the call to the C/C++ function is stopped.
  - If the user's dynamic library is not found, the user-defined function is not found, or a fault occurs during the call, the xError variable will be TRUE, and the relevant fault code will be displayed in eErrorID.
- **Program example**

ST:



LD:



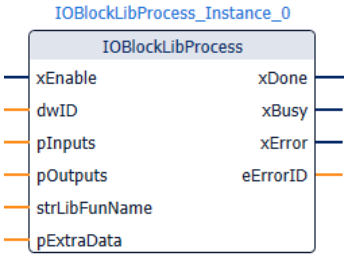
- **Precautions**

  When replacing the dynamic library, it is recommended to place the PLC application in STOP mode. The C/C++ dynamic library will be reloaded when the PLC enters RUN mode, ensuring that the dynamic library in memory is the latest version.

18.2.19.5 IOBlockLibProcess

Calls a function with a specified format in the user-defined C/C++ dynamic and passes IEC variables into the C/C++ program via pointers for processing.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
IOBlockLibProcess	Call a C/C++ function from a user-defined library with custom input and output parameters	FB		IOBlockLibProcess_Instance_0(xEnable := , dwID := , pInputs := , pOutputs := , strLibFunName := , pExtraData := , xDone => , xBusy => , xError => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Enable instruction	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled; FALSE: Disabled
	dwID	An ID specified by the user	DWORD	0 to 65535	0	An ID specified by the user to identify the call in C/C++ code
	pInputs	POU of the input	POINTER	Follows data type	NULL	Typically a pointer to the mapped input structure
	pOutputs	POU of the output	POINTER	Follows data type	NULL	Typically a pointer to the mapped output structure
	strLibFunName	Function name in C/C++	STRING	Follows data type	Empty string	Name of the library and function to be called, in the format "LibraryName:FunctionName". For example, if the library is libmylib.so and the function name is func, then this parameter is mylib:func (case-sensitive).
	pExtraData	Additional parameters	POINTER	Follows data type	NULL	Pointer to additional data to be passed to the C/C++ function
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is completed.
	xBusy	Running status flag	BOOL	TRUE, FALSE	FALSE	Indicates the instruction is running.
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Indicates whether the instruction is faulty.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	eErrorID	Fault code	ERROR	Follow data type	NO_ERROR	Fault code during C/C++ function call: NO_ERROR: No error; LIB_NOT_FOUND: Dynamic library not found; FUN_NAME_ERROR: Function name error; FUN_NOT_FOUND: Function not found; FUN_CALL_ERROR: Function call failed; In addition to the above standard error codes, eErrorID also returns the return value of the C/C++ function (user-defined).

- **Function**
  - When the input parameter xEnable is TRUE, the instruction calls the user-written C/C++ dynamic library. After certain conditions are met, the pointer specified by pInput, pOutput, and pExtraData is passed into the C/C++ function context, allowing the user to process the data pointed to. When the input parameter xEnable is FALSE, the call to the C/C++ function is stopped.
  - If the user's dynamic library is not found, the user-defined function is not found, or a fault occurs during the call, the xError variable will be TRUE, and the relevant fault code will be displayed in eErrorID.
- **Program example**

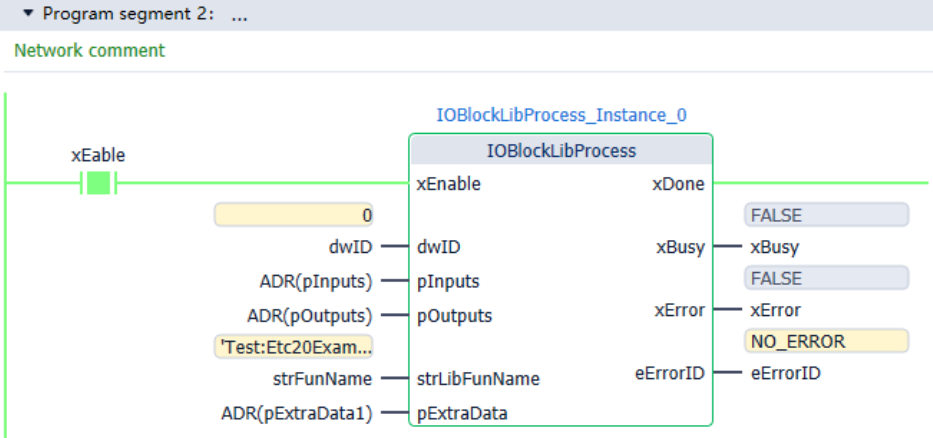
  ST:

Name	Type	Address	Value
IOBlockLibProcess_Instance_0	IOBlockLibProcess		
xEnable	BOOL		TRUE
dwID	DWORD		0
pInputs	POINTER TO BYTE		2995753536
pOutputs	POINTER TO BYTE		2995753552
strLibFunName	STRING		'evplc:ExampleRelease'
pExtraData	POINTER TO BYTE		2995753528
xDone	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
eErrorID	ERROR		NO_ERROR

```
IOBlockLibProcess_Instance_0(xEnable TRUE :=enable2 TRUE ,
                             dwID 0 :=id2 0 ,
                             pInputs 2995753536 :=adr(inputs) ,
                             pOutputs 2995753552 :=adr(ouputs) ,
                             strLibFunName 'evplc:ExampleRelea... :=name 'evplc:ExampleRelea... ,
                             pExtraData 2995753528 :=adr(data2) ,
                             xDone TRUE =>done2 TRUE ,
                             xBusy FALSE =>busy2 FALSE ,
                             xError FALSE =>error2 FALSE ,
                             eErrorID NO_ERROR =>errorid2 NO_ERROR );
RETURN
```

LD:

Name	Type	Address	Value
VAR			
IOBlockLibProcess_Instance_0	IOBlockLibProcess		
xEable	BOOL		TRUE
pExtraData1	pExtraData		
dwID	DWORD		0
xDone	BOOL		FALSE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
strFunName	STRING		'Test:Etc20Example
eErrorID	ERROR		NO_ERROR
pInputs	pInputs		
pOutputs	pOutput		



- **Precautions**

  When replacing the dynamic library, it is recommended to place the PLC application in STOP mode. The C/C++ dynamic library will be reloaded when the PLC enters RUN mode, ensuring that the dynamic library in memory is the latest version.

18.2.20 Sequencer

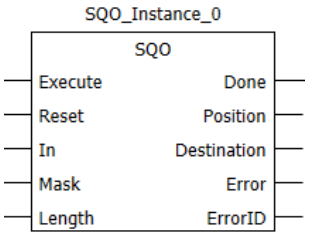
18.2.20.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
SQO	FB	Sequencer output	Util	Library version 1.3.0.1 and later

18.2.20.2 SQO

Outputs the set value for each step in sequence via a pre-filled data table. It is suitable for replacing complex self-locking and interlocking logic in controling processes such as assembly lines and packaging machines.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SQO	Sequencer output	FB		SQO_Instance_0(Execute := , Reset := , In := , Mask := , Length := , Done => , Position => , Destination => , Error => , ErrorID =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Execute	Input signal	Follows data type.	FALSE	Rising edge-triggered
	Reset	Reset step	Follows data type	FALSE	"Position" becomes 0 when triggered.
	In	Sequence array	Follows data type	0	Specifies the first element of the sequence array.

Input/Output	Variable	Name	Valid Range	Default Value	Description
	Mask	Mask	Follows data type	0	Defines the enable/mask strategy for output data, taking effect during output assignment. 1: Output enabled. 0: Output disabled (masked).
	Length	Total steps	Follows data type	0	Total number of steps. Number of elements in the output sequence array.
Output	Done	Done	Follows data type	FALSE	Instruction is completed. Set to TRUE when "Length" equals "Position".
	Position	Current step	Follows data type	0	Current executing step. Current position in the array.
	Destination	Output value	Follows data type	0	Output value Array element value after mask operation.
	Error	Function block fault flag	Follows data type	FALSE	Function block fault flag
	ErrorID	Function block fault ID	Follows data type	0	Function block fault ID

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Execute	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Reset	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
In	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Mask	-	-	-	-	-	-	-	-	-	√	√	√	-	-	-	-	-	-	-	-

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
Length	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Done	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Position	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Destination	-	-	-	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-
Error	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ErrorID	-	-	√	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **Function**
  - Executes step advancement and outputs data only on the rising edge of Execute.
  - When Execute is first triggered on a rising edge, Position = 1, and the instruction outputs the value corresponding to the second element of the In array.
  - When the rising edge of Execute is triggered normally, the Position value increments; when Position equals Length, Done is set. On the next trigger, Position resets to 1 and a new cycle begins.
  - Each rising edge of Execute performs a bitwise AND between the array element at the current Position and the Mask, and a bitwise AND between Destination and the inverted Mask. The results are combined using a bitwise OR, and the final result is written to Destination.
    Output value calculation formula:

    $$\text{Destination} = (\text{In}[Position]\ AND\ Mask)\ OR\ \left(\text{Destination AND}\ \left(\text{NOT}(\text{Mask})\right)\right)$$

  - When Reset is TRUE, all output variables are cleared.
  - To modify the Position parameter, use the Set accessor of the ModifyPosition property to force-write the Position value into the function block.
- **Program example**

ST:

- When the input signal Execute changes from FALSE to TRUE, Destination outputs the calculated result. Position indicates the current execution position of the sequence array.

	Name	Type	Address	Value
1	SQO_Instance_0	SQO		
2	xExecute	BOOL		TRUE
3	arrIn	ARRAY [0..4] OF DINT		
4	dMask	DINT		65535
5	dLength	DINT		4
6	xReset	BOOL		FALSE
7	xDone	BOOL		FALSE
8	dPosition	DINT		1
9	dDestination	DINT		1
10	xError	BOOL		FALSE
11	wErrorID	DWORD		0

```
SQO_Instance_0(
Execute TRUE := xExecute TRUE,
Reset FALSE := xReset FALSE,
In := arrIn[0] 0,
Mask := dMask 65535,
Length 4 := dLength 4,
Done FALSE => xDone FALSE,
Position 1 => dPosition 1,
Destination 1 => dDestination 1,
Error FALSE => xError FALSE,
ErrorID 0 => wErrorID 0);RETURN
```

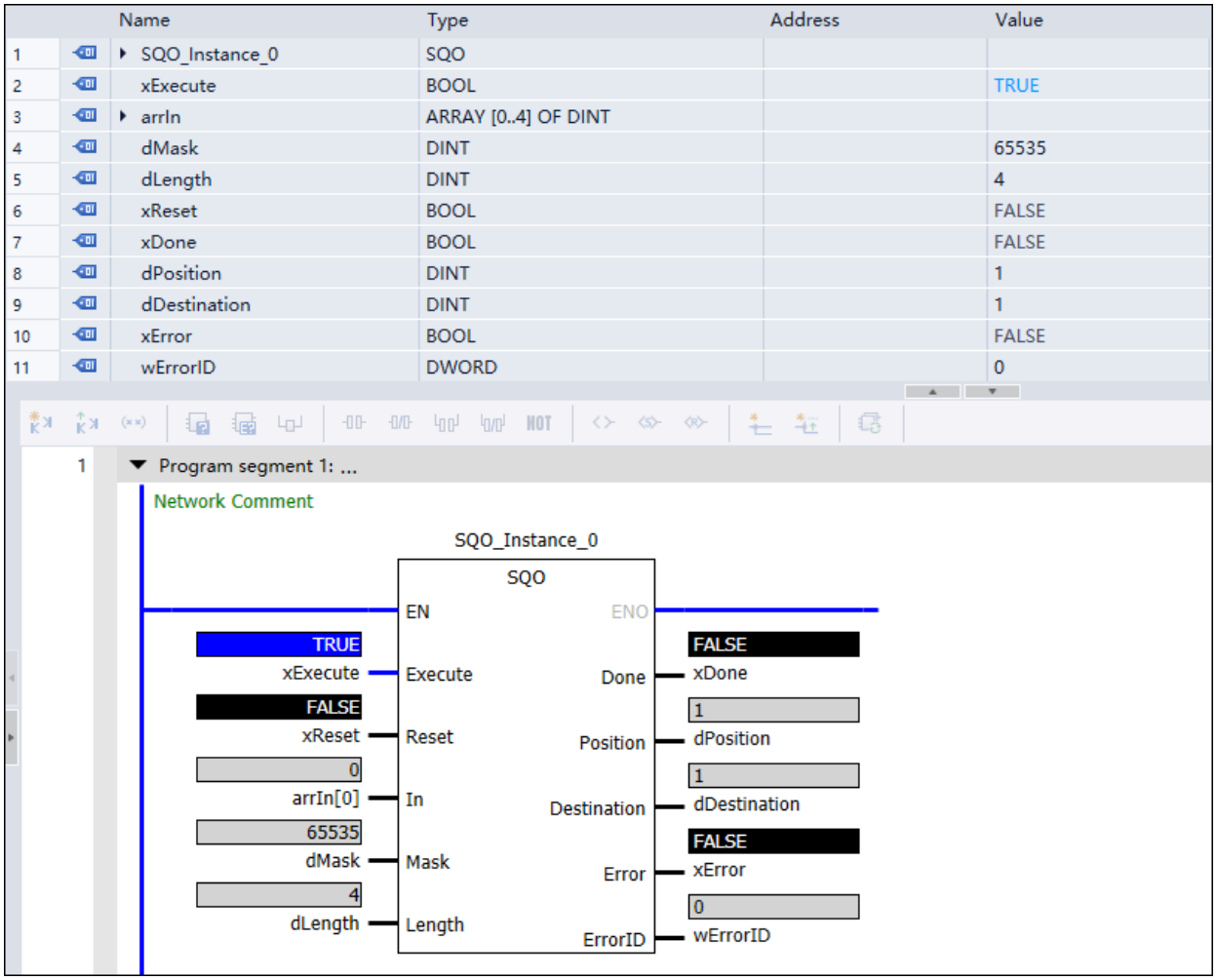
- When the input signal Execute is TRUE, the rising edge of Mod_Position serves as the modification trigger signal, and the set value dPosition_Mod is forced into Position through the Set accessor of the ModifyPosition property.

	Name	Type	Address	Value
1	SQO_Instance_0	SQO		
2	xExecute	BOOL		TRUE
3	arrIn	ARRAY [0..4] OF DINT		
4	dMask	DINT		65535
5	dLength	DINT		4
6	xReset	BOOL		FALSE
7	xDone	BOOL		FALSE
8	dPosition	DINT		1
9	dDestination	DINT		1
10	xError	BOOL		FALSE
11	wErrorID	DWORD		0

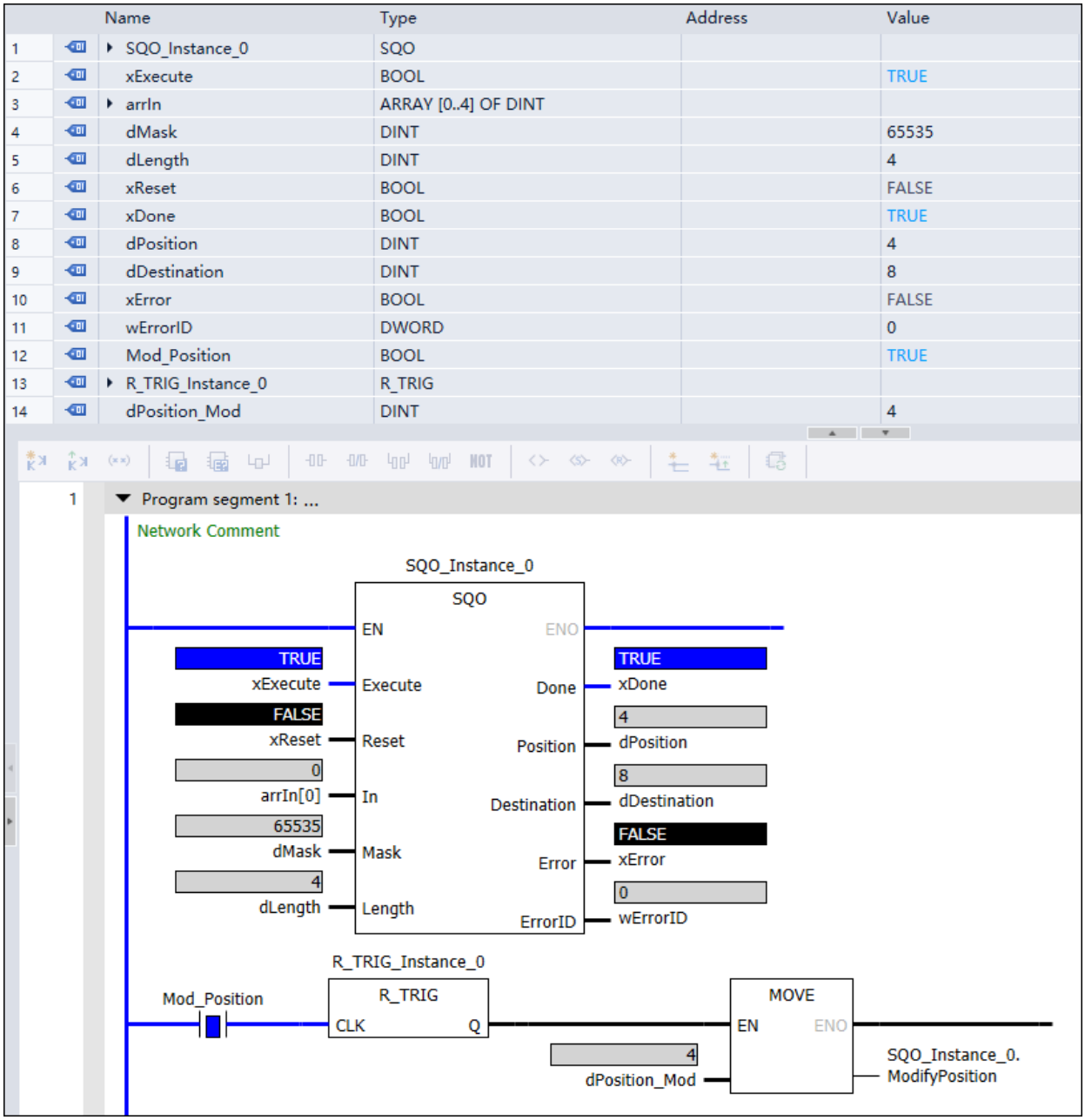
```
SQO_Instance_0(
Execute TRUE := xExecute TRUE,
Reset FALSE := xReset FALSE,
In := arrIn[0] 0,
Mask := dMask 65535,
Length 4 := dLength 4,
Done FALSE => xDone FALSE,
Position 1 => dPosition 1,
Destination 1 => dDestination 1,
Error FALSE => xError FALSE,
ErrorID 0 => wErrorID 0);RETURN
```

LD:

- When the input signal Execute changes from FALSE to TRUE, Destination outputs the calculated result. Position indicates the current execution position of the sequence array.



◦ When the input signal Execute is TRUE, the rising edge of Mod_Position serves as the modification trigger signal, and the set value dPosition_Mod is forced into Position through the Set accessor of the ModifyPosition property.



- **Precautions**
  - The In array supports input arrays and array index elements.
  - When Execute is TRUE and Position is modified, Destination outputs the corresponding calculated result of In[Position]; when Execute is FALSE and Position is modified, Destination does not change. When Execute transitions from FALSE to TRUE, Destination outputs the result corresponding to In[Position+1].
  - When Length is less than or equal to 0, Error is TRUE, and ErrorID is 1.
  - When the In array size is less than Length, Error is TRUE, and ErrorID is 2.
  - When Position is greater than Length, Error is TRUE, and ErrorID is 3.
  - When Position is less than 0, Error is TRUE, and ErrorID is 4.
  - When the dimension of the In array is greater than 3, Error is TRUE, and ErrorID is 5.
  - When the In array or Mask is not bound to a variable, Error is TRUE, and ErrorID is 6.

18.3 Process Instructions

18.3.1 PID Control

18.3.1.1 Instruction List

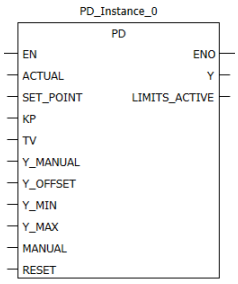
Instruction	FB/FC	Function	Library	Applicable Library Version
18.3.1.2 PD	FB	PD controller	Util	Library version 1.0.0.0 and later
18.3.1.3 PID	FB	PID controller		Library version 1.0.0.0 and later
18.3.1.4 PID_FIXCYCLE	FB	Universal PID control that can manually set the cycle time		Library version 1.0.0.0 and later
18.3.1.5 PID_AT	FB	Universal PID controller	LoopControl	Library version 1.0.0.0 and later
18.3.1.6 PID_ATC	FB	Bipolar temperature PID controller		Library version 1.0.0.0 and later
18.3.1.7.1 FPTLGSim	FB	Simulation of first-order lag plus pure dead time	Logic	Library version 1.0.0.0 and later
18.3.1.7.2 FSMITHSim	FB	First-order lag plus pure dead time compensation simulation		Library version 1.0.0.0 and later
18.3.1.7.3 SPTLGSim	FB	Simulation of second-order lag plus pure dead time		Library version 1.0.0.0 and later
18.3.1.7.4 SSMITHSim	FB	Second-order lag plus pure dead time compensation simulation		Library version 1.0.0.0 and later
18.3.1.7.5 LEAD_LAG	FB	Lead-lag compensator		Library version 1.0.0.0 and later
18.3.1.7.6 DELIM	FC	Deviation determination		Library version 1.0.0.0 and later
18.3.1.7.7 DZLIM	FB	Dead zone determination		Library version 1.0.0.0 and later
18.3.1.7.8 RALIM	FB	Rate determination		Library version 1.0.0.0 and later
18.3.1.7.9 MANCON	FB	Manual operator	ProcessControl	Library version 1.0.0.0 and later
18.3.1.7.10 ORSEL	FB	Override control		Library version 1.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.3.1.7.11 PWMC	FB	Pulse modulation control		Library version 1.0.0.0 and later
18.3.1.7.12 RATIO	FB	Ratio control		Library version 1.0.0.0 and later
18.3.1.7.13 SPLRANGE	FB	Split-range control		Library version 1.0.0.0 and later
18.3.1.7.14 THPOSC	FB	Three-position control.		Library version 1.0.0.0 and later
18.3.1.7.15 TWPOSC	FB	Two-position control		Library version 1.0.0.0 and later

18.3.1.2 PD

This instruction performs proportional-derivative (PD) control.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PD	PD controller	FB		PD_Instance_0(ACTUAL := , SET_POINT := , KP := , TV := , Y_MANUAL := , Y_OFFSET := , Y_MIN := , Y_MAX := , MANUAL := , RESET := , Y => , LIMITS_ACTIVE =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	ACTUAL	Current value of the control variable	Follows data type	-	Current value of the control variable.

Input/Output	Variable	Name	Valid Range	Default Value	Description
	SET_POINT	Set value	Follows data type	-	Set value.
	KP	Proportional coefficient	Follows data type	-	Proportional coefficient, the proportional gain of the P-part.
	TV	Derivative time	Follows data type	-	Derivative time, time of the D part in seconds. For example, 0.5 indicates 500 microseconds.
	Y_MANUAL	Manual output value	Follows data type	-	When the value of MANUAL is TRUE, the output value of Y is Y_MANUAL.
	Y_OFFSET	Offset output of Y	Follows data type	-	Offset output of Y.
	Y_MIN	Minimum output value of Y	Follows data type	-	Minimum output value of Y.
	Y_MAX	Maximum output value of Y	Follows data type	-	Maximum output value of Y.
	MANUAL	Manual output	Follows data type	-	Manual output control. If the value is TRUE, manual operation is activated and the output value of Y is Y_MANUAL.

Input/Output	Variable	Name	Valid Range	Default Value	Description
	RESET	Reset a bit	Follows data type	-	When the value is TRUE, the controller is reset, and the value of Y is Y_OFFSET during re-initialization.
Output	Y	Output value	Follows data type	-	Output value of the PD controller.
	LIMITS_ACTIVE	Set limit reached	Follows data type	FALSE	When the value is TRUE, Y has reached the given limit (Y_MIN, Y_MAX).

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
ACTUAL	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
SET_POINT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
KP	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
TV	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Y_MANUAL	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Y_OFFSET	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Y_MIN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Y_MAX	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
MANUAL	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Y	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
LIMITS_ACTIVE	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Program example**

  This instruction performs incremental proportional-derivative (PD) control.

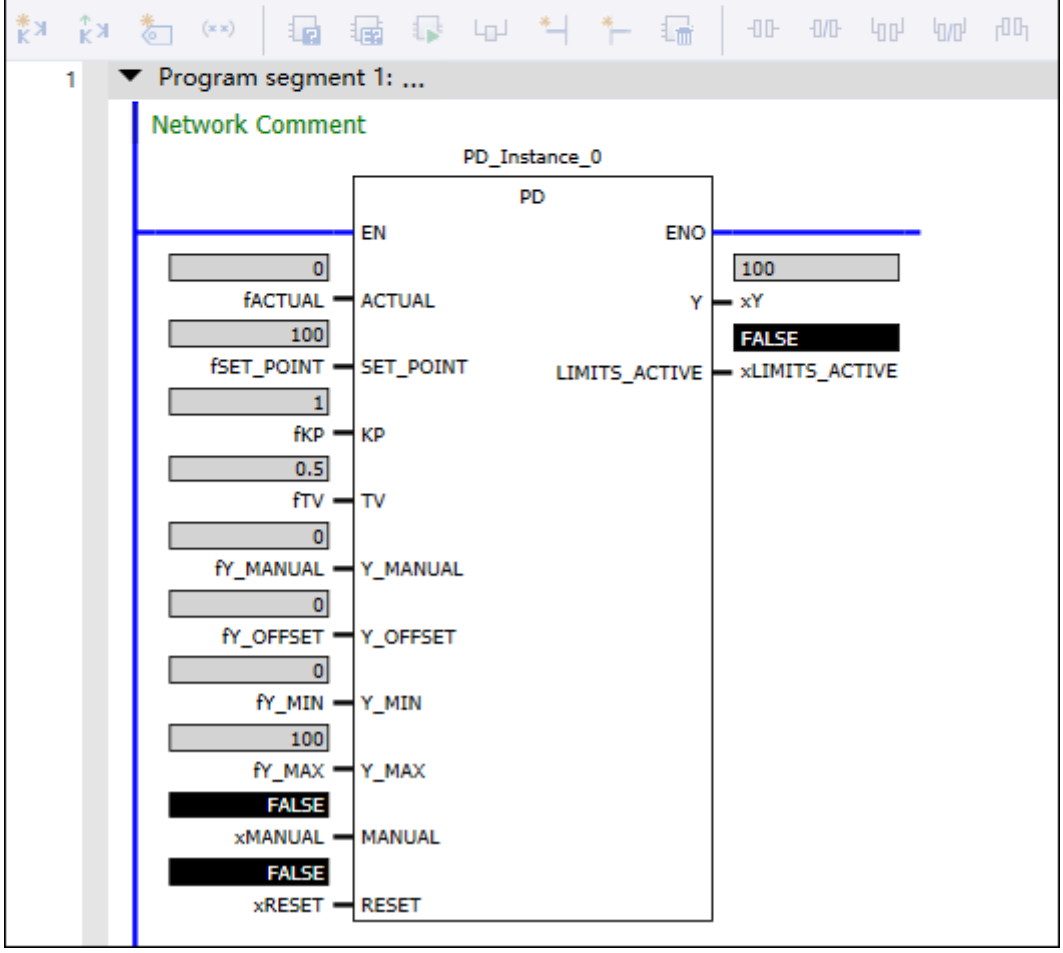
  ST:

	Name	Type	Address	Value	Prepared val
1	PD_Instance_0	PD			
2	ACTUAL	REAL		0	
3	SET_POINT	REAL		100	
4	KP	REAL		1	
5	TV	REAL		0.5	
6	Y_MANUAL	REAL		0	
7	Y_OFFSET	REAL		0	
8	Y_MIN	REAL		0	
9	Y_MAX	REAL		100	
10	MANUAL	BOOL		FALSE	
11	RESET	BOOL		FALSE	
12	Y	REAL		100	
13	LIMITS_ACTIVE	BOOL		FALSE	

```
IF  CAS  FOR  WHI  REP  RET  EX  //  //
1  PD_Instance_0(
2  ACTUAL 0 := fACTUAL 0 ,
3  SET_POINT 100 := fSET_POINT 100
4  KP 1 := fKP 1 ,
5  TV 0.5 := fTV 0.5 ,
6  Y_MANUAL 0 := fY_MANUAL 0
7  Y_OFFSET 0 := fY_OFFSET 0
8  Y_MIN 0 := fY_MIN 0 ,
9  Y_MAX 100 := fY_MAX 100 ,
10 MANUAL FALSE := xMANUAL FALSE ,
11 RESET FALSE := xRESET FALSE ,
12 Y 100 => xY 100 ,
13 LIMITS_ACTIVE FALSE => xLIMITS_ACTIVE FALSE );RETURN
```

LD:

	Name	Type	Address	Value
1	PD_Instance_0	PD		
2	ACTUAL	REAL		0
3	SET_POINT	REAL		100
4	KP	REAL		1
5	TV	REAL		0.5
6	Y_MANUAL	REAL		0
7	Y_OFFSET	REAL		0
8	Y_MIN	REAL		0
9	Y_MAX	REAL		100
10	MANUAL	BOOL		FALSE
11	RESET	BOOL		FALSE
12	Y	REAL		100
13	LIMITS_ACTIVE	BOOL		FALSE



- **Precautions**
  - The output value of Y is calculated as follows:

$$Y = KP \cdot (\Delta + 1/TN \int edt + TV\ \delta\Delta/\delta t) + Y_OFFSET$$

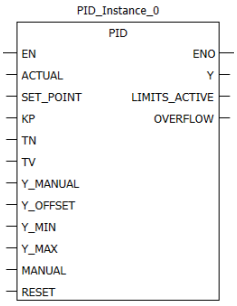
  - If there are no output limit values, the values of Y_MIN and Y_MAX must be set to 0.
  - These controls will only take effect if Y_MIN is less than Y_MAX.
  - If the value of the MANUAL is TRUE, and the value of KP is non-zero, Y_MANUAL will be written to the output value of Y.

◦ If RESET and MANUAL are set simultaneously, the value of Y will be Y_OFFSET during re-initialization.

18.3.1.3 PID

This is the instruction for proportional integral derivative control.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PID	PID controller	FB		PID_Instance_0(ACTUAL := , SET_POINT := , KP := , TN := , TV := , Y_MANUAL := , Y_OFFSET := , Y_MIN := , Y_MAX := , MANUAL := , RESET := , Y => , LIMITS_ACTIVE => , OVERFLOW =>);

- **Related variables**

Variable

Input / Output	Variable	Name	Valid Range	Default Value	Description
Input	ACTUAL	Current value of the control variable	Follows data type	-	Current value of the control variable.
	SET_POINT	Set value	Follows data type	-	Set value of the control variable.
	KP	Proportional coefficient	Follows data type	-	Proportional coefficient, the proportional gain of the P-part.

Input / Output	Variable	Name	Valid Range	Default Value	Description
	TN	Integral time	Follows data type	-	Integral time, time of the I part in seconds. For example, 0.5 indicates 500 milliseconds. The value must be greater than 0. Otherwise, the function block does not perform any calculation.
	TV	Derivative time	Follows data type	-	Derivative time, time of the D part in seconds. For example, 0.5 indicates 500 milliseconds.
	Y_MANUAL	Manual output value	Follows data type	-	Manual output value. When the value of MANUAL is TRUE, the output value of Y is Y_MANUAL.
	Y_OFFSET	Offset output of Y	Follows data type	-	Offset output of Y.
	Y_MIN	Minimum output value of Y	Follows data type	-	Minimum output value of Y.
	Y_MAX	Maximum output value of Y	Follows data type	-	Maximum output value of Y.
	MANUAL	Manual output	Follows data type	-	Manual output control. If the value is TRUE, manual operation is activated and the output value of Y is Y_MANUAL.
	RESET	Reset a bit	Follows data type	-	When the value is TRUE, the controller is reset, and the value of Y is Y_OFFSET during re-initialization.
Output	Y	Output value	Follows data type	-	Output value of the PID controller.
	LIMITS_ACTIVE	Set limit reached	Follows data type	FALSE	When the value is TRUE, Y has reached the given limit (Y_MIN, Y_MAX).
	OVERFLOW	Overflow flag	Follows data type	FALSE	Overflow flag. If the integral part of e is larger, incorrect parameter settings for the controller may cause overflow. For the sake of security, the output of OVERFLOW is TRUE in this case.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
ACTUAL	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
SET_POINT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
KP	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
TN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
TV	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Y_MANUAL	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Y_OFFSET	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Y_MIN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Y_MAX	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
MANUAL	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
Y	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
LIMITS_ACTIVE	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OVERFLOW	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

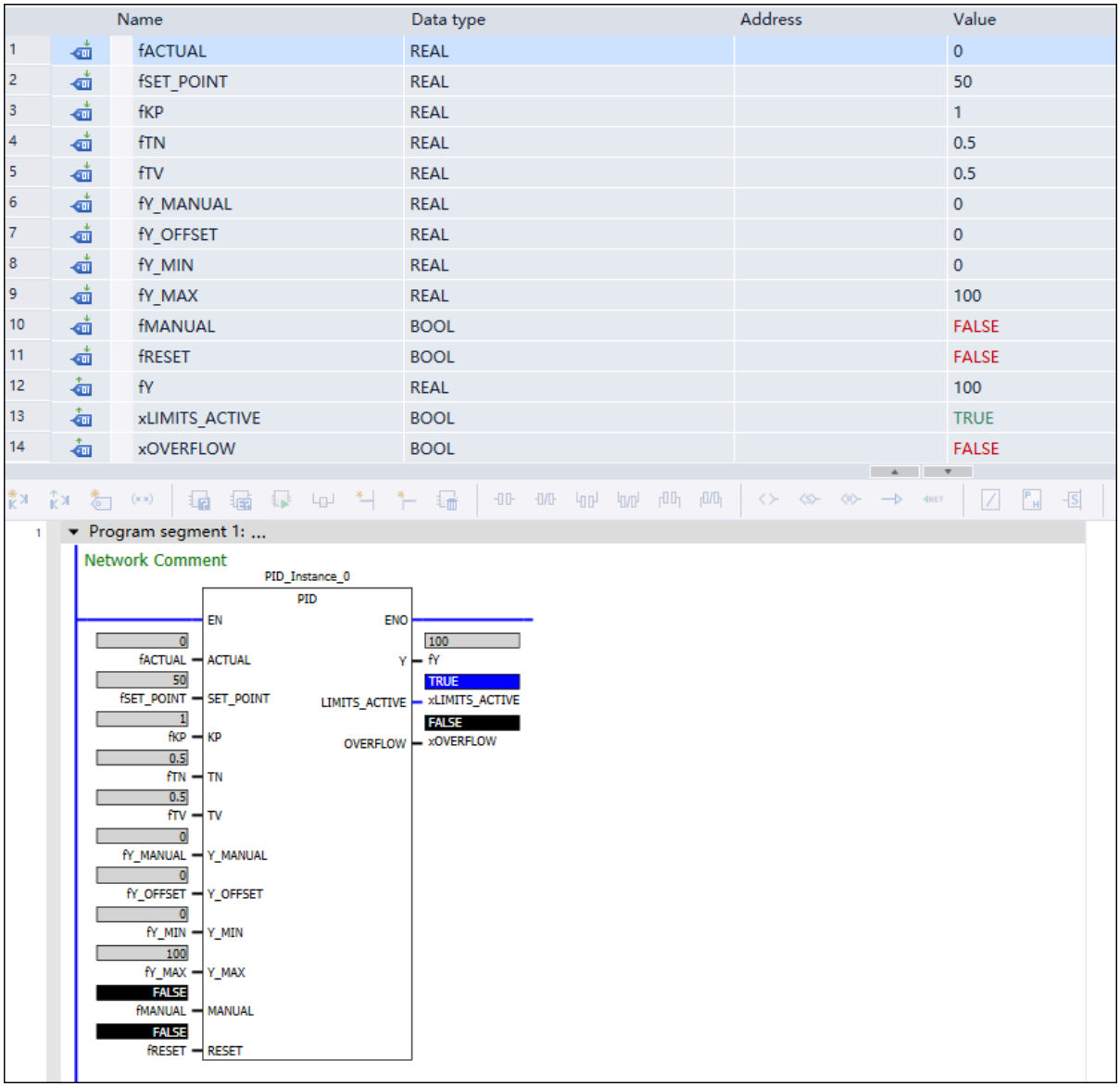
- **Program example**

  ST:

	Name	Data type	Address	Value
1	fACTUAL	REAL		0
2	fSET_POINT	REAL		50
3	fKP	REAL		1
4	fTN	REAL		0.5
5	fTV	REAL		0.5
6	fY_MANUAL	REAL		0
7	fY_OFFSET	REAL		0
8	fY_MIN	REAL		0
9	fY_MAX	REAL		100
10	fMANUAL	BOOL		FALSE
11	fRESET	BOOL		FALSE
12	fY	REAL		100
13	xLIMITS_ACTIVE	BOOL		TRUE
14	xOVERFLOW	BOOL		FALSE

```
IF  CAS  FOR  WHI  REP  RET  EX
1  PID_Instance_0(
2  ACTUAL 0 := fACTUAL 0,
3  SET_POINT 50 := fSET_POINT 50,
4  KP 1 := fKP 1,
5  TN 0.5 := fTN 0.5,
6  TV 0.5 := fTV 0.5,
7  Y_MANUAL 0 := fY_MANUAL 0,
8  Y_OFFSET 0 := fY_OFFSET 0,
9  Y_MIN 0 := fY_OFFSET 0,
10 Y_MAX 100 := fY_MAX 100,
11 MANUAL FALSE := fMANUAL FALSE,
12 RESET FALSE := fRESET FALSE,
13 Y 100 => fY 100,
14 LIMITS_ACTIVE TRUE => xLIMITS_ACTIVE TRUE,
15 OVERFLOW FALSE => xOVERFLOW FALSE );RETURN
```

LD:



- **Precautions**
  - The output value of Y is calculated as follows:

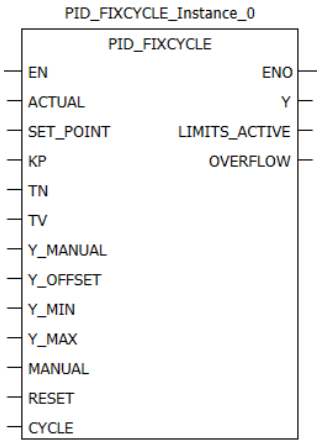
$$Y = KP * \left(e + \frac{1}{TN}\int edt + TV\frac{de}{dt}\right) + Y_OFFSET$$

  - If there are no output limit values, the values of Y_MIN and Y_MAX must be set to 0.
  - These controls will only take effect if Y_MIN is less than Y_MAX.
  - If the value of the MANUAL is TRUE, Y_MANUAL will be written to the output value of Y.

18.3.1.4 PID_FIXCYCLE

This instruction implements universal PID control that can manually set the cycle time.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PID_FIXCYCLE	Universal PID control that can manually set the cycle time	FB		PID_FIXCYCLE_Instance_0(ACTUAL := , SET_POINT := , KP := , TN := , TV := , Y_MANUAL := , Y_OFFSET := , Y_MIN := , Y_MAX := , MANUAL := , RESET := , CYCLE := , Y => , LIMITS_ACTIVE => , OVERFLOW =>);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	ACTUAL	Current value of the control variable	Follows data type	-	Current value of the control variable.
	SET_POINT	Set value	Follows data type	-	Set value of the control variable.
	KP	Proportional coefficient	Follows data type	-	Proportional coefficient, the proportional gain of the P-part.

Input/ Output	Variable	Name	Valid Range	Default Value	Description
	TN	Integral time	Follows data type	-	Integral time, time of the I part in seconds. For example, 0.5 indicates 500 milliseconds. The value must be greater than 0. Otherwise, the function block does not perform any calculation.
	TV	Derivative time	Follows data type	-	Derivative time, time of the D part in seconds. For example, 0.5 indicates 500 milliseconds.
	Y_MANUAL	Manual output value	Follows data type	-	When the value of MANUAL is TRUE, the output value of Y is Y_MANUAL.
	Y_OFFSET	Offset output of Y	Follows data type	-	Offset output of Y.
	Y_MIN	Minimum output value of Y	Follows data type	-	Minimum output value of Y.
	Y_MAX	Maximum output value of Y	Follows data type	-	Maximum output value of Y.
	MANUAL	Manual output	Follows data type	-	Manual output control. If the value is TRUE, manual operation is activated and the output value of Y is Y_MANUAL.
	RESET	Reset a bit	Follows data type	-	When the value is TRUE, the controller is reset, and the value of Y is Y_OFFSET during re-initialization.
	CYCLE	Processing time	Follows data type	-	It specifies the processing time.
Output	Y	Output value	Follows data type	-	Output value of the PID controller in seconds. For example, 0.5 indicates 500 milliseconds.
	LIMITS_ACTIVE	Set limit reached	Follows data type	FALSE	When the value is TRUE, Y has reached the given limit (Y_MIN, Y_MAX).

Input/ Output	Variable	Name	Valid Range	Default Value	Description
	OVERFLOW	Overflow flag	Follows data type	FALSE	Overflow flag. If the integral part of e is larger, incorrect parameter settings for the controller may cause overflow. For the sake of security, the output of OVERFLOW is TRUE in this case.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
ACTUAL	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
SET_POINT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
KP	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
TN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
TV	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Y_MANUAL	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Y_OFFSET	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Y_MIN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Y_MAX	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
MANUAL	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
RESET	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CYCLE	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Y	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
LIMITS_ACTIVE	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
OVERFLOW	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Program example**

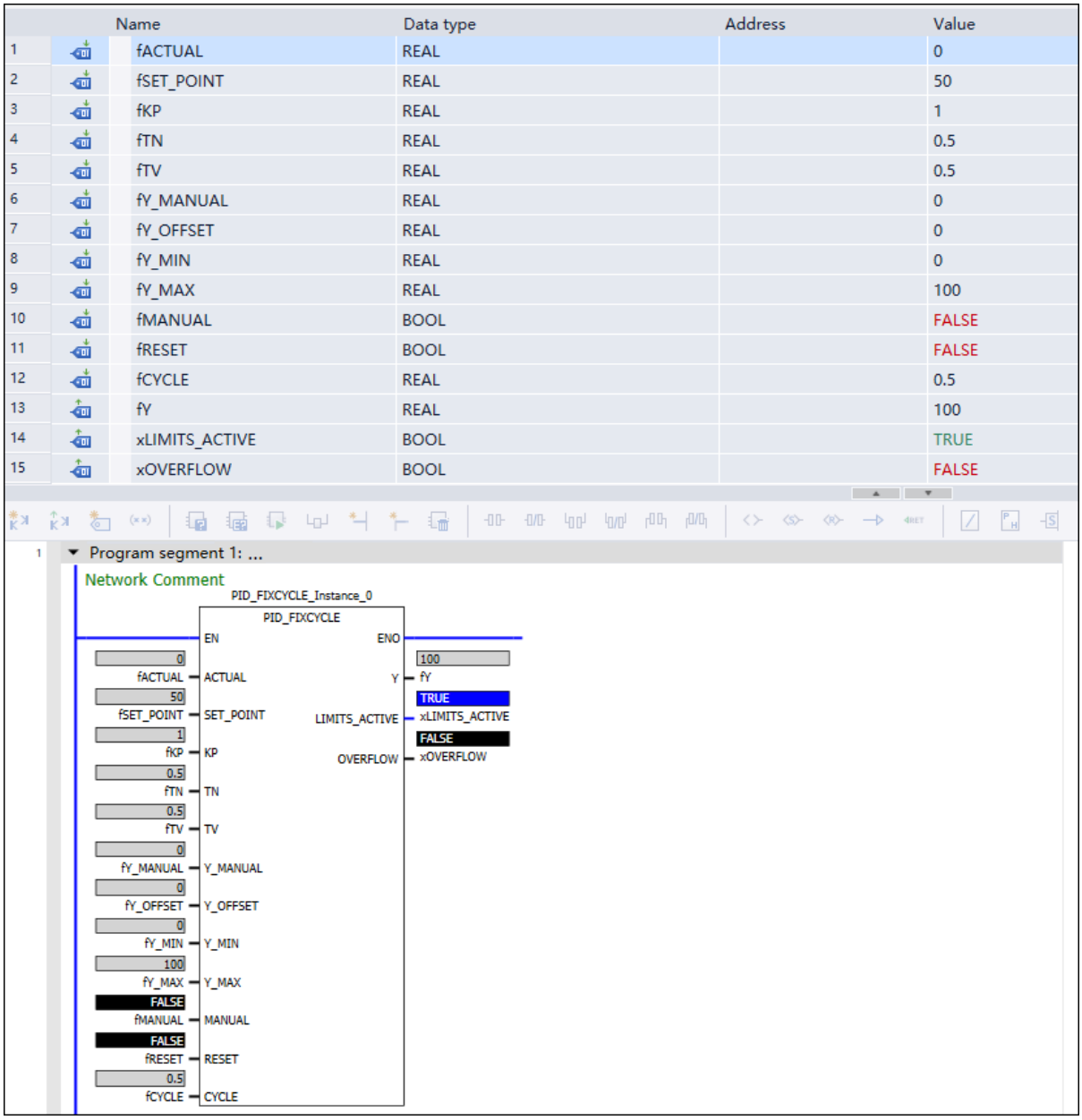
  This instruction implements universal PID control that can manually set the cycle time.

  ST:

	Name	Data type	Address	Value
1	fACTUAL	REAL		0
2	fSET_POINT	REAL		50
3	fKP	REAL		1
4	fTN	REAL		0.5
5	fTV	REAL		0.5
6	fY_MANUAL	REAL		0
7	fY_OFFSET	REAL		0
8	fY_MIN	REAL		0
9	fY_MAX	REAL		100
10	fMANUAL	BOOL		FALSE
11	fRESET	BOOL		FALSE
12	fCYCLE	REAL		0.5
13	fY	REAL		100
14	xLIMITS_ACTIVE	BOOL		TRUE
15	xOVERFLOW	BOOL		FALSE

```
IF  CAS  FOR  WHI  REP  RET  EX
1  PID_FIXCYCLE_Instance_0(
2  ACTUAL 0 := fACTUAL 0,
3  SET_POINT 50 := fSET_POINT 50,
4  KP 1 := fKP 1,
5  TN 0.5 := fTN 0.5,
6  TV 0.5 := fTV 0.5,
7  Y_MANUAL 0 := fY_MANUAL 0,
8  Y_OFFSET 0 := fY_OFFSET 0,
9  Y_MIN 0 := fY_MIN 0,
10 Y_MAX 100 := fY_MAX 100,
11 MANUAL FALSE := fMANUAL FALSE,
12 RESET FALSE := fRESET FALSE,
13 CYCLE 0.5 := fCYCLE 0.5,
14 Y 100 => fY 100,
15 LIMITS_ACTIVE TRUE => xLIMITS_ACTIVE TRUE,
16 OVERFLOW FALSE => xOVERFLOW FALSE );RETURN
```

  LD:



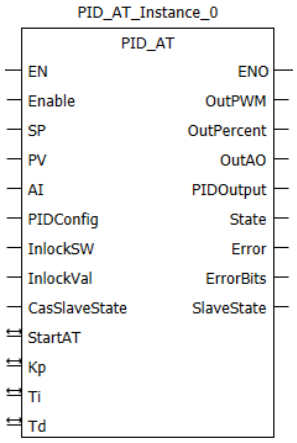
	Name	Data type	Address	Value
1	fACTUAL	REAL		0
2	fSET_POINT	REAL		50
3	fKP	REAL		1
4	fTN	REAL		0.5
5	fTV	REAL		0.5
6	fY_MANUAL	REAL		0
7	fY_OFFSET	REAL		0
8	fY_MIN	REAL		0
9	fY_MAX	REAL		100
10	fMANUAL	BOOL		FALSE
11	fRESET	BOOL		FALSE
12	fCYCLE	REAL		0.5
13	fY	REAL		100
14	xLIMITS_ACTIVE	BOOL		TRUE
15	xOVERFLOW	BOOL		FALSE

- **Precautions**
  - If there are no output limit values, the values of Y_MIN and Y_MAX must be set to 0.
  - These controls will only take effect if Y_MIN is less than Y_MAX.
  - If the value of the MANUAL is TRUE, Y_MANUAL will be written to the output value of Y.

18.3.1.5 PID_AT

This is the instruction for universal PID controller.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PID_AT	Universal PID controller	FB		PID_AT_Instance_0(Enable := , SP := , PV := , AI := , PIDConfig := , InlockSW := , InlockVal := , CasSlaveState := , StartAT := , Kp := , Ti := , Td := , OutPWM => , OutPercent => , OutAO => , PIDOutput => , State => , Error => , ErrorBits => , SlaveState =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Enable	Enable operation	BOOL	TRUE, FALSE	FALSE	Used as a reset switch. FALSE: PID block is disabled and output is cleared. TRUE: PID block is enabled and performs PV acquisition > PID calculation > Output of manipulated variable.
	SP	Setpoint	REAL	Follows data type	0.0	Setpoint.
	PV	Feedback value	REAL	Follows data type	0.0	Feedback value.
	AI	Analog input value	DINT	Follows data type	0	Analog feedback value.
	InlockSW	Interlock switch	BOOL	TRUE, FALSE	FALSE	FALSE: Off. TRUE: On.
	InlockVal	Interlock safety value	REAL	-100.0 to 100.0	0.0	Interlock safety value.
	CasSlaveState	Slave data	Slave_Data	-	-	Slave data structure.

<table>
<tr><th>Input/Output</th><th>Variable</th><th>Name</th><th>Data Type</th><th>Valid Range</th><th>Default Value</th><th>Description</th></tr>
<tr><td></td><td colspan="6">In case of master control, the slave data must be transmitted to this structure.
<table>
<tr><th>Name</th><th>Data Type</th><th>Valid Range</th><th>Default Value</th><th>Description</th></tr>
<tr><td>PV</td><td>REAL</td><td>Follows data type</td><td>0.0</td><td>Slave process value::
When the slave is detected to be in non-cascade automatic mode, the output of the master control tracks the slave process value.</td></tr>
<tr><td>OVE</td><td>BYTE</td><td>0, 1, 2</td><td>0</td><td>Overlimit status of slave circuit:
0: Normal.
1: Upper limit reached.
2: Lower limit reached.</td></tr>
<tr><td>CAS_State</td><td>UINT</td><td>0, 1</td><td>0</td><td>Cascade use mode:
0: Slave non-automatic mode.
1: Slave automatic mode. In case of slave, the value should be returned to the master controller SUBCAS_MODE.</td></tr>
</table></td></tr>
<tr><td></td><td>PIDConfig</td><td>PID parameter configuration structure</td><td>PID_AT_CONFIG</td><td>-</td><td>-</td><td>PID parameter configuration structure.</td></tr>
<tr><td></td><td colspan="6"></td></tr>
</table>

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
	GeneralConfig		PID_GENERALCONFIG	-	-	PID general configuration structure.
	GeneralConfig	SampleTime	DINT	1 to 1000000	100	Sampling time in ms. That is the PID calculation cycle.
		CycleTime	DINT	1 to 1000000	1000	PWM output cycle in ms. Takes effect when the output mode is set to PWM mode.
		PIDType	DINT	0, 1, 2, 3	0	PID type: 0: PID calculation. 1: PI calculation. 2: PD calculation. 3: ID calculation.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
		ControlMode	DINT	0, 1	0	PID control mode: 0: Reverse (unipolar heating control). 1: Forward (unipolar cooling control). If other values are input, a parameter error will be triggered.
		ManualEnable	BOOL	TRUE, FALSE	FALSE	Manual mode enable. FALSE: Automatic mode is enabled. TRUE: Manual mode is enabled.
		ManualValue	REAL	OutMin~OutMax	0.0	Output percentage in manual mode, unit: %.
		ATCoefficient	REAL	0.0 to 1.0	0.5	Self-tuning adjustment coefficient, ranging from 0.1 to 1.0. A larger value results in faster adjustment, with a default setting of 0.5.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
		DelayStartUp	DINT	0 to 1000000	0	Delay startup time in ms.
		DelayShutDown	DINT	0 to 1000000	0	Delay shutdown time in ms.
		PhysicalQuantity	INT	0 to 3	0	Physical object to control: 0: General. 1: Temperature and other parameters. For details, see object description. Default: General
		PhysicalUnit	DINT	Follows data type	-	Unit of the physical object.
		IsMaster	BOOL	TRUE, FALSE	FALSE	Master control: 0: No. 1: Yes. Value must be assigned during cascade configuration.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
		IsSlave	BOOL	TRUE, FALSE	FALSE	Slave control: 0: No. 1: Yes. Value must be assigned during cascade configuratio n.
		ATNum	DINT	0 to 20	2	Number of auto-tuning waveforms (applicable only to unipolar). 0: Reserved. 1-20: Number of oscillation waveforms.
		ATOut	DINT	OutMin~Out Max	100	Output percentage in auto-tuning (applicable only to unipolar, unit: %).
	InputConfig		PID_INPUTC ONFIG	-	-	PID input configuration structure.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
	InputConfig	PVType	DINT	0, 1	0	PV type: 0: Use current PV as input. 1: Use peripheral analog signal as input, with conversion based on analog range.
		AIMin	DINT	Follows data type	0	Minimum value for analog conversion.
		AIMax	DINT	Follows data type	20000	Maximum value for analog conversion.
		PVMin	REAL	Follows data type	0.0	Minimum value of PV
		PVMax	REAL	Follows data type	100.0	Maximum value of PV.
		PVLowLimit	REAL	Follows data type	-100.0	Lower input limit.
		PVHighLimit	REAL	Follows data type	1000.0	Lower input limit.
		ErrorAction	BOOL	TRUE, FALSE	FALSE	Output on error alarm: FALSE: Maintain previous value. TRUE: Output substitute value.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
		ErrorOut	REAL	OutMin~OutMax	0.0	Output of a predefined percentage value on error alarm.
		SPSel	BOOL	TRUE, FALSE	FALSE	Setpoint source selection: FALSE: External setpoint (SP configuration). TRUE: Internal setpoint (panel operation setpoint).
		SPRatio	REAL	Follows data type	0.0	SP setpoint change. If set to 0, no change judgment is performed. If set greater than 0, the final setpoint will approach the setpoint at the rate defined by SPRatio when the setpoint changes. Calculated once per task cycle.
		SPInt	REAL	Follows data type	-	Internal setpoint.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
		SPTrack	BOOL	TRUE, FALSE	TRUE	The tracking switch of internal setpoint is on by default to ensure bumpless switchover when switching between external and internal setpoint sources. FALSE: Internal setpoint does not track external setpoint. TRUE: Internal setpoint tracks external setpoint.
	OutputConfig		PID_OUTPUT CONFIG	-	-	PID output configuration structure.
	OutputConfig	OutType	DINT	0, 1, 2	0	Output type: 0: PWM. 1: Analog output. 2: Percentage output.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
		OutMin	REAL	-100.0~OutMax	0.0	Minimum value of output percentage. The upper limit of the PID calculation output, and also the lower limit of the analog output conversion range.
		OutMax	REAL	OutMin~100.0	100.0	Maximum value of output percentage. The upper limit of the PID calculation output, and also the upper limit of the analog output conversion range.
		AOMin	DINT	Follows data type	0	Minimum value of analog output.
		AOMax	DINT	Follows data type	20000	Maximum value of analog output.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
		OutputOffset	REAL	-100.0 to 100.0	0.0	Compensation value of the output offset. Directly added to the PID calculation output, limited by the output min and max values.
		MinPT	DINT	0 to 10000	0	Minimum on-off pulse time in ms. Active in PWM mode.
	SuperConfig		PID_SUPERCONFIG	-	-	Advanced PID configuration structure.
	SuperConfig	Fuzzy	BOOL	TRUE, FALSE	FALSE	Fuzzy control enable: 0: Disable fuzzy self-adaptation. 1: Enable fuzzy self-adaptation.
		ECMax	REAL	Follows data type	1.0	Fuzzy control, maximum heating rate, °C/s.
		IntegralSuspend	BOOL	TRUE, FALSE	FALSE	Integral pause: 0: Normal integral. 1: Integral paused.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
		IntegralDivision	BOOL	TRUE, FALSE	TRUE	Integral separation for non-temperature objects: 0: No integral separation. 1: Integral separation.
		IntegralClear	BOOL	TRUE, FALSE	FALSE	Integral clearance: 0: Normal integral. 1: Integral cleared.
		IntegralMax	REAL	-100.0 to 100.0	100.0	Maximum integral value / the upper limit of the accumulated integral value.
		IntegralCut	REAL	0.0 to 1.0	0.75	Cut the integral coefficient when the setpoint changes.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
		KdGain	REAL	0.0 to 1.0	0.65	Incomplete derivative coefficient (0.0 to 1.0). The larger the value, the smoother the differential effect. When set to 1, the differential action is disabled.
		IntegralDivLimit	REAL	0.0 to 100.0	50.0	Integral separation limit. When the value of IntegralDivision is TRUE for non-temperature objects, integral action is disabled if the deviation is greater than the integral separation limit.
		DeadBand	REAL	0.0 to 100.0	0.0	Deviation deadzone limit

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input/Output	StartAT	Auto-tuning switch	BOOL	TRUE, FALSE	FALSE	TRUE: Start tuning. After parameter tuning is completed, it will reset to FALSE and can be triggered multiple times.
	Kp	Proportional coefficient	REAL	0.01 ~ 1000.0	60.0	Proportional coefficient. Configuration error will report an error ID. The larger the Kp value, the stronger the proportional action.
	Ti	Integral time	REAL	0.0 to 10000.0	400.0	Integral time. Configuration error will report an error ID. 0 indicates no integral action. The larger the Ti value, the weaker the integral action.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Td	Derivative time	REAL	0.0 to 10000.0	50.0	Derivative time. Configuration error will report an error ID. 0 indicates no derivative action. The larger the Td value, the stronger the derivative action.
Output	OutPWM	PWM output	BOOL	TRUE, FALSE	FALSE	PWM output.
	OutPercent	Percentage output	REAL	-100.0%~100.0%	0.0	Percentage output in %.
	OutAO	Analog output	DINT	Follows data type	0	Analog output.
	PIDOutput	PID output variable structure	PID_Output	-	-	Other PID output variables.

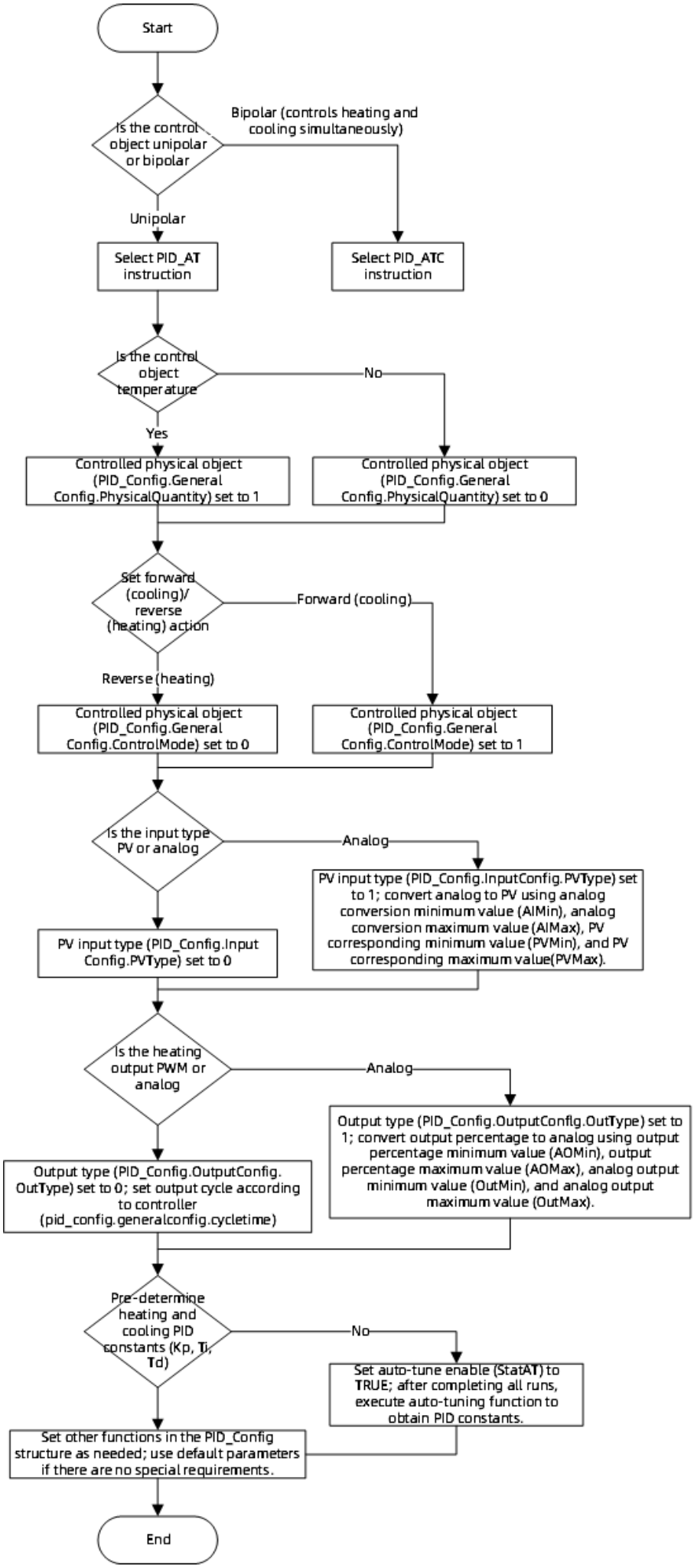
Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description

Name	Data Type	Valid Range	Default Value	Description
PVOut	REAL	Follows data type	0.0	Actually calculated PV value.
ECMaxOut	REAL	Follows data type	0.0	Fuzzy control, maximum heating rate, ℃/s.
ATState	DINT	0, 1, 2	0	Auto-tuning state 0: Not tuned. 1: Tuning. 2: Tuned.
ErrorID	DINT	Follows data type	0	Parameter fault code.
SPOut	REAL	Follows data type	0.0	Actually calculated SP value.

State	Operation state	DINT	Follows data type	0	Operation state: 0: Not activated. 1: Manual. 2: Automatic. 3: Interlocked. 4: Tuning. 5: Last valid output on error. 6: Substitute value output on error. 7: Master tracking.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Error	Operation error	BOOL	TRUE, FALSE	FALSE	Operation error marking, see ErrorBits for details.
	ErrorBits	Fault code	WORD	-	-	Fault code, expressed by bits: bit0 = 1: PV overlimit. bit1 = 1: Parameter error (For detailed information about the parameter error, see PIDOutput.ErrorID.)
	SlaveState	Slave status	Slave_Data	-	-	In case of slave, the structure of the operation state must be connected back to CasSlaveState, the input of the master control.

- **Parameter initialization flow for PID_AT**


Start
Is the control object unipolar or bipolar
Bipolar (controls heating and cooling simultaneously)
Unipolar
Select PID_AT instruction
Select PID_ATC instruction
Is the control object temperature
No
Yes
Controlled physical object (PID_Config.General Config.PhysicalQuantity) set to 1
Controlled physical object (PID_Config.General Config.PhysicalQuantity) set to 0
Set forward (cooling)/ reverse (heating) action
Forward (cooling)
Reverse (heating)
Controlled physical object (PID_Config.General Config.ControlMode) set to 0
Controlled physical object (PID_Config.General Config.ControlMode) set to 1
Is the input type PV or analog
Analog
PV input type (PID_Config.InputConfig.PVType) set to 1; convert analog to PV using analog conversion minimum value (AIMin), analog conversion maximum value (AIMax), PV corresponding minimum value (PVMin), and PV corresponding maximum value(PVMax).
PV input type (PID_Config.Input Config.PVType) set to 0
Is the heating output PWM or analog
Analog
Output type (PID_Config.OutputConfig.OutType) set to 1; convert output percentage to analog using output percentage minimum value (AOMin), output percentage maximum value (AOMax), analog output minimum value (OutMin), and analog output maximum value (OutMax).
Output type (PID_Config.OutputConfig. OutType) set to 0; set output cycle according to controller (pid_config.generalconfig.cycletime)
Pre-determine heating and cooling PID constants (Kp, Ti, Td)
No
Set auto-tune enable (StatAT) to TRUE; after completing all runs, execute auto-tuning function to obtain PID constants.
Set other functions in the PID_Config structure as needed; use default parameters if there are no special requirements.
End

- **Parameter description**
  - StartAT auto-tuning

    This instruction has an auto-tuning function, which can automatically calculate the optimal values of the PID constants Kp, Ti, Td.

    Auto-tuning execution conditions for heating applications:

    1) The current value is less than the setpoint. Generally, the setpoint should be the common operating temperature of the equipment.

    2) When StartAT is set to TRUE: If Enable changes from FALSE to TRUE, auto-tuning will be executed at the start of operation. If Enable is TRUE and PID calculation is in progress, auto-tuning will start during operation when StartAT changes from FALSE to TRUE.

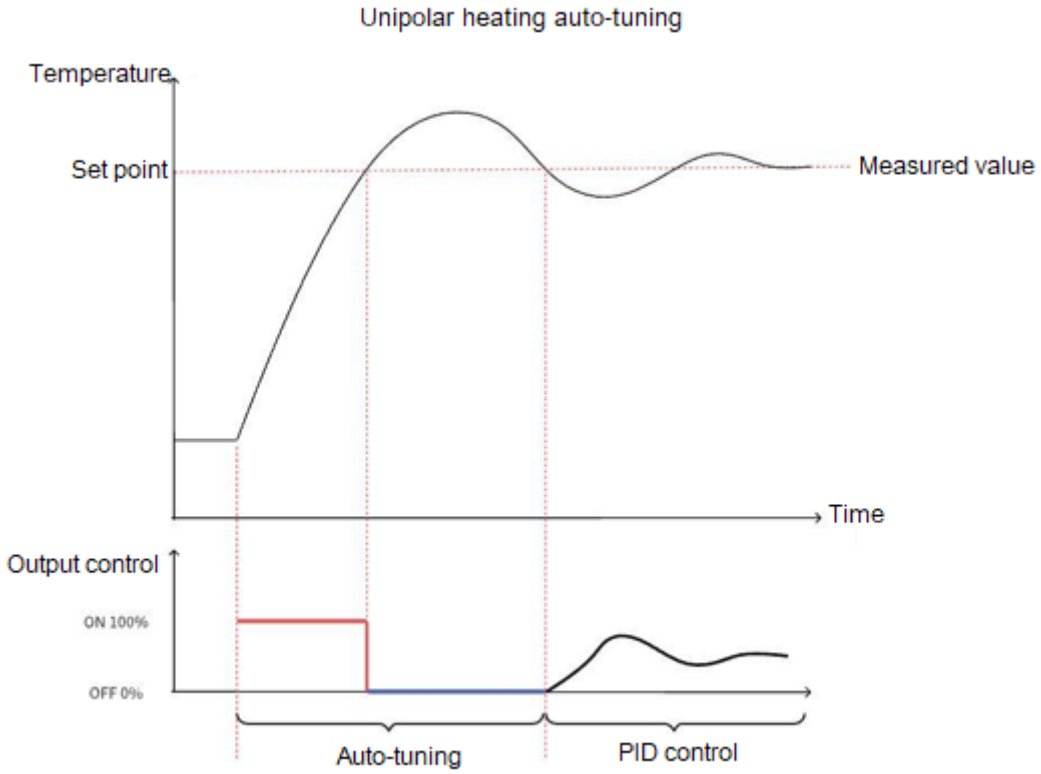
    After auto-tuning starts, the value of State.1 will be set to 1. When auto-tuning is completed, the value of State.1 will be set to 0 and the value of State.2 will be set to 1.

    In any of the above cases, if StartAT changes from TRUE to FALSE during auto-tuning, the auto-tuning process will be terminated, with State.1 = 0 and State.2 = 0.

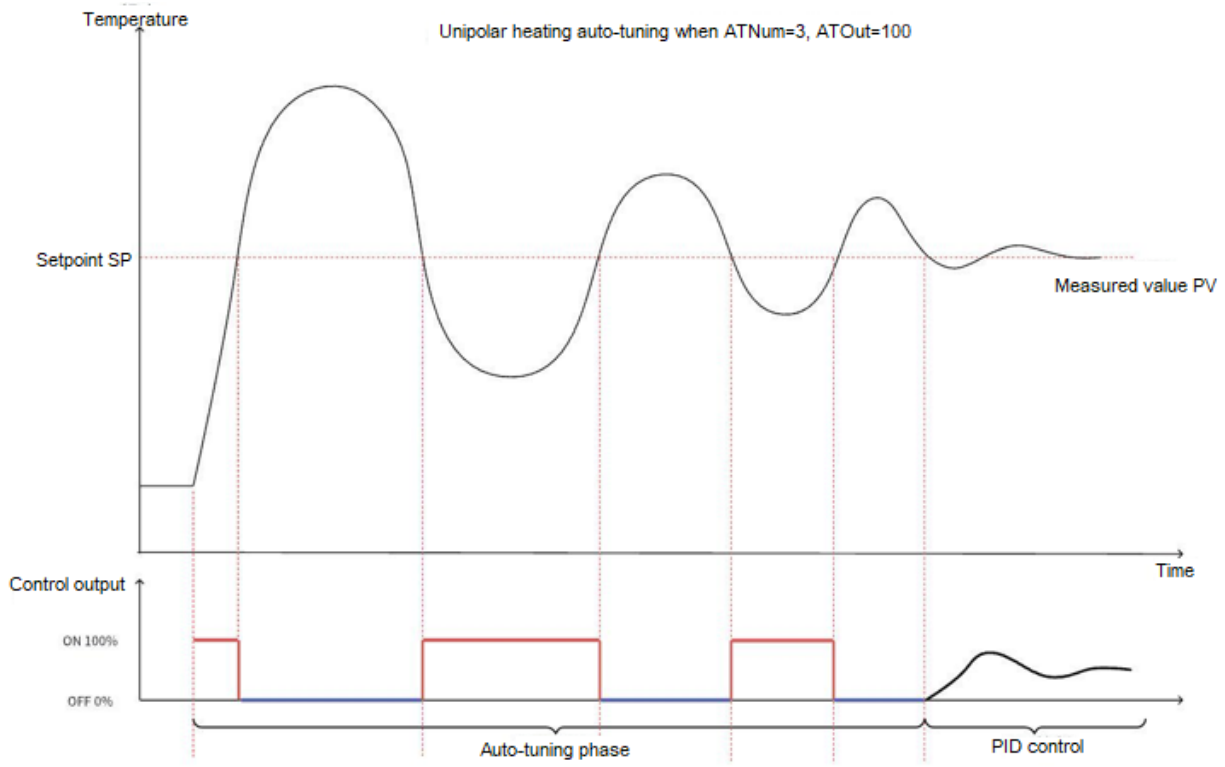
    Note: After auto-tuning is completed, StartAT will automatically be set to FALSE.

    3) Auto-tuning output percentage ATOut: Used to set the heating/cooling output percentage during the auto-tuning phase, supports only unipolar mode.
    Number of auto-tuning waveforms ATNum: Auto-tuning is performed with multiple oscillation wavefroms based on the oscillation parameter ATNum.

  - When ATOut = 100 and ATNum = 1, the schematic diagram of the unipolar auto-tuning process is shown below.



  - When ATOut = 100 and ATNum = 3, the calculation is performed based on 3 oscillation parameters, and the schematic diagram of the unipolar auto-tuning process is shown below.



NOTE

LoopControl 1.1.0.0 and later versions supports multi-waveforms auto-tuning algorithm.

- **Function**

  A single circuit, also known as a simple control system, refers to a closed-circuit feedback control system composed of one controlled object, a sensor/transmitter, a controller, and an actuator. It applies to scenarios with low time lag, slight load/interference fluctuations and moderate control quality requirements.

  - **Calculation formula**

$$\mathbf{AV} = K_p * \left[(SP - PV) + \frac{1}{T_i * S}(SP - PV) + \frac{T_d}{S}(PV) \right]$$

  - **Discretization formula:**

$$\mathbf{AV(k)} = K_p * \mathbf{E}(k) + K_p * \frac{cycle}{T_i} \sum \mathbf{E}(k) + \frac{K_p * T_D}{cycle}(P(k) - \mathbf{P}(k-1))$$

  - **Operation mode**

Operation Mode	Action
Automatic	Calculate according to the proportional-integral-derivative (PID) principle and generate the output value of the PID module. The calculation result of the PID module is limited by "output amplitude".

Operation Mode	Action
Manual	Directly control the output (value written by the operator or online logic). The output in this mode is limited by "output amplitude".
Interlock	In this mode, the PID stops automatic calculation, and the output value of the PID module is a preset value "InlockVal", generally referred to as the interlock setpoint. The interlock function can also be used as a "tracking" function. When certain conditions are met (interlock "InlockSW" = TRUE), the PID module automatically switches to this mode. When the interlock conditions are not met, the PID module switches back to the previous mode.

1. Interlock mode has the highest priority, followed by manual mode, and then automatic mode.
2. Automatic mode: Primarily used for setpoint control. The setpoint can be configured via the interface.
3. The selection of external or internal setpoint can be configured by the parameter PIDConfig.InputConfig.SPSel:

   SPSel = FALSE: External setpoint (the setpoint configured in the logic configuration).

   SPSel = TRUE: Internal setpoint (the preset internal setpoint).

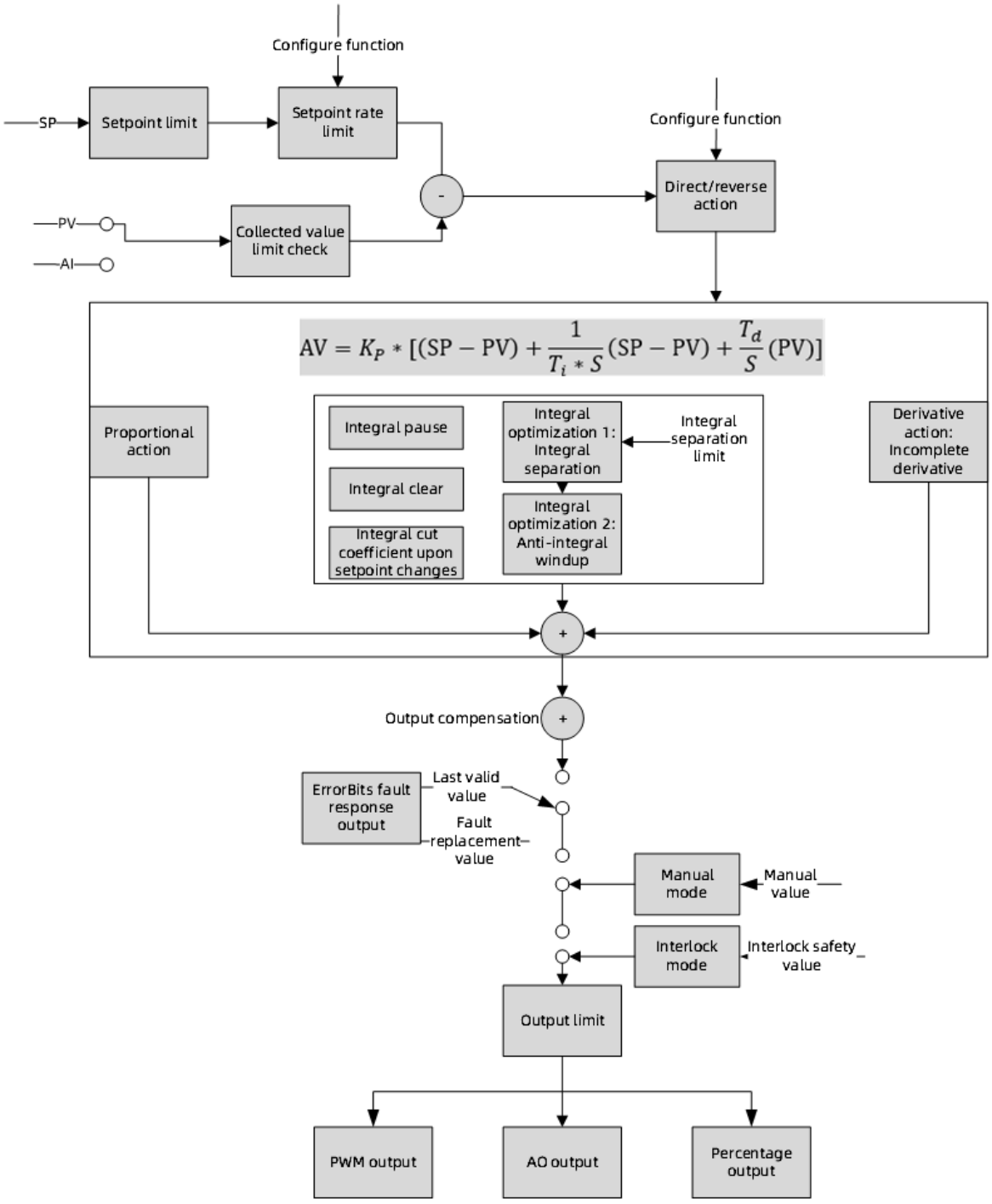
   Note: Enable "Internal setpoint tracks external setpoint" (PIDConfig.InputConfig.SPTrack = TRUE) to prevent loop fluctuations during setpoint switching. When external setpoint is selected, internal setpoint tracks it for bumpless switchover. Switching from internal to external setpoint directly adopts the real-time logic configuration value, which may cause bumpy switchover.

◦ **Bumpless switchover**

Operation Mode	Action
Interlock -> Manual	In interlock mode, the interlock value is output directly. The manual value tracks the output value to ensure bumpless switchover.
Interlock -> Automatic	In interlock mode, the interlock value is output directly. When switching to automatic mode, if the external setpoint is used, the logic calculation value will be adopted directly for PID calculation and output, which may cause disturbance.
Manual -> Interlock	The manual output is the manual value. Switching to interlock mode will output the interlock value directly, resulting in bumpy switchover.

Operation Mode	Action
Manual -> Automatic	The manual output is the operator-set value. When switching to automatic mode, if the external setpoint is used, the logic calculation value will be adopted directly for PID calculation and output, enabling bumpless switchover (disturbance is limited to the incremental part).
Automatic->Interlock	In automatic mode, the output is the normal calculation value. Switching to interlock mode will output the interlock value directly, resulting in bumpy switchover.
Automatic->Manual	In automatic mode, the output is the normal calculation value. The manual value tracks the output value to ensure bumpless switchover

- **Functional logic diagram (reverse-acting example)**



- **Supports direct/reverse action**

  Direct and reverse action: In automatic mode, direct action means output variation matches feedback variation; reverse action means the opposite. Simply put, direct action requires increasing output to eliminate deviation when feedback exceeds setpoint, while reverse action does the opposite. Configurable via parameter "PIDConfig.GeneralConfig.ControlMode":

  ControlMode = 0: Reverse action, such as heating control.

  ControlMode = 1: Direct action, such as cooling control.

- **Setpoint change limit function**

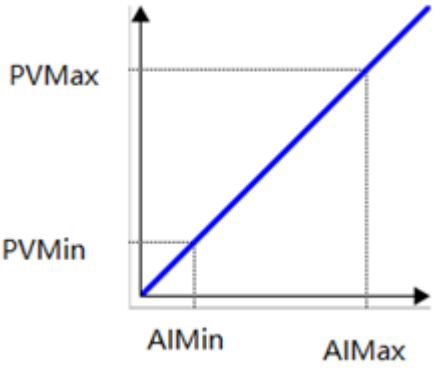
  In automatic mode, the change in setpoint (external and internal) is detected. If the change exceeds the predefined limit (PIDConfig.InputConfig.SPRatio), the actual setpoint used for calculation is limited to the change, and the setpoint will gradually approach the target value within the change limit range.

△SP > SPRatio (setpoint change): SP = LastSP + SPRatio

△SP < SPRatio (setpoint change): SP = LastSP - SPRatio

◦ **Input selection function**

Two modes of input signals can be selected.

PIDConfig.InputConfig.PVType	Input	Description
0	Current input "PV"	-
1	Analog input "AI"	Calibrate via linear conversion based on analog input limits and PV limits.  Analog upper limit - PIDConfig.InputConfig.AIMax Analog lower limit - PIDConfig.InputConfig.AIMin PV upper limit - PIDConfig.InputConfig.AIMax PV upper limit - PIDConfig.InputConfig.AIMin

◦ **Integral function**

Integral separation

Integral windup is mainly caused by the integral term. To overcome this issue, it is crucial to limit integral action and prevent excessive integral accumulation. The integral separation function can eliminate integral windup and enable the function block to resist integral windup.

For non-temperature-controlled quantities, the configuration of integral separation is described in the table below.

Integral separation enable PIDConfig.GeneralConfig.IntegralDivision	Integral separation limit PIDConfig.GeneralConfig.IntegralDivLimit
TRUE	|Deviation| >= Integral Separation Limit: Integral component calculation is disabled (control law operates as PD control).
	|Deviation| < Integral Separation Limit: Full integral calculation is enabled (control law operates as PID control).
FALSE	Full integral calculation is always enabled.

Integral pause

When integral suspend is triggered "PIDConfig.GeneralConfig.IntegralSuspend = TRUE", the integral action remains unchanged and stops accumulating.

Integral clearance

When integral clear is triggered "PIDConfig.GeneralConfig.IntegralClear = TRUE", integral action will be cleared to 0.

Integral cut coefficient

When the setpoint changes, the initial integral action can be reduced by setting the integral cut coefficient "PIDConfig.SuperConfig.IntegralCut". The setting range is [0,1], default is 0.75.

- **Incomplete derivative function**

  Incomplete derivative is equivalent to adding a first-order inertial element to the derivative action, which moderates the derivative effect and effectively improves the system's control performance.

Configuration parameter: PIDConfig.SuperConfig.KdGain	Derivative action
0	Derivative action is only active for the current cycle (default value).
(0,1)	The larger the value, the smoother the derivative action.
1	The derivative action is disabled.

- **Deadzone control**

  Supports width configuration of control dead zone via "PIDConfig.SuperConfig.DeadBand". Within the dead zone, the system is deemed stable with unchanged PID output; beyond it, PID operates normally.

- **Delay start/delay shutdown function**

  For multi-channel device control, set different delayed startup/shutdown times to stagger on/off sequences. Prevent large current fluctuations caused by simultaneous multi-channel startup.

Delay start/shutdown time	Description
PIDConfig.GeneralConfig.DelayStartUp	After delaying the DelayStartUp time, the PID calculation starts.
PIDConfig.GeneralConfig.DelayShutdown	After delaying the DelayShutdown time, the PID calculation stops.

- **Output compensation function**

  The output compensation function directly adds the output compensation value to the result of the PID automatic calculation to form the output value. The output compensation function is active in automatic mode and inactive in manual mode.

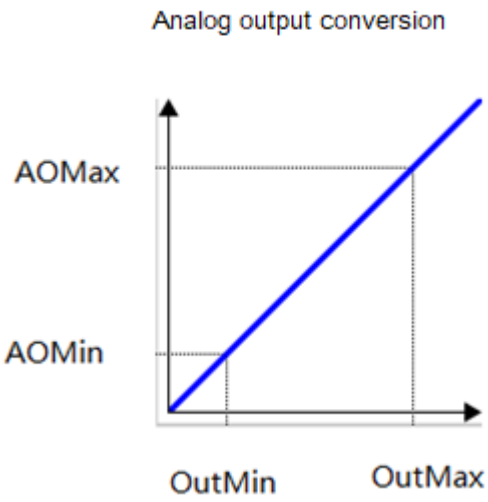
- **Output amplitude limit function**

  If output > "PIDConfig.OutputConfig.OutMax" (Output Upper Limit), output = OutMax.

If output < "PIDConfig.OutputConfig.OutMin" (OutputLower Limit), output = OutMin.

◦ **Output selection function**

Supports 3 modes of output control:

PIDConfig.OutputConfig.OutType	Output	Description
0	PWM output	Convert to switching output based on the PWM output cycle.
1	Analog output	Calibrate and output via linear conversion based on the analog output limits and PID output limits.  Analog output upper limit - PIDConfig.OutputConfig.AOMax Analog output lower limit - PIDConfig.OutputConfig.AOMin PID output upper limit - PIDConfig.OutputConfig.OutMax PID output lower limit - PIDConfig.OutputConfig.OutMin
2	Percentage output	Output within 0~100%

◦ **Fault detection**

If any bit of ErrorBits is triggered, Error is set to TRUE.

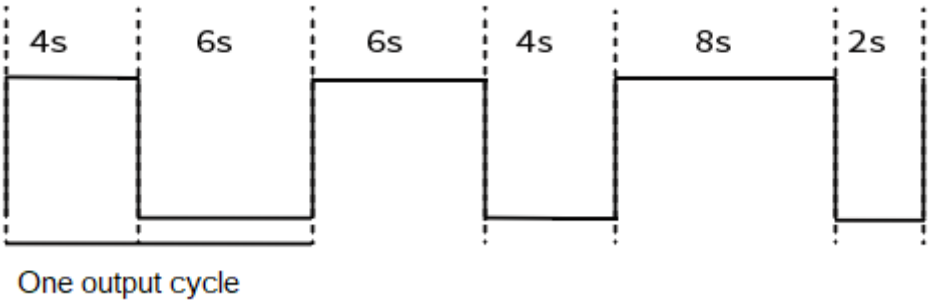
Bitwise description of ErrorBits		Error output configuration parameters PIDConfig.InputConfig.ErrorAction
Bit0 = 1	PV overlimit alarm	FALSE: Maintain previous output TRUE: Output the substitute value "PIDConfig.InputConfig.ErrorOut"
Bit1 = 1	Parameter configuration error alarm	

See ErrorID for detailed error information.

◦ **PWM output**

For PWM, the output is ON or OFF. The analog of PID output needs to be converted to digital output (ON/OFF) based on the duty cycle. The duty cycle is the percentage of the high-level pulse time to a whole pulse cycle.

For example: If the PID regulator output is 40% and the PWM output cycle "PIDConfig.GeneralConfig.CycleTime" is set to 10s, the ON duration is 10s × 40% = 4s, and the OFF duration is 10s – 4s = 6s. The duty cycle is 40%, meaning the signal stays ON for 4s and OFF for 6s per 10s cycle before repeating the calculation for the next cycle.

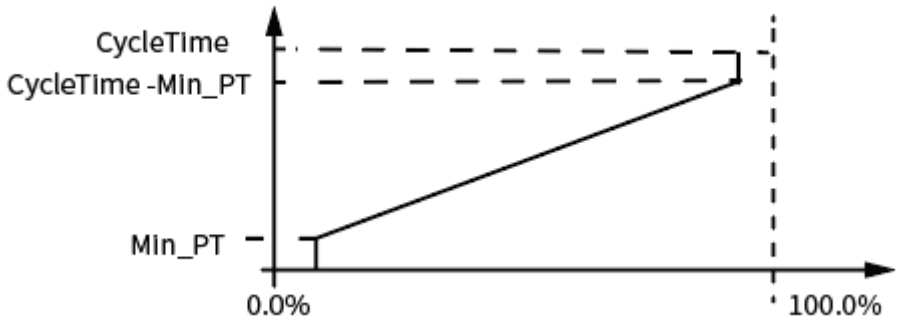


- Minimum On/Off pulse time

  To avoid frequent actuator operations, set the minimum on/off pulse time "PIDConfig.OutputConfig.Min_PT".

  If the calculated high-level pulse time < Min_PT, the output duty cycle is 0%.

  If the calculated high-level pulse time > PWM output cycle – Min_PT, the output duty cycle is 100%.



◦ **Cascade function**

Cascade control involves two or more cascaded PID controllers. The control variable of the master controller is interconnected with the setpoint of the slave controller, nesting the two control circuits. Cascade control has advantages that the slave circuit responds faster than the main circuit and can pre-compensate for interference in the inner circuit.

Note: Cascade control is only possible when there are other measurable variables in the process besides the main control variable, and the internal control circuit is much faster than the external circuit.

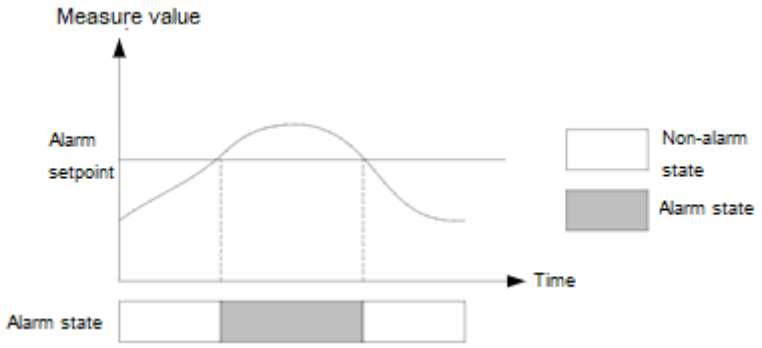
Bumpless switchover: If the slave circuit is not operating in cascade mode (automatic mode with external setpoint), the master circuit switches to tracking mode. The output value of the master circuit tracks the process value of the slave circuit to ensure a smooth transition to cascade mode.

Priority: Error response > Interlock > Manual > Master tracking mode (triggered by slave) > Automatic.

◦ **Upper and lower limit alarms**

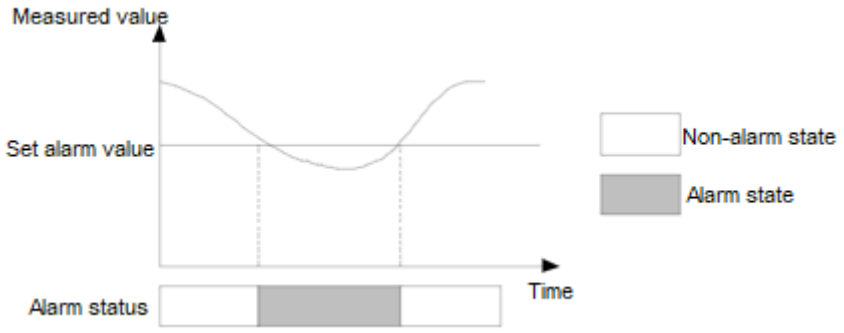
- Upper limit alarm of the input PV value

An alarm is triggered when the measured PV value (or PV value after analog conversion) exceeds the upper limit alarm value.



- Lower limit alarm of the input PV value

  An alarm is triggered when the measured PV value (or PV value after analog conversion) falls below the lower limit alarm value.



- **Description**

  1. First, select the controlled physical object GeneralConfig.PhysicalQuantity, which includes General, Temperature, Pressure, and Flow options. For temperature control, select Temperature which features multiple optimizations; for non-temperature control, select the corresponding quantity, and configure Integral Separation via IntegralDivision and IntegralDivLimit in Advanced Settings.
  2. Enable the auto-tuning function at room temperature. Once the auto-tuning ends, xStartAT automatically sets to FALSE, and the tuned parameters Kp, Ti, Td are generated. Continue program operation to check if the controlled quantity reaches a steady state. If the usage needs are met after the controlled quantity reaches steady state, record the values of Kp, Ti, and Td, and write these values into the program as defaults. The Kp, Ti, and Td values can also be manually adjusted to meet the usage requirements.
  3. The larger the auto-tuning coefficient rATCoefficient, the faster the set temperature can be reached, but it may cause greater overshoot. The smaller the auto-tuning coefficient, the slower the set temperature can be reached. However, the temperature curve will be very smooth, without overshoot or with very small overshoot. If there is no requirement to use the default value of 0.5 and no customer needs to reach the set temperature faster, increase rATCoefficient and re-run auto-tuning. If the customer needs a smooth curve and small overshoot, decrease rATCoefficient and re-run auto-tuning.
  4. Interlock function: When the interlock switch InlockSW is triggered, the interlock safety value InlockVal is output.
  5. Output priority: Error output > Interlock output > Manual output > Automatic output.

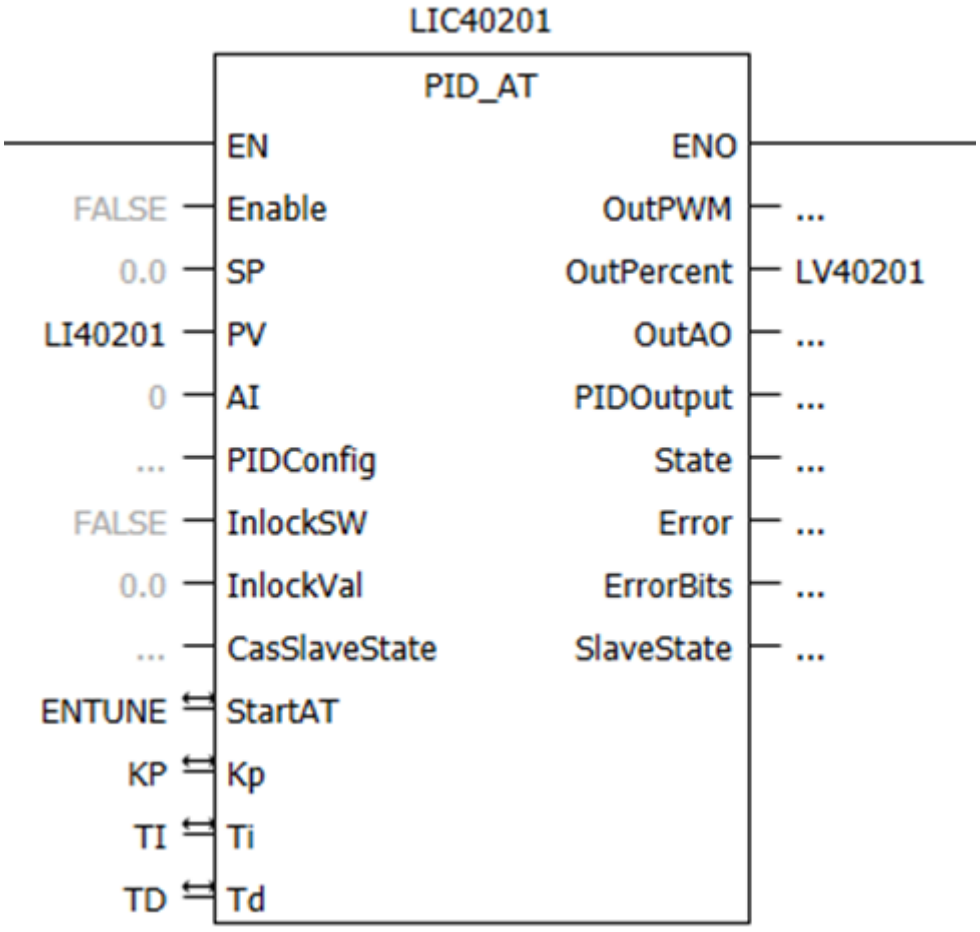
  Note:

  1. Always verify parameter correctness before running the PID.

2. Auto-tuning is recommended to start at room temperature.
3. For common operating temperatures of 150 to 200 °C, tuning can be performed at 175 °C. Select the customer's typical operating temperature for tuning to ensure parameter accuracy.
4. Perform auto-tuning first to obtain initial PID parameters, then adjust them appropriately based on control performance.

- **Application instance**

  Control scheme configuration:



  Example:

No.	Instance	Type
1	LIC40201	Liquid level controller
2	LI40201	AI detection signal of liquid level
3	LV40201	Liquid level control valve

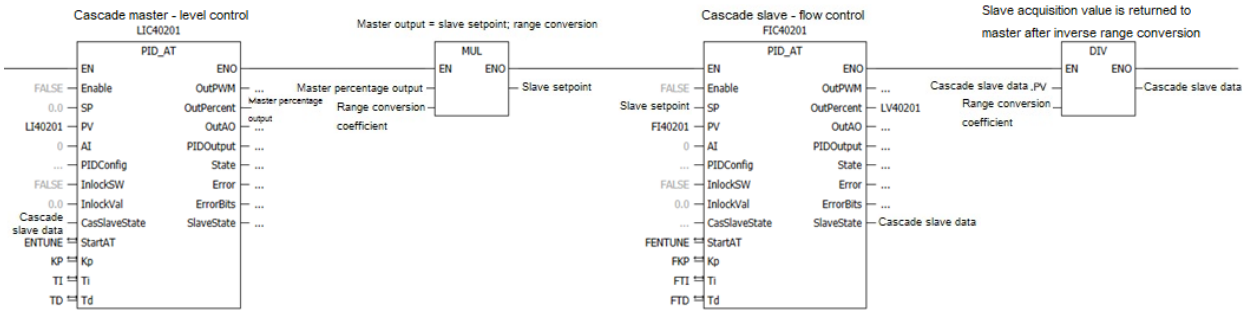
  Cascade control scheme configuration:

  LIC40201.PIDConfig.GeneralConfig.IsMaster = TRUE; // LIC40201 level control is master control.

  LIC40201.PIDConfig.GeneralConfig.PhysicalQuantity = Liquid Level; // The controlled quantity of LIC40201 is liquid level;

  FIC40201.PIDConfig.GeneralConfig. IsSlave = TRUE; // FIC40201 liquid level control is slave control.

FIC40201.PIDConfig.GeneralConfig.PhysicalQuantity = Flow; // The controlled quantity of LIC40201 is flow;



Example:

No.	Instance	Type
1	LIC40201	Liquid level controller
2	LI40201	AI detection signal of liquid level
3	FIC40201	Flow controller
4	LV40201	Flow control valve
5	FI40201	Flow AI detection signal

- **Fault codes (ErrorID)**

Grouping	Error No.	Fault Codes	Detailed Description
No error	0	NO_ERROR	No error
Wrong input interface parameters	1	KP_ERROR	Kp configuration error
	2	TI_ERROR	Ti configuration error
	3	TD_ERROR	Td configuration error
	4	KPCOOL_ERROR	Cooling Kp configuration error
	5	TICOOL_ERROR	Cooling Ti configuration error
	6	TDCOOL_ERROR	Cooling Ti configuration error
	7	COOL_COEFFICIENT_ERROR	Cooling coefficient - configuration error
General configuration	8	SAMPLE_TIME_ERROR	Sampling period - general configuration error

Grouping	Error No.	Fault Codes	Detailed Description
	9	CYCLE_TIME_ERROR	Output cycle - general configuration error
	10	PID_TYPE_ERROR	PID Type - general configuration error
	11	MODE_ERROR	PID control mode - general configuration error
	12	MANUAL_VALUE_ERROR	Manual output value - general configuration
	13	ATCOEFFICIENT_ERROR	Auto-tuning coefficient - general configuration error
	14	DELAY_STARTUP_ERROR	Delay startup - general configuration error
	15	DELAY_SHUTDOWN_ERROR	Delay shutdown - general configuration error
Input configuration	16	PVTYPE_ERROR	PV type - input configuration error
	17	AI_MIN_ERROR	Minimum value for analog conversion - input configuration error
	18	AI_MAX_ERROR	Maximum value for analog conversion - input configuration error
	19	PV_MIN_ERROR	Minimum value of PV - input configuration error
	20	PV_MAX_ERROR	Maximum value of PV - input configuration error
	21	PV_LOW_LIMIT_ERROR	Lower input limit - input configuration error
	22	PV_HIGH_LIMIT_ERROR	Upper input limit - input configuration error
	23	ERROR_OUT_ERROR	Alarm output substitute value - input configuration error

Grouping	Error No.	Fault Codes	Detailed Description
	24	PV_UNIT_ERROR	PV unit configuration error
Heating output configuration	25	OUT_TYPE_ERROR	Heating output type - output configuration error
	26	OUT_MIN_ERROR	Minimum heating output percentage - output configuration error
	27	OUT_MAX_ERROR	Maximum heating output percentage - output configuration error
	28	AO_MIN_ERROR	Minimum value of heating analog output - output configuration error
	29	AO_MAX_ERROR	Maximum value of heating analog output - output configuration error
	30	OUTPUT_OFFSET_ERROR	Compensation value of heating output - output configuration error
Cooling output configuration (applicable to PID_ATC)	31	COOL_OUT_TYPE_ERROR	Cooling output type - cooling output configuration error
	32	COOL_OUT_MIN_ERROR	Minimum cooling output percentage - cooling output configuration error
	33	COOL_OUT_MAX_ERROR	Maximum cooling output percentage - cooling output configuration error
	34	COOL_AO_MIN_ERROR	Minimum value of cooling analog output - Cooling output configuration error

Grouping	Error No.	Fault Codes	Detailed Description
	35	COOL_AO_MAX_ERROR	Maximum value of cooling analog output - Cooling output configuration error
	36	COOL_OUTPUT_OFFSET_ERROR	Compensation value of cooling output - cooling output configuration error
Advanced	37	ECMAX_ERROR	Maximum heating speed - advanced configuration error
	38	KD_GAIN_ERROR	Incomplete derivative coefficient - advanced configuration error
	39	INTEGRAL_MAX_ERROR	Maximum integral value - advanced configuration error
	40	INTEGRAL_CUT_ERROR	Reduction of integral coefficient upon setpoint change - advanced configuration error
Cooling configuration (applicable to PID_ATC)	41	COOL_PID_TYPE_ERROR	Cooling PID type - cooling configuration error
	42	SAMPLE_COOL_TIME_ERROR	Cooling sampling period - cooling configuration error
	43	CYCLE_COOL_TIME_ERROR	Cooling output cycle - cooling configuration error
	44	COOL_OFFSET_ERROR	Cooling set deviation - cooling configuration error

Grouping	Error No.	Fault Codes	Detailed Description
PV limit alarm	45	PV_LOW_LIMIT_ALARM	PV low limit alarm. Check the PIDConfig.InputConfig.PVLowLimit and PIDConfig.InputConfig.PVHighLimit parameters to ensure the PV value is within the limits.
	46	PV_HIGH_LIMIT_ALARM	PV upper limit alarm. Check PIDConfig.InputConfig.PVLowLimit and PIDConfig.InputConfig.PVHighLimit parameters to ensure the PV value is within the limits.
Input interface configuration error	47	PV_AND_AI_NOINPUT	PV and AI are not configured for input simultaneously
Input configuration error	48	SP_RAT_ERROR	Setpoint change setting error
Advanced configuration fault	49	INTEGRAL_DIV_ERROR	Integral separation limit setting error
Advanced configuration fault	50	DEADBAND_ERROR	The control deadzone is incorrectly configured.
Cooling output configuration error (applicable to PID_ATC)	51	COOL_MINPT_ERROR	The minimum ON/OFF time for cooling is incorrectly configured.
Output configuration error	52	OUT_MINPT_ERROR	The minimum ON/OFF time is incorrectly configured.
Wrong Input interface parameters	53	INLOCKVAL_ERR	Interlock safety value configuration error
Wrong Input interface parameters	54	CASSLAVESTATE_PV_ERR	PV overlimit in slave data
Wrong input interface parameter	55	CASSLAVESTATE_OVE_ERR	OVE configuration error in slave data

Grouping	Error No.	Fault Codes	Detailed Description
Wrong input interface parameters	56	CASSLAVESTATE_CASSTATE_ERR	Slave status configuration error in slave data
General configuration fault	57	AT_NUM_ERROR	Number of Auto-tuning waveforms configuration error
General configuration fault	58	AT_OUT_ERROR	Auto-tuning output percentage configuration error

- **Error handling**

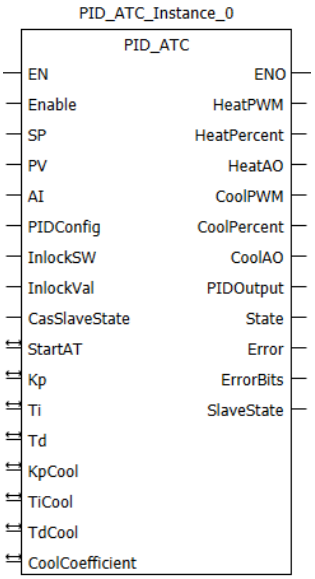
  Action on error alarm for PIDConfig.InputConfig.ErrorAction

  1. When PIDConfig.InputConfig.ErrorAction = FALSE, maintain previous output on error alarm. Recalculate current output upon error recovery.
  2. When PIDConfig.InputConfig.ErrorAction = TRUE, output ErrorOut substitute value on error alarm. Recalculate current output upon error recovery.

18.3.1.6 PID_ATC

Bipolar temperature PID controller.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PID_ATC	Bipolar temperature PID controller	FB		PID_ATC_Instance_0(Enable := , SP := , PV := , AI := , PIDConfig := , InlockSW := , InlockVal := , CasSlaveState := , StartAT := , Kp := , Ti := , Td := , KpCool := , TiCool := , TdCool := , CoolCoefficient := , HeatPWM => , HeatPercent => , HeatAO => , CoolPWM => , CoolPercent => , CoolAO => , PIDOutput => , State => , Error => , ErrorBits => , SlaveState =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Enable	Enable operation	BOOL	TRUE, FALSE	FALSE	Used as a reset switch. FALSE: PID block is disabled and output is cleared. TRUE: PID block is enabled and performs PV acquisition > PID calculation > Output of manipulated variable.
	SP	Set target value	REAL	Follows data type	0.0	Setpoint (target value).
	PV	Current process value	REAL	Follows data type	0.0	Feedback value (process value).
	AI	Analog input value	DINT	Follows data type	0	Analog input value.
	InlockSW	Interlock switch	BOOL	TRUE, FALSE	FALSE	FALSE: Off. TRUE: On.
	InlockVal	Interlock safety value	REAL	-100.0 to 100.0	0.0	Interlock safety value.
	CasSlaveState	Slave data	Slave_Data	Follows data type	-	Slave data structure.

<table>
<tr><th>Input/Output</th><th>Variable</th><th>Name</th><th>Data Type</th><th>Valid Range</th><th>Default Value</th><th>Description</th></tr>
<tr><td></td><td colspan="6">In case of master control, the slave data must be transmitted to this structure.
<table>
<tr><th>Name</th><th>Data Type</th><th>Valid Range</th><th>Default Value</th><th>Description</th></tr>
<tr><td>PV</td><td>REAL</td><td>Follows data type</td><td>0.0</td><td>Slave process value.
When the slave is detected to be in non-cascade automatic mode, the output of the master control tracks the slave process value.</td></tr>
<tr><td>OVE</td><td>BYTE</td><td>0, 1, 2</td><td>0</td><td>Overlimit status of slave circuit:
0: Normal.
1: Upper limit reached.
2: Lower limit reached.</td></tr>
<tr><td>CAS_State</td><td>UINT</td><td>0, 1</td><td>0</td><td>Cascade use mode:
0: Slave non-automatic mode.
1: Slave automatic mode. In case of slave, the value should be returned to the master controller SUBCAS_MODE.</td></tr>
</table></td></tr>
<tr><td></td><td>PIDConfig</td><td>PID parameter configuration structure</td><td>PID_ATC_CONFIG</td><td>-</td><td>-</td><td>PID parameter configuration structure</td></tr>
</table>

<table>
<tr><th>Input/Output</th><th>Variable</th><th>Name</th><th>Data Type</th><th>Valid Range</th><th>Default Value</th><th>Description</th></tr>
<tr><td>Input</td><td colspan="6">
<table>
<tr><th>Variable</th><th>Name</th><th>Data Type</th><th>Valid Range</th><th>Default Value</th><th>Description</th></tr>
<tr><td colspan="2">GeneralConfig</td><td>PID_GENERALCONFIG</td><td>-</td><td>-</td><td>PID general configuration structure.</td></tr>
<tr><td rowspan="3">GeneralConfig</td><td>SampleTime</td><td>DINT</td><td>1 to 1000000</td><td>100</td><td>Sampling time in ms.
That is the PID calculation cycle.</td></tr>
<tr><td>CycleTime</td><td>DINT</td><td>1 to 1000000</td><td>1000</td><td>PWM output cycle in ms.
Takes effect when the output mode is set to PWM mode.</td></tr>
<tr><td>PIDType</td><td>DINT</td><td>0, 1, 2, 3</td><td>0</td><td>PID type:
0: PID calculation.
1: PI calculation.
2: PD calculation.
3: ID calculation.</td></tr>
</table>
</td></tr>
</table>

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
		ControlMode	DINT	0, 1, 2, 3	0	PID control mode: 0: Unipolar Heating. 1: Unipolar Cooling. 2: Bipolar with cooling coefficient control. 3: Bipolar with independent PID control. If other values are input, a parameter error will be triggered.
		ManualEnable	BOOL	TRUE, FALSE	FALSE	Manual mode enable, TRUE: Manual mode is enabled.
		ManualValue	REAL	OutMin~OutMax	0.0	Output percentage in manual mode, unit: %.
		ATCoefficient	REAL	0.0 to 1.0	0.5	Self-tuning adjustment coefficient. A larger value results in faster adjustment, with a default setting of 0.5.
		DelayStartUp	DINT	0 to 1000000	0	Delay startup time in ms.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
		DelayShutDown	DINT	0 to 1000000	0	Delay shutdown time in ms.
		PhysicalQuantity	INT	0 to 3	0	Physical object to control: 0: General. 1: Temperature and other parameters. For details, see object description. Default: General
		PhysicalUnit	DINT	Follows data type	0	Unit of the physical object.
		IsMaster	BOOL	TRUE, FALSE	FALSE	Master control: 0: No. 1: Yes. Value must be assigned during cascade configuration.
		IsSlave	BOOL	TRUE, FALSE	FALSE	Slave control: 0: No. 1: Yes. Value must be assigned during cascade configuration.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
		ATNum	DINT	0 to 20	2	Number of auto-tuning waveforms (applicable only to unipolar). 0: Reserved. 1-20: Number of oscillation waveforms.
		ATOut	DINT	OutMin~OutMax	100	Output percentage in auto-tuning (applicable only to unipolar, unit: %).
	InputConfig		PID_INPUTCONFIG	-	-	PID input configuration structure.
	InputConfig	PVType	DINT	0, 1	0	PV type: 0: Use current PV as input. 1: Use peripheral analog signal as input, with conversion based on analog range.
		AIMin	DINT	Follows data type	0	Minimum value for analog conversion.
		AIMax	DINT	Follows data type	20000	Maximum value for analog conversion.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
		PVMin	REAL	Follows data type	0.0	Minimum value of PV
		PVMax	REAL	Follows data type	100.0	Maximum value of PV.
		PVLowLimit	REAL	Follows data type	-100.0	Lower input limit.
		PVHighLimit	REAL	Follows data type	1000.0	Upper input limit.
		ErrorAction	BOOL	TRUE, FALSE	FALSE	Output on error alarm: FALSE: Maintain previous value. TRUE: Output substitute value.
		ErrorOut	REAL	OutMin~OutMax	0.0	Output of a predefined percentage value on error alarm.
		SPSel	BOOL	TRUE, FALSE	FALSE	Setpoint source selection: FALSE: External setpoint (SP configuration). TRUE: Internal setpoint (panel operation setpoint).

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
		SPRatio	REAL	Follows data type	0.0	SP setpoint change. If set to 0, no change judgment is performed. If set greater than 0, the final setpoint will approach the setpoint at the rate defined by SPRatio when the setpoint changes. Calculated once per task cycle.
		SPInt	REAL	Follows data type	0.0	Internal setpoint.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
		SPTrack	BOOL	TRUE, FALSE	TRUE	The tracking switch of internal setpoint is on by default to ensure bumpless switchover when switching between external and internal setpoint sources. FALSE: Internal setpoint does not track external setpoint. TRUE: Internal setpoint tracks external setpoint.
	OutputConfigHeat		PID_OUTPUT CONFIG	-	-	Output configuration structure of PID heating.
	OutputConfigHeat	OutType	DINT	0, 1, 2	0	Output type: 0: PWM. 1: Analog output. 2: Percentage output.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
		OutMin	REAL	-100.0~OutMax	0.0	Minimum value of output percentage. The lower limit of the PID calculation output, which is also the range upper limit for analog output conversion.
		OutMax	REAL	OutMin~100.0	100.0	Maximum value of output percentage. The upper limit of the PID calculation output, and also the upper limit of the analog output conversion range.
		AOMin	DINT	Follows data type	0	Minimum value of analog output.
		AOMax	DINT	Follows data type	20000	Maximum value of analog output.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
		OutputOffset	REAL	-100.0 to 100.0	0.0	Compensation value of the output. Directly added to the PID calculation output, limited by the output min and max values.
		MinPT	DINT	0 to 10000	0	Minimum on-off pulse time in ms. Active in PWM mode.
	OutputConfigCool		PID_OUTPUT CONFIG	-	-	Output configuration structure of PID cooling.
	OutputConfigCool	OutType	DINT	0, 1, 2	0	Output type: 0: PWM. 1: Analog output. 2: Percentage output.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
		OutMin	REAL	-100.0~OutMax	0.0	Minimum value of output percentage. The upper limit of the PID calculation output, and also the lower limit of the analog output conversion range.
		OutMax	REAL	OutMin~100.0	100.0	Maximum value of output percentage. The upper limit of the PID calculation output, and also the upper limit of the analog output conversion range.
		AOMin	DINT	Follows data type	0	Minimum value of analog output.
		AOMax	DINT	Follows data type	20000	Maximum value of analog output.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
		OutputOffset	REAL	-100.0 to 100.0	0.0	Compensation value of the output. Directly added to the PID calculation output, limited by the output min and max values.
		MinPT	DINT	0 to 10000	0	Minimum on-off pulse time in ms. Active in PWM mode.
	CoolConfig		PID_COOLCONFIG	-	-	PID cooling configuration structure.
	CoolConfig	CoolPIDType	DINT	0, 1, 2, 3	0	Cooling PID type: 0: PID 1: PI 2: PD 3: ID
		CoolSampleTime	DINT	1 to 1000000	100	Sampling time in ms.
		CoolCycleTime	DINT	1 to 1000000	5000	PWM cooling output cycle.
		CoolOffset	REAL	-1000.0 to 1000.0	0.0	Compensation value of the cooling output.
	SuperConfig		PID_SUPERCONFIG	-	-	Advanced PID configuration structure.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
	SuperConfig	Fuzzy	BOOL	TRUE, FALSE	FALSE	Fuzzy control enable: 0: Disable fuzzy self-adaptation. 1: Enable fuzzy self-adaptation.
		ECMax	REAL	Follows data type	1.0	Fuzzy control, maximum heating rate, °C/s.
		IntegralSuspend	BOOL	TRUE, FALSE	FALSE	Integral pause: 0: Normal integral. 1: Integral paused.
		IntegralDivision	BOOL	TRUE, FALSE	TRUE	Integral separation for non-temperature objects: 0: No integral separation. 1: Integral separation.
		IntegralClear	BOOL	TRUE, FALSE	FALSE	Integral clearance: 0: Normal integral. 1: Integral cleared.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
		IntegralMax	REAL	-100.0 to 100.0	100.0	Maximum integral value / the upper limit of the accumulated integral value.
		IntegralCut	REAL	0.0 to 1.0	0.75	Cut the integral coefficient when the setpoint changes.
		KdGain	REAL	0.0 ~ 1.0	0.65	Incomplete derivative coefficient. The larger the value, the smoother the differential effect. When set to 1, the differential action is disabled.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Variable	Name	Data Type	Valid Range	Default Value	Description
		IntegralDivLimit	REAL	0.0 to 100.0	50.0	Integral separation limit. When the value of IntegralDivision is TRUE for non-temperature objects, integral action is disabled if the deviation is greater than the integral separation limit.
		DeadBand	REAL	0.0 to 100.0	0.0	Deviation deadzone limit
Input/Output	StartAT	Auto-tuning switch	BOOL	TRUE, FALSE	FALSE	TRUE: Start tuning. After parameter tuning is completed, it will reset to FALSE and can be triggered multiple times.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Kp	Proportional coefficient	REAL	0.01 ~ 1000.0	60.0	Proportional coefficient. Configuration error will report an error ID. The larger the Kp value, the stronger the proportional action.
	Ti	Integral time	REAL	0.0 ~100000.	400.0	Integral time. Configuration error will report an error ID. 0 indicates no integral action. The larger the Ti value, the weaker the integral action.
	Td	Derivative time	REAL	0.0 ~ 10000.0	50.0	Derivative time. Configuration error will report an error ID. 0 indicates no derivative action. The larger the Td value, the stronger the derivative action.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	KpCool	Cooling proportional coefficient	REAL	0.01 ~ 1000.0	0.0	Cooling proportional coefficient, an error ID will be reported if configured incorrectly. The larger the KpCool value, the stronger the proportional action.
	TiCool	Cooling integral time	REAL	0.0 ~ 10000.0	0.0	Cooling integral time, configuration error will report an error ID. 0 indicates no integral action. The larger the TiCool value, the weaker the integral action.
	TdCool	Cooling derivative time	REAL	0.0 ~ 10000.0	0.0	Cooling derivative time, configuration error will report an error ID. 0 indicates no derivative action. The larger the TdCool value, the stronger the derivative action.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	CoolCoefficient	Cooling coefficient	REAL	0.0 to 100.0	0.1	Cooling coefficient, When mode GeneralConfig. ControlMode = 2, cooling output is calculated based on heating control coefficient and cooling coefficient. 0 indicates no cooling action.
Output	HeatPWM	Heating PWM output	BOOL	TRUE, FALSE	FALSE	PWM heating output. TRUE: On. FALSE: Off.
	HeatPercent	Heating percentage output	REAL	0.0 to 100.0 %	0.0	Heating output percentage (depends on output type selection).
	HeatAO	Heating analog output	DINT	Follows data type	0	Heating output analog value.
	CoolPWM	PWM cooling output	BOOL	TRUE, FALSE	FALSE	Cooling PWM output. TRUE: On. FALSE: Off.
	CoolPercent	Cooling percentage output	REAL	0.0 to 100.0 %	0.0	Cooling output percentage (depends on output type selection).

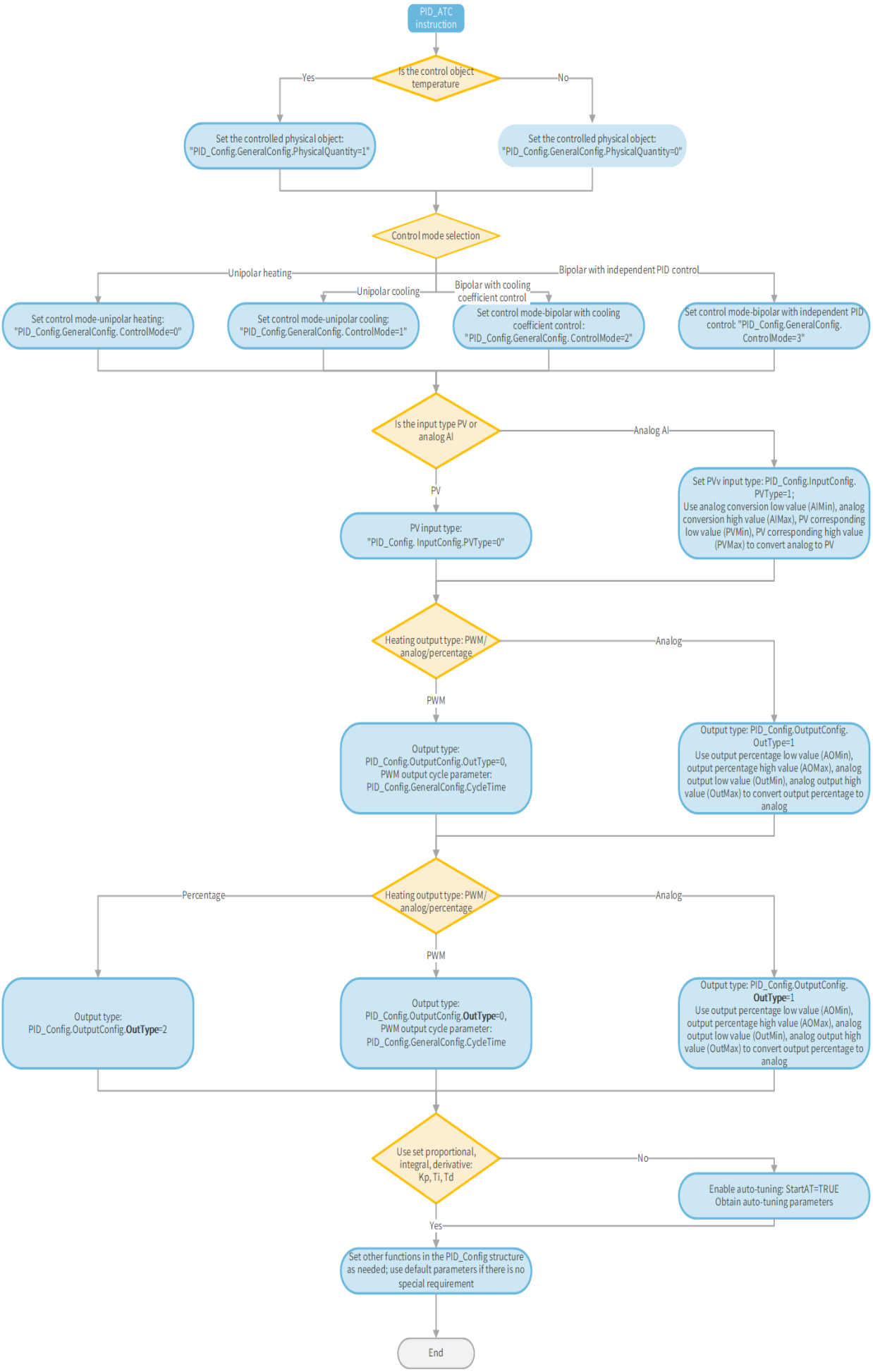
Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	CoolAO	Cooling analog output	DINT	Follows data type	0	Cooling output analog value.
	PIDOutput	PID output variable structure	PID_OUTPUT	-	-	Output structure.

Name	Data Type	Valid Range	Default Value	Description
PVOut	REAL	Follows data type	0.0	Actually calculated PV value.
ECMaxOut	REAL	Follows data type	0.0	Fuzzy control, maximum heating rate, ℃/s.
ATState	DINT	0, 1, 2	0	Auto-tuning state 0: Untuned. 1: Tuning. 2: Tuned.
ErrorID	DINT	Follows data type	0	Parameter fault code.
SPOut	REAL	Follows data type	0.0	Actually calculated SP value.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	State	Operation state	DINT	Follows data type	0	Operation state: 0: Not activated. 1: Manual. 2: Automatic. 3: Interlocked. 4: Tuning. 5: Last valid output on error. 6: Substitute value output on error. 7: Master tracking.
	Error	Operation error	BOOL	TRUE, FALSE	FALSE	Operation error marking, see ErrorBits for details.
	ErrorBits	Fault code	WORD	Follows data type	0	Fault code, expressed by bits: bit0 = 1: PV overlimit. bit1 = 1: Parameter error (For detailed information about the parameter error, see PIDOutput.ErrorID.)

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	SlaveState	Slave status	Slave_Data	-	-	In case of slave, the structure of the operation state must be connected back to CasSlaveState, the input of the master control.

- **Parameter initialization flow for PID_ATC**



- **Parameter description**
  - StartAT auto-tuning

This instruction has an auto-tuning function, which can automatically calculate the optimal values of the PID constants Kp, Ti, Td.

Auto-tuning execution conditions for heating applications:

1) The current value is less than the setpoint. Generally, the setpoint should be the common operating temperature of the equipment.

2) When StartAT is set to TRUE: If Enable changes from FALSE to TRUE, auto-tuning will be executed at the start of operation. If Enable is TRUE and PID calculation is in progress, auto-tuning will start during operation when StartAT changes from FALSE to TRUE.

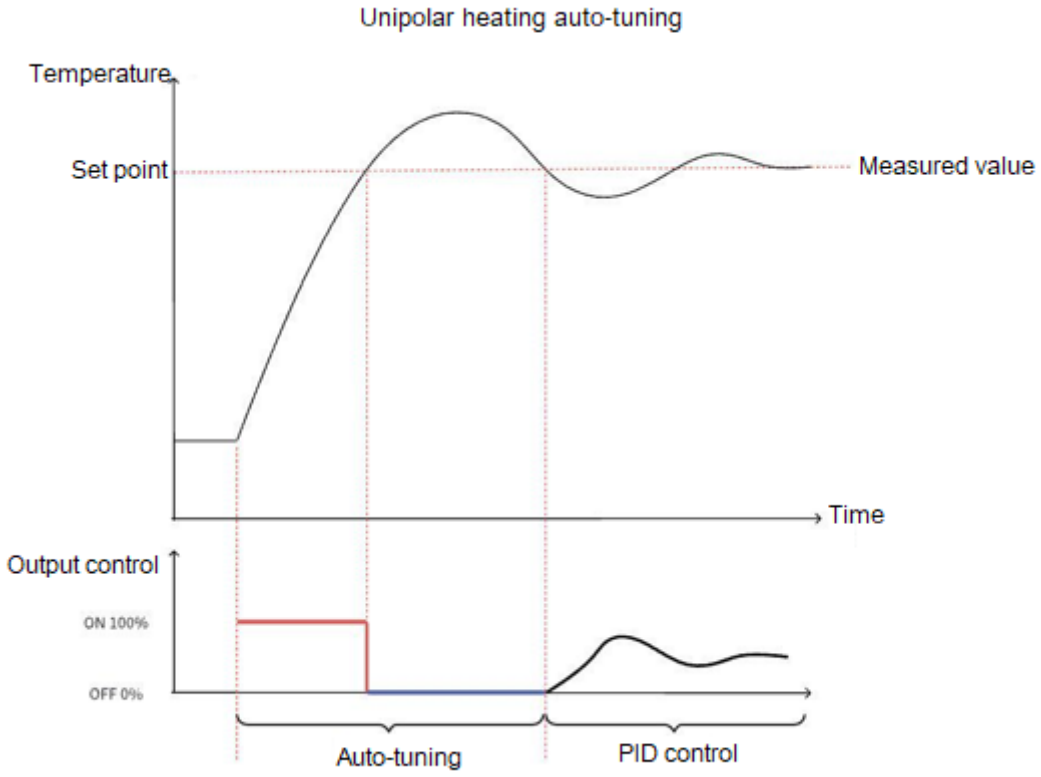
After auto-tuning starts, the value of State.1 will be set to 1. When auto-tuning is completed, the value of State.1 will be set to 0 and the value of State.2 will be set to 1.

In any of the above cases, if StartAT changes from TRUE to FALSE during auto-tuning, the auto-tuning process will be terminated, with State.1 = 0 and State.2 = 0.

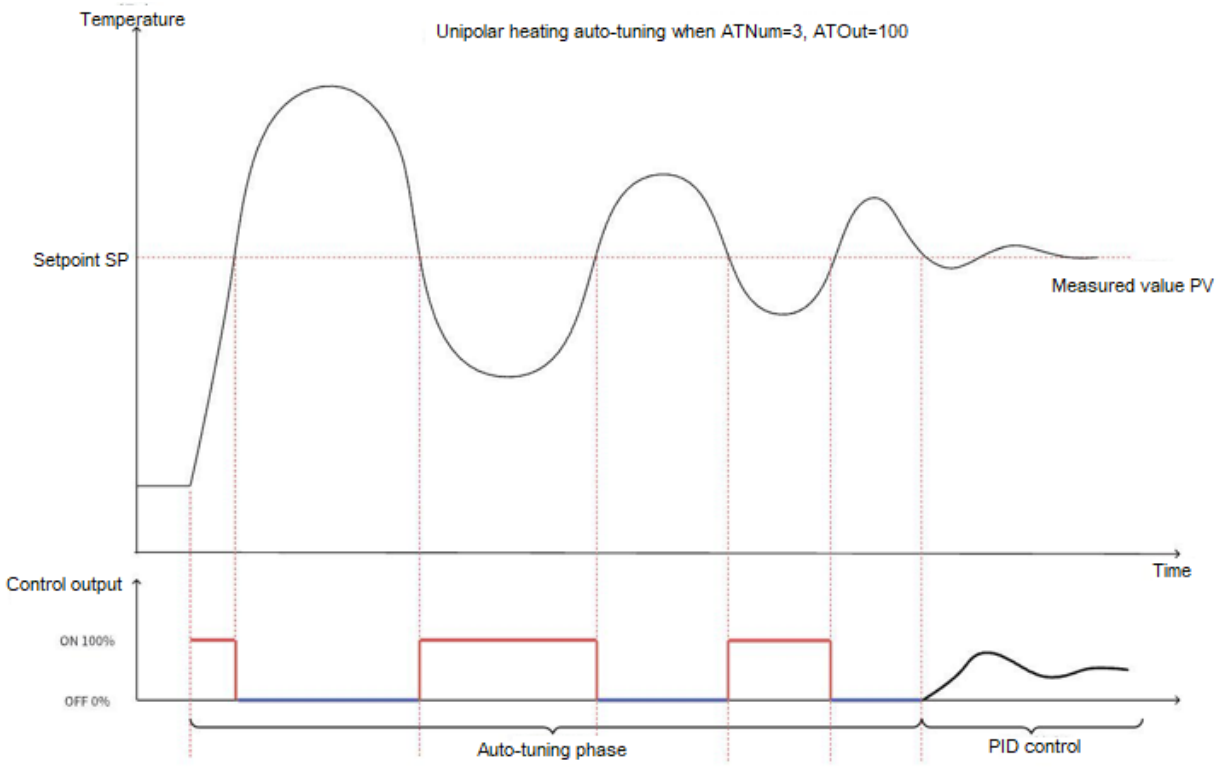
Note: After auto-tuning is completed, StartAT will automatically be set to FALSE.

3) Auto-tuning output percentage ATOut: Used to set the heating/cooling output percentage during the auto-tuning phase, supports only unipolar mode.
Number of auto-tuning waveforms ATNum: Auto-tuning is performed with multiple oscillation wavefroms based on the oscillation parameter ATNum.

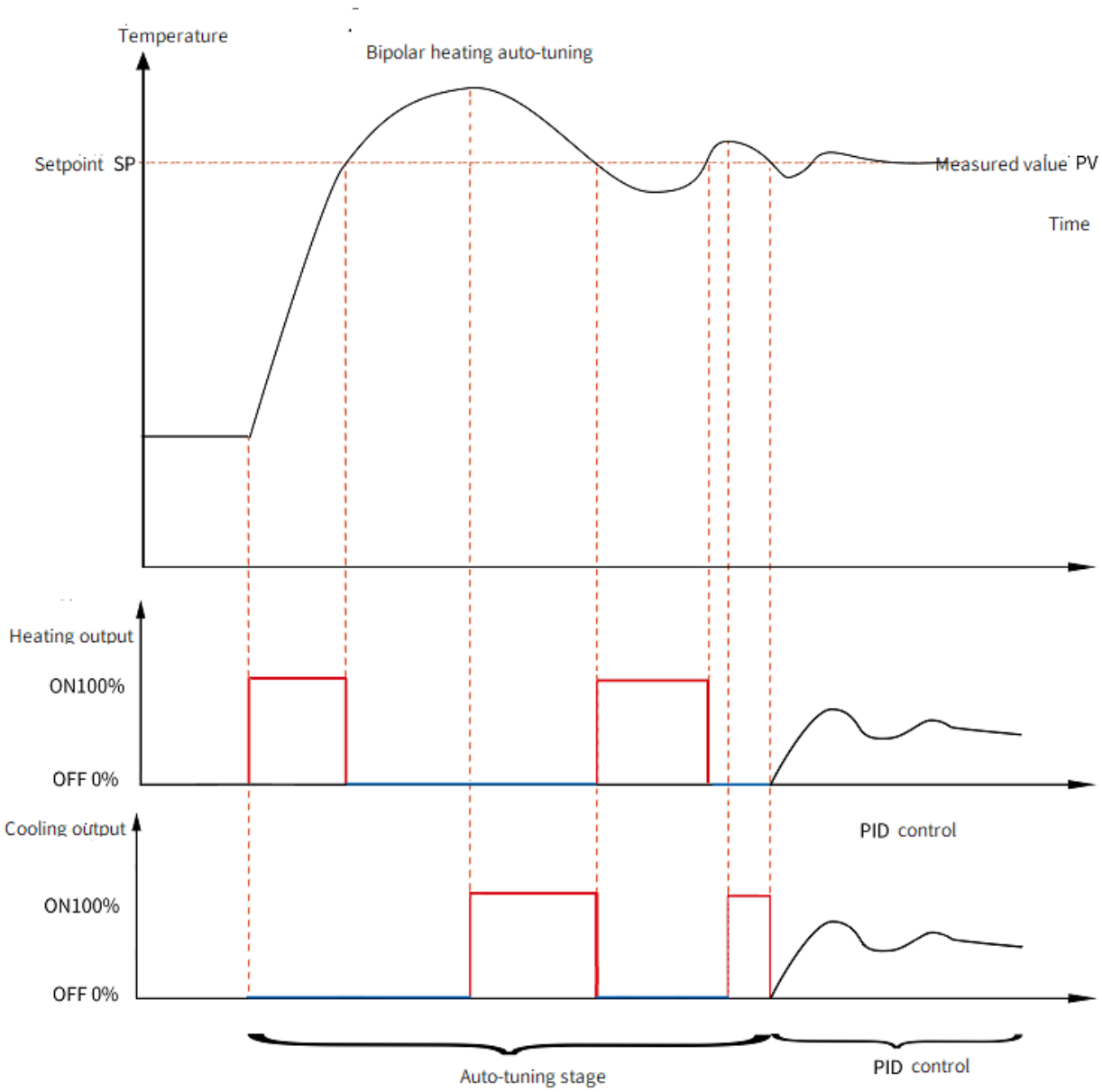
- When ATOut = 100 and ATNum = 1, the diagram of the unipolar auto-tuning process is shown below.



- When ATOut = 100 and ATNum = 3, the calculation is performed based on three oscillation parameters, and the diagram of the unipolar auto-tuning process is shown below.



- Bipolar auto-tuning will oscillate two waveforms around the setpoint. After auto-tuning is completed, StartAT will be set to FALSE;



Note: During the auto-tuning process, parameter configurations cannot be modified; otherwise, unreasonable parameters may be obtained.

NOTE

LoopControl 1.1.0.0 and later versions supports multi-waveforms auto-tuning algorithm.

- **Function**

  PID_ATC is a dedicated temperature controller for heating, cooling, and heating-cooling applications. It provides two independent outputs for heating and cooling respectively, and supports cascade configuration as well as auto-tuning.

  - **Calculation formula**

$$\mathbf{AV} = \boldsymbol{K_p} * \left[(\boldsymbol{SP} - \boldsymbol{PV}) + \frac{\mathbf{1}}{\boldsymbol{T_i} * \boldsymbol{S}} (\boldsymbol{SP} - \boldsymbol{PV}) + \frac{T_d}{\boldsymbol{S}} (\boldsymbol{SP} - \boldsymbol{PV}) \right]$$

  Discretization formula:

$$\mathbf{AV(k)} = K_p * \mathbf{E}(k) + K_p * \frac{cycle}{T_i} \sum \mathbf{E}(k) + \frac{K_p * T_D}{cycle} (\mathbf{E}(k) - \mathbf{E}(k-1))$$

◦ **Operation mode**

Operation Mode	Action
Automatic	Calculate according to the proportional-integral-derivative (PID) principle and generate the output value of the PID module. The calculation result of the PID module is limited by "output amplitude".
Manual	Directly control the output (value written by the operator or online logic). The output in this mode is limited by "output amplitude".
Interlock	In this mode, the PID stops automatic calculation, and the output value of the PID module is a preset value "InlockVal", generally referred to as the interlock setpoint. The interlock function can also be used as a "tracking" function. When certain conditions are met (interlock "InlockSW" = TRUE), the PID module automatically switches to this mode. When the interlock conditions are not met, the PID module switches back to the previous mode.

1. Interlock mode has the highest priority, followed by manual mode, and then automatic mode.
2. Automatic mode: Primarily used for setpoint control. The setpoint can be configured via the interface.
3. The selection of external or internal setpoint can be configured by the parameter PIDConfig.InputConfig.SPSel:

   SPSel = FALSE: External setpoint (the setpoint configured in the logic configuration).

   SPSel = TRUE: Internal setpoint (the preset internal setpoint).

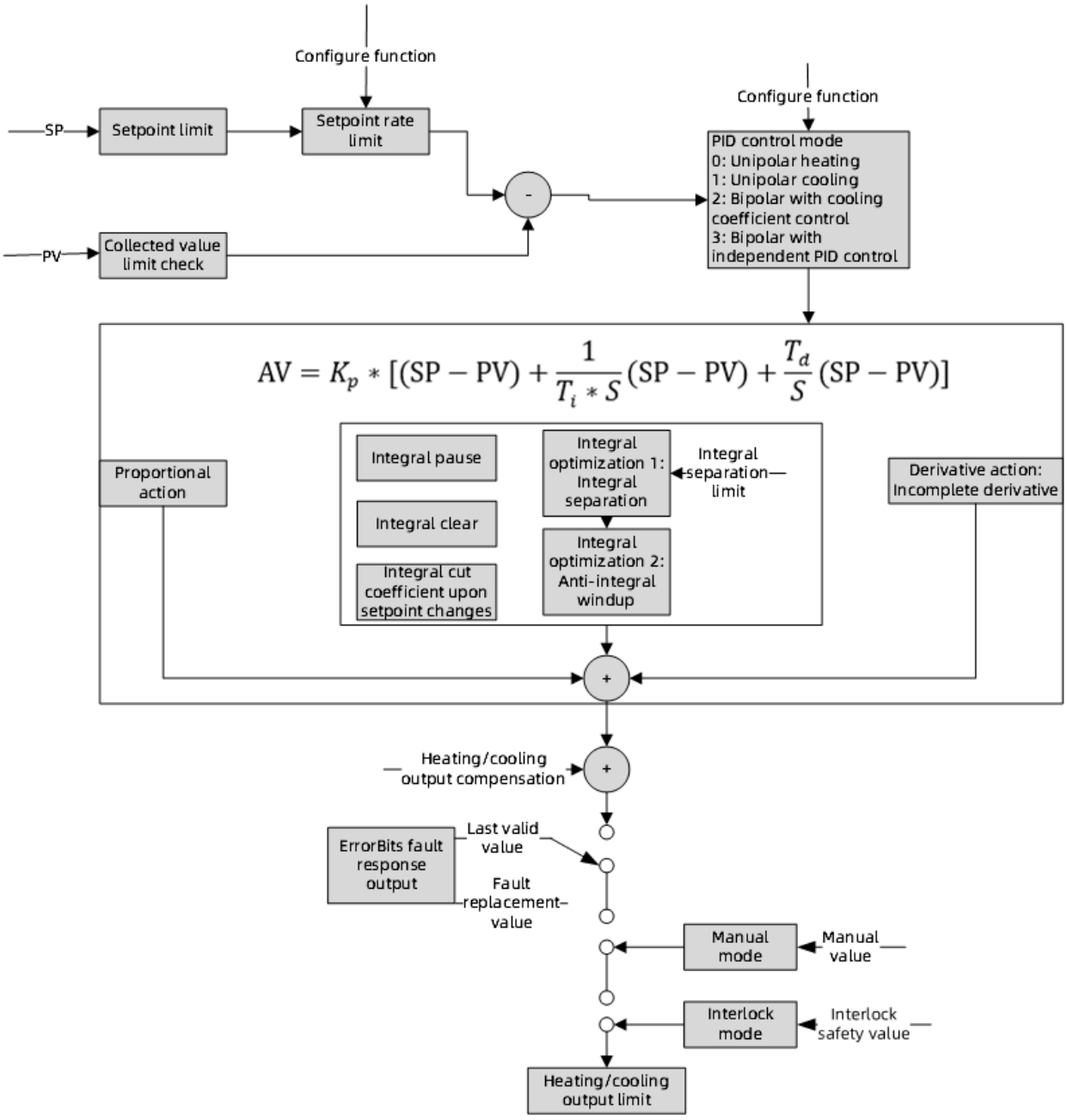
   Note: Enable "Internal setpoint tracks external setpoint" (PIDConfig.InputConfig.SPTrack = TRUE) to prevent loop fluctuations during setpoint switching. When external setpoint is selected, internal setpoint tracks it for bumpless switchover. Switching from internal to external setpoint directly adopts the real-time logic configuration value, which may cause bumpy switchover.

◦ **Bumpless switchover**

Operation Mode	Action
Interlock -> Manual	In interlock mode, the interlock value is output directly. The manual value tracks the output value to ensure bumpless switchover.

Operation Mode	Action
Interlock -> Automatic	In interlock mode, the interlock value is output directly. When switching to automatic mode, if the external setpoint is used, the logic calculation value will be adopted directly for PID calculation and output, which may cause disturbance.
Manual -> Interlock	The manual output is the manual value. Switching to interlock mode will output the interlock value directly, resulting in bumpy switchover.
Manual -> Automatic	The manual output is the operator-set value. When switching to automatic mode, if the external setpoint is used, the logic calculation value will be adopted directly for PID calculation and output, enabling bumpless switchover (disturbance is limited to the incremental part).
Automatic->Interlock	In automatic mode, the output is the normal calculation value. Switching to interlock mode will output the interlock value directly, resulting in bumpy switchover.
Automatic->Manual	In automatic mode, the output is the normal calculation value. The manual value tracks the output value to ensure bumpless switchover

- **Functional logic diagram (reverse-acting example)**



◦ **Multiple control modes**

Supports selection of 4 control modes, which is configurable based on actual requirements:

Control Mode PIDConfig.GeneralConfig.Control Mode	Control mode	Note
0	Unipolar heating	For heating-only control scenarios
1	Unipolar cooling	For cooling-only control scenarios

Control Mode PIDConfig.GeneralConfig.Control Mode	Control mode	Note
2	Bipolar with cooling coefficient control	Applicable for scenarios where heating and cooling actuators have similar time responses but different gains. The cooling output is calculated based on the heating PID parameters and the cooling coefficient CoolCoefficient.
3	Bipolar with independent PID control	It applies to cases where heating and cooling actuators differ in time response and gain. Output calculation adopts independent PID parameters for heating and cooling.

◦ **Setpoint change limit function**

In automatic mode, the change in setpoint (external and internal) is detected. If the change exceeds the predefined limit (PIDConfig.InputConfig.SPRatio), the actual setpoint used for calculation is limited to the change, and the setpoint will gradually approach the target value within the change limit range.

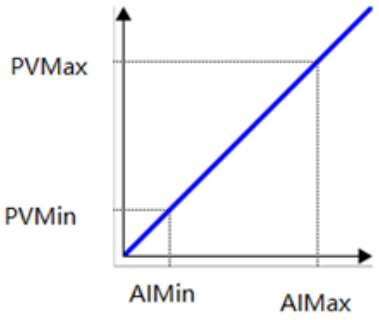
△SP > SPRatio (setpoint change): SP = LastSP + SPRatio

△SP < SPRatio (setpoint change): SP = LastSP - SPRatio

◦ **Input selection function**

Two modes of input signals can be selected.

PIDConfig.InputConfig.PVType	Input	Description
0	Current input "PV"	-

PIDConfig.InputConfig.PVType	Input	Description
1	Analog input "AI"	Calibrate via linear conversion based on analog input limits and PV limits.  Analog upper limit - PIDConfig.InputConfig.AIMax Analog lower limit - PIDConfig.InputConfig.AIMin PV upper limit - PIDConfig.InputConfig.AIMax PV lower limit - PIDConfig.InputConfig.AIMin

- **Integral function**

  Integral separation

  Integral windup is mainly caused by the integral term. To overcome this issue, it is crucial to limit integral action and prevent excessive integral accumulation. The integral separation function can eliminate integral windup and enable the function block to resist integral windup.

  For non-temperature-controlled objects, the configuration instructions for integral separation are as follows:

Integral separation enable PIDConfig.GeneralConfig.IntegralDivision	Integral separation limit PIDConfig.GeneralConfig.IntegralDivLimit
TRUE	|Deviation| >= Integral Separation Limit: Integral component calculation is disabled (control law operates as PD control).
	|Deviation| < Integral Separation Limit: Full integral calculation is enabled (control law operates as PID control).
FALSE	Full integral calculation is always enabled.

  Integral pause

  When integral suspend is triggered "PIDConfig.GeneralConfig.IntegralSuspend = TRUE", the integral action remains unchanged and stops accumulating.

Integral clearance

When integral clear is triggered "PIDConfig.GeneralConfig.IntegralClear = TRUE", integral action will be cleared to 0.

Integral cut coefficient

When the setpoint changes, the initial integral action can be reduced by setting the integral cut coefficient "PIDConfig.SuperConfig.IntegralCut". The setting range is [0,1], default is 0.75.

- **Incomplete derivative function**

  Incomplete derivative is equivalent to adding a first-order inertial element to the derivative action, which moderates the derivative effect and effectively improves the system's control performance.

Configuration parameter: PIDConfig.SuperConfig.KdGain	Derivative action
0	Derivative action is only active for the current cycle (default value).
(0,1)	The larger the value, the smoother the derivative action.
1	The derivative action is disabled.

- **Deadzone control**

  Supports width configuration of control dead zone via "PIDConfig.SuperConfig.DeadBand". Within the dead zone, the system is deemed stable with unchanged PID output; beyond it, PID operates normally.

- **Delay start/delay shutdown function**

  For multi-channel device control, set different delayed startup/shutdown times to stagger on/off sequences. Prevent large current fluctuations caused by simultaneous multi-channel startup.

Delay start/shutdown time	Description
PIDConfig.GeneralConfig.DelayStartUp	After delaying the DelayStartUp time, the PID calculation starts.
PIDConfig.GeneralConfig.DelayShutdown	After delaying the DelayShutdown time, the PID calculation stops.

- **Output compensation function**

  The output compensation function directly adds the output compensation value to the result of the PID automatic calculation to form the output value. The output compensation function is active in automatic mode and inactive in manual mode.

  Heating output = Heating PID calculation output + Heating output compensation (PIDConfig.OutputConfigHeat.OutputOffset)

  Cooling output = Cooling PID calculation output + Cooling output compensation (PIDConfig.OutputConfigCool.OutputOffset)

- **Output amplitude limit function**

When the heating output value > Heating output upper limit "PIDConfig.OutputConfigHeat.OutMax", th e heating output value = Heating output upper limit.

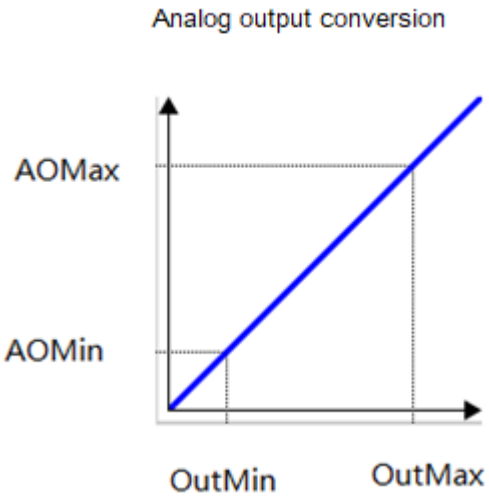
When the heating output value < Heating output lower limit "PIDConfig.OutputConfigHeat.OutMin", the heating output value = Heating output lower limit.

When the cooling output value > the cooling output upper limit "PIDConfig.OutputConfigHeat.OutMax" , the cooling output value = the cooling output upper limit.

When the cooling output value < the cooling output lower limit "PIDConfig.OutputConfigHeat.OutMin", the cooling output value = the cooling output lower limit.

- **Output selection function**

  Supports 3 modes of output control:

Heating output type - PIDConfig.OutputConfigHeat.OutType Cooling output type - PIDConfig.OutputConfigCool.OutType	Output	Description
0	PWM output	Convert to switching output based on the PWM output cycle.
1	Analog output	Calibrate and output via linear conversion based on the analog output limits and PID output limits.  Analog output upper limit - PIDConfig.OutputConfig.AOMax Analog output lower limit - PIDConfig.OutputConfig.AOMin PID output upper limit - PIDConfig.OutputConfig.OutMax PID output lower limit - PIDConfig.OutputConfig.OutMin

Heating output type - PIDConfig.OutputConfigHeat.OutType Cooling output type - PIDConfig.OutputConfigCool.OutType	Output	Description
2	Percentage output	Output within 0~100%

◦ **Fault detection**

If any bit of ErrorBits is triggered, Error is set to TRUE.

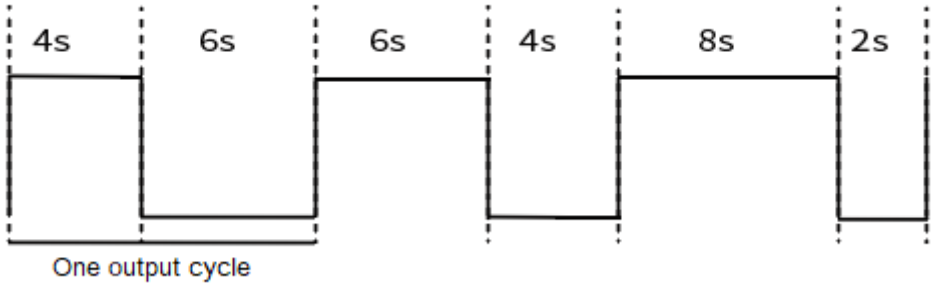
Bitwise description of ErrorBits		Error output configuration parameters PIDConfig.InputConfig.ErrorAction
Bit0 = 1	PV overlimit alarm	FALSE: Maintain previous output TRUE: Output the substitute value "PIDConfig.InputConfig.ErrorOut"
Bit1 = 1	Parameter configuration error alarm	

See ErrorID for detailed error information.

◦ **PWM output**

For PWM, the output is ON or OFF. The analog of PID output needs to be converted to digital output (ON/OFF) based on the duty cycle. The duty cycle is the percentage of the high-level pulse time to a whole pulse cycle.

For example: If the PID regulator output is 40% and the PWM output cycle "PIDConfig.GeneralConfig.CycleTime" is set to 10s, the ON duration is 10s × 40% = 4s, and the OFF duration is 10s – 4s = 6s. The duty cycle is 40%. The signal stays ON for 4s and OFF for 6s per 10s cycle before repeating the calculation for the next cycle.

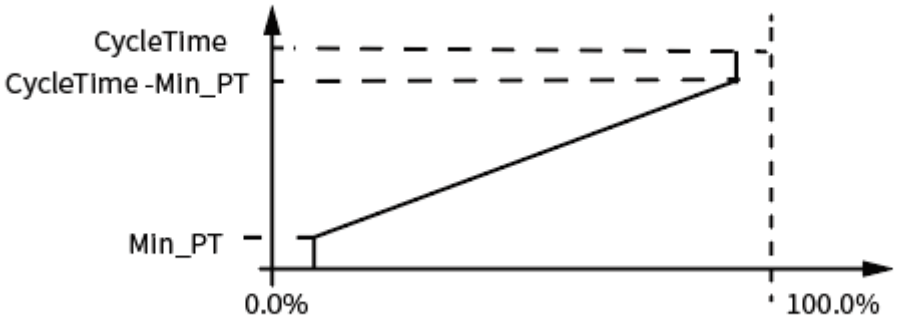


▪ Minimum On/Off pulse time

To avoid frequent operation of the actuators, the minimum heating on/off pulse time PIDConfig.OutputConfigHeat.Min_PT and minimum cooling on/off pulse time PIDConfig.OutputConfigCool.Min_PT can be set separately.

If the calculated high-level pulse time < Min_PT, the output duty cycle is 0%.

If the calculated high-level pulse time > PWM output cycle – Min_PT, the output duty cycle is 100%.



- **Cascade function**

  Cascade control involves two or more cascaded PID controllers. The control variable of the master controller is interconnected with the setpoint of the slave controller, nesting the two control circuits. Cascade control has advantages that the slave circuit responds faster than the main circuit and can pre-compensate for interference in the inner circuit.
  Note: Cascade control is only possible when there are other measurable variables in the process besides the main control variable, and the internal control circuit is much faster than the external circuit.

  Bumpless switchover: If the slave circuit is not operating in cascade mode (automatic mode with external setpoint), the master circuit switches to tracking mode. The output value of the master circuit tracks the process value of the slave circuit to ensure a smooth transition to cascade mode.

  Priority: Error response > Interlock > Manual > Master tracking mode (triggered by slave) > Automatic.

- **Description**

1. First, select the controlled physical quantity GeneralConfig.PhysicalQuantity, which includes General and Temperature options.
   0: General, primarily for pressure, flow, and other control scenarios. For general control, integral separation can be configured via the IntegralDivision and IntegralDivLimit in Advanced Settings.

   1: Temperature, optimized for various temperature controls.

2. Enable the auto-tuning function at room temperature. Once the auto-tuning ends, xStartAT automatically sets to FALSE, and the tuned parameters Kp, Ti, Td are generated. Continue program operation to check if the controlled quantity reaches a steady state.

3. If the usage needs are met after the controlled quantity reaches steady state, record the values of Kp, Ti, and Td, and write these values into the program as defaults. The Kp, Ti, and Td values can also be manually adjusted to meet the usage requirements.

4. The larger the auto-tuning coefficient rATCoefficient, the faster the set temperature can be reached, but it may cause greater overshoot. The smaller the auto-tuning coefficient, the slower the set temperature can be reached. However, the temperature curve will be very smooth, without overshoot or with very small overshoot. If there is no requirement to use the default value of 0.5 and no customer needs to reach the set temperature faster, increase rATCoefficient and re-run auto-tuning. If the customer needs a smooth curve and small overshoot, decrease rATCoefficient and re-run auto-tuning.

5. Interlock function: When the interlock switch InlockSW is triggered, the interlock safety value InlockVal is output.

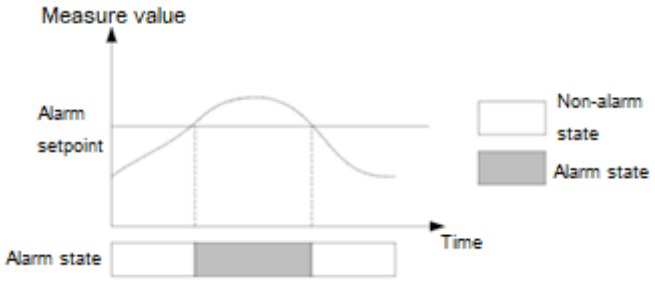
6. Output priority: Error output > Interlock output > Manual output > Automatic output.

Note:

1. Always verify parameter correctness before running the PID.
2. Auto-tuning is recommended to start at room temperature.
3. For common operating temperatures of 150 to 200 °C, tuning can be performed at 175 °C. Select the customer's typical operating temperature for tuning to ensure parameter accuracy.
4. Perform auto-tuning first to obtain initial PID parameters, then adjust them appropriately based on control performance.

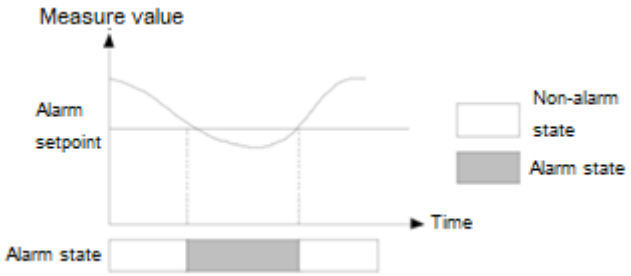
- **Upper and lower limit alarms**
  - Upper limit alarm of the input PV value

    An alarm is triggered when the measured PV value (or PV value after analog conversion) exceeds the upper limit alarm value.



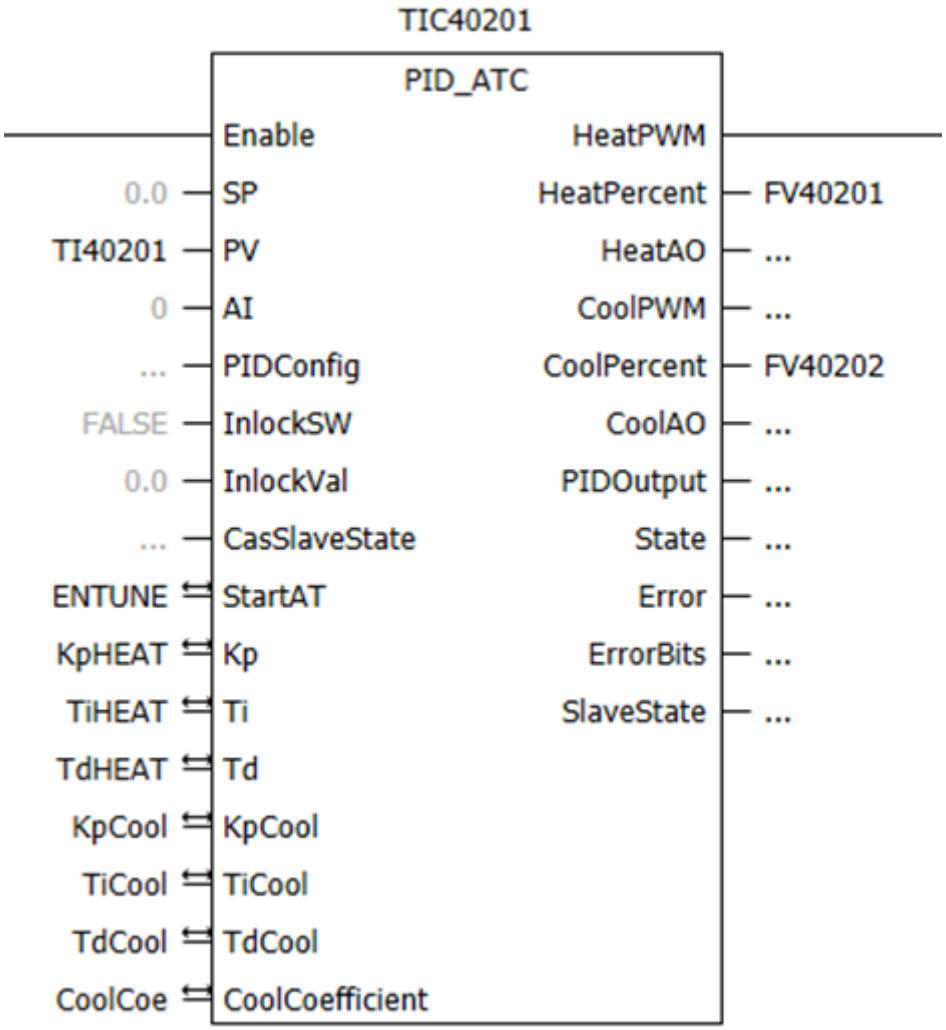
  - Lower limit alarm of the input PV value

    An alarm is triggered when the measured PV value (or PV value after analog conversion) falls below the lower limit alarm value.



- **Application example**

  Control scheme configuration



Example

No.	Instance	Type
1	TIC40201	Temperature controller
2	TI40201	Temperature TC detection signal
3	FV40201	Hot water flow control valve
4	FV40202	Cold water flow control valve

Cascade control scheme configuration

Example

No.	Instance	Type
1	TIC40201	Temperature controller
2	TI40201	Temperature TC detection signal
3	FIC40201	Flow controller
4	LV40201	Liquid level control valve

No.	Instance	Type
5	FI40201	Flow AI detection signal

- **Fault codes (ErrorID)**

  The fault codes (ErrorID) are the same as PID_AT. For details, see Fault codes (ErrorID).

- **Error handling**

  Action on error alarm for PIDConfig.InputConfig.ErrorAction

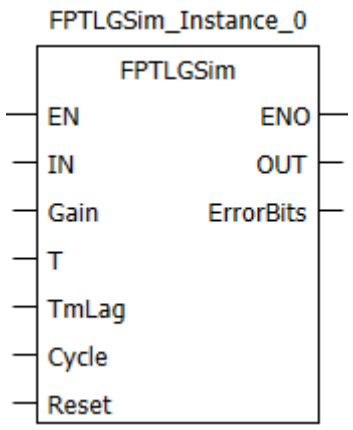
1. When PIDConfig.InputConfig.ErrorAction = FALSE, maintain previous output during error alarm, and recalculate current output upon error recovery.
2. When PIDConfig.InputConfig.ErrorAction = TRUE, output ErrorOut substitute value on error alarm, and recalculate current output upon error recovery.

18.3.1.7 Help Function

18.3.1.7.1 FPTLGSim

This is the instruction for simulation of first-order lag plus pure dead time.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FPTLGSim	Simulation of first-order lag plus pure dead time	FB		FPTLGSim_Instance_0(IN := , Gain := , T := , TmLag := , Cycle := , Reset := , OUT => , ErrorBits =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input	Follows data type	-	Input

Input/Output	Variable	Name	Valid Range	Default Value	Description
	Gain	Static gain	Follows data type	1	This parameter must be within the range [-1000000, 1000000].
	T	Time constant	Follows data type	1	The maximum value of this parameter is 10000.
	TmLag	Pure delay time constant	Follows data type	20	Unit: calculation cycle. Range: 0 to 36000.
	Cycle	Calculation cycle	Follows data type	0.5	The logic calculation cycle must be set to a non-negative number.
	Reset	Reset flag	Follows data type	FALSE	If the reset flag is set to TRUE, the OUT is reset to 0.
Output	OUT	Output	Follows data type	-	Output.
	ErrorBits	Error report	Follows data type	-	0: No error. 4: Parameter setting error.

Data type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Gain	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
T	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
TmLag	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Cycle	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Reset	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Error Bits	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction can be used to simulate a plant/device object with first-order lag plus pure dead time.

  For systems with continuously varying parameters in industrial process control, such as temperature, pressure, and flow, the mathematical model can be simplified as a first-order lag plus pure dead time.

  Mathematical model:

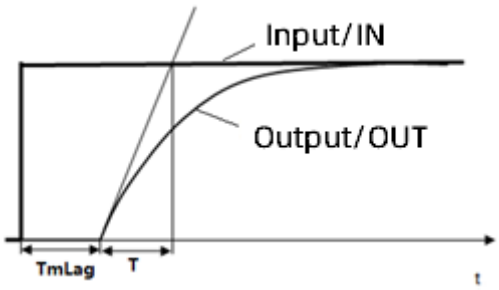
$$G(s) = \frac{\mathrm{k}}{\mathrm{Ts}+1}\mathrm{e}^{-\tau s}$$

  K is the static gain of the object; T is the time constant of the object; τ is the pure dead time constant of the object.

  Difference equation:

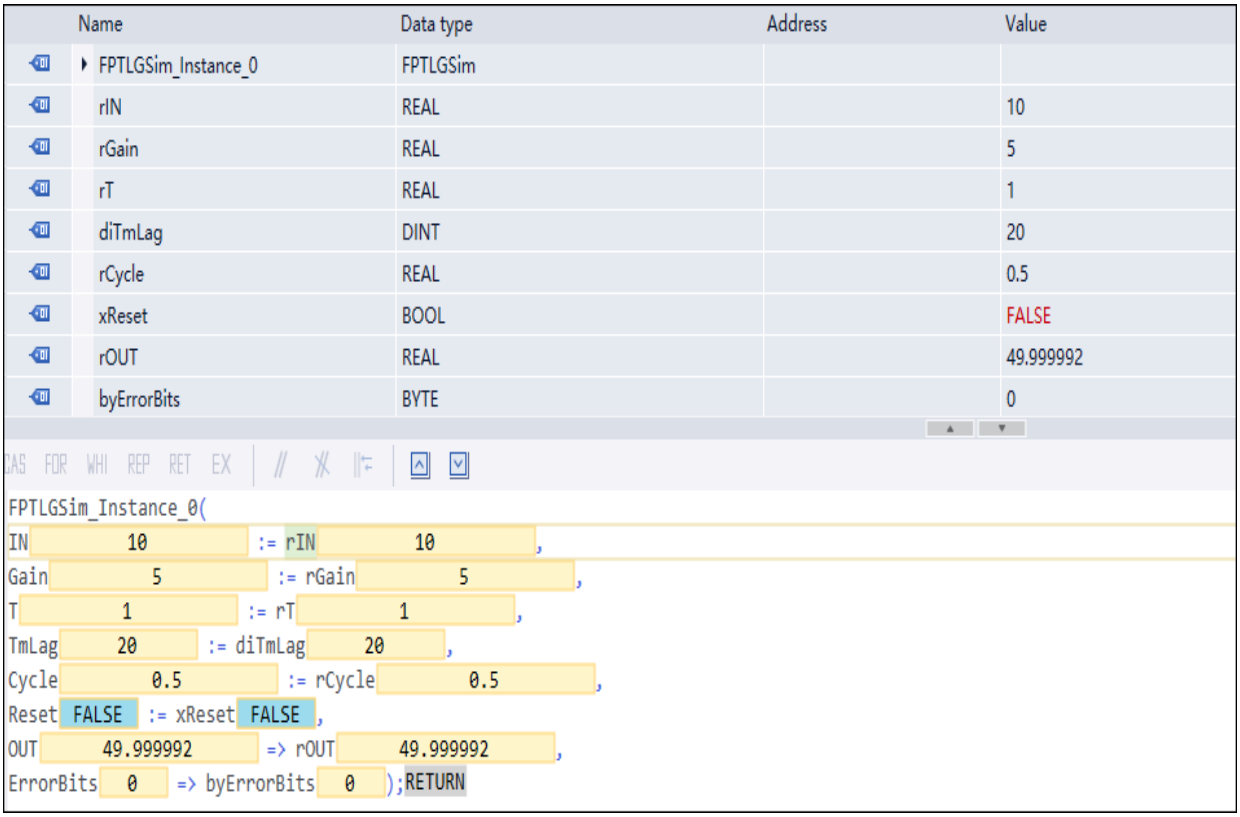
$$OUT(n) = \mathrm{e}^{-\frac{Cycle}{\mathrm{T}}} * OUT(n-1) + Gain * \left(1 - \mathrm{e}^{-\frac{Cycle}{\mathrm{T}}}\right) * IN(n - \mathrm{TmLag})$$

  The dynamic response diagram is shown in the figure below:



- **Program example**

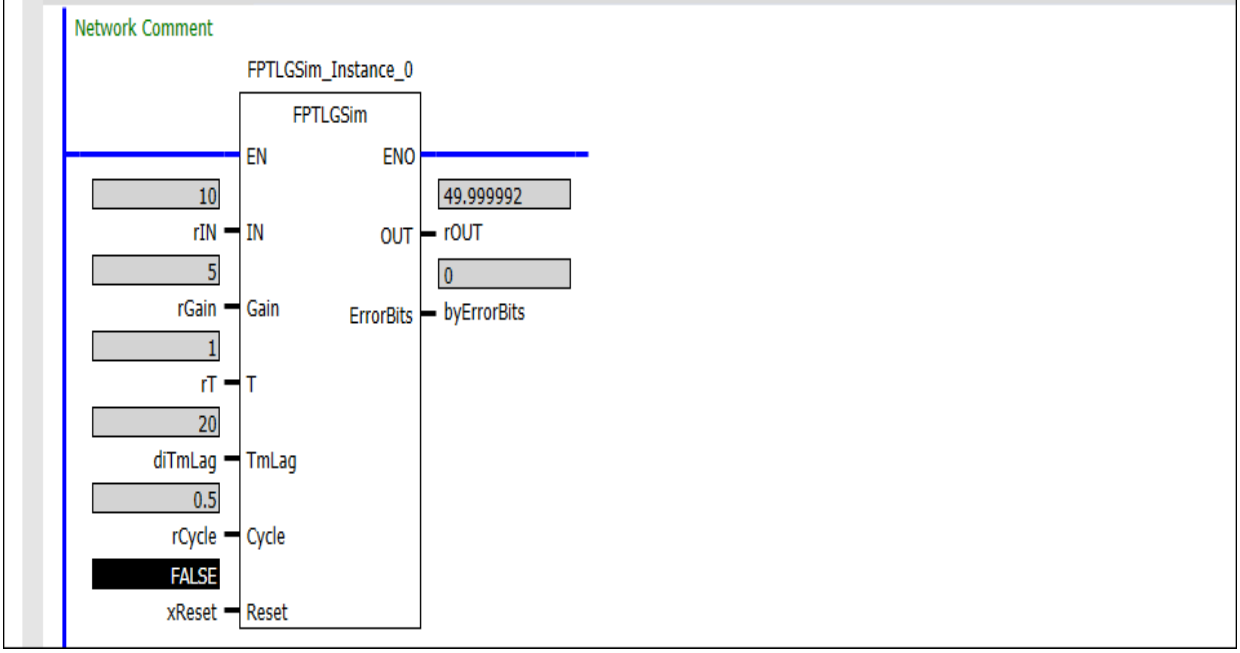
  ST:



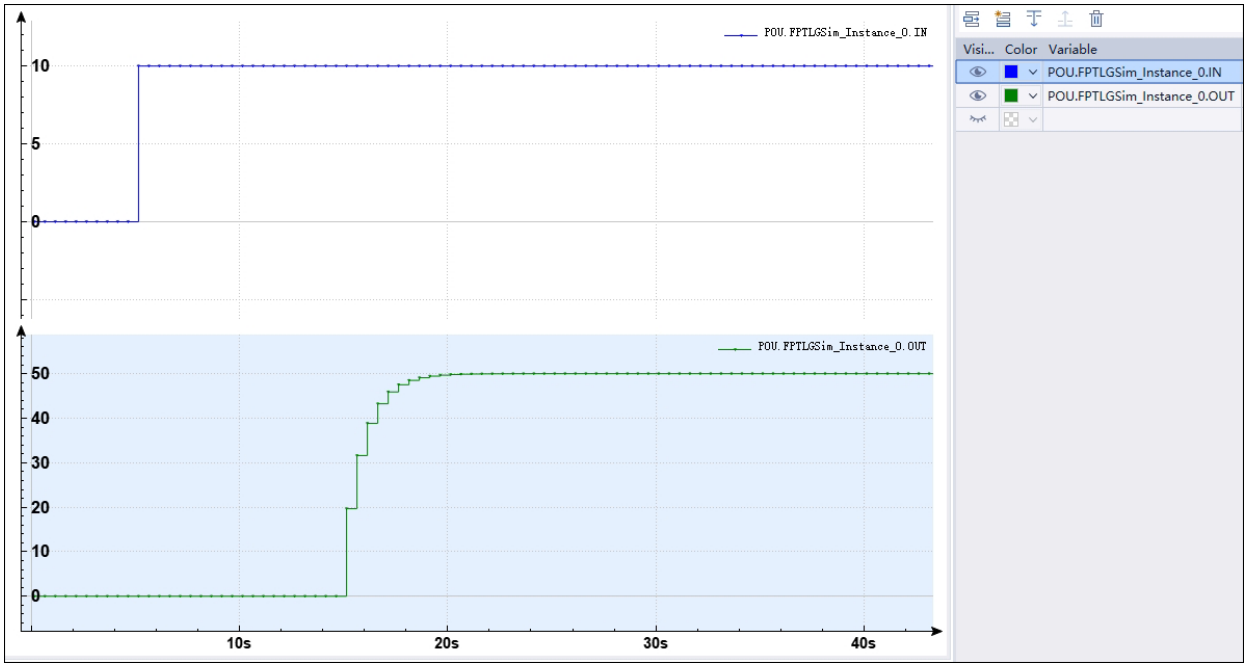
Name	Data type	Address	Value
FPTLGSim_Instance_0	FPTLGSim		
rIN	REAL		10
rGain	REAL		5
rT	REAL		1
diTmLag	DINT		20
rCycle	REAL		0.5
xReset	BOOL		FALSE
rOUT	REAL		49.999992
byErrorBits	BYTE		0

LD:

	Name	Data type	Address	Value
	FPTLGSim_Instance_0	FPTLGSim		
	rIN	REAL		10
	rGain	REAL		5
	rT	REAL		1
	diTmLag	DINT		20
	rCycle	REAL		0.5
	xReset	BOOL		FALSE
	rOUT	REAL		49.999992
	byErrorBits	BYTE		0



Operation result:



- **Precautions**
  - A larger pure dead time constant TmLag indicates a later curve response.
  - The actual dead time is the product of TmLag and cycle period.
  - Set the Cycle parameter to be consistent with the POU scheduling cycle when using the instruction.
  - A larger time constant T (the time for the output to reach 60% of the input) indicates a smoother curve response.
  - If the time constant T is negative, the result of the inverse Laplace transform is an exponentially increasing curve, which diverges over time. However, a divergent first-order system does not conform to practical application scenarios.
  - Affected by the cycle period, the OUT value may not accurately reach the theoretical value. The calculation result can be considered correct if it falls within the interval of one program cycle before or after.
  - Model verification can be performed using a step input or by comparison with MATLAB. Under a unit step input signal, a first-order inertial system (non-dead-time model) has two important performance indicators:
    - Time constant Ta (the time for the system output to reach approximately 63% of its final value $(1-\frac{1}{e})*100\%$): $T_a=\frac{1}{a}$.
    - Settling time Ts (the time required for the system output to stay within a 2% tolerance of the final value): $T_s=\frac{4}{a}$.
    - Considering the dead time of the system, the actual time constant and settling time of the output response are the sum of the dead time and the theoretical calculated time. For example, if the pure dead time constant is 5, the Cycle is set to 1 s, and the theoretical calculated T is 10 s, the actual time is 10 + 5 × 1 = 15 s.
  - Due to model accuracy limitations, the accuracy of OUT is 1.0E-7.
  - If system input parameters (other than the input IN) need to be modified, go offline, reconfigure, download, and then perform the simulation operation.
  - This instruction requires a significant amount of calculation. When using it, pay attention to designing the system task scheduling cycle appropriately. When calling a single instruction block, the minimum recommended task scheduling cycle is 5 ms. When multiple instructions of this type are called in the same task, the corresponding task scheduling cycle must be increased proportionally. Otherwise, PLC may malfunction.
  - It should be noted that when the value of ErrorBits is 4, it indicates that the system parameters are incorrectly set, and the output value will remain unchanged. In this case, one or more of the following conditions may exist. You must check the parameter settings before performing configuration simulation.

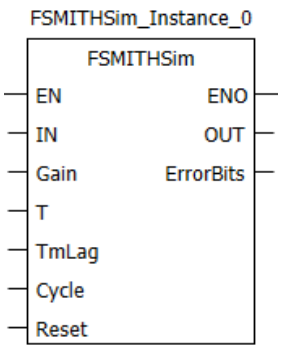
Error Report Result	Cause	Modification Suggestion
ErrorBits = 4	Input value IN overlimit	Modify the value of IN so that it is within the range of the REAL data type.

Error Report Result	Cause	Modification Suggestion
	Gain value GAIN overlimit	Modify the value of GAIN so that it is within the range [-1000000,1000000].
	Lag time Tmlag overlimit	Modify the dead time TmLag so that it is within the range of [0, 36000].
	Time constant T overlimit	Modify the time constant T so that it is within the range (-∞,10000].
	Cycle value Cycle overlimit	Modify the time constant T so that it is within the range (0,∞).

18.3.1.7.2 FSMITHSim

This is the instruction for first-order lag plus pure dead time compensation simulation.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
FSMITHSim	First-order lag plus pure dead time compensation simulation	FB		FSMITHSim_Instance_0(IN := , Gain := , T := , TmLag := , Cycle := , Reset := , OUT => , ErrorBits =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input	Follows data type	-	Input

Input/Output	Variable	Name	Valid Range	Default Value	Description
	Gain	Static gain	Follows data type	1	This parameter must be within the range [-1000000, 1000000].
	T	Time constant	Follows data type	1	The maximum value of this parameter is 10000.
	TmLag	Pure delay time constant	Follows data type	20	Unit: calculation cycle. Range: 0 to 36000.
	Cycle	Calculation cycle	Follows data type	0.5	The logic calculation cycle must be set to a non-negative number.
	Reset	Reset flag	Follows data type	FALSE	If the reset flag is set to TRUE, the OUT is reset to 0.
Output	OUT	Output	Follows data type	-	Output.
	ErrorBits	Error report	Follows data type	-	0: No error. 4: Parameter setting error.

Data type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Gain	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
T	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
TmLag	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Cycle	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Reset	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Error Bits	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  Smith predictive control for first-order lag plus pure dead time objects, is used to compensate for the pure dead time in the control circuit and improve control performance.

  Smith predictive control is a widely used compensation method for dead time objects. It pre-estimates the dynamic characteristics of the system under a given signal, and then compensate through the predictor, allowing the delayed control variable to be reflected to the controller earlier, causing the controller to act in advance, thereby reducing overshoot and accelerating the regulation process. In practical applications, a Smith compensation block is typically connected in parallel with the PID controller.

  The transfer function of the compensation for the first-order lag plus pure dead time model is:

$$G(s) = \frac{\mathrm{k}}{\mathrm{Ts}+1}(1-\mathrm{e}^{-\tau \mathrm{s}})$$

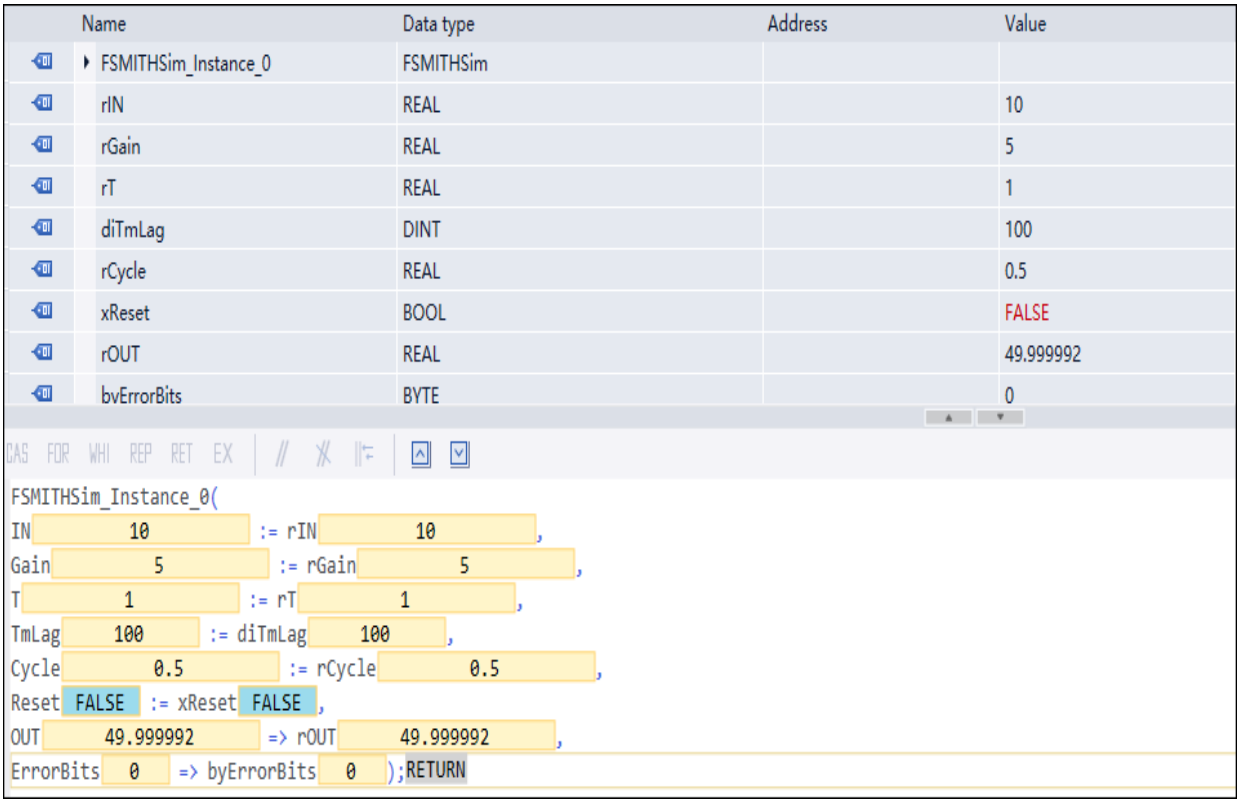
  K is the static gain of the object; T is the time constant of the object; τ is the pure dead time constant of the object.

  Converted difference equation:

$$OUT(n) = \mathrm{e}^{-\frac{Cycle}{\mathrm{T}}} * OUT(n-1) + Gain * \left(1-\mathrm{e}^{-\frac{Cycle}{\mathrm{T}}}\right) * (IN(n) - IN(n-\mathrm{TmLag}))$$

- **Program example**

  ST:

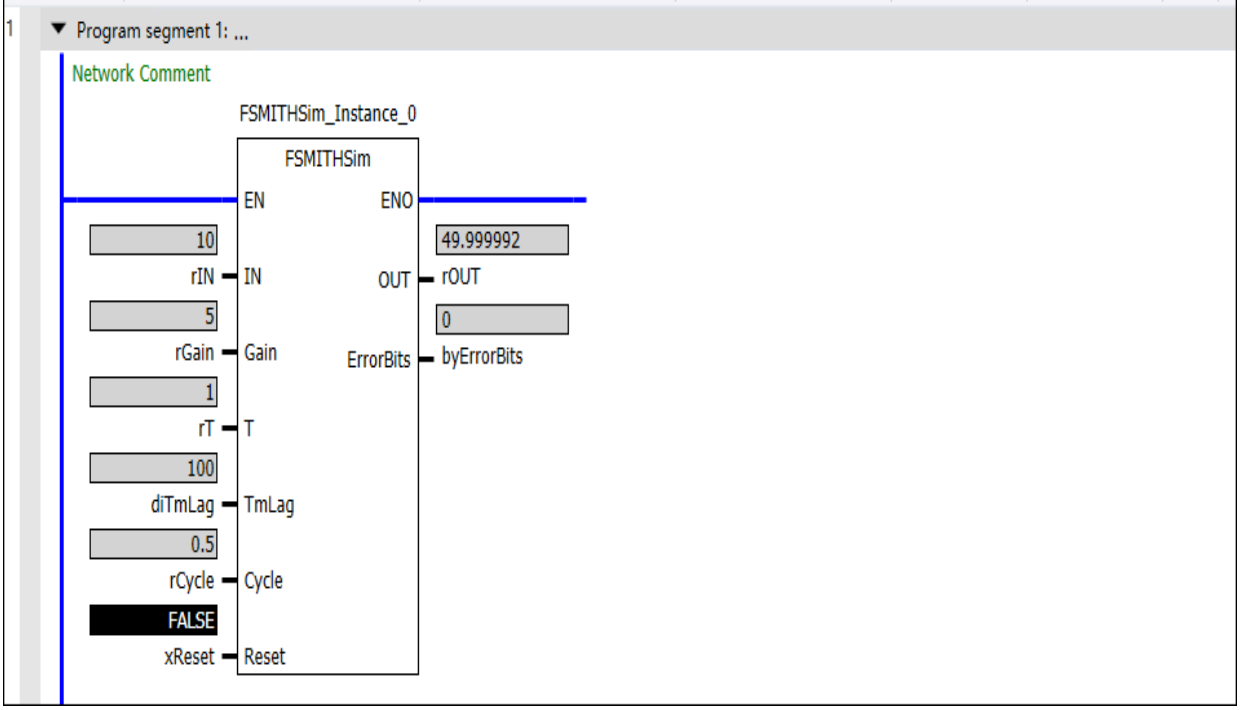


Name	Data type	Address	Value
FSMITHSim_Instance_0	FSMITHSim		
rIN	REAL		10
rGain	REAL		5
rT	REAL		1
diTmLag	DINT		100
rCycle	REAL		0.5
xReset	BOOL		FALSE
rOUT	REAL		49.999992
byErrorBits	BYTE		0

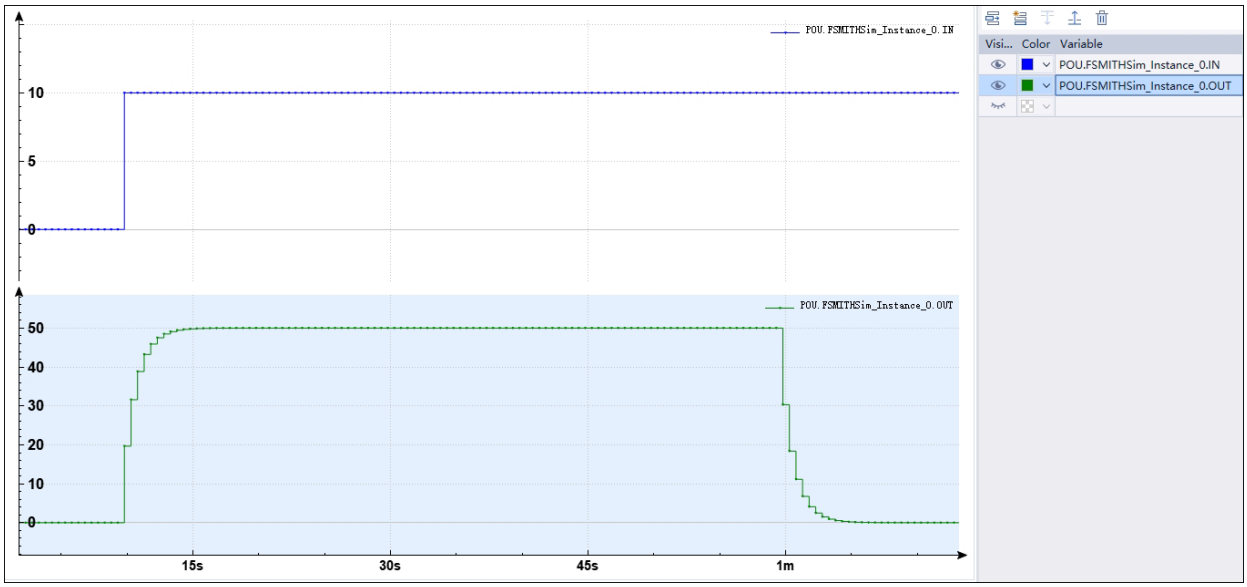
```
CAS FOR WHI REP RET EX
FSMITHSim_Instance_0(
IN 10 := rIN 10,
Gain 5 := rGain 5,
T 1 := rT 1,
TmLag 100 := diTmLag 100,
Cycle 0.5 := rCycle 0.5,
Reset FALSE := xReset FALSE,
OUT 49.999992 => rOUT 49.999992,
ErrorBits 0 => byErrorBits 0 );RETURN
```

LD:

Name	Data type	Address	Value
FSMITHSim_Instance_0	FSMITHSim		
rIN	REAL		10
rGain	REAL		5
rT	REAL		1
diTmLag	DINT		100
rCycle	REAL		0.5
xReset	BOOL		FALSE
rOUT	REAL		49.999992
byErrorBits	BYTE		0



Operation result:



- **Precautions**
  - Set the Cycle parameter to be consistent with the POU scheduling cycle when using the instruction.
  - If the time constant T is negative, the original inertial system will diverge. However, a divergent first-order system does not conform to practical application scenarios.

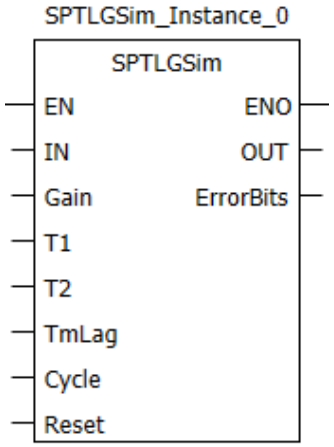
- Affected by the cycle period, the OUT value may not accurately reach the theoretical value. The calculation result can be considered correct if it falls within the interval of one program cycle before or after.
- Model verification can be performed using a step input or by comparison with MATLAB. After completing the verification of the first-order lag plus dead time model, it can also be connected in parallel with the model to sum the outputs, thereby verifying the system under dead-time-free conditions. In this case, the system output will follow the theoretical value of the first-order system time-domain response.
  - Time constant Ta (the time for the system output to reach approximately 63% of its final value $(1-\frac{1}{e})*100\%$): $T_a=\frac{1}{a}$.
  - Settling time Ts (the time required for the system output to stay within a 2% tolerance of the final value): $T_s=\frac{4}{a}$.
- Due to model accuracy limitations, the accuracy of OUT is $1.0e^{-7}$.
- If system input parameters (other than the input IN) need to be modified, go offline, reconfigure, download, and then perform the simulation operation.
- This instruction requires a significant amount of calculation. When using it, pay attention to designing the system task scheduling cycle appropriately. When calling a single instruction block, the minimum recommended task scheduling cycle is 5 ms. When multiple instructions of this type are called in the same task, the corresponding task scheduling cycle must be increased proportionally. Otherwise, PLC may malfunction.
- It should be noted that when the value of ErrorBits is 4, it indicates that the system parameters are incorrectly set, and the output value will remain unchanged. In this case, one or more of the following conditions may exist. You must check the parameter settings before performing configuration simulation.

Error Report Result	Cause	Modification Suggestion
ErrorBits=4	Input value IN overlimit	Modify the value of IN so that it is within the range of the REAL data type.
	Gain value GAIN overlimit	Modify the value of GAIN so that it is within the range [-1000000,1000000].
	Lag time Tmlag overlimit	Modify the dead time TmLag so that it is within the range of [0, 36000].
	Time constant T overlimit	Modify the time constant T so that it is within the range (-∞,10000].
	Cycle value Cycle overlimit	Modify the time constant T so that it is within the range (0,∞).

18.3.1.7.3 SPTLGSim

This is the instruction for simulation of second-order lag plus pure dead time.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SPTLGSim	Simulation of second-order lag plus pure dead time	FB		SPTLGSim_Instance_0(IN := , Gain := , T1 := , T2 := , TmLag := , Cycle := , Reset := , OUT => , ErrorBits =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input	Follows data type	-	Input
	Gain	Static gain	Follows data type	1	This parameter must be within the range [-1000000, 1000000].
	T1	First-order time constant	Follows data type	1	The maximum value of this parameter is 10000.
	T2	Second-order time constant	Follows data type	1	The maximum value of this parameter is 10000.
	TmLag	Pure delay time constant	Follows data type	20	Unit: calculation cycle. Range: 0 to 36000.

Input/Output	Variable	Name	Valid Range	Default Value	Description
	Cycle	Calculation cycle	Follows data type	0.5	The logic calculation cycle must be set to a non-negative number.
	Reset	Reset flag	Follows data type	FALSE	If the reset flag is set to TRUE, the OUT is reset to 0.
Output	OUT	Output	Follows data type	-	Output.
	ErrorBits	Error report	Follows data type	-	0: No fault. 1: Calculation result overlimit. 4: Parameter setting error.

Data type

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Gain	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
T1	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
T2	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
TmLag	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Cycle	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Reset	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Error Bits	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction is used to simulate an object that exhibits second-order lag pure dead time.

  Transfer function:

$$G(s) = \frac{\mathrm{k}}{\mathrm{T2} * s^2 + \mathrm{T1} * \mathrm{s} + 1} \mathrm{e}^{-\tau \mathrm{s}}$$

  The model can be used with a step input. Under a unit step input signal, the performance indices of a second-order underdamped system are as follows:

  According to the standard form of a second-order system:

$$\Phi(\mathrm{s}) = \frac{\omega_{\mathrm{n}}^2}{\mathrm{s}^2 + 2\zeta\omega_{\mathrm{n}}\mathrm{s} + \omega_{\mathrm{n}}^2}$$

  By combining the transfer function of a second-order system, the damping ratio and natural frequency can be determined by the following equations:

$$\delta = \frac{T_{1} * \sqrt{T_2}}{2 * T_2}, \quad W_n = \frac{1}{\sqrt{T_2}}$$

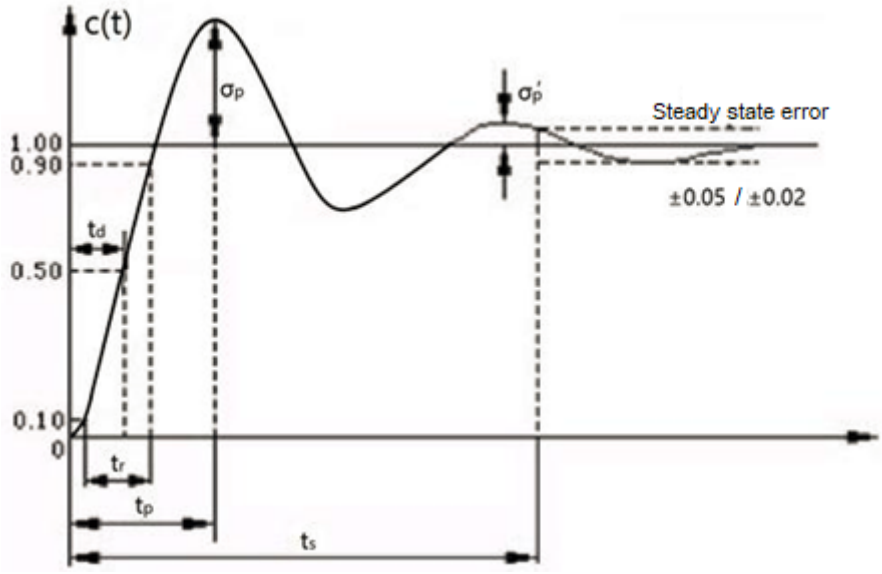
  When 0 < δ < 1, the system is underdamped. The unit step response of an underdamped second-order system is stable and eventually approaches the unit step signal.

  When δ = 0, the system is undamped. In this case, the system exhibits constant-amplitude oscillations (the time-domain response is a sine function), with amplitude k and period $2*\pi/(W_n)$.

  When δ < 0, the system is negatively damped, and the system will diverge over time.

  When δ = 1, the system is critically damped. When δ > 1, the system is overdamped. In both cases, the response slowly rises to steady state without overshoot.

  As the damping ratio increases, the system's response speed gradually slows down, and the overshoot decreases accordingly.



1. Rise time t_r: The time required for the step response curve to rise from 10% to 90% of the steady-state value.

$$t_r = \frac{\pi - \beta}{W_n * \sqrt{1-\delta^2}}, \quad \beta = arctan\frac{\sqrt{1-\delta^2}}{\delta}$$

2. Peak time t_p: The time required for the step response curve to cross the steady-state value and reach the first peak.

$$t_p = \frac{\pi}{W_n * \sqrt{1-\delta^2}}$$

3. Settling time t_s: The shortest time required for the step response curve to reach and permanently remain within a specified tolerance (typically ±2%).

$$t_p = \frac{4}{W_n * \delta}$$

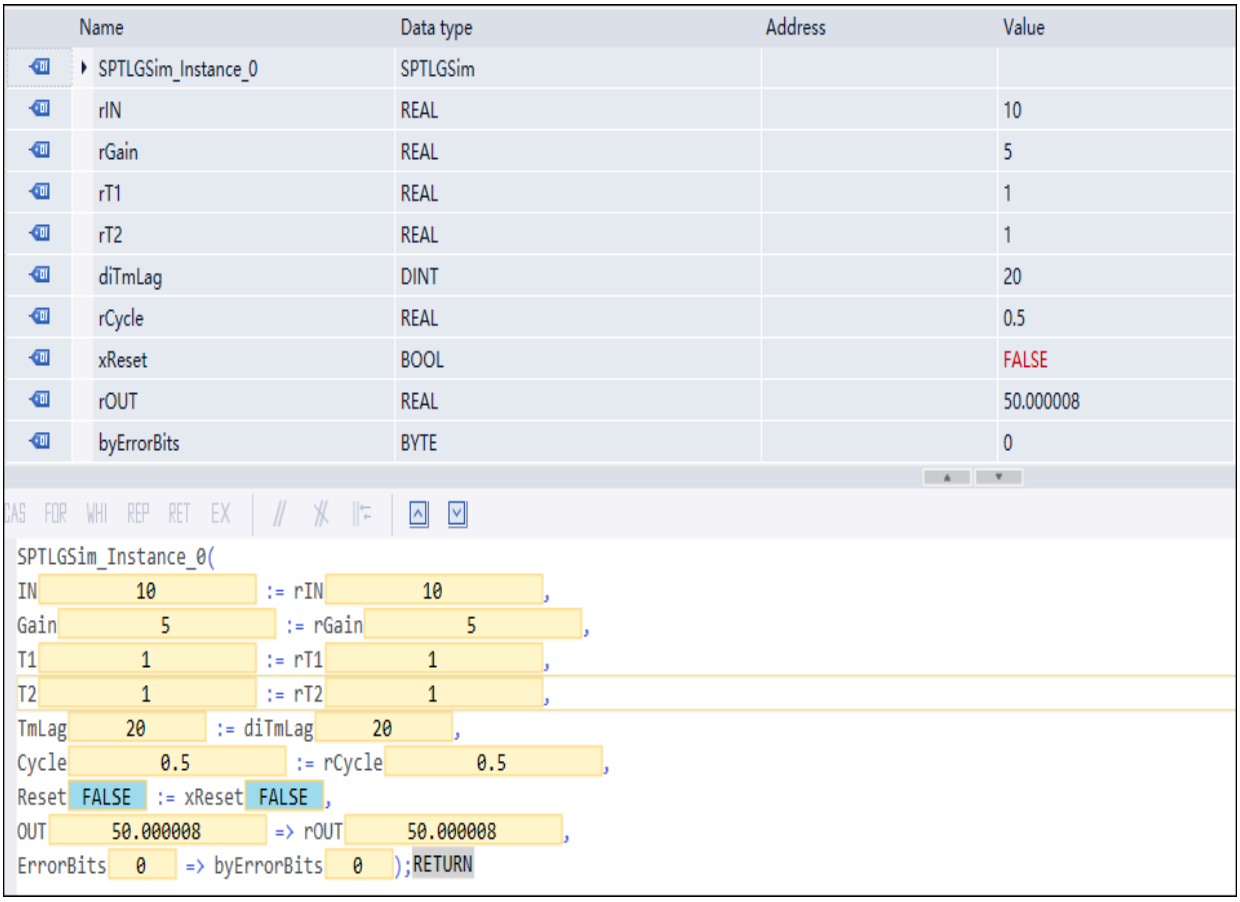
4. Overshoot σ_p: The percentage of the maximum deviation of the step response curve from the final value.

$$\sigma_p = e^{-\frac{\delta*\pi}{\sqrt{1-\delta^2}}}$$

TmLag is the number of dead time cycles of the system.

- **Program example**

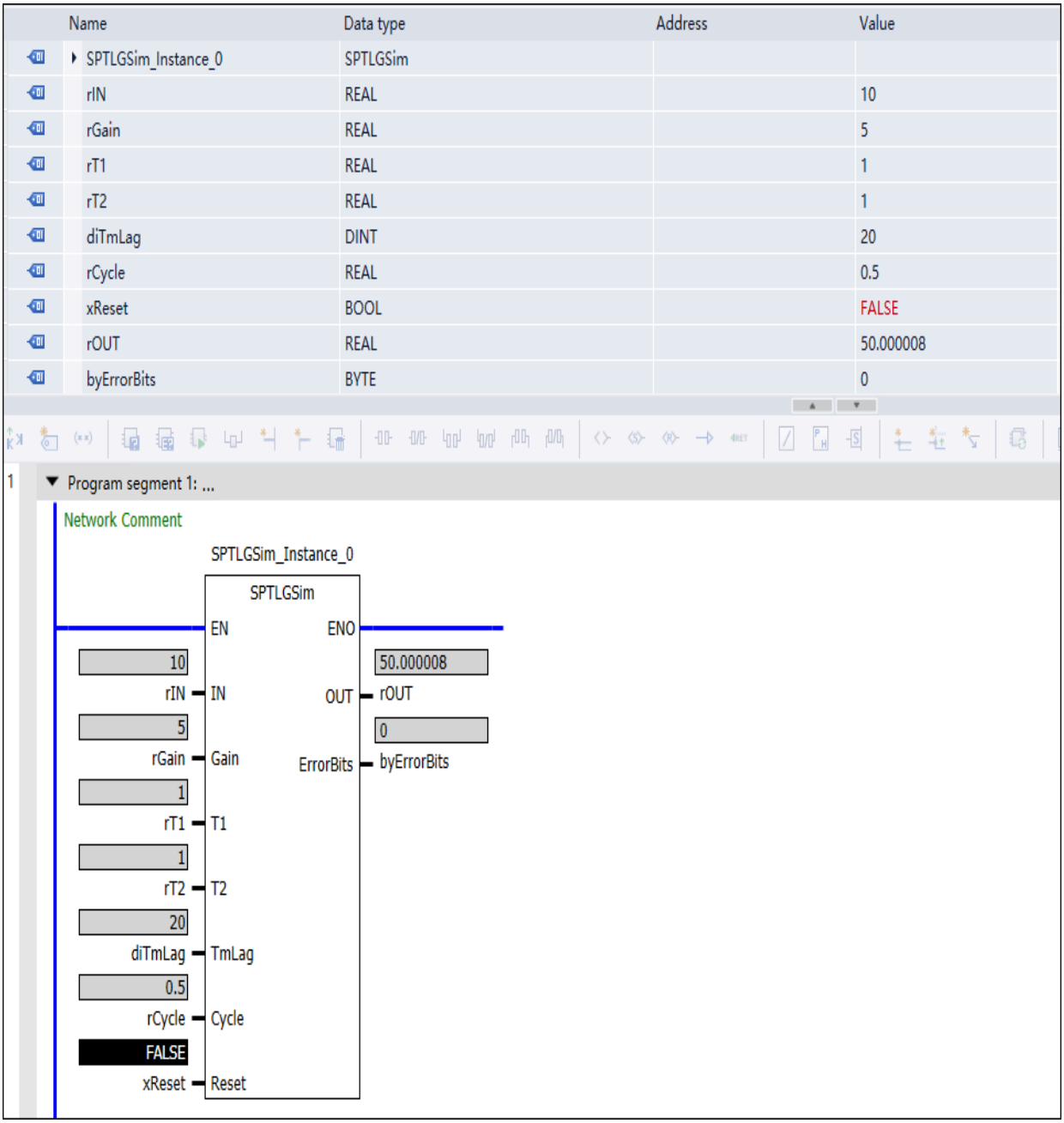
ST:



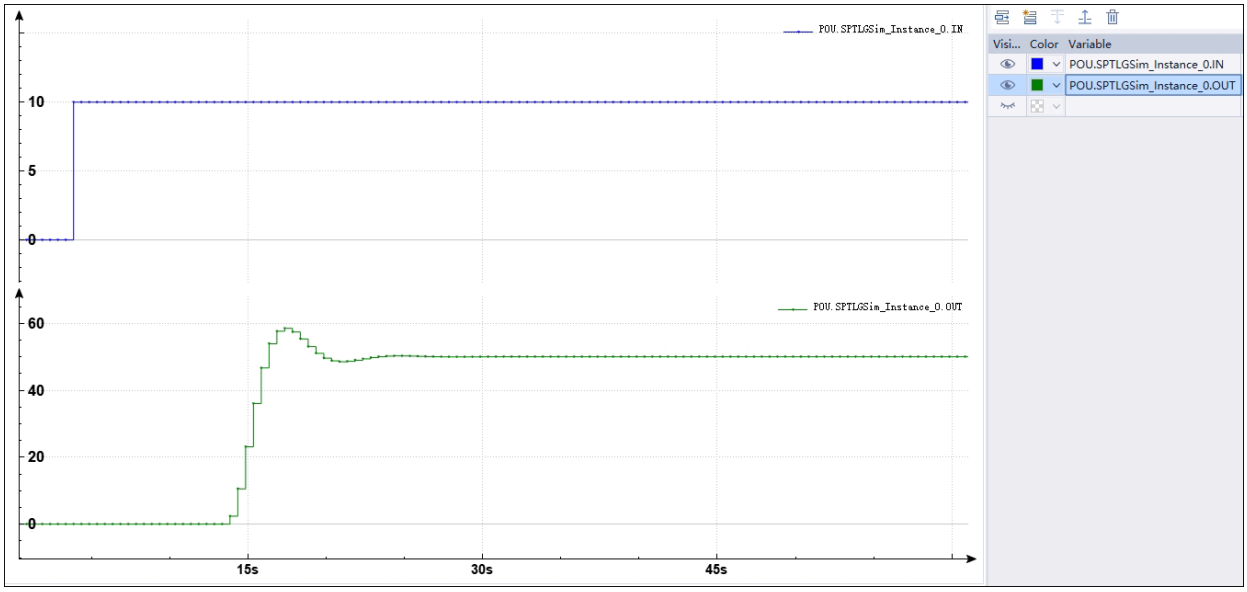
Name	Data type	Address	Value
SPTLGSim_Instance_0	SPTLGSim		
rIN	REAL		10
rGain	REAL		5
rT1	REAL		1
rT2	REAL		1
diTmLag	DINT		20
rCycle	REAL		0.5
xReset	BOOL		FALSE
rOUT	REAL		50.000008
byErrorBits	BYTE		0

```
SPTLGSim_Instance_0(
IN := rIN,
Gain := rGain,
T1 := rT1,
T2 := rT2,
TmLag := diTmLag,
Cycle := rCycle,
Reset := xReset,
OUT => rOUT,
ErrorBits => byErrorBits);RETURN
```

LD:



Operation result:



- **Precautions**
  - A larger pure dead time constant TmLag indicates a later curve response.
  - The actual dead time is the product of TmLag and cycle period.
  - Set the Cycle parameter to be consistent with the POU scheduling cycle when using the instruction.
  - When IN exceeds the limit, the system will hold the output from the previous cycle and ErrorBits will be set to 4, indicating incorrect parameter settings.
  - If an input parameter value exceeds the specified range, the system holds the previous output and sets ErrorBits to 4, indicating incorrect parameter settings.
  - The second-order time constant T2 must be set to a non-negative value. Otherwise, ErrorBits will be set to 4. When T2 is 0, the system can be simplified to a first-order dead time system.
  - Due to model accuracy limitations, the accuracy of OUT is $1.0E^{-7}$.
  - If system input parameters (other than the input IN) need to be modified, go offline, reconfigure, download, and then perform the simulation operation.
  - This instruction requires a significant amount of calculation. When using it, pay attention to designing the system task scheduling cycle appropriately. When calling a single instruction block, the minimum recommended task scheduling cycle is 5 ms. When multiple instructions of this type are called in the same task, the corresponding task scheduling cycle must be increased proportionally. Otherwise, PLC may malfunction.
  - Affected by the cycle period, the OUT value may not accurately reach the theoretical value. The calculation result can be considered correct if it falls within the interval of one program cycle before or after.
  - It should be noted that when the value of ErrorBits is 4, it indicates that the system parameters are incorrectly set, and the output value will remain unchanged. In this case, one or more of the following conditions may exist. You must check the parameter settings before performing configuration simulation.

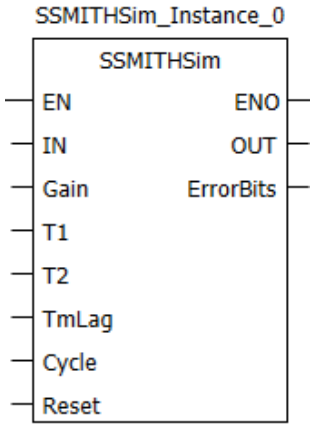
Error Report Result	Cause	Modification Suggestion
ErrorBits = 1	The calculated output value OUT exceeds the range of the data type.	Check whether the input parameter values are reasonable.

Error Report Result	Cause	Modification Suggestion
ErrorBits = 4	Input value IN overlimit	Modify the value of IN so that it is within the range of the REAL data type.
	Gain value GAIN overlimit	Modify the value of GAIN so that it is within the range [-1000000,1000000].
	Lag time Tmlag overlimit	Modify the dead time TmLag so that it is within the range of [0, 36000].
	First-order time constant T1overlimit	Modify the time constant T1 so that it is within the range (-∞, 10000].
	Second-order time constant T2 overlimit	Modify the time constant T2 so that it is within the range (0,10000].
	Cycle value Cycle overlimit	Modify the time constant T so that it is within the range (0,∞).

18.3.1.7.4 SSMITHSim

This is the instruction for second-order lag plus pure dead time compensation simulation.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SSMITHSim	Second-order lag plus pure dead time compensation simulation	FB		SSMITHSim_Instance_0(IN := , Gain := , T1 := , T2 := , TmLag := , Cycle := , Reset := , OUT => , ErrorBits =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input	Follows data type	0	Input
	Gain	Static gain	Follows data type	1	This parameter must be within the range [-1000000, 1000000].
	T1	First-order time constant	Follows data type	1	The maximum value of this parameter is 10000.
	T2	Second-order time constant	Follows data type	1	The maximum value of this parameter is 10000.
	TmLag	Pure delay time constant	Follows data type	20	Unit: calculation cycle. Range: 0 to 36000.
	Cycle	Calculation cycle	Follows data type	0.5	The logic calculation cycle must be set to a non-negative number.
	Reset	Reset flag	Follows data type	FALSE	If the reset flag is set to TRUE, the OUT is reset to 0.
Output	OUT	Output	Follows data type	0	Output.
	ErrorBits	Error report	Follows data type	-	0: No fault. 1: Calculation result overlimit. 4: Parameter setting error.

Data type

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Gain	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
T1	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
T2	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
TmLag	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
Cycle	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Reset	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Error Bits	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

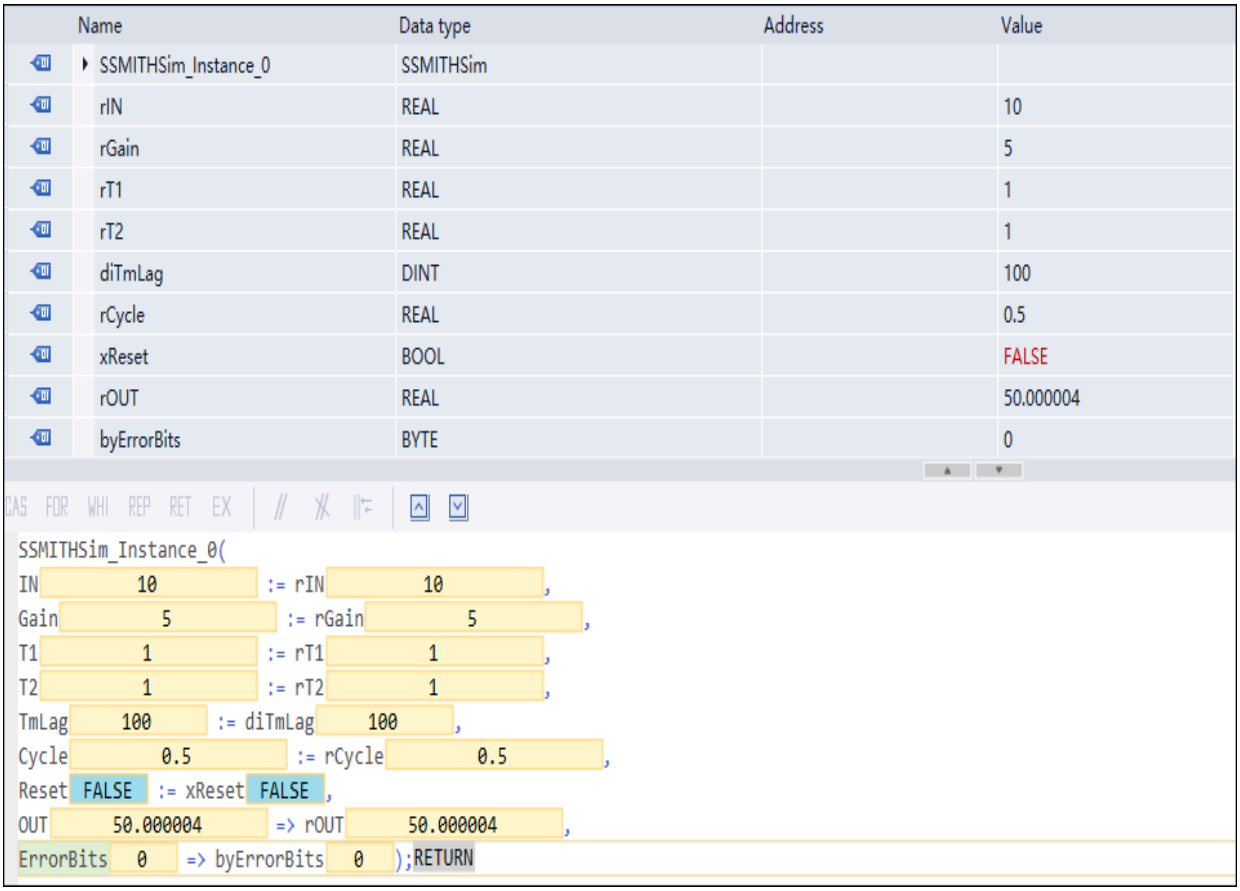
  Smith predictive control for second-order lag plus pure dead time objects, is used to compensate for the pure dead time in the control circuit and improve control performance.

  Transfer function:

$$G(s) = \frac{\mathrm{k}}{\mathrm{T2} * s^2 + \mathrm{T1} * \mathrm{s} + 1}(1 - \mathrm{e}^{-\tau \mathrm{s}})$$

- **Program example**

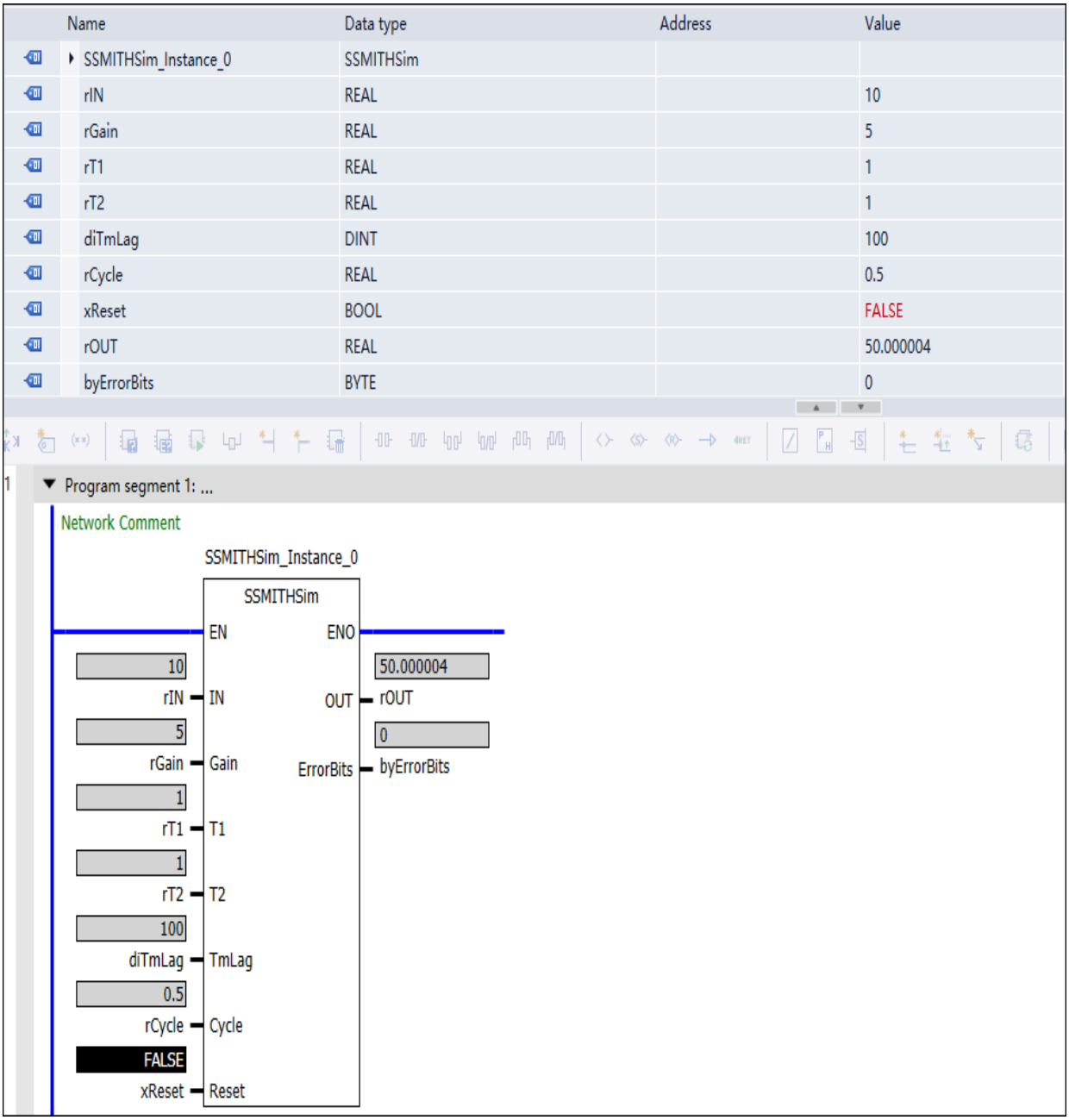
  ST:



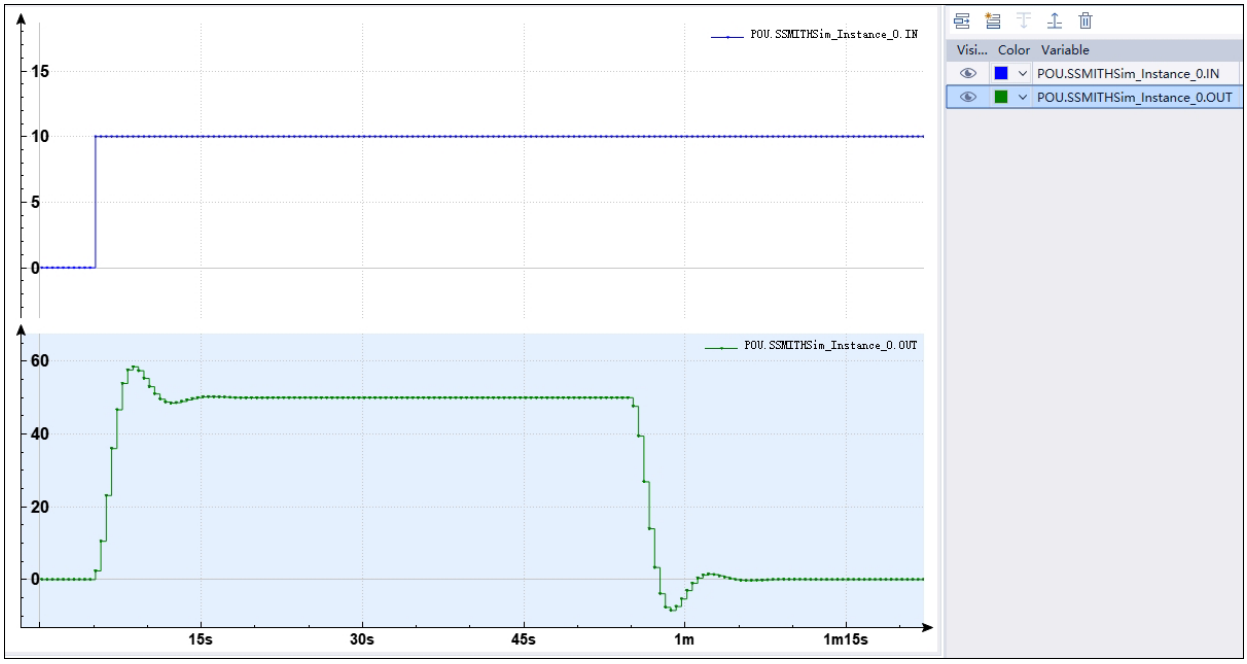
Name	Data type	Address	Value
SSMITHSim_Instance_0	SSMITHSim		
rIN	REAL		10
rGain	REAL		5
rT1	REAL		1
rT2	REAL		1
diTmLag	DINT		100
rCycle	REAL		0.5
xReset	BOOL		FALSE
rOUT	REAL		50.000004
byErrorBits	BYTE		0

```
SSMITHSim_Instance_0(
IN := rIN,
Gain := rGain,
T1 := rT1,
T2 := rT2,
TmLag := diTmLag,
Cycle := rCycle,
Reset := xReset,
OUT => rOUT,
ErrorBits => byErrorBits);RETURN
```

LD:



Operation result:



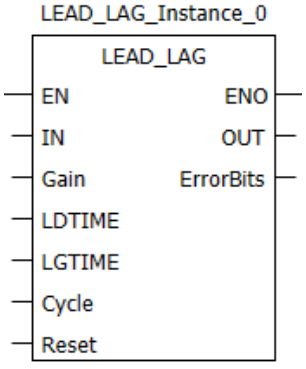
- **Precautions**
  - Set the Cycle parameter to be consistent with the POU scheduling cycle when using the instruction.
  - The second-order time constant T2 must be set to a non-negative value. Otherwise, ErrorBits will be set to 4. When T2 is 0, the system can be simplified to a first-order dead time system.
  - Within the allowed range, you can freely configure the system. Note, however, that if the system is designed as a negatively damped system (negative damping) or an undamped system (zero damping), sustained oscillations or even divergence may occur.
  - Affected by the cycle period, the OUT value may not accurately reach the theoretical value. The calculation result can be considered correct if it falls within the interval of one program cycle before or after.
  - This model is typically paired with the second-order lag plus dead time simulation model to compensate for dead time effects. After verifying the second-order dead time model, connect it in parallel with this model and sum the outputs to eliminate dead time. In this case, the output can be calculated theoretically according to the standard second-order model.
  - Due to model accuracy limitations, the accuracy of OUT is 1.0E-7.
  - If system input parameters (other than the input IN) need to be modified, go offline, reconfigure, download, and then perform the simulation operation.
  - This instruction requires a significant amount of calculation. When using it, pay attention to designing the system task scheduling cycle appropriately. When calling a single instruction block, the minimum recommended task scheduling cycle is 5 ms. When multiple instructions of this type are called in the same task, the corresponding task scheduling cycle must be increased proportionally. Otherwise, PLC may malfunction.
  - It should be noted that when the value of ErrorBits is 4, it indicates that the system parameters are incorrectly set, and the output value will remain unchanged. In this case, one or more of the following conditions may exist. You must check the parameter settings before performing configuration simulation.

Error Report Result	Cause	Modification Suggestion
ErrorBits = 1	The calculated output value OUT exceeds the range of the data type.	Check whether the input parameter values are reasonable.
ErrorBits = 4	Input value IN overlimit	Modify the value of IN so that it is within the range of the REAL data type.
	Gain value GAIN overlimit	Modify the value of GAIN so that it is within the range [-1000000,1000000].
	Lag time Tmlag overlimit	Modify the dead time TmLag so that it is within the range of [0, 36000].
	First-order time constant T1overlimit	Modify the time constant T1 so that it is within the range (-∞, 10000].
	Second-order time constant T2 overlimit	Modify the time constant T2 so that it is within the range [0,10000].
	Cycle value Cycle overlimit	Modify the time constant T so that it is within the range (0,∞).

18.3.1.7.5 LEAD_LAG

Lead-lag

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
LEAD_LAG	Lead-lag compensator	FB		LEAD_LAG_Instance_0(IN := , Gain := , LDTIME := , LGTIME := , Cycle := , Reset := , OUT => , ErrorBits =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input	Follows data type	-	Input
	Gain	Static gain	Follows data type	1	This parameter must be within the range [-1000000, 1000000].
	LDTIME	First-order time constant	Follows data type	1	The maximum value of this parameter is 10000.
	LGTIME	Second-order time constant	Follows data type	1	The maximum value of this parameter is 10000.
	Cycle	Calculation cycle	Follows data type	0.5	The logic calculation cycle must be set to a non-negative number.
	Reset	Reset flag	Follows data type	FALSE	If the reset flag is set to TRUE, the OUT is reset to 0.
Output	OUT	Output	Follows data type	-	Output.
	ErrorBits	Error report	Follows data type	-	0: No error. 4: Parameter setting error.

Data type

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Gain	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
LDTIME	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
LGTIME	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Cycle	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Reset	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
OUT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Error Bits	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  The LEAD_LAG controller is a frequency-domain correction method that improves phase margin, gain margin, and sensitivity to enhance stability and accuracy. It is primarily used for lead/lag compensation of disturbances in feedforward control.

  Transfer function:

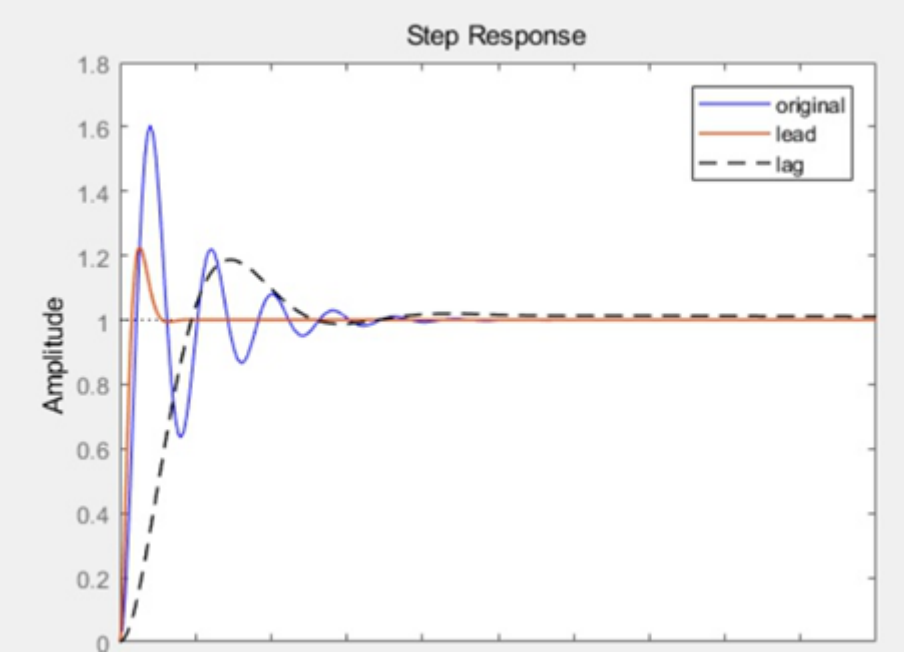
$$G(s) = GAIN * \frac{1 + \mathrm{LDTIME} * \mathrm{S}}{1 + \mathrm{LGTIME} * \mathrm{S}}$$

  Converted difference form:

$$OUT(n) = e^{-\frac{\mathrm{Cycle}}{\mathrm{LGTIME}}} * OUT(n-1) + GAIN * \frac{\mathrm{LDTIME}}{\mathrm{LGTIME}} * \mathrm{IN}(n) + GAIN * (1 - e^{-\frac{\mathrm{Cycle}}{\mathrm{LGTIME}}} - \frac{\mathrm{LDTIME}}{\mathrm{LGTIME}}) * \mathrm{IN}(n-1)$$

  T1 > T2: Lead control.

  T1 < T2: Lag control.



Both lead compensation and lag compensation increase system damping and reduce overshoot. However, lead compensation significantly speeds up the system response, while lag compensation slows it down. Lead compensation improves system responsiveness and stability. Lag compensation improves steady-state accuracy and stability but reduces responsiveness. They can be combined as "lead-lag compensation". Generally, the lead correction coefficient is designed first, followed by the lag correction coefficient.

The "lead and lag algorithm" instruction can be used with loops as a compensator in dynamic feedforward control. This instruction consists of two operations. The "lead" operation shifts the phase of the output OUT, making the output lead the input. Conversely, the "lag" operation shifts the output, making the output lag behind the input. Since the "lag" operation is equivalent to integration, it can be used as a noise suppressor or low-pass filter. The "lead" operation is equivalent to differentiation, so it can be used as a high-pass filter. When both "lead" and "lag" are used together, the output phase lags the input at low frequencies and leads the input at high frequencies. This means the "lead and lag algorithm" instruction can be used as a bandpass filter.

- **Program example**

  ST:

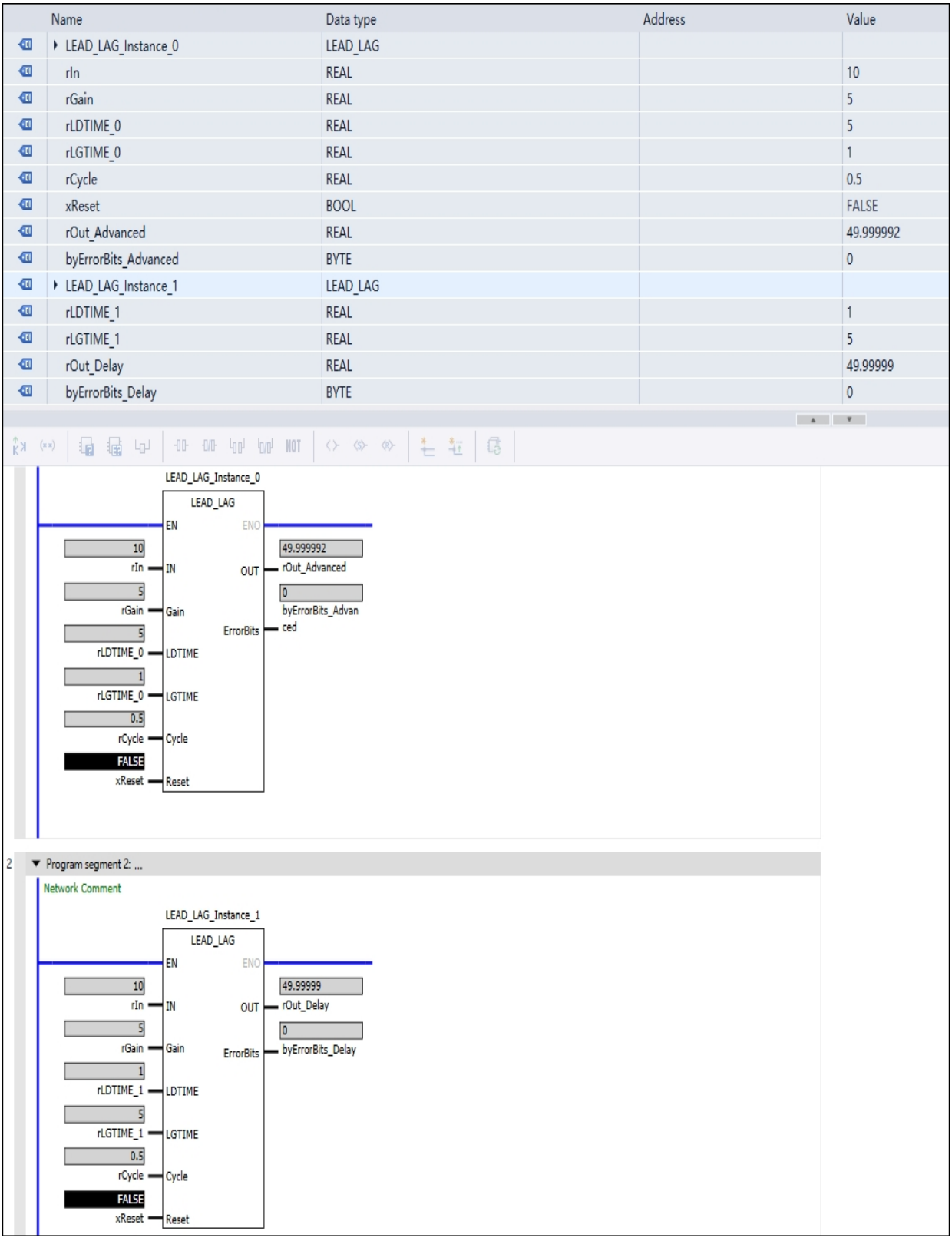
Name	Data type	Address	Value
LEAD_LAG_Instance_0	LEAD_LAG		
rIn	REAL		10
rGain	REAL		5
rLDTIME_0	REAL		5
rLGTIME_0	REAL		1
rCycle	REAL		0.5
xReset	BOOL		FALSE
rOut_Advanced	REAL		49.999992
byErrorBits_Advanced	BYTE		0
LEAD_LAG_Instance_1	LEAD_LAG		
rLDTIME_1	REAL		1
rLGTIME_1	REAL		5
rOUt_Delay	REAL		49.99999
byErrorBits_Delay	BYTE		0

```
LEAD_LAG_Instance_0(
IN          10          := rIn          10          ,
Gain        5           := rGain        5           ,
LDTIME      5           := rLDTIME_0    5           ,
LGTIME      1           := rLGTIME_0    1           ,
Cycle       0.5         := rCycle       0.5         ,
Reset FALSE             := xReset FALSE             ,
OUT       49.999992     => rOut_Advanced  49.999992 ,
ErrorBits   0     =>byErrorBits_Advanced  0     );

LEAD_LAG_Instance_1(
IN          10          := rIn          10          ,
Gain        5           := rGain        5           ,
LDTIME      1           := rLDTIME_1    1           ,
LGTIME      5           := rLGTIME_1    5           ,
Cycle       0.5         := rCycle       0.5         ,
Reset FALSE             := xReset FALSE             ,
OUT       49.99999      => rOUt_Delay   49.99999    ,
ErrorBits   0     =>byErrorBits_Delay   0     );

RETURN
```

LD:



- **Precautions**
  - Set the Cycle parameter to be consistent with the POU scheduling cycle when using the instruction.
  - Considering the actual application scenario, neither the lead time constant LDTIME nor the lag time constant LGTIME can be set to negative numbers, and the lag time constant cannot be 0. Otherwise, ErrorBits will be set to 4, indicating that the parameter settings are incorrect.
  - Affected by the cycle period, the OUT value may not accurately reach the theoretical value. The calculation result can be considered correct if it falls within the interval of one program cycle before or after.
  - Due to model accuracy limitations, the accuracy of OUT is 1.0E-7.

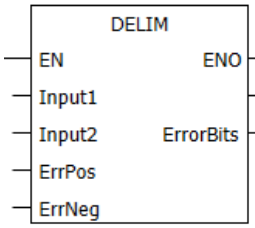
- If system input parameters (other than the input IN) need to be modified, go offline, reconfigure, download, and then perform the simulation operation.
- This instruction requires a significant amount of calculation. When using it, pay attention to designing the system task scheduling cycle appropriately. When calling a single instruction block, the minimum recommended task scheduling cycle is 5 ms. When multiple instructions of this type are called in the same task, the corresponding task scheduling cycle must be increased proportionally. Otherwise, PLC may malfunction.
- Model verification can be performed using a step input or by comparison with MATLAB.
- It should be noted that when the value of ErrorBits is 4, it indicates that the system parameters are incorrectly set, and the output value will remain unchanged. In this case, one or more of the following conditions may exist. You must check the parameter settings before performing configuration simulation.

Error Report Result	Cause	Modification Suggestion
ErrorBits = 4	Input value IN overlimit	Modify the value of IN so that it is within the range of the REAL data type.
	Gain value GAIN overlimit	Modify the value of GAIN so that it is within the range [-1000000,1000000].
	Lead time constant LDTIME overlimit	Modify the lead time constant LDTIME so that it is within the range [0,10000].
	Lag time constant LGTIME overlimit	Modify the lag time constant LGTIME so that it is within the range (0,10000].
	Cycle value Cycle overlimit	Modify the time constant T so that it is within the range (0,∞).

18.3.1.7.6 DELIM

This is the instruction for deviation determination.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DELIM	Deviation determination	FC		:= DELIM(Input1 := , Input2 := , ErrPos := , ErrNeg := , ErrorBits =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	Input1	Input Signal 1	Follows data type	-	Input Signal 1.
	Input2	Input Signal 2	Follows data type	-	Input Signal 2.
	ErrPos	Positive deviation	Follows data type	-	This value must be set to a non-negative number.
	ErrNeg	Negative deviation	Follows data type	-	This value must be set to a non-negative number.
Output	ErrorBits	Error report	Follows data type	-	0: No error. 4: Parameter setting error.

Data type

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
Input1	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Input2	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
ErrPos	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
ErrNeg	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
ErrorBits	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction determines whether the deviation between two input analog values exceeds the limit based on the set positive or negative deviation, and outputs the overlimit flag.

  The deviations are defined as ErrPos and ErrNeg. If the analog value exceeds the positive deviation ErrPos or is less than –ErrNeg, the overlimit flag outputs TRUE; otherwise, it outputs FALSE.

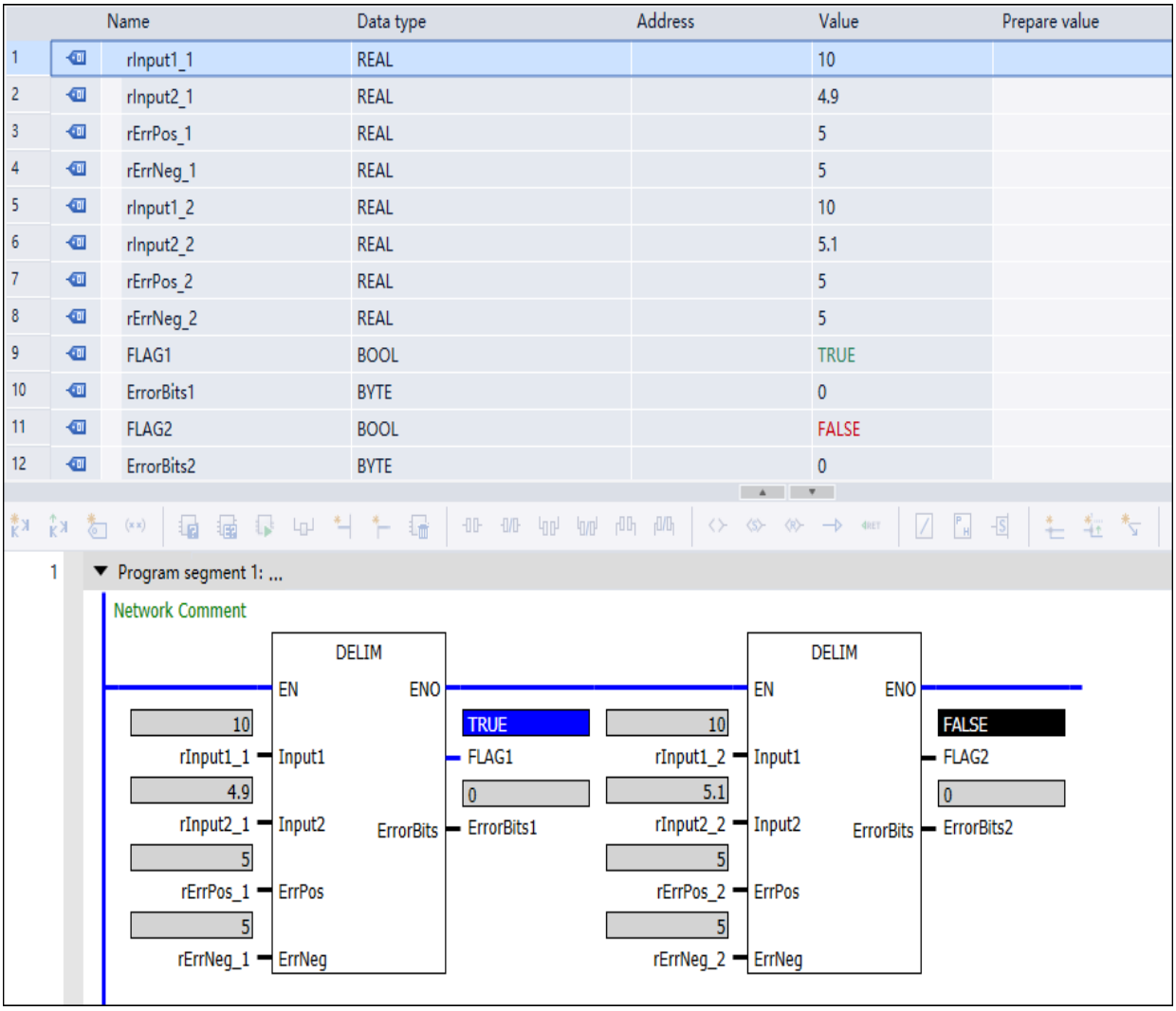
- **Program example**

  ST:

	Name	Data type	Address	Value
1	rInput1_1	REAL		10
2	rInput2_1	REAL		4.9
3	rErrPos_1	REAL		5
4	rErrNeg_1	REAL		5
5	rInput1_2	REAL		10
6	rInput2_2	REAL		5.1
7	rErrPos_2	REAL		5
8	rErrNeg_2	REAL		5
9	FLAG1	BOOL		TRUE
10	ErrorBits1	BYTE		0
11	FLAG2	BOOL		FALSE
12	ErrorBits2	BYTE		0

```
1  FLAG1 TRUE := DELIM(Input1 := rInput1_1 10 ,
2  Input2 := rInput2_1 4.9 ,
3  ErrPos := rErrPos_1 5 ,
4  ErrNeg := rErrNeg_1 5 ,
5  ErrorBits => ErrorBits1 0 );
6
7  FLAG2 FALSE := DELIM(Input1 := rInput1_2 10 ,
8  Input2 := rInput2_2 5.1 ,
9  ErrPos := rErrPos_2 5 ,
10 ErrNeg := rErrNeg_2 5 ,
11 ErrorBits => ErrorBits2 0 );RETURN
```

LD:



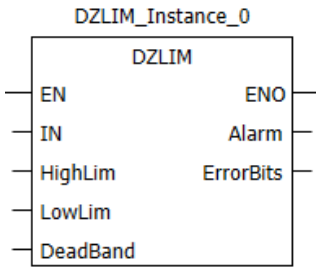
- **Precautions**
  - Due to the precision limitation of the REAL type, the measurement accuracy of this module is 10^-7.
  - Both the positive and negative deviation values must be set to non-negative numbers. Otherwise, ErrorBits will be set to 4, indicating a parameter configuration error, and the calculation result may be incorrect.
  - It should be noted that when the value of ErrorBits is 4, it indicates that the system parameters are incorrectly set. In this case, one or more of the following conditions may exist. You must check the parameter settings before performing configuration simulation.

Error Report Result	Cause	Modification Suggestion
ErrorBits = 4	Positive deviation ErrPos overlimit	Modify the positive deviation ErrPos and set it to a non-negative number
	Negative deviation ErrNeg overlimit	Modify the negative deviation ErrNeg and set it to a non-negative number

18.3.1.7.7 DZLIM

This is the instruction for dead zone check.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
DZLIM	Dead zone check	FB		DZLIM_Instance_0(IN := , HighLim := , LowLim := , DeadBand := , Alarm => , ErrorBits =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input Value	Follows data type	-	Input value.
	HighLim	Alarm high limit	Follows data type	-	Alarm high limit.
	LowLim	Alarm low limit	Follows data type	-	Low alarm limit.
	DeadBand	Alarm dead zone	Follows data type	-	Alarm dead zone
Output	Alarm	Alarm limit value	Follows data type	-	Alarm limit value.
	ErrorBits	Error report	Follows data type	-	0: No error. 4: Parameter setting error.

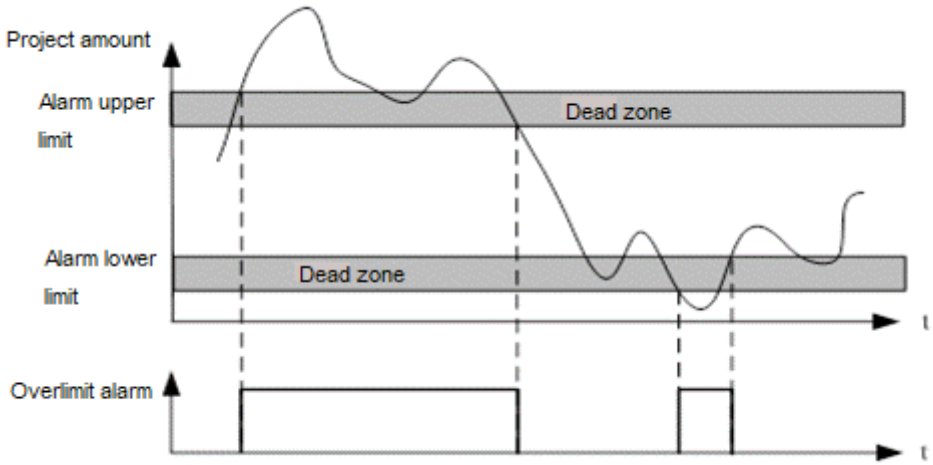
Data type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
HighLim	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
LowLim	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
DeadBand	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
Alarm	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorBits	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction judges whether an analog input exceeds the high limit (or low limit) and incorporates a dead zone function. It can be used in industrial sites to determine if a signal exceeds the high or low limit value.

  When the input value first crosses the alarm high limit or alarm low limit, the Alarm is set to TRUE, and it remains TRUE until the input value crosses the dead zone, at which point the Alarm returns to FALSE.



- **Program example**

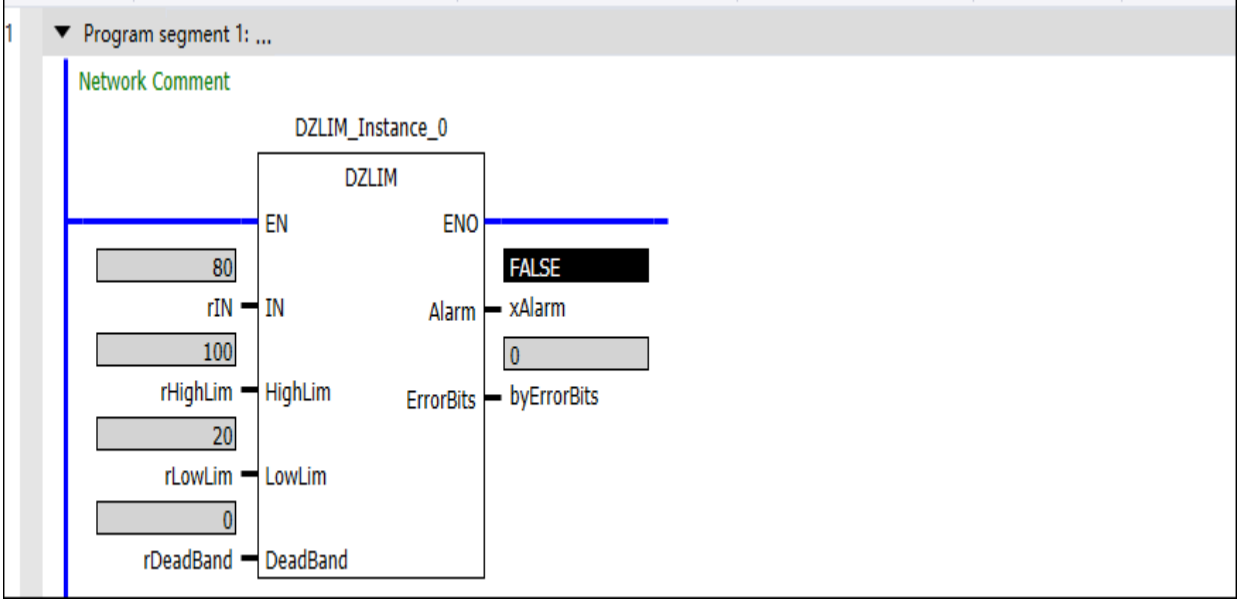
  ST:

Name	Data type	Address	Value
DZLIM_Instance_0	DZLIM		
rIN	REAL		80
rHighLim	REAL		100
rLowLim	REAL		20
rDeadBand	REAL		0
xAlarm	BOOL		FALSE
byErrorBits	BYTE		0

```
DZLIM_Instance_0(
IN := 80,
HighLim := 100,
LowLim := 20,
DeadBand := 0,
Alarm => xAlarm,
ErrorBits => byErrorBits);RETURN
```

LD:

Name	Data type	Address	Value
DZLIM_Instance_0	DZLIM		
rIN	REAL		80
rHighLim	REAL		100
rLowLim	REAL		20
rDeadBand	REAL		0
xAlarm	BOOL		FALSE
byErrorBits	BYTE		0



- **Precautions**
  - Limited by the precision of the REAL type, the measurement accuracy of this module is 10^{-7}.
  - If the dead zone length is negative, the previous output is maintained and ErrorBits is set to 4, indicating incorrect parameter settings.

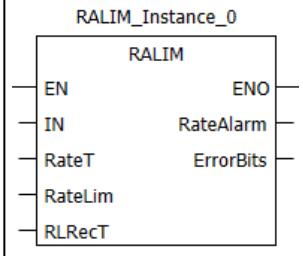
- If the input high limit is less than or equal to the input low limit, the previous output is maintained and ErrorBits is set to 4, indicating incorrect parameter settings.
- It must be noted that when the ErrorBits = 4, it indicates that the system's parameter are set incorrectly, and the system will maintain the previous output. In this case, one or more of the following situations may occur, and you need to check the parameter settings before performing configuration simulation:

Error Report Result	Cause	Modification Suggestion
ErrorBits = 4	In the input parameters, the input high limit HighLim is less than the input low limit LowLim.	Modify the setpoint so that the input high limit HighLim is greater than the input low limit LowLim.
	Dead zone length DeadBand overlimit	Set the dead zone length DeadBand to a non-negative value.

18.3.1.7.8 RALIM

This is the instruction for rate determination.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
RALIM	Rate determination	FB		RALIM_Instance_0(IN := , RateT := , RateLim := , RLRecT := , RateAlarm => , ErrorBits =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	IN	Input	Follows data type	-	Input
	RateT	Rate determination time	Follows data type	-	Unit: s.
	RateLim	Rate alarm limit	Follows data type	-	This is an absolute value limit.

Input/Output	Variable	Name	Valid Range	Default Value	Description
	RLRecT	Rate alarm auto-recovery delay	Follows data type	-	Unit: s.
Output	RateAlarm	Rate alarm	Follows data type	-	Rate alarm.
	ErrorBits	Error report	Follows data type	-	0: No error. 4: Parameter setting error.

Data type

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
RateT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
RateLim	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
RLRecT	–	–	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–
RateAlarm	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ErrorBits	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction determines whether the change rate of the input analog exceeds the limit within the set time.

  - **Rate alarm determination:**

    The time unit of rate change: seconds (internally converted to cycles). To determine whether the rate exceeds the limit, the current value is compared with the value from the set time ago. If the difference exceeds the configured rate limit, a rate alarm is triggered. The memory stores up to 20 historical data values. If the set time increases, the sampling interval also increases, reducing accuracy. For example, if the rate change time is set to 20s and the calculation cycle is 0.5s, data is recorded every two cycles.

  - **Rate alarm recovery:**

An alarm recovery time can be set. After a rate alarm occurs, if no new alarm is detected during the recovery period, the alarm automatically clears.

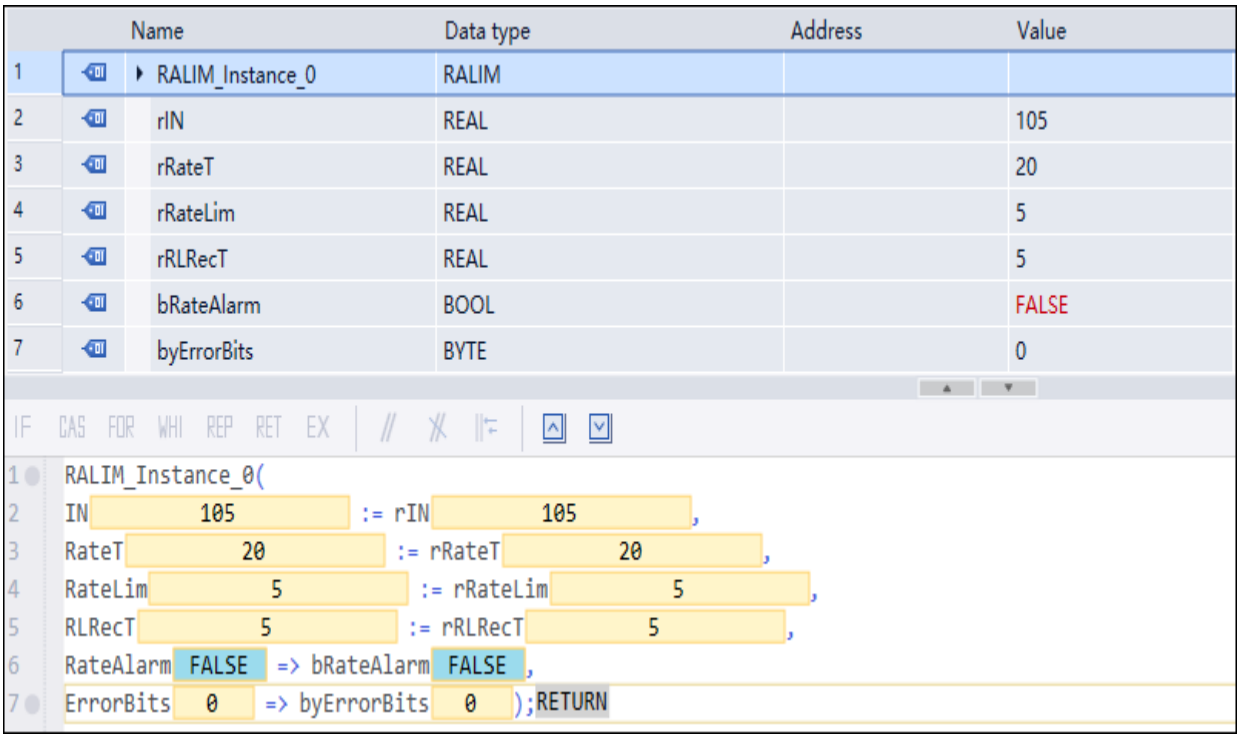
- **Program example**

  Set MainTask cycle to 1s, rate determination time RateT to 20s, rate alarm limit RateLim to 5, and rate alarm auto-recovery delay RLRecT to 5s.

  Set input IN to 0 first, and then change it to 105. Since (105 - 0) > 5, the input rate of change exceeds the rate alarm limit and the output is set high. An overlimit is detected within 20s.

  After 20s, as the stored values remain 105, the rate change is 0 and no overlimit condition exists. Alarm recovery timing then starts. Upon expiration of the rate alarm auto-recovery delay (5s), the system output is set low. The total high-level output duration is 25s. The operation result is shown in the figure below.

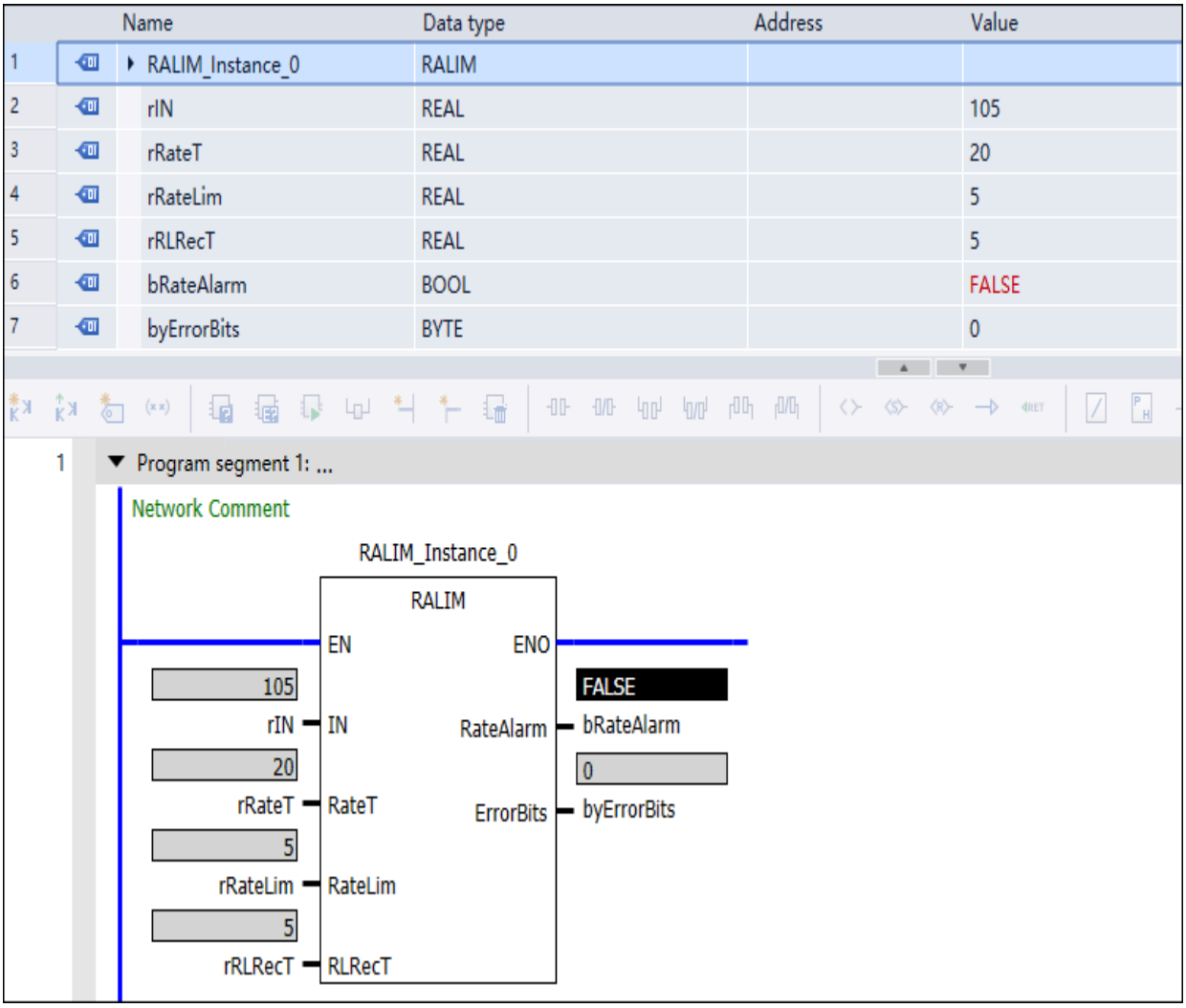
  ST:



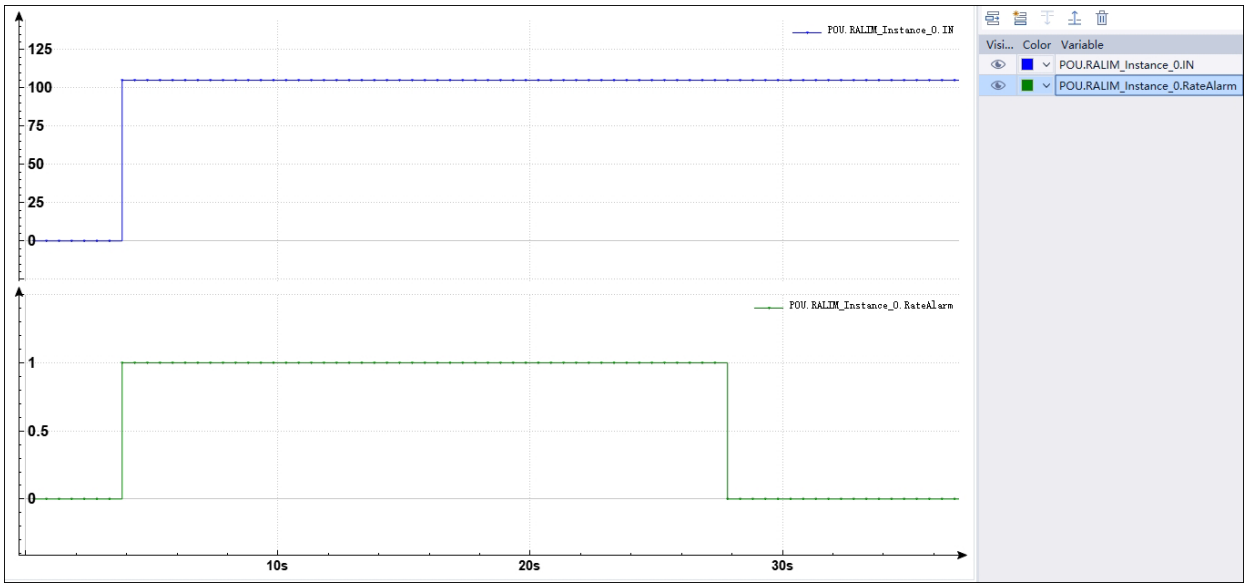
	Name	Data type	Address	Value
1	RALIM_Instance_0	RALIM		
2	rIN	REAL		105
3	rRateT	REAL		20
4	rRateLim	REAL		5
5	rRLRecT	REAL		5
6	bRateAlarm	BOOL		FALSE
7	byErrorBits	BYTE		0

```
RALIM_Instance_0(
IN := 105,
RateT := 20,
RateLim := 5,
RLRecT := 5,
RateAlarm => bRateAlarm,
ErrorBits => byErrorBits );RETURN
```

  LD:



Operation result:



- **Precautions**
  - The memory stores a maximum of 20 historical values. If the calculation cycle is 1s and the rate change time is 10s, data is recorded each cycle (10 records total). For a rate change time of 20s and 35s, data is recorded each cycle (20 records total) and every two cycles (18 records total, rounded up) respectively.
  - Limited by the precision of the REAL type, the measurement accuracy of this module is 10^{-7}.

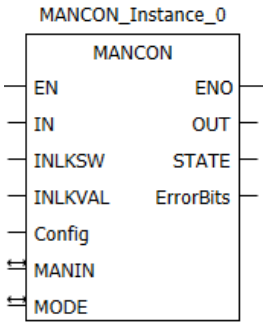
- If parameters such as RateT or RLRecT are set to negative numbers, the calculation results are not affected but may deviate from your expectations. In this case, ErrorBits is set to 4, indicating a parameter configuration error.
- The rate determination time and recovery time must be greater than the POU scheduling cycle.
- When the rate alarm limit changes, there are two scenarios: If the rate previously exceeded the limit but no longer does, the alarm recovery function is triggered and the test is completed. If the rate previously did not exceed the limit but now does, a rate alarm is triggered.
- If time-related parameters (alarm time RateT, alarm recovery time RLRecT) are modified online during operation, the interval between data record cycles may become shorter. For example, originally the value of RateT is 40, calculation cycle is 1s, and data is collected every two cycles; now RateT is changed to 20, and data is collected every one cycle. After parameter modification, some sampled values may not have corresponding previous interval data. Therefore, when time-related parameters are modified in the algorithm block, the system reacquires signal values over the RateT alarm time period and the alarm output remains unchanged before determination and calculation.
- On first run of the block, overlimit determination is performed after the RateT alarm time has elapsed. Until then, the alarm output remains unchanged (default: FALSE).
- The rate alarm limit RateLim is an absolute value limit. For example, an actual rate of change of 5 or -5 will both trigger a slope alarm when the value of RateLim is 4.5.
- It should be noted that when the value of ErrorBits is 4, it indicates that the system parameters are incorrectly set, and the output value will remain unchanged. In this case, one or more of the following conditions may exist. You must check the parameter settings before performing configuration simulation.

Error Report Result	Cause	Modification Suggestion
ErrorBits = 4	Rate determination time RateT overlimit	Set RateT to a non-negative number
	Alarm recovery time RLRecT overlimit	Set RLRecT to a non-negative number
	Rate alarm limit RateLim overlimit	Set RateLim to a non-negative number

18.3.1.7.9 MANCON

This is the instruction for manual operator.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
MANCON	Manual operator	FB		MANCON_Instance_0(IN := , INLKSW := , INLKVAL := , Config := , MANIN := , MODE := , OUT => , STATE => , ErrorBits =>);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Data type	Initial Value	Description
Input	IN	Input value	REAL	0.0	Input value.
	INLKSW	Interlock switch	BOOL	FALSE	Interlock switch. When set to TRUE, the interlock mode is enabled.
	INLKVAL	Interlock safety value	REAL	0.0	Interlock safety value.
	Config	Parameter configuration	MANCON_Config	-	Parameter configuration

<table>
<tr><th>Input/ Output</th><th>Variable</th><th>Name</th><th>Data type</th><th>Initial Value</th><th>Description</th></tr>
<tr><td></td><td></td><td colspan="5">
<table>
<tr><th>Variable</th><th>Name</th><th>Data type</th><th>Initial Value</th><th>Description</th></tr>
<tr><td>OutOffset</td><td>Output compensation</td><td>REAL</td><td>0.0</td><td>The compensation value is added to the input, which is active in automatic mode.</td></tr>
<tr><td>OUT_HLM</td><td>Upper output limit</td><td>REAL</td><td>100.0</td><td>Upper output limit.</td></tr>
<tr><td>OUT_LLM</td><td>Lower output limit</td><td>REAL</td><td>0.0</td><td>Lower output limit.</td></tr>
<tr><td>OUT_RAT</td><td>Output change limit</td><td>REAL</td><td>0.0</td><td>Output change limit. The default is 0, indicating no change detection.</td></tr>
</table>
</td></tr>
<tr><td rowspan="2">In-out</td><td>MANIN</td><td>Manual value</td><td>REAL</td><td>-</td><td>Manual input value.</td></tr>
<tr><td>MODE</td><td>Operation mode</td><td>UINT</td><td>0</td><td>0: Manual.
1: Automatic.</td></tr>
<tr><td rowspan="3">Output</td><td>OUT</td><td>Output value</td><td>REAL</td><td>0.0</td><td>Output value</td></tr>
<tr><td>STATE</td><td>Operation state</td><td>BYTE</td><td>0</td><td>0: Initial.
1: Manual.
2: Automatic.
3: Interlocked.</td></tr>
<tr><td>ErrorBits</td><td>Error message list</td><td>BYTE</td><td>0</td><td>Bitwise analysis:
Bit0 = 1: Output overlimit.
Bit1 = 1: Output change overlimit.
Bit2 = 1: Parameter setting error.</td></tr>
</table>

- **Function**

  MANCON manual operator refers to open-loop control of analog regulation objects. It is commonly used for manual control of valves, frequency converter speed references, and similar actuators.

The MANCON manual operator accepts a single input signal (e.g., the output from a PID function block) and provides a single output signal that is normally sent directly to an output module.

The specific functions of the MANCON manual operator are described as follows.

- **Operation mode**

Operation Mode	Action
Automatic	The output passes the processed input signal, which is limited by the output change rate and output limit.
Manual	The output is directly controlled (by writing a value from the upper layer or assigning a value online). The manual output is subject to the output limit.
Interlock	The output is set to a predefined interlock safety value. When the interlock is exited, the manual operator defaults to manual mode. The interlock output is subject to the output amplitude.

Interlock mode has the highest priority, followed by manual mode, and then automatic mode.

- **Bumpless switchover**

Operation Mode	Action
Automatic -> Manual Interlock -> Manual	In automatic or interlock mode, the internal manual value automatically tracks the current mode output value to ensure a bumpless switchover to manual.
Manual -> Interlock Automatic -> Interlock	In interlock mode, the interlock safety value is directly output, and there will be disturbances during switching.
Manual -> Automatic Interlock -> Automatic	In automatic mode, the input value is directly associated, and there will be disturbances during switching.

- **Output compensation**

  The output compensation directly adds the compensation value to the input. The output compensation is active in automatic mode and is subject to the output change rate and output limit. It is inactive in manual or interlock mode.

  Example: In automatic mode, the input value is 35 and the output bias is 10. Without considering output limit or output change rate, the output value is 35 + 10 = 45.

- **Output change limit**

In automatic mode, the output cycle change rate is monitored. If the change between adjacent cycles exceeds the set change limit, the current output change is limited to the change limit value.

Example: The output value range is 0 to 100, and the output change limit is 10. The output value of the previous cycle is 25, and the calculated value of the MANCON module in the current cycle is 45. Then, the output change = 45 - 25 = 20. Since 20 > 10, a change limit violation alarm is triggered (ErrorBits is set to 2). The actual output of the current cycle from the MANCON is 25 + 10 = 35.

The output change limit must be set to a value greater than 0. The default value is 10. If you want to disable this function, set the limit to a sufficiently large value.

- **Output limit**

  Active in automatic, manual, and interlock modes:

  If the output value is greater than or equal to the upper limit of output range, the output is set to the upper limit.

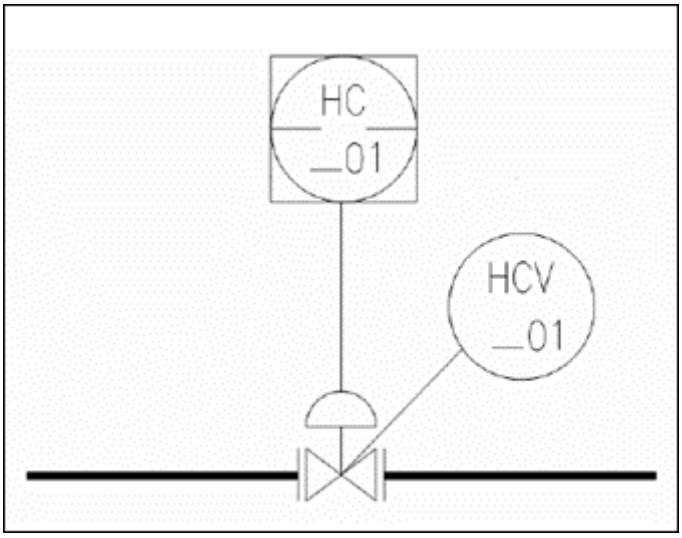
  If the output value is less than or equal to the lower limit of output range, the output is set to the lower limit.

  Errorbits description

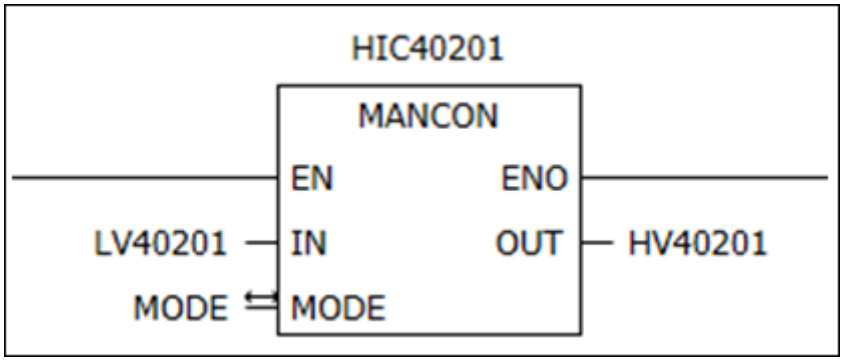
Errorbits	Note	Solution
16#0000	No error	/
16#0001	Output overlimit alarm. There are two possibilities: • OUT >= OUT_HLM (Output value greater than or equal to upper limit) • OUT <= OUT_LLM (Output value less than or equal to lower limit)	Output: Output is limited When the output restores to the normal range (OUT_LLM, OUT_HLM), the alarm is automatically cleared.
16#0002	Output rate-of-change alarm. Output change exceeds set limit in automatic mode	Output: Output change is limited The alarm is automatically cleared when the output change falls below the set limit or when switching to another mode.
16#0004	Parameter setting error alarm Includes: • The upper output limit (OUT_HLM) is less than or equal to the lower output limit (OUT_LLM). • In automatic mode, the output change limit (OUT_RAT) is less than or equal to 0.	Output: If parameters are invalid, they do not take effect and the output holds. • Check output limits to ensure OUT_HLM > OUT_LLM • Check output change limit to ensure OUT_RAT > 0

- **Application instance**

  Manual-automatic switching can be performed after the open-loop control of on-site manual control valve is completed.



Control scheme configuration:



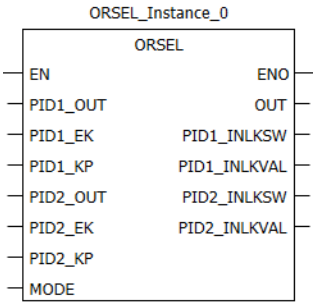
Example:

No.	Instance	Type
1	HIC40201	Tag of manual operator function block
2	HV40201	Analog output
3	LV40201	Control input

18.3.1.7.10 ORSEL

This is the instruction for override control.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ORSEL	Override control	FB		ORSEL_Instance_0(PID1_OUT := , PID1_EK := , PID1_KP := , PID2_OUT := , PID2_EK := , PID2_KP := , MODE := , OUT => , PID1_INLKSW => , PID1_INLKVAL => , PID2_INLKSW => , PID2_INLKVAL =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Data type	Initial Value	Description
Input	PID1_OUT	PID1 output	REAL	0.0	-
	PID1_EK	PID1 deviation	REAL	0.0	-
	PID1_KP	PID1 proportional gain	REAL	0.0	-
	PID2_OUT	PID2 output	REAL	0.0	-
	PID2_EK	PID2 deviation	REAL	0.0	-
	PID2_KP	PID2 proportional gain	REAL	0.0	-
Output	OUT	Output	REAL	0.0	-
	PID1_INLKSW	Associated with PID1 interlock switch INLKSW	BOOL	FALSE	-
	PID1_INLKVAL	Associated with PID1 interlock switch INLKVAL	REAL	0.0	-
	PID2_INLKSW	Associated with PID2 interlock switch INLKSW	BOOL	FALSE	-

Input/Output	Variable	Name	Data type	Initial Value	Description
	PID2_INLKVAL	Associated with PID2 interlock switch INLKVAL	REAL	0.0	-

- **Function**

  When the operation state is out of control, the system automatically switches to a new control system that takes over the control of the production process instead of initiating interlock or shutdown. When the state returns to normal, it automatically switches back to the original control system. It is necessary to determine whether the operating state is normal and to choose between the two control systems. Therefore, it is a selective control system, sometimes also referred to as substitution control or override control.

  In an override control system, the deviation of the override controller persists under normal operating conditions. If the override controller has integral action, integral windup will occur. Similarly, deviation of the normal controller persists under override conditions. If the normal controller is equipped with integral action, integral windup will also take place. In case of integral windup, timely controller switching cannot be realized.

  The key and difficulty of override control lies in fast and bumpless switchover between the normal controller and the override controller when process conditions exceed limits. The external feedback method is commonly adopted, which takes the selector output as the integral feedback signal and feeds it back to the controller in open-loop state.

  - **Calculation formula**

    Both low-select and high-select modes follow the rule:

    When Controller 1 is selected, the output of Controller 2 tracks the following value:
    OUT2=Override output+Kp2 EK2

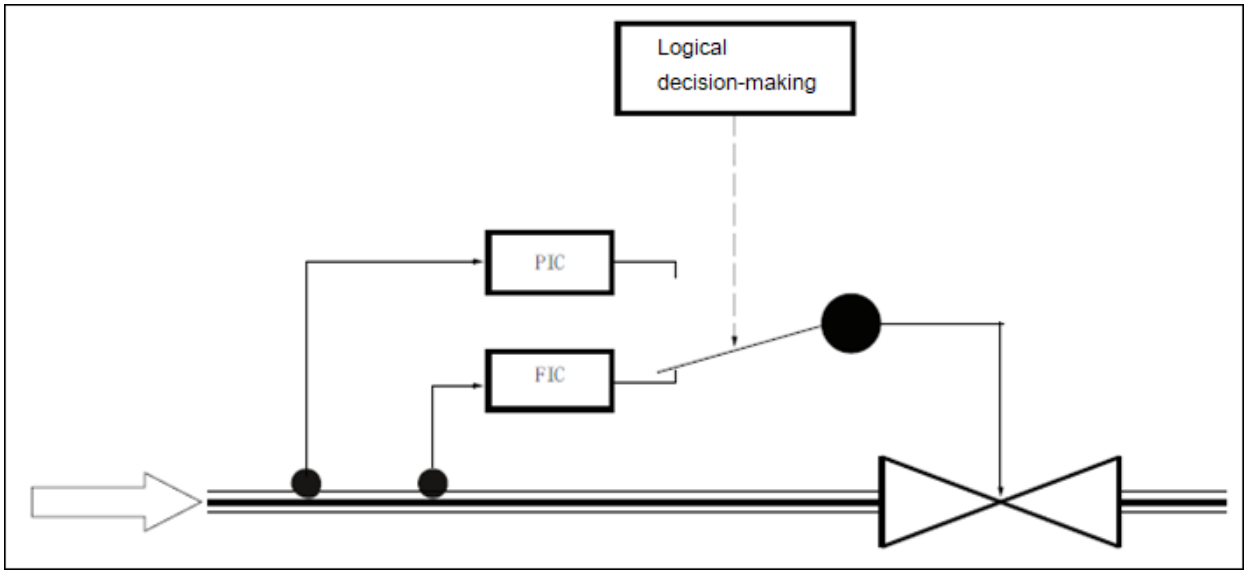
    - Kp2: proportional gain of controller 2
    - EK2: deviation of controller 2

    When Controller 2 is selected, the output of Controller 1 tracks the following value:
    OUT1=Override output+Kp1 EK1

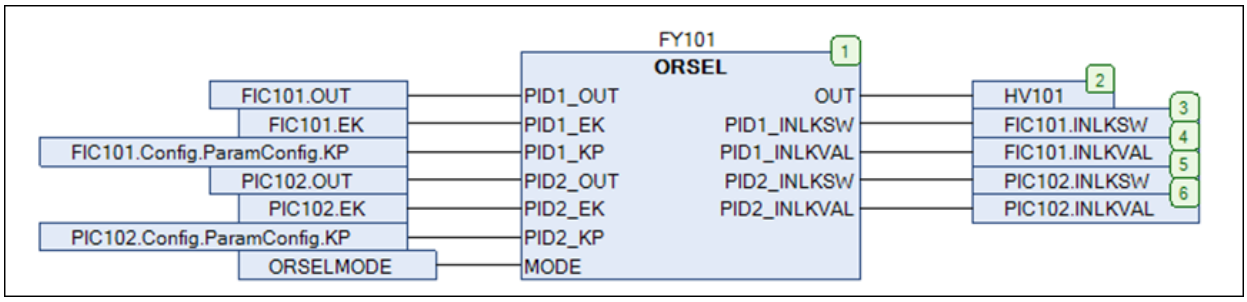
    - Kp1: proportional gain of controller 1
    - EK1: deviation of controller 1

- **Application instance**

  This is an example of a gas pipeline where pressure and flow are controlled by a single valve. The primary control is to maintain a specified flow rate, while pressure must be kept within a limited range for safety. Therefore, the pressure controller serves as the override controller.



Control scheme configuration:



Example

No.	Instance	Type
1	FIC101	Tag of PID1 function block
2	PIC102	Tag of PID2 function block
3	FY101	Tag of override function block
4	HV101	Tag of AO output

18.3.1.7.11 PWMC

This is the instruction for pulse modulation control.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PWMC	Pulse modulation control	FB	PWMC_Instance_0 PWMC EN ENO IN OUT Config ErrorBits	PWMC_Instance_0(IN := , Config := , OUT => , ErrorBits =>);

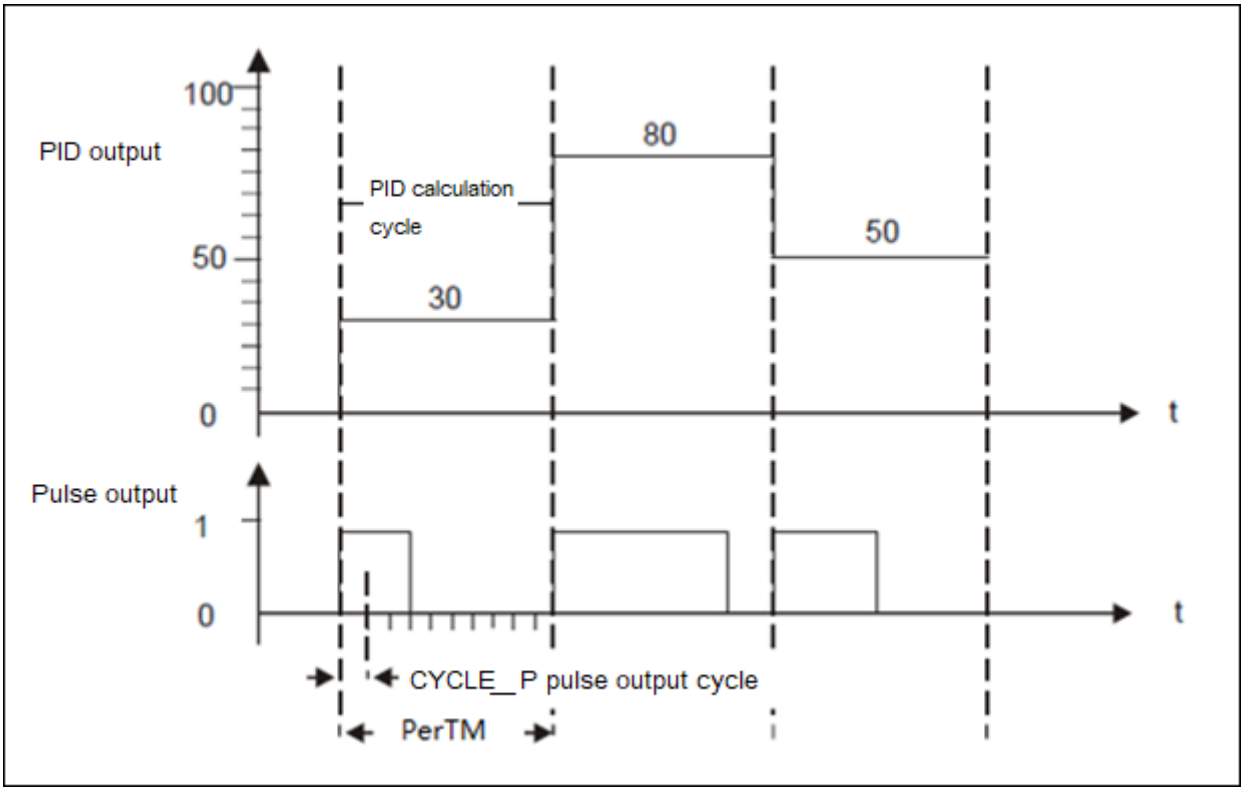
- **Related variables**

Variable

<table>
<tr><th>Input/Output</th><th>Variable</th><th>Name</th><th>Data type</th><th>Initial Value</th><th>Description</th></tr>
<tr><td rowspan="2">Input</td><td>IN</td><td>Input</td><td>REAL</td><td>0.0</td><td>Generally PID output</td></tr>
<tr><td>Config</td><td>Configuration information</td><td>PWMC_CONFIG</td><td>-</td><td>-</td></tr>
<tr><td>Input</td><td>Parameter description</td><td colspan="4"><table>
<tr><th>Variable</th><th>Name</th><th>Data type</th><th>Initial Value</th><th>Valid Range</th><th>Description</th></tr>
<tr><td>PerTM</td><td>Pulse cycle</td><td>TIME</td><td>TIME#1s0ms</td><td>-</td><td>Pulse cycle, generally set as a multiple of the calculation cycle</td></tr>
<tr><td>Min_PT</td><td>Minimum pulse time</td><td>TIME</td><td>TIME#10ms</td><td>Less than or equal to PerTM / 2</td><td>Minimum pulse time.</td></tr>
<tr><td>IN_HLM</td><td>Upper input limit</td><td>REAL</td><td>100.0</td><td>-</td><td>Upper input limit.</td></tr>
<tr><td>IN_LLM</td><td>Lower input limit</td><td>REAL</td><td>0.0</td><td>-</td><td>Lower input limit.</td></tr>
</table></td></tr>
<tr><td rowspan="2">Output</td><td>OUT</td><td>Output 1</td><td>BOOL</td><td>FALSE</td><td>-</td></tr>
<tr><td>ErrorBits</td><td>Error message list</td><td>BYTE</td><td>0</td><td>Bitwise analysis:
Bit0 = 1: Input overlimit
Bit1 = 1: Parameter setting error</td></tr>
</table>

- **Function**

  For PWM, the output is ON or OFF. The analog of PID output needs to be converted to digital output (ON/OFF) based on the duty cycle. The duty cycle is the percentage of the high-level pulse time to a whole pulse cycle.



As shown above, the PID calculation cycle is identical to the pulse cycle (PerTM). Cycle_P is the PWMC call cycle, where PerTM = 10 Cycle_P. When the PID output is 30%, PWMC outputs ON in the first three cycles and OFF in the remaining seven cycles.

- **Position accuracy**

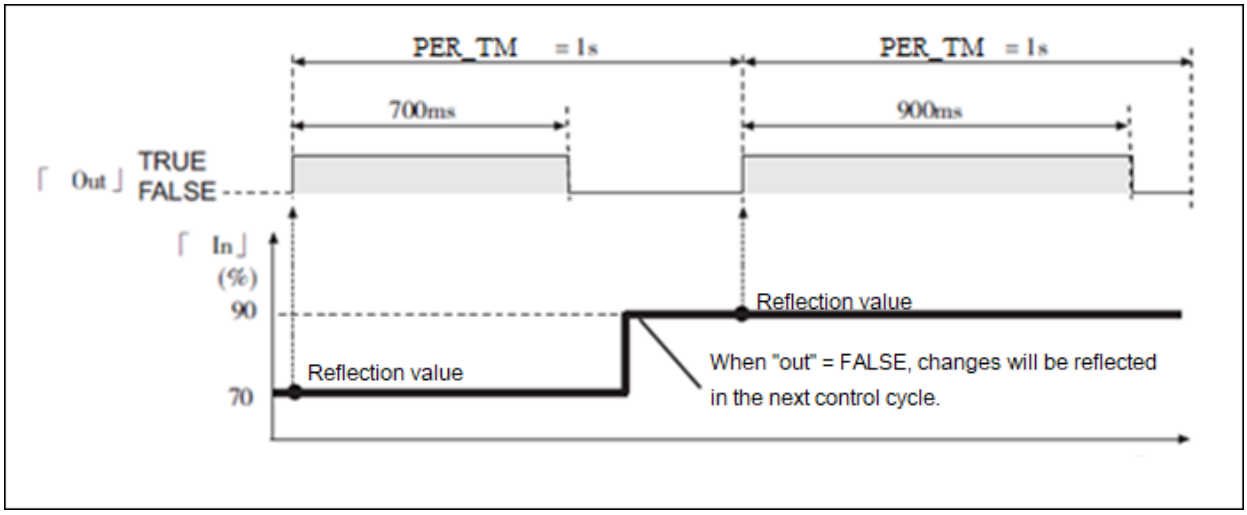
  The number of PWMC calls per PER_TM cycle determines the accuracy of pulse width, which improves as the call frequency increases.

  Control accuracy calculation formula:

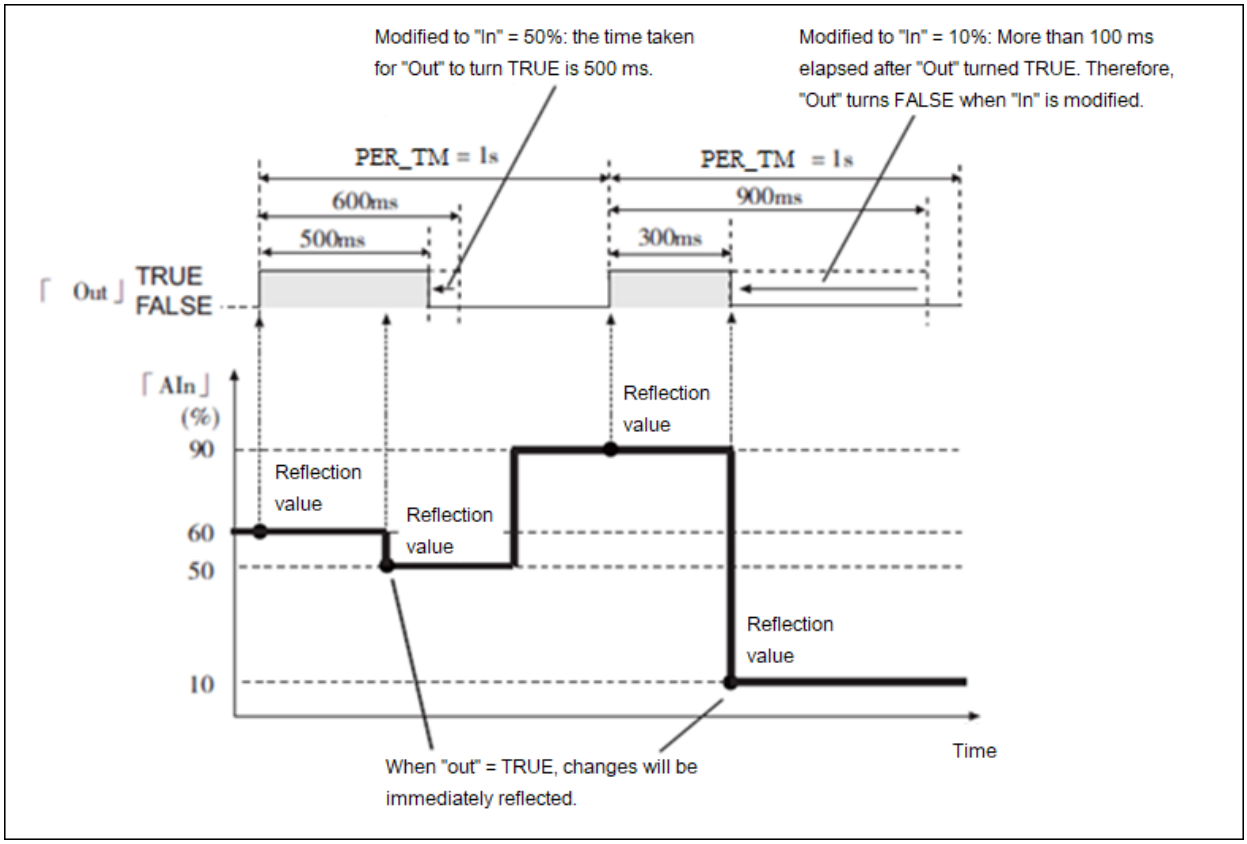
$$\text{Accuracy} = \frac{\text{Cycle}}{\text{PER TM}} * 100\%$$

  For example, if the task call cycle is 100ms and the PER_TM circle is 10s, the accuracy of the operation value range is 1%.

- **Response process**
  - When OUT is FALSE and IN changes, the change will be reflected in the next control cycle.



  - When OUT is TRUE and IN changes, the change is reflected immediately.

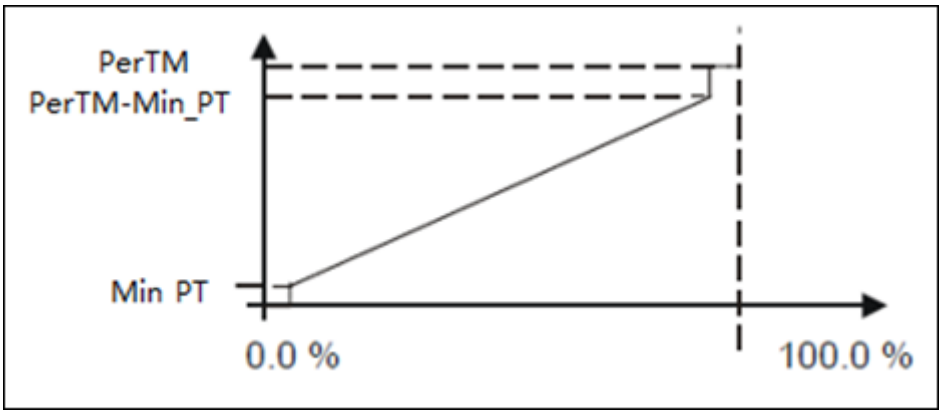


- **Minimum pulse interval**

  To avoid frequent operation of the actuator, set the minimum pulse interval Min_PT.

  If the calculated high-level pulse time is less than the minimum pulse interval Min_PT, the output is always set to FALSE.

  If the calculated high-level pulse time is greater than the pulse cycle (PerTM) and the minimum pulse interval (Min_PT), the output is always set to TRUE.



- **Precautions**
  - Set the control cycle (PER_TM) to an integer multiple of the task cycle assigned to the program. When a value that is not an integer multiple of the task cycle is set, the actual control cycle spans from the end of control cycle (PER_TM) to the start of the next task execution. For example, when the task cycle is 3ms and the value of PER_TM is 1s, the actual control cycle of 1002ms is from the end of PER_TM to the next task execution.

  - Set the values of task cycle and control cycle (PER_TM) so that the output resolution is less than 0.1%. If the output resolution exceeds 0.1%, the error between the input value and the proportion at which the output switches to TRUE will rise, degrading control performance. For example, when the value of PER_TM is 10s, set the task cycle to less than 10ms.

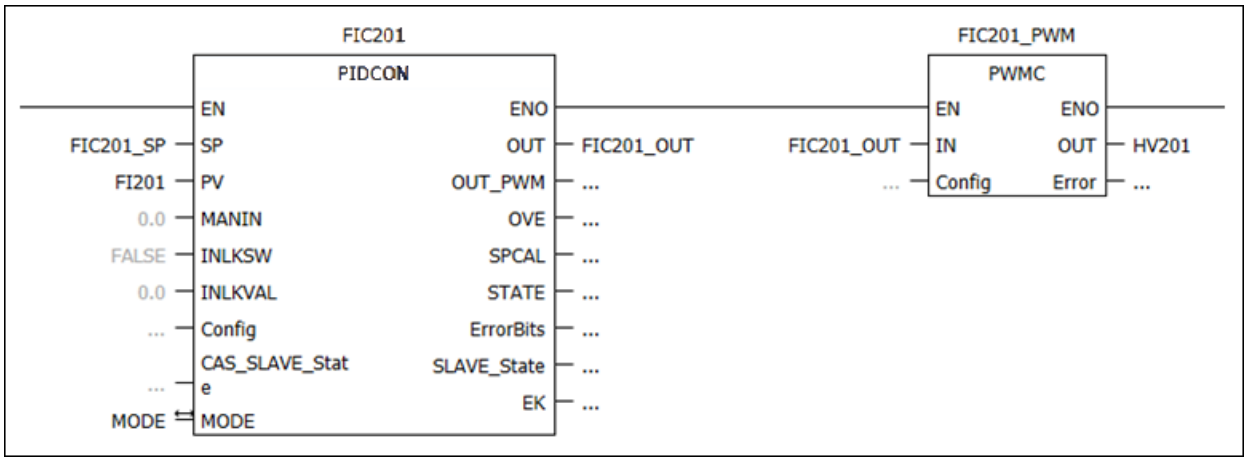
◦ An error will occur if the parameter setting is incorrect. The value of OUT is FALSE, and ErrorBits will be set. After the error is cleared, the output will respond in the next cycle.

Errorbits description

Errorbits	Note	Solution
16#0000	No error	/
16#0001	Two possible conditions trigger input limit alarm: • IN > IN_HLM (Input value greater than or equal to upper limit) • IN < IN _LLM (Input value less than or equal to lower limit)	Output OUT = FALSE
16#0002	Two possible conditions cause incorrect parameter setting: • IN_HLM <= IN_LLM • Min_PT > PerTM 0.5	Output OUT = FALSE

- **Application instance**

Control scheme configuration:



Config	PWMC_CONFIG					Private
PerTM	TIME		TIME#1s0ms			Private
Min_PT	TIME		TIME#10ms			Private
IN_HLM	REAL		100.0			Private
IN_LLM	REAL		0.0			Private

The task scheduling cycle can be set to 1ms and the sampling cycle of PIDCON (PIDCON. Config. ParamConfig. PIDSampleTime) can be set to 1s, achieving a control accuracy of 0.1%.

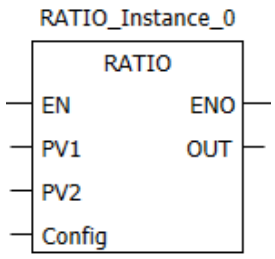
Example:

No.	Instance	Type
1	FIC201	Tag of PID1 function block
2	FI201	AI input
3	HV201	DO output

18.3.1.7.12 RATIO

This is the instruction for ratio control.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
RATIO	Ratio control	FB		RATIO_Instance_0(PV1 := , PV2 := , Config := , OUT =>);

- **Related variable**

Variable

Input/Output	Variable	Name	Data type	Initial Value	Description
Input	PV1	Main Input	REAL	-	Main input
	PV2	Slave input	REAL	-	Slave input.
	Config	Configuration information	RATIO_CONFIG	-	-

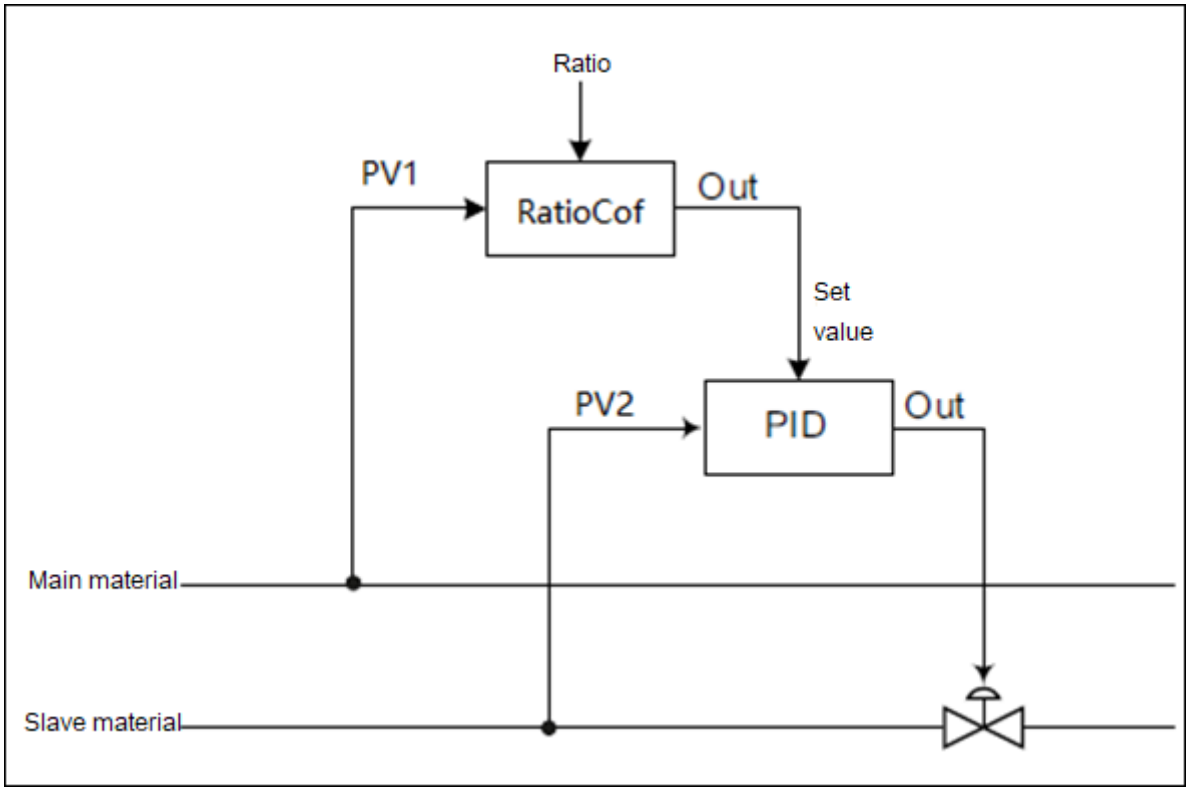
<table>
<tr><th>Input/Output</th><th>Variable</th><th>Name</th><th>Data type</th><th>Initial Value</th><th>Description</th></tr>
<tr><td>Input</td><td>Parameter description</td><td colspan="4"><table>
<tr><th>Variable</th><th>Name</th><th>Data type</th><th>Initial Value</th><th>Description</th></tr>
<tr><td>RatioOpt</td><td>Ratio source selection</td><td>BOOL</td><td>FALSE</td><td>FALSE: Internal setpoint ratio.
TRUE: External setpoint ratio.</td></tr>
<tr><td>RatioInt</td><td>Internal setpoint ratio</td><td>REAL</td><td>0</td><td>Internal setpoint ratio.</td></tr>
<tr><td>Offset</td><td>Offset value</td><td>REAL</td><td>0</td><td>Offset value.</td></tr>
<tr><td>RatTrk</td><td>Internal ratio tracks external ratio</td><td>BOOL</td><td>FALSE</td><td>A bumpless switchover from the external ratio to the internal ratio is achieved through tracking.</td></tr>
<tr><td>OUT_HLM</td><td>Output high limit</td><td>REAL</td><td>100</td><td>Output high limit.</td></tr>
<tr><td>OUT_LLM</td><td>Output low limit</td><td>REAL</td><td>0</td><td>Output low limit.</td></tr>
</table></td></tr>
<tr><td>Output</td><td>OUT</td><td>Output</td><td>REAL</td><td>0</td><td>Output.</td></tr>
</table>

- **Function**

  A ratio control system associates two or more parameters in a certain proportional relationship for control. Ratio control is a widely adopted control method in process control, extensively used in industrial production processes such as chemical and pharmaceutical industries where raw material ratios need to be controlled. Undesired ratio fluctuation usually leads to degraded product quality, waste of energy and materials, and environmental pollution, and may even cause equipment damage and personal safety accidents.

  In industrial production processes, a control system maintains two or more parameters in a fixed proportional relationship. In a ratio control system, the dominant material is not regulated by the ratio controller and defined as the master parameter (active quantity) G1. Other subordinate

controlled materials are defined as slave parameters (dependent quantities) such as G2. The purpose of ratio control is to maintain a fixed ratio G2/G1 = K. The slave parameter is adjusted to match the master parameter and stabilize the set ratio.



Function block calculation formula:

Out = PV1 RatioCof + Offset

Where:

PV1: Acquisition value of master controller parameter

RatioCof: Ratio coefficient, which can be selected based on the configuration parameter RatioOpt

RatioOpt=0: RatioInt is selected as internal ratio setpoint with configurable parameters

RatioOpt=1: RatioExt (external ratio) is selected as the actual ratio. It is calculated automatically as RatioExt = (PV2 - Offset) / PV1. The output value of OUT is PV2. PV2: Acquisition value of the slave parameter

Offset: Offset value added to the output

- **Bumpless switchover between internal and external setpoints**

  When external ratio is selected, configuration depends on the bumpless ratio parameter RatTrk. If bumpless mode is enabled with RatTrk=TRUE, the internal ratio setpoint tracks the external ratio, that is, RatioInt = RatioExt, to ensure bumpless switchover.

- **Output limiting function**

  If the output exceeds OUT_HLM, the OUT is limited to OUT_HLM.

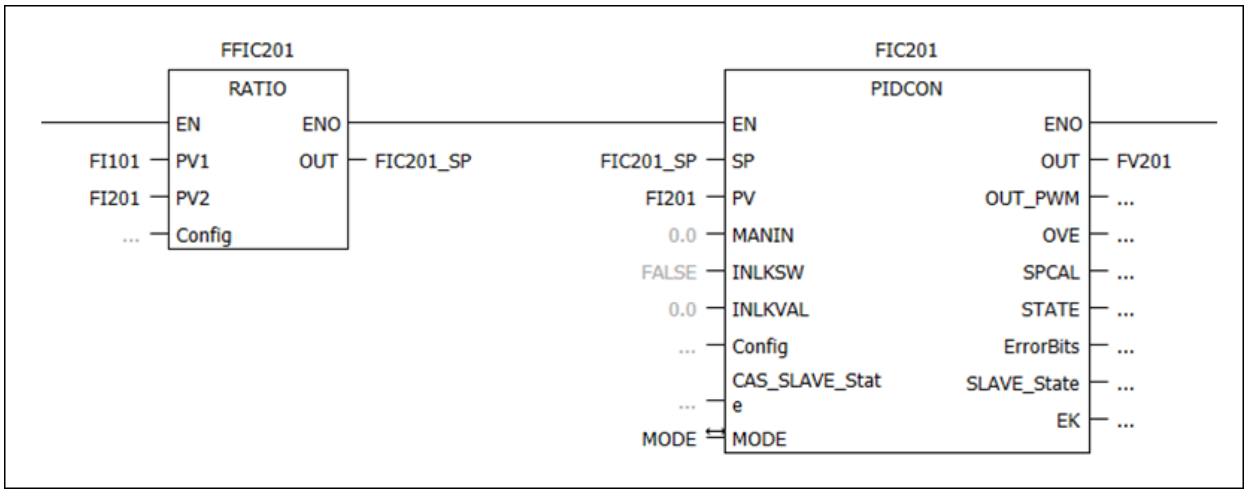
  If the output drops below OUT_LLM, the OUT is limited to OUT_LLM.

- **Application instance**

  When the lower-level PID is in automatic mode, the internal ratio is selected. The output is calculated as Out = PV1 RatioInt + Offset.

When the lower-level PID in non-automatic mode, the external ratio can be selected. The ratio block outputs PV2 and transmits it to the setpoint of the lower-level PID, keeping the setpoint consistent with the acquisition value to realize bumpless PID mode switchover. Meanwhile, set the value of ratio bumpless tracking parameter RatTrk to TRUE, ensuring bumpless switchover of ratio block parameters.

Control scheme configuration:



Example:

No.	Instance	Type
1	FFIC201	Tag of ratio control function block
2	FIC201	Tag of PID function block
3	FI101	Input of AI master variable
4	FI201	Input of AI slave variable

18.3.1.7.13 SPLRANGE

This is the split-range control instruction.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SPLRANGE	Split-range control	FB	SPLRANGE_Instance_0 SPLRANGE EN ENO IN OUT1 Config OUT2 Error	SPLRANGE_Instance_0(IN := , Config := , OUT1 => , OUT2 => , Error =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Data type	Initial Value	Description
Input	IN	Input	REAL	-	Input
	Config	Configuration information	SPLRANGE_CONFIG	-	-

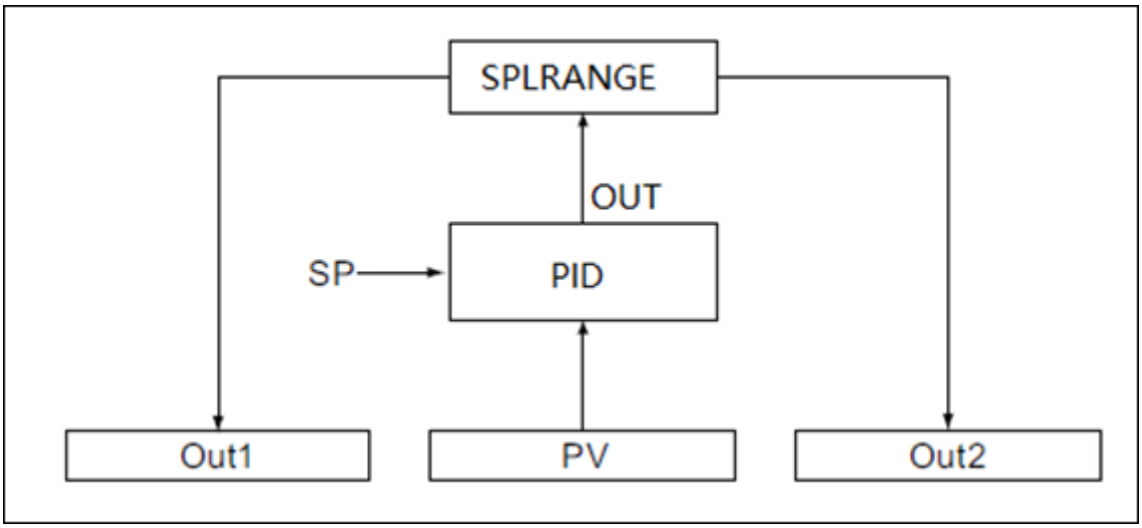
Input/Output	Variable	Name	Data type	Initial Value	Description
Input	Parameter description				

Variable	Name	Data type	Initial Value	Description
Out1OPT	Action mode of output 1	BOOL	FALSE	FALSE: Direct action. TRUE: Reverse action.
Out2OPT	Action mode of output 2	BOOL	FALSE	FALSE: Direct action. TRUE: Reverse action.
S1_HLM	Split-range upper limit of output 1	REAL	50.0	Split-range upper limit of output 1
S1_LLM	Split-range lower limit of output 1	REAL	0.0	Split-range lower limit of output 1
OUT1_HLM	Output upper limit of output 1	REAL	100.0	Output upper limit of output 1
OUT1_LLM	Output lower limit of output 1	REAL	0.0	Output lower limit of output 1
S2_HLM	Split-range upper limit of output 2	REAL	100.0	Split-range upper limit of output 2
S2_LLM	Split-range lower limit of output 2	REAL	50.0	Split-range lower limit of output 2
OUT2_HLM	Output upper limit of output 2	REAL	100.0	Output upper limit of output 2
OUT2_LLM	Output lower limit of output 2	REAL	0.0	Output lower limit of output 2

Input/Output	Variable	Name	Data type	Initial Value	Description
Output	OUT1	Output 1	REAL	-	Output 1
	OUT2	Output 2	REAL	-	Output 2
	Error	Incorrect parameter	BOOL	-	Parameter error

- **Function**

  Split-range control means that the output of a regulator drives two or more control valves at the same time, and each valve only operates within a specified range of the regulator output. It applies to scenarios where two or more control medium are adopted for regulation, or to meet control requirements under different process load conditions. The split-range control determines the split points and the action direction (direct/reverse acting) of each control valve, which needs to be configured based on actual process conditions.



  In split-range regulation, there are two types of opening/closing modes for control valves:

  ◦ Direct acting: same-direction split-range control. As the regulator output signal increases (or decreases), the valve opens wider (or closes tighter).

  ◦ Reverse acting: opposite-direction split-range control. As the regulator output signal increases (or decreases), the valve closes tighter (or opens wider).

  ◦ **Calculation formula**

  Positive action: OUTx = (IN - Sx_LLM) (OUTx_HLM - OUTx_LLM) / (Sx_HLM - Sx_LLM) + OUTx_LLM

  Reverse action: OUTx = (Sx_HLM – IN) (OUTx_HLM - OUTx_LLM) / (Sx_HLM - Sx_LLM) + OUTx_LLM

  Note: If the set upper limit is less than or equal to the lower limit (i.e., OUTx_LLM ≥ OUTx_HLM or Sx_LLM ≥ Sx_HLM), the parameter setting is invalid. In this case, Error is set to TRUE and the output holds. Normal calculation runs only when parameters are set correctly.

  The commonly used split forms are described as follows.

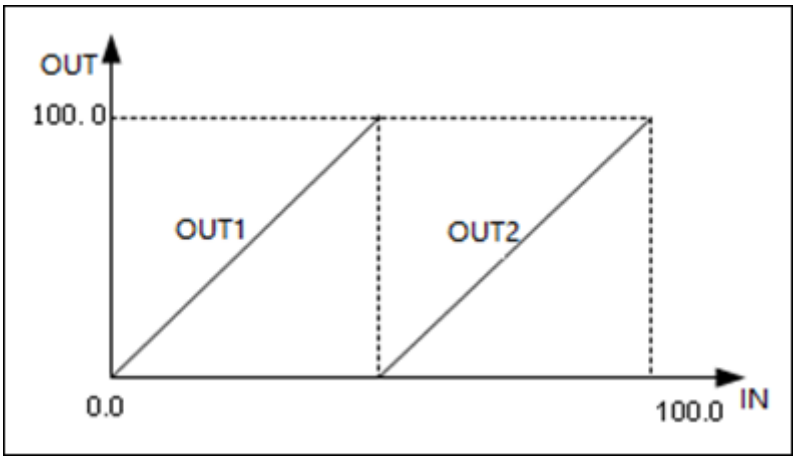
  Direct-direct split-range

  ◦ Out1OPT = false, OUT1_HLM = 100, OUT1_LLM = 0, S1_HLM = 50, S1_LLM = 0

Direct acting: OUT1=(IN - S1_LLM) (OUT1_HLM - OUT1_LLM) / (S1_HLM - S1_LLM) + OUT1_LLM = (IN - 0) 100 / 50 + 0

- Out2OPT =false, OUT2_HLM = 100, OUT2_LLM = 0, S2_HLM = 100, S2_LLM = 50

  Direct acting: OUT2=(IN - S2_LLM) (OUT2_HLM - OUT2_LLM) / (S2_HLM - S2_LLM) + OUT2_LLM = (IN - 50) 100 / 50 + 0



Reverse-reverse split-range

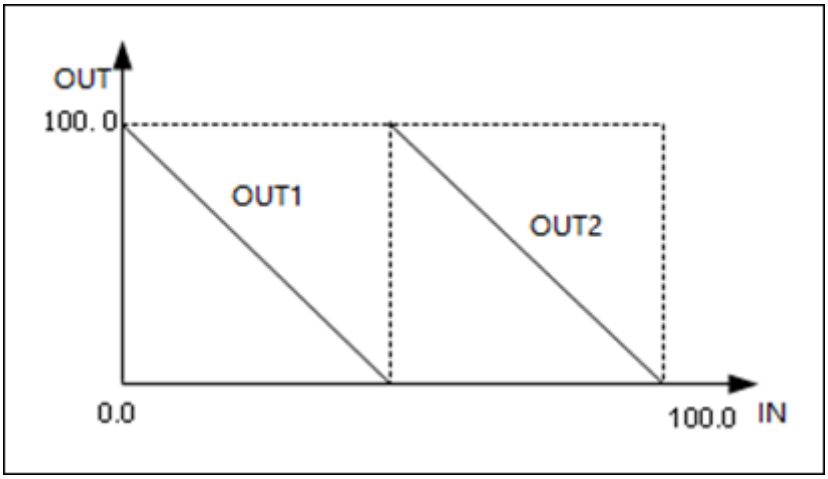
Out1OPT = true, Out2OPT = true, S1_HLM = 50, S1_LLM = 0, S2_HLM = 100, S2_LLM = 50.

- Out1OPT = true, OUT1_HLM = 100, OUT1_LLM = 0, S1_HLM = 50, S1_LLM = 0

  Reverse acting: OUT1 = (S1_HLM - IN)(OUT1_HLM - OUT1_LLM) / (S1_HLM - S1_LLM) + OUT1_LLM= (50 - IN)100 / 50 + 0

- Out2OPT = true, OUT2_HLM =100, OUT2_LLM =0, S2_HLM =100, S2_LLM =50.

  Reverse acting: OUT2 = (S2_HLM - IN)(OUT2_HLM - OUT2_LLM) / (S2_HLM - S2_LLM) + OUT2_LLM = (100 - IN) 100 / 50 + 0



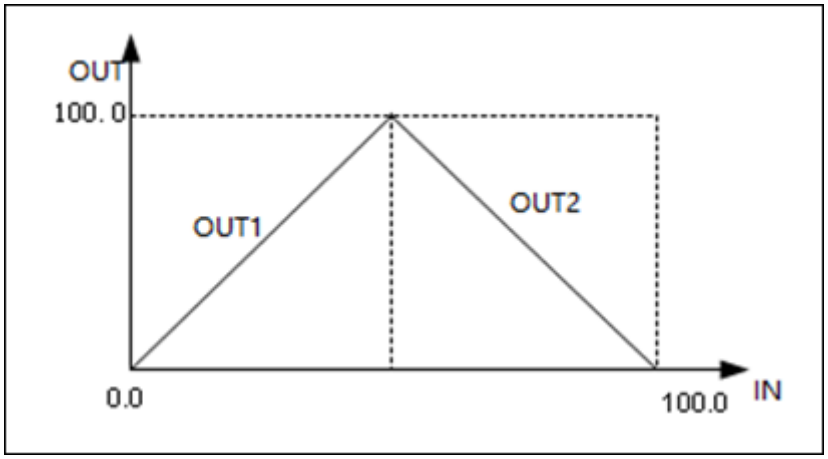
Direct-reverse split-range

- Out1OPT =false, OUT1_HLM =100, OUT1_LLM =0, S1_HLM =50, S1_LLM =0

  Direct acting: OUT1 = (IN - S1_LLM) (OUT1_HLM - OUT1_LLM) / (S1_HLM - S1_LLM) + OUT1_LLM = (IN - 50) 100 / 50 + 0

- Out2OPT = true, OUT2_HLM = 100, OUT2_LLM = 0, S2_HLM = 100, S2_LLM = 50

  Reverse acting: OUT2 = (S2_HLM - IN) (OUT2_HLM - OUT2_LLM) / (S2_HLM - S2_LLM) + OUT2_LLM = (100 - IN)100 / 50 + 0



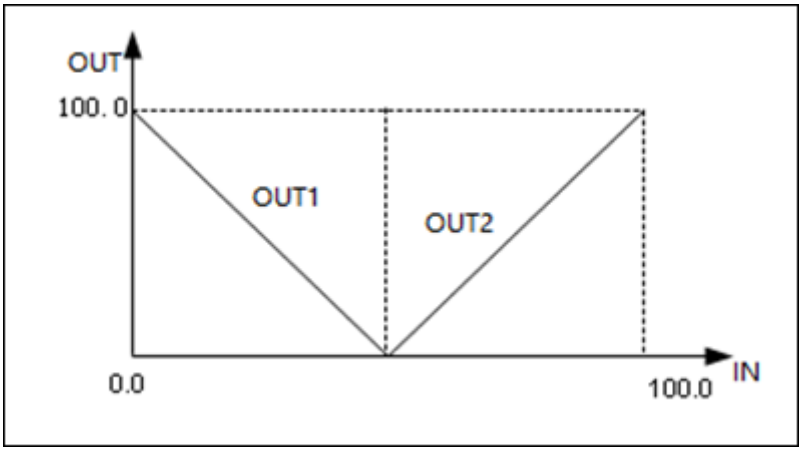
Reverse-direct split-range

- Out1OPT = true, OUT1_HLM = 100, OUT1_LLM = 0, S1_HLM = 50, S1_LLM = 0

  Reverse acting: OUT1 = (S1_HLM – IN) (OUT1_HLM – OUT1_LLM) / (S1_HLM – S1_LLM) + OUT1_LLM = (50 - IN) 100 / 50 + 0

- Out2OPT = false, OUT2_HLM = 100, OUT2_LLM = 0, S2_HLM = 100, S2_LLM = 50

  Direct acting: OUT2 = (IN - S2_LLM) (OUT2_HLM – OUT2_LLM) / (S2_HLM – S2_LLM) + OUT2_LLM= (IN - 50) 100 / 50 + 0



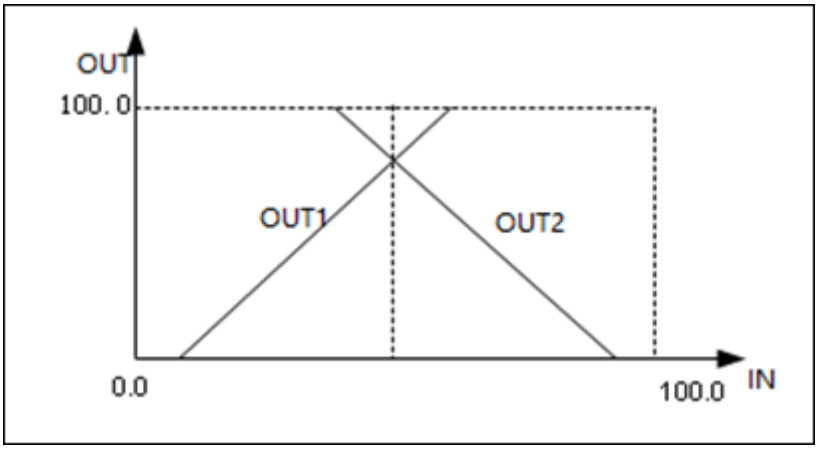
Cross direct-reverse split-range

- Out1OPT = false, OUT1_HLM = 100, OUT1_LLM = 0, S1_HLM = 60, S1_LLM = 10

  Direct acting: OUT1 = (IN - S1_LLM) (OUT1_HLM – OUT1_LLM) / (S1_HLM – S1_LLM) + OUT1_LLM = (IN - 10) 100 / 50 + 0

- Out2OPT = true, OUT2_HLM = 100, OUT2_LLM = 0, S2_HLM = 90, S2_LLM = 40

  Reverse acting: OUT2= (S2_HLM – IN) (OUT2_HLM – OUT2_LLM) / (S2_HLM – S2_LLM) + OUT2_LLM = (90 - IN) 100 / 50 + 0



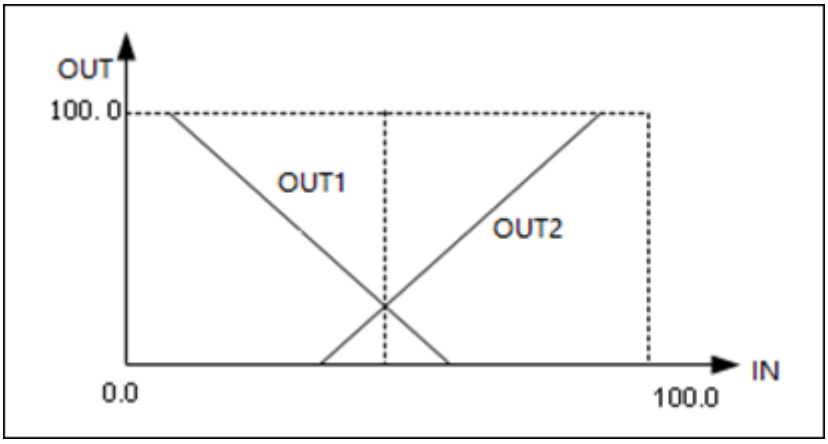
Cross reverse-direct split-range

- Out1OPT = true, OUT1_HLM =100, OUT1_LLM =0, S1_HLM =60, S1_LLM =10

  Reverse acting: OUT1= (S1_HLM - IN) (OUT1_HLM - OUT1_LLM) / (S1_HLM - S1_LLM) + OUT1_LLM = (60 - IN) 100 / 50 + 0

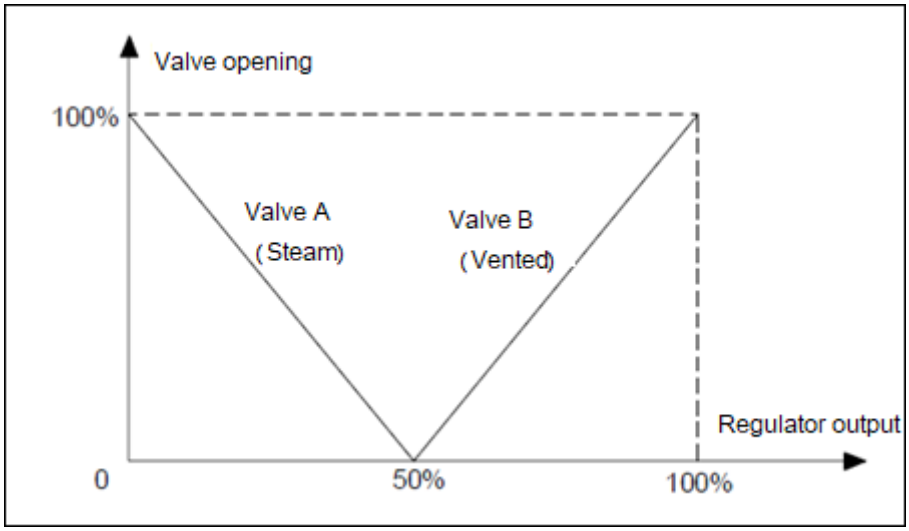
- Out2OPT = false, OUT2_HLM = 100, OUT2_LLM = 0, S2_HLM = 90, S2_LLM = 40

  Direct acting: OUT2 = (IN - S2_LLM) (OUT2_HLM - OUT2_LLM) / (S2_HLM - S2_LLM) + OUT2_LLM = (IN - 40) 100 / 50 + 0

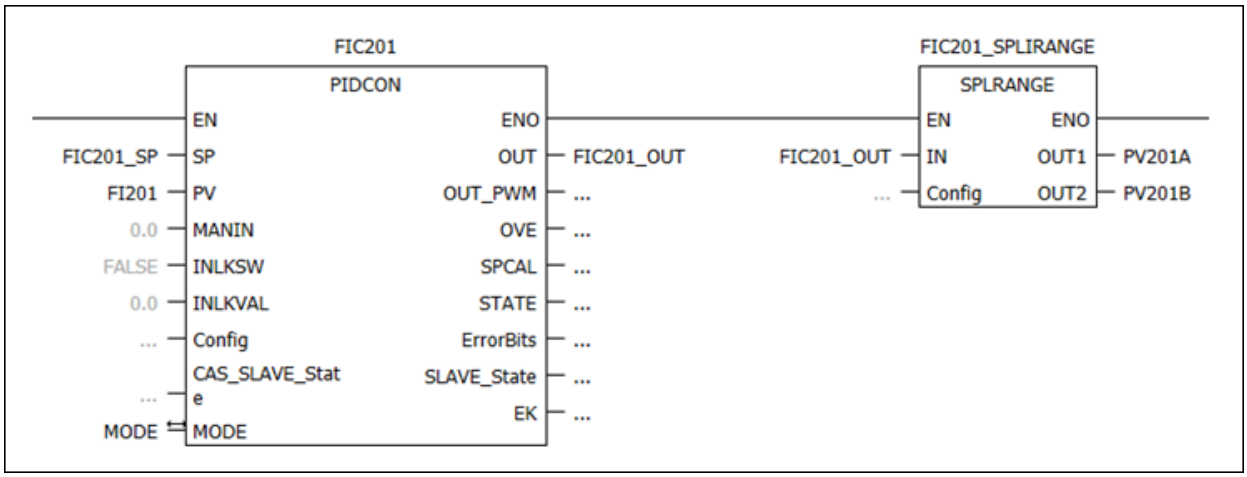


- **Application instance**

  A split-range control scheme is required for the pressure of a steam header. The control object is header pressure, and the actuators are two control valves: steam inlet valve (Valve A) and vent valve (Valve B). The action curve is shown in the figure below. When the output of the PID regulator ranges from 0% to 50%, Valve A moves from fully open to fully closed while Valve B remains fully closed. When the output ranges from 50% to 100%, Valve A stays fully closed, and Valve B opens gradually from fully closed to fully open.



Control scheme configuration:



The split-range configuration is shown in the figure below.

Config	SPLRANGE_CONFIG			☐	☐	Private
Out1OPT	BOOL		FALSE	☐	☐	Private
Out2OPT	BOOL		FALSE	☐	☐	Private
S1_HLM	REAL		50.0	☐	☐	Private
S1_LLM	REAL		0.0	☐	☐	Private
OUT1_HLM	REAL		100.0	☐	☐	Private
OUT1_LLM	REAL		0.0	☐	☐	Private
S2_HLM	REAL		100.0	☐	☐	Private
S2_LLM	REAL		50.0	☐	☐	Private
OUT2_HLM	REAL		100.0	☐	☐	Private
OUT2_LLM	REAL		0.0	☐	☐	Private

Example:

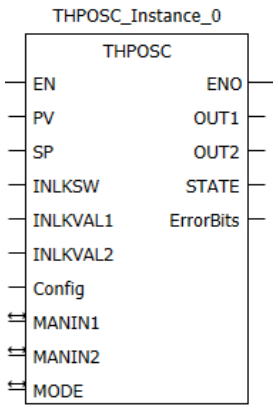
No.	Instance	Type
1	FIC201	Tag of PID function block
2	FI201	AI input
3	FIC201_SPLIRANGE	Tag of split-range control function block
4	PV201A	Split-range output 1

No.	Instance	Type
5	PV201B	Split-range output 2

18.3.1.7.14 THPOSC

This is the three-position control instruction.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
THPOSC	Three-position control.	FB		THPOSC_Instance_0(PV := , SP := , INLKSW := , MANIN1 := , MANIN2 := , INLKVAL1 := , INLKVAL2 := , Config := , MODE := , OUT1 => , OUT2 => , STATE => , ErrorBits =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Data type	Initial Value	Description
Input	PV	Process variable	REAL	0.0	Process variable.
	SP	Setpoint	REAL	0.0	Setpoint.
	MANIN1	Manual mode input 1	BOOL	FALSE	Manual mode input 1
	MANIN2	Manual mode input 2	BOOL	FALSE	Manual mode input 2
	INLKSW	Interlock switch	BOOL	FALSE	FALSE: Off. TRUE: On.

Input/Output	Variable	Name	Data type	Initial Value	Description
	INLKVAL1	Output 1 safety interlock value	BOOL	FALSE	Output 1 safety interlock value.
	INLKVAL2	Output 2 safety interlock value	BOOL	FALSE	Output 2 safety interlock value.
	Config	User internal configuration value	THPOSC_ Config	-	User internal configuration value.

Input

Parameter description

Variable	Name	Data type	Initial Value	Description
PV_HLM	Process variable upper limit	REAL	100	Process variable upper limit.
PV_LLM	Process variable lower limit	REAL	0	Process variable lower limit.
EKLIM	Deviation alarm limit	REAL	100	Deviation from setpoint > 0
EKDB	Deviation deadzone limit	REAL	0.0	Deadzone for deviation alarm reset ≥ 0
DeadZone	Deadzone control	REAL	0.0	The deadzone differs from that in two-position control. It functions as a control interval for logic control. For details, see the mode description below.
ACTSEL	Action mode	BOOL	TRUE	FALSE: Direct action. TRUE: Reverse action.
SP_RAT	SP rate limit value	REAL	100.0	The rate limit value of SP change.

Input/Output	Variable	Name	Data type	Initial Value	Description
		<table><tr><th>Variable</th><th>Name</th><th>Data type</th><th>Initial Value</th><th>Description</th></tr><tr><td>SPSEL</td><td>Setpoint source selection</td><td>BOOL</td><td>FALSE</td><td>FALSE: External setting (sp configurati on). TRUE: Internal setpoint (panel operation setpoint).</td></tr><tr><td>SPINT</td><td>Internal setpoint</td><td>REAL</td><td>0.0</td><td>Internal setpoint.</td></tr><tr><td>SPTRK</td><td>Internal setpoint tracking switch</td><td>BOOL</td><td>TRUE</td><td>0: No tracking. 1: Tracking.</td></tr></table>			
Input/Output	MANIN1	Controller 1 manual input	BOOL	-	Controller 1 manual input
	MANIN2	Controller 2 manual input	BOOL	-	Controller 2 manual input
	MODE	Operation mode	UINT	0	0: Manual. 1: Automatic.
Output	Errorbits	Fault alarm value	DWORD	0	Bit0: Process value overlimit. Bit1: Deviation overlimit. Bit2: Parameter setting error. Bit3: Setpoint change rate overlimit in automatic mode.
	OUT1	Controller output value 1	BOOL	FALSE	TRUE: On. FALSE: Off.

Input/Output	Variable	Name	Data type	Initial Value	Description
	OUT2	Controller output value 2	BOOL	FALSE	TRUE: On. FALSE: Off.
	STATE	Current operating status	UINT	0	0: Initial. 1: Manual. 2: Automatic. 3: Interlocked.

- **Function**

  Three-step control is a developed to overcome the contradiction between the heating rate and temperature overshoot that often occurs with two-step control. It has one additional output switch compared to a two-step controller. For example, when used to control an electric furnace, it can control two groups of heaters. When the furnace temperature is below the lower setpoint, both output switches are activated, and both groups of heaters heat simultaneously, causing the furnace temperature to rise quickly. When the furnace temperature is between the upper and lower setpoints, only one output switch is activated, and one group of heaters heats, causing the temperature to rise more slowly. When the furnace temperature exceeds the upper setpoint, both switches are disconnected, stopping the heating, and the furnace temperature decreases until it falls below the lower setpoint, at which point both groups of heaters are reactivated.

  Three-step control heats faster than two-step control, and after entering the constant temperature regulation status, the temperature fluctuation is smaller, with higher precision. A three-step controller provides better control performance than a two-step controller.

  - **Operation mode**

Operation Mode	Action
Automatic mode	The system will automatically output switch signals based on the comparison between the user-set setpoint and the process value of the controlled object.
Manual mode	In this mode, the system outputs manual values.
Interlock mode	In this mode, the system outputs interlock safety values.

  Priority order: automatic mode < manual mode < interlock mode

  - **Action mode**

  There are two action modes: reverse action, where if the measured value PV is less than the setpoint SP, TWPOSC outputs positively (ON), and if PV is greater than SP, TWPOSC outputs negatively (OFF); direct action is the opposite, where if the measured value PV is less than the setpoint SP, TWPOSC outputs negatively (OFF), and if PV is greater than SP, TWPOSC outputs positively (ON).

  Compared to two-step control, three-step control has two output control signals, and the automatic mode works as follows:

Conditions	Output
PV > setpoint + control deadzone (SP + DeadZone)	Both OUT1 and OUT2 outputs are "OFF"
(SP - DeadZone) setpoint - control deadzone ≤ PV ≤ setpoint + control deadzone (SP + DeadZone)	OUT1 output is "ON" OUT2 output is "OFF"
PV < setpoint - control deadzone (SP - DeadZone)	Both OUT1 and OUT2 outputs are "ON"

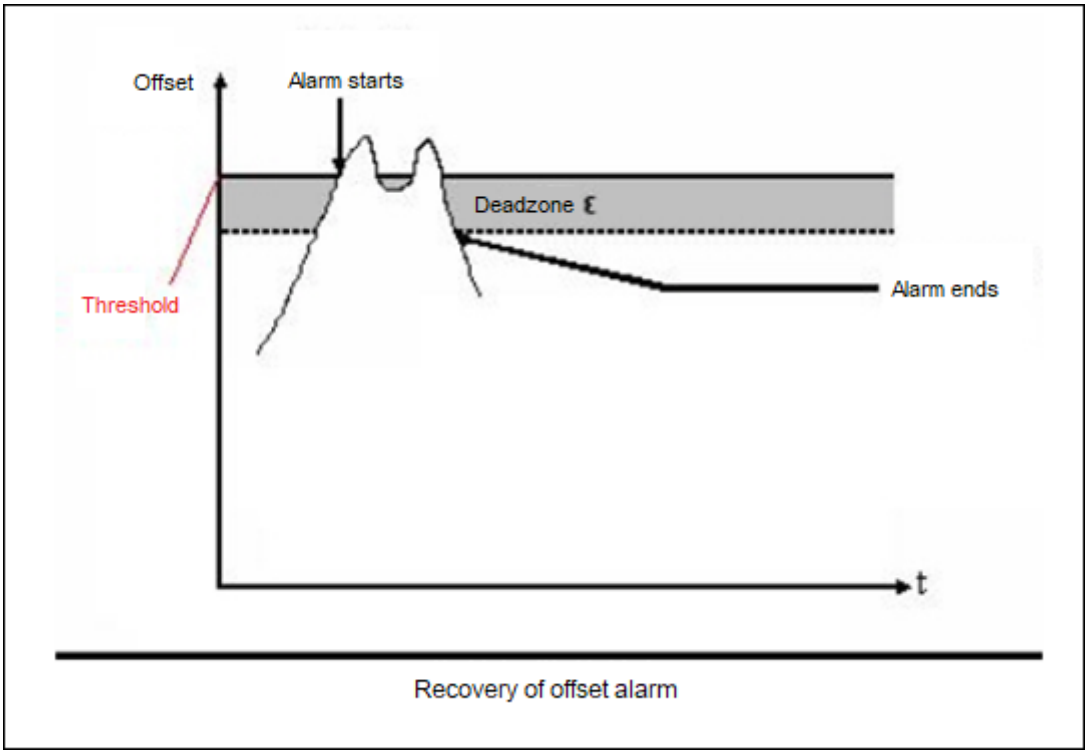
- **Functional logic diagram (reverse-acting example)**

- **Deviation alarm (in automatic mode)**

  A deviation alarm refers to a situation in automatic control mode where the difference between the measured value and the setpoint (usually the absolute value) exceeds the preset deviation alarm limit, triggering a deviation alarm and switching the operating mode from automatic to manual.

- **Deviation alarm recovery**

  The deviation alarm recovery involves the concept of a deadzone. When the deviation value falls below the set threshold, the deviation alarm does not immediately disappear. This is because if it did, and the deviation value quickly rose above the threshold again, or fluctuated around the threshold, it would cause the system to repeatedly issue and end deviation alarms, leading to ping-pong alarms. Therefore, a deadzone is used to address this issue. If the deviation value falls below the set threshold, the deviation alarm does not immediately disappear. The alarm only disappears when the deviation value continues to decrease and falls outside the deadzone range, as shown in the figure below.



- **Deadzone control**

  It functions as a control interval, which affects the control mode of automatic mode. When the measured value PV fluctuates within the set range around the setpoint SP [SP-DeadZone, SP+DeadZone], THPOSC output value OUT1 is TRUE, and OUT2 is FALSE.

- **Bumpless switchover**

Bumpless switchover: When the module's operation mode changes, the module's output remains unchanged during the switching. After the switching is done, the module's output begins to calculate based on the previous output value, ensuring that there is no disturbance to the control of on-site equipment.

There are six operation modes for the THPOSC control module. Mode switching is generally bumpless, but disturbances may occur under certain conditions, as detailed below.

Automatic -> Manual Interlock -> Manual	Manual -> Interlock Automatic -> Interlock	Manual -> Automatic Interlock -> Automatic
In automatic or interlock mode, the internal manual value automatically tracks the current mode output value to ensure a bumpless switchover to manual.	In interlock mode, the interlock safety value is directly output, and the switch may cause disturbances.	In automatic mode, the input value is directly associated, and the switch may cause a disturbance.

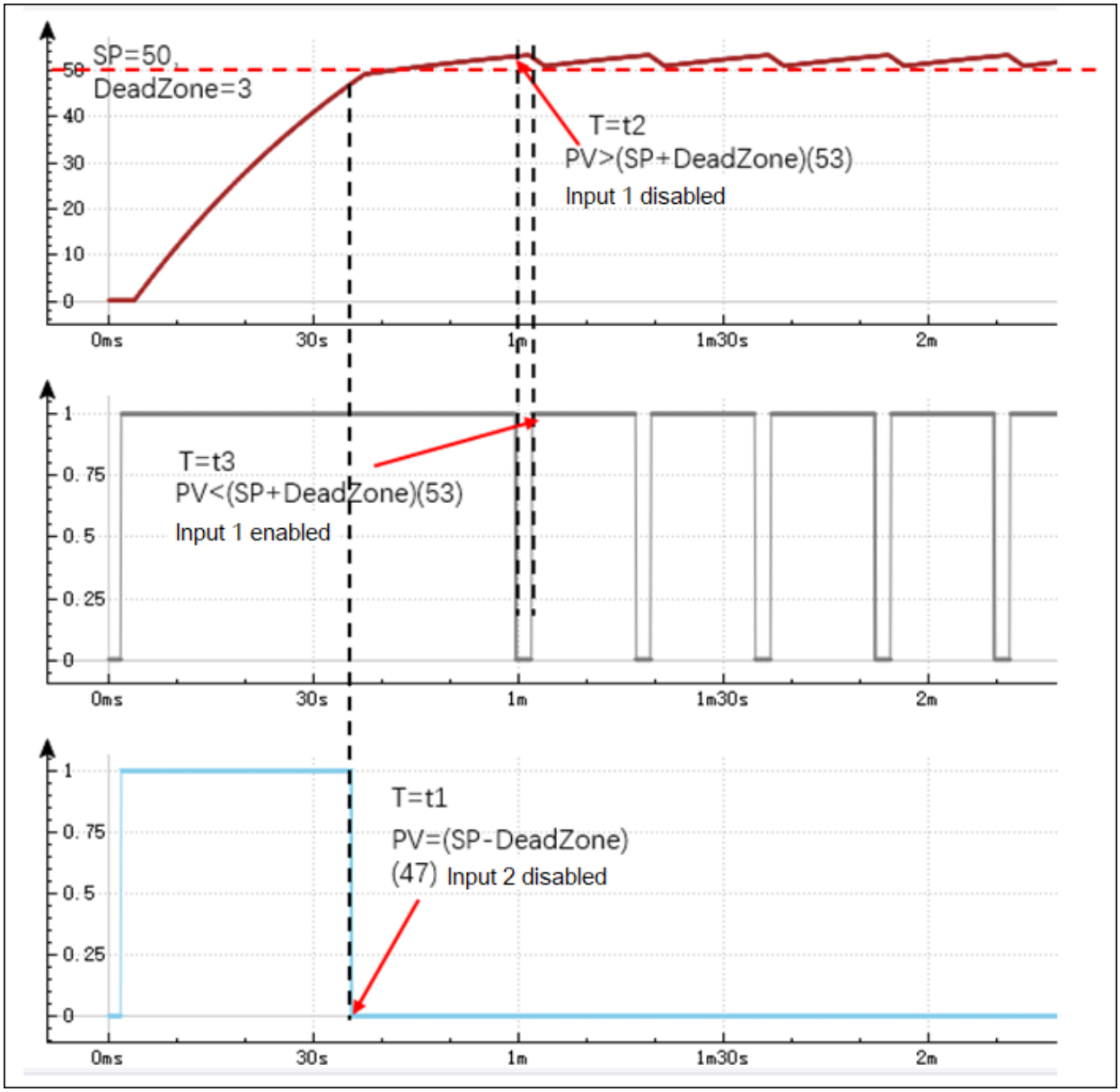
Errorbits description

Errorbits	Note	Solution
16#0000	No error	/
16#0001	PV upper/lower limit alarm There are two possibilities: • PV > PV_HLM (Process value greater than upper limit). • PV < PV_LLM (Process value less than lower limit).	Output: Switch to manual mode, output current manual value (interlock mode does not switch) After checking and confirming, the user can switch to automatic mode to resume operation.
16#0002	Deviation overlimit alarm In automatic mode, the absolute value of the deviation between process value PV and setpoint SP is greater than the deviation alarm value.	Output: Switch to manual mode, output current manual value After checking and confirming, the user switches to automatic to continue running In automatic mode, if an alarm is triggered and switched to manual, the alarm remains until it is confirmed on-site, then switch back to automatic mode for deviation alarm recovery.

Errorbits	Note	Solution
16#0004	Parameter setting error alarm Includes: In all modes, the PV upper/lower limit setting is invalid if PV_LLM ≥ PV_HLM. In automatic mode: • EKDB >= EKLIM (Deviation deadzone greater than or equal to deviation alarm value) • DeadZone >= EKLIM (Control deadzone greater than or equal to deviation alarm value) • DeadZone >= EKLIM - EKDB (Deviation alarm reset within control deadzone) • Control deadzone (DeadZone) or deviation deadzone (EKDB) is less than 0, or the deviation alarm limit (EKLIM) is less than or equal to 0. • Setpoint rate limit SP_RAT≤0	Output: If parameters are invalid, they do not take effect and the output holds. • Reset the upper and lower limits of PV to ensure PV_HLM > PV_LLM • Reset the deviation alarm value (EKLIM), deviation deadzone (EKDB), and control deadzone (DeadZone). Ensure the following conditions are met: EKDB < EKLIM DeadZone < EKLIM; DeadZone < EKLIM - EKDB DeadZone, EKDB ≥ 0 EKLIM > 0 • Reset the setpoint rate limit (SP_RAT) to ensure that SP_RAT > 0.
16#0008	Setpoint overspeed alarm. In automatic mode, the setpoint change exceeds the rate limit (SP_RAT).	Output: Perform normal calculation based on the setpoint after rate limiting. The alarm automatically resets after the setpoint change is less than SP_RAT.

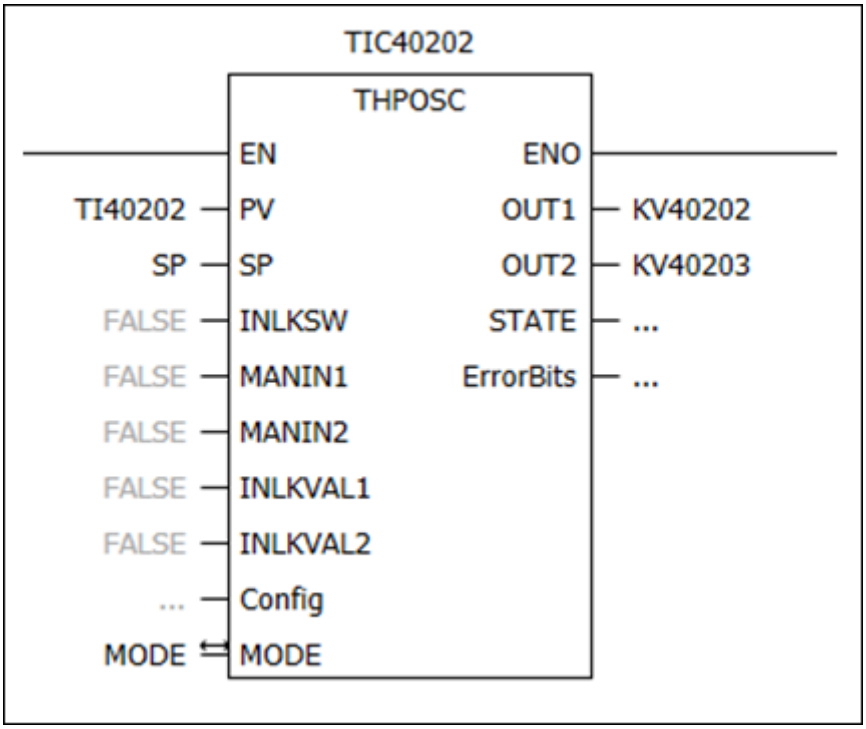
- **Application instance**

  For example, in a three-step temperature control system for an electric heating furnace, three-step regulation can be achieved using two relays to form three output statuses: heating up, temperature regulation, and stop heating. The control process is as follows:



1. Two groups of heaters, auxiliary heater A and main heater B, are adopted. When the measured value is below SP-DeadZone from time 0 to t1, both upper and lower limit relays are energized, and the system enters the "heating up" state. At this time, both heaters A and B heat simultaneously, so the heating rate is faster.
2. When the measured value reaches SP-DeadZone but is still below SP+DeadZone from time t1 to t2, the lower limit relay releases with the power supply to auxiliary heater A cutting off, and the heating rate decreases. The system enters the "temperature regulation" state.
3. When the measured value reaches SP+DeadZone from time t2 to t3, the lower limit relay remains open, and the upper limit relay begins to release, cutting off the power supply to main heater B. Since both the main and auxiliary heaters lose their power supply, the temperature gradually decreases until it falls within SP+DeadZone. From time t3 to t4, the upper limit relay re-energizes to reconnect the power supply to main heater B, and the temperature gradually rises again. The process repeats in cycles. Therefore, three-step regulation ensures low temperature fluctuation and high precision in constant-temperature mode.

Control scheme configuration:



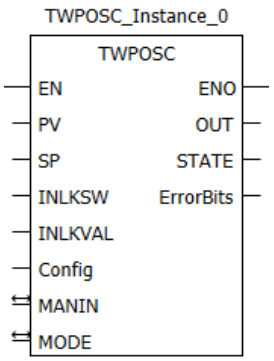
Example:

No.	Instance	Type
1	TIC40202	Tag of three-position control function block
2	KV40202	Heater B control valve
3	KV40202	Heater A control valve
3	TI40201	Temperature collection input

18.3.1.7.15 TWPOSC

This is the two-position control instruction.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
TWPOSC	Two-position control	FB		TWPOSC_Instance_0(PV := , SP := , MANIN := , INLKSW := , INLKVAL := , Config := , MODE := , OUT => , STATE => , ErrorBits =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Data type	Initial Value	Description
Input	PV	Process variable	REAL	0.0	Process variable.
	SP	Setpoint	REAL	0.0	Setpoint.
	MANIN	Manual mode input	BOOL	FALSE	Manual mode input
	INLKSW	Interlock switch	BOOL	FALSE	FALSE: Off. TRUE: On.
	INLKVAL	Interlock safety value	BOOL	FALSE	Interlock safety value.
	Config	Parameter configuration	TWPOSC_ Config	-	Parameter configuration

<table>
<tr><th>Input/ Output</th><th>Variable</th><th>Name</th><th>Data type</th><th>Initial Value</th><th>Description</th></tr>
<tr><td>Input</td><td>Parameter description</td><td colspan="4">
<table>
<tr><th>Variable</th><th>Name</th><th>Data type</th><th>Initial Value</th><th>Description</th></tr>
<tr><td>PV_HLM</td><td>Process variable upper limit</td><td>REAL</td><td>100</td><td>Process variable upper limit.</td></tr>
<tr><td>PV_LLM</td><td>Process variable lower limit</td><td>REAL</td><td>0.0</td><td>Process variable lower limit.</td></tr>
<tr><td>EKLIM</td><td>Deviation alarm limit</td><td>REAL</td><td>100.0</td><td>Deviation from setpoint > 0</td></tr>
<tr><td>EKDB</td><td>Deviation deadzone limit</td><td>REAL</td><td>0.0</td><td>Deadzone for deviation alarm reset ≥ 0</td></tr>
<tr><td>DeadZone</td><td>Deadzone control</td><td>REAL</td><td>0.0</td><td>Deviation within control deadzone for stable regulation ≥ 0</td></tr>
<tr><td>ACTSEL</td><td>Action mode</td><td>BOOL</td><td>TRUE</td><td>FALSE: Direct action.
TRUE: Reverse action.</td></tr>
<tr><td>SP_RAT</td><td>SP rate limit value</td><td>REAL</td><td>100</td><td>The rate limit value of SP change.</td></tr>
<tr><td>SPSEL</td><td>Setpoint source selection</td><td>BOOL</td><td>FALSE</td><td>FALSE: External setpoint (SP configuration).
TRUE: Internal setpoint (panel operation setpoint).</td></tr>
<tr><td>SPINT</td><td>Internal setpoint</td><td>REAL</td><td>0.0</td><td>Internal setpoint.</td></tr>
<tr><td>SPTRK</td><td>Internal setpoint tracking switch</td><td>BOOL</td><td>TRUE</td><td>0: No tracking.
1: Tracking.</td></tr>
</table>
</td></tr>
</table>

Input/ Output	Variable	Name	Data type	Initial Value	Description
Input/ Output	MANIN	Manual mode	BOOL	-	Manual mode input
	MODE	Operation mode	UINT	-	0: Manual. 1: Automatic.
Output	OUT	Output value	BOOL	FALSE	TRUE: On. FALSE: Off.
	STATE	Current operating status	UINT	0	0: Initial. 1: Manual. 2: Automatic. 3: Interlocked.
	Errorbits	Fault alarm value	DWORD	0	Fault alarm value

- **Function**

  Position control refers to an analog controller that outputs intermittent signals to drive actuators. Its output is typically on/off switching signals for power supply, so it is also known as on-off controller. The simplest position controller is two-position controller.

  Two-position control adopts two logic states (ON and OFF) of a single digital output to complete control of an analog, so that the controlled parameter is stabilized within a control range. That is, the controlled parameter is an analog. When it exceeds the setpoint, the controller outputs an OFF signal (the action can also be reversed); when it drops below the setpoint, the controller outputs an ON signal (the action can also be reversed). Repeated two-position regulation keeps the controlled parameter close to or at the setpoint. Two-position regulators are applicable to control systems with low precision requirements, such as thermostats, electric heating furnaces, air conditioners, air pumps and household appliances.

  - **Operation Mode**

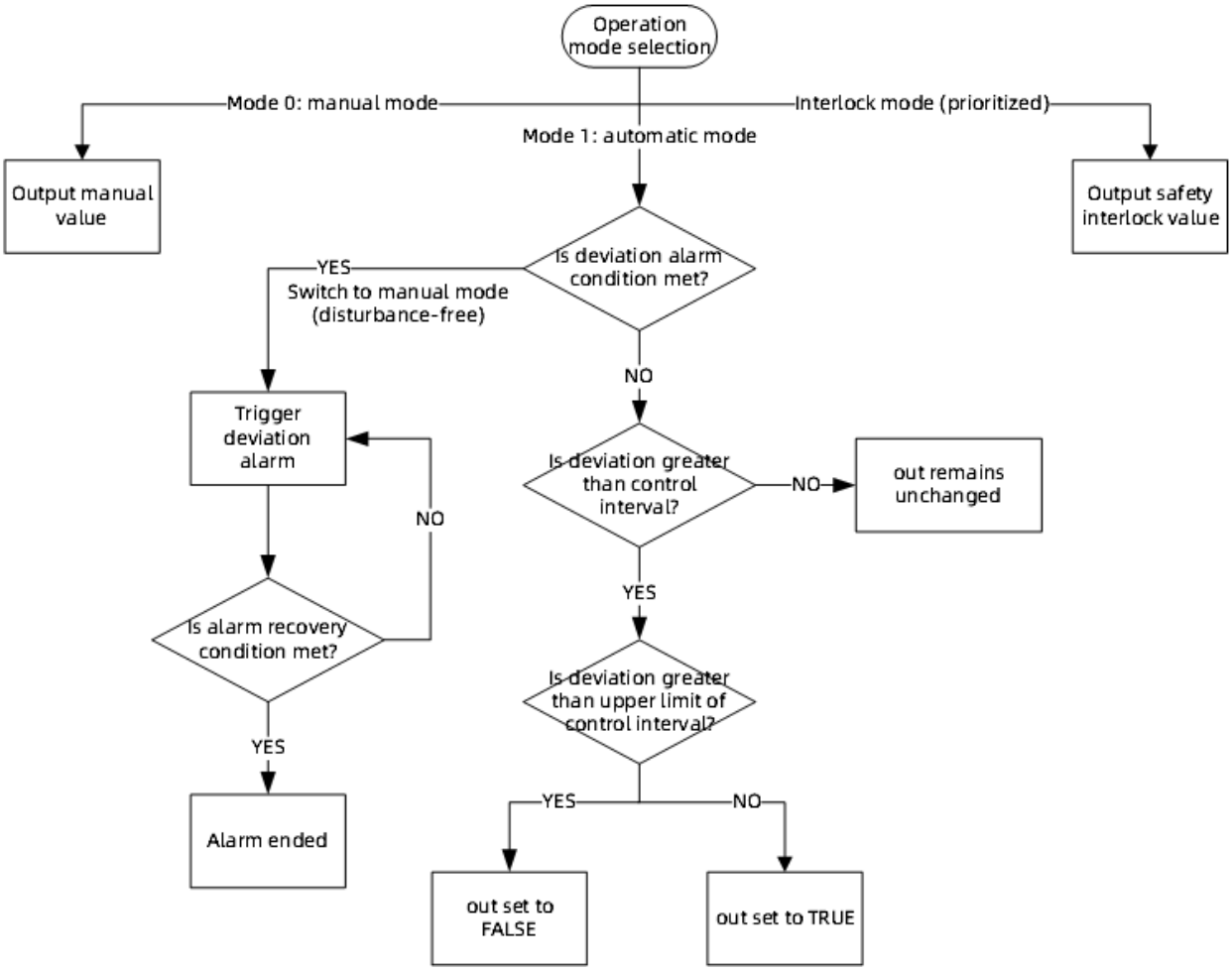
Operation Mode	Action
Automatic mode	The system will automatically output switch signals based on the comparison between the user-set setpoint and the process value of the controlled object.
Manual mode	In this mode, the system outputs manual values.
Interlock mode	In this mode, the system outputs the interlock safety value. It switches to manual mode by default when interlock is released.

  Priority order: automatic mode < manual mode < interlock mode

  - **Action mode**

There are two modes of action: reverse acting, where TWPOSC outputs positive (ON) when the process value (PV) is less than the setpoint (SP) and outputs negative (OFF) when PV is greater than SP; and direct acting, which is the opposite, where TWPOSC outputs negative (OFF) when PV is less than SP and outputs positive (ON) when PV is greater than SP.

- **Functional logic diagram (reverse-acting example)**

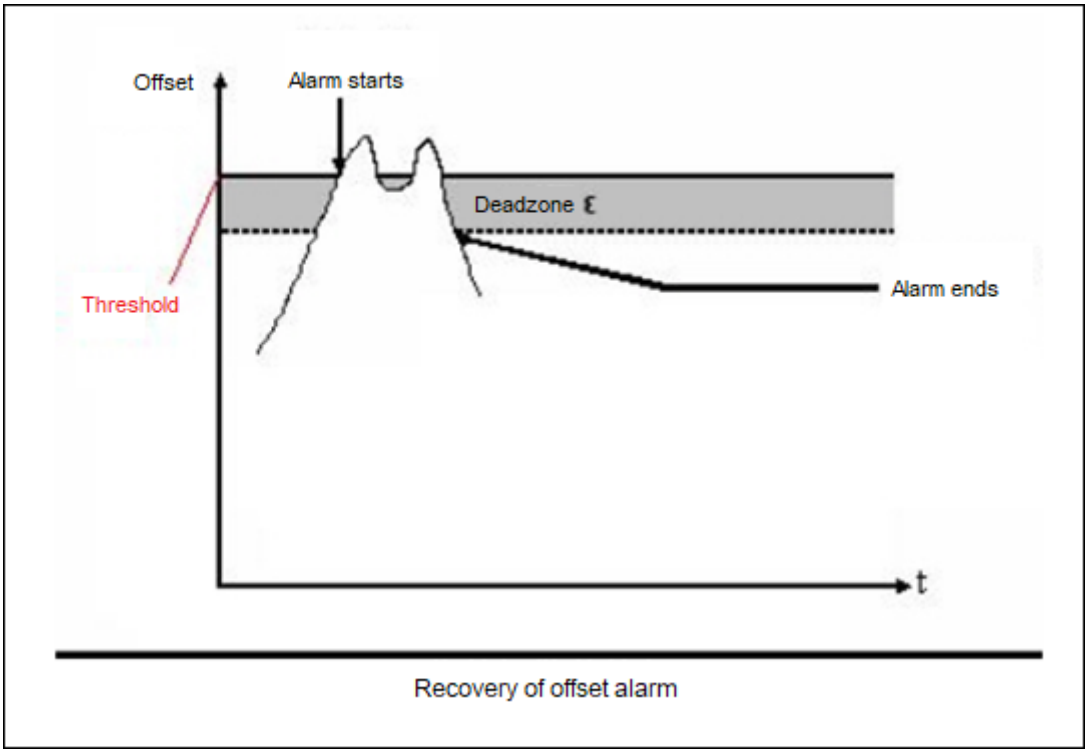


- **Deviation alarm (in automatic mode)**

  A deviation alarm refers to a situation in automatic control mode where the difference between the measured value and the setpoint (usually the absolute value) exceeds the preset deviation alarm limit, triggering a deviation alarm and switching the operating mode from automatic to manual.

- **Deviation alarm recovery**

  The deviation alarm recovery involves the concept of a deviation deadzone. When the deviation value falls below the set threshold, the deviation alarm does not immediately disappear. This is because if it did, and the deviation value quickly rose above the threshold again, or fluctuated around the threshold, it would cause the system to repeatedly issue and end deviation alarms, leading to ping-pong alarms. Therefore, a deadzone is used to address this issue. If the deviation value falls below the set threshold, the deviation alarm does not immediately disappear. The alarm only disappears when the deviation value continues to decrease and falls outside the deadzone range, as shown in the figure below.



- **Deadzone control**

  During the automatic control process, when the measured value fluctuates within a very small range around the setpoint, the output signal may frequently switch between ON and OFF. This is detrimental to on-site equipment and is ineffective. Therefore, an automatic control deadzone (DeadZone) is introduced. If the measured value PV fluctuates within the set range [SP-DeadZone, SP+DeadZone], TWPOSC considers the system to be stable and does not change the output.

- **Bumpless switchover**

  Bumpless switchover: When the module's operation mode changes, the module's output remains unchanged during the switching. After the switching is completed, the module's output begins to calculate based on the previous output value, ensuring that there is no disturbance to the control of on-site equipment.

  There are six operation modes for the TWPOSC control module. Mode switching is generally bumpless, but disturbances may occur under certain conditions, as detailed below.

Automatic -> Manual Interlock -> Manual	Manual -> Interlock Automatic -> Interlock	Manual -> Automatic Interlock -> Automatic
In automatic or interlock mode, the internal manual value automatically tracks the current mode output value to ensure a bumpless switchover to manual.	In interlock mode, the interlock safety value is directly output, and the switch may cause disturbances.	In automatic mode, the input value is directly associated, and the switch may cause disturbances.

- **Corresponding faults to Errorbits**

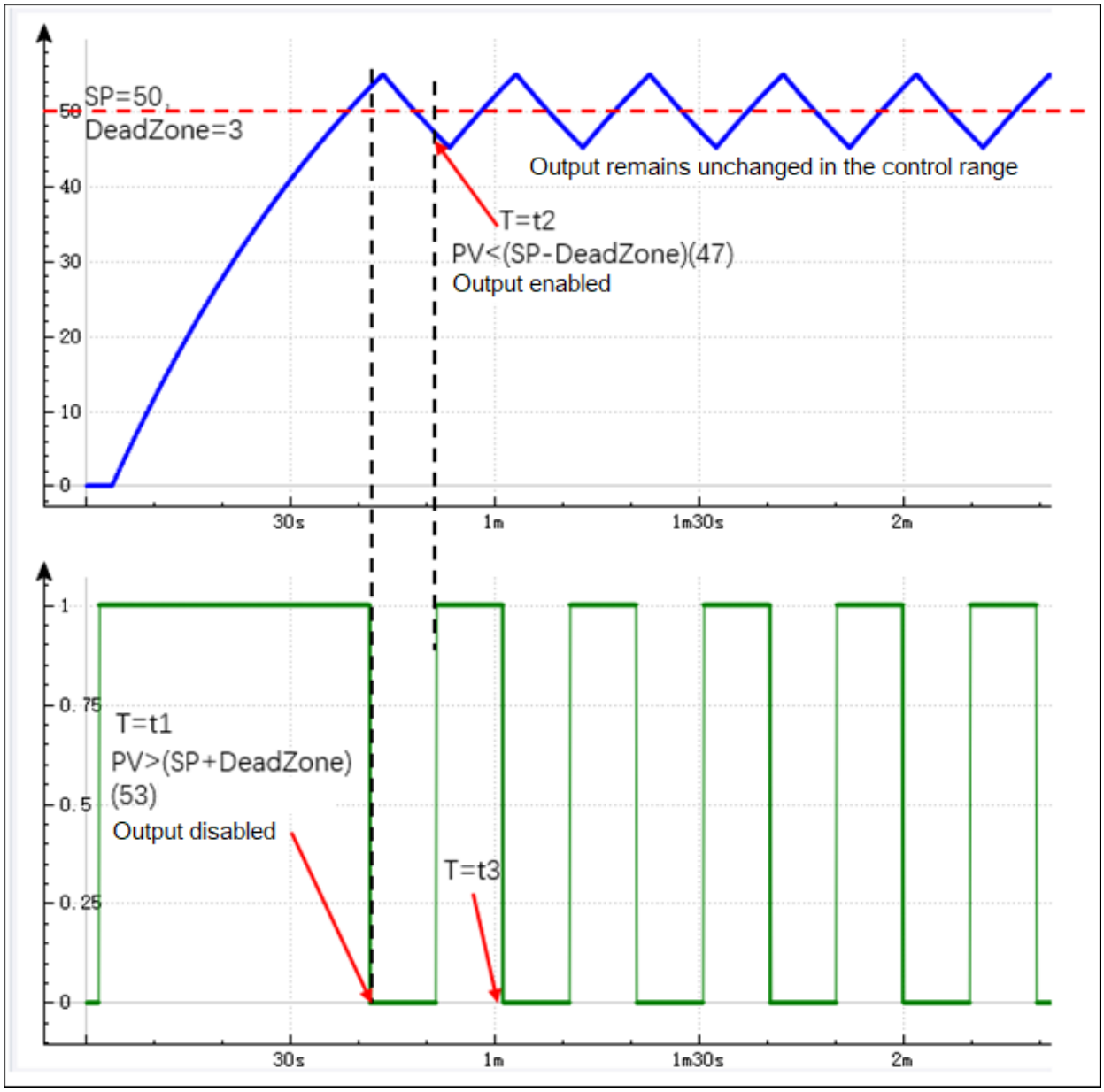
Errorbits	Note	Solution
16#0000	No error	/

Errorbits	Note	Solution
16#0001	PV upper/lower limit alarm There are two possibilities: • PV > PV_HLM (Process value greater than upper limit) • PV < PV_LLM (Process value less than lower limit)	Output: Switch to manual mode, output current manual value (interlock mode does not switch) After checking and confirming, the user can switch to automatic mode to resume operation.
16#0002	Deviation limit alarm In automatic mode, the absolute value of the deviation between process value PV and setpoint SP is greater than the deviation alarm value.	Output: Switch to manual mode, output current manual value After checking and confirming, the user switches to automatic to continue running In automatic mode, if an alarm is triggered and switched to manual, the alarm remains until it is confirmed on-site, then switch back to automatic mode for deviation alarm recovery.
16#0004	Parameter setting error alarm Includes: In all modes, the PV upper/lower limit setting is invalid if PV_LLM ≥ PV_HLM. In automatic mode: • EKDB >= EKLIM (Deviation deadzone greater than or equal to deviation alarm value) • DeadZone >= EKLIM (Control deadzone greater than or equal to deviation alarm value) • DeadZone >= EKLIM - EKDB (Deviation alarm reset within control deadzone) • Control deadzone (DeadZone) or deviation deadzone (EKDB) is less than 0, or the deviation alarm limit (EKLIM) is less than or equal to 0. • Setpoint rate limit (SP_RAT) ≤ 0	Output: If parameters are invalid, they do not take effect and the output holds. • Reset the upper and lower limits of PV to ensure PV_HLM > PV_LLM. • Reset the deviation alarm value (EKLIM), deviation deadzone (EKDB), and control deadzone (DeadZone). Ensure the following conditions are met: EKDB < EKLIM DeadZone < EKLIM DeadZone < EKLIM - EKDB DeadZone, EKDB ≥ 0 EKLIM > 0 • Reset the setpoint rate limit (SP_RAT) to ensure that SP_RAT > 0.

Errorbits	Note	Solution
16#0008	Setpoint overrate alarm In automatic mode, the setpoint change exceeds the rate limit (SP_RAT).	Output: Perform normal calculation based on the setpoint after rate limiting. The alarm automatically resets after the setpoint change is less than SP_RAT.

- **Application instance**

  Take the on-off temperature control system of an electric heating furnace as an example, the control process is shown in the figure below.

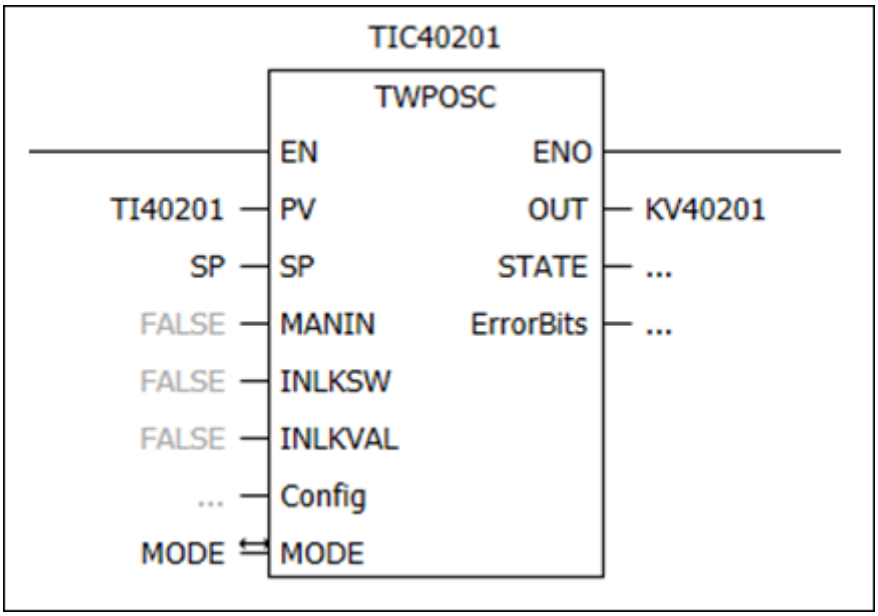


- At time 0-t1, since PV is less than SP + DeadZone, the heater remains powered and the temperature continues to rise. At time t = t1, since PV = SP + DeadZone, the power is turned off and heating stops, but the temperature does not immediately drop. On the contrary, it continues to rise for a period of time τ1, and only starts to gradually decrease from T' at t = t1 + τ1.

- At t = t2, since PV = SP - DeadZone, the heater power is turned on to start heating, but the temperature does not immediately rise. Instead, it continues to drop for a period of time τ2, and only begins to gradually increase from T'x at t = t2 + τ2.
- At t = t3, T equals SP + DeadZone again, and the temperature controller turns off the power to stop heating. The system and object then repeat the above process once again, and this cycle continues indefinitely. Since the control action alternates between two states, "on" and "off," the temperature of the controlled heating furnace will periodically fluctuate.

With on-off control, the deadzone (DeadZone) must be set appropriately based on the specific control object and the allowable fluctuation range. This method is mainly used for heating objects with small delay and large time constant.

Control scheme configuration:



Example:

No.	Instance	Type
1	TIC40201	Tag of two-position control function block
2	KV40201	Temperature control valve
3	TI40201	Temperature collection input

18.3.2 Advanced Process Control

18.3.2.1 Instruction List

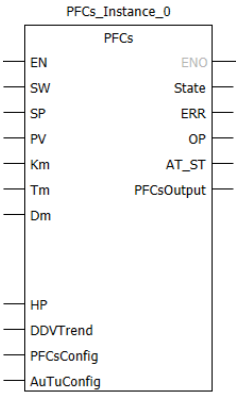
Instruction	FB/FC	Function	Library	Applicable Library Version
18.3.2.2 PFCs	FB	Predictive functional controller	PFC	Library version 1.0.0.0 and later
18.3.2.3 PFCm	FB	Multivariable advanced control		Library version 1.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.3.2.4 PFCs_dv	FB	Feedforward predictive control		Library version 1.0.0.0 and later
18.3.2.5 PFCs_v2	FB	Single-loop advanced control V2		Library version 1.1.0.0 and later
18.3.2.6 PFCm_v2	FB	Multivariable advanced control V2		Library version 1.1.0.0 and later
18.3.2.7 PFCs_dv_v2	FB	Feedforward predictive control V2		Library version 1.1.0.0 and later
18.3.2.8 PFCm_dv	FB	Feedforward predictive control M		Library version 1.1.0.0 and later
18.3.2.9 PIDR	FB	Robust PID		Library version 1.1.0.0 and later
18.3.2.10 PFCm22	FB	Multivariable input/ output advanced control		Library version 1.1.0.0 and later
18.3.2.11 PRBS	FB	Pseudo-Random Binary Sequence		Library version 1.1.0.0 and later
18.3.2.12 PFC11	FB	Single-loop advanced control		Library version 1.2.0.0 and later
18.3.2.13 PFC22	FB	Multivariable input/ output advanced control		Library version 1.2.0.0 and later
18.3.2.14 PFC_DV	FB	Feedforward predictive control		Library version 1.2.0.0 and later

18.3.2.2 PFCs

PFCs are simplified predictive function controller (PFC) components for single-input, single-output systems, primarily used in one-to-one control scenarios such as temperature heating control.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PFCs	Predictive functional controller	FB		PFCs_Instance_0(SW := , SP := , PV := , Km := , Tm := , Dm := , HP := , DDVTrend := , PFCsConfig := , AuTuConfig := , State => , ERR => , OP => , AT_ST => , PFCsOutput =>);

- **Related variables**

  Input variables

Variable	Name	Data Type	Valid Range	Default Value	Description
SW	Operating switch	DINT	0, 1	0	0: Component off 1: Component on Ensure the component is in the on state when in use.
SP	Setpoint	LREAL	Follows data type	0	Setpoint for the controlled variable, which defines the control target of the controller. If properly configured, the actual control result (PV) must closely follow the SP.
PV	Measured value	LREAL	Follows data type	0	Input value for the measured value of the controlled variable, which must be provided via an external physical object or simulation object.

Variable	Name	Data Type	Valid Range	Default Value	Description
Km	Model gain	LREAL	Follows data type	1	An algorithm parameter that describes the response gain of the controlled object. It must be determined based on the influence of the output OP on the controlled object. The greater this influence, the larger the gain value.
Tm	Model time constant	LREAL	(0,1.7E 307]	1	An algorithm parameter that describes the response speed of the controlled object. The greater the inertia of the controlled object, the slower the response, and the larger this value.
Dm	Model hysteresis	LREAL	(0,10000]	0	An algorithm parameter that represents the response hysteresis of the controlled object. The greater the hysteresis of the object to input changes, the larger this value.
HP	Prediction horizon marker	ARRAY[0..199] OF DINT	The array contains values of 0 or 1.	-	0: Invalid 1: The predicted value for the current cycle is used in the prediction calculation. The HP sequence adjusts control performance. Ensure valid points are present in HP after the hysteresis period. The longer the valid length of the HP sequence, the further into the future the trend prediction extends.
DDVTrend	Future disturbance sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Disturbance sequence generated by the PFCs_dv component. When using this variable, pass its "DDVTrend" output to this variable. It is used to predict future disturbances so that the controller can take proactive action to suppress them.

Variable	Name	Data Type	Valid Range	Default Value	Description
AuTuConfig	Auto-tuning parameter structure	AuTuConfig	Follows data type	-	Auto-tuning parameter configuration structure.
PFCsConfig	Input parameter structure	PFCsConfig	Follows data type	-	Input parameter configuration structure.

AuTuConfig structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
AT_Start	Auto-tuning switch	DINT	0, 1	0	0: Inactive 1: On Set to 1 if auto-tuning function is required.
AT_Model	Model type	DINT	0, 1, 2	0	Model type for the controlled object. 0: Non-integral 1: Pseudo-integral 2: Pure integral Assign values based on actual conditions.
AT_SW	Operating switch	DINT	0, 1, 2	1	0: Off 1: Coarse tuning 2: Fine tuning If the auto-tuning function is enabled, assign values according to the conditions.
AT_OPInit	Initial OP value for tuning	LREAL	Follows data type	0	Initial output value that ensures a stable PV measurement of the object being tuned at the start of the tuning process. The object must remain essentially stable under the AT_OPInit output before tuning begins.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
AT_OPTarget	Tuning output step value	LREAL	Follows data type	100	Target output. After tuning starts, the OP output steps from AT_OPInit to AT_OPTarget, and returns to AT_OPInit when tuning is complete.
AT_PVTarget	PV value at tuning end.	LREAL	Follows data type	100	PV value at tuning end. After tuning starts, the PV of the controlled object will change from an initial steady state to AT_PVTarget under the influence of OP, completing the stepwise coarse tuning.
AT_PVDeltTar	Tuning target value range	LREAL	Follows data type	0	Required oscillation range of the controlled object during fine-tuning oscillation tuning When the PV measurement reaches the upper or lower limit of this range, the OP output switches. Set this value to 0 during coarse tuning.
AT_InitRate	Steady-state slope before tuning	LREAL	Follows data type	0	The initial steady-state slope range at the start of tuning. When the controlled object changes within this slope range, it is considered stable. The larger the AT_InitRate, the broader the range of slopes accepted as steady-state. If AT_InitRate equals 0, the steady-state slope of the system must be 0 for tuning to begin. Therefore, this value is generally not set to 0.

PFCsConfig structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Ts	Sampling cycle	LREAL	(0,1.7E 307]	1	This value represents the control and tuning cycle. The value must be greater than 0. Ensure that this value is not set too low; otherwise it could exceed the operation cycle. If Ts is less than 0, the algorithm will automatically terminate, the switch will turn off, and the output will be held.
Model	Model type	DINT	0, 1, 2	1	Model types include: 0: Non-integral 1: Pseudo-integral 2: Pure integral Note that the algorithm internally distinguishes between non-integral and integral models. The logic for pseudo-integral models follows the same rules as those for integral models. When the model selection is not 0, 1, or 2, the algorithm terminates, the switch is turned off, and the output will be held.
COF	Filter coefficient	LREAL	[0,1]	0.9	An algorithm parameter that affects the performance of feedback correction. The feedback value balances the measurement for the current cycle with the model prediction. The magnitude of the COF determines whether the algorithm places greater weight on the measured value or the predicted value in the feedback loop. • When set to 0, the feedback value uses the model prediction, and no feedback correction is applied. • When set to 1, the feedback value uses the measured value. For other values, an average is calculated.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
COP	Performance coefficient	LREAL	(0,1.7E 307]	1	An algorithm parameter that affects the control performance. Decreasing the value speeds up the system response to some extent, while increasing it slows down the system response to some extent.
Tref	Closed-loop reference time	LREAL	(0,1.7E 307]	1	An algorithm parameter that affects the control performance. It is used to reduce the system response speed. Increasing this value slows down the system response to some extent, while decreasing it will speed up the system response to some extent. If the value exceeds the limit, it will be capped at 1e-6.
I_COF	Integral deviation correction coefficient	LREAL	[0,1]	0.1	Correction coefficient for integral speed. This value will take effect when the integral model is used. The larger the value, the faster the integral model speed operates in the algorithm, resulting in greater changes in future prediction, and thus causing output changes. Generally, fine-tuning this value after reaching steady state can reduce steady-state fluctuations.
DOPmax	Output increment limit	LREAL	>=0	100	Sets the upper limit of output increment.
OP_Low_Val	Output valid lower limit	LREAL	Follows data type	0	When the value falls below this output valid lower limit, the switch will be turned off and the control will be disabled.
OP_High_Val	Output valid upper limit	LREAL	Follows data type	100	When the value exceeds this output valid upper limit, the switch will be turned off and the control will be disabled.
OP_Low_Ope	Output operation lower limit	LREAL	Follows data type	0	When the value falls below this output operation lower limit, a warning is issued and the output is clamped to this limit.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
OP_High_Ope	Output operation upper limit	LREAL	Follows data type	100	When the value exceeds this output operation upper limit, a warning will be issued and the output is clamped to this limit.
SP_Low_Val	Setpoint valid lower limit	LREAL	Follows data type	0	Valid lower limit of the setpoint. When the value falls below this lower limit, the switch will be turned off and the control will be disabled.
SP_High_Val	Setpoint valid upper limit	LREAL	Follows data type	100	Valid upper limit of the setpoint. When the value exceeds this upper limit, the switch will be turned off and the control will be disabled.
SP_Low_Ope	Setpoint operation lower limit	LREAL	Follows data type	0	Operation lower limit of the setpoint. When the value falls below this lower limit, the system will display a warning and limit the value to the lower limit.
SP_High_Ope	Setpoint operation upper limit	LREAL	Follows data type	100	Operation upper limit of the setpoint. When the value exceeds this upper limit, the system will display a warning and limit the value to the upper limit.
SP_DMax	Setpoint increment upper range	LREAL	>=0	0.1	An algorithm parameter that affects the control performance. The system is considered stable if PV is within this range of SP + SP_DMax.
SP_DMin	Setpoint increment lower range	LREAL	<=0	-0.1	An algorithm parameter that affects the control performance. The system is considered stable if PV is within this range of SP - SP_DMin.

Output variable

Variable	Name	Data Type	Valid Range	Default Value	Description
State	Switch state	DINT	0, 1	-	Indicates the actual state of the controller.

Variable	Name	Data Type	Valid Range	Default Value	Description
ERR	Fault code	DINT	Follows data type	-	0 for normal operation; non-zero when an error occurs. See PFCs fault codes.
OP	Valve position output value	LREAL	Follows data type	-	Controller output value.
AT_ST	Tuning result	DINT	0, 1, 2, 3	-	Tuning state display. 0: Not tuned 1: Coarse tuning in progress 2: Fine tuning in progress 3: Tuning completed
PFCsOutput	Output parameter structure	PFCsOutput	Follows data type	-	Output parameter configuration structure.

PFCsOutput structure configuration

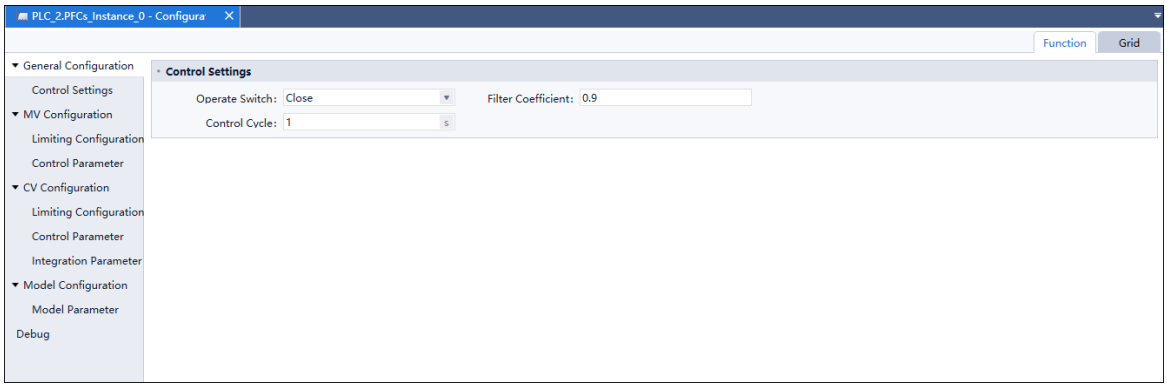
Output Variable	Name	Data Type	Valid Range	Default Value	Description
Version	Version information	ARRAY[0..3] OF DINT	Follows data type	-	Used to record internal version information.
DOP	Valve position increment output	LREAL	Follows data type	-	Increment of the output value OP.
PV_Pred_t	Estimated value	LREAL	Follows data type	-	Estimated value, that is, the response value of the controlled object, calculated by the internal model using historical data.
PV_Pred_tf	Predicted value	LREAL	Follows data type	-	Predicted value, that is, the estimated value corrected based on the actual measured value PV.
PV_Pred_e	Prediction error	LREAL	Follows data type	-	Difference between the actual measured value PV and the predicted value.
PV_Pred_eSum	Cumulative prediction error	LREAL	Follows data type	-	Cumulative sum of prediction errors.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
DCV_S	Integral speed	LREAL	Follows data type	-	Integral deviation speed.
PV_Pred	Open-loop prediction sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Records the sequence of future measured values predicted when the current cycle is not applied.
Tref_Step	Reference trajectory sequence	ARRAY[0..199] OF LREAL	Follows data type	-	A trajectory sequence generated by the Tref parameter.
OP_Low_Ope_Past	Output lower limit history sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Sequence of historical operation lower limits for the controller output OP.
OP_High_Ope_Past	Output upper limit history sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Sequence of historical operation upper limits for the controller output OP.
OP_Low_Ope_Future	Output lower limit future sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Records the sequence of predicted operation lower limits for the controller output at future sampling instants, computed after the current cycle has been applied.
OP_High_Ope_Future	Output upper limit future sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Records the sequence of predicted operation upper limits for the controller output at future sampling instants, computed after the current cycle has been applied.
SP_Past	Setpoint history sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Records the sequence of the historical setpoints.
SP_Future	Setpoint future sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Records the sequence of future setpoints predicted by the algorithm after the current cycle has been applied.
SP_Low_Ope_Past	Setpoint lower limit history sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Records the sequence of historical operation lower limits for the setpoint.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
SP_High_Ope_Past	Setpoint upper limit history sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Records the sequence of historical operation upper limits for the setpoint.
SP_Low_Ope_Future	Setpoint lower limit future sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Records the sequence of predicted operation lower limits for the setpoint at future sampling instants, computed after the current control cycle has been applied.
SP_High_Ope_Future	Setpoint upper limit future sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Records the sequence of predicted operation upper limits for the setpoint at future sampling instants, computed after the current control action has been applied.
PV_Past	Measured value history sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Records the sequence of historical measured values.
OP_Past	Output value history sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Records the sequence of historical output values.
OP_Future	Output value future sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Records the sequence of future output values predicted by the algorithm after the current cycle has been applied.
AT_Km	Tuning parameter Km	LREAL	Follows data type	-	Model gain result obtained from auto-tuning.
AT_Tm	Tuning parameter Tm	LREAL	> 0	-	Model time constant result obtained from auto-tuning.
AT_Dm	Tuning parameter Dm	LREAL	>=0	-	Model hysteresis parameter result obtained from auto-tuning.

- **Function**
  - PFCs is a simplified 1x1 single-input, single-output predictive function controllers. It includes one manipulated variable (output via OP) and one controlled variable (measured via PV), and support a feedforward input channel (for use with the PFC_DV function block). The PFCs component features a built-in auto-tuning function to obtain model parameters of the controlled object. This component is primarily used for rapid deployment in single-input single-output processes.
  - The PFCs function block includes an intuitive interface for parameter configuration and tuning. After adding the PFCs component, select "Control object > PFC" in the left project tree, and then double-click

the PFCs instance under the PFC node to open the configuration and tuning interface, as shown in the figure below.



- Parameter configuration

  The parameter configuration of the PFCs instruction includes general configuration, MV configuration, CV configuration, and model configuration. For detailed parameter descriptions, see Relevant variables.

- Parameter auto-tuning

  The PFCs instruction includes a built-in parameter auto-tuning function, automatically obtaining the model parameters (Km, Tm, and Dm) of the controlled object.

  Click the "Tuning" button on the "Commissioning" page to start parameter tuning.

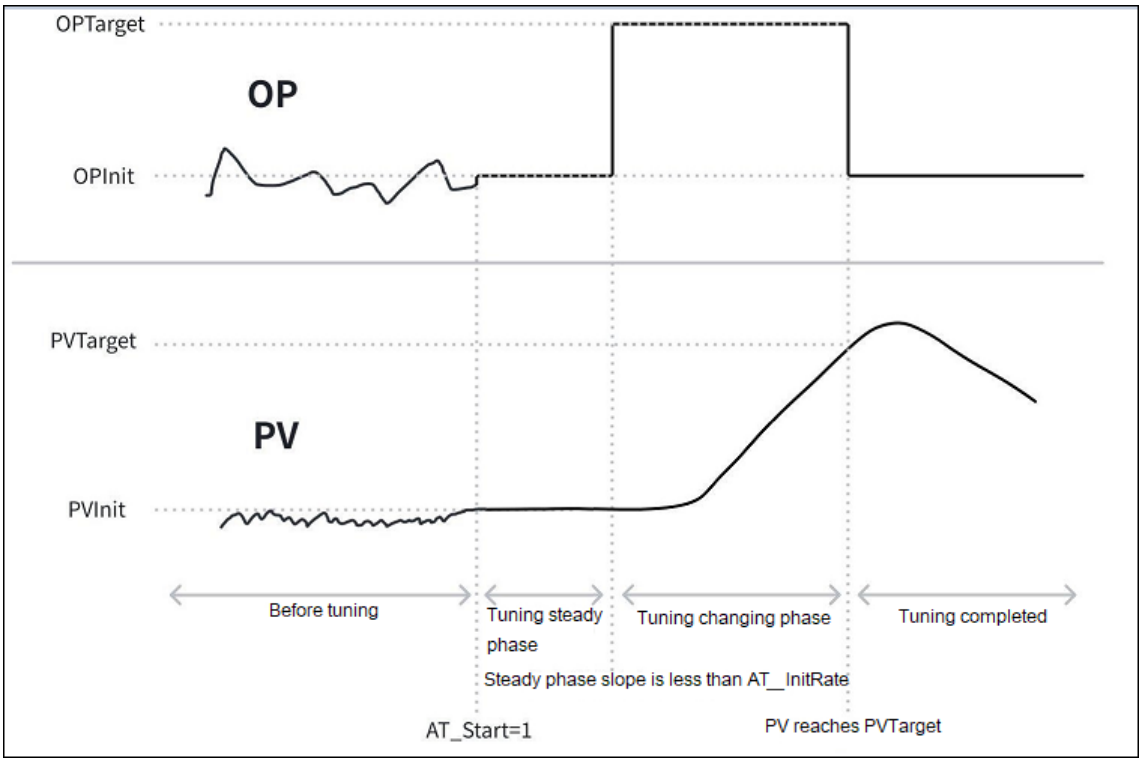


Because the controller internal variable array has a capacity of only 200, ensure that the duration from the OP output step to the completion of tuning stays within 200 control cycles. 100 to 150 cycles is recommended. Additionally, the tuning process does not require the system to reach final steady state after the step change.

The tuning process is as follows:

1. Use the auto-tuning switch AT_SW to select the model type for the tuning object.
2. Configure the tuning parameter "Initial OP value for tuning" (AT_OPInit). AT_OPInit is the OP output value at the initial steady state. Ensure that the controlled object reaches a basic stable state at this output. Open-loop control can be used before the tuning process to stabilize the controlled object at this output, then switch to auto-tuning. The initial steady state is critical for the tuning results. When it is difficult to achieve the initial steady state, the "Steady-state slope" (AT_InitRate) parameter can be used to appropriately relax the steady-state requirements.

3. Configure the tuning parameter "Tuning step value" (AT_OPTarget). This value is the step output value at the start of tuning. The controller will perform a step change from "AT_OPInit" to "AT_OPTarget" to excite the response of the controlled object.
4. Configure the tuning parameter "PV target value at the end of tuning" (AT_PVTarget). The tuning process completes when the measured value (PV) of the controlled object reaches this value after tuning begin. Pay special attention to the response polarity of the controlled object. Ensure that the measured value reaches AT_PVTarget after tuning starts, rather than the PV changing in the opposite direction after the OP step change.
5. Click the "Start tuning" button.
6. After tuning ends, click "Confirm" or "Cancel" to select parameters, and automatically switch to control mode.



- **Program example**

  This instruction is a control instruction and requires connection to the inputs and outputs of an external or simulated controlled object. Accordingly, the input PV of the PFCs instruction must be connected to the output of the controlled object, while the output OP of the instruction is connected to the input of the controlled object.

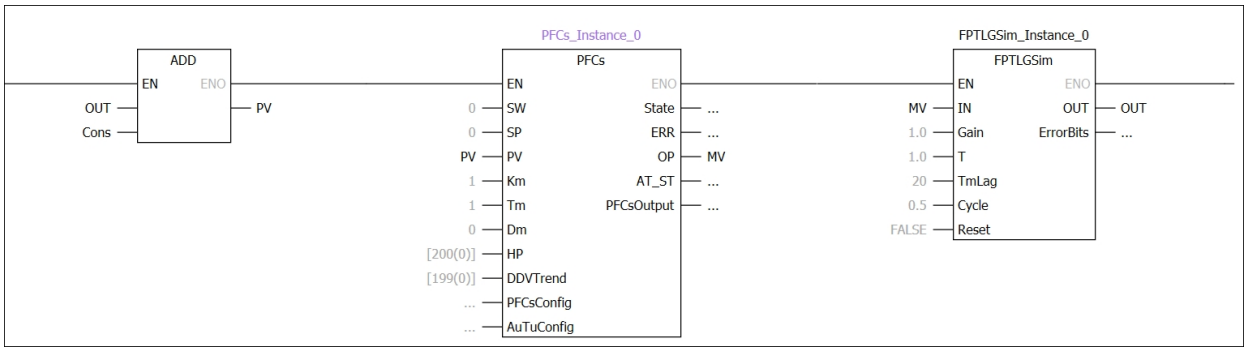
  Here, the collaborative simulation between the PFCs instruction and the first-order inertia plus pure delay simulation instruction (FPTLGsim) is used as an example. The parameter configuration and control process are as follows:

1. Parameter configuration
   Based on the characteristics of the control object, configure the relevant parameters of the PFCs instruction in the configuration interface.
2. Control process
   The PFCs instruction responds to the output (PV) of the FPTLGsim object, calculates the control output (OP), and applies it to the original controlled object, forming a closed control loop.

3. Control expectation

   If all parameters are configured properly, the controller will eventually keep the output value (PV) of the controlled object near the setpoint SP.

   LD language code example:

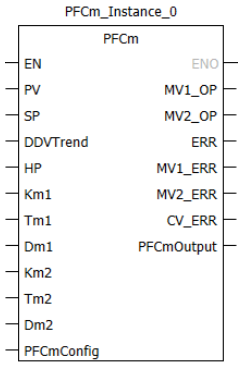


- **Precautions**
  - The module execution cycle must be consistent with the function block cycle parameter Ts.
  - Ensure the switch SW is in the open state (Value: 1) and the configuration is correct for use.
  - The execution time of the PFCs instruction is 8 ms. With 130 PFCs function blocks loaded, the total execution time is estimated to reach approximately 1 second. Loading 562 PFCs function blocks will reach the controller's resource limit. During operation, configure the scheduling task cycle properly. For example, if 10 PFCs are required to perform loop control simultaneously, with a block calculation time of 80 ms. The task cycle must be at least twice this value, that is, 160 ms, to ensure the CPU load does not become too high.

18.3.2.3 PFCm

This instruction is a single-input dual-output control instruction. It is primarily used in "2-to-1" control scenarios, such as regulating temperature via heating and cooling.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PFCm	Multivariable advanced control	FB		PFCm_Instance_0(PV := , SP := , DDVTrend := , HP := , Km1 := , Tm1 := , Dm1 := , Km2 := , Tm2 := , Dm2 := , PFCmConfig := , MV1_OP => , MV2_OP => , ERR => , MV1_ERR => , MV2_ERR => , CV_ERR => , PFCmOutput =>);

- **Related variables**

  Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
PV	Measured value of the controlled variable	LREAL	Follows data type	0	Input value of the controlled variable measurement, which must be provided by an external physical object or simulation object.
SP	Controlled variable setpoint value	LREAL	Follows data type	50	Setpoint value of the controlled variable, which determines the control objective of the controller so that the final actual control result (PV) must closely follow the SP.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
DDVTrend	Disturbance sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Disturbance sequence generated by the PFCs_dv component. When using this variable, pass its DDVTrend output to this variable. It is used to predict future disturbances so that the controller can take proactive action to suppress them.
HP	Prediction horizon marker	ARRAY[0..199] OF DINT	Follows data type	-	Note that if there are no valid values (that is, 1) in this array after M_Dm_CV_min, there will be no valid prediction. The internal logic will overwrite the values in this array to ensure a valid prediction horizon.
Km1	Model 1 gain	LREAL	Follows data type	1	An algorithm parameter that describes the response gain of the controlled object. It must be determined based on the influence of the output OP on the controlled object. The greater this influence, the larger the gain value.
Tm1	Model 1 time constant	LREAL	(0,1.7E 307]	1	An algorithm parameter that describes the response speed of the controlled object. The greater the inertia of the controlled object, the slower the response, and the larger this value. If the value exceeds the limit, it will be limited to 1e-6.
Dm1	Model 1 hysteresis	LREAL	[0,1.7E 307]	0	An algorithm parameter that represents the response hysteresis of the controlled object. The greater the hysteresis of the object to input changes, the larger this value. If the value exceeds the limit, it will be limited to 0.
Km2	Model 2 gain	LREAL	Follows data type	1	Same as Km1; this is the gain for model 2.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Tm2	Model 2 time constant	LREAL	(0,1.7E 307]	1	Same as Tm1, this is the time constant for model 2. If the value exceeds the limit, it will be limited to 1e-6.
Dm2	Model 2 hysteresis	LREAL	[0,1.7E 307]	0	Same as Dm2; this is the hysteresis for model 2. If the value exceeds the limit, it will be limited to 0.
PFCmConfig	Input parameter structure	PFCmConfig	Follows data type	-	Input parameter configuration structure.

PFCmConfig structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Ts	Sampling cycle	LREAL	(0,1.7E 307]	1	This value determines the control cycle and must be greater than 0. Note that this value must not be too low; otherwise it can cause a time-out during computation, affecting control. When Ts is set to an invalid value (that is, less than 0), the algorithm will automatically terminate, the switch is turned off, and the output will remain unchanged.
Model	Model type	DINT	0, 1, 2	1	Model type: 0: Non-integral 1: Pseudo-integral 2: Pure integral Note that the algorithm internally distinguishes between non-integral and integral models. The logic for pseudo-integral models follows the same rules as those for integral models. When the model selection is not 0, 1, or 2, the algorithm terminates, the switch is turned off, and the output will be held.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
COF	Filter coefficient	LREAL	[0,1]	0.9	An algorithm parameter that affects the performance of feedback correction. The feedback value balances the measurement for the current cycle with the model prediction. When set to 0, the feedback value uses the model prediction, and no feedback correction is applied. When set to 1, the feedback value uses the measured value. For other values, an average is calculated. If this value exceeds the upper limit, it will be limited to the upper limit. If the value falls below the lower limit, it will be limited to the lower limit.
Tref	Closed-loop reference time	LREAL	(0,1.7E 307]	1	An algorithm parameter that affects the control performance. Increasing the value slows down the system response to some extent, while decreasing it speeds up the system response to some extent. If the value exceeds the limit, it will be limited to 1e-6.
I_COF	Integral deviation correction coefficient	LREAL	[0,1]	0.1	Correction coefficient for integral speed. This value will take effect when the integral model is used. The larger the value, the faster the integral model speed operates in the algorithm, resulting in greater changes in future prediction, and thus causing output changes. Generally, fine-tuning this value after reaching steady state can reduce steady-state fluctuations. If this value exceeds the upper limit, it will be limited to the upper limit. If the value falls below the lower limit, it will be limited to the lower limit.
MV1_Config	MV variable input parameter structure	MVConfig	Follows data type	-	MV1 variable input parameter configuration structure.
MV2_Config	MV variable input parameter structure	MVConfig	Follows data type	-	MV2 variable input parameter configuration structure.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
CV_SW	Controlled variable switch	DINT	0, 1	1	Controlled variable switch. When the switch is off, the control is disabled. When used, it must be set to 1. Invalid values are automatically set to 0.
CV_SP_Dmax	Upper deviation range	LREAL	[0,1.7E 307]	0.01	An algorithm parameter that affects the control performance. The system is considered stable if PV + CV_SP_Dmax is within this range. Invalid input values are limited to 0.
CV_SP_Dmin	Lower deviation range	LREAL	[-1.7E 307,0]	-0.01	An algorithm parameter that affects the control performance. The system is considered stable if PV + CV_SP_Dmax is within this range. Invalid input values are limited to 0.
DeltaCV	Maximum change rate of the controlled variable	LREAL	[0,1.7E 307]	100	This algorithm parameter defines the maximum change value of CV during prediction. Invalid values are limited to 100.
CV_Low_Val	Valid lower limit of the controlled variable	LREAL	Follows data type	0	Valid lower limit. When the value falls below the valid lower limit, the switch will be turned off and the control will be disabled.
CV_High_Val	Valid upper limit of the controlled variable	LREAL	Follows data type	100	Valid upper limit. When the value exceeds the valid upper limit, the switch will be turned off and the control will be disabled.
CV_Low_Ope	Operation lower limit of the controlled variable	LREAL	Follows data type	0	Operation lower limit. When the value falls below this operation lower limit, a warning will be issued and the value will be clamped to this operation lower limit.
CV_High_Ope	Operation upper limit of the controlled variable	LREAL	Follows data type	100	Operation upper limit. When the value exceeds this operation upper limit, a warning will be issued and the value will be clamped to this operation upper limit.

MVConfig structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
MV_SW	Manipulated variable switch	DINT	0, 1	1	MV switch; when this switch is off, the control is disabled. Invalid values are automatically set to 0.
MV_High_Val	Manipulated variable valid upper limit	LREAL	Follows data type	200	Valid upper limit. When the value exceeds the valid upper limit, the switch will be turned off and the control will be disabled.
MV_Low_Val	Manipulated variable valid lower limit	LREAL	Follows data type	0	Valid lower limit. When the value falls below the valid lower limit, the switch will be turned off and the control will be disabled.
MV_High_Ope	Operation upper limit of the manipulated variable	LREAL	Follows data type	100	Operation upper limit. When the value exceeds this operation upper limit, a warning will be issued and the value will be clamped to this operation upper limit.
MV_Low_Ope	Operation lower limit of the manipulated variable	LREAL	Follows data type	0	Operation lower limit. When the value falls below this operation lower limit, a warning will be issued and the value will be clamped to this operation lower limit.
MV_DMax	Maximum positive increment for the manipulated variable	LREAL	[0,1.7E 307]	100	Defines the maximum allowable positive increment of the output. Invalid input values are limited to 100.
MV_DMin	Maximum negative increment for the manipulated variable	LREAL	[-1.7E 307, 0]	-100	Defines the maximum allowed negative increment of the output. Invalid input values are limited to -100.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
MV_COP	Manipulated variable performance coefficient	LREAL	[0,1.7E 307]	1	An algorithm parameter that affects the control performance. Decreasing the value speeds up the system response to some extent, while increasing it slows down the system response to some extent. If this value exceeds the upper limit, it will be limited to the upper limit. If the value falls below the lower limit, it will be limited to the lower limit.
MV_Irv	Ideal injection value for the manipulated variable	LREAL	Follows data type	1	The value that the corresponding MV output eventually converges to.
MV_Irv_C	Weight for manipulated variable ideal injection value	LREAL	Follows data type	1	Corresponding weight. The higher the weight, the closer the MV will converge to MV_Irv.
MV_Irv_SW	Ideal injection value switch	DINT	0, 1	0	When set to 0, the ideal injection value is inactive. When used, the value must be set to 1. Invalid values are automatically set to 0.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
MV1_OP	Manipulated variable 1 output value	LREAL	Follows data type	-	Output value of manipulated variable 1. This value must be output to the simulation object or actual object.
MV2_OP	Manipulated variable 2 output value	LREAL	Follows data type	-	Output value of manipulated variable 2. This value must be output to the simulation object or actual object.
ERR	Fault code	DINT	Follows data type	-	For fault codes, see PFCm fault codes.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
MV1_ERR	Manipulated variable 1 fault code	DINT	Follows data type	-	Fault code of manipulated variable 1.
MV2_ERR	Manipulated variable 2 fault code	DINT	Follows data type	-	Fault code of manipulated variable 2.
CV_ERR	Controlled variable fault code	DINT	Follows data type	-	Fault code of controlled variable.
PFCmOutput	Output parameter structure	PFCmOutPut	Follows data type	-	Output parameter structure.

PFCmOutput structure configuration

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Version	Version information	ARRAY[0..3] OF DINT	Follows data type	-	Used to obtain the version information.
CV_Pred_t	Estimated value of controlled variable	LREAL	Follows data type	-	Estimated value, that is, the response value of the controlled object, calculated by the internal model using historical data.
CV_Pred_tf	Predicted value of controlled variable	LREAL	Follows data type	-	Predicted value, that is, the corrected predicted value based on the predicted value and the actual measured value PV.
DCV_S_i	Velocity after integral deviation correction	LREAL	Follows data type	-	Velocity after integral deviation correction.
CV_PV_e	Predicted error of the controlled variable	LREAL	Follows data type	-	Difference between the actual measured value and the predicted value.
CV_PV_eSum	Cumulative sum of predicted errors for the controlled variable	LREAL	Follows data type	-	The cumulative sum of errors (regardless of sign). Its magnitude indicates the degree of model mismatch.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
NFlag	Number of errors	DINT	[0, 10]	-	Counts the number of errors.
Flag	Error information	ARRAY[0..9] OF DINT	Follows data type	-	Records detailed information of each error.
MV1_OutConfig	Manipulated variable 1 output structure	MVOutConfig	Follows data type	-	Output structure of manipulated variable 1.
MV2_OutConfig	Manipulated variable 2 output structure	MVOutConfig	Follows data type	-	Output structure of manipulated variable 2.
CV_NFlag	Controlled variable error count	DINT	[0, 10]	-	Number of recorded errors for the controlled variable
CV_Flag	Controlled variable error information	ARRAY[0..9] OF DINT	Follows data type	-	Detailed information of recorded errors for the controlled variable
CV_ST	Controlled variable switch status	DINT	0, 1	-	Status of the controlled variable switch. Initially the value equals the switch input value. When an internal error occurs, this value is set to 1.
CV_Pred_t_Past	Sequence of historical estimated values for the controlled variable	ARRAY[0..199] OF LREAL	Follows data type	-	Records historical values.
CV_Pred	Open-loop prediction sequence of controlled variables	ARRAY[0..199] OF LREAL	Follows data type	-	Records the sequence of future values predicted when the current cycle is not applied.
CV_Future	Closed-loop prediction sequence of controlled variables	ARRAY[0..199] OF LREAL	Follows data type	-	Records the sequence of future values predicted after the current cycle has been applied.

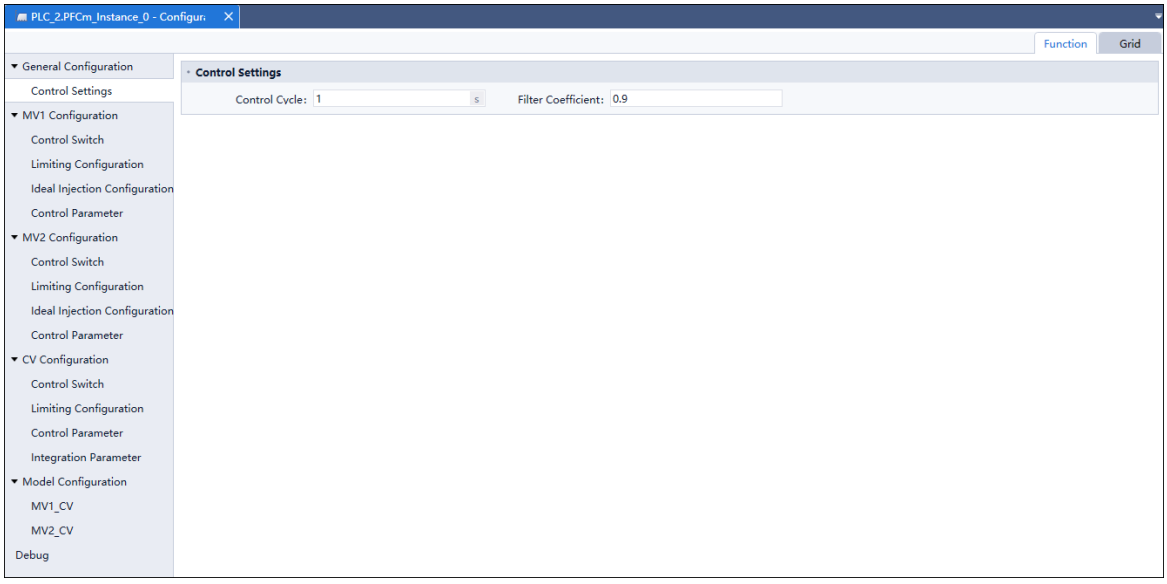
Output Variable	Name	Data Type	Valid Range	Default Value	Description
CV_PV_Past	Sequence of historical measured values for the controlled variable	ARRAY[0..199] OF LREAL	Follows data type	-	Records historical measured values.
SP_Past	Setpoint history sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Sequence of historical setpoint values, for plotting.
SP_Future	Setpoint future sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Sequence of the future setpoint values, for plotting.
CV_SP_Href	Upper reference trajectory sequence	ARRAY[0..199] OF LREAL	Follows data type	-	The predicted upper trajectory that the controlled variable is expected to converge to.
CV_SP_Lref	Lower reference trajectory sequence	ARRAY[0..199] OF LREAL	Follows data type	-	The predicted lower trajectory that the controlled variable is expected to converge to.
CV_Low_Past	Controlled variable lower limit history sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Sequence of historical operation lower limits for the controlled variable, for plotting.
CV_High_Past	Controlled variable upper limit history sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Sequence of historical upper limits for the controlled variable, for plotting.
CV_Low_Future	Controlled variable lower limit future sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Sequence of future operation lower limits for the controlled variable, for plotting.
CV_High_Future	Controlled variable upper limit future sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Sequence of future operation upper limits for the controlled variable, for plotting.

MVOutConfig structure configuration

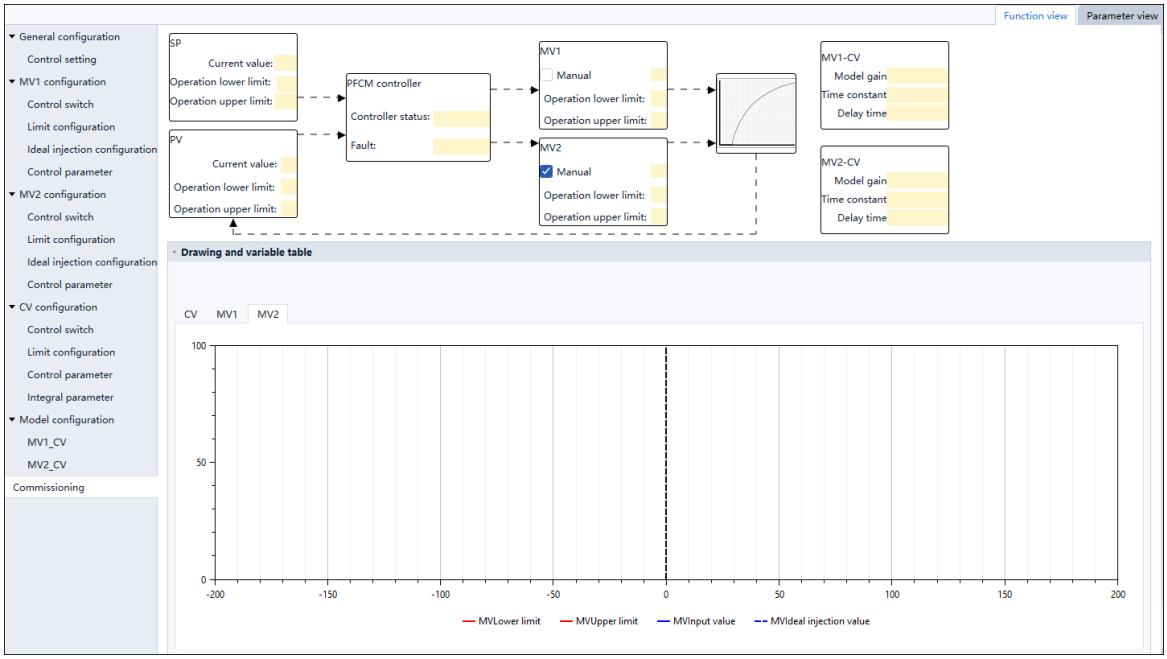
Input Variable	Name	Data Type	Valid Range	Default Value	Description
MV_NFlag	MV error count	DINT	[0, 10]	-	Number of errors counted for the manipulated variable.
MV_Flag	MV error message	ARRAY[0..9] OF DINT	Follows data type	-	Detailed information of recorded errors for the manipulated variable.
MV_Low_Past	Output lower limit history sequence	ARRAY[0..9] OF LREAL	Follows data type	-	Sequence of historical operation lower limits for the manipulated variable, for plotting.
MV_High_Past	Output upper limit history sequence	ARRAY[0..9] OF LREAL	Follows data type	-	Sequence of historical operation upper limits for the manipulated variable, for plotting.
MV_Low_Future	Output lower limit future sequence	ARRAY[0..9] OF LREAL	Follows data type	-	Sequence of future operation lower limits for the manipulated variable, for plotting.
MV_High_Future	Output upper limit future sequence	ARRAY[0..9] OF LREAL	Follows data type	-	Sequence of future operation upper limits for the manipulated variable, for plotting.
MV_Past	Output value history sequence	ARRAY[0..9] OF LREAL	Follows data type	-	Sequence of historical measured values for the manipulated variable, for plotting.
MV_Future	Output value future sequence	ARRAY[0..9] OF LREAL	Follows data type	-	Sequence of future measured values for the manipulated variable, for plotting.
MV_DOP_Past	Historical increment sequence of the valve position	ARRAY[0..9] OF LREAL	Follows data type	-	Sequence of historical increment values for the manipulated variable, for plotting.
MV_DOP	Manipulated variable action increment	LREAL	Follows data type	-	Increment value of the manipulated variable.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
MV_ST	Manipulated variable switch state	DINT	0, 1	-	Manipulated variable switch state. Initially, it equals the switch value. This value is set to 1 when an internal error occurs.
MV_Irv_Past	Ideal injection value history sequence	ARRAY[0..9] OF LREAL	Follows data type		Sequence of historical ideal injection values, for plotting.
MV_Irv_Future	Ideal injection value future sequence	ARRAY[0..9] OF LREAL	Follows data type		Sequence of future ideal injection values, for plotting.

- **Function**
  - PFCm is a 1×2 single-input, dual-output control instruction. The controlled object can support non-integrating, integrating, or pseudo-integrating responses. The instruction includes two manipulated variables (MV), one controlled variable (CV), and a feedforward input channel (for use with PFCs_dv). It does not rely on an external solver for optimization calculations. It is primarily intended for applications that require two manipulated variables but not high-end products. It is suitable for fast control application scenarios with millisecond-level control cycles.
  - The PFCm function block includes an intuitive interface for parameter configuration and commissioning. After adding the PFCm component, select "Control Objects -> PFC" in the left project tree, and then double-click the PFCm instance under the PFC node to open the configuration and tuning interface, as shown in the figure below.



    - Parameter configuration
      The parameter configuration of the PFCm instruction includes general configuration, MV configuration, CV configuration, and model configuration. For detailed parameter descriptions, see Relevant variables.
    - Commissioning
      After the parameters are configured, the control effect can be viewed directly in the commissioning interface.



- **Program example**

  Here, the two-input, single-output control simulation between the PFCm instruction and two first-order inertia plus pure delay simulation instruction (FPTLGsim) is used as an example. The parameter configuration and control process are as follows:

1. Parameter configuration

   Configure the PFCm instruction parameters based on the characteristics of the control object (Model parameters: Km1, Km2, Dm1 and so on; controller parameters: COP, COF, etc.).

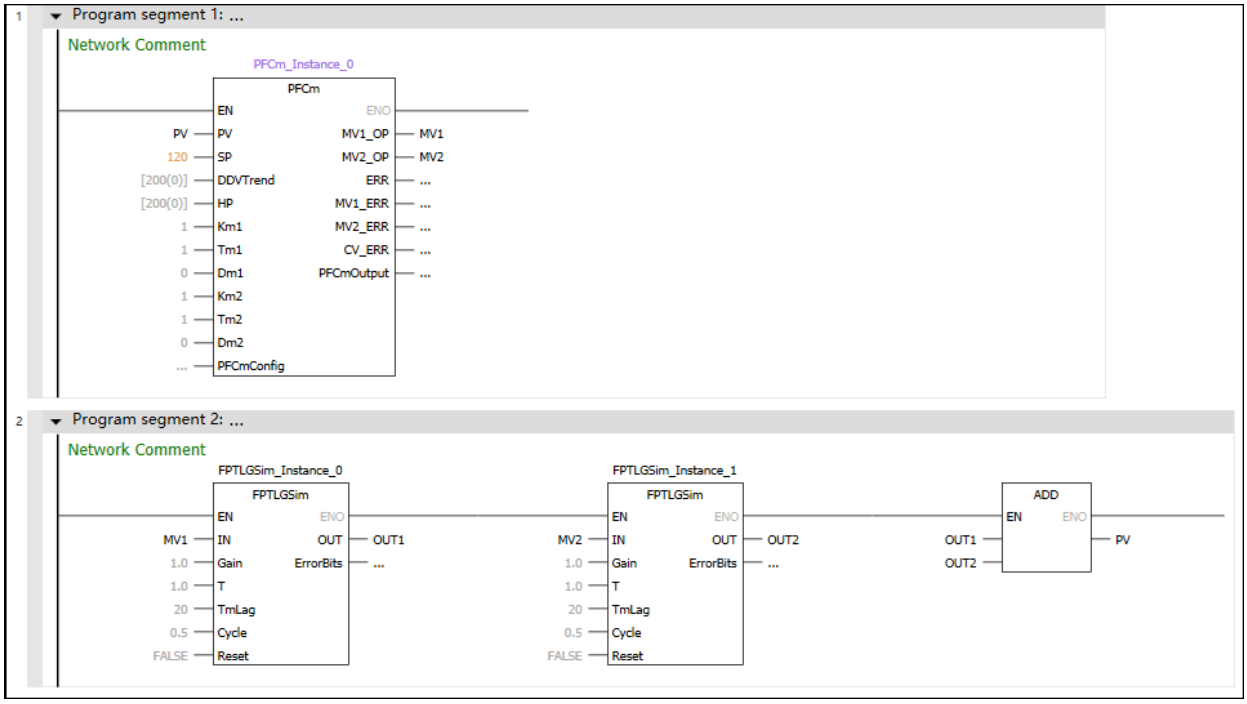
2. Control process

   The PFCm instruction responds to the output (PV) of the FPTLGsim object, obtains the control quantity (OP) output, and applies it to the original control object, forming a control loop.

3. Control expectation

   If all parameters are configured properly, the controller will eventually keep the output value (PV) of the controlled object near the setpoint SP.

   LD language code example:

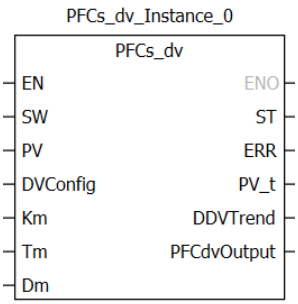


- **Precautions**
  - During operation, ensure that the switch is on, and the configuration is correct.
  - The execution time of the PFCm instruction is 20 ms. With 50 PFCm function blocks loaded, the total execution time is estimated to reach approximately 1 second. Loading 364 PFCm function blocks will reach the controller's resource limit. During operation, configure the scheduling task cycle properly. For example, if 10 PFCm are required to perform loop control simultaneously, with a block calculation time of 200 ms. The task cycle must be at least twice this value, that is, 400 ms, to ensure the CPU load does not become too high.

18.3.2.4 PFCs_dv

The disturbance instruction outputs a predicted disturbance sequence and must be used in conjunction with a corresponding control instruction.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PFCs_dv	Feedforward predictive control	FB		PFCs_dv_Instance_0(SW := , PV := , DVConfig := , Km := , Tm := , Dm := , ST => , ERR => , PV_t => , DDVTrend => , PFCdvOutput =>);

- **Related variables**

  Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
SW	Disturbance switch	DINT	0, 1	1	The disturbance switch disables the control when the switch is off. When used, it must be 1. Any invalid value is set to 0.
PV	Input value	LREAL	Follows data type	0	Disturbance input value provided externally. Based on this input value, a corresponding disturbance sequence is generated.
DVConfig	Input parameter structure	DVConfig	Follows data type	-	Input parameter configuration structure.
Km	Model gain	LREAL	Follows data type	1	An algorithm parameter that describes the response gain of the controlled object. It must be determined based on the influence of the output OP on the controlled object. The greater this influence, the larger the gain value.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Tm	Model time constant	LREAL	(0,1.7E 307]	1	An algorithm parameter that describes the response speed of the controlled object. The greater the inertia of the controlled object, the slower the response, and the larger this value. If the value exceeds the limit, it will be limited to 1e-6.
Dm	Model hysteresis	LREAL	[0,1.7E 307]	0	An algorithm parameter that represents the response hysteresis of the controlled object. The greater the hysteresis of the object to input changes, the larger this value. If the value exceeds the limit, it will be limited to 0.

DVConfig structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Ts	Sampling cycle	LREAL	(0,1.7E 307]	1	This value determines the control cycle and must be greater than 0. Note that this value must not be too low; otherwise it can cause a time-out during computation, affecting control. When Ts is set to an invalid value (that is, less than 0), the algorithm will automatically terminate, the switch is turned off, and the output will remain unchanged.
Model	Model type	DINT	0, 1, 2	1	Model type: 0: Non-integral 1: Pseudo-integral 2: Pure integral Currently, types 0 (Non-integral) and 1 (Pseudo-integral) are primarily used. Type 2 (Pure integral) is not implemented. In practice, integral models are typically handled as pseudo-integral. Note that this parameter only serves as display effect in DV. When used, it must match the model type of the control algorithm component. If the model selection does not follow 0, 1, 2, it is set to 0 by default.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Low	Lower limit of the disturbance variable	LREAL	Follows data type	0	Lower limit; if the value falls below the lower limit, it is clamped to the lower limit.
High	Upper limit of the disturbance variable	LREAL	Follows data type	100	Upper limit; if the value exceeds the upper limit, it is clamped to the upper limit.
DMax	Increment limit	LREAL	Follows data type	100	Increment limit; when exceeded, it will be clamped to the maximum increment value. Any invalid value is set to 0.

Output variable

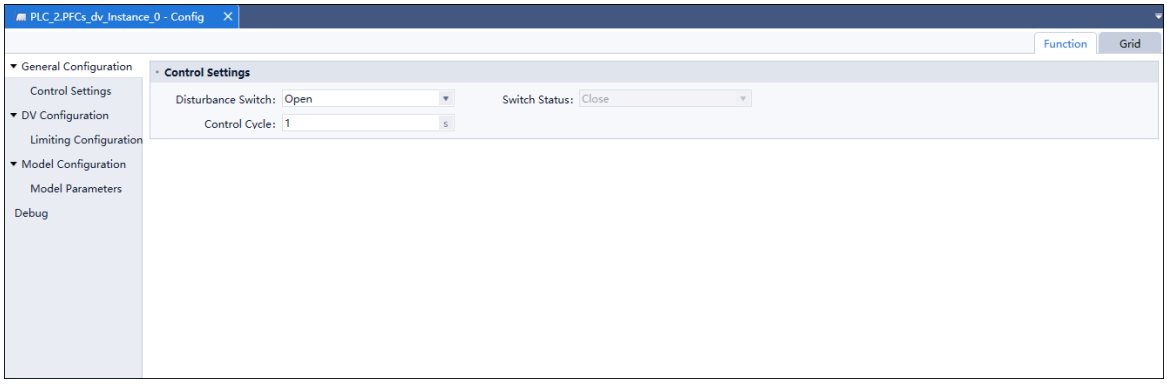
Output Variable	Name	Data Type	Valid Range	Default Value	Description
ST	Switch state	DINT	0, 1	-	Switch status. Initially the value equals the switch input value. When an internal error occurs, this value is set to 1.
ERR	Fault code	DINT	Follows data type	-	For fault codes, see PFCs_dv fault codes.
PV_t	Feedforward correction value	LREAL	Follows data type	-	Feedforward value after limit.
DDVTrend	Disturbance sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Disturbance sequence. After generating this sequence, it must be output to the input interface with the same name on the control instruction for feedforward correction.
PFCdvOutput	Output parameter structure	PFCdvOutput	Follows data type	-	Output parameter structure

PFCdvOutput structure configuration

Output Variable	Name	Data Type	Valid Range	Default Value	Description
PV_Past	Measured value history sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Records historical values for the disturbance input.
PVt_Past	Feedforward value history sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Records historical values of the feedforward signal.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
DV_Low_Past	Disturbance value lower limit history sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Sequence of historical lower limits for the disturbance value, for plotting.
DV_High_Past	Disturbance value upper limit history sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Sequence of historical upper limits for the disturbance value, for plotting.
DV_Low_Future	Disturbance value lower limit future sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Sequence of future lower limits for the disturbance value, for plotting.
DV_High_Future	Disturbance value upper limit future sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Sequence of future upper limits for the disturbance value, for plotting.
DV_PV_Future	Disturbance value future sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Sequence of future disturbance values, for plotting.
DDV_Past	Feedforward value history increment sequence	ARRAY[0..199] OF LREAL	Follows data type	-	Sequence of historical increments for the feedforward value, for plotting.
Version	Version information	ARRAY[0..3] OF DINT	Follows data type	-	Records the version information.

- **Function**
  - PFCs_dv is a disturbance feedforward prediction module. Its primary function is to predict the future disturbance sequence of the controlled object based on the current disturbance input. The predicted disturbance sequence can be fed forward to the feedforward input channel of the PFCs/PFCm instruction, enabling proactive compensation of disturbances and improving disturbance rejection.
  - The PFCs_dv function block includes an intuitive interface for parameter configuration and commissioning. After adding the PFCs_dv component, select "Control object > PFC" in the left project tree, and then double-click the PFCs_dv instance under the PFC node to open the configuration and tuning interface, as shown in the figure below.



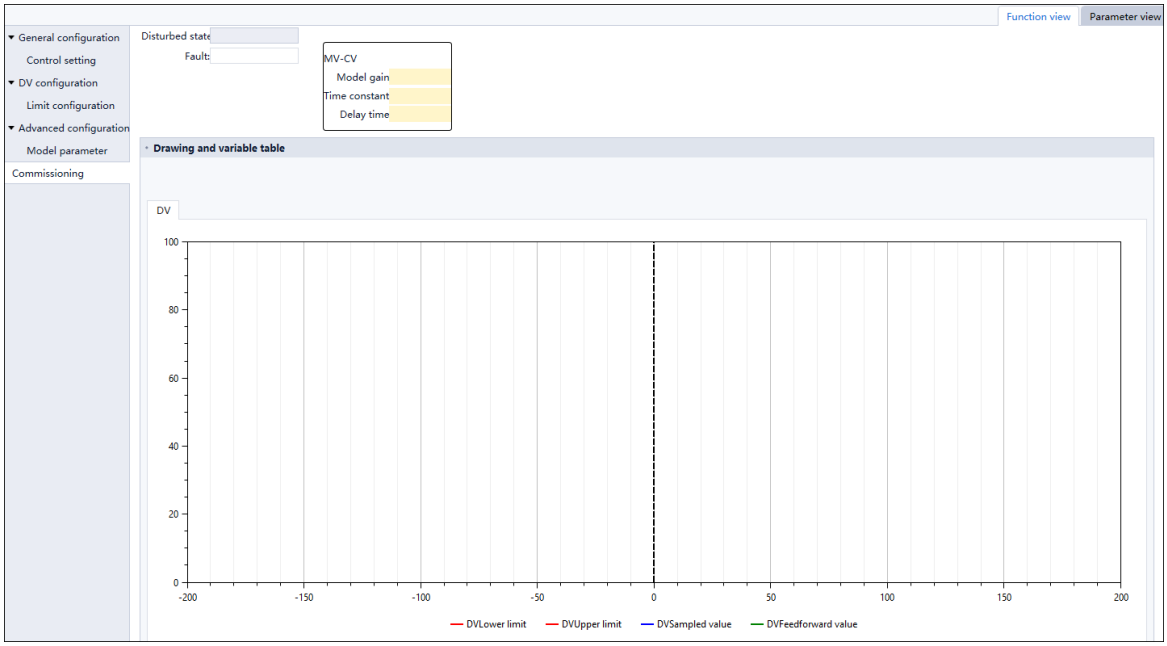
    - Parameter configuration

      The parameter configuration of the PFCs_dv instruction includes general configuration, CV

configuration, and model configuration. For detailed parameter descriptions, see Relevant variables.

- Commissioning

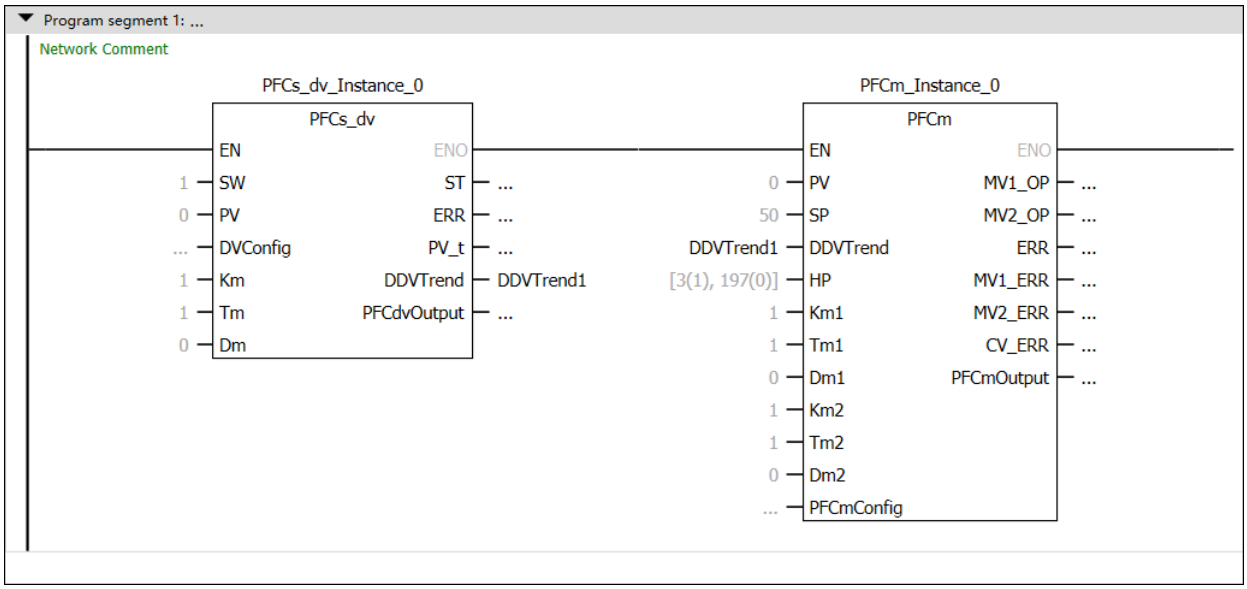
  After the parameters are configured, the sequence of the predicted future disturbance values for the controlled variable can be viewed directly in the commissioning interface.



- **Program example**

  During operation, this instruction primarily feeds the sequence output by DDVTrend into the control component PFCm or PFCs. The generated sequence is determined by the model parameters Km, Tm, and Dm, as well as the input disturbance PV.

  LD:



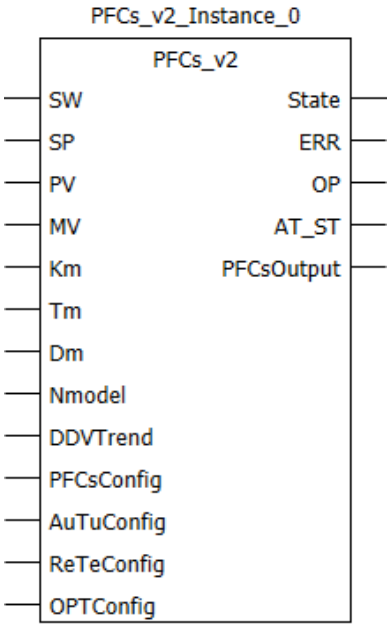
- **Precautions**
  - During operation, ensure that the switch is on, and the configuration is correct.
  - When setting the model parameters, ensure they are consistent with the controller parameter type, either all integral or non-integral.
  - The execution time of the PFCs_dv instruction is 8 ms. With 140 PFCs_dv function blocks loaded, the total execution time is estimated to reach approximately 1 second. Loading 1154 PFCs_dv function

blocks will reach the controller's resource limit. During operation, configure the scheduling task cycle properly. For example, if 10 PFCs_dv are required to perform loop control simultaneously, with a block calculation time of 80 ms. The task cycle must be at least twice this value, that is, 160 ms, to ensure the CPU load does not become too high.

18.3.2.5 PFCs_v2

PFCs are simplified predictive function controller (PFCs_v2) components for single-input, single-output systems, primarily used in 1x1 control scenarios such as temperature heating control.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PFCs_v2	Single-loop advanced control V2	FB		PFCs_v2_Instance_0(SW := , SP := , PV := , MV := , Km := , Tm := , Dm := , Nmodel := , DDVTrend := , PFCsConfig := , AuTuConfig := , ReTeConfig := , OPTConfig := , State => , ERR => , OP => , AT_ST => , PFCsOutput =>);

- **Related variables**

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
SW	Operating switch	DINT	0, 1	0	0: Component off. 1: Component on. Ensure the component is in the on state when in use.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
SP	Setpoint	REAL	Follows data type	0	Setpoint for the controlled variable, which defines the control target of the controller. If properly configured, the actual control result (PV) must closely follow the SP.
PV	Measured value	REAL	Follows data type	0	Input value of the controlled variable measurement, which must be provided by an external physical object or simulation object.
MV	Measured value of the valve position	REAL	Follows data type	0	Measured value of the manipulated variable. The actual output value must be written back to this variable.
Km	Model gain	REAL	Follows data type	1	An algorithm parameter that describes the response gain of the controlled object. It must be determined based on the influence of the output OP on the controlled object. The greater this influence, the larger the gain value.
Tm	Model time constant	REAL	[0, 10000]	1	An algorithm parameter that describes the response time of the controlled object. The greater the inertia of the controlled object, the slower the response, and the larger this value.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Dm	Model hysteresis	REAL	[0, 10000]	0	An algorithm parameter that represents the response hysteresis of the controlled object. The greater the hysteresis of the object to input changes, the larger this value.
Nmodel	Model length	DINT	[1, 200]	100	The model length is used to adjust the control effect. The larger this value, the longer the effective prediction horizon, and the further the future trends can be predicted.
DDVTrend	Future disturbance sequence	ARRAY [0..199] OF REAL	Follows data type	-	Disturbance sequence generated by the PFCs_dv_v2 component. When using this variable, pass its "DDVTrend" output to this variable. It is used to predict future disturbances so that the controller can take proactive action to suppress them.
PFCsConfig	Input parameter structure	PFCsConfig_v2	Follows data type	-	Input parameter configuration structure.
AuTuConfig	Auto-tuning parameter structure	AuTuConfig_v2	Follows data type	-	Auto-tuning parameter configuration structure.
ReTeConfig	Relay test structure	ReTeConfig_v2	Follows data type	-	Relay test structure.
OPTConfig	Closed-loop OP test structure	OPTConfig_v2	Follows data type	-	Closed-loop OP test structure.

PFCsConfig_v2 structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Ts	Sampling cycle	REAL	(0, 10000]	1	This value represents the control and tuning cycle. The value must be greater than 0. Ensure that this value is not set too low (below the operation cycle). If Ts is less than 0, the algorithm will automatically terminate, the switch will turn off, and the output will be held.
Model	Model type	DINT	0, 1, 2	1	Model types include: 0: Non-integral. 1: Pseudo-integral. 2: Pure integral. **Note:** The algorithm internally distinguishes between non-integral and integral models. The logic for pseudo-integral models follows the same rules as those for integral models. When the model selection is not 0, 1, or 2, the algorithm terminates, the switch is turned off, and the output will be held.
COF	Filter coefficient	REAL	[0, 1]	0.9	An algorithm parameter that affects the performance of feedback correction. The feedback value balances the measurement for the current cycle with the model prediction. The magnitude of the COF determines whether the algorithm places greater weight on the measured value or the predicted value in the feedback loop. • When set to 0, the feedback value uses the model prediction, and no feedback correction is applied. • When set to 1, the feedback value uses the measured value. • For other values, an average is calculated.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
COP	Performance coefficient	REAL	(0, 10000]	1	An algorithm parameter that affects the control performance. Increasing the value speeds up the system response to some extent, while decreasing it slows down the system response to some extent.
Tref	Closed-loop reference time	REAL	(0, 10000]	1	An algorithm parameter that affects the control performance. It is used to reduce the system response speed. Increasing this value slows down the system response to some extent, while decreasing it will speed up the system response to some extent. If the value exceeds the limit, it will be capped at 1e-6.
I_COF	Integral deviation correction coefficient	REAL	[0, 1]	0.1	Correction coefficient for integral speed. This value will take effect when the integral model is used. The larger the value, the faster the integral model speed operates in the algorithm, resulting in greater changes in future prediction, and thus causing output changes. Generally, fine-tuning this value after reaching steady state can reduce steady-state fluctuations.
DOPmax	Output increment limit	REAL	>=0	100	Sets the upper limit of output increment.
OP_Low_Val	Output valid lower limit	REAL	Follows data type	0	When the value falls below this output valid lower limit, the switch will be turned off and the control will be disabled.
OP_High_Val	Output valid upper limit	REAL	Follows data type	100	When the value exceeds this output valid upper limit, the switch will be turned off and the control will be disabled.
OP_Low_Ope	Output operation lower limit	REAL	Follows data type	0	When the value falls below this output operation lower limit, a warning is issued and the output is clamped to this limit.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
OP_High_Ope	Output operation upper limit	REAL	Follows data type	100	When the value exceeds this output operation upper limit, a warning will be issued and the output is clamped to this limit.
SP_Low_Val	Setpoint valid lower limit	REAL	Follows data type	0	Valid lower limit of the setpoint. When the value falls below this lower limit, the switch will be turned off and the control will be disabled.
SP_High_Val	Setpoint valid upper limit	REAL	Follows data type	100	Valid upper limit of the setpoint. When the value exceeds this upper limit, the switch will be turned off and the control will be disabled.
SP_Low_Ope	Setpoint operation lower limit	REAL	Follows data type	0	Operation lower limit of the setpoint. When the value falls below this lower limit, the system will display a warning and limit the value to the lower limit.
SP_High_Ope	Setpoint operation upper limit	REAL	Follows data type	100	Operation upper limit of the setpoint. When the value exceeds this upper limit, the system will display a warning and limit the value to the upper limit.
SP_DMax	Setpoint increment upper range	REAL	>=0	0.1	Algorithm parameter that affects the control effect. The system is considered stable when PV is within the range SP + SP_DMax.
SP_DMin	Setpoint increment lower range	REAL	<=0	-0.1	Algorithm parameter that affects the control effect. The system is considered stable when PV is within the range SP + SP_DMin.
Cost_SS	Speed steady state weight	REAL	>=0	0	In an integral model, this weight determines how quickly the speed converges to steady state. A larger value slows down the convergence. If set too large, the output may remain unchanged.

AuTuConfig_v2 structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
AT_Start	Auto-tuning switch	DINT	0, 1	0	Auto-tuning switch. If this function is used, set AT_Start to 1.
AT_Model	Tuning model type	DINT	Follows data type	0	Tuning model types include: 0: Non-integral. 1: Pseudo-integral. 2: Pure integral. The auto-tuning function outputs the corresponding model parameters according to the model type. It is recommended to select according to the actual characteristics of the controlled object.
AT_OPInit	Initial OP output for tuning	REAL	Follows data type	0	OP output value at initial steady state. The controlled object must reach a stable state at this output.
AT_OPTarget	Target OP output for tuning	REAL	Follows data type	100	This value is the step output value at the start of tuning. The controller will perform a step change from "AT_OPInit" to "AT_OPTarget" to excite the response of the controlled object.
AT_PVTarget	PV target value at the end of tuning	REAL	Follows data type	100	This value is the target PV value to be achieved during tuning. The tuning process completes when the measured value (PV) of the controlled object reaches this value after tuning begin. Then, the tuning result will be output.
AT_PVDeltTar	Advance end threshold	REAL	Follows data type	0	When the PV value reaches AT_PVTarget – AT_PVDeltTar, the tuning process terminates early and outputs the tuning result.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
AT_InitRate	Initial steady-state slope	REAL	Follows data type	0	PV measured value slope that allows the step response tuning to start.

ReTeConfig_v2 structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
RT_Start	Relay test switch	DINT	0, 1	0	If this function is used, set RT _Start to 1.
RT_Direction	Input-output action direction	DINT	0, 1	0	0: Direct acting. When the output increases, the measured value increases. 1: Reverse acting. When the output increases, the measured value decreases.
RT_OP_Init	Initial value for the test	REAL	Follows data type	0	OP output at the start of the test.
RT_OP_DLow	Test output lower range	REAL	< 0	-1	Lower limit of the OP oscillation output at the start of the test = RT_OP_Init + RT_OP_DLow.
RT_OP_DHigh	Test output upper range	REAL	> 0	1	Upper limit of the OP oscillation output at the start of the test = RT_OP_Init + RT_OP_DHigh.
RT_PV_Target	Step measured value	REAL	Follows data type	100	Target PV measured value after the test starts.
RT_PV_DLow	Oscillation lower range	REAL	< 0	-1	Lower limit of PV oscillation switching after the test starts = RT_PV_Target + RT_PV_DLow.
RT_PV_DHigh	Oscillation upper range	REAL	> 0	1	Upper limit of PV oscillation switching after the test starts = RT_PV_Target + RT_PV_DHigh.
RT_Warn_Low	Warning lower range	REAL	Follows data type	0	Test protection lower range. If the value falls below this range, the test stops. Restart the test after reconfiguring the parameters.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
RT_Warn_High	Warning upper range	REAL	Follows data type	100	Test protection upper range. If the value exceeds this range, the test stops. Restart the test after reconfiguring the parameters.

OPTConfig_v2 structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
OPT_Start	Closed-loop OP test switch	DINT	0, 1	0	If this function is used, set OPT_Start to 1.
OPT_Type	PRBS type	DINT	0, 1, 2, 3	0	Type mode: 0: PRBS7 1: PRBS15 2: PRBS23 3: PRBS31
OPT_TS	PRBS cycle	DINT	>=1	1	Integer multiple of the task call cycle.
OPT_Amp	PRBS amplitude	REAL	>=0	0	Amplitude of the superimposed PRBS cycle during closed-loop testing.
OPT_Seed	PRBS initial seed	DINT	[1, 65535]	100	The seed value determines the initial value of the generated sequence.
OPT_Warn_Low	Warning lower range	REAL	Follows data type	0	Test protection lower range. If the value falls below this range, the test stops. Restart the test after reconfiguring the parameters.
OPT_Warn_High	Warning upper range	REAL	Follows data type	100	Test protection upper range. If the value exceeds this range, the test stops. Restart the test after reconfiguring the parameters.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
State	Switch state	DINT	0, 1	-	Indicates the actual state of the controller. 0: Normal control. 1: Switched to feedforward.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ERR	Fault code	DINT	Follows data type	-	0 for normal operation; non-zero when an error occurs. See PFCs_v2 fault codes.
OP	Valve position output value	REAL	Follows data type	-	Controller output value.
AT_ST	Tuning result	DINT	0, 1, 2, 3	-	Tuning state display. 0: Not tuned 1: Coarse tuning in progress 2: Fine tuning in progress 3: Tuning completed
PFCsOutput	Output parameter structure	PFCsOutput_v2	Follows data type	-	Output parameter configuration structure.

PFCsOutput_v2 structure configuration

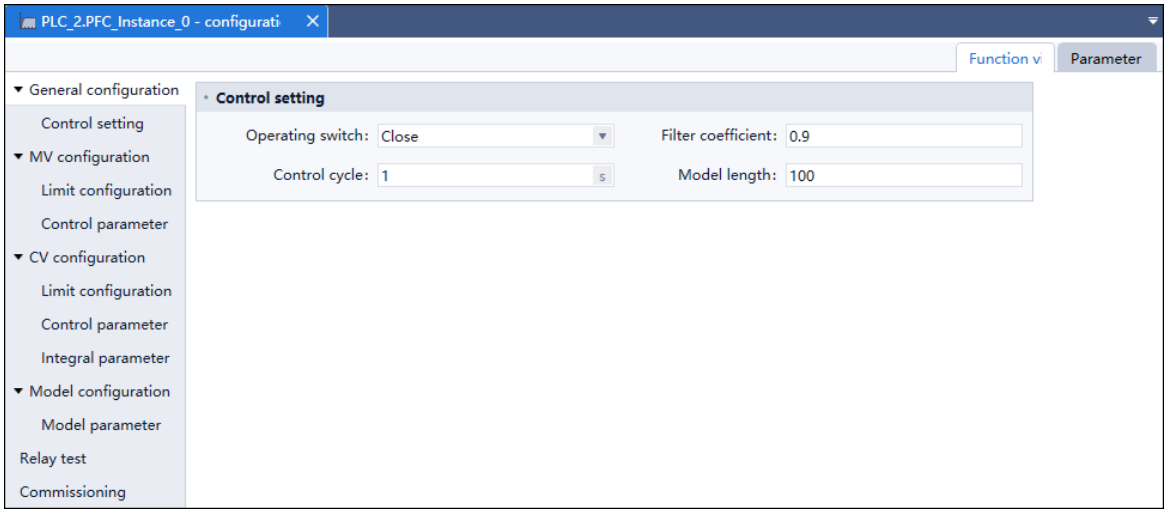
Output Variable	Name	Data Type	Valid Range	Default Value	Description
Version	Version information	ARRAY [0..3] OF DINT	Follows data type	-	Used to record internal version information.
DOP	Valve position output increment	REAL	Follows data type	-	Increment of the output value OP.
PV_Pred_t	Estimated value	REAL	Follows data type	-	Estimated value, that is, the response value of the controlled object, calculated by the internal model using historical data.
PV_Pred_tf	Predicted value	REAL	Follows data type	-	Predicted value, that is, the estimated value corrected based on the actual measured value PV.
PV_Pred_e	Prediction error	REAL	Follows data type	-	Difference between the actual measured value PV and the predicted value.
PV_Pred_eSum	Cumulative prediction error	REAL	Follows data type	-	Cumulative sum of prediction errors.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
DCV_S	Integral deviation speed.	REAL	Follows data type	-	Integral deviation speed.
PV_Pred	Open-loop prediction sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of future measured values predicted when the current cycle is not applied.
Tref_Step	Reference trajectory sequence	ARRAY [0..199] OF REAL	Follows data type	-	A trajectory sequence generated by the Tref parameter.
OP_Low_Ope_Past	Output lower limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical operation lower limits for the controller output OP.
OP_High_Ope_Past	Output upper limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical operation upper limits for the controller output OP.
OP_Low_Ope_Future	Output lower limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of predicted operation lower limits for the controller output at future sampling instants, computed after the current cycle has been applied.
OP_High_Ope_Future	Output upper limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of predicted operation upper limits for the controller output at future sampling instants, computed after the current cycle has been applied.
SP_Past	Setpoint history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of the historical setpoints.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
SP_Future	Setpoint future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of future setpoints predicted by the algorithm after the current cycle has been applied.
SP_Low_Ope_Past	Setpoint lower limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of historical operation lower limits for the setpoint.
SP_High_Ope_Past	Setpoint upper limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of historical operation upper limits for the setpoint.
SP_Low_Ope_Future	Setpoint lower limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of predicted operation lower limits for the setpoint at future sampling instants, computed after the current control cycle has been applied.
SP_High_Ope_Future	Setpoint upper limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of predicted operation upper limits for the setpoint at future sampling instants, computed after the current control cycle has been applied.
SP_Href	Upper reference trajectory sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the upper reference trajectory of the setpoint.
SP_Lref	Lower reference trajectory sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the lower reference trajectory of the setpoint.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
PV_Past	Measured value history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of the historical measured values.
OP_Past	Output value history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of the historical output values.
OP_Future	Output value future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of future output values predicted by the algorithm after the current cycle has been applied.
AT_Km	Tuning parameter Km	REAL	Follows data type	-	Model gain result obtained from auto-tuning.
AT_Tm	Tuning parameter Tm	REAL	> 0	-	Model time constant result obtained from auto-tuning.
AT_Dm	Tuning parameter Dm	REAL	>=0	-	Model hysteresis parameter result obtained from auto-tuning.

- **Function**
  - PFCs_v2 is a simplified 1x1 single-input, single-output predictive function controllers. It includes one manipulated variable (output via OP) and one controlled variable (measured via PV), and support a feedforward input channel (for use with the PFCs_dv function block). The PFCs_v2 component features a built-in auto-tuning function to obtain model parameters of the controlled object. This component is primarily used for rapid deployment in single-input single-output processes.
  - The PFCs_v2 function block provide easy-to-use interfaces for parameter configuration and tuning/commissioning. After adding the PFCs_v2 component, select "Control object > PFC" in the left project tree, and then double-click the PFCs_v2 instance under the PFC node to open the configuration and tuning interface, as shown in the figure below.

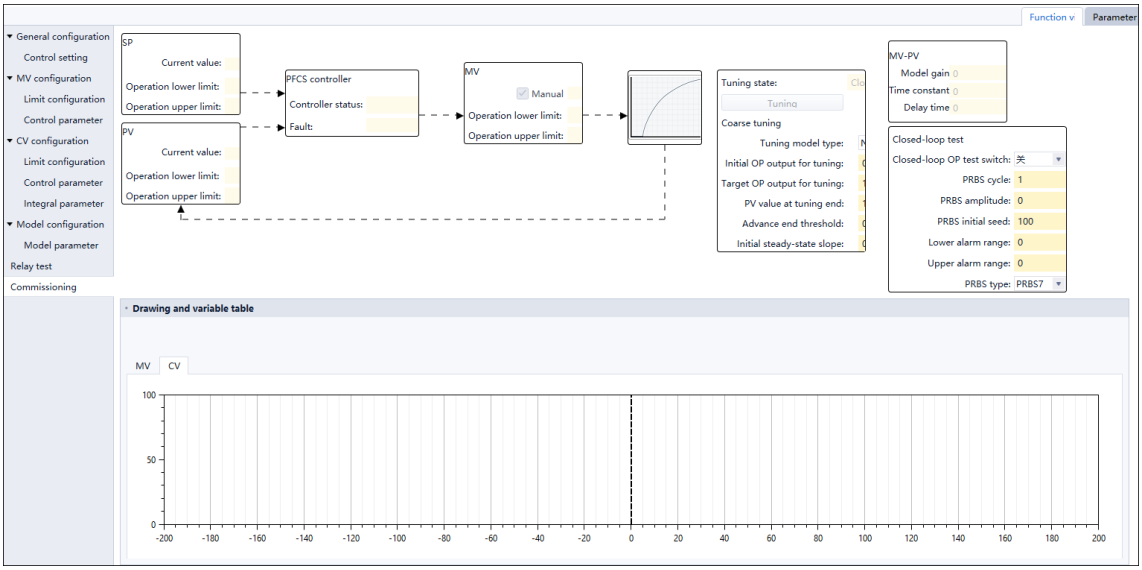


- Parameter configuration

  The parameter configuration of the PFCs_v2 instruction includes general configuration, MV configuration, CV configuration, model configuration and relay test. For detailed parameter descriptions, see Relevant variables.

- Parameter auto-tuning

  The PFCs_v2 instruction includes a built-in parameter auto-tuning function, automatically obtaining the model parameters (Km, Tm, and Dm) of the controlled object. Click the "Tuning" button on the "Commissioning" page to start parameter tuning.

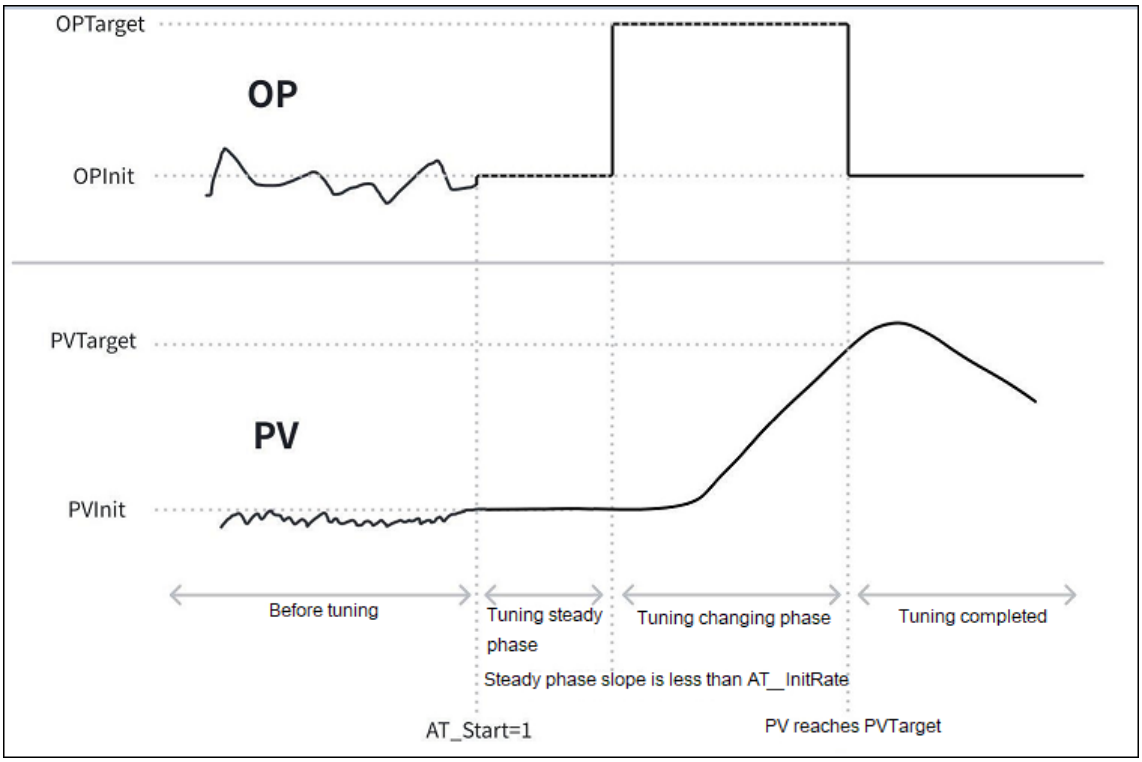


Because the controller internal variable array has a capacity of only 200, ensure that the duration from the OP output step to the completion of tuning stays within 200 control cycles. 100 to 150 cycles is recommended. An excessively long or short tuning process may lead to tuning failure. Furthermore, the tuning process does not require a complete steady state after the step. Generally, it only needs to capture the portion of the controlled object's response with the maximum slope.

The tuning process is as follows:

1. Select the model type of the tuning object.
2. Configure the tuning parameter "Initial OP output for tuning" (AT_OPInit). AT_OPInit is the OP output value at the initial steady state. Ensure that the controlled object reaches a basic stable state at this output. Open-loop control can be used before the tuning process to stabilize the controlled object at this output, then switch to auto-tuning. The initial steady state is critical for the tuning results. When it is difficult to achieve the initial steady state, the "Initial steady-state slope" (AT_InitRate) parameter can be used to appropriately relax the steady-state requirements.

3. Configure the tuning parameter "Target OP output for tuning" (AT_OPTarget). This value is the step output value at the start of tuning. The controller will perform a step change from "AT_OPInit" to "AT_OPTarget" to excite the response of the controlled object.
4. Configure the tuning parameter "PV value at tuning end" (AT_PVTarget) and "Advance end threshold" (AT_PVDeltTar). The tuning process completes when the measured value (PV) of the controlled object reaches this value after tuning begin. Pay special attention to the response polarity of the controlled object. Ensure that the measured value reaches AT_PVTarget after tuning starts, rather than the PV changing in the opposite direction after the OP step change.
5. Click the "Tuning" button.
6. After tuning ends, click "Confirm" or "Cancel" to select parameters, and automatically switch to control mode.

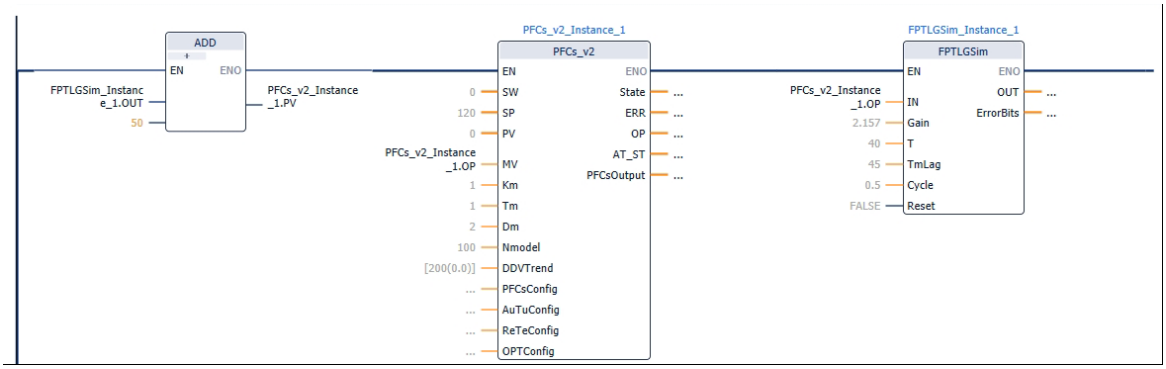


- Program example

  Here, the control simulation between the PFCs_v2 instruction and the first-order inertia plus pure delay simulation instruction (FPTLGsim) is used as an example. The parameter configuration and control process are as follows:

  1. Parameter configuration
     Configure the PFCs_v2 instruction parameters based on the characteristics of the control object (Model parameters: Km, Tm, Dm and so on; controller parameters: COP, COF, etc.).
  2. Control process
     The PFCs_v2 instruction responds to the output (PV) of the FPTLGsim object, obtains the control quantity (OP) output, and applies it to the original control object, forming a control loop.
  3. Control expectation
     If all parameters are configured properly, the controller will eventually keep the output value (PV) of the controlled object near the setpoint (SP).

  LD language code example:

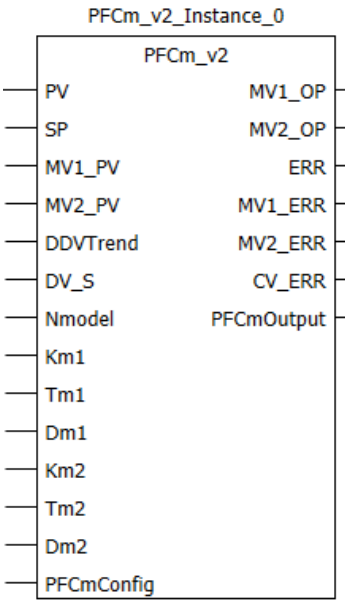


- Precautions
  - The module execution cycle must be consistent with the function block cycle parameter Ts.
  - During operation, ensure that the switch is on, and the configuration is correct.
  - The execution time of the PFCs_v2 instruction is 10 ms. With 50 PFCs_v2 function blocks loaded, the total execution time is estimated to reach 500 ms. During operation, configure the scheduling task cycle properly to avoid high CPU load.

18.3.2.6 PFCm_v2

This instruction is a single-input dual-output control instruction. It is primarily used in 2x1 scenarios, such as regulating temperature via heating and cooling. This version is v2, with updates to several functions, including constrained solution handling, integral/non-integral decomposition, and speed steady state.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PFCm_v2	Multivariable advanced control V2	FB		PFCm_v2_Instance_0(PV := , SP := , MV1_PV := , MV2_PV := , DDVTrend := , DV_S := , Nmodel := , Km1 := , Tm1 := , Dm1 := , Km2 := , Tm2 := , Dm2 := , PFCmConfig := , MV1_OP => , MV2_OP => , ERR => , MV1_ERR => , MV2_ERR => , CV_ERR => , PFCmOutput =>);

- **Related variables**

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
PV	Measured value of the controlled variable	REAL	Follows data type	0	Input value of the controlled variable measurement, which must be provided by an external physical object or simulation object.
SP	Controlled variable setpoint value	REAL	Follows data type	0	Setpoint value of the controlled variable, which determines the control objective of the controller so that the final actual control result (PV) must closely follow the SP.
MV1_PV	Measured value of the manipulated variable 1	REAL	Follows data type	0	Measured value of manipulated variable 1. The actual output value 1 must be written back to this variable. In a heating application: if only algorithmic control is active, write back the value of MV1_OP; if an external input is present, write back the actual value sent to the actuator.
MV2_PV	Measured value of the manipulated variable 2	REAL	Follows data type	0	Measured value of manipulated variable 2. The actual output value 2 must be written back to this variable. In a heating application: if only algorithmic control is active, write back the value of MV2_OP; if an external input is present, write back the actual value sent to the actuator.
DDVTrend	Disturbance sequence	ARRAY [0..199] OF REAL	Follows data type	-	Disturbance sequence generated by the dv series component. When using this variable, pass its output "DDVTrend" to this variable. It is used to predict future disturbances so that the controller can take proactive action to suppress them.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
DV_S	Disturbance velocity	REAL	Follows data type	0	Disturbance velocity generated by the PFCm_dv component. When using this variable, pass its output DV_S to this variable. It is used to predict future disturbances so that the controller can take proactive action to suppress them.
Nmodel	Model length	DINT	[1, 200]	100	The model length is used to adjust the control effect. The larger this value, the longer the effective prediction horizon, and the further the future trends can be predicted.
Km1	Model 1 gain	REAL	Follows data type	1	An algorithm parameter that affects the control performance. Increasing the value speeds up the system response to some extent, while decreasing it slows down the system response to some extent.
Tm1	Model 1 time constant	REAL	(0, 10000]	1	An algorithm parameter that affects the control performance. The smaller Tm is, the fewer control cycles needed to reach steady state. When Tm exceeds the allowable limit, it is limited to 1e-6.
Dm1	Model 1 hysteresis	REAL	[0, 10000]	0	An algorithm parameter that affects the control performance. The delay expressed in control cycles is Dm/Ts. If the value exceeds the limit, it will be limited to 0.
Km2	Model 2 gain	REAL	Follows data type	1	Same as Km1; this is the gain for model 2.
Tm2	Model 2 time constant	REAL	(0, 10000]	1	Same as Tm1, this is the time constant for model 2. If the value exceeds the limit, it will be limited to 1e-6.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Dm2	Model 2 hysteresis	REAL	[0, 10000]	0	Same as Dm1; this is the hysteresis for model 2. If the value exceeds the limit, it will be limited to 0.
PFCmConfig	Input parameter structure	PFCmConfig_v2	Follows data type	-	Input parameter configuration structure.

PFCmConfig_v2 structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Ts	Sampling cycle	REAL	(0, 10000]	1	This value determines the control cycle. This value must be greater than 0, and can not be set too low. An excessively small value may cause a calculation timeout and degrade control performance. If Ts is less than 0, the algorithm will automatically terminate, the switch will turn off, and the output will be held.
Model1	Model type 1	DINT	0, 1, 2	1	Model for MV1-CV. Model type: 0: Non-integral 1: Pseudo-integral 2: Pure integral Currently, types 0 (Non-integral) and 1 (Pseudo-integral) are primarily used. Type 2 (Pure integral) is not implemented. In practice, integral models are typically handled as pseudo-integral. When the model selection is not 0, 1, or 2, the algorithm terminates, the switch is turned off, and the output will be held.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Model2	Model type 2	DINT	0, 1, 2	1	Model for MV2-CV. Model type: 0: Non-integral 1: Pseudo-integral 2: Pure integral Currently, types 0 (Non-integral) and 1 (Pseudo-integral) are primarily used. Type 2 (Pure integral) is not implemented. In practice, integral models are typically handled as pseudo-integral. When the model selection is not 0, 1, or 2, the algorithm terminates, the switch is turned off, and the output will be held.
Cost_SS	Speed steady state weight	REAL	[0, 10000]	0	In an integral model, this weight determines how quickly the speed converges to steady state. A larger value slows down the convergence. If set too large, the output may remain unchanged.
COF	Filter coefficient	REAL	[0, 1]	0.9	An algorithm parameter that affects the control performance. The smaller this value, the greater the confidence in the model prediction. When set to 0, no feedback correction is performed. Use a value greater than 0. If this value exceeds the upper limit, it will be limited to the upper limit. If the value falls below the lower limit, it will be limited to the lower limit.
Tref	Closed-loop reference time	REAL	(0, 10000]	1	An algorithm parameter that affects the control performance. Increasing the value slows down the system response to some extent, while decreasing it speeds up the system response to some extent. If the value exceeds the limit, it will be limited to 1e-6.
I_COF	Integral deviation correction coefficient	REAL	[0, 1]	0.1	An algorithm parameter that affects the control performance. The smaller the value, the greater the confidence in the model prediction. If this value exceeds the upper limit, it will be limited to the upper limit. If the value falls below the lower limit, it will be limited to the lower limit.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
MV1_Config	MV1 variable input parameter structure	MVConfig_v2	Follows data type	-	MV1 variable input parameter configuration structure.
MV2_Config	MV2 variable input parameter structure	MVConfig_v2	Follows data type	-	MV2 variable input parameter configuration structure.
Single_SW	Single output switch	DINT	0, 1	1	Single output switch. If this switch is set to ON, it ensures that when the two MV gains have opposite signs, only one output direction is active. This is typically used to save energy.
CV_SW	Controlled variable switch	DINT	0, 1	1	Controlled variable switch. When the switch is off, the control is disabled. When used, it must be set to 1. Invalid values are automatically set to 0.
CV_SP_DMax	Upper deviation range	REAL	[0, 10000]	0.1	An algorithm parameter that affects the control performance. The system is considered stable if CV_PV is within this range CV_SP + CV_SP_DMax. Invalid input values are limited to 0.
CV_SP_DMin	Lower deviation range	REAL	[-10000, 0]	-0.1	An algorithm parameter that affects the control performance. The system is considered stable if CV_PV is within the range CV_SP + CV_SP_Dmin. Invalid input values are limited to 0.
DeltaCV	Maximum change rate of the controlled variable	REAL	[0, 10000]	100	This algorithm parameter defines the maximum change value of CV during prediction. Invalid values are limited to 100.
CV_Low_Val	Valid lower limit of the controlled variable	REAL	Follows data type	0	Valid lower limit. When the value falls below the valid lower limit, the switch will be turned off and the control will be disabled.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
CV_High_Val	Valid upper limit of the controlled variable	REAL	Follows data type	100	Valid upper limit. When the value exceeds the valid upper limit, the switch will be turned off and the control will be disabled.
CV_Low_Ope	Operation lower limit of the controlled variable	REAL	Follows data type	0	Operation lower limit. When the value falls below this operation lower limit, a warning will be issued and the value will be clamped to this operation lower limit.
CV_High_Ope	Operation upper limit of the controlled variable	REAL	Follows data type	100	Operation upper limit. When the value exceeds this operation upper limit, a warning will be issued and the value will be clamped to this operation upper limit.

MVConfig_v2 structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
MV_SW	Manipulated variable switch	DINT	0, 1	1	MV switch; when this switch is off, the control is disabled. Invalid values are automatically set to 0.
MV_High_Val	Manipulated variable valid upper limit	REAL	Follows data type	200	Valid upper limit. When the value exceeds the valid upper limit, the switch will be turned off and the control will be disabled.
MV_Low_Val	Manipulated variable valid lower limit	REAL	Follows data type	0	Valid lower limit. When the value falls below the valid lower limit, the switch will be turned off and the control will be disabled.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
MV_High_Ope	Operation upper limit of the manipulated variable	REAL	Follows data type	100	Operation upper limit. When the value exceeds this operation upper limit, a warning will be issued and the value will be clamped to this operation upper limit.
MV_Low_Ope	Operation lower limit of the manipulated variable	REAL	Follows data type	0	Operation lower limit. When the value falls below this operation lower limit, a warning will be issued and the value will be clamped to this operation lower limit.
MV_DMax	Maximum positive increment for the manipulated variable	REAL	[0, 10000]	100	Defines the maximum allowable positive increment of the output. Invalid input values are limited to 100.
MV_DMin	Maximum negative increment for the manipulated variable	REAL	[-10000, 0]	-100	Defines the maximum allowed negative increment of the output. Invalid input values are limited to -100.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
MV_COP	Manipulated variable performance coefficient	REAL	[0, 10000]	1	An algorithm parameter that affects the control performance. Decreasing the value speeds up the system response to some extent, while increasing it slows down the system response to some extent. If this value exceeds the upper limit, it will be limited to the upper limit. If the value falls below the lower limit, it will be limited to the lower limit.
MV_Irv	Ideal injection value for the manipulated variable	REAL	Follows data type	1	The value that the corresponding MV output eventually converges to.
MV_Irv_C	Weight for manipulated variable ideal injection value	REAL	[0, 10000]	1	The larger the value, the closer the MV will converge to MV_Irv.
MV_Irv_SW	Ideal injection value switch	DINT	0, 1	0	When set to 0, the ideal injection value is inactive. When used, the value must be set to 1. Invalid values are automatically set to 0.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
MV1_OP	Manipulated variable 1 output value	REAL	Follows data type	-	Output value of manipulated variable 1. This value must be output to the simulation object or actual object.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
MV2_OP	Manipulated variable 2 output value	REAL	Follows data type	-	Output value of manipulated variable 2. This value must be output to the simulation object or actual object.
ERR	Fault code	DINT	Follows data type	-	For fault codes, see PFCm_v2 fault codes.
MV1_ERR	Manipulated variable 1 fault code	DINT	Follows data type	-	Fault code of manipulated variable 1.
MV2_ERR	Manipulated variable 2 fault code	DINT	Follows data type	-	Fault code of manipulated variable 2.
CV_ERR	Controlled variable fault code	DINT	Follows data type	-	Fault code of controlled variable.
PFCmOutput	Output parameter structure	PFCmOutput_v2	Follows data type	-	Output parameter structure.

PFCmOutput_v2 structure configuration

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Version	Version information	ARRAY [0..3] OF DINT	Follows data type	-	Used to obtain the version information.
CV_Pred_t	Estimated value of controlled variable	REAL	Follows data type	-	Estimated value, that is, the response value of the controlled object, calculated by the internal model using historical data.
CV_Pred_tf	Predicted value of controlled variable	REAL	Follows data type	-	Predicted value, that is, the corrected predicted value based on the predicted value and the actual measured value PV.
DCV_S_i	Velocity after integral deviation correction	REAL	Follows data type	-	Velocity after integral deviation correction.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
DCV_S_SS	Future speed steady state value	REAL	Follows data type	-	Steady-state speed value, which is the speed value predicted after 200 cycles.
CV_PV_e	Predicted error of the controlled variable	REAL	Follows data type	-	Difference between the actual measured value and the predicted value.
CV_PV_eSum	Cumulative sum of predicted errors for the controlled variable	REAL	Follows data type	-	The cumulative sum of errors (regardless of sign). Its magnitude indicates the degree of model mismatch.
NFlag	Number of errors	DINT	[0, 10]	-	Counts the number of errors.
Flag	Error information	ARRAY [0..9] OF DINT	Follows data type	-	Records detailed information of each error.
MV1_OutConfig	Manipulated variable 1 output structure	MVOutConfig_v2	Follows data type	-	Manipulated variable 1 output structure
MV2_OutConfig	Manipulated variable 2 output structure	MVOutConfig_v2	Follows data type	-	Manipulated variable 2 output structure
CV_NFlag	Controlled variable error count	DINT	[0, 10]	-	Number of recorded errors for the controlled variable
CV_Flag	Controlled variable error information	ARRAY [0..9] OF DINT	Follows data type	-	Detailed information of recorded errors for the controlled variable
CV_ST	Controlled variable switch status	DINT	0, 1	-	Controlled variable switch state. Initially the value equals CV_SW. When an internal error occurs, this value is set to 1.
CV_Pred_t_Past	Sequence of historical estimated values for the controlled variable	ARRAY [0..199] OF REAL	Follows data type	-	Records historical values.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
CV_Pred	Open-loop prediction sequence of controlled variables	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of future values predicted when the current cycle is not applied.
CV_Future	Closed-loop prediction sequence of controlled variables	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of future values predicted after the current cycle has been applied.
CV_PV_Past	Sequence of historical measured values for the controlled variable	ARRAY [0..199] OF REAL	Follows data type	-	Records historical measured values.
SP_Past	Setpoint history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical setpoint values, for plotting.
SP_Future	Setpoint future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of the future setpoint values, for plotting.
CV_SP_Href	Upper reference trajectory sequence	ARRAY [0..199] OF REAL	Follows data type	-	Predicted upper trajectory to be approached.
CV_SP_Lref	Lower reference trajectory sequence	ARRAY [0..199] OF REAL	Follows data type	-	Predicted lower trajectory to be approached.
CV_Low_Past	Controlled variable lower limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical operation lower limits for the controlled variable, for plotting.
CV_High_Past	Controlled variable upper limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical upper limits for the controlled variable, for plotting.
CV_Low_Future	Controlled variable lower limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future operation lower limits for the controlled variable, for plotting.

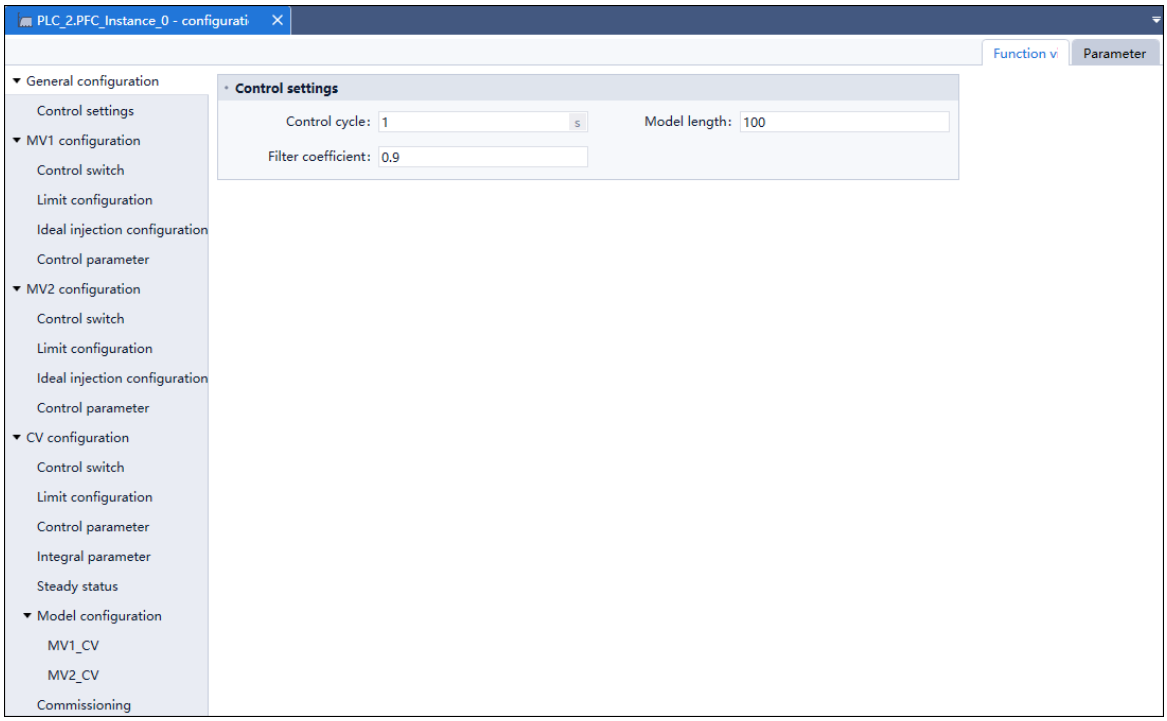
Output Variable	Name	Data Type	Valid Range	Default Value	Description
CV_High_Future	Controlled variable upper limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future operation upper limits for the controlled variable, for plotting.

MVOutConfig_v2 structure configuration

Output Variable	Name	Data Type	Valid Range	Default Value	Description
MV_NFlag	MV error count	DINT	0, 1	-	Number of errors counted for the manipulated variable.
MV_Flag	MV error message	ARRAY [0..9] OF DINT	[0, 10]	-	Detailed information of recorded errors for the manipulated variable.
MV_Low_Past	Output lower limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical operation lower limits for the manipulated variable, for plotting.
MV_High_Past	Output upper limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical operation upper limits for the manipulated variable, for plotting.
MV_Low_Future	Output lower limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future operation lower limits for the manipulated variable, for plotting.
MV_High_Future	Output upper limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future operation upper limits for the manipulated variable, for plotting.
MV_Past	Output value history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical measured values for the manipulated variable, for plotting.
MV_Future	Output value future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future measured values for the manipulated variable, for plotting.
MV_DOP_Past	Historical increment sequence of the valve position	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical increment values for the manipulated variable, for plotting.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
MV_DOP	Manipulated variable action increment	REAL	Follows data type	-	Increment value of the manipulated variable.
MV_ST	Manipulated variable switch state	DINT	0, 1	-	Manipulated variable switch state. Initially, it equals the switch value. This value is set to 1 when an internal error occurs.
MV_Irv_Past	Ideal injection value history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical ideal injection values, for plotting.
MV_Irv_Future	Ideal injection value future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future ideal injection values, for plotting.

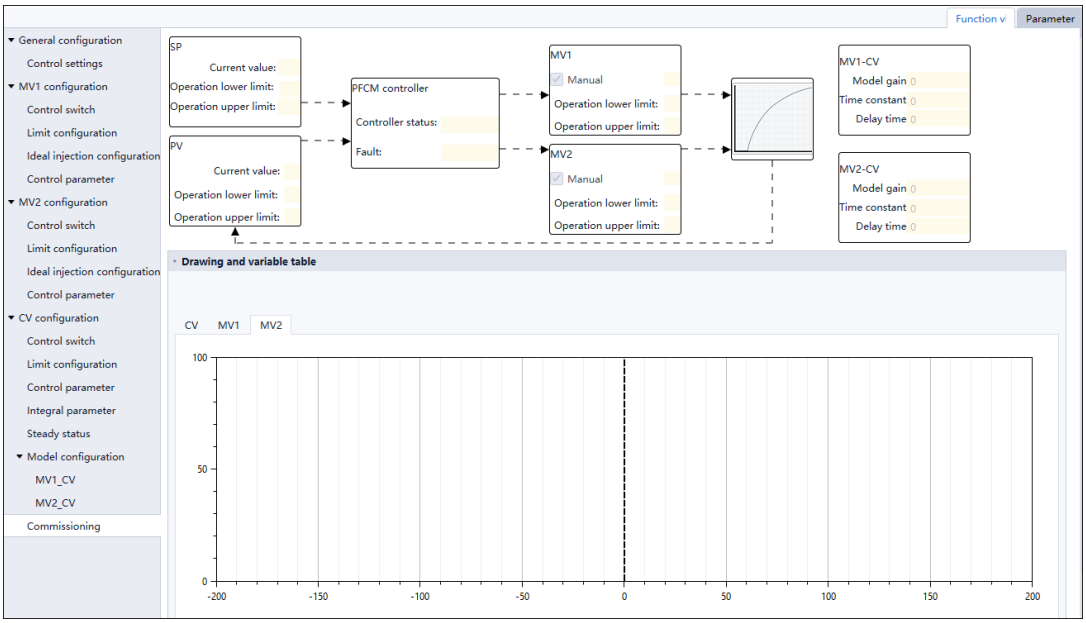
- **Function**
  - PFCm_v2 is a 1×2 single-input, dual-output control instruction. The controlled object can support non-integrating, integrating, or pseudo-integrating responses. The instruction includes two manipulated variables (MV), one controlled variable (CV), and a feedforward input channel (for use with PFCm_dv). It does not rely on an external solver for optimization calculations. It is primarily intended for applications that require two manipulated variables but not high-end products. It is suitable for fast control application scenarios with millisecond-level control cycles.
  - The PFCm_v2 function block includes an intuitive interface for parameter configuration and commissioning. After adding the PFCm_v2 component, select "Control object > PFC" in the left project tree, and then double-click the PFCm_v2 instance under the PFC node to open the configuration and tuning interface, as shown in the figure below.



    - Parameter configuration

The parameter configuration of the PFCm_v2 instruction includes general configuration, MV configuration, CV configuration, and model configuration. For detailed parameter descriptions, see Relevant variables.

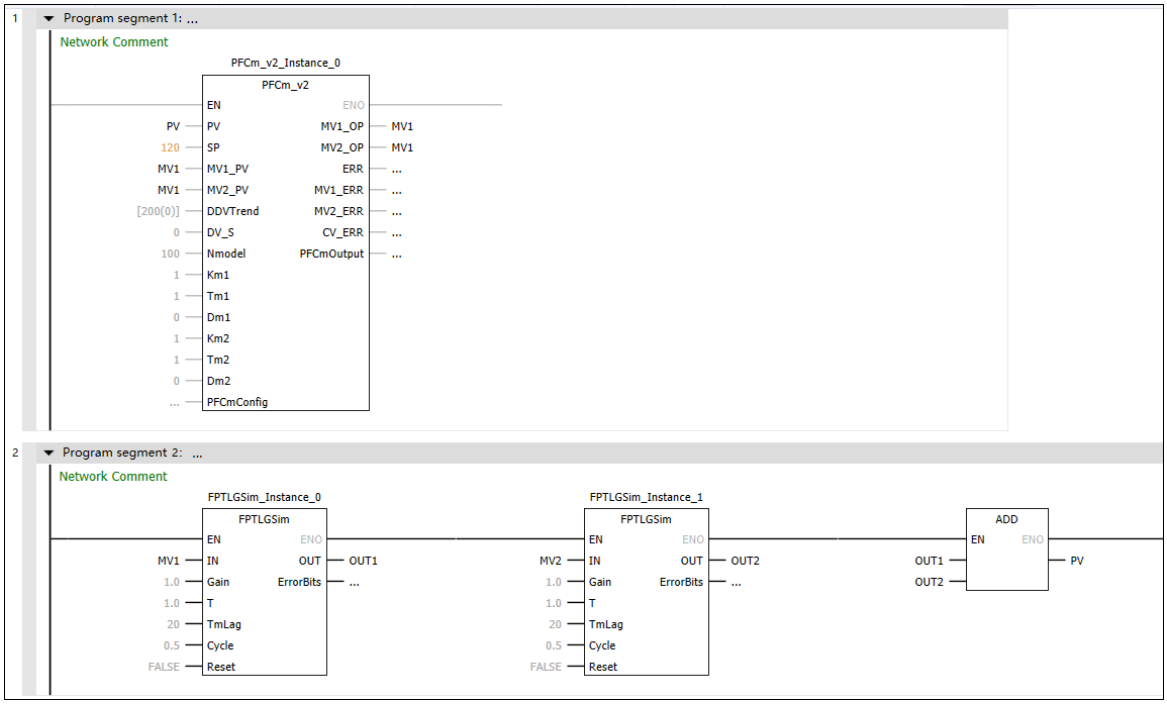
- Commissioning
  After the parameters are configured, the control effect can be viewed directly in the commissioning interface.



- **Program example**
  Here, the two-input, single-output control simulation between the PFCm_v2 instruction and two first-order inertia plus pure delay simulation instruction (FPTLGsim) is used as an example. The parameter configuration and control process are as follows:

  1. Parameter configuration
     Configure the PFCm_v2 instruction parameters based on the characteristics of the control object (Model parameters: Km1, Km2, Dm1 and so on; controller parameters: COP, COF, etc.).
  2. Control process
     The PFCm_v2 instruction responds to the output (PV) of the FPTLGsim object, obtains the control quantity (OP) output, and applies it to the original control object, forming a control loop.
  3. Control expectation
     If all parameters are configured properly, the controller will eventually keep the output value (PV) of the controlled object near the setpoint SP.

     LD language code example:



- **Precautions**
  - During operation, ensure that the switch is on, and the configuration is correct.
  - The execution time of the PFCm_v2 instruction is 20 ms. With 50 PFCm_v2 function blocks loaded, the total execution time is estimated to reach approximately 1 second. Loading 364 PFCm_v2 function blocks will reach the controller's resource limit. During operation, configure the scheduling task cycle properly. For example, if 10 PFCm_v2 are required to perform loop control simultaneously, with a block calculation time of 200 ms. The task cycle must be at least twice this value, that is, 400 ms, to ensure the CPU load does not become too high.

18.3.2.7 PFCs_dv_v2

The disturbance instruction outputs a predicted disturbance sequence and must be used in conjunction with a corresponding control instruction.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PFCs_dv_v2	Feedforward predictive control V2	FB	PFCs_dv_v2_Instance_0 PFCs_dv_v2 SW PV DVConfig Km Tm Dm ST ERR PV_t DDVTrend PFCdvOutput	PFCs_dv_v2_Instance_0(SW := , PV := , DVConfig := , Km := , Tm := , Dm := , ST => , ERR => , PV_t => , DDVTrend => , PFCdvOutput =>);

- **Related variables**

  Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
SW	Disturbance switch	DINT	0, 1	1	The disturbance switch disables the control when the switch is off. When used, it must be 1. Any invalid value is set to 0.
PV	Input value	REAL	Follows data type	0	Disturbance input value provided externally. Based on this input value, a corresponding disturbance sequence is generated.
DVConfig	Input parameter structure	DVConfig_v2	Follows data type	-	Input parameter configuration structure.
Km	Model gain	REAL	Follows data type	1	An algorithm parameter that affects the control performance. Increasing the value speeds up the system response to some extent, while decreasing it slows down the system response to some extent. It is used to predict the disturbance sequence.
Tm	Model time constant	REAL	(0, 10000]	1	An algorithm parameter that affects the control performance. The smaller the Tm value, the fewer cycles required to reach a steady state. It is used to predict the disturbance sequence. Any invalid value is set to 1e-6.
Dm	Model hysteresis	REAL	[0, 10000]	0	An algorithm parameter that affects the control performance. The delay expressed in control cycles is Dm/Ts. It is used to predict the disturbance sequence. Any invalid value is set to 0.

DVConfig_v2 structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Ts	Sampling cycle	REAL	(0, 10000]	1	This value determines the control cycle. This value must be greater than 0, and can not be set too low. An excessively small value may cause a calculation timeout and degrade control performance. If Ts is less than 0, the algorithm will automatically terminate, the switch will turn off, and the output will be held.
Model	Model type	DINT	0, 1, 2	1	Model type: 0: Non-integral. 1: Pseudo-integral. 2: Pure integral. Currently, types 0 (Non-integral) and 1 (Pseudo-integral) are primarily used. Type 2 (Pure integral) is not implemented. In practice, integral models are typically handled as pseudo-integral. **Note:** This parameter only serves as display effect in DV. When used, it must match the model type of the control algorithm component. If the model selection does not follow 0, 1, 2, it is set to 0 by default.
Low	Lower limit of the disturbance variable	REAL	Follows data type	0	Lower limit; if the value falls below the lower limit, it is clamped to the lower limit.
High	Upper limit of the disturbance variable	REAL	Follows data type	100	Upper limit; if the value exceeds the upper limit, it is clamped to the upper limit.
DMax	Increment limit	REAL	Follows data type	100	Increment limit; when exceeded, it will be clamped to the maximum increment value. Any invalid value is set to 0.

Output variable

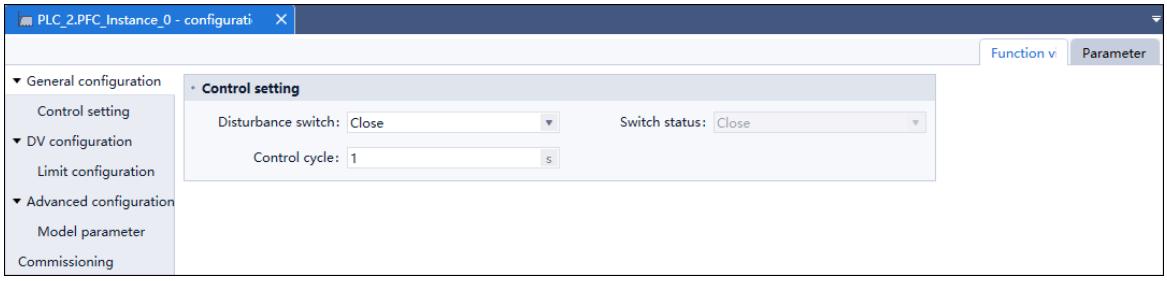
Output Variable	Name	Data Type	Valid Range	Default Value	Description
ST	Switch state	DINT	0, 1	-	Switch status. Initially the value equals the switch input value. When an internal error occurs, this value is set to 1.
ERR	Fault code	DINT	Follows data type	-	For fault codes, see PFCs_dv_v2 fault codes.
PV_t	Feedforward correction value	REAL	Follows data type	-	Feedforward value after limit.
DDVTrend	Disturbance sequence	ARRAY [0..199] OF REAL	Follows data type	-	Disturbance sequence. After generating this sequence, it must be output to the input interface with the same name on the control instruction for feedforward correction.
PFCdvOutput	Output parameter structure	PFCdvOutput_v2	Follows data type	-	Output parameter structure.

PFCdvOutput_v2 structure configuration

Output Variable	Name	Data Type	Valid Range	Default Value	Description
PV_Past	Measured value history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records historical values for the disturbance input.
PVt_Past	Feedforward value history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records historical values of the feedforward signal.
DV_Low_Past	Disturbance value lower limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical lower limits for the disturbance value, for plotting.
DV_High_Past	Disturbance value upper limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical upper limits for the disturbance value, for plotting.
DV_Low_Future	Disturbance value lower limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future lower limits for the disturbance value, for plotting.
DV_High_Future	Disturbance value upper limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future upper limits for the disturbance value, for plotting.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
DV_PV_Future	Disturbance value future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future disturbance values, for plotting.
DDV_Past	Feedforward value history increment sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical increments for the feedforward value, for plotting.
Version	Version information	ARRAY [0..3] OF DINT	Follows data type	-	Records the version information.

- **Function**
  - PFCs_dv_v2 is a disturbance feedforward prediction module. The module is an upgrade from version v1. It has the same functions as v1 and differs only in data length. Its primary function is to predict the future disturbance sequence of the controlled object based on the current disturbance input. The predicted disturbance sequence can be fed forward to the feedforward input channel of the PFCs/PFCm instruction, enabling proactive compensation of disturbances and improving disturbance rejection.
  - The PFCs_dv_v2 function block includes an intuitive interface for parameter configuration and commissioning. After adding the PFCs_dv_v2 component, select "Control object > PFC" in the left project tree, and then double-click the PFCs_dv_v2 instance under the PFC node to open the configuration and tuning interface, as shown in the figure below.

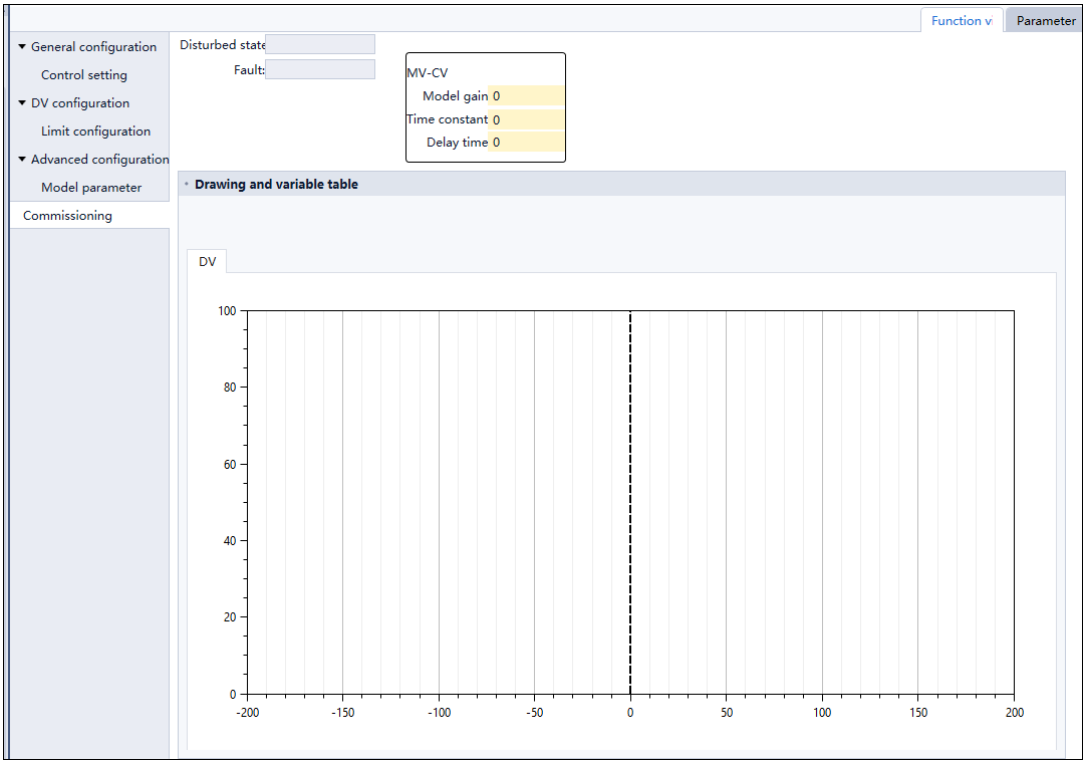


    - Parameter description

      The parameter configuration of the PFCs_dv_v2 instruction includes general configuration, DV configuration, and model configuration. For detailed parameter descriptions, see Relevant variables.

    - Commissioning

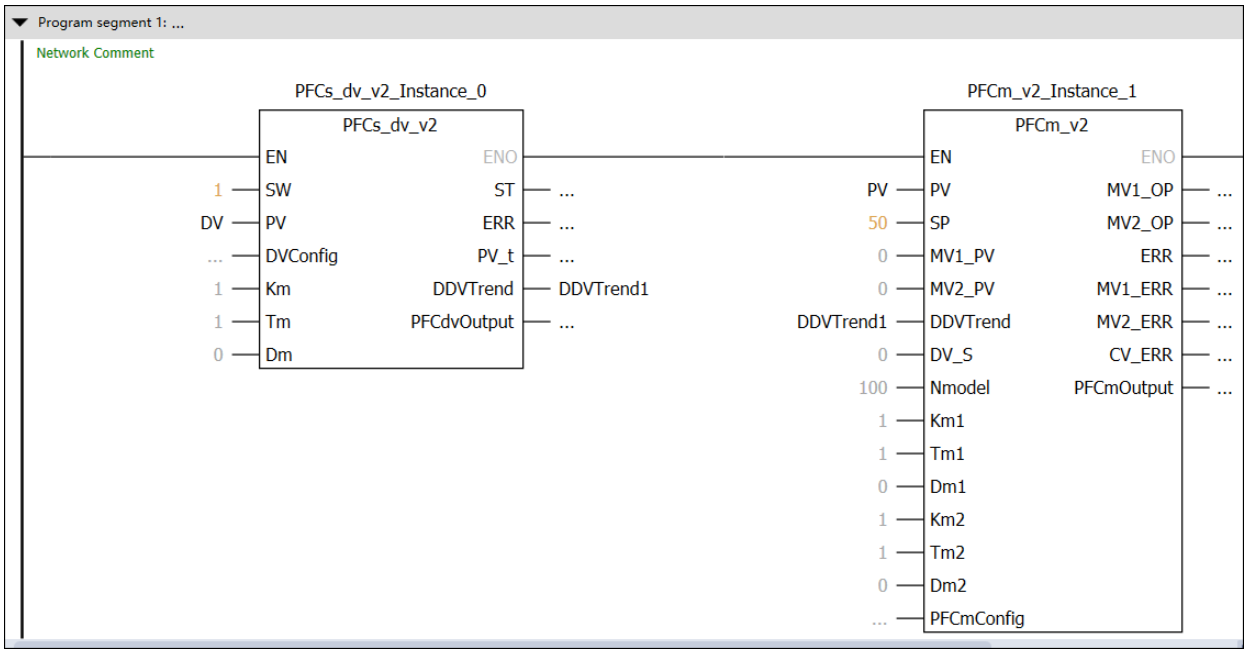
      After the parameters are configured, the historical and future disturbance value sequence can be viewed directly in the commissioning interface.



- **Program example**

  During operation, this instruction primarily feeds the sequence output by DDVTrend into the control component PFCm or PFCs. The generated sequence is determined by the model parameters Km, Tm, and Dm, as well as the input disturbance PV.

  LD:



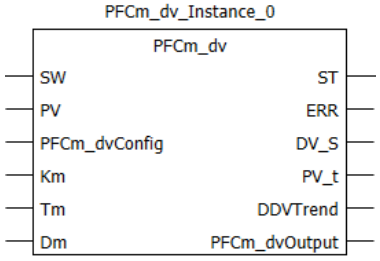
- **Precautions**
  - During operation, ensure that the switch is on, and the configuration is correct.
  - When setting the model parameters, ensure they are consistent with the controller parameter type, either all integral or non-integral.
  - The execution time of the PFCs_dv_v2 instruction is 8 ms. With 140 PFCs_dv_v2 function blocks loaded, the total execution time is estimated to reach approximately 1 second. Loading 1154 PFCs_dv_v2 function blocks will reach the controller's resource limit. During operation, configure the scheduling task cycle properly. For example, if 10 PFCs_dv_v2 are required to perform loop control simultaneously,

with a block calculation time of 80 ms. The task cycle must be at least twice this value, that is, 160 ms, to ensure the CPU load does not become too high.

18.3.2.8 PFCm_dv

The disturbance instruction outputs a predicted disturbance sequence and must be used in conjunction with a corresponding control instruction.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PFCm_dv	Feedforward predictive control M	FB		PFCm_dv_Instance_0(SW := , PV := , PFCm_dvConfig := , Km := , Tm := , Dm := , ST => , ERR => , DV_S => , PV_t => , DDVTrend => , PFCm_dvOutput =>);

- **Related variables**

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
SW	Disturbance switch	DINT	0, 1	1	The disturbance switch disables the control when the switch is off. When used, it must be 1. Any invalid value is set to 0.
PV	Input value	REAL	Follows data type	0	Disturbance input value provided externally. Based on this input value, a corresponding disturbance sequence is generated.
PFCm_dvConfig	Input parameter structure	PFCm_dvConfig	Follows data type	-	Input parameter configuration structure.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Km	Model gain	REAL	Follows data type	1	An algorithm parameter that affects the control performance. Increasing the value speeds up the system response to some extent, while decreasing it slows down the system response to some extent. It is used to predict the disturbance sequence.
Tm	Model time constant	REAL	(0, 10000]	1	An algorithm parameter that affects the control performance. The smaller the Tm value, the fewer cycles required to reach a steady state. It is used to predict the disturbance sequence. Any invalid value is set to 1e-6.
Dm	Model hysteresis	REAL	[0, 10000]	0	An algorithm parameter that affects the control performance. The delay expressed in control cycles is Dm/Ts. It is used to predict the disturbance sequence. Any invalid value is set to 0.

PFCm_dvConfig structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Ts	Sampling cycle	REAL	(0, 10000]	1	This value determines the control cycle. This value must be greater than 0, and can not be set too low. An excessively small value may cause a calculation timeout and degrade control performance. If Ts is less than 0, the algorithm will automatically terminate, the switch will turn off, and the output will be held.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Model	Model type	DINT	0, 1, 2	1	Model type: 0: Non-integral. 1: Pseudo-integral. 2: Pure integral. Currently, types 0 (Non-integral) and 1 (Pseudo-integral) are primarily used. Type 2 (Pure integral) is not implemented. In practice, integral models are typically handled as pseudo-integral. If the model selection does not follow 0, 1, 2, it is set to 0 by default.
Low	Lower limit of the disturbance variable	REAL	Follows data type	0	Lower limit; if the value falls below the lower limit, it is clamped to the lower limit.
High	Upper limit of the disturbance variable	REAL	Follows data type	100	Upper limit; if the value exceeds the upper limit, it is clamped to the upper limit.
DMax	Increment limit	REAL	Follows data type	100	Increment limit; when exceeded, it will be clamped to the maximum increment value. Any invalid value is set to 0.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ST	Switch state	DINT	0, 1	-	Switch status. Initially the value equals the switch input value. When an internal error occurs, this value is set to 1.
ERR	Fault code	DINT	Follows data type	-	For fault codes, see PFCm_dv fault codes.
DV_S	Disturbance velocity value	REAL	Follows data type	-	Disturbance velocity value calculated in integral mode and used by the PFCm component. Once generated, it must be output to the input interface with the same name on the control instruction for feedforward correction.

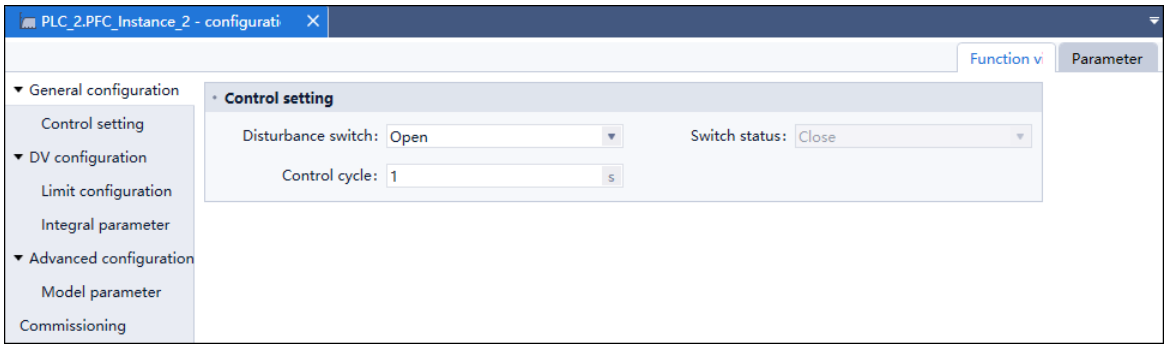
Output Variable	Name	Data Type	Valid Range	Default Value	Description
PV_t	Feedforward correction value	REAL	Follows data type	-	Feedforward value after limit.
DDVTrend	Disturbance sequence	ARRAY [0..199] OF REAL	Follows data type	-	Disturbance sequence. After generating this sequence, it must be output to the input interface with the same name on the control instruction for feedforward correction.
PFCm_dvOutput	Output parameter structure	PFCm_dvOutput	Follows data type	-	Output parameter structure.

PFCm_dvOutput structure configuration

Output Variable	Name	Data Type	Valid Range	Default Value	Description
PV_Past	Measured value history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records historical values for the disturbance input.
PVt_Past	Feedforward value history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records historical values of the feedforward signal.
DV_Low_Past	Disturbance value lower limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical lower limits for the disturbance value, for plotting.
DV_High_Past	Disturbance value upper limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical upper limits for the disturbance value, for plotting.
DV_Low_Future	Disturbance value lower limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future lower limits for the disturbance value, for plotting.
DV_High_Future	Disturbance value upper limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future upper limits for the disturbance value, for plotting.
DV_PV_Future	Disturbance value future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future disturbance values, for plotting.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
DDV_Past	Feedforward value history increment sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical increments for the feedforward value, for plotting.
Version	Version information	ARRAY [0..3] OF DINT	Follows data type	-	Records the version information.

- **Function**
  - PFCm_dv is a disturbance feedforward prediction module. Its primary function is to predict the future disturbance sequence of the controlled object based on the current disturbance input. The predicted disturbance sequence can be fed forward to the feedforward input channel of the PFCm instruction, enabling proactive compensation of disturbances and improving disturbance rejection.
  - The PFCm_dv function block includes an intuitive interface for parameter configuration and commissioning. After adding the PFCm_dv component, select "Control object > PFC" in the left project tree, and then double-click the PFCm_dv instance under the PFC node to open the configuration and tuning interface, as shown in the figure below.

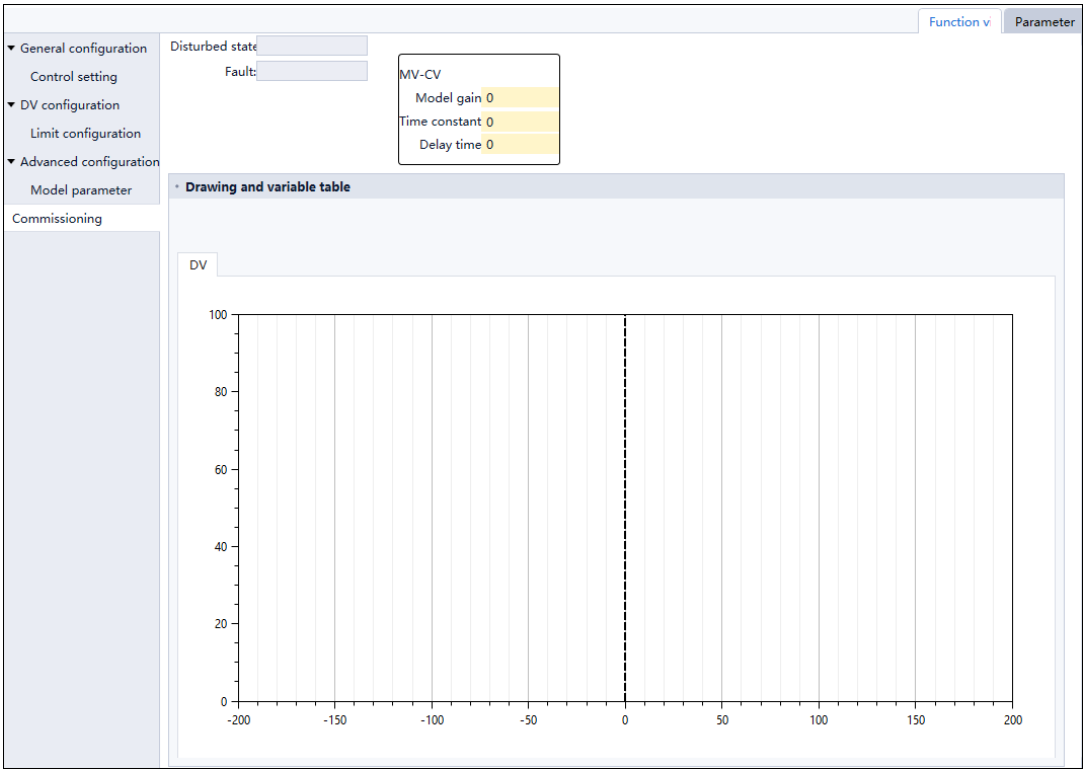


    - Parameter description

      The parameter configuration of the PFCm_dv instruction includes general configuration, DV configuration, and model configuration. For detailed parameter descriptions, see Relevant variables.

    - Commissioning

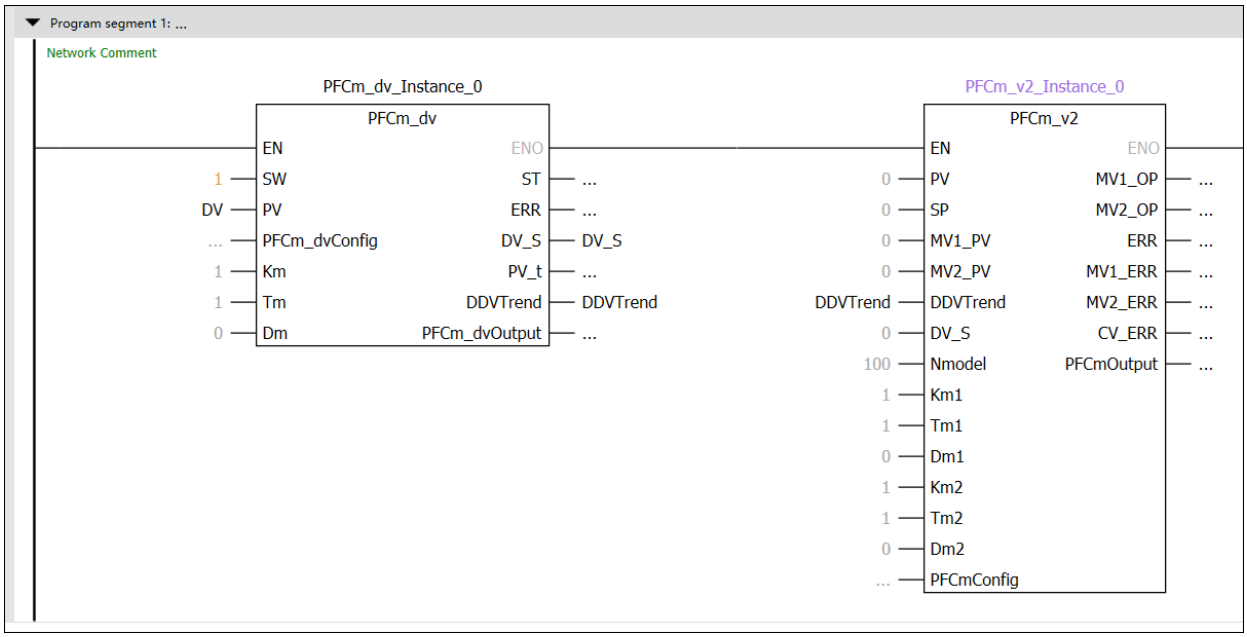
      After the parameters are configured, the historical and future disturbance value sequence can be viewed directly in the commissioning interface.



- **Program example**

  During operation, this instruction primarily feeds the sequence output by DDVTrend into the control component PFCm. The generated sequence is determined by the model parameters Km, Tm, and Dm, as well as the input disturbance PV.

  LD:



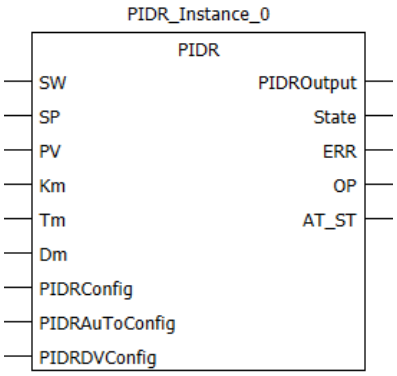
- **Precautions**
  - During operation, ensure that the switch is on, and the configuration is correct.
  - When setting the model parameters, ensure they are consistent with the controller parameter type, either all integral or non-integral.
  - The execution time of the PFCm_dv_dv instruction is 8 ms. With 140 PFCm_dv function blocks loaded, the total execution time is estimated to reach approximately 1 second. Loading 1154 PFCm_dv function blocks will reach the controller's resource limit. During operation, configure the scheduling task cycle properly. For example, if 10 PFCm_dv are required to perform loop control simultaneously,

with a block calculation time of 80 ms. The task cycle must be at least twice this value, that is, 160 ms, to ensure the CPU load does not become too high.

18.3.2.9 PIDR

The PIDR component is designed for 1×1 loop object control. It features fast computation and low memory usage, making it suitable for fast dynamic and resource-constrained systems. It includes built-in feedforward and auto-tuning functions, and can be deployed with simple configuration.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PIDR	Robust PID	FB		PIDR_Instance_0(SW := , SP := , PV := , Km := , Tm := , Dm := , PIDRConfig := , PIDRAuToConfig := , PIDRDVConfig := , PIDROutput => , State => , ERR => , OP => , AT_ST =>);

- **Related variables**

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
SW	Operating switch	DINT	0, 1	0	0: Component off. 1: Component on. Ensure the component is in the on state when in use.
SP	Setpoint	REAL	Follows data type	0	Setpoint for the controlled variable, which defines the control target of the controller. If properly configured, the actual control result (PV) must closely follow the SP.
PV	Measured value	REAL	Follows data type	0	Input value of the controlled variable measurement, which must be provided by an external physical object or simulation object.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Km	Model gain	REAL	Follows data type	1	An algorithm parameter that describes the response gain of the controlled object. It must be determined based on the influence of the output OP on the controlled object. The greater this influence, the larger the gain value.
Tm	Model time constant	REAL	(0, 10000]	1	An algorithm parameter that describes the response speed of the controlled object. The greater the inertia of the controlled object, the slower the response, and the larger this value.
Dm	Model hysteresis	REAL	[0, 10000]	0	An algorithm parameter that represents the response hysteresis of the controlled object. The greater the hysteresis of the object to input changes, the larger this value.
PIDRConfig	Input parameter structure	PIDRConfig	Follows data type	-	Input parameter configuration structure.
PIDRAuToConfig	Auto-tuning parameter structure	PIDRAuToConfig	Follows data type	-	Auto-tuning parameter configuration structure.
PIDRDVConfig	Feedforward structure	PIDRDVConfig	Follows data type	-	Feedforward parameter configuration structure.

PIDRConfig structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Ts	Control cycle	REAL	(0, 10000]	1	This value represents the control and tuning cycle. The value must be greater than 0. Ensure that this value is not set too low; otherwise it could exceed the operation cycle. If Ts is less than 0, the algorithm will automatically terminate, the switch will turn off, and the output will be held.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Model	Model type	DINT	0, 1, 2	1	Model type: 0: Non-integral. 1: Pseudo-integral. 2: Pure integral. **Note:** The algorithm internally distinguishes between non-integral and integral models. The logic for pseudo-integral models follows the same rules as those for integral models. When the model selection is not 0, 1, or 2, the algorithm terminates, the switch is turned off, and the output will be held.
Model_Feature	Control performance	DINT	0, 1, 2	1	Control performance: 0: Fast. 1: Medium. 2: Slow. Selecting "Fast" results in more aggressive control performance. Selecting "Slow" results in smoother control. Select the setting according to the actual control requirements.
COF	Filter coefficient	REAL	[0, 1]	0.9	An algorithm parameter that affects the performance of feedback correction. The feedback value balances the measurement for the current cycle with the model prediction. The magnitude of the COF determines whether the algorithm places greater weight on the measured value or the predicted value in the feedback loop. • When set to 0, the feedback value uses the model prediction, and no feedback correction is applied. • When set to 1, the feedback value uses the measured value. • For other values, an average is calculated.
COP	Performance coefficient	REAL	(0, 10000]	1	An algorithm parameter that affects the control performance. Increasing the value speeds up the system response to some extent, while decreasing it slows down the system response to some extent.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Tref	Closed-loop reference time	REAL	(0, 10000]	10	An algorithm parameter that affects the control performance. It is used to reduce the system response speed. Increasing this value slows down the system response to some extent, while decreasing it will speed up the system response to some extent. If the value exceeds the limit, it will be capped at 1e-6.
I_COF	Integral deviation correction coefficient	REAL	[0, 1]	0.1	Correction coefficient for integral speed. This value will take effect when the integral model is used. The larger the value, the faster the integral model speed operates in the algorithm, resulting in greater changes in future prediction, and thus causing output changes. Generally, fine-tuning this value after reaching steady state can reduce steady-state fluctuations.
DOPmax	Output increment limit	REAL	>=0	100	Sets the upper limit of output increment.
OP_Low_Val	Output valid lower limit	REAL	Follows data type	0	When the value falls below this output valid lower limit, the switch will be turned off and the control will be disabled.
OP_High_Val	Output valid upper limit	REAL	Follows data type	100	When the value exceeds this output valid upper limit, the switch will be turned off and the control will be disabled.
OP_Low_Ope	Output operation lower limit	REAL	Follows data type	0	When the value falls below this output operation lower limit, a warning is issued and the output is clamped to this limit.
OP_High_Ope	Output operation upper limit	REAL	Follows data type	100	When the value exceeds this output operation upper limit, a warning will be issued and the output is clamped to this limit.
SP_Low_Val	Setpoint valid lower limit	REAL	Follows data type	0	Valid lower limit of the setpoint. When the value falls below this lower limit, the switch will be turned off and the control will be disabled.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
SP_High_Val	Setpoint valid upper limit	REAL	Follows data type	100	Valid upper limit of the setpoint. When the value exceeds this upper limit, the switch will be turned off and the control will be disabled.
SP_Low_Ope	Setpoint operation lower limit	REAL	Follows data type	0	Operation lower limit of the setpoint. When the value falls below this lower limit, the system will display a warning and limit the value to the lower limit.
SP_High_Ope	Setpoint operation upper limit	REAL	Follows data type	100	Operation upper limit of the setpoint. When the value exceeds this upper limit, the system will display a warning and limit the value to the upper limit.
SP_DLimit	Setpoint increment range	REAL	>=0	0.1	Algorithm parameter that affects the control effect; the system is considered stable when PV is within the range SP ± SP_DLimit. When high steady-state accuracy is required, reduce the interval width.

PIDRAuToConfig structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
AT_Start	Auto-tuning switch	DINT	0, 1	0	Auto-tuning switch. If this function is used, set AT_Start to 1.
AT_Model	Tuning model type	DINT	Follows data type	0	Tuning model type: 0: Non-integral 1: Pseudo-integral 2: Pure integral The auto-tuning function outputs the corresponding model parameters according to the selection. It is recommended to select according to the actual characteristics of the controlled object.
AT_OPInit	Initial OP output for tuning	REAL	Follows data type	0	OP output value at initial steady state. The controlled object must reach a stable state at this output.
AT_OPTarget	Target OP output for tuning	REAL	Follows data type	100	This value is the step output value at the start of tuning. The controller will perform a step change from "AT_OPInit" to "AT_OPTarget" to excite the response of the controlled object.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
AT_PVTarget	PV value at tuning end	REAL	Follows data type	100	This value is the target PV value to be achieved during tuning. The tuning process completes when the measured value (PV) of the controlled object reaches this value after tuning begin. Then, the tuning result will be output.
AT_PVDeltTar	Early termination threshold	REAL	Follows data type	0	When the PV value reaches AT_PV Target – AT_PVDeltTar, the tuning process terminates early and outputs the tuning result.
AT_InitRate	Initial steady-state slope	REAL	Follows data type	0	PV measured value slope that allows the step response tuning to start.

PIDRDVConfig structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
DV_Start	Feedforward function switch	DINT	0, 1	0	Feedforward switch. If this function is used, set DV_Start to 1.
DV_Km	Disturbance model gain	REAL	Follows data type	1	Inputs the model gain of the measurable first-order disturbance model.
DV_Tm	Disturbance model time constant	REAL	(0, 10000]	1	Inputs the model time constant of the measurable first-order disturbance model.
DV_Dm	Disturbance model hysteresis	REAL	[0, 10000]	0	Inputs the model hysteresis of the measurable first-order disturbance model.
DV_PV	Measured disturbance value	REAL	Follows data type	0	Inputs the measured disturbance value.
DV_PVHigh	Disturbance input upper limit	REAL	Follows data type	100.0	When the measured disturbance input DV_PV exceeds this value, an alarm is triggered and the value is limited to this upper limit.
DV_PVLow	Disturbance input lower limit	REAL	Follows data type	-100.0	When the measured disturbance input DV_PV is below this value, an alarm is triggered and the value is limited to this lower limit.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
DDV_Pred_t	Current feedforward increment	REAL	Follows data type	0.0	Displays the feedforward increment calculated at the current moment. This value is superimposed on the prediction part.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
PIDROutput	Output parameter structure	PIDROutput	Follows data type	-	Output parameter configuration structure.
State	Switch state	DINT	0, 1	-	Indicates the actual state of the controller.
ERR	Fault code	DINT	Follows data type	-	0 for normal operation; non-zero when an error occurs. See PIDR fault codes.
OP	Valve position output value	REAL	Follows data type	-	Controller output value.
AT_ST	Tuning result	DINT	0, 1, 2, 3	-	Tuning state display. 0: Not tuned 1: Coarse tuning in progress 2: Fine tuning in progress 3: Tuning completed

PIDROutput structure configuration

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Version	Version information	ARRAY [0..3] OF DINT	Follows data type	-	Used to record internal version information.
DOP	Valve position increment output	REAL	Follows data type	-	Increment of the output value OP.
PV_Pred_t	Estimated value	REAL	Follows data type	-	Estimated value, that is, the response value of the controlled object, calculated by the internal model using historical data.
PV_Pred_tf	Predicted value	REAL	Follows data type	-	Predicted value, that is, the estimated value corrected based on the actual measured value PV.
PV_Pred_e	Prediction error	REAL	Follows data type	-	Difference between the actual measured value PV and the predicted value.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
PV_Pred_eSum	Cumulative prediction error	REAL	Follows data type	-	Cumulative sum of prediction errors.
DCV_S	Integral deviation accumulation rate	REAL	Follows data type	-	Used for integral deviation correction.
AT_Km	Tuning parameter Km	REAL	Follows data type	-	Model gain result obtained from auto-tuning.
AT_Tm	Tuning parameter Tm	REAL	> 0	-	Model time constant result obtained from auto-tuning.
AT_Dm	Tuning parameter Dm	REAL	>=0	-	Model hysteresis parameter result obtained from auto-tuning.

- **Function**
  PIDR includes one manipulated variable (output by OP), one controlled variable (measured by PV), and supports feedforward and auto-tuning functions. This component is primarily used for rapid deployment in single-input single-output processes.
- **Program example**
  1. Simulation example
     This example works with the FPTLGsim instruction, which simulates a first-order plus pure hysteresis process, to form a single-input single-output control simulation.
     Configure the PIDR instruction parameters (model parameters: Km, Tm, Dm; controller parameters: COP, COF, etc.) according to the characteristics of the controlled object.
     The PIDR instruction responds to the output (PV) of the FPTLGsim object, obtains the control quantity (OP) output, and applies it to the original control object, forming a control loop.
     Control expectation: If all parameters are configured properly, the controller will eventually keep the output value (PV) of the controlled object near the setpoint SP.
     ST language example is shown in the figure:

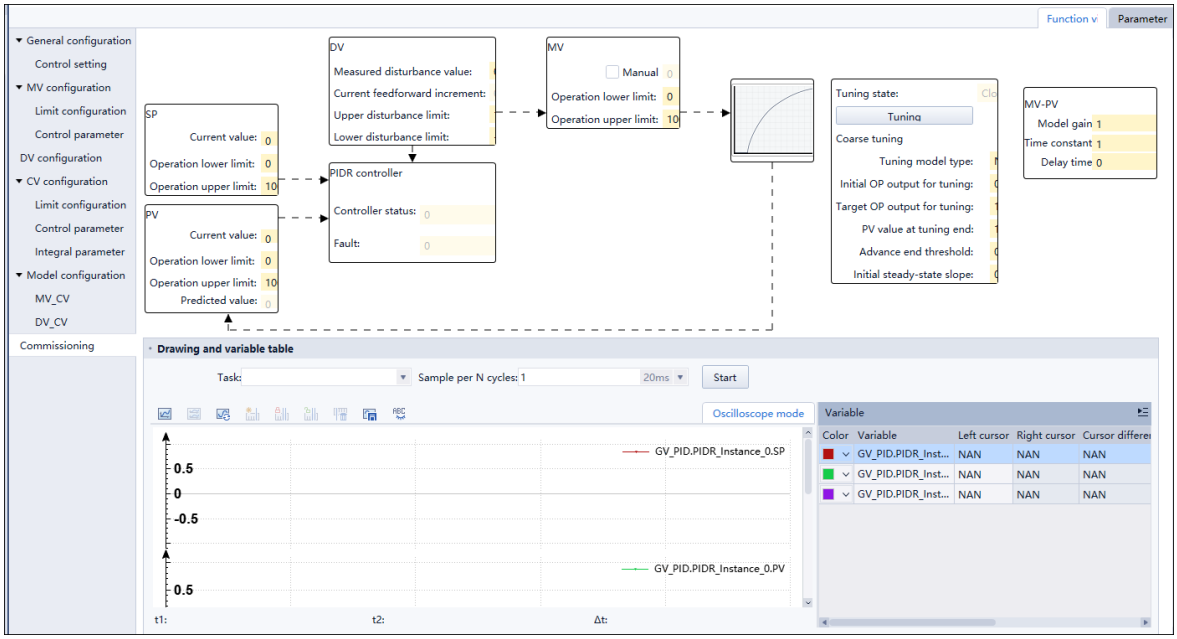
	Name	Data type	Address	Initial value
2	FPTLGSim_Instance_0	FPTLGSim		
3	PIDR_Instance_0	PIDR	First-order lag controlled object	
4	SP	REAL	PIDR controller	50
5	var1	REAL		

IF CAS FOR WHI REP RET EX REG

```
FPTLGSim_Instance_0(IN := ,
                    Gain := 20,
                    T := 13,
                    TmLag := 1,
                    Cycle := 1,
                    Reset := ,
                    OUT => ,
                    ErrorBits => );

PIDR_Instance_0(SW := 1,
                SP := SP,
                PV := FPTLGSim_Instance_0.OUT,
                Km := 20,
                Tm := 13,
                Dm := 1,
                PIDRConfig := ,
                PIDRAuToConfig := ,
                PIDRDVConfig := ,
                PIDROutput => ,
                State => ,
                ERR => ,
                OP => FPTLGSim_Instance_0.IN,
                AT_ST => );
```

In the ST programming area, right-click the module name and select "Instruction configuration" to open the configuration interface. You can monitor the parameters on the "Commissioning" tab. After clicking "Start" in the "Drawing and variable table" area, the curves for the setpoint (SP), process value (PV), and controller output (OP) are displayed.



2. Auto-tuning function application

The PIDR instruction includes a built-in parameter auto-tuning function, automatically obtaining the model parameters (Km, Tm, and Dm) of the controlled object. Click the "Tuning" button on the "Commissioning" tab of the instruction configuration page. Configure the auto-tuning parameters according to the variable description to start parameter tuning.
Because the controller internal variable array has a capacity of only 200, ensure that the duration from the OP output step to the completion of tuning stays within 200 control cycles. 100 to 150 cycles is recommended. Furthermore, the tuning process does not require waiting for the response to reach a complete steady state after the step. Generally, it only needs to capture the portion of the controlled object's response with the maximum slope.

The tuning process is as follows:

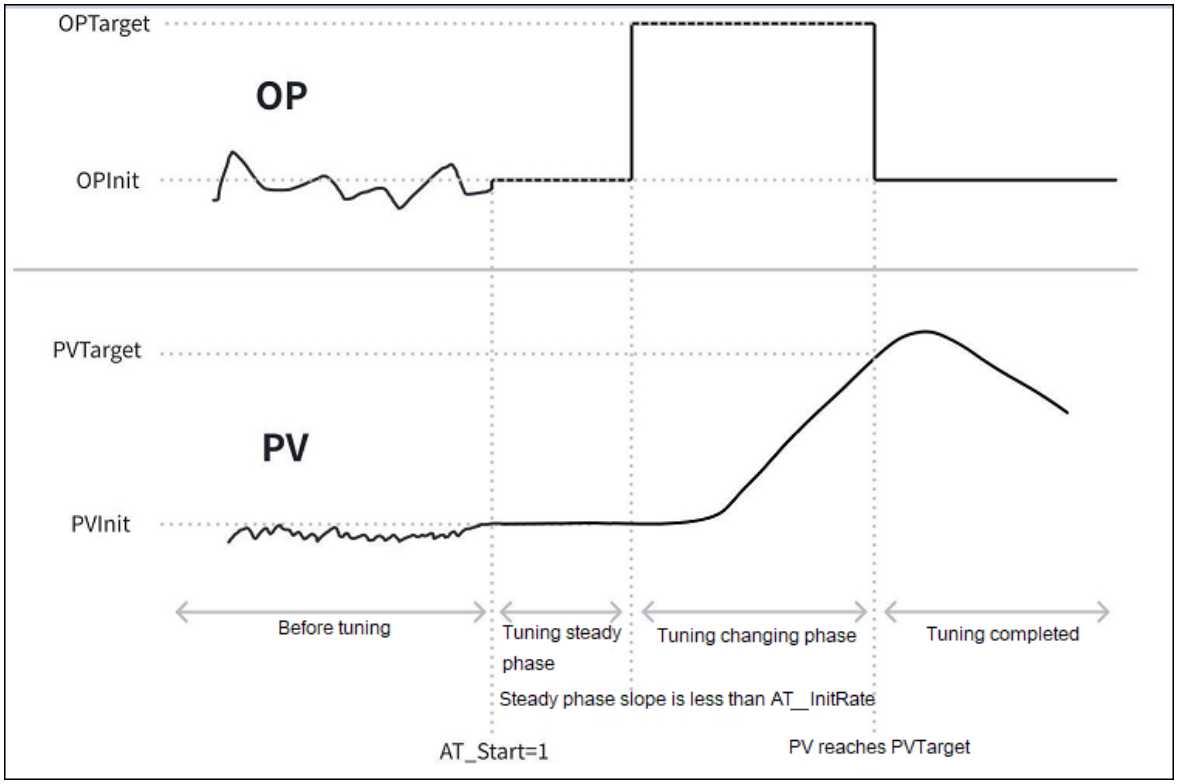
a. Set "AT_Start" to 1 to enable the auto-tuning switch.

b. Configure the tuning parameter "Initial OP output for tuning" (AT_OPInit). This value is the OP output value at the initial steady state. Ensure that the controlled object reaches a basic stable state at this output. Open-loop control can be used before the tuning process to stabilize the controlled object at this output. Then, use the auto-tuning function.

c. Configure the tuning parameter "Target OP output for tuning" (AT_OPTarget). This value is the step output value at the start of tuning. The controller will perform a step change from "AT_OPInit" to "AT_OPTarget" to excite the response of the controlled object.

d. Configure the tuning parameter "PV value at tuning end" (AT_PVTarget). This value is the target PV value to be achieved during tuning. The tuning process completes when the measured value (PV) of the controlled object has reached this value after tuning begin. Pay special attention to the response polarity of the controlled object. Ensure that the measured value reaches AT_PVTarget after tuning starts, rather than the PV changing in the opposite direction after the OP step change.

e. Click "Tuning" to run the auto-tuning, and the auto-tuning results can be obtained at AT_Km, AT_Tm, and AT_Dm.



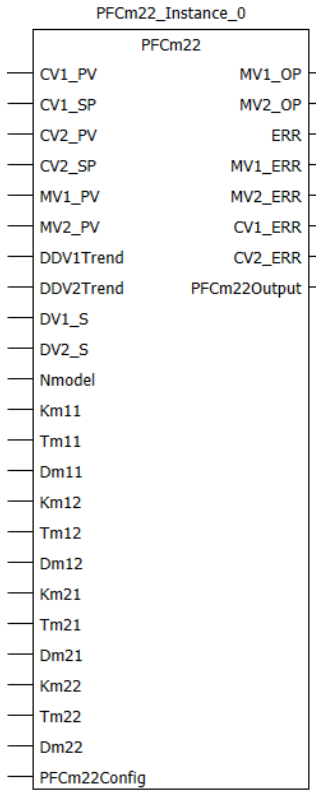
- **Precautions**
  - During operation, ensure that the switch is on, and the configuration is correct.

◦ The execution time of a single PIDR instruction is 75 μs, and the memory usage is 1.6 KB. During operation, configure the scheduling task cycle properly. For example, if five PIDRs need to run simultaneously, the computation time is about 218 μs. It is recommended to set the task period to at least twice this value.

18.3.2.10 PFCm22

This instruction is a dual-input dual-output control instruction. It is primarily used in 2x2 scenarios, such as the coupled control of temperature 1 and temperature 2 by heater 1 and heater 2, involving mutual coupling between the two temperature zones.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PFCm22	Multivariable input/output advanced control	FB		PFCm22_Instance_0(CV1_PV := , CV1_SP := , CV2_PV := , CV2_SP := , MV1_PV := , MV2_PV := , DDV1Trend := , DDV2Trend := , DV1_S := , DV2_S := , Nmodel := , Km11 := , Tm11 := , Dm11 := , Km12 := , Tm12 := , Dm12 := , Km21 := , Tm21 := , Dm21 := , Km22 := , Tm22 := , Dm22 := , PFCm22Config := , MV1_OP => , MV2_OP => , ERR => , MV1_ERR => , MV2_ERR => , CV1_ERR => , CV2_ERR => , PFCm22Output =>);

- **Related variables**

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
CV1_PV	Measured value of the controlled variable 1	REAL	Follows data type	0	Input value of the controlled variable 1 measurement, which must be provided by an external physical object or simulation object.
CV1_SP	Setpoint value of the controlled variable 1	REAL	Follows data type	0	Setpoint value of the controlled variable 1, which determines the control objective of the controller so that the final actual control result (PV) must closely follow CV1_SP.
CV2_PV	Measured value of the controlled variable 2	REAL	Follows data type	0	Input value of the controlled variable 2 measurement, which must be provided by an external physical object or simulation object.
CV2_SP	Setpoint value of the controlled variable 2	REAL	Follows data type	0	Setpoint value of the controlled variable 2, which determines the control objective of the controller so that the final actual control result (PV) must closely follow CV2_SP.
MV1_PV	Measured value of the manipulated variable 1	REAL	Follows data type	0	Measured value of manipulated variable 1. The actual output value 1 must be written back to this variable. In a heating application: if only algorithmic control is active, write back the value of MV1_OP; if an external input is present, write back the actual value sent to the actuator.
MV2_PV	Measured value of the manipulated variable 2	REAL	Follows data type	0	Measured value of manipulated variable 2. The actual output value 2 must be written back to this variable. In a heating application: if only algorithmic control is active, write back the value of MV2_OP; if an external input is present, write back the actual value sent to the actuator.
DDV1Trend	Disturbance sequence 1	ARRAY [0..199] OF REAL	Follows data type	-	Disturbance sequence 1 generated by the dv series component. When using this variable, pass its output "DDVTrend" to this variable. It is used to predict future disturbances so that the controller can take proactive action to suppress them. This variable corresponds to CV1 disturbance.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
DDV2Trend	Disturbance sequence 2	ARRAY [0..199] OF REAL	Follows data type	-	Disturbance sequence 2 generated by the dv series component. When using this variable, pass its output "DDVTrend" to this variable. It is used to predict future disturbances so that the controller can take proactive action to suppress them. This variable corresponds to CV2 disturbance.
DV1_S	Disturbance velocity 1	REAL	Follows data type	-	Disturbance velocity generated by the PFCm_dv component. When using this variable, pass its DV_S output to this variable. It is used to predict future disturbances so that the controller can take proactive action to suppress them. This variable corresponds to CV1 disturbance.
DV2_S	Disturbance velocity 2	REAL	Follows data type	-	Disturbance velocity generated by the PFCm_dv component. When using this variable, pass its DV_S output to this variable. It is used to predict future disturbances so that the controller can take proactive action to suppress them. This variable corresponds to CV2 disturbance.
Nmodel	Model length	DINT	[1, 200]	100	The model length is used to adjust the control effect. The larger this value, the longer the effective prediction horizon, and the further the future trends can be predicted.
Km11	Model MV1-CV1 gain	REAL	Follows data type	1	An algorithm parameter that affects the control performance. Increasing the value speeds up the system response to some extent, while decreasing it slows down the system response to some extent.
Tm11	Model MV1-CV1 time constant	REAL	(0, 10000]	1	An algorithm parameter that affects the control performance. The smaller the Tm value, the fewer cycles required to reach a steady state. If the value exceeds the limit, it will be limited to 1e-6.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Dm11	Model MV1-CV1 hysteresis	REAL	[0, 10000]	0	An algorithm parameter that affects the control performance. The delay expressed in control cycles is Dm/Ts. If the value exceeds the limit, it will be limited to 0.
Km12	Model MV2-CV1 gain	REAL	Follows data type	1	An algorithm parameter that affects the control performance. Increasing the value speeds up the system response to some extent, while decreasing it slows down the system response to some extent.
Tm12	Model MV2-CV1 time constant	REAL	(0, 10000]	1	An algorithm parameter that affects the control performance. The smaller the Tm value, the fewer cycles required to reach a steady state. If the value exceeds the limit, it will be limited to 1e-6.
Dm12	Model MV2-CV1 hysteresis	REAL	[0, 10000]	0	An algorithm parameter that affects the control performance. The delay expressed in control cycles is Dm/Ts. If the value exceeds the limit, it will be limited to 0.
Km21	Model MV1-CV2 gain	REAL	Follows data type	1	An algorithm parameter that affects the control performance. Increasing the value speeds up the system response to some extent, while decreasing it slows down the system response to some extent.
Tm21	Model MV1-CV2 time constant	REAL	(0, 10000]	1	An algorithm parameter that affects the control performance. The smaller the Tm value, the fewer cycles required to reach a steady state. If the value exceeds the limit, it will be limited to 1e-6.
Dm21	Model MV1-CV2 hysteresis	REAL	[0, 10000]	0	An algorithm parameter that affects the control performance. The delay expressed in control cycles is Dm/Ts. If the value exceeds the limit, it will be limited to 0.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Km22	Model MV2-CV2 gain	REAL	Follows data type	1	An algorithm parameter that affects the control performance. Increasing the value speeds up the system response to some extent, while decreasing it slows down the system response to some extent.
Tm22	Model MV2-CV2 time constant	REAL	(0, 10000]	1	An algorithm parameter that affects the control performance. The smaller the Tm value, the fewer cycles required to reach a steady state. If the value exceeds the limit, it will be limited to 1e-6.
Dm22	Model MV2-CV2 hysteresis	REAL	[0, 10000]	0	An algorithm parameter that affects the control performance. The delay expressed in control cycles is Dm/Ts. If the value exceeds the limit, it will be limited to 0.
PFCmConfig	Input parameter structure	PFCm22Config	Follows data type	-	Input parameter configuration structure.

PFCm22Config structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Ts	Sampling cycle	REAL	(0, 10000]	1	This value determines the control cycle. This value must be greater than 0, and can not be set too low. An excessively small value may cause a calculation timeout and degrade control performance. If Ts is less than 0, the algorithm will automatically terminate, the switch will turn off, and the output will be held.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Model1	Model type 1	DINT	0, 1, 2	1	Model for MV1-CV1. Model type: 0: Non-integral. 1: Pseudo-integral. 2: Pure integral. Currently, types 0 (Non-integral) and 1 (Pseudo-integral) are primarily used. Type 2 (Pure integral) is not implemented. In practice, integral models are typically handled as pseudo-integral. When the model selection is not 0, 1, or 2, the algorithm terminates, the switch is turned off, and the output will be held.
Model2	Model type 2	DINT	0, 1, 2	1	Model for MV2-CV1. Model type: 0: Non-integral. 1: Pseudo-integral. 2: Pure integral. Currently, types 0 (Non-integral) and 1 (Pseudo-integral) are primarily used. Type 2 (Pure integral) is not implemented. In practice, integral models are typically handled as pseudo-integral. When the model selection is not 0, 1, or 2, the algorithm terminates, the switch is turned off, and the output will be held.
Model3	Model type 3	DINT	0, 1, 2	1	Model for MV1-CV2. Model type: 0: Non-integral. 1: Pseudo-integral. 2: Pure integral. Currently, types 0 (Non-integral) and 1 (Pseudo-integral) are primarily used. Type 2 (Pure integral) is not implemented. In practice, integral models are typically handled as pseudo-integral. When the model selection is not 0, 1, or 2, the algorithm terminates, the switch is turned off, and the output will be held.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Model4	Model type 4	DINT	0, 1, 2	1	Model for MV2-CV2. Model type: 0: Non-integral 1: Pseudo-integral 2: Pure integral Currently, types 0 (Non-integral) and 1 (Pseudo-integral) are primarily used. Type 2 (Pure integral) is not implemented. In practice, integral models are typically handled as pseudo-integral. When the model selection is not 0, 1, or 2, the algorithm terminates, the switch is turned off, and the output will be held.
MV1_Config	MV1 variable input parameter structure	PFCm22_MVConfig	Follows data type	-	MV1 variable input parameter configuration structure.
MV2_Config	MV2 variable input parameter structure	PFCm22_MVConfig	Follows data type	-	MV2 variable input parameter configuration structure.
CV1_Config	CV1 variable input parameter structure	PFCm22_CVConfig	Follows data type	-	CV1 variable input parameter configuration structure.
CV2_Config	CV2 variable input parameter structure	PFCm22_CVConfig	Follows data type	-	CV2 variable input parameter configuration structure.
Single_SW	Single output switch	DINT	0, 1	0	Single output switch. If this switch is set to ON, it ensures that when the two MV gains have opposite signs, only one output direction is active. This is typically used to save energy.

PFCm22_MVConfig structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
MV_SW	Manipulated variable switch	DINT	0, 1	1	MV switch; when this switch is off, the control is disabled. Invalid values are automatically set to 0.
MV_High_Val	Manipulated variable valid upper limit	REAL	Follows data type	200	Valid upper limit. When the value exceeds the valid upper limit, the switch will be turned off and the control will be disabled.
MV_Low_Val	Manipulated variable valid lower limit	REAL	Follows data type	0	Valid lower limit. When the value falls below the valid lower limit, the switch will be turned off and the control will be disabled.
MV_High_Ope	Operation upper limit of the manipulated variable	REAL	Follows data type	100	Operation upper limit. When the value exceeds this operation upper limit, a warning will be issued and the value will be clamped to this operation upper limit.
MV_Low_Ope	Operation lower limit of the manipulated variable	REAL	Follows data type	0	Operation lower limit. When the value falls below this operation lower limit, a warning will be issued and the value will be clamped to this operation lower limit.
MV_DMax	Maximum positive increment for the manipulated variable	REAL	[0, 10000]	100	Defines the maximum allowable positive increment of the output. Invalid input values are limited to 100.
MV_DMin	Maximum negative increment for the manipulated variable	REAL	[-10000, 0]	-100	Defines the maximum allowed negative increment of the output. Invalid input values are limited to -100.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
MV_COP	Manipulated variable performance coefficient	REAL	[0, 10000]	1	An algorithm parameter that affects the control performance. Decreasing the value speeds up the system response to some extent, while increasing it slows down the system response to some extent. If this value exceeds the upper limit, it will be limited to the upper limit. If the value falls below the lower limit, it will be limited to the lower limit.
MV_Irv	Ideal injection value for the manipulated variable	REAL	Follows data type	1	The value that the corresponding MV output eventually converges to.
MV_Irv_C	Weight for manipulated variable ideal injection value	REAL	[0, 10000]	1	The larger the value, the closer the MV will converge to MV_Irv.
MV_Irv_SW	Ideal injection value switch	DINT	0, 1	0	When set to 0, the ideal injection value is inactive. When used, the value must be set to 1. Invalid values are automatically set to 0.

PFCm22_CVConfig structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
CV_SW	Controlled variable switch	INT	0, 1	1	Controlled variable switch. When the switch is off, the control is disabled. When used, it must be set to 1. Invalid values are automatically set to 0.
Cost_SS	Speed steady state weight	REAL	[0, 10000]	0	Speed steady state weight. In an integral model, this weight determines how quickly the speed converges to steady state. A larger value slows down the convergence. If set too large, the output may remain unchanged.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
CV_ECE	Equivalent deviation	REAL	[0, 10000]	1	Determines the weight of the corresponding CV variable. A larger value gives higher priority to controlling that CV.
CV_COF	Filter coefficient	REAL	[0, 1]	0.9	An algorithm parameter that affects the control performance. The smaller this value, the greater the confidence in the model prediction. When set to 0, no feedback correction is performed. Use a value greater than 0. If this value exceeds the upper limit, it will be limited to the upper limit. If the value falls below the lower limit, it will be limited to the lower limit.
CV_Tref	Closed-loop reference time	REAL	(0, 10000]	1	An algorithm parameter that affects the control performance. Increasing the value slows down the system response to some extent, while decreasing it speeds up the system response to some extent. If the value exceeds the limit, it will be limited to 1e-6.
I_COF	Integral deviation correction coefficient	REAL	[0, 1]	0.1	An algorithm parameter that affects the control performance. The smaller the value, the greater the confidence in the model prediction. If this value exceeds the upper limit, it will be limited to the upper limit. If the value falls below the lower limit, it will be limited to the lower limit.
CV_SP_DMax	Upper deviation range	REAL	[0, 10000]	0.1	An algorithm parameter that affects the control performance. The system is considered stable if CV_PV is within this range CV_SP + CV_SP_DMax. Invalid input values are limited to 0.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
CV_SP_DMin	Lower deviation range	REAL	[-10000, 0]	-0.1	An algorithm parameter that affects the control performance. The system is considered stable if CV_PV is within this range CV_SP + CV_SP_DMin. Invalid input values are limited to 0.
DeltaCV	Maximum change rate of the controlled variable	REAL	[0, 10000]	100	This algorithm parameter defines the maximum change value of CV during prediction. Invalid values are limited to 100.
CV_Low_Val	Valid lower limit of the controlled variable	REAL	Follows data type	0	Valid lower limit. When the value falls below the valid lower limit, the switch will be tu rned off and the control will be disabled.
CV_High_Val	Valid upper limit of the controlled variable	REAL	Follows data type	100	Valid upper limit. When the value exceeds the valid upper limit, the switch will be turned off and the control will be disabled.
CV_Low_Ope	Operation lower limit of the controlled variable	REAL	Follows data type	0	Operation lower limit. When the value falls below this operation lower limit, a warning will be issued and the value will be clamped to this operation lower limit.
CV_High_Ope	Operation upper limit of the controlled variable	REAL	Follows data type	100	Operation upper limit. When the value exceeds this operation upper limit, a warning will be issued and the value will be clamped to this operation upper limit.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
MV1_OP	Manipulated variable 1 output value	REAL	Follows data type	-	Output value of manipulated variable 1. This value must be output to the simulation object or actual object.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
MV2_OP	Manipulated variable 2 output value	REAL	Follows data type	-	Output value of manipulated variable 2. This value must be output to the simulation object or actual object.
ERR	Fault code	DINT	Follows data type	-	For fault codes, see PFCm 22 fault codes.
MV1_ERR	Manipulated variable 1 fault code	DINT	Follows data type	-	Fault code of manipulated variable 1.
MV2_ERR	Manipulated variable 2 fault code	DINT	Follows data type	-	Fault code of manipulated variable 2.
CV1_ERR	Controlled variable 1 fault code	DINT	Follows data type	-	Fault code of controlled variable 1.
CV2_ERR	Controlled variable 2 fault code	DINT	Follows data type	-	Fault code of controlled variable 2.
PFCm22Output	Output parameter structure	PFCm22Output	Follows data type	-	Output parameter structure.

PFCm22Output structure configuration

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Version	Version information	ARRAY [0..3] OF REAL	Follows data type	-	Used to obtain the version information.
NFlag	Number of errors	DINT	[0, 10]	-	Counts the number of errors.
Flag	Error information	ARRAY [0..9] OF DINT	Follows data type	-	Records detailed information of each error.
CV1_OutConfig	Controlled variable 1 output structure	PFCm22_CVOutConfig	Follows data type	-	Output structure of controlled variable 1.
CV2_OutConfig	Controlled variable 2 output structure	PFCm22_CVOutConfig	Follows data type	-	Output structure of controlled variable 2.
MV1_OutConfig	Manipulated variable 1 output structure	PFCm22_MVOutConfig	Follows data type	-	Output structure of manipulated variable 1.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
MV2_OutConfig	Manipulated variable 2 output structure	PFCm22_MVOutConfig	Follows data type	-	Output structure of manipulated variable 2.

PFCm22_CVOutConfig structure configuration

Output Variable	Name	Data Type	Valid Range	Default Value	Description
CV_NFlag	Controlled variable error count	DINT	[0, 10]	-	Number of recorded errors for the controlled variable
CV_Flag	Controlled variable error information	ARRAY [0..9] OF DINT	Follows data type	-	Detailed information of recorded errors for the controlled variable
CV_ST	Controlled variable switch status	DINT	0, 1	-	Status of the controlled variable switch. Initially the value equals the switch input value. When an internal error occurs, this value is set to 1.
CV_Pred_t	Estimated value of controlled variable	REAL	Follows data type	-	Estimated value, that is, the response value of the controlled object, calculated by the internal model using historical data.
CV_Pred_tf	Predicted value of controlled variable	REAL	Follows data type	-	Predicted value, that is, the corrected predicted value based on the predicted value and the actual measured value PV.
DCV_S_i	Velocity after integral deviation correction	REAL	Follows data type	-	Velocity after integral deviation correction.
DCV_S_SS	Future speed steady state value	REAL	Follows data type	-	Steady-state speed value, which is the speed value predicted after 200 cycles.
CV_PV_e	Predicted error of the controlled variable	REAL	Follows data type	-	Difference between the actual measured value and the predicted value.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
CV_PV_eSum	Cumulative sum of predicted errors for the controlled variable	REAL	Follows data type	-	The cumulative sum of errors (regardless of sign). Its magnitude indicates the degree of model mismatch.
CV_Pred_t_Past	Sequence of historical estimated values for the controlled variable	ARRAY [0..199] OF REAL	Follows data type	-	Records historical values.
CV_Pred	Open-loop prediction sequence of controlled variables	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of future values predicted when the current cycle is not applied.
CV_Future	Closed-loop prediction sequence of controlled variables	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of future values predicted after the current cycle has been applied.
CV_PV_Past	Sequence of historical measured values for the controlled variable	ARRAY [0..199] OF REAL	Follows data type	-	Records historical measured values.
SP_Past	Setpoint history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical setpoint values, for plotting.
SP_Future	Setpoint future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of the future setpoint values, for plotting.
CV_SP_Href	Upper reference trajectory sequence	ARRAY [0..199] OF REAL	Follows data type	-	Predicted upper trajectory to be approached.
CV_SP_Lref	Lower reference trajectory sequence	ARRAY [0..199] OF REAL	Follows data type	-	Predicted lower trajectory to be approached.
CV_Low_Past	Controlled variable lower limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical operation lower limits for the controlled variable, for plotting.

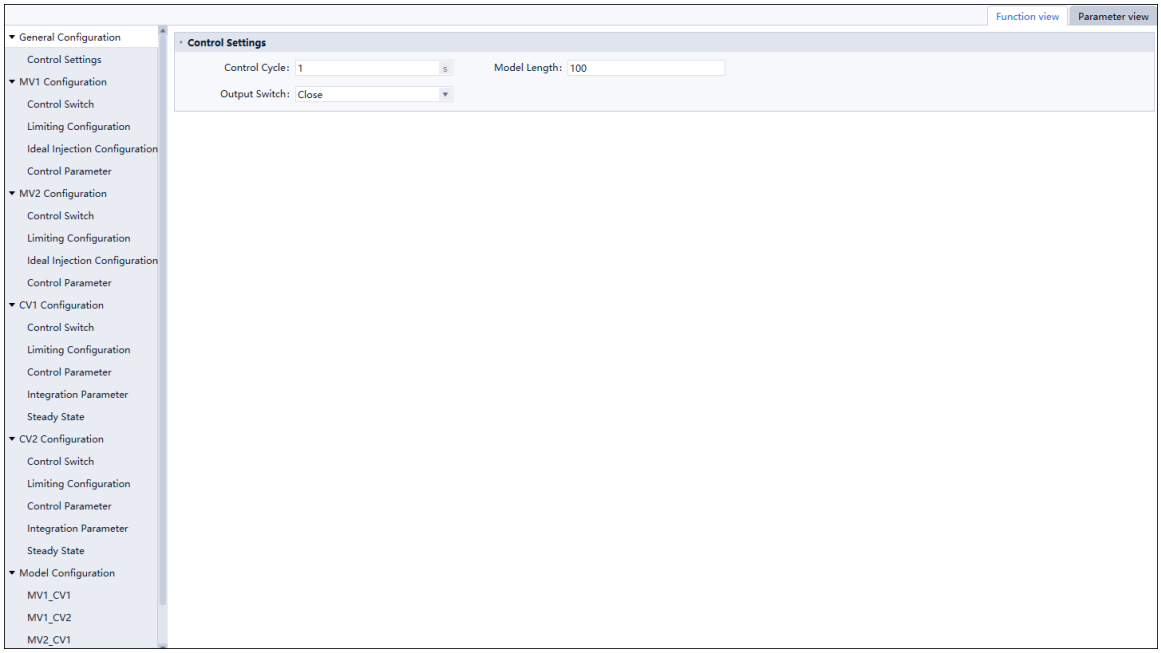
Output Variable	Name	Data Type	Valid Range	Default Value	Description
CV_High_Past	Controlled variable upper limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical upper limits for the controlled variable, for plotting.
CV_Low_Future	Controlled variable lower limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future operation lower limits for the controlled variable, for plotting.
CV_High_Future	Controlled variable upper limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future operation upper limits for the controlled variable, for plotting.

PFCm22_MVOutConfig structure configuration

Output Variable	Name	Data Type	Valid Range	Default Value	Description
MV_NFlag	MV error count	DINT	0, 1	-	Number of errors counted for the manipulated variable.
MV_Flag	MV error message	ARRAY [0..9] OF DINT	[0, 10]	-	Detailed information of recorded errors for the manipulated variable.
MV_Low_Past	Output lower limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical operation lower limits for the manipulated variable, for plotting.
MV_High_Past	Output upper limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical operation upper limits for the manipulated variable, for plotting.
MV_Low_Future	Output lower limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future operation lower limits for the manipulated variable, for plotting.
MV_High_Future	Output upper limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future operation upper limits for the manipulated variable, for plotting.
MV_Past	Output value history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical measured values for the manipulated variable, for plotting.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
MV_Future	Output value future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future measured values for the manipulated variable, for plotting.
MV_DOP_Past	Historical increment sequence of the valve position	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical increment values for the manipulated variable, for plotting.
MV_DOP	Manipulated variable action increment	REAL	Follows data type	-	Increment value of the manipulated variable.
MV_ST	Manipulated variable switch state	DINT	0, 1	-	Manipulated variable switch state. Initially, it equals the switch value. This value is set to 1 when an internal error occurs.
MV_Irv_Past	Ideal injection value history sequence	ARRAY [0..199] OF REAL	Follows data type		Sequence of historical ideal injection values, for plotting.
MV_Irv_Future	Ideal injection value future sequence	ARRAY [0..199] OF REAL	Follows data type		Sequence of future ideal injection values, for plotting.

- **Function**
  - PFCm22 is a 2x2 dual-input, dual-output control instruction. The controlled object can support non-integrating, integrating, or pseudo-integrating responses. The instruction includes two manipulated variables (MV), two controlled variables (CV), and a feedforward input channel (for use with PFCm_dv). The instruction does not rely on an external solver for optimization calculations. It is mainly intended for applications with simple coupling and does not require high-end products. It is suitable for fast control application scenarios with millisecond-level control cycles.
  - The PFCm22 function block includes an intuitive interface for parameter configuration and commissioning. After adding the PFCm22 component, select "Control object > PFC" in the left project tree, and then double-click the PFCm22 instance under the PFC node to open the configuration and tuning interface, as shown in the figure below.

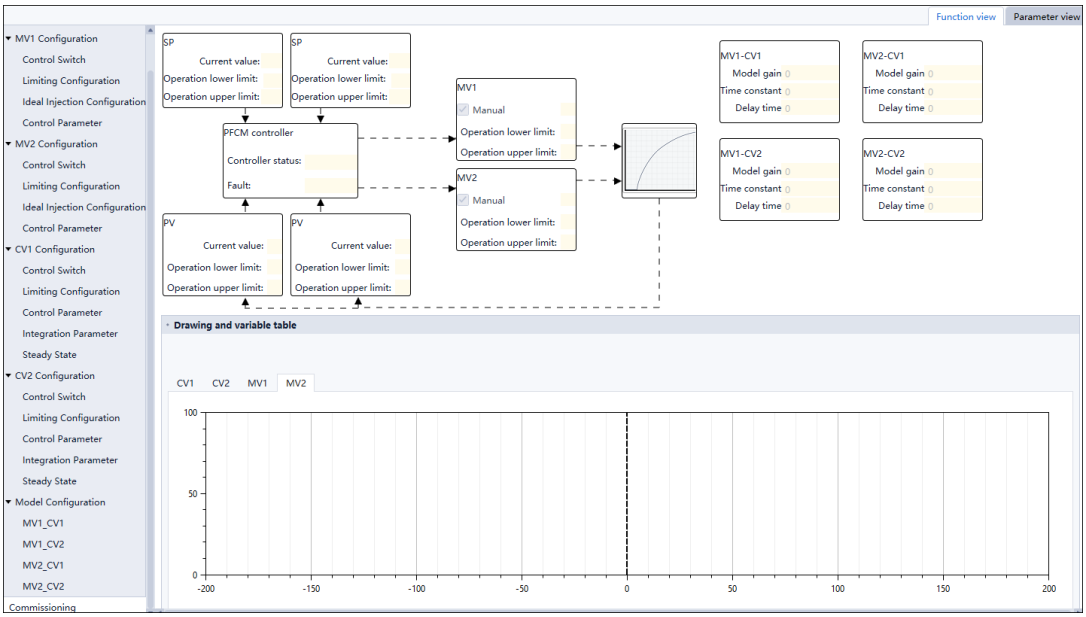


- Parameter description

  The parameter configuration of the PFCm22 instruction includes general configuration, MV configuration, CV configuration, and model configuration. For detailed parameter descriptions, see Relevant variables.

- Commissioning

  After the parameters are configured, the control effect can be viewed directly in the commissioning interface.



- **Program example**

  Here, the two-input, two-output control simulation between the PFCm22 instruction and four first-order inertia plus pure delay simulation instruction (FPTLGsim) is used as an example. The parameter configuration and control process are as follows:

  - Parameter configuration

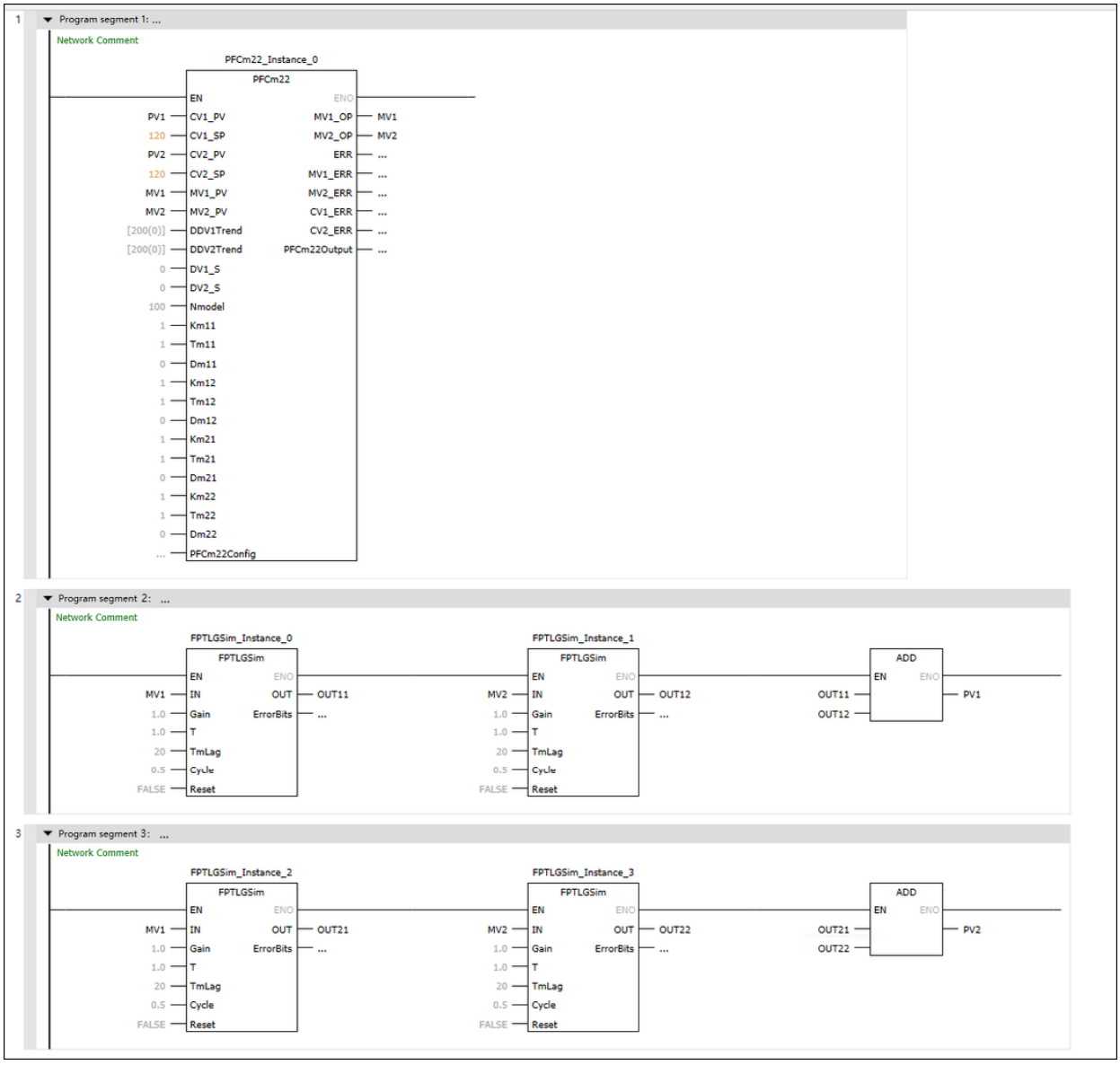
    Configure the PFCm22 instruction parameters based on the characteristics of the control object (Model parameters: Km11, Tm11, Dm11 and so on; controller parameters: COP, COF, etc.).

  - Control process

The PFCm22 instruction responds to the output (PV) of the FPTLGsim object, obtains the control quantity (OP) output, and applies it to the original control object, forming a control loop.

- Control expectation
  If all parameters are configured properly, the controller will eventually keep the output value (CV_PV) of the controlled object near the setpoint CV_SP.

LD language code example:

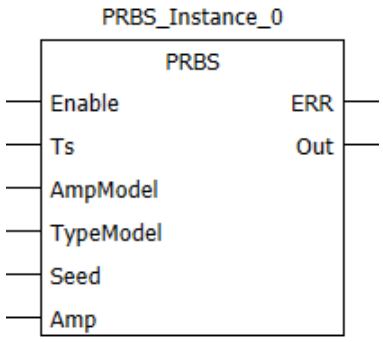


- **Precautions**
  - During operation, ensure that the switch is on, and the configuration is correct.
  - The execution time of the PFCm22 instruction is 20 ms. With 50 PFCm22 function blocks loaded, the total execution time is estimated to reach approximately 1 second. Loading 364 PFCm22 function blocks will reach the controller's resource limit. During operation, configure the scheduling task cycle properly. For example, if 10 PFCm22 are required to perform loop control simultaneously, with a block calculation time of 200 ms. The task cycle must be at least twice this value, that is, 400 ms, to ensure the CPU load does not become too high.

18.3.2.11 PRBS

A pseudo-random binary sequence (PRBS) is a reproducible binary sequence generated according to certain rules.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PRBS	Pseudo-Random Binary Sequence	FB		PRBS_Instance_0(Enable := , Ts := , AmpModel := , TypeModel := , Seed := , Amp := , ERR => , Out =>);

- **Related variables**

Input variables

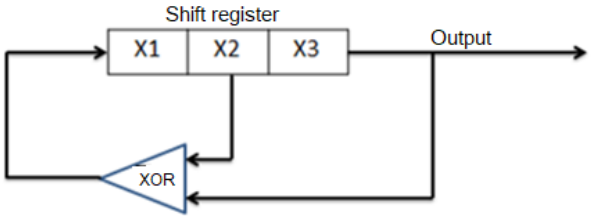
Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable switch	BOOL	0, 1	0	Enable switch. 0: Off (disabled) 1: On (enabled) The switch must be On when in use.
Ts	Periodic	UINT	>=1	1	Generation period of the pulse signal, that is, a multiple of the task cycle.
AmpModel	Amplitude type	UINT	0, 1	0	0: Unipolar (positive-zero). The output pulses can be switched between 0 and a positive amplitude. 1: Bipolar (positive-negative). The output pulses can be switched between a positive and a negative amplitude.
TypeModel	PRBS type	UINT	0, 1, 2, 3	1	Type mode. 0: PRBS7 1: PRBS15 2: PRBS23 3: PRBS31
Seed	Initial seed	UDINT	1 to 65535	1	Initial seed. It must not be 0.
Amp	Amplitude	REAL	Follows data type	1	Amplitude.

Output variables

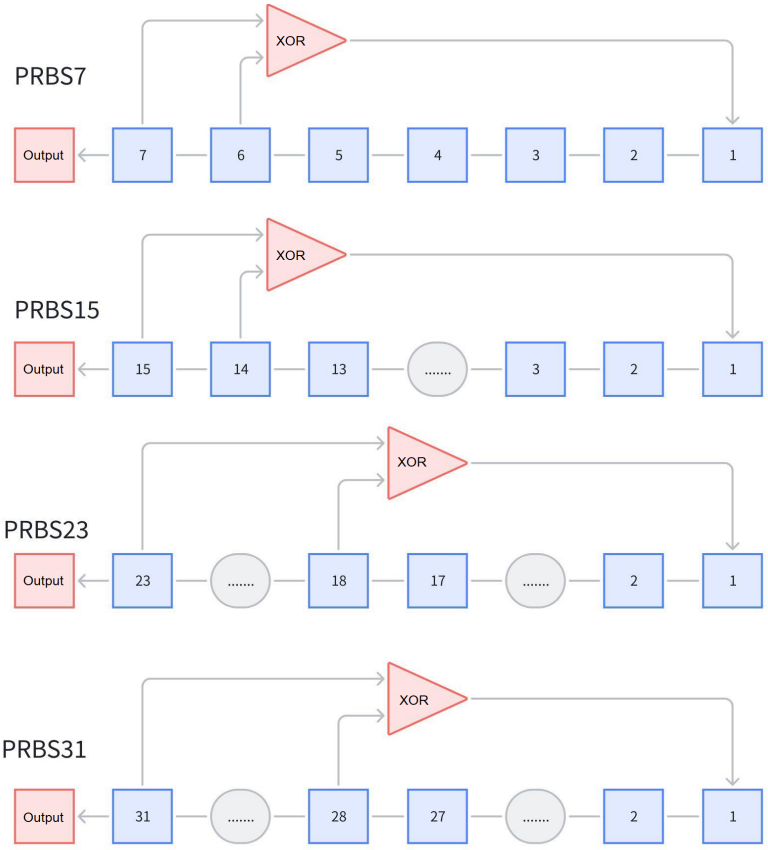
Output Variable	Name	Data Type	Valid Range	Default Value	Description
Out	PRBS output	REAL	Follows data type	0	PRBS output.
ERR	Fault code	DINT	Follows data type	0	0 indicates normal operation. A non-zero value indicates an error.

- **Function**

  PRBS patterns are generated by a PRBS pattern generator. A simplified schematic of a typical generator is shown below. It mainly consists of shift registers and XOR gates.

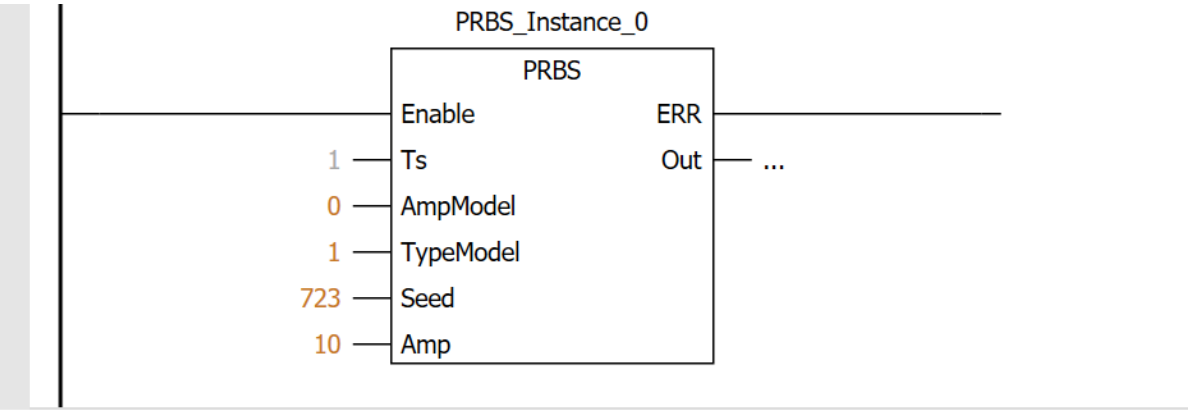


  PRBS signals can be classified into several types, such as PRBS7, PRBS15, PRBS23, and PRBS31. The algorithm primarily implements these four types. The key difference between these types lies in the XOR bit indices, and there are also some variations in the implementation methods.



- **Program example**

  The PRBS component can generate specific pulse waveforms through proper parameter configuration, as shown below:



18.3.2.12 PFC11

PFCs are simplified predictive function controller (PFC11) components for single-input, single-output systems, primarily used in 1x1 control scenarios such as temperature heating control.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PFC11	Single-loop advanced control	FB	PFC11_Instance_0 PFC11 SW SP PV MV Km Tm Dm Nmodel DDVTrend DDV_T PFC11_Config PFC11_AuTu PFC11_ReTe PFC11_OPT State ERR OP AT_ST PFC11_Output	PFC11_Instance_0(SW := , SP := , PV := , MV := , Km := , Tm := , Dm := , Nmodel := , DDVTrend := , DDV_T := , PFC11_Config := , PFC11_AuTu := , PFC11_ReTe := , PFC11_OPT := , State => , ERR => , OP => , AT_ST => , PFC11_Output =>);

- **Related variables**

  Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
SW	Operating switch	DINT	0, 1	0	0: Component off. 1: Component on. Ensure the component is in the on state when in use.
SP	Setpoint	REAL	Follows data type	0	Setpoint for the controlled variable, which defines the control target of the controller. If properly configured, the actual control result (PV) must closely follow the SP.
PV	Measured value	REAL	Follows data type	0	Input value of the controlled variable measurement, which must be provided by an external physical object or simulation object.
MV	Measured value of the valve position	REAL	Follows data type	0	Measured value of the manipulated variable. The actual output value must be written back to this variable.
Km	Model gain	REAL	Follows data type	1	An algorithm parameter that describes the response gain of the controlled object. It must be determined based on the influence of the output OP on the controlled object. The greater this influence, the larger the gain value.
Tm	Model time constant	REAL	[0, 10000]	1	An algorithm parameter that describes the response time of the controlled object. The greater the inertia of the controlled object, the slower the response, and the larger this value.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Dm	Model hysteresis	REAL	[0, 10000]	0	An algorithm parameter that represents the response hysteresis of the controlled object. The greater the hysteresis of the object to input changes, the larger this value.
Nmodel	Model length	DINT	[1, 200]	100	The model length is used to adjust the control effect. The larger this value, the longer the effective prediction horizon, and the further the future trends can be predicted.
DDVTrend	Future disturbance sequence	ARRAY [0..199] OF REAL	Follows data type	-	Disturbance sequence ge nerated by the PFC_DV component. When using this variable, pass its "DDVTrend" output to this variable. It is used to predict future disturbances so that the controller can take proactive action to suppress them.
DDV_T	Disturbance increment	REAL	Follows data type	-	Disturbance increment value, generated by the PFC_DV input interface of the same name and used for disturbance prediction and feedback correction.
PFC11_Config	Input parameter structure	PFC11_Config	Follows data type	-	Input parameter configuration structure.
PFC11_AuTu	Auto-tuning parameter structure	PFC11_AuTu	Follows data type	-	Auto-tuning parameter configuration structure.
PFC11_ReTe	Relay test structure	PFC11_ReTe	Follows data type	-	Relay test structure.
PFC11_OPT	Closed-loop OP test structure	PFC11_OPT	Follows data type	-	Closed-loop OP test structure.

PFC11_Config structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Ts	Sampling cycle	REAL	(0, 10000]	1	This value represents the control and tuning cycle. The value must be greater than 0. Ensure that this value is not set too low (below the operation cycle). If Ts is less than 0, the algorithm will automatically terminate, the switch will turn off, and the output will be held.
Model	Model type	DINT	0, 1, 2	1	Model types include: 0: Non-integral. 1: Pseudo-integral. 2: Pure integral. **Note:** The algorithm internally distinguishes between non-integral and integral models. The logic for pseudo-integral models follows the same rules as those for integral models. When the model selection is not 0, 1, or 2, the algorithm terminates, the switch is turned off, and the output will be held.
COF	Filter coefficient	REAL	[0, 1]	0.9	An algorithm parameter that affects the performance of feedback correction. The feedback value balances the measurement for the current cycle with the model prediction. The magnitude of the COF determines whether the algorithm places greater weight on the measured value or the predicted value in the feedback loop. • When set to 0, the feedback value uses the model prediction, and no feedback correction is applied. • When set to 1, the feedback value uses the measured value. • For other values, an average is calculated.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
COP	Performance coefficient	REAL	(0, 10000]	1	An algorithm parameter that affects the control performance. Increasing the value speeds up the system response to some extent, while decreasing it slows down the system response to some extent.
Tref	Closed-loop reference time	REAL	(0, 10000]	1	An algorithm parameter that affects the control performance. It is used to reduce the system response speed. Increasing this value slows down the system response to some extent, while decreasing it will speed up the system response to some extent. If the value exceeds the limit, it will be capped at 1e-6.
I_COF	Integral deviation correction coefficient	REAL	[0, 1]	0.1	Correction coefficient for integral speed. This value will take effect when the integral model is used. The larger the value, the faster the integral model speed operates in the algorithm, resulting in greater changes in future prediction, and thus causing output changes. Generally, fine-tuning this value after reaching steady state can reduce steady-state fluctuations.
DOPmax	Output increment limit	REAL	>=0	100	Sets the upper limit of output increment.
OP_Low_Val	Output valid lower limit	REAL	Follows data type	0	When the value falls below this output valid lower limit, the switch will be turned off and the control will be disabled.
OP_High_Val	Output valid upper limit	REAL	Follows data type	100	When the value exceeds this output valid upper limit, the switch will be turned off and the control will be disabled.
OP_Low_Ope	Output operation lower limit	REAL	Follows data type	0	When the value falls below this output operation lower limit, a warning is issued and the output is clamped to this limit.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
OP_High_Ope	Output operation upper limit	REAL	Follows data type	100	When the value exceeds this output operation upper limit, a warning will be issued and the output is clamped to this limit.
SP_Low_Val	Setpoint valid lower limit	REAL	Follows data type	0	Valid lower limit of the setpoint. When the value falls below this lower limit, the switch will be turned off and the control will be disabled.
SP_High_Val	Setpoint valid upper limit	REAL	Follows data type	100	Valid upper limit of the setpoint. When the value exceeds this upper limit, the switch will be turned off and the control will be disabled.
SP_Low_Ope	Setpoint operation lower limit	REAL	Follows data type	0	Operation lower limit of the setpoint. When the value falls below this lower limit, the system will display a warning and limit the value to the lower limit.
SP_High_Ope	Setpoint operation upper limit	REAL	Follows data type	100	Operation upper limit of the setpoint. When the value exceeds this upper limit, the system will display a warning and limit the value to the upper limit.
SP_DMax	Setpoint increment upper range	REAL	>=0	0.1	Algorithm parameter that affects the control effect. The system is considered stable when PV is within the range SP + SP_DMax.
SP_DMin	Setpoint increment lower range	REAL	<=0	-0.1	Algorithm parameter that affects the control effect. The system is considered stable when PV is within the range SP + SP_DMin.
Cost_SS	Speed steady state weight	REAL	>=0	0	In an integral model, this weight determines how quickly the speed converges to steady state. A larger value slows down the convergence. If set too large, the output may remain unchanged.
TrefModel	Setpoint reference trajectory mode	DINT	0, 1, 2	0	Setpoint reference trajectory mode. 0: Normal control. 1: Slope control. 2: Range control.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
SPRatio	Setpoint filter coefficient	REAL	0~1	1	Setpoint filter coefficient.
DeltaCV	Maximum change rate of the controlled variable	REAL	[0, 10000]	100	This algorithm parameter defines the maximum change value of CV during prediction. Invalid values are limited to 100.

PFC11_AuTu structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
AT_Start	Auto-tuning switch	DINT	0, 1	0	Auto-tuning switch. If this function is used, set AT_Start to 1.
AT_Model	Tuning model type	DINT	Follows data type	0	Tuning model types include: 0: Non-integral. 1: Pseudo-integral. 2: Pure integral. The auto-tuning function outputs the corresponding model parameters according to the model type. It is recommended to select according to the actual characteristics of the controlled object.
AT_OPInit	Initial OP output for tuning	REAL	Follows data type	0	OP output value at initial steady state. The controlled object must reach a stable state at this output.
AT_OPTarget	Target OP output for tuning	REAL	Follows data type	100	This value is the step output value at the start of tuning. The controller will perform a step change from "AT_OPInit" to "AT_OPTarget" to excite the response of the controlled object.
AT_PVTarget	PV value at tuning end	REAL	Follows data type	100	This value is the target PV value to be achieved during tuning. The tuning process completes when the measured value (PV) of the controlled object reaches this value after tuning begin. Then, the tuning result will be output.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
AT_PVDeltTar	Early termination threshold	REAL	Follows data type	0	When the PV value reaches AT_PVTarget – AT_PVDeltTar, the tuning process terminates early and outputs the tuning result.
AT_InitRate	Initial steady-state slope	REAL	Follows data type	0	PV measured value slope that allows the step response tuning to start.

PFC11_ReTe structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
RT_Start	Relay test switch	DINT	0, 1	0	If this function is used, set RT_Start to 1.
RT_Direction	Input-output action direction	DINT	0, 1	0	0: Direct acting. When the output increases, the measured value increases. 1: Reverse acting. When the output increases, the measured value decreases.
RT_OP_Init	Initial value for the test	REAL	Follows data type	0	OP output at the start of the test.
RT_OP_DLow	Test output lower range	REAL	< 0	-1	Lower limit of the OP oscillation output at the start of the test = RT_OP_Init + RT_OP_DLow.
RT_OP_DHigh	Test output upper range	REAL	> 0	1	Upper limit of the OP oscillation output at the start of the test = RT_OP_Init + RT_OP_DHigh.
RT_PV_Target	Step measured value	REAL	Follows data type	100	Target PV measured value after the test starts.
RT_PV_DLow	Oscillation lower range	REAL	< 0	-1	Lower limit of PV oscillation switching after the test starts = RT_PV_Target + RT_PV_DLow.
RT_PV_DHigh	Oscillation upper range	REAL	> 0	1	Upper limit of PV oscillation switching after the test starts = RT_PV_Target + RT_PV_DHigh.
RT_Warn_Low	Warning lower range	REAL	Follows data type	0	Test protection lower range. If the value falls below this range, the test stops. Restart the test after reconfiguring the parameters.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
RT_Warn_High	Warning upper range	REAL	Follows data type	100	Test protection upper range. If the value exceeds this range, the test stops. Restart the test after reconfiguring the parameters.

PFC11_OPT structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
OPT_Start	Closed-loop OP test switch	DINT	0, 1	0	If this function is used, set OPT_Start to 1.
OPT_Type	PRBS type	DINT	0, 1, 2, 3	0	Type mode: 0: PRBS7 1: PRBS15 2: PRBS23 3: PRBS31
OPT_TS	PRBS cycle	DINT	>=1	1	Integer multiple of the task call cycle.
OPT_Amp	PRBS amplitude	REAL	>=0	0	Amplitude of the superimposed PRBS cycle during closed-loop testing.
OPT_Seed	PRBS initial seed	DINT	[1, 65535]	100	The seed value determines the initial value of the generated sequence.
OPT_Warn_Low	Warning lower range	REAL	Follows data type	0	Test protection lower range. If the value falls below this range, the test stops. Restart the test after reconfiguring the parameters.
OPT_Warn_High	Warning upper range	REAL	Follows data type	100	Test protection upper range. If the value exceeds this range, the test stops. Restart the test after reconfiguring the parameters.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
State	Switch state	DINT	0, 1	-	Indicates the actual state of the controller. 0: Normal control. 1: Switched to feedforward.
ERR	Fault code	DINT	Follows data type	-	0 for normal operation; non-zero when an error occurs. See PFC11 fault codes.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
OP	Valve position output value	REAL	Follows data type	-	Controller output value.
AT_ST	Tuning result	DINT	0, 1, 2, 3	-	Tuning state display. 0: Not tuned 1: Coarse tuning in progress 2: Fine tuning in progress 3: Tuning completed
PFC11_Output	Output parameter structure	PFC11_Output	Follows data type	-	Output parameter configuration structure.

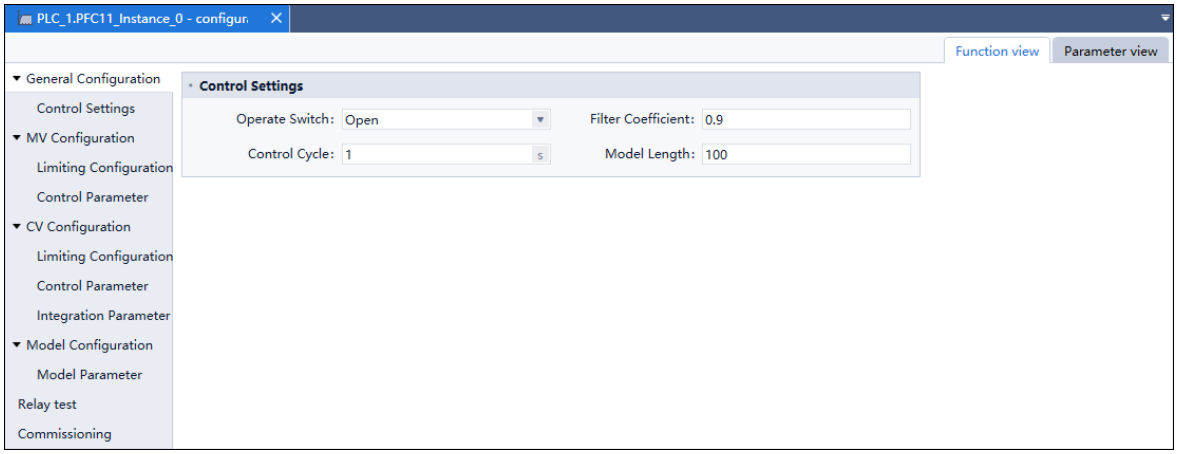
PFC11_Output structure configuration

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Version	Version information	ARRAY [0..3] OF DINT	Follows data type	-	Used to record internal version information.
DOP	Valve position output increment	REAL	Follows data type	-	Increment of the output value OP.
PV_Pred_t	Estimated value	REAL	Follows data type	-	Estimated value, that is, the response value of the controlled object, calculated by the internal model using historical data.
PV_Pred_tf	Predicted value	REAL	Follows data type	-	Predicted value, that is, the estimated value corrected based on the actual measured value PV.
PV_Pred_e	Prediction error	REAL	Follows data type	-	Difference between the actual measured value PV and the predicted value.
PV_Pred_eSum	Cumulative prediction error	REAL	Follows data type	-	Cumulative sum of prediction errors.
DCV_S	Integral deviation speed.	REAL	Follows data type	-	Integral deviation speed.
PV_Pred	Open-loop prediction sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of future measured values predicted when the current cycle is not applied.
Tref_Step	Reference trajectory sequence	ARRAY [0..199] OF REAL	Follows data type	-	A trajectory sequence generated by the Tref parameter.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
OP_Low_Ope_Past	Output lower limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical operation lower limits for the controller output OP.
OP_High_Ope_Past	Output upper limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical operation upper limits for the controller output OP.
OP_Low_Ope_Future	Output lower limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of predicted operation lower limits for the controller output at future sampling instants, computed after the current cycle has been applied.
OP_High_Ope_Future	Output upper limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of predicted operation upper limits for the controller output at future sampling instants, computed after the current cycle has been applied.
SP_Past	Setpoint history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of the historical setpoints.
SP_Future	Setpoint future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of future setpoints predicted by the algorithm after the current cycle has been applied.
SP_Low_Ope_Past	Setpoint lower limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of historical operation lower limits for the setpoint.
SP_High_Ope_Past	Setpoint upper limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of historical operation upper limits for the setpoint.
SP_Low_Ope_Future	Setpoint lower limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of predicted operation lower limits for the setpoint at future sampling instants, computed after the current control cycle has been applied.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
SP_High_Ope_Future	Setpoint upper limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of predicted operation upper limits for the setpoint at future sampling instants, computed after the current control cycle has been applied.
SP_Href	Upper reference trajectory sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the upper reference trajectory of the setpoint.
SP_Lref	Lower reference trajectory sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the lower reference trajectory of the setpoint.
PV_Past	Measured value history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of the historical measured values.
OP_Past	Output value history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of the historical output values.
OP_Future	Output value future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of future output values predicted by the algorithm after the current cycle has been applied.
AT_Km	Tuning parameter Km	REAL	Follows data type	-	Model gain result obtained from auto-tuning.
AT_Tm	Tuning parameter Tm	REAL	> 0	-	Model time constant result obtained from auto-tuning.
AT_Dm	Tuning parameter Dm	REAL	>=0	-	Model hysteresis parameter result obtained from auto-tuning.

- **Function**
  - PFC11 is a simplified 1x1 single-input, single-output predictive function controllers. It includes one manipulated variable (output via OP) and one controlled variable (measured via PV), and support a feedforward input channel (for use with the PFC_DV function block). The PFC11 component features a built-in auto-tuning function to obtain model parameters of the controlled object. This component is primarily used for rapid deployment in single-input single-output processes.
  - The PFC11 function block provide easy-to-use interfaces for parameter configuration and tuning/commissioning. After adding the PFC11 component, select "Control Objects -> PFC" in the left project tree, and then double-click the PFC11 instance under the PFC node to open the configuration and tuning interface, as shown in the figure below.

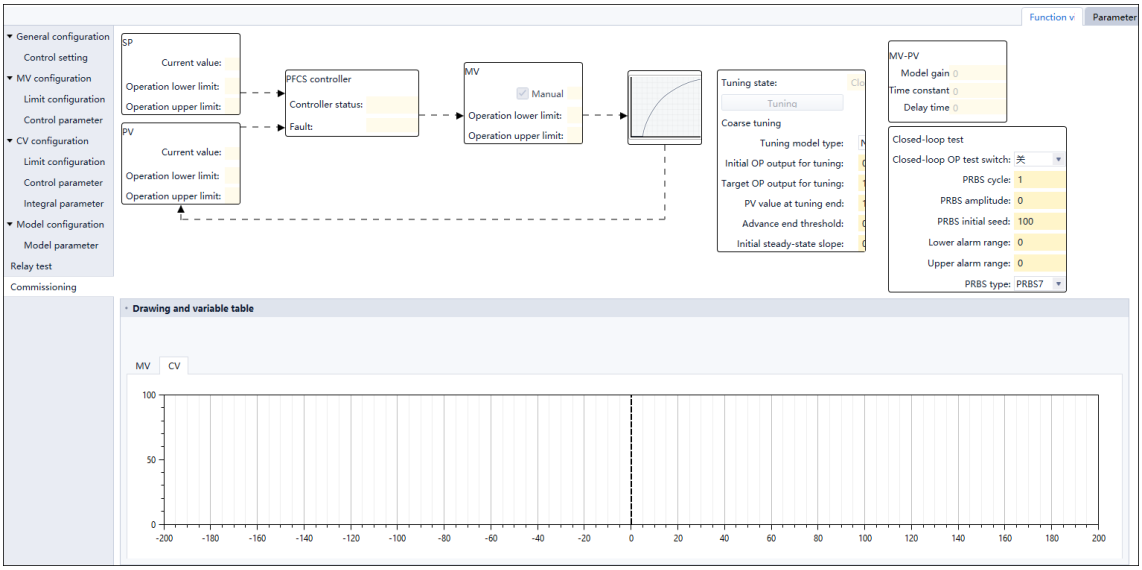


- Parameter configuration

  The parameter configuration of the PFC11 instruction includes general configuration, MV configuration, CV configuration, model configuration and relay test. For detailed parameter descriptions, see Relevant variables.

- Parameter auto-tuning

  The PFC11 instruction includes a built-in parameter auto-tuning function, automatically obtaining the model parameters (Km, Tm, and Dm) of the controlled object. Click the "Tuning" button on the "Commissioning" page to start parameter tuning.

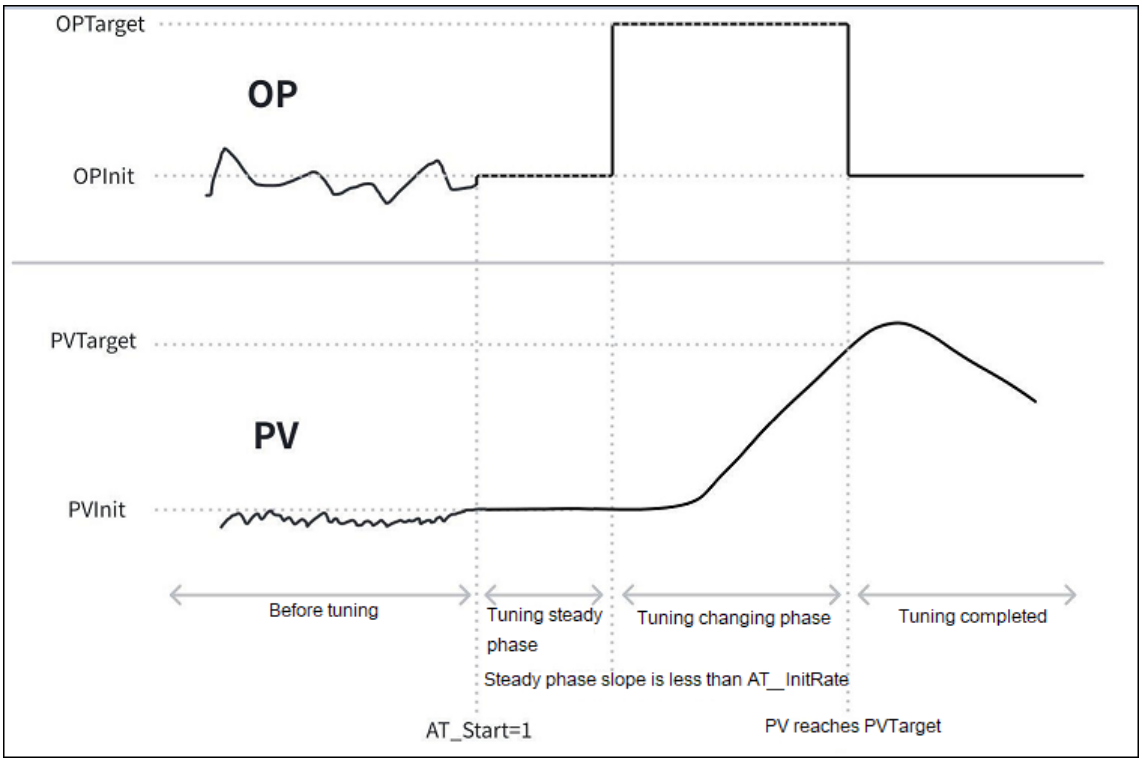


Because the controller internal variable array has a capacity of only 200, ensure that the duration from the OP output step to the completion of tuning stays within 200 control cycles. 100 to 150 cycles is recommended. An excessively long or short tuning process may lead to tuning failure. Furthermore, the tuning process does not require a complete steady state after the step. Generally, it only needs to capture the portion of the controlled object's response with the maximum slope.

The tuning process is as follows:

1. Select the model type of the tuning object.
2. Configure the tuning parameter "Initial OP output for tuning" (AT_OPInit). AT_OPInit is the OP output value at the initial steady state. Ensure that the controlled object reaches a basic stable state at this output. Open-loop control can be used before the tuning process to stabilize the controlled object at this output, then switch to auto-tuning. The initial steady state is critical for the tuning results. When it is difficult to achieve the initial steady state, the "Initial steady-state slope" (AT_InitRate) parameter can be used to appropriately relax the steady-state requirements.

3. Configure the tuning parameter "Target OP output for tuning" (AT_OPTarget). This value is the step output value at the start of tuning. The controller will perform a step change from "AT_OPInit" to "AT_OPTarget" to excite the response of the controlled object.
4. Configure the tuning parameter "PV value at tuning end" (AT_PVTarget) and "Advance end threshold" (AT_PVDeltTar). The tuning process completes when the measured value (PV) of the controlled object reaches this value after tuning begin. Pay special attention to the response polarity of the controlled object. Ensure that the measured value reaches AT_PVTarget after tuning starts, rather than the PV changing in the opposite direction after the OP step change.
5. Click the "Tuning" button.
6. After tuning ends, click "Confirm" or "Cancel" to select parameters, and automatically switch to control mode.

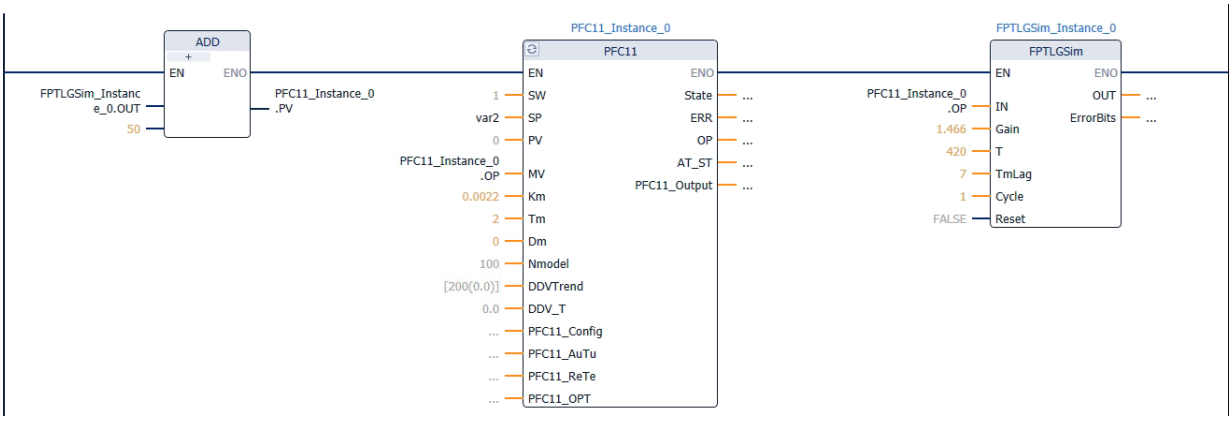


- Program example

  Here, the control simulation between the PFC11 instruction and the first-order inertia plus pure delay simulation instruction (FPTLGSim) is used as an example. The parameter configuration and control process are as follows:

1. Parameter configuration
   Configure the PFC11 instruction parameters based on the characteristics of the control object (Model parameters: Km, Tm, Dm and so on; controller parameters: COP, COF, etc.).
2. Control process
   The PFC11 instruction responds to the output (PV) of the FPTLGSim object, obtains the control quantity (OP) output, and applies it to the original control object, forming a control loop.
3. Control expectation
   If all parameters are configured properly, the controller will eventually keep the output value (PV) of the controlled object near the setpoint (SP).

   LD language code example:

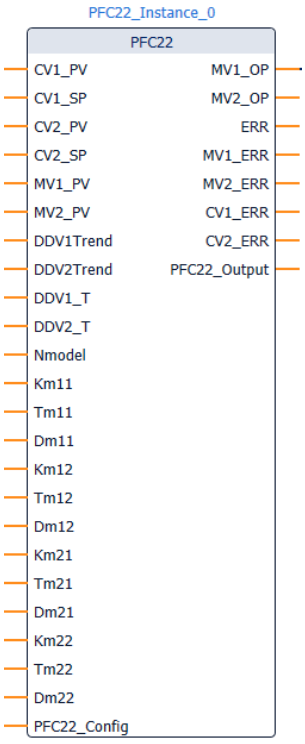


- Precautions
  - The module execution cycle must be consistent with the function block cycle parameter Ts.
  - During operation, ensure that the switch is on, and the configuration is correct.
  - The execution time of the PFC11 instruction is 10 ms. With 50 PFC11 function blocks loaded, the total execution time is estimated to reach 500 ms. During operation, configure the scheduling task cycle properly to avoid high CPU load.

18.3.2.13 PFC22

This instruction is a dual-input dual-output control instruction. It is primarily used in 2x2 scenarios, such as the coupled control of temperature 1 and temperature 2 by heater 1 and heater 2, involving mutual coupling between the two temperature zones.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PFC22	Multivariable input/ output advanced control	FB		PFC22_Instance_0(CV1_PV := , CV1_SP := , CV2_PV := , CV2_SP := , MV1_PV := , MV2_PV := , DDV1Trend := , DDV2Trend := , DDV1_T := , DDV2_T := , Nmodel := , Km11 := , Tm11 := , Dm11 := , Km12 := , Tm12 := , Dm12 := , Km21 := , Tm21 := , Dm21 := , Km22 := , Tm22 := , Dm22 := , PFC22_Config := , MV1_OP => , MV2_OP => , ERR => , MV1_ERR => , MV2_ERR => , CV1_ERR => , CV2_ERR => , PFC22_Output =>);

- **Related variables**

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
CV1_PV	Measured value of the controlled variable 1	REAL	Follows data type	0	Input value of the controlled variable 1 measurement, which must be provided by an external physical object or simulation object.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
CV1_SP	Setpoint value of the controlled variable 1	REAL	Follows data type	0	Setpoint value of the controlled variable 1, which determines the control objective of the controller so that the final actual control result (PV) must closely follow CV1_SP.
CV2_PV	Measured value of the controlled variable 2	REAL	Follows data type	0	Input value of the controlled variable 2 measurement, which must be provided by an external physical object or simulation object.
CV2_SP	Setpoint value of the controlled variable 2	REAL	Follows data type	0	Setpoint value of the controlled variable 2, which determines the control objective of the controller so that the final actual control result (PV) must closely follow CV2_SP.
MV1_PV	Measured value of the manipulated variable 1	REAL	Follows data type	0	Measured value of manipulated variable 1. The actual output value 1 must be written back to this variable. In a heating application: if only algorithmic control is active, write back the value of MV1_OP; if an external input is present, write back the actual value sent to the actuator.
MV2_PV	Measured value of the manipulated variable 2	REAL	Follows data type	0	Measured value of manipulated variable 2. The actual output value 2 must be written back to this variable. In a heating application: if only algorithmic control is active, write back the value of MV2_OP; if an external input is present, write back the actual value sent to the actuator.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
DDV1Trend	Disturbance sequence 1	ARRAY [0..199] OF REAL	Follows data type	-	Disturbance sequence 1 generated by the dv series component. When using this variable, pass its output "DDVTrend" to this variable. It is used to predict future disturbances so that the controller can take proactive action to suppress them. This variable corresponds to CV1 disturbance.
DDV2Trend	Disturbance sequence 2	ARRAY [0..199] OF REAL	Follows data type	-	Disturbance sequence 2 generated by the dv series component. When using this variable, pass its output "DDVTrend" to this variable. It is used to predict future disturbances so that the controller can take proactive action to suppress them. This variable corresponds to CV2 disturbance.
DDV1_T	Disturbance increment 1	REAL	Follows data type	0	Disturbance increment value, generated by the PFC_DV input interface of the same name and used for disturbance prediction and feedback correction.
DDV2_T	Disturbance Increment 2	REAL	Follows data type	0	Disturbance increment value, generated by the PFC_DV input interface of the same name and used for disturbance prediction and feedback correction.
Nmodel	Model length	DINT	[1, 200]	100	The model length is used to adjust the control effect. The larger this value, the longer the effective prediction horizon, and the further the future trends can be predicted.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Km11	Model MV1-CV1 gain	REAL	Follows data type	1	An algorithm parameter that affects the control performance. Increasing the value speeds up the system response to some extent, while decreasing it slows down the system response to some extent.
Tm11	Model MV1-CV1 time constant	REAL	(0, 10000]	1	An algorithm parameter that affects the control performance. The smaller the Tm value, the fewer cycles required to reach a steady state. If the value exceeds the limit, it will be limited to 1e-6.
Dm11	Model MV1-CV1 hysteresis	REAL	[0, 10000]	0	An algorithm parameter that affects the control performance. The delay expressed in control cycles is Dm/Ts. If the value exceeds the limit, it will be limited to 0.
Km12	Model MV2-CV1 gain	REAL	Follows data type	1	An algorithm parameter that affects the control performance. Increasing the value speeds up the system response to some extent, while decreasing it slows down the system response to some extent.
Tm12	Model MV2-CV1 time constant	REAL	(0, 10000]	1	An algorithm parameter that affects the control performance. The smaller the Tm value, the fewer cycles required to reach a steady state. If the value exceeds the limit, it will be limited to 1e-6.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Dm12	Model MV2-CV1 hysteresis	REAL	[0, 10000]	0	An algorithm parameter that affects the control performance. The delay expressed in control cycles is Dm/Ts. If the value exceeds the limit, it will be limited to 0.
Km21	Model MV1-CV2 gain	REAL	Follows data type	1	An algorithm parameter that affects the control performance. Increasing the value speeds up the system response to some extent, while decreasing it slows down the system response to some extent.
Tm21	Model MV1-CV2 time constant	REAL	(0, 10000]	1	An algorithm parameter that affects the control performance. The smaller the Tm value, the fewer cycles required to reach a steady state. If the value exceeds the limit, it will be limited to 1e-6.
Dm21	Model MV1-CV2 hysteresis	REAL	[0, 10000]	0	An algorithm parameter that affects the control performance. The delay expressed in control cycles is Dm/Ts. If the value exceeds the limit, it will be limited to 0.
Km22	Model MV2-CV2 gain	REAL	Follows data type	1	An algorithm parameter that affects the control performance. Increasing the value speeds up the system response to some extent, while decreasing it slows down the system response to some extent.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Tm22	Model MV2-CV2 time constant	REAL	(0, 10000]	1	An algorithm parameter that affects the control performance. The smaller the Tm value, the fewer cycles required to reach a steady state. If the value exceeds the limit, it will be limited to 1e-6.
Dm22	Model MV2-CV2 hysteresis	REAL	[0, 10000]	0	An algorithm parameter that affects the control performance. The delay expressed in control cycles is Dm/Ts. If the value exceeds the limit, it will be limited to 0.
PFC22_Config	Input parameter structure	PFC22_Config	Follows data type	-	Input parameter configuration structure.

PFC22_Config structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Ts	Sampling cycle	REAL	(0, 10000]	1	This value determines the control cycle. This value must be greater than 0, and can not be set too low. An excessively small value may cause a calculation timeout and degrade control performance. If Ts is less than 0, the algorithm will automatically terminate, the switch will turn off, and the output will be held.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Model1	Model type 1	DINT	0, 1, 2	1	Model for MV1-CV1. Model type: 0: Non-integral. 1: Pseudo-integral. 2: Pure integral. Currently, types 0 (Non-integral) and 1 (Pseudo-integral) are primarily used. Type 2 (Pure integral) is not implemented. In practice, integral models are typically handled as pseudo-integral. When the model selection is not 0, 1, or 2, the algorithm terminates, the switch is turned off, and the output will be held.
Model2	Model type 2	DINT	0, 1, 2	1	Model for MV2-CV1. Model type: 0: Non-integral. 1: Pseudo-integral. 2: Pure integral. Currently, types 0 (Non-integral) and 1 (Pseudo-integral) are primarily used. Type 2 (Pure integral) is not implemented. In practice, integral models are typically handled as pseudo-integral. When the model selection is not 0, 1, or 2, the algorithm terminates, the switch is turned off, and the output will be held.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Model3	Model type 3	DINT	0, 1, 2	1	Model for MV1-CV2. Model type: 0: Non-integral. 1: Pseudo-integral. 2: Pure integral. Currently, types 0 (Non-integral) and 1 (Pseudo-integral) are primarily used. Type 2 (Pure integral) is not implemented. In practice, integral models are typically handled as pseudo-integral. When the model selection is not 0, 1, or 2, the algorithm terminates, the switch is turned off, and the output will be held.
Model4	Model type 4	DINT	0, 1, 2	1	Model for MV2-CV2. Model type: 0: Non-integral 1: Pseudo-integral 2: Pure integral Currently, types 0 (Non-integral) and 1 (Pseudo-integral) are primarily used. Type 2 (Pure integral) is not implemented. In practice, integral models are typically handled as pseudo-integral. When the model selection is not 0, 1, or 2, the algorithm terminates, the switch is turned off, and the output will be held.
MV1_Config	MV1 variable input parameter structure	PFC22_MVConfig	Follows data type	-	MV1 variable input parameter configuration structure.
MV2_Config	MV2 variable input parameter structure	PFC22_MVConfig	Follows data type	-	MV2 variable input parameter configuration structure.
CV1_Config	CV1 variable input parameter structure	PFC22_CVConfig	Follows data type	-	CV1 variable input parameter configuration structure.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
CV2_Config	CV2 variable input parameter structure	PFC22_CVConfig	Follows data type	-	CV2 variable input parameter configuration structure.
Single_SW	Single output switch	DINT	0, 1	0	Single output switch. If this switch is set to ON, it ensures that when the two MV gains have opposite signs, only one output direction is active. This is typically used to save energy.

PFC22_MVConfig structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
MV_SW	Manipulated variable switch	DINT	0, 1	1	MV switch; when this switch is off, the control is disabled. Invalid values are automatically set to 0.
MV_High_Val	Manipulated variable valid upper limit	REAL	Follows data type	200	Valid upper limit. When the value exceeds the valid upper limit, the switch will be turned off and the control will be disabled.
MV_Low_Val	Manipulated variable valid lower limit	REAL	Follows data type	0	Valid lower limit. When the value falls below the valid lower limit, the switch will be turned off and the control will be disabled.
MV_High_Ope	Operation upper limit of the manipulated variable	REAL	Follows data type	100	Operation upper limit. When the value exceeds this operation upper limit, a warning will be issued and the value will be clamped to this operation upper limit.
MV_Low_Ope	Operation lower limit of the manipulated variable	REAL	Follows data type	0	Operation lower limit. When the value falls below this operation lower limit, a warning will be issued and the value will be clamped to this operation lower limit.
MV_DMax	Maximum positive increment for the manipulated variable	REAL	[0, 10000]	100	Defines the maximum allowable positive increment of the output. Invalid input values are limited to 100.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
MV_DMin	Maximum negative increment for the manipulated variable	REAL	[-10000, 0]	-100	Defines the maximum allowed negative increment of the output. Invalid input values are limited to -100.
MV_COP	Manipulated variable performance coefficient	REAL	[0, 10000]	1	An algorithm parameter that affects the control performance. Decreasing the value speeds up the system response to some extent, while increasing it slows down the system response to some extent. If this value exceeds the upper limit, it will be limited to the upper limit. If the value falls below the lower limit, it will be limited to the lower limit.
MV_Irv	Ideal injection value for the manipulated variable	REAL	Follows data type	1	The value that the corresponding MV output eventually converges to.
MV_Irv_C	Weight for manipulated variable ideal injection value	REAL	[0, 10000]	1	The larger the value, the closer the MV will converge to MV_Irv.
MV_Irv_SW	Ideal injection value switch	DINT	0, 1	0	When set to 0, the ideal injection value is inactive. When used, the value must be set to 1. Invalid values are automatically set to 0.

PFC22_CVConfig structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
CV_SW	Controlled variable switch	DINT	0, 1	1	Controlled variable switch. When the switch is off, the control is disabled. When used, it must be set to 1. Invalid values are automatically set to 0.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Cost_SS	Speed steady state weight	REAL	[0, 10000]	0	Speed steady state weight. In an integral model, this weight determines how quickly the speed converges to steady state. A larger value slows down the convergence. If set too large, the output may remain unchanged.
TrefModel	Setpoint reference trajectory mode	DINT	0, 1, 2	0	Setpoint reference trajectory mode. 0: Normal control. 1: Slope control. 2: Range control.
SPRatio	Setpoint filter coefficient	REAL	0~1	1	Setpoint filter coefficient.
CV_ECE	Equivalent deviation	REAL	[0, 10000]	1	Determines the weight of the corresponding CV variable. A larger value gives higher priority to controlling that CV.
CV_COF	Filter coefficient	REAL	[0, 1]	0.9	An algorithm parameter that affects the control performance. The smaller this value, the greater the confidence in the model prediction. When set to 0, no feedback correction is performed. Use a value greater than 0. If this value exceeds the upper limit, it will be limited to the upper limit. If the value falls below the lower limit, it will be limited to the lower limit.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
CV_Tref	Closed-loop reference time	REAL	(0, 10000]	1	An algorithm parameter that affects the control performance. Increasing the value slows down the system response to some extent, while decreasing it speeds up the system response to some extent. If the value exceeds the limit, it will be limited to 1e-6.
I_COF	Integral deviation correction coefficient	REAL	[0, 1]	0.1	An algorithm parameter that affects the control performance. The smaller the value, the greater the confidence in the model prediction. If this value exceeds the upper limit, it will be limited to the upper limit. If the value falls below the lower limit, it will be limited to the lower limit.
CV_SP_DMax	Upper deviation range	REAL	[0, 10000]	0.1	An algorithm parameter that affects the control performance. The system is considered stable if CV_PV is within this range CV_SP + CV_SP_DMax. Invalid input values are limited to 0.
CV_SP_DMin	Lower deviation range	REAL	[-10000, 0]	-0.1	An algorithm parameter that affects the control performance. The system is considered stable if CV_PV is within this range CV_SP + CV_SP_DMin. Invalid input values are limited to 0.
DeltaCV	Maximum change rate of the controlled variable	REAL	[0, 10000]	100	This algorithm parameter defines the maximum change value of CV during prediction. Invalid values are limited to 100.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
CV_Low_Val	Valid lower limit of the controlled variable	REAL	Follows data type	0	Valid lower limit. When the value falls below the valid lower limit, the switch will be turned off and the control will be disabled.
CV_High_Val	Valid upper limit of the controlled variable	REAL	Follows data type	100	Valid upper limit. When the value exceeds the valid upper limit, the switch will be turned off and the control will be disabled.
CV_Low_Ope	Operation lower limit of the controlled variable	REAL	Follows data type	0	Operation lower limit. When the value falls below this operation lower limit, a warning will be issued and the value will be clamped to this operation lower limit.
CV_High_Ope	Operation upper limit of the controlled variable	REAL	Follows data type	100	Operation upper limit. When the value exceeds this operation upper limit, a warning will be issued and the value will be clamped to this operation upper limit.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
MV1_OP	Manipulated variable 1 output value	REAL	Follows data type	-	Output value of manipulated variable 1. This value must be output to the simulation object or actual object.
MV2_OP	Manipulated variable 2 output value	REAL	Follows data type	-	Output value of manipulated variable 2. This value must be output to the simulation object or actual object.
ERR	Fault code	DINT	Follows data type	-	For fault codes, see PFC22 fault codes.
MV1_ERR	Manipulated variable 1 fault code	DINT	Follows data type	-	Fault code of manipulated variable 1.
MV2_ERR	Manipulated variable 2 fault code	DINT	Follows data type	-	Fault code of manipulated variable 2.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
CV1_ERR	Controlled variable 1 fault code	DINT	Follows data type	-	Fault code of controlled variable 1.
CV2_ERR	Controlled variable 2 fault code	DINT	Follows data type	-	Fault code of controlled variable 2.
PFC22_Output	Output parameter structure	PFC22_Output	Follows data type	-	Output parameter structure.

PFC22_Output structure configuration

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Version	Version information	ARRAY [0..3] OF DINT	Follows data type	-	Used to obtain the version information.
NFlag	Number of errors	DINT	[0, 10]	-	Counts the number of errors.
Flag	Error information	ARRAY [0..9] OF DINT	Follows data type	-	Records detailed information of each error.
MV1_OutConfig	Manipulated variable 1 output structure	PFC22_MVOutConfig	Follows data type	-	Output structure of manipulated variable 1.
MV2_OutConfig	Manipulated variable 2 output structure	PFC22_MVOutConfig	Follows data type	-	Output structure of manipulated variable 2.
CV1_OutConfig	Controlled variable 1 output structure	PFC22_CVOutConfig	Follows data type	-	Output structure of controlled variable 1.
CV2_OutConfig	Controlled variable 2 output structure	PFC22_CVOutConfig	Follows data type	-	Output structure of controlled variable 2.

PFC22_CVOutConfig structure configuration

Output Variable	Name	Data Type	Valid Range	Default Value	Description
CV_NFlag	Controlled variable error count	DINT	[0, 10]	-	Number of recorded errors for the controlled variable
CV_Flag	Controlled variable error information	ARRAY [0..9] OF DINT	Follows data type	-	Detailed information of recorded errors for the controlled variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
CV_ST	Controlled variable switch status	DINT	0, 1	-	Status of the controlled variable switch. Initially the value equals the switch input value. When an internal error occurs, this value is set to 1.
CV_Pred_t	Estimated value of controlled variable	REAL	Follows data type	-	Estimated value, that is, the response value of the controlled object, calculated by the internal model using historical data.
CV_Pred_tf	Predicted value of controlled variable	REAL	Follows data type	-	Predicted value, that is, the corrected predicted value based on the predicted value and the actual measured value PV.
DCV_S_i	Velocity after integral deviation correction	REAL	Follows data type	-	Velocity after integral deviation correction.
DCV_S_SS	Future speed steady state value	REAL	Follows data type	-	Steady-state speed value, which is the speed value predicted after 200 cycles.
CV_PV_e	Predicted error of the controlled variable	REAL	Follows data type	-	Difference between the actual measured value and the predicted value.
CV_PV_eSum	Cumulative sum of predicted errors for the controlled variable	REAL	Follows data type	-	The cumulative sum of errors (regardless of sign). Its magnitude indicates the degree of model mismatch.
CV_Pred_t_Past	Sequence of historical estimated values for the controlled variable	ARRAY [0..199] OF REAL	Follows data type	-	Records historical values.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
CV_Pred	Open-loop prediction sequence of controlled variables	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of future values predicted when the current cycle is not applied.
CV_Future	Closed-loop prediction sequence of controlled variables	ARRAY [0..199] OF REAL	Follows data type	-	Records the sequence of future values predicted after the current cycle has been applied.
CV_PV_Past	Sequence of historical measured values for the controlled variable	ARRAY [0..199] OF REAL	Follows data type	-	Records historical measured values.
SP_Past	Setpoint history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical setpoint values, for plotting.
SP_Future	Setpoint future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of the future setpoint values, for plotting.
CV_SP_Href	Upper reference trajectory sequence	ARRAY [0..199] OF REAL	Follows data type	-	Predicted upper trajectory to be approached.
CV_SP_Lref	Lower reference trajectory sequence	ARRAY [0..199] OF REAL	Follows data type	-	Predicted lower trajectory to be approached.
CV_Low_Past	Controlled variable lower limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical operation lower limits for the controlled variable, for plotting.
CV_High_Past	Controlled variable upper limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical upper limits for the controlled variable, for plotting.
CV_Low_Future	Controlled variable lower limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future operation lower limits for the controlled variable, for plotting.

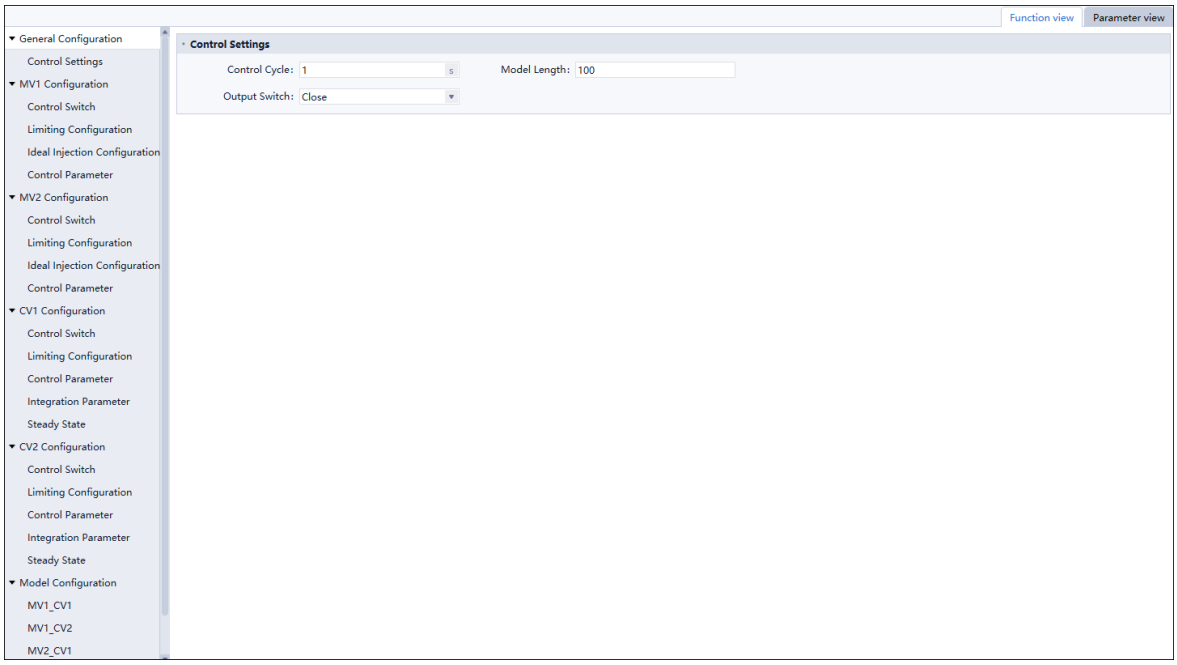
Output Variable	Name	Data Type	Valid Range	Default Value	Description
CV_High_Future	Controlled variable upper limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future operation upper limits for the controlled variable, for plotting.

PFCm22_MVOutConfig structure configuration

Output Variable	Name	Data Type	Valid Range	Default Value	Description
MV_NFlag	MV error count	DINT	0, 1	-	Number of errors counted for the manipulated variable.
MV_Flag	MV error message	ARRAY [0..9] OF DINT	[0, 10]	-	Detailed information of recorded errors for the manipulated variable.
MV_Low_Past	Output lower limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical operation lower limits for the manipulated variable, for plotting.
MV_High_Past	Output upper limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical operation upper limits for the manipulated variable, for plotting.
MV_Low_Future	Output lower limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future operation lower limits for the manipulated variable, for plotting.
MV_High_Future	Output upper limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future operation upper limits for the manipulated variable, for plotting.
MV_Past	Output value history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical measured values for the manipulated variable, for plotting.
MV_Future	Output value future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future measured values for the manipulated variable, for plotting.
MV_DOP_Past	Historical increment sequence of the valve position	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical increment values for the manipulated variable, for plotting.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
MV_DOP	Manipulated variable action increment	REAL	Follows data type	-	Increment value of the manipulated variable.
MV_ST	Manipulated variable switch state	DINT	0, 1	-	Manipulated variable switch state. Initially, it equals the switch value. This value is set to 1 when an internal error occurs.
MV_Irv_Past	Ideal injection value history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical ideal injection values, for plotting.
MV_Irv_Future	Ideal injection value future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future ideal injection values, for plotting.

- **Function**
  - PFC22 is a 2x2 dual-input, dual-output control instruction. The controlled object can support non-integrating, integrating, or pseudo-integrating responses. The instruction includes two manipulated variables (MV), two controlled variables (CV), and a feedforward input channel (for use with PFC_DV). The instruction does not rely on an external solver for optimization calculations. It is mainly intended for applications with simple coupling and does not require high-end products. It is suitable for fast control application scenarios with millisecond-level control cycles.
  - The PFC22 function block includes an intuitive interface for parameter configuration and commissioning. After adding the PFC22 component, select "Control Objects -> PFC" in the left project tree, and then double-click the PFC22 instance under the PFC node to open the configuration and tuning interface, as shown in the figure below.



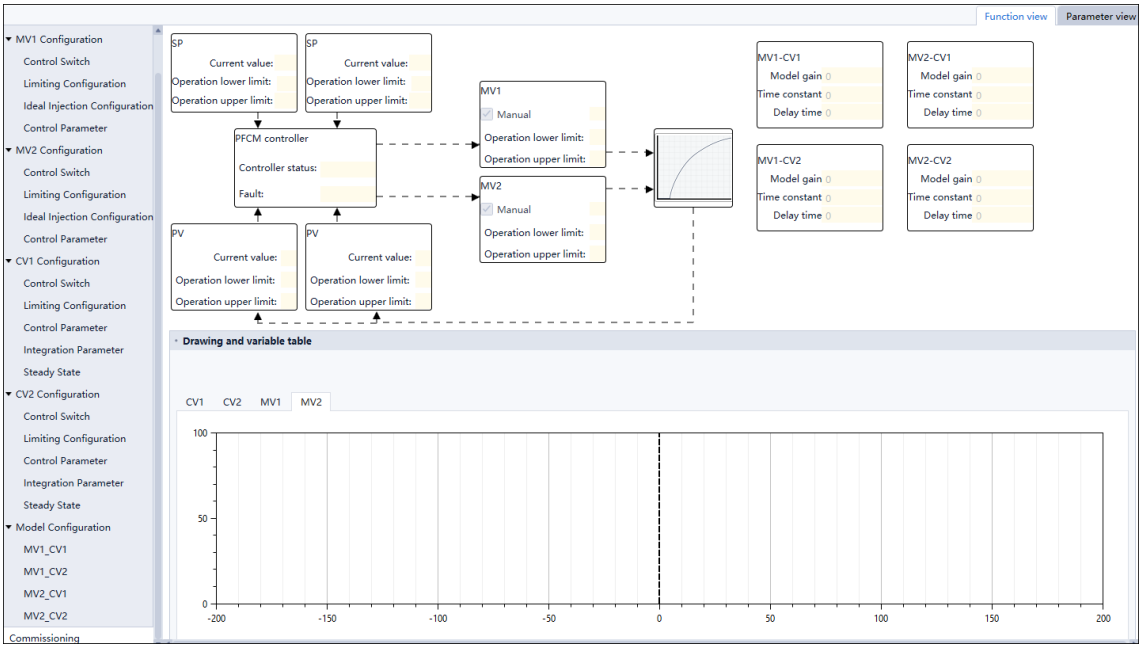
    - Parameter description

      The parameter configuration of the PFC22 instruction includes general configuration, MV

configuration, CV configuration, and model configuration. For detailed parameter descriptions, see Relevant variables.

- Commissioning

  After the parameters are configured, the control effect can be viewed directly in the commissioning interface.



- **Program example**

  Here, the two-input, two-output control simulation between the PFC22 instruction and four first-order inertia plus pure delay simulation instruction (FPTLGSim) is used as an example. The parameter configuration and control process are as follows:

  - Parameter configuration

    Configure the PFC22 instruction parameters based on the characteristics of the control object (Model parameters: Km11, Tm11, Dm11 and so on; controller parameters: COP, COF, etc.).

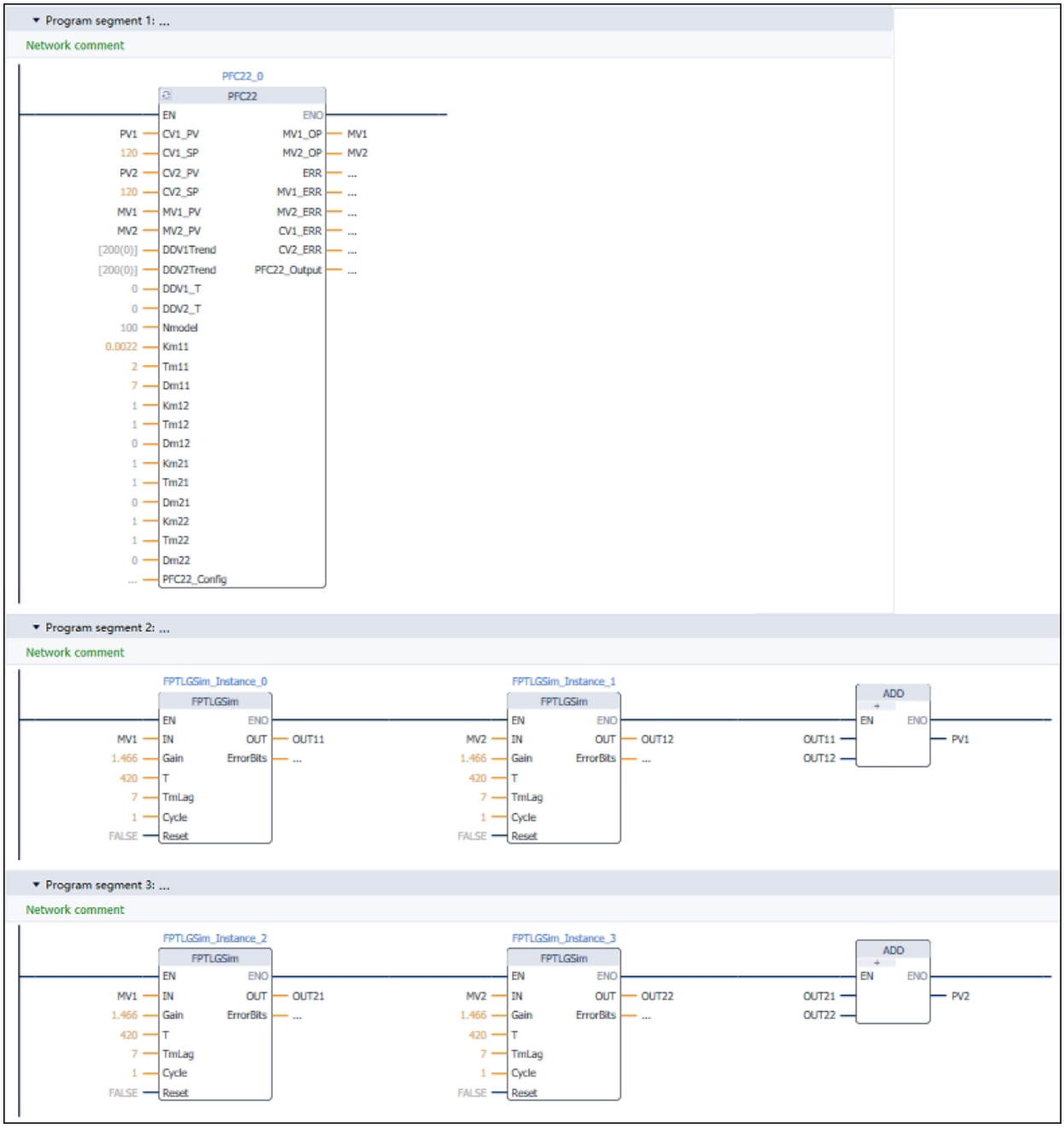
  - Control process

    The PFC22 instruction responds to the output (PV) of the FPTLGSim object, obtains the control quantity (OP) output, and applies it to the original control object, forming a control loop.

  - Control expectation

    If all parameters are configured properly, the controller will eventually keep the output value (CV_PV) of the controlled object near the setpoint CV_SP.

    LD language code example:

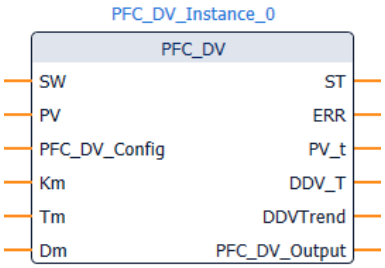


- **Precautions**
  - During operation, ensure that the switch is on, and the configuration is correct.
  - The execution time of the PFC22 instruction is 20 ms. With 50 PFC22 function blocks loaded, the total execution time is estimated to reach approximately 1 second. Loading 364 PFC22 function blocks will reach the controller's resource limit. During operation, configure the scheduling task cycle properly. For example, if 10 PFC22 are required to perform loop control simultaneously, with a block calculation time of 200 ms. The task cycle must be at least twice this value, that is, 400 ms, to ensure the CPU load does not become too high.

18.3.2.14 PFC_DV

The disturbance instruction outputs a predicted disturbance sequence and must be used in conjunction with a corresponding control instruction.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
PFC_DV	Feedforward predictive control	FB		PFC_DV_Instance_0(SW := , PV := , PFC_DV_Config := , Km := , Tm := , Dm := , ST => , ERR => , PV_t => , DDV_T => , DDVTrend => , PFC_DV_Output =>);

- **Related variables**

  Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
SW	Disturbance switch	DINT	0, 1	1	The disturbance switch disables the control when the switch is off. When used, it must be 1. Any invalid value is set to 0.
PV	Input value	REAL	Follows data type	0	Disturbance input value provided externally. Based on this input value, a corresponding disturbance sequence is generated.
PFC_DV_Config	Input parameter structure	PFC_DV_Config	Follows data type	-	Input parameter configuration structure.
Km	Model gain	REAL	Follows data type	1	An algorithm parameter that affects the control performance. Increasing the value speeds up the system response to some extent, while decreasing it slows down the system response to some extent. It is used to predict the disturbance sequence.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Tm	Model time constant	REAL	(0, 10000]	1	An algorithm parameter that affects the control performance. The smaller the Tm value, the fewer cycles required to reach a steady state. It is used to predict the disturbance sequence. Any invalid value is set to 1e-6.
Dm	Model hysteresis	REAL	[0, 10000]	0	An algorithm parameter that affects the control performance. The delay expressed in control cycles is Dm/Ts. It is used to predict the disturbance sequence. Any invalid value is set to 0.

PFC_DV_Config structure configuration

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Ts	Sampling cycle	REAL	(0, 10000]	1	This value determines the control cycle. This value must be greater than 0, and can not be set too low. An excessively small value may cause a calculation timeout and degrade control performance. If Ts is less than 0, the algorithm will automatically terminate, the switch will turn off, and the output will be held.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Model	Model type	DINT	0, 1, 2	1	Model type: 0: Non-integral. 1: Pseudo-integral. 2: Pure integral. Currently, types 0 (Non-integral) and 1 (Pseudo-integral) are primarily used. Type 2 (Pure integral) is not implemented. In practice, integral models are typically handled as pseudo-integral. If the model selection does not follow 0, 1, 2, it is set to 0 by default.
Low	Lower limit of the disturbance variable	REAL	Follows data type	-100	Lower limit; if the value falls below the lower limit, it is clamped to the lower limit.
High	Upper limit of the disturbance variable	REAL	Follows data type	100	Upper limit; if the value exceeds the upper limit, it is clamped to the upper limit.
DMax	Increment limit	REAL	Follows data type	100	Increment limit; when exceeded, it will be clamped to the maximum increment value. Any invalid value is set to 0.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ST	Switch state	DINT	0, 1	-	Switch status. Initially the value equals the switch input value. When an internal error occurs, this value is set to 1.
ERR	Fault code	DINT	Follows data type	-	For fault codes, see PFC_DV fault codes.
PV_t	Feedforward correction value	REAL	Follows data type	-	Feedforward value after limit.
DDV_T	Disturbance increment	REAL	Follows data type	-	Disturbance increment value for disturbance prediction and feedback correction.

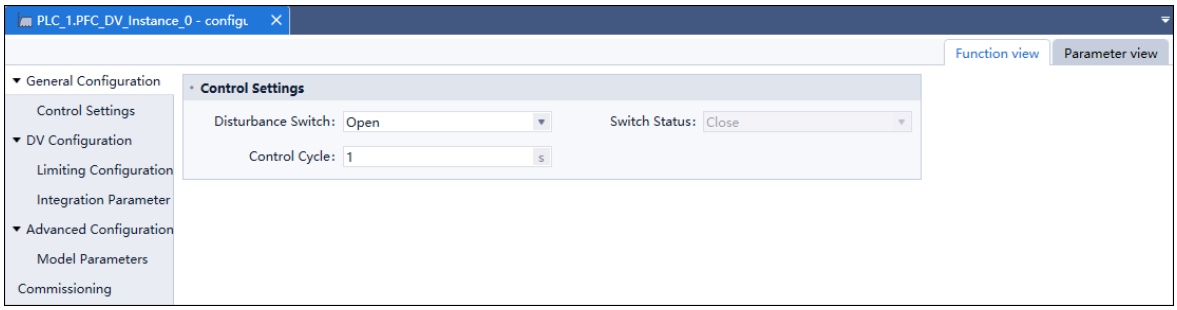
Output Variable	Name	Data Type	Valid Range	Default Value	Description
DDVTrend	Disturbance sequence	ARRAY [0..199] OF REAL	Follows data type	-	Disturbance sequence. After generating this sequence, it must be output to the input interface with the same name on the control instruction for feedforward correction.
PFC_DV_Output	Output parameter structure	PFC_DV_Output	Follows data type	-	Output parameter structure.

PFC_DV_Output structure configuration

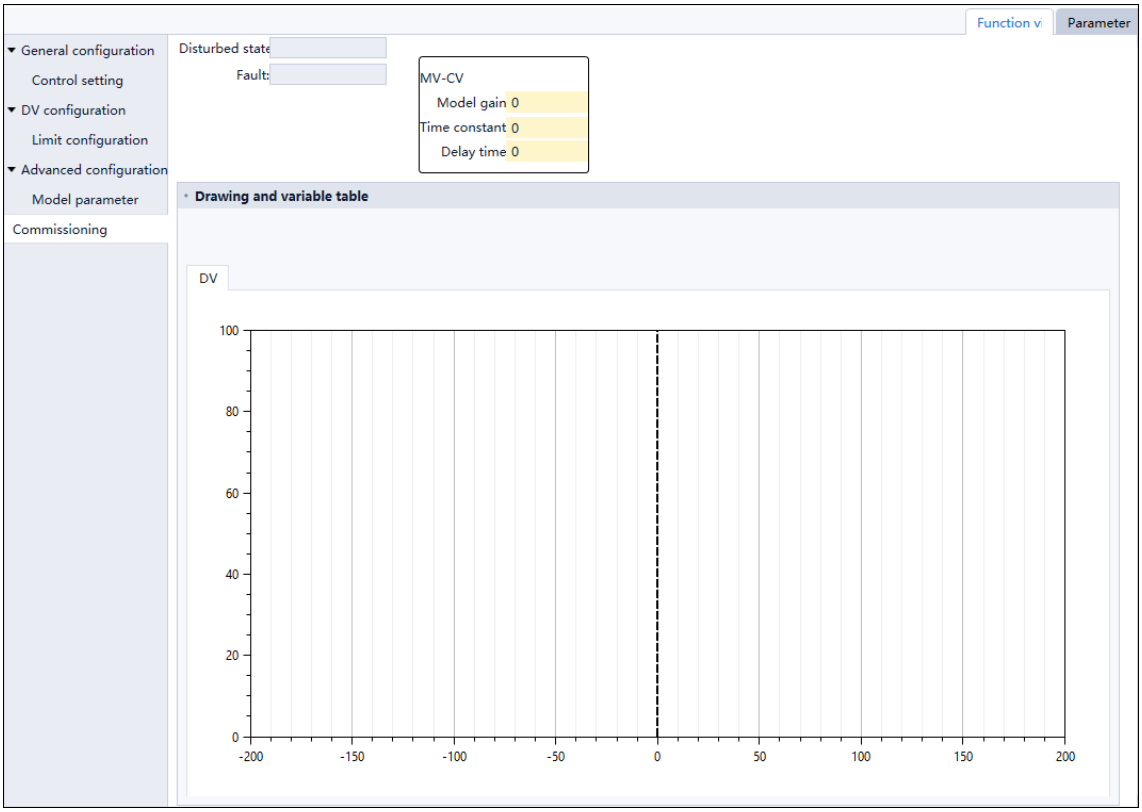
Output Variable	Name	Data Type	Valid Range	Default Value	Description
PV_Past	Measured value history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records historical values for the disturbance input.
PVt_Past	Feedforward value history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Records historical values of the feedforward signal.
DV_Low_Past	Disturbance value lower limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical lower limits for the disturbance value, for plotting.
DV_High_Past	Disturbance value upper limit history sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical upper limits for the disturbance value, for plotting.
DV_Low_Future	Disturbance value lower limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future lower limits for the disturbance value, for plotting.
DV_High_Future	Disturbance value upper limit future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future upper limits for the disturbance value, for plotting.
DV_PV_Future	Disturbance value future sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of future disturbance values, for plotting.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
DDV_Past	Feedforward value history increment sequence	ARRAY [0..199] OF REAL	Follows data type	-	Sequence of historical increments for the feedforward value, for plotting.
Version	Version information	ARRAY [0..3] OF DINT	Follows data type	-	Records the version information.

- **Function**
  - PFC_DV is a disturbance feedforward prediction module. Its primary function is to predict the future disturbance sequence of the controlled object based on the current disturbance input. The predicted disturbance sequence can be fed forward to the feedforward input channel of the PFC11/PFC22 instruction, enabling proactive compensation of disturbances and improving disturbance rejection.
  - The PFC_DV function block includes an intuitive interface for parameter configuration and commissioning. After adding the PFC_DV component, select "Control object > PFC" in the left project tree, and then double-click the PFC_DV instance under the PFC node to open the configuration and tuning interface, as shown in the figure below.



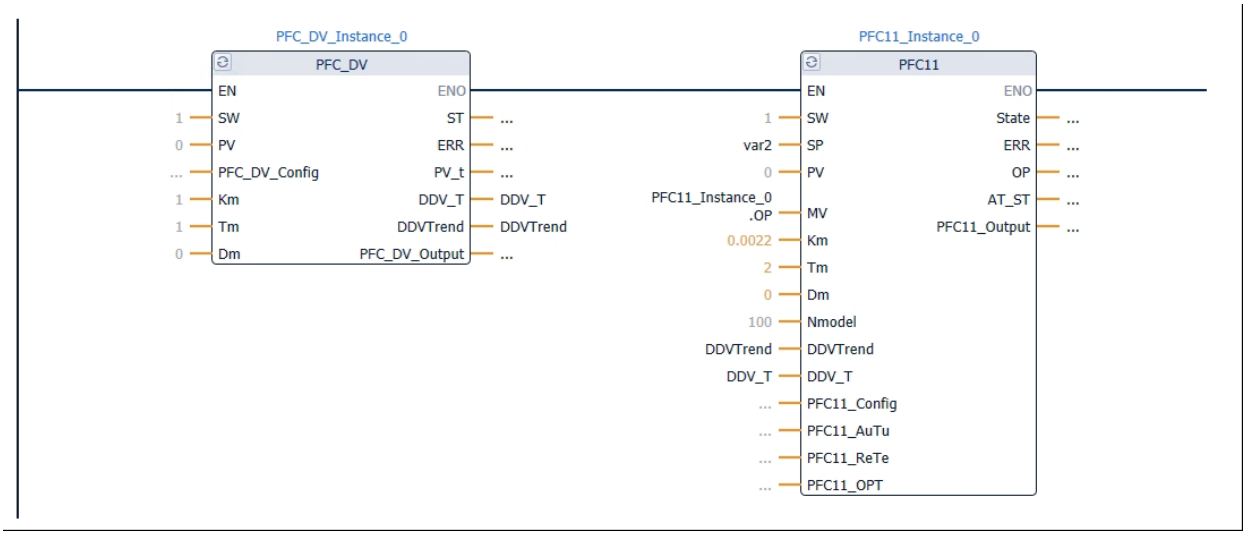
    - Parameter description
      The parameter configuration of the PFC_DV instruction includes general configuration, DV configuration, and model configuration. For detailed parameter descriptions, see Relevant variables.
    - Commissioning
      After the parameters are configured, the historical and future disturbance value sequence can be viewed directly in the commissioning interface.



- **Program example**

  During operation, this instruction primarily feeds the sequence output by DDVTrend into the control component PFC11/PFC22. The generated sequence is determined by the model parameters Km, Tm, and Dm, as well as the input disturbance PV.

  LD:



- **Precautions**
  - During operation, ensure that the switch is on, and the configuration is correct.
  - When setting the model parameters, ensure they are consistent with the controller parameter type, either all integral or non-integral.
  - The execution time of the PFC_DV instruction is 8 ms. With 140 PFC_DV function blocks loaded, the total execution time is estimated to reach approximately 1 second. Loading 1154 PFC_DV function blocks will reach the controller's resource limit. During operation, configure the scheduling task cycle properly. For example, if 10 PFC_DV are required to perform loop control simultaneously, with a block calculation time of 80 ms. The task cycle must be at least twice this value, that is, 160 ms, to ensure the CPU load does not become too high.

18.3.2.15 Fault Codes

PFCs fault codes

Fault code	Value	Description
APC_STATE_ERROR	-207	Tuning state error.
APC_NULL_POINT	-101	Null pointer.
APC_LENGTH_MISMATCH	-6	Length mismatch.
APC_OUT_OF_MEMORY	-5	Insufficient memory.
APC_FUNCTION_ERROR	-4	Component operation fault.
APC_PARAME_ID_NOT_FOUND	-2	ID not found.
APC_TS_OUT_OF_LIMIT	1	Ts is less than 0.
APC_SP_OUT_OF_LOW	100	The setpoint is below the lower limit.
APC_SP_OUT_OF_HIGH	101	The setpoint is above the upper limit.
APC_PARAM_OUT_OF_MAX	102	The parameter is above the upper limit of the range.
APC_PARAM_OUT_OF_MIN	103	The parameter is below the lower limit of the range.
APC_TM_OUT_OF_LIMIT	104	Tm is less than or equal to 0.
APC_TREF_OUT_OF_LIMIT	105	Tref is less than or equal to 0.
APC_DM_OUT_OF_LIMIT	106	Dm is less than 0.
APC_OP_LIMIT_ERROR	210	The OP output lower limit is greater than the upper limit.
APC_SP_LIMIT_ERROR	211	The SP lower limit is greater than the upper limit.
APC_SW_OUT_OF_LIMIT	212	SW is neither 0 nor 1.
APC_MODEL_OUT_OF_LIMIT	213	Model is not 0, 1, or 2.
APC_COF_OUT_OF_LIMIT	214	COF is neither 0 nor 1.
APC_COP_OUT_OF_LIMIT	215	COP is less than 0.
APC_I_COF_OUT_OF_LIMIT	216	I_COF is neither 0 nor 1.
APC_DOPMAX_OUT_OF_LIMIT	218	DOPMax is less than 0.
APC_SP_DMAX_OUT_OF_LIMIT	219	SP_DMax is less than 0.
APC_SP_DMIN_OUT_OF_LIMIT	220	SP_DMin is greater than 0.
APC_PARAM_DISABLE_CHANGE	1001	Parameter modification is prohibited.
APC_COMP_IS_NOT_RUNNING	1002	Tuning is not running.

PFCm/PFCm_v2/PFCm22/PFC22 fault code

Fault code	Value	Description
APC_OUT_NOTNULL_BEFORE_MEMORY	-7	The pointer is not null before memory allocation.
APC_OUT_OF_MEMORY	-5	Insufficient memory.
APC_TS_OUT_OF_LIMIT	1	Ts is less than 0.
APC_TM1_OUT_OF_LIMIT	101	Tm1 is less than or equal to 0.
APC_TM2_OUT_OF_LIMIT	102	Tm2 is less than or equal to 0.
APC_DM1_OUT_OF_LIMIT	103	Dm1 is less than 0.
APC_DM2_OUT_OF_LIMIT	104	Dm1 is less than 0.
APC_TREF_OUT_OF_LIMIT	105	Tref is less than or equal to 0.
APC_MODEL_OUT_OF_LIMIT	106	Model is not 0, 1 or 2.
APC_COF_OUT_OF_LIMIT	107	COF is neither 0 nor 1.
APC_I_COF_OUT_OF_LIMIT	108	I_COF is neither 0 nor 1.
APC_CONTROLLER_OPEN	110	The controller is in open-loop mode.
APC_MV_OUT_OF_CONTROL	201	MV variable has no target; no valid CV is available.
APC_MV_VAL_LIMIT_ERROR	202	The valid lower limit for MV exceeds the upper limit.
APC_MV_PV_OUT_OF_VAL_LOW	203	MV_PV quality is good, but the value below the valid lower limit.
APC_MV_PV_OUT_OF_VAL_HIGH	204	MV_PV quality is good, but the value exceeds the valid upper limit.
APC_MV_ST_OUT_OF_5	206	Five consecutive ON/ST anomalies trigger the alarm.
APC_MV_OPE_LIMIT_ERROR	210	The MV operation lower Limit exceeds the operation upper limit.
APC_MV_PV_OUT_OF_OPE_LOW	211	MV_PV quality is good, but the value below the operation lower limit.
APC_MV_PV_OUT_OF_OPE_HIGH	212	MV_PV quality is good, but the value exceeds the operation upper limit.
APC_MV_SW_OUT_OF_LIMIT	213	The MV switch signal is out of limit.
APC_MV_IRV_SW_OUT_OF_LIMIT	214	The MV_IRV switch signal is out of limit.
APC_MV_DMAX_OUT_OF_LIMIT	215	The MV maximum output limit is out of range.
APC_MV_OPE_OVER_VAL	215	The MV operation upper and lower limits exceed the valid upper and lower limits.
APC_MV_DMIN_OUT_OF_LIMIT	216	The MV minimum output limit is out of range.

Fault code	Value	Description
APC_MV_COP_OUT_OF_LIMIT	217	The MV_COP output is out of limit.
APC_CV_OUT_OF_CONTROL	401	The CV variable is out of control; no valid MV is available.
APC_CV_VAL_LIMIT_ERROR	402	The valid lower limit of CV exceeds the valid upper limit.
APC_SP_OUT_OF_VAL_LIMIT	403	The setpoint is outside the valid upper and lower limits.
APC_CV_OUT_OF_VAL_LOW	404	CV is below the valid lower limit.
APC_CV_OUT_OF_VAL_HIGH	405	CV exceeds the valid upper limit.
APC_CV_ST_OUT_OF_5	406	Five consecutive ON/ST anomalies trigger the alarm.
APC_CV_OPE_LIMIT_ERROR	410	CV operation upper and lower limit configuration error.
APC_SP_OUT_OF_OPE_LIMIT	411	The setpoint is outside the operation upper and lower limits.
APC_CV_SW_OUT_OF_LIMIT	413	The CV switch signal is out of limit.
APC_CV_SP_DMAX_OUT_OF_LIMIT	414	The CV maximum setpoint limit is out of range.
APC_CV_SP_DMIN_OUT_OF_LIMIT	415	The CV minimum setpoint limit is out of range.
APC_CV_DELTACV_OUT_OF_LIMIT	416	The CV increment limit is out of range.
APC_CV_OUT_OF_OPE_LOW	1415	CV is below the operation lower limit.
APC_CV_OUT_OF_OPE_HIGH	1416	CV exceeds the operational upper limit.

PFCs_dv/PFCs_dv_v2/PFCm_dv/PFC_DV fault code

Fault code	Value	Description
APC_TS_OUT_OF_LIMIT	1	Ts is less than 0.
APC_PV_LOW_GREATER_THAN_HIGH	2	The PV lower limit is greater than the upper limit.
APC_PV_OUT_OF_HIGH	100	PV exceeds the upper limit.
APC_PV_OUT_OF_LOW	101	PV is below the lower limit.
APC_PV_OUT_OF_DMAX	102	The disturbance value exceeds the increment upper limit.
APC_TM_OUT_OF_LIMIT	103	TM variable is out of limit.
APC_SW_OUT_OF_LIMIT	104	SW variable is out of limit.
APC_MODEL_OUT_OF_LIMIT	105	Model is not 0, 1 or 2.
APC_DMAX_OUT_OF_LIMIT	106	The maximum output limit is out of range.

PFCs_v2/PFC11 fault code

Fault code	Value	Description
APC_STATE_ERROR	-207	Tuning state error.
APC_NULL_POINT	-101	Null pointer.
APC_LENGTH_MISMATCH	-6	Length mismatch.
APC_OUT_OF_MEMORY	-5	Insufficient memory.
APC_FUNCTION_ERROR	-4	Component operation fault.
APC_PARAME_ID_NOT_FOUND	-2	ID not found.
APC_TS_OUT_OF_LIMIT	1	Ts is less than 0.
APC_SP_OUT_OF_LOW	100	The setpoint is below the lower limit.
APC_SP_OUT_OF_HIGH	101	The setpoint is above the upper limit.
APC_PARAM_OUT_OF_MAX	102	The parameter is above the upper limit of the range.
APC_PARAM_OUT_OF_MIN	103	The parameter is below the lower limit of the range.
APC_TM_OUT_OF_LIMIT	104	Tm is less than or equal to 0.
APC_TREF_OUT_OF_LIMIT	105	Tref is less than or equal to 0.
APC_DM_OUT_OF_LIMIT	106	Dm is less than 0.
APC_OP_LIMIT_ERROR	210	The OP output lower limit is greater than the upper limit.
APC_SP_LIMIT_ERROR	211	The SP lower limit is greater than the upper limit.
APC_SW_OUT_OF_LIMIT	212	SW is neither 0 nor 1.
APC_MODEL_OUT_OF_LIMIT	213	Model is not 0, 1 or 2.
APC_COF_OUT_OF_LIMIT	214	COF is neither 0 nor 1.
APC_COP_OUT_OF_LIMIT	215	COP is less than 0.
APC_I_COF_OUT_OF_LIMIT	216	I_COF is neither 0 nor 1.
APC_DOPMAX_OUT_OF_LIMIT	218	DOPMax is less than 0.
APC_SP_DMAX_OUT_OF_LIMIT	219	SP_DMax is less than 0.
APC_SP_DMIN_OUT_OF_LIMIT	220	SP_DMin is greater than 0.
APC_PV_OUT_OF_LOW	221	The PV value is below the alarm lower limit RT_Warn_Low.
APC_PV_OUT_OF_HIGH	222	The PV value exceeds the alarm upper limit RT_Warn_High.
APC_PARAM_DISABLE_CHANGE	1001	Parameter modification is prohibited.
APC_COMP_IS_NOT_RUNNING	1002	Tuning is not running.

PIDR fault codes

Fault code	Value	Description
APC_STATE_ERROR	-207	Tuning state error.
APC_NULL_POINT	-101	Null pointer.
APC_LENGTH_MISMATCH	-6	Length mismatch.
APC_OUT_OF_MEMORY	-5	Insufficient memory.
APC_FUNCTION_ERROR	-4	Component operation fault.
APC_PARAME_ID_NOT_FOUND	-2	ID not found.
APC_TS_OUT_OF_LIMIT	1	Ts is less than 0.
APC_SP_OUT_OF_LOW	100	The setpoint is below the lower limit.
APC_SP_OUT_OF_HIGH	101	The setpoint is above the upper limit.
APC_PARAM_OUT_OF_MAX	102	The parameter is above the upper limit of the range.
APC_PARAM_OUT_OF_MIN	103	The parameter is below the lower limit of the range.
APC_TM_OUT_OF_LIMIT	104	Tm is less than or equal to 0.
APC_TREF_OUT_OF_LIMIT	105	Tref is less than or equal to 0.
APC_DM_OUT_OF_LIMIT	106	Dm is less than 0.
APC_OP_LIMIT_ERROR	210	The OP output lower limit is greater than the upper limit.
APC_SP_LIMIT_ERROR	211	The SP lower limit is greater than the upper limit.
APC_SW_OUT_OF_LIMIT	212	SW is neither 0 nor 1.
APC_MODEL_OUT_OF_LIMIT	213	Model is not 0, 1 or 2.
APC_COF_OUT_OF_LIMIT	214	COF is neither 0 nor 1.
APC_COP_OUT_OF_LIMIT	215	COP is less than 0.
APC_I_COF_OUT_OF_LIMIT	216	I_COF is neither 0 nor 1.
APC_DOPMAX_OUT_OF_LIMIT	218	DOPMax is less than 0.
APC_SP_DMIN_OUT_OF_LIMIT	221	SP_DMin is greater than 0.
APC_DV_LIMIT_ERROR	223	Upper/lower limit error for measured disturbance values.
APC_DV_OUT_OF_LOW	224	The measured disturbance value is below the lower limit.
APC_DV_OUT_OF_HIGH	225	The measured disturbance value exceeds the upper limit.
APC_PARAM_DISABLE_CHANGE	1001	Parameter modification is prohibited.
APC_COMP_IS_NOT_RUNNING	1002	Tuning is not running.

18.3.3 Motion Control

18.3.3.1 Overview

18.3.3.1.1 PLCopen Specifications

IEC 61131 is an international standard for general-purpose programmable logic controllers (PLCs). It was initially proposed as an industry standard by several leading European PLC companies. IEC 61131-3 is a section of the IEC 61131 standard for PLC programming, defining six programming language standards.

PLCopen, headquartered in Europe, is a global organization promoting the IEC 61131-3. Its members, several well-known PLC manufacturers, have refined technical details based on IEC 61131-3 to harmonize the programming across PLCs from different manufacturers, reducing technical differences and barriers. This allows users to switch between different PLC brands without needing to learn new programming methods.

In China, the corresponding national recommended standard GB/T 15969.3 was released in 1995 and upgraded in 2005. It serves as the recommended design standard for PLC manufacturers. PLCopen also has members in China. In addition to recommendations for common logic control instructions, program structures, and keywords in various languages, the PLCopen specifications contain the technical specifications for motion control (MC) function blocks, including the naming, functions, in-out variable definitions, and related temporal logic of MC function blocks, maximizing the interoperability of users' programming techniques.

The MC function blocks in iFA motion control are based on the PLCopen specifications for MC function blocks. The following table presents the differences between the MC function blocks in iFA motion control and in PLCopen specifications.

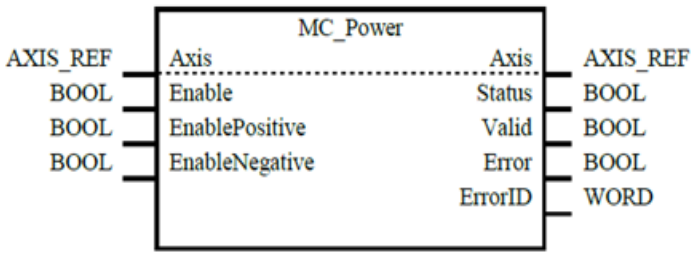
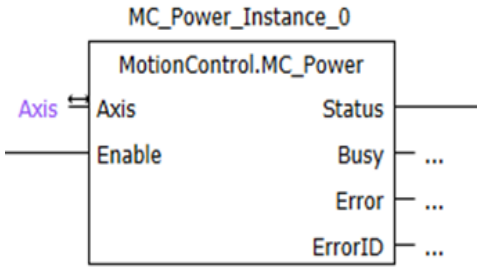
- **Differences in structure data**

Data Type		Description
PLCopen	iFA	
AXIS_REF	AXIS_REF	Single axis structure.
AXES_GROUP_REF	AXES_GROUP_REF	Axis group structure.
MC_TRIGGER_REF	Unused	Touch probe structure. The structure is split in the iFA to enhance usability.
MC_CAM_ID	Unused	Cam table ID structure. The MC_CAM_TABLE_REF structure is used in the iFA to simplify the cam function block calling in PLCopen specifications.

Data Type		Description
PLCopen	iFA	
MC_INPUT_REF	Unused	Input reference structure. The input parameter is axis instance in iFA.
MC_OUTPUT_REF	Unused	Output reference structure. The output parameter is axis instance in iFA.

- **Differences in FB forms**

  In PLCopen specifications, the input and output variables of FBs are represented by dashed lines; in iFA, the input and output variables of FBs are represented by double arrows. In PLCopen specifications, no instance names are shown in FBs, but other entity names, input variables, and output variables have the same forms.

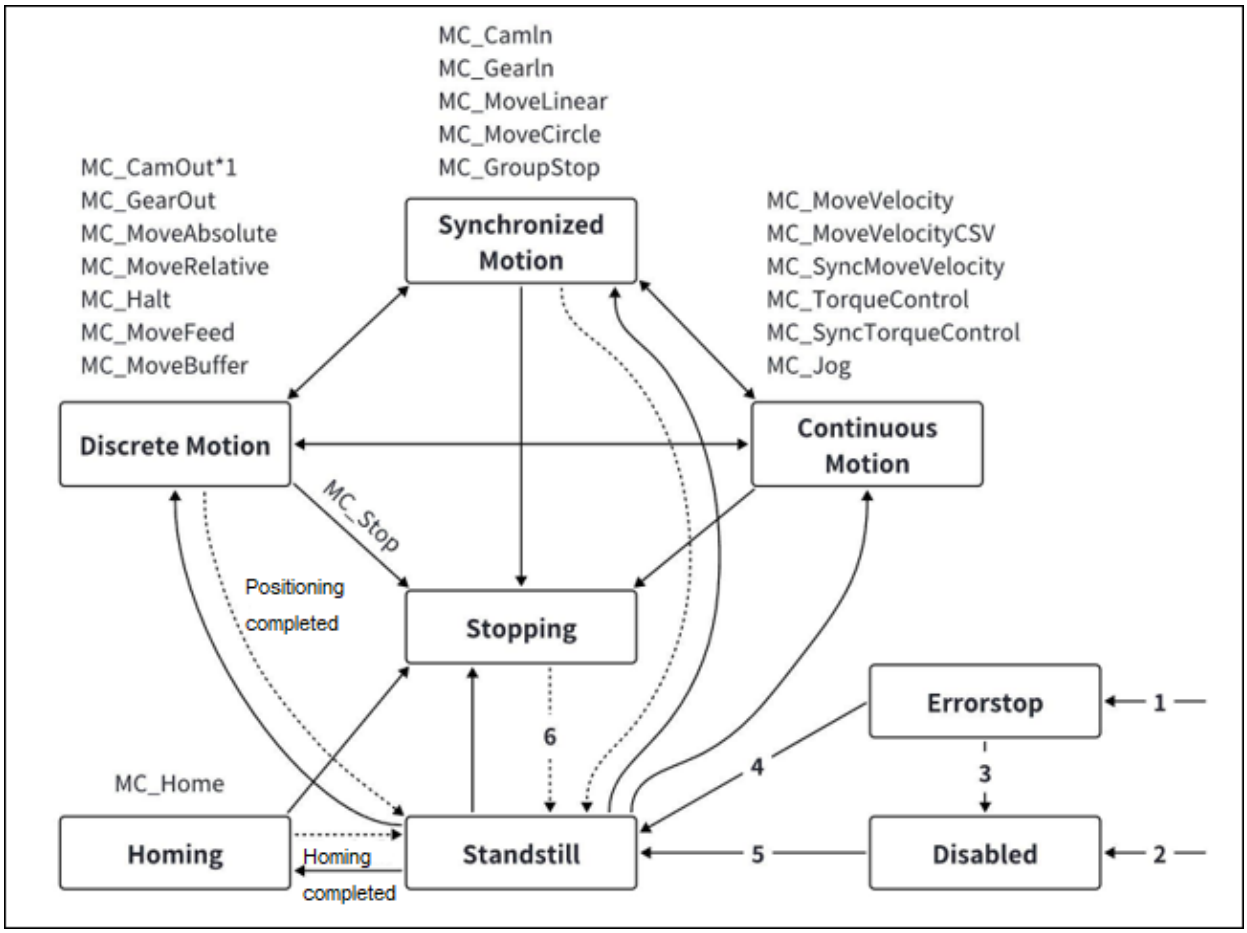
PLCopen FB	iFA FB
	

- **Differences in general functions**

Function	PLCopen	iFA
Default behaviors of motion parameters	Instruction block fault reporting, compilation warning, specifying and selecting limits for motion parameters in the axis structure.	The instruction block automatically corrects improper input parameters. See specific instructions for correction details.
Positive and negative limits of motion parameters	Acceleration, deceleration, and jerk must be positive. Velocity, position, and distance can be either positive or negative.	Acceleration, deceleration, jerk, and velocity must be positive. The instruction block will automatically correct any abnormal values. Position and distance can be either positive or negative.
FB naming rules	MC_FBname_SupplierID	MC_FBname_InstanceID

- **Single axis state machine**

The axis state and motion are managed based on the state machines in PLCopen specifications.



The state machines are detailed in the following table:

Table18-3-3-1-1-1 State definitions

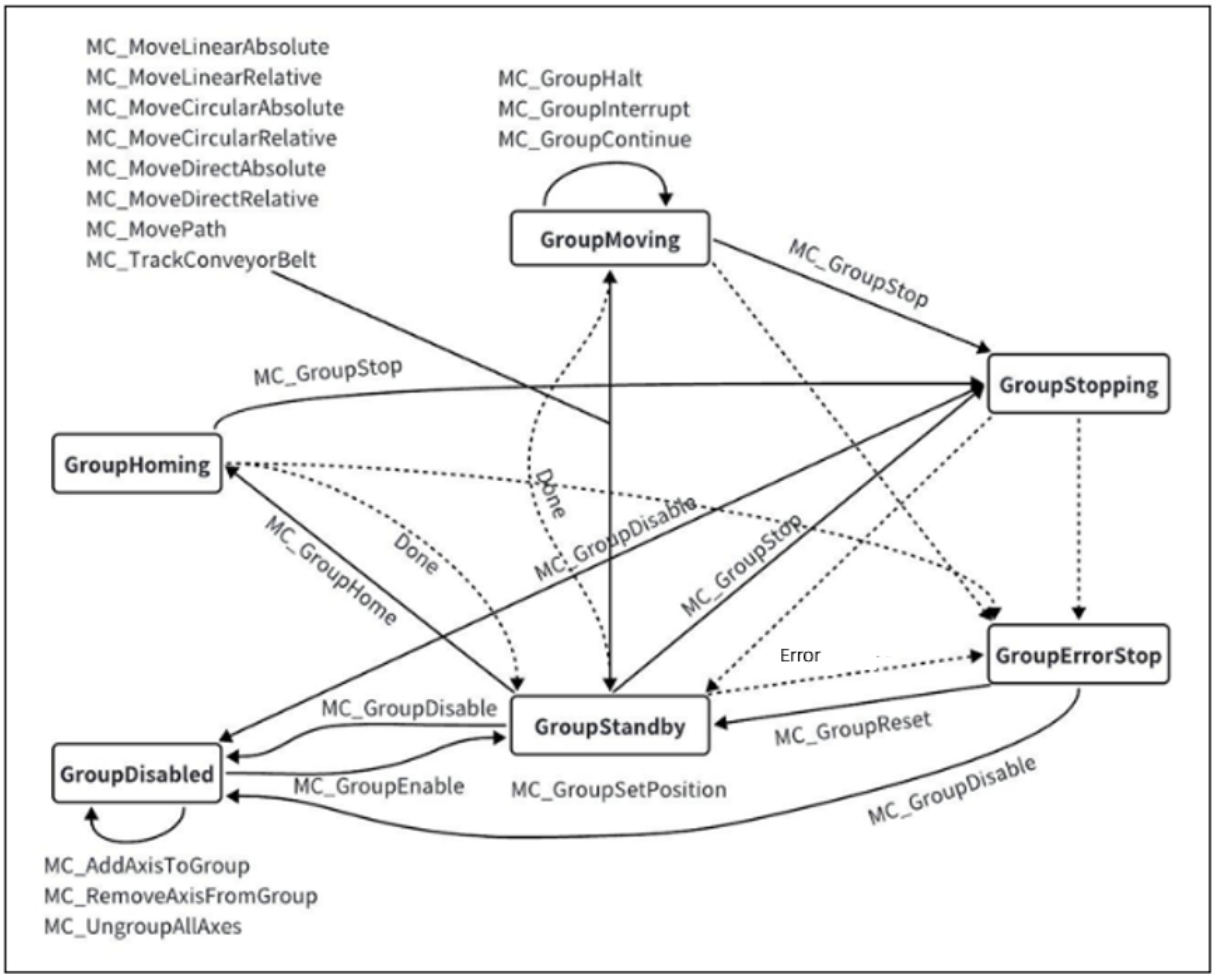
State	Description
Disabled	Disabled.
ErrorStop	Stop upon fault.
Standstill	Standstill.
Homing	Homing.
Stopping	Decelerating to stop.
Discrete Motion	Discrete motion.
Continuous Motion	Continuous motion.
Synchronized Motion	Synchronized motion.

Table18-3-3-1-1-2 State switching conditions

Switched to	Conditions
1	Immediately after a fault is detected by the fault detection logic of an axis.

Switched to	Conditions
2	When no fault occurs and the MC_Power.Enable is set to FALSE.
3	When the MC_Reset is called to reset an axis fault and the MC_Power.Status is set to FALSE.
4	When the MC_Reset is called to reset an axis fault and the MC_Power.Status is set to TRUE.
5	When the MC_Power.Enable is set to TRUE and the MC_Power.Status is set to TRUE.
6	When the MC_Stop.Done is set to TRUE and the MC_Stop.Execute is set to FALSE.

- **Axis group state machine**



The axis group state machines are used to describe the states of an axis group, including: GroupDisabled, GroupHoming, GroupMoving, GroupStopping, GroupErrorStop, and GroupStandby.

- **Precautions**
  - To switch the axis group state from GroupDisabled to GroupStandby, be sure that all axes in the axis group are connected and that there are no fault messages for any axes in the axis group.
  - If an axis in the axis group has a fault message while the axis group is in the GroupStandby state, the axis group state will be switched to GroupErrorStop.

- To switch the axis group state from GroupErrorStop to GroupStandby, call the MC_GroupReset to clear all errors of the axis group and the axes within the axis group. If any faults cannot be cleared, the axis group state will not be switched successfully.
- To switch the axis group state from GroupStandby to GroupMoving or GroupHoming, be sure that each axis in the axis group is in StandStill state. Otherwise, calling an axis group motion control instruction will result in a fault, but this will not affect the execution of the axis group state machine.
- To switch the axis group state from GroupStopping to GroupStandby, be sure that the output variable Done of the MC_GroupStop is set and the input variable Execute is reset.
- Do not execute a single axis instruction on a configured axis in the axis group.
- The MC_GroupSetPosition can be called only when the axis group is in the GroupStandby state.
- The MC_GroupDisable can switch the axis group state from GroupErrorStop to GroupDisabled.
- The GroupErrorStop state does not affect the states of individual axes within the axis group.

18.3.3.1.2 Single-Axis Instructions

- **Buffer**

  1. For buffered instructions with abnormal motion parameters, they will be executed normally if the abnormal parameters can be automatically corrected; otherwise, the instructions will immediately report a fault and abort the executing instructions.
  2. The MC_Jog instruction does not have a buffer, but it can be followed by buffered instructions. Since the MC_Jog instruction is always being executed without an end flag, the buffered instructions will never be executed. If the MC_Jog instruction stops executing, the buffered instructions can be executed. When the MC_Home instruction is being executed, no instructions can be buffered.
  3. An unsupported buffer mode will cause a fault.
  4. The motion is planned starting from the intermediate velocity. How to plan the motion in case of insufficient displacement depends on whether the axis is allowed to move in the reverse direction. If allowed, the axis will move in the reverse direction before reaching the target position. If not allowed, the jerk or acceleration/deceleration will be automatically corrected.
  5. When a buffered instruction is in Busy and not in Active state, its input parameters can be modified.
  6. Currently, only one instruction supports the buffer function.

- **Abort**

  1. If an instruction is aborted and the CommandAborted changes to TRUE, the instruction performing the abort will be executed with automatically corrected motion parameters.
  2. When the option No ErrorStop upon hitting the limit and triggering abnormal parameters is checked on the IDE interface, the axis state will not switch to ErrorStop. When this option is not checked, the axis state will switch to ErrorStop if the current velocity is not 0, and return to StandStill if the current velocity is 0.
  3. When the current motion control instruction and buffered instructions are aborted, the buffer will be cleared; the CommandAborted will change to TRUE for the current motion control instruction and buffered instructions; the instruction performing the abort will be executed normally.
  4. If the current motion state is not the state for executing an instruction, a fault will occur, without affecting the axis or the current motion control instruction. The priority of fault identification for

abnormal states is higher than that for abnormal parameters, with Stop and ImmediateStop having the highest priority.

5. The MC_Home instruction can be aborted by the MC_Stop and MC_ImmediateStop instructions only during execution.

- **Position limit**

1. Identification of hitting the limit/Handling rules for hitting the negative limit while moving in the positive direction: Limit check is performed only against the positive limit when the axis moves in the positive direction, and a fault occurs when the axis moves beyond the positive limit. Limit check is performed only against the negative limit when the axis moves in the negative direction, and a fault occurs when the axis moves beyond the negative limit.
2. Change rules for status bit, axis state, and instruction output flag upon hitting the limit: When the option No ErrorStop upon hitting the limit and triggering abnormal parameters is checked on the IDE interface, the axis state does not switch to ErrorStop upon hitting the limit, but a fault occurs. When this option is not checked, the axis state directly switches to ErrorStop with a fault message.
3. Handling rules for hitting the limit: The instruction identifies whether the axis will move beyond the limit during the execution. If the axis trajectory under the current instruction may exceed the limit, software limit-induced deceleration will be performed, and the axis will ultimately stop at the limit boundary with the deceleration parameters set in the software. Stopping at the limit boundary will cause the instruction to issue a fault message.
4. Stopping rules for hitting the software limit: If the software-restricted deceleration is less than the maximum deceleration for software limit, the axis will stop with the maximum deceleration. If the distance required to stop at the maximum deceleration is greater than the predicted maximum stopping distance based on the software limit, the axis will stop based on the maximum distance. The instruction identifies the current velocity and distance from the position limit and then selects one deceleration parameter that makes the axis to move the shortest deceleration distance from the maximum software-restricted deceleration, the deceleration for software limit, and the maximum deceleration distance for software limit. During the software limit-induced deceleration, the trajectory is the T-curve, which means that after the software limit-inducted deceleration is triggered, the acceleration immediately changes to the deceleration calculated based on the deceleration for software limit, the maximum software-restricted deceleration, and the maximum deceleration distance for software limit.
5. If the initial position is beyond the limit or on the limit and moving towards the limit direction, the axis will move normally. If the initial position is within the limit and moving outwards, the changes in axis processing logic and function block output flag are as follows: When the option No ErrorStop upon hitting the limit and triggering abnormal parameters is checked on the IDE interface, the axis state does not switch to ErrorStop upon hitting the limit, but a fault occurs. When this option is not checked, the axis state directly switches to ErrorStop with a fault message.

- **FB fault message display**

1. The name and path of the instruction causing a fault as well as the fault time stamp and ErrorID are recorded in the axis parameter fbeFBError. Up to 15 faults can be recorded in the fbeFBError.
2. When the axis structure parameter bDisableErrorLogging is set to TRUE, no fault message is recorded in the fbeFBError.
3. After the FB is reset, the fault state and fault code are cleared.

4. The MC_Reset instruction can clear the faults recorded by the fbeFBError.

- **Motion parameters**
  1. When a parameter value is greater than the maximum limit, it will be treated as the limit value.
  2. The decimal precision of floating-point motion parameters is within 10^{-6}.
  3. With an S-curve trajectory planning, if the parameter Jerk value is 0, it will be automatically corrected to 1e-5 by the instruction block. With a T-curve trajectory planning, the parameter Jerk will not be evaluated.
- **Axis pointer being null**

  Null pointer check is performed in all function blocks that receive axis variables, involving single axis instructions, synchronous axis instructions, and axis group instructions. If the axis pointer passed to the function block is null, a fault will occur after the instruction is executed, which will not cause project errors or a controller crash.

  During the execution of an instruction, it is prohibited to switch the axis object through REF=.
- **Instructions triggered within one cycle**

  The output Done of an instruction is used as the input for Execute in the instruction trigger logic. The supported instructions include MC_MoveAbsolute, MC_MoveRelative, MC_MoveSuperImposed, and MC_MoveFeed.
- **Servo terminal parameters**
  1. The specific number of supported touch probe channels depends on the servo drive. SV660N supports two probe channels, while SV670/SV680 supports one probe channel only.

Table18-3-3-1-2-1 SV660N DI/DO terminals

Signal		Default Function	Pin Number	Description
General	DI1	P-OT	10	Positive overtravel switch.
	DI2	N-OT	9	Negative overtravel switch.
	DI3	HomeSwitch	8	Home switch.
	DI4	TouchProbe2	7	Touch probe 2.
	DI5	TouchProbe1	11	Touch probe 1.
	+24V		15	Internal 24 V power supply; voltage range: 20 to 28 V; maximum output current: 200 mA.
	COM-		14	
	COM+		13	Common terminal of DI terminals.
	DO1+	S-RDY+	1	Servo ready.
	DO1-	S-RDY-	6	
	DO2+	ALM+	3	Alarm.
	DO2-	ALM-	2	
	DO1+	BK+	5	Brake.

Signal		Default Function	Pin Number	Description
	DO1-	BK-	4	

Table18-3-3-1-2-2 SV670N DI/DO terminals

Signal		Default Function	Pin Number	Description
General	DI1	P-OT	9	Positive overtravel switch.
	DI2	N-OT	7	Negative overtravel switch.
	DI3	HomeSwitch	8	Home switch.
	DI4	Emergence Stop	6	Emergency stop.
	DI5	TouchProbe1	10	Touch probe 1.
	COM+		5	Common terminal of DI terminals.
	+24V		3	Internal 24 V power supply; voltage range: 20 to 30 V; maximum output current: 200 mA.
	COM-		4	
	DO1+	S-RDY+	11	Servo ready.
	DO1-	S-RDY-	12	
	DO2+	BK+	13	Brake.
	DO2-	BK-	14	
	PZ-OUT	Phase-Z frequency-division output signal	1	Home pulse open-collector output signal.
	GND	Home pulse open-collector output signal ground	2	
	PE	Enclosure	15	-
	PE	Enclosure	16	-

2. Due to the differences in DI terminals of the servo, if you need to use the DI4 terminal of SV670 or SV680 as an emergency stop signal, you must obtain the terminal state from the 60FD object and add proper processing in the user program. The axis does not respond to the emergency stop signal from DI4.

3. PLC homing terminal binding: HomeSwitch, PositiveLimitSwitch, and NegativeLimitSwitch are bound to bit2, bit1, bit0 of the 60FD object by default. LatchInput must be manually mapped in the 60FD object.

Index 60FDh	Name	Digital Input			Setting Condition & Effective Time	-	Data Structure	VAR	Data Type	Int32
	Access	RO	Mapping	TPDO	Related Mode	PP HM CSP	Data Range	-	Factory Setting	0

Reflect the current DI terminal logic of the servo drive:

0: Logic invalid

1: Logic valid

Each bit represents the following DI signals:

bit	Signal
0	1: Negative overtravel active
1	1: Positive overtravel active
2	1: Home signal active
3-15	N/A
16	1: DI1 input active
17	1: DI2 input active
18	1: DI3 input active
19	1: DI4 input active
20	1: DI5 input active
21-26	N/A
27	1: STO1 signal input
28	1: STO2 signal input
29	1: EDM output active
30-31	N/A

- **Local pulse axis**

  Even if you select a single terminal such as Y0 or Y1, you can also use motion control instructions. In this case, the output signal is unclear. Therefore, it is not recommended to do so. When you select a single terminal, only the MC_SyncMoveVelocityCSV instruction can be executed.

18.3.3.1.3 Synchronous Axis Instructions

- **General specifications for cam**

  1. A fault will occur if the salve axis velocity exceeds the limit during synchronized motion.
  2. The start point (0,0) of the cam table cannot be modified, but the velocity ratio can be modified.
  3. A fault will occur if the master axis and the slave axis are specified as the same axis.
  4. For abnormal parameters, see the Parameter Correction section of respective instructions.
  5. Dense nodes are connected using the straight line.

6. The phase and displacement of the cam table are specified as relative quantities from the start point (0,0).

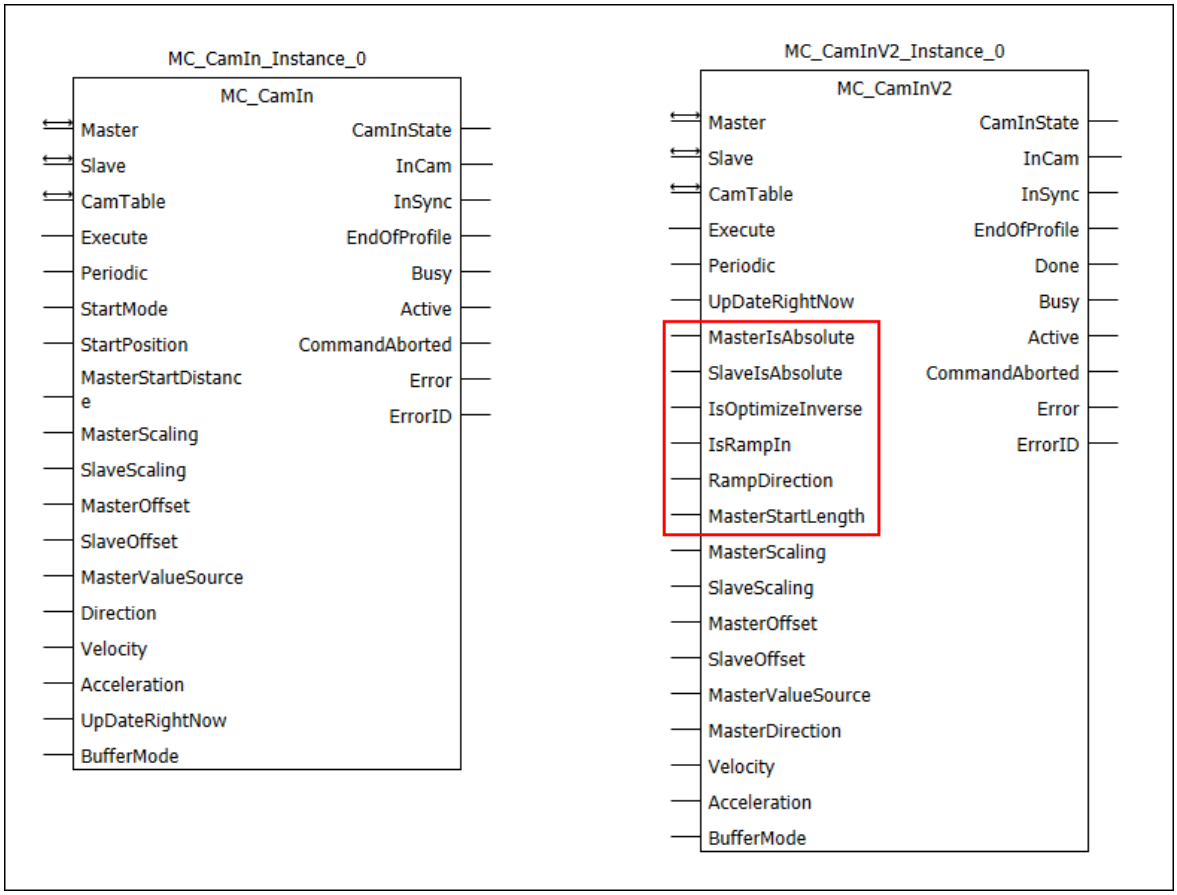
- **Operating specifications for cam table**

1. The velocity ratio of linear segments in the cam table cannot be modified.
2. The velocity ratios at the start and end points can be different.
3. The instruction MC_GenerateCamTable pre-checks the modified cam table parameters, and issues a fault message if any invalid parameter is identified. User-defined cam variables take effect only after passing the MC_CheckCamTable check. Precaution: Manually modifying checked cam data online may cause the slave axis to stall. It is recommended to recheck the data after modification.
4. When the node type in the cam table is dense node, the number of nodes cannot exceed 65535. When the node type is key node, the number of nodes cannot exceed 20000, and the minimum pitch between adjacent nodes is 0.0001.

- **Function differences between MC_CamInV2 and MC_CamIn**

1. Interface parameters

   Their interface parameters are different. For details, see the following function blocks.



2. Following modes of master and slave axes

   For MC_CamInV2, the scenario where the master axis in the cam table is in absolute mode and the slave axis is in absolute mode has been added, expanding the number of following modes from 1 to 4.

Following Mode	MC_CamInV2	MC_CamIn
Master axis in absolute mode and slave axis in absolute mode	√	×
Master axis in absolute mode and slave axis in relative mode	√	×

Following Mode	MC_CamInV2	MC_CamIn
Master axis in relative mode and slave axis in absolute mode	√	×
Master axis in relative mode and slave axis in relative mode	√	√

3. Coupling function

	MC_CamInV2	MC_CamIn
Enable selection	Selected through the parameter IsRampIn.	Cannot be selected; enabled by default.
Coupling method	Select positive or negative coupling through the parameter RampDirection.	×
Slave axis inverse optimization during coupling	Select whether to perform optimization or complete compensation as soon as possible through the parameter IsOptimizeInverse.	Cannot be selected; enabled by default.
Entry velocity	IsRampIn = FALSE: The slave axis velocity jumps (equal to the slave axis displacement increment in the current cycle/ task cycle); IsRampIn = TRUE: The slave axis moves without velocity jump and is smoothly coupled.	Controlled by acceleration input. Acceleration = 0: The slave axis velocity jumps directly (equal to cam velocity + coupling velocity); Acceleration ≠ 0: The slave axis moves without velocity jump and is smoothly coupled.
Entry direction	When the slave axis is a rotary axis, the slave axis increment can be switched between positive and negative based on the parameter RampDirection.	During entry, the slave axis displacement increment is calculated and compensated based on the master axis position. No direction parameter is available.

4. Master axis offset correction

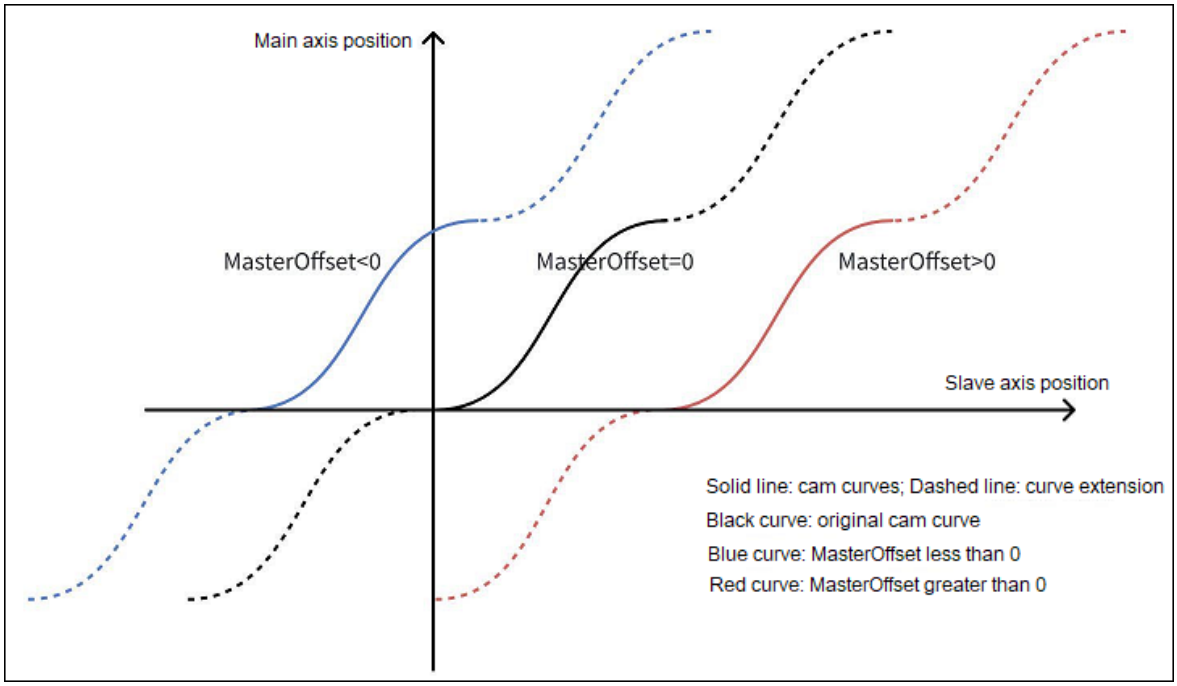


Figure 18-3-3-1-3-1 MC_CamInV2

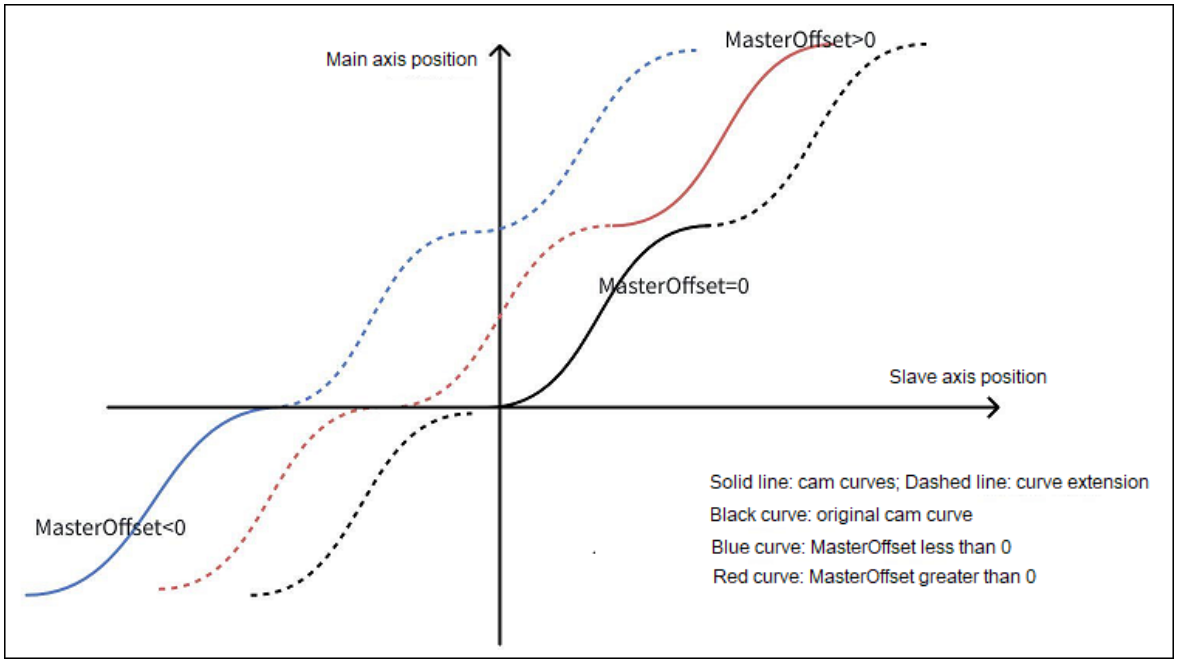


Figure 18-3-3-1-3-2 MC_CamIn

5. Cam table start point

   MC_CamIn: The cam table start point is (0,0).

   MC_CamInV2: The cam table start point can be specified arbitrarily.

6. Conditions for setting InSync and InCam

	MC_CamInV2	MC_CamIn
InCam	TRUE when the cumulative movement of the master axis reaches the input parameter MasterStartLength value after the instruction is executed.	TRUE when the master axis passes the start point of the cam table, which is specified by the input parameter StartPosition.

	MC_CamInV2	MC_CamIn
InSync	TRUE when the slave axis starts to follow the master axis and coupling is completed.	TRUE immediately after the slave axis starts to follow the master axis.

7. Conditions for the slave axis to start following

MC_CamInV2	MC_CamIn
The accumulative movement of the master axis reaches the MasterStartLength value (same sign as the direction of the master axis velocity).	The position of the master axis when the slave axis starts following the master axis is calculated based on the StartMode, MasterStartDistance, and StartPosition.

8. Re-execution

MC_CamInV2	MC_CamIn
It can be re-triggered after any parameters are modified. The modified parameters take effect immediately after the instruction is re-executed.	It can be re-executed after some parameters are modified. The modified parameters take effect in the next cycle.

9. Effective directions of the master axis

Master Axis Direction Parameter	MC_CamInV2	MC_CamIn
CAM_NO_DIRECTION	√	√
CAM_ABS_POSITIVE	√	√
CAM_REL_NEGATIVE	√	×
CAM_ABS_NEGATIVE	√	√
CAM_REL_NEGATIVE	√	×

10. Master axis offset

    MC_CamInV2: No modulo operation will be performed if the MasterOffset is greater than the master axis phase in corresponding camming cycle.

    MC_CamIn: Modulo operation will be performed if the MasterOffset is greater than the master axis phase in corresponding camming cycle.

	MC_CamInV2	MC_CamIn
MasterOffset greater than the master axis phase in corresponding camming cycle	No modulo operation.	Modulo operation of the MasterOffset will be performed based on the master axis phase in corresponding camming cycle.

18.3.3.1.4 Axis Group Instructions

- **General specifications for axis group**
  - **Multiple execution of axis group instructions**

    Other instructions can be executed during the execution of an axis group instruction.

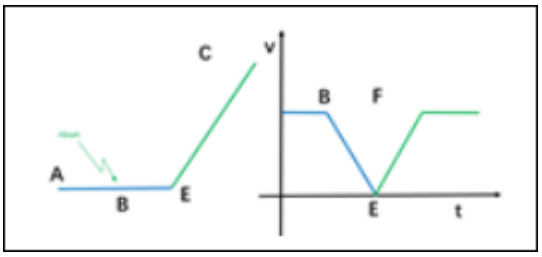
Axis group motion control instructions can be executed immediately or stored in a buffer for later execution, which is controlled by the parameter BufferMode. Up to eight instructions can be stored in each axis group.

The parameter BufferMode is defined as follows. mcAborting is executed immediately, while all others are buffered.

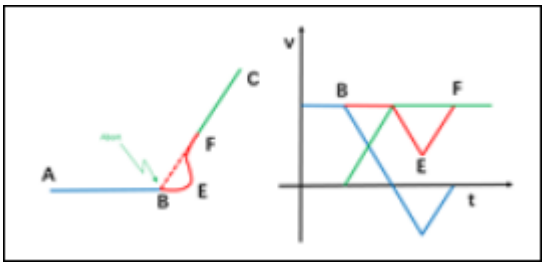
- **BufferMode**
  - mcAborting: Aborts the instruction being executed and switches to this instruction.
    New motion is re-planned starting from the current position and a transition trajectory is used to achieve smooth transition (velocity coordination) between the current trajectory and the new trajectory.

    This mode supports three transition modes, which are specified by the TransitionMode:

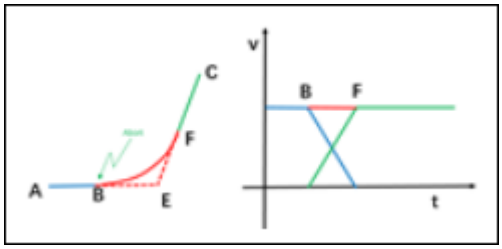
    (1) Transition disabled (equivalent to the effect of mcBuffered)
    TMNone: No transition curve.



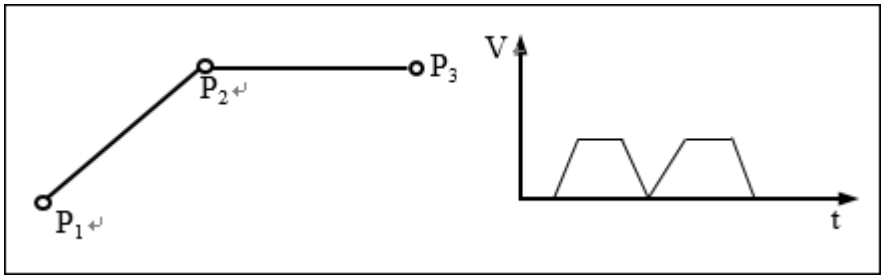
    (2) Reset transition
    TM_RESET_ABORT: Deceleration is performed with the deceleration and jerk calculated based on the maximum values of the set motion parameters multiplied by the conversion factor.



    (3) Deceleration transition
    TM_DEC_ABORT: Deceleration is performed with the deceleration and jerk set by the new aborting instruction.

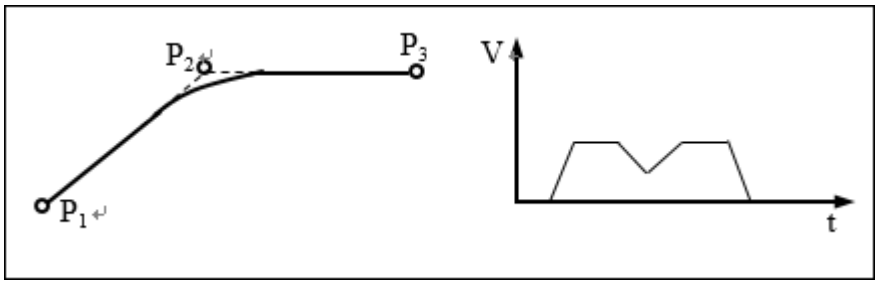


  - mcBuffered: Buffers this instruction and executes it after the current instruction is completed.



  - Blending: mcBlendingHigh, mcBlendingLow, mcBlendingPrevious, mcBlendingNext: The velocities of the two adjacent instructions are blended in the specified manner in

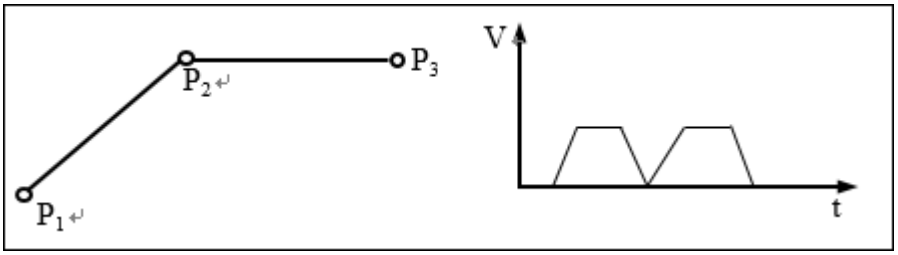
TransitionMode.



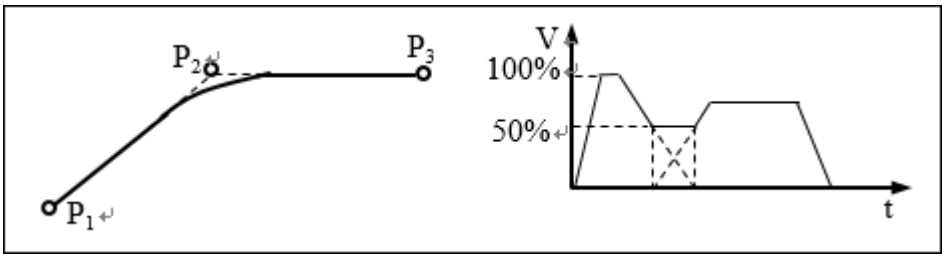
- **TransitionMode**

  This variable specifies how to blend the velocities of two instructions when Blending is specified for the BufferMode.

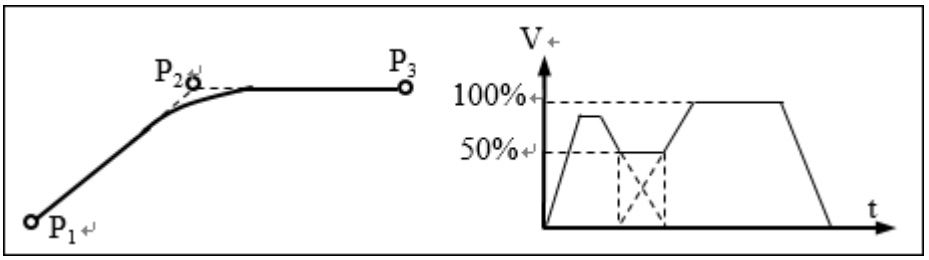
  - TMNone: Insert no transition curve. If TMNone is selected in a blending mode, the result is consistent with the transition between the velocities of the two instructions in buffered mode.



  - TMStartVelocity: Transition with given start velocity. Blending transition at a certain ratio (0, 1] of the actual maximum velocity of the current motion control instruction. Effective for transitions between any motion control instructions.



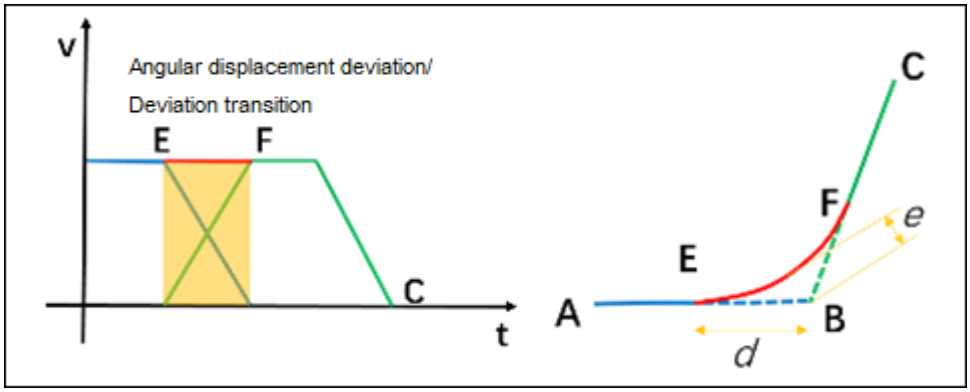
  - TMDefinedVelocity: Transition with defined velocity. Blending transition at a certain ratio (0, 1] of the actual maximum velocity of the instruction following the current motion control instruction. Effective for transitions between any motion control instructions.



  - TMCornerDistance: Transition with given corner distance. In Cartesian space, the distance to the end is specified by the preceding instruction in the main dimensional space, and the transition trajectory is initiated with this distance to connect the adjacent trajectory. The distance refers to a generalized distance, which depends on the main unit type setting for the axis group:
    - If the main unit is position, the parameter is the displacement from the start to the end of the transition.
    - If the main unit is orientation, the parameter is the orientation change from the start to the end of the transition.
    - If the main unit is a percent, the parameter is the percent (0, 1) of the path distance from the start to the end of the transition to the total path displacement.

Effective when both adjacent instructions are Cartesian space instructions (LIN/CIRC); otherwise, they are processed with transition disabled.

- TMMaxCornerDeviation: Transition with given maximum corner deviation. The maximum position deviation in the position subspace during the transition between adjacent instructions in Cartesian space. The adjacent trajectories are connected by constructing an appropriate transition trajectory to keep the maximum position deviation within a certain range. This is only the position deviation, independent of the main unit. Effective when both adjacent instructions are Cartesian space instructions (LIN/CIRC); otherwise, they are processed with transition disabled.



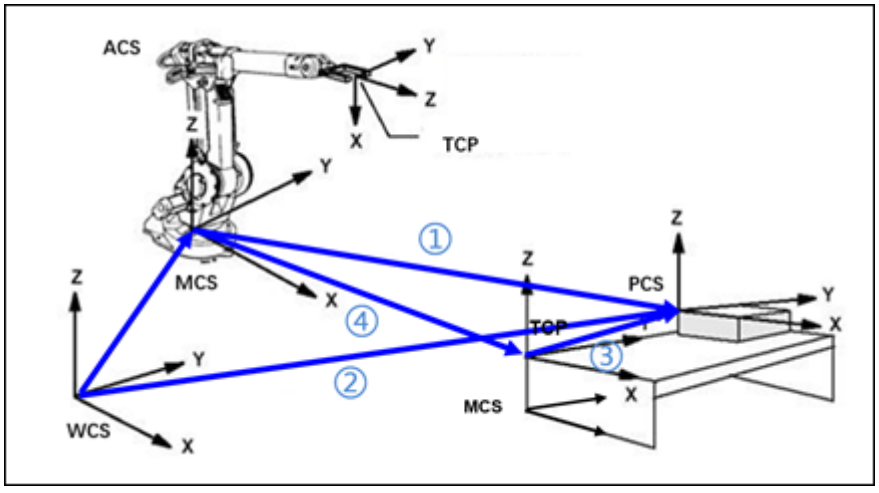
- **Axis group trajectory curve types**

  The trajectory curve types of the axes in the axis group do not affect the trajectory curve type of the axis group. The trajectory curve type eCurveType of the axis group supports three types of curves: trapezoidal curve, cubic S-curve, and seventh-order S-curve.

Curve Type	Curve Code	Enumeration Value	Continuity
Trapezoidal curve	TRAPEZOIDAL_CURVE	0	C1
Cubic S-curve	CUBIC_CURVE	1	C2
Seventh-order S-curve	SEVENTH_CURVE	2	C3

- **Definitions of coordinate system types related to axis groups**

  The core function of an axis group is coordinated motion. The coordinate system types (CoordSystem) involved in the axis group are shown in the following figure, and the corresponding names, definitions, and descriptions are shown in the following table.



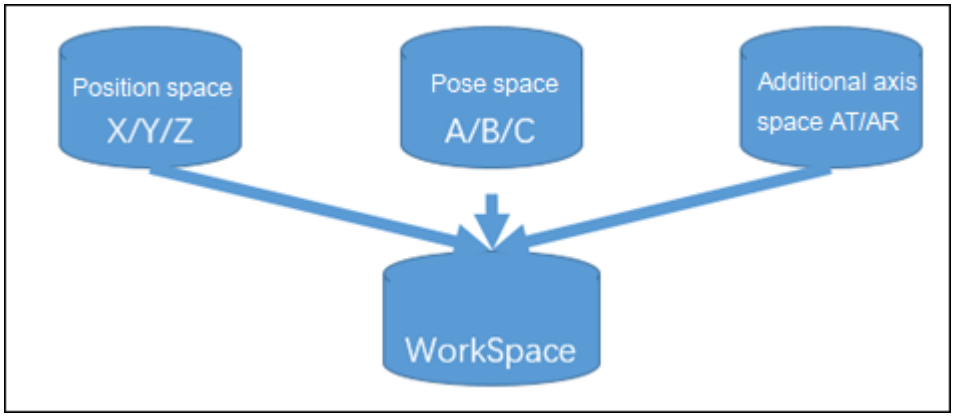
Coordinate System	Definition	Description
Axis coordinate system	ACS	Axis related.
Motor axis coordinate system	MACS	Motor axis (actual physical axis).
Joint axis coordinate system	JACS	Joint axis (logical joint axis).
Machine coordinate system	MCS	Base coordinate system for the axis group, relative to the world coordinate system.
Part coordinate system	PCS	Work space.
MCS-based PCS	PART_BASE_MCS = 0	Static coordinate system ①: PCS defined based on the MCS for an axis group.
WCS-based PCS	PART_BASE_WCS = 1	Static coordinate system ②: PCS defined based on the WCS.
End dynamic coordinate system	DYN_PART_BASE_GROUP = 6	Dynamic coordinate system ③: Dynamically fixed PCS defined based on the TCP point at the end of another axis group.
Dynamic PCS based on the X axis	DYN_PART_BASE_AXIS_X = 7	Dynamic coordinate system ④: Dynamically fixed PCS based on the X direction of an axis.
Dynamic PCS based on the A rotary axis	DYN_PART_BASE_AXIS_A = 8	Dynamic coordinate system ④: Dynamically fixed PCS based on a rotation direction (around Z axis) of axis A.
World coordinate system	WCS	The reference space shared by all MCSs.

- **Axis group space composition and configuration**

  The axis group coordinate space can be divided into two spaces: Axis group work space (WorkSpace) and joint axis drive space (AxisSpace).

  - **Work space configuration**

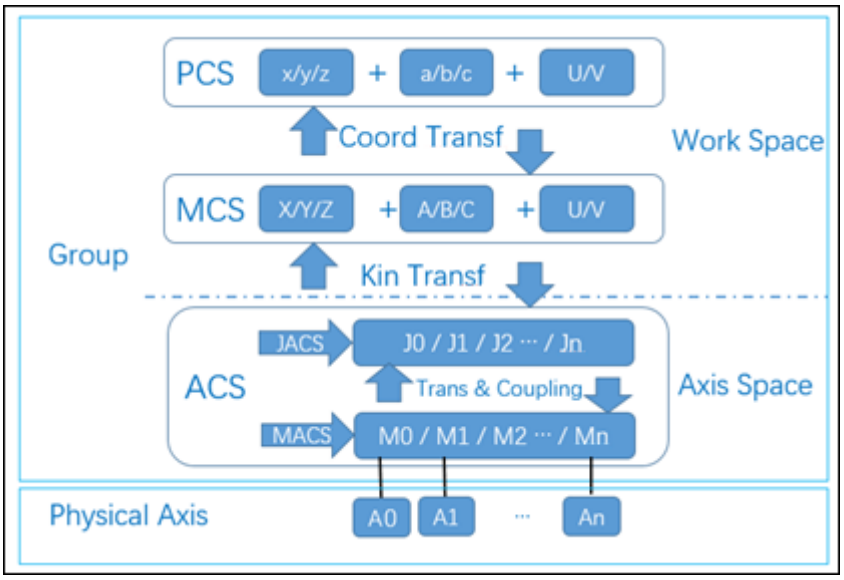
    Axis group work space (WorkSpace): The operational space constituted by the Cartesian space where the TCP at the end of the axis group is located and its auxiliary axis. It is composed of position space, orientation space, and auxiliary axis space. According to the dimension of the axis group space, configure the position X/Y/Z subspaces the orientation A/B/C subspaces, and the auxiliary axis AR/AT subspaces to form the axis group work space.



- **Axis space configuration**
  The axis group AxisSpace is divided into two subspaces, the motor axis coordinate system (MACS) and the joint axis coordinate system (JACS). The AxisSpace describes the software limits of the joint space by configuring the positive limit (fPositiveLimit) and negative limit (fNegativeLimit).
- **Coordinate system transformation**
  Workpiece coordinate system transformation (PCS←→MCS).

  Kinematic model transformation (axis space, base coordinate system transformation ACS←→MCS).

  Axis coordinate system decoupling and anti-decoupling transformation (MACS←→JACS).



18.3.3.1.5 Position Retention upon Power Failure

When a motor with an absolute encoder is used, its actual position is retained and restored after a power cycle of the PLC or servo drive. When the motor is used with a supported servo drive of Inovance, position data can be saved on the servo drive or the PLC. When the motor is used with an unsupported servo drive of Inovance or a third party, position data can be saved on the PLC only. Additionally, if the servo drive is set to absolute rotary mode, position data can be saved on the PLC only, and Rotary Mode must be selected in the axis configuration interface.

Working principle

(1) Position data saved on the servo drive

With this option, when the servo drive is power-cycled, the PLC will obtain the motor position data saved in the servo drive and use it to restore the PLC position system, achieving position retention.

Note:

- In this mode, the servo drive must be set to absolute linear mode without encoder overflow alarm (i.e., H02.01 = 3); otherwise, this function cannot be used.
- This option does not restore the user-defined position offset in the PLC.
- Currently, this option supports the use of Inovance servo drives only. If another manufacturer's servo drive is used, position data can be saved on the PLC only.

(2) Position data saved on the PLC with the axis being set to linear mode

With this option, when the PLC is power-cycled, it will obtain the position data before power failure from the retentive memory area and combine it with the position data after power-on to restore the PLC position system, achieving position retention.

Note:

- For linear axes, this option does not support the use of the servo drive being set to incremental mode, absolute rotary mode, or single-turn absolute mode.
- For linear axes, this option can restore the user-defined position offset in the PLC.
- For linear axes, this option supports multi-turn position retention.

(3) Position data saved on the PLC with the axis being set to rotary mode and the servo drive being set to multi-turn mode

With this option, when the PLC is power-cycled, it will obtain the position data before power failure from the retentive memory area and combine it with the position data after power-on to restore the PLC position system, achieving position retention.

Note:

- For rotary axes, this option does not support the use of the servo drive being set to incremental mode, absolute rotary mode, or single-turn absolute mode.
- For rotary axes, this option can restore the user-defined position offset in the PLC.
- For rotary axes, this option supports multi-turn position retention.

(4) Position data saved on the PLC with the axis being set to rotary mode and the servo drive being set to single-turn mode

This option is designed to accommodate the absolute rotary mode of the servo drive. With this option and in the absolute rotary mode, the pulse position of the servo drive varies within the user-defined range.

Note:

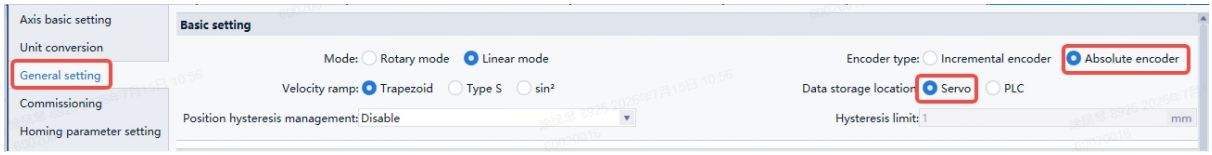
- For rotary axes, this option supports the absolute rotary mode only.
- For rotary axes, this option does not restore the user-defined position offset in the PLC.
- For rotary axes, this option supports single-turn position retention only.

Examples

(1) Position data saved on the servo drive

1. Software interface settings
   In the "General setting" for the axis, select "Encoder Type" as "Absolute Encoder" and select "Data storage

location" as "Servo".



2. Servo drive settings

   Connect the encoder of the servo drive to a battery, and set the servo parameter H02.01 to 3 in the servo drive, and then power cycle the servo drive.

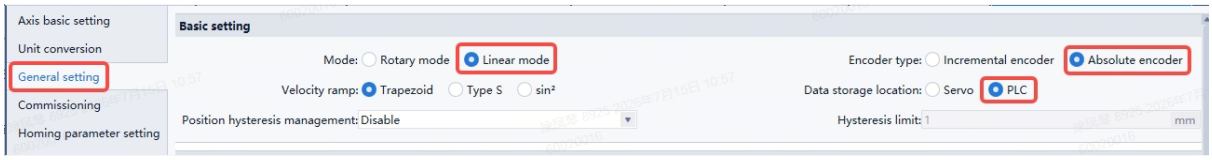
3. Servo drive power cycle

   After steps 1 and 2 are complete, compile and run the program. At this point, if the servo drive powers off, the position will be retained after a power cycle.

(2) Position data saved on the PLC with the axis being set to linear mode

1. Software interface settings

   In the "General setting" for the axis, select "Mode" as "Linear mode", select "Encoder type" as "Absolute encoder", and then select "Data storage location" as "PLC".



2. Servo drive settings

   Connect the encoder of the servo drive to a battery, and set the servo parameter H02.01 in the servo drive to 1 or 3, then power on the servo drive.

3. PLC power cycle

   After steps 1 and 2 are complete, compile and run the program. At this point, if the PLC powers off, the position will be retained after a power cycle.

(3) Position data saved on the PLC with the axis being set to rotary mode and the servo drive being set to multi-turn mode

1. Software interface settings

   In the "General setting" for the axis, select "Mode" as "Rotary mode", select "Encoder type" as "Absolute encoder", select "Data storage location" as "PLC", and then select "Servo operation mode" as "Multi-turn mode".



2. Servo drive settings

   Connect the encoder of the servo drive to a battery, and set the servo parameter H02.01 to 1 or 3 in the servo drive, and then power cycle the servo drive.

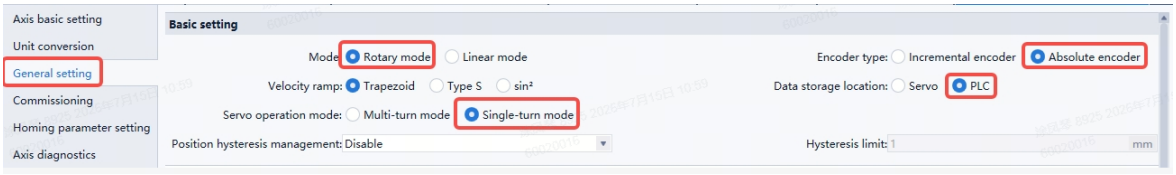
3. PLC power cycle

   After steps 1 and 2 are complete, compile and run the program. At this point, if the PLC powers off, the position will be retained after a power cycle.

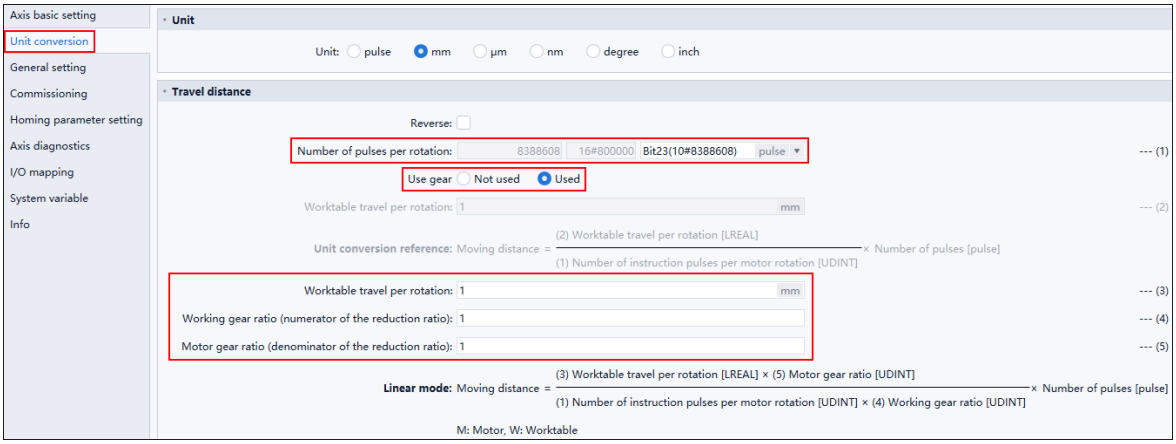
(4) Position data saved on the PLC with the axis being set to rotary mode and the servo drive being set to single-turn mode

Under this option, to match the absolute rotary mode of the servo drive, some configurations must be made on both the iFA and the servo drive, and these configurations must be correlated.

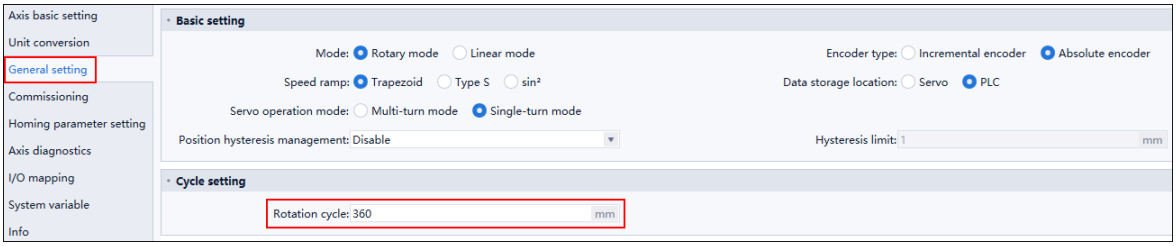
1. Servo drive settings
   - Set H02.01 to 2 (Rotary mode with absolute positioning);
   - Set the number of motor pulses corresponding to one full rotation of the axis by configuring H05.52 and H05.54;
   - If the number of motor pulses corresponding to one full rotation of the axis exceeds 2^31, set the gear ratio H05.07 and H05.09 to proportionally reduce the number of motor pulses, ensuring that the number of motor pulses does not exceed 2^31.
2. Software interface settings
   - In the "General setting" for the axis, select "Mode" as "Rotary mode", select "Encoder type" as "Absolute encoder", select "Data storage location" as "PLC", and then select "Servo operation mode" as "Single-turn mode".



   - In the "Unit conversion" for the axis, set the "Number of pulses per rotation", "Worktable travel per rotation," "Working gear ratio," and "Motor gear ratio."



   - In the "General setting" for the axis, set the "Rotation cycle" to 360°.



NOTE

The parameters set above have the following relationships with the parameters set in the servo drive:

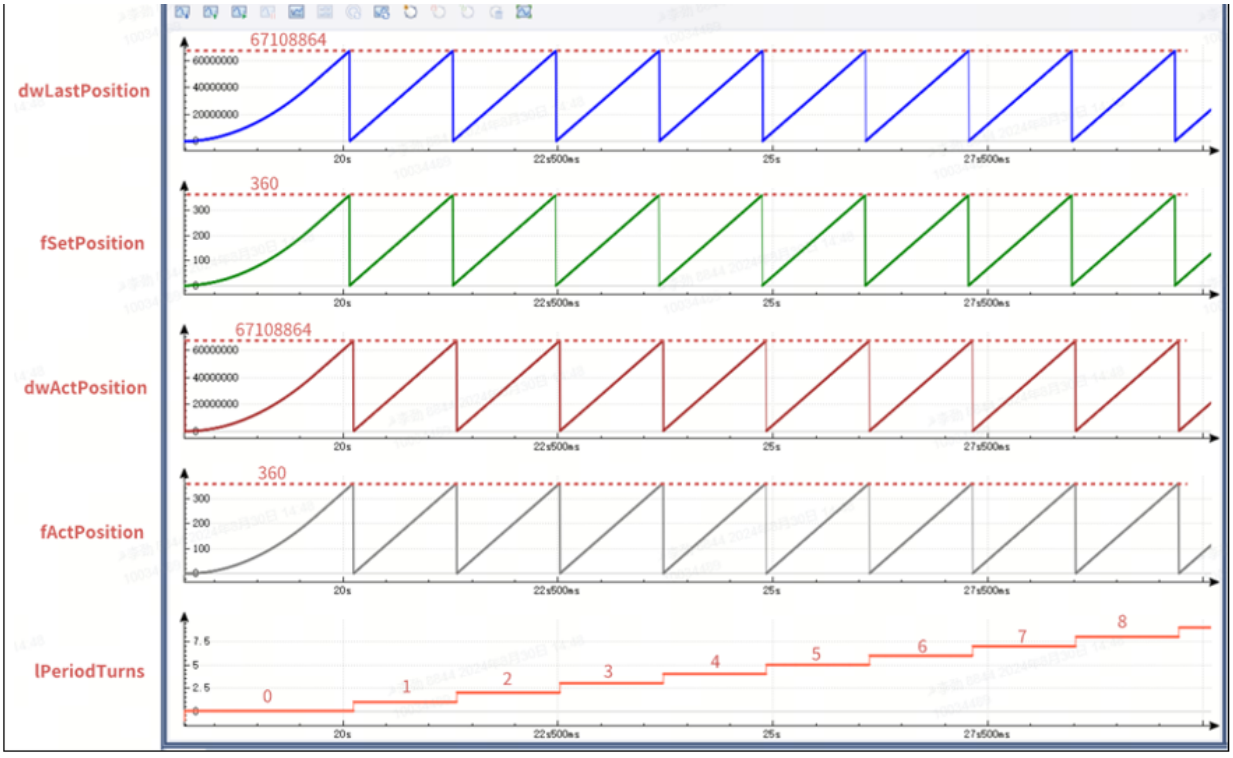
(H05.52 + H05.54 * 2^32) * H05.09 / H05.07 = (Number of pulses per rotation * Working gear ratio / (Worktable travel per rotation* Motor gear ratio)) * Rotation cycle

3. Example:

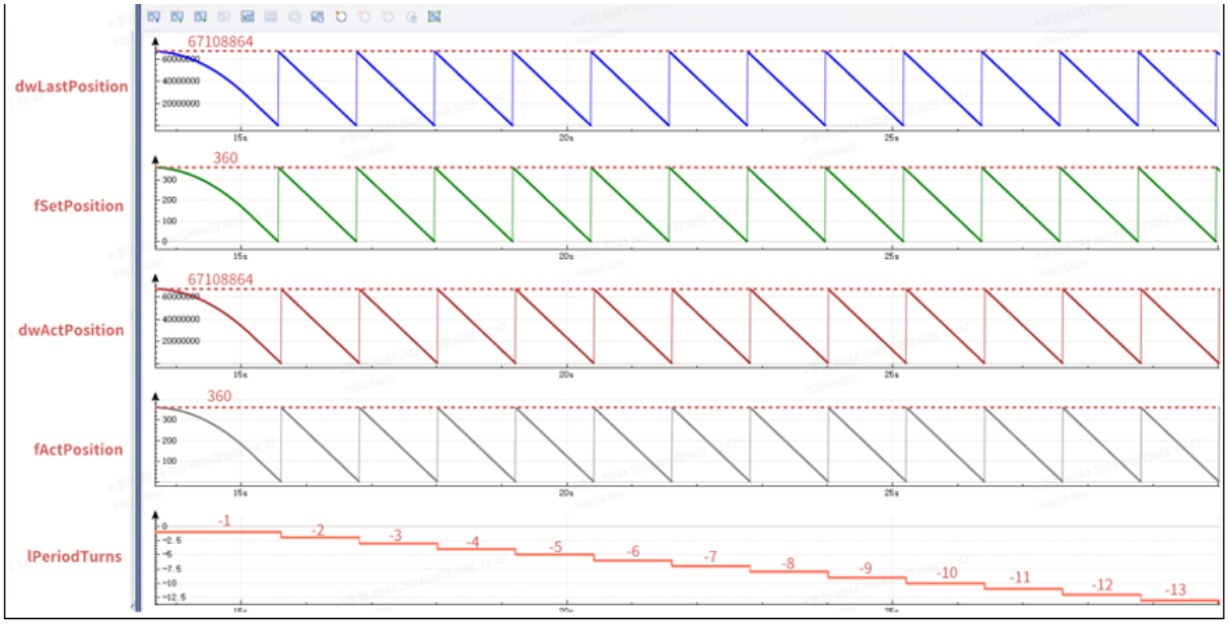
The relevant settings on the servo drive and the iFA are as follows:

	Parameter	Value
Servo drive	H02.01	2 (Absolute rotary mode)
	H05.52	2^26 (67108864)
	H05.54	0
	H05.07	1
	H05.09	1
iFA	Rotation cycle	360°
	Number of pulses per rotation	2^23
	Worktable travel per rotation	360°
	Working gear ratio	8
	Motor gear ratio	1

When you set the pulse unit to 2^26 according to the parameters in the preceding table, the servo drive will be reset for another rotation. The diagram for the axis moving in the positive motion is as follows:



The diagram for the axis moving in the negative motion is as follows:



Correspondence between the controller mode and the servo drive's absolute system

The servo drive can be configured with different types of absolute systems, and iFA can be also used to set different axis types and modes for the PLC controller. However, the absolute system in the servo drive may not match the axis type and mode set in the PLC controller, so it is necessary to know their matching relationship to allow users to make different combinations during use. See the following table for details:

Controller Mode (Through iFA)		Servo Drive's Absolute System	Remarks
Rotary mode	Incremental encoder	0 (Incremental mode)	Position reset after power-on
		1 (Absolute linear mode)	After a power cycle, lTurn is reset, and single-turn position is retained.
		3 (Absolute linear mode without encoder overflow alarm)	After a power cycle, lTurn is reset, and single-turn position is retained.
	Absolute encoder + PLC + single-turn mode	2 (Absolute rotary mode)	Infinite rotation can be achieved, but the position offset recorded on the axis will not be restored.
	Absolute encoder + PLC + multi-turn mode	1 (Absolute linear mode)	Position system is restored after a power cycle.
		3 (Absolute linear mode without encoder overflow alarm)	Position system is restored after a power cycle.
	Absolute encoder + servo drive	3 (Absolute linear mode without encoder overflow alarm)	Position system is restored after a power cycle, but if the PLC powers off, the position offset recorded on the axis will not be restored.
Linear mode	Incremental encoder	0 (Incremental mode)	The position data is reset after a power cycle.

Controller Mode (Through iFA)		Servo Drive's Absolute System	Remarks
		1 (Absolute linear mode)	After a power cycle, lTurn is reset, and single-turn position is retained.
		3 (Absolute linear mode without encoder overflow alarm)	After a power cycle, lTurn is reset, and single-turn position is retained.
	Absolute encoder + PLC	1 (Absolute linear mode)	Position system is restored after a power cycle.
		3 (Absolute linear mode without encoder overflow alarm)	Position system is restored after a power cycle.
	Absolute encoder + servo drive	3 (Absolute linear mode without encoder overflow alarm)	Position system is restored after a power cycle, but if the PLC powers off, the position offset recorded on the axis will not be restored.

For incremental encoders, if the PLC powers off, the position offset will not be restored; if servo drive powers off, the position offset will be restored.

Special cases

In some special cases, after the position system is restored, homing must be performed for the servo drive before the axis operation; otherwise, the axis may move unexpectedly.

Position Retention Option	Special Cases
Position data saved on the servo drive	Switched the incremental encoder/absolute encoder.
	Switched the PLC/servo drive.
	Modified the axis parameters Linear Mode/Rotary Mode, Rotation Cycle, Unit Conversion, etc.
	Modified the servo parameters Gear Ratio (6091h), Rotation Direction, etc.
	The motor is not provided with an absolute encoder or has been replaced.
	The absolute encoder is used for the first time or the servo battery is depleted.
	The servo drive abnormality causes loss of absolute position information.
	Position offset is specified and the controller is replaced.
Position data saved on the PLC	Switched the incremental encoder/absolute encoder.

Position Retention Option	Special Cases
	Switched the PLC/servo drive.
	Modified the axis parameters Rotation Cycle, Unit Conversion, etc.
	The servo drive is not in absolute mode or the motor is not an absolute encoder motor.
	Modified the servo parameters Rotary/Linear Mode, Rotation Direction, and Gear Ratio (6091h).
	The absolute encoder is used for the first time or the servo battery is depleted.
	The PLC power-failure retention variable area experiences an exception or is cleared.
	After the PLC powers off, the motor has rotated multiple turns.
	The controller/motor has been replaced.

18.3.3.2 Object Definitions

18.3.3.2.1 Data Object Description

The following data structures are replaced with aliases to standardize the structure and enumeration definitions (PLCOpen standard) and ensure project compatibility. **The existing data type definitions are identical with the newly added data alias type definitions.**

Existing Data Type Definition	Newly Added Data Alias Type Definition	Type
ST_BUS_SERVO_AXIS	MC_BUS_SERVO_AXIS_REF	Structure
MC_HOMING_PARAMETER	MC_HOMING_PARAMETER_REF	Structure
MC_FBERROR	MC_FBERROR_REF	Structure
MC_CAM_TABLE	MC_CAM_TABLE_REF	Structure
MC_CAM_NODE_TYPE	MC_CAM_NODE_TYPE_REF	Structure
MC_CAM_KEY_NODE	MC_CAM_KEY_NODE_REF	Structure
MC_CAM_FITTING_NODE	MC_CAM_FITTING_NODE_REF	Structure
MC_CAM_IN_STATE	MC_CAM_IN_STATE_REF	Structure
S_AXIS_SPACE	MC_AXIS_SPACE_REF	Structure
S_WORK_SPACE	MC_WORK_SPACE_REF	Structure
S_GROUP_CFG	MC_GROUP_CFG_REF	Structure

Existing Data Type Definition	Newly Added Data Alias Type Definition	Type
E_AXES_GROUP_STATE	MC_AXES_GROUP_STATE	Enumeration
E_CIRCE_TYPE	MC_CIRCE_TYPE	Enumeration
E_COORD_TYPE	MC_COORD_TYPE	Enumeration
E_CURVE_TYPE	MC_CURVE_TYPE	Enumeration
E_ERROR_STOP	MC_ERROR_STOP	Enumeration
E_UNIT_SPACE	MC_UNIT_SPACE	Enumeration
E_STOP_MODE	MC_GROUP_STOP_MODE	Enumeration
E_CORNER_TYPE	MC_CORNER_TYPE	Enumeration
E_HOME_MODE	MC_HOME_MODE	Enumeration
E_LOAD_PATH_STATUS	MC_LOAD_PATH_STATUS	Enumeration
E_PATH_CHOICE	MC_PATH_CHOICE	Enumeration
E_PART_STATUS	MC_PART_STATUS	Enumeration

18.3.3.2.2 Single-Axis Data Object Definitions

18.3.3.2.2.1 AXIS_REF: Axis Structure

Variable	Data Type	Description
wAxisStructID	WORD	Axis structure identifier. It represents that this structure is used for a certain type of axis, such as the ECAT control axis, local control axis, ECAT encoder, or local encoder.
AxisID	INT	Axis identifier.
bVirtual	BOOL	Specify whether it is a simulated axis: 0: Real axis. 1: Virtual axis.
bCommunication	BOOL	Specify whether ECAT communication is established.
eAxisState	18.3.3.2.2.5 MC_AXIS_STATE: Axis State Machine Enumeration	Axis state machine.
bAxisEnableState	BOOL	Axis enable signal state, corresponding to the value of 6041 or the enable state of a local axis.

Variable	Data Type	Description
iCommandAbortCount	INT	Command aborted flag. This variable increases when each instruction is executed, and the increased variable value is recorded. When this value changes during execution, it indicates that the FB is aborted.
fTaskCycle	LREAL	Set value of a task cycle.
uiDriveError	UINT	Drive fault code: Correspond to 603F in the object dictionary.
bError	BOOL	Fault status.
dwErrorID	DWORD	Error identifier.
fbeFBError	ARRAY [0..14] OF 18.3.3.2.2.4 MC_FBERROR_REF: Fault Record Structure	Identify log information and record the fault of an instruction.
fMaxDecelDistance	LREAL	Maximum deceleration distance.
fSwLimitDeceleration	LREAL	Deceleration rate in position limit state.
eDirection	18.3.3.2.2.9 MC_DIRECTION: Motion Direction Enumeration	Current motion direction of an axis.
eRampType	18.3.3.2.2.15 MC_RAMPTYPE: Ramp Type Enumeration	Ramp type. 0: Trapezoidal. 1: S-curve. 2: Sin2.
eControllerMode	18.3.3.2.2.7 MC_CONTROLLER_MODE: Controller Mode Enumeration	Axis control mode. 10: Torque control. 9: Velocity control. 8: Position control.
eRealControllerMode	18.3.3.2.2.7 MC_CONTROLLER_MODE: Controller Mode Enumeration	Actual controller mode from an axis: Corresponds to object dictionary 6061.
iMovementType	INT	Specify the axis type. 0: Cyclic rotary mode. 1: Linear mode.
fSetPosition	LREAL	Set position (unit) calculated in this cycle.

Variable	Data Type	Description
fActPosition	LREAL	Actual position (unit).
dwActPosition	DWORD	Encoder value corresponding to the actual position of an axis in the current cycle, corresponding to the value (pulse) of 6064.
lTurn	LINT	Number of turns the axis position parameter exceeds 2^32.
fPositionPeriod	LREAL	Set modulus length (unit)
dwOneTurn	DWORD	Number of pulses per revolution in rotary mode.
dwRatioTechUnitsDenom	DWORD	Denominator of the gear ratio.
fRatioTechUnitsNum	LREAL	Numerator of the gear ratio.
fOffsetPosition	LREAL	Offset between the linear axis position and the servo position. When the MC_SetPosition instruction is executed, this value changes according to the set value and represents the cumulative offset between the set axis position and the servo position.
fOffsetPeriod	LREAL	Position offset in rotary mode. When the MC_SetPosition instruction is executed, this value changes according to the set value.
wAxNo	WORD	Axis number, corresponding to the axis array index value.
fSetVelocity	LREAL	Set velocity (unit) for current cycle.
fActVelocity	LREAL	Actual velocity (unit).
fSetAcceleration	LREAL	Set acceleration (unit) for current cycle.
fActAcceleration	LREAL	Actual acceleration (unit).
fSetJerk	LREAL	Planned jerk.
fSetTorque	LREAL	Set torque (unit) for current cycle.
fActTorque	LREAL	Actual torque (unit).
fFollowingError	LREAL	Converted following error value.
fSWLimitPositive	LREAL	Positive software limit position.
fSWLimitNegative	LREAL	Negative software limit position.
bSWLimitEnable	BOOL	Software position limit enable flag. TRUE: Enabled.
bHWLimitEnable	BOOL	Hardware position limit enable flag. TRUE: Enabled.
lSetTargetPosition	LINT	Target position (unit) of a pulse axis.

Variable	Data Type	Description
bIsDriverHoming	LREAL	Drive homing and pulse homing flag: Drive homing (TRUE during the drive homing); Pulse axis homing (Done flag of the function block after homing delay).
bHomingWithParaFlag	BOOL	PLC homing start flag.
HomeSwitch	BOOL	Home switch: Corresponds to bit2 of object dictionary 60FD.
PositiveLimitSwitch	BOOL	Positive limit switch: Corresponds to bit1 of object dictionary 60FD.
NegativeLimitSwitch	BOOL	Negative limit switch: Corresponds to bit0 of object dictionary 60FD.
LatchInput	BOOL	Touch probe 1 signal switch.
fRefPosition	LREAL	Filtered position, used for the new MasterSource.
fRefVelocity	LREAL	Filtered velocity, used for the new MasterSource and to calculate fRefPosition.
fExternalVel	LREAL	External velocity reference.
bHomed	BOOL	TRUE when the controller home is defined. The absolute encoder saves the PLC mode, with the home being defined by default upon power-up. Even if the bHomed is set to TRUE, the home must be re-defined in any of the following cases. • Changing the position counter or unit conversion setting. For example, the MC_SetPosition or SMC_ChangeGearingRatio instruction is executed, or related variables for position calculation are modified online. • A fault on the servo drive or missing absolute position information due to an operation. For example, the encoder cable is disconnected or the absolute encoder data is cleared.
uiPhysSlaveAddr	UINT	Physical address of the slave corresponding to an axis.
uiAliasSlaveAddr	UINT	Alias address of the slave corresponding to an axis.

18.3.3.2.2.2 AXIS Communication State Machines

Communication Status Value (Decimal)	Description
65535	Initial.
0	Unknown.
1	Initialized.

Communication Status Value (Decimal)	Description
2	Pre-operational.
3	Booting.
4	Safe operating.
20	Operating with the position system not restored.
90	Absolute encoder position system recovering.
95	Incremental encoder position system recovering.
100	Operating with the position system not restored.
1017	Initialization failure.
1018	Pre-operational fault.
1019	Booting fault.
1020	Safe operating fault.
1024	Operating fault.

18.3.3.2.2.3 MC_BUS_SERVO_AXIS_REF: Bus Servo Axis Structure

Example of accessing CIA402 object : The value of the CIA402 object can be obtained from the axis in the following way: Axis.uAxisEtcRT.stBusServoAxis.diEtcDigitalInput.

Variable	Type	Description
uiEtcControlWord	UINT	Control word (6040Hex)
uiEtcTargetPosition	DINT	Target position (607AHex)
uiEtcProfileVelocity	DINT	Target velocity (60FFHex)
iEtcTargetTorque	INT	Target torque (6071Hex)
siEtcModesOfOperation	SINT	Operation mode (6060Hex)
iEtcTouchProbeFunction	UINT	Touch probe function (60B8Hex)
udiEtcMaxProfileVelocity	UDINT	Maximum velocity (607FHex) 100* equivalent
uiEtcForwardTorqueLimit	UINT	Forward torque limit (60E0Hex)
uiEtcBackwardTorqueLimit	UINT	Backward torque limit (60E1Hex)
diEtcDigitalOutput	DINT	Forced digital output (60FEHex sub-index=1)
diEtcPositionOffset	DINT	Position offset (60B0Hex)
diEtcVelocityOffset	DINT	Velocity offset (60B1Hex)
diEtcTorqueOffset	DINT	Torque offset (60B2Hex)

Variable	Type	Description
udiEtcTorqueSlope	UDINT	Torque slope (6087Hex)
udiEtcProfileVelocity	UDINT	Profile velocity (6081Hex)
udiEtcProfileAcceleration	UDINT	Profile acceleration (6083Hex)
udiEtcProfileDeceleration	UDINT	Profile deceleration (6084Hex)
uiEtcErrorCode	UINT	Fault code (603FHex)
uiEtcStatusWord	UINT	Status word (6041Hex)
uiEtcActualPostion	DINT	Actual position (6064Hex)
iEtcActTorque	INT	Actual torque (6077Hex)
siEtcActModeOfOperation	SINT	Operation mode display (6061Hex)
uiEtcTouchProbeStatus	UINT	Touch probe status (60B9Hex)
diEtcTouchProbePosition1R	DINT	Touch probe position 1 rising edge (60BAHex)
diEtcTouchProbePosition1F	DINT	Touch probe position 1 falling edge (60BBHex)
diEtcTouchProbePosition2R	DINT	Touch probe position 2 rising edge (60BCHex)
diEtcTouchProbePosition2F	DINT	Touch probe position 2 falling edge (60BDHex)
diEtcDigitalInput	DINT	Digital input (60FDHex)
diEtcActVelocity	DINT	Actual velocity (606CHex)
diEtcActFollowingError	DINT	Actual following error (60F4Hex)
uiEtcTouchProbeEdgeCounter1R	UINT	Touch probe position 1 rising edge counter (60D5Hex)
uiEtcTouchProbeEdgeCounter1F	UINT	Touch probe position 1 falling edge counter (60D6Hex)
uiEtcTouchProbeEdgeCounter2R	UINT	Touch probe position 2 rising edge counter (60D7Hex)
uiEtcTouchProbeEdgeCounter2F	UINT	Touch probe position 2 falling edge counter (60D8Hex)

18.3.3.2.2.4 MC_FBERROR_REF: Fault Record Structure

In the fbeFBError structure, up to 15 axis fault messages can be recorded, and they are stored in fbeFBError.wID in the order of occurrence. Once the 15 records are occupied, each new fault will overwrite the oldest record. The values stored in fbeFBError.wID correspond to the decimal value of the MC_XXX.ErrorID.

Variable	Data Type	Description
wID	WORD	Fault code.

Variable	Data Type	Description
strErrorInstance	WSTRING[254]	Source of the fault.
tTimeStamp	TIME	Time stamp of the fault occurrence.
eAxisStateAtError	18.3.3.2.2.5 MC_AXIS_STATE: Axis State Machine Enumeration	Axis state at the time of the fault.
eControllerModeAtError	18.3.3.2.2.7 MC_CONTROLLER_MODE: Controller Mode Enumeration	Controller mode at the time of the fault.
uiDriveErrorAtError	UINT	Drive fault code at the fault occurrence.
wCommunicationStateAtError	WORD	Communication state at the time of the fault.

18.3.3.2.2.5 MC_AXIS_STATE: Axis State Machine Enumeration

Variable	Value	Description
AXIS_POWER_OFF	0	Power-off.
AXIS_ERRORSTOP	1	Error stop.
AXIS_STOPPING	2	Decelerating to stop.
AXIS_STANDSTILL	3	Standstill.
AXIS_DISCRETE_MOTION	4	Discrete motion.
AXIS_CONTINUOUS_MOTION	5	Continuous motion.
AXIS_SYNCHRONIZED_MOTION	6	Synchronized motion.
AXIS_HOMING	7	Homing.

18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration

Variable	Value	Description
mcAborting	0	Start FB immediately.
mcBuffered	1	Start FB after current motion has finished.
mcBlendingLow	2	Blending with the lowest velocity of both FBs.
mcBlendingPrevious	3	Blending with the velocity of the first FB.
mcBlendingNext	4	Blending with the velocity of the second FB.

Variable	Value	Description
mcBlendingHigh	5	Blending with the highest velocity of both FBs.

18.3.3.2.2.7 MC_CONTROLLER_MODE: Controller Mode Enumeration

Variable	Value	Description
MC_CSP	8	CSP (Cyclic synchronous position).
MC_CSV	9	CSV (Cyclic synchronous velocity).
MC_CST	10	CST (Cyclic synchronous torque).

18.3.3.2.2.8 MC_MOVE_MODE: Move Mode Enumeration

Variable	Value	Description
AbsoluteMode	0	Absolute positioning.
RelativeMode	1	Relative positioning.
VelocityMode	2	Velocity control.

18.3.3.2.2.9 MC_DIRECTION: Motion Direction Enumeration

Variable	Value	Description
DIR_SHORTEST	0	Shortest way.
DIR_POSITIVE	1	Positive direction.
DIR_NEGATIVE	2	Negative direction.
DIR_CURRENT	3	Current direction.
DIR_NODIRECTION	4	No direction.
DIR_FASTEST	5	Fastest way.

18.3.3.2.2.10 MC_STOP_MODE: Stop Mode Enumeration

Variable	Value	Description
ImmediateStop	0	The commanded velocity drops directly to 0 from current velocity.
FreeRunStop	1	The axis immediately stops and the servo drive is turned off.

18.3.3.2.2.11 MC_REFERENCE_TYPE: Data Type Enumeration

Variable	Value	Description
mcCommand	0	Commanded position (Value calculated in the previous task cycle, fSavePosition).
mcFeedback	1	Actual position (Value obtained in the same task cycle, fActPosition).
mcLatestCommand	2	Commanded position (Value calculated in the same task cycle, fSavePosition).
mcMasterPhase	3	Phase of the master axis selected when the slave axis starts camming.
mcPositionFilter	4	RefPosition after the MC_PositionFilter instruction is executed.

18.3.3.2.2.12 MC_COMBINE_MODE: Combine Mode Enumeration

Variable	Value	Description
mcAddAxes	0	Addition.
mcSubAxes	1	Subtraction.

18.3.3.2.2.13 MC_HOMING_MODE: Homing Mode Enumeration

Variable	Value	Description
PassHomeSwHomeSwTurn	1	Home switch + Zero switch + Positive/Negative limit. Traverse the home switch, search for the zero, and then reverse at the home.
InHomeSwHomeSwTurn	2	Home switch + Zero switch + Positive/Negative limit. Search for the zero within the home switch, and then reverse at the home.
PassHomeSwHome	3	Home switch + Zero switch + Positive/Negative limit. Traverse the home switch, search for the zero, and then reverse at the limit.
InHomeSw	4	Home switch + Zero switch + Positive/Negative limit. Search for the zero within the home switch, and then reverse at the limit.

Variable	Value	Description
PassLimitSw	5	Zero switch + Positive/Negative limit. Traverse the limit switch and then search for the zero. Homing can be started outside the limit.
HomeSwMaskDistanceHomeSwTurn	6	Home switch + Zero switch + Positive/Negative limit. Homing at home input mask distance, and then reverse at the home.
OnlyLimit	7	Positive/negative limit only, no zero switch.
HomeSwHoldTimeHomeSwTurn	8	Home switch + Positive/negative limit. Start homing after holding time, and reverse at the home.
ZeroSw	9	Zero switch + Positive/Negative limit. Start homing without a home. Homing cannot be started outside the limit.
ZeroPositionPreset	10	Zero position preset.

18.3.3.2.2.14 MC_LIMIT_REVERSE_MODE: Limit Reverse Mode Enumeration

Variable	Value	Description
Errorstop	0	No reverse turn/fault stop.
RevImmediateStop	1	Reverse turn/immediate stop.
RevDeceleration	2	Reverse turn/Deceleration stop.

18.3.3.2.2.15 MC_RAMPTYPE: Ramp Type Enumeration

Variable	Value	Description
TRAPEZ	0	Trapezoidal curve.
CURVE	1	S-curve.
SINSQUARE	2	Sin2 curve.

18.3.3.2.2.16 MC_ZERO_SIGNAL: Zero Signal Enumeration

Variable	Value	Description
ZPhase	0	Z-phase input.
ExternalSignal	1	External home input.

18.3.3.2.2.17 MC_CONTROLLER_MODE_SWITCH: Controller Mode Switch Mode Enumeration

Variable	Value	Description
MC_AUTO	0	Automatic switching.
MC_MANUAL	1	Manual switching.

18.3.3.2.3 Synchronous Axis Data Object Definitions

18.3.3.2.3.1 MC_CAM_TABLE_REF: Cam Table Structure

Variable	Type	Description
strCamTableName	STRING	Name of the cam table.
iNodeType	INT	Node type: 1: Key node. 2: Dense node.
iNodeNumber	UINT	Number of nodes. Up to 65,535 nodes are supported.
pNodePtr	POINTER TO BYTE	Pointer to node data.

18.3.3.2.3.2 MC_CAM_NODE_TYPE_REF: Cam Node Type Structure

Variable	Type	Description
KeyNode	18.3.3.2.3.3 MC_CAM_KEY_NODE_REF: Key Node Structure	Cam curve key node.
FittingNode	18.3.3.2.3.4 MC_CAM_FITTING_NODE_REF: Dense Node Structure	Cam curve dense node.

18.3.3.2.3.3 MC_CAM_KEY_NODE_REF: Key Node Structure

Variable	Type	Description
fPhase	LREAL	Phase (of the master axis) at the node.
fDistance	LREAL	Displacement (of the slave axis position) at the node.
fVel	LREAL	Velocity at the node.

Variable	Type	Description
fAcc	LREAL	Acceleration at the node.
fJerk	LREAL	Jerk at the node.
iCurveType	INT	Curve type: 1: Straight line. 2: Parabolic curve. 3: Cubic curve. 4: Quintic curve. 5: Septic curve. 6: Cosine acceleration curve. 7: Sine acceleration curve. 8: Modified trapezoidal acceleration curve.

18.3.3.2.3.4 MC_CAM_FITTING_NODE_REF: Dense Node Structure

Variable	Type	Description
fPhase	LREAL	Phase (of the master axis) at the node.
fDistance	LREAL	Displacement (of the slave axis position) at the node.

18.3.3.2.3.5 MC_CAM_IN_STATE_REF: Camming State Structure

Variable	Type	Description
strCamTableName	STRING	Name of the cam table.
iMasterID	INT	Master axis identifier.
iSlaveID	INT	Slave axis identifier.
diCamCnt	DINT	Number of camming cycles executed.
diNodeCnt	DINT	Current node index.
fPhase	LREAL	Master axis phase.
fDistance	LREAL	Slave axis displacement.
fPhaseShift	LREAL	Phase shift.
fPhaseVelocity	LREAL	Velocity for phase shifting.
fPhaseAcc	LREAL	Acceleration for phase shifting.

18.3.3.2.3.6 MC_SWITCH_REF: Switch Structure

Variable	Type	Description
fPosition	REAL	Starting position for the output signal ON.
iMode	INT	Specify the switch mode: 0: Disabled. 1: Position based. 2: Time based.
iDirection	INT	Direction.
fParameter	REAL	Position based: ON end position. Time based: ON duration (ms).

18.3.3.2.3.7 MC_OUTMODE: Desynchronization Mode Enumeration

Variable	Value	Description
mcDecelerationStop	0	Deceleration stop.
mcConstantVelocity	1	Constant velocity before desynchronization.
mcImmediateStop	2	Immediate stop after the camming cycle.

18.3.3.2.3.8 MC_MASTER_DIRECTION: Master Axis Motion Direction Enumeration

Variable	Value	Description
CAM_NO_DIRECTION	0	No direction specified. The slave axis follows the master axis regardless of the motion direction of the master axis.
CAM_ABS_POSITIVE	1	Absolute positive direction. The slave axis follows the master axis when the master axis moves in the positive direction, and stops following if the master axis reverses the direction. The master axis must return to positive direction and reach the reversal point for the slave axis to continue following.
CAM_REL_POSITIVE	2	Relative positive direction. The slave axis follows the master axis when the master axis moves in the positive direction, and stops following if the master axis reverses the direction. The master axis must return to positive direction without reaching the reversal point for the slave axis to continue following.
CAM_ABS_NEGATIVE	3	Absolute negative direction. The slave axis follows the master axis when the master axis moves in the negative position, and stops following if the master axis reverses the direction. The master axis must return to negative direction and reach the reversal point for the slave axis to continue following.

Variable	Value	Description
CAM_REL_NEGATIVE	4	Relative negative direction. The slave axis follows the master axis when the master axis moves in the negative direction, and stops following if the master axis reverses the direction. The master axis must return to negative direction without reaching the reversal point for the slave axis to continue following.

18.3.3.2.4 Axis Group Data Object Definitions

18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure

Variable	Type	Description
iId	E_GROUP_ID	Axis group type identifier.
iNo	DWORD	Number of an axis group.
bDisableErrorLogging	BOOL	Specify whether to print logged information.
fbeFBError	ARRAY [0..5] OF 18.3.3.2.2.4 MC_FBERROR_REF: Error Record Structure	Error message.
bError	BOOL	Fault status.
iErrorID	DWORD	Fault ID.
eState	18.3.3.2.4.5 MC_AXES_GROUP_STATE: Axis Group State Enumeration	Axis group state machine.
stWorkSpace	18.3.3.2.4.3 MC_WORK_SPACE_REF: Work Space Configuration Structure	Work space configuration.
stAxisSpace	18.3.3.2.4.2 MC_AXIS_SPACE_REF: Logical Axis Configuration Structure	Logical axis configuration.
stCfg	18.3.3.2.4.4 MC_GROUP_CFG_REF: Axis Group State Structure	Axis group state.
eMoveCoord	18.3.3.2.4.7 MC_COORD_TYPE: Coordinate System Type Enumeration	Motion coordinate type.

Variable	Type	Description
ePartStatus	18.3.3.2.4.20 MC_PART_STATUS: Coordinate System Switch Status Enumeration	Coordinate system switch type.
stCmd	ARRAY [0..1] OF S_PNT	Commanded [ACS/PCS] coordinate position, velocity, and acceleration in the current cycle.
stFrontCmd	ARRAY [0..1] OF S_PNT	Commanded [ACS/PCS] coordinate position, velocity, and acceleration in the previous cycle.
stActual	ARRAY [0..1] OF S_PNT	Actual [ACS/PCS] coordinate position, velocity, and acceleration in the current cycle.
stFrontActual	ARRAY [0..1] OF S_PNT	Actual [ACS/PCS] coordinate position, velocity, and acceleration in the previous cycle.
fCmdPathVel	LREAL	Commanded path velocity of the Tool Center Point (TCP) in the Product Coordinate System (PCS).
fCmdPathAcc	LREAL	Commanded path acceleration of the TCP in the PCS.
fActualPathVel	LREAL	Actual path velocity of the TCP in the PCS.
fActualPathAcc	LREAL	Actual path acceleration of the TCP in the PCS.

18.3.3.2.4.2 MC_AXIS_SPACE_REF: Logical Axis Configuration Structure

Variable	Type	Description
iType	ARRAY [0..(GROUP_DEF.AXIS_NUM- 1)] OF E_LOGIC_AXIS_TYPE	Logical axis type.
iAxisNum	INT	Number of logical axes.
stMaxMove	ARRAY [0..(GROUP_DEF.AXIS_NUM- 1)] OF S_TRAN_PLAN	Planned maximum parameter for a logical axis.
fLowCoeff	LREAL	Scaling factor for low-velocity planning.
fDecCoeff	LREAL	Scaling factor for default planning.
fLimitCoeff	LREAL	Scaling factor for early-warning planning.

Variable	Type	Description
fTransfCoeff	LREAL	Transformation coefficient.
fNegativeLimit	ARRAY [0..(GROUP_DEF.AXIS_NUM- 1)] OF LREAL	Negative limit for a logical axis.
fPositiveLimit	ARRAY [0..(GROUP_DEF.AXIS_NUM- 1)] OF LREAL	Positive limit for a logical axis.

18.3.3.2.4.3 MC_WORK_SPACE_REF: Work Space Configuration Structure

Variable	Type	Description
iAxisType	ARRAY[0..GROUP_DEF.AXIS_NUM-1] OF E_LOGIC_AXIS_TYPE	Space axis type.
iAxisNum	INT	Number of axes in an axis group.
iLinAxisNum	INT	Number of moving axes.
iAngAxisNum	INT	Number of orientation axes.
iAuxAxisNum	INT	Number of auxiliary axes.
stBase	S_CARTESIAN	Base coordinate system for an axis group based on the world coordinate system.
stPart	S_CARTESIAN	Current Cartesian part coordinate system.
stPartSet	S_PART_SET	Part coordinate system setting.
fBaseOffset	LREAL	Base offset.
stTool	S_CARTESIAN	Tool coordinate system.
stKin	MC_KIN_REF	Kinematic parameter table.
stPartList	ARRAY[0..GV_GROUP.TOOL_NUM-1] OF S_CARTESIAN	Part coordinate system.
stToolList	ARRAY[0..GV_GROUP.TOOL_NUM-1] OF S_CARTESIAN	Tool coordinate system.
stMaxLinMove	S_TRAN_PLAN	Planned maximum parameter for a moving axis.
stMaxAngMove	S_TRAN_PLAN	Planned maximum parameter for an orientation axis.

Variable	Type	Description
fLowCoeff	LREAL	Scaling factor for low-velocity planning.
fDecCoeff	LREAL	Scaling factor for default planning.
fLimitCoeff	LREAL	Scaling factor for early-warning planning.
fTransfCoeff	LREAL	Transformation coefficient.

18.3.3.2.4.4 MC_GROUP_CFG_REF: Axis Group State Structure

Variable	Type	Description
bCheck	BOOL	Check enable switch.
eCurveType	18.3.3.2.4.8 MC_CURVE_TYPE: Curve Type Enumeration	Trajectory curve type.
fTrajectUnit	LREAL	Trajectory unit.
fOrientUnit	LREAL	Orientation unit.
iMainUnit	18.3.3.2.4.11 MC_UNIT_SPACE: Main Unit Enumeration	Main unit.
fCycleTime	LREAL	Interpolation cycle for an axis group.
aHomeSequence	ARRAY [0..GROUP_DEF.AXIS_NUM-1] OF INT	Homing sequence for an axis group.
aHomeMode	ARRAY [0..GROUP_DEF.AXIS_NUM-1] OF 18.3.3.2.4.14 MC_HOME_MODE: Homing Mode Enumeration	Homing mode for an axis group.
bInput	ARRAY [0..GV_GROUP.IO_NUM-1] OF BYTE	Input channel.
bOutput	ARRAY [0..GV_GROUP.IO_NUM-1] OF BYTE	Output channel.

18.3.3.2.4.5 MC_AXES_GROUP_STATE: Axis Group State Enumeration

Variable	Value	Description
GROUP_ERRORSTOP	0	Axis group error stop.
GROUP_DISABLED	1	Axis group disabled.
GROUP_STOPPING	2	Axis group emergency stop.
GROUP_STANDBY	3	Axis group standby.

Variable	Value	Description
GROUP_MOVING	4	Axis group moving.
GROUP_HOMING	5	Axis group homing.

18.3.3.2.4.6 MC_CIRCE_TYPE: Circular Path Type Enumeration

Variable	Value	Description
BORDER_ARC	0	Circular interpolation is performed through the border point (AuxPoint) from the current position (start point) to the end point (EndPoint).
CENTER_ARC	1	Circular interpolation is performed for circle specified by the center point (AuxPoint) from the current position (start point) to the end point (EndPoint).
RADIUS_ARC	2	Circular interpolation is performed for the circle specified by the radius (AuxPoint) from the current position (start point) to the end point (EndPoint).

18.3.3.2.4.7 MC_COORD_TYPE: Coordinate System Type Enumeration

Variable	Value	Description
ACS	0	Axes coordinate system.
PCS	1	Part coordinate system.
MCS	2	Machine coordinate system.
WCS	3	World coordinate system.

18.3.3.2.4.8 MC_CURVE_TYPE: Curve Type Enumeration

Variable	Value	Description
TRAPEZOIDAL_CURVE	0	Trapezoidal curve.
CUBIC_CURVE	1	Cubic S-curve.
SEVENTH_CURVE	2	Seventh-order S-curve.

18.3.3.2.4.9 MC_ERROR_STOP: Stop Mode Enumeration

Variable	Value	Description
T_ERROR_STOPPED	0	Error stop completed.
T_ERROR_STOPPING	1	In error stop.
T_ERROR_STOP	2	Error stop activated.

18.3.3.2.4.10 MC_TRANSITION_MODE: Transition Mode Enumeration

Variable	Value	Description
TMNone	0	Insert no transition curve.
TMStartVelocity	1	Transition with start velocity reference.
TMCornerDistance	3	Transition with corner distance reference.
TMMaxCornerDeviation	4	Transition with maximum corner deviation reference.
TMDefinedVelocity	10	Transition with commanded maximum velocity.
TMResetAbort	11	Reset to abort.
TMDecAbort	12	Decelerate to abort.

18.3.3.2.4.11 MC_UNIT_SPACE: Main Unit Enumeration

Variable	Value	Description
UNIT_TRAJECT	0	Trajectory.
UNIT_ORIENT	1	Orientation.
UNIT_PERCENT	2	Single-axis scaling factor.

18.3.3.2.4.12 MC_GROUP_STOP_MODE: Stop Mode Enumeration

Variable	Value	Description
DEC_STOP	0	Deceleration stop.
IMMED_STOP	1	Immediate stop.

18.3.3.2.4.13 MC_CORNER_TYPE: Corner Type Enumeration

Variable	Value	Description
NoneCorner	0	No corner.
CornerDistance	1	Corner distance.
MaxCornerDeviation	2	Maximum corner deviation.
CornerRadius	3	Corner radius.

18.3.3.2.4.14 MC_HOME_MODE: Home Mode Enumeration

Variable	Value	Description
MC_SERVO_HOME	0	Servo homing.
MC_PLANNING_HOME	1	PLC homing.

18.3.3.2.4.15 MC_LOAD_PATH_STATUS: Path Status Enumeration

Variable	Value	Description
PATH_ERROR	0	Path error.
PATH_READY	1	Path ready.
PATH_BUSY	2	Path busy.
PATH_ACTIVE	3	Path active.
PATH_ABORT	4	Path aborted.
PATH_DONE	5	Path done.

18.3.3.2.4.16 MC_PATH_CHOICE: Path Choice Enumeration

Variable	Value	Description
SHORT_ARC	0	Short arc.
LONG_ARC	1	Long arc.

18.3.3.2.4.17 MC_CIRC_PATHCHOICE: Circular Path Choice Enumeration

Variable	Value	Description
SHORTPART	0	Short arc.
LONGPART	1	Long arc.

18.3.3.2.4.18 MC_CIRCMODE: Circular Interpolation Mode Enumeration

Variable	Value	Description
BORDER_ARC_TYPE	0	Circular interpolation is performed through the border point (AuxPoint) from the current position (start point) to the end point (EndPoint).
CENTER_ARC_TYPE	1	Circular interpolation is performed for circle specified by the center point (AuxPoint) from the current position (start point) to the end point (EndPoint).
RADIUS_ARC_TYPE	2	Circular interpolation is performed for the circle specified by the radius (AuxPoint) from the current position (start point) to the end point (EndPoint).

18.3.3.2.4.19 MC_COORDSYSTEM: Coordinate System Enumeration

Variable	Value	Description
ACS	0	Axis coordinate system.
PCS	1	Machine coordinate system.

18.3.3.2.4.20 MC_PART_STATUS: Coordinate System Switch Status Enumeration

Variable	Value	Description
PART_SWITCH	0	Coordinate system switching.
PART_SETTING	1	Coordinate system to be set.
PART_ACTIVE	2	Active coordinate system.
PART_SWITCH_FAITH	3	Coordinate system switching failure.

18.3.3.3 Single Axis

18.3.3.3.1 Instruction List

18.3.3.3.2 MC_Power	Enable axis	MotionControl	Library version 1.1.0.0 and later	√	√	√	×

Instruction	Function	Library	Applicable Library Version	Virtual Axis	Bus-Controlled Axis	Local Pulse Axis	Bus Encoder Axis
18.3.3.3.3 MC_ReadActualPosition	Read actual position		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.4 MC_ReadAxisError	Read axis fault		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.5 MC_ReadStatus	Read axis state		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.6 MC_Reset	Reset axis fault		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.7 MC_ReadActualTorque	Read actual torque		Library version 1.1.0.0 and later	×	√	×	×
18.3.3.3.8 MC_ReadActualVelocity	Read actual velocity		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.9 MC_SetPosition	Set current position		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.10 MC_TouchProbe	Touch probe function		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.11 MC_ReadBoolParameter	Read BOOL parameter		Library version 1.1.0.0 and later	√	√	√	×

Instruction	Function	Library	Applicable Library Version	Virtual Axis	Bus-Controlled Axis	Local Pulse Axis	Bus Encoder Axis
18.3.3.3.12 MC_ReadParameter	Read REAL parameter		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.13 MC_VibrationShapping	Vibration shaping		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.14 MC_WriteBoolParameter	Write BOOL parameter		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.15 MC_WriteParameter	Write REAL parameter		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.16 MC_AbortTrigger	Abort triggered operation		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.17 MC_ReadSetPosition	Read set target position		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.18 MC_InPosition	Monitor axis deviation		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.19 MC_ChangeGearingRatio	Set gear ratio and axis type		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.20 MC_SetSoftwareLimits	Set software limits		Library version 1.1.0.0 and later	√	√	√	×

Instruction	Function	Library	Applicable Library Version	Virtual Axis	Bus-Controlled Axis	Local Pulse Axis	Bus Encoder Axis
18.3.3.3.21 MC_ReadDigitalInput	Read digital input		Library version 1.1.0.0 and later	×	√	×	×
18.3.3.3.22 MC_SetControllerMode	Set controller mode		Library version 1.1.0.0 and later	×	√	×	×
18.3.3.3.23 MC_CheckLimits	Check drive reference values against upper limits		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.24 MC_SetTorqueLimit	Set upper torque limit		Library version 1.1.0.0 and later	×	√	×	×
18.3.3.3.25 MC_GetMaxSetAccDec	Read maximum acceleration / deceleration		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.26 MC_GetMaxSetVelocity	Read maximum velocity		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.27 MC_GetTrackingError	Measure actual or maximum tracking error		Library version 1.1.0.0 and later	√	√	×	×
18.3.3.3.28 MC_Halt	Pause axis		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.29 MC_Stop	Stop axis		Library version 1.1.0.0 and later	√	√	√	×

Instruction	Function	Library	Applicable Library Version	Virtual Axis	Bus-Controlled Axis	Local Pulse Axis	Bus Encoder Axis
18.3.3.3.30 MC_Home	Drive homing		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.31 MC_MoveAbsolute	Absolute positioning		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.32 MC_MoveRelative	Relative positioning		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.33 MC_MoveModulo	Positioning in rotary axis mode		Applicable to library version 2.4.0.0 and later	×	√	×	×
18.3.3.3.34 MC_FollowPosition	Specify axis position without any checks		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.35 MC_FollowVelocity	Specify axis velocity without any checks		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.36 MC_FollowPositionVelocity	Specify axis position and velocity without any checks		Library version 2.0.0.0 and later	√	√	√	×
18.3.3.3.37 MC_MoveVelocity	Velocity planning		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.38 MC_Jog	Axis jogging		Library version 1.1.0.0 and later	√	√	√	×

Instruction	Function	Library	Applicable Library Version	Virtual Axis	Bus-Controlled Axis	Local Pulse Axis	Bus Encoder Axis
18.3.3.3.39 MC_TorqueControl	Torque control		Library version 1.1.0.0 and later	×	√	×	×
18.3.3.3.40 MC_ImmediateStop	Immediate stop		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.41 MC_MoveFeed	Interrupt feeding		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.42 MC_SDORead	Read specified SDO data		Library version 2.0.0.0 and later	×	√	×	√
18.3.3.3.43 MC_SDOWrite	Write specified SDO data to input axis		Library version 2.0.0.0 and later	×	√	×	√
18.3.3.3.44 MC_PositionFilter	Position filter		Library version 2.0.0.0 and later	√	√	√	√
18.3.3.3.45 MC_SetOverride	Set override factor		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.46 MC_ResetFollowingError	Reset following error		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.47 MC_HaltSuperImposed	Halt superimposed motion		Library version 1.1.0.0 and later	√	√	√	×

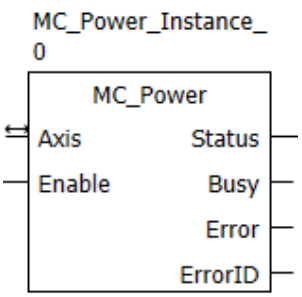
Instruction	Function	Library	Applicable Library Version	Virtual Axis	Bus-Controlled Axis	Local Pulse Axis	Bus Encoder Axis
18.3.3.3.48 MC_MoveSuperImposed	Superimposed motion		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.49 MC_MoveContinuousAbsolute	Move to a specified absolute position ending with the specified velocity		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.50 MC_MoveContinuousRelative	Move to a specified relative position ending with the specified velocity		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.51 MC_CheckAxisCommunication	Check axis communication		Library version 1.1.0.0 and later	×	√	×	×
18.3.3.3.52 MC_Inch	Axis inching		Library version 1.1.0.0 and later	√	√	√	×
18.3.3.3.53 MC_Homing	PLC homing		Library version 1.1.0.0 and later	×	√	×	×
18.3.3.3.54 MC_HomingWithParameter	PLC homing with parameters		Library version 1.1.0.0 and later	×	√	×	×
18.3.3.3.55 MC_SyncMoveVelocityCSV	Velocity control with adjustable pulse width		Library version 1.1.0.0 and later	×	√	√	×

Instruction	Function	Library	Applicable Library Version	Virtual Axis	Bus-Controlled Axis	Local Pulse Axis	Bus Encoder Axis
18.3.3.3.56 MC_BacklashCompensate	Backlash compensation		Library version 1.1.0.0 and later	×	√	√	×
18.3.3.3.57 MC_LeadScrewCompensate	Leadscrew pitch error compensation		Library version 1.1.0.0 and later	×	√	√	×
18.3.3.3.58 MC_MoveAdditive	Add a distance to the last specified target position		Library version 2.3.0.0 and later	√	√	√	×
18.3.3.3.59 MC_GetGearingRatio	Get the gear ratio, rotation cycle, and motion type		Applicable to library version 2.4.0.0 and later	√	√	√	×

18.3.3.3.2 MC_Power

This instruction enables an axis.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_Power	Enable axis		MC_Power_Instance_0(Axis :=, Enable :=, Status =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled; FALSE: Disabled

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Status	Ready for operation	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is enabled.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs in the function block.

- **Function**

  Bus-controlled axis:

  - When the input parameter Enable is set to TRUE, the axis is enabled, and axis control can be performed when the axis is ready for operation. When the input parameter Enable is set to FALSE, the ready status is cleared. You cannot execute any motion control instructions when the axis is disabled. However, you can execute non-motion control instructions such as MC_Power, MC_Reset, and MC_SetPosition.
  - If a power failure occurs during the execution, the axis state will be ErrorStop, and the function block will output a fault code.
  - When the axis state is ErrorStop, it cannot be switched to StandStill if this instruction is re-executed. You must execute the MC_Reset instruction to clear the error message.
  - This instruction can be used for control axes. If it is executed on an encoder axis, the function block will report a fault.
  - Multiple identical instances can be triggered simultaneously on the same axis, and the enable status depends on the input parameter Enable of the last triggered function block in the user program.

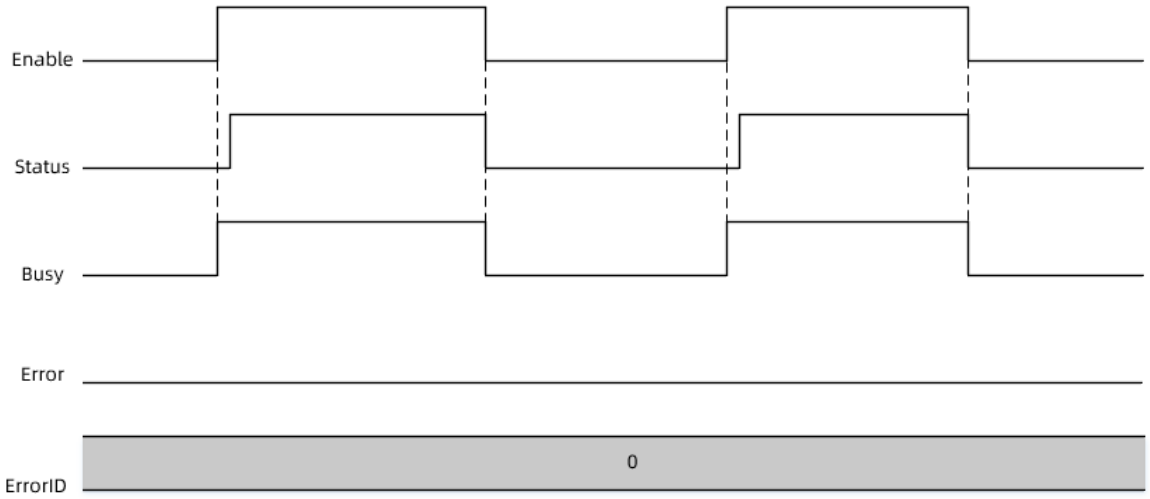
Local pulse axis:

- This instruction can be used for local pulse axes.

- **Timing diagram**

  When the input parameter Enable is set to TRUE, the output parameter Busy shows the value TRUE; when the axis is ready for operation, the output parameter Status shows the value TRUE.

  When the input parameter Enable is set to FALSE, the output parameter Busy shows the value FALSE; the ready status is cleared; the output parameter Status shows the value FALSE.



- **Re-execution of motion control instructions**

  You cannot re-execute motion control instructions with an input parameter Enable.

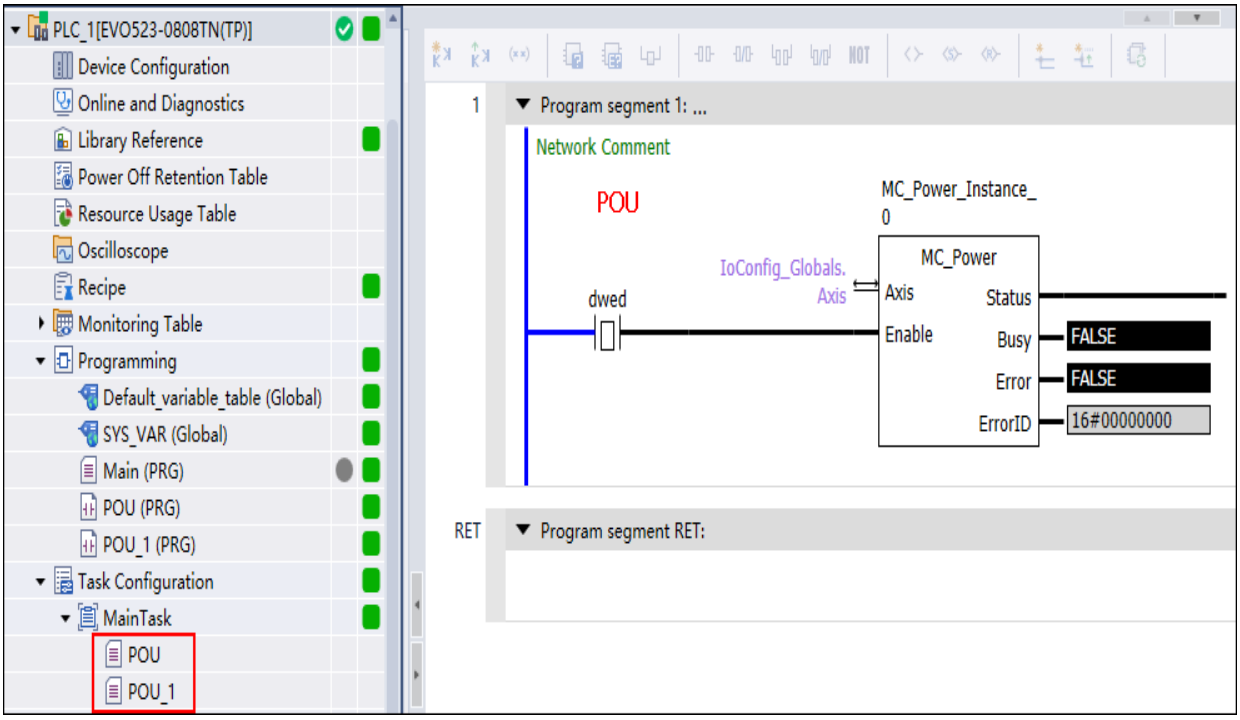
- **Multiple execution of motion control instructions**

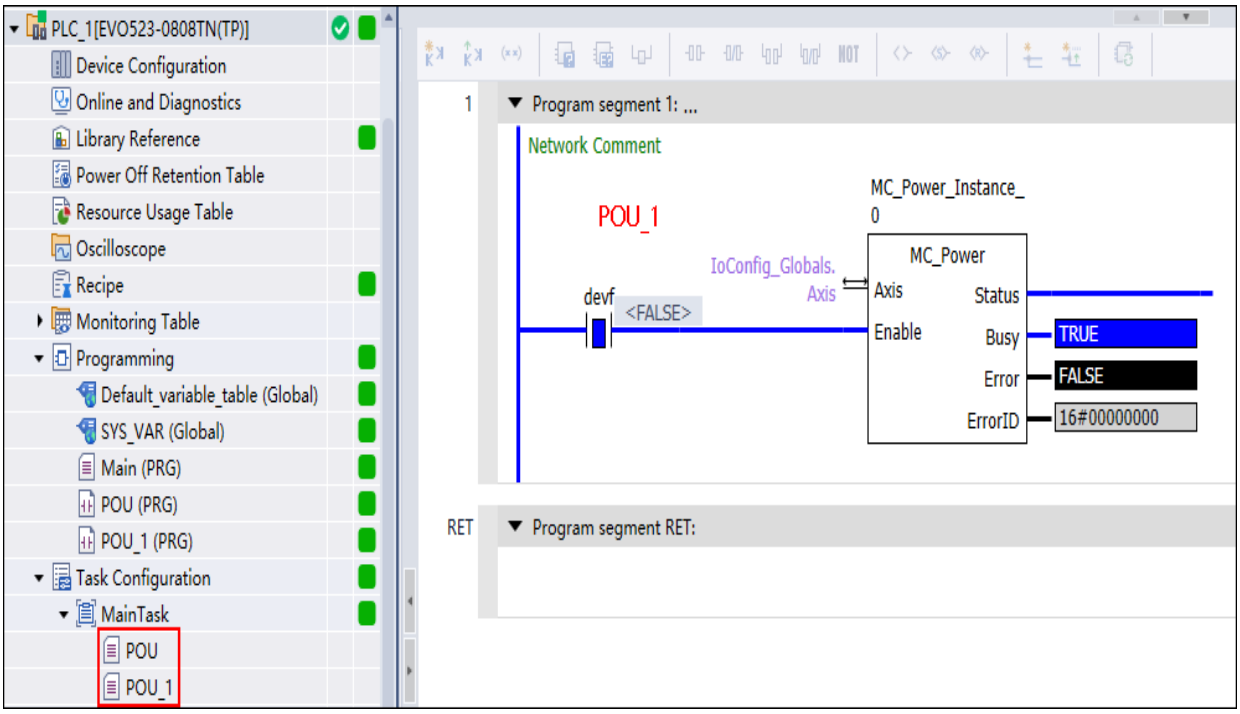
  When multiple MC_Power instructions are executed in a cycle, the last MC_Power instruction takes priority.

- **Faults**

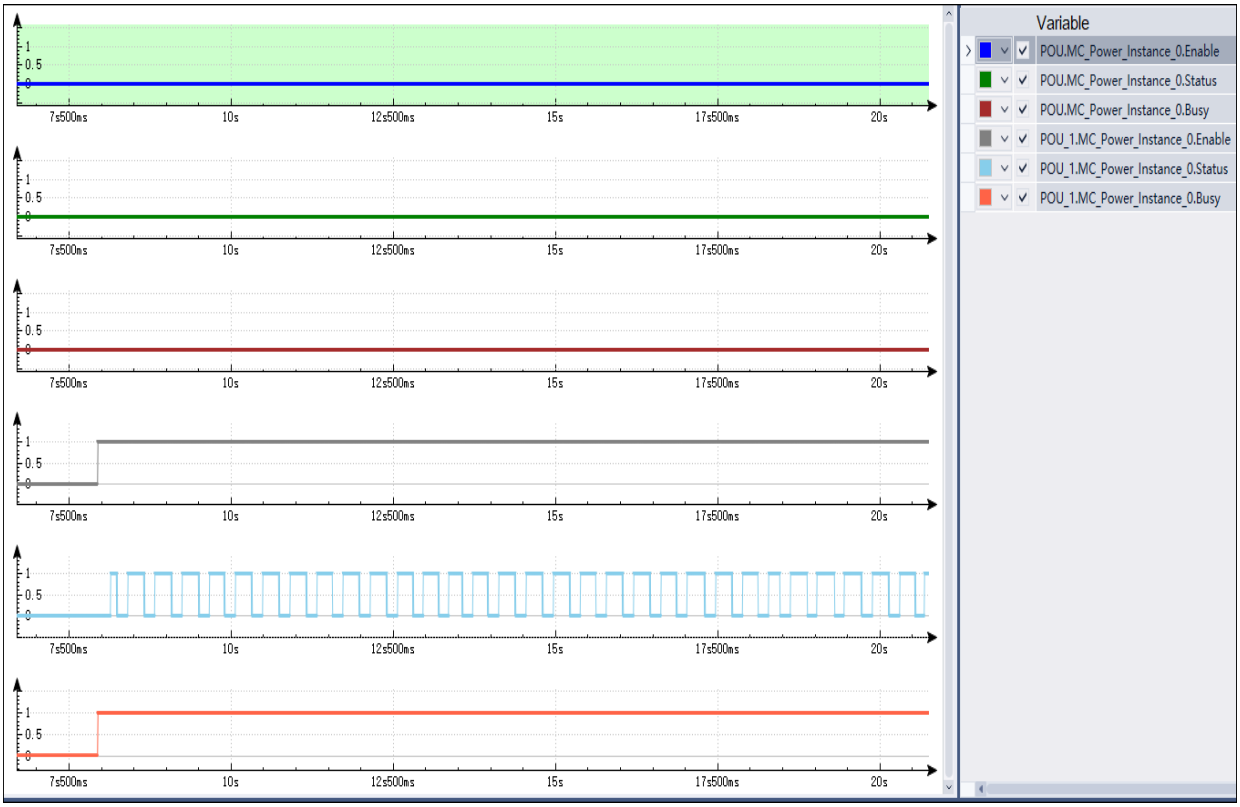
  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

- **Precautions**
  - In the case where the axis-bound servo cannot be enabled:
    1. Check if the variable bound to the address in the program is duplicated with the address in the I/O mapping area of the PDO.
    2. Check if the configured PDOs are valid and within limits.
  - When the input parameter Enable is set to TRUE, the output parameter Status will not show the value TRUE until the servo operation is complete. Perform any axis operations only after the output parameter Status shows the value TRUE.
  - In field applications, it is not recommended to place the MC_Power instruction in multiple POUs:





If the MC_Power instruction in POU1 is triggered but the one in POU is not triggered, the axis may be disabled during the program execution.



18.3.3.3.3 MC_ReadActualPosition

This instruction reads the actual position of an axis.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_ReadActualPosition	Read actual axis position	MC_ReadActualPosition_Instance_0 MC_ReadActualPosition Axis Enable Valid Busy Error ErrorID Position Turns	MC_ReadActualPosition_Instance_0(Axis :=, Enable :=, Valid =>, Busy =>, Error =>, ErrorID =>, Position =>, Turns =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled. FALSE: Disabled.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Valid output	BOOL	TRUE, FALSE	FALSE	TRUE when there is a valid output from the function block.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs in the function block.
Position	Axis position	LREAL	-	0	Return to the axis position [user unit].
Turns	Number of turns	DINT	-	0	Number of turns in rotary axis mode.

- **Function**

  Bus-controlled axis:

  - This instruction is a function block controlled by the Enable signal level. When the input parameter Enable is set to TRUE, the instruction reads the actual position of an axis. This instruction can be executed repeatedly without affecting each other. Note: Multiple different instantiated function blocks

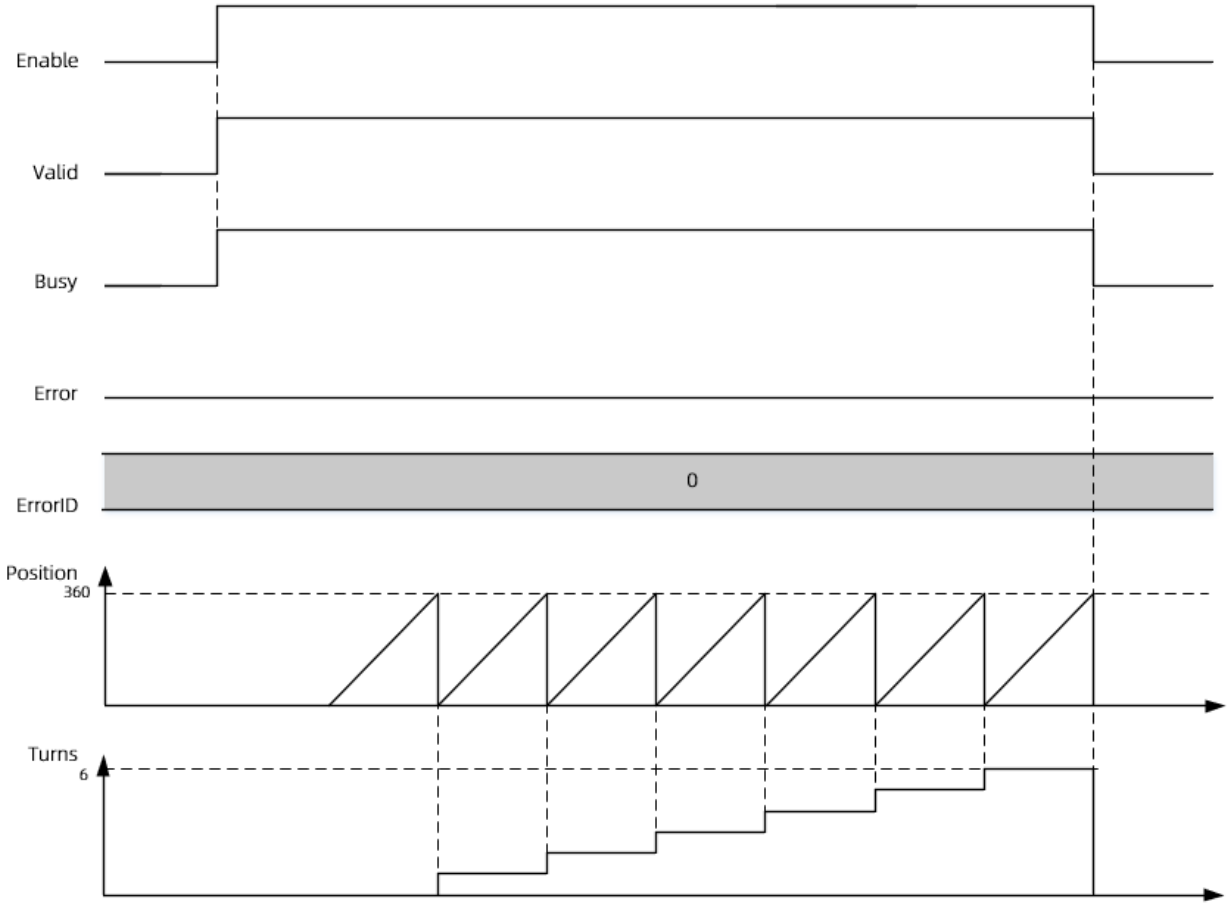
of this instruction can be executed simultaneously, instead of multiple identical instantiated function blocks.

- When the axis is in abnormal communication, this instruction issues a fault message with a fault code. After normal communication is recovered, the fault will be automatically cleared.

Local pulse axis:

- This instruction can be used for local pulse axes.

- **Timing diagram**
  - When the input parameter Enable is set to TRUE, the output parameter Busy shows the value TRUE.
  - When the output parameter Valid is set to TRUE, it indicates that the output parameter Position is a valid value.
  - In rotary axis mode, the output parameter Turns indicates the number of turns.



- **Re-execution of motion control instructions**

  You cannot re-execute motion control instructions with an input parameter Enable.

- **Faults**

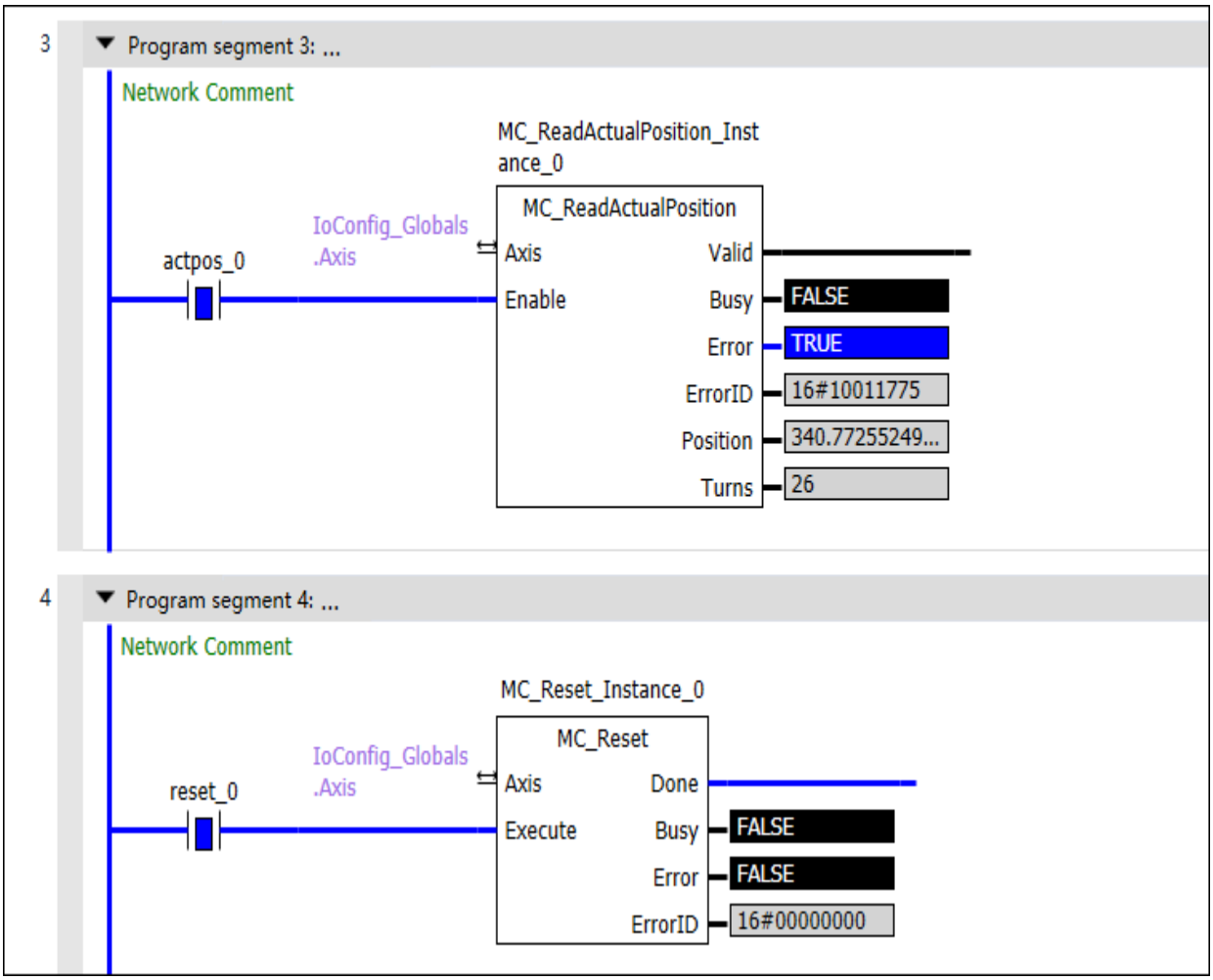
  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

- **Precautions**

  Similarly, other instructions reading related motion parameters also issue fault messages due to abnormal communication.

Although these instructions are executed when the Enable is set to TRUE, if a fault message is still issued after the axis fault is cleared, reset the Enable and then re-trigger it to re-execute the instructions.

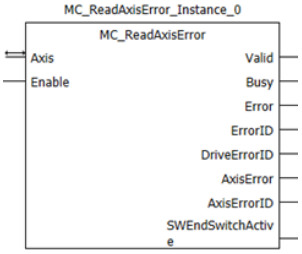
The instruction MC_Power automatically clears faults without the above operations.



18.3.3.3.4 MC_ReadAxisError

This instruction reads general faults (associated with axis, drive, communication, etc.).

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_ReadAxisError	Read axis fault		MC_ReadAxisError_Instance_0(Axis :=, Enable :=, Valid =>, Busy =>, Error =>, ErrorID =>, DriveErrorID =>, AxisError => , AxisErrorID =>, SWEndSwitchActive =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled. FALSE: Disabled.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Valid output	BOOL	TRUE, FALSE	FALSE	TRUE when there is a valid output from the function block.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs in the function block.
DriveErrorID	Drive fault code	DWORD	-	0	A drive fault code is displayed.
AxisError	Axis fault flag	BOOL	TRUE, FALSE	FALSE	TRUE when an axis fault occurs.
AxisErrorID	Axis fault code	DWORD	-	0	An axis fault code is returned.
SWEndSwitchActive	Software limit switch active	BOOL	TRUE, FALSE	FALSE	Check the state of the software limit switch. TRUE: The position is outside the software limit. FALSE: The position is within the software limit.

- **Function**

  Bus-controlled axis:

  - This instruction is a function block controlled by the Enable signal level. When the input parameter Enable is set to TRUE, it reads the fault code from the drive (the value of PDO object 603F) or from the axis (range 10#6000-10#6018)(see 18.3.3.8 Fault Codesfor details). The fault codes from other function blocks cannot be read by this instruction and must be obtained by monitoring the axis member Axis.fbeFBError[0..14] (18.3.3.2.2.4 MC_FBERROR_REF: Fault Record Structure).
  - Multiple instances of this instruction can be used simultaneously without interfering with each other. Note: Multiple different instantiated function blocks of this instruction can be executed simultaneously, instead of multiple identical instantiated function blocks.
  - When the axis is in abnormal communication, this instruction issues a fault message with a fault code. After normal communication is recovered, the fault will be automatically cleared.

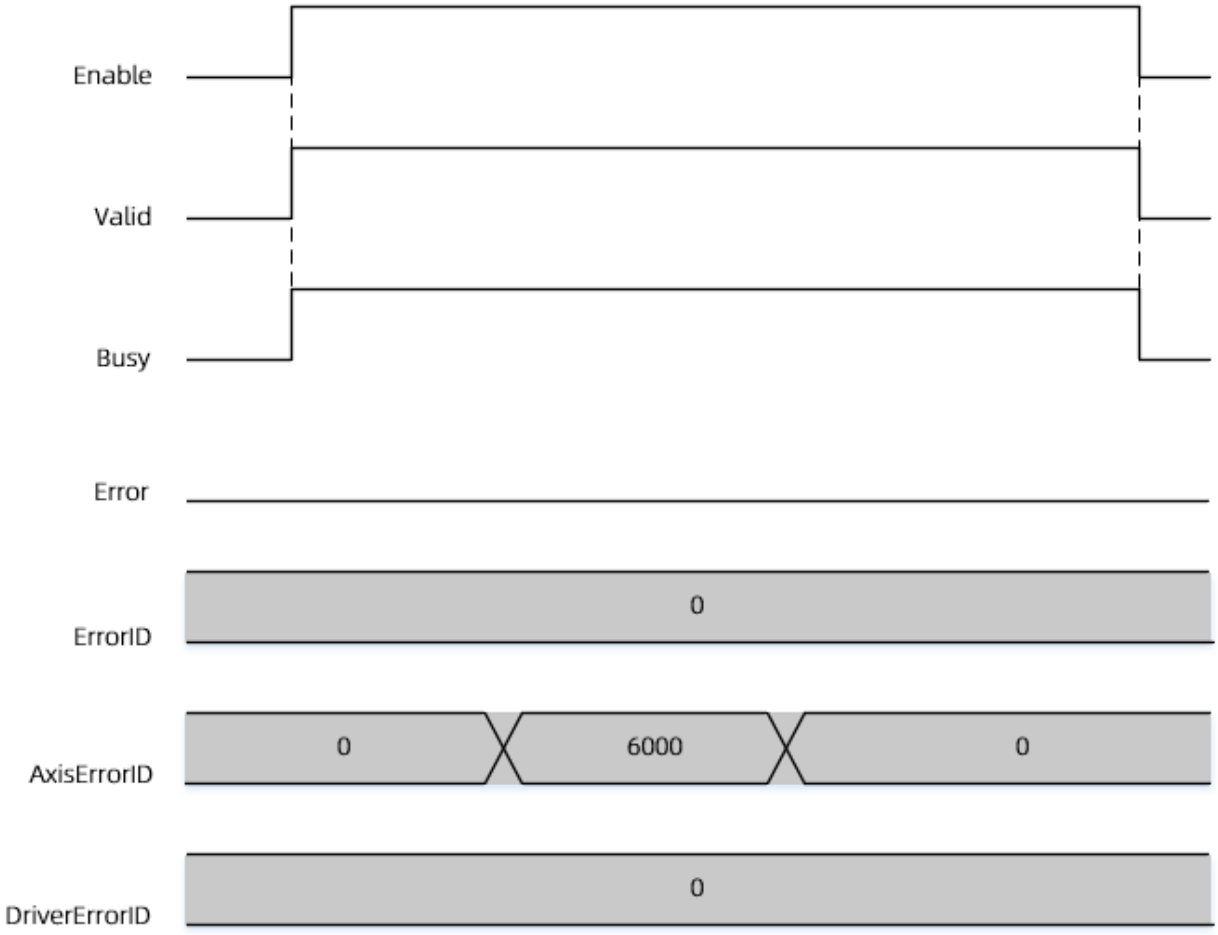
  Local pulse axis:

  - This instruction can be used for local pulse axes.

- **Timing diagram**

  When the input parameter Enable is set to TRUE, the output parameter Busy shows the value TRUE. When the output parameter Valid is set to TRUE, it indicates that the output parameters are valid.

  If an axis fault or drive fault occurs during the instruction execution, the axis fault code or drive fault code will be output to the AxisErrorID or DriverErrorID.



- **Re-execution of motion control instructions**

  You cannot re-execute motion control instructions with an input parameter Enable.

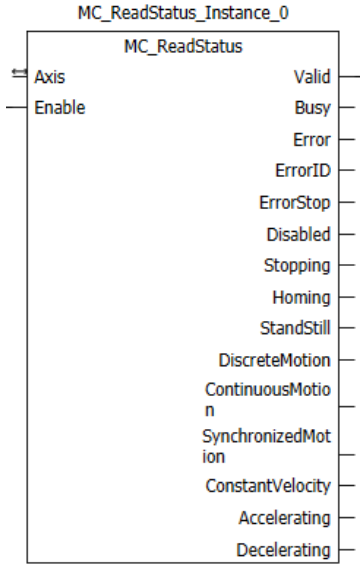
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.5 MC_ReadStatus

This instruction reads the axis state.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_ReadStatus	Read axis state		MC_ReadStatus_Instance_0(Axis :=, Enable :=, Valid =>, Busy =>, Error =>, ErrorID =>, ErrorStop =>, Disabled =>, Stopping =>, Homing =>, StandStill =>, DiscreteMotion =>, ContinuousMotion =>, SynchronizedMotion =>, ConstantVelocity =>, Accelerating =>, Decelerating =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled. FALSE: Disabled.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Valid output	BOOL	TRUE, FALSE	FALSE	TRUE when there is a valid output from the function block.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs in the function block.
ErrorStop	Error stop	BOOL	TRUE, FALSE	FALSE	TRUE: The axis stops due to a fault.
Disabled	Disabled	BOOL	TRUE, FALSE	FALSE	TRUE: The axis is disabled.
Stopping	Stopping	BOOL	TRUE, FALSE	FALSE	TRUE: The axis is decelerating to stop.
Homing	Homing	BOOL	TRUE, FALSE	FALSE	TRUE: The axis is being homed.
StandStill	Standstill	BOOL	TRUE, FALSE	FALSE	TRUE: The axis is standstill.
DiscreteMotion	Discrete motion	BOOL	TRUE, FALSE	FALSE	TRUE: The axis is in discrete motion.
ContinuousMotion	Continuous motion	BOOL	TRUE, FALSE	FALSE	TRUE: The axis is in continuous motion.
SynchronizedMotion	Synchronized motion	BOOL	TRUE, FALSE	FALSE	TRUE: The axis is in synchronized motion.
ConstantVelocity	Constant velocity	BOOL	TRUE, FALSE	FALSE	TRUE: The axis moves at constant velocity.
Accelerating	Accelerating	BOOL	TRUE, FALSE	FALSE	TRUE: The axis is accelerating.
Decelerating	Decelerating	BOOL	TRUE, FALSE	FALSE	TRUE: The axis is decelerating.

- **Function**

Bus-controlled axis:

- This instruction is a function block controlled by the Enable signal level. When the input parameter Enable is set to TRUE, the instruction reads the axis state. This instruction can be executed repeatedly without affecting each other. Note: Multiple different instantiated function blocks of this instruction can be executed simultaneously, instead of multiple identical instantiated function blocks.
- When the axis is in abnormal communication, this instruction issues a fault message with a fault code.

Local pulse axis:

- This instruction can be used for local pulse axes.

- **Timing diagram**

  The MC_ReadStatus instruction reads the axis state. It is enabled through the Enable signal level. This instruction can be executed repeatedly without affecting each other.

  When the input parameter Enable is set to TRUE, the output parameter Busy shows the value TRUE. When the output parameter Valid is set to TRUE, it indicates that the other output parameters are valid.

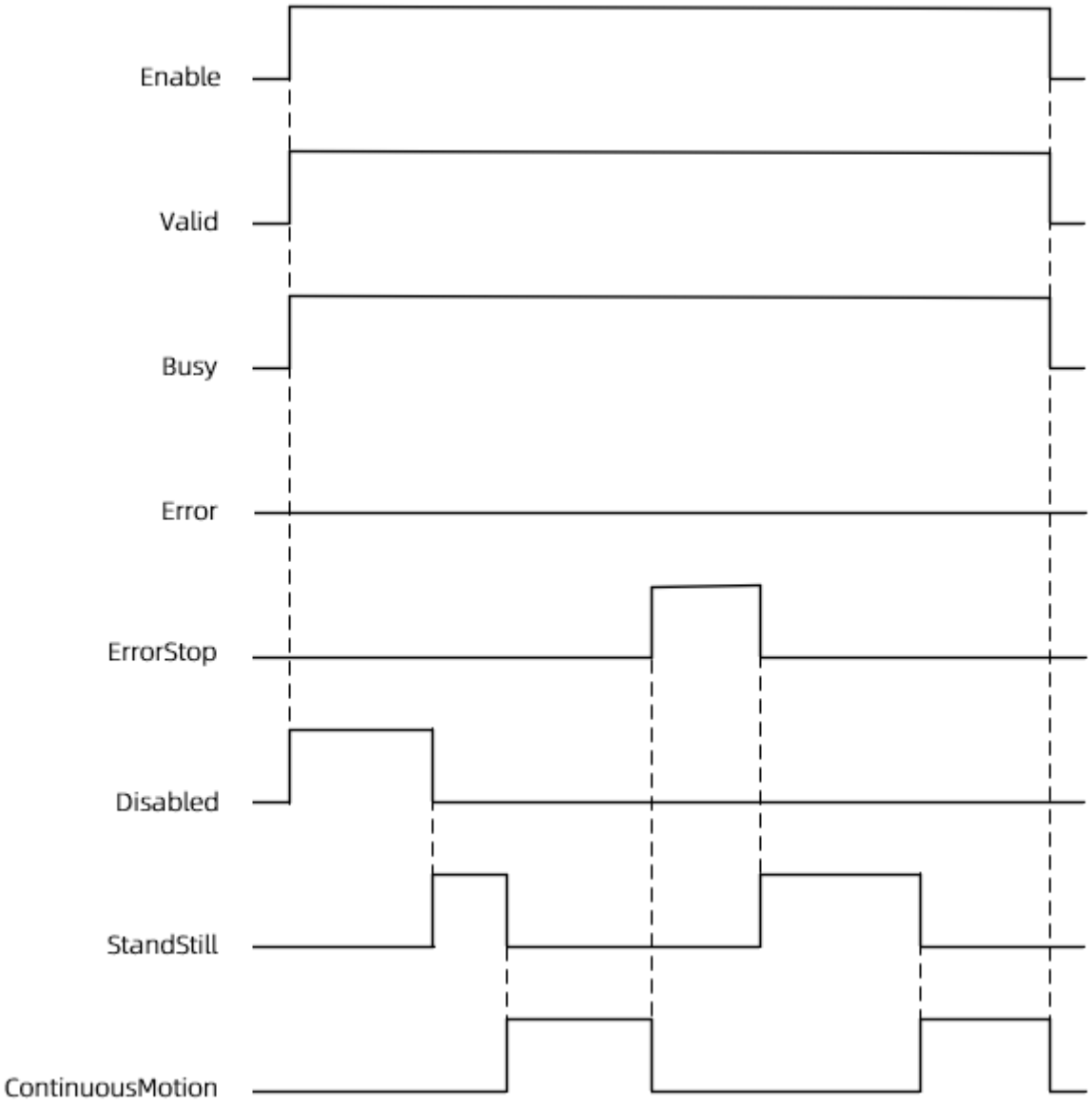
  When the axis is disabled, the axis state machine is in the Disabled state.

  When both MC_Power.Enable and MC_Power.Status are set to TRUE, the axis state machine is in the StandStill state.

  When the MC_MoveVelocity instruction is executed, the axis state machine is in the ContinuousMotion state.

  When the MC_ImmediateStop instruction is executed, the axis state machine is in the ErrorStop state.

  When the MC_Reset is executed and the MC_Power.Status is set to TRUE, the axis state machine is in the StandStill state.



- **Re-execution of motion control instructions**

  You cannot re-execute motion control instructions with an input parameter Enable.

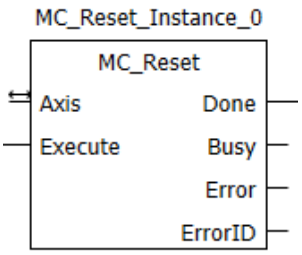
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.6 MC_Reset

This instruction resets axis faults.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_Reset	Reset axis faults		MC_Reset_Instance_0(Axis :=, Execute := , Done =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs in the function block.

- **Function**

  Bus-controlled axis:

  - This instruction resets communication faults, some servo faults (not associated with restart upon power failure), and axis state faults.
  - This instruction starts fault reset for the specified axis on a rising edge of the input variable Execute. Fault reset is performed regardless of the current axis state. If the reset is successful, the output parameter Done is set to TRUE; otherwise, the output parameter Error is set to TRUE.
  - This instruction only clears faults that occur on a rising edge of the input parameter Execute, but does not clear faults that occur during fault clearing.
  - If there is a drive fault that cannot be cleared by this instruction, restart the servo drive. If the fault is still not cleared, see the guide for the corresponding servo model for remedies.
  - If this instruction is triggered without any axis fault, the output parameter Done is directly set to TRUE.
  - This instruction resets axis faults or FBError except for faults that cannot be cleared as indicated in the servo related documentation.
  - After the instruction has been executed successfully, the axis state switches to Standstill if the current drive is enabled; the axis state switches to Disabled if the drive is not enabled.

  Local pulse axis:

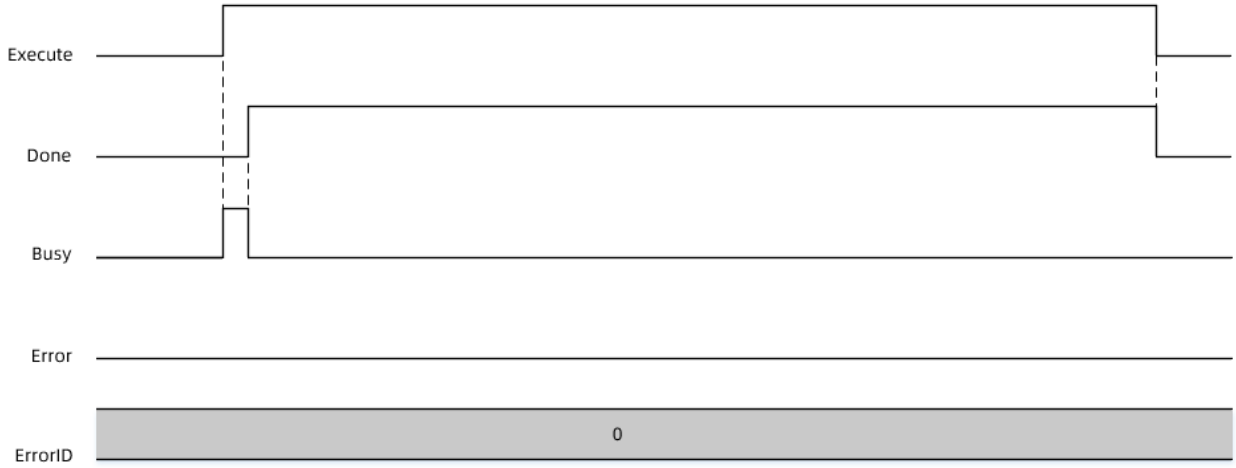
  - This instruction can be used for local pulse axes.

- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates that the instruction has been executed successfully, and the parameter Busy indicates that the current function block is being executed.

  After this instruction is called, the output parameter Busy becomes TRUE on a rising edge of Execute. After successful reset, the output parameter Done is set to TRUE.

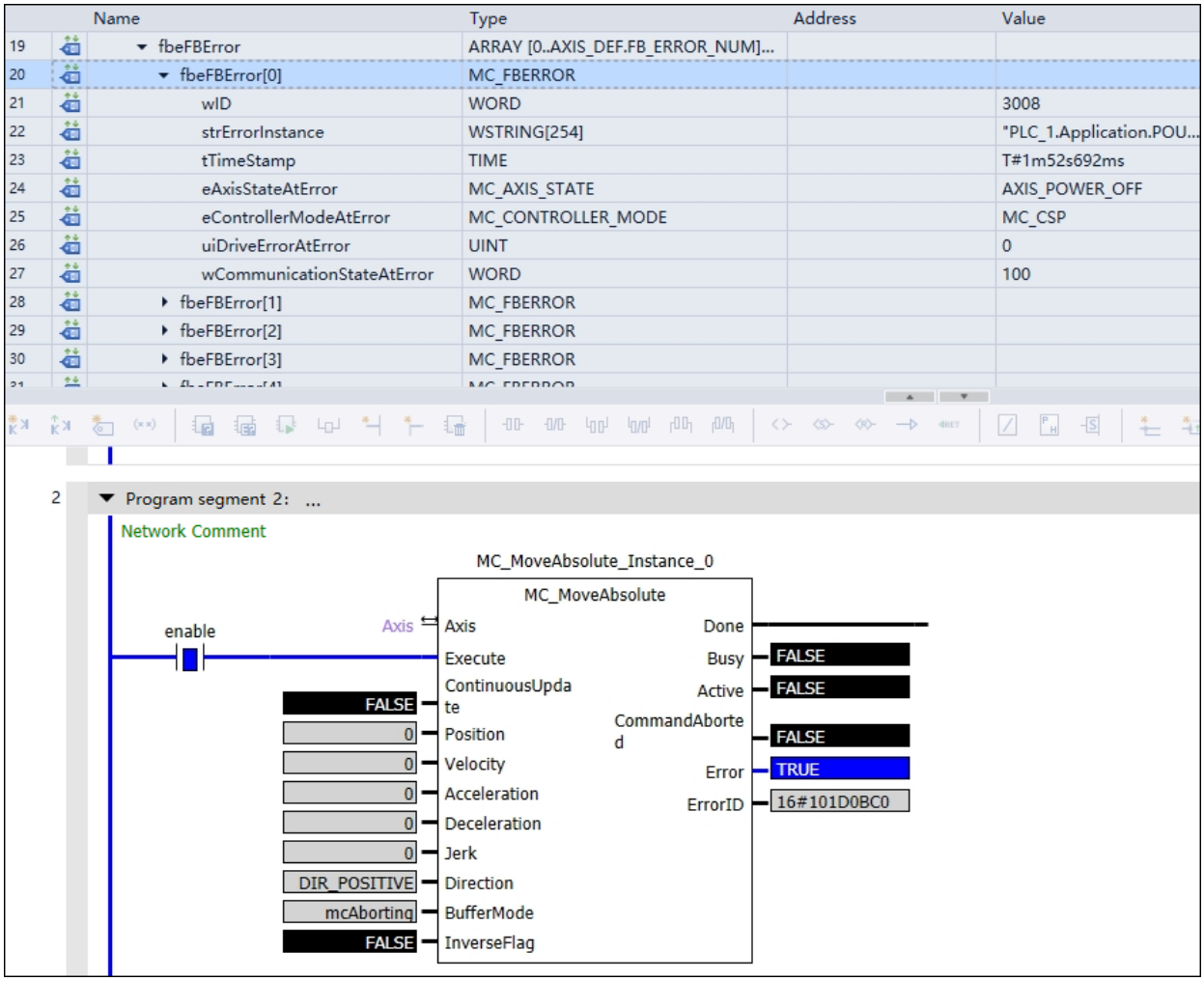
  The following is a timing diagram for calling the MC_Reset instruction to reset the axis fault.



- **Faults**

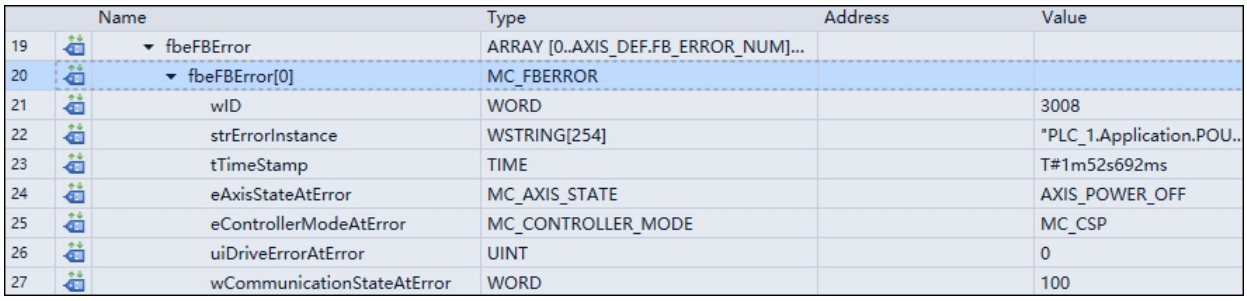
  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

- **Precautions**
  - Fault clearing with this instruction may require more than one control cycle.
  - Execute this instruction only after you confirm that the axes have stopped completely.
- **Example**
  - Add a new watch table to monitor the variables bDisableErrorLogging and fbeFBError.
  - The parameter bDisableErrorLogging is set to FALSE by default. When this parameter is set to TRUE, any instruction faults are not logged in FBError.
  - You can find up to 15 faults by expanding the parameter FBError. No more faults or repeated faults are logged.

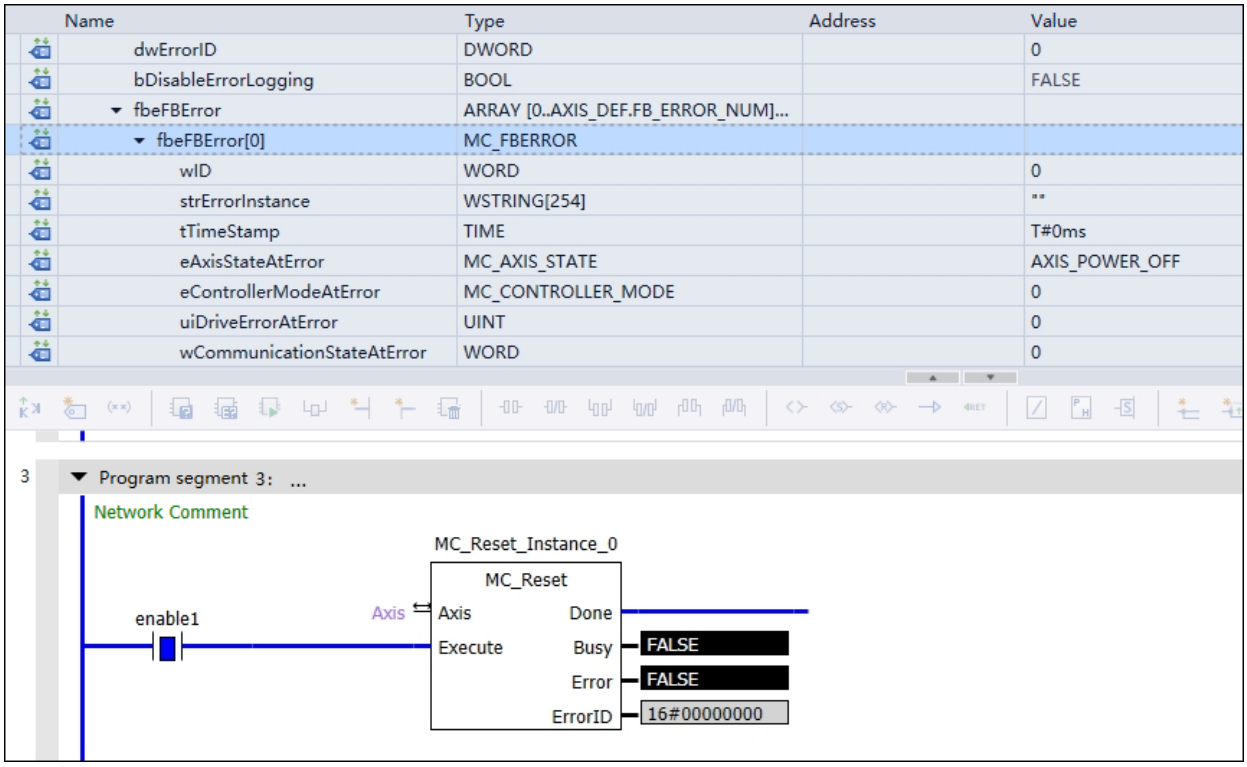


Any faults associated with the MC_MoveAbsolute instruction will be logged in fbeFBError[0], where strErrorInstance indicates the location of the fault source, and tTimeStamp records the system time of the fault occurrence.

wID "101D" corresponds to the fault code “101D0BC0” returned by the instruction, where the last four digits “0BC0” indicate the fault cause. To know more information about faults in the FBError, see the Fault Codes.



When the MC_Reset instruction is executed, the axis faults and all faults in FBError will be cleared.



18.3.3.3.7 MC_ReadActualTorque

This instruction reads the value of the current actual torque of an axis.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_ReadActualTorque	Read actual torque	MC_ReadActualTorque_Instance_0 MC_ReadActualTorque Axis Enable Valid Busy Error ErrorID Torque	MC_ReadActualTorque_Instance_0(Axis :=, Enable :=, Valid =>, Busy =>, Error =>, ErrorID =>, Torque =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled. FALSE: Disabled.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Valid output	BOOL	TRUE, FALSE	FALSE	TRUE when there is a valid output from the function block.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs in the function block.
Torque	Torque	LREAL	-	-	A torque value (unit: 0.1%) is returned.

- **Function**

  Bus-controlled axis:

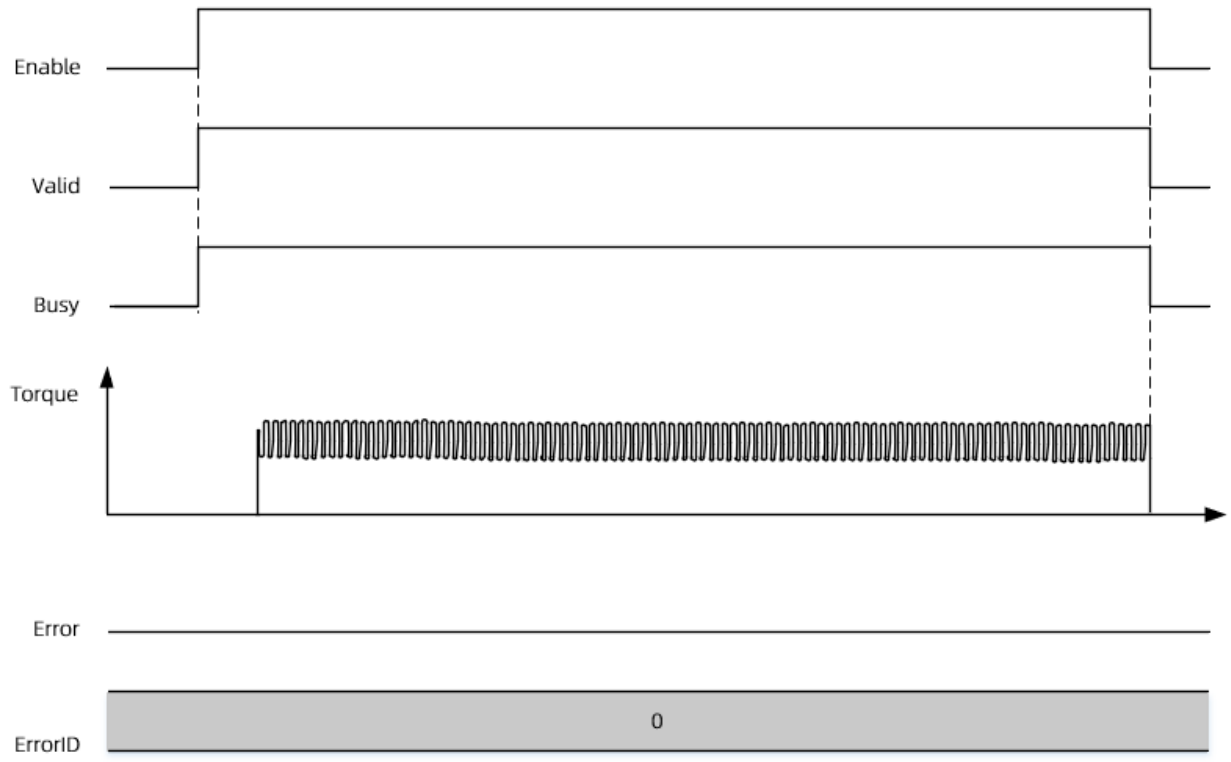
  - This instruction is a function block controlled by the Enable signal level. When the input parameter Enable is set to TRUE, the instruction reads the current torque value in the drive. This instruction can be executed repeatedly without affecting each other. Resetting the Enable value will invalidate the output data. Note: Multiple different instantiated function blocks of this instruction can be executed simultaneously, instead of multiple identical instantiated function blocks.
  - When the parameter Valid is set to TRUE, the returned torque value is a valid signed number.
  - When the axis is in abnormal communication, this instruction issues a fault message with a fault code.

  Local pulse axis:

  - This instruction cannot be used for local pulse axes.

- **Timing diagram**

  When the input parameter Enable is set to TRUE, the output parameter Busy shows the value TRUE. When the output parameter Valid is set to TRUE, it indicates that the output parameter Torque is valid.



- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.8 MC_ReadActualVelocity

This instruction reads the actual velocity of an axis.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_ReadActualVelocity	Read actual velocity	MC_ReadActualVelocity_Instance_0 MC_ReadActualVelocity Axis Enable Valid Busy Error ErrorID Velocity	MC_ReadActualVelocity_Instance_0(Axis :=, Enable :=, Valid =>, Busy =>, Error =>, ErrorID =>, Velocity =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled. FALSE: Disabled.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Valid output	BOOL	TRUE, FALSE	FALSE	TRUE when there is a valid output from the function block.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs in the function block.
Velocity	Velocity	LREAL	-	0	Obtained velocity value [u/s] of the axis.

- **Function**

  Bus-controlled axis:

  - This instruction is a function block controlled by the Enable signal level. When the input parameter Enable is set to TRUE, the instruction reads the current velocity of the specified axis. This instruction can be executed repeatedly. Resetting the Enable value will invalidate the output data. Note: Multiple different instantiated function blocks of this instruction can be executed simultaneously, instead of multiple identical instantiated function blocks.
  - When the axis is in abnormal communication, this instruction issues a fault message with a fault code.

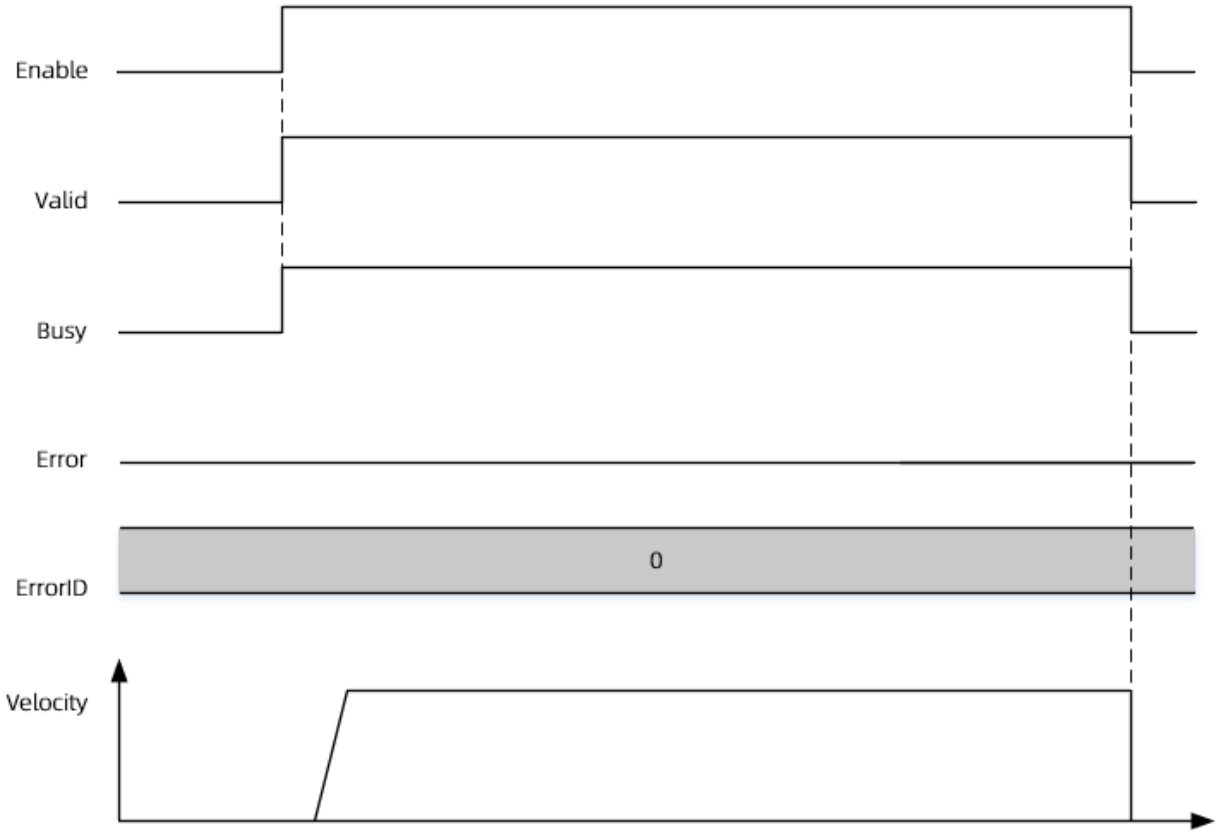
Local pulse axis:

◦ This instruction can be used for local pulse axes.

- **Timing diagram**

  When the input parameter Enable is set to TRUE, the output parameter Busy shows the value TRUE, indicating the current function block is being executed.

  When the output parameter Valid is set to TRUE, it indicates that the output parameter Velocity is valid.



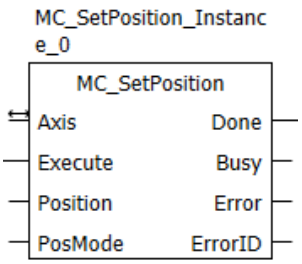
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.9 MC_SetPosition

This instruction sets the current position of an axis.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_SetPosition	Set current position		MC_SetPosition_Instance _0(Axis :=, Execute :=, Position :=, PosMode :=, Done =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Position	Axis position	LREAL	-	0	Set the target position [u] of the axis. Note: In rotary axis mode, there are two modes for the target position: 1. Absolute position: This parameter value is processed as the remainder of the specified Position value divided by the rotation cycle. 2. Relative position: This parameter value is processed as the current position + the remainder of the specified Position value divided by the rotation cycle, ignoring the multi-turn displacement.
PosMode	Position mode	BOOL	TRUE, FALSE	FALSE	TRUE: Relative position. FALSE: Absolute position.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus-controlled axis:

  - This instruction sets the position parameter of an axis and performs an offset operation on the coordinate system only without causing any displacement.
  - This instruction specifies the target position in absolute coordinates. You can modify the position when this instruction is disabled. You can specify the position with this instruction in CSP control mode only.
  - When this instruction is executed during the execution of the MC_MoveAbsolute instruction, the final actual position does not change with the change in the position reference.
  - When this instruction is executed during the execution of the MC_MoveRelative instruction, the final actual position changes with the change in the position reference.
  - When the axis state is AXIS_HOMING, this instruction cannot be triggered. During execution of a synchronization instruction, this instruction can be enabled on both the master axis and slave axis in synchronized motion. If this instruction is enabled on the master axis, the signal source "MasterValueSource" of the synchronization instruction must be set to "mcLatestCommand"; otherwise, the slave axis will overspeed.
  - In rotary axis mode, if you specify the parameter Position to a value greater than the rotation cycle, the parameter value will be processed as the remainder of the specified Position value divided by the rotation cycle.

  Local pulse axis:

  - This instruction can be used for local pulse axes.

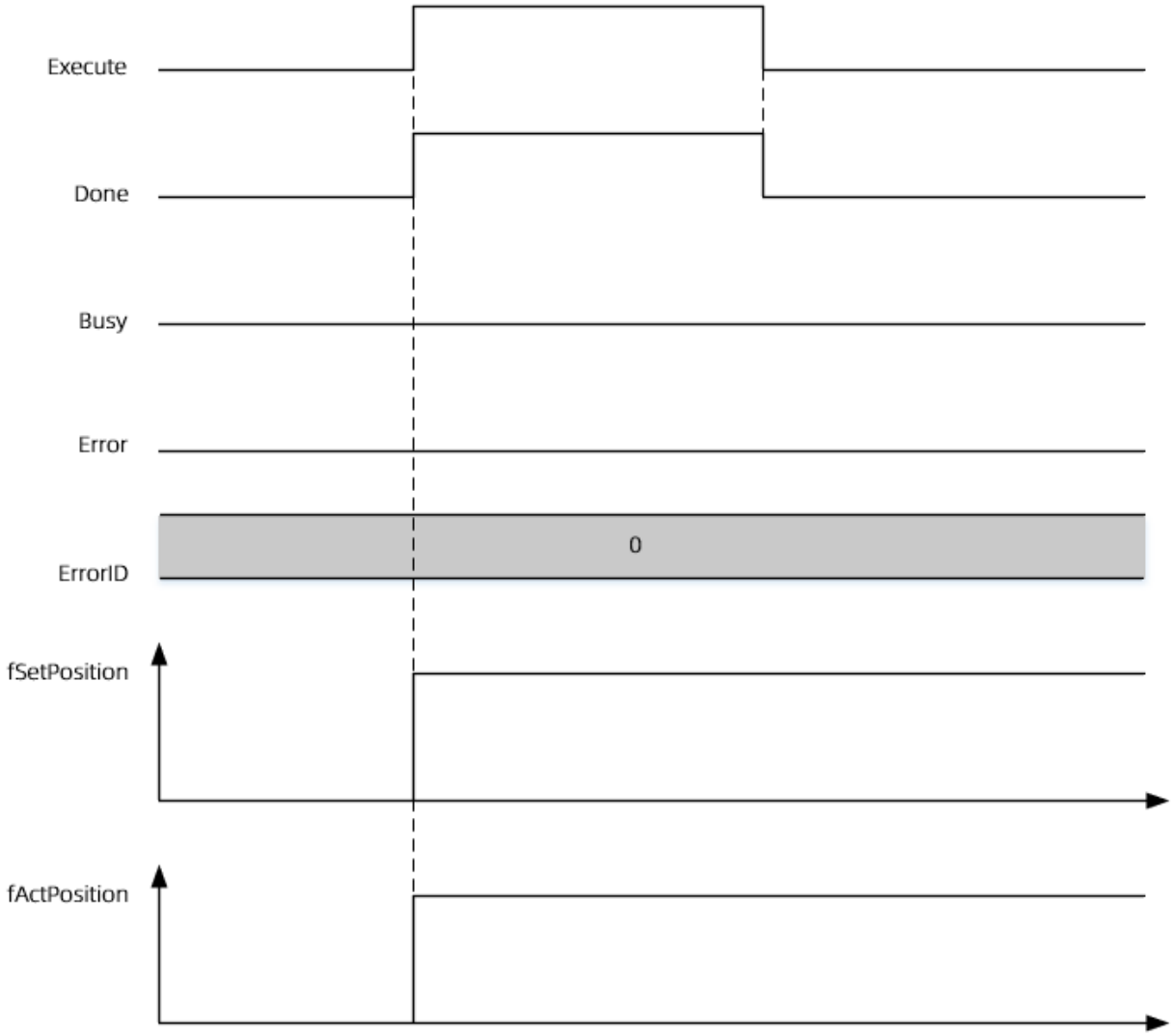
- **Timing diagram**

  The change in the current position begins on a rising edge of the input parameter Execute.

  When the input parameter Execute is set to TRUE, the output parameter Busy shows the value TRUE, indicating the current function block is being executed. After the current position changes, the output parameter Done shows the value TRUE, indicating that the instruction is done.

  When the input parameter Execute is set to FALSE, the output parameter Done shows the value FALSE.

The axis structure variable fSetPosition is the set position (unit) calculated in this task cycle, and the axis structure variable fActPosition is the actual position (unit).



- **Re-execution of motion control instructions**

  This instruction cannot be re-executed.

- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

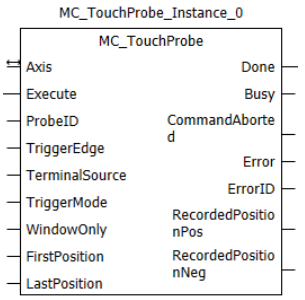
- **Precautions**

  When using the MC_SetPosition instruction, do not set an absolute value of the parameter Position greater than 10^17; otherwise, the servo drive issues a fault message.

18.3.3.3.10 MC_TouchProbe

This instruction records an axis position when a trigger signal occurs.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_TouchProbe	Enable touch probe		MC_TouchProbe_Instance_0(Axis :=, Execute :=, ProbeID :=, TriggerEdge :=, TerminalSource :=, TriggerMode :=, WindowOnly :=, FirstPosition :=, LastPosition :=, Done =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>, RecordedPositionPos =>, RecordedPositionNeg =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
ProbeID	Touch probe ID	INT	0, 1	0	0: Touch probe 1. 1: Touch probe 2.
TriggerEdge	Trigger edge	INT	0, 1, 2	0	0: Rising edge. 1: Falling edge. 2: Rising edge and falling edge.
TerminalSource	Touch probe signal source	INT	0, 1	0	0: DI terminal. 1: Encoder Z-phase signal.
TriggerMode	Trigger mode	INT	0, 1, 2	0	0: Single trigger. 1: High-performance continuous trigger. 2: Standard continuous trigger (for bus-controlled axes only).
WindowOnly	Specify whether to enable the window to detect touch probe signals	BOOL	TRUE, FALSE	FALSE	FALSE: Disabled. Touch probe signals are detected at any positions. TRUE: Enabled. Touch probe signals are detected only when the current position is within the window.
FirstPosition	First Position	LREAL	-	0	Start position [u] from where the touch probe signals are detected.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
LastPosition	Last Position	LREAL	-	0	Stop position [u] of the window.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.
RecordedPositionPos	Rising edge position	LREAL	-	0	The touch probe signals are detected on a rising edge.
RecordedPositionNeg	Falling edge position	LREAL	-	0	The touch probe signals are detected on a falling edge.

- **Function**

  Bus-controlled axis:

  - This instruction records an axis position when an external signal is input or when the motor Z-phase signal is triggered.
  - Specify the probe signal trigger edge of rising edge, falling edge, or both rising edge and falling edge through the input parameter TriggerEdge.
    - When the TriggerMode is set to "Single trigger" and the TriggerEdge is set to "Rising edge (Falling edge)", the instruction writes the detected value on a rising edge (falling edge) to RecordedPositionPos (RecordedPositionNeg), and the output parameter Done is set when one detection cycle is finished.
    - When the TriggerMode is set to "Continuous trigger" and the TriggerEdge is set to "Rising edge (Falling edge)", the instruction writes the detected value on a rising edge (falling edge) to RecordedPositionPos (RecordedPositionNeg), and the output parameter Done is reset one cycle after being set to continue detecting the signal on next rising edge (falling edge).
    - When the TriggerMode is set to "Single trigger" and the TriggerEdge is set to "Rising edge and falling edge", the instruction immediately writes the detected value on a rising edge to RecordedPositionPos and the detected value on a falling edge to RecordedPositionNeg, and the output parameter Done is set when one detection cycle is finished.
    - When the TriggerMode is set to "Continuous trigger" and the TriggerEdge is set to "Rising edge and falling edge", the instruction immediately writes the detected value on a rising edge to RecordedPositionPos, and the output parameter Done is reset one cycle after being set; the

instruction immediately writes the detected value on a falling edge to RecordedPositionNeg, and the output parameter Done is reset one cycle after being set. This instruction is executed repeatedly as described above.

- This instruction can be executed regardless of the axis state and the control mode. One function block instance represents only one MC_TouchProbe instruction. If this instruction is executed for multiple times, only the axis position when the last trigger signal occurs will be recorded.
- Re-executing this instruction will cause the function block to issue a fault message.
- This instruction issues a fault message when the touch probe channel is occupied.
- The touch probe channel is automatically released when the output parameter Done, Error, or CommandAborted is set.
- This instruction can be aborted by the MC_AbortTrigger, MC_MoveLink, and MC_MoveFeed instructions only during execution.
- **Before this instruction is executed, configure the following PDOs as appropriate:**
  - PDOs that must be configured:

    60B8 (touch probe control word), 60B9 (touch probe status word).
  - ProbeID = 0 and TriggerEdge = 0:

    60BA (touch probe 1 rising edge trigger position).
  - ProbeID = 0 and TriggerEdge = 1:

    60BB (touch probe 1 falling edge trigger position).
  - ProbeID = 0 and TriggerEdge = 2:

    60BA and 60BB.
  - ProbeID = 1 and TriggerEdge = 0:

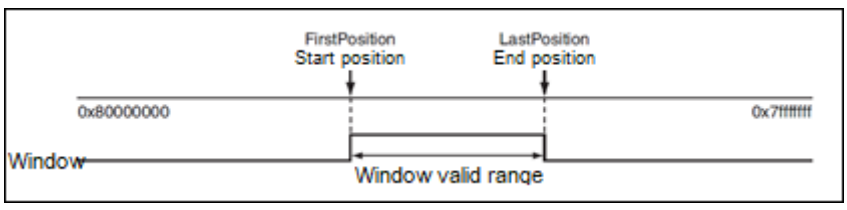
    60BC (touch probe 2 rising edge trigger position).
  - ProbeID = 1 and TriggerEdge = 1:

    60BD (touch probe 2 falling edge trigger position).
  - ProbeID = 1 and TriggerEdge = 2:

    60BC and 60BD.
- **When the input parameter WindowOnly is set to FALSE**

  The instruction does not identify the FirstPosition or LastPosition.
- **When the input parameter WindowOnly is set to TRUE**
  - Linear mode

    The FirstPosition must be less than the LastPosition.

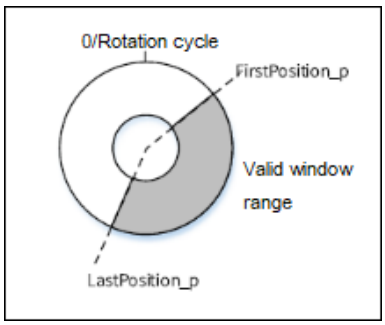
    

  - Rotary mode

    The FirstPosition can be less than or greater than the LastPosition.

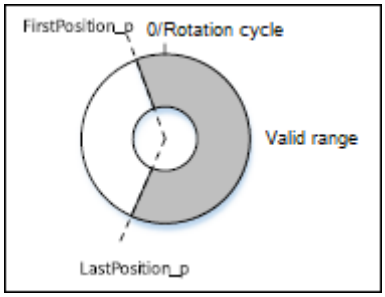
FirstPosition < LastPosition

The window is valid when the actual position is between the FirstPosition and the LastPosition, as shown in the following figure:



FirstPosition > LastPosition

The window is valid when the actual position is greater than the FirstPosition and less than the cycle length, or less than the LastPosition and greater than the cycle length, as shown in the following figure:



- For the EtherCAT bus-controlled axis, the input parameter TerminalSource can be specified as DI terminal or encoder Z-phase signal (requiring drive support). If the Z-phase signal is not supported by the drive, the instruction will not issue a fault message.
- This instruction can be triggered once or continuously, which is specified using the input parameter TriggerMode.
  - When the input parameter TriggerMode is set to 0 (single trigger), the output parameter Done is set when the instruction execution ends.
  - When the input parameter TriggerMode is set to 1 or 2 (continuous trigger), the instruction can continuously detect touch probe signals. The parameter Done is set for one cycle when successful position capture is performed in the continuous trigger mode
  - TriggerMode1: This mode applies to servo axes whose PDO 60B9 supports returning the number of latches in continuous mode. In this mode, continuous latching performance is high, supporting higher trigger frequencies. The following 60B9 objects meet the requirements for this mode:

bit	Description	Note
0	Touch probe 1 enable: 0: Disabled 1: Enable	bit0 to bit2: status of touch probe 1
1	Touch probe 1 latch on a rising edge: 0: Disabled 1: Enabled	

bit	Description	Note
2	Probe 1 latch on a falling edge: 0: Disabled 1: Enabled	
3 to 5	N/A	-
6 to 7	Trigger count of the touch probe when the function of touch probe 1 is set to continuous sampling	Trigger count (0 to 3) of the touch probe when the function of touch probe 1 is set to continuous sampling
8	Touch probe 2 enable: 0: Disabled 1: Enable	bit8 to bit10: status of touch probe 2
9	Touch probe 2 latch on a rising edge: 0: Disabled 1: Enabled	
10	Probe 2 latch on a falling edge: 0: Disabled 1: Enabled	
11 to 13	N/A	-
14 to 15	Trigger count of the touch probe when the function of touch probe 2 is set to continuous sampling	Trigger count (0 to 3) of the touch probe when the function of touch probe 2 is set to continuous sampling

- TriggerMode2: This mode applies to servo axes whose PDO 60B9 does not support returning the number of latches in continuous mode. In this mode, continuous latching performance is lower. In case of higher-frequency touch probe signals, use Mode 1. The following 60B9 objects meet the requirements for this mode:

bit	Description	Note
0	Touch probe 1 enable: 0: Disabled 1: Enabled	bit0 to bit2: status of touch probe 1
1	Touch probe 1 latch on a rising edge: 0: Disabled 1: Enabled	
2	Probe 1 latch on a falling edge: 0: Disabled 1: Enabled	
3 to 7	N/A	-

bit	Description	Note
8	Touch probe 2 enable: 0: Disabled 1: Enabled	bit8 to bit10: status of touch probe 2
9	Touch probe 2 latch on a rising edge: 0: Disabled 1: Enabled	
10	Probe 2 latch on a falling edge: 0: Disabled 1: Enabled	
11 to 15	N/A	-

Local pulse axis:

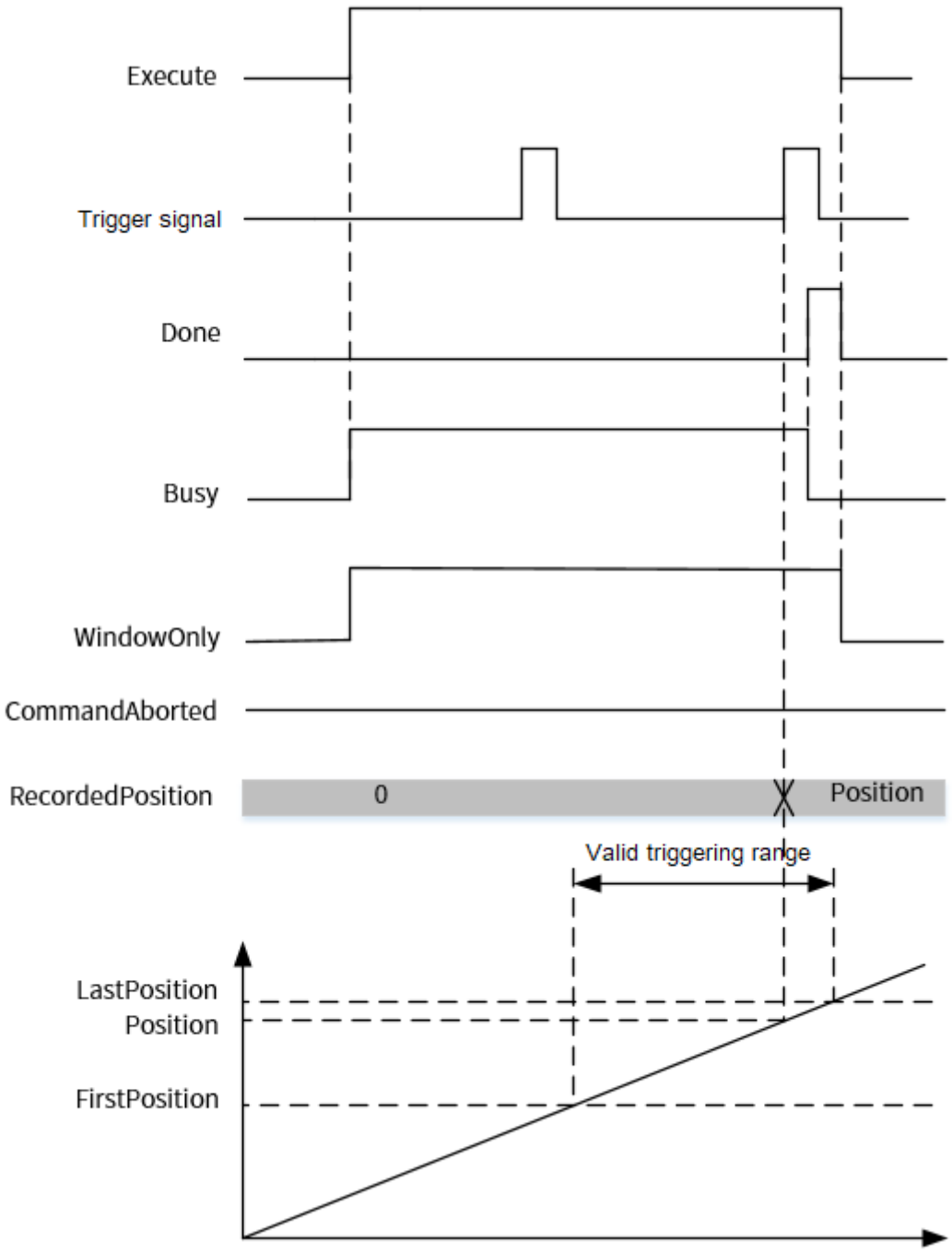
This instruction can be used for local pulse axes.

Virtual axis probe:

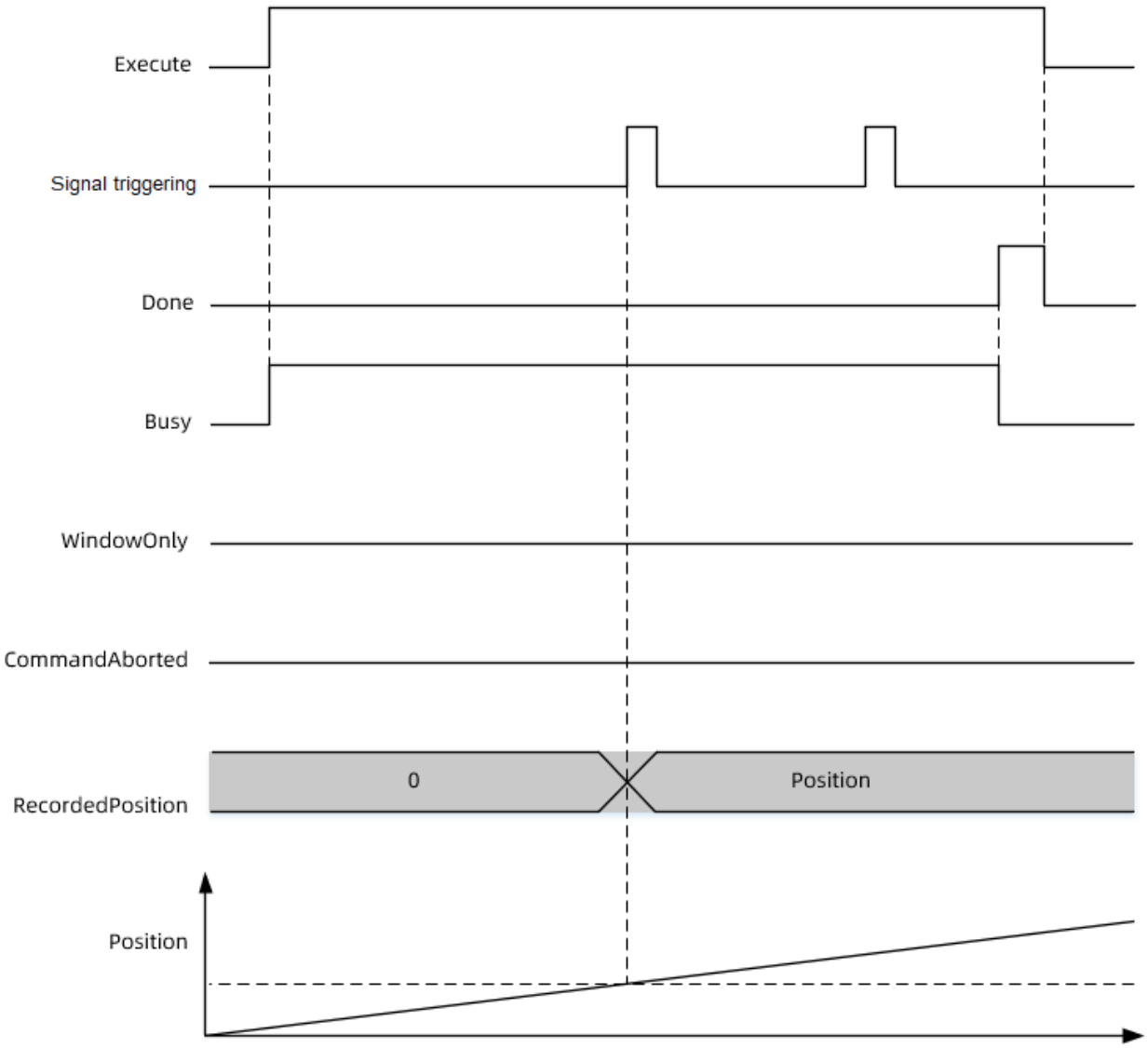
Only two-channel touch probe signals are supported for each controller corresponding to a virtual axis probe.

- **Timing diagram**

  TriggerEdge = Rising edge and WindowOnly = TRUE



TriggerEdge = Rising edge and WindowOnly = FALSE

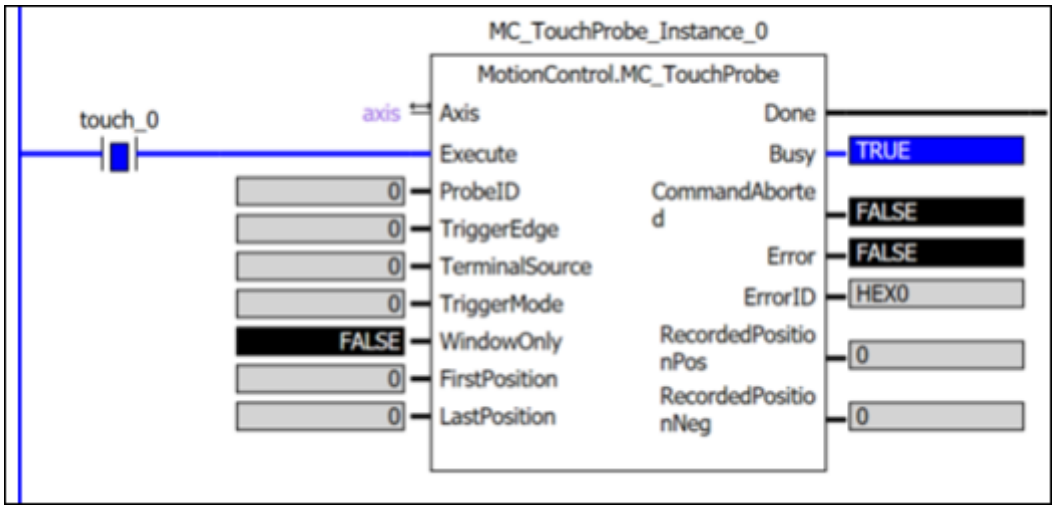


- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

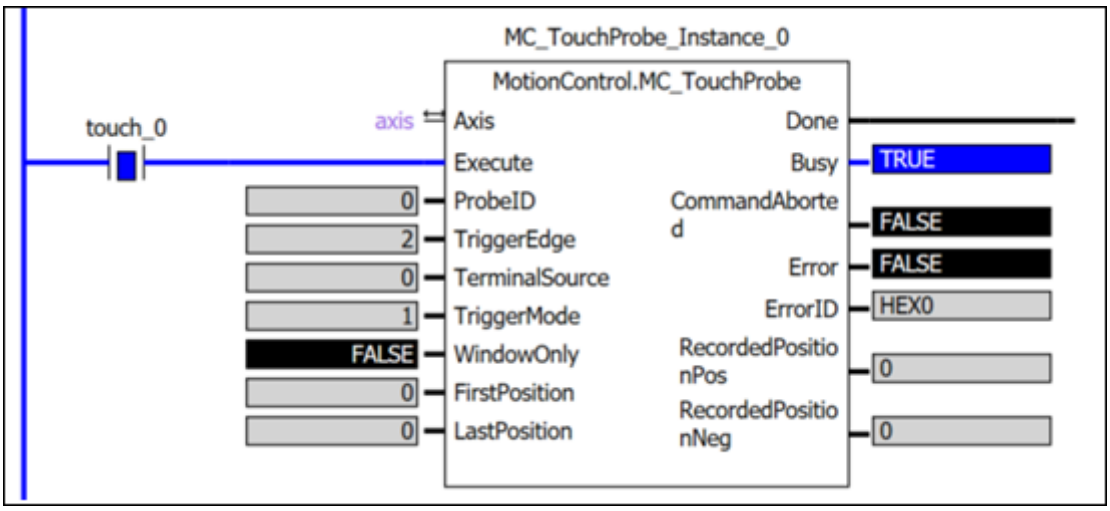
- **Example**

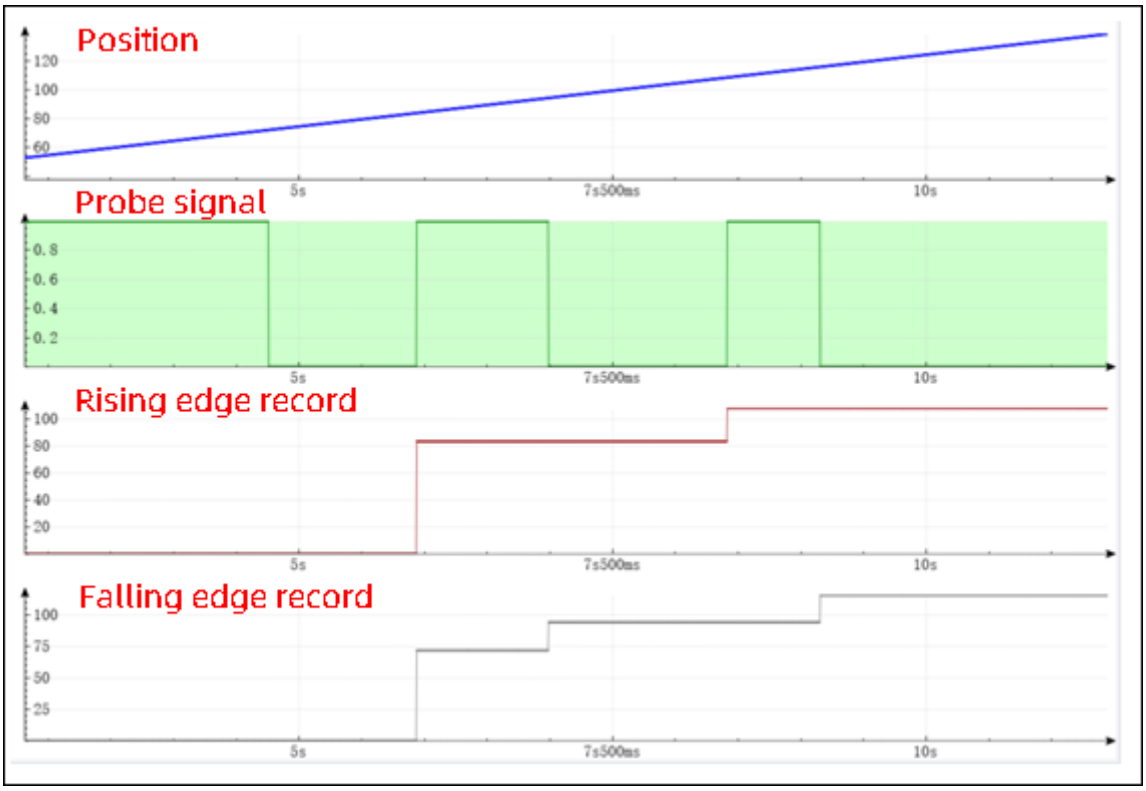
  Add proper probe-specific PDOs, select the first group of PDOs as inputs, and add 60BB and 60BD to the output configuration.



Directly call the instruction to receive the recorded position on a rising edge of drive touch probe 1.

- **Precautions**
  - This instruction is only applicable to bus-controlled axes, virtual axes, and local pulse axes. For local encoder axes or bus encoder axes, use the ENC_TouchProbe instruction.
  - Regarding compatibility with Inovance's common IS620N, the second touch probe is not supported by IS620N. The ProbeID of the axis applicable to IS620N must be set to 1, and no Bit can trigger the touch probe when the TerminalSource is set to 0.
  - For servo products compatible with IS620N, when the TriggerMode is set to "Continuous trigger" and the TriggerEdge is set to "Rising edge and falling edge", the first position cannot be recorded if the RecordedPositionNeg is the first record.

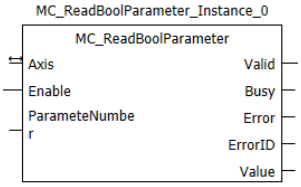




18.3.3.3.11 MC_ReadBoolParameter

This instruction reads the bit parameter of an axis.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_ReadBoolParameter	Read BOOL parameter		MC_ReadBoolParameter _Instance_0(Axis :=, Enable :=, ParameterNumber :=, Valid =>, Busy =>, Error =>, ErrorID =>, Value =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled. FALSE: Disabled.
ParameterNumber	Parameter number	DINT	See 18.3.3.9 ParameterNumberin the appendix	-	Index, subindex, or sequence number for accessing an axis parameter.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Valid output	BOOL	TRUE, FALSE	FALSE	TRUE when there is a valid output from the function block.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.
Value	Return value	BOOL	TRUE, FALSE	FALSE	Boolean value of the parameter to be read.

- **Function**

  Bus-controlled axis:

  - This instruction reads the bit data status in the drive. It is controlled by the Enable signal level and can be executed repeatedly without affecting each other. Note: Multiple different instantiated function blocks of this instruction can be executed simultaneously, instead of multiple identical instantiated function blocks.
  - This instruction must be re-executed to modify the parameter number (PN). When this instruction reads the current PN value, directly modifying the PN value causes an instruction fault. The instruction's function is restored after the trigger is canceled.
  - When the axis is in abnormal communication, this instruction issues a fault message with a fault code.

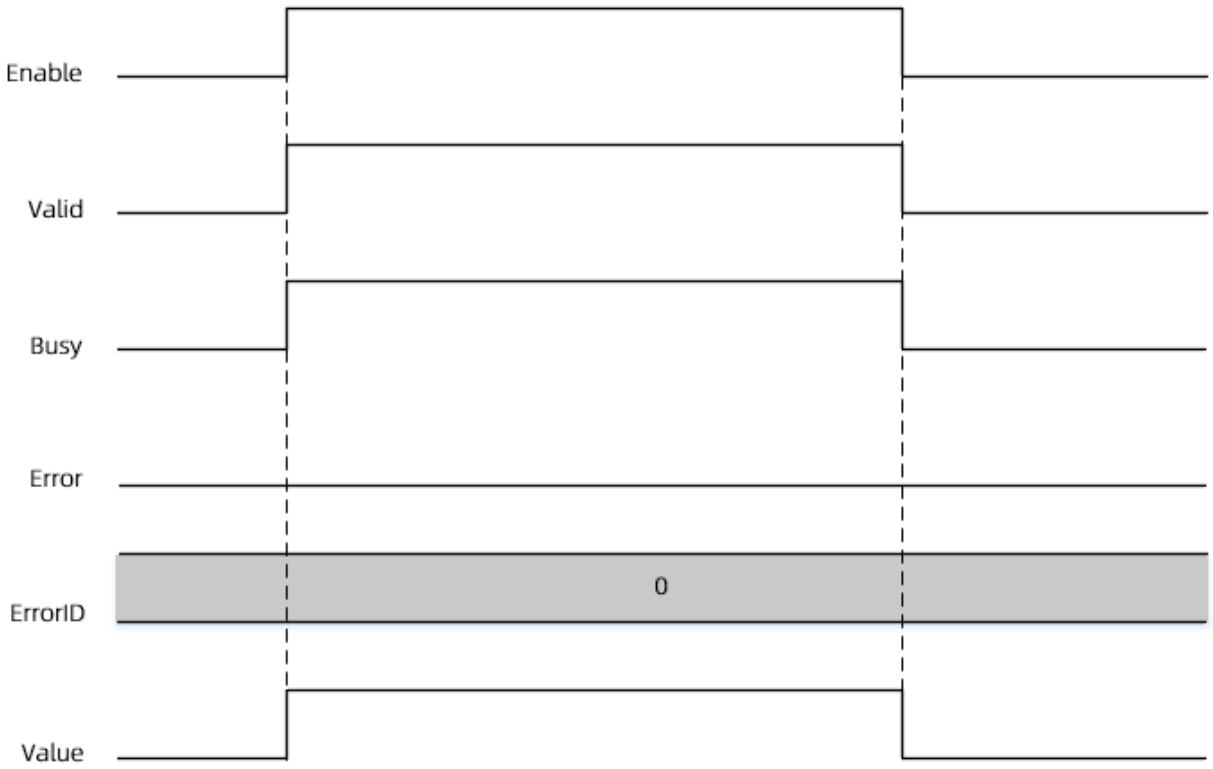
  Local pulse axis:

  - This instruction can be used for local pulse axes.

- **Timing diagram**

  The input parameter Enable must be triggered on a rising edge. The input parameter Busy indicates that the current function block is being executed, and the input parameter Valid indicates that the output parameters are valid.

  When the input parameter Enable is set to TRUE, the output parameter Busy shows the value TRUE.



- **Re-execution of motion control instructions**

  This instruction cannot be re-executed.

- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

- **Precautions**

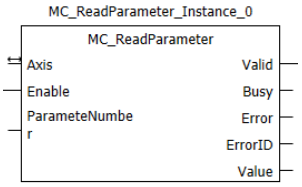
  The ParameterNumber pin can be specified based on the object dictionary index:
  diParameterNumber := -DWORD_TO_DINT(SHL(TO_DWORD(usiDataLength),24) + SHL(TO_DWORD(uiIndex),8) + usisubIndex)
  Where, usiDataLength(USINT) represents the object dictionary data length, such as USINT corresponding to 1 byte 16#01 and UINT corresponding to 2 bytes 16#02; uiIndex(UINT) represents the object dictionary, such as 16#60E0 and 16#60E1; usisubIndex(USINT) represents the object dictionary subindex, such as 16#6091 (01 corresponding to the numerator of the gear ratio, 02 corresponding to the denominator of the gear ratio).
  Example: Read and write object dictionary 16#60E0 with data type = UINT and no subindex. Relevant parameters are shown in the following figure:

```
usiDataLength: USINT := 2;
uiIndex: UINT := 16#60E0;
usisubIndex: USINT := 0;
diParameterNumber: DINT := -DWORD_TO_DINT(SHL(TO_DWORD(usiDataLength),24) + SHL(TO_DWORD(uiIndex),8) + usisubIndex);
```

18.3.3.3.12 MC_ReadParameter

This instruction reads an axis parameter.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_ReadParameter	Read LREAL parameter		MC_ReadParameter_Instance_0(Axis :=, Enable :=, ParameterNumber :=, Valid =>, Busy =>, Error =>, ErrorID =>, Value =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled. FALSE: Disabled.
ParameterNumber	Parameter number	DINT	See 18.3.3.9 ParameterNumberin the appendix	-	Number of the parameter to be read.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Valid output	BOOL	TRUE, FALSE	FALSE	TRUE when there is a valid output from the function block.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.
Value	Return value	LREAL	-	-	Value of the parameter to be read.

- **Function**

  Bus-controlled axis:

  - This instruction reads an axis parameter. It is a function block controlled by the Enable signal level. This instruction can be executed repeatedly without affecting each other. Note: Multiple different instantiated function blocks of this instruction can be executed simultaneously, instead of multiple identical instantiated function blocks.
  - When the axis is in abnormal communication, this instruction issues a fault message with a fault code.

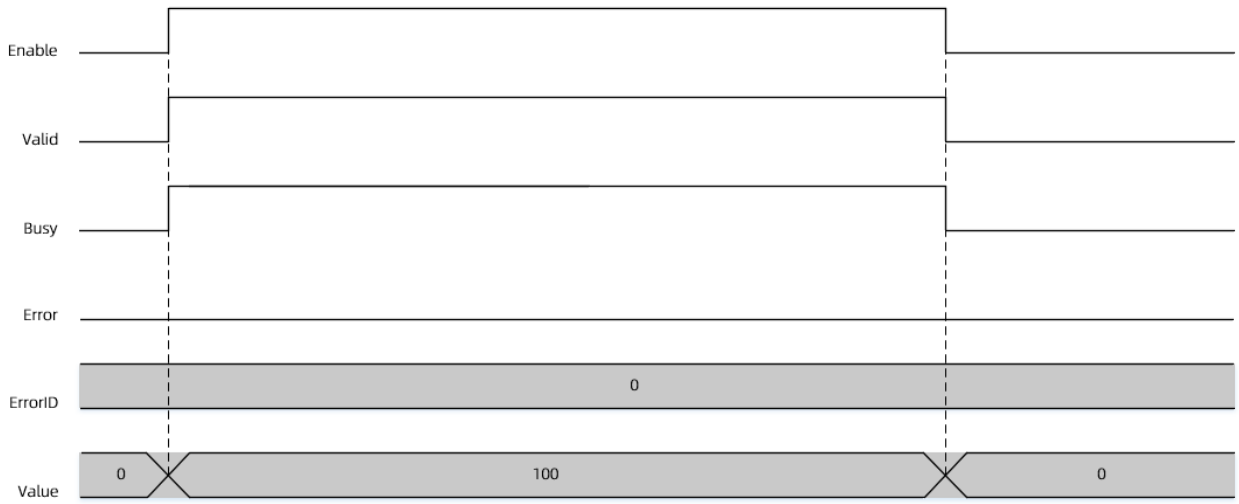
  Local pulse axis:

  - This instruction can be used for local pulse axes.

- **Timing diagram**

  The input parameter Enable must be triggered on a rising edge. The input parameter Busy indicates that the current function block is being executed, and the input parameter Valid indicates that the output parameters are valid.

  When the input parameter Enable is set to TRUE, the output parameter Busy shows the value TRUE.



- **Re-execution of motion control instructions**

  This instruction cannot be re-executed.

- **Faults**

For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

- **Precautions**

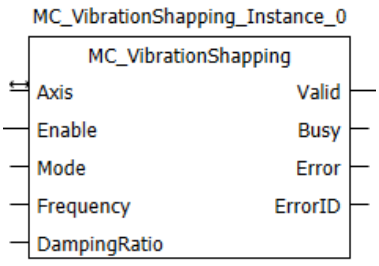
The ParameterNumber pin can be specified based on the object dictionary index:
diParameterNumber := -DWORD_TO_DINT(SHL(TO_DWORD(usiDataLength),24) + SHL(TO_DWORD(uiIndex),8) + usisubIndex)
Where, usiDataLength(USINT) represents the object dictionary data length, such as USINT corresponding to 1 byte 16#01 and UINT corresponding to 2 bytes 16#02; uiIndex(UINT) represents the object dictionary, such as 16#60E0 and 16#60E1; usisubIndex(USINT) represents the object dictionary subindex, such as 16#6091 (01 corresponding to the numerator of the gear ratio, 02 corresponding to the denominator of the gear ratio).
Example: Read and write object dictionary 16#60E0 with data type = UINT and no subindex. Relevant parameters are shown in the following figure:

```
usiDataLength: USINT := 2;
uiIndex: UINT := 16#60E0;
usisubIndex: USINT := 0;
diParameterNumber: DINT := -DWORD_TO_DINT(SHL(TO_DWORD(usiDataLength),24) + SHL(TO_DWORD(uiIndex),8) + usisubIndex);
```

18.3.3.3.13 MC_VibrationShapping

This instruction performs vibration shaping filtering to reduce residual vibration in a single-mode system.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_VibrationShapping	Input shaping		MC_VibrationShapping_Instance_0 (Axis :=, Enable :=, Mode :=, Frequency :=, DampingRatio :=, Valid =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	Enable flag for the instruction execution.
Mode	Mode	INT	0, 1	0	0: ZV shaper model, suitable for single-frequency stable systems. 1: ZVD shaper model, suitable for multiple-frequency systems; more robust.
Frequency	Frequency	LREAL	> 0	0	Frequency.
DampingRatio	Damping ratio	LREAL	(0,1)	0	Damping ratio.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Input shaping	BOOL	TRUE, FALSE	FALSE	TRUE: Input shaping is being executed.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE: The function block is being executed.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE: A fault occurs during input shaping.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus-controlled axis:

  - If an attempt is made to execute multiple instances of this instruction, the last triggered instance will issue a fault message, without affecting the currently executing instruction.
  - During the output shaping, when the parameter Valid is set and the parameter Enable is set to FALSE, the instruction will issue an axis fault message. It is not recommended to frequently trigger Enable, as this may cause servo overspeed.
  - This instruction dynamically allocates memory during execution, with a maximum allowed array length of 1000. The size of memory allocated depends on the input parameters. If the input parameters are invalid and result in excessive memory allocation, the instruction will issue a fault message.
  - This instruction can be used in linear axis mode only.
  - This instruction can be used for servo axes or local axes, and will issue a fault message for other axis types.
  - This instruction requires the axis to be enabled before execution; otherwise, the instruction will issue a fault message.

- If the slave is disconnected during normal execution of the instruction, the instruction will issue a fault message.
- In the project configuration, there is no requirement on the placement order of this instruction; it may be placed either before or after the function block.
- The input parameters of this instruction take effect only when the instruction is triggered, and any input parameter changes after the trigger will not take effect in real time.
- This instruction can be executed only when the axis is Standstill; otherwise, the instruction will issue a fault message. After the instruction is executed, it will issue a fault message if the axis performs homing or stopping operation, and output in normal shaping mode if the axis performs any other motions.
- This instruction will issue a fault message in a non-CSP control mode.
- The parameter value 0 of the Mode represents the ZV shaper model, and 1 represents the ZVD shaper model. The ZVD shaper is more robust than the ZV shaper.
- When this instruction is triggered, triggering the MC_SetPosition instruction will result in a fault message.
- When this instruction is used with software limits being checked, limit identification is performed based on the planned position at the axis. After the limit position is reached, the shaping position is synchronized to the planned position to prevent jumping change in position.

Local pulse axis:

- This instruction can be used for local pulse axes.

- **Timing diagram**

  When the input parameter Enable is set to TRUE, the output parameter Busy shows the value TRUE, indicating the current function block is being executed. When the output parameter Valid is set to TRUE, it indicates that input shaping is being executed.



- **Re-execution of motion control instructions**

  You cannot re-execute motion control instructions with an input parameter Enable.

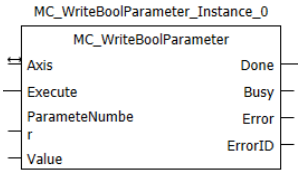
- **Faults**

For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.14 MC_WriteBoolParameter

This instruction writes the BOOL parameters of an axis.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_WriteBoolParameter	Write BOOL parameters		MC_WriteBoolParameter _Instance_0(Axis :=, Execute :=, ParameterNumber :=, Value :=, Done =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.
ParameterNumber	Parameter number	DINT	See 18.3.3.9 ParameterNumberin the appendix	-	Number of the axis parameter to be written.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Value	Setpoint	BOOL	TRUE, FALSE	FALSE	Setpoint of the axis parameter to be written.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus-controlled axis:

  ◦ This instruction sets the bit parameters of an axis and is triggered on a rising edge of Execute. This instruction can be executed repeatedly without affecting each other. Note: Multiple different instantiated function blocks of this instruction can be executed simultaneously, instead of multiple identical instantiated function blocks.

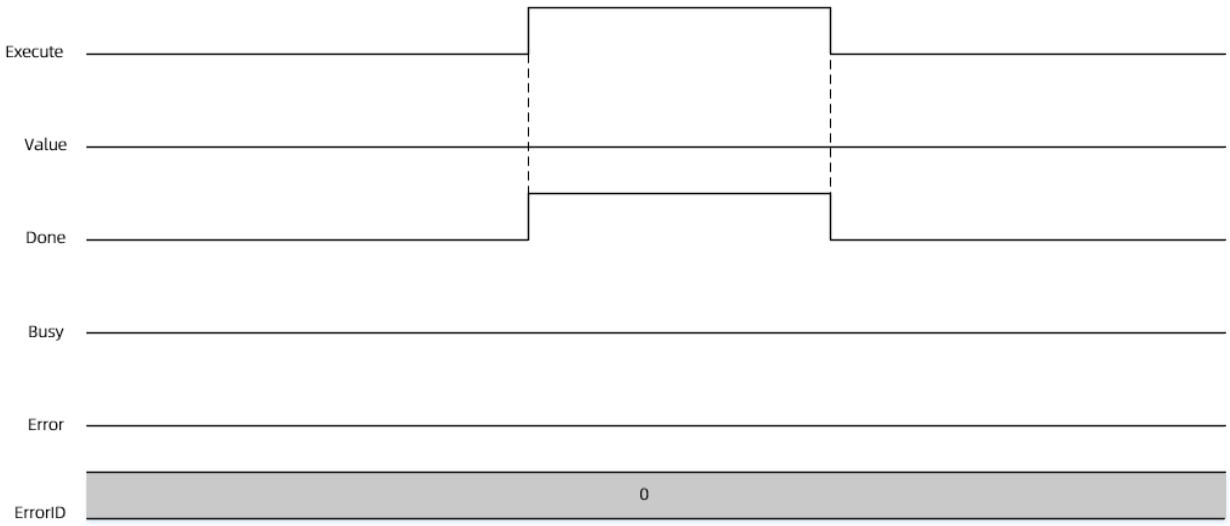
  Local pulse axis:

  ◦ This instruction can be used for local pulse axes.

- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Value of the function block indicates the setpoint of the axis parameter written; the parameter Done indicates that the instruction has been executed successfully; the parameter Busy indicates that the current function block is being executed.

  After this instruction is called, the output parameter Busy becomes TRUE on a rising edge of Execute. After the written axis parameter is set successfully, the output parameter Done is set to TRUE.



- **Re-execution of motion control instructions**

  This instruction cannot be re-executed.

- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

- **Precautions**

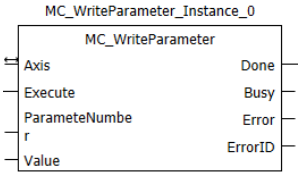
  The ParameterNumber pin can be specified based on the object dictionary index:
  diParameterNumber := -DWORD_TO_DINT(SHL(TO_DWORD(usiDataLength),24) + SHL(TO_DWORD(uiIndex),8) + usisubIndex)
  Where, usiDataLength(USINT) represents the object dictionary data length, such as USINT corresponding to 1 byte 16#01 and UINT corresponding to 2 bytes 16#02; uiIndex(UINT) represents the object dictionary, such as 16#60E0 and 16#60E1; usisubIndex(USINT) represents the object dictionary subindex, such as 16#6091 (01 corresponding to the numerator of the gear ratio, 02 corresponding to the denominator of the gear ratio).
  Example: Read and write object dictionary 16#60E0 with data type = UINT and no subindex. Relevant parameters are shown in the following figure:

```
usiDataLength: USINT := 2;
uiIndex: UINT := 16#60E0;
usisubIndex: USINT := 0;
diParameterNumber: DINT := -DWORD_TO_DINT(SHL(TO_DWORD(usiDataLength),24) + SHL(TO_DWORD(uiIndex),8) + usisubIndex);
```

18.3.3.3.15 MC_WriteParameter

This instruction writes the LREAL parameters of the axis structure.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_WriteParameter	Write LREAL parameters		MC_WriteParameter_Instance_0(Axis :=, Execute :=, ParameterNumber :=, Value :=, Done =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.
ParameterNumber	Parameter number	DINT	See 18.3.3.9 ParameterNumberin the appendix	-	Number of the axis parameter to be written.
Value	Setpoint	LREAL	-	0	Setpoint of the axis parameter to be written.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus-controlled axis:

  ◦ This instruction sets the parameters of an axis and is triggered on a rising edge of Execute. This instruction can be executed repeatedly without affecting each other.

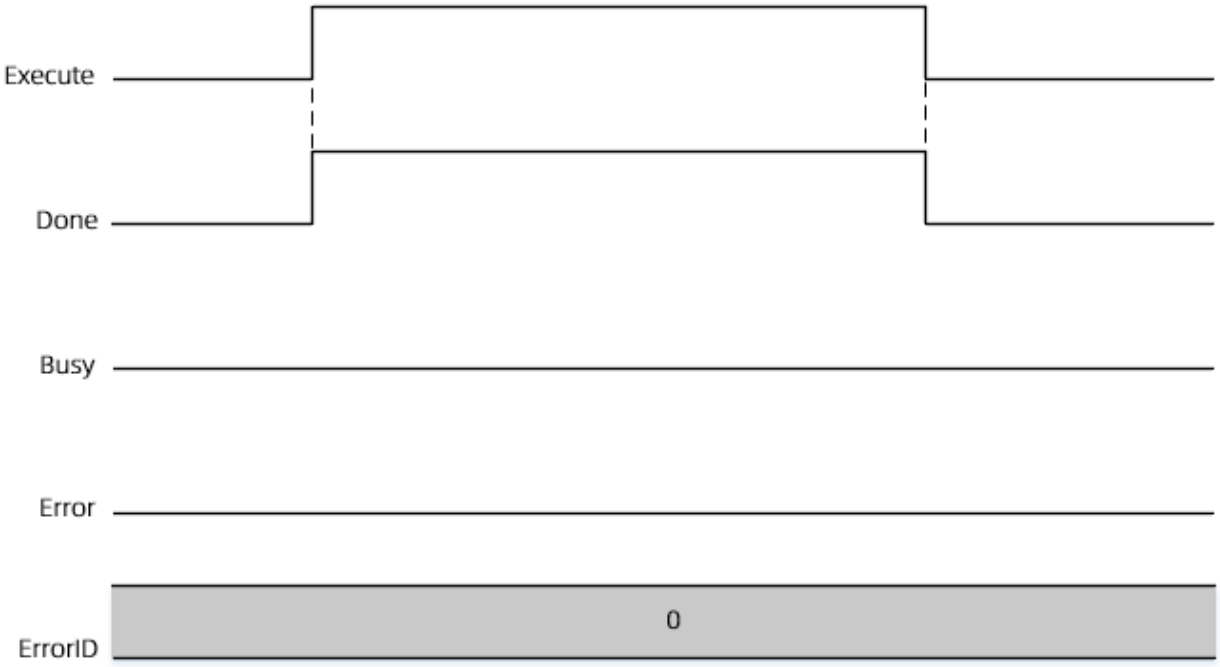
  Local pulse axis:

  ◦ This instruction can be used for local pulse axes.

- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Value of the function block indicates the setpoint of the axis parameter written; the parameter Done indicates that the instruction has been executed successfully; the parameter Busy indicates that the current function block is being executed.

  After this instruction is called, the output parameter Busy becomes TRUE on a rising edge of Execute. After the written axis parameter is set successfully, the output parameter Done is set to TRUE.



- **Re-execution of motion control instructions**

  This instruction cannot be re-executed.

- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

- **Precautions**

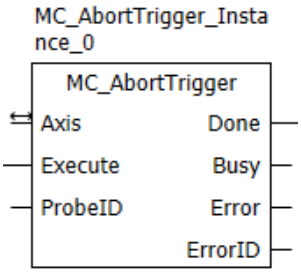
  The ParameterNumber pin can be specified based on the object dictionary index:
  diParameterNumber := -DWORD_TO_DINT(SHL(TO_DWORD(usiDataLength),24) + SHL(TO_DWORD(uiIndex),8) + usisubIndex)
  Where, usiDataLength(USINT) represents the object dictionary data length, such as USINT corresponding to 1 byte 16#01 and UINT corresponding to 2 bytes 16#02; uiIndex(UINT) represents the object dictionary, such as 16#60E0 and 16#60E1; usisubIndex(USINT) represents the object dictionary subindex, such as 16#6091 (01 corresponding to the numerator of the gear ratio, 02 corresponding to the denominator of the gear ratio).
  Example: Read and write object dictionary 16#60E0 with data type = UINT and no subindex. Relevant parameters are shown in the following figure:

```
usiDataLength: USINT := 2;
uiIndex: UINT := 16#60E0;
usisubIndex: USINT := 0;
diParameterNumber: DINT := -DWORD_TO_DINT(SHL(TO_DWORD(usiDataLength),24) + SHL(TO_DWORD(uiIndex),8) + usisubIndex);
```

18.3.3.3.16 MC_AbortTrigger

This instruction aborts the latch operation triggered by the MC_TouchProbe instruction.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_AbortTrigger	Abort triggered operation		MC_AbortTrigger_Instance_0(Axis :=, Execute :=, ProbeID :=, Done =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.
ProbeID	Touch probe ID	INT	0, 1	0	0: Touch probe 1. 1: Touch probe 2.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus-controlled axis:

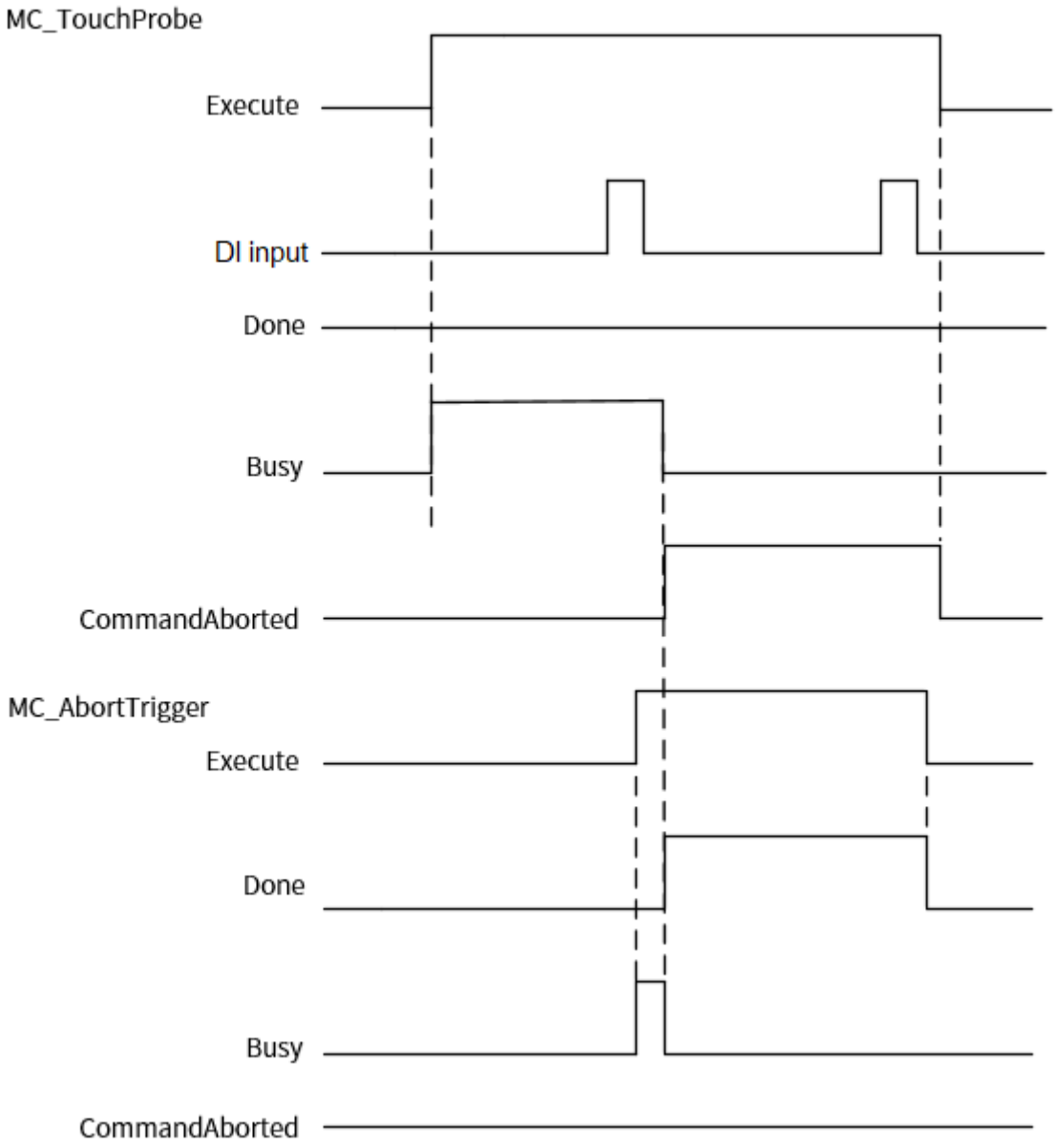
  - This instruction is used in conjunction with the MC_TouchProbe instruction.
  - You can use this instruction to specify the Axis and ProbeID to abort the latch operation triggered by the MC_TouchProbe instruction.
  - If the MC_TouchProbe instruction is not executed on the current axis and the MC_AbortTrigger instruction is applied directly, the Done output flag of the MC_AbortTrigger is set. If the MC_AbortTrigger instruction is executed after the MC_TouchProbe instruction is executed on the current axis with the TriggerMode being set to "Single trigger" and having a successful position recording, the Done output flag of the MC_AbortTrigger is set.

  Local pulse axis:

  - This instruction can be used for local pulse axes.

- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates that the instruction has been executed successfully, and the parameter Busy indicates that the current function block is being executed. After this instruction is called, the output parameter Busy becomes TRUE on a rising edge of Execute. After the latch operation triggered by the MC_TouchProbe instruction is aborted, the output parameter Done is set to TRUE.



- **Re-execution of motion control instructions**

  This instruction cannot be re-executed.

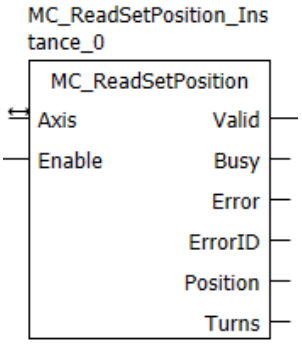
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.17 MC_ReadSetPosition

This instruction reads the set position of an axis.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_ReadSetPosition	Read set position		MC_ReadSetPosition_Instance_0(Axis :=, Enable :=, Valid =>, Busy =>, Error =>, ErrorID =>, Position =>, Turns =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled. FALSE: Disabled.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Valid output	BOOL	TRUE, FALSE	FALSE	TRUE when there is a valid output from the function block.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.
Position	Position	LREAL	-	0	Return the set position [u] for the current task cycle.
Turns	Number of turns	LINT	-	0	Number of turns in rotary mode.

- **Function**

  Bus-controlled axis:

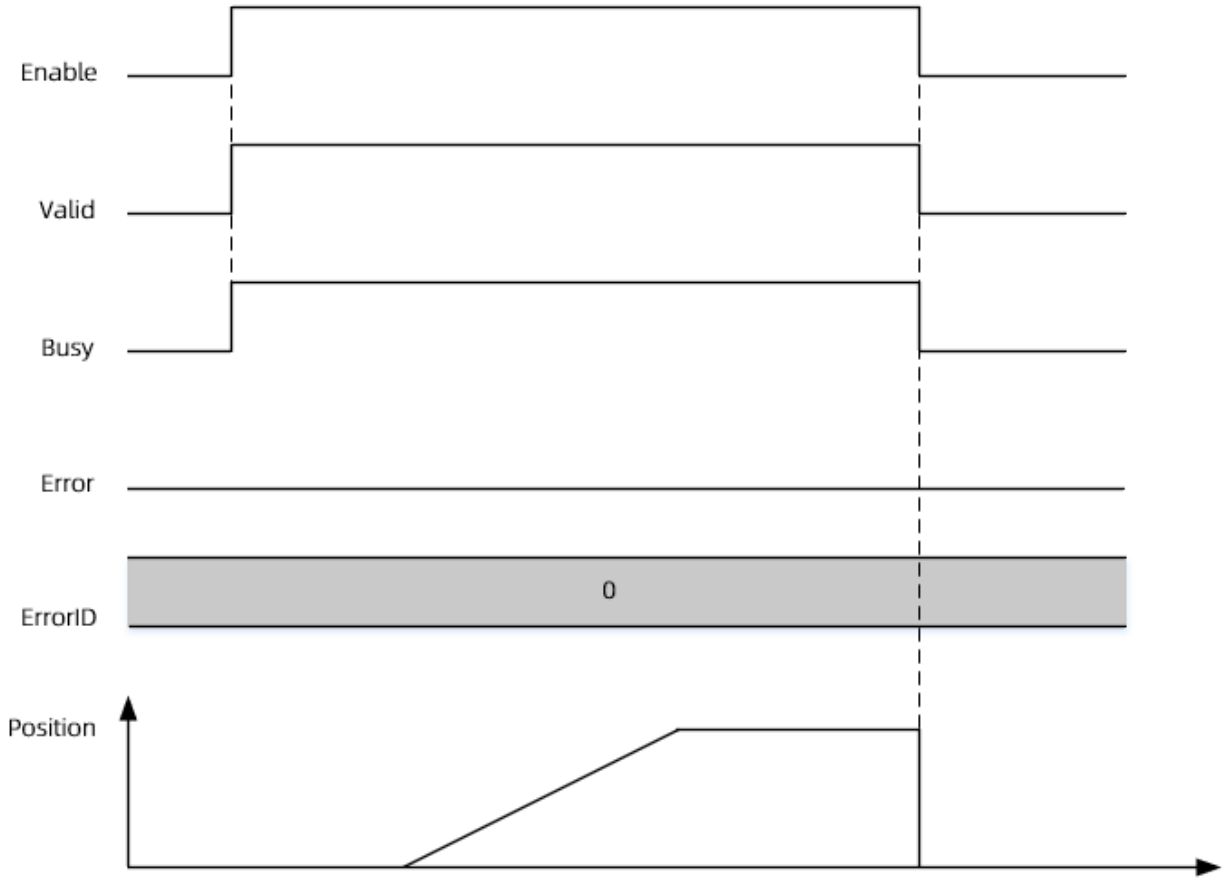
  - This instruction reads the set position of an axis.
  - The value of the output variable Position is the set position for the current task cycle.
  - The output variable Turns is based on the absolute linear position. It counts 0 when the axis position is within the first full revolution in the positive direction. It counts from -1 when the axis position is within the first full revolution in the negative direction.
  - After the parameter Enable of the instruction is reset, the output parameters are reset.
  - The instruction reading is not affected by axis state change or mode switching.
  - This instruction reads the parameter normally even if the MC_Power instruction is not triggered.
  - The axis type must be control axis. Abnormal communication results in an instruction fault.

  Local pulse axis:

  - This instruction can be used for local pulse axes.

- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Valid of the function block indicates valid outputs; the parameter Done indicates that the instruction has been executed successfully; the parameter Busy indicates that the current function block is being executed.



- **Re-execution of motion control instructions**

  This instruction cannot be re-executed.

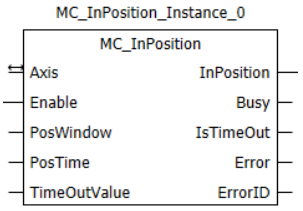
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.18 MC_InPosition

This instruction monitors the deviation between the set position and actual position of an axis and identifies whether the axis is within the required deviation range based on a specified deviation window.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_InPosition	Monitor axis deviation		MC_InPosition_Instance_0(Axis :=, Enable :=, PosWindow :=, PosTime :=, TimeOutValue :=, InPosition =>, Busy =>, IsTimeOut =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled. FALSE: Disabled.
PosWindow	Position deviation window	LREAL	Positive number	0	Specify a window for position deviation monitoring.
PosTime	Holding time	LREAL	Positive number	0	If the following error is less than or equal to the input parameter PosWindow for the duration (s) greater than the input parameter PosTime, the output parameter InPosition is set to TRUE.
TimeOut Value	Timeout value	LREAL	-	0	If the following error is greater than the input parameter PosWindow for the duration greater than the input parameter TimeOutValue, the output parameter IsTimeOut is set to TRUE.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
InPosition	In position	BOOL	TRUE, FALSE	FALSE	TRUE when the position deviation is within the specified window for the holding time.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
IsTimeOut	Timeout flag	BOOL	TRUE, FALSE	FALSE	If the InPosition is not set to TRUE after the TimeOutValue, the output parameter IsTimeOut is set to TRUE.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus-controlled axis:

  - This instruction monitors the deviation between the set position and actual position of an axis and identifies whether the axis is within the required deviation range based on a specified deviation window.
  - The setpoints of input parameters PosWindow, PosTime, and TimeOutValue must be greater than or equal to 0; otherwise, the function block will issue a fault message.
  - When the Enable shows the value TRUE and the axis state is AXIS_STANDSTILL, AXIS_STOPPING, AXIS_DISCRETE_MOTION, AXIS_CONTINUOUS_MOTION, or AXIS_SYNCHRONIZED_MOTION, if the detected deviation value (Position reference - Actual position) is less than or equal to the setpoint of PosWindow, and the duration for the detected deviation value being less than the setpoint of PosWindow is greater than the setpoint of PosTime, the output parameter InPosition is set to TRUE; otherwise, the InPosition is set to FALSE. When the detected deviation value is greater than the setpoint of PosWindow, the InPosition becomes FALSE and the timer restarts.

- Timing starts from the moment the instruction is triggered. When the duration is greater than the setpoint of TimeOutValue, the output parameter IsTimeOut is set to TRUE. After the IsTimeOut shows the value TRUE, it will not change back to FALSE even if the deviation is less than the setpoint of PosWindow.
- It is recommended to specify the value of TimeOutValue to be greater than the value of PosTime when this instruction is used.
- The axis controlled by this instruction must be bus-controlled axis, local pulse axis, or virtual axis. This instruction will issue a fault message if it is executed on any other axis type.

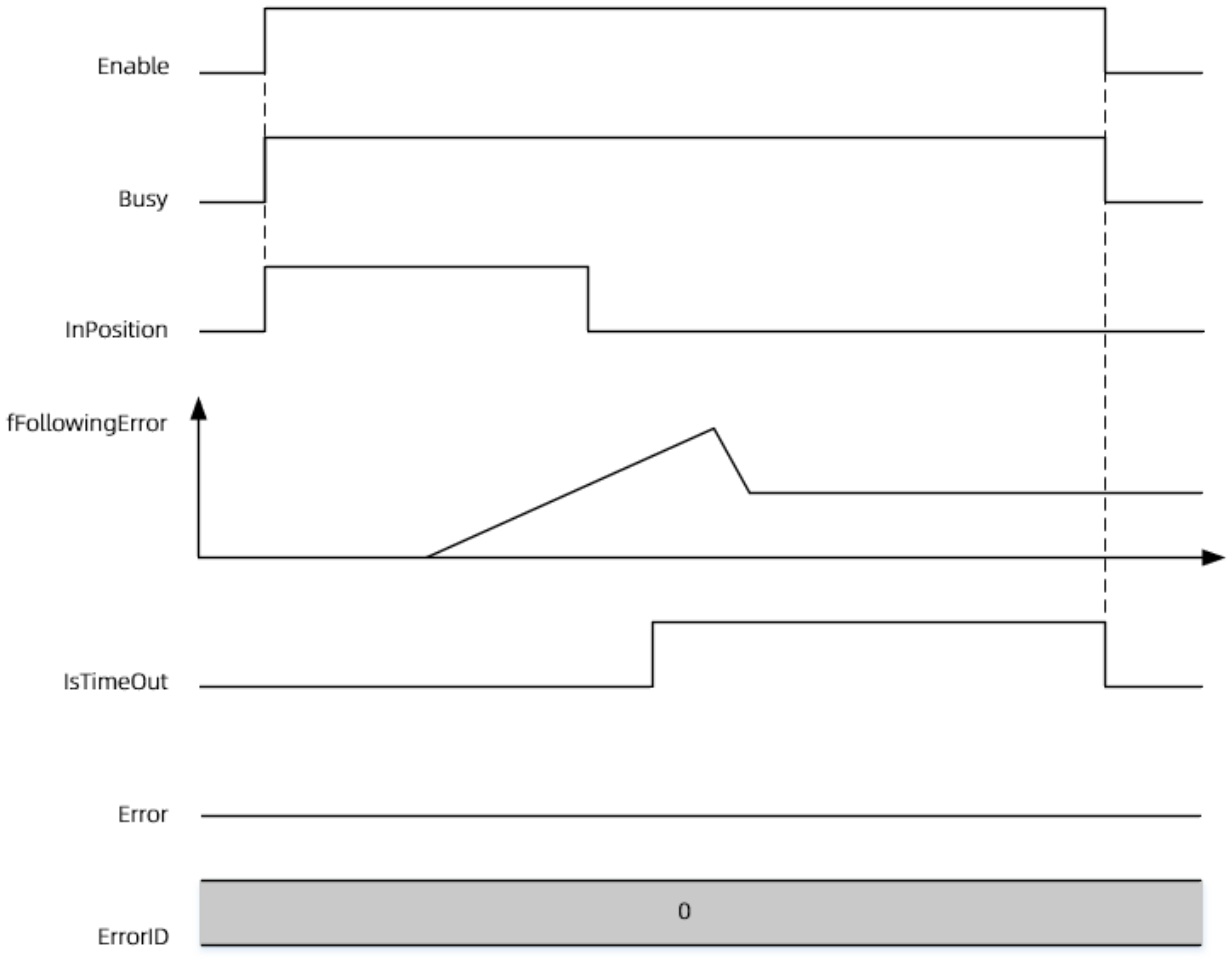
  Local pulse axis:

- This instruction can be used for local pulse axes.

- **Timing diagram**

  When the input parameter Enable is set to TRUE, the output parameter Busy shows the value TRUE, indicating the current function block is being executed. When the input parameter Enable is set to FALSE, the output parameter Busy shows the value FALSE.

  The axis structure variable fFollowingError represents the following error. When the fFollowingError is greater than the setpoint of the PosWindow, the InPosition is set to FALSE immediately, and the timer restarts. If the duration is greater than the value of TimeOutValue at the end of the planning, the output parameter changes to TRUE.



- **Re-execution of motion control instructions**

  You cannot re-execute motion control instructions with an input parameter Enable.

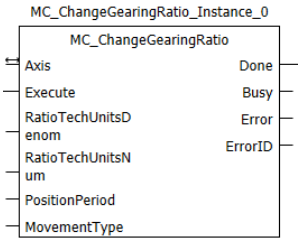
- **Faults**

For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.19 MC_ChangeGearingRatio

This instruction sets the gear ratio (pulse unit to user unit) and the axis movement type.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_ChangeGearingRatio	Set gear ratio and axis type		MC_ChangeGearingRatio_Instance_0(Axis :=, Execute :=, RatioTechUnitsDenom :=, RatioTechUnitsNum :=, PositionPeriod :=, MovementType :=, Done =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.

RatioTechUnits Denom	Denominator	DWORD	Not equal to 0	1	Denominator of the conversion factor for converting drive increments into count units.
RatioTechUnits Num	Numerator	LREAL	Not equal to 0	1	Numerator of the conversion factor for converting drive increments into count units.
PositionPeriod	Rotation cycle	LREAL	Greater than 0	-	Rotation cycle [u], for rotary axes only.
MovementType	Movement type	INT	0, 1	0	0: Rotary axis. 1: Linear axis.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus-controlled axis:

  - This instruction can be called only when the axis state is AXIS_POWEROFF; otherwise, the function block reports a fault.
  - This instruction can be used for bus-controlled axes, local pulse axes, and virtual axes only. This instruction will issue a fault message if it is executed on any other axis type.
  - If the input parameter iMovementType is not 0 or 1, a fault will occur. The input parameters RatioTechUnitsDenom and RatioTechUnitsNum cannot be 0. When the MovementType is set to 0, the

input parameter PositionPeriod must be greater than 0, and the number of pulses corresponding to the rotation cycle cannot be greater than 2,147,483,647.

- If the axis has been connected to an axis group (i.e., the MC_GroupEnable instruction has been executed on the axis group), using this instruction to switch the axis to rotary axis will result in an instruction fault. This instruction is executed normally with the denominator and numerator being specified for a linear axis.
- The relationship between the numerator, denominator, set position, and output pulses: User unit position = RatioTechUnitsNum * Pulse value / RatioTechUnitsDenom.

Local pulse axis:

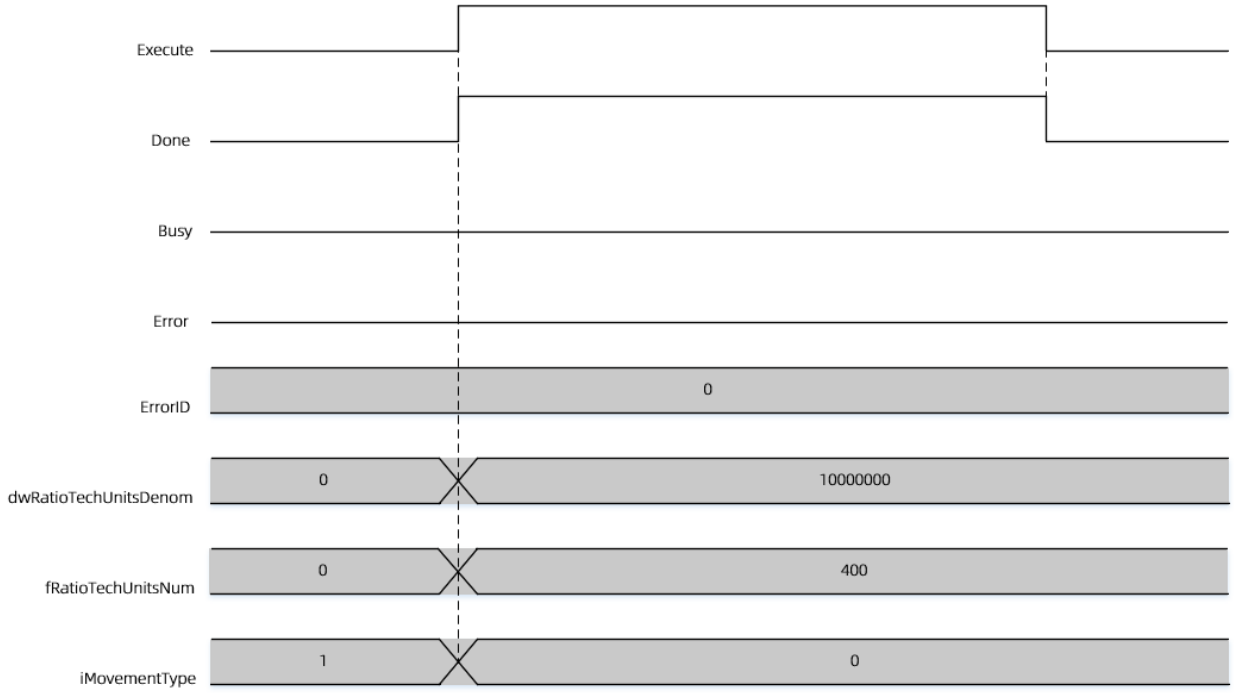
- This instruction can be used for local pulse axes.

- **Timing diagram**

  Specify the input parameter RatioTechUnitsDenom to 8,388,608, the RatioTechUnitsNum to 360, the PositionPeriod to 360, and the MovementType to 0.

  The structure variable dwRatioTechUnitsDenom is the gear ratio denominator; the fRatioTechUnitsNum is the gear ratio numerator; and the iMovementType represents the axis type.

  The gear ratio and axis movement type change if there are no faults on a rising edge of the input parameter Execute. When the input parameter Execute is set to TRUE, the output parameter Busy shows the value TRUE. After the current instruction completes the specified changes, the output parameter Done is set to TRUE.



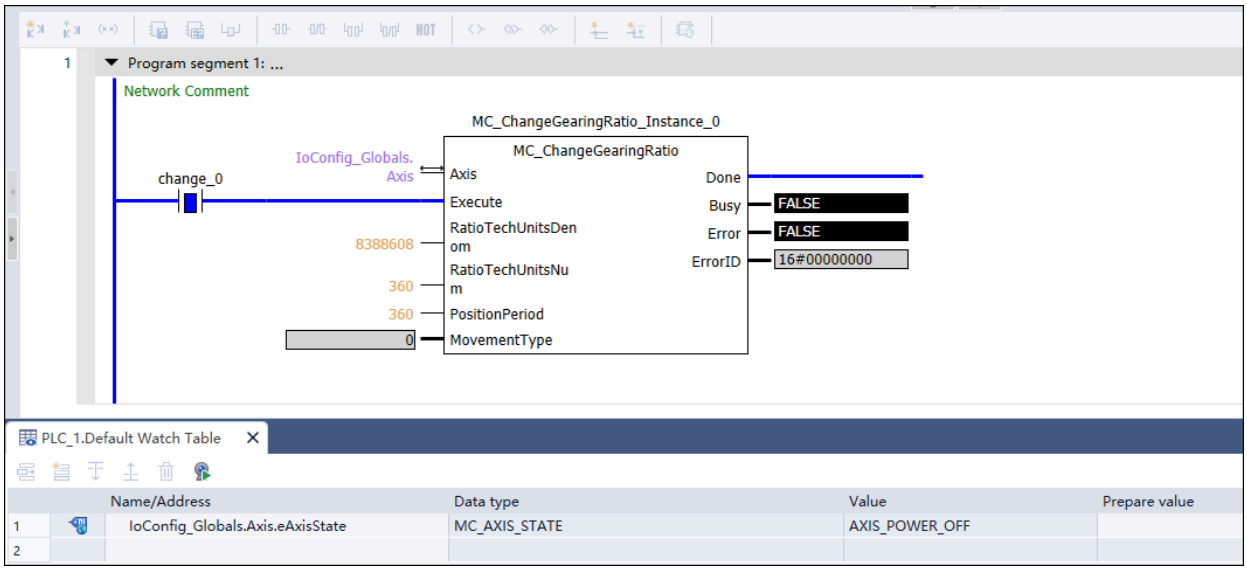
- **Re-execution of motion control instructions**

  This instruction cannot be re-executed.

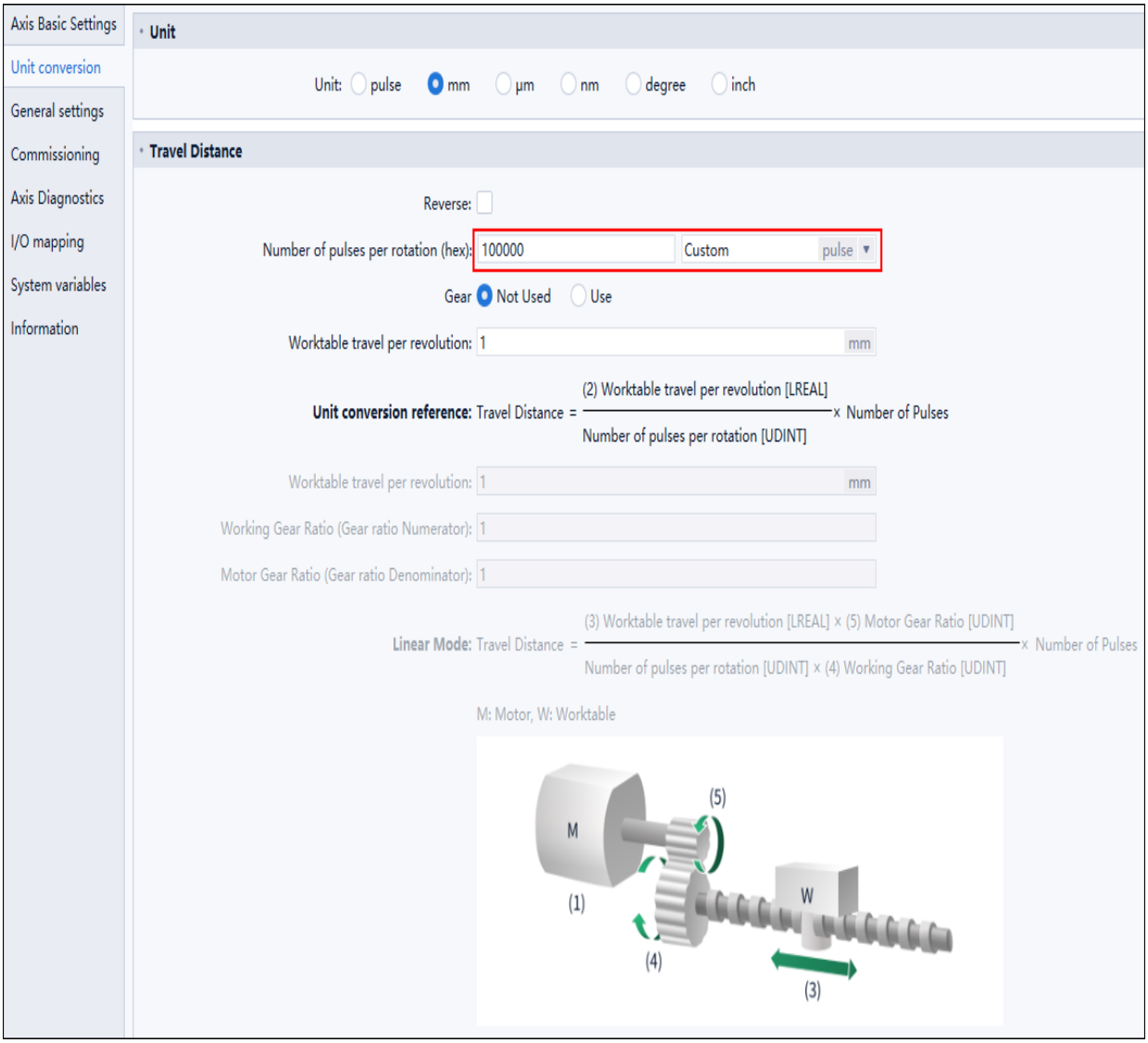
- **Faults**

For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

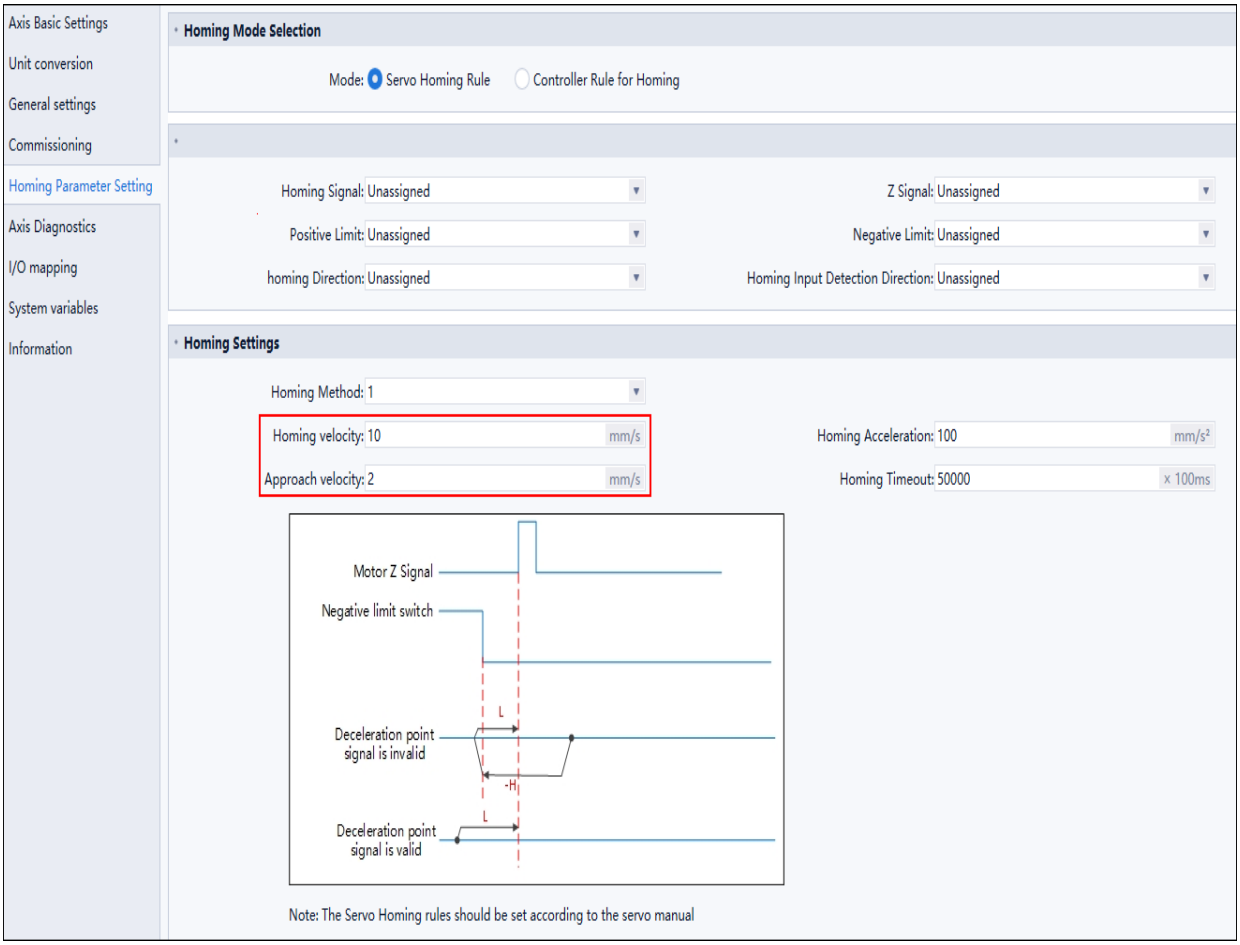
- **Precautions**
  - To adjust the number of pulses per motor revolution, you can modify the value of RatioTechUnitsDenom. For Inovance's common servo drives (23-bit), the number of pulses per revolution is 8,388,608.
  - The instruction parameters are decimal, while the software parameters are hexadecimal. The number systems must be unified before parameter conversion.
  - After the motor is disabled, adjust the instruction parameters as shown in the figure, and then enable the motor. At this point, the motor serves as a rotary axis, and one full turn of the motor corresponds exactly to 360 degrees of the user's motor.

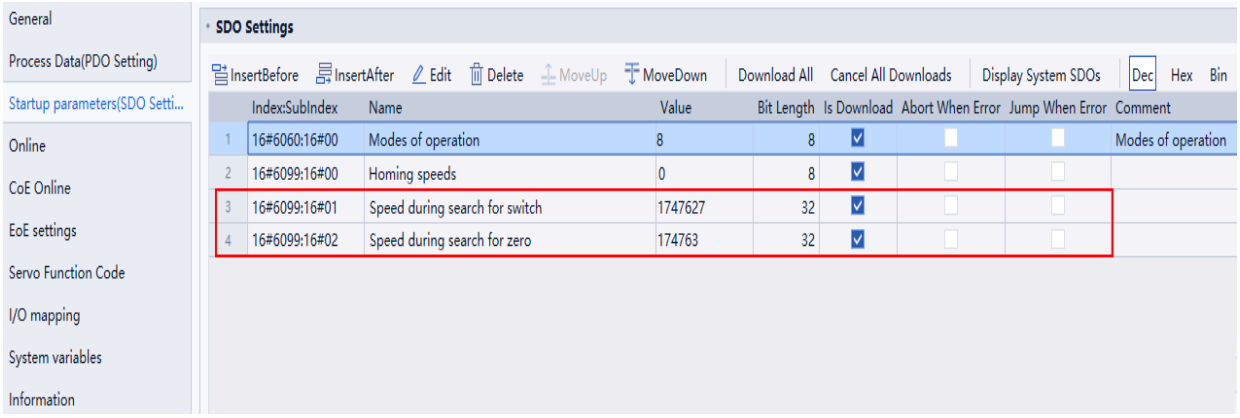


  - To ensure the accuracy of variables related to the axis and position, when the axis type is a rotary axis and the product of the rotation cycle x gear ratio is greater than 2,147,483,647, this instruction will issue a fault message.
  - After the gear ratio is modified using this instruction, the homing velocity of the axis during the servo homing is inconsistent with the parameter value in Servo Homing Rule mode on the Homing Parameter Setting interface.



In this case, in Servo Homing Rule mode on the Homing Parameter Setting interface, if the homing velocity and approach velocity are specified to 10 and 2 user units respectively, the system will automatically convert them to pulse units, as shown in the following figure.



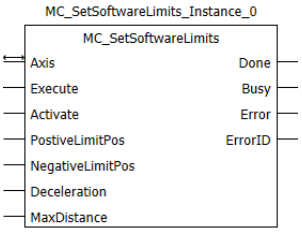


After the iFA goes online, if the MC_ChangeGearingRatio is used to modify the gearing ratio in this group of parameters, that is, to modify the ratio between user units and pulse units, and the homing instruction is executed, the homing velocity and approach velocity during the axis homing will be displayed according to the modified gearing ratio, and different from those on the Homing Parameter Setting interface.

18.3.3.3.20 MC_SetSoftwareLimits

This instruction sets the limit positions and predictive parameters of an axis.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_SetSoftwareLimits	Set software limits		MC_SetSoftwareLimits_Instance_0(Axis :=, Execute :=, Activate:=, PositiveLimitPos :=, NegativeLimitPos :=, Deceleration :=, MaxDistance :=, Done =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.
Activate	Activate software limit	BOOL	TRUE, FALSE	FALSE	TRUE: Activated. FALSE: Deactivated. The setpoint is not passed to the axis parameters.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
PositiveLimitPos	Positive software limit position	LREAL	-	0	Value of the positive limit position.
NegativeLimitPos	Negative software limit position	LREAL	-	0	Value of the negative limit position.
Deceleration	Stop deceleration	LREAL	Positive number	0	Stop deceleration when the axis position is beyond the limit position (if the setpoint is less than the maximum stop deceleration, the axis will stop according to the maximum stop deceleration).
MaxDistance	Maximum braking distance	LREAL	Positive number	0	Maximum value of the braking distance (if the stopping distance according to the stop deceleration is greater than the maximum braking distance, the axis will stop according to the maximum braking distance).

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus-controlled axis:

  - The axis type must be control axis. This instruction is only valid in Linear Mode; otherwise, the instruction will issue a fault message.
  - The setpoint of PostiveLimitPos cannot be less than or equal to that of NegativeLimitPos. If the setpoint of Deceleration or MaxDistance is 0 or invalid, it will not take effect, and the limit deceleration will be self-adaptive to a group of limit deceleration and deceleration distance.
  - Whether the Activate is set to TRUE or not, the parameters and status will be checked as long as the Execute is set.
  - The parameters set by this instruction will not be saved in the project, and when the program is re-downloaded, the parameters will be configured according to the software parameters.
  - When this instruction is triggered during the axis movement, the set parameters will take effect if the parameters Activate and Done are set to TRUE.

  Local pulse axis:

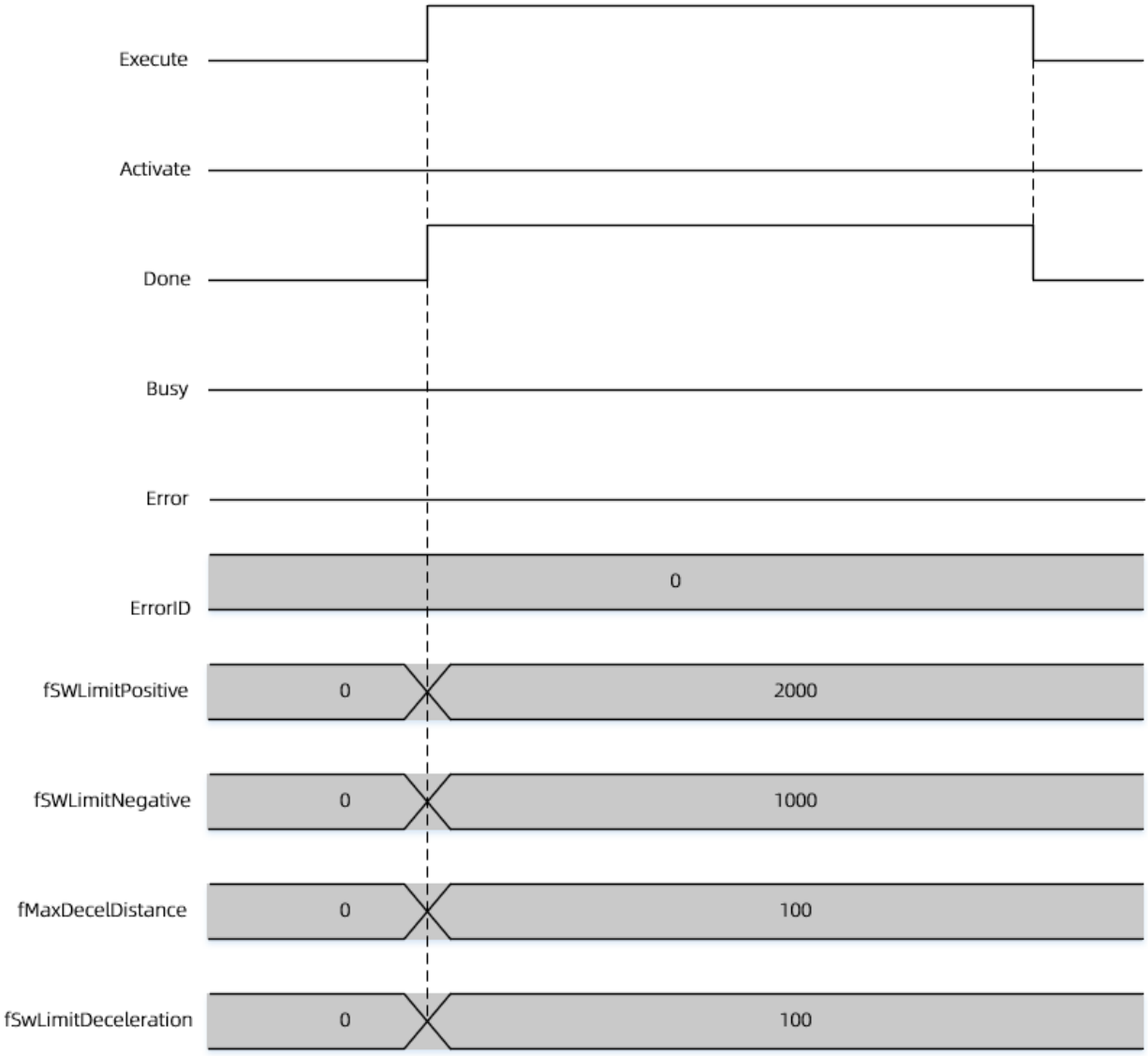
  - This instruction can be used for local pulse axes.

- **Timing diagram**

  When the input parameter Execute is set to TRUE, the output parameter Busy shows the value TRUE.

  After the axis limit position and predictive parameters are specified, the output parameter Done shows the value TRUE.

  When the input parameter Activate is set to TRUE, the input parameters will be passed to the axis and take effect. When the Activate is set to FALSE, even if the Done shows the value TRUE, the output parameters will not take effect.



- **Re-execution of motion control instructions**

  This instruction cannot be re-executed.

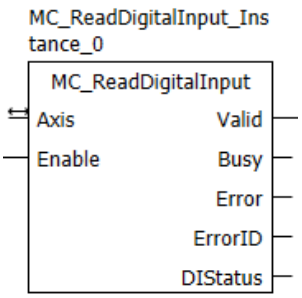
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.21 MC_ReadDigitalInput

This instruction reads the digital input of an axis.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_ReadDigitalInput	Read digital input		MC_ReadDigitalInput_Instance_0(Axis :=, Enable :=, Valid =>, Busy =>, Error =>, ErrorID =>, DIStatus =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled. FALSE: Disabled.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Valid output	BOOL	TRUE, FALSE	FALSE	TRUE when there is a valid output from the function block.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.
DIStatus	Input terminal status	DINT	-	0	Status of the digital input terminal. The standard formats compliant with 402 protocol are defined as follows: bit0: Negative limit signal. bit1: Positive limit signal. bit2: Home signal. bit3 to bit31: User-defined.

- **Function**

  Bus-controlled axis:

  - This instruction cannot be used for virtual axes.
  - If the control object is a bus-controlled axis, the PDO 60FD must be configured.

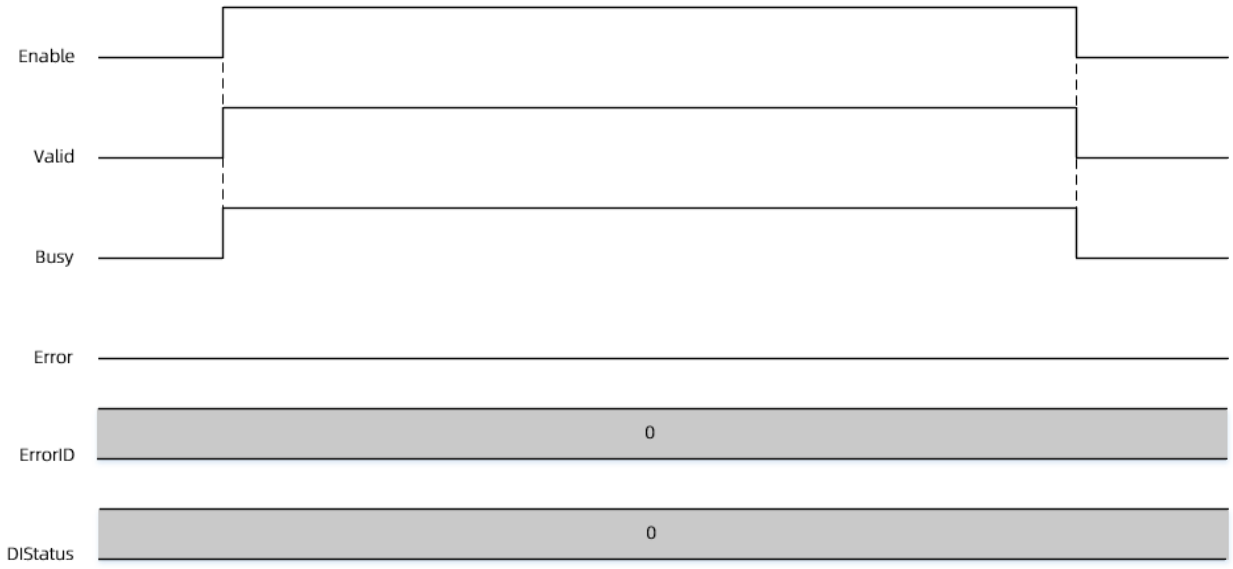
  Local pulse axis:

  - This instruction cannot be used for local pulse axes.

- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Valid of the function block indicates valid outputs, and the parameter Busy indicates that the current function block is being executed.

  After this instruction reads the digital inputs, the status of the input terminal will be output to the DIStatus.



- **Re-execution of motion control instructions**

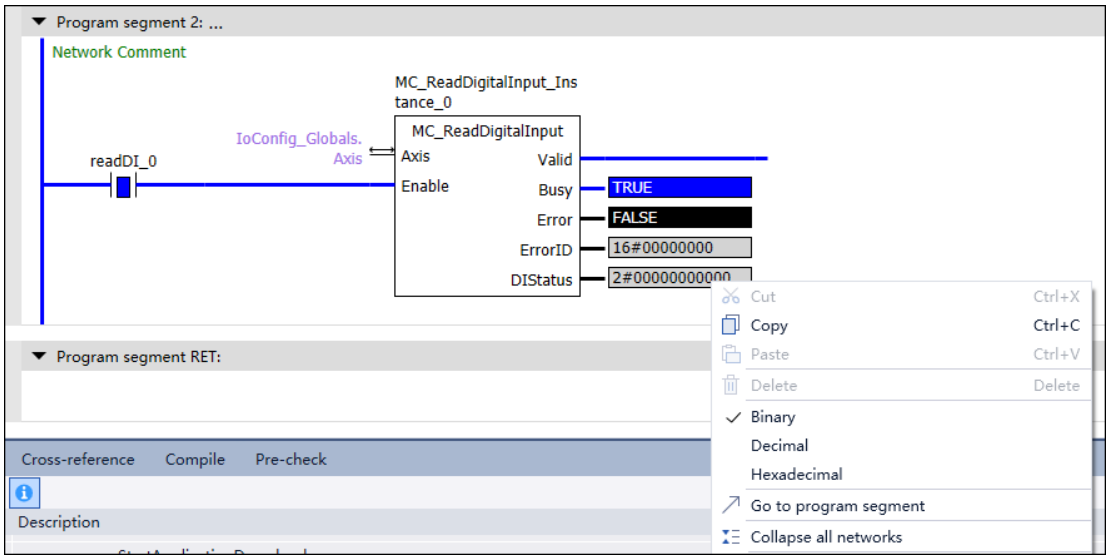
  This instruction cannot be re-executed.

- **Errors**

  For errors related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

- **Precautions**

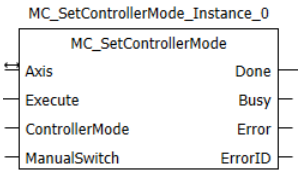
  The output variable DIStatus of this instruction is displayed in decimal by default, and must be converted to binary format to check the bit positions of different signals.



18.3.3.3.22 MC_SetControllerMode

This instruction sets the current operation mode of the servo, which is CSP (cyclic synchronous position control mode) by default.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_SetControllerMode	Set current operation mode		MC_SetControllerMode_Instance_0(Axis :=, Execute :=, ControllerMode :=, ManualSwitch :=, Done =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.
ControllerMode	Controller mode	18.3.3.2.2.7 MC_CONTROLLER_MODE: Controller Mode Enumeration	10: MC_CST 9: MC_CSV 8: MC_CSP	MC_CSP	10: MC_CST (cyclic synchronous torque control mode). 9: MC_CSV (cyclic synchronous velocity control mode). 8: MC_CSP.
ManualSwitch	Controller mode switch mode	18.3.3.2.2.17 MC_CONTROLLER_MODE_SWITCH: Controller Mode Switch Mode Enumeration	0: MC_AUTO 1: MC_MANUAL	MC_AUTO	0: MC_AUTO: Automatic switch. 1: MC_MANUAL: Manual switch.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus-controlled axis:

  - After this instruction completes the controller mode switching, signal output will immediately follow the switched mode. For example, if the MC_MoveAbsolute instruction is in the planning state in CSP mode, and a position instruction is output to the axis, the planned velocity by the MC_MoveAbsolute will remain unchanged after switching to CSV mode until the MC_MoveAbsolute planning ends.
  - The input parameter ManualSwitch of this instruction is used to specify the switch mode of control modes for different motion control instructions based on the input parameters after the Done is set. If the ManualSwitch is set to MC_AUTO, subsequent control mode switch will be performed internally by the instruction. If the ManualSwitch is set to MC_MANUAL, subsequent control mode switch will be performed through this instruction. The ManualSwitch is set to MC_AUTO by default.
  - The input parameter ManualSwitch is used to specify the switch mode of control modes for different motion control instructions.
    - MC_AUTO: Automatic switch. The logic for this method is as follows: ① When the current control mode matches the required control mode for the instruction, the instruction is executed in the current control mode. ② When the current control mode does not match the required control mode for the instruction, the instruction is executed after switching the control mode to the required mode.
    - MC_MANUAL: Manual switch. The logic for this method is as follows: ① When the current control mode matches the required control mode for the instruction, the instruction is executed directly. ② When the current control mode does not match the required control mode for the motion control instruction, the instruction must be executed after the MC_SetControllerMode instruction is executed to switch the control mode to the required one; otherwise, the motion control instruction will issue a fault message. When the MC_SetControllerMode instruction is executed with the ManualSwitch = 1, if an instruction is being executed, it will be aborted with the axis state switching to AXIS_STANDSTILL.
  - Different control modes for applicable motion control instructions are shown in the following table.

Controller mode	Motion Control Instruction
MC_CSP	MC_MoveAbsolute, MC_MoveRelative, MC_MoveContinuousAbsolute, MC_MoveContinuousRelative, MC_MoveFeed, MC_Inch, MC_CamIn, MC_CamInV2
MC_CSV	MC_SyncMoveVelocityCSV

Controller mode	Motion Control Instruction
MC_CST	MC_TorqueControl
MC_CSP/MC_CSV	MC_MoveVelocity, MC_Jog, MC_FollowPosition, MC_FollowVelocity, MC_FollowPositionVelocity, MC_GearIn, MC_GearInPos, MC_MoveLink, MC_CombineAxes, MC_GearOut, MC_CamOut

Notes: ① Regardless of the input parameter ManualSwitch, this instruction does not affect the MC_MoveSuperImposed instruction, which can be executed in the MC_CSP mode only. When the current control mode is MC_CST, a fault message will appear if the MC_MoveSuperImposed instruction is executed. ② For instructions that can be executed in both CSP and CSV mode, if high accuracy of position information is required, it is recommended to execute the instructions in the CSP mode; if only velocity information is needed, the instructions can be executed in either control mode.

- Control mode switching is supported under any mode and conditions, but control mode switching during operation may cause a sudden change in velocity.
- If the control mode is switched to CSV without PDO objects 60FF and 606C being configured, a fault message will appear.
- If the control mode is switched to CST without PDO objects 6077 and 6071 being configured, a fault message will appear.
- Control mode switching is supported among CSP, CSV, and CST modes only, and the axis type must be a control axis.
- This instruction cannot be triggered when the axis state is AXIS_STOPPING, AXIS_HOMING, or AXIS_ER RORSTOP.
- This instruction does not abort the motion control instructions. It only modifies the current control mode.
- This instruction can be executed even if PDO objects 6060 and 6061 are not configured.
- Repeated or multiple execution of this instruction results in a fault. This instruction cannot be called again before the mode switching is completed.

Local pulse axis:

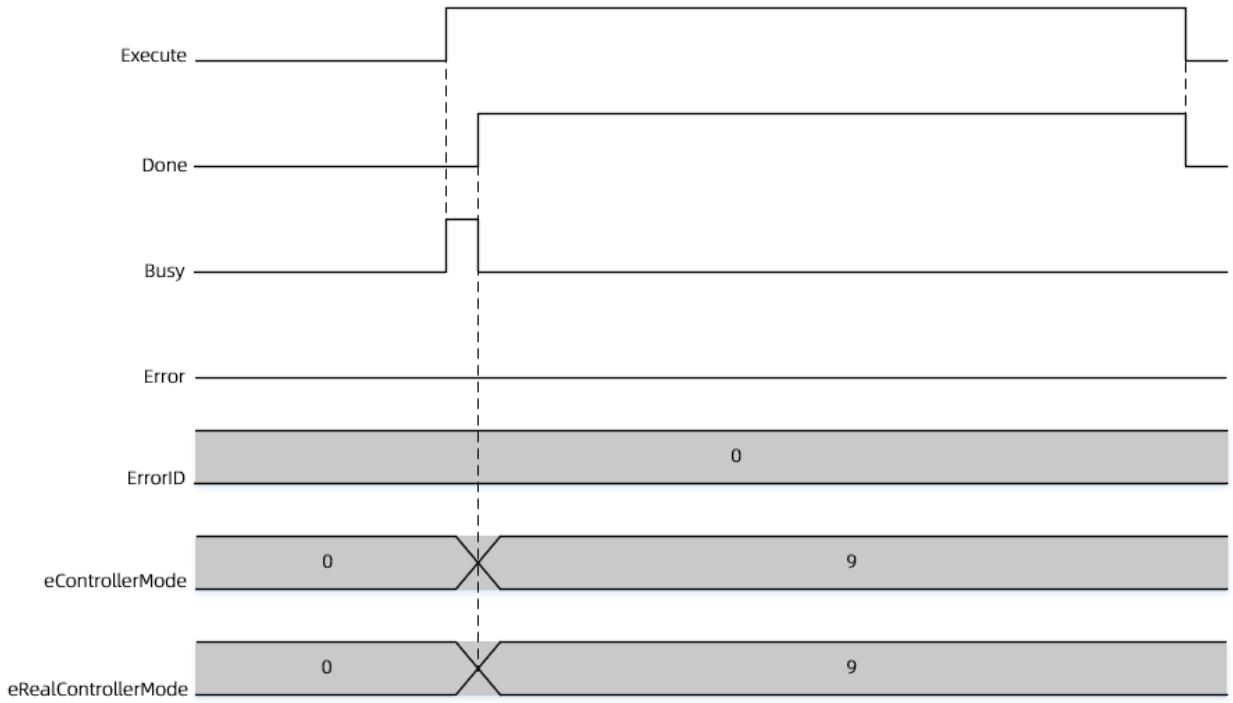
- This instruction cannot be used for local pulse axes.

- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates that the instruction has been executed successfully, and the parameter Busy indicates that the current function block is being executed.

  After this instruction is called, the output parameter Busy is set to TRUE on a rising edge of Execute. When the control mode switching is completed, the output parameter Done is set to TRUE.

  The axis structure variable eControllerMode represents the axis control mode, and the eRealControllerMode represents the actual control mode of an axis.



- **Re-execution of motion control instructions**

  This instruction cannot be re-executed.

- **Faults**

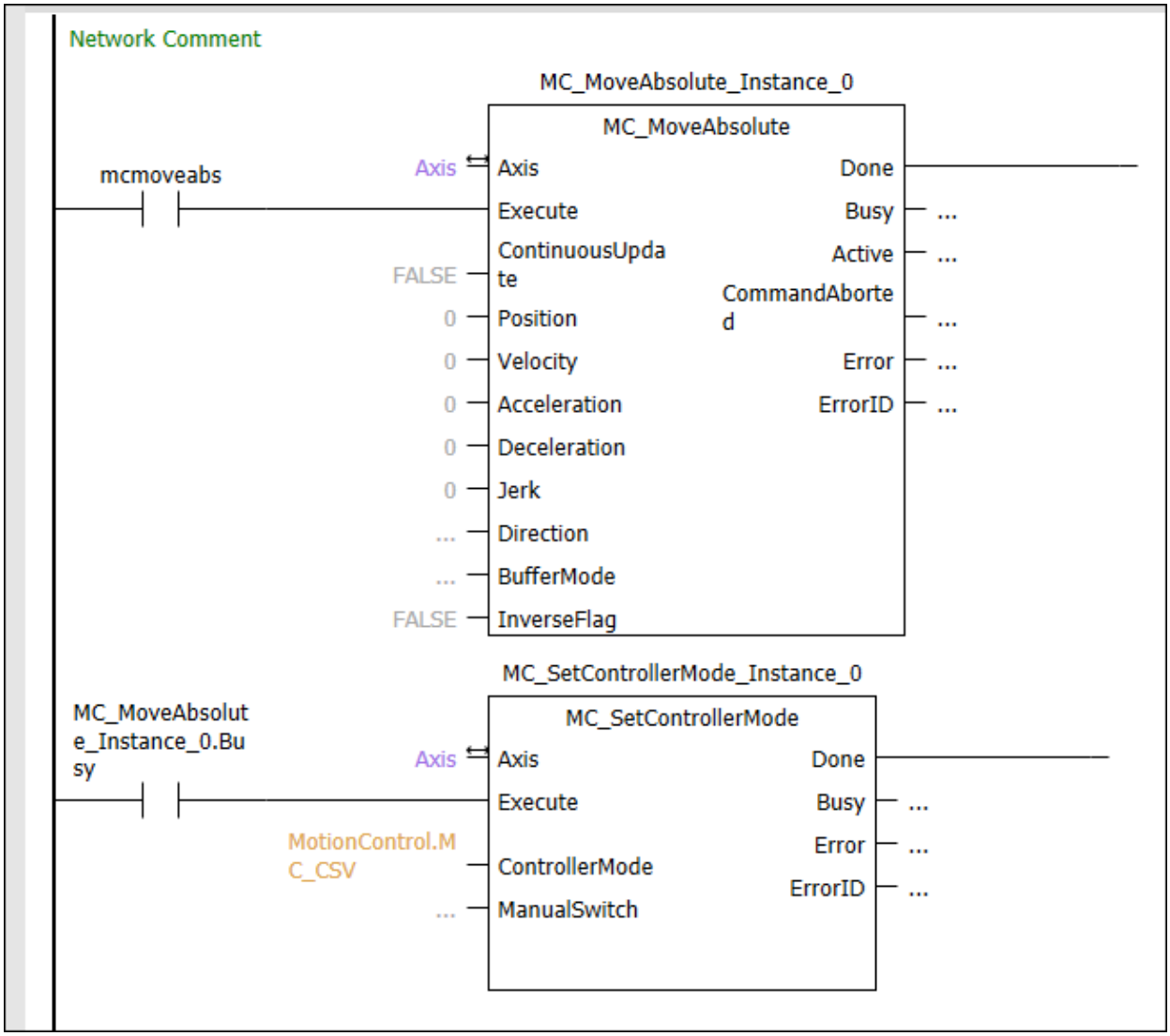
  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

- **Example**

  Currently, positioning instructions and velocity instructions are executed in the CSP mode by default. If you want such an instruction to be executed in the CSV mode, you must use the MC_SetControllerMode instruction to switch the control mode.

  First, configure the PDOs: Select the first group of input parameters and configure the 606C. Select the output parameters other than the first and second group of output parameters, or select the first group of output parameters and add the 60FF.

  Call a motion control instruction. After the instruction is executed, the parameter Busy serves as the trigger condition for the MC_SetControllerMode instruction. When the ControllerMode is set to CSV, the control mode can be switched after the MC_MoveAbsolute instruction is triggered.



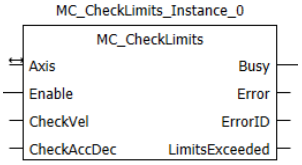
- **Precautions**

  It is not recommended to execute positioning instructions in the CSV mode, as it may cause significant deviation.

18.3.3.3.23 MC_CheckLimits

This instruction checks if the reference values for the current axis exceed the upper limits configured in the controller.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_CheckLimits	Check axis reference values against upper limits		MC_CheckLimits_Instance_0(Axis :=, Enable :=, CheckVel :=, CheckAccDec :=, Busy =>, Error =>, ErrorID =>, LimitsExceeded =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled. FALSE: Disabled.
CheckVel	Check velocity	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled. FALSE: Disabled.
CheckAccDec	Check acceleration/ deceleration	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled. FALSE: Disabled.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.
LimitsExceeded	Limits exceeded	BOOL	TRUE, FALSE	FALSE	TRUE if the velocity or acceleration/deceleration reference exceeds the upper limit setpoint.

- **Function**

  Bus-controlled axis:

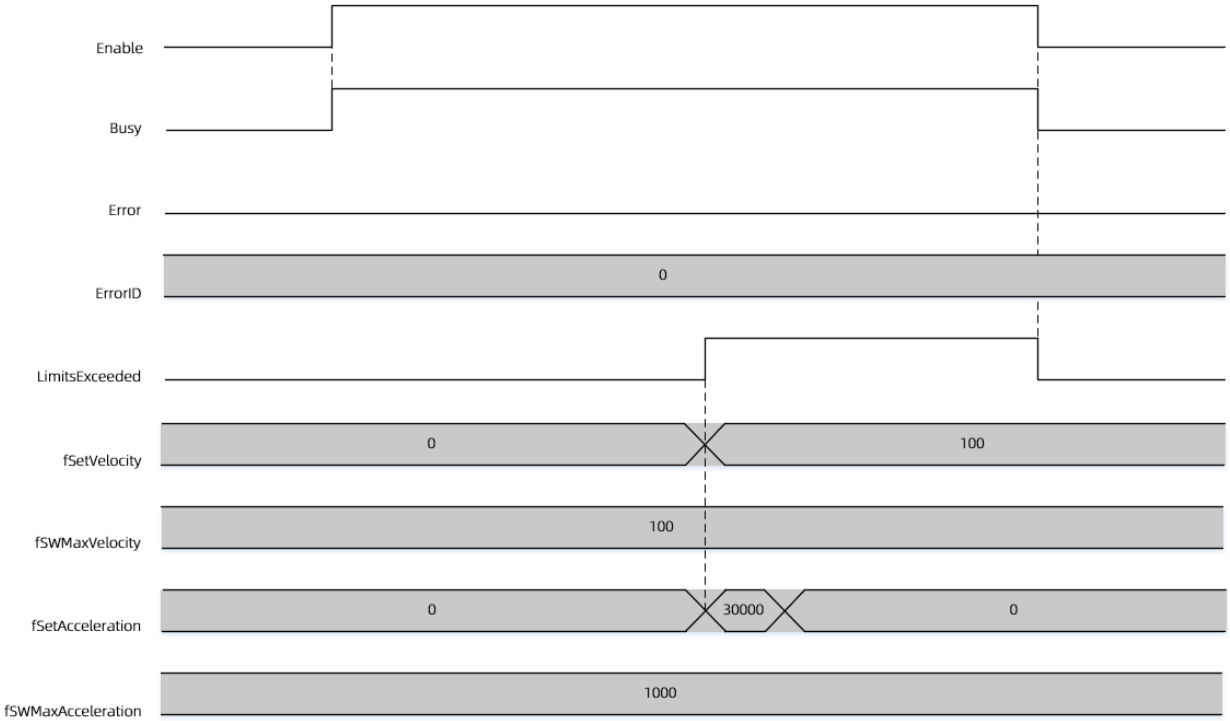
  - This instruction only checks if the velocity or acceleration/deceleration reference exceeds the upper limit setpoint in the software and does not stop the axis movement.
  - When the input parameter CheckVel is set to TRUE, this instruction checks if the velocity reference exceeds the upper limit.
  - When the input parameter CheckAccDec is set to TRUE, this instruction checks if the acceleration/deceleration reference exceeds the upper limit.
  - When input parameters CheckVel and CheckAccDec are set to TRUE, and either the velocity or acceleration/deceleration reference exceeds the upper limit, the output parameter LimitsExceeded flag will be set. This flag is refreshed periodically. If the velocity and acceleration/deceleration reference in the next cycle do not exceed the upper limits, the flag will be reset automatically.

  Local pulse axis:

  - This instruction can be used for local pulse axes.

- **Timing diagram**

  When the input parameter Enable is set to TRUE, the output parameter Busy shows the value TRUE. If the velocity or acceleration/deceleration reference exceeds the upper limit setpoint, the output parameter LimitsExceeded will be set to TRUE.



- **Re-execution of motion control instructions**

  This instruction cannot be re-executed.

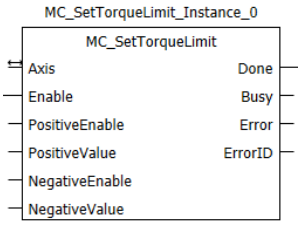
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.24 MC_SetTorqueLimit

This instruction limits the axis torque output using the torque limiting function of the servo drive.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_SetTorqueLimit	Set upper torque limit		MC_SetTorqueLimit _Instance_0(Axis :=, Enable :=, PositiveEnable :=, PositiveValue :=, NegativeEnable :=, NegativeValue :=, Done =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled. FALSE: Disabled.
PositiveEnable	Positive enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled. FALSE: Disabled.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
PositiveValue	Positive torque limit	LREAL	0.1 to 1000	0	Set the positive torque limit, in increments of 0.1%. If the value is greater than the axis parameter PositiveValue, the positive torque limit will be clamped to the PositiveValue.
NegativeEnable	Negative enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled. FALSE: Disabled.
NegativeValue	Negative torque limit	LREAL	0.1 to 1000	0	Set the negative torque limit, in increments of 0.1%. If the value is greater than the axis parameter NegativeValue, the negative torque limit will be clamped to the NegativeValue.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus-controlled axis:

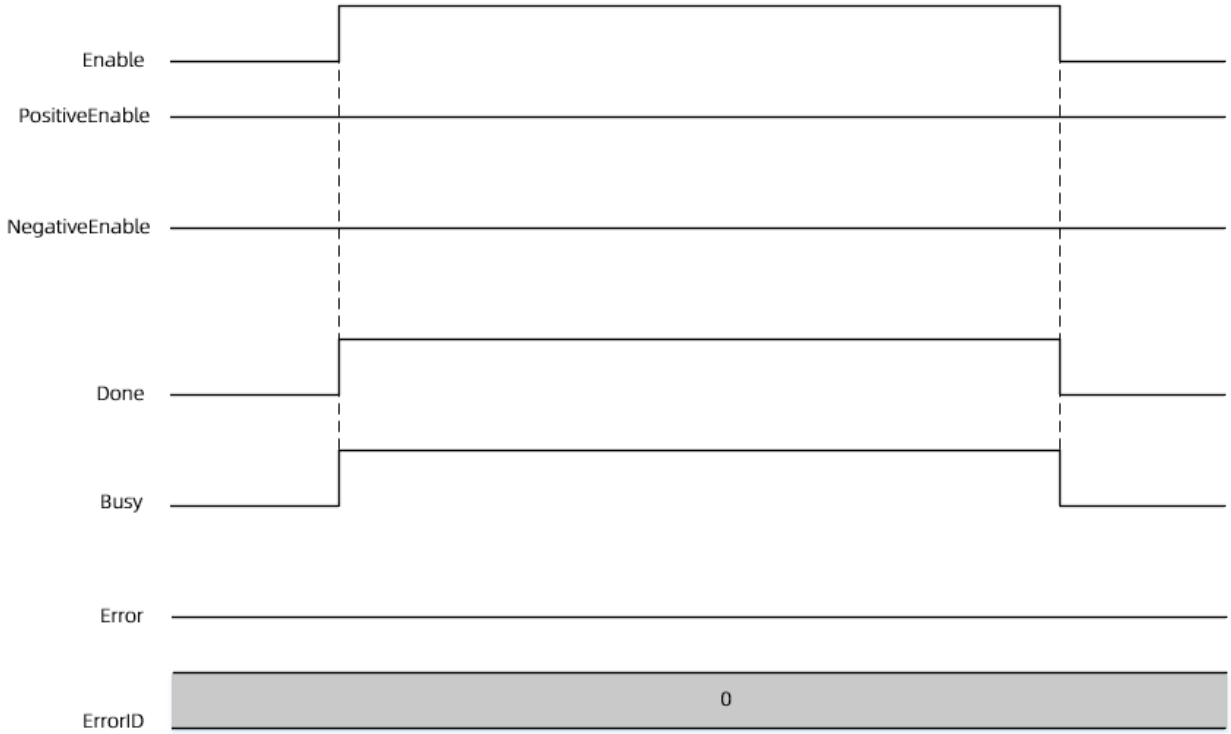
  - This instruction sets the torque limit of an axis.
  - When the parameter Enable is set to TRUE, the limit will be the setpoint of the input parameter PositiveValue if the input parameter PositiveEnable shows the value TRUE; the limit will be the setpoint of the input parameter NegativeValue if the input parameter NegativeEnable shows the value TRUE.
  - When the PositiveEnable is set to FALSE, the PositiveValue will not take effect, and the axis torque limit will return to the setpoint in the software. Similarly, when the NegativeEnable is set to FALSE, the NegativeValue will not take effect, and the axis torque limit will return to the setpoint in the software.
  - When the parameter Enable is set to FALSE, you can specify the [PositiveValue] and [NegativeValue] on the servo drive, and set the Busy to FALSE.
  - If the set value of PositiveValue or NegativeValue is less than or equal to 0, it will be treated as 0. If the setpoint of PositiveValue or NegativeValue is greater than or equal to the setpoint in the software, it will be treated as setpoint in the software.
  - If the control object is a bus-controlled axis, the PDOs 60E0 (PositiveValue) and 60E1 (NegativeValue) must be configured; otherwise, the execution of this instruction will result in an error.

  Local pulse axis:

  - This instruction cannot be used for local pulse axes.

- **Timing diagram**

  When the input parameter Enable is set to TRUE, the output parameter Busy shows the value TRUE. When the axis torque limit is specified during execution, the output parameter Done shows the value TRUE.



- **Re-execution of motion control instructions**

  You cannot re-execute motion control instructions with an input parameter Enable.

- **Multiple execution of motion control instructions**

  When multiple MC_SetTorqueLimit instructions are triggered simultaneously, the parameters of the last triggered instruction take effect.

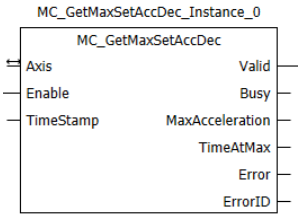
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.25 MC_GetMaxSetAccDec

This instruction reads the maximum acceleration/deceleration of an axis and records the time at which the maximum acceleration/deceleration is reached.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GetMaxSetAccDec	Read maximum acceleration/ deceleration		MC_GetMaxSetAccDec_Instance_0(Axis :=, Enable :=, TimeStamp :=, Valid =>, Busy =>, MaxAcceleration =>, TimeAtMax =>, Error =>, ErrorID =>);

Related variables

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled. FALSE: Disabled.
TimeStamp	Time stamp	DWORD	-	-	Optional time reference source (for example, varying with the task cycle or the fixed interval).

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Valid output	BOOL	TRUE, FALSE	FALSE	TRUE when there is a valid output from the function block.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
MaxAcceleration	Maximum acceleration/ deceleration	LREAL	Positive number, negative number, or 0	0	Maximum value of acceleration /deceleration [u/s^2] (positive value for acceleration, negative value for deceleration; the absolute maximum value of acceleration /deceleration is the final value).
TimeAtMax	Time stamp at the maximum acceleration/ deceleration	DWORD	-	0	This value is continuously updated while the acceleration/ deceleration is increasing and the recorded time stamp is returned when the maximum acceleration/ deceleration is reached.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18 .3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus-controlled axis:

  - If the parameter Enable is set to TRUE and there are no faults, the instruction will start reading the maximum acceleration/deceleration of an axis with the parameter Valid being set.

- If the absolute value of acceleration/deceleration is greater than the recorded value in the previous cycle, the MaxAcceleration and TimeAtMax will be refreshed continuously. The value of TimeAtMax corresponds to the value of TimeStamp when the maximum acceleration/deceleration is reached. Therefore, the parameter TimeStamp can be specified as a variable, such as a count value varying with the task cycle or the fixed interval.
- When the axis is in abnormal communication, this instruction issues a fault message with a fault code.

Local pulse axis:

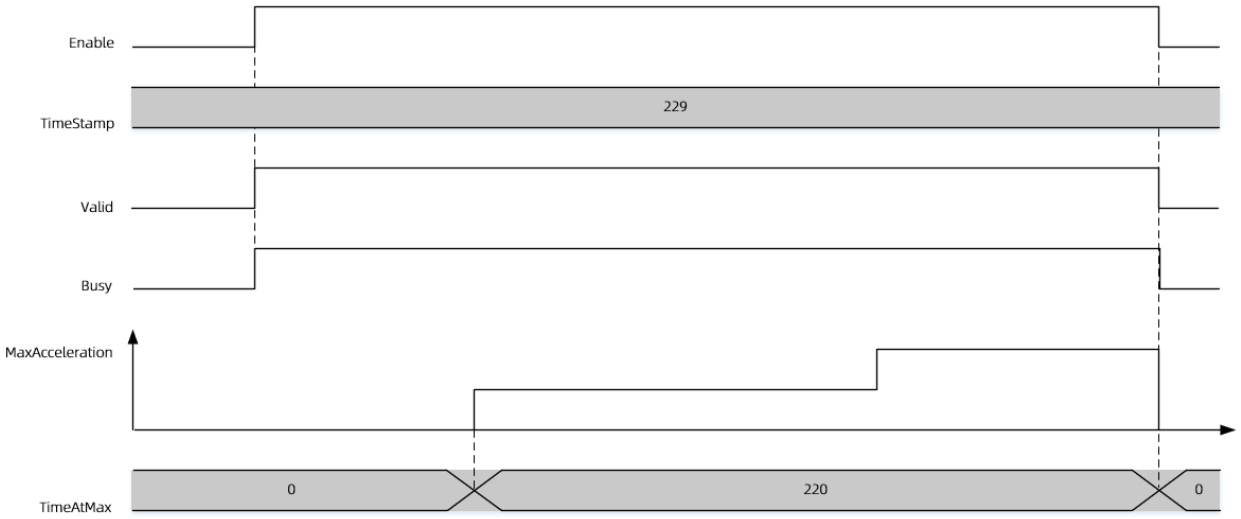
- This instruction can be used for local pulse axes.

- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Valid of the function block indicates valid outputs, and the parameter Busy indicates that the current function block is being executed.

  When the input parameter Enable is set to TRUE, the output parameter Valid shows the value TRUE and this instruction checks the maximum acceleration/deceleration of the axis and records the time at which the maximum acceleration/deceleration is reached. If the absolute value of acceleration/deceleration is greater than the recorded value, the MaxAcceleration and TimeAtMax will be refreshed.

  The value of TimeAtMax corresponds to the value of TimeStamp when the maximum velocity is reached. Therefore, the parameter TimeStamp can be specified as a variable, such as a count value varying with the task cycle or the fixed interval.



- **Re-execution of motion control instructions**

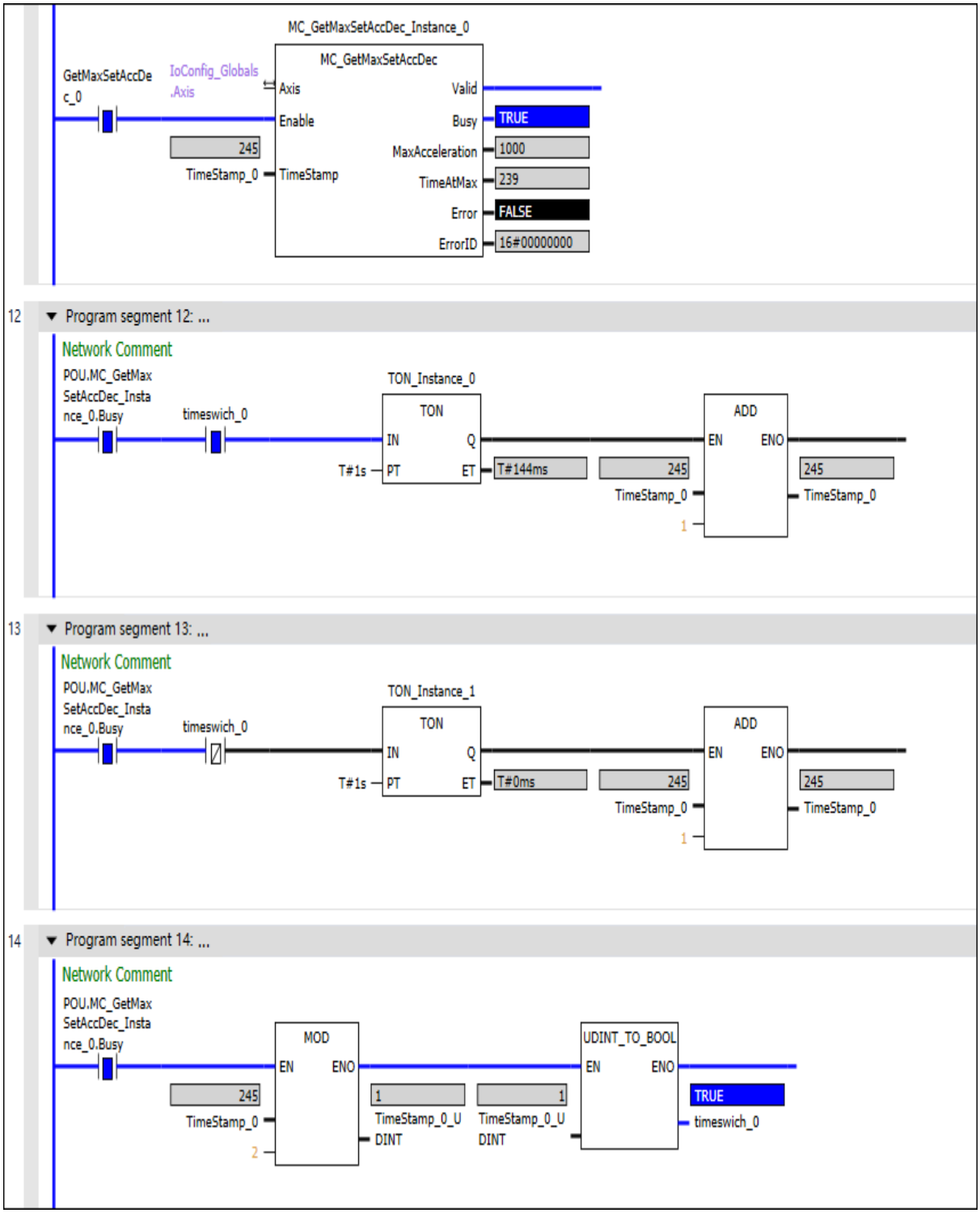
  This instruction cannot be re-executed.

- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

- **Example**

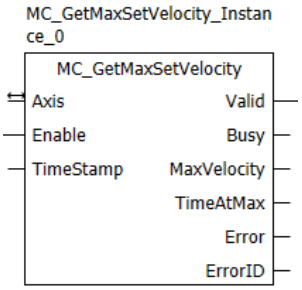
  You can configure DWORD data that increases in a cycle and associate it with TimeStamp. In the following example, the timeStamp_0 that increases by 1 every second is associated with the input terminal TimeStamp of this instruction. When the instruction detects the maximum velocity, it records the maximum acceleration and the corresponding TimeStamp on the output terminals MaxAcceleration and TimeAtMax. You can check the time when the maximum acceleration is reached.



18.3.3.3.26 MC_GetMaxSetVelocity

This instruction reads the maximum velocity of an axis and records the time at which the maximum velocity is reached.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GetMaxSetVelocity	Read maximum velocity		MC_GetMaxSetVelocity_Instance_0(Axis :=, Enable :=, TimeStamp :=, Valid =>, Busy =>, MaxVelocity =>, TimeAtMax =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled. FALSE: Disabled.
TimeStamp	Time stamp	DWORD	-	-	Optional time reference source (for example, varying with the task cycle or the fixed interval).

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Valid output	BOOL	TRUE, FALSE	FALSE	TRUE when there is a valid output from the function block.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
MaxVelocity	Maximum velocity	LREAL	-	0	Maximum value of velocity [u/s] (positive value for acceleration, negative value for deceleration; the absolute maximum value of velocity is the final value).
TimeAtMax	Time stamp at the maximum acceleration/ deceleration	DWORD	-	0	This value is continuously updated while the velocity is increasing and the recorded time stamp is returned when the maximum velocity is reached.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus-controlled axis:

  - If the parameter Enable is set to TRUE and there are no faults, the instruction will start detecting the maximum velocity of an axis with the parameter Valid being set.
  - If the absolute value of velocity is greater than the recorded value, the MaxVelocity and TimeAtMax will be refreshed continuously. The value of TimeAtMax corresponds to the value of TimeStamp when the maximum velocity is reached. Therefore, the parameter TimeStamp can be specified as a variable, such as a count value varying with the task cycle or the fixed interval.
  - When the axis is in abnormal communication, this instruction issues a fault message with a fault code.

Local pulse axis:

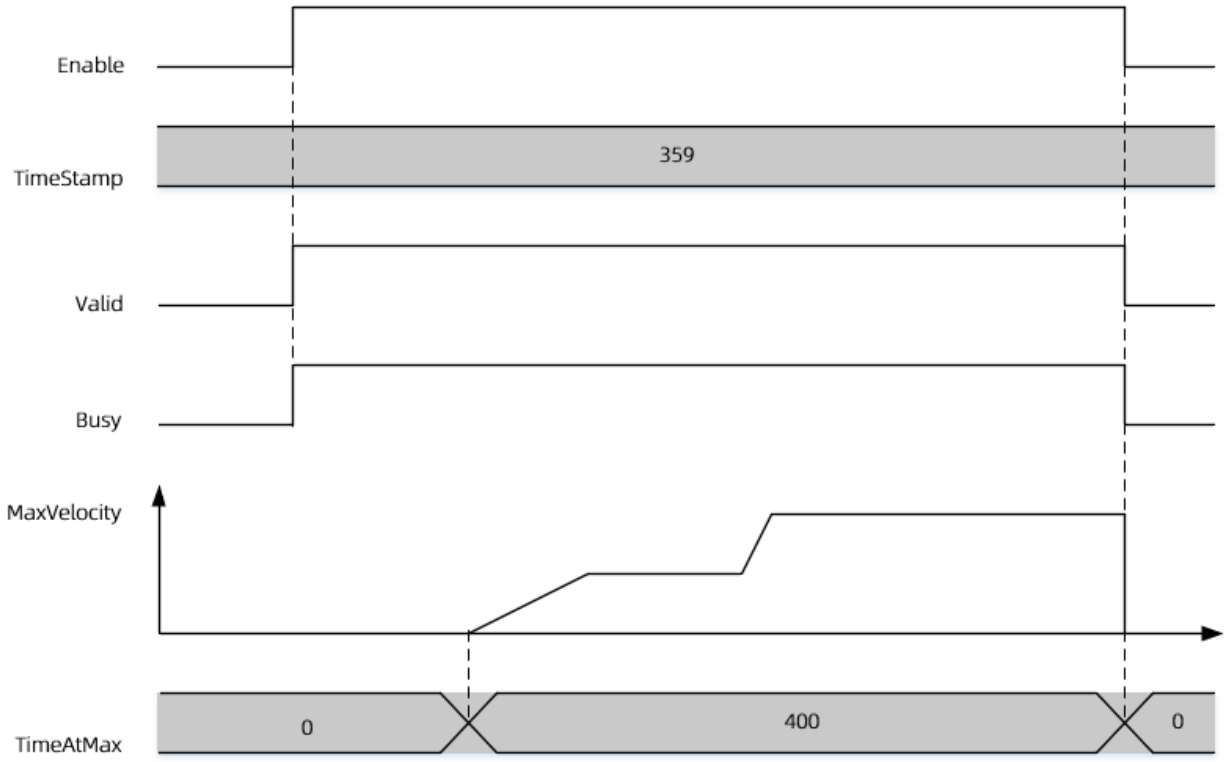
- This instruction can be used for local pulse axes.

- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Valid of the function block indicates valid outputs, and the parameter Busy indicates that the current function block is being executed.

  When the input parameter Enable is set to TRUE, the output parameter Valid shows the value TRUE. This instruction reads the maximum velocity of an axis and records the time at which the maximum velocity is reached. If the absolute value of velocity is greater than the recorded value, the MaxVelocity and TimeAtMax will be refreshed.

  The value of TimeAtMax corresponds to the value of TimeStamp when the maximum velocity is reach ed. Therefore, the parameter TimeStamp can be specified as a variable, such as a count value varying with the task cycle or the fixed interval.



- **Re-execution of motion control instructions**

  This instruction cannot be re-executed.

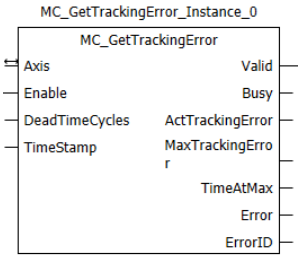
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.27 MC_GetTrackingError

This instruction measures the actual or maximum tracking error.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GetTrackingError	Measure actual or maximum tracking error		MC_GetTrackingError_Instance_0(Axis :=, Enable :=, DeadTimeCycles :=, TimeStamp :=, Valid =>, Busy =>, ActTrackingError =>, MaxTrackingError =>, TimeAtMax =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled. FALSE: Disabled.
DeadTimeCycles	Dead time cycles	BYTE	-	2	Delay duration before starting lag detection after the Enable trigger.
TimeStamp	Time stamp	DWORD	-	-	Optional time reference source (for example, varying with the task cycle or the fixed interval).

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Valid output	BOOL	TRUE, FALSE	FALSE	TRUE when there is a valid output from the function block.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
ActTrackingError	Actual tracking error	LREAL	-	0	Actual deviation (positive value for delay, negative value for lead; the absolute maximum value is the final value).
MaxTrackingError	Maximum tracking error	LREAL	-	0	Maximum deviation (positive value for delay, negative value for lead; the absolute maximum value is the final value).
TimeAtMax	Time stamp at the maximum tracking error	DWORD	-	0	Time stamp at the maximum deviation.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus-controlled axis:

  - When the TimeStamp is less than the DeadTimeCycles, no tracking error is recorded. The maximum tracking error is recorded after the TimeStamp exceeds the DeadTimeCycles. When the tracking error is greater than the recorded value, the MaxTrackingError and TimeAtMax are refreshed continuously. The

parameter value of TimeAtMax corresponds to the value of TimeStamp when the maximum tracking error is detected. Therefore, the parameter TimeStamp can be specified as a variable, such as a count value varying with the task cycle or the fixed interval.

- In CSV and CST modes, the tracking error remains 0 when there is no position reference output.
- When the axis is in abnormal communication, this instruction issues a fault message with a fault code.
- This instruction can be used for bus-controlled axes.

Local pulse axis:

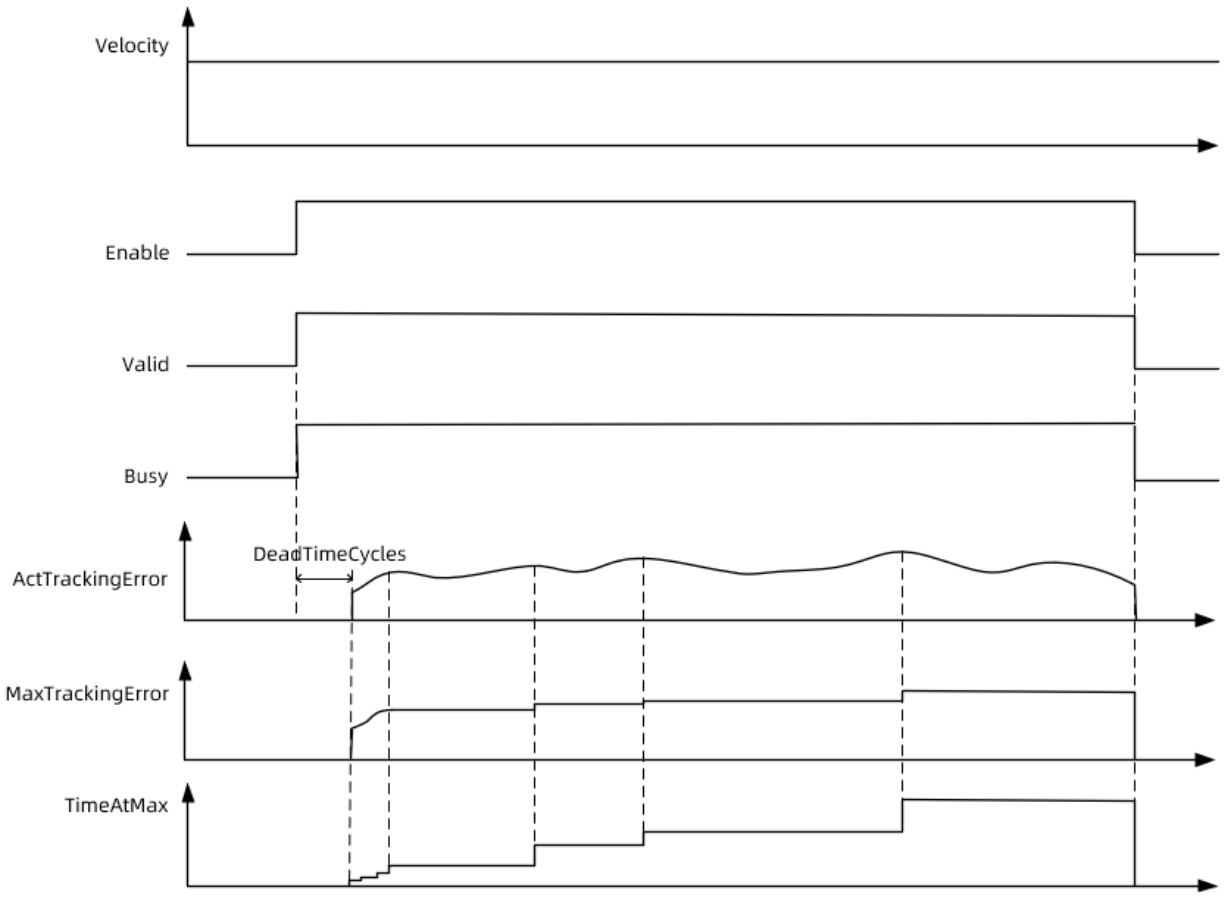
- This instruction cannot be used for local pulse axes.

- **Timing diagram**

  When the input parameter Enable is set to TRUE, the output parameter Valid shows the value TRUE and this instruction measures the actual or maximum tracking error.

  When the absolute deviation is greater than the recorded value, the MaxTrackingError and TimeAtMax are refreshed.

  The value of TimeAtMax corresponds to the value of TimeStamp when the maximum tracking error is detected. Therefore, the parameter TimeStamp can be specified as a variable, such as a count value varying with the task cycle or the fixed interval.



- **Re-execution of motion control instructions**

  This instruction cannot be re-executed.

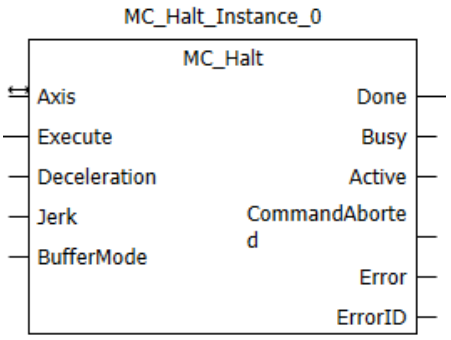
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.28 MC_Halt

This instruction pauses an axis in normal motion.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_Halt	Pause axis		MC_Halt _Instance_0(Axis :=, Execute :=, Deceleration :=, Jerk :=, BufferMode :=, Done =>, Busy =>, Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.
Deceleration	Deceleration	LREAL	Positive number	0	Specify the deceleration [u/s^2] for an axis to decelerate.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Jerk	Jerk	LREAL	Positive number	0	Slope of deceleration [u/s^3]. Note: This parameter is ignored when the T-curve is specified in the software; This parameter must be greater than 0 when the S-curve is specified in the software.
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting 1: mcBuffered	0	0: mcAborting (Aborting). 1: mcBuffered (Buffered).

- **Output variables**

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Controlling	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs in the function block.

- **Parameter correction**

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the deceleration limit.	Any deceleration less than 0 will be automatically corrected to 10% of the deceleration limit.	The deceleration equal to 0 will be automatically corrected to 10% of the deceleration limit.
Jerk	Jerk	If the motion profile is set to S-curve, any jerk that exceeds the software limit will be automatically corrected to the jerk limit.	If the motion profile is set to S-curve, any jerk less than 0 will be automatically corrected to 10% of the jerk limit.	If the motion profile is set to S-curve, the jerk equal to 0 will be automatically corrected to 10% of the jerk limit.

- **Function**

  Bus-controlled axis:

  - This instruction pauses an axis in normal motion. Triggering other motion control instructions abort the execution of this instruction.
  - During the execution of this instruction, do not modify the input parameter BufferMode to a mode other than MC_ABORTING before re-executing the instruction; otherwise, the function block reports a fault and the axis decelerates to stop according to the set motion parameters in the software.
  - This instruction does not support BLENDING mode. If the input parameter BufferMode is set improperly, the function block will report a fault and the axis will decelerate to stop under a T-curve planning, using the maximum deceleration limit in the software.
  - When the axis state is AXIS_ERRORSTOP, triggering this instruction causes the function block to report a fault. When the axis state is AXIS_HOMING or AXIS_STOPPING, the MC_halt instruction is aborted. During the execution of this instruction, the axis state is AXIS_DISCRETE_MOTION; after completion of this instruction, the axis state is AXIS_STANDSTILL.
  - The axis type must be control axis.
  - In CSV mode, this instruction is used to halt the motion of the axis, while the servo axis remains in CSV mode.

  Local pulse axis:

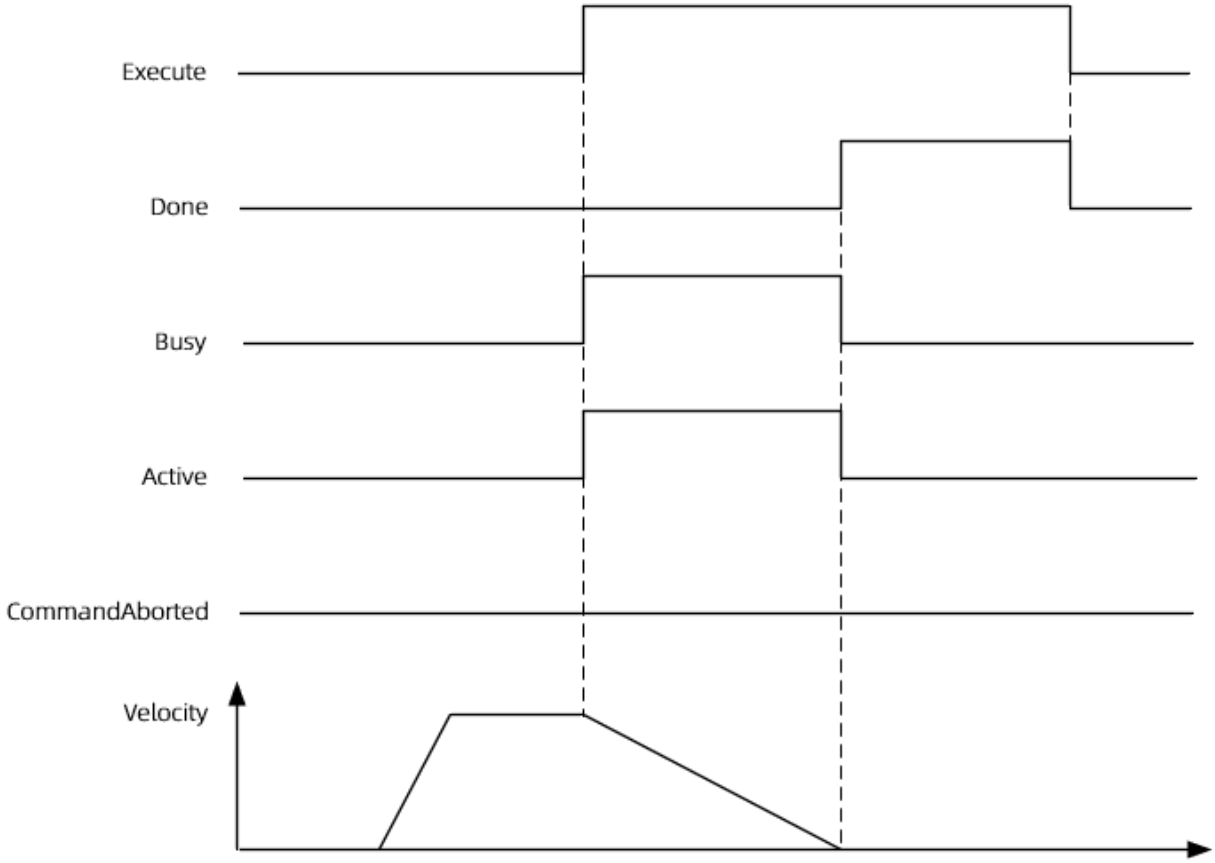
  This instruction can be used for local pulse axes.

- **Timing diagram**

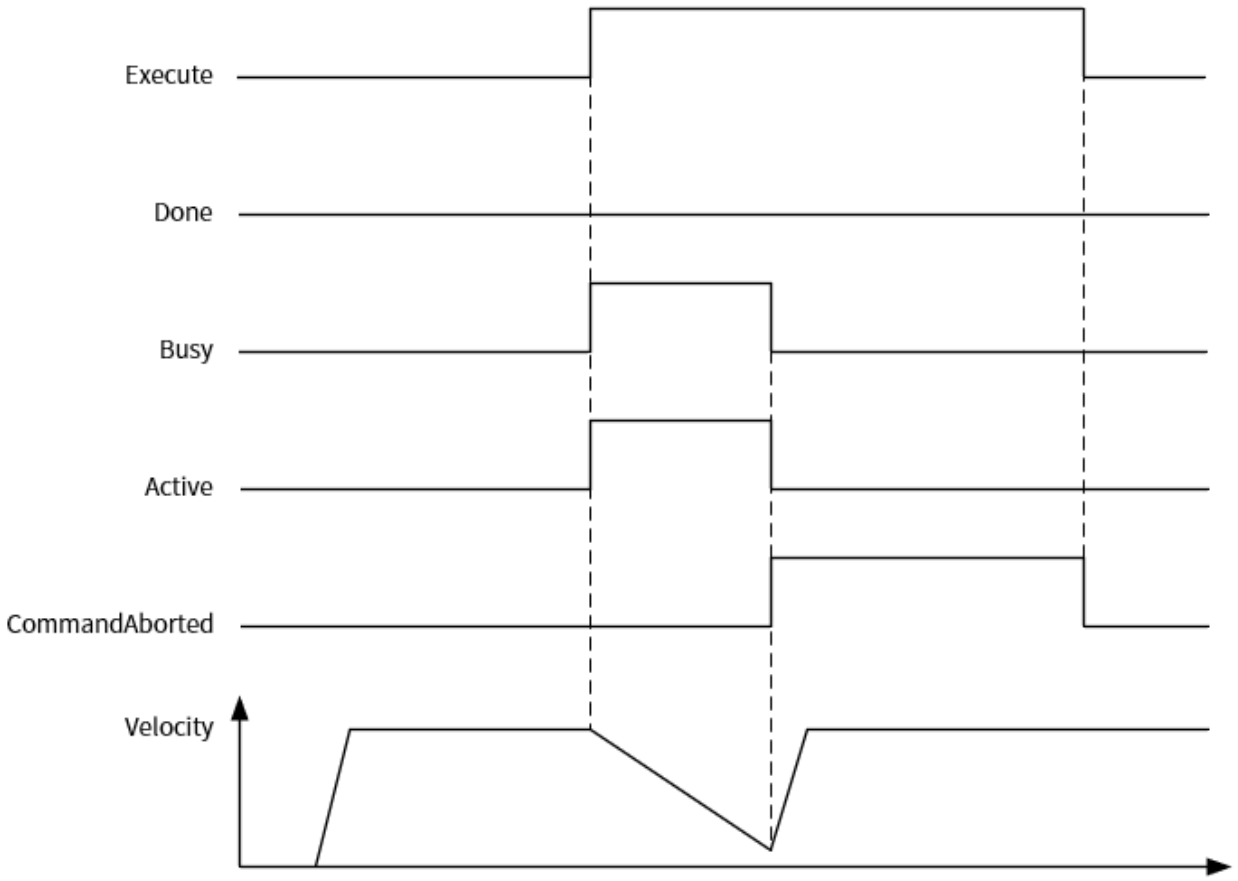
The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates the instruction has been executed successfully; the parameter Busy indicates that the current function block is being executed; the parameter CommandAborted indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

The parameter Busy shows the value TRUE on a rising edge of Execute. In the next cycle, the output parameter Active shows the value TRUE.

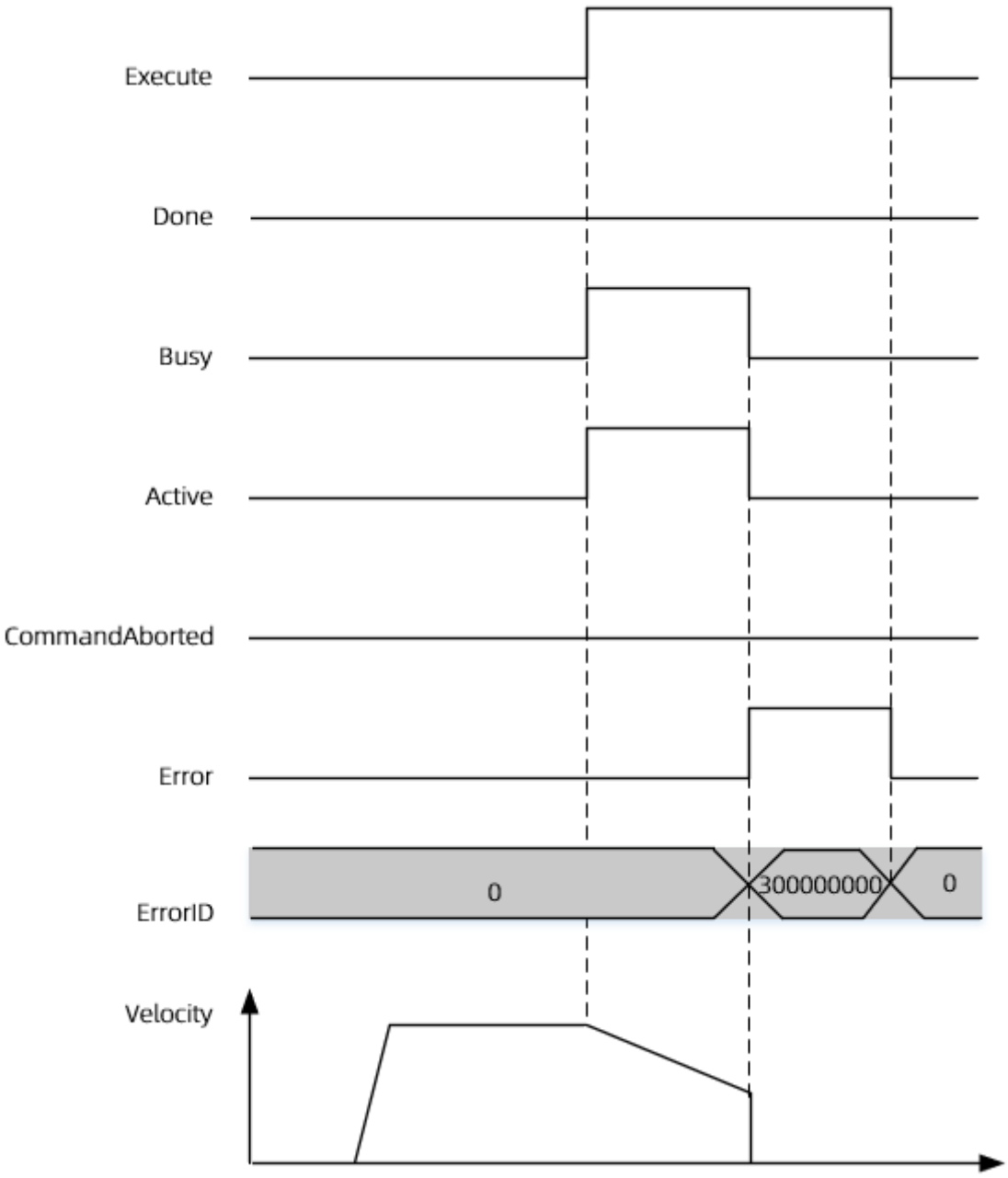
◦ After the MC_MoveVelocity instruction is called, trigger the MC_Halt instruction.



◦ After the MC_Halt instruction is triggered, re-call the MC_MoveVelocity instruction to abort the MC_Halt instruction.



◦ The drive stops upon a fault during execution of the MC_Halt instruction.



- **Re-execution of motion control instructions**

  The re-executed instruction will perform planning according to new motion parameters, and the current buffered instructions will be cleared.

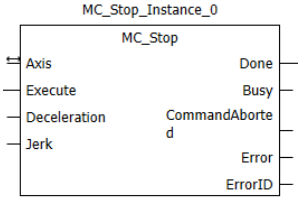
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.29 MC_Stop

This instruction decelerates an axis to stop.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_Stop	Stop		MC_Stop _Instance_0(Axis :=, Execute :=, Deceleration :=, Jerk :=, Done =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

  Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.
Deceleration	Deceleration	LREAL	Positive number	0	Specify the deceleration [u /s^2] for an axis to decelerate.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Jerk	Jerk	LREAL	Positive number	0	Slope of deceleration [u /s^3]. Note: This parameter is ignored when the T-curve is specified in the software; This parameter must be greater than 0 when the S-curve is specified in the software.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs in the function block.

- **Parameter correction**

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the deceleration limit.	Any deceleration less than 0 will be automatically corrected to 10% of the deceleration limit.	The deceleration equal to 0 will be automatically corrected to 10% of the deceleration limit.
Jerk	Jerk	If the motion profile is set to S-curve, any jerk that exceeds the software limit will be automatically corrected to the jerk limit.	If the motion profile is set to S-curve, any jerk less than 0 will be automatically corrected to 10% of the jerk limit.	If the motion profile is set to S-curve, the jerk equal to 0 will be automatically corrected to 10% of the jerk limit.

- **Function**

  Bus-controlled axis:

  - This instruction controls the deceleration of an axis. If the axis state is AXIS_POWEROFF or AXIS_ERROR STOP, or the single axis being operated is bound to an axis group, triggering this instruction will cause the function block to report a fault.
  - On a rising edge of Execute, the axis decelerates from the current velocity to 0 according to the input parameters Deceleration and Jerk; once the axis is standstill, the output parameter Done is set.
  - During the execution of this instruction, the axis state is AXIS_STOPPING. After this instruction is done, the axis state remains AXIS_STOPPING. If you want to switch the axis state to AXIS_STANDSTILL, reset the input parameter Execute of the function block.
  - This instruction can be aborted by the MC_ImmediateStop instruction only. When the instruction is aborted during execution, the output parameter CommandAborted is set.
  - During the execution of this instruction, you can re-execute it after modifying parameters. If the instruction is triggered while the axis state is AXIS_STANDSTILL, the axis state will switch to AXIS_STOPPING with the output parameter Done being set.
  - During the execution of this instruction, do not execute another instance of this instruction; otherwise, the subsequently triggered function block will issue a fault message.
  - After the instruction is done, the axis is in the Stopping state. If another MC_Stop instruction is triggered, the subsequently triggered function block will issue a fault message.
  - The axis controlled by this instruction must be control axis.
  - If you want to switch the control mode to CSP mode with the PDO, ensure that 6060 and 6061 in the object dictionary are correctly configured; otherwise, switch the control mode to CSP mode with the SDO.

  Local pulse axis:

  - This instruction can be used for local pulse axes.

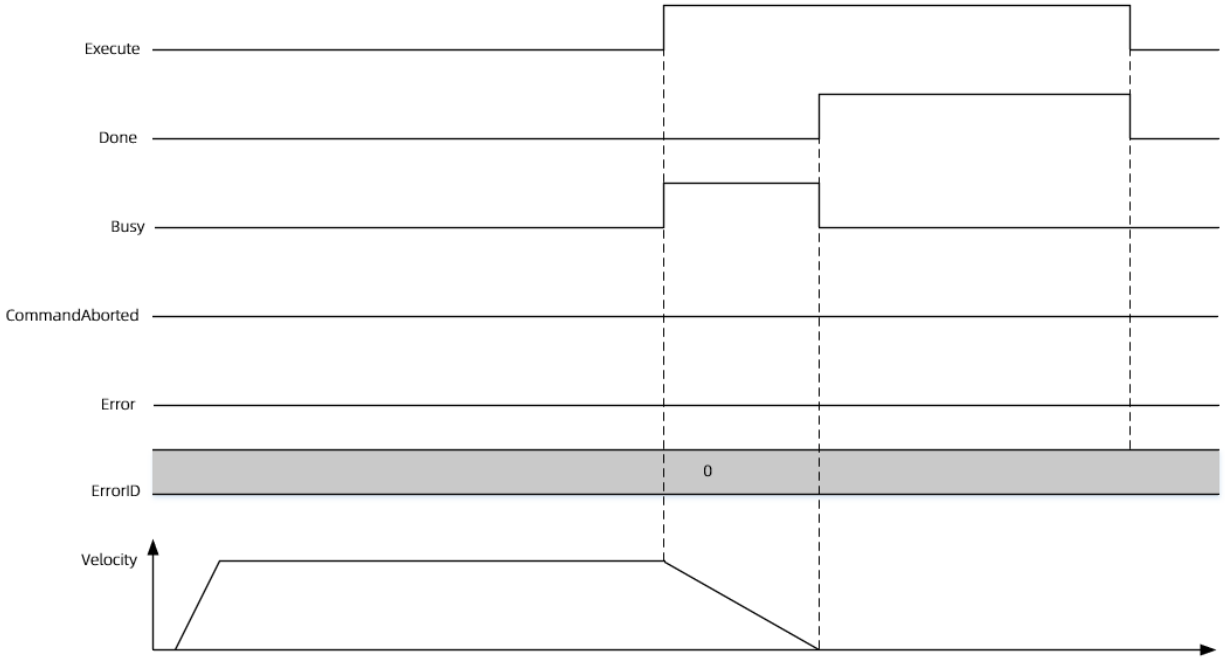
- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates the instruction has been executed successfully; the parameter Busy indicates that the current function block is being executed; the parameter CommandAborted

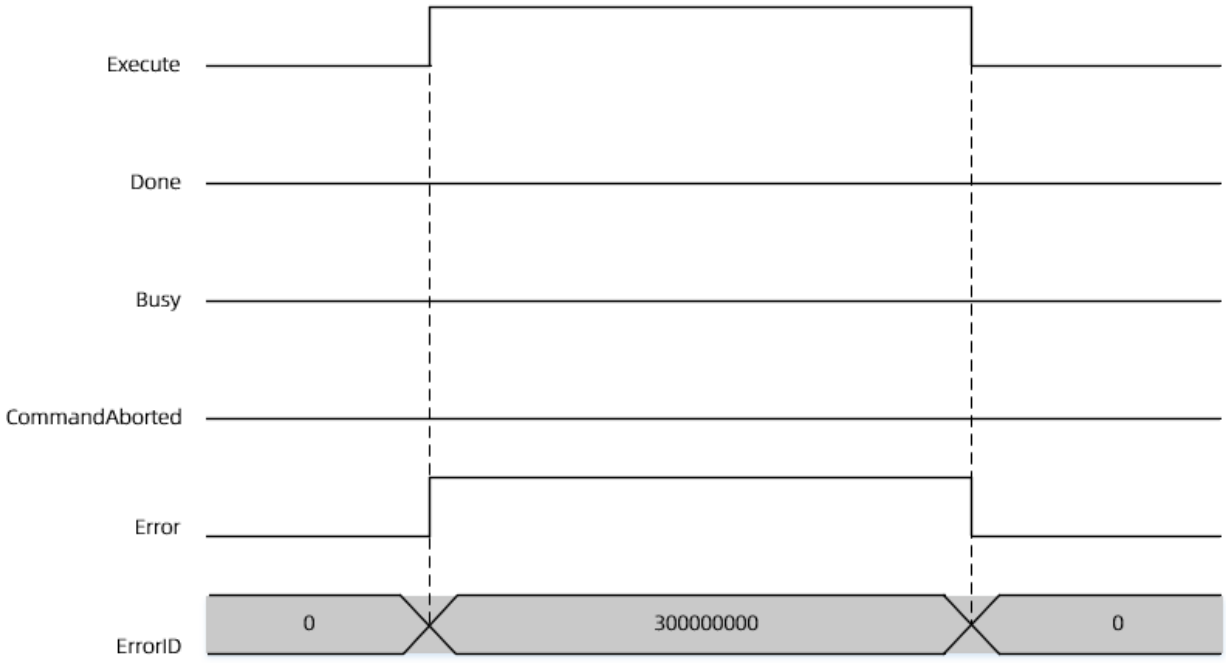
indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

The parameter Busy shows the value TRUE on a rising edge of Execute. In the next cycle, the output parameter Active shows the value TRUE.

- After the MC_MoveVelocity instruction is called, trigger the MC_Stop instruction.



- The drive malfunctions during the instruction execution.



- **Multiple execution of motion control instructions**
  - When multiple MC_Stop instructions are executed simultaneously, the first instruction takes effect, while subsequent instructions are aborted.
  - If the current axis state is AXIS_STOPPING and the MC_Stop instruction is executed, the output parameter CommandAborted of the instruction will be set, without affecting the active MC_Stop instruction or the deceleration stop process.

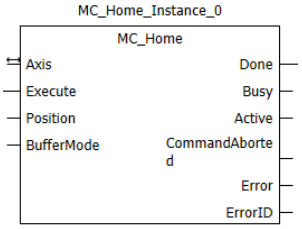
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.30 MC_Home

This instruction operates the drive to define the home.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_Home	Drive homing		MC_Home _Instance_0(Axis :=, Execute :=, Position :=, BufferMode :=, Done =>, Busy =>, Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

  Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction.
Position	Position	LREAL	Positive number, negative number, or 0	0	Target position [u] in absolute coordinates.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
BufferMode	Buffer mode (reserved)	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting 1: mcBuffered	0	0: mcAborting (Aborting). 1: mcBuffered (Buffered).

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Controlling	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled by the current FB.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus-controlled axis:

  - This instruction can be triggered only when the axis state is AXIS_STANDSTILL. During the execution of the instruction, the axis state is AXIS_HOMING. After the instruction is done, the axis state switches to AXIS_STANDSTILL.
  - On a rising edge of Execute, this instruction records the value of the input parameter Position, which is used for defining the home offset.
  - Configure the PDOs 6060 and 6061 before calling this instruction; otherwise, switch the control mode with the SDO.
  - When the axis state is AXIS_ERRORSTOP, homing cannot be performed, and a fault message will appear if an attempt is made to execute this instruction.

- This instruction can be aborted by the MC_Stop or MC_ImmediateStop instruction only during execution.
- This instruction cannot be re-called. When this instruction is triggered and the axis state switches to AXIS_HOMING, calling other MC_Home instructions result in a fault, without affecting the ongoing homing.
- If homing is aborted by the MC_ImmediateStop instruction, the control mode will not be directly switched to CSP mode. You must call the MC_Reset instruction to clear the axis fault and then switch the control mode.
- The axis type must be control axis.
- Software limits have no effect on this instruction.
- Frequently triggering this instruction may bring warnings regarding frequently reading single data from the servo drive. Taking the SV660N servo drive as an example, you can set bit 6 of H0A.71 to 1 to disable such a warning. For other servo drives, see their user guides for details.
- The arrival check function is only applicable to MotionControl library version 2.4.0.0 or later. Based on the "Positioning range" and "Positioning check time" parameters in the "General setting" of the axis configuration page, it checks in real time whether the actual position has entered the specified target range. After the instruction output for the target position is completed, positioning is complete and the Done output is set if the difference between the actual position and the target position is smaller than the "Positioning range". If positioning is not completed within the "Positioning check time", a fault will occur.

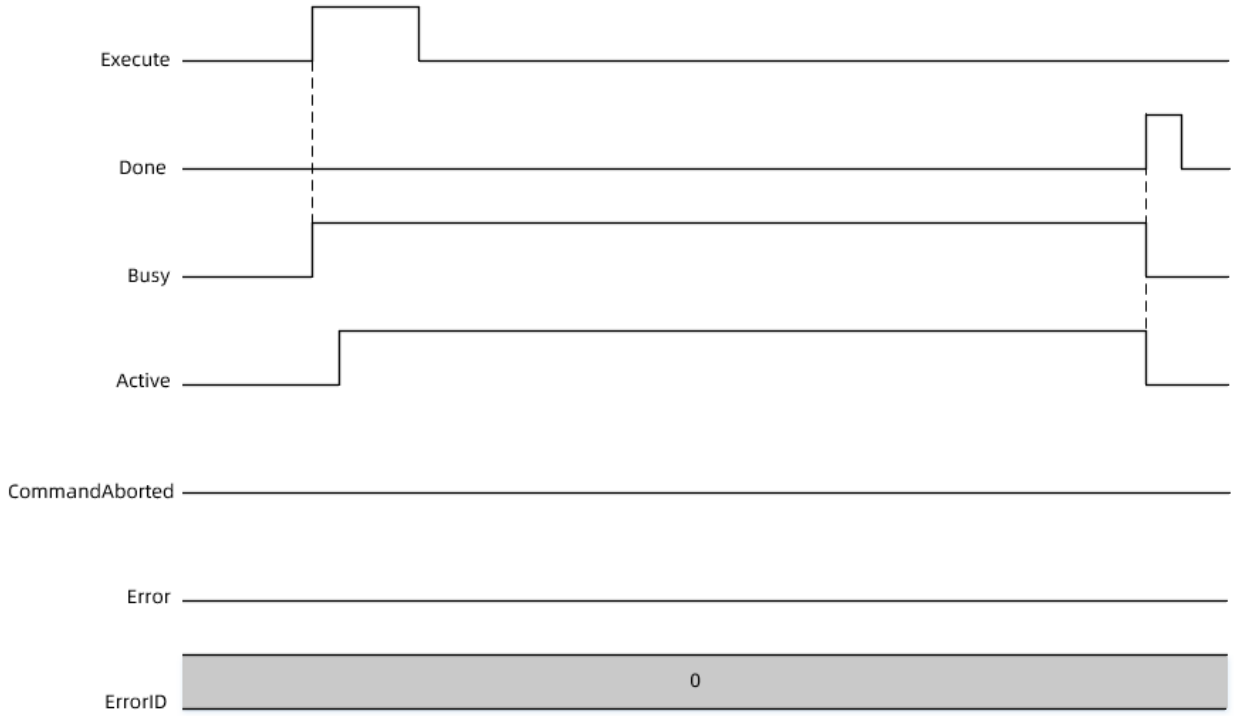
  Local pulse axis:

- This instruction can be used for local pulse axes.

- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates the instruction has been executed successfully; the parameter Busy indicates that the current function block is being executed; the parameter CommandAborted indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

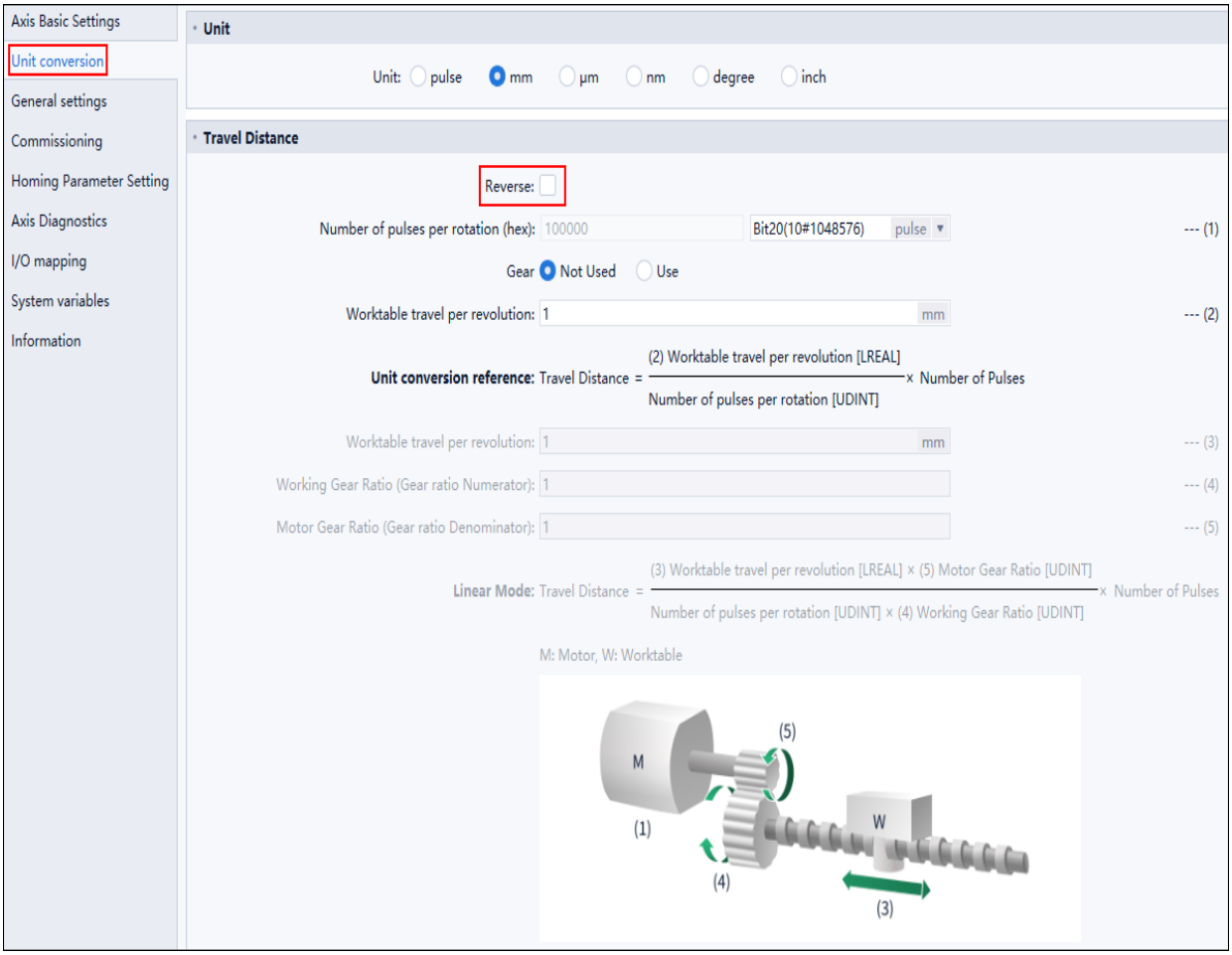
  After this instruction is called, the output parameter Busy is set to TRUE on a rising edge of Execute In the next cycle, the output parameter Active shows the value TRUE. After homing is done, the output parameter Done is set to TRUE.



- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

- **Precautions**
  - This instruction will cause slight jitter in the axis position reference due to control mode switching if it is executed immediately after a positioning instruction or a relative move instruction; if the axis is the master axis of a synchronization instruction, the slave axis velocity may experience a sudden change.
  - Checking Reverse on the interface has no effect on the MC_Home instruction.

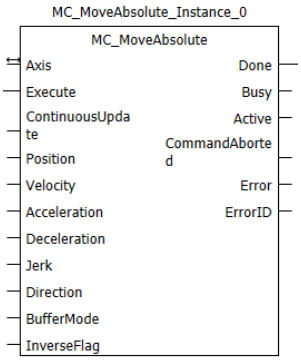


- After the servo drive homing is done, there may be jitter. If the Direction of the subsequent instruction is set to Current, it may be inconsistent with the planned direction at the end of the homing. It is recommended that users define proper motion direction when using the subsequent instruction.

18.3.3.3.31 MC_MoveAbsolute

This instruction moves an axis to a specified absolute target position.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_MoveAbsolute	Absolute positioning		MC_MoveAbsolute_Instance_0(Axis :=, Execute :=, ContinuousUpdate :=, Position :=, Velocity :=, Acceleration :=, Deceleration :=, Jerk :=, Direction :=, BufferMode :=, InverseFlag :=, Done =>, Busy =>, Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Continuous Update	Continuous update	BOOL	TRUE, FALSE	FALSE	Continuous parameter update effective flag. Prerequisite: The parameter ContinuousUpdate shows the value TRUE and the parameter Active shows the value TRUE. Effective parameters: Position, Velocity. Note: The modified parameter values must differ from the original parameter values by more than 0.01.
Position	Position	LREAL	Positive number, negative number, or 0	0	Target position [u] in absolute coordinates.
Velocity	Velocity	LREAL	Positive number	0	Maximum velocity [u/s], which may not be reached (cannot be zero).
Acceleration	Acceleration	LREAL	Positive number	0	Acceleration [u/s^2] of the acceleration phase.
Deceleration	Deceleration	LREAL	Positive number	0	Deceleration [u/s^2] of the deceleration phase.
Jerk	Jerk	LREAL	Positive number	0	Slope [u/s^3] of acceleration or deceleration. (Only valid for S-curve planning).
Direction	Motion direction	18.3.3.2.2.9 MC_DIRECTION: Motion Direction Enumeration	0: DIR_SHORTEST 1: DIR_POSITIVE 2: DIR_NEGATIVE 3: DIR_CURRENT 4: DIR_NODIRECTION 5: DIR_FASTEST	1	Specify the rotation direction in Rotary Mode: 0: DIR_SHORTEST (Direction based on the shortest way). 1: DIR_POSITIVE (Positive direction). 2: DIR_NEGATIVE (Negative direction). 3: DIR_CURRENT (Current direction). 4: DIR_NODIRECTION (No direction). 5: DIR_FASTEST (Direction based on the shortest movement time).

Input Variable	Name	Data Type	Valid Range	Default Value	Description
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting 1: mcBuffered 2: mcBlendingLow 3: mcBlendingPrevious 4: mcBlendingNext 5: mcBlendingHigh	0	0: mcAborting (Aborting). 1: mcBuffered (Buffered). 2: mcBlendingLow (Blending with the lowest velocity of both FBs). 3: mcBlendingPrevious (Blending with the velocity of the first FB). 4: mcBlendingNext (Blending with velocity of the second FB). 5: mcBlendingHigh (Blending with highest velocity of both FBs).
InverseFlag	Inverse enable	BOOL	TRUE, FALSE	FALSE	FALSE:The planned position cannot be beyond the target position. When the target velocity is not reached, the planned parameters (End Velocity, Acceleration/Deceleration, Jerk) will be modified to meet the planning requirements. TRUE: The planned position can be beyond the target position. If the axis moves beyond the target position, it will move for an additional cycle.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Controlling	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled by the current FB.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

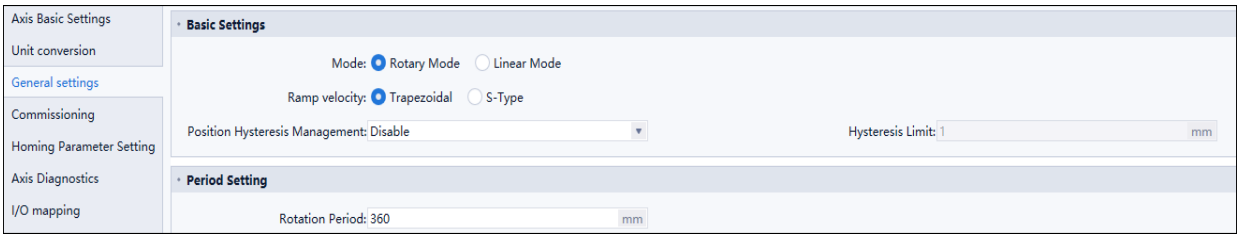
Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity	Velocity	Any velocity that exceeds the motion parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	• When the MotionControl library version is 2.1.0.0 or earlier, the velocity equal to 0 will be automatically corrected to 1e-5. • When the MotionControl library version is 2.2.0.0 or later, the velocity equal to 0 will not be corrected.
Acceleration	Acceleration	Any acceleration that exceeds the motion parameter limit will be automatically corrected to the acceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration less than 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration equal to 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration equal to 0 will be automatically corrected to 10 times the velocity.

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the deceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration less than 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration equal to 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration equal to 0 will be automatically corrected to 10 times the velocity.
Jerk	Jerk	If the motion profile is set to S-curve, any jerk that exceeds the software limit will be automatically corrected to the jerk limit.	If the motion profile is set to S-curve, any jerk less than 0 will be automatically corrected to 10% of the jerk limit.	If the motion profile is set to S-curve, the jerk equal to 0 will be automatically corrected to 10% of the jerk limit.

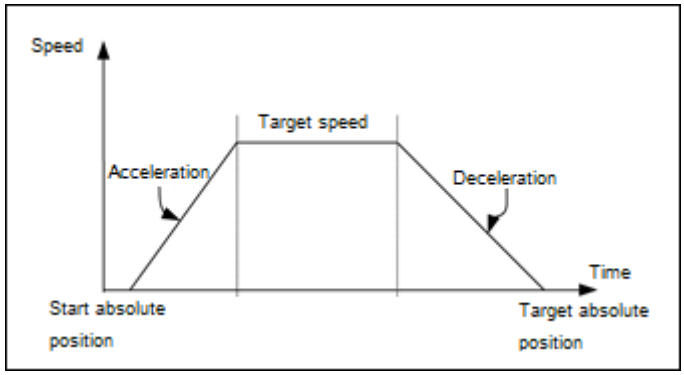
- **Function**

  Bus-controlled axis:

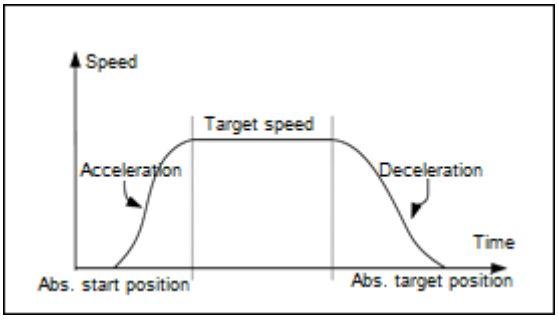
  - This instruction specifies an absolute target position for positioning. The absolute positioning starts on a rising edge of Execute. You can specify the values of Position, Velocity, Acceleration, Deceleration, and Jerk in the input parameters.
  - When the T-curve is specified in the software, the input parameter Jerk is invalid.



Acceleration/deceleration under T-curve planning:



Acceleration/deceleration under S-curve planning:



- When the setpoint of the input parameter Position is equal to the current position reference, the axis will not move after this instruction is executed, but the output parameter Done will show the value TRUE.
- When the input parameter ContinuousUpdate is set to TRUE, modified parameters Position and Velocity will take effect only when the function block is in the Active state. Parameter changes greater than 0.01 are considered updates. Modified BufferMode cannot take effect during this period, and the buffer mode at the time of the instruction trigger is retained. When the output parameter Done is set, it indicates that the instruction has completed the motion, and any changes to the parameters will not take effect.
- Direction and InverseFlag flags:
  - Linear axis

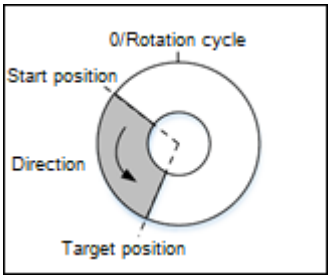
    When reversal is caused due to insufficient displacement: If the InverseFlag is set to True, the Jerk or Acceleration/Deceleration will not be adjusted. If the InverseFlag is set to False, the Jerk or Acceleration/Deceleration will be adjusted to ensure no change in the velocity sign.
  - Rotary axis

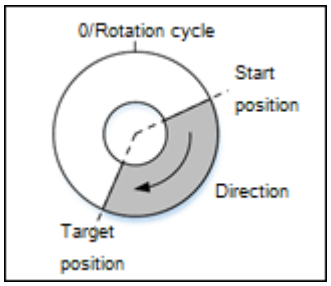
    If the InverseFlag is set to True, parameter self-adaptation will not be performed, and insufficient displacement will be compensated by adding circles. If the InverseFlag is set to False, the Jerk or Acceleration/Deceleration will be adjusted to avoid overshooting the target position and then reversing.
- **Input parameter Direction:**
  - Linear mode

    Ignore the value of the input parameter Direction, and determine the positive or negative direction of motion based on the relationship between the setpoint of the input parameter Position and the current position reference.
  - Rotary mode

    1. Input parameter Direction = 0 (DIR_SHORTEST)

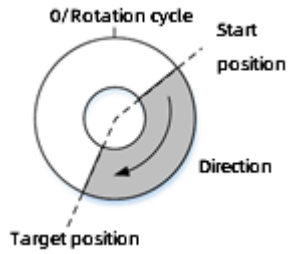
       The axis moves to the target position along the shortest way. When the setpoint of Position exceeds one rotation cycle, the target position is the remainder of the setpoint divided by the rotation cycle. At the moment the instruction is executed, the relationship between the position reference after the current axis velocity decelerates to zero and the target position is determined. If it is within half a cycle in the positive direction, the axis trajectory is consistent with the Positive direction. If it is within half a cycle in the negative direction, the axis trajectory is consistent with the Negative direction. If it is exactly half the cycle length, the axis moves in the positive direction.





2. Input parameter Direction = 1 (DIR_POSITIVE)

The axis moves to the target position in the positive direction. When the setpoint of Position exceeds one rotation cycle, the target position is the remainder of the setpoint divided by the rotation cycle. For a rotary axis, the positive direction is the direction in which the position (user units) increases within one cycle. If the current velocity is greater than zero, the axis will continue to move in the current direction until it stops at the position specified by Position. If the current velocity is less than zero, the axis will decelerate to zero and then start moving in the positive direction. If the setpoint of Position is the same as the current position reference, the axis will not move.



3. Input parameter Direction = 2 (DIR_NEGATIVE)

The axis moves to the target position in the negative direction. When the setpoint of Position exceeds one rotation cycle, the target position is the remainder of the setpoint divided by the rotation cycle. For a rotary axis, the negative direction is the direction in which the position (user units) decreases within one cycle. If the current velocity is less than zero, the axis will continue to move in the current direction until it stops at the position specified by Position. If the current velocity is greater than zero, the axis will decelerate to zero and then start moving in the opposite direction. If the setpoint of Position is the same as the current position reference, the axis will not move.

4. Input parameter Direction = 3 (DIR_CURRENT)

The axis moves to the target position in the direction of the previous movement on a rising edge of Execute. When the setpoint of Position exceeds one rotation cycle, the target position is the remainder of the setpoint divided by the rotation cycle. During the first power-on, the axis moves in the positive direction (target velocity greater than zero). If the setpoint of Position is the same as the current position reference, the axis will not move.

5. Input parameter Direction = 4 (DIR_NODIRECTION)

In this mode, the Position can be specified as a value exceeding one rotation cycle. When the Position is specified as a value equal to one rotation cycle, the axis moves in the positive or negative direction to the position in the current rotation cycle (equivalent to SHORTEST). When the Position is specified as a value less than zero, the axis moves to an absolute target position in the negative direction of zero position in the current cycle. When the Position is specified as a value greater than zero and exceeding one rotation cycle, the axis moves to an absolute target position in the positive direction of 360 position in the current cycle

6. Input parameter Direction = 5 (DIR_FASTEST)

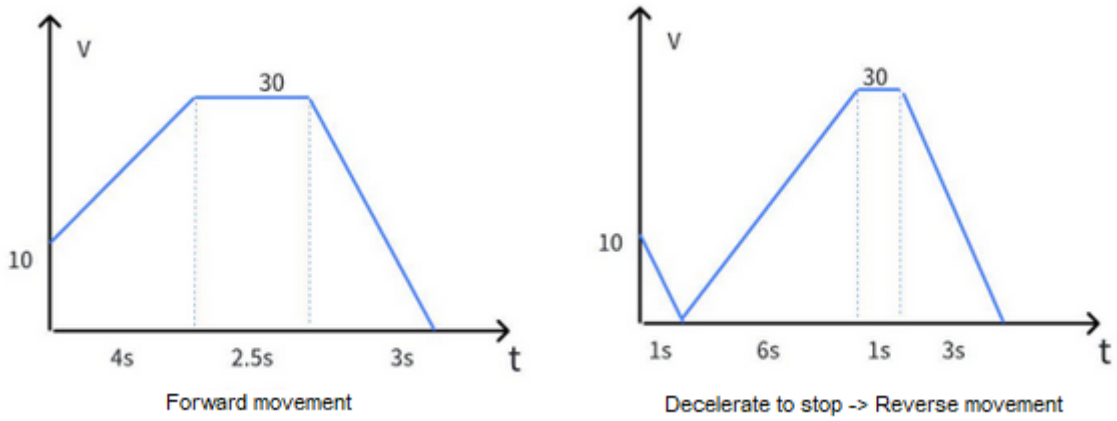
The axis moves in the direction that reaches the target position the fastest. When the setpoint of Position exceeds one rotation cycle, the target position is the remainder of the setpoint divided by the rotation cycle. When the instruction is executed, if the initial velocity and acceleration of the axis are zero, the behavior of the axis moving in the direction specified by DIR_FASTEST is the same as that in direction specified by DIR_SHORTEST. When the velocity and acceleration are not zero, the direction of the axis moving in direction specified by DIR_FASTEST may not be the same as the direction of the axis moving in direction specified by DIR_SHORTEST. The examples are provided for illustration:

Example 1. When the instruction is executed, the axis moves at a velocity of 0 with the specified initial position = 50, target position = 150, acceleration = 5, deceleration = 10, target velocity = 20, and jerk = 5, and the Direction is set to 5(DIR_FASTEST). Which direction will the axis move? (Using T-curve planning as an example)

Analysis: Since the initial velocity of the axis is 0, the positive displacement increment is 100 and it takes 8s; the negative displacement increment is -260 and it takes 16s. Therefore, the axis will move in the positive direction to reach the target position.

Example 2. When the instruction is executed, the axis moves at a velocity of 10 with the specified initial position = 50, target position = 250, acceleration = 5, deceleration = 10, target velocity = 30, and jerk = 5, and the Direction is set to 5(DIR_FASTEST). Which direction will the axis move? (Using T-curve planning as an example)

Analysis: When the instruction is executed, the axis velocity is 10. If the axis moves in the positive direction, the displacement increment is 200 (i.e., 250-50), and it takes 9.5s for the master axis to reach the target position. If the axis moves in the negative direction, the master axis first decelerates to 0 (at this point, the axis is at position 55; which takes 1s), and then moves in the opposite direction to reach the target position, with a displacement increment of (250-55-360=-165), which takes 10s. The total time is 11s (1s + 10s). Although the displacement increment when the axis move in the positive direction to reach the target position is larger, due to its shorter time consumption, the DIR_FASTEST will be selected, rather than the DIR_SHORTEST. The velocity curves for the axis moving in the positive and negative directions are shown below:



- When the instruction is triggered on a rising edge of the input parameter Execute, the input parameters are checked for validity. If any of invalid parameters is detected, it will be automatically corrected by the function block with no fault message; any buffered instructions after this instruction will be cleared.
- If there is no buffered instruction when this instruction is executed, this instruction will end at a velocity of zero. If there is a buffered instruction with the input parameter BufferMode not being set to MC_Aborting or MC_Buffered, the instruction will end at the velocity corresponding to the transition mode of the subsequent buffered instruction.
- If the current axis state is not AXIS_STANDSTILL, AXIS_DISCRETE_MOTION, AXIS_CONTINUOUS_MOTION, or AXIS_SYNCHRONIZED_MOTION, this instruction cannot be executed.
- The axis type must be control axis.
- If any setpoint of related motion parameters is outside the limits and the current velocity reference is not zero, the axis will decelerate to stop under a T-curve planning, using the maximum deceleration in the software.
- After PDOs 6060 and 6061 are configured, control mode switching between instructions does not rely on the MC_SetControllerMode instruction. You can directly call instructions under different control modes. Each instruction internally checks if the current mode meets the requirements and automatically switches the mode if necessary.
- During the deceleration stop due to limits, the axis state is AXIS_STOPPING with the output status of the instruction unchanged. After the axis stops and comes to rest, whether to switch to AXIS_ERRORSTOP will be determined based on the IDE interface options, and the instruction will output a fault message.
- During the deceleration due to limits, do not execute the MC_Reset instruction to reset fault messages.
- This instruction cannot be changed to buffered mode for re-execution during execution.
- During the execution of this instruction, the MC_MoveSuperImposed instruction can be used.
- When the axis is enabled after PLC cold reset and the MC_Home instruction is executed using No. 35 homing method, if this instruction is executed with DIR_CURRENT being set, the motion direction of this instruction will be affected.
- When this instruction is executed with the ContinuousUpdate being set to TRUE along with the MC_SuperImposed instruction, the MC_SuperImposed instruction will be aborted if any modifiable parameters are modified.
- Switching the MC_SetControllerMode to a non-CSP mode during motion causes inaccurate positioning.
- The arrival check function is only applicable to MotionControl library version 2.4.0.0 or later. Based on the "Positioning range" and "Positioning check time" parameters in the "General setting" of the axis

configuration page, it checks in real time whether the actual position has entered the specified target range. After the instruction output for the target position is completed, positioning is complete and the Done output is set if the difference between the actual position and the target position is smaller than the "Positioning range". If positioning is not completed within the "Positioning check time", a fault will occur.

Local pulse axis:

- This instruction can be used for local pulse axes.
- When the local pulse axis is used with Y0 or Y1 single-ended output, it is not recommended to use this instruction. ① No pulse output; ② No position feedback.

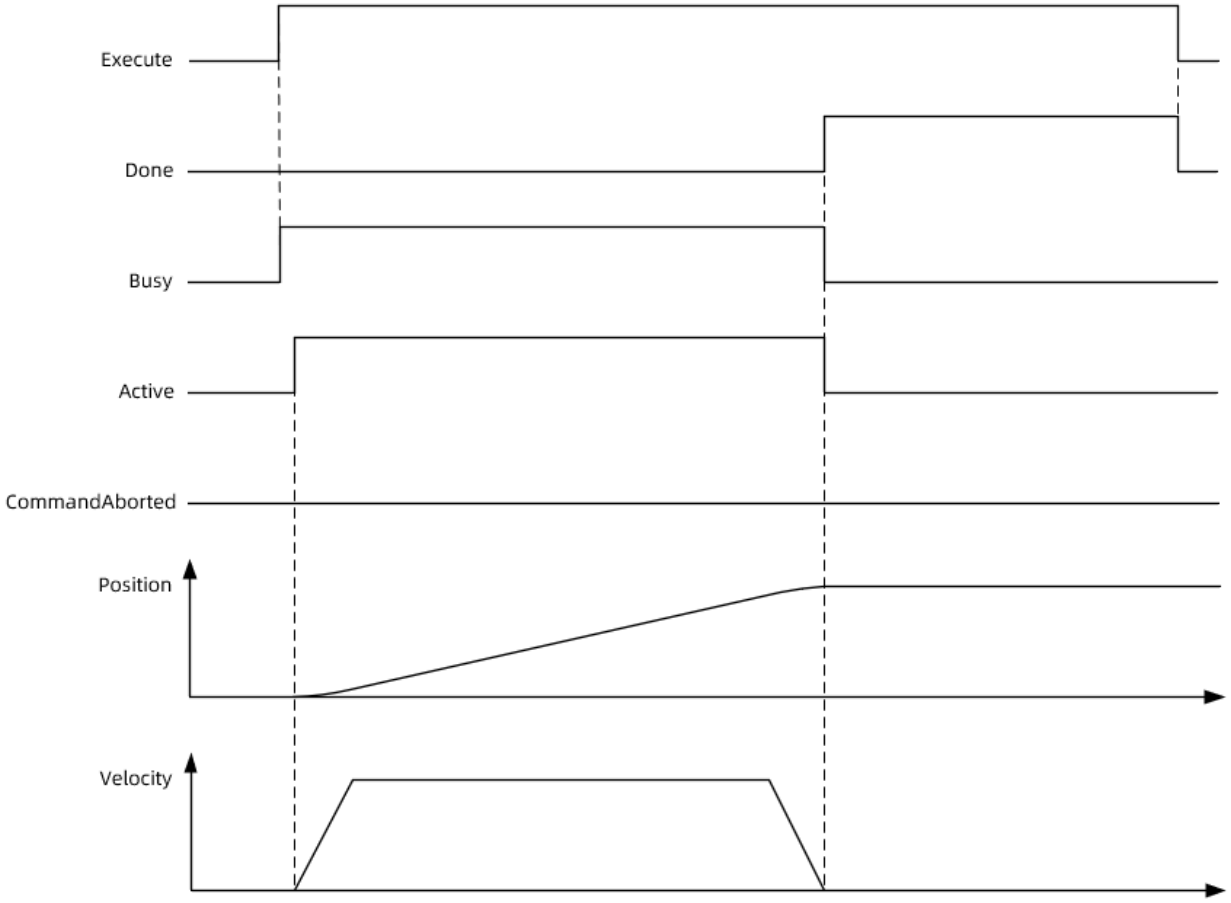
- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates the instruction has been executed successfully; the parameter Busy indicates that the current function block is being executed; the parameter CommandAborted indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

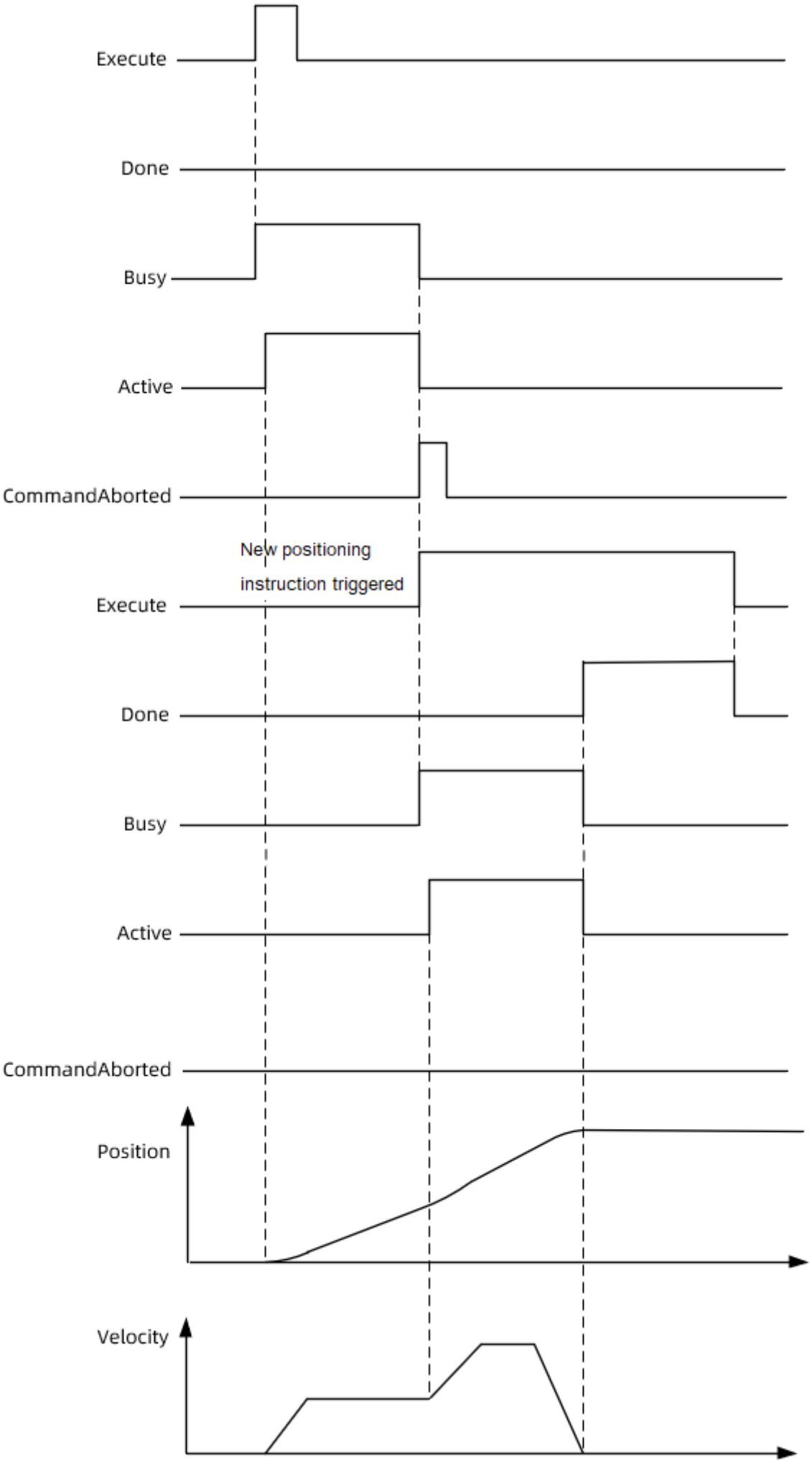
  The parameter Busy shows the value TRUE on a rising edge of Execute. In the next cycle, the output parameter Active shows the value TRUE. When the setpoint of Position is reached and positioning is complete, the parameter Done shows the value TRUE.

  When this instruction is aborted by another instruction, the CommandAborted is set to TRUE, and the Busy and Active are set to FALSE.

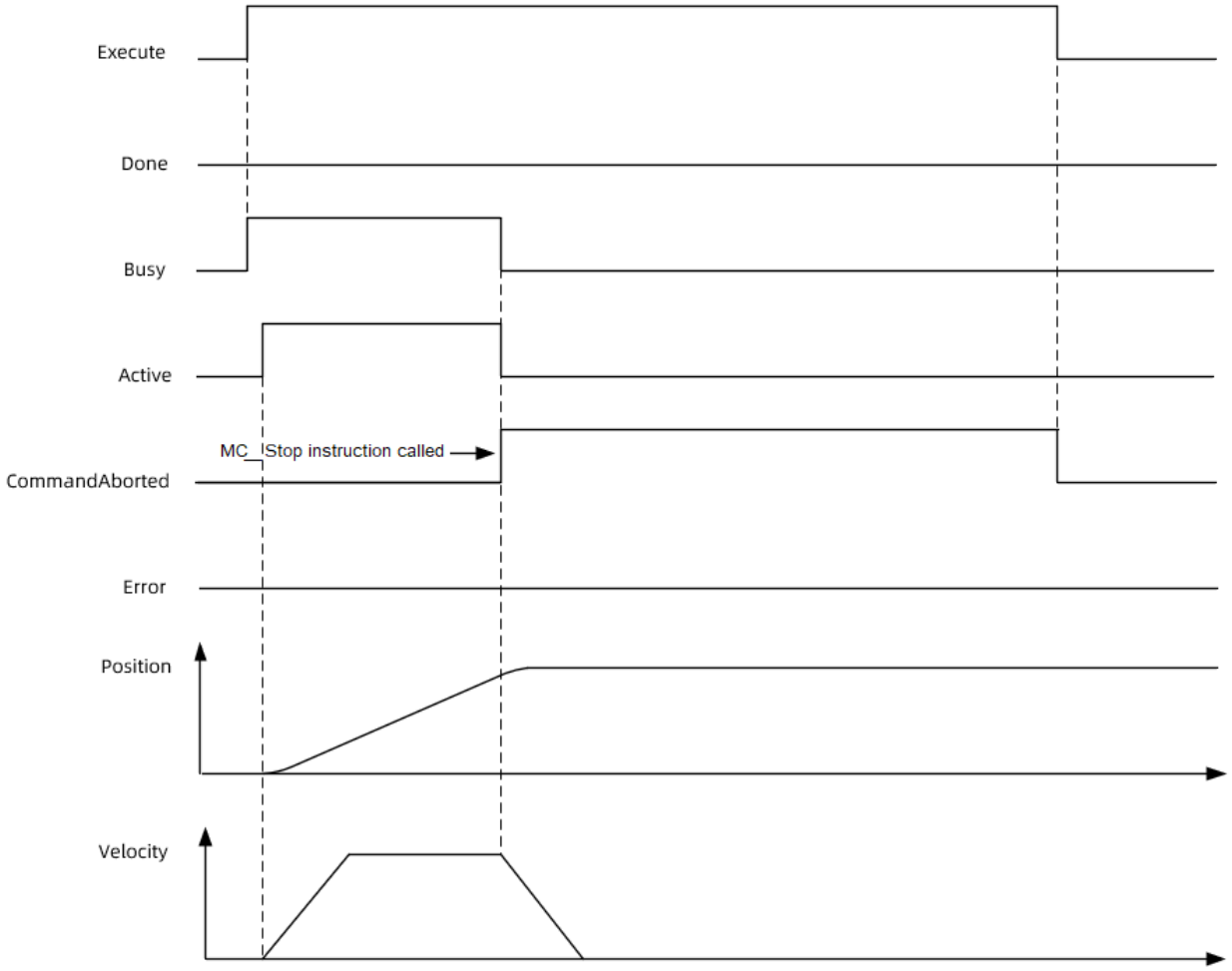
  - When this instruction is called to perform absolute positioning under a T-curve planning while the axis state is StandStill, the output variable Done will show the value TRUE if the MC_MoveAbsolute_Instance_0 function block reaches the position reference (with a velocity 0).



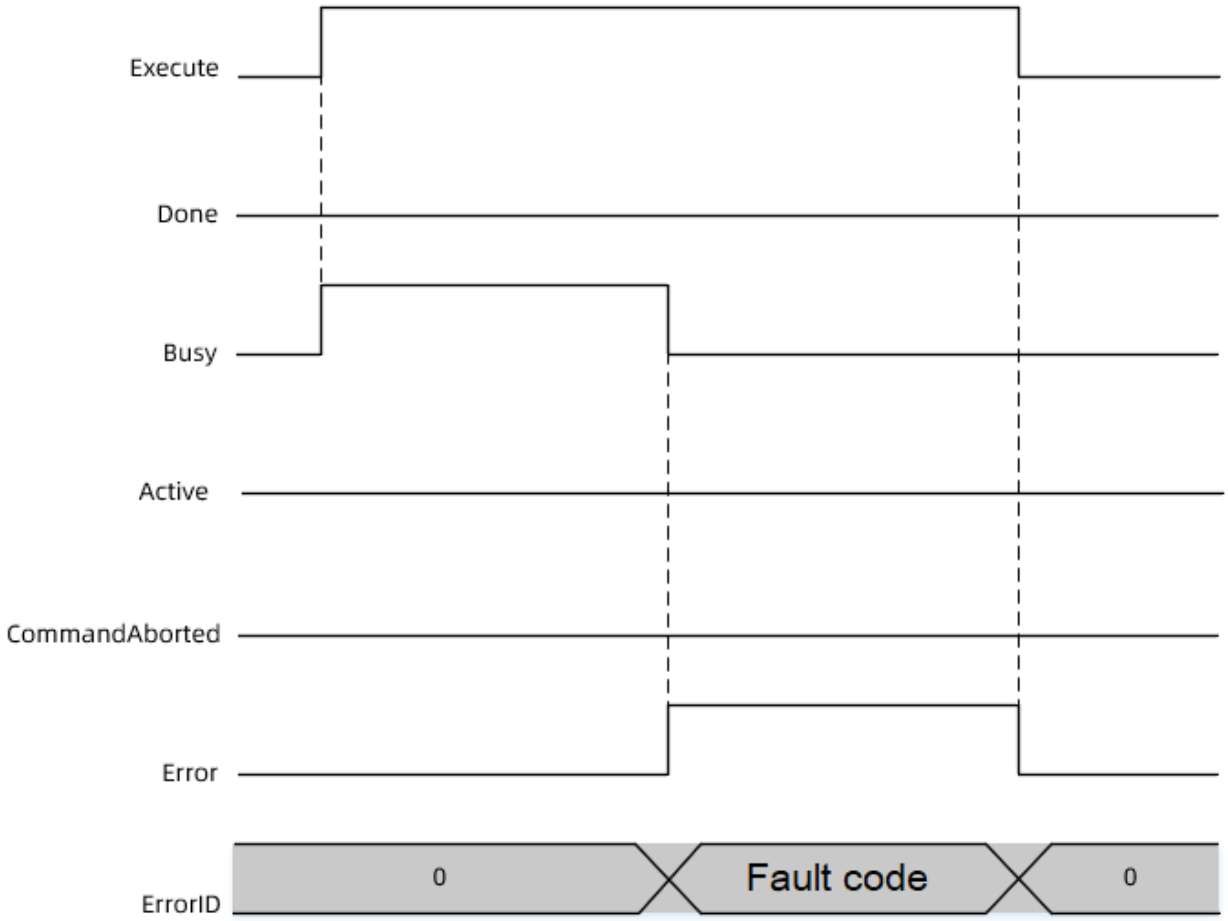
- When another MC_MoveAbsolute instruction is executed during the absolute positioning, and the second MC_MoveAbsolute_Instance_1 FB is executed during the execution of first MC_MoveAbsolute_Instance_0 FB, the absolute positioning is aborted by the second FB. Even though the first FB has not reached the position reference, the second FB directly moves to the position reference.


Execute
Done
Busy
Active
CommandAborted
New positioning
instruction triggered
Execute
Done
Busy
Active
CommandAborted
Position
Velocity

◦ Absolute positioning of the axis is aborted by MC_Stop instruction.



◦ The drive malfunctions during the absolute positioning of the axis.

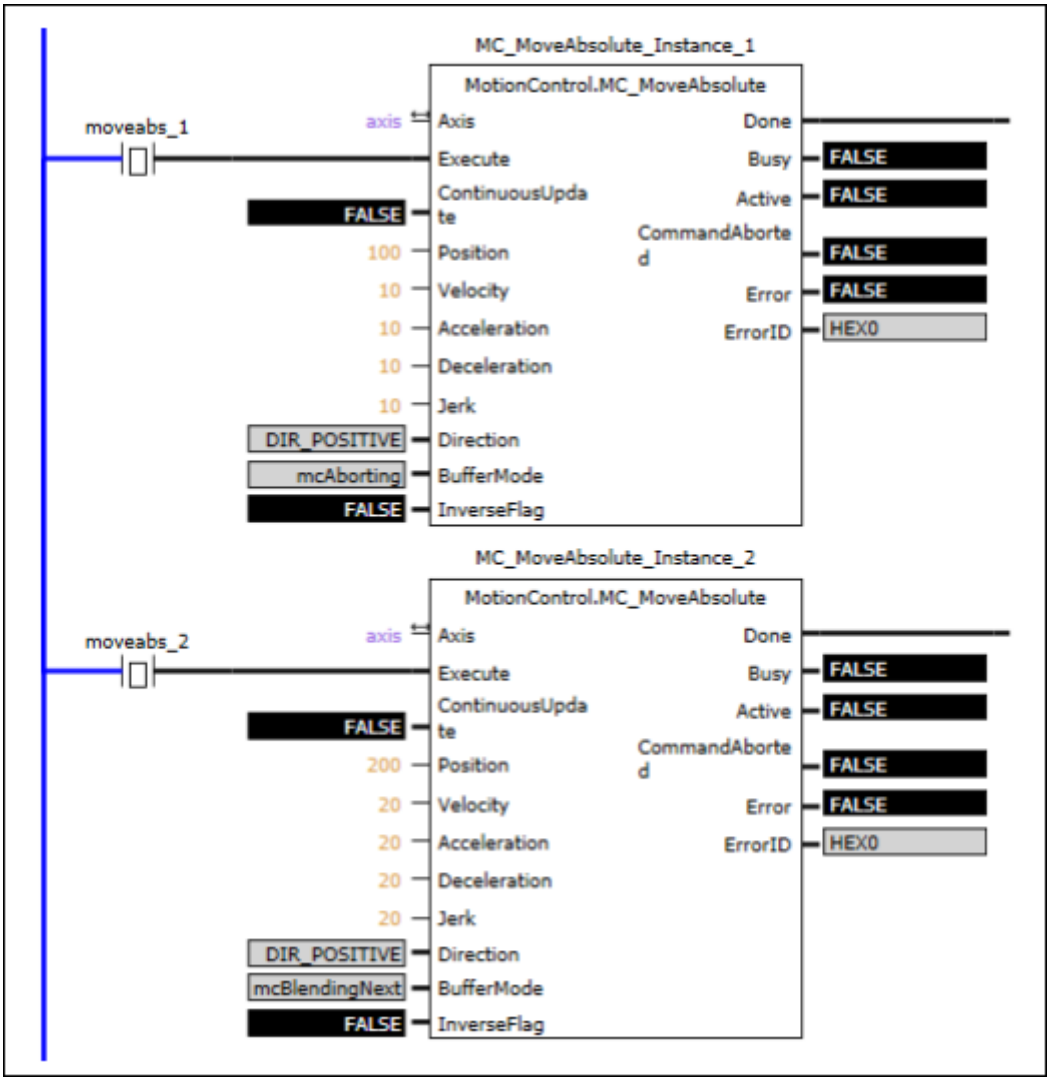


- **Faults**

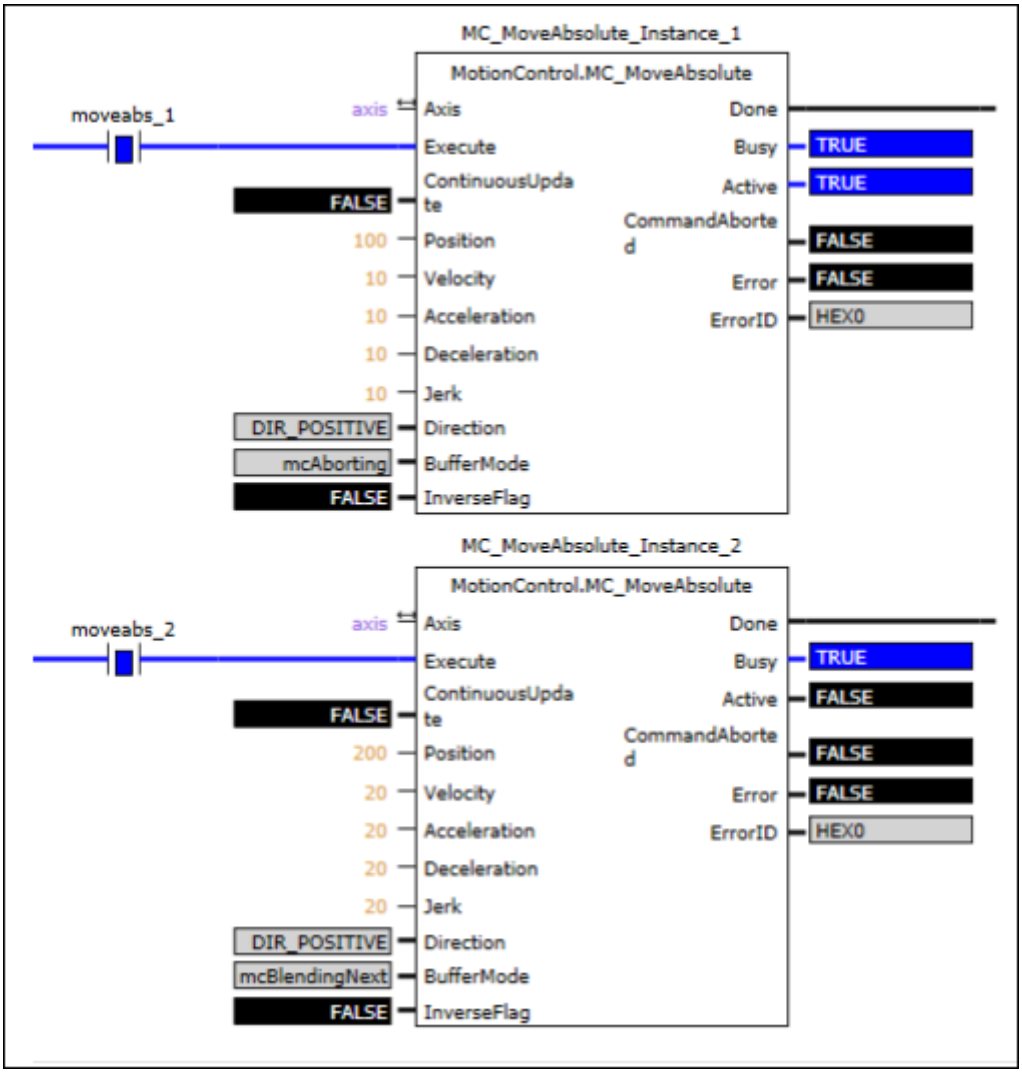
  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

- **Example**

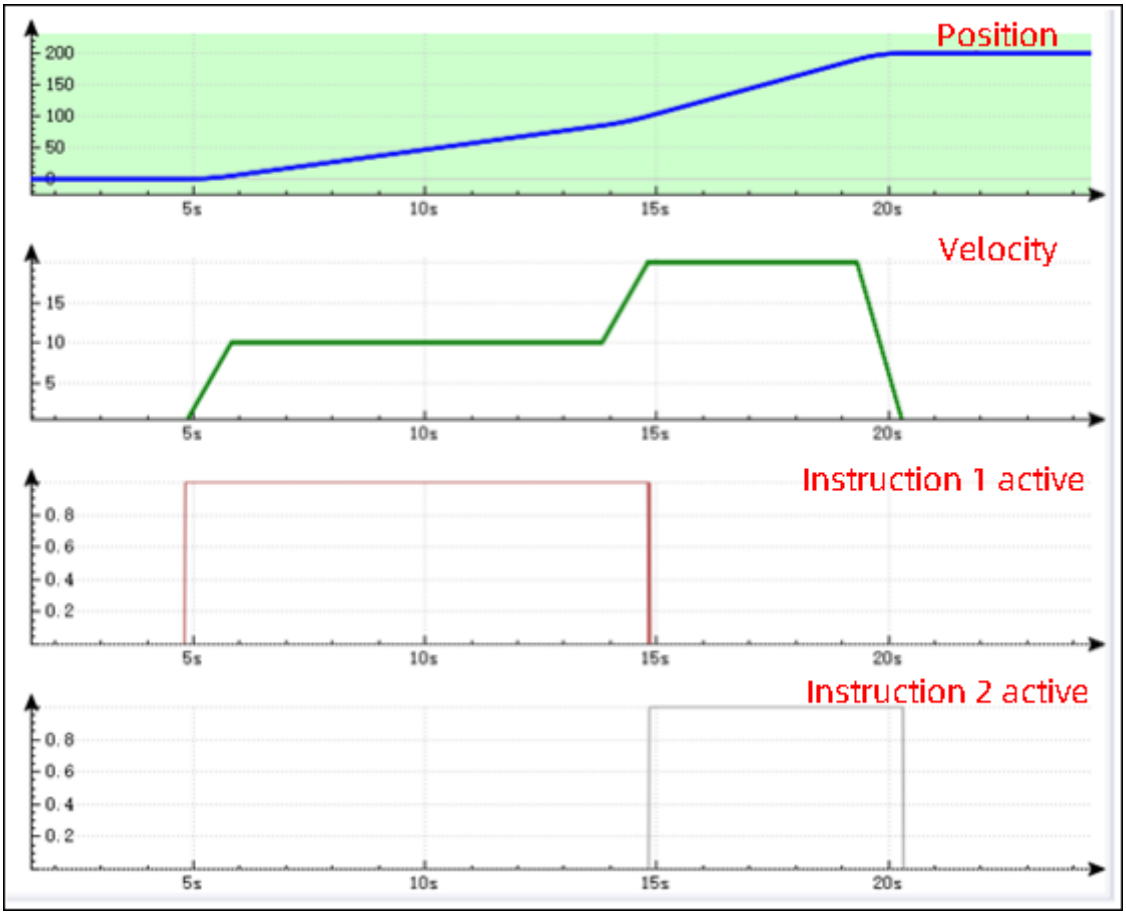
  Instantiate two MC_MoveAbsolute instructions with the motion parameters being specified according to actual process requirements. Specify the velocity as 10 for the first instruction and 20 for the second instruction. Specify the BufferMode as mcBlendingNext for the second executed instruction.



◦ If the second instruction is executed during the execution of the first instruction, the second instruction will be buffered.



◦ The first instruction finally stops to position 100 at a velocity of 20, and then the second instruction starts moving at the inherited velocity of 20 and stops at position 200.



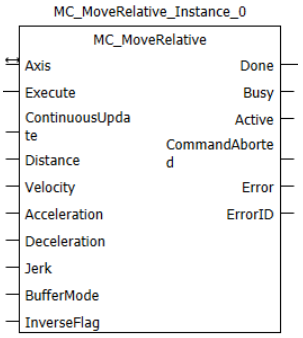
- **Precautions**

  This instruction can be re-executed. The motion parameters can be modified and re-triggered while the instruction is being executed or in the buffer. However, during the execution of the instruction, only mcAborting can be re-triggered, especially when the instruction just starts moving from the buffer. The BufferMode cannot be modified when the instruction is re-triggered in the buffer; otherwise, the instruction will issue a fault message.

18.3.3.3.32 MC_MoveRelative

This instruction moves an axis a specified distance relative to the actual position at the time of the execution.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_MoveRelative	Relative positioning		MC_MoveRelative _Instance_0(Axis :=, Execute :=, ContinuousUpdate :=, Distance :=, Velocity :=, Acceleration :=, Deceleration :=, Jerk :=, BufferMode :=, InverseFlag :=, Done =>, Busy =>, Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Continuous Update	Continuous update	BOOL	TRUE, FALSE	FALSE	Continuous parameter update effective flag. Prerequisite: The parameter ContinuousUpdate shows the value TRUE and the parameter Active shows the value TRUE. Effective parameters: Position, Velocity. Note: The modified parameter values must differ from the original parameter values by more than 0.01.
Distance	Relative distance	LREAL	Positive number, negative number, or 0	0	Distance [u] the axis moves relative to its current position.
Velocity	Velocity	LREAL	Positive number	0	Maximum velocity [u/s].
Acceleration	Acceleration	LREAL	Positive number	0	Acceleration [u/s^2] of the acceleration phase.
Deceleration	Deceleration	LREAL	Positive number	0	Deceleration [u/s^2] of the deceleration phase.
Jerk	Jerk	LREAL	Positive number	0	Slope [u/s^3] of acceleration or deceleration. (Only valid for S-curve planning).
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting 1: mcBuffered 2: mcBlendingLow 3: mcBlendingPrevious 4: mcBlendingNext 5: mcBlendingHigh	0	0: mcAborting (Aborting). 1: mcBuffered (Buffered). 2: mcBlendingLow (Blending with the lowest velocity of both FBs). 3: mcBlendingPrevious (Blending with the velocity of the first FB). 4: mcBlendingNext (Blending with velocity of the second FB). 5: mcBlendingHigh (Blending with highest velocity of both FBs).

Input Variable	Name	Data Type	Valid Range	Default Value	Description
InverseFlag	Inverse enable	BOOL	TRUE, FALSE	FALSE	FALSE: If the velocity cannot be reached, the velocity in the next segment will be changed. The planned position does not exceed the target position. TRUE: The planned position can be beyond the target position. In this case, the axis will move in the opposite direction to the target position.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Controlling	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled by the current FB.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

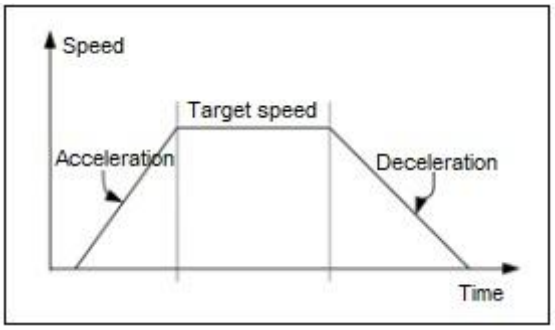
Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity	Velocity	Any velocity that exceeds the motion parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	• When the MotionControl library version is 2.1.0.0 or earlier, the velocity equal to 0 will be automatically corrected to 1e-5. • When the MotionControl library version is 2.2.0.0 or later, the velocity equal to 0 will not be corrected.
Acceleration	Acceleration	Any acceleration that exceeds the motion parameter limit will be automatically corrected to the acceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration less than 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration equal to 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration equal to 0 will be automatically corrected to 10 times the velocity.
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the deceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration less than 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration equal to 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration equal to 0 will be automatically corrected to 10 times the velocity.
Jerk	Jerk	If the motion profile is set to S-curve, any jerk that exceeds the software limit will be automatically corrected to the jerk limit.	If the motion profile is set to S-curve, any jerk less than 0 will be automatically corrected to 10% of the jerk limit.	If the motion profile is set to S-curve, the jerk equal to 0 will be automatically corrected to 10% of the jerk limit.

- **Function**

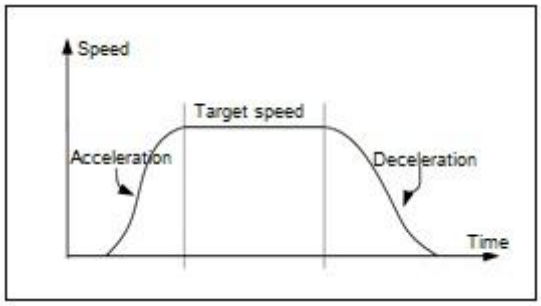
Bus-controlled axis:

- This instruction specifies a relative target position for positioning. The relative positioning starts on a rising edge of Execute. You can specify the values of Distance, Velocity, Acceleration, Deceleration, and Jerk in the input parameters.
- When the instruction is triggered on a rising edge of the input parameter Execute, the input parameters are checked for validity. If any of invalid parameters is detected, it will be automatically corrected by the function block with no fault message; any buffered instructions after this instruction will be cleared.
- When the T-curve is specified in the software, the input parameter Jerk is invalid.

  Acceleration/deceleration under T-curve planning:



  Acceleration/deceleration under S-curve planning:



- When the input parameter ContinuousUpdate is set to TRUE, modified parameters Distance and Velocity will take effect only when the function block is in the Active state. Parameter changes greater than 0.01 are considered updates. Modified BufferMode cannot take effect during this period, and the buffer mode at the time of the instruction trigger is retained. When the output parameter Done is set, it indicates that the instruction has completed the motion, and any changes to the parameters will not take effect.
- In Rotary Mode, when the setpoint of the input parameter Distance is greater than one cycle, multi-turn positioning can be performed.
- If there are no other pending operations, the instruction will be done at a velocity of 0.
- If the setpoint of the input parameter BufferMode is neither MC_Aborting nor MC_Buffered, the end velocity will depend on the specified transition mode.
- If the current axis state is not AXIS_STANDSTILL, AXIS_DISCRETE_MOTION, AXIS_CONTINUOUS_MOTION, or AXIS_SYNCHRONIZED_MOTION, this instruction cannot be executed.
- The axis type must be control axis.
- If any setpoint of related motion parameters is outside the limits and the current velocity reference is not 0, the axis will decelerate to stop under a T-curve planning, using the maximum deceleration in the software.

- After PDOs 6060 and 6061 are configured, control mode switching between instructions does not rely on the MC_SetControllerMode instruction. You can directly call instructions under different control modes. Each instruction internally checks if the current mode meets the requirements and automatically switches the mode if necessary.
- During the decelerate-to-stop process due to limits, the axis state is AXIS_STOPPING with the output status of the instruction unchanged. After the axis stops and comes to rest, whether to switch to AXIS_ERRORSTOP will be determined based on the IDE interface options, and the instruction will output an error message.
- During the deceleration due to limits, do not execute the MC_Reset instruction to reset error messages.
- This instruction cannot be changed to buffered mode for re-execution during execution.
- During the execution of this instruction, the MC_MoveSuperImposed instruction can be used.
- When the axis is enabled after PLC cold reset and the MC_Home instruction is executed using No. 35 homing method, if this instruction is executed with DIR_CURRENT being set, the motion direction of this instruction will be affected.
- When this instruction is executed with the ContinuousUpdate being set to TRUE along with the MC_SuperImposed instruction, the MC_SuperImposed instruction will be aborted if any modifiable parameters are modified.
- The arrival check function is only applicable to MotionControl library version 2.4.0.0 or later. Based on the "Positioning range" and "Positioning check time" parameters in the "General setting" of the axis configuration page, it checks in real time whether the actual position has entered the specified target range. After the instruction output for the target position is completed, positioning is complete and the Done output is set if the difference between the actual position and the target position is smaller than the "Positioning range". If positioning is not completed within the "Positioning check time", a fault will occur.

  Local pulse axis:

- This instruction can be used for local pulse axes.
- When the local pulse axis is used with Y0 or Y1 single-ended output, it is not recommended to use this instruction. ① No pulse output; ② No position feedback.

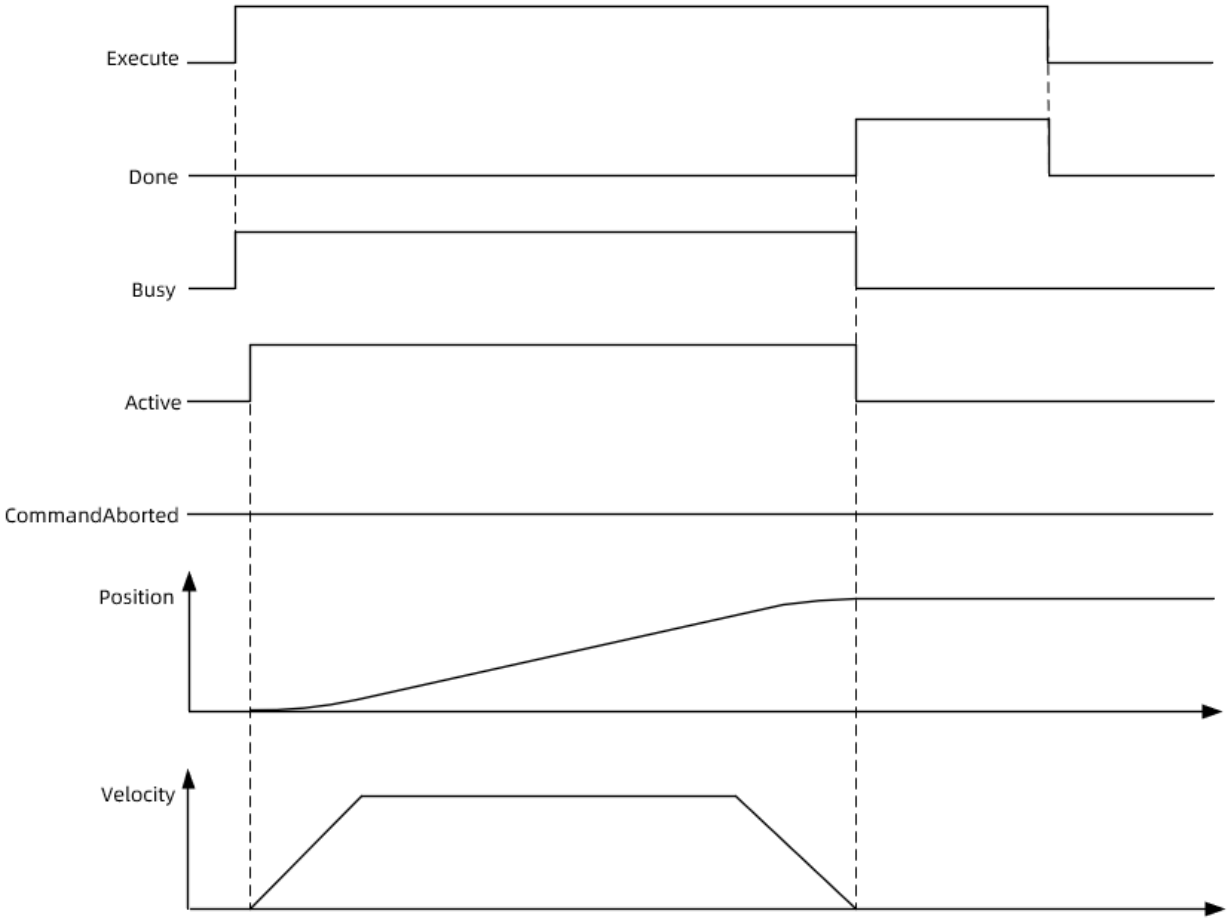
- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates the instruction has been executed successfully; the parameter Busy indicates that the current function block is being executed; the parameter CommandAborted indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

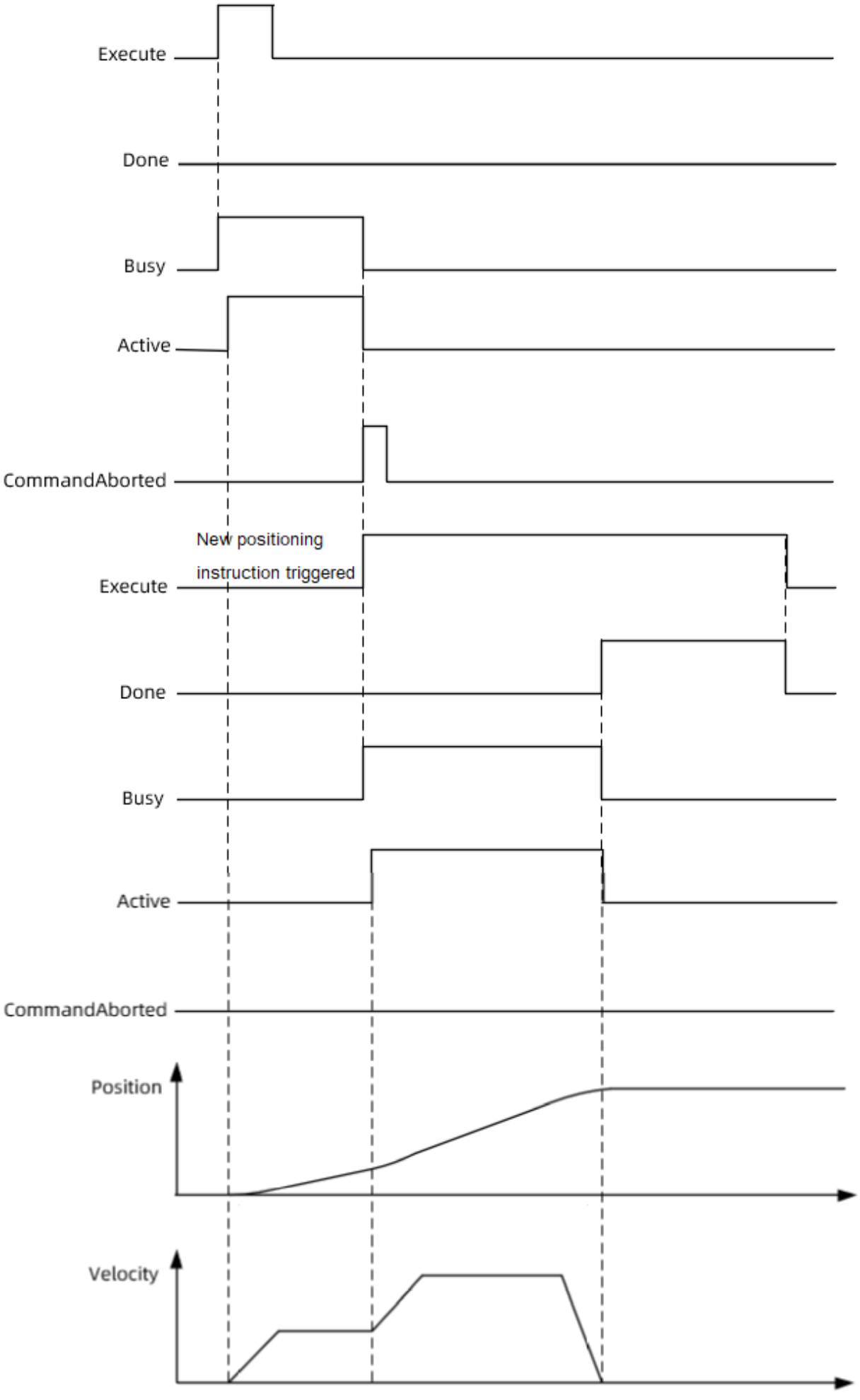
  The parameter Busy shows the value TRUE on a rising edge of Execute. In the next cycle, the output parameter Active shows the value TRUE. When the setpoint of Position is reached and positioning is complete, the parameter Done shows the value TRUE.

  When this instruction is aborted by another instruction, the CommandAborted is set to TRUE, and the Busy and Active are set to FALSE.

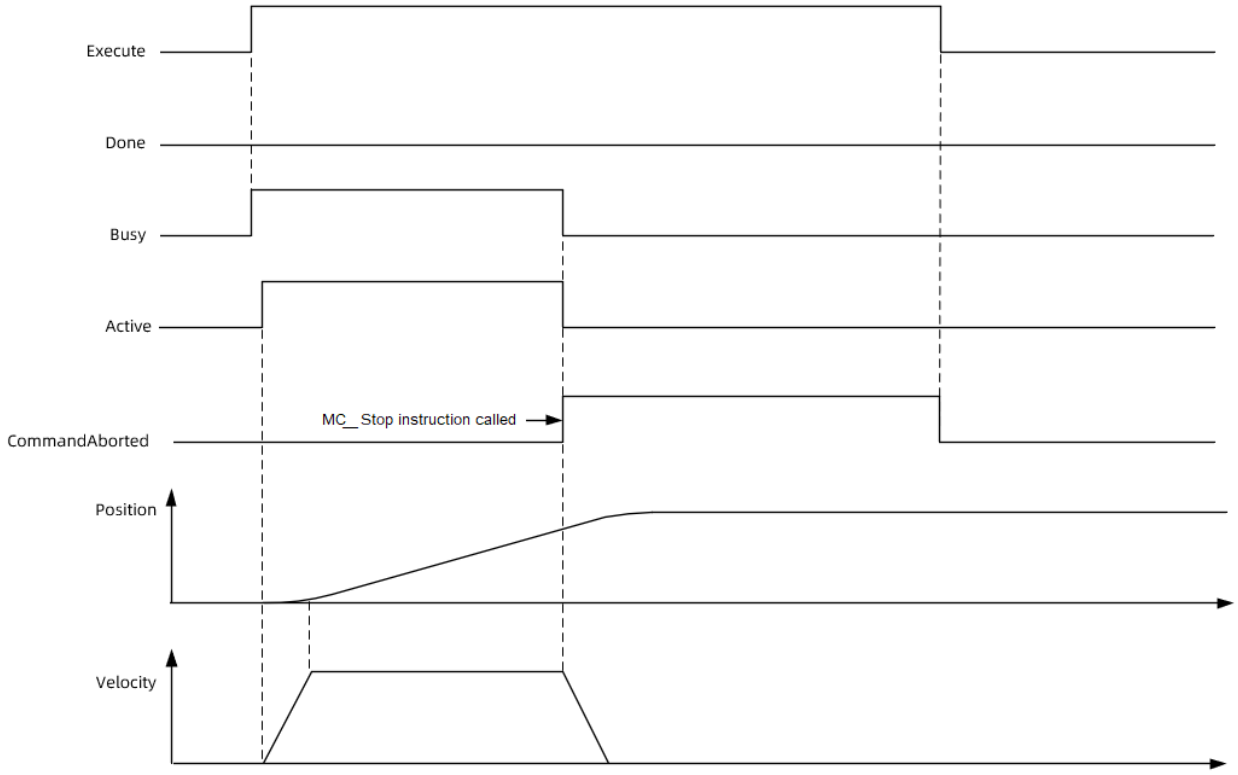
- When this instruction is called while the axis state is StandStill, the output variable Done will show the value TRUE if the MC_MoveAbsolute_Instance_0 function block reaches the position reference (with a velocity of zero).



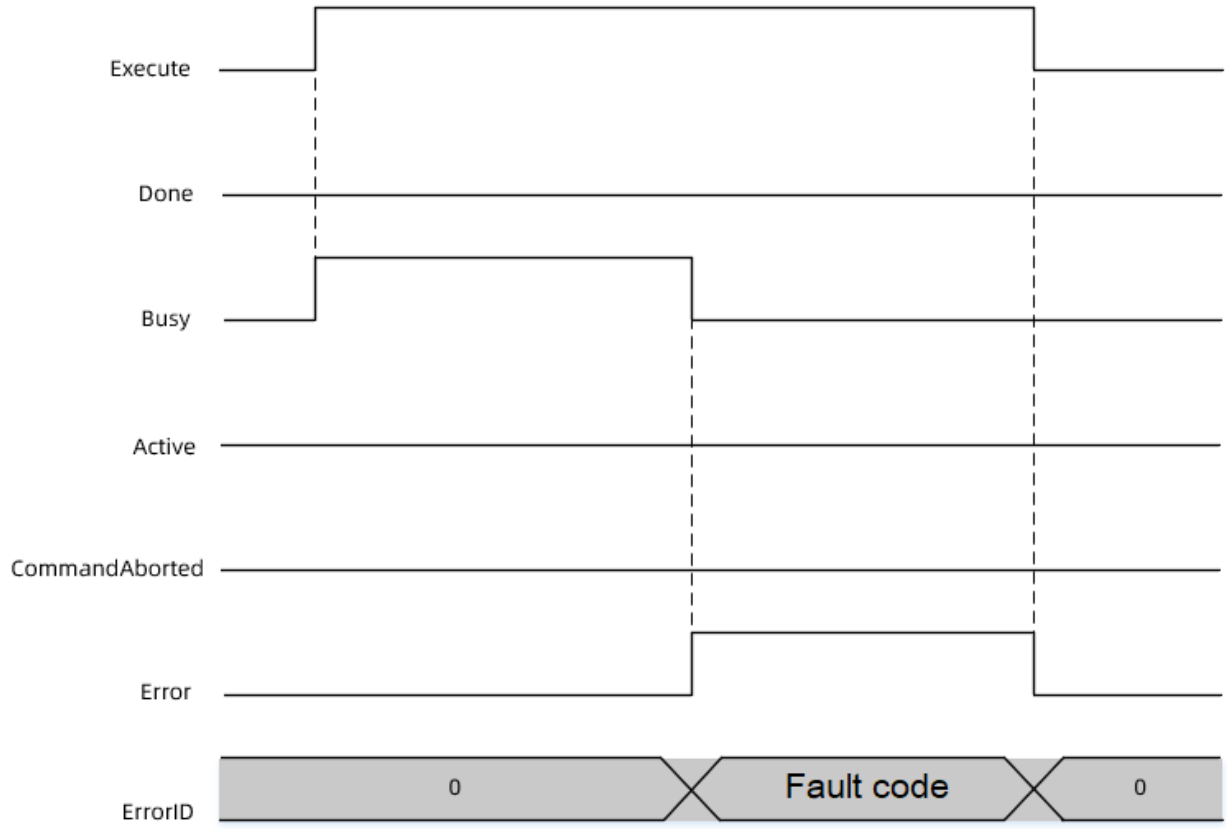
- When another MC_MoveRelative instruction is executed during the relative positioning, and the second MC_MoveRelative_Instance_1 FB is executed during the execution of first MC_MoveRelative_Instance_0 FB, the first FB is aborted by the second FB during. Even though the first FB has not reached the position reference, the second FB directly moves to the position reference.


Execute
Done
Busy
Active
CommandAborted
New positioning instruction triggered
Execute
Done
Busy
Active
CommandAborted
Position
Velocity

◦ Relative positioning of the axis is aborted by MC_Stop instruction.



◦ The drive malfunctions during the relative positioning of the axis.



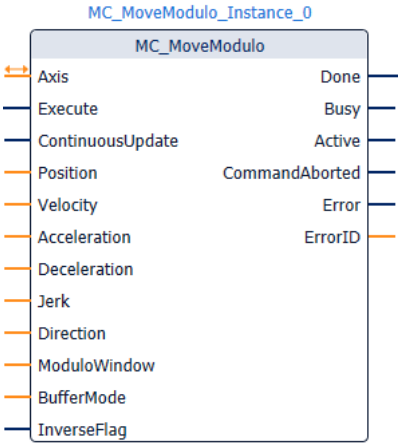
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.33 MC_MoveModulo

This instruction is used to position a rotary axis based on a modulo period.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_MoveModulo	Rotary axis modulo positioning		MC_MoveModulo_Instance_0(Axis := , Execute :=, ContinuousUpdate :=, Position := , Velocity := , Acceleration :=, Deceleration :=, Jerk :=, Direction :=, ModuloWindow := , BufferMode :=, InverseFlag :=, Done =>, Busy => , Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.
ContinuousUpdate	Continuous update	BOOL	TRUE, FALSE	FALSE	Continuous parameter update effective flag. Prerequisite: The parameter ContinuousUpdate shows the value TRUE and the parameter Active shows the value TRUE. Effective parameters: Position, Velocity. Note: The modified parameter values must differ from the original parameter values by more than 0.01.
Position	Position	LREAL	Positive number or 0	0	Target position [u] in absolute coordinates.
Velocity	Velocity	LREAL	Positive number	0	Maximum velocity [u/s], which may not be reached (cannot be zero).

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Acceleration	Acceleration	LREAL	Positive number	0	Acceleration [u/s^2] of the acceleration phase.
Deceleration	Deceleration	LREAL	Positive number	0	Deceleration [u/s^2] of the deceleration phase.
Jerk	Jerk	LREAL	Positive number	0	Slope [u/s^3] of acceleration or deceleration. (Only valid for S-curve planning).
Direction	Motion direction	18.3.3.2.2.9 MC_DIRECTION: Motion Direction Enumeration	0: DIR_SHORTEST 1: DIR_POSITIVE 2: DIR_NEGATIVE 3: DIR_CURRENT	0	0: DIR_SHORTEST (Direction based on the shortest way). 1: DIR_POSITIVE (Positive direction). 2: DIR_NEGATIVE (Negative direction). 3: DIR_CURRENT (Current direction).
ModuloWindow	Modulo factor window	LREAL	Positive number or 0	0	When the target position and current position are within the window, the axis moves along the shortest way.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting 1: mcBuffered 2: mcBlendingLow 3: mcBlendingPrevious 4: mcBlendingNext 5: mcBlendingHigh	0	0: mcAborting (Aborting). 1: mcBuffered (Buffered). 2: mcBlendingLow (Blending with the lowest velocity of both FBs). 3: mcBlendingPrevious (Blending with the velocity of the first FB). 4: mcBlendingNext (Blending with velocity of the second FB). 5: mcBlendingHigh (Blending with highest velocity of both FBs).

Input Variable	Name	Data Type	Valid Range	Default Value	Description
InverseFlag	Inverse enable	BOOL	TRUE, FALSE	FALSE	FALSE:The planned position cannot be beyond the target position. When the target velocity is not reached, the planned parameters (End Velocity, Acceleration/ Deceleration, Jerk) will be modified to meet the planning requirements. TRUE: The planned position can be beyond the target position. If the axis moves beyond the target position, it will move for an additional cycle.

Output variables

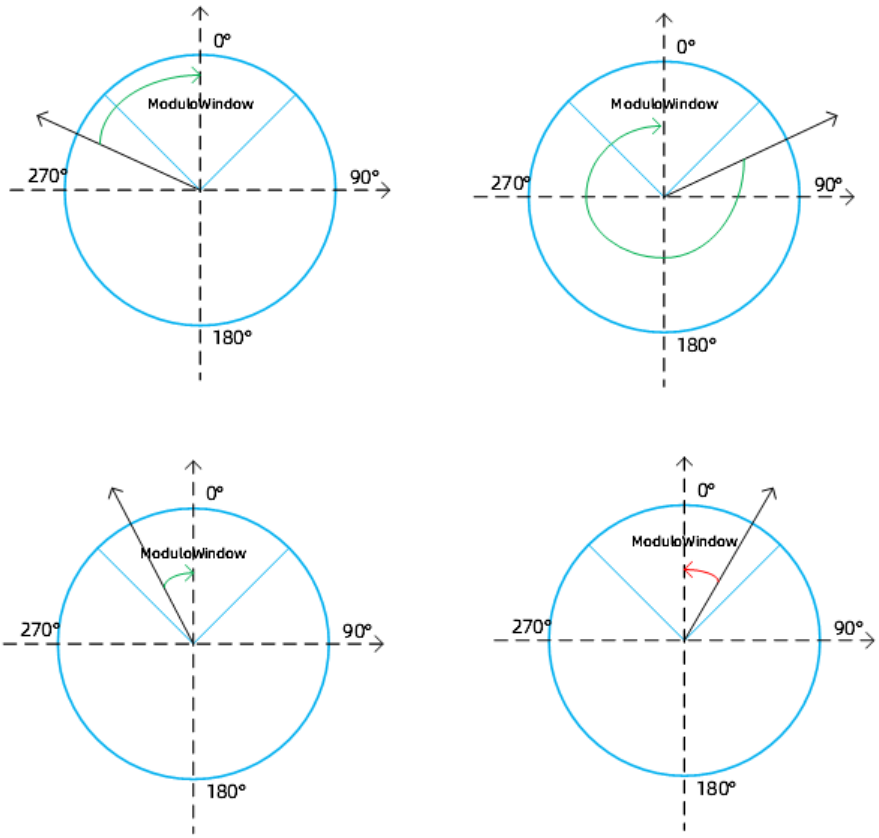
Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is being executed.
Active	Controlling	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled by the current FB.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**
  - This instruction specifies an absolute target position for positioning. The absolute positioning starts on a rising edge of Execute. You can specify the values of Position, Velocity, Acceleration, Deceleration, and Jerk in the input parameters.
  - When the T-curve is specified in the software, the input parameter Jerk is invalid.
  - When the setpoint of the input parameter Position is equal to the current position reference, the axis will not move after this instruction is executed, but the output parameter Done will show the value TRUE.
  - When the input parameter ContinuousUpdate is set to TRUE, modified parameters Position and Velocity will take effect only when the function block is in the Active state. Parameter changes greater than 0.01 are considered updates. Modified BufferMode, ModuloWindow, and Direction cannot take effect during this period, and the buffer mode at the time of the instruction trigger is retained.
  - If the InverseFlag is set to True, parameter auto-tuning will not be performed, and insufficient displacement will be compensated by adding circles. If the InverseFlag is set to False, the Jerk or Acceleration/Deceleration will be adjusted to avoid overshooting the target position and then reversing.
  - **Input parameter Position:**
    Unlike the MC_MoveAbsolute, the MC_MoveModulo enables the axis to rotate multiple turns when the Position is greater than or equal to the specified rotation cycle. Negative positions are not permitted.

    For example, if the rotation cycle is 360, the start position is 10, the input value for Position is 360, and Direction is DIR_POSITIVE, the axis moves in positive direction by 710 and finally positions to 0. If multi-turn rotations are not required and positive movement to 0 is desired, Position must be input as 0. In this case, the axis rotates in positive direction by 350 and finally positions to 0.
  - **Input parameter ModuloWindow:**
    When the difference between the current position and the target position is less than or equal to the specified tolerance window, the system automatically selects the shortest way to move to the target position, behaving the same as if the Direction were set to DIR_SHORTEST. In this case, the user-defined Direction will be invalid. Note that the permitted window is ± ModuloWindow, resulting in an effective error window that is twice the configured value.
    As shown in the figures below, when the direction is set to DIR_POSITIVE and the current position is outside the ModuloWindow, the axis moves in the DIR_POSITIVE direction (upper two figure); when the

current position is within the ModuloWindow, the axis moves in the DIR_SHORTEST direction (lower two figures).



- **Input parameter Direction:**
  - Direction = 0 (DIR_SHORTEST)
    The axis moves to the target position along the shortest way. When the setpoint of Position exceeds one rotation cycle, the target position is the remainder of the setpoint divided by the rotation cycle. At the moment the instruction is executed, the relationship between the position reference after the current axis velocity decelerates to zero and the target position is determined. If it is within half a cycle in the positive direction, the axis trajectory is consistent with the Positive direction. If it is within half a cycle in the negative direction, the axis trajectory is consistent with the Negative direction. If it is exactly half the cycle length, the axis moves in the positive direction.
  - Direction = 1 (DIR_POSITIVE)
    The axis moves to the target position in the positive direction. When the setpoint of Position exceeds one rotation cycle, the target position is the remainder of the setpoint divided by the rotation cycle. For a rotary axis, the positive direction is the direction in which the position (user units) increases within one cycle. If the current velocity is greater than zero, the axis will continue to move in the current direction until it stops at the position specified by Position. If the current velocity is less than zero, the axis will decelerate to zero and then start moving in the positive direction. If the setpoint of Position is the same as the current position reference, the axis will not move.
  - Direction = 2 (DIR_NEGATIVE)
    The axis moves to the target position in the negative direction. When the setpoint of Position exceeds one rotation cycle, the target position is the remainder of the setpoint divided by the rotation cycle. For a rotary axis, the negative direction is the direction in which the position (user units) decreases within one cycle. If the current velocity is less than zero, the axis will continue to move in the current direction until it stops at the position specified by Position. If the current velocity is greater than zero, the axis will decelerate to zero and then start moving in the opposite

direction. If the setpoint of Position is the same as the current position reference, the axis will not move.

- Input parameter Direction = 3 (DIR_CURRENT)
  The axis moves to the target position in the direction of the previous movement on a rising edge of Execute. When the setpoint of Position exceeds one rotation cycle, the target position is the remainder of the setpoint divided by the rotation cycle. During the first power-on, the axis moves in the positive direction (target velocity greater than zero). If the setpoint of Position is the same as the current position reference, the axis will not move.

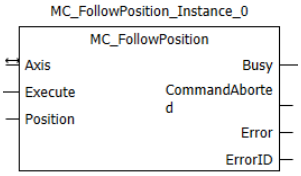
- **Example of displacement calculation:**
  Assume the rotation cycle is 360°.

Direction	Start position	Position	ModuloWindow	Displacement
DIR_POSITIVE	90	0	1	270
DIR_POSITIVE	90	360	1	630
DIR_POSITIVE	90	89.5	1	-0.5
DIR_POSITIVE	90	88.5	1	358.5
DIR_NEGATIVE	90	0	1	-90
DIR_NEGATIVE	90	360	1	-450
DIR_NEGATIVE	90	90.8	1	0.8
DIR_NEGATIVE	90	92	1	-358

18.3.3.3.34 MC_FollowPosition

This instruction enables the position reference to be written into the axis without any checks.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_FollowPosition	Specify axis position without any checks		MC_FollowPosition_Instance_0(Axis :=, Execute :=, Position :=, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.
Position	Position	LREAL	Positive number, negative number, or 0	0	Specify the position reference [u] for the axis in the current cycle.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is being executed.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus-controlled axis:

  - Unlike the MC_MoveAbsolute instruction, the position reference is sent to the axis in each task cycle after this instruction is executed on a rising edge. This instruction can be used for camming.

- When the current axis state is AXIS_STOPPING or AXIS_HOMING, triggering this instruction causes it to be immediately aborted and terminated.
- The velocity reference of the axis is: ΔL / Δt (ΔL is the difference between the position reference in current cycle and the position reference in previous cycle; Δt is the period time).
- For this instruction, the velocity reference is calculated based on the difference between the position reference in current cycle and the position reference in previous cycle. Therefore, after this instruction is executed, use the MC_SetPosition instruction with caution to prevent excessive position differences between adjacent cycles, which may cause overspeed.
- When this instruction is executed: ① If the control mode is MC_CSP mode, input the Position value with caution to prevent the difference between the current axis position and the Position value from causing excessive single-period displacement, which may cause a servo fault; ② If the control mode is MC_CSV mode, input the Position value with caution to prevent the difference between the current axis position and the Position value from causing the velocity back-calculated based on the single-period displacement to be greater than the maximum servo velocity, which may result in a deviation between the actual position and the position reference.
- This instruction does not apply any velocity limitation to either rotary or linear axes; however, linear axes are subject to software limit.
- For this instruction, the motion direction in one cycle is the Shortest way, and the motion direction in half a cycle is the positive direction.
- This instruction does not allow target positions beyond the limit. If the position is beyond the software limit, the axis will stop with the maximum deceleration.
- The axis type must be control axis.
- Control mode switching between instructions does not rely on the MC_SetControllerMode instruction. You can directly call instructions under different control modes. Each instruction internally checks if the current mode meets the requirements and automatically switches the mode if necessary.
- The instruction can be aborted by other instructions during execution, at which point the output parameter CommandAborted is set.
- During the execution of this instruction, the MC_MoveSuperImposed instruction cannot be used; once the MC_MoveSuperImposed instruction is triggered, CommandAborted is set for the MC_MoveSuperImposed instruction.

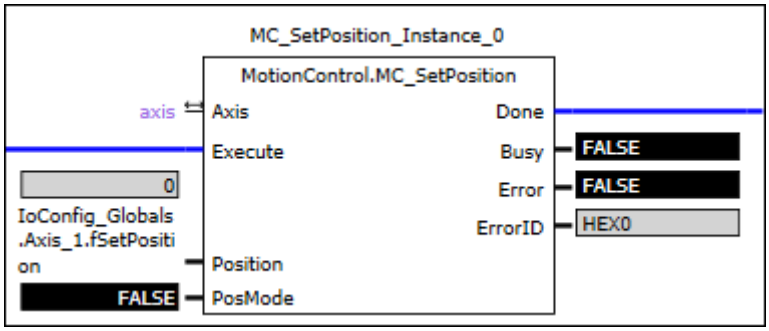
  Local pulse axis:

- This instruction can be used for local pulse axes.

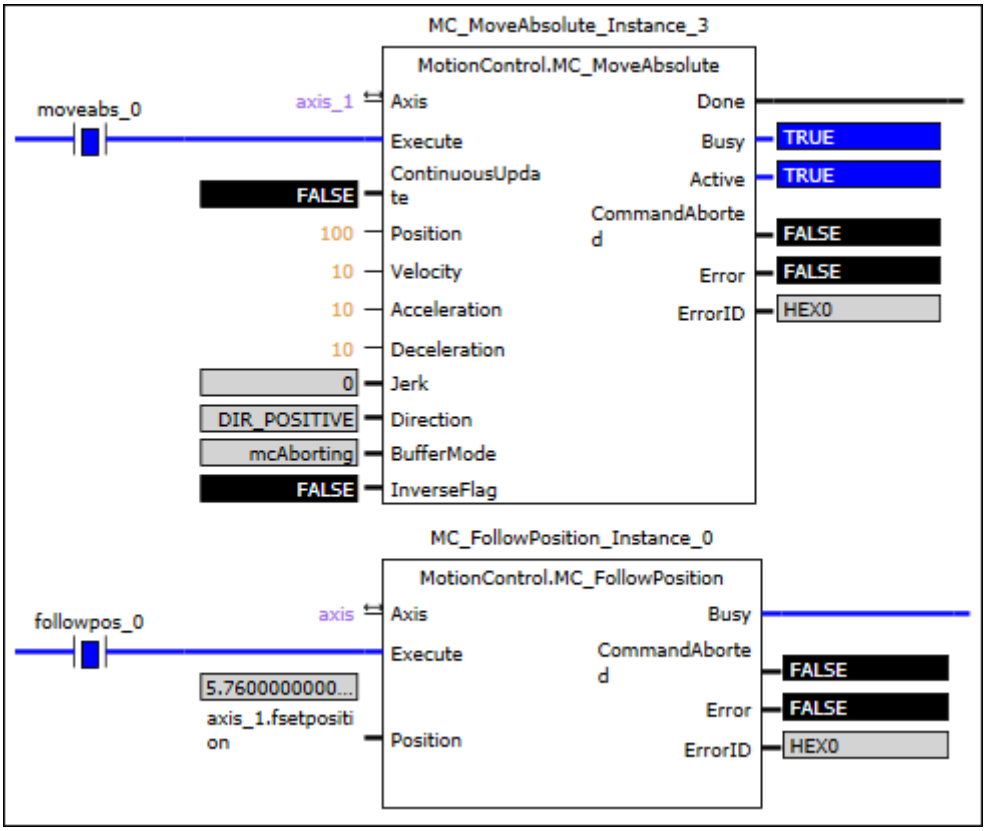
- **Example**

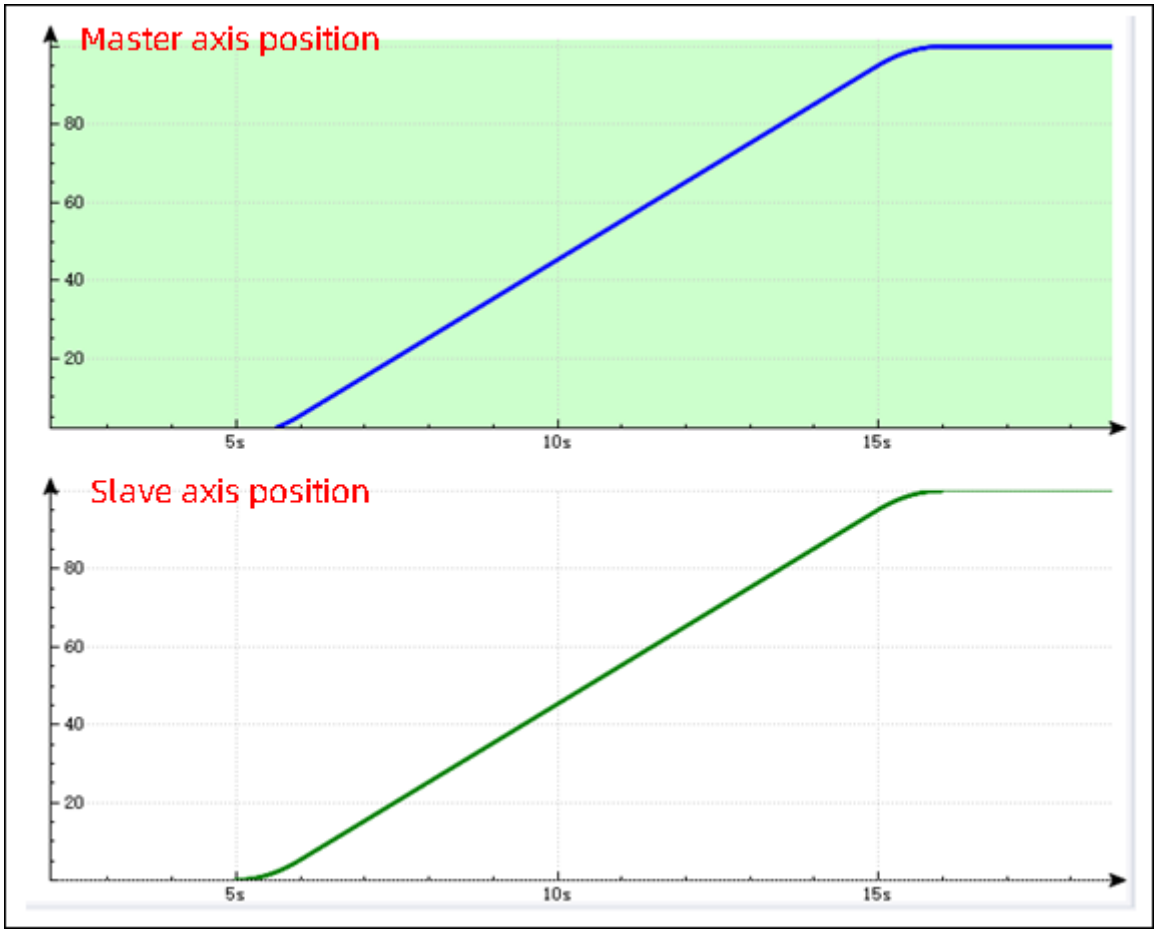
  This instruction enables simple following motion. Configure two axes, where Axis_1 is a virtual axis (master axis), and Axis is a real axis (slave axis).

  First, execute the MC_SetPostion instruction to assign the position of the master axis to the slave axis, ensuring that their start positions are synchronized.



Execute the MC_MoveAbsolute instruction on Axis_1, execute the MC_FollowPosition instruction on Axis, and assign the position reference (fSetPosition) for the Axis_1 to the input parameter Position of the MC_FollowPosition instruction, thus achieving synchronization.

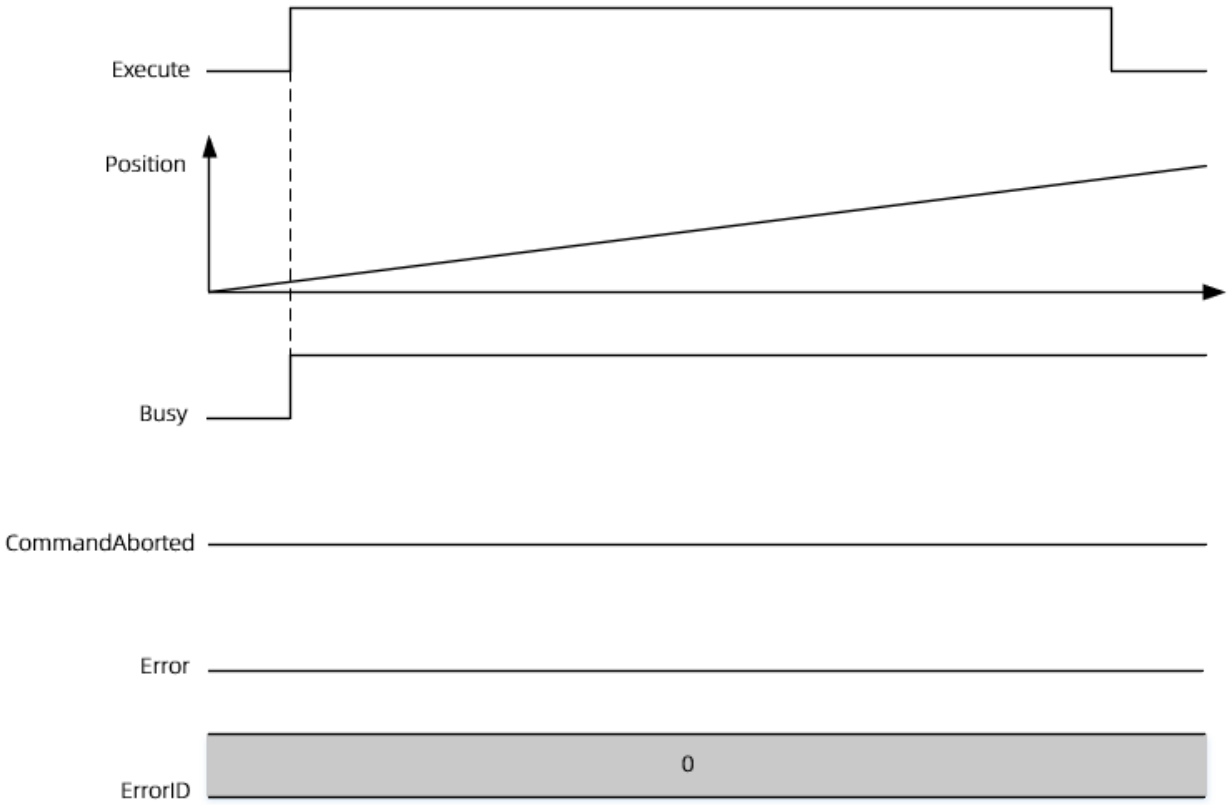




- **Timing diagram**

  After the instruction is executed on a rising edge of Execute, the velocity reference is sent to the axis i n each task cycle.

  When the Execute signal is TRUE, the output parameter Busy shows the value TRUE. If this instruction is aborted by another control instruction, the output parameter Busy shows the value FALSE, and the output parameter CommandAborted is set.



- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.35 MC_FollowVelocity

This instruction specifies an axis velocity without any checks.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_FollowVelocity	Specify axis velocity without any checks	MC_FollowVelocity_Instance_0 MC_FollowVelocity Axis Execute Velocity Busy CommandAborted Error ErrorID	MC_FollowVelocity_Instance_0(Axis :=, Execute :=, Velocity :=, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.
Velocity	Velocity	LREAL	Positive number, negative number, or 0	0	Specify the velocity reference [u/s] for an axis.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is being executed.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus-controlled axis:

  - Unlike the MC_MoveVelocity instruction, a velocity reference is sent to the axis in each task cycle after this instruction is triggered.

- When the parameter Busy shows the value TRUE, the modified input parameter Velocity takes effect immediately without re-execution of the instruction.
- After the output parameter Busy is set, the axis is in SYNCHRONIZED_MOTION state.
- When the current axis state is AXIS_STOPPING or AXIS_HOMING, triggering this instruction causes it to be immediately aborted and terminated.
- If the position is beyond the software limit, the axis will stop with the maximum deceleration.
- The axis type must be control axis.
- Control mode switching between instructions does not rely on the MC_SetControllerMode instruction. You can directly call instructions under different control modes. Each instruction internally checks if the current mode meets the requirements and automatically switches the mode if necessary.
- The instruction can be aborted by other instructions during execution, at which point the output parameter CommandAborted is set.
- During the execution of this instruction, the MC_MoveSuperImposed instruction cannot be used; once the MC_MoveSuperImposed instruction is triggered, CommandAborted is set for the MC_MoveSuperImposed instruction.

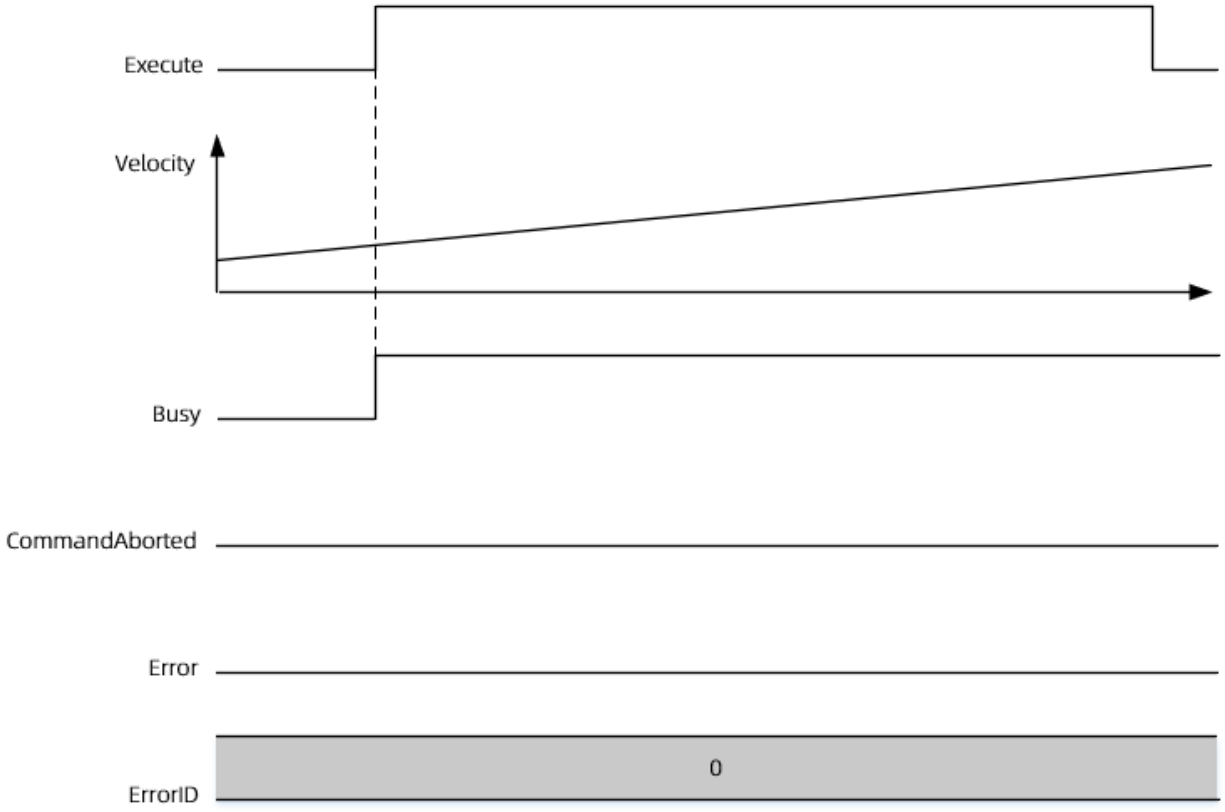
  Local pulse axis:

- This instruction can be used for local pulse axes.

- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Busy of the function block indicates that the current function block is being executed, and the parameter CommandAborted indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

  After the instruction is executed on a rising edge of Execute, a velocity reference is sent to the axis in each task cycle.
  When the Execute signal is TRUE, the output parameter Busy shows the value TRUE. If this instruction is aborted by another control instruction, the output parameter Busy shows the value FALSE, and the output parameter CommandAborted is set.



- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.36 MC_FollowPositionVelocity

The instruction enables the position and velocity reference to be written into the axis without any checks. In CSP control mode, the specified position parameter takes effect. In CSV control mode, the specified velocity parameter takes effect.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_FollowPositionVelocity	Specify axis position and velocity without any checks	MC_FollowPositionVelocity_Instance_0 MC_FollowPositionVelocity Axis Execute Position Velocity Busy CommandAborted Error ErrorID	MC_FollowPositionVelocity_Instance_0(Axis :=, Execute :=, Position :=, Velocity :=, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.
Position	Position	LREAL	Positive number, negative number, or 0	0	Specify the position reference [u] for the axis in the current cycle.
Velocity	Velocity	LREAL	Positive number, negative number, or 0	0	Specify the velocity reference [u] for the axis in the current cycle.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is being executed.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus-controlled axis:

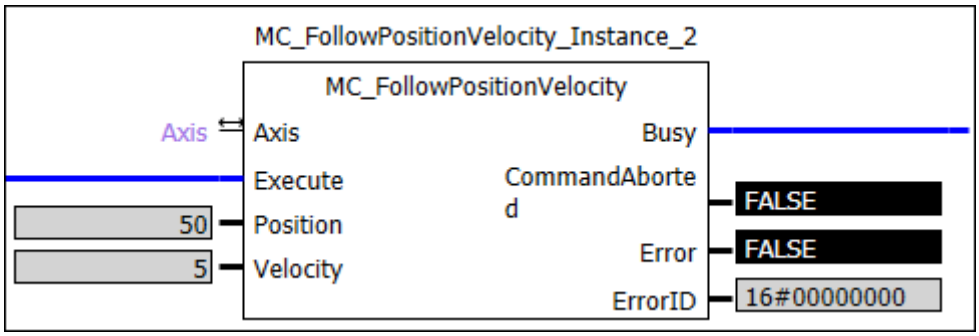
  - After this instruction is triggered on a rising edge of Execute, the position and velocity references are sent to the axis in each task cycle, and they take effect immediately once being modified during the motion.
  - If the axis is not in CSP or CSV mode when the instruction is triggered on a rising edge, the instruction will terminate immediately with a fault message.
  - When the axis is in CSP mode, the position reference refreshed in each cycle takes effect**.** If the input velocity does not match the actual rate of position change, the velocity reference may be not equal the actual velocity.
  - In CSP mode, the motion direction of a rotary axis over one cycle is the Shortest way. When the position difference deltaPos is greater than half a PositionPeriod in the positive direction, the axis moves in the negative direction by PositionPeriod - deltaPos to the target position. When the position difference deltaPos is less than half a PositionPeriod in the negative direction, the axis moves in the positive direction by PositionPeriod + deltaPos to the target position.
  - In CSP mode, when this instruction is executed for position setting after the MC_SetPosition instruction is executed, the actual axis position is not the position reference but the sum of the SetPosition and the position difference between adjacent cycles.
  - When the axis is in CSV mode, the velocity reference refreshed in each cycle takes effect. At this time, the position reference aligns with the actual position in each cycle, and the input position does not take effect.
  - After this instruction is triggered, the axis is in synchronized motion. When the control mode of the axis switches from CSP to CSV or from CSV to CSP, the instruction execution continues, and the axis moves according to the current mode and parameter references. In case of switching to another mode, the instruction will terminate immediately with a fault message.
  - During the execution of this instruction, the MC_MoveSuperImposed instruction cannot be used; once t he MC_MoveSuperImposed instruction is triggered, CommandAborted is set for the MC_MoveSuperImposed instruction.
  - When the axis is in CSP or CSV control mode, and the corresponding 607A/60FF is not configured, the function block is terminated with a fault message.
  - When the current axis state is Stopping or Homing, this instruction is immediately aborted and terminated.
  - This instruction does not allow target positions beyond the limit. If the position is beyond the software limit, the axis will stop at the maximum deceleration.
  - The function block has no buffer mode. It can abort or be aborted by other motion function blocks.

Local pulse axis:

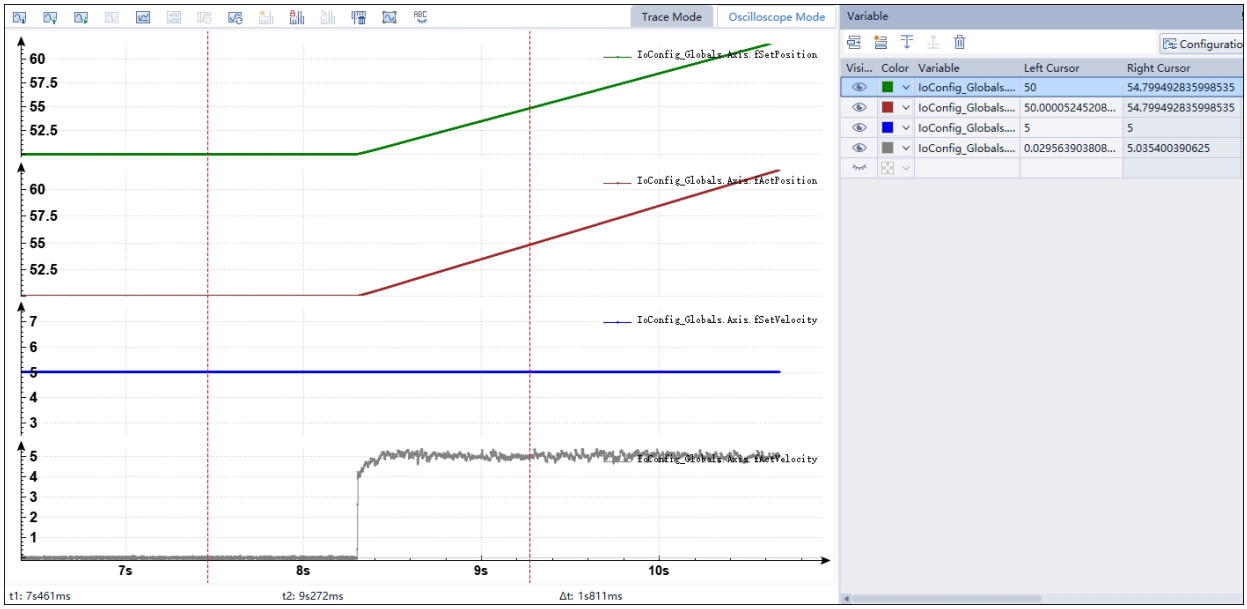
- This instruction can be used for local pulse axes in CSP mode. If the position reference is too large, causing the velocity to exceed 200 kHz, the function block will be terminated with a fault message.

- **Example**

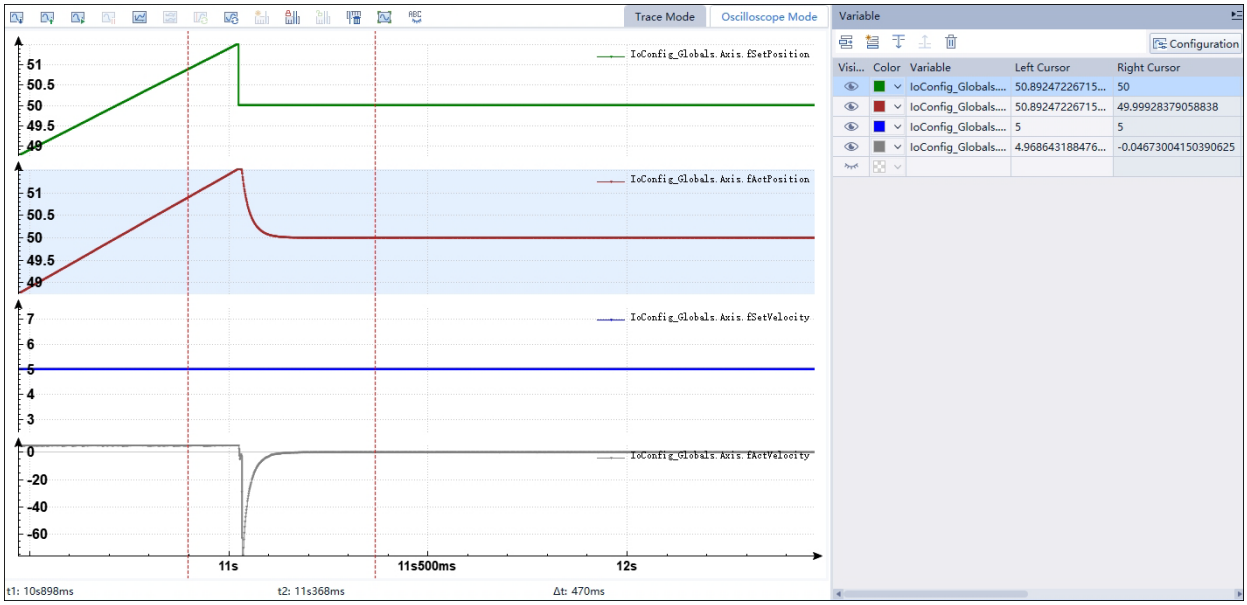
  This instruction enables position following and velocity following during the motion. The following is an instance of how to execute the MC_FollowPositionVelocity instruction. The set position is 50, and the velocity is 5.



  After the instruction is triggered, the axis first moves to position 50 in CSP mode, and then switches to CSV mode without re-executing the function block and moves at the velocity of 5.



  Next, the axis switches to CSP mode and moves to position 50 according to the set position.

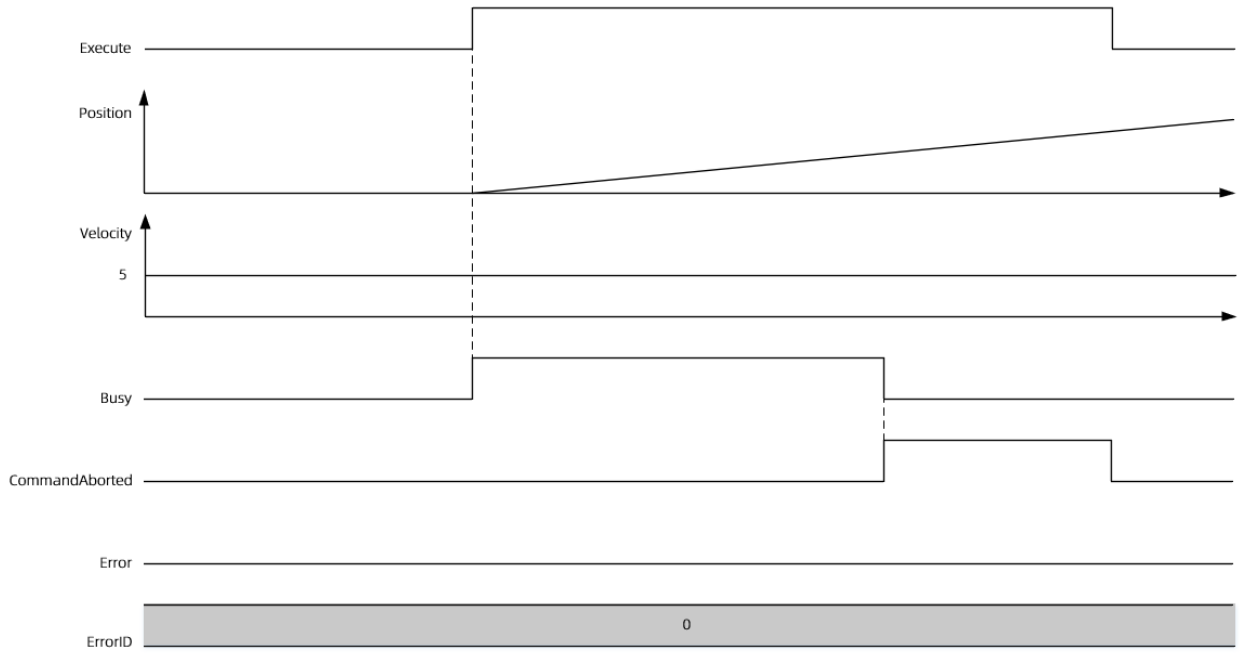


- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Busy of the function block indicates that the current function block is being executed, and the parameter CommandAborted indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

  After the instruction is executed on a rising edge of Execute, the position and velocity references are sent to the axis in each task cycle.

  When the Execute signal is TRUE, the output parameter Busy shows the value TRUE. If this instruction is aborted by another control instruction, the output parameter Busy shows the value FALSE, and the output parameter CommandAborted is set.



- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

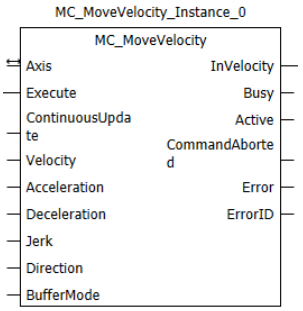
- **Precautions**

The set velocity must satisfy the change in set position, i.e., Set velocity = The first derivative of the difference in set positions between adjacent task cycles with respect to time. For example, if the set positions for two adjacent cycles are the same, the velocity will be set to 0.

18.3.3.3.37 MC_MoveVelocity

This instruction controls the axis velocity in CSP mode.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_MoveVelocity	Velocity planning		MC_MoveVelocity _Instance_0(Axis :=, Execute :=, ContinuousUpdate :=, Velocity :=, Acceleration :=, Deceleration :=, Jerk :=, Direction:=, BufferMode :=, InVelocity =>, Busy =>, Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.
ContinuousUpdate	Continuous update	BOOL	TRUE, FALSE	FALSE	Continuous parameter update effective flag. Prerequisite: The parameter ContinuousUpdate shows the value TRUE and the parameter Active shows the value TRUE. Applicable parameters: Velocity, Acceleration, Deceleration, Jerk, Direction. Note: The modified parameter values must differ from the original parameter values by more than 0.01.
Velocity	Velocity	LREAL	Positive number or 0	0	Maximum velocity [u/s] for the motion. When the velocity is 0, the InVelocity flag of the axis is TRUE, but the axis is standstill.
Acceleration	Acceleration	LREAL	Positive number	0	Acceleration [u/s^2] of the acceleration phase.
Deceleration	Deceleration	LREAL	Positive number	0	Deceleration [u/s^2] of the deceleration phase.
Jerk	Jerk	LREAL	Positive number	0	Slope [u/s^3] of acceleration or deceleration (Valid for S-curve planning only).
Direction	Motion direction	18.3.3.2.2.9 MC_DIRECTION: Motion Direction Enumeration	1: DIR_POSITIVE 2: DIR_NEGATIVE 3: DIR_CURRENT	1	1: DIR_POSITIVE (Positive direction). 2: DIR_NEGATIVE (Negative direction). 3: DIR_CURRENT (Current direction).
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting 1: mcBuffered	0	0: mcAborting (Aborting). 1: mcBuffered (Buffered).

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
InVelocity	Velocity reference reached	BOOL	TRUE, FALSE	FALSE	TRUE when the axis reaches the velocity reference.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Controlling	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled by the current FB.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity	Velocity	Any velocity that exceeds the motion parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	No correction.

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Acceleration	Acceleration	Any acceleration that exceeds the motion parameter limit will be automatically corrected to the acceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration less than 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration equal to 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration equal to 0 will be automatically corrected to 10 times the velocity.
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the deceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration less than 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration equal to 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration equal to 0 will be automatically corrected to 10 times the velocity.
Jerk	Jerk	If the motion profile is set to S-curve, any jerk that exceeds the software limit will be automatically corrected to the jerk limit.	If the motion profile is set to S-curve, any jerk less than 0 will be automatically corrected to 10% of the jerk limit.	If the motion profile is set to S-curve, the jerk equal to 0 will be automatically corrected to 10% of the jerk limit.

- **Function**

  Bus-controlled axis:

  ◦ This instruction controls the velocity based on position. It starts velocity control on a rising edge of Execute and switches the axis state to AXIS_CONTINUOUS_MOTION.

  ◦ When the input parameter Direction is specified as DIR_POSITIVE (Positive direction), the axis moves in the positive direction. When input parameter Direction is specified as DIR_NEGATIVE (Negative direction), the axis moves in the negative direction. When the Direction is specified as DIR_CURRENT, the axis moves in the direction in which it was traveling previously.

  ◦ When this instruction is aborted by another motion control instruction, the CommandAborted is set, and the subsequent instruction starts planning the axis motion.

- When used with the MC_MoveSuperimposed instruction, the output parameter InVelocity is set to TRUE as soon as this instruction's contribution to velocity equals the value of its input parameter Velocity.
- Online update is not supported for the buffer mode, which is determined by the mode specified when this instruction is triggered for the first time. Online update is supported for Velocity, Acceleration, Deceleration, Jerk, and Direction. If the change in a variable is greater than 0.01, the variable will be considered modified, and the motion will be planned according to the modified variable.
- This instruction supports MC_ABORTING and MC_BUFFERED modes only. During execution of this instruction, do not modify the input parameter BufferMode to MC_BUFFERED mode for re-execution; otherwise, the function block will report a fault.
- The starting velocity of a buffered instruction after this instruction is the velocity when the output parameter InVelocity is set to TRUE.
- During execution of this instruction, you can call MC_Stop, MC_Halt, or MC_ImmediateStop instruction to stop the motion of the axis.
- This instruction automatically switches modes internally, so control mode switching between instructions does not rely on the MC_SetControllerMode instruction. You can directly call instructions under different control modes. Each instruction internally checks if the current mode meets the requirements and automatically switches the mode if necessary.
- During the execution of this instruction, the MC_MoveSuperImposed instruction can be used.
- When the instruction is triggered on a rising edge of the input parameter Execute, the input parameters are checked for validity. If any of invalid parameters is detected, it will be automatically corrected by the function block without a fault message; any buffered instructions after this instruction will be cleared.
- When the axis is enabled after PLC cold reset and the MC_Home instruction is executed using No. 35 homing method, if this instruction is executed with DIR_CURRENT being set, the motion direction of this instruction will be affected.

  Local pulse axis:

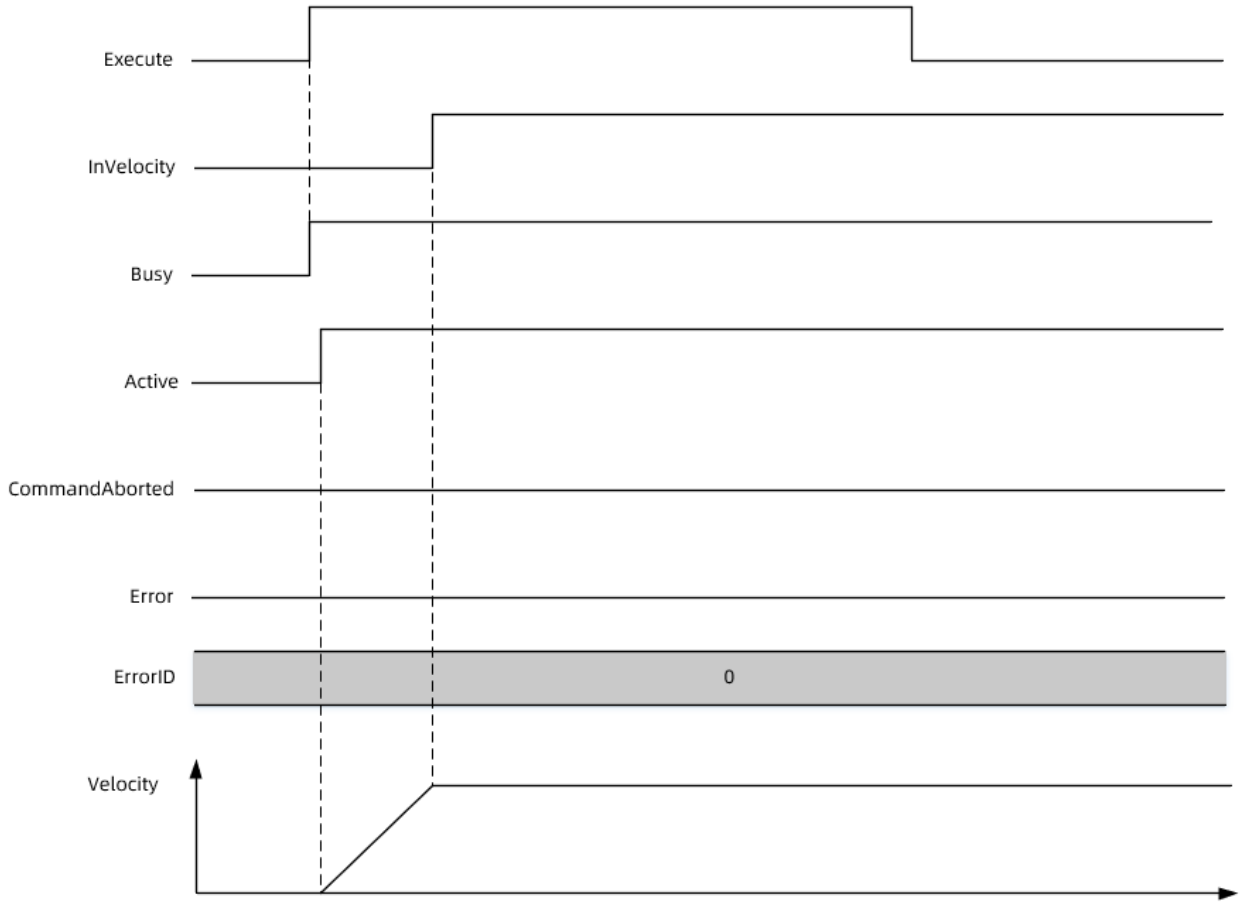
- This instruction can be used for local pulse axes.
- When the local pulse axis is used with Y0 or Y1 single-ended output, it is not recommended to use this instruction. ① No pulse output; ② No position feedback.

- **Timing diagram**

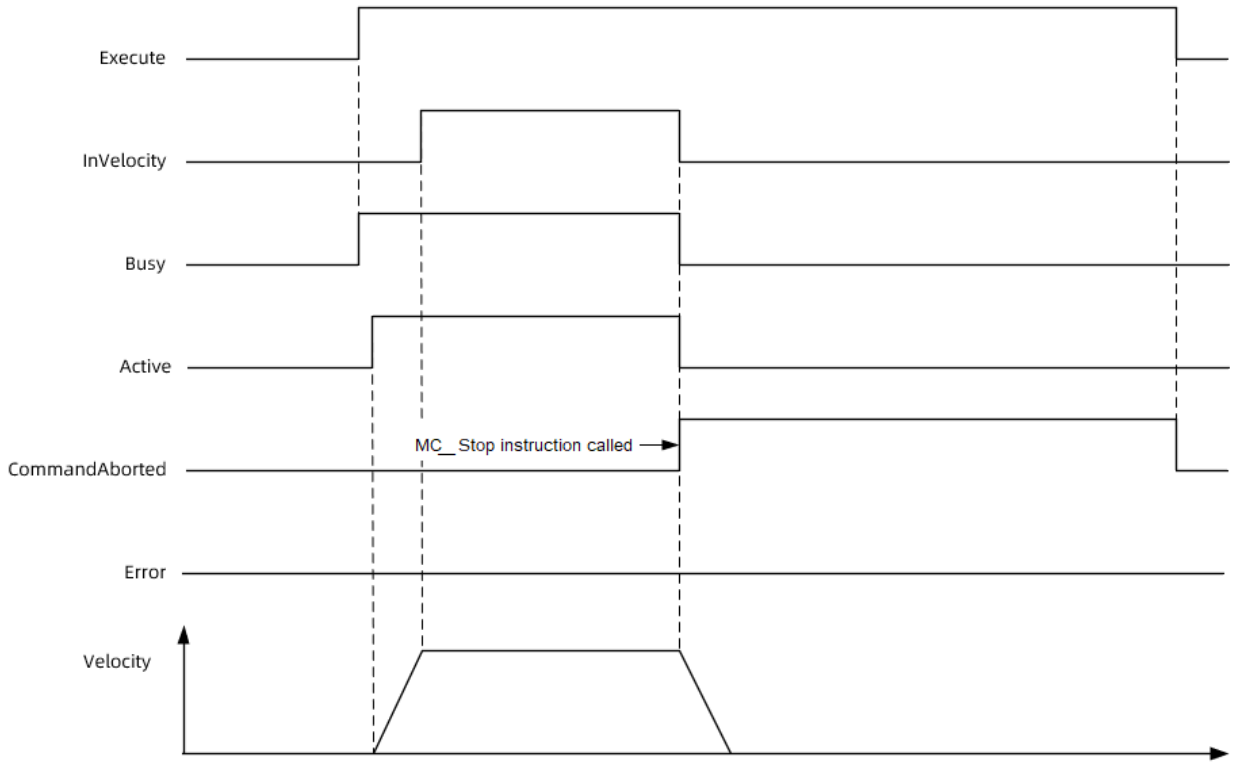
  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates the instruction has been executed successfully; the parameter Busy indicates that the current function block is being executed; the parameter CommandAborted indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

  The parameter Busy shows the value TRUE on a rising edge of Execute. In the next cycle, the output parameter Active shows the value TRUE.

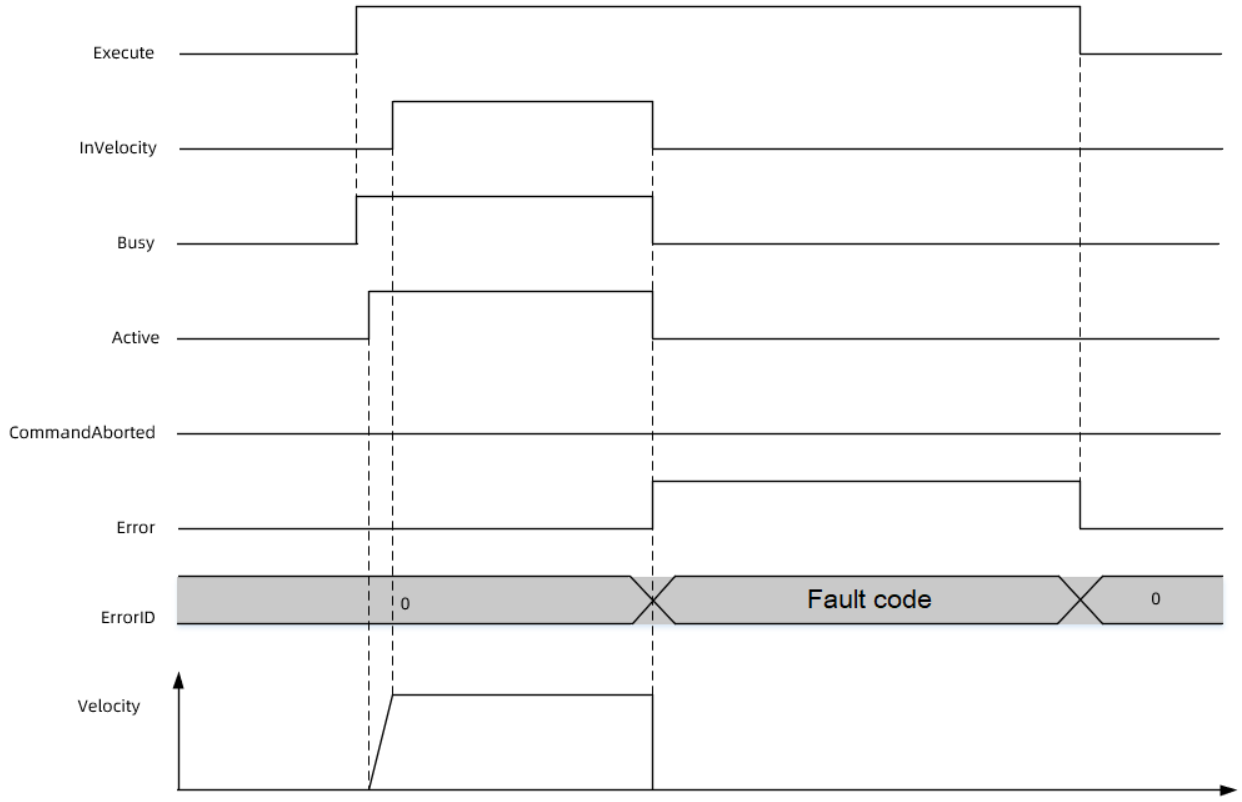
  - The MC_MoveVelocity instruction is executed to implement continuous motion based on the T-curve when the axis is in StandStill state.



◦ Axis motion is aborted by the MC_Stop instruction.



◦ The drive malfunctions during the axis acceleration.



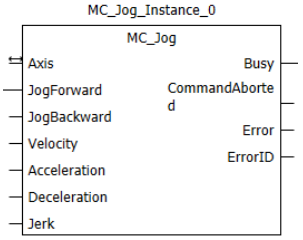
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.38 MC_Jog

This instruction enables the axis to jog at the specified target velocity.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_Jog	Jog		MC_Jog _Instance_0(Axis :=, JogForward :=, JogBackword :=, Velocity :=, Acceleration :=, Deceleration :=, Jerk :=, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
JogForward	Jog forward	BOOL	TRUE, FALSE	FALSE	TRUE: The axis starts moving forward. FALSE: The axis stops moving forward.
JogBackward	Jog backward	BOOL	TRUE, FALSE	FALSE	TRUE: The axis starts moving backward. FALSE: The axis stops moving forward.
Velocity	Target velocity	LREAL	Positive number or 0	0	Target velocity [u/s] for the motion.
Acceleration	Acceleration	LREAL	Positive number	0	Acceleration [u/s^2] of the acceleration phase.
Deceleration	Deceleration	LREAL	Positive number	0	Deceleration [u/s^2] of the deceleration phase.
Jerk	Jerk	LREAL	Positive number	0	Slope [u/s^3] of acceleration or deceleration (Valid for S-curve planning only). The value 0 indicates T-curve acceleration or deceleration.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity	Velocity	Any velocity that exceeds the commissioning parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	No correction.
Acceleration	Acceleration	Any acceleration that exceeds the commissioning parameter limit will be automatically corrected to the acceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration less than 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration equal to 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration equal to 0 will be automatically corrected to 10 times the velocity.

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Deceleration	Deceleration	Any deceleration that exceeds the commissioning parameter limit will be automatically corrected to the deceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration less than 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration equal to 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration equal to 0 will be automatically corrected to 10 times the velocity.
Jerk	Jerk	If the motion profile is set to S-curve, any jerk that exceeds the software limit will be automatically corrected to the jerk limit.	If the motion profile is set to S-curve, any jerk less than 0 will be automatically corrected to 10% of the jerk limit.	If the motion profile is set to S-curve, the jerk equal to 0 will be automatically corrected to 10% of the jerk limit.

- **Function**

  Bus-controlled axis:

  - After this instruction is triggered, the setpoints of the input parameters Velocity, Acceleration, Deceleration, and Jerk are latched, and the instructions controls the velocity for jogging and switches the axis state to AXIS_CONTINUOUS_MOTION.
  - If you want the axis to move forward, set the input parameter JogForward to TRUE. If you want the axis to move backward, set the input parameter JogBackward to TRUE. When both JogForward and JogBackward are set to TRUE, the axis remains standstill. After the axis is moving with JogForward or JogBackward being set, the axis will decelerate to stop and be kept in DiscreteMotion mode during the deceleration if both JogForward and JogBackward are set to TRUE simultaneously.
  - For the input parameters Velocity, Acceleration, or Deceleration of this instruction, if the setpoint of any parameter is greater than the maximum limit in the software, the parameter will be automatically corrected by the function block without a fault message.
  - If the axis hits a limit switch while moving in one direction, the instruction will report a fault, and the axis enters ErrorStop state. If the software is configured to not report faults associated with limit switches, the axis state will automatically return to StandStill after deceleration to stop.
  - When the current axis state is Stopping or Homing, this instruction is immediately aborted and terminated.
  - The control object for this instruction must be control axis. During the execution of this instruction, if there is a fault on the axis, the function block will report the fault and exit.

- This instruction automatically switches control modes internally, so control mode switching between instructions does not rely on the MC_SetControllerMode instruction. You can directly call instructions under different control modes. Each instruction internally checks if the current mode meets the requirements and automatically switches the mode if necessary.
- During the execution of this instruction, the MC_MoveSuperImposed instruction cannot be used; once the MC_MoveSuperImposed instruction is triggered, CommandAborted is set for the MC_MoveSuperImposed instruction.

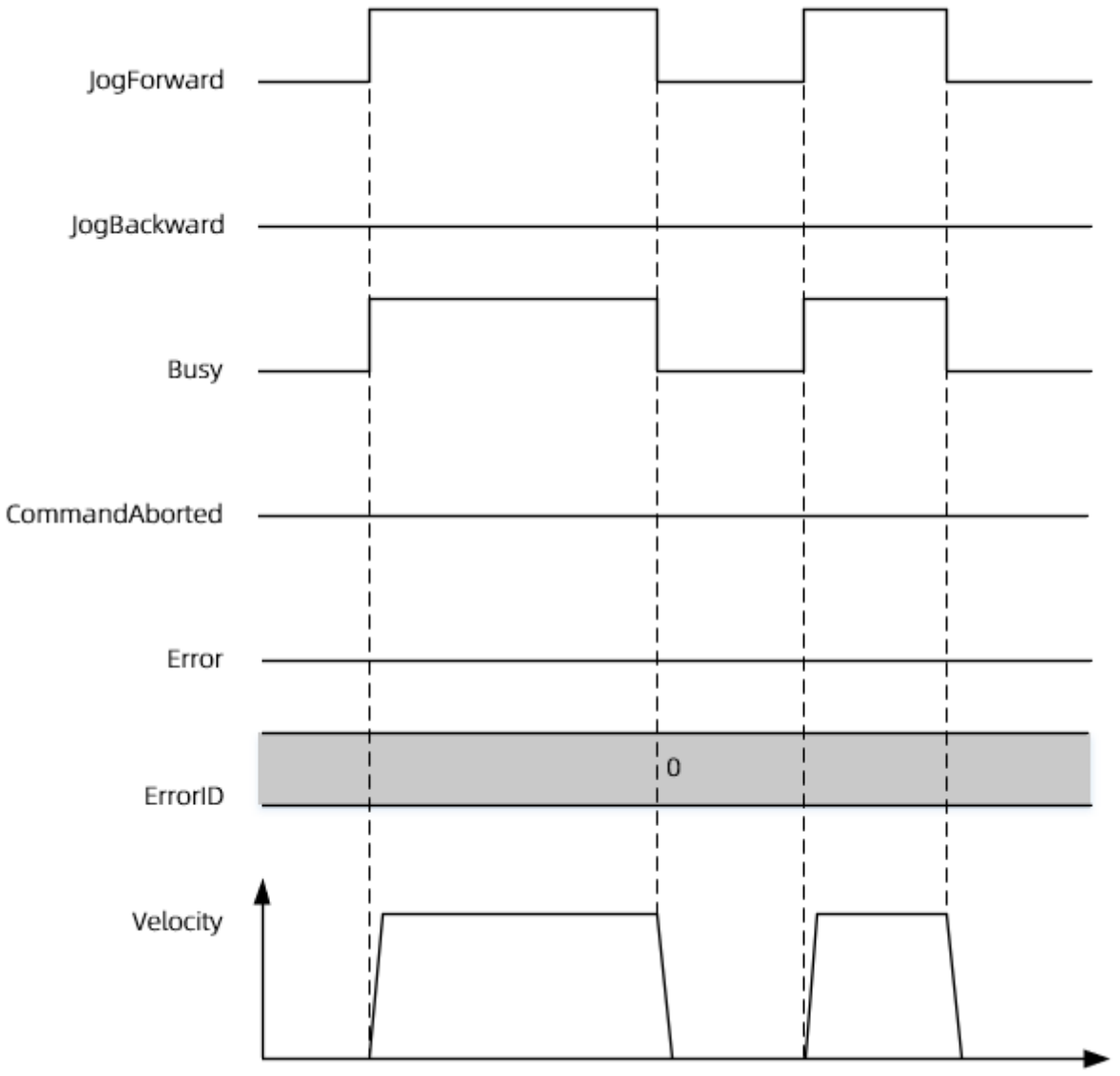
Local pulse axis:

- This instruction can be used for local pulse axes.

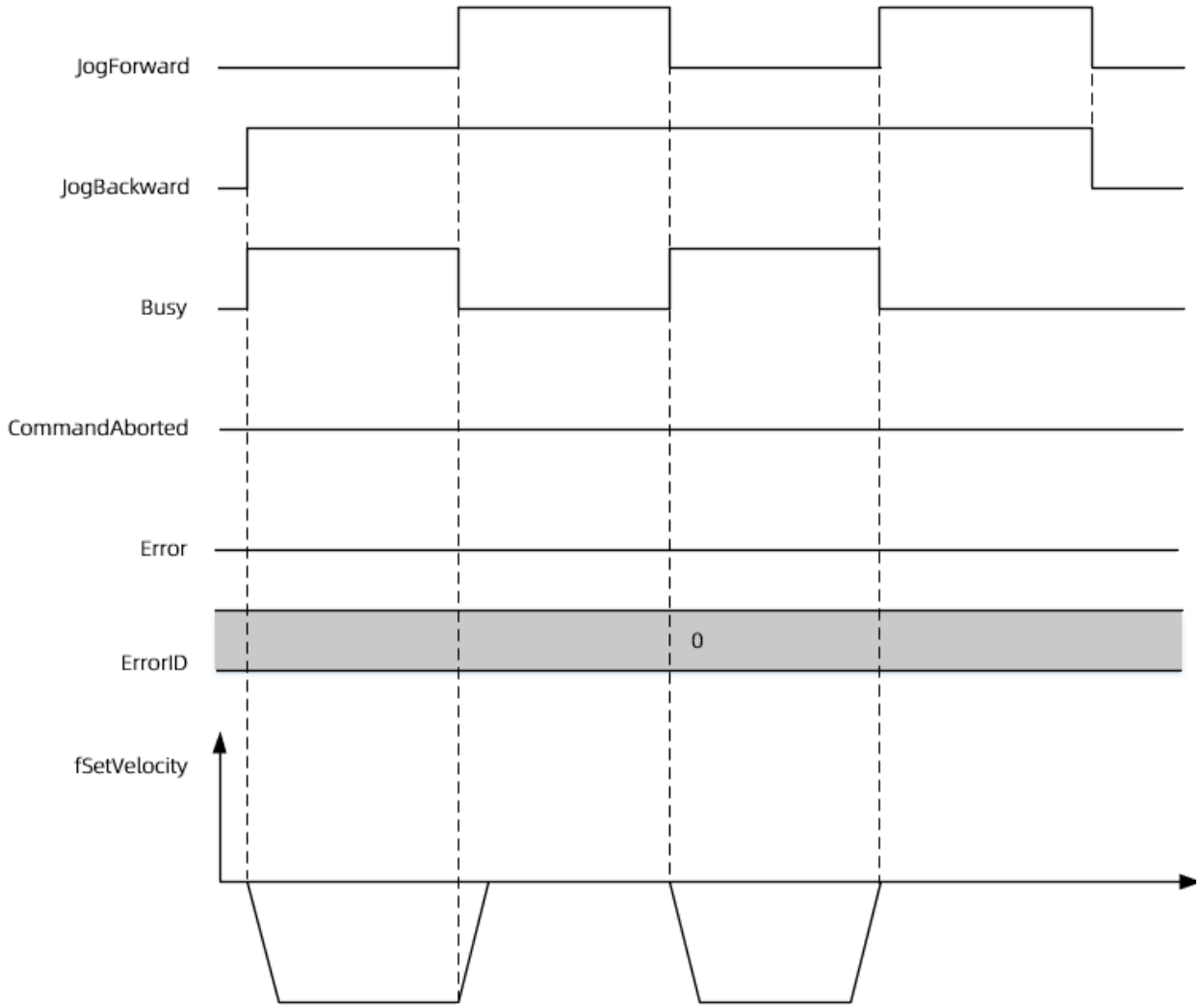
- **Timing diagram**

When JogForward or JogBackward is enabled, the parameter Busy shows the value TRUE.
When the axis starts decelerating to stop on a falling edge of JogForward or JogBackward, the parameter Busy shows the value FALSE.
When this instruction is aborted by another instruction, the CommandAborted is set to TRUE, and the Busy is set to FALSE.

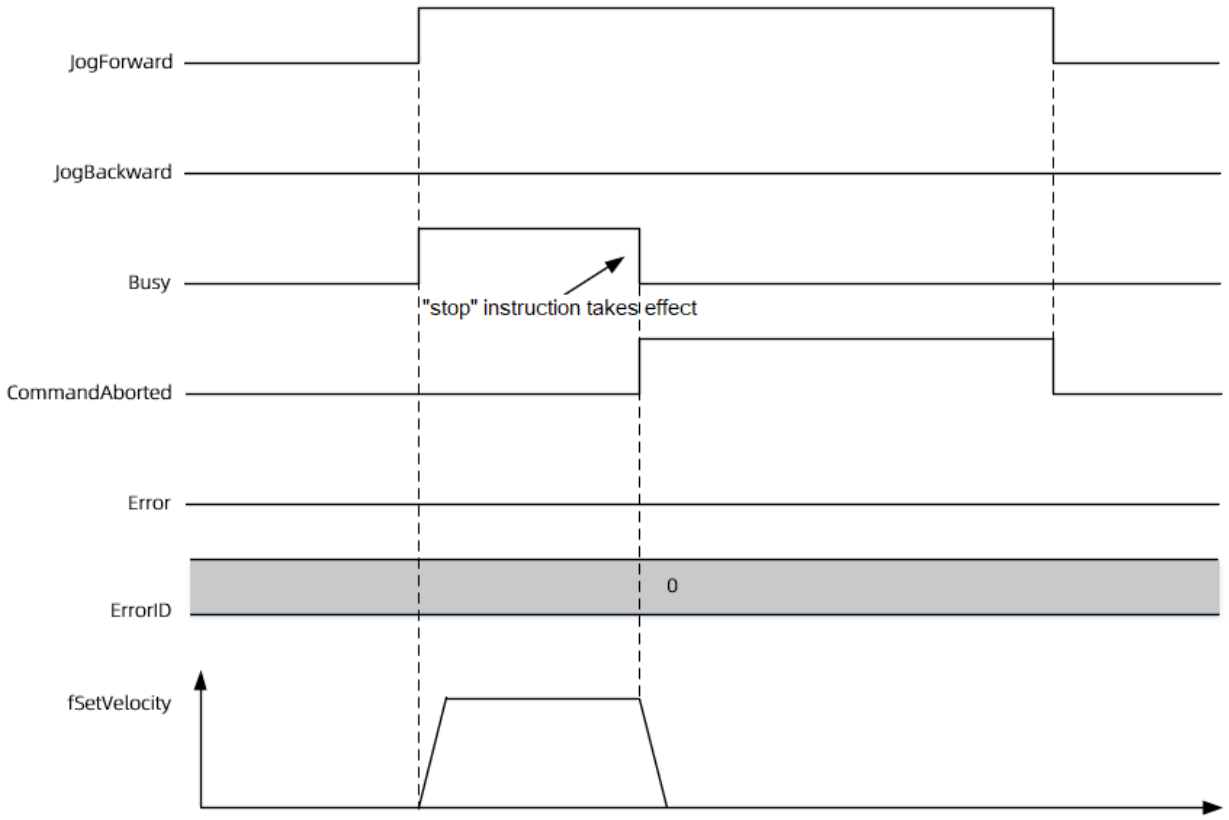
- Timing diagram when the JogForward is set to TRUE



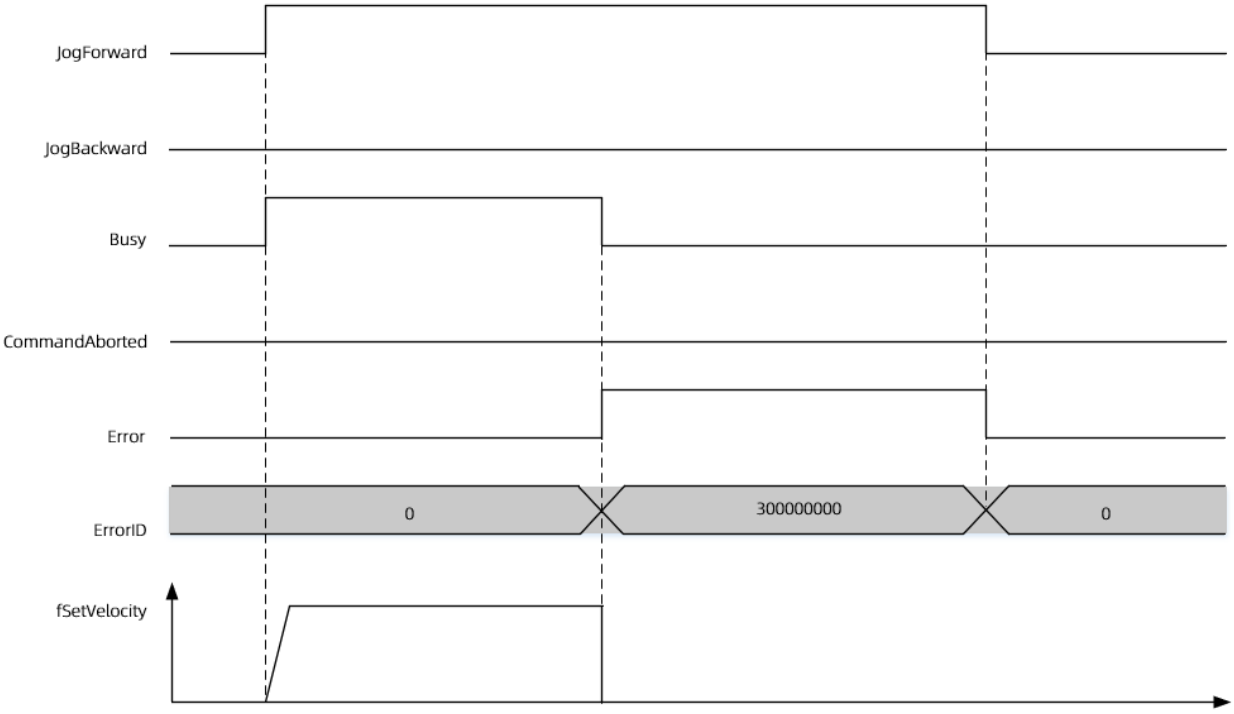
- Timing diagram when the JogBackward is set to TRUE



◦ Timing diagram when the instruction is aborted by the MC_Stop instruction during execution



◦ Timing diagram when the axis reports a fault



- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

- **Precautions**

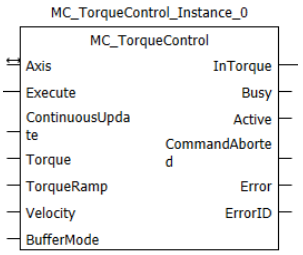
  ◦ This instruction can be followed by a buffered instruction. The buffered motion starts after the commissioning of this instruction ends.

◦ The axis is in AXIS_CONTINUOUSMOTION state during the jogging, and AXIS_DISCRETEMOTION during the deceleration phase.

18.3.3.3.39 MC_TorqueControl

This instruction controls the torque.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_TorqueControl	Torque control		MC_TorqueControl_Instance_0(Axis :=, Execute :=, ContinuousUpdate :=, Torque :=, TorqueRamp :=, Velocity :=, BufferMode :=, InTorque =>, Busy =>, Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
ContinuousUpdate	Continuous update	BOOL	TRUE, FALSE	FALSE	Continuous parameter update effective flag. Prerequisite: The ContinuousUpdate is set to TRUE. Applicable parameters: Torque, TorqueRamp, Velocity.
Torque	Target torque	LREAL	Torque value: Target torque to output to the servo drive in increments of 0.1%.	300	Positive torque cannot be greater than the fMaxPositiveTorque of the axis structure, and the absolute value of negative torque cannot be greater than the fMaxNegativeTorque of the axis structure. Torque value: Target torque to output to the servo drive in increments of 0.1%.
TorqueRamp	Torque ramp	LREAL	Positive number or 0	0	Change rate of torque from the current value to the target torque. The value 0 indicates direct output at the target torque.
Velocity	Velocity	LREAL	Positive number or 0	0	Maximum velocity.
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting 1: mcBuffered	0	0: mcAborting (Aborting). 1: mcBuffered (Buffered).

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
InTorque	Target torque reached	BOOL	TRUE, FALSE	FALSE	TRUE when the target torque is reached.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Controlling	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled by the current FB.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity	Velocity	Any velocity that exceeds the motion parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	No correction.
Torque	Target torque	Any torque that exceeds the torque limit in the software will be automatically corrected to the torque limit.	No correction.	No correction.
TorqueRamp	Torque ramp	No limit.	Any torque ramp less than 0 will be automatically corrected to 0.	No correction.

- **Function**

  Bus-controlled axis:

  - This instruction switches the control mode to torque mode (MC_CST) for controlling the output torque of the servo motor. The target torque is specified in increments of 0.1%. The actual torque output by the drive is limited by the maximum positive/negative torque specified for the configuration parameters.
  - The input parameter Velocity is a positive value, and its direction depends on the torque or load.
  - When the input parameter ContinuousUpdate is set to TRUE, online modification is supported for Torque, TorqueRamp, and Velocity.
  - Before triggering this instruction, configure the PDOs 6060, 6061, 6071 and 6077; otherwise, the function block will report a fault.
  - When the actual torque deviation from the target torque is less than 5%, the parameter InTorque is set. The InTorque is set only when the condition is first met and will not be refreshed periodically thereafter.
  - Before using this instruction, check whether the servo device corresponding to the axis executing the instruction is a stepper drive. Torque control is not supported for stepper drives. If it is not a stepper drive, configure the corresponding PDOs.
  - If 607F is not configured, the Velocity is converted into pulse units by this instruction and written to the 607F object via the SDO channel. After the MC_TorqueControl instruction terminates, triggering the MC_Home instruction may limit the homing velocity.

- **Velocity control in torque mode:**
  - For Inovance servo drives

    The PDO to be configured for torque control is 6071 (Torque). If 6071 is not configured, the function block will report a fault.
  - For third-party drives

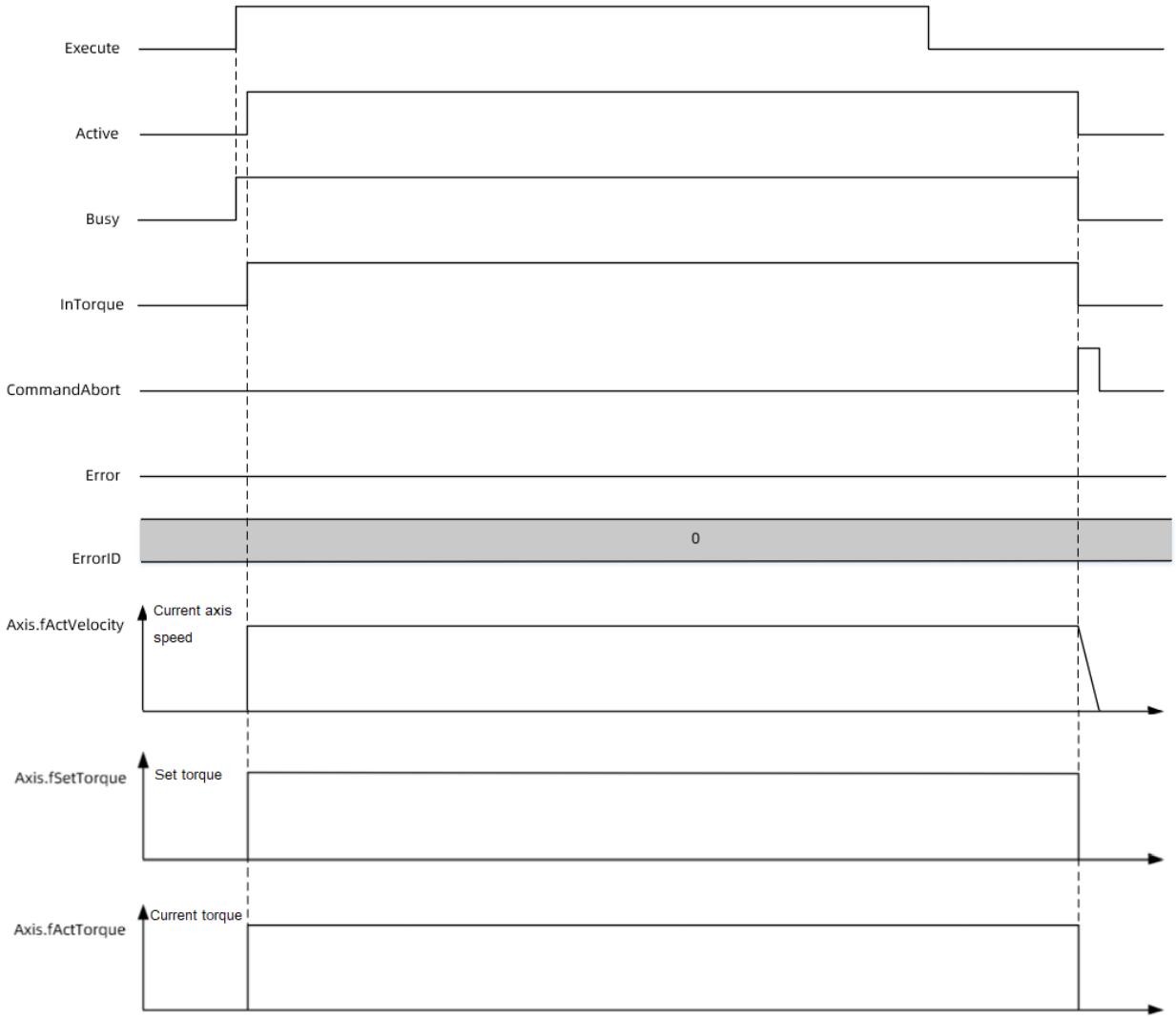
    Configure 607F (Maximum servo motor velocity).
- **Stop control in torque mode:**
  - In torque mode, the MC_Stop instruction can be executed to stop the drive. After the MC_Stop instruction is received, the drive switches to the synchronous position mode and decelerates according to the deceleration specified by the MC_Stop instruction.
  - During the execution of this instruction, the MC_MoveSuperImposed instruction cannot be used; once the MC_MoveSuperImposed instruction is triggered, CommandAborted is set for the MC_MoveSuperImposed instruction.

  Local pulse axis:
- This instruction cannot be used for local pulse axes.

- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Busy of the function block indicates that the current function block is being executed, and the parameter CommandAborted indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

  After this instruction is called, the output parameter Busy is set to TRUE on a rising edge of Execute. When the parameter fActVelocity reaches the limit, the actual torque will no longer ramp up, but the commanded torque will still ramp up normally without the InTorque being set.

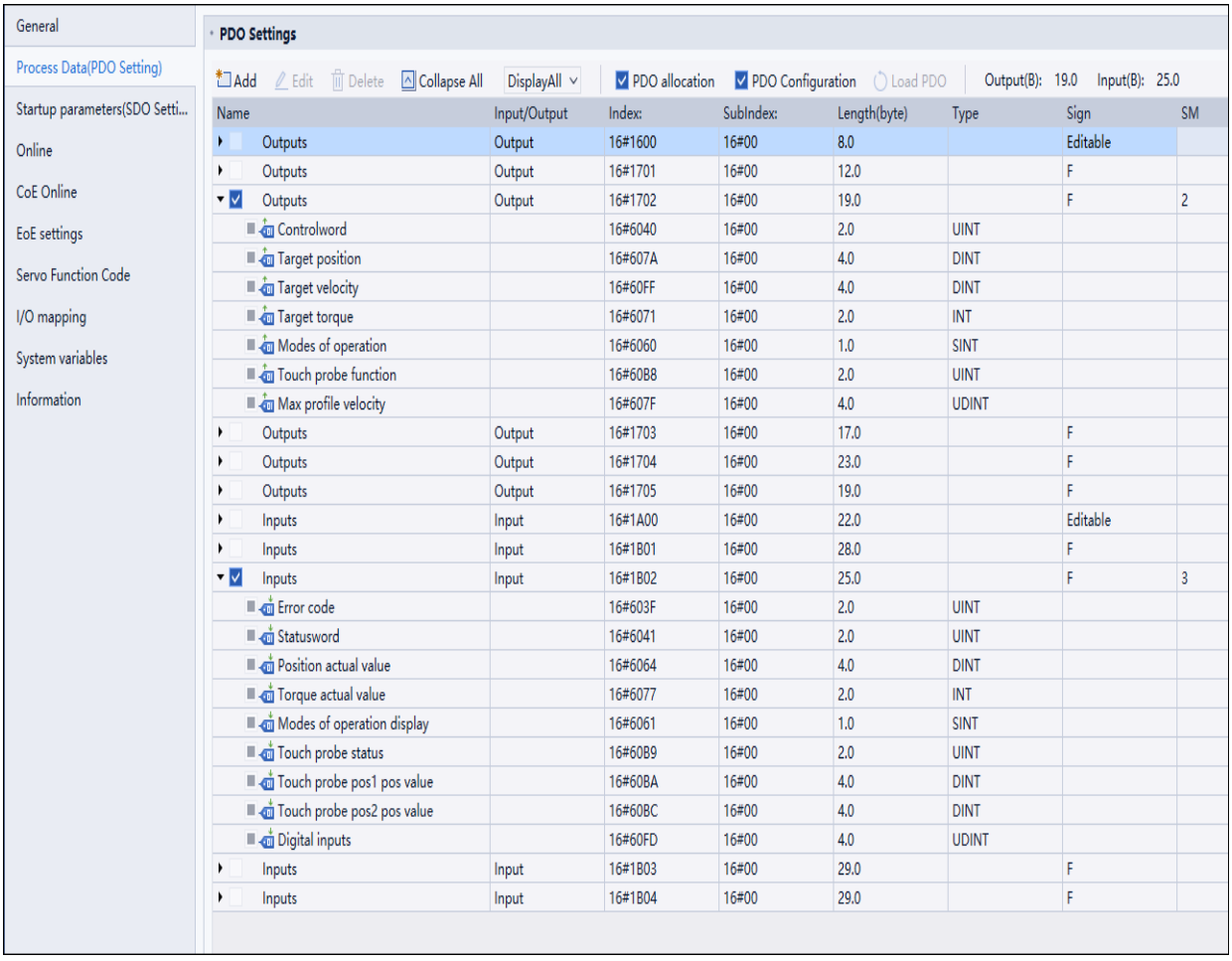


- **Faults**

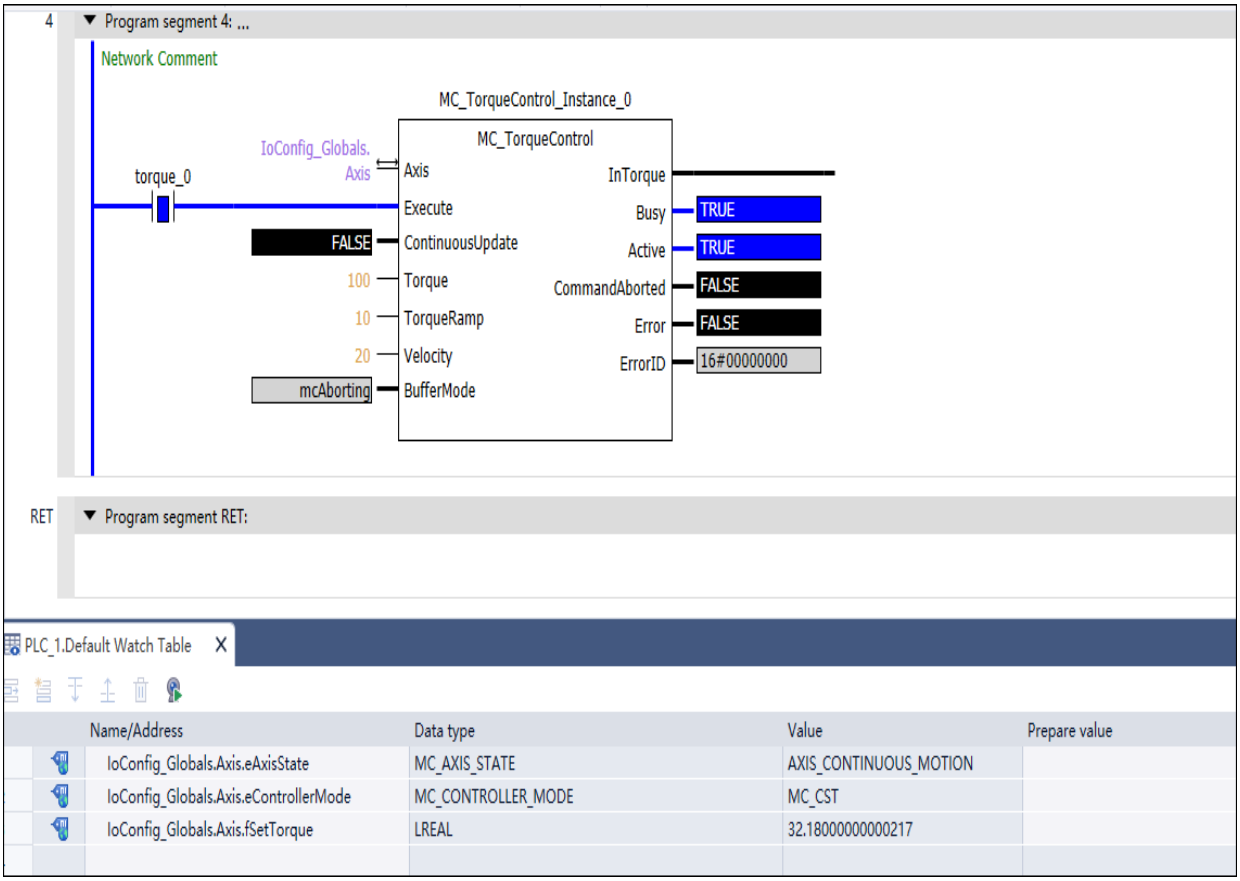
  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

- **Example**

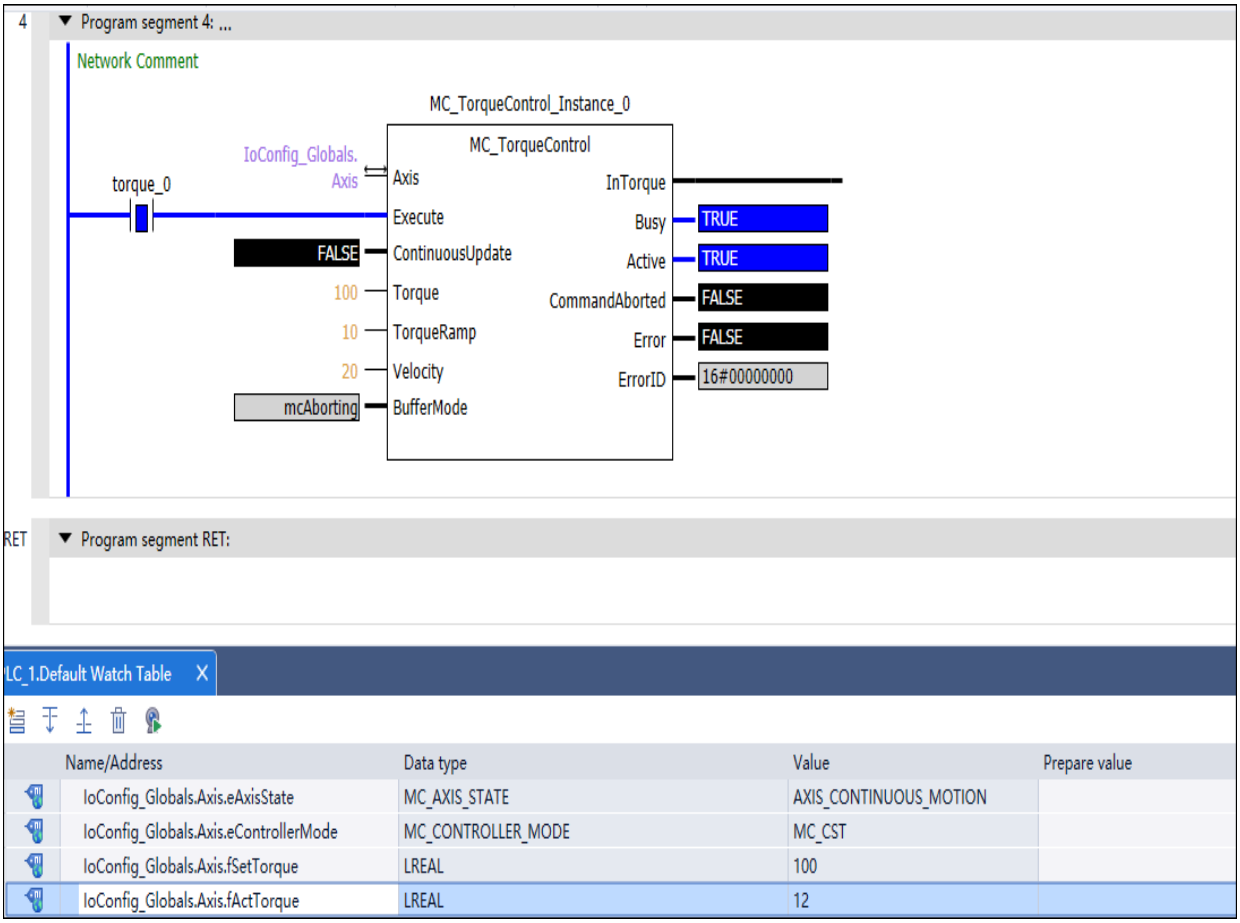
  To configure the PDO in the software, you can select the third group of inputs and outputs directly, or select the first group of inputs and outputs and add the 607A, 6060, 6077, and 6061.



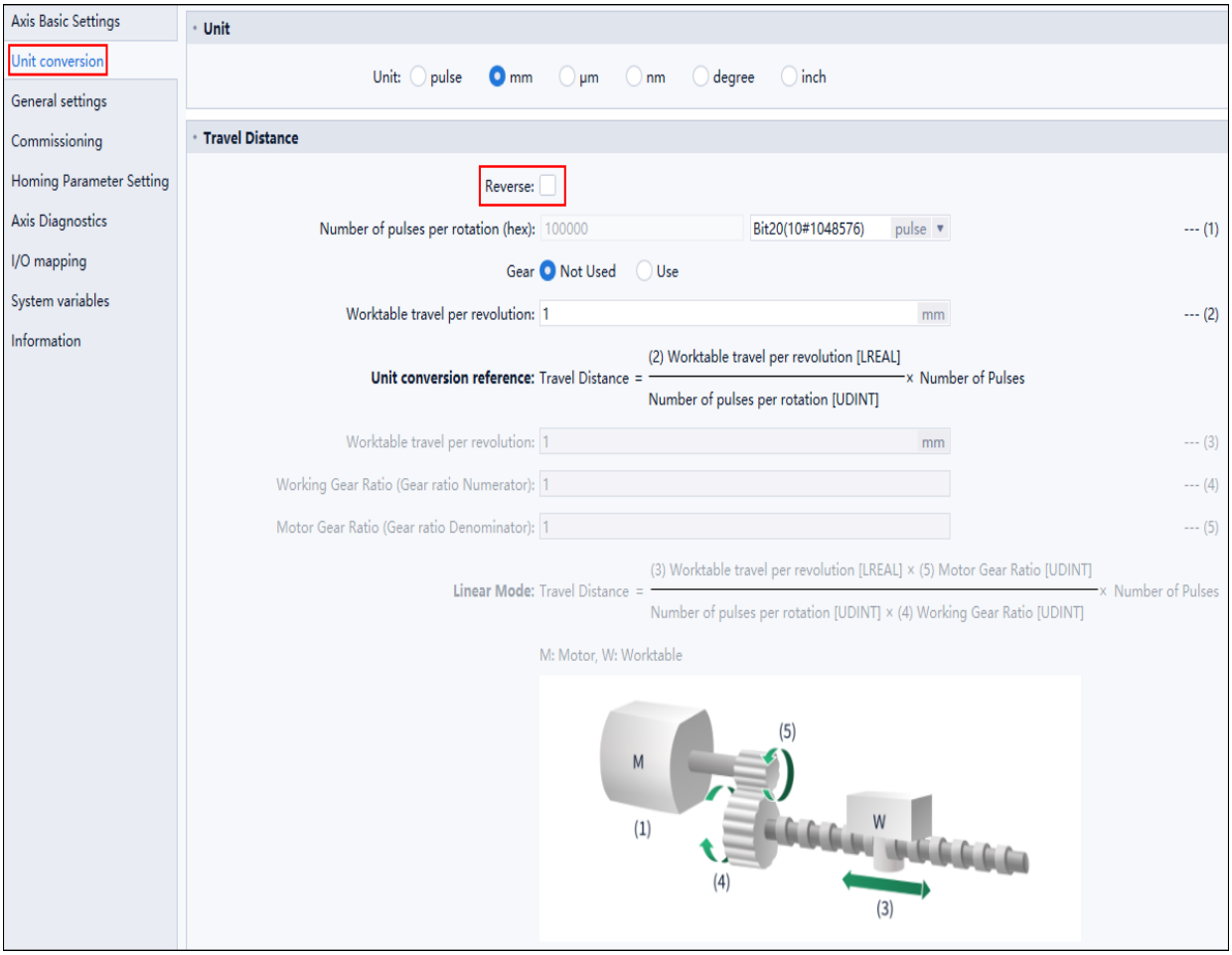
Configure the target torque. To prevent the torque from reaching the maximum value directly, you can configure the TorqueRamp. The torque will ramp up according to the specified TorqueRamp.



When the velocity parameter reaches the limit, the actual torque will no longer ramp up, but the commanded torque will still ramp up normally without the InTorque being set.



- **Precautions**
  - The parameter InTorque of this instruction is not checked in real time. If the torque ramp Tramp is not configured, the initial torque will reach the torque reference, at which point the InTorque is set. However, as the velocity ramps up and reaches the velocity limit, the corresponding torque limit will ramp down and no longer meet the actual torque range, at which point the InTorque is still not reset.
  - Checking Reverse on the interface has no effect on the MC_TorqueControl instruction.



18.3.3.3.40 MC_ImmediateStop

This instruction stops an axis using the specified mode.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_ImmediateStop	Immediate stop	MC_ImmediateStop_Instance_0 MC_ImmediateStop Axis Done Execute Busy StopMode Error ErrorID	MC_ImmediateStop_Instance_0(Axis :=, Execute :=, StopMode :=, Done =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.
StopMode	Stop mode	18.3.3.2.2.10 MC_STOP_MODE: Stop Mode Enumeration	0, 1	0	0: Perform an immediate stop. 1: Perform an immediate stop, and turn OFF the servo.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs in the function block.

- **Function**

  Bus-controlled axis:

  - This instruction can abort a single axis attached to the axis group, causing the axis group to perform an immediate stop.

- This instruction can be executed at any axis state. Even if the axis is in deceleration stop due to a fault, it can immediately stop with this instruction.
- After this instruction is triggered, all currently executing instructions are aborted, and the axis performs an immediate stop according to the specified value of the input parameter StopMode. When the StopMode is set to 1, the axis cannot be enabled after the immediate stop until the MC_Reset instruction is executed for reset.
- When this instruction is triggered with the axis state of POWER_OFF, the instruction is done directly without switching the axis state to a faulty state.
- The control object of this instruction is the control axis.

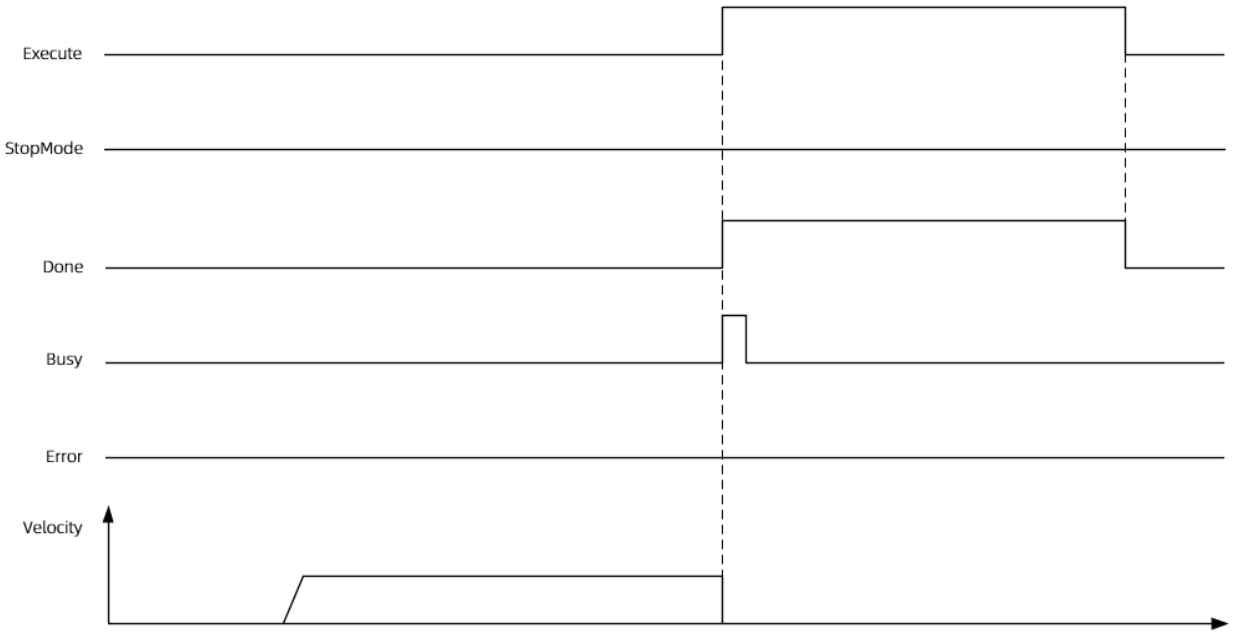
  Local pulse axis:

- This instruction can be used for local pulse axes.

- **Timing diagram**

The input parameter Execute must be triggered on a rising edge. When the output parameter Done is set to TRUE, it indicates that the instruction has been executed successfully. When the output parameter Busy is set to TRUE, it indicates the current function block is being executed.

After this instruction is called, the output parameter Busy is set to TRUE on a rising edge of Execute After the axis group performs an immediate stop, the output parameter Done is set to TRUE.



- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

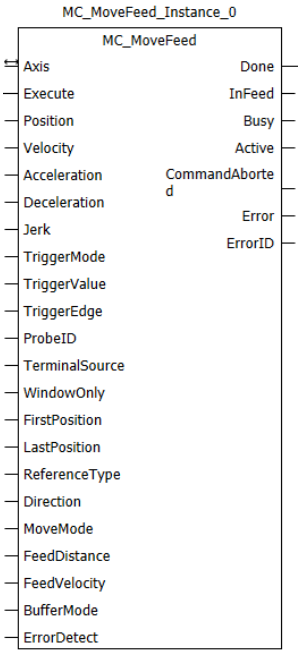
- **Precautions**

  It is not recommended to use this instruction when the axis state is AXIS_POWER_OFF.

18.3.3.3.41 MC_MoveFeed

This instruction performs positioning for the specified travel distance from the position where an external device triggers an interrupt input. Interrupt feeding is possible for absolute positioning, relative positioning, and velocity control.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_MoveFeed	Interrupt feeding		MC_MoveFeed_Instance_0(Axis :=, Execute :=, Position :=, Velocity :=, Acceleration :=, Deceleration :=, Jerk :=, TriggerMode :=, TriggerValue :=, TriggerEdge :=, ProbeID :=, TerminalSource :=, WindowOnly :=, FirstPosition :=, LastPosition :=, ReferenceType :=, Direction :=, MoveMode :=, FeedDistance :=, FeedVelocity :=, BufferMode :=, ErrorDetect :=, Done =>, InFeed =>, Busy =>, Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.
Position	Target position	LREAL	Positive number, negative number, or 0	0	Specify the absolute target position [u].
Velocity	Target velocity	LREAL	Positive number	0	Specify the target velocity [u/s].
Acceleration	Acceleration	LREAL	Positive number	0	Specify the acceleration [u/s^2].
Deceleration	Deceleration	LREAL	Positive number	0	Specify the deceleration [u/s^2].
Jerk	Jerk	LREAL	Positive number	0	Specify the slope of acceleration (deceleration) [u/s^3] (Valid for S-curve planning only).
TriggerMode	Trigger mode	INT	0, 1	0	0: Drive mode. 1: Controller mode.
TriggerValue	Trigger value	BOOL	TRUE, FALSE	FALSE	Trigger value.
TriggerEdge	Trigger edge	INT	0, 1	0	0: Rising edge. 1: Falling edge.
ProbeID	Touch probe identification	INT	0, 1	0	For ECAT axes, only touch probe 0/1 is supported. 0: Touch probe 1. 1: Touch probe 2. For local axes, it depends on the actual hardware channel.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
TerminalSource	Trigger source	INT	0, 1	0	0: DI terminal. 1: Z-phase signal.
WindowOnly	Window only	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled. FALSE: Disabled.
FirstPosition	First position	LREAL	-	0	Specify the position [u] to enable the touch probe.
LastPosition	Last position	LREAL	-	0	Specify the position [u] to disable the touch probe.
ReferenceType	Position type selection	18.3.3.2.2.11 MC_REFERENCE_TYPE: Data Type Enumeration	0: mcCommand 1: mcFeedback	0	0: Value calculated in the previous task cycle (fSavePosition). 1: Value obtained in the same task cycle (fActPosition). (Reserved parameter)
Direction	Motion direction	18.3.3.2.2.9 MC_DIRECTION: Motion Direction Enumeration	0: DIR_SHORTEST 1: DIR_POSITIVE 2: DIR_NEGATIVE 3: DIR_CURRENT 4: DIR_NODIRECTION	1	0: DIR_SHORTEST (Direction based on the shortest way). 1: DIR_POSITIVE (Positive direction). 2: DIR_NEGATIVE (Negative direction). 3: DIR_CURRENT (Current direction). 4: DIR_NODIRECTION (No direction).
MoveMode	Move mode	18.3.3.2.2.8 MC_MOVE_MODE: Move Mode Enumeration	0: AbsoluteMode 1: RelativeMode 2: VelocityMode	-	Selects the move mode.
FeedDistance	Feed distance	LREAL	Positive number, negative number, or 0	0	Travel distance after the interrupt feed input. Specify a positive value [u] to feed in the same direction as the axis is moving before the interrupt input and a negative value [u] to feed in the opposite direction. Note: For rotary axes, the target position for feeding is the remainder of the capture position + feed distance divided by the rotation cycle, ignoring multi-turn displacement.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
FeedVelocity	Feed velocity	LREAL	Positive number	0	Travel velocity [u/s] after the interrupt feed input.
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting 1: mcBuffered	0	0: mcAborting (Aborting). 1: mcBuffered (Buffered).
ErrorDetect	Error detection	BOOL	TRUE, FALSE	FALSE	Specify whether to enable fault detection when there is no interrupt feed input. TRUE: Enabled. FALSE: Disabled.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
InFeed	Feeding	BOOL	TRUE, FALSE	FALSE	TRUE when feeding after receiving a latch input.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Controlling	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity	Velocity	Any velocity that exceeds the motion parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	• When the MotionControl library version is 2.1.0.0 or earlier, the velocity equal to 0 will be automatically corrected to 1e-5. • When the MotionControl library version is 2.2.0.0 or later, the velocity equal to 0 will not be corrected.
Acceleration	Acceleration	Any acceleration that exceeds the motion parameter limit will be automatically corrected to the acceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration less than 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration equal to 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration equal to 0 will be automatically corrected to 10 times the velocity.

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the deceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration less than 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration equal to 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration equal to 0 will be automatically corrected to 10 times the velocity.
Jerk	Jerk	If the motion profile is set to S-curve, any jerk that exceeds the software limit will be automatically corrected to the jerk limit.	If the motion profile is set to S-curve, any jerk less than 0 will be automatically corrected to 10% of the jerk limit.	If the motion profile is set to S-curve, the jerk equal to 0 will be automatically corrected to 10% of the jerk limit.
FeedVelocity	Feed velocity	Any velocity that exceeds the motion parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	• When the MotionControl library version is 2.1.0.0 or earlier, the velocity equal to 0 will be automatically corrected to 1e-5. • When the MotionControl library version is 2.2.0.0 or later, the velocity equal to 0 will not be corrected.

- **Function**

  Bus-controlled axis:

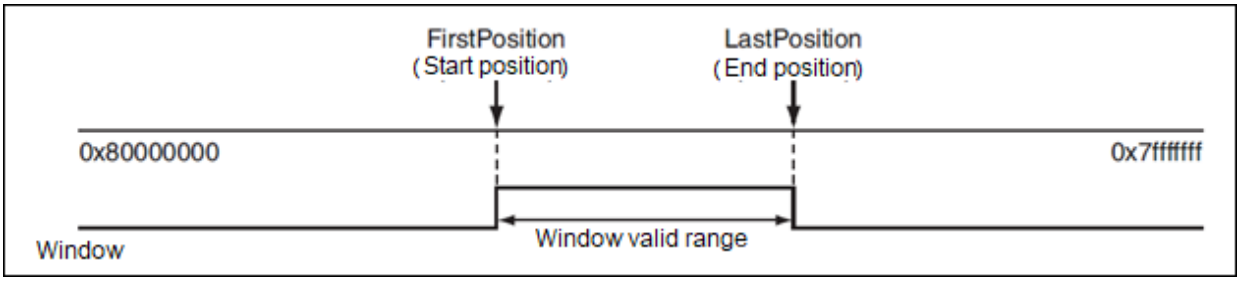
  - This instruction performs positioning in one of the travel methods (Absolute positioning, Relative positioning, and Velocity control) specified by the input parameter MoveMode on a rising edge of Execute, based on the input parameters Position, Velocity, Direction, Acceleration, and Deceleration.
  - When Absolute positioning is selected, specify the target position in the input parameter Position. When Relative positioning is selected, specify the target distance in the input parameter Position. Regardless of the travel method, the axis moves at the velocity specified by the input parameter Velocity.

- Relative positioning is performed at the FeedVelocity from the actual position where the external input is turned ON or OFF (according to the set value of TriggerEdge) during travel for the distance specified by the FeedDistance. The movement distance is specified by FeedDistance, starting from the actual position at FeedVelocity.
- When this instruction is aborted by the MC_ImmediateStop instruction, the MC_AbortTrigger instruction must be used to release the occupied channel (if any).
- If no interrupt signal is input before the axis reaches the target position during interrupt feeding in absolute or relative move mode, the axis will stop at the target position.
- You can specify whether to detect abnormal output through ErrorDetect. When there is no interrupt input and the axis stops moving, if the ErrorDetect is set to TRUE, the function block will report a fault after the motion stops, but the axis state will be Standstill rather than ErrorStop; if the ErrorDetect is set to FALSE, the output parameter Done will be set after the motion stops.
- To execute this instruction, set the WindowOnly to TRUE, and then specify FirstPosition and LastPosition. Interrupt feeding is performed for the interrupt signal generated by the actual position between the FirstPosition and the LastPosition.
- For bus-controlled axes, the PDOs 60B8/60B9 must be configured; otherwise, a fault will occur. When the instruction is executed on a rising edge with ProbeID = 0, a fault will occur if the 60BA object is not configured; when the instruction is executed on a falling edge, a fault will occur if the 60BB object is not configured. When the instruction is executed on a rising edge wit ProbeID = 1, a fault will occur if the 60BC object is not configured; when the instruction is executed on a falling edge, a fault will occur if the 60BD object is not configured.
- The motion parameters Acceleration, Deceleration, and Jerk are shared for the first segment of motion and fixed-length motion.
- When the input parameter WindowOnly is set to FALSE:

  The instruction does not identify the FirstPosition or LastPosition.

- When the input parameter WindowOnly is set to TRUE:
  - Linear mode

    The FirstPosition must be less than the LastPosition.

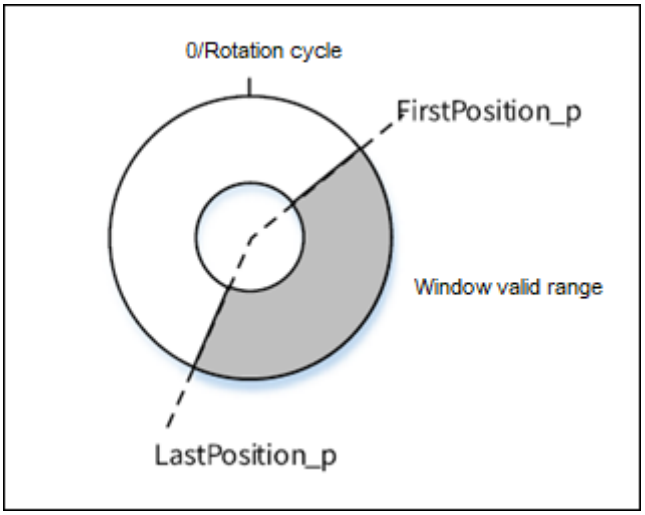


  - Rotary mode

    The FirstPosition can be less than or greater than the LastPosition.

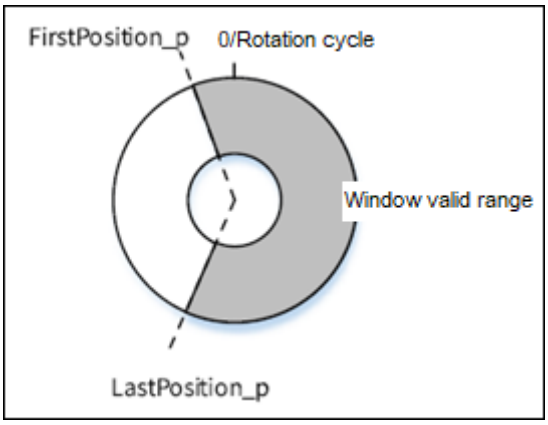
    1. FirstPosition < LastPosition

       The window is valid when the actual position is between the FirstPosition and the LastPosition, as shown in the following figure:



2. FirstPosition > LastPosition

The window is valid when the actual position is greater than the FirstPosition and less than the cycle length, or greater than 0 and less than the LastPosition, as shown in the following figure:



- In Rotary Mode, if the MoveMode is set to Absolute positioning, this instruction will check the value of Direction, which must be one of the five direction modes specified by the instruction; if the MoveMode is set to Velocity control, the instruction will check whether the direction is Positive, Negative, or Current; in other cases, the direction parameter will not be checked.
- This instruction does not occupy the probe channel when in the buffer. If the probe channel is occupied during execution, this instruction will abort the current instruction occupying the probe channel.
- In PLCopen specification, this instruction is valid with the axis state being Discrete_Motion. Other instructions that can place the axis in Discrete_Motion or comply with PLCopen state machine transitions can abort this instruction. When this instruction is aborted, the output parameter CommandAborted is set.
- During the execution of this instruction, if the axis decelerates to stop due to hitting the software limit, the output parameter Error will be set when the axis state being switched to Errorstop.
- This instruction automatically switches the control mode internally, so control mode switching between instructions does not rely on the MC_SetControllerMode instruction. You can directly call instructions. Each instruction internally checks if the current mode meets the requirements and automatically switches the mode if necessary.
- When this instruction is used for a virtual axis, it must be executed with a configured corresponding probe; otherwise, a fault will occur.

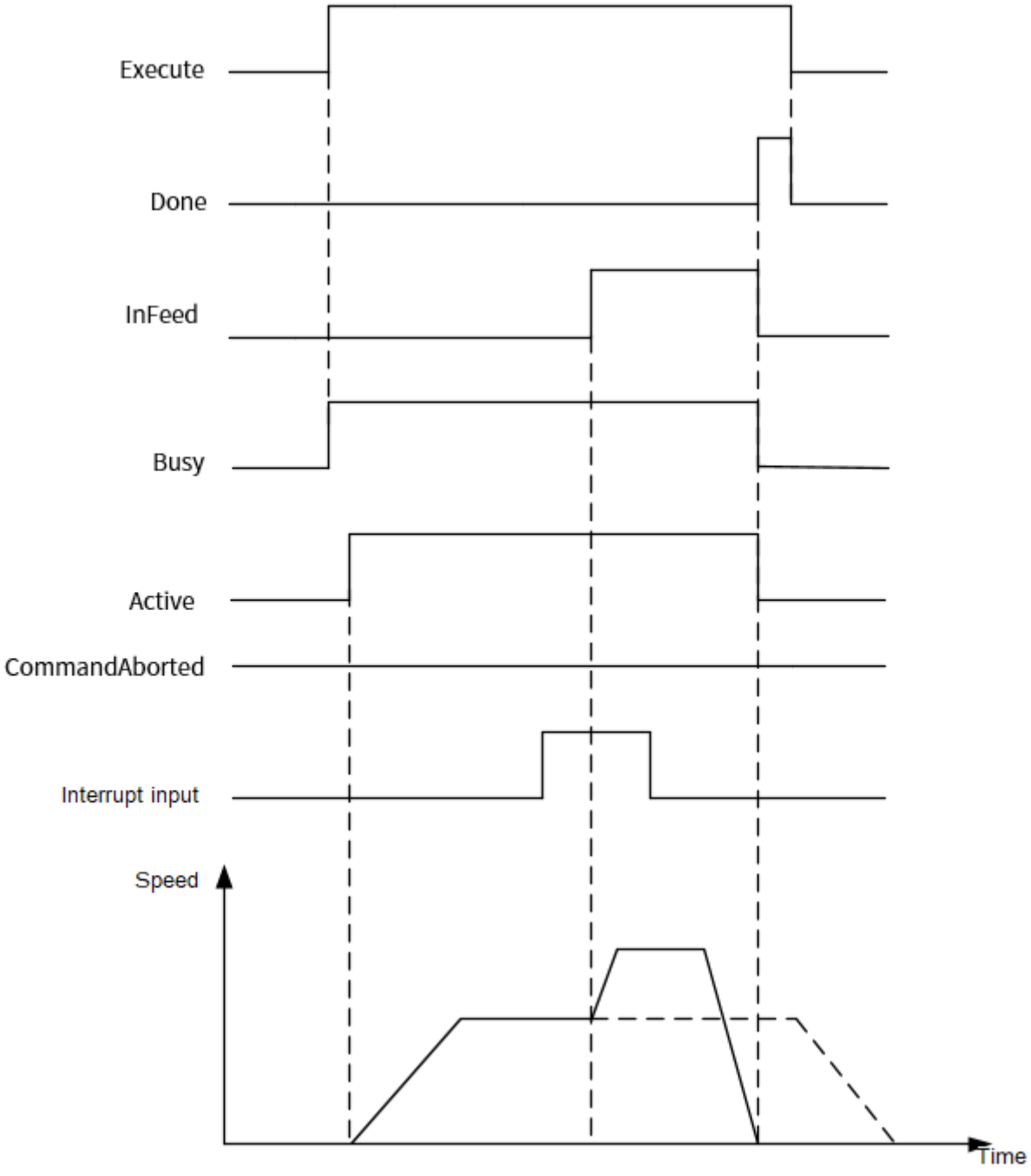
- During the execution of this instruction, the MC_MoveSuperImposed instruction cannot be used; once the MC_MoveSuperImposed instruction is executed, the input parameter CommandAborted is set for the MC_MoveSuperImposed instruction.
- When the axis is enabled after PLC cold reset and the MC_Home instruction is executed using No. 35 homing method, if this instruction is executed with DIR_CURRENT being set, the motion direction of this instruction will be affected.

Local pulse axis:

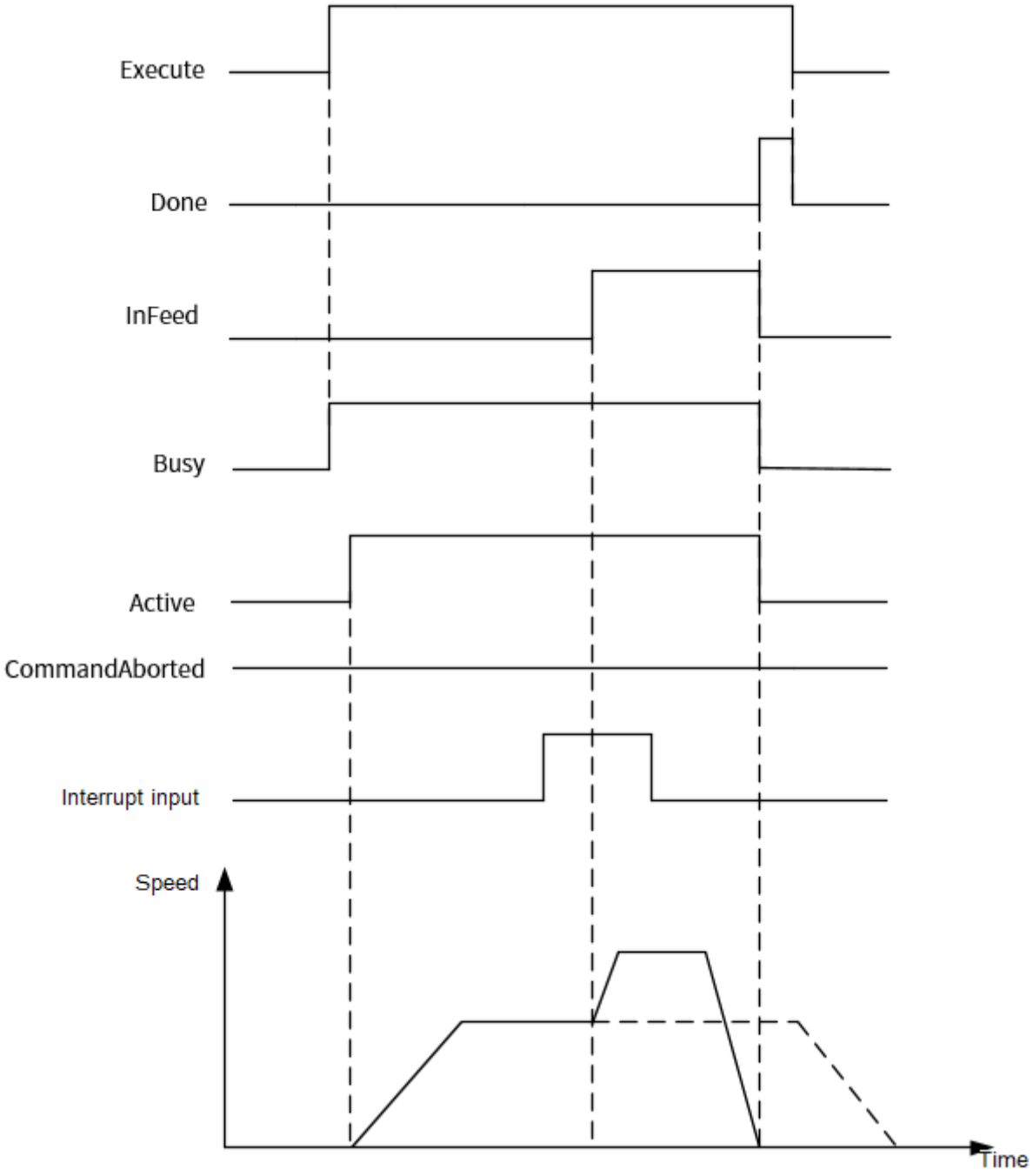
- This instruction can be used for local pulse axes.

- **Timing diagram**

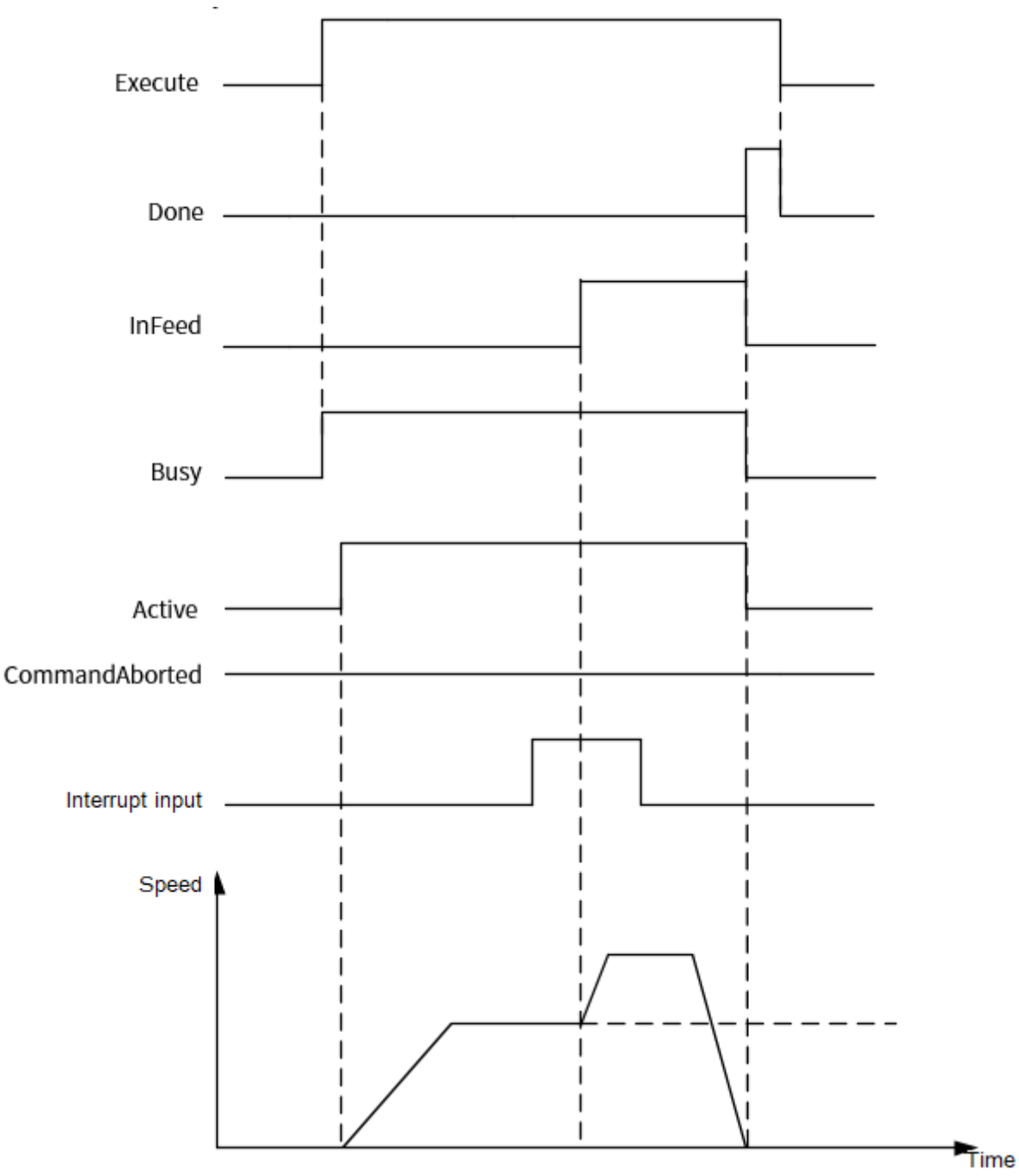
MoveMode = Absolute positioning



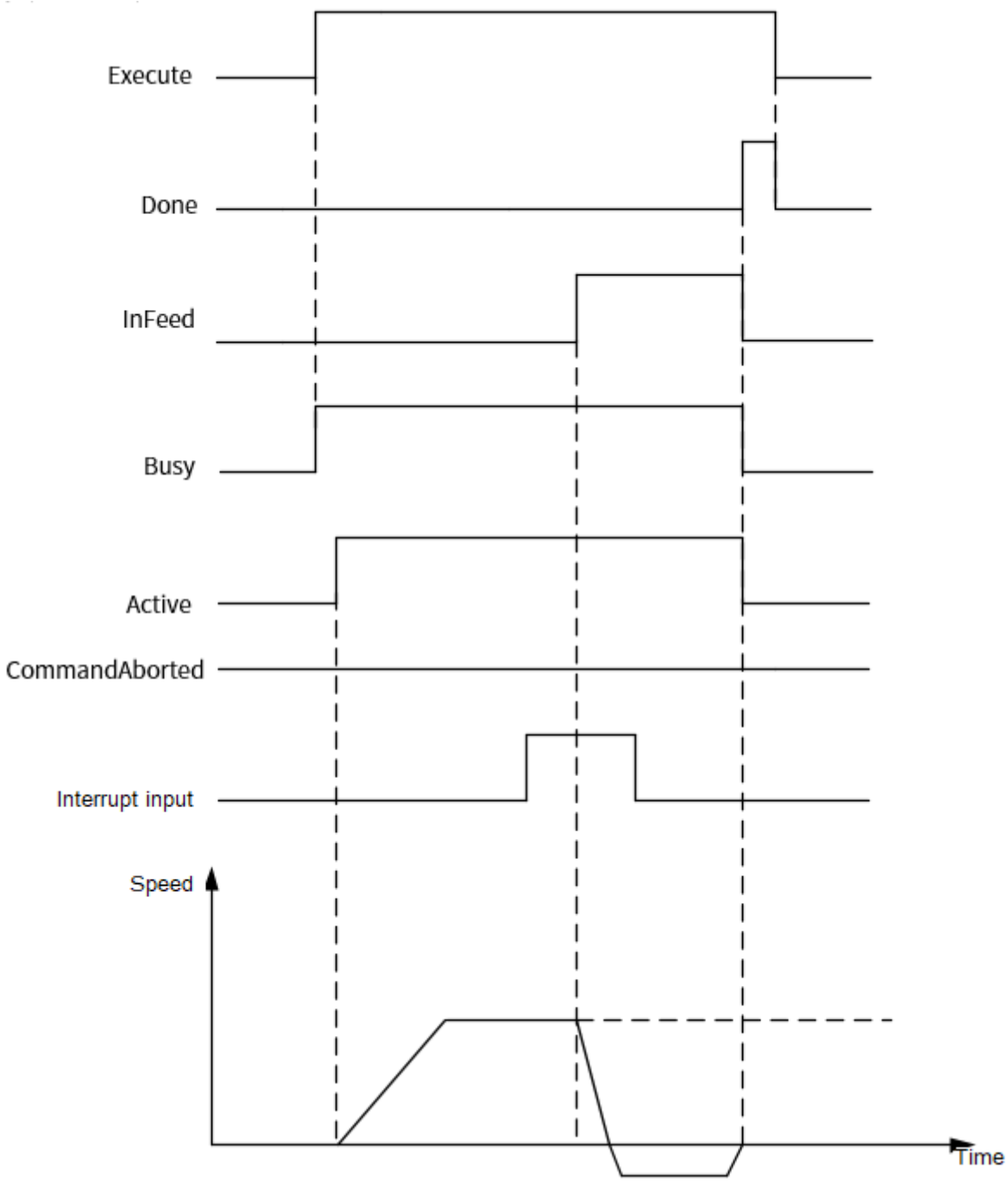
MoveMode = Relative positioning



MoveMode = Velocity control



Reversing after the interruption input



- **Re-execution of motion control instructions**

  This instruction cannot be re-executed.

- **Faults**

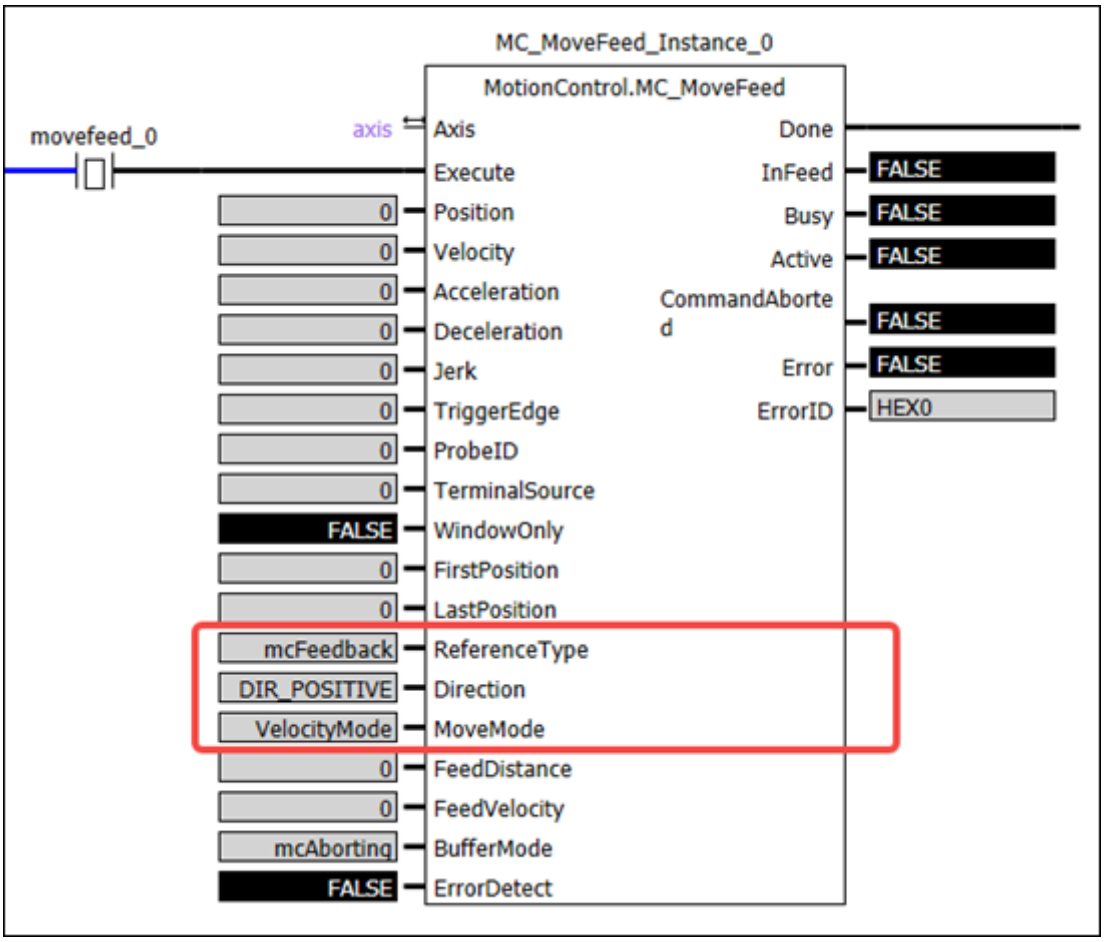
  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

- **Example**

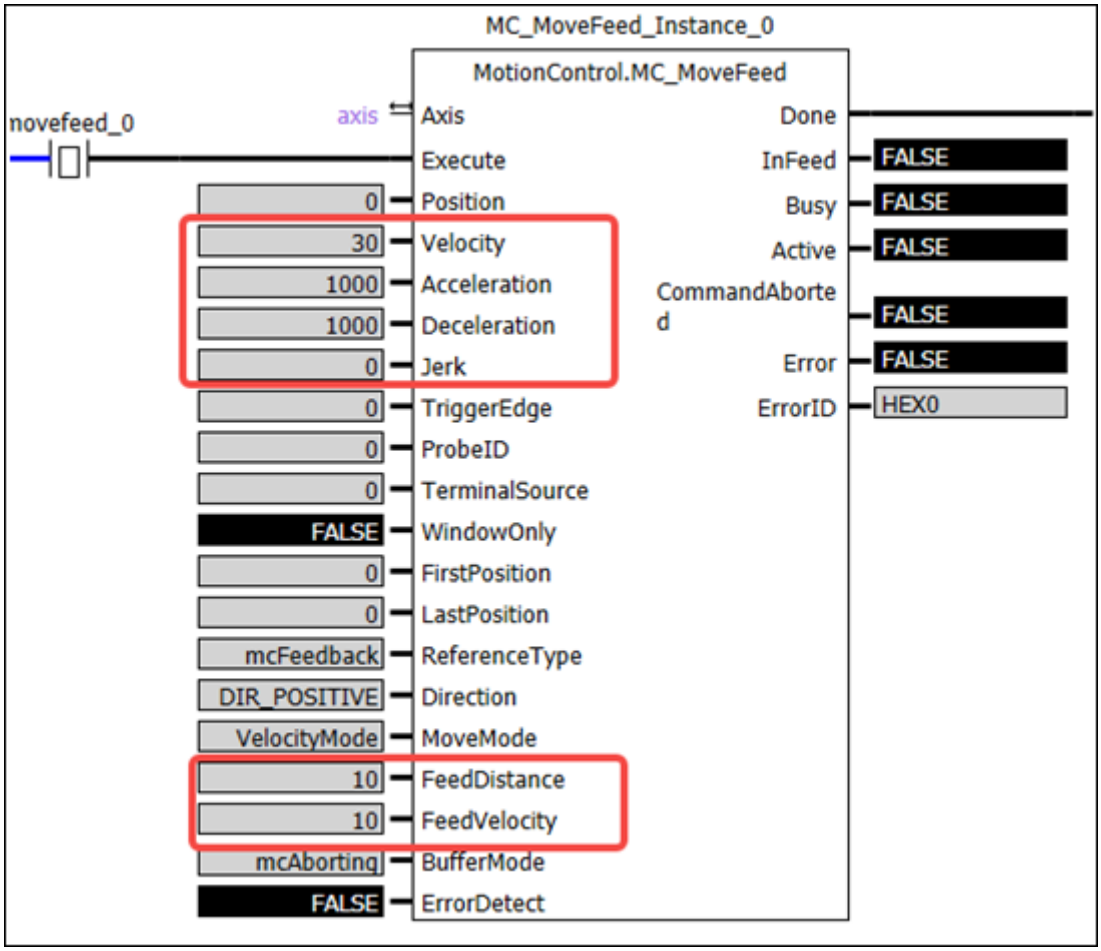
  Simulating a fixed-length marking scenario: Specify the main conveyor belt as Axis and the marking axis as Axis_1. When the main conveyor belt moves for a fixed length of 10 cm upon identifying the photo mark, the marking machine starts marking. After marking is complete, the interrupted fixed-length operation is restarted.

  First, configure the PDO. The rising edge of touch probe 1 is guaranteed by default PDO configuration. See the MC_TouchProbe instruction for other configurations.

Select the running program for the main conveyor belt, set MoveMode = VelocityMode, and specify the Direction as Positive direction or Negative direction.

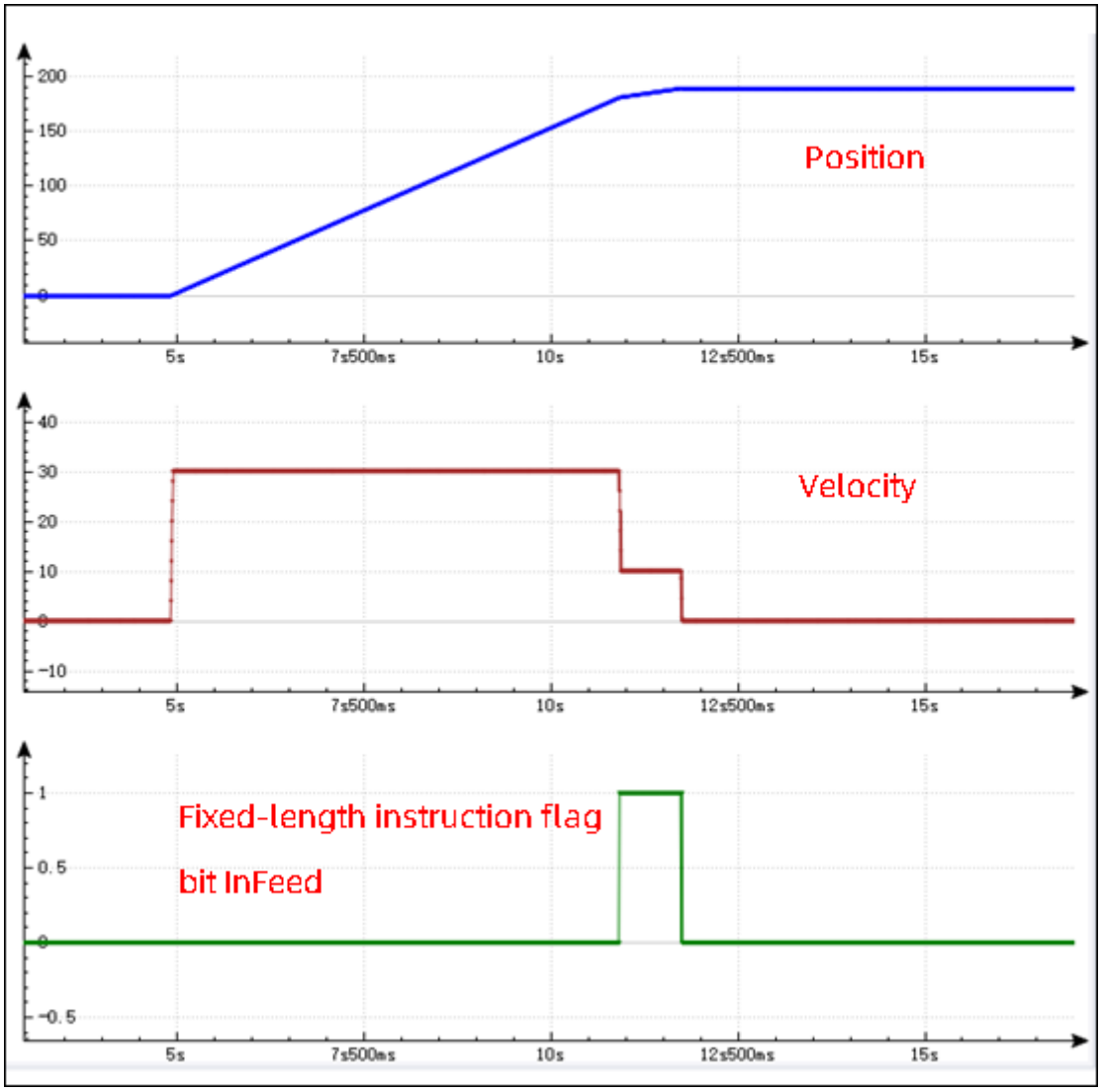


Specify the motion parameters Velocity, Acceleration, and Deceleration to prevent the motor reversal due to fixed-length deceleration. It is recommended to change the acceleration/deceleration to the limit values in the software. Specify the motion parameters FeedDistance and FeedVelcity for fixed-length movement.

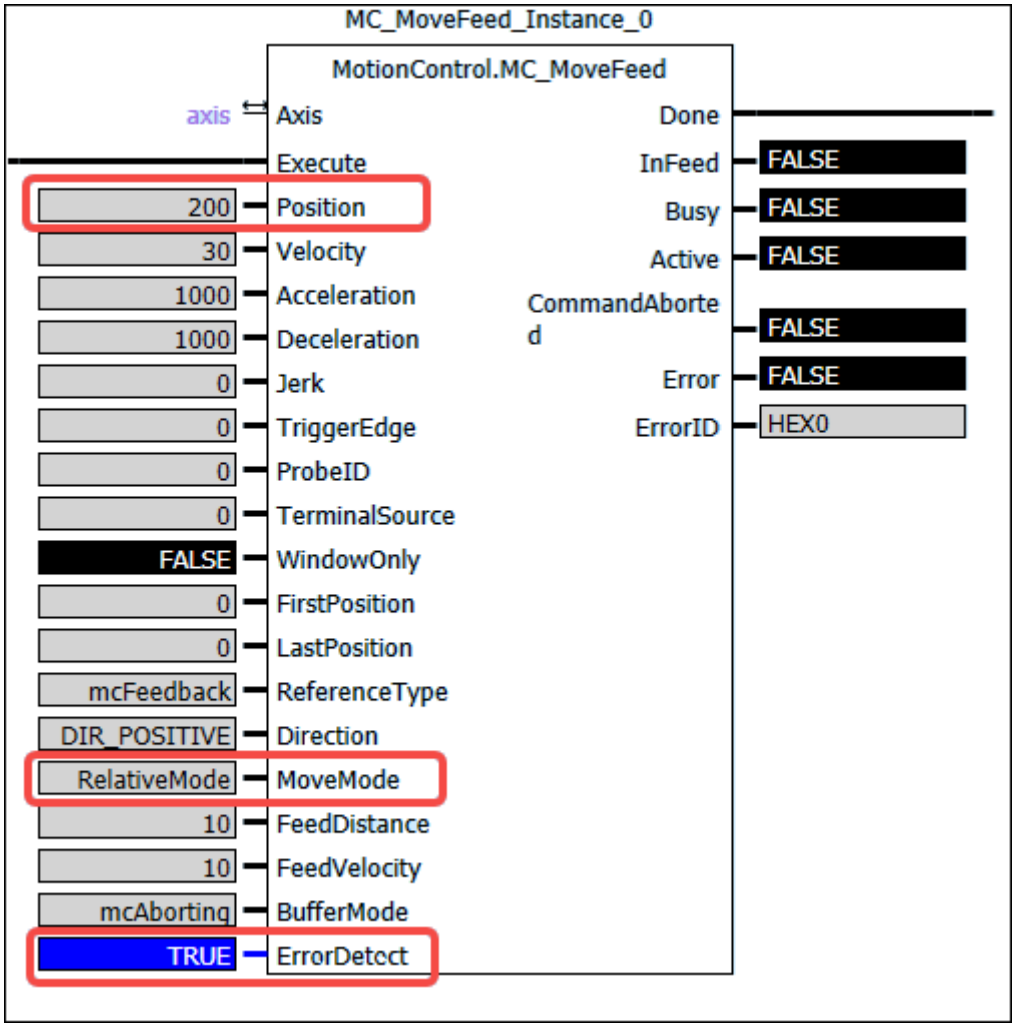


By default, the value 0 corresponds to the rising edge trigger of touch probe 1. You can specify the probe source. See the MC_TouchProbe instruction for specific configurations.

The instruction controls the axis to move in a straight line at the velocity set by Velocity. After the photo mark is identified, the InFeed will be set. Once the axis moves for a fixed length of 10 cm at the set value of FeedVelocity, the main motion will stop and the marking machine will start working. After the marking is complete, re-executing the MoveFeed instruction will start the next cycle.



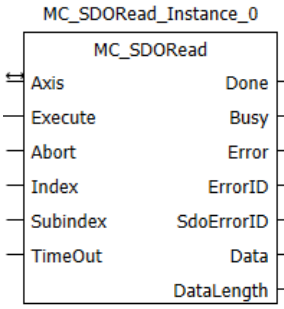
- **Precautions**
  - When the fixed-length position is close to the photo mark position, the marking machine can use the InFeed of this instruction as a flag to start printing during the fixed-length movement, improving the efficiency.
  - You can modify the MoveMode to RelativeMode and specify the Position as a distance covering multiple item spacing. You can specify the unit of the preceding distance on the FA interface. When the ErrorDetect is set, and the program will stop with a fault message if no photo mark is detected after a travel distance greater than 200.



18.3.3.3.42 MC_SDORead

This instruction reads SDO data. Reading SDO is independent of the axis state. The instruction can be triggered regardless of the axis state.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_SDORead	Read specified SDO data		MC_SDORead_Instance_0(Axis :=, Execute :=, Abort :=, Index :=, Subindex :=, TimeOut :=, Done =>, Busy =>, Error =>, ErrorID =>, SdoErrorID =>, Data =>, DataLength =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.
Abort	Abort	BOOL	TRUE, FALSE	FALSE	The function block is aborted on a rising edge of this input variable.
Index	Object dictionary index	WORD	Positive number or 0	0	Object dictionary index of SDO to be read.
Subindex	Object dictionary subindex	BYTE	Positive number or 0	0	Object dictionary subindex of SDO to be read.
TimeOut	Timeout period	UDINT	Positive number or 0	0	SDO read timeout period (ms).

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is being executed.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs in the function block.
SdoErrorID	SDO fault code	UDINT	Positive number or 0	0	If a fault occurs in the device, more information about the fault can be obtained with this output variable. For details, see 18.4.9.17 SDO Fault Codes.
Data	Read data	DWORD	-	0	SDO data read. It is effective only when the parameter Done is set to TRUE.
DataLength	Read data length	BYTE	1 to 4	0	SDO data length read. It is effective only when the parameter Done is set to TRUE.

- **Function**

  Bus-controlled axis:

  - This instruction is triggered on a rising edge of Execute. The input parameters other than Abort are latched on the rising edge, among which any modified parameters will not be effective during the execution of the instruction.
  - You can specify SDO read timeout period (ms) through the parameter TimeOut. The value 0 indicates 1000 task cycles by default. If the read operation times out, the function block will report a timeout fault.
  - The MC_SDORead instruction is independent of the axis state. It can be triggered regardless of the axis state. If the instruction is triggered before successful communication with the axis, it will be executed after the communication is established. If no communication is established within the specified TimeOut value, the function block will report a timeout fault.
  - If the parameter Abort is set to TRUE, the function block will be aborted with the output parameters being reset.
  - The fault codes of this function block include both function block-specific fault codes and SDO fault codes. When a function block-specific fault occurs, the function block reports the fault via ErrorID only. When an SDO read/write fault occurs, the function block reports the fault via ErrorID and SdoErrorID.

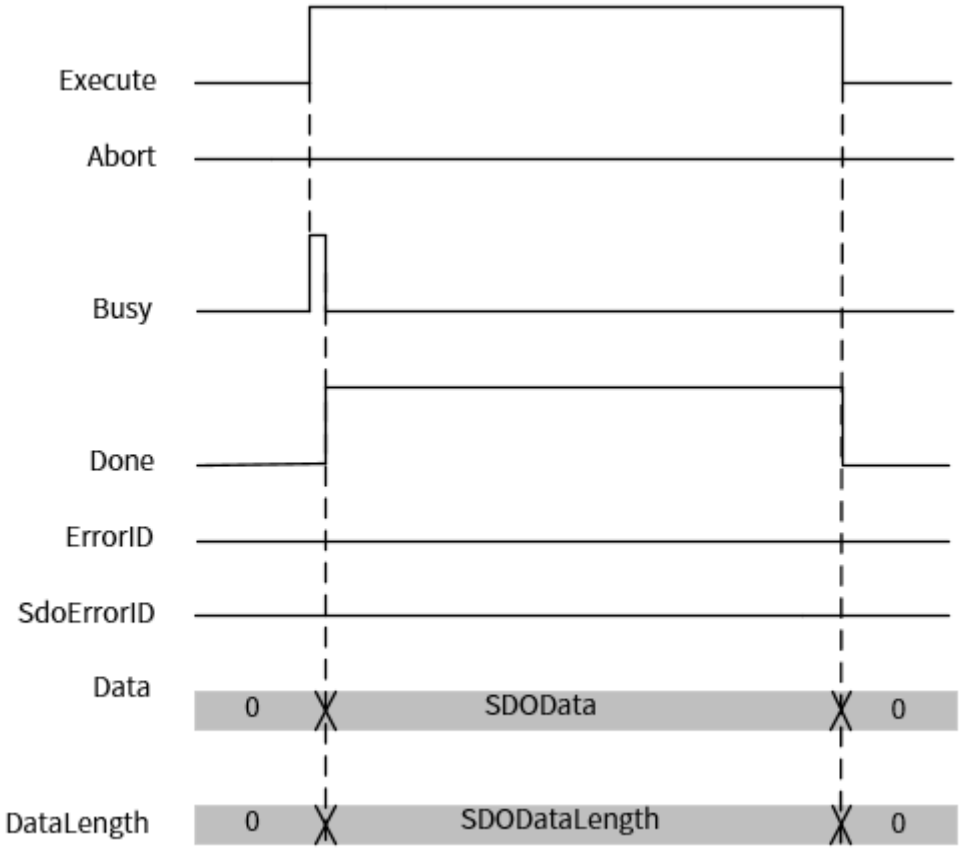
Local pulse axis

◦ This instruction cannot be used for local pulse axes.

- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Busy of the function block indicates that the current function block is being executed. When the parameter Abort is set to TRUE, the function block is aborted and reset.

  After this instruction is called, the output parameter Busy will be set to TRUE if the Execute signal is TRUE; the output parameter Done will be set to TRUE upon successful reading, with the Busy showing the value FALSE.

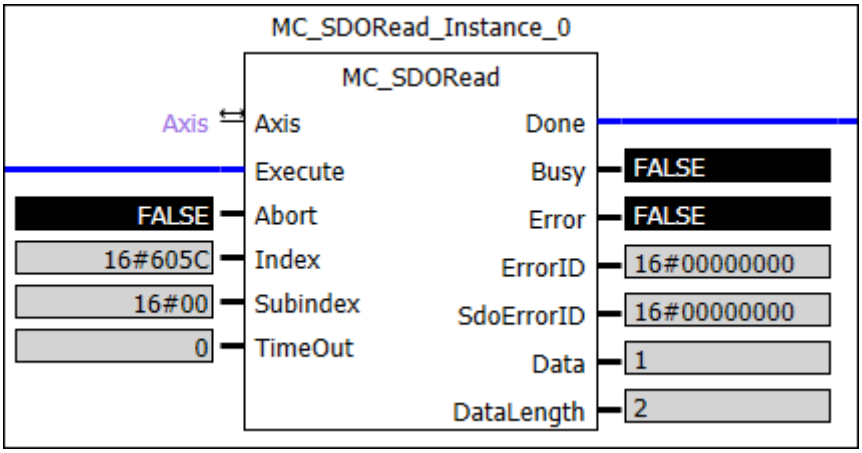


- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

- **Example**

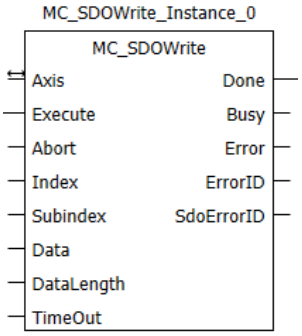
  Using the bus-controlled axis as an example, input the specified axis, index 0x605C, and subindex 0. The execution result is as follows:



18.3.3.3.43 MC_SDOWrite

This instruction writes SDO data. Writing SDO is independent of the axis state. The instruction can be triggered regardless of the axis state.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_SDOWrite	Write specified SDO data to input axis		MC_SDOWrite_Instance_0(Axis :=, Execute :=, Abort :=, Index :=, Subindex :=, Data :=, DataLength :=, TimeOut :=, Done =>, Busy =>, Error =>, ErrorID =>, SdoErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.
Abort	Abort	BOOL	TRUE, FALSE	FALSE	The function block is aborted on a rising edge of this input variable.
Index	Object dictionary index	WORD	Positive number or 0	0	Object dictionary index of SDO to be written.
Subindex	Object dictionary subindex	BYTE	Positive number or 0	0	Object dictionary subindex of SDO to be written.
Data	Data to be written	DWORD	-	0	SDO data to be written.
DataLength	Written data length	BYTE	1 to 4	0	Length of the SDO data to be written.
TimeOut	Timeout period	UDINT	Positive number or 0	0	SDO write timeout period (ms).

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is being executed.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs in the function block.
SdoErrorID	SDO fault code	UDINT	Positive number or 0	0	If a fault occurs in the device, more information about the fault can be obtained with this output variable. For details, see 18.4.9.17 SDO Fault Codes.

- **Function**

  Bus-controlled axis:

  - This instruction is triggered on a rising edge of Execute. The input parameters other than Abort are latched on the rising edge, among which any modified parameters will not be effective during the execution of the instruction.

- You can specify SDO write timeout period (ms) through the parameter TimeOut. The value 0 indicates 1000 task cycles by default. If the write operation times out, the function block will report a timeout fault.
- When SDO parameters are written, the length of the written parameters must be between 1 and 4; otherwise, the function block reports a fault.
- The MC_SDOWrite instruction is independent of the axis state. It can be triggered regardless of the axis state. If the instruction is triggered before successful communication with the axis, it will be executed after the communication is established. If no communication is established within the specified TimeOut value, the function block will report a timeout fault.
- If the parameter Abort is set to TRUE, the function block will be aborted with the output parameters being reset.
- The fault codes of this function block include both function block-specific fault codes and SDO fault codes. When a function block-specific fault occurs, the function block reports the fault via ErrorID only. When an SDO read/write fault occurs, the function block reports the fault via ErrorID and SdoErrorID.

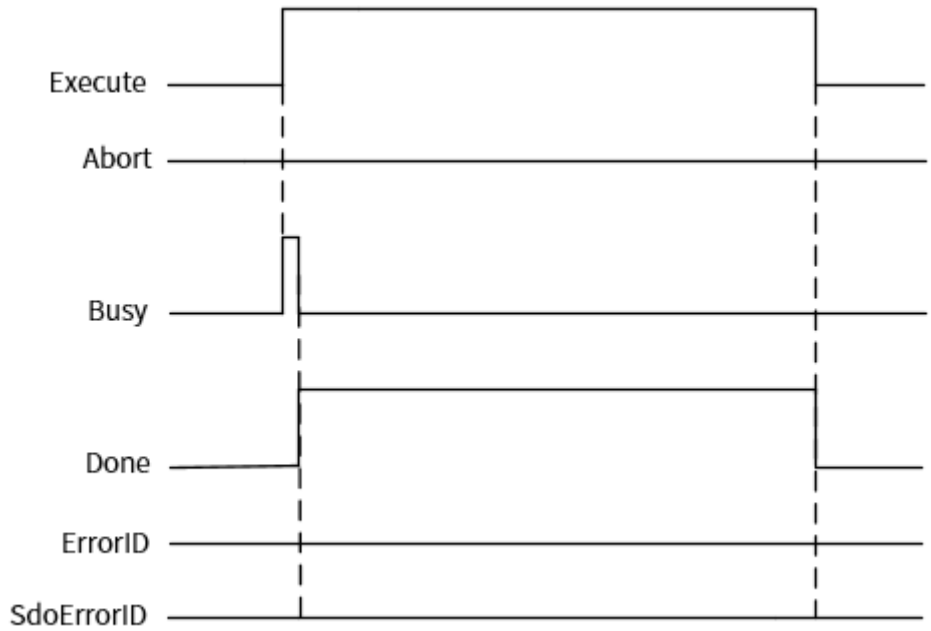
Local pulse axis

- This instruction cannot be used for local pulse axes.

- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Busy of the function block indicates that the current function block is being executed. When the parameter Abort is set to TRUE, the function block is aborted and reset.

  After this instruction is called, the output parameter Busy will be set to TRUE if the Execute signal is TRUE; the output parameter Done will be set to TRUE upon successful writing, with the Busy showing the value FALSE.

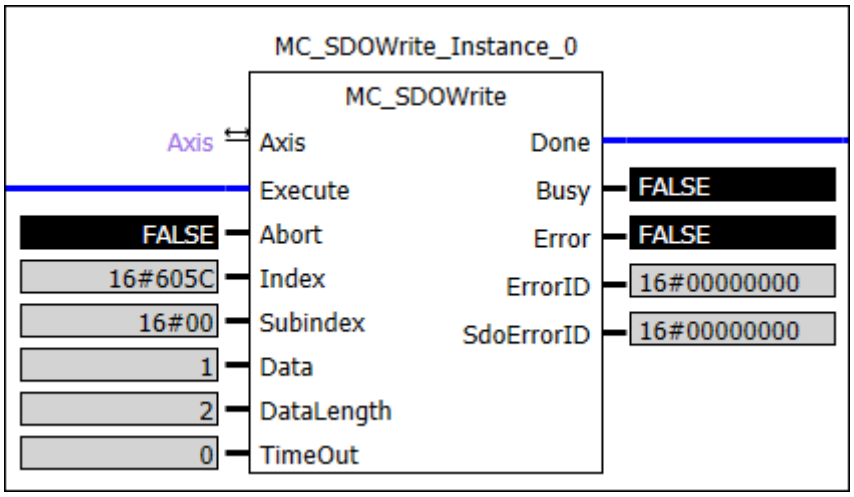


- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

- **Example**

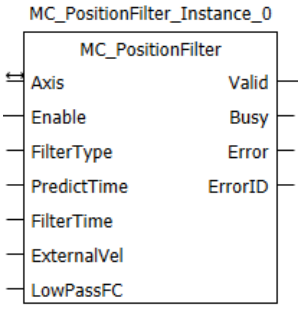
Using the bus-controlled axis as an example, input the specified axis, index 0x605C, subindex 0, Data = 1, and DataLength = 2. The execution result is as follows:



18.3.3.3.44 MC_PositionFilter

This instruction filters the position and velocity information based on the specified filter type and outputs the filtered velocity and position. It does not affect the position reference, velocity reference, actual position, or actual velocity, so the actual output of the corresponding axis will not be affected. The filtered position and velocity will be used as the signal source for the master axis in synchronized motion.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_PositionFilter	Position filter		MC_PositionFilter_Instance_0(Axis :=, Enable :=, FilterType :=, PredictTime :=, FilterTime :=, ExternalVel :=, LowPassFC :=, Valid =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction.
FilterType	Filter type	INT	0 to 11	0	Position source selection. For details, see Filter types.
PredictTime	Predicted cycles	INT	Compliant with data type	3	Predicted number of cycles. Positive and negative values are supported. A positive value indicates lead; a negative value indicates delay. This parameter is effective when the FilterType is not set to 0 or 1.
FilterTime	Filtering cycle/ window	INT	2 to 100	0	Cycle/Window for filtering. Value range: 2 to 100. This parameter is effective when the FilterType is not set to 0 or 1. It has different meanings with different values of the FilterType.
External Vel	External velocity	LREAL	Compliant with data type	0.0	External velocity reference, effective when the FilterType is set to 4 or 5.
LowPass Fc	Cut-off frequency	LREAL	Positive number	0.0	Higher cut-off frequency results in less filter delay and decreased high-frequency noise suppression capability. Conversely, lower cut-off frequency results in smoother signal and increased phase lag.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Input shaping	BOOL	TRUE, FALSE	FALSE	TRUE: Input shaping is being executed.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE: The function block is being executed.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE: A fault occurs during input shaping.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs in the function block.

- **Function**
  - This instruction does not affect the position reference, velocity reference, actual position, or actual velocity, so the actual output of the corresponding axis will not be affected. The filtered position and velocity will be used as the signal source for the master axis in synchronized motion.
  - This instruction is level-triggered. Continuous update of the input parameter ExternalVel is supported only when the FilterType is set 4 or 5; any other parameters modified during the execution of the instruction will not be effective.
  - FilterTime is an input parameter for this instruction. When the value of FilterType is between 2 and 5, it indicates the filtering cycle. When the value of FilterType is 8 or 9, it indicates the filtering cycle for moving filter. When the value of FilterType is 10 or 11, it indicates the window for median filter.
  - Filter types

    The MC_PositionFilter instruction determines the filter type based on the input parameter FilterType, performs filtering according to the related filter parameters, and outputs the filtered position of the fRefPosition and filtered velocity of the fRefVelocity. The value range of FilterType is 0 to 11, corresponding to the following filter types:

Reference Source	Condition	Output
Master axis position reference (Direct mode)	FilterType = 0	$fPosition = fSetPosition$ $fVelocity = fSetVelocity$
Actual master axis position (Direct mode)	FilterType = 1	$fPosition = fActPosition$ $fVelocity = fActVelocity$
Actual master axis position + Actual master axis velocity	FilterType = 2	$fVelocity = \frac{\sum_{1}^{FilterTime} fActVelocity}{FilterTime}$ $fAcceleration = \frac{fVelocity_{FilterTime} - fVelocity_{FilterTime-1}}{\Delta t}$ $fPosition = fActPosition + fVelocity * \Delta t * \Pr edictTime + (\Delta t * \Pr edictTime)^2 * \frac{1}{2} * fAcceleration$

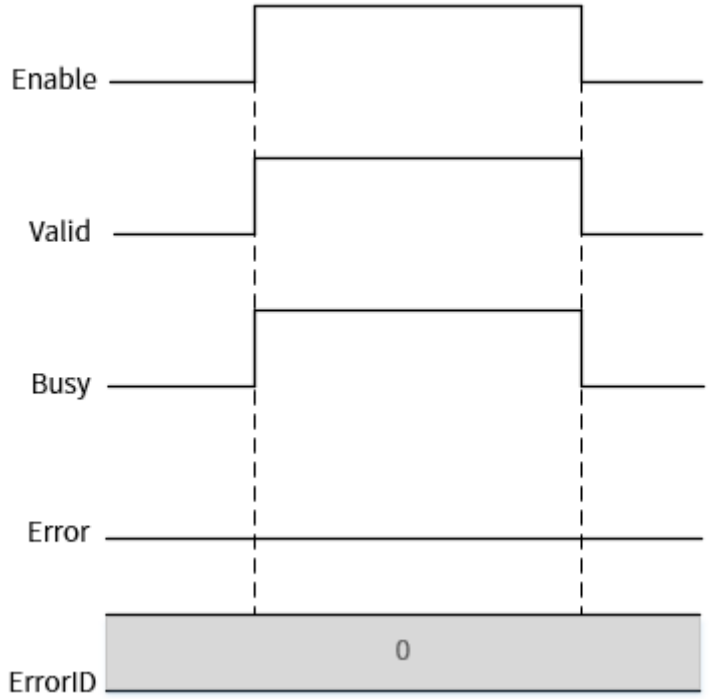
Reference Source	Condition	Output
Master axis position reference + Master axis velocity reference	FilterType = 3	$fVelocity = \frac{\sum_{1}^{FilterTime} fSetVelocity}{FilterTime}$ $fAcceleration = \frac{fVelocity_{FilterTime} - fVelocity_{FilterTime-1}}{\Delta t}$ $fPosition = fSetPosition + fVelocity * \Delta t * \Pr edictTime +$ $(\Delta t * \Pr edictTime)^2 * \frac{1}{2} * fAcceleration$
Actual master axis position + External velocity	FilterType = 4	$fVelocity = \frac{\sum_{1}^{FilterTime} fExternalVel}{FilterTime}$ $fAcceleration = \frac{fVelocity_{FilterTime} - fVelocity_{FilterTime-1}}{\Delta t}$ $fPosition = fActPosition + fVelocity * \Delta t * \Pr edictTime +$ $(\Delta t * \Pr edictTime)^2 * \frac{1}{2} * fAcceleration$
Master axis position reference + External velocity	FilterType = 5	$fVelocity = \frac{\sum_{1}^{FilterTime} fExternalVel}{FilterTime}$ $fAcceleration = \frac{fVelocity_{FilterTime} - fVelocity_{FilterTime-1}}{\Delta t}$ $fPosition = fSetPosition + fVelocity * \Delta t * \Pr edictTime +$ $(\Delta t * \Pr edictTime)^2 * \frac{1}{2} * fAcceleration$
Low-pass filter: position and velocity reference	FilterType = 6	$fVelocity_n = (1-\alpha) fVelocity_{n-1} + \alpha fSetVelocity$ $fPosition_n = (1-\alpha) fPosition_{n-1} + \alpha fSetPosition$ $a = \frac{\Delta t}{\Delta t + \frac{1}{2 * \pi * LowPassFc}}$
Low-pass filter: actual position and velocity	FilterType = 7	$fVelocity_n = (1-\alpha) fVelocity_{n-1} + \alpha fActVelocity$ $fPosition_n = (1-\alpha) fPosition_{n-1} + \alpha fActPosition$ $a = \frac{\Delta t}{\Delta t + \frac{1}{2 * \pi * LowPassFc}}$

Reference Source	Condition	Output
Moving average filter: position and velocity reference	FilterType = 8	$fVelocity = \frac{\sum_{1}^{FilterTime} fSetVelocity}{FilterTime}$ $fPosition = \frac{\sum_{1}^{FilterTime} fSetPosition}{FilterTime}$
Moving average filter: actual position and velocity	FilterType = 9	$fVelocity = \frac{\sum_{1}^{FilterTime} fActVelocity}{FilterTime}$ $fPosition = \frac{\sum_{1}^{FilterTime} fActPosition}{FilterTime}$
Median filter: position and velocity reference	FilterType = 10	The filter window is determined through FilterTime. The fSetPosition and fSetVelocity are placed into the window periodically and continuously refreshed (FIFO). The values within the window are sorted in an ascending order, with the median value being output.
Median filter: actual position and velocity	FilterType = 11	The filter window is determined through FilterTime. The fActPosition and fActVelocity are placed into the window periodically and continuously refreshed (FIFO). The values within the window are sorted in an ascending order, with the median value being output.

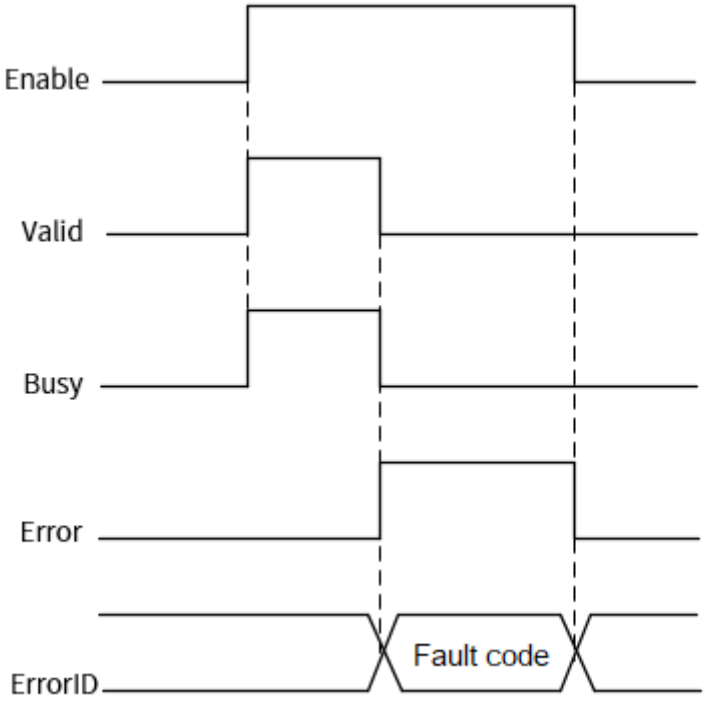
- **Timing diagram**

  The input parameter Enable indicates the start of executing the function block. The input parameter Busy indicates that the current function block is being executed, and the input parameter Valid indicates that the output parameters are valid.

  - When the input parameter Enable is set to TRUE, the output parameter Busy shows the value TRUE.



  - when a fault occurs during the execution of this instruction, the parameter Error shows the value TRUE. You can find the cause of the fault through the output value of ErrorID.



- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

- **Precautions**
  - You cannot execute this instruction in multiple instances simultaneously, i.e., triggering another MC_PositionFilter instruction while this instruction is being executed; otherwise, the later-triggered instruction reports a fault, without affecting the ongoing filter operation.

- When using filter types 2, 3, 4, 5, 8, and 9, note that if the actual position or position reference of the axis and the FilterTime are set to a large value, there may be a significant deviation between the filtered position output in pre-filter periods (input value of FilterTime) and the input position. Specify proper parameter values based on your actual application.
- This instruction checks the value range of input parameters based on the selected filter type. If an input parameter is not applicable to the selected filter type, this instruction will not report a fault even if the parameter value is invalid.
- Note the value of the input parameter PredictTime. When the parameter is set to a large value, the filtered position may differ significantly from the actual position. Specify a proper value for PredictTime based on your needs.
- If a synchronization instruction is executed, and the mcPositionFilter signal source is selected, this instruction must be executed first; otherwise, the synchronization instruction will report a fault. If this instruction and a synchronization instruction are executing, and the mcPositionFilter signal source is selected, disabling this instruction will cause the synchronization instruction to report a fault.
- You cannot execute this instruction while the parameter PowerOff, ErrorStop, or Homing is TRUE.
- Since the fSetXXX information of the encoder axis is 0 and not updated, do not specify the FilterType to 0, 3, 5, 6, 8, or 10 when using this instruction on the encoder axis.
- You can execute this instruction in CSP/CSV control mode only.

18.3.3.3.45 MC_SetOverride

This instruction changes the target velocity of an axis.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_SetOverride	Change target velocity	MC_SetOverride_Instance_0 MC_SetOverride Axis Enabled Enable Busy VelFactor Error AccFactor ErrorID JerkFactor	MC_SetOverride_Instance_0(Axis :=, Enable :=, VelFactor :=, AccFactor :=, JerkFactor :=, Enabled =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: The function block is enabled. FALSE: Disabled.
VelFactor	Velocity override factor	LREAL	0 to 200	100	Specify the velocity override factor. The valid range of the override factor is between 0.01 and 200.00. The unit is %. 100 means 100%. The override factor will be 0 only when 0 is specified. Any values less than 0 will be corrected to the minimum value of 0.01 without a fault message. Any values greater than 200 will be corrected to the maximum value of 200 without a fault message. Note: When the MotionControl library version is 2.2.0.0 or later, the valid range is adjusted to between 0.01 and 500.00.
AccFactor	Acceleration/ Deceleration override factor	LREAL	0 to 100	100	Reserved parameter.
JerkFactor	Jerk override factor	LREAL	0 to 100	100	Reserved parameter.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Enabled	Enabled	BOOL	TRUE, FALSE	FALSE	TRUE when the factor is successfully set.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is being executed.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
VelFactor	Velocity override factor	Any values greater than 200 are treated as 200.	Any values less than 0 are treated as 0.01.	No correction.

- **Function**

  Bus-controlled axis:

  - You can change the target velocity by changing the override factor of the parameter VelFactor for this instruction. For example: New target velocity = Currently velocity reference × Override factor (%).
  - You can change the override factor through the following instructions:

Instructions	
MC_MoveAbsolute	MC_MoveRelative
MC_MoveVelocity	MC_MoveContinuseAbsolute
MC_MoveContinuseRelative	MC_Jog
MC_MoveFeed	MC_Inch

  - When the target velocity after change exceeds the upper limit of the velocity in the software, the axis moves at the maximum velocity.
  - The axis accelerates or decelerates to the target velocity after change. After the setpoint of VelFactor is modified, if the target velocity exceeds the upper limit of the velocity in the software, the axis will move at the maximum velocity, regardless of whether the current axis state is AXIS_STANDSTILL or in motion.
  - When the VelFactor is set to 0, the axis will decelerate if the axis velocity is greater than 0; otherwise the axis will accelerate. When the target velocity becomes 0, the axis is Standstill with no change in the axis state machine. In practical applications, if you want to pause the motion without changing the axis state, you can set the VelFactor to 0.

- This instruction does not affect the slave axis in synchronous motion.
- If the input parameter Enable of this instruction remains high level, the value of the VelFactor can be changed at any time and directly affects the executing motion control instruction. When the Enable is manually set to FALSE or the axis alarm is reset, the override factor changes to 100, which remains unchanged after the axis is disabled and then enabled.
- If the axis state is ERRORSTOP, executing this instruction will result in a fault. When the axis state is STOPPING, this instruction cannot be executed.

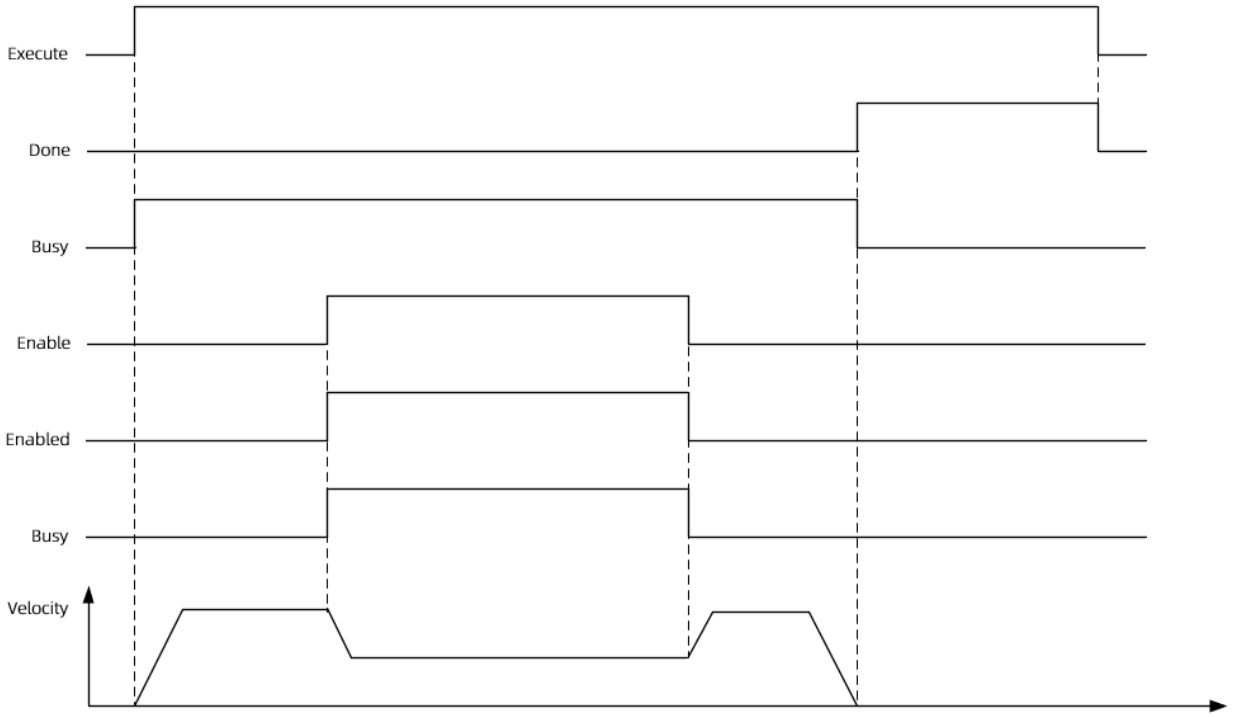
Local pulse axis:

- This instruction can be used for local pulse axes.

- **Timing diagram**
  - Modifying the override factor of the MC_MoveAbsolute instruction

    The following is a timing diagram illustrating the use of the MC_SetOverride instruction for the MC_MoveAbsolute instruction.

    The input parameter Execute of the MC_MoveAbsolute instruction must be triggered on a rising edge. When the parameter Busy is set to TRUE, it indicates that the current function block is being executed. During this process, the input parameter Enable of the MC_SetOverride instruction must be triggered on a rising edge. When the output parameter Enabled is set to TRUE, it indicates that the instruction has been executed normally. When the output parameter Busy is set to TRUE, it indicates that the current function block is being executed.

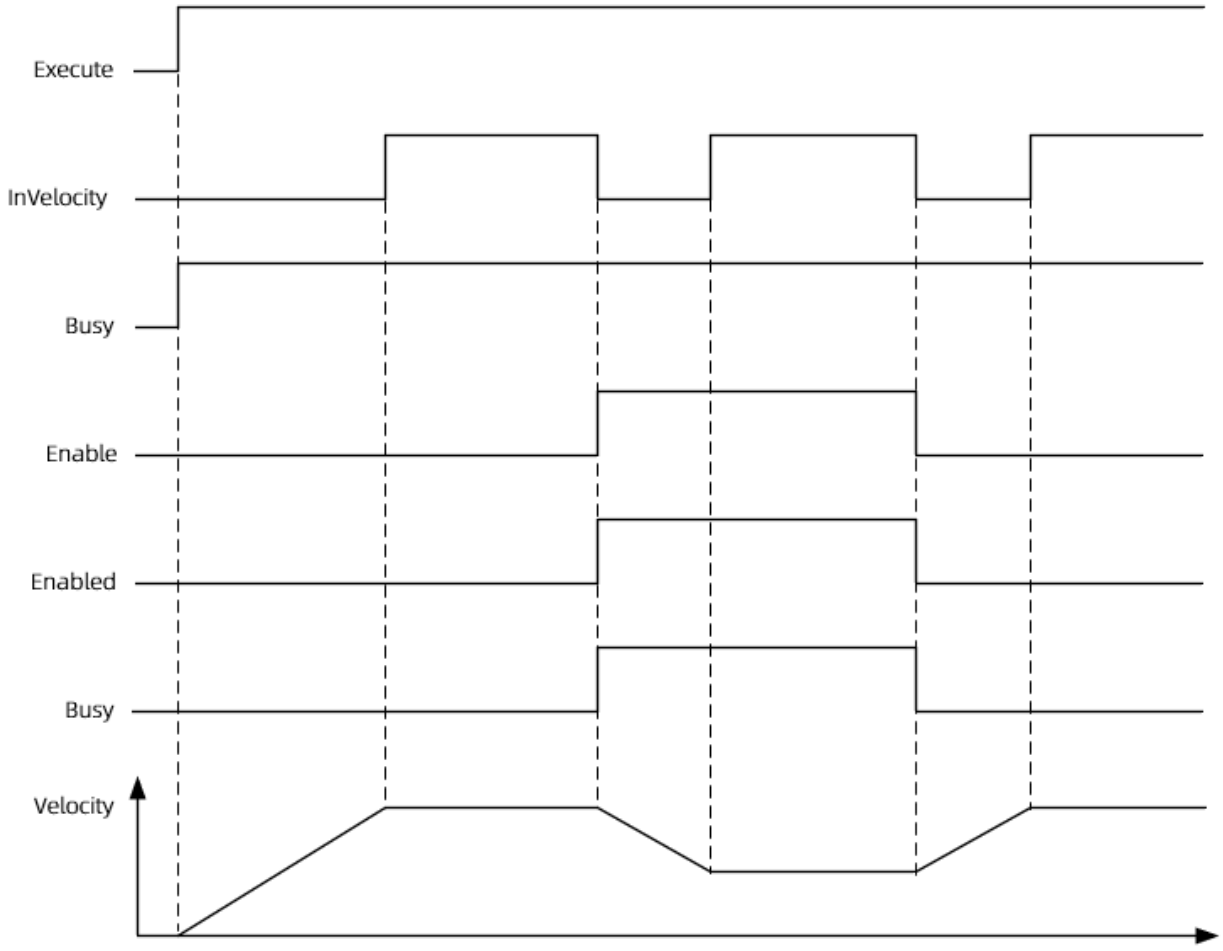


  - Modifying the override factor of the MC_MoveVelocity

    The following is a timing diagram illustrating the use of the MC_SetOverride instruction for the MC_MoveVelocity.

    The input parameter Execute of the MC_MoveVelocity instruction must be triggered on a rising edge. When the parameter Busy is set to TRUE, it indicates that the current function block is being executed. During this process, the input parameter Enable of the MC_SetOverride instruction must be triggered on a rising edge. When the output parameter Enabled is set to

TRUE, it indicates that the instruction has been executed normally. When the output parameter Busy is set to TRUE, it indicates that the current function block is being executed.



- **Re-execution of motion control instructions**

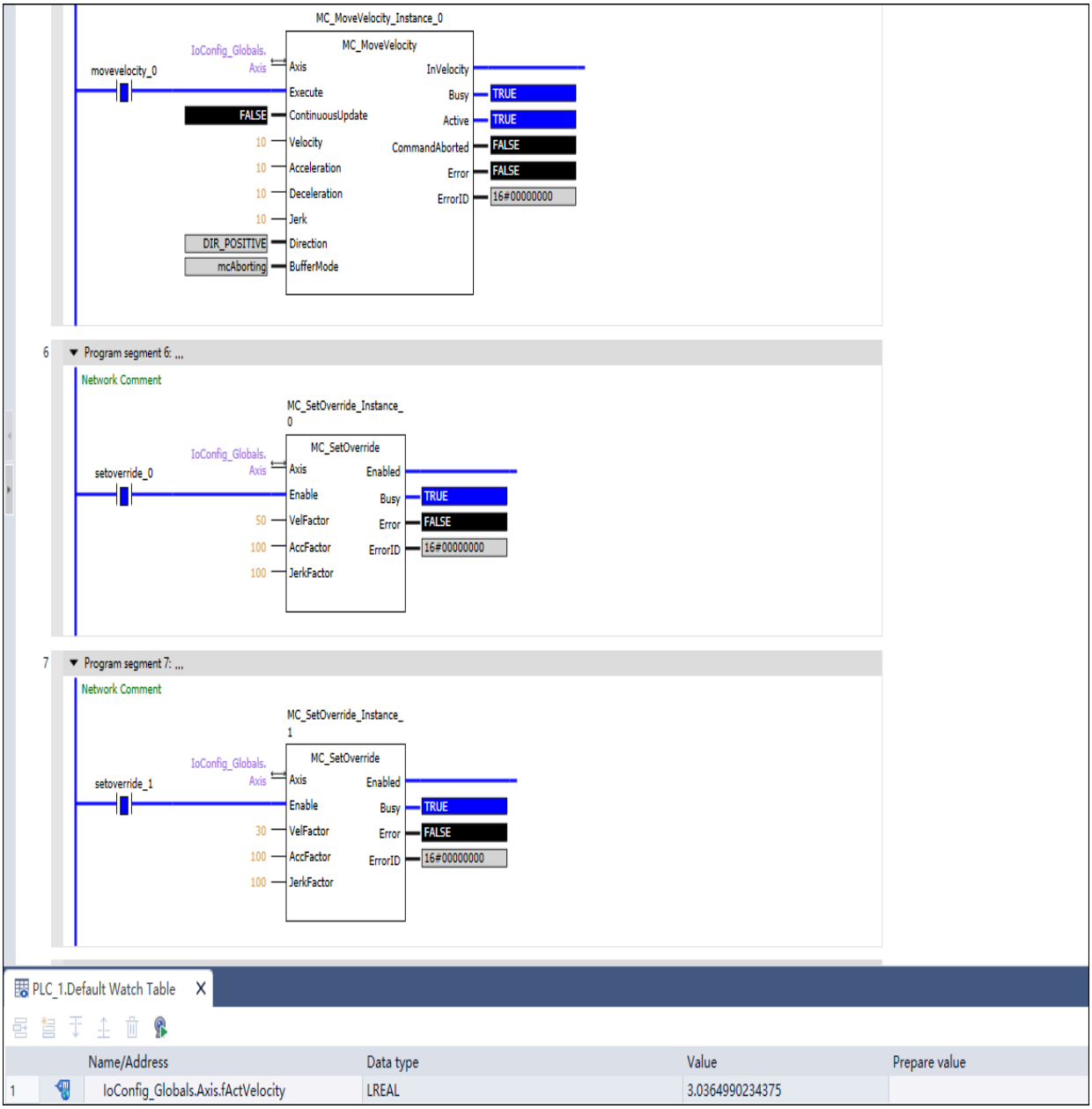
  You cannot re-execute motion control instructions with an input parameter Enable.

- **Faults**

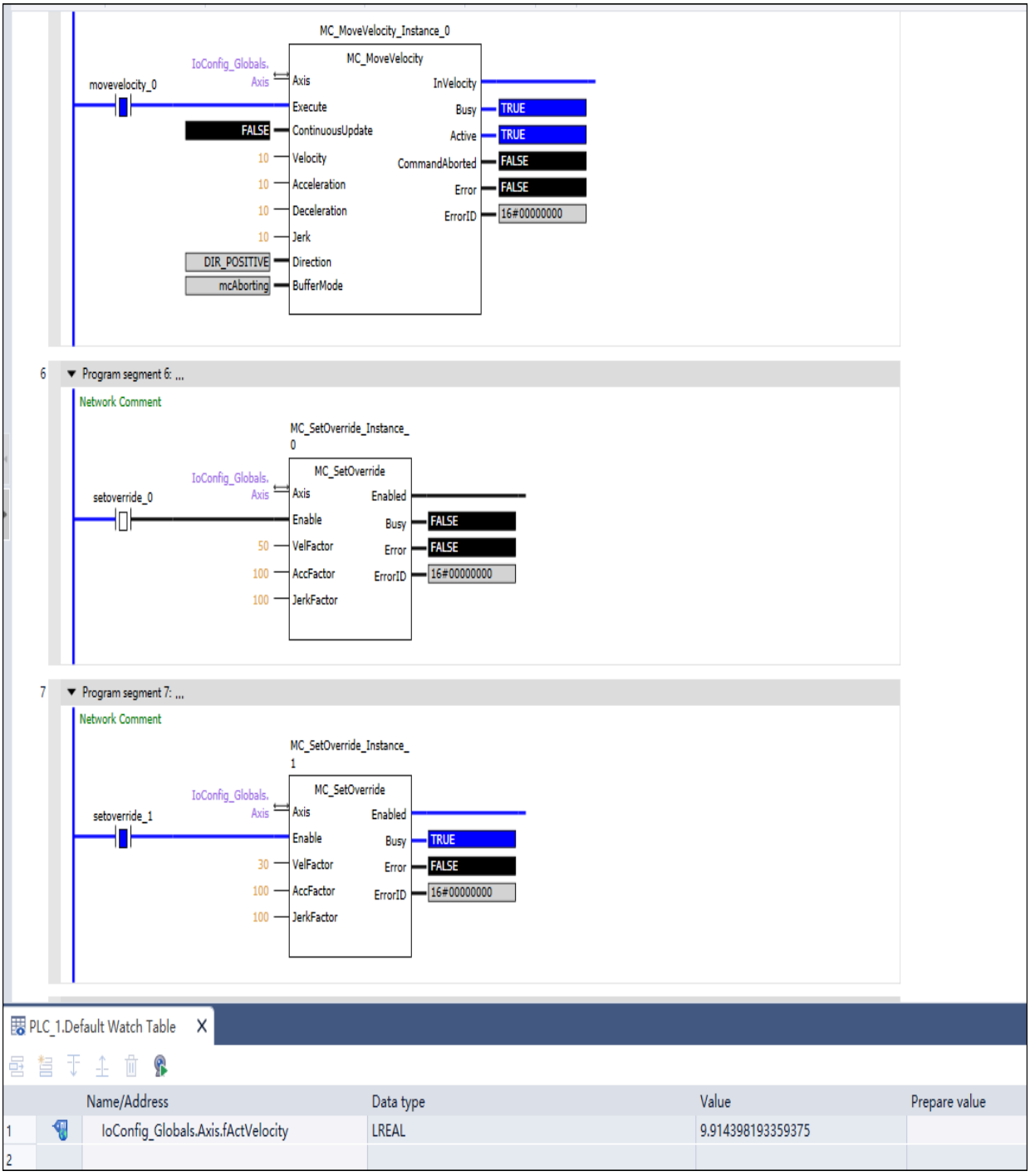
  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

- **Precautions**

  If another instances of MC_SetOverride are executed on the same axis during MC_SetOverride instruction execution, the last MC_SetOverride instance that is executed takes priority in processing. If this instruction is executed with the following configurations, the velocity of the final instruction will be 30% of the original velocity, as shown in the following figure.



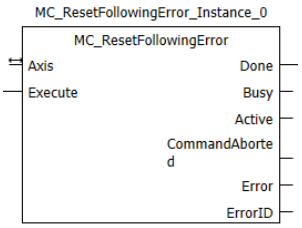
At this point, canceling any of the MC_SetOverride instructions will reset the VelFactor to 100%, even if the canceled instruction is not the currently active one, and the currently active instruction will no longer be active. If the instruction with VelFactor = 50% is canceled, the currently active instruction with VelFactor = 30% will no longer be active, as shown in the following figure.



18.3.3.3.46 MC_ResetFollowingError

This instruction resets the following error between the current position reference and the current actual position.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_ResetFollowingError	Reset following error		MC_ResetFollowingError_Instance_0(Axis :=, Execute :=, Done =>, Busy =>, Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Active	Controlling	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus-controlled axis:

  - This instruction resets the following error between the current position reference and the current actual position to 0 in MC_CSP mode.
  - During the execution of this instruction, do not execute another instance of this instruction; otherwise, the subsequently triggered function block will report a fault.
  - This instruction can be triggered only when the axis type is control axis and the control mode is MC_CSP; otherwise, the function block will report a fault.
  - When the parameter Execute is set to TRUE, the actual position at that point is used as the position reference (i.e., the target position).
  - When this instruction is triggered, the axis state switches to AXIS_STOPPING. After the instruction is done, the axis state switches to AXIS_STANDSTILL.
  - When the axis state is AXIS_HOMING, this instruction cannot be triggered; otherwise, the function block will report a fault.
  - When the axis state is AXIS_STOPPING, this instruction will exit immediately after being triggered, with the output parameter CommandAborted showing the value TRUE.
  - Do not trigger the MC_Stop or MC_SetPosition instruction during the execution of this instruction.
  - In Rotary Mode, when this instruction is called for compensation, compensation is performed in the direction with the shortest distance, and the compensation position is calculated based on the difference between the target position and the actual position after multiple rotation cycles.
  - If this instruction is triggered during the axis motion, there is a jitter deviation after the first error elimination. It is recommended to trigger the instruction again after the axis stops to eliminate the deviation.

  Local pulse axis:

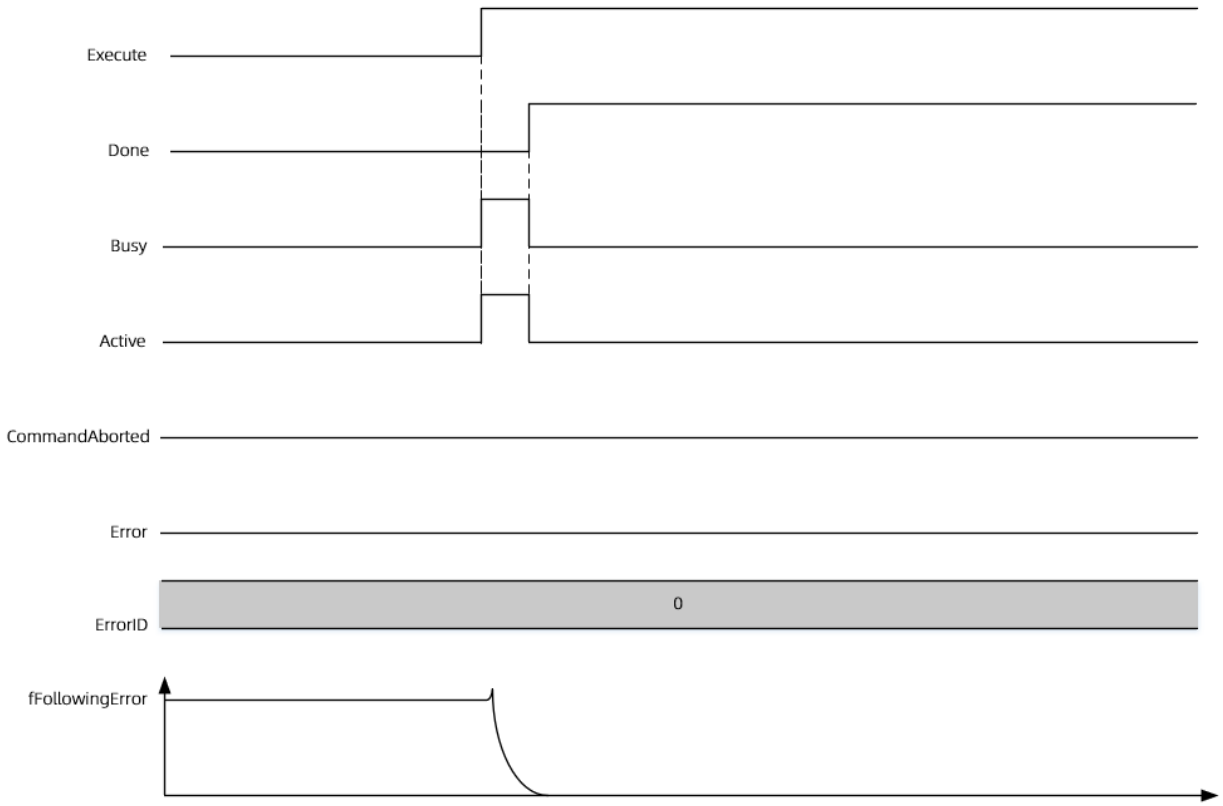
  - This instruction can be used for local pulse axes.

- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates the instruction has been executed successfully; the parameter Busy indicates that the current function block is being executed; the parameter CommandAborted of the function block indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

  Use the MC_InPosition instruction to check the current deviation. Use the IsTimeOut output flag as the input for the variable Execute. The output parameter Busy is set to TRUE on a rising edge of Execute. In the next cycle, the output parameter Active shows the value TRUE.

  The axis clears the deviation according to the parameters of this instruction until the following error between the current position reference and the actual position is 0, at which point the output parameter Done is set to TRUE.



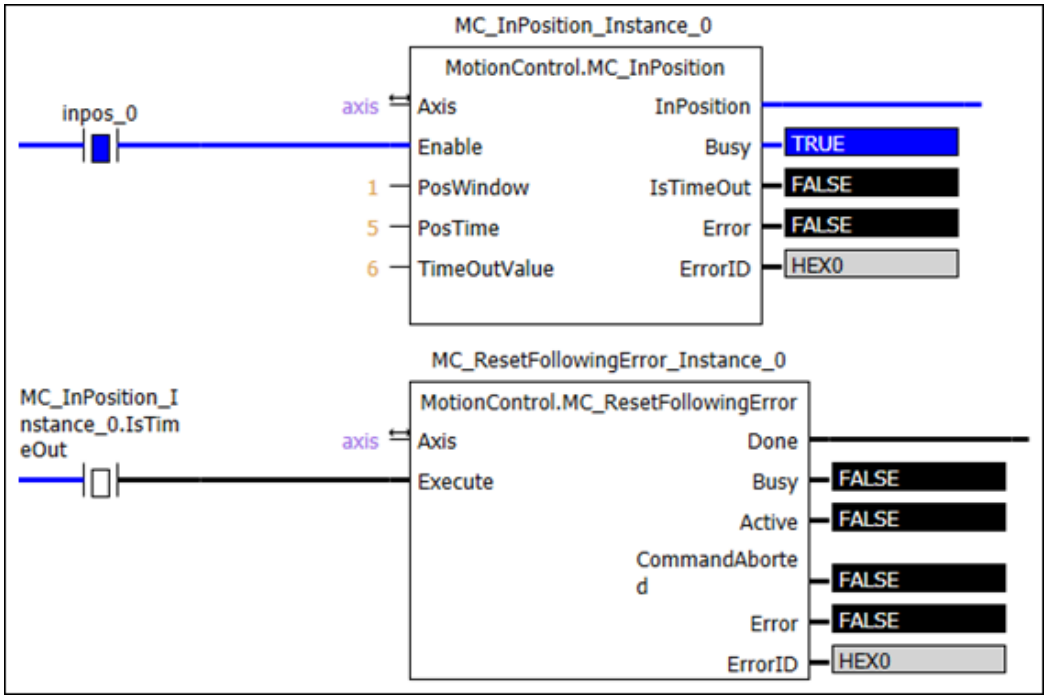
- **Re-execution of motion control instructions**

  This instruction cannot be re-executed.

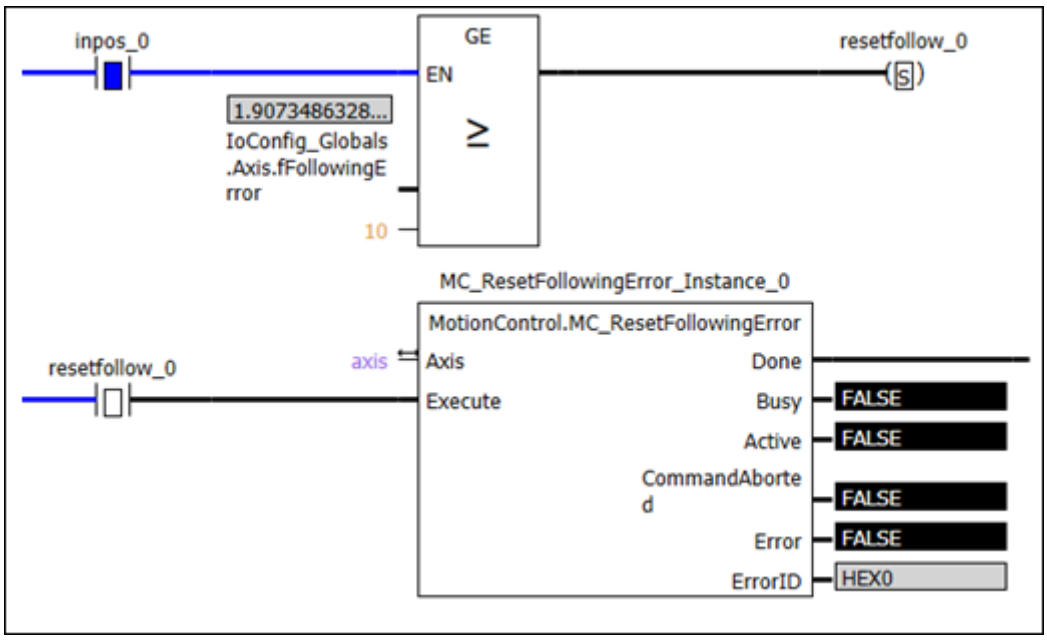
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

- **Example**
  - If the actual trajectory does not reach the planned trajectory, you can use this instruction to abort the current motion, clear the deviation, and recheck the user program. You can first use the MC_InPosition instruction to check the current deviation, specify a proper deviation for the parameter PosWindow, and specify the TimeOutValue (s). If the deviation does not fall within a reasonable range within the specified cycle, the IsTimeOut instruction will be set. You can use the IsTimeOut output flag as the input for the MC_FollowPositionError instruction.



◦ In case of motor stalling or excessive deviation due to external factors, it is necessary to constantly monitor the operation to avoid more serious issues such as servo drive stall or deviation error. At this time, you can observe the variable FfollowingError, and execute the MC_ResetFollowingError instruction to terminate motion and clear the deviation when the deviation exceeds the threshold.

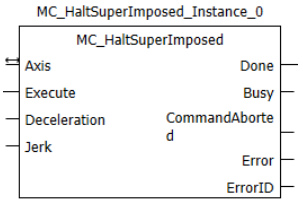


- **Precautions**
  - ◦ The actual deviation source may be insufficient rigidity. Therefore, you must ensure sufficient servo rigidity (H09.01 for our product). This instruction is not applicable to some applications requiring lower rigidity.
  - ◦ The actual deviation source may be an excessively high motor speed. Therefore, you must manually adjust the maximum value for deviation detection based on different speed conditions. To prevent jitter error, a hold-off time is recommended to be included in the user-designed deviation detection program.
  - ◦ The instruction clears the deviation according to the maximum velocity limit in the software. You must design a proper axis velocity limit to prevent stalling due to too rapid velocity planning after the instruction is triggered.

18.3.3.3.47 MC_HaltSuperImposed

This instruction halts the superimposed motion of an axis and remains its basic motion unchanged.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_HaltSuperImposed	Halt superimposed motion		MC_HaltSuperImposed_Instance_0(Axis :=, Execute :=, Deceleration :=, Jerk :=, Done =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.
Deceleration	Deceleration	LREAL	Positive number	0	Deceleration of the deceleration phase.
Jerk	Jerk	LREAL	Positive number	0	Slope of acceleration or deceleration.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is being executed.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the deceleration limit.	Any deceleration less than 0 will be automatically corrected to 10% of the deceleration limit.	The deceleration equal to 0 will be automatically corrected to 10% of the deceleration limit.
Jerk	Jerk	If the motion profile is set to S-curve, any jerk that exceeds the software limit will be automatically corrected to the jerk limit.	If the motion profile is set to S-curve, any jerk less than 0 will be automatically corrected to 10% of the jerk limit.	If the motion profile is set to S-curve, the jerk equal to 0 will be automatically corrected to 10% of the jerk limit.

- **Function**

  Bus-controlled axis:

  - This instruction is used to halt the ongoing superimposed motion for an axis and remain its basic motion unchanged.
  - When the axis state is AXIS_STOPPING or AXIS_HOMING, this instruction will exit immediately after being triggered, with the output parameter CommandAborted showing the value TRUE.
  - The axis type must be control axis.

- If the MC_MoveSuperImposed instruction is not executing, triggering this instruction will result in a fault.
- This instruction cannot be triggered repeatedly during execution; otherwise, the function block reports a fault.
- If this instruction reports a fault, the executing instruction will be aborted.
- This instruction can be aborted by other motion control instructions during execution.
- In PLC projects, this instruction cannot be placed and triggered before the main motion of the specified axis.

Local pulse axis:

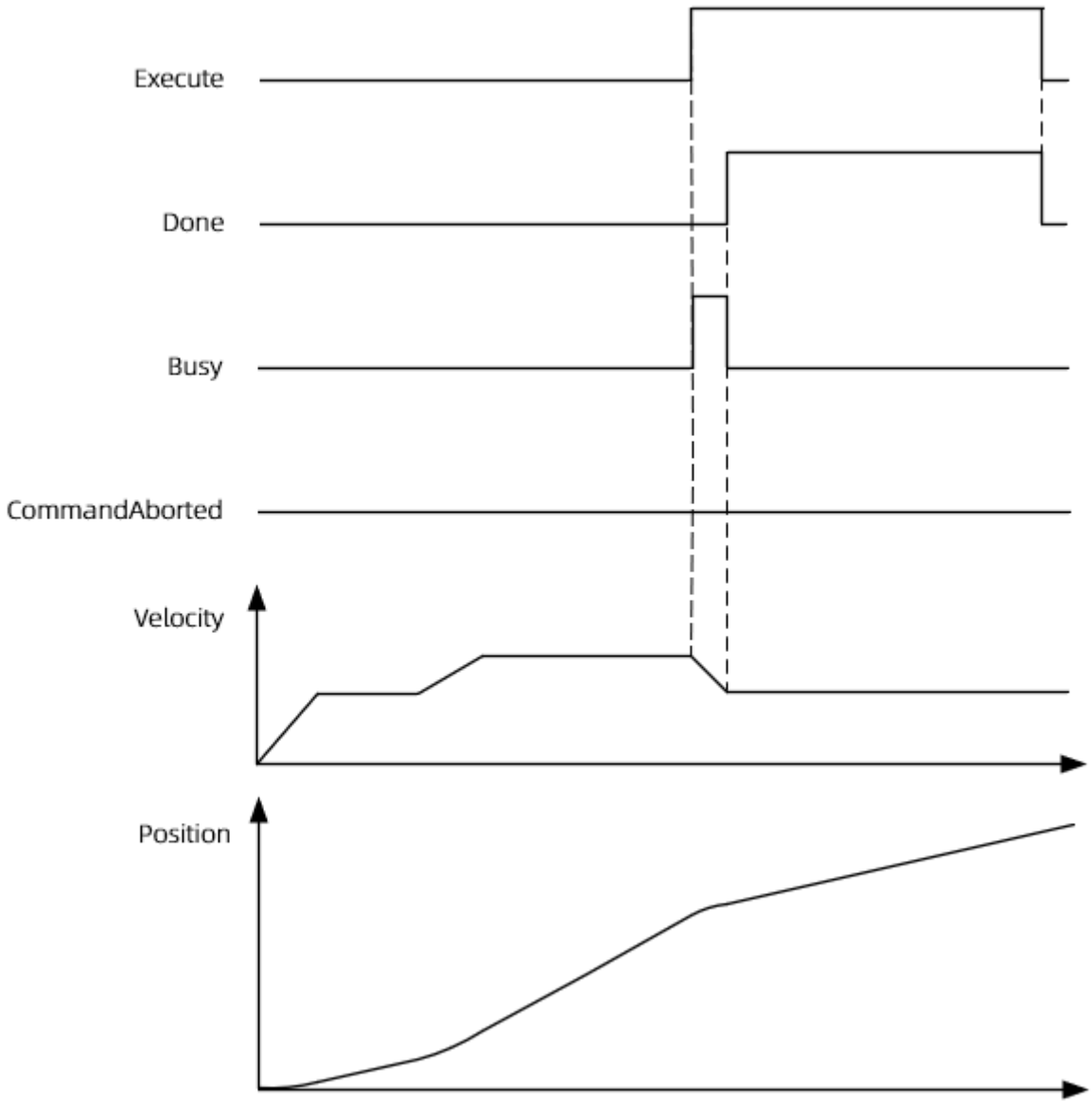
- This instruction can be used for local pulse axes.

- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates the instruction has been executed successfully; the parameter Busy indicates that the current function block is being executed; the parameter CommandAborted indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

  After this instruction is called, the output parameter Busy is set to TRUE on a rising edge of Execute.

  The output parameter Done is set to TRUE when the axis decelerates to the velocity before the superimposed motion according to the parameters of this instruction.



- **Re-execution of motion control instructions**

  This instruction cannot be re-executed.

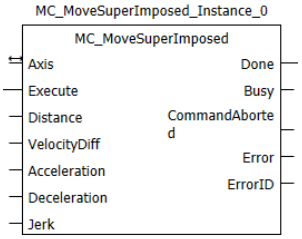
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.48 MC_MoveSuperImposed

This instruction enables the axis to move with additional velocity and distance on top of the original velocity and distance reference.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_MoveSuperImposed	Superimposed motion		MC_MoveSuperImposed_Instance_0(Axis :=, Execute :=, Distance :=, VelocityDiff :=, Acceleration :=, Deceleration :=, Jerk :=, Done =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.
Distance	Additional distance	LREAL	Positive number, negative number, or 0	0	Additional distance for overlapping positioning.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
VelocityDiff	Additional velocity	LREAL	Positive number	0	Additional velocity on top of the original velocity reference.
Acceleration	Acceleration	LREAL	Positive number	0	Acceleration of the acceleration phase.
Deceleration	Deceleration	LREAL	Positive number	0	Deceleration of the deceleration phase.
Jerk	Jerk	LREAL	Positive number or 0	0	Slope of acceleration or deceleration.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is being executed.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Acceleration	Acceleration	Any acceleration that exceeds the motion parameter limit will be automatically corrected to the acceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration less than 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration equal to 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration equal to 0 will be automatically corrected to 10 times the velocity.
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the deceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration less than 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration equal to 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration equal to 0 will be automatically corrected to 10 times the velocity.
Jerk	Jerk	If the motion profile is set to S-curve, any jerk that exceeds the software limit will be automatically corrected to the jerk limit.	If the motion profile is set to S-curve, any jerk less than 0 will be automatically corrected to 10% of the jerk limit.	If the motion profile is set to S-curve, the jerk equal to 0 will be automatically corrected to 10% of the jerk limit.

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity Diff	Additional velocity	Any velocity that exceeds the motion parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	• When the MotionControl library version is 2.1.0.0 or earlier, the velocity equal to 0 will be automatically corrected to 1e-5. • When the MotionControl library version is 2.2.0.0 or later, the velocity equal to 0 will not be corrected.

- **Function**

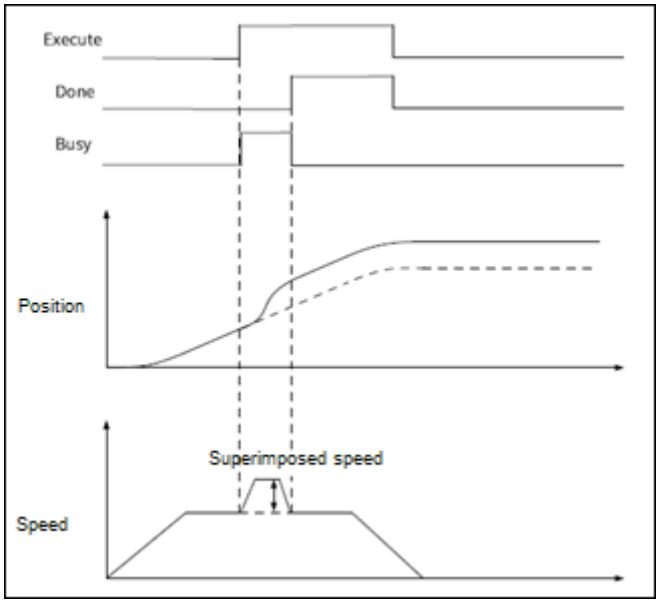
  Bus-controlled axis:

  - This instruction enables the axis to move with additional distance and velocity on top of the active motion control instruction, compensating for the error caused by the clearance between the belt and gears in actual project.
  - The input parameters VelocityDiff and Distance are the velocity and distance to be added on the active motion control instruction.
  - This instruction must be triggered in CSP mode.
  - During the execution of this instruction, do not execute another instance of this instruction; otherwise, the function block reports a fault.
  - When the axis state is STOPPING or HOMING, this instruction will exit immediately after being triggered, with the output parameter CommandAborted showing the value TRUE.
  - This instruction cannot be called during the execution of the MC_Halt instruction; otherwise, the function block reports a fault.
  - When multiple instructions are buffered, the specified additional values for superimposed motion must be reached as much as possible before the active main motion is finished. The specified additional values for superimposed motion will not be carried over to the motion in the buffer. When a buffered instruction is executed, this instruction is aborted with the reached additional values for superimposed motion being retained.
  - If a fault occurs during execution and the current velocity reference is not 0, the axis will decelerate to stop under a T-curve planning, using the maximum deceleration in the software.
  - The motion status of the master axis is not affected by this instruction.
  - If this instruction reports a fault, the executing instruction will be aborted.
  - **Instruction execution in different control modes**

    Single axis positioning instructions

    - When the axis state is AXIS_STANDSTILL, this instruction performs a function equivalent to the MC_MoveRelative instruction during execution, with the axis state switching to AXIS_DISCRETE_MOTION.

- When this instruction is called during the execution of other instructions that put a servo axis in CSP mode (MC_MoveVelocity, MC_MoveAbsolute, MC_MoveRelative, MC_MoveContinuousAbsolute, MC_MoveContinuousRelative, MC_GearIn, MC_CamIn, MC_GearInPos), it performs superimposed motion. The current axis state is determined by the main motion. It switches to AXIS_DISCRETE_MOTION of superimposed motion only after the motion has ended.



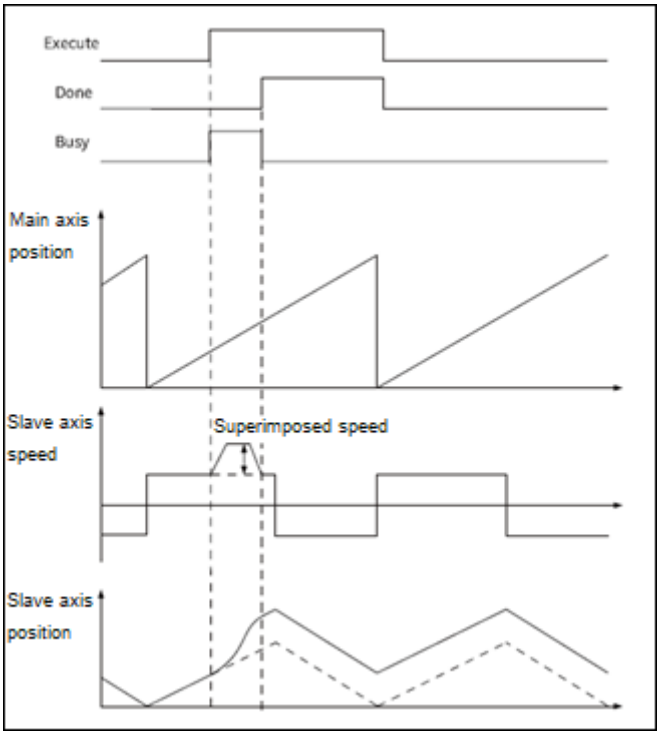
- If any other instructions that put an axis in CSP mode (MC_MoveVelocity, MC_MoveAbsolute, MC_MoveRelative, MC_MoveContinuousAbsolute, MC_MoveContinuousRelative, MC_GearIn, MC_CamIn, MC_GearInPos) are triggered during the execution of this instruction, this instruction will be aborted.

Axis group instructions

- If this instruction is called during the execution of an axis group instruction, it will report a fault without performing the superimposed motion.

Synchronous axis instructions

- When this instruction is called during the execution of the MC_GearIn or MC_CamIn instruction by the slave axis, it performs the superimposed action.



- If the MC_GearIn instruction is re-executed during the execution of this instruction, this instruction will be aborted.
- If the MC_CamIn instruction is buffered, the re-triggered parameters can take effect. If the MC_CamIn instruction has been executed, re-executing this instruction will report a fault.
- If this instruction is executed together with a synchronous motion control instruction, the MC_GearOut and MC_CamOut instructions can terminate their execution.

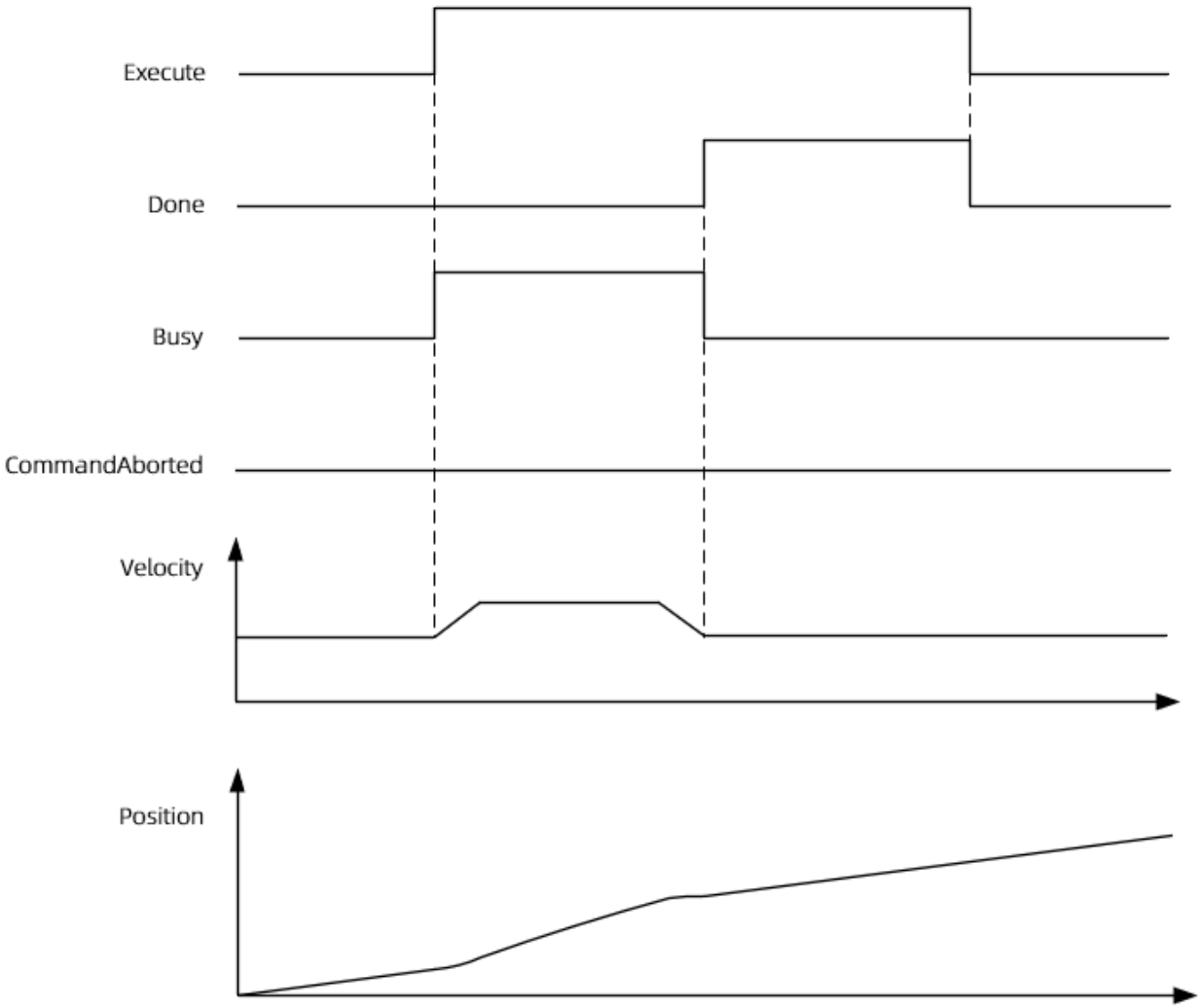
Local pulse axis:

- This instruction can be used for local pulse axes.

- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates the instruction has been executed successfully; the parameter Busy indicates that the current function block is being executed; the parameter CommandAborted indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

  After this instruction is called, the output parameter Busy is set to TRUE on a rising edge of Execute. The output parameter Done is set to TRUE when the axis reaches the specified Distance and VelocityDiff on the top of original distance and velocity reference.



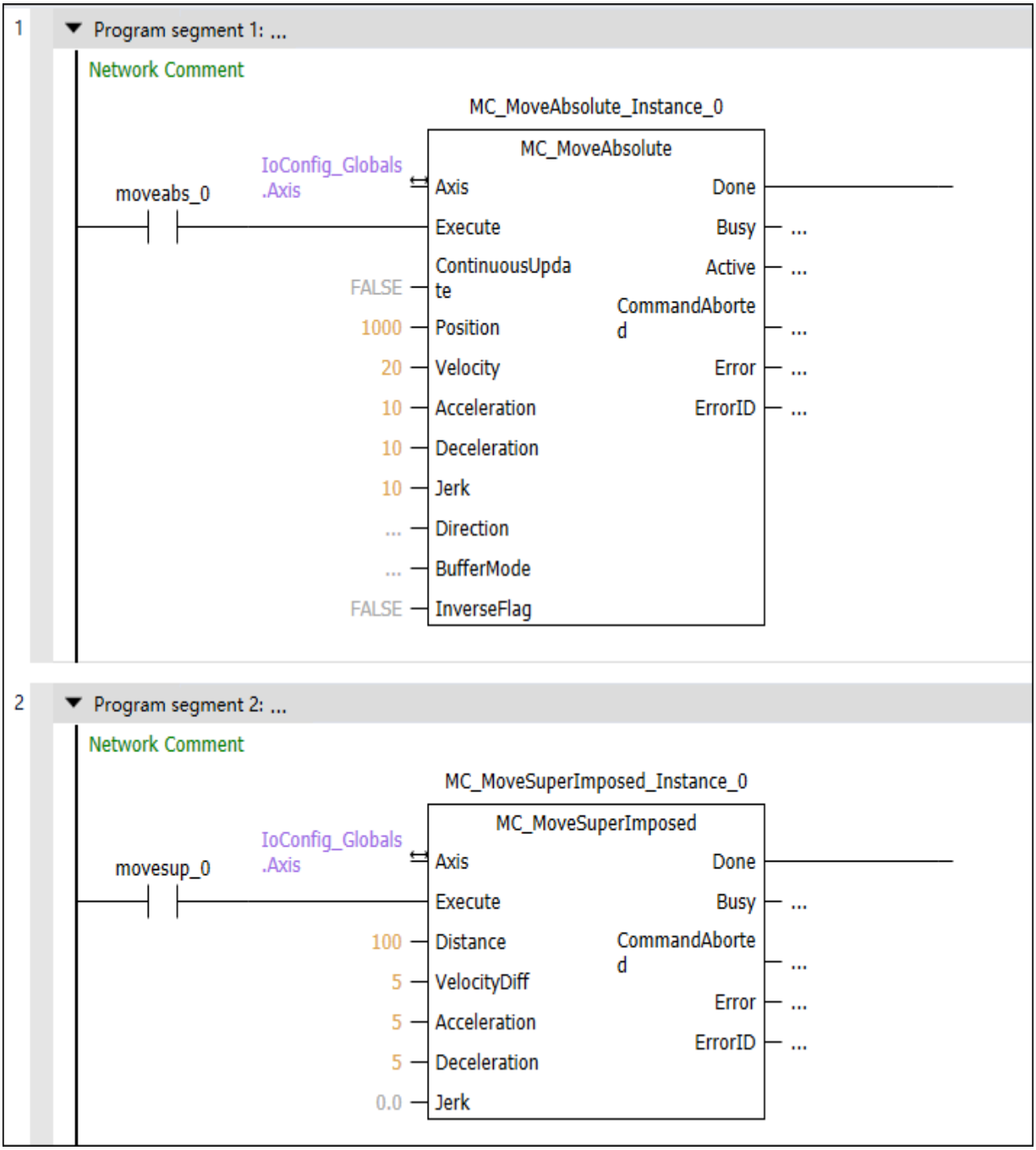
- **Re-execution of motion control instructions**

  This instruction cannot be re-executed.

- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

- **Precautions**
  - Superimposed motion cannot be carried over to the buffered instructions.
  - The instruction must be triggered after the main motion has been triggered and completed; otherwise, the logic determines that the main motion aborts the superimposed motion instead of the superimposed motion being added to the main motion.

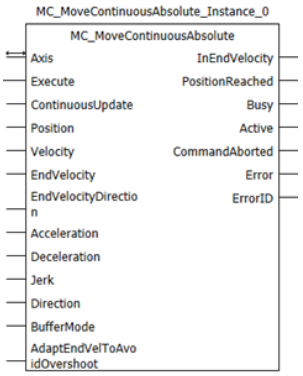


◦ If the combined velocity of this instruction and the main motion exceeds the velocity limit, the instruction will be terminated with a fault message, while the main motion will not be affected. Note the current master axis state before execution of this instruction.

18.3.3.3.49 MC_MoveContinuousAbsolute

This instruction controls the axis to continuously move in absolute position.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_MoveContinuousAbsolute	Move to a specified absolute position ending with the specified velocity		MC_MoveContinuousAbsolute _Instance_0(Axis :=, Execute :=, ContinuousUpdate :=, Position :=, Velocity :=, EndVelocity :=, EndVelocityDirection :=, Acceleration :=, Deceleration :=, Jerk :=, Direction :=, BufferMode :=, AdaptEndVelToAvoidOvershoot := , InEndVelocity =>, PositionReached => , Busy =>, Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.
Continuous Update	Continuous update	BOOL	TRUE, FALSE	FALSE	Continuous parameter update effective flag. Prerequisite: The parameter ContinuousUpdate shows the value TRUE and the parameter Active shows the value TRUE. Effective parameters: Position, Velocity, EndVelocity, EndVelocityDirection. Note: The modified parameter values must differ from the original parameter values by more than 0.01.
Position	Absolute Position	LREAL	Positive number, negative number, or 0	0	Target position in absolute coordinates.
Velocity	Velocity	LREAL	Positive number	0	Maximum velocity.
EndVelocity	End velocity	LREAL	Positive number or 0	0	End velocity.
EndVelocity Direction	End velocity direction	18.3.3.2.2.9 MC_DIRECTION: Motion Direction Enumeration	1: DIR_POSITIVE 2: DIR_NEGATIVE 3: DIR_CURRENT	3	1: DIR_POSITIVE: Positive direction. 2: DIR_NEGATIVE: Negative direction. 3: DIR_CURRENT: Current direction.
Acceleration	Acceleration	LREAL	Positive number	0	Acceleration of the acceleration phase.
Deceleration	Deceleration	LREAL	Positive number	0	Deceleration of the deceleration phase.
Jerk	Jerk	LREAL	Positive number or 0	0	Slope of acceleration or deceleration.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Direction	Motion direction	18.3.3.2.2.9 MC_DIRECTION: Motion Direction Enumeration	0: DIR_SHORTEST 1: DIR_POSITIVE 2: DIR_NEGATIVE 3: DIR_CURRENT 4: DIR_NODIRECTION 5: DIR_FASTEST	1	Specify one of the following directions for linear axes: 1: DIR_POSITIVE (Positive direction). 2: DIR_NEGATIVE (Negative direction). Specify one of the following rotation directions for rotary axes: 0: DIR_SHORTEST (Direction based on the shortest way). 1: DIR_POSITIVE (Positive direction). 2: DIR_NEGATIVE (Negative direction). 3: DIR_CURRENT (Current direction). 4: DIR_NODIRECTION (No direction). 5: DIR_FASTEST (Direction based on the shortest movement time).
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting 1: mcBuffered	0	0: mcAborting (Aborting). 1: mcBuffered (Buffered).
AdaptEndVelToAvoidOvershoot	End velocity adaptation flag	BOOL	TRUE, FALSE	FALSE	TRUE: Adapt the end velocity to avoid overshoot. FALSE: No adaptation.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
InEndVelocity	End velocity reached	BOOL	TRUE, FALSE	FALSE	TRUE when the end velocity is reached.
PositionReached	Position reached	BOOL	TRUE, FALSE	FALSE	TRUE when the expected position is reached.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is being executed.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Active	Controlling	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled by the current FB.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity	Velocity	Any velocity that exceeds the motion parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	• When the MotionControl library version is 2.1.0.0 or earlier, the velocity equal to 0 will be automatically corrected to 1e-5. • When the MotionControl library version is 2.2.0.0 or later, the velocity equal to 0 will not be corrected.

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Acceleration	Acceleration	Any acceleration that exceeds the motion parameter limit will be automatically corrected to the acceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration less than 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration equal to 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration equal to 0 will be automatically corrected to 10 times the velocity.
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the deceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration less than 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration equal to 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration equal to 0 will be automatically corrected to 10 times the velocity.
Jerk	Jerk	If the motion profile is set to S-curve, any jerk that exceeds the software limit will be automatically corrected to the jerk limit.	If the motion profile is set to S-curve, any jerk less than 0 will be automatically corrected to 10% of the jerk limit.	If the motion profile is set to S-curve, the jerk equal to 0 will be automatically corrected to 10% of the jerk limit.
EndVelocity	End velocity	Any velocity that exceeds the motion parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	No correction.

- **Function**

Bus-controlled axis:

- This instruction controls the axis to move to an absolute position ending with the velocity reference.
- The setpoint of the input parameter Position specifies the absolute position where the axis velocity reaches the specified end velocity. The setpoint of the input parameter Velocity specifies the maximum velocity during the planning process. The setpoint of the input parameter EndVelocity specifies the end velocity of the axis.
- The setpoint of the input parameter EndVelocityDirection specifies the direction of the EndVelocity. The setpoint of the input parameter Direction specifies the direction of the Velocity, which is valid in Rotary Mode only. In Linear Mode, the direction of the Velocity is determined by the positive or negative value of the Position.
- When the instruction is triggered on a rising edge of the input parameter Execute, the input parameters are checked for validity. If any of invalid parameters is detected, it will be automatically corrected by the function block without a fault message; any buffered instructions after this instruction will be cleared.
- When the input parameter ContinuousUpdate is set to TRUE, modified parameters Position, Velocity, EndVelocity and EndVelocityDirection will take effect only when the instruction is in the Active state. Parameter changes greater than 0.01 are considered updates. Modified BufferMode cannot take effect during this period, and the buffer mode at the time of the instruction trigger is retained.
- The axis type must be control axis.
- If the software limit is hit during the instruction execution, the parameter Error is set with the axis state being set to ErrorStop.
- During the deceleration due to limits, do not execute the MC_Reset instruction to reset fault messages.
- During the execution of this instruction, modifying the BufferMode and re-executing the instruction will cause the buffer to be cleared with a fault message.
- When the current axis is not in CSP mode, the instruction internally switches the control mode.
- During the execution of this instruction, the MC_MoveSuperImposed instruction can be used.
- When the axis is enabled after PLC cold reset and the MC_Home instruction is executed using No. 35 homing method, if this instruction is executed with DIR_CURRENT being set, the motion direction of this instruction will be affected.
- When this instruction is executed with the ContinuousUpdate being set to TRUE along with the MC_SuperImposed instruction, the MC_SuperImposed instruction will be aborted if any modifiable parameters are modified.

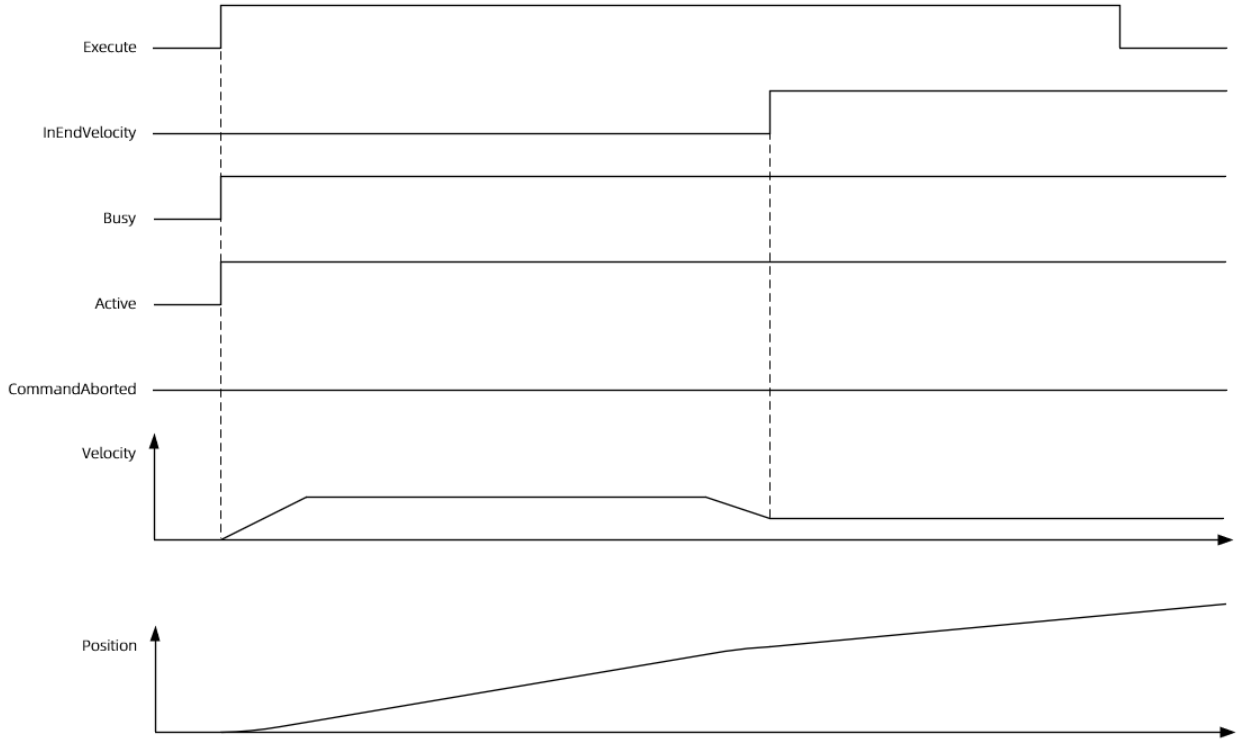
Local pulse axis:

- This instruction can be used for local pulse axes.
- When the local pulse axis is used with Y0 or Y1 single-ended output, it is not recommended to use this instruction. ① No pulse output; ② No position feedback.

- **Timing diagram**

  The input parameter Execute must be triggered on a rising edge. When the output parameter Busy is set to TRUE, it indicates that the current function block is being executed. When the output parameter Busy is set to TRUE, it indicates the current function block is being executed.

After this instruction is called, the output parameter Busy is set to TRUE on a rising edge of Execute. The output parameter Active shows the value TRUE in the same cycle. When the end velocity is reached, the output parameter InEndVelocity is set to TRUE.

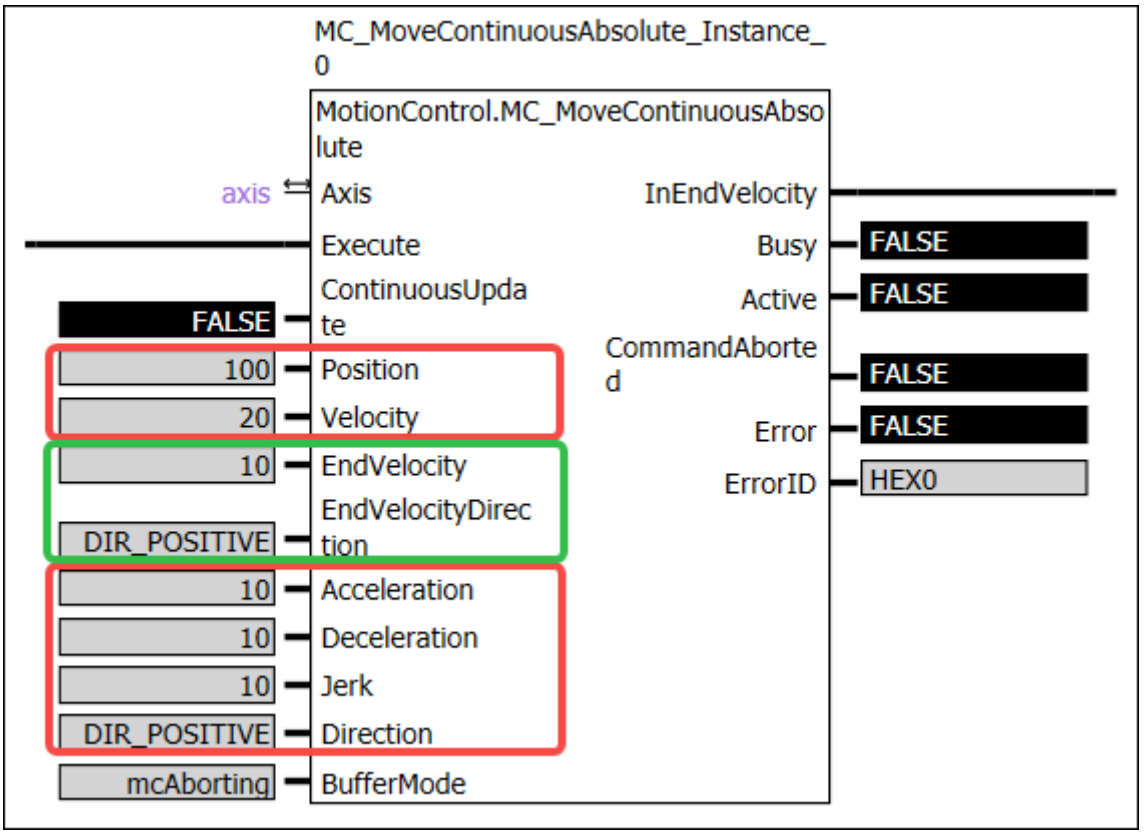


- **Faults**

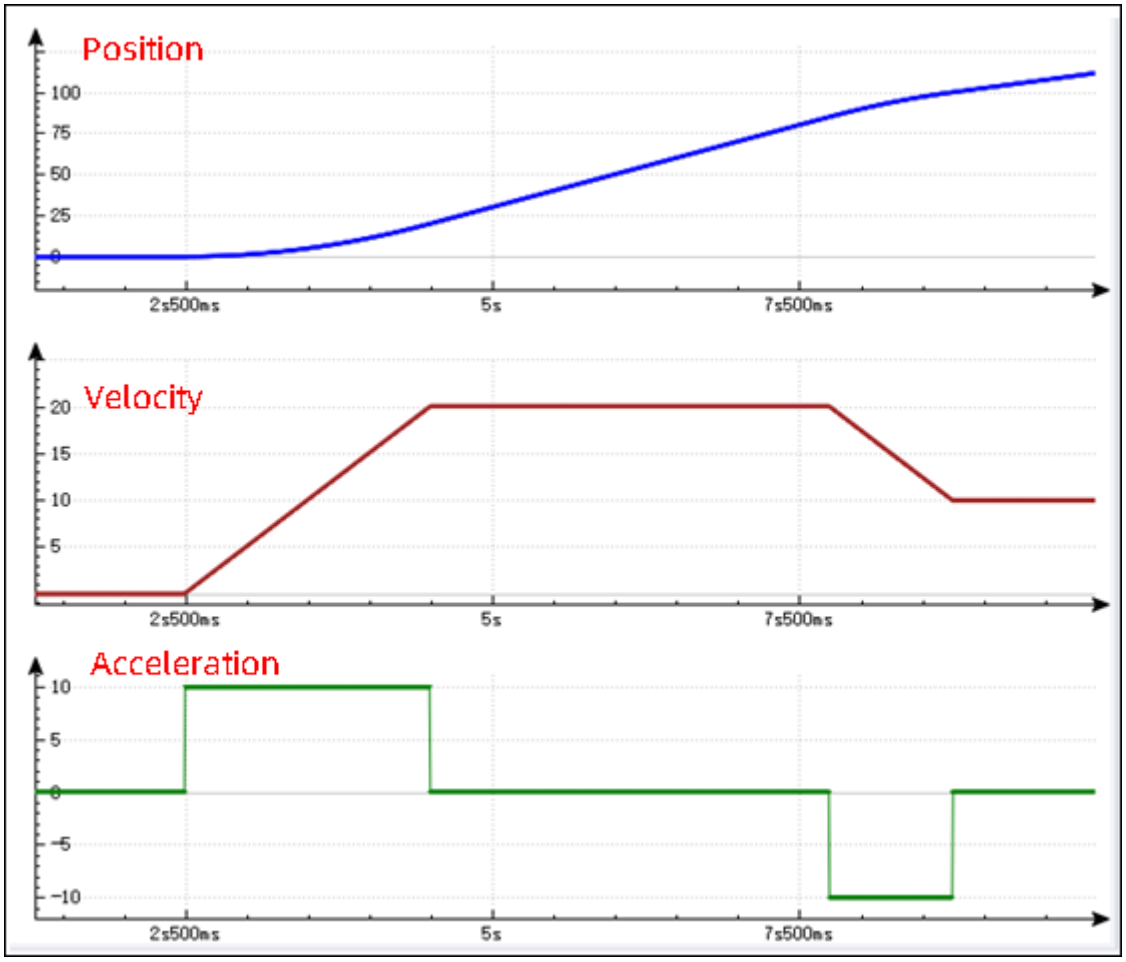
  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

- **Example**

  Specify the planned values for the motion parameters Position, Velocity, Acceleration, Deceleration, Jerk, and Direction of this instruction, and then specify the EndVelocity and EndVelocityDirection.

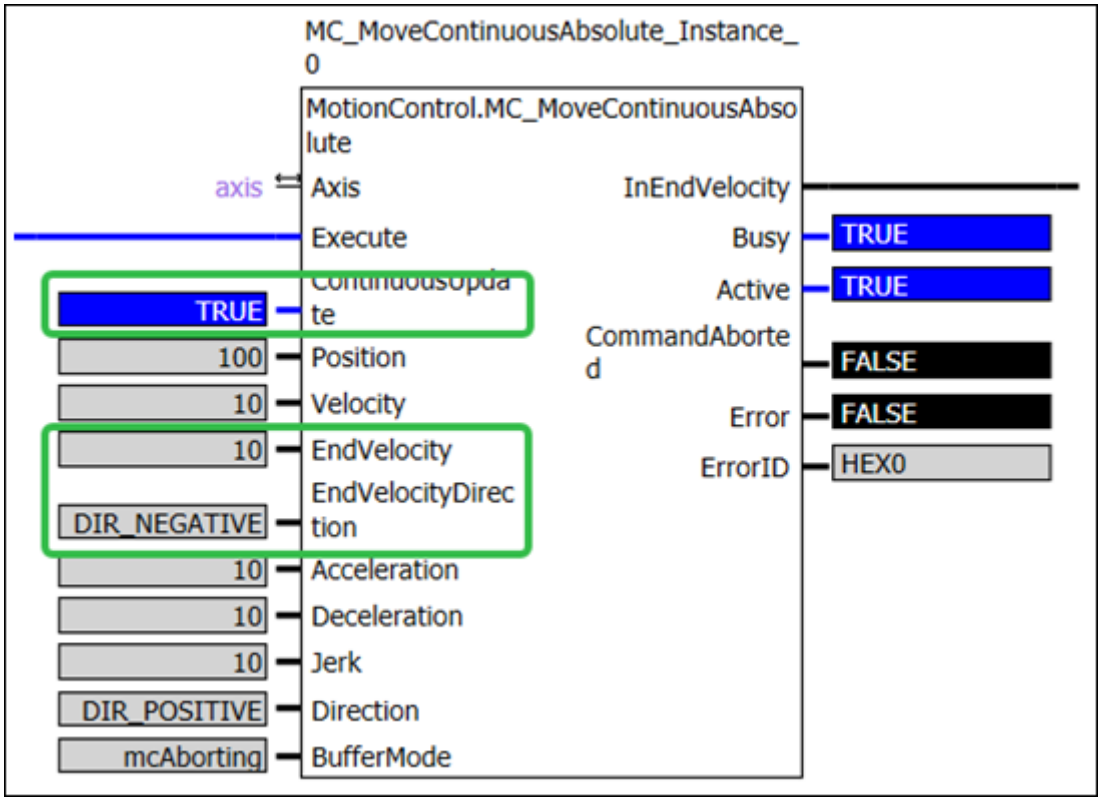


After the Execute of this instruction is set, the instruction controls the axis to move from the initial position to position 100 according to the specified motion parameters and then continue moving at a velocity of 10.



- **Precautions**

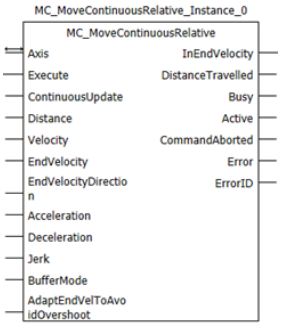
  The ContinuousUpdate function is effective for modified basic motion parameters before the axis reaches the target position. After the InEndVelocity of the instruction is set, only the EndVelocity and EndVelocityDirection can be modified.



18.3.3.3.50 MC_MoveContinuousRelative

This instruction controls the axis to move with a specified relative distance.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_MoveContinuousRelative	Move to a specified relative position ending with the specified velocity		MC_MoveContinuousRelative_Instance_0(Axis :=, Execute :=, ContinuousUpdate :=, Distance :=, Velocity :=, EndVelocity :=, EndVelocityDirection :=, Acceleration :=, Deceleration :=, Jerk :=, BufferMode :=, AdaptEndVelToAvoidOvershoot :=, InEndVelocity =>, DistanceTravelled => , Busy =>, Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Continuous Update	Continuous update	BOOL	TRUE, FALSE	FALSE	Continuous parameter update effective flag. Prerequisite: The parameter ContinuousUpdate shows the value TRUE and the parameter Active shows the value TRUE. Effective parameters: Distance, Velocity, EndVelocity, EndVelocityDirection. Note: The modified parameter values must differ from the original parameter values by more than 0.01.
Distance	Distance	LREAL	Positive number, negative number, or 0	0	Relative distance of the movement.
Velocity	Velocity	LREAL	Positive number	0	Maximum velocity.
EndVelocity	End velocity	LREAL	Positive number or 0	0	End velocity.
EndVelocity Direction	End velocity direction	18.3.3.2.2.9 MC_DIRECTION: Motion Direction Enumeration	1: DIR_POSITIVE 2: DIR_NEGATIVE 3: DIR_CURRENT	3	1: DIR_POSITIVE: Positive direction. 2: DIR_NEGATIVE: Negative direction. 3: DIR_CURRENT: Current direction.
Acceleration	Acceleration	LREAL	Positive number	0	Acceleration of the acceleration phase.
Deceleration	Deceleration	LREAL	Positive number	0	Deceleration of the deceleration phase.
Jerk	Jerk	LREAL	Positive number or 0	0	Slope of acceleration or deceleration.
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting 1: mcBuffered	0	0: mcAborting (Aborting). 1: mcBuffered (Buffered).

Input Variable	Name	Data Type	Valid Range	Default Value	Description
AdaptEndVelToAvoidOvershoot	End velocity adaptation flag	BOOL	TRUE, FALSE	FALSE	TRUE: Adapt the end velocity to avoid overshoot. FALSE: No adaptation.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
InEndVelocity	End velocity reached	BOOL	TRUE, FALSE	FALSE	TRUE when the end velocity is reached.
DistanceTraveled	Position reached	BOOL	TRUE, FALSE	FALSE	TRUE when the expected travel distance is reached.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is being executed.
Active	Controlling	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled by the current instruction.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity	Velocity	Any velocity that exceeds the motion parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	• When the MotionControl library version is 2.1.0.0 or earlier, the velocity equal to 0 will be automatically corrected to 1e-5. • When the MotionControl library version is 2.2.0.0 or later, the velocity equal to 0 will not be corrected.

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Acceleration	Acceleration	Any acceleration that exceeds the motion parameter limit will be automatically corrected to the acceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration less than 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration equal to 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration equal to 0 will be automatically corrected to 10 times the velocity.
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the deceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration less than 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration equal to 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration equal to 0 will be automatically corrected to 10 times the velocity.
Jerk	Jerk	If the motion profile is set to S-curve, any jerk that exceeds the software limit will be automatically corrected to the jerk limit.	If the motion profile is set to S-curve, any jerk less than 0 will be automatically corrected to 10% of the jerk limit.	If the motion profile is set to S-curve, the jerk equal to 0 will be automatically corrected to 10% of the jerk limit.
EndVelocity	End velocity	Any velocity that exceeds the motion parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	No correction.

- **Function**

  Bus-controlled axis:

  ◦ This instruction controls the axis to move to a relative position ending with the specified end velocity.

  ◦ The setpoint of the input parameter Distance specifies the relative position where the axis velocity reaches the specified end velocity. The setpoint of the input parameter Velocity specifies the maximum

velocity during the planning process. The setpoint of the input parameter EndVelocity specifies the end velocity of the axis.

- When the instruction is triggered on a rising edge of the input parameter Execute, the input parameters are checked for validity. If any of invalid parameters is detected, it will be automatically corrected by the function block without a fault message; any buffered instructions after this instruction will be cleared.
- The setpoint of the input parameter EndVelocityDirection specifies the direction of the EndVelocity.
- In Linear Mode, the positive or negative setpoint of the input parameter Distance determines the direction of the Velocity. In rotary mode, when the setpoint of the input parameter Distance exceeds one cycle, multi-turn positioning is performed, and the direction of the Velocity is determined by the positive or negative value of the Distance.
- When the input parameter ContinuousUpdate is set to TRUE, modified parameters Distance, Velocity, EndVelocity and EndVelocityDirection will take effect only when the instruction is in the Active state. Parameter changes greater than 0.01 are considered updates. Modified BufferMode cannot take effect during this period, and the buffer mode at the time of the instruction trigger is retained.
- The axis type must be control axis.
- When the current axis is not in CSP mode, the instruction internally switches the control mode.
- If the software limit is hit during the instruction execution, the parameter Error is set with the axis state being set to ErrorStop.
- During the deceleration due to limits, do not execute the MC_Reset instruction to clear fault messages.
- During the execution of this instruction, modifying the BufferMode and re-executing the instruction will cause the buffer to be cleared with a fault message.
- During the execution of this instruction, the MC_MoveSuperImposed instruction can be used.
- When the axis is enabled after PLC cold reset and the MC_Home instruction is executed using No. 35 homing method, if this instruction is executed with DIR_CURRENT being set, the motion direction of this instruction will be affected.
- When this instruction is executed with the ContinuousUpdate being set to TRUE along with the MC_SuperImposed instruction, the MC_SuperImposed instruction will be aborted if any parameters are modified.

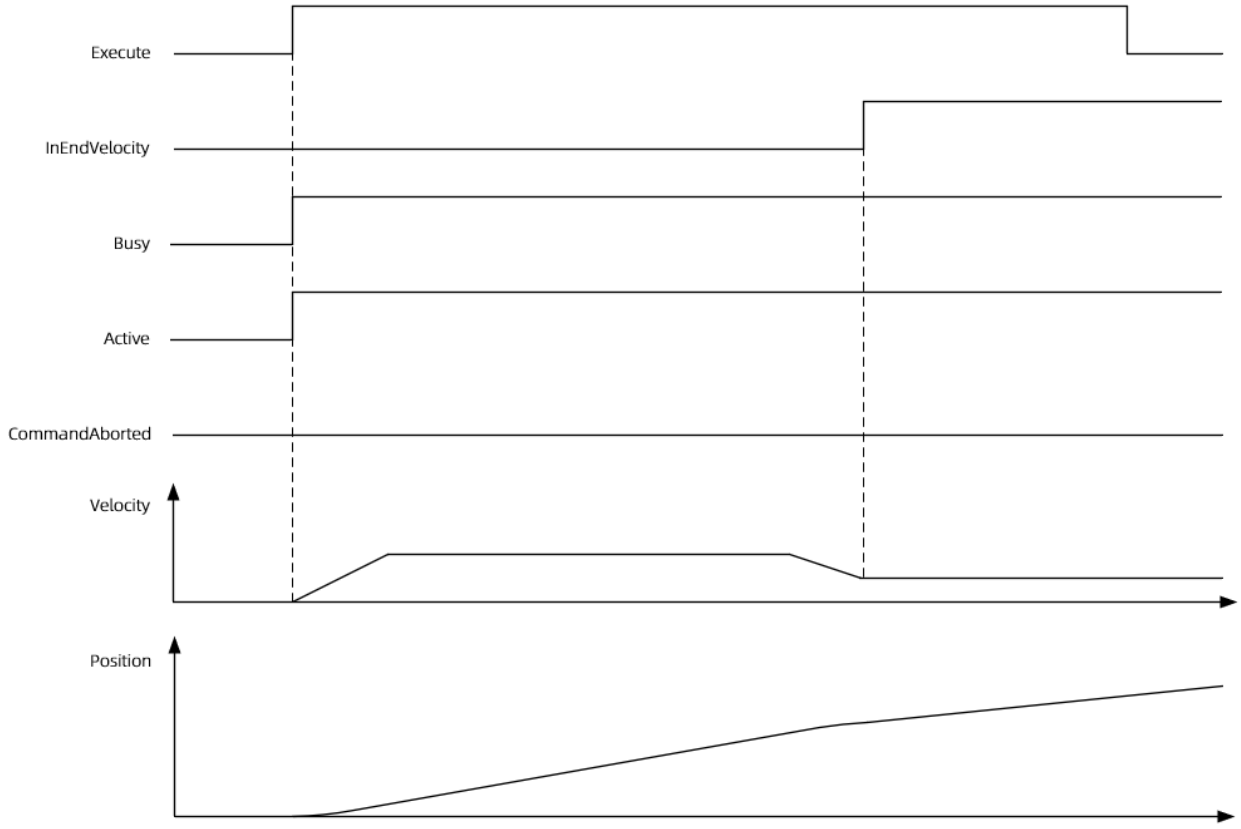
Local pulse axis:

- This instruction can be used for local pulse axes.
- When the local pulse axis is used with Y0 or Y1 single-ended output, it is not recommended to use this instruction. ① No pulse output; ② No position feedback.

- **Timing diagram**

  The input parameter Execute must be triggered on a rising edge. When the output parameter InEndVelocity is set to TRUE, it indicates that the instruction has been executed normally. When the output parameter Busy is set to TRUE, it indicates that the current function block is being executed. When the output parameter Active is set to TRUE, it indicates that the axis in current function block is being controlled.

  After this instruction is called, the output parameter Busy is set to TRUE on a rising edge of Execute. The output parameter Active shows the value TRUE in the same cycle. When the end velocity is reached, the output parameter InEndVelocity is set to TRUE.



- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.51 MC_CheckAxisCommunication

This instruction checks the current communication state of the drive with the axis.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_CheckAxisCommunication	Check axis communication	MC_CheckAxisCommunication_Instance_0 MC_CheckAxisCommunication Axis Enable Valid Error ErrorID Operational ComState	MC_CheckAxisCommunication_Instance_0(Axis :=, Enable :=, Valid =>, Error =>, ErrorID =>, Operational =>, ComState =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description

Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: The function block is enabled. FALSE: Disabled.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Valid output	BOOL	TRUE, FALSE	FALSE	TRUE when there is a valid output from the function block.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.
Operational	Normal communication	BOOL	TRUE, FALSE	FALSE	TRUE: The communication is normal (code 100) and axis operation is supported. FALSE: The communication is abnormal and axis operation is not supported.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ComState	Communication state	WORD	-	-	65535: Initial state. 0: Unknown. 1: Initialized. 2: Pre-operational. 3: Booting. 4: Safe operating. 20: Active communication with the axis position system not restored. 90: Absolute encoder position system recovering. 95: Incremental encoder position system recovering. 100: Active communication with the axis position system restored. 1017: Initialization failure. 1018: Pre-operational fault 1019: Booting fault. 1020: Safe operating fault. 1024: Operational fault

- **Function**

  Bus-controlled axis:

  - This instruction checks the current communication state of the drive with the axis.
  - The output parameters Operational and ComState output the axis communication state and related information.
  - This instruction cannot be executed for virtual axes. The type of the axis being operated must be control axis.

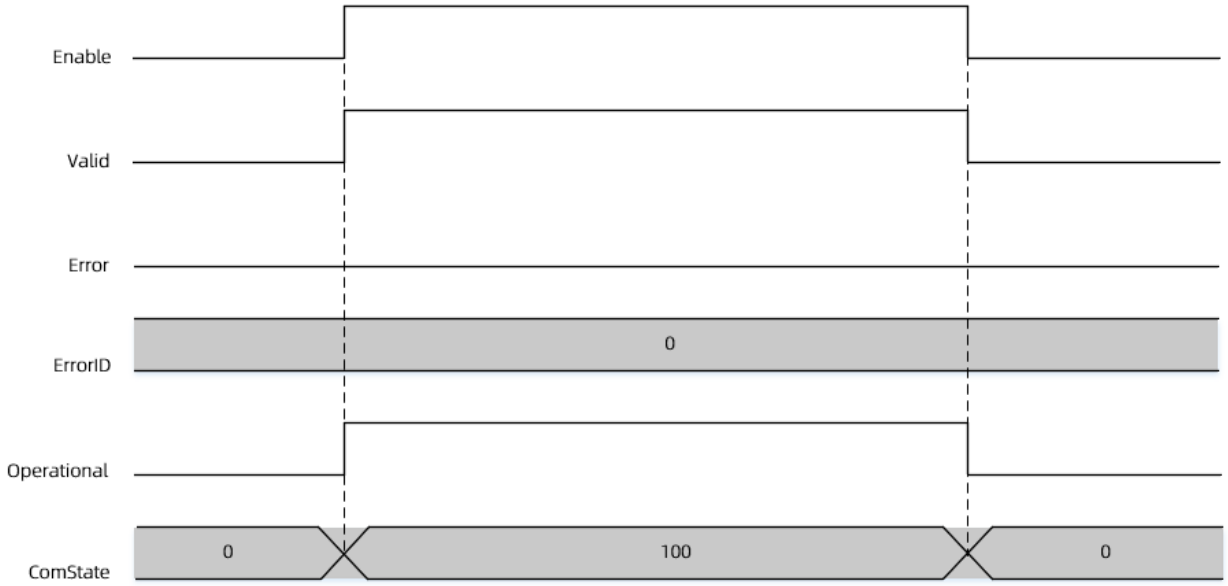
  Local pulse axis:

  - This instruction cannot be used for local pulse axes.

- **Timing diagram**

  The parameter Enable of the function block indicates whether to check the current communication state of the drive, and the parameter Valid indicates valid outputs.

  When the input parameter Enable is set to TRUE, the current instruction is executed. When there are no faults, the output parameter Valid shows the value TRUE, and the axis communication state is checked.



- **Re-execution of motion control instructions**

  This instruction cannot be re-executed.

- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.52 MC_Inch

This instruction is used to inch an axis step by step.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_Inch	Axis inching	MC_Inch_Instance_0 MC_Inch Axis InchForward InchBackward Distance Velocity Acceleration Deceleration Jerk Busy CommandAborted Error ErrorID	MC_Inch_Instance_0(Axis :=, InchForward :=, InchBackword :=, Distance :=, Velocity :=, Acceleretion :=, Deceleration :=, Jerk :=, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
InchForward	Inch forward	BOOL	TRUE, FALSE	FALSE	TRUE: The axis starts moving forward. FALSE: The axis stops moving forward.
InchBackward	Inch backward	BOOL	TRUE, FALSE	FALSE	TRUE: The axis starts moving backward. FALSE: The axis stops moving backward.
Distance	Travel distance	LREAL	Positive number, negative number, or 0	0	Distance for axis inching.
Velocity	Target velocity	LREAL	Positive number	0	Target velocity.
Acceleration	Acceleration	LREAL	Positive number	0	Acceleration of the acceleration phase.
Deceleration	Deceleration	LREAL	Positive number	0	Deceleration of the deceleration phase.
Jerk	Jerk	LREAL	Positive number	0	Slope of acceleration or deceleration.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is being executed.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity	Velocity	Any velocity that exceeds the commissioning parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	• When the MotionControl library version is 2.1.0.0 or earlier, the velocity equal to 0 will be automatically corrected to 1e-5. • When the MotionControl library version is 2.2.0.0 or later, the velocity equal to 0 will not be corrected.
Acceleration	Acceleration	Any acceleration that exceeds the commissioning parameter limit will be automatically corrected to the acceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration less than 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration equal to 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration equal to 0 will be automatically corrected to 10 times the velocity.

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Deceleration	Deceleration	Any deceleration that exceeds the commissioning parameter limit will be automatically corrected to the deceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration less than 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration equal to 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration equal to 0 will be automatically corrected to 10 times the velocity.
Jerk	Jerk	If the motion profile is set to S-curve, any jerk that exceeds the software limit will be automatically corrected to the jerk limit.	If the motion profile is set to S-curve, any jerk less than 0 will be automatically corrected to 10% of the jerk limit.	If the motion profile is set to S-curve, the jerk equal to 0 will be automatically corrected to 10% of the jerk limit.

- **Function**

  Bus-controlled axis:

  - This instruction is used to achieve single-step axis control.
  - When the input parameter InchForward shows the value TRUE, the axis moves forward at the specified velocity to the specified distance of Distance. This instruction cannot be re-triggered unless the InchForward is set to FALSE and then back to TRUE. If the InchForward is set to FALSE before the axis reaches the specified position, the axis will immediately decelerate to 0 and the output parameter Busy will be set to FALSE.
  - When the input parameter InchBackward is set to TRUE, the axis moves backward with other conditions being the same as for forward motion.
  - If the input parameters InchForward and InchBackward are set to TRUE, the axis stops moving.
  - When this function block is triggered with the axis state being AXIS_STOPPING or AXIS_HOMING, the MC_Inch instruction is aborted; the axis state is DISCRETE_MOTION when the MC_Inch instruction is executed.
  - If a fault occurs during execution and the current velocity reference is not 0, the axis will decelerate to stop under a T-curve planning, using the maximum deceleration in the software.
  - The axis type must be control axis.
  - When the current axis is not in CSP or CSV mode, the instruction internally switches the control mode.
  - This instruction can abort or be aborted by another motion control instruction.

◦ During the execution of this instruction, the MC_MoveSuperImposed instruction cannot be used; once the MC_MoveSuperImposed instruction is triggered, CommandAborted is set for the MC_MoveSuperImposed instruction.

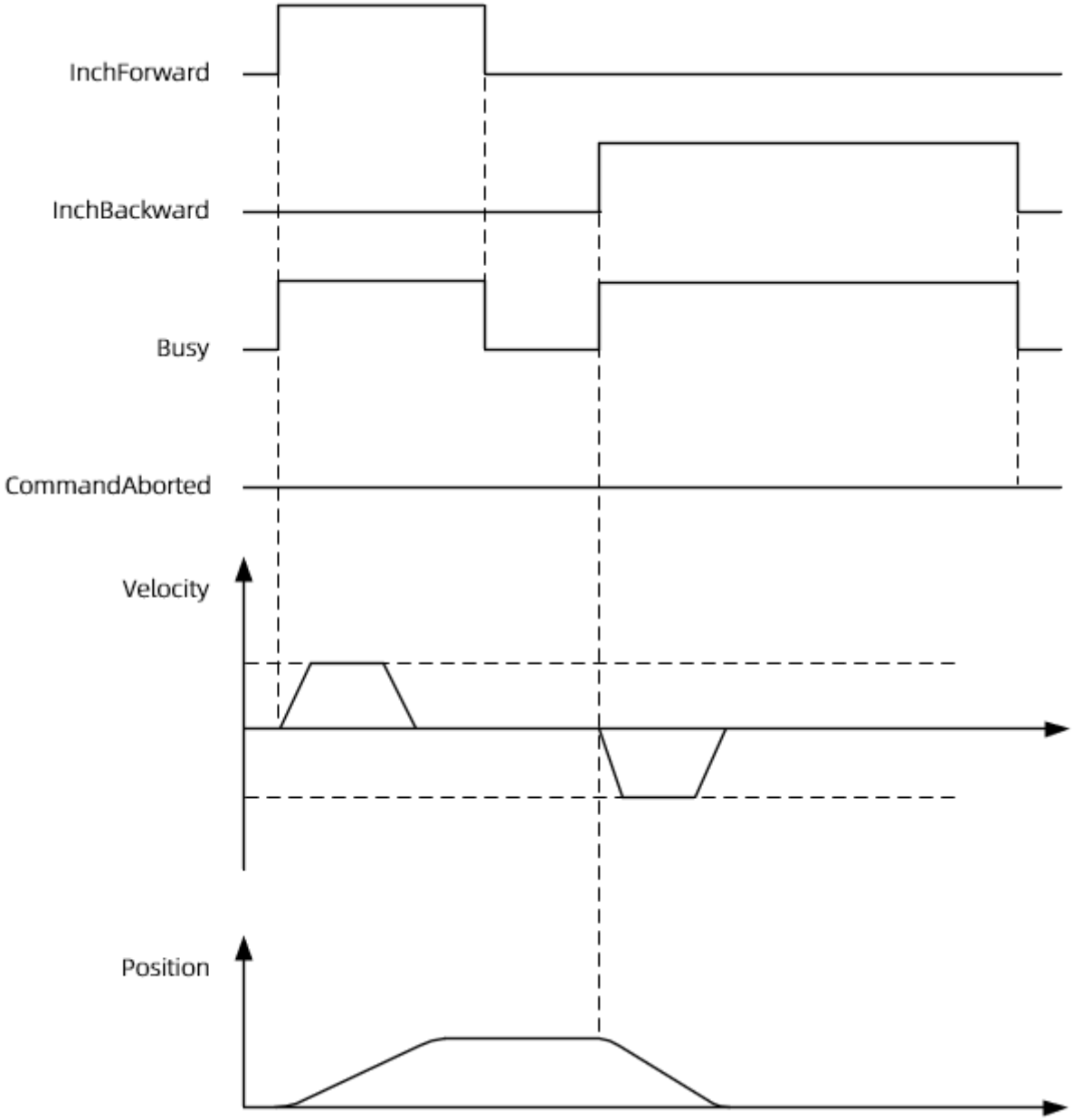
Local pulse axis:

◦ This instruction can be used for local pulse axes.

- **Timing diagram**

  The InchForward and InchBackward of the function block must support both TRUE and FALSE conditions.

  The parameter Busy of the function block indicates that the current function block is being executed. The parameter CommandAborted of the function block indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.



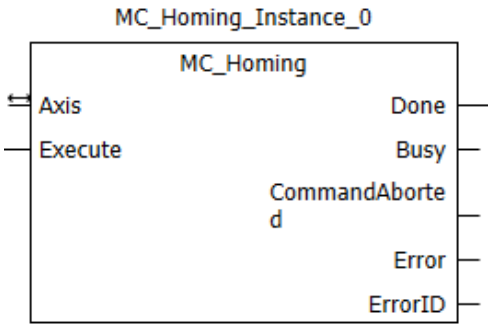
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.53 MC_Homing

This instruction enables the PLC to be homed.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_Homing	Homing		MC_Homing _Instance_0(Axis :=, Execute :=, Done =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Acceleration	Acceleration	Any acceleration that exceeds the motion parameter limit will be automatically corrected to the acceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration less than 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration equal to 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration equal to 0 will be automatically corrected to 10 times the velocity.

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the deceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration less than 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration equal to 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration equal to 0 will be automatically corrected to 10 times the velocity.

- **Function**

  Bus-controlled axis:

  - Homing starts on a rising edge of Execute. The MC_Homing instruction can be executed only when the axis is in the Standstill state. If the current axis state is STOPPING or HOMING, the instruction will be directly aborted with the output parameter CommandAborted being set. If the current axis state is Disabled, ErrorStop, Discrete Motion, Continuous Motion, or Synchronized Motion, the function block will stop running with a fault message. If the current control mode is not CSP, the function block will stop running with a fault message.
  - The MC_Homing instruction can only be aborted by the MC_Stop or MC_ImmediateStop instruction during execution. Once being aborted, the output parameter CommandAborted is set.
  - You can specify axis parameters for homing in the software and select any of the following 10 operations to define home:

    1. Proximity reverse turn/home proximity input OFF
    2. Proximity reverse turn/home proximity input ON
    3. Home proximity input OFF
    4. Home proximity input OFF
    5. Limit input OFF
    6. Proximity reverse turn/home input mask distance
    7. Limit inputs only
    8. Proximity reverse turn/holding time

9. No home proximity input/holding home input

10. Zero position preset

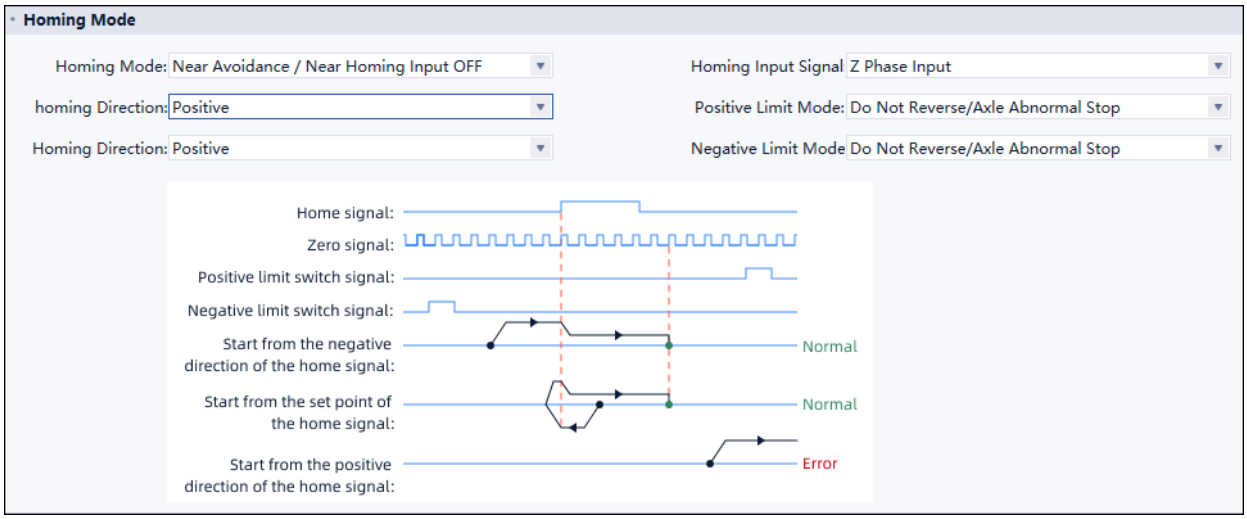
- Before calling the MC_Homing instruction, configure relevant PDOs in the software according to different homing operation modes:
  - For homing operation modes: 7 Limit inputs only; 8 Proximity reverse turn/holding time; 10 Zero position preset

    Configure PDO 60FD.
  - For other homing operation modes

    Configure PDOs 60B8, 60B9, 60BA, and 60FD.
- Operation selection at "Positive limit input" and "Negative limit input" during homing:

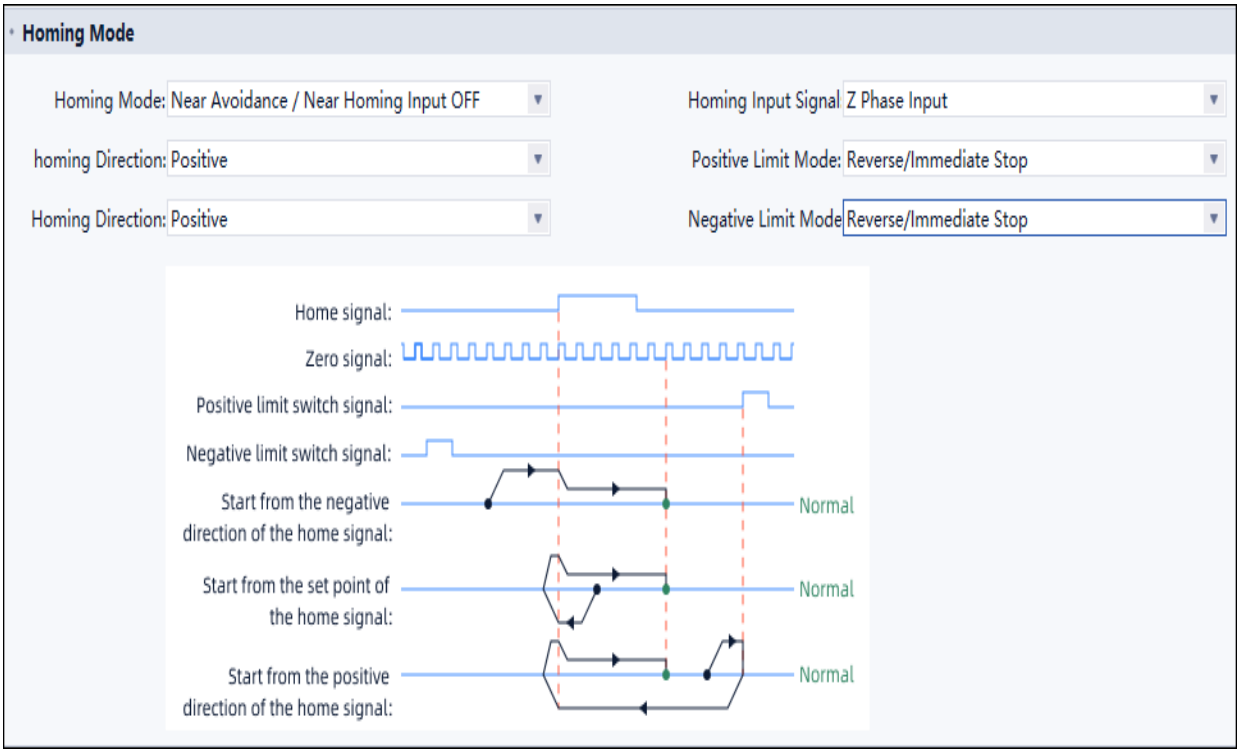
  Select the operation when the axis reaches a limit input in the operating direction during homing: do not reverse the axis, create an error, and stop the axis, or reverse the axis and continue with homing. To reverse the axis, also select the stopping method.
  - No reverse turn/fault stop

    During homing, if a limit is detected, the axis will experience a fault stop with a fault message.



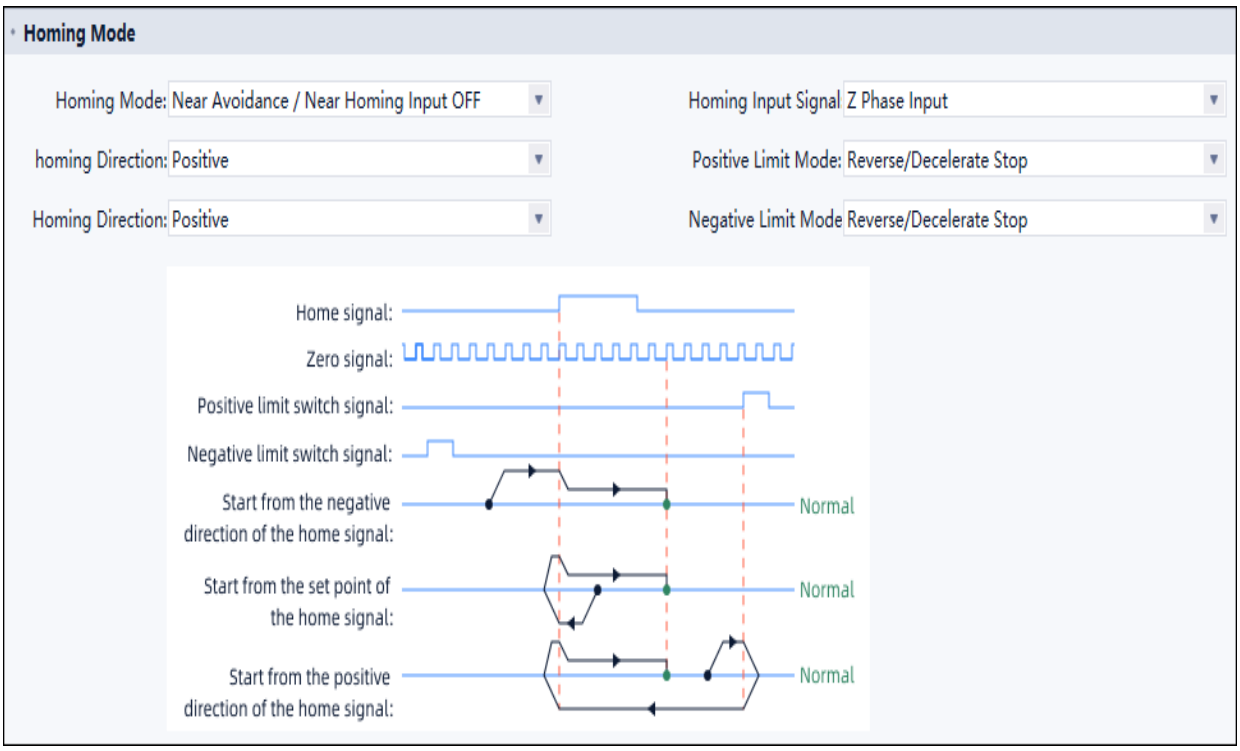
  - Reverse turn/immediate stop

    During homing, if a limit is detected, the axis will stop immediately and reverse its direction.



- Reverse turn/Deceleration stop

  During homing, if a limit is detected, the axis will also reversely move at the maximum deceleration.

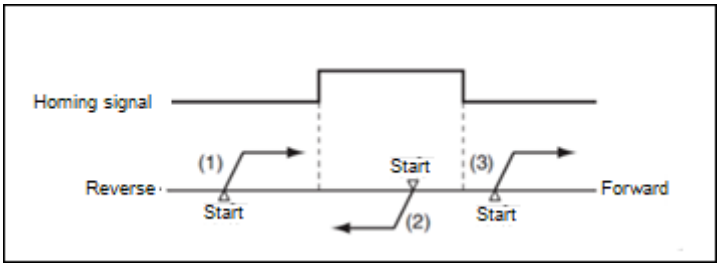


- **Homing start direction**

  It refers to the direction in which the axis starts moving when homing is started.

  If homing starts while the home proximity input signal is ON in a homing operation mode that includes reversal operation, the axis starts moving in the direction opposite to the home input detection direction.

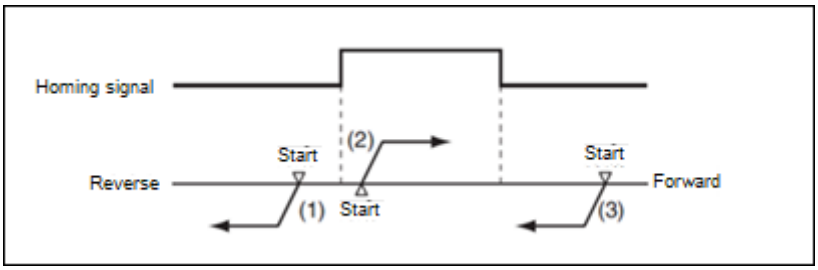
- Homing start direction: positive; Home input detection direction: positive



(1), (3): The home proximity input signal is OFF, so the axis starts moving in the homing start direction.

(2): The home proximity input signal is ON, so the axis starts moving in the direction opposite to the home input detection direction.

- Homing start direction: negative; Home input detection direction: negative



(1), (3): The home proximity input signal is OFF, so the axis starts moving in the homing start direction.

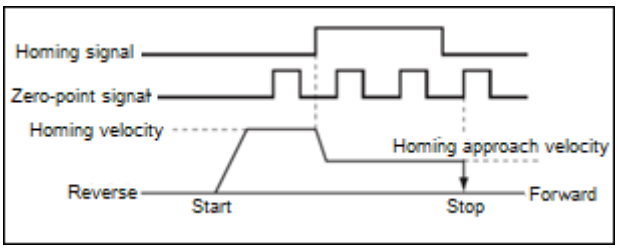
(2): The home proximity input signal is ON, so the axis starts moving in the direction opposite to the home input detection direction.

◦ **Home input detection direction**

It specifies the direction of the homing approach velocity.

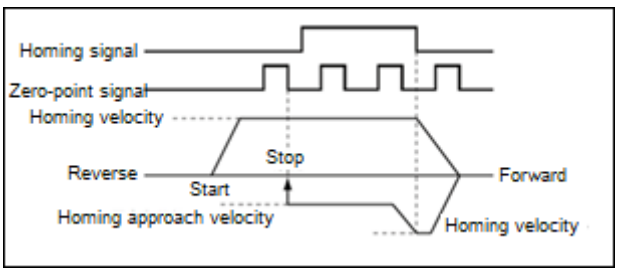
- The following figure shows the operation when the home input detection direction is positive direction.

Operation example: Proximity reverse turn/home proximity input OFF



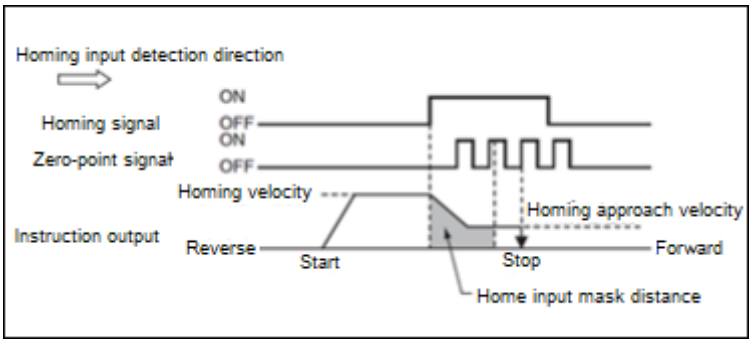
- The following figure shows the operation when the home input detection direction is negative direction.

Operation example: Proximity reverse turn/home proximity input OFF



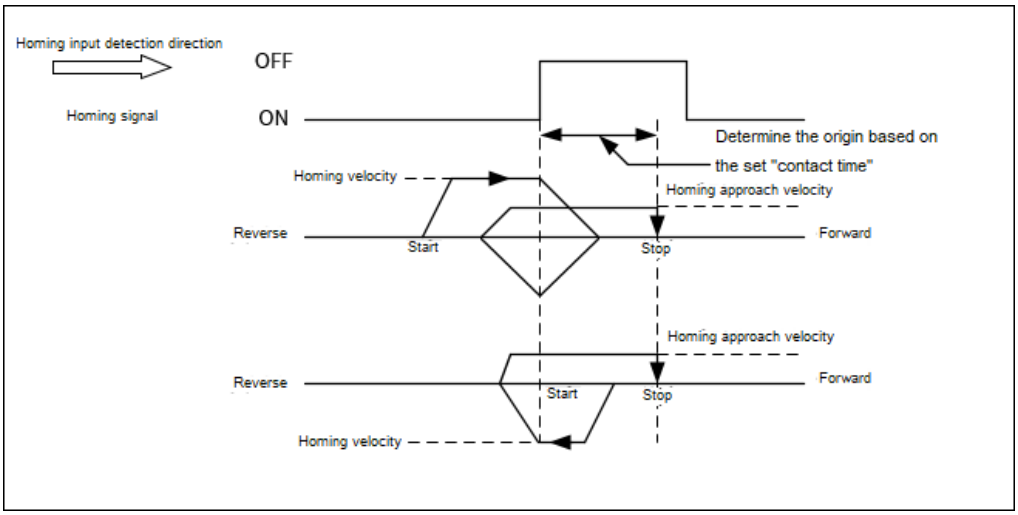
◦ **Home input mask distance**

When the home proximity input signal turns ON, mechanical home is defined on the first rising edge of the home input signal after the axis moves the distance set by "Home input mask distance". This parameter is valid when the homing operation mode is specified as "Proximity reverse turn/home input mask distance."



- **Homing holding time**

  When the home proximity input signal turns ON, mechanical home is defined after the axis move for the time set by "Homing holding time". This parameter is valid when the homing operation mode is specified as "Proximity reverse turn/holding time."



- **Homing compensation value**

  It refers to the homing compensation value that is applied after the home input signal is detected. After the mechanical home is defined in a mechanical system, it is often difficult to adjust the workpiece position. Sometimes fine tuning is required. In this case, the offset of the initially defined mechanical home can be set as compensation value.

- **Home position offset**

  It is used to preset the actual position after homing is completed.

- **Homing velocity**

  It refers to the high velocity during homing.

- **Homing acceleration**

  It refers to the acceleration during homing.

- **Homing jerk**

  It refers to the jerk during homing.

- **Homing deceleration**

It refers to the deceleration during homing.

◦ **Homing approach velocity**

It refers to the proximity velocity during homing.

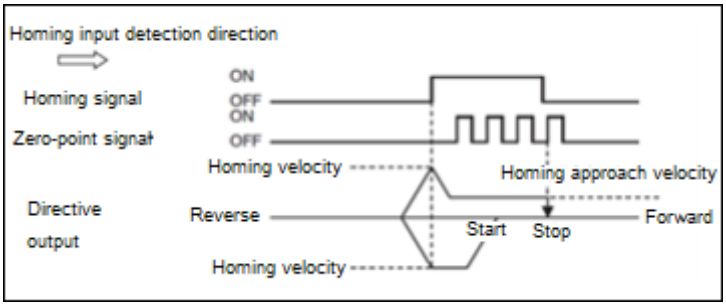
◦ **Homing definition operation**

The following describes the 10 homing operation modes.

1. Proximity reverse turn/home proximity input OFF

a. After the instruction is triggered, the axis starts moving at the homing velocity. When the home proximity input signal turns ON, the axis starts decelerating to the homing approach velocity.

b. After the axis reaches the homing approach velocity, the axis stops on the first rising edge of the home input signal after the home proximity input signal turns OFF. In this way, the mechanical home is defined.

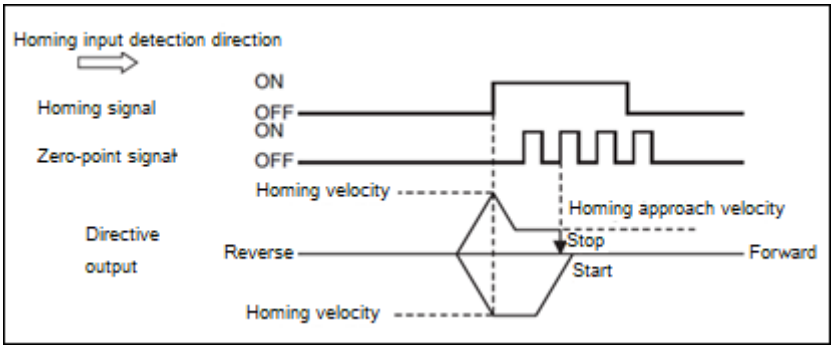


- If you start homing while the home proximity input signal is ON, homing will start at the homing velocity in the direction opposite to the home input detection direction. After the home proximity input signal turns OFF, the axis starts decelerating to 0 at the homing deceleration, and then accelerates at the homing acceleration in reverse direction to the homing velocity. After the home proximity input signal turns ON, the axis starts decelerating to the homing approach velocity.
- After homing is started, the mechanical home is defined when the home input signal turns ON after the home proximity input signal turns ON and then OFF while the velocity is below the homing approach velocity.

2. Proximity reverse turn/home proximity input ON

a. After the instruction is triggered, the axis starts moving at the homing velocity. When the home proximity input signal turns ON, the axis starts decelerating to the homing approach velocity.

b. After the axis reaches the homing approach velocity, the axis stops on the first rising edge of the home input signal. In this way, the mechanical home is defined. After the home proximity input signal turns ON, its status does not affect operation.



- If you start homing while the home proximity input signal is ON, homing will start at the homing velocity in the direction opposite to the home input detection direction. After the home proximity

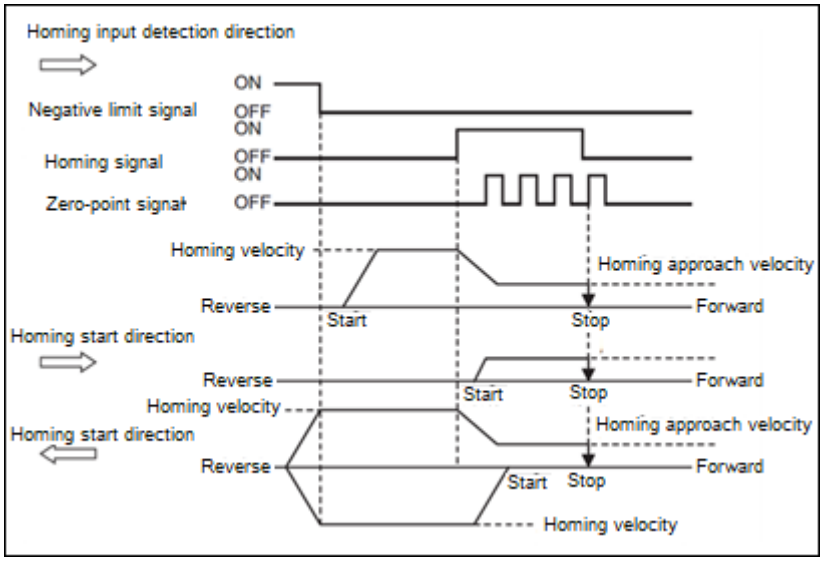
input signal turns OFF, the axis starts decelerating to 0 at the homing deceleration, and then accelerates at the homing acceleration in reverse direction to the homing velocity. After the home proximity input signal turns ON, the axis starts decelerating to the homing approach velocity.

- After homing is started, the mechanical home is defined when the home input signal turns ON after the home proximity input signal turns ON while the velocity is below the homing approach velocity.

3. Home proximity input OFF

a. After the instruction is triggered, the axis starts moving at the homing velocity. When the home proximity input signal turns ON, the axis starts decelerating to the homing approach velocity.

b. After the axis reaches the homing approach velocity, the axis stops on the first rising edge of the home input signal after the home proximity input signal turns OFF. In this way, the mechanical home is defined.



- If you start homing while the home proximity input signal is ON, the axis performs the following operation depending on the setting of the homing start direction.

  Homing start direction same as home input detection direction

  The axis does not perform a reverse turn and homing starts in the home input detection direction at the homing approach velocity.

  Homing start direction different from home input detection direction

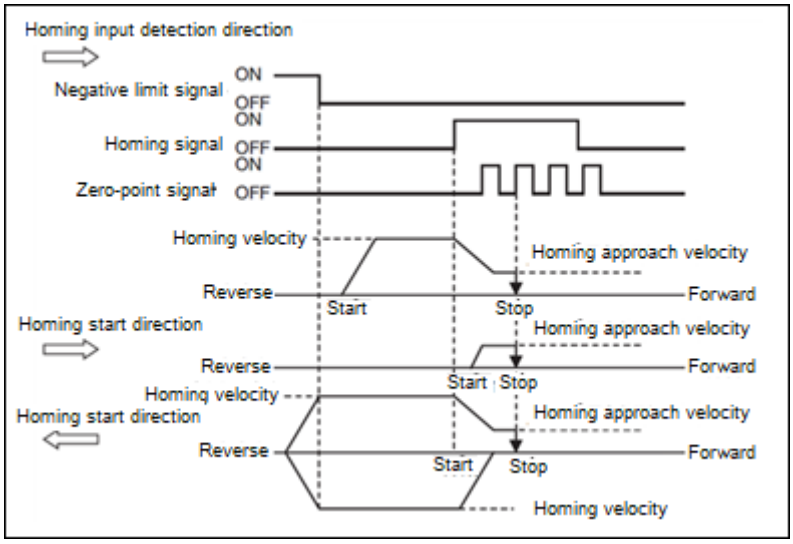
  Homing starts in the homing start direction at the homing velocity, regardless of the status of the home proximity input signal. If you set the operation at the limit input in the homing start direction to "Reverse turn/Immediate stop" or "Reverse turn/Deceleration stop", the axis reverses its direction when limit input is detected, and performs homing in the home input detection direction.

- After homing is started, the mechanical home is defined when the home input signal turns ON after the home proximity input signal turns ON and then OFF while the velocity is below the homing approach velocity.

4. Home proximity input ON

a. After the instruction is triggered, the axis starts moving at the homing velocity. When the home proximity input signal turns ON, the axis starts decelerating to the homing approach velocity.

b. After the axis reaches the homing approach velocity, the axis stops on the first rising edge of the home input signal. In this way, the mechanical home is defined. After the home proximity input signal turns ON, its status does not affect operation.



- If you start homing while the home proximity input signal is ON, the axis performs the following operation depending on the setting of the homing start direction.

  Homing start direction same as home input detection direction

  The axis does not perform a reverse turn and homing starts in the home input detection direction at the homing approach velocity.

  Homing start direction different from home input detection direction

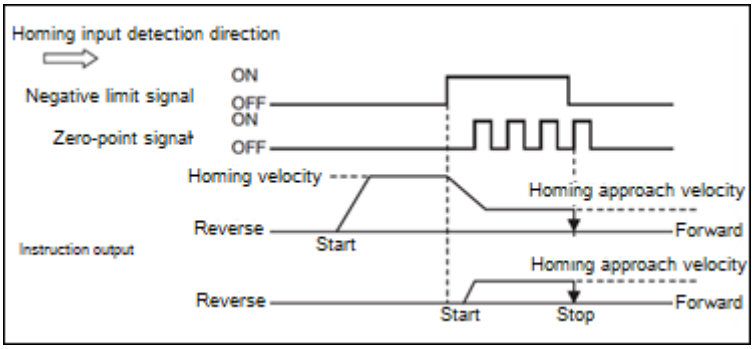
  Homing starts in the homing start direction at the homing velocity, regardless of the status of the home proximity input signal. If you set the operation at the limit input in the homing start direction to "Reverse turn/Immediate stop" or "Reverse turn/Deceleration stop", the axis reverses its direction when limit input is detected, and performs homing in the home input detection direction.

- After homing is started, the mechanical home is defined when the home input signal turns ON after the home proximity input signal turns ON while the velocity is below the homing approach velocity.

5. Limit input OFF

a. After the instruction is triggered, the axis starts moving at the homing velocity. When the limit signal in the direction opposite to the home input detection direction turns ON, the axis starts decelerating to the homing approach velocity.

b. After the axis reaches the homing approach velocity, the axis stops on the first rising edge of the home input signal after the limit signal turns OFF. In this way, the mechanical home is defined.

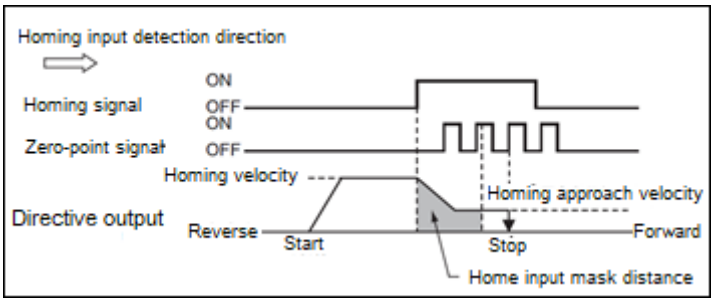


- If you perform homing while the limit input on the opposite side of the home input detection direction is ON, homing starts at the homing approach velocity in the home input detection direction.
- After homing is started, the mechanical home is defined when the home input signal turns ON after the limit signal in the direction opposite to the home input detection direction turns ON and then OFF while the velocity is below the homing approach velocity.

6. Proximity reverse turn/home input mask distance

a. After the instruction is triggered, the axis starts moving at the homing velocity. When the home proximity input signal turns ON, the axis starts decelerating to the homing approach velocity.

b. The axis moves by the home input mask distance after the home proximity input signal turns ON and stops on the first rising edge of the home input signal. In this way, the mechanical home is defined.



- If you start homing while the home proximity input signal is ON, homing will start at the homing velocity in the direction opposite to the home input detection direction. After the home proximity input signal turns OFF, homing will start at the homing approach velocity in the home input detection direction

  .

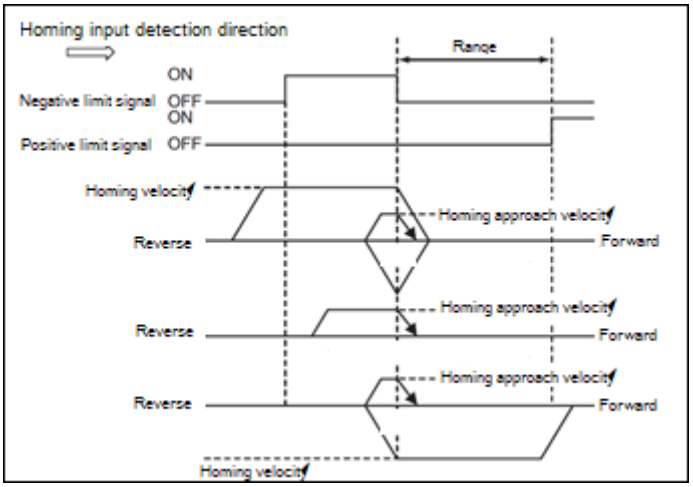
- After homing is started, the mechanical home is defined when the home input signal turns ON after the axis move by the home input mask distance after the home proximity input signal turns ON while the velocity is below the homing approach velocity.

7. Limit inputs only

a. After the instruction is triggered, the axis starts moving at the homing velocity. When the limit signal in the direction opposite to the home input detection direction turns OFF, the axis decelerates to stop.

b. After the axis stops, the axis moves in the other direction at the homing velocity and decelerates to a stop when the limit signal turns ON.

c. After the axis stops, the axis moves in the other direction at the homing approach velocity. The position where the limit signal turns OFF is defined as home and an immediate stop is performed. However, the axis does not return to the mechanical home position.

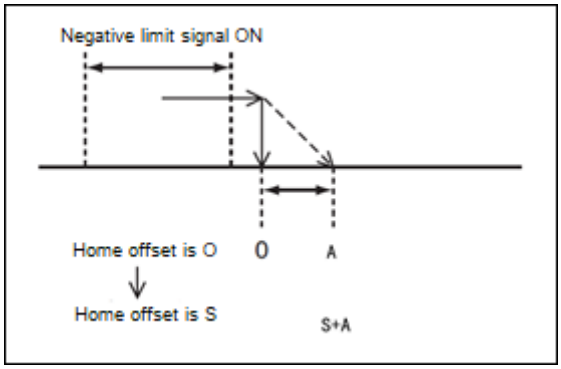


- If you use only the limit signals to perform homing, the point at which the limit signal turns OFF during operation in the home input detection direction is set as the home detection position.
- If you start homing while the limit signal in the home input detection direction is ON, homing will start at the homing velocity in the direction opposite to the home input detection direction. When the limit signal in the direction opposite to the home input detection direction turns ON, the axis decelerates to stop.
- After the limit signal is detected, even if the limit signal is set to OFF before deceleration stop, the mechanical home cannot be defined and deceleration continues. In this case, no fault will occur.
- Home is defined if the limit signal turns OFF before the homing approach velocity is reached and after the axis moves in the other direction or starts moving while the limit signal is ON.

Differences between this mode and other homing operation modes

In this mode, the MC function block detects when the limit signal turns OFF to define home. It differs from a homing operation mode that uses the home input signal as an external latch signal in the following ways.

- The MC function block detects the limit input without an external latch function, such as one provided by a servo drive. MC function block processing is the processing performed in the task cycle. Therefore, the precision of the home definition will depend on the homing approach velocity and the task control cycle.
- The axis does not return to the mechanical home position. If the homing compensation value is 0, processing for homing will end with the axis at a different position (i.e., not at the mechanical home). If the homing compensation value is not 0, homing compensation is performed with the homing compensation value as a relative position from mechanical home in the same way as for other homing operation modes.
- The home position offset is used to change the position of mechanical home. If the stop position is offset from home by distance A, as shown in the following figure, the position after the completion of processing for homing will be S + A if the home position offset is S and the homing compensation value is 0.

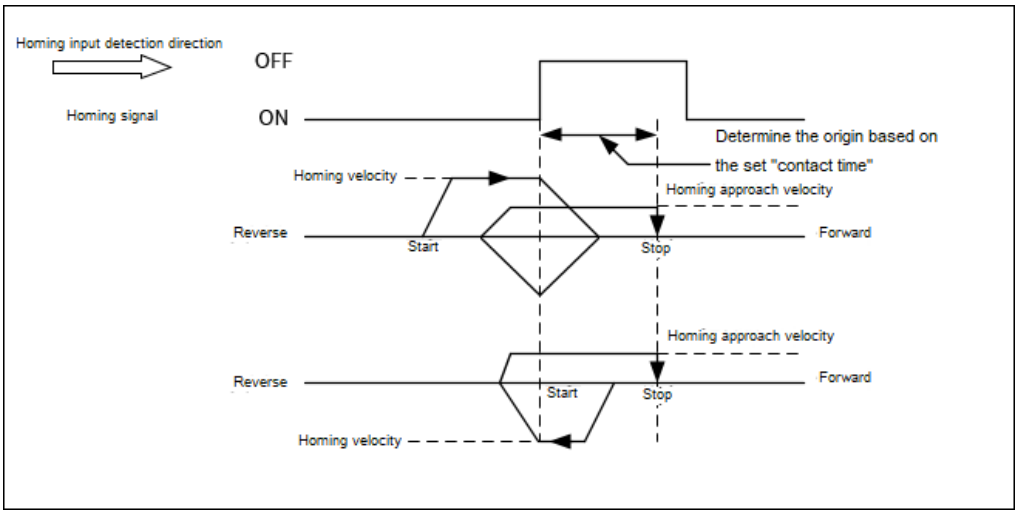


8. Proximity reverse turn/holding time

a. After the instruction is triggered, the axis starts moving at the homing velocity. When the home proximity input signal turns ON, the axis starts decelerating to 0 at the homing deceleration and then accelerates at the homing acceleration in reverse direction to the homing velocity.

b. When the home proximity input signal turns OFF, the axis decelerates to 0 at the homing deceleration and then accelerates at the homing acceleration in reverse direction to the homing approach velocity.

c. When the home proximity input signal turns ON, time monitoring starts and the mechanical home is defined after the defined "Holding time". After the home proximity input signal turns ON, its status does not affect operation.

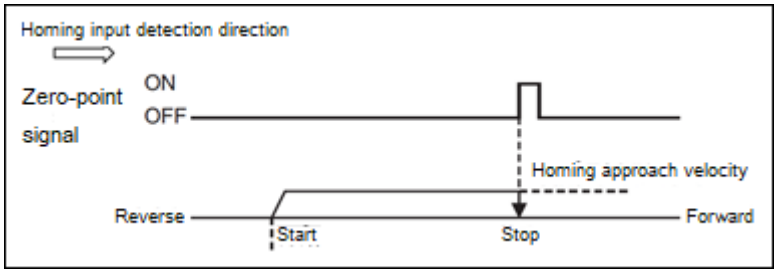


- If you start homing while the home proximity input signal is ON, homing will start at the homing velocity in the direction opposite to the home input detection direction. When the home proximity input signal turns OFF, the axis decelerates to 0 and then accelerates at the homing acceleration in reverse direction to the homing approach velocity. Homing starts when the home proximity input signal turns ON.
- After homing is started, the mechanical home is defined if the holding time elapses after the home proximity input signal turns ON while the velocity is below the homing approach velocity.

9. No home proximity input/holding home input

a. After the instruction is triggered, the axis starts moving at the homing approach velocity.

b. The mechanical home is defined when the home input signal turns ON.

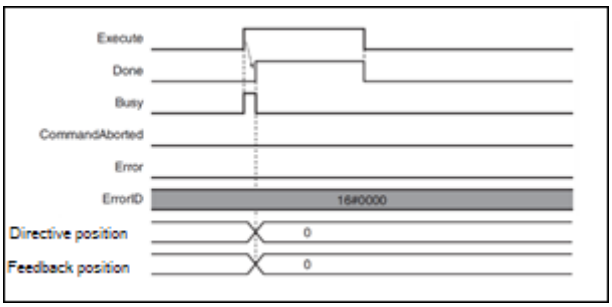


- After homing is started, the mechanical home is defined when the home input signal turns ON before the homing approach velocity is reached.

10. Zero position preset

a. The current position reference and the actual position are set to the home position offset to define the mechanical home. Also, the following error between the current position reference and the actual position is retained.

The operation when the home position offset is set to 0 is shown in the following figure:

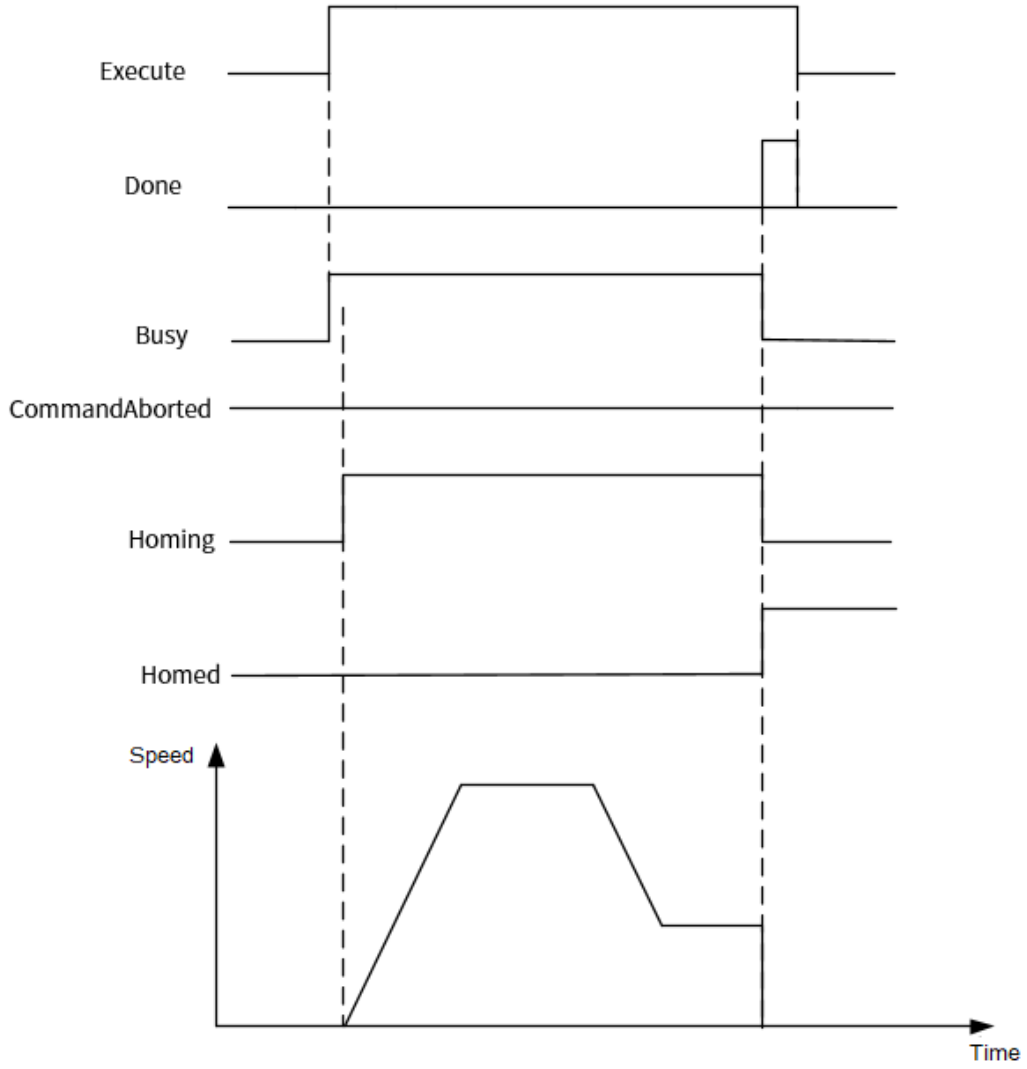


◦ The arrival check function is only applicable to MotionControl library version 2.4.0.0 or later. Based on the "Positioning range" and "Positioning check time" parameters in the "General setting" of the axis configuration page, it checks in real time whether the actual position has entered the specified target range. After the instruction output for the target position is completed, positioning is complete and the Done output is set if the difference between the actual position and the target position is smaller than the "Positioning range". If positioning is not completed within the "Positioning check time", a fault will occur.

Local pulse axis:

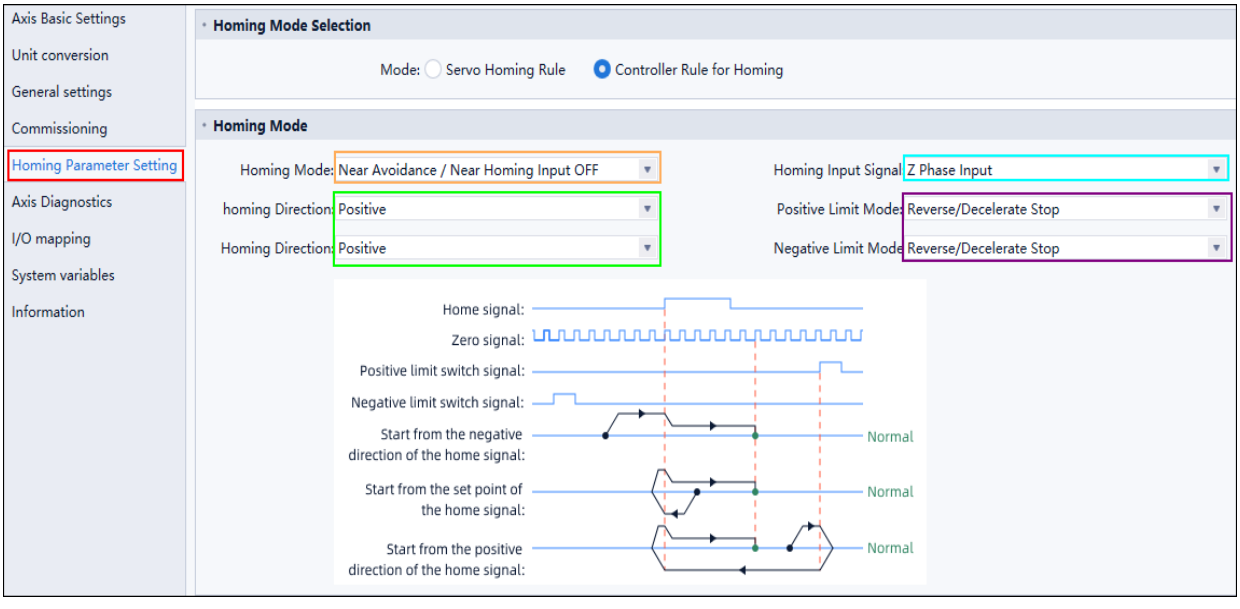
◦ This instruction cannot be used for local pulse axes.

- **Timing diagram**

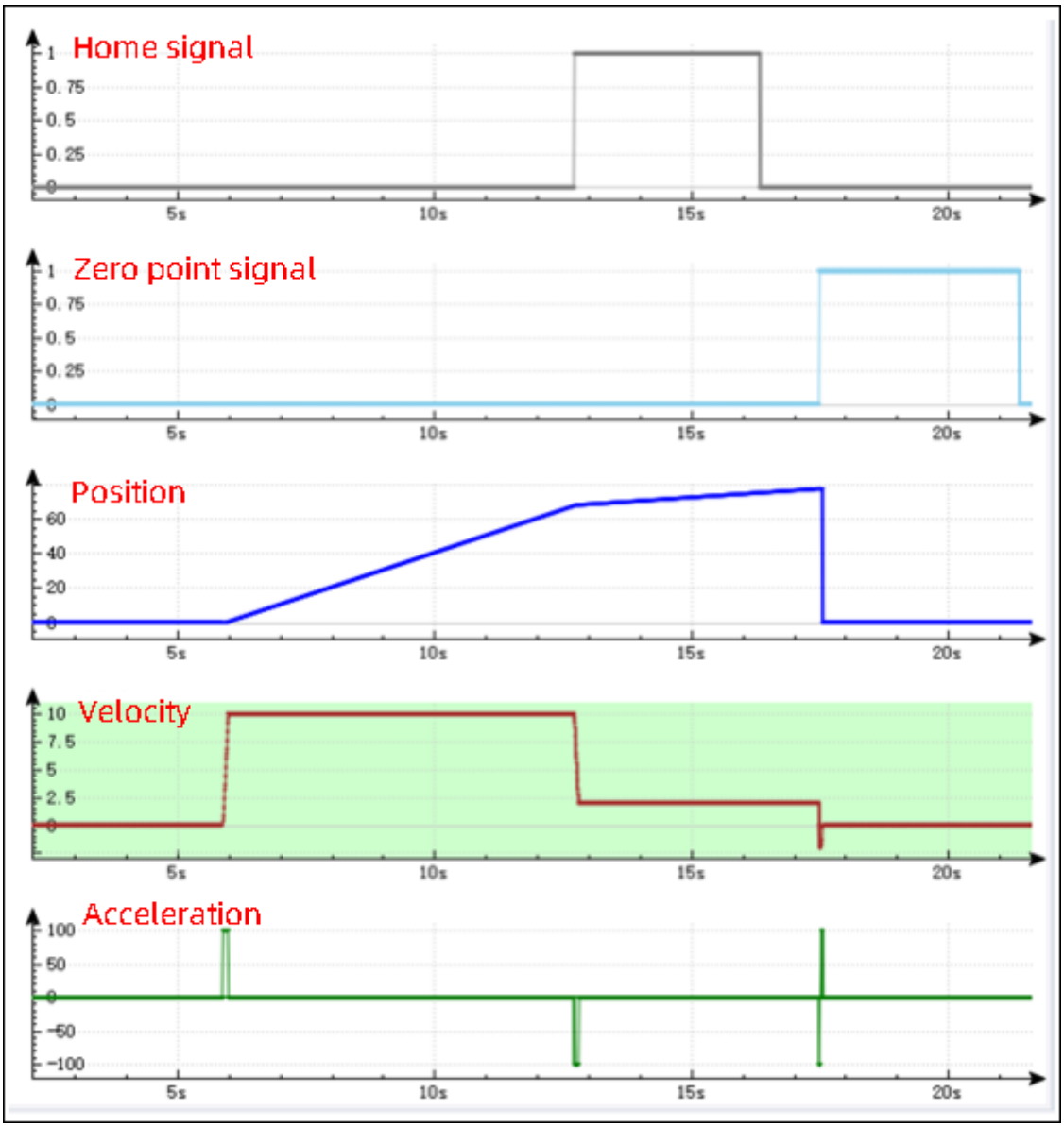


- **Example**

  Open the servo axis settings interface, access the Homing Setting interface, modify the Homing Method to select an external signal that meets the on-site requirements; modify the Homing Start Direction and Home Input Detection Direction to select a homing method that fits the on-site process; modify the Positive Limit Input and Negative Limit Input to "Reverse turn/immediate stop", and modify the Home Input Signal to the signal source for on-site homing. The motion parameters are the required motion parameters.



Call the MC_Homing instruction and directly trigger the parameter Execute of the instruction. The axis will start moving according to the trajectory set in the software.



- **Precautions**
  - Unlike the MC_Home, the number of revolutions of a rotatory axis will not be reset when the MC_Homing is executed.
  - To maintain absolute precision, when the instruction detects the home input signal at the low velocity, it moves back a certain distance to search for the home position. Therefore, after homing is completed and a motion control instruction is called, if the motion direction is specified as Current, the actual motion direction may be inconsistent with the homing parameter "Home input detection direction".
  - If a slave device bound to the servo axis (e.g., a servo) supports Z-phase signal and probe functions, it can support all homing operation modes. If a slave device bound (e.g., a module) does not support Z-phase signal but supports probe functions, set the home input signal to be probe signal to complete homing.
  - Currently, the positive and negative limit terminals are, by default, assigned to the positive and negative limit signals of the servo end, and binding general-purpose terminals as limit signals is not supported. Therefore, if you select the deceleration mode for "Positive/negative limit input", the axis will stop immediately.

- Do not check the reverse configuration when using the MC_Homing instruction; otherwise, the servo end will give an alarm if the positive/negative limit is triggered during execution.
- When the MotionControl library version is 2.2.0.0 or later, the PLC homing function is supported for local pulse axes.

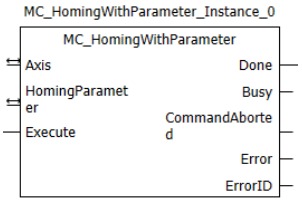
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.54 MC_HomingWithParameter

This instruction sets the homing parameters and controls the motor to define the home.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_HomingWithParameter	Homing with parameters		MC_HomingWithParameter_Instance_0(Axis :=, HomingParameter :=, Execute :=, Done =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.
HomingParameter	Homing parameters	MC_HOMING_PARAMETER_REF	-	-	Specify the homing parameters.

  Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction.

  Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity	Velocity	Any velocity that exceeds the motion parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	The velocity equal to 0 will be automatically corrected to 1e-5.

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Acceleration	Acceleration	Any acceleration that exceeds the motion parameter limit will be automatically corrected to the acceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration less than 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration equal to 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration equal to 0 will be automatically corrected to 10 times the velocity.
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the deceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration less than 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration equal to 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration equal to 0 will be automatically corrected to 10 times the velocity.

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Jerk	Jerk	If the motion profile is set to S-curve, any jerk that exceeds the software limit will be automatically corrected to the jerk limit.	If the motion profile is set to S-curve, any jerk less than 0 will be automatically corrected to 10% of the jerk limit.	If the motion profile is set to S-curve, the jerk equal to 0 will be automatically corrected to 10% of the jerk limit.
ApproachVel	Homing approach velocity	Any velocity that exceeds the motion parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	Any velocity less than 0 will be automatically corrected to 1e-5.
CompensationVelocity	Compensation velocity	Any velocity that exceeds the motion parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	Any velocity less than 0 will be automatically corrected to 1e-5.

- **Function**

  Bus-controlled axis:

  - Homing starts on a rising edge of Execute.
  - Specify the homing parameters through the in-out variable HomingParameter. The axis parameters are not changed when the MC_HomingWithParameter instruction is executed.
  - The only difference between the MC_HomingWithParameter instruction and the MC_Home instruction is how to specify the parameters. For this instruction, the parameters are specified using the in-out parameter HomingParameter. For the MC_Homing instruction, the homing parameters are specified in the software.
  - When the HomeTimeOut is set to 0, there is no limit on the homing timeout period.
  - Description of the HomingParameter structure

Parameter	Definition	Data Type	Valid Range	Description
HomeMode	Homing mode	18.3.3.2.2.13 MC_HOMING_MODE: Homing Mode Enumeration	1: PassHomeSwHomeSwTurn 2: InHomeSwHomeSwTurn 3: PassHomeSwHome 4: InHomeSw 5: PassLimitSw 6: HomeSwMaskDistanceHomeSwTurn 7: OnlyLimit 8: HomeSwHoldTimeHomeSwTurn 9: ZeroSw 10: ZeroPositionPreset	1: Proximity reverse turn/home proximity input OFF 2: Proximity reverse turn/home proximity input ON 3: Home proximity input OFF 4: Home proximity input ON 5: Limit input OFF 6: Proximity reverse turn/home input mask distance 7: Limit inputs only 8: Proximity reverse turn/holding time 9: No home proximity input/holding home input 10: Zero position preset
ZeroSignal	Zero signal	18.3.3.2.2.16 MC_ZERO_SIGNAL: Zero Signal Enumeration	0: ZPhase 1: ExternalSignal	0: Z-phase signal input. 1: External signal input.
StartDirection	Homing start direction	18.3.3.2.2.9 MC_DIRECTION: Motion Direction Enumeration	1: DIR_POSITIVE 2: DIR_NEGATIVE	1: Positive direction. 2: Negative direction.
EndDirection	Homing end direction	18.3.3.2.2.9 MC_DIRECTION: Motion Direction Enumeration	1: DIR_POSITIVE 2: DIR_NEGATIVE	1: Positive direction. 2: Negative direction.
PosLimitMode	Positive limit mode	18.3.3.2.2.14 MC_LIMIT_REVERSE_MODE: Limit Reverse Mode Enumeration	0: Errorstop 1: RevImmediateStop 2: RevDeceleration	0: No reverse turn/fault stop. 1: Reverse turn/immediate stop. 2: Reverse turn/Deceleration stop.
NegaLimitMode	Negative limit mode	18.3.3.2.2.14 MC_LIMIT_REVERSE_MODE: Limit Reverse Mode Enumeration	0: Errorstop 1: RevImmediateStop 2: RevDeceleration	0: No reverse turn/fault stop. 1: Reverse turn/immediate stop. 2: Reverse turn/Deceleration stop.

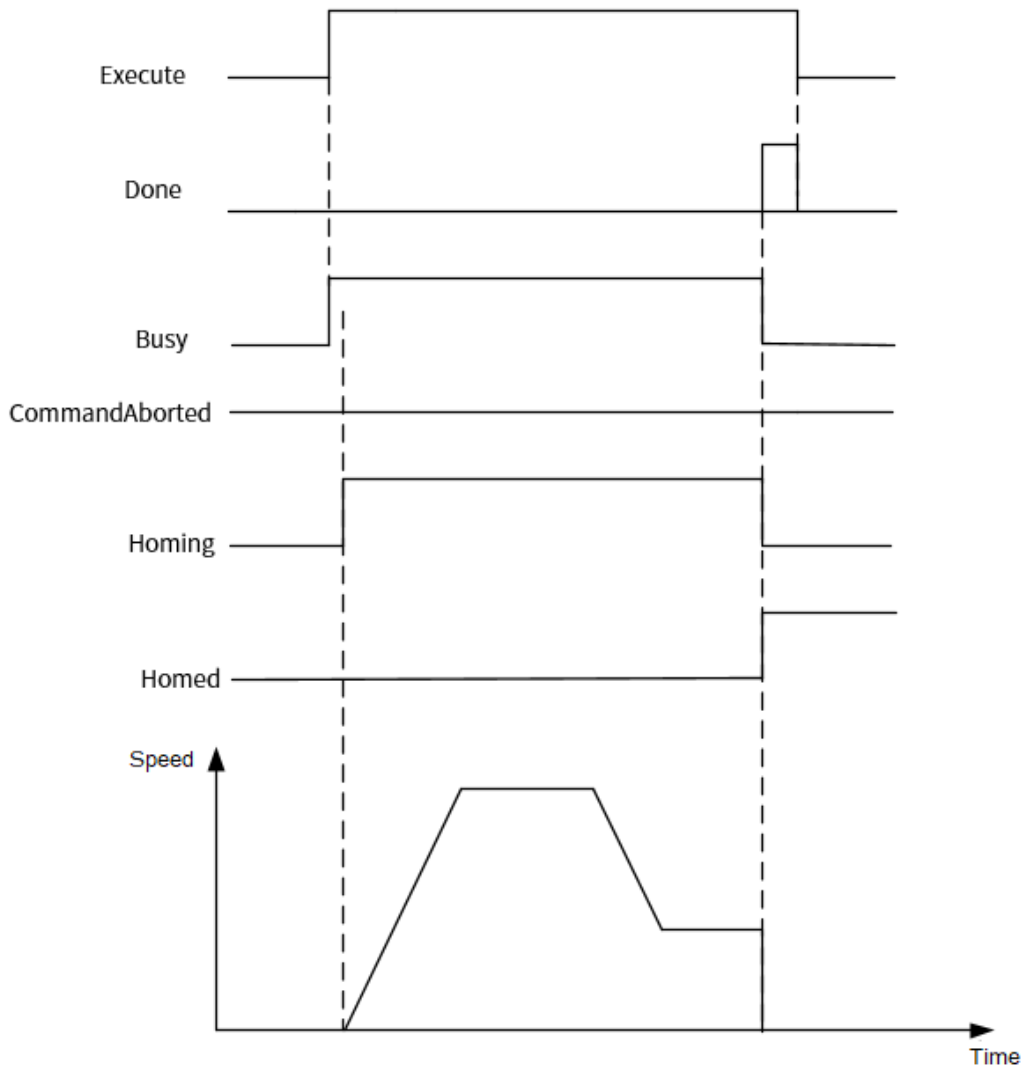
Parameter	Definition	Data Type	Valid Range	Description
Vel	Homing velocity	LREAL	Positive number	Homing approach velocity ≤ Homing velocity (in user units/s).
ApproachVel	Homing approach velocity	LREAL	Positive number	Homing approach velocity ≤ Homing velocity (in user units/s).
Acc	Homing acceleration	LREAL	Positive number or 0	Specify the homing acceleration (in user units/s^2).
Dec	Homing deceleration	LREAL	Positive number or 0	Specify the homing deceleration (in user units/s^2).
Jerk	Homing jerk	LREAL	Positive number or 0	Specify the homing jerk (in user units /s^2).
MaskDistance	Home input mask distance	LREAL	Positive number or 0	Specify the home input mask distance (in user units).
HomePositionOffset	Home position offset	LREAL	Positive number, negative number, or 0	Presets the actual position (in user units) after homing. In Rotary Mode, lower limit ≤ offset < upper limit.
PushTime	Homing holding time	UINT	0 to 10000	Specify the homing holding time (in ms).
Compensation	Homing compensation value	LREAL	Positive number, negative number, or 0	Specify the homing compensation value (in user units) after the home is defined.
HomeTimeOut	Homing timeout period	UDINT	0 to 655350	Specify the homing timeout period (in ms). When it is set to 0, homing timeout is not checked.
RoundTripEnable	Round trip enable	BOOL	TRUE, FALSE	TRUE: Enabled. FALSE: Disabled.

- The arrival check function is only applicable to MotionControl library version 2.4.0.0 or later. Based on the "Positioning range" and "Positioning check time" parameters in the "General setting" of the axis configuration page, it checks in real time whether the actual position has entered the specified target range. After the instruction output for the target position is completed, positioning is complete and the Done output is set if the difference between the actual position and the target position is smaller than the "Positioning range". If positioning is not completed within the "Positioning check time", a fault will occur.

Local pulse axis:

- This instruction cannot be used for local pulse axes.

- **Timing diagram**



- **Precautions**
  - If a slave device bound to the servo axis (e.g., a servo) supports Z-phase signal and probe functions, it can support all homing operation modes. If a slave device bound (e.g., a module) does not support Z-phase signal but supports probe functions, set the home input signal to be probe signal to complete homing.
  - Currently, the positive and negative limit terminals are, by default, assigned to the positive and negative limit signals of the servo end, and binding general-purpose terminals as limit signals is not supported. Therefore, if you select the deceleration mode for "Positive/negative limit input", the axis will stop immediately.
  - Do not check the reverse configuration when using the MC_Homing instruction; otherwise, the servo end will give an alarm if the positive/negative limit is triggered during execution.
  - When the MotionControl library version is 2.2.0.0 or later, the PLC homing function is supported for local pulse axes.

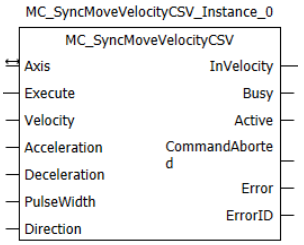
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.55 MC_SyncMoveVelocityCSV

This instruction is used for CSV-based velocity control with adjustable pulse width.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_SyncMoveVelocityCSV	Velocity control with adjustable pulse width		MC_SyncMoveVelocityCSV_Instance_0(Axis :=, Execute :=, Velocity :=, Acceleration :=, Deceleration :=, PulseWidth :=, Direction :=, InVelocity =>, Busy =>, Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction.
Velocity	Velocity	LREAL	Positive number or 0	0	Target velocity.

Acceleration	Acceleration	LREAL	Positive number or 0	0	Target acceleration.
Deceleration	Deceleration	LREAL	Positive number or 0	0	Target deceleration.
PulseWidth	Pulse width	INT	1 to 9999	5000	Pulse width (in 0.01%).
Direction	Motion direction	18.3.3.2.2.9 MC_DIRECTION: Motion Direction Enumeration	1: DIR_POSITIVE 2: DIR_NEGATIVE 3: DIR_CURRENT	1	1: DIR_POSITIVE (Positive direction). 2: DIR_NEGATIVE (Negative direction). 3: DIR_CURRENT (Current direction).

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
InVelocity	Velocity reference reached	BOOL	TRUE, FALSE	FALSE	TRUE when the axis reaches the velocity reference.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Controlling	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled by the current FB.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity	Velocity	Any velocity that exceeds the motion parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	No correction.
Acceleration	Acceleration	Any acceleration that exceeds the motion parameter limit will be automatically corrected to the acceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration less than 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration equal to 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration equal to 0 will be automatically corrected to 10 times the velocity.
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the deceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration less than 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration equal to 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration equal to 0 will be automatically corrected to 10 times the velocity.

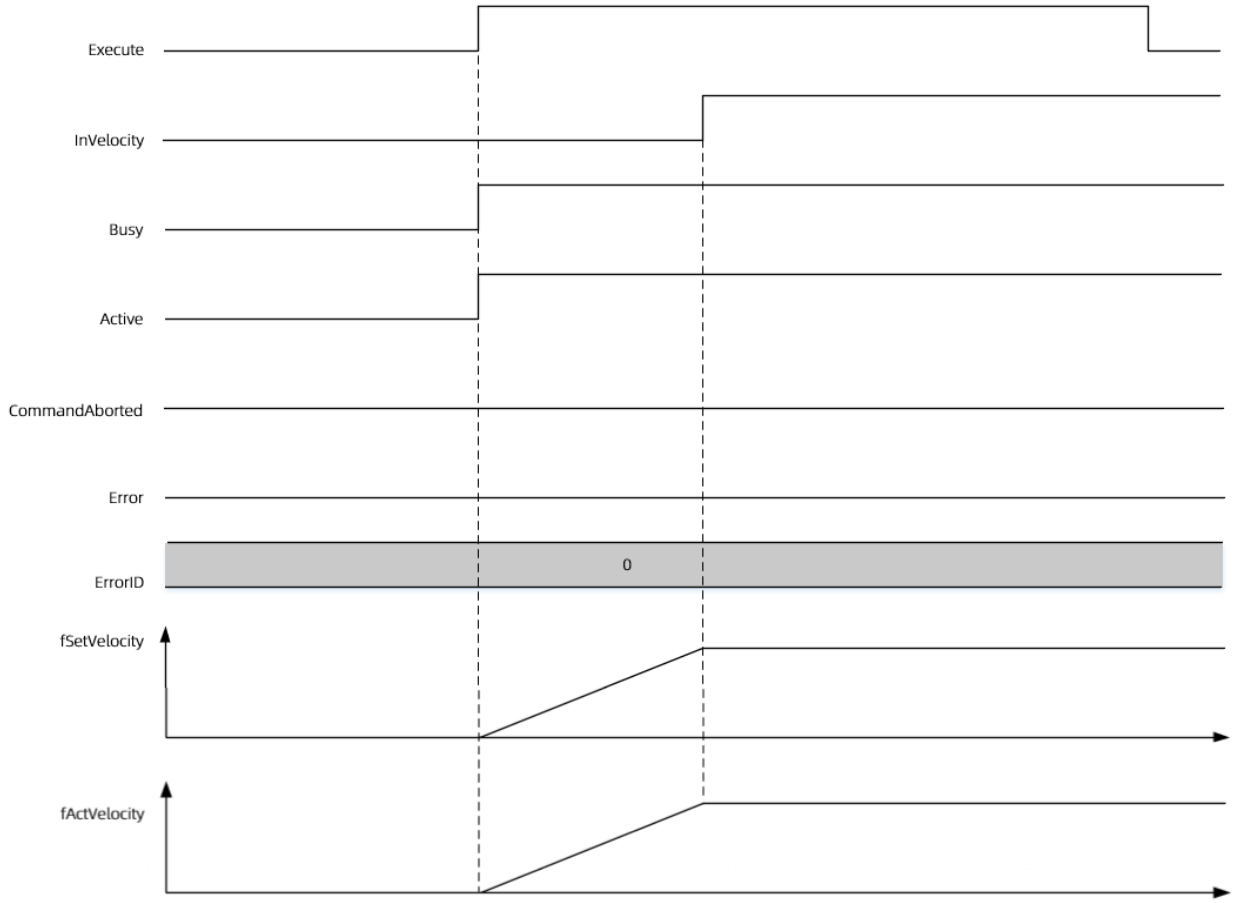
Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
PulseWidth	Pulse width	If the value exceeds 9999, it will be corrected to 9999.	If the value is less than 1, it will be corrected to 1.	It will be corrected to 1.

- **Function**
  - This instruction can be used for bus-controlled axes and local pulse axes. When the bus-controlled axis is controlled in the Cyclic Synchronous Velocity (CSV) control mode or the local pulse axis is controlled in the Cyclic Synchronous Position (CSP) control mode, set the axis in the AXIS_CONTINUOUS_MOTION state.
  - Before this instruction is executed on a bus-controlled axis, configure three PDOs: 0x6060, 0x6061, and 0x60FF.
  - When a local pulse axis is used, this instruction enables PWM (pulse width modulation) output with acceleration/deceleration and without acceleration/deceleration: When the input parameters Acceleration and Deceleration are set to non-zero, the PWM output is based on the specified values of the parameters Acceleration and Deceleration. When the input parameters Acceleration and Deceleration are set to zero or one of them is set to zero, the target velocity is reached without acceleration or deceleration in one cycle after the instruction is executed and the PWM output starts. The input parameter PulseWidth is used to control the pulse width of the PWM waveform.
  - The EVO760 series devices have no single-terminal Y0 or Y1. Therefore, it is not recommended to use this instruction on the EVO760 series devices through a pulse axis.
  - During the execution of this instruction, if you want to stop the axis, you can call the MC_Stop, MC_Halt, or MC_ImmediateStop instruction; it is prohibited to call the MC_MoveSuperImposed instruction for motion superimposition.
- **Timing diagram**

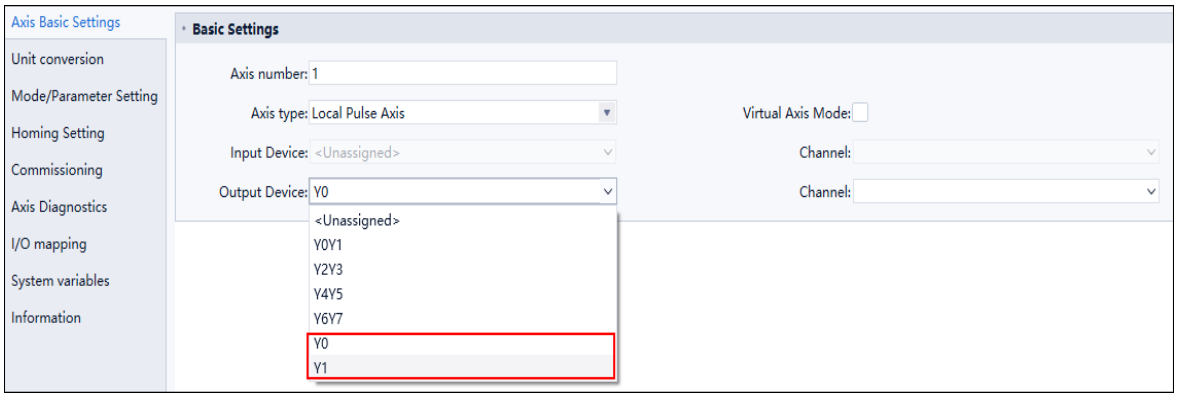
  The parameter Execute of the function block must be triggered on a rising edge. The InVelocity state of the function block indicates the velocity reference is reached; the parameter Busy indicates that the current function block is being executed; the parameter Active indicates the axis is being controlled; the parameter CommandAborted indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

  After this instruction is called, the output parameter Busy is set to TRUE on a rising edge of Execute. In the next cycle, the output parameter Active shows the value TRUE.

  When the axis reaches the velocity reference, the output parameter InVelocity is set to TRUE.



- **Precautions**
  - If this instruction is executed on a local pulse axis, select the output device as Y0 or Y1. If you select dual-ended output, a fault will occur.



  - If this instruction is executed on a bus-controlled axis, do not use the MC_SetPosition instruction.
  - When this instruction is executed on a local pulse axis, it is recommended that the minimum pulse frequency be 5 Hz.
  - When a servo axis is enabled after PLC project cold reset and the MC_Home instruction is executed using No. 35 homing method, if this instruction is executed with DIR_CURRENT being set, the motion direction of this instruction will be affected.

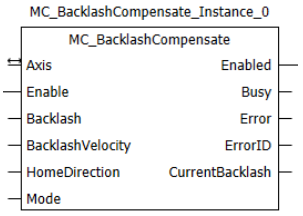
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.56 MC_BacklashCompensate

This instruction enables backlash compensation and is used to compensate for the dead travel in the mechanical system.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_BacklashCompensate	Backlash compensation		MC_BacklashCompensate_Instance_0(Axis :=, Enable :=, Backlash :=, BacklashVelocity :=, HomeDirection :=, Mode :=, Enabled =>, Busy =>, Error =>, ErrorID =>, CurrentBacklash =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	Enable flag for backlash compensation.
Backlash	Backlash compensation value	LREAL	Greater than 0	0	Backlash compensation value, greater than 0. The positive or negative direction of the compensation value is determined by internal calculation.

BacklashVelocity	Compensation velocity	LREAL	Greater than or equal to 0	0	Velocity for backlash compensation. When the compensation velocity is set to 0, the backlash is compensated over one cycle; if greater than 0, the backlash is compensated according to the specified compensation velocity.
HomeDirection	Absolute homing direction	BOOL	TRUE, FALSE	FALSE	TRUE: Positive direction. FALSE: Negative direction.
Mode	Compensation mode	INT	0	0	0: Linear compensation. Other: To be determined (not yet implemented).

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Enabled	Enabled	BOOL	TRUE, FALSE	FALSE	TRUE: Compensation enabled. FALSE: Compensation not enabled.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Error	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.
CurrentBacklash	Current backlash compensation	LREAL	-	0	Current backlash compensation value.

- **Function**
  - Sources of backlash:

    Clearance introduced by keyed connection between the motor shaft and gear, clearance between the gears, clearance introduced by keyed connection between the gear and screw, clearance introduced by keyed connection in the coupling, and clearance between the screw

and its nut. These clearances are negligible in one direction and mainly occur during reverse motion.

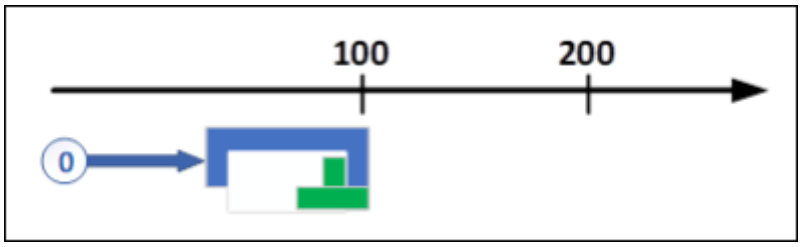
- Impacts of backlash:

  When the worktable moves from positive to negative direction, its actual displacement deviates the theoretical value, and the deviation accumulates in the total worktable displacement. Due to the backlash, the motor may be idling during direction reversal without actual worktable movement, which also affects positioning accuracy and repeatability.

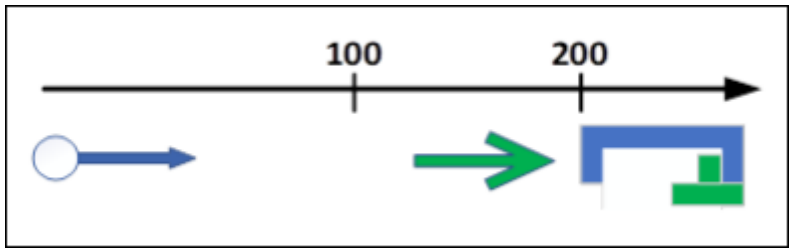
- Method for measuring backlash:

  1. Start homing to define the reference position.

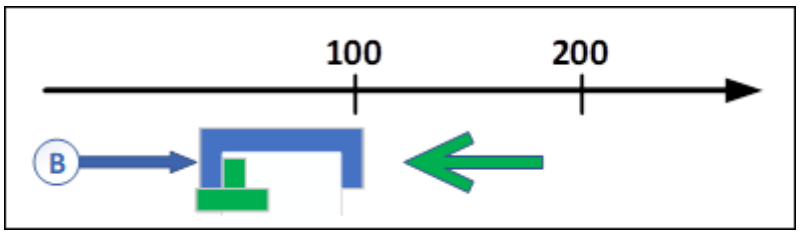
  2. Install a dial gauge and set the scale to 0.



  3. Move absolutely in the positive direction by 200.



  4. Move absolutely in the negative direction to 0.



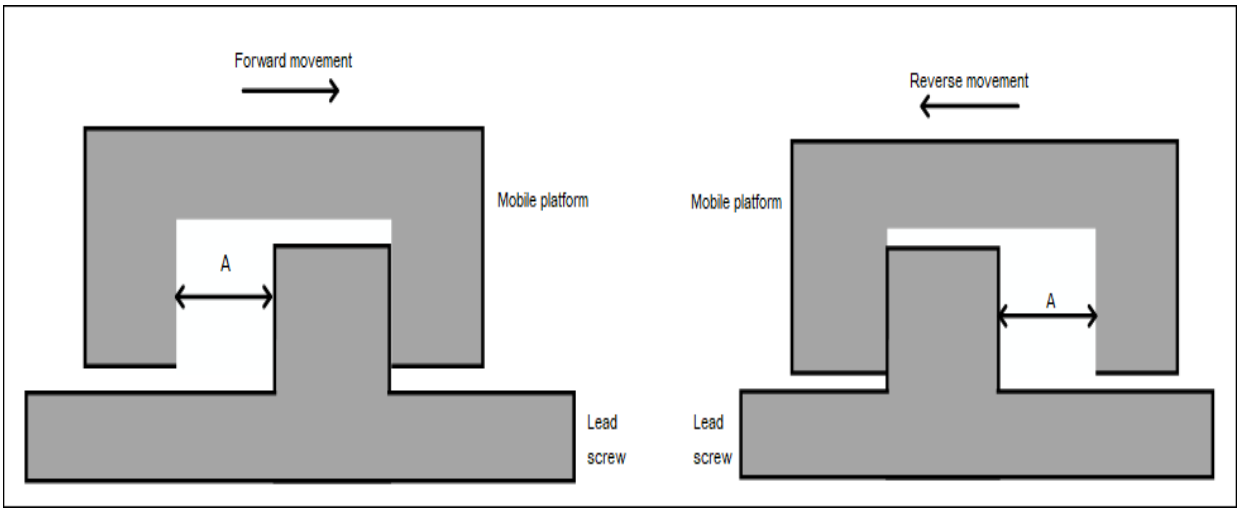
  5. Read the dial gauge. The reading represents the error.

Shaft end variable parameter names

Data Type	Parameter	Definition
BOOL	bBacklashEnable	Enable flag for backlash compensation
LREAL	fCurrentBacklash	Backlash compensation value

To use the backlash compensation function block, first perform homing or confirm whether the motion direction is the same as the previous motion's end direction. The compensation during the first execution of the instruction is shown in the following figure, where A represents the compensation amount. Without performing homing, the compensation may be incomplete, or the worktable displacement may be inaccurate.

Homing end direction and moving direction upon instruction execution	Compensation value
Same direction	0
Different direction	± A



- **Precautions**
  - It is not recommended to modify input parameters during the compensation; modified parameters will not take effect until the instruction is re-executed.
  - If backlash error and pitch error exist simultaneously, compensate the backlash error before the pitch error.
  - The instruction can be executed only when the axis is in the Standstill state; otherwise, a fault will occur.
  - You must specify proper compensation velocity and compensation value; otherwise, it may cause servo overspeed. When the compensation velocity is 0, the compensation value is added to the planned value over one cycle. This may cause a significant jump in the actual velocity, so it must be used with caution.
  - If the Enable is set, Enabled is set during the compensation and then reset after the compensation is completed.
  - The CurrentBacklash shows the current compensation value in real time. The compensation value will be cleared if the Enable is reset or a fault occurs, resulting in that the compensation ramp-down to 0 at the maximum velocity.
  - The instruction can be placed in any order.
  - After the motion stops, disabling the MC_BacklashCompensate instruction may cause slight axis movement.
  - In multi-instance scenarios, if a subsequent instruction triggers a fault, it will not affect the normal compensation of the first instruction. When the second function block is triggered during the execution of the first function block, the second function block will report a fault but not affect the normal execution of the first function block.
  - The compensation velocity must be less than or equal to the velocity limit set in the software. Excessive compensation velocity may cause servo overspeed.
  - The compensation function can be used for servo axes, virtual axes, and pulse axes.

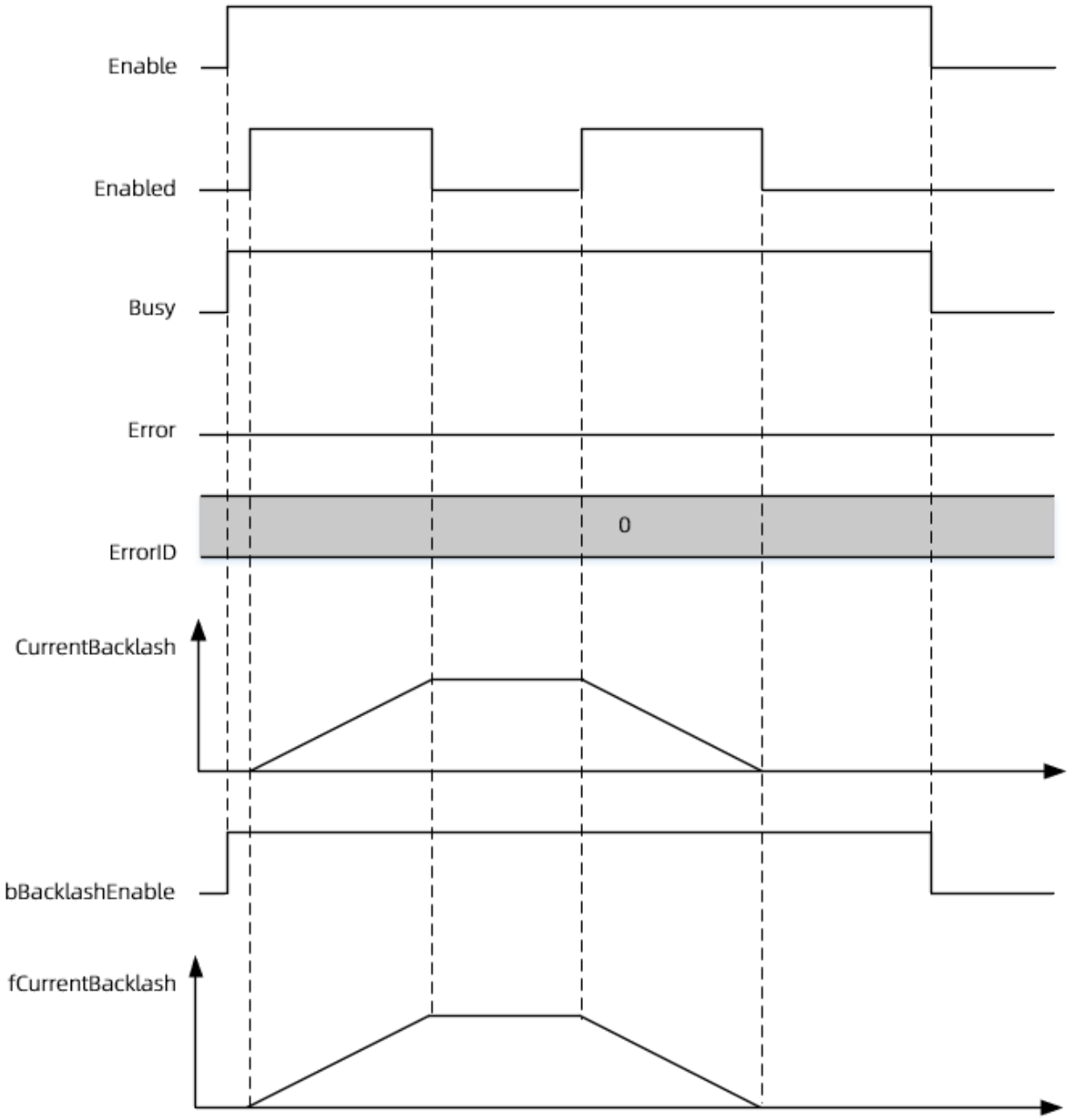
- If homing is triggered upon the start of compensation, a fault will occur.
- Although this instruction is a single axis instruction, it can also be applied to synchronous axes or axes groups.
- During axis commissioning, the compensation value is retained and the compensation continues without a fault message.
- When this instruction is triggered in a mode other than CSP or CSV control mode, a fault occurs. To use any instructions in other control modes, first clear the compensation value.
- For a rotary axis, the compensation value must be less than the rotation cycle; otherwise, the actual axis position may be inaccurate.
- Disabling or enabling the servo stop mode of immediate stop instruction during compensation can cause a sudden position change over one cycle.
- This instruction enables compensation in position mode only. It will not take effect for single-ended pulse axes.

- **Timing diagram**

  After this instruction is called, the output parameter Busy is set to TRUE on a rising edge of Execute.

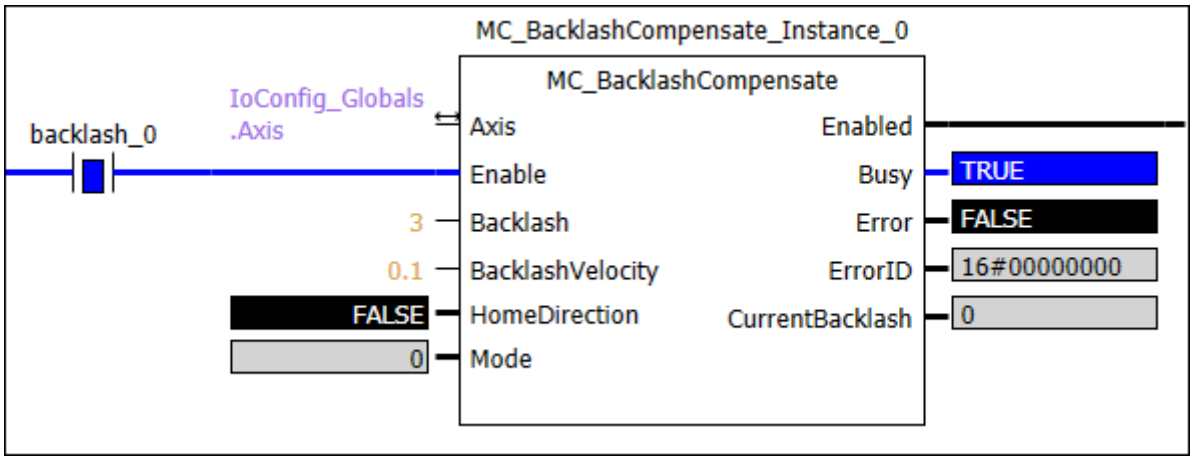
  At this point, execute the MC_MoveVelocity instruction in the positive direction. While the axis is being compensated, the output parameter Enabled is set to TRUE; after the compensation, the output parameter Enabled is set to FALSE.

  Change the direction of the MC_MoveVelocity instruction to negative for reverse operation. While the axis is being compensated, the output parameter Enabled is set to TRUE; after the compensation, the output parameter Enabled is set to FALSE.

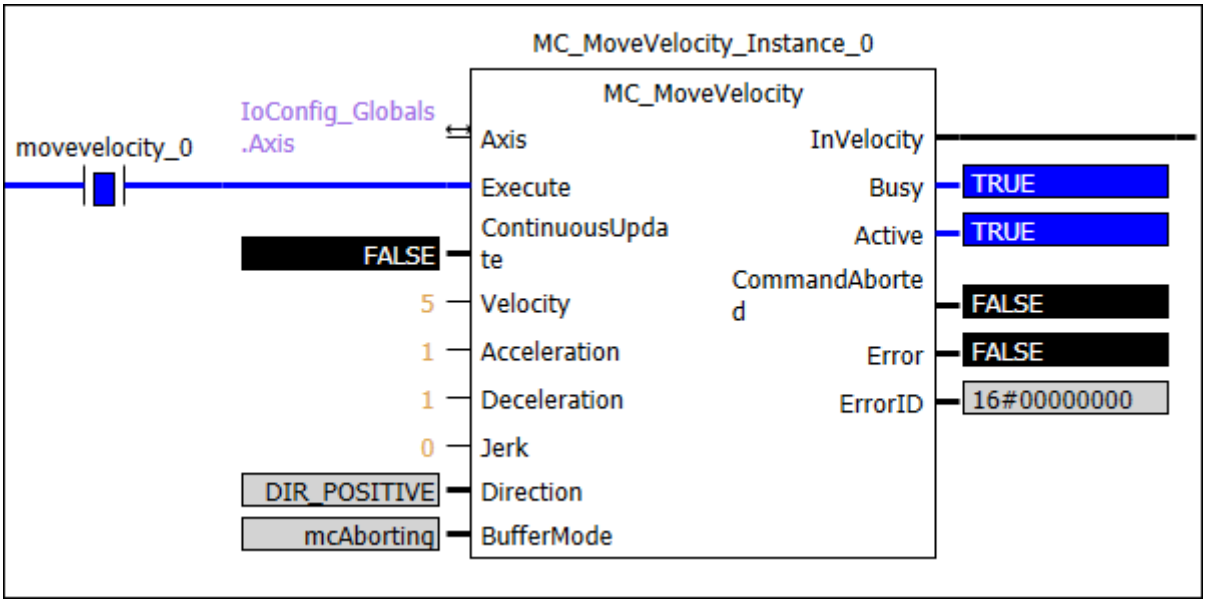


- **Example**

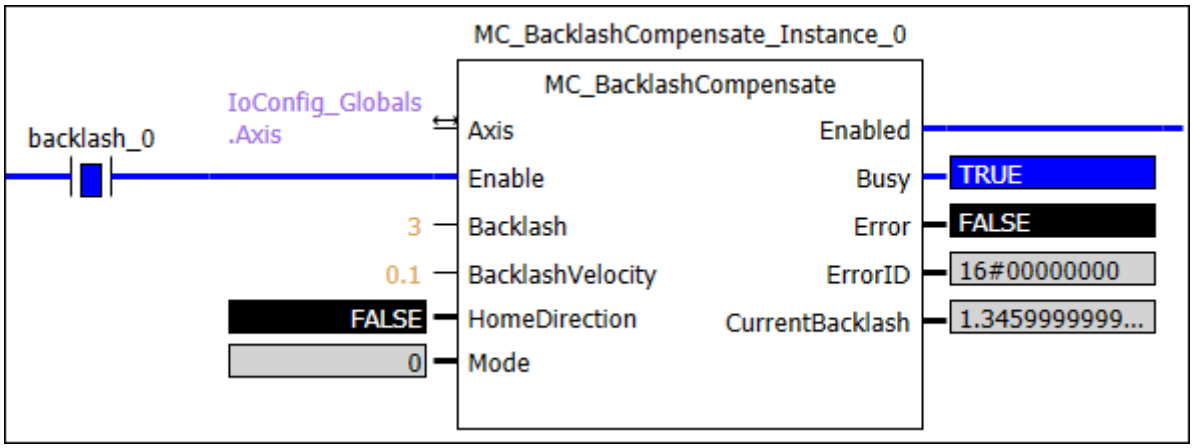
1. Execute the instruction.



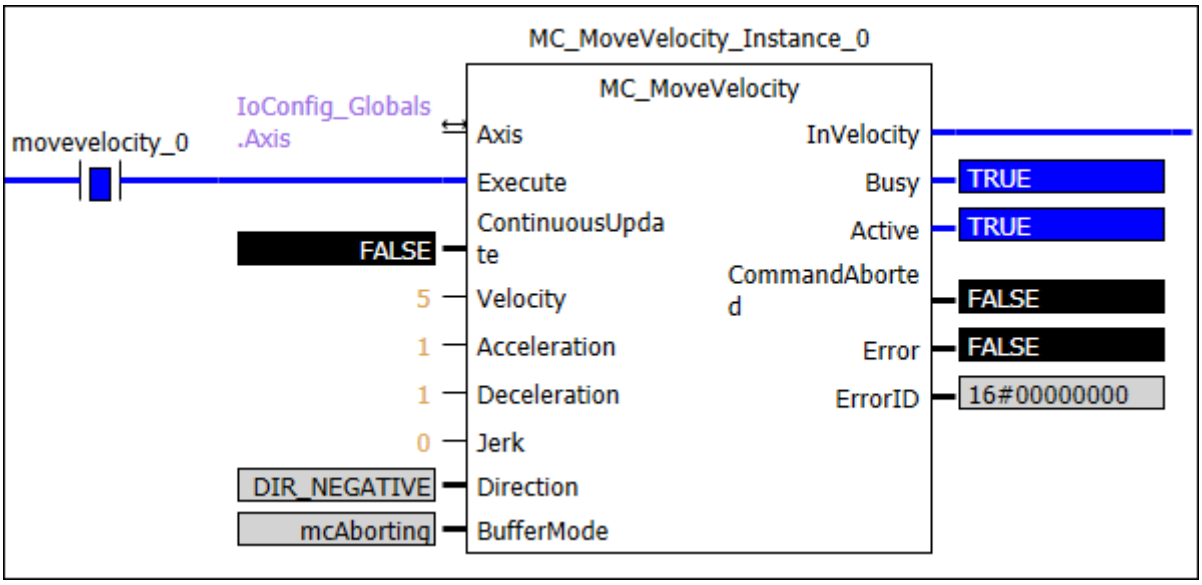
2. Trigger the velocity instruction in the positive direction.



3. At this point, the instruction is being executed.



4. Change the direction of the velocity instruction to negative.



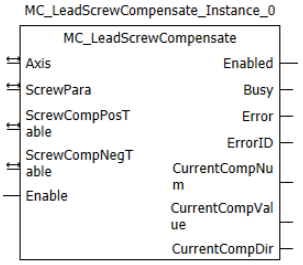
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.57 MC_LeadScrewCompensate

Pitch error generally occurs when leadscrews are used. During the transmission of the leadscrew pairs, various errors exist between the leadscrew and nut, such as pitch error, thread raceway error, and diameter error. At the same time, the leadscrew stretching, extruding, or parallelism during assembly may introduce pitch errors. To improve the actual motion accuracy of the leadscrew, it is necessary to compensate for proper error values in the motion planning.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_LeadScrewCompensate	Leadscrew pitch error compensation		MC_LeadScrewCompensate_Instance_0(Axis :=, ScrewPara :=, ScrewCompPosTable :=, ScrewCompNegTable :=, Enable :=, Enabled =>, Busy =>, Error =>, ErrorID =>, CurrentCompNum =>, CurrentCompValue =>, CurrentCompDir =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.
ScrewPara	Screw compensation structure	ScrewConStruct	-	0	Screw compensation structure.
ScrewCompPosTable	Positive screw compensation table	Variable-length array	-	0	Positive screw compensation table with 0 as the first element value.

ScrewCompNegTable	Negative screw compensation table	Variable-length array	-	0	Negative screw compensation table with 0 as the first element value.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: Compensation enabled. FALSE: Compensation disabled.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Enabled	Enabled	BOOL	TRUE, FALSE	FALSE	TRUE: Compensation enabled. FALSE: Compensation not enabled.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Error	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.
CurrentCompNum	Current compensated segment number	INT	-	0	Current compensated segment number.
CurrentCompValue	Current segment compensation value	LREAL	-	0	Current segment compensation value.

CurrentCompDir	Current compensation direction	INT	-	0	Current compensation direction: Positive direction: 1. Negative direction: 0.

ScrewConStruct structure

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Mode	Compensation mode	INT	0, 1	0	0: Unidirectional. 1: Bidirectional.
StartPos	Start position	LREAL	-	0	Compensation start position.
EndPos	End position	LREAL	-	0	Compensation end position.

- **Function**

  Method for obtaining the screw compensation table:

  - First, select a section of the travel path with the start position specified as startPos and the stop position specified as endPos (endPos > startPos), and determine the number of segments the selected travel is divided into. Assume the travel is divided into N segments, the axis moves in the positive direction from startPos. At each position spaced by (endPos - startPos)/N, the position reference and the actual position are recorded to calculate their difference cmpVal, resulting in N compensation values.
  - Next, after the axis reaches the actual endPos in the positive direction, it starts moving in the negative direction from endPos. Use the same method described above to obtain N negative compensation values.
  - Finally, generate a compensation table with a length of N + 1 based on the number of segments N. For the positive compensation table, add a starting compensation value of 0 at the startPos. For the negative compensation table, add a starting compensation value of 0 at the endPos.



- **Precautions**
  - The parameter check for the compensation structure and compensation tables are only performed once when the instruction is triggered. It is prohibited to modify the parameters in the structure after the instruction is executed.
  - In Mode 0, StartPos < EndPos indicates forward compensation. Conversely, when the value of the start position is greater than that of the end position, it is considered negative compensation. When the

Mode is set to 1, the function block automatically calculates the start and end positions. For positive direction, the value of the start position is less than that of the stop position; for negative direction, the value of the start position is greater than that of the end position.

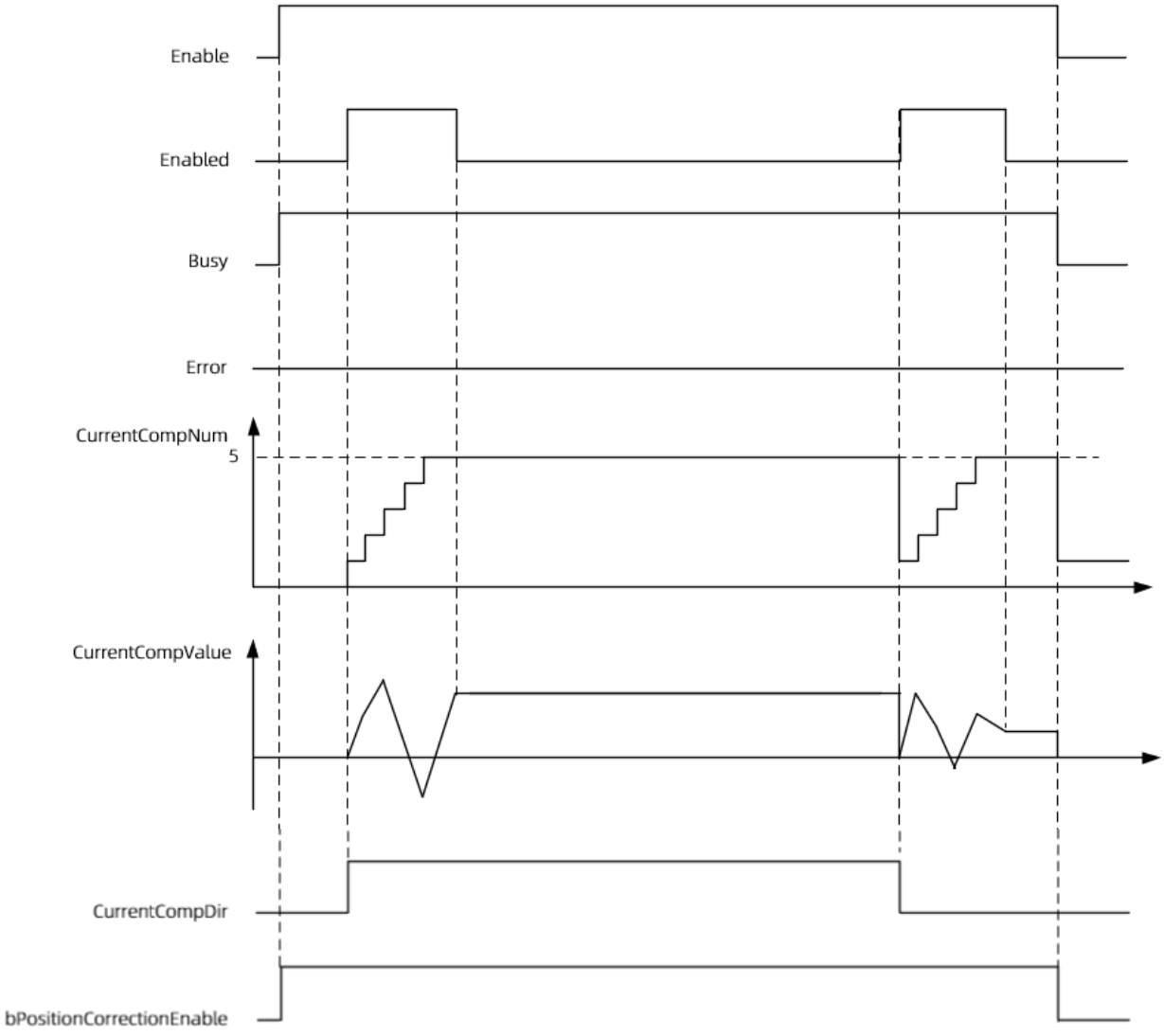
- When the value of the start position is greater than that of the end position, perform negative compensation based on the negative compensation table with 0 as the first element.
- When the value of the start position is less than that of the end position, perform positive compensation based on the positive compensation table with 0 as the first element.
- After the Enable is reset or a fault occurs, clear the compensation value at the maximum velocity.
- After the MC_LeadScrewCompensate instruction is executed, it is prohibited to execute the MC_Setposition instruction.
- Leadscrew pitch compensation is not supported for rotary axes.
- Specify proper value for the error parameter in the compensation table. An improper error compensation may result in overspeed. It is recommended to test the parameter value within a proper range. Generally, the errors in the compensation table are very small, often several orders of magnitude smaller than the user units.
- During the pitch error measurement, if the backlash of the screw is relatively large, it is recommended to set the backlash parameters of the screw before measuring the pitch error. This is because if the backlash is large, the pitch error measured during the reversal will be compensated in an instant, which may cause the motor to fail to respond or trigger an overspeed alarm.
- In multi-instance scenarios, if a subsequent instruction triggers a fault, it will not affect the normal compensation of the first instruction. When the second function block is executed during the execution of the first function block, the second function block will report a fault but not affect the normal execution of the first function block.
- Pitch compensation, as a property of a single axis, is also supported by synchronous axes and axis groups.
- During axis commissioning, the compensation value is retained and the compensation continues without a fault message.
- When this instruction is triggered in a mode other than CSP or CSV control mode, a fault occurs. To use any instructions in other control modes, first clear the compensation value. Before using a homing or torque instruction, reset and then execute the instruction; otherwise, it will cause inaccurate actual velocity.
- This instruction enables compensation in position mode only. It will not take effect for single-ended pulse axes.
- The compensation for reciprocating motion will be cleared to 0.
- Disabling or enabling the servo stop mode of immediate stop instruction during compensation can cause a sudden position change over one cycle.
- Up to 1000 points of compensation are supported.
- If multiple segments need to be compensated, a longer table can be set, with segments that do not require compensation being set to 0.

- **Timing diagram**

  After this instruction is called, the output parameter Busy is set to TRUE on a rising edge of Execute.

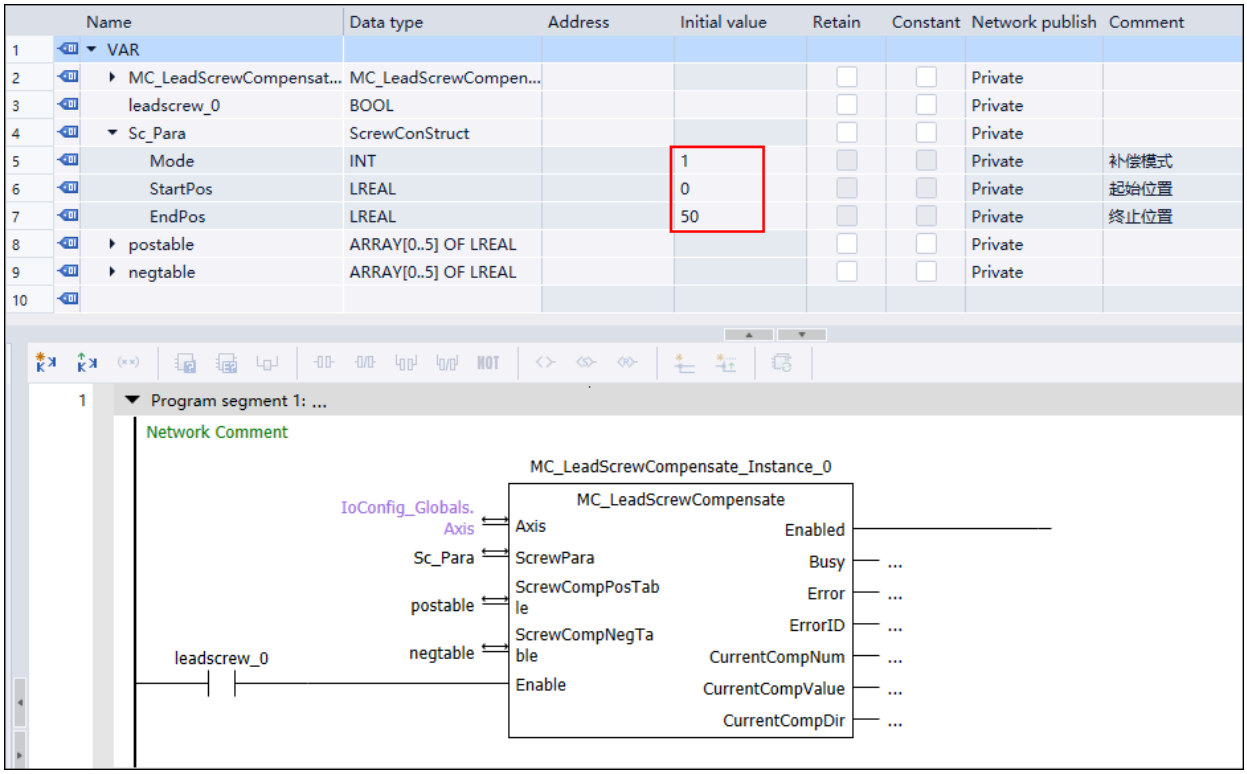
At this point, when the MC_Jog is executed, the axis motion is being compensated with the output parameter Enabled being set to TRUE and the output parameter CurrentCompDir being set to 1 (Positive direction); after the compensation, the output parameter Enabled is set to FALSE.

When the MC_Jog is executed in the negative direction, the axis motion is being compensated with the output parameter Enabled being set to TRUE and the output parameter CurrentCompDir being set to 0 (Negative direction); after the compensation, the output parameter Enabled is set to FALSE.

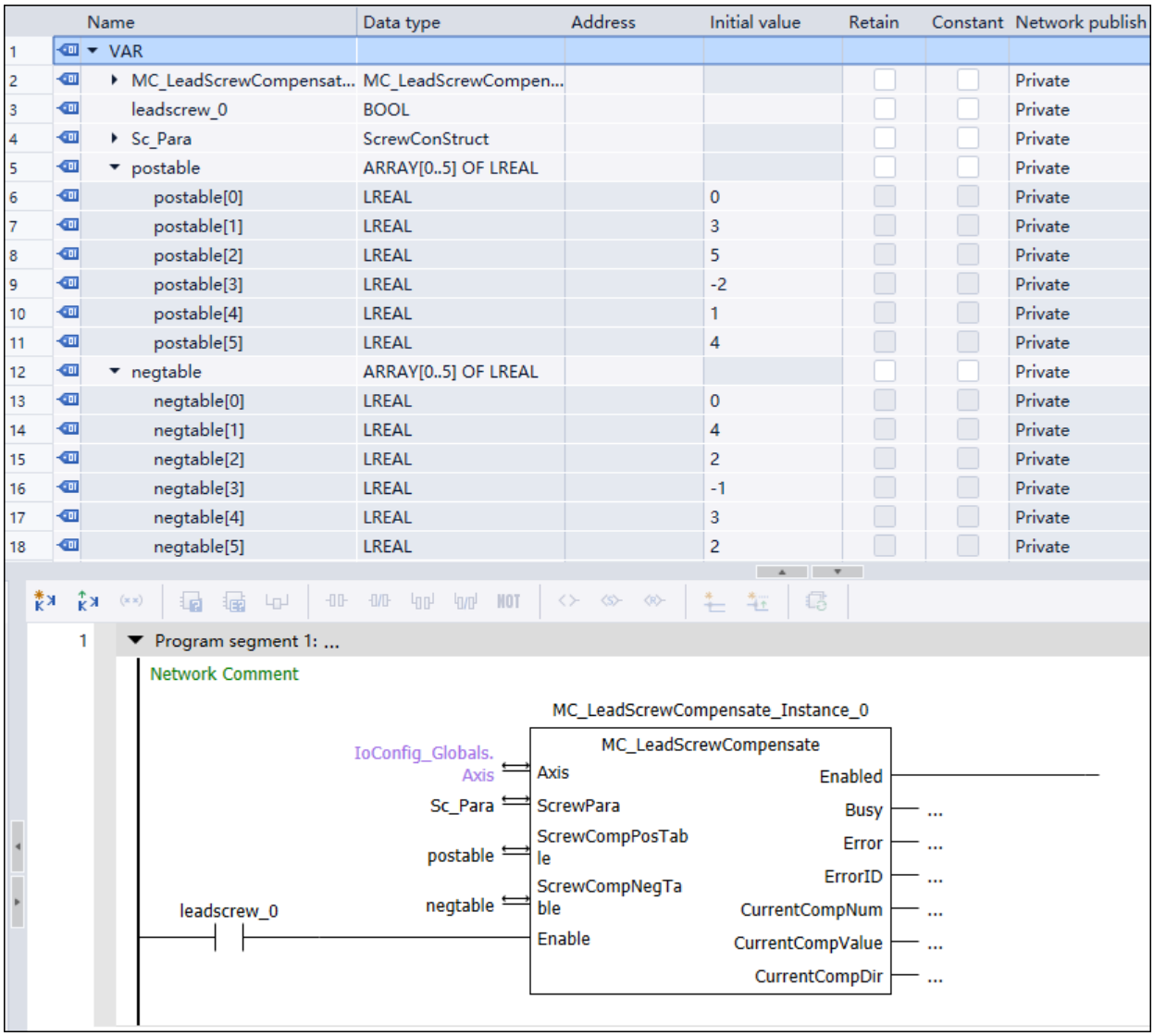


- **Example**

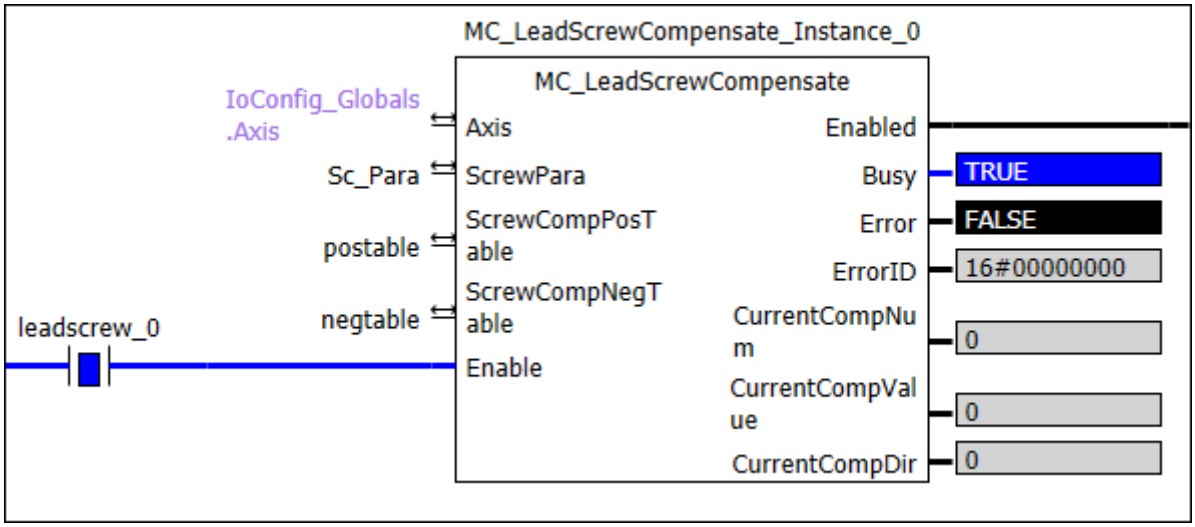
  1. Input the structure parameters, with the start and end points for bidirectional compensation being 0 and 50.



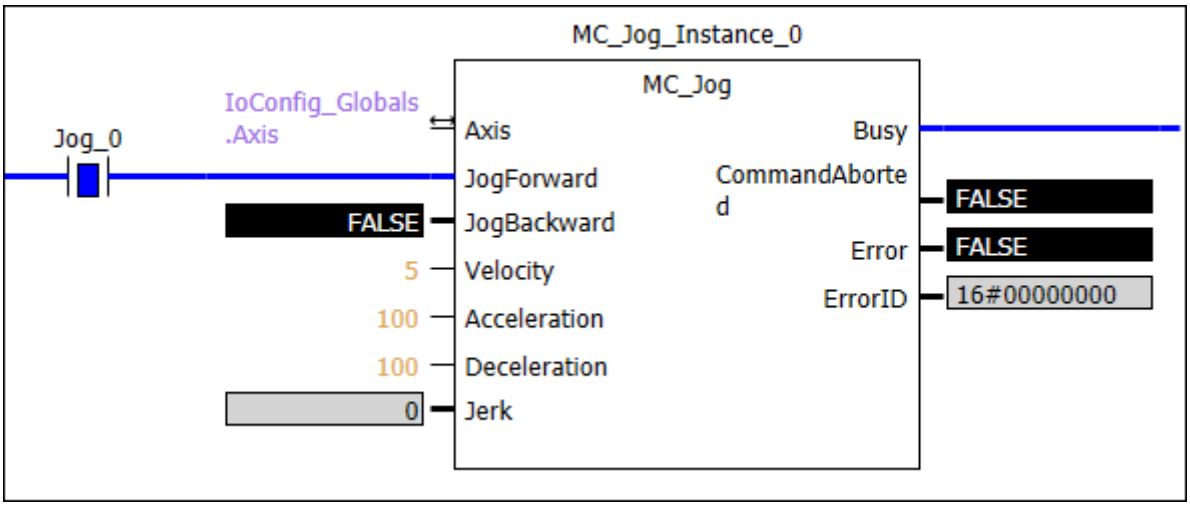
2. Input the positive and negative compensation tables.



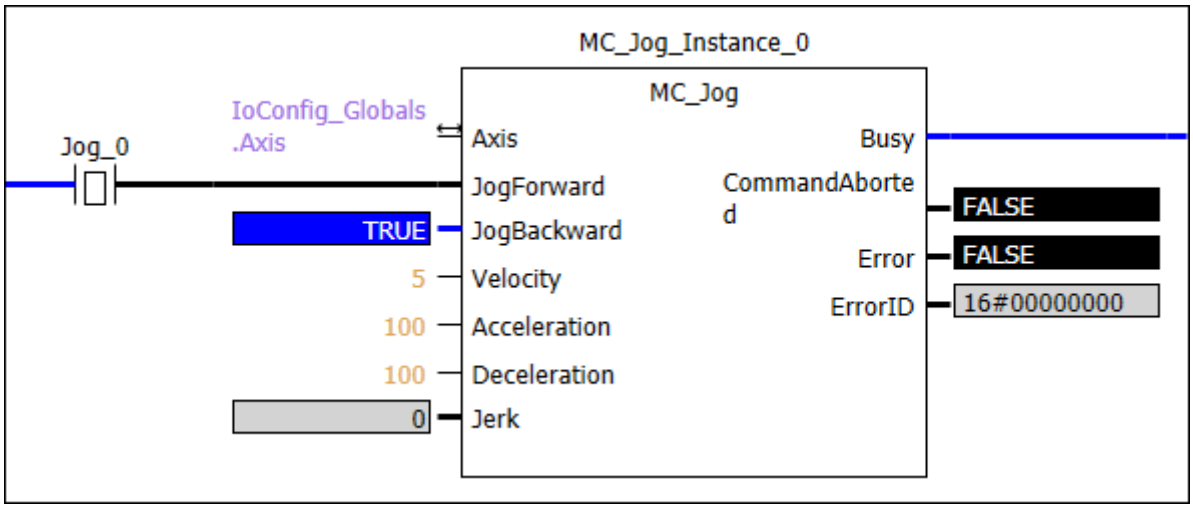
3. Execute the compensation instruction.



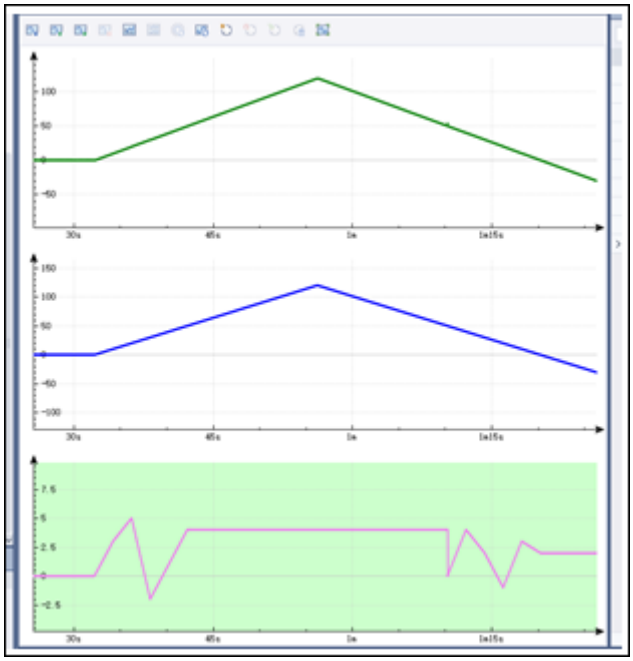
4. Execute the motion control instruction.



5. Execute the motion control instruction in the negative direction.



6. The actual position, planned position, and compensation value collected by the oscilloscope are shown in the following figure.



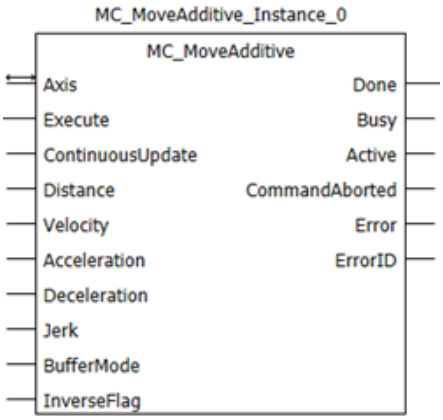
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.58 MC_MoveAdditive

This instruction enables the original axis position reference to be added with a value specified in Distance. It is used for online position superimposition in the axis control process. This instruction can be executed at any time when the axis is in the Discrete Motion state. When the axis is in the Standstill state, this instruction is equivalent to MC_MoveRelative.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_MoveAdditive	Add a distance to the last specified target position		MC_MoveAdditive_Instance_0(Axis :=, Execute :=, ContinuousUpdate :=, Distance := , Velocity := , Acceleration :=, Deceleration :=, Jerk := , BufferMode := , InverseFlag :=, Done =>, Busy => , Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
ContinuousUpdate	Continuous update	BOOL	TRUE, FALSE	FALSE	Continuous parameter update effective flag. Prerequisite: The parameter ContinuousUpdate shows the value TRUE and the parameter Active shows the value TRUE. Effective parameters: Position, Velocity. Note: The modified parameter values must differ from the original parameter values by more than 0.01.
Distance	Position reached	LREAL	Positive number, negative number, or 0	0	Position data [u] to be added.
Velocity	Velocity	LREAL	Positive number	0	Maximum velocity [u/s], which may not be reached (cannot be zero).
Acceleration	Acceleration	LREAL	Positive number	0	Acceleration [u/s^2] of the acceleration phase.
Deceleration	Deceleration	LREAL	Positive number	0	Deceleration [u/s^2] of the deceleration phase.
Jerk	Jerk	LREAL	Positive number	0	Slope [u/s^3] of acceleration or deceleration. (Only valid for S-curve planning).

Input Variable	Name	Data Type	Valid Range	Default Value	Description
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting 1: mcBuffered 2: mcBlendingLow 3: mcBlendingPrevious 4: mcBlendingNext 5: mcBlendingHigh	0	0: mcAborting (Aborting). 1: mcBuffered (Buffered). 2: mcBlendingLow (Blending with the lowest velocity of both FBs). 3: mcBlendingPrevious (Blending with the velocity of the first FB). 4: mcBlendingNext (Blending with velocity of the second FB). 5: mcBlendingHigh (Blending with highest velocity of both FBs).
InverseFlag	Inverse enable	BOOL	TRUE, FALSE	FALSE	FALSE:The planned position cannot be beyond the target position. When the target velocity is not reached, the planned parameters (End Velocity, Acceleration/Deceleration, Jerk) will be modified to meet the planning requirements. TRUE: The planned position can be beyond the target position. If the axis moves beyond the target position, it will move for an additional cycle.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Controlling	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled by the current FB.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity	Velocity	Any velocity that exceeds the motion parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	• When the MotionControl library version is 2.1.0.0 or earlier, the velocity equal to 0 will be automatically corrected to 1e-5. • When the MotionControl library version is 2.2.0.0 or later, the velocity equal to 0 will not be corrected.

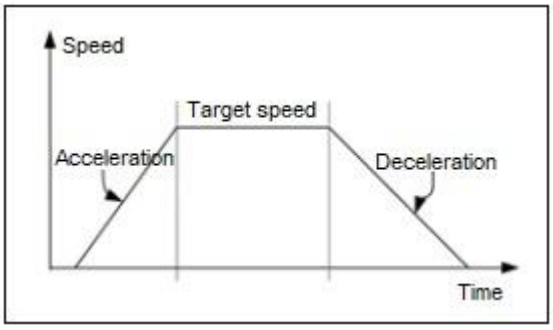
Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Acceleration	Acceleration	Any acceleration that exceeds the motion parameter limit will be automatically corrected to the acceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration less than 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the acceleration equal to 0 will be automatically corrected to 10% of the acceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the acceleration equal to 0 will be automatically corrected to 10 times the velocity.
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the deceleration limit.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration less than 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration less than 0 will be automatically corrected to 10 times the velocity.	• When the MotionControl library version is 2.3.0.0 or earlier, the deceleration equal to 0 will be automatically corrected to 10% of the deceleration limit. • When the MotionControl library version is 2.4.0.0 or later, the deceleration equal to 0 will be automatically corrected to 10 times the velocity.

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Jerk	Jerk	If the motion profile is set to S-curve, any jerk that exceeds the software limit will be automatically corrected to the jerk limit.	If the motion profile is set to S-curve, any jerk less than 0 will be automatically corrected to 10% of the jerk limit.	If the motion profile is set to S-curve, the jerk equal to 0 will be automatically corrected to 10% of the jerk limit.

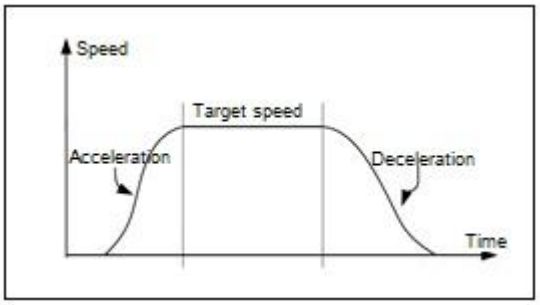
- **Function**

  Bus-controlled axis:

  - This function block is used to add a Distance to an active motion instruction for an axis.
  - When the axis state is Discrete Motion, CommandAborted of other instructions will be set when this function block is running.
  - When the axis state is Standstill, this instruction can be executed independently to achieve relative positioning.
  - This function block is executed on the rising edge of Execute.
  - When the input parameter ContinuousUpdate is set to TRUE, modified parameters Distance and Velocity will take effect only when the function block is in the Active state. Parameter changes greater than 0.01 are considered updates. Modified BufferMode cannot take effect during this period, and the buffer mode at the time of the instruction trigger is retained. When the output parameter Done is set, it indicates that the instruction has completed the motion, and any changes to the parameters will not take effect.
  - When the T-curve is specified in the software, the input parameter Jerk is invalid.
    T-curve acceleration/deceleration motion:

    S-curve acceleration/deceleration motion:

  - The arrival check function is only applicable to MotionControl library version 2.4.0.0 or later. Based on the "Positioning range" and "Positioning check time" parameters in the "General setting" of the axis configuration page, it checks in real time whether the actual position has entered the specified target range. After the instruction output for the target position is completed, positioning is complete and the Done output is set if the difference between the actual position and the target position is smaller than

the "Positioning range". If positioning is not completed within the "Positioning check time", a fault will occur.

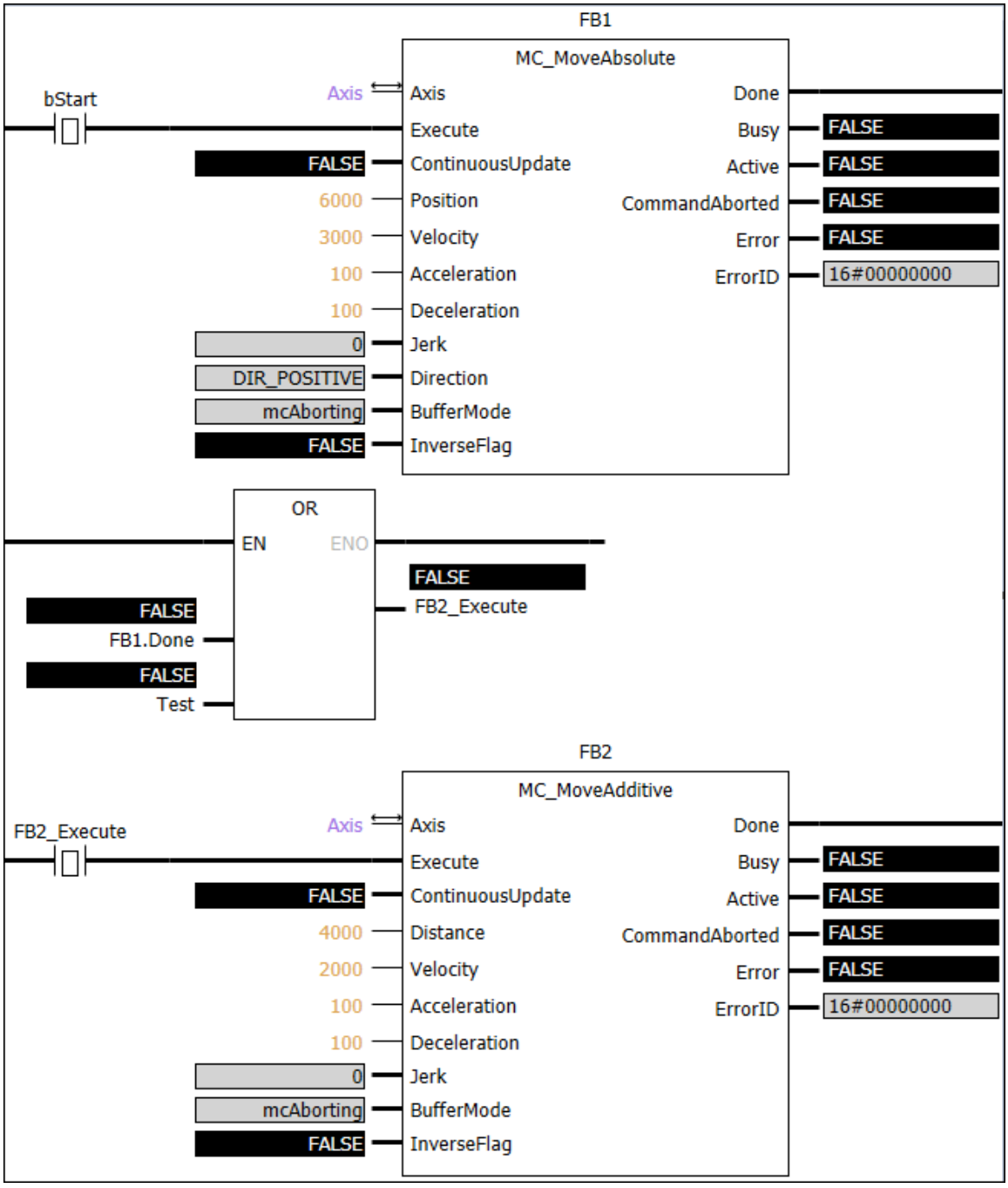
Local pulse axis:

This instruction can be used for local pulse axes.

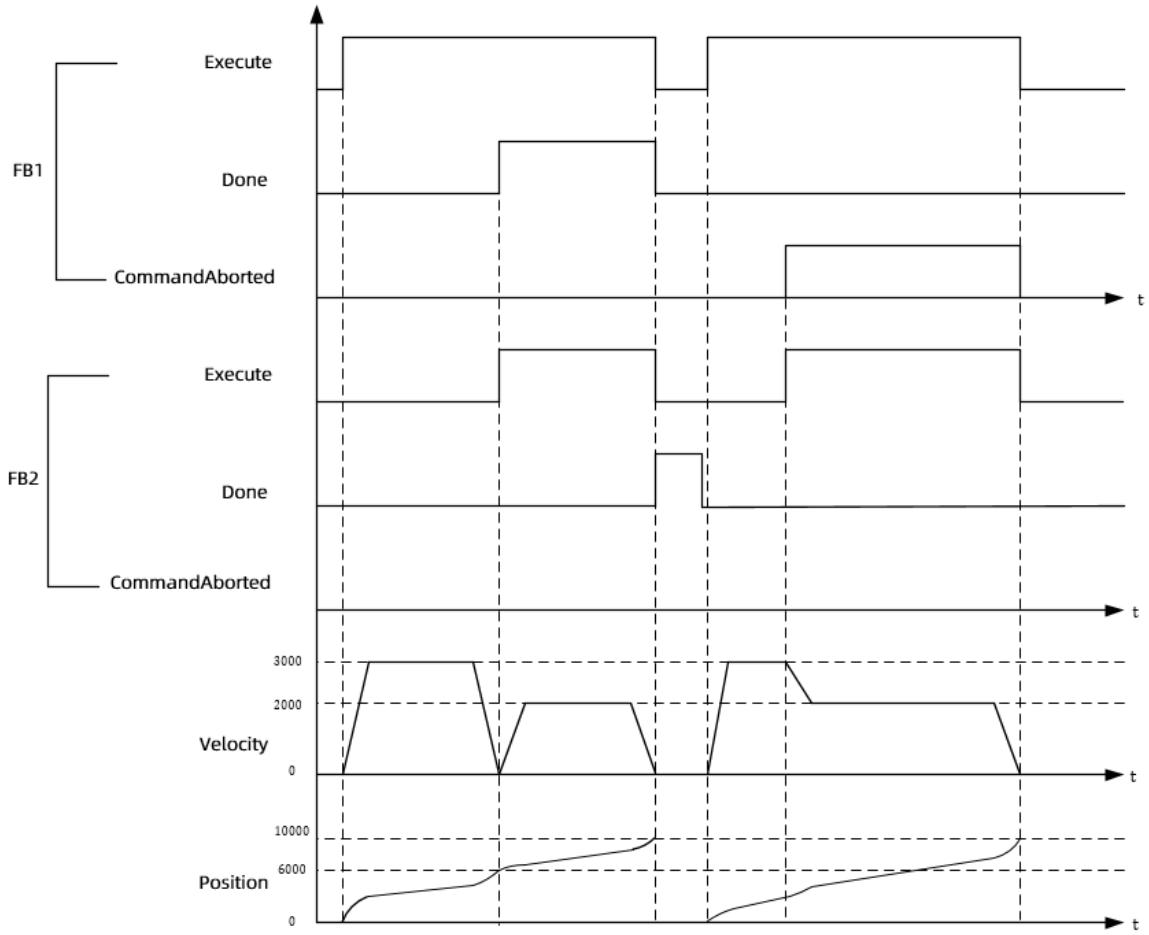
- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates the instruction has been executed successfully; the parameter Busy indicates that the current function block is being executed; the parameter CommandAborted indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

  This instruction is usually used in combination with primary motion instructions such as MC_MoveAbsolute and MC_MoveRelative. For example, it can be executed after or during absolute positioning. An example is shown below.



The timing diagram is shown below.



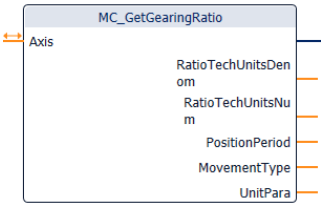
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.3.59 MC_GetGearingRatio

This instruction reads the electronic gear ratio, rotation cycle, and motion type parameters of the axis.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GetGearingRatio	Get the electronic gear ratio, rotation cycle, and motion type of the axis		:= MC_GetGearingRatio(Axis := , RatioTechUnitsDenom => , RatioTechUnitsNum => , PositionPeriod => , MovementType => , UnitPara =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
RatioTechUnitsDenom	Denominator of gear ratio	DWORD	Depends on data type	0	Denominator of the conversion factor for converting pulse unit to user unit.
RatioTechUnitsNum	Numerator of gear ratio	LREAL	Depends on data type	0	Numerator of the conversion factor for converting pulse unit to user unit.
PositionPeriod	Rotation cycle	LREAL	Depends on data type	0	Valid only in rotary mode.
MovementType	Motion type	INT	0, 1	0	0: Rotary mode. 1: Linear mode.
UnitPara	Unit conversion parameter	MC_UNITCONVERSION_PARAMETER	Depends on data type	-	Includes user display unit, reversal direction, number of instruction pulses per motor rotation, reducer use, worktable travel per rotation, gear ratio on the workpiece side, and gear ratio on the motor side.

- **Function**

  Bus-controlled axis:

  - This instruction is valid only for bus-controlled axes and local pulse axes.
  - The data type of UnitPara is MC_UNITCONVERSION_PARAMETER. The meanings of the structure members are shown in the table below.

Variable	Data Type	Description
eUnitDisplay	MC_UNITS	User display unit 0: pulse 1: mm 2: μm 3: nm 4: degree 5: inch
xInvertDirection	BOOL	If set to TRUE, the direction is reversed.
udiCmdPlsCountPerMotorRotation	UDINT	Number of command pulses per motor rotation
fWorkTravelDistancePerMotorRotation	LREAL	Worktable travel per rotation
xReducerUse	BOOL	TRUE when the reducer is used.
udiWorkSideGearRatio	UDINT	Gear ratio on the workpiece side
udiMotorSideGearRatio	UDINT	Gear ratio on the motor side

- This function block dynamically modifies the content in the axis configuration interface during program execution, as shown below.

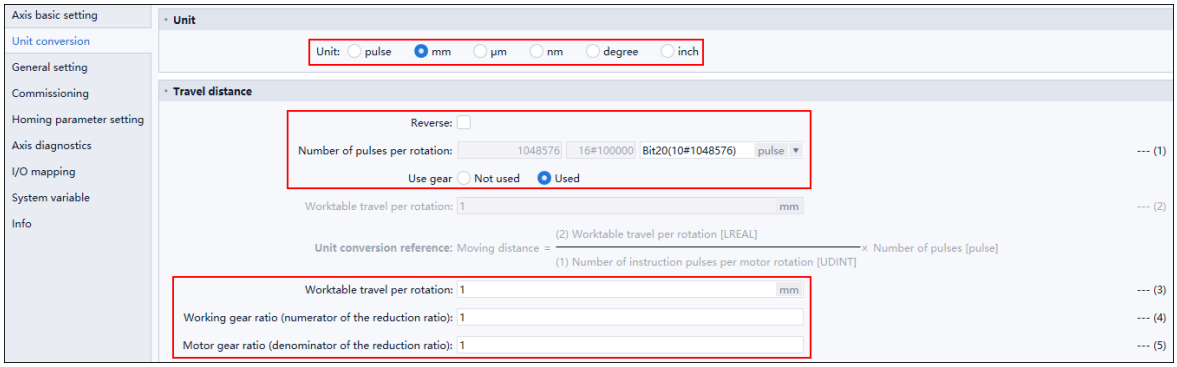


Figure 18-3-3-3-59-1 Unit conversion interface

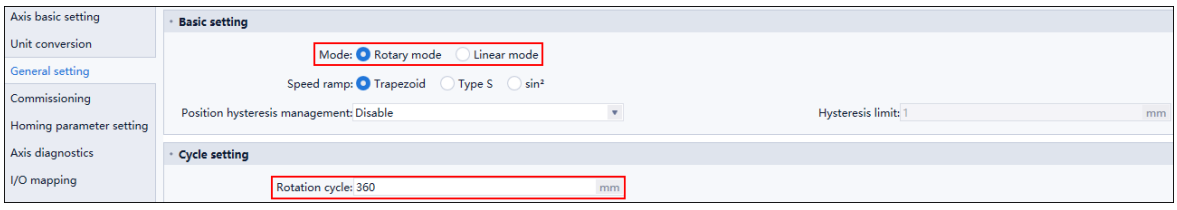


Figure 18-3-3-3-59-2 General setting interface

- During program execution, if the gear ratio related data is modified using the function block MC_ChangeGearingRatio or other methods, the data in the parameter UnitPara will not be updated and will remain consistent with the value set in the axis configuration interface.

Local pulse axis:

This instruction can be used for local pulse axes.

- **Faults**

This instruction is a function (FC). When the function return value is FALSE, it indicates that the "Axis" parameter is not an axis instance or the axis type is mismatched.

18.3.3.4 Cam

18.3.3.4.1 Instruction List

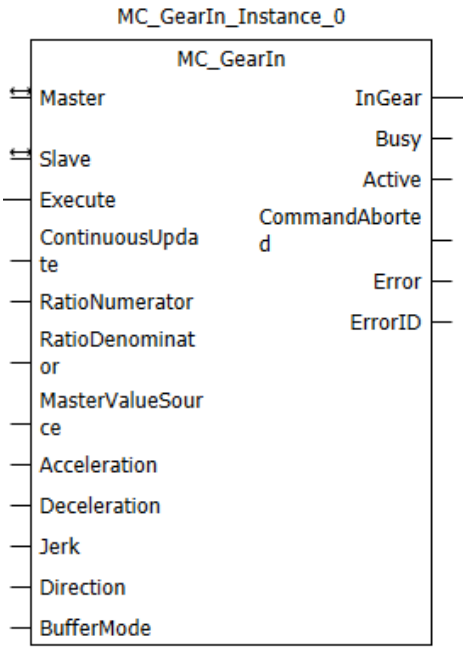
Instruction	Function	Library	Applicable Library Version
18.3.3.4.2 MC_GearIn	Start gearing	MotionControl	Library version 1.1.0.0 and later
18.3.3.4.3 MC_GearOut	Desynchronize gearing		Library version 1.1.0.0 and later
18.3.3.4.4 MC_Phasing	Master axis phase shifting		Library version 1.1.0.0 and later
18.3.3.4.5 MC_StopPhasing	Stop master axis phase shifting		Library version 1.1.0.0 and later
18.3.3.4.6 MC_GearInPos	Start gearing with specified positions		Library version 1.1.0.0 and later
18.3.3.4.7 MC_GetCamTablePhase	Get cam table phase		Library version 1.1.0.0 and later
18.3.3.4.8 MC_GetCamTableDistance	Get cam table displacement		Library version 1.1.0.0 and later
18.3.3.4.9 MC_DigitalCamSwitch	Enable digital cam switch		Library version 1.1.0.0 and later
18.3.3.4.10 MC_GenerateCamTable	Update the cam table with specified cam data		Library version 1.1.0.0 and later
18.3.3.4.11 MC_WriteCAM	Store the edited cam table as a file		Library version 1.1.0.0 and later
18.3.3.4.12 MC_ReadCAM	Read the cam data file into a cam table		Library version 1.1.0.0 and later
18.3.3.4.13 MC_CAMBounds	Calculate maximum position, velocity, and acceleration of the slave axis		Library version 1.1.0.0 and later
18.3.3.4.14 MC_CamIn	Start camming		Library version 1.1.0.0 and later
18.3.3.4.15 MC_CamInV2	Start camming		Library version 2.0.0.0 and later
18.3.3.4.16 MC_CamOut	Desynchronize camming		Library version 1.1.0.0 and later
18.3.3.4.17 MC_CombineAxes	Output the sum or difference of the position references of two axes as a new position reference		Library version 1.1.0.0 and later
18.3.3.4.18 MC_AxesObserve	Observe position deviation between master and slave axes		Library version 1.1.0.0 and later
18.3.3.4.19 MC_CheckCamTable	Check cam table node data		Library version 2.0.0.0 and later

Instruction	Function	Library	Applicable Library Version
18.3.3.4.20 MC_CamNodeBinding	Cam node binding		Library version 2.1.0.0 and later
18.3.3.4.21 MC_MoveLink	Synchronous positioning		Library version 1.1.0.0 and later
18.3.3.4.22 MC_PhasingEx	Complete phase shifting within specified master axis travel distance		Library version 1.1.0.0 and later

18.3.3.4.2 MC_GearIn

Start electronic gearing according to the specified gear ratio of the master axis to the slave axis.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GearIn	Start electronic gearing		MC_GearIn_Instance_0(Master :=, Slave :=, Execute :=, ContinuousUpdate :=, RatioNumerator :=, RatioDenominator :=, MasterValueSource :=, Acceleration :=, Deceleration :=, Jerk :=, Direction :=, BufferMode :=, InGear =>, Busy =>, Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Master	Master axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the master axis; an instance of AXIS_REF.
Slave	Slave axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the slave axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
ContinuousUpdate	Continuous update	BOOL	TRUE, FALSE	FALSE	TRUE: Modified parameter values are valid with no need for re-executing this instruction. FALSE: Modified parameter values are valid only after this instruction is re-executed.
RatioNumerator	Gear ratio numerator	DINT	Positive number or negative number	1	Specify the gear ratio numerator. A negative value indicates that the slave axis moves in the opposite direction of the master axis.
RatioDenominator	Gear ratio denominator	UDINT	Positive number	1	Specify the gear ratio denominator (range: 0–2^{31}).
MasterValueSource	Master axis data type	18.3.3.2.2.11 MC_REFERENCE_TYPE: Data Type Enumeration	0: mcCommand 1: mcFeedback 2: mcLaterCommand	0	0: Value calculated in the previous task cycle (fSavePosition). 1: Value obtained in the same task cycle (fActPosition). 2: Value calculated in the same task cycle (fSetPosition).
Acceleration	Acceleration	LREAL	Positive number	0	Specify the maximum acceleration [u/s^2] of the slave axis during the catching phase.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Deceleration	Deceleration	LREAL	Positive number	0	Specify the maximum deceleration [u/s^2] of the slave axis during the catching phase.
Jerk	Jerk	LREAL	0	0	Reserved parameter. It currently applies to the T-curve only.
Direction (reserved parameter for future function development)	Direction	18.3.3.2.2.9 MC_DIRECTION: Motion Direction Enumeration	1: DIR_POSITIVE 2: DIR_NEGATIVE 4: DIR_NODIRECTION	1	1: DIR_POSITIVE: The slave axis follows the master axis only when the master axis moves in the positive direction. The slave axis decelerates to stop according to settings when the master axis moves in the negative direction, at which point the InGear state is inactive. The InGear state is active when the master axis moves in the positive direction again. 2: DIR_NEGATIVE: Opposite of POSITIVE. 4: DIR_NODIRECTION: The slave axis follows the master axis regardless of the motion directions of the master axis. The InGear state remains latched once active.
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting 1: mcBuffered	0	0: Aborting. 1: Buffered.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
InGear	Gear ratio reached	BOOL	TRUE, FALSE	FALSE	Conditions for setting the variable to TRUE: 1. The velocity reference equals the target velocity. 2. Slave axis acceleration: Velocity greater than the target velocity. 3. Slave axis deceleration: Velocity less than the target velocity.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Active	Controlling	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

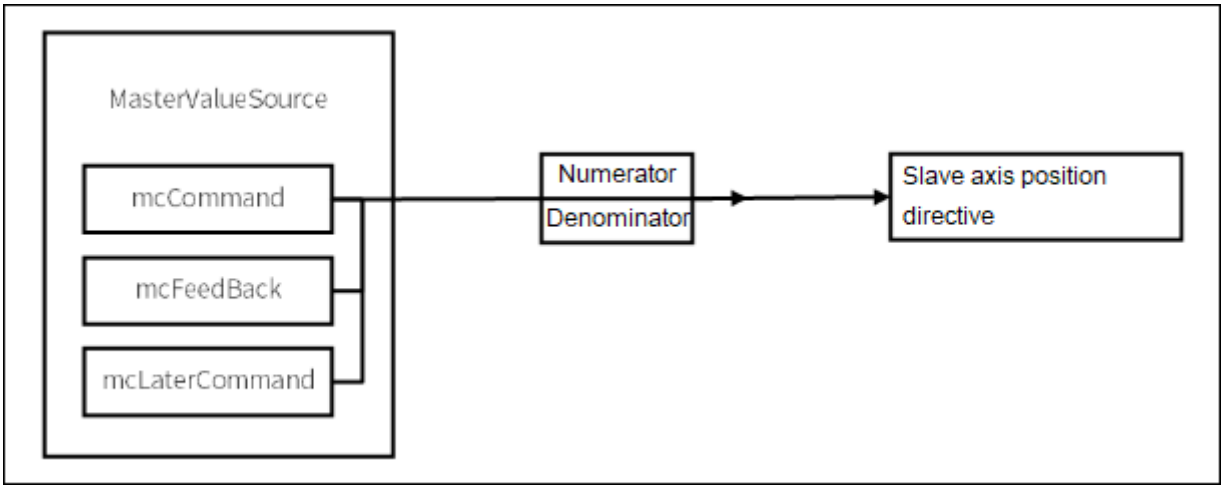
- **Parameter correction**

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Acceleration	Acceleration	Any acceleration that exceeds the motion parameter limit will be automatically corrected to the acceleration limit.	Any acceleration less than 0 will be automatically corrected to 10% of the acceleration limit.	Any acceleration equal to 0 will be automatically corrected to 10% of the acceleration limit.
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the deceleration limit.	Any deceleration less than 0 will be automatically corrected to 10% of the deceleration limit.	The deceleration equal to 0 will be automatically corrected to 10% of the deceleration limit.
Jerk	Jerk	If the motion profile is set to S-curve, any jerk that exceeds the software limit will be automatically corrected to the jerk limit.	If the motion profile is set to S-curve, any jerk less than 0 will be automatically corrected to 10% of the jerk limit.	If the motion profile is set to S-curve, the jerk equal to 0 will be automatically corrected to 10% of the jerk limit.

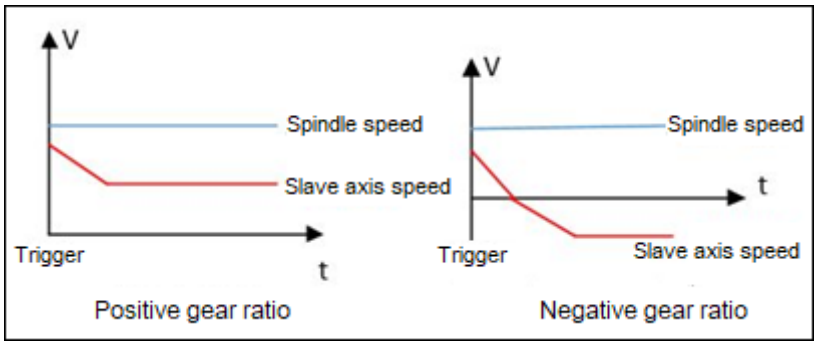
- **Function**

  Bus-controlled axis:

  - This instruction can be used to specify the slave axis as the target axis and specify the RatioNumerator, RatioDenominator, MasterValueSource, Acceleration, and Deceleration for gearing.
  - When the instruction is triggered on a rising edge of the input parameter Execute, the input parameters are checked for validity. If any of invalid parameters is detected, it will be automatically corrected by the function block without a fault message; any buffered instructions after this instruction will be cleared.



- After the corresponding instruction is started, the slave axis accelerates or decelerates to the target velocity, which is the product of the master axis velocity multiplied by the gear ratio.
- The period before reaching the target velocity is called the catching phase, and the period after reaching the target velocity is called the InGear phase.
- When the gear ratio is positive, the slave axis moves in the same direction as the master axis. When the gear ratio is negative, the slave axis moves in the opposite direction to the master axis. The relationships of the master and slave axis velocity with the gear ratio are shown in the following figure.



Relationships of the master and slave axis velocity with the gear ratio

- The master and slave axes specified in this instruction cannot be the same.
- This instruction starts electronic gearing on a rising edge of the parameter Execute.
- This instruction can be used in combination with other synchronous control instructions. Its in-out variable Master can be the master or slave axis of other synchronous control instructions; its in-out variable Slave can serve as the master axis of other synchronous control instructions.
- If bus cycle is specified for the cycle task of the master axis, the velocity of the slave axis may show a sudden jump after synchronization. It is recommended that users specify EtherCAT cycle for the cycle task of both the master and slave axes before enabling a synchronization instruction.
- MC_Home, MC_Homing, or MC_HomingWithParameter can be used during the instruction execution.

  This instruction places the master axis in the AXIS_HOMING state. If the instruction is executed while the master axis is already in this state, the signal source "MasterValueSource" must be set to "mcLatestCommand"; otherwise, the slave axis will overspeed.

Software limit:

- During the gearing with specified positions, if the slave axis position is beyond the software limit, this instruction will be aborted, and the slave axis will decelerate to 0 within one cycle with the axis state switched from AXIS_SYNCHRONIZED_MOTION to AXIS_ERRORSTOP.

Position type selection:

You can specify the MasterValueSource for synchronizing the slave axis based on the following options.

- mcCommand: Commanded position (Value calculated in the previous task cycle, fSavePosition)

  For the current cycle, use the master axis position calculated in the previous task cycle.

- mcFeedback: Actual position (Value obtained in the same task cycle, fActPosition).

  Use the actual master axis position obtained in the same task cycle.

- mcLaterCommand: Commanded position (Value calculated in the same task cycle, fSetPosition).

  Use the master axis position calculated in the same task cycle.

- When the master axis starting electronic gearing is specified as a virtual axis, the value of the input variable MasterValueSource cannot be set to 1 (mcFeedback); otherwise, a fault will occur.

Local pulse axis:

- This instruction can be used for local pulse axes.
- When the local pulse axis is used as the master axis with Y0 or Y1 single-ended output, it is not recommended to use this instruction. ① No pulse output; ② No position feedback.

Re-execution of motion control instructions:

- If this instruction is activated (Active = TRUE), re-executing it in mcBuffered mode will result in a fault, while re-triggering in mcAborting mode will allow it to run normally.
- If this instruction is not activated (Active = FALSE), it can be re-executed after its input parameters are modified.

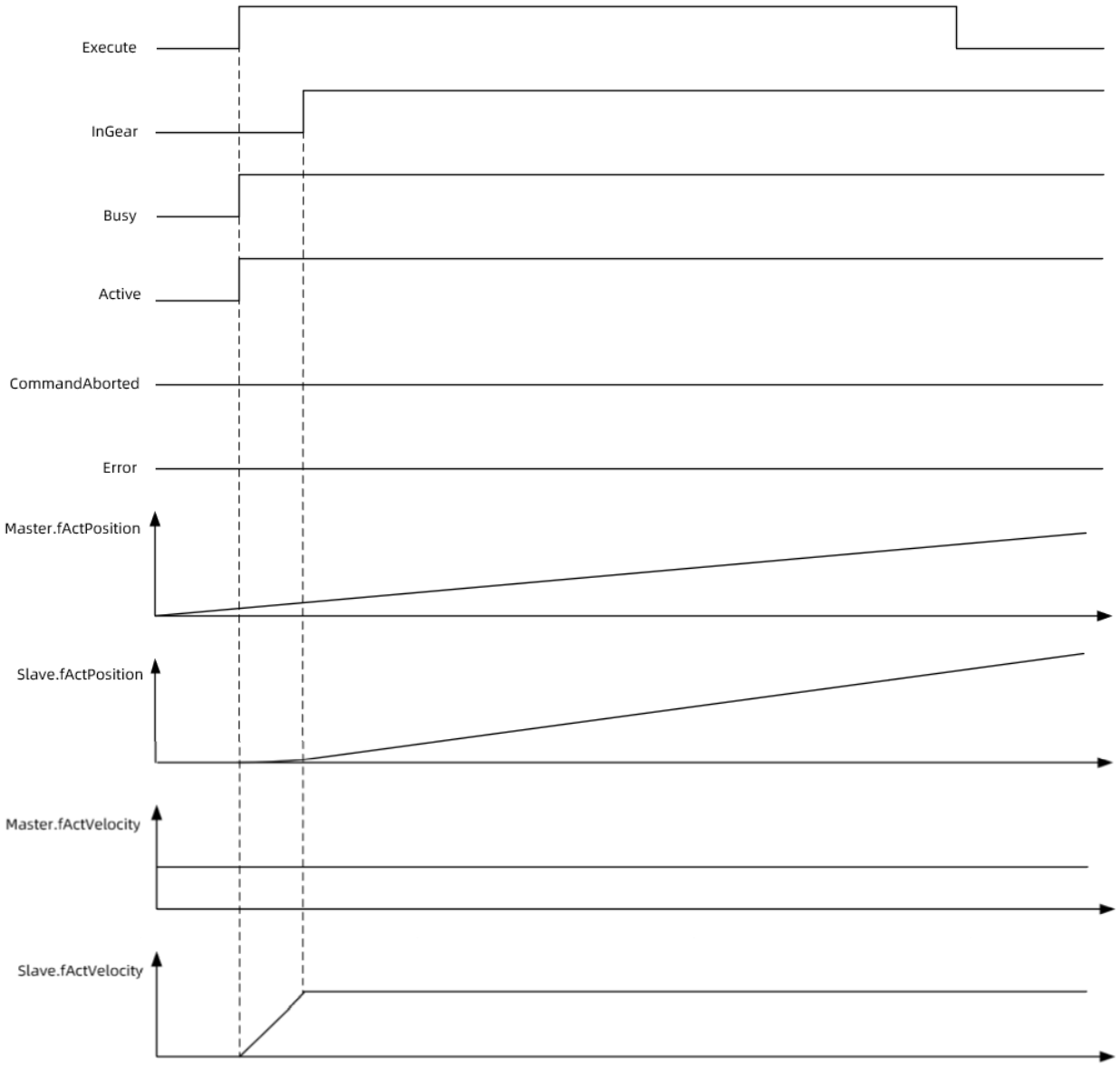
- **Timing diagram**

  When the parameter “Execute” is activated, the parameter “Busy” shows the value TRUE. After this instruction is activated, the parameter “Active” shows the value TRUE.

  When the target velocity is reached, the output parameter “InGear” shows the value TRUE.

  When this instruction is aborted by another instruction, the parameter “CommandAborted” shows the value TRUE, and the parameters “Busy”, “Active”, and “InGear” show the value FALSE.

  To terminate electronic gearing mid-way, use the MC_GearOut or MC_Stop instruction.



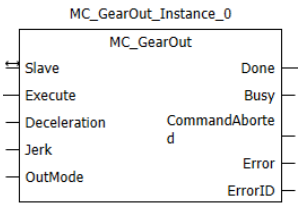
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.4.3 MC_GearOut

This instruction desynchronizes electronic gearing between the master and slave axes triggered by the MC_GearIn or MC_GearInPos instruction.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GearOut	Desynchronize gearing		MC_GearOut_Instance_0(Slave :=, Execute :=, Deceleration :=, Jerk :=, OutMode :=, Done =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Slave	Slave axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the slave axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
Deceleration	Deceleration	LREAL	Positive number	0	Specify the maximum deceleration [u /s^2] of the slave axis when the electronic gearing between the master and slave axes is desynchronized.
Jerk	Jerk	LREAL	Positive number or 0	0	Slave axis jerk parameter.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
OutMode	Desynchronization mode	18.3.3.2.3.7 MC_OUTMODE: Desynchronization Mode Enumeration	0: mcDeceleration Stop 1: mcConstantVelocity	0	0: Deceleration stop. 1: Constant velocity before desynchronization.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the deceleration limit.	Any deceleration less than 0 will be automatically corrected to 10% of the deceleration limit.	The deceleration equal to 0 will be automatically corrected to 10% of the deceleration limit.
Jerk	Jerk	If the motion profile is set to S-curve, any jerk that exceeds the software limit will be automatically corrected to the jerk limit.	If the motion profile is set to S-curve, any jerk less than 0 will be automatically corrected to 10% of the jerk limit.	If the motion profile is set to S-curve, the jerk equal to 0 will be automatically corrected to 10% of the jerk limit.

- **Function**

Bus-controlled axis:

- This instruction can be used to specify the slave axis as the target axis, specify the OutMode, Deceleration, and Jerk, and abort the executing MC_GearIn or MC_GearInPos instruction.
- This instruction is executed on a rising edge of the input variable Execute.
- This instruction does not affect the operation of the master axis specified by the MC_GearIn or MC_GearInPos instruction.
- This instruction must be executed after the slave axis has executed the MC_GearIn or MC_GearInPos instruction. A fault occurs when this instruction is executed on an axis that does not start gearing.
- During the execution of this instruction, the MC_MoveSuperImposed instruction cannot be used. When the MC_MoveSuperImposed instruction is triggered, CommandAborted is set for the MC_MoveSuperImposed instruction.

Desynchronization mode:

- You can specify the mode by which the slave axis disengages from the master axis by defining the value of the input variable OutMode.
- When the OutMode is set to 0, the slave axis decelerates to "0" according to the Deceleration specified in this instruction. In the deceleration stop mode, the slave axis state switches from SynchronizedMotion to DiscreteMotion, and becomes StandStill when its velocity reaches 0.
- When the OutMode is set to 1, the slave axis continues to move at a constant velocity while the gearing stops, and switches from SynchronizedMotion to ContinuousMotion state.

Local pulse axis:

- This instruction can be used for local pulse axes.

Re-execution of motion control instructions:

- This instruction can be re-executed.
- Do not modify the value of OutMode when this instruction is re-executed.
- When the OutMode is set to 0 (mcDecelerationStop), and this instruction is re-triggered after modifying the Deceleration and Jerk values, the slave axis disengages from the master axis in the deceleration stop mode according to the modified parameter values.
- If either of the modified Deceleration and Jerk values is invalid, it will be automatically corrected by the function block without a fault message. The slave axis velocity will decelerate to 0 within one cycle, with the axis state switching from ContinuousMotion to ErrorStop.

Software limit:

- When the slave axis position is beyond the software limit while disengaging from the master axis, this instruction is aborted, and the slave axis velocity decelerates to 0 within one cycle. In the mcDecelerationStop mode, the slave axis state switches from DiscreteMotion to ErrorStop. In the mcConstantVelocity mode, the slave axis state switches from ContinuousMotion to ErrorStop.

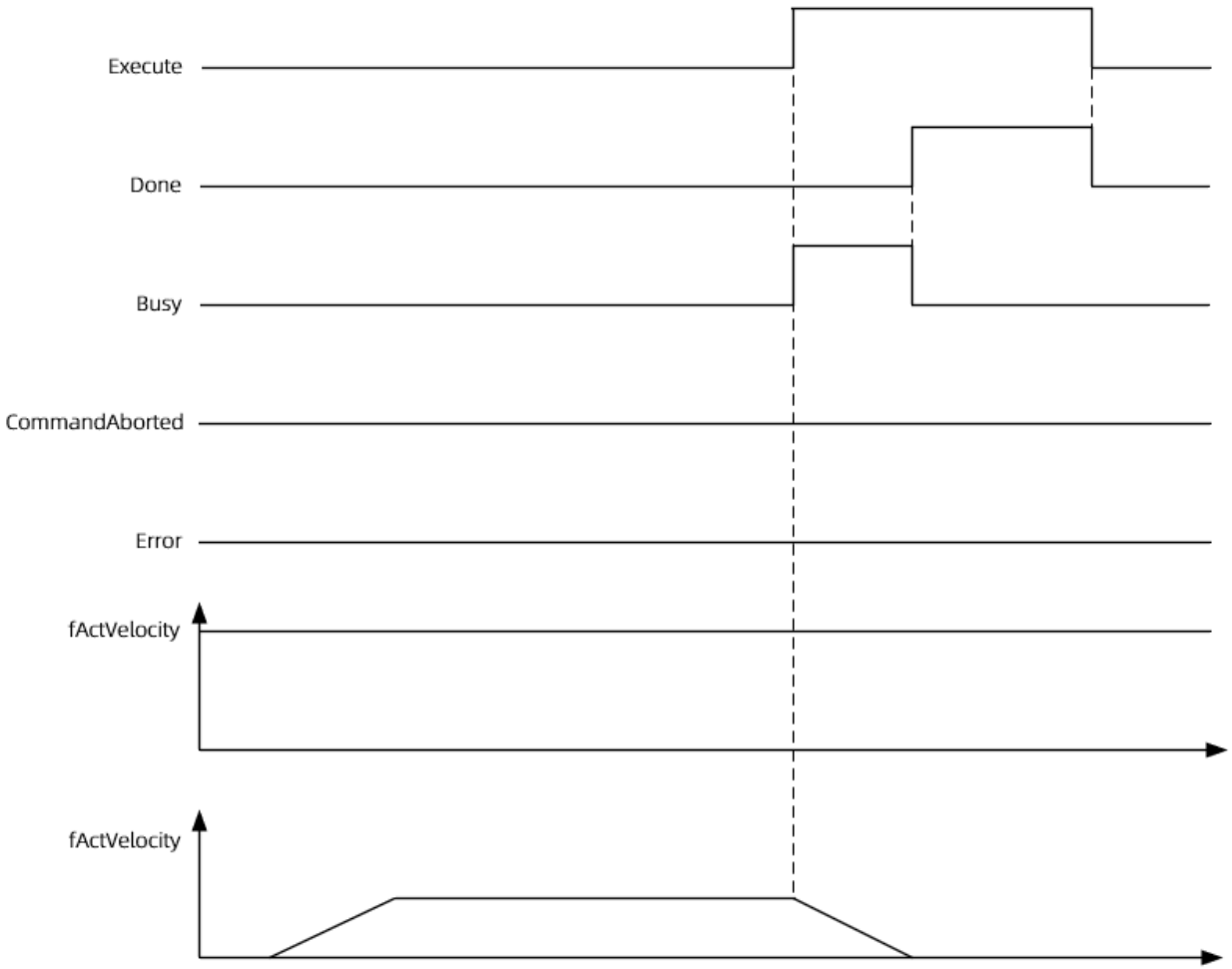
- **Precautions**
  - When the OutMode is set to 1 (mcConstantVelocity) and the instruction is executed, the parameter Done is set to TRUE, and the parameter Busy remains TRUE until this instruction is aborted by next instruction, at which point the parameter Busy is set to FALSE. Therefore, to stop the slave axis in the mcConstantVelocity mode, this instruction must be used in conjunction the MC_Stop instruction.
  - During the execution of this instruction, if a fault occurs on the slave axis, the command aborted flag will be set, and the slave axis will stop immediately.

- This instruction can abort an executing MC_MoveSuperImposed instruction.
- If the MC_MoveSuperImposed instruction is executed during the execution of this instruction, the OutMode = 0 (mcDecelerationStop) will cause the MC_MoveSuperImposed instruction to abort directly.
- When the OutMode is set to 0 (mcDecelerationStop) or 1 (mcConstantVelocity) by this instruction, the MC_SetOverride (Change the motion parameter scaling factor) instruction is disabled.

- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates the instruction has been executed successfully; the parameter Busy indicates that the current function block is being executed; the parameter CommandAborted indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

  This instruction is executed on a rising edge of the parameter Execute to end electronic gearing, with the output parameter Busy showing the value TRUE. After electronic gearing ends completely, the output parameter Done shows the value TRUE. If the OutMode is set to mcDecelerationStop, the parameter Busy shows the value FALSE. If the OutMode is set to mcConstantVelocity, the parameter Busy remains the value TRUE. The following is the timing diagram when the OutMode is set to mcDecelerationStop.



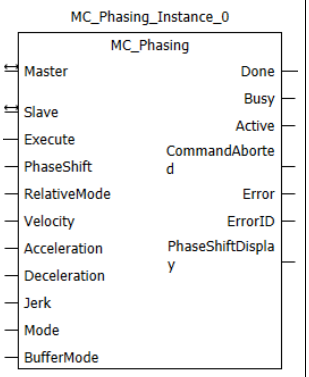
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.4.4 MC_Phasing

This instruction starts master axis phase shifting according to the phase shift reference.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_Phasing	Master axis phase shifting		MC_Phasing_Instance_0(Master :=, Slave :=, Execute :=, PhaseShift :=, RelativeMode :=, Velocity :=, Acceleration :=, Deceleration :=, Jerk :=, Mode :=, BufferMode :=, Done =>, Busy =>, Active =>, CommandAborted =>, Error =>, ErrorID =>, PhaseShiftDisplay =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Master	Master axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the master axis; an instance of AXIS_REF.
Slave	Slave axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the slave axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
PhaseShift	Phase shift	LREAL	-	0	Phase shift [u] between the master and slave axes. Positive values indicate that the slave axis leads the master axis.
RelativeMode	Phase shifting mode	BOOL	TRUE, FALSE	FALSE	FALSE: Specify PhaseShift to be an absolute phase shift. TRUE: Specify PhaseShift to be the original phase shift plus an additional phase.
Velocity	Target velocity	LREAL	Positive number	0	Specify the target velocity [u/s] for phase shifting.
Acceleration	Target acceleration	LREAL	Positive number	0	Specify the target acceleration [u/s^2] for phase shifting.
Deceleration	Target deceleration	LREAL	Positive number	0	Specify the target deceleration [u/s^2] for phase shifting.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Jerk	Jerk	LREAL	Positive number or 0	0	Specify the target jerk [u/s^3] for phase shifting. 0: T-curve. Non-zero: S-curve.
Mode	Compensation mode	INT	0, 1	0	0: Phase shifting continues when the master axis velocity is 0. 1: Phase shifting stops when the master axis velocity is 0, with the slave axis performing an immediate stop; phase shifting resumes when the master axis moves again.
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting	0	0: Aborting. (Reserved for other modes)

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Controlling	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.
PhaseShiftDisplay	Phase shift display	LREAL	-	0	Real-time display of the master axis phase shift [u].

- **Parameter correction**

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity	Velocity	Any velocity that exceeds the motion parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	The velocity equal to 0 will be automatically corrected to 1e-5.
Acceleration	Acceleration	Any acceleration that exceeds the motion parameter limit will be automatically corrected to the acceleration limit.	Any acceleration less than 0 will be automatically corrected to 10% of the acceleration limit.	Any acceleration equal to 0 will be automatically corrected to 10% of the acceleration limit.
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the deceleration limit.	Any deceleration less than 0 will be automatically corrected to 10% of the deceleration limit.	The deceleration equal to 0 will be automatically corrected to 10% of the deceleration limit.

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Jerk	Jerk	If the motion profile is set to S-curve, any jerk that exceeds the software limit will be automatically corrected to the jerk limit.	If the motion profile is set to S-curve, any jerk less than 0 will be automatically corrected to 10% of the jerk limit.	If the motion profile is set to S-curve, the jerk equal to 0 will be automatically corrected to 10% of the jerk limit.

- **Function**

  Bus-controlled axis:

  - When this instruction is executed in the synchronous control mode, the master axis phase shifting is started based on the PhaseShift, Velocity, Acceleration, and Deceleration references.
  - This instruction is triggered on a rising edge of the input parameter Execute.
  - This instruction must be used during the execution of the synchronous control instructions. If it is triggered in a non-synchronous control state, a fault will occur.
  - If this instruction reports a fault, any buffered instructions after this instruction will be cleared.
  - Once a synchronous control instruction is completed, any incomplete phase shifting will stop and a fault will occur.
  - You can shift the phase of the master axis for the following synchronous control instructions: MC_CamIn, MC_GearIn, MC_GearInPos, and MC_MoveLink.

  PhaseShift

  - This parameter is used to specify the master axis phase shift relative to the slave axis.
  - Positive PhaseShift values indicate that the slave axis position leads the master axis position.

  Velocity, Acceleration, Deceleration, Jerk

  - The target velocity, target acceleration, target deceleration, and target jerk for phase shifting are specified through the input parameters Velocity, Acceleration, Deceleration, and Jerk.
  - If the Velocity, Acceleration, and/or Deceleration are set to values less than or equal to 0, they will be automatically corrected by the function block without any fault message.
  - If the input value of Jerk is not equal to 0 and the slave axis motion profile is set to T-curve, the Jerk value will not be valid. If the slave axis motion profile is set to S-curve, the Jerk value will be checked for validity, and any input values less than or equal to 0 will be automatically corrected by the function block without any fault message.

  Phase shifting mode (RelativeMode)

  Set the phase shifting mode by defining the input variable RelativeMode.

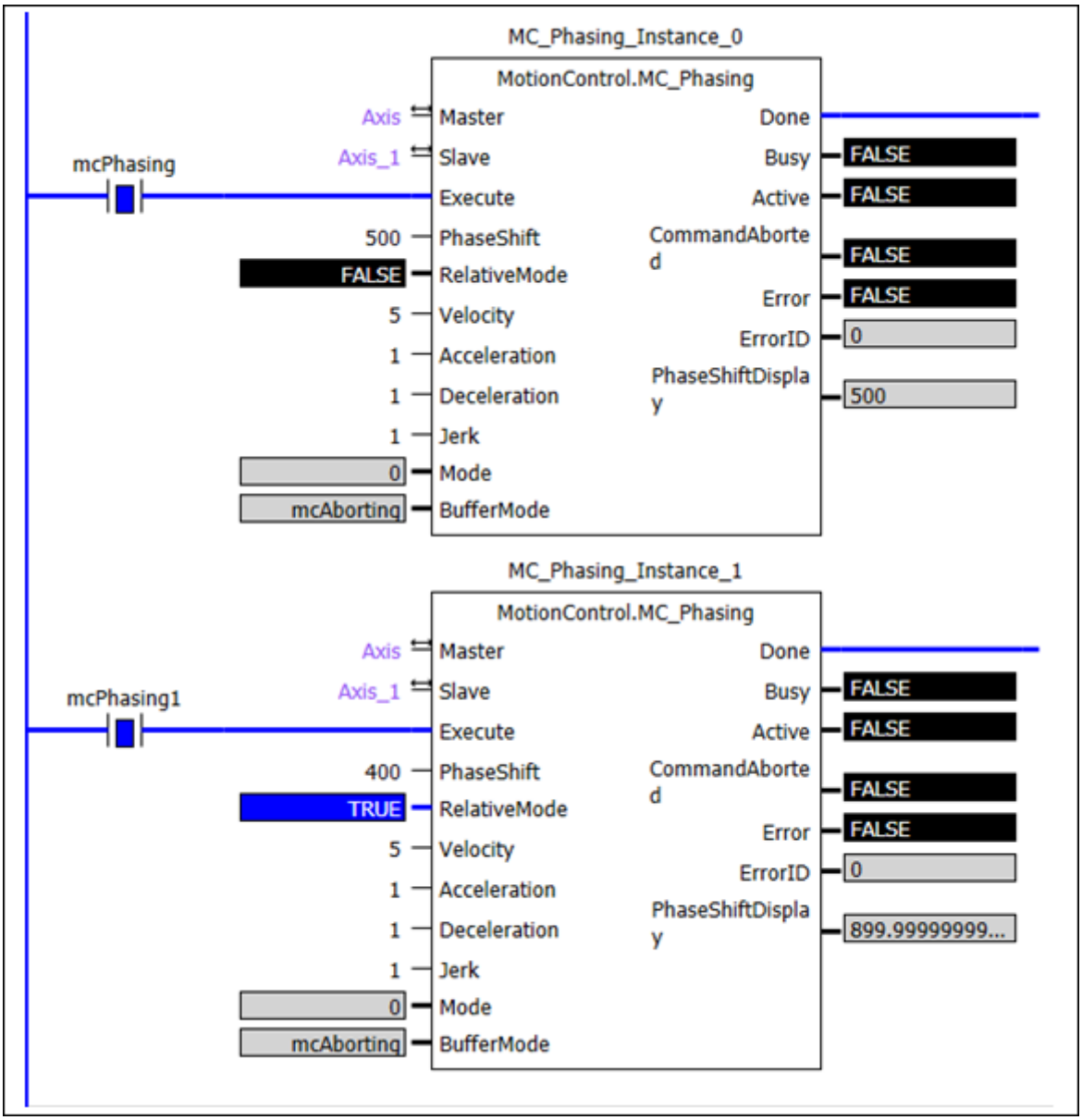
  FALSE: Specify PhaseShift to be an absolute phase shift.

  TRUE: Specify PhaseShift to be the original phase shift plus an additional phase.

  When multiple instantiated MC_Phasing instructions are executed sequentially or the same MC_Phasing instruction is executed multiple times, phase addition is performed cumulatively, which means that the previously added phases affect the phase addition specified by subsequent instantiated MC_Phasing instructions. The instances are as follows.

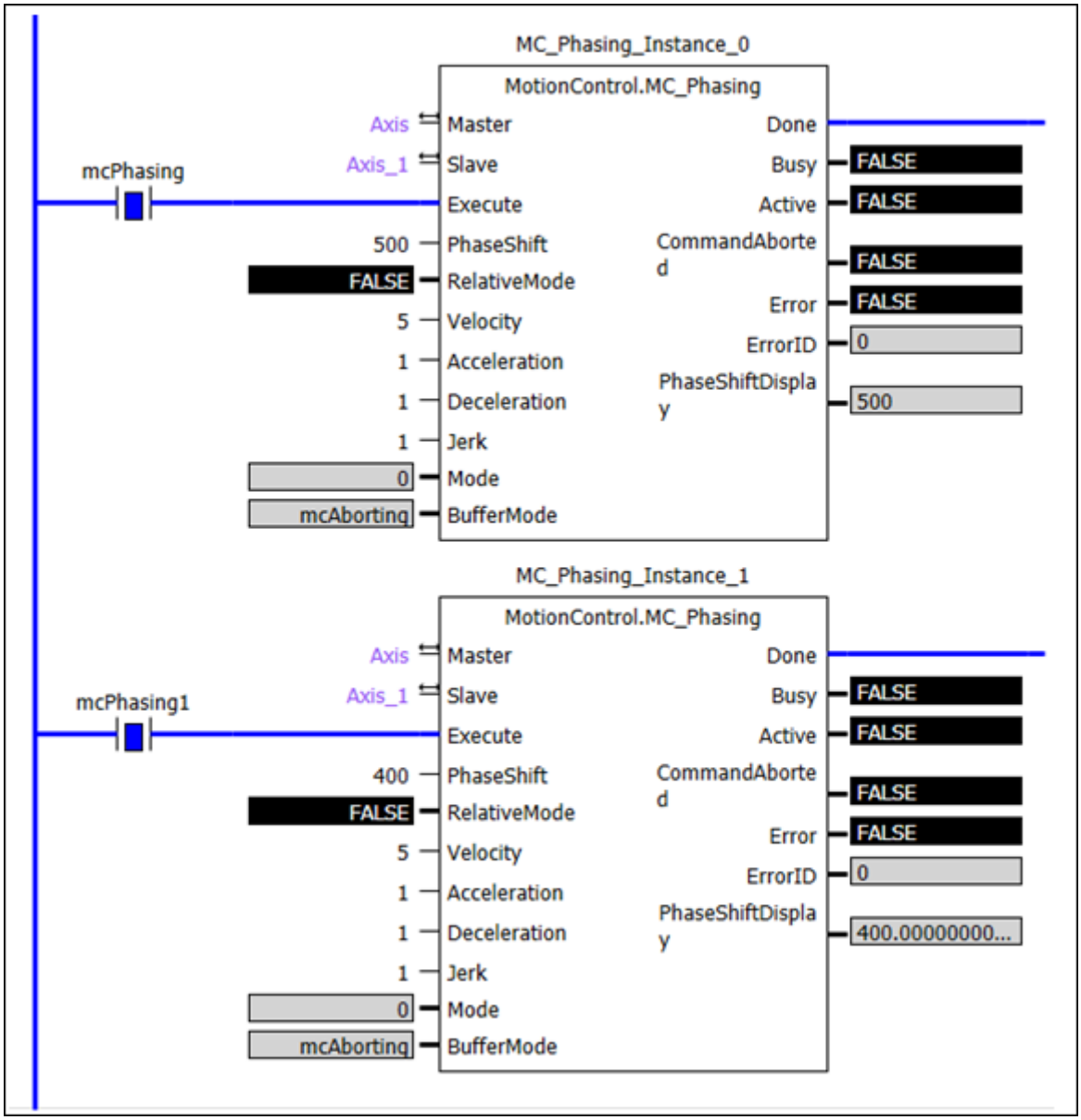
  For example:

① When MC_Phasing_Instance_0 is executed with RelativeMode = FALSE and PhaseShift = 500, phase addition is performed; when MC_Phasing_Instance_1 is executed with RelativeMode = TRUE and PhaseShift = 400 after the phase addition specified by MC_Phasing_Instance_0, phase addition is performed again.



That is, when MC_Phasing_Instance_1 is executed with RelativeMode = TRUE, the PhaseShift specified in this instruction is added on top of the previously added phases.

② When MC_Phasing_Instance_0 is executed with RelativeMode = FALSE and PhaseShift = 500, phase addition is performed; when MC_Phasing_Instance_1 is executed with RelativeMode = FALSE and PhaseShift = 400 after the phase addition specified by MC_Phasing_Instance_0, phase addition is performed again.



Regardless of how many phases have been added based on previous instructions, the finally executed MC_Phasing_Instance_1 with RelativeMode = FALSE results in an absolute phase addition of 400 relative to the master axis.

Phase shifting mode (Mode)

You can set the shifting mode by defining the input variable Mode for this instruction.

Mode = 0: Phase shifting continues when the master axis velocity is 0.

Mode = 1: When the instruction is executed and the master axis velocity is 0, both Busy and parameter Actives are set for the instruction with no phase shifting performed; during the instruction execution, if the master axis velocity is not 0, phase shifting is started; when the master axis velocity drops to 0, the slave axis velocity decelerates to 0 within one cycle, causing the end of phase shifting, and phase shifting resumes when the master axis velocity is not 0 again.

Re-execution of motion control instructions

- This instruction can be re-executed.
- If this instruction is re-executed after input variables such as PhaseShift, Velocity, Acceleration, Deceleration, and Jerk are modified, it will be executed with the modified parameter values.
- If any of the modified parameter values is invalid, it will be automatically corrected by the function block with no fault message.

Software limit

During phase shifting, if the slave axis position is beyond the software limit, the slave axis will decelerate to 0 within one cycle with the state switching from SynchronizedMotion to ErrorStop, and the corresponding synchronous control instruction for the slave axis will be aborted.

Local pulse axis:

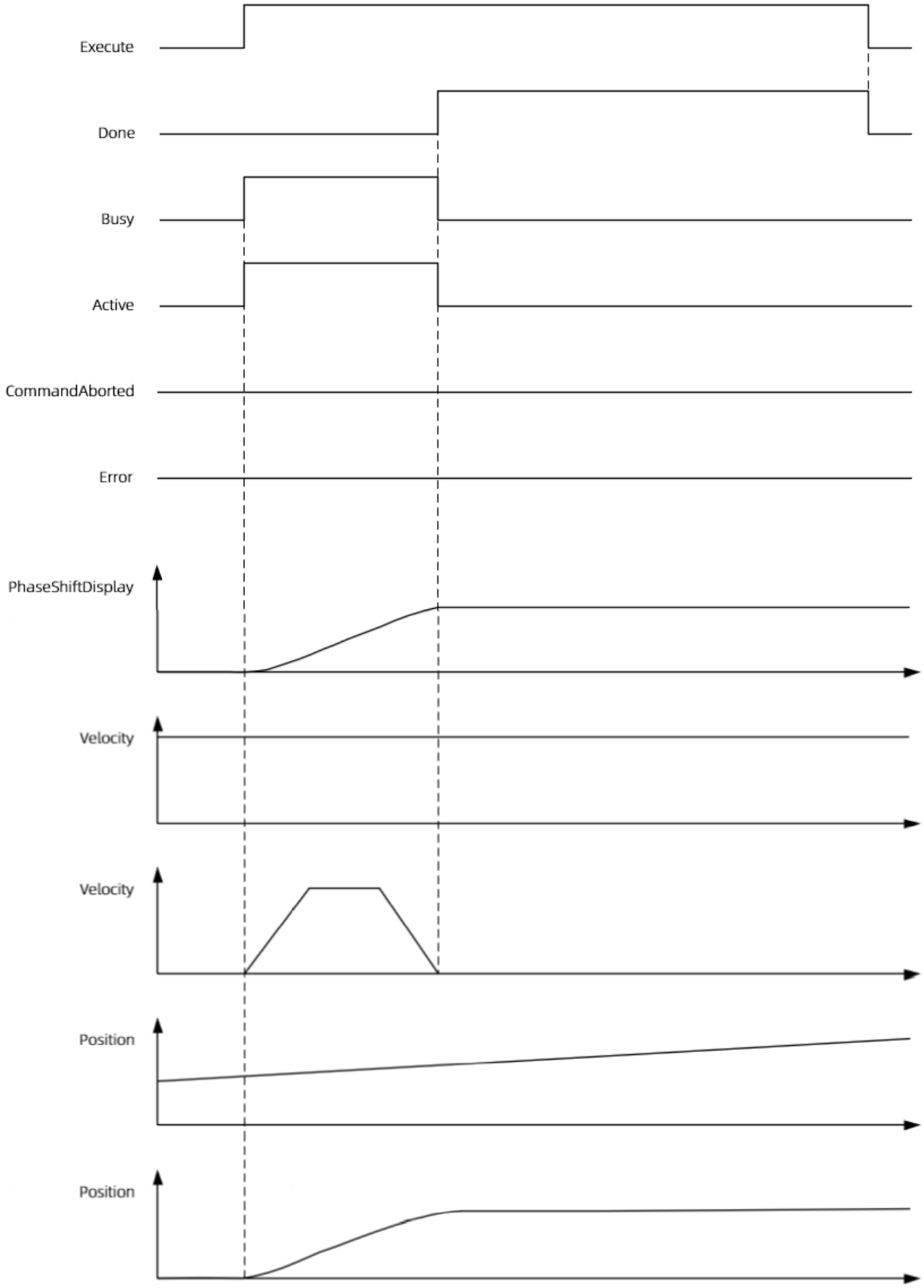
This instruction can be used for local pulse axes.

- **Precautions**
  - During the execution of this instruction, it is prohibited to execute the MC_SetPosition instruction on the master axis.
  - Phase shifting start time:
    - MC_GearIn instruction: After the output variable InGear is set.
    - MC_GearInPos instruction: After the output variable InSync is set.
    - MC_CamIn instruction: After the output variable InSync is set.
  - If the sum of the target velocity specified in this instruction plus the master axis velocity exceeds the maximum velocity reference for the master axis set in the software, the instruction profile will remain unchanged.
  - A fault will occur if the phase shifting has not yet ended after the synchronization instruction is executed.
- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates the instruction has been executed successfully; the parameter Busy indicates that the current function block is being executed; the parameter CommandAborted of the function block indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

  Phase shifting is triggered on a rising edge of the input parameter Execute. The slave axis automatically calculates a smooth curve to complete the phase shift relative to the master axis. The phase shift between the master and slave axes is the value of the input signal PhaseShift, with positive values indicating that the slave axis lags behind the master axis. The phase shift between the master and slave axes is displayed as the value of the output signal PhaseShiftDisplay. When the PhaseShiftDisplay value is equal to the PhaseShift value, the output parameter Done is set to TRUE.

  After the adjustment, the phase shift between the slave and master axes is the value set for "PhaseShift".



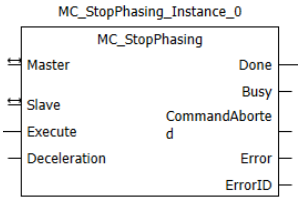
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.4.5 MC_StopPhasing

This instruction stops the phase shifting corresponding to the MC_Phasing and MC_PhasingEx instructions, without affecting the cam or gear motion of the master and slave axes.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_StopPhasing	Stop master axis phase shifting		MC_StopPhasing_Instance_0(Master :=, Slave :=, Execute :=, Deceleration :=, Done =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Master	Master axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the master axis; an instance of AXIS_REF.
Slave	Slave axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the slave axis; an instance of AXIS_REF.

  Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Deceleration	Target deceleration	LREAL	Positive number or 0	0	Specify the target deceleration [u/s^2] when the phase shifting stops. If it is set to 0, phase shifting stops immediately.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the deceleration limit.	Any deceleration less than 0 will be automatically corrected to 10% of the deceleration limit.	No correction.

- **Function**

  Bus-controlled axis:

  - This instruction stops the phase shifting.
  - This instruction is executed on a rising edge of the input variable Execute.
  - This instruction must be used during the execution of a synchronous control instruction. A fault will occur if this instruction is executed in a non-synchronous control state.
  - If this instruction reports a fault, any buffered instructions after this instruction will be cleared.
  - In case of no phase shifting between the current master and slave axes, this instruction will be called with the output variable of Done being set immediately.

  Deceleration

  - You can specify the method of stopping phasing by specifying the value of the input variable Deceleration.
  - When the Deceleration is set to 0, phase shifting stops immediately.
  - If the phase shift between the current master and slave axes is 0, this instruction will be called with the output variable Done being set immediately.
  - A too small Deceleration may result in the total phase shift after stopping phasing being greater than the original total phase shift reference.
  - If the Deceleration is set to less than 0, a fault will occur.

  Re-execution of motion control instructions

  - This instruction can be re-executed.
  - If the value of Deceleration is modified and the instruction is re-executed, the instruction will be executed with the modified parameter value.
  - If the modified value of Deceleration is 0 and the instruction is re-executed, the slave axis velocity will promptly decelerate to 0, with the output variable Done being set immediately.

  Software limit

  When phase shifting stops, if the slave axis position is beyond the software limit, the slave axis will decelerate to 0 within one cycle with the state switching from SynchronizedMotion to ErrorStop, and the corresponding synchronous control instruction for the slave axis will be aborted.

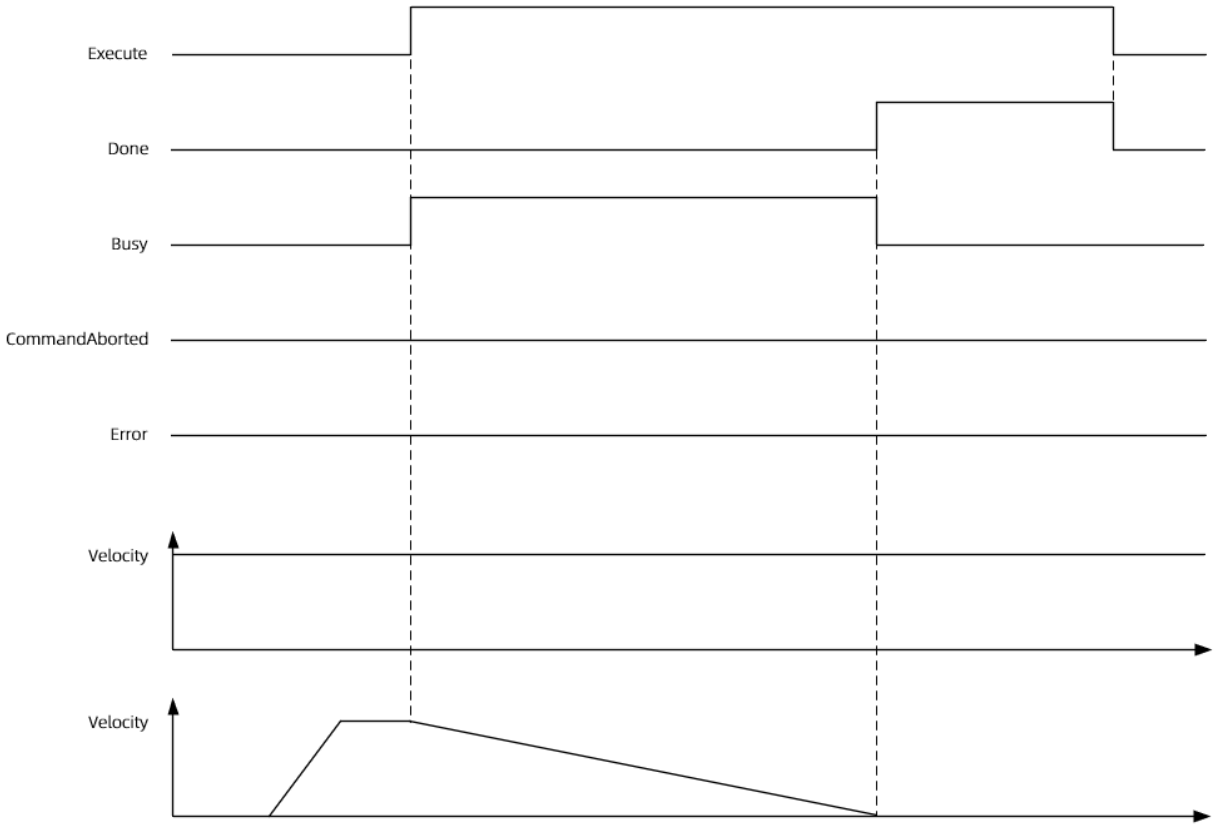
Local pulse axis:

This instruction can be used for local pulse axes.

- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates the instruction has been executed successfully; the parameter Busy indicates that the current function block is being executed; the parameter CommandAborted of the function block indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

  Phase shifting stops on a rising edge signal of the input parameter Execute. When the velocity for phase shifting decelerates to 0, the output parameter Done is set to TRUE.



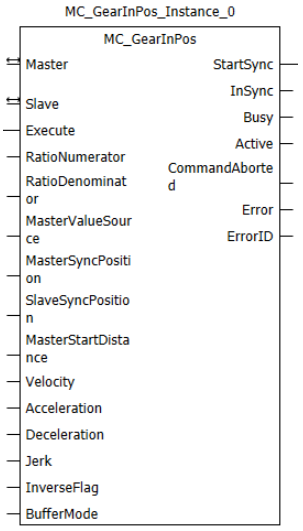
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.4.6 MC_GearInPos

This instruction performs electronic gearing when the gear ratio between the master and slave axes is established at the specified synchronization positions.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GearInPos	Start gearing at specified position		MC_GearInPos_Instance_0(Master :=, Slave :=, Execute :=, RatioNumerator :=, RatioDenominator :=, MasterValueSource :=, MasterSyncPosition :=, SlaveSyncPosition :=, MasterStartDistance :=, Velocity :=, Acceleration :=, Deceleration :=, Jerk :=, InverseFlag :=, BufferMode :=, StartSync =>, InSync =>, Busy =>, Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Master	Master axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the master axis; an instance of AXIS_REF.
Slave	Slave axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the slave axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
RatioNumerator	Gear ratio numerator	DINT	Positive number or negative number	1	Specify the numerator of the gear ratio. Negative values indicate that the slave axis moves in the opposite direction of the master axis. An input value of 0 is automatically corrected to 1.
RatioDenominator	Gear ratio denominator	UDINT	Positive number	1	Specify the denominator of the gear ratio. Any input value of less than 0 is automatically corrected to 1.
MasterValueSource	Master axis data type	18.3.3.2.2.11 MC_REFERENCE_TYPE: Data Type Enumeration	0: mcCommand 1: mcFeedback 2: mcLaterCommand	0	0: Value calculated in the previous task cycle (fSavePosition). 1: Value obtained in the same task cycle (fActPosition). 2: Value calculated in the same task cycle (fSetPosition).
MasterSyncPosition	Master axis synchronization position	LREAL	Positive number, negative number, or 0	0	Absolute position [u] of the master axis when the gear ratio is established.
SlaveSyncPosition	Slave axis synchronization position	LREAL	Positive number, negative number, or 0	0	Absolute position [u] of the slave axis when the gear ratio is established.
MasterStartDistance	Master axis leading distance	LREAL	Positive number, negative number, or 0	0	Travel distance [u] of the master axis after entering the catching phase. If the input value is 0, the slave axis immediately enters the catching phase, and StartSync is set to TRUE.
Velocity (reserved)	Velocity	LREAL	Positive number	0	Specify the maximum velocity [u/s] of the slave axis during the catching phase.
Acceleration (reserved)	Acceleration	LREAL	Positive number	0	Specify the maximum acceleration [u/s^2] of the slave axis during the catching phase.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Deceleration (reserved)	Deceleration	LREAL	Positive number	0	Specify the maximum deceleration [u/s^2] of the slave axis during the catching phase.
Jerk (reserved)	Jerk	LREAL	0	0	Reserved parameter.
InverseFlag	Inverse enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enable the slave axis to run reversely during the catching phase. FALSE: Disable the slave axis to run reversely during the catching phase.
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting	0	0: Aborting.

Output variables

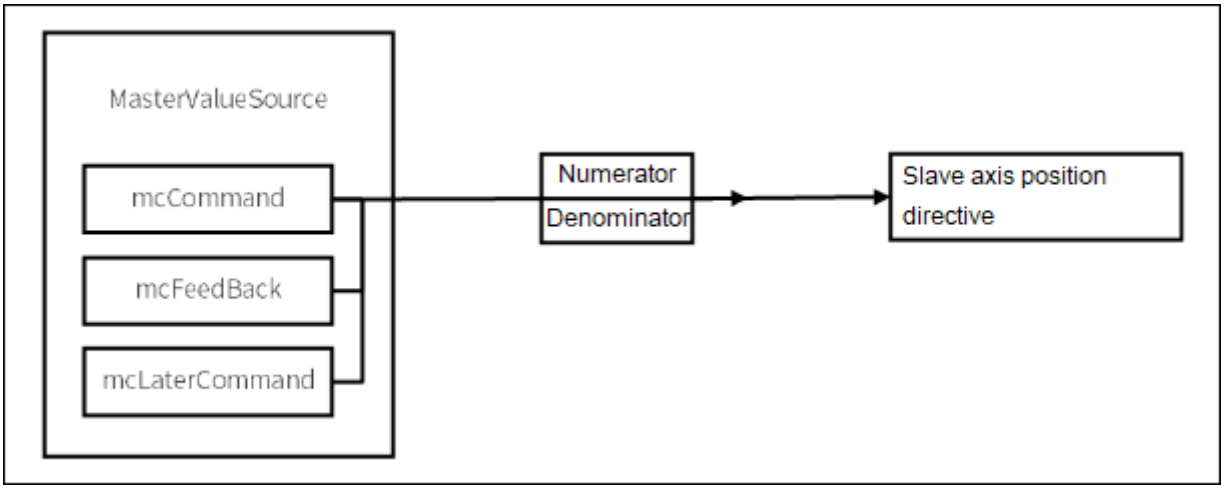
Output Variable	Name	Data Type	Valid Range	Default Value	Description
StartSync	Start synchronization	BOOL	TRUE, FALSE	FALSE	TRUE when the master and slave axes enter the catching phase, that is, when the master axis arrives at the (MasterSyncPosition - MasterStartDistance) position.
InSync	In synchronization	BOOL	TRUE, FALSE	FALSE	TRUE when the master and slave axes reach the specified synchronization positions and the specified gear ratio.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Active	Controlling	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

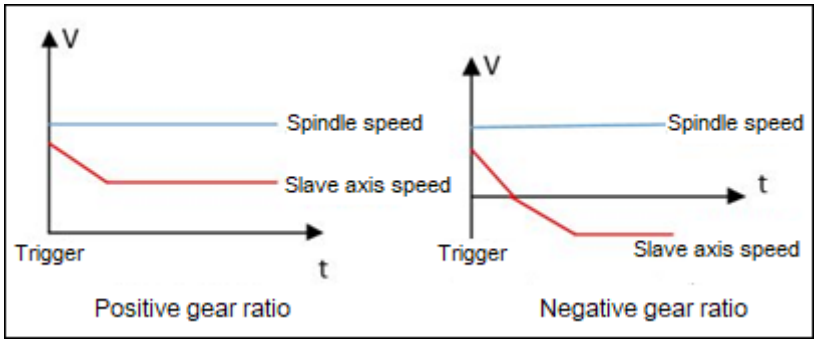
- **Function**

  Bus-controlled axis:

  - This instruction can be used to specify the slave axis as the target axis and specify the RatioNumerator, RatioDenominator, MasterValueSource, Acceleration, and Deceleration for gearing.



  - When the instruction is triggered on a rising edge of the input parameter Execute, the input parameters are checked for validity. If any of invalid parameters is detected, the function block will report a fault, abort the active motion, and clear any buffered instructions.
  - The period before reaching the target velocity is called the catching phase, and the period after reaching the target velocity is called the InGear phase.
  - When the gear ratio is positive, the slave axis moves in the same direction as the master axis.



Relationships of the master and slave axis velocity with the gear ratio

- The master and slave axes specified in this instruction cannot be the same.
- This instruction starts electronic gearing on a rising edge of the parameter Execute.
- After the InSync state is reached, the slave axis is forced to follow the master axis according to the specified gear ratio.
- This instruction can be used in combination with other synchronous control instructions. The in-out variable Master of this instruction can be the master or slave axis of other synchronous control instructions; the in-out variable Slave of this instruction can serve as the master axis of other synchronous control instructions.
- During the execution of this instruction, the MC_MoveSuperImposed instruction can be used.
- If bus cycle is specified for the cycle task of the master axis, the velocity of the slave axis may show a sudden jump after synchronization. It is recommended that users specify EtherCAT cycle for the cycle task of both the master and slave axes before enabling a synchronization instruction.
- MC_Home, MC_Homing, or MC_HomingWithParameter can be used during the instruction execution.

  This instruction places the master axis in the AXIS_HOMING state. If the instruction is executed while the master axis is already in this state, the signal source "MasterValueSource" must be set to "mcLatestCommand"; otherwise, the slave axis will overspeed.

Software limit

- During gearing, if the master axis position is beyond the software limit, the velocity of the master axis will decelerate to 0 within one cycle, and the velocity of the slave axis will also decelerate to 0, maintaining the synchronized motion (SynchronizedMotion) state.
- During gearing, if the slave axis position is beyond the software limit, a fault will occur, and the velocity of the slave axis will decelerate to 0 within one cycle, with no impact on the motion status of the master axis.

Conditions for executing the instruction

- This instruction can be executed when the master axis (Master) specified by the instruction is in the ErrorStop or PowerOff state.
- This instruction can be executed when the slave axis specified by the instruction is not in the ErrorStop, Stopping, Homing, or PowerOff state.

Not meeting the conditions for catching up

- If the slave axis is a linear axis and the InverseFlag is set to FALSE, and when the instruction is executed, the target motion direction of the slave axis (determined by the sign of the product of the gear ratio and MasterStartDistance). deviates from the synchronization position, the master and slave axes cannot start electronic gearing, and a fault occurs.

Position type selection

You can select the MasterValueSource for synchronizing the slave axis based on the following options.

- mcCommand: Commanded position (Value calculated in the previous task cycle, fSavePosition).

  For the current cycle, use the master axis position calculated in the previous task cycle.

- mcFeedback: Actual position (Value obtained in the same task cycle, fActPosition).

  Use the actual master axis position obtained in the same task cycle.

- mcLaterCommand: Commanded position (Value calculated in the same task cycle, fSetPosition).

  Use the master axis position calculated in the same task cycle.

- When the master axis starting electronic gearing is specified as a virtual axis, the value of the input variable MasterValueSource cannot be set to 1 (mcFeedback); otherwise, a fault will occur.

MasterSyncPosition (Master axis synchronization position), MasterStartDistance (Master axis distance), SlaveSyncPosition (Slave axis synchronization position)

- If the master axis is specified as a rotary axis, you can specify the MasterSyncPosition outside the rotation cycle as the reminder of MasterSyncPosition divided by the rotation cycle.
- If the slave axis is specified as a rotary axis, you can specify the SlaveSyncPosition outside the rotation cycle as the reminder of SlaveSyncPosition divided by the rotation cycle.
- Inconsistent signs of MasterStartDistance and master axis velocity may result in the master and slave axes failing to achieve the gear ratio, maintaining the instruction in a parameter Busy and preventing it from reaching InSync.

Master axis position, slave axis position, and synchronization position

- When this instruction is executed with the master axis being specified as a rotary axis, and the master axis moves in the positive direction at positions greater than MasterSyncPosition - MasterStartDistance, the slave axis enters the catching phase when the master axis position reaches MasterSyncPosition - MasterStartDistance. (This also applies to reverse motion)
- When this instruction is executed with the slave axis being specified as a rotary axis, and the slave axis position moves in the positive direction at positions greater than the SlaveSyncPosition, the gear ratio is established when the slave axis reaches the SlaveSyncPosition.

Inverse enable (InverseFlag)

- This instruction determines whether the slave axis is allowed to run reversely during the catching phase by specifying the value of the input variable InverseFlag.

Local pulse axis:

- This instruction can be used for local pulse axes.

- **Precautions**
  - After this instruction is triggered, if the master axis velocity is 0, a fault will occur.
  - If the values of RatioNumerator and RatioDenominator are set to 0, they will be automatically corrected to 1.
  - If the value of RatioDenominator is set to a negative value, it will be automatically corrected to 1.
  - The MC_Phasing instruction can take effect at any time during the execution of this instruction, but phase shifting only begins after reaching the InGear.

- If the slave axis velocity exceeds the software-specified maximum velocity or deceleration during the catching process, a fault will occur.

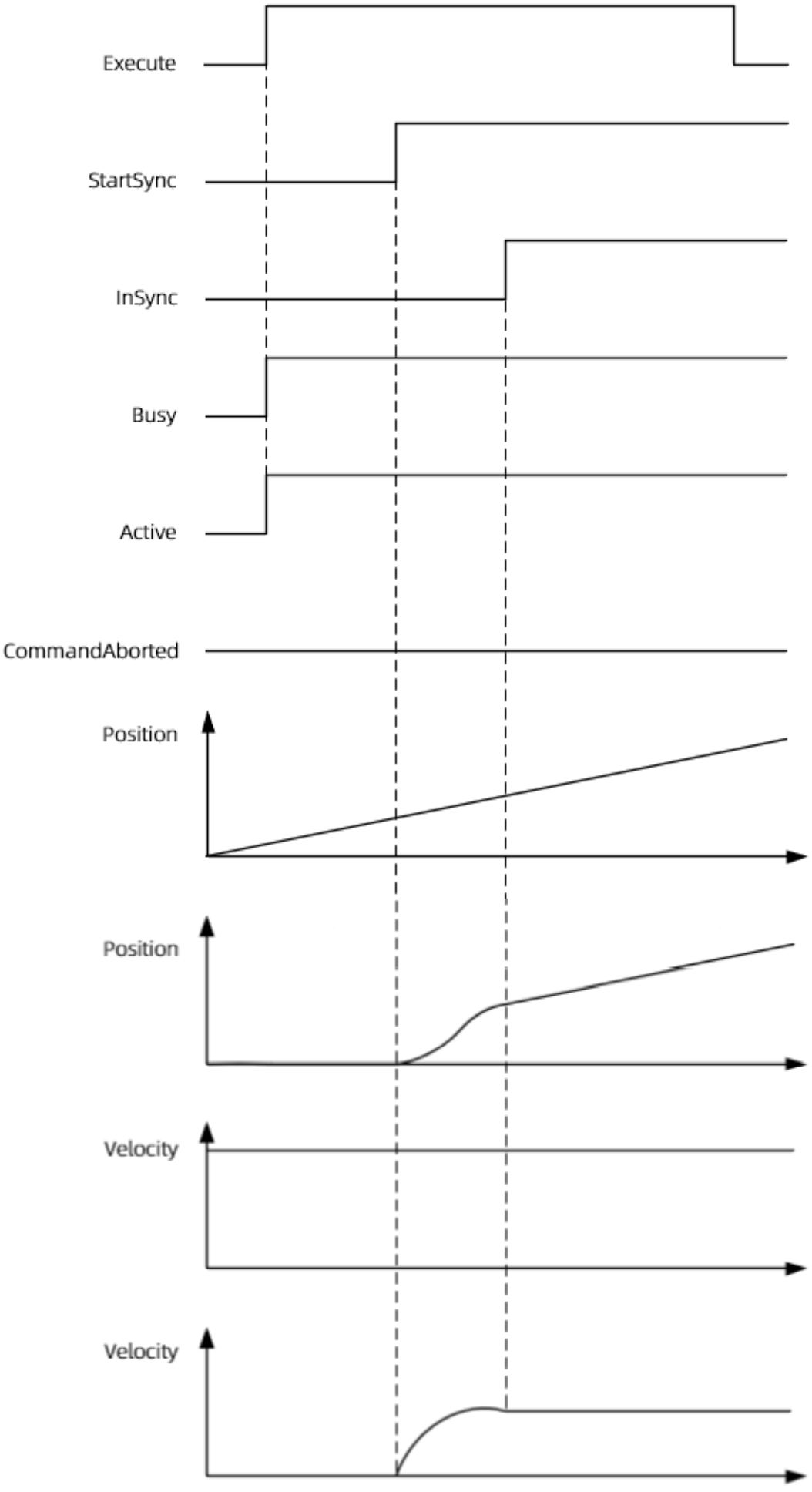
- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Busy of the function block indicates that the current function block is being executed; the parameter Active indicates that the axis is being controlled; the parameter CommandAborted indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

  Electronic gearing starts on a rising edge of the input parameter Execute. The parameter Busy shows the value TRUE at the same time as Execute is set to TRUE. After the gearing starts, the parameter "Active" shows the value TRUE. When the master axis position reaches the input variable "MasterStartDistance", the output variable "StartSync" shows the value TRUE, and the slave axis starts gearing.

  When the input variables "MasterSyncPosition" and "SlaveSyncPosition" are reached, the parameter "InSync" shows the value TRUE.

  When this instruction is aborted by another instruction, the parameter CommandAborted shows the value TRUE, while the parameters Busy, Active, StartSync, and InSync show the value FALSE.


Execute
StartSync
InSync
Busy
Active
CommandAborted
Position
Position
Velocity
Velocity

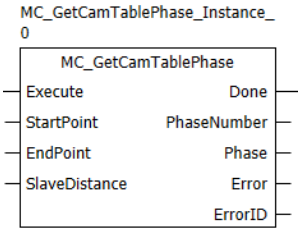
- **Faults**

For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.4.7 MC_GetCamTablePhase

This instruction gets the phase of a specified displacement between two key nodes in the cam table (currently supported cam curve types include: straight line, parabolic curve, cubic curve and quintic curve).

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GetCamTablePhase	Get cam table phase		MC_GetCamTablePhase_Instance_0(Execute :=, StartPoint :=, EndPoint :=, SlaveDistance :=, Done =>, PhaseNumber =>, Phase =>, Error =>, ErrorID =>);

- **Related variables**

  Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
StartPoint	Start point	18.3.3.2.3.3 MC_CAM_KEY_NODE_REF: Key node structure	-	-	Specify a key node for the cam curve.
EndPoint	End point	18.3.3.2.3.3 MC_CAM_KEY_NODE_REF: Key node structure	-	-	Specify a key node for the cam curve.
SlaveDistance	Slave axis displacement	LREAL	-	0	Specify the displacement [u] for which the phase must be obtained.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
PhaseNumber	Phase number	DINT	-1 to 5	0	Number of phases corresponding to the displacement. -1: Infinite. 0: No phase for the specified displacement. Greater than 0: Number of phases.
Phase	Phase	LREAL[6]	Positive number, negative number, or 0	0	Array of master axis phases corresponding to the slave axis.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

Key node structure of StartPoint

Input Variable	Name	Data Type	Valid Range	Default Value	Description
fPhase	Phase	LREAL	-	-	Phase at the node.
fDistance	Displacement	LREAL	-	-	Displacement at the node.
fVel	Velocity	LREAL	-	-	Velocity at the node.
fAcc	Acceleration	LREAL	-	-	Acceleration at the node.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
fJerk	Jerk	LREAL	-	-	Jerk at the node.
iCurveType	Curve type	INT	1, 2, 3, 4, 5, 6, 7, 8	-	Curve type: 1: Straight line. 2: Parabolic curve. 3: Cubic curve. 4: Quintic curve. 5: Septic curve. 6: Cosine acceleration curve. 7: Sine acceleration curve. 8: Modified trapezoidal acceleration curve.

Key node structure of EndPoint

Input Variable	Name	Data Type	Valid Range	Default Value	Description
fPhase	Phase	LREAL	-	-	Phase at the node.
fDistance	Displacement	LREAL	-	-	Displacement at the node.
fVel	Velocity	LREAL	-	-	Velocity at the node.
fAcc	Acceleration	LREAL	-	-	Acceleration at the node.
fJerk	Jerk	LREAL	-	-	Jerk at the node.

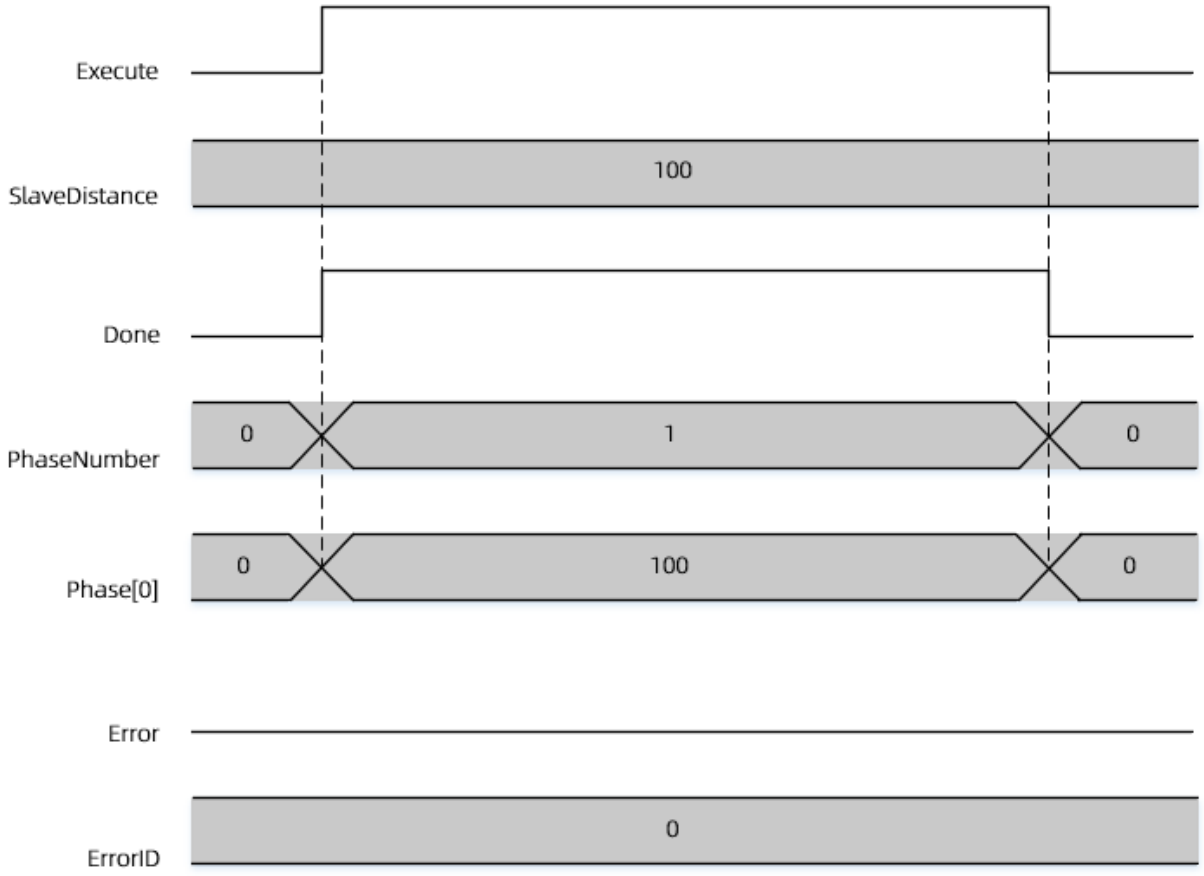
Input Variable	Name	Data Type	Valid Range	Default Value	Description
iCurveType	Curve type	INT	1, 2, 3, 4, 5, 6, 7, 8	-	Curve type: 1: Straight line. 2: Parabolic curve. 3: Cubic curve. 4: Quintic curve. 5: Septic curve. 6: Cosine acceleration curve. 7: Sine acceleration curve. 8: Modified trapezoidal acceleration curve.

The StartPoint and EndPoint are user-defined as required.

- **Function**
  - This instruction gets the phase of a specified displacement between two key nodes in the cam table.
  - This instruction does not support phase calculation for advanced curves. If an advanced curve (polynomic 7 curve, cosine acceleration curve, sine acceleration curve, modified trapezoidal acceleration curve) is selected, an error will occur.
  - This instruction is executed on a rising edge of the input variable Execute.
  - When the phase of the EndPoint specified in this instruction is less than the phase of the StartPoint, a fault occurs.
  - When the phase of the EndPoint specified in this instruction coincides with the phase of the StartPoint (a phase shift less than 0.0001 is considered coincident), a fault occurs.
  - The cam curve types currently supported by this instruction are: linear curve, parabolic curve, cubic curve, and quintic curve.
  - When the iCurveType of the EndPoint or StartPoint specified in this instruction is out of the specified range, a fault occurs.
  - The value of SlaveDistance must be between the maximum and minimum slave axis displacement values of the cam curve from StartPoint to EndPoint; otherwise, the PhaseNumber is set to 0, with no phase for the specified displacement, and a fault occurs.
- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates that the instruction has been executed successfully.

Note: The following example shows the timing diagram with the curve types of the input variables StartPoint and EndPoint being specified as straight line, the output variable "PhaseNumber" being set to "1", and the output variable "Phase" being set to "100".



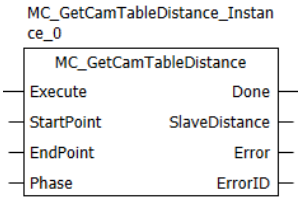
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.4.8 MC_GetCamTableDistance

This instruction gets the displacement at a specified phase between two key nodes in the cam table.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GetCamTableDistance	Get cam table displacement		MC_GetCamTableDistance_Instance_0(Execute :=, StartPoint :=, EndPoint :=, Phase :=, Done =>, SlaveDistance =>, Error =>, ErrorID =>);

- **Related variables**

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
StartPoint	Start point	18.3.3.2.3.3 MC_CAM_KEY_NODE_REF: Key node structure	-	-	Specify a key node for the cam curve.
EndPoint	End point	18.3.3.2.3.3 MC_CAM_KEY_NODE_REF: Key node structure	-	-	Specify a key node for the cam curve.
Phase	Phase	LREAL	-	0	Specify the phase [u] for which displacement is required.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
SlaveDistance	Slave axis displacement	LREA	Positive number, negative number, or 0	0	Slave axis displacement [u].
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

Key node structure of StartPoint

Input Variable	Name	Data Type	Valid Range	Default Value	Description
fPhase	Phase	LREAL	-	-	Phase at the node.
fDistance	Displacement	LREAL	-	-	Displacement at the node.
fVel	Velocity	LREAL	-	-	Velocity at the node.
fAcc	Acceleration	LREAL	-	-	Acceleration at the node.
fJerk	Jerk	LREAL	-	-	Jerk at the node.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
iCurveType	Curve type	INT	1, 2, 3, 4, 5, 6, 7, 8	-	Curve type: 1: Straight line. 2: Parabolic curve. 3: Cubic curve. 4: Polynomic 5 curve. 5: Septic curve. 6: Cosine acceleration curve. 7: Sine acceleration curve. 8: Modified trapezoidal acceleration curve.

Key node structure of EndPoint

Input Variable	Name	Data Type	Valid Range	Default Value	Description
fPhase	Phase	LREAL	-	-	Phase at the node.
fDistance	Displacement	LREAL	-	-	Displacement at the node.
fVel	Velocity	LREAL	-	-	Velocity at the node.
fAcc	Acceleration	LREAL	-		Acceleration at the node.
fJerk	Jerk	LREAL	-	-	Jerk at the node.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
iCurveType	Curve type	INT	1, 2, 3, 4, 5, 6, 7, 8	-	Curve type: 1: Straight line. 2: Parabolic curve. 3: Cubic curve. 4: Polynomic 5 curve. 5: Septic curve. 6: Cosine acceleration curve. 7: Sine acceleration curve. 8: Modified trapezoidal acceleration curve.

The StartPoint and EndPoint are user-defined as required.

- **Function**

  Bus-controlled axis:

  - This instruction gets the displacement at a specified phase between two key nodes in the cam table.
  - This instruction is executed on a rising edge of the input variable Execute.
  - When the phase of the EndPoint specified in this instruction is less than the phase of the StartPoint, a fault occurs.
  - When the phase of the EndPoint specified in this instruction coincides with the phase of the StartPoint (a phase shift less than 0.0001 is considered coincident), a fault occurs.
  - When the specified phases of StartPoint and/or EndPoint are negative, a fault occurs.
  - This instruction currently supports the following cam curve types: linear curve, parabolic curve, cubic curve, quintic curve, septic curve, sine acceleration curve, cosine acceleration curve, and modified trapezoidal acceleration curve.
  - When the iCurveType of the EndPoint or StartPoint specified in this instruction is out of the specified range, a fault occurs.
  - The value of Phase must be between the fPhase of the StartPoin and the fPhase of the EndPoint (inclusive); otherwise, a fault occurs.

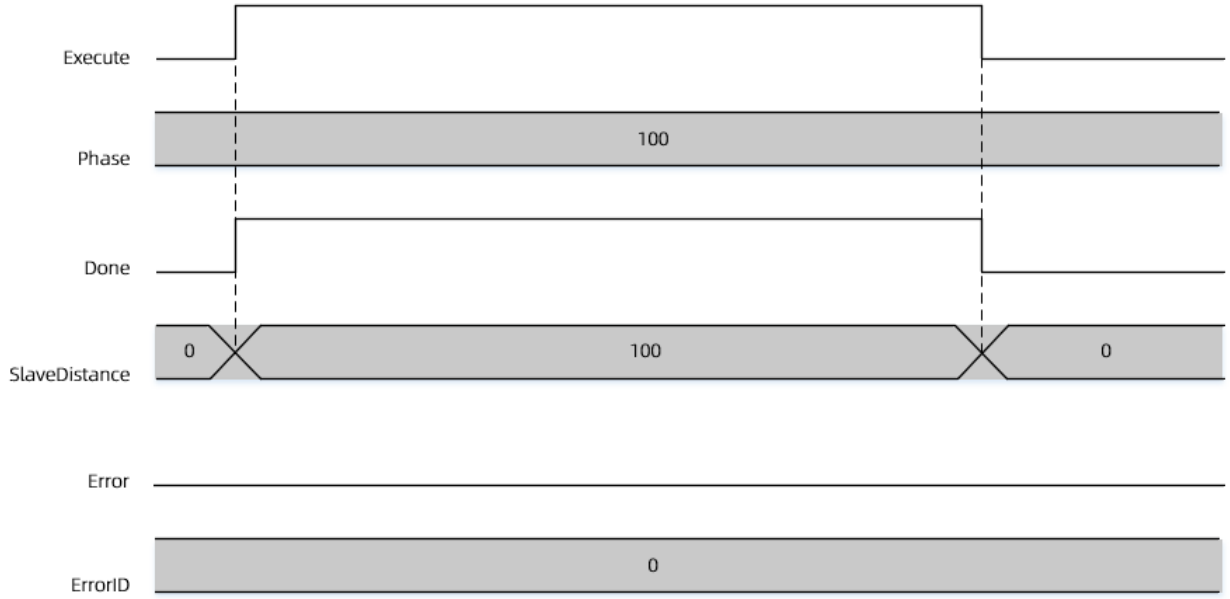
  Local pulse axis:

  - This instruction can be used for local pulse axes.

- **Timing diagram**

The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates that the instruction has been executed successfully.

Note: The following example shows the timing diagram with the curve type of input variables "StartPoint" and "EndPoint" being specified as straight line and the output variable "SlaveDistance" being specified as "100".



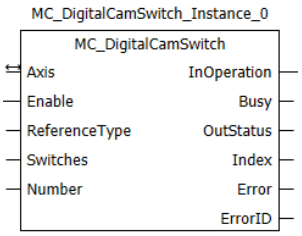
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.4.9 MC_DigitalCamSwitch

This instruction outputs a specific signal at the specified axis position.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_DigitalCamSwitch	Enable digital cam switch		MC_DigitalCamSwitch_Instance_0(Axis :=, Enable :=, ReferenceType :=, Switches :=, Number :=, InOperation =>, Busy =>, OutStatus =>, Index =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled; FALSE: Disabled.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Reference Type	Position type selection	18.3.3.2.2.11 MC_REFERENCE_TYPE: Data Type Enumeration	0: mcCommand 1: mcFeedback 2: mcLaterCommand 3: mcMasterPhase	0	0: Value calculated in the previous task cycle (fSavePosition). 1: Value obtained in the same task cycle (fActPosition). 2: Value calculated in the same task cycle (fSetPosition). 3: Phase of the master axis when the axis specified by Axis works as the cam slave axis (fMasterPhase).
Switches	Switch state	ARRAY [1..64] OF 18.3.3.2.3.6 MC_SWITCH_REF: Switch structure	-	-	Address of the variable used to input the switch state.
Number	Switch number	INT	1-64	-	Up to 64 switches.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
InOperation	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
OutStatus	Output status	BOOL	TRUE, FALSE	FALSE	Output status of the switch.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Index	Index	INT	1-64	0	Index of the comparison point to be executed.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

Switches structure

Input Variable	Name	Data Type	Valid Range	Default Value	Description
fPosition	Position for the output to turn ON.	REAL	-	0	Absolute position for the output to turn ON.
iMode	Switch mode	INT	0, 1, 2	0	Switch mode selection. 0: Disabled; 1: Position based; 2: Time based.
iDirection	Direction	INT	0, 1, 2	0	Master axis direction selection. 0: Positive; 1: Negative; 2: No direction.
fParameter	Position type	REAL	-	0	iMode = 1: fParameter is the position where the switch turns OFF; iMode = 2: fParameter is the duration (in ms) for which the switch remains ON; in time-based mode, the decimal part is ignored, and the value of fParameter cannot be more than 10,000 ms.

- **Function**

  Bus-controlled axis:

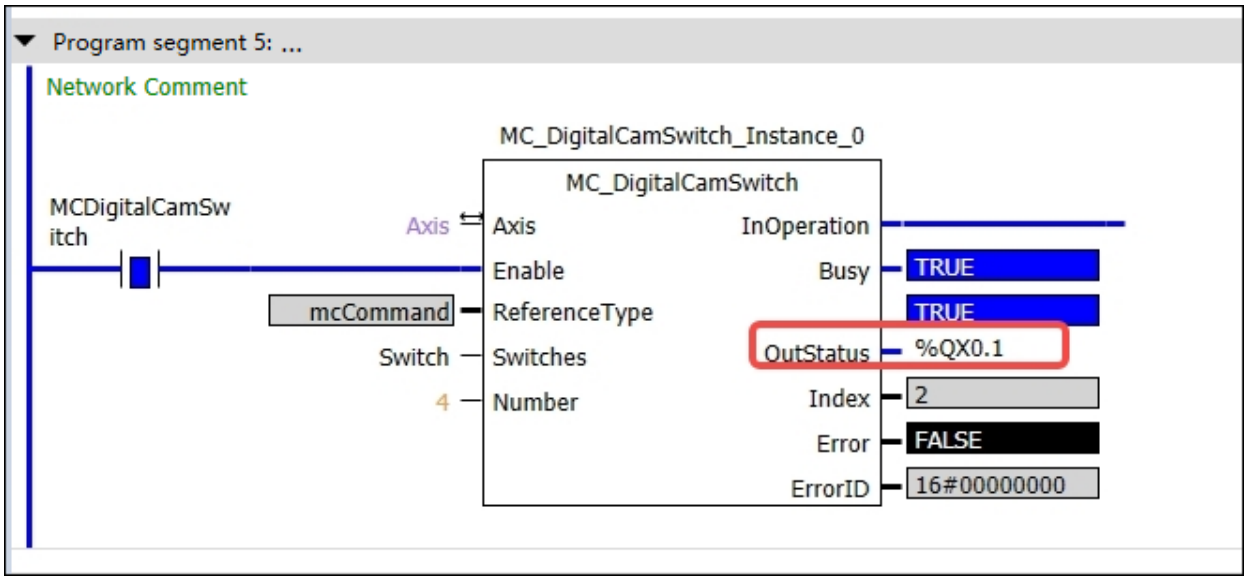
  - This instruction outputs a specific signal at the specified axis position.
  - The instruction is executed when the input variable Enable is set, after which modifying the input parameters is invalid. When Enable is reset, the OutStatus and Index are cleared.

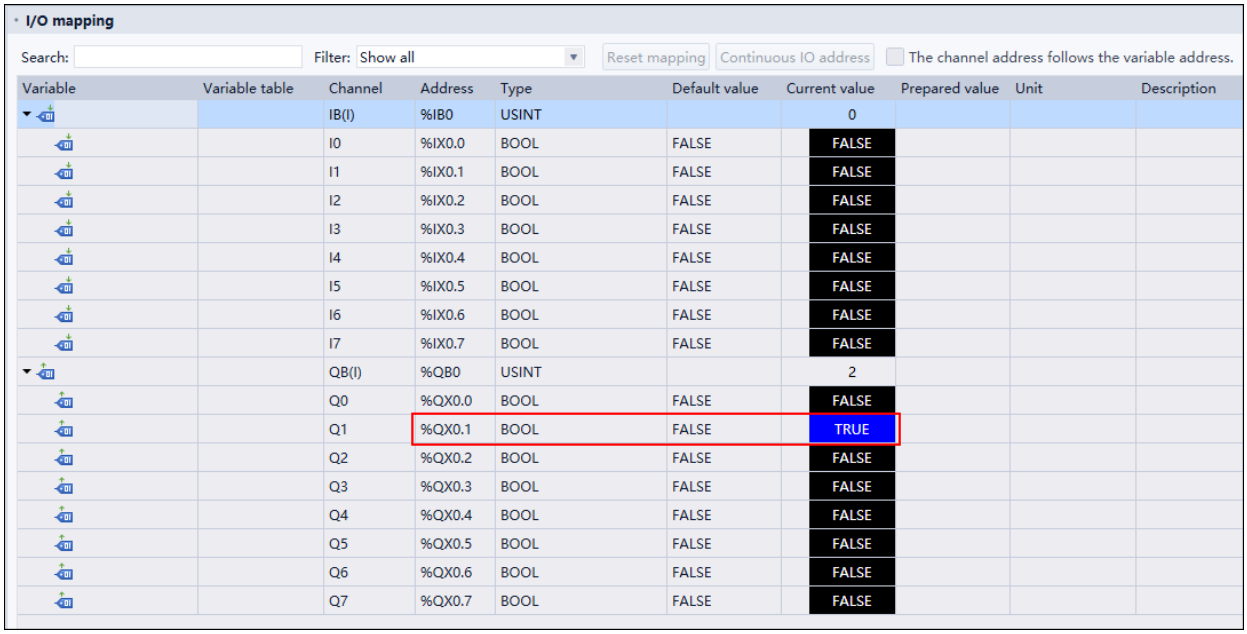
  Local pulse axis:

  - This instruction can be used for local pulse axes.

OutStatus

- The output parameter "OutStatus" can be bound to a boolean variable or the address of an IO signal, allowing the value of the bound variable or IO signal to be controlled via the MC_DigitalCamSwitch instruction.
- The output parameter OutStatus indicates the tappet output status of the MC_DigitalCamSwitch instruction. When the axis position information matches the fPosition in the input parameter Switch structure, OutStatus is set to TRUE and remains so for the specified time or distance.





I/O mapping

Search: Filter: Show all Reset mapping Continuous IO address The channel address follows the variable address.

Variable	Variable table	Channel	Address	Type	Default value	Current value	Prepared value	Unit	Description
		IB(I)	%IB0	USINT		0			
		I0	%IX0.0	BOOL	FALSE	FALSE			
		I1	%IX0.1	BOOL	FALSE	FALSE			
		I2	%IX0.2	BOOL	FALSE	FALSE			
		I3	%IX0.3	BOOL	FALSE	FALSE			
		I4	%IX0.4	BOOL	FALSE	FALSE			
		I5	%IX0.5	BOOL	FALSE	FALSE			
		I6	%IX0.6	BOOL	FALSE	FALSE			
		I7	%IX0.7	BOOL	FALSE	FALSE			
		QB(I)	%QB0	USINT		2			
		Q0	%QX0.0	BOOL	FALSE	FALSE			
		Q1	%QX0.1	BOOL	FALSE	TRUE			
		Q2	%QX0.2	BOOL	FALSE	FALSE			
		Q3	%QX0.3	BOOL	FALSE	FALSE			
		Q4	%QX0.4	BOOL	FALSE	FALSE			
		Q5	%QX0.5	BOOL	FALSE	FALSE			
		Q6	%QX0.6	BOOL	FALSE	FALSE			
		Q7	%QX0.7	BOOL	FALSE	FALSE			

ReferenceType

You can specify the ReferenceType based on the following options.

- mcCommand: Commanded position (Value calculated in the previous task cycle, fSavePosition).

  For the current cycle, use the master axis position calculated in the previous task cycle.
- mcFeedback: Actual position (Value obtained in the same task cycle, fActPosition).

  Use the actual master axis position obtained in the same task cycle.
- mcLaterCommand: Commanded position (Value calculated in the same task cycle, fSetPosition).

  Use the master axis position calculated in the same task cycle.
- mcMasterPhase: Phase of the master axis when the axis specified by Axis works as the cam slave axis.

When this type is selected, the axis must be a cam slave axis; otherwise, a fault occurs.

When the slave axis velocity is 0, there is no output for OutStatus.

iMode

- You can specify the switch mode by defining the value of iMode in Switches of this instruction.
- When iMode is set to 1 (Position based mode), the fParameter in Switches is the position where the switch turns OFF. When the switch terminal output position is insufficient for one complete ETC cycle, there is no output for OutStatus.
- When iMode is set to 2 (Time based mode), the fParameter in Switches is the duration (in ms) for which the switch remains ON. In time-based mode, the value of fParameter cannot be more than 10,000 ms. When the ON duration of the switch terminal output is shorter than the time of one ETC cycle, OutStatus remains active for the duration of one complete ETC cycle.

- **Precautions**
  - When the axis is specified as a linear axis, the fPosition of the input variable Switches must be less than the fParameter.
  - When the axis is specified as a rotary axis, the fPosition and fParameter of the input variable Switches must be greater than 0 and less than or equal to the rotation cycle specified in this instruction.
  - When the axis is specified as a slave axis for camming, the fPosition and fParameter of the input variable Switches must be greater than 0 and less than or equal to the phase of the master axis synchronized to the axis for camming specified in this instruction.
  - When this instruction is executed in the homing mode, a fault occurs.
  - Index and OutStatus are not refreshed in the same task cycle, as Index evaluates if the target point is reached within the displacement range of a single cycle.
- **Timing diagram**

  Start and end of the switch comparison output: After this instruction is executed, the tappet point array is sorted internally to identify the tappet point closest to the current position of the axis. When the axis reaches this point, the switch is turned ON immediately.

  iMode is used to set the comparison mode:

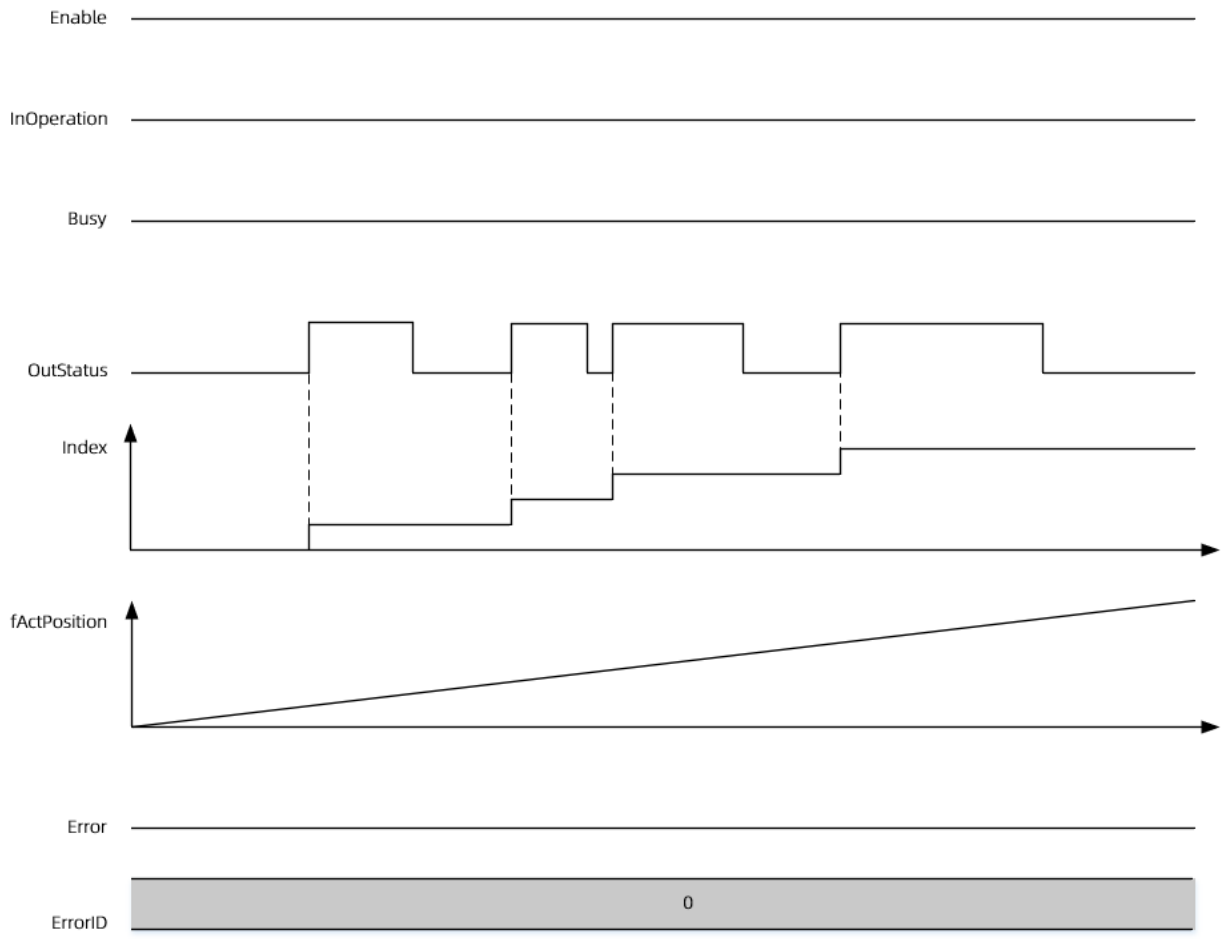
  - When iMode is set to 0, this comparison point is disabled. When the axis passes this point, the switch has no output.
  - When iMode is set to 1, fParameter is the position where the switch turns OFF.
  - When iMode is set to 2, fParameter is the duration (in ms) during which the switch remains ON.

  iDirection specifies the running direction of the master axis. It must be used together with iMode.

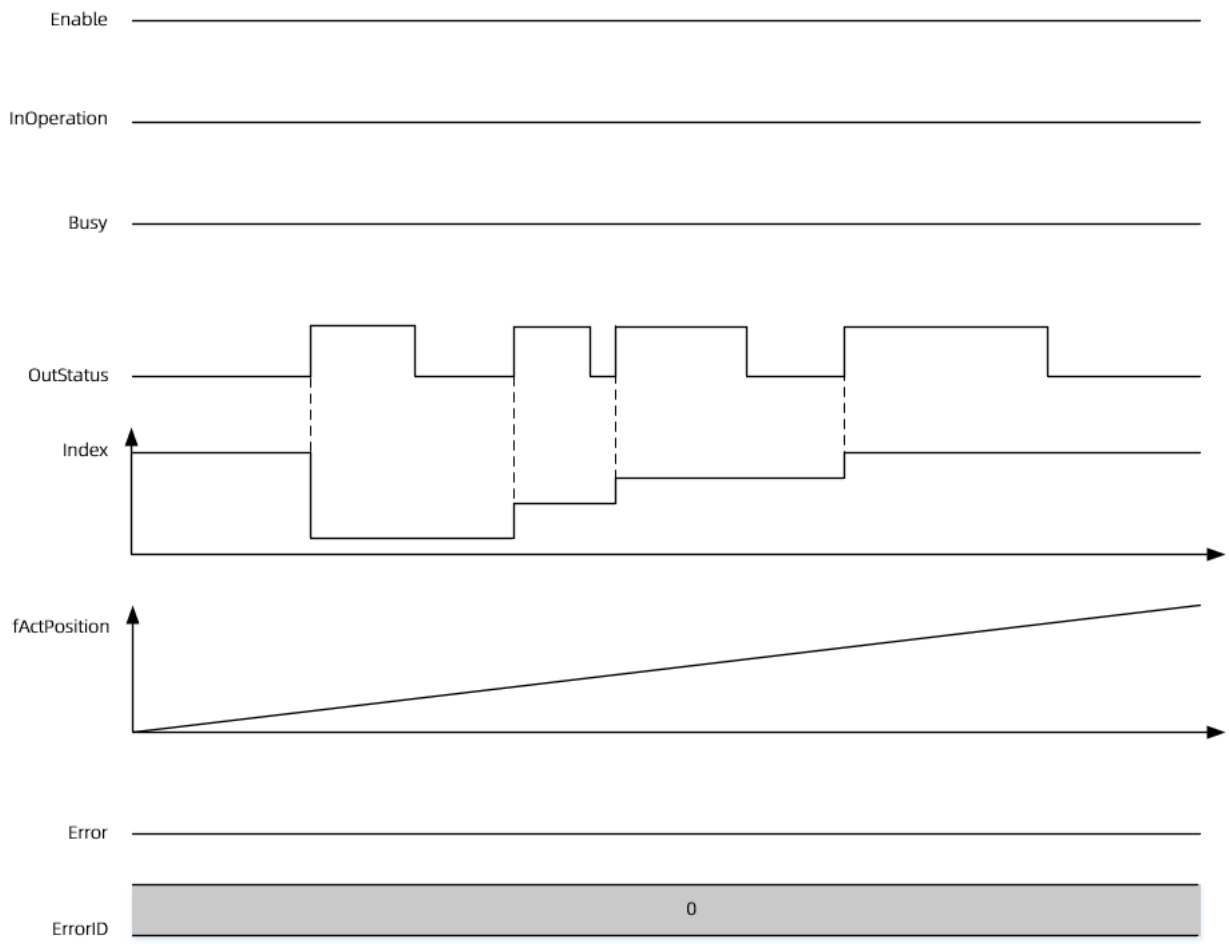
No.	fPosition	fParameter	iMode	iDirection
1	10	15	1	0
2	20	25	1	0
3	24	30	1	0
4	35	100	2	0
5	40	45	0	0

No.	fPosition	fParameter	iMode	iDirection
6	10	15	1	1
7	20	25	1	1
8	24	30	1	1
9	35	100	2	1

When the master axis moves in the forward direction:



When the master axis moves in the negative direction:



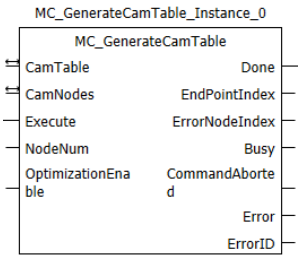
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.4.10 MC_GenerateCamTable

This instruction checks or updates the cam table data.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GenerateCamTable	Update the cam table with specified cam data		MC_GenerateCamTable_Instance_0(CamTable :=, CamNodes :=, Execute :=, NodeNum :=, OptimizationEnable :=, Done =>, EndPointIndex =>, ErrorNodeIndex =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Value Range	Default Value	Description
CamTable	Cam table	18.3.3.2.3.1 MC_CAM_TABLE_REF: Cam table structure	-	-	Cam table.
CamNodes	Cam nodes	ARRAY[*] OF 18.3.3.2.3.3 MC_CAM_KEY_NODE_REF: Key node structure	-	-	Array of cam table node data structures.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
NodeNum	Number of nodes	UINT	-	0	Number of cam table nodes. If it is specified as 0, use the original number of cam table nodes.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
OptimizationEnable	Velocity optimization switch	BOOL	TRUE, FALSE	FALSE	TRUE: Optimize the suddenly changed velocity at node connections. FALSE: Do not optimize the suddenly changed velocity.

MC_CAM_KEY_NODE_REF structure composition

Input Variable	Name	Data Type	Valid Range	Default Value	Description
fPhase	Phase	LREAL	-	0	Phase (of the master axis) at the node.
fDistance	Displacement	LREAL	-	0	Displacement (of the slave axis position) at the node.
fVel	Velocity	LREAL	-	0	Velocity at the node.
fAcc	Acceleration	LREAL	-	0	Acceleration at the node.
fJerk	Jerk	LREAL	-	0	Jerk at the node.
iCurveType	Curve type	INT	1, 2, 3, 4, 5, 6, 7, 8	-	Curve type: 1: Straight line. 2: Parabolic curve. 3: Cubic curve. 4: Quintic curve. 5: Septic curve. 6: Cosine acceleration curve. 7: Sine acceleration curve. 8: Modified trapezoidal acceleration curve.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
EndPointIndex	End point index	UINT	-	-	Output the cam table end point index after the instruction is executed.
ErrorNodeIndex	Error node index	UINT	-	-	Output the number of any fault node.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**
  - This instruction generates a cam table by specifying cam table data through input parameters.
  - This instruction is executed on a rising edge of the input variable Execute.
  - By setting the input parameter OptimizationEnable, you can optimize the suddenly changed velocity between nodes while updating cam nodes to the cam table, and adjust the acceleration and jerk according to the optimized velocity.

    Re-execution of motion control instructions

  - This instruction can be re-executed.
  - During the execution of this instruction with the output variable Busy being set, if the instruction is re-executed after any parameters are modified, it will be executed with the modified parameter values.

    Velocity optimization

Currently, you can optimize sudden velocity changes due to the following two scenarios. The optimization is performed for the velocity of the entire cam table, rather than the velocity between two nodes.

- When a parabolic, cubic, quintic, or septic curve is followed by a linear curve, the velocity of the current node and subsequent nodes is adjusted to match the velocity of the linear curve.
- When a parabolic, cubic, quintic, or septic curve is followed by a cosine acceleration curve, a sine acceleration curve, or a modified trapezoidal acceleration curve, the velocity at the current node and the next node is set to 0.

The acceleration and jerk are adjusted based on the optimized velocity.

Parameter validity check

- This instruction performs a validity check on the input values of CamNodes.
- If any value of CamNodes is invalid, the corresponding node number is output to ErrorNodeIndex.
- The phase and displacement of the first CamNodes must be 0; otherwise, a fault will occur.
- The phase, displacement, and velocity ratio cannot be more than 9,999,999; otherwise, a fault will occur.
- The number of key nodes for CamNodes cannot be more than 20000, and the number of dense nodes cannot be more than 65535; otherwise, a fault will occur.
- The value of CamNodes cannot be less than 2; otherwise, a fault will occur.
- The phases of CamNodes must be in ascending order; otherwise, a fault will occur.
- The phase shift between adjacent nodes for CamNodes must be greater than 0.0001; otherwise, a fault will occur.
- The curve type for CamNodes must be within the valid range; values outside the range are treated as a quintic curve.

CamNodes

- The instruction only allows input of key nodes for CamNodes.
- CamNodes is an in-out variable and can be compiled successfully with input parameters. CamNodes is bound to automatically declared variable-length array, the declared variables must be assigned valid values. The data in CamNodes is the **data to be updated**.

NodeNum

- The NodeNum value of this instruction should be less than or equal to the number of nodes in CamNodes.
- The NodeNum value of this instruction cannot be greater than the number of nodes in CamTable.

Velocity ratio adjustment

- When this instruction is executed to update cam node parameters, any improper parameter value of velocity ratio at the node is automatically adjusted.
- If the curve type between adjacent nodes is a straight line:

  When the coordinates of point A1 are (x1, y1) and the coordinates of point A2 are (x2, y2), V2 = (y2 - y1) / (x2 - x1), and the ratio value of V2 is automatically adjusted.

- When a quintic curve is followed by a linear curve:

Points A1 and A2 define a quintic curve, and points A2 and A3 define a linear curve. To prevent a sudden velocity change from the quintic curve to the linear curve, the velocity ratio must be automatically adjusted.

First, calculate the velocity ratio of the straight line segment, set the velocity value of the straight line segment as V3, and then update the value of V3 to V2: V3 = (y3 - y2) / (x3 - x2), V2 = V3.

- When a quintic curve is followed by another quintic curve, no adjustment is made.
- If the curve type is a straight line followed by another straight line, a sudden change in velocity ratio will be allowed.

The specific behaviors of this instruction are as follows when the input parameter CamNodes is the data to be updated and NodeNum is equal to 0 or not equal to 0.

- **CamNodes is the data to be updated, and NodeNum is specified as 0**

  With this parameter combination, the number of cam nodes for Cam_0 in the cam table is updated with the number of nodes for CamNodes, and the updated data is checked for validity. If any invalid data is detected, a fault will occur.

  If the number of nodes for CamNodes is greater than the number of nodes for Cam_0 in the cam table, the node data will be updated according to the number of nodes for Cam_0 in the cam table, followed by the validity check.

  If the number of nodes for CamNodes is less than the number of nodes for Cam_0 in the cam table, a fault will occur.

- **CamNodes is the data to be updated, and NodeNum is specified as a value other than zero**

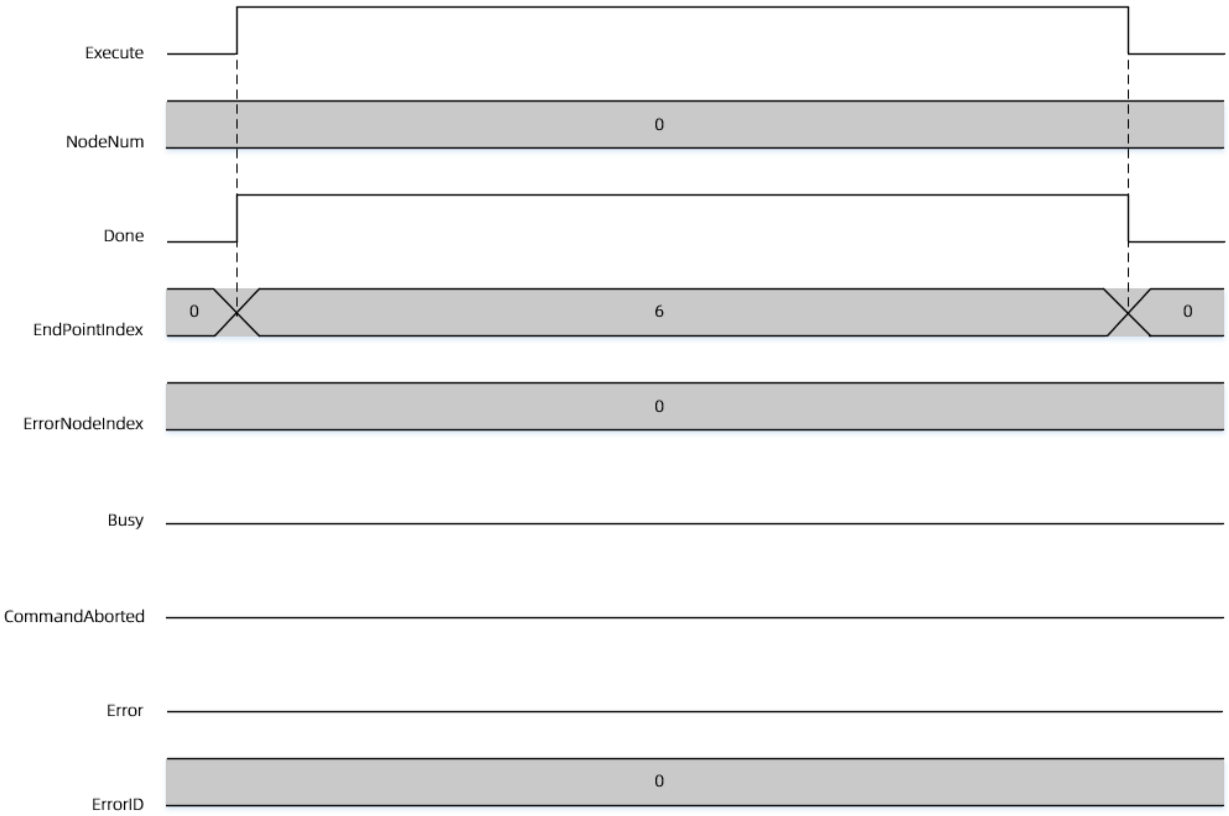
  With this parameter combination, the number of cam nodes for Cam_0 in the cam table is updated with the number of nodes for CamNodes, and the updated data is checked for validity. If any invalid data is detected, a fault will occur. After a successful update, the number of nodes for Cam_0 in the cam is equal to the value of NodeNum.

  The value of NodeNum cannot be more than the number of nodes for Cam_0 in the cam table; otherwise, a fault will occur.

- **Precautions**
  - When the in-out variable CamTable is a user-defined cam table, you must first use this instruction MC_CheckCamTable to check the validity of the user-defined cam table before executing the instruction to generate the cam table.
  - This instruction cannot be executed during the execution of MC_WriteCam or MC_ReadCam instruction; otherwise, a cam data read/write fault occurs.
  - Modifying the cam table by using methods other than the MC_GenerateCamTable instruction may result in abnormal results for the MC_CamIn or MC_CamInV2 instruction that are currently executing or about to execute.
- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates the instruction has been executed successfully; the parameter Busy indicates that the current function block is being executed; the parameter CommandAborted of the function block indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

The output variable Busy shows the value TRUE on a rising edge of the variable Execute, and the output variable Done shows the value TRUE in the next cycle.



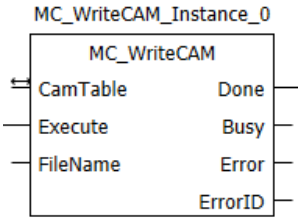
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.4.11 MC_WriteCAM

This instruction stores the cam table created within the cam table editor to a file at runtime and makes it the available in the module.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_WriteCAM	Store the edited cam table as a file		MC_WriteCAM_Instance_0(CamTable :=, Execute :=, FileName :=, Done =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
CamTable	Cam table	18.3.3.2.3.1 MC_CAM_TABLE_REF: Cam table structure	-	-	Specify the cam table to be stored.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
FileName	File name	STRING	-	''	Contain the cam description in the specified ASCII format.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**
  - This instruction stores the cam table created within the cam table editor into a file.
  - This instruction is executed on a rising edge of the input variable Execute.

File write format

  - The format of the file written by this instruction is as follows. Each element is written with a maximum of 20 bytes.

Name	Data Type	Description
CamTable version	STRING	Used to distinguish whether it is a CamTable text.
CamTable name	STRING	Name of a cam table.
Node type	INT	Used to distinguish whether it is written as key or dense nodes.
Number of nodes	INT	Number of nodes written.
fPhase	LREAL	Master axis phase.
fDistance	LREAL	Slave axis displacement.
fVel	LREAL	Velocity ratio (only written in key nodes).
fAcc	LREAL	Acceleration ratio (only written in key nodes).
Jerk	LREAL	Jerk ratio (only written in key nodes).
iCurveType	INT	Curve type (only written in key nodes).

FileName

  - When read/write is executed, a fixed suffix ".cam" is added internally.
  - The input variable FileName of this instruction cannot contain Chinese characters; otherwise, a fault will occur.

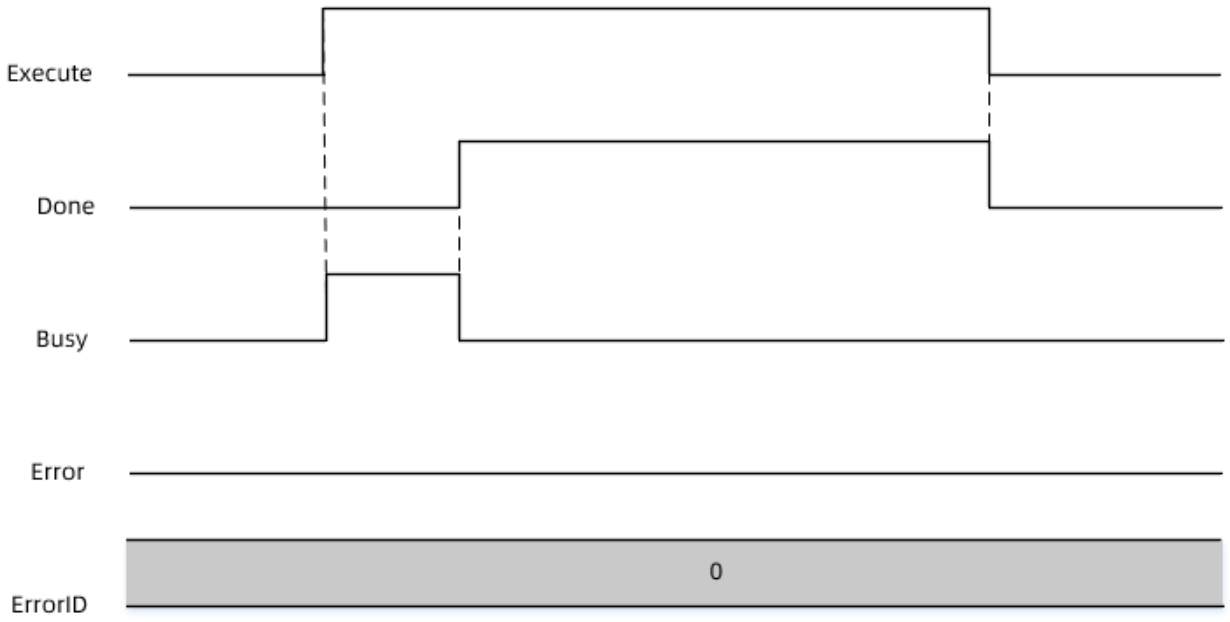
Node type

- The node type of CamTable must be key or dense nodes; otherwise, a fault will occur.
- When key nodes are specified for CamTable, the number of nodes must be between 2 and 20000; otherwise, a fault will occur.
- When dense nodes are specified for CamTable, the number of nodes cannot be less than 2 or more than 65,536; otherwise, a fault will occur.

- **Precautions**
  - It is recommended to execute this instruction in a non-real-time task.
  - Do not use the MC_GenerateCamTable instruction during the execution of this instruction.
  - Absolute path writing is not supported.
- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates that the instruction has been executed successfully, and the parameter Busy indicates that the current function block is being executed.

  After this instruction is called, the output parameter Busy is set to TRUE on a rising edge of Execute. After the edited cam table is stored, the output parameter Done shows a value TRUE.

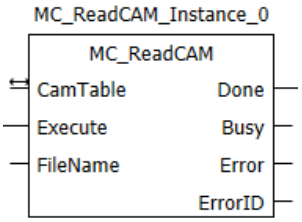


- **Faults**
  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.4.12 MC_ReadCAM

This instruction reads a cam table file, loads the file at runtime, and makes it available for the other instructions (MC_CamIn).

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_ReadCAM	Read the cam data file into a cam table		MC_ReadCAM_Instance_0(CamTable :=, Execute :=, FileName :=, Done =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
CamTable	Cam table	18.3.3.2.3.1 MC_CAM_TABLE_REF: Cam table structure	-	-	Specify the cam table to be stored.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
FileName	File name	STRING	-	''	Cam table storage location.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**
  - This instruction reads a cam table file, loads the file at runtime, and makes it available for the other instructions (MC_CamIn).
  - This instruction is executed on a rising edge of the input variable Execute.

    FileName

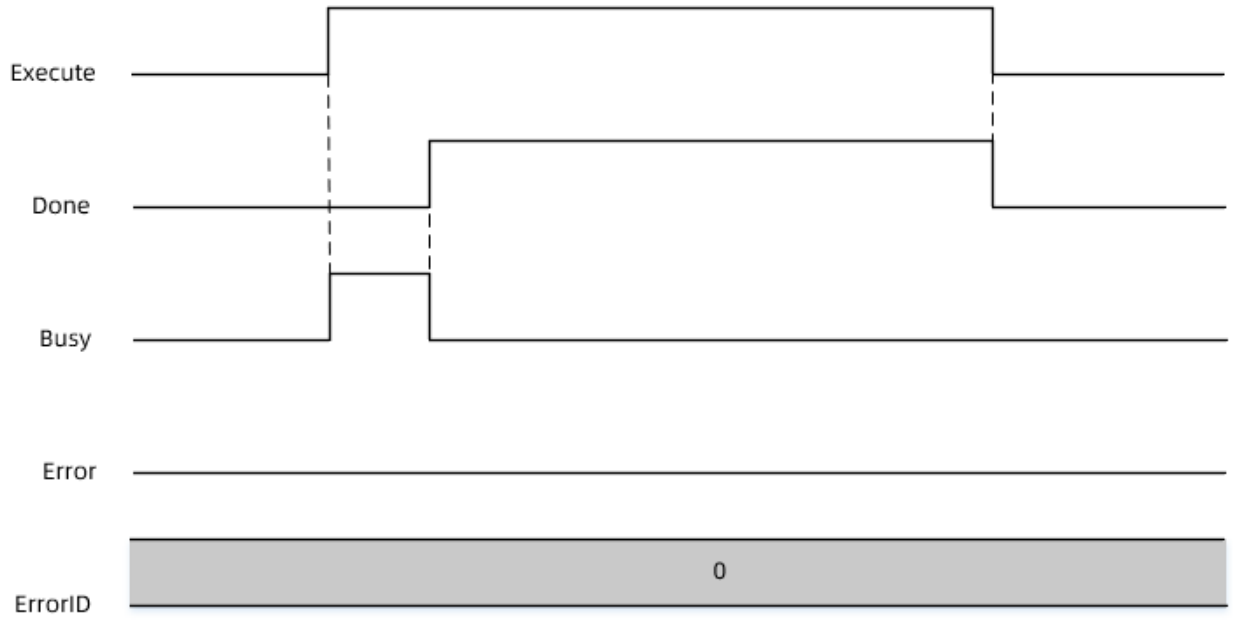
  - The input variable FileName of this instruction specifies the file name to be written to CamTable. If another file is read, a fault will occur.
  - When read/write is executed, a fixed suffix ".cam" is added internally.
  - The input variable FileName of this instruction cannot contain Chinese characters; otherwise, a fault will occur.

    Node type

  - The node type of CamTable must be key or dense nodes; otherwise, a fault will occur.
  - When key nodes are specified for CamTable, the number of nodes must be between 2 and 20000; otherwise, a fault will occur.
  - When dense nodes are specified for CamTable, the number of nodes must be between 2 and 65535; otherwise, a fault will occur.
- **Precautions**
  - It is recommended to execute this instruction in a non-real-time task.
  - The number of nodes for the in-out variable CamTable of this instruction cannot be less than that in the cam table file to be read.
  - The node type for the in-out variable CamTable of this instruction must match the node type in the cam table file being read; otherwise, a fault will occur.
- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates that the instruction has been executed successfully, and the parameter Busy indicates that the current function block is being executed.

  After this instruction is called, the output parameter Busy is set to TRUE on a rising edge of Execute. After the cam data file is read into a cam table, the output parameter Done is set to TRUE.

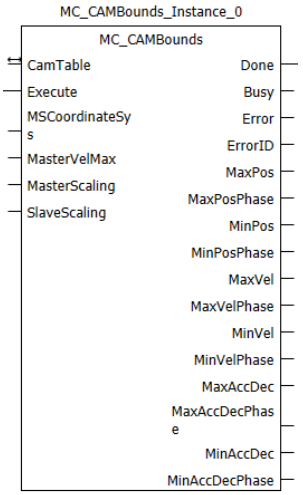


- **Faults**
  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.4.13 MC_CAMBounds

This instruction calculates the position, velocity and acceleration values of the slave axis being coupled to a master axis, which is moved according to a specified maximum velocity. It is used to check the curve when a cam table is created.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_CAMBounds	Calculate the maximum values of the slave axis motion parameters		MC_CAMBounds_Instance_0(CamTable :=, Execute :=, MSCoordinateSys :=, MasterVelMax :=, MasterScaling :=, SlaveScaling :=, Done =>, Busy =>, Error =>, ErrorID =>, MaxPos =>, MaxPosPhase =>, MinPos =>, MinPosPhase =>, MaxVel =>, MaxVelPhase =>, MinVel =>, MinVelPhase =>, MaxAccDec =>, MaxAccDecPhase =>, MinAccDec =>, MinAccDecPhase =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
CamTable	Cam table	18.3.3.2.3.1 MC_CAM_TABLE_REF: Cam table structure	-	-	Specify the cam table to be stored.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
MScoordinateSys	MS coordinate system	BOOL	TRUE, FALSE	FALSE	FALSE: Physical coordinate system TRUE: Master-slave coordinate system
MasterVelMax	Maximum velocity	LREAL	Positive number	0	Maximum velocity of the master axis.
MasterScaling	Master axis scaling	LREAL	Positive number	1	Scaling factor for the master axis applied to cam.
SlaveScaling	Slave axis scaling	LREAL	Positive number	1	Scaling factor for the slave axis applied to cam.

Output variables

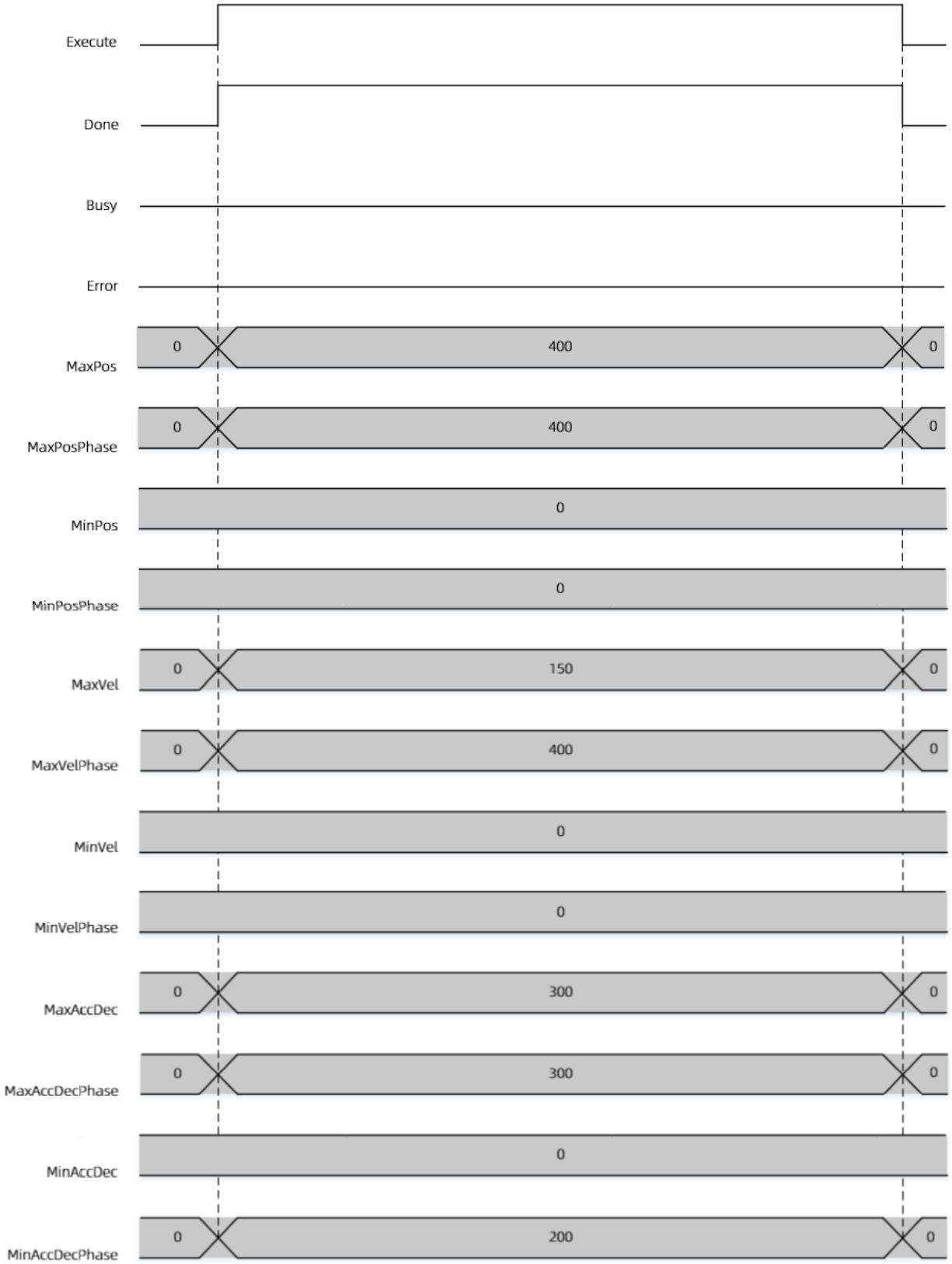
Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
MaxPos	Maximum position	LREAL	-	0	Calculate the maximum position of the slave axis based on the cam table.
MaxPosPhase	Phase at the maximum position	LREAL	-	0	Master axis phase at the maximum displacement of the cam curve.
MinPos	Minimum position	LREAL	-	0	Calculate the minimum position of the slave axis based on the cam table.
MinPosPhase	Phase at the minimum position	LREAL	-	0	Master axis phase at the minimum displacement of the cam curve.
MaxVel	Maximum velocity	LREAL	-	0	Maximum velocity.
MaxVelPhase	Phase at the maximum velocity	LREAL	-	0	Master axis phase at the maximum velocity of the cam curve.
MinVel	Minimum velocity	LREAL	-	0	Minimum velocity.
MinVelPhase	Phase at the minimum velocity	LREAL	-	0	Master axis phase at the minimum velocity of the cam curve.
MaxAccDec	Maximum acceleration	LREAL	-	0	Maximum acceleration.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
MaxAccDecPhase	Phase at the maximum acceleration	LREAL	-	0	Master axis phase at the maximum acceleration of the cam curve.
MinAccDec	Minimum acceleration	LREAL	-	0	Minimum acceleration.
MinAccDecPhase	Phase at the minimum acceleration	LREAL	-	0	Master axis phase at the minimum acceleration of the cam curve.

- **Function**
  - This instruction is executed on a rising edge of the input variable Execute.
  - You can use this instruction to calculate information such as the velocity and acceleration of the slave axis, regardless of the states of the master and slave axes.
- **Precautions**
  - If the in-out variable CamTable of this instruction is in read/write or update state, a fault will occur.
  - If the in-out variable CamTable of this instruction is user-defined, the MC_GenerateCamTable (Update the cam table with specified cam data) instruction must be used to check the validity of the cam table. After the check results are acceptable, this instruction can be executed to calculate information such as the velocity and acceleration of the slave axis; otherwise, a fault will occur.
  - If the input variables MasterScaling and/or SlaveScaling of this instruction are less than or equal to 0, they will be automatically corrected to 1.
  - When other input variables of this instruction are specified improperly, a fault occurs.
  - When dense nodes are specified for the in-out variable CamTable of this instruction, a fault occurs.
- **Timing diagram**

  When the parameter “Execute” is activated, the parameter “Busy” shows the value TRUE. After the position, velocity, and acceleration of the slave axis are calculated, the parameter Done shows the value TRUE.

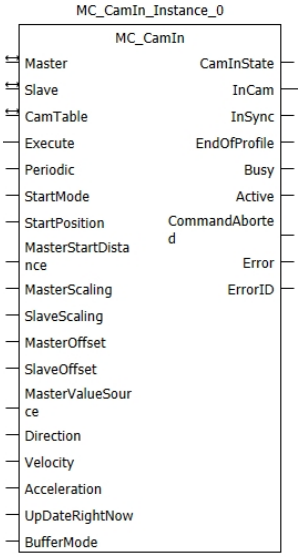


- **Faults**
  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.4.14 MC_CamIn

This instruction controls the camming of the specified master and slave axis according to a specified cam table.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_CamIn	Start camming		MC_CamIn_Instance_0(Master :=, Slave :=, CamTable :=, Execute :=, Periodic :=, StartMode :=, StartPosition :=, MasterStartDistance :=, MasterScaling :=, SlaveScaling :=, MasterOffset :=, SlaveOffset :=, MasterValueSource :=, Direction :=, Velocity :=, Acceleration :=, UpDateRightNow :=, BufferMode :=, CamInState =>, InCam =>, InSync =>, EndOfProfile =>, Busy =>, Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Master	Master axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the master axis; an instance of AXIS_REF.
Slave	Slave axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the slave axis; an instance of AXIS_REF.

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
CamTable	Cam table	18.3.3.2.3.1 MC_CAM_TABLE_REF: Cam table structure	-	-	Specify the cam table for camming.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
Periodic	Periodic	INT	Positive number or 0	0	Specify the number of executing the cam table. 0: Periodically. Others: Number of cycles.
StartMode	Start position mode	INT	0, 1, 2	0	0: Absolute position. 1: Relative position. 2: Immediate start.
StartPosition	Cam table start position	LREAL	Positive number, negative number, or 0	0	Start position of the cam table.
MasterStartDistance	Master axis position	LREAL	Positive number, negative number, or 0	0	Position [u] of the master axis when the slave axis starts camming.
MasterScaling	Master axis scaling	LREAL	Positive number	1	Extend or contract the master axis phase with the specified scale.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
SlaveScaling	Slave axis scaling	LREAL	Positive number	1	Extend or contract the slave axis phase with the specified scale.
MasterOffset	Master axis offset	LREAL	Positive number, negative number, or 0	0	Shift the phase of the master axis with the specified offset.
SlaveOffset	Slave axis offset	LREAL	Positive number, negative number, or 0	0	Shift the displacement of the slave axis with the specified offset.
MasterValueSource	Master axis data type	18.3.3.2.2.11 MC_REFERENCE_TYPE: Data Type Enumeration	0: mcCommand 1: mcFeedback 2: mcLatestCommand	0	0: Value calculated in the previous task cycle (fSavePosition). 1: Value obtained in the same task cycle (fActPosition). 2: Value calculated in the same task cycle (fSetPosition).
Direction	Direction	18.3.3.2.2.9 MC_DIRECTION: Motion Direction Enumeration	1: DIR_POSITIVE 2: DIR_NEGATIVE 4: DIR_NODIRECTION	1	1: Positive direction. 2: Negative direction. 4: No direction specified.
Velocity	Velocity	LREAL	Positive number	-	Maximum superimposed velocity during dynamic coupling.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Acceleration	Acceleration/ deceleration	LREAL	Positive number or 0	0	Maximum superimposed acceleration during smooth coupling. The input value of 0 indicates that coupling is disabled and the slave axis velocity directly jumps to the sum of the camming velocity and coupling velocity. After the coupling ends, the superimposed velocity is directly removed.
UpDateRightNow	Immediate effect flag	BOOL	TRUE, FALSE	TRUE	TRUE: Take effect immediately. FALSE: Take effect in the next cycle.
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting 1: mcBuffered	0	0: Aborting. 1: Buffered.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
CamInState	Camming state	18.3.3.2.3.5 MC_CAM_IN_STATE_REF: Camming state structure	-	-	Describe the current execution state of the cam table, including master and slave axis IDs, master axis phase, slave axis displacement, number of cycles executed, etc. See the variable description for structure information.
InCam	Camming	BOOL	TRUE, FALSE	FALSE	TRUE when the master axis passes the start point of the cam table.
InSync	Synchronizing	BOOL	TRUE, FALSE	FALSE	TRUE when the slave axis starts camming.
EndOfProfile	End of camming cycle	BOOL	TRUE, FALSE	FALSE	TRUE when the end point of the cam table is used as the position reference, and FALSE in the next cycle.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Controlling	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted by another instruction.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity	Velocity	Any velocity that exceeds the motion parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	The velocity equal to 0 will be automatically corrected to 1e-5.
Acceleration	Acceleration	Any acceleration that exceeds the motion parameter limit will be automatically corrected to the acceleration limit.	Any acceleration less than 0 will be automatically corrected to 10% of the acceleration limit.	No correction.

- **Function**

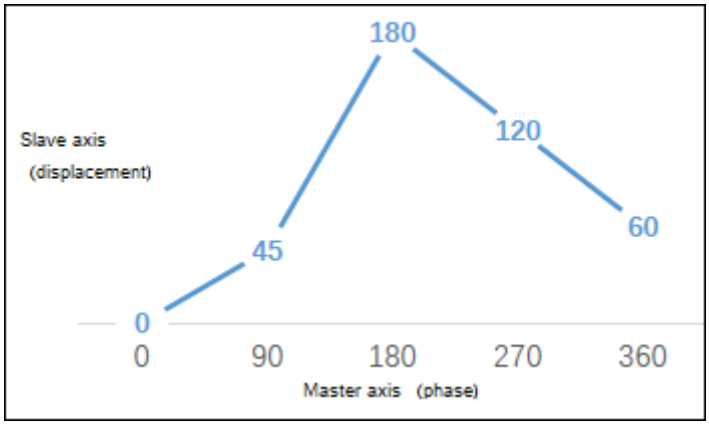
  Bus-controlled axis:

  - This instruction synchronizes the camming of the master and slave axes according to the cam table.
  - This instruction is executed on a rising edge of the input variable Execute.
  - The master and slave axes executing the camming cannot be the same.
  - This instruction can be used in combination with other synchronous control instructions. Its in-out variable Master can be the master or slave axis of other synchronous control instructions; its in-out variable Slave can serve as the master axis of other synchronous control instructions.
  - During the execution of this instruction, the MC_MoveSuperImposed instruction can be used.

- When the instruction is triggered on a rising edge of the input parameter Execute, the input parameters are checked for validity. If any of invalid parameters is detected, it will be automatically corrected by the function block with no fault message; any buffered instructions after this instruction will be cleared.
- The cam table specified by the in-out variables CamTable of this instruction must be edited within the cam editor and downloaded to the controller in advance. When it is a user-defined cam table, the MC_GenerateCamTable instruction must be used to check the validity of the cam table before executing the MC_CamIn instruction.

Cam table 1

Master Axis (Phase)	Slave Axis (Displacement)
0	0
90	45
180	180
270	120
360	60

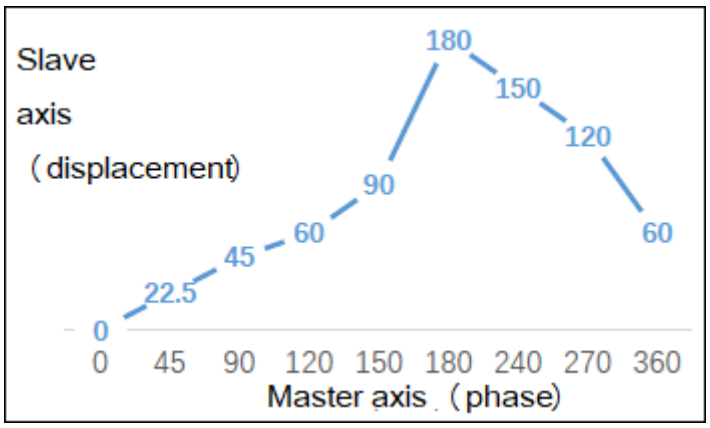


Position references of the master and slave axes executing camming (large phase interval)

Cam table 2

Master Axis (Phase)	Slave Axis (Displacement)
0	0
45	22.5
90	45
120	60
150	90
180	180
240	150
270	120

Master Axis (Phase)	Slave Axis (Displacement)
360	60



Position references of the master and slave axes executing camming (small phase interval)

- The phase of the master axis and the displacement of the slave axis in the cam table start from "0". The minimum interval between adjacent phases cannot be less than 0.0001.
- The relative mode is set for the cam table, which means all values in the cam table are relative to the start point of the table.
- If bus cycle is specified for the cycle task of the master axis, the velocity of the slave axis may show a sudden jump after synchronization. It is recommended that users specify EtherCAT cycle for the cycle task of both the master and slave axes before enabling a synchronous control instruction.
- MC_Home, MC_Homing, or MC_HomingWithParameter can be used during the instruction execution.

  This instruction places the master axis in the AXIS_HOMING state. If the instruction is executed while the master axis is already in this state, the signal source "MasterValueSource" must be set to "mcLatestCommand"; otherwise, the slave axis will overspeed.

Software limit

- During camming, if the master axis position is beyond the software limit, the master axis velocity will decelerate to 0 within one cycle, without affecting the synchronization state of the slave axis. At this point, if the master axis is reset and restarts its motion, the slave axis will remain position synchronized.
- During camming, if the slave axis position is beyond the software limit, the instruction will be aborted, and the slave axis will decelerate to 0 within one cycle with the state switching from SynchronizedMotion to ErrorStop.

Re-execution

This instruction can be re-executed:

1. You can re-trigger this instruction only after modifying the parameters Periodic, Direction, MasterValueSource, MasterScaling, and SlaveScaling. When you re-execute the instruction after modifying other parameters, a fault occurs. After the instruction is re-executed, the modified parameters will be checked for validity.
2. The changes to the parameters will take effect in the next camming cycle after the instruction is re-executed.
3. If the instruction is re-executed, the number of executed camming cycles will be recalculated.

4. If any instructions are buffered after this instruction, the buffered instructions will be aborted after this instruction is re-executed, and must be re-buffered.
5. After the instruction is re-executed with the modified value of Direction, if the motion direction of the master axis is opposite to the modified Direction after the current camming cycle ends, the slave axis will perform an emergency stop and start to follow the master axis when the master axis direction matches the active following direction.
6. When this instruction is re-executed, a new camming cycle will start directly with StartMode = 2 at the end of the current camming cycle.
7. When the MC_CamIn instruction is being executed with a triggered phase superposition instruction, re-executing the MC_CamIn instruction will abort the phase superposition instruction.
8. During the execution of the MC_CamIn instruction, if this instruction is re-executed with a new cam table, it will be executed according to the new cam table and re-executed parameters after the current cycle ends.

Conditions for executing the instruction

- This instruction can be executed when the master axis (Master) specified by the instruction is in the ErrorStop or PowerOff state.
- This instruction can be triggered when the slave axis specified by the instruction is not in the ErrorStop or PowerOff state. If the slave axis is in the STOPPING or HOMING state, the instruction will be aborted.

Start of camming

After the instruction is executed, the master axis has to reach the StartPosition. After the master axis passes the StartPosition, the cable table start point is executed with the output flag InCam showing the value TRUE, which means that the synchronous motion between the master and slave axes is about to start at the current position. For example, if the count mode of the master axis is 0° to 360° in Rotary mode, the StartPosition is 60. The absolute position of the master axis is the phase of the cam table plus the value of the StartPosition. The absolute position of the slave axis is the displacement from the cam table plus the absolute position of the slave axis at the cam table start point.

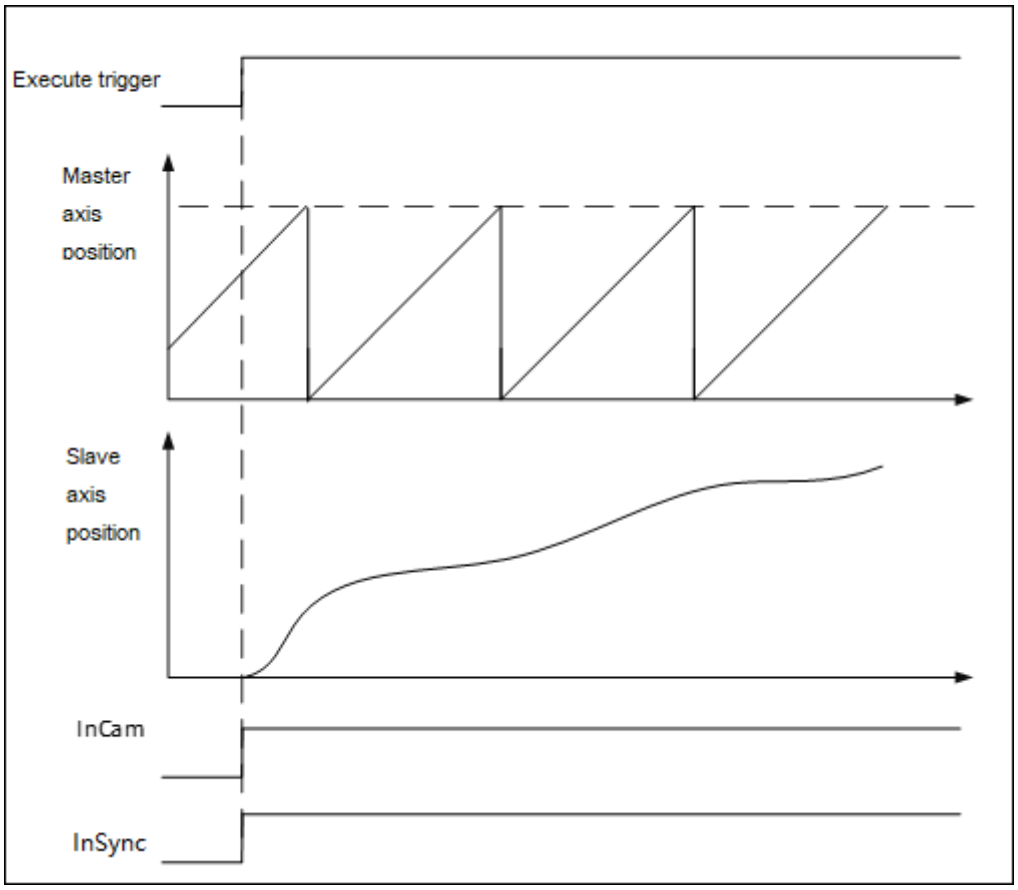
Absolute positions of the master and slave axes in the cam table 3

Master Axis (Phase)	Slave Axis (Displacement)
60	0
150	45
240	180
330	120
60	60

- **StartMode**

  You can specify the start mode through StartMode for this instruction. There are three start modes, two of which specify the parameter MasterStartDistance: one is Relative Position mode, which specifies the start position of the camming to be calculated based on the cam table start point; the other is Absolute Position mode, which directly specifies the start position of the camming. The last mode is Immediate Start mode, which means that the camming starts immediately after the instruction is triggered.

(1) For Immediate Start mode, the output flags InCam and InSync of MC_CamIn are set as shown in the following figure:



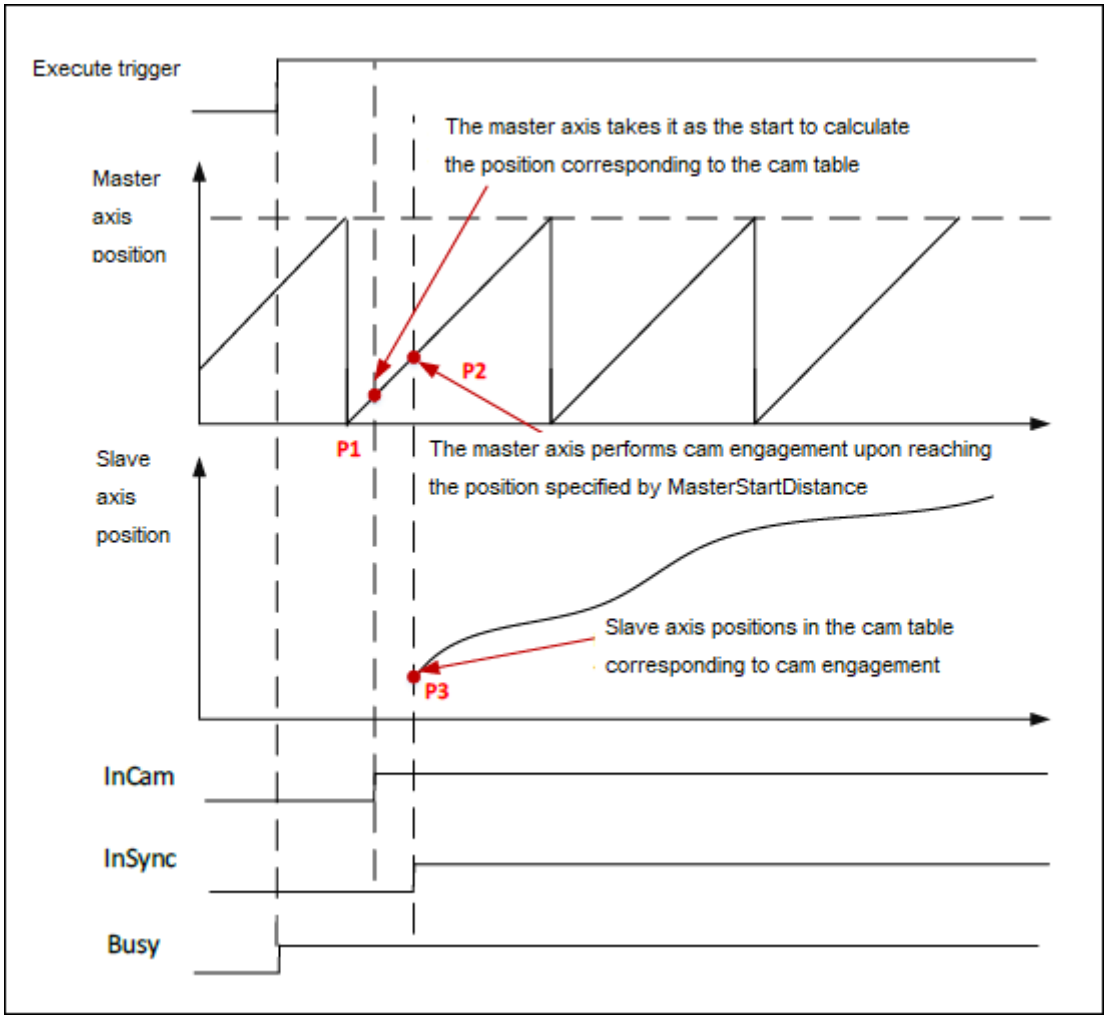
(2) You can start camming with specified positions in Relative Position mode (StartMode = 1) and Absolute Position mode (StartMode = 0).

After the instruction is executed, the output variable InCam shows the value TRUE when the master axis reaches the StartPosition.

StartMode = 0: When the master axis moves in the specified motion direction and reaches the MasterStartDistance, the slave axis starts to follow the master axis, and the output variable InSync shows the value TRUE.

StartMode = 1: When the master axis moves in the specified motion direction and reaches the MasterStartDistance + StartPosition position, the slave axis starts to follow the master axis, and the output variable InSync shows the value TRUE.

The execution of the cam table is shown in the following figure:



◦ **InCam and InSync**

- When the master axis is specified as a rotary axis:

  If the value of StartPosition is greater than one cycle length, the start position will be identified as the StartPosition modulo the cycle length.

  If the value of MasterStartDistance is greater than one camming cycle and the StartMode is set to 0, the InSync position will be identified as the MasterStartDistance modulo the camming cycle.

- With certain parameter combinations, the InSync cannot be set to TRUE. Specifically, this includes the following cases:

  1. Absolute mode:

  a. The master axis is specified as a linear axis; the value of StartPosition is greater than the value of MasterStartDistance; the master axis motion direction is specified as DIR_POSITIVE.

  b. The master axis is specified as a linear axis; the value of StartPosition is less than the value of MasterStartDistance; the master axis motion direction is specified as DIR_NEGATIVE.

  2. Relative mode:

  a. The value of MasterStartDistance is less than 0; the master axis motion direction is specified as DIR_POSITIVE.

b. The value of MasterStartDistance is greater than 0; the master axis motion direction is specified as DIR_NEGATIVE.

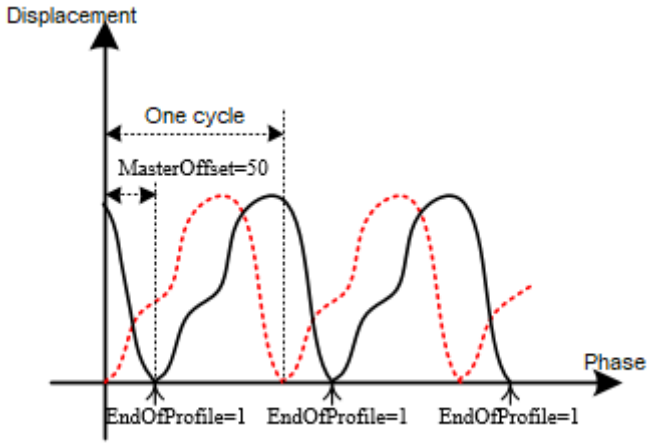
- The InSync will be set to TRUE only when the master axis moves in the specified direction and passes the specified synchronization position.
- The InCam will be set to TRUE only when the master axis moves in the specified direction and passes the specified cam table start point.
- The InSync is set only after the InCam has been set, or both InSync and InCam may be set simultaneously.
- When the StartMode is set to 2, the InSync and InCam instructions are set to TRUE after the MC_CamIn is executed, and this start mode does not trigger the validity check of the MasterStartDistance, as this parameter has no actual effect. In this mode, whether to enable coupling depends on the values of MasterOffset, SlaveOffset, and Acceleration.

◦ **MasterOffset and SlaveOffset**

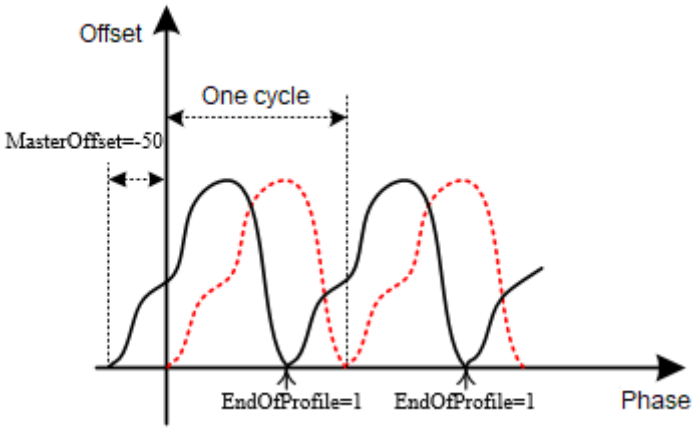
After the StartPositon of the cam is determined, the cam is scaled proportionally and then applied with an offset, the value of which is not affected by the scaling factor.

If the value of the input variable MasterOffset is greater than one camming cycle, the modulo operation will be performed.

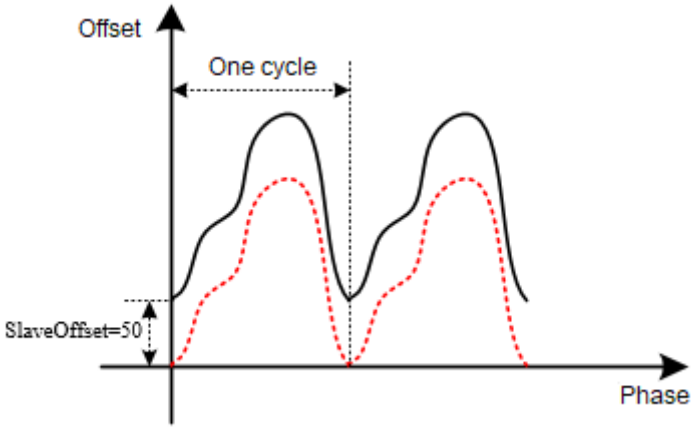
MasterOffset > 0



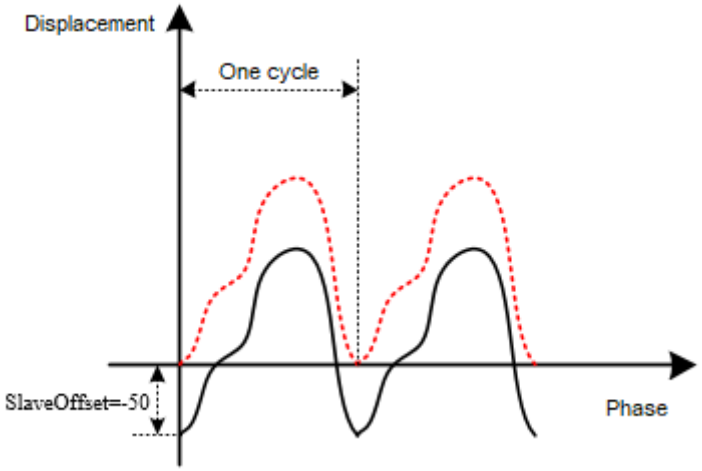
MasterOffset < 0



SlaveOffset > 0

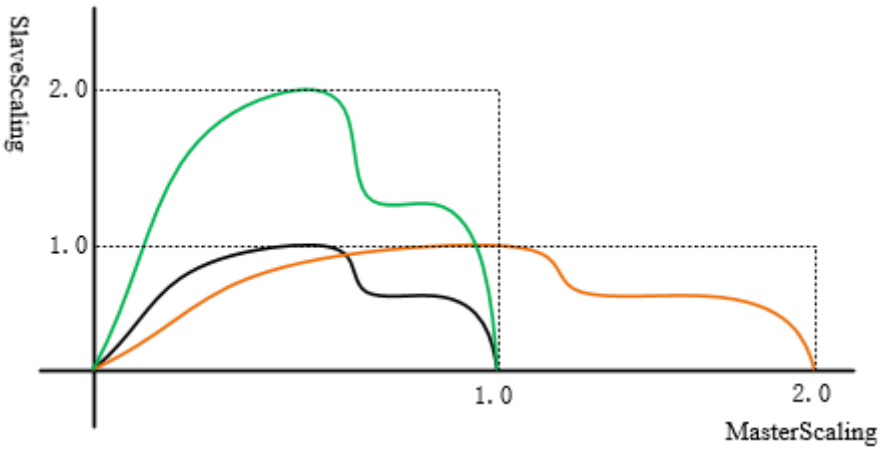


SlaveOffset < 0



- **MasterScaling and SlaveScaling**

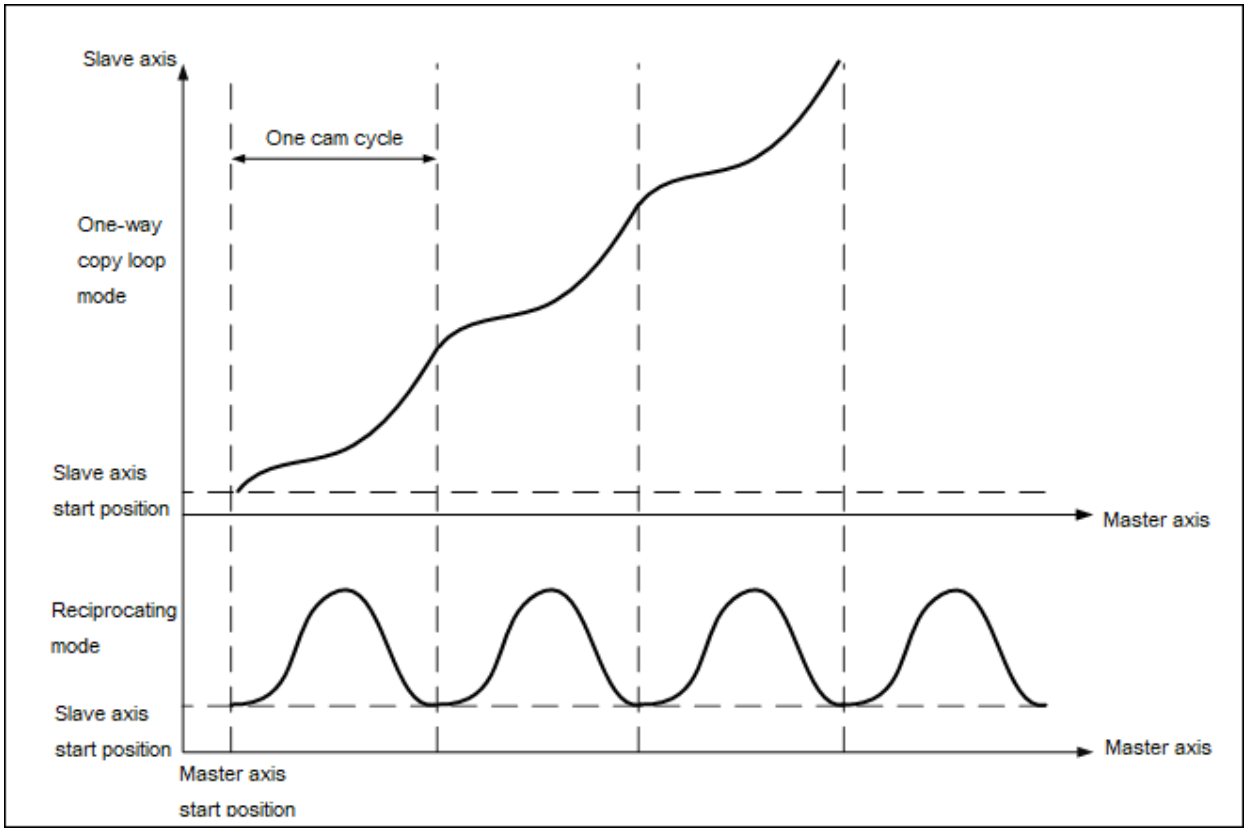
  The master axis phase and slave axis displacement can be scaled according to a specified scale. Different scaling coefficients can be set for the master axis and slave axis, respectively. If the input variables MasterScaling and/or SlaveScaling of this instruction are less than or equal to 0, they will be automatically corrected to 1.



- **Periodic**

  You can specify the number of times the MC_CamIn instruction is executed by defining the value of the input variable Periodic. If this variable is set to 0, the instruction will be executed periodically. In this case, there are two types of motion for the slave axis: one is reciprocating motion, and the other is unidirectional contour following motion.

When the Periodic is set to N, the slave axis automatically ends synchronization after the specified cycles have been completed. If the slave axis still moves at the velocity of non-zero after the cam table is executed, it will perform an immediate stop.



Changes in the master axis phase and slave axis displacement during periodic execution

- **EndOfProfile**

  The output variable EndOfProfile is set to TRUE when the cam table end point is used as the position reference, and changes to FALSE in the next cycle.

- **End of camming**

  To abort camming midway, you can execute the MC_CamOut, MC_Stop, or MC_ImmediateStop instruction on the slave axis.

- **Position type selection**

  You can select the MasterValueSource for synchronizing the slave axis based on the following options.

  - mcCommand: Commanded position (Value calculated in the previous task cycle, fSavePosition).

    For the current cycle, use the master axis position calculated in the previous task cycle.

  - mcFeedback: Actual position (Value obtained in the same task cycle, fActPosition).

    Use the actual master axis position obtained in the same task cycle.

  - mcLaterCommand: Commanded position (Value calculated in the same task cycle, fSetPosition).

    Use the master axis position calculated in the same task cycle.

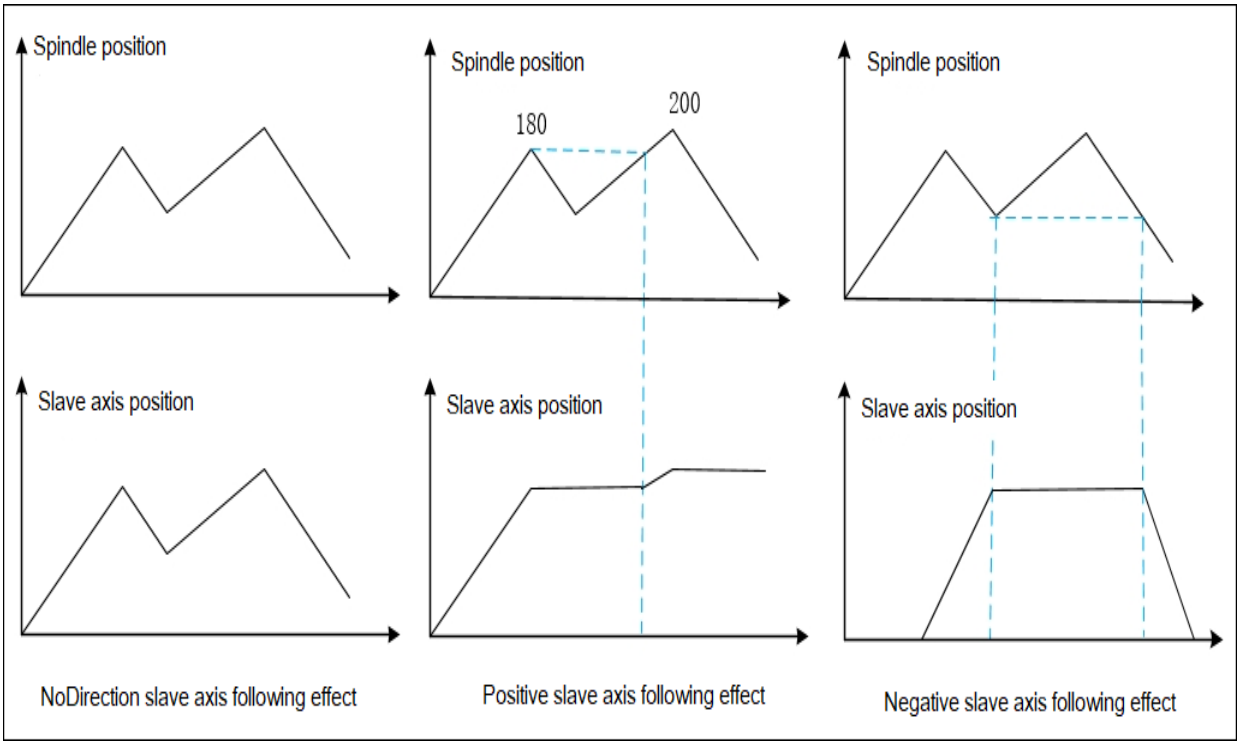
- **Direction**

  The slave axis executes camming when the master axis moves in the specified direction. In other words, the camming operation is started for the slave axis only when the master axis moves in the direction specified by Direction.

DIR_NODIRECTION: Camming starts regardless of whether the motion direction of the master axis.

DIR_POSITIVE: Camming starts when the master axis moves in the positive direction.

DIR_NEGATIVE: Camming starts when the master axis moves in the negative direction.



- **Precautions**
  - When this instruction is buffered with Active not being set, you can re-execute the instruction after modifying parameters.
  - During the execution of this instruction, the master and slave axes cannot be changed; otherwise, a fault occurs.
  - If the StartMode of a buffered MC_CamIn instruction is set to Immediate start, the buffered instruction will immediately execute the corresponding cam table following the end of the previous MC_CamIn instruction.
  - During smooth coupling, there may be instances where the slave axis does not strictly follow the camming of the master axis.
- **Example**

  This instruction is a synchronous motion control instruction. The cam motion is based on a master axis and one or more slave axes. The master axis can be a physical axis or a virtual axis.

  When the master axis is a virtual axis, the motion is called electronic camming. Electronic camming enables convenient definition of machining trajectories with simple changes to the cam table, eliminating the need to frequent mechanical cam replacement.

  The following shows the steps for using synchronization instructions:

  ① Specify the master and slave axes for synchronous motion based on the specific application scenario. The master axis for a synchronous motion control instruction can be a slave axis of another synchronous motion control instruction. The slave axis of one synchronous motion control instruction can also serve as the master axis of another synchronous motion control instruction.

  ② Based on the specific application scenario, determine the position relationship between the master and slave axes, and create a cam table. In multiple different instantiated MC_CamIn instructions,

when Axis is specified as the master axis and Axis_1 is specified as the slave axis, the cam table for each instruction can be different.

③ Use the MC_CamIn instruction to bind the master axis, slave axis, and cam table.

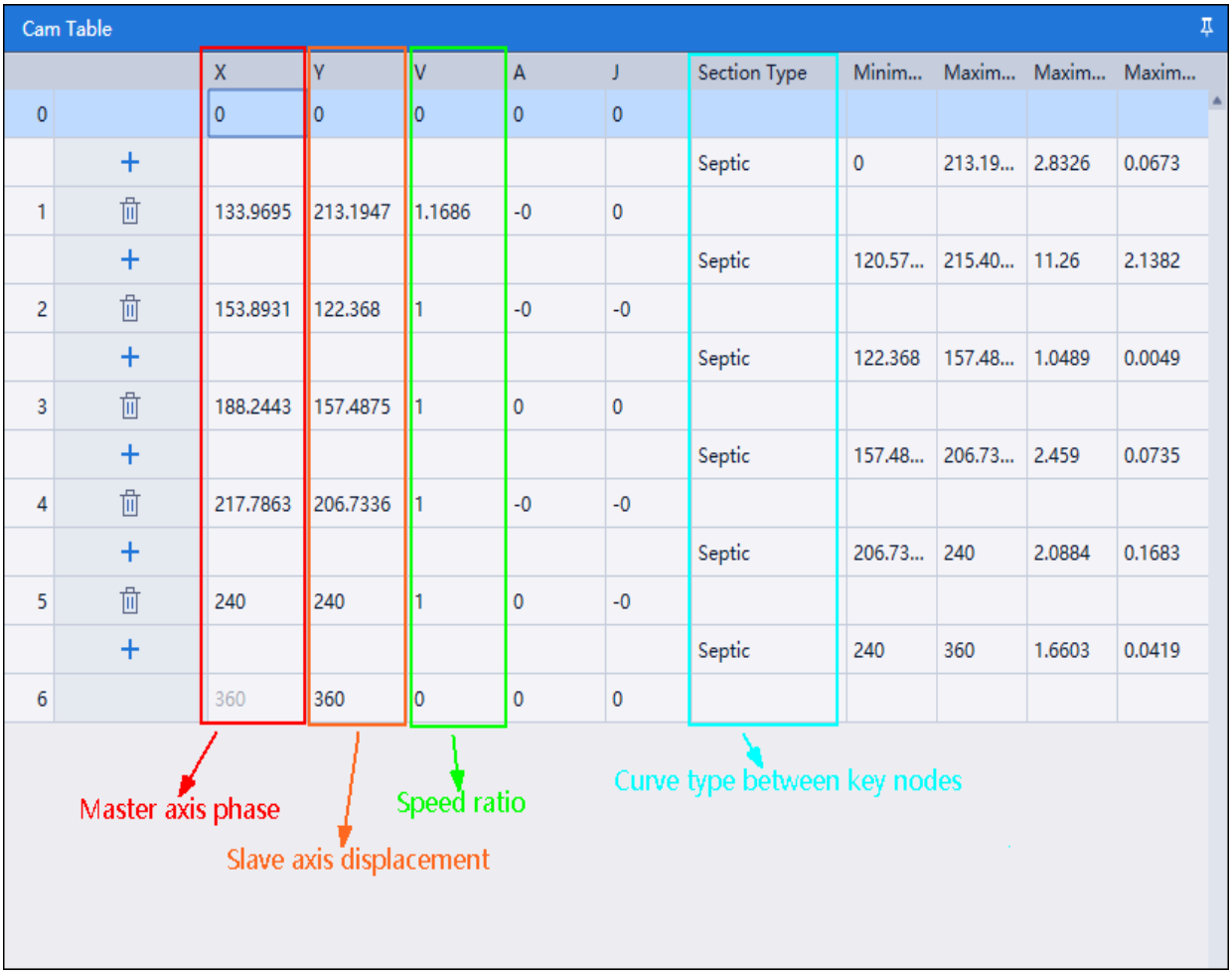
④ Based on the specific application scenario, specify the input parameters of the MC_CamIn instruction and execute this instruction.

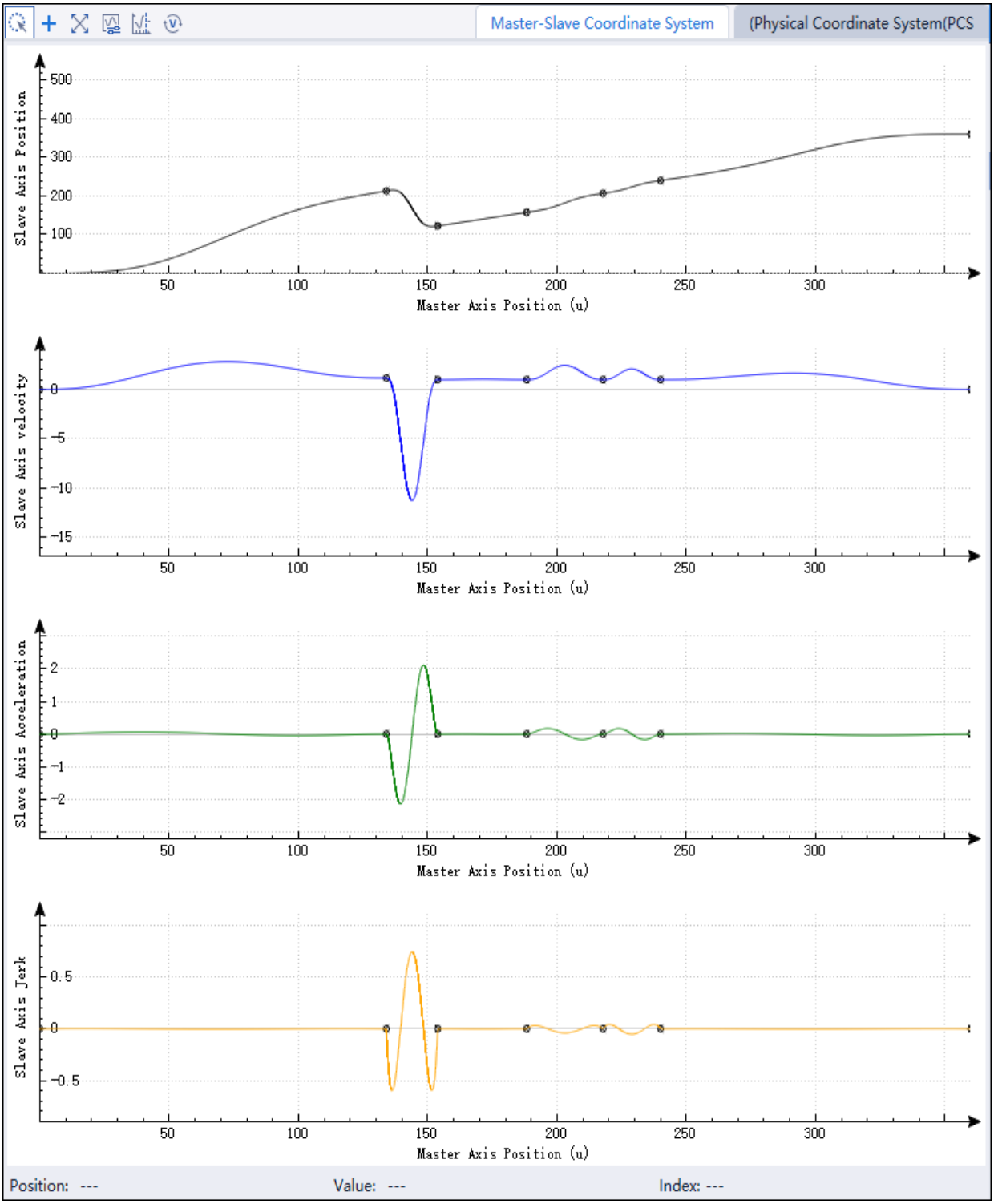
⑤ Wait for the synchronous motion control instruction to complete, or use the MC_CamOut instruction to end the synchronization between the master and slave axes.

In the following program example, the MC_CamIn instruction is executed with the parameter UpDateRightNow defaulting to FALSE.

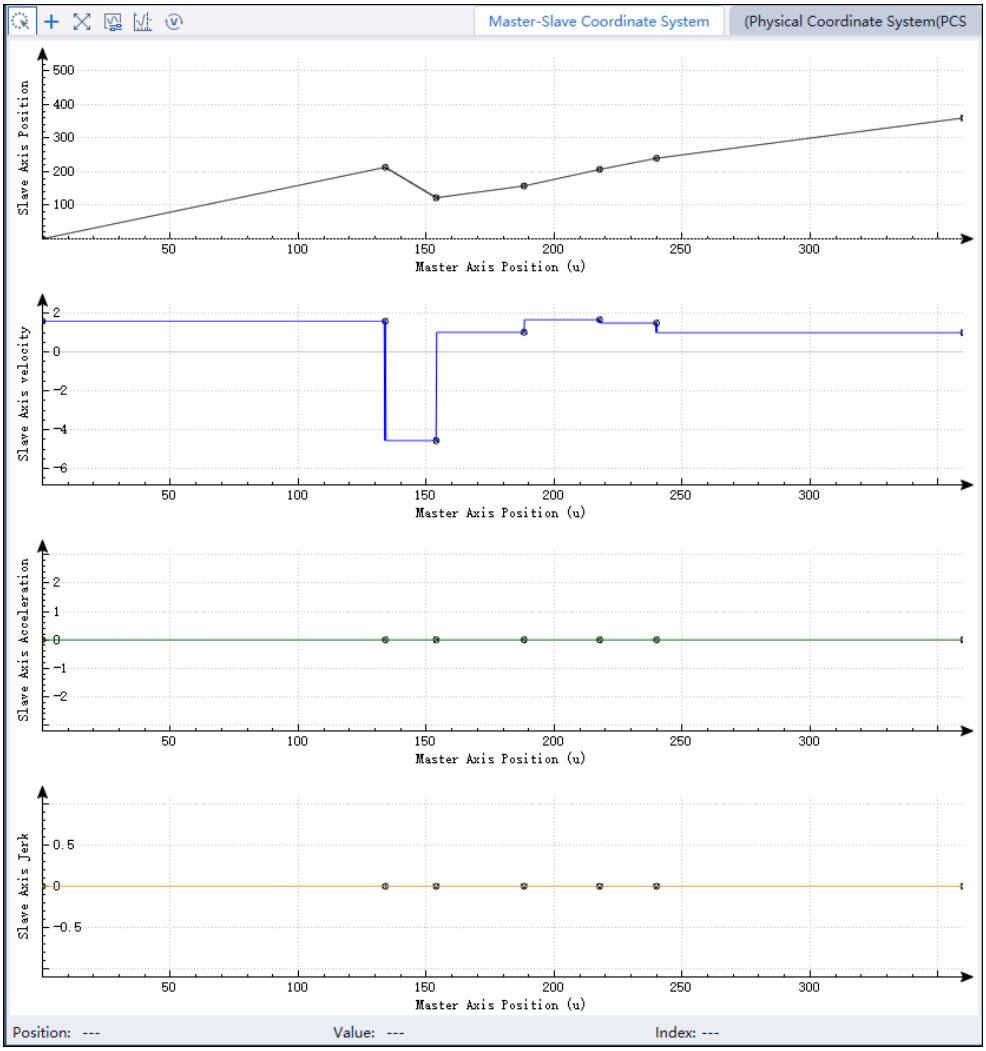
The created cam table automatically generates the position relationship between the master and slave axes, and the velocity information of the slave axis at key nodes. You can adjust the data in the cam table to fit specific application scenarios.

When the master axis phase and the slave axis displacement are adjusted, the velocity ratio is automatically adjusted based on the curve type between key nodes.

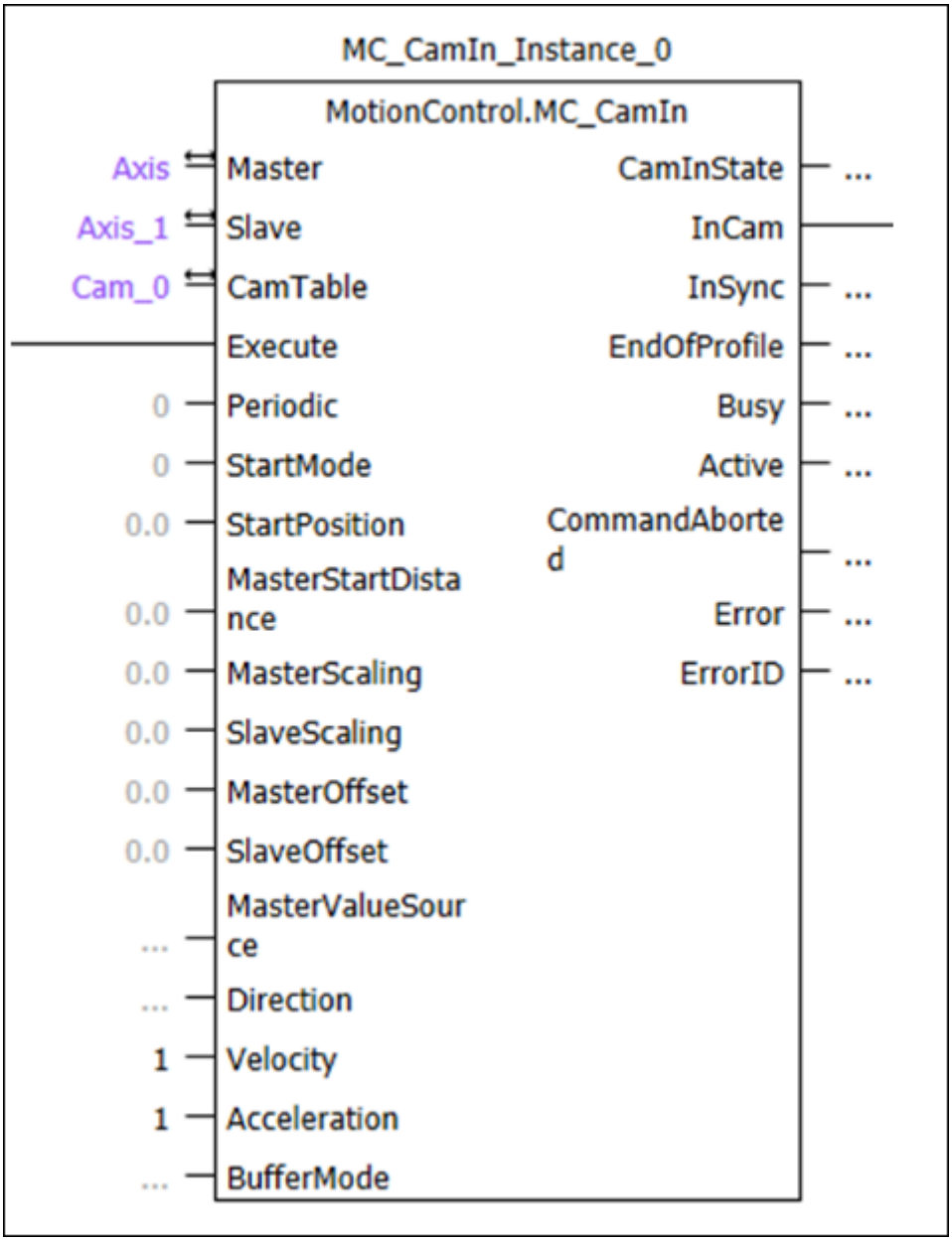




When the curve type between key nodes is changed to "Straight line", the velocity, acceleration, and jerk of the slave axis also change.

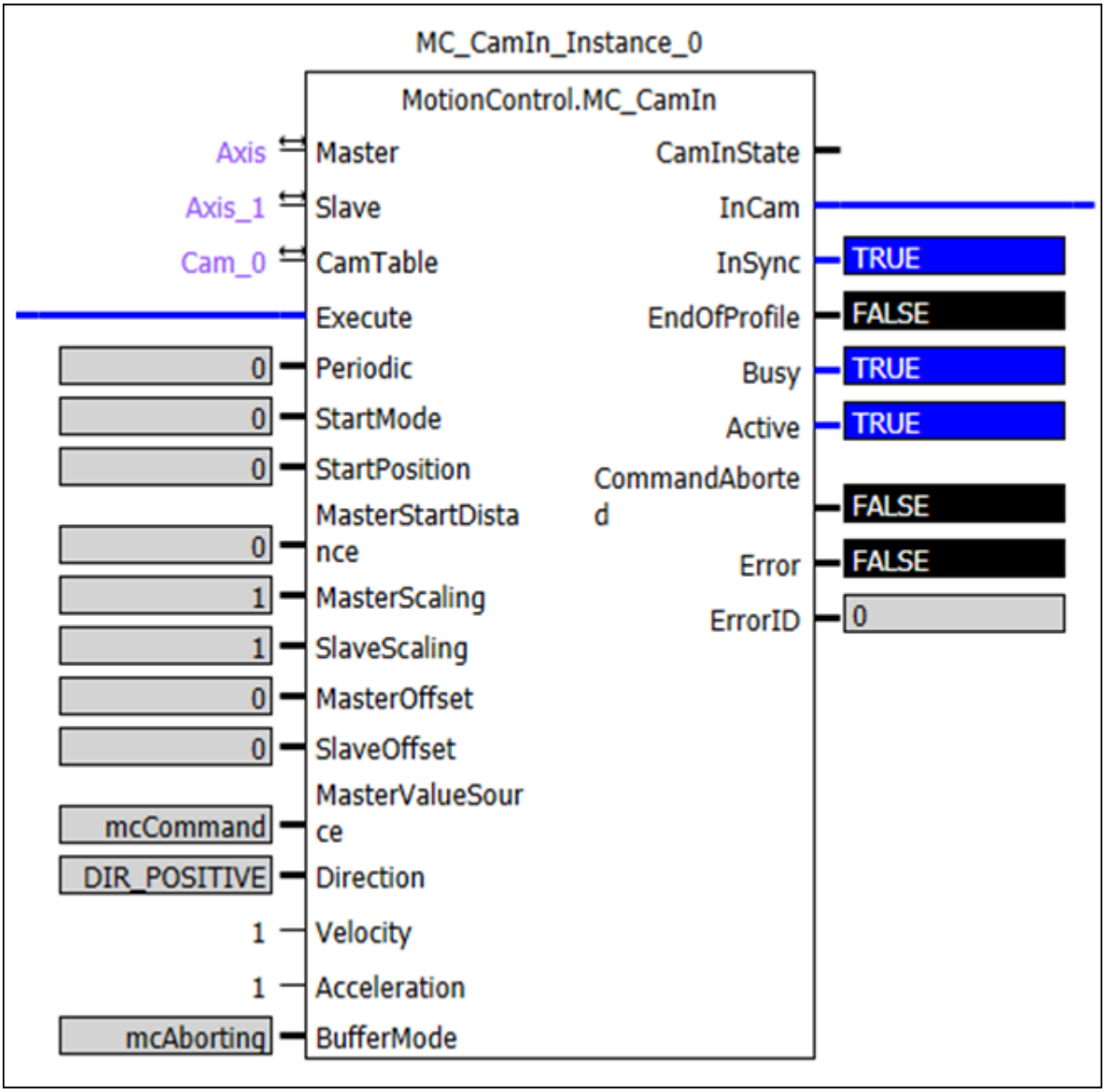


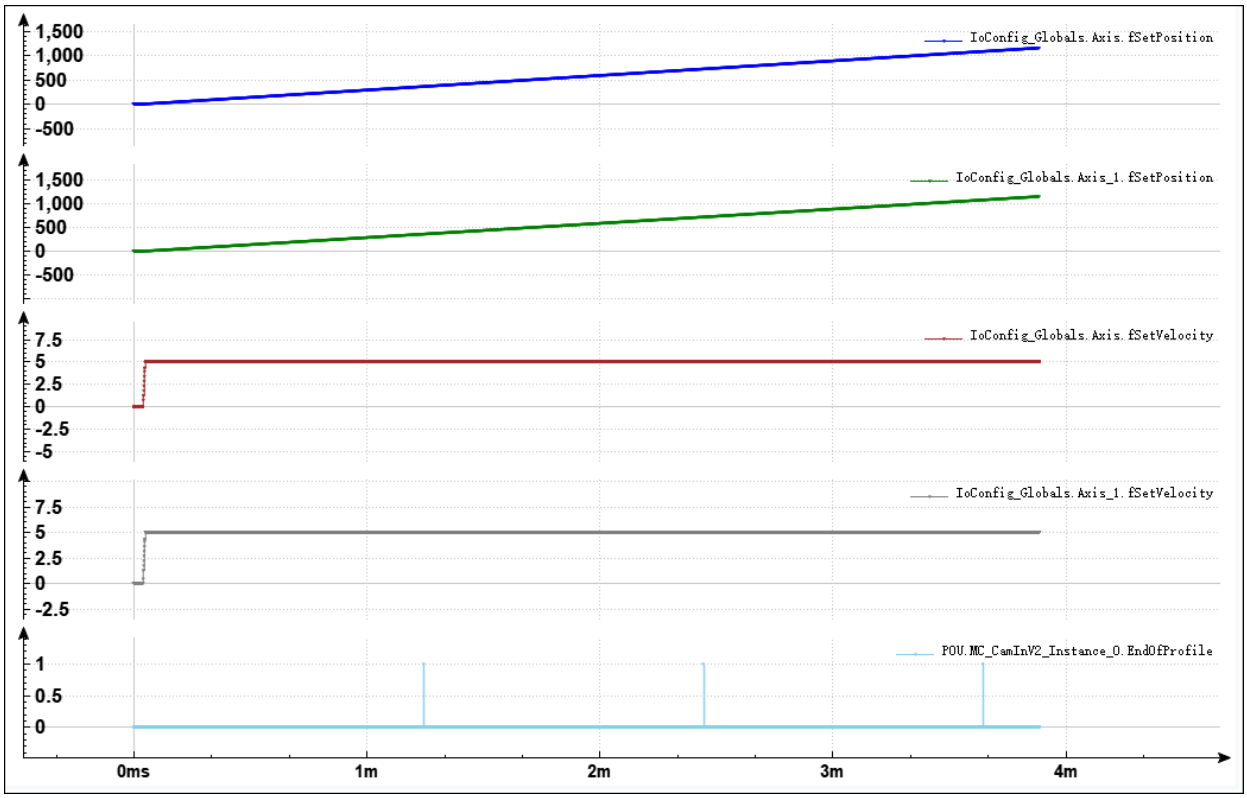
Create a cam table, specify the position relationship between the master and slave axes, and specify the input parameters of the MC_CamIn instruction according to specific application scenarios.



You can specify the number of times the cam table is executed through the parameter Periodic. When the Periodic is set to 0, it indicates that this instruction is executed repeatedly according to the cam table. When the Periodic is set to a specific value, this instruction is executed according to the specified number of cycles.

Periodic = 0:

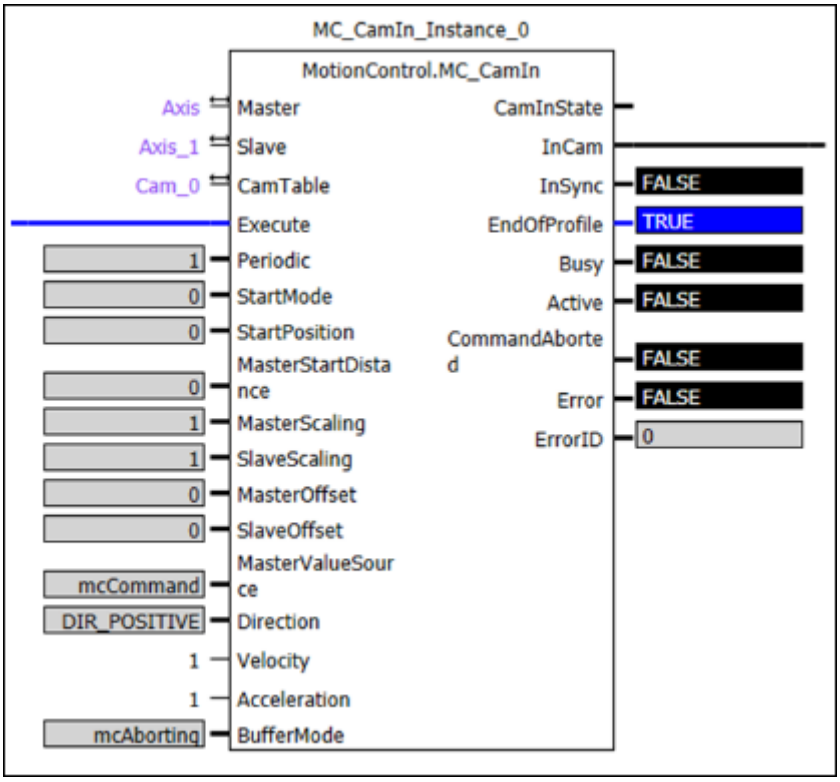

MC_CamIn_Instance_0
MotionControl.MC_CamIn
Axis
Master
Axis_1
Slave
Cam_0
CamTable
Execute
0
Periodic
0
StartMode
0
StartPosition
0
MasterStartDistance
1
MasterScaling
1
SlaveScaling
0
MasterOffset
0
SlaveOffset
mcCommand
MasterValueSource
DIR_POSITIVE
Direction
1
Velocity
1
Acceleration
mcAborting
BufferMode
CamInState
InCam
InSync
TRUE
EndOfProfile
FALSE
Busy
TRUE
Active
TRUE
CommandAborted
FALSE
Error
FALSE
ErrorID
0


1,500
1,000
500
0
-500
IoConfig_Globals.Axis.fSetPosition
IoConfig_Globals.Axis_1.fSetPosition
7.5
5
2.5
0
-2.5
-5
IoConfig_Globals.Axis.fSetVelocity
IoConfig_Globals.Axis_1.fSetVelocity
1
0.5
0
POU.MC_CamInV2_Instance_0.EndOfProfile
0ms
1m
2m
3m
4m

The EndOfProfile is set once for each execution of the cam table, and then reset in the next cycle.

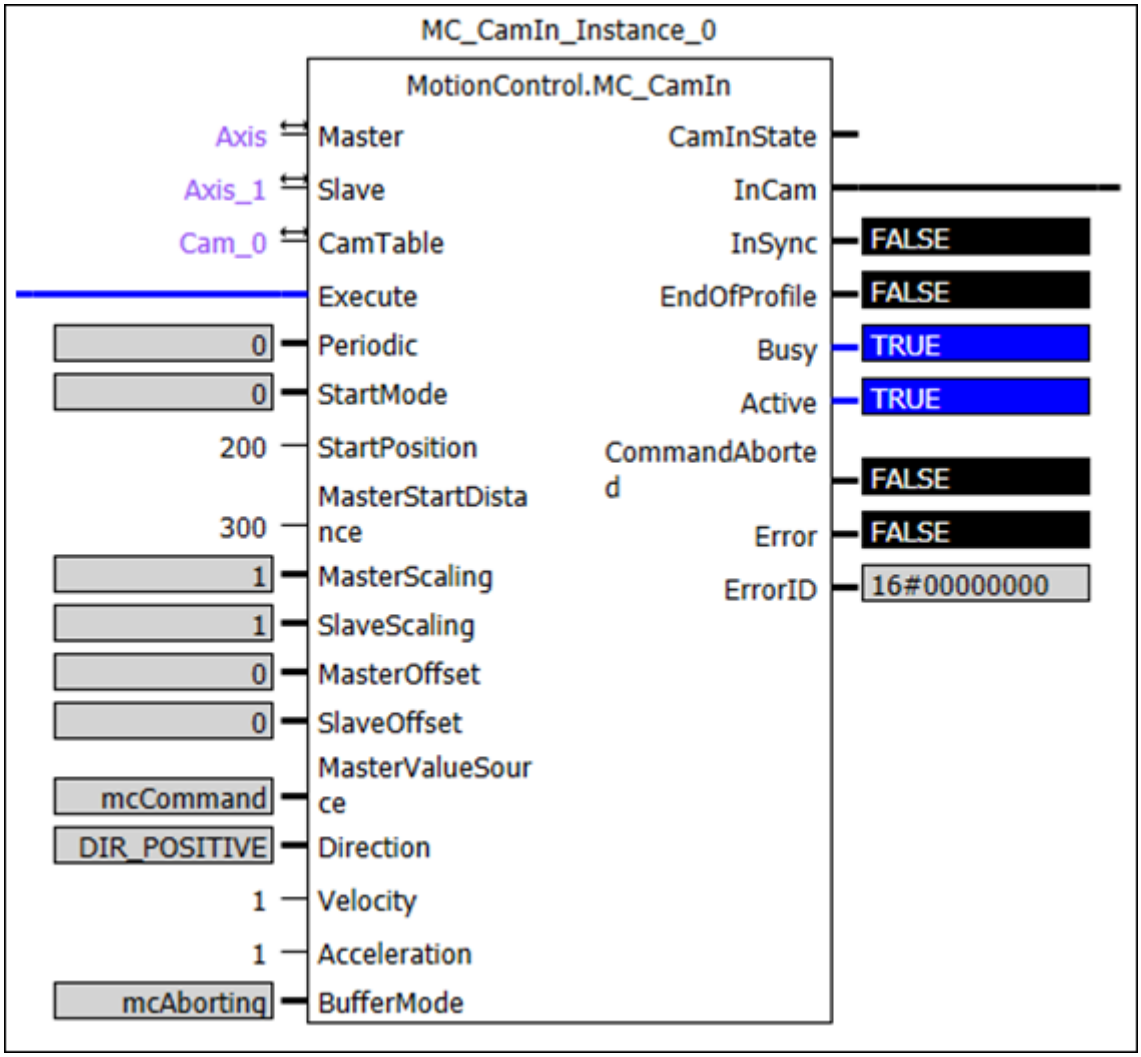
Periodic = 1:

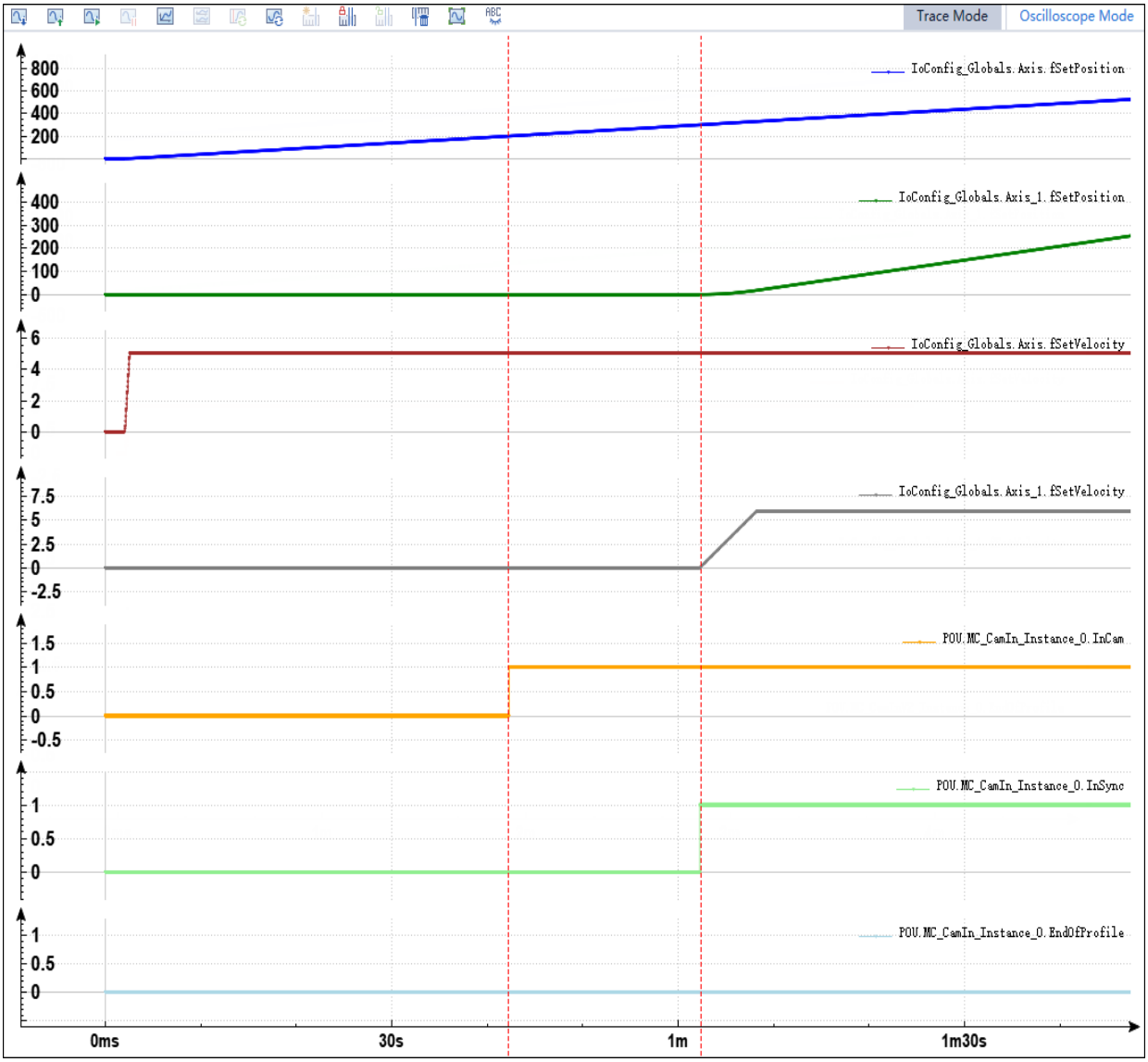
The EndOfProfile is set once for each execution of the cam table, and then reset in the next cycle. After the last cycle ends, the EndOfProfile is kept set.

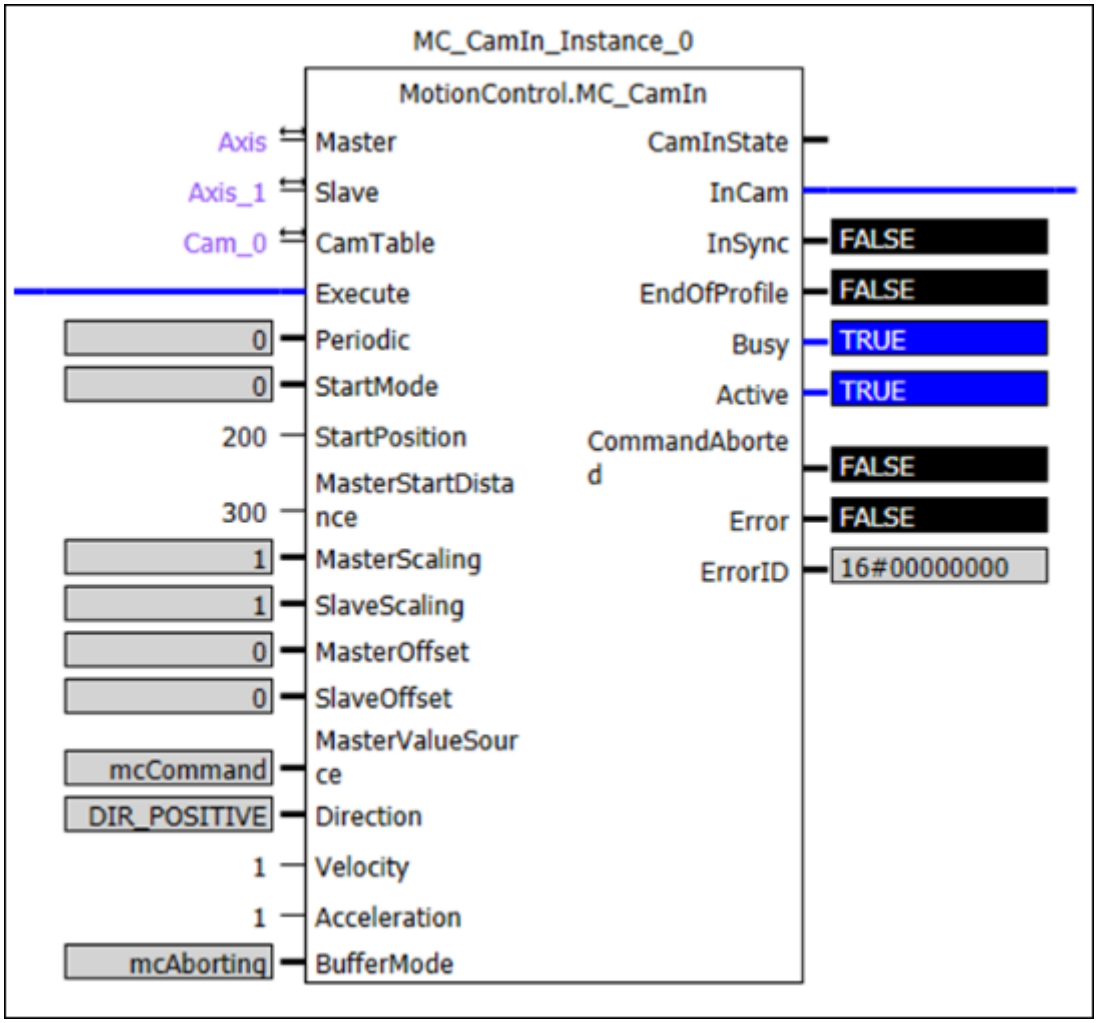


StartPosition: The value of StartPosition indicates the start point of the cam table for the absolute position of the master axis. If the master axis position does not match the specified value of StartPosition, the InCam will not be set.

MasterStartDistance : The value of MasterStartDistance indicates the position of the master axis when the slave axis starts camming. If the master axis position does not match the specified value of MasterStartDistance, the InSync will not be set, and the slave axis will not start moving. When the value of StartPosition is greater than the value of MasterStartDistance, a fault occurs.


MC_CamIn_Instance_0
MotionControl.MC_CamIn
Axis
Master
Axis_1
Slave
Cam_0
CamTable
Execute
0
Periodic
0
StartMode
200
StartPosition
300
MasterStartDistance
1
MasterScaling
1
SlaveScaling
0
MasterOffset
0
SlaveOffset
mcCommand
MasterValueSource
DIR_POSITIVE
Direction
1
Velocity
1
Acceleration
mcAborting
BufferMode
CamInState
InCam
InSync
FALSE
EndOfProfile
FALSE
Busy
TRUE
Active
TRUE
CommandAborted
FALSE
Error
FALSE
ErrorID
16#00000000


Trace Mode
Oscilloscope Mode
IoConfig_Globals.Axis.fSetPosition
IoConfig_Globals.Axis_1.fSetPosition
IoConfig_Globals.Axis.fSetVelocity
IoConfig_Globals.Axis_1.fSetVelocity
POU.MC_CamIn_Instance_0.InCam
POU.MC_CamIn_Instance_0.InSync
POU.MC_CamIn_Instance_0.EndOfProfile
800
600
400
200
400
300
200
100
0
6
4
2
0
7.5
5
2.5
0
-2.5
1.5
1
0.5
0
-0.5
1
0.5
0
1
0.5
0
0ms
30s
1m
1m30s



For the slave axis specified as a rotary axis:

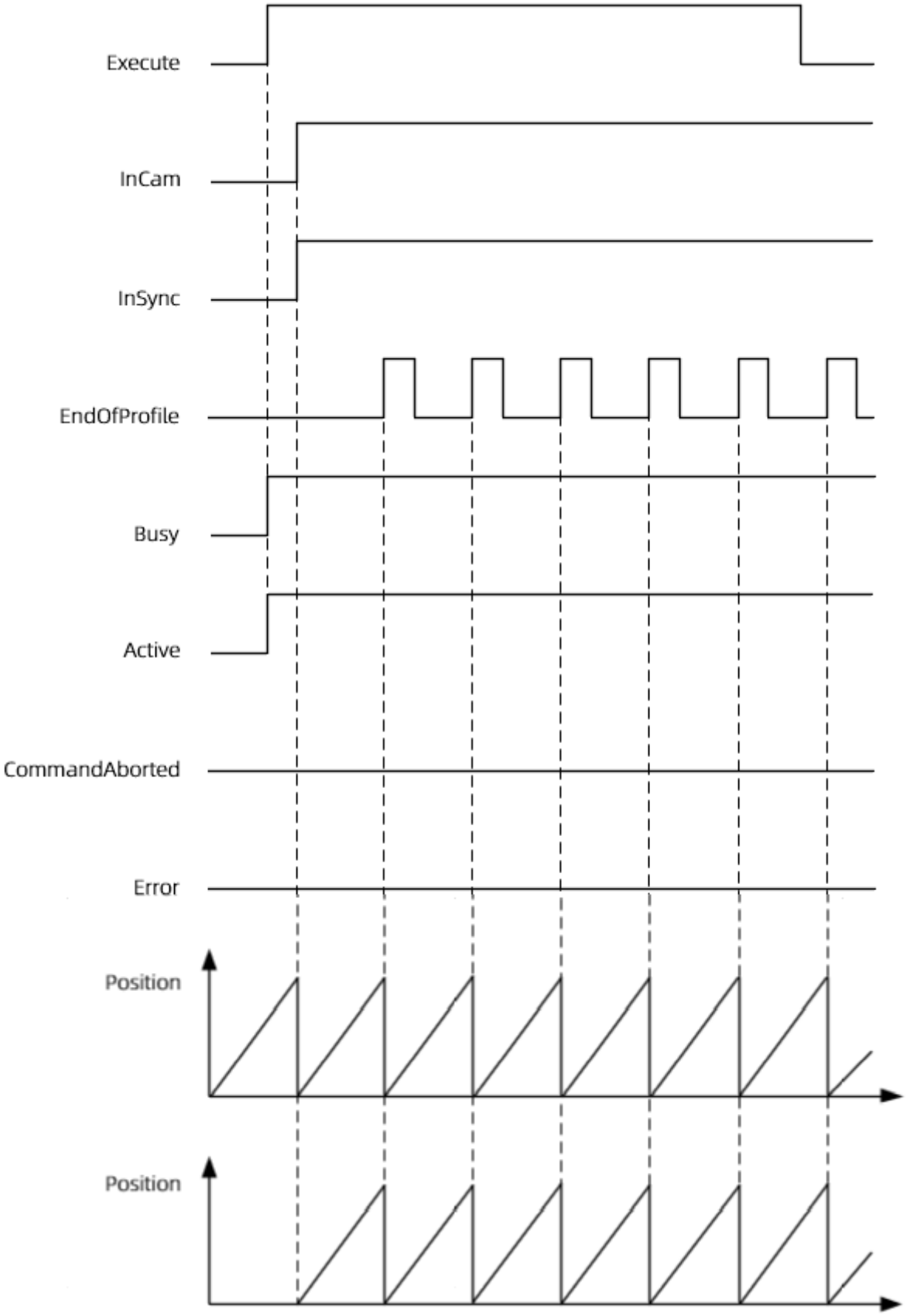


Local pulse axis:

- This instruction can be used for local pulse axes.
- When the local pulse axis is used as the master axis with Y0 or Y1 single-ended output, it is not recommended to use this instruction. ① No pulse output; ② No position feedback.

- **Timing diagram**
  - During periodic execution

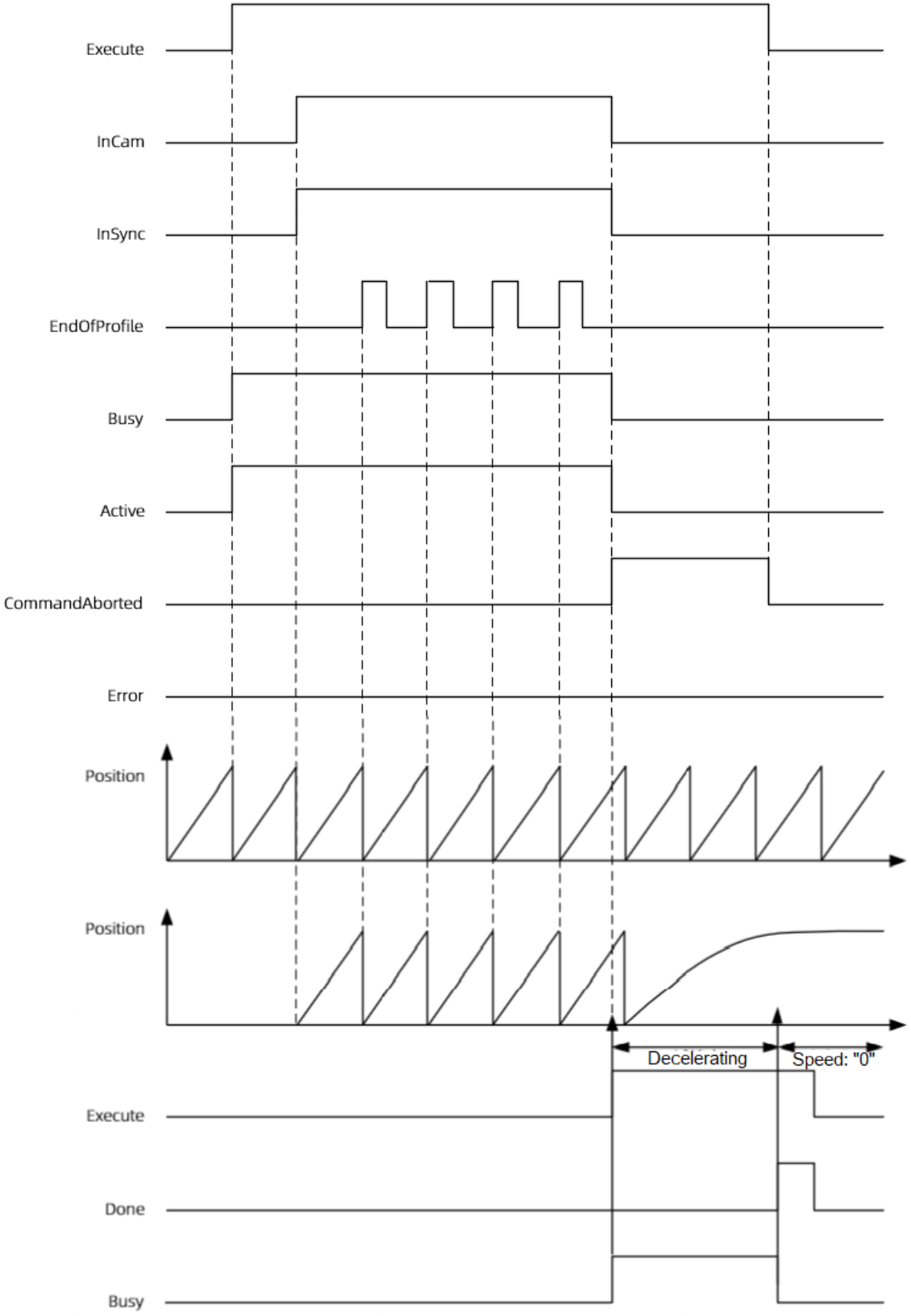
    When the Periodic of the MC_CamIn instruction is set to 0, the timing diagram is shown in the following figure.



◦ With a number of cycles

Camming is repeated. When the MC_CamOut instruction is executed to stop the camming, the slave axis decelerates to 0 without affecting the master axis.

When the Periodic of the MC_CamIn instruction is set to 5 and the MC_CamOut instruction is executed, the timing diagram is shown in the following figure.

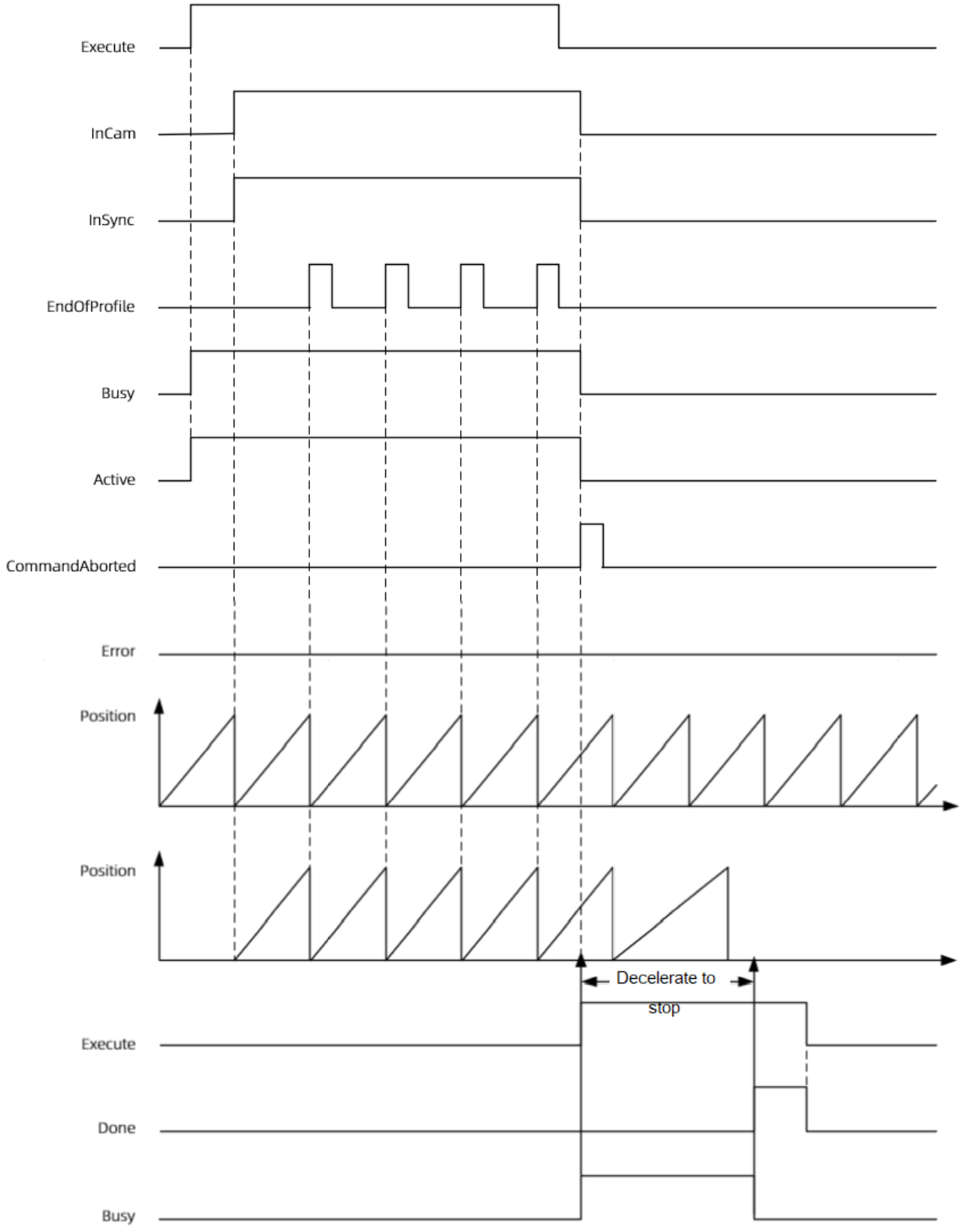


◦ MC_Stop

When the MC_Stop instruction is executed on the master axis during camming, the master and slave axes remain synchronized.

When the MC_Stop instruction is executed on the slave axis during camming, the synchronization between the master and slave axes is terminated.

When the Periodic of the MC_CamIn instruction is set to 5 and the MC_Stop instruction is executed on the slave axis, the timing diagram is shown in the following figure.

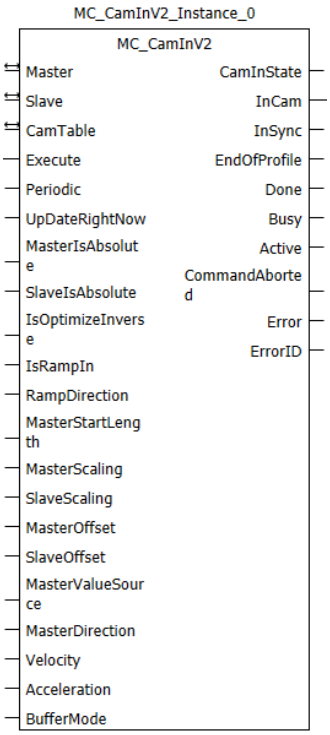


- **Faults**
  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.4.15 MC_CamInV2

This instruction controls the camming of the specified master and slave axes according to a specified cam table.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_CamInV2	Start camming		MC_CamInV2_Instance_0(Master :=, Slave :=, CamTable :=, Execute :=, Periodic :=, UpDateRightNow :=, MasterIsAbsolute :=, SlaveIsAbsolute :=, IsOptimizeInverse :=, IsRampIn :=, RampDirection :=, MasterStartLength :=, MasterScaling :=, SlaveScaling :=, MasterOffset :=, SlaveOffset :=, MasterValueSource :=, MasterDirection :=, Velocity :=, Acceleration :=, BufferMode :=, CamInState =>, InCam =>, InSync =>, EndOfProfile =>, Done =>, Busy =>, Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Master	Master axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the master axis; an instance of AXIS_REF.
Slave	Slave axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the slave axis; an instance of AXIS_REF.
CamTable	Cam table	18.3.3.2.3.1 MC_CAM_TABLE_REF: Cam table structure	-	-	Specify the cam table for camming.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.
Periodic	Periodic	INT	Positive number or 0	0	Specify the number of executing the cam table. 0: Periodically. Other: Number of cycles. Any number less than 0 will result in a fault.
UpDateRightNow	Effective immediately after modifying the cam table data	BOOL	TRUE, FALSE	TRUE	TRUE: Effective immediately after modifying the cam table data. FALSE: Effective in the next cycle after modifying the cam table data.
MasterIsAbsolute	Master axis absolute mode	BOOL	TRUE, FALSE	FALSE	TRUE: The master axis in the cam table is in absolute coordinate mode. FALSE: The master axis in the cam table is in relative coordinate mode.
SlaveIsAbsolute	Slave axis absolute mode	BOOL	TRUE, FALSE	FALSE	TRUE: The slave axis in the cam table is in absolute coordinate mode. FALSE: The slave axis in the cam table is in relative coordinate mode.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
IsOptimizeInverse	Optimize slave axis inverse motion during coupling	BOOL	TRUE, FALSE	FALSE	This parameter is valid only when the IsRampIn is set to TRUE. TRUE: When the actual motion direction of the slave axis is opposite the velocity direction obtained from the cam table during the RampIn, the optimization function is enabled to prevent the slave axis from inverse motion. FALSE: Do not intervene the coupling process.
IsRampIn	Coupling enabled	BOOL	TRUE, FALSE	FALSE	Specify whether to enable coupling when the slave axis starts to follow the master axis. TRUE: Enabled. The slave axis smoothly transits into cam following. FALSE: Disabled. The slave axis directly jumps to the target position.
RampDirection	Dynamic slave axis coupling direction	18.3.3.2.2.9 MC_DIRECTION: Motion Direction Enumeration	0: DIR_SHORTEST 1: DIR_POSITIVE 2: DIR_NEGATIVE	0	This parameter is valid when the IsRampIn is set to TRUE. DIR_SHORTEST: Based on the shortest compensation way. DIR_POSITIVE: Based on the positive compensation value. DIR_NEGATIVE: Based on the negative compensation value. Parameters DIR_POSITIVE and DIR_NEGATIVE are only applicable to rotary axes; all three parameters have the same effect for linear axes.
MasterStartLength	Master axis leading length	LREAL	Positive number or 0	0	After the instruction is triggered, the slave axis starts camming after the master axis moves for the distance specified by MasterStartLength.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
MasterScaling	Master axis scaling	LREAL	Positive number	1	Extend or contract the master axis phase with the specified scale. When the MasterScaling is set to > 1, the phase range is extended; when the MasterScaling is set to < 1, the phase range is contracted. If the setpoint is less than or equal to 0, it will be automatically corrected to 1 by the function block.
SlaveScaling	Slave axis scaling	LREAL	Positive number	1	Extend or contract the slave axis phase with the specified scale. When the SlaveScaling is set to > 1, the phase range is extended; when the SlaveScaling is set to < 1, the phase range is contracted. If the setpoint is less than or equal to 0, it will be automatically corrected to 1 by the function block.
MasterOffset	Master axis offset	LREAL	Positive number, negative number, or 0	0	Shift the phase of the master axis with the specified offset. When the MasterOffset is set to > 0, the cam curve shifts to the right along the coordinate axis; when the MasterOffset is set to < 0, the cam curve shifts to the left along the coordinate axis.
SlaveOffset	Slave axis offset	LREAL	Positive number, negative number, or 0	0	Shift the displacement of the slave axis with the specified offset. When the SlaveOffset is set to > 0, the cam curve shifts upward; when the SlaveOffset is set to < 0, the cam curve shifts downward.
MasterValueSource	Master axis data type	18.3.3.2.2.11 MC_REFERENCE_TYPE: Data Type Enumeration	0: mcCommand 1: mcFeedback 2: mcLatestCommand	0	0: Value calculated in the previous task cycle (fSavePosition). 1: Value obtained in the same task cycle (fActPosition). 2: Value calculated in the same task cycle (fSetPosition).

Input Variable	Name	Data Type	Valid Range	Default Value	Description
MasterDirection	Master axis effective direction	18.3.3.2.3.8 MC_MASTER_DIRECTION: Master Axis Motion Direction Enumeration	0: CAM_NO_DIRECTION 1: CAM_ABS_POSITIVE 2: CAM_REL_POSITIVE 3: CAM_ABS_NEGATIVE 4: CAM_REL_NEGATIVE	0	When the master axis moves in the specified direction, the slave axis starts camming. CAM_NO_DIRECTION: No direction specified, which means the slave axis follows the master axis regardless of the motion direction of the master axis. CAM_ABS_POSITIVE: Absolute positive. The slave axis follows the master axis when the master axis moves in the positive direction, and stops following if the master axis reverses the direction. The master axis must return to positive direction and reach the reversal point for the slave axis to continue following. CAM_REL_POSITIVE: Relative positive. The slave axis follows the master axis when the master axis moves in the positive direction, and stops following if the master axis reverses the direction. The master axis must return to positive direction without reaching the reversal point for the slave axis to continue following. CAM_ABS_NEGATIVE: Absolute negative. The slave axis follows the master axis when the master axis moves in the negative position, and stops following if the master axis reverses the direction. The master axis must return to negative direction and reach the reversal point for the slave axis to continue following. CAM_REL_NEGATIVE: Relative negative. The slave axis follows the master axis when the master axis moves in the negative direction, and stops following if the master axis reverses the direction. The master axis must return to negative direction without reaching the reversal point for the slave axis to continue following.
Velocity	Velocity	LREAL	Positive number	-	Maximum superimposed velocity during dynamic coupling.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Acceleration	Acceleration/deceleration	LREAL	Positive number or 0	0	Maximum superimposed acceleration during smooth coupling.
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting 1: mcBuffered	0	0: Aborting. 1: Buffered.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
CamInState	Camming state	18.3.3.2.3.5 MC_CAM_IN_STATE_REF: Camming state structure	-	-	Describe the current execution state of the cam table, including master and slave axis IDs, master axis phase, slave axis displacement, number of cycles executed, etc.
InCam	Camming	BOOL	TRUE, FALSE	FALSE	TRUE when the cumulative movement of the master axis reaches the MasterStartLength value after the instruction is executed.
InSync	Synchronizing	BOOL	TRUE, FALSE	FALSE	TRUE when the slave axis fully follows the cam curve after coupling. When IsRampIn is set to FALSE, InSync and InCam are set to TRUE in the same cycle.
EndOfProfile	End of camming cycle	BOOL	TRUE, FALSE	FALSE	TRUE when the end point of the cam table is used as the position reference, and FALSE in the next cycle.
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE after execution for the specified number of cycles; this flag remains FALSE when Periodic is set to 0.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Controlling	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted by another instruction.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**
  - MC_Home, MC_Homing, or MC_HomingWithParameter can be used during the instruction execution.

    This instruction places the master axis in the AXIS_HOMING state. If the instruction is executed while the master axis is already in this state, the signal source "MasterValueSource" must be set to "mcLatestCommand"; otherwise, the slave axis will overspeed.
  - **Master and slave axis modes of the cam table**

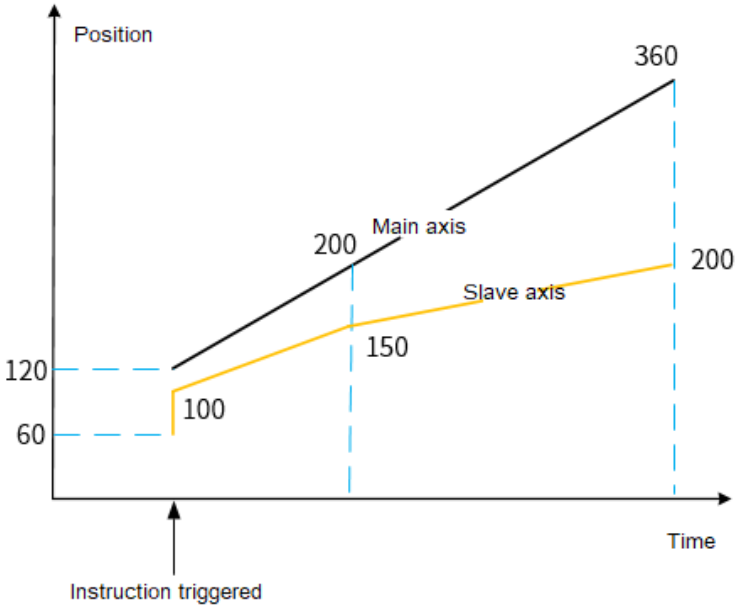
    This instruction supports four master and slave axis position modes. You can specify the master axis mode through the parameter MasterIsAbsolute. When the MasterIsAbsolute is set to TRUE, the master axis is in absolute position mode. When the MasterIsAbsolute is set to FALSE, the master axis is in relative position mode. You can specify the slave axis mode through the parameter SlaveIsAbsolute. When the SlaveIsAbsolute is set to TRUE, the slave axis is in absolute position mode. When the SlaveIsAbsolute is set to FALSE, the slave axis is in relative position mode.

Axis	Mode 1	Mode 2	Mode 3	Mode 4
Master axis	Absolute	Absolute	Relative	Relative
Slave axis	Absolute	Relative	Absolute	Relative

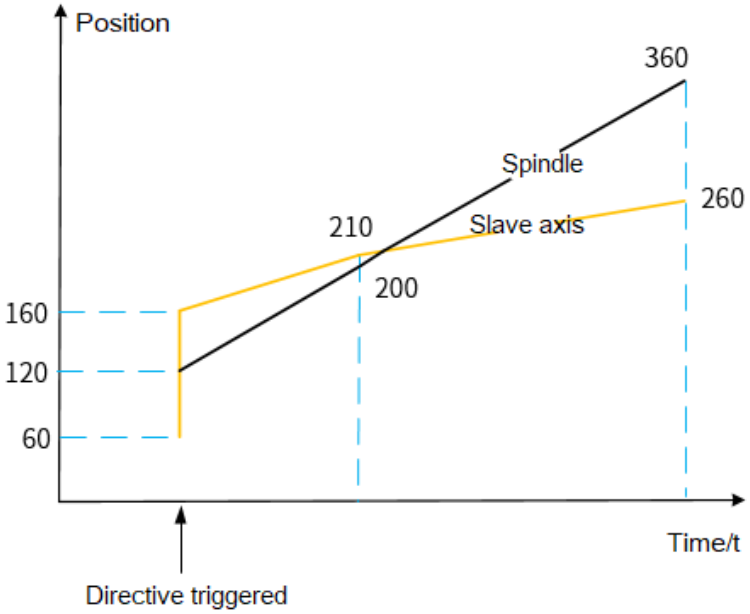
The following examples illustrate the relationship between the master and slave axis positions under each mode. Assume there is a cam table as follows: **(The master axis start position is 120, and the slave axis start position is 60; MasterStartLength = 0, IsRampIn = FALSE)**

Master axis position	0	120	200	360
Slave axis position	0	100	150	200

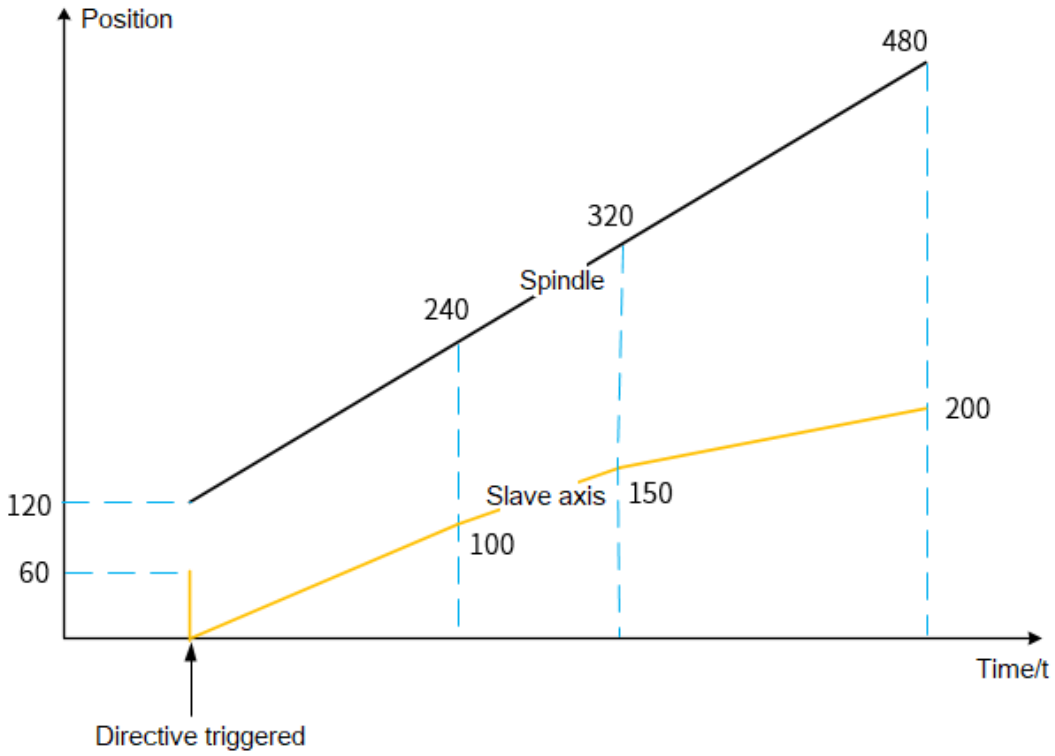
(1) Mode 1–Master axis in absolute mode, slave axis in absolute mode



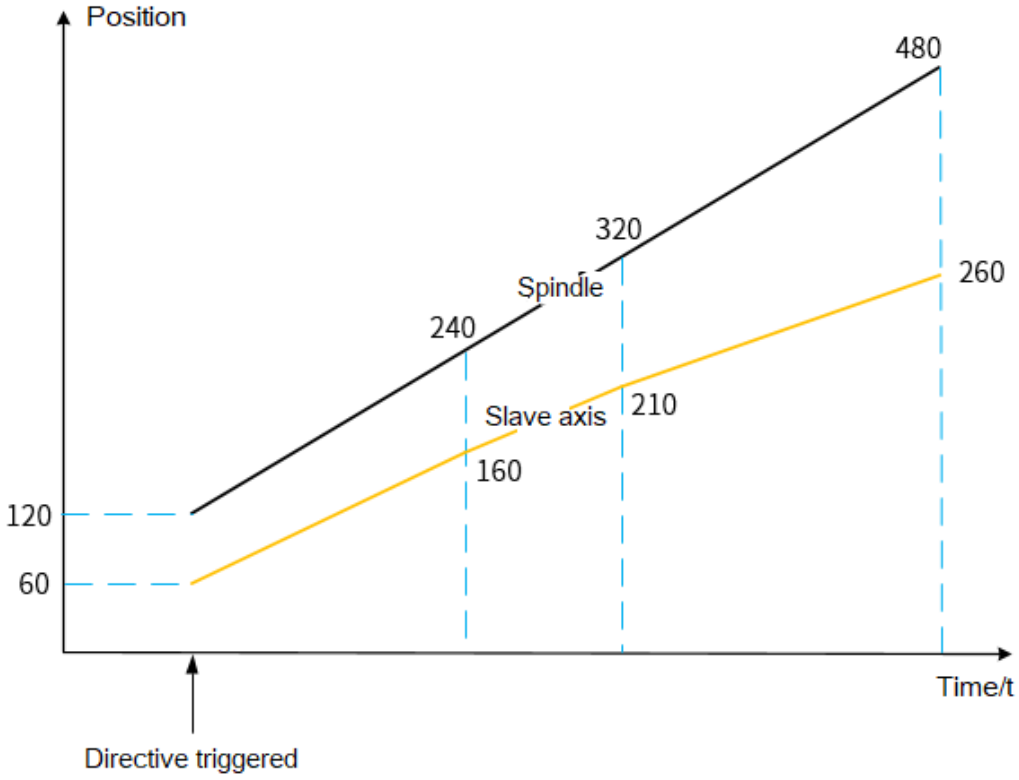
(2) Mode 2–Master axis in absolute mode, slave axis in relative mode



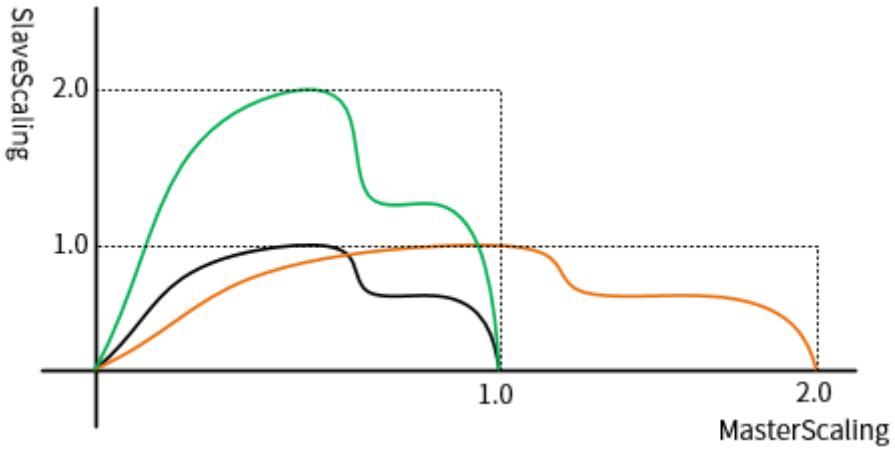
(3) Mode 3–Master axis in relative mode, slave axis in absolute mode



(4) Mode 4–Master axis in relative mode, slave axis in relative mode



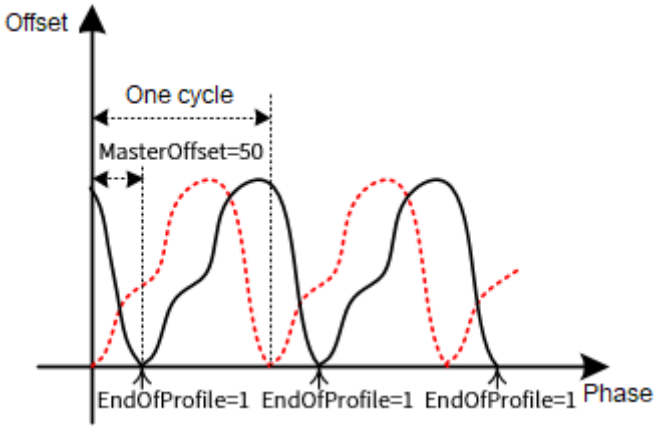
◦ **MasterScaling and SlaveScaling**



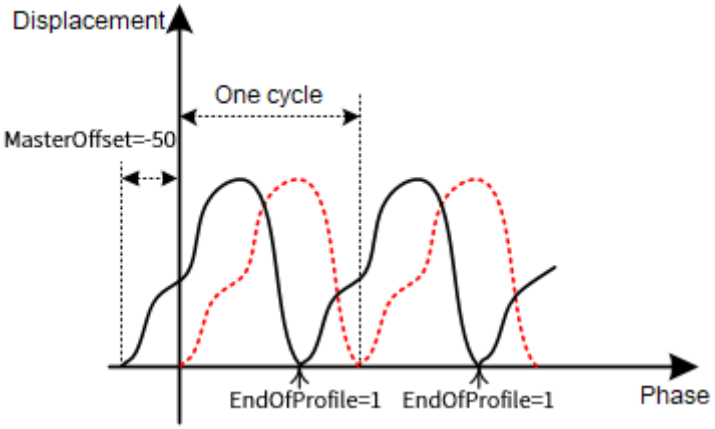
- For the master axis, when the MasterScaling is set to > 1, its phase range is extended; when the MasterScaling is set to < 1, its phase range is contracted.
- For the slave axis, when the SlaveScaling is set to > 1, its displacement is extended; when the SlaveScaling is set to < 1, its displacement is contracted.
- When the input value of MasterScaling is less than 0, it will be automatically corrected to 1 by the function block;
- When the input value of SlaveScaling is less than 0, it will be automatically corrected to 1 by the function block;
- The output values of phase and displacement in C_CAM_IN_STATE have been scaled according to corresponding scaling factors for the master and slave axes.

◦ **MasterOffset and SlaveOffset**

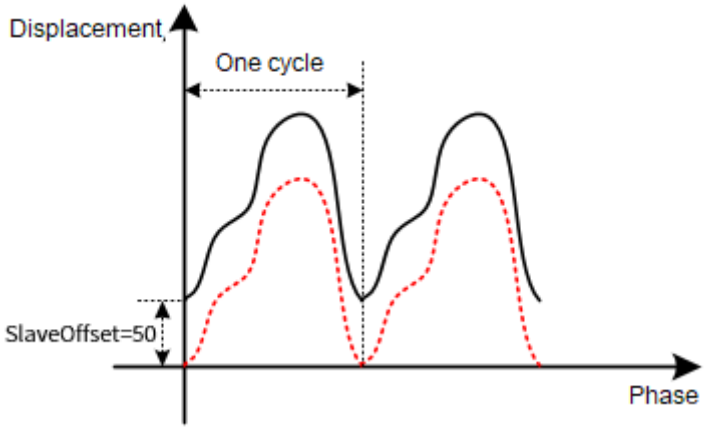
MasterOffset > 0



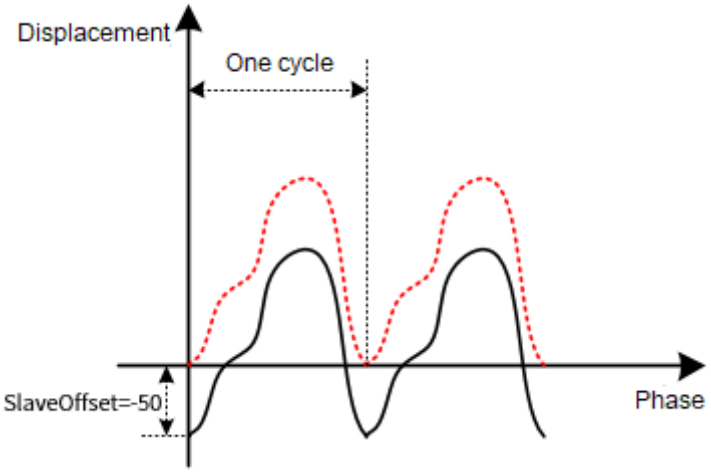
MasterOffset < 0



SlaveOffset > 0



SlaveOffset < 0



- When the MasterOffset is set to > 0, the cam curve shifts to the right; when the MasterOffset is set to < 0, the cam curve shifts to the left.
- When the SlaveOffset is set to > 0, the cam curve shifts upward; when the SlaveOffset is set to < 0, the cam curve shifts downward.
- The output value of displacement in C_CAM_IN_STATE includes the SlaveOffset.

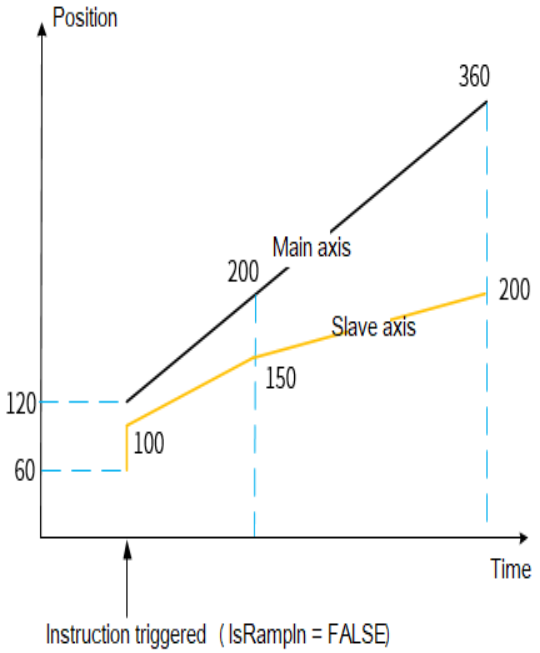
◦ **IsRampIn**

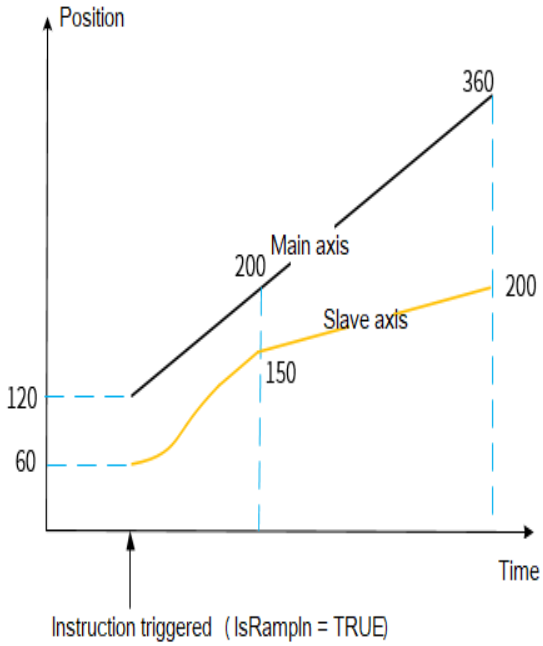
Enabling this function ensures that a smooth velocity transition during camming of the slave axis, but acceleration may be discontinuous. When you select to start camming in a non-periodic mode, a fault

will occur if the coupling compensation has not been fully executed after all camming cycles have been executed.

The following example illustrates the difference in effects when coupling is enabled and disabled. Assume there is a cam table as follows: **(The master axis start position is 120, and the slave axis start position is 60; MasterStartLength = 0)**

Master axis position	0	120	200	360
Slave axis position	0	100	150	200





◦ **RampDirection**

For linear axes, this parameter has no effect on the coupling curve regardless of its value; for rotary axes, different coupling directions result in different coupling curves.

Assume there is a cam table as follows: **(Both master and slave axes in absolute mode)**

Master axis position	0	120	200	360
Slave axis position	0	120	300	0

When the IsRampIn is enabled with the master axis start position of 120 and the slave axis start position of 100:

- For the slave axis specified as **a rotary axis**:

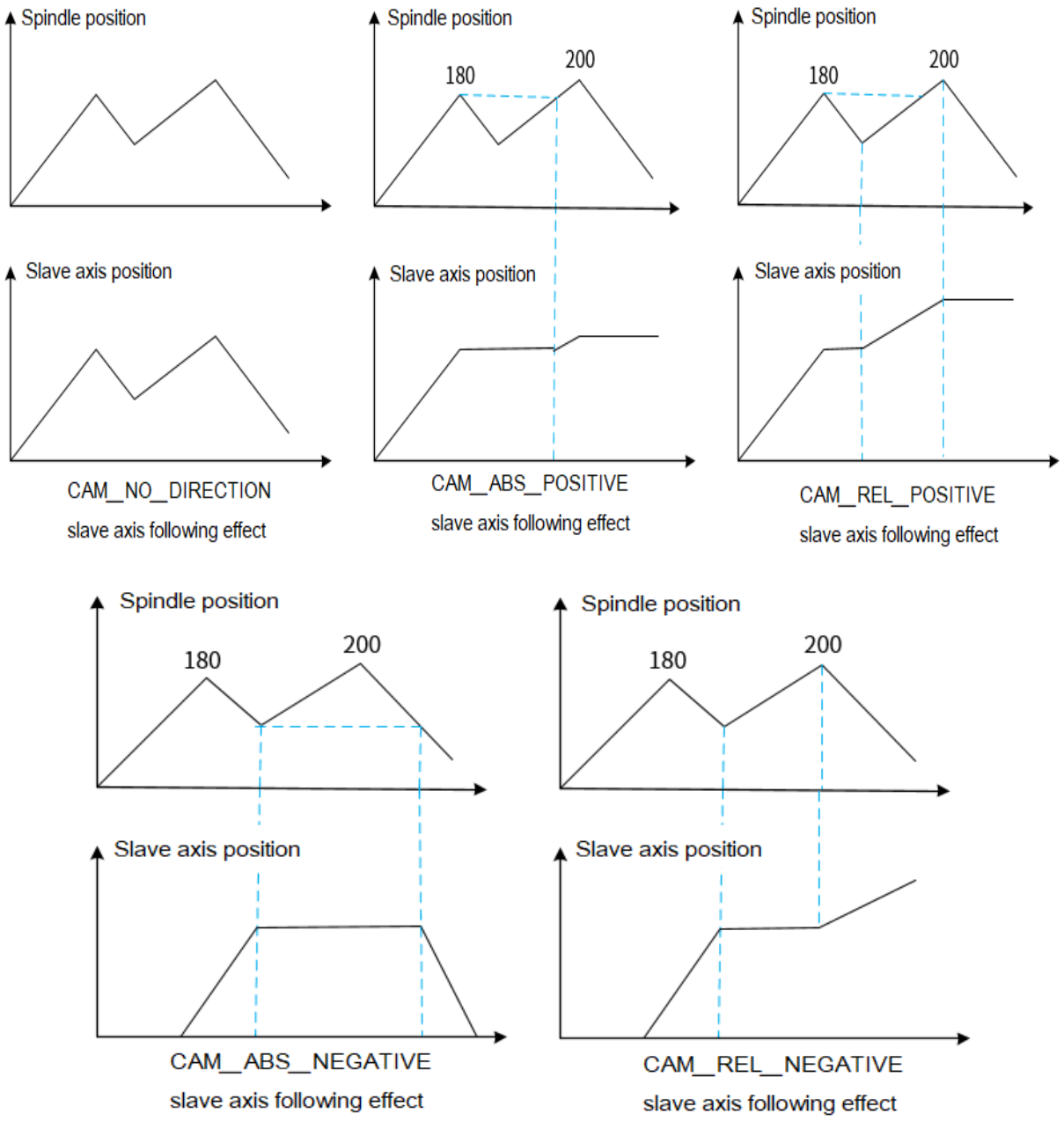
  DIR_SHORTEST: Coupling compensation = 20.

  DIR_POSITIVE: Coupling compensation = 20.

  DIR_NEGATIVE: Coupling compensation = -340 (20 - 360).

- For the slave axis specified **as a linear axis**: The effects are the same for the three coupling directions, with a compensation of 20.

◦ **MasterDirection**



- When the CAM_REL_POSITIVE or CAM_REL_NEGATIVE is selected, the slave axis stops following the master axis when the master axis reverses to the opposite direction, and immediately resumes following when the master axis reverses back to the specified direction.
- When the CAM_ABS_POSITIVE or CAM_ABS_NEGATIVE is selected, the slave axis stops following the master axis when the master axis reverses to the opposite direction, and immediately resumes following when the master axis reverses back to the specified direction and returns to the reversal point. In case of a rotary axis, the number of cycles during reversal will be recorded, and the slave axis will resume following only when the master axis returns to its original cycle count.

◦ **MasterStartLength**

- A positive or negative input value for MasterStartLength does not affect the normal operation of the function block; the input value is processed by the function block based on the motion direction of the master axis. For example, when the MasterDirection is set to positive direction, the MasterStartLength is internally processed as a positive value. When the MasterDirection is set to NoDirection, the sign of MasterStartLength is determined by the motion direction of the master axis when its velocity first becomes non-zero after the instruction is triggered.

- After the instruction is triggered, if the motion direction of the master axis matches the specified MasterDirection, the master axis begins to record the following distance. When this distance reaches the setpoint for MasterStartLength, InCam is set, and the slave axis begins to follow the master axis.

◦ **Velocity overlimit and hitting the position limit**

- When the slave axis follows the master axis, velocity monitoring is executed to see if the velocity exceeds the software limit; if yes, a fault will occur.
- When the master axis position is beyond the software limit, the master axis velocity decelerates to 0 within one cycle, without affecting the state of the slave axis. When the master axis resets and resumes movement, the slave axis continues to follow the master axis.
- When the slave axis position is beyond the software limit, the instruction is aborted, and the slave axis decelerates to 0 within one cycle with the state switching to ErrorStop.

◦ **InCam and InSync**

- When the MasterStartLength is set to 0, InCam is set to TRUE in the same cycle after the instruction is triggered. When the MasterStartLength is set to non-zero, the InCam is set to TRUE when the master axis moves in the specified direction and its cumulative movement reaches MasterStartLength after the instruction is triggered.
- When the IsRampIn is not set, the InCam and InSync are set to TRUE in the same cycle.
- When the IsRampIn is set (IsRampIn = TRUE), the InCam is set to TRUE first; the InSync is set to TRUE after the coupling ends.

◦ **EndOfProfile**

- The output variable EndOfProfile is set to TRUE when the cam table end point is used as the position reference, and changes to FALSE in the next cycle. After the last camming cycle is completed, the EndOfProfile remains TRUE.

◦ **BufferMode**

- The CamInV2 instruction supports BufferMode, allowing the instruction to be triggered in buffered mode. When Periodic is set to 0 (cyclic cam motion) and a buffered instruction follows, the buffered instruction takes effect after the current camming cycle completes.

◦ **Cam table parameter update methods**

- You can select one of the two cam table parameter update methods through the parameter UpDateRightNow. When the instruction is re-executed with changes to the cam table parameters, the modified cam table data will take effect immediately, rather than following the UpDateRightNow mode. If you want the modified cam table data to take effect without re-executing the instruction, the instruction will respond to the new cam data based on the value of the parameter UpDateRightNow. For the specific way the parameter UpDateRightNow takes effect, see UpDateRightNow in the instance description.

◦ **Table switching**

- During the table switching, internal smoothing is applied within the instruction to prevent velocity overlimit due to sudden change. Additionally, if there is any residual motion due to coupling, it will be superimposed to the new table for execution.

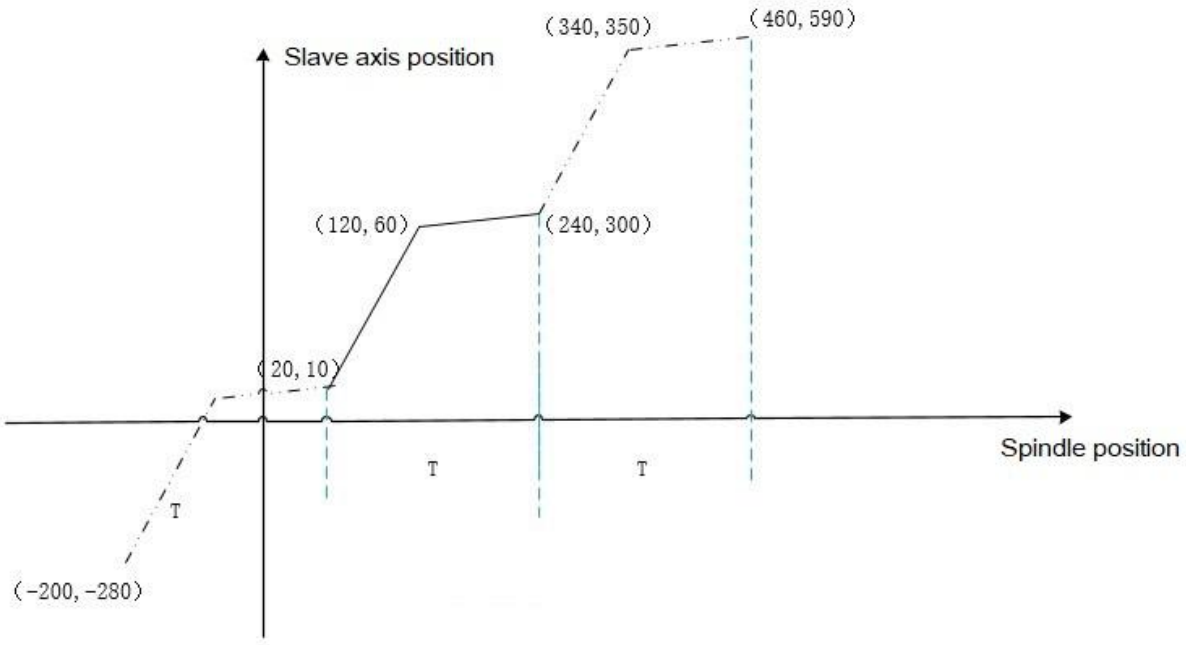
◦ **Support for setting cam table start point phase and displacement to non-zero**

Assume there is a cam table as follows: **(Key nodes are connected by straight lines)**

Master axis position	20	120	240

Slave axis position	10	60	300

When cam table following starts, if the master axis position is beyond the end point of the cam table, the slave axis position is obtained by forward extension of the cam curve; if the master axis position is before the start point of the cam table, the slave axis position is obtained by backward extension of the cam curve. The extended cam curve is as follows (the figure shows the extension of one camming cycle in positive and negative directions).



◦ **Re-execution**

▪ During the execution of the instruction (Active = TRUE), you can modify parameters online and then re-execute the instruction to make the modified parameters take effect. In this case, the corresponding number of cycles will be reset to 0, and the MasterStartLength will also take effect again. If you want the slave axis to immediately follow the master axis upon re-execution, set the MasterStartLength to 0. You can also re-execute the buffered instruction (Active = FALSE) after modifying parameters.

- **Example**

This instruction is a synchronous control instruction. Camming motion is based on a master axis and one or more slave axes. The master axis can be a physical axis or a virtual axis.

The following shows the steps for using synchronization instructions:

① **Specify the master and slave axes:** Specify the master and slave axes for synchronous motion based on the specific application scenario. The master axis for a synchronous motion control instruction can be a slave axis of another synchronous motion control instruction. The slave axis of one synchronous motion control instruction can also serve as the master axis of another synchronous motion control instruction.

② **Create a cam table:** Based on the specific application scenario, determine the position relationship between the master and slave axes, and create a cam table. In multiple different instantiated MC_CamInV2 instructions, when Axis is specified as the master axis and Axis_1 is specified as the slave axis, the cam table for each instruction can be different.

③ **Call the cam function block:** Use the MC_CamInV2 instruction to bind the master axis, slave axis, and cam table.

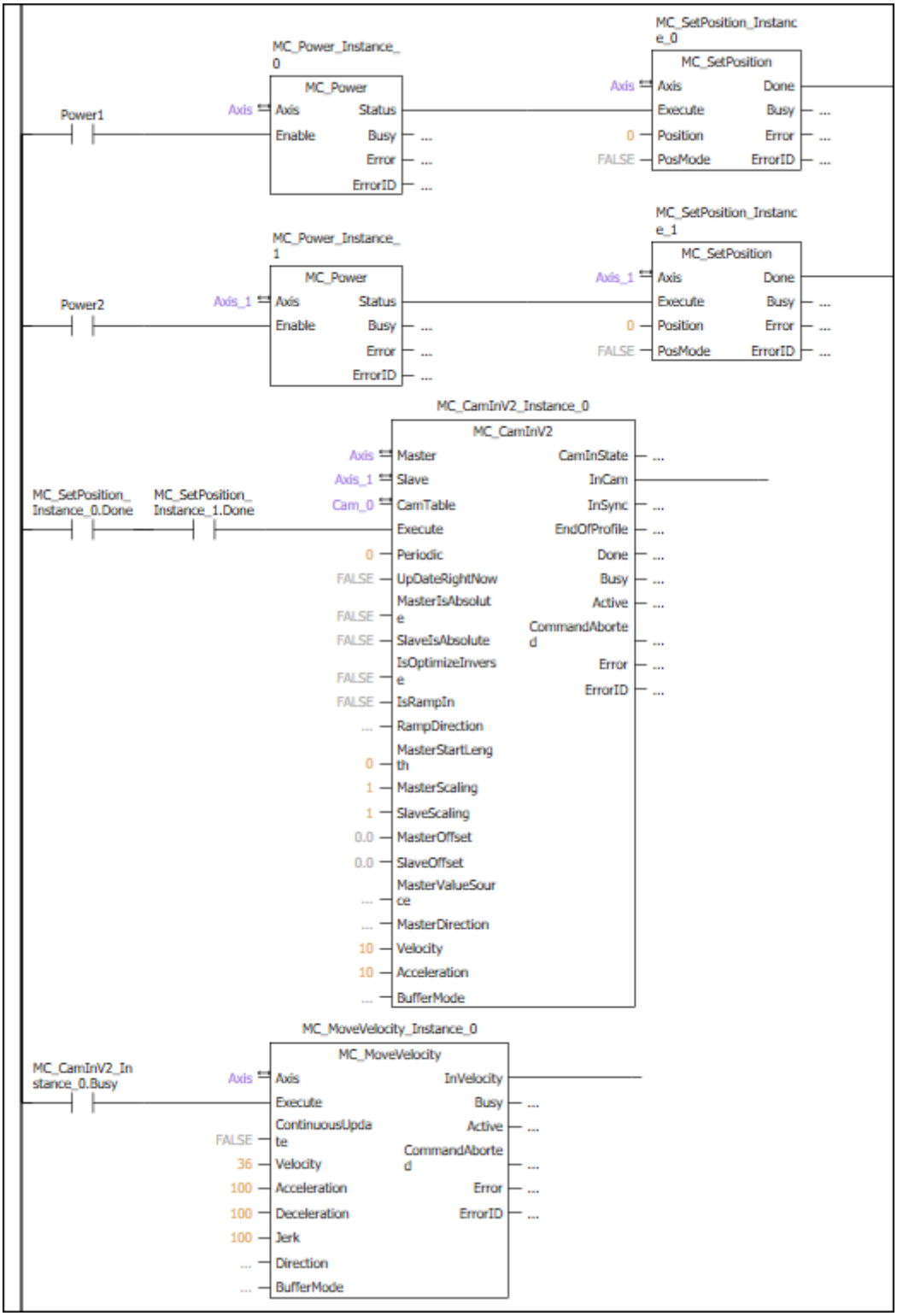
④ **Axis enable and current axis position setting function block:** Use the MC_Power instruction to switch the axis to the enabled state, and then use the MC_SetPosition instruction to set the current position of the axis.

⑤ **Specify the input parameters of the function block:** Based on the specific application scenario, specify the input parameters of the MC_CamInV2 instruction and execute this instruction.

⑥ **Master axis motion function block:** Use the displacement/velocity control instruction to move the master axis. In the example, the MC_MoveVelocity instruction is used to control the velocity of the axis.

⑦ **Desynchronize the master and slave axes:** Wait for the synchronous control instruction to complete, or use the MC_CamOut instruction for desynchronization of the master and slave axes.

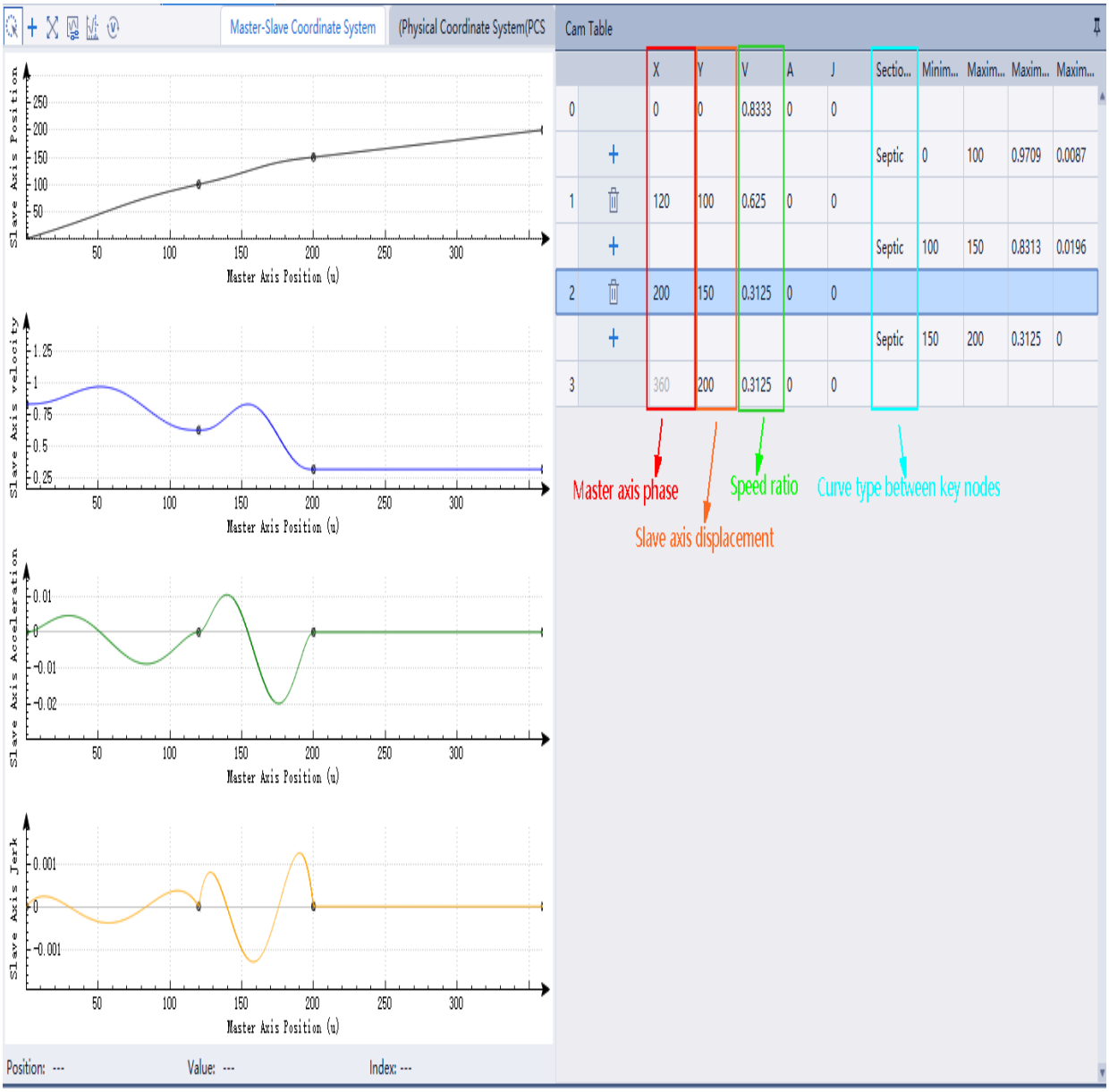
The following example illustrates how to execute simple camming. Configure two axes: Axis as the master axis and Axis_1 as the slave axis. Create a cam table Cam_0. First, execute the MC_Power instruction to switch the axis to the enabled state, and then specify the current axis position through the MC_SetPosition instruction, which is triggered by the Busy output. After the execution of the MC_SetPosition instruction, camming is started when the MC_CamInV2 instruction has been triggered by the Done output. After the execution of the MC_CamInV2 instruction, enable the Axis to start moving according to the input parameters of the MC_MoveVelocity instruction, which is triggered by the Busy output. At the same time, the slave axis starts the camming according to the input parameters of the MC_CamInV2 instruction and the created cam table.



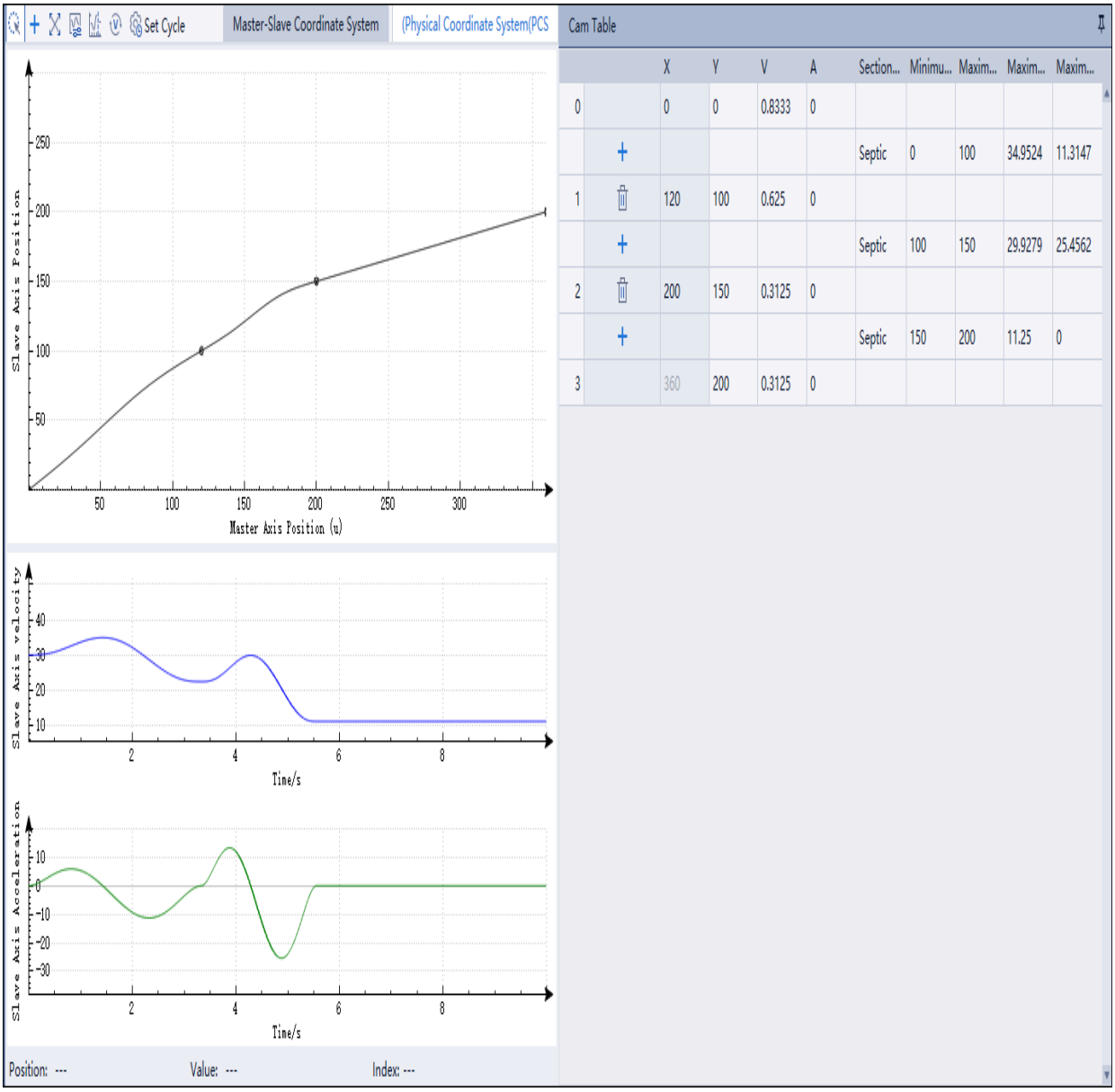
- **Create a cam table**

  The created cam table automatically generates the position relationship between the master and slave axes, and the velocity information of the slave axis at key nodes. You can adjust the data in the cam table to fit specific application scenarios.

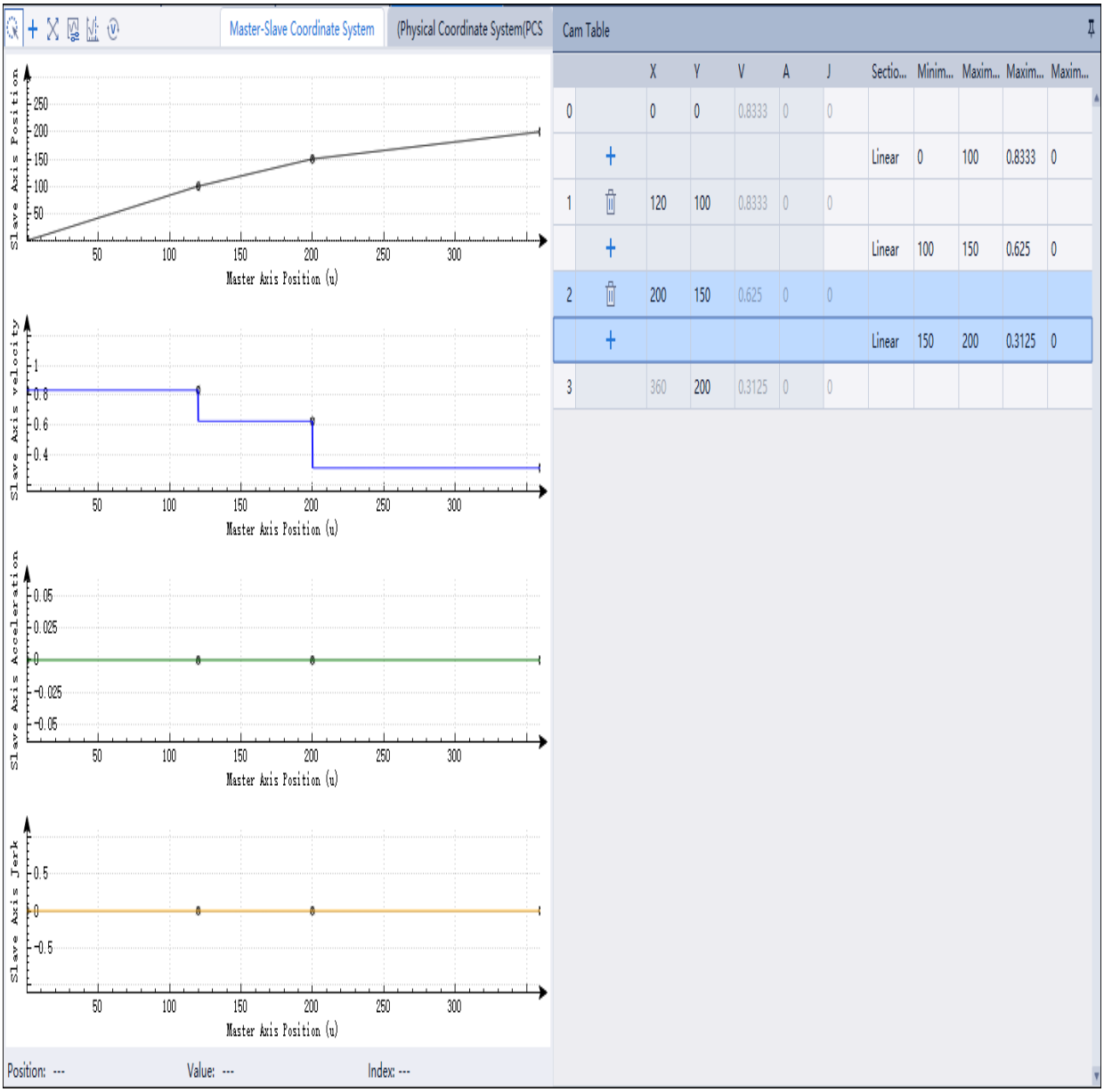
  As shown in the following figure, the master-slave coordinate systems show the relationships between the master axis position and the slave axis position, velocity, acceleration, and jerk.



As shown in the following figure, the physical coordinate systems show the relationships between time and the slave axis position, velocity, and acceleration.



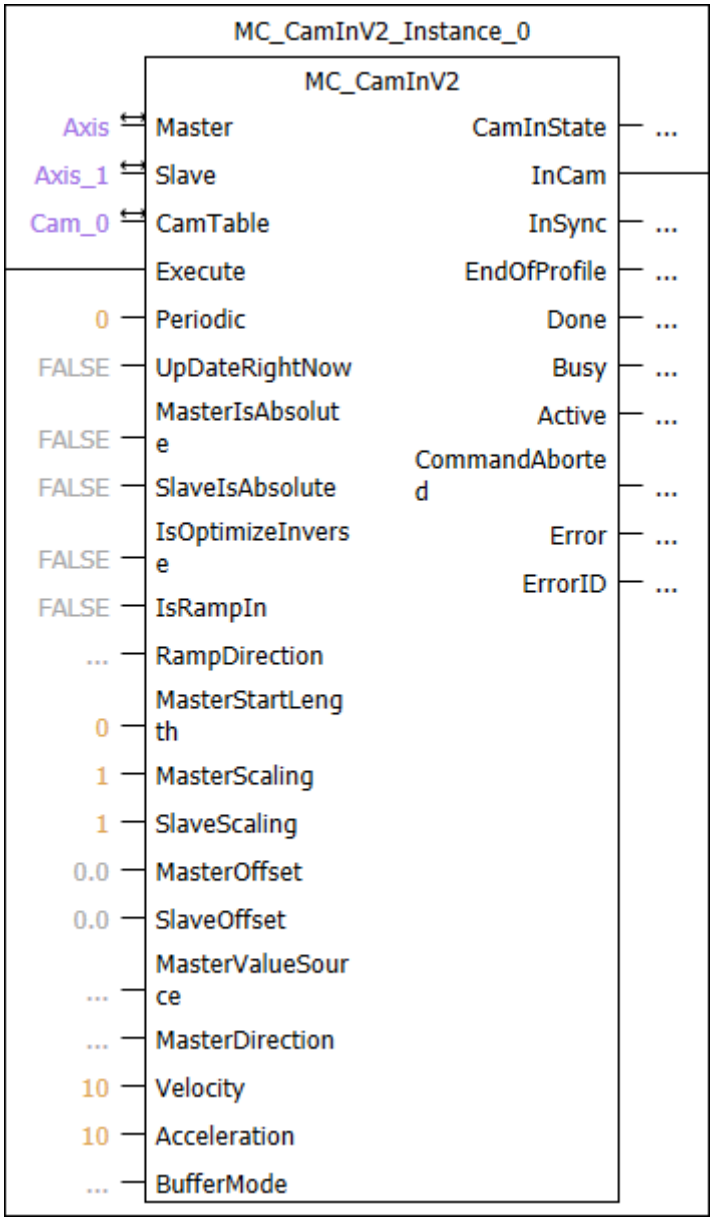
When the curve type between key nodes is changed, the velocity, acceleration, and jerk of the slave axis also change. As shown in the following figure, the curve type between key nodes is changed to "Straight line".



- **Call the cam function block**

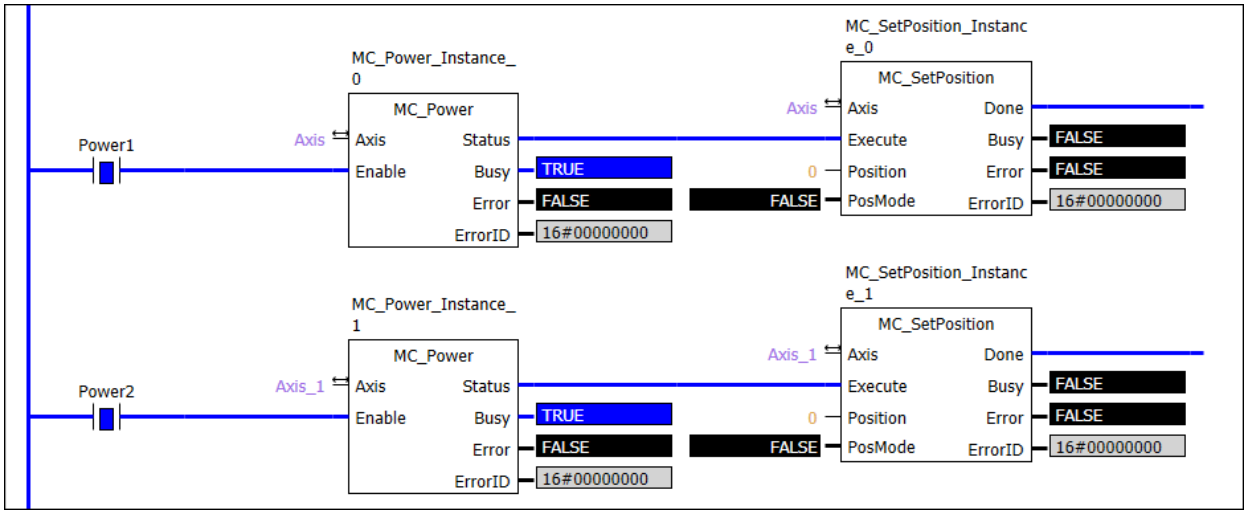
  Create a cam table, specify the position relationship between the master and slave axes, specify the input parameters of the MC_CamInV2 instruction according to specific application scenarios, and then execute this instruction.

  In the following program example, the MC_CamInV2 instruction is executed with the cam table created above and the input parameter UpDateRightNow defaulting to FALSE.



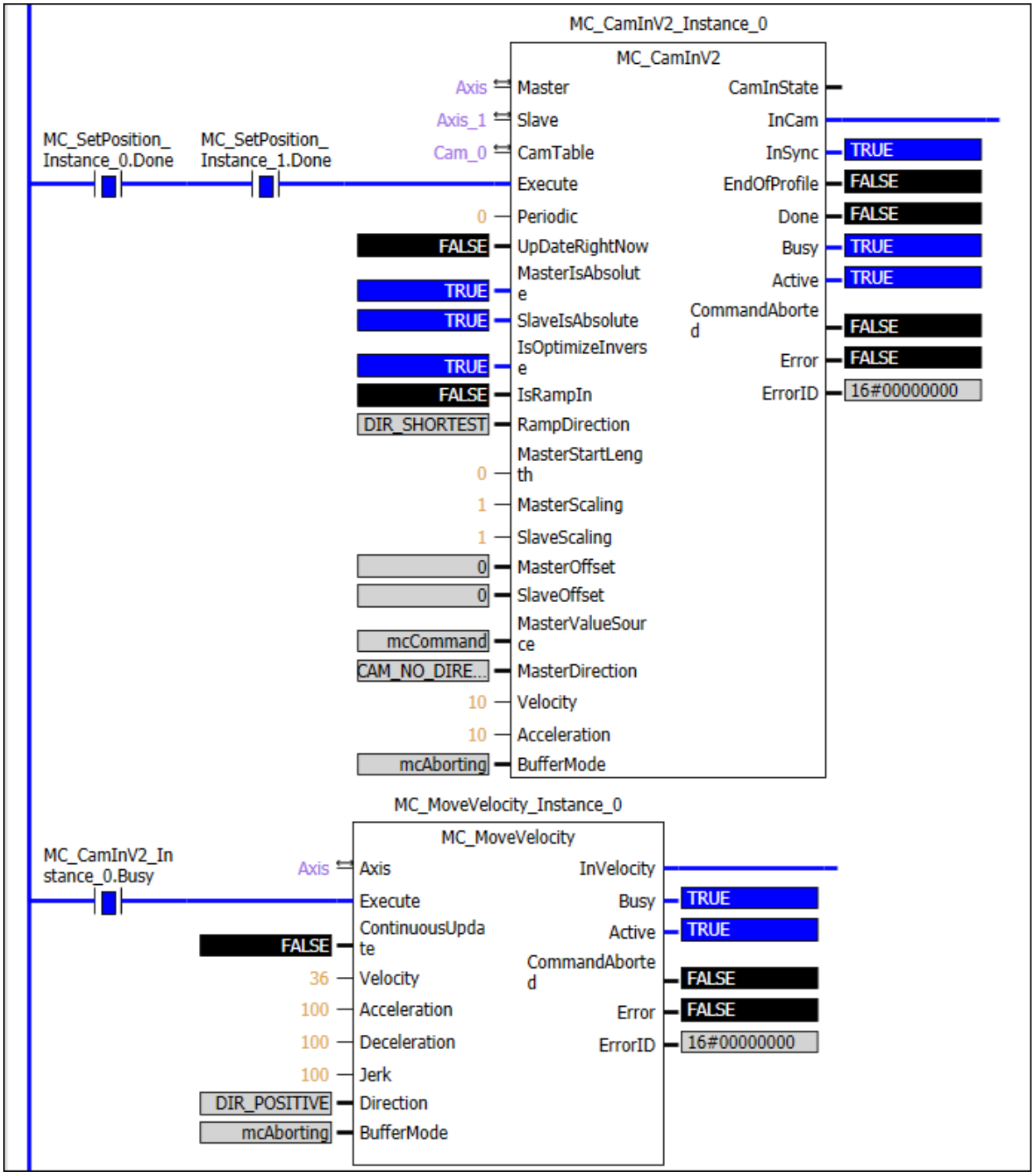
- **Axis enable and current axis position setting function block**

  Set contacts Power1 and Power2 to high level to trigger the MC_Power and MC_SetPosition instructions.



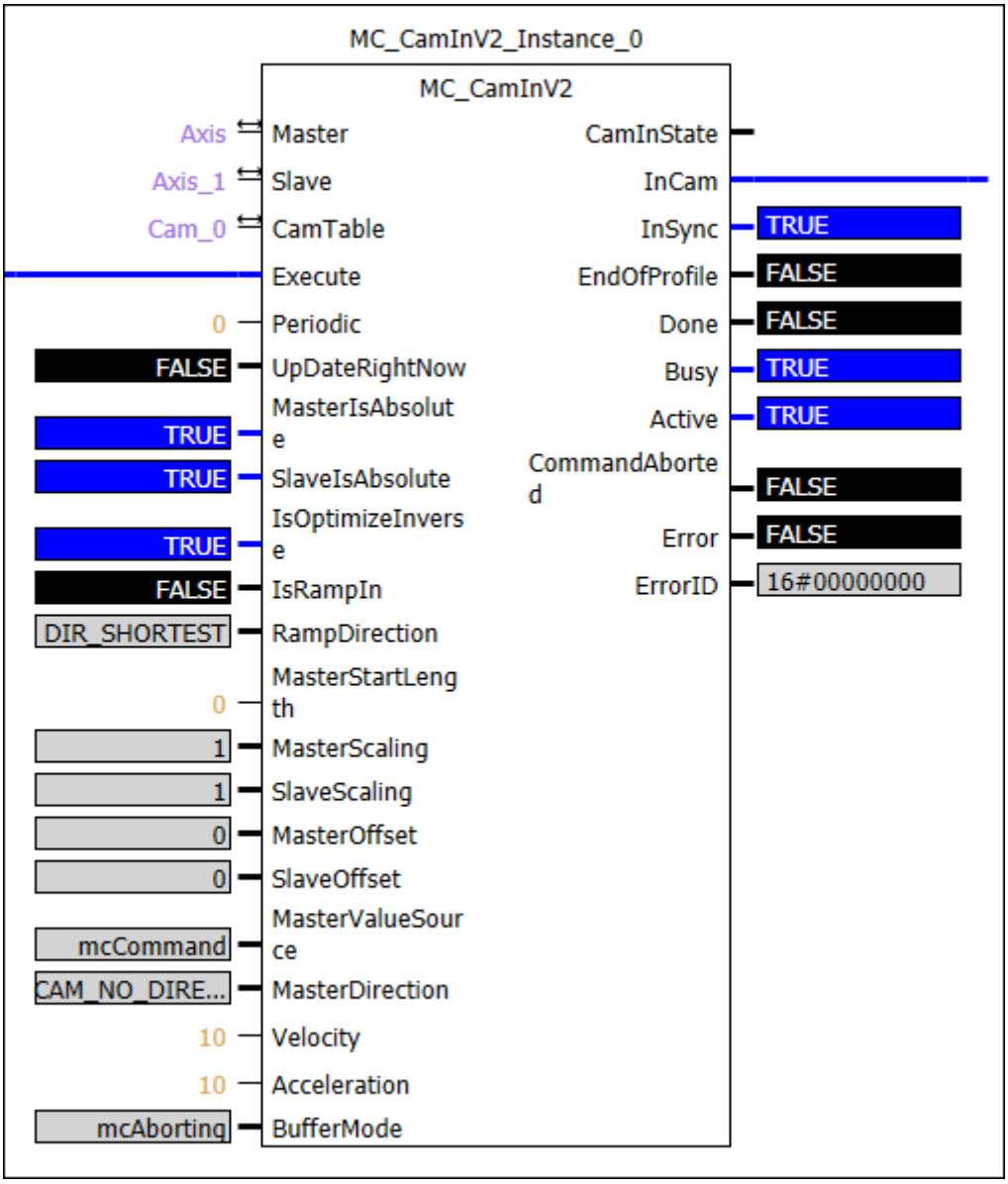
- **Execute the MC_CamInV2 instruction**

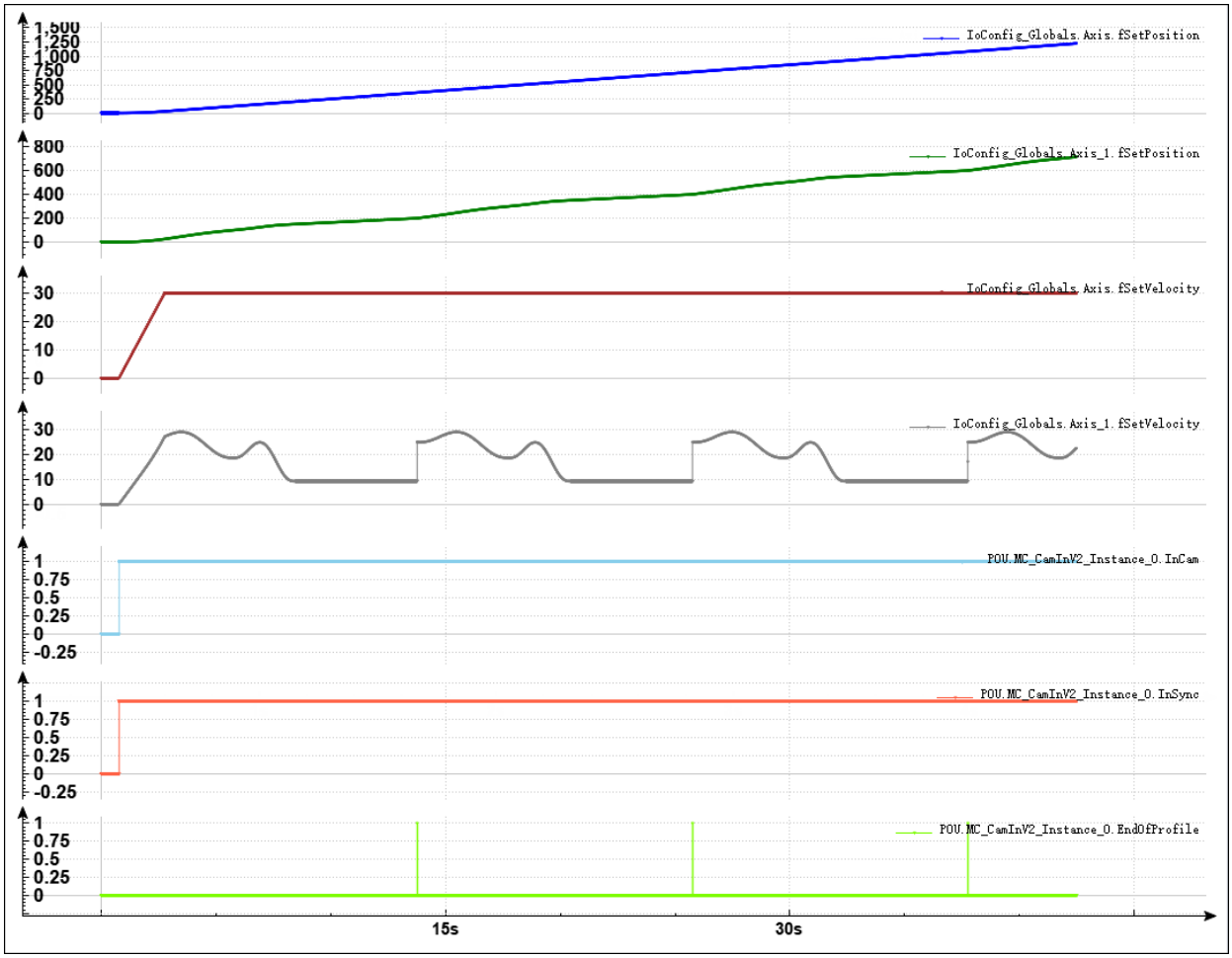
While the MC_CamInV2 instruction is being executed, execute the MC_MoveVelocity instruction to control the velocity of the Axis. Specify the velocity to 36 to make the Axis start moving. At this time, the slave axis starts the camming according to the created cam table.



- You can specify the number of times the cam table is executed through the parameter Periodic. When the Periodic is set to 0, it indicates that the cam table is executed periodically, and this instruction is executed repeatedly according to the cam table. When the Periodic is set to a specific value, this instruction is executed according to the specified number of cycles.

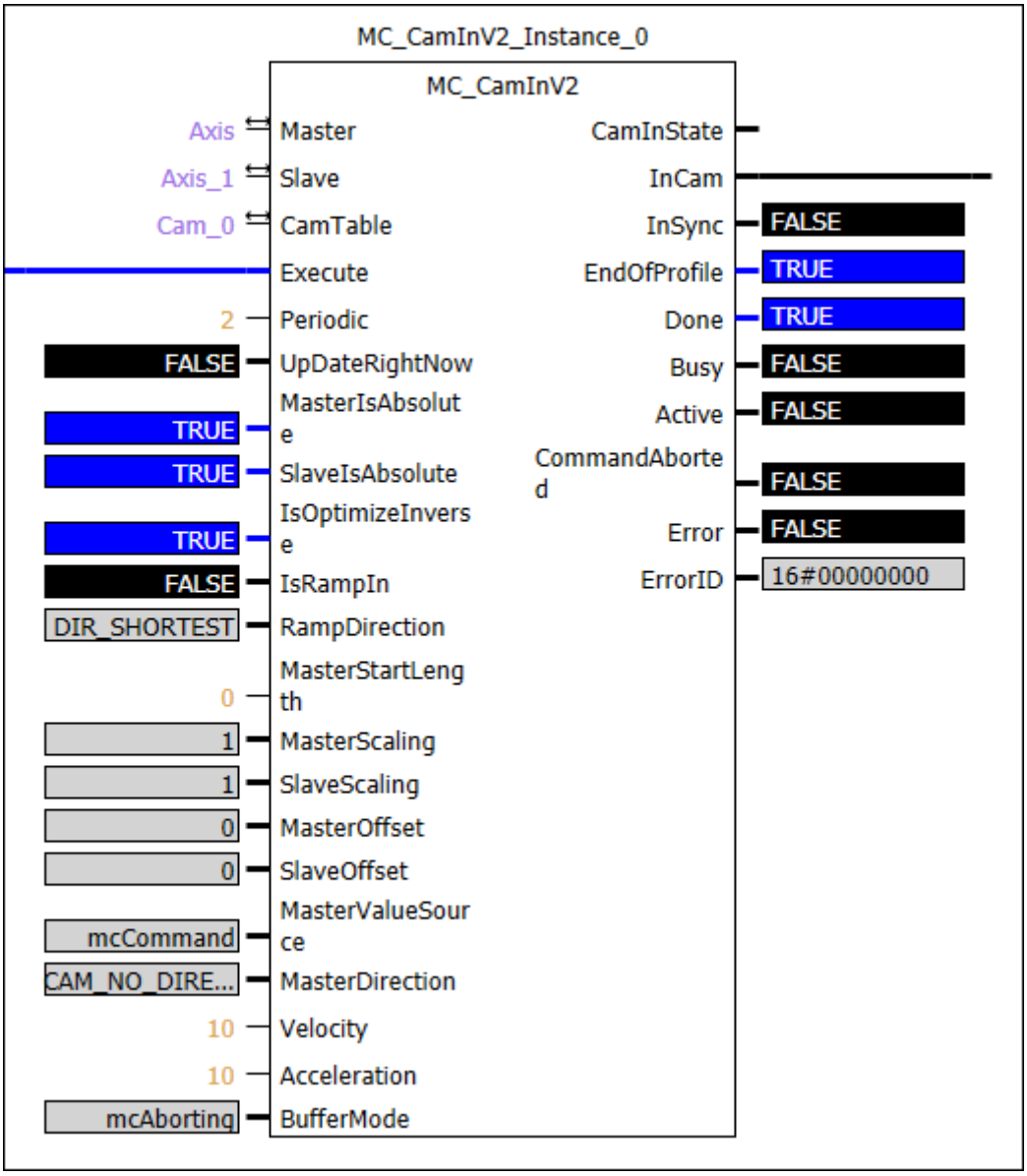
  Periodic = 0, master axis in absolute position mode, and slave axis in absolute position mode (The master axis start position is 0, and the slave axis start position is 0)

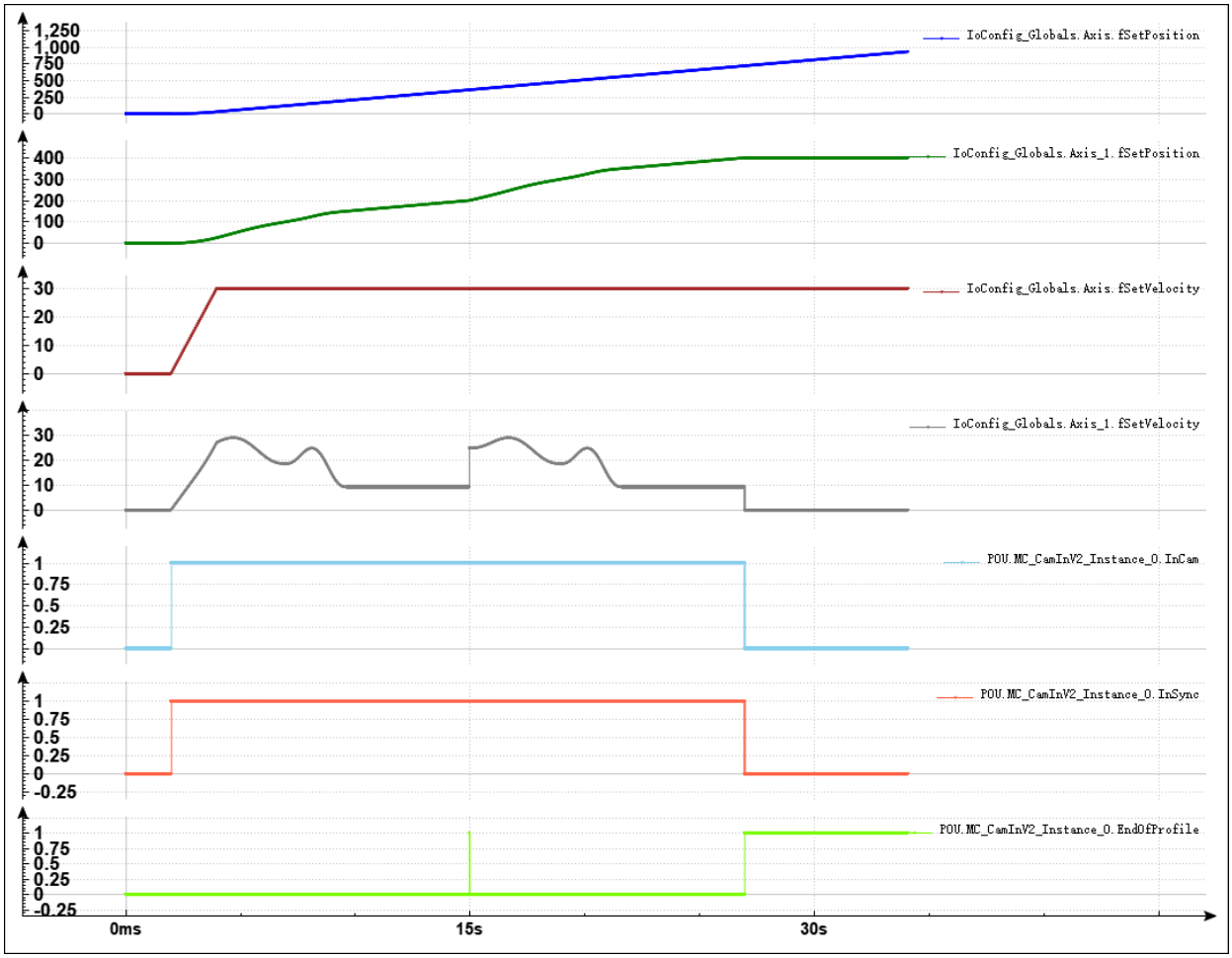

MC_CamInV2_Instance_0
MC_CamInV2
Axis
Master
Axis_1
Slave
Cam_0
CamTable
Execute
0
Periodic
FALSE
UpDateRightNow
TRUE
MasterIsAbsolute
TRUE
SlaveIsAbsolute
TRUE
IsOptimizeInverse
FALSE
IsRampIn
DIR_SHORTEST
RampDirection
0
MasterStartLength
1
MasterScaling
1
SlaveScaling
0
MasterOffset
0
SlaveOffset
mcCommand
MasterValueSource
CAM_NO_DIRE...
MasterDirection
10
Velocity
10
Acceleration
mcAborting
BufferMode
CamInState
InCam
InSync
TRUE
EndOfProfile
FALSE
Done
FALSE
Busy
TRUE
Active
TRUE
CommandAborted
FALSE
Error
FALSE
ErrorID
16#00000000



The EndOfProfile is set once for each execution of the cam table, and then reset in the next cycle.

Periodic = 2, master axis in absolute position mode, and slave axis in absolute position mode (The master axis start position is 0, and the slave axis start position is 0)


MC_CamInV2_Instance_0
MC_CamInV2
Axis
Master
Axis_1
Slave
Cam_0
CamTable
Execute
2
Periodic
FALSE
UpDateRightNow
TRUE
MasterIsAbsolute
TRUE
SlaveIsAbsolute
TRUE
IsOptimizeInverse
FALSE
IsRampIn
DIR_SHORTEST
RampDirection
0
MasterStartLength
1
MasterScaling
1
SlaveScaling
0
MasterOffset
0
SlaveOffset
mcCommand
MasterValueSource
CAM_NO_DIRE...
MasterDirection
10
Velocity
10
Acceleration
mcAborting
BufferMode
CamInState
InCam
InSync
FALSE
EndOfProfile
TRUE
Done
TRUE
Busy
FALSE
Active
FALSE
CommandAborted
FALSE
Error
FALSE
ErrorID
16#00000000



The EndOfProfile is set once for each execution of the cam table, and then reset in the next cycle. After the last cycle ends, the EndOfProfile is kept set, and the Done flag is also set.

- UpDateRightNow: indicates whether the modified cam table takes effect immediately.

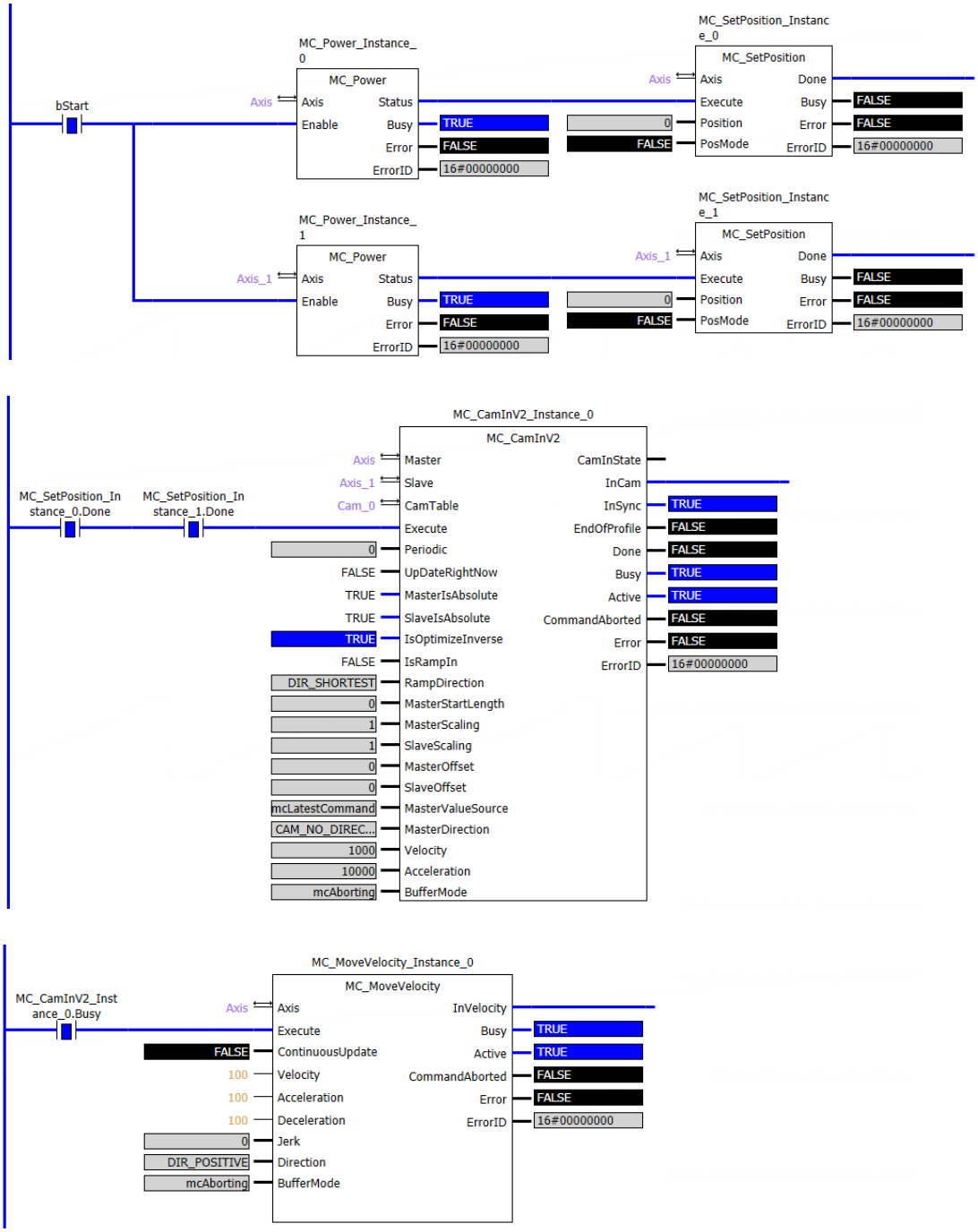
  When the UpDateRightNow is set to FALSE, the modified cam table data takes effect in the next camming cycle.

  - Cam table generated by iFA

  a. Use iFA to generate the cam table Cam_0.

		X	Y	V	A	J	Fragm...	Min. {p...	Max. {...	Max. {v...	Max. {...
0		0	0	0	0	0					
	+						Septic...	0	120	1.6603	0.0419
1	🗑	120	120	1	0	0					
	+						Septic...	120	240	1	0
2	🗑	240	240	1	0	0					
	+						Septic...	240	360	1.6603	0.0419
3		360	360	0	0	0					

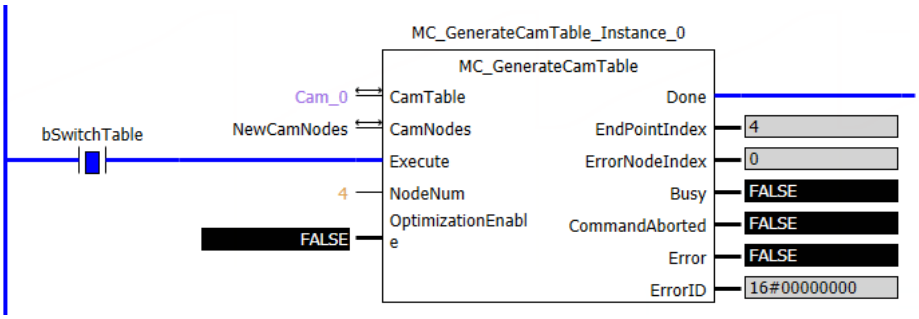
b. Execute the MC_CamInV2 instruction.



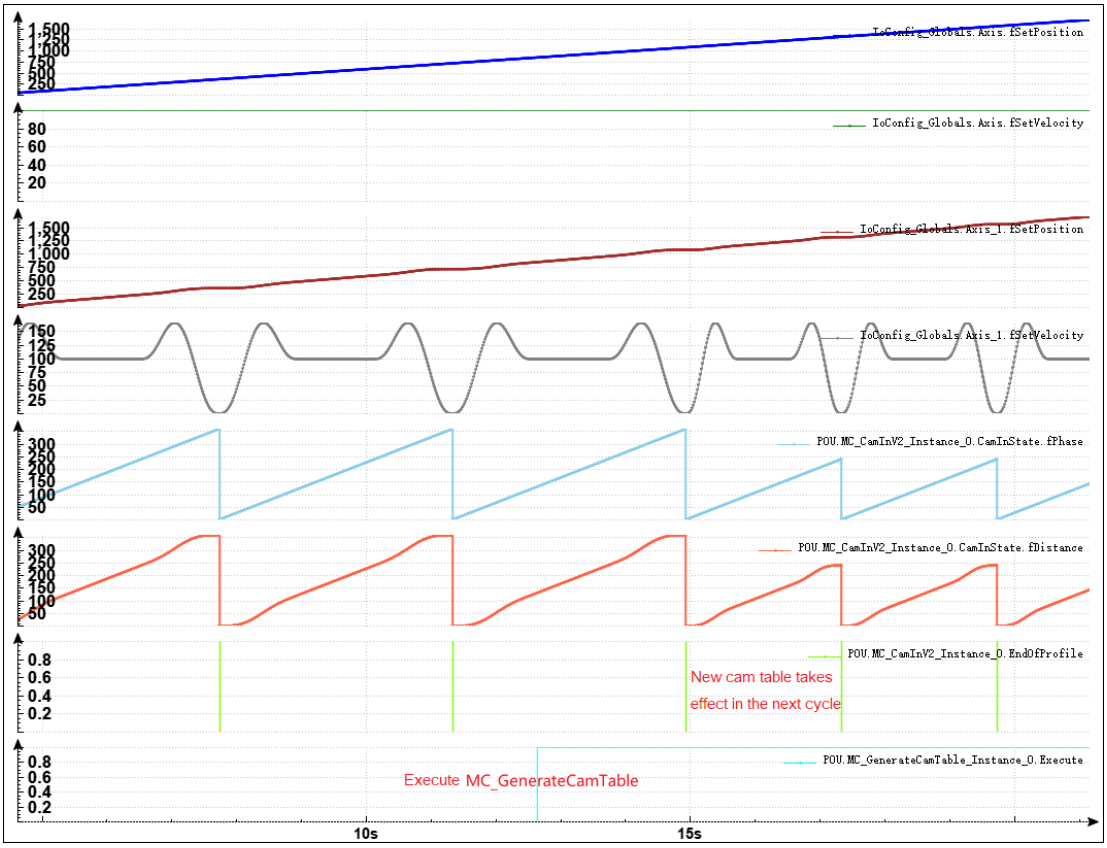
c. During the execution, enter new node data in the field corresponding to NewCamNodes to update the cam node data.

NewCamNodes	ARRAY[0..3] OF MC_CA...	
NewCamNodes[0]	MC_CAM_KEY_NODE	
fPhase	LREAL	0
fDistance	LREAL	0
fVel	LREAL	0
fAcc	LREAL	0
fJerk	LREAL	0
iCurveType	INT	5
NewCamNodes[1]	MC_CAM_KEY_NODE	
fPhase	LREAL	80
fDistance	LREAL	80
fVel	LREAL	1
fAcc	LREAL	0
fJerk	LREAL	0
iCurveType	INT	5
NewCamNodes[2]	MC_CAM_KEY_NODE	
fPhase	LREAL	160
fDistance	LREAL	160
fVel	LREAL	1
fAcc	LREAL	0
fJerk	LREAL	0
iCurveType	INT	5
NewCamNodes[3]	MC_CAM_KEY_NODE	
fPhase	LREAL	240
fDistance	LREAL	240
fVel	LREAL	0
fAcc	LREAL	0
fJerk	LREAL	0
iCurveType	INT	5

d. Enter the number of nodes to be replaced in NodeNum of the MC_GenerateCamTable.



e. The operation result is shown in the figure below.



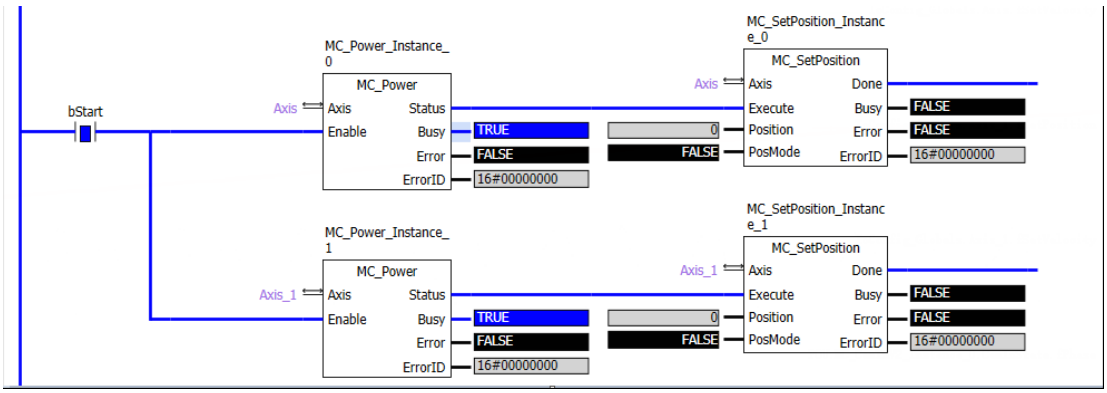
- User-defined cam table

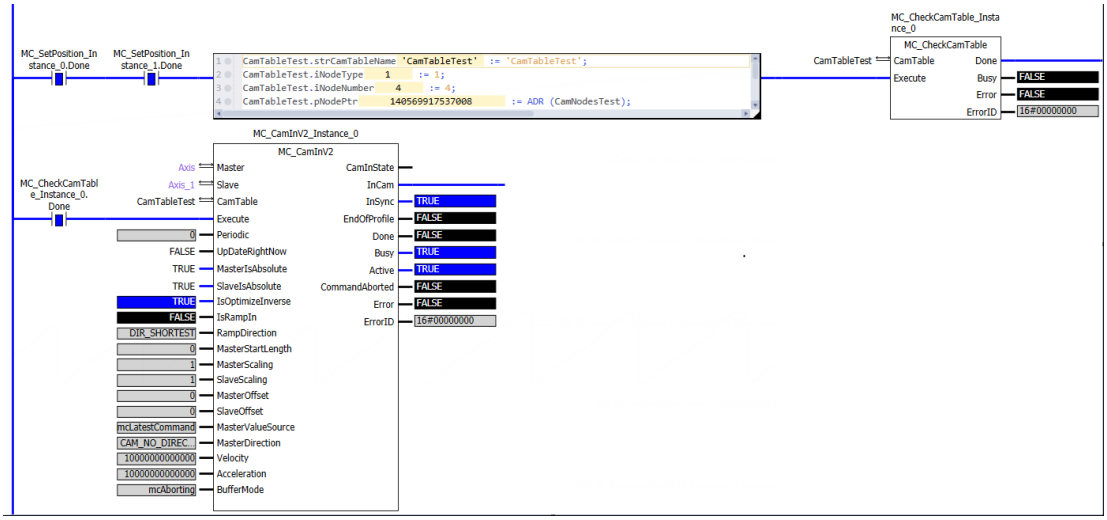
a. Specify a CamTableTest and use CamNodesTest as its key nodes.

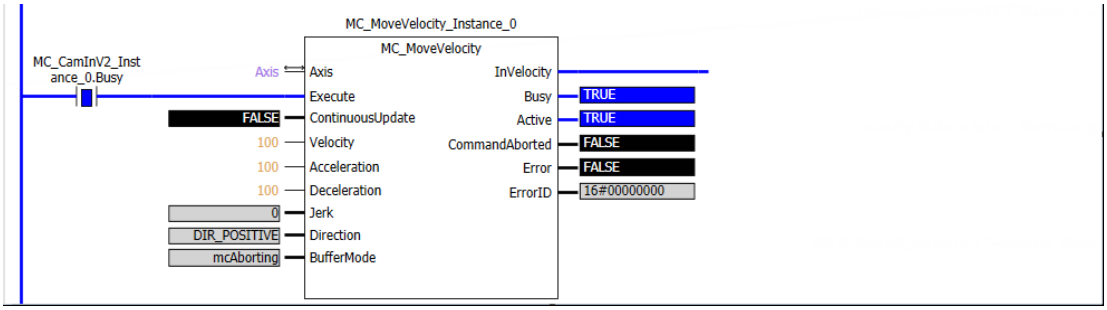
```
CamTableTest.strCamTableName 'CamTableTest'  := 'CamTableTest';
CamTableTest.iNodeType    1     := 1;
CamTableTest.iNodeNumber    4     := 4;
CamTableTest.pNodePtr      140569917537008      := ADR (CamNodesTest);
```

CamNodesTest	ARRAY[0..3] OF MC_CA...	
CamNodesTest[0]	MC_CAM_KEY_NODE	
fPhase	LREAL	0
fDistance	LREAL	0
fVel	LREAL	0
fAcc	LREAL	0
fJerk	LREAL	0
iCurveType	INT	5
CamNodesTest[1]	MC_CAM_KEY_NODE	
fPhase	LREAL	80
fDistance	LREAL	80
fVel	LREAL	1
fAcc	LREAL	0
fJerk	LREAL	0
iCurveType	INT	5
CamNodesTest[2]	MC_CAM_KEY_NODE	
fPhase	LREAL	160
fDistance	LREAL	160
fVel	LREAL	1
fAcc	LREAL	0
fJerk	LREAL	0
iCurveType	INT	5
CamNodesTest[3]	MC_CAM_KEY_NODE	
fPhase	LREAL	240
fDistance	LREAL	240
fVel	LREAL	0
fAcc	LREAL	0
fJerk	LREAL	0
iCurveType	INT	5

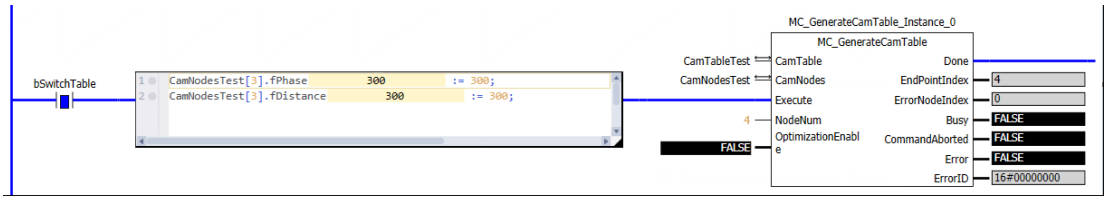
b. Execute the MC_CamInV2 instruction.



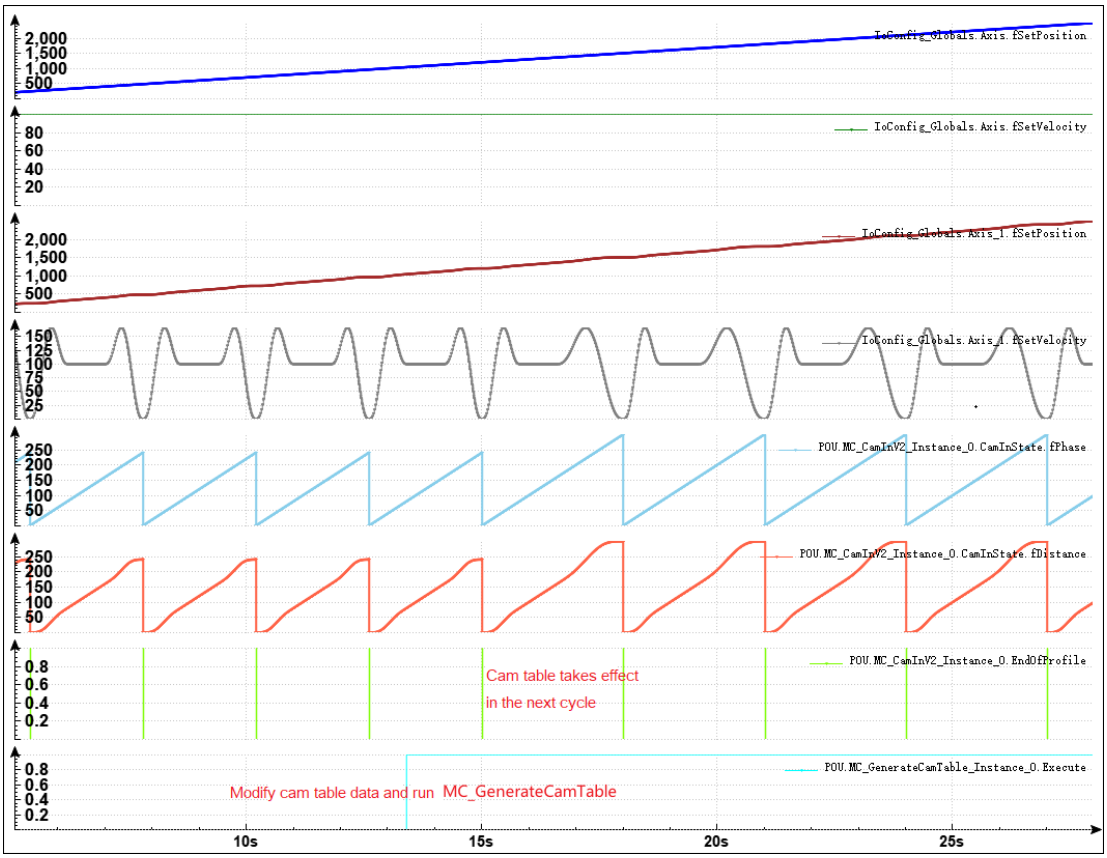




c. During the execution, enter new node data in NewCamNodes to update the cam node data, and then enter the number of nodes to be replaced in NodeNum of the MC_GenerateCamTable.



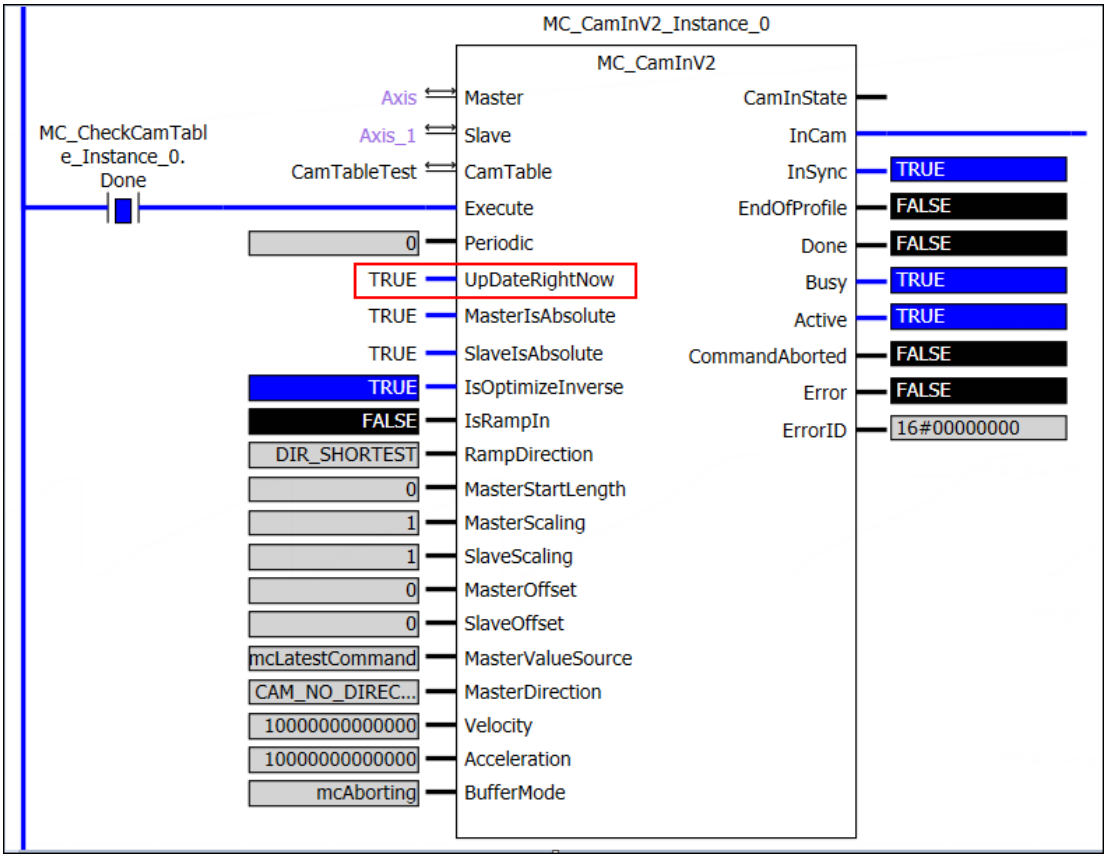
d. The operation result is shown in the figure below.



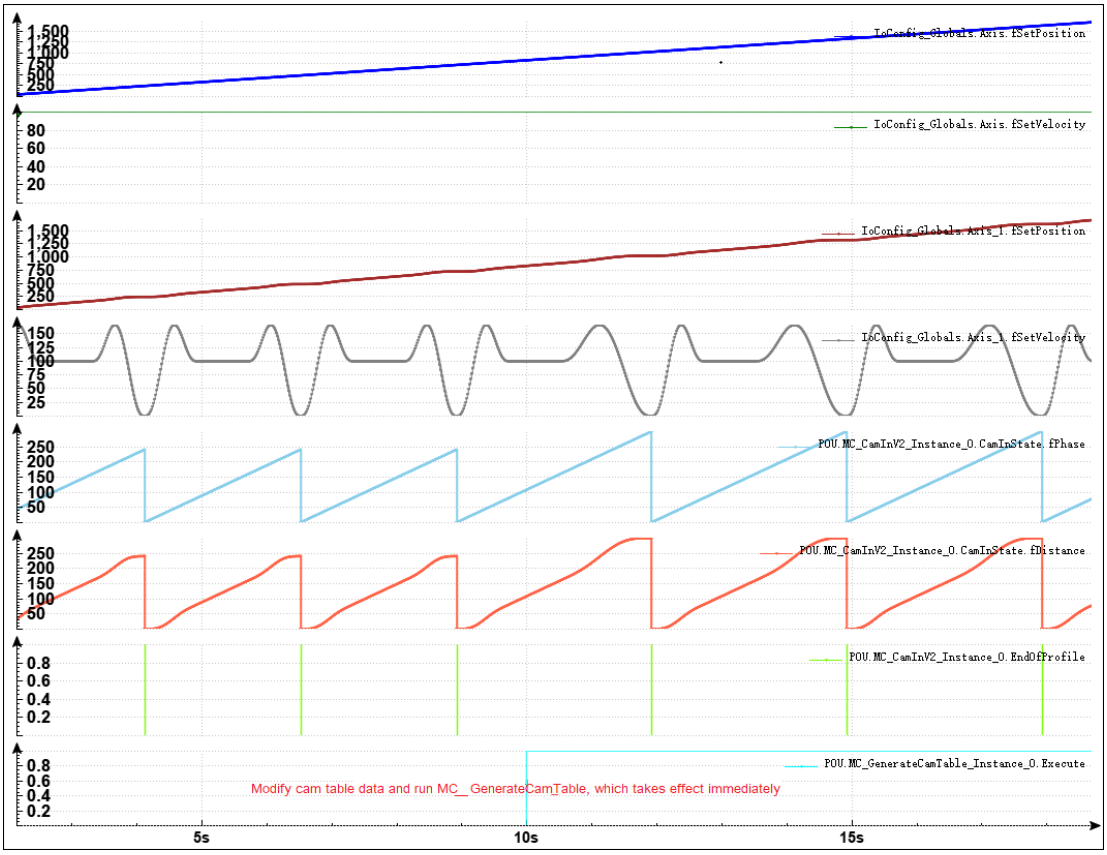
When the UpDateRightNow is set to TRUE, the modified cam table data takes effect immediately.

- Cam table generated by iFA

a. Set the UpDateRightNow pin to TRUE, and then execute the MC_CamInV2 instruction.

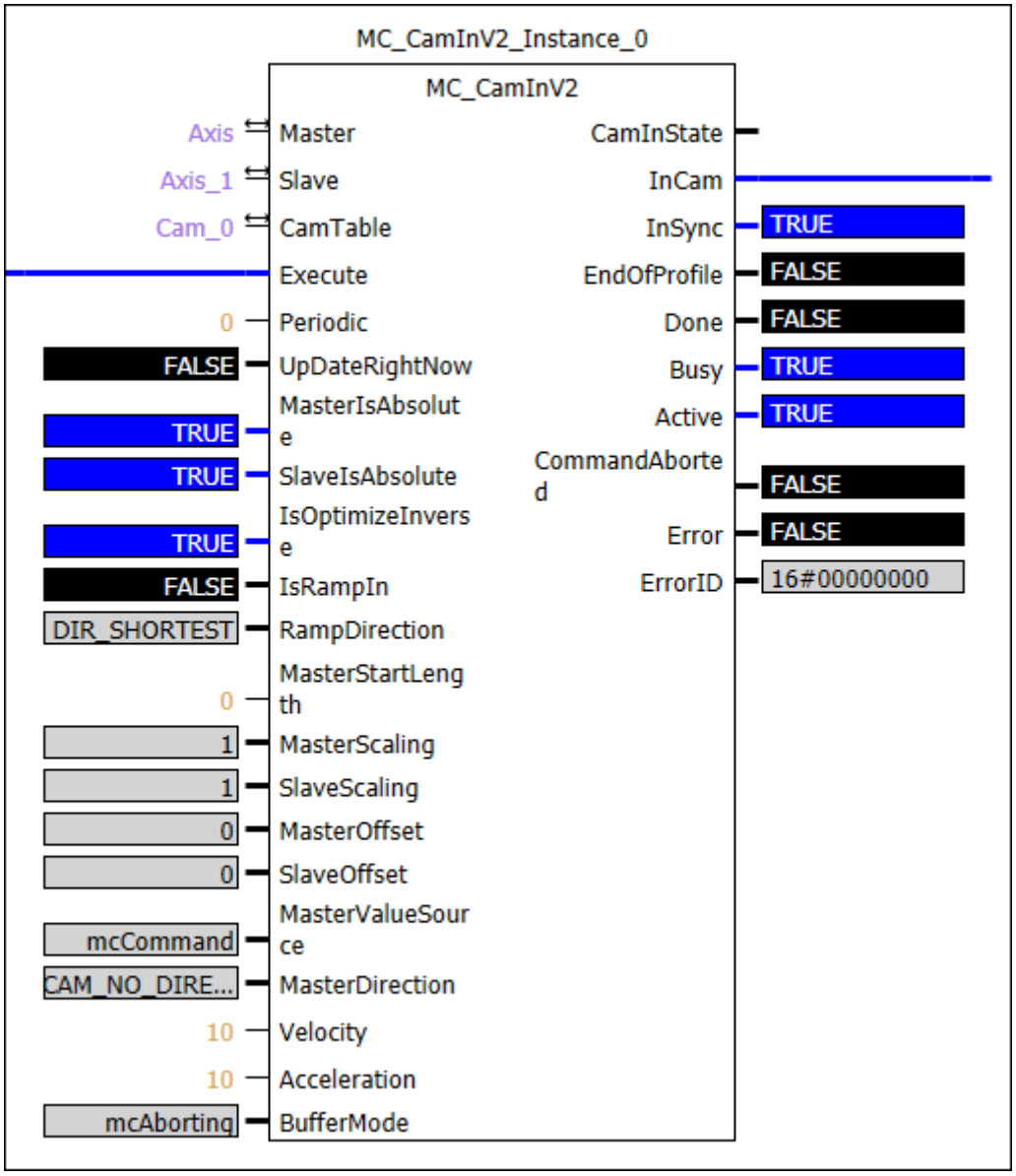


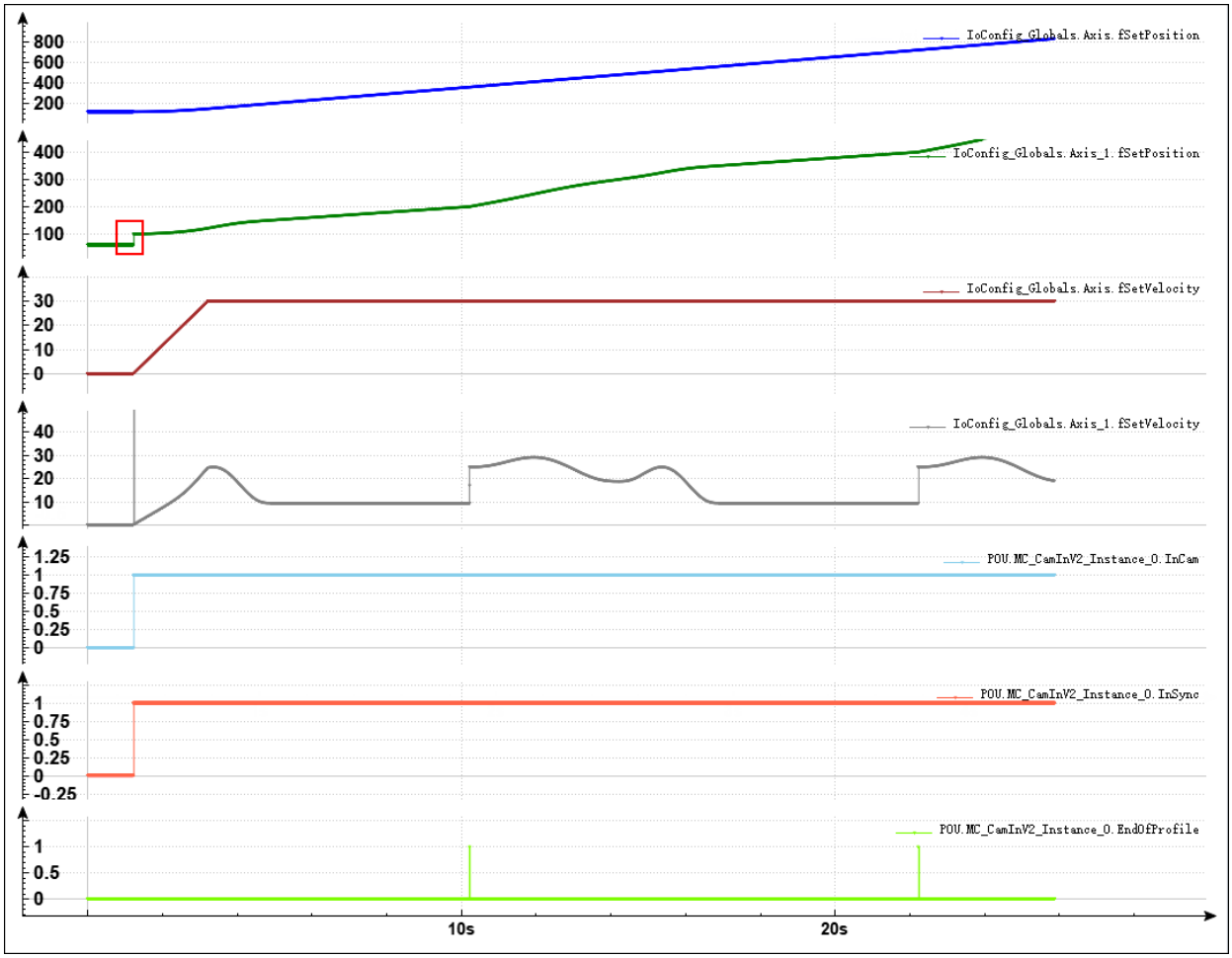
b. The operation result is shown in the figure below.



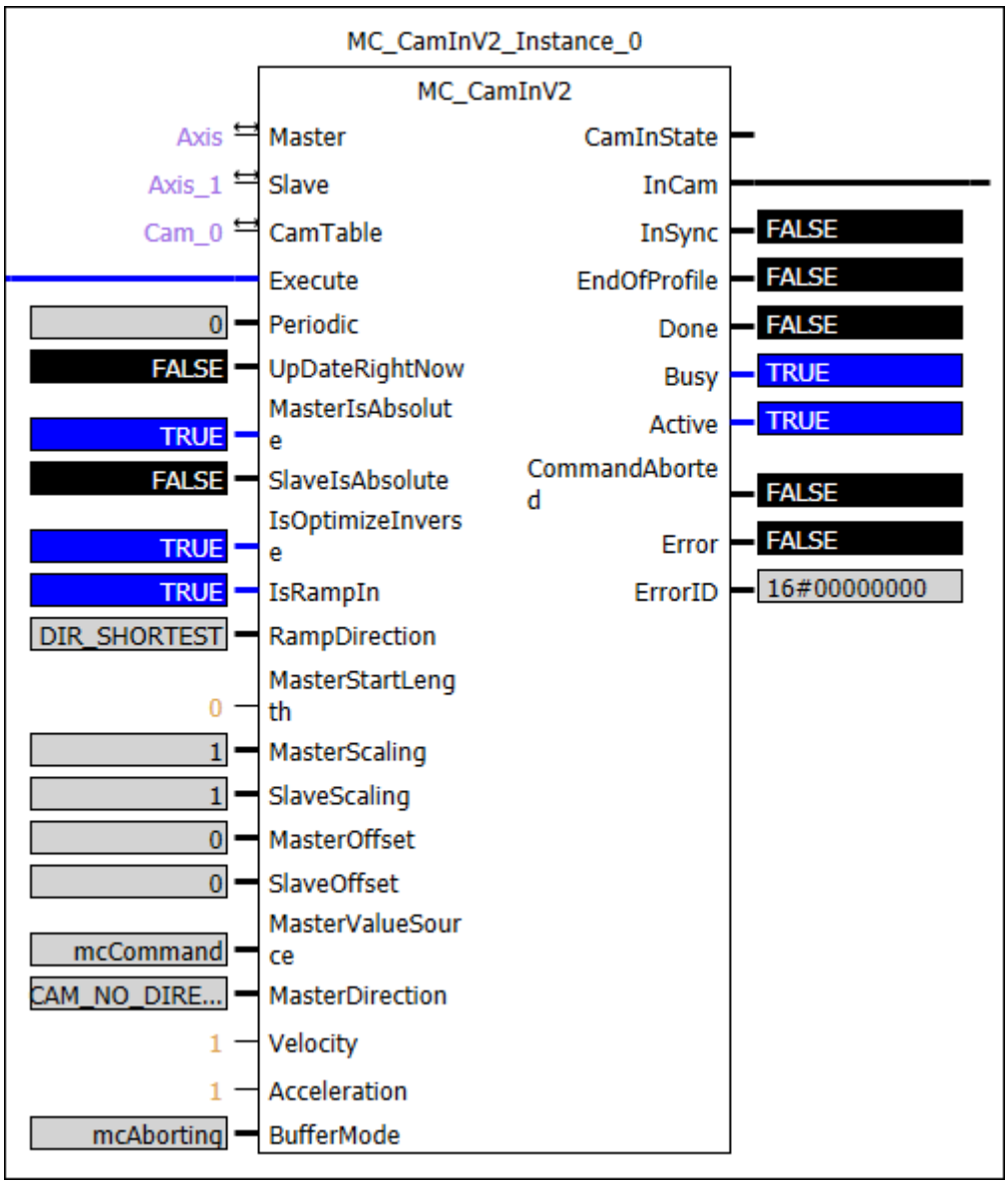
- IsRampIn: This parameter value indicates whether coupling is enabled when the slave axis starts following the master axis and decides whether the slave axis smoothly transitions or directly jumps to the target position. If the coupling compensation has not been fully executed, the InSync will not be set.

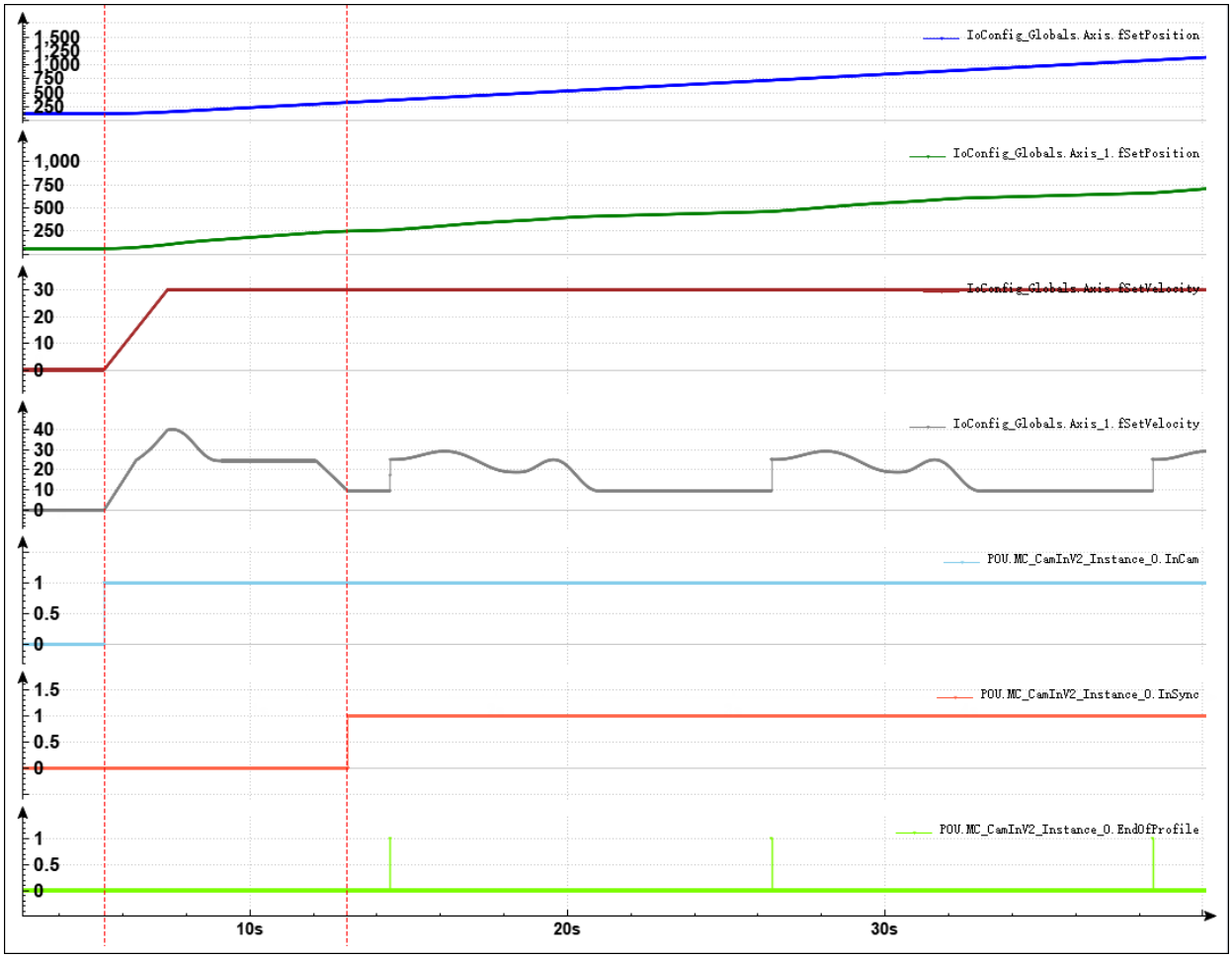
  IsRampIn = FALSE, master axis in absolute position mode, and slave axis in absolute position mode (The master axis start position is 120, and the slave axis start position is 60)


MC_CamInV2_Instance_0
MC_CamInV2
Axis
Master
Axis_1
Slave
Cam_0
CamTable
Execute
0
Periodic
FALSE
UpDateRightNow
TRUE
MasterIsAbsolute
TRUE
SlaveIsAbsolute
TRUE
IsOptimizeInverse
FALSE
IsRampIn
DIR_SHORTEST
RampDirection
0
MasterStartLength
1
MasterScaling
1
SlaveScaling
0
MasterOffset
0
SlaveOffset
mcCommand
MasterValueSource
CAM_NO_DIRE...
MasterDirection
10
Velocity
10
Acceleration
mcAborting
BufferMode
CamInState
InCam
InSync
TRUE
EndOfProfile
FALSE
Done
FALSE
Busy
TRUE
Active
TRUE
CommandAborted
FALSE
Error
FALSE
ErrorID
16#00000000



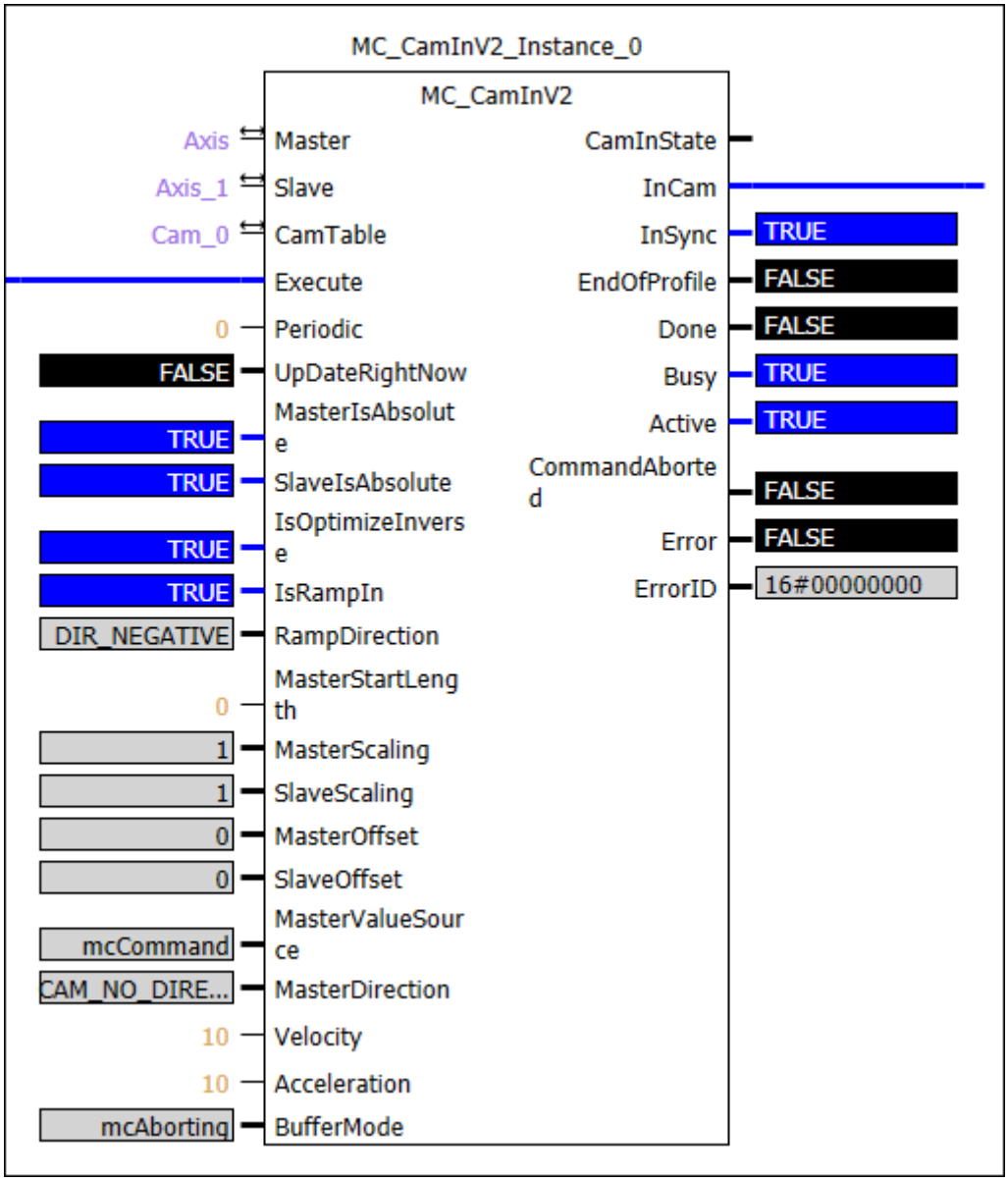
IsRampIn = TRUE, master axis in absolute position mode, and slave axis in absolute position mode (The master axis start position is 120, and the slave axis start position is 60)


MC_CamInV2_Instance_0
MC_CamInV2
Axis
Master
Axis_1
Slave
Cam_0
CamTable
Execute
0
Periodic
FALSE
UpDateRightNow
TRUE
MasterIsAbsolute
FALSE
SlaveIsAbsolute
TRUE
IsOptimizeInverse
TRUE
IsRampIn
DIR_SHORTEST
RampDirection
0
MasterStartLength
1
MasterScaling
1
SlaveScaling
0
MasterOffset
0
SlaveOffset
mcCommand
MasterValueSource
CAM_NO_DIRE...
MasterDirection
1
Velocity
1
Acceleration
mcAborting
BufferMode
CamInState
InCam
InSync
FALSE
EndOfProfile
FALSE
Done
FALSE
Busy
TRUE
Active
TRUE
CommandAborted
FALSE
Error
FALSE
ErrorID
16#00000000

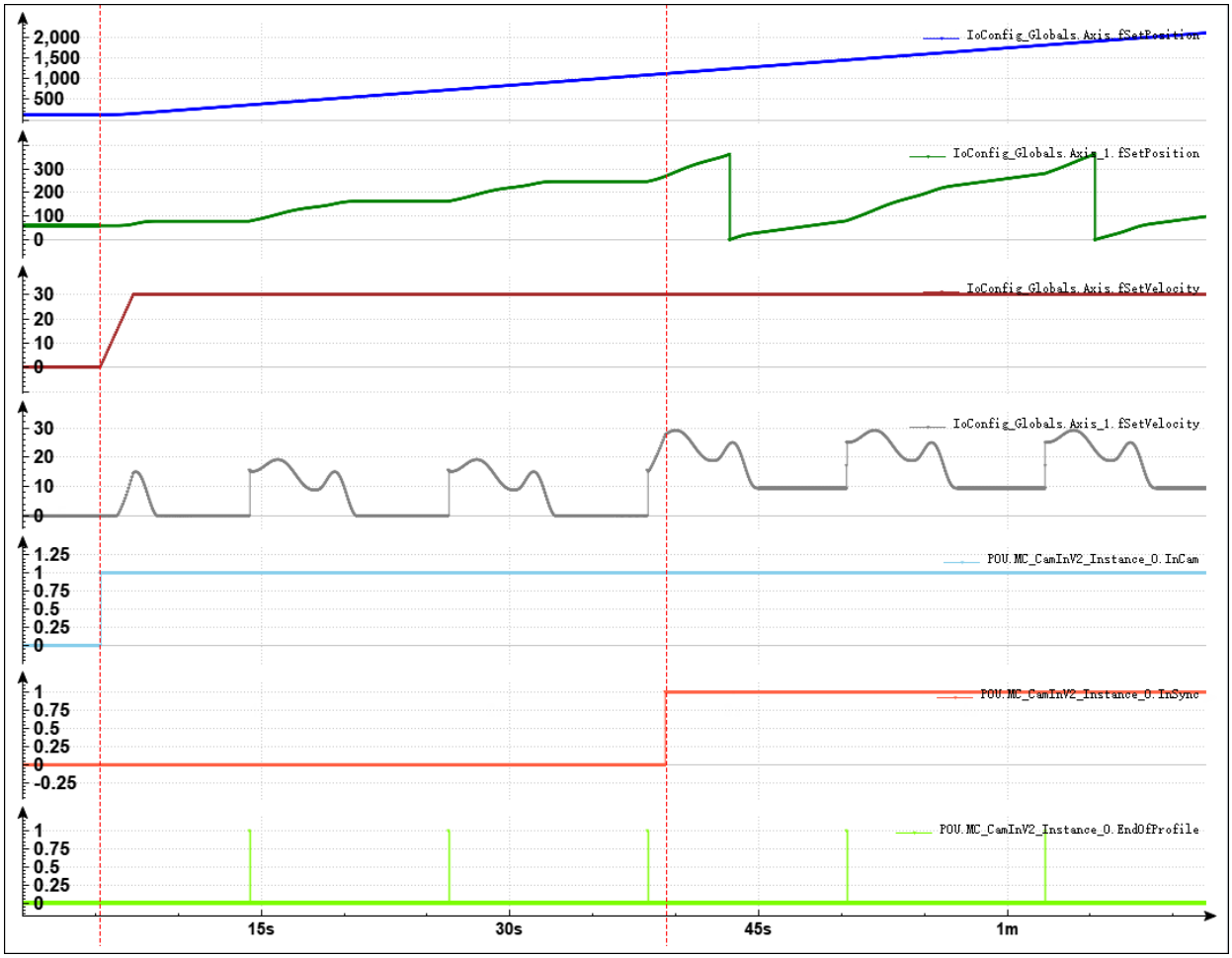


- RampDirection: For linear axes, this parameter has no effect on the coupling curve regardless of its value; for rotary axes, different coupling directions result in different coupling curves.

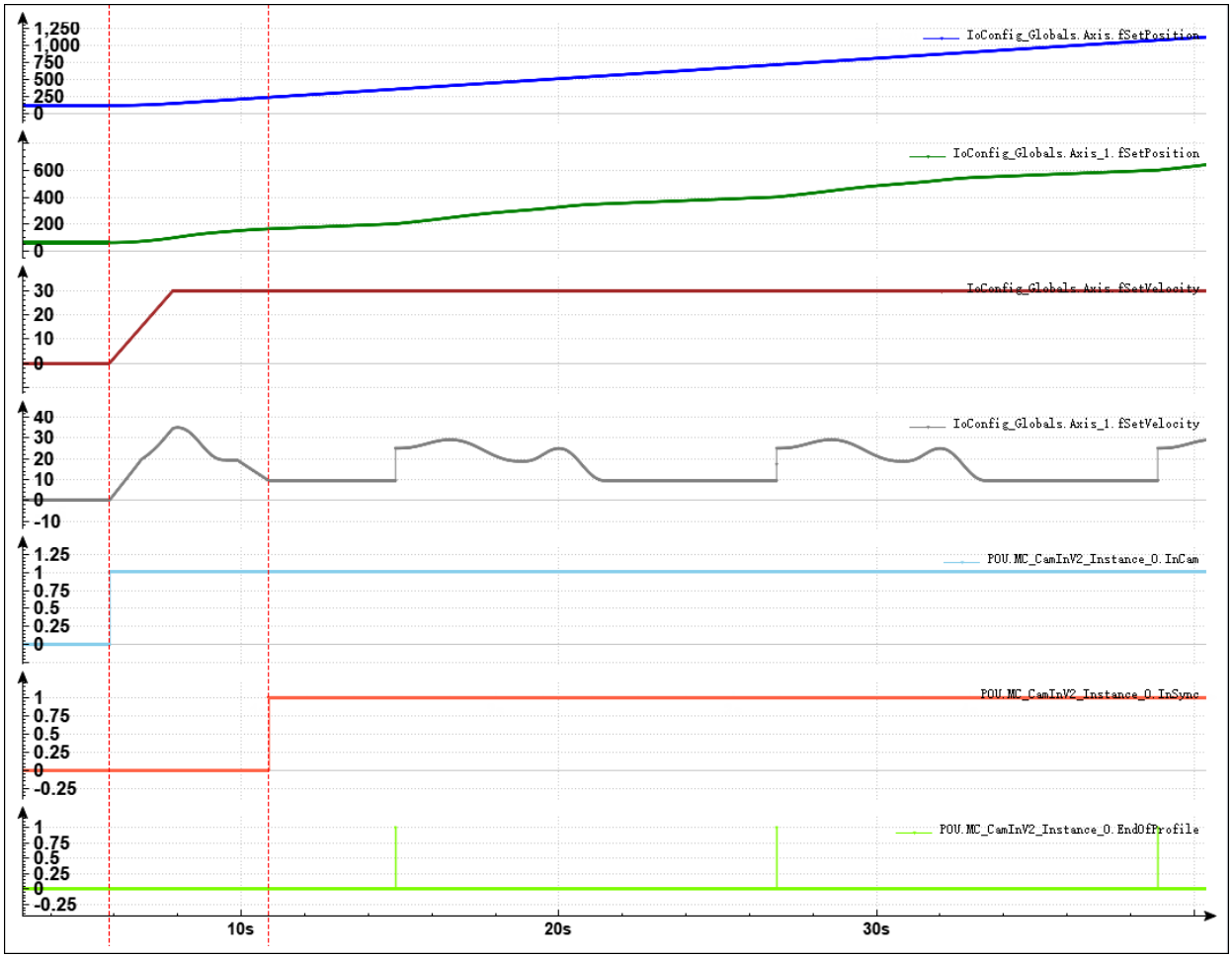
  RampDirection = DIR_NEGATIVE, IsRampIn = TRUE, master axis in absolute position mode, and slave axis in absolute position mode (The master axis start position is 120, and the slave axis start position is 60)



Slave axis = Rotary axis, DIR_NEGATIVE: Coupling compensation = -320 (40 - 360).

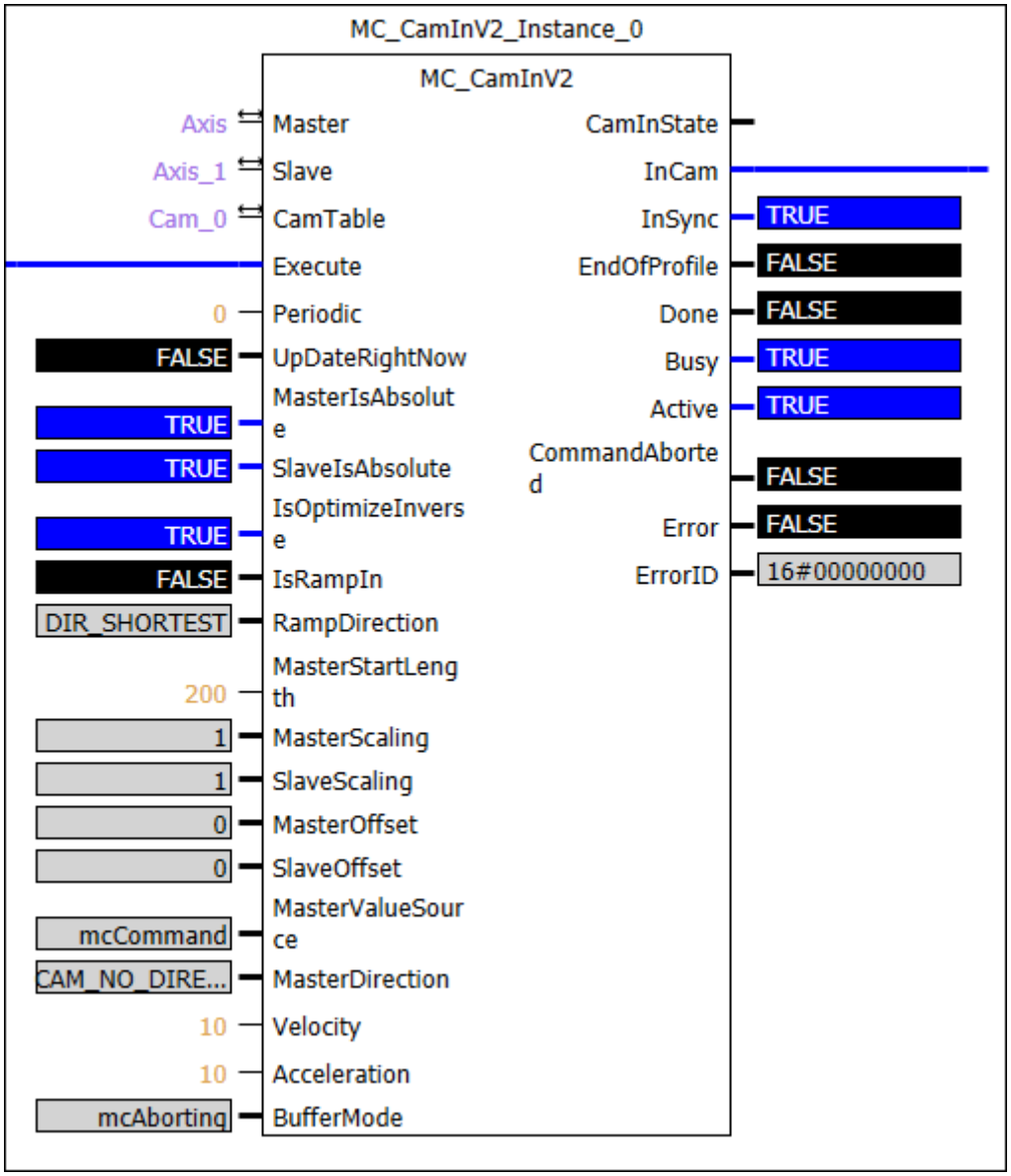


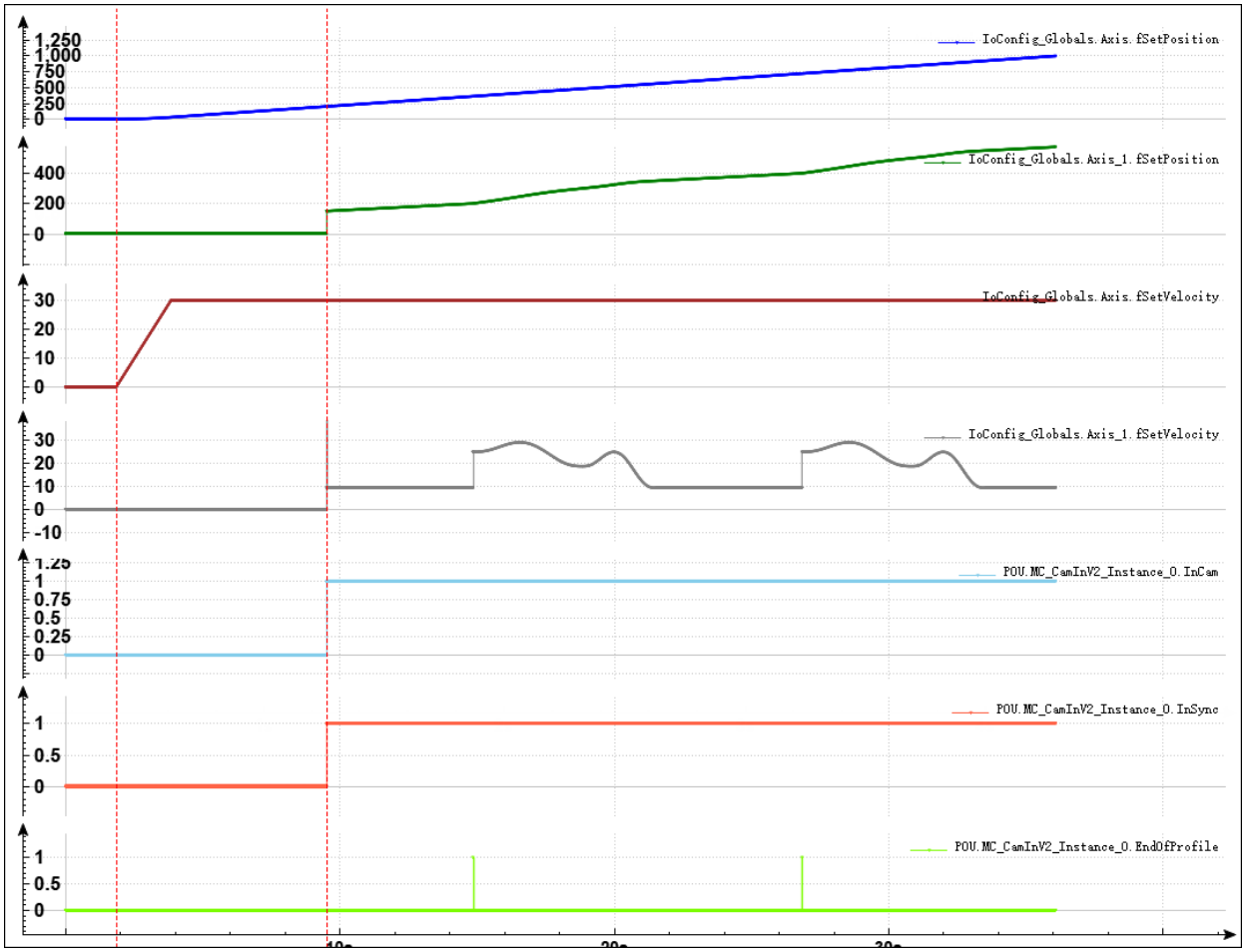
Slave axis = Linear axis, DIR_NEGATIVE: Coupling compensation = 40.



- MasterStartLength: After the instruction is triggered, the slave axis does not immediately follow the master axis. When the cumulative master axis travel distance is equal to the input value of MasterStartLength, the slave axis starts following the master axis, and the InCam is set.

  MasterStartLength = 200, master axis in absolute position mode, and slave axis in absolute position mode (The master axis start position is 0, and the slave axis start position is 0)


MC_CamInV2_Instance_0
MC_CamInV2
Axis
Master
Axis_1
Slave
Cam_0
CamTable
Execute
0
Periodic
FALSE
UpDateRightNow
TRUE
MasterIsAbsolute
TRUE
SlaveIsAbsolute
TRUE
IsOptimizeInverse
FALSE
IsRampIn
DIR_SHORTEST
RampDirection
200
MasterStartLength
1
MasterScaling
1
SlaveScaling
0
MasterOffset
0
SlaveOffset
mcCommand
MasterValueSource
CAM_NO_DIRE...
MasterDirection
10
Velocity
10
Acceleration
mcAborting
BufferMode
CamInState
InCam
InSync
TRUE
EndOfProfile
FALSE
Done
FALSE
Busy
TRUE
Active
TRUE
CommandAborted
FALSE
Error
FALSE
ErrorID
16#00000000



- **Local pulse axis**

  This instruction can be used for local pulse axes.

  When the local pulse axis is used as the master axis with Y0 or Y1 single-ended output, it is not recommended to use this instruction. ① No pulse output; ② No position feedback.

- **Timing diagram**

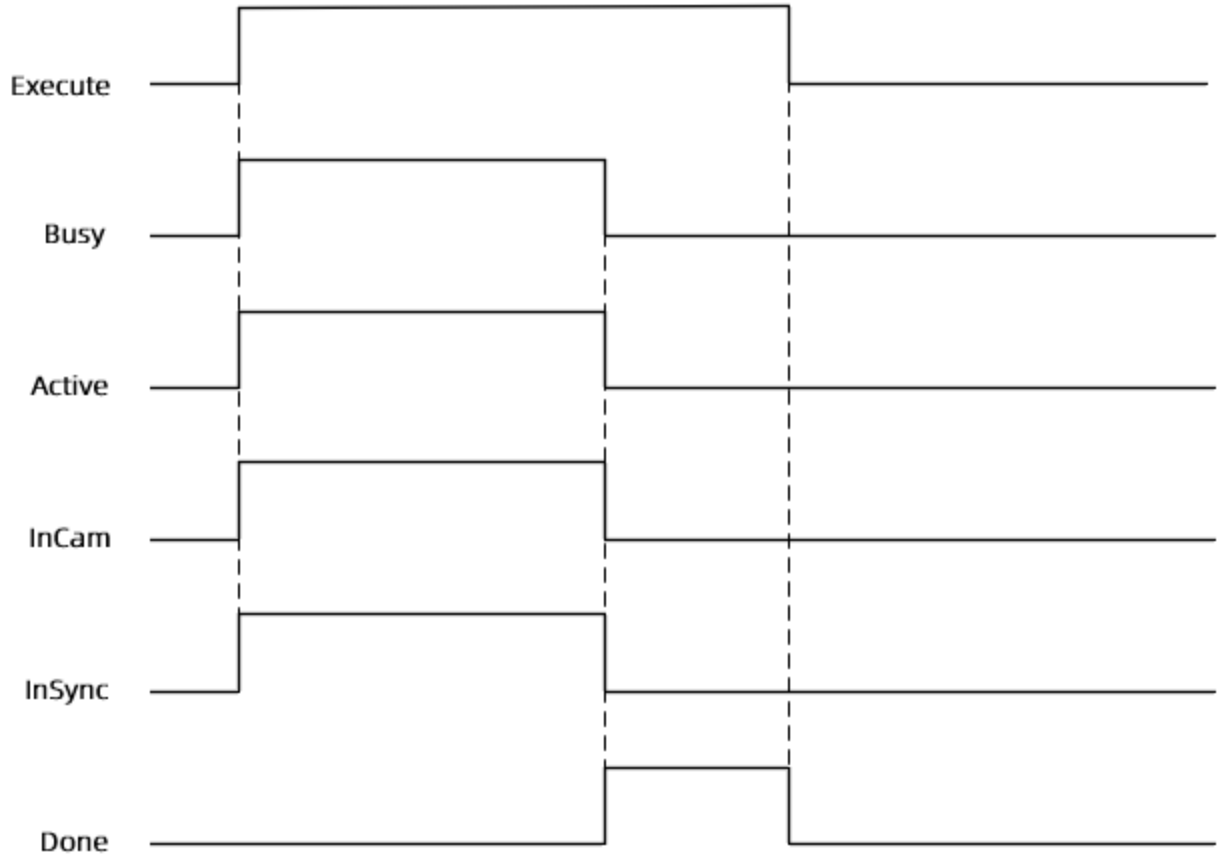


Figure 18-3-3-4-15-1 MasterStartLength = 0, IsRampIn disabled

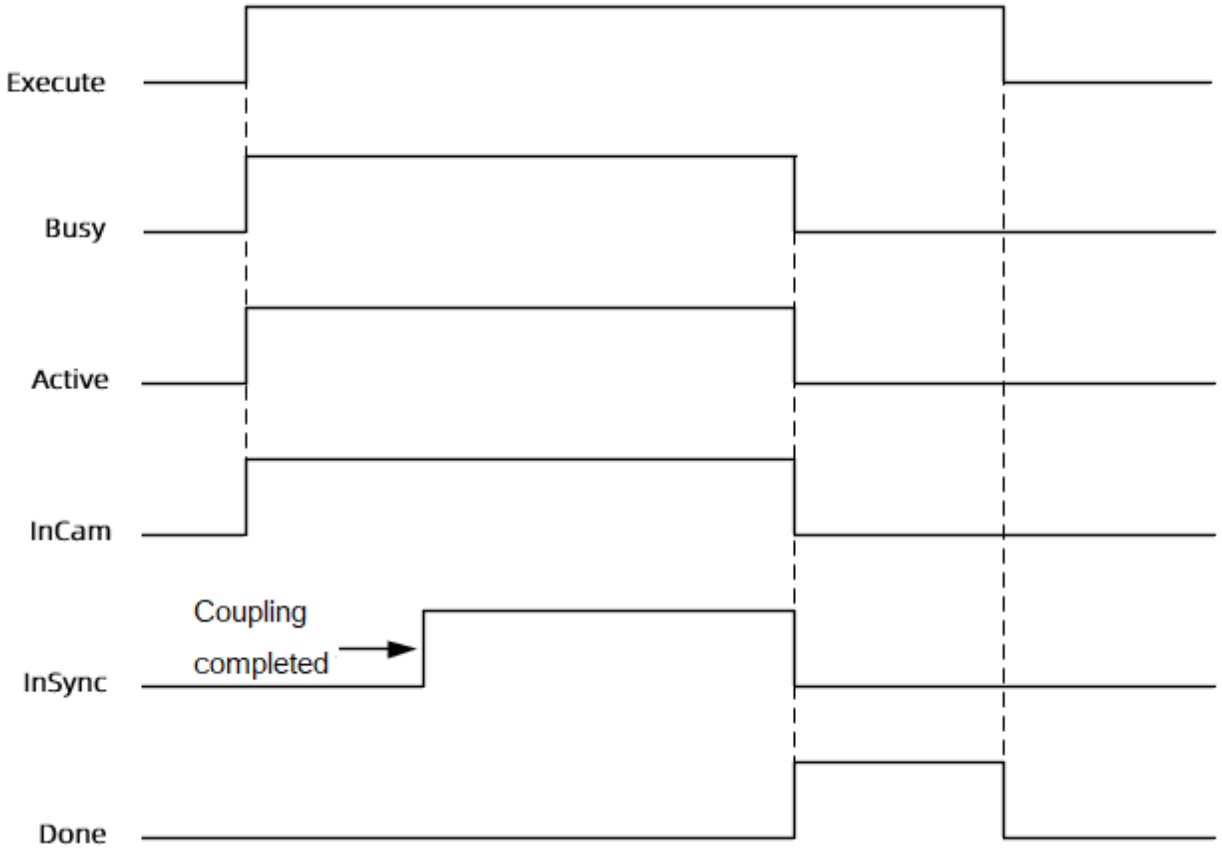


Figure 18-3-3-4-15-2 MasterStartLength = 0, IsRampIn enabled

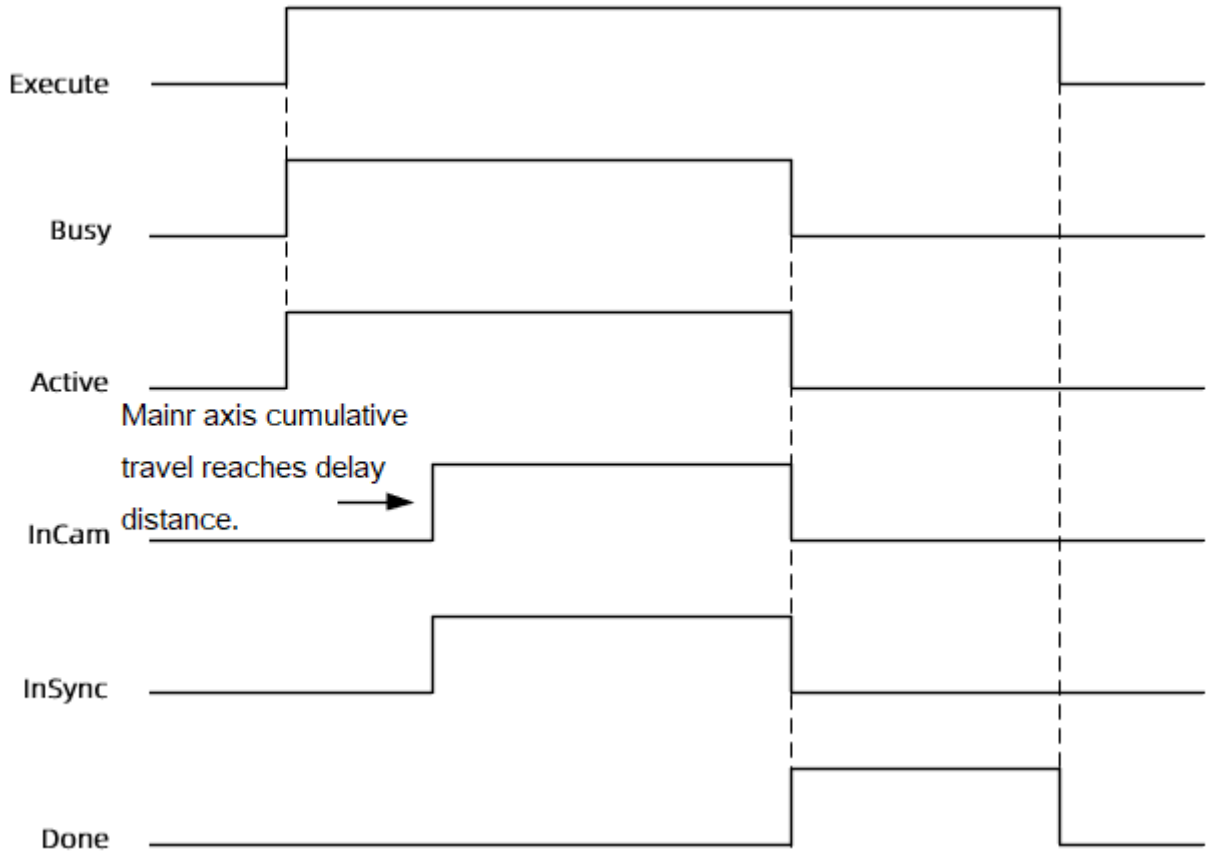


Figure 18-3-3-4-15-3 MasterStartLength ≠ 0, IsRampIn disabled

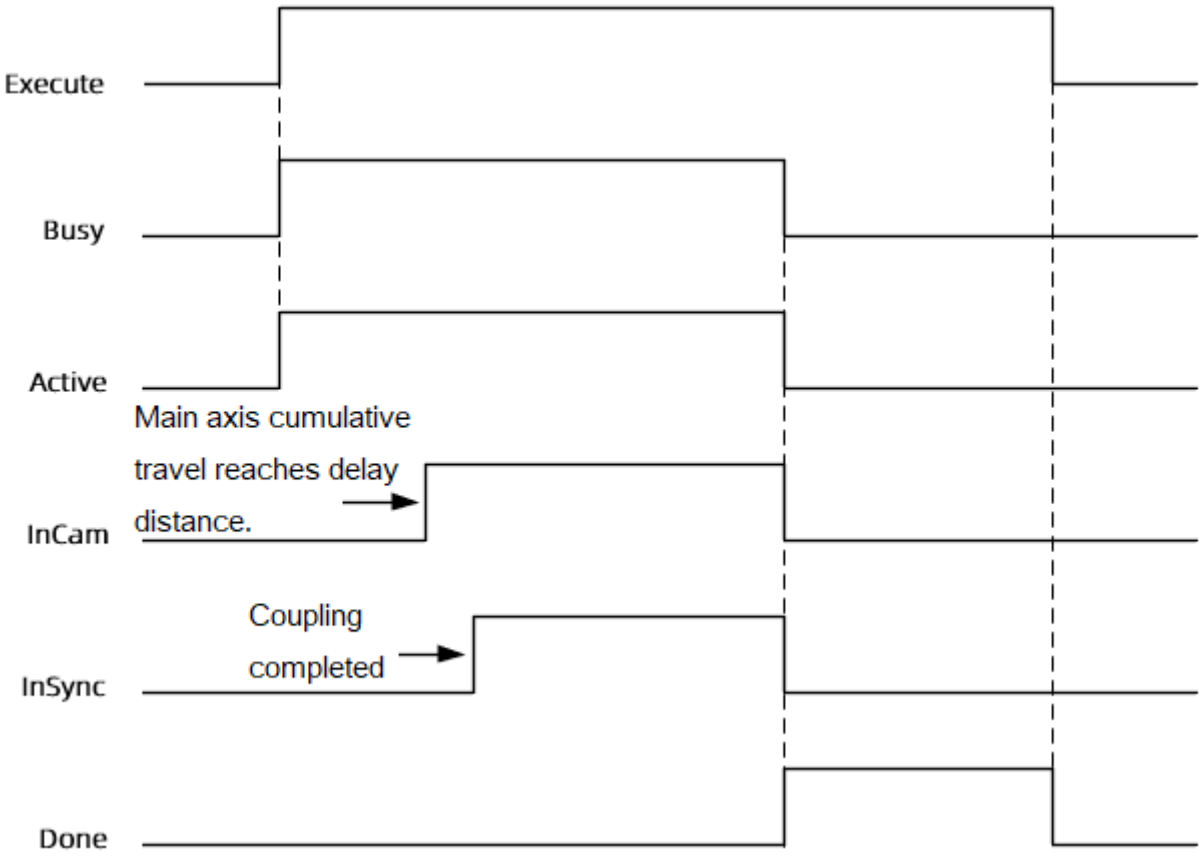


Figure 18-3-3-4-15-4 MasterStartLength ≠ 0, IsRampIn enabled

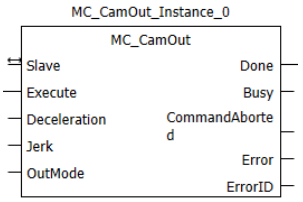
- **Faults**

For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.4.16 MC_CamOut

This instruction desynchronizes the camming between the master and slave axes.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_CamOut	Desynchronize camming		MC_CamOut_Instance_0(Slave :=, Execute :=, Deceleration :=, Jerk :=, OutMode :=, Done =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Slave	Slave axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the slave axis; an instance of AXIS_REF.

  Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Deceleration	Deceleration	LREAL	Positive number	0	Specify the maximum deceleration [u/s^2] of the slave axis when the camming between the master and slave axes is desynchronized.
Jerk	Jerk	LREAL	Positive number or 0	0	Slave axis jerk parameter.
OutMode	Desynchronization mode	18.3.3.2.3.7 MC_OUTMODE: Desynchronization Mode Enumeration	0: mcDecelerationStop 1: mcConstantVelocity 2: mcImmediateStop	0	0: Deceleration stop. 1: Constant velocity before desynchronization. 2: Immediate stop after performing the current cam table.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received. OutMode = 0: The variable shows the value FALSE after the instruction is executed. OutMode = 1: The variable still shows the value TRUE after the instruction is executed until it is aborted by the next instruction.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted by another instruction.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the deceleration limit.	Any deceleration less than 0 will be automatically corrected to 10% of the deceleration limit.	The deceleration equal to 0 will be automatically corrected to 10% of the deceleration limit.

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Jerk	Jerk	If the motion profile is set to S-curve, any jerk that exceeds the software limit will be automatically corrected to the jerk limit.	If the motion profile is set to S-curve, any jerk less than 0 will be automatically corrected to 10% of the jerk limit.	If the motion profile is set to S-curve, the jerk equal to 0 will be automatically corrected to 10% of the jerk limit.

- **Function**

  Bus-controlled axis:

  - The MC_CamOut instruction desynchronizes the camming of the slave axis.
  - This instruction is executed on a rising edge of the input variable Execute.
  - This instruction must be executed after the MC_CamIn instruction is enabled. If you execute this instruction on an axis that is not in camming, a fault will occur.
  - During the execution of this instruction, the MC_MoveSuperImposed instruction cannot be used; once the MC_MoveSuperImposed instruction is triggered, CommandAborted is set for the MC_MoveSuperImposed instruction.
  - During the execution of the MC_CamIn and MC_MoveSuperImposed instructions, if the MC_CamOut instruction is triggered, the superimposed motion will not be aborted during the camming; when the camming ends, the slave axis performs an immediate stop with the superimposed motion aborted.

    OutMode

    You can specify the mode by which the slave axis disengages from the master axis by defining the value of the input variable OutMode.

  - mcDecelerationStop: Deceleration stop.

    The slave axis decelerates to stop according to the Deceleration specified in this instruction.

  - mcConstantVelocity: Constant velocity.

    The slave axis continues to move at a constant velocity while the camminng is desynchronized.

  - mcImmediateStop: Immediate stop.

    The slave axis immediately stops after performing the current cam table.

  Local pulse axis:

  This instruction can be used for local pulse axes.

  Re-execution of motion control instructions

  - This instruction can be re-executed.
  - Do not modify the value of OutMode when this instruction is re-executed.
  - When the OutMode is set to 0 (mcDecelerationStop), and this instruction is re-executed after modifying the Deceleration and Jerk values, the slave axis disengages from the master axis in the deceleration stop mode according to the modified parameter values.
  - If either of the modified Deceleration and Jerk values is invalid, it will be automatically corrected by the function block without a fault message. The slave axis velocity will decelerate to 0 within one cycle, with the axis state switching from DiscreteMotion to ErrorStop.

Software limit

When the slave axis position is beyond the software limit while disengaging from the master axis, this instruction is aborted, and the slave axis velocity decelerates to zero within one cycle. In the mcDecelerationStop mode, the slave axis state switches from DiscreteMotion to ErrorStop. In the mcConstantVelocity mode, the slave state switches from ContinuousMotion to ErrorStop. In the mcImmediateStop mode, the slave axis state switches from SynchronizedMotion to ErrorStop.

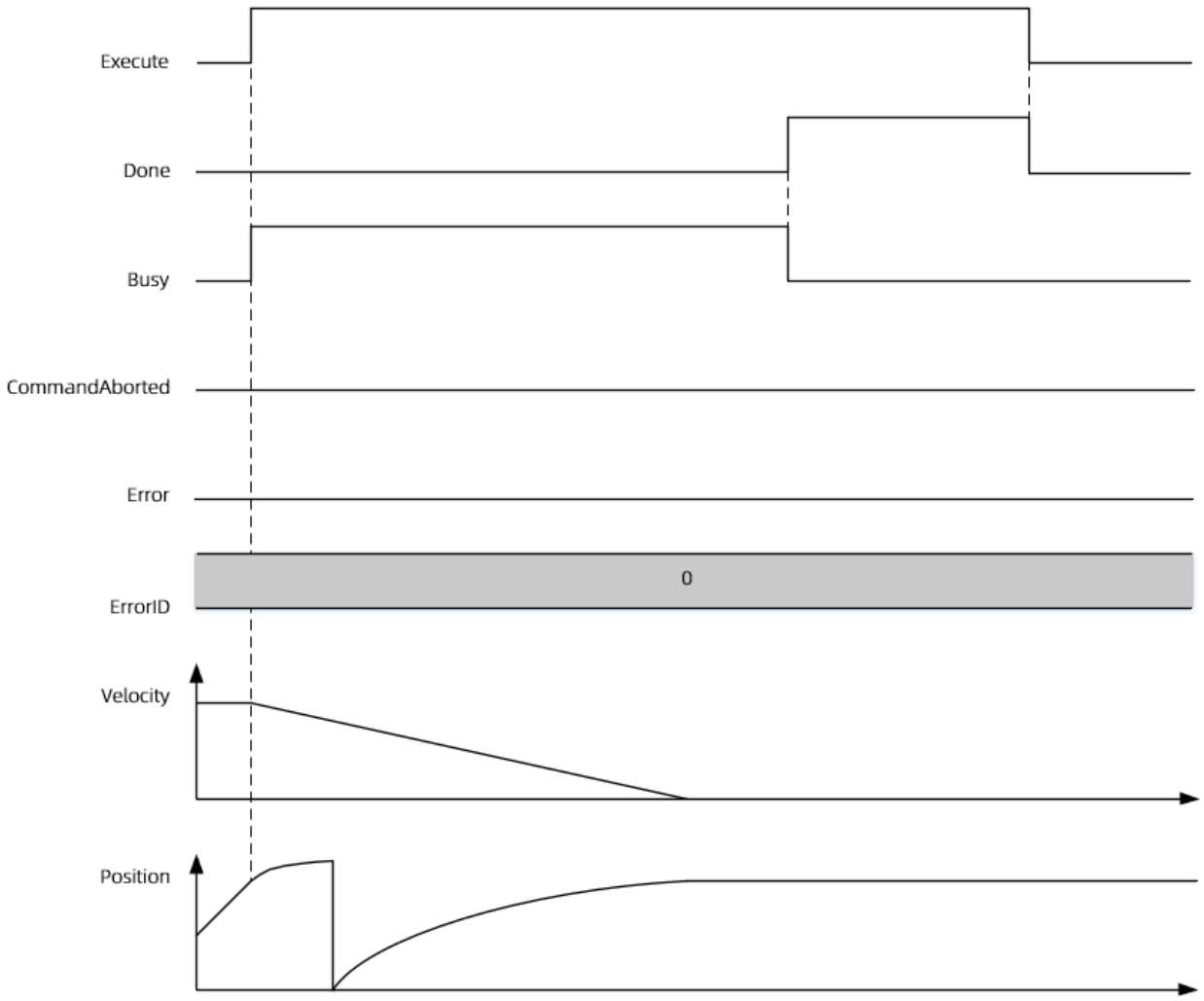
- **Precautions**
  - When the OutMode is set to 1 (mcConstantVelocity) After the instruction is executed, the parameter Done is set to TRUE, and the parameter Busy remains TRUE until this instruction is aborted by next instruction, at which point the parameter Busy is set to FALSE. Therefore, to stop the slave axis in the mcConstantVelocity mode, this instruction must be used in conjunction the MC_Stop instruction.
  - When the OutMode is set to 1, the slave axis state switches SynchronizedMotion to AXIS_CONTINUOUSMOTION after the instruction is executed. When the OutMode is set to 0 (mcDecelerationStop), the slave axis state switches from SynchronizedMotion to DiscreteMotion after the instruction is executed, and becomes StandStill after the output variable Done is set. When the OutMode is set to 2 (mcImmediateStop), the slave axis state remains SynchronizedMotion after the instruction is executed until the current camming ends, and becomes StandStill after the output variable Done is set.
  - During the execution of this instruction, if a fault occurs on the slave axis, the command aborted flag will be set, and the slave axis will stop immediately.
  - This instruction can abort an executing MC_MoveSuperImposed instruction.
  - If the MC_MoveSuperImposed instruction is executed during the execution of this instruction, the OutMode = 0 (mcDecelerationStop) will cause the MC_MoveSuperImposed instruction to abort directly.
  - If the MC_MoveSuperImposed instruction is executed during the execution of this instruction, the OutMode = 2 (mcImmediateStop) will cause the MC_MoveSuperImposed instruction to abort after the current camming cycle ends, with the velocity being set to zero.
- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates the instruction has been executed successfully; the parameter Busy indicates that the current function block is being executed; the parameter CommandAborted of the function block indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

  The camming of the slave axis is desynchronized on a rising edge of the input variable Execute. At this point, the slave axis may not be stopped.

  If the velocity of the slave axis is not zero before this instruction is executed, the camming will be desynchronized after the output variable Done is set, but the slave axis will keep moving at its previous velocity.

  When this instruction is executed on a slave axis not in camming, the output variable Error shows the value TRUE.

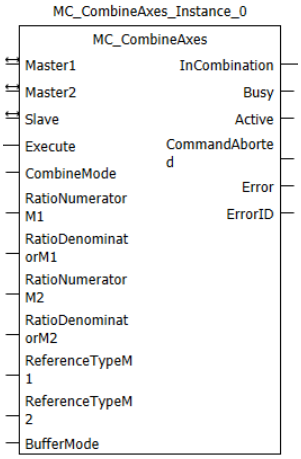


- **Faults**
  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.4.17 MC_CombineAxes

This instruction outputs the sum or difference of the instruction positions of two axes as a combined instruction position. It is designed primarily for use in scenarios requiring multiple master axes.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_CombineAxes	Output the sum or difference of the instruction positions of two axes as a combined instruction position.		MC_CombineAxes_Instance_0(Master1 :=, Master2 :=, Slave :=, Execute :=, CombineMode :=, RatioNumeratorM1 :=, RatioDenominatorM1 :=, RatioNumeratorM2 :=, RatioDenominatorM2 :=, ReferenceTypeM1 :=, ReferenceTypeM2 :=, BufferMode :=, InCombination =>, Busy =>, Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Master1	Master axis 1	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the master axis 1; an instance of AXIS_REF.
Master2	Master axis 2	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the master axis 2; an instance of AXIS_REF.
Slave	Slave axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the slave axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
CombineMode	Combine mode	18.3.3.2.2.12 MC_COMBINE_MODE: Combine Mode Enumeration	0: mcAddAxes 1: mcSubAxes	0	0: Addition. 1: Subtraction.
RatioNumeratorM1	Master axis 1 gear ratio numerator	DINT	Not equal to 0	1	Master axis 1 gear ratio numerator.
RatioDenominatorM1	Master axis 1 gear ratio denominator	UINT	Positive number	1	Master axis 1 gear ratio denominator.
RatioNumeratorM2	Master axis 2 gear ratio numerator	DINT	Not equal to 0	1	Master axis 2 gear ratio numerator
RatioDenominatorM2	Master axis 2 gear ratio denominator	UINT	Positive number	1	Master axis 2 gear ratio denominator.
ReferenceTypeM1	Master axis 1 position type selection	18.3.3.2.2.11 MC_REFERENCE_TYPE: Data Type Enumeration	0: mcCommand 1: mcFeedback 2: mcLaterCommand	0	0: Value calculated in the previous task cycle (fSavePosition). 1: Value obtained in the same task cycle (fActPosition). 2: Value calculated in the same task cycle (fSetPosition).

Input Variable	Name	Data Type	Valid Range	Default Value	Description
ReferenceType M2	Master axis 2 position type selection	18.3.3.2.2.11 MC_REFERENCE_TYPE: Data Type Enumeration	0: mcCommand 1: mcFeedback 2: mcLaterCommand	0	0: Value calculated in the previous task cycle (fSavePosition). 1: Value obtained in the same task cycle (fActPosition). 2: Value calculated in the same task cycle (fSetPosition).
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting	0	0: Aborting. (Reserved for other modes)

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
InCombination	Axes combined	BOOL	TRUE, FALSE	FALSE	TRUE when axes are combined.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Controlling	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted by another instruction.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus-controlled axis:

  - You can calculate the position reference of the slave axis by specifying parameters such as Master1, Master2, CombineMode, RatioNumeratorM1, RatioDenominatorM1, RatioNumeratorM2, RatioDenominatorM2, ReferenceTypeM1, and ReferenceTypeM2.
  - This instruction is executed on a rising edge of the input variable Execute.
  - If bus cycle is specified for the cycle task of the master axis, the velocity of the slave axis may show a sudden jump after synchronization. It is recommended that users specify EtherCAT cycle for the cycle task of both the master and slave axes before enabling a synchronization instruction.
  - MC_Home, MC_Homing, or MC_HomingWithParameter can be used during the instruction execution.

    This instruction places the master axis in the AXIS_HOMING state. If the instruction is executed while the master axis is already in this state, the master axis signal source "ReferenceTypeM1" or "ReferenceTypeM2" must be set to "mcLatestCommand"; otherwise, the slave axis will overspeed.

    Local pulse axis:

  - This instruction can be used for local pulse axes.
  - When the local pulse axis is used as the master axis with Y0 or Y1 single-ended output, it is not recommended to use this instruction. ① No pulse output; ② No position feedback.

    Re-execution of motion control instructions

    This instruction cannot be re-executed.

    Position type selection

    You can specify ReferenceTypeM1 and ReferenceTypeM2 for Master1 and Master2 based on the following options:

  - mcCommand: Commanded position (Value calculated in the previous task cycle, fSavePosition).

    For the current cycle, use the master axis position calculated in the previous task cycle.
  - mcFeedback: Actual position (Value obtained in the same task cycle, fActPosition).

    Use the actual master axis position obtained in the same task cycle.
  - mcLaterCommand: Commanded position (Value calculated in the same task cycle, fSetPosition).

    Use the master axis position calculated in the same task cycle.

    CombineMode

  - You can select the mode for calculating the position reference of the Slave by defining the value of CombineMode of this instruction. When CombineMode is set to 0 (Addition), the position reference of the Slave is the sum of the relative positions of Master1 and Master2.
  - When CombineMode is set to 1 (Subtraction), the position reference of the Slave is the difference of the relative positions of Master1 and Master2.

- **Precautions**
  - It is not recommended to use this instruction in combination with the MC_Phasing instruction.
  - The variables Master1, Master2, and Slave must be different; otherwise, this instruction will issue a fault message.
  - The Slave position output by this instruction can be used as the master axis input parameter for synchronous control instructions.
  - If this instruction encounters a fault or is aborted, the Slave will stop immediately, without affecting the operation of Master1 and Master2.
- **Timing diagram**

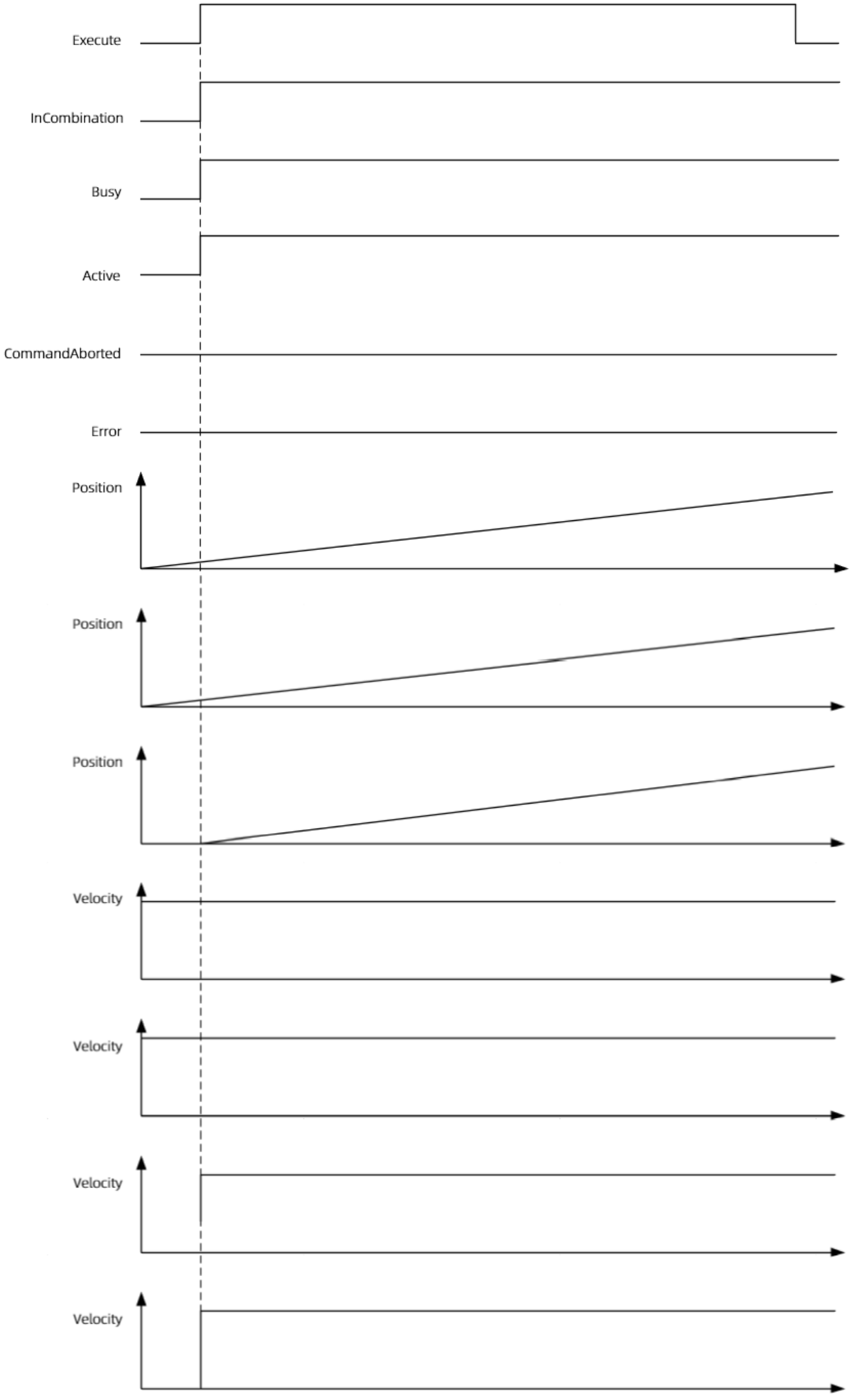
  The parameter Execute of the function block must be triggered on a rising edge. The parameter Active of the function block indicates the axis is being controlled; the parameter Busy indicates that the current function block is being executed; the parameter CommandAborted indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

  The output variable Busy shows the value TRUE at the same time as Execute is set to TRUE. In the next cycle, the output parameter Active shows the value TRUE.

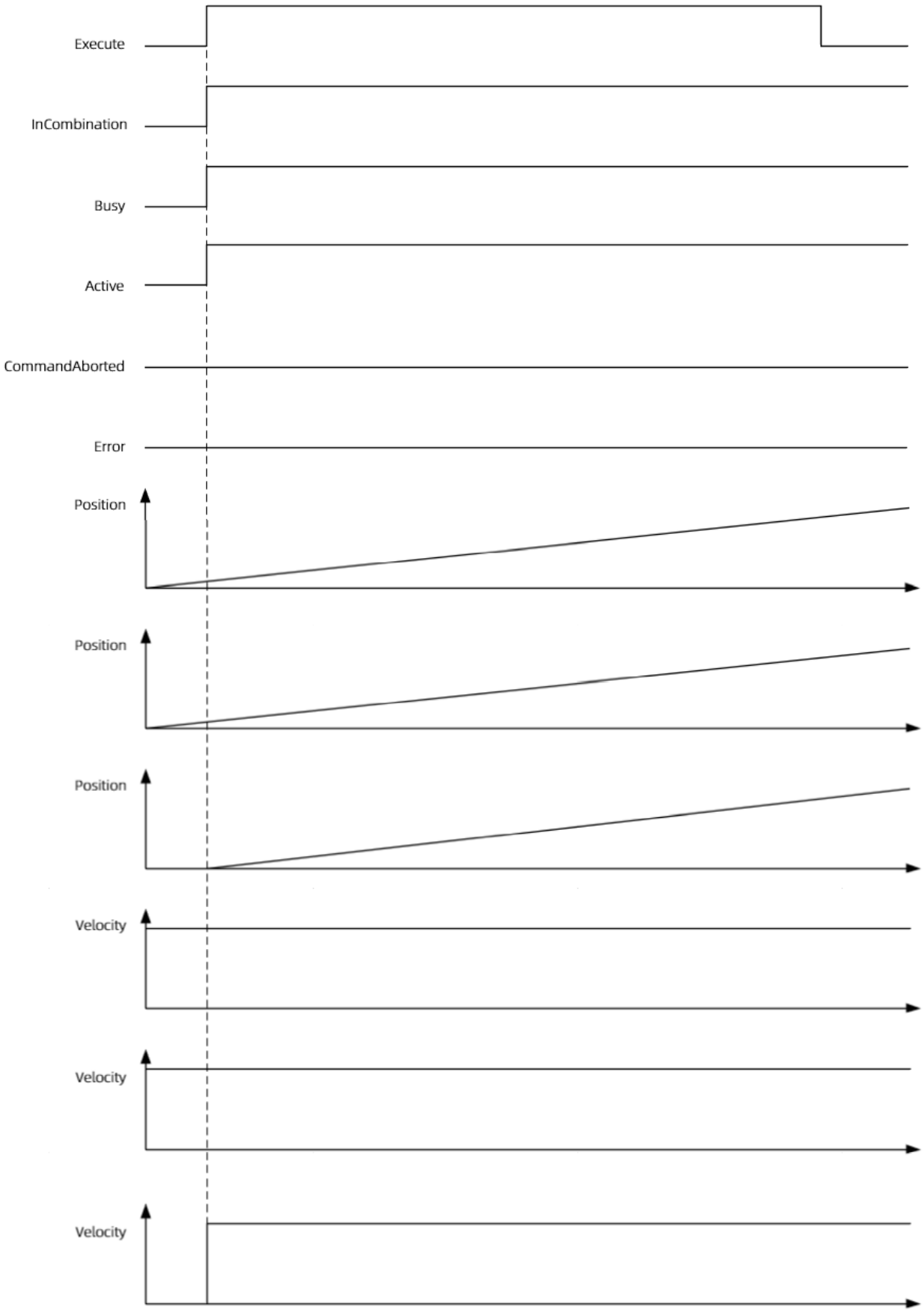
  The output variable InCombination shows the value TRUE in the cycle where the combined output starts.

  When this instruction is aborted by another instruction, the output variable CommandAborted shows the value TRUE, and the variables Busy, Active, and InCombination show the value FALSE.

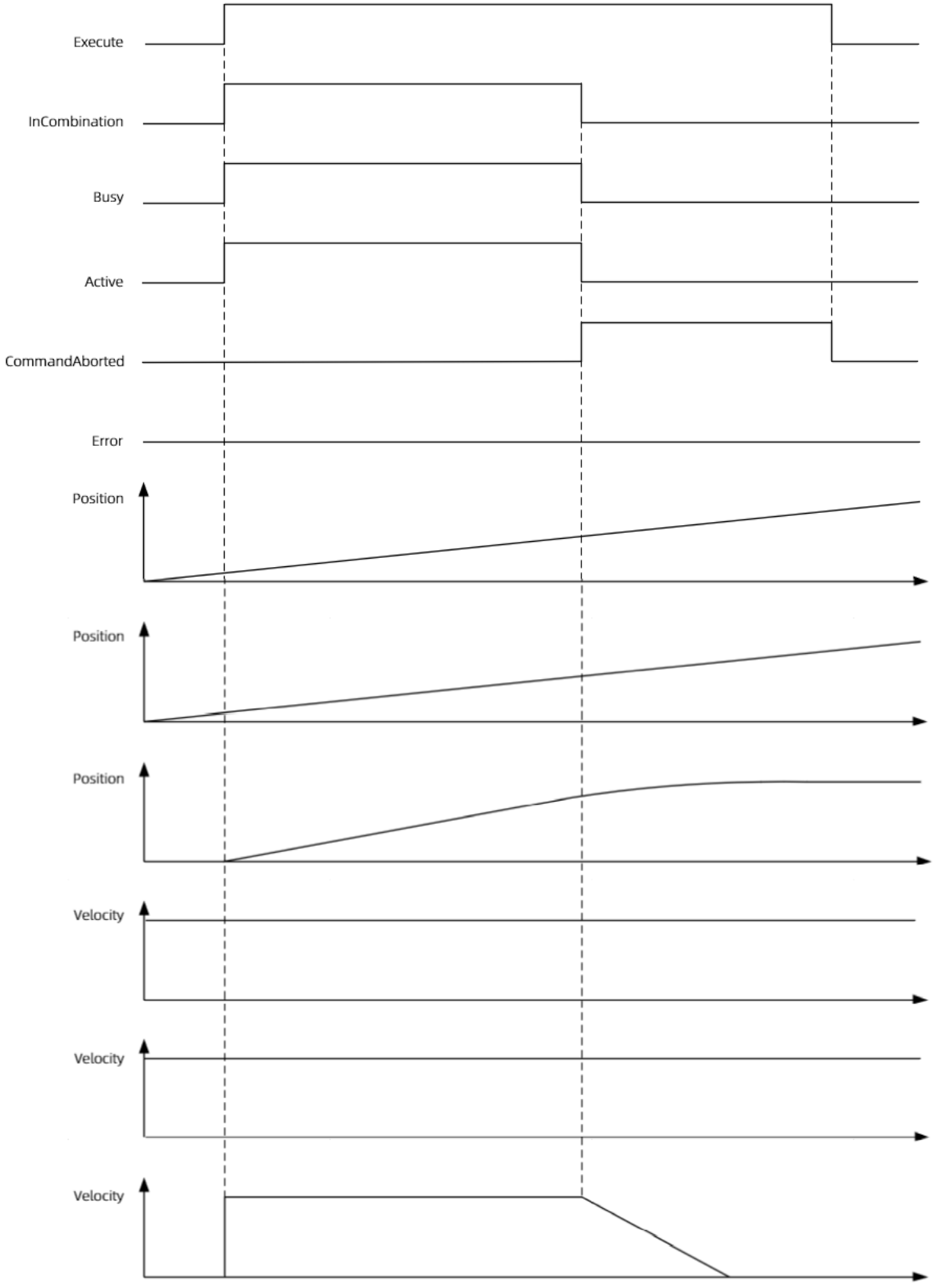
  - When CombineMode is set to _mcAddAxes, the timing diagram is as follows.


Execute
InCombination
Busy
Active
CommandAborted
Error
Position
Position
Position
Velocity
Velocity
Velocity
Velocity

◦ When CombineMode is set to _mcSubAxes, the timing diagram is as follows.



◦ When the instruction is aborted, the timing diagram is as follows.



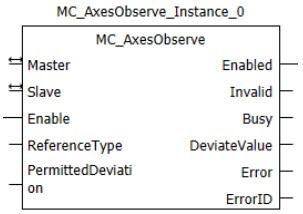
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.4.18 MC_AxesObserve

This instruction observes the deviation between the position references or actual positions for the specified axes to see if it exceeds the permitted value.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_AxesObserve	Observe position deviation between master and slave axes		MC_AxesObserve_Instance_0(Master :=, Slave :=, Enable :=, ReferenceType :=, PermittedDeviation :=, Enabled =>, Invalid =>, Busy =>, DeviateValue =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Master	Master axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the master axis; an instance of AXIS_REF.
Slave	Slave axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the slave axis; an instance of AXIS_REF.

  Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enabled. FALSE: Disabled.

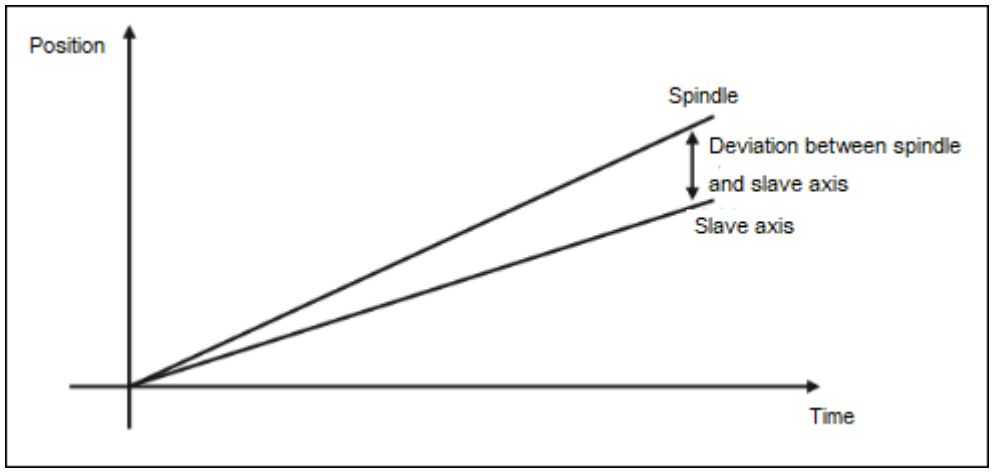
Input Variable	Name	Data Type	Valid Range	Default Value	Description
ReferenceType	Position type selection	18.3.3.2.2.11 MC_REFERENCE_TYPE: Data Type Enumeration	0: mcCommand 1: mcFeedback 2: mcLaterCommand	0	0: Value calculated in the previous task cycle (fSavePosition). 1: Value obtained in the same task cycle (fActPosition). 2: Value calculated in the same task cycle (fSetPosition).
PermittedDeviation	Permitted position deviation between master and slave axes	LREAL	Positive number or 0	0	Permitted maximum position deviation between master and slave axes.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Enabled	Enabled	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is enabled.
Invalid	Invalid	BOOL	TRUE, FALSE	FALSE	TRUE when the position deviation is out of the PermittedDeviation; FALSE if not (this flag is continuously updated as long as Enable shows the value TRUE).
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
DeviateValue	Real-time position deviation between axes	LREAL	-	-	Output the position difference between the specified master and slave axes.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**
  - This instruction observes the deviation between the position references or actual positions for the specified axes to see if it exceeds the permitted value.

- This instruction is executed when the input variable Enable shows the value TRUE.
- This instruction does not affect the motion of the master and slave axes.
- When DeviatedValue is greater than PermittedDeviation, the variable Invalid shows the value TRUE.



Position type selection

You can select ReferenceType for the in-out variable Master of this instruction based on the following options.

- mcCommand: Commanded position (Value calculated in the previous task cycle, fSavePosition).

  For the current cycle, use the master axis position calculated in the previous task cycle.

- mcFeedback: Actual position (Value obtained in the same task cycle, fActPosition).

  Use the actual master axis position obtained in the same task cycle.

- mcLaterCommand: Commanded position (Value calculated in the same task cycle, fSetPosition).

  Use the master axis position calculated in the same task cycle.

Method for calculating DeviatedValue

The master and slave positions depend on the selected ReferenceType. If ReferenceType is set to mcCommand, the master and slave positions are calculated based on the position references.

- When the variables Master and Slave are specified as linear axis

  DeviatedValue = Master position - Slave position.

- When the variables Master and Slave are specified as rotary axis with consistent rotation cycle.

  The DeviatedValue is the shorter distance between the Master position and Slave position in the rotation range.

  If Master position is ≥ Slave position, the sign of DeviatedValue is “+”.

  If Master position is < Slave position, the sign of DeviatedValue is “-”.

- When the variables Master and Slave are specified as rotary axis with inconsistent rotation cycles.

  DeviatedValue = Master position - Slave position.

- When the variables Master and Slave are specified as different axes

  DeviatedValue = Master position - Slave position.

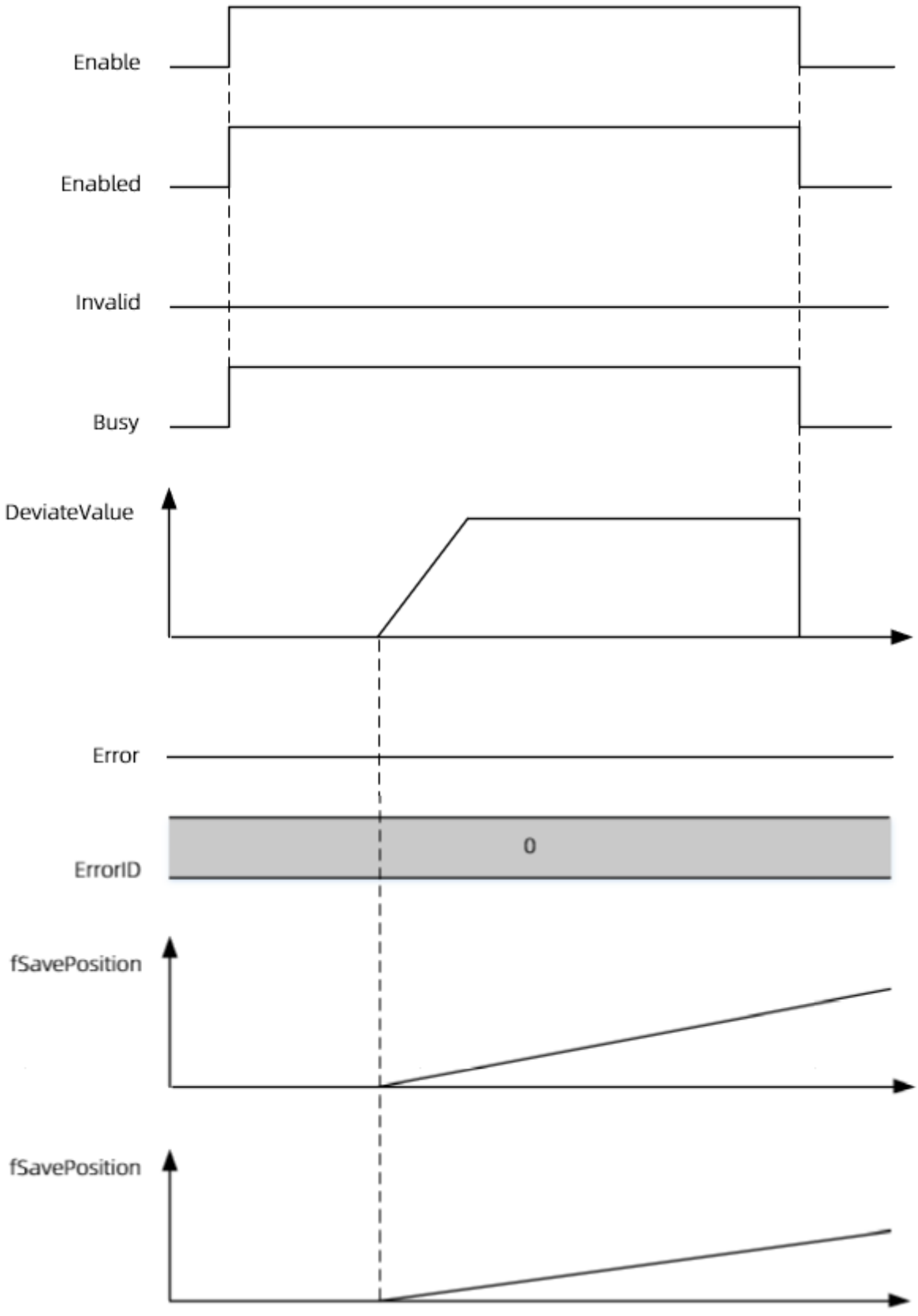
- **Precautions**
  - The master and slave axes specified in this instruction cannot be the same; otherwise, a fault will occur.

- The slave axis specified in this instruction must be bus-controlled axis or local pulse axis; otherwise, a fault will occur.

- **Timing diagram**

  Any fault issued by this instruction does not affect the operation of the axis or axis group.

  With an input parameter PermittedDeviation of 10.0, the output parameter Busy shows the value TRUE on a rising edge of Enable. When the output parameter Enabled shows the value TRUE, it indicates that the instruction has been enabled.

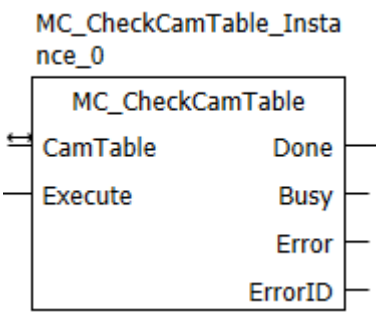


- **Faults**
  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.4.19 MC_CheckCamTable

This instruction checks the validity of user-defined cam table node data.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_CheckCamTable	Check cam table node data		MC_CheckCamTable_Instance_0(CamTable :=, Execute :=, Done =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
CamTable	Cam table	18.3.3.2.3.1 MC_CAM_TABLE_REF: Cam table structure	-	-	Specify the cam table to be checked.

  Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**
  - If the input cam table data corresponds to a cam table newly created by the system, the instruction will be directly done.
  - This instruction is designed for checking the validity of user-defined cam table node data. If this instruction is not called to check the data validity, a fault will be reported by the MC_GenerateCamTable, MC_CamIn, and MC_CamInV2 instructions.
- **Example**

The following describes how to crease a user-defined cam table.

1. First, declare the required variables in the variable declaration block of the POU:
   - Declare CamTest: User-defined cam table; Type: MC_CAM_TABLE_REF.

     This variable consists of the following members:

     You can specify the node type through iNodeType, for which the initial value is set to "1" in the preceding figure, indicating the key node. You can specify the number of nodes as required through

iNodeNumber, which is set to 5 in the figure. pNodePtr indicates the address of the variable for the cam node data, so you must declare a variable of MC_KEY_NODE_REF and enter the address here.

◦ Declare NodesTest: Cam node data; Type: MC_CAM_KEY_NODE_REF. Enter the address of NodesTest as the initial value of CamTest. pNodePtr.

2. Write cam data to the nodes of NodesTest by manual entry or programming in the programming area. Ignore the type of the curve to the first cam node.

The above is the process of creating a user-defined cam table. When the MC_CamIn instruction is executed with such a cam table, it must check the cam table validity first.

3. Use MC_CheckCamTable to check the validity of the cam table.

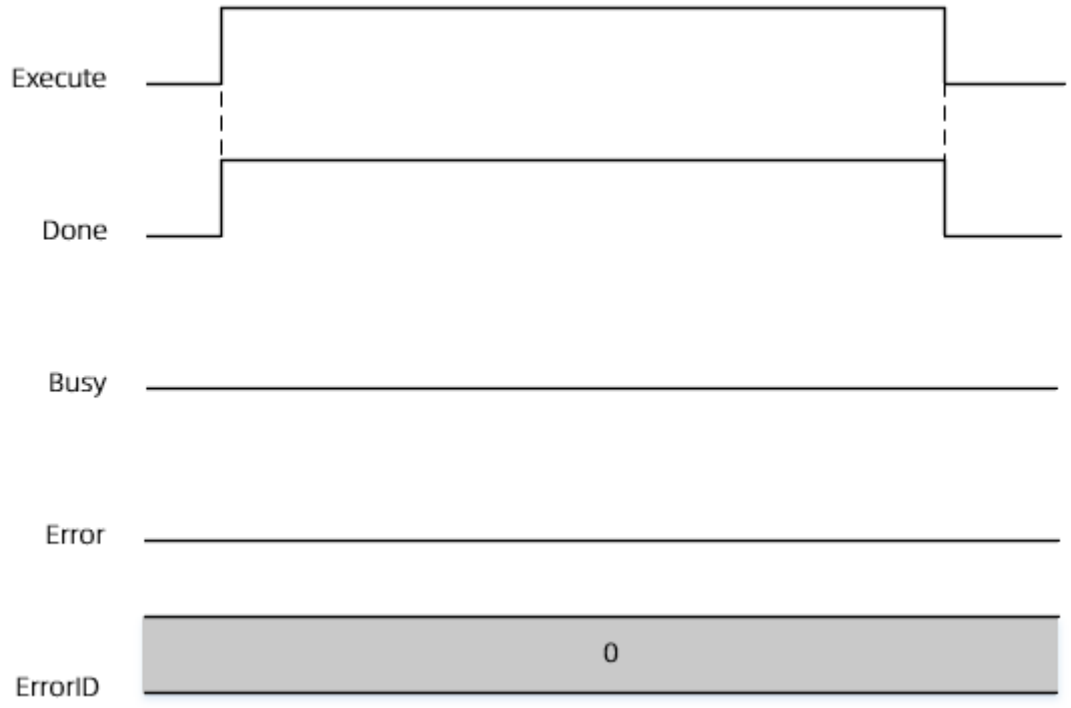
4. Execute the MC_CamIn instruction.

5. During the execution of the instruction, you can use the MC_GenerateCamTable instruction to update the cam node data. Input the node data to be replaced through CamNodes, and input the number of nodes to be replaced through NodeNum.

- **Precautions**

  The number of nodes (iNodeNumber) in a user-defined cam table must be less than the length of the key node array bound to pNodePtr; otherwise, a system fault will occur.

- **Timing diagram**



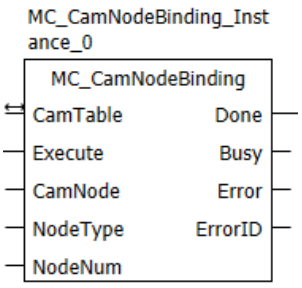
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.4.20 MC_CamNodeBinding

This instruction binds user-defined cam nodes to a user-defined cam table.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_CamNodeBinding	Cam node binding		MC_CamNodeBinding_Instance_0(CamTable :=, Execute :=, CamNode :=, NodeType :=, NodeNum :=, Done =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
CamTable	User-defined cam table	18.3.3.2.3.1 MC_CAM_TABLE_REF: Cam table structure	-	-	Cam table to which the nodes will be bound.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Enable	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.
CamNode	Cam node	ARRAY[*] OF 18.3.3.2.3.3 MC_CAM_KEY_NODE_REF: Key node structure ARRAY[*] OF 18.3.3.2.3.4 MC_CAM_FITTING_NODE_REF: Dense node structure	-	-	Array of cam nodes to be bound.

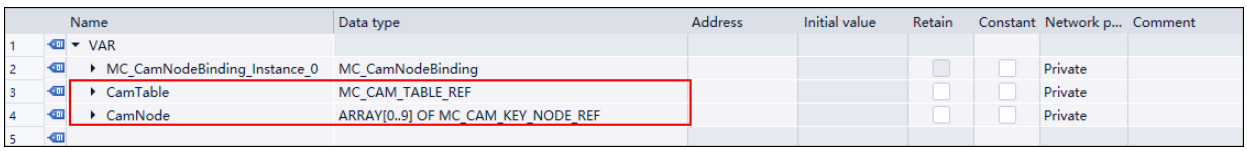
Input Variable	Name	Data Type	Valid Range	Default Value	Description
NodeType	Node type	INT	1, 2	0	Cam table node type to be associated: 1: Key node 2: Dense node
NodeNum	Node number	UDINT	2 to 65535	0	Number of key nodes: 2 to 20000. Number of dense nodes: 2 to 65535.

Output variables

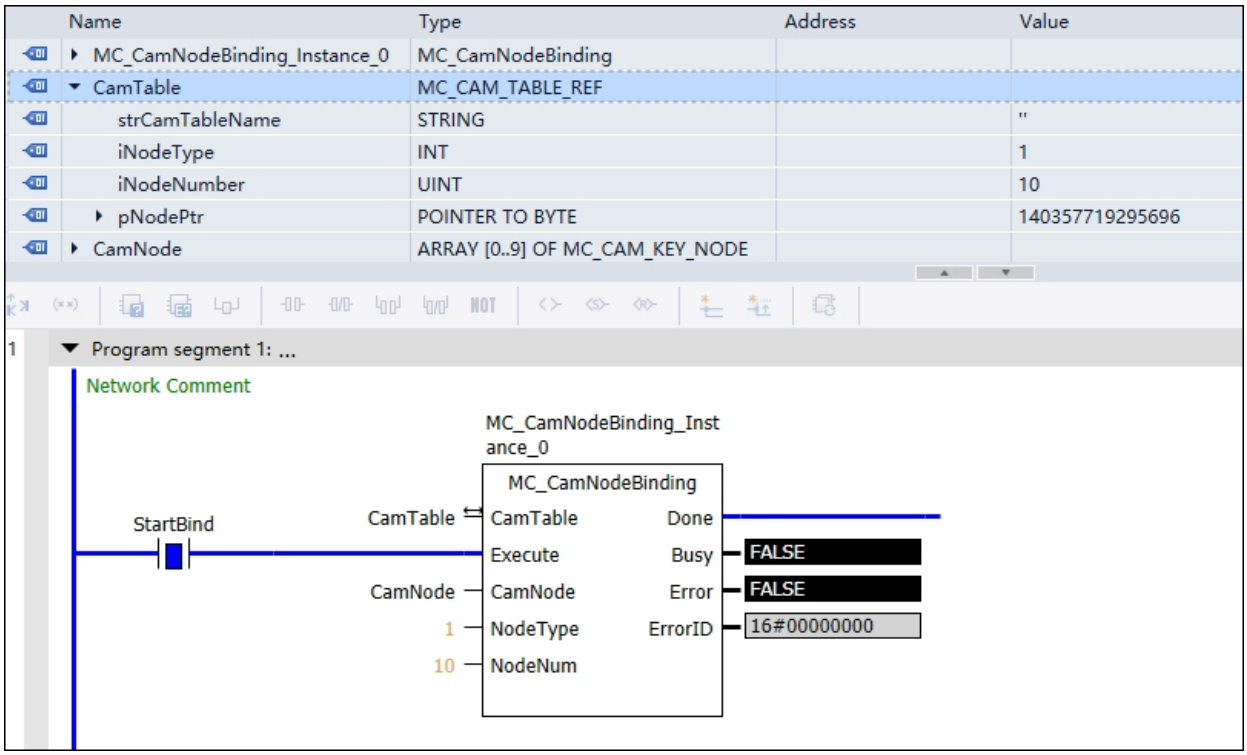
Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**
  - This instruction binds user-defined cam nodes to a cam table. After binding, it initializes the cam table node type, node number, and other information.
  - This instruction starts the cam parameter check, including whether the cam type matches the specified cam node array type, the cam node type, and whether the number of cam nodes is within the specified range.
  - This instruction initializes the maximum number of nodes in the cam table. After the cam table is bound with this instruction, if the number of cam nodes exceeds the specified number when calling the MC_GenerateCamTable instruction to update the number of cam table nodes, the MC_GenerateCamTable instruction will report fault 0FB5.
- **Precautions**
  - Do not execute the MC_CamNodeBinding instruction during the execution of MC_GenerateCamTable, MC_CamIn, or MC_CamInV2; otherwise, a fault may occur.
  - The MC_CamNodeBinding instruction is mainly used for user-defined cam tables. When a system-created cam table is used, do not use this instruction to bind the cam node array.
- **Example**

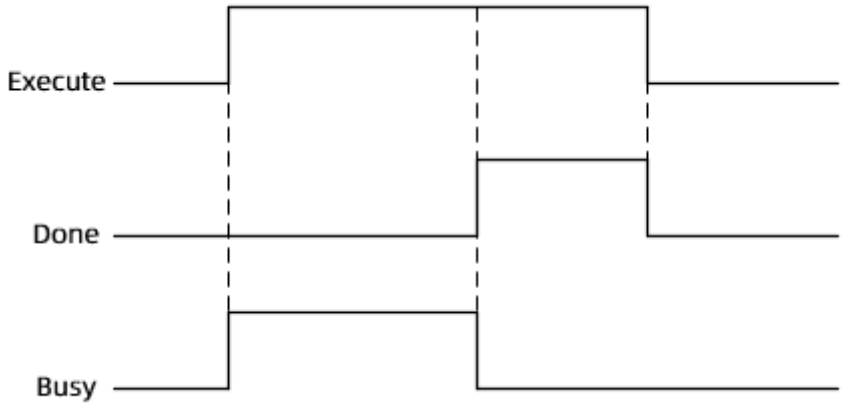
  First, declare a user-defined cam table and specify a cam node array as key node.



Then call the function block (FB) and specify the cam node type as key node. The instruction is triggered when the number of nodes matches the number of cam nodes.



- **Timing diagram**



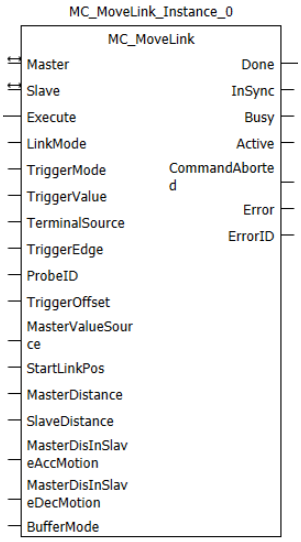
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.4.21 MC_MoveLink

This instruction is used for positioning the slave axis in sync with the master axis with specified conditions for triggering synchronized motion between the master and slave axes.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_MoveLink	Synchronous positioning		MC_MoveLink_Instance_0(Master :=, Slave :=, Execute :=, LinkMode :=, TriggerMode :=, TriggerValue :=, TerminalSource :=, TriggerEdge :=, ProbeID :=, TriggerOffset :=, MasterValueSource :=, StartLinkPos :=, MasterDistance :=, SlaveDistance :=, MasterDisInSlaveAccMotion :=, MasterDisInSlaveDecMotion :=, BufferMode :=, Done =>, InSync =>, Busy =>, Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Master	Master axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the master axis; an instance of AXIS_REF.

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Slave	Slave axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the slave axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.
LinkMode	Synchronization start condition	INT	0, 1, 2	0	0: When the instruction is executed. 1: When the master axis reaches the position specified by StartLinkPos. 2: When the TriggerValue is detected to be TRUE.
TriggerMode	Trigger mode	INT	0, 1	0	0: Drive mode. 1: Controller mode. (This parameter is valid only when the LinkMode is set to 2)
TriggerValue	Trigger switch in controller mode	BOOL	TRUE, FALSE	FALSE	TRUE: Triggered. FALSE: Untriggered. (This parameter is valid only when the TriggerMode is set to 1)
TerminalSource	Touch probe signal source	INT	0, 1	0	0: DI terminal. 1: Encoder Z-phase signal. (Used when the TriggerMode is specified as Drive mode)
TriggerEdge	Trigger edge	INT	0, 1	0	0: Rising edge. 1: Falling edge.
ProbeID	Touch probe ID	INT	0, 1	0	0: Touch probe 1. 1: Touch probe 2. (Used when the TriggerMode is specified as Drive mode)

Input Variable	Name	Data Type	Valid Range	Default Value	Description
TriggerOffset	Master axis synchronization offset	LREAL	-	0	Describe the offset between the actual synchronized position of the master and slave axes and the trigger position. Position of the master axis when the slave axis starts the synchronized motion: Latched trigger position + TriggerOffset. (This parameter is valid only when the LinkMode is set to 2)
MasterValueSource	Master axis data type	18.3.3.2.2.11 MC_REFERENCE_TYPE: Data Type Enumeration	0, 1, 2	0	0: mcCommand: Value calculated in the previous task cycle (fSavePosition). 1: mcFeedback: Value obtained in the same task cycle (fActPosition). 2: mcLaterCommand: Value calculated in the same task cycle (fSetPosition).
StartLinkPos	Master axis position at camming start	LREAL	Positive number, negative number, or 0	0	Specify the absolute position [u] of the master axis when the slave axis starts following the master axis. In Rotary mode, if the parameter value is a negative number or greater than the number of rotation cycles, it will be converted to a positive value and then taken modulo.
MasterDistance	Master axis travel distance	LREAL	Positive number or 0	0	Travel distance [u] of the master axis. If a negative value is entered, a fault will occur. The sign of MasterDistance is determined by the velocity of the master axis at the time of triggering synchronization. In Rotary Mode, the number of cycles is counted.
SlaveDistance	Slave axis travel distance	LREAL	Positive number, negative number, or 0	0	Travel distance [u] of the slave axis during the synchronization between the master and slave axes. When the input value is greater than 0, the slave axis moves in the same direction as the master axis. When the input value is less than 0, the slave axis moves in the opposite direction of the master axis. In Rotary Mode, the number of cycles is counted.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
MasterDisInSlaveAccMotion	Master axis travel distance during slave axis acceleration	LREAL	Positive number or 0	0	If a negative value is entered, a fault will occur. The sign of MasterDisInSlaveAccMotion is determined by the velocity of the master axis at the time of triggering synchronization.
MasterDisInSlaveDecMotion	Master axis travel distance during slave axis deceleration	LREAL	Positive number or 0	0	If a negative value is entered, a fault will occur. The sign of MasterDisInSlaveDecMotion is determined by the velocity of the master axis at the time of triggering synchronization.
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0, 1	0	mcAborting = 0: Aborting. mcBuffered = 1: Buffered.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
InSync	Synchronizing	BOOL	TRUE, FALSE	FALSE	TRUE when the master and slave axes are synchronizing.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Controlling	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Error	BOOL	TRUE, FALSE	FALSE	TRUE when an error occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**
  - This instruction is used for positioning the slave axis in sync with the master axis with specified conditions for triggering synchronized motion between the master and slave axes.
  - During the execution of this instruction, you cannot execute the MC_SetOverride instruction on the slave axis.
  - During the execution of this instruction, no checks are performed to see if the acceleration of the slave axis exceeds the limit; however, a fault will occur if the velocity of the slave axis exceeds the software limit.
  - During the execution of this instruction, the MC_MoveSuperImposed instruction can be used.
  - When the instruction is triggered on a rising edge of the input parameter Execute, the input parameters are checked for validity. If any of invalid parameters is detected, it will be automatically corrected by the function block without a fault message; any buffered instructions after this instruction will be cleared.
  - If bus cycle is specified for the cycle task of the master axis, the velocity of the slave axis may show a sudden jump after synchronization. It is recommended that users specify EtherCAT cycle for the cycle task of both the master and slave axes before enabling a synchronous control instruction.
  - When the local pulse axis is used as the master axis with Y0 or Y1 single-ended output, it is not recommended to use this instruction. ① No pulse output; ② No position feedback.
  - When the InSync is sent to TRUE, the master axis moves in the effective trigger direction. If the master axis reverses during subsequent synchronization with the slave axis following the master axis, the slave axis will stop moving when the master axis moves in reverse direction to the position specified by StartLinkPos, and the InSync will be set to FALSE. Synchronization can be re-triggered when the master axis moves in the effective trigger direction and reaches the position specified by StartLinkPos.
  - MC_Home, MC_Homing, or MC_HomingWithParameter can be used during the instruction execution.

    This instruction places the master axis in the AXIS_HOMING state. If the instruction is executed while the master axis is already in this state, the signal source "MasterValueSource" must be set to "mcLatestCommand"; otherwise, the slave axis will overspeed.
  - The arrival check function is only applicable to MotionControl library version 2.4.0.0 or later. Based on the "Positioning range" and "Positioning check time" parameters in the "General setting" of the axis configuration page, it checks in real time whether the actual position has entered the specified target

range. After the instruction output for the target position is completed, positioning is complete and the Done output is set if the difference between the actual position and the target position is smaller than the "Positioning range". If positioning is not completed within the "Positioning check time", a fault will occur.

- **Precautions**
  - Invalid parameter input values:
    - If the input value of the TriggerMode, LinkMode, ProbeID, TerminalSource, or MasterValueSource is out of the limit, a fault will occur.
    - If the input value of the MasterDistance, MasterDisInSlaveAccMotion, or MasterDisInSlaveDecMotion is negative, a fault will occur.
  - If the constant-velocity travel distance of the master axis is negative (MasterDistance - MasterDisInSlaveAccMotion - MasterDisInSlaveDecMotion < 0), a fault will occur.
  - If you want the slave axis velocity to equal the master axis velocity in the stable synchronization zone (both axes moving at constant velocity), you can set the SlaveDistance to:

$$\text{MasterDistance} + \frac{MasterDisInSlaveAccMotion}{2} + \frac{\text{MasterDisInSlaveDecMotion}}{2}$$

  - If you want the slave axis velocity to be greater than the master axis velocity in the stable synchronization zone (both axes moving at constant velocity), you can set the SlaveDistance to be greater than:

$$\text{MasterDistance} + \frac{MasterDisInSlaveAccMotion}{2} + \frac{\text{MasterDisInSlaveDecMotion}}{2}$$

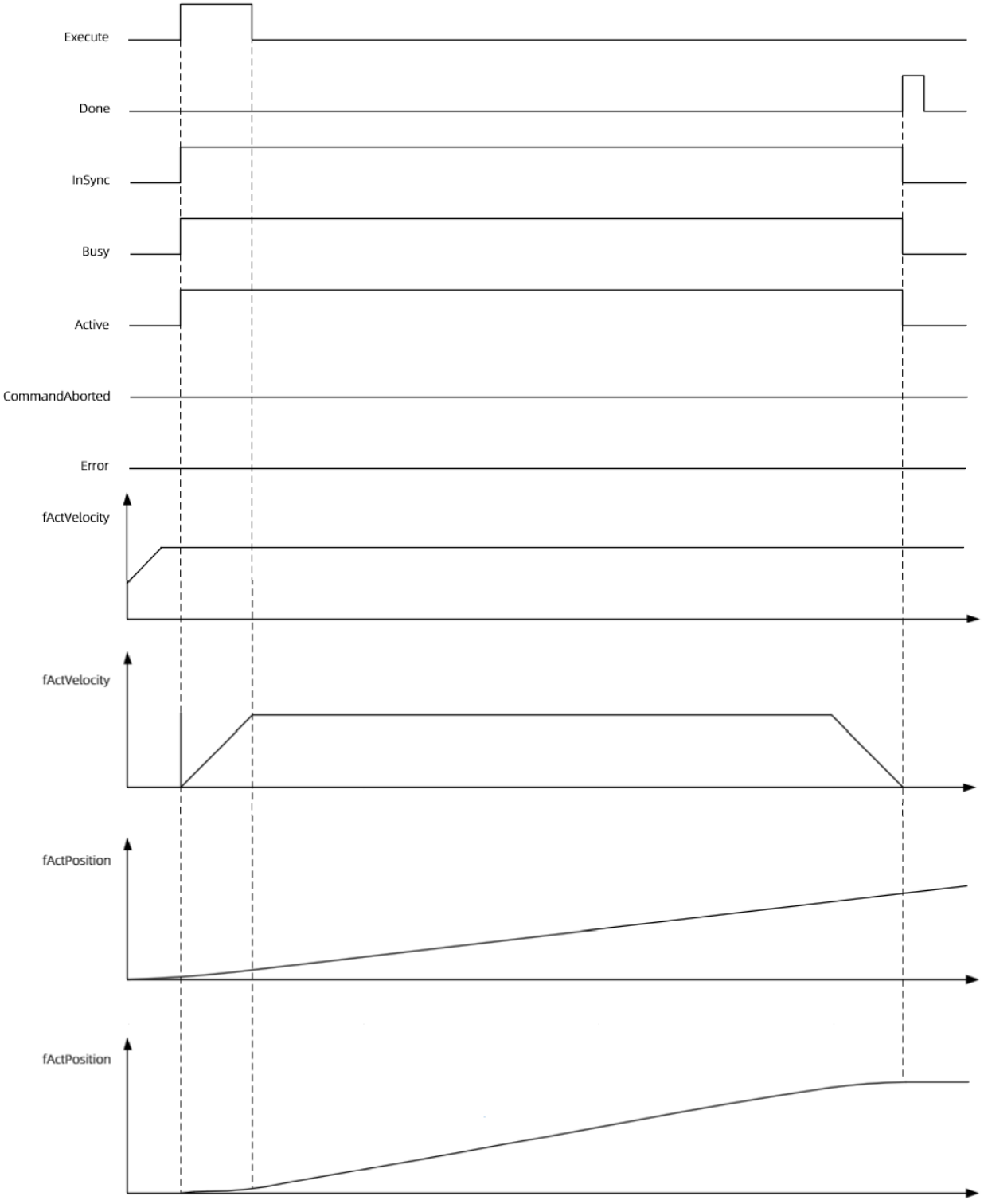
  - If you want the slave axis velocity to be less than the master axis velocity in the stable synchronization zone (both axes moving at constant velocity), you can set the SlaveDistance to be less than:

$$\text{MasterDistance} + \frac{MasterDisInSlaveAccMotion}{2} + \frac{\text{MasterDisInSlaveDecMotion}}{2}$$

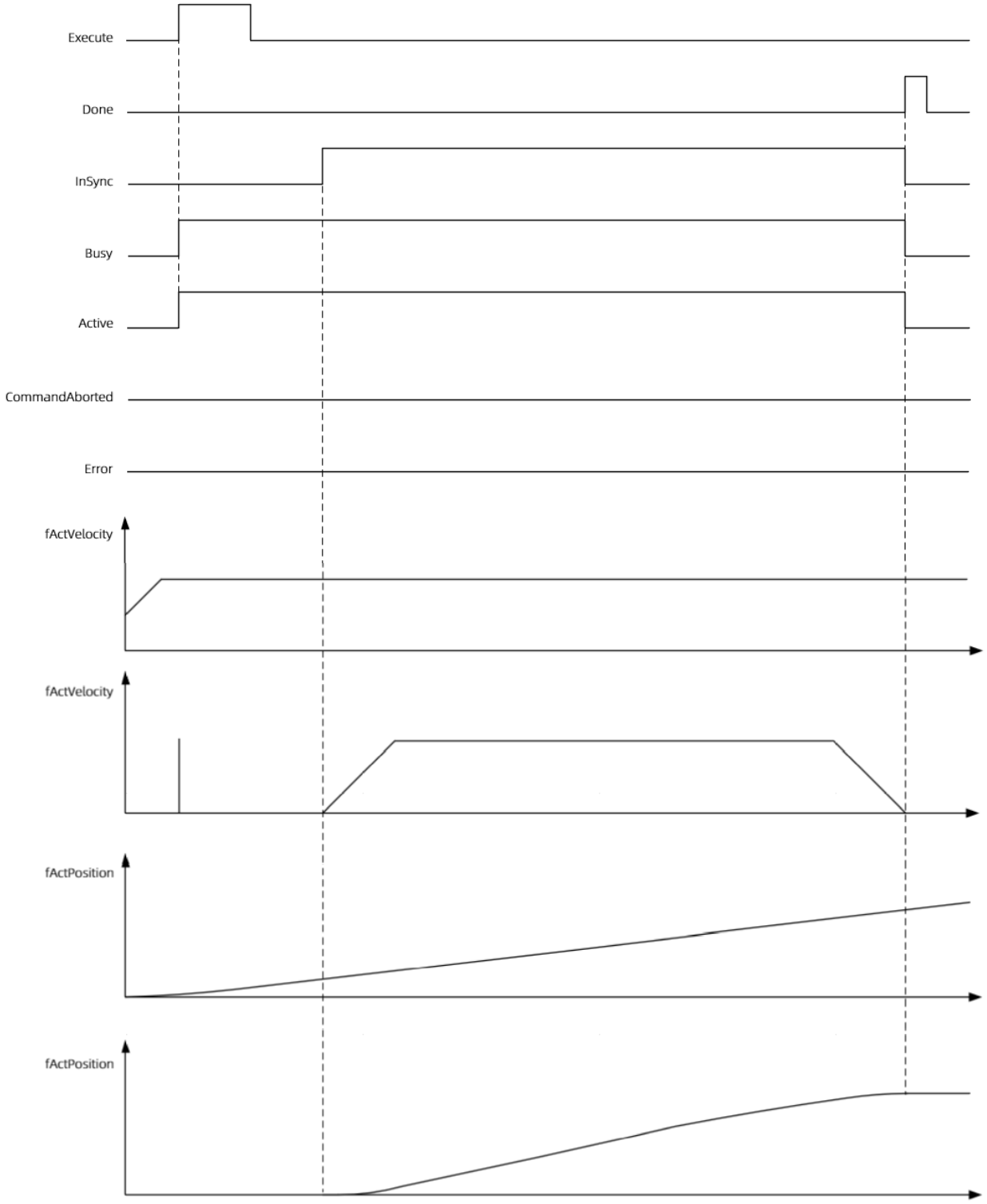
  - When the TriggerMode is set to Drive mode, the touch probe signal comes from the external touch probe signal of the master axis.
  - In Drive mode, the touch probe signal occupies the probe channel and conflicts with the MC_TocuhProbe and MC_MoveFeed instructions. To use the MC_Movelink instruction, do not use the MC_MoveFeed or MC_TouchProbe instruction with the same probe channel on the master axis.
  - When the MC_MoveLink instruction is executed with the TriggerMode being set to Drive mode, the touch probe channel may be cleared by the MC_AbortTrigger instruction, resulting in the same effect as the instruction being aborted.
  - When the TriggerMode is set to Controller mode, the instruction does not occupy the touch probe channel, and the settings of the touch probe channel are invalid.
  - The touch probe recorded position in Controller mode may have a position deviation within one task cycle.
  - There are only two touch probe channels, so a maximum of one master axis with two slave axes can be configured in Drive mode. To configure a single master axis with multiple slave axes, modify the TriggerMode to Controller mode, and then map the variable Trigger of each axis to the corresponding IO device, replacing the touch probe signal from a drive with the IO device.

- **Timing diagram**
  - Starting this instruction

    LinkMode = When the instruction is executed



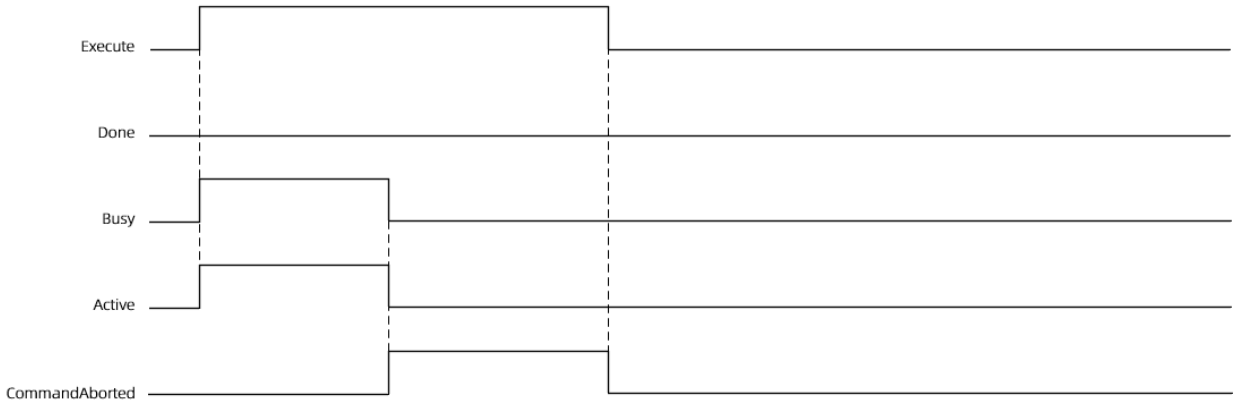
    LinkMode = When the master axis reaches the position specified by StartLinkPos



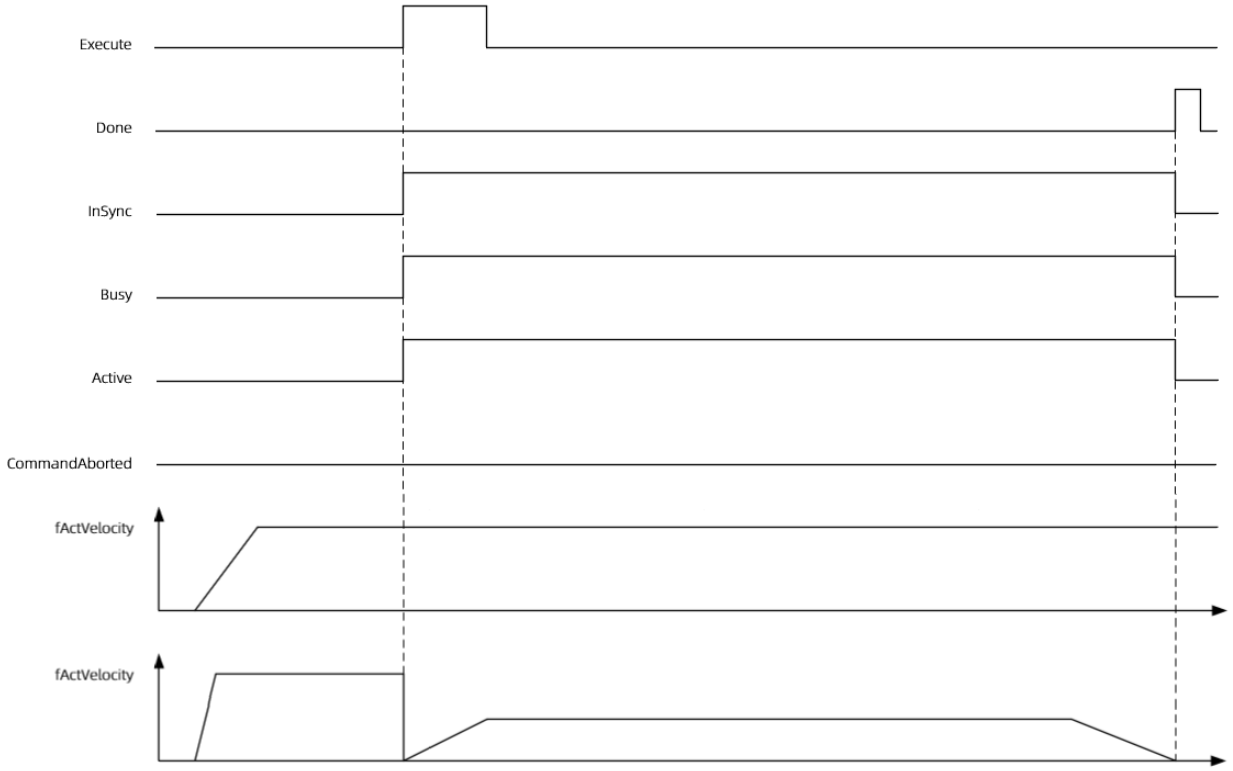
◦ When the BufferMode is set to Aborting in the previous operation

When the MC_MoveAbsolute instruction is executed in the previous operation with the start of this instruction being specified through LinkMode, the execution of this instruction is as follows.

When the previous MC_MoveAbsolute instruction is executed



When the current instruction is executed

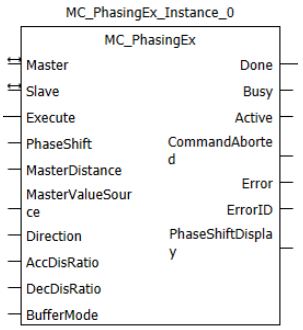


- **Faults**
  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.4.22 MC_PhasingEx

This instruction enables phase shifting completed within the specified master axis travel distance (which has an effect on the slave axis but no effect on the master axis), so that the slave axis is synchronized to the shifted master axis phase. You can specify the master axis travel distance (in percentage) corresponding to the shifted slave axis motion during the acceleration and deceleration phases.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_PhasingEx	Complete phase shifting within specified master axis travel distance		MC_PhasingEx_Instance_0(Master :=, Slave :=, Execute :=, PhaseShift :=, MasterDistance :=, MasterValueSource :=, Direction :=, AccDisRatio :=, DecDisRatio :=, BufferMode :=, Done =>, Busy => , Active =>, CommandAborted =>, Error =>, ErrorID =>, PhaseShiftDisplay =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Master	Master axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the master axis; an instance of AXIS_REF.
Slave	Slave axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the slave axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	The function block is executed on a rising edge of this input variable.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
PhaseShift	Phase shift	LREAL	Positive number, negative number, or 0	0	Phase shift [u] between the master and slave axes.
MasterDistance	Master axis travel distance	LREAL	Positive number	0	Travel distance [u] of the master axis during phase shifting.
MasterValueSource	Master axis data type	18.3.3.2.2.11 MC_REFERENCE_TYPE: Data Type Enumeration	0, 1, 2	0	0: mcCommand: Value calculated in the previous task cycle (fSavePosition). 1: mcFeedback: Value obtained in the same task cycle (fActPosition). 2: mcLaterCommand: Value calculated in the same task cycle (fSetPosition).
Direction	Direction	18.3.3.2.2.9 MC_DIRECTION: Motion Direction Enumeration	1, 2	1	Phase shifting is executed on the slave axis when the master axis moves in the specified direction. 1: DIR_POSITIVE: Positive direction. 2: DIR_NEGATIVE: Negative direction.
AccDisRatio	Acceleration distance ratio	LREAL	0 to 1	0	Ratio of the master axis travel distance to the specified value of MasterDistance during the change process from the compensated velocity at the instruction triggering to the target velocity (calculated based on the input conditions) on the phase shifting curve.
DecDisRatio	Deceleration distance ratio	LREAL	0 to 1	0	Ratio of the master axis travel distance to the specified value of MasterDistance during the change process from the target velocity to 0 on the phase shifting curve.
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0	0	mcAborting = 0. Other values: Reserved

Output variables

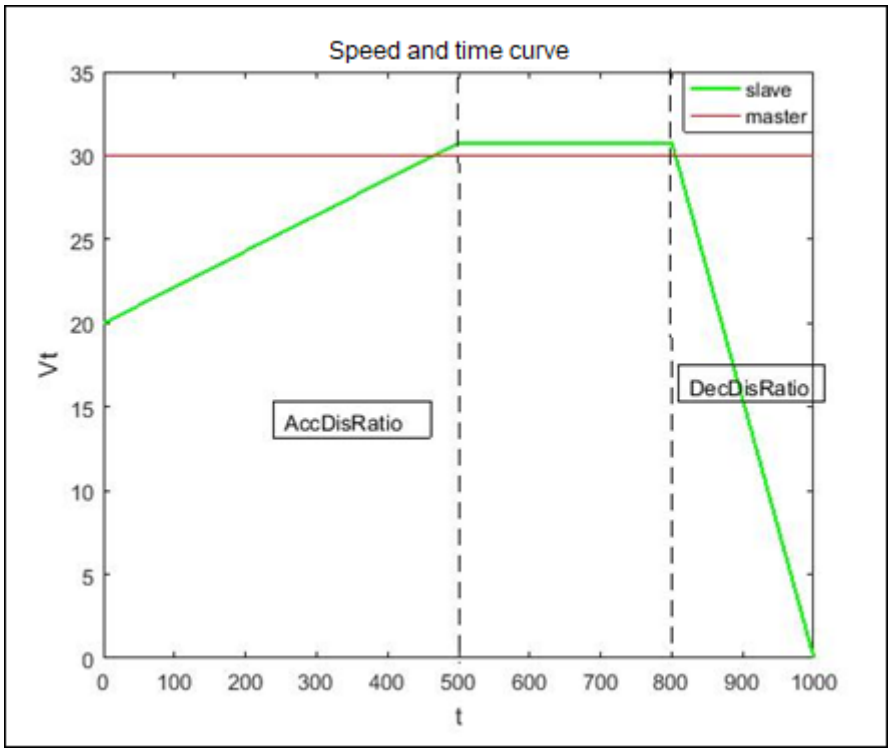
Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the PhaseShift is reached.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Controlling	BOOL	TRUE, FALSE	FALSE	TRUE when the phase shifting is started.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Error	BOOL	TRUE, FALSE	FALSE	TRUE when an error occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.
PhaseShiftDisplay	Phase shift display	LREAL	-	0	Real-time display of the master axis phase shifting value.

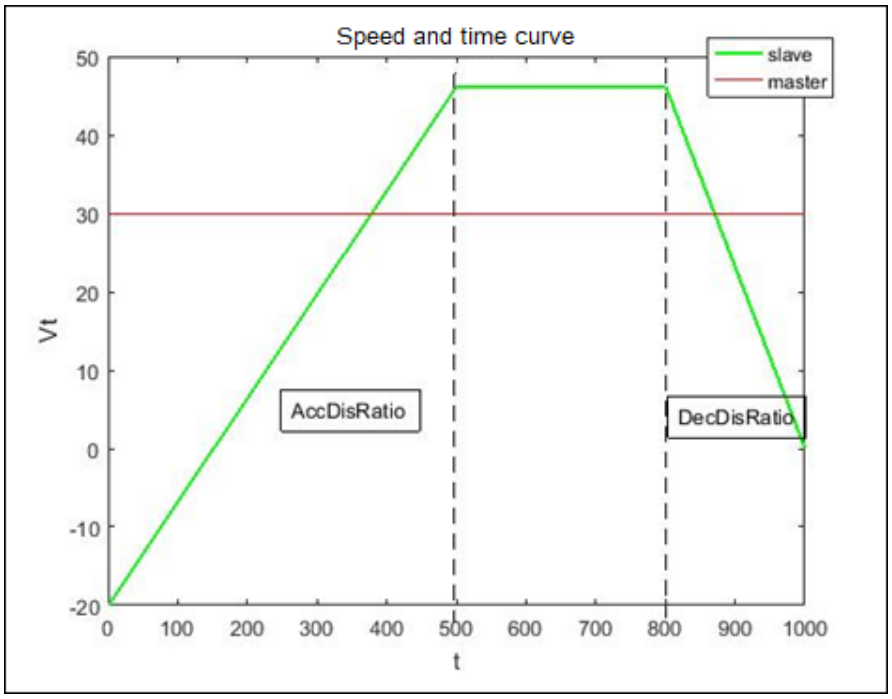
- **Function**
  - You can shift the phase of the master axis for the following synchronous control instructions: MC_GearIn, MC_CamIn, MC_GearInPos, and MC_MoveLink. A fault will occur if this function block is executed in non-synchronous control.
  - This instruction can be re-executed. However, multiple MC_PhasingEx instructions cannot be executed simultaneously. When an MC_PhasingEx instruction is being executed, it is prohibited to trigger a new MC_PhasingEx instruction; otherwise, the subsequently triggered instruction reports an error.
  - The master and slave axes cannot be specified as the same axis.
  - The instruction will issue a fault message if it is executed with the slave axis in non-synchronized motion.
  - Phase shifting start time:
    1. MC_GearIn: After the input variable InGear is set.
    2. MC_GearInPos: After the input variable InSync is set.
    3. MC_CamIn: After the input variable InSync is set.
    4. MC_MoveLink: After the input variable InSync is set.
  - Faults for parameter check:
    1. A fault occurs when the input parameter MasterDistance is set to be less than or equal to 0.

2. A fault occurs when the input parameter AccDisRatio or DecDisRatio is set to be less than 0 or greater than 1.
3. A fault occurs when the sum of the input parameters AccDisRatio and DecDisRatio is greater than 1.
4. A fault occurs when the slave axis is specified as a non-supported axis type. The supported axis types include ECT control axis and local control axis.
5. A fault occurs when any other input parameters are outside the limits.

- When the input parameter AccDisRatio is set to 0, the phase shifting curve contains no acceleration phase, which means that the phase shifting velocity jumps to the calculated target velocity immediately, resulting in significant velocity fluctuation and increased impact on the slave axis. When the input parameter DecDisRatio is set to zero, the phase shifting curve contains no deceleration phase.
- When the input parameter PhaseShift is set to zero, the instruction ends normally without affecting the synchronization, and the Done is set normally.
- This instruction is executed when the master axis moves in the direction specified by Direction. During shifting, if the master axis changes its direction, the instruction is aborted. When the master axis moves in the specified direction again and reaches the reversal point, phase shifting restarts for the slave axis to complete the remaining amount of phase shift.
- If the instruction is executed with a master axis velocity equal to zero, phase shifting will be started until the master axis velocity changes to non-zero.
- AccDisRatio and DecDisRatio: (The compensated velocity curve of the slave axis is in green, and the velocity curve of the master axis is in red)



(a) The initial compensated velocity has the same sign as the phase shift amount



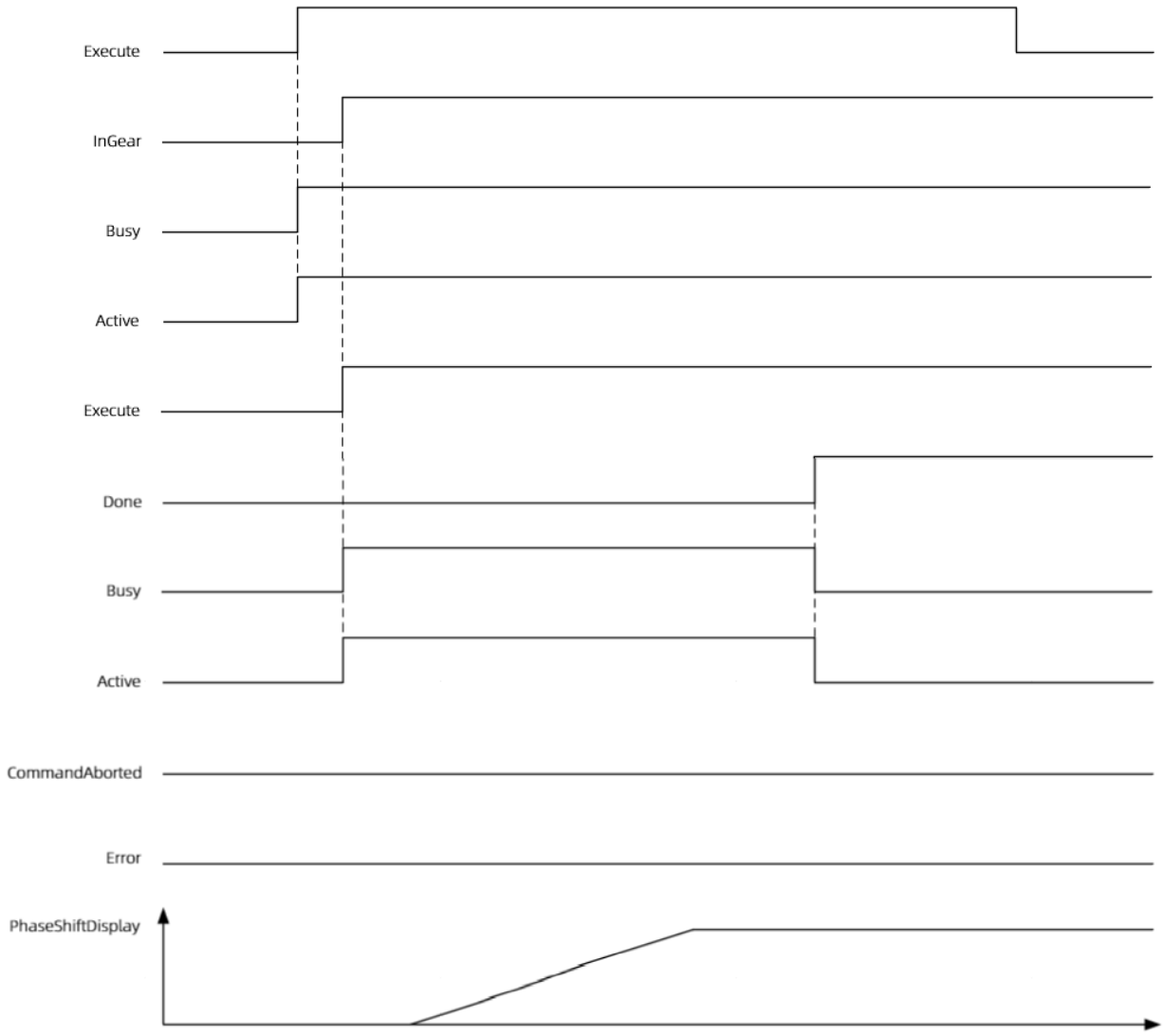
(b) The initial compensated velocity has a different sign from the phase shift amount

- **Precautions**
  It is not recommended to execute two same instances of MC_PhasingEx simultaneously.
- **Timing diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates the instruction has been executed successfully; the parameter Busy indicates that the current function block is being executed; the parameter Active indicates the axis is being controlled; the parameter CommandAborted indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

  For the MC_GearIn instruction, the parameter InGear being set is the rising edge for the parameter Execute of this instruction, at which point the output parameter Busy shows the value TRUE and the Active shows the value TRUE in the next cycle.

  When the input variable PhaseShift is set to 40.0 and the output variable PhaseShiftDisplay shows the value specified by PhaseShift, the output parameter Done shows the value TRUE.



- **Faults**
  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5 Axis Group

18.3.3.5.1 Instruction List

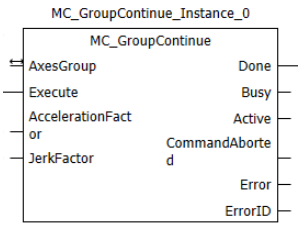
Instruction	Function	Library	Applicable Library Version
18.3.3.5.2 MC_GroupContinue	Continue axis group motion	MotionControl	Library version 1.1.0.0 and later
18.3.3.5.3 MC_GroupDisable	Disable axis group		Library version 1.1.0.0 and later
18.3.3.5.4 MC_GroupEnable	Enable axis group		Library version 1.1.0.0 and later
18.3.3.5.5 MC_GroupHalt	Stop axis group		Library version 1.1.0.0 and later
18.3.3.5.6 MC_GroupHome	Axis group homing		Library version 1.1.0.0 and later

Instruction	Function	Library	Applicable Library Version
18.3.3.5.7 MC_GroupInterrupt	Axis group pause		Library version 1.1.0.0 and later
18.3.3.5.8 MC_GroupReadActualAcceleration	Read axis group actual acceleration		Library version 1.1.0.0 and later
18.3.3.5.9 MC_GroupReadActualPosition	Read axis group actual position		Library version 1.1.0.0 and later
18.3.3.5.10 MC_GroupReadActualVelocity	Read axis group actual velocity		Library version 1.1.0.0 and later
18.3.3.5.11 MC_GroupReadError	Axis group error reading		Library version 1.1.0.0 and later
18.3.3.5.12 MC_GroupReadStatus	Axis group status reading		Library version 1.1.0.0 and later
18.3.3.5.13 MC_GroupReset	Reset axis group		Library version 1.1.0.0 and later
18.3.3.5.14 MC_GroupSetOverride	Axis group override setting		Library version 1.1.0.0 and later
18.3.3.5.15 MC_GroupSetPosition	Set axis group current position		Library version 1.1.0.0 and later
18.3.3.5.16 MC_GroupStop	Axis group emergency stop		Library version 1.1.0.0 and later
18.3.3.5.17 MC_MoveCircularAbsolute	Circular absolute motion		Library version 1.1.0.0 and later
18.3.3.5.18 MC_MoveCircularRelative	Circular relative motion		Library version 1.1.0.0 and later
18.3.3.5.19 MC_MoveDirectAbsolute	Point-to-point absolute motion		Library version 1.1.0.0 and later
18.3.3.5.20 MC_MoveDirectRelative	Point-to-point relative motion		Library version 1.1.0.0 and later
18.3.3.5.21 MC_MoveLinearAbsolute	Linear absolute motion		Library version 1.1.0.0 and later
18.3.3.5.22 MC_MoveLinearRelative	Linear relative motion		Library version 1.1.0.0 and later
18.3.3.5.23 MC_SetCartesianTransform	Set transformation between part coordinate system and machine coordinate system		Library version 1.1.0.0 and later
18.3.3.5.24 MC_TrackConveyorBelt	Conveyor belt tracking		Library version 1.1.0.0 and later
18.3.3.5.25 MC_GroupPower	Enable/disable axes in axis group		Library version 2.3.0.0 and later

18.3.3.5.2 MC_GroupContinue

This instruction resumes the axis group, allowing a paused or pausing motion to continue along its original trajectory.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GroupContinue	Continue axis group motion		MC_GroupContinue_Instance_0(AxesGroup := , Execute := , AccelerationFactor := , JerkFactor := , Done => , Busy => , Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
AccelerationFactor	Acceleration factor	LREAL	[0.01,1.0]	0.3	Factor relative to the maximum acceleration of the axis group. If the factor is greater than 1, it will be corrected to 1. If the factor is less than 0.01, it will be corrected to 0.3.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
JerkFactor	Jerk factor	LREAL	[0.0,1.0]	0.3	Factor relative to the maximum jerk of the axis group. For the non-trapezoidal motion profiling, the factor will be corrected to 1 if it is greater than 1. If it is less than or equal to 0, it will be corrected to 0.3.

Output variable

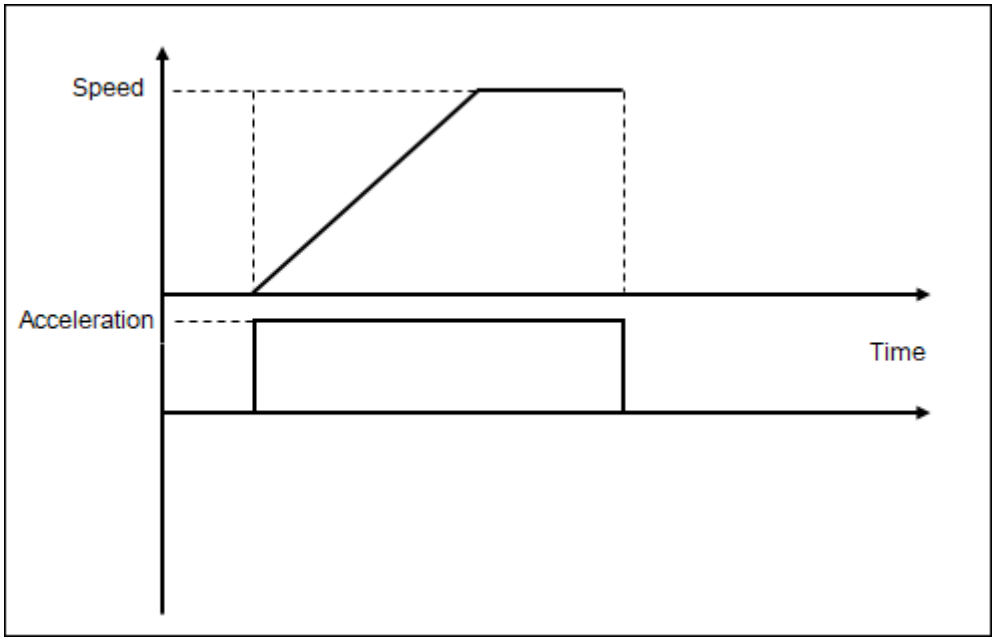
Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Instruction has controlled the axis group	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

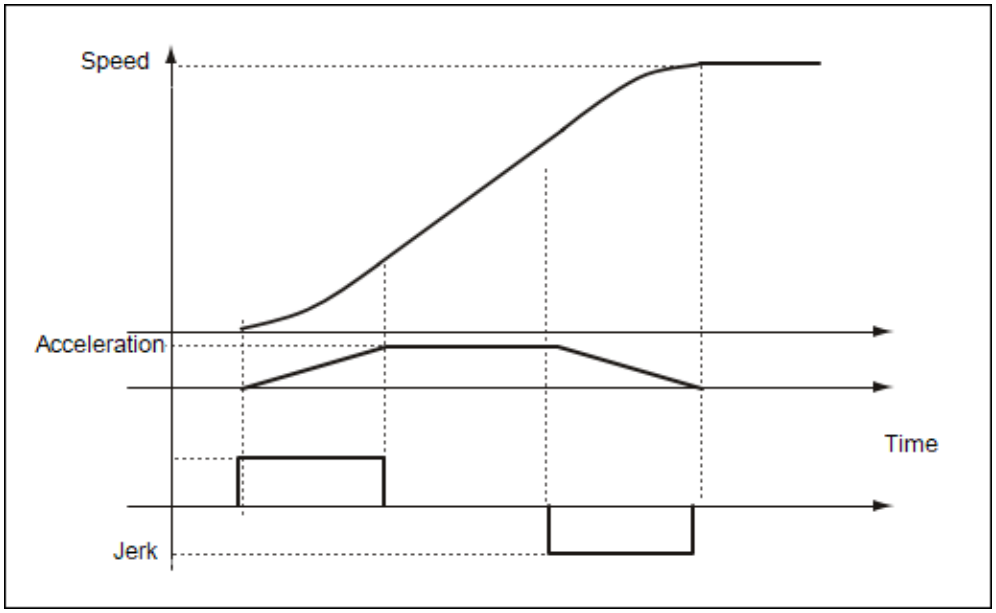
Input Variable	Name	Correction Description for Values Greater Than 1 or Backend Limit	Correction Description for Parameters Below Minimum Value Range	Correction Description for Parameters Equal to Minimum Value Range
AccelerationFactor	Acceleration factor	If the acceleration factor is greater than 1, it is automatically corrected to 1.	If the acceleration factor is less than 0.01, it is automatically corrected to 0.3.	If the acceleration factor is equal to 0.01, no correction is applied.
JerkFactor	Jerk factor	If the jerk factor is greater than 1 in the non-trapezoidal motion profiling, it is automatically corrected to 1.	If the jerk factor is less than 0 in the non-trapezoidal motion profiling, it is automatically corrected to 0.3 for operation.	If the jerk factor is equal to 0 in the non-trapezoidal motion profiling, it is automatically corrected to 0.3 for operation.

- **Function**
  - This instruction is executed on the rising edge of Execute and is used to restart AxesGroup, allowing the currently paused or pausing motion to return to its original path.
  - After this instruction starts and both instructions Busy and Active are activated, Active of the axis group motion instructions previously paused by the MC_GroupInterrupt instruction will become TRUE again. The axis group will accelerate according to the instruction parameters until it returns to the speed at which it was paused, set the Done instruction to TRUE.
  - This instruction is used in conjunction with the MC_GroupInterrupt instruction.
  - This instruction can be interrupted by MC_GroupInterrupt, MC_GroupHalt, MC_GroupStop, and other MC_GroupContinue instructions, or it can be interrupted by new motion instructions for AxesGroup.
  - **AccelerationFactor, JerkFactor**
    - The acceleration and jerk during the instruction execution are determined by the input variables AccelerationFactor and JerkFactor.
    - In joint space (ACS), the restart operation is performed based on the product of the maximum axis acceleration limited by software for each Axis variable in AxesGroup and AccelerationFactor, and the product of the maximum axis jerk limited by software for each axis and JerkFactor.
    - In work space (PCS), the restart operation is performed based on the product of the maximum acceleration of AxesGroup and AccelerationFactor, and the product of the maximum jerk of the axis group and JerkFactor.
    - JerkFactor

      Velocity and acceleration curves with JerkFactor = 0



Velocity, acceleration, and jerk curves with JerkFactor ≠ 0



- **Re-triggering the instruction**
  - This instruction cannot be re-triggered.
- **Precautions**
  - When AxesGroup is in the GroupStandby, GroupErrorstop, GroupDisabled, GroupHoming, or GroupStopping state, this instruction cannot be started. Otherwise, the instruction will return an error.
  - When AxesGroup is controlled by the MC_GroupHalt instruction and the axis group is in the homing process, this instruction cannot be started. Otherwise, the instruction will return an error.
  - The trajectory planning curve type of AxesGroup must be set as required. If the planning curve type is set to a trapezoidal curve, the input variable JerkFactor becomes invalid, and the instruction performs the motion resumption based on AccelerationFactor. If the planning curve type is set to a cubic curve or a septic curve, the instruction performs the motion resumption based on both AccelerationFactor and JerkFactor.
  - Whether the planning curve type of each Axis variable in AxesGroup is set to a trapezoidal curve or an S-curve does not affect the planning method of AxesGroup.

- During the execution of the instruction, all axes in AxesGroup must be in the enabled status. Otherwise, an error will be reported.

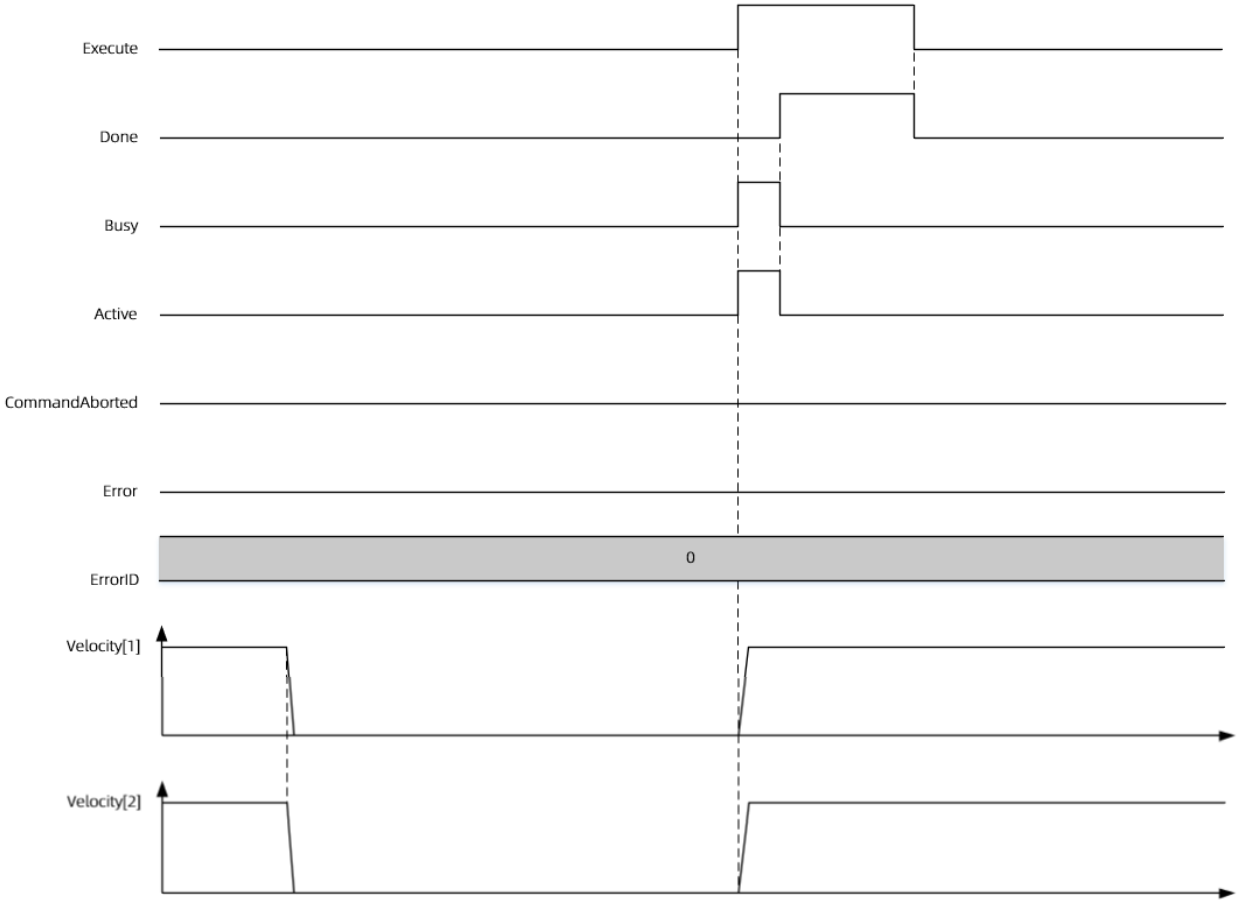
- **Sequence diagram**

  The Execute input of the function block must be triggered on a rising edge. The Done output of the function block indicates the instruction is executed successfully. The Busy output of the function block indicates that the current function block is being executed. The CommandAborted output of the function block indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

  Call the motion control instruction to put the axis group in the GROUP_MOVING status, and then the axis group is controlled by the MC_GroupInterrupt instruction, all axes in the axis group reach a speed of "0".

  After this instruction is called, the output parameter Busy is set to TRUE on a rising edge of Execute, and "Active" also becomes TRUE.

  The axis group accelerates according to the instruction parameters until it returns to the speed under the paused status, and the output parameter "Done" is set to TRUE.



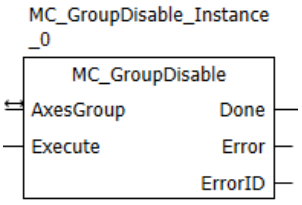
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.3 MC_GroupDisable

This instruction disables the axis group, changing its state to GroupDisabled.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GroupDisable	Disable axis group		MC_GroupDisable_Instance_0(AxesGroup := , Execute := , Done => , Error => , ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

  Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.

  Output variable

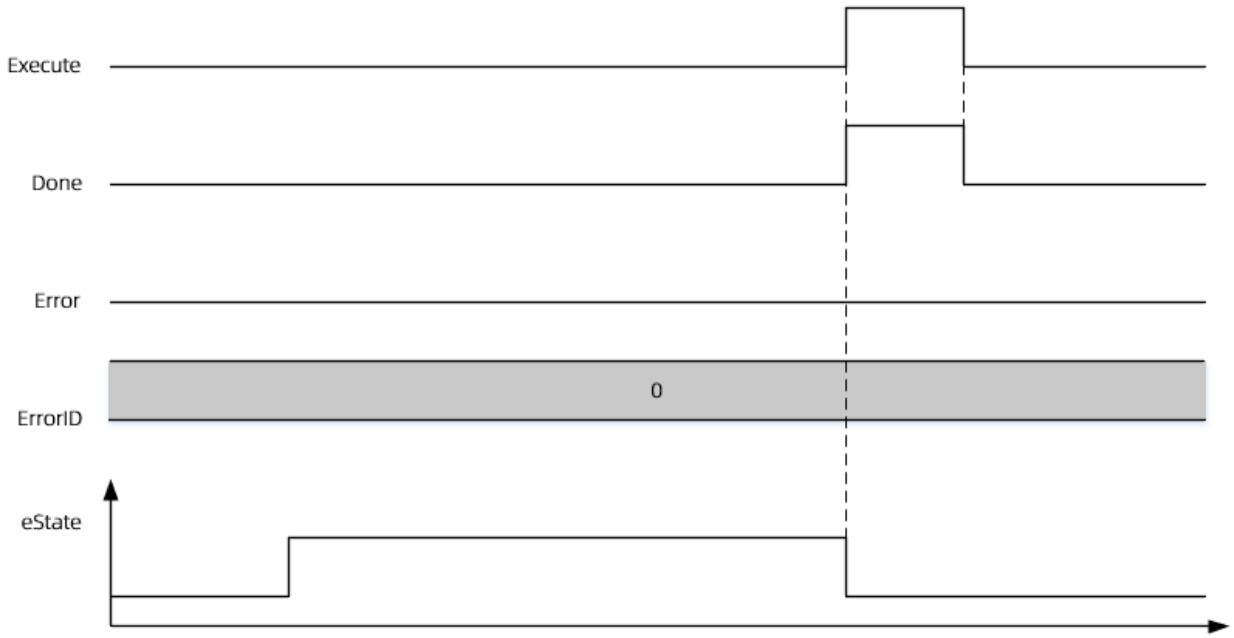
Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is done.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**
  - This instruction can disable the axis group specified by AxesGroup, changing its axis group status to GroupDisabled.

- When AxesGroup is in the GroupDisabled state, enable this instruction to directly set the output variable Done to TRUE.
- When AxesGroup is in the GroupDisabled state, the single axis instruction can be executed on the Axis variable in AxesGroup.
- This instruction may change the state of the Axis variable in the axis group specified by AxesGroup. When the output variable Done of this instruction becomes TRUE, if the Axis variable in AxesGroup is already in the ErrorStop or Disabled state, its state remains unchanged. Otherwise, if the axis is actually enabled, the state of the Axis variable will be changed to StandStill. If the axis is not yet enabled, the Axis variable is in the Disabled state.

- **Precautions**
  - This instruction can be used when the axis group (AxesGroup) is in the GroupStandby, GroupDisabled, GroupStopping, or GroupErrorStop state. Do not use this instruction in any other states. Otherwise, the instruction will return an error.
  - After the instruction has been used, the single axis instruction can be executed on the axes in AxesGroup.
- **Sequence diagram**

  When the input parameter "Execute" is TRUE, the state of the AxesGroup will be changed from GroupStandby to GroupDisabled. The output parameter "Done" becomes TRUE.



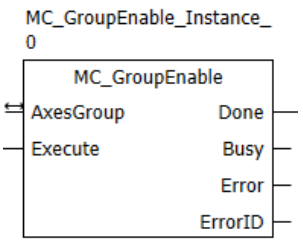
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.4 MC_GroupEnable

The instruction can enable the axis group, transitioning its state from GroupDisabled to GroupStandby.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GroupEnable	Enable axis group		MC_GroupEnable_Instance_0(AxesGroup := , Execute := , Done =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.

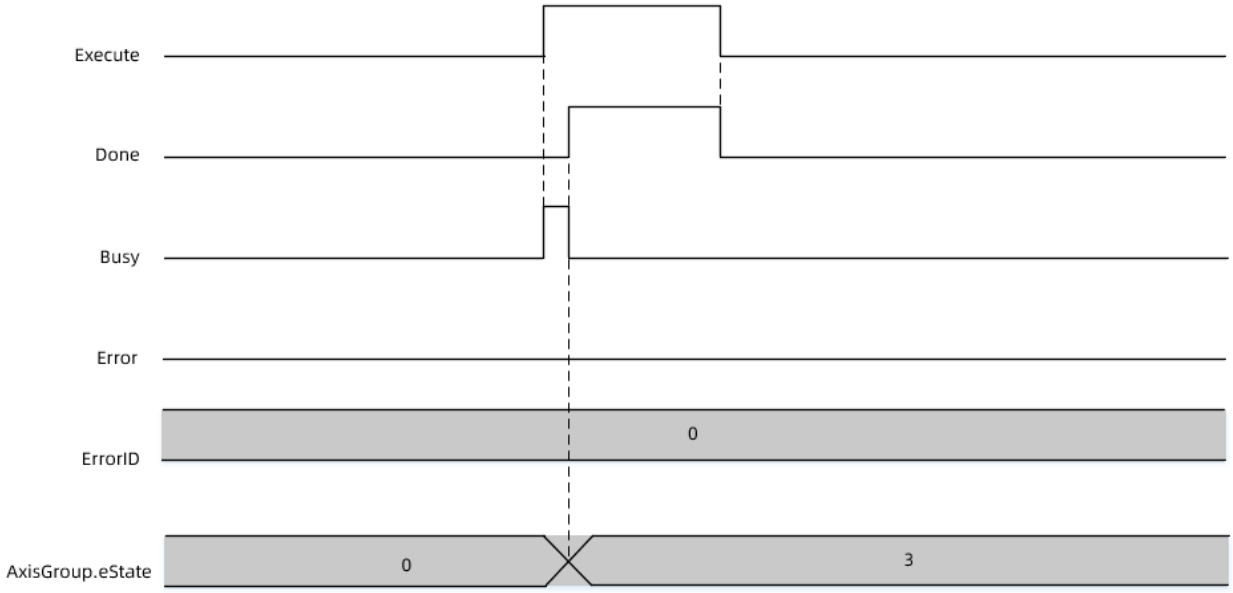
Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**
  - This instruction can change the axis group state specified by AxesGroup from GroupDisabled to GroupStandby.
  - When the axis group is in the GroupStandby state, other axis group instructions can be executed.
  - This instruction will change the state of the Axis variable within the axis group specified by AxesGroup. When the output variable Done of this instruction becomes TRUE, **if AxesGroup is already enabled, the state of the Axis variable will be changed to Synchronized_Motion. If AxesGroup is not yet enabled, the Axis variable will not enter the group synchronized status. The bound axis can still be independently controlled.**
- **Precautions**
  - If the axes in the axis group type specified by AxesGroup of the instruction are not configured with the corresponding Axis variable, the instruction will return an error. Configure the Axis variable for the axis group specified by AxesGroup as required.
  - If the axis group type is SCARA, use MC_SetPosition or MC_Home to set the position of the corresponding axis for each joint within the SCARA configuration range before using the MC_GroupEnable instruction.
  - When this instruction is enabled, the Axis variable mapped to the axis group specified by AxesGroup of the instruction must be in the PowerOff or StandStill state.
  - When this instruction is enabled, if the Axis variable mapped to the axis group specified by AxesGroup of the instruction is in the ErrorStop state, output the error code of the corresponding axis.
  - **The same Axis variable can be added to different AxesGroup variables, but only one of these AxesGroup variables can be enabled.** If an AxesGroup variable that includes the configured Axis variable is already enabled, all other AxesGroup variables that include this configured Axis variable cannot be enabled. Only after the MC_GroupDisable instruction is used on the AxesGroup that includes this configured Axis, all other AxesGroup variables that include this Axis can be enabled.
  - After the output variable Done of this instruction becomes TRUE, it indicates that the axis group specified by AxesGroup has been enabled, and the single axis instruction cannot be executed on any Axis variable in AxesGroup.
  - The Axis variable in AxesGroup can be used as Master in a synchronous control instruction.
  - The Axis variable in AxesGroup must be in CSP or CSV mode, otherwise the instruction will return an error.
  - **The Axis variable in AxesGroup can only be configured in linear mode, otherwise the instruction will return an error.**
- **Sequence diagram**

When the input parameter Execute is set to TRUE, the output parameter Busy shows the value TRUE. The AxesGroup state is changed from GroupDisabled to GroupStandby, and the output parameter "Done" becomes TRUE.



- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.5 MC_GroupHalt

This instruction stops the motion of an axis group. It is generally used under normal conditions.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GroupHalt	Stop axis group	MC_GroupHalt_Instance_0 MC_GroupHalt AxesGroup Execute DecelerationFactor JerkFactor Done Busy Active CommandAborted Error ErrorID	MC_GroupHalt_Instance_0(AxesGroup := , Execute := , DecelerationFactor := , JerkFactor := , Done => , Busy => , Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
DecelerationFactor	Deceleration factor	LREAL	[0.01,1.0]	0.5	Ratio relative to the maximum deceleration in the axis group workspace/joint space. If the factor is greater than 1, it is corrected to 1. If the factor is less than 0.01, it is corrected to 0.5. If the factor equals 0.01, no correction is applied.
JerkFactor	Jerk factor	LREAL	[0.0,1.0]	0.5	Ratio relative to the maximum jerk in the axis group workspace/joint space. For the non-trapezoidal motion profiling, the factor will be corrected to 1 if it is greater than 1. If the factor is less than or equal to 0, it will be corrected to 0.5.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Instruction has controlled the axis group	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis is being controlled.
Commandborted	Command aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

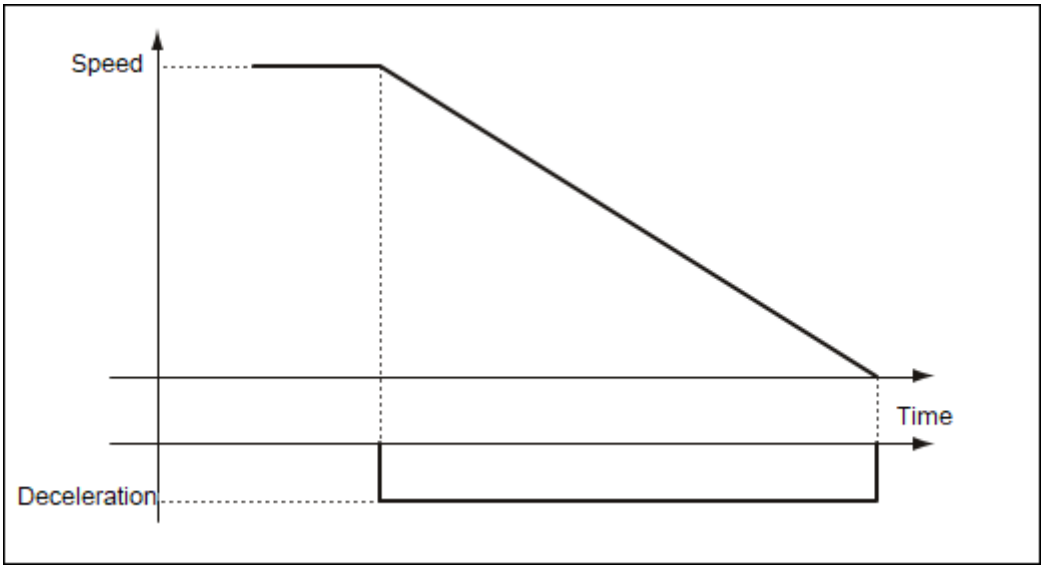
Input Variable	Name	Correction Description for Values Greater Than 1 or Backend Limit	Correction Description for Parameters Below Minimum Value Range	Correction Description for Parameters Equal to Minimum Value Range
DecFactor	Deceleration factor	If the deceleration factor is greater than 1, it is automatically corrected to 1.	If the deceleration factor is less than 0.01, it is automatically corrected to 0.5 for operation.	If the deceleration factor is equal to 0.01, no correction is applied.
JerkFactor	Jerk factor	If the jerk factor is greater than 1 in the non-trapezoidal motion profiling, it is automatically corrected to 1.	If the jerk factor is less than 0 in the non-trapezoidal motion profiling, it is automatically corrected to 0.5 for operation.	If the jerk factor is equal to 0 in the non-trapezoidal motion profiling, it is automatically corrected to 0.5 for operation.

- **Function**

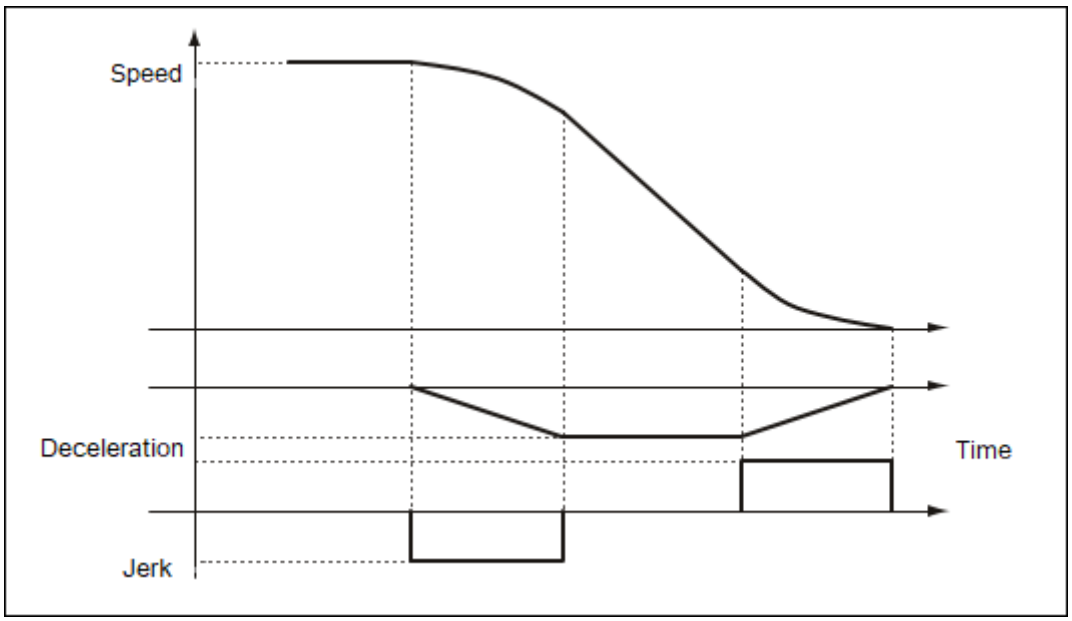
  This instruction is executed on the rising edge of Execute and is used to stop the AxesGroup during its normal operation.

  - **DecelerationFactor, JerkFactor**
    - The deceleration and jerk during the execution of this instruction are determined by the input variables DecelerationFactor and JerkFactor.
    - In the ACS coordinate system, the deceleration stop is performed on each Axis variable in AxesGroup according to the product of the software-limited maximum axis deceleration and DecelerationFactor, and the product of the software-limited maximum axis jerk and JerkFactor. In the PCS coordinate system, the deceleration stop is performed according to the product of the maximum deceleration for AxesGroup and DecelerationFactor, and the product of the maximum axis group jerk and JerkFactor.
    - JerkFactor

      For JerkFactor = 0, the velocity curve and deceleration curve are shown as follows.



For JerkFactor ≠ 0, the velocity curve, deceleration curve, and jerk curve are shown as follows.



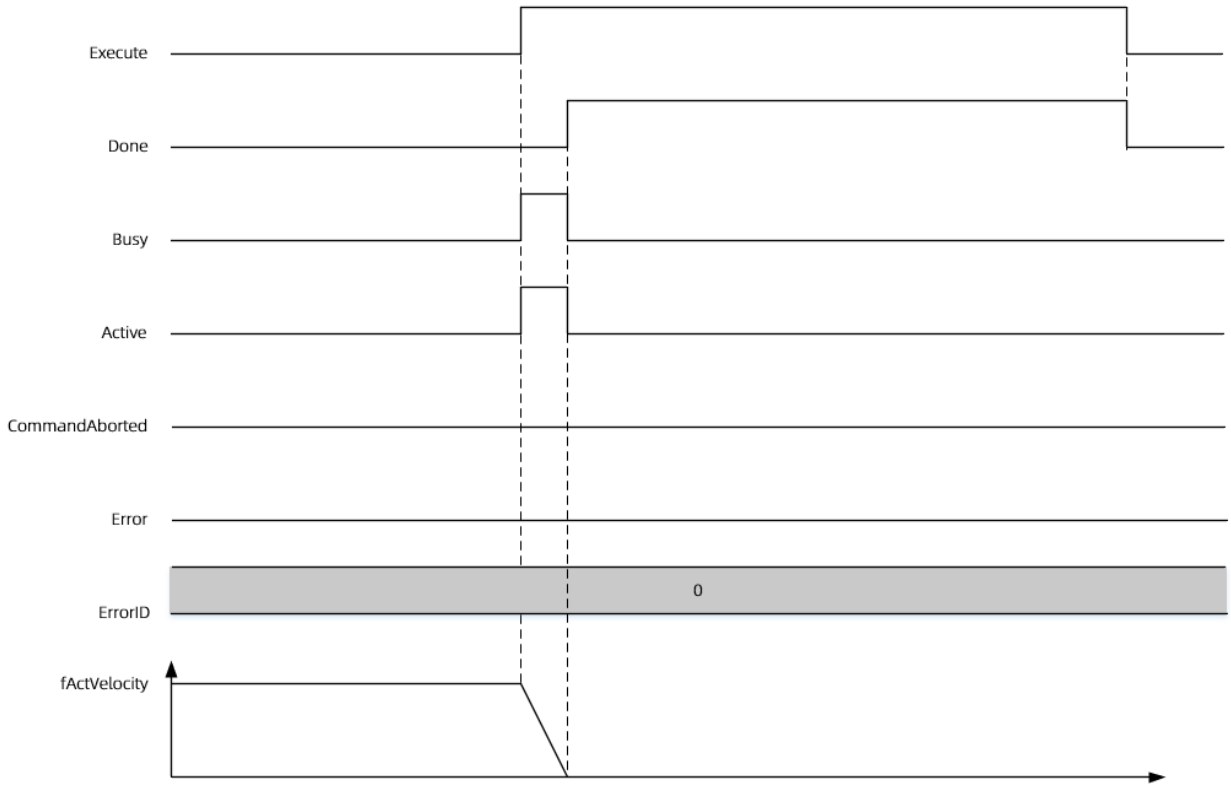
- This instruction can only be initiated when the axis group is in the GroupMoving or GroupStandby state. Attempting to start this instruction in any other state will result in an error.
- This instruction can interrupt the current motion of AxesGroup. AxesGroup remains in the GroupMoving state until the combined velocity of the axes in the current motion space (work space/axis space) reaches 0. Then, the instruction Done is set to TRUE, and the group state returns to GroupStandby or remains in GroupMoving, with the following status changes:
  - If the current motion is in a static coordinate system, this instruction will switch the AxesGroup from GroupMoving to GroupStandby.
  - If the current motion is in a dynamic coordinate system, this instruction remains in the GroupMoving state.
- This instruction can be aborted by other motion control instructions.
- When AxesGroup is in the homing process, this instruction cannot be used, otherwise an error will occur.
- When the AxesGroup is in the GroupStopping state, this instruction cannot be used, otherwise an error will occur.

- This instruction can be interrupted by other MC_GroupHalt instructions, which can interrupt each other sequentially. That is, the MC_GroupHalt instruction that starts later can interrupt the one that starts earlier, and the operation from deceleration to stop will be performed according to the parameters of the MC_GroupHalt instruction that starts later.
- During the execution of the instruction, all axes in AxesGroup must be in the enabled status. Otherwise, an error will be reported.

- **Re-triggering the instruction**
  - This instruction cannot be re-executed.
- **Precautions**
  - The trajectory planning curve type of AxesGroup must be set as required. If the planning curve type is set to a trapezoidal curve, the input variable JerkFactor for this instruction becomes invalid, and the instruction executes a deceleration to a stop according to DecelerationFactor. If the planning curve type is set to a cubic curve or a septic curve, the instruction executes a deceleration to a stop according to DecelerationFactor and JerkFactor.
  - Whether the planning curve type of each Axis variable in AxesGroup is set to a trapezoidal curve or an S-curve does not affect the planning method of AxesGroup.
- **Sequence diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates the instruction has been executed successfully; the parameter Busy indicates that the current function block is being executed; the parameter CommandAborted indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

  The parameter Busy shows the value TRUE on a rising edge of Execute. In the next cycle, "Active" becomes TRUE.



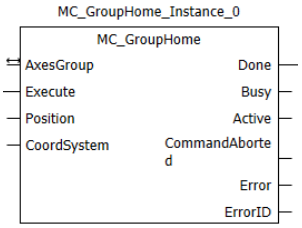
- **Faults**

For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.6 MC_GroupHome

The instruction homes the axis group.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GroupHome	Axis group homing		MC_GroupHome_Instance_0(AxesGroup := , Execute := , Position := , CoordSystem := , Done => , Busy => , Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
Position	Position	ARRAY [1..GROUP_DEF.AXIS_NUM] OF LREAL	-	[0,0,0…]	Absolute coordinates of the mechanical homing position

Input Variable	Name	Data Type	Valid Range	Default Value	Description
CoordSystem	Coordinate system type	18.3.3.2.4.19 MC_COORDSYSTEM: Coordinate System Enumeration	ACS	ACS	Select the application coordinate system type.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Instruction has controlled the axis group	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**
  - This instruction is triggered on the rising edge of Execute, moving AxesGroup to its mechanical homing position.
  - This instruction can define the homing sequence of each axis in the AxesGroup through the iFA interface, as shown in the following figure:

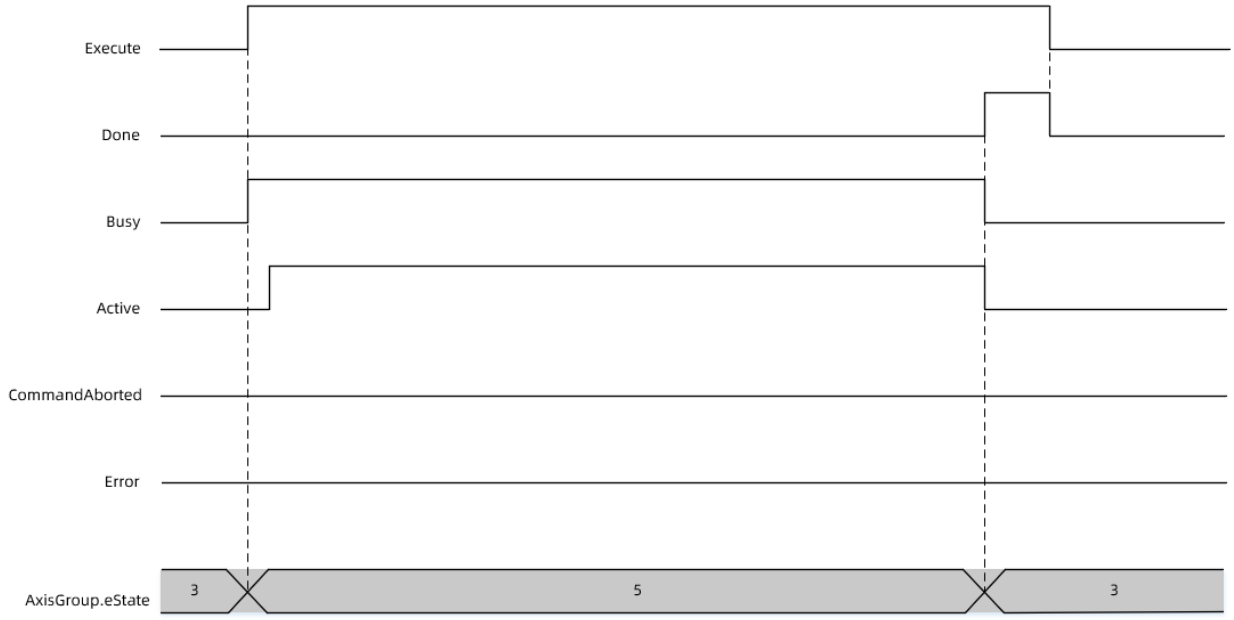


  - The defined homing sequence must not contain descending order, and the homing sequence number of the first axis must be 1.
  - Take the XYZA axis group type as an example: The definable axis group homing sequences include: 1, 1, 1, 1; 1, 2, 3, 4; 1, 2, 2, 2; 1, 2, 2, 3; 1, 2, 3, 3; and other sequences.
  - The homing sequence is currently only allowed to be defined in the axis group structure.

- **Re-triggering the instruction**
  - This instruction cannot be re-triggered.
- **Precautions**
  - This instruction can only be executed when the AxesGroup is in the GroupStandby status (also in the GroupStandby status when not following in a dynamic coordinate system). An error is reported when this instruction is activated in other AxesGroup states.
  - This instruction can only be interrupted by the MC_GroupStop instruction.
  - This instruction only supports homing in the ACS coordinate system.
  - Before executing the MC_GroupHome instruction, verify that the post-homing joint angles will place the axis group within its defined workspace limits according to the motion range of the axis group configuration in the workspace. Otherwise, it may lead to abnormal operation or errors of the axis group.
  - This instruction currently supports the servo homing method. Other homing methods will be developed later.
- **Sequence diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates the instruction has been executed successfully; the parameter Busy indicates that the current function block is being executed; the parameter CommandAborted indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

  The parameter Busy shows the value TRUE on a rising edge of Execute. In the next cycle, "Active" becomes TRUE.



- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.7 MC_GroupInterrupt

This instruction pauses an axis group along the current motion path of the axis group, which does not change the current motion state of the axis group.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GroupInterrupt	Axis group pause	MC_GroupInterrupt_Instance_0 MC_GroupInterrupt AxesGroup Execute DecelerationFactor JerkFactor Done Busy Active CommandAborted Error ErrorID	MC_GroupInterrupt_Instance_0(AxesGroup := , Execute := , DecelerationFactor := , JerkFactor := , Done => , Busy => , Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
DecelerationFactor	Deceleration factor	LREAL	[0.01,1.0]	0.3	Ratio relative to the maximum deceleration in the axis group workspace/joint space. If the factor is greater than 1, it will be corrected to 1. If the factor is less than 0.01, it will be corrected to 0.3.
JerkFactor	Jerk factor	LREAL	[0.0,1.0]	0.3	Ratio relative to the maximum jerk in the axis group workspace/joint space. For the non-trapezoidal motion profiling, the factor will be corrected to 1 if it is greater than 1. If it is less than or equal to 0, it will be corrected to 0.3.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Instruction has controlled the axis group	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

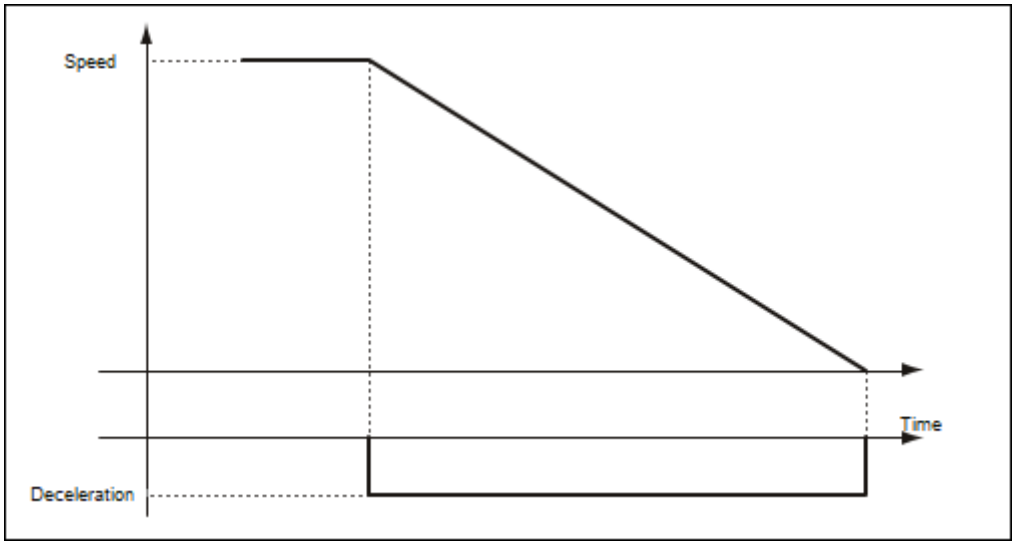
Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

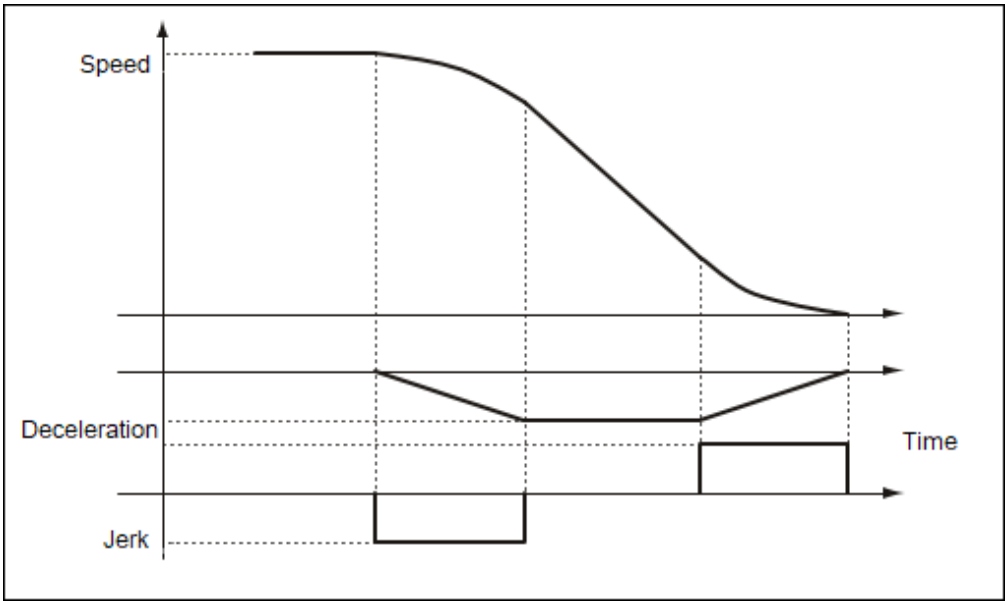
Input Variable	Name	Correction Description for Values Greater Than 1 or Backend Limit	Correction Description for Parameters Below Minimum Value Range	Correction Description for Parameters Equal to Minimum Value Range
DecelerationFactor	Deceleration factor	If the deceleration factor is greater than 1, it is automatically corrected to 1.	If the deceleration factor is less than 0.01, it is automatically corrected to 0.3.	If the deceleration factor is equal to 0.01, no correction is applied.
JerkFactor	Jerk factor	If the jerk factor is greater than 1 in the non-trapezoidal motion profiling, it is automatically corrected to 1.	If the jerk factor is less than 0 in the non-trapezoidal motion profiling, it is automatically corrected to 0.3 for operation.	If the jerk factor is equal to 0 in the non-trapezoidal motion profiling, it is automatically corrected to 0.3 for operation.

- **Function**
  - This instruction is executed on the rising edge of Execute to pause an AxesGroup. It will temporarily stop along the current motion path of the AxesGroup without changing its current motion status.
  - After this instruction has been activated, the deceleration motion is performed based on DecelerationFactor and JerkFactor. After activation, the currently executing motion instruction of the axis group is not interrupted, the Busy flag remains set, the Active flag is reset, and the axis group maintains its original state.
  - To resume the motion paused by this instruction, the MC_GroupContinue instruction must be triggered.
  - This instruction can be interrupted by MC_GroupContinue, MC_GroupHalt, MC_GroupStop, and other MC_GroupInterrupt instructions, or by new motion instructions for AxesGroup.
  - **DecelerationFactor, JerkFactor**
    - The deceleration and jerk during the execution of this instruction are determined by the input variables DecelerationFactor and JerkFactor.
    - In joint space (ACS), the pause operation is performed based on the product of the maximum axis deceleration limited by software for each Axis variable in AxesGroup and DecelerationFactor, and the product of the maximum axis jerk limited by software for each axis and JerkFactor.
    - In work space (PCS), the pause operation is performed based on the product of the maximum deceleration for AxesGroup and DecelerationFactor, and the product of the maximum axis group jerk and JerkFactor.
    - JerkFactor

      For JerkFactor = 0, the velocity curve and deceleration curve are shown as follows.



For JerkFactor ≠ 0, the velocity curve, deceleration curve, and jerk curve are shown as follows.



- **Re-triggering the instruction**
  - This instruction cannot be re-triggered.
- **Precautions**
  - When AxesGroup is in the GroupStandby, GroupErrorstop, GroupDisabled, GroupHoming, or GroupStopping state, this instruction cannot be started. Otherwise, the instruction will return an error.
  - When AxesGroup is controlled by the MC_GroupHalt instruction or the axis group is in the homing process, this instruction cannot be started. Otherwise, an error will be generated.
  - The trajectory planning curve type of AxesGroup must be set as required. If the planning curve type is set to a trapezoidal curve, the input variable JerkFactor for this instruction becomes invalid, and the instruction performs the pause operation according to DecelerationFactor. If the planning curve type is set to a cubic curve or a septic curve, the instruction performs the pause operation according to DecelerationFactor and JerkFactor.
  - Whether the planning curve type of each Axis variable in AxesGroup is set to a trapezoidal curve or an S-curve does not affect the planning method of AxesGroup.
  - During the execution of the instruction, all axes in AxesGroup must be in the enabled status. Otherwise, an error will be reported.

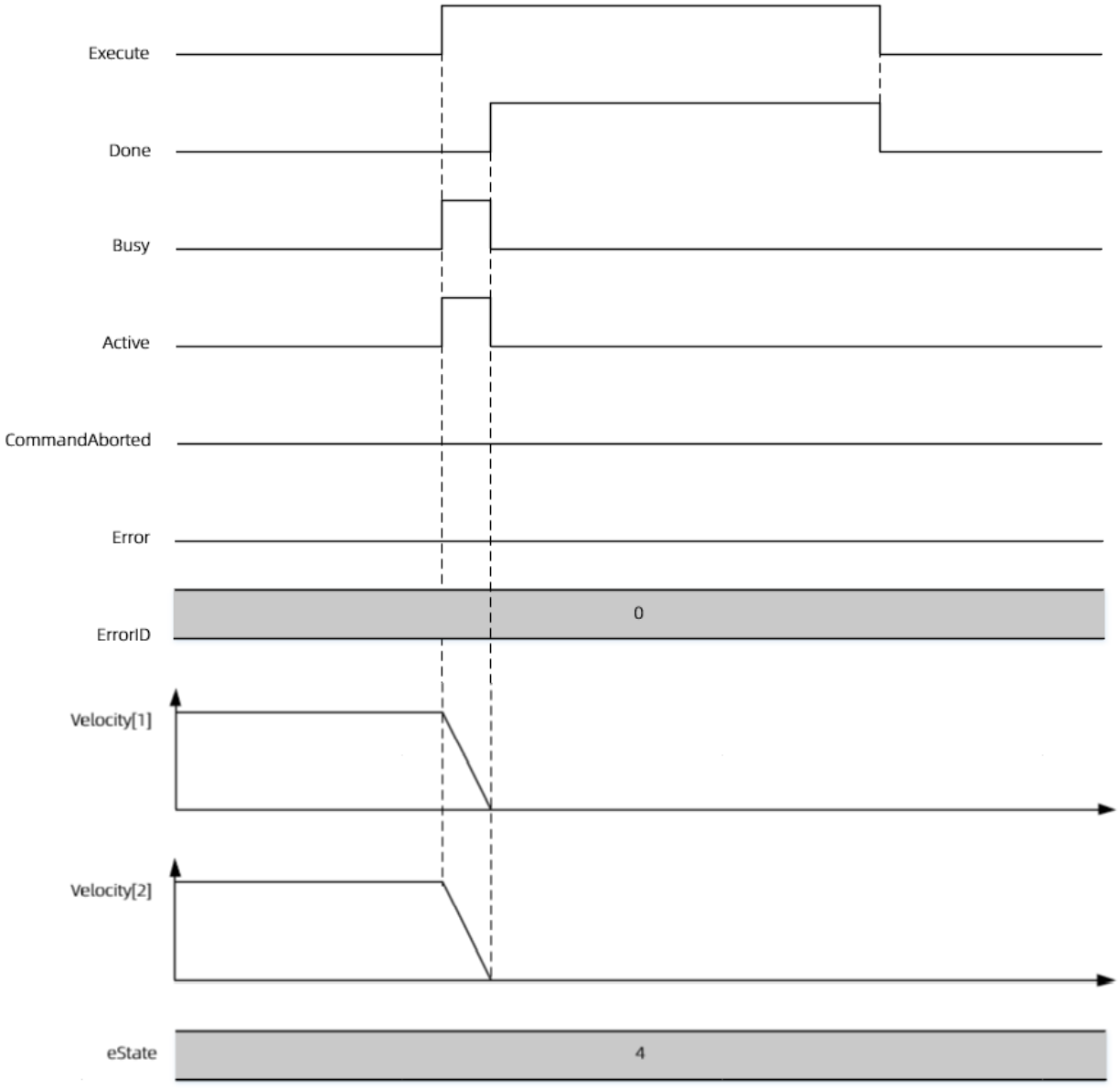
- **Sequence diagram**

  The Execute input of the function block must be triggered on a rising edge. The Done output of the function block indicates the instruction is executed successfully. The Busy output of the function block indicates that the current function block is being executed. The CommandAborted output of the function block indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

  On the rising edge of Execute, both Busy and Active outputs are set to TRUE.

  When the velocity of all axes reaches "0", "Done" becomes TRUE. During this process, "eState" (Axis group state machine) remains at 4, with the axis group state machine in GROUP_MOVING.

  When this instruction is aborted by another instruction, CommandAborted becomes TRUE, and "Busy" and "Active" become FALSE.



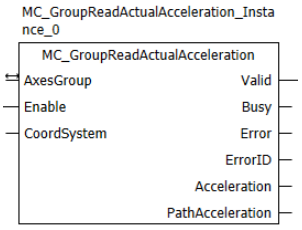
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.8 MC_GroupReadActualAcceleration

The instruction reads the actual acceleration of the axis group in the selected coordinate system.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GroupReadActualAcceleration	Read axis group actual acceleration		MC_GroupReadActualAcceleration_Instance_0(AxesGroup := , Enable := , CoordSystem := , Valid => , Busy =>, Error =>, ErrorID => , Acceleration => , PathAcceleration =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: The instruction is enabled. FALSE: The instruction is not enabled.
CoordSystem	Coordinate system type	18.3.3.2.4.19 MC_COORDSYSTEM: Coordinate System Enumeration	ACS, PCS	ACS	Select the application coordinate system type.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Valid output	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.
Acceleration	Acceleration component	ARRAY [1..GROUP_DEF.AXIS_NUM] OF LREAL	-	[0,0,…]	Return the acceleration component of the axis group in the corresponding coordinate systemAxis group type description.
PathAcceleration	Resultant acceleration	LREAL	-	0	Returns the resultant acceleration of the TCP of the axis group in the primary dimension space.

- **Function**
  - When the Valid output is TRUE, this function captures two sets of data per control cycle: the actual acceleration components of all Axis variables in AxesGroup under the selected CoordSystem and the resultant acceleration of the axis group in the PCS coordinate system.
- **Re-triggering the instruction**
  - An "Enable"-type instruction does not allow re-triggering while it is active (Busy).
- **Precautions**
  - During the execution of the instruction, the axis group specified by AxesGroup must not be in the GroupDisabled state.
  - During the execution of the instruction, if the axis group specified by AxesGroup is in the GroupErrorStop state and an error occurs in the Axis variable within the axis group, the instruction will return an error. When the axis group is in any other states, there is no requirement for the state of the Axis variable.
  - The output variable Acceleration refers to the acceleration components in the given coordinate system (ACS/PCS), and the unit of each acceleration component is consistent with the user unit of the corresponding axis type in the given coordinate system.
  - The output variable PathAcceleration is the resultant acceleration of the path in the primary dimension space. Regardless of whether the PCS or ACS coordinate system is selected, this parameter returns the resultant acceleration of the TCP of the axis group in the user's primary dimension subspace of the part coordinate system PCS (position, orientation).

    When the primary dimension is set to position: The instruction outputs the resultant linear acceleration of the TCP in the position subspace of the user part coordinate system.

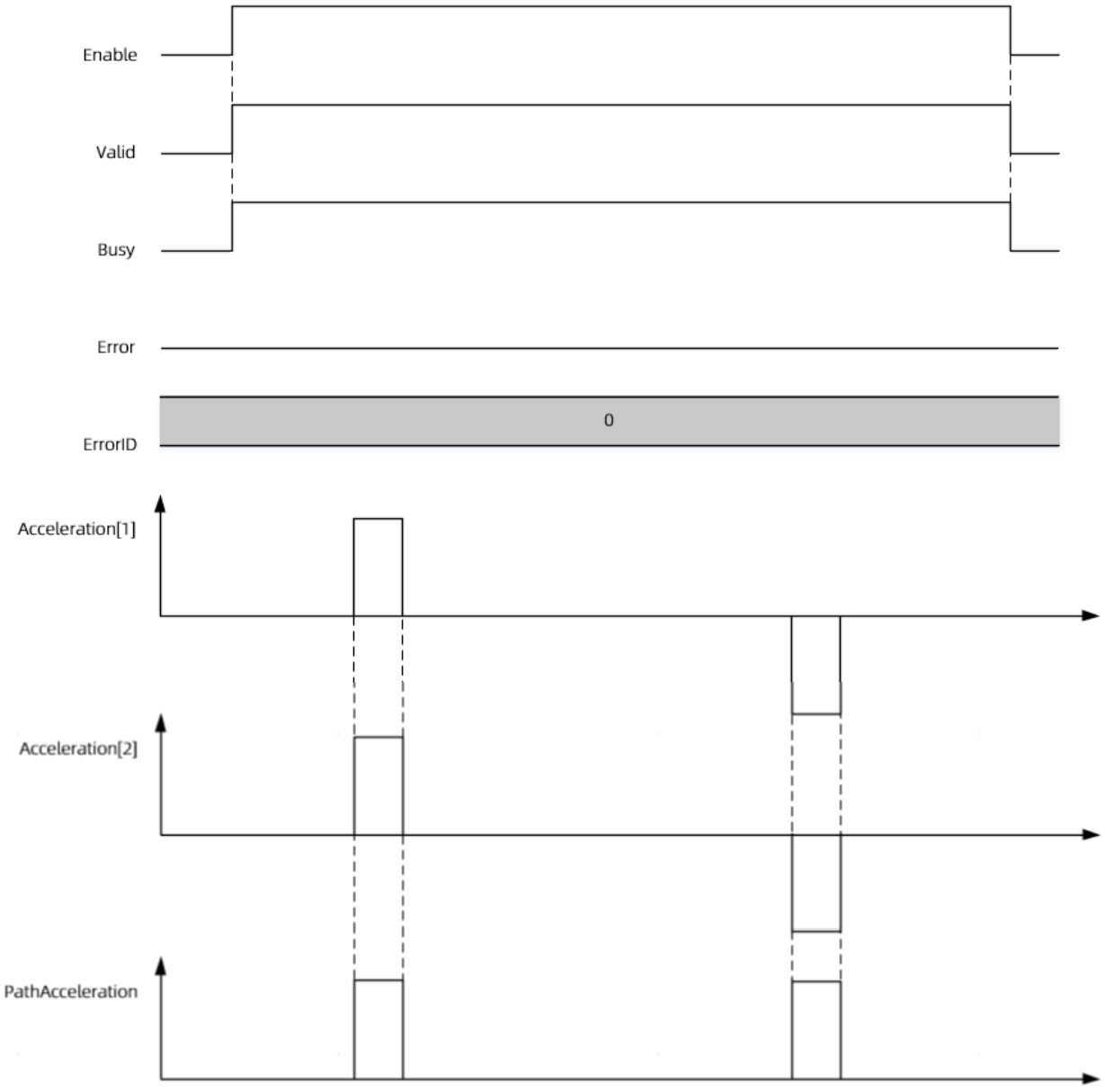
When the primary dimension is set to orientation: The instruction outputs the orientation angular acceleration in the orientation subspace of the user part coordinate system.

When the primary dimension is set to ratio: The instruction outputs the resultant acceleration of the TCP in the user part coordinate system (dimensionless).

- **Sequence diagram**

  When the input parameter "Enable" is set to TRUE, the output parameter "Valid" becomes TRUE, indicating that the output values "Acceleration" and "Path Acceleration" are valid; the output parameter "Busy" becomes TRUE, indicating that the current function block is executing.

  When the input parameter "Enable" is set to FALSE, the output parameter "Valid" becomes FALSE, and the output parameter "Busy" becomes FALSE.



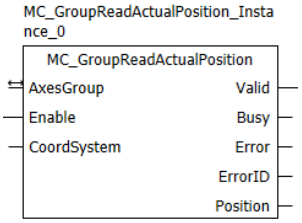
- **Error description**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.9 MC_GroupReadActualPosition

This instruction reads the actual position of an axis group in the selected coordinate system.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GroupReadActualPosition	Read axis group actual position		MC_GroupReadActualPosition_Instance_0(AxesGroup := , Enable := , CoordSystem := , Valid => , Busy =>, Error =>, ErrorID => , Position =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: The instruction is enabled. FALSE: The instruction is not enabled.
CoordSystem	Coordinate system type	18.3.3.2.4.19 MC_COORDSYSTEM: Coordinate System Enumeration	ACS, PCS	ACS	Select the application coordinate system type.

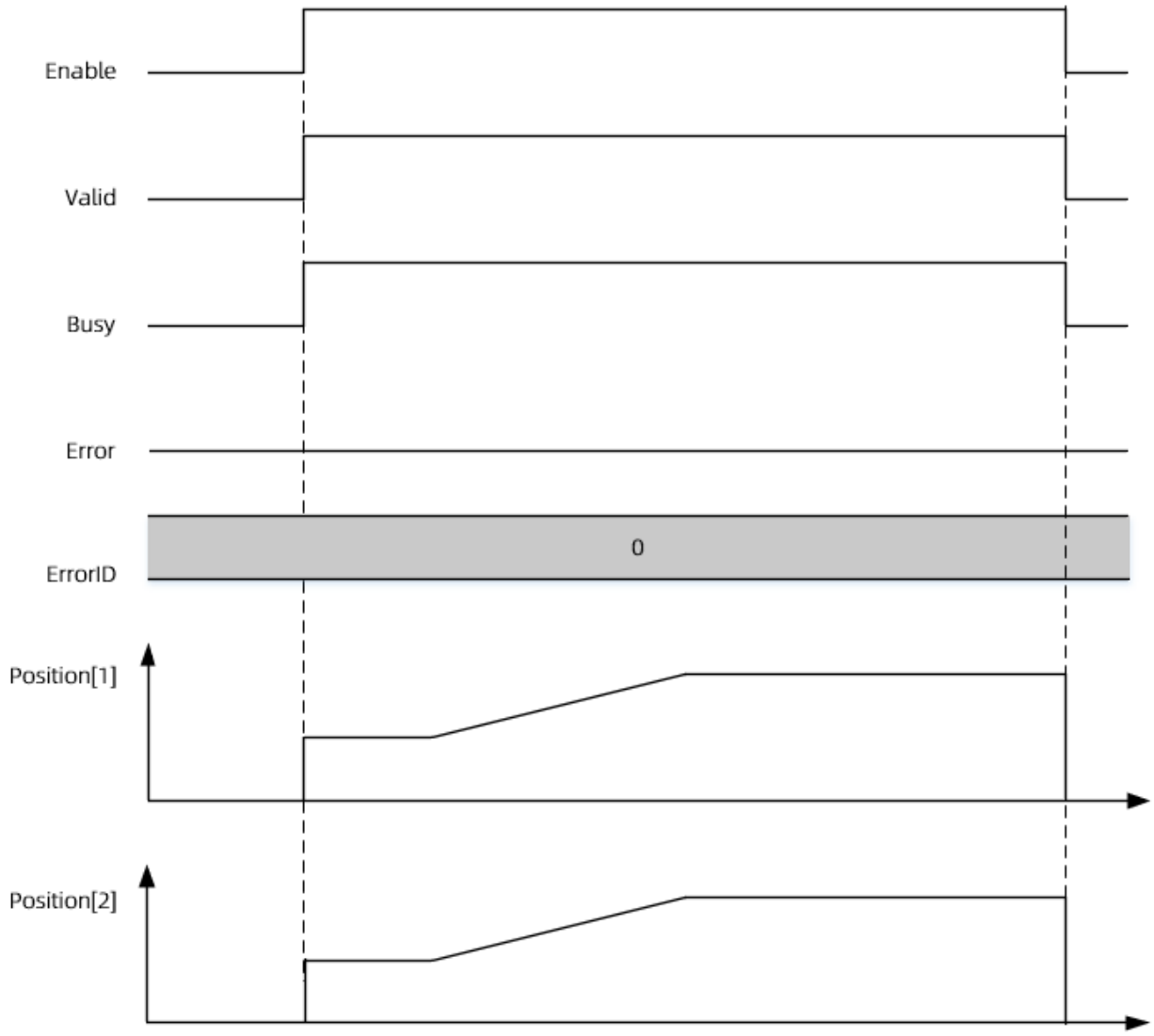
Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Valid output	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.
Position	Position	ARRAY [1..GROUP_DEF.AXIS_NUM] OF LREAL	-	[0,0,…]	Returns the current position [u] of the axis group in the specified coordinate systemAxis group type description.

- **Function**
  - When the Valid output is TRUE, the actual positions of all Axis variables in the axis group specified by AxesGroup of the instruction under the selected CoordSystem are obtained per control cycle.
- **Re-triggering the instruction**
  - An "Enable"-type instruction does not allow re-triggering while it is active (Busy).
- **Precautions**
  - During the execution of the instruction, the axis group specified by AxesGroup must not be in the GroupDisabled state.
  - During the execution of the instruction, if the axis group specified by AxesGroup is in the GroupErrorStop state and an error occurs in the Axis variable within the axis group, the instruction will return an error. When the axis group is in any other states, there is no requirement for the state of the Axis variable.
  - The output variable Position represents the position components of the axis group in the given coordinate system (ACS/PCS). The unit of each position component is consistent with the user unit of the corresponding axis type in the given coordinate system.
- **Sequence diagram**

  When the input parameter "Enable" is set to TRUE, the output parameter "Valid" becomes TRUE, indicating that the output value "Position" is valid. The output parameter "Busy" becomes TRUE, indicating that the current function block is executing.

  When the input parameter "Enable" is set to FALSE, the output parameter "Valid" becomes FALSE, and the output parameter "Busy" becomes FALSE.



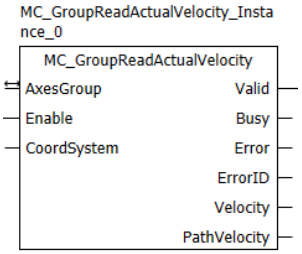
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.10 MC_GroupReadActualVelocity

This instruction reads the actual velocity of an axis group in the selected coordinate system.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GroupReadActualVelocity	Read axis group actual velocity		MC_GroupReadActualVelocity_Instance_0(AxesGroup := , Enable := , CoordSystem := , Valid => , Busy =>, Error =>, ErrorID => , Velocity => , PathVelocity =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: The instruction is enabled. FALSE: The instruction is not enabled.
CoordSystem	Coordinate system type	18.3.3.2.4.19 MC_COORDSYSTEM: Coordinate System Enumeration	ACS, PCS	ACS	Select the application coordinate system type.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Valid output	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.
Velocity	Velocity component	ARRAY [1..GROUP_DEF.AXIS_NUM] OF LREAL	-	[0,0,….]	Return the velocity component [u] of the axis group in the corresponding coordinate systemAxis group type description.
PathVelocity	Resultant velocity	LREAL	-	0	Returns the resultant velocity of the TCP of the axis group in the primary dimension space.

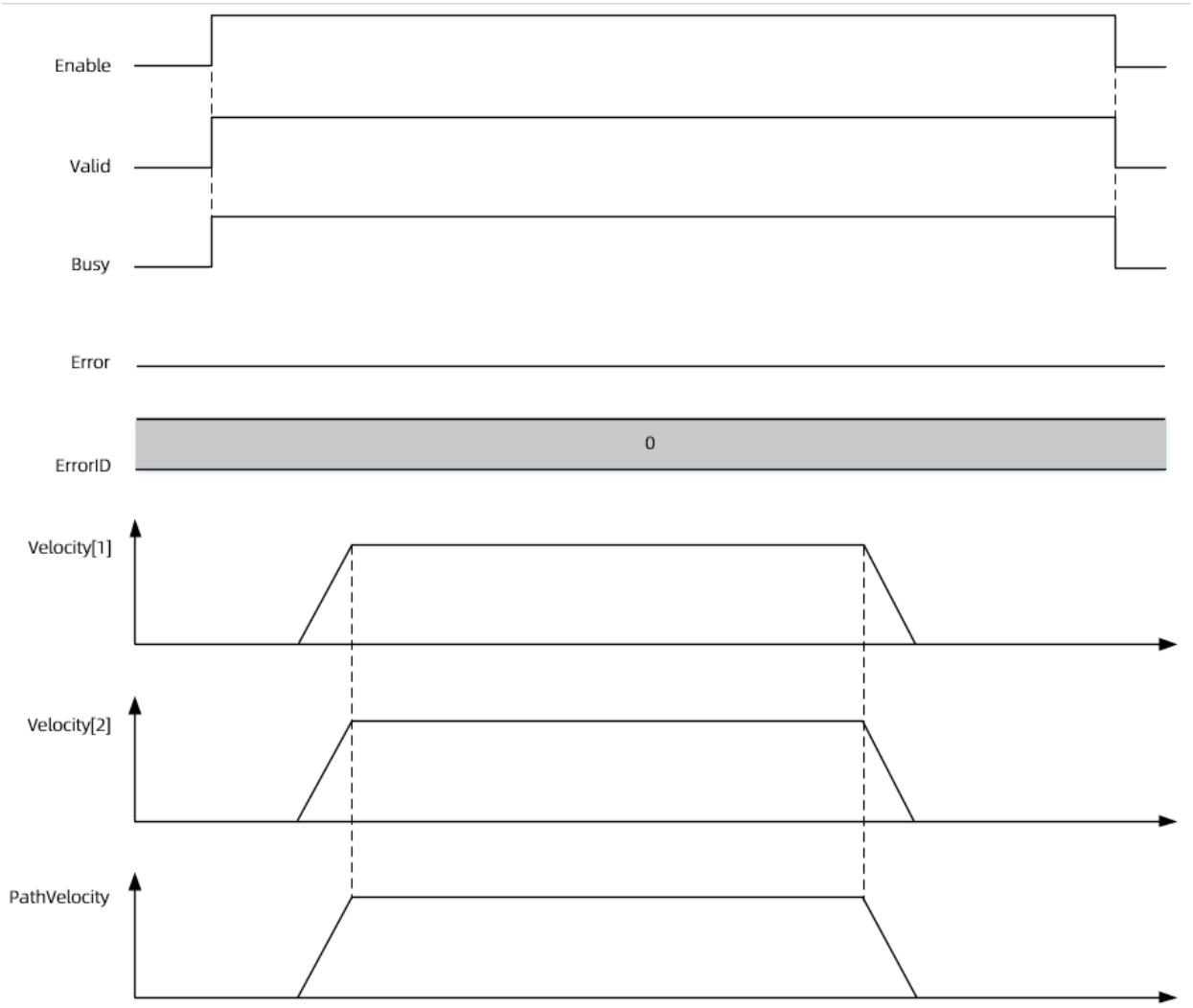
- **Function**
  - When the Valid output is TRUE, this function captures two sets of data per control cycle: the actual velocity components of all Axis variables in the axis group specified by AxesGroup under the selected CoordSystem and the resultant acceleration of the axis group in the PCS coordinate system.
- **Re-triggering the instruction**
  - An "Enable"-type instruction does not allow re-triggering while it is active (Busy).
- **Precautions**
  - During the execution of the instruction, the axis group specified by AxesGroup must not be in the GroupDisabled state.
  - During the execution of the instruction, if the axis group specified by AxesGroup is in the GroupErrorStop state and an error occurs in the Axis variable within the axis group, the instruction will return an error. When the axis group is in any other states, there is no requirement for the state of the Axis variable.
  - The output variable Velocity represents the velocity components in the given coordinate system (ACS/PCS). The unit of each velocity component is consistent with the user unit of the corresponding axis type in the given coordinate system.
  - The output variable PathVelocity represents the resultant velocity in the primary dimension space path. Regardless of whether PCS or ACS coordinate system is selected, this parameter returns the resultant velocity of the TCP of the axis group in the user's primary dimension subspace of the part coordinate system PCS (position, orientation).
    - When the primary dimension is set to position: The instruction outputs the resultant linear velocity of the TCP in the position subspace of the user part coordinate system.

- When the primary dimension is set to orientation: The instruction outputs the orientation angular velocity in the orientation subspace of the user part coordinate system.
- When the primary dimension is set to ratio: The instruction outputs the resultant velocity of the TCP in the user part coordinate system (dimensionless).

- **Sequence diagram**

  When the input parameter "Enable" is set to TRUE, the output parameter "Valid" becomes TRUE, indicating that the output values "Velocity" and "PathVelocity" are valid. The output parameter "Busy" becomes TRUE, indicating that the current function block is executing.

  When the input parameter "Enable" is set to FALSE, the output parameter "Valid" becomes FALSE, and the output parameter "Busy" becomes FALSE.



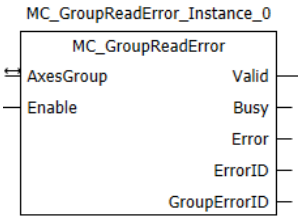
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.11 MC_GroupReadError

This instruction reads the error message of the specified axis group and describes the general axis group errors that are not related to the axis group instruction error.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GroupReadError	Axis group error reading		MC_GroupReadError_Instance_0(AxesGroup := , Enable := , Valid =>, Busy =>, Error =>, ErrorID => , GroupErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: The instruction is enabled. FALSE: The instruction is not enabled.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Valid output	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

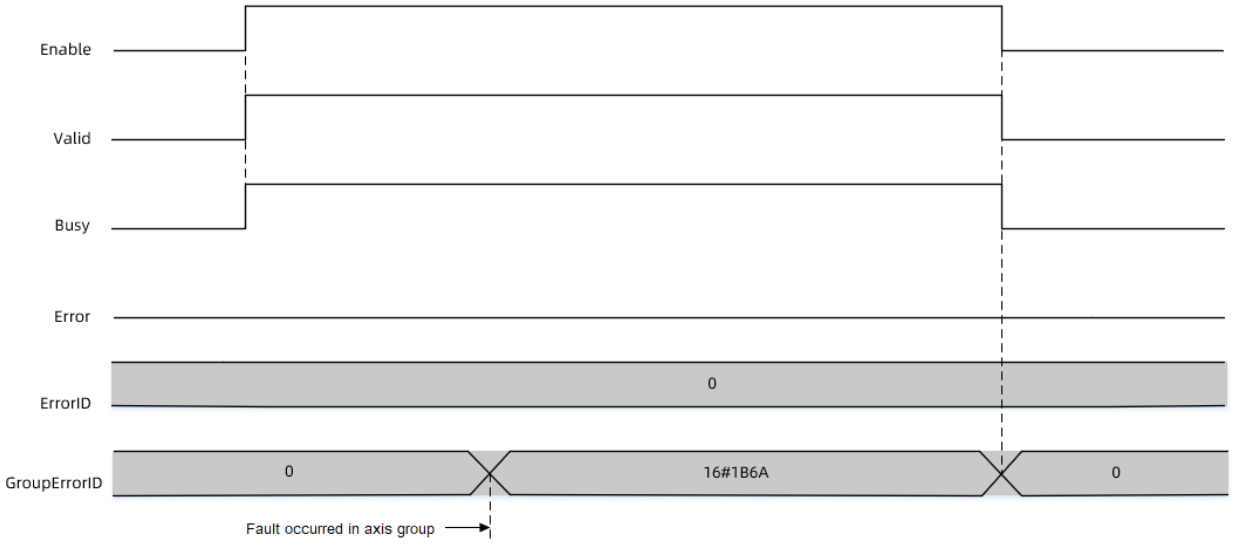
Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.
GroupErrorID	Fault code	DWORD	-	0	Axis group fault code.

- **Function**
  - When the variable Valid is TRUE, get the error information of the axis group specified by AxesGroup of the instruction. If an error occurs in the axis group, GroupErrorID outputs the corresponding error code. If the axis group has no error information, GroupErrorID outputs 0.
- **Re-triggering the instruction**
  - An "Enable"-type instruction does not allow re-triggering while it is active (Busy).
- **Precautions**
  - When the instruction is used, the axis group specified by AxesGroup must not be in the GroupDisabled state. However, there is no requirement for the state of the Axis variable in AxesGroup.
  - The output variable ErrorID of this instruction is the error code corresponding to the execution error of the instruction itself; GroupErrorID is the error code corresponding to the error information read from AxesGroup during the execution of the instruction.
- **Sequence diagram**

  When the input parameter Enable is set to TRUE, the output parameter Busy shows the value TRUE.

  When the output parameter Valid is set to TRUE, it indicates that the other output parameter GroupErrorID is valid.

  If a fault occurs on the axis group side during operation, the axis group fault code will be output to "GroupErrorID" on the instruction side.



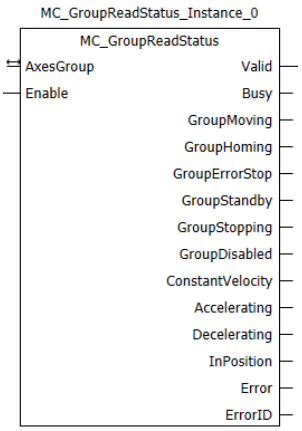
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.12 MC_GroupReadStatus

This instruction is used to read the status information of the specified axis group.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GroupReadStatus	Obtaining the axis group state		MC_GroupReadStatus_Instance_0(AxesGroup := , Enable := , Valid =>, Busy => , GroupMoving => , GroupHoming => , GroupErrorStop => , GroupStandby => , GroupStopping => , GroupDisabled => , ConstantVelocity =>, Accelerating =>, Decelerating => , InPosition => , Error => , ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enable the instruction; FALSE: Disable the instruction.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Valid output	BOOL	TRUE, FALSE	FALSE	Set to TRUE when under instruction control.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is received.
GroupMoving	The axis group is in motion.	BOOL	TRUE, FALSE	FALSE	The axis group state machine indicates that the axis group is moving.
GroupHoming	Axis group homing	BOOL	TRUE, FALSE	FALSE	The axis group state machine indicates that the axis group is performing homing.
GroupErrorStop	Axis group stopped due to an error	BOOL	TRUE, FALSE	FALSE	The axis group state machine indicates an axis group error stop.
GroupStandby	Axis group standby	BOOL	TRUE, FALSE	FALSE	The axis group state machine indicates that the axis group is in standby state.
GroupStopping	Axis group stopping	BOOL	TRUE, FALSE	FALSE	The axis group state machine indicates the axis group is stopping.
GroupDisabled	Disable axis group	BOOL	TRUE, FALSE	FALSE	The axis group state machine indicates the axis group is disabled.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ConstantVelocity	Constant speed	BOOL	TRUE, FALSE	FALSE	Move at a constant velocity along the instruction path.
Accelerating	Acceleration	BOOL	TRUE, FALSE	FALSE	Accelerating motion along the instruction path.
Decelerating	Deceleration	BOOL	TRUE, FALSE	FALSE	Decelerating motion along the instruction path.
InPosition	Target arrival	BOOL	TRUE, FALSE	FALSE	Target position reached
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**
  - When "Valid" is TRUE, get the status of the axis group specified by AxesGroup of the instruction.
- **Re-triggering the instruction**
  - An "Enable"-type instruction does not allow re-triggering while it is active (Busy).
- **Precautions**
  - During the execution of the instruction, there is no state requirement for the axis group specified by AxesGroup. There is no state requirement for the Axis variable in AxesGroup.
- **Sequence diagram**

  The MC_GroupReadStatus instruction reads each status of the axis group. It is enabled through the Enable signal level. This instruction can be executed repeatedly without affecting each other.

  When the input parameter Enable is set to TRUE, the output parameter Busy shows the value TRUE. When the output parameter Valid is TRUE, it indicates that the other output parameters are valid.

  When the axis group is not enabled, the axis group state machine is in the GroupDisabled state.

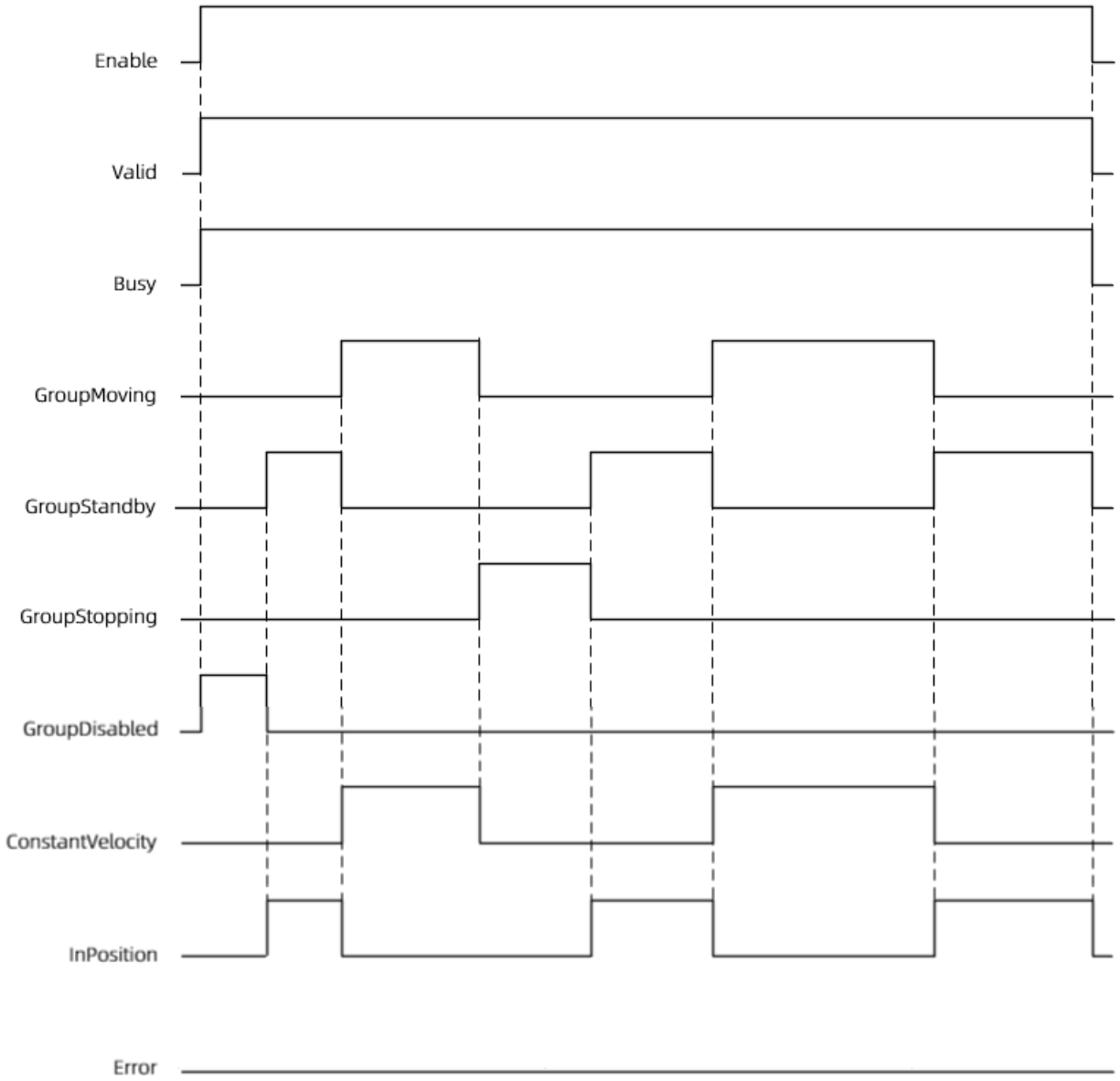
  When all axes in the axis group are mapped to the axis group, and all axes in the axis group have no error information, after the axis group is enabled, the axis group state machine is switched to the GroupStandby state.

  When the motion control instruction is called to perform speed control based on position, the axis group state machine is switched to the GroupMoving state. "ConstantVelocity" is TRUE when the axis

group is moving at a constant velocity along the path. "InPosition" is TRUE when the axis group move to the target position.

When the MC_GroupStop instruction is executed to perform the emergency stop function, the axis group state machine is switched to the GroupStopping state.

When the output variable Done of the MC_GroupStop instruction is set, and the input variable Execute is reset, the axis group state machine is switched to the GroupStandby state.



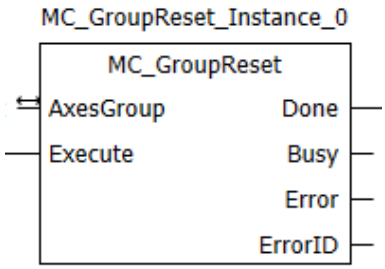
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.13 MC_GroupReset

This instruction resets the axis group, switching the axis group state from GroupErrorStop to GroupStandby by resetting the relevant errors within the axis group. It also resets axis errors on all axes within the axis group.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GroupReset	Reset axis group		MC_GroupReset_Instance_0(AxesGroup := , Execute := , Done =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

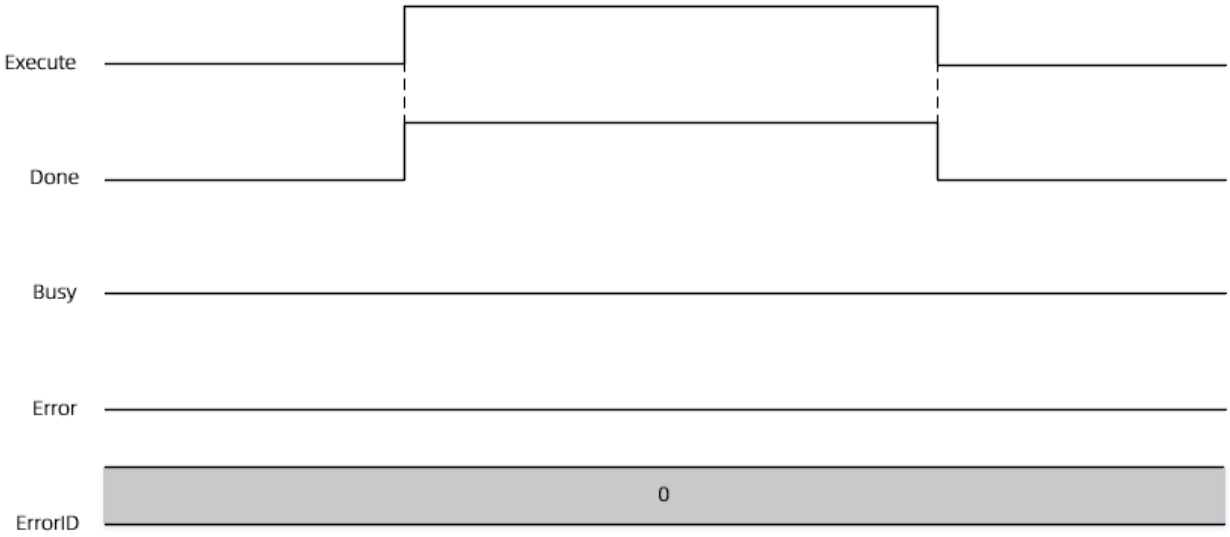
Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**
  - On the rising edge of Execute, this instruction resets errors in an axis group that is in the GroupErrorStop state, including errors of individual axes within the group.
  - **Execution of the other instructions when executing this instruction**
    - If multiple instances of the MC_GroupReset instruction are executed for the same AxesGroup, the instructions for the axis groups are processed sequentially in the order of their triggers.
    - When the MC_GroupReset instruction is executed on AxesGroup and the MC_Reset instruction is simultaneously executed on the Axis variable within the specified AxesGroup, the two instructions will be executed in the order of their triggers.
- **Precautions**
  - During the execution of the instruction, if the axis group is in the GroupErrorStop state, but the stop action has not yet been completed, the axis group fails to be reset. The instruction will return an error.
  - When an error occurs on an Axis variable in AxesGroup, all Axis variables can be reset using this instruction. A timeout during the reset process will trigger an error. Once the reset process is complete, AxesGroup is reinitialized and its coordinate systems (ACS/PCS) are updated. The AxesGroup state is changed from GroupErrorStop to GroupStandby. If the axis in AxesGroup is already enabled, the state of the Axis variable is changed to Synchronized_Motion. If the axis in AxesGroup are not yet enabled, the Axis variable remains in the Disabled state.
  - It is not allowed to use this instruction when the axis group is in the GroupDisabled status; otherwise, an error is reported. When the axis group is not in the GroupErrorStop or GroupDisabled status, Done is directly set to TRUE using this instruction, and the status of AxesGroup remains unchanged.
- **Sequence diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates that the instruction has been executed successfully, and the parameter Busy indicates that the current function block is being executed.

  After this instruction is called, the output parameter Busy becomes TRUE on a rising edge of Execute. After successful reset, the output parameter Done is set to TRUE.

  The MC_GroupReset instruction is called when the axis group fails, and the faults of the axis group are successfully reset.



- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.14 MC_GroupSetOverride

The axis group override instruction sets a scaling factor for the axis group's coordinated motion within the coordinate system. The override parameter acts as a factor on the velocity, acceleration, deceleration, and jerk of each function block in the axis group simultaneously.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GroupSetOverride	Axis group override setting	MC_GroupSetOverride_Instance_0 MC_GroupSetOverride AxesGroup Enabled Enable Busy VelFactor Error AccFactor ErrorID JerkFactor	MC_GroupSetOverride_Instance_0(AxesGroup := , Enable := , VelFactor := , AccFactor := , JerkFactor := , Enabled =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: The instruction is enabled. FALSE: The instruction is not enabled. When Enable = FALSE, the velocity, acceleration, and jerk overrides revert to their default values (1).
VelFactor	Velocity override factor	LREAL	[0.0,100.0]	100.0	Scaling factor for velocity, expressed as a percentage. If the factor is greater than 100, it is corrected to 100. If the factor is less than 0, it is corrected to 1. If the factor equals 0, no correction is applied.
AccFactor	Acceleration factor	LREAL	[1.0,100.0]	100.0	Factor relative to the maximum acceleration of the axis group. If the factor is greater than 100, it is corrected to 100. If the factor is less than 1, it is corrected to 1. If the factor equals 1, no correction is applied.
JerkFactor	Jerk factor	LREAL	[0.0,100.0]	100.0	Factor relative to the maximum jerk of the axis group. For the non-trapezoidal motion profiling, the factor will be corrected to 100 if it is greater than 100. If the factor is less than or equal to 0, it will be corrected to 1.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Enabled	Instruction executed successfully	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the override factor is successfully set.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the function block is not completed.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

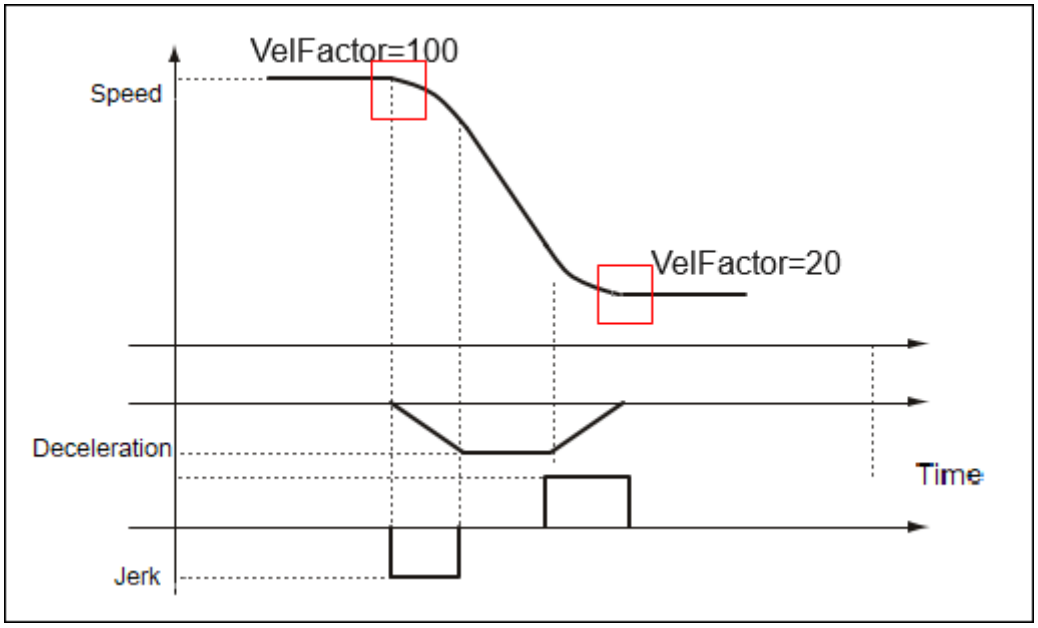
- **Parameter correction**

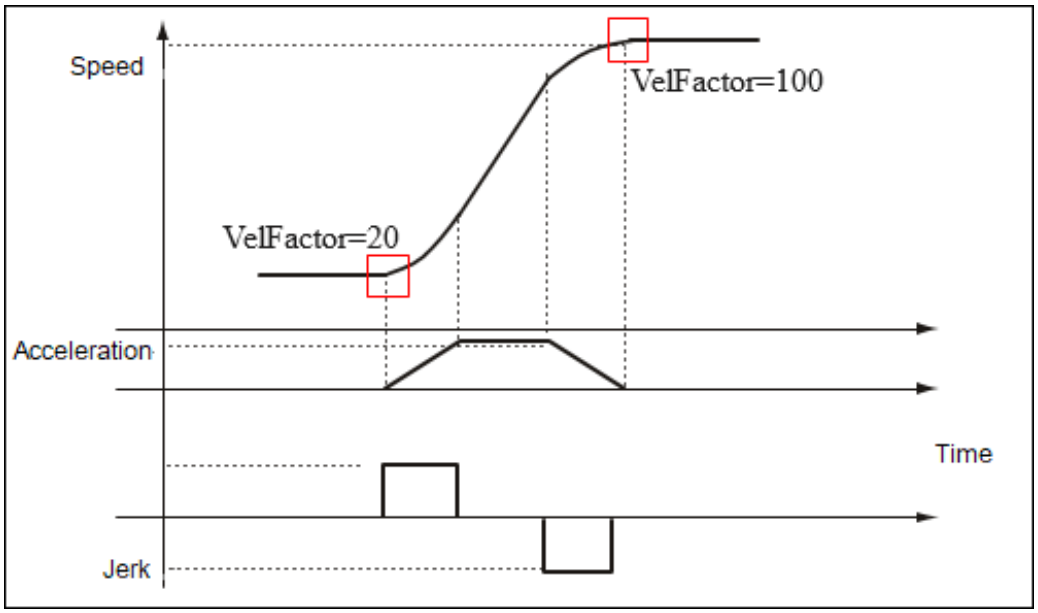
Input Variable	Name	Correction Description for Values Greater Than 1 or Backend Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
VelFactor	Velocity factor	If the velocity factor is greater than 100, it is automatically corrected to 100.	If the velocity factor is less than 0, it is automatically corrected to 1.	No correction.
AccFactor	Acceleration factor	If the acceleration factor is greater than 100, it is automatically corrected to 100.	If the acceleration factor is less than 1, it is automatically corrected to 1 for operation.	If the acceleration factor is equal to 1, no correction is applied.
JerkFactor	Jerk factor	If the jerk factor is greater than 100 in the non-trapezoidal motion profiling, it is automatically corrected to 100.	If the jerk factor is less than 0 in the non-trapezoidal motion profiling, it is automatically corrected to 1 for operation.	If the jerk factor is equal to 0 in the non-trapezoidal motion profiling, it is automatically corrected to 1 for operation.

- **Function**

  This instruction is executed on the rising edge of Enable to set the rate for the AxesGroup during coordinate system motion.

  - The default value for VelFactor, AccFactor, and JerkFactor is 100.0. Both VelFactor and JerkFactor can be set to 0.0.
  - The unit for the override factor is [%], "100" represents 100%, and "50" represents 50%.
  - When Enable is set to FALSE, VelFactor, AccFactor, and JerkFactor revert to their default values.
  - For VelFactor = 0, the AxesGroup will run at a speed of 0, but the AxesGroup state will not return to GroupStandby. Instead, AxesGroup will continue running at a speed of 0.
  - This instruction does not interrupt the current motion of AxesGroup. It only affects the current motion speed of AxesGroup.
  - VelFactor can respond in real time and affect the current motion. When the instruction output Enabled becomes TRUE, it indicates that the velocity override factor has been successfully set, but it takes some time for the actual override to respond to the target value. AccFactor and JerkFactor reflect the override adjustment response time and are effective only during changes in the motion override.

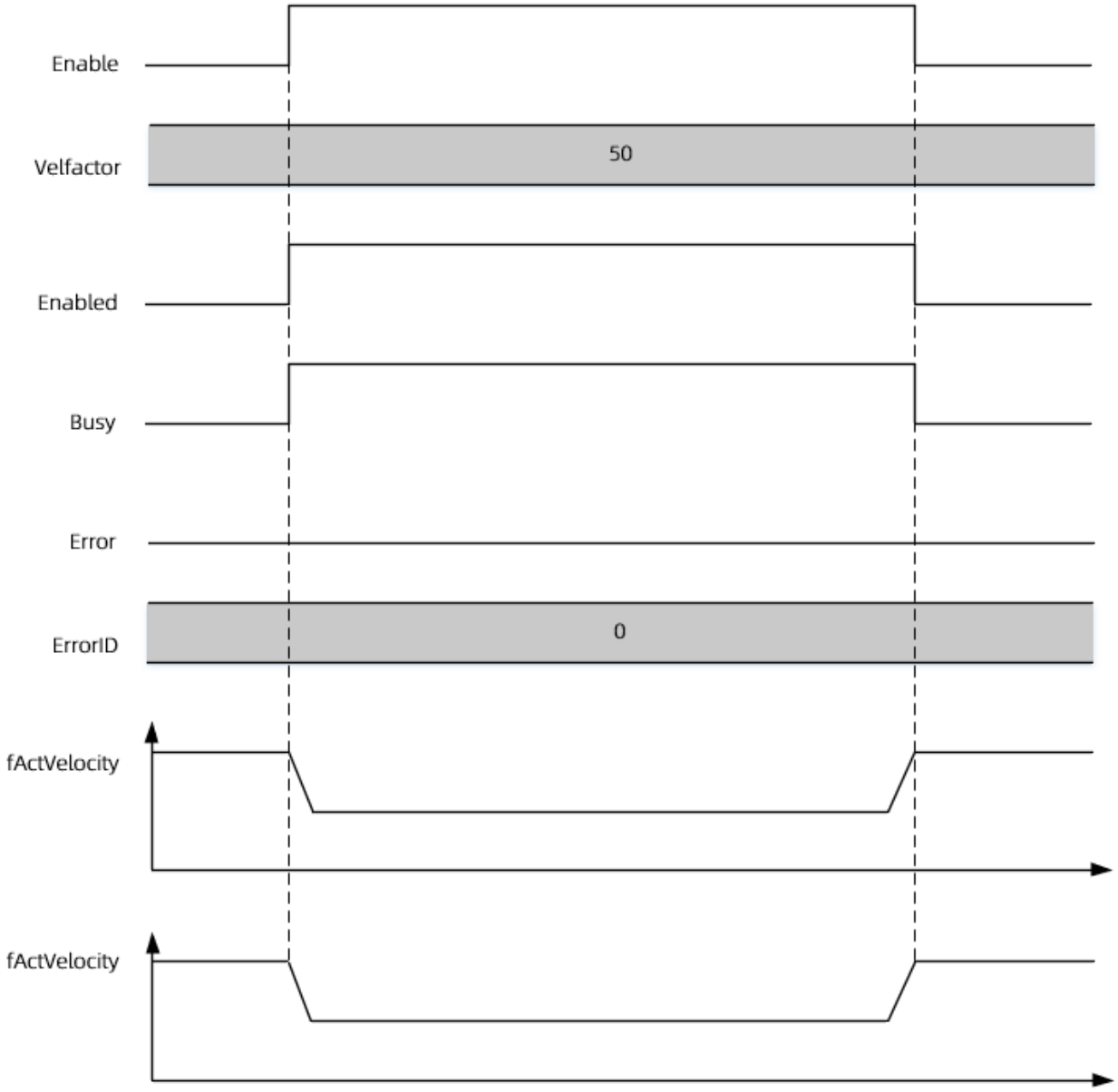




  - Deceleration factor and jerk factor are effective for both PCS/ACS spatial motion instructions. Rate adjustments are made based on the ratio of the maximum planning parameters for the axis group in the current motion space (working space/axis space). However, these adjustments do not guarantee that the planning parameters will be exactly equal.
  - The trajectory planning curve type of AxesGroup is set according to requirements. If the planning curve type is a trapezoidal curve, the JerkFactor variable does not take effect.

- **Re-triggering the instruction**
  - When the input is an "Enable-type" instruction, the instruction will not be re-triggered.
- **Precautions**
  - This instruction can only be initiated when AxesGroup is in the GroupMoving or GroupStandby state. Otherwise, the instruction will return an error.
  - When AxesGroup is controlled by the MC_GroupStop, MC_GroupHalt, or MC_GroupInterrupt instructions, this instruction cannot be initiated. Otherwise, the instruction will return an error.
- **Sequence diagram**

  The input parameter "Enable" must have a valid level signal. When the output parameter "Enabled" is TRUE, it indicates that the target override has been successfully set. When the output parameter "Busy" is TRUE, it indicates that the current function block is being executed.



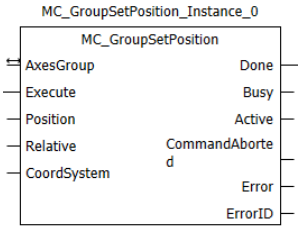
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.15 MC_GroupSetPosition

This instruction can be used to set the current position of each axis in the axis group in the given coordinate system.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GroupSetPosition	Set axis group current position		MC_GroupSetPosition_Instance_0(AxesGroup := , Execute := , Position := , Relative := , CoordSystem := , Done => , Busy => , Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
Position	Position	ARRAY [1..GROUP_DEF.AXIS_NUM] OF LREAL	-	[0,0…]	Position in the corresponding coordinate type.
Relative	Enabling relative coordinates	BOOL	TRUE, FALSE	FALSE	RELATIVE = TRUE. ABSOLUTE = FALSE.
CoordSystem	Coordinate system type	18.3.3.2.4.19 MC_COORDSYSTEM: Coordinate System Enumeration	ACS, PCS	ACS	Select the application coordinate system type.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Instruction has controlled the axis group	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis group is being controlled.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**
  - With this instruction, you can set the current position of each axis in the specified AxesGroup.
  - You can set the current position of each Axis variable in the coordinate system specified by AxesGroup under ACS or PCS coordinate system, update the axis offset or part coordinate system without changing the type of the part coordinate system, and support the position setting in a dynamic coordinate system.
  - With the Relative input, you can select the method to set the position by setting the input variable Relative of this instruction:

    If the Relative input is FALSE, it is an absolute coordinate system. After the instruction is used, the actual position of each Axis variable in AxesGroup equals the value set by the input parameter Position of this instruction.

    If the Relative input is TRUE, it is a relative coordinate system. After the instruction is used, the actual position of each Axis variable in AxesGroup equals the value set by the input parameter Position for this instruction plus the actual position of each Axis variable in AxesGroup before this instruction is used.

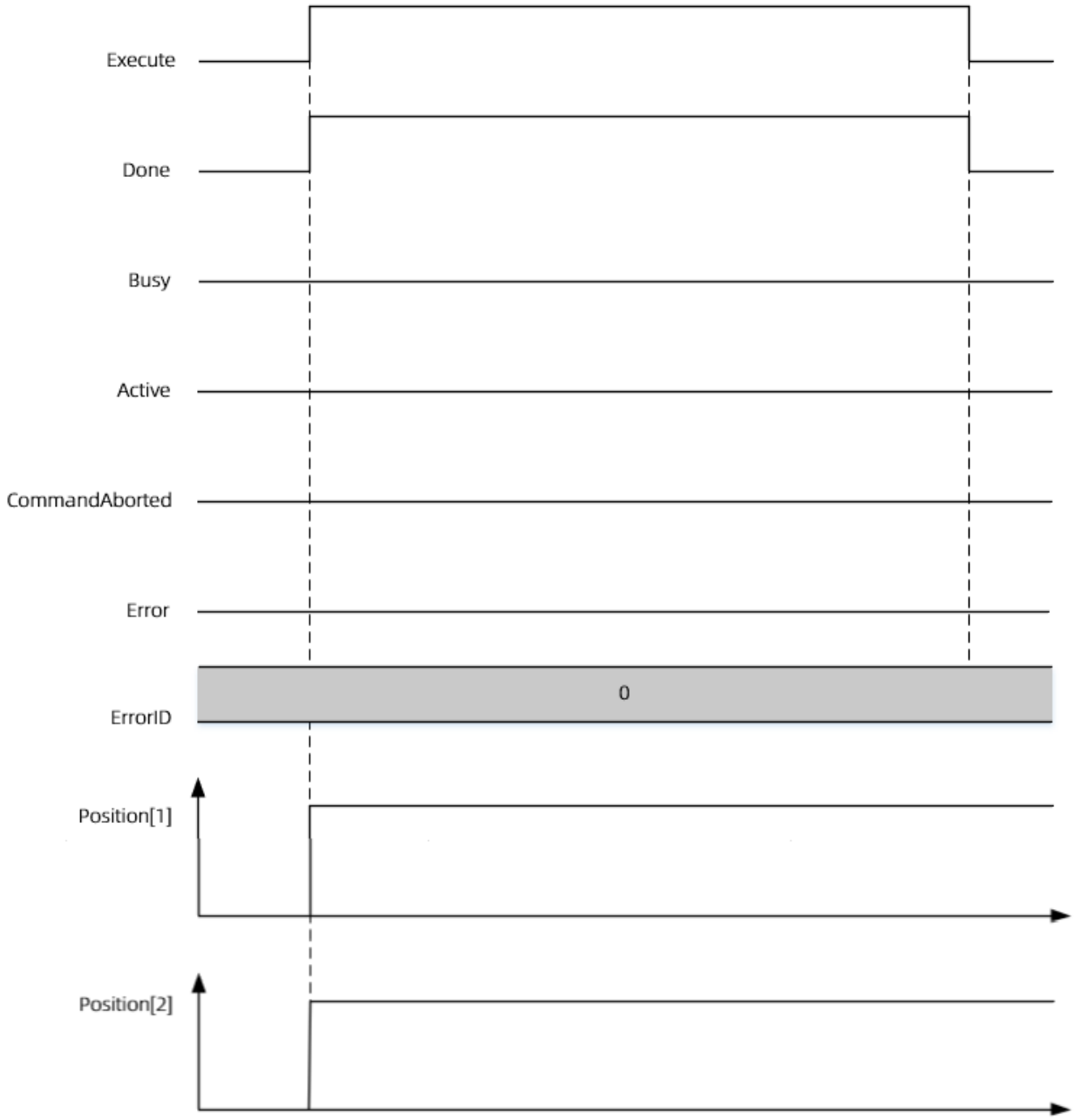
- **Precautions**
  - When the instruction is used, the axis group specified by AxesGroup must be in the GroupStandby, GroupMoving and axis group idle status. There is no state requirement for the Axis variable in AxesGroup.

- Setting the position under the ACS will affect the position of the axis group under the PCS. Setting the position under the PCS will not affect the position of the axis group under the ACS.
- Do not set the coordinate system during continuous trajectory module execution. Otherwise, the function block reports an error, and the axis group status remains unchanged.
- Before the axis group coordinate system is set in Setting, but not activated (Active), it is prohibited to set the coordinate system. Otherwise, the function block reports an error.

- **Sequence diagram**

  When the input parameter "Execute" is TRUE, the output parameter "Busy" becomes TRUE, indicating that the current function block is executing. The output parameter "Active" becomes TRUE, indicating that the instruction has controlled the axis group. After the current position changes, the output parameter "Done" becomes TRUE, indicating that the instruction has completed its execution.

  When the input parameter Execute is set to FALSE, the output parameter Done shows the value FALSE.



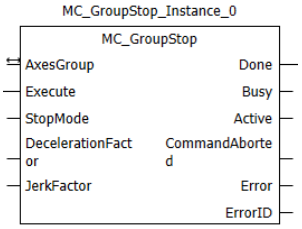
- **Faults**

For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.16 MC_GroupStop

This instruction makes the motion stop immediately. It is generally used for stopping an axis group under an abnormal situation, and the current motion cannot be recovered.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GroupStop	Axis group emergency stop		MC_GroupStop_Instance_0(AxesGroup := , Execute := , StopMode := , DecelerationFactor := , JerkFactor := , Done => , Busy => , Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
StopMode	Stop mode	18.3.3.2.4.12 MC_GROUP_STOP_MODE: Stop Mode Enumeration	0: DEC_STOP 1: IMMED_STOP	0	DEC_STOP: Decelerate to stop. IMMED_STOP: Stop immediately.
DecelerationFactor	Deceleration factor	LREAL	[0.01,1.0]	0.5	Ratio relative to the maximum deceleration of the axis group
JerkFactor	Jerk factor	LREAL	[0.0,1.0]	0.5	Factor relative to the maximum jerk of the axis group.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Instruction has controlled the axis group	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

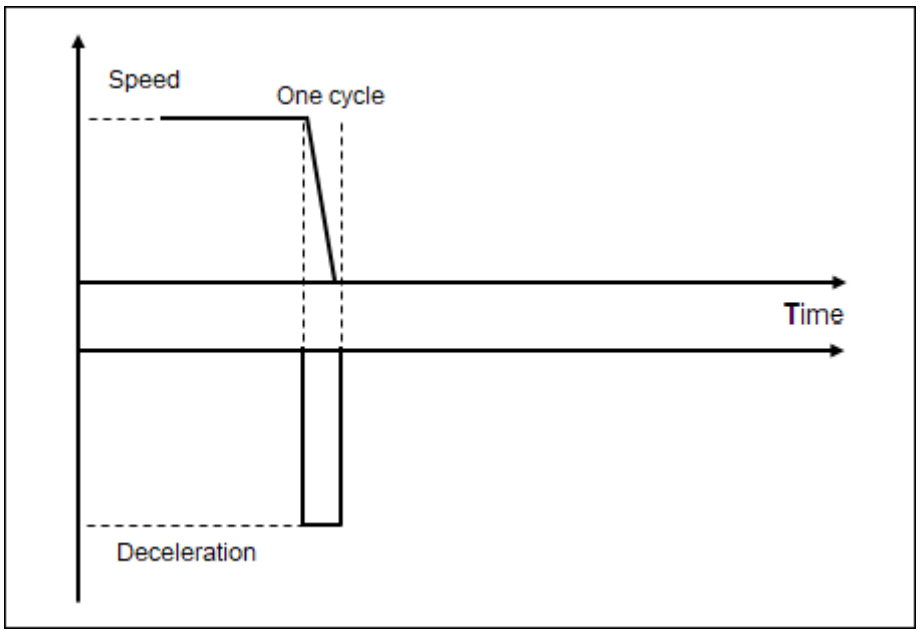
  This instruction is executed on the rising edge of Execute and is used for emergency stop during abnormal operation of the axis group (AxesGroup).

  - **StopMode**
    - This instruction sets the emergency stop method for abnormal operation of the axis group (AxesGroup) by inputting the variable StopMode.

      In StopMode = 0, DEC_STOP, the deceleration stop is performed on each Axis variable in AxesGroup according to the parameters set by the instruction. After the deceleration stop is

completed, the Done output of the instruction is set to TRUE, and the axis group state is switched to GroupStopping.

In StopMode = 1, IMMED_STOP, the speed of each Axis variable in AxesGroup is reduced to 0 within one cycle. The Done output of the instruction is set to TRUE, and the axis group state is switched to GroupStopping.

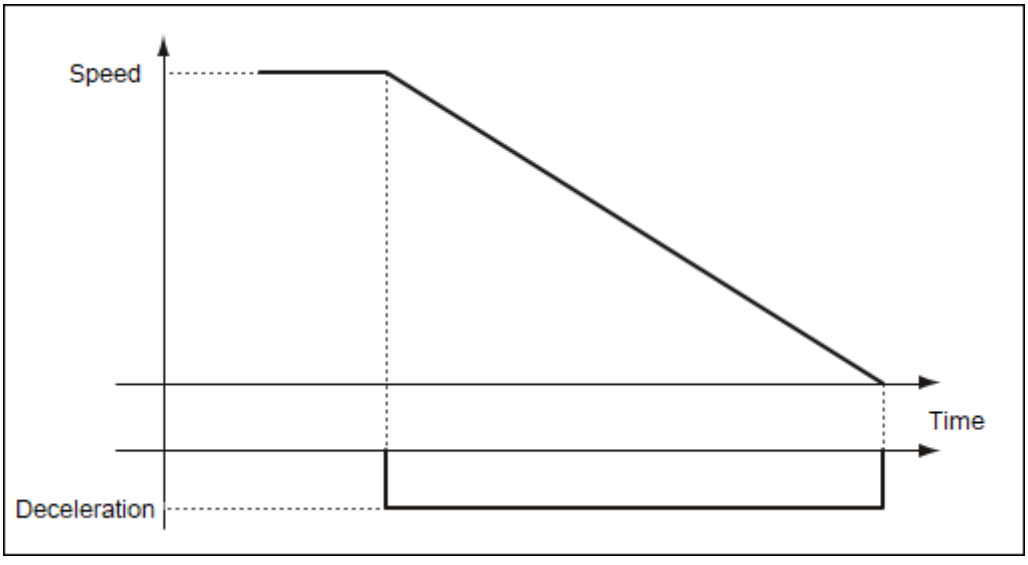


◦ **DecelerationFactor, JerkFactor**

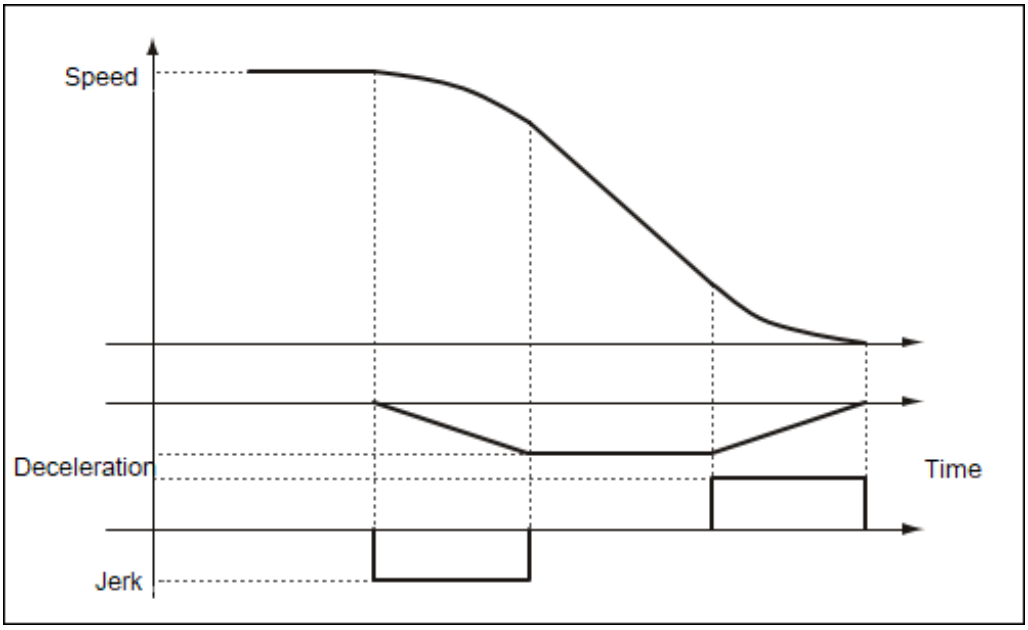
- The deceleration and jerk during the execution of this instruction are determined by the input variables DecelerationFactor and JerkFactor.
- In StopMode = 0, DEC_STOP, the deceleration stop is performed on each Axis variable in AxesGroup according to the product of the maximum axis acceleration limited by software and DecelerationFactor, and the product of the maximum axis jerk limited by software and JerkFactor.
- JerkFactor

  The jerk factor parameter is invalid for the T-profile, but valid for the S-profile.

  For JerkFactor = 0, the velocity curve and deceleration curve are shown as follows.

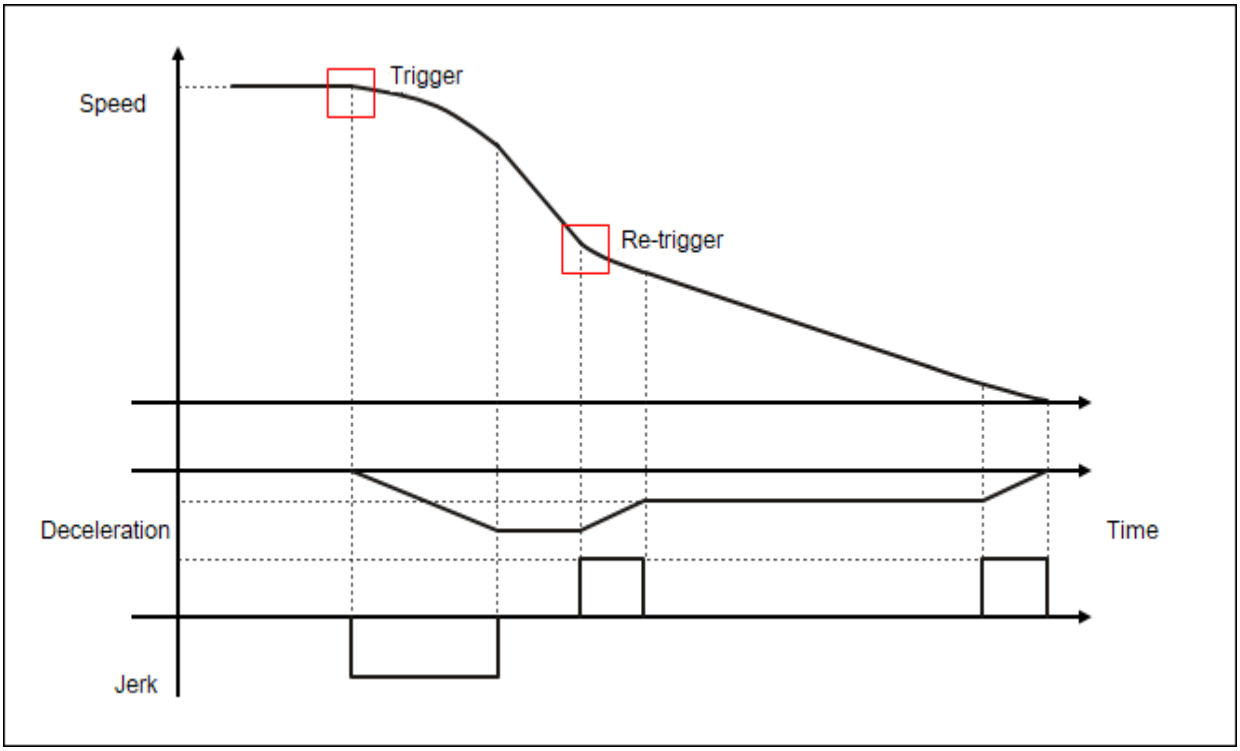


  For JerkFactor ≠ 0, the velocity curve, deceleration curve, and jerk curve are shown as follows.



  - StopMode is set to DEC_STOP by default.
  - When StopMode is set to IMMED_STOP, DecelerationFactor and JerkFactor are invalid.
  - When this instruction is used, the axis group must be in the GroupStandby, GroupMoving, or GroupHoming state. Attempting to start this instruction in any other state will result in an error.
  - This instruction can interrupt the motion instruction currently being executed by AxesGroup and set the axis group to GroupStopping. **In this status, no motion instructions are allowed to execute for AxesGroup. Otherwise, an error will be generated for the corresponding motion instruction**.
  - If the instruction is used during the homing process of AxesGroup, StopMode, DecelerationFactor, and JerkFactor are all invalid. A deceleration stop along the current path is not guaranteed. Instead, AxesGroup will decelerate to a stop according to the servo stop method.
  - **When AxesGroup is in the GroupStopping state, do not execute other motion instructions. Otherwise, an error will occur. However, for an MC_GroupStop instruction instances that have triggered the axis group to enter the GroupStopping state, it is permitted to re-trigger the instruction after modifying the parameters. (For example, start stopping in DEC_STOP mode, then change the mode to IMMED_STOP during stop and re-trigger the instruction.)**

- **Re-triggering the instruction**
  - This instruction can be re-executed. If the instruction is re-triggered after the parameter modification, execute the instruction according to the modified parameters. If the modified parameters are invalid, the instruction will return an error. The speed of the axis group (AxesGroup) will quickly reduce to 0, and the axis group state will change to GroupErrorStop.



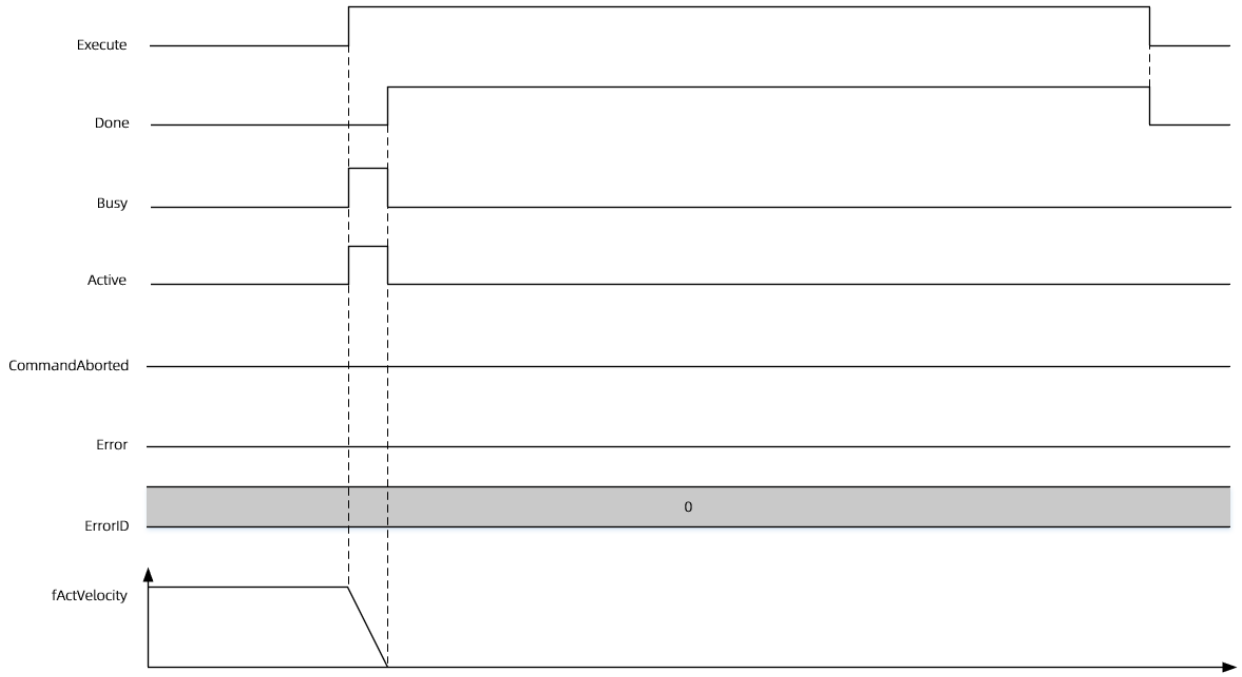
- **Precautions**
  - The trajectory planning curve type of AxesGroup must be set as required. If the planning curve type is set to a trapezoidal curve, the input variable JerkFactor for this instruction becomes invalid, and the instruction executes a deceleration to a stop according to DecelerationFactor. If the planning curve type is set to a cubic curve or a septic curve, the instruction executes a deceleration to a stop according to DecelerationFactor and JerkFactor.
  - Whether the planning curve type of each Axis variable in AxesGroup is set to a trapezoidal curve or an S-curve does not affect the planning method of AxesGroup.
- **Sequence diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates the instruction has been executed successfully; the parameter Busy indicates that the current function block is being executed; the parameter CommandAborted indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

  The parameter Busy shows the value TRUE on a rising edge of Execute. In the next cycle, the output parameter Active shows the value TRUE.

  When the velocity of all axes reaches "0", "Done" becomes TRUE.

  When this instruction is aborted using another instruction, CommandAborted becomes TRUE, and "Busy" and "Active" become FALSE.



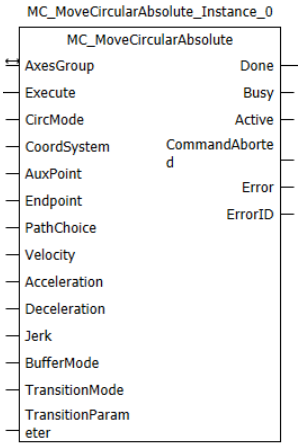
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.17 MC_MoveCircularAbsolute

This enables the arc interpolation motion of the axis group from the current TCP point to the absolute position point under the given coordinate system.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_MoveCircularAbsolute	Circular absolute motion		MC_MoveCircularAbsolute_Instance_0(AxesGroup := , Execute := , CircMode := , CoordSystem := , AuxPoint := , Endpoint := , PathChoice := , Velocity := , Acceleration :=, Deceleration :=, Jerk := , BufferMode := , TransitionMode := , TransitionParameter := , Done => , Busy => , Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
CircMode	Circular mode	18.3.3.2.4.18 MC_CIRCMODE: Arc Mode Enumeration	0, 1, 2	0	0: Three-point arc. 1: Center-point arc. 2: Radius-defined arc.
CoordSystem	Coordinate system type	18.3.3.2.4.19 MC_COORDSYSTEM: Coordinate System Enumeration	ACS, PCS	PCS	Currently only supports PCS.
AuxPoint	Auxiliary point	ARRAY [1..GROUP_DEF.AXIS_NUM] OF LREAL	-	[0,0,...]	For the auxiliary point position of each dimension in the specified coordinate system, see Axis group type description.
EndPoint	End point	ARRAY [1..GROUP_DEF.AXIS_NUM] OF LREAL	-	[0,0,...]	For the end position in each dimension in the specified coordinate system, see Axis group type description.
PathChoice	Path choice	18.3.3.2.4.17 MC_CIRC_PATHCHOICE: Arc Path Choice Enumeration	0, 1	0	This parameter is valid when CircMode is set to CENTER_ARC or RADIUS_ARC arc mode.
Velocity	Velocity	LREAL	> 0	-	Maximum linear velocity. This value may not necessarily be reached.
Acceleration	Acceleration	LREAL	> 0	-	Maximum linear acceleration. This value may not necessarily be reached.
Deceleration	Deceleration	LREAL	> 0	-	Maximum linear deceleration. This value may not necessarily be reached.
Jerk	Jerk	LREAL	>=0	-	Maximum linear jerk. This value may not necessarily be reached.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting 1: mcBuffered 2: mcBlendingLow 3: mcBlendingPrevious 4: mcBlendingNext 5: mcBlendingHigh	0	0: mcAborting (Aborting). 1: mcBuffered (Buffered). 2: mcBlendingLow (Blending with the lowest velocity of both FBs). 3: mcBlendingPrevious (Blending with the velocity of the first FB). 4: mcBlendingNext (Blending with velocity of the second FB). 5: mcBlendingHigh (Blending with the highest velocity of both FBs). Note: Buffer modes 2 to 5 currently have the same effect. All are blending modes. The behavior depends on the TransitionMode and TransitionParameter parameters.
TransitionMode	Transition mode	18.3.3.2.4.10 MC_TRANSITION_MODE: Transition Mode Enumeration	0: TMNone 1: TMStartVelocity 3: TMCornerDistance 4: TMMaxCornerDeviation 10: TMDefinedVelocity 11: TMResetAbort 12: TMDecAbort	0	Definition of the transition trajectory characteristic types. When BufferMode is set to MC_ABORTING, only the transition mode parameters TMNone, TMResetAbort, and TMDecAbort are allowed. When BufferMode is set to MC_BUFFERED, the transition mode is invalid. When BufferMode is a blending mode (2 to 5), TMResetAbort or TMDecAbort is not allowed.
TransitionParameter	Transition parameter	LREAL	Velocity [0.0,1.0] Distance [0.0,+∞)	1	Transition parameters vary with the transition mode.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Active	Instruction has controlled the axis group	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

  Position dimension

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity	Velocity	Any velocity that exceeds the motion parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	The velocity equal to 0 will be automatically corrected to 1e-5.
Acceleration	Acceleration	Any acceleration that exceeds the motion parameter limit will be automatically corrected to the acceleration limit.	Any acceleration less than 0 will be automatically corrected to 10% of the acceleration limit.	Any acceleration equal to 0 will be automatically corrected to 10% of the acceleration limit.
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the acceleration limit.	Any deceleration less than 0 will be automatically corrected to 10% of the acceleration limit.	Any deceleration equal to 0 will be automatically corrected to 10% of the acceleration limit.
Jerk	Jerk	For the non-trapezoidal motion profiling, any jerk that exceeds the software limit will be automatically corrected to the jerk limit.	For the non-trapezoidal motion profiling, any jerk that exceeds the software limit will be automatically corrected to 10% of the jerk limit.	For the non-trapezoidal motion profiling, any jerk that exceeds the software limit will be automatically corrected to 10% of the jerk limit.

  Percentage dimension

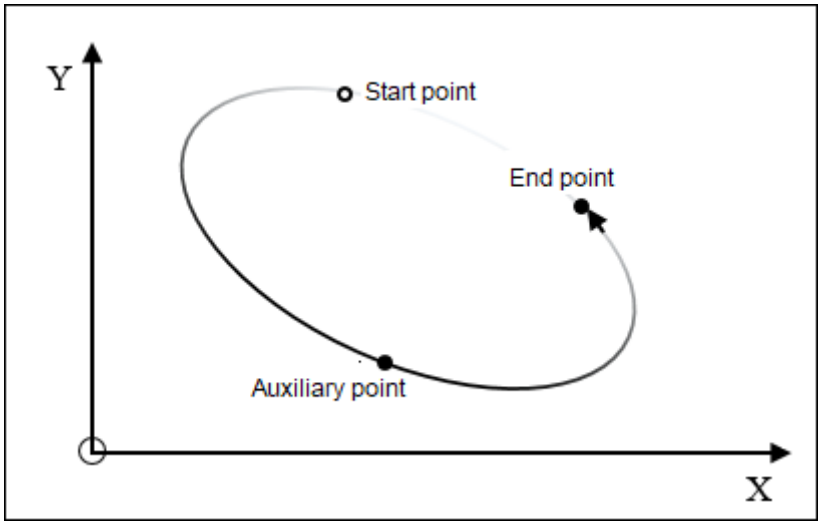
Input Variable	Name	Correction Description for Parameter Greater than 1	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity	Velocity	Any velocity exceeding 1 is automatically corrected to 1.	Any velocity less than 0 will be automatically corrected to 1e-5.	The velocity equal to 0 will be automatically corrected to 1e-5.
Acceleration	Acceleration	Any acceleration exceeding 1 is automatically corrected to 1.	Any acceleration less than 0 will be automatically corrected to 10%.	Any acceleration equal to 0 will be automatically corrected to 10%.
Deceleration	Deceleration	The value exceeding 1 is automatically corrected to 1.	Any deceleration less than 0 will be automatically corrected to 10%.	Any deceleration equal to 0 will be automatically corrected to 10%.
Jerk	Jerk	For the non-trapezoidal motion profiling, the value exceeding 1 is automatically corrected to 1.	For the non-trapezoidal motion profiling, the value exceeding 1 is automatically corrected to 10%.	For the non-trapezoidal motion profiling, the value exceeding 1 is automatically corrected to 10%.

- **Function**
  - This instruction is executed on the rising edge of Execute and performs an absolute arc motion, supporting planar or spatial arcs.
  - CircMode, CoordSystem, AuxPoint, EndPoint, PathChoice, Velocity, Acceleration, Deceleration, Jerk, BufferMode, TransitionMode, and TransitionParameter can be specified in the input variables.
  - For the instruction, blending modes 2, 3, 4, and 5 for BufferMode have the same behavior. The transition effect depends on the TransitionMode and TransitionParameter settings.
  - Supports the same-cycle triggering of instructions. The Done output signal of the instruction serves as the trigger signal for Execute.
  - **AuxPoint**
    - For the axis type corresponding to each position vector in the AuxPoint array, see Axis group type description. Only PCS coordinates are supported.
    - The ABC rotation axes of the coordinate system correspond sequentially to the ZYX-type Euler angles. If the configuration supports orientation motion, the orientation parameters set for the auxiliary points will not be included in the orientation trajectory planning.
  - **EndPoint**
    - For the axis type corresponding to each position vector in the EndPoint array, see Axis group type description. Only PCS coordinates are supported.
    - In absolute motion mode, the axis group uses the set values in the EndPoint array as the target position.
    - If the target position would exceed any joint axis limit after kinematic transformation into joint space, starting this instruction will cause an error.
    - If the configuration supports orientation motion, the axis group will move to the target orientation along the shortest path.

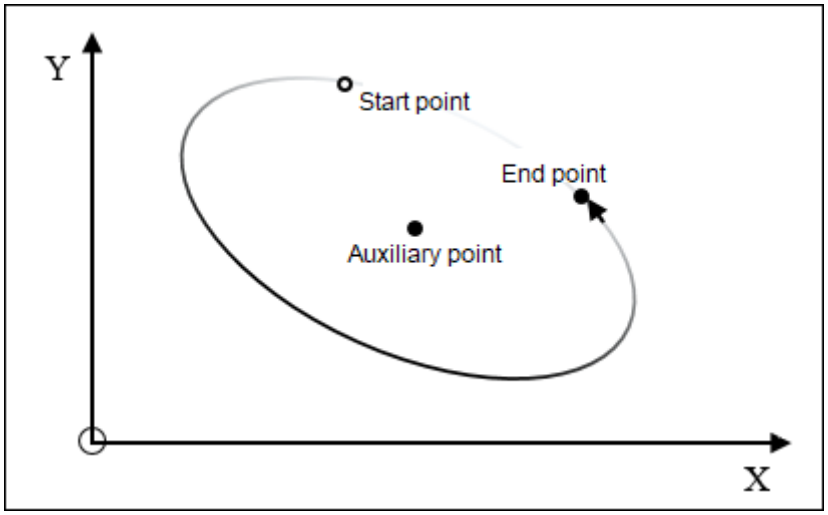
- The ABC rotation axes of the coordinate system correspond sequentially to the ZYX-type Euler angles. If the configuration supports orientation motion, the configured ABC absolute positions must be within ±180°, otherwise an error will be generated. Note that for the EVO500 series PLC (version V1.2), when the axis group type is set to "XYZA configuration", the A-axis functions as an auxiliary axis. Its movement is not limited to ±180 degrees. If it exceeds ±180 degrees, the axis continues moving according to the preset position.

◦ **CircMode**

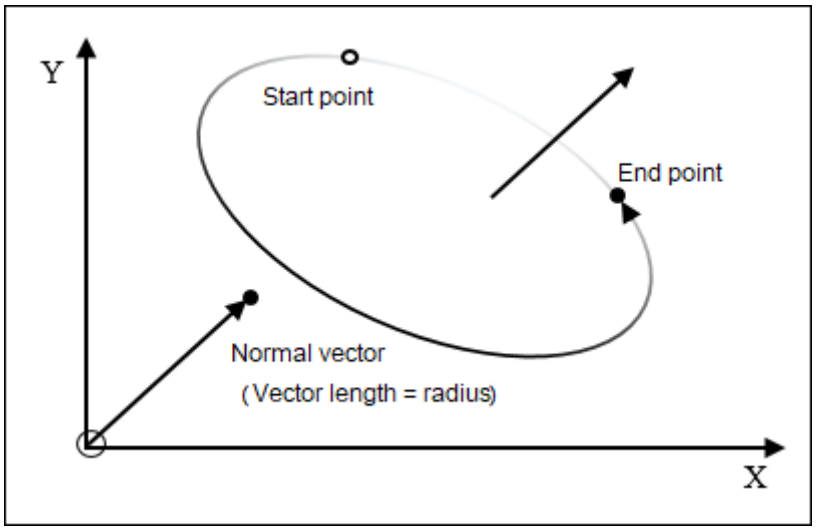
- BORDER_ARC three-point circular arc: a circular arc defined by three points—the current position of AxesGroup as start point, AuxPoint, and EndPoint. **If the three points are collinear or two of them coincide, the trajectory is a straight line connecting the start point and the end point**. If the three points are not collinear, the trajectory is a circular arc passing through the three points in sequence. The range of the arc angle is (0, 2π). The figure shows a three-point arc in space. It follows the right-hand rule, with the Z-axis pointing out of the screen.



- CENTER_ARC center circular arc: a circular arc is defined by the current position of AxesGroup as start point, AuxPoint as the center, and EndPoint. The arc is specified as either a long arc or short arc through PathChoice. The arc angle range is less than 2π, and the arc angle range is not equal to π. **When the three points are collinear, the trajectory is a straight line.** If the distance between the start point and the end point to the center of the circle deviates too much, an error will be reported. The distance from the start point to the center is named L1, and the distance from the end point to the center is named L2. The center will be automatically corrected if $|L1-L2| < 0.01 \times \min(L1,L2)$. Exceeding this deviation will trigger an error. (Due to floating-point arithmetic precision issues, no strict equality exists.) The figure shows the composition of a center-point arc in space, with axis Z facing off-screen.



- RADIUS_ARC radius arc: **the radius of the arc is determined by the vector length of** AuxPoint, and the vector direction determines the normal vector of the arc. PathChoice determines whether the arc is long or short. The arc angle range is less than 2π. When the start and end points are not perpendicular to the normal vector, an error will be reported if the deviation angle is too large. If the deviation is small (<10°), the direction of the normal vector will be corrected according to the parameter. If the distance from the endpoint exceeds twice the length of the normal vector and also exceeds one ten-thousandth of the length (that is, radius), an arc error is reported. When the deviation is within one ten-thousandth, the arc radius is corrected to a semicircle. An error is reported when the length of the normal vector or the distance between start and end points is less than the computational precision of 1e-6. **For an axis group configured with only two position axes, AuxPoint[0] indicates the radius of the circular arc. A positive value indicates rotation about the arc's normal axis in the positive direction (typically following the right-hand rule), while a negative value indicates rotation in the opposite (negative) direction.**



◦ **CoordSystem**

- Specify the coordinate system for circular absolute motion. Currently, only absolute circular motion under the PCS is supported.

◦ **Velocity constraint**

- The arc trajectory is bound by the maximum acceleration and deceleration parameters of the system. The centripetal acceleration is kept within limits by restricting the velocity along the trajectory segment. The velocity is constrained as follows:

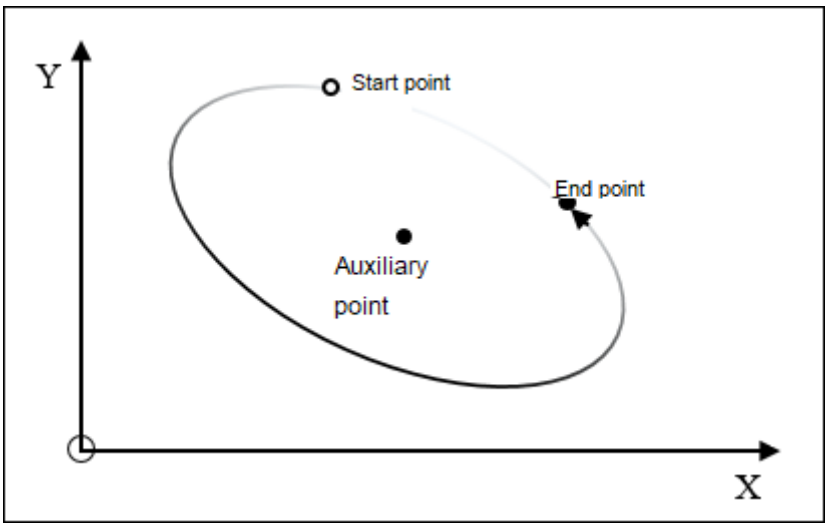
  Vmax = Sqrt(Amax *R*Ccoef)

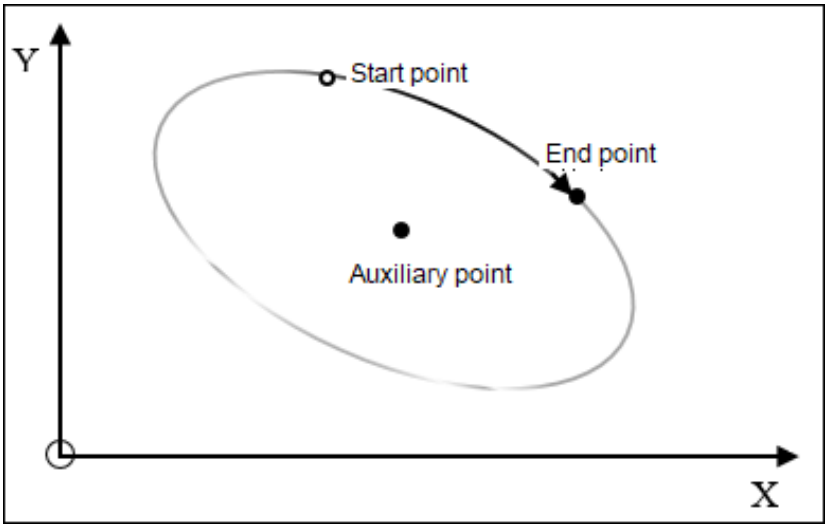
Here, Ccoef indicates the transition factor, R indicates the arc radius, and Amax indicates the maximum allowable acceleration of the system.

◦ **PathChoice**

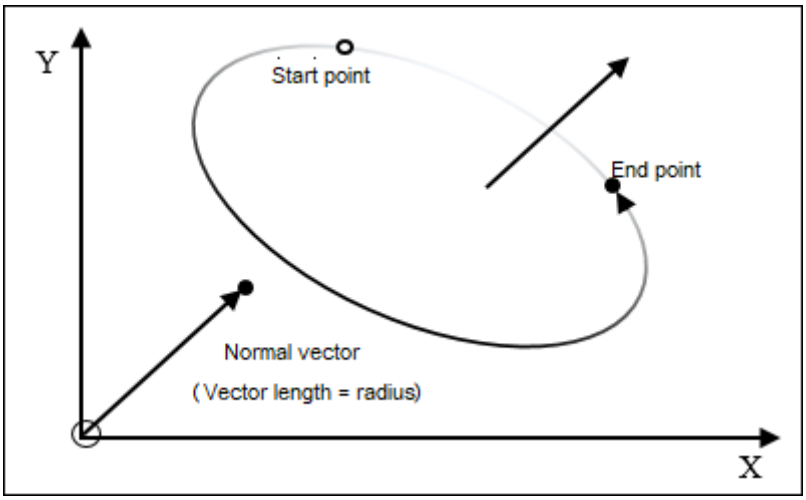
Depending on the CircMode, there are different choices for the arc direction and length.

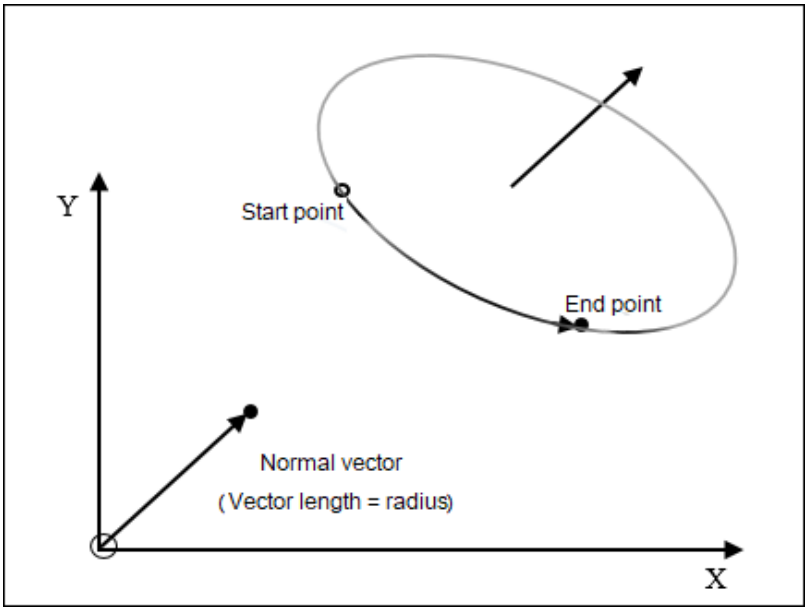
- CENTER_ARC center circular arc: In the figure below, the first setting is the LONG_ARC path, and the second setting is the SHORT_ARC path.





- RADIUS_ARC radius arc: In the figure below, the first setting is the LONG_ARC path, and the second setting is the SHORT_ARC path. .

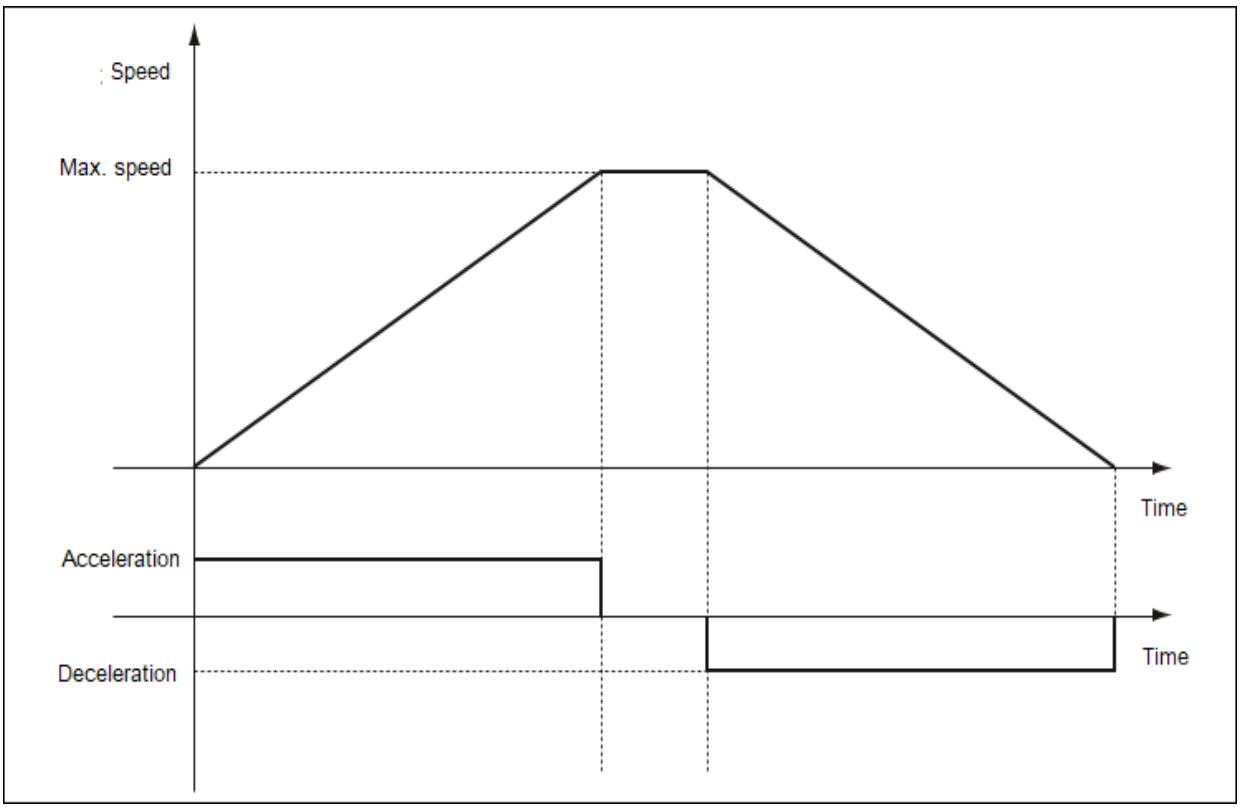




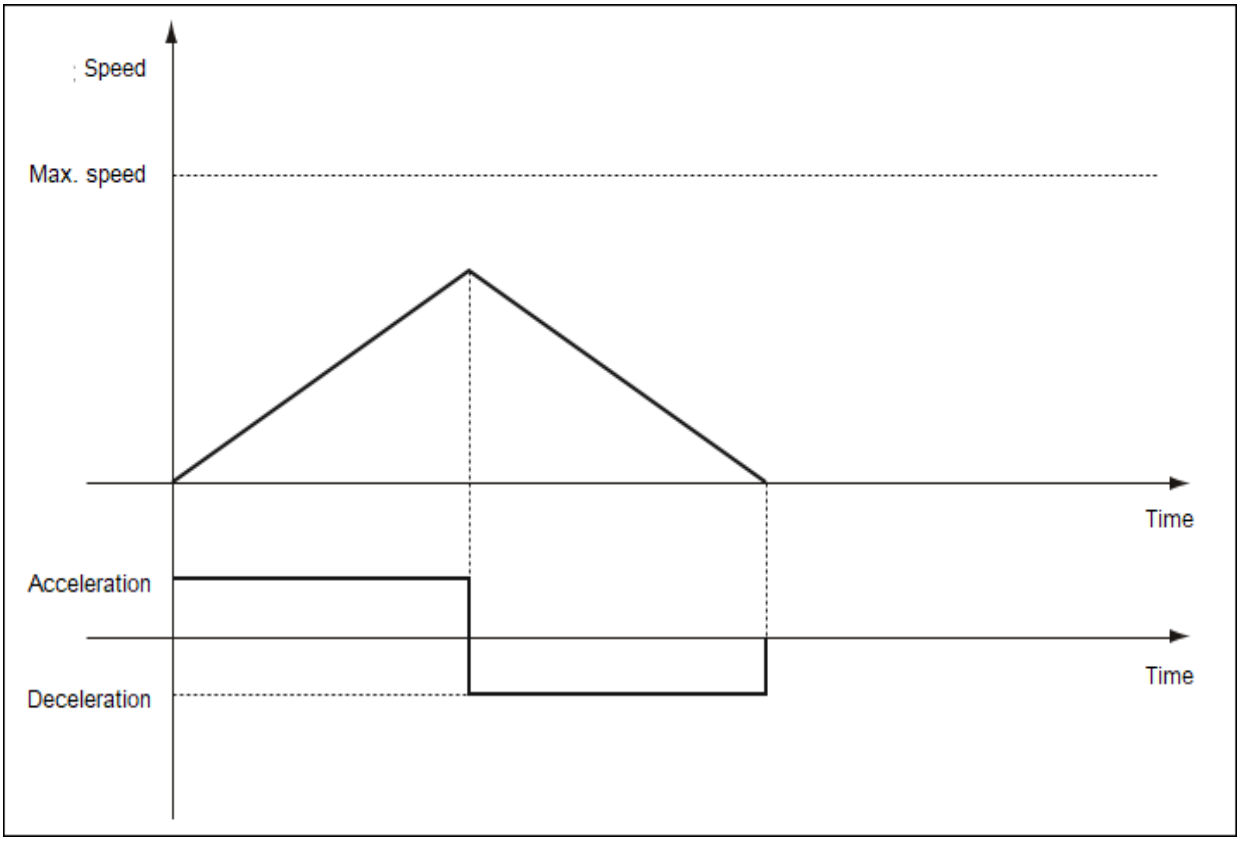
◦ **Velocity, Acceleration, Deceleration, Jerk**

- If the velocity of any Axis variable in AxesGroup exceeds the maximum axis velocity set by the software multiplied by the warning planning scaling factor, it will trigger an error-stop condition.
- The trajectory planning curve type of AxesGroup is set according to requirements. If the planning curve type is a trapezoidal curve, the Jerk variable does not take effect and is treated as 0. (The planning curve type of AxesGroup is specified in the axis group data structure.)
- For Jerk = 0 or Jerk ≠ 0, the velocity, acceleration, deceleration, and jerk curves are shown in the figure below.

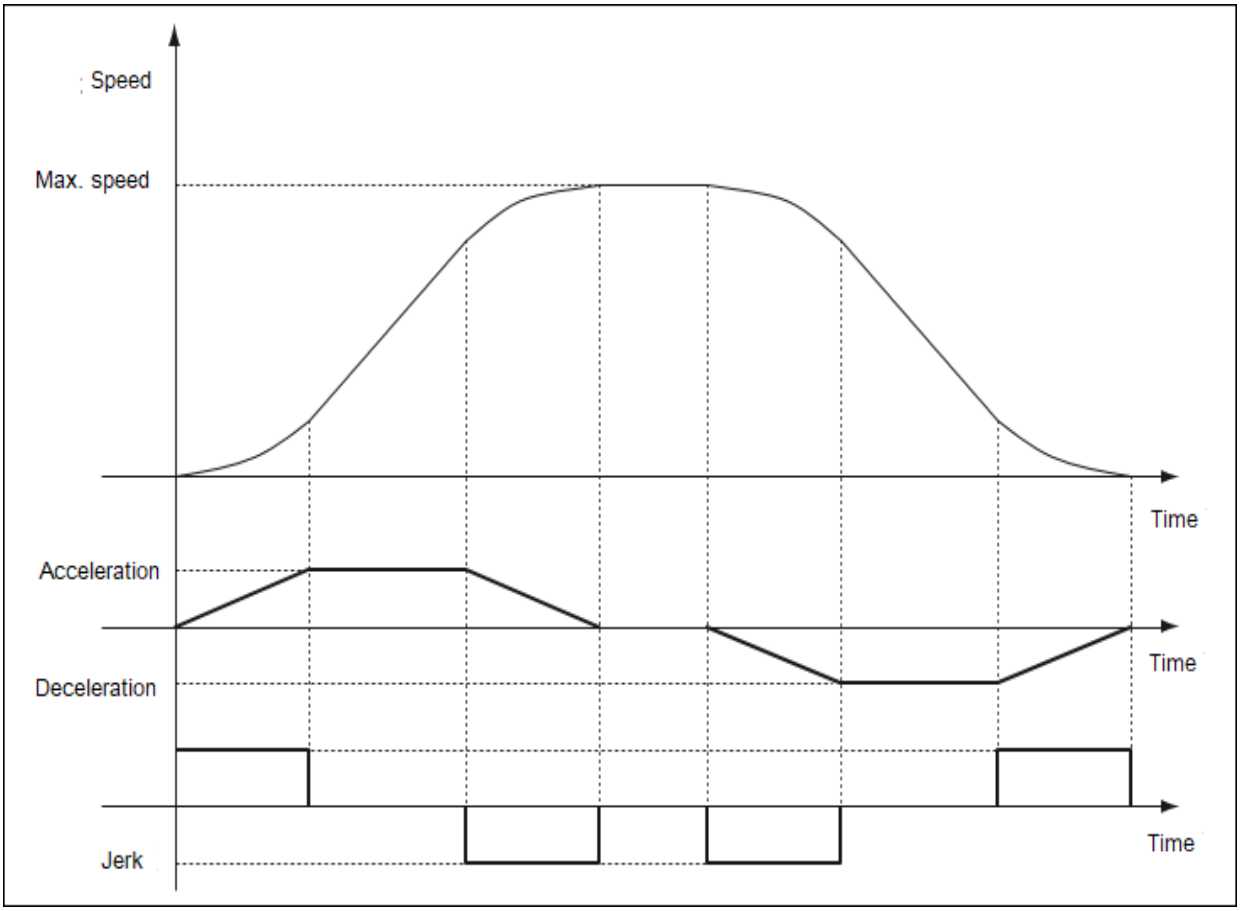
For Jerk = 0, the axis group can reach the maximum interpolation velocity.



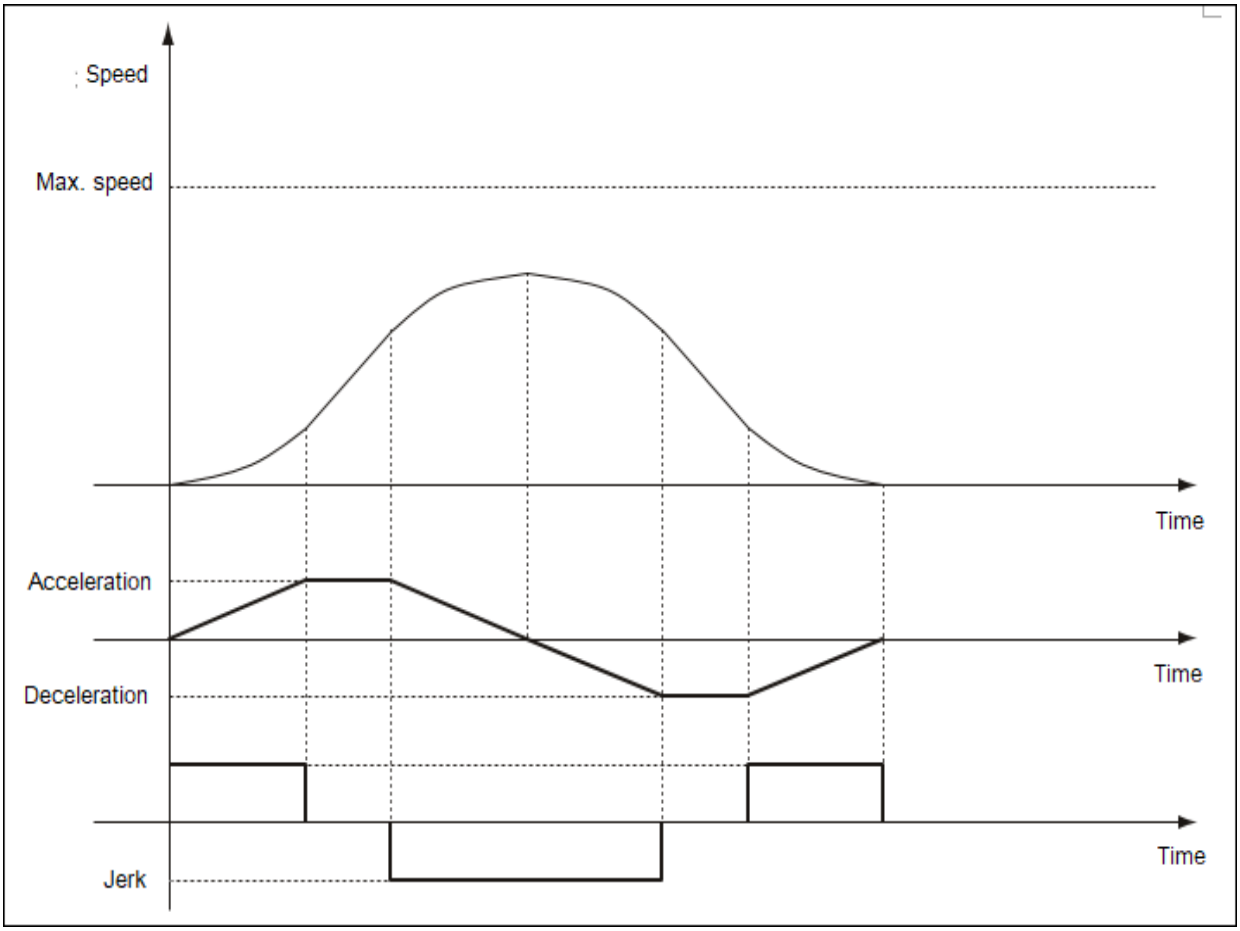
For Jerk = 0, the axis group cannot reach the maximum interpolation velocity due to too short travel distance.



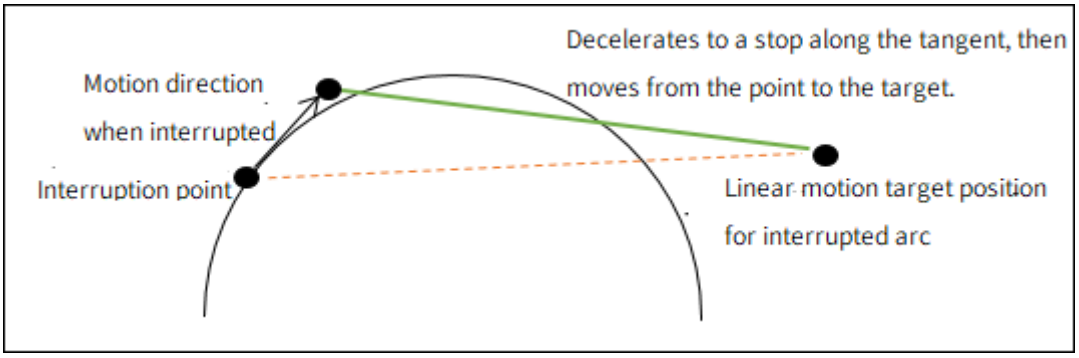
For Jerk ≠ 0, the axis group can reach the maximum interpolation velocity. (Non-trapezoidal planning curve)



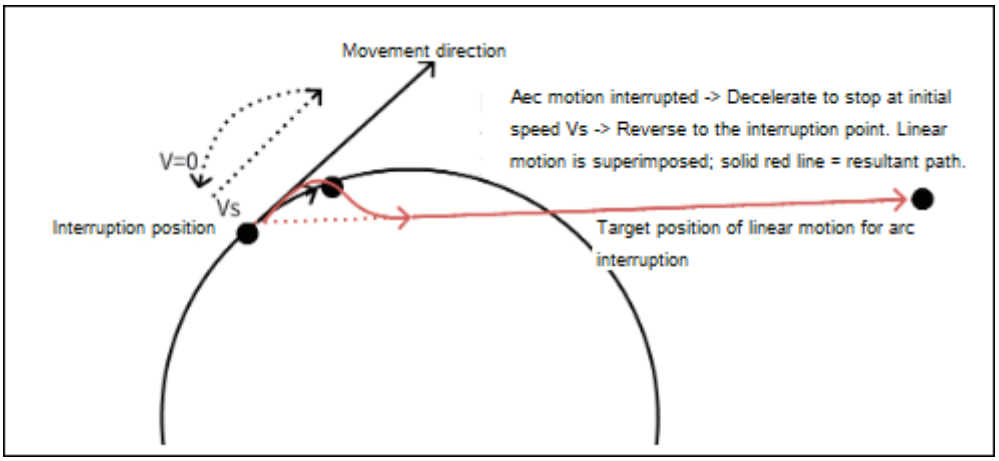
For Jerk ≠ 0, the axis group cannot reach the maximum interpolation velocity due to too short travel distance. (Non-trapezoidal planning curve)



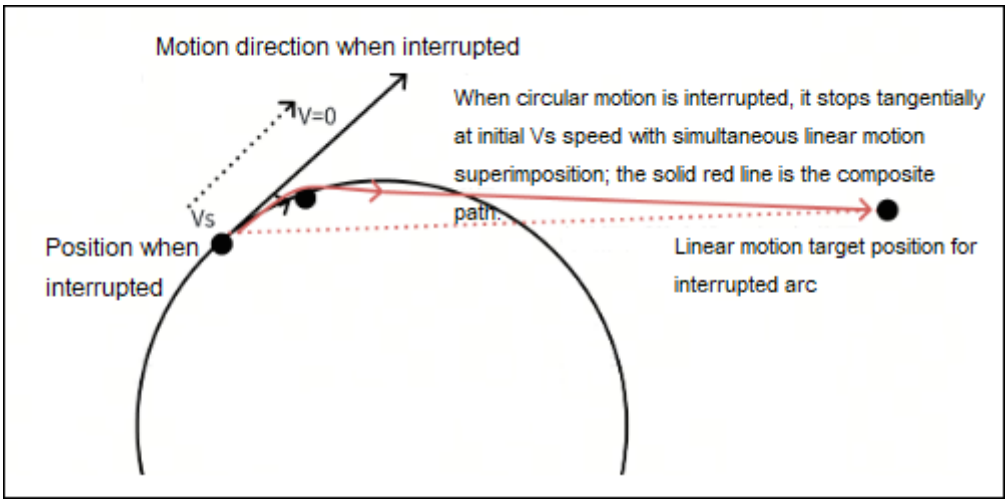
- **Re-triggering the instruction**
  - While the instruction is in motion, that is, when the instruction output parameters "Busy" and "Active" are both TRUE, this instruction cannot be re-triggered.
  - While the instruction is in buffer mode, that is, when Busy is TRUE and Active is FALSE, this instruction can be re-triggered after parameter modification. If the modified parameters are invalid, or if the buffered instruction data has already been used in motion calculations, retriggering the buffered instruction will cause an error. Then, the axis group will enter the GroupStop state, and the currently executing instruction will also report an error.
- **Multiple execution of motion instructions**
  - When this instruction is issued while another motion instruction is being executed, the new instruction can either be switched to immediately or buffered for later execution. Each axis group can buffer up to 8 instructions. In the case of multiple executions, the behavior of this instruction is determined by BufferMode.
  - BufferMode
    - Aborting: MC_ABORTING aborts the currently executing instruction and begins executing this instruction. The abort operation supports three modes, configurable via TransitionMode: TMNone, TMResetAbort, and TMDecAbort.
      TMNONE mode: During execution, the motion decelerates to a stop along the tangential direction from the point of interruption. The next motion segment then starts from this stop point, as shown in the figure below.



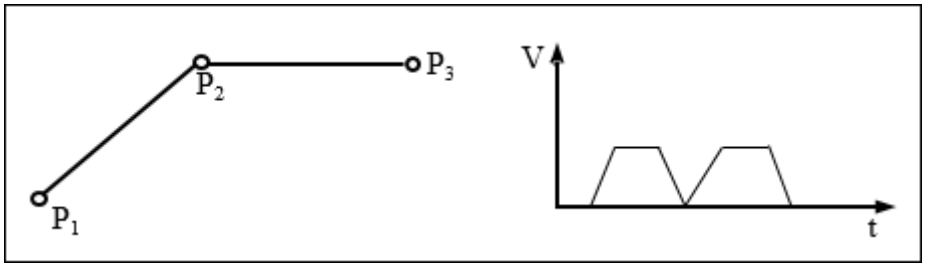
TMResetAbort mode: During execution, the axis decelerates along the tangential direction at the initial velocity of the interruption point, reverses its direction to return to that point, while simultaneously superimposing (blending) with the next motion segment, as shown in the figure below.



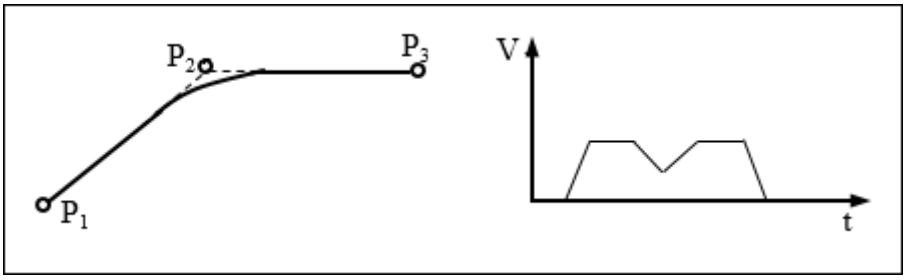
TMDecAbort mode: During execution, the axis decelerates to 0 along the tangential direction at the initial velocity of the interruption point, while simultaneously superimposing (blending) with the next motion segment, as shown in the figure below.



- Buffer: MC_BUFFERED, this instruction is buffered and executed after the current instruction is completed.



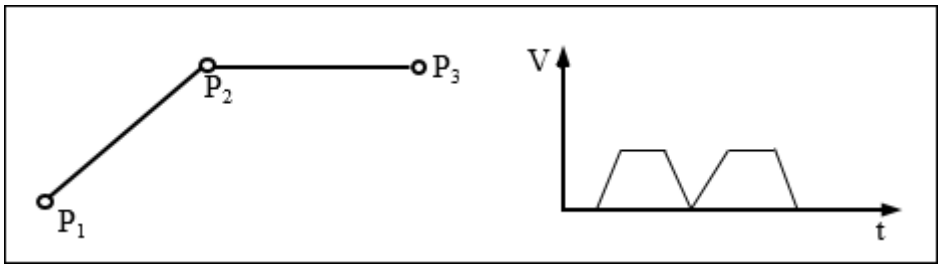
- Blending: MC_BLEND_XX, the velocities of the two adjacent instructions are blended in the specified manner. This manner is specified in TransitionMode.



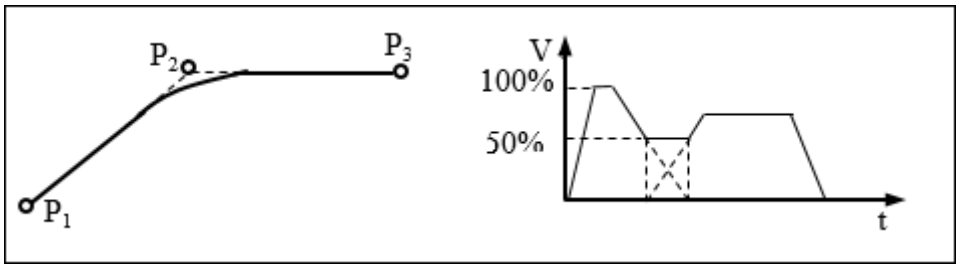
◦ TransitionMode

This variable specifies how to blend the velocities of two instructions when Blending is specified for the BufferMode.

- TMNone: Insert no transition curve. If TMNone is selected in a blending mode, the result is currently consistent with the transition between the velocities of the two instructions in buffered mode. Further extensions are planned.

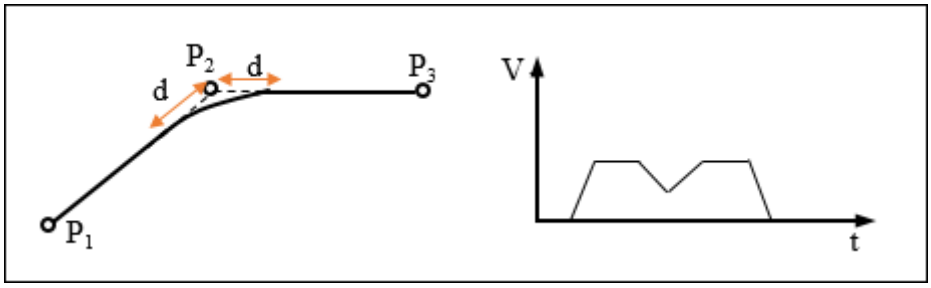


- TMStartVelocity: Transition with given start velocity. Blending transition at a certain ratio (0, 1] of the actual maximum velocity of the current motion control instruction. Effective for transitions between any motion control instructions.



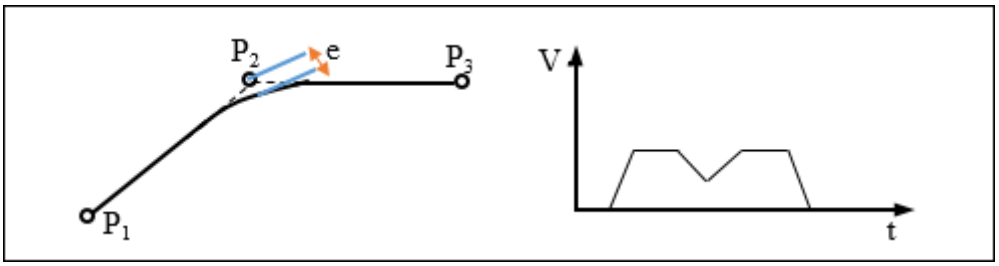
- TMCornerDistance: Corner transition mode In Cartesian space, this parameter specifies the distance before the end of the previous instruction, measured in the main dimension space. The transition trajectory is initiated at this distance to connect with the adjacent trajectory. The distance here is a generalized distance, which depends on the main dimension type setting of the axis group.
  When the primary dimension is set to position: the parameter is the displacement from the transition point to the end point.

  When the primary dimension is set to ratio: The parameter is the ratio (0 to 1) of the resultant distance from the transition point to the endpoint to the total resultant displacement.

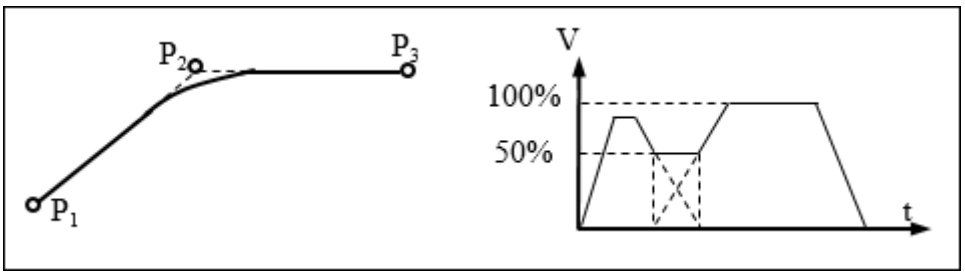


- TMMaxCornerDeviation: Transition mode using the maximum corner position deviation. In Cartesian space, this parameter specifies the maximum position deviation in the position subspace during the transition between adjacent instructions. An appropriate transition trajectory is constructed to connect the two adjacent paths, keeping the maximum position deviation within this

limit. This is only the position deviation, independent of the main unit. Effective when both adjacent instructions are Cartesian space instructions (LIN/CIRC). Otherwise, they are processed with transition disabled.



- TMDefinedVelocity: Transition with defined velocity. Blending transition at a certain ratio (0, 1] of the actual maximum velocity of the instruction following the current motion control instruction. Effective for transitions between any motion control instructions.



- **Precautions**
  - This instruction only supports absolute circular motion in PCS coordinates.
  - The dimensions of the axis group planning parameters (velocity, acceleration, deceleration, and jerk) depend on the primary dimension set during axis group initialization. Different primary dimension types result in different dimensions and meanings for these parameters, and the parameter units are consistent with the axis group settings for the user.
    - UNIT_TRAJECT position. The input planning parameters are processed according to workspace position planning parameters.
    - UNIT_PERCENT maximum planning parameter ratio. The input planning parameters are processed as a combined ratio coefficient based on workspace pose planning parameters.
  - When the current motion instruction is interrupted, circular motion constructs a transition trajectory in working space along the direction of the velocity at the breakpoint to ensure smooth connection with the subsequent trajectory. This ensures velocity continuity only. However, the acceleration continuity is not guaranteed. For details on the three transition modes supported for aborting, see the motion abort description.
  - The transition trajectory calculation for the aborting instruction is planned according to the acceleration segment parameters set in the instruction. The higher the acceleration segment parameter value, the shorter the transition trajectory and the faster the transition.
  - **If the instruction motion mode is set to an interruption mode and the deceleration distance is insufficient, the function block reports an error. But the axis group does not enter the GroupErrorStop state; the original motion continues to execute.**
  - This instruction can only be initiated when the axis group is in the GroupMoving or GroupStandby state. Attempting to start this instruction in any other state will result in an error.
  - During the execution of this instruction, it can be interrupted by other motion instructions, MC_GroupStop, or MC_GroupHalt.

- After the execution of this instruction, the axis group (AxesGroup) state is changed from GroupMoving to GroupStandby, or remains in the GroupMoving state.
- If the current motion is in a static coordinate system, the axis group is changed from the GroupMoving state to the GroupStandby state. If the current motion is in a dynamic coordinate system, the axis group remains in the GroupMoving state.
- If the motion includes orientation information, the orientation of the intermediate point for the BORDER_ARC three-point arc instruction is ignored; only the orientations of the start and end points are considered.

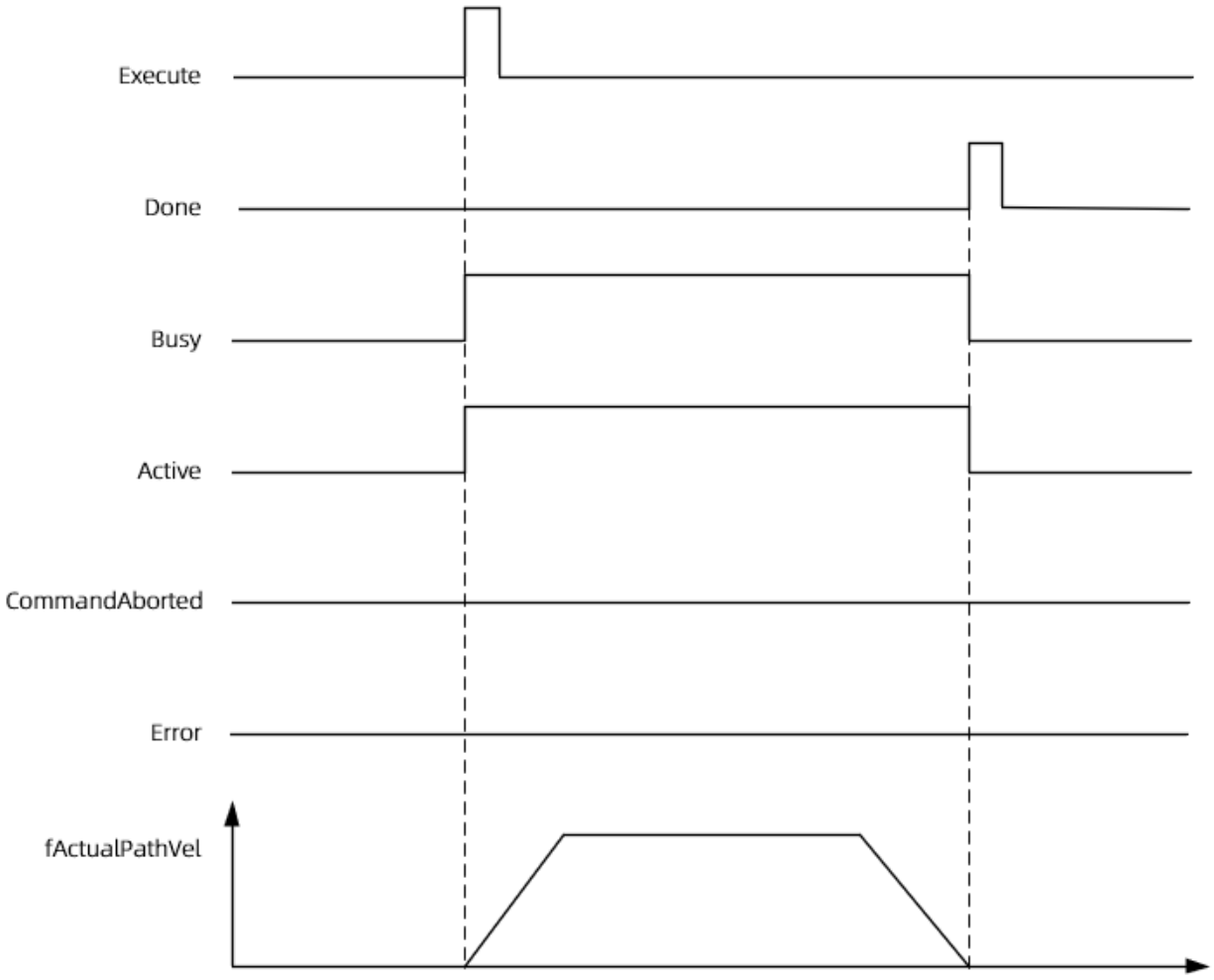
- **Program example**

  The example program for this instruction can refer to the MC_MoveDirectRelative instruction. This instruction can also be combined with other AxesGroup motion instructions.

- **Sequence diagram**

  When "Execute" is started, the output variables "Busy" and "Active" become TRUE. When the target position is reached, the axis group speed becomes 0, the output variable "Done" becomes TRUE, and the output variables "Busy" and "Active" become FALSE.

  An arc interpolation instruction is called to perform interpolation along the X-axis and Y-axis.



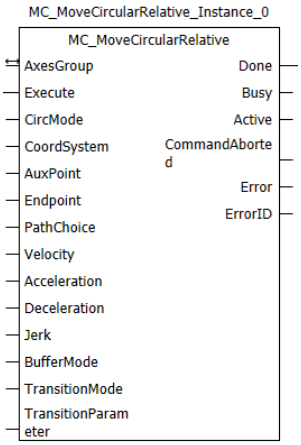
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.18 MC_MoveCircularRelative

This enables a circular relative motion for an axis group, executing a circular interpolation from the current TCP position to a relative target position in the specified coordinate system.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_MoveCircularRelative	Circular relative motion		MC_MoveCircularRelative_Instance_0(AxesGroup := , Execute := , CircMode := , CoordSystem := , AuxPoint := , Endpoint := , PathChoice := , Velocity := , Acceleration :=, Deceleration :=, Jerk := , BufferMode := , TransitionMode := , TransitionParameter := , Done => , Busy => , Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
CircMode	Circular mode	18.3.3.2.4.18 MC_CIRCMODE: Circular Interpolation Mode Enumeration	0, 1, 2	0	0: Three-point arc. 1: Center-point arc. 2: Radius-defined arc.
CoordSystem	Coordinate system type	18.3.3.2.4.19 MC_COORDSYSTEM: Coordinate System Enumeration	ACS, PCS	PCS	Currently only supports PCS.
AuxPoint	Auxiliary point	ARRAY [1..GROUP_DEF.AXIS_NUM] OF LREAL	-	[0,0,…]	For the auxiliary point position (relative to the start point) of each dimension in the specified coordinate system, see Axis group type description.
EndPoint	End point	ARRAY [1..GROUP_DEF.AXIS_NUM] OF LREAL		[0,0,…]	For the end point position (relative to the start point) of each dimension in the specified coordinate system, see Axis group type description.
PathChoice	Path choice	18.3.3.2.4.17 MC_CIRC_PATHCHOICE: Circular Path Choice Enumeration	0, 1	0	This parameter is valid when CircMode is set to CENTER_ARC or RADIUS_ARC arc mode.
Velocity	Velocity	LREAL	> 0	-	Maximum linear velocity. This value may not necessarily be reached.
Acceleration	Acceleration	LREAL	> 0	-	Maximum linear acceleration. This value may not necessarily be reached.
Deceleration	Deceleration	LREAL	> 0	-	Maximum linear deceleration. This value may not necessarily be reached.
Jerk	Jerk	LREAL	>= 0	-	Maximum linear jerk. This value may not necessarily be reached.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting 1: mcBuffered 2: mcBlendingLow 3: mcBlendingPrevious 4: mcBlendingNext 5: mcBlendingHigh	0	0: mcAborting (Aborting). 1: mcBuffered (Buffered). 2: mcBlendingLow (Blending with the lowest velocity of both FBs). 3: mcBlendingPrevious (Blending with the velocity of the first FB). 4: mcBlendingNext (Blending with velocity of the second FB). 5: mcBlendingHigh (Blending with the highest velocity of both FBs). Note: Buffer modes 2 to 5 currently have the same effect. All are blending modes. The behavior depends on the TransitionMode and TransitionParameter parameters.
TransitionMode	Transition mode	18.3.3.2.4.10 MC_TRANSITION_MODE: Transition Mode Enumeration	0: TMNone 1: TMStartVelocity 3: TMCornerDistance 4: TMMaxCornerDeviation 10: TMDefinedVelocity 11: TMResetAbort 12: TMDecAbort	0	Definition of the transition trajectory characteristic types. When BufferMode is set to MC_ABORTING, only the transition mode parameters TMNone, TMResetAbort, and TMDecAbort are allowed. When BufferMode is set to MC_BUFFERED, the transition mode is invalid. When BufferMode is a blending mode (2 to 5), TMResetAbort or TMDecAbort is not allowed.
TransitionParameter	Transition parameter	LREAL	Velocity [0.0,1.0] Distance [0.0,+∞)	1	Transition parameters vary with the transition mode.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Instruction has controlled the axis group	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

  Position dimension

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity	Velocity	Any velocity that exceeds the motion parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	The velocity equal to 0 will be automatically corrected to 1e-5.
Acceleration	Acceleration	Any acceleration that exceeds the motion parameter limit will be automatically corrected to the acceleration limit.	Any acceleration less than 0 will be automatically corrected to 10% of the acceleration limit.	Any acceleration equal to 0 will be automatically corrected to 10% of the acceleration limit.
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the acceleration limit.	Any deceleration less than 0 will be automatically corrected to 10% of the acceleration limit.	Any deceleration equal to 0 will be automatically corrected to 10% of the acceleration limit.
Jerk	Jerk	For the non-trapezoidal motion profiling, any jerk that exceeds the software limit will be automatically corrected to the jerk limit.	For the non-trapezoidal motion profiling, any jerk that exceeds the software limit will be automatically corrected to 10% of the jerk limit.	For the non-trapezoidal motion profiling, any jerk that exceeds the software limit will be automatically corrected to 10% of the jerk limit.

  Percentage dimension

Input Variable	Name	Correction Description for Parameter Greater than 1	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity	Velocity	Any velocity exceeding 1 is automatically corrected to 1.	Any velocity less than 0 will be automatically corrected to 1e-5.	The velocity equal to 0 will be automatically corrected to 1e-5.
Acceleration	Acceleration	Any acceleration exceeding 1 is automatically corrected to 1.	Any acceleration less than 0 will be automatically corrected to 10%.	Any acceleration equal to 0 will be automatically corrected to 10%.
Deceleration	Deceleration	The value exceeding 1 is automatically corrected to 1.	Any deceleration less than 0 will be automatically corrected to 10%.	Any deceleration equal to 0 will be automatically corrected to 10%.
Jerk	Jerk	For the non-trapezoidal motion profiling, the value exceeding 1 is automatically corrected to 1.	For the non-trapezoidal motion profiling, the value exceeding 1 is automatically corrected to 10%.	For the non-trapezoidal motion profiling, the value exceeding 1 is automatically corrected to 10%.

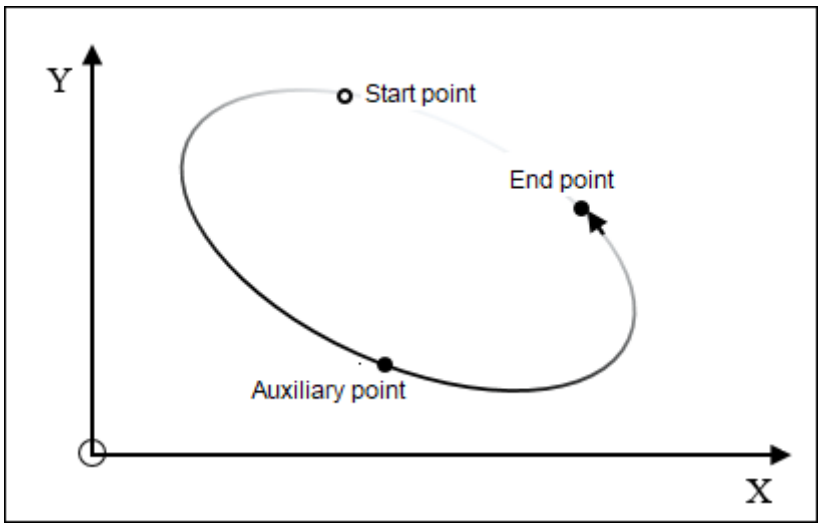
- **Function**
  - This instruction is executed on the rising edge of Execute and performs an absolute arc motion, supporting planar or spatial arcs.
  - CircMode, CoordSystem, AuxPoint, EndPoint, PathChoice, Velocity, Acceleration, Deceleration, Jerk, BufferMode, TransitionMode, and TransitionParameter can be specified in the input variables.
  - For the instruction, blending modes 2, 3, 4, and 5 for BufferMode have the same behavior. The transition effect depends on the TransitionMode and TransitionParameter settings.
  - Supports the same-cycle triggering of instructions. The Done output signal of the instruction serves as the trigger signal for Execute.
  - **AuxPoint**
    - For the axis type corresponding to each position vector in the AuxPoint array, see Axis group type description. Only PCS coordinates are supported.
    - The ABC rotation axes of the coordinate system correspond sequentially to the ZYX-type Euler angles. If the configuration supports orientation motion, the orientation parameters set for the auxiliary points will not be included in the orientation trajectory planning.
  - **EndPoint**
    - For the axis type corresponding to each position vector in the EndPoint array, see Axis group type description. Only PCS coordinates are supported.
    - The ABC rotation axes of the coordinate system correspond sequentially to the ZYX-type Euler angles. If the configuration supports orientation motion, the configured ABC relative positions must be within ±180°, otherwise an error will be generated.
    - In relative motion mode, the rules for calculating the target position of the axis group are shown in the table below.

Coordinate system type (calculation of target positions for each axis in the existing axis group configuration)
PCS
X-axis target position = X- axis current position + EndPoint [1]
Y-axis target position = Y-axis current position + EndPoint [2]
Z-axis target position = Z-axis current position + EndPoint [3]
A-axis target position = A-axis current position + EndPoint [4]

- If the A-axis target position is not within ±180°, the system will make the following adjustments: if the A-axis target position is greater than 180°, the system will subtract 360° until the result falls within ±180°; if the A-axis target position is less than -180°, the system will add 360° until the result falls within ±180°. Example: $A_{\text{calculated target position}}$ = 260°, $A_{\text{converted target position}}$ = -100°. Note that for the EVO500 series PLC (version V1.2), when the axis group type is set to "XYZA configuration", the A-axis functions as an auxiliary axis. Its movement is not limited to ±180 degrees. If it exceeds ±180 degrees, the axis continues moving according to the preset position.
- If the target position would exceed any joint axis limit after kinematic transformation into joint space, starting this instruction will cause an error.
- If the configuration supports orientation motion, the axis group will move to the target orientation along the shortest path.

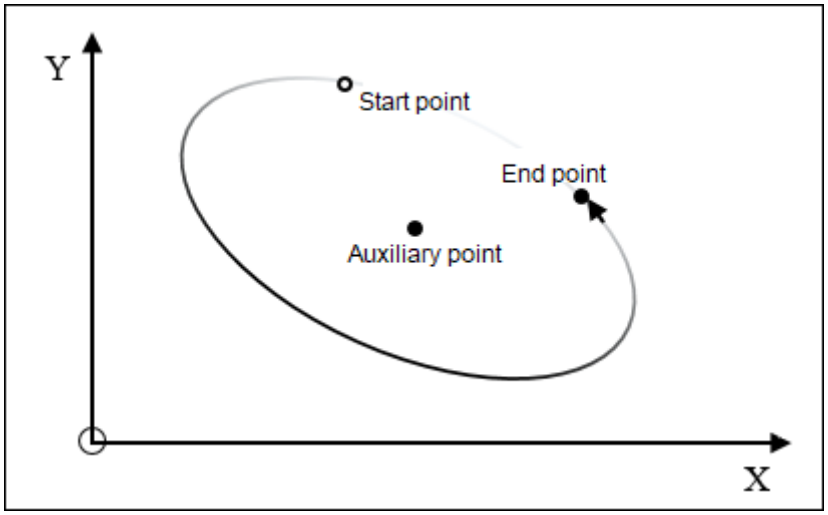
◦ **CircMode**

- BORDER_ARC three-point circular arc: a circular arc defined by three points—the current position of AxesGroup as start point, AuxPoint, and EndPoint. **If the three points are collinear or two of them coincide, the trajectory is a straight line connecting the start point and the end point**. If the three points are not collinear, the trajectory is a circular arc passing through the three points in sequence. The range of the arc angle is (0, 2π). The figure shows the composition of a three-point arc in space, with axis Z facing off-screen.

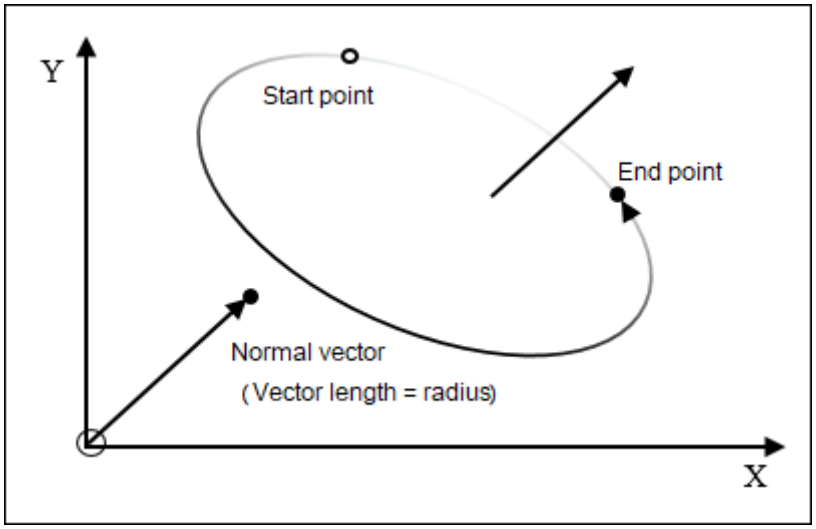


- CENTER_ARC center circular arc: a circular arc is defined by the current position of AxesGroup as start point, AuxPoint as the center, and EndPoint. The arc is specified as either a long arc or short arc through PathChoice. The arc angle range is less than 2π, and the arc angle range is not equal to π. **When the three points are collinear, the trajectory is a straight line.** If the distance between the start point and the end point to the center of the circle deviates too much, an error will be reported. The distance from the start point to the center is named L1, and the distance from the end point to the center is named L2. The center will be automatically corrected if $|L1-L2| < 0.01 \times \min(L1,L2)$.

Exceeding this deviation will trigger an error. (Due to floating-point arithmetic precision issues, no strict equality exists.) The figure shows the composition of a center-point arc in space, with axis Z facing off-screen.



- RADIUS_ARC radius arc: **the radius of the arc is determined by the vector length of** AuxPoint, and the vector direction determines the normal vector of the arc. PathChoice determines whether the arc is long or short. The arc angle range is less than 2π. When the start and end points are not perpendicular to the normal vector, an error will be reported if the deviation angle is too large. If the deviation is small (<10°), the direction of the normal vector will be corrected according to the parameter. If the distance from the endpoint exceeds twice the length of the normal vector and also exceeds one ten-thousandth of the length (that is, radius), an arc error is reported. When the deviation is within one ten-thousandth, the arc radius is corrected to a semicircle. An error is reported when the length of the normal vector or the distance between start and end points is less than the computational precision of 1e-6. **For an axis group configured with only two position axes, AuxPoint[0] indicates the radius of the circular arc. A positive value indicates rotation about the arc's normal axis in the positive direction (typically following the right-hand rule), while a negative value indicates rotation in the opposite (negative) direction.**

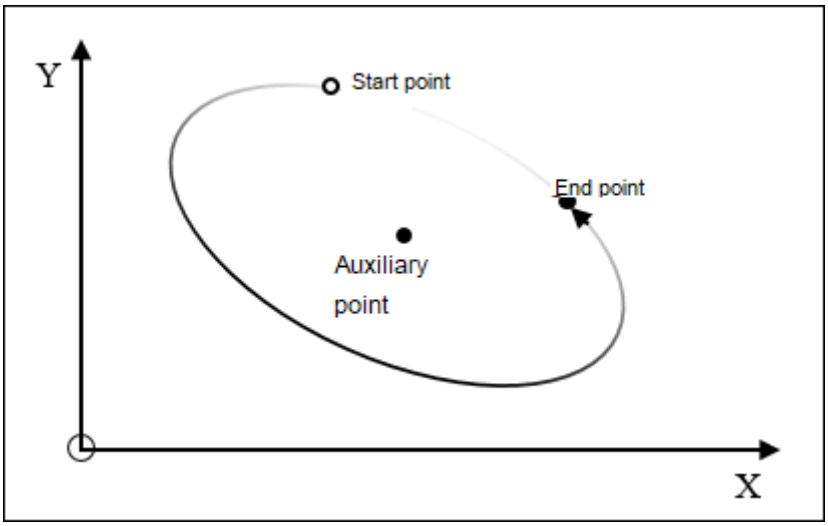


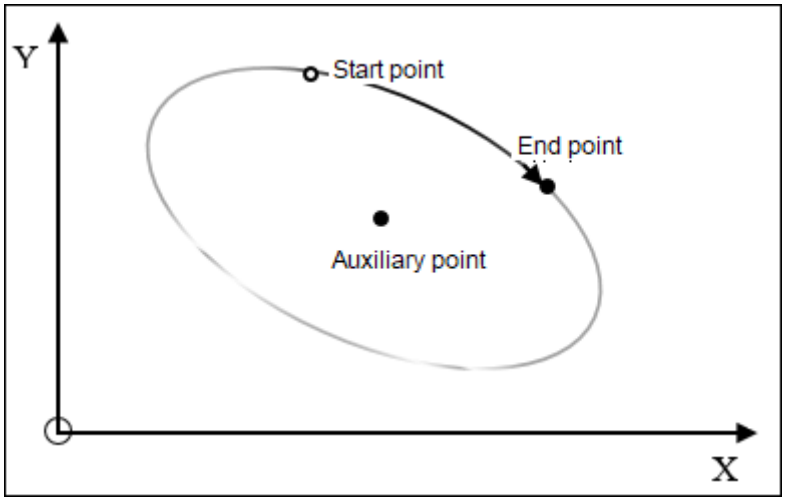
- **CoordSystem**
  - Specify the coordinate system for circular relative motion. Currently, only relative circular motion under the PCS is supported.
- **Velocity constraint**
  - The arc trajectory is bound by the maximum acceleration and deceleration parameters of the system. The centripetal acceleration is kept within limits by restricting the velocity along the trajectory segment. The velocity is constrained as follows:

    Vmax = Sqrt(Amax *R*Ccoef)

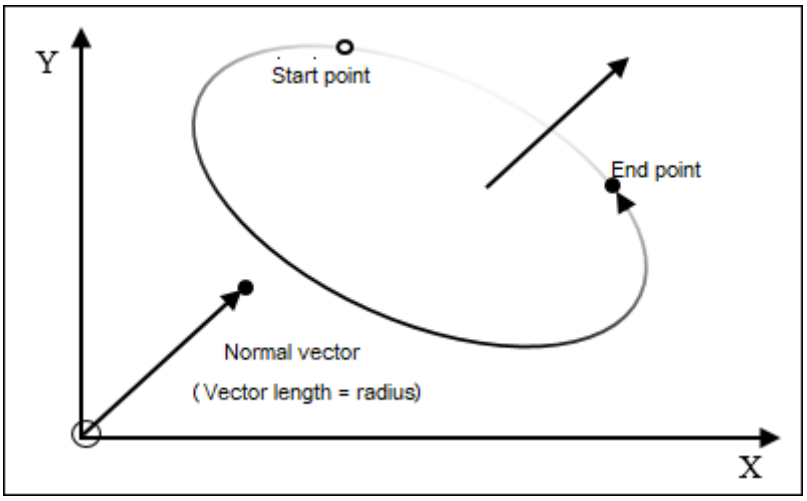
    Here, Ccoef indicates the transition factor, R indicates the arc radius, and Amax indicates the maximum allowable acceleration of the system.
- **PathChoice**

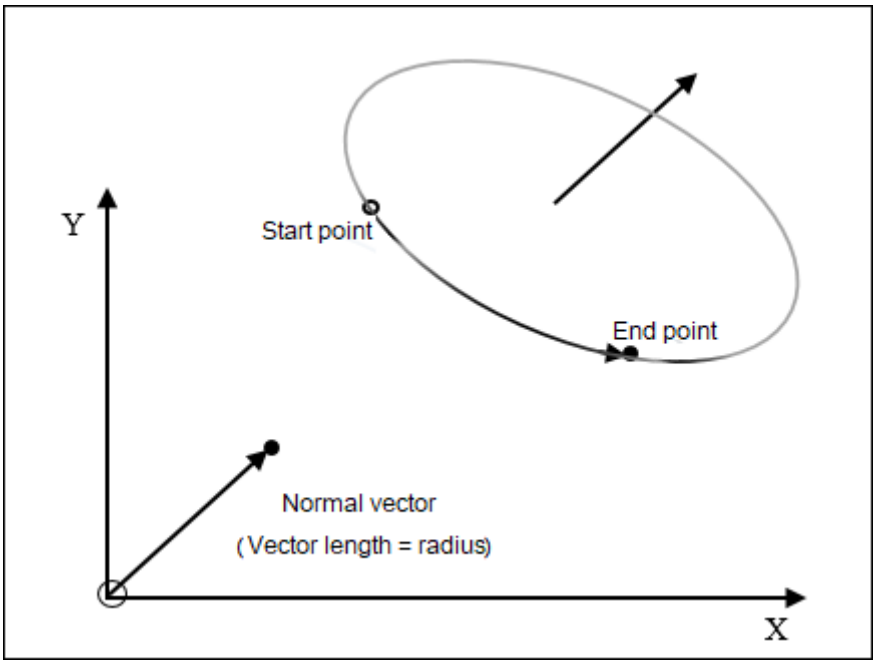
  Depending on the CircMode, there are different choices for the arc direction and length.
  - CENTER_ARC center circular arc: In the figure below, the first setting is the LONG_ARC path, and the second setting is the SHORT_ARC path.





- RADIUS_ARC radius arc: In the figure below, the first setting is the LONG_ARC path, and the second setting is the SHORT_ARC path.



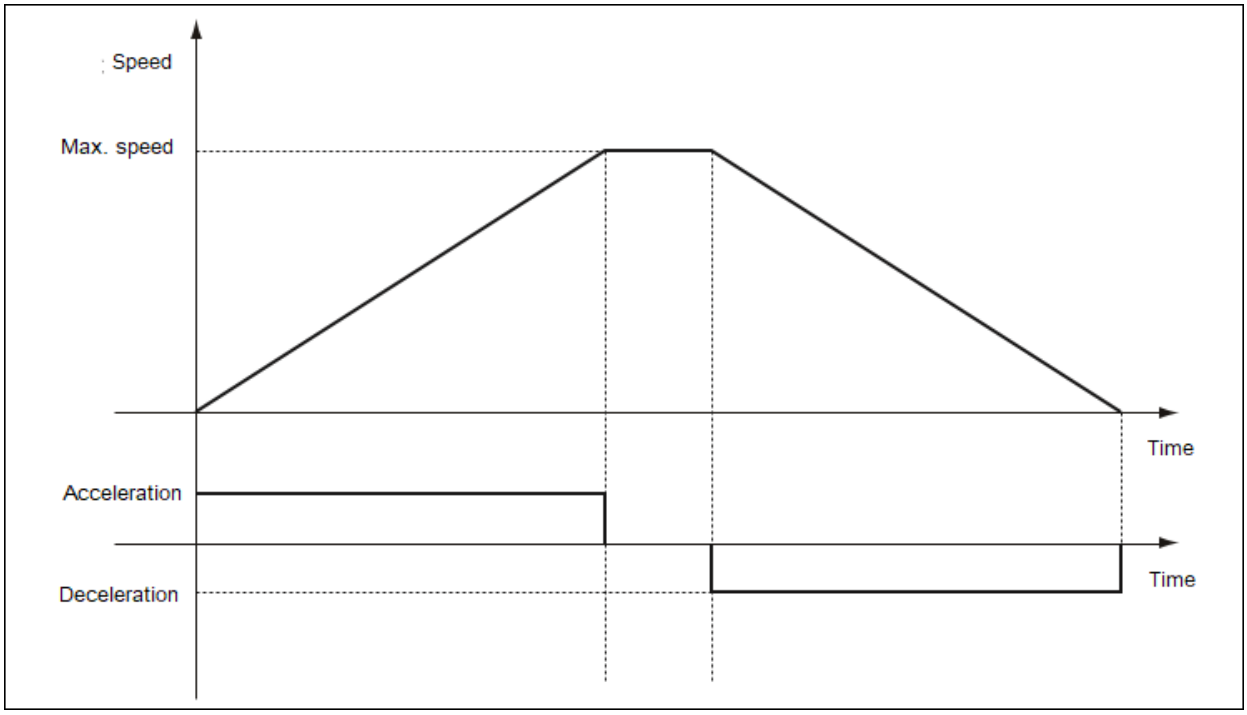


◦ **Velocity, Acceleration, Deceleration, Jerk**

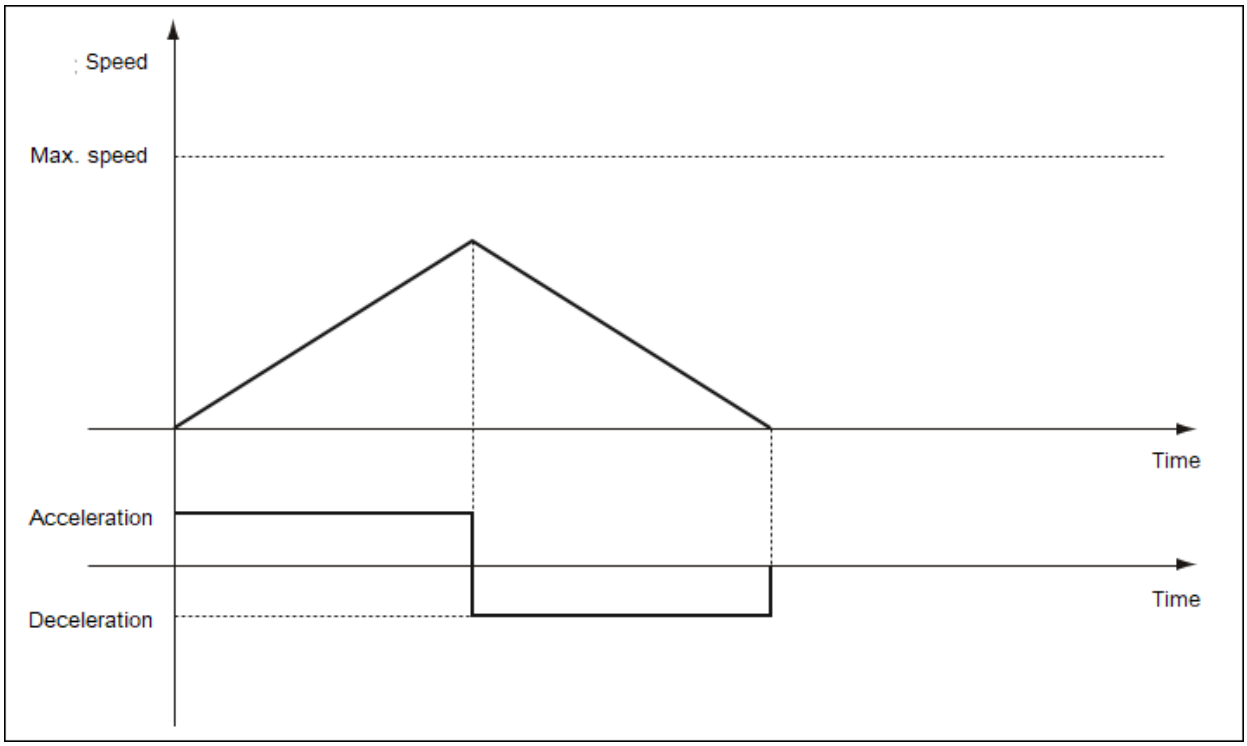
- If the velocity of any Axis variable in AxesGroup exceeds the maximum axis velocity set by the software multiplied by the warning planning scaling factor, it will trigger an error-stop condition.

- The trajectory planning curve type of AxesGroup is set according to requirements. If the planning curve type is a trapezoidal curve, the Jerk variable does not take effect and is treated as 0. (The planning curve type of AxesGroup is specified in the axis group data structure).
- For Jerk = 0 or Jerk ≠ 0, the velocity, acceleration, deceleration, and jerk curves are shown in the figure below.

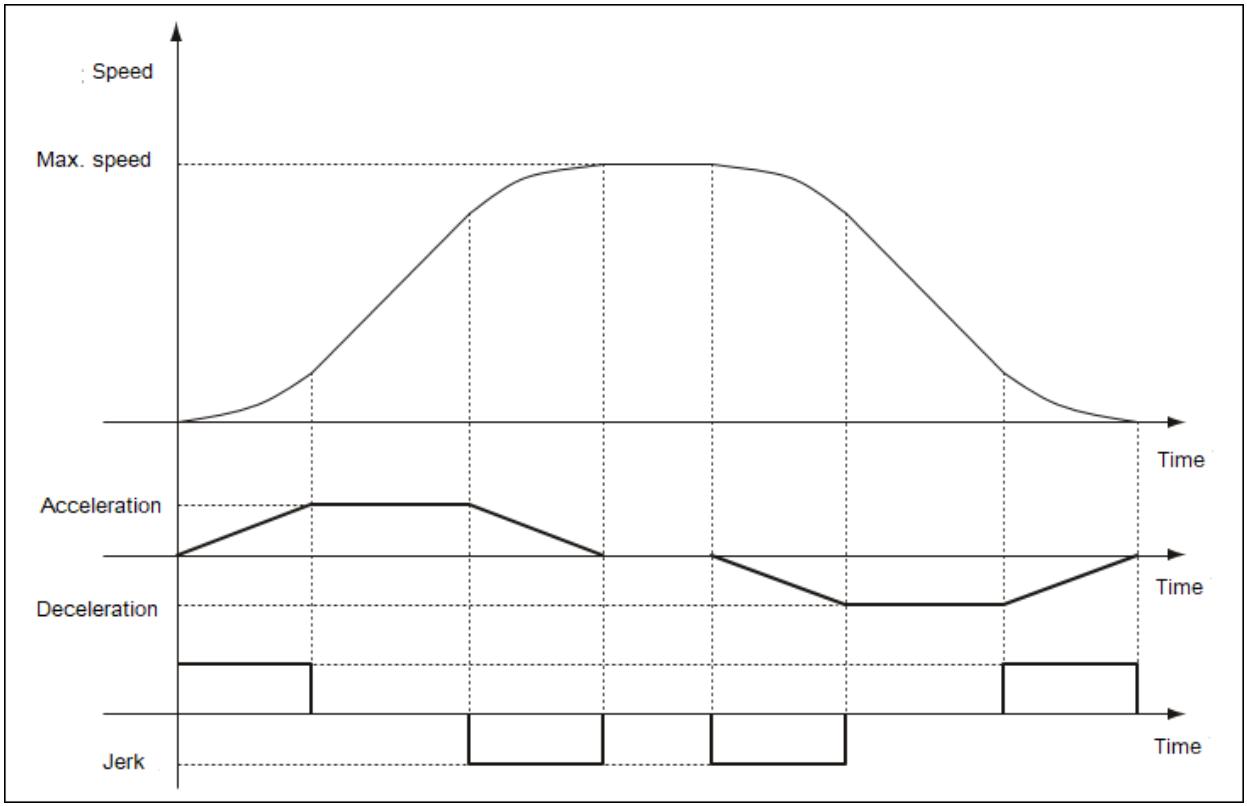
For Jerk = 0, the axis group can reach the maximum interpolation velocity.



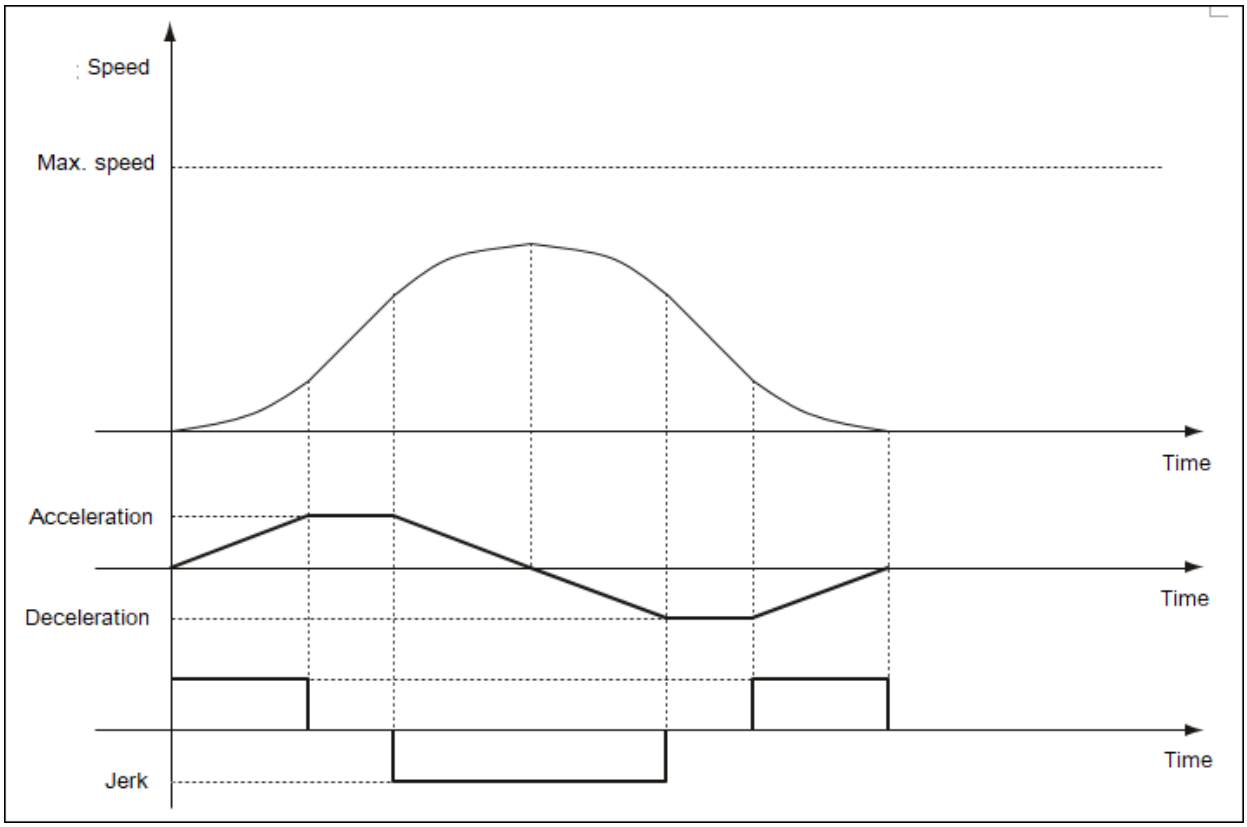
For Jerk = 0, the axis group cannot reach the maximum interpolation velocity due to too short travel distance.



For Jerk ≠ 0, the axis group can reach the maximum interpolation velocity. (Non-trapezoidal planning curve)



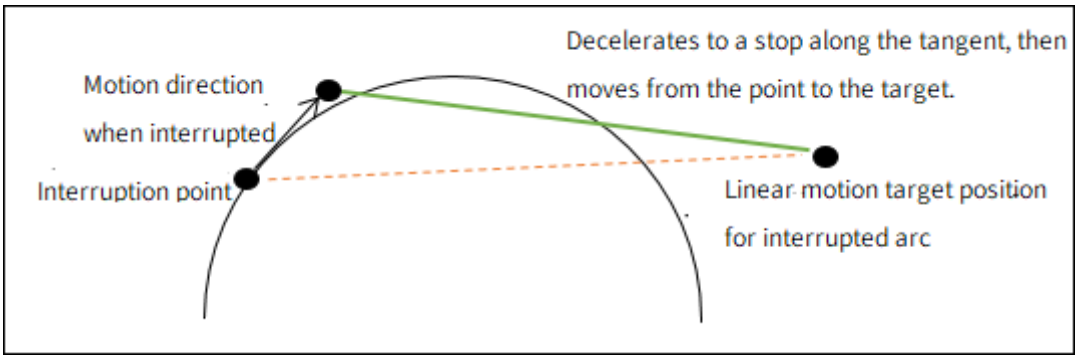
For Jerk ≠ 0, the axis group cannot reach the maximum interpolation velocity due to too short travel distance. (Non-trapezoidal planning curve)



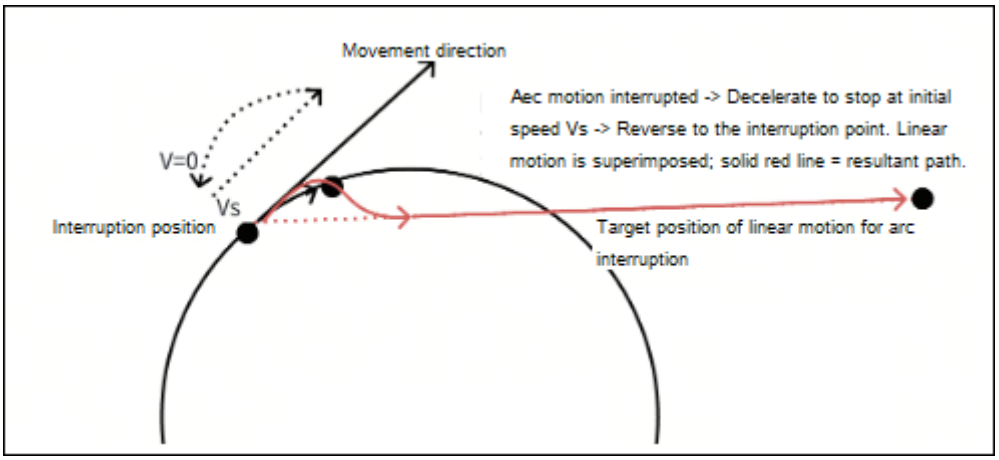
- **Re-triggering the instruction**
  - While the instruction is in motion, that is, when the instruction output parameters "Busy" and "Active" are both TRUE, this instruction cannot be re-triggered.
  - While the instruction is in buffer mode, that is, when Busy is TRUE and Active is FALSE, this instruction can be re-triggered after parameter modification. If the modified parameters are invalid, or if the buffered instruction data has already been used in motion calculations, retriggering the buffered

instruction will cause an error. Then, the axis group will enter the GroupStop state, and the currently executing instruction will also report an error.

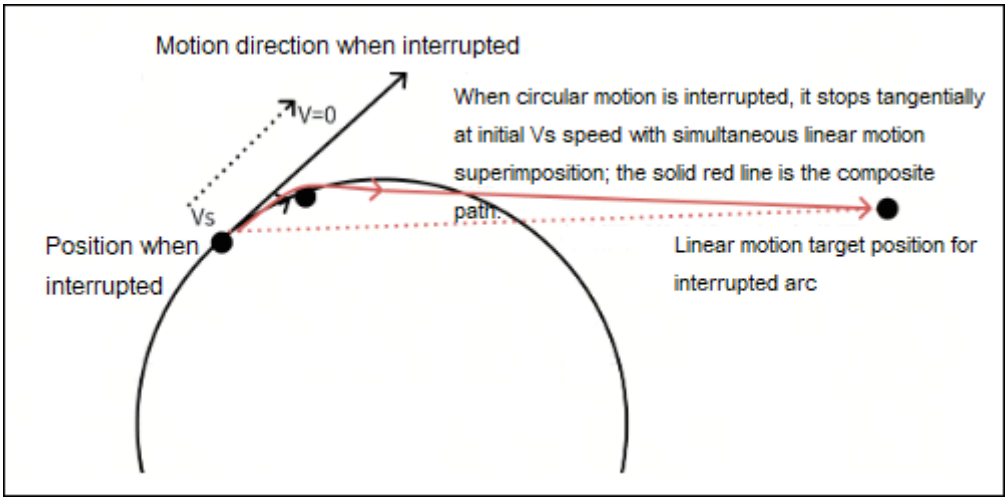
- **Multiple execution of motion instructions**
  - When this instruction is issued while another motion instruction is being executed, the new instruction can either be switched to immediately or buffered for later execution. Each axis group can buffer up to 8 instructions. In the case of multiple executions, the behavior of this instruction is determined by BufferMode.
  - BufferMode
    - Aborting: MC_ABORTING aborts the currently executing instruction and begins executing this instruction. The abort operation supports three modes, configurable via TransitionMode: TMNone, TMResetAbort, and TMDecAbort.
      TMNONE mode: During execution, the motion decelerates to a stop along the tangential direction from the point of interruption. The next motion segment then starts from this stop point, as shown in the figure below.



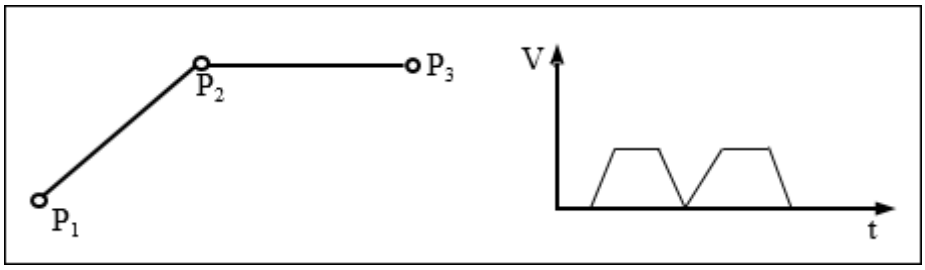
TMResetAbort mode: During execution, the axis decelerates along the tangential direction at the initial velocity of the interruption point, reverses its direction to return to that point, while simultaneously superimposing (blending) with the next motion segment, as shown in the figure below.



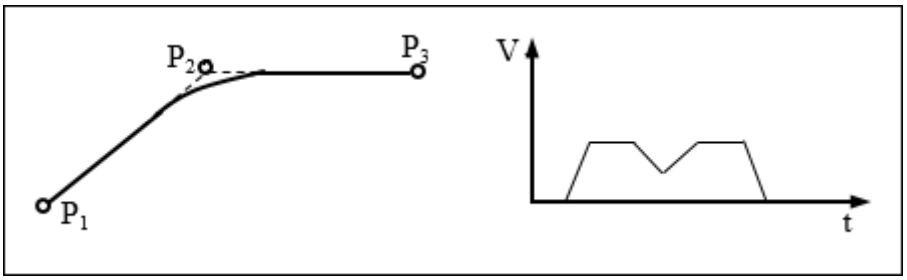
TMDecAbort mode: During execution, the axis decelerates to 0 along the tangential direction at the initial velocity of the interruption point, while simultaneously superimposing (blending) with the next motion segment, as shown in the figure below.



- Buffer: MC_BUFFERED, this instruction is buffered and executed after the current instruction is completed.



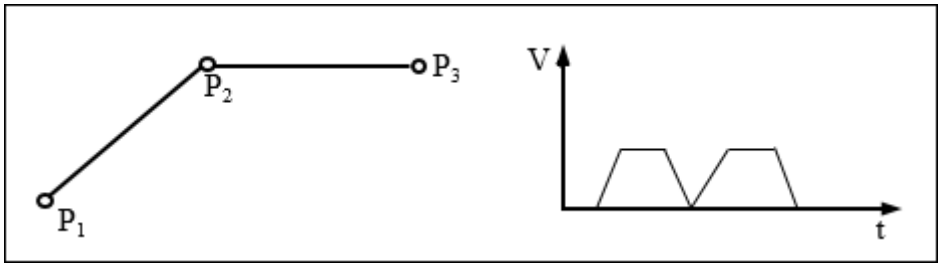
- Blending: MC_BLEND, the velocities of the two adjacent instructions are blended in the specified manner. This manner is specified in TransitionMode.



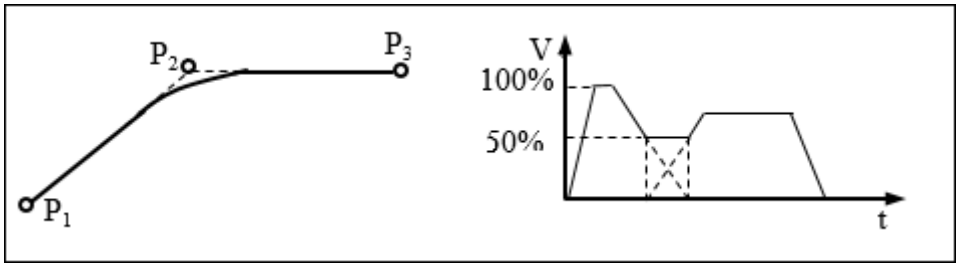
◦ TransitionMode

This variable specifies how to blend the velocities of two instructions when Blending is specified for the BufferMode.

- TMNone: Insert no transition curve. If TMNone is selected in a blending mode, the result is currently consistent with the transition between the velocities of the two instructions in buffered mode. Further extensions are planned.

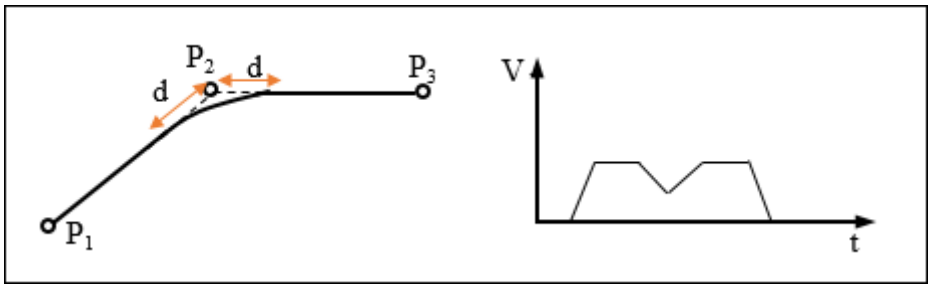


- TMStartVelocity: Transition with given start velocity. Blending transition at a certain ratio (0, 1] of the actual maximum velocity of the current motion control instruction. Effective for transitions between any motion control instructions.

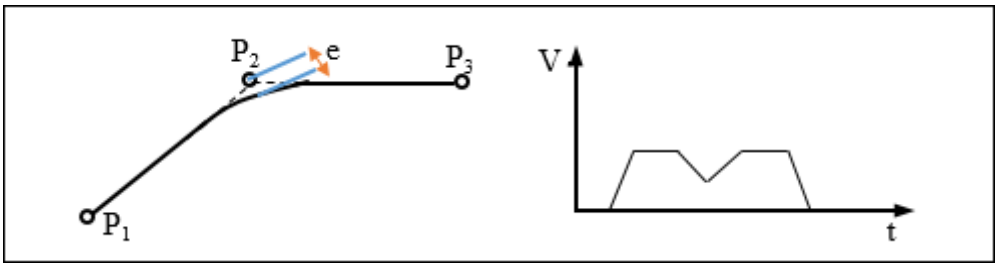


- TMCornerDistance: Corner transition mode In Cartesian space, this parameter specifies the distance before the end of the previous instruction, measured in the main dimension space. The transition trajectory is initiated at this distance to connect with the adjacent trajectory. The distance here is a generalized distance, which depends on the main dimension type setting of the axis group.
  When the primary dimension is set to position: the parameter is the displacement from the transition point to the end point.

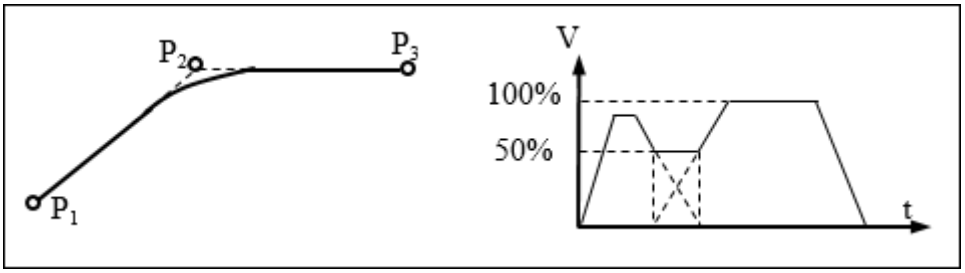
  When the primary dimension is set to ratio: The parameter is the ratio (0 to 1) of the resultant distance from the transition point to the endpoint to the total resultant displacement.



- TMMaxCornerDeviation: Transition mode using the maximum corner position deviation. In Cartesian space, this parameter specifies the maximum position deviation in the position subspace during the transition between adjacent instructions. An appropriate transition trajectory is constructed to connect the two adjacent paths, keeping the maximum position deviation within this limit. This is only the position deviation, independent of the main unit. Effective when both adjacent instructions are Cartesian space instructions (LIN/CIRC). Otherwise, they are processed with transition disabled.



- TMDefinedVelocity: Transition with defined velocity. Blending transition at a certain ratio (0, 1] of the actual maximum velocity of the instruction following the current motion control instruction. Effective for transitions between any motion control instructions.



- **Precautions**
  - This instruction only supports relative circular motion in PCS coordinates.
  - The dimensions of the axis group planning parameters (velocity, acceleration, deceleration, and jerk) depend on the primary dimension set during axis group initialization. Different primary dimension types result in different dimensions and meanings for these parameters, and the parameter units are consistent with the axis group settings for the user.
    - UNIT_TRAJECT position. The input planning parameters are processed according to workspace position planning parameters.
    - UNIT_PERCENT and maximum planning parameter ratio: The input planning parameters are processed as a combined ratio coefficient based on workspace pose planning parameters.
  - When the current motion instruction is interrupted, circular motion constructs a transition trajectory in working space along the direction of the velocity at the breakpoint to ensure smooth connection with the subsequent trajectory. This ensures velocity continuity only. However, the acceleration continuity is not guaranteed. For details on the three transition modes supported for aborting, see the motion abort description.
  - The transition trajectory calculation for the aborting instruction is planned according to the acceleration segment parameters set in the instruction. The higher the acceleration segment parameter value, the shorter the transition trajectory and the faster the transition.
  - **If the instruction motion mode is set to an interruption mode and the deceleration distance is insufficient, the function block reports an error. But the axis group does not enter the GroupErrorStop state; the original motion continues to execute.**
  - This instruction can only be initiated when the axis group is in the GroupMoving or GroupStandby state. Attempting to start this instruction in any other state will result in an error.
  - During the execution of this instruction, it can be interrupted by other motion instructions, MC_GroupStop, or MC_GroupHalt.
  - After the execution of this instruction, the axis group (AxesGroup) state is changed from GroupMoving to GroupStandby, or remains in the GroupMoving state.
  - If the current motion is in a static coordinate system, the axis group is changed from the GroupMoving state to the GroupStandby state. If the current motion is in a dynamic coordinate system, the axis group remains in the GroupMoving state.
  - If the motion includes orientation information, the orientation of the intermediate point for the BORDER_ARC three-point arc instruction is ignored; only the orientations of the start and end points are considered.

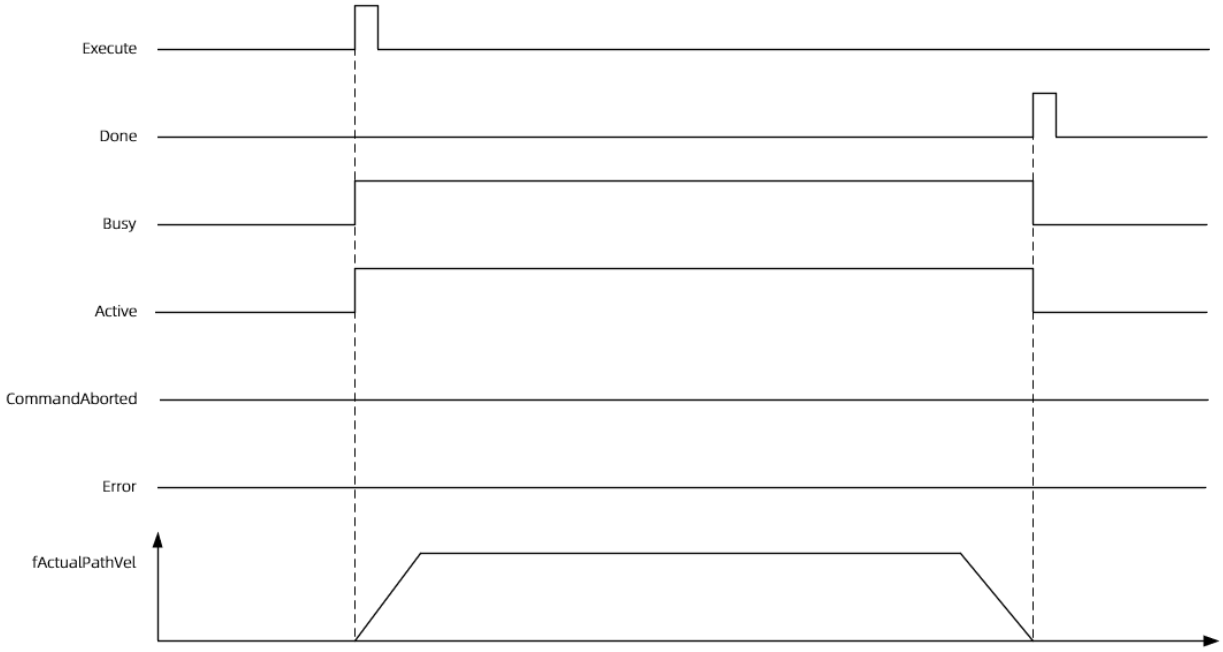
- **Program example**

  The example program for this instruction can refer to the MC_MoveDirectRelative instruction. This instruction can also be combined with other AxesGroup motion instructions.

- **Sequence diagram**

  When "Execute" is started, the output variables "Busy" and "Active" become TRUE. When the target position is reached, the axis group speed becomes 0, the output variable "Done" becomes TRUE, and the output variables "Busy" and "Active" become FALSE.

  An arc interpolation instruction is called to perform interpolation along the X-axis and Y-axis.



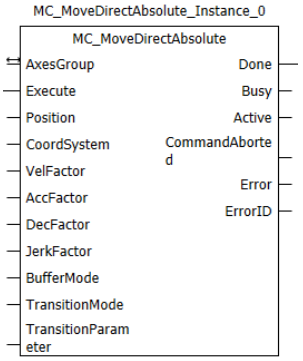
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.19 MC_MoveDirectAbsolute

The point-to-point instruction for the axis group commands the axis group to move from the current TCP point to an absolute position without requiring a specific path.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_MoveDirectAbsolute	Point-to-point absolute motion		MC_MoveDirectAbsolute_Instance_0(AxesGroup := , Execute := , Position := , CoordSystem := , VelFactor := , AccFactor := , DecFactor := , JerkFactor := , BufferMode := , TransitionMode := , TransitionParameter := , Done => , Busy => , Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
Position	Absolute Position	ARRAY [1..GROUP_DEF.AXIS_NUM] OF LREAL	-	-	For the end position in each dimension in the specified coordinate system, see Axis group type description.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
CoordSystem	Coordinate system type	18.3.3.2.4.19 MC_COORDSYSTEM: Coordinate System Enumeration	ACS, PCS	ACS	Select the application coordinate system type.
VelFactor	Velocity factor	LREAL	(0,1]	0.10	Maximum velocity factor. This value may not necessarily be reached. If the value is greater than 1, it is corrected to 1. If the value is less than or equal to 0, it is corrected to 1e-5.
AccFactor	Acceleration factor	LREAL	(0,1]	0.10	Maximum acceleration factor. This value may not necessarily be reached. If the value is greater than 1, it is corrected to 1. If the value is less than or equal to 0, it is corrected to 0.1.
DecFactor	Deceleration factor	LREAL	(0,1]	0.10	Maximum deceleration factor. This value may not necessarily be reached. If the value is greater than 1, it is corrected to 1. If the value is less than or equal to 0, it is corrected to 0.1.
JerkFactor	Jerk factor	LREAL	[0,1]	0.10	Maximum jerk factor. This value may not necessarily be reached. For the non-trapezoidal motion profiling, the factor will be corrected to 1 if it is greater than 1. If it is less than or equal to 0, it will be corrected to 0.1.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting 1: mcBuffered 2: mcBlendingLow 3: mcBlendingPrevious 4: mcBlendingNext 5: mcBlendingHigh	0	0: mcAborting (Aborting). 1: mcBuffered (Buffered). 2: mcBlendingLow (Blending with the lowest velocity of both FBs). 3: mcBlendingPrevious (Blending with the velocity of the first FB). 4: mcBlendingNext (Blending with velocity of the second FB). 5: mcBlendingHigh (Blending with the highest velocity of both FBs). Note: Buffer modes 2 to 5 currently have the same effect. All are blending modes. The behavior depends on the TransitionMode and TransitionParameter parameters.
TransitionMode	Transition mode	18.3.3.2.4.10 MC_TRANSITION_MODE: Transition Mode Enumeration	0: TMNone 1: TMStartVelocity 10: TMDefinedVelocity 11: TMResetAbort 12: TMDecAbort	0	Definition of the transition trajectory characteristic types. When BufferMode is set to MC_ABORTING, only the transition mode parameters TMNone, TMResetAbort, and TMDecAbort are allowed. When BufferMode is set to MC_BUFFERED, the transition mode is invalid. When BufferMode is a blending mode (2 to 5), only TMNone, TMStartVelocity, and TMDefinedVelocity are allowed.
TransitionParameter	Transition parameter	LREAL	Velocity [0.0,1.0]	1.0	Transition parameters vary with the transition mode.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Active	Instruction has controlled the axis group	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis group is being controlled.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

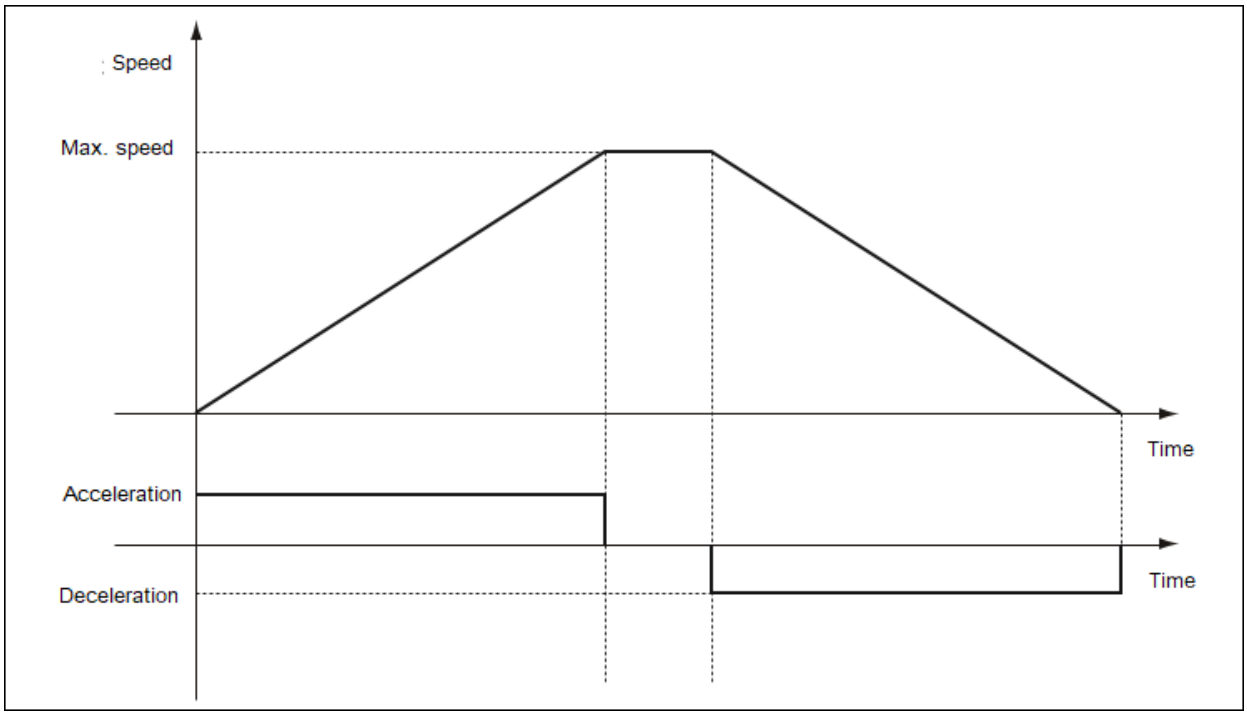
Input Variable	Name	Correction Description for Values Greater Than 1 or Backend Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
VelFactor	Velocity factor	If the velocity factor is greater than 1, it is automatically corrected to 1.	If the velocity factor is less than 0, it is automatically corrected to 1e-5 for operation.	If the velocity factor is equal to 0, it is automatically corrected to 1e-5 for operation.
AccFactor	Acceleration factor	If the acceleration factor is greater than 1, it is automatically corrected to 1.	If the acceleration factor is less than 0, it is automatically corrected to 0.1 for operation.	If the acceleration factor is equal to 0, it is automatically corrected to 0.1 for operation.
DecFactor	Deceleration factor	If the deceleration factor is greater than 1, it is automatically corrected to 1.	If the deceleration factor is less than 0, it is automatically corrected to 0.1 for operation.	If the deceleration factor is equal to 0, it is automatically corrected to 0.1 for operation.

Input Variable	Name	Correction Description for Values Greater Than 1 or Backend Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
JerkFactor	Jerk factor	If the jerk factor is greater than 1 in the non-trapezoidal motion profiling, it is automatically corrected to 1.	If the jerk factor is less than 0 in the non-trapezoidal motion profiling, it is automatically corrected to 0.1 for operation.	If the jerk factor is equal to 0 in the non-trapezoidal motion profiling, it is automatically corrected to 0.1 for operation.

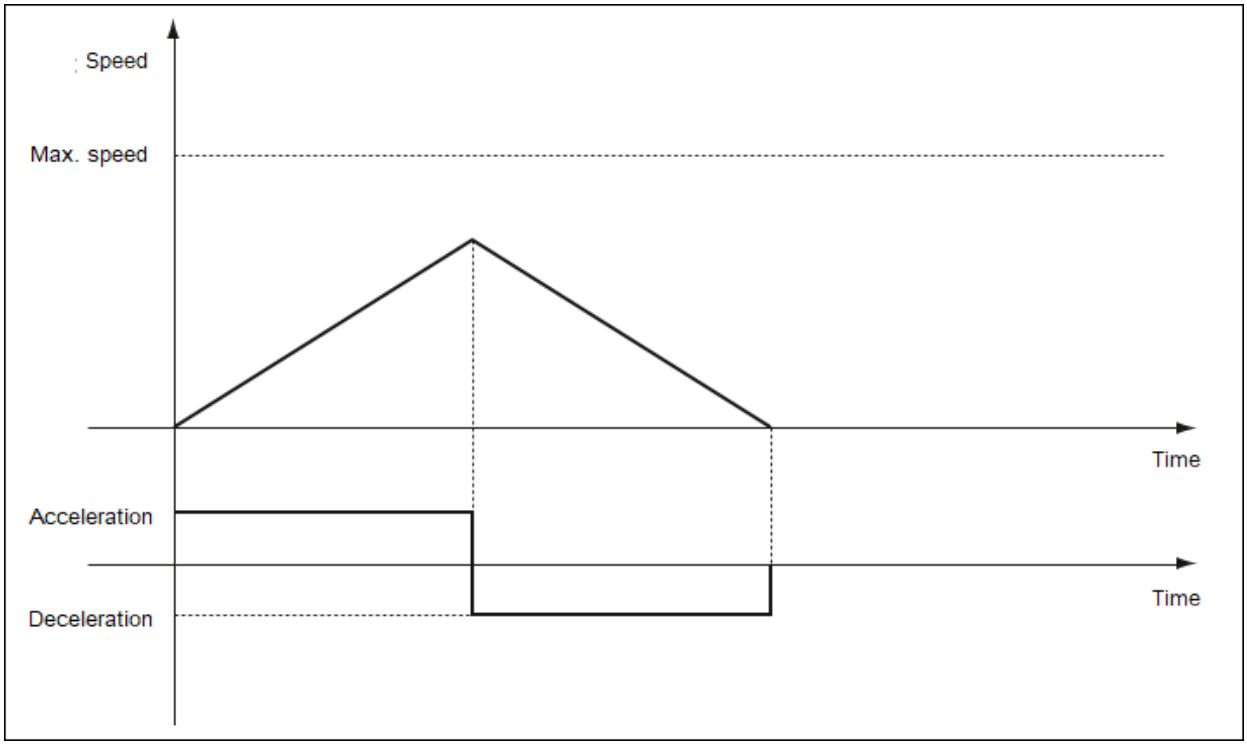
- **Function**
  - This instruction is executed on the rising edge of Execute and performs point-to-point absolute motion from two axes to four axes.
  - Position, CoordSystem, VelFactor, AccFactor, DecFactor, JerkFactor, BufferMode, TransitionMode, TransitionParameter can be specified in the input variables.
  - For the instruction, blending modes 2, 3, 4, and 5 for BufferMode have the same behavior. The transition effect depends on the TransitionMode and TransitionParameter settings.
  - Supports the same-cycle triggering of instructions. The Done output signal of the instruction serves as the trigger signal for Execute.
  - **CoordSystem**
    - Specifies the coordinate system for point-to-point absolute motion.
  - **Position (Absolute position coordinate)**
    - For the axis type corresponding to each position vector in the Position array, see Axis group type description.
    - In absolute motion mode, the axis group uses the set values in the Position array as the target position.
    - When the coordinate system type is set to PCS, the ABC rotary axes of the coordinate system correspond to the ZYX-type Euler angles, respectively. If the configuration supports orientation motion, the configured ABC absolute positions must be within ±180°, otherwise an error will be generated.
    - If the target position exceeds any joint axis limit, starting this instruction will cause an error.
    - If the configuration supports orientation motion, the axis group will move to the target orientation along the shortest path.
  - **VelFactor, AccFactor, DecFactor, JerkFactor**
    - If the velocity of any Axis variable in AxesGroup exceeds the maximum axis velocity limited by the software, it will trigger an error-stop condition.
    - The maximum interpolation speed of each axis in AxesGroup equals the maximum axis velocity limited by the software multiplied by VelFactor. The maximum interpolation acceleration equals the maximum axis acceleration limited by the software multiplied by AccFactor. The maximum interpolation deceleration equals the maximum axis deceleration limited by the software multiplied by DecFactor. The maximum interpolation jerk equals the maximum axis jerk limited by the software multiplied by JerkFactor.

- The trajectory planning curve type of AxesGroup is set according to requirements. If the planning curve type is a trapezoidal curve, the JerkFactor variable does not take effect and is treated as 0.
- For JerkFactor = 0 or JerkFactor ≠ 0, the velocity, acceleration, deceleration, and jerk curves are shown in the figure below.

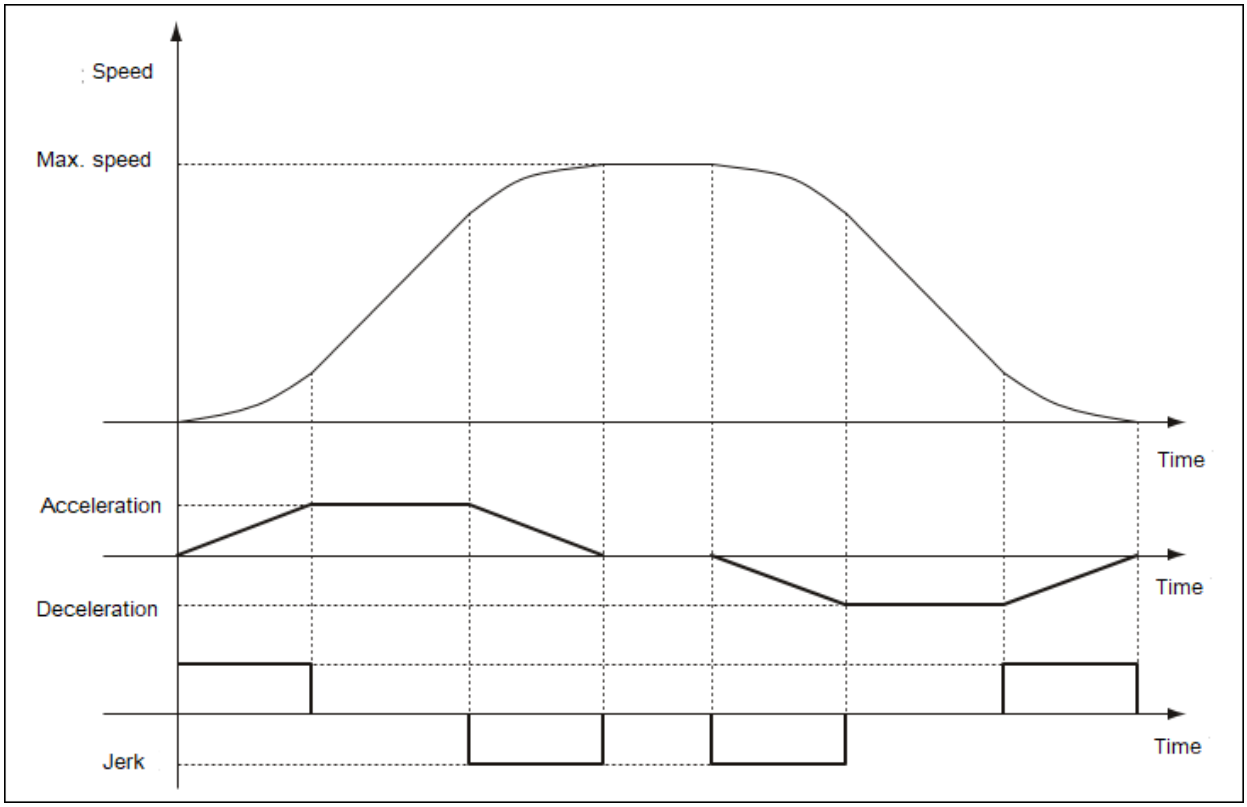
For JerkFactor = 0, the axis group can reach the maximum interpolation velocity.



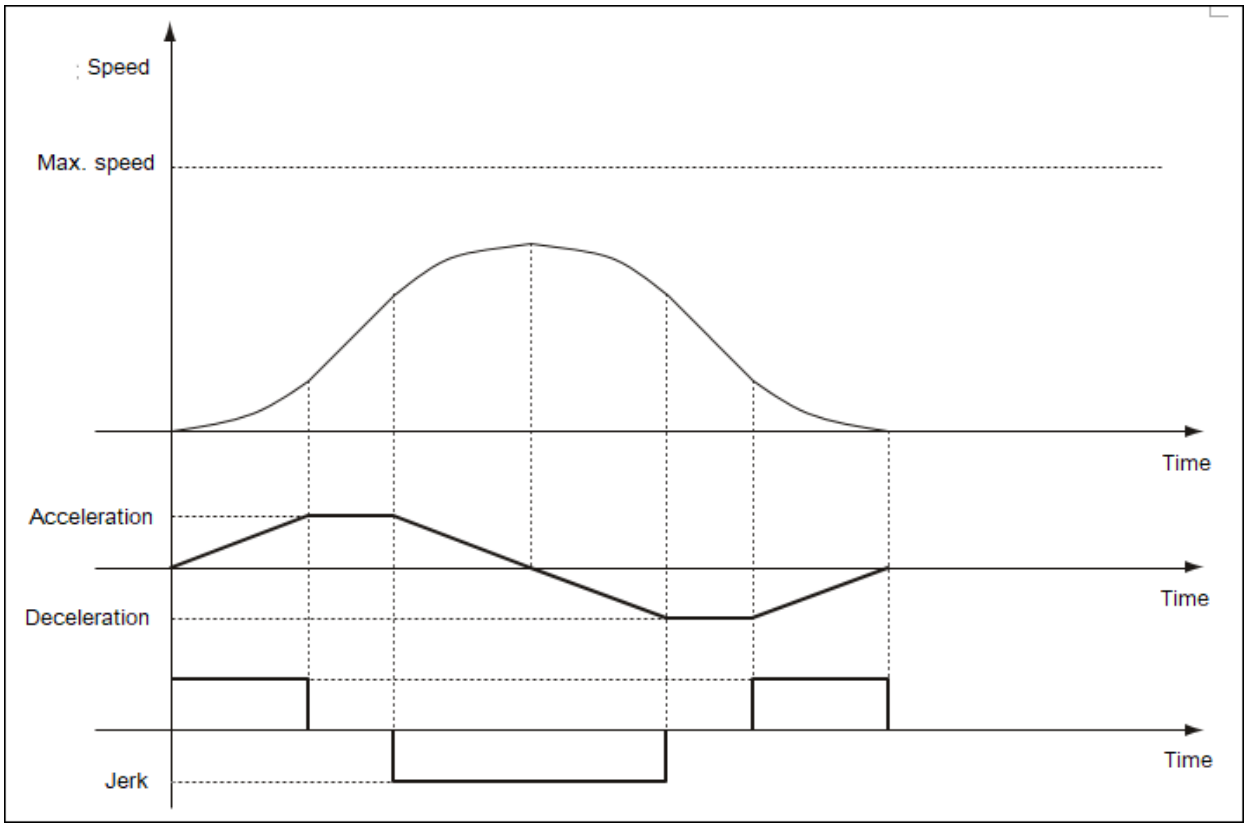
For JerkFactor = 0, the axis group cannot reach the maximum interpolation velocity due to too short travel distance.



For JerkFactor ≠ 0, the axis group can reach the maximum interpolation velocity. (Non-trapezoidal planning curve)



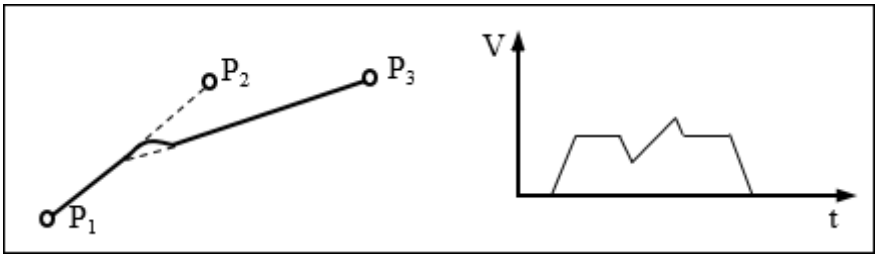
For JerkFactor ≠ 0, the axis group cannot reach the maximum interpolation velocity due to too short travel distance. (Non-trapezoidal planning curve)



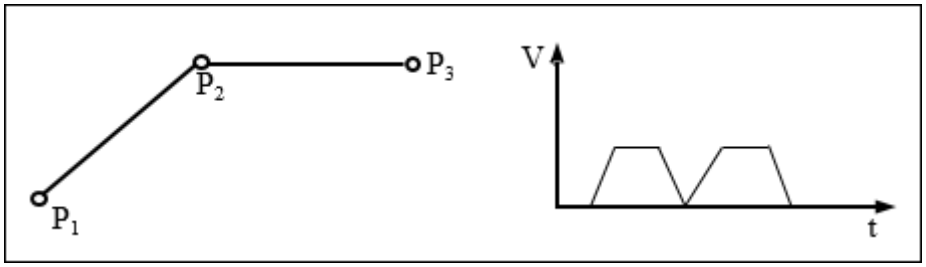
- **Re-triggering the instruction**
  - While the instruction is in motion, that is, when the instruction output parameters "Busy" and "Active" are both TRUE, this instruction cannot be re-triggered.
  - While the instruction is in buffer mode, that is, when Busy is TRUE and Active is FALSE, this instruction can be re-triggered after parameter modification. If the modified parameters are invalid, or if the buffered instruction data has already been used in motion calculations, retriggering the buffered

instruction will cause an error. Then, the axis group will enter the GroupStop state, and the currently executing instruction will also report an error.

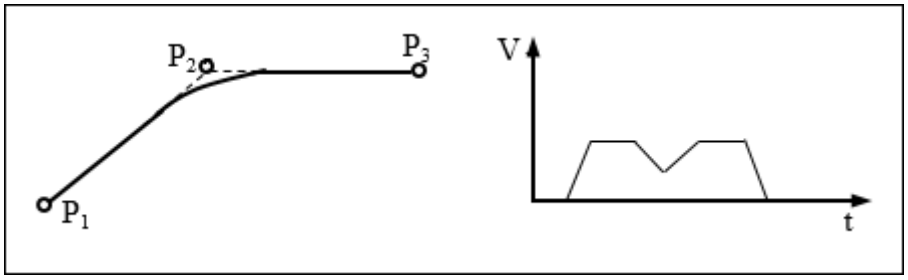
- **Multiple execution of motion instructions**
  - When this instruction is issued while another motion instruction is being executed, the new instruction can either be switched to immediately or buffered for later execution. Each axis group can buffer up to 8 instructions. In the case of multiple executions, the behavior of this instruction is determined by BufferMode.
  - BufferMode
    - Aborting: MC_ABORTING aborts the currently executing instruction and begins executing this instruction. The abort operation supports three modes, configurable via TransitionMode: TMNone, TMResetAbort, and TMDecAbort. For the meanings of these modes, see the MC_MoveLinearXX instruction description. The following figure illustrates the TMResetAbort mode:



    - Buffer: MC_BUFFERED, this instruction is buffered and executed after the current instruction is completed.



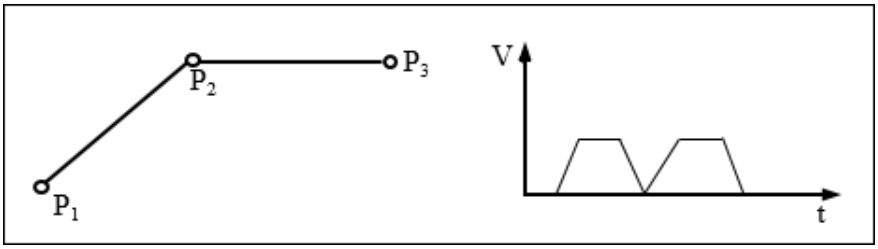
    - Blending: MC_BLEND, the velocities of the two adjacent instructions are blended in the specified manner. This manner is specified in TransitionMode.



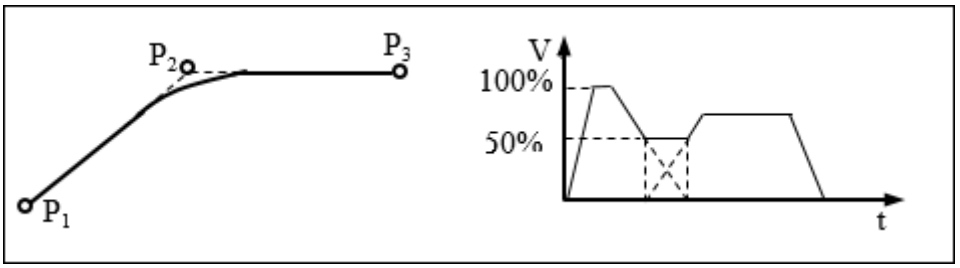
  - TransitionMode

    This variable specifies how to blend the velocities of two instructions when Blending is specified for the BufferMode.

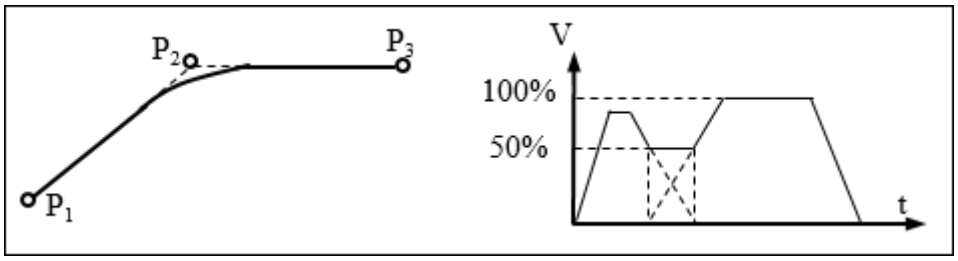
    - TMNone: Insert no transition curve. If TMNone is selected in a blending mode, the result is consistent with the transition between the velocities of the two instructions in buffered mode.



- TMStartVelocity: Transition with given start velocity. Blending transition at a certain ratio (0, 1] of the actual maximum velocity of the current motion control instruction. Effective for transitions between any motion control instructions.



- TMDefinedVelocity: Transition with defined velocity. Blending transition at a certain ratio (0, 1] of the actual maximum velocity of the instruction following the current motion control instruction. Effective for transitions between any motion control instructions.



- **Precautions**
  - This instruction supports point-to-point relative motion in the ACS coordinate system and static PCS coordinates, but does not support point-to-point relative motion in the dynamic PCS coordinates. Motions in both coordinate systems are transformed into the ACS coordinate system for execution. If the target point exceeds any axis limit, an error is generated when this instruction is initiated.
  - When the current motion instruction is interrupted, Direct motion constructs a transition trajectory in joint space along the direction of the velocity at the breakpoint to ensure smooth connection with the subsequent trajectory. This ensures velocity continuity only. However, the acceleration continuity is not guaranteed. For details on the three transition modes supported for aborting, see the motion abort description.
  - The transition trajectory calculation for the aborting instruction is planned according to the acceleration segment parameters set in the instruction. The higher the acceleration segment parameter value, the shorter the transition trajectory and the faster the transition.
  - In dynamic coordinate system, the point-to-point motion is prohibited. Otherwise, the function block will report an error.
  - During the part coordinate system switching process, the point-to-point motion is prohibited before activation. Otherwise, the function block reports an error.
  - This instruction can only be initiated when the axis group is in the GroupMoving or GroupStandby state. Attempting to start this instruction in any other state will result in an error.
  - During the execution of this instruction, it can be interrupted by other motion instructions, MC_GroupStop, or MC_GroupHalt.

- After the execution of this instruction, the axis group (AxesGroup) state is changed from GroupMoving to GroupStandby.
- When this instruction is used in a dynamic coordinate system, the blending transition mode in the PCS coordinate system is not recommended. Due to the tracking of the synchronous belt movement, the execution of this instruction in such scenarios (blending transitions in the PCS) may introduce a dynamic deviation between the programmed point and the actual achieved point in absolute space.

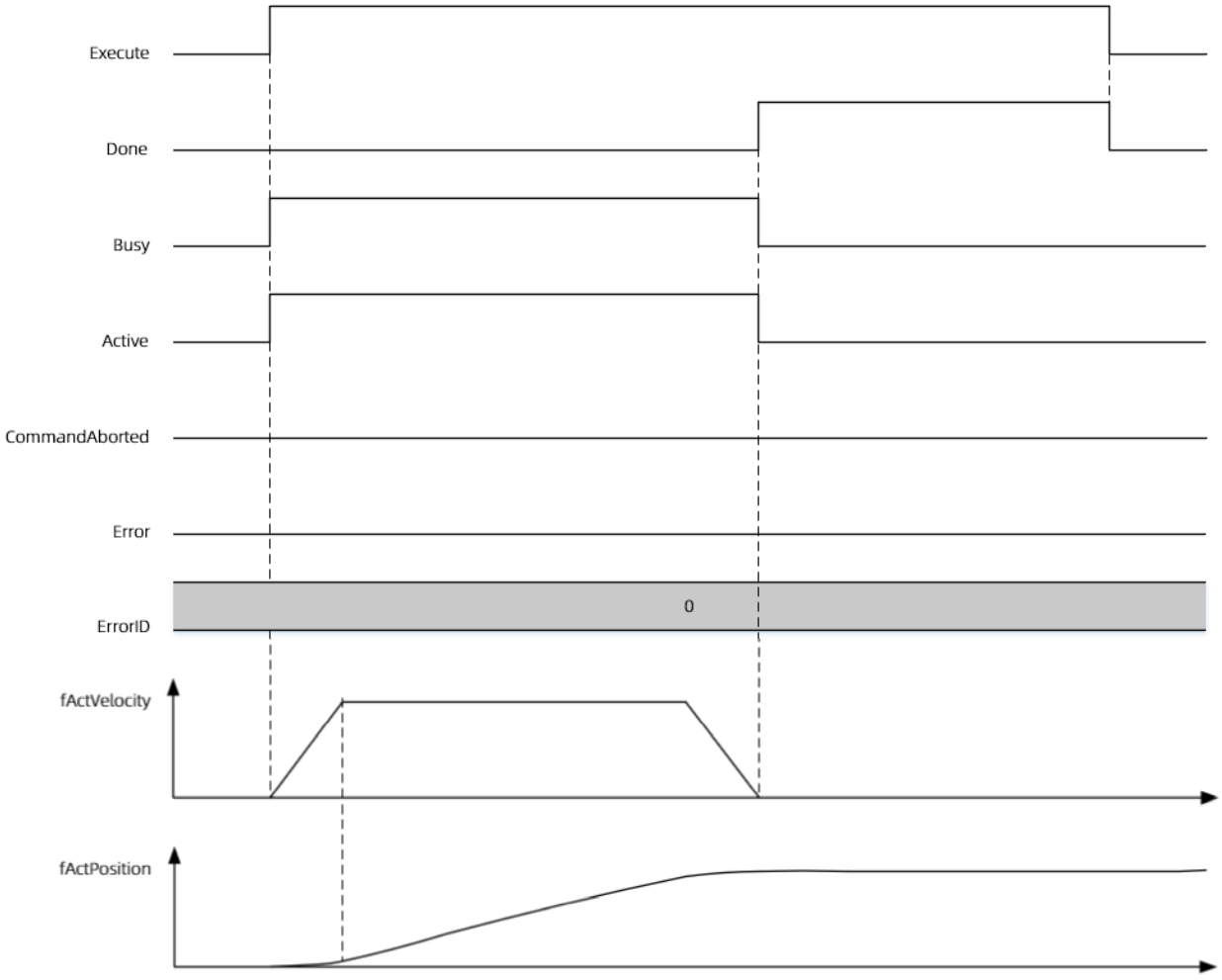
- **Program example**

  The example program for this instruction can refer to the 18.3.3.5.20 MC_MoveDirectRelative instruction. This instruction can also be combined with other AxesGroup motion instructions.

- **Sequence diagram**

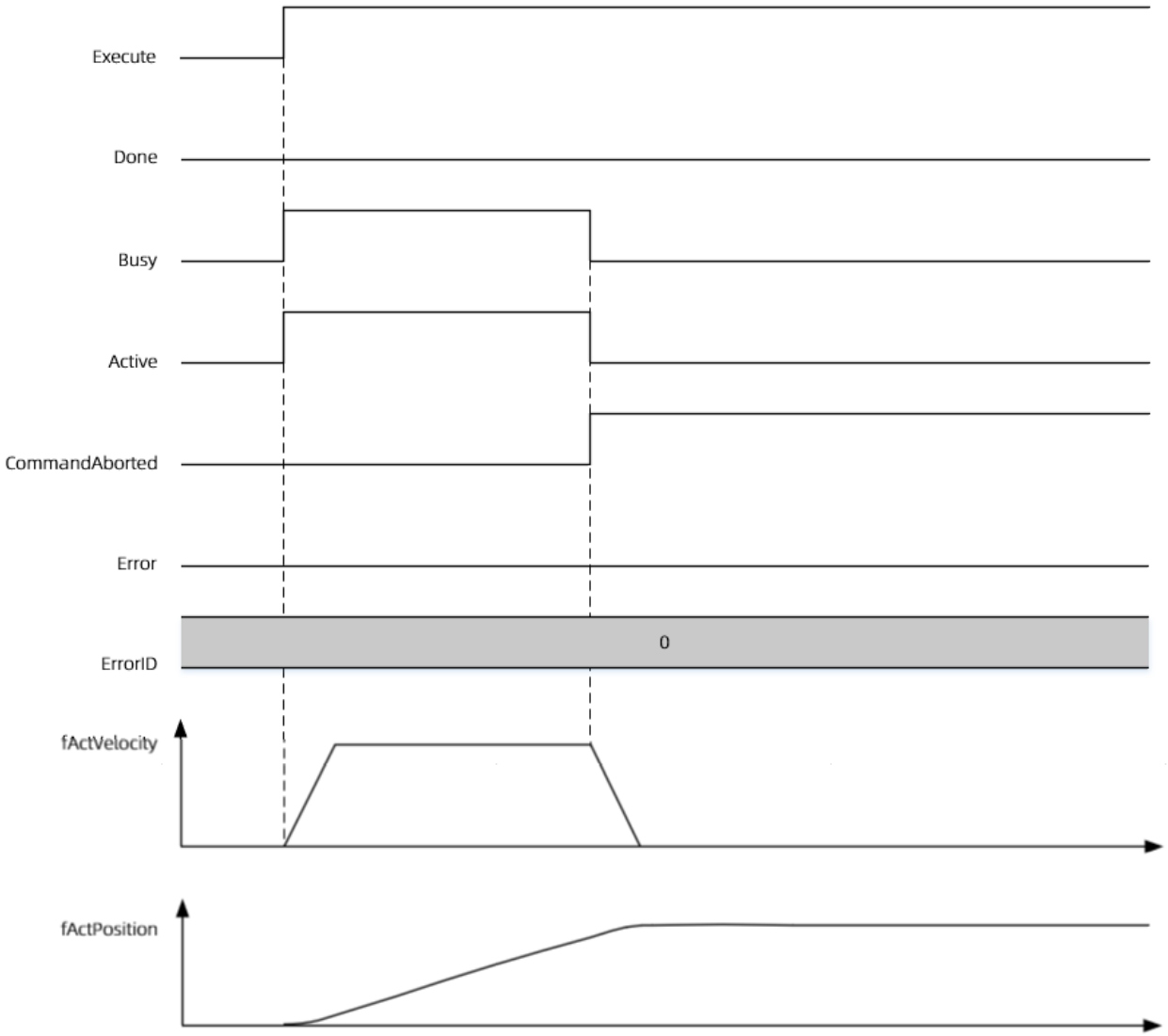
  - The instruction runs normally.

    When Execute is triggered, the Busy and Active output become TRUE. When the target position is reached, the axis group velocity changes to 0, Done becomes TRUE, and Busy and Active become FALSE.



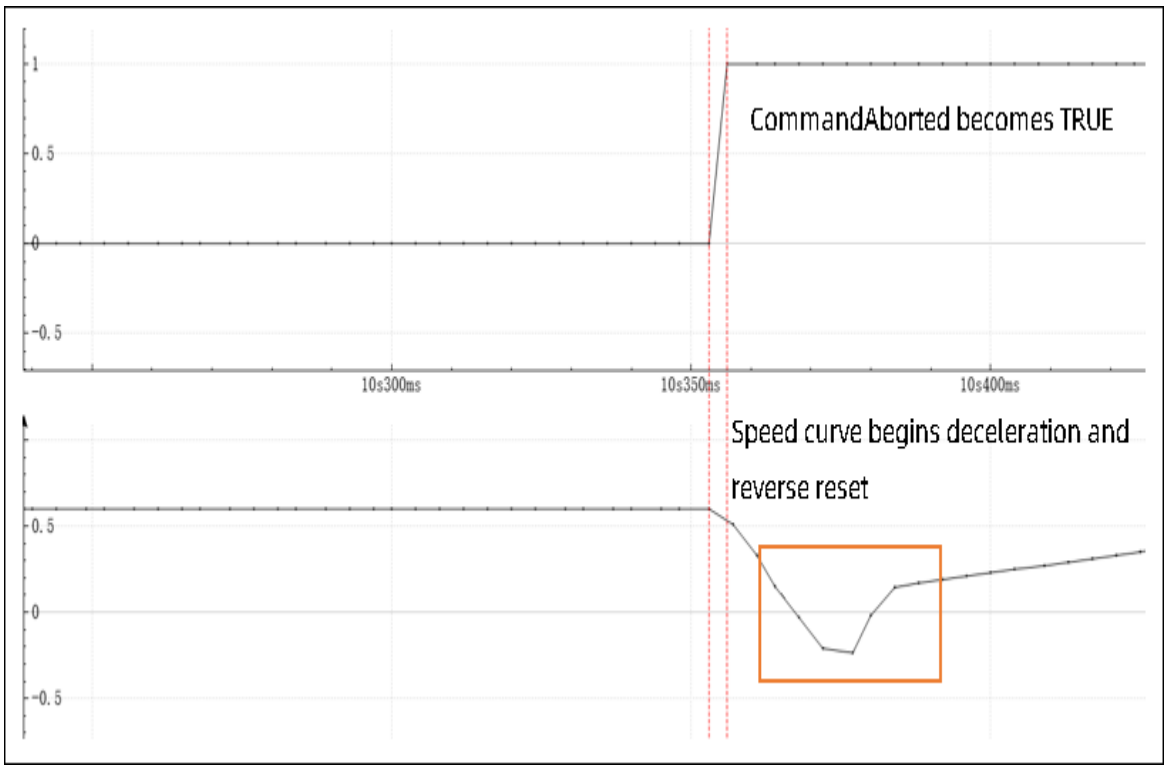
  - The instruction is aborted by other instructions.

    When Execute is triggered, the Busy and Active output become TRUE. When this instruction is aborted by another instruction, CommandAborted becomes TRUE, and Busy and Active become FALSE.



As noted in the precautions, when a motion instruction is aborted, the point-to-point absolute motion immediately replans from the interruption point. In joint space, a deceleration-reversal

transition trajectory is generated along the velocity direction at that point, ensuring smooth blending with the subsequent trajectory segment.



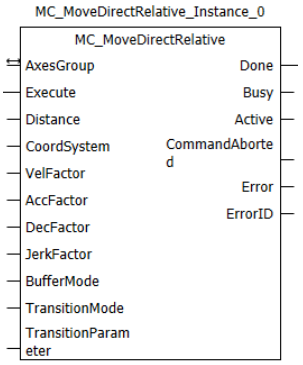
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.20 MC_MoveDirectRelative

The point-to-point instruction for the axis group commands the axis group to move from the current TCP point to a relative position without requiring a specific path.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_MoveDirectRelative	Point-to-point relative motion		MC_MoveDirectRelative_Instance_0(AxesGroup := , Execute := , Distance := , CoordSystem := , VelFactor := , AccFactor := , DecFactor := , JerkFactor := , BufferMode := , TransitionMode := , TransitionParameter := , Done => , Busy => , Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
Distance	Relative Position	ARRAY [1..GROUP_DEF.AXIS_NUM] OF LREAL	-	[0,0,…]	For the displacement components in each dimension in the specified coordinate system, see Axis group type description.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
CoordSystem	Coordinate system type	18.3.3.2.4.19 MC_COORDSYSTEM: Coordinate System Enumeration	ACS, PCS	ACS	Select the application coordinate system type.
VelFactor	Velocity factor	LREAL	(0,1]	0.10	Maximum velocity factor. This value may not necessarily be reached. If the value is greater than 1, it is corrected to 1. If the value is less than or equal to 0, it is corrected to 1e-5.
AccFactor	Acceleration factor	LREAL	(0,1]	0.10	Maximum acceleration factor. This value may not necessarily be reached. If the value is greater than 1, it is corrected to 1. If the value is less than or equal to 0, it is corrected to 0.1.
DecFactor	Deceleration factor	LREAL	(0,1]	0.10	Maximum deceleration factor. This value may not necessarily be reached. If the value is greater than 1, it is corrected to 1. If the value is less than or equal to 0, it is corrected to 0.1.
JerkFactor	Jerk factor	LREAL	[0,1]	0.10	Maximum jerk factor. This value may not necessarily be reached. For the non-trapezoidal motion profiling, the factor will be corrected to 1 if it is greater than 1. If it is less than or equal to 0, it will be corrected to 0.1.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting 1: mcBuffered 2: mcBlendingLow 3: mcBlendingPrevious 4: mcBlendingNext 5: mcBlendingHigh	0	0: mcAborting (Aborting). 1: mcBuffered (Buffered). 2: mcBlendingLow (Blending with the lowest velocity of both FBs). 3: mcBlendingPrevious (Blending with the velocity of the first FB). 4: mcBlendingNext (Blending with velocity of the second FB). 5: mcBlendingHigh (Blending with the highest velocity of both FBs). Note: Buffer modes 2 to 5 currently have the same effect. All are blending modes. The behavior depends on the TransitionMode and TransitionParameter parameters.
TransitionMode	Transition mode	18.3.3.2.4.10 MC_TRANSITION_MODE: Transition Mode Enumeration	0: TMNone 1: TMStartVelocity 10: TMDefinedVelocity 11: TMResetAbort 12: TMDecAbort	0	Definition of the transition trajectory characteristic types. When BufferMode is set to MC_ABORTING, only the transition mode parameters TMNone, TMResetAbort, and TMDecAbort are allowed. When BufferMode is set to MC_BUFFERED, the transition mode is invalid. When BufferMode is a blending mode (2 to 5), only TMNone, TMStartVelocity, and TMDefinedVelocity are allowed.
TransitionParameter	Transition parameter	LREAL	Velocity [0.0,1.0]	1.0	Transition parameters vary with the transition mode.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Active	Instruction has controlled the axis group	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

Input Variable	Name	Correction Description for Values Greater Than 1 or Backend Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
VelFactor	Velocity factor	If the velocity factor is greater than 1, it is automatically corrected to 1.	If the velocity factor is less than 0, it is automatically corrected to 1e-5 for operation.	If the velocity factor is equal to 0, it is automatically corrected to 1e-5 for operation.
AccFactor	Acceleration factor	If the acceleration factor is greater than 1, it is automatically corrected to 1.	If the acceleration factor is less than 0, it is automatically corrected to 0.1 for operation.	If the acceleration factor is equal to 0, it is automatically corrected to 0.1 for operation.
DecFactor	Deceleration factor	If the deceleration factor is greater than 1, it is automatically corrected to 1.	If the deceleration factor is less than 0, it is automatically corrected to 0.1 for operation.	If the deceleration factor is equal to 0, it is automatically corrected to 0.1 for operation.
JerkFactor	Jerk factor	If the jerk factor is greater than 1 in the non-trapezoidal motion profiling, it is automatically corrected to 1.	If the jerk factor is less than 0 in the non-trapezoidal motion profiling, it is automatically corrected to 0.1 for operation.	If the jerk factor is equal to 0 in the non-trapezoidal motion profiling, it is automatically corrected to 0.1 for operation.

- **Function**
  - This instruction is executed on the rising edge of Execute and performs point-to-point relative motion from two axes to four axes.

- The parameter values for Distance, CoordSystem, VelFactor, AccFactor, DecFactor, JerkFactor, BufferMode, TransitionMode and TransitionParameter can be specified in the input variables.
- When this instruction is initiated, the target position is determined by adding the input variable Distance (relative position) to the starting position of AxesGroup.
- For the instruction, blending modes 2, 3, 4, and 5 for BufferMode have the same behavior. The transition effect depends on the TransitionMode and TransitionParameter settings.
- Supports the same-cycle triggering of instructions. The Done output signal of the instruction serves as the trigger signal for Execute.
- **CoordSystem**
  - Specifies the coordinate system for point-to-point relative motion.
- **Distance (Relative position coordinate)**
  - For the axis type corresponding to each position vector in the Distance array, see Axis group type description.
  - When the coordinate system type is set to PCS, the ABC rotary axes of the coordinate system correspond to the ZYX-type Euler angles, respectively. If the configuration supports orientation motion, the configured ABC relative positions must be within ±180°, otherwise an error will be generated.
  - In relative motion mode, the rules for calculating the target position of the axis group are shown in the table below.

Coordinate system type (calculation of target positions for each axis in the existing axis group configuration)	
ACS	PCS
A1 axis target position = A1 axis current position + Distance [1]	X-axis target position = X-axis current position + Distance [1]
A2 axis target position = A2 axis current position + Distance [2]	Y-axis target position = Y-axis current position + Distance [2]
A3 axis target position = A3 axis current position + Distance [3]	Z-axis target position = Z-axis current position + Distance [3]
A4 axis target position = A4 axis current position + Distance [4]	A-axis target position = A-axis current position + Distance [4]

  - If the A-axis target position is not within ±180°, the system will make the following adjustments: if the A-axis target position is greater than 180°, the system will subtract 360° until the result falls within ±180°; if the A-axis target position is less than -180°, the system will add 360° until the result falls within ±180°. Example: $A_{\text{calculated target position}} = 260°$, $A_{\text{converted target position}} = -100°$.
  - If the target position exceeds any joint axis limit, starting this instruction will cause an error.
  - If the configuration supports orientation motion, the axis group will move to the target orientation along the shortest path.
- **VelFactor, AccFactor, DecFactor, JerkFactor**
  - If the velocity of any Axis variable in AxesGroup exceeds the maximum axis velocity limited by the software, it will trigger an error-stop condition.
  - For each axis variable in AxesGroup:

Maximum interpolation velocity = Maximum axis velocity limited by the software x VelFactor.

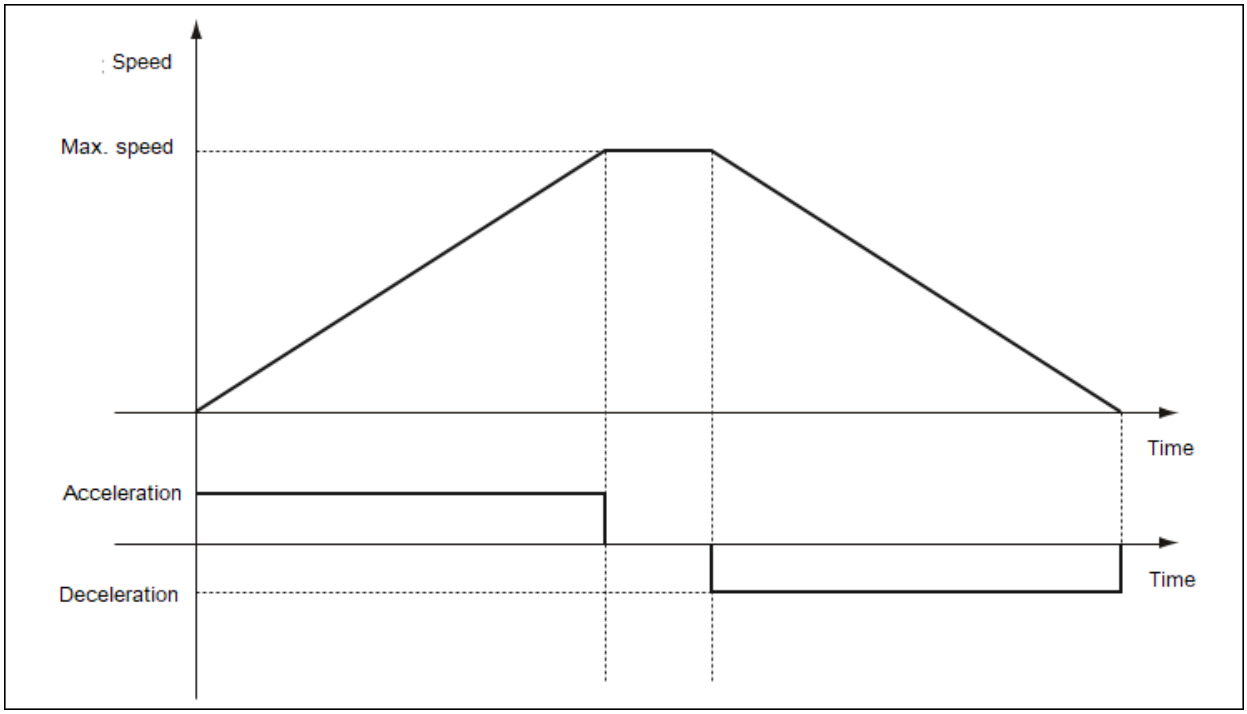
Maximum interpolation acceleration = Maximum axis acceleration limited by the software x AccFactor.

Maximum interpolation deceleration = Maximum axis deceleration limited by the software x DecFactor.

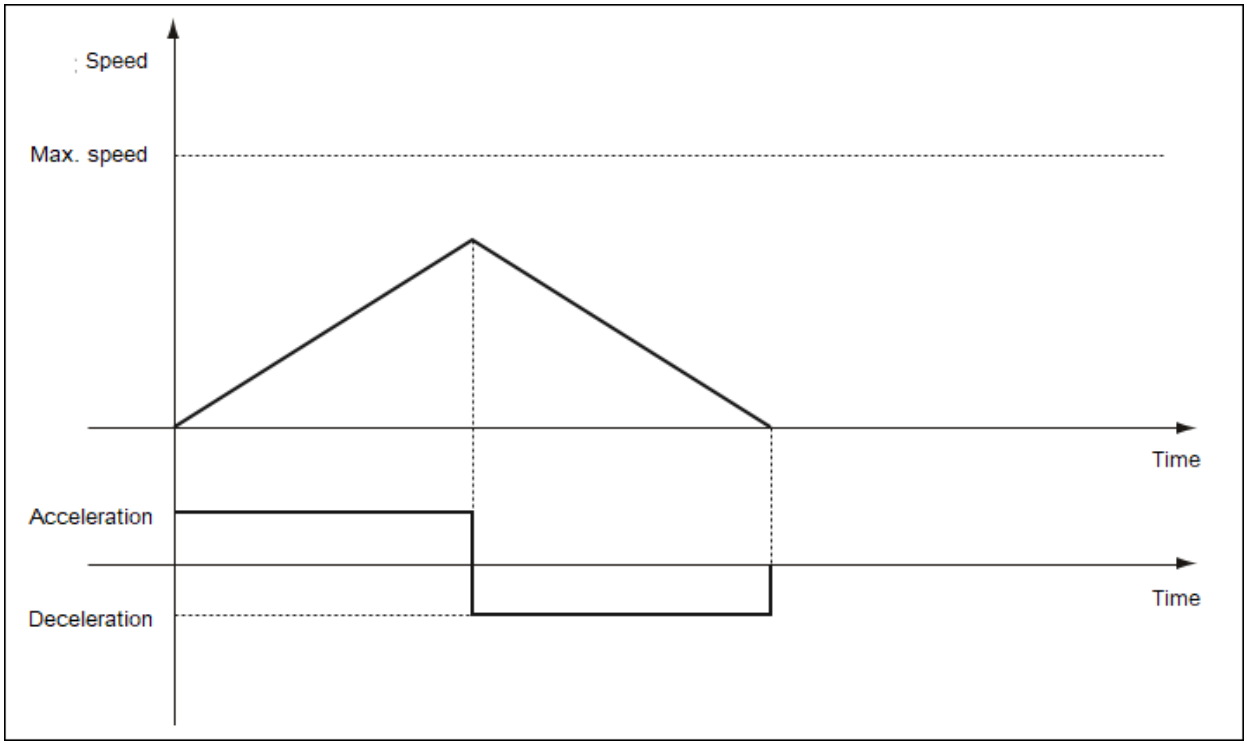
Maximum interpolation jerk = Maximum axis jerk limited by the software x JerkFactor.

- The trajectory planning curve type of AxesGroup is set according to requirements. If the planning curve type is a trapezoidal curve, the JerkFactor variable does not take effect and is treated as 0.
- For JerkFactor = 0 or JerkFactor ≠ 0, the velocity, acceleration, deceleration, and jerk curves are shown in the figure below.

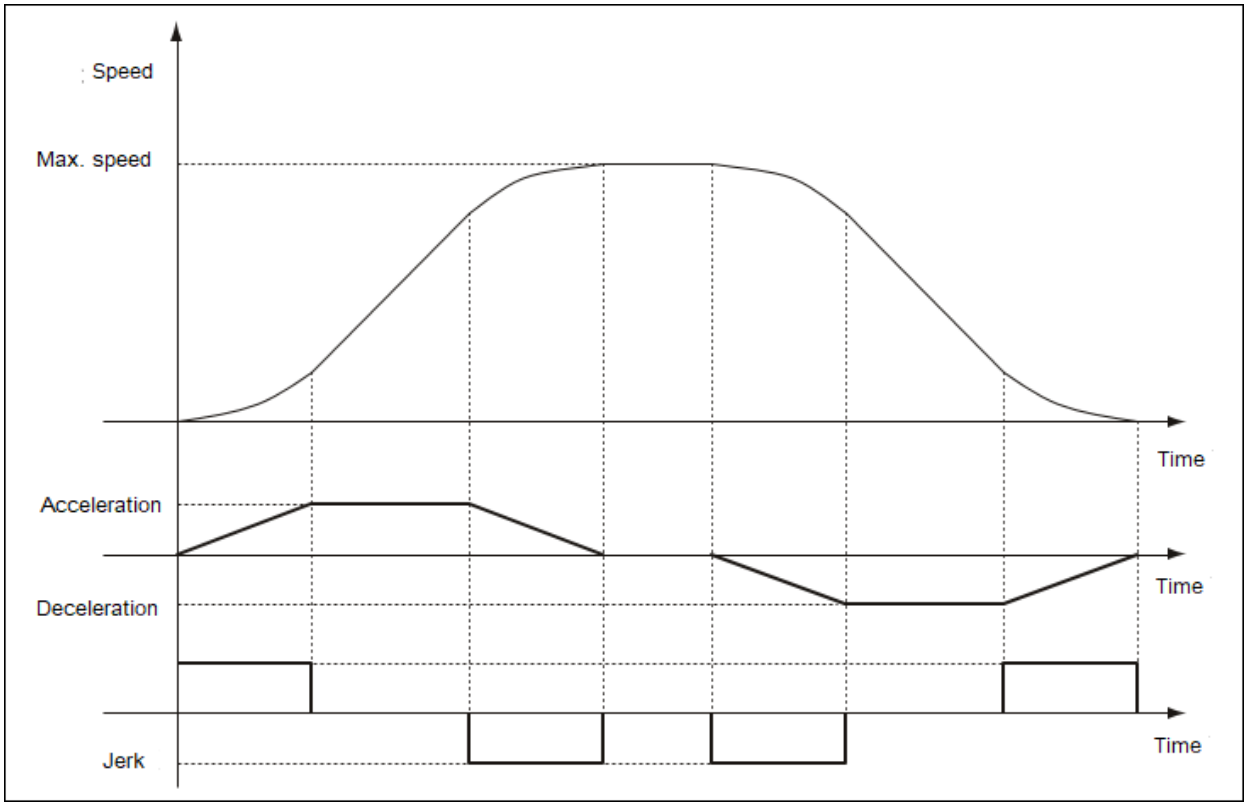
  For JerkFactor = 0, the axis group can reach the maximum interpolation velocity.



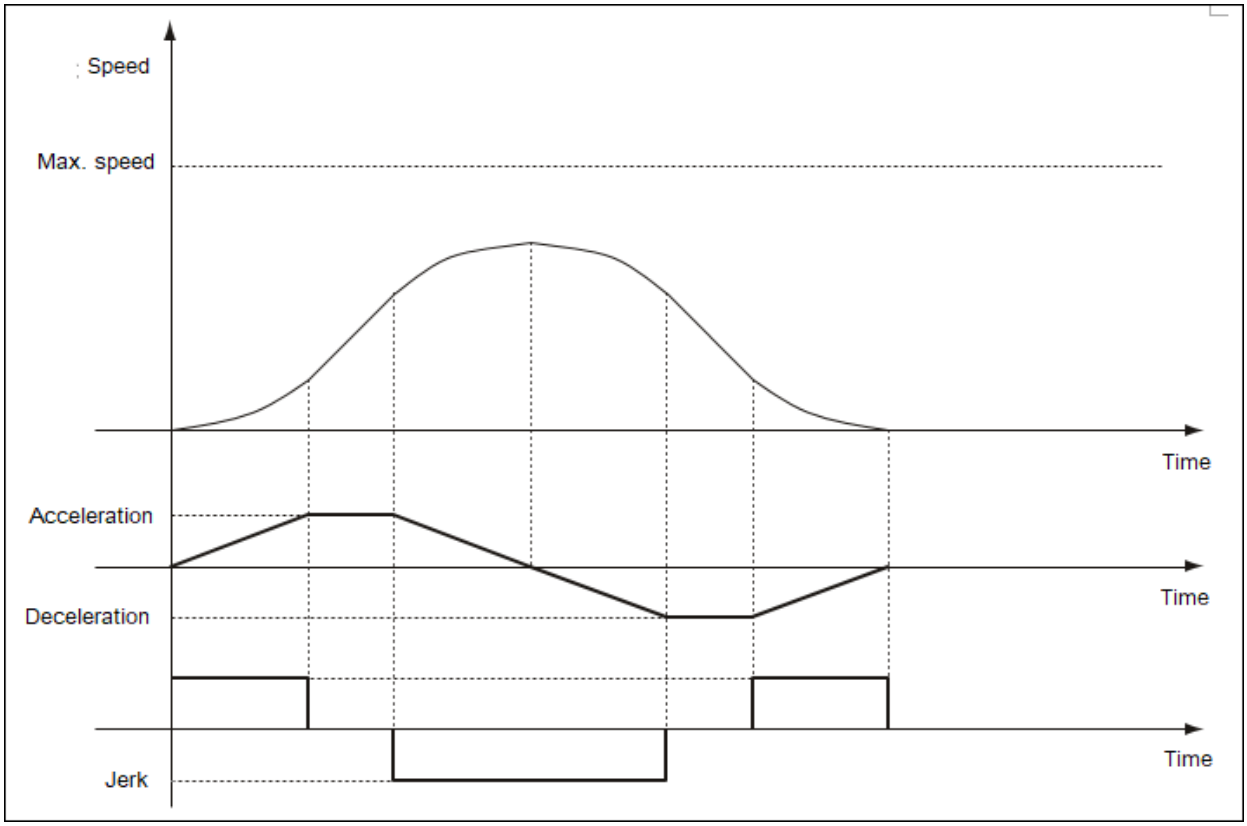
For JerkFactor = 0, the axis group cannot reach the maximum interpolation velocity due to too short travel distance.



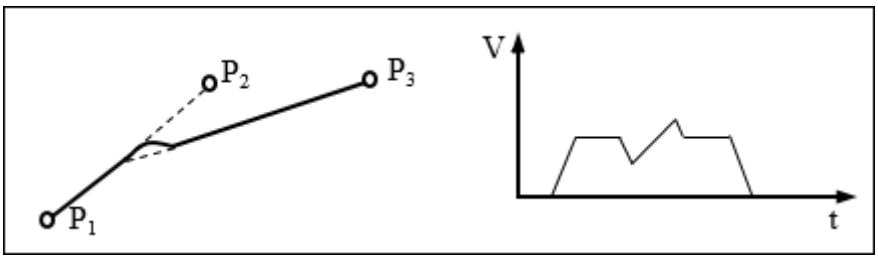
For JerkFactor ≠ 0, the axis group can reach the maximum interpolation velocity. (Non-trapezoidal planning curve)



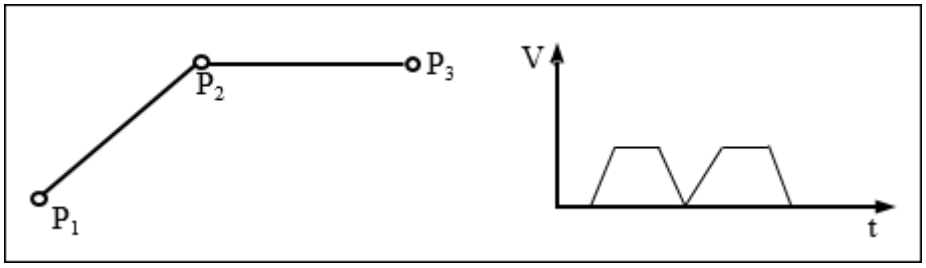
For JerkFactor ≠ 0, the axis group cannot reach the maximum interpolation velocity due to too short travel distance. (Non-trapezoidal planning curve)



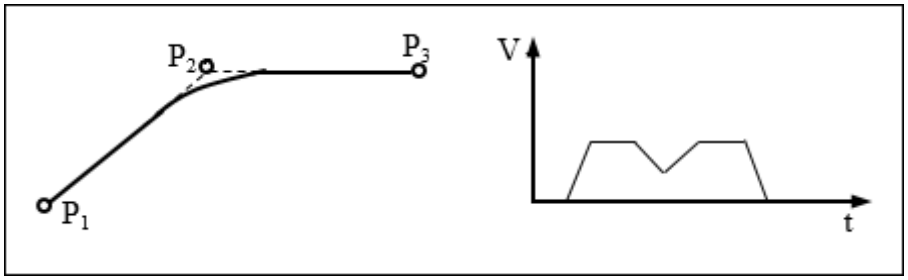
- **Re-triggering the instruction**
  - While the instruction is in motion, that is, when the instruction output parameters "Busy" and "Active" are both TRUE, this instruction cannot be re-triggered.
  - While the instruction is in buffer mode, that is, when Busy is TRUE and Active is FALSE, this instruction can be re-triggered after parameter modification. If the modified parameters are invalid, or if the buffered instruction data has already been used in motion calculations, retriggering the buffered instruction will cause an error. Then, the axis group will enter the GroupStop state, and the currently executing instruction will also report an error.
- **Multiple execution of motion instructions**
  - When this instruction is issued while another motion instruction is being executed, the new instruction can either be switched to immediately or buffered for later execution. Each axis group can buffer up to 8 instructions. In the case of multiple executions, the behavior of this instruction is determined by BufferMode.
  - BufferMode
    - Aborting: MC_ABORTING aborts the currently executing instruction and begins executing this instruction. The abort operation supports three modes, configurable via TransitionMode: TMNone, TMResetAbort, and TMDecAbort. For the meanings of these modes, see the MC_MoveLinearXX instruction description. The following figure illustrates the TMResetAbort mode:



    - Buffer: MC_BUFFERED, this instruction is buffered and executed after the current instruction is completed.



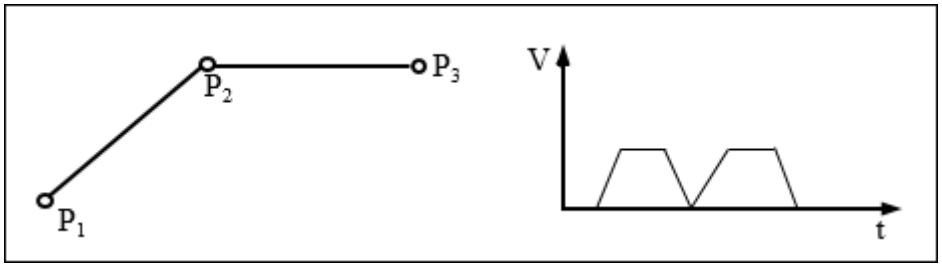
- Blending: MC_BLEND, the velocities of the two adjacent instructions are blended in the specified manner. This manner is specified in TransitionMode.



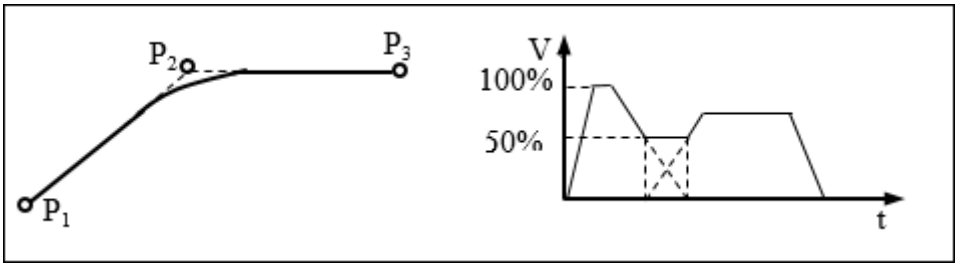
◦ TransitionMode

When BufferMode is set to blend, this instruction uses the TransitionMode input variable to specify the blending method for the velocity profiles of the two instructions.

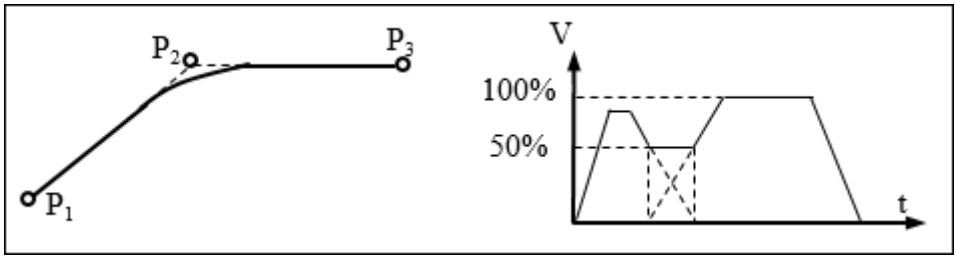
- TMNone: Insert no transition curve. If TMNone is selected in a blending mode, the result is consistent with the transition between the velocities of the two instructions in buffered mode.



- TMStartVelocity: Transition with given start velocity. Blending transition at a certain ratio (0, 1] of the actual maximum velocity of the current motion control instruction. Effective for transitions between any motion control instructions.



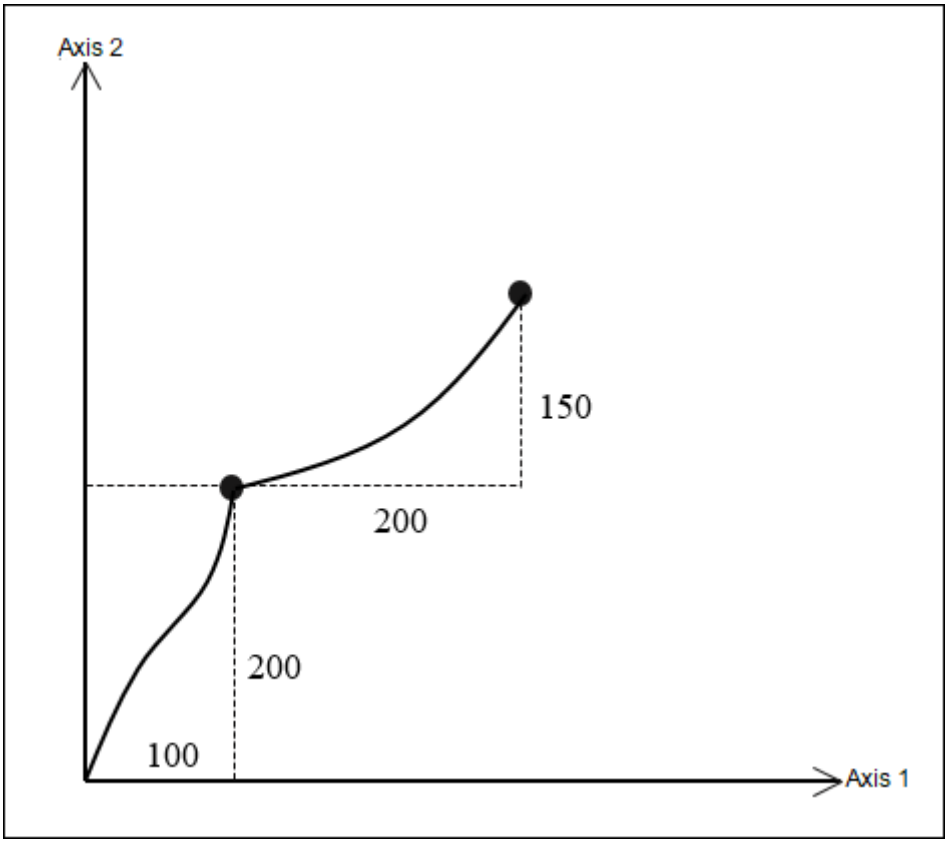
- TMDefinedVelocity: Transition with defined velocity. Blending transition at a certain ratio (0, 1] of the actual maximum velocity of the instruction following the current motion control instruction. Effective for transitions between any motion control instructions.



- **Precautions**
  - This instruction supports point-to-point relative motion in the ACS coordinate system and static PCS coordinates, but does not support point-to-point relative motion in the dynamic PCS coordinates. Motions in both coordinate systems are transformed into the ACS coordinate system for execution. If the target point exceeds any axis limit, an error is generated when this instruction is initiated.
  - When the current motion instruction is interrupted, Direct motion constructs a transition trajectory in joint space along the direction of the velocity at the breakpoint to ensure smooth connection with the subsequent trajectory. This ensures velocity continuity only. However, the acceleration continuity is not guaranteed. For details on the three transition modes supported for aborting, see the motion abort description.
  - The transition trajectory calculation for the aborting instruction is planned according to the acceleration segment parameters set in the instruction. The higher the acceleration segment parameter value, the shorter the transition trajectory and the faster the transition.
  - In dynamic coordinate system, the point-to-point motion is prohibited. Otherwise, the function block will report an error.
  - During the part coordinate system switching process, the point-to-point motion is prohibited before activation. Otherwise, the function block reports an error.
  - This instruction can only be initiated when the axis group is in the GroupMoving or GroupStandby state. Attempting to start this instruction in any other state will result in an error.
  - During the execution of this instruction, it can be interrupted by other motion instructions, MC_GroupStop, or MC_GroupHalt.
  - After the execution of this instruction, the axis group (AxesGroup) state is changed from GroupMoving to GroupStandby.
  - When this instruction is used in a dynamic coordinate system, the blending transition mode in the PCS coordinate system is not recommended. Due to the tracking of the synchronous belt movement, the execution of this instruction in such scenarios (blending transitions in the PCS) may introduce a dynamic deviation between the programmed point and the actual achieved point in absolute space.
- **Program example**

  This is an example of performing a point-to-point relative motion using a two-axis group. Starting point: (0,0); first relative displacement: (100,200); second relative displacement: (200,150)

  Two point-to-point relative motion instructions need to be executed sequentially, and the second instruction must be in MC_BUFFERED.



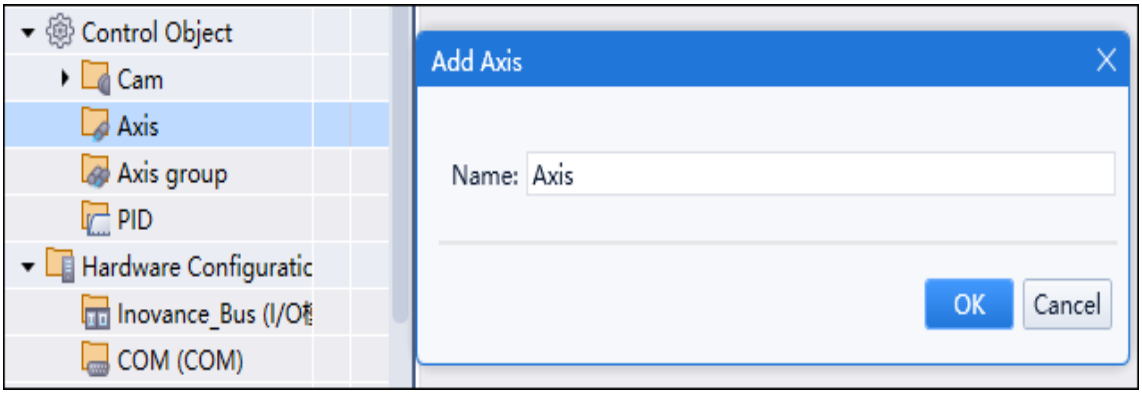
◦ Step

1. Determine the AxesGroup type and the Axis variable in the axis group in iFA.
2. Use the MC_Power instruction to enable the Axis variable in AxesGroup.
3. Use the MC_Home instruction to perform a homing operation on an Axis variable in AxesGroup.
4. Alternatively, use the MC_GroupHome instruction to perform a homing operation on an Axis variable in AxesGroup.
5. Use the MC_GroupEnable instruction to enable AxesGroup.
6. Set the parameters for the first point-to-point relative motion instruction.
7. Set the parameters for the second point-to-point relative motion instruction.
8. Start the two point-to-point relative motion instructions.

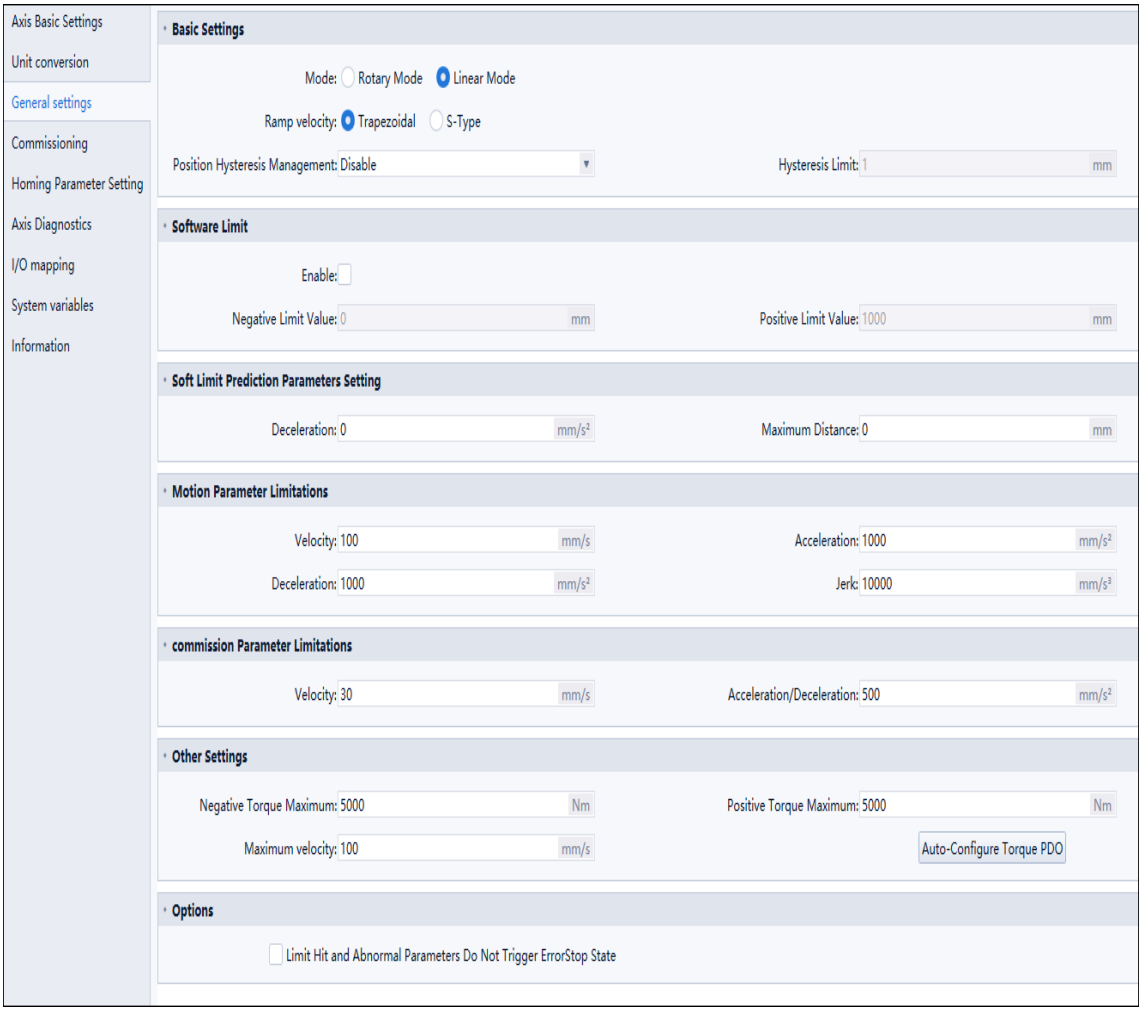
◦ Adding axes and axis groups

▪ Add axes (two axes) in FA and set the relevant axis parameters.

1. Click "Control Object > Axis" on the left project tree to add an axis.



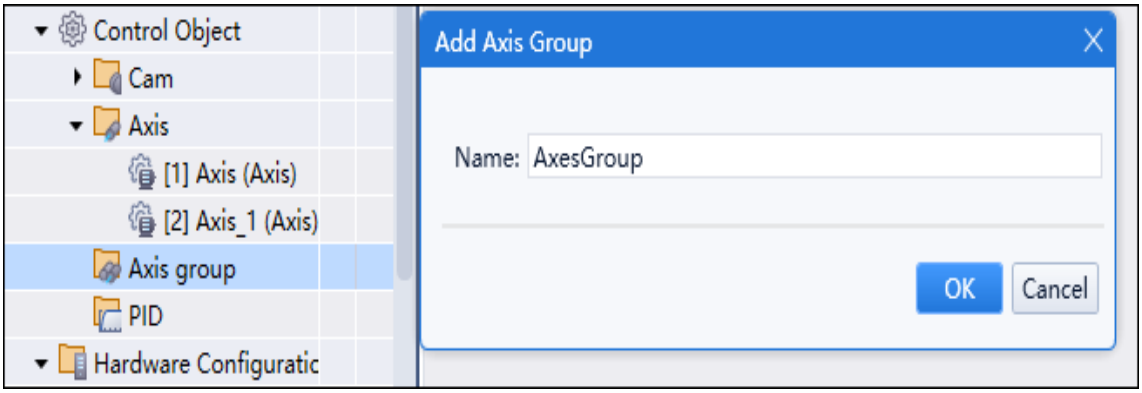
2. After adding the axes, double-click the axis to be set, enter the setting interface, and perform the corresponding configuration.



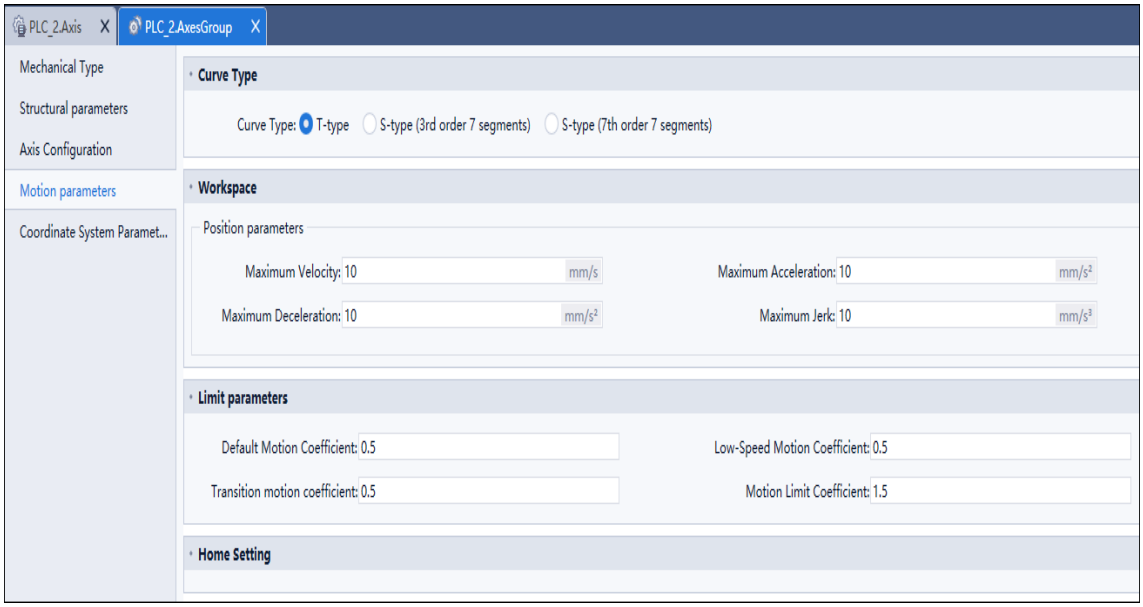
Note: The planning curve type of the axis is only valid in the axis motion instruction. When the axis is added to the axis group, the planning curve type of the axis is specified by the axis group.

- Add an axis group and set the axis group parameters.

1. Click "Control Object > Axis group" on the left project tree to add an axis group.



2. After adding the axis group, double-click the axis group to be set, enter the setting interface, and perform the corresponding configuration.

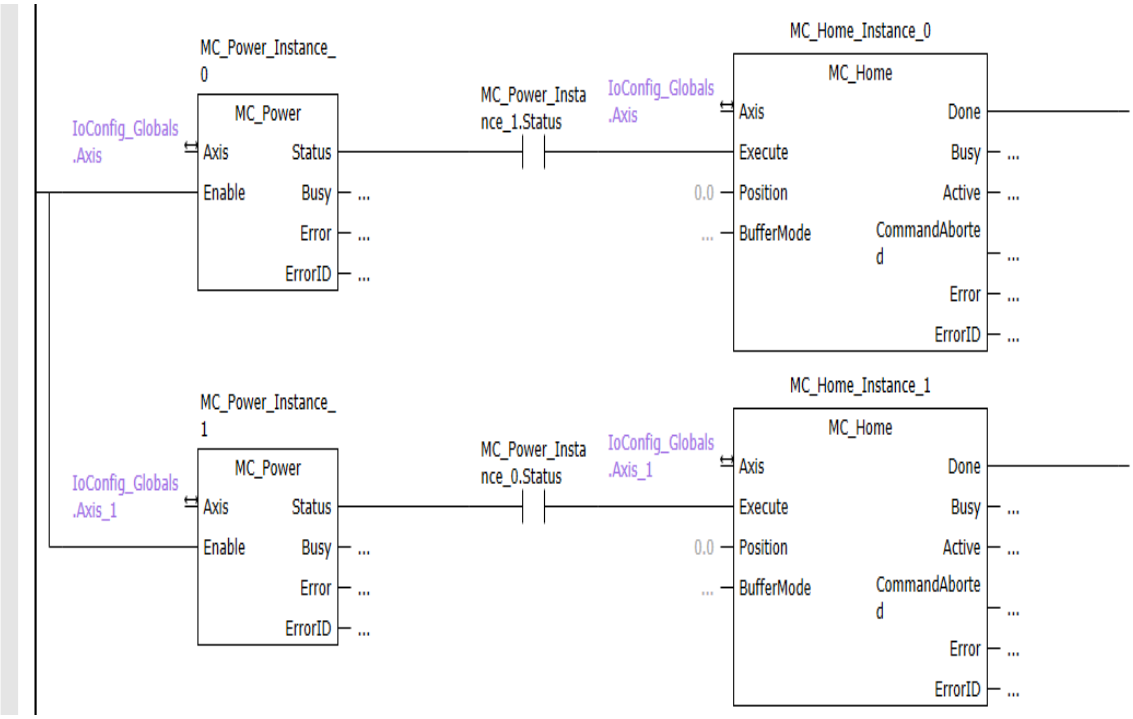


Note: The planning curve type of the axis group must be specified in the axis group structure. By default, the profile type is trapezoidal, and the jerk settings do not take effect.

◦ Ladder diagram

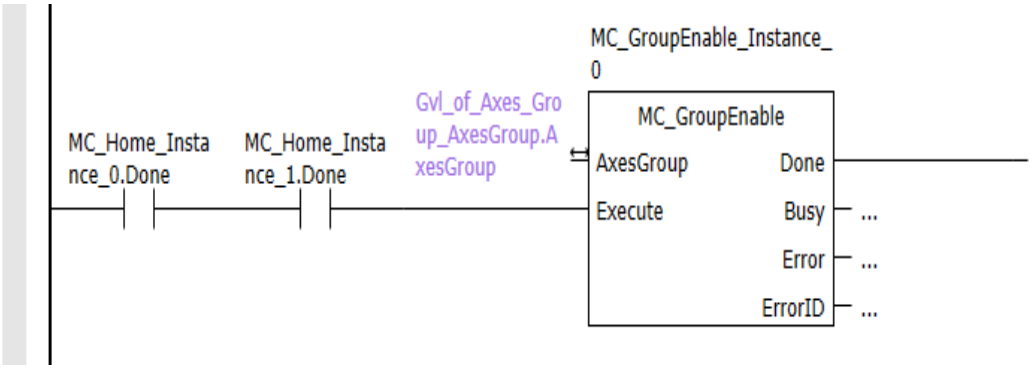
MC_Power_Instance_0 and MC_Power_Instance_1 are two instances of the MC_Power instruction, which perform enable operations on Axis and Axis_1 respectively. When MC_Power_Instance_0.Status and MC_Power_Instance_1.Status are set to TRUE, the two axes have completed the enable operation respectively. Then, the axis homing instruction is executed.

MC_Home_Instance_0 and MC_Home_Instance_1 are two instances of the MC_Home instruction, which return Axis and Axis_1 to the mechanical zero point respectively.

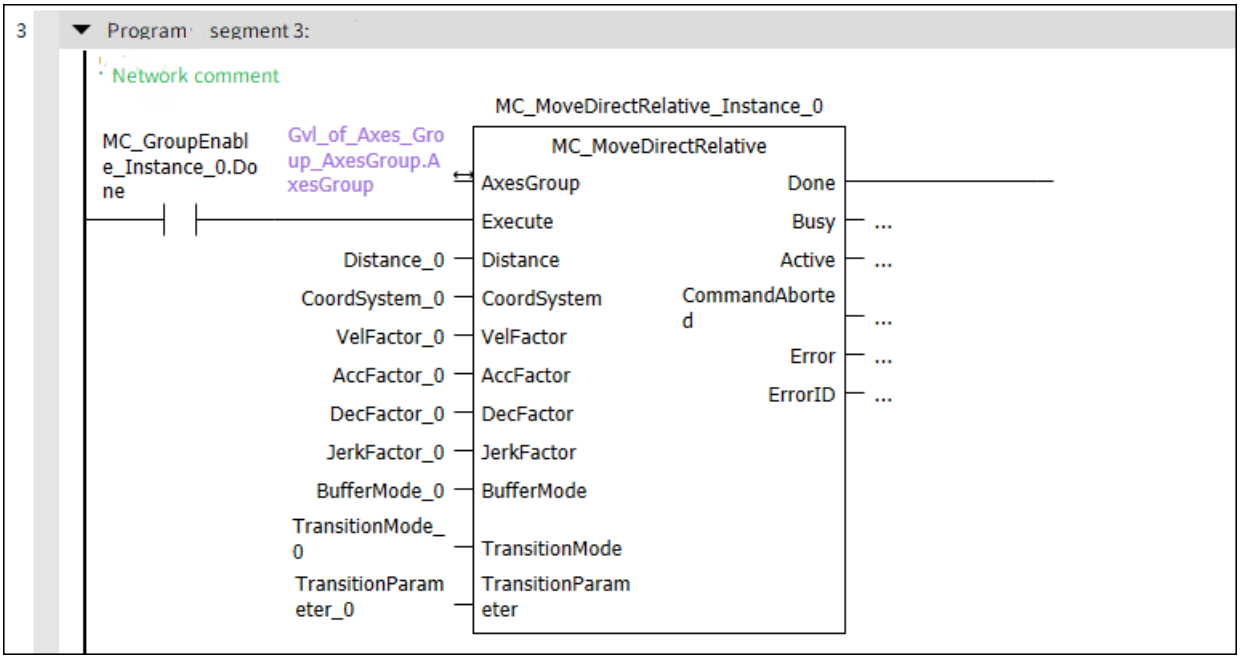


MC_Home_Instance_0.Done and MC_Home_Instance_1.Done indicate that the two axes have completed the homing instruction. Only when both axes have completed the homing instruction, the MC_GroupEnable instruction is executed for AxesGroup.

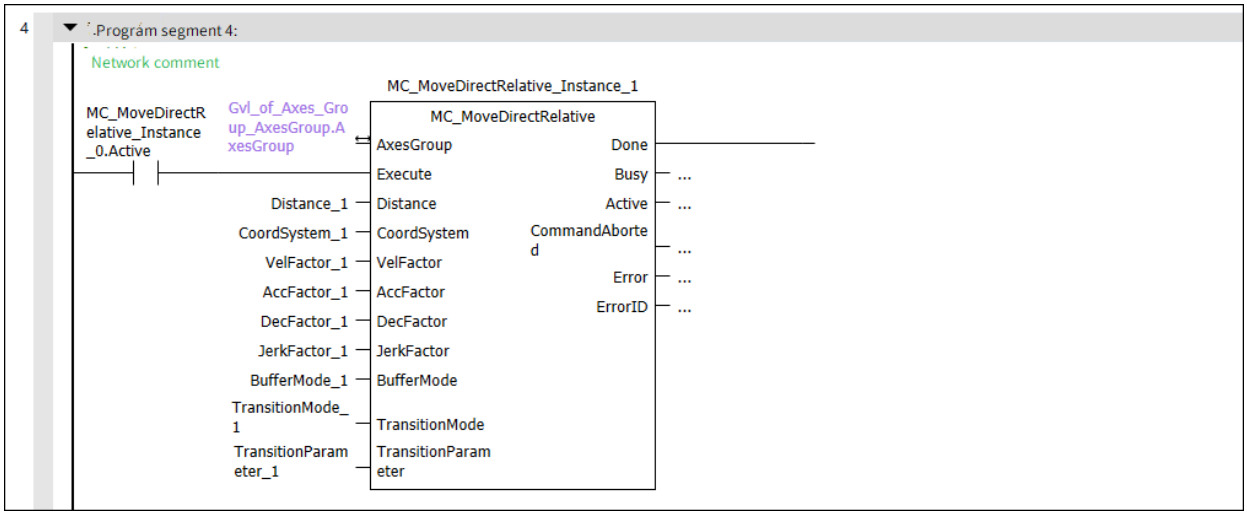
MC_GroupEnable_Instance_0 is an instance of the MC_GroupEnable instruction.



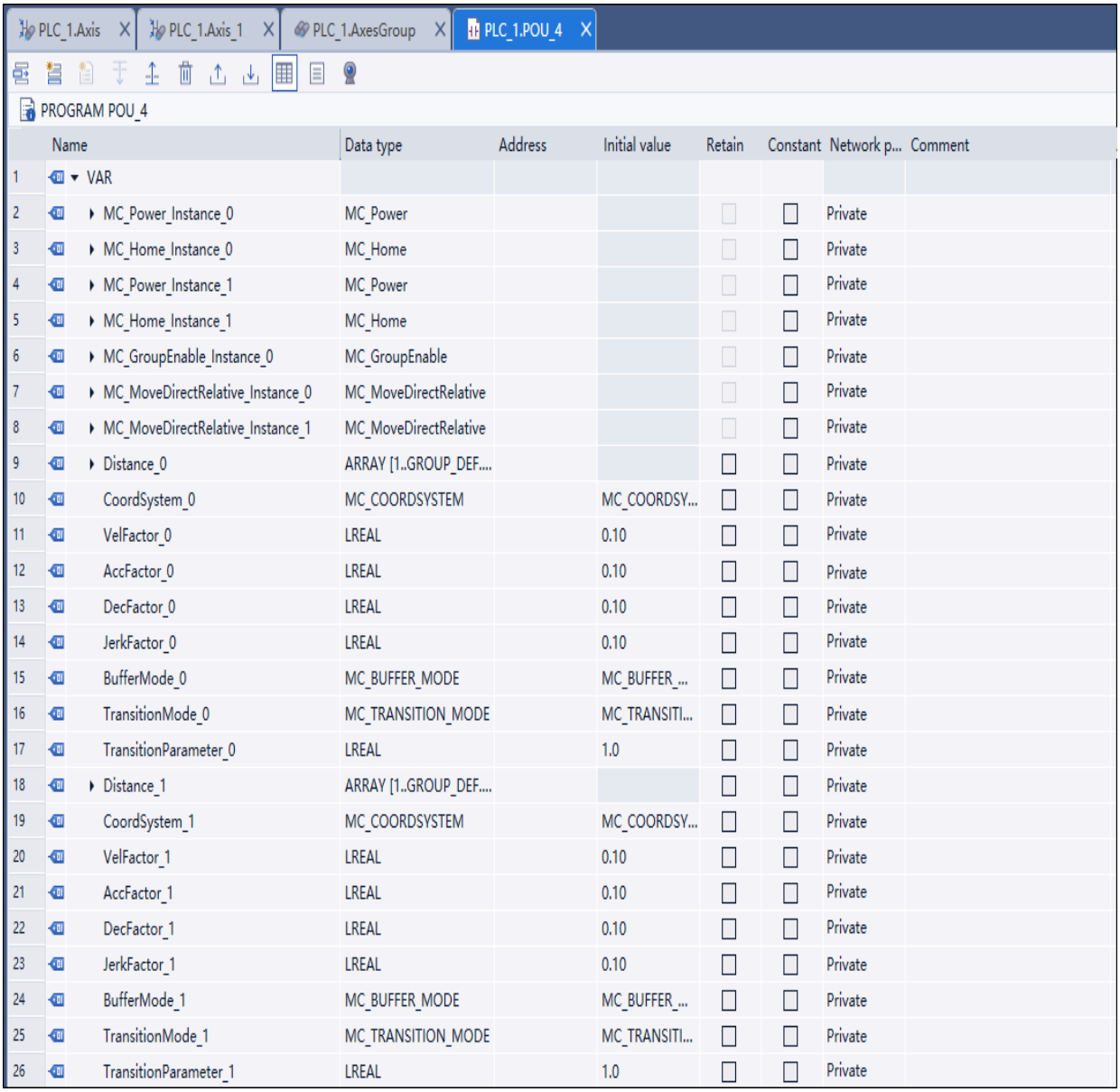
After the axis group is enabled, start the first point-to-point relative motion. MC_MoveDirectRelative_Instance_0 and MC_MoveDirectRelative_Instance_1 are two instances of the MC_MoveDirectRelative instruction.



When the first instruction MC_MoveDirectRelative_Instance_0.Active is set to TRUE, the second point-to-point relative motion instruction is initiated.



The types and parameter settings for each instruction are shown in the figure below.

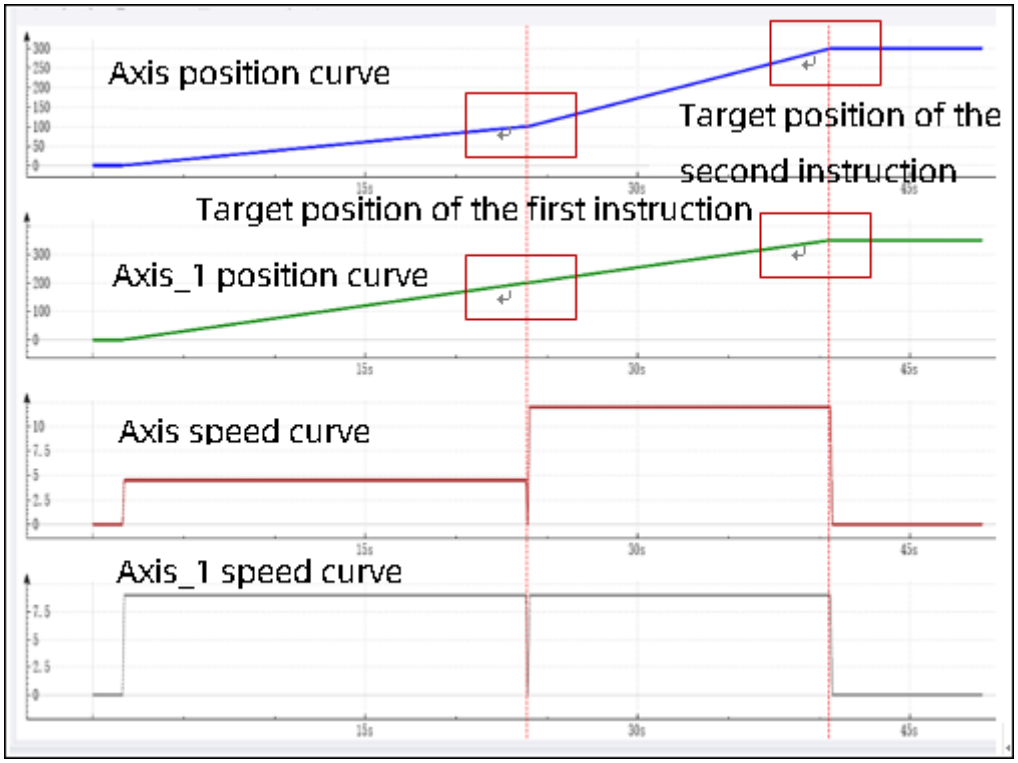


PLC_1.Axis | PLC_1.Axis_1 | PLC_1.AxesGroup | PLC_1.POU_4

PROGRAM POU_4

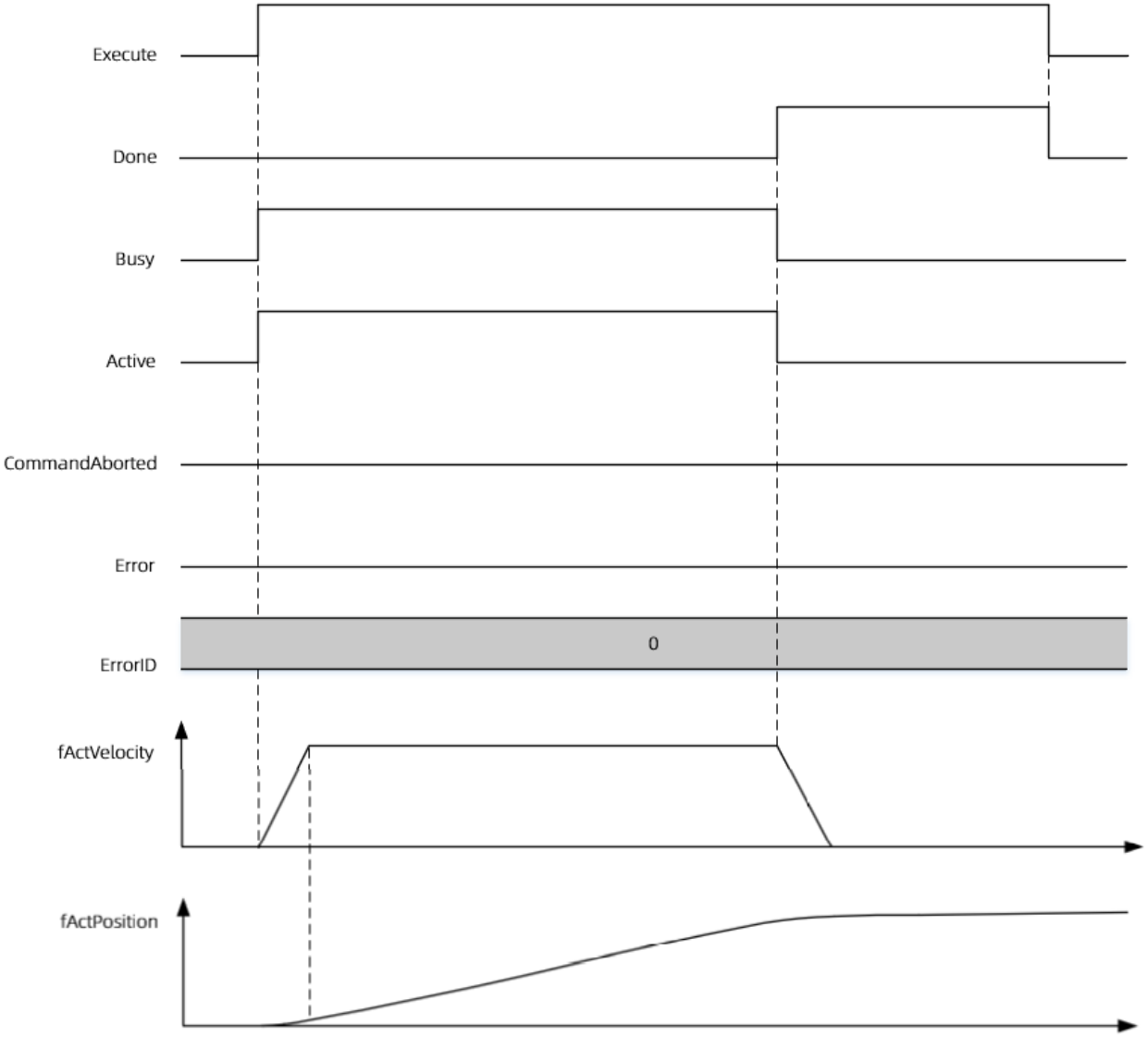
	Name	Data type	Address	Initial value	Retain	Constant	Network p...	Comment
1	VAR							
2	MC_Power_Instance_0	MC_Power					Private	
3	MC_Home_Instance_0	MC_Home					Private	
4	MC_Power_Instance_1	MC_Power					Private	
5	MC_Home_Instance_1	MC_Home					Private	
6	MC_GroupEnable_Instance_0	MC_GroupEnable					Private	
7	MC_MoveDirectRelative_Instance_0	MC_MoveDirectRelative					Private	
8	MC_MoveDirectRelative_Instance_1	MC_MoveDirectRelative					Private	
9	Distance_0	ARRAY [1..GROUP_DEF....					Private	
10	CoordSystem_0	MC_COORDSYSTEM		MC_COORDSY...			Private	
11	VelFactor_0	LREAL		0.10			Private	
12	AccFactor_0	LREAL		0.10			Private	
13	DecFactor_0	LREAL		0.10			Private	
14	JerkFactor_0	LREAL		0.10			Private	
15	BufferMode_0	MC_BUFFER_MODE		MC_BUFFER_...			Private	
16	TransitionMode_0	MC_TRANSITION_MODE		MC_TRANSITI...			Private	
17	TransitionParameter_0	LREAL		1.0			Private	
18	Distance_1	ARRAY [1..GROUP_DEF....					Private	
19	CoordSystem_1	MC_COORDSYSTEM		MC_COORDSY...			Private	
20	VelFactor_1	LREAL		0.10			Private	
21	AccFactor_1	LREAL		0.10			Private	
22	DecFactor_1	LREAL		0.10			Private	
23	JerkFactor_1	LREAL		0.10			Private	
24	BufferMode_1	MC_BUFFER_MODE		MC_BUFFER_...			Private	
25	TransitionMode_1	MC_TRANSITION_MODE		MC_TRANSITI...			Private	
26	TransitionParameter_1	LREAL		1.0			Private	

The position profiles and velocity profiles of axes in AxesGroup are shown below.



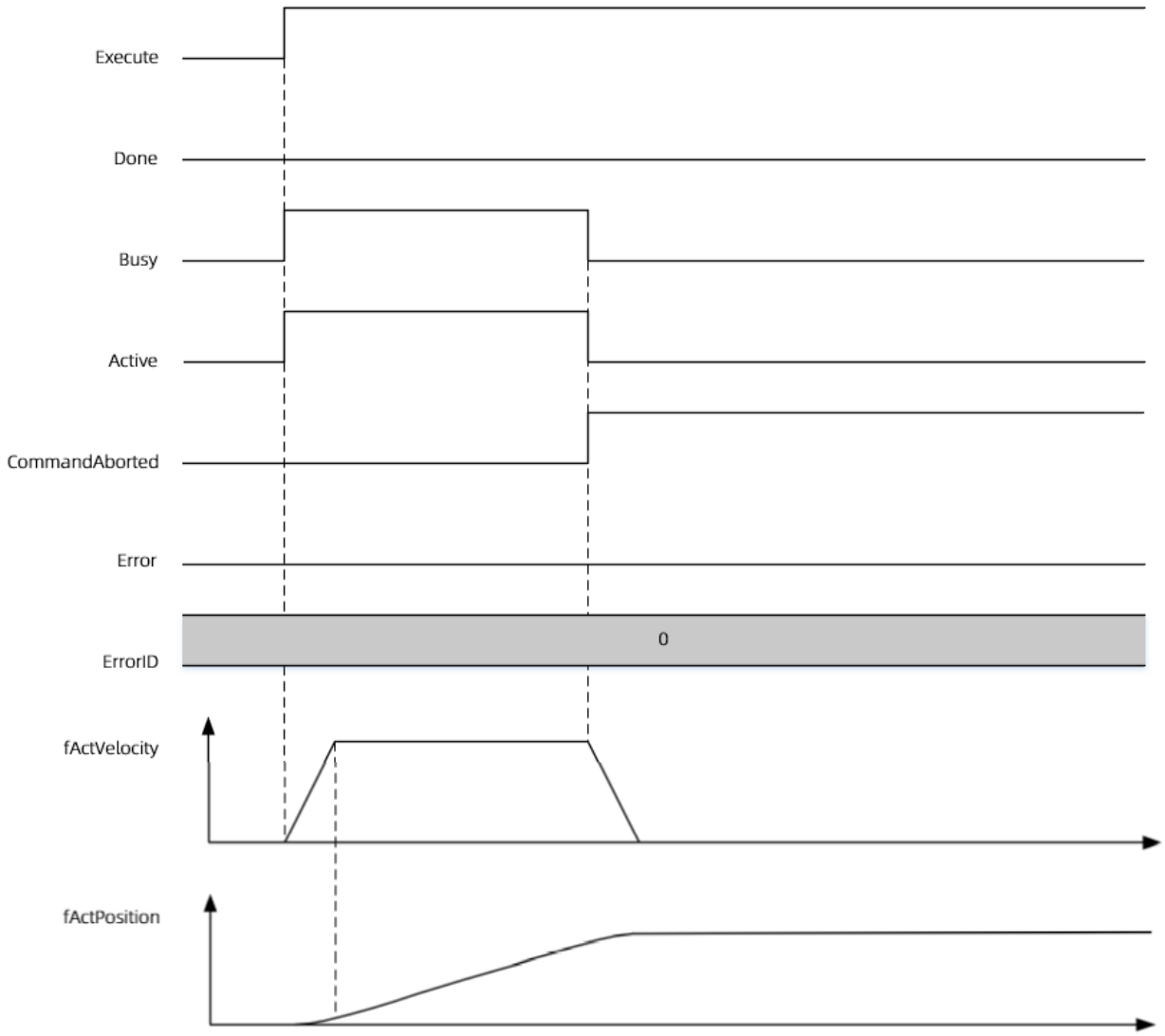
- **Sequence diagram**
  - The instruction runs normally.

    When Execute is triggered, the Busy and Active output become TRUE. When the target position is reached, the axis group velocity changes to 0, Done becomes TRUE, and Busy and Active become FALSE.

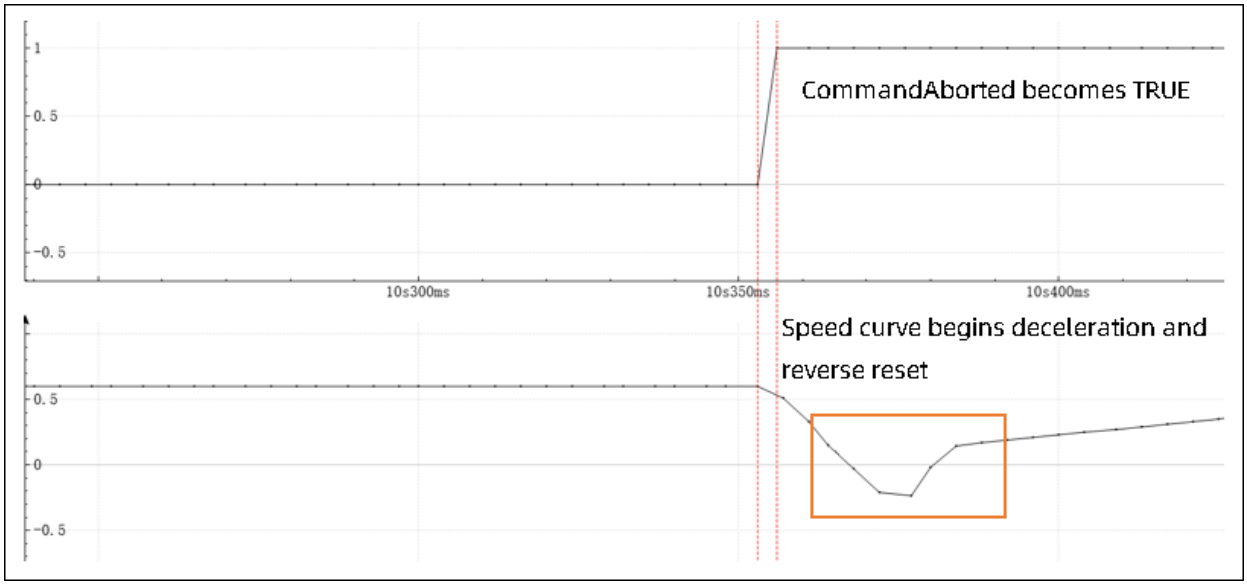


◦ The instruction is interrupted by another instruction.

When Execute is triggered, the Busy and Active output become TRUE. When this instruction is aborted by another instruction, CommandAborted becomes TRUE, and Busy and Active become FALSE.



As noted in the precautions, when a motion instruction is aborted, the point-to-point relative motion immediately replans from the interruption point. In joint space, a deceleration-reversal transition trajectory is generated along the velocity direction at that point, ensuring smooth blending with the subsequent trajectory segment.



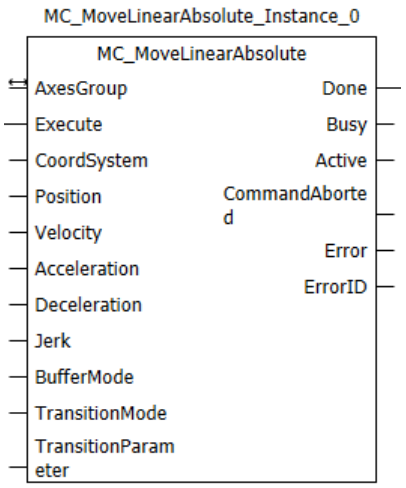
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.21 MC_MoveLinearAbsolute

This instruction enables the linear interpolation motion of the axis group from the current TCP point to the absolute position point under the given coordinate system.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_MoveLinearAbsolute	Linear absolute motion		MC_MoveLinearAbsolute_Instance_0(AxesGroup := , Execute := , CoordSystem := , Position := , Velocity := , Acceleration :=, Deceleration :=, Jerk := , BufferMode := , TransitionMode := , TransitionParameter := , Done => , Busy => , Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
CoordSystem	Coordinate system type	18.3.3.2.4.19 MC_COORDSYSTEM: Coordinate System Enumeration	ACS, PCS	PCS	Currently only supports PCS.
Position	Position	ARRAY [1..GROUP_DEF.AXIS_NUM] OF LREAL	-	0	For the absolute position coordinate for each dimension in the specified coordinate system, see Axis group type description.
Velocity	Velocity	LREAL	> 0	-	Maximum main velocity. This value may not necessarily be reached.
Acceleration	Acceleration	LREAL	> 0	-	Maximum main acceleration. This value may not necessarily be reached.
Deceleration	Deceleration	LREAL	> 0	-	Maximum main deceleration. This value may not necessarily be reached.
Jerk	Jerk	LREAL	>=0	-	Maximum main jerk. This value may not necessarily be reached.
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting 1: mcBuffered 2: mcBlendingLow 3: mcBlendingPrevious 4: mcBlendingNext 5: mcBlendingHigh	0	0: mcAborting (Aborting). 1: mcBuffered (Buffered). 2: mcBlendingLow (Blending with the lowest velocity of both FBs). 3: mcBlendingPrevious (Blending with the velocity of the first FB). 4: mcBlendingNext (Blending with velocity of the second FB). 5: mcBlendingHigh (Blending with the highest velocity of both FBs). Note: Buffer modes 2 to 5 currently have the same effect. All are blending modes. The behavior depends on the TransitionMode and TransitionParameter parameters.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
TransitionMode	Transition mode	18.3.3.2.4.10 MC_TRANSITION_MODE: Transition Mode Enumeration	0: TMNone 1: TMStartVelocity 3: TMCornerDistance 4: TMMaxCornerDeviation 10: TMDefinedVelocity 11: TMResetAbort 12: TMDecAbort	0	Definition of the transition trajectory characteristic types. When BufferMode is set to MC_ABORTING, only the transition mode parameters TMNone, TMResetAbort, and TMDecAbort are allowed. When BufferMode is set to MC_BUFFERED, the transition mode is invalid. When BufferMode is a blending mode (2 to 5), TMResetAbort or TMDecAbort is not allowed.
TransitionParameter	Transition parameter	LREAL	Velocity [0.0,1.0] Distance [0.0,+∞)	1	Transition parameters vary with the transition mode.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Instruction has controlled the axis group	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

  Position dimension

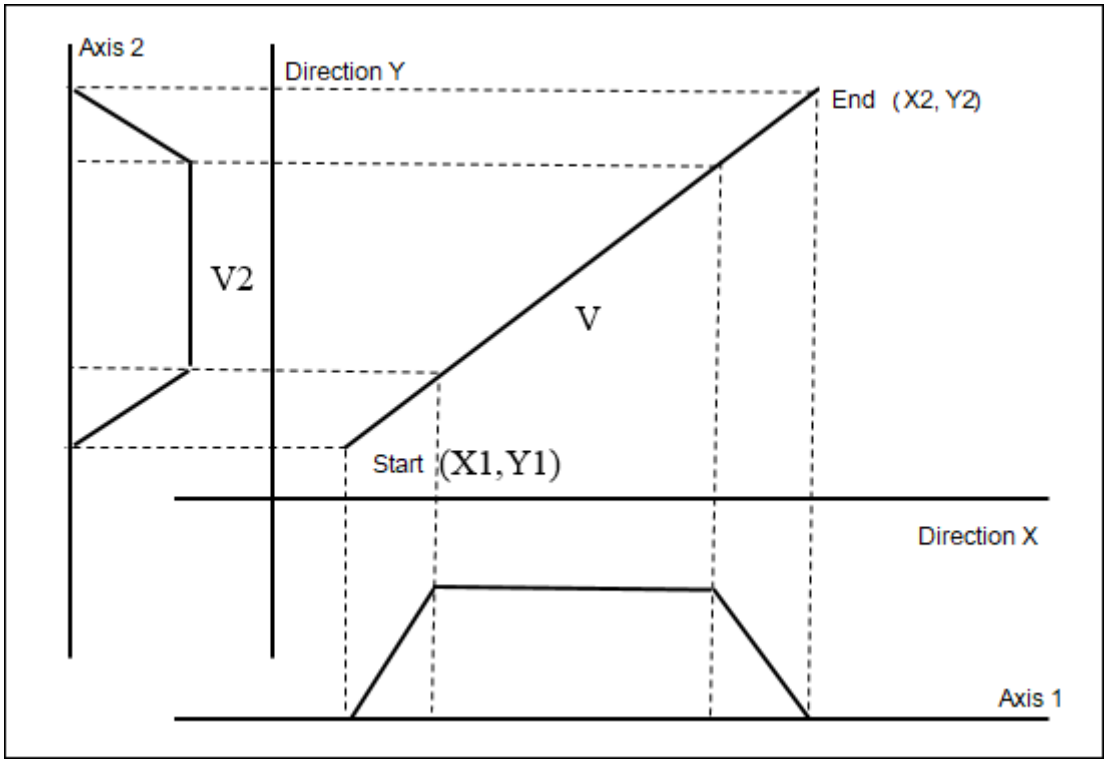
Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity	Velocity	Any velocity that exceeds the motion parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	The velocity equal to 0 will be automatically corrected to 1e-5.
Acceleration	Acceleration	Any acceleration that exceeds the motion parameter limit will be automatically corrected to the acceleration limit.	Any acceleration less than 0 will be automatically corrected to 10% of the acceleration limit.	Any acceleration equal to 0 will be automatically corrected to 10% of the acceleration limit.
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the acceleration limit.	Any deceleration less than 0 will be automatically corrected to 10% of the acceleration limit.	Any deceleration equal to 0 will be automatically corrected to 10% of the acceleration limit.
Jerk	Jerk	For the non-trapezoidal motion profiling, any jerk that exceeds the software limit will be automatically corrected to the jerk limit.	For the non-trapezoidal motion profiling, any jerk that exceeds the software limit will be automatically corrected to 10% of the jerk limit.	For the non-trapezoidal motion profiling, any jerk that exceeds the software limit will be automatically corrected to 10% of the jerk limit.

  Percentage dimension

Input Variable	Name	Correction Description for Parameter Greater than 1	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity	Velocity	Any velocity exceeding 1 is automatically corrected to 1.	Any velocity less than 0 will be automatically corrected to 1e-5.	The velocity equal to 0 will be automatically corrected to 1e-5.
Acceleration	Acceleration	Any acceleration exceeding 1 is automatically corrected to 1.	Any acceleration less than 0 will be automatically corrected to 10%.	Any acceleration equal to 0 will be automatically corrected to 10%.

Input Variable	Name	Correction Description for Parameter Greater than 1	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Deceleration	Deceleration	The value exceeding 1 is automatically corrected to 1.	Any deceleration less than 0 will be automatically corrected to 10%.	Any deceleration equal to 0 will be automatically corrected to 10%.
Jerk	Jerk	For the non-trapezoidal motion profiling, the value exceeding 1 is automatically corrected to 1.	For the non-trapezoidal motion profiling, the value exceeding 1 is automatically corrected to 10%.	For the non-trapezoidal motion profiling, the value exceeding 1 is automatically corrected to 10%.

- **Function**
  - This instruction is executed on the rising edge of Execute and performs linear absolute motion for two to four axes.
  - CoordSystem, Position, Velocity, Acceleration, Deceleration, Jerk, BufferMode, TransitionMode, and TransitionParameter can be specified in the input variables.
  - For the instruction, blending modes 2, 3, 4, and 5 for BufferMode have the same behavior. The transition effect depends on the TransitionMode and TransitionParameter settings.
  - Supports the same-cycle triggering of instructions. The Done output signal of the instruction serves as the trigger signal for Execute.
  - **CoordSystem**
    - Specify the coordinate system for linear relative motion.
  - **Position (Absolute position coordinate)**
    - For the axis type corresponding to each position vector in the Position array, see Axis group type description. Only PCS coordinates are supported.
    - In absolute motion mode, the axis group uses the set values in the Position array as the target position.
    - The ABC rotation axes of the coordinate system correspond sequentially to the ZYX-type Euler angles. If the configuration supports orientation motion, the configured ABC absolute positions must be within ±180°, otherwise an error will be generated.
    - If the target position would exceed any joint axis limit after kinematic transformation into joint space, starting this instruction will cause an error.
    - If the configuration supports orientation motion, the axis group will move to the target orientation along the shortest path.
  - **Velocity, Acceleration, Deceleration, Jerk**
    - If the velocity of any Axis variable in AxesGroup exceeds the maximum axis velocity limited by the software, it will trigger an error-stop condition.
    - The trajectory planning curve type of AxesGroup is set according to requirements. If the planning curve type is a trapezoidal curve, the Jerk variable does not take effect and is treated as 0. (The planning curve type of AxesGroup is specified in the axis group data structure.)
    - The linear motion is decomposed to each axis when linear absolute motion is performed.



V: Target velocity for the axis group to perform linear motion

V1: The speed decomposed to Axis 1:

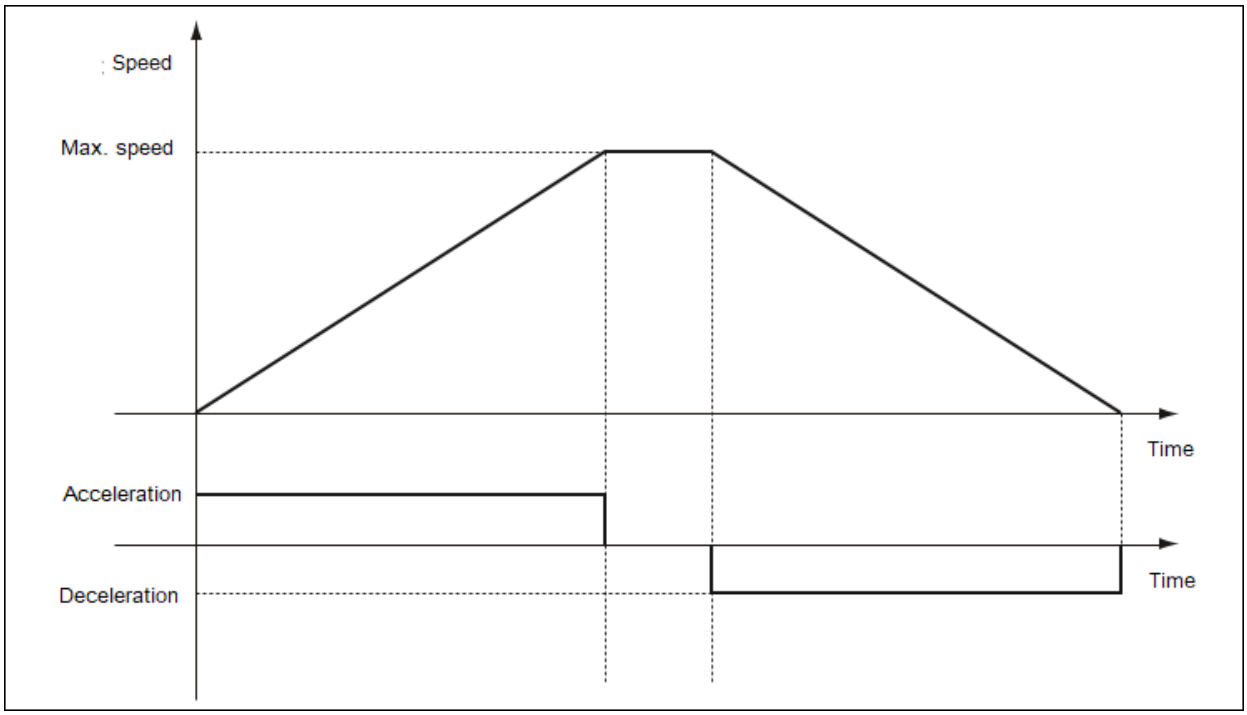
$$V1 = (X2 - X1) * V/\sqrt{(X2 - X1)^2 + (Y2 - Y1)^2}$$

V2: The speed decomposed to Axis 2:

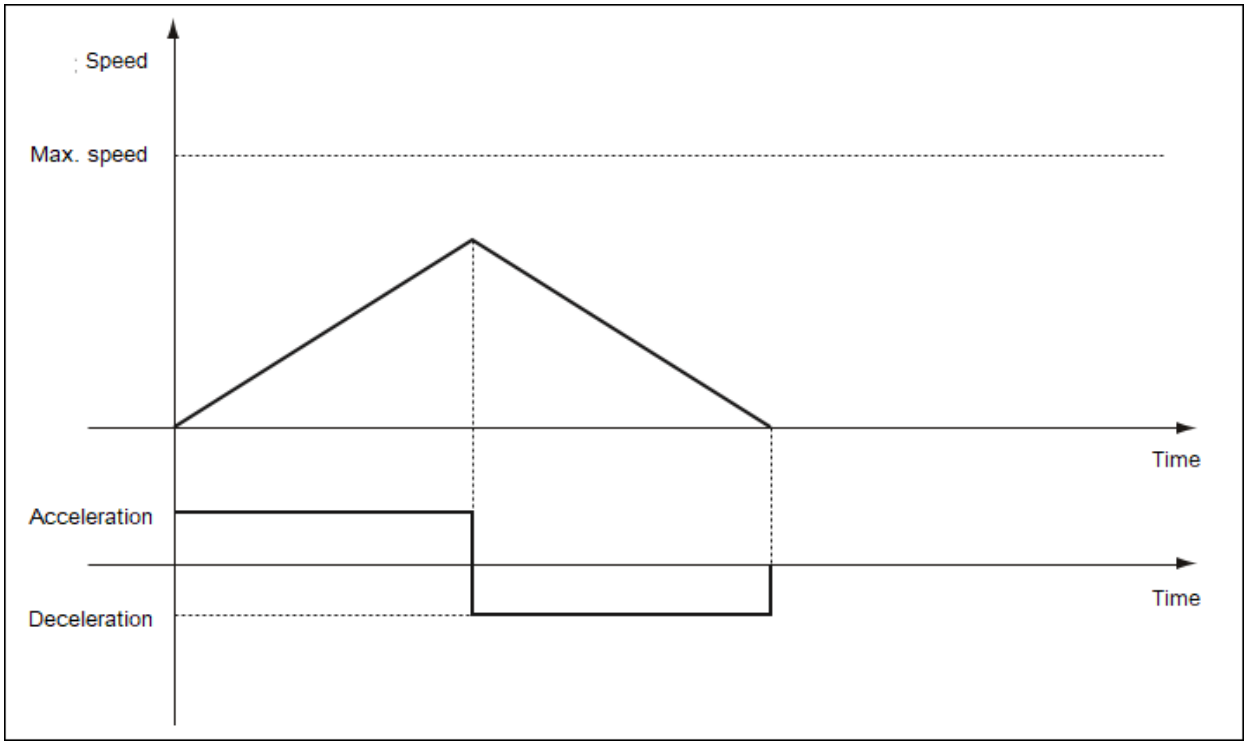
$$V2 = (Y2 - Y1) * V/\sqrt{(X2 - X1)^2 + (Y2 - Y1)^2}$$

- For Jerk = 0 or Jerk ≠ 0, the velocity, acceleration, deceleration, and jerk curves are shown in the figure below.

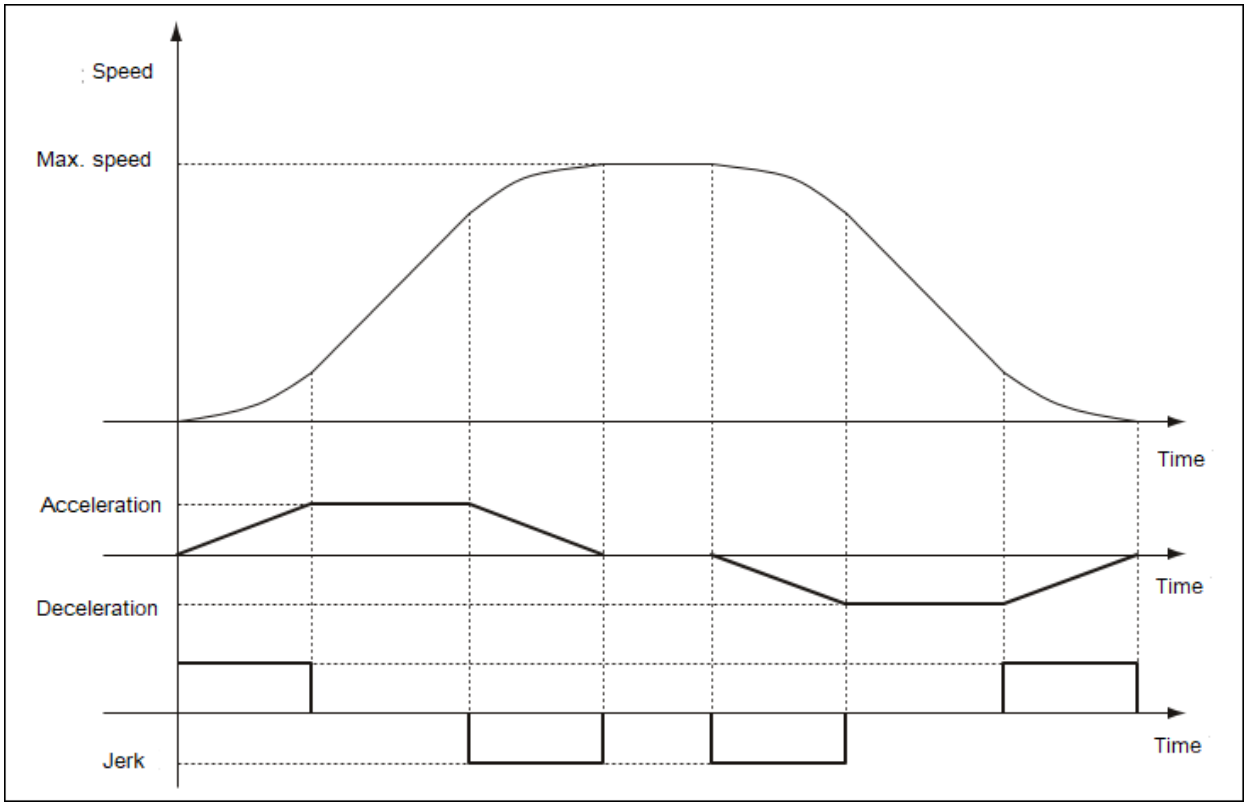
  For Jerk = 0, the axis group can reach the maximum interpolation velocity.



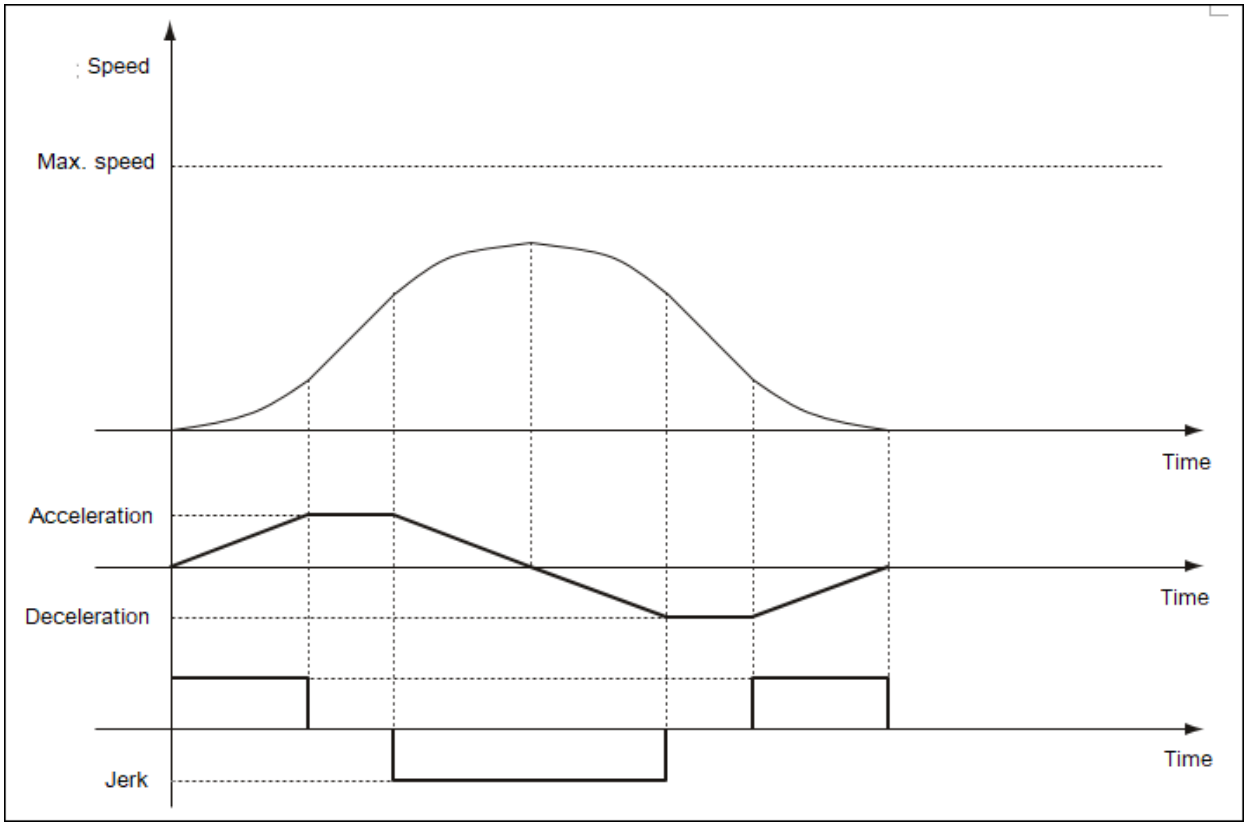
For Jerk = 0, the axis group cannot reach the maximum interpolation velocity due to too short travel distance.



For Jerk ≠ 0, the axis group can reach the maximum interpolation velocity. (Non-trapezoidal planning curve)

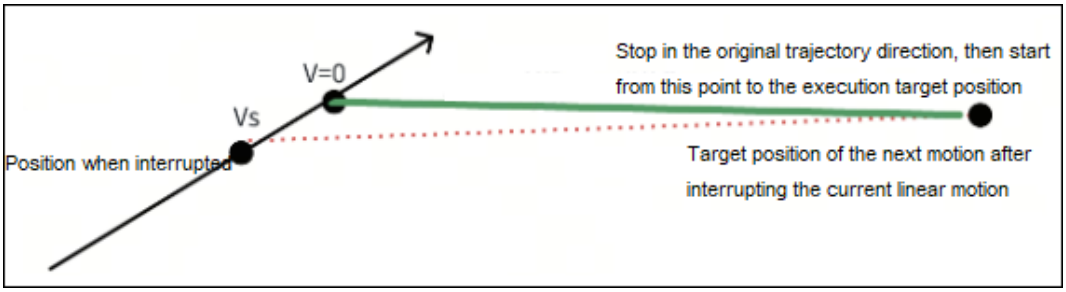


For Jerk ≠ 0, the axis group cannot reach the maximum interpolation velocity due to too short travel distance. (Non-trapezoidal planning curve)

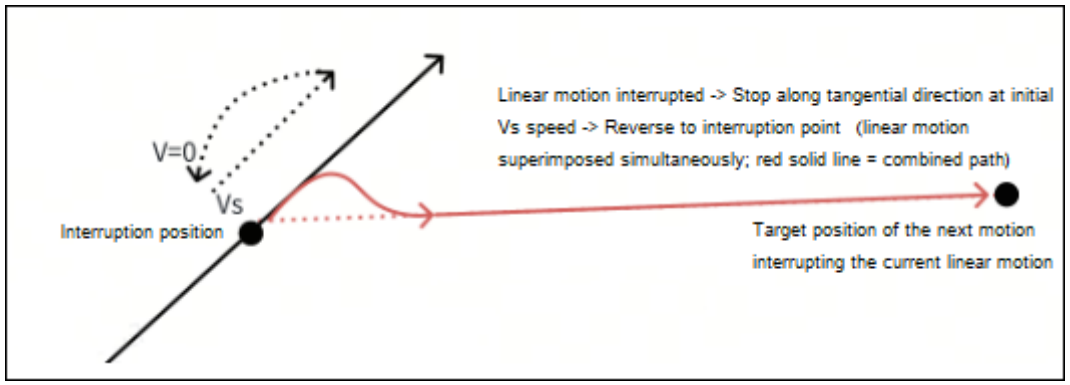


- **Re-triggering the instruction**
  - While the instruction is in motion, that is, when the instruction output parameters "Busy" and "Active" are both TRUE, this instruction cannot be re-triggered.
  - While the instruction is in buffer mode, that is, when Busy is TRUE and Active is FALSE, this instruction can be re-triggered after parameter modification. If the modified parameters are invalid, or if the buffered instruction data has already been used in motion calculations, retriggering the buffered instruction will cause an error. Then, the axis group will enter the GroupStop state, and the currently executing instruction will also report an error.
- **Multiple execution of motion instructions**
  - When this instruction is issued while another motion instruction is being executed, the new instruction can either be switched to immediately or buffered for later execution. Each axis group can buffer up to 8 instructions. In the case of multiple executions, the behavior of this instruction is determined by BufferMode.
  - BufferMode
    - Aborting: MC_ABORTING aborts the currently executing instruction and begins executing this instruction. The abort operation supports three modes, configurable via TransitionMode: TMNone, TMResetAbort, and TMDecAbort.

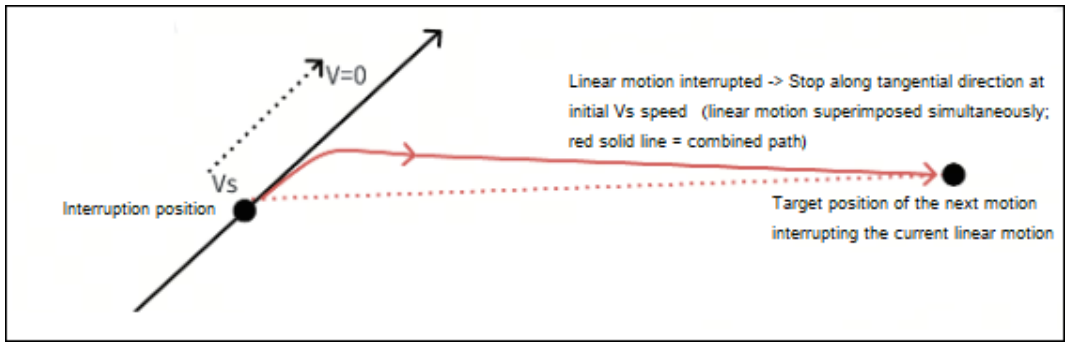
      TMNONE mode: During execution, the motion decelerates to a stop along the trajectory from the point of interruption. The next motion segment then starts from this stop point, as shown in the figure below.



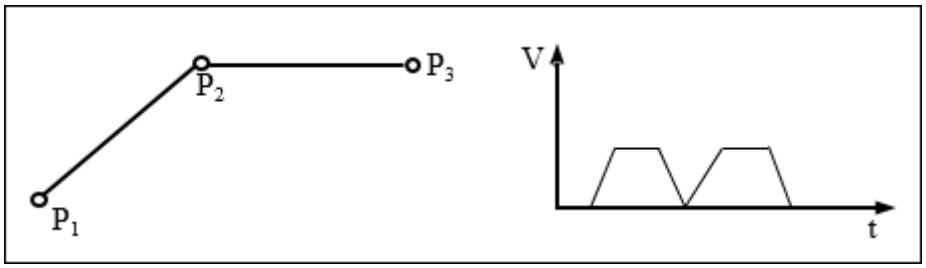
TMResetAbort mode: During execution, the axis decelerates along the straight line direction at the initial velocity of the interruption point, reverses its direction to return to that point, while simultaneously superimposing (blending) with the next motion segment, as shown in the figure below.



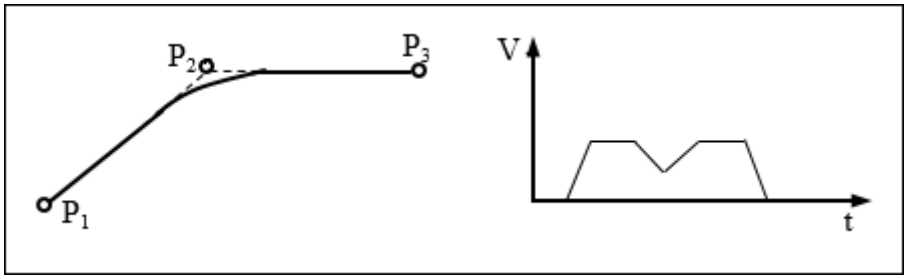
TMDecAbort mode: During execution, the axis decelerates to 0 along the straight line direction at the initial velocity of the interruption point, while simultaneously superimposing (blending) with the next motion segment, as shown in the figure below.



- Buffer: MC_BUFFERED, this instruction is buffered and executed after the current instruction is completed.



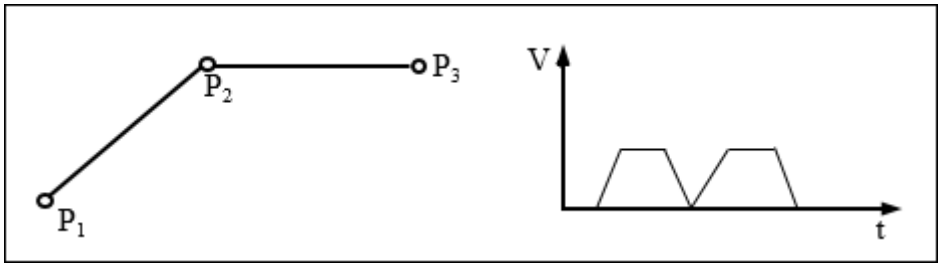
- Blending: MC_BLEND, the velocities of the two adjacent instructions are blended in the specified manner. This manner is specified in TransitionMode.



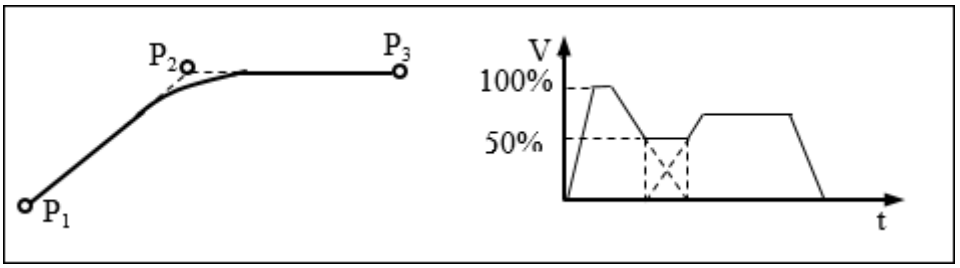
◦ TransitionMode

This variable specifies how to blend the velocities of two instructions when Blending is specified for the BufferMode.

- TMNone: Insert no transition curve. If TMNone is selected in a blending mode, the result is consistent with the transition between the velocities of the two instructions in buffered mode.



- TMStartVelocity: Transition with given start velocity. Blending transition at a certain ratio (0, 1] of the actual maximum velocity of the current motion control instruction. Effective for transitions between any motion control instructions.

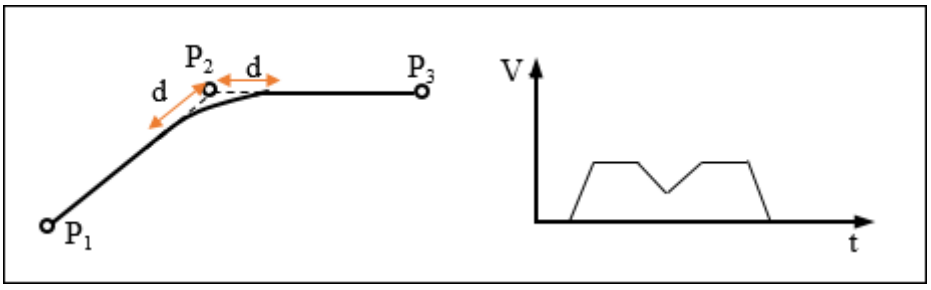


- TMCornerDistance: Corner transition mode In Cartesian space, this parameter specifies the distance before the end of the previous instruction, measured in the main dimension space. The transition trajectory is initiated at this distance to connect with the adjacent trajectory. The distance here is a generalized distance, which depends on the main dimension type setting of the axis group.

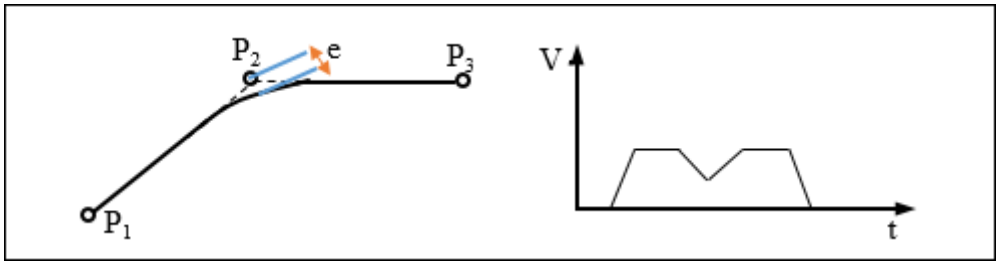
  When the primary dimension is set to position: the parameter is the displacement from the transition point to the end point.

  When the primary dimension is set to orientation: the parameter is the orientation from the transition point to the end point.

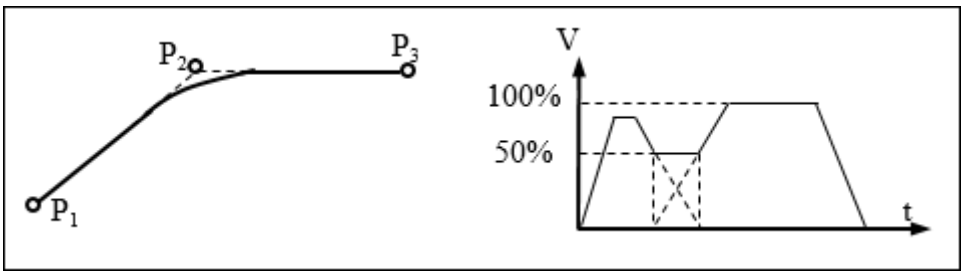
  When the primary dimension is set to ratio: The parameter is the ratio (0,1) of the resultant distance from the transition point to the endpoint to the total resultant displacement.



- TMMaxCornerDeviation: Transition mode using the maximum corner position deviation. In Cartesian space, this parameter specifies the maximum position deviation in the position subspace during the transition between adjacent instructions. An appropriate transition trajectory is constructed to connect the two adjacent paths, keeping the maximum position deviation within this limit. This is only the position deviation, independent of the main unit. Effective when both adjacent instructions are Cartesian space instructions (LIN/CIRC). Otherwise, they are processed with transition disabled.



- TMDefinedVelocity: Transition with defined velocity. Blending transition at a certain ratio (0, 1] of the actual maximum velocity of the instruction following the current motion control instruction. Effective for transitions between any motion control instructions.



- **Precautions**
  - This instruction only supports linear absolute motion in PCS coordinates.
  - The dimensions of the axis group planning parameters (velocity, acceleration, deceleration, and jerk) depend on the primary dimension set during axis group initialization. Different primary dimension types result in different dimensions and meanings for these parameters, and the parameter units are consistent with the axis group settings for the user.
    - UNIT_TRAJECT position. The input planning parameters are processed according to workspace position planning parameters.
    - UNIT_PERCENT maximum planning parameter ratio. The input planning parameters are processed as a combined ratio coefficient based on workspace pose planning parameters.
  - When the current motion instruction is interrupted, linear motion constructs a transition trajectory in working space along the direction of the velocity at the breakpoint to ensure smooth connection with the subsequent trajectory. This ensures velocity continuity only. However, the acceleration continuity is not guaranteed. For details on the three transition modes supported for aborting, see the motion abort description.
  - The transition trajectory calculation for the aborting instruction is planned according to the acceleration segment parameters set in the instruction. The higher the acceleration segment parameter value, the shorter the transition trajectory and the faster the transition.
  - **If the instruction motion mode is set to an interruption mode and the deceleration distance is insufficient, the function block reports an error. But the axis group does not enter the GroupErrorStop status; the original motion continues to execute.**
  - This instruction can only be initiated when the axis group is in the GroupMoving or GroupStandby state. Attempting to start this instruction in any other state will result in an error.
  - During the execution of this instruction, it can be interrupted by other motion instructions, MC_GroupStop, or MC_GroupHalt.

- After the execution of this instruction, the axis group (AxesGroup) state is changed from GroupMoving to GroupStandby, or remains in the GroupMoving state.
- If the current motion is in a static coordinate system, the axis group is changed from the GroupMoving state to the GroupStandby state. If the current motion is in a dynamic coordinate system, the axis group remains in the GroupMoving state.

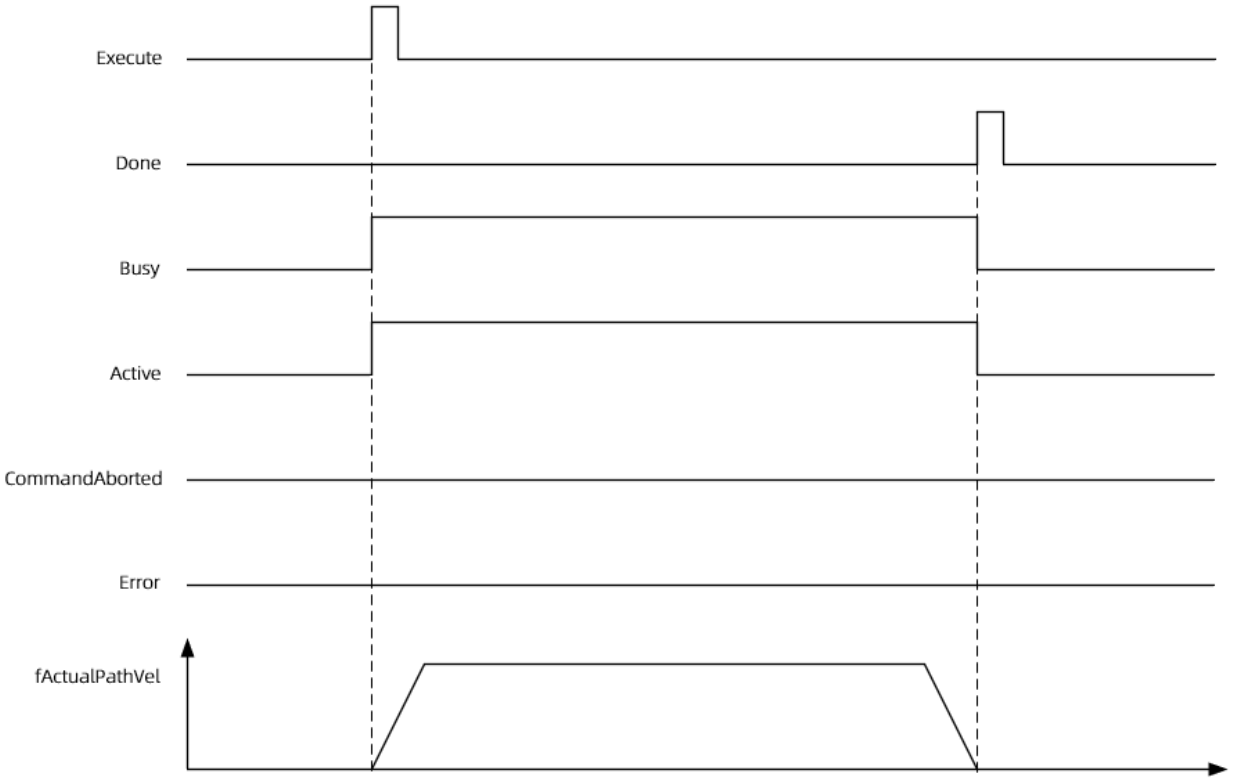
- **Program example**

  The example program for this instruction can refer to the MC_MoveDirectRelative instruction. This instruction can also be combined with other AxesGroup motion instructions.

- **Sequence diagram**

  When "Execute" is started, the output variables "Busy" and "Active" become TRUE. When the target position is reached, the axis group speed becomes 0, the output variable "Done" becomes TRUE, and the output variables "Busy" and "Active" become FALSE.

  A linear interpolation instruction is called to perform interpolation along the X-axis and Y-axis.



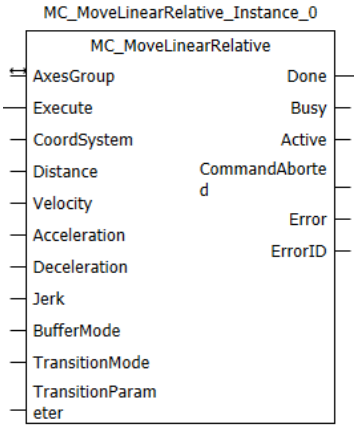
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.22 MC_MoveLinearRelative

Performs linear relative motion for an axis group, moving the axis group from the current TCP position to a relative target position via linear interpolation within the specified coordinate system.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_MoveLinearRelative	Linear relative motion		MC_MoveLinearRelative_Instance_0(AxesGroup := , Execute := , CoordSystem := , Distance := , Velocity := , Acceleration :=, Deceleration :=, Jerk := , BufferMode := , TransitionMode := , TransitionParameter := , Done => , Busy => , Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
CoordSystem	Coordinate system type	18.3.3.2.4.19 MC_COORDSYSTEM: Coordinate System Enumeration	ACS, PCS	PCS	Currently only supports PCS.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Distance	Relative Position	ARRAY [1..GROUP_DEF.AXIS_NUM] OF LREAL	-	[0,0,...]	For the displacement components in each dimension in the specified coordinate system, see Axis group type description.
Velocity	Velocity	LREAL	> 0	-	Maximum main velocity. This value may not necessarily be reached.
Acceleration	Acceleration	LREAL	> 0	-	Maximum main acceleration. This value may not necessarily be reached.
Deceleration	Deceleration	LREAL	> 0	-	Maximum main deceleration. This value may not necessarily be reached.
Jerk	Jerk	LREAL	>= 0	-	Maximum main jerk. This value may not necessarily be reached.
BufferMode	Buffer mode	18.3.3.2.2.6 MC_BUFFER_MODE: Buffer Mode Enumeration	0: mcAborting 1: mcBuffered 2: mcBlendingLow 3: mcBlendingPrevious 4: mcBlendingNext 5: mcBlendingHigh	0	0: mcAborting (Aborting). 1: mcBuffered (Buffered). 2: mcBlendingLow (Blending with the lowest velocity of both FBs). 3: mcBlendingPrevious (Blending with the velocity of the first FB). 4: mcBlendingNext (Blending with velocity of the second FB). 5: mcBlendingHigh (Blending with the highest velocity of both FBs). Note: Buffer modes 2 to 5 currently have the same effect. All are blending modes. The behavior depends on the TransitionMode and TransitionParameter parameters.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
TransitionMode	Transition mode	18.3.3.2.4.10 MC_TRANSITION_MODE: Transition Mode Enumeration	0: TMNone 1: TMStartVelocity 3: TMCornerDistance 4: TMMaxCornerDeviation 10: TMDefinedVelocity 11: TMResetAbort 12: TMDecAbort	0	Definition of the transition trajectory characteristic types. When BufferMode is set to MC_ABORTING, only the transition mode parameters TMNone, TMResetAbort, and TMDecAbort are allowed. When BufferMode is set to MC_BUFFERED, the transition mode is invalid. When BufferMode is a blending mode (2 to 5), TMResetAbort or TMDecAbort is not allowed.
TransitionParameter	Transition parameter	LREAL	Velocity [0.0,1.0] Distance [0.0,+∞)	1	Transition parameters vary with the transition mode.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Active	Instruction has controlled the axis group	BOOL	TRUE, FALSE	FALSE	TRUE when the axis is being controlled.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Parameter correction**

  Position dimension

Input Variable	Name	Correction Description for Parameter Value Greater than the Software Limit	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity	Velocity	Any velocity that exceeds the motion parameter limit will be automatically corrected to the velocity limit.	Any velocity less than 0 will be automatically corrected to 1e-5.	The velocity equal to 0 will be automatically corrected to 1e-5.
Acceleration	Acceleration	Any acceleration that exceeds the motion parameter limit will be automatically corrected to the acceleration limit.	Any acceleration less than 0 will be automatically corrected to 10% of the acceleration limit.	Any acceleration equal to 0 will be automatically corrected to 10% of the acceleration limit.
Deceleration	Deceleration	Any deceleration that exceeds the motion parameter limit will be automatically corrected to the acceleration limit.	Any deceleration less than 0 will be automatically corrected to 10% of the acceleration limit.	Any deceleration equal to 0 will be automatically corrected to 10% of the acceleration limit.
Jerk	Jerk	For the non-trapezoidal motion profiling, any jerk that exceeds the software limit will be automatically corrected to the jerk limit.	For the non-trapezoidal motion profiling, any jerk that exceeds the software limit will be automatically corrected to 10% of the jerk limit.	For the non-trapezoidal motion profiling, any jerk that exceeds the software limit will be automatically corrected to 10% of the jerk limit.

  Percentage dimension

Input Variable	Name	Correction Description for Parameter Greater than 1	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Velocity	Velocity	Any velocity exceeding 1 is automatically corrected to 1.	Any velocity less than 0 will be automatically corrected to 1e-5.	The velocity equal to 0 will be automatically corrected to 1e-5.
Acceleration	Acceleration	Any acceleration exceeding 1 is automatically corrected to 1.	Any acceleration less than 0 will be automatically corrected to 10%.	Any acceleration equal to 0 will be automatically corrected to 10%.

Input Variable	Name	Correction Description for Parameter Greater than 1	Correction Description for Parameter Value Less than 0	Correction Description for Parameter Value Equal to 0
Deceleration	Deceleration	The value exceeding 1 is automatically corrected to 1.	Any deceleration less than 0 will be automatically corrected to 10%.	Any deceleration equal to 0 will be automatically corrected to 10%.
Jerk	Jerk	For the non-trapezoidal motion profiling, the value exceeding 1 is automatically corrected to 1.	For the non-trapezoidal motion profiling, the value exceeding 1 is automatically corrected to 10%.	For the non-trapezoidal motion profiling, the value exceeding 1 is automatically corrected to 10%.

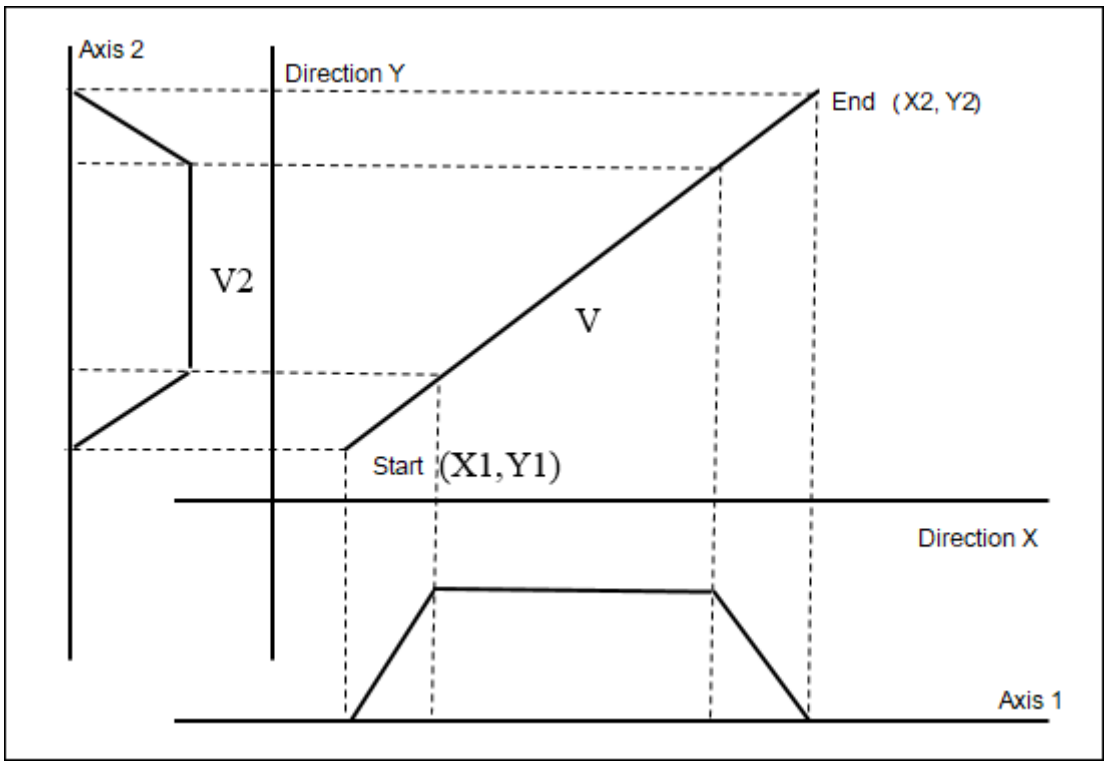
- **Function**
  - This instruction is executed on the rising edge of Execute and performs linear relative motion for two to four axes.
  - CoordSystem, Distance, Velocity, Acceleration, Deceleration, Jerk, BufferMode, TransitionMode, TransitionParameter can be specified in the input variables.
  - When this instruction is initiated, the target position is determined by adding the input variable Distance (relative position) to the starting position of AxesGroup.
  - For the instruction, blending modes 2, 3, 4, and 5 for BufferMode have the same behavior. The transition effect depends on the TransitionMode and TransitionParameter settings.
  - Supports the same-cycle triggering of instructions. The Done output signal of the instruction serves as the trigger signal for Execute.
  - **CoordSystem**
    - Specify the coordinate system for linear relative motion. Currently, only linear relative motion under the PCS is supported.
  - **Distance (Relative position coordinate)**
    - For the axis type corresponding to each position vector in the Distance array, see Axis group type description. Only PCS coordinates are supported.
    - The ABC rotation axes of the coordinate system correspond sequentially to the ZYX-type Euler angles. If the configuration supports orientation motion, the configured ABC relative positions must be within ±180°, otherwise an error will be generated.
    - In relative motion mode, the rules for calculating the target position of the axis group are shown in the table below.

Coordinate system type (calculation of target positions for each axis in the existing axis group configuration)
PCS
X-axis target position = X-axis current position + Distance [1]
Y-axis target position = Y-axis current position + Distance [2]
Z-axis target position = Z-axis current position + Distance [3]
A-axis target position = A-axis current position + Distance [4]

- If the A-axis target position is not within ±180°, the system will make the following adjustments: if the A-axis target position is greater than 180°, the system will subtract 360° until the result falls within ±180°; if the A-axis target position is less than -180°, the system will add 360° until the result falls within ±180°. Example: $A_{\text{calculated target position}}$ = 260°, $A_{\text{converted target position}}$ = -100°.
- If the target position would exceed any joint axis limit after kinematic transformation into joint space, starting this instruction will cause an error.
- If the configuration supports orientation motion, the axis group will move to the target orientation along the shortest path.

◦ **Velocity, Acceleration, Deceleration, Jerk**

- If the velocity of any Axis variable in AxesGroup exceeds the maximum axis velocity limited by the software, it will trigger an error-stop condition.
- The trajectory planning curve type of AxesGroup is set according to requirements. If the planning curve type is a trapezoidal curve, the Jerk variable does not take effect and is treated as 0. (The planning curve type of AxesGroup is specified in the axis group data structure.)
- The linear motion is decomposed to each axis when linear relative motion is performed.



V: Target velocity for the axis group to perform linear motion

V1: The speed decomposed to Axis 1:

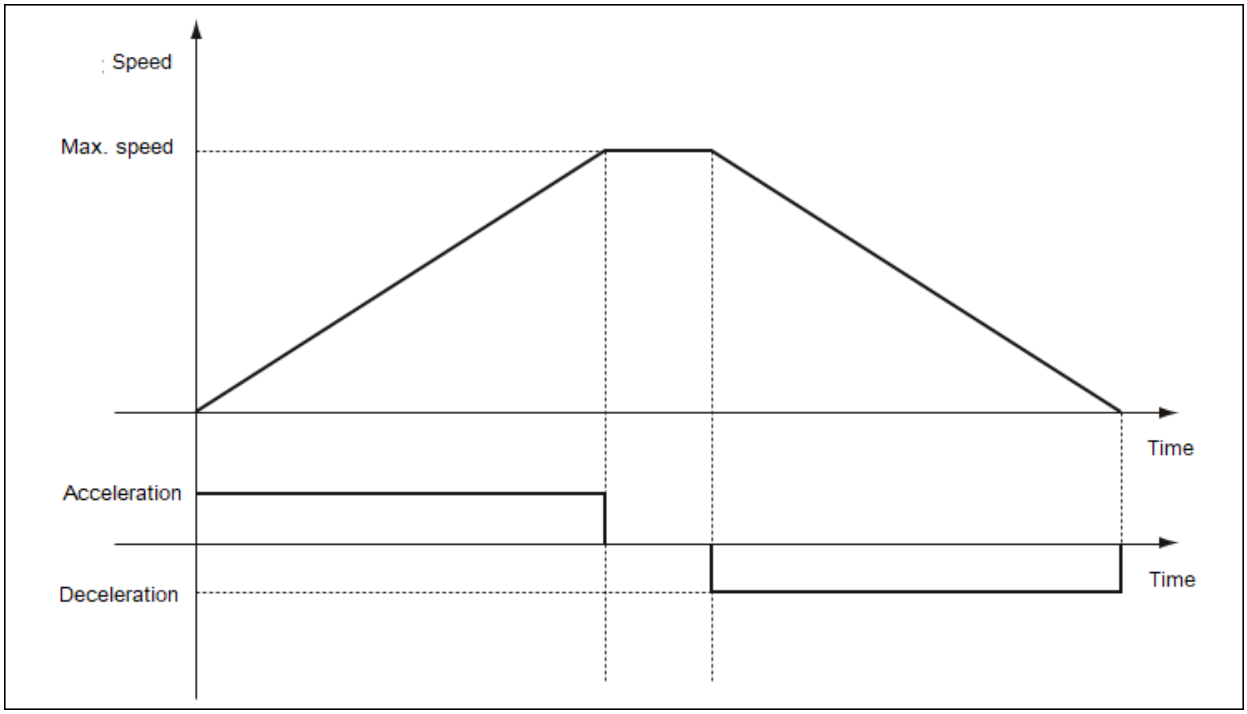
$$V1 = (X2 - X1) * V/\sqrt{(X2 - X1)^2 + (Y2 - Y1)^2}$$

V2: The speed decomposed to Axis 2:

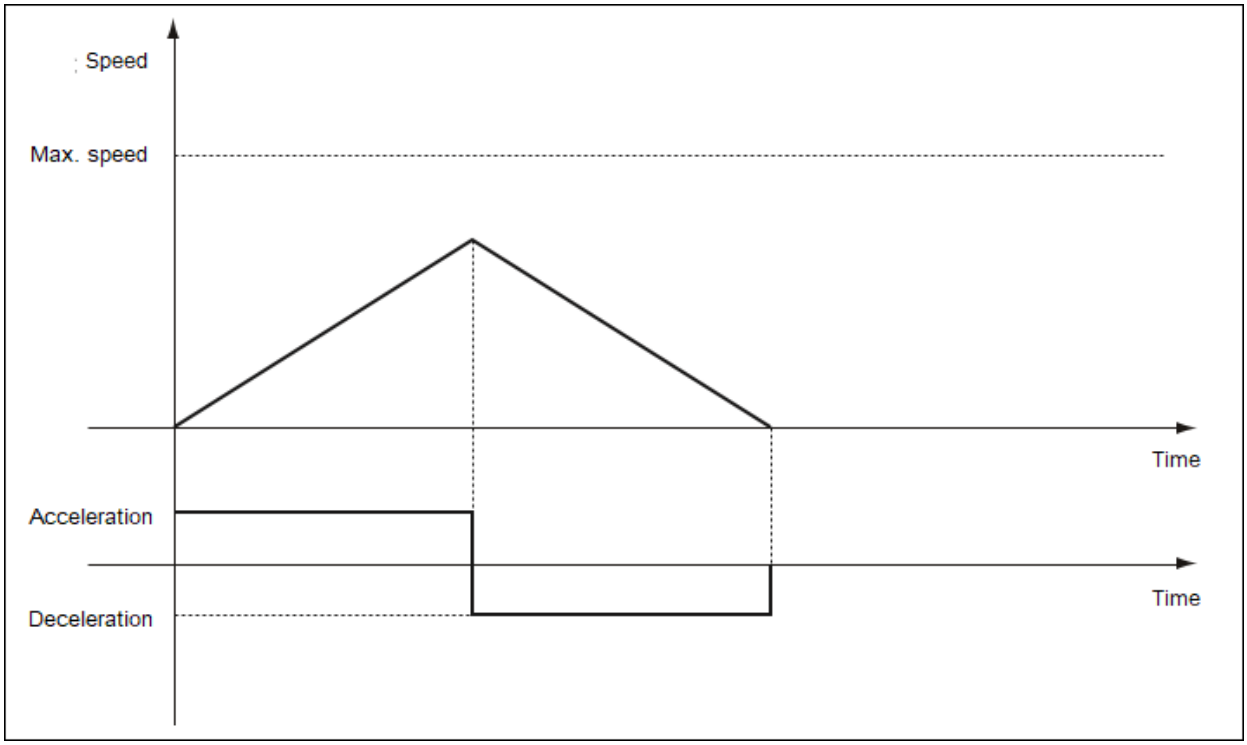
$$V2 = (Y2 - Y1) * V/\sqrt{(X2 - X1)^2 + (Y2 - Y1)^2}$$

- For Jerk = 0 or Jerk ≠ 0, the velocity, acceleration, deceleration, and jerk curves are shown in the figure below.

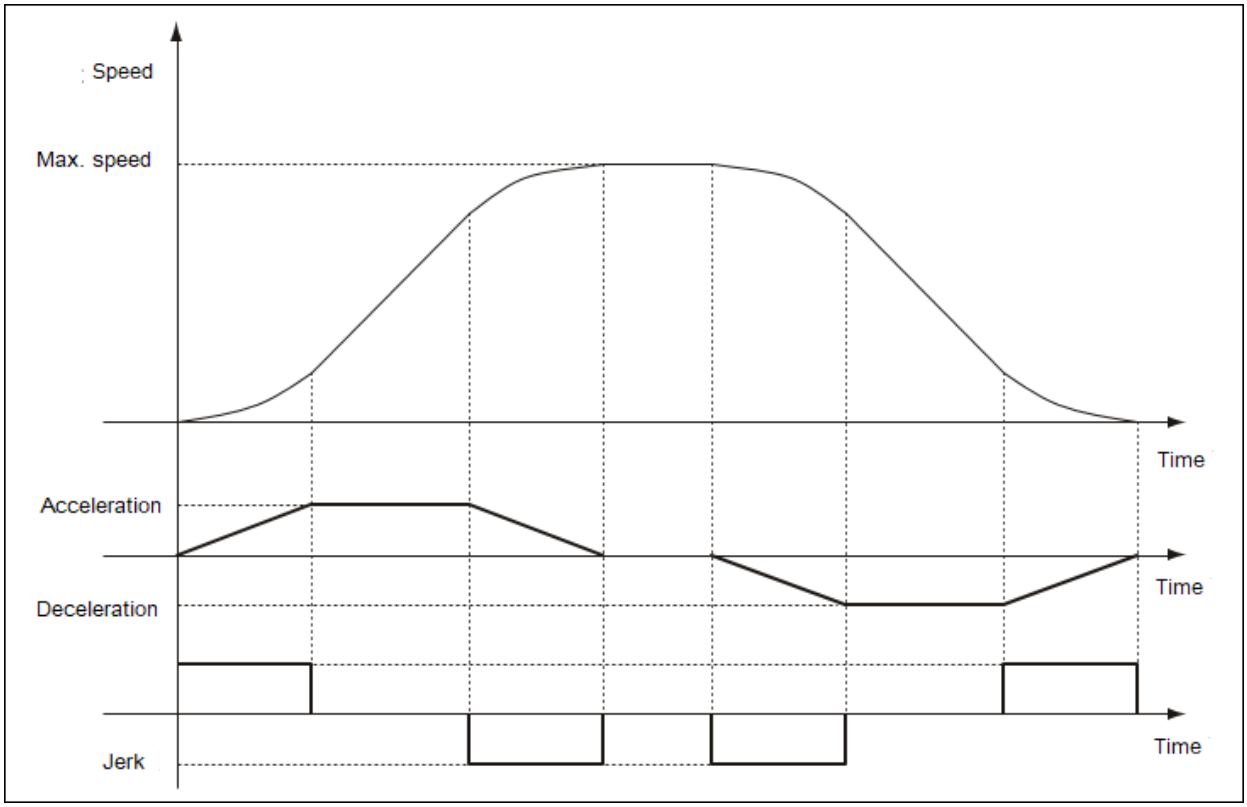
For Jerk = 0, the axis group can reach the maximum interpolation velocity.



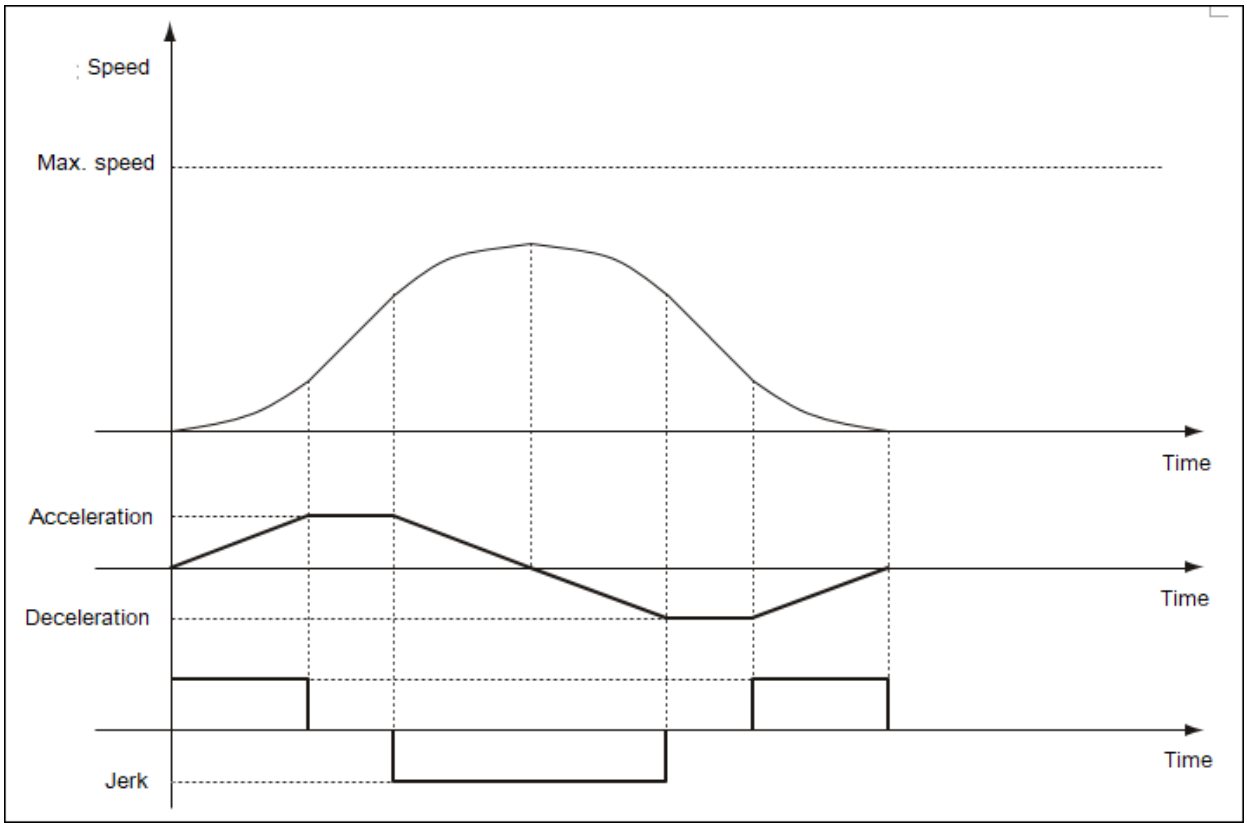
For Jerk = 0, the axis group cannot reach the maximum interpolation velocity due to too short travel distance.



For Jerk ≠ 0, the axis group can reach the maximum interpolation velocity. (Non-trapezoidal planning curve)



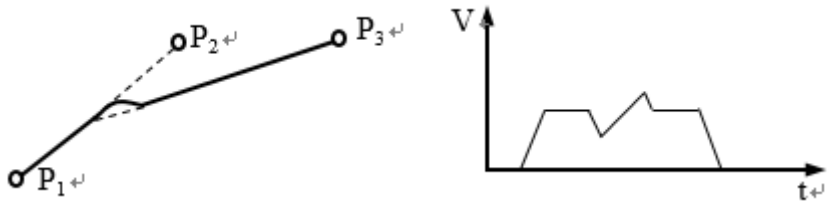
For Jerk ≠ 0, the axis group cannot reach the maximum interpolation velocity due to too short travel distance. (Non-trapezoidal planning curve)



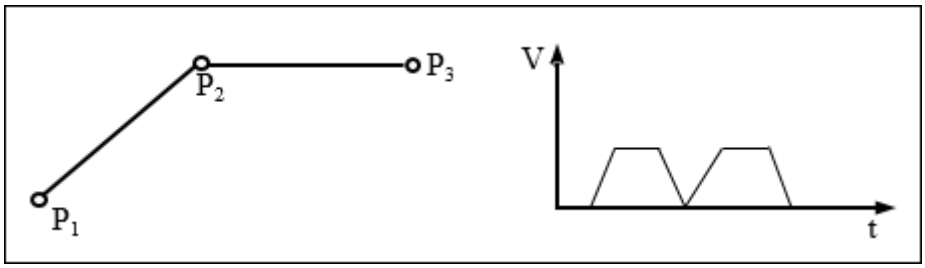
- **Re-triggering the instruction**
  - While the instruction is in motion, that is, when the instruction output parameters "Busy" and "Active" are both TRUE, this instruction cannot be re-triggered.
  - While the instruction is in buffer mode, that is, when Busy is TRUE and Active is FALSE, this instruction can be re-triggered after parameter modification. If the modified parameters are invalid, or if the buffered instruction data has already been used in motion calculations, retriggering the buffered

instruction will cause an error. Then, the axis group will enter the GroupStop state, and the currently executing instruction will also report an error.

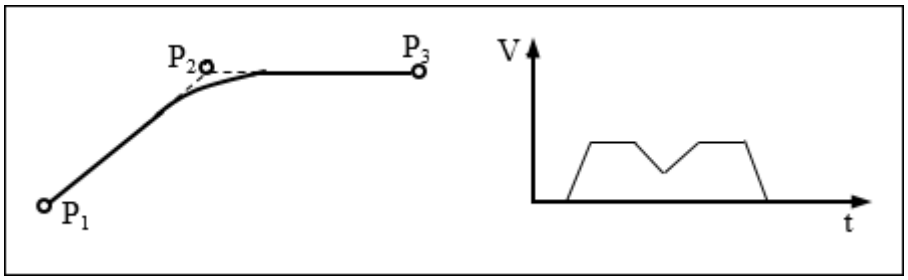
- **Multiple execution of motion instructions**
  - When this instruction is issued while another motion instruction is being executed, the new instruction can either be switched to immediately or buffered for later execution. Each axis group can buffer up to 8 instructions. In the case of multiple executions, the behavior of this instruction is determined by BufferMode.
  - BufferMode
    - Aborting: MC_ABORTING aborts the currently executing instruction and begins executing this instruction.



    - Buffer: MC_BUFFERED, this instruction is buffered and executed after the current instruction is completed.



    - Blending: MC_BLEND, the velocities of the two adjacent instructions are blended in the specified manner. This manner is specified in TransitionMode.

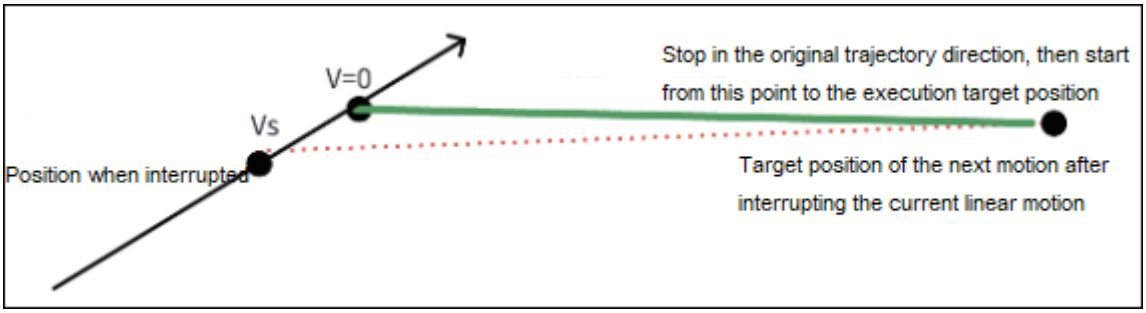


  - TransitionMode

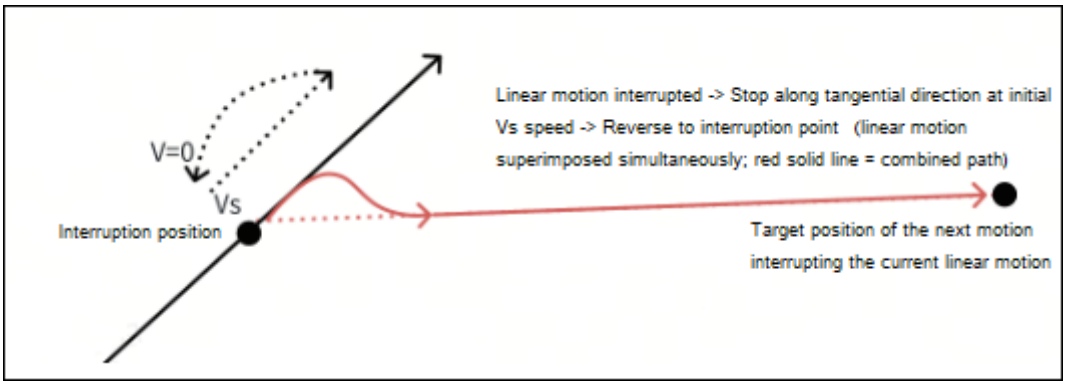
    This variable specifies how to blend the velocities of two instructions when Blending is specified for the BufferMode.

    - TMNone: Insert no transition curve. If TMNone is selected in a blending mode, the result is consistent with the transition between the velocities of the two instructions in buffered mode. The abort operation supports three modes, configurable via TransitionMode: TMNone, TMResetAbort, and TMDecAbort.

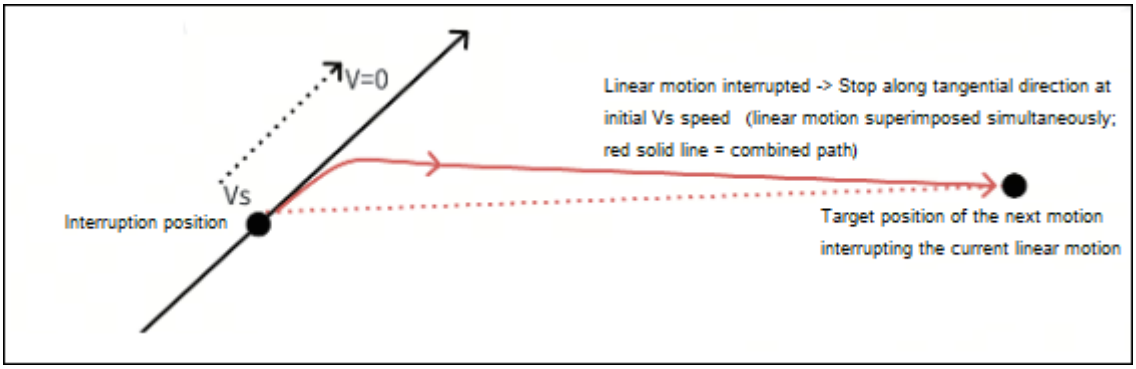
      TMNONE mode: During execution, the motion decelerates to a stop along the trajectory from the point of interruption. The next motion segment then starts from this stop point, as shown in the figure below.



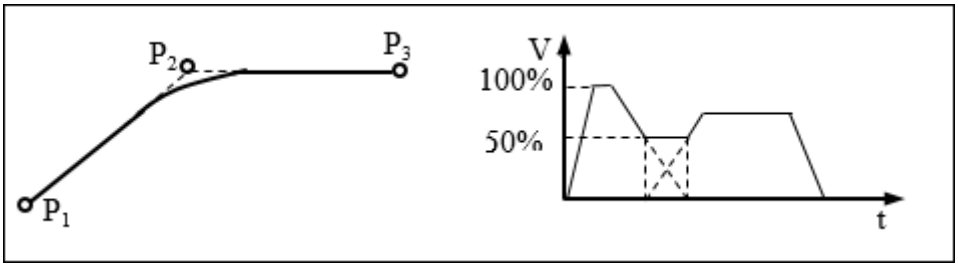
TMResetAbort mode: During execution, the axis decelerates along the straight line direction at the initial velocity of the interruption point, reverses its direction to return to that point, while simultaneously superimposing (blending) with the next motion segment, as shown in the figure below.



TMDecAbort mode: During execution, the axis decelerates to 0 along the straight line direction at the initial velocity of the interruption point, while simultaneously superimposing (blending) with the next motion segment, as shown in the figure below.



- TMStartVelocity: Transition with given start velocity. Blending transition at a certain ratio (0, 1] of the actual maximum velocity of the current motion control instruction. Effective for transitions between any motion control instructions.

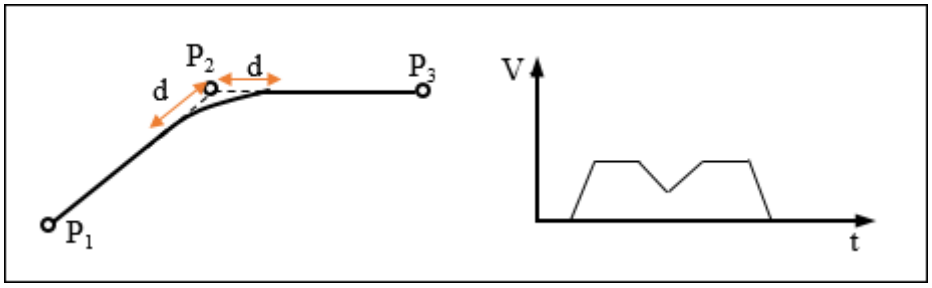


- TMCornerDistance: Corner transition mode In Cartesian space, this parameter specifies the distance before the end of the previous instruction, measured in the main dimension space. The transition trajectory is initiated at this distance to connect with the adjacent trajectory. The distance here is a generalized distance, which depends on the main dimension type setting of the axis group.

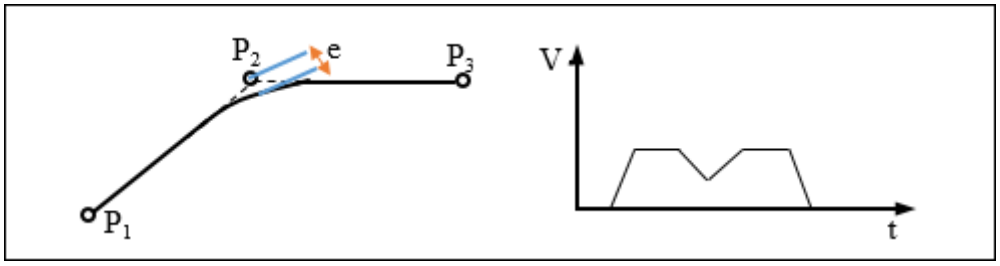
When the primary dimension is set to position: the parameter is the displacement from the transition point to the end point.

When the primary dimension is set to orientation: the parameter is the orientation from the transition point to the end point.

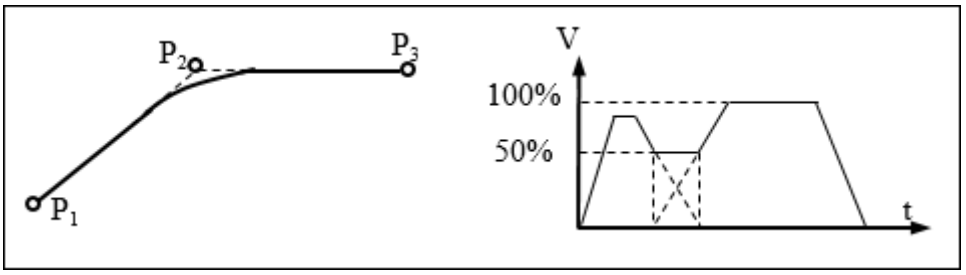
When the primary dimension is set to ratio: The parameter is the ratio (0 to 1) of the resultant distance from the transition point to the endpoint to the total resultant displacement.



- TMMaxCornerDeviation: Transition mode using the maximum corner position deviation. In Cartesian space, this parameter specifies the maximum position deviation in the position subspace during the transition between adjacent instructions. An appropriate transition trajectory is constructed to connect the two adjacent paths, keeping the maximum position deviation within this limit. This is only the position deviation, independent of the main unit. Effective when both adjacent instructions are Cartesian space instructions (LIN/CIRC). Otherwise, they are processed with transition disabled.



- TMDefinedVelocity: Transition with defined velocity. Blending transition at a certain ratio (0, 1] of the actual maximum velocity of the instruction following the current motion control instruction. Effective for transitions between any motion control instructions.



- **Precautions**
  - This instruction only supports linear absolute motion in PCS coordinates.
  - The dimensions of the axis group planning parameters (velocity, acceleration, deceleration, and jerk) depend on the primary dimension set during axis group initialization. Different primary dimension types result in different dimensions and meanings for these parameters, and the parameter units are consistent with the axis group settings for the user.
    - UNIT_TRAJECT position. The input planning parameters are processed according to workspace position planning parameters.
    - UNIT_PERCENT maximum planning parameter ratio. The input planning parameters are processed as a combined ratio coefficient based on workspace pose planning parameters.

- When the current motion instruction is interrupted, linear motion constructs a transition trajectory in working space along the direction of the velocity at the breakpoint to ensure smooth connection with the subsequent trajectory. This ensures velocity continuity only. However, the acceleration continuity is not guaranteed. For details on the three transition modes supported for aborting, see the motion abort description.
- The transition trajectory calculation for the aborting instruction is planned according to the acceleration segment parameters set in the instruction. The higher the acceleration segment parameter value, the shorter the transition trajectory and the faster the transition.
- **If the instruction motion mode is set to an interruption mode and the deceleration distance is insufficient, the function block reports an error. But the axis group does not enter the GroupErrorStop status; the original motion continues to execute.**
- This instruction can only be initiated when the axis group is in the GroupMoving or GroupStandby state. Attempting to start this instruction in any other state will result in an error.
- During the execution of this instruction, it can be interrupted by other motion instructions, MC_GroupStop, or MC_GroupHalt.
- After the execution of this instruction, the axis group (AxesGroup) state is changed from GroupMoving to GroupStandby, or remains in the GroupMoving state.
- If the current motion is in a static coordinate system, the axis group is changed from the GroupMoving state to the GroupStandby state. If the current motion is in a dynamic coordinate system, the axis group remains in the GroupMoving state.

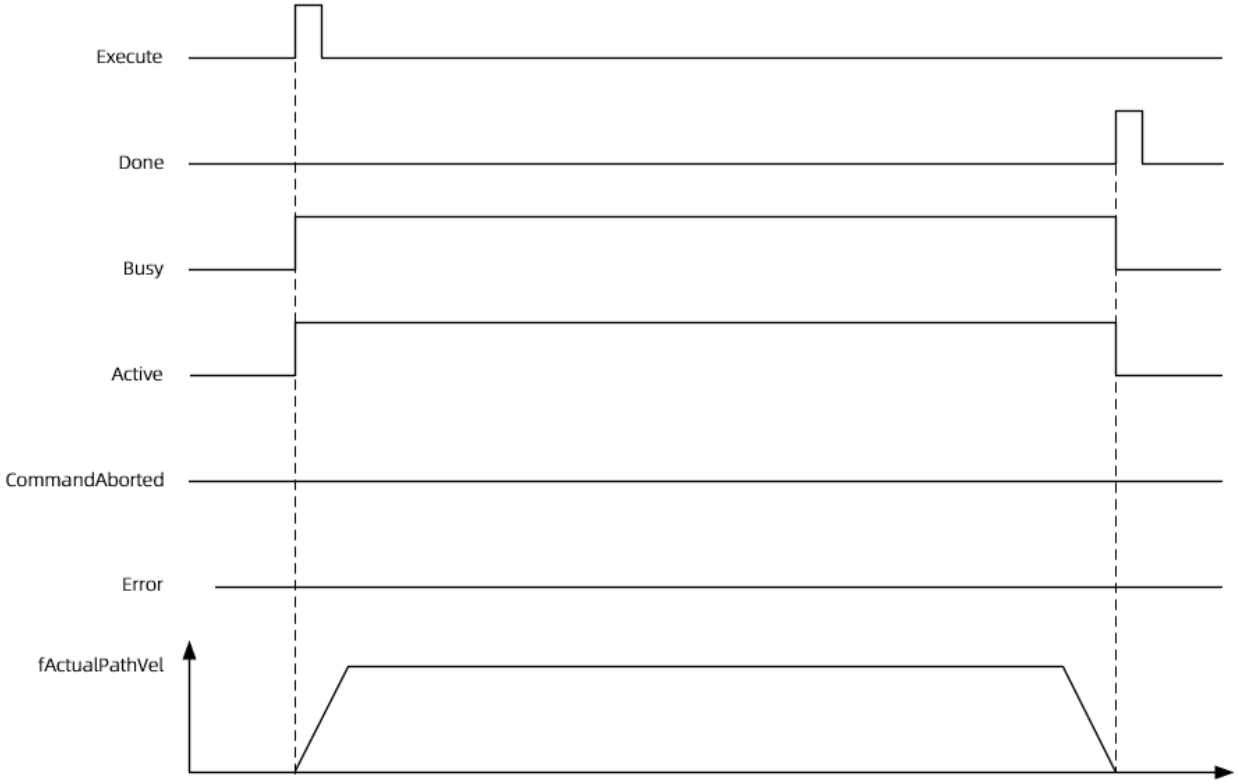
- **Program example**

  The example program for this instruction can refer to the MC_MoveDirectRelative instruction. This instruction can also be combined with other AxesGroup motion instructions.

- **Sequence diagram**

  When "Execute" is started, the output variables "Busy" and "Active" become TRUE. When the target position is reached, the axis group speed becomes 0, the output variable "Done" becomes TRUE, and the output variables "Busy" and "Active" become FALSE.

  A linear interpolation instruction is called to perform interpolation along the X-axis and Y-axis.



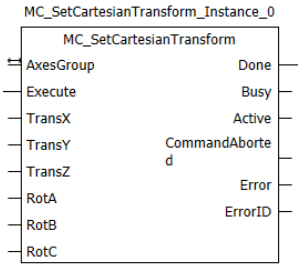
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.23 MC_SetCartesianTransform

This instruction is used to set the basic static part coordinate system (PART_BASE_MCS), which defines the transformation between the machine coordinate system (MCS) and the part coordinate system (PCS).

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_SetCartesianTransform	Set transformation between part coordinate system and machine coordinate system		MC_SetCartesianTransform_Instance_0(AxesGroup := , Execute := , TransX := , TransY := , TransZ := , RotA := , RotB := , RotC := , Done => , Busy => , Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.
TransX	X-axis	LREAL	-	0	X-axis component in the Cartesian transformation coordinate system.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
TransY	Y-axis	LREAL	-	0	Y-axis component in the Cartesian transformation coordinate system.
TransZ	Z-axis	LREAL	-	0	Cartesian transformation coordinate system Z-axis component.
RotA	A-axis	LREAL	-	0	A-axis component in the Cartesian transformation coordinate system.
RotB	B-axis	LREAL	-	0	B-axis component in the Cartesian transformation coordinate system.
RotC	C-axis	LREAL	-	0	C-axis component in the Cartesian transformation coordinate system.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Static coordinate switch response.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Active	Instruction has controlled the axis group	BOOL	TRUE, FALSE	FALSE	Static coordinate switching in progress.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**
  - This instruction is executed on the rising edge of Execute, establishing the standard Cartesian part coordinate system for transformation between the machine coordinate system (MCS) and the part coordinate system (PCS).
  - The default initial value for each component of the Cartesian coordinate system is 0.0.
  - The order ABC of the rotation axes corresponds to ZYX Euler angles.
  - The part coordinate system can be defined multiple times within the same AxesGroup at the same time. The function block can be reset after successful static coordinate system setting, interruption, or error.
  - The part coordinate system can be set only under either of two conditions: the AxesGroup is in GroupStandby state, or in GroupMoving state with its motion idle (Idle = TRUE); the part coordinate system of the axis group is in the transformation process. All other axis group states will trigger an error. In addition, it is prohibited to set the part coordinate system when the mounting axis in the axis group is in the PowerOff state. Otherwise, an error will be reported.
  - When a new static part coordinate system is set: The function block responds with "Busy" and updates the axis group position, velocity, and acceleration data in the new coordinate system (ACS/PCS) within one cycle (Active). If the part coordinate system was in static mode before the switch, the function block completes the switch (Done). If the part coordinate system was in dynamic mode before the switch, the axis group starts decelerating from its current dynamic tracking state and performs zero-displacement planning (Active). Then, it decelerates to a stop, enters a stationary state, and completes the static coordinate system switch (Done).
  - Do not set the part coordinate system during continuous trajectory operation of the axis group; otherwise, the function block will report an error, but the current state of the axis group remains unchanged.
  - The set values of the input variables RotA, RotB, and RotC must be within ±180 degrees. Otherwise, the instruction will error.

- Due to mechanical configuration constraints, only the TransX and TransY coordinate axis values can be set for XY and Delta2 configurations. If a value is set for any other coordinate axis, the instruction will report an error.
- For XYZA, SCARA, and Delta3+1 configurations, only the A-axis orientation is available. Therefore, the spatial tilt orientations defined by RotB and RotC in the user coordinate system cannot be achieved. The actual orientation executed by the equipment is the orthogonal projection of the orientation defined in the specified conveyor belt coordinate system onto the XY plane of the base coordinate system.

- **Precautions**

  Do not re-trigger the function block during the switching process. Otherwise, the function block reports an error.

- **Sequence diagram**

  The Execute input of the function block must be triggered on a rising edge. The Done output of the function block indicates the instruction is executed successfully. The Busy output of the function block indicates that the current function block is being executed. The CommandAborted output of the function block indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

  On the rising edge of Execute, both Busy and Active outputs are set to TRUE.

  After the transformation between the machine coordinate system (MCS) and the part coordinate system (PCS) is completed, the output variable "Done" becomes TRUE.



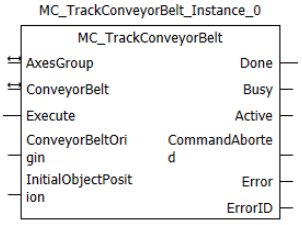
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.24 MC_TrackConveyorBelt

This instruction enables the conveyor belt tracking by providing an abstraction layer for the conveyor belt to assist in tracking objects moving in a straight line in space. This instruction activates a dynamic coordinate transformation from the base frame to the selected axis group frame, thereby enabling dynamic tracking of a synchronous belt.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_TrackConveyorBelt	Conveyor belt tracking		MC_TrackConveyorBelt_Instance_0(AxesGroup := , ConveyorBelt := , Execute := , ConveyorBeltOrigin := , InitialObjectPosition := , Done => , Busy => , Active =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.
ConveyorBelt	Conveyor belt	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
ConveyorBeltOrigin	Conveyor belt initial pose	S_CARTESIAN	-	[0,0,…]	Pose of the conveyor belt relative to the base coordinate system. The X-axis of the conveyor belt coordinate system should point in the direction of conveyor belt movement.
InitialObjectPosition	Initial object pose	S_CARTESIAN	-	[0,0,…]	Pose of the object relative to the conveyor belt coordinate system at the initial moment.

Output variable

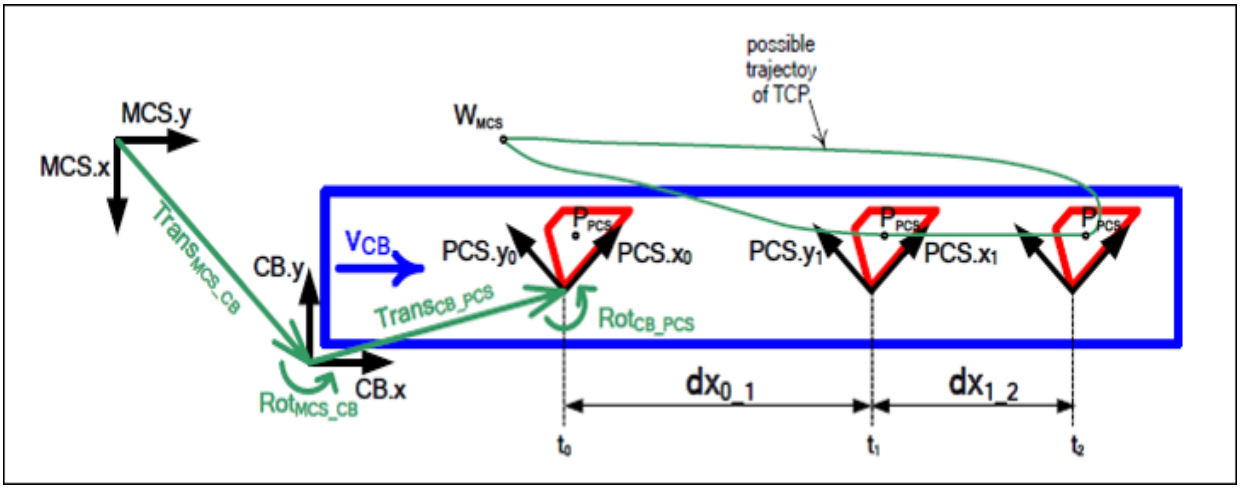
Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the conveyor belt tracking is complete.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the coordinate following task has responded.
Active	Instruction has controlled the axis group	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the coordinate following is in progress.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the selected coordinate system transformation changes (The new part coordinate system is set).
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**
  - This instruction is executed on the rising edge of Execute.
  - The dynamic tracking of a synchronous belt is one of the dynamic part coordinate systems. By associating the current axis group with a known belt and its designated dynamic coordinate system attached to and moves with the belt, the axis group achieves synchronous tracking of the belt.
  - The orientation values (A, B, C) set for the input variables ConveyorBeltOrigin and InitialObjectPosition must be within ±180 degrees. Otherwise, the instruction will report an error.
  - Due to mechanical configuration constraints, for XY and Delta2 configurations, only the X- and Y-axis values can be set for the input variables ConveyorBeltOrigin and InitialObjectPosition. If a value is set for any other coordinate axis, the instruction will report an error.
  - For XYZA, SCARA, and Delta3+1 configurations, only the A-axis orientation is available. Therefore, the spatial tilt orientations defined by the conveyor coordinate systems B and C cannot be achieved. The actual orientation executed by the equipment is the orthogonal projection of the orientation defined in the specified conveyor belt coordinate system onto the XY plane of the base coordinate system.
- **Re-triggering the instruction**

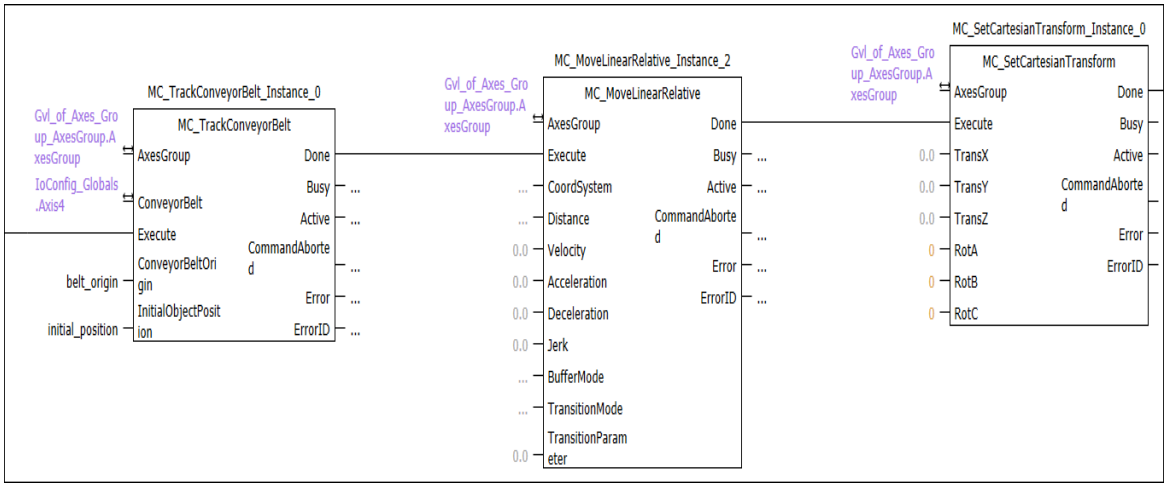
  This instruction cannot be re-triggered; otherwise the function block will be rejected with an error. The axis group enters the GroupErrorStop status.
- **Example of the tracking process:**

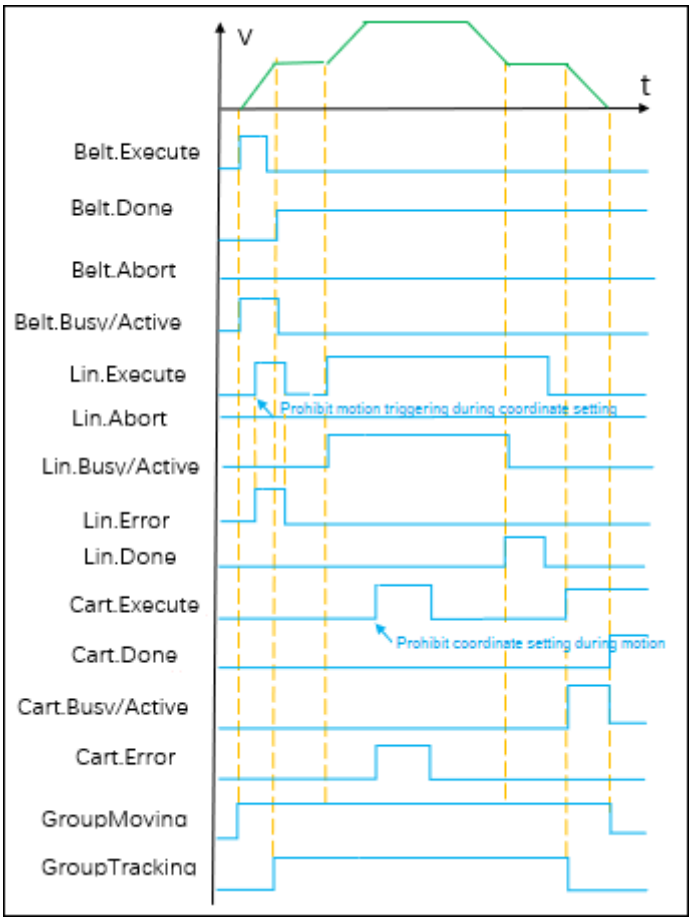
  1. Set the position of the conveyor belt relative to the robot base coordinate system. When an object is detected on the conveyor belt, associate the coordinate values of the object-attached coordinate system(relative to the conveyor base coordinate system) with the conveyor, and activate the tracking of the conveyor belt in the dynamic part coordinate system (Busy).
  2. Once the conveyor tracking is activated (Active), the axis group initiates motion planning relative to the dynamic coordinate system. This planning allows the Tool Center Point (TCP) to gradually synchronize its velocity with the conveyor belt within the dynamic workpiece space. Once achieved, the TCP maintains synchronous motion with the rigidly attached moving frame, and the Done output is set to TRUE.
  3. After the target position has been reached, execute the operation program in the dynamic part coordinate system, and call the motion function block to move within the dynamic coordinate system.
  4. After the operation is completed, switch back to the static coordinate system, and the axis group state is changed from GroupMoving to GroupStandby.



- **Precautions**
  - After the dynamic coordinate system is set, the point-to-point motion is not supported. The point-to-point motion function block in the dynamic coordinate system will directly report an error. It is prohibited to switch the coordinate system while the axis group motion function block is in motion. Otherwise, the function block will report an error.
  - After completing the dynamic coordinate system motion, select method (b) to return to the waiting position. When calling point-to-point instructions, avoid using the specified PCS coordinates in blending transition mode. Blending mode may introduce deviations in the dynamic coordinate system (PCS coordinates); therefore, ACS-type coordinates must be used instead.
  - The in-out variable ConveyorBelt of this instruction cannot be configured as the Axis variable in AxesGroup, otherwise the instruction will return an error.
  - The in-out variable ConveyorBelt of this instruction cannot be a rotary axis, otherwise an error will be generated.
  - This instruction cannot be executed when the axis group (AxesGroup) is in the following states: GroupErrorStop, homing, GroupDisabled, GroupStopping, or GroupMoving, and it is not idle. Otherwise, an error will be generated.
  - After the continuous trajectory module is enabled, it is prohibited to switch the coordinate system regardless of whether the axis group is in motion.
  - If an axis bound to a conveyor belt rotates while disabled, the axis group will follow the axis.
  - If an axis is in the disabled status, do not set the coordinate system.
  - After the conveyor tracking is enabled by setting a dynamic workpiece coordinate system for the axis group, it is allowed to reset the coordinate positions of the slave axis group (GroupSetPosition). However, resetting the coordinate positions of the tracked conveyor belt axis (via SetPosition) may easily cause an overspeed error in the tracking axis group. No restrictions are applied. Exercise caution.
- **Sequence diagram**

  Example sequence:

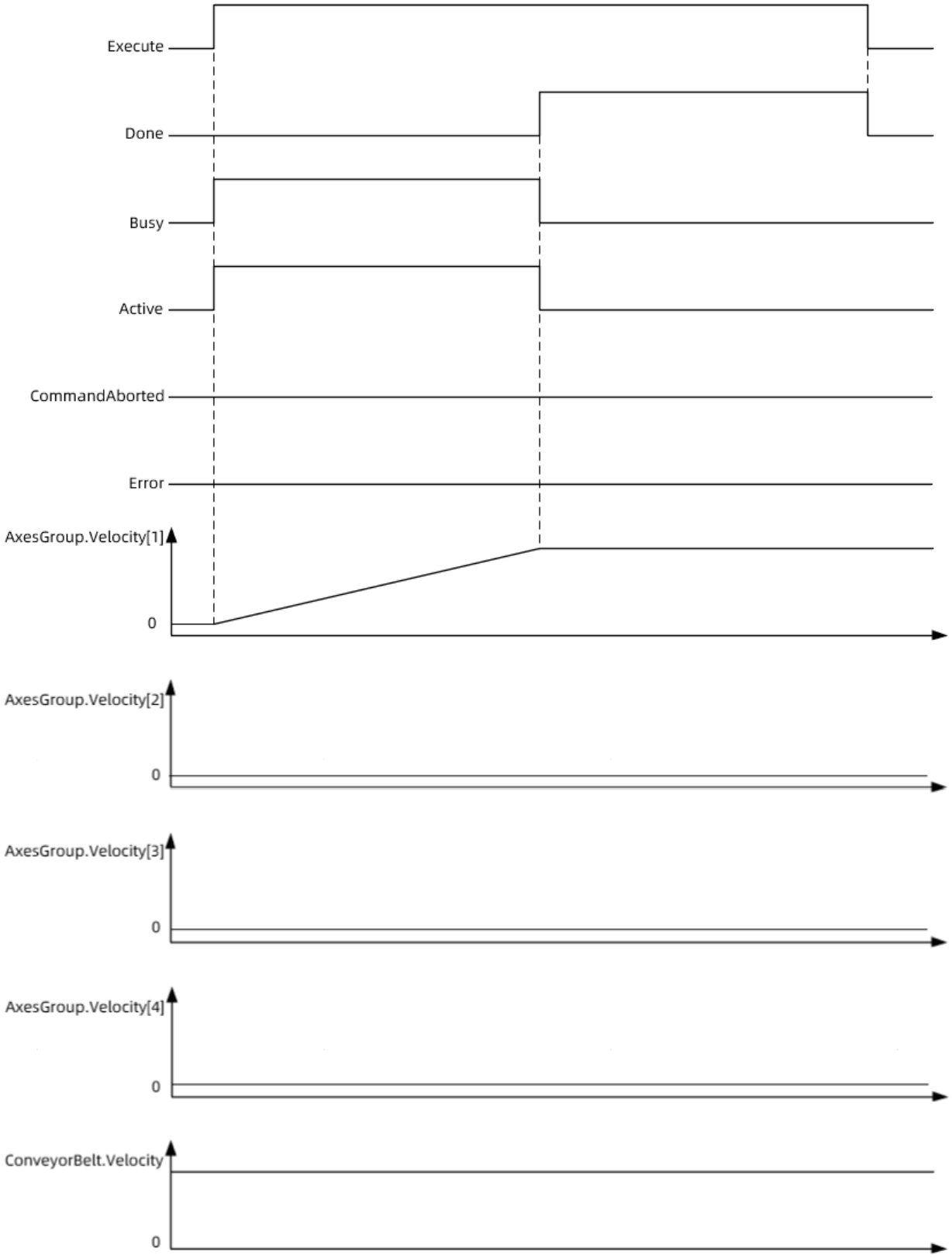




The Execute input of the function block must be triggered on a rising edge. The Done output of the function block indicates the instruction is executed successfully. The Busy output of the function block indicates that the current function block is being executed. The CommandAborted output of the function block indicates that the instruction is aborted by another motion control instruction, at which point the flag is set to TRUE.

On the rising edge of Execute, both Busy and Active outputs are set to TRUE.

The axis group initiates motion planning based on the dynamic coordinate system. This planning enables the Tool Center Point (TCP) to gradually synchronize its velocity with the synchronous belt within the dynamic workpiece space. Upon successful synchronization, the output variable "Done" is set to TRUE.



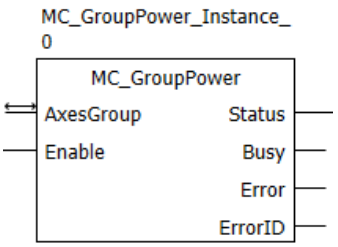
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.5.25 MC_GroupPower

This instruction switches the enable/disable state of a single axis within an axis group.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_GroupPower	Enable/disable axes in axis group		MC_GroupPower_Instance_0(AxesGroup := , Enable := , Status =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
AxesGroup	Axis group	18.3.3.2.4.1 AXES_GROUP_REF: Axis Group Structure	-	-	Mapped to an axis group, which is an instance of AXES_GROUP_REF.

  Input variable

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: The axes in the axis group are enabled. FALSE: The axes in the axis group are disabled.

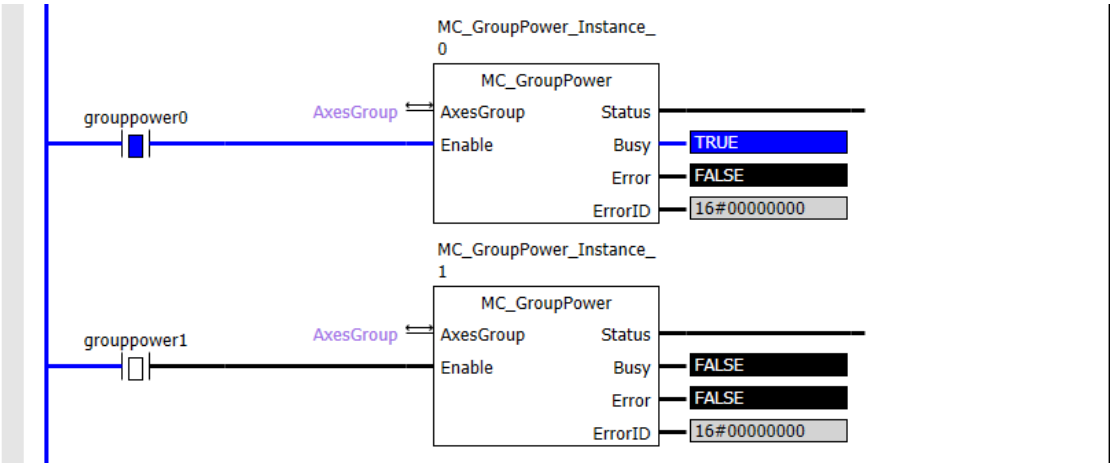
  Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Status	Ready for operation	BOOL	TRUE, FALSE	FALSE	Set to TRUE when all axes in the axis group are enabled.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**
  - This instruction can switch the enable state of the axis associated in AxesGroup.
  - When this instruction is triggered, it does not require the AxesGroup to be in any specific state.
  - When this instruction is executed normally, it does not change the axis group state. If you need to switch the axis group from GroupDisabled to GroupStandby, use the MC_GroupEnable instruction.

- If an error occurs during the instruction execution, the axis group state will switch to GroupErrorStop.

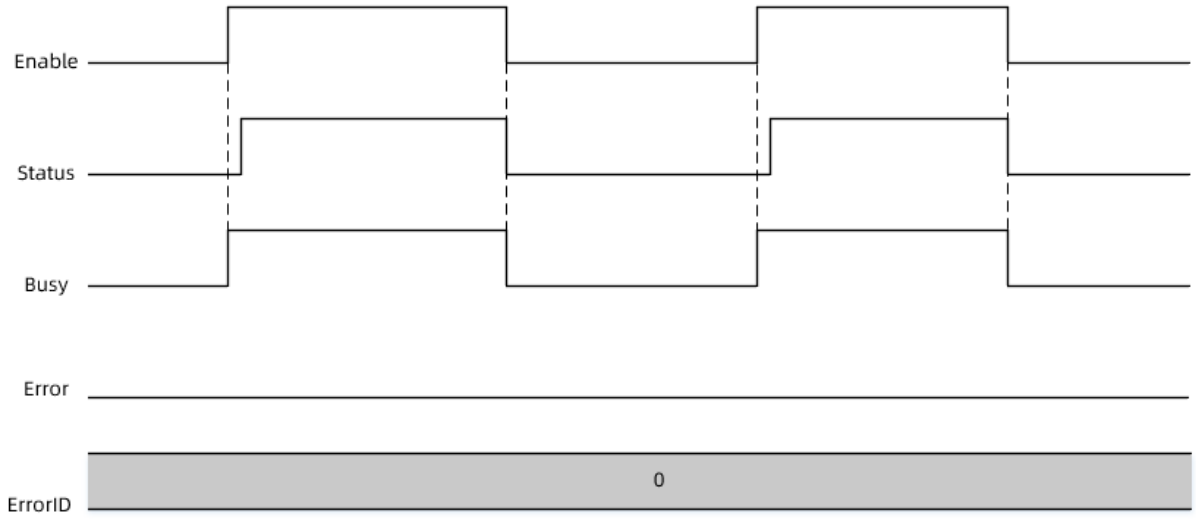
- **Precautions**
  - After the input parameter Enable is set to TRUE, the output parameter Status will not become TRUE until the servo operation on the axes bound to the axis group is complete. Perform any axis group operations only after the output parameter Status shows the value TRUE.
  - In field applications, it is recommended not to create multiple instances of the MC_GroupPower instruction for the same axis group. If this occurs, the enable state of the axes in the group is determined by the last instance of the MC_GroupPower instruction. The enable state of the axis is determined by the MC_GroupPower_Instance_1 function block, as shown in the following figure.



  - In field applications, it is recommended not to use the MC_Power instruction on an axis within an axis group where the MC_GroupPower instruction has already been used. Doing so may cause the enable state of that axis to become abnormal.

- **Sequence diagram**

  When the input parameter Enable is set to TRUE, the output parameter Busy shows the value TRUE; the axis bound to the axis group is ready for operation; the output parameter Status shows the value TRUE. When the input parameter Enable is set to FALSE, the output parameter Busy shows the value FALSE; the ready status for all axes bound to the axis group is cleared; the output parameter Status shows the value FALSE.



- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.6 High-Speed Axis

18.3.3.6.1 Instruction List

Instruction	Function	Library	Applicable Library Version	Supported encoder axis type
18.3.3.6.2 ENC_Counter	Encoder enable	MotionControl	Library version 1.1.0.0 and later	Bus, local
18.3.3.6.3 ENC_Reset	Reset encoder		Library version 1.1.0.0 and later	Bus, local
18.3.3.6.4 ENC_Preset	Encoder preset		Library version 1.1.0.0 and later	Bus, local
18.3.3.6.5 ENC_TouchProbe	Encoder probe		Library version 1.1.0.0 and later	Bus, local
18.3.3.6.6 ENC_ArrayCompare	Encoder one-dimensional array comparison		Library version 1.1.0.0 and later	Local
18.3.3.6.7 ENC_StepCompare	Encoder one-dimensional step comparison		Library version 1.1.0.0 and later	Local
18.3.3.6.8 ENC_ReadStatus	Encoder state read		Library version 1.1.0.0 and later	Bus
18.3.3.6.9 ENC_DigitalOutput	Encoder digital output control		Library version 1.1.0.0 and later	Bus
18.3.3.6.10 ENC_Compare	Encoder axis comparison output		Library version 1.1.0.0 and later	Local
18.3.3.6.11 ENC_SetUnit	Set encoder scaling (pulse-to-unit ratio)		Library version 1.1.0.0 and later	Local
18.3.3.6.12 ENC_SetLineRotationMode	Set linear rotation mode for encoder axis		Library version 1.1.0.0 and later	Local

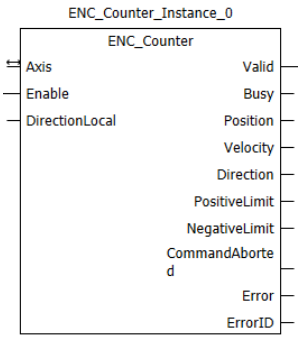
NOTE

Local encoder axis instructions and bus encoder axis instructions share the same instruction list.

18.3.3.6.2 ENC_Counter

This instruction is used to enable counting of the encoder axis.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
ENC_Counter	Encoder enable		ENC_Counter_Instance_0(Axis :=, Enable :=, DirectionLocal := , Valid => , Busy => , Position =>, Velocity => , Direction =>, PositiveLimit => , NegativeLimit => , CommandAborted => , Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: The instruction is enabled. FALSE: The instruction is not enabled.
DirectionLocal	Direction	INT	0: Positive 1: Negative	0	0: Positive. 1: Negative. Valid only for the local encoder axis.

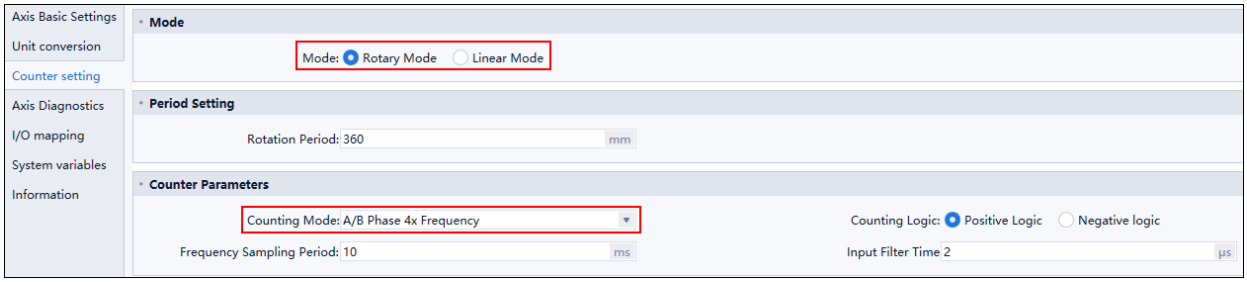
Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Valid status	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the status is valid.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is received.
Position	Current position	LREAL	-	0	Current position.
Velocity	Current velocity	LREAL	-	0	Current velocity.
Direction	Counting direction	BOOL	TRUE, FALSE	FALSE	Local encoder axis: TRUE indicates positive; FALSE indicates negative. Bus encoder axis: TRUE indicates positive; FALSE indicates negative.
PositiveLimit	Positive limit	BOOL	TRUE, FALSE	FALSE	The positive limit value reached in linear mode.
NegativeLimit	Negative limit	BOOL	TRUE, FALSE	FALSE	The negative limit value reached in linear mode.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE: The current instruction has been aborted by another instruction.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs in the function block.

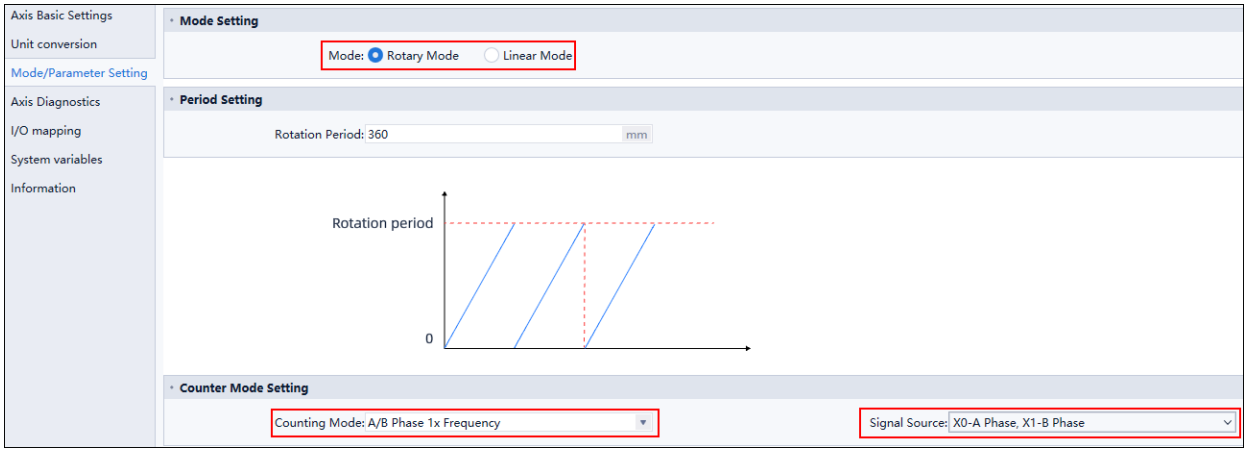
- **Function**

  Bus encoder axis:



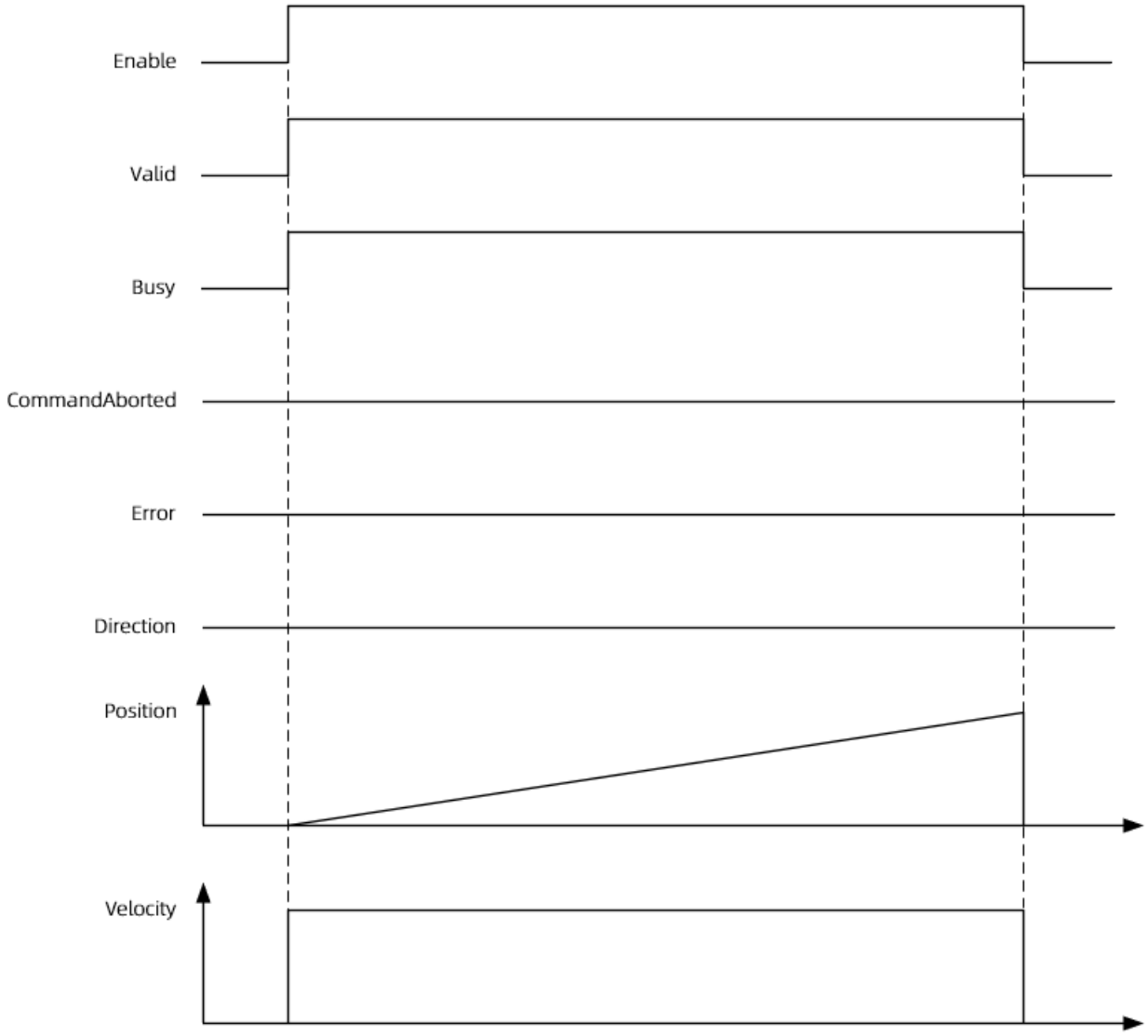
  - If the encoder axis is used as the master axis for a synchronization instruction, at least one pulse position must be sampled within one task cycle. Otherwise, the slave axis speed of the synchronization instruction will fluctuate between 0 and the target speed, causing motor vibration.
  - After Enable is triggered, modifying input parameters has no effect. Then, re-trigger the instruction.
  - When using the bus encoder counter instruction, configure the mode parameters and counter parameters. The mode shown in the figure above is rotation mode, and the counting mode is phase A/B 4x frequency multiplication counting mode.
  - The input parameter DirectionLocal is invalid for a bus encoder axis.
  - An error occurs if another counter is re-triggered when a counter has already been triggered. This does not affect the normal operation of the first counter.
  - After the instruction reports an error, all output status parameters (except position) must be cleared.
  - After Enable is reset, all output parameters are cleared.
  - If multiple gating functions are configured, the count will be incremented only when multiple configured input terminals are active simultaneously.
  - When multiple instructions are triggered, and they are associated with the same axis, the subsequently triggered instruction is treated as an error.

  Local encoder axis:

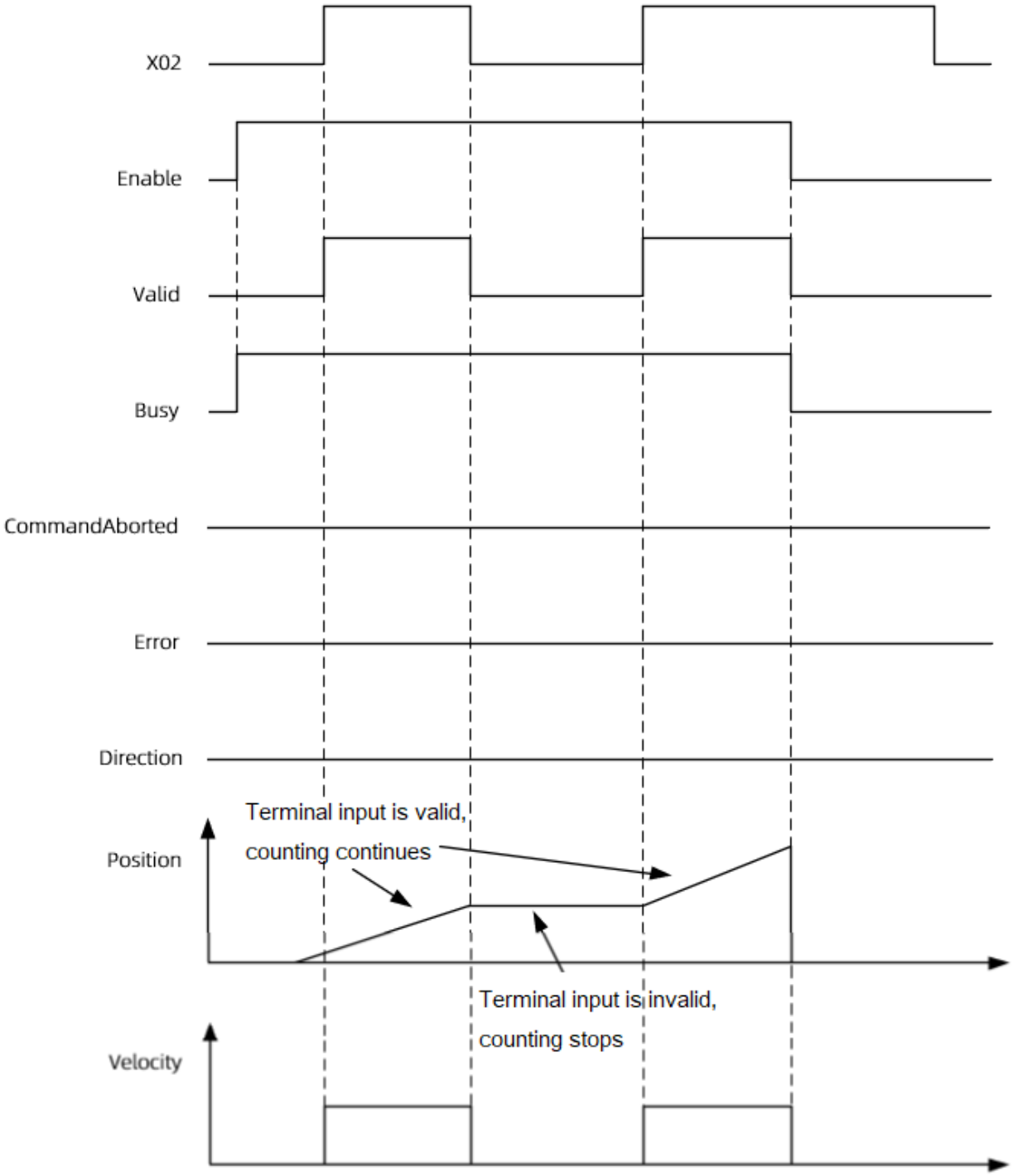


- The received pulse frequency must be greater than 1 Hz; if the frequency is less than 1, the displayed speed will be 0.
- When using the counter, configure related parameters such as mode, limits and counter mode. The above configuration is set to phase A/B 1x frequency multiplication counting mode, with signal sources X0 and X1. It operates in linear mode, with no upper or lower limit settings.

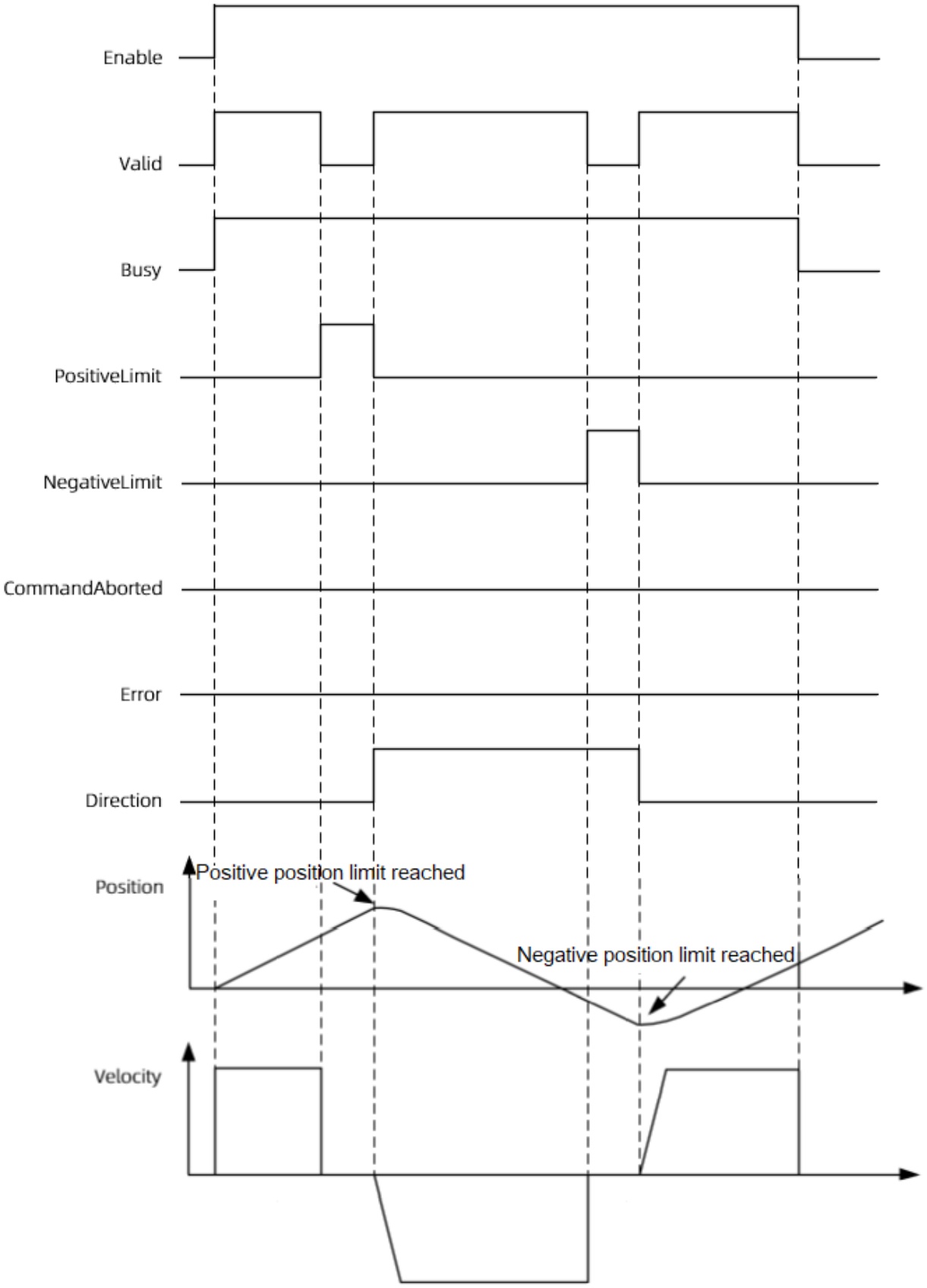
- **Sequence diagram**
  - **Bus encoder axis**
    - When the gating function of the digital terminal is not used, the encoder axis starts counting when the instruction input variable Enable is set to ON. The encoder axis stops counting when the instruction input variable Enable is set to OFF.



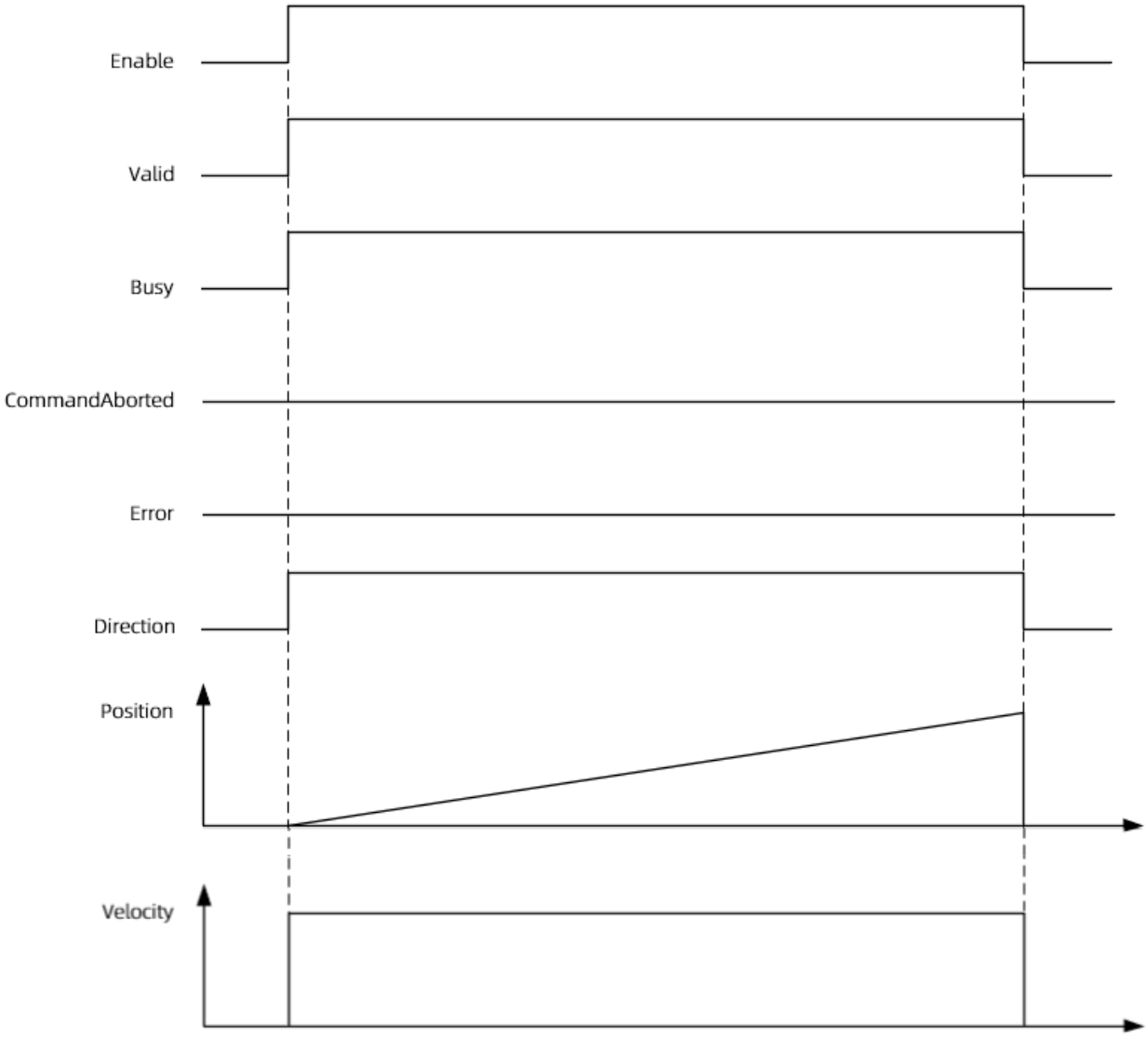
- When the gating function of the DI terminal is used: If the instruction input variable Enable is ON and the gating terminal signal is OFF, the instruction output variable Busy is ON, the variable Valid is OFF, and the bus encoder axis suspends counting. If the instruction input Enable is ON and the gating terminal signal is ON, the instruction output variables Busy and Valid are ON, and the bus encoder axis starts counting. If the instruction input Enable is OFF, the encoder axis stops counting.



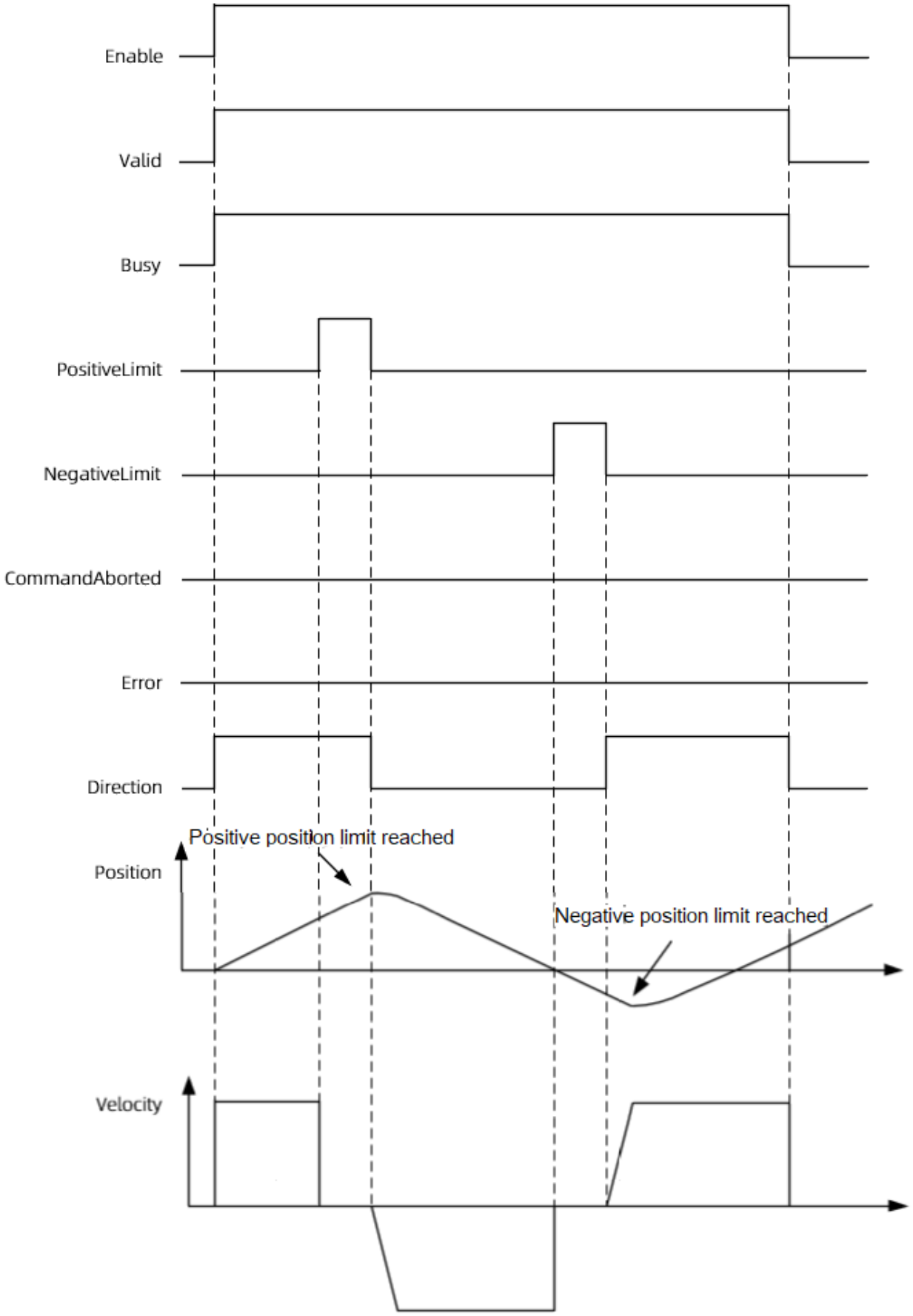
- In linear mode, if software limits are enabled, the counter stops counting when it reaches the limit value and the limit signal becomes active. When the pulse input reverses direction, the limit signal is reset and the counter counts in the opposite direction.



- **Local encoder axis**
  - When the instruction input variable Enable is ON, the output variables Busy and Valid are ON, and the encoder axis starts counting. When the instruction input variable Enable is OFF, the output variables Busy and Valid are OFF, and the encoder axis stops counting.



- In linear mode, if software limits are enabled, the counter stops counting when it reaches the limit value and the limit signal becomes active. When the pulse input reverses direction, the limit signal is reset and the counter counts in the opposite direction.



- **Precautions**

  When this instruction is used for a local encoder axis: If the variable Direction is set to TRUE, it indicates positive direction; if Direction is set to FALSE, it indicates negative direction. For a bus

encoder axis, if the variable Direction is set to TRUE, it indicates negative direction; if Direction is set to FALSE, it indicates positive direction.

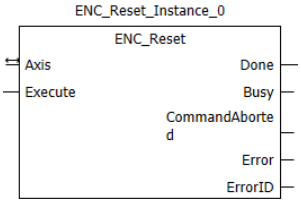
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.6.3 ENC_Reset

The instruction resets the faults for both the bus encoder axis and the local encoder axis.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
ENC_Reset	Reset encoder		ENC_Reset_Instance_0(Axis := , Execute := , Done =>, Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

  Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Execute the instruction on a rising edge.

  Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs in the function block.

- **Function**

  Bus encoder:

  - The axis status can only be switched to normal after an error occurs on an axis. If the instruction is used in a non-error status, it does not affect the axis status, and the Done output of the instruction is directly set.
  - This instruction can reset communication errors of the bus module and clear the axis-side error information in the fbeFBError structure. If the module error cannot be cleared, the instruction will return an error.

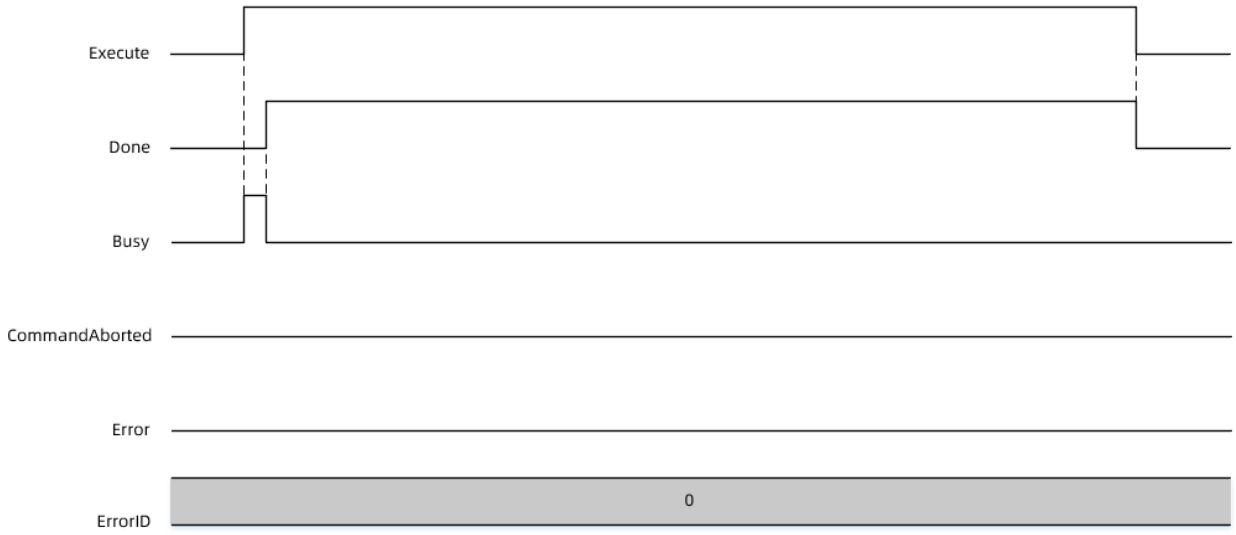
  Local encoder:

  - The error information in the fbeFBError structure on the local encoder axis can be cleared.

- **Sequence diagram**

  The parameter Execute of the function block must be triggered on a rising edge. The parameter Done of the function block indicates that the instruction has been executed successfully, and the parameter Busy indicates that the current function block is being executed.

  After this instruction is called, the output parameter Busy becomes TRUE on a rising edge of Execute. After successful reset, the output parameter Done is set to TRUE.



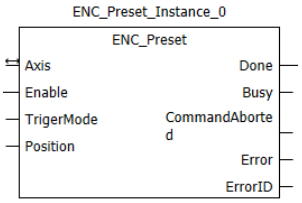
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.6.4 ENC_Preset

This instruction can be used to set the current position of the encoder axis as the value of the input parameter "Position".

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
ENC_Preset	Encoder preset		ENC_Preset_Instance_0(Axis :=, Enable :=, TrigerMode := , Position := , Done => , Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enable the instruction; FALSE: Disable the instruction.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
TrigerMode	Preset mode	INT	0 to 4	0	0: Triggered by the rising edge of the instruction (bus, local). 1: Triggered by the rising edge of the digital input terminal (bus, local). 2: Triggered by the falling edge of the digital input terminal (local). 3: Triggered by both rising and falling edges of the digital input terminal (local). 4: Triggered by the Z-phase signal (bus).
Position	Preset position	LREAL	-	0	Preset position.

Output variable

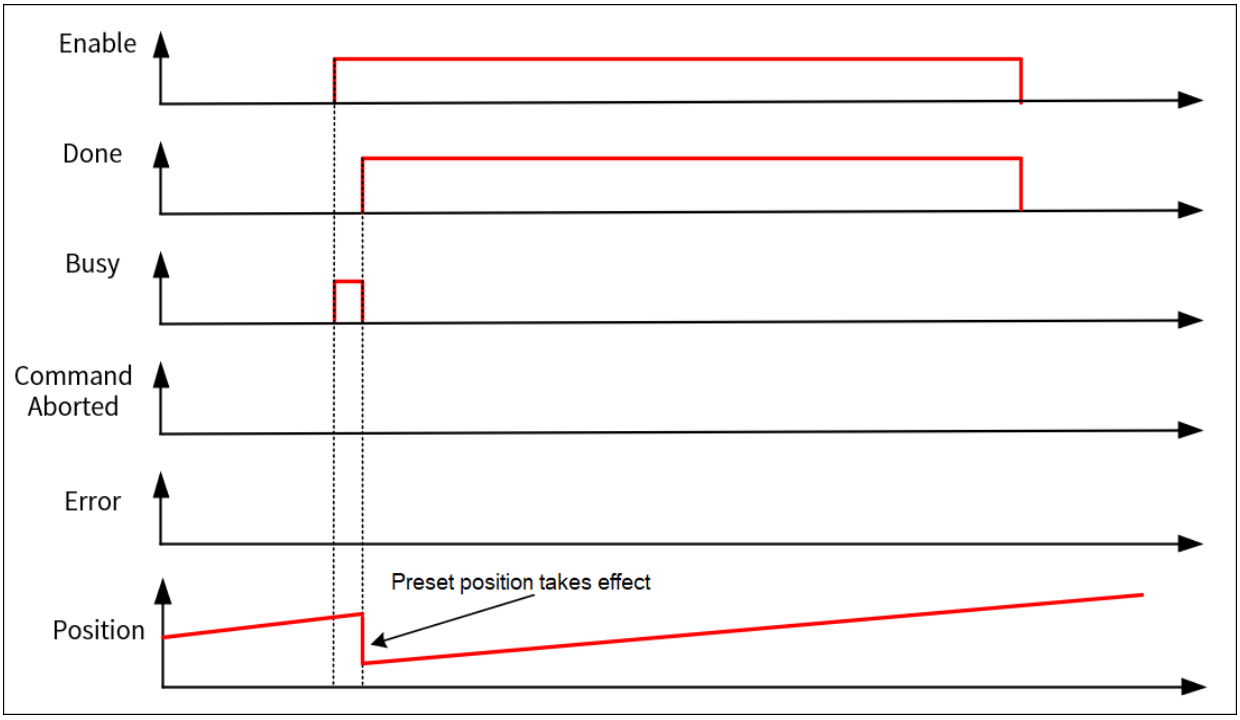
Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs in the function block.

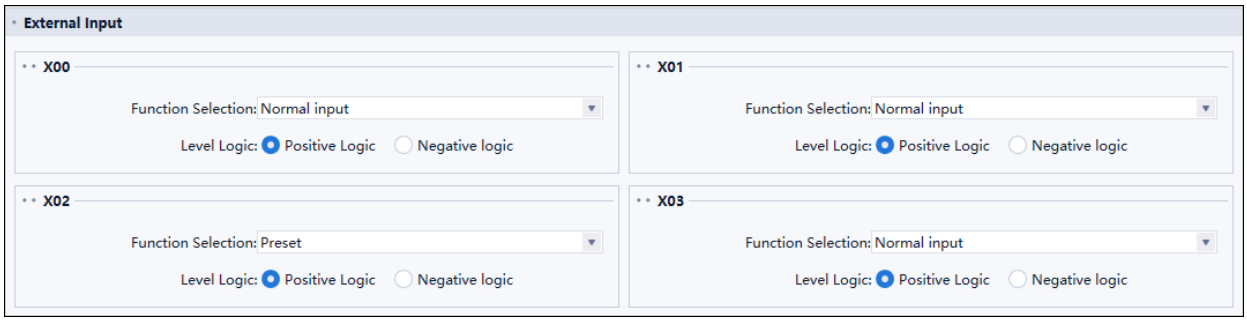
- **Function**
  - After Enable is triggered, the instruction sets the current position of the encoder axis to the preset position value specified by the input parameter Position if the condition is met according to the preset mode specified by the input parameter TrigerMode.
  - If the Enable input is active while any of the outputs (Done, Busy, CommandAborted, or Error) is active, changes to TrigerMode and Position will have no effect. Enable must be reset and then re-trigger the instruction.
  - If the axis controlled by this instruction is the master axis which is currently executing a synchronous motion instruction (such as MC_GearIn or MC_CamIn), triggering this instruction will result in an error.
  - When multiple instances of this instruction are triggered simultaneously, if TrigerMode is 0 (triggered on the rising edge of the instruction), the input parameter Position of the last triggered instance will be used as the reference.
  - When multiple instances of the function block for this instruction are triggered simultaneously, if TrigerMode is not 0, the first triggered function block takes effect, and the subsequent function blocks will return an error.
  - If TrigerMode is not 0 during the execution of the instruction, triggering another instance of it will cause the newly triggered function block to fault.

    Bus encoder axis:

    - When TrigerMode is set to 0, the position of the bus encoder axis is set when the Enable input is high. All output statuses are shown in the following figure:

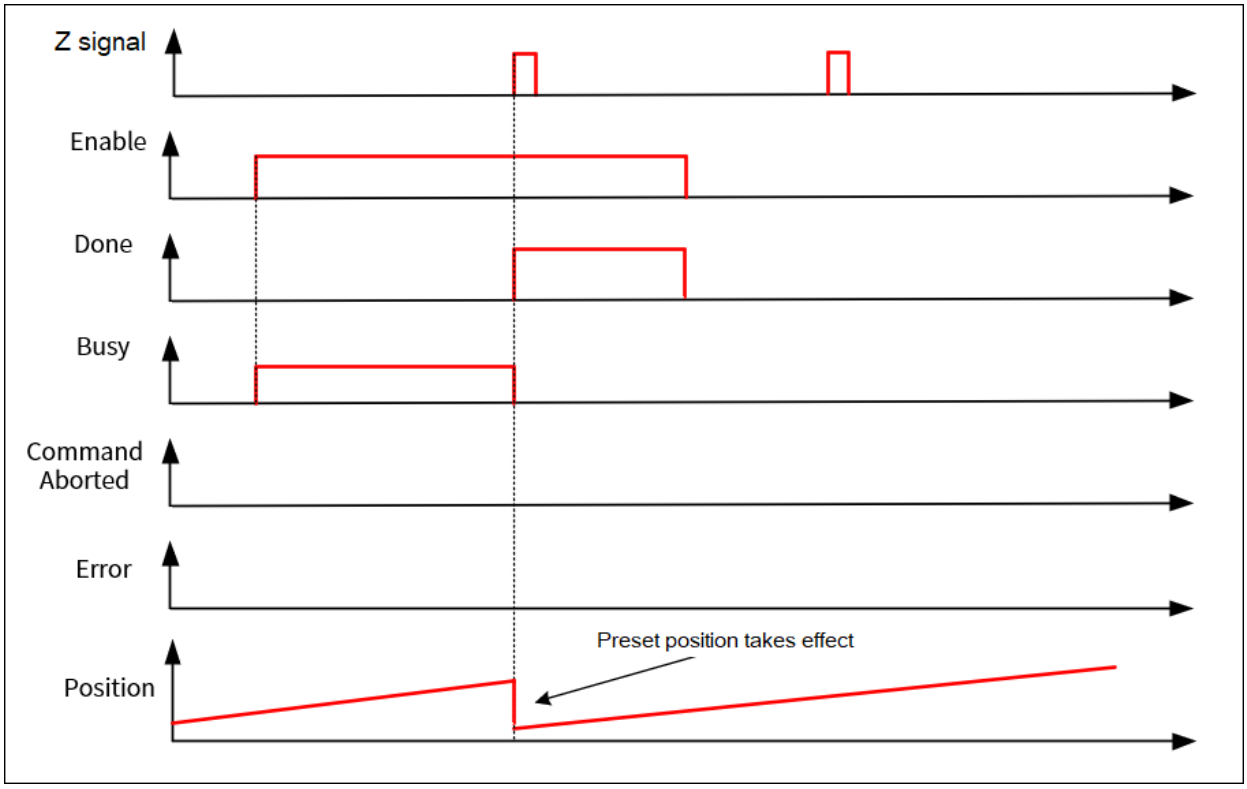


- When TrigerMode is set to 1, the position of the bus encoder axis is set on the rising edge of the digital input terminal. To select this mode, set the digital input terminal to preset function. If multiple digital terminals are set to preset function at the same time, the position setting can be completed when one of the inputs is valid.



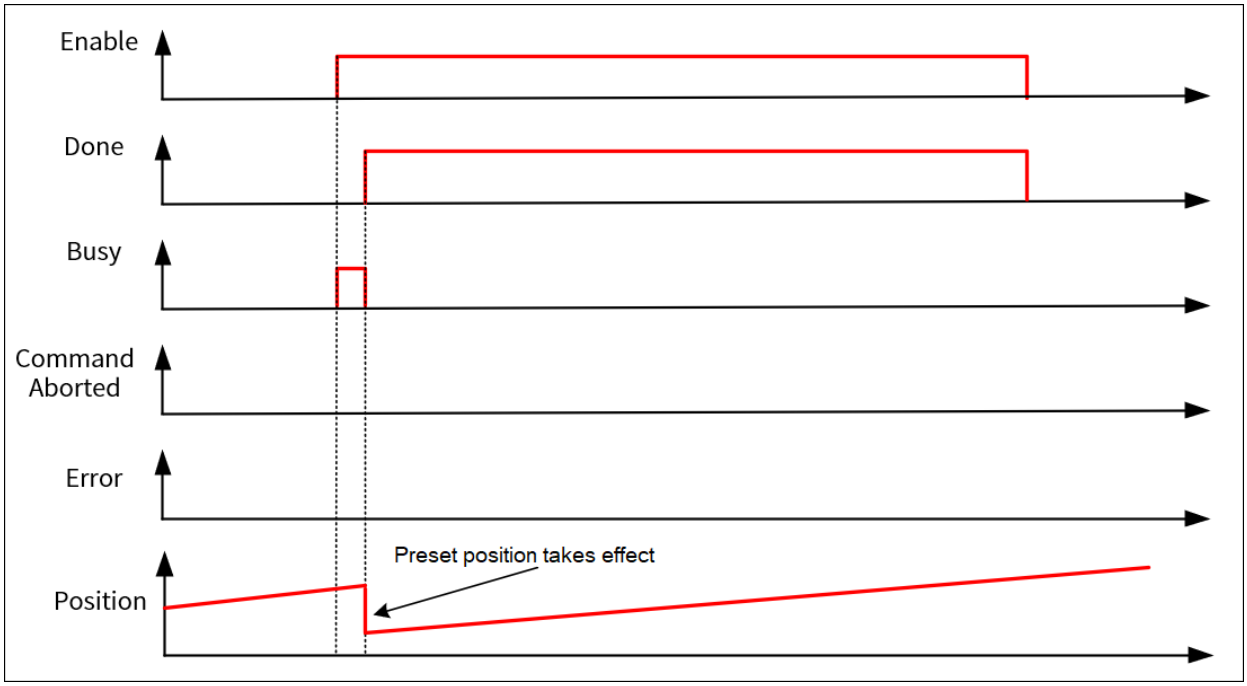


- When TrigerMode is set to 4, the position of the bus encoder axis is set on the rising edge of the Z-phase signal detected. All output statuses are shown in the following figure:



Local encoder axis:

- When TrigerMode is set to 0, the position of the local encoder axis is set when the Enable input is high. All output statuses are shown in the following figure:



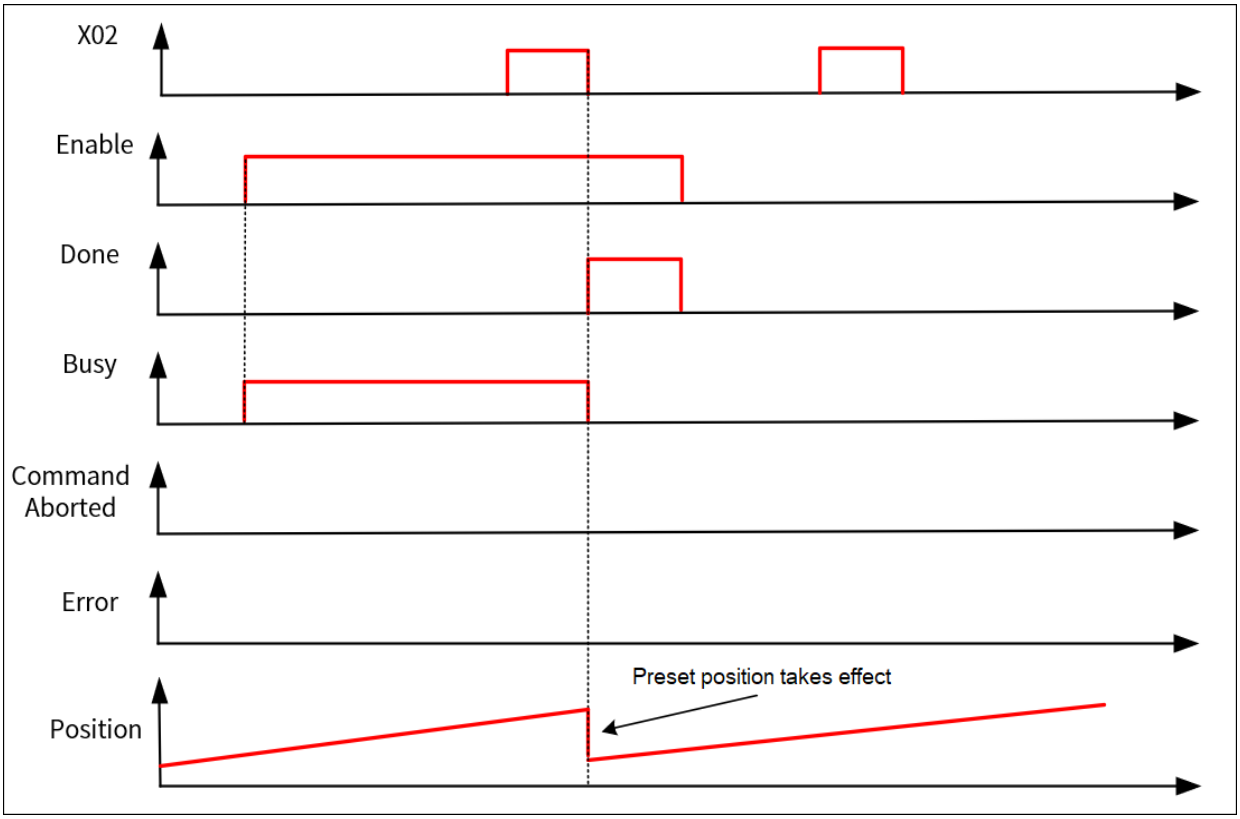
- The local encoder axis position is set on the rising edge, falling edge, or both edges (rising and falling edge) of the digital input terminal. To select this mode, the digital input terminal must be set to the preset function.



- When TrigerMode is set to 1, the position of the local encoder axis is set on the rising edge of the digital input terminal. When X02 rises after the instruction is triggered, the position preset function is completed.



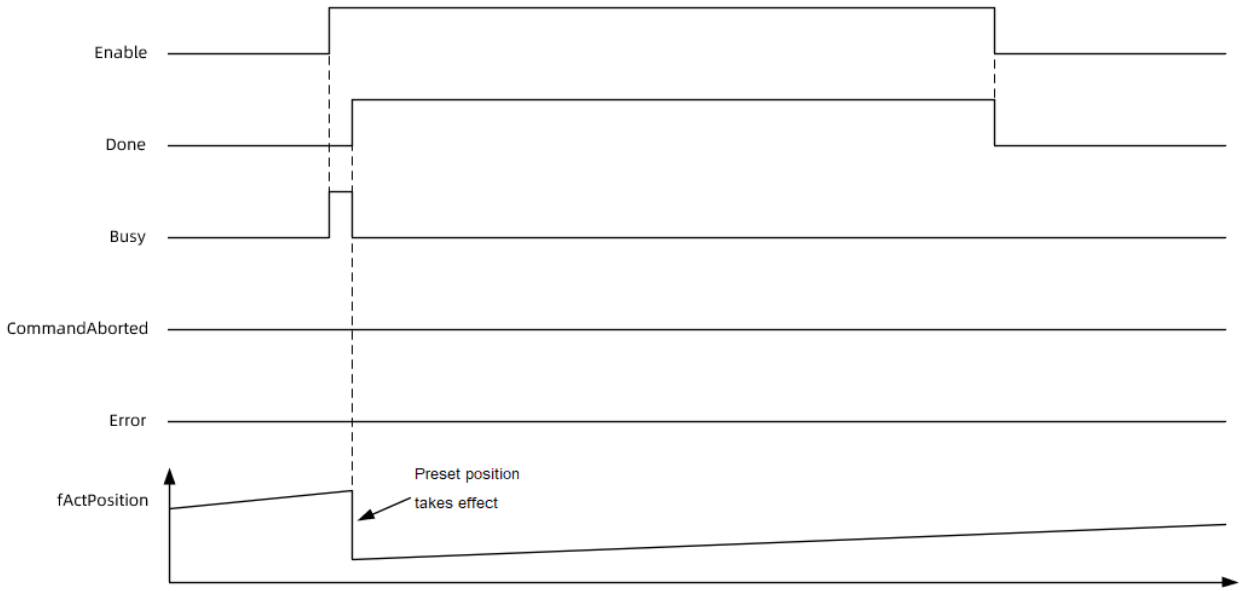
- When TriggerMode is set to 2, the position of the local encoder axis is set on the falling edge of the digital input terminal.



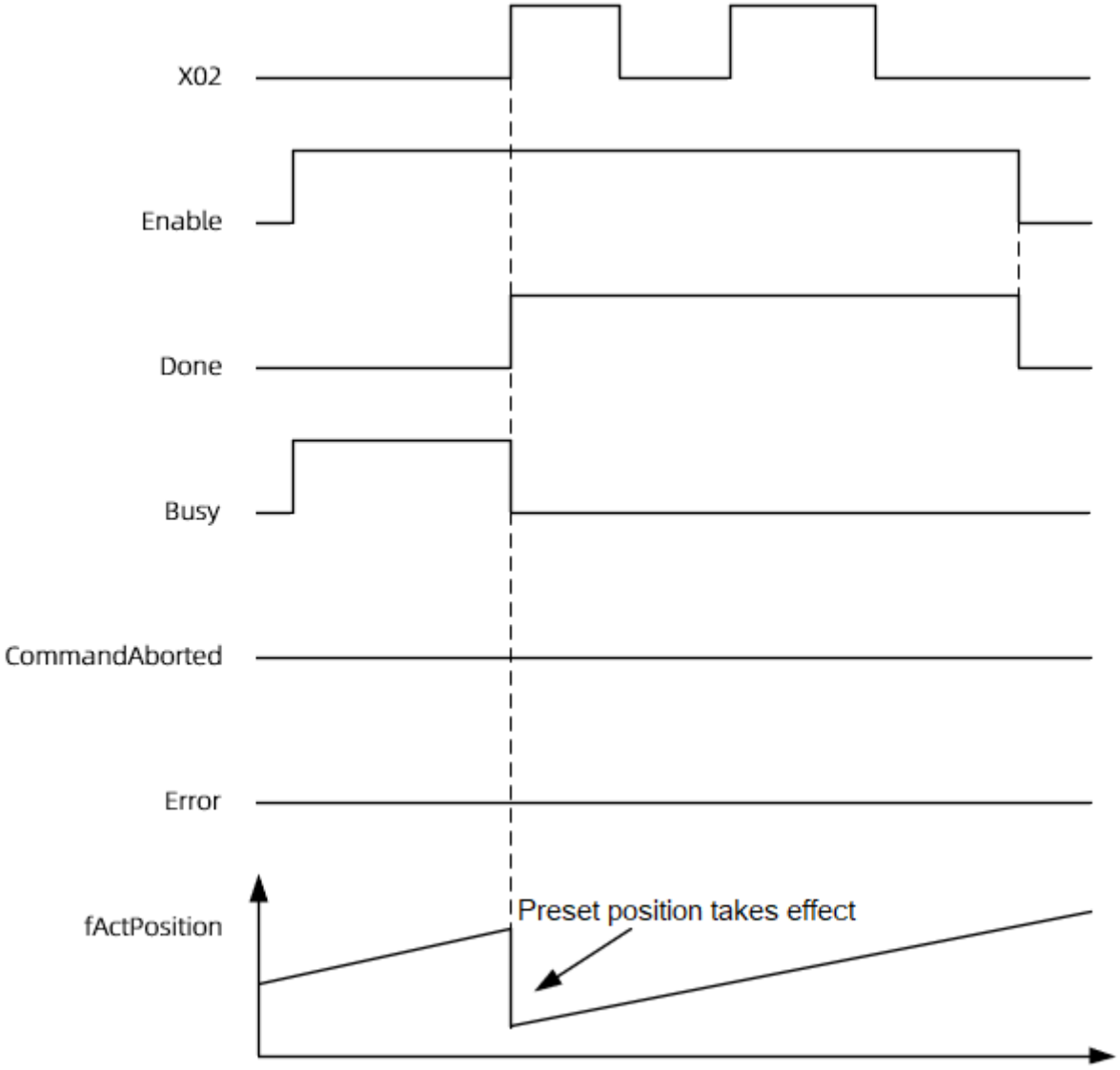
- When TrigerMode is set to 3, the position of the local encoder axis is set on the rising edge or the falling edge of the digital input terminal. When the instruction is active, whichever edge signal (rising or falling) that is generated first will trigger the process.



- **Sequence diagram**
  - When TrigerMode is set to 0, the position of the bus encoder axis is set when the Enable input is high.



  - When TrigerMode is set to 1, the position of the bus encoder axis is set on the rising edge of the digital input terminal X02.



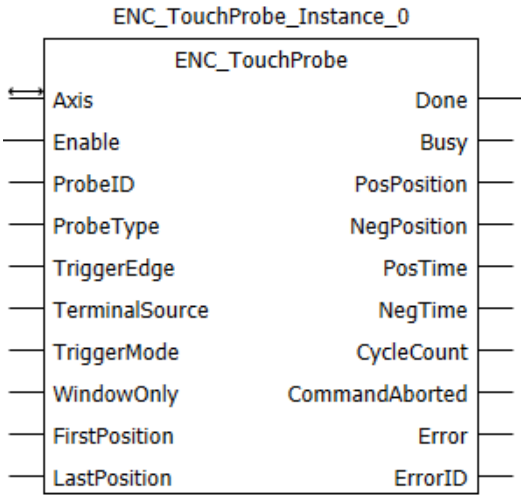
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.6.5 ENC_TouchProbe

This instruction records an axis position when a trigger signal occurs.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
ENC_TouchProbe	Encoder probe function		ENC_TouchProbe_Instance_0(Axis := , Enable := , ProbeID := , ProbeType := , TriggerEdge := , TerminalSource := , TriggerMode := , WindowOnly := , FirstPosition := , LastPosition := , Done => , Busy => , PosPosition => , NegPosition => , PosTime => , NegTime => , CycleCount => , CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: Enable the instruction; FALSE: Disable the instruction.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
ProbeID	Touch probe ID	INT	0 or 1	0	0: Touch probe 1. 1: Touch probe 2. Note: The local encoder axis only supports touch probe 1.
ProbeType	Probe type	WORD	-	0	0: Position probe. 1: Time probe. 2: Position-Time probe. Note: Only valid for bus encoder axes.
TriggerEdge	Trigger edge	INT	0, 1, 2	0	0: Rising edge. 1: Falling edge. 2: Rising edge and falling edge.
TerminalSource	Touch probe signal source	INT	0, 1	0	0: DI terminal. 1: Encoder Z-phase signal. Note: The local encoder axis only supports DI terminals.
TriggerMode	Trigger mode	INT	0, 1	0	0: Single trigger. 1: Continuous trigger.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
WindowOnly	Specify whether to enable the window to detect touch probe signals	BOOL	TRUE, FALSE	FALSE	0: Disabled. Probe signals are detected at all positions. 1: Probe signals are detected only when the current position is within the window.
FirstPosition	First Position	LREAL	-	0	Start position [u] from where the touch probe signals are detected.
LastPosition	Last Position	LREAL	-	0	Stop position [u] of the window.

Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is received.
PosPosition	Rising edge position	LREAL	-	0	The touch probe signals are detected on a rising edge.
NegPosition	Falling edge position	LREAL	-	0	The touch probe signals are detected on a falling edge.
PosTime	Rising edge latch time	ULINT	-	0	Rising edge latch time, in ns.
NegTime	Falling edge latch time	ULINT	-	0	Falling edge latch time, in ns.
CycleCount	Continuous mode trigger count	WORD	-	0	In continuous trigger mode, this value represents the valid count of the probe. CycleCount increments by 1 upon each probe trigger and cycles between 0 and 3.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE: The instruction has been aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**
  - In continuous trigger mode, Busy remains set, and Done is set for one cycle after successful latching.
  - After Enable is triggered, changes to input parameters have no effect. The instruction must be re-triggered for changes to take effect.

Bus encoder axis:



  - When the bus encoder axis is associated with the channel CH0 of the GR10_2HCE module, there are three cases depending on the selected probe type:
    - When ProbeType is set to position probe, PDO parameters 1B01 and 1B02 must be configured in the PDO configuration interface. At this time, the X00 and X01 terminals of the GR10_2HCE module can be used as probe terminals.

Name	Input/Output	Index	Subindex	Length (byte)	Type	Flag
Ch0 RPDO Mapping parameter 0	Output	16#1700	16#00	10.0		Editable
Ch0 RPDO Mapping parameter 1	Output	16#1701	16#00	12.0		F
Ch1 RPDO Mapping parameter 0	Output	16#1710	16#00	10.0		Editable
Ch1 RPDO Mapping parameter 1	Output	16#1711	16#00	12.0		F
Ch0 TPDO Mapping Parameter	Input	16#1B00	16#00	16.0		Editable
Ch0 touch probe pos value TPDO m...	Input	16#1B01	16#00	10.0		F
Ch0 touch probe neg value TPDO m...	Input	16#1B02	16#00	8.0		F
Ch0 touch probe pos time stamp TP...	Input	16#1B03	16#00	16.0		F
Ch0 touch probe neg time stamp T...	Input	16#1B04	16#00	16.0		F
Ch1 TPDO Mapping Parameter	Input	16#1B10	16#00	16.0		Editable
Ch1 touch probe pos value TPDO m...	Input	16#1B11	16#00	10.0		F
Ch1 touch probe neg value TPDO m...	Input	16#1B12	16#00	8.0		F
Ch1 touch probe pos time stamp TP...	Input	16#1B13	16#00	16.0		F
Ch1 touch probe neg time stamp T...	Input	16#1B14	16#00	16.0		F

    - When ProbeType is set to time probe, PDO parameters 1B03 and 1B04 must be configured in the PDO configuration interface. At this time, the X00 and X01 terminals of the GR10_2HCE module can be used as probe terminals.

Name	Input/Output	Index	Subindex	Length (byte)	Type	Flag
Ch0 RPDO Mapping parameter 0	Output	16#1700	16#00	10.0		Editable
Ch0 RPDO Mapping parameter 1	Output	16#1701	16#00	12.0		F
Ch1 RPDO Mapping parameter 0	Output	16#1710	16#00	10.0		Editable
Ch1 RPDO Mapping parameter 1	Output	16#1711	16#00	12.0		F
Ch0 TPDO Mapping Parameter	Input	16#1B00	16#00	16.0		Editable
Ch0 touch probe pos value TPDO m...	Input	16#1B01	16#00	10.0		F
Ch0 touch probe neg value TPDO m...	Input	16#1B02	16#00	8.0		F
Ch0 touch probe pos time stamp TP...	Input	16#1B03	16#00	16.0		F
Ch0 touch probe neg time stamp T...	Input	16#1B04	16#00	16.0		F
Ch1 TPDO Mapping Parameter	Input	16#1B10	16#00	16.0		Editable
Ch1 touch probe pos value TPDO m...	Input	16#1B11	16#00	10.0		F
Ch1 touch probe neg value TPDO m...	Input	16#1B12	16#00	8.0		F
Ch1 touch probe pos time stamp TP...	Input	16#1B13	16#00	16.0		F
Ch1 touch probe neg time stamp T...	Input	16#1B14	16#00	16.0		F

- When ProbeType is set to position-time probe, PDO parameters 1B01, 1B02, 1B03 and 1B04 must be configured in the PDO configuration interface. At this time, the X00 and X01 terminals of the GR10_2HCE module can be used as probe terminals.

Name	Input/Output	Index	Subindex	Length (byte)	Type	Flag
Ch0 RPDO Mapping parameter 0	Output	16#1700	16#00	10.0		Editable
Ch0 RPDO Mapping parameter 1	Output	16#1701	16#00	12.0		F
Ch1 RPDO Mapping parameter 0	Output	16#1710	16#00	10.0		Editable
Ch1 RPDO Mapping parameter 1	Output	16#1711	16#00	12.0		F
Ch0 TPDO Mapping Parameter	Input	16#1B00	16#00	16.0		Editable
Ch0 touch probe pos value TPDO m...	Input	16#1B01	16#00	10.0		F
Ch0 touch probe neg value TPDO m...	Input	16#1B02	16#00	8.0		F
Ch0 touch probe pos time stamp TP...	Input	16#1B03	16#00	16.0		F
Ch0 touch probe neg time stamp T...	Input	16#1B04	16#00	16.0		F
Ch1 TPDO Mapping Parameter	Input	16#1B10	16#00	16.0		Editable
Ch1 touch probe pos value TPDO m...	Input	16#1B11	16#00	10.0		F
Ch1 touch probe neg value TPDO m...	Input	16#1B12	16#00	8.0		F
Ch1 touch probe pos time stamp TP...	Input	16#1B13	16#00	16.0		F
Ch1 touch probe neg time stamp T...	Input	16#1B14	16#00	16.0		F

◦ When the bus encoder axis is associated with the channel CH1 of the GR10_2HCE module, there are three cases depending on the selected probe type:

- When ProbeType is set to position probe, PDO parameters 1B11 and 1B12 must be configured in the PDO configuration interface. At this time, the X10 and X11 terminals of the GR10_2HCE module can be used as probe terminals.

Name	Input/Output	Index	Subindex	Length (byte)	Type	Flag
Ch0 RPDO Mapping parameter 0	Output	16#1700	16#00	10.0		Editable
Ch0 RPDO Mapping parameter 1	Output	16#1701	16#00	12.0		F
Ch1 RPDO Mapping parameter 0	Output	16#1710	16#00	10.0		Editable
Ch1 RPDO Mapping parameter 1	Output	16#1711	16#00	12.0		F
Ch0 TPDO Mapping Parameter	Input	16#1B00	16#00	16.0		Editable
Ch0 touch probe pos value TPDO m...	Input	16#1B01	16#00	10.0		F
Ch0 touch probe neg value TPDO m...	Input	16#1B02	16#00	8.0		F
Ch0 touch probe pos time stamp TP...	Input	16#1B03	16#00	16.0		F
Ch0 touch probe neg time stamp T...	Input	16#1B04	16#00	16.0		F
Ch1 TPDO Mapping Parameter	Input	16#1B10	16#00	16.0		Editable
Ch1 touch probe pos value TPDO m...	Input	16#1B11	16#00	10.0		F
Ch1 touch probe neg value TPDO m...	Input	16#1B12	16#00	8.0		F
Ch1 touch probe pos time stamp TP...	Input	16#1B13	16#00	16.0		F
Ch1 touch probe neg time stamp T...	Input	16#1B14	16#00	16.0		F

- When ProbeType is set to time probe, PDO parameters 1B13 and 1B14 must be configured in the PDO configuration interface. At this time, the X10 and X11 terminals of the GR10_2HCE module can be used as probe terminals.

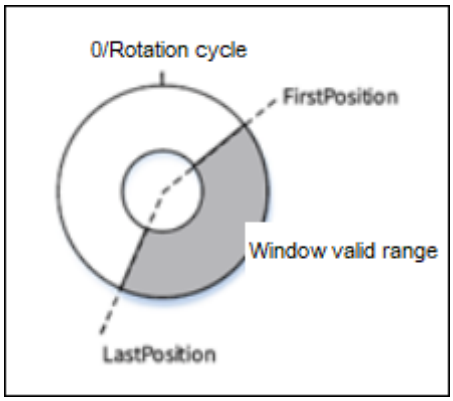
Name	Input/Output	Index	Subindex	Length (byte)	Type	Flag
Ch0 RPDO Mapping parameter 0	Output	16#1700	16#00	10.0		Editable
Ch0 RPDO Mapping parameter 1	Output	16#1701	16#00	12.0		F
Ch1 RPDO Mapping parameter 0	Output	16#1710	16#00	10.0		Editable
Ch1 RPDO Mapping parameter 1	Output	16#1711	16#00	12.0		F
Ch0 TPDO Mapping Parameter	Input	16#1B00	16#00	16.0		Editable
Ch0 touch probe pos value TPDO m...	Input	16#1B01	16#00	10.0		F
Ch0 touch probe neg value TPDO m...	Input	16#1B02	16#00	8.0		F
Ch0 touch probe pos time stamp TP...	Input	16#1B03	16#00	16.0		F
Ch0 touch probe neg time stamp T...	Input	16#1B04	16#00	16.0		F
Ch1 TPDO Mapping Parameter	Input	16#1B10	16#00	16.0		Editable
Ch1 touch probe pos value TPDO m...	Input	16#1B11	16#00	10.0		F
Ch1 touch probe neg value TPDO m...	Input	16#1B12	16#00	8.0		F
Ch1 touch probe pos time stamp TP...	Input	16#1B13	16#00	16.0		F
Ch1 touch probe neg time stamp T...	Input	16#1B14	16#00	16.0		F

- When ProbeType is set to position-time probe, PDO parameters 1B11, 1B12, 1B13 and 1B14 must be configured in the PDO configuration interface. At this time, the X10 and X11 terminals of the GR10_2HCE module can be used as probe terminals.

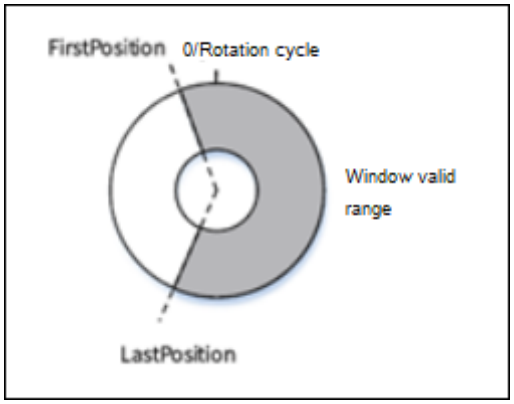
Name	Input/Output	Index	Subindex	Length (byte)	Type	Flag
Ch0 RPDO Mapping parameter 0	Output	16#1700	16#00	10.0		Editable
Ch0 RPDO Mapping parameter 1	Output	16#1701	16#00	12.0		F
Ch1 RPDO Mapping parameter 0	Output	16#1710	16#00	10.0		Editable
Ch1 RPDO Mapping parameter 1	Output	16#1711	16#00	12.0		F
Ch0 TPDO Mapping Parameter	Input	16#1B00	16#00	16.0		Editable
Ch0 touch probe pos value TPDO m...	Input	16#1B01	16#00	10.0		F
Ch0 touch probe neg value TPDO m...	Input	16#1B02	16#00	8.0		F
Ch0 touch probe pos time stamp TP...	Input	16#1B03	16#00	16.0		F
Ch0 touch probe neg time stamp T...	Input	16#1B04	16#00	16.0		F
Ch1 TPDO Mapping Parameter	Input	16#1B10	16#00	16.0		Editable
Ch1 touch probe pos value TPDO m...	Input	16#1B11	16#00	10.0		F
Ch1 touch probe neg value TPDO m...	Input	16#1B12	16#00	8.0		F
Ch1 touch probe pos time stamp TP...	Input	16#1B13	16#00	16.0		F
Ch1 touch probe neg time stamp T...	Input	16#1B14	16#00	16.0		F

- The instruction records the latched input parameters on the rising edge. Modification to the output parameters after latching is not effective. A new trigger is required to take effect.
- When WindowOnly is OFF, the window detection function is disabled. The axis position can be latched as long as the probe input signal is active.
- When WindowOnly = ON, the window detection function is active, and location is recorded only within the window.
  - In linear mode, the instruction can only detect the probe signal when the current position of the axis is within the interval set by FirstPosition and LastPosition.
  - In rotation mode:

    When FirstPosition is less than LastPosition, the valid window range is as follows:



    When FirstPosition is greater than LastPosition, the valid window range is as follows:



- After a single successful trigger, Done is set immediately. In continuous trigger mode (rising edge, falling edge, or both edges), Busy remains set. Upon a successful trigger, the captured value is updated to the PosPosition or NegPosition output parameter, and the Done flag is set for one cycle.

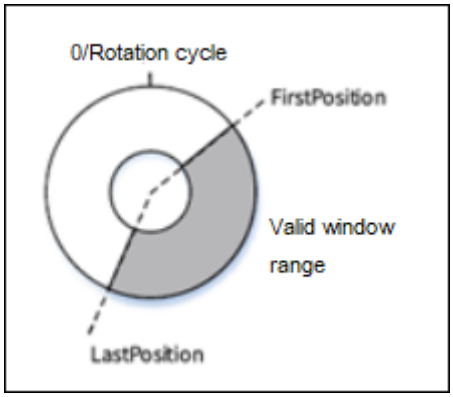
- Interruption relationship: On the same axis, if the probe IDs are the same, the subsequent instruction reports an error while the preceding instruction continues normal execution. If the probe IDs are different on the same axis, they can operate independently.
- When the window function is enabled, probe signal loss or out-of-range detection may occur near the window area. For example, in linear mode with a window range of 10 to 100, an ETC task cycle of 8 ms, and an instruction speed of 100, the position increments by 0.8 per cycle. If the starting position is 9.9, the probe signal is not responded to in this cycle. In the next cycle, the current position becomes 10.7, and position data between 10 and 10.7 will be lost. Similarly, if the current position is 99.9, the probe signal is detected within the window range during this cycle. In the next cycle, the current position becomes 100.7, and a probe signal can be output within the range 100 to 100.7.
- The window detection specifications are the same as those of MC_TouchProbe.

Local encoder axis:

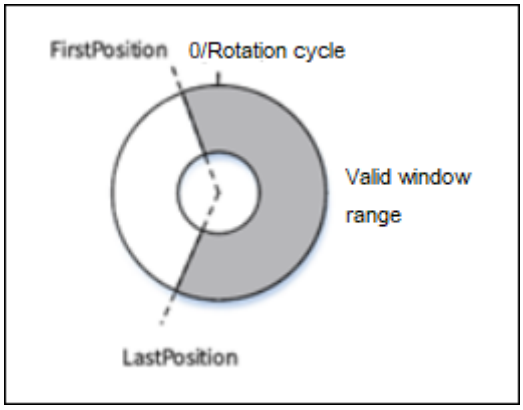


- The instruction records the latched input parameters on the rising edge. Modification to the output parameters after latching is not effective. A new trigger is required to take effect. The local encoder supports only one touch probe.
- When Enable = TRUE, if the instruction detects that the probe input specified by ProbeID is valid and meets the probe detection conditions, the current position of the axis will be latched.
- When WindowsOnly is FALSE, the window detection function is disabled. The current position of the axis can be latched as long as the probe input signal is active.
- When WindowsOnly = TRUE, the window detection function is active, and location is recorded only within the window.
  - In linear mode, the instruction can only detect the probe signal when the current position of the axis is within the interval set by FirstPosition and LastPosition.
  - In rotation mode:

    When FirstPosition is less than LastPosition, the valid window range is as follows:

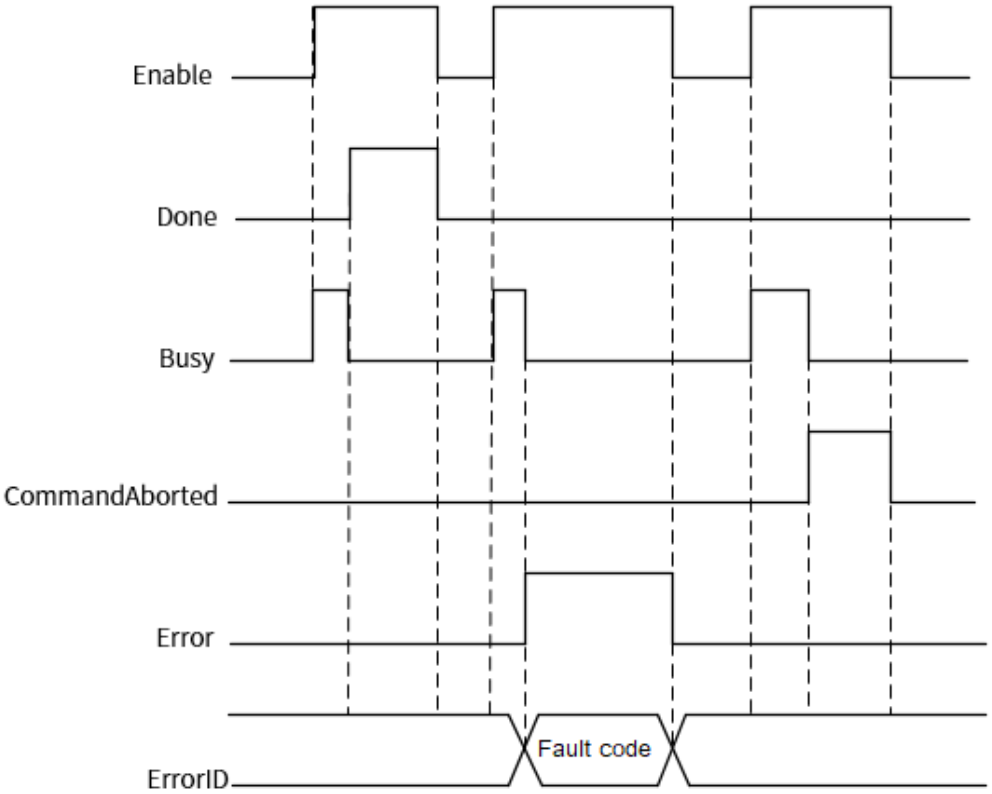


    When FirstPosition is greater than LastPosition, the valid window range is as follows.



- You can define the input parameter TriggerEdge of this instruction to confirm the detection of the rising edge, falling edge, or both rising edge and falling edge of the probe signal.
  - When the TriggerMode is set to "Single trigger" and the TriggerEdge is set to "Rising edge (Falling edge)", the instruction writes the detected value of a rising edge (falling edge) to "PosPosition"("NegPosition"), with the output parameter Done being set when one detection cycle is finished.
  - When the TriggerMode is set to "Continuous trigger" and the TriggerEdge is set to "Rising edge (Falling edge)", the instruction writes the detected value of a rising edge (falling edge) to "PosPosition"("NegPosition"), with the output parameter Done being reset one cycle after being set. The instruction continues to detect the signal of next rising edge (falling edge).
  - When the TriggerMode is set to "Single trigger" and the TriggerEdge is set to "Rising edge and falling edge", the instruction immediately writes the detected value of a rising edge to "PosPosition" and the detected value of a falling edge to "NegPosition", with the output parameter Done being set when one detection cycle is finished.
  - When the TriggerMode is set to "Continuous trigger" and the TriggerEdge is set to "Rising edge and falling edge", the instruction immediately writes the detected value of a rising edge to "PosPosition", with the output parameter Done being reset one cycle after being set; the instruction immediately writes the detected value of a falling edge to "NegPosition", with the output parameter Done being reset one cycle after being set. This instruction is executed repeatedly.
- When the window function is enabled, probe signal loss or out-of-range detection can occur near the window area. For example, in linear mode with a window range of 10 to 100, an ETC task cycle of 8 ms, and an instruction speed of 100, the position increments by 0.8 per cycle. If the starting position is 9.9, the probe signal is not responded to in this cycle. In the next cycle, the current position becomes 10.7, and position data between 10 and 10.7 will be lost. Similarly, if the current position is 99.9, the probe signal is detected within the window range during this cycle. In the next cycle, the current position becomes 100.7, and a probe signal can be output within the range 100 to 100.7.
- Only counter 0 supports both time and position latching, while other counters only support position latching.
- There is only one time probe register, so the same channel can be occupied by only one instruction of one axis at a time.
  - Configure the virtual probe with the local encoder 0 at the same time. If the probe function of the local encoder 0 is not used, the virtual probe will be not affected; conversely, the requirement must also be met.
  - Configure the virtual probe with the local encoder 0 at the same time. If the same probe channel is used for different instructions, an error will be reported. Use different probe channels to ensure normal operation.

- **Sequence diagram**



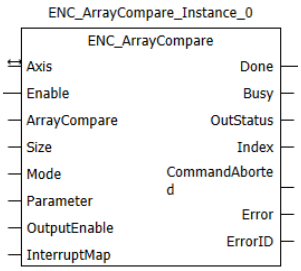
- **Precautions**
  - This instruction is only applicable to local encoder axes or bus encoder axes. For bus control axes, virtual axes, and local pulse axes using the probe function, use the MC_TouchProbe instruction.
  - In continuous trigger mode (TriggerMode set to 1), the Done pin has no output, and CycleCount increases by one after the latching is completed.
  - In rising and falling edge mode (TriggerEdge set to 2), the latching completes upon detection of one rising edge and one falling edge, and CycleCount increases by one.
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.6.6 ENC_ArrayCompare

This instruction compares one-dimensional arrays for the encoder.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
ENC_ArrayCompare	Encoder one-dimensional array comparison		ENC_ArrayCompare_Instance_0(Axis :=, Enable :=, ArrayCompare := , Size := , Mode := , Parameter := , OutputEnable := , InterruptMap := , Done => , Busy => , OutStatus =>, Index =>, CommandAborted => , Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: The instruction is enabled. FALSE: The instruction is not enabled.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
ArrayCompare	First element of the comparison point array.	ANY	-	0	Stores the first element of the comparison point array.
Size	Number of comparison values	INT	1 to 1000	1	Number of comparison values.
Mode	Comparison mode	INT	0 to 3	0	0: Reserved. 1: Time control 2: Pulse control 3: Level control
Parameter	Control parameters	LREAL	-	0	Time control: output active duration, in 100 μs. Pulse control: number of output valid pulses, in Unit. Level control: initial level; 0 indicates low level and non-zero indicates high level.
OutputEnable	Hardware output	INT	-	0	Hardware output enable (Valid only for local encoder axes) 0: The hardware terminal does not output. Non-zero: The hardware terminal outputs.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
InterruptMap	Interrupt number	INT	0 to 16	0	0: No interrupt is generated. 1: Mapped to external event Cmp1InterruptEvent. 2: Mapped to external event Cmp2InterruptEvent. ... 16: Mapped to external event Cmp16InterruptEvent.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
OutStatus	Port output	BOOL	TRUE, FALSE	FALSE	Port output status (reserved parameter).
Index	Compare value index	INT	0 to 999	0	Compare value index.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE: The instruction has been aborted. Aborted only by MC_AbortTrigger.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus encoder axis:

  This instruction does not support bus encoder axes.

  Local encoder axis:

  ◦ On the rising edge of the Enable input, check if all input parameters are valid. Enter ArrayCompare to compare arrays. If the number of comparison points is 2, the points must not overlap. If the number of

comparison points exceeds 2, neither overlapping points nor out-of-order arrangement is allowed. For example: [0,10,20,30,40,50,40,60...], which is an out-of-order sequence. The input parameter Size indicates the number of comparison points, which must not be less than 0 or greater than 1000. During instruction execution, modifying Size will not take effect. Modifying data in ArrayCompare will take effect, but the modified data will not be checked for validity. When modifying data in ArrayCompare, do so in ascending or descending order.

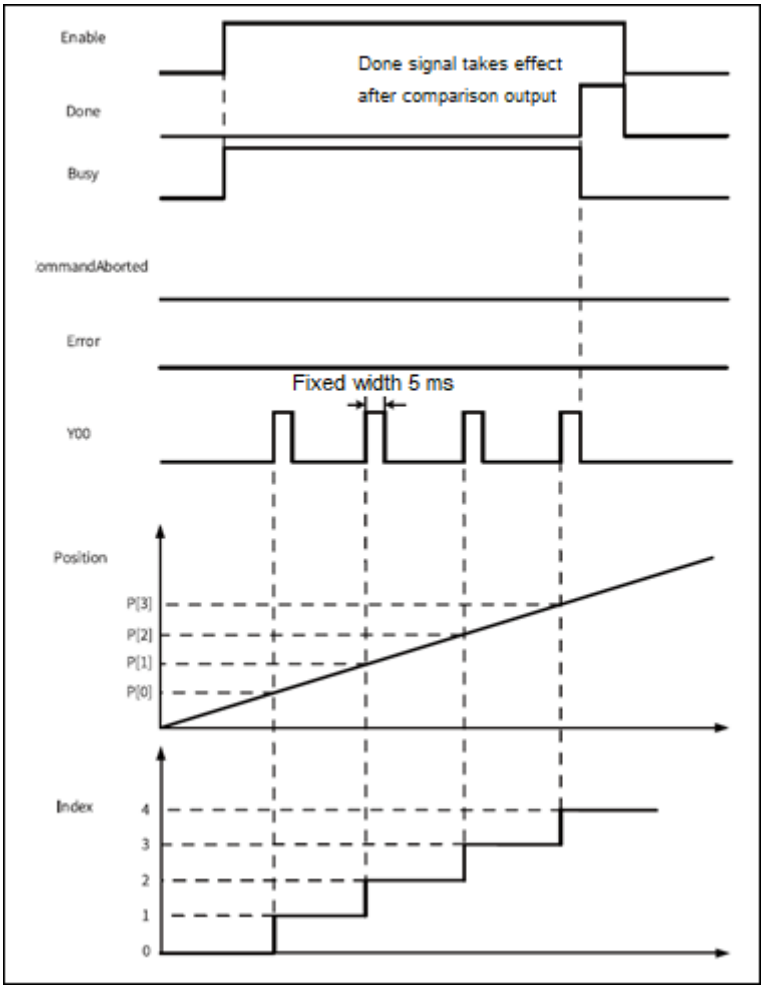
- The input parameter ArrayCompare defines the array of comparison points, and Size defines the number of points to compare. If the Size value exceeds the array length, the instruction will detect out-of-order or duplicate points when the rising edge check is triggered. When configuring Size, ensure it matches the actual number of comparison points in ArrayCompare.
- If the axis is a linear axis with soft limits, the points in ArrayCompare must not exceed the soft limit boundaries. If the axis is a rotary axis, the points in ArrayCompare must not exceed the rotation cycle or be less than 0.
- In the comparison output setting interface for the local encoder axis, enable "Comparison output enable", select the comparison output terminal, and choose whether to output based on time or user units of pulses. If the option "Comparison output enable" to download the program is not selected in the IDE, or if this option is selected, but no comparison output terminal is assigned, the instruction will return an error after it is triggered with the OutputEnable parameter set to a non-zero value.
- The basic principle of hardware output is that when the encoder axis reaches the set position, the digital output terminal is set to a high level. The duration or the number of pulses for which the output remains high can be configured in the IDE interface.



- Set the mode to keep the output terminal at a high level in the IDE interface:
  - Time mode: unit in ms.

    The digital output terminal outputs high level after the local encoder axis reaches the comparison point. The time duration during which the output terminal remains high level is determined by the parameter set in the background interface. Size specifies the number of comparison points and ArrayCompare specifies the comparison point array.

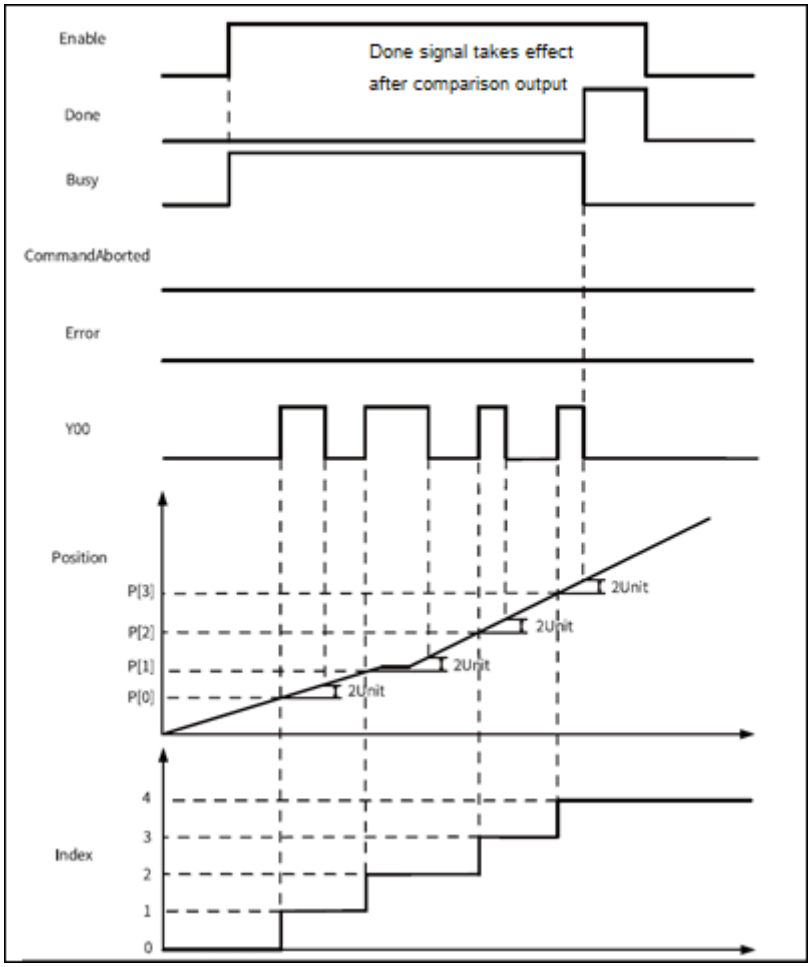
    Example of time mode: Set the comparison point array P[4], with an output time of 5ms at each comparison point, and the instruction starts running after Enable is enabled.



- Pulse mode: specifies the number of pulses.

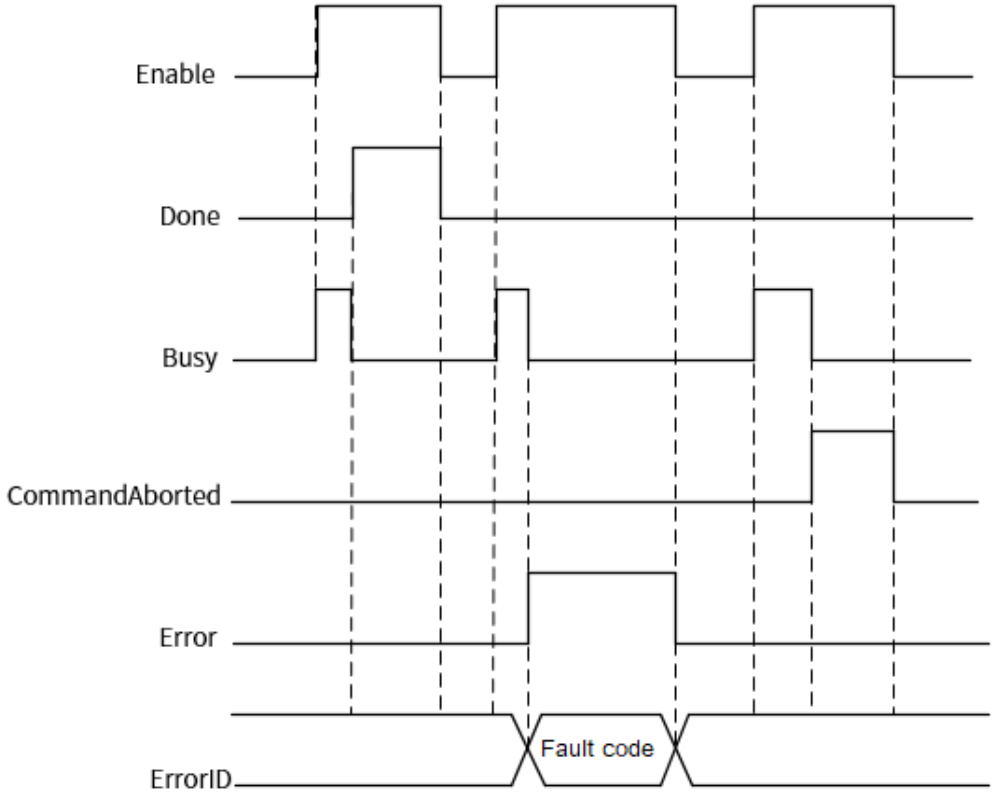
  The digital output terminal outputs high level after the local encoder axis reaches the comparison point. The high level will remain for the number of encoder pulses specified by the user. Size specifies the number of comparison points and ArrayCompare specifies the comparison point array.

  Example of pulse mode: Set the comparison point array P[4], with a duration of 2Unit at each comparison point, and the instruction starts running after Enable is enabled.



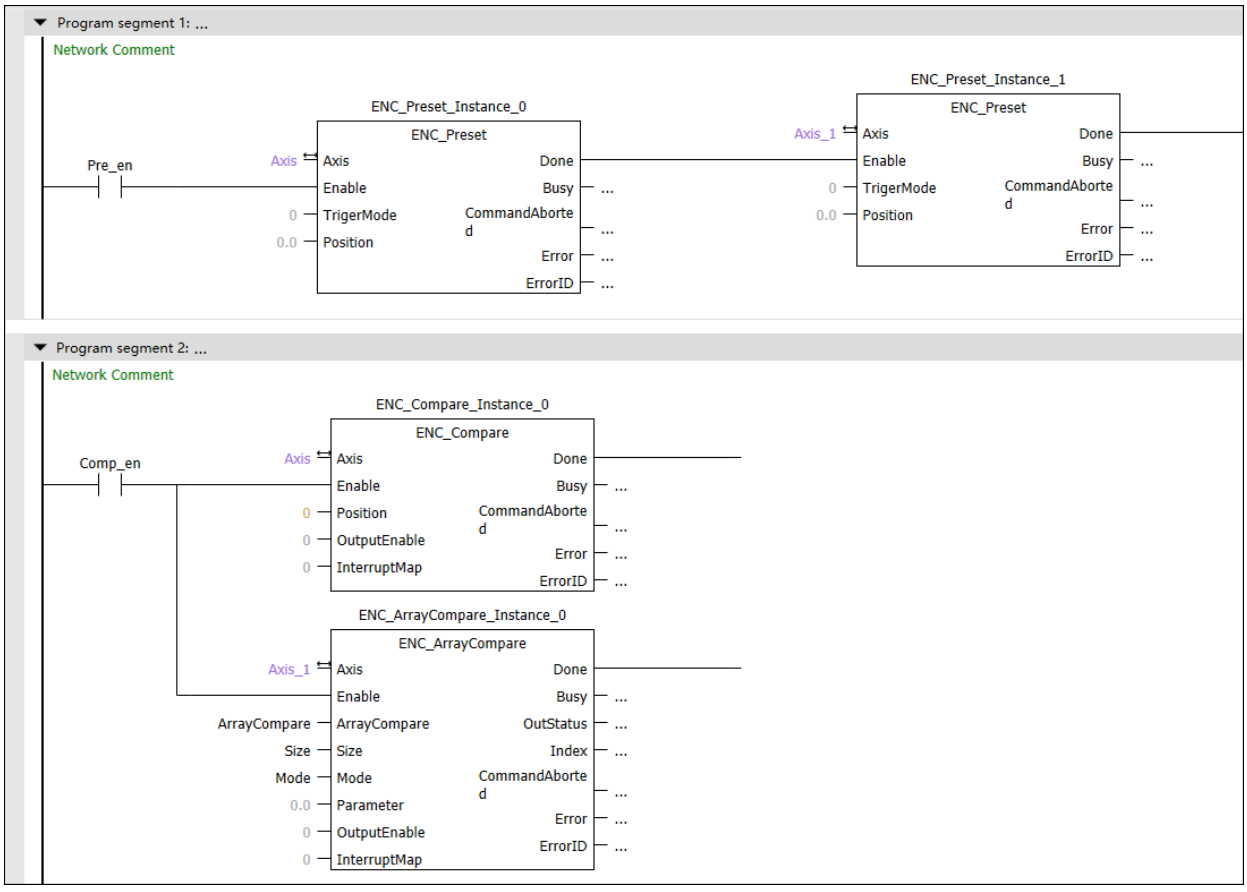
- If Enable is reset to FALSE during instruction execution, the remaining comparison points in ArrayCompare will not be compared.
- When OutputEnable is set to 0, the hardware terminal does not generate a comparison output signal. When OutputEnable is set to a non-zero value, the hardware terminal generates a comparison output signal.
- The InterruptMap parameter is temporarily retained and has no effect on the instruction blocks.

- **Sequence diagram**



- **Precautions**

  When using the comparison instruction and ENC_Preset instruction, pay attention to the order of the instructions in the project configuration. The following situations will prevent the comparison instruction from generating an output.



In the above project configuration, after ENC_Compare_Instance_0 and ENC_ArrayCompare_Instance_1 are executed for the first time, that is, when ENC_Compare_Instance_0.Done and ENC_ArrayCompare_Instance_1.Done are both TRUE, the comparison instruction will not generate an output if you reset Pre_en and Comp_en after triggering Pre_en and Comp_en simultaneously. That is, when the encoder axis is counting, if ENC_Preset_Instance_0 and ENC_Preset_Instance_1 are all triggered simultaneously, the comparison instruction is activated only after the axis position has already passed comparison point. Therefore, the comparison instruction assumes that the encoder axis has never passed comparison point, and thus does not generate any corresponding output.

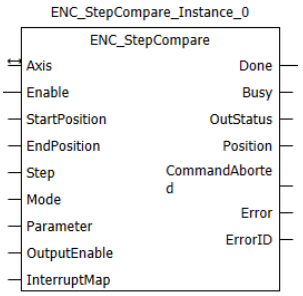
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.6.7 ENC_StepCompare

This instruction is used for encoder one-dimensional step comparison.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
ENC_StepCompare	Encoder one-dimensional step comparison		ENC_StepCompare_Instance_0(Axis :=, Enable :=, StartPosition := , EndPosition := , Step := , Mode := , Parameter := , OutputEnable := , InterruptMap := , Done => , Busy => , OutStatus =>, Position =>, CommandAborted => , Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: The instruction is enabled. FALSE: The instruction is not enabled.
StartPosition	Comparison start position	LREAL	-	0	Start position for comparison.
EndPosition	Comparison end position	LREAL	-	0	End position for comparison.
Step	Step	LREAL	-	0	Step size.
Mode	Comparison mode	INT	0 to 3	0	0: Reserved. 1: Time control 2: Pulse control 3: Level control
Parameter	Control parameters	LREAL	-	0	Time control: output active duration, in 100 μs. Pulse control: number of output valid pulses, in Unit. Level control: initial level; 0 indicates low level and non-zero indicates high level.
OutputEnable	Hardware terminal output enable	INT	0 to 1	0	Hardware output enable (Valid only for local encoder axes). 0: The hardware terminal does not output. Non-zero: The hardware terminal outputs.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
InterruptMap	Interrupt number	INT	0 to 16	0	0: No interrupt is generated. 1: Mapped to external event Cmp1InterruptEvent. 2: Mapped to external event Cmp2InterruptEvent. ... 16: Mapped to external event Cmp16InterruptEvent.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the axis instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE after the instruction is received.
OutStatus	Port output	BOOL	TRUE, FALSE	FALSE	Port output status (reserved parameter).
Position	Comparison value	LREAL	-	0	Comparison value
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE: The instruction has been aborted. Aborted only by MC_AbortTrigger.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

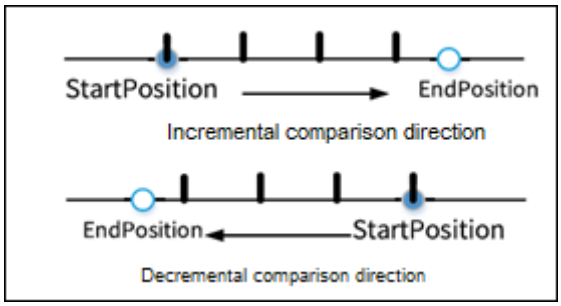
- **Function**

Bus encoder axis:

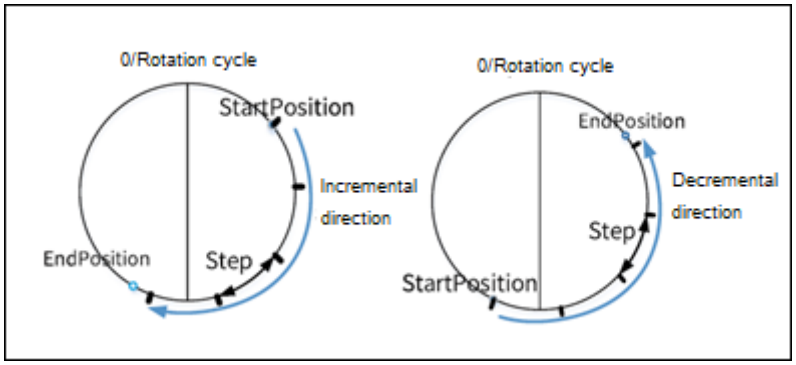
This instruction does not support bus encoder axes.

Local encoder axis:

- On the rising edge of the Enable input, check if all input parameters are valid. If the axis is defined as a linear axis and software limits are set, the input parameters StartPosition and EndPosition must not exceed the soft limit boundaries. If it is defined as a rotary axis, the input parameters StartPosition and EndPosition must not be less than 0 or greater than the rotation cycle.
- In this instruction, StartPosition is used to set the start position of the comparison point, and EndPosition is used to set the output of the comparison point. After Enable is triggered, changes to input parameters have no effect. The instruction must be re-triggered for changes to take effect.
- In linear mode, the rules for setting comparison points are as follows: When StartPosition is less than EndPosition, Step must be set to a positive number, indicating an increment counting comparison method. When StartPosition is greater than EndPosition, Step must be set to a negative number, indicating a decrement counting comparison method.



- In rotation mode, the rules for setting comparison points are as follows: When StartPosition is less than EndPosition, Step must be set to a positive number, indicating an increment counting comparison method. When StartPosition is greater than EndPosition, Step must be set to a negative number, indicating a decrement counting comparison method.



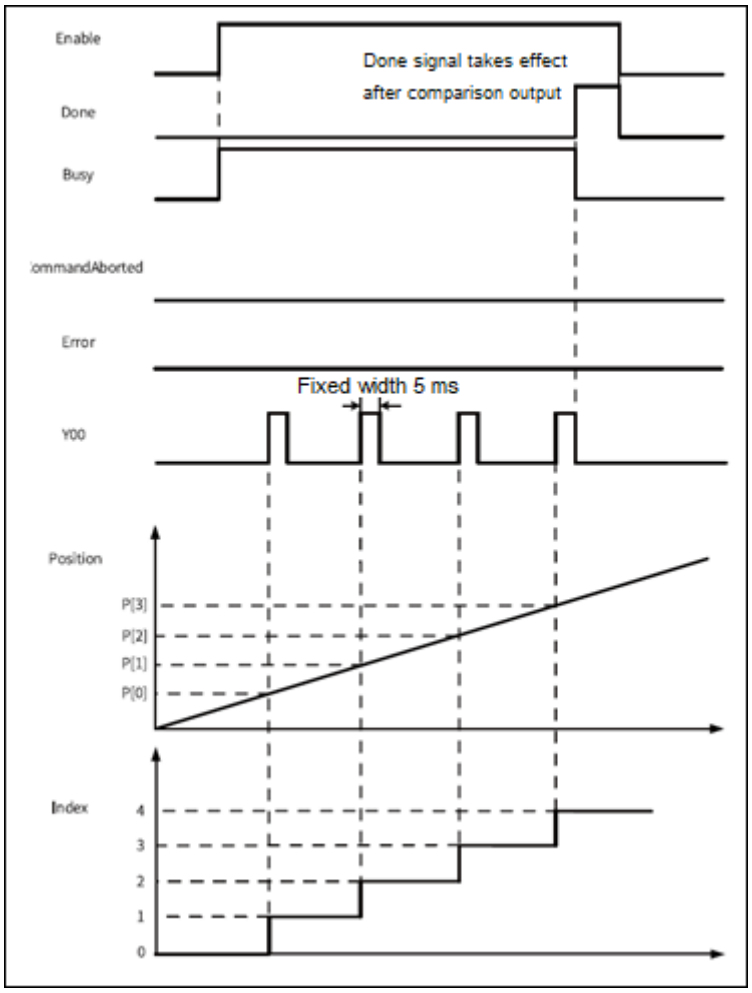
- The first comparison point must be the point specified by StartPosition, but the last comparison point is not necessarily the one specified by EndPosition. When StartPosition is 10, EndPosition is 25. When Step is 10, the comparison points are 10 and 20.
- This instruction supports both time and pulse modes, which are configured via the backend configuration interface.
- In the comparison output setting interface for the local encoder axis, enable "Comparison output enable", select the comparison output terminal, and choose whether to output based on time or user units of pulses. If the option "Comparison output enable" to download the program is not selected in the IDE, or if this option is selected, but no comparison output terminal is assigned, the instruction will return an error after it is triggered with the OutputEnable parameter set to a non-zero value.



- The basic principle of hardware output is that when the encoder axis reaches the set position, the digital output terminal is set to a high level. The duration or the number of pulses for which the output remains high can be configured in the IDE interface.
- Set the mode to keep the output terminal at a high level in the IDE interface:
  - Time mode: unit in ms.

    The digital output terminal outputs high level after the local encoder axis reaches the comparison point. The time duration during which the output terminal remains high level is determined by the parameter set in the background interface. The number of comparison points is specified by |(StartPos-EndPosition)/Step|. The comparison points are generated by incrementing StartPos by Step until it does not exceed EndPosition.

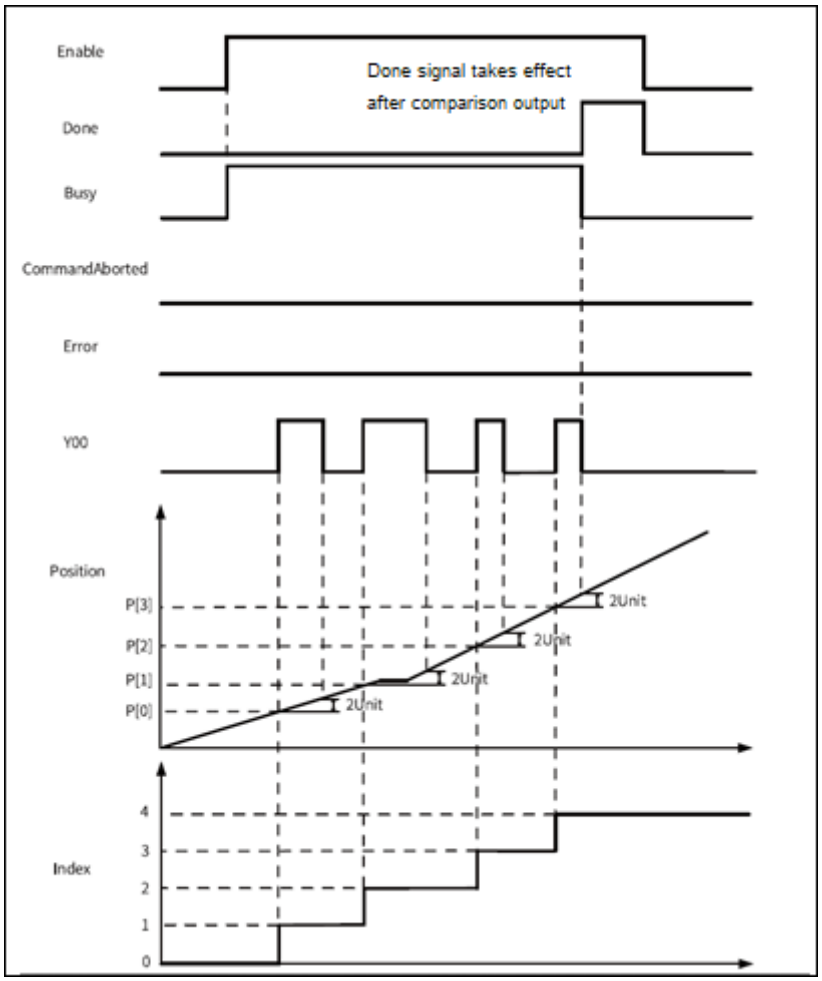
    Example of time mode: Set the comparison point array P[4], with an output time of 5 ms at each comparison point. The instruction starts running after Enable is enabled.



  - Pulse mode: specifies the number of pulses.

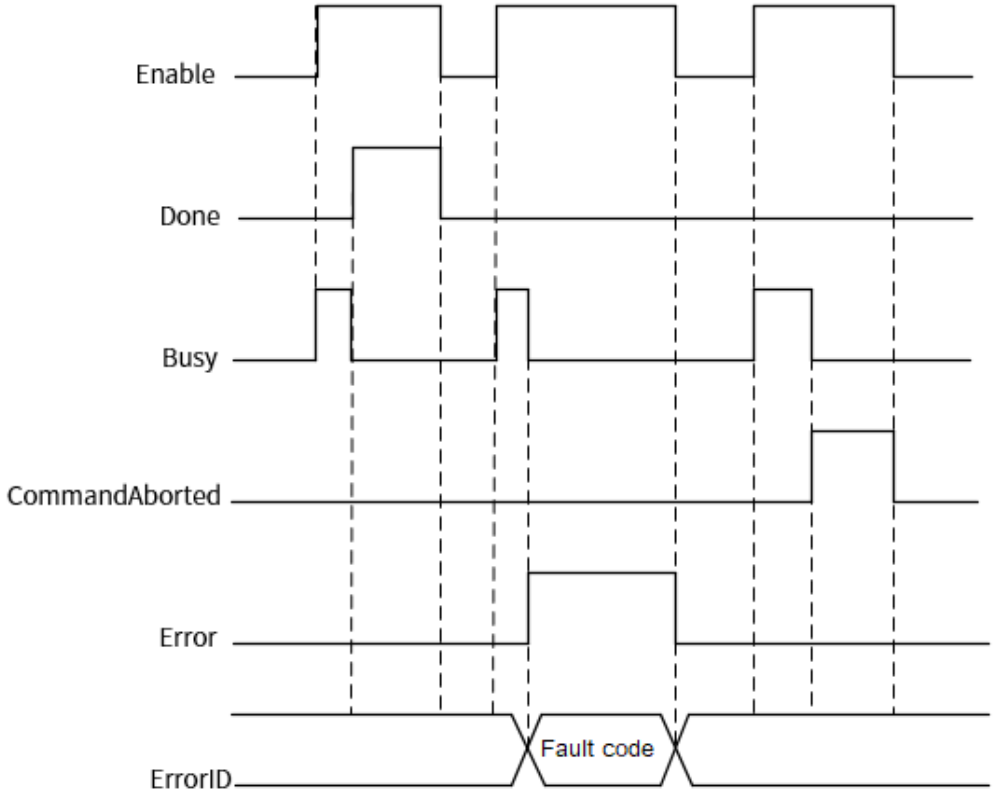
    The digital output terminal outputs high level after the local encoder axis reaches the comparison point. The high level will remain for the number of encoder pulses specified by the user. The number of comparison points is specified by |(StartPos-EndPosition)/Step|. The comparison points are generated by incrementing StartPos by Step until it does not exceed EndPosition.

Example of pulse mode: Set the comparison point array P[4], with a duration of 2Unit at each comparison point. The instruction starts running after Enable is enabled.



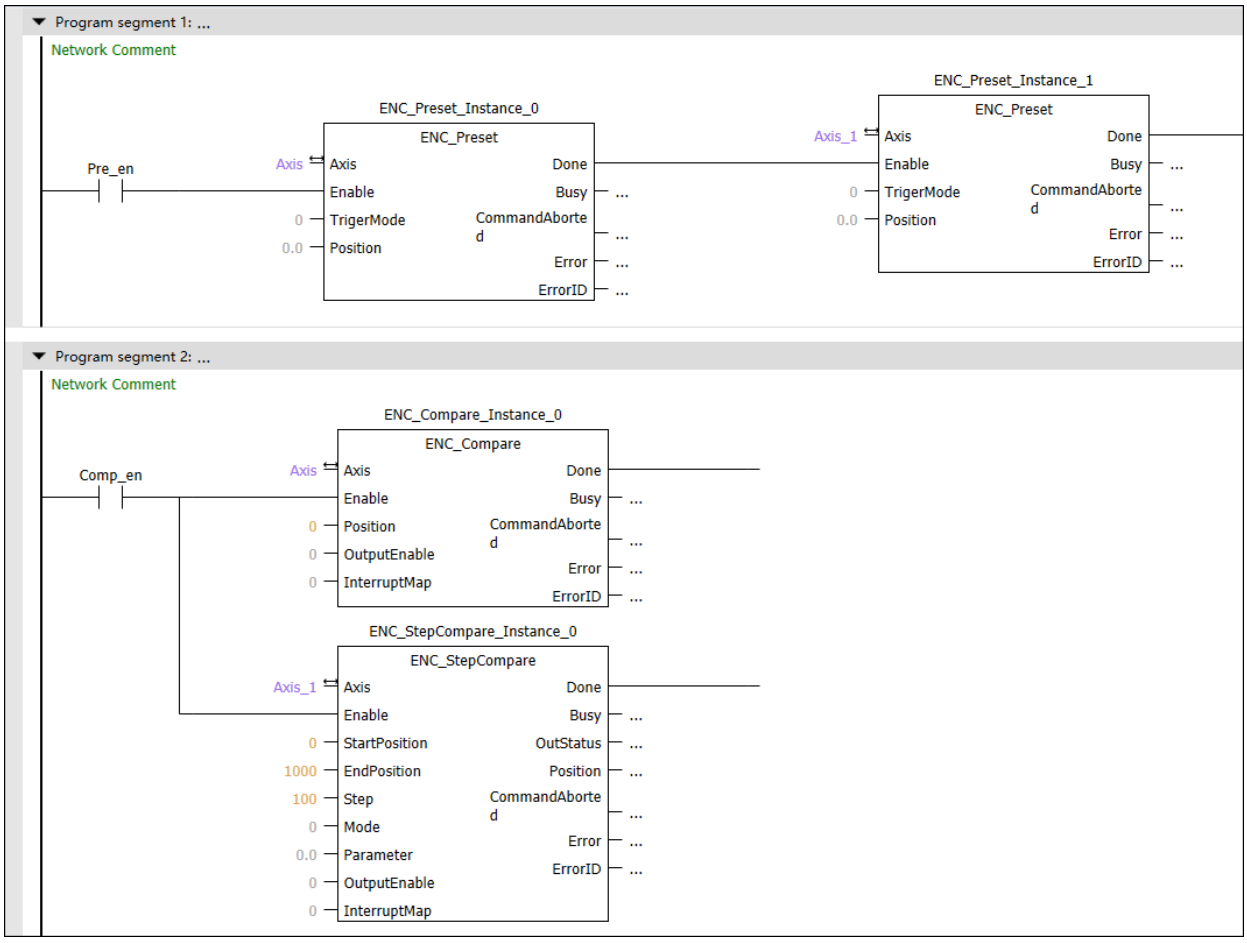
- If Enable is reset to FALSE during instruction execution, the remaining comparison points in ArrayCompare will not be compared.
- When OutputEnable is set to 0, the hardware terminal does not generate a comparison output signal. When OutputEnable is set to a non-zero value, the hardware terminal generates a comparison output signal.

- **Sequence diagram**



- **Precautions**

  When using the comparison instruction and ENC_Preset instruction, pay attention to the order of the instructions in the project configuration. The following situations will prevent the comparison instruction from generating an output.



In the above project configuration, after ENC_Compare_Instance_0 and ENC_StepCompare_Instance_0 are executed for the first time, that is, when ENC_Compare_Instance_0.Done and ENC_StepCompare_Instance_0.Done are both TRUE, the comparison instruction will not generate an output if you reset Pre_en and Comp_en after triggering Pre_en and Comp_en simultaneously. That is, when the encoder axis is counting, if ENC_Preset_Instance_0 and ENC_Preset_Instance_1 are all triggered simultaneously, the comparison instruction is activated only after the axis position has already passed comparison point 0. Therefore, the comparison instruction assumes that the encoder axis has never passed comparison point 0, and thus does not generate any corresponding output.

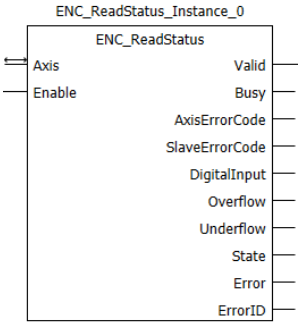
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.6.8 ENC_ReadStatus

Obtain the encoder status.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
ENC_ReadStatus	Encoder state read		ENC_ReadStatus_Instance_0(Axis :=, Enable :=, Valid =>, Busy => , AxisErrorCode => , SlaveErrorCode => , DigitalInput => , Overflow => , Underflow => , State => , Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

  Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: The instruction is enabled. FALSE: The instruction is not enabled.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Active	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the read status is valid.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is received.
AxisErrorCode	Axis fault code	INT	-	0	Axis error code.
SlaveErrorCode	Driver error code	INT	-	0	Driver error code.
DigitalInput	DI terminal state	INT	-	0	Bit0: CHn-X0 terminal state. Bit1: CHn-X1 terminal state. Bit2: CHn-X2 terminal state. Bit3: CHn-X3 terminal state. Others: Reserved.
Overflow	Count overflow flag	BOOL	TRUE, FALSE	FALSE	Counter overflow flag.
Underflow	Count underflow flag	BOOL	TRUE, FALSE	FALSE	Counter underflow flag.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
State	Counter enable status	18.3.3.2.2.5 MC_AXIS_STATE: Axis State Machine Enumeration	0: AXIS_POWER_OFF 1: AXIS_ERRORSTOP 2: AXIS_STOPPING 3: AXIS_STANDSTILL 4: AXIS_DISCRETE_MOTION 5: AXIS_CONTINUOUS_MOTION 6: AXIS_SYNCHRONIZED_MOTION 7: AXIS_HOMING	0	Axis state. 0: Axis power off state. 1: Axis error stop state. 2: Axis decelerate-to-stop state. 3: Axis standstill state. 4: Axis discrete motion state. 5: Axis continuous motion state. 6: Axis synchronized motion state. 7: Axis homing state.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus encoder axis:

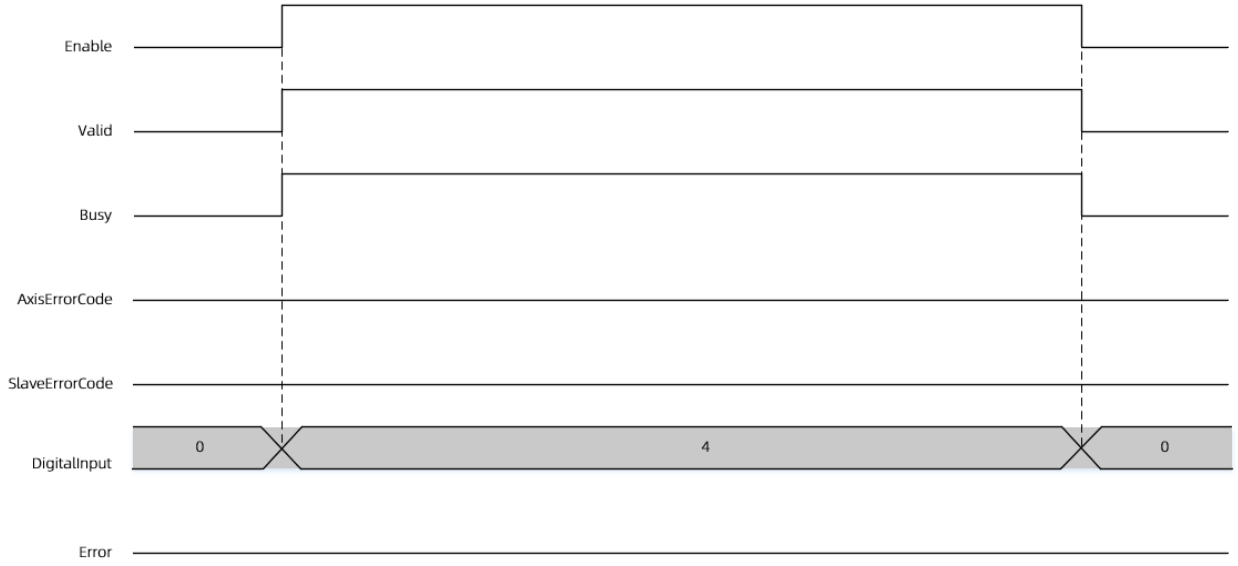
  - This instruction is used to read the status of the bus encoder axis.
  - The output parameter AxisErrorCode displays the axis error of the bus encoder axis, while the output parameter SlaveErrorCode displays the fault code of the GR10_2HCE module.
  - The output parameter DigitalInput indicates the status of the digital input terminals on the GR10_2HCE module. When the axis is bound to channel CH0, it shows the status of terminals X00, X01, X02, and X03. When the axis is bound to channel CH1, it shows the status of terminals X10, X11, X12, and X13.

  Local encoder axis:

  This instruction can be used to read the counter overflow/underflow flags and the counter status of the local encoder axes.

- **Sequence diagram**

When the input variable "Enable" is on the rising edge, it reads the status of the bus encoder axis. The output variable "Valid" indicates that the read status of the instruction is valid, and the output variable "Busy" indicates that the current function block is executing.



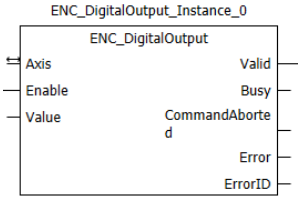
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.6.9 ENC_DigitalOutput

This instruction controls the encoder digital output.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
ENC_DigitalOutput	Encoder digital output control.		ENC_DigitalOutput_Instance_0(Axis :=, Enable :=, Value :=, Valid => , Busy => , CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: The instruction is enabled. FALSE: The instruction is not enabled.
Value	Set value for the digital output terminal	INT	-	0	Bit0: CHn-Y0 terminal status. Bit1: CHn-Y1 terminal status. Bit2: CHn-Y2 terminal status. Others: Reserved.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Active	BOOL	TRUE, FALSE	FALSE	TRUE when the output terminal status is set to valid.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus encoder axis:

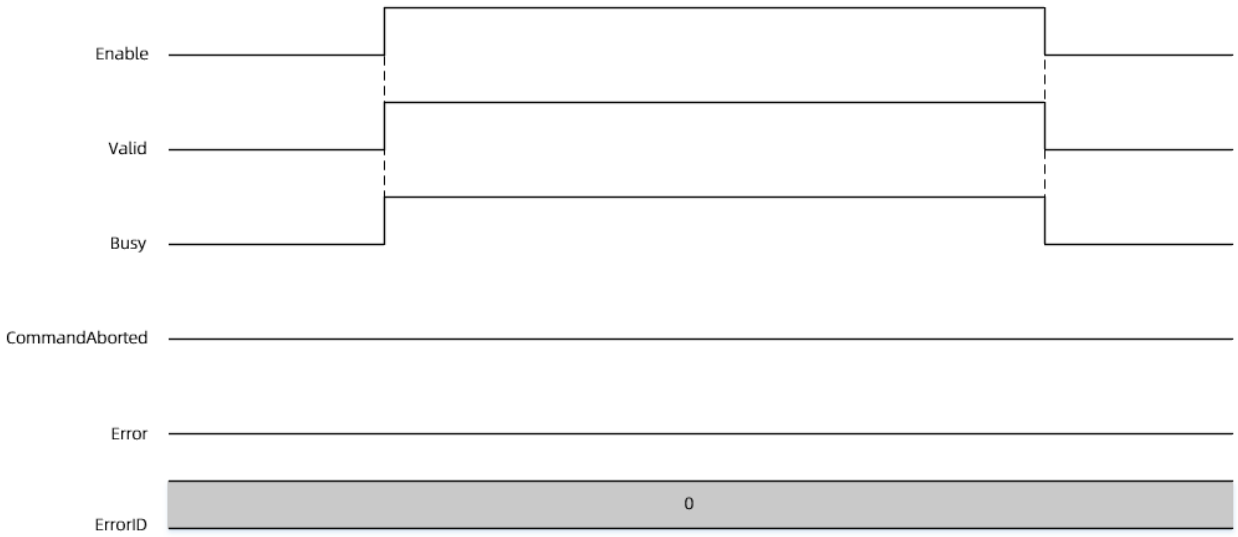
  - This instruction is used to set the DO terminal state of the GR10-2HCE module.
  - After the input parameter Enable is triggered, modifying the input parameter Value takes effect in real time.
  - During the execution of the instruction, if another instance of the function block for this instruction is triggered, the later-triggered instruction will return an error. If multiple instances of the function block for this instruction are triggered simultaneously, the first triggered function block will take effect, and the subsequent function blocks will return an error.
  - When the axis is bound to CH0 channel, it is used to set the status of terminals Y00, Y01, and Y02. When the axis is bound to CH1 channel, it is used to set the status of terminals Y10, Y11, and Y12.
  - During the instruction execution, if the Enable input is reset to FALSE, the value of the Value input will no longer be effective, and the corresponding terminal states will also be reset.

  Local encoder axis:

  This instruction does not support local encoder axes.

- **Sequence diagram**

  When the input variable "Enable" is on the rising edge, the status of the digital output terminal of the GR10_2HCE module is set. The output variable "Valid" indicates that the set status of the output terminal is valid, and the output variable "Busy" indicates that the current function block is executing.



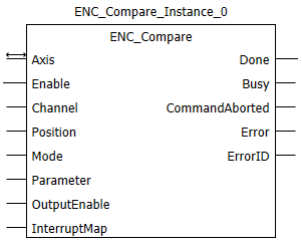
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.6.10 ENC_Compare

This instruction outputs a single-point comparison.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
ENC_Compare	Encoder single-point comparison output		ENC_Compare_Instance_0(Axis :=, Enable :=, Channel := , Position := , Mode := , Parameter := , OutputEnable := , InterruptMap := , Done => , Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: The instruction is enabled. FALSE: The instruction is not enabled.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Channel	Channel	BYTE	1~2	1	Comparison channel, valid only for bus encoder axes.
Position	Comparison position	LREAL	Positive number	0	Comparison position.
Mode	Comparison mode	INT	0 to 3	0	Valid only for local encoder axes. The bus encoder axes only support time control. The Mode parameter is invalid. 0: Reserved. 1: Time control 2: Pulse control 3: Level control
Parameter	Control parameters	LREAL	-	0	The local encoder axis level control is invalid. The bus encoder axes only support time control. Time control: output active duration, in 100 μs. Pulse control: number of output valid pulses, in Unit. Level control: initial level; 0 indicates low level and non-zero indicates high level.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
OutputEnable	Hardware output enable	INT	-	0	Valid only for the local encoder axis. 0: Disabled Non-zero: Enabled
InterruptMap	Interrupt number	INT	0 to 16	0	Valid only for the local encoder axis. 0: No interrupt is generated. 1: Mapped to external event Cmp1InterruptEvent. 2: Mapped to external event Cmp2InterruptEvent. ... 16: Mapped to external event Cmp16InterruptEvent.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus encoder axis:

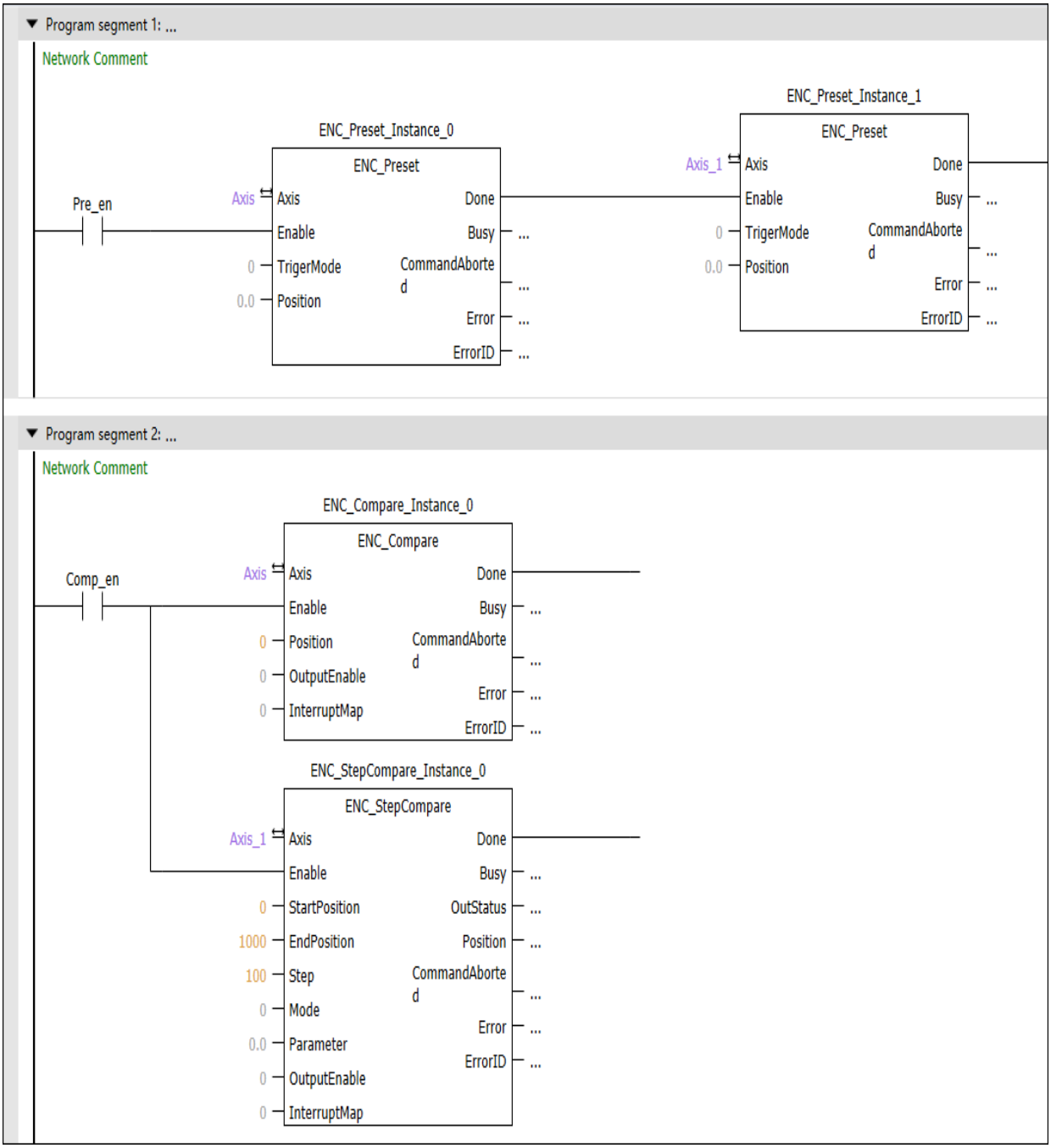
  - Supports only comparison function of the GR10-2HCE module.
  - After Enable is triggered, changes to input parameters have no effect. The instruction must be re-triggered for changes to take effect.
  - After the instruction is reset, the hardware output port will not be affected.
  - Configure the corresponding PDO before using the instruction. Otherwise, the instruction will report an error:
    Channel 1 corresponds to PDO: 7003h, 7004h, 7006h (The comparison output function needs to be configured.), 600Eh.
    Channel 2 corresponds to PDO: 7003h, 7005h, 7007h (The comparison output function needs to be configured.), 600Eh.
  - For the comparison output, you must configure the corresponding port as the comparison output port in the background. For configuration details, refer to Counter Settings > External output.
  - Set the comparison output time via the Parameter input of ENC_Compare. Only the comparison output time can be set on the 2HCE module.

  Local encoder axis:

  - After Enable is triggered, changes to input parameters have no effect. The instruction must be re-triggered for changes to take effect.
  - After the instruction is reset, the hardware output port will not be affected.
  - In the comparison output setting interface for the local encoder axis, enable Comparison Output Enable, select the comparison output terminal, and choose whether to output based on time or user units of pulses. When time mode is selected, the parameter range is 0.1 to 6553.5 ms; when pulse mode is selected, the parameter range is 1 to 65535 units.



  - After the instruction is enabled, when the encoder axis reaches the position specified by Position, the comparison output becomes active. If OutputEnable is set to 1, the configured comparison output terminal generates a comparison output signal. If OutputEnable is set to 0, no comparison output signal is generated.
  - If the "Comparison output enable" option is not enabled or if the "Comparison output enable" option is enabled while no output terminal is assigned, the instruction will return an error.
  - When using the comparison instruction and ENC_Preset instruction, pay attention to the order of the instructions in the project configuration. The following situations will prevent the comparison instruction from generating an output.

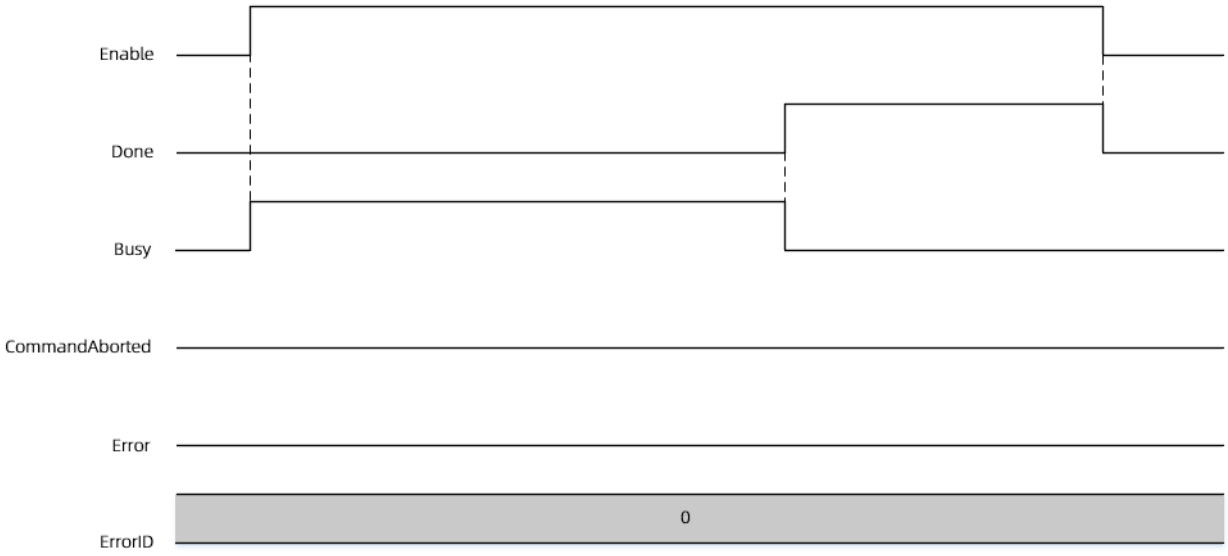


In the above project configuration, after ENC_Compare_Instance_0 and ENC_StepCompare_Instance_0 are executed for the first time, that is, when ENC_Compare_Instance_0.Done and ENC_StepCompare_Instance_0.Done are both TRUE, the comparison instruction will not generate an output if you reset Pre_en and Comp_en after triggering Pre_en and Comp_en simultaneously. That is, when the encoder axis is counting, if ENC_Preset_Instance_0 and ENC_Preset_Instance_1 are all triggered simultaneously, the comparison instruction is activated only after the axis position has already passed comparison point 0. Therefore, the comparison instruction assumes that the encoder axis has never passed comparison point 0, and thus does not generate any corresponding output.

- **Sequence diagram**

  After the ENC_Compare instruction is enabled, the output parameter "Busy" is set to TRUE when the input parameter "Enable" is set to TRUE.

  When the encoder axis reaches the position specified by Position, the output parameter "Done" is set to TRUE.



- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.6.11 ENC_SetUnit

This instruction sets the gear ratio.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
ENC_SetUnit	Set the gear ratio	ENC_SetUnit_Instance_0 ENC_SetUnit Axis Enable PlusePerCycle DisPerCycle Numerator Denominator Done Busy CommandAborted Error ErrorID	ENC_SetUnit_Instance_0(Axis :=, Enable :=, PlusePerCycle := , DisPerCycle := , Numerator := , Denominator := , Done => , Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

  In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: The instruction is enabled. FALSE: The instruction is not enabled.
PlusePerCycle	Number of encoder pulses per revolution	DINT	Positive values	0	Number of pulses per revolution of the encoder.
DisPerCycle	Distance per revolution of the worktable	LREAL	-	0	Distance traveled by the worktable per revolution.
Numerator	Gear ratio numerator	LREAL	Positive number	0	Numerator of the gear ratio.
Denotinator	Gear ratio denominator	DWORD	Positive values	-	Denominator of the gear ratio.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is received.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus encoder axis:

  This instruction does not support bus encoder axes.

  Local encoder axis:

  - This instruction is used to set the gear ratio of the local encoder axis via input parameters.
  - After the Enable input parameter is triggered, changes to input parameters have no effect until Enable is reset and the instruction is re-triggered.
  - The input parameter PlusePerCycle must not be less than or equal to 0. The input parameter DisPerCycle must not be less than or equal to 0.
  - The input parameters Numerator and Denominator cannot be set to 0.
  - The gear ratio of the local encoder axis is calculated as follows:

$$\text{Gear ratio} = \frac{\text{PlusePreCycle} * \text{Numerator}}{\text{DisPreCycle} * \text{Denotinator}}$$

  - If this instruction is triggered during program execution, the gear ratio of the local encoder axis changes immediately, resulting in an abrupt change in the feedback position or velocity of local encoder axis.
  - For a linear axis with the specified soft limit, the upper and lower limit values will be recalculated after the gear ratio is set. For a rotary axis, the rotation cycle will be recalculated after the gear ratio is set.

- **Sequence diagram**

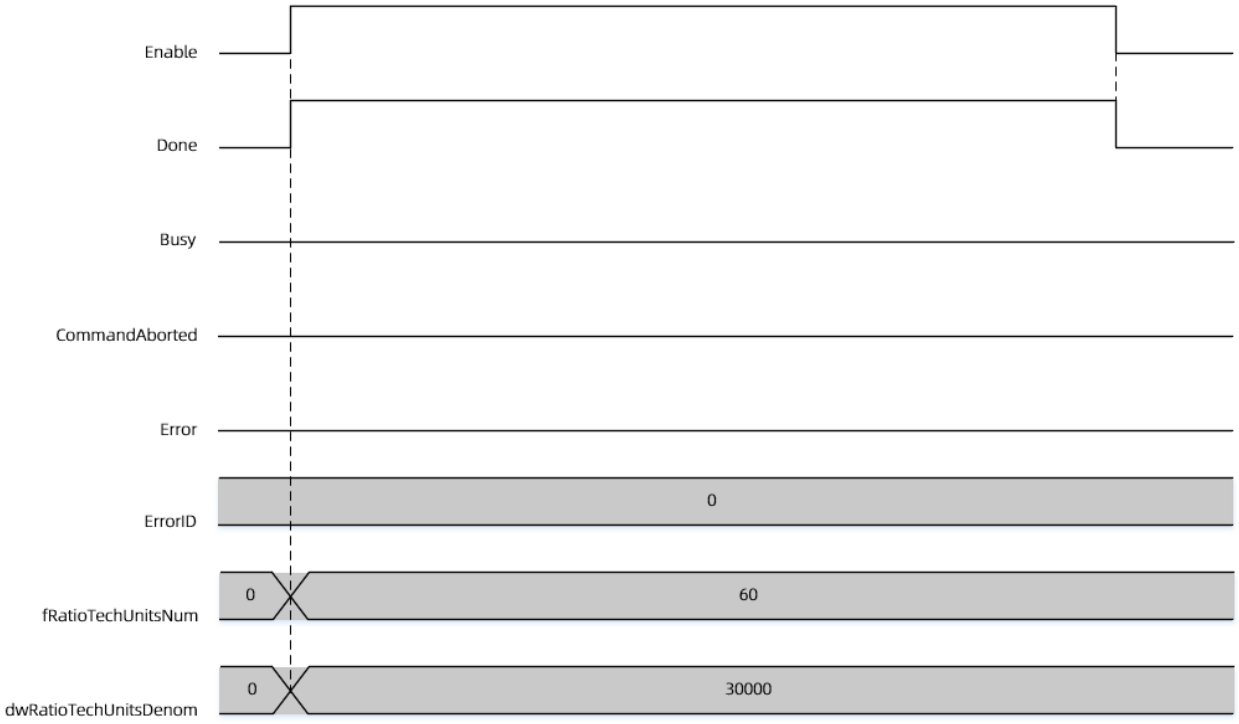
  This example describes how to set the local encoder axis parameters as shown in the following table automatically after power-on of the PLC.

Description	Value
Number of encoder pulses per revolution	10000
Distance per revolution of the worktable	30
Gear ratio numerator	3
Gear ratio denominator	2

  When the input parameter "Enable" is set to TRUE, the output parameter "Busy" is set to TRUE. After the gear ratio has been set, the output parameter "Done" is set to TRUE.

  The axis structure variable "fRatioTechUnitsNum" is the gear ratio numerator, and the axis structure variable "dwRatioTechUnitsDenom" is the gear ratio denominator. After modifying the parameters in

the above table using the ENC_SetUnit instruction, the value of variable "fRatioTechUnitsNum" becomes "60", and the value of variable "dwRatioTechUnitsDenom" becomes "30000".



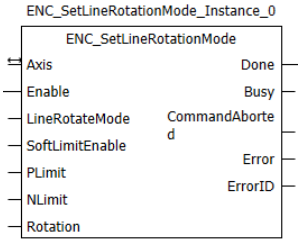
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.6.12 ENC_SetLineRotationMode

This instruction is used to set the axis operation mode.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
ENC_SetLineRotationMode	This instruction is used to set the axis operation mode.		ENC_SetLineRotationMode_Instance_0(Axis :=, Enable :=, LineRotateMode := , SoftLimitEnable := , PLimit := , NLimit := , Rotation := , Done => , Busy =>, CommandAborted =>, Error =>, ErrorID =>);

- **Related variables**

In-out variables

In-out Variable	Name	Data Type	Valid Range	Default Value	Description
Axis	Axis	18.3.3.2.2.1 AXIS_REF: Axis Structure	-	-	Reference to the axis; an instance of AXIS_REF.

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: The instruction is enabled. FALSE: The instruction is not enabled.
LineRotateMode	Linear/Rotation mode: 0: Linear mode. 1: Rotation mode	INT	0 to 1	0	Linear/Rotation mode: 0: Linear mode. 1: Rotation mode.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
SoftLimitEnable	Limit enable in linear mode: ON: Enabled. OFF: Disabled.	BOOL	-	0	Limit enable in linear mode: ON: Enabled. OFF: Disabled.
PLimit	Positive limit value in linear mode	LREAL	-	0	Positive limit value in linear mode.
NLimit	Negative limit value in linear mode	LREAL	-	-	Negative limit value in linear mode.
Rotation	Rotation cycle in rotation mode.	LREAL	-	-	Rotation cycle in rotation mode.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is done.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when the instruction is received.
CommandAborted	Command aborted	BOOL	TRUE, FALSE	FALSE	TRUE when the current instruction is aborted.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	DWORD	_	0	A fault code (18.3.3.8 Fault Codes) is output when a fault occurs.

- **Function**

  Bus encoder axis:

  This instruction does not support bus encoder axes.

  Local encoder axis:

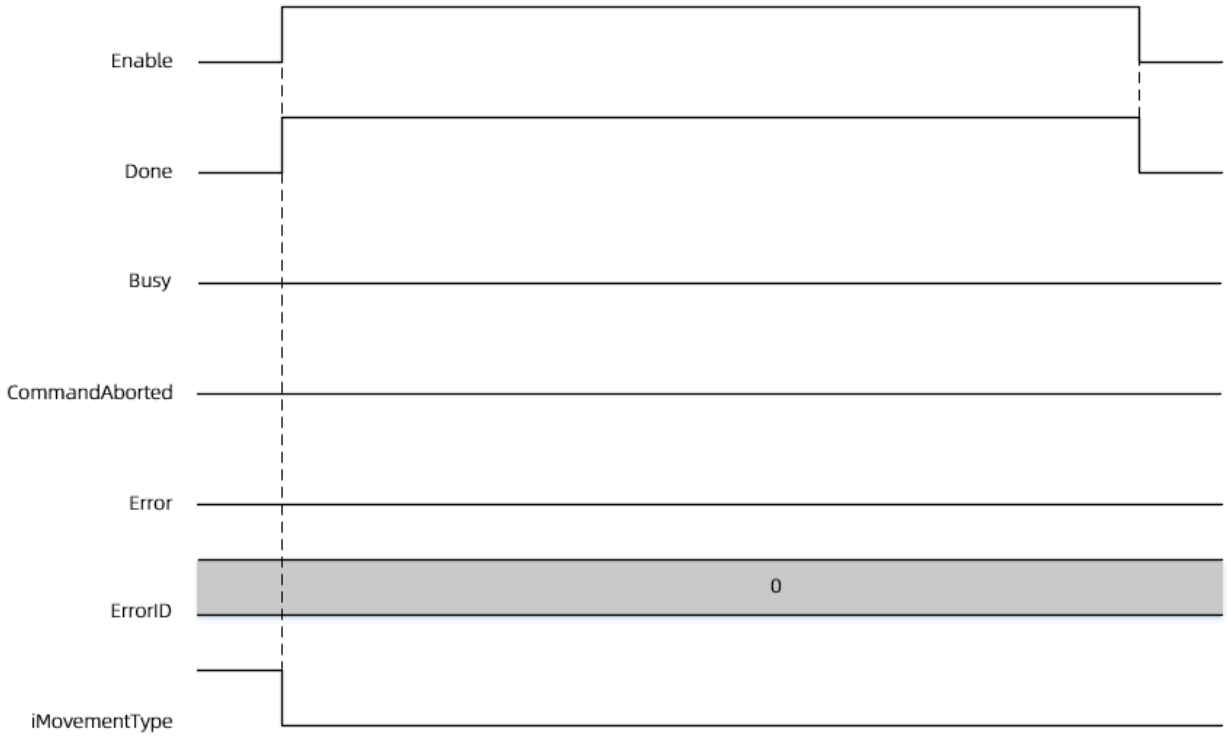
  ◦ This instruction is used to set the operation mode of the local encoder axis. It includes linear mode and rotation mode.

- After the Enable input parameter is triggered, changes to input parameters have no effect until Enable is reset and the instruction is re-triggered.
- When the LineRotateMode input parameter is set to 0, the local encoder axis is in linear mode.
- In linear mode, the input parameter SoftLimitEnable is set to OFF, indicating that soft limits are disabled.
- SoftLimitEnable = ON indicates that soft limits are enabled. In this case, the input parameter PLimit represents the positive soft limit value, and NLimit represents the negative soft limit value. The input parameter PLimit must not be less than or equal to NLimit. PLimit must be greater than 0, and NLimit must be less than 0. If the pulse count corresponding to PLimit exceeds 2147483647 or the pulse count corresponding to NLimit is less than -2147483648, the instruction reports an error.
- When LineRotateMode is set to 1, the local encoder axis is in rotation mode. In this case, the Rotation input parameter represents the rotation cycle, which must not be 0.
- This instruction can only be used in the POWER_OFF status. If the ENC_Counter instruction is running, triggering the instruction will cause an error.

- **Sequence diagram**

  When the input parameter "Enable" is set to TRUE, the output parameter "Busy" is set to TRUE. After the axis operation mode has been configured, the output parameter "Done" is set to TRUE.

  When the input parameter "LineRotateMode" is set to "1", the axis is set to rotation mode. The axis structure variable "iMovementType" indicates the axis type. (0: periodic rotation mode, 1: linear mode)



- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.7 Drive

18.3.3.7.1 Instruction List

Instruction	Function	Library	Applicable Library Version	Supported Axis Type
18.3.3.7.2 MC_BackupServoParameter	Backup servo parameter	MotionControl	Library version 1.1.0.0 and later	Bus servo axis
18.3.3.7.3 MC_RestoreServoParameter	Restore servo parameters		Library version 1.1.0.0 and later	Bus servo axis
18.3.3.7.4 MC_BackupServoParameter2	Backup servo parameter		Library version 2.1.0.0 and later	Bus servo axis
18.3.3.7.5 MC_RestoreServoParameter2	Restore servo parameters		Library version 2.1.0.0 and later	Bus servo axis

18.3.3.7.2 MC_BackupServoParameter

This instruction can obtain the corresponding servo parameter values from the servo drive according to the servo parameter template, and save the values as a binary file to the SD card.

> NOTE
>
> This instruction is only applicable to a part of servo drive models. It is recommended to use the 18.3.3.7.4 MC_BackupServoParameter2 instruction, which has been adapted for more device models.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_BackupServoParameter	Backup servo parameter	MC_BackupServoParameter_Instance_0 MC_BackupServoParameter Execute / Done SlaveID / Busy FileName / Error OverWrite / ErrorID	MC_BackupServoParameter_Instance_1(Execute := , SlaveID := , FileName := , OverWrite := , Done => , Busy =>, Error =>, ErrorID =>);

- **Related variables**

  Input variables

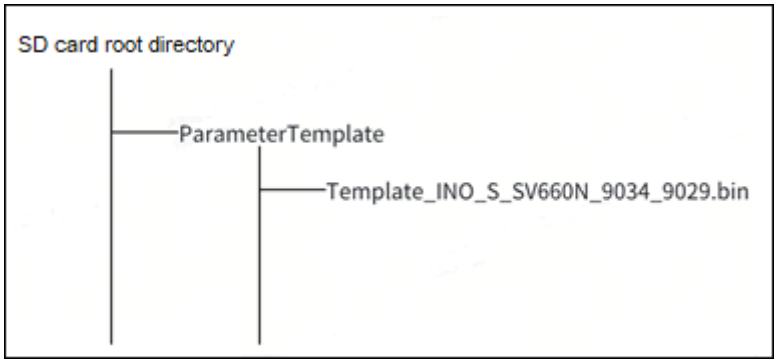
Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered execute enable flag
SlaveID	ID number of the slave address	DINT	[1,65535]	1	The specified slave address where the servo parameters need to be backed up.
FileName	File name	STRING	Maximum 255 characters	''	The name of the file saving the servo parameters on the SD card.
OverWrite	File overwrite mode	BOOL	TRUE, FALSE	FALSE	TRUE: Enable overwrite.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE indicates that the servo parameters for the servo drive have been backed up.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE indicates that the servo parameters for the servo drive are being backed up.
Error	Error occurred	BOOL	TRUE, FALSE	FALSE	TRUE indicates that an error has occurred during the servo parameter backup.
ErrorID	Fault code	DWORD	-	0	Fault code (18.3.3.8 Fault Codes) recorded when a fault occurred.

- **Function**
  - On the rising edge of "Execute", the servo parameter backup process is initiated. According to the servo parameter backup template, the instruction retrieves the corresponding parameter values from the servo drive and stores them as a binary file in the root directory of the SD card. The file name is defined by the input variable FileName.

- The servo parameter backup template must be stored in the ParameterTemplate folder in the root directory of the SD card. If the ParameterTemplate folder does not exist, the user must create it manually.

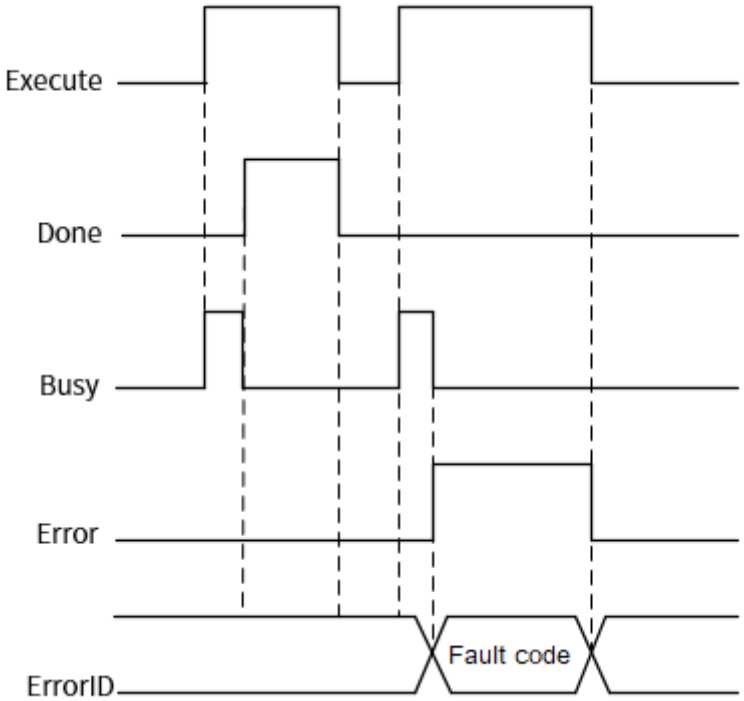


- The servo parameter backup template contains the following information: vendor ID, device type, customized number, MCU software version, FPGA software version, servo parameter main index, servo parameter sub-index, and servo parameter length. The file is binary with a .bin extension.
- The file name of the servo parameter backup template must not be changed arbitrarily.
- For different servo types or different versions of the same servo type, the corresponding servo parameter backup templates may vary. Therefore, before use, verify that the servo parameter backup template matches the actual servo in terms of device type, MCU software version, and FPGA software version.
- For a given servo type and version, only one corresponding servo parameter backup template is allowed in the ParameterTemplate folder on the SD card.
- If the user does not set the FileName parameter for the servo parameter backup file, the default file name ServoParameter is used.
- The user-configured name FileName for the servo parameter backup file can only contain numbers, letters, and underscores.
- If a file with the same name as the input variable FileName exists in the root directory of the SD card, and the OverWrite mode is set to FALSE, the function block will report an error.
- If the servo parameter backup file specified by FileName on the SD card is open for writing or reading when the function block is triggered, the function block will report an error.
- When Execute is set, the servo parameter backup function is activated. After Execute is reset, the servo parameter backup function continues until all servo parameters in the servo parameter backup template are backed up or an error occurs.
- Before using this function block, ensure that the SD card is inserted into the controller and that EtherCAT communication is in the OP status.
- After inserting or removing the SD card, the controller must be power-cycled before executing this function block.
- Multiple instances of the servo parameter backup function block are not permitted to operate simultaneously on the same servo parameter backup file within the SD card.
- When a servo parameter restoration function block instance is operating on a servo parameter backup file on the SD card, the servo parameter backup function block is not permitted to perform operations on the same file simultaneously.
- If multiple servo parameter backup instructions MC_BackupServoParameter are executed simultaneously for the same slave, or if the servo parameter backup instruction

MC_BackupServoParameter and the restoration instruction MC_RestoreServoParameter are executed at the same time, one of these instructions will return an error.

- Before triggering the function block, ensure that the program is running and the slave is in the OP status.

- **Sequence diagram**



- **Precautions**
  - Only Inovance EtherCAT servo slaves are supported. (Currently, only the SV660N standard series servos are supported. Their MCU and FPGA software versions must be 903.4 and 902.9, respectively.)
  - Do not execute two identical MC_BackupServoParameter instances in the same project.
  - It is recommended to execute this function block in a non-real-time task.
- **Restarting the motion instruction**

  This function block does not support re-triggering.
- **Faults**

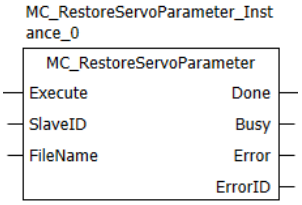
  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.7.3 MC_RestoreServoParameter

This instruction reads the servo parameters backed up on the SD card and sends them to the servo drive via SDO communication.

> **NOTE**
>
> This instruction is only applicable to a part of servo drive models. It is recommended to use the 18.3.3.7.5 MC_RestoreServoParameter2 instruction, which has been adapted for more device models.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_RestoreServoParameter	Restore servo parameters		MC_RestoreServoParameter_Instance_0(Execute := , SlaveID := , FileName := , Done =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered execute enable flag
SlaveID	ID number of the slave address	DINT	[1,65535]	1	The specified slave address where the servo parameters need to be restored.
FileName	File name	STRING	Maximum 255 characters	''	The name of the file containing the servo parameters on the SD card.

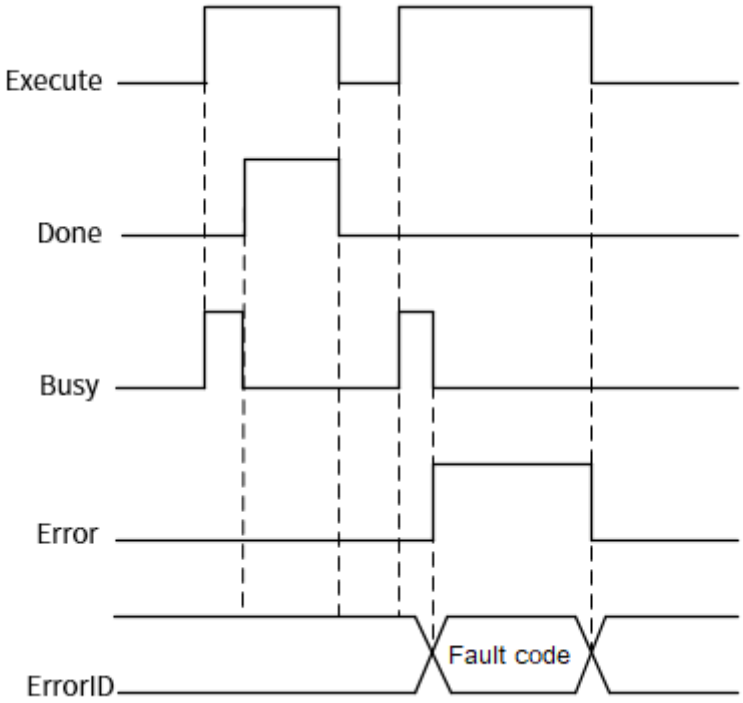
Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE indicates that the servo parameters for the servo driver have been configured.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE indicates that the servo parameters for the servo drive are being configured.
Error	Error occurred	BOOL	TRUE, FALSE	FALSE	TRUE indicates that an error has occurred during the servo parameter configuration.
ErrorID	Fault code	DWORD	-	0	Fault code (18.3.3.8 Fault Codes) recorded when a fault occurred.

- **Function**
  - On the rising edge of the "Execute", the servo parameter restoration starts. The instruction retrieves the corresponding servo parameter values from the specified servo parameter backup file and writes them into the servo slave. The name of the servo parameter backup file is specified by the input variable FileName.
  - The servo parameter backup file is stored in the root directory of the SD card. It contains the following information: vendor ID, device type, customized number, MCU software version, FPGA software version, servo parameter main index, servo parameter sub-index, servo parameter length, and the corresponding servo parameter values. The file format is binary with a .bin extension.
  - If the user has not configured the file name FileName for the servo parameter backup file, the function block will report an error.
  - The user-configured name FileName for the servo parameter backup file can only contain numbers, letters, and underscores.
  - If the file specified in the input variable FileName does not exist in the SD card root directory, the function block will report an error.
  - When Execute is set, the servo parameter restoration function is activated. After Execute is reset, the servo parameter restoration function continues until all servo parameters in the backup file are set or an error occurs.

- Before using this function block, ensure that the SD card is inserted into the controller and that EtherCAT communication is in the OP status.
- After inserting or removing the SD card, the controller must be power-cycled before executing this function block.
- Before using this function block, the servo must be in one of the following statuses: "No fault", "Servo ready", "Waiting for the S-ON signal", or "Fault".
- If the slave requiring servo parameter restoration is assigned to an axis, the MC_Power instruction must not be triggered to enable the axis before using this function block.
- If the servo parameter backup file specified by FileName on the SD card is open for writing when the function block is triggered, the function block will report an error.
- If the servo parameter file does not match the vendor ID, device type, customized number, MCU software version or FPGA software version of the servo drive, the function block will report an error.
- When a servo parameter backup function block instance is operating on a servo parameter restoration file on the SD card, the servo parameter restoration function block is not permitted to perform operations on the same file simultaneously.
- If multiple servo parameter restoration instructions MC_RestoreServoParameter are executed simultaneously for the same slave, or if the servo parameter backup instruction MC_BackupServoParameter and the restoration instruction MC_RestoreServoParameter are executed at the same time, one of these instructions will return an error.
- Before triggering the function block, ensure that the program is running and the slave is in the OP status.

- **Sequence diagram**



- **Precautions**
  - Only Inovance EtherCAT servo slaves are supported. (Currently, only the SV660N standard series servos are supported. Their MCU and FPGA software versions must be 903.4 and 902.9, respectively.)
  - It is recommended to execute this function block in a non-real-time task.

- Do not execute two identical instances of MC_RestoreServoParameter in the same project.
- After the function block has been executed successfully, the servo must be power-cycled for all written servo parameters to take full effect.

- **Restarting the motion instruction**

  This function block does not support re-triggering.

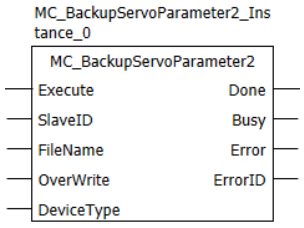
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.7.4 MC_BackupServoParameter2

This instruction can obtain the corresponding servo parameter values from the servo drive according to the servo parameter template, and save the values as a binary file to the SD card.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_BackupServoParameter2	Backup servo parameter		MC_BackupServoParameter2_Instance_0(Execute := , SlaveID := , FileName := , OverWrite := , DeviceType := , Done => , Busy => , Error => , ErrorID =>);

- **Related variables**

  Input variables

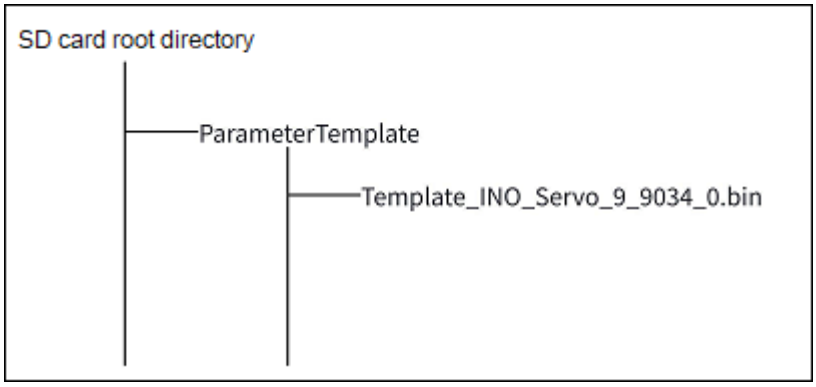
Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered execute enable flag
SlaveID	ID number of the slave address	DINT	[1,65535]	1	The specified slave address where the servo parameters need to be backed up.
FileName	File name	STRING	Maximum 255 characters	''	The name of the file saving the servo parameters on the SD card.
OverWrite	File overwrite mode	BOOL	TRUE, FALSE	FALSE	TRUE: Enable overwrite.

Input Variable	Name	Data Type	Valid Range	Default Value	Description
DeviceType	Device type	INT	-	0	Type of the connected device. Reserved parameter.

Output variable

Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE indicates that the servo parameters for the servo drive have been backed up.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE indicates that the servo parameters for the servo drive are being backed up.
Error	Error occurred	BOOL	TRUE, FALSE	FALSE	TRUE indicates that an error has occurred during the servo parameter backup.
ErrorID	Fault code	DWORD	-	0	Fault code (18.3.3.8 Fault Codes) recorded when a fault occurred.

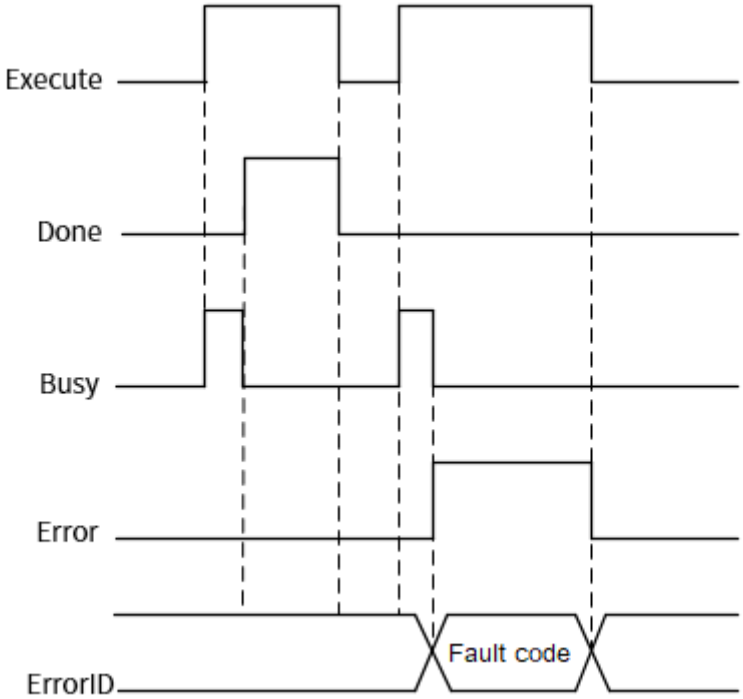
- **Function**
  - On the rising edge of "Execute", the servo parameter backup process is initiated. According to the servo parameter backup template, the instruction retrieves the corresponding parameter values from the servo drive and stores them as a binary file in the root directory of the SD card. The file name is defined by the input variable FileName.
  - The servo parameter backup template must be stored in the ParameterTemplate folder in the root directory of the SD card. If the ParameterTemplate folder does not exist, the user must create it manually.



  - The servo parameter backup template contains the following information: vendor ID, device type, customized number, MCU software version, servo parameter main index, servo parameter sub-index, and servo parameter length. The file is binary with a .bin extension.
  - The file name of the servo parameter backup template must not be changed arbitrarily.

- For different servo types or different versions of the same servo type, the corresponding servo parameter backup templates may vary. Therefore, before use, verify that the servo parameter backup template matches the actual servo in terms of device type, and MCU software version.
- For a given servo type and version, only one corresponding servo parameter backup template is allowed in the ParameterTemplate folder on the SD card.
- If the user does not set the FileName parameter for the servo parameter backup file, the default file name ServoParameter is used.
- The user-configured name FileName for the servo parameter backup file can only contain numbers, letters, and underscores.
- If a file with the same name as the input variable FileName exists in the root directory of the SD card, and the OverWrite mode is set to FALSE, the function block will report an error.
- If the servo parameter backup file specified by FileName on the SD card is open for writing or reading when the function block is triggered, the function block will report an error.
- When Execute is set, the servo parameter backup function is activated. After Execute is reset, the servo parameter backup function continues until all servo parameters in the servo parameter backup template are backed up or an error occurs.
- Before using this function block, ensure that the SD card is inserted into the controller and that EtherCAT communication is in the OP status.
- After inserting or removing the SD card, the controller must be power-cycled before executing this function block.
- Multiple instances of the servo parameter backup function block are not permitted to operate simultaneously on the same servo parameter backup file within the SD card.
- When a servo parameter restoration function block instance is operating on a servo parameter backup file on the SD card, the servo parameter backup function block is not permitted to perform operations on the same file simultaneously.
- If multiple servo parameter backup instructions MC_BackupServoParameter2 are executed simultaneously for the same slave, or if the servo parameter backup instruction MC_BackupServoParameter2 and the restoration instruction MC_RestoreServoParameter2 are executed at the same time, one of these instructions will return an error.
- Before triggering the function block, ensure that the program is running and the slave is in the OP status.

- **Sequence diagram**



- **Precautions**
  - Only Inovance EtherCAT servo slaves are supported.
  - Two identical MC_BackupServoParameter2 instances must not be executed within the same project.
  - It is recommended to execute this function block in a non-real-time task.
- **Restarting the motion instruction**

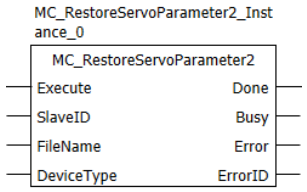
  This function block does not support re-triggering.
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.7.5 MC_RestoreServoParameter2

This instruction reads the servo parameters backed up on the SD card and sends them to the servo drive via SDO communication.

- **Instruction format**

Instruction	Name	LD Representation	ST Representation
MC_RestoreServoParameter2	Restore servo parameters		MC_RestoreServoParameter2_Instance_0(Execute := , SlaveID := , FileName := , DeviceType := , Done => , Busy => , Error => , ErrorID =>);

- **Related variables**

Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Execute	Execute	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered execute enable flag
SlaveID	ID number of the slave address	DINT	[1,65535]	1	The specified slave address where the servo parameters need to be restored.
FileName	File name	STRING	Maximum 255 characters	''	The name of the file containing the servo parameters on the SD card.
DeviceType	Device type	INT	-	0	Type of the connected device. Reserved parameter.

Output variable

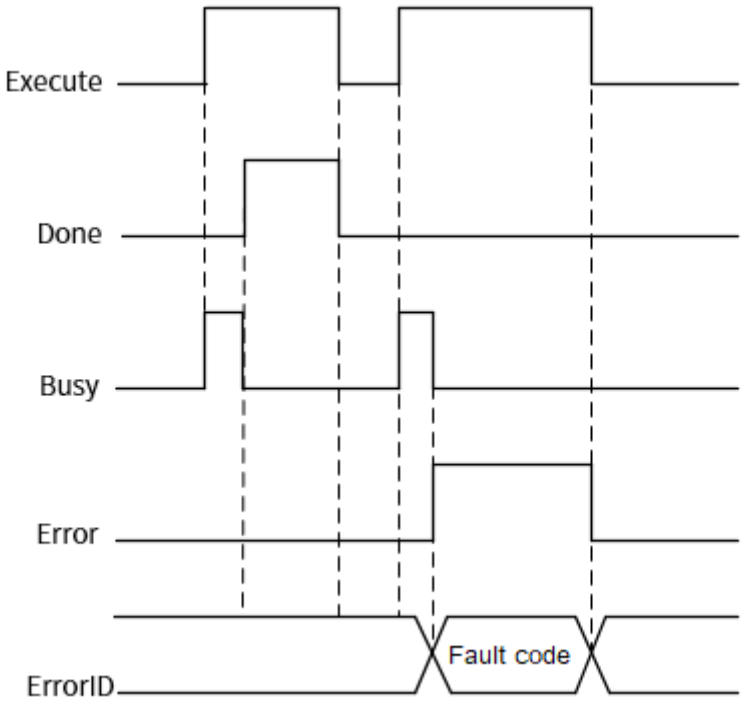
Output Variable	Name	Data Type	Valid Range	Default Value	Description
Done	Done	BOOL	TRUE, FALSE	FALSE	TRUE indicates that the servo parameters for the servo driver have been configured.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE indicates that the servo parameters for the servo drive are being configured.
Error	Error occurred	BOOL	TRUE, FALSE	FALSE	TRUE indicates that an error has occurred during the servo parameter configuration.

Output Variable	Name	Data Type	Valid Range	Default Value	Description
ErrorID	Fault code	DWORD	-	0	Fault code (18.3.3.8 Fault Codes) recorded when a fault occurred.

- **Function**
  - On the rising edge of the "Execute", the servo parameter restoration starts. The instruction retrieves the corresponding servo parameter values from the specified servo parameter backup file and writes them into the servo slave. The name of the servo parameter backup file is specified by the input variable FileName.
  - The servo parameter backup file is stored in the root directory of the SD card. It contains the following information: vendor ID, device type, customized number, MCU software version, servo parameter main index, servo parameter sub-index, servo parameter length, and the corresponding servo parameter values. The file format is binary with a .bin extension.
  - If the user has not configured the file name FileName for the servo parameter backup file, the function block will report an error.
  - The user-configured name FileName for the servo parameter backup file can only contain numbers, letters, and underscores.
  - If the file specified in the input variable FileName does not exist in the SD card root directory, the function block will report an error.
  - When Execute is set, the servo parameter restoration function is activated. After Execute is reset, the servo parameter restoration function continues until all servo parameters in the backup file are set or an error occurs.
  - Before using this function block, ensure that the SD card is inserted into the controller and that EtherCAT communication is in the OP status.
  - After inserting or removing the SD card, the controller must be power-cycled before executing this function block.
  - Before using this function block, the servo must be in one of the following statuses: "No fault", "Servo ready", "Waiting for the S-ON signal", or "Fault".
  - If the slave requiring servo parameter restoration is assigned to an axis, the MC_Power instruction must not be triggered to enable the axis before using this function block.
  - If the servo parameter backup file specified by FileName on the SD card is open for writing when the function block is triggered, the function block will report an error.
  - If the servo parameter file does not match the vendor ID, device type, customized number, or MCU software version of the servo drive, the function block will report an error.
  - When a servo parameter backup function block instance is operating on a servo parameter restoration file on the SD card, the servo parameter restoration function block is not permitted to perform operations on the same file simultaneously.
  - If multiple servo parameter restoration instructions MC_RestoreServoParameter2 are executed simultaneously for the same slave, or if the servo parameter backup instruction MC_BackupServoParameter2 and the restoration instruction MC_RestoreServoParameter2 are executed at the same time, one of these instructions will return an error.

- Before triggering the function block, ensure that the program is running and the slave is in the OP status.

- **Sequence diagram**



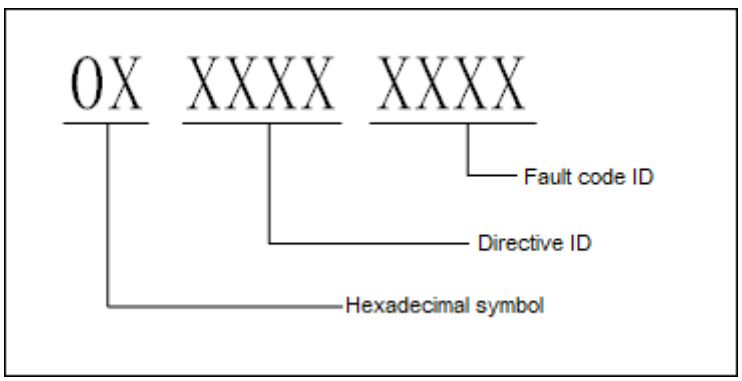
- **Precautions**
  - Only Inovance EtherCAT servo slaves are supported.
  - Two identical MC_RestoreServoParameter2 instances must not be executed within the same project.
  - It is recommended to execute this function block in a non-real-time task.
  - After the function block has been executed successfully, the servo must be power-cycled for all written servo parameters to take full effect.
- **Restarting the motion instruction**

  This function block does not support re-triggering.
- **Faults**

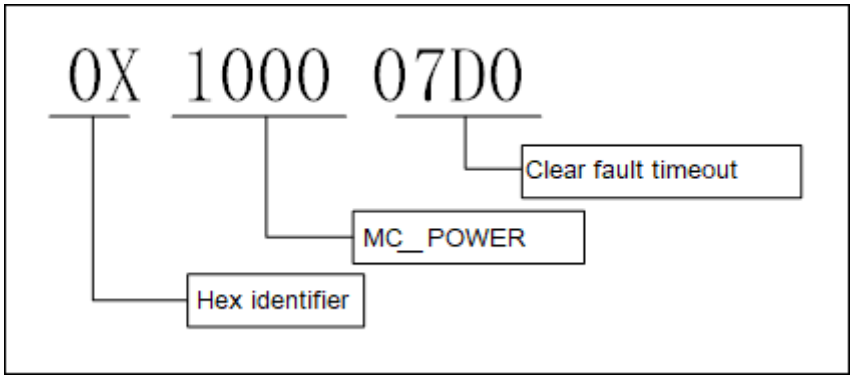
  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.3.8 Fault Codes

The following figure shows an example of the fault code display for the related function:



Take 0x100007D0 as an example:



Instruction code

Instruction Code (hexadecimal)	Name	Function
1000	MC_Power	Enable axis
1001	MC_ReadActualPosition	Read actual position
1002	MC_ReadAxisError	Read axis error
1003	MC_ReadStatus	Read axis status
1004	MC_Reset	Axis error reset
1005	MC_ReadActualTorque	Read actual axis torque
1006	MC_ReadActualVelocity	Read actual axis velocity
1007	MC_SetPosition	Set current axis position
1008	MC_TouchProbe	Touch probe function
1009	MC_ReadBoolParameter	Read BOOL parameter
100A	MC_ReadParameter	Read REAL parameter
100B	MC_WriteBoolParameter	Write BOOL parameter
100C	MC_WriteParameter	Write REAL parameter
100D	MC_AbortTrigger	Abort a triggered motion sequence.
100E	MC_ReadSetPosition	Read the current set target position of the axis.
100F	MC_InPosition	Determine if in position
1011	MC_ChangeGearingRatio	Set or change the electronic gear ratio and the axis type.
1012	MC_SetSoftwareLimits	Set software limits
1013	MC_ReadDigitalInput	Read digital input

Instruction Code (hexadecimal)	Name	Function
1014	MC_SetControllerMode	Set the current operation mode of the servo
1015	MC_CheckLimits	Check if the current driver settings are within the limits.
1016	MC_SetTorqueLimit	Set maximum torque
1017	MC_GetMaxSetAccDec	Read the maximum acceleration and deceleration of the axis.
1018	MC_GetMaxSetVelocity	Read the maximum velocity of the axis.
1019	MC_GetTrackingError	Measure tracking error (current/maximum).
101A	MC_Halt	Axis halt function
101B	MC_Stop	Axis stop function
101C	MC_Home	Drive homing
101D	MC_MoveAbsolute	Point-to-point axis movement to absolute position
101E	MC_MoveRelative	Point-to-point axis movement by relative distance
101F	MC_FollowPosition	Set axis position directly without any check.
1020	MC_FollowVelocity	Set axis velocity directly without any check.
1021	MC_MoveVelocity	Axis velocity instruction
1022	MC_Jog	Axis teach-in jogging
1023	MC_TorqueControl	Torque control function
1024	MC_ImmediateStop	Axis servo emergency stop
1025	MC_MoveFeed	Interrupting feed
1026	MC_SetOverride	Change the axis motion parameter ratio.
1027	MC_ResetFollowingError	Reset following error
1028	MC_HaltSuperImposed	Terminate superimposed motion
1029	MC_MoveSuperImposed	Superimposed motion
102A	MC_MoveContinuousAbsolute	Move to absolute position and continue at the end-of-motion velocity.
102B	MC_MoveContinuousRelative	Move to relative position and continue at the end-of-motion velocity.

Instruction Code (hexadecimal)	Name	Function
102C	MC_CheckAxisCommunication	Check the communication status of the current drive.
102D	MC_Inch	Single-step motion control
102E	MC_Homing	Host controller homing
102F	MC_HomingWithParameter	Set controller homing parameters.
1030	MC_SyncMoveVelocityCSV	Velocity instruction with adjustable pulse width
1031	MC_BacklashCompensate	Backlash compensation function
1032	MC_LeadScrewCompensate	Lead screw pitch error compensation instruction
1033	MC_BackupServoParameter/ MC_BackupServoParameter2	Servo parameter backup instruction
1034	MC_RestoreServoParameter/ MC_RestoreServoParameter2	Servo parameter restoration instruction
1035	MC_VibrationShapping	Write axis variable parameters
1036	MC_FollowPositionVelocity	Set axis position or axis velocity directly without any check.
1037	MC_SDORead	Read from SDO
1038	MC_SDOWrite	Write to SDO
1039	MC_PositionFilter	Position filter
1044	MC_MoveAdditive	Add a distance to the last specified target position
1045	MC_MoveModulo	Rotary axis modulo positioning
2000	MC_GearIn	Start gearing synchronization.
2001	MC_GearOut	Stop gearing synchronization.
2002	MC_Phasing	Master axis phase shifting
2003	MC_StopPhasing	Stop master axis phase shifting
2004	MC_GearInPos	Start synchronization upon reaching a specified master axis position.
2005	MC_GetCamTablePhase	Get phase value in the cam table.
2006	MC_GetCamTableDistance	Get displacement value in the cam table.

Instruction Code (hexadecimal)	Name	Function
2007	MC_DigitalCamSwitch	Tappet control
2008	MC_GenerateCamTable	Update the cam table with defined cam data
2009	MC_WriteCAM	Store the edited cam table as a file
200A	MC_ReadCAM	Read the cam data file and load it into the cam table.
200B	MC_CAMBounds	Calculate maximum position, velocity, and acceleration of the slave axis
200D	MC_CamIn	Start cam synchronization.
200E	MC_CamOut	Stop cam synchronization.
200F	MC_CombineAxes	Output the sum or difference of the instruction positions of two axes as a combined instruction position.
2010	MC_AxesObserve	Observe the position deviation between master and slave axes.
2011	MC_MoveLink	Synchronous positioning
2012	MC_PhasingEx	Complete phase compensation within specified master axis travel distance.
2013	MC_CamInV2	Start cam synchronization.
2014	MC_CheckCamTable	Check the cam table node data.
2015	MC_CamNodeBinding	Binds custom cam nodes to a custom cam table.
3003	MC_GroupEnable	Enable axis group
3004	MC_GroupDisable	Disable axis group
3005	MC_GroupHome	Axis group homing
3006	MC_GroupSetPosition	Set axis group current position
3007	MC_GroupReadActualPosition	Read axis group actual position
3008	MC_GroupReadActualVelocity	Read axis group actual velocity
3009	MC_GroupReadActualAcceleration	Read axis group actual acceleration
300A	MC_GroupStop	Axis group emergency stop
300B	MC_GroupHalt	Stop axis group

Instruction Code (hexadecimal)	Name	Function
300C	MC_GroupSetOverride	Axis group override setting
300D	MC_MoveDirectRelative	Point-to-point relative motion
300E	MC_MoveDirectAbsolute	Point-to-point absolute motion
300F	MC_MoveCircularRelative	Circular relative motion
3010	MC_MoveCircularAbsolute	Circular absolute motion
3011	MC_MoveLinearRelative	Linear relative motion
3012	MC_MoveLinearAbsolute	Linear absolute motion
3013	MC_GroupReset	Reset axis group
3014	MC_GroupReadError	Axis group error reading
3015	MC_GroupReadStatus	Axis group status reading
3016	MC_TrackConveyorBelt	Conveyor belt tracking
3017	MC_SetCartesianTransform	Set transformation between part coordinate system and machine coordinate system
3018	MC_GroupInterrupt	Axis group pause
3019	MC_GroupContinue	Continue axis group motion
3030	MC_GroupPower	Enable/disable axes in axis group
4000	ENC_Counter	Enable encoder (local, bus)
4001	ENC_Reset	Reset encoder (local, bus)
4002	ENC_Preset	Preset encoder (local, bus)
4003	ENC_TouchProbe	Encoder touch probe (local, bus)
4004	ENC_ArrayCompare	Encoder one-dimensional array comparison (local, bus)
4005	ENC_StepCompare	Encoder one-dimensional step comparison (local, bus)
4007	ENC_ReadStatus	Read encoder status (bus)
4008	ENC_DigitalOutput	Encoder digital output control (bus)
400A	ENC_Compare	Encoder axis comparison output (local, bus)
400B	ENC_SetUnit	Set encoder scaling (local, pulse-to-unit ratio)

Instruction Code (hexadecimal)	Name	Function
400C	ENC_SetLineRotationMode	Set linear rotation mode for encoder axis (local)

Fault code

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
0100	0064	The slave address ID exceeds the specified range.	1. The input variable SlaveID of the instruction MC_BackupServoParameter2 exceeded the range [1,65535]. 2. The input variable SlaveID of the instruction MC_RestoreServoParameter2 exceeded the range [1,65535].	1. Set the input variable SlaveID of the instruction MC_BackupServoParameter2 within the range [1,65535]. 2. Set the input variable SlaveID of the instruction MC_RestoreServoParameter2 within the range [1,65535].
0101	0065	The mount path for the SD card does not exist, or the SD card is not inserted.	1. The SD card was not inserted into the controller. 2. The mount path for the SD card in the controller did not exist.	1. Insert the SD card into the controller and power cycle the controller. 2. Check if the mount path for the SD card in the controller is /mnt/sdcard/.
0102	0066	The file name specified by the instruction is null.	1. The input variable FileName of the instruction MC_RestoreServoParameter2 was NULL, meaning the user has not specified a servo parameter backup file for the instruction MC_RestoreServoParameter2.	1. The input variable FileName of the instruction MC_RestoreServoParameter2 must be set to the name of the servo parameter backup file to be restored to the servo.
0103	0067	The file name specified by the instruction exceeds 255 characters.	1. The input variable FileName of the instruction MC_BackupServoParameter2 exceeded 255 characters. 2. The input variable FileName of the instruction MC_RestoreServoParameter2 exceeded 255 characters.	1. The input variable FileName of the instruction MC_BackupServoParameter2 must be limited to 255 characters or fewer. 2. The input variable FileName of the instruction MC_RestoreServoParameter2 must be limited to 255 characters or fewer.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
0104	0068	The file name specified by the instruction contains illegal characters.	1. The input variable FileName of the instruction MC_BackupServoParameter2 contained characters other than numbers, letters, and underscores. 2. The input variable FileName of the instruction MC_RestoreServoParameter2 contained characters other than numbers, letters, and underscores.	1. Set the input variable FileName of the instruction MC_BackupServoParameter2 to a combination of numbers, letters, and underscores. 2. Set the input variable FileName of the instruction MC_RestoreServoParameter2 to a combination of numbers, letters, and underscores.
0105	0069	The servo backup file specified by the instruction already exists in the path, and overwriting the file is prohibited.	1. The servo parameter backup file specified by the input variable FileName of the instruction MC_BackupServoParameter2 already existed in the SD card root directory, while the input variable OverWrite was set to FALSE.	1. Manually delete the servo parameter backup file specified by FileName in the SD card root directory, and then trigger the instruction. 2. Set the input variable OverWrite of the instruction MC_BackupServoParameter2 to TRUE.
0106	006A	The path to the servo parameter backup file does not exist.	1. The SD card was not inserted into the controller. 2. The mount path for the SD card in the controller did not exist.	1. Insert the SD card into the controller and restart the controller. 2. Verify if the mount path for the SD card in the controller is /mnt/sdcard/. If not, modify the mount path.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
0107	006B	The servo parameter backup file to be operated on by the instruction is in write status.	1. When triggering the instruction MC_BackupServoParameter2, the servo parameter backup file to be operated on by the instruction was being written to by another instruction (such as another instance of the instruction MC_BackupServoParameter2). 2. When triggering the instruction MC_RestoreServoParameter2, the servo parameter backup file to be operated on by the instruction was being written to by another instruction (such as the instruction MC_BackupServoParameter2).	1. Before triggering the instruction MC_BackupServoParameter2, check if the servo parameter backup file to be operated on is being written to by another instruction. If so, wait for the other instruction to complete writing before triggering the instruction MC_BackupServoParameter2. 2. Before triggering the instruction MC_RestoreServoParameter2, check if the servo parameter backup file to be operated on is being written to by another instruction. If so, wait for the other instruction to complete writing before triggering the instruction MC_RestoreServoParameter2.
0108	006C	The servo parameter backup file to be operated on by the instruction is in read-only status.	1. When triggering the instruction MC_BackupServoParameter2, the servo parameter backup file to be operated on was being read by another instruction (such as the instruction MC_RestoreServoParameter2).	1. Before triggering the instruction MC_BackupServoParameter2, check if the servo parameter backup file to be operated on is being read by another instruction. If so, wait for the other instruction to complete reading before triggering the instruction MC_BackupServoParameter2.
0109	006D	The servo parameter file specified by the instruction cannot be created or opened.	1. The SD card was not inserted. 2. The mount path for the SD card in the controller did not exist. 3. Check if the servo parameter backup file name contains illegal characters. 4. Check if the specified servo parameter backup file is corrupted.	1. Insert the SD card into the controller and restart the controller. 2. Verify if the mount path for the SD card in the controller is /mnt/sdcard/. If not, modify the mount path. 3. Rename the servo parameter backup file to contain only numbers, letters, and underscores. 4. Import the intact servo parameter backup file to the root directory of the SD card.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
0110	006E	The write-status file (with the .w suffix) for the servo parameter file cannot be created.	1. The SD card was not inserted. 2. The mount path for the SD card in the controller did not exist. 3. Check if the servo parameter backup file name contains illegal characters.	1. Insert the SD card into the controller and restart the controller. 2. Verify if the mount path for the SD card in the controller is /mnt/sdcard/. If not, modify the mount path. 3. Rename the servo parameter backup file to contain only numbers, letters, and underscores.
0111	006F	The read-only file (with the .r suffix) for the servo parameter file cannot be created.	1. The SD card was not inserted. 2. The mount path for the SD card in the controller did not exist. 3. Check if the servo parameter backup file name contains illegal characters.	1. Insert the SD card into the controller and restart the controller. 2. Verify if the mount path for the SD card in the controller is /mnt/sdcard/. If not, modify the mount path. 3. Rename the servo parameter backup file to contain only numbers, letters, and underscores.
0112	0070	The servo parameter backup and restore instruction cannot find the slave specified by the input variable SlaveID of the instruction.	1. The slave specified by SlaveID was not configured.	1. Configure the slave specified by SlaveID.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
0113	0071	The slave specified by the SlaveID in the servo parameter backup and restore instruction is not in the OP status.	1. The communication cables between the controller and the servo slave were improperly connected. 2. The EtherCAT communication of the servo slave was not in the OP status.	1. Check if the controller and the servo slave are powered on and if their communication cables are properly connected. 2. Verify the EtherCAT communication status of the servo slave, and trigger the instruction once the slave is in the OP status.
0114	0072	SDO read operation failed	1. The controller cannot read the specified servo parameters from the servo via SDO communication.	1. Verify whether the servo parameters to be read exist in the servo. 2. Verify whether the servo parameters to be read are readable.
0115	0073	SDO write operation failed	1. The controller cannot write the specified servo parameters to the servo via SDO communication.	1. Verify whether the servo parameters to be written exist in the servo. 2. Verify whether the servo parameters to be written are writable. 3. Verify whether the data length of the servo parameters to be written is correct.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
0116	0074	No matching servo parameter backup template found.	1. The ParameterTemplate folder did not exist in the root directory of the SD card. 2. The ParameterTemplate folder on the SD card did not contain a matching servo parameter backup template for the servo.	1. Verify whether the ParameterTemplate folder exists in the root directory of the SD card; if it does not, manually create it and place the matching servo parameter backup template in the ParameterTemplate folder. 2. Verify whether the corresponding servo parameter backup template exists in the ParameterTemplate folder on the SD card. Verify whether the corresponding servo parameter backup template matches the actual servo model. (For example, the servo parameter backup template for 660N servo corresponds to the 680N servo.)
0117	0075	The version of the servo parameter backup template does not match the version of the servo.	1. The vendor ID, device type, customized number, and MCU software version recorded in the servo parameter backup template did not match the actual servo.	1. Verify whether the servo parameter backup template matches the actual servo; if it does not, replace or modify the template.
0118	0076	The slave is not an Inovance product.	1. The slave specified by SlaveID was a third-party product.	1. The servo parameter backup and restore instructions only support Inovance EtherCAT servos. Third-party products are not supported and must be replaced with Inovance EtherCAT servos.
0119	0077	The slave is not a servo product.	1. The slave specified by SlaveID was a non-servo device.	1. The servo parameter backup and restore instructions only support Inovance EtherCAT servos. Non-servo devices are not supported and must be replaced with Inovance EtherCAT servos.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
0120	0078	The version of the servo parameter backup file does not match the version of the servo.	1. The vendor ID, device type, customized number, and MCU software version recorded in the servo parameter backup file did not match the actual servo.	1. Verify if the servo parameter backup file matches the actual servo. If it does not match, replace the servo parameter backup file.
0121	0079	SD card read operation failed	1. The SD card was not inserted. 2. The mount path for the SD card in the controller did not exist. 3. The servo parameter backup template or the servo parameter backup file to be read did not exist or is damaged. 4. The servo parameter backup template to be read contained servo parameters that do not exist on the current servo or are not accessible via SDO.	1. Insert the SD card into the controller and restart the controller. 2. Verify if the mount path for the SD card in the controller is /mnt/sdcard/. If not, modify the mount path. 3. Verify if the servo parameter backup template or servo parameter backup file exists in its corresponding location and is intact. 4. Verify if the servo parameter backup template contains servo parameters that do not exist on the current servo or are not accessible via SDO.
0122	007A	SD card write operation failed	1. The SD card was not inserted. 2. The mount path for the SD card in the controller did not exist. 3. The servo parameter backup file to be written was damaged.	1. Insert the SD card into the controller and restart the controller. 2. Verify if the mount path for the SD card in the controller is /mnt/sdcard/. If not, modify the mount path. 3. Verify if the servo parameter backup file is intact.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
0123	007B	The current servo status does not meet the requirements for executing the MC_RestoreServoParameter2 instruction.	1. When triggering the MC_RestoreServoParameter2 instruction, the state machine of the servo slave was in any status other than "No fault", "Servo ready", "Waiting for the S-ON signal", or "Fault".	1. Adjust the state machine of the servo slave to one of the following statuses: "No fault", "Servo ready", "Waiting for the S-ON signal", or "Fault". Then, trigger the MC_RestoreServoParameter2 instruction.
0124	007C	The servo parameter backup instruction MC_BackupServoParameter2 cannot be re-triggered.	1. After triggering an instance of the servo parameter backup instruction MC_BackupServoParameter2, the same instance was re-triggered while it was in the Busy status.	1. After triggering an instance of the servo parameter backup instruction MC_BackupServoParameter2, wait for the same instance to complete execution before triggering it again.
0125	007D	The servo parameter restoration instruction MC_RestoreServoParameter2 cannot be re-triggered.	1. After an instance of the servo parameter restoration instruction MC_RestoreServoParameter2 had been triggered, the same instance was re-triggered while it was in the Busy status.	1. After triggering an instance of the servo parameter restore instruction MC_RestoreServoParameter2, wait for the same instance to complete execution before triggering it again.
0126	007E	The SlaveID specified for the servo parameter backup or restore instruction is in a busy status.	1. When triggering the servo parameter backup or restore instruction, another instruction was already occupying the SDO channel of the slave specified by SlaveID for communication.	1. After all other instructions have finished using the SDO channel of the slave specified by SlaveID, trigger the servo parameter backup or restore instruction.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
0127	007F	Servo parameter length mismatch	1. The servo parameter length was wrong in the servo parameter backup template, so the backup file generated from the template by the servo parameter backup instruction would also contain the wrong parameter length. If this corrupted data was restored with the wrong parameter length by triggering the servo restore instruction MC_RestoreServoParameter2, it would cause an instruction failure.	1. Check if the length data of the servo parameters is correct in the servo parameter backup template. If not, correct the template. 2. Check if the length data of the servo parameters is correct in the servo parameter backup file. If not, replace the file.
1001	03E9	Invalid position parameter	1. When the software limits were enabled, the value of the "Position" input parameter in the MC_FollowPosition instruction exceeded the positive or negative limit. 2. The value of the "PosWindow" input parameter in the MC_InPosition instruction was less than 0. 3. For linear axes, the value of the "FirstPosition" input parameter in the MC_TouchProbe and MC_MoveFeed instructions was greater than or equal to the value of "LastPosition". 4. For a rotary axis, the value of the "FirstPosition" input parameter in the MC_TouchProbe and MC_MoveFeed instructions was equal to the value of "LastPosition".	1. Adjust the input parameter of the corresponding instructions to the value within the valid range as required.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
1002	03EA	Invalid velocity parameter	1. The value of the "Velocity" input parameter in the MC_Phasing instruction was less than or equal to 0. 2. For the MC_CamIn, MC_GearIn, MC_GearInPos, MC_MoveLink, MC_TorqueControl, MC_HomingWithParameter, MC_MoveAbsolute, MC_MoveRelative, MC_MoveContinuousAbsolute, MC_MoveContinuousRelative, MC_MoveVelocity, MC_SyncMoveVelocityCSV, and MC_MoveFeed instructions, the value of the "Velocity" input parameter was less than or equal to 0, or greater than the maximum velocity specified by the software. 3. For the MC_Jog and MC_Inch instructions, the value of the input parameter "Velocity" was less than or equal to 0, or greater than the maximum commissioning velocity specified by the software. 4. For the MC_FollowVelocity instruction, the absolute value of the input parameter "Velocity" was greater than the maximum velocity specified by the software. 5. The value of the input parameter "VelocityDiff" for the MC_MoveSuperImposed instruction was less than or equal to 0, or greater than the maximum velocity specified by the software.	1. Adjust the input parameter of the corresponding instructions to the value within the valid range as required.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
1003	03EB	Invalid acceleration parameter	1. The value of the input parameter "Acceleration" for the MC_Phasing instruction was less than or equal to 0. 2. For the MC_CamIn, MC_GearIn, MC_GearInPos, MC_MoveLink, MC_TorqueControl, MC_HomingWithParameter, MC_MoveAbsolute, MC_MoveRelative, MC_MoveContinuousAbsolute, MC_MoveContinuousRelative, MC_MoveVelocity, MC_SyncMoveVelocityCSV, and MC_MoveFeed instructions, the value of the "Acceleration" input parameter was less than or equal to 0, or greater than the maximum acceleration specified by the software. 3. For the MC_Jog and MC_Inch instructions, the value of the input parameter "Acceleration" was less than or equal to 0, or greater than the maximum commissioning acceleration specified by the software. 4. The value of the input parameter "Acceleration" for the MC_MoveSuperImposed instruction was less than or equal to 0, or greater than the maximum acceleration specified by the software.	1. Adjust the input parameter of the corresponding instructions to the value within the valid range as required.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
1004	03EC	Invalid deceleration parameter	1. The value of the input parameter "Deceleration" for the MC_Phasing and MC_StopPing instructions was less than or equal to 0. 2. For the MC_CamIn, MC_GearIn, MC_GearInPos, MC_MoveLink, MC_TorqueControl, MC_HomingWithParameter, MC_MoveAbsolute, MC_MoveRelative, MC_MoveContinuousAbsolute, MC_MoveContinuousRelative, MC_MoveVelocity, MC_SyncMoveVelocityCSV, MC_MoveFeed, MC_Stop, MC_Halt instructions, the value of the "Deceleration" input parameter was less than or equal to 0, or greater than the maximum deceleration specified by the software. 3. For the MC_Jog and MC_Inch instructions, the value of the input parameter "Deceleration" was less than or equal to 0, or greater than the maximum commissioning deceleration specified by the software. 4. The value of the input parameter "Deceleration" for the MC_MoveSuperImposed and MC_HaltSuperImposed instructions was less than or equal to 0, or greater than the maximum deceleration specified by the software.	1. Adjust the input parameter of the corresponding instructions to the value within the valid range as required.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
1005	03ED	Invalid jerk parameter	When using an S-curve motion profile: 1. The value of the input parameter "Jerk" for the MC_Phasing and MC_StopPing instructions was less than or equal to 0. 2. For the MC_CamIn, MC_GearIn, MC_GearInPos, MC_MoveLink, MC_TorqueControl, MC_HomingWithParameter, MC_MoveAbsolute, MC_MoveRelative, MC_MoveContinuousAbsolute, MC_MoveContinuousRelative, MC_MoveVelocity, MC_SyncMoveVelocityCSV, MC_MoveFeed, MC_Stop, MC_Halt instructions, the value of the "Jerk" input parameter was less than or equal to 0, or greater than the maximum jerk specified by the software. 3. For the MC_Jog and MC_Inch instructions, the value of the input parameter "Jerk" was less than or equal to 0, or greater than the maximum jerk specified by the software. 4. The value of the input parameter "Jerk" for the MC_MoveSuperImposed and MC_HaltSuperImposed instructions was less than or equal to 0, or greater than the maximum jerk specified by the software.	1. Adjust the input parameter of the corresponding instructions to the value within the valid range as required.

1007	03EF	Invalid motion direction parameter	1. The input parameter "Direction" for the MC_CamIn instruction has the following value range: DIR_POSITIVE(1), DIR_NEGATIVE(2), DIR_NODIRECTION(4). 2. The input parameter "DirectionLocal" for the ENC_Counter instruction has the following value range: DIR_SHORTEST(0), DIR_POSITIVE(1). 3. The input parameter "Direction" for the MC_MoveAbsolute and MC_MoveContinuousAbsolute instructions has the following value range: DIR_SHORTEST(0), DIR_POSITIVE(1), DIR_NEGATIVE(2), DIR_CURRENT(3), DIR_NODIRECTION(4), DIR_FASTEST(5). 4. The input parameter "Direction" for the MC_MoveVelocity and MC_SyncMoveVelocityCSV instructions has the following value range: DIR_POSITIVE(1), DIR_NEGATIVE(2), DIR_CURRENT(3). 5. When the MC_MoveFeed instruction operates in velocity mode, the input parameter "Direction" has the following value range: DIR_POSITIVE(1), DIR_NEGATIVE(2), DIR_CURRENT(3). 6. For the MC_MoveFeed instruction in absolute mode on a rotary axis, the input parameter "Direction" has the following value range: DIR_SHORTEST(0), DIR_POSITIVE(1), DIR_NEGATIVE(2), DIR_CURRENT(3), DIR_NODIRECTION(4). 7. The input parameter "DirectionLocal" for the ENC_Counter instruction has the following value range: positive direction (0), negative direction (1).	1. Adjust the input parameter of the corresponding instructions to the value within the valid range as required.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
1010	03F2	Invalid torque parameter	1. The absolute value of the input parameter "Torque" for the MC_TorqueControl instruction cannot be greater than fMaxPositiveTorque or fMaxNegativeTorque of the axis structure.	1. Adjust the input parameter of the corresponding instructions to the value within the valid range as required.
1011	03F3	Invalid torque ramp parameter	1. The value of the input parameter "TorqueRamp" for the MC_TorqueControl instruction was less than 0.	1. Adjust the input parameter of the corresponding instructions to the value within the valid range as required.
1012	03F4	Invalid control mode parameter	1. The input parameter "ControllerMode" for the MC_SetControllerMode instruction has the following value range. 10: MC_TORQUE; 9: MC_VELOCITY; 8: MC_POSITION.	1. Adjust the input parameter of the corresponding instructions to the value within the valid range as required.
1013	03F5	The probe ID number is not within the supported range.	1. The input parameter "ProbeID" for the MC_MoveLink, MC_TouchProbe, MC_AbortTrigger, and MC_MoveFeed instructions has the following value range: 0 (probe 1), 1 (probe 2).	1. Adjust the input parameter of the corresponding instructions to the value within the valid range as required.
1014	03F6	The edge trigger parameter is not within the supported range.	1. The input parameter "TriggerEdge" for the MC_MoveLink and MC_MoveFeed instructions has the following value range: 0 (Only rising edge-triggered), 1 (Only falling edge-triggered). 1. The input parameter "TriggerEdge" for the MC_TouchProbe instruction has the following value range: 0 (Only rising edge-triggered), 1 (Only falling edge-triggered), 2 (Both rising and falling edges triggered).	1. Adjust the input parameter of the corresponding instructions to the value within the valid range as required.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
1015	03F7	The trigger source parameter is not within the supported range.	1. When the axis is a virtual axis or a local pulse axis, the input parameter "TerminalSource" for the MC_MoveLink, MC_TouchProbe, and MC_MoveFeed instructions is set to 1 (encoder Z signal). 2. When the axis is a bus-controlled axis, the input parameter "TerminalSource" for the MC_MoveLink, MC_TouchProbe, and MC_MoveFeed instructions is set to a value other than 0 (DI terminal) or a value other than 1 (encoder Z signal).	1. Adjust the input parameter of the corresponding instructions to the value within the valid range as required.
1016	03F8	The trigger mode parameter is not within the supported range.	1. When the input parameter "LinkMode" for the MC_MoveLink instruction is 2 (Active when the TriggerValue is detected to be TRUE), "TriggerMode" has the following value range: 0 (drive mode), 1 (controller mode). 2. The input parameter "TriggerMode" for the MC_TouchProbe instruction has the following value range: 0 (single trigger), 1 (continuous trigger)	1. Adjust the input parameter of the corresponding instructions to the value within the valid range as required.
1017	03F9	Invalid time parameter	1. The input parameter "PosTime" or "TimeOutValue" for the MC_InPosition instruction was less than 0.	1. For the MC_InPosition instruction, the input parameter "PosTime" or "TimeOut Value" must be greater than 0.
1018	03FA	Invalid electronic gear ratio denominator setting	1. The user input parameter "RatioTechUnitsDenom" on the instruction side was set to 0.	1. Modify the set value of the user input parameter "RatioTechUnitsDenom" on the instruction side.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
1019	03FB	Invalid electronic gear ratio numerator setting	1. The user input parameter "RatioTechUnitsNum" on the instruction side was set to 0. 2. ENC_SetUnit: The user input parameter "Numerator" on the instruction side was set to a value less than or equal to 0.	1. Modify the set value of the user input parameter "RatioTechUnitsNum" on the instruction side. 2. Modify the set value of the user input parameter "Numerator" on the instruction side.
1020	03FC	Invalid cycle length setting for the rotation mode	1. MC_ChangeGearingRatio: When the user input parameter "MovementType" on the instruction side was set to 0 (rotary axis), the set value of the user input parameter "PositionPeriod" on the instruction side was less than or equal to 0. 2. ENC_SetLineRotationMode: When the user input parameter "MovementType" on the instruction side was set to 1 (rotary axis), the set value of the user input parameter "PositionPeriod" on the instruction side was less than or equal to 0, or the pulse unit corresponding to "PositionPeriod" was greater than 2147483647. 3. ENC_SetLineRotationMode: When the user input parameter "MovementType" on the instruction side was set to 1 (rotary axis), the product of the "PositionPeriod" set value on the instruction side and the axis gear ratio was greater than 2147483647.	1. Modify the set value of the input parameter "PositionPeriod" on the instruction side.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
1021	03FD	Axis motion type error	1. MC_ChangeGearingRatio: The user input parameter "MovementType" on the instruction side has the following value range: 0 (rotary axis), 1 (linear axis). 2. ENC_SetLineRotationMode: The user input parameter "MovementType" on the instruction side has the following value range: 0 (linear axis), 1 (rotational axis).	1. Modify the set value of the input parameter "MovementType" on the instruction side.
1022	03FE	The positive limit position is less than or equal to negative limit position.	1. The set value of the user input parameter "PositiveLimitPos" on the instruction side was less than or equal to that of the user input parameter "NegativeLimitPos" on the instruction side.	1. Modify the set value of the user input parameter "PositiveLimitPos" on the instruction side. 2. Modify the set value of the user input parameter "NegativeLimitPos" on the instruction side.
1023	03FF	Invalid deceleration distance parameter setting	1. The user input parameter "MaxDistance" on the instruction side was set to a value less than 0.	1. Modify the set value of the input parameter "MaxDistance" on the instruction side.
1025	0401	Emergency stop mode parameter error in emergency stop	1. The user input parameter "StopMode" on the instruction side was not set to 0 (immediate stop) or 1 (immediate stop with servo disable).	1. Modify the set value of the user input parameter "StopMode" on the instruction side.
1026	0402	The motion mode parameter is not within the supported range.	1. The user input parameter "MoveMode" on the instruction side has the following value range: AbsoluteMode(0), RelativeMode(1), VelocityMode(2).	1. Modify the set value of the user input parameter "MoveMode" on the instruction side.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
1027	0403	Buffer mode out of range	1. The input parameter "BufferMode" for the instructions MC_CamIn, MC_GearIn, MC_MoveLink, MC_MoveContinuousAbsolute, MC_MoveContinuousRelative, MC_MoveVelocity, MC_Halt, MC_TorqueControl, and MC_MoveFeed has a value range: mcAbort (0), mcBuffered (1). 2. The input parameter "BufferMode" for the instruction MC_GearInPos must be set to mcAbort (0). 3. The input parameter "BufferMode" for the instructions MC_MoveAbsolute and MC_MoveRelative has the following value range: mcAbort (0), mcBuffered (1), mcBlendingLow (2), mcBlendingPrevious (3), mcBlendingNext (4), and mcBlendingHigh (5). 4. For the instructions MC_PhasingEx, MC_CombineAxes, and MC_Phasing, the input parameter "BufferMode" must be set to mcAbort (0).	1. Modify the set value of the user input parameter "BufferMode" on the instruction side.
1028	0404	Gear parameter modification not allowed after axis is enabled	1. The instruction was triggered once the axis was enabled.	1. Disable the axis before triggering this instruction.
1029	0405	Feed velocity parameter out of range	1. The user input parameter "FeedVelocity" on the instruction side was set to a value less than or equal to 0, or exceeded the set value of "Velocity" in the "Motion parameter limit" on the FA side.	1. Modify the set value of the user input parameter "FeedVelocity" on the instruction side. 2. Modify the set value of "Velocity" in the "Motion parameter limit" on the FA side.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
1030	0406	Set position exceeds software limits	1. When the "Enable" checkbox in "Axis - General settings - Basic settings - Software limit" on the FA side was checked, the set value of the input parameter "FirstPosition" on the instruction side was less than that of "Negative limit" in the "Software limit" on the FA side. 2. When the "Enable" checkbox in "Axis - General settings - Basic settings - Software limit" on the FA side was checked, the set value of the user input parameter "LastPosition" on the instruction side was greater than that of "Positive limit" in "Software limit" on the FA side.	1. Modify "Negative limit" or "Positive limit" in "Software limit" on the FA side. 2. Modify the set value of the user input parameter "FirstPosition" or "LastPosition" on the instruction side.
1031	0407	Incorrect homing direction setting	1. The "StartDirection" member in the user input parameter "HomingParameter" of type structure on the instruction side was set to a value that is neither "DIR_POSITIVE" nor "DIR_NEGATIVE".	1. Modify the set value of "StartDirection" in the structured user input parameter "HomingParameter" on the instruction side.
1032	0408	Creeping speed greater than homing speed	1. The set value of "ApproachVel" in the structured user input parameter "HomingParameter" on the instruction side was greater than that of "Vel".	1. Modify the set value of "ApproachVel" in the structured user input parameter "HomingParameter" on the instruction side. 2. Modify the set value of "Vel" in the structured user input parameter "HomingParameter" on the instruction side.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
1034	040A	Incorrect homing detection direction configuration	1. The "EndDirection" member in the structured user input parameter "HomingParameter" on the instruction side was set to a value that is neither "DIR_POSITIVE" nor "DIR_NEGATIVE". 2. The "ZeroSignal" member in the structured user input parameter "HomingParameter" on the instruction side was set to a value that is neither "HOME_INPUT_SIGNAL_EXTERNALSIGNAL" nor "HOME_INPUT_SIGNAL_ZPHASE".	1. Modify the set value of "EndDirection" in the structured user input parameter "HomingParameter" on the instruction side. 2. Modify the set value of "ZeroSignal" in the structured user input parameter "HomingParameter" on the instruction side.
1036	040C	Invalid ending velocity parameter	1. The user input parameter "EndVelocity" on the instruction side was set to a value less than or equal to 0, or exceeded the set value of "Velocity" in the "Motion parameter limits" on the FA side.	1. Modify the set value of the user input parameter "EndVelocity" on the instruction side. 2. Modify the set value of "Velocity" in the "Motion parameter limit" on the FA side.
1037	040D	Invalid end motion direction parameter	1. The user input parameter "EndVelocityDirection" on the instruction side was not set to "DIR_POSITIVE", "DIR_NEGATIVE", or "DIR_CURRENT".	1. Modify the set value of the user input parameter "EndVelocityDirection" on the instruction side.
1038	040E	Invalid creeping speed parameter setting	1. The set value of "ApproachVel" in the structured user input parameter "HomingParameter" on the instruction side was greater than that of "Velocity" in the "Motion parameter limit" on the FA side, or the set value of "ApproachVel" was less than or equal to 0.	1. Modify the set value of "ApproachVel" in the structured user input parameter "HomingParameter" on the instruction side. 2. Modify the set value of "Velocity" in the "Motion parameter limit" on the FA side.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
1039	040F	Invalid mask distance setting in the host controller homing mode	1. When the value of "HomingMode" in the structured input user parameter "HomingParameter" on the instruction side was set to "HOMING_METHOD_HOMESWMASKDISTANCEHOMESWTURN", the set value of "MaskDistance" in the structured "HomingParameter" was less than or equal to 0.	1. If the value of "HomingMode" in the structured input user parameter "HomingParameter" on the instruction side is set to "HOMING_METHOD_HOMESWMASKDISTANCEHOMESWTURN", modify the set value of "MaskDistance" in the structured parameter "HomingParameter".
1040	0410	Invalid value range for the user input parameter "Distance" on the instruction side	1. The value of the user input parameter "Distance" on the instruction side was less than 0.	1. Modify the value of the user input parameter "Distance" on the instruction side.
1043	0413	Phase superimposition mode parameter error	1. The set value of the user input parameter "Mode" on the instruction side was invalid.	1. The user input parameter "Mode" on the instruction side has the following value range: 0 (Phase compensation continues when master velocity is 0) or 1 (Phase compensation stops when master velocity is 0). Set the appropriate phase compensation mode according to your requirements.
1044	0414	Phase superimposition shift type parameter error	1. The set value of the user input parameter "RelativeMode" on the instruction side was invalid.	1. The user input parameter "RelativeMode" on the instruction side has the following value range: 0 (Specify PhaseShift as an absolute phase offset relative to the master reference, overwriting any existing offset) or 1 (Specify PhaseShift as a relative phase offset, adding to the current offset between the master and slave axes). Set the appropriate phase compensation mode as required.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
1046	0416	Probe 1 terminal not configured	1. The probe 1 of the axis corresponding to the instruction was not configured.	1. Enable the required probe channel in the Axis - Probe settings on the FA side. Select the required terminal.
1047	0417	Probe 2 terminal not configured	1. The probe 2 of the axis corresponding to the instruction was not configured.	1. Enable the required probe channel in the Axis - Probe settings on the FA side. Select the required terminal.
1048	0418	Filter time parameter out of range	1. The input parameter "FilterTime" of the instruction exceeded the valid range of 2 to 100.	1. Adjust the input parameter "FilterTime" to the value within the valid range.
1049	0419	Unsupported filter mode for local encoder axis	1. The filter modes 0, 3, 5, 6, 8, and 10 were incompatible with local encoder axes.	1. If using a local encoder axis, select a compatible filter mode.
1050	041A	Filter type out of range	1. The input parameter "FilterType" of the instruction exceeded the range of 0 to 11.	1. Adjust the input parameter "FilterType" to the value within the valid range.
1051	041B	The motion filtering has been applied.	1. There was already an ongoing filtering function, when the current instruction was triggered.	1. Stop the ongoing filtering function, and then re-trigger the current instruction.
1052	041C	The signal source for the synchronization instruction is set to mcPositionFilter but the position filtering function is disabled.	1. When the synchronization instruction was triggered, the signal source was set to mcPositionFilter, but the position filtering function was not enabled. 2. When the synchronization instruction was executed, the signal source was set to mcPositionFilter, and the position filtering function was disabled.	1. When selecting mcPositionFilter for synchronization instruction, the position filtering function must be activated before startup.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
1053	041D	The input SDO parameter length is invalid.	The value "DataLength" was invalid.	Adjust DataLength to a value within the range of 1 to 4.
1054	041E	For the filter mode 6 or 7, the low-pass filter parameter is less than or equal to 0.	1. When the filter mode 6 or 7 was selected, the value of the low-pass filter parameter was invalid.	1. When the filter mode 6 or 7 is selected, the value of the low-pass filter parameter must be greater than 0. Set a proper low-pass filter parameter based on the actual conditions.
2000	07D0	Timeout while clearing error	1. The communication status of the servo or slave was faulty.	1. Check the status of the slave. 2. Check the fault code on the servo. 3. See the servo manual.
2001	07D1	Shut Down waiting timeout	1. The communication status of the servo or slave was faulty.	1. Check the status of the slave. 2. Check the fault code on the servo. 3. See the servo manual.
2002	07D2	Switch On waiting timeout	1. The communication status of the servo or slave was faulty.	1. Check the status of the slave. 2. Check the fault code on the servo. 3. See the servo manual.
2003	07D3	Enable switchover timeout	1. The communication status of the servo or slave was faulty.	1. Check the status of the slave. 2. Check the fault code on the servo. 3. See the servo manual.
2004	07D4	Servo enable loss during operation	1. The communication status of the servo or slave was faulty.	1. Check the status of the slave. 2. Check the fault code on the servo. 3. See the servo manual.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
3000	0BB8	Incorrect control mode for the current axis	1. The axis control mode corresponding to the MC_SetPosition instruction was not MC_CSP. This instruction can only be triggered in the above axis control mode. 2. During the execution of the MC_TorqueControl instruction, the axis control mode has been switched to a non-CST mode. 3. The axis control mode corresponding to the instruction was neither MC_CSP nor MC_CSV. The instruction can only be triggered in the above two modes.	1. Modify the current axis control mode.
3001	0BB9	Instruction not permitted in the axis state	The instruction was initiated while the axis was in the STOPPING or HOMING state.	1. Reset the input parameter Execute of the MC_Stop or MC_Homing instruction. Only after the axis has come to a complete standstill may you switch states and restart the instruction.
3003	0BBB	Motion planning parameter solver failed	1. The user input motion parameters on the instruction side caused the motion planning solver to fail.	1. Modify the user input motion parameters on the instruction side. 2. Adjust the trigger point of the instruction.
3004	0BBC	Buffer data full	1. When the user input parameter "BufferMode" was set to a value other than "MC_ABORTING" on the instruction side, the buffer data was out of limit.	1. Modify the user program to ensure that no more than one instruction has the parameter "BufferMode" set to a value other than "MC_ABORTING".
3008	0BC0	Axis not enabled	1. The axis was not enabled. 2. The MC_Power instruction was executed, but the servo had an unresolvable error, causing the enabling failure. 3. The axis was disabled during the instruction execution.	1. Check if the user program executes the MC_Power instruction. 2. Check and resolve the servo error. 3. Check if the axis is disabled when executing the user program.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
3009	0BC1	The axis to execute the instruction is a non-control axis.	1. The instruction was triggered using the non-control axis.	1. The instruction is only applicable to bus-controlled axes and local pulse axes.
3010	0BC2	The axis is in the homing process and does not meet the execution conditions.	1. The instruction was triggered when the axis was in the AXIS_HOMING status (The axis is in the homing process).	1. Modify the user program to prevent triggering the instruction when the axis in the AXIS_HOMING status.
3011	0BC3	Axis homing interrupted or not started	1. The axis homing has been interrupted or has not been started.	1. See the servo manual.
3012	0BC4	Axis homing completed but the target position not reached	1. The axis has completed the homing, but did not reach the target position.	1. See the servo manual.
3013	0BC5	Axis homing error with non-zero speed	1. The error occurred during the axis homing and the speed was not 0. 2. The positive and negative limits were activated simultaneously during the homing process.	1. See the servo manual. 2. Trigger the limit and home signals correctly during the homing process according to the homing method.
3014	0BC6	Axis homing error with the speed at zero	1. The error occurred during the axis homing and the speed was 0.	1. See the servo manual.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
3017	0BC9	Probe trigger position out of defined range	1. For a linear axis, the value of the current instruction position was less than the set value of the input parameter "firstPosition" or greater than the set value of the input parameter "lastPosition". 2. For a rotary axis, when the input parameter "firstPosition" was greater than "lastPosition", the current instruction position was greater than "lastPosition" and less than "firstPosition"; when the input parameter "firstPosition" was less than "lastPosition", the current instruction position was greater than "lastPosition" and less than 360°, or the current instruction position was greater than 0 and less than "firstPosition".	1. Modify the set value of the input parameter "firstPosition" or of the input parameter "lastPosition".
3018	0BCA	Probe trigger interrupted	1. While waiting for a probe trigger, the process was interrupted by a new MC_TouchProbe instruction.	1. Check if a new MC_TouchProbe instruction is triggered while waiting for a probe trigger.
3019	0BCB	Waiting timeout	1. The running time for the MC_InPosition instruction exceeded the set value of the input parameter "TimeOutValue" on the instruction side. 2. After the MC_TorqueControl instruction ran for over 5 seconds, the control mode of the current axis still has not been switched to MC_CST.	1. Modify the set value of the input parameter "TimeOutValue" on the instruction side. 2. Check if any error occurred on the servo side.
3020	0BCC	Virtual axis mode not supported	1. This instruction did not support triggering in virtual axis mode.	1. Modify the axis type in "Axis - Axis basic settings - Axis type" on the FA side. 2. Modify the user program.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
3021	0BCD	Position type parameter error (planning or feedback)	1. The value of the input parameter "ReferenceType" on the instruction side was neither 0 (Value calculated in the previous task period) nor 1 (Value obtained in the same task period).	1. Modify the set value of the input parameter "ReferenceType" on the instruction side.
3022	0BCE	Interrupting feed motion completed (absolute/ relative)	1. When the input parameter "ErrorDetect" of the MC_MoveFeed instruction was set to TRUE, the set signal source was not triggered upon completion of the absolute/relative interrupting feed motion.	1. Check if the corresponding signal source is triggered normally. 2. Modify the value of the input parameter "ErrorDetect" to FALSE. Do not check if the signal source is triggered during the instruction execution.
3023	0BCF	Duplicate setting of probe trigger function	1. The probe channel enabled by the instruction was occupied.	1. Do not enable the same probe channel from different instruction instances.
3024	0BD0	The axis is in the AXIS_STOPPING status.	1. The instruction was initiated while the axis was in the AXIS_STOPPING status.	1. Do not trigger the instruction when the axis is in the AXIS_STOPPING status.
3025	0BD1	Rotary axis type not supported	1. The instruction did not support triggering in rotary axis mode.	1. Change the motion mode to "Linear mode" in "Axis - General settings - Basic settings - Software limit" on the FA side. 2. Modify the user program.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
3026	0BD2	When executing the same function block of the instruction, the function block is re-triggered after changing the buffer mode to a mode other than mcAborting.	1. During the instruction execution, the buffer mode was changed to a mode other than mcAborting, and then the same function block was re-triggered.	1. Do not re-trigger the same function block after changing the buffer mode to a mode other than mcAborting during the instruction execution.
3028	0BD4	Axis under planning	1. When triggering the instruction, the axis was in motion.	1. Do not trigger the instruction when the axis is in motion.
3029	0BD5	Limit triggered while the planning speed direction matches the limit trigger direction.	1. During the instruction execution, the current instruction position exceeded the value of the positive or negative software limits, and the speed direction was consistent with the limit trigger direction. 2. During the instruction execution, the positive or negative hardware limit switches were triggered, and the speed direction was consistent with the limit trigger direction. 3. The motion instruction drove the axis into the limit.	1. Modify the set value of "Positive limit" or "Negative limit" in "Software Limit" on the FA side. 2. Change the speed direction and re-trigger the instruction.
3031	0BD7	Invalid input PN parameter	1. The set value of the input parameter "ParameterNumber" on the instruction side was incorrect.	1. Modify the set value of the input parameter "ParameterNumber" on the instruction side.
3032	0BD8	Invalid input value	1. "Value" corresponding to the set value of the input parameter "ParameterNumber" on the instruction side was not within a valid range.	1. Modify "Value" associated with the set value of the input parameter "ParameterNumber" on the instruction side.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
3033	0BD9	Invalid input PDO parameter	1. The set value of the input parameter "ParameterNumber" on the instruction side was incorrect.	1. Modify the set value of the input parameter "ParameterNumber" on the instruction side.
3034	0BDA	Buffer overflow	1. The file write buffer for the MC_AxisDiagnosticLog instruction has exceeded its limit.	1. Change the logging mode. 2. Rename the file and re-trigger the instruction.
3035	0BDB	Failed to open file	1. The file cannot be opened by the MC_AxisDiagnosticLog or MC_WriteCAM instruction.	1. Check if the file path is correct. 2. Check if the file is already in use.
3036	0BDC	The value of "ParameterNumber" cannot be modified while the axis parameters are being read.	1. The set value of the input parameter "ParameterNumber" on the instruction side was modified by the instruction while Enable was active.	1. The set value of the input parameter "ParameterNumber" cannot be modified while Enable was active. To modify the value of "ParameterNumber," re-trigger the instruction after making the change.
3037	0BDD	Log mode parameter error	1. The set value of the input parameter "Mode" on the instruction side was neither 0 (log_continuous: continuous logging to the file) nor 1 (log_at_close: continuous logging to the buffer (10kbyte)).	1. Modify the set value of the input parameter "Mode" on the instruction side.
3038	0BDE	Reverse motion disallowed: zero displacement with non-zero velocity and acceleration	1. The InverseFlag in the function block was set false to disallow the reverse motion. The given speed and acceleration with the instruction displacement at zero would cause reverse motion.	1. Set the InverseFlag to TRUE to allow reverse motion based on actual requirements. 2. Adjust the target position to achieve a non-zero instruction displacement. This displacement must be greater than the stopping distance calculated from the current speed and acceleration.
3039	0BDF	Following error reset in progress	1. When the current instruction was executing, another instance of the same instruction has been triggered.	1. When the instruction is still running, do not trigger another instance of the same instruction.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
3040	0BE0	Servo error cannot be cleared	1. The number of cycles executed for the instruction has exceeded the maximum waiting period for clearing servo errors.	1. If the critical error that the instruction cannot clear has occurred in the servo, see the servo manual.
3041	0BE1	MC_ResetFollowingError already active	1. The MC_ResetFollowingError instruction was re-triggered while the instruction was active.	1. Modify the user program. Do not re-trigger the MC_ResetFollowingError instruction while the instruction is active.
3042	0BE2	Current axis status not in POWER_OFF status	1. The axis status for the MC_ChangeGeaingRatio instruction was not in the POWER_OFF status.	1. Disable the axis to be operated before triggering this instruction.
3043	0BE3	Current axis status not in AXIS_STANDSTILL	1. The axis status was not AXIS_STANDSTILL when triggering the instruction.	1. Do not trigger the instruction when the axis status is not AXIS_STANDSTILL.
3044	0BE4	Feedback control mode not switched to the set mode during switching	1. The current MC_SetControllerMode instruction was re-triggered, when the instruction was running and switching modes. 2. The current MC_SetControllerMode instruction was triggered again, when another MC_SetControllerMode instruction was running and switching modes.	1. Do not re-trigger the current instruction, when the instruction is running and switching modes. 2. Do not trigger a new MC_SetControllerMode instruction, when another MC_SetControllerMode instruction is running and switching modes.
3045	0BE5	Do not re-trigger the function block when it is executing.	1. The current MC_TouchProbe instruction was re-triggered when the instruction was running.	1. Do not re-trigger the current instruction MC_TouchProbe when the instruction is running.
3046	0BE6	Do not re-trigger the function block when it is executing.	1. The current MC_MoveFeed instruction was re-triggered when the instruction was running.	1. Do not re-trigger the current instruction MC_MoveFeed when the instruction is running.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
3047	0BE7	Unconfigured PDO object 607A for target position	1. When the control mode of the axis operated by this instruction was MC_CSP, "Target position" was not configured in "Process data (PDO configuration)" on the FA side.	1. Configure "Target position" in "Process data (PDO configuration)" on the FA side.
3048	0BE8	Unconfigured PDO object 60FF for target velocity	1. When the control mode of the axis operated by this instruction was MC_CSV, "Target velocity" was not configured in "Process data (PDO configuration)" on the FA side.	1. Configure "Target velocity" in "Process data (PDO configuration)" on the FA side.
3049	0BE9	Unconfigured PDO object 6071 for target torque	1. When the control mode of the axis operated by this instruction was MC_CST, "Target torque" was not configured in "Process data (PDO configuration)" on the FA side.	1. Configure "Target torque" in "Process data (PDO configuration)" on the FA side.
3050	0BEA	Unconfigured PDO object 6060 for modes of operation	1. "Modes of operation" was not configured in "Process data (PDO configuration)" on the FA side.	1. Configure "Modes of operation" in "Process data (PDO configuration)" on the FA side.
3051	0BEB	Unconfigured PDO object 6061 for modes of operation display	1. "Modes of operation display" was not configured in "Process data (PDO configuration)" on the FA side.	1. Configure "Modes of operation display" in "Process data (PDO configuration)" on the FA side.
3052	0BEC	Unconfigured PDO object 60B8 for touch probe function	1. "Touch probe function" was not configured in "Process data (PDO configuration)" on the FA side.	1. Configure "Touch probe function" in "Process data (PDO configuration)" on the FA side.
3053	0BED	Unconfigured PDO object 60B9 for touch probe status	1. "Touch probe status" was not configured in "Process data (PDO configuration)" on the FA side.	1. Configure "Touch probe status" in "Process data (PDO configuration)" on the FA side.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
3054	0BEE	Unconfigured PDO object 60BA for touch probe pos1 pos value	1. "Touch probe pos1 pos value" was not configured in "Process data (PDO configuration)" on the FA side.	1. Configure "Touch probe pos1 pos value" in "Process data (PDO configuration)" on the FA side.
3055	0BEF	Unconfigured PDO object 60BB for touch probe pos1 neg value	1. "Touch probe pos1 neg value" was not configured in "Process data (PDO configuration)" on the FA side.	1. Configure "Touch probe pos1 neg value" in "Process data (PDO configuration)" on the FA side.
3056	0BF0	Unconfigured PDO object 60BC for touch probe pos2 pos value	1. "Touch probe pos2 pos value" was not configured in "Process data (PDO configuration)" on the FA side.	1. Configure "Touch probe pos2 pos value" in "Process data (PDO configuration)" on the FA side.
3057	0BF1	Unconfigured PDO object 60BD for touch probe pos2 neg value	1. "Touch probe pos2 neg value" was not configured in "Process data (PDO configuration)" on the FA side.	1. Configure "Touch probe pos2 neg value" in "Process data (PDO configuration)" on the FA side.
3058	0BF2	Unconfigured PDO object 6077 for torque actual value	1. "Torque actual value" was not configured in "Process data (PDO configuration)" on the FA side.	1. Configure "Torque actual value" in "Process data (PDO configuration)" on the FA side.
3059	0BF3	Unconfigured PDO object 60FD for digital inputs	1. "Digital inputs" was not configured in "Process data (PDO configuration)" on the FA side.	1. Configure "Digital inputs" in "Process data (PDO configuration)" on the FA side.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
3060	0BF4	Unconfigured PDO object 60E0 for positive torque limit value	1. "Positive torque limit value" was not configured in "Process data (PDO configuration)" on the FA side.	1. Configure "Positive torque limit value" in "Process data (PDO configuration)" on the FA side.
3061	0BF5	Unconfigured PDO object 60E1 for negative torque limit value	1. "Negative torque limit value" was not configured in "Process data (PDO configuration)" on the FA side.	1. Configure "Negative torque limit value" in "Process data (PDO configuration)" on the FA side.
3062	0BF6	SDO operation failed	1. The communication was abnormal.	1. Check if the communication status of the slave is normal.
3063	0BF7	The MC_MoveSuperImposed instruction cannot be called before the main motion instruction.	1. Before triggering the main motion instruction, the MC_MoveSuperImposed instruction was located above the motion instruction in the program logic and was already triggered.	1. Place and trigger the MC_MoveSuperImposed instruction only after the main motion instruction in the program logic. The MC_MoveSuperImposed instruction must be located below the motion instruction.
3064	0BF8	Multiple instances of the instruction are not allowed to run simultaneously while the current instruction is executing.	1. When an instance of the instruction function block was already running, another instance of the same instruction was triggered again.	1. Do not trigger additional instances of the same instruction while an instance of the instruction function block is running.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
3065	0BF9	Homing mode re-trigger error	1. The instruction was re-triggered during its execution.	1. During the instruction execution, do not re-trigger the same instruction.
3067	0BFB	The action upon contacting the positive limit switch during homing is set to immediate fault.	1. When "Positive limit mode" in the "Homing parameter settings" was set to "No reverse/Axis fault stop" on the FA side, the axis contacted the positive limit switch during homing.	1. Set the homing motion parameters correctly. 2. Set the limit switch positions correctly.
3068	0BFC	The action upon contacting the negative limit switch during homing is set to immediate fault.	1. When "Negative limit mode" in the "Homing parameter settings" was set to "No reverse/Axis fault stop" on the FA side, the axis contacted the negative limit switch during homing.	1. Set the homing motion parameters correctly. 2. Set the limit switch positions correctly.
3069	0BFD	Failed to find origin after triggering the positive limit switch twice during homing	1. The positive limit switch was not installed or was installed in the incorrect position.	1. Check the status of the positive and negative limit switches.
3070	0BFE	Failed to find origin after triggering the negative limit switch twice during homing	1. The negative limit switch was not installed or was installed in the incorrect position.	1. Check the status of the positive and negative limit switches.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
3076	0C04	During initial check in "Limit Switch Only" homing mode, both positive and negative limit switches are active simultaneously.	1. Both positive and negative limits were activated simultaneously.	1. Check the status of the positive and negative limit switches.
3077	0C05	Host controller homing timeout	1. The actual homing time from the host controller exceeded the set value of "HomeTimeOut" in the structured user input parameter "HomingParameter" on the instruction side.	1. Set the value of "HomeTimeOut" in the structured user input parameter "HomingParameter" on the instruction side to 0 (Ignore the time limit at this time). 2. Increase the set value of "HomeTimeOut" in the structured user input parameter "HomingParameter" on the instruction side.
3080	0C08	Triggering the zero point switch during deceleration within the home switch masking distance	1. When the host controller executed the homing instruction, the zero point switch was triggered during the deceleration phase within the home switch masking distance.	1. Check the status of the zero point switch.
3081	0C09	Position cannot be set during superimposed motion.	1. During the execution of the MC_MoveSuperImposed and MC_HaltSuperImposed instructions, the MC_SetPosition instruction was triggered.	1. Do not trigger the MC_SetPosition instruction during the execution of the MC_MoveSuperImposed and MC_HaltSuperImposed instructions.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
3082	0C0A	Axis state condition not met during the host controller homing process	1. The instruction cannot be triggered because the current axis state is not AXIS_STANDSTILL.	1. Before triggering this instruction, ensure the axis state is AXIS_STANDSTILL.
3083	0C0B	Control mode not met during the host controller homing process	1. The current axis control mode was not MC_CSP. This instruction can only be triggered in the above axis control mode.	1. Modify the current axis control mode.
3084	0C0C	Home signal not found	1. In homing modes HOMING_METHOD_PASSHOMESWHOMESWTURN or HOME_METHOD_INHOMESWHOMESWTURN, if the start direction does not match the end direction, and both the positive and negative limit switches have been triggered once, the system concludes that no valid home signal was detected.	1. Set the zero point signal, positive and negative limit switches, and home signal correctly.
3085	0C0D	Positive limit triggered during low-speed creeping or compensation motion	1. The positive limit was triggered during low-speed creeping or compensation motion.	1. Modify the set value of "Positive limit" in "Software limit" on the FA side. 2. Modify the set value of "Compensation" in the structured user input parameter "HomingParameter" on the instruction side.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
3086	0C0E	Negative limit triggered during low-speed creeping or compensation motion	1. The negative limit was triggered during low-speed creeping or compensation motion.	1. Modify the set value of "Negative limit" in "Software limit" on the FA side. 2. Modify the set value of "Compensation" in the structured user input parameter "HomingParameter" on the instruction side.
3087	0C0F	The instruction was re-triggered during execution.	1. The current instruction was re-triggered while it is running. 2. Another instance of the current instruction function block was triggered while the instruction was running.	1. Do not re-trigger the instruction during execution. 2. Do not re-trigger another instance of the instruction function block during execution.
3088	0C10	Input PDO parameters not configured	1. The PDO corresponding to the set value of the input parameter "ParameterNumber" on the instruction side was not configured in "EtherCAT I/O mapping" on the FA side.	1. In "EtherCAT I/O mapping" on the FA side, configure the PDO parameters corresponding to the set value of the input parameter "ParameterNumber" on the instruction side.
3089	0C11	The instruction does not support superimposed motion.	1. The MC_MoveSuperImposed instruction was triggered while the main instruction is executing.	1. Modify the user program to avoid triggering the MC_MoveSuperImposed instruction while the main instruction is executing.
3091	0C13	An axis already assigned to a group. Operation not permitted.	1. The axis variable operated by the instruction was already assigned to an axis group. 2. Using the MC_ChangeGearingRatio instruction, the axis in the group was switched to rotation mode.	1. Modify the user program. 2. Modify the axis group configuration on the FA side to remove the axis variable operated by the instruction from the axis group. 3. Do not use the MC_ChangeGearingRatio instruction to switch the axis in the axis group to rotation mode.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
3092	0C14	Cannot set position of the master axis during synchronized motion	1. The axis targeted for position preset was the master axis during synchronized motion.	1. Do not set the master axis to a position during synchronized motion. Re-trigger the instruction only after the axis completed the synchronized motion.
3093	0C15	Home switch dwell time too short to reach creeping speed	1. After the home switch signal was triggered, the actual homing time exceeded the set value of "PushTime" in the structured user input parameter "HomingParameter" on the instruction side, while the actual speed has not yet decelerated to the value set for "ApproachVel" in "HomingParameter".	1. Modify the set values of "Vel", "ApproachVel", "Acc", "Dec", "Jerk", and "PushTime" in the structured user input parameter "HomingParameter" on the instruction side.
3094	0C16	The instruction does not support superimposed motion during execution.	1. The MC_MoveSuperImposed instruction was triggered when the MC_Halt, MC_GearOut, or MC_CamOut instruction was running.	1. Modify the user program to ensure that the MC_MoveSuperImposed instruction is not triggered when the MC_Halt, MC_GearOut, or MC_CamOut instruction is running.
3095	0C17	The MC_Inch /MC_Jog instruction cannot be called when the axis is in the STOP state.	1. When the current axis status was AXIS_STOPPING, the instruction was triggered.	1. Do not trigger the instruction when the axis status is AXIS_STOPPING.
3096	0C18	Pulse axes are not supported.	1. This instruction did not support pulse axes.	1. Select the appropriate axis to execute the instruction based on requirements.
3097	0C19	Only servo axes are supported.	1. The instruction only supported servo axes.	1. Use a servo axis to execute the instruction.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
3098	0C1A	The axis is in a faulty state when homing is triggered.	1. When the MC_Home instruction is initiated, the axis is in a faulty state.	1. Check if the axis type is a module axis. If so, verify whether its limit switch signals are active-low. If they are, adjust the trigger level of the limit switch signals as required.
3099	0C1B	Incorrect backlash compensation mode	1. The value of the input parameter "Mode" in the MC_BacklashCompensate instruction was out of the valid range.	1. The value of the input parameter "Mode" in the MC_BacklashCompensate instruction must be 0.
3100	0C1C	Incorrect backlash compensation velocity parameter	1. The value of the input parameter "BacklashVelocity" in the MC_BacklashCompensate instruction was out of the valid range.	1. The value of the input parameter "BacklashVelocity" in the MC_BacklashCompensate instruction must be greater than or equal to 0 and less than or equal to the maximum velocity specified by the software.
3101	0C1D	Incorrect parameter for lead screw pitch compensation mode	1. The "Mode" value of the structured input parameter "ScrewPara" in the MC_LeadScrewCompensate instruction was out of the valid range.	1. The "Mode" value of the structured input parameter "ScrewPara" in the MC_LeadScrewCompensate instruction must be 0 or 1. (Currently it only supports 0, linear compensation)
3102	0C1E	The first compensation value in the negative lead screw compensation table must be 0.	1. For the MC_LeadScrewCompensate instruction, the first compensation value in the input "crewCompNegTable" was not set to 0.	1. For the MC_LeadScrewCompensate instruction, the first compensation value in the input parameter "crewCompNegTable" must be set to 0.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
3103	0C1F	The first compensation value in the positive lead screw pitch compensation table must be 0.	1. For the MC_LeadScrewCompensate instruction, the first compensation value in the input "crewCompPosTable" was not set to 0.	1. For the MC_LeadScrewCompensate instruction, the first compensation value in the input parameter "crewCompPosTable" must be set to 0.
3104	0C20	Start and end positions within the screw pitch compensation parameter structure cannot be identical.	1. "StartPos" and "EndPos" in the structured input parameter "ScrewPara" of the MC_LeadScrewCompensate instruction must not be the same.	1. Modify "StartPos" and "EndPos" in the structured input parameter "ScrewPara" of the MC_LeadScrewCompensate instruction.
3105	0C21	The backlash compensation value is less than 0.	1. The value of the input parameter "Backlash" in the MC_BacklashCompensate instruction was not within the valid range.	1. The value of the input parameter "Backlash" in the MC_BacklashCompensate instruction must be greater than 0.
3106	0C22	Start position for pitch compensation on rotary axis exceeds limit.	1. The value "StartPos" in the structured input parameter "ScrewPara" of the MC_LeadScrewCompensate instruction was less than 0.	1. The value "StartPos" in the structured input parameter "ScrewPara" of the MC_LeadScrewCompensate instruction must be greater than 0.
3107	0C23	End position for pitch compensation on rotary axis exceeds limit.	1. The value "EndPos" in the structured input parameter "ScrewPara" of the MC_LeadScrewCompensate instruction was less than 0.	1. The value "EndPos" in the structured input parameter "ScrewPara" of the MC_LeadScrewCompensate instruction must be greater than 0.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
3108	0C24	The MC_SetPosition instruction is not allowed while the axis is in lead screw pitch compensation.	1. The MC_SetPosition instruction is not allowed while the axis is in lead screw pitch compensation.	1. The MC_SetPosition instruction is not allowed while the axis is in lead screw pitch compensation.
3109	0C25	Backlash compensation reset not complete	1. The MC_BacklashCompensate instruction was re-triggered while the compensation values were being cleared after an error reset.	1. Do not re-trigger the MC_BacklashCompensate instruction during the compensation clearance process that follows an error reset. The instruction can only be triggered again after all the compensation values have been fully cleared.
3110	0C26	Lead screw pitch compensation not complete	1. The MC_LeadScrewCompensate instruction was re-triggered while the compensation values were being cleared after an error reset.	1. Do not re-trigger the MC_LeadScrewCompensate instruction during the compensation clearance process that follows an error reset. The instruction can only be triggered again after all the compensation values have been fully cleared.
3111	0C27	Lead screw pitch compensation is in progress.	1. Another instance of the MC_LeadScrewCompensate instruction was already running when the current one was triggered.	1. Do not trigger another MC_LeadScrewCompensate instruction when an MC_LeadScrewCompensate instruction is already executing.
3112	0C28	Backlash compensation is in progress.	1. Another instance of the MC_BacklashCompensate instruction was already running when the current one was triggered.	1. Do not trigger another MC_BacklashCompensate instruction when an MC_BacklashCompensate instruction is already executing.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
3113	0C29	Homing position out of range	1. The pulse value corresponding to the input parameter "Position" of the MC_Home instruction has exceeded the value range of PDO object 607C [-2147483648~2147483647].	1. Set an appropriate value for the input parameter "Position" in MC_Home so that its pulse value is within the value range of PDO object 607C [-2147483648~2147483647].
3114	0C2A	Vibration shaping mode out of range	1. The value of the input parameter "Mode" in the MC_VibrationShapping instruction must be set to either 0 (ZV) or 1 (ZVD).	1. Select an appropriate vibration shaping mode.
3115	0C2B	Vibration shaping frequency parameter error	1. The set value of the input parameter "Frequence" in the MC_VibrationShapping instruction was less than or equal to 0.	1. The set value of the input parameter "Frequence" in the MC_VibrationShapping instruction must be greater than 0.
3116	0C2C	Vibration shaping damping ratio parameter error	1. The set value of the input parameter "DampingRatio" in the MC_VibrationShapping instruction was less than or equal to 0, or greater than or equal to 1.	1. The set value of the input parameter "DampingRatio" in the MC_VibrationShapping instruction must be greater than 0 and less than 1.
3117	0C2D	Insufficient memory for vibration shaping	1. There is insufficient memory in the PLC to support the execution of the MC_VibrationShapping instruction.	1. Release the memory in the PLC to support the execution of the MC_VibrationShapping instruction.
3118	0C2E	Vibration shaping is not allowed during homing or while the axis is stopping.	1. The MC_VibrationShapping instruction was triggered when the axis was in the AXIS_HOMING or AXIS_STOPPING state.	1. Trigger the MC_VibrationShapping instruction when the axis is already in the AXIS_STANDSTILL state.
3119	0C2F	Position cannot be set during vibration shaping.	1. The MC_SetPosition instruction was triggered while MC_VibrationShapping was running.	1. Do not trigger the MC_SetPosition instruction while MC_VibrationShapping is running.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
3120	0C30	Internal error due to too low vibration shaping frequency parameter	1. The input parameter "Frequency" of the MC_VibrationShapping instruction was set so low that the length of the buffer data was out of range.	1. Set an appropriate value for the input parameter "Frequency" of the MC_VibrationShapping instruction as required.
3121	0C31	Vibration shaping is in progress.	1. Another instance of the MC_VibrationShapping instruction was already running, when the current one was triggered.	1. Do not trigger another MC_VibrationShapping instruction when an MC_VibrationShapping instruction is already running.
3122	0C32	The instruction is not allowed for the axis type and encoder configuration.	1. An MC_SetPosition, MC_Homing, or MC_HomingWithParameter instruction was used on a rotary axis whose absolute encoder was configured for position restoration on the PLC side.	1. Do not use the MC_SetPosition, MC_Homing, or MC_HomingWithParameter instruction on a rotary axis whose absolute encoder is configured for position restoration on the PLC side.
3123	0C33	The current control mode does not match the executed instruction.	1. The executed instruction did not match the current control mode.	1. When the current control mode does not match the executed instruction, use MC_SetControllerMode to switch the axis to the appropriate mode.
3124	0C34	Re-triggering is not allowed.	1. Another instance of the instruction was already executing when an instance of the same instruction was called again.	1. Do not call the instance of the same instruction again when another instance of the instruction was already executing.
3125	0C35	The homing instructions for host controller cannot be used with absolute encoders in servo-side retention mode.	The MC_Homing or MC_HomingWithParameter instruction was issued to an axis whose absolute encoder was configured for position retention on the servo side.	Do not use the MC_Homing or MC_HomingWithParameter instruction to an axis whose absolute encoder was configured for position retention on the servo side.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
3126	0C36	Unsupported instruction	The current PLC device does not support this instruction.	Modify the user program to remove this instruction.
3127	0C37	Axis type error	MC_MoveModulo only supports local pulse axes and bus servo axes, and the axes must be in rotary mode.	Change the motion type of the control axis to rotary mode.
3128	0C38	ModuloWindow parameter out of range	The ModuloWindow input for the MC_MoveModulo instruction is outside the allowed range.	Set the ModuloWindow input to a value greater than 0 and less than or equal to half the rotation period of the input axis.
3129	0C39	Exceeded in-position check time	Within the in-position check time, the axis tracking error exceeds the in-position width.	1. Increase the in-position width or in-position check time. 2. Adjust the drive's three-loop gain.
4000	0FA0	The slave axis position cannot be modified during electronic gear engagement.	1. The instruction was re-triggered during the execution of the synchronization instruction.	1. Modify the user program to avoid re-triggering the instruction during the execution of the synchronization instruction.
4001	0FA1	The instruction cannot be executed while the master axis is in homing mode.	1. The master axis was executing a homing instruction when the instruction was triggered.	1. Do not trigger the homing instruction when the master axis is executing.
4002	0FA2	Incorrect axis planning configuration	1. When the MC_CamOut instruction was triggered, the slave axis controlled by the instruction was not in cam motion. 2. When the MC_GearOut instruction was triggered, the slave axis controlled by the instruction was not in gear motion.	1. When the MC_CamOut instruction is triggered, the slave axis controlled by the instruction must be in cam motion. 2. When the MC_GearOut instruction is triggered, the slave axis controlled by the instruction must be in gear motion.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
4003	0FA3	The cam table node pointer is null.	1. The input/output parameter CamNote of the instruction referenced a manually created cam table whose node pointer was null.	1. Before using a manually created cam table, a value must be assigned to the cam table node pointer.
4004	0FA4	OUTMODE cannot be switched during the instruction execution.	1. The MC_CamOut and MC_GearOut instructions were re-triggered after the modification of the input parameter "OutMode" while the instructions were executing.	1. Do not re-trigger the instruction after modifying the input parameter "OutMode", while the instruction is executing.
4005	0FA5	FileName file name error	1. The file name of the cam table contained an absolute path.	1. Check if the file name of the cam table contains an absolute path; if so, rename the cam table file.
4007	0FA7	MC_GearInPos cannot be re-triggered	1. The MC_GearInPos instruction was re-triggered while it was executing.	1. Do not re-trigger the MC_GearInPos instruction while it is executing.
4009	0FA9	The phase or displacement of the first point is not 0.	1. The phase or displacement of the first point in the input/output parameter "CamTable" of the MC_GenerateCamTable instruction was not 0.	1. Modify the value of the first point in the input/output parameter "CamTable" of the MC_GenerateCamTable instruction.
4010	0FAA	The number of dense nodes is less than the minimum 2.	1. When the cam table was configured to use dense nodes, the number of dense nodes was less than 2.	1. When the cam table is configured to use dense nodes, the minimum number of dense nodes must be 2.
4011	0FAB	The number of dense nodes exceeds the maximum 65535.	1. When the cam table was configured to use dense nodes, the number of dense nodes was greater than 65535.	1. When the cam table is configured to use dense nodes, the maximum number of dense nodes must be 65535.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
4012	0FAC	Phases of dense nodes are not in ascending order.	1. The phase points in dense node type in the cam table were not in ascending order.	1. The phases of dense nodes in the cam table must be in ascending order.
4013	0FAD	Phase overlap of dense nodes in the cam table, threshold 0.0001	1. The interval between adjacent phases of dense nodes in the cam table was less than 0.001.	1. The interval between adjacent phases of dense nodes in the cam table must be greater than or equal to 0.001.
4014	0FAE	The phase or displacement of the first point is not 0.	1. The phase or displacement of the first point in the cam table was not 0.	1. The phase and displacement of the first point in the cam table must be 0.
4015	0FAF	The number of key nodes is less than the minimum 2	1. When the cam table was configured to use key nodes, the number of key nodes was less than 2.	1. When the cam table is configured to use key nodes, the minimum number of key nodes must be 2.
4016	0FB0	The number of key nodes exceeds the maximum 20000.	1. When the cam table was configured to use key nodes, the number of key nodes was greater than 20000.	1. When the cam table is configured to use key nodes, the maximum number of key nodes must be 20000.
4017	0FB1	Phases of key nodes are not in ascending order.	1. In the MC_GetCamTableDistance or MC_GetCamTablePhase instruction, the phase of the input variable "StartPoint" was greater than that of the input variable "EndPoint". 2. In the input parameter CamNodes of the MC_GenerateCamTable instruction, the phases of key nodes are not in ascending order.	1. In the MC_GetCamTableDistance or MC_GetCamTablePhase instruction, the phase of the input variable "StartPoint" must be less than that of the input variable "EndPoint". 2. In the input variable CamNodes of the MC_GenerateCamTable instruction, the phases of key nodes must be in ascending order.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
4018	0FB2	Phase overlap of key nodes in the cam table, threshold 0.0001	1. In the MC_GetCamTableDistance or MC_GetCamTablePhase instruction, the phase difference of the input variables "StartPoint" and "EndPoint" was less than 0.0001. 2. The cam table checked by the MC_GenerateCamTable instruction was invalid, and the phase difference between two points in the cam table was less than 0.0001.	1. In the MC_GetCamTableDistance or MC_GetCamTablePhase instruction, the phase of the input variable "StartPoint" cannot coincide with that of the input variable "EndPoint". 2. Check if the phase in the cam is valid and adjust the phase difference in the cam table.
4019	0FB3	The phase, displacement, speed ratio, acceleration, and jerk exceed the maximum limit (9,999,999).	1. One or more parameters (including phase, displacement, speed ratio, acceleration, and jerk) exceeded the maximum limit (9,999,999).	1. Adjust the cam table data as required.
4020	0FB4	The cam table is in read-write or update status.	1. When the instruction was started, the cam table was in the process of being read or updated.	1. Wait until the read-write or update instruction has been completed for the current cam table. Then, execute the instruction.
4021	0FB5	The number of nodes configured in the system is less than the input variable "NodeNum" of the instruction.	1. The number of the cam table nodes configured in the system was less than the value of the input variable "NodeNum" in the MC_GenerateCamTable instruction.	1. The number of the cam table nodes configured in the system must be greater than or equal to the value of the input variable "NodeNum" in the MC_GenerateCamTable instruction.
4022	0FB6	Updating dense node data is not supported.	1. When the cam table was configured to use dense nodes, the dense node data cannot be updated.	1. When the cam table was configured to use dense nodes, the dense node data cannot be updated.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
4023	0FB7	Cam table data is not checked.	1. A user-defined cam table was imported without any check.	1. If a cam table is defined by the user, execute the MC_GenerateCamTable instruction again after using the MC_CheckCamTable instruction to check the validity of this cam table.
4024	0FB8	Failed to create the cam table file	1. The system failed to create the cam table file.	1. Check if there are any special characters in the input file name.
4027	0FBB	The file read is not a cam table file.	1. The file read by the MC_ReadCAM instruction is not a cam table file.	1. Check if the file type read by the MC_ReadCAM instruction is correct. If not, use the MC_ReadCAM instruction to read the correct cam table file as required.
4028	0FBC	The number of the cam table nodes on the FA side is less than that the number of nodes in the file.	1. The number of the cam table nodes on the FA side was less than that the number of nodes in the file.	1. The number of the cam table nodes on the FA side must be greater than or equal to the number of nodes in the file.
4029	0FBD	The slave axis speed exceeds the software limit speed.	1. The speed of the slave axis to execute the instruction was greater than the maximum axis speed specified by the software.	1. The speed of the slave axis to execute the instruction must be set according to the requirements and the maximum speed specified by the software.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
4030	0FBE	Master axis data type error	1. The value set for the input variable "ReferenceTypeM1" or "ReferenceTypeM2" in the MC_CombineAxes instruction was invalid. 2. The value set for the input variable "ReferenceType" in the MC_AxesObserve instruction was invalid. 3. The value set for the input variable "MasterValueSource" was invalid.	1. The input variable "ReferenceTypeM1" or "ReferenceTypeM2" in the MC_CombineAxes instruction has the following value range: 0 (value calculated in the previous task period (fSavePosition)), 1 (value obtained in the same task period (fActPosition)), 2 (value calculated in the same task period (fSetPosition)). Select the appropriate position source type as required. 2. The input variable "ReferenceType" in the MC_AxesObserve instruction has the following value range: 0 (value calculated in the previous task period (fSavePosition)), 1 (value obtained in the same task period (fActPosition)), 2 (value calculated in the same task period (fSetPosition)). Select the appropriate position source type as required. 3. The input variable "MasterValueSource" has the following value range: 0 (value calculated in the previous task period (fSavePosition)), 1 (value obtained in the same task period (fActPosition)), 2 (value calculated in the same task period (fSetPosition)). Select the appropriate position source type as required.
4031	0FBF	Invalid parameter settings for master or slave axis	1. The input parameter InverseFlag in the MC_GearInPos instruction was set FALSE to disallow the reverse motion, but the master or slave axis parameters were set so that achieving the gear ratio at the specified synchronization position would require a reverse motion.	1. According to the requirements, adjust the input parameter InverseFlag of the MC_GearInPos instruction to allow reverse motion (TRUE), or modify the relevant parameters of the master/slave axis.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
4032	0FC0	Slave axis reverse motion is disabled but is being forced to reverse by the master axis.	1. The input parameter InverseFlag in the MC_GearInPos instruction was set FALSE to disallow the reverse motion, but the slave axis was being forced to reverse by the master axis.	1. According to the requirements, adjust the input parameter InverseFlag of the MC_GearInPos instruction to allow reverse motion (TRUE), or modify the relevant parameters of the master/slave axis.
4033	0FC1	Error when opening cam table file	1. The cam table file to be opened did not exist.	1. Check if the file name "FileName" of the input variable in the MC_ReadCAM instruction matches that of the cam table to be read. If not, enter the correct "FileName".
4034	0FC2	The cam table node type on the FA side does not match the node type in the file.	1. The cam table node type on the FA side did not match the node type in the file.	1. Check if the cam table node type on the FA side matches the node type in the cam table file to be read. If not, correct the cam table node type on the FA side or the node type in the cam table file to be read.
4036	0FC4	Incorrect the cam table node type	1. The cam table node type corresponding to the MC_CAMBounds instruction was incorrect. 2. The cam table node type was out of range. Dense nodes or key nodes must be defined for the node type.	1. The node type in the cam table corresponding to the MC_CAMBounds instruction must be set to key nodes. 2. Check the cam table node type and select the appropriate node type as required.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
4037	0FC5	OUTMODE mode out of range	1. The value of the input variable "OutMode" in the MC_GearOut instruction was out of range. 2. The value of the input variable "OutMode" in the MC_CamOut instruction was out of range.	1. The input variable "OutMode" in the MC_GearOut instruction has the following value range: 0 (decelerate to stop), 1 (maintain constant velocity before synchronization end). Set the appropriate synchronization end mode according to the requirements. 2. The input variable "OutMode" in the MC_CamOut instruction has the following value range: 0 (decelerate to stop), 1 (continue at the previous speed), 2 (Immediate stop (emergency stop) after performing the current cam table). Set the appropriate synchronization end mode as required.
4038	0FC6	The master and slave axes cannot be identical.	1. When the instruction was triggered, the master and slave axes are configured as the same axis.	1. Check if the master and slave axes in this instruction are configured as the same axis. If so, reconfigure the master and slave axes as required.
4039	0FC7	Error triggered by MC_DigitalCamSwitch due to ended cam cycle	1. When the input parameter "Reference Type" of the MC_DigitalCamSwitch instruction was set to mcMasterPhase (3, phase of the master axis when this axis is configured as a cam slave axis), the synchronous motion terminated during the execution of the instruction.	1. Check if the synchronous motion has been interrupted or if there are any error messages from the slave axis.
4041	0FC9	Periodic parameter input exceeds specified range.	1. The value set for the input variable "Periodic" in the MC_CamIn was invalid.	1. The value of the input variable "Periodic" in the MC_CamIn instruction must be greater than or equal to 0.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
4042	0FCA	Slave axis displacement exceeds the maximum specification limit.	1. The value set for the input variable "SlaveDistance" for the MC_GetCamTablePhase instruction was invalid.	1. Set a valid value for the input variable "SlaveDistance" according to the requirements and specification limits.
4043	0FCB	The curve fitting type is out of range specified.	1. The fitting curve type of the input variable "StartPoint" or "EndPoint" in the MC_GetCamTableDistance instruction was outside the valid range. 2. The fitting curve type of the input variable "StartPoint" or "EndPoint" in the MC_GetCamTablePhase instruction was outside the valid range.	1. The fitting curve type of the input variables "StartPoint" and "EndPoint" in the MC_GetCamTableDistance instruction has the following value range. 1: Linear curve; 2: Parabolic curve; 3: Cubic curve; 4: Quintic curve; 5: Septic curve; 6: Cosine acceleration curve; 7: Sine acceleration curve; 8: Modified T-type acceleration curve. 2. The fitting curve type of the input variables "StartPoint" and "EndPoint" in the MC_GetCamTablePhase instruction has the following value range. 1: Linear curve; 2: Parabolic curve; 3: Cubic curve; 4: Quintic curve.
4044	0FCC	Invalid value for phase parameter	1. The value set for the input variable "Phase" for the MC_GetCamTableDistance instruction was invalid.	1. Check if the value of the input variable "Phase" is less than the phase value of the input variable "StartPoint" or greater than the phase value of the input variable "EndPoint". If so, set a valid phase value as required.
4046	0FCE	Multiple switches with the same position	1. Multiple switches were configured at the same position.	1. The switches with the identical position are not allowed in the input variable "Switches" of the MC_DigitalCamSwitch instruction.
4047	0FCF	Multiple switches with the same parameter	1. Multiple switches were configured with the same parameter.	1. The switches with the identical parameter are not allowed in the input variable "Switches" of the MC_DigitalCamSwitch instruction.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
4048	0FD0	The parameter "fParameter" is out of range when the tappet switch mode is set to 2.	1. When the switch mode "iMode" of the input variable "Switches" for the MC_DigitalCamSwitch instruction was set to 2, the value set for the parameter "fParameter" of the input variable "Switches" was invalid.	1. When the switch mode "iMode" of the input variable "Switches" for the MC_DigitalCamSwitch instruction was set to 2, the value of "fParameter" in "Switches" must be less than or equal to 10000 and greater than 0.
4049	0FD1	The number of master axis jump points exceeds the limit.	1. In a cam table with only two nodes, the cam table is executed within a single task cycle.	1. Increase the cam table phase length or decrease the master axis running velocity.
4050	0FD2	The specified parameter for the start position method is out of range.	1. The value set for the input variable "StartMode" in the MC_CamIn instruction was invalid.	1. The input variable "StartMode" in the MC_CamIn instruction has the following value range. 0: Absolute value; 1: Relative value; 2: Immediate start. Select the appropriate "StartMode" as required.
4051	0FD3	The tappet switch mode parameter is out of range.	1. The value set for the switch mode "iMode" in the input variable "Switches" of the MC_DigitalCamSwitch instruction was invalid.	1. The switch mode "iMode" in the input variable "Switches" of the MC_DigitalCamSwitch instruction has the following value range. 0: Disabled; 1: Position type; 2: Time type. Select the appropriate switch mode as required.
4052	0FD4	The master axis direction parameter is out of range.	1. The value for the master axis direction selection "fDirection" in the input variable "Switches" of the MC_DigitalCamSwitch instruction was invalid.	1. "fDirection" in the input variable "Switches" of the MC_DigitalCamSwitch instruction has the following value range: 0 (Positive); 1 (Negative); 2 (No direction). Select the appropriate master axis direction as required.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
4054	0FD6	A file with the same name is being written.	1. When the current MC_WriteCAM instruction was started, the current cam table was already being written.	1. Wait until the ongoing write instruction has been completed for the current cam table. Then, execute the MC_WriteCAM instruction.
4055	0FD7	The MC_Phasing or MC_PhasingEX instruction are executed in single axis mode or during the execution of the MC_CombineAxes instruction.	1. The MC_Phasing or MC_PhasingEx instruction can only be started during synchronization instruction execution.	1. Start the MC_Phasing or MC_PhasingEx instruction after the appropriate synchronization control instruction as required.
4056	0FD8	Simultaneous execution of multiple instructions is not permitted during instruction planning.	1. Different instances of the current instruction function block were executed while the instruction was being executed.	1. Wait until the instruction has been completed. Then, execute the current MC_Phasing or MC_PhasingEx instruction again. 2. Stop the ongoing phase compensation using the MC_StopPhasing instruction. Then, execute the current MC_Phasing or MC_PhasingEx instruction.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
4057	0FD9	Invalid parameter settings for master axis position and motion direction	1. For a linear axis and StartMode set to 0 (Absolute value), the value of the master axis position was less than that of the cam table start position while the motion direction was set to forward, or the value of the master axis position was greater than that of the cam table start position while the motion direction was set to reverse. For a linear axis and StartMode set to 1 (Relative value), the value of the master axis position was less than 0 while the motion direction was set to forward, or the value of the master axis position was greater than 0 while the motion direction was set to reverse.	1. Set the valid input variables "MasterStartDistance" and "Direction" in the MC_CamIn instruction according to the requirements and the start position method.
4058	0FDA	The maximum input speed is less than or equal to 0	1. The value set for the input variable "MaxVel" in the MC_CAMBounds instruction was invalid.	1. Set a valid value for the input variable "MaxVel" in the MC_CAMBounds instruction as required.
4059	0FDB	The maximum input acceleration is less than or equal to 0	1. The value set for the input variable "MaxAccDec" in the MC_CAMBounds instruction was invalid.	1. Set a valid value for the input variable "MaxAccDec" in the MC_CAMBounds instruction as required.
4060	0FDC	The position type selection of the cam switch does match the associated axis.	1. When the input variable "ReferenceType" in the MC_DigitalCamSwitch instruction was set to MCAXISMASTERPHASE, the axis was not configured as a cam slave axis.	1. When the input variable "ReferenceType" in the MC_DigitalCamSwitch instruction is set to MCAXISMASTERPHASE, the axis must be set as a cam slave axis using the MC_CamIn instruction.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
4061	0FDD	FileName cannot contain Chinese characters.	1. The cam table file name contained Chinese characters.	1. Check if the file name of the cam table contains Chinese characters. If yes, modify the file name as required.
4062	0FDE	FileName is null.	1. The cam table file name was null.	1. The naming convention for the cam table file is 'FileName'. Name the cam table file according to this convention as required.
4063	0FDF	All cam cycles have been completed, but the engagement coupling process is not yet finished.	1. The values of the input variables "Velocity" and "Acceleration" for the MC_CamIn instruction were too low. 2. The values of the input variables "Velocity" and "Acceleration" for the MC_CamInV2 instruction were too low.	1. Set the valid values for the input variables "Velocity" and "Acceleration" in the MC_CamIn instruction as required. 2. Set the valid values for the input variables "Velocity" and "Acceleration" in the MC_CamInV2 instruction as required.
4064	0FE0	Different instances of the instruction function block were re-triggered during instruction execution.	1. Do not re-trigger the current MC_StopPhasing instruction when MC_StopPhasing is executing.	1. Wait until the MC_StopPhasing instruction have been completed. Then, execute the current MC_StopPhasing instruction.
4065	0FE1	The MC_CombineAxes instruction was re-triggered.	1. The MC_CombineAxes instruction was retriggered while it was executing.	1. Do not re-trigger the current MC_CombineAxes instruction while it is executing.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
4066	0FE2	Switch number parameter out of range	1. The value set for the input variable "Number" of the MC_DigitalCamSwitch instruction was invalid.	1. Check if the value for the input variable "Number" of the MC_DigitalCamSwitch instruction is greater than 64 or less than or equal to 0. If yes, set the switch number properly as required.
4069	0FE5	The master axis is disabled during synchronization or engagement.	1. The master axis that has executed the synchronous motion instruction was disabled during the execution of the instruction.	1. Before other operations, the master axis that performs the cam action must be switched from the AXIS_POWEROFF state to the AXIS_STANDSTILL state using MC_Power.
4071	0FE7	Start position equals end position in the cam switch.	1. In the input variable "Switches" of the MC_DigitalCamSwitch instruction, the "fPosition" value was equal to the "fParameter" value.	1. In the input variable "Switches" of the MC_DigitalCamSwitch instruction, the "fPosition" value must not equal the "fParameter" value.
4073	0FE9	Invalid cam switch position parameters on a linear axis	1. When the axis to execute the MC_DigitalCamSwitch instruction was configured as a linear axis, the position parameter of the input variable "Switches" of MC_DigitalCamSwitch was set incorrectly.	1. Check if the axis type is configured correctly for the MC_DigitalCamSwitch instruction according to the requirements. If not, select an appropriate axis type under "PLC -> Motion control -> Axis -> Axis to execute MC_DigitalCamSwitch -> General setting -> Basic setting". 2. In the input variable "Switches" of the MC_DigitalCamSwitch instruction, the "fPosition" value must be less than the "fParameter" value, when the axis to execute the MC_DigitalCamSwitch instruction is configured as a linear axis.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
4074	0FEA	Invalid cam switch position parameters on a rotary axis	1. When the axis to execute the MC_DigitalCamSwitch instruction was configured as a rotary axis, the position parameter of the input variable "Switches" of MC_DigitalCamSwitch was set incorrectly.	1. Check if the axis type is configured correctly for the MC_DigitalCamSwitch instruction according to the requirements. If not, select an appropriate axis type under "PLC -> Motion control -> Axis -> Axis to execute MC_DigitalCamSwitch -> General setting -> Basic setting". 2. In the input variable "Switches" of the MC_DigitalCamSwitch instruction, the "fPosition" value must be greater than 0, but less than or equal to the "fParameter" value, when the axis to execute the MC_DigitalCamSwitch instruction is configured as a rotary axis.
4076	0FEC	Permitted deviation value must be positive or 0.	1. The value set for the input variable "PermittedDeviation" of the MC_AxesObserve instruction was invalid.	1. The value set for the input variable "PermittedDeviation" of the MC_AxesObserve instruction must be greater than or equal to 0.
4077	0FED	The input parameter LinkMode of the MC_MoveLink instruction is out of range.	1. The input parameter LinkMode of the MC_MoveLink instruction was out of range.	1. The input parameter LinkMode of the MC_MoveLink instruction has the following range. 0: Immediate start; 1: Start when the master axis reaches the StartLinkPos position; 2: Active when the TriggerValue is detected to be TRUE.
4078	0FEE	The input parameter MasterDistance of the MC_MoveLink instruction is set to 0, but the SlaveDistance value is not 0.	1. The input parameter MasterDistance of the MC_MoveLink instruction was set to 0, but the SlaveDistance value was not 0.	1. Select the appropriate values for MasterDistance and SlaveDistance as required.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
4079	0FEF	The input parameters of the MC_MoveLink instruction cause the subsequent running distance of the master axis to be less than 0.	1. The values of the input parameters MasterDistance, MasterDisInSlaveAccMotion, MasterDisInSlaveDecMotion of the MC_MoveLink instruction were less than 0.	1. Select the appropriate values for MasterDistance, MasterDisInSlaveAccMotion, and MasterDisInSlaveDecMotion as required.
4080	0FF0	The value set for the input parameter MasterDistance of the MC_MoveLink instruction is invalid.	1. The value set for the input parameter MasterDistance of the MC_MoveLink instruction was less than 0.	1. The value set for the input parameter MasterDistance of the MC_MoveLink instruction must be greater than or equal to 0.
4081	0FF1	The value set for the input parameter MasterDisInSlaveAccMotion of the MC_MoveLink instruction is less than 0.	1. The value set for the input parameter MasterDisInSlaveAccMotion of the MC_MoveLink instruction was less than 0.	1. The value set for the input parameter MasterDisInSlaveAccMotion of the MC_MoveLink instruction must be greater than or equal to 0.
4082	0FF2	The value set for the input parameter MasterDisInSlaveDecMotion of the MC_MoveLink instruction is less than 0.	1. The value set for the input parameter MasterDisInSlaveDecMotion of the MC_MoveLink instruction was less than 0.	1. The value set for the input parameter MasterDisInSlaveDecMotion of the MC_MoveLink instruction must be greater than or equal to 0.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
4083	0FF3	The acceleration factor for the input parameter of the MC_PhasingEx instruction is less than 0 or greater than 1.	1. The acceleration factor set for the input parameter of the MC_PhasingEx instruction was less than 0 or greater than 1.	1. The acceleration factor set for the input parameter of the MC_PhasingEx instruction has the value range (0, 1).
4084	0FF4	The deceleration factor for the input parameter of the MC_PhasingEx instruction is less than 0 or greater than 1.	1. The deceleration factor set for the input parameter of the MC_PhasingEx instruction was less than 0 or greater than 1.	1. The deceleration factor set for the input parameter of the MC_PhasingEx instruction has the value range (0, 1).
4085	0FF5	The sum of the acceleration factor and deceleration factor for the input parameters in the MC_PhasingEx instruction is greater than 1.	1. The sum of the acceleration factor and deceleration factor for the input parameters in the MC_PhasingEx instruction was greater than 1.	1. The sum of the acceleration factor and deceleration factor for the input parameters in the MC_PhasingEx instruction has the value range (0, 1).

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
4086	0FF6	Switching axis objects is not permitted during instruction execution.	1. The axis object was switched during the execution of the instruction.	1. Do not switch the axis object during the execution of the instruction.
4087	0FF7	Invalid cam switch position parameters on a cam slave axis	1. When the axis to execute the MC_DigitalCamSwitch instruction is configured as a cam slave axis, the position parameter set for the input variable "Switches" of the MC_DigitalCamSwitch instruction was invalid.	1. In the input variable "Switches" in the MC_DigitalCamSwitch instruction, the "fPosition" and "fParameter" values must be greater than 0, but less than or equal to the master axis phase of the cam, when the axis to execute the MC_DigitalCamSwitch instruction is configured as a cam slave axis.
4088	0FF8	The array size for the cam table nodes is less than the node number.	1. The allocated array size for the cam table nodes was less than the node number.	1. The allocated array size for the cam table nodes must be greater than or equal to the node number.
4089	0FF9	The value for the MasterDistance input parameter of the MC_PhasingEx instruction is invalid.	1. The value set for the MasterDistance input parameter of the MC_PhasingEx instruction was invalid.	1. The value set for the MasterDistance input parameter of the MC_PhasingEx instruction must be greater than 0.
4090	0FFA	The direction parameter input for the MC_PhasingEx instruction exceeds the range.	1. The direction parameter input for the MC_PhasingEx instruction was out of range.	1. The direction parameter input for the MC_PhasingEx instruction has the following value range. 1: DIR_POSITIVE: Valid for forward master motion. 2: DIR_NEGATIVE: Valid for reverse master motion.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
4091	0FFB	The master axis has been replaced.	1. The master axis controlled by the instruction was replaced.	1. Check if multiple copies of the same instance for the instruction are running. If yes, assign a unique instance to each instruction.
4092	0FFC	The slave axis has been replaced.	1. The slave axis controlled by the instruction was replaced.	1. Check if multiple copies of the same instance for the instruction are running. If yes, assign a unique instance to each instruction.
4093	0FFD	MC_MoveLink cannot be re-triggered.	1. Do not re-trigger the current ongoing MC_MoveLink instruction.	1. Do not re-trigger the current ongoing MC_MoveLink instruction.
4094	0FFE	Master axis 1 has been replaced.	1. The master axis 1 controlled by the MC_CombineAxes instruction was replaced.	1. Check if multiple copies of the same instance for the MC_CombineAxes instruction is running. If yes, assign a corresponding instance to each MC_CombineAxes instruction.
4095	0FFF	Master axis 2 has been replaced.	1. The master axis 2 controlled by the MC_CombineAxes instruction was replaced.	1. Check if multiple copies of the same instance for the MC_CombineAxes instruction is running. If yes, assign a corresponding instance to each MC_CombineAxes instruction.
4096	1000	The master axis for the phase compensation instruction does not match the master axis when the slave axis is in synchronous motion.	1. The master axis for the phase compensation instruction did not match the master axis for the ongoing synchronous instruction.	1. Select the appropriate master axis for the MC_Phasing or MC_PhasingEx instruction based on the master and slave axes for the ongoing synchronous motion instruction.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
4097	1001	The displacement input for the MC_GetCamTablePhase instruction does not have any valid phase.	1. The displacement input for the MC_GetCamTablePhase instruction did not have any valid phase.	1. Modify the input displacement as required.
4098	1002	The node phase value is negative.	1. The node phase value for the MC_GetCamTableDistance instruction was negative.	1. The node phase value for the MC_GetCamTableDistance instruction must not be negative.
4099	1003	The cam table pointer is null.	1. The pointer of the cam table corresponding to the instruction was null.	1. Check if the pointer of the cam table corresponding to the instruction is null. If yes, select an appropriate cam table to execute the instruction as required.
4100	1004	The axis is in commissioning state.	1. The specified axis for the instruction was in commissioning state.	1. Exit the axis commissioning before executing the instruction.
4101	1005	Attempt to modify non-effective parameters upon re-trigger	1. When the MC_CamIn instruction was re-triggered, an attempt was made to modify parameters that will not be effective.	1. When the MC_CamIn instruction is re-triggered, only the parameters including Periodic, MasterScaling, SlaveScaling, Direction, and MasterValueSource can be modified.
4102	1006	Insufficient space to execute the instruction	1. There was insufficient space within the PLC to support the operation of the MC_CamIn or MC_VibrationShapping instruction.	1. Run the corresponding instruction(s) when there is sufficient space (greater than 5120KB) within the PLC.
4103	1007	CamNode pointer is null.	1. The pointer of the input and output parameter "CamNode" for the MC_GenerateCamTable instruction was null.	1. Check if the variable corresponding to the input and output parameter "CamNode" is null.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
4104	1008	Switches pointer is null.	1. The pointer of the input parameter "Switches" for the MC_DigitalCamSwitch instruction was null.	1. Assign a valid value to the input parameter "Switches" for the MC_DigitalCamSwitch instruction.
4105	1009	The number of node data provided is less than the capacity of the cam table when the cam data was updated using MC GenerateCamTable.	1. When the cam data was updated using MC GenerateCamTable, the number of node data provided was less than the capacity of the cam table.	1. Set a correct node number.
4106	100A	The master axis synchronous position overlaps with the current master axis position.	1. The input parameter "MasterSyncPosition" of the MC_GearInPos instruction overlapped with the current master axis position.	1. The input parameter "MasterSyncPosition" of the MC_GearInPos instruction must not overlap with the current master axis position. Set the valid value for "MasterSyncPosition".
4107	100B	The slave axis synchronous position overlaps with the current slave axis position.	1. The input parameter "SlaveSyncPosition" of the MC_GearInPos instruction overlapped with the current slave axis position.	1. The input parameter "SlaveSyncPosition" of the MC_GearInPos instruction must not overlap with the current slave axis position. Set the valid value for "SlaveSyncPosition".
4108	100C	The master axis position of the first point in the cam table is less than 0.	1. The master axis position of the first point in the cam table must not be less than 0.	1. The master axis position of the first point in the cam table must be greater than or equal to 0.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
4109	100D	The NodeNum parameter exceeds the size of the CamNode array.	The array size of the cam nodes input for the MC_CamNodeBinding instruction was smaller than the input number of nodes to process.	1. Adjust NodeNum to a value less than the number of elements in the CamNode array. 2. Increase the size of the CamNode array so that the number of array elements is greater than or equal to the number of NodeNum elements input.
4110	100E	The cam node type does not match the data type of the cam node array.	The data type of the cam node array input in the MC_CamNodeBinding instruction did not match the specified cam node type.	1. Ensure NodeType and CamNode correspond correctly: When a key node cam is specified for NodeType, the data type of CamNode array must be set to MC_CAM_KEY_NODE. When a dense node cam is specified for NodeType, the data type of CamNode array must be set to MC_CAM_FITTING_NODE.
4111	100F	When the input parameter UpDateRightNow of the MC_CamIn or MC_CamInV2 instruction is set to FALSE, the number of cam table points exceeds the limit.	When the input parameter UpDateRightNow of the MC_CamIn or MC_CamInV2 instruction is set to FALSE, the number of cam table points exceeds the limit: The number of key nodes is greater than 4000, or the number of dense nodes is greater than 20000.	Adjust the number of points in the cam table according to the requirements.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
6000	1770	The axis is in error state.	1. The axis has entered an error state, but no specific fault code is available.	1. Check if the motion position has exceeded the software or hardware limits. 2. Check whether the enable signal was interrupted during execution. 3. Check if an emergency stop instruction was triggered. 4. Use the MC_Reset instruction to switch the axis state from AXIS_ERRORSTOP to AXIS_STANDSTILL before attempting any further operations.
6001	1771	The axis is in commissioning state.	1. The specified axis for the instruction was in commissioning state.	1. Exit the axis commissioning before executing the instruction.
6004	1774	The following error is too large.	1. The following error was greater than the value set for "Hysteresis limit" of "Position hysteresis management" on the FA side.	1. Disable the "Position hysteresis management" enable switch on the FA side. 2. Increase the set value of "Hysteresis limit". 3. If possible, increase the stiffness parameters of the servo drive.
6005	1775	ECAT communication error	1. The communication of the slave is abnormal.	1. Check if the physical network of the slave station is disconnected. 2. Check for external interference. 3. Check if there is a driver fault.
6006	1776	The axis cycle time is not correctly set.	1. The task cycle time was set to a value less than or equal to 0.	1. Check the task cycle parameter settings.
6007	1777	The unexpected emergency stop is triggered.	1. After the emergency stop instruction had been triggered, the motion instruction or synchronous motion instruction was also triggered.	1. After the emergency stop instruction has been triggered, the MC_Reset instruction must be first triggered. Then, trigger the motion instruction or synchronous motion instruction.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
6008	1778	The planned speed exceeds the maximum speed set by the software.	1. When the local pulse axis executed the instruction, the value (speed x gear ratio denominator / gear ratio numerator) exceeded 200 kHz, causing an over-frequency.	1. Adjust the position parameters of the instruction or the gear ratio of the local pulse axis as required.
6009	1779	Axis driver alarm	1. An error occurred on the driver.	1. See the servo manual for troubleshooting. 2. If no fault code is displayed on the servo panel, check whether the network cable is intact and whether the connection of the network port is tight.
6010	177A	The path planning is interrupted abnormally, and the motion instruction stops.	1. The planned position was lost.	1. Ensure the program (PRG) containing the axis control function blocks is called within the task "EtherCAT_Master_Task". 2. If the program is written in ST, check whether the motion function block is being inhibited by an IF condition during execution. 3. If the program is written in LD language EN/ENO, check whether the motion function block is being inhibited by an EN during execution.
6011	177B	Faults occurred during homing.	1. One or more faults occurred during servo homing.	1. See the servo manual for troubleshooting.
6012	177C	SDO read data timeout	1. The SDO read operation in the current instruction timed out.	1. Check if there are any issues with the connection between the servo and PLC. 2. Configure the corresponding PDO object according to the instruction.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
6013	177D	The servo absolute position system does not match the position restoration mode.	1. The absolute position system of the servo was incompatible with the position restoration mode.	1. Select an appropriate position restoration mode based on the absolute position system of the servo.
6014	177E	The area mapped to the axis and storing its key data is not found.	1. The area mapped to the axis and storing its key data was not found.	1. The position restoration cannot be performed because the data required for position restoration is missing.
6015	177F	The power failure retention variable, which stores the information required for the servo position restoration scheme, does not exist. Therefore, the position restoration operation cannot be performed.	1. The power failure retention variable, which stores the information required for the servo position restoration scheme, did not exist. Therefore, the position restoration operation cannot be performed.	1. The position restoration cannot be performed because the data required for position restoration is missing.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
6016	1780	The power failure retention variable, which stores the information required for the PLC position restoration scheme, does not exist. Therefore, the position restoration operation cannot be performed.	1. The power failure retention variable, which stores the information required for the servo position restoration scheme, did not exist. Therefore, the position restoration operation cannot be performed.	1. The position restoration cannot be performed because the data required for position restoration is missing.
6017	1781	The encoder revolution count after restoration exceeds the limited range of the data type.	1. After restoration, the encoder revolution count is outside the permissible range of the data type.	1. After restoration, the encoder revolution count was outside the permissible range of the data type.
6018	1782	The vibration shaping process terminated abnormally.	1. The MC_VibrationShapping instruction was reset during execution.	1. Do not reset the MC_VibrationShapping instruction during execution.
7001	1B59	The axis type of the axis group is not defined.	1. The axis type of the axis group controlled by this instruction was not defined.	1. In the IDE interface, select an appropriate axis type for the axis in the axis group under PLC -> Motion control -> Axis -> "Axis name in the axis group" -> General settings as required.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
7002	1B5A	The instruction cannot be executed when the axis group is in an illegal state.	1. The axis group with the ratio to be set was in the GROUP_DISABLED or GROUP_HOMING state.	1. If the axis group with the ratio to be set is in GROUP_DISABLED, use the MC_GroupEnable instruction to switch the axis group from GROUP_DISABLED to GROUP_STANDBY before executing the instruction. 2. If the axis group with the ratio to be set is in the GROUP_HOMING state, wait until the axis group has completed the homing instruction. Then, execute the instruction when the axis group is in the GROUP_STANDBY state.
7003	1B5B	The axis group is in the GROUP_DISABLED state.	1. The axis group controlled by the instruction was in the GROUP_DISABLED state. The instruction can only be triggered and executed when the axis group is in the GROUP_MOVING or GROUP_STANDBY state.	1. Use the MC_GroupEnable instruction to switch the axis group to GROUP_STANDBY. Then, execute the instruction.
7004	1B5C	The axis group is in the GROUP_STOPPING state.	1. The axis group controlled by the instruction was in the GROUP_STOPPING state. The instruction can only be triggered and executed when the axis group is in the GROUP_MOVING or GROUP_STANDBY state.	1. Use the MC_GroupDisable instruction to switch the axis group from GROUP_STOPPING to GROUP_DISABLED. Then, use the MC_GroupEnable instruction to switch the axis group to GROUP_STANDBY before executing the instruction. 2. Check whether the output variable Done of the MC_GroupStop instruction (which placed axis group in GROUP_STOPPING) is activated. If so, reset the input variable Execute of the MC_GroupStop instruction. At this point, the axis group is in the GROUP_STANDBY state. Then, execute the instruction again.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
7005	1B5D	The axis group in the GROUP_HOMING state.	1. The axis group controlled by the instruction was in the GROUP_HOMING state. The instruction can only be triggered and executed when the axis group is in the GROUP_MOVING or GROUP_STANDBY state.	1. Wait until the axis group has completed the homing instruction. Then, execute the instruction when the axis group is in the GROUP_STANDBY state.
7006	1B5E	The axis group is in the GROUP_ERRORSTOP state.	1. The axis group controlled by the instruction was in the GROUP_ERRORSTOP state. The instruction can only be triggered and executed when the axis group is in the GROUP_MOVING or GROUP_STANDBY state.	1. Use the MC_GroupReset instruction to switch the axis group to GROUP_STANDBY. Then, execute the instruction.
7007	1B5F	The axis group is in the process of GROUP_ERRORSTOP.	1. In the axis group to be reset, one or more axes were switched to AXIS_ERRORSTOP due to the MC_ImmediateStop instruction. 2. The axis group to be reset was in the GROUP_ERRORSTOP state and the stop action has not yet completed.	1. Check whether any axis has entered AXIS_ERRORSTOP due to the MC_ImmediateStop instruction in the axis group to be reset. If so, reset the corresponding axis using the MC_Reset instruction before executing the MC_GroupReset instruction. 2. Check if multiple copies for the instance of the MC_GroupReset instruction are executing simultaneously on the axis group to be reset. If yes, delete the duplicate instances of the MC_GroupReset instruction, and then execute the MC_GroupReset instruction again.
7008	1B60	The axis in the axis group are in the error state.	1. In the current axis group, one or more axes were in the AXIS_ERRORSTOP state.	1. Check whether the axis in the current axis group is in the AXIS_ERRORSTOP state. Use the MC_Reset instruction to switch the axis from AXIS_ERRORSTOP to AXIS_STANDSTILL or AXIS_POWER_OFF before executing the instruction.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
7009	1B61	The cycle time of the axis in the axis group has not been set correctly.	1. An axis within the axis group had the task cycle time set to a value less than or equal to 0.	1. Correct the parameter value set for the task cycle of the axis in the axis group.
7010	1B62	The axis group is controlled by the stop instruction.	1. The axis group was executing the MC_GroupHalt instruction.	1. Wait until the MC_GroupHalt instruction for the axis group has been completed. Then, execute the instruction.
7011	1B63	The axis group is controlled by the emergency stop instruction.	1. The axis group was executing the MC_GroupStop instruction.	1. Wait until the MC_GroupStop instruction for the axis group has been completed. Then, execute the instruction.
7012	1B64	The axis group is controlled by the pause instruction.	1. The axis group was controlled by the MC_GroupInterrupt instruction.	1. Use the MC_GroupContinue instruction to switch the instruction type of the axis group from GROUP_INTERRUPT to GROUP_SETOVERRIDE before executing the instruction.
7016	1B68	The axis is not configured in the axis group.	1. The axis was not configured into the axis group.	1. When the axis is not configured into the axis group, check if there are any unassigned axes in "Project tree" -> "Axis group" -> "Axis configuration".

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
7017	1B69	The axis coordinate exceeds the positive limit.	1. For the relative motion, the input variable "Distance" would cause the axis group to exceed the positive limit of the axis group during the execution of the instruction. 2. For the absolute motion, the input variable "Position" would cause the axis group to exceed the positive limit of the axis group during the execution of the instruction.	1. Based on the positive limit, select an appropriate movement distance for the relative motion of the axis group before executing the instruction. 2. Based on the positive limit, select an appropriate target position for the absolute motion of the axis group before executing the instruction.
7018	1B6A	The axis coordinate exceeds the negative limit.	1. For the relative motion, the input variable "Distance" would cause the axis group to exceed the negative limit of the axis group during the execution of the instruction. 2. For the absolute motion, the input variable "Position" would cause the axis group to exceed the negative limit of the axis group during the execution of the instruction.	1. Based on the negative limit, select an appropriate movement distance for the relative motion of the axis group before executing the instruction. 2. Based on the negative limit, select an appropriate target position for the absolute motion of the axis group before executing the instruction.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
7019	1B6B	The axis coordinate speed exceeds the limit.	1. In the joint axes coordinate system (ACS), the absolute value of the axis speed was greater than that of the product of the warning planned factor and the axis planned speed in the axis group.	1. Adjust the speed parameters of the axis group motion as required. 2. If the axis group type is SCARA and the coupling setting has been checked, check if the coupling coefficient is set correctly. 3. If the axis group type is SCARA, use MC_SetPosition or MC_Home to set the position of the corresponding axis for each joint within the SCARA configuration range before using the MC_GroupEnable instruction. 4. If the axis group type is Delta2, Delta3 or Delta3+1, use MC_SetPosition or MC_Home to set the position of the corresponding axis for each joint within the corresponding Delta configuration range before using the MC_GroupEnable instruction.
7020	1B6C	The axis group is switching the part coordinate system.	1. When the axis group is switching the part coordinate system, some function blocks are triggered.	1. Wait until the switch has been completed, and then trigger the function blocks.
7021	1B6D	The axis group is referenced to a dynamic part coordinate system.	1. When the axis group was referenced to a dynamic part coordinate system, the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction has been started.	1. Do not start the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction, when the axis group was referenced to a dynamic part coordinate system.
7022	1B6E	The axis group is in motion.	1. During the interpolation motion of the axis group, it is prohibited to set the part coordinate system or trigger continuous trajectory function blocks and jog function blocks.	1. Do not trigger the related function blocks during the axis group motion. Trigger the function block after the interpolation motion has ended.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
7023	1B6F	The coupling coefficient and transmission ratio are set incorrectly.	1. The coupling factor settings should be reversible. Otherwise, an error will be reported when the axis group is enabled or reset.	1. Check and modify the settings of the coupling coefficient and transmission ratio to ensure they are set correctly.
7024	1B70	The axis in the axis group is in the disabled state.	1. When the axis group was in motion or in a dynamic coordinate system, an axis in the axis group was in the disabled state.	1. Check and set the enable status of any axis in the axis group to meet the instruction requirements before the re-execution of the instruction.
7026	1B72	The axis in the axis group is in the stop state.	1. When the axis group was in motion or in a dynamic coordinate system, the axis in the axis group was in the stop state.	1. Check and set the enable status of any axis in the axis group to meet the instruction requirements before the re-execution of the instruction.
7040	1B80	The axis group velocity exceeds the minimum limit.	1. The input variable "VelFactor" for the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction must be within the range (0.0,1.0]. 2. The value of the input variable "Velocity" was less than or equal to 0.	1. Correct the value of the input variable "VelFactor" for the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction. 2. Correct the value of the input variable "Velocity" for the instruction.
7041	1B81	The axis group velocity exceeds the maximum limit.	1. The input variable "VelFactor" for the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction must be within the range (0.0,1.0]. 2. The value of the input variable "Velocity" for the instruction was greater than the maximum value of the system motion parameters.	1. Correct the value of the input variable "VelFactor" for the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction. 2. Correct the value of the input variable "Velocity" for the instruction.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
7042	1B82	The axis group acceleration exceeds the minimum limit.	1. The input variable "AccFactor" for the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction must be within the range (0.0,1.0]. 2. The value of the input variable "Velocity" for the instruction was less than or equal to 0.	1. Correct the value of the input variable "AccFactor" for the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction. 2. Correct the value of the input variable "Acceleration" for the instruction.
7043	1B83	The axis group acceleration exceeds the maximum limit.	1. The input variable "AccFactor" for the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction must be within the range (0.0,1.0]. 2. The value of the input variable "Acceleration" for the instruction was greater than the maximum value of the system motion parameters.	1. Correct the value of the input variable "AccFactor" for the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction. 2. Correct the value of the input variable "Acceleration" for the instruction.
7044	1B84	The axis group deceleration exceeds the minimum limit.	1. The input variable "AccFactor" for the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction must be within the range (0.0, 1.0]. 2. The value of the input variable "Deceleration" for the instruction was less than or equal to 0.	1. Correct the value of the input variable "AccFactor" for the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction. 2. Correct the value of the input variable "Deceleration" for the instruction.
7045	1B85	The axis group deceleration exceeds the maximum limit.	1. The input variable "AccFactor" for the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction must be within the range (0.0, 1.0]. 2. The value of the input variable "Deceleration" for the instruction was greater than the maximum value of the system motion parameters.	1. Correct the value of the input variable "AccFactor" for the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction. 2. Correct the value of the input variable "Deceleration" for the instruction.
7046	1B86	The axis group jerk exceeds the minimum limit.	1. The input variable "JerkFactor" for the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction must be within the range [0.0,1.0]. 2. The value of the input variable "Jerk" was less than or equal to 0.	1. Correct the value of the input variable "JerkFactor" for the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction. 2. Correct the value of the input variable "Jerk" for the instruction.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
7047	1B87	The axis group jerk exceeds the maximum limit.	1. The input variable "JerkFactor" for the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction must be within the range [0.0,1.0]. 2. The value of the input variable "Jerk" for the instruction was greater than the maximum value of the system motion parameters.	1. Correct the value of the input variable "JerkFactor" for the MC_MoveDirectRelative or MC_MoveDirectAbsolute instruction. 2. Correct the value of the input variable "Jerk" for the instruction.
7048	1B88	The axis group velocity factor exceeds the minimum limit.	1. The value range of the input variable "VelFactor" for the MC_GroupSetOverride instruction must be within the range [0,100].	1. Correct the value of the input variable "VelFactor" for the instruction.
7049	1B89	The axis group velocity factor exceeds the maximum limit.	1. The value range of the input variable "VelFactor" for the MC_GroupSetOverride instruction must be within the range [0,100].	1. Correct the value of the input variable "VelFactor" for the instruction.
7050	1B8A	The axis group acceleration factor is below the minimum limit.	1. The input variable "AccFactor" of the MC_GroupSetOverride instruction must be within the range [1,100].	1. Correct the value of the input variable "AccFactor".
7051	1B8B	The axis group acceleration factor exceeds the maximum limit.	1. The input variable "AccFactor" of the MC_GroupSetOverride instruction must be within the range [1,100].	1. Correct the value of the input variable "AccFactor".

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
7052	1B8C	The axis group jerk factor is below the minimum limit.	1. The input variable "JerkFactor" of the MC_GroupSetOverride instruction has a value range of [0,100].	1. Correct the value of the input variable "JerkFactor".
7053	1B8D	The axis group jerk factor exceeds the maximum limit.	1. The input variable "JerkFactor" of the MC_GroupSetOverride instruction has a value range of [0,100].	1. Correct the value of the input variable "JerkFactor".
7054	1B8E	The axis group transition parameter is below the minimum limit.	1. The value range of the input variable "TransitionParameter" varies depending on the input variable "TransitionMode". If "TransitionMode" is related to Velocity, the value range of "TransitionParameter" is [0,1]; if "TransitionMode" is related to Distance, the value range of "TransitionParameter" is [0,+∞).	1. Correct the value of the input variable "TransitionParameter" for the instruction.
7055	1B8F	The axis group transition parameter exceeds the maximum limit.	1. The value range of the input variable "TransitionParameter" varies depending on the input variable "TransitionMode". If "TransitionMode" is related to Velocity, the value range of "TransitionParameter" is [0,1]; if "TransitionMode" is related to Distance, the value range of "TransitionParameter" is [0,+∞).	1. Correct the value of the input variable "TransitionParameter" for the instruction.
7056	1B90	The axis group buffer mode is undefined.	1. The value of the input variable "BufferMode" for the instruction was out of range.	1. Select an appropriate buffer mode to enable the axis group motion as required.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
7057	1B91	The axis group transition mode is undefined.	1. The value of the input variable "TransitionMode" for the instruction was out of range.	1. Select an appropriate transition mode to enable the axis group motion as required.
7058	1B92	The axis group transition mode is not allowed	1. When the input variable "BufferMode" of the MC_MoveXXXRelative or MC_MoveXXXAbsolute instruction is in blending mode, the input variable "TransitionMode" of this instruction must not be TMResetAbort or TMDecAbort. 2. When the input variable "BufferMode" of the MC_MoveXXXRelative or MC_MoveXXXAbsolute instruction is set to Aborting, the input variable "TransitionMode" of this instruction must be TMResetAbort, TMDecAbort or TMnone.	1. Select an appropriate transition mode to enable the axis group motion as required.
7059	1B93	The axis group circular mode is not defined.	1. The value of the input variable "CircMode" for the MC_MoveCircularRelative or MC_MoveCircularAbsolute instruction was outside the valid range.	1. Select an appropriate circular mode to enable the axis group motion as required.
7060	1B94	The axis group circular path selection is not defined.	1. The value of the input variable "PathChoice" for the MC_MoveCircularRelative or MC_MoveCircularAbsolute instruction was outside the valid range.	1. Select an appropriate path selection to enable the axis group motion as required.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
7061	1B95	Failed to reset buffer motion	1. The instruction was re-triggered after the buffer mode of the buffered instruction had been changed to the MC_ABORTING mode. 2. The buffered instruction was triggered again after its parameters were modified, and the modified parameters were invalid. 3. The motion data of the buffered instruction has already been used in motion calculations.	1. Do not change the buffer mode for the buffered instruction to the MC_ABORTING mode before re-triggering. 2. Check whether the coordinate system of the buffered instruction is correct. 3. The motion data of the buffered instruction has already been used in motion calculations, but the time to re-trigger the instruction is not correct.
7063	1B97	The kinematic model does not exist.	1. The kinematic model of the axis group controlled by the instruction did not exist.	1. In the IDE interface, select an appropriate axis group type for the axis group under PLC -> Motion control -> Axis group -> "Axis group name" -> Basic settings -> Axis group type as required.
7065	1B99	The axis group instruction buffer is full.	1. The maximum number of buffered instructions currently allowed for the axis group is 8.	1. Wait for the executing and buffered instructions for the current axis group to complete before executing other motion-related buffered instructions.
7068	1B9C	Incorrect communication status of the axis in the axis group	1. The axis group to be enabled contained the axes with incorrect communication status.	1. Check the communication status of each axis in the axis group.
7069	1B9D	Pose space configuration error	1. The A, B, C orientation parameters of the stTool input parameter for the MC_SetToolTransform instruction did not match the axis group type.	1. Check the input parameter stTool of the MC_SetToolTransform instruction to match the axis group type.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
7081	1BA9	The axis group motion could not be safely aborted due to insufficient deceleration time.	1. When a subsequent instruction aborted the current instruction, the deceleration time was greater than the motion time of the subsequent instruction.	1. When the axis group is in GroupErrorStop, perform the corresponding operations, after using the MC_GroupReset instruction.
7083	1BAB	The axis group path planning is interrupted abnormally, and the motion instruction stops.	1. EN/ENO of LD language was used, and the axis group motion instruction was inhibited by EN during execution.	1. Check whether EN/ENO in the LD language is used, and whether the axis group motion instruction is being inhibited by an EN during execution.
7091	1BB3	The kinematic model does not exist.	1. The kinematic model of the axis group controlled by the instruction did not exist.	1. In the IDE interface, select an appropriate axis group type for the axis group under PLC -> Motion control -> Axis group -> "Name of the axis group for the circular relative motion" -> Basic setting -> Axis group type as required.
7092	1BB4	Unknown motion instruction	1. The motion instruction was undefined.	1. Check if the motion instruction is correct.
7093	1BB5	Setting a dynamic coordinate system based on the same axis group is not permitted.	1. The follower axis group was identical to the master axis group.	1. When setting up a dynamic coordinate system, ensure that the follower and master axis groups are not identical.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
7096	1BB8	The reference axis is identical to as the axis in the axis group.	1. The axis corresponding to the input/output parameter "ConveyorBelt" of the MC_TrackConveyorBelt instruction was identical to the axis in the axis group. 2. The axis corresponding to the input/output parameter "RotaryTable" of the MC_TrackRotaryTable instruction was identical to the axis in the axis group. 3. The axis corresponding to the input/output parameter "BaseAxis" of the MC_SetCoordinateTransform instruction was identical to the axis in the axis group.	1. Adjust the axis corresponding to the input/output parameter "ConveyorBelt" of the MC_TrackConveyorBelt instruction, or modify the axis mapped to the axis group. 2. Adjust the axis corresponding to the input/output parameter "RotaryTable" of the MC_TrackRotaryTable instruction, or modify the axis mapped to the axis group. 3. Adjust the axis corresponding to the input/output parameter "BaseAxis" of the MC_SetCoordinateTransform instruction, or modify the axis mapped to the axis group.
7100	1BBC	An axis identifier mapped to the axis group exceeds the specified range.	1. The axis identifier to be added has exceeded the maximum permitted number of physical axes.	1. Check if the axis identifier to be added is within the valid range.
7101	1BBD	The axis mapped to the axis group was assigned to the identical axis group.	1. The axis to be added has been assigned to this axis group.	1. The axis to be added has already been assigned to this axis group, so the axis does not need to be added repeatedly.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
7102	1BBE	The current axis is already mapped to an axis object, so the axis needs to be unmapped and then remapped, or the axis has been added in the axis identifier.	1. The axis to be added has been already mapped to another axis group. 2. The axis was added into the axis identifier in the axis group. Do not add more axes into this axis identifier.	1. Check if the axis to be added is correct. 2. Remove the axis to be added from other axis groups before using this instruction.
7103	1BBF	The axis to be removed do not exist.	1. The axis to be removed was not in the axis group.	1. The axis to be removed is not in the axis group, so do not remove the axis repeatedly.
7106	1BC2	The number of the axes in the axis group does not match the axis group type.	1. The number of the axes in the axis group to be enabled did not match the axis group type.	1. Check if the axes in the axis group have been removed using the MC_UngroupAllAxes or MC_RemoveAxisFromGroup instructions. If so, use the MC_AddAxisToGroup instruction to add the axes back to the axis group.
7107	1BC3	This axis has already been enabled by another axis group.	1. The axis group to be enabled contained the axes that are already enabled by another axis group.	1. Check if any axes in the axis group to be enabled are mapped to another enabled axis group. If yes, disable the enabled axis group before using this instruction.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
7108	1BC4	The operation is prohibited due to the axis group state.	1. The instruction was executed in the current axis group state.	1. Do not execute the MC_GroupDisabled instruction if the axis group is in either the GROUPMOVING or GROUPHOMING state. 2. Do not execute the MC_GroupHome, MC_GroupSetPosition, MC_SetKinTransform, MC_GroupInterrupt, and MC_GroupContinue instructions if the axis group is not in the GROUPSTANDBY state. 3. Do not execute the MC_GroupReadXXX instructions if the axis group is in the GROUPDISABLED state.
7109	1BC5	The axis group is in the GROUP_STOPPING state.	1. The axis group was in the GROUP_STOPPING state.	1. Do not execute the MC_GroupHalt instruction if the axis group is in the GROUP_STOPPING state.
7110	1BC6	Invalid axis state for group addition	1. The axis to be added by the MC_AddAxisToGroup instruction was not in the StandStill state.	1. Switch the axis to be added to the StandStill or PowerOff state.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
7111	1BC7	Do not add the axis to the axis group if this axis is set as a rotary axis.	1. The axis to be added by the MC_AddAxisToGroup instruction was set as a rotary axis. 2. The axis to be added by the MC_AddAxisToGroup instruction was set as an encoder axis. 3. The conveyor belt axis of the MC_TrackConveyorBelt instruction was set as a rotary axis. 4. The axis group to be enabled by the MC_GroupEnable instruction contained an axis that was set as a rotary axis. 5. The rotary table axis of the MC_TrackRotaryTable instruction was set as a rotary axis. 6. The reference axis of the MC_SetCoordinateTransform instruction was set as a rotary axis.	1. Check if the axis group of the faulty instruction includes any axis in the rotary type, or if the axis of the instruction is set as a rotary axis. If so, select an appropriate linear axis to use the instruction.
7112	1BC8	Do not use the instruction when an axis is not in CSP or CSV mode.	1. The axis to be added by the MC_AddAxisToGroup instruction was not in CSP or CSV mode. 2. The axis group to be enabled by the MC_GroupEnable instruction contained the axis that was not in CSP or CSV mode.	1. Use the MC_SetControllerMode instruction to switch the control mode of the axis to CSV mode or CSP mode; then execute the MC_AddAxisToGroup instruction. 2. Use the MC_SetControllerMode instruction to switch the control mode of the axis to CSV mode or CSP mode; then execute the MC_GroupEnable instruction.
7113	1BC9	Non-cyclic position update	1. The axes in the axis group space configuration were missing during the execution of the MC_GroupReadActualPosition, MC_GroupReadActualVelocity, or MC_GroupReadActualAcceleration instructions, or the forward kinematics calculation was incorrect.	1. Reconfigure the axis group. 2. Check if the kinematic parameters of the axis group are correct.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
7114	1BCA	The axis group object does not exist.	1. The axis group object for the faulty instruction did not exist.	1. Check if the axis group corresponding to the faulty instruction is configured in the IDE interface. If not configured, reconfigure the axis group and then use the corresponding instruction again.
7115	1BCB	The axis object is null.	1. The axis to be added by the MC_AddAxisToGroup instruction did not exist.	1. If the axis to be added by the MC_AddAxisToGroup instruction did not exist, check whether the axis is configured in the IDE interface, if not, configure the axis and then execute the instruction again.
7116	1BCC	The communication status of the axis group is abnormal.	1. The status of the servo slave for the axis in the axis group of the faulty instruction was abnormal.	1. Check if the slave device mapped to the axis variable is a servo drive. 2. Check if there are any abnormal alarm information from the servo. 3. Check if the network cable connecting the slave is normal. 4. Replace the network cable. 5. Restart the servo drive.
7117	1BCD	Abnormal parameter input for COORDSYSTEM	1. The MC_GroupHome instruction only supports the homing process under the ACS coordinate system. 2. The MC_GroupSetPosition, MC_GroupReadActualPosition, MC_GroupReadActualVelocity, MC_GroupReadActualAcceleration, MC_MoveDirectRelative, and MC_MoveDirectAbsolute instructions can only be executed under the ACS or PCS coordinate systems. 3. The MC_MoveCircularRelative, MC_MoveCircularAbsolute, MC_MoveLinearRelative, and MC_MoveLinearAbsolute instructions currently only support the PCS mode.	1. Select the appropriate coordinate system parameter based on the faulty instruction.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
7119	1BCF	The axis group type is not defined.	1. The axis group type was undefined.	1. Check if the axis group type is selected in "Project tree" -> "Axis group" -> "Mechanical type".
7120	1BD0	The axis group motion cannot be re-triggered.	1. The instruction was executed repeatedly while the axis group was executing this instruction.	1. Check if multiple identical instruction instance are being executed in the PLC program. If so, delete the duplicated instruction instance and execute the instruction again. 2. The instruction cannot be re-triggered while the axis group is executing this instruction.
7121	1BD1	The axis variable in the axis group is not enabled	1. The axis group of the instruction contained a disabled axis.	1. Use the MC_Power instruction to switch the state of the axis from AXIS_POWER_OFF to AXIS_STANDSTILL, and then execute the instruction again.
7123	1BD3	The starting sequence in axis group homing procedure is invalid.	1. In the homing sequence of the axis group, the homing order of the starting axis must be set to 1.	1. In the homing sequence of the axis group, the homing order of the starting axis must be set to 1.
7124	1BD4	The axis group homing sequence is out of range.	1. In the homing sequence of the axis group, the homing order assigned to an axis was greater than the total number of axes in the group or less than 0.	1. Correct the homing order for the axes so that each value is greater than 0 and less than or equal to the total number of axes in the group.
7125	1BD5	Incorrect axis group homing method	1. The homing method of the axis group is not a servo homing method or a host controller homing method.	1. Adjust the homing method of the axis in the axis group to a servo homing method or a host controller homing method.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
7126	1BD6	The axis group homing sequence is not in ascending order.	1. The axis group homing sequence must be in ascending order. The valid sequences include: 1, 1, 1, 1; 1, 2, 3, 4; 1, 2, 2, 2; 1, 2, 2, 3; 1, 2, 3, 3, and so on.	1. Correct the axis group homing sequence to ensure it is in ascending order.
7128	1BD8	A limit was triggered on an axis mapped to the axis group when its planned velocity matched the limit's direction.	1. A limit on an axis mapped to the axis group has been triggered. The planned velocity direction of the axis to trigger the limit matched the direction of the activated limit.	1. Check if the limits are configured for the axis mapped in the axis group. If yes, modify the positive and negative limit ranges as required, or disable the limits.
7129	1BD9	Switching the axis group instance during its operation is not permitted.	1. While the axis group was executing an instruction, the MC_GroupStop instruction was initiated by another axis group repeatedly.	1. Check if multiple axis groups are executing the identical instruction instance. If yes, create a separate instruction instance for each axis group and re-execute the corresponding instruction.
7131	1BDB	The distance between the start point and the end point is too long (greater than the given diameter)	1. When the input variable "CircMode" of either the MC_MoveCircularRelative or MC_MoveCircularAbsolute instruction was set to RADUS_ARC_TYPE, the invalid value of the input variable "EndPoint" resulted in a too long distance between the start point and the end point.	1. According to the requirements, select an appropriate end point for the circular motion of the axis group before the execution of the instruction.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
7132	1BDC	The distance between the start point and the end point is too short.	1. When the input variable "CircMode" of either the MC_MoveCircularRelative or MC_MoveCircularAbsolute instruction was set to RADUS_ARC_TYPE, the invalid value of the input variable "EndPoint" resulted in a too short distance between the start point and the end point.	1. According to the requirements, select an appropriate end point for the circular motion of the axis group before the execution of the instruction.
7133	1BDD	The radius is too short.	1. When the input variable "CircMode" of either the MC_MoveCircularRelative or MC_MoveCircularAbsolute instruction was set to RADUS_ARC_TYPE, the arc radius was too short.	1. According to the requirements, select an appropriate end point for the circular motion of the axis group before the execution of the instruction.
7134	1BDE	The normal vector is not perpendicular. (An error will be generated if the deviation exceeds 10°.)	1. When the input variable "CircMode" of either the MC_MoveCircularRelative or MC_MoveCircularAbsolute instruction was set to RADUS_ARC_TYPE, the normal vector is not perpendicular.	1. According to the requirements, select an appropriate end point for the circular motion of the axis group before the execution of the instruction.
7135	1BDF	The distance from the start point to the center is not equal to that from the end point to the center.	1. When the input variable "CircMode" of either the MC_MoveCircularRelative or MC_MoveCircularAbsolute instruction was set to ENTER_ARC_TYPE, the distance from the start point to the center was not equal to that from the end point to the center.	1. According to the requirements, select an appropriate end point for the circular motion of the axis group before the execution of the instruction.
7240	1C48	Inverse kinematics failure	1. The inverse kinematics calculation for the axis group pose has failed.	1. Check if the axis group type is correct. 2. Check if the target or via positions in the axis group motion path are at or near a kinematic singularity.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
7241	1C49	Forward solution failure	1. The forward kinematics calculation for the axis group pose has failed.	1. Check if the axis group type is correct. 2. Check if the target or via positions in the axis group motion path are at or near a kinematic singularity.
7243	1C4B	Euler angles are out of range.	The value of the A-axis position parameter set in the PCS space was out of range.	Adjust the A-axis value to within ±180°.
7246	1C4E	The degrees of freedom for the axis group do not match the configured coordinate system parameters.	Due to mechanical configuration constraints, only the X and Y coordinate axis values can be set for XY and Delta2 configurations.	Reconfigure the coordinate system parameters according to the instruction function description.
8002	1F42	The counter channel status (0x6000) is not configured.	1. The PDO object 6000 was not configured in the software backend.	1. Configure the PDO object 6000 in the software backend.
8003	1F43	The digital input terminal status (0x6001) is not configured.	1. The PDO object 6001 was not configured in the software backend.	1. Configure the PDO object 6001 in the software backend.
8006	1F46	The touch probe status (0x6004) is not configured.	1. The PDO object for touch probe status 6004 was not configured in the software backend.	1. Configure the touch probe control word 6004 in the software backend.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
8007	1F47	The touch probe 1 rising edge position (0x6005) is not configured.	1. The PDO object for touch probe 1 rising edge position 6005 was not configured in the software backend.	1. Configure the PDO object for touch probe 1 rising edge position 6005 in the software backend.
8008	1F48	The touch probe 1 falling edge position (0x6006) is not configured.	1. The PDO object for touch probe 1 falling edge position 6006 was not configured in the software backend.	1. Configure the PDO object for touch probe 1 falling edge position 6006 in the software backend.
8009	1F49	The touch probe 2 rising edge position (0x6007) is not configured.	1. The PDO object for touch probe 2 rising edge position 6007 was not configured in the software backend.	1. Configure the PDO object for touch probe 2 rising edge position 6007 in the software backend.
8010	1F4A	The touch probe 2 falling edge position (0x6008) is not configured.	1. The PDO object for touch probe 2 falling edge position 6008 was not configured in the software backend.	1. Configure the PDO object for touch probe 2 falling edge position 6008 in the software backend.
8012	1F4C	The rising edge time PDO object for probe 1 (1B03/1B13) is not configured.	The ENC_TouchProbe instruction, which is used for a bus encoder axis, is configured to trigger on the rising edge of probe 1 time probe, but 1B03/1B13 PDO is not configured.	1. Configure 1B03/1B13 PDO in the software backend.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
8013	1F4D	The falling edge time PDO object for probe 1 (1B04/1B14) is not configured.	The ENC_TouchProbe instruction, which is used for a bus encoder axis, is configured to trigger on the falling edge of probe 1 time probe, but 1B04/1B14 PDO is not configured.	1. Configure 1B04/1B14 PDO in the software backend.
8014	1F4E	The rising edge time PDO object for probe 2 (1B03/1B13) is not configured.	The ENC_TouchProbe instruction, which is used for a bus encoder axis, is configured to trigger on the rising edge of probe 2 time probe, but 1B03/1B13 PDO is not configured.	1. Configure 1B03/1B13 PDO in the software backend.
8015	1F4F	The falling edge time PDO object for probe 2 (1B04/1B14) is not configured.	The ENC_TouchProbe instruction, which is used for a bus encoder axis, is configured to trigger on the falling edge of probe 2 time probe, but 1B04/1B14 PDO is not configured.	1. Configure 1B04/1B14 PDO in the software backend.
8017	1F51	The channel control word (0x7000) is not configured.	1. The PDO object 7000 was not configured in the software backend.	1. Configure the PDO object 7000 in the software backend.
8019	1F53	The touch probe control word (0x7002) is not configured.	1. The PDO object for touch probe control word 7002 was not configured in the software backend.	1. Configure the touch probe control word 7002 in the software backend.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
8020	1F54	The digital output terminal function selection (0x7003) is not configured.	1. The PDO object 7003 was not configured in the software backend.	1. Configure the PDO object 7003 in the software backend.
8028	1F5C	The axis type controlled by the instruction is incorrect.	1. The ENC_Counter, ENC_Reset, ENC_TouchProbe, ENC_Compare, ENC_StepCompare, ENC_ArrayCompare and ENC_Preset instructions only support local encoder axes and bus encoder axes. 2. The ENC_ReadStatus and ENC_DigitalOutput instructions only support bus encoder axis types.	1. Based on the axis types controlled by the instruction, configure the correct axis type in the software backend.
8029	1F5D	Bus encoder module failure	1. The bus encoder module has failed.	1. Power cycle the bus encoder module.
8030	1F5E	This instruction only supports bus encoder axes instead of local encoder axes.	1. This instruction only supports bus encoder axes and does not support local encoder axes.	1. This instruction only supports bus encoder axes and does not support local encoder axes. Correct the axis type in the software backend.
8031	1F5F	The ENC_Reset instruction is called on the bus encoder axis without an error occurring.	1. This instruction was triggered without an error occurring.	1. Check whether the axis is in an error state, and then trigger the instruction.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
8032	1F60	Bus encoder axis trigger mode selection error	1. When the axis type was set to a bus encoder axis, the set value of the input parameter TrigerMode was not 0, 1, or 4.	1. Modify the set value of the input parameter TrigerMode.
8033	1F61	The preset value is greater than 2147483647 or less than -2147483648.	1. When the set value of the input parameter Position was converted from user unit to pulse unit, it was greater than 2147483647 or less than -2147483648.	1. Modify the set value of the input parameter Position.
8034	1F62	Local encoder axis trigger mode selection error	1. When the axis type was set to a local encoder axis, the set value of the input parameter TrigerMode was not 0, 1, 2, 3.	1. Modify the set value of the input parameter TrigerMode.
8035	1F63	The preset value is greater than the positive software limit value.	1. In linear mode, the set value of the input parameter Position was greater than the positive limit value set in the software backend.	1. Modify the set value of the input parameter Position or the set value of the positive limit in the software backend. 2. Modify the set value of the negative limit in the software backend or disable the limit function.
8036	1F64	The preset value is less than the negative software limit value.	1. In linear mode, the set value of the input parameter Position was less than the negative limit value set in the software backend.	1. Modify the set value of the input parameter Position. 2. Modify the set value of the negative limit in the software backend or disable the limit function.
8037	1F65	In rotation mode, the preset value is greater than or equal to the cycle value.	1. In rotation mode, the set value of the input parameter Position is greater than or equal to the cycle value set in the backend.	1. Modify the set value of the input parameter Position or modify the cycle value set in the backend.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
8038	1F66	The preset value is incorrect.	1. The value was out of the count value range.	1. Adjust the set value to within the valid range.
8039	1F67	The comparison value is incorrect.	1. The comparison value was invalid.	1. Adjust the comparison value to within the valid range.
8040	1F68	The preset function of X input terminal is set incorrectly.	1. The selected terminal has not been set to the preset function.	1. In the IDE interface, select an appropriate terminal in the preset settings.
8041	1F69	The preset function of Y input terminal is set incorrectly.	1. The selected terminal has not been set to the reset function.	1. In the IDE interface, select an appropriate terminal in the reset settings.
8042	1F6A	The touch probe function of X input terminal is set incorrectly.	1. The selected terminal has not been set to the touch probe function.	1. In the IDE interface, select an appropriate terminal in the probe settings.
8043	1F6B	The comparison function of Y input terminal is set incorrectly.	1. The selected terminal has not been set to the comparison function.	1. In the IDE interface, select an appropriate terminal in the comparison settings.
8044	1F6C	Input frequency overload	1. The input frequency exceeded its limit.	1. Modify the gear ratio or the target velocity of the axis.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
8045	1F6D	Linear counter overflow	1. The linear counter reached the maximum count value.	1. Adjust the counting range as required, or use the ring counting mode.
8046	1F6E	Linear counter underflow	1. The linear counter reached the minimum count value.	1. Adjust the counting range as required, or use the ring counting mode.
8047	1F6F	Module ECAT communication error	1. A communication error has occurred in the module.	1. Check the communication status of the module.
8048	1F70	Bus encoder axis instruction does not support the virtual axis mode.	1. The axis to execute the instruction was configured a virtual axis.	1. Modify the axis type in the software backend.
8049	1F71	The probe ID number for the encoder axis is not within the supported range.	1. The set value of the input parameter ProbeID for the bus encoder axis was not 0 or 1. 2. The set value of the input parameter ProbeID for the local encoder axis was not 0.	1. Select an appropriate probe ID as required.
8050	1F72	The probe trigger function is configured repeatedly on the bus encoder axis.	1. When this instruction was triggered, the probe channel was occupied.	1. Modify the value of the input parameter ProbeID.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
8051	1F73	The trigger edge parameter for the bus encoder axis is not within the supported range.	1. The set value of the input parameter TriggerEdge was not 0, 1, or 2.	1. Modify the set value of the input parameter TriggerEdge.
8052	1F74	The trigger source parameter is not within the supported range.	1. The set value of the input parameter TerminalSource was not 0 or 1. Note: The local encoder axis only supports a trigger source value of 0.	1. Modify the set value of the input parameter TerminalSource.
8053	1F75	The trigger mode parameter is not within the supported range.	1. The set value of the input parameter TrigerMode was not 0 or 1.	1. Modify the set value of the input parameter TrigerMode.
8054	1F76	The linear axis or rotary axis window position parameters are invalid.	1. In linear mode, the set value for the input parameter FirstPosition was greater than or equal to that for LastPosition. 2. In rotation mode, the set value for the input parameter FirstPosition was equal to that for LastPosition.	1. Modify the set value for the input parameter FirstPosition or LastPosition.
8055	1F77	The position set in the window parameters for the linear axis exceeds the limit range.	1. In linear mode, the set value of the input parameter FirstPosition was greater than the positive limit value set in the backend, or the set value of LastPosition was less than the negative limit value set in the backend.	1. Modify the set value for the input parameter FirstPosition or LastPosition.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
8056	1F78	The probe trigger position is out of the set range.	1. For a linear axis, the value of the current instruction position was less than the set value of the input parameter "firstPosition" or greater than the set value of the input parameter "lastPosition". 2. For a rotary axis, when the input parameter "firstPosition" was greater than "lastPosition", the current instruction position was greater than "lastPosition" and less than "firstPosition"; when the input parameter "firstPosition" was less than "lastPosition", the current instruction position was greater than "lastPosition" and less than 360°, or the current instruction position was greater than 0 and less than "firstPosition".	1. Modify the set value of the input parameter "firstPosition" or of the input parameter "lastPosition".
8057	1F79	Probe trigger interrupted	1. While waiting for a probe trigger, the process was interrupted by a new ENC_TouchProbe instruction.	1. Check if a new ENC_TouchProbe instruction is triggered while waiting for a probe trigger.
8058	1F7A	Module error not cleared within timeout	1. After 200 task cycles of the instruction execution, the error has not been cleared yet.	1. Check the error status of the servo.
8059	1F7B	Switching axis objects is not permitted during encoder axis operation.	1. The axis object was switched during the execution of the instruction.	1. Do not switch axis objects during the execution of the instruction.
8060	1F7C	The counter is already running.	1. The counter was already running when a new counting instruction was triggered.	1. Modify the user program to avoid re-triggering this instruction.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
8061	1F7D	The preset value was less than 0 in ring mode.	1. In rotation mode, the set value of the input parameter Position was less than 0.	1. Modify the set value of the input parameter Position.
8062	1F7E	The ENC_Preset instruction is re-triggered.	1. When the instruction was executing, another instance of the identical instruction has been triggered.	1. During the execution of the instruction, do not trigger another instance of this instruction.
8063	1F7F	The instruction is re-triggered.	1. When the instruction was executing, another instance of the identical instruction has been triggered.	1. During the execution of the instruction, do not trigger another instance of this instruction.
8064	1F80	The one-dimensional output comparison mode is out of range.	1. When the axis type was set to a local encoder axis, the set value of the input parameter Mode was not 0, 1, 2 or 3.	1. Modify the set value of the input parameter Mode.
8065	1F81	The local encoder axis preset enable function is not active.	1. When the axis type was set to a local encoder axis, the preset enable function was not active in the software backend before the instruction was triggered.	1. Enable the preset enable function in the software backend.
8066	1F82	The local encoder axis preset terminal is not selected.	1. When the axis type was set to a local encoder axis, the preset terminal in the software backend is not selected before the execution of the instruction.	1. Activate the preset enable function in the software backend and select the corresponding terminal.
8067	1F83	The one-dimensional comparison output nodes are out of order or overlapping.	1. The point values in the arrays used for one-dimensional array comparison were out of order or overlapping.	1. Check if the values in the arrays used for one-dimensional array comparison contained any out-of-order or overlapping points.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
8070	1F86	The communication is in the non-operational state.	1. The axis was in a non-operational state.	1. Check the operation state of the axis.
8071	1F87	The axis pointer is null.	1. The axis pointer controlled by the instruction was null.	1. Check if the axis pointer controlled by the instruction is null. If yes, select an appropriate axis to execute the instruction as required.
8073	1F89	The input parameter PlusePerCycle for the instruction was set improperly.	1. The set value of the input parameter PlusePerCycle was less than or equal to 0.	1. Modify the set value of the input parameter PlusePerCycle.
8074	1F8A	Improper DisPerCycle setting	1. The set value of the input parameter DisPerCycle was invalid.	1. Adjust the DisPerCycle parameter to a value between 0.01 and 9999999.
8075	1F8B	Setting up the axis operation mode during counting is prohibited.	1. The instruction was triggered during the execution of ENC_Counter.	1. Do not trigger the instruction during the execution of ENC_Counter.
8076	1F8C	The input parameters of the comparison instruction are out of range.	1. The input parameter Size of the ENC_ArrayCompare instruction was out of range. 2. The input parameter Step of the ENC_StepCompare instruction was out of range.	1. The input parameter Size of the ENC_ArrayCompare instruction must be within the range of 1-1000. 2. The input parameter Step of the ENC_StepCompare instruction must be greater than 0.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
8077	1F8D	The value of the start position is greater than that of the end position, and the step size can only be set to a negative value.	1. When the set value of the input parameter StartPosition was greater than that of EndPosition, the set value of the input parameter Step was greater than 0.	1. Modify the set value of the input parameter StartPosition or of the input parameter EndPosition. 2. Modify the set value of the input parameter Step.
8078	1F8E	The value of the start position is less than that of the end position, and the step size can only be set to a positive value.	1. When the set value of the input parameter StartPosition was less than that of EndPosition, the set value of the input parameter Step was less than 0.	1. Modify the set value of the input parameter StartPosition or of the input parameter EndPosition. 2. Modify the set value of the input parameter Step.
8079	1F8F	The start position is equal to the end position.	1. The set values of the input parameters StartPosition and EndPosition were equal.	1. Modify the set values of the input parameters StartPosition and EndPosition.
8080	1F90	The operation mode setting only supports local encoder axes.	1. The axis type set in the software backend was not a local encoder axis.	1. Set the axis type to a local encoder axis in the software backend.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
8081	1F91	The probe channel is not enabled.	1. The probe channel corresponding to the axis was not enabled.	1. Enable the probe channel corresponding to the axis on the FA side. 2. There is no probe channel for the virtual axis in the PAC device. If the main axis is set to a virtual axis, the driver mode cannot be selected.
8082	1F92	Probe 1 terminal not configured	1. The corresponding terminal for touch probe 1 has not been configured.	1. Enable the probe channel corresponding to the axis on the FA side. Select the corresponding terminal.
8083	1F93	Probe 2 terminal not configured	1. The corresponding terminal for touch probe 2 has not been configured.	1. Enable the probe channel corresponding to the axis on the FA side. Select the corresponding terminal.
8084	1F94	The comparison output instruction is executing.	1. Another comparison output instruction is executing.	1. Wait for other comparison output instructions to complete before executing the current comparison output instruction.
8085	1F95	The "Comparison output enable" option is not enabled for the comparison output instruction.	1. The "Comparison output enable" option was not enabled on the FA side.	1. Enable the "Comparison output enable" option on the FA side.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
8086	1F96	The comparison output terminal is not selected for the comparison output instruction.	1. The comparison output terminal was not selected on the FA side.	1. On the FA side, enable the "Comparison output enable" option and select the corresponding comparison output terminal.
8087	1F97	The comparison point of the comparison instruction exceeds the rotation cycle.	1. The input parameter Postion of the ENC_Compare instruction exceeded the range of the rotation cycle for the executing axis. 2. The input parameter ArrayCompare of the ENC_ArrayCompare instruction contained the points that exceeded the range of the rotation cycle for the executing axis. 3. The input parameters StartPosition or EndPosition of the ENC_StepCompare instruction exceeded the range of the rotation cycle for the executing axis.	1. The comparison point of the comparison instruction must not exceed the range of the rotation cycle for the executing axis.
8088	1F98	The comparison point of the comparison instruction is outside the position limits.	1. The input parameter Position of the ENC_Compare instruction was outside the axis position limits. 2. The input parameter ArrayCompare of the ENC_ArrayCompare instruction contained the points that were outside the axis position limits. 3. The input parameters StartPosition and EndPosition of the ENC_StepCompare instruction were outside the axis position limits.	1. The comparison points of the comparison instruction must not be outside the axis position limits.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
8090	1F9A	Local pulse axis homing timeout	1. The homing process for the local pulse axis timed out. The actual duration exceeded the configured homing timeout value.	1. According to the requirements, set an appropriate homing timeout duration in Axis -> Homing Settings on the FA side.
8091	1F9B	Invalid limit triggered during homing	1. An unexpected limit signal was detected during the local pulse axis homing process.	1. Check if an invalid limit signal was triggered based on the local pulse axis homing mode and its limit signals.
8092	1F9C	The homing mode does not enable the negative limit switch.	1. The negative limit switch needs to be enabled in the selected homing mode for the local pulse axis.	1. On the FA side, enable the negative limit switch in the Axis -> Homing Settings interface.
8093	1F9D	The homing mode does not enable the positive limit switch.	1. The positive limit switch needs to be enabled in the selected homing mode for the local pulse axis.	1. On the FA side, enable the positive limit switch in the Axis -> Homing Settings interface.
8094	1F9E	The homing mode does not enable the homing switch.	1. The homing switch needs to be enabled in the selected homing mode for the local pulse axis.	1. On the FA side, enable the homing switch in the Axis -> Homing Settings interface.
8095	1F9F	The pulse width parameter is invalid.	1. The value of the parameter "PulseWidth" on the instruction side was less than or equal to 0 or greater than 9999.	1. Modify the value of the parameter "PulseWidth" on the instruction side.
8096	1FA0	When the axis is a pulse axis, the output device is neither Y0 nor Y1.	1. When the instruction for the pulse axis was executed, the dual terminals were selected for the output device.	1. Select the appropriate output device Y0 or Y1 as required.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
8097	1FA1	There are no active overlay instruction or phase compensation instructions.	1. There were no active phase compensation instructions.	1. The MC_StopPhasing instruction is not required when no phase compensation instructions are active.
8098	1FA2	The touch probe function is not set for the Xn0 channel.	1. When the ENC_TouchProbe instruction was used for a bus encoder axis, the Xn0 channel has not been set to the touch probe function. (n represents the channel value.)	1. Set the Xn0 channel to the touch probe function as required.
8099	1FA3	The Xn1 channel is not set to the touch probe function.	1. When the ENC_TouchProbe instruction was used for a bus encoder axis, the Xn1 channel has not been set to probe function. (n represents the channel value.)	1. Set the Xn1 channel to the touch probe function as required.
8100	1FA4	The positive and negative limit settings for the local encoder axis are invalid.	1. After the limits of the local encoder were enabled, the set value of the positive limit must be greater than 0, and the set value of the negative limit must be less than or equal to 0.	1. Set the appropriate positive and negative limit parameters as required.
8101	1FA5	Interrupt parameter out of range	1. The input parameter InterruptMap of the ENC_Compare instruction was outside the range [0,16]. 2. The input parameter InterruptMap of the ENC_ArrayCompare instruction was outside the range [0,16]. 3. The input parameter InterruptMap of the ENC_StepCompare instruction was outside the range [0,16].	1. Set the input parameter InterruptMap of the ENC_Compare instruction within the range [0,16]. 2. Set the input parameter InterruptMap of the ENC_ArrayCompare instruction within the range [0,16]. 3. Set the input parameter InterruptMap of the ENC_StepCompare instruction within the range [0,16].

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
8102	1FA6	The input parameter Channel for the ENC_Compare instruction is out of range.	The input parameter Channel of the ENC_Compare instruction only accepts the value 1 or 2, corresponding to 2HCE channels 1 and 2, respectively.	The input parameter Channel for ENC_Compare must be 1 or 2. Ensure that the Channel parameter is within this range.
8103	1FA7	When the comparison output is enabled using the ENC_Compare instruction, the comparison output time PDO is not configured.	When OutputEnable of the ENC_Compare instruction is TRUE, the corresponding PDO parameter must be configured so that the comparison output will take effect.	Confirm whether the corresponding PDO is configured: Configure 7006h.01h for comparison channel 1 of 2HCE channel 0. Configure 7007h.01h for comparison channel 2 of 2HCE channel 0. Configure 7006h.02h for comparison channel 1 of 2HCE channel 1. Configure 7007h.02h for comparison channel 2 of 2HCE channel 1.
8104	1FA8	No responding PDO is configured for the ENC_Compare instruction.	Before using the ENC_Compare instruction, configure the corresponding PDO mapping.	Confirm whether the corresponding PDO is configured: Configure 7003h.01h and 600E.01h for 2HCE channel 0. Configure 7004h.01h for compare channel 1. Configure 7005h.01h for compare channel 2. Configure 7003h.02h and 600E.02h for 2HCE channel 1. Configure 7004h.02h for compare channel 1. Configure 7005h.02h for compare channel 2.

Fault Code (decimal)	Fault Code (hexadecimal)	Error Message	Cause	Solution
8105	1FA9	The probe type parameter is not within the supported range.	The ENC_TouchProbe instruction is used for a bus encoder axis, but probetype is set to the value greater than 2.	The value of the probetype pin of the ENC_TouchProbe instruction must be limited to 2.
8106	1FAA	The array elements of the multi-dimension comparison output are empty.	The array elements of the multi-dimension comparison output cannot be empty.	1. Bind the input variables to the ENC_ArrayCompare and ArrayCompare pins.
8107	1FAB	The number of comparison values in the multi-dimension comparison output exceeds the array length.	The number of comparison values in the multi-dimension comparison output cannot exceed the array length.	1. Modify the number of comparison values in the multi-dimension comparison output to be less than the array length.
8108	1FAC	The element type of the multi-dimension comparison output array is incorrect.	The element type of the multi-dimension comparison output array must be LREAL.	1. Modify the array element type to LREAL.
8109	1FAD	The input Parameter of the ENC_Compare instruction is out of range.	The value of the Parameter input for the ENC_Compare instruction is out of the allowed range (0-65535).	Set the Parameter input of the ENC_Compare instruction to a value between 0 and 65535.

18.3.3.9 ParameterNumber

ParameterNumber code of the parameter read/write function block.

Name	Type	Init	Comment	Read/Write Property	ParameterNumber
Axis identifier.	-	-	-	R: ReadBoolParameter r: ReadParameter W: WriteBoolParameter w: WriteParameter	-
bVirtual	BOOL	FALSE	Is it a virtual axis? 0 indicates real axis, and 1 indicates virtual axis.	R W	1020
bCommunication	BOOL	FALSE	Bus communication establishment flag: Set to TRUE if the communication is successfully established; otherwise it will be set to FALSE if the communication fails.	R	1021
wCommunicationState	WORD	16#FFFF	Status of communication state machine. Use numbers to represent different states. For example, 10 indicates no connection established, and 100 indicates the connection is established.	r	1022

Name	Type	Init	Comment	Read/Write Property	ParameterNumber
eAxisState	18.3.3.2.2.5 MC_AXIS_STATE: Axis State Machine Enumeration	AXIS_POWER_OFF	Axis state machine: 0: AXIS_POWER_OFF (Axis power off state). 1: AXIS_ERRORSTOP (Axis error stop). 2: AXIS_STOPPING (Axis decelerate-to-stop state). 3: AXIS_STANDSTILL (Axis standstill state). 4: AXIS_DISCRETE_MOTION (Axis discrete motion state). 5: AXIS_CONTINUOUS_MOTION (Axis continuous motion state). 6: AXIS_SYNCHRONIZED_MOTION (Axis continuous motion state). 7: AXIS_HOMING (Axis homing state).	r	1025
bAxisEnableOn	BOOL	FALSE	Enable axis	R	1040
bAxisEnableState	BOOL	FALSE	Axis enable signal state, corresponding to the value of 6041 or the enable state of a local axis.	R	1041
fTaskCycle	LREAL	0.004	Set value of a task cycle.	r w	1081
uiDriveError	UINT	0	Driver fault code	r	1100
bError	BOOL	FALSE	Axis error	R	1103
dwErrorID	DWORD	0	Axis error identification code, including driver errors, instruction errors, configuration errors and so on.	r	1104
bForwardDirectionLimit	BOOL	FALSE	Forward rotation direction limit	RW	1120
bBackwardDirectionLimit	BOOL	FALSE	Reverse rotation direction limit	RW	1121
bResetFollowingErrorFlag	BOOL	FALSE	Is the following error currently being cleared?	R	1124

Name	Type	Init	Comment	Read/Write Property	ParameterNumber
fMaxDecelDistance	LREAL	0	Maximum deceleration distance.	r w	1127
fSwLimitDeceleration	LREAL	0	Deceleration rate in position limit state.	r w	1128
eDirection	18.3.3.2.2.9 MC_DIRECTION: Motion Direction Enumeration	DIR_POSITIVE	Current motion direction of an axis: 0: DIR_SHORTEST (Shortest way). 1: DIR_POSITIVE (Positive direction). 2: DIR_NEGATIVE (Negative direction). 3: DIR_CURRENT (Current direction). 4: DIR_NODIRECTION (No direction). 5: DIR_FASTEST (Fastest motion direction).	r	1129
eRampType	18.3.3.2.2.15 MC_RAMPTYPE: Ramp Type Enumeration	TRAPEZ	Acceleration mode: 0: TRAPEZ (Trapezoidal curve). 1: CURVE (S-curve). 2: SINSQUARE (Sin2 curve).	r w	1130
eControllerMode	18.3.3.2.2.7 MC_CONTROLLER_MODE: Controller Mode Enumeration	MC_CSP	Axis control mode: 8: MC_CSP (Cyclic synchronous position mode). 9: MC_CSV (Cyclic synchronous velocity mode). 10: MC_CST (Cyclic synchronous torque mode).	r	1131
eRealControllerMode	18.3.3.2.2.7 MC_CONTROLLER_MODE: Controller Mode Enumeration	MC_CSP	Actual control mode obtained from an axis: 8: MC_CSP (Cyclic synchronous position mode). 9: MC_CSV (Cyclic synchronous velocity mode). 10: MC_CST (Cyclic synchronous torque mode).	r	1132
fFactorVel	LREAL	1	Velocity scaling factor	r w	1151

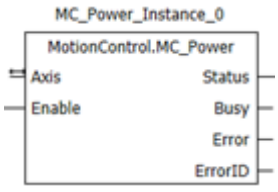
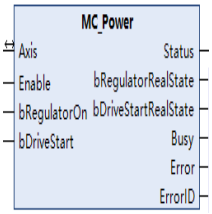
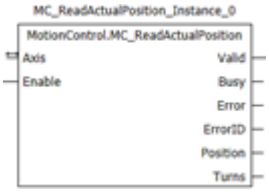
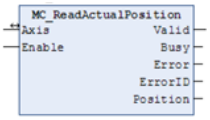
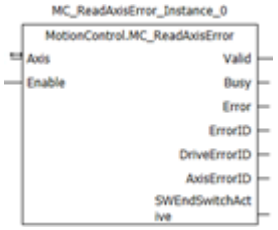
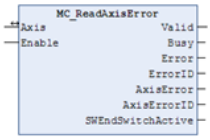
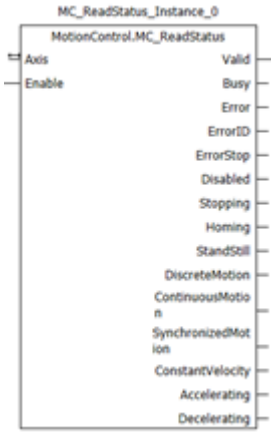
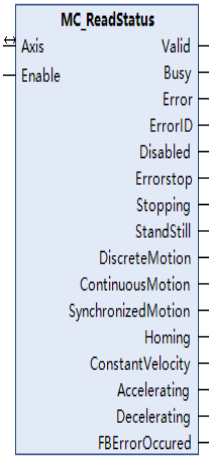
Name	Type	Init	Comment	Read/Write Property	ParameterNumber
fFactorAcc	LREAL	1	Acceleration scaling factor	r w	1152
iMovementType	INT	1	Axis motion type (0: rotational motion, 1: linear motion)	rw	1190
fSetPosition	LREAL	0	Instruction position set in this cycle	r	1191
fActPosition	LREAL	0	Feedback value: the position where the motor operated to in the previous cycle.	r	1192
fPositionPeriod	LREAL	0	Configured module length (unit)	r	1201
dwRatioTechUnitsDenom	DWORD	0	Gear ratio denominator	r w	1204
fRatioTechUnitsNum	LREAL	0	Gear ratio numerator	r w	1205
fSetVelocity	LREAL	0	Instruction velocity set in this cycle	r	1220
fActVelocity	LREAL	0	Feedback value: operating velocity, that is, the motor velocity from the previous cycle.	r	1221
fSWMaxVelocity	LREAL	0	Maximum axis velocity limited by the software.	r w	1225
bConstantVelocity	BOOL	FALSE	Flag indicates the axis is operating at a constant velocity.	R	1226
fSetAcceleration	LREAL	0	Instruction acceleration set in this cycle	r	1240
fActAcceleration	LREAL	0	Feedback value: operating acceleration, that is, the motor acceleration from the previous cycle.	r	1241
bAccelerating	BOOL	FALSE	Flag indicating that the axis is accelerating.	R	1243
bDecelerating	BOOL	FALSE	Flag indicating that the axis is decelerating.	R	1244
fSWMaxAcceleration	LREAL	0	Maximum axis acceleration limited by the software.	r w	1245

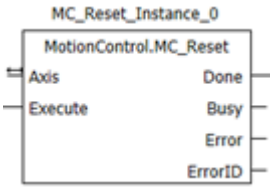
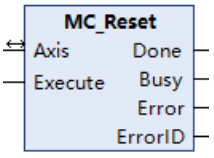
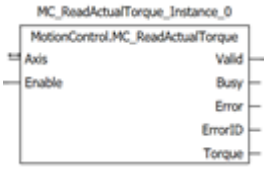
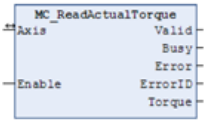
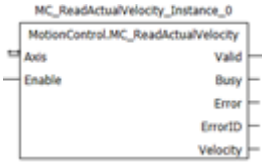
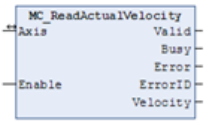
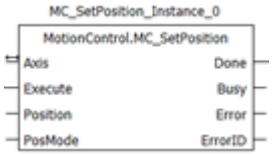
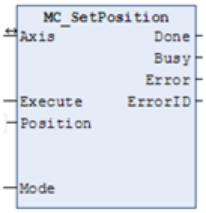
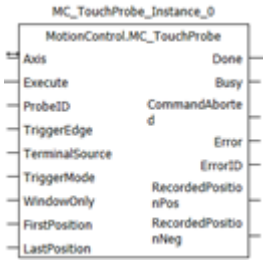
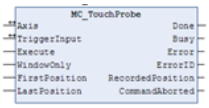
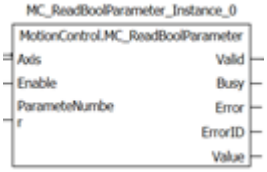
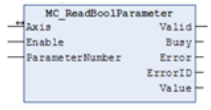
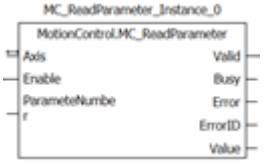
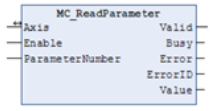
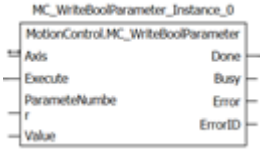
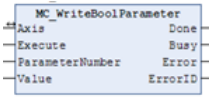
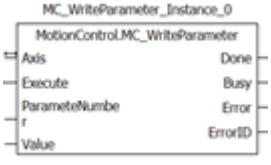
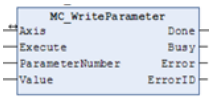
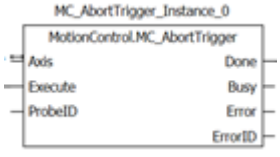
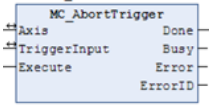
Name	Type	Init	Comment	Read/Write Property	ParameterNumber
fSWMaxDeceleration	LREAL	0	Maximum deceleration limited by the software. When the axis deceleration exceeds this limit but does not trigger a brake or error, this value is used to calculate the deceleration distance after an error occurs.	r w	1246
fSWMaxJerk	LREAL	0	Maximum axis jerk limited by the software.	r w	1247
fSetTorque	LREAL	0	Torque set in this cycle	r	1260
fActTorque	LREAL	0	Feedback value: motor feedback torque	r	1261
fMaxPositiveTorque	LREAL	0	Maximum positive torque limit for the servo.	w	1262
fMaxNegativeTorque	LREAL	0	Maximum negative torque limit for the servo.	w	1263
fSWLimitPositive	LREAL	0	Position limit in the positive direction	w	1300
fSWLimitNegative	LREAL	0	Position limit in the negative direction	w	1301
bSWLimitEnable	BOOL	FALSE	Software position limit flag. TRUE: Enabled.	R W	1303
bHWLimitEnable	BOOL	TRUE	Hardware position limit flag. TRUE: Enabled.	R W	1304
fJogMaxVelocity	LREAL	0	Jog velocity	r w	1340
fJogMaxAcceleration	LREAL	0	Jog acceleration	r w	1341
fInPosRange	LREAL	0	• MotionControl library version 2.3.0.0 or earlier: Not supported • MotionControl library version 2.4.0.0 or later: Positioning range	r w	1342

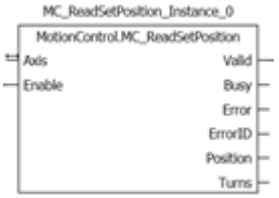
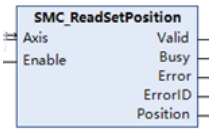
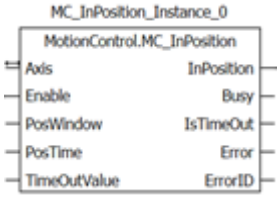
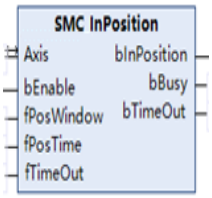
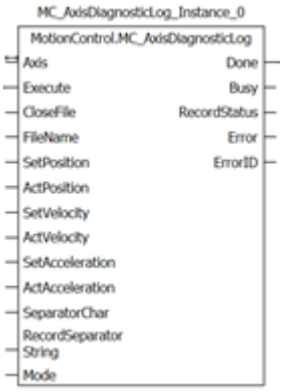
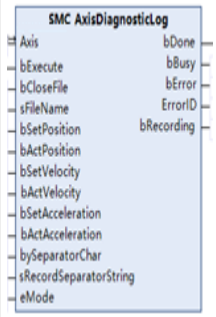
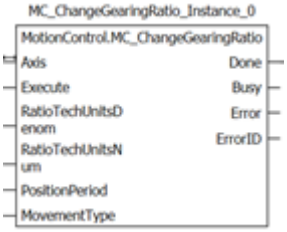
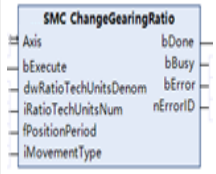
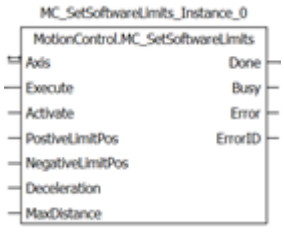
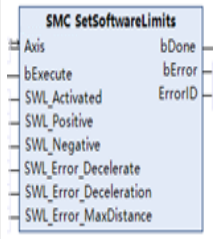
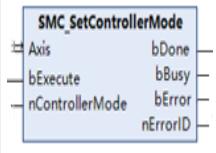
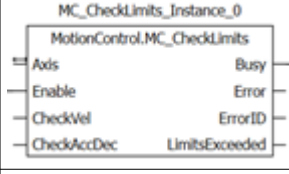
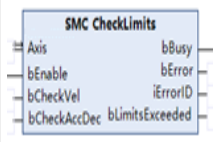
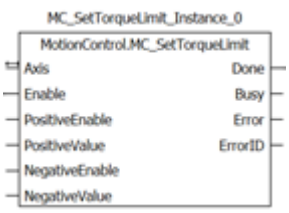
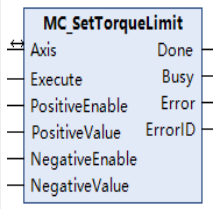
Name	Type	Init	Comment	Read/Write Property	ParameterNumber
uiInPosTime	UINT	0	• MotionControl library version 2.3.0.0 or earlier: Not supported • MotionControl library version 2.4.0.0 or later: Positioning check time	r w	1343

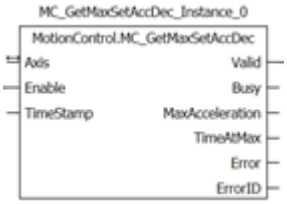
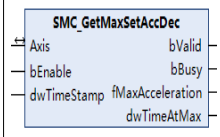
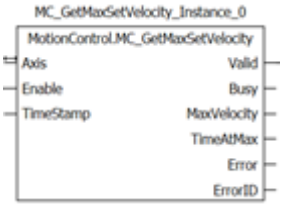
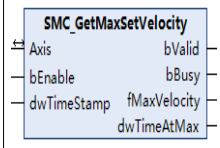
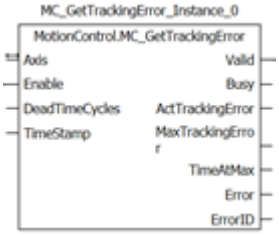
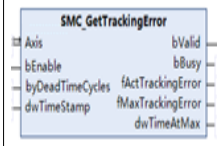
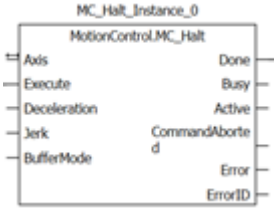
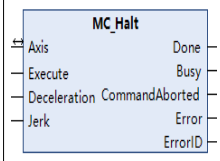
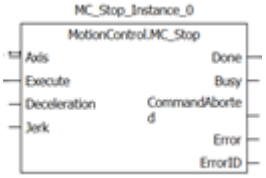
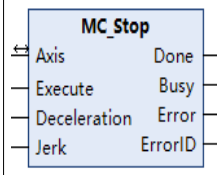
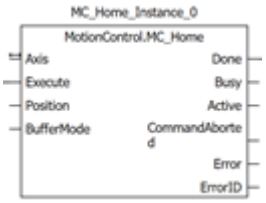
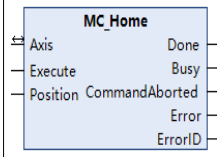
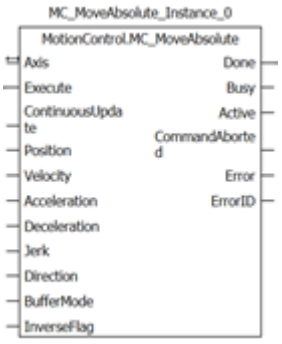
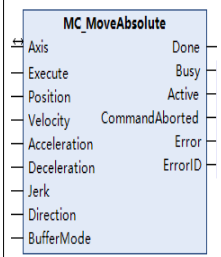
18.3.3.10 Pin Differences

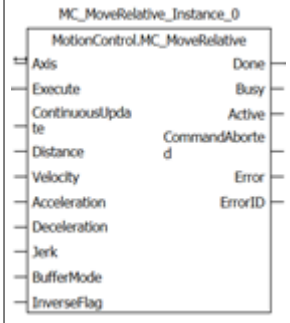
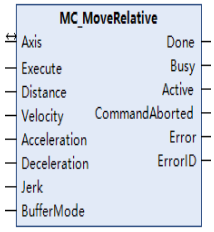
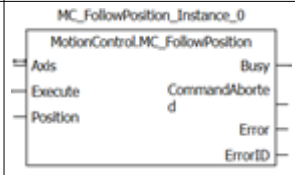
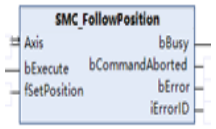
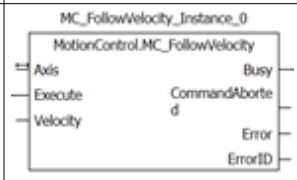
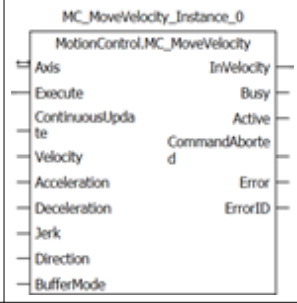
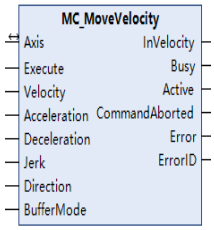
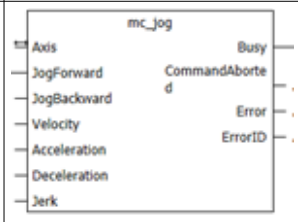
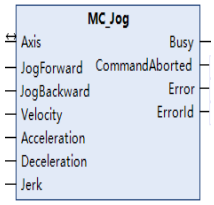
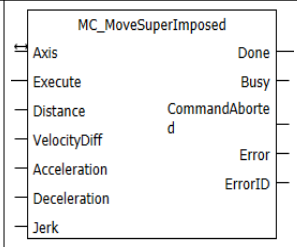
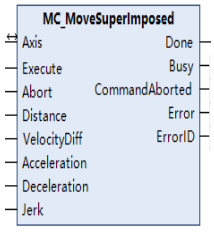
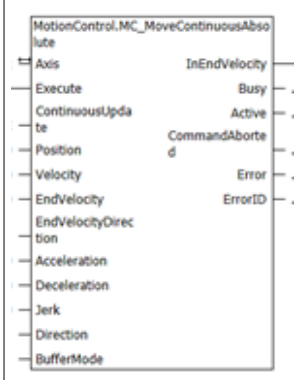
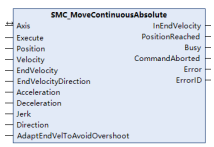
Explanation of pin differences on the instruction side:

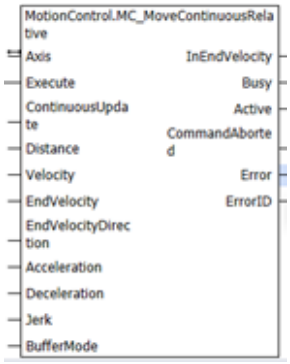
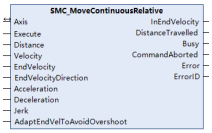
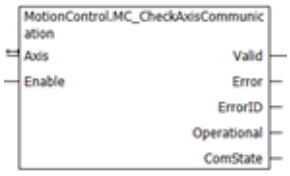
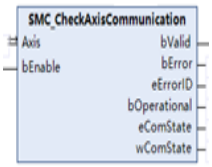
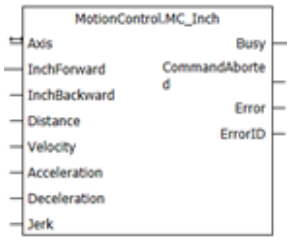
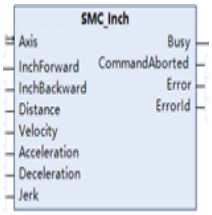
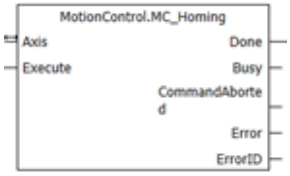
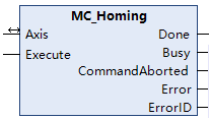
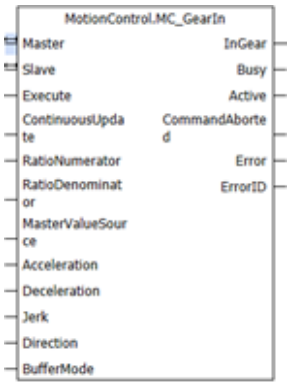
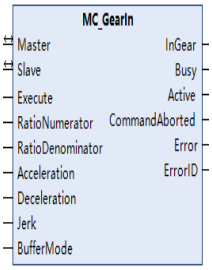
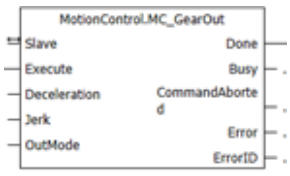
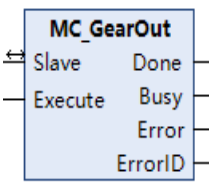
Name	SoftMotion Instruction	Medium Instruction	Explanation of Difference Points Based on Medium Instruction
MC_Power			Missing: bRgulationOn bDriverStart bRgulationRealState bDriverStartRealState
MC_ReadActualPosition			Added: Turns
MC_ReadAxisError			Missing: AxisError
MC_ReadStatus			Missing: FBErrorOccured

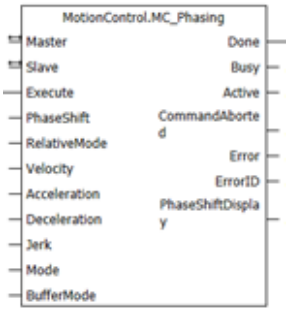
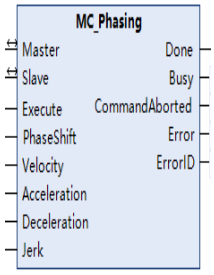
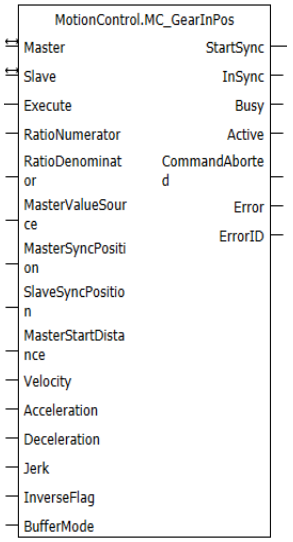
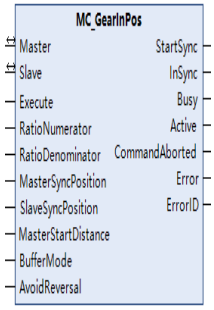
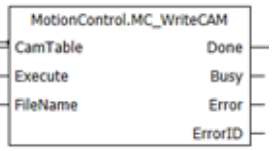
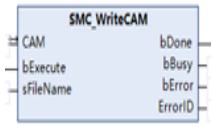
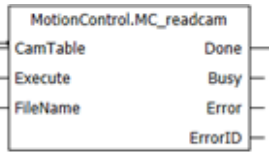
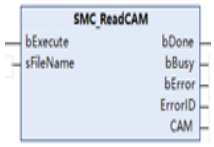
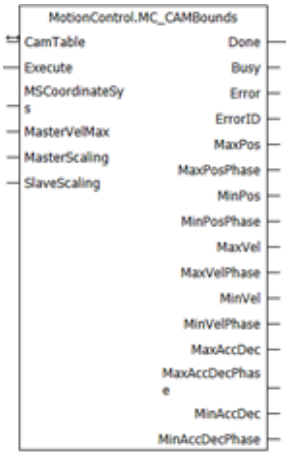
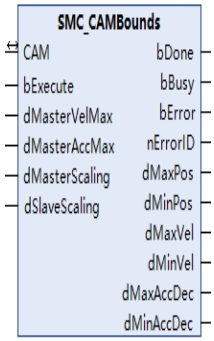
Name	SoftMotion Instruction	Medium Instruction	Explanation of Difference Points Based on Medium Instruction
MC_Reset			-
MC_ReadActualTorque			-
MC_ReadActualVelocity			-
MC_SetPosition			-
MC_TouchProbe			List the parameters from the TRIGGER_REF structure into the input parameters, and add position parameters for rising and falling edge latching.
MC_ReadBoolParameter			-
MC_ReadParameter			-
MC_WriteBoolParameter			-
MC_WriteParameter			-
MC_AbortTrigger			Missing: TruggerInput

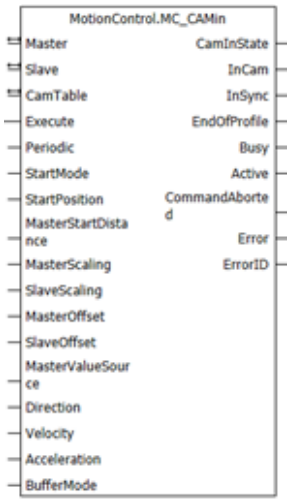
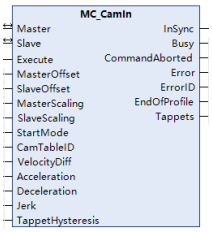
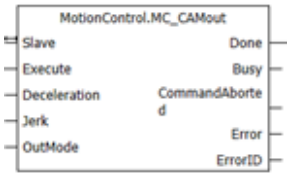
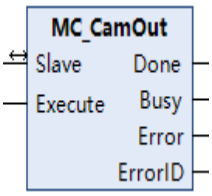
Name	SoftMotion Instruction	Medium Instruction	Explanation of Difference Points Based on Medium Instruction
MC_ReadSetPosition			Added: Turns
MC_InPosition			Added: Error ErrorID
MC_AxisDiagnosticLog			-
MC_ChangeGearingRatio			-
MC_SetSoftwareLimits			Missing: SWL_Error_Decelerate Added: Busy
MC_SetControllerMode			-
MC_CheckLimits			-
MC_SetTorqueLimit			-

Name	SoftMotion Instruction	Medium Instruction	Explanation of Difference Points Based on Medium Instruction
MC_GetMaxSetAccDec			Added: Error ErrorID
MC_GetMaxSetVelocity			Added: Error ErrorID
MC_GetTrackingError			Added: Error ErrorID
MC_Halt			Added: BufferMode Active
MC_Stop			Added: CommandAborted
MC_Home			Added: BufferMode Active
MC_MoveAbsolute			Added: ContinuousUpdate InverseFlag

Name	SoftMotion Instruction	Medium Instruction	Explanation of Difference Points Based on Medium Instruction
MC_MoveRelative			Added: ContinuousUpdate InverseFlag
MC_FollowPosition			-
MC_FollowVelocity			-
MC_MoveVelocity			Added: ContinuousUpdate
MC_Jog			-
MC_MoveSuperImposed			Removed: Abort
MC_MoveContinuousAbsolute			Missing: AdaptEndVelToAvoidOversshoot PositionReached Added: ContinuousUpdate BufferMode Active

Name	SoftMotion Instruction	Medium Instruction	Explanation of Difference Points Based on Medium Instruction
MC_MoveContinuous Relative			Missing: AdaptEndVelToAvoidOverssshoot DistanceTravelled Added: ContinuousUpdate BufferMode Active
MC_CheckAxisComm unication			Missing: wComState
MC_Inch			-
MC_Homing			-
MC_GearIn			Added: ContinuousUpdate MasterValueSource Direction
MC_GearOut			Added: Deceleration Jerk OutMode CommandAborted

Name	SoftMotion Instruction	Medium Instruction	Explanation of Difference Points Based on Medium Instruction
MC_Phasing			Added: RelativeMode Mode BufferMode PhaseShiftDisplay
MC_GearInPos			Added: MasterValueSource Velocity Acceleration Deceleration Jerk BufferMode
MC_WriteCAM			-
MC_ReadCAM			-
MC_CAMBounds			Added: MSCOOrdinateSys MaxPosPhase MinPosPhase MaxVelPhase MinVelPhase MaxAccDecPhase MinAccDecPhase Missing: dMasterAccMax

Name	SoftMotion Instruction	Medium Instruction	Explanation of Difference Points Based on Medium Instruction
MC_Camin			Added: CamTable Periodic StartPosition MasterStartDistance MasterValueSource Direction Velocity BufferMode CamInState InCam Active Missing: CamTableID TappetHysteresis Jerk Tappets
MC_CAMout			Added: Deceleration Jerk OutMode CommandAborted

18.3.3.11 Function Differences

Function Overview of the Platform						
Project Description	Function Block or Instruction	Scenario Description	FA Motion Control Instruction	AM600	H5U	Comparison Result

Abnormalities caused by using the same instance	Four MC_Power	Multiple Power instructions of the same instance	The instruction in the last engineering logic determines the output.	The last module determines the output status. The bRegulatorOn parameter of the last conducting module determines the axis enable status.	N/A	Compared to AM600 'busy' output bit, where all modules can be set, this may not match actual operation results.
	Motion instruction	Multiple motion instructions of the same instance	The instruction will return an error 6010 to avoid user program errors.	Confusion between axis status and output bit status	N/A	FA assists users by performing checks to prevent forgetting to modify instances during copying.
	Background configuration interface	Axis motion parameters	For each axis, motion parameter limits can be set. For each axis, commissioning motion parameters can be set individually.	The motion parameter limits for a single axis cannot be set.	The motion parameter limits for a single axis can be set.	Each axis in FA can be independently configured with motion parameters (such as maximum velocity) to prevent motion hazards. This individual axis commissioning simplifies the overall tuning process.
Motion Trajectory Planning for Positioning Motion						

Project Description	Function Block or Instruction	Scenario Description	FA Motion Control Instruction	AM600	H5U	Comparison Result
Rotary axis	Moveabsolute and Moverelative	Position	1. Position and revolution count can be monitored 2. Absolute positioning allows target positions beyond one revolution to be specified.	1. Only the current position is available. 2. If the set position exceeds one revolution, the modulo operation will be performed automatically.	1. Only the current position is available. 2. If the set position exceeds one revolution, the modulo operation will be performed automatically.	1. The total distance traveled can be determined by monitoring the number of revolutions. 2. The application method of automatic modulo operations will be more flexible.
Motion direction	Moveabsolute	Available directions on a rotary axis	0. DIR_SHORTEST 1. DIR_POSITIVE 2. DIR_NEGATIVE 3. DIR_CURRENT 4. DIR_NODIRECTION 5. DIR_FASTEST	1. Positive 2. Negative 3. Current 4. Shortest 5. Fastest	1. Positive 2. Negative 3. Current 4. Shortest	1. AM600 supports the Fastest direction mode, but there is an issue with the fastest path selection. 2. FA supports the No_Direction mode for absolute positioning to a position that is not in the current revolution.

Reverse function	Moveabsolute and Moverelative	When the initial velocity is too high, the deceleration of the instruction is too small to reverse before reaching the target position.	InveserFlag 1. Reverse allowed (According to the instruction motion parameters, first decelerate, and then reverse to target position.) 2. Reverse prohibited (The instruction adaptively adjusts motion parameters to ensure deceleration distance matches the set target displacemen t.)	Only reverse allowed	Only reverse prohibited	FA provides two modes including reverse allowed and reverse prohibited, covering various complex and abnormal operating conditions.
Ease of Use						
Project Description	Function Block or Instruction	Scenario Description	FA Motion Control Instruction	AM600	H5U	Comparison Result
Default motion parameters	Motion instruction	The instructions are called directly with no pin parameters configured.	Default parameter: 0. Error triggered directly.	Default parameter: 0. Error triggered directly.	Default parameter: 0. Error triggered directly.	Future optimization can make the initial parameters follow the default values during operation.

Errors	Universal module	Axis error caused by instruction error	1. For a virtual axis, errors caused by instructions will automatically clear the axis error. 2. For a real axis, whether the axis transitions to the error state due to an instruction-generated error can be configured via an interface option.	On a real axis, errors caused by instructions will not trigger axis errors when the velocity is not 0.	An instruction fault will trigger an axis error.	FA builds on the AM600 design, with further optimization for the virtual axis. This design allows an axis error to be triggered via an instruction, while avoiding the need for users to repeatedly execute the reset instruction unnecessarily.
Instruction-Side Optimization						
Project Description	Function Block or Instruction	Scenario Description	FA Motion Control Instruction	AM600	H5U	Comparison Result
New functions added for motion instructions	Positioning instruction Speed instruction	Trapezoidal motion instruction planning	The minimum precision of LREAL is planned per cycle.	The minimum precision of LREAL is planned per cycle.	H5U ignores some very small displacement values, accumulates several small positions over a single cycle to make up the difference.	FA optimizes the minimum displacement per cycle based on H5U, ensuring smoother motion curves.
		Velocity parameter reading	Instantaneous velocity	The set velocity of AM600 is the instantaneous velocity.	The velocity is the average velocity within a cycle.	FA optimizes the set velocity of H5U, ensuring precise velocity in each cycle.

		Modify parameters during motion	Added ContinueUpdate function: 1. The positioning instruction can modify the velocity and position. 2. The velocity instruction can modify all parameters including direction.	Re-trigger only after modifying the parameters.	Re-trigger only after modifying the parameters.	FA allows motion parameters to be changed at any time while an instruction is executing, offering flexible application and ease of use.
Control mode	Torque control	Torque control	Controlled by separate TorqueControl instruction. The maximum velocity can be set by using this instruction.	TorqueControl instruction. However, the control mode must first be switched to CST. This instruction cannot be executed independently.	Controlled by separate TorqueControl instruction.	FA optimizes the AM600 torque control system, adds a torque control instruction, and includes a velocity safety feature.
	Control mode switching	SetController mode	1. The instruction can be configured via SDO 2. This instruction can be executed during the motion instruction execution.	This instruction can be started only in the standstill state.	No controller mode switching	FA retains the AM600 SetController mode, allowing motion instructions to be executed in CSV mode.

Error clearing	Reset instruction	Clears errors.	It can clear instruction errors, axis errors, and most servo-specified clearable errors, as well as communicati on errors.	Three different instructions are required to clear instruction errors, axis errors, and communicati on errors.	All the errors can be cleared.	FA optimizes the AM600 error clearing function, enabling one-click resolution of all errors including axis status, servo, and communicati on errors.
Motion Instructions and Limits						
Project Description	Function Block and Instruction	Scenario Description	FA Motion Control Instruction	AM600	H5U	Comparison Result

Limit deceleration	Motion instruction	Output status bits and axis status bits	1. For limit deceleration, the system will select the deceleration parameter with the shortest deceleration distance from among the deceleration limit in normal operation, limit deceleration after limit trigger, and maximum limit deceleration distance. 2. The limit deceleration process is in the "Stopping" state. After deceleration is complete, whether the axis enters the error state can be configured via an option in the IDE interface.	1. The deceleration motion planning remains consistent. 2. The limit deceleration directly triggers an ErrorStop.	1. The deceleration motion planning remains consistent. 2. The limit deceleration maintains the previous motion direction.	FA not only implements the limit deceleration process from AM600 and H5U, but also distinguishes between the limit deceleration process and the error-reporting state of the axis after limit deceleration is complete. This allows users to clearly identify the limit deceleration points in their engineering designs.

	Motion instruction	Initial position outside the limit	1. Motion from outside the limit toward the limit is allowed. 2. Attempting to move further away from the limit only triggers an instruction error.	Motion is not allowed when the axis is outside the limit.	When the axis is outside the limit, only inward movement is permitted. Attempting to move further away from the limit triggers an error.	1. FA optimizes motion beyond the AM600 limit switches, ensuring that instructions can be executed to return the system to its original position when the axis is outside the limit range. 2. FA has optimized the response to limit switch collisions, reducing axis errors in the instruction interface. This feature can be selected based on specific application scenarios, eliminating the need for users to manually clear errors.

	Motion instruction	Insufficient limit deceleration distance	Adaptively adjust the deceleration to ensure the speed is reduced to zero at the limit boundary.	Maintaining the current deceleration will eventually cause the axis to exceed the limit boundary.	Maintaining the current deceleration will eventually cause the axis to exceed the limit boundary.	FA supports users in ensuring the axis stops at the limit boundary under various special and complex conditions. It provides a safety net for abnormal operations, guaranteeing absolute safety during use.
	Motion instruction	Position jumps outside the limit	Immediate emergency stop	Slow deceleration	Slow deceleration	
Set Absolute and Relative Positions Using Setposition						
Project Description	Function Block or Instruction	Scenario Description	FA Motion Control Instruction	AM600	H5U	Comparison Result
Motion instruction settings	Moveabsolute	Using a set position in absolute positioning	Calculates the final position and replan the trajectory in the new coordinate system.	The motion trajectory remains unchanged, and the final position does not match the set position.	Setposition instruction is not supported during motion.	1. FA can correct the current position coordinates at any time, providing greater operational convenience.
	Moverelative	Using a set position in relative positioning	Maintain the relative displacement.	The relative displacement remains unchanged.		
Buffer Mode						
Project Description	Function Block or Instruction	Scenario Description	FA Motion Control Instruction	AM600	H5U	Comparison Result

Buffer mode	Motion instruction	Check the buffer mode of the instruction.	1. Aborted 2. Buffered 3. BlendingHigh 4. BlendingNext 5. BlendingLow 6. BlendingPrevious	Only Aborted is supported.	Only relative positioning provides a dedicated function block that supports Buffered execution.	FA supports all PLCOpen buffer modes, enhancing continuity between instructions.
Abnormal buffering	Between positioning instructions and velocity instructions.	Buffering in Blending mode after a velocity instruction	Automatically optimized to Buffered mode	N/A	N/A	FA has implemented adaptive optimizations for logic errors in the buffer, enhancing user convenience.
Cam						
Project Description	Function Block or Instruction	Scenario Description	FA Motion Control Instruction	AM600	H5U	Comparison Result

Cam table function	Backend IDE	Cam data	1. Node precision: up to four decimal places 2. Maximum number of cam nodes: 20000 3. Maximum number of dense nodes: 65535	1. Node precision: up to four decimal places 2. Maximum number of cam nodes: 1024	1. Node data precision: up to four decimal places 2. Maximum number of cam nodes: 361	1. The FA configuration environment directly supports checking cam data in dense node type, making import more convenient and intuitive. 2. The data precision of FA is sufficient, and the maximum data length of its instructions is higher than that of H5U and InoProshop.

Data transmission mode	Data exchange	Data transmission and retention function	1. Supports the conversion between key node cam data and dense node data. 2. Import function: Supports importing cam data in both key node and dense node types. 3. Export function: Supports exporting cam data in both key node and dense node types. (File format: CSV) 4. Uploads cam data from the controller.	1. Import function: Supports importing key node data and dense node data. (Data type for dense nodes: cam; data type for key nodes: CSV) 2. Export function: Supports exporting to key node data and dense node data. (Data type for dense nodes: cam; data type for key nodes: CSV)	1. Import function: Only supports the files with special node types. 2. Export function: Only supports the files with special node types. 3. Uploads the cam table from the controller (After a test, the uploaded cam table is used as the cam table parameters during download. The Save instruction is used to store the cam table data during download, instead of the data modified by Generate.)	FA supports data in both key node and dense node types. All exported files are in the widely adopted CSV format, facilitating data exchange with other products. Distinguishing between key node and dense node data enhances data usability and comprehensibility, making it simpler and more comprehensive.

Cam data modification method	GenerateCam instruction	Online modification of cam data	1. Supports WriteCam and ReadCam functions. 2. Allows user-defined cam data to be modified at any time. 3. Provides the Generatecam instruction for modification, which can also be used to validate custom cam data.	1. Modifies user-defined cam data. 2. Uploads the data using WriteCam and ReadCam instructions, allowing it to be modified and then written back.	1. Supports modifying cam data using GenerateCam.	1. FA incorporates all online methods for modifying cam data from H5U and InoProshop, ensuring a consistent user experience. 2. Adds a cam data verification function to ensure the accuracy of user cam data.
Cam curves	Backend IDE	Supported curve connections between cam nodes	1. Straight line 2. Parabola 3. Cubic curve 4. Quintic curve 5. Septic curve 6. Cosine acceleration curve 7. Sine acceleration curve 8. Modified trapezoidal acceleration curve	1. Straight line 2. Quintic curve	1. Straight line 2. Quintic curve	The cam curves significantly enrich the range of multi-order curves that are not available in H5U or InoProshop.

Cam chasing optimization	CamIn	The master and slave axes have an initial position deviation, and the slave axis chases the offset to synchronize.	1. The motion parameters in the instruction ensure continuous position chasing. 2. If a velocity jump occurs at the first node, the dynamic chasing mode can be set to automatically optimize velocity continuity at the start point.	-	1. The position deviation jumps directly.	1. FA ensures continuity of position parameters, preventing stall issues in H5U or InoProshop. 2. The dynamic velocity coupling allows users to ignore velocity jumps during cam curve design, reducing user commissioning difficulty.
Cam triggering mode	CamIn	Start mode	1. Absolute phase trigger 2. Relative phase trigger 3. Immediate trigger	1. Absolute phase trigger 2. Relative phase trigger	1. Absolute phase trigger 2. Relative phase trigger 3. Immediate trigger	FA retains the three triggering modes of H5U, meeting the usage habits of cam users.

Cam deviation compensation	Phase	Slave axis deviation compensation method	1. Motion parameters 2. Master axis phase length	1. Motion parameters	1. Motion parameters	FA has developed its compensation method for master axis phase deviation length. Users no longer need to calculate the entire motion trajectory or various motion parameters. Instead, an object-oriented instruction approach allows users to achieve the desired trajectory without designing complex motion parameters.
New Functions						
Project Description	Function Block or Instruction	Scenario Description	FA Motion Control Instruction	AM600	H5U	Comparison Result
Output the sum or difference of the instruction positions of two axes as a combined instruction position.	MC_Combine axes	Multi-master axis following	New	N/A	N/A	Supports slave axis following mode with multiple master axes.

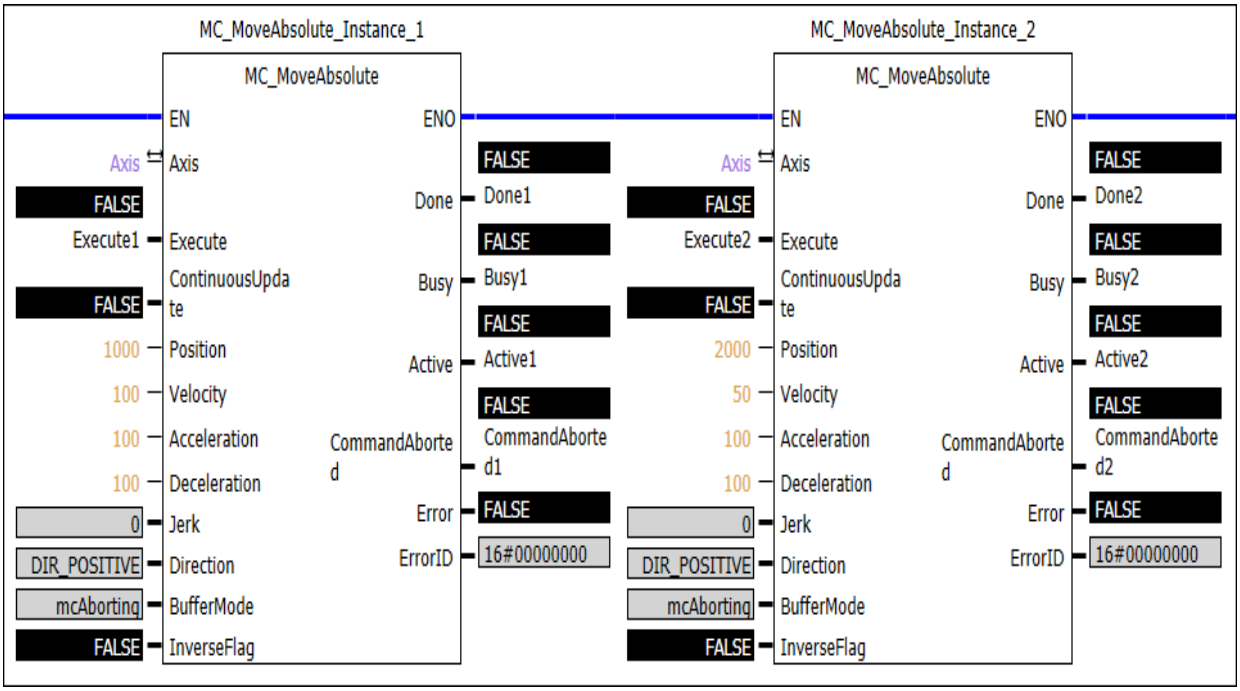
Synchronous positioning	MC_MoveLink	Simplified flying cut	New	N/A	N/A	Simplifies the common four-node flying cut cam into an extremely user-friendly instruction.
Output a specific signal at a specified position	MC_DigitalCamSwitch	Tappet control	Inheritance	N/A	Exists	Inherits the simplified digital cam switch from H5U, streamlining operation while retaining its core functionality.
Periodically record axis parameters to a file.	MC_AxisDiagnosticLog	Parameter recording	Inheritance	Exists	N/A	Logs motion parameters in the controller, facilitating subsequent troubleshooting of trajectories and process adjustments.
Interrupting feed function	MC_MoveFeed	Interrupting feed	New	Exists	N/A	By combining speed, positioning, probing, and relative motion, the engineering design for processes such as marking is simplified.
Reset following error	MC_ResetFollowingError	Following error clearing	New	N/A	N/A	Corrects the servo position deviation.

Execute the emergency stop using a specified method.	MC_ImmediateStop	Emergency stop	New	Exists	N/A	An axis error triggers an emergency stop or shuts down the servo. The introduction of the emergency stop instruction has greatly simplified the engineering design for users.
Monitor axis deviation	MC_InPosition	Accuracy assessment	New	N/A	N/A	Enables users to monitor positioning accuracy more intuitively and easily during motion monitoring.
Halt superimposed motion	MC_HaltSuperImposed	Halting superimposed motion	New	N/A	N/A	Stopping the superimposed motion alone does not affect the main motion, which better aligns with user engineering practices.

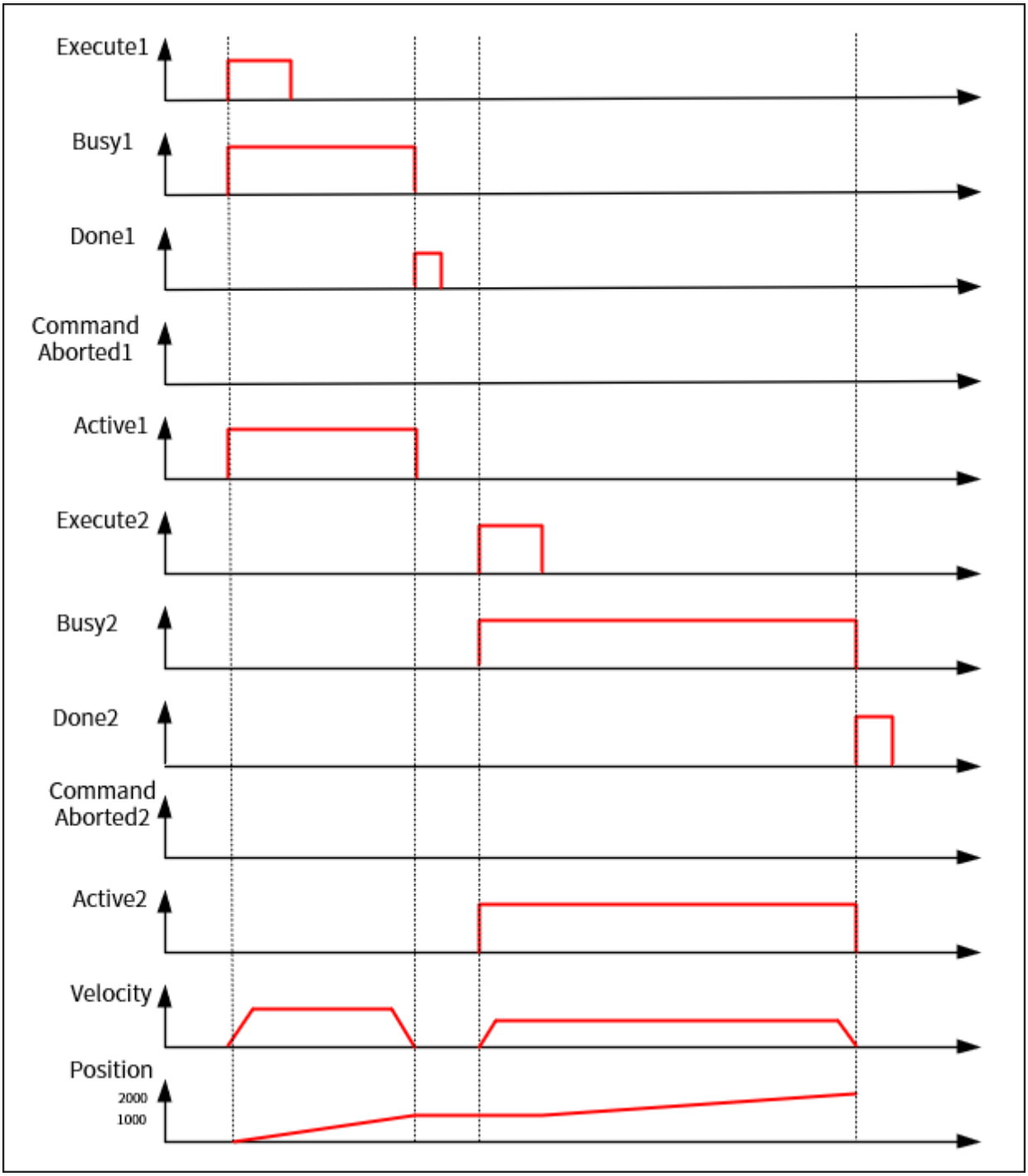
Stop master axis phase shifting	MC_StopPhasing	Stopping phasing compensation	New	N/A	N/A	Stopping only the compensation instruction allows the cam motion to continue running normally, which better aligns with user engineering practices.
Homing operation	MC_Homing WithParameter MC_Homing	Host controller homing	New	N/A	N/A	In addition to the original servo-based homing, a new host controller homing system has been added. Meanwhile, the national standard homing instructions and methods have been refined and simplified so that the system can cover all ten national standard homing methods.

18.3.3.12 Appendix A: Examples of Different Buffer Modes

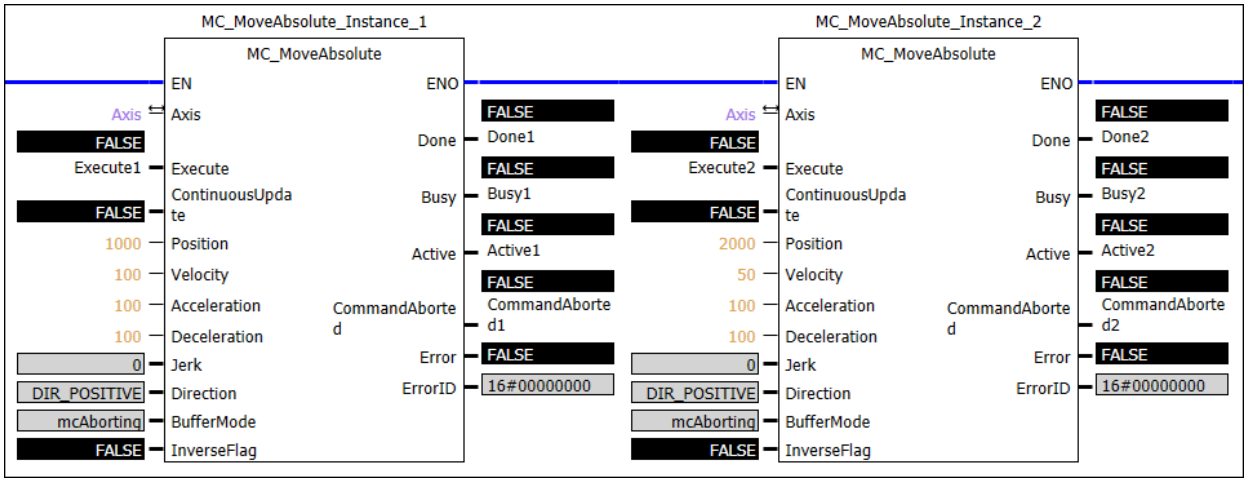
- Example 1: Standard behavior of two absolute movements
  - The figure below shows two basic examples of the MC_MoveAbsolute instructions on the same axis. Their buffer modes are set to "mcAborting".



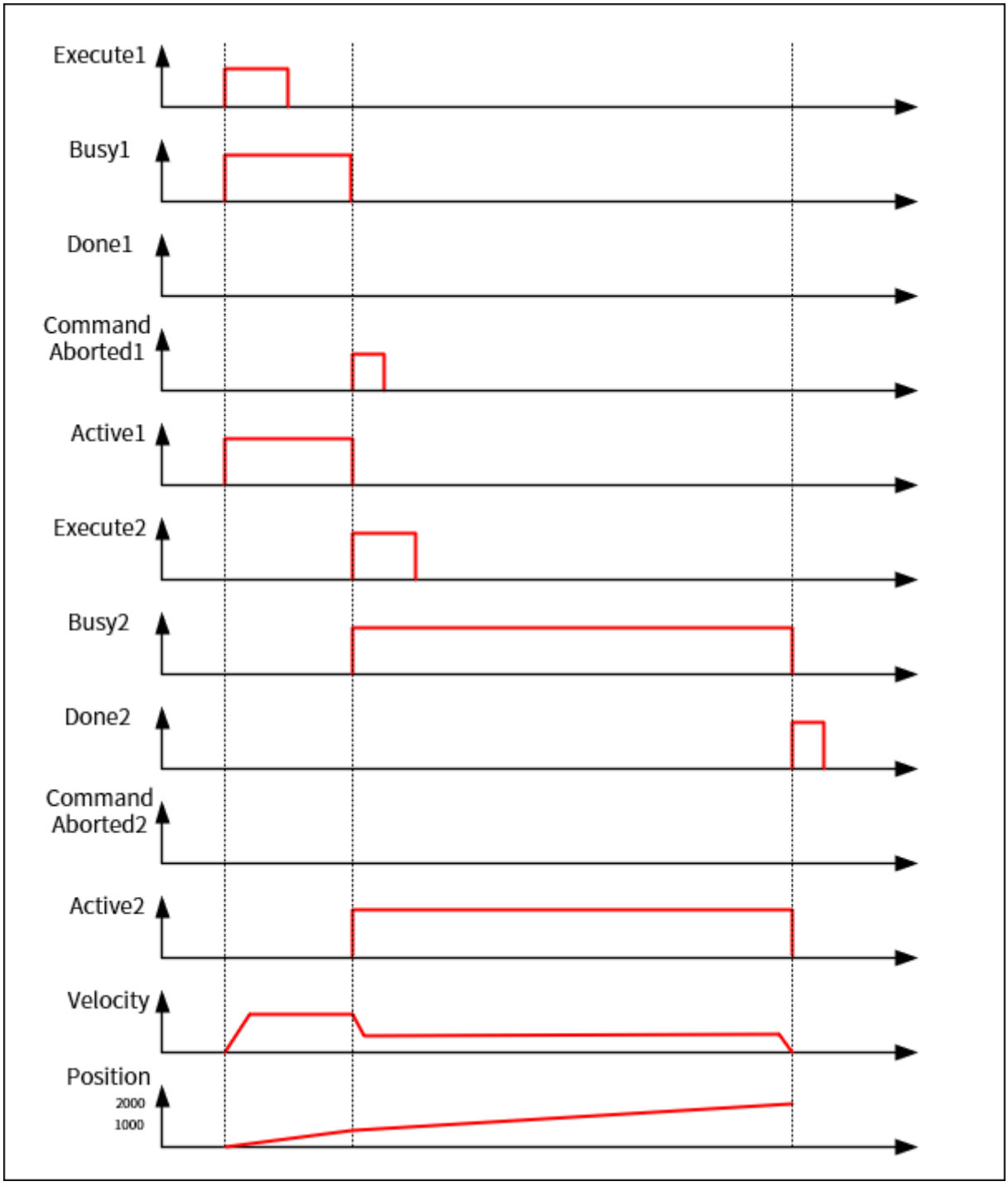
◦ The sequence diagram of the above example shows no interference between the function blocks MC_MoveAbsolute_Instance_1 and MC_MoveAbsolute_Instance_2.



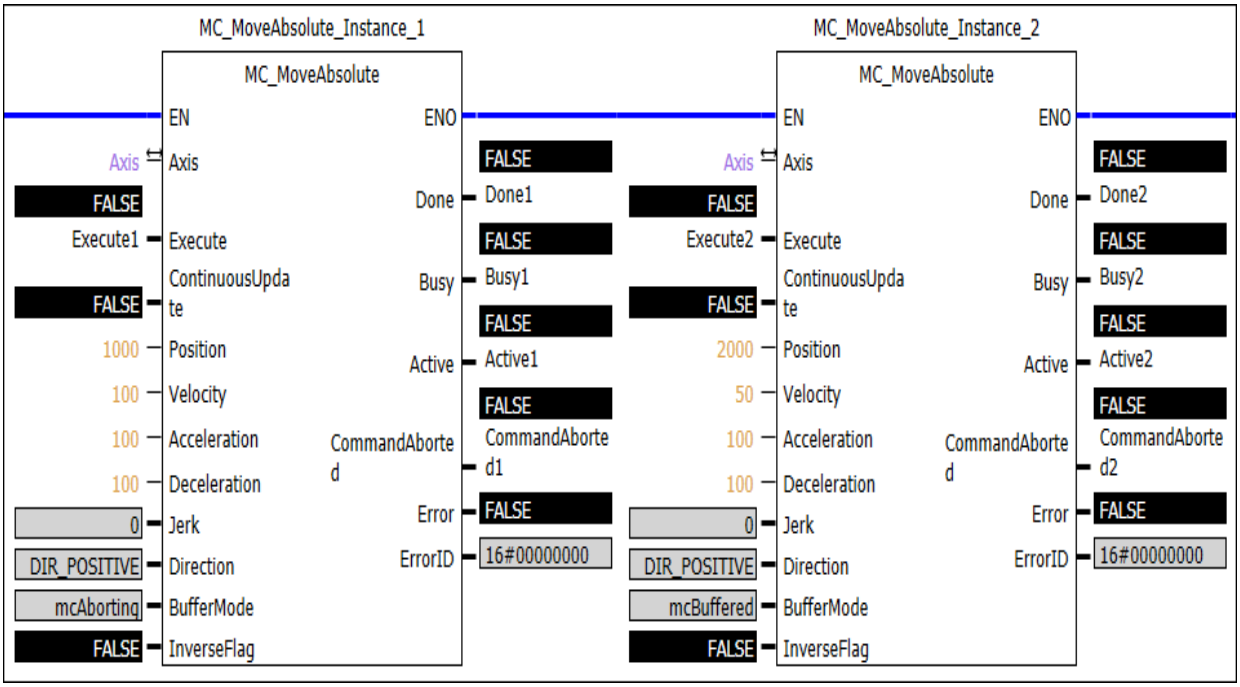
- Example 2: mcAborting
  - The figure below shows two basic examples of the MC_MoveAbsolute instructions on the same axis. Their buffer modes are set to "mcAborting".



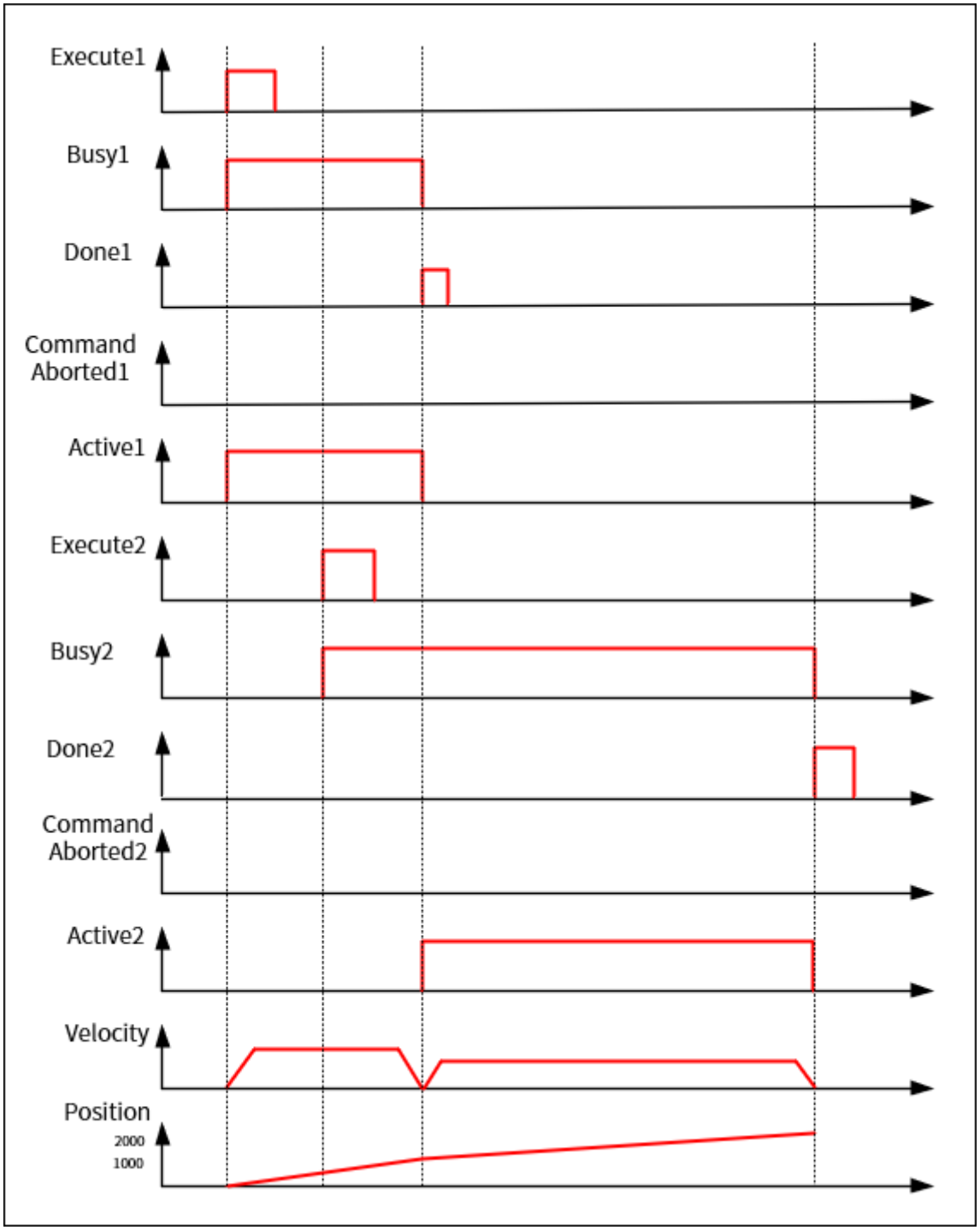
- The sequence diagram of the above example shows that the function block MC_MoveAbsolute_Instance_2 is executed before the function block MC_MoveAbsolute_Instance_1 reaches its setpoint. The currently running function block MC_MoveAbsolute_Instance_1 is moving at a speed of 100, and the function block MC_MoveAbsolute_Instance_2 aborts the function block MC_MoveAbsolute_Instance_1. The subsequently triggered aborting mode function block MC_MoveAbsolute_Instance_2 starts its motion at a speed of 100.



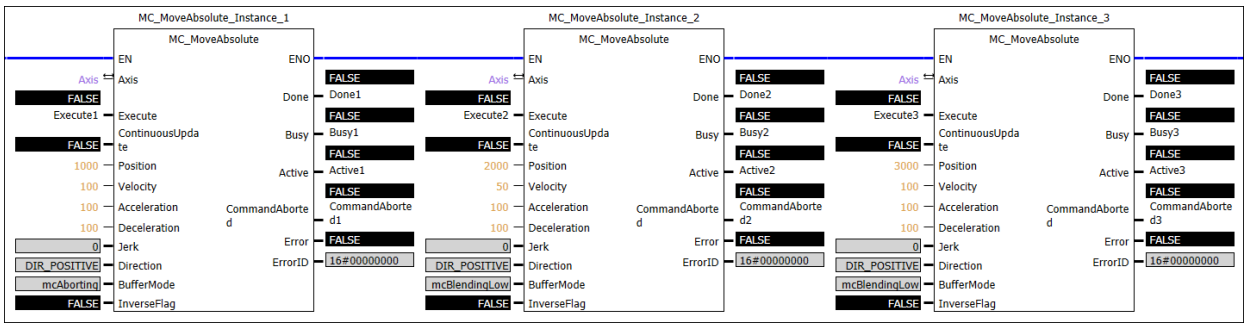
- Example 3: mcBuffered
  - The figure below shows two basic examples of the MC_MoveAbsolute instructions on the same axis. Their buffer modes are set to "mcAborting" and "mcBuffered", respectively.



- The sequence diagram of the above example shows that the function block MC_MoveAbsolute_Instance_2 is executed before the function block MC_MoveAbsolute_Instance_1 reaches its setpoint. When the currently running function block MC_MoveAbsolute_Instance_1 reaches its target position at speed 0, the axis controlled by the subsequently triggered buffer mode function block MC_MoveAbsolute_Instance_2 starts moving, accelerating from 0 to 100.

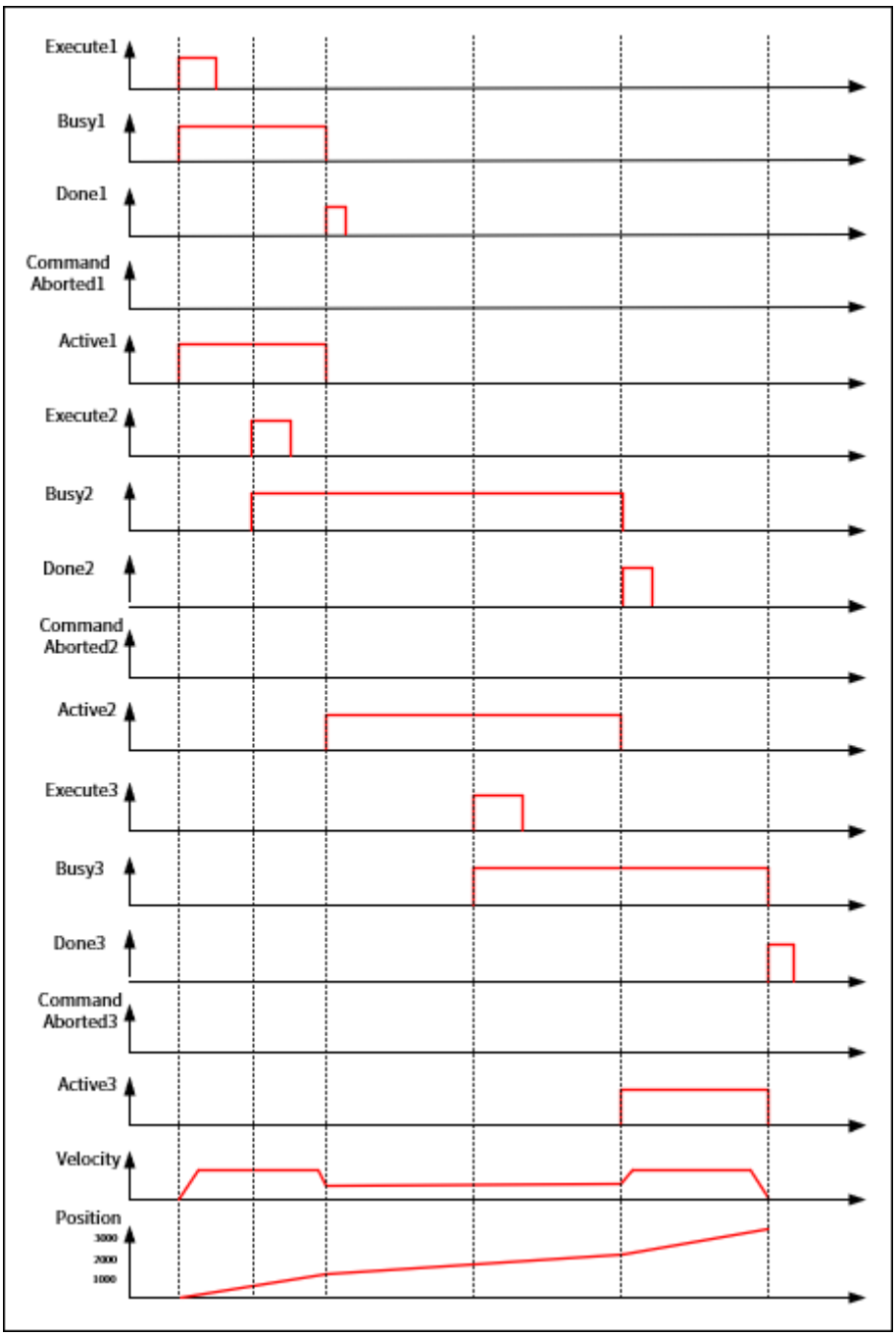


- Example 4: mcBlendingLow
  - The figure below shows three basic examples of the MC_MoveAbsolute instructions on the same axis. Their buffer modes are set to "mcAborting", "mcBlendingLow", and "mcBlendingLow" respectively.

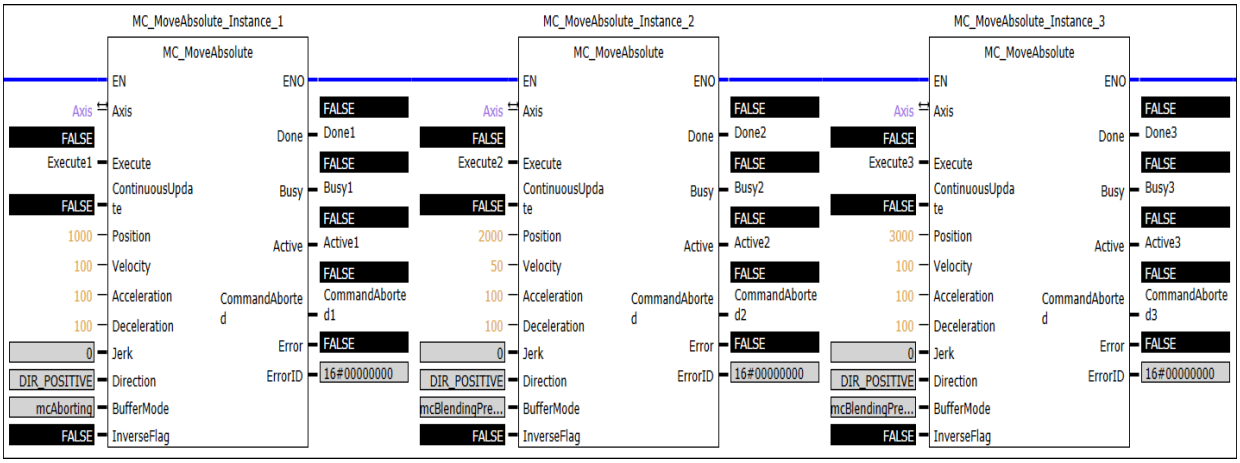


  - The sequence diagram of the above example shows that the function block MC_MoveAbsolute_Instance_2 is executed before the function block MC_MoveAbsolute_Instance_1 reaches its setpoint. Execute the function block MC_MoveAbsolute_Instance_3 before the function

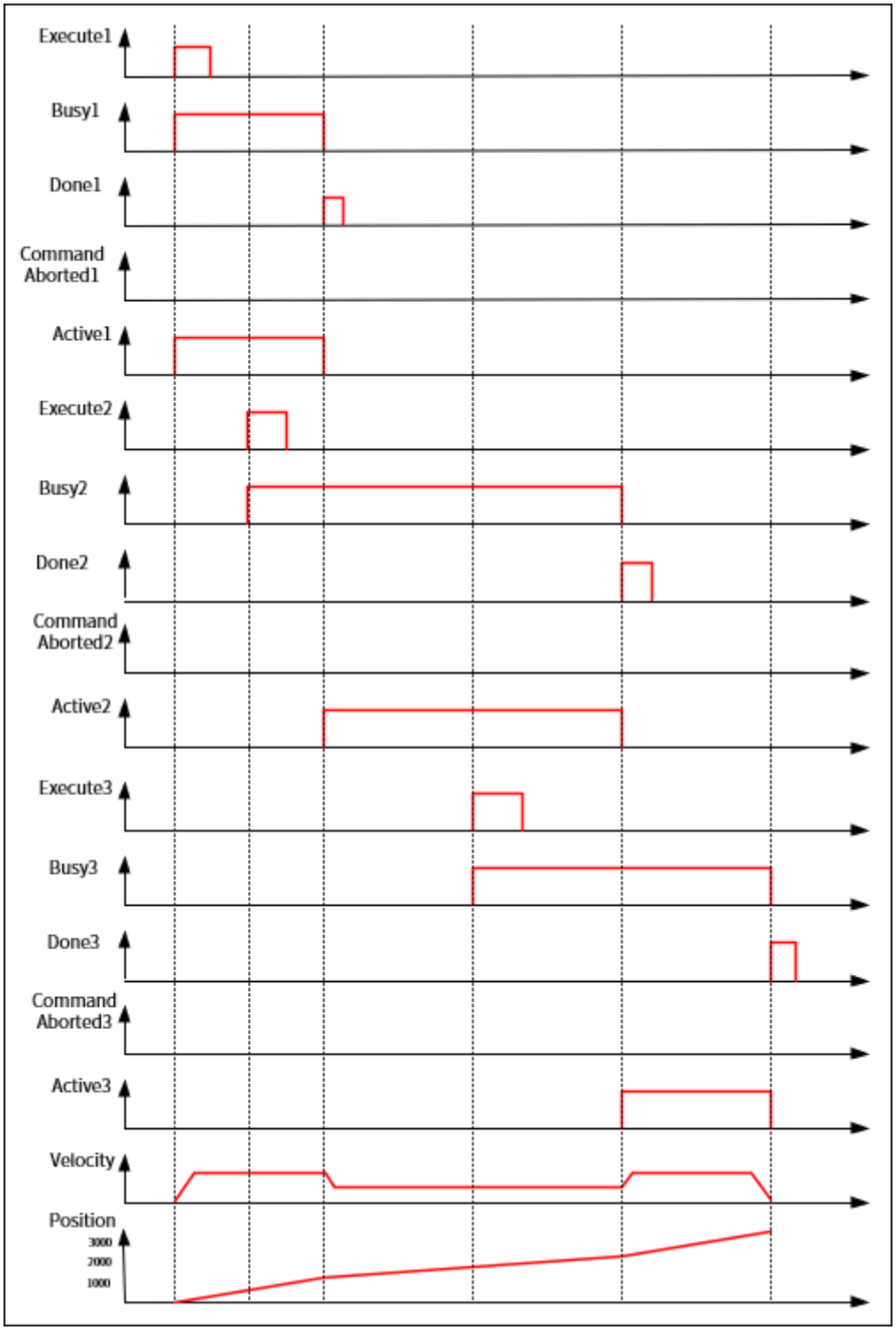
block MC_MoveAbsolute_Instance_2 reaches its setpoint. When the currently running function block MC_MoveAbsolute_Instance_1 reaches its target position at a speed of 50 (using the lower target speed between the current instruction and the buffered instruction), the axis controlled by the subsequently triggered buffer mode function block MC_MoveAbsolute_Instance_2 starts moving at a speed of 50. When the currently running function block MC_MoveAbsolute_Instance_2 reaches its target position at a speed of 50 (using the lower target speed between the current instruction and the buffered instruction), the axis controlled by the subsequently triggered buffer mode function block MC_MoveAbsolute_Instance_3 starts moving at a speed of 50.



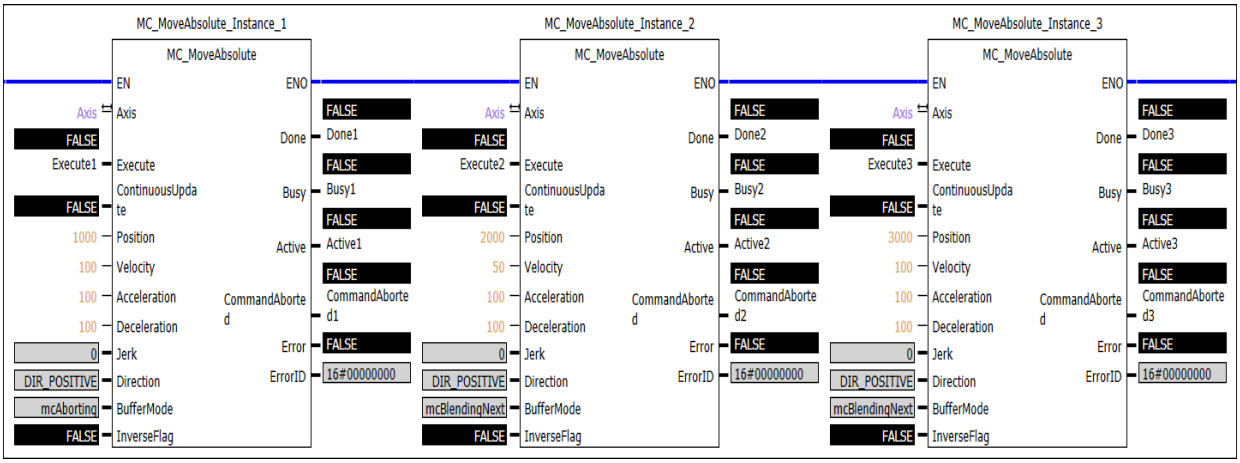
- Example 5: mcBlendingPrevious
  - The figure below shows three basic examples of the MC_MoveAbsolute instructions on the same axis. Their buffer modes are set to "mcAborting", "mcBlendingPrevious", and "mcBlendingPrevious" respectively.



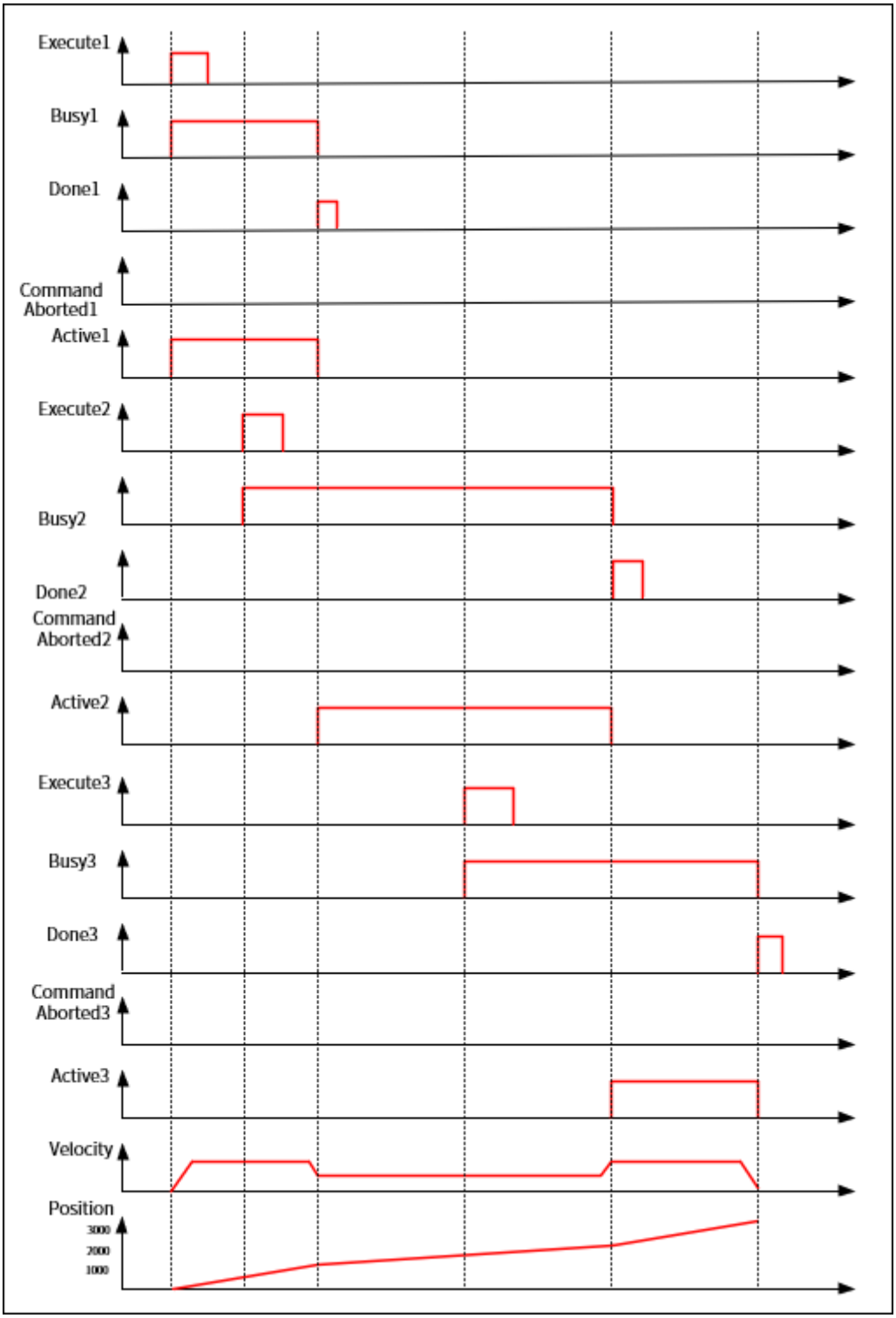
- The sequence diagram of the above example shows that the function block MC_MoveAbsolute_Instance_2 is executed before the function block MC_MoveAbsolute_Instance_1 reaches its setpoint. Execute the function block MC_MoveAbsolute_Instance_3 before the function block MC_MoveAbsolute_Instance_2 reaches its setpoint. When the currently running function block MC_MoveAbsolute_Instance_1 reaches its target position at a speed of 100 (using the target speed for the current instruction), the axis controlled by the subsequently triggered buffer mode function block MC_MoveAbsolute_Instance_2 starts moving at a speed of 100. When the currently running function block MC_MoveAbsolute_Instance_2 reaches its target position at a speed of 50 (using the target speed for the current instruction), the axis controlled by the subsequently triggered buffer mode function block MC_MoveAbsolute_Instance_3 starts moving at a speed of 50.



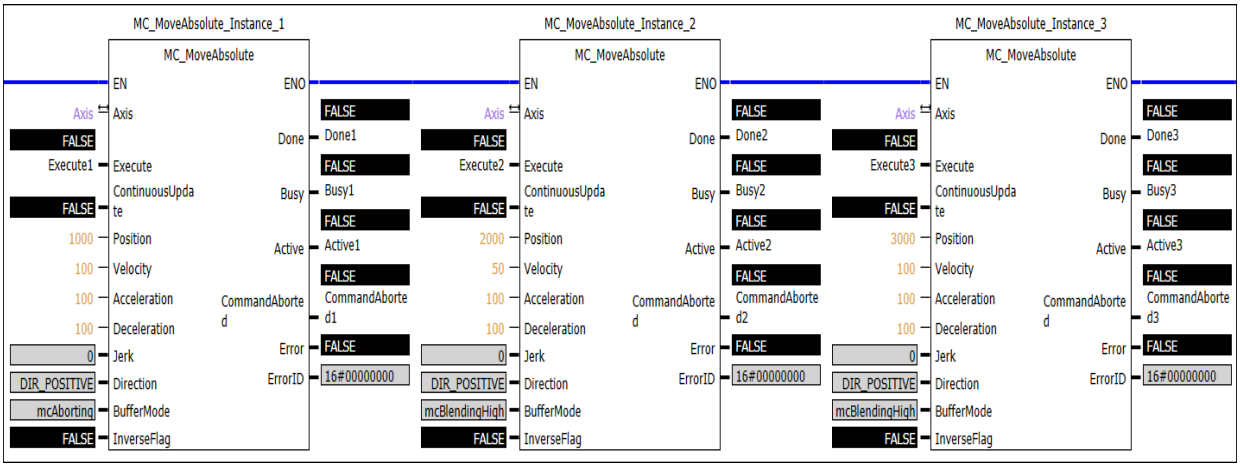
- Example 6: mcBlendingNext
  - The figure below shows three basic examples of the MC_MoveAbsolute instructions on the same axis. Their buffer modes are set to "mcAborting", "mcBlendingNext", and "mcBlendingNext" respectively.



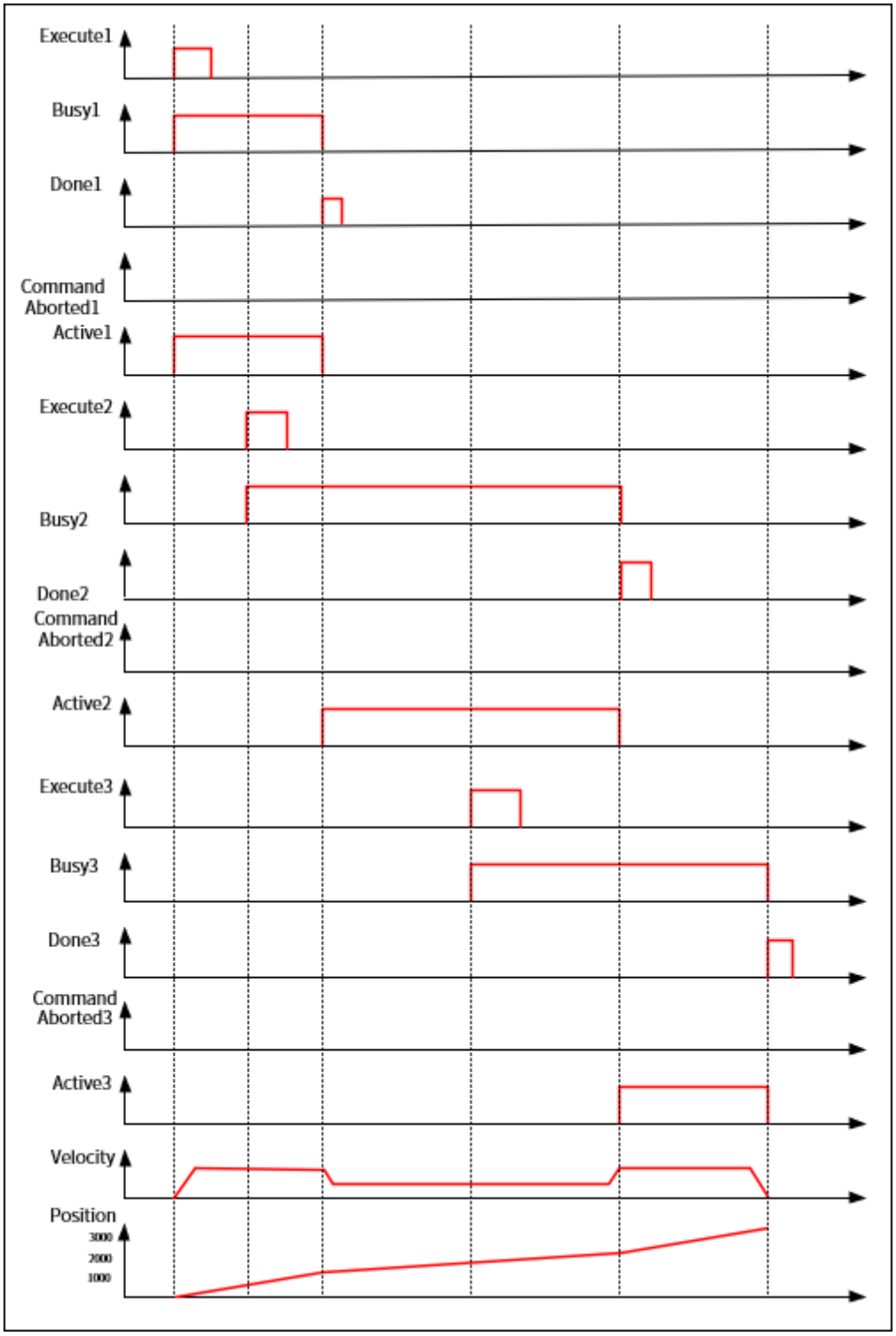
- The sequence diagram of the above example shows that the function block MC_MoveAbsolute_Instance_2 is executed before the function block MC_MoveAbsolute_Instance_1 reaches its setpoint. Execute the function block MC_MoveAbsolute_Instance_3 before the function block MC_MoveAbsolute_Instance_2 reaches its setpoint. When the currently running function block MC_MoveAbsolute_Instance_1 reaches its target position at a speed of 50 (using the target speed for the buffered instruction), the axis controlled by the subsequently triggered buffer mode function block MC_MoveAbsolute_Instance_2 starts moving at a speed of 50. When the currently running function block MC_MoveAbsolute_Instance_2 reaches its target position at a speed of 100 (using the target speed for the buffered instruction), the axis controlled by the subsequently triggered buffer mode function block MC_MoveAbsolute_Instance_3 starts moving at a speed of 100.



- Example 7: mcBlendingHigh
  - The figure below shows three basic examples of the MC_MoveAbsolute instructions on the same axis. Their buffer modes are set to "mcAborting", "mcBlendingHigh", and "mcBlendingHigh" respectively.



- The sequence diagram of the above example shows that the function block MC_MoveAbsolute_Instance_2 is executed before the function block MC_MoveAbsolute_Instance_1 reaches its setpoint. Execute the function block MC_MoveAbsolute_Instance_3 before the function block MC_MoveAbsolute_Instance_2 reaches its setpoint. When the currently running function block MC_MoveAbsolute_Instance_1 reaches its target position at a speed of 100 (using the higher target speed between the current instruction and the buffered instruction), the subsequently triggered buffer mode function block MC_MoveAbsolute_Instance_2 starts moving at a speed of 100. When the currently running MC_MoveAbsolute_Instance_2 reaches its target position at a speed of 100 (using the higher target speed between the current instruction and the buffered instruction), the subsequently triggered buffer mode function block MC_MoveAbsolute_Instance_3 starts moving at a speed of 100.



18.3.4 Motor Control

18.3.4.1 Instruction List

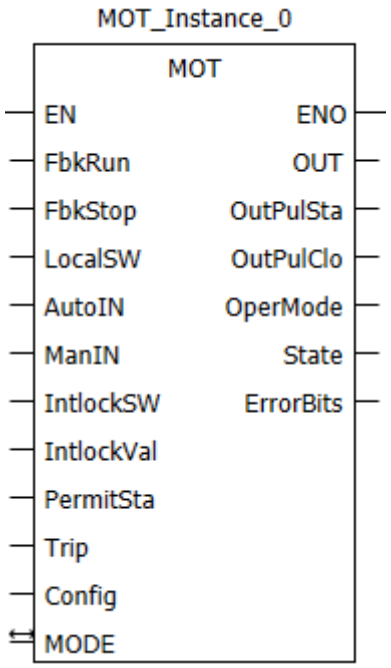
Instruction	FB/FC	Function	Library	Applicable Library Version
MOT	FB	Motor control block	ProcessControl	Library version 1.0.0.0 and later
18.3.4.3 MOTREV	FB	Bidirectional motor control block		Library version 1.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.3.4.4 SOVLv	FB	Single output control		Library version 1.0.0.0 and later
18.3.4.5 VAL	FB	Valve control block		Library version 1.0.0.0 and later

18.3.4.2 MOT

This is the instruction for motor control block.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
MOT	Motor control block	FB		MOT_Instance_0(FbkRun := , FbkStop := , LocalSW := , AutoIN := , ManIN := , IntlockSW := , IntlockVal := , PermitSta := , Trip := , Config := , MODE := , OUT => , OutPulSta => , OutPulClo => , OperMode => , State => , ErrorBits =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Data type	Initial Value	Description
Input	FbkRun	Run feedback	BOOL	FALSE	It is valid when FbkRunEN = TRUE in the configuration information.
	FbkStop	Stop feedback	BOOL	FALSE	It is valid when FbkStopEN = TRUE in the configuration information.
	LocalSW	Local switch	BOOL	FALSE	1: Local mode. 0: Remote mode.
	AutoIN	Automatic instruction	BOOL	FALSE	TRUE: Run. FALSE: Stop.
	ManIN	Manual instruction	BOOL	FALSE	TRUE: Run. FALSE: Stop.
	IntlockSW	Interlock switch	BOOL	FALSE	Interlock switch.
	IntlockVal	Interlock safety output value	BOOL	FALSE	Interlock safety output value.
	PermitSta	Whether to permit startup	BOOL	TRUE	TRUE: Yes. FALSE: No. It is valid only when a run command is issued in the stopped status.
	Trip	Motor overheat alarm	BOOL	FALSE	External alarm triggered when the value = TRUE.
	Config	Configuration information	MOT_CONFIG	-	-

<table>
<tr><th>Input/Output</th><th>Variable</th><th>Name</th><th>Data type</th><th>Initial Value</th><th>Description</th></tr>
<tr><td>Input</td><td>Parameter description</td><td colspan="4"><table>
<tr><th>Variable</th><th>Name</th><th>Data type</th><th>Initial Value</th><th>Description</th></tr>
<tr><td>FbkRunEN</td><td>Whether there is a startup feedback signal</td><td>BOOL</td><td>TRUE</td><td>1: Yes.
0: No.</td></tr>
<tr><td>FbkStopEN</td><td>Whether there is a stop feedback signal</td><td>BOOL</td><td>FALSE</td><td>1: Yes.
0: No.</td></tr>
<tr><td>OutPulseT</td><td>Pulse output time</td><td>TIME</td><td>TIME#2s0ms</td><td>Pulse output time.</td></tr>
<tr><td>TimeOutT</td><td>Timeout threshold</td><td>TIME</td><td>TIME#2s0ms</td><td>Timeout threshold.</td></tr>
<tr><td>MainTenSW</td><td>Inspection state</td><td>BOOL</td><td>FALSE</td><td>Inspection state.</td></tr>
</table></td></tr>
<tr><td>Input/Output</td><td>MODE</td><td>Mode</td><td>BYTE</td><td>0</td><td>0: Manual.
1: Automatic.</td></tr>
<tr><td rowspan="4">Output</td><td>OUT</td><td>Output</td><td>BOOL</td><td>FALSE</td><td>Output.</td></tr>
<tr><td>OutPulSta</td><td>Start pulse output</td><td>BOOL</td><td>FALSE</td><td>Start pulse output.</td></tr>
<tr><td>OutPulClo</td><td>Stop pulse output</td><td>BOOL</td><td>FALSE</td><td>Start pulse output.</td></tr>
<tr><td>OperMode</td><td>Operation mode</td><td>BYTE</td><td>0</td><td>0: Manual.
1: Automatic.
2: Interlock.
3: Local.
4: Inspection.</td></tr>
</table>

Input/Output	Variable	Name	Data type	Initial Value	Description
	State	Operation state	BYTE	0	0: Initial. 1: Started. 2: Starting. 3: Stopped. 4: Stopping. 5: Feedback error.
	ErrorBits	Error output	WORD	0	bit0: Start timeout. bit1: Stop timeout. bit2: Feedback from start to stop. bit3: Feedback from stop to start. bit4: Dual-1 feedback error. bit5: Motor overheat alarm.

- **Function**

  The motor control block controls on-site motors using single DO high/low level signals or dual DO pulse signal. It is applicable to devices with up to two status feedback signals (On/Run or Off/Stop) and is typically applied to the control of motors and similar equipment.

  - **Operation mode**

Operation Mode	Action
Automatic	Outputs signals automatically based on automatic input commands.
Manual	Outputs signals according to manual input commands.
Interlock	When the interlock switch is triggered, the interlock safety value is output. (safety value is configurable)
Local	In this mode, the device is started/stopped locally via the control box located at the local site or controlled by switch. The system tracks the actual status of the device. If no valid feedback is received from the device, the safety value is output.

Priority order: Local > Interlock > Manual > Automatic

- **Bumpless switchover**

Operation Mode	Action
Local → Automatic	In Local mode, the output interlocks with the running status of the on-site device (if there is no feedback or the feedback status is correct, the output remains unchanged). In Automatic mode, the output is the auto-calculated value directly. This is a bumpy switchover.
Local → Manual	In Local mode, the output interlocks with the running status of the on-site device (if there is no feedback or the feedback status is correct, the output remains unchanged). The manual value is interlocked with the output to ensure bumpless switchover.
Local → Interlock	In Local mode, the output interlocks with the running status of the on-site device (if there is no feedback or the feedback status is correct, the output remains unchanged). In Interlock mode, the output is the interlock safety value directly. This is a bumpy switchover.
Interlock → Automatic	In Interlock mode, the output is the interlock safety value directly. In Automatic mode, the output is the auto-calculated value directly. This is a bumpy switchover.
Interlock → Manual	In Interlock mode, the manual value is interlocked with the interlock safety value to ensure bumpless switchover.
Interlock → Local	In Interlock mode, the output is the interlock safety value. At the moment of switching to Local mode without any operation, the on-site device must also be in the interlock safety value state. After that, when the device is operated on-site, the output interlocks with the running status of the on-site device (if there is no feedback or the feedback status is correct, the output remains unchanged). This is a bumpless switchover.
Manual → Automatic	The output switches directly from the manual output value to the auto-calculated value. This is a bumpy switchover.
Manual → Interlock	The output switches directly from the manual output value to the interlock safety value. This is a bumpy switchover.
Manual → Local	In Manual mode, the output is the current manual value. At the moment of switching to Local mode without any operation, the on-site device must also be in the manual value state. After that, when the device is operated on-site, the output interlocks with the running status of the on-site device (if there is no feedback or the feedback status is correct, the output remains unchanged). This is a bumpless switchover.
Automatic → Manual	In Automatic mode, the output is the auto-calculated value. At this time, the manual value is interlocked with the auto-calculated value to ensure bumpless switchover.
Automatic → Interlock	The output switches directly from the auto-calculated value to the interlock safety value. This is a bumpy switchover.

Operation Mode	Action
Automatic → Local	In Automatic mode, the output is the auto-calculated value. At the moment of switching to Local mode without any operation, the on-site device must also be in the state of automatic calculation. After that, when the device is operated on-site, the output interlocks with the running status of the on-site device (if there is no feedback or the feedback status is correct, the output remains unchanged). This is a bumpless switchover.

- **Timeout alarm**

  Timeout alarm triggers if the function block status feedback disagrees with its output beyond the configured timeout period. Prerequisite: The device has running status feedback.

  For example, the function block issues an "open" instruction to the on-site motor. If the device responds normally and starts within the set time, it returns the startup status to the function block – this is the normal startup process. If no correct feedback is received within the set time, the device is considered not properly opened, triggering a timeout alarm. The timeout detection period is configured by the parameter "Config. TimeOutT", which can be configured based on the actual device response.

  Note: Timeout alarm clears automatically when valid feedback is received or command is switched with no manual acknowledgment required.

- **Inspection (Stop Running)**

  After an inspection command is issued, the interlock safety value ILIN is directly output. No automatic, manual, or interlock commands are accepted, ensuring the safety of on-site personnel. In addition, on-site fault signals from will not generate any alarm messages.

- **Start permit**

  When the equipment is in the stopped status and ready to start, the motor can only be started if the st art conditions are met.

  If the start conditions are met and there is a start command, but the feedback already indicates a running status, a restart is not allowed, and the output remains unchanged.

- **Output function**

  One DO level signal can be output to the site to control the opening and closing of motors or valves; alternatively, two pulse-type DO signals can also be output to separately control the motors or valves.

  When Open Signal = TRUE and Close Signal = FALSE, the motor is started; when Open Signal = FALSE and Close Signal = TRUE, the motor is stopped. The pulse width is adjustable via Config.OutPulseT. The pulse stops once the set time elapses.

- **Deviation alarm (feedback fluctuation)**

  If the running signal is lost during motor operation, an alarm is triggered.

  For example: If the device is started but the feedback changes to stop, a "feedback open-to-close" alarm is triggered with state set to "feedback error". If the device is stopped but the feedback changes to start, a "feedback close-to-open" alarm is triggered with state set to "feedback error". Deviation alarms can clear automatically when the state returns to normal.

- **Motor trip fault**

It is used for motor overheat protection to stop the motor (Trip = 1). When a fault occurs, it outputs the interlock safety value and switches to manual mode.

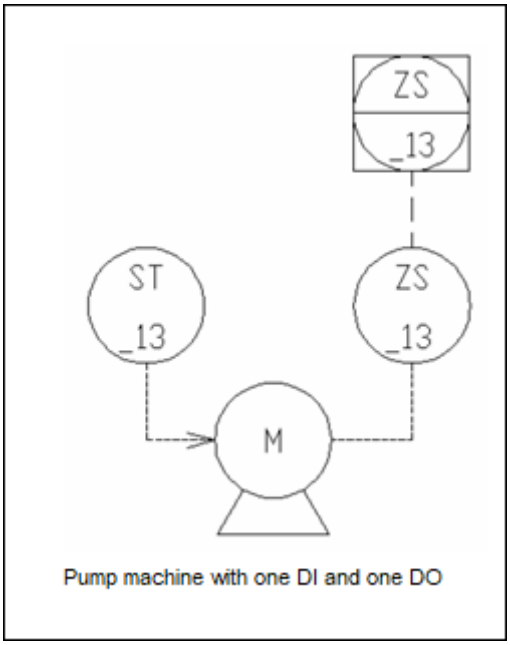
Example of working conditions:

Working Condition	OUT	OperMode (Operation Mode)	State (Operation State)	ErrorBits (Error Output) Bitwise Analysis
Initial status		Determined by the current specific mode: 0: Manual 1: Automatic 2: Interlock 3: Local 4: Inspection	0: Initial status	0
Start command issued, with start feedback received within the timeout period	Output start command Output the start pulse of the set time		1: Started	0
Start command issued, with no start feedback received within the timeout period	Output start command Output the start pulse of the set time		2: Starting	bit0: Start timeout
Stop command issued, with stop feedback received within the timeout period	Output stop command Output the stop pulse of the set time		3: Stopped	0
Stop command issued, with no stop feedback received within the timeout period	Output stop command Output the stop pulse of the set time		4: Stopping	Bit1: Stop timeout
In the started status, feedback changes from start to stop.	/		5: Feedback error	Bit2: Feedback from start to stop.
In the stopped status, feedback changes from stop to start.	/			Bit3: Feedback from stop to start
Trip=1 Motor overheat fault	Output safety value	0: Manual	Set based on the actual feedback	Bit5: Motor overheat alarm
Start feedback and stop feedback are both 1	-	-	5: Feedback error	Bit4: Dual 1 feedback error

Working Condition	OUT	OperMode (Operation Mode)	State (Operation State)	ErrorBits (Error Output) Bitwise Analysis
After the command is issued, start feedback and stop feedback are both 0	-	-	5: Feedback error	

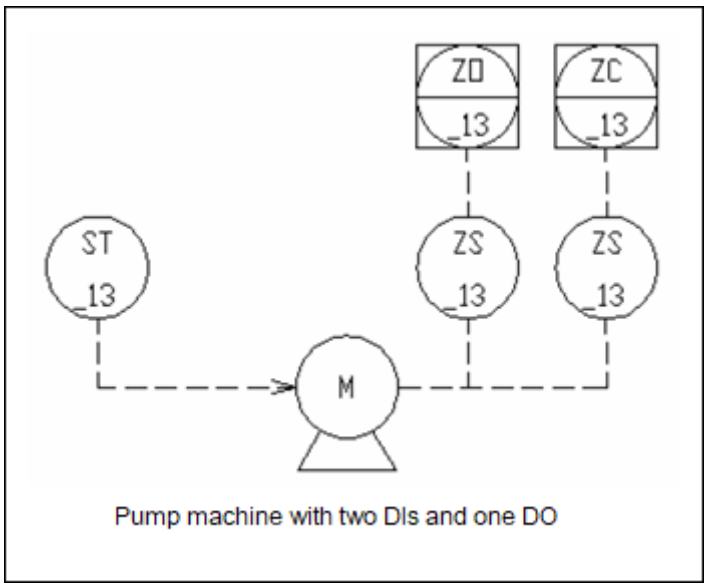
- **Application instance**
  - Single DI single DO pump

    There is a fan drive on-site requires start/stop control of this motor drive. A pump operation fault alarm must be triggered without confirmation if a mismatch occurs between the pump status feedback and the control output after the stroke time has elapsed.



  - Dual DI single DO pump

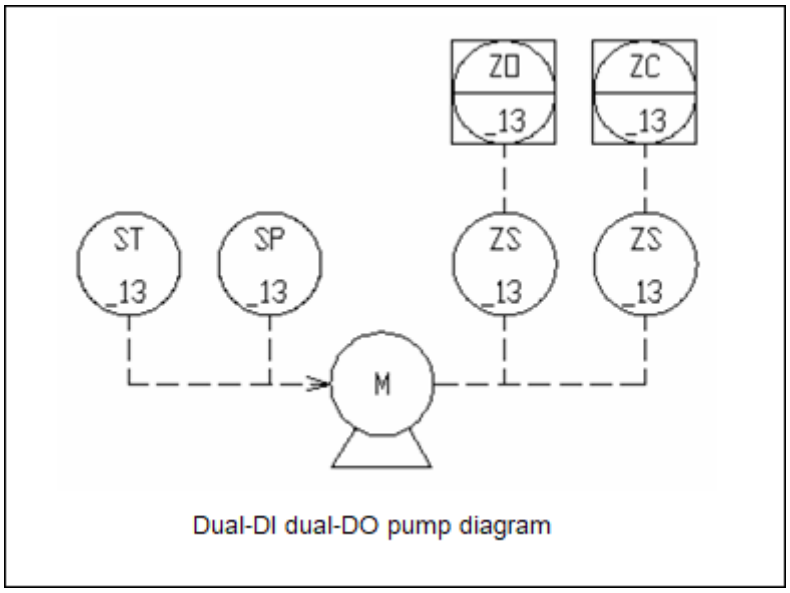
    A motor on-site requires start/stop control of the pump. A pump operation fault alarm must be triggered without confirmation if a mismatch occurs between the pump status feedback and the control output after the stroke time has elapsed.



Pump machine with two DIs and one DO

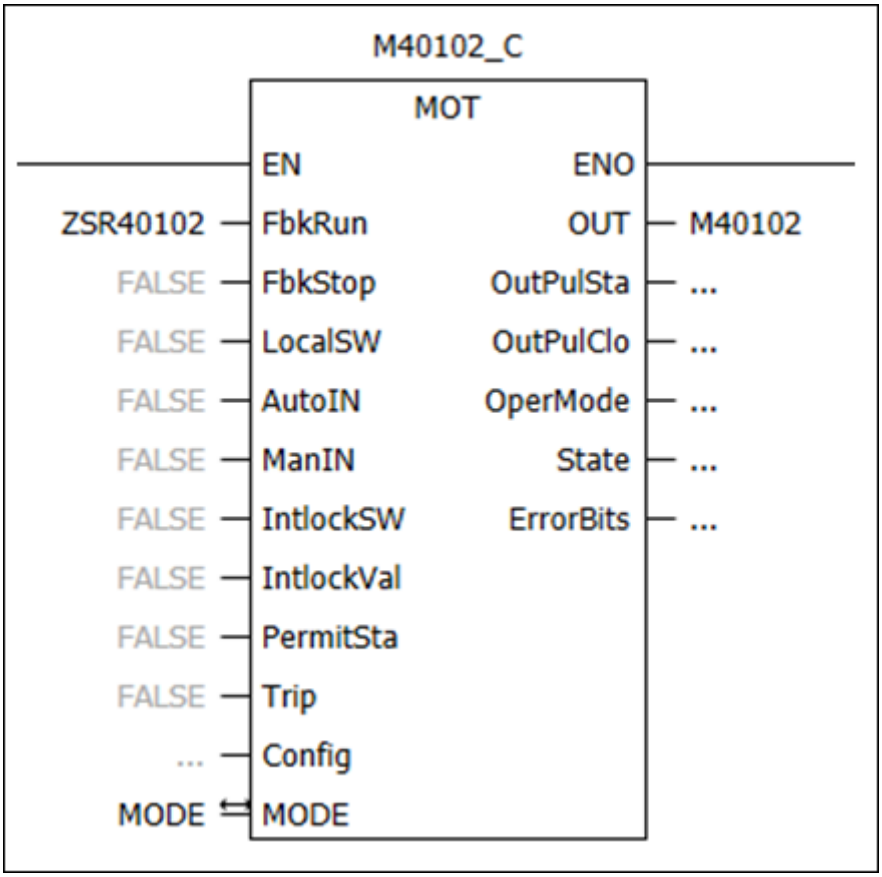
◦ Dual DI dual DO pump

A high-power motor one-site requires start/stop control of the pump. A pump operation fault alarm must be triggered without confirmation if a mismatch occurs between the pump status feedback and the control output after the stroke time has elapsed.



Dual-DI dual-DO pump diagram

Control scheme configuration



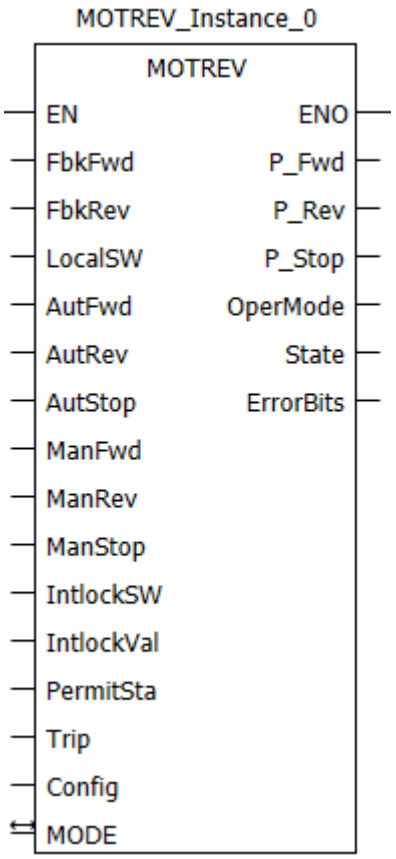
Example

No.	Instance	Type	Remarks
1	M40102_C	Motor control function block	-
2	M40102	DO output	-
3	ZSR40102	DI feedback input	In the configuration information, FbkRunEN = TRUE is valid when there is start feedback information.

18.3.4.3 MOTREV

This is the instruction for bidirectional motor control block.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
MOTREV	Bidirectional motor control block	FB		MOTREV_Instance_0(FbkFwd := , FbkRev := , LocalSW := , AutFwd := , AutRev := , AutStop := , ManFwd := , ManRev := , ManStop := , IntlockSW := , IntlockVal := , PermitSta := , Trip := , Config := , MODE := , P_Fwd => , P_Rev => , P_Stop => , OperMode => , State => , ErrorBits =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Data type	Initial Value	Description
Input	FbkFwd	Forward feedback	BOOL	-	Forward feedback.
	FbkRev	Reverse feedback	BOOL	-	Reverse feedback.
	LocalSW	Local switch	BOOL	-	1: Local mode. 0: Remote mode.

Input/Output	Variable	Name	Data type	Initial Value	Description
	AutFwd	Automatic command: Forward start	BOOL	-	Automatic command: Forward start.
	AutRev	Automatic command: Reverse start	BOOL	-	Automatic command: Reverse start.
	AutStop	Automatic command: Stop start	BOOL	-	Automatic command: Stop Start.
	ManFwd	Manual command: Forward start	BOOL	-	Manual command: Forward start.
	ManRev	Manual command: Reverse start	BOOL	-	Manual command: Reverse start.
	ManStop	Manual command: Stop start	BOOL	-	Manual command: Stop start.
	IntlockSW	Interlock switch	BOOL	-	Interlock switch.
	IntlockVal	Interlock safety output value	BYTE	0	0: Stop. 1: Forward. 2: Reverse.
	PermitSta	Whether to permit startup	BOOL	TRUE	TRUE: Yes. FALSE: NO. The start command is only valid when the motor is in the stopped state.
	Trip	Motor overheat alarm	BOOL	FALSE	External alarm triggered when the value = TRUE.
	Config	Configuration information	MOTREV_CONFIG	-	-

<table>
<tr><th>Input/Output</th><th>Variable</th><th>Name</th><th>Data type</th><th>Initial Value</th><th>Description</th></tr>
<tr><td></td><td>Parameter description</td><td colspan="4"><table>
<tr><th>Variable</th><th>Name</th><th>Data type</th><th>Initial Value</th><th>Description</th></tr>
<tr><td>MainTenSW</td><td>Inspection state</td><td>BOOL</td><td>FALSE</td><td>Inspection state.</td></tr>
<tr><td>OutPulseT</td><td>Pulse output time</td><td>TIME</td><td>TIME#2s0ms</td><td>Pulse output time.</td></tr>
<tr><td>TimeOutT</td><td>Timeout threshold</td><td>TIME</td><td>TIME#2s0ms</td><td>Timeout threshold.</td></tr>
</table></td></tr>
<tr><td>Input/Output</td><td>MODE</td><td>Mode</td><td>BYTE</td><td>-</td><td>0: Manual.
1: Automatic.</td></tr>
<tr><td rowspan="5">Output</td><td>P_Fwd</td><td>Forward start pulse output</td><td>BOOL</td><td>FALSE</td><td>Forward start pulse output.</td></tr>
<tr><td>P_Rev</td><td>Reverse start pulse output</td><td>BOOL</td><td>FALSE</td><td>Reverse start pulse output.</td></tr>
<tr><td>P_Stop</td><td>Stop pulse output</td><td>BOOL</td><td>FALSE</td><td>Stop pulse output.</td></tr>
<tr><td>OperMode</td><td>Operation mode</td><td>BYTE</td><td>0</td><td>0: Manual.
1: Automatic.
2: Interlock.
3: Local.
4: Inspection.</td></tr>
<tr><td>State</td><td>Operation state</td><td>BYTE</td><td>0</td><td>0: Initial.
1: Forwarded.
2: Forwarding.
3: Reversed.
4: Reversing.
5: Stopped.
6: Stopping.
7: Feedback error.</td></tr>
</table>

Input/Output	Variable	Name	Data type	Initial Value	Description
	ErrorBits	Error output	WORD	0	bit0: Forward timeout. bit1: Reverse running timeout. bit2: Stop timeout. bit3: Feedback from forward to reverse. bit4: Feedback from forward to stop. bit5: Feedback from reverse to forward. bit6: Feedback from reverse to stop. bit7: Feedback from stop to forward. bit8: Feedback from stop to reverse. bit9: Motor overheat alarm. bit10: Input instruction error. bit11: Dual-1 feedback error.

- **Function**

  The bidirectional motor control block refers to the on-site motor equipment being controlled by three switch outputs (pulse signals), that is: three DO pulse signals control the motor's forward, reverse, and stop. Forward and reverse cannot be directly switched; they must go through the stop status. Applicable to devices with at most two status feedback signals (forward, reverse), typically used in motor and motor control.

  - **Operation mode**

Operation Mode	Action
Automatic	Output based on automatic input command.
Manual	Output based on manual input command.
Interlock	When the interlock switch is triggered, the interlock safety value (configurable) is output directly.
Local	In this mode, the device is started/stopped or switched on/off locally via the control box located at the local site. With correct feedback, the system tracks the actual status of the device.

Priority order: Local > Interlock > Manual > Automatic

- **Bumpless switchover**

Operation Mode	Action
Local → Automatic	In Local mode, the output interlocks with the running status of the on-site device (if there is no feedback or the feedback status is correct, the output remains unchanged). In Automatic mode, the output is the auto-calculated value directly. This is a bumpy switchover.
Local → Manual	In Local mode, the output interlocks with the running status of the on-site device (if there is no feedback or the feedback status is correct, the output remains unchanged). The manual value is interlocked with the output to ensure bumpless switchover.
Local → Interlock	In Local mode, the output interlocks with the running status of the on-site device (if there is no feedback or the feedback status is correct, the output remains unchanged). In Interlock mode, the output is the interlock safety value directly. This is a bumpy switchover.
Interlock → Automatic	In Interlock mode, the output is the interlock safety value directly. In Automatic mode, the output is the auto-calculated value directly. This is a bumpy switchover.
Interlock → Manual	In Interlock mode, the manual value is interlocked with the interlock safety value to ensure bumpless switchover.
Interlock → Local	In Interlock mode, the output is the interlock safety value. At the moment of switching to Local mode without any operation, the on-site device must also be in the interlock safety value state. After that, when the device is operated on-site, the output interlocks with the running status of the on-site device (if there is no feedback or the feedback status is correct, the output remains unchanged). This is a bumpless switchover.
Manual → Automatic	The output switches directly from the manual output value to the auto-calculated value. This is a bumpy switchover.
Manual → Interlock	The output switches directly from the manual output value to the interlock safety value. This is a bumpy switchover.

Operation Mode	Action
Manual → Local	In Manual mode, the output is the current manual value. At the moment of switching to Local mode without any operation, the on-site device must also be in the manual value state. After that, when the device is operated on-site, the output interlocks with the running status of the on-site device (if there is no feedback or the feedback status is correct, the output remains unchanged). This is a bumpless switchover.
Automatic → Manual	In Automatic mode, the output is the auto-calculated value. At this time, the manual value is interlocked with the auto-calculated value to ensure bumpless switchover.
Automatic → Interlock	The output switches directly from the auto-calculated value to the interlock safety value. This is a bumpy switchover.
Automatic → Local	In Automatic mode, the output is the auto-calculated value. At the moment of switching to Local mode without any operation, the on-site device must also be in the state of automatic calculation. After that, when the device is operated on-site, the output interlocks with the running status of the on-site device (if there is no feedback or the feedback status is correct, the output remains unchanged). This is a bumpless switchover.

- **Timeout alarm**

  A timeout alarm is triggered when the status feedback of the function block does not match its output within the set timeout period.

  For example, the function block issues a "forward" instruction to the on-site motor. If the device responds normally and forward within the set time, it returns the forward status to the function block - this is the normal forward process. If no correct feedback is received within the set time, the device is considered not properly forwarded, triggering a timeout alarm. The timeout detection period is configured by the parameter "Config. TimeOutT", which can be configured based on the actual device response.

  Note: Timeout alarm clears automatically when valid feedback is received or command is switched with no manual acknowledgment required.

- **Inspection (Stop Running)**

  After an inspection command is issued, the interlock safety value ILIN is directly output. No automatic, manual, or interlock commands are accepted, ensuring the safety of on-site personnel. In addition, on-site fault signals from will not generate any alarm messages.

- **Start permit**

  When the device is in the stop position and ready for forward running or reverse running, the motor can only start if the start conditions are met.

- **Output function**

  Three pulse-type DO signals are output to the site to control the on-site motors.

P_Fwd (Forward)	P_Rev (Reverse)	P_Stop (Stop)	Output Command
1	0	0	Forward start pulse
0	1	0	Reverse start pulse

P_Fwd (Forward)	P_Rev (Reverse)	P_Stop (Stop)	Output Command
0	0	1	Stop pulse

The pulse width is adjustable via Config.OutPulseT. The pulse stops once the set time elapses.

- **Deviation alarm (feedback fluctuation)**

  If the running signal is lost during motor operation, an alarm is triggered.

  For example: If the device is in forward running but the feedback changes to stop, a "feedback forward-to-stop" alarm is triggered and the state is set to "feedback error". If the device is in reverse running but the feedback changes to forward, a "feedback reverse-to-forward" alarm is triggered and the state is set to "feedback error". Deviation alarms can clear automatically when the state returns to normal.

- **Motor trip fault**

  It is used for motor overheat protection to stop the motor (Trip=0). When a fault occurs, it outputs the interlock safety value and switches to manual mode.

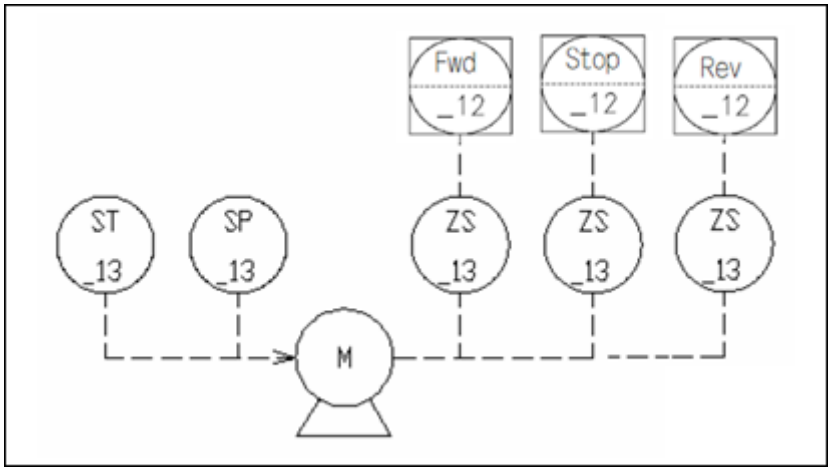
  Example of working conditions:

Working Condition	OUT	OperMode (Operation Mode)	State (Operation State)	ErrorBits (Error Output) Bitwise Analysis
Initial status	-	Determined by the current specific mode: 0: Manual 1: Automatic 2: Interlock 3: Local 4: Inspection	0: Initial status	0
Start (forward or reverse) command issued, with start (forward or reverse) feedback received within the timeout period	Output start (forward or reverse) command Output the pulse of the set time		1: Forwarded 3: Reversed	0
Start (forward or reverse) command issued, with no start (forward or reverse) feedback received within the timeout period	Output start (forward or reverse) command Output the pulse of the set time		2: Forwarding 4: Reversing	Bit0: Forward timeout Bit1: Reverse timeout
Stop command issued, with stop feedback received within the timeout period	Output stop command Output the stop pulse of the set time		5: Stopped	0
Stop command issued, with no stop feedback received within the timeout period	Output stop command Output the stop pulse of the set time		6: Stopping	Bit2: Stop timeout

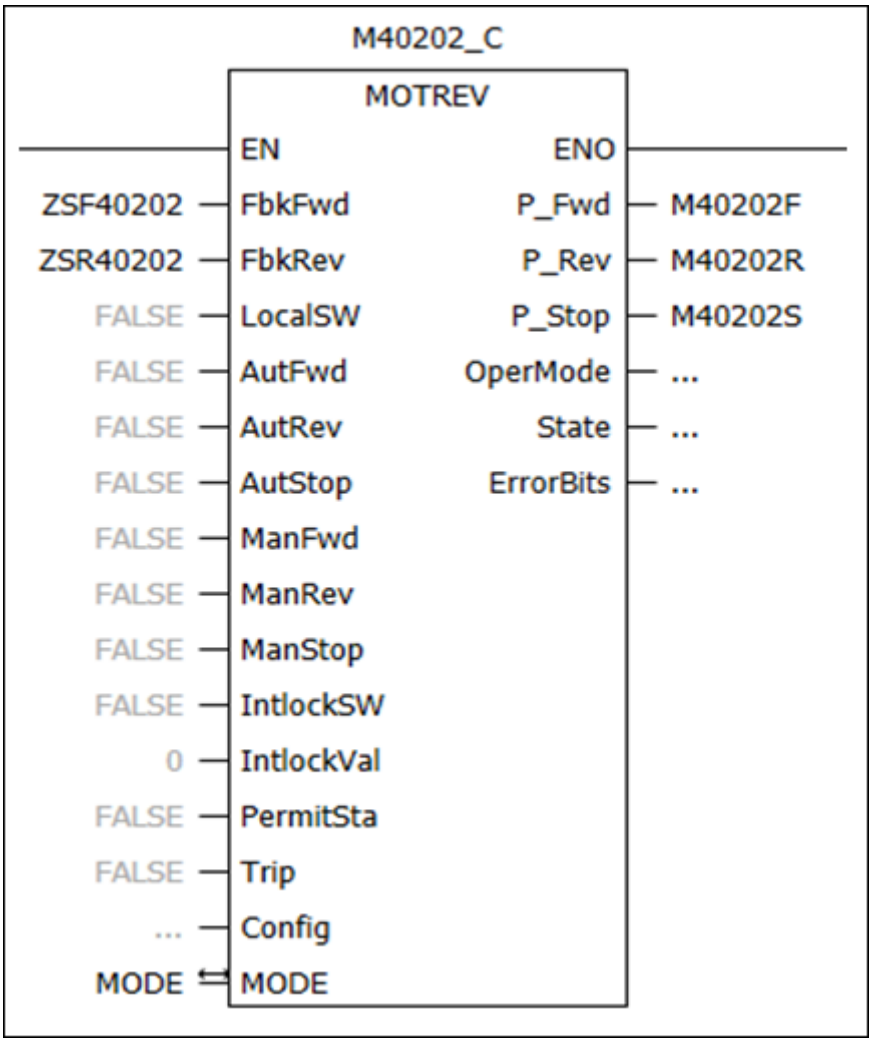
Working Condition	OUT	OperMode (Operation Mode)	State (Operation State)	ErrorBits (Error Output) Bitwise Analysis
In the started state (forwarded or reversed), feedback changes from start to stop	/		7: Feedback error	bit3: Feedback from forward to reverse bit4: Feedback from forward to stop bit5: Feedback from reverse to forward bit6: Feedback from reverse to stop
In the stopped state (forward or reverse), feedback changes from stop to start	/			bit7: Feedback from stop to forward bit8: Feedback from stop to reverse
Trip=1 Motor overheat fault	Output safety value	0: Manual	Set based on the actual feedback	Bit9: Motor overheat alarm
Forward feedback and reverse feedback are both 1	-	Determined by the current specific mode	5: Feedback error	Bit11: Dual 1 feedback error
There are multiple input commands, such as forward command and stop command	Remain		-	bit10: Input command error

- **Application instance**
  - **Dual DI triple DO pump machine**

    A bidirectional motor at a site requires the implementation of forward, reverse, and stop control for the motor.



Control scheme configuration



Example

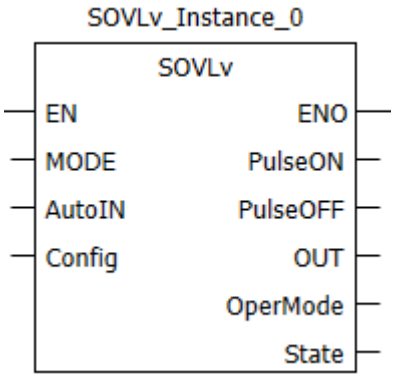
No.	Instance	Type
1	M40202_C	Bidirectional motor control function block
2	M40202F	DO output
3	M40202R	DO output
4	M40202S	DO output

No.	Instance	Type
5	ZSF40202	DI input
6	ZSR40202	DI input

18.3.4.4 SOVLv

This is the single output control instruction.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SOVLv	Single output control	FB		SOVLv_Instance_0(MODE := , AutoIN := , Config := , PulseON => , PulseOFF => , OUT => , OperMode => , State =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Data type	Initial Value	Description
Input	MODE	Operation mode selection	BOOL	-	FALSE: Manual. TRUE: Automatic.
	AutoIN	Automatic instruction	BOOL	FALSE	Automatic command: 1: Start. 0: Stop.
	Config	Configuration information	SOLVV_CONFIG	-	-

<table>
<tr><th>Input/Output</th><th>Variable</th><th>Name</th><th>Data type</th><th>Initial Value</th><th>Description</th></tr>
<tr><td></td><td colspan="5"><table>
<tr><th>Variable</th><th>Name</th><th>Data type</th><th>Initial Value</th><th>Description</th></tr>
<tr><td>ManIN</td><td>Manual command</td><td>BOOL</td><td>FALSE</td><td>Manual command:
1: Start.
0: Stop.</td></tr>
<tr><td>PlusWt</td><td>Output pulse width</td><td>TIME</td><td>TIME#2s0ms</td><td>Output pulse width.</td></tr>
</table></td></tr>
<tr><td rowspan="5">Output</td><td>PulseON</td><td>Output on pulse</td><td>BOOL</td><td>-</td><td>Output on pulse.</td></tr>
<tr><td>PulseOFF</td><td>Output off pulse</td><td>BOOL</td><td>-</td><td>Output off pulse.</td></tr>
<tr><td>OUT</td><td>Output level instruction</td><td>BOOL</td><td>-</td><td>Output level instruction.</td></tr>
<tr><td>OperMode</td><td>Operation mode</td><td>BYTE</td><td>-</td><td>0: Manual.
1: Automatic.</td></tr>
<tr><td>State</td><td>Output state</td><td>BYTE</td><td>-</td><td>1: Start.
2: Stop.</td></tr>
</table>

- **Function**

  It is typically used for the simple control of on-site valve device without feedback signals, including one input command, two pulse output commands, and one level output command.

  - **Operation mode**

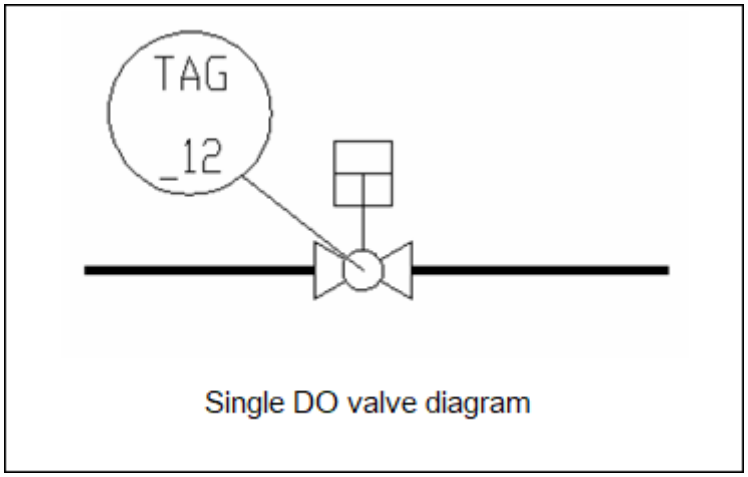
    The "MODE" supports both manual and automatic modes. In the automatic mode ("MODE = TRUE"), the output directly responds to the input, and the outpu pulse width of command can be configured via the "PlusWt" parameter.

  - **Bumpless switchover**

    Automatic-to-manual switching is bumpless, while manual-to-automatic switching responds directly to the automatic command, which is a bumpy switchover.

- **Application instance**
  - Single DO valve control

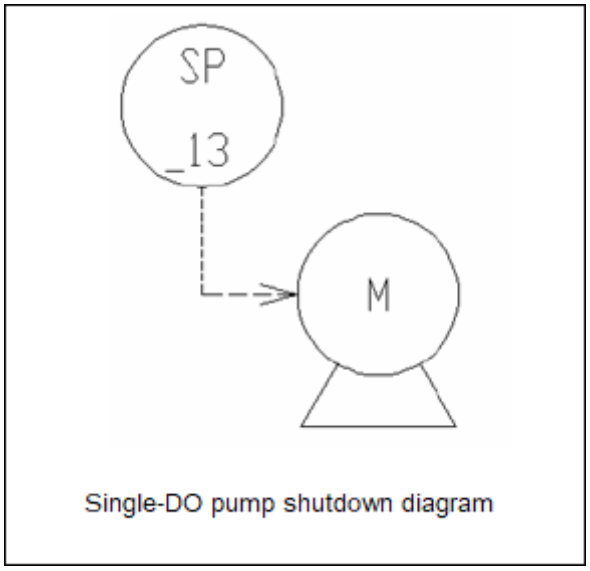
    There is a pneumatic valve switch control on a certain device, with no feedback signal from the valve, requiring the implementation of the valve's on-off control.



Single DO valve diagram

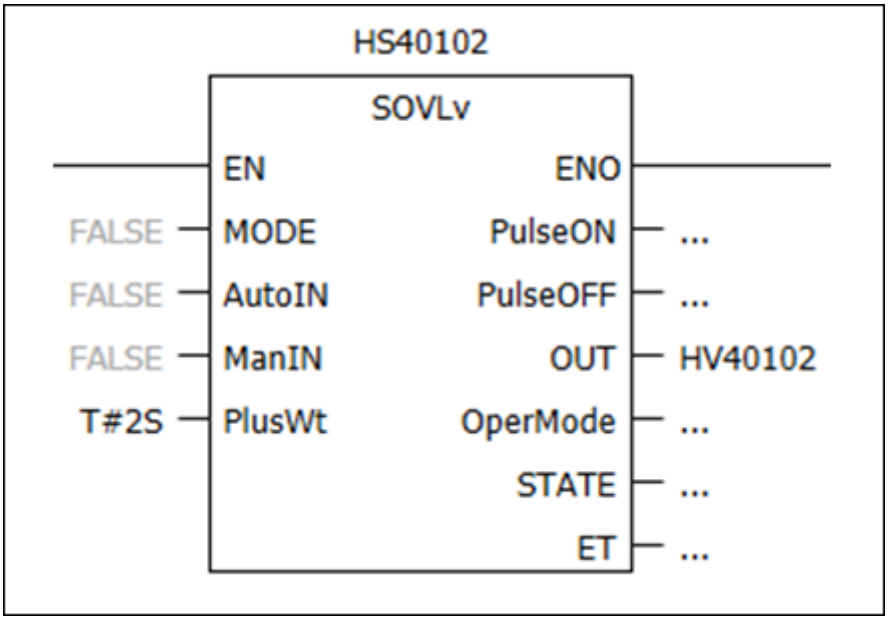
◦ Single DO pump stop control

At a certain chemical plant site, there is no feedback information from the pump, requiring the implementation of the pump's start-stop control.



Single-DO pump shutdown diagram

Control scheme configuration



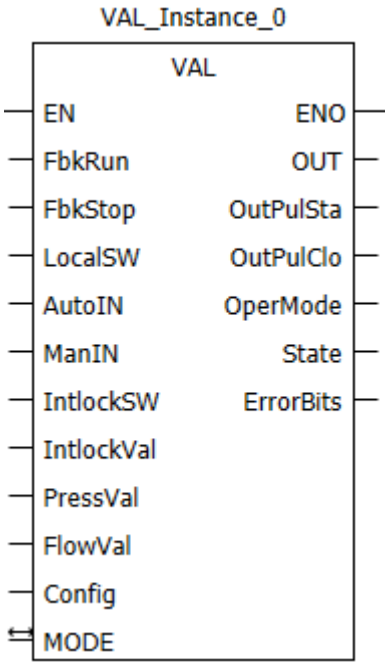
Example

No.	Instance	Type
1	HS40102	Single output control function block
2	HV40102	DO output

18.3.4.5 VAL

This is the valve control block instruction.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
VAL	Valve control block	FB		VAL_Instance_0(FbkRun := , FbkStop := , LocalSW := , AutoIN := , ManIN := , IntlockSW := , IntlockVal := , PressVal := , FlowVal := , Config := , MODE := , OUT => , OutPulSta => , OutPulClo => , OperMode => , State => , ErrorBits =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Data type	Initial Value	Description
Input	FbkRun	Feedback for open limit	BOOL	FALSE	It is valid when FbkRunEN = TRUE in the configuration information.

Input/Output	Variable	Name	Data type	Initial Value	Description
	FbkStop	Feedback for close limit	BOOL	FALSE	It is valid when FbkStopEN = TRUE in the configuration information.
	LocalSW	Local switch	BOOL	FALSE	TRUE: Local mode. FALSE: Remote mode.
	AutoIN	Automatic instruction	BOOL	FALSE	TRUE: Open. FALSE: Close.
	ManIN	Manual instruction	BOOL	FALSE	TRUE: Open. FALSE: Close.
	IntlockSW	Interlock switch	BOOL	FALSE	Interlock switch.
	IntlockVal	Interlock safety output value	BOOL	FALSE	Interlock safety output value.
	PressVal	Pressure	REAL	0.0	Pressure.
	FlowVal	Flow	REAL	0.0	Flow.
	Config	Configuration information	VAL_CONFIG	-	-
	Parameter description				

Variable	Name	Data type	Initial Value	Description
FbkRunEN	Open feedback enabled	BOOL	TRUE	1: Yes. 0: No.
FbkStopEN	Close feedback enabled	BOOL	FALSE	1: Yes. 0: No.
OutPulseT	Pulse output time	TIME	TIME#2s0ms	Pulse output time.
TimeOutT	Timeout threshold	TIME	TIME#2s0ms	Timeout threshold.
MainTenSW	Inspection state	BOOL	FALSE	Inspection state.

Input/Output	Variable	Name	Data type	Initial Value	Description
Input/Output	MODE	Mode	BYTE	0	0: Manual. 1: Automatic.
Output	OUT	Output	BOOL	FALSE	Output.
	OutPulSta	Open pulse output	BOOL	FALSE	Open pulse output.
	OutPulClo	Close pulse output	BOOL	FALSE	Close pulse output.
	OperMode	Operation mode	BYTE	0	0: Manual. 1: Automatic. 2: Interlock. 3: Local. 4: Inspection.
	State	Operation state	BYTE	0	0: Initial. 1: Started. 2: Starting. 3: Stopped. 4: Stopping. 5: Feedback error.
	ErrorBits	Error output	WORD	0	bit0: Start timeout. bit1: Stop timeout. bit2: Feedback from start to stop. bit3: Feedback from stop to start. bit4: Dual-1 feedback error.

- **Function**

  The Valve Control Block controls on-site valves using single DO high/low level signals or dual DO pulse signals. It applies to devices with up to two status feedback signals (open/close limit feedback) and is usually used for various valve control applications.

  - **Operation mode**

Operation Mode	Action
Automatic	Output based on automatic input command.
Manual	Output based on manual input command.
Interlock	When the interlock switch is triggered, the interlock safety value (configurable) is output directly.
Local	In this mode, the device is started/stopped locally via the control box located at the local site or c ontrolled by switch. The system tracks the actual status of the device. If no valid feedback is received from the device, the safety value is output.

Priority order: Local > Interlock > Manual > Automatic

- **Bumpless switchover**

Operation Mode	Action
Local → Automatic	In Local mode, the output interlocks with the running status of the on-site device (if there is no feedback or the feedback status is correct, the output remains unchanged). In Automatic mode, the output is the auto-calculated value directly. This is a bumpy switchover.
Local → Manual	In Local mode, the output interlocks with the running status of the on-site device (if there is no feedback or the feedback status is correct, the output remains unchanged). The manual value is interlocked with the output to ensure bumpless switchover.
Local → Interlock	In Local mode, the output interlocks with the running status of the on-site device (if there is no feedback or the feedback status is correct, the output remains unchanged). In Interlock mode, the output is the interlock safety value directly. This is a bumpy switchover.
Interlock → Automatic	In Interlock mode, the output is the interlock safety value directly. In Automatic mode, the output is the auto-calculated value directly. This is a bumpy switchover.
Interlock → Manual	In Interlock mode, the manual value is interlocked with the interlock safety value to ensure bumpless switchover.
Interlock → Local	In Interlock mode, the output is the interlock safety value. At the moment of switching to Local mode without any operation, the on-site device must also be in the interlock safety value state. After that, when the device is operated on-site, the output interlocks with the running status of the on-site device (if there is no feedback or the feedback status is correct, the output remains unchanged). This is a bumpless switchover.
Manual → Automatic	The output switches directly from the manual output value to the auto-calculated value. This is a bumpy switchover.
Manual → Interlock	The output switches directly from the manual output value to the interlock safety value. This is a bumpy switchover.

Operation Mode	Action
Manual → Local	In Manual mode, the output is the current manual value. At the moment of switching to Local mode without any operation, the on-site device must also be in the manual value state. After that, when the device is operated on-site, the output interlocks with the running status of the on-site device (if there is no feedback or the feedback status is correct, the output remains unchanged). This is a bumpless switchover.
Automatic → Manual	In Automatic mode, the output is the auto-calculated value. At this time, the manual value is interlocked with the auto-calculated value to ensure bumpless switchover.
Automatic → Interlock	The output switches directly from the auto-calculated value to the interlock safety value. This is a bumpy switchover.
Automatic → Local	In Automatic mode, the output is the auto-calculated value. At the moment of switching to Local mode without any operation, the on-site device must also be in the state of automatic calculation. After that, when the device is operated on-site, the output interlocks with the running status of the on-site device (if there is no feedback or the feedback status is correct, the output remains unchanged). This is a bumpless switchover.

- **Timeout alarm**

  Timeout alarm triggers if the function block status feedback disagrees with its output beyond the configured timeout period. Prerequisite: The device has running status feedback.

  For example, the function block issues an "open" instruction to the on-site valve. If the device responds normally and starts within the set time, it returns the startup status to the function block - this is the normal startup process. If no correct feedback is received within the set time, the device is considered not properly opened, triggering a timeout alarm. The timeout detection period is configured by the parameter "Config. TimeOutT", which can be configured based on the actual device response.

  Note: Timeout alarm clears automatically when valid feedback is received or command is switched with no manual acknowledgment required.

  Feedback status and configuration:

Feedback Switch Settings	Feedback Status	
FbkRunEN = TRUE FbkStopEN = FALSE	FbkRun = TRUE	Open feedback
	FbkRun = FALSE	Close feedback
FbkRunEN = FALSE FbkStopEN = TRUE	FbkStop = TRUE	Close feedback
	FbkStop = FALSE	Open feedback
FbkRunEN = TRUE FbkStopEN = TRUE	FbkRun = TRUE FbkStop = FALSE	Open feedback

Feedback Switch Settings	Feedback Status	
	FbkRun = FALSE FbkStop = TRUE	Close feedback
	FbkRun = TRUE FbkStop = TRUE	Feedback error
FbkRunEN = FALSE FbkStopEN = FALSE	No feedback signal	No feedback signal (e.g., send Open instruction → time elapses → deemed Open)

- **Maintenance**

  After a maintenance command is issued, the interlock safety value ILIN is directly output. No automatic, manual, or interlock commands are accepted, ensuring the safety of on-site personnel. In addition, on-site fault signals from will not generate any alarm messages.

- **Output function**

  One DO level signal can be output to the site to control the opening and closing of valves; alternatively, two pulse-type DO signals can also be output to separately control the opening and closing.

  When Open Signal = TRUE and Close Signal = FALSE, the valve is opened; when Open Signal = FALSE and Close Signal = TRUE , the valve is closed. The pulse width is adjustable via Config.OutPulseT. The pulse stops once the set time elapses.

- **Deviation alarm (feedback fluctuation)**

  If the operating signal is lost for more than the set time during motor operation, an alarm is triggered.

- **Numerical display function**

  Pressure and flow in the valve pipeline are displayed (for monitoring only, not used for control).

  Example of working conditions:

Working Condition	OUT	OperMode (Operation Mode)	State (Operation State)	ErrorBits (Error Output) Bitwise Analysis
Initial status	-	Determined by the current specific mode: 0: Manual 1: Automatic 2: Interlock 3: Local 4: Inspection	0: Initial status	0
Open command issued, with open feedback within timeout period	Output open command Output open pulse of configured duration		1: Open limit	0

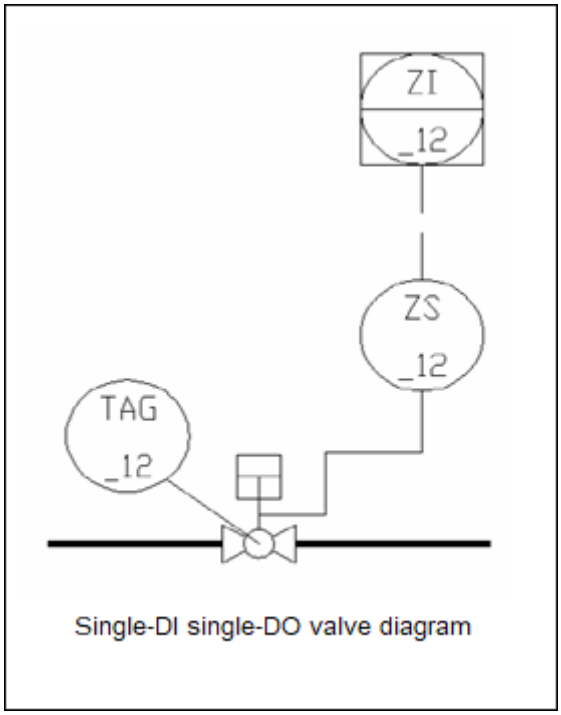
Working Condition	OUT	OperMode (Operation Mode)	State (Operation State)	ErrorBits (Error Output) Bitwise Analysis
Open command issued, with no open feedback received within timeout	Output open command Output open pulse of configured duration		2: Opening	Bit0: Open timeout
Open command issued, with close feedback received within timeout	Output close command Output close pulse of configured duration		3: Close limit	0
Close command issued, with no close feedback received within timeout	Output close command Output close pulse of configured duration		4: Closing	Bit1: Close timeout
Feedback switches from open to close when in open limit	/		5: Feedback error	Bit2: Feedback (Open→Close)
Feedback switches from close to open when in close limit	/			Bit3: Feedback (Close→Open)
Both open and close feedback are 1			5: Feedback error	Bit4: Dual 1 feedback error

When both the open feedback and close feedback are 0, the valve may be in an intermediate running state which must not be deemed a feedback error.

- **Application instance**
  - **Single DI single DO valve**

    A solenoid valve on-site requires open/close control. A valve operation fault alarm be triggered without confirmation if a mismatch occurs between the valve status feedback and the control output after the stroke time has elapsed.

    Configuration

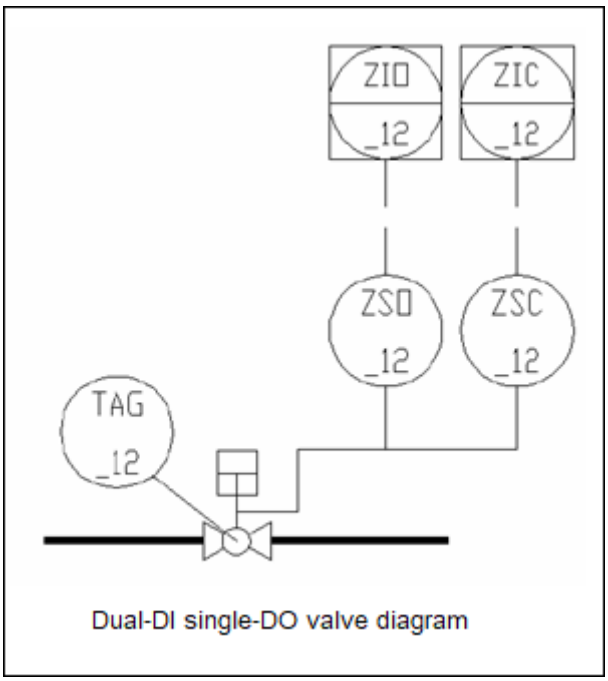


Single-DI single-DO valve diagram

- **Dual DI single DO valve**

  A solenoid valve on a plant unit requires open/close control and operability on the flow chart. A valve operation fault alarm be triggered without confirmation if a mismatch occurs between the valve status feedback and the control output after the stroke time has elapsed.

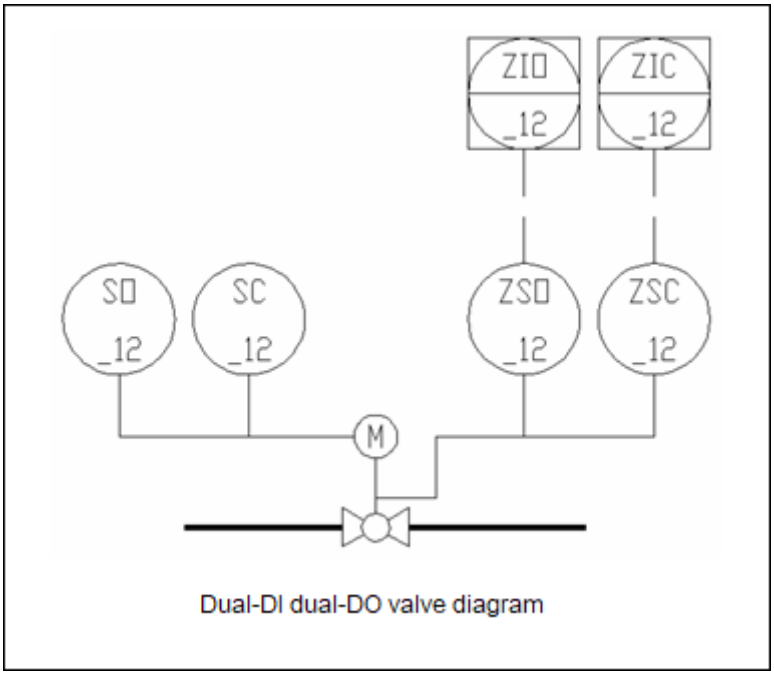
  Configuration



Dual-DI single-DO valve diagram

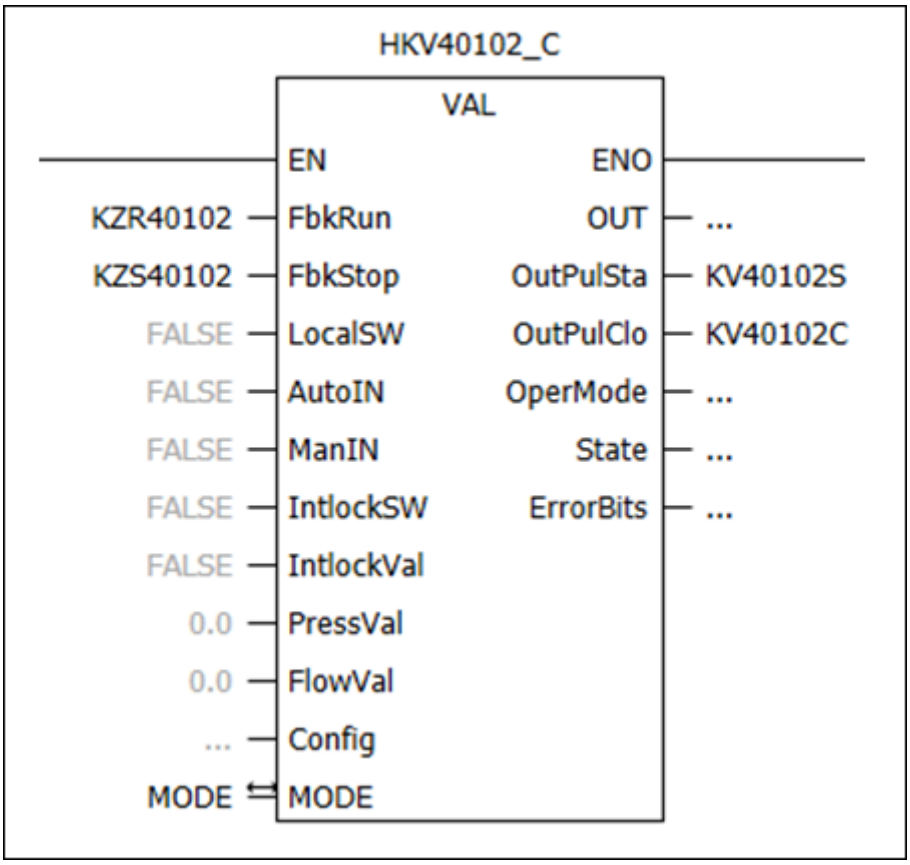
- **Dual DI dual DO valve**

  An electric water valve requires open/close control. A valve operation fault alarm be triggered without confirmation if a mismatch occurs between the valve status feedback and the control output after the stroke time has elapsed.



Dual-DI dual-DO valve diagram

Control scheme configuration



Example

No.	Instance	Type	Remarks
1	HKV40102_C	Valve control function block	-
2	KV40102S	DO output	Open pulse
3	KV40102C	DO output	Close pulse

No.	Instance	Type	Remarks
4	KZR40102	DI input	In the configuration information, FbkRunEN = TRUE is valid when there is open feedback information.
5	KZS40102	DI input	In the configuration information, FbkStopEN = TRUE is valid when there is close feedback information.

18.3.5 High-Speed I/O Instructions

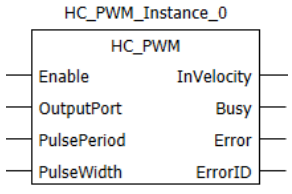
18.3.5.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.3.5.2 HC_PWM	FB	High-speed pulse width modulation	CmpHCHighSpeedIo	Library version 1.0.0.0 and later
18.3.5.3 HC_EnableInterrupt	FB	Enable interrupt		Library version 1.0.0.0 and later

18.3.5.2 HC_PWM

This instruction implements high-speed pulse width modulation (PWM).

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
HC_PWM	High-speed pulse width modulation	FB		HC_PWM_Instance_0(Enable := , OutputPort := , PulsePeriod := , PulseWidth := , InVelocity => , Busy => , Error => , ErrorID =>);

- **Related variables**

  Input variables

Input Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: The instruction is enabled. FALSE: The instruction is not enabled.
OutputPort	Pulse output terminal	BYTE	Positive number	0	PWM output terminal. • EVO400 series: Y0, Y2, Y4, Y6. • EVO500 series: Y0, Y2, Y4, Y6. • EVO700 series: Y0, Y2.
PulsePeriod	Pulse cycle	DINT	5 to 214748364	0	Width (in 100 ns) of a full square wave.
PulseWidth	High-level pulse width	DINT	2 to 214748364	0	Width (in 100 ns) of the high level in a square wave.

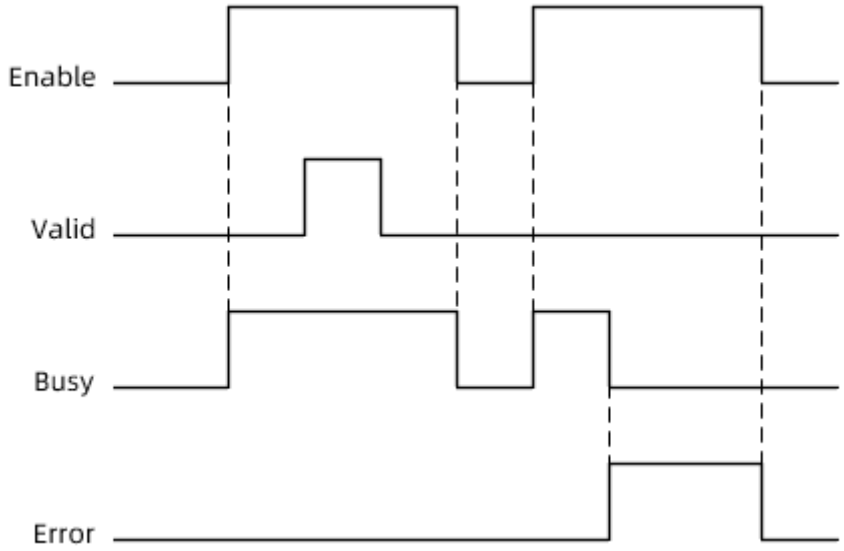
Output variables

Output Variable	Name	Data Type	Valid Range	Default Value	Description
InVelocity	Frequency setpoint reached flag	BOOL	TRUE, FALSE	FALSE	TRUE when the pulse width or frequency reaches the setpoint.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	TRUE when outputting the high-level square wave signal.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	UINT	-	0	A fault code (18.3.5.4 Fault Codes) is output when a fault occurs.

- **Function**
  This function block implements dynamic PWM.

  Taking an EVO500 device as an example, after the terminal Y0/Y2 is set to PWM mode, the Y0+Y1/Y2+Y3 output cannot be controlled by the user program or the device I/O mapping interface of the programming software during the execution of the function block HC_PWM.

- **Timing diagram**



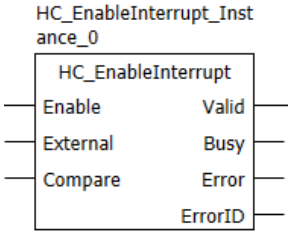
- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.5.3 HC_EnableInterrupt

This instruction enables interrupts.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
HC_EnableInterrupt	Enable interrupt	FB		HC_EnableInterrupt_Instance_0(Enable := , External := , Compare := , Valid => , Busy => , Error => , ErrorID =>);

- **Related variables**

  Input variables

Variable	Name	Data Type	Valid Range	Default Value	Description
Enable	Enable	BOOL	TRUE, FALSE	FALSE	TRUE: The instruction is enabled. FALSE: The instruction is not enabled.

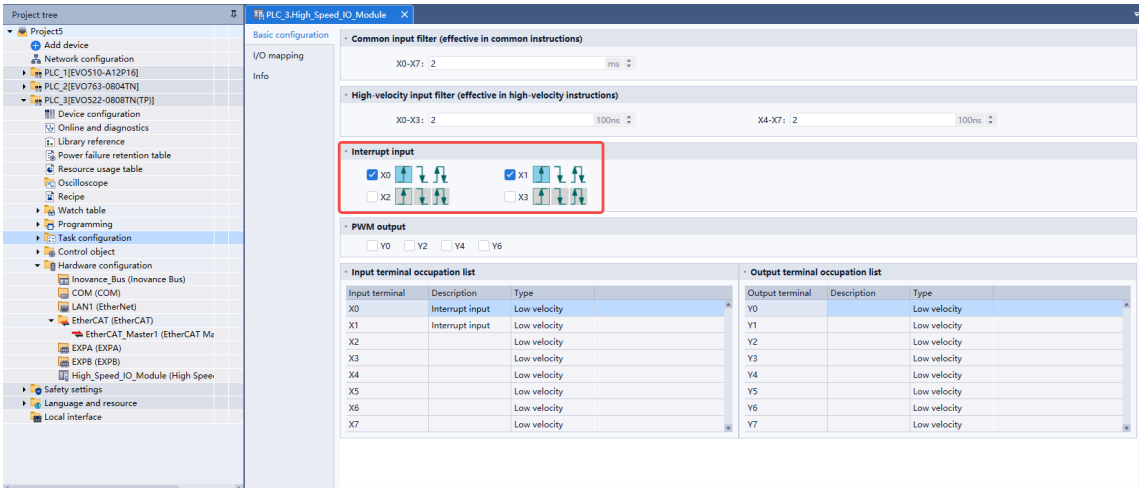
Variable	Name	Data Type	Valid Range	Default Value	Description
External	External input interrupted	UINT	0 to 15	0	Enable external interrupt. For example, if this variable value is set to 3 (binary 2#11), terminals 0 and 1 are enabled.
Compare	Compare interrupt	UINT	0 to 15	0	Enable compare interrupt. For example, if this variable value is set to 3 (binary 2#11), the high-speed counters 0 and 1 compare interrupts are enabled.

Output variables

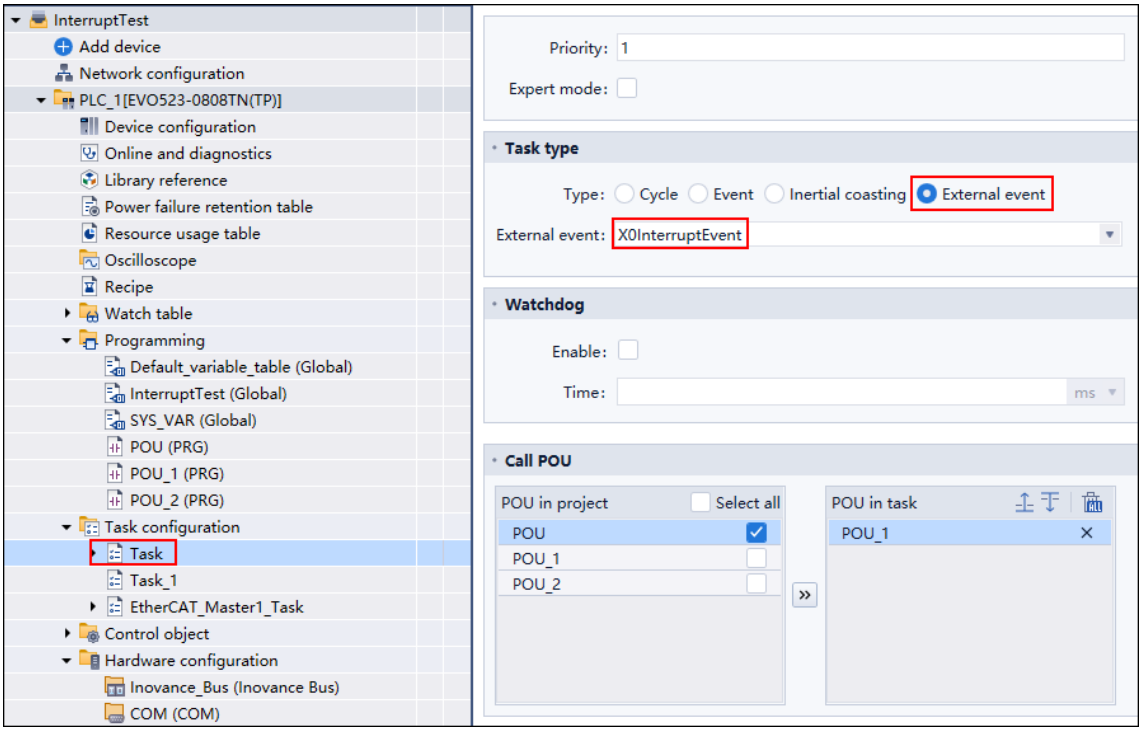
Variable	Name	Data Type	Valid Range	Default Value	Description
Valid	Interrupt valid	BOOL	TRUE, FALSE	FALSE	The interrupt is valid.
Busy	Executing	BOOL	TRUE, FALSE	FALSE	The function block is being executed.
Error	Fault	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
ErrorID	Fault code	UINT	-	0	A fault code (18.3.5.4 Fault Codes) is output when a fault occurs.

- **Function**
  - This function block is used to enable the external interrupt and high-speed counter compare interrupt.
  - Only EVO400, EVO500, and APC500 series PLCs support this function.
  - Instructions for using the external interrupt:

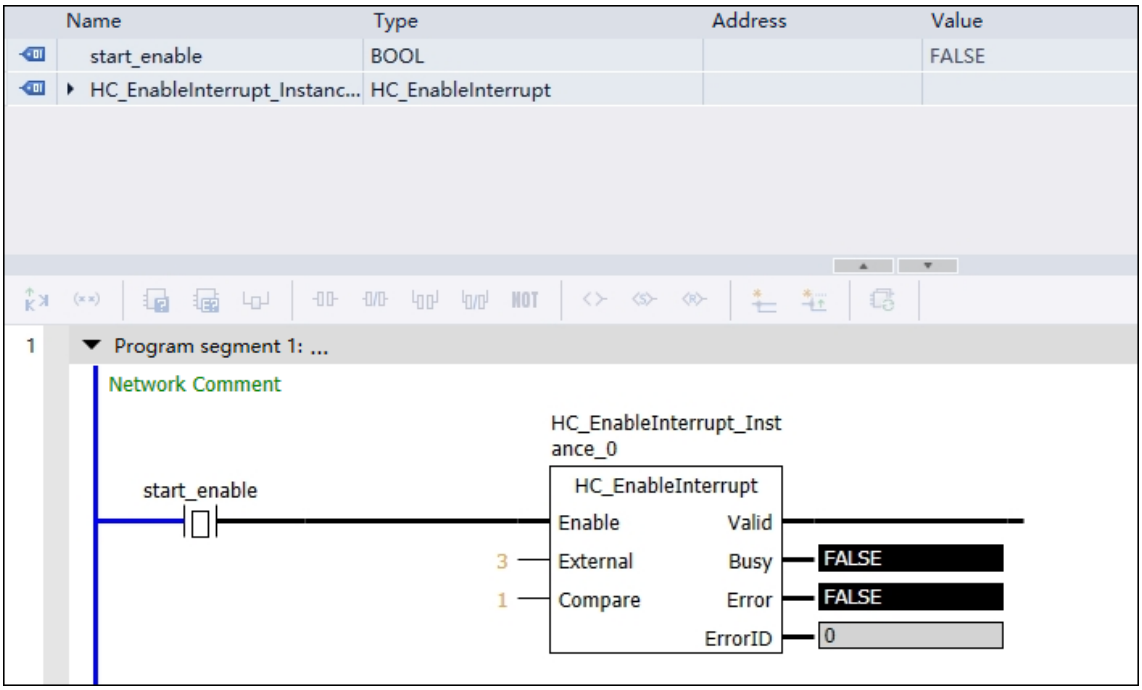
    1. Before enabling the external input interrupt, configure the external interrupt enable and interrupt trigger mode in the software.



    2. Create the task of external interrupt event.

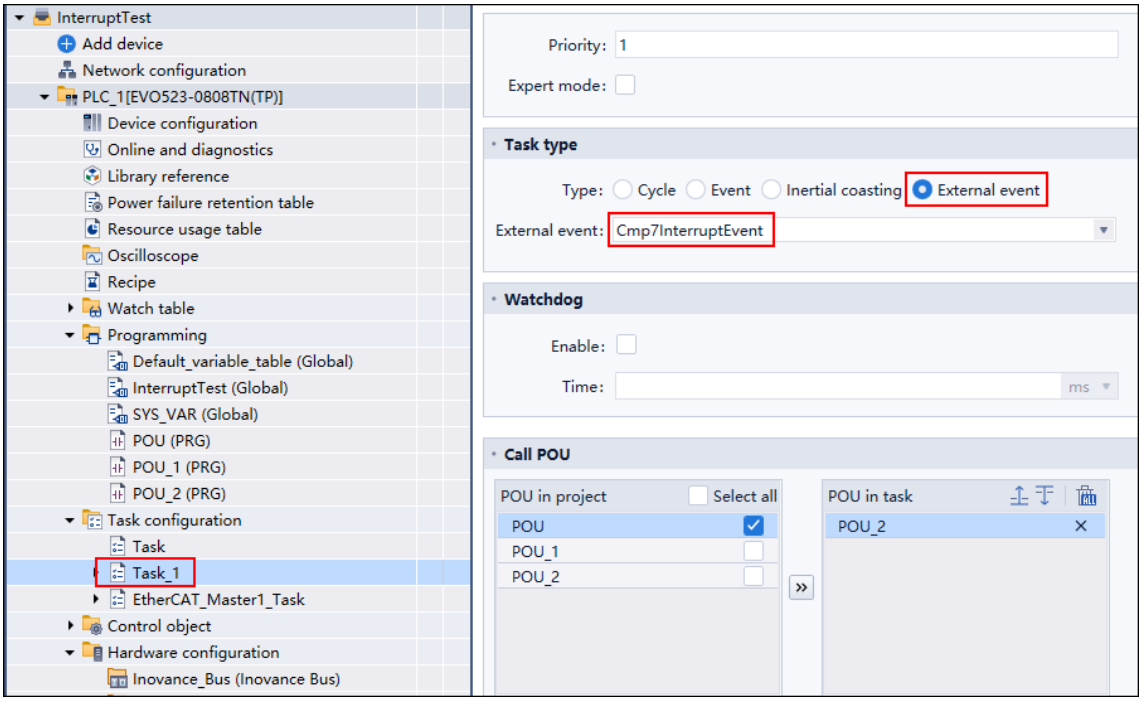


3. Call the HC_EnableInterrupt function block to enable the interrupt.

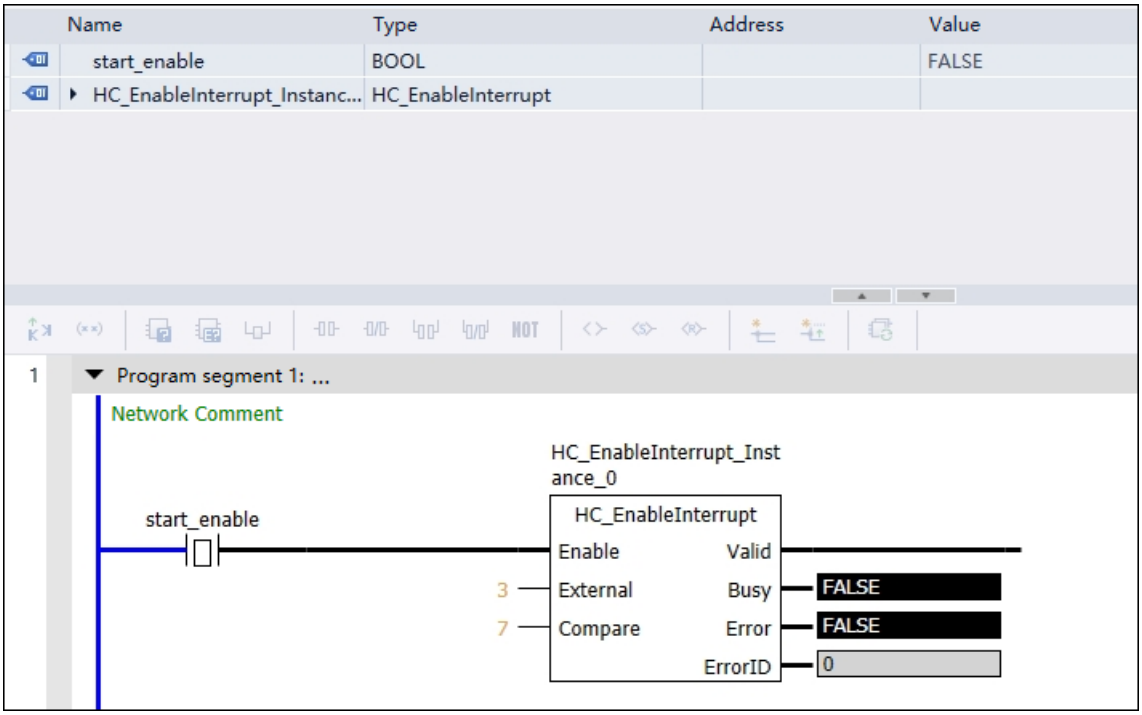


◦ Instructions for using the compare interrupt:

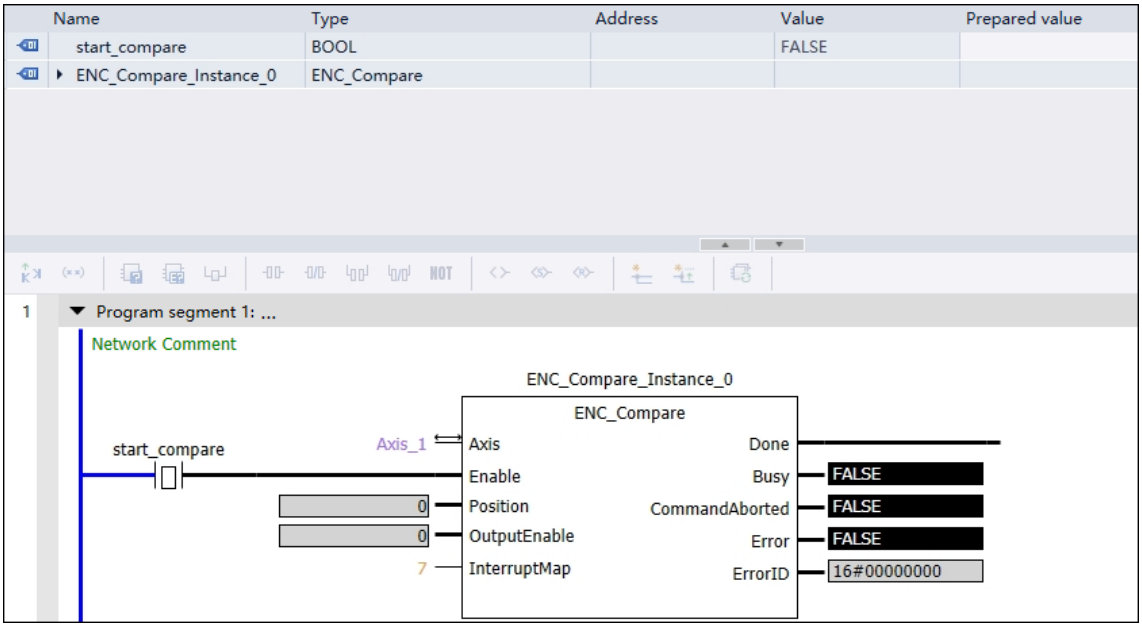
1. Create the task of compare interrupt event.



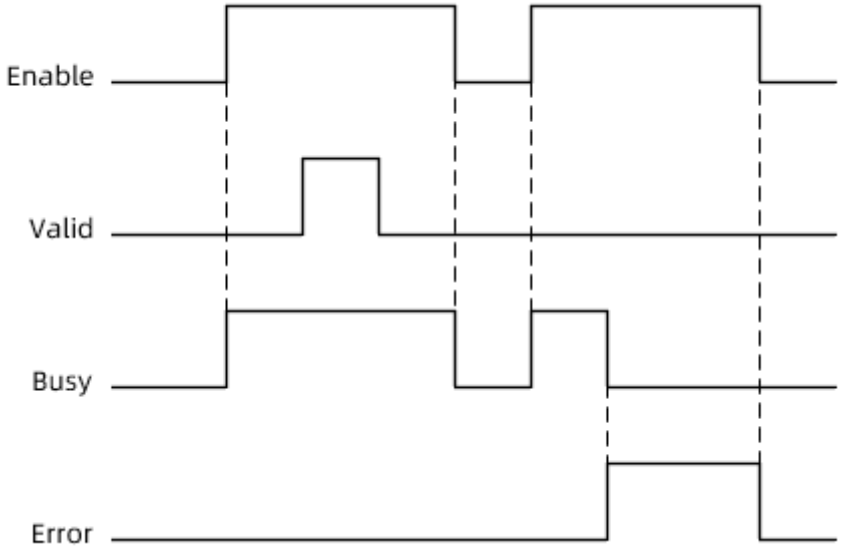
2. Use the HC_EnableInterrupt to enable the high-speed counters 0 to 3 compare interrupts.



3. Before triggering the compare interrupt using ENC_Compare, ENC_ArrayCompare, or ENC_StepCompare, input the created interrupt task to the InterruptMap pin.



- **Timing diagram**



- **Faults**

  For faults related to this instruction with specific descriptions, causes and remedies, see the Fault Codes.

18.3.5.4 Fault Codes

Fault Code (Decimal)	Fault Code (Hexadecimal)	Description	Cause	Remedy
0001	0001	Missing configuration parameter for the PLC's general I/O function: Bus cycle task.	The iFA software version is incompatible. The required parameters are missing in the downloaded device configuration file.	Update the iFA software version. (The current run uses default parameters.)

Fault Code (Decimal)	Fault Code (Hexadecimal)	Description	Cause	Remedy
0002	0002	Axis in commissioning state.	The axis associated with this instruction is currently in commissioning state.	Exit the axis commissioning mode and then execute this instruction.
0003	0003	Invalid terminal setting for HC_PWM.	No appropriate terminal is checked on the iFA interface.	Check the PWM output terminal on the iFA interface.
0004	0004	Terminal conflict for HC_PWM.	Multiple instances are output to the same terminal simultaneously.	Check whether multiple instances are output to the same terminal.
0005	0005	HC_PWM pulse period and pulse width equal to 0.	The pulse period and pulse width are set to 0.	Set a non-zero value.
0006	0006	HC_PWM pulse period equal to 0.	The pulse period is set to 0.	Set a non-zero value.
0007	0007	HC_PWM pulse width equal to 0.	The pulse width is set to 0.	Set a non-zero value.
0008	0008	HC_PWM pulse width exceeding the pulse period.	The pulse width exceeds the pulse period.	Set the pulse width to be less than the pulse period.
0009	0009	HC_PWM pulse period and pulse width less than the minimum values.	The pulse period and pulse width are less than their respective minimum values.	Set the pulse period to be not less than the minimum value 5 and the pulse width to be not less than the minimum value 2.
0010	000A	HC_PWM pulse period less than the minimum value.	The pulse period is less than the minimum value.	Set the pulse period to be not less than the minimum value 5.
0011	000B	HC_PWM pulse width less than the minimum value.	The pulse width is less than the minimum value.	Set the pulse width to be not less than the minimum value 2.

Fault Code (Decimal)	Fault Code (Hexadecimal)	Description	Cause	Remedy
0012	000C	HC_PWM pulse period and pulse width greater than the maximum values.	The pulse period and pulse width are greater than their respective maximum values.	Set the pulse period and pulse width to be not greater than the maximum value 214748364.
0013	000D	HC_PWM pulse period greater than the minimum value.	The pulse period is greater than the maximum value.	Set the pulse period to be not greater than the maximum value 214748364.
0014	000E	HC_PWM pulse width greater than the maximum value.	The pulse width is greater than the maximum value.	Set the pulse width to be not greater than the maximum value 214748364.
0015	000F	HC_EnableInterrupt External value out of limit.	1. The External value is out of limit. 2. The pin for External is not configured in the software.	1. Set the External value to be within the range of 0 to 15. 2. Enable external interrupt for the External pin in the iFA.
0016	0010	HC_EnableInterrupt Compare value out of limit.	1. The Compare value is out of limit. 2. The corresponding channel is not configured with a high-speed counter.	1. Set the Compare value to be within the range of 0 to 15. 2. Assign the high-speed counter channel for Compare to a local encoder axis.
0017	0011	HC_EnableInterrupt enabled repeatedly.	Multiple HC_EnableInterrupt function blocks are not allowed to be triggered.	Use the same HC_EnableInterrupt function block to enable all interrupts.
0018	0012	HC_EnableInterrupt not supported on this device.	The interrupt task is supported on EVO400, EVO500, and APC500 series devices only.	Check whether the PLC model is EVO400, EVO500 or APC500 series.

18.4 Communication and Diagnostics

18.4.1 System Information

18.4.1.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.4.1.2 SysHC_HWInfo	FB	Obtain hardware Information	SysHCPlcInfo	Library version 1.0.0.0 and later
18.4.1.3 SysHC_SWInfo	FB	Obtain software information		Library version 1.0.0.0 and later
18.4.1.4 SysHC_NetworkInfo	FB	Get network information		Library version 1.0.0.0 and later
18.4.1.5 SysHC_NetworkConfig	FB	Configure network properties		Library version 1.0.0.0 and later
18.4.1.6 GetTaskInterval	FC	Obtain task interval		Library version 1.1.0.0 and later
18.4.1.7 GetDeviceDiag	FB	Obtain device diagnostic information		Library version 1.2.0.0 and later
18.4.1.8 GetDeviceStatus	FB	Obtain device state		Library version 1.2.0.0 and later
18.4.1.9 GetTaskInfo	FC	Obtain task information		Library version 1.1.0.0 and later
18.4.1.10 SysHC_GetPersistentDatainfo	FB	Obtain device state		Library version 1.3.0.0 and later
18.4.1.11 SysHC_WritePersistentData	FB	Save backup file		Library version 1.3.0.0 and later
18.4.1.12 SysHC_ReadPersistentData	FB	Read backup file		Library version 1.3.0.0 and later
18.4.1.13 SysHC_Umount	FB	Unmount an external storage device		Library version 1.3.0.0 and later
18.4.1.14 SysHC_GetSystemDate	FB	Get system date		Library version 1.3.0.0 and later
18.4.1.15 SysHC_SetSystemDate	FB	Set system date		Library version 1.3.0.0 and later
18.4.1.16 SysHC_GetSystemTimeZone	FB	Get system timezone		Library version 1.3.0.0 and later

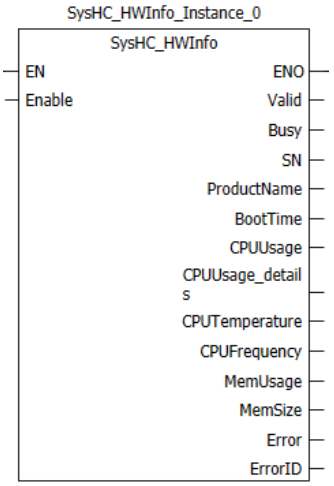
Instruction	FB/FC	Function	Library	Applicable Library Version
18.4.1.17 SysHC_SetSystemTimeZone	FB	Set system timezone		Library version 1.3.0.0 and later
18.4.1.18 SysHC_CPUInfo	FB	Obtain CPU information such as memory, frequency, and temperature		Library version 1.3.0.0 and later
18.4.1.19 SysHC_UDiskPath	FB	Get the system path of the PLC's USB interface		Library version 1.3.0.0 and later
18.4.1.20 GetBootVersion	FC	Get BIOS version		Library version 1.3.0.0 and later
18.4.1.21 GetPLCVersion	FC	Get PLC firmware version		Library version 1.3.0.0 and later
18.4.1.22 GetProductName	FC	Get PLC device name		Library version 1.3.0.0 and later
18.4.1.23 GetSerialNumber	FC	Get the PLC serial number (unique)		Library version 1.3.0.0 and later
18.4.1.24 GetFPGALogicVersion	FC	Get the FPGA logic version		Library version 1.3.0.0 and later
18.4.1.25 SysHC_Appbackup	FB	Backup user program data by calling instruction		Library version 1.3.0.0 and later
18.4.1.26 SysHC_FaultRecord	FB	Get fault data recording by calling instruction		Library version 1.4.0.0 and later
18.4.1.27 SysHC_OperationRecord	FB	Get program data recording by calling instruction		Library versions 1.4.0.0 and later
18.4.1.28 SysHC_AppDownInfo	FB	Get the latest user program download information of the PLC		Library versions 1.4.0.0 and later
18.4.1.29 UTF16ToGB2312	FC	UTF16 to GB2312 encoding conversion		Applicable to library version 1.5.0.0 and later
18.4.1.30 UTF8ToGB2312	FC	UTF8 To GB2312 encoding conversion		Applicable to library version 1.5.0.0 and later
18.4.1.31 GB2312ToUTF16	FC	GB2312 To UTF16 encoding conversion		Applicable to library version 1.5.0.0 and later
18.4.1.32 GB2312ToUTF8	FC	GB2312 To UTF8 encoding conversion		Applicable to library version 1.5.0.0 and later
18.4.1.33 SDCard1StatusGet	FC	Get all the status flags of SD memory card		Applicable to library version 1.5.0.0 and later
18.4.1.34 SysHC_NtpClient	FB	NTP time sync		Applicable to library version 1.5.0.0 and later
18.4.1.35 SysHC_NetworkInfoV2	FB	Get network information		Applicable to library version 1.5.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.4.1.36 SysHC_NetworkConfigV2	FB	Configure network properties		Applicable to library version 1.5.0.0 and later

18.4.1.2 SysHC_HWInfo

This instruction obtains the hardware information of the device, including the total power-on time, average CPU usage, CPU usage per core, CPU temperature, CPU frequency, memory utilization, and total memory size.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SysHC_HWInfo	Obtain hardware Information	FB		SysHC_HWInfo_Instance_0(Enable := , Valid =>, Busy => , SN => , ProductName => , BootTime => , CPUUsage => , CPUUsage_details => , CPUTemperature => , CPUFrequency => , MemUsage => , MemSize => , Error => , ErrorID =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Enable	Active high	BOOL	TRUE, FALSE	FALSE	Active High.
Output	Valid	Output active	BOOL	TRUE, FALSE	FALSE	Instruction is actively executing.
	Busy	Instruction executing	BOOL	TRUE, FALSE	FALSE	Instruction is executing.
	SN	Product SN code	STRING	-	''	Product SN code.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	ProductName	Product model	STRING	-	''	Product model.
	BootTime	Startup time	STRING	-	''	Total power-on time.
	CPUUsage	CPU usage	BYTE	0 to 100	0	Average CPU usage.
	CPUUsage_details	CPU usage per core	STRING	-	''	CPU usage per core.
	CPUTemperature	CPU temperature	INT	-32768 to 32767	0	CPU temperature in °C
	CPUFrequency	CPU frequency	STRING	-	''	CPU frequency in MHz.
	MemUsage	Memory usage	BYTE	0 to 100	0	Memory utilization.
	MemSize	Memory size	Word	0 to 65535	0	Memory size in M.
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs.
	ErrorID	Fault code	SYS_HC_ERROR	-	NO_ERROR	See Fault Codes.

- **Function**

  This instruction obtains the hardware information of the device.

- **Program example**

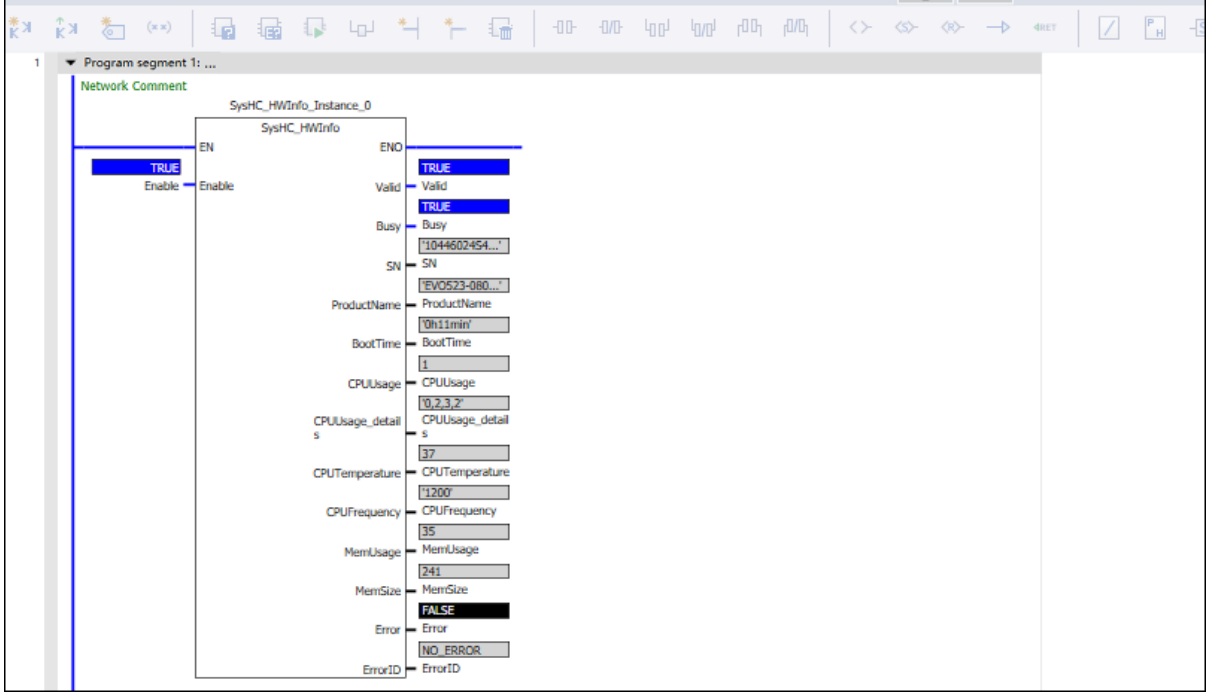
  ST:

	Name	Data type	Address	Value
1	SysHC_HWInfo_Instance_0	SysHC_HWInfo		
2	Enable	BOOL		TRUE
3	Valid	BOOL		TRUE
4	Busy	BOOL		TRUE
5	SN	STRING		'10446024S4160010'
6	ProductName	STRING		'EVO523-0808TN'
7	BootTime	STRING		'0h11min'
8	CPUUsage	BYTE		1
9	CPUUsage_details	STRING		'0,2,3,2'
10	CPUTemperature	BYTE		37
11	CPUFrequency	STRING		'1200'
12	MemUsage	BYTE		35
13	MemSize	WORD		241
14	Error	BOOL		FALSE
15	ErrorID	SYS_HC_ERROR		NO_ERROR

```
SysHC_HWInfo_Instance_0(
Enable := Enable,
Valid => Valid,
Busy => Busy,
SN => SN,
ProductName => ProductName,
BootTime => BootTime,
CPUUsage => CPUUsage,
CPUUsage_details => CPUUsage_details,
CPUTemperature => CPUTemperature,
CPUFrequency => CPUFrequency,
MemUsage => MemUsage,
MemSize => MemSize,
Error => Error,
ErrorID => ErrorID);RETURN
```

LD:

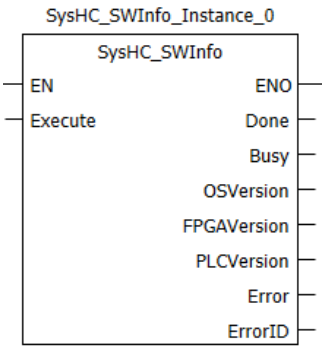
	Name	Data type	Address	Value
1	SysHC_HWInfo_Instance_0	SysHC_HWInfo		
2	Enable	BOOL		TRUE
3	Valid	BOOL		TRUE
4	Busy	BOOL		TRUE
5	SN	STRING		'10446024S4160010'
6	ProductName	STRING		'EVO523-0808TN'
7	BootTime	STRING		'0h11min'
8	CPUUsage	BYTE		1
9	CPUUsage_details	STRING		'0,2,3,2'
10	CPUTemperature	BYTE		37
11	CPUFrequency	STRING		'1200'
12	MemUsage	BYTE		35
13	MemSize	WORD		241
14	Error	BOOL		FALSE
15	ErrorID	SYS_HC_ERROR		NO_ERROR



18.4.1.3 SysHC_SWInfo

This instruction obtains the software information of the device, including the PLC system software version, FPGA software version, and controller firmware version.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SysHC_SWInfo	Obtain software information	FB		SysHC_SWInfo_Instance_0(Execute := , Done =>, Busy => , OSVersion => , FPGAVersion => , PLCVersion => , Error => , ErrorID =>);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Rising edge-triggered	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
Output	Done	Done	BOOL	TRUE, FALSE	FALSE	Instruction is completed.
	Busy	Instruction executing	BOOL	TRUE, FALSE	FALSE	Instruction is executing.
	OSVersion	System software version	STRING	-	' '	System software version.
	FPGAVersion	FPGA software version	STRING	-	' '	FPGA software version.
	PLCVersion	Controller firmware version	STRING	-	' '	Controller firmware version.
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs.
	ErrorID	Fault code	SYS_HC_ERROR	-	NO_ERROR	See Fault Codes.

- **Function**

  This instruction obtains device software information.

- **Program example**

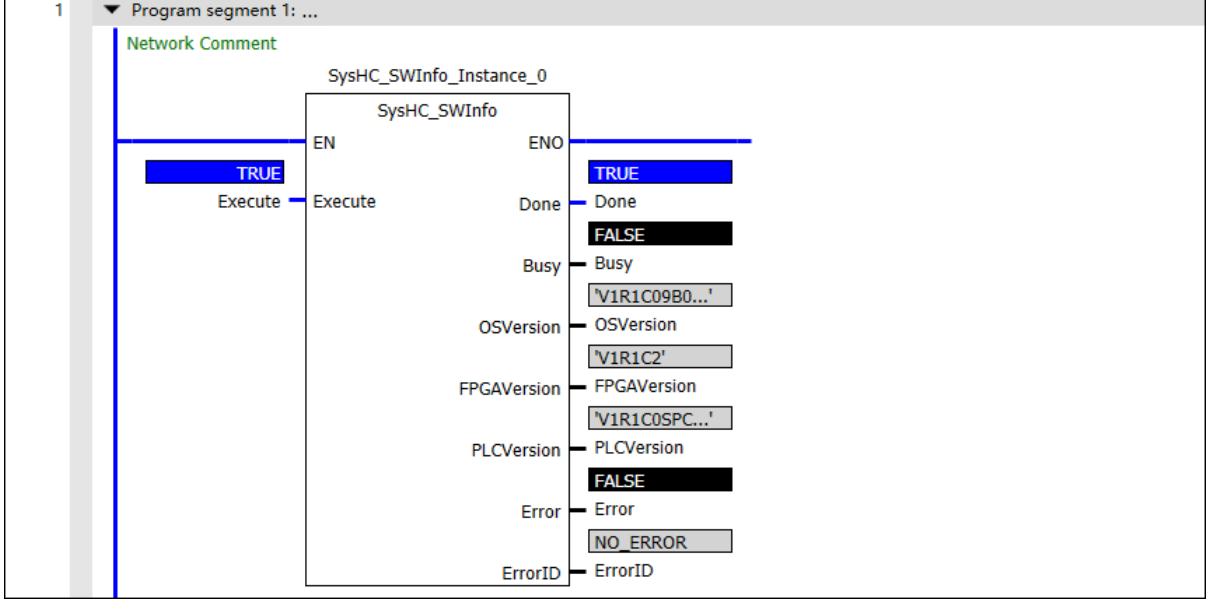
  ST:

	Name	Data type	Address	Value
1	SysHC_SWInfo_Instance_0	SysHC_SWInfo		
2	Execute	BOOL		TRUE
3	Done	BOOL		TRUE
4	Busy	BOOL		FALSE
5	OSVersion	STRING		'V1R1C09B06_20240907_2538'
6	FPGAVersion	STRING		'V1R1C2'
7	PLCVersion	STRING		'V1R1C0SPC0B10Beta'
8	Error	BOOL		FALSE
9	ErrorID	SYS_HC_ERROR		NO_ERROR

```
SysHC_SWInfo_Instance_0(
Execute TRUE := Execute TRUE,
Done TRUE => Done TRUE,
Busy FALSE => Busy FALSE,
OSVersion 'V1R1C09B06_20240907_25... => OSVersion 'V1R1C09B06_20240907_25...,
FPGAVersion 'V1R1C2' => FPGAVersion 'V1R1C2',
PLCVersion 'V1R1C0SPC0B10Be... => PLCVersion 'V1R1C0SPC0B10Be...,
Error FALSE => Error FALSE,
ErrorID NO_ERROR => ErrorID NO_ERROR );RETURN
```

LD:

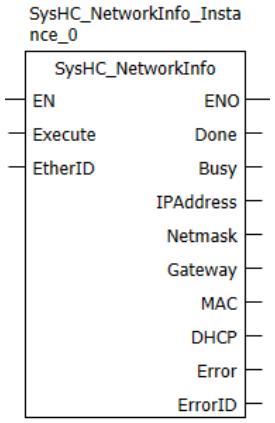
	Name	Data type	Address	Value
1	SysHC_SWInfo_Instance_0	SysHC_SWInfo		
2	Execute	BOOL		TRUE
3	Done	BOOL		TRUE
4	Busy	BOOL		FALSE
5	OSVersion	STRING		'V1R1C09B06_20240907_2538'
6	FPGAVersion	STRING		'V1R1C2'
7	PLCVersion	STRING		'V1R1C0SPC0B10Beta'
8	Error	BOOL		FALSE
9	ErrorID	SYS_HC_ERROR		NO_ERROR



18.4.1.4 SysHC_NetworkInfo

This instruction obtains the network information of the device, including IP address, subnet mask, gateway information, and DHCP status.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SysHC_NetworkInfo	Get network information	FB		SysHC_NetworkInfo_Instance_0(Execute := , EtherID := , Done => , Busy => , IPAddress => , Netmask => , Gateway => , MAC => , DHCP => , Error => , ErrorID =>);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Rising edge-triggered	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	EtherID	Network port ID	BYTE	1 ~ 255	0	Network port ID number.
Output	Done	Done	BOOL	TRUE, FALSE	FALSE	Instruction is completed.
	Busy	Instruction executing	BOOL	TRUE, FALSE	FALSE	Instruction is executing.
	IPAddress	IP address	STRING	-	''	IP address.
	Netmask	Subnet mask	STRING	-	''	Subnet mask.
	Gateway	Gateway information	STRING	-	''	Gateway information.
	MAC	MAC address	STRING	-	''	MAC address.
	DHCP	DHCP status	BOOL	TRUE, FALSE	FALSE	DHCP status.
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	ErrorID	Fault code	SYS_HC_ERROR	-	NO_ERROR	See Fault Codes.

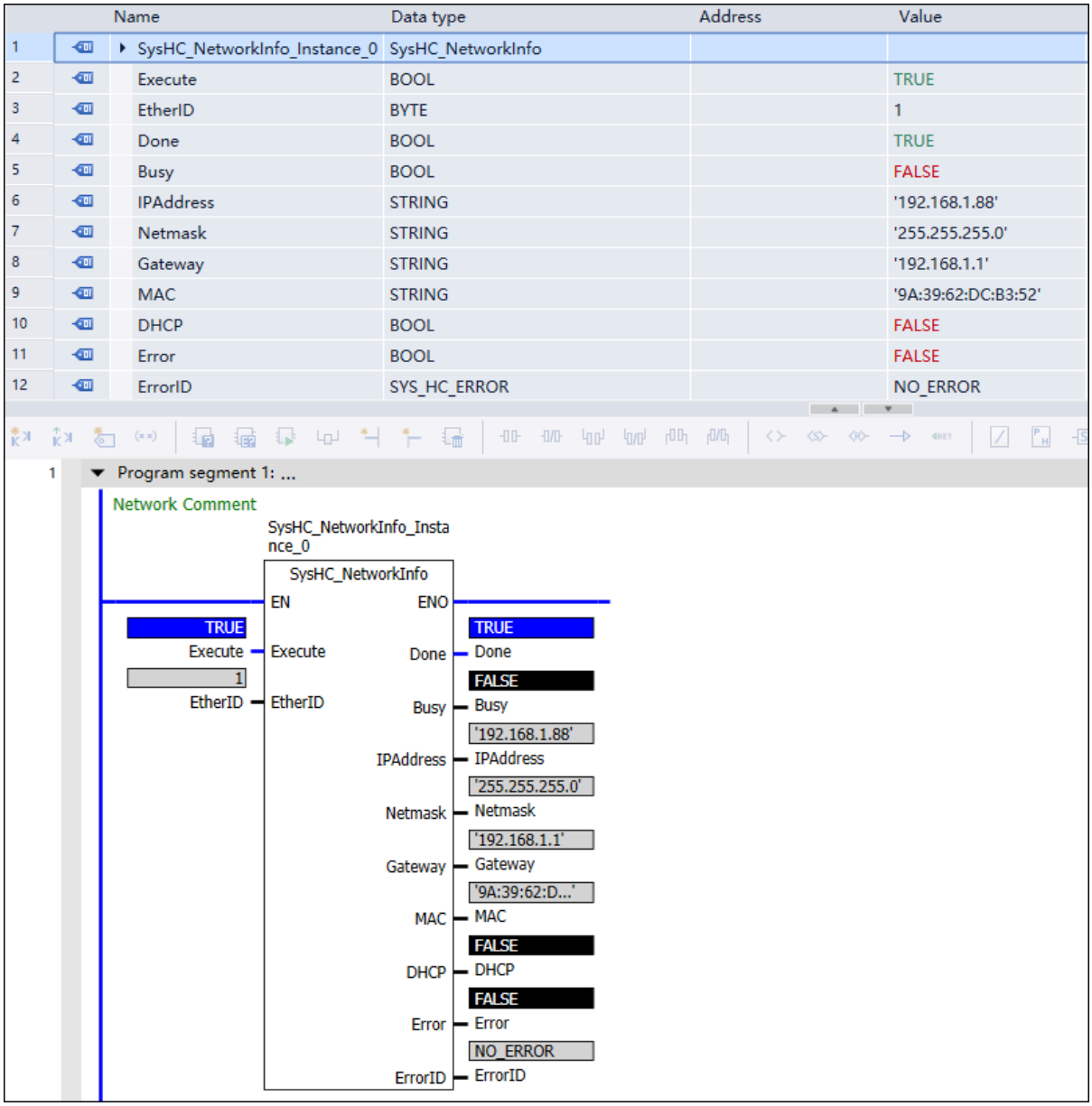
- **Function**
  - Devices that do not support DHCP default to FALSE.
  - The EtherID must correspond to the actual hardware Ethernet port, starting from 1.
- **Program example**

  ST:

	Name	Data type	Address	Value
1	SysHC_NetworkInfo_Instance_0	SysHC_NetworkInfo		
2	Execute	BOOL		TRUE
3	EtherID	BYTE		1
4	Done	BOOL		TRUE
5	Busy	BOOL		FALSE
6	IPAddress	STRING		'192.168.1.88'
7	Netmask	STRING		'255.255.255.0'
8	Gateway	STRING		'192.168.1.1'
9	MAC	STRING		'9A:39:62:DC:B3:52'
10	DHCP	BOOL		FALSE
11	Error	BOOL		FALSE
12	ErrorID	SYS_HC_ERROR		NO_ERROR

```
SysHC_NetworkInfo_Instance_0(
Execute := Execute,
EtherID := EtherID,
Done => Done,
Busy => Busy,
IPAddress => IPAddress,
Netmask => Netmask,
Gateway => Gateway,
MAC => MAC,
DHCP => DHCP,
Error => Error,
ErrorID => ErrorID);RETURN
```

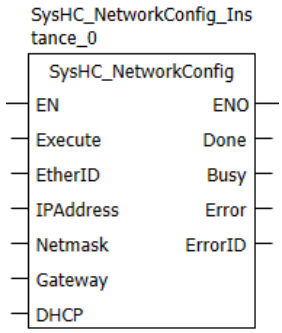
  LD:



18.4.1.5 SysHC_NetworkConfig

This instruction configures the network information of device, including the IP address, subnet mask, and gateway information.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SysHC_NetworkConfig	Configure network properties	FB		SysHC_NetworkConfig_Instance_0(Execute := , EtherID:= , IPAddress := , Netmask := , Gateway := , DHCP := , Done => , Busy =>, Error =>, ErrorID =>);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Rising edge-triggered	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	EtherID	Network port ID	BYTE	1 ~ 255	0	Network port ID.
	IPAddress	IP address	STRING	-	''	IP address.
	Netmask	Subnet mask	STRING	-	''	Subnet mask.
	Gateway	Gateway information	STRING	-	''	Gateway information.
	DHCP	DHCP status	BOOL	TRUE, FALSE	FALSE	DHCP status.
Output	Done	Done	BOOL	TRUE, FALSE	FALSE	Instruction is completed.
	Busy	Instruction executing	BOOL	TRUE, FALSE	FALSE	Instruction is executing.
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs.
	ErrorID	Fault code	SYS_HC_ERROR	-	NO_ERROR	See Fault Codes.

- **Function**
  - This instruction currently does not support setting DHCP.
  - The EtherID must correspond to the actual hardware Ethernet port, starting from 1 and so on.

- **Program example**

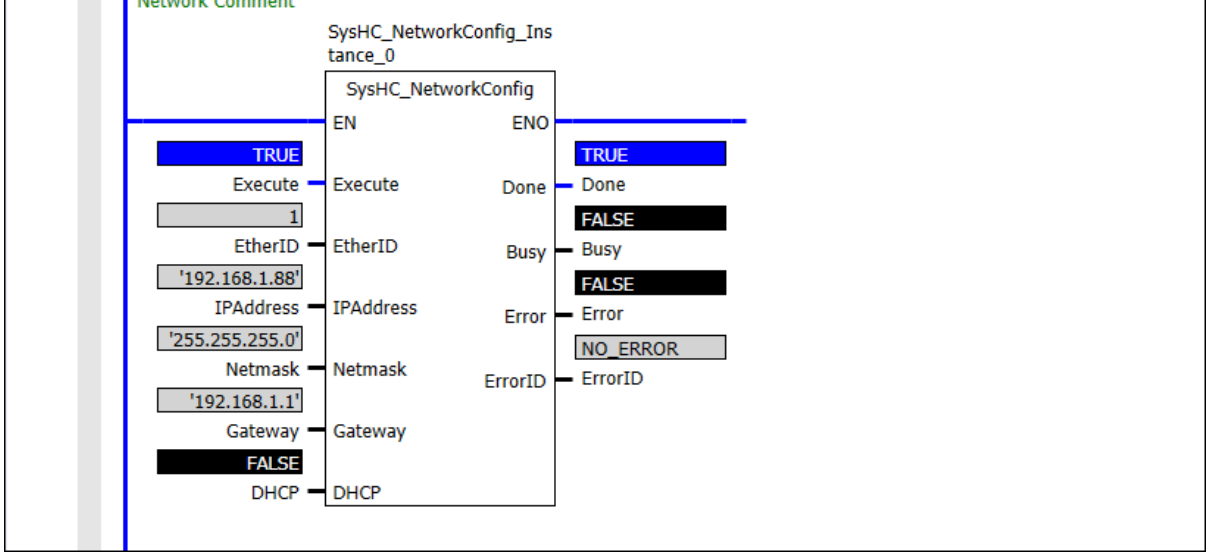
ST:

	Name	Data type	Address	Value
1	SysHC_NetworkConfig_Instanc...	SysHC_NetworkConfig		
2	Execute	BOOL		TRUE
3	EtherID	BYTE		1
4	IPAddress	STRING		'192.168.1.88'
5	Netmask	STRING		'255.255.255.0'
6	Gateway	STRING		'192.168.1.1'
7	DHCP	BOOL		FALSE
8	Done	BOOL		TRUE
9	Busy	BOOL		FALSE
10	Error	BOOL		FALSE
11	ErrorID	SYS_HC_ERROR		NO_ERROR

```
SysHC_NetworkConfig_Instance_0(
Execute TRUE := Execute TRUE ,
EtherID 1 := EtherID 1 ,
IPAddress '192.168.1.88' := IPAddress '192.168.1.88' ,
Netmask '255.255.255.0' := Netmask '255.255.255.0' ,
Gateway '192.168.1.1' := Gateway '192.168.1.1' ,
DHCP FALSE := DHCP FALSE ,
Done TRUE => Done TRUE ,
Busy FALSE => Busy FALSE ,
Error FALSE => Error FALSE ,
ErrorID NO_ERROR => ErrorID NO_ERROR );RETURN
```

LD:

	Name	Data type	Address	Value
1	SysHC_NetworkConfig_Instanc...	SysHC_NetworkConfig		
2	Execute	BOOL		TRUE
3	EtherID	BYTE		1
4	IPAddress	STRING		'192.168.1.88'
5	Netmask	STRING		'255.255.255.0'
6	Gateway	STRING		'192.168.1.1'
7	DHCP	BOOL		FALSE
8	Done	BOOL		TRUE
9	Busy	BOOL		FALSE
10	Error	BOOL		FALSE
11	ErrorID	SYS_HC_ERROR		NO_ERROR



18.4.1.6 GetTaskInterval

This instruction obtains task interval.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetTaskInterval	Obtain task interval	FC		:= GetTaskInterval();

- **Related variables**

  Variable

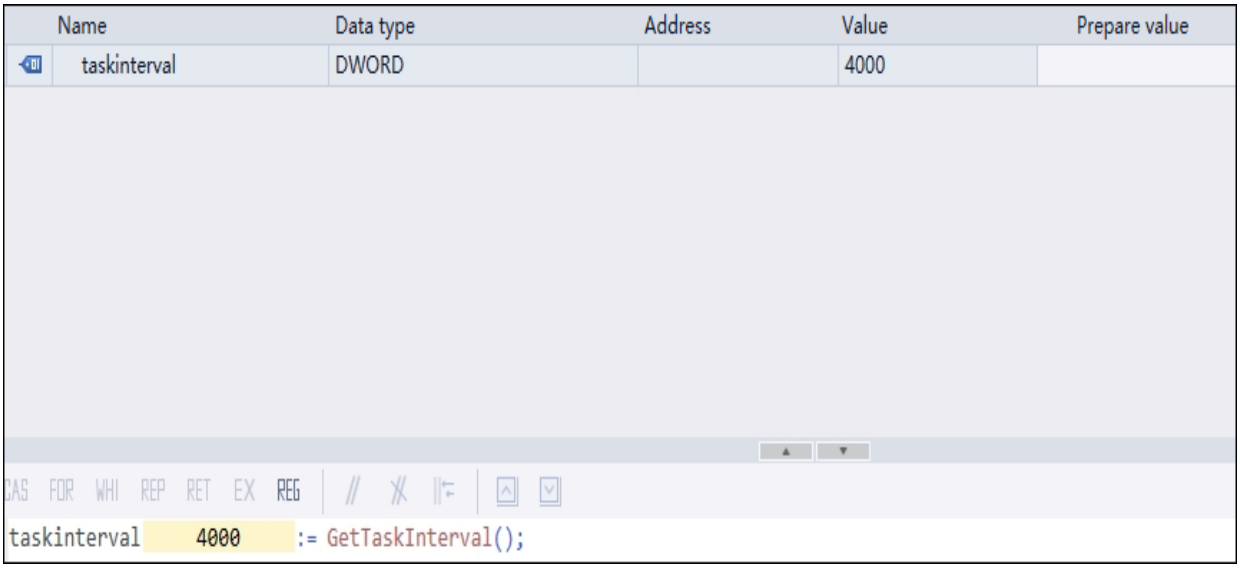
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	GetTaskInterval	Task interval	DWORD	Depends on data type	0	Task interval in us.

- **Function**

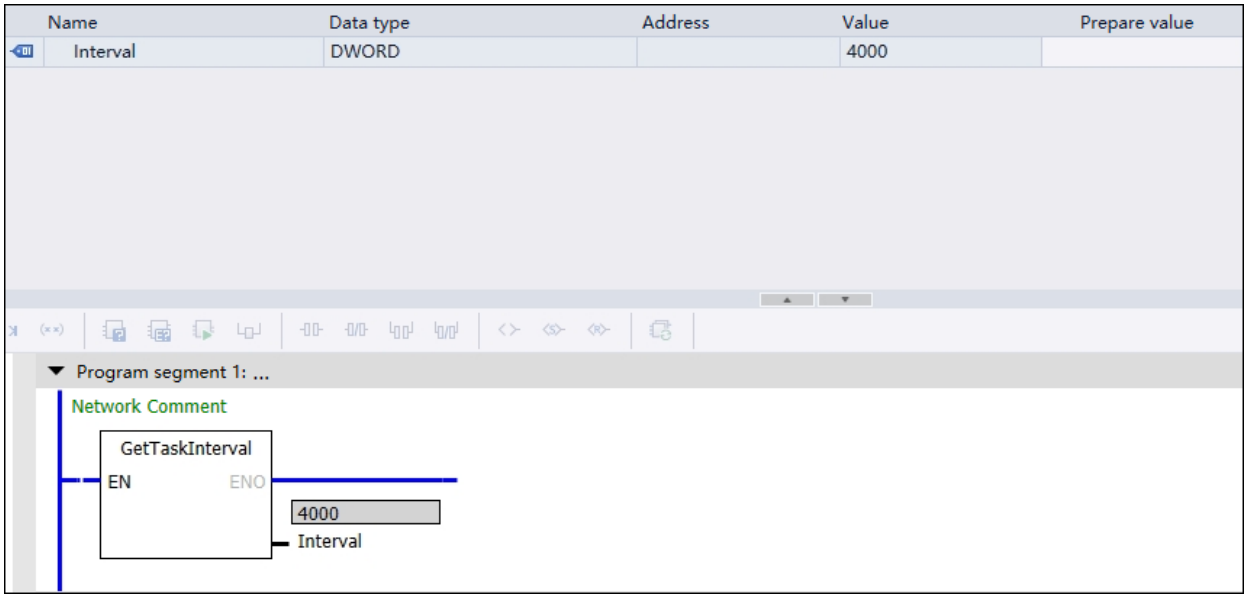
  This instruction is valid only for periodic tasks, and other tasks a value of 0.return 0.

- **Program example**

  ST:



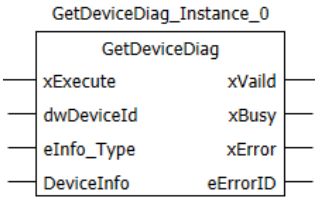
  LD:



18.4.1.7 GetDeviceDiag

This instruction obtains device diagnostic information and device status.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetDeviceDiag	Obtain device diagnostic information	FB		GetDeviceDiag_Instance_0(xExecute := , dwDeviceId := , eInfo_Type := , DeviceInfo := , xVaild => , xBusy => , xError => , eErrorID =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Enable	BOOL	TRUE, FALSE	FALSE	SF_R_TRIG is a rising edge-triggered instruction.
	dwDeviceId	Device ID	DWORD	0 to FFFFFFFF	0	The identifier ID of the device, see 10.6.1.2 Device ID for details.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	eInfo_Type	Information category	Info_Type	0 to 3	0	Select the device status or device diagnostic information to be obtained. 1: DeviceStatus (device status) 2: Reserved 3: DeviceDiag (device diagnostic information)
	DeviceInfo	Device info	ANY	Determined by the user-defined DeviceInfo type	0	Returned diagnostic information or device status.
Output	xVaild	Valid instruction	BOOL	TRUE, FALSE	FALSE	It indicates whether the instruction output is valid.
	xBusy	Instruction in use	BOOL	TRUE, FALSE	FALSE	It Indicates whether the instruction is being called.
	xError	Instruction error	BOOL	TRUE, FALSE	FALSE	It indicates whether the instruction call ha s failed.
	eErrorID	Instruction error ID	ErrorID	0 to 4	0	For fault codes generated when an instruction error occurs, see Fault description for details.

- **Function**
  - When the input parameter "xExecute" changes from FALSE to TRUE, the instruction is enabled and can be used to obtain diagnostic information after certain conditions are met.
  - You can determine whether the expected output is diagnostic information or device status by inputting different eInfo_type values. Diagnostic information is received as an array of DeviceDiag structures defined in the library. Device status is received by DeviceInfo using the DeviceStatus (ENUM type) data defined in the library.
    - Internal members of the DeviceDiag structure data type

Element	Note
ErrorTime	Fault occurrence time
ErrorCode	Fault code
ExternalCode	Extended fault code
GeneralStateCode	General status code

Element	Note
ExternalStateCode	Extended status code

- Enumerated values of the DeviceStatus data type

DeviceStatus Value	Note
1	Disable: The device is disabled
2	Error: The device is in a fault state.
3	Normal: The device is in normal operation.

- **Program example**

  Input dwDeviceID corresponding to the device to be diagnosed, and select eInfo_type as needed to determine whether to obtain device diagnostic information or device status.

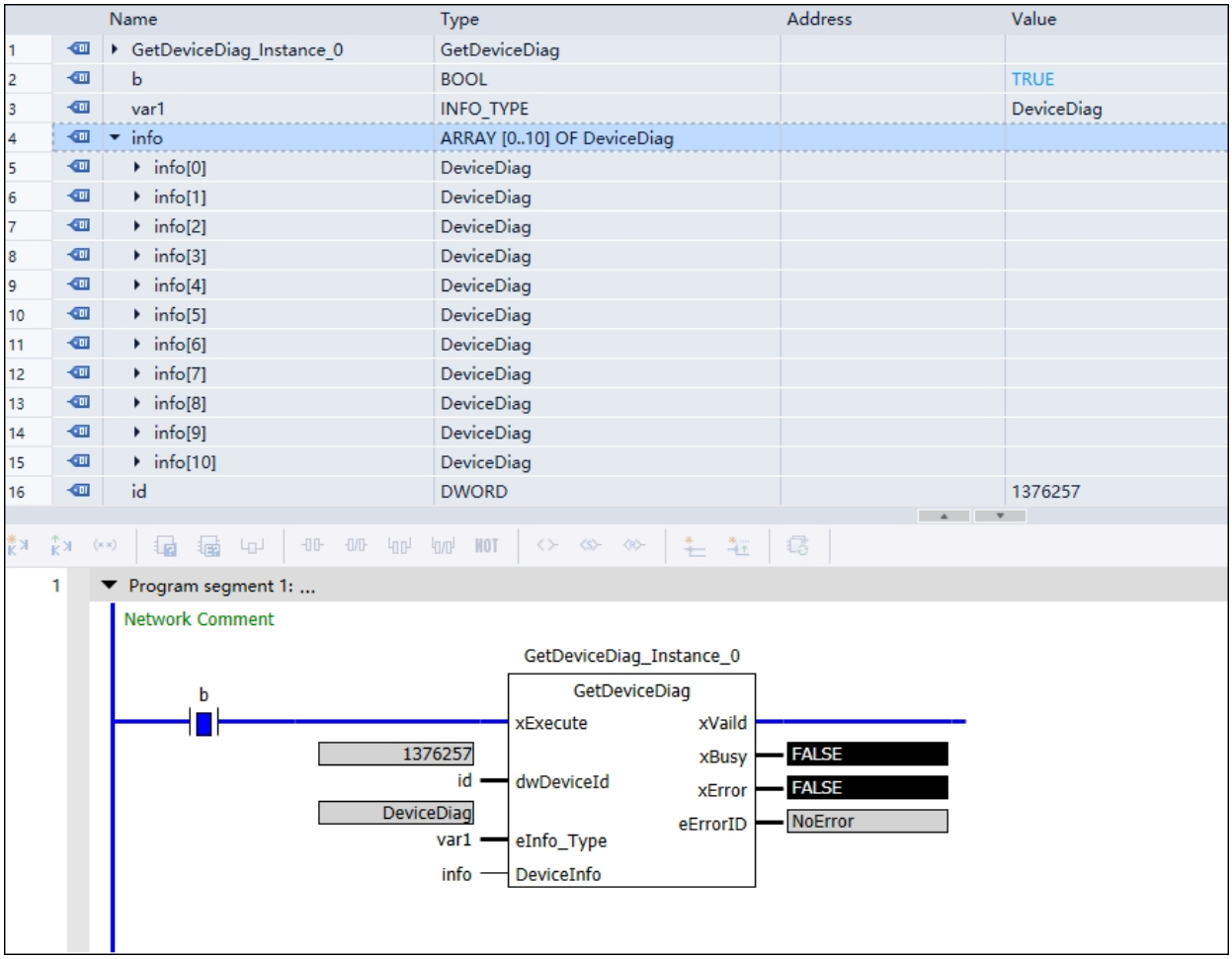
  - When eInfo_type is input as DeviceStatus, the data type pointed to by DeviceInfo must be DeviceStatus.
  - When eInfo_type is input as DeviceDiag, the data type pointed to by DeviceInfo must be an array, where the array elements are the DeviceDiag structure defined in the library.

  ST:

	Name	Type	Address	Value
1	GetDeviceDiag_Instance_0	GetDeviceDiag		
2	executeFlag	BOOL		TRUE
3	Id	DWORD		1376257
4	DiagType	INFO_TYPE		DeviceDiag
5	DiagInfo	ARRAY [0..10] OF DeviceDiag		
6	DiagInfo[0]	DeviceDiag		
7	ErrorTime	STRING		'1970/1/7 6:13:28'
8	ErrorCode	STRING		'0x400001'
9	ExternelCode	STRING		'0x0'
10	GeneralStateCode	STRING		'0x0'
11	ExternelStateCode	STRING		'0x0'
12	DiagInfo[1]	DeviceDiag		
13	DiagInfo[2]	DeviceDiag		
14	DiagInfo[3]	DeviceDiag		
15	DiagInfo[4]	DeviceDiag		
16	DiagInfo[5]	DeviceDiag		
17	DiagInfo[6]	DeviceDiag		
18	DiagInfo[7]	DeviceDiag		
19	DiagInfo[8]	DeviceDiag		
20	DiagInfo[9]	DeviceDiag		
21	DiagInfo[10]	DeviceDiag		
22	Out_VaildFlag	BOOL		TRUE
23	Out_BusyFlag	BOOL		FALSE
24	Out_ErrorFlag	BOOL		FALSE
25	Out_ErrorID	ERRORID		NoError

```
GetDeviceDiag_Instance_0(
xExecute TRUE := executeFlag TRUE,
dwDeviceId 1376257 := Id 1376257 ,
eInfo_TypeDeviceD... := DiagTypeDeviceD...,
DeviceInfo :=DiagInfo ,
xVaild TRUE => Out_VaildFlag TRUE ,
xBusy FALSE => Out_BusyFlag FALSE ,
xError FALSE =>Out_ErrorFlag FALSE ,
eErrorID NoError =>Out_ErrorID NoError );RETURN
```

  LD:



- **Fault description**

  When an error occurs, find the corresponding error via eErrorID, and locate the root cause of the error.

Fault code	Value	Description
0	NoError	No fault.
1	DeviceIDError	DeviceID error (for scenarios where the DeviceID is out of range).
2	NoSupport	The device does not support this type (for the device corresponding to DeviceID, either the device node has not been created or the device does not support this diagnostic information).
3	NotMatch	The information category does not match the diagnostic information category bound by the user.
4	GetFiled	Information acquisition failed.

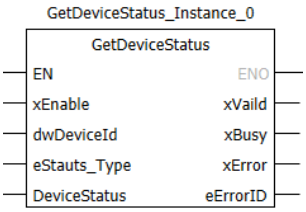
- **Note:**

  The user determines whether the expected output is diagnostic information or device status based on the input value of eInfo_type. Diagnostic information must be received by DeviceInfo in the form of a structure array, where the structure is defined as DeviceDiag in the library, and the number of array elements is customizable by the user. Device status needs to be received using the DeviceStatus (ENUM type) data as DeviceInfo. If eInfo_type does not match the specified receiving data type, the instruction will report an error: NotMatch, indicating that the information category is inconsistent with the diagnostic information category bound by the user.

18.4.1.8 GetDeviceStatus

This instruction obtains the device status in real time.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetDeviceStatus	Obtain device state	FB		GetDeviceStatus_Instance_0(xEnable := , dwDeviceId := , eStauts_Type := , DeviceStatus := , xVaild => , xBusy => , xError => , eErrorID =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Enable instruction	BOOL	TRUE, FALSE	FALSE	Active High.
	dwDeviceId	Device ID	DWORD	0 to FFFFFFFF	0	The identifier ID of the device, see 10.6.1.2 Device ID for details.
	eStatus_Type	Information category	Status_Type	0 to 2	0	Select to obtain device status. 1: DeviceStatus: Device status. 2: Reserved.
	DeviceStatus	Device info	ANY	Determined by the user-defined DeviceStatus type	0	The returned device status.
Output	xVaild	Valid instruction	BOOL	TRUE, FALSE	FALSE	It indicates whether the instruction output is valid.
	xBusy	Instruction in use	BOOL	TRUE, FALSE	FALSE	It Indicates whether the instruction is being called.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	xError	Instruction error	BOOL	TRUE, FALSE	FALSE	It indicates whether the instruction call has failed.
	eErrorID	Instruction error ID	ErrorID	0 to 4	0	For fault codes generated when an instruction error occurs, see Fault description for details.

- **Function**
  - When the input parameter "xEnable" = TRUE, the diagnostic instruction enters the enabled state. After the operable conditions are met, diagnostic information acquisition can be performed. When the parameter "xEnable" is FALSE, diagnostic information acquisition is stopped.
  - The output device status is determined by the input eStatus_Type. The device status must be received using the DeviceStatus (ENUM type) data defined in the library.
    - Enumerated values of the DeviceStatus data type

DeviceStatus Value	Note
1	Disable: The device is disabled
2	Error: The device is in a fault state.
3	Normal: The device is in normal operation.

- **Program example**

  Input the DeviceID corresponding to the device that needs diagnosis, select eStatus_Type, and then get the device status. When the eStatus_Type = DevieStatus, the received data type pointed to by DeviceStatus must be DeviceStatus.

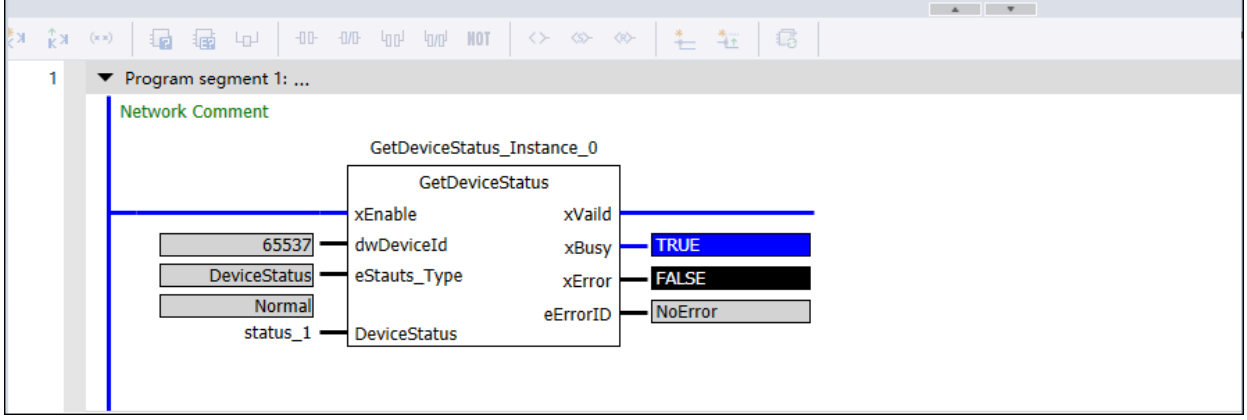
  ST:

	Name	Data type	Address	Value
1	GetDeviceStatus_Instance_0	GetDeviceStatus		
2	xEnable	BOOL		TRUE
3	dwDeviceId	DWORD		65537
4	eStauts_Type	STATUS_TYPE		DeviceStatus
5	DeviceStatus	ANY		
6	xVaild	BOOL		TRUE
7	xBusy	BOOL		TRUE
8	xError	BOOL		FALSE
9	eErrorID	ERRORID		NoError
10	enable	BOOL		TRUE
11	ID	DWORD		65537
12	status	DEVICESTATUS		Normal

```
GetDeviceStatus_Instance_0(xEnable TRUE :=enable TRUE ,
                           dwDeviceId 65537 := ID 65537 ,
                           eStauts_Type := ,
                           DeviceStatus :=status Normal ,
                           xVaild => ,
                           xBusy => ,
                           xError => ,
                           eErrorID => );
```

LD:

	Name	Type	Address	Value
1	GetDeviceStatus_Instance_0	GetDeviceStatus		
2	xEnable	BOOL		TRUE
3	dwDeviceId	DWORD		65537
4	eStauts_Type	STATUS_TYPE		DeviceStatus
5	DeviceStatus	ANY		
6	xVaild	BOOL		TRUE
7	xBusy	BOOL		TRUE
8	xError	BOOL		FALSE
9	eErrorID	ERRORID		NoError
10	status_1	DEVICESTATUS		Normal



- **Fault description**

  When an error occurs, find the corresponding error via eErrorID, and locate the root cause of the error.

Fault code	Value	Description
0	NoError	No fault.
1	DeviceIDError	DeviceID error (for scenarios where the DeviceID is out of range).
2	NoSupport	The device does not support this type (for the device corresponding to DeviceID, either the device node has not been created or the device does not support this diagnostic information).

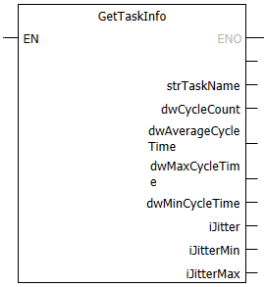
Fault code	Value	Description
3	NotMatch	The information category does not match the diagnostic information category bound by the user.
4	GetFiled	Information acquisition failed.

- **Note**
  When obtaining the device status, DeviceStatus (ENUM type) is used as the DeviceStatus receiver. If Status_type does not match the specified receiving data type, the instruction will report an error: NotMatch, indicating that the information category is inconsistent with the diagnostic information category bound by the user.

18.4.1.9 GetTaskInfo

This instruction obtains the task information including task name/cycle count/average execution time/max execution time/min execution time/jitter time/min jitter/max jitter.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetTaskInfo	Obtain task information	FC		:= GetTaskInfo(strTaskName => , dwCycleCount => , dwAverageCycleTime => , dwMaxCycleTime => , dwMinCycleTime => , iJitter => , iJitterMin => , iJitterMax =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	GetTaskInfo	Instruction execution result	SYS_HC_ERROR	Follows data type	NO_ERROR	It is the result of instruction execution. For details, see Fault Codes.
	strTaskName	Task name	STRING	Follows data type	''	Task name.
	dwCycleCount	Number of cycles	DWORD	Depends on data type	0	Number of cycles.
	dwAverageCycleTime	Average execution time	DWORD	Depends on data type	0	Average execution time in ms

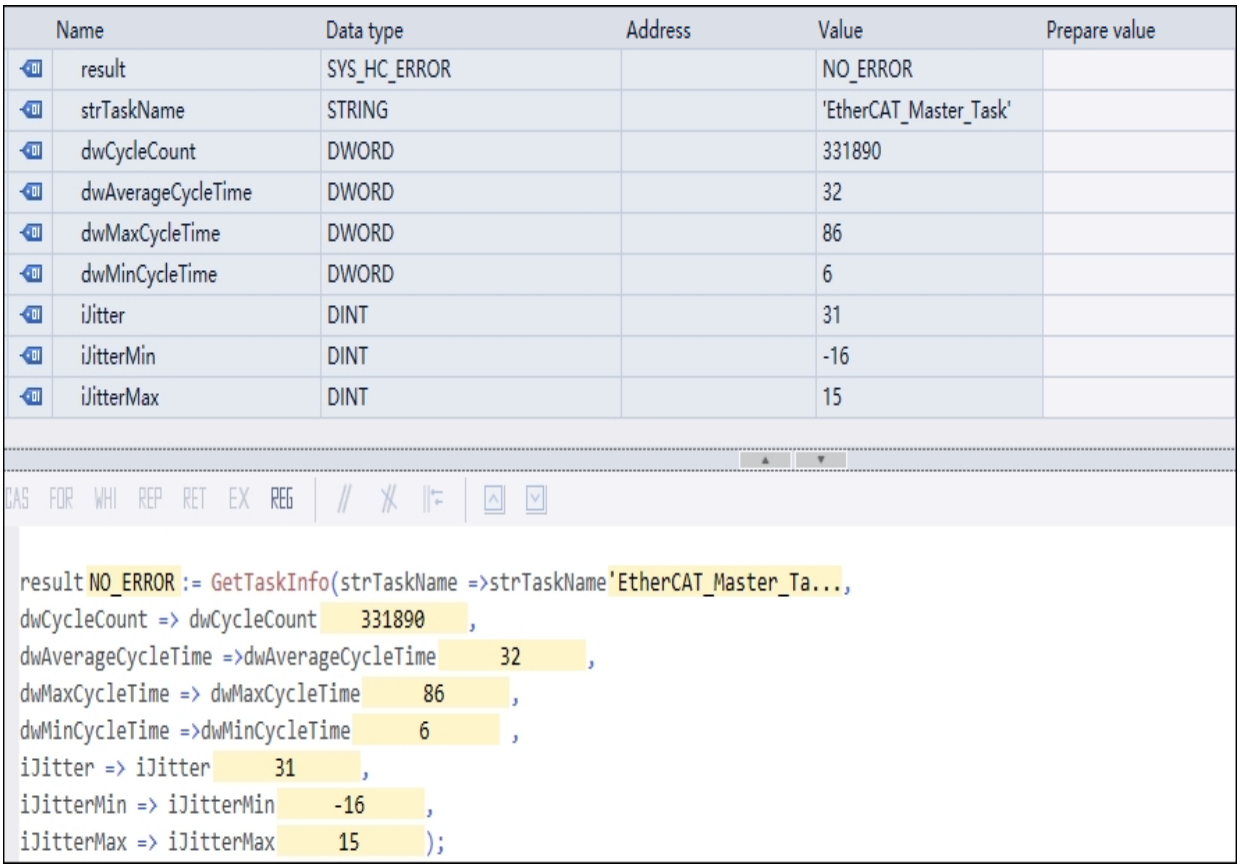
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	dwMaxCycleTime	Maximum execution time	DWORD	Depends on data type	0	Maximum execution time in us.
	dwMinCycleTime	Minimum execution time	DWORD	Depends on data type	0	Minimum execution time in us.
	iJitter	Jitter time	DINT	Follows data type	0	Jitter time in us.
	iJitterMin	Minimum jitter time	DINT	Follows data type	0	Minimum jitter time in us.
	iJitterMax	Maximum jitter time	DINT	Follows data type	0	Maximum jitter time in us.

- **Function**

  Obtainment successful: GetTaskInfo returns NO_ERROR. Obtainment failed: GetTaskInfo returns FAILURE_OBTAIN.

- **Program example**

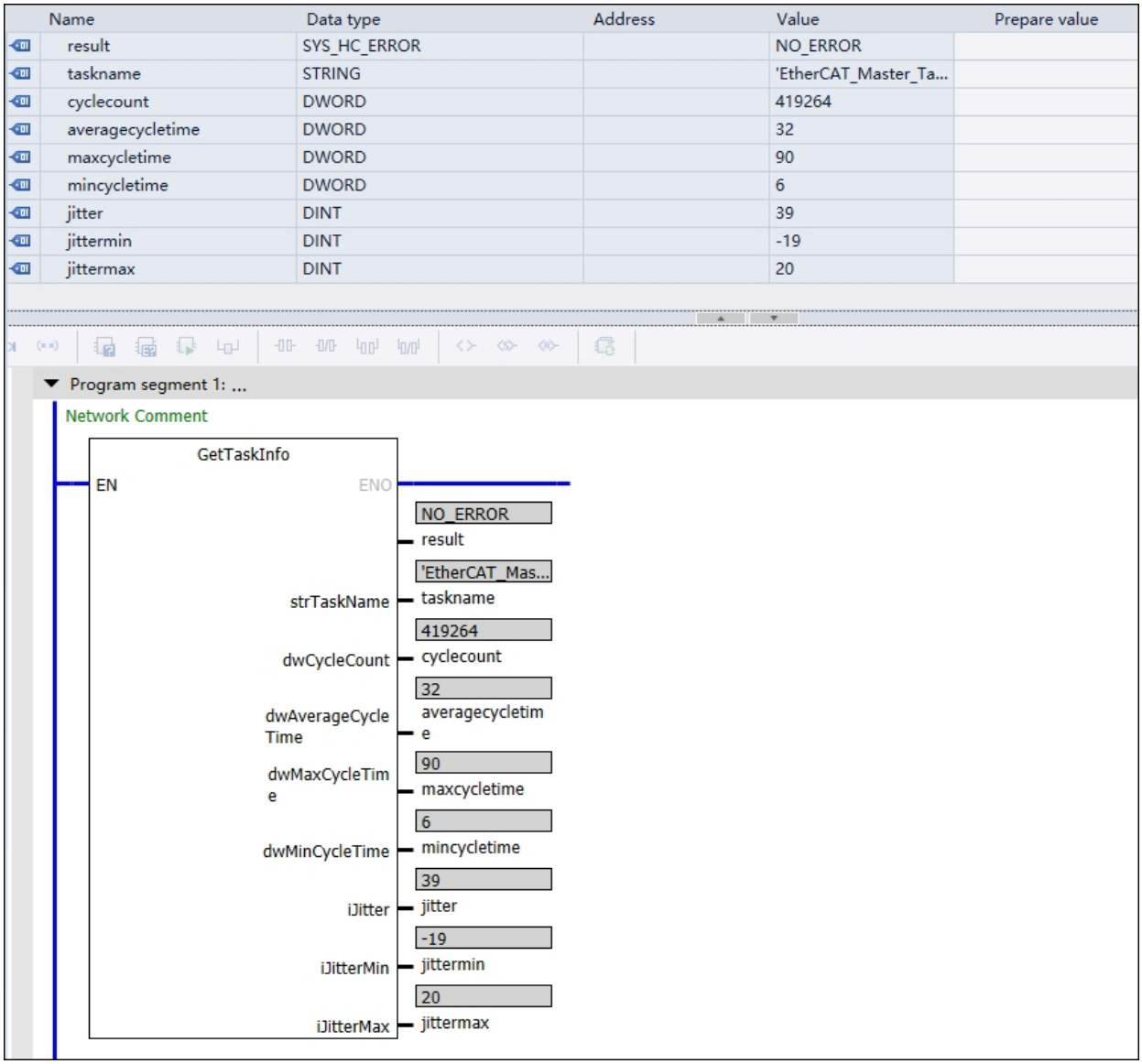
  ST:



Name	Data type	Address	Value	Prepare value
result	SYS_HC_ERROR		NO_ERROR	
strTaskName	STRING		'EtherCAT_Master_Task'	
dwCycleCount	DWORD		331890	
dwAverageCycleTime	DWORD		32	
dwMaxCycleTime	DWORD		86	
dwMinCycleTime	DWORD		6	
iJitter	DINT		31	
iJitterMin	DINT		-16	
iJitterMax	DINT		15	

```
result NO_ERROR := GetTaskInfo(strTaskName =>strTaskName'EtherCAT_Master_Ta...,
dwCycleCount => dwCycleCount    331890    ,
dwAverageCycleTime =>dwAverageCycleTime    32    ,
dwMaxCycleTime => dwMaxCycleTime    86    ,
dwMinCycleTime =>dwMinCycleTime    6    ,
iJitter => iJitter    31    ,
iJitterMin => iJitterMin    -16    ,
iJitterMax => iJitterMax    15    );
```

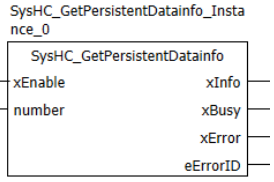
  LD:



18.4.1.10 SysHC_GetPersistentDatainfo

This instruction enables user programs to read information from retentive backup files.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SysHC_GetPersistentDatainfo	Obtain device state	FB		SysHC_GetPersistentDatainfo_Instance_0(xEnable := , number := , xInfo => , xBusy => , xError => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Active high	BOOL	TRUE, FALSE	FALSE	Active High.
	number	Backup file index	PERSISTENT_SUB	1 to 5	1	Backup file sub-index, supporting backup of up to five files PERSISTENT_1: File 1. PERSISTENT_2: File 2. PERSISTENT_3: File 3. PERSISTENT_4: File 4. PERSISTENT_5: File 5.
Output	xInfo	Backup file information	PERSISTENT_INFO	-	NULL	• strProName: Name of the project for the backup file. • strReadTime: Read the backup timestamp. • strWriteTime: Read the backup timestamp.
	xBusy	Instruction executing	BOOL	TRUE, FALSE	FALSE	Executing
	xError	Fault	BOOL	TRUE, FALSE	FALSE	Error.
	eErrorID	Fault ID	PERSISTENT_ERROR	-	NO_ERROR	See Fault Codes.

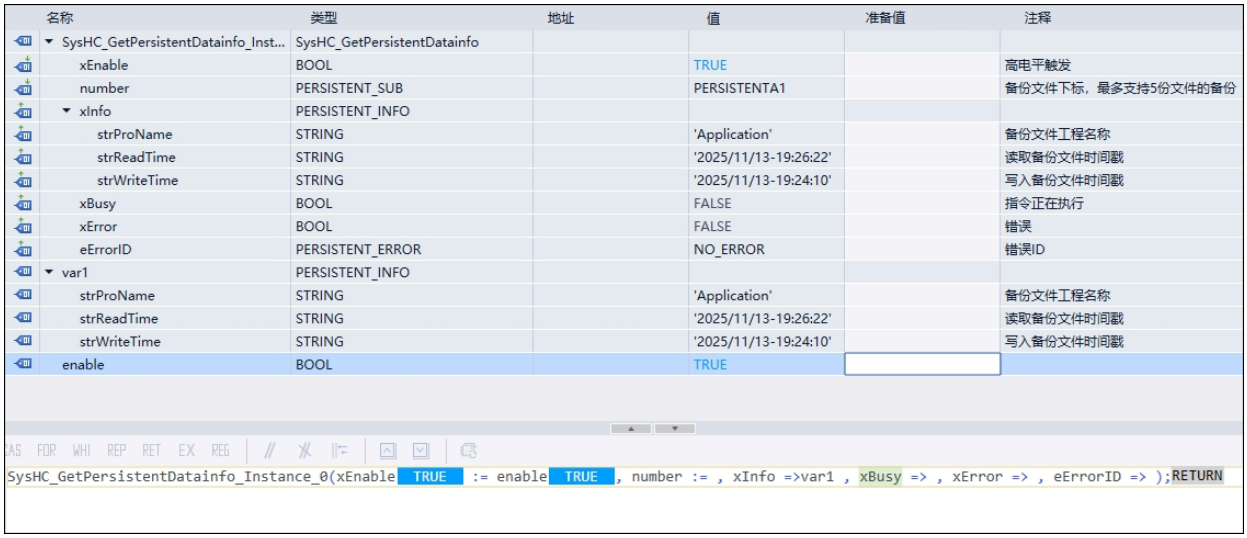
NOTE

For strProName read as an unknown string (possibly as Chinese), you can use the ConvertUTF8toUTF16() conversion function to convert the read information into Chinese. The Chinese structure type is WSTRING.

- **Function description**
  This instruction enables user programs to read information from retentive backup files.
- **Fault scenarios**
  - The PLC version does not support data backup at power failure.
  - The backup and restore function blocks are repeated within 10 seconds.
  - The backup and restore function blocks run simultaneously.
  - The backup file does not match the running program of the user.
  - The background version does not support the file backup function.
  - The backup file fails to be restored.

- **Program example**

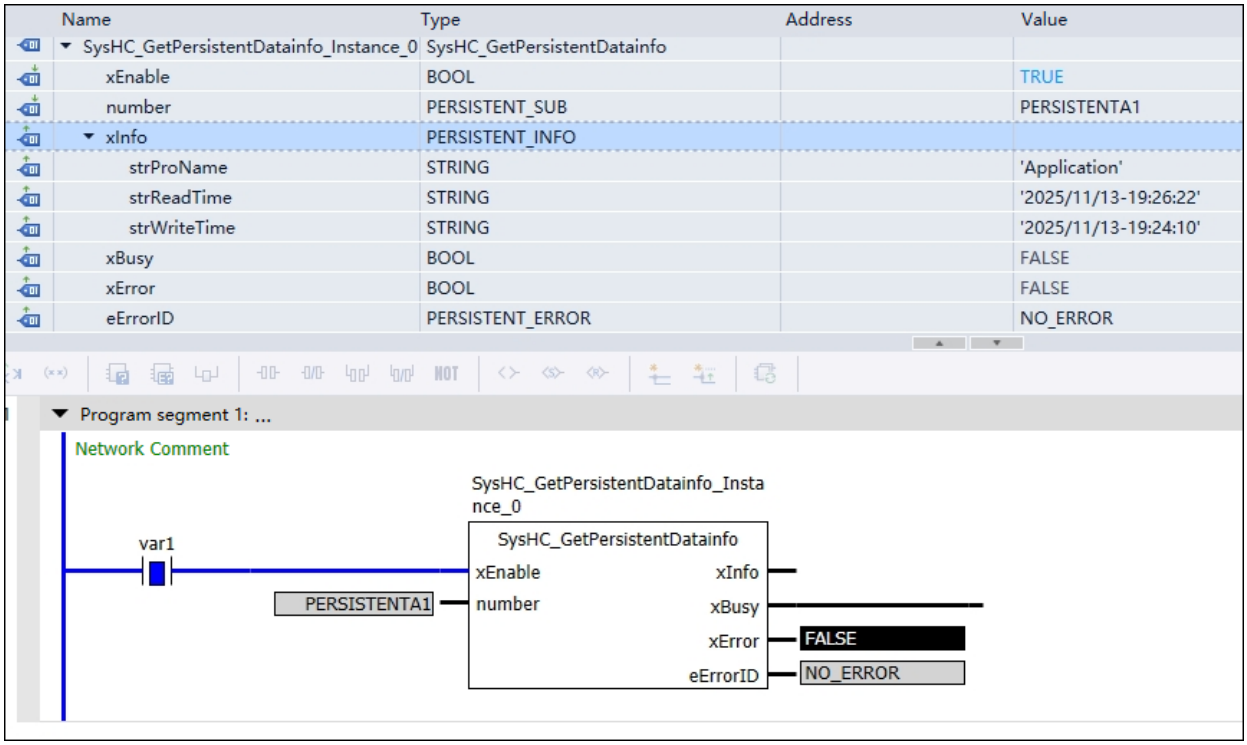
ST:



名称	类型	地址	值	准备值	注释
SysHC_GetPersistentDatainfo_Inst...	SysHC_GetPersistentDatainfo				
xEnable	BOOL		TRUE		高电平触发
number	PERSISTENT_SUB		PERSISTENTA1		备份文件下标，最多支持5份文件的备份
xInfo	PERSISTENT_INFO				
strProName	STRING		'Application'		备份文件工程名称
strReadTime	STRING		'2025/11/13-19:26:22'		读取备份文件时间戳
strWriteTime	STRING		'2025/11/13-19:24:10'		写入备份文件时间戳
xBusy	BOOL		FALSE		指令正在执行
xError	BOOL		FALSE		错误
eErrorID	PERSISTENT_ERROR		NO_ERROR		错误ID
var1	PERSISTENT_INFO				
strProName	STRING		'Application'		备份文件工程名称
strReadTime	STRING		'2025/11/13-19:26:22'		读取备份文件时间戳
strWriteTime	STRING		'2025/11/13-19:24:10'		写入备份文件时间戳
enable	BOOL		TRUE		

```
SysHC_GetPersistentDatainfo_Instance_0(xEnable := enable, number := , xInfo =>var1 , xBusy => , xError => , eErrorID => );RETURN
```

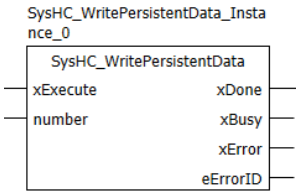
LD:



18.4.1.11 SysHC_WritePersistentData

This instruction enables the user program to quickly back up retentive data.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SysHC_WritePersistentData	Save backup file	FB		SysHC_WritePersistentData_Instance_0(xExecute := , number := , xDone => , xBusy => , xError => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Active on rising edge	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered.
	number	Backup file index	PERSISTENT_SUB	1 to 5	1	Backup file sub-index, supporting backup of up to five files PERSISTENT_1: File 1. PERSISTENT_2: File 2. PERSISTENT_3: File 3. PERSISTENT_4: File 4. PERSISTENT_5: File 5.
Output	xDone	Output active	BOOL	TRUE, FALSE	FALSE	Output active.
	xBusy	Instruction executing	BOOL	TRUE, FALSE	FALSE	Executing
	xError	Fault	BOOL	TRUE, FALSE	FALSE	Error.
	eErrorID	Fault ID	PERSISTENT_ERROR	-	NO_ERROR	See Fault Codes.

NOTE

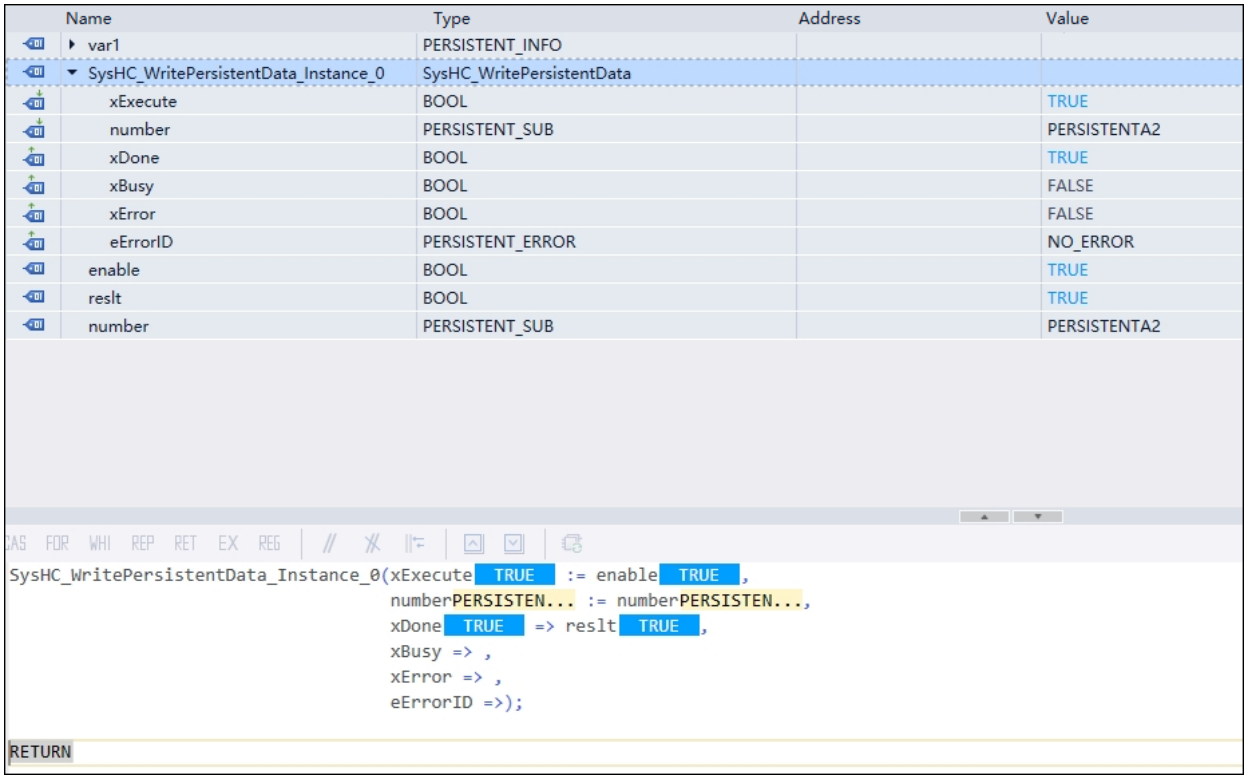
For strProName read as an unknown string (possibly as Chinese), you can use the ConvertUTF8toUTF16() conversion function to convert the read information into Chinese. The Chinese structure type is WSTRING.

- **Function description**

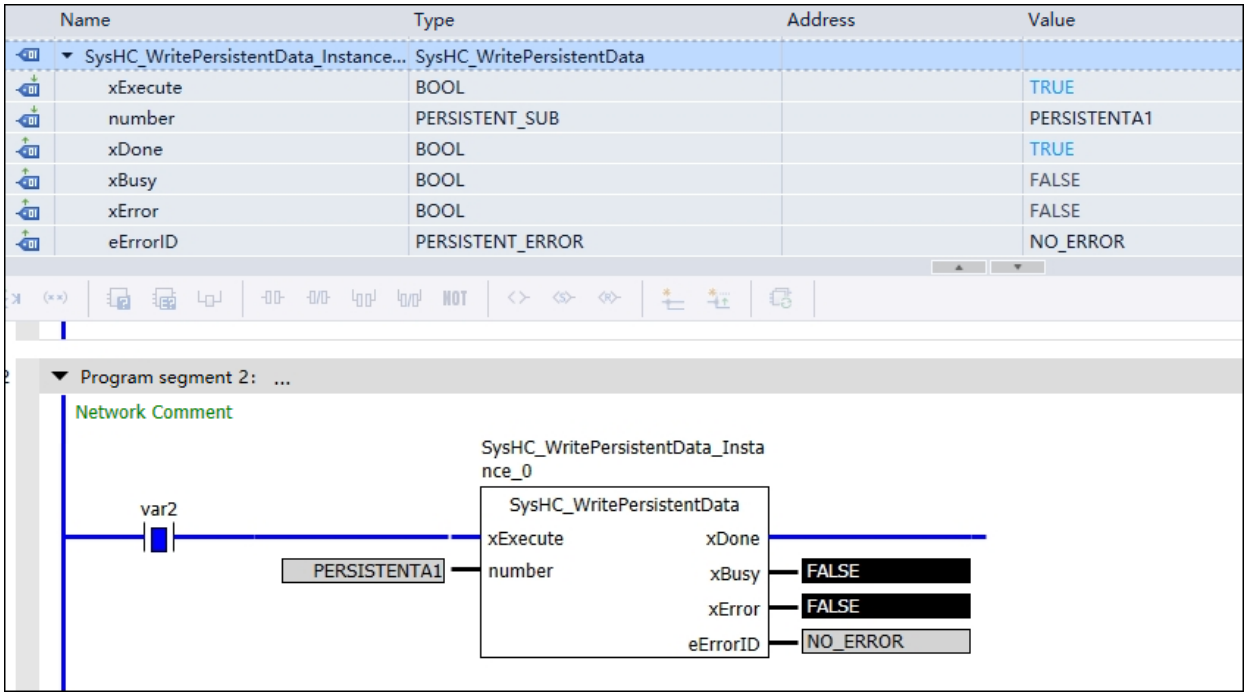
  This instruction enables the user program to quickly backup retentive data. After completing this operation, you can transfer the file to an external storage device. For detailed instructions, see 10.11.8 Backing Up the Program to an External Storage Device.

- **Fault scenarios**
  - The PLC version does not support data backup at power failure.
  - The backup and restore function blocks are repeated within 10 seconds.
  - The backup and restore function blocks run simultaneously.
  - The backup file does not match the running program of the user.
  - The background version does not support the file backup function.
  - The backup file fails to be restored.
- **Program example**

  ST:



  LD:

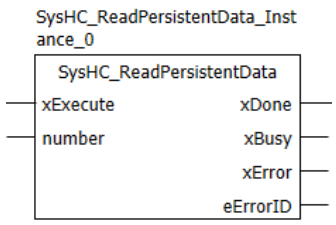


- **Precautions**
  - It is not recommended to use the current backup, restore, and backup information reading function blocks in tasks with high time requirements.
  - It is recommended to use the backup file information reading function block in tasks with a larger cycle and lower priority.
  - When the CPU utilization is high, operating the backup and restore function blocks will cause jitter in task execution.
  - The backup and restore function blocks cannot be called repeatedly within 60 s.
  - In the obtained function block information, the name of the file read is the project name of the power failure data written to the backup.
  - Performing backup at runtime is to save data in the running backup area.
  - When the restore function block is used, data in the backup area during running changes greatly.
  - When file backup is not completed, the file reading function block does not take effect. When file reading is not completed, the file backup function does not take effect.
  - The enable variable of the triggered function block cannot be the persistent variable.
  - It is not recommended to frequently call the backup and restore function blocks.

18.4.1.12 SysHC_ReadPersistentData

This instruction enables the user program to quickly restore retentive data.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SysHC_ReadPersistentData	Read backup file	FB		SysHC_ReadPersistentData_Instance_0(xExecute := , number := , xDone => , xBusy => , xError => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Active on rising edge	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered.
	number	Backup file index	PERSISTENT_SUB	1 to 5	PERSISTENT_1	Backup file sub-index, supporting backup of up to five files. PERSISTENT_1: File 1. PERSISTENT_2: File 2. PERSISTENT_3: File 3. PERSISTENT_4: File 4. PERSISTENT_5: File 5.
Output	xDone	Output active	BOOL	TRUE, FALSE	FALSE	Output active.
	xBusy	Instruction executing	BOOL	TRUE, FALSE	FALSE	Executing
	xError	Fault	BOOL	TRUE, FALSE	FALSE	Error.
	eErrorID	Fault ID	PERSISTENT_ERROR	-	NO_ERROR	See Fault Codes.

NOTE

For strProName read as an unknown string (possibly as Chinese), you can use the ConvertUTF8toUTF16() conversion function to convert the read information into Chinese. The Chinese structure type is WSTRING.

- **Function description**
  This instruction enables the user program to quickly restore retentive data.
- **Fault scenarios**
  - The PLC version does not support data backup at power failure.
  - The backup and restore function blocks are repeated within 10 seconds.

- The backup and restore function blocks run simultaneously.
- The backup file does not match the running program of the user.
- The background version does not support the file backup function.
- The backup file fails to be restored.

- **Program example**

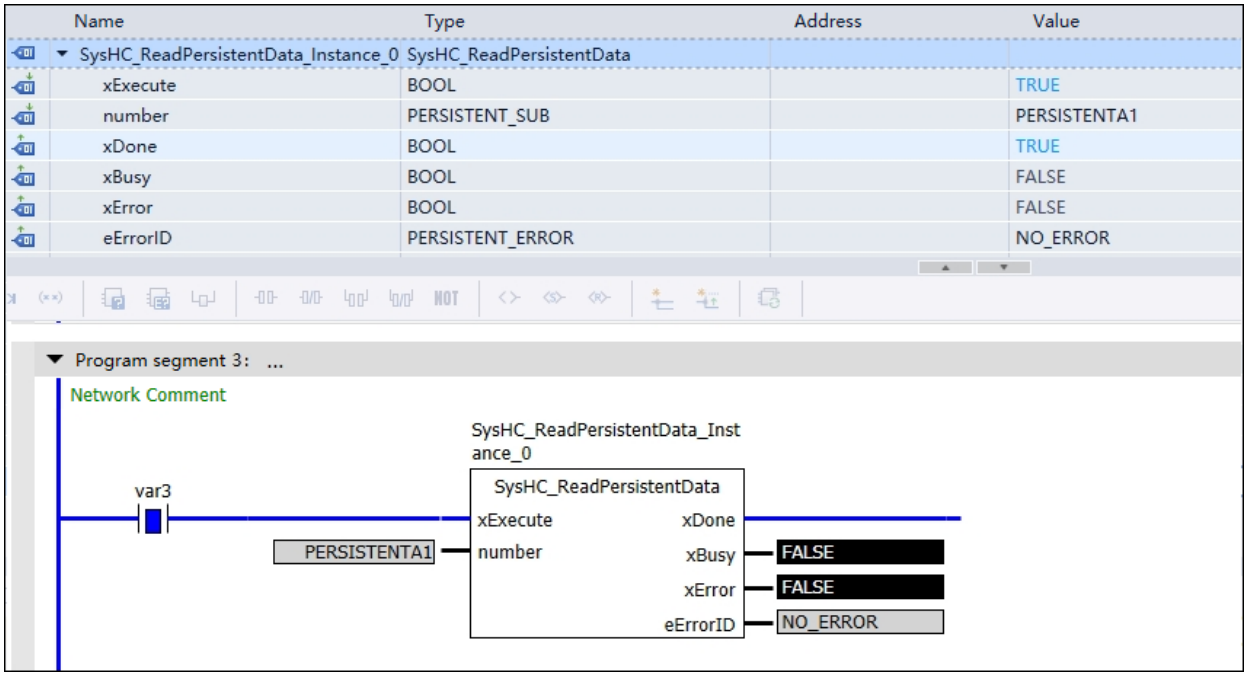
ST:

Name	Type	Address	Value
enable	BOOL		TRUE
reslt	BOOL		FALSE
number	PERSISTENT_SUB		PERSISTENTA2
SysHC_ReadPersistentData_Instance...	SysHC_ReadPersistentData		
xExecute	BOOL		TRUE
number	PERSISTENT_SUB		PERSISTENTA2
xDone	BOOL		FALSE
xBusy	BOOL		FALSE
xError	BOOL		TRUE
eErrorID	PERSISTENT_ERROR		PERSIS_NO_BAK

```
SysHC_ReadPersistentData_Instance_0(xExecute TRUE := enable TRUE ,
                                    numberPERSISTEN... := numberPERSISTEN...,
                                    xDone FALSE =>reslt FALSE ,
                                    xBusy => ,
                                    xError => ,
                                    eErrorID =>);

RETURN
```

LD:

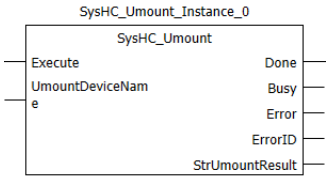


- **Precautions**
  - It is not recommended to use the current backup, restore, and backup information reading function blocks in tasks with high time requirements.
  - It is recommended to use the backup file information reading function block in tasks with a larger cycle and lower priority.
  - When the CPU utilization is high, operating the backup and restore function blocks will cause jitter in task execution.
  - The backup and restore function blocks cannot be called repeatedly within 60 s.
  - In the obtained function block information, the name of the file read is the project name of the power failure data written to the backup.
  - Performing backup at runtime is to save data in the running backup area.
  - When the restore function block is used, data in the backup area during running changes greatly.
  - When file backup is not completed, the file reading function block does not take effect. When file reading is not completed, the file backup function does not take effect.
  - The enable variable of the triggered function block cannot be the persistent variable.
  - It is not recommended to frequently call the backup and restore function blocks.

18.4.1.13 SysHC_Umount

This instruction unmounts external storage devices. It uses an external library to ensure that you can perform a safe unmount before manually ejecting the SD card or USB device.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SysHC_Umount	Unmount an external storage device	FB		SysHC_Umount_Instanc e_0(Execute := , UmountDeviceName := , Done => , Busy => , Error => , ErrorID => , StrUmountResult =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Function block enable	BOOL	TRUE, FALSE	FALSE	The instruction is triggered on a rising edge.
	UmountDevice Name	External device name to unmount	STRING[32]	-	'SD'	External device name to unmount, which can be up to 32 bytes. SD and USB devices are supported. • SD parameter: SD. • USB parameter: Fill in according to SysHC_UDiskPath query, for example sdb1.
Output	Done	Output active	BOOL	TRUE, FALSE	FALSE	Output active.
	Busy	Instruction executing	BOOL	TRUE, FALSE	FALSE	Instruction is executing.
	Error	Fault	BOOL	TRUE, FALSE	FALSE	Error.
	ErrorID	Error ID	SYS_HC_ER ROR	-	NO_ERR OR	See Fault Codes. When the ErrorID is UNKNOWN_ERROR, strUmountResult outputs the content of the unmount failure.
	StrUmountRes ult	Output the unmount result	STRING[32]	-	-	Output the unmount result.

- **Function description**

  To unmount an external USB storage device (USB flash disk), first call the SysHC_UDiskPath instruction to

obtain the file path of the connected USB storage device to the PLC, and then assign the last field of the file path (e.g., sdb1) to UmountDeviceName.

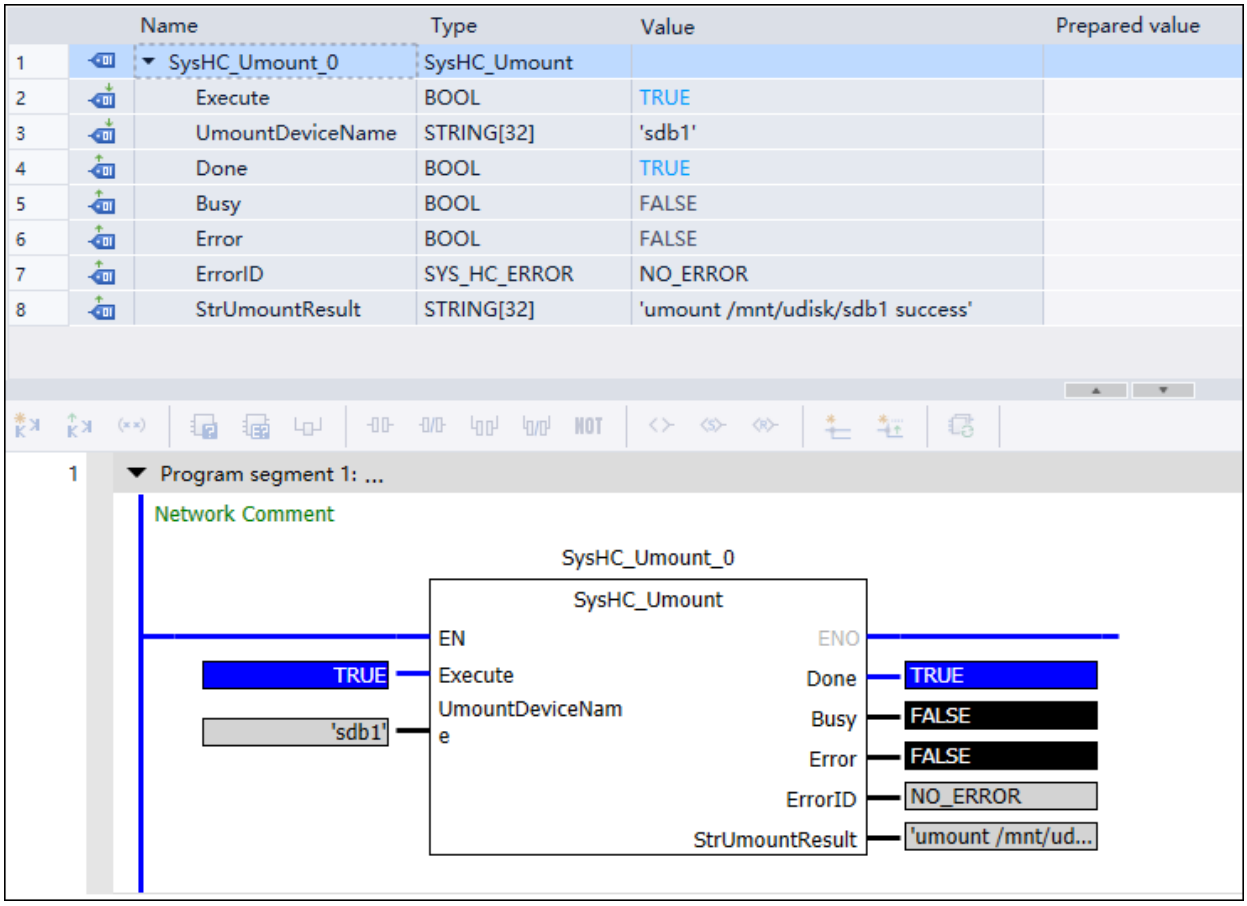
- **Program example**

  ST:

	Name	Type	Add...	Value
1	SysHC_Umount_Instance_0	SysHC_Umount		
2	Execute	BOOL		TRUE
3	UmountDeviceName	STRING[32]		'sdb1'
4	Done	BOOL		TRUE
5	Busy	BOOL		FALSE
6	Error	BOOL		FALSE
7	ErrorID	SYS_HC_ERROR		NO_ERROR
8	StrUmountResult	STRING[32]		'umount /mnt/udisk/sdb1 success'
9	test	BOOL		TRUE

```
1 SysHC_Umount_Instance_0(Execute TRUE := test TRUE ,
2                         UmountDeviceName := ,
3                         Done => ,
4                         Busy => ,
5                         Error => ,
6                         ErrorID => ,
7                         StrUmountResult => );RETURN
```

  LD:



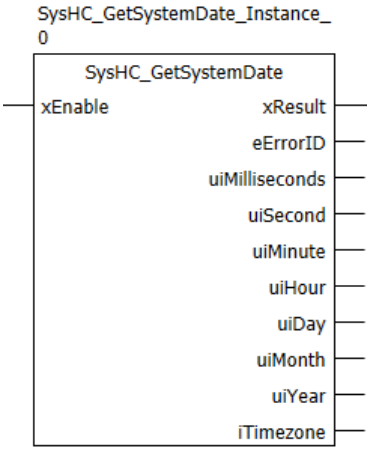
- **Note**

  After successful execution of the instruction, manually eject the SD card or the corresponding USB external device to be unmounted.

18.4.1.14 SysHC_GetSystemDate

This instruction gets the system date.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SysHC_GetSystemDate	Get system date	FB		SysHC_GetSystemDate_Instance_0(xEnable := , xResult => , eErrorID => , uiMilliseconds => , uiSecond => , uiMinute => , uiHour => , uiDay => , uiMonth => , uiYear => , iTimezone =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Function block enable	BOOL	TRUE, FALSE	FALSE	Level triggered
Output	XResult	Execution result	BOOL	TRUE, FALSE	FALSE	Execution result.
	eErrorID	Fault ID	SYS_HC_ERROR	Follows data type	-	Fault ID.
	uiMilliseconds	Milliseconds	UINT	0 to 999	-	Milliseconds (ms)
	uiSecond	Second	UINT	0 to 59	-	Seconds (s)
	uiMinute	Minute	UINT	0 to 59	-	Minutes (min)
	uiHour	Hour	UINT	0 to 23	-	Hours (h)
	uiDay	Day	UINT	1 to 31	-	Day
	uiMonth	Month	UINT	1 to 12	-	Month
	uiYear	Year	UINT	2000 to 2038	-	Year
	iTimezone	Time zone	INT	-12 to +14	-	Time zone.

- **Function description**

  This function block is mainly used to get the system date.

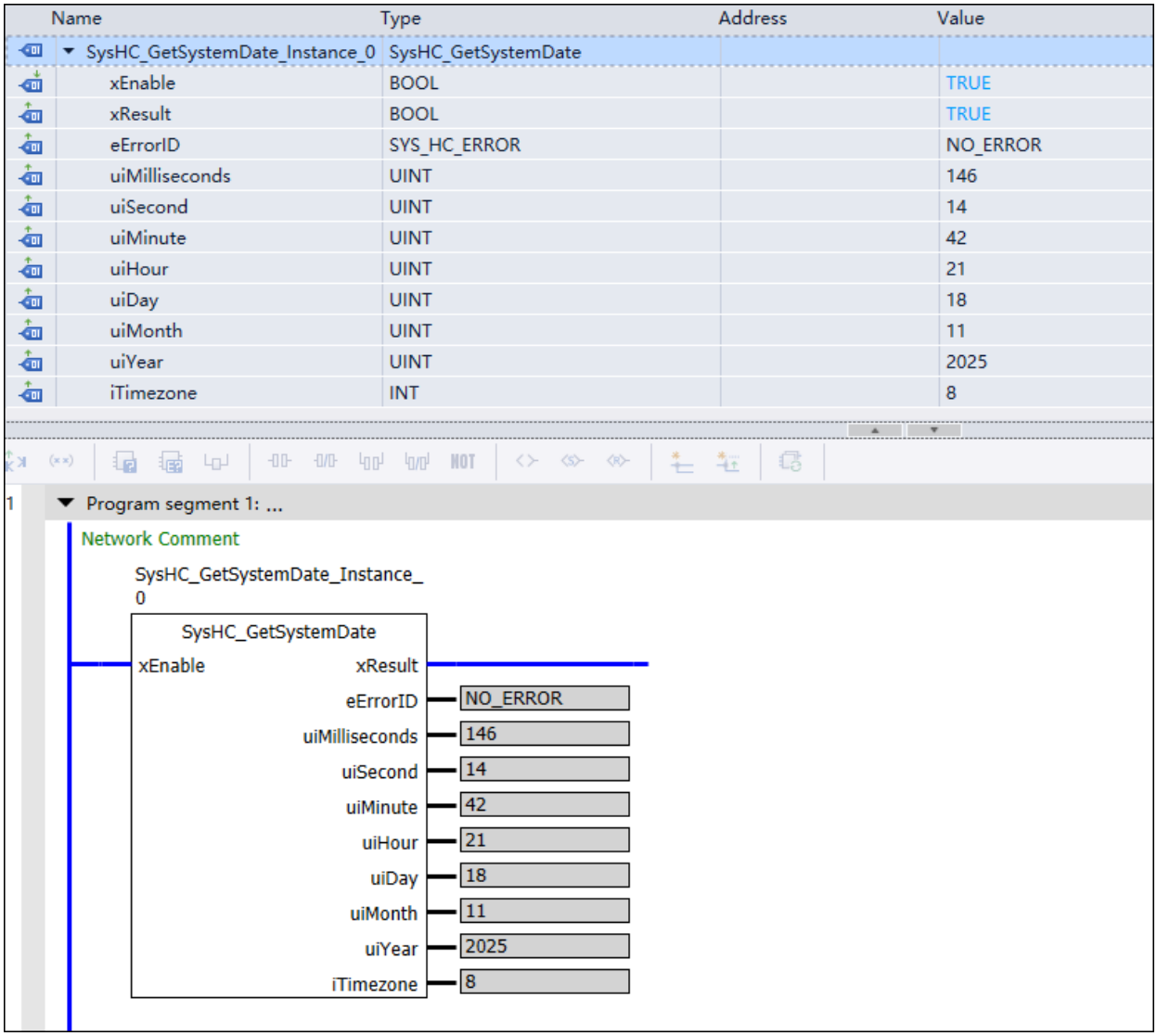
- **Program example**

  ST:

Name	Type	Address	Value
SysHC_GetSystemDate_Instan...	SysHC_GetSystemDate		
Enable	BOOL		TRUE
Result	BOOL		TRUE
ErrorID	SYS_HC_ERROR		NO_ERROR
uiMilliseconds	UINT		966
uiSecond	UINT		48
uiMinute	UINT		41
uiHour	UINT		21
uiDay	UINT		18
uiMonth	UINT		11
uiYear	UINT		2025
iTimezone	INT		8

```
SysHC_GetSystemDate_Instance_0(
xEnable TRUE := Enable TRUE,
xResult TRUE => Result TRUE,
eErrorID NO_ERROR => ErrorID NO_ERROR,
uiMilliseconds 966 => uiMilliseconds 966,
uiSecond 48 => uiSecond 48,
uiMinute 41 => uiMinute 41,
uiHour 21 => uiHour 21,
uiDay 18 => uiDay 18,
uiMonth 11 => uiMonth 11,
uiYear 2025 => uiYear 2025,
iTimezone 8 => iTimezone 8);
```

  LD:



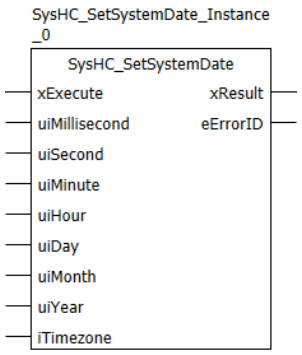
- **Note**

  SysHC_GetSystemDate is an asynchronous instruction. Frequent enabling within a single task cycle may cause data acquisition error or induce task jitter.

18.4.1.15 SysHC_SetSystemDate

This instruction sets the system date.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SysHC_SetSystemDate	Set system date	FB		SysHC_SetSystemDate_Instance_0(xExecute := , uiMillisecond := , uiSecond := , uiMinute := , uiHour := , uiDay := , uiMonth := , uiYear := , iTimezone := , xResult => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Function block enable	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	uiMillisecond	Milliseconds	UINT	0 to 999	-	Milliseconds (ms)
	uiSecond	Second	UINT	0 to 59	-	Seconds (s)
	uiMinute	Minute	UINT	0 to 59	-	Minutes (min)
	uiHour	Hour	UINT	0 to 23	-	Hours (h)
	uiDay	Day	UINT	1 to 31	-	Day
	uiMonth	Month	UINT	1 to 12	-	Month
	uiYear	Year	UINT	2000 to 2038	-	Year
	iTimezone	Time zone	INT	-12 to +14	-	Time zone (reserved and cannot be set).
Output	xResult	Execution result	BOOL	TRUE, FALSE	FALSE	Execution result.
	eErrorID	Fault ID	SYS_HC_ERROR	Follows data type	-	See Fault Codes.

- **Function description**

  This function block is mainly used to set the system date.

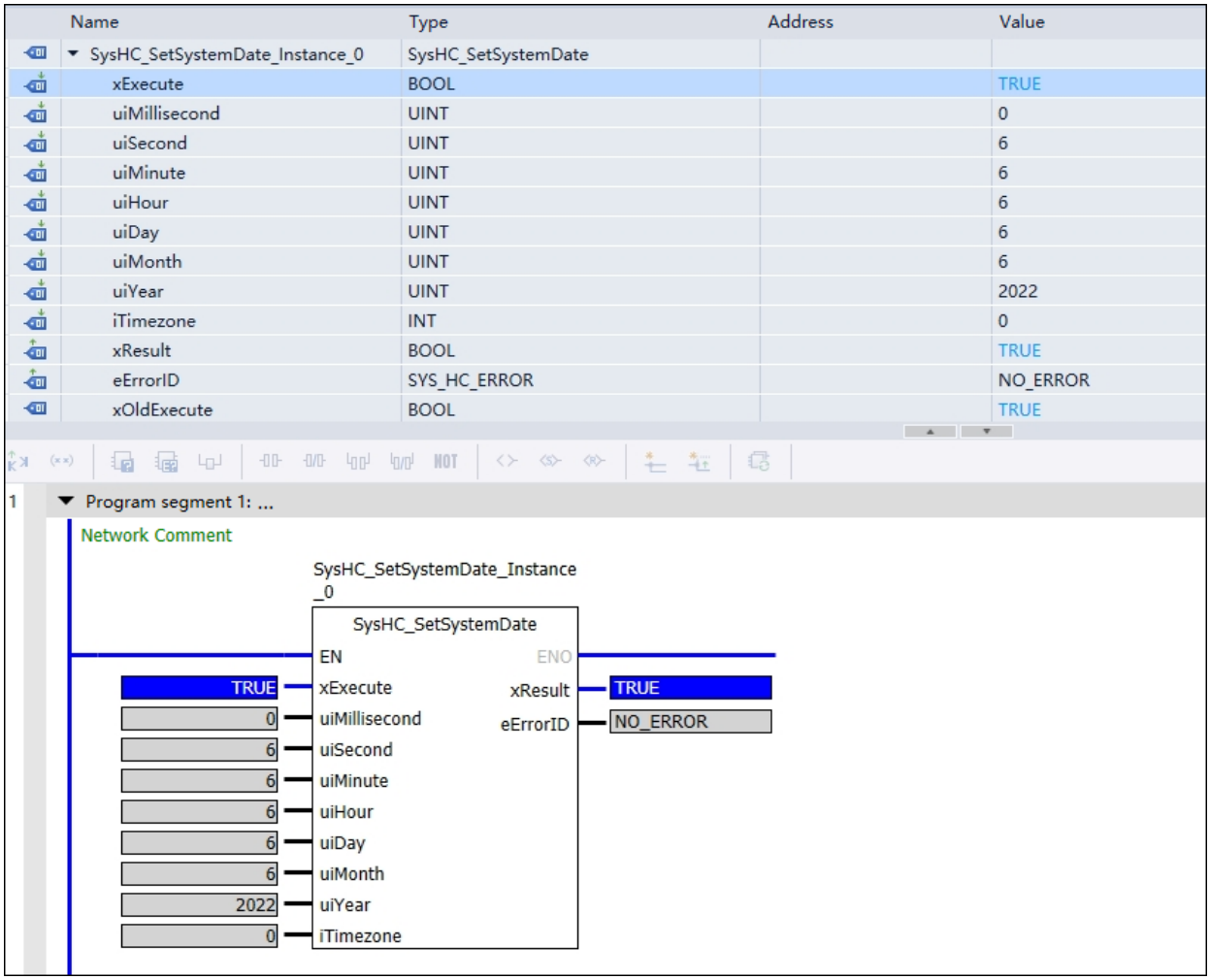
- **Program example**

  ST:

Name	Type	Address	Value
SysHC_SetSystemDate_Instanc...	SysHC_SetSystemDate		
Execute	BOOL		TRUE
uiMilliseconds	UINT		0
uiSecond	UINT		2
uiMinute	UINT		2
uiHour	UINT		2
uiDay	UINT		2
uiMonth	UINT		2
uiYear	UINT		2025
iTimezone	INT		0
Result	BOOL		FALSE
ErrorID	SYS_HC_ERROR		NO_ERROR

```
SysHC_SetSystemDate_Instance_0(
xExecute TRUE := Execute TRUE,
uiMillisecond 0 := uiMilliseconds 0,
uiSecond 2 := uiSecond 2,
uiMinute 2 := uiMinute 2,
uiHour 2 := uiMinute 2,
uiDay 2 := uiDay 2,
uiMonth 2 := uiMonth 2,
uiYear 2025 := uiYear 2025,
iTimezone 0 := iTimezone 0,
xResult FALSE => Result FALSE,
eErrorID NO_ERROR => ErrorID NO_ERROR );
```

LD:

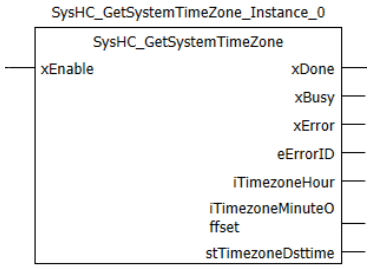


- **Precautions**
  - Set the system time of EVO400/EVO500 via SysHC_SetSystemDate. If no external RTC expansion card is installed, the instruction reports "UNKNOWN_ERROR", signifying a failure to write time data to the RTC chip. Nevertheless, the local system time of the device will be updated normally. Without an RTC module, the device time may reset to January 1, 1970 after a power loss.
  - SysHC_SetSystemDate is an asynchronous instruction. Frequent enabling within a single task cycle may cause data writing error or induce task jitter.

18.4.1.16 SysHC_GetSystemTimeZone

This instruction gets the system timezone information.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SysHC_GetSystemTimeZone	Get system time zone	FB		SysHC_GetSystemTimeZone_Instance_0(xEnable := , xDone => , xBusy => , xError => , eErrorID => , iTimezoneHour => , iTimezoneMinuteOffset => , stTimezoneDsttime =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Enable	BOOL	TRUE, FALSE	FALSE	Function enable, level-triggered
Output	xDone	FB completion flag	BOOL	TRUE, FALSE	-	Function block completion flag.
	xBusy	Function block execution flag	BOOL	TRUE, FALSE	-	Function block execution flag
	xError	Function block fault flag	BOOL	TRUE, FALSE	-	Function block fault flag
	eErrorID	Function block fault ID	SYS_HC_ERROR	Follows data type	-	See Fault Codes.
	iTimezoneHour	Time zone information	INT	-12 to +14	8	Integer time zone offset in hours.
	iTimezoneMinuteOffset	Timezone information	INT	0 to 59	0	Non-integer time zone offset in minutes.
	stTimezoneDsttime	DST configuration	TIMEZONE_INFO	-	-	DST structure.

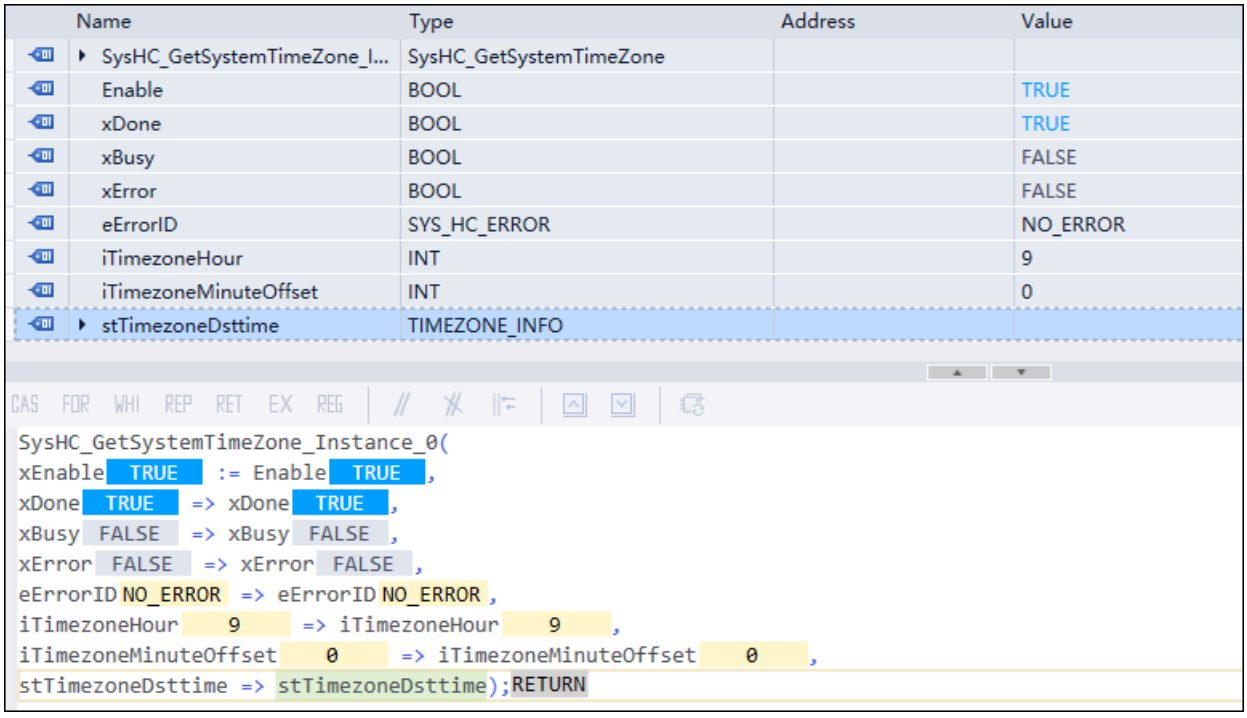
- **Function**
  - This function block is mainly used to get the system time zone information.
  - Description of internal members of the TIMEZONE_INFO data structure:

Name of Structure Member	Name	Data Type	Valid Range	Default Value	Description
iDstTimeZoneOffset	DST offset	INT	0~120	-	DST offset in minutes.
iDstStartWeekth	DST start time	INT	0 to 4	-	DST start time: week (0: 1st week)

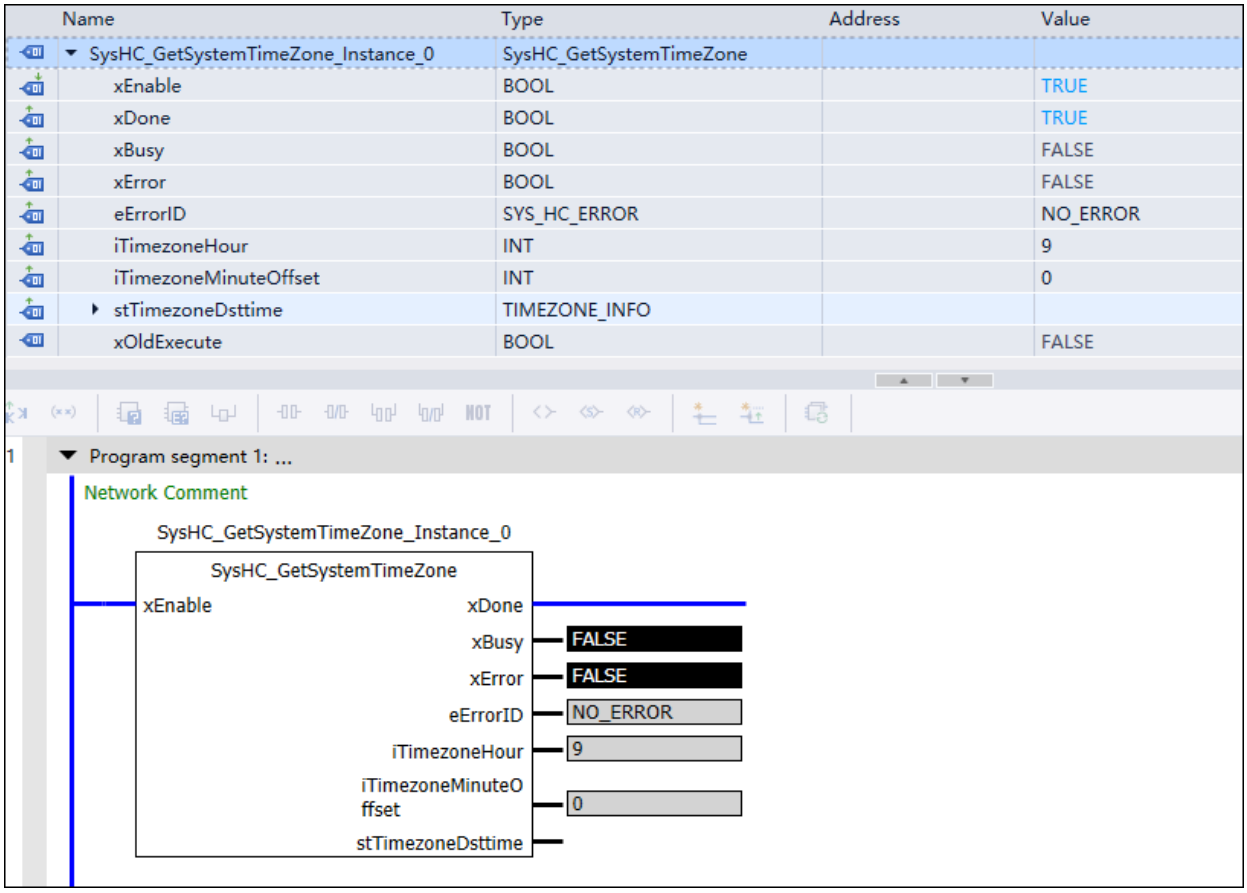
Name of Structure Member	Name	Data Type	Valid Range	Default Value	Description
iDstStartWeekday	DST start time	INT	0 to 6	-	DST start time: day of the week (0: Sunday)
iDstStartMonth	DST start time	INT	0 to 11	-	DST start time: month (0: January)
iDstStartHour	DST start time	INT	0 to 23	-	DST start time: hour
iDstEndWeekth	DST end time	INT	0 to 4	-	DST end time: week (0: 1st week)
iDstEndWeekday	DST end time	INT	0 to 6	-	DST end time: day of the week (0: Sunday)
iDstEndMonth	DST end time	INT	0 to 11	-	DST end time: month (0: January)
iDstEndHour	DST end time	INT	0 to 23	-	DST end time: hour

- **Program example**

ST:



LD:

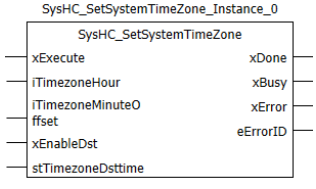


- **Note**
  The SysHCPlcInfo 1.3.0.0 library version does not support DST time zone setting. The DST function is unavailable.

18.4.1.17 SysHC_SetSystemTimeZone

Set system time zone information.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SysHC_SetSystemTimeZone	Set system timezone	FB		SysHC_SetSystemTimeZone_Instance_0(xExecute := , iTimezoneHour := , iTimezoneMinuteOffset := , xEnableDst := , stTimezoneDsttime := , xDone => , xBusy => , xError => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Enable	BOOL	TRUE, FALSE	FALSE	Triggered on the rising edge
	iTimezoneHour	Time zone information	INT	-12 to +14	8	Integer time zone offset in hours.
	iTimezoneMinuteOffset	Timezone information	INT	0 to 59	0	Non-integer time zone offset in minutes.
	xEnableDst	Enable DST setting	BOOL	TRUE, FALSE	FALSE	Enable daylight saving time.
	stTimezoneDsttime	DST configuration	TIMEZONE_INFO	-	-	DST structure.
Output	xDone	FB completion flag	BOOL	TRUE, FALSE	-	Function block completion flag.
	xBusy	Function block execution flag	BOOL	TRUE, FALSE	-	Function block execution flag
	xError	Function block fault flag	BOOL	TRUE, FALSE	-	Function block fault flag
	eErrorID	Function block fault ID	SYS_HC_ERROR	Follows data type	-	See Fault Codes.

- **Function**
  - This function block is mainly used for system time zone settings, including integer time zone offset, time zone minute offset, and daylight saving time.
  - Description of internal members of the TIMEZONE_INFO data structure:

Name of Structure Member	Name	Data Type	Valid Range	Default Value	Description
iDstTimeZoneOffset	DST offset	INT	0~120	-	DST offset in minutes.
iDstStartWeekth	DST start time	INT	0 to 4	-	DST start time: week (0: 1st week)
iDstStartWeekday	DST start time	INT	0 to 6	-	DST start time: day of the week (0: Sunday)
iDstStartMonth	DST start time	INT	0 to 11	-	DST start time: month (0: January)
iDstStartHour	DST start time	INT	0 to 23	-	DST start time: hour
iDstEndWeekth	DST end time	INT	0 to 4	-	DST end time: week (0: 1st week)

Name of Structure Member	Name	Data Type	Valid Range	Default Value	Description
iDstEndWeekday	DST end time	INT	0 to 6	-	DST end time: day of the week (0: Sunday)
iDstEndMonth	DST end time	INT	0 to 11	-	DST end time: month (0: January)
iDstEndHour	DST end time	INT	0 to 23	-	DST end time: hour

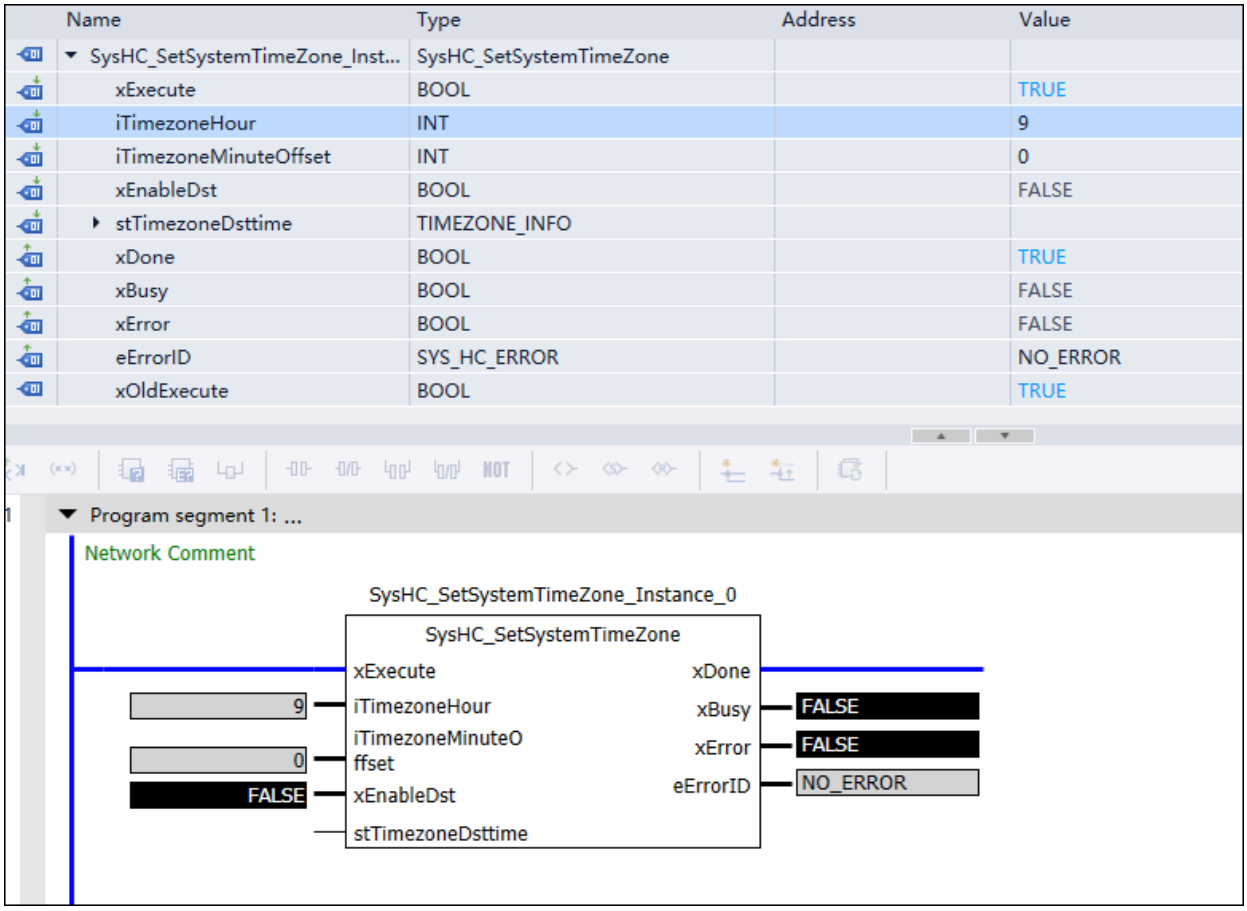
- **Program example**

ST:

Name	Type	Address	Value
SysHC_SetSystemTimeZone_In...	SysHC_SetSystemTimeZone		
xExecute	BOOL		TRUE
iTimezoneHour	INT		9
iTimezoneMinuteOffset	INT		0
xEnableDst	BOOL		FALSE
stTimezoneDsttime	TIMEZONE_INFO		
xDone	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
eErrorID	SYS_HC_ERROR		NO_ERROR

```
SysHC_SetSystemTimeZone_Instance_0(
xExecute TRUE := xExecute TRUE,
iTimezoneHour 9 := iTimezoneHour 9,
iTimezoneMinuteOffset 0 := iTimezoneMinuteOffset 0,
xEnableDst FALSE := xEnableDst FALSE,
stTimezoneDsttime := stTimezoneDsttime,
xDone TRUE => xDone TRUE,
xBusy FALSE => xBusy FALSE,
xError FALSE => xError FALSE,
eErrorID NO_ERROR => eErrorID NO_ERROR );RETURN
```

LD:

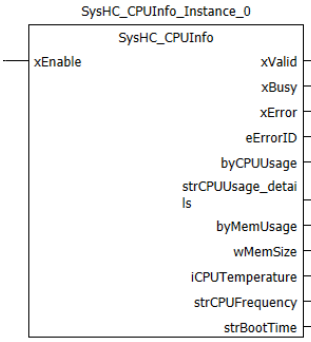


- **Precautions**
  - The interval between DST end and start times must not be less than the set DST offset. Consecutive or same months for DST start and end are not supported.
  - The SysHCPlcInfo 1.3.0.0 library version does not support DST time zone setting. The DST function is unavailable.

18.4.1.18 SysHC_CPUInfo

This instruction obtains the CPU information such as memory, frequency, and temperature.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SysHC_CPUInfo	Obtain CPU information such as memory, frequency, and temperature	FB		SysHC_CPUInfo_Instance_0(xEnable := , xValid => , xBusy => , xError => , eErrorID => , byCPUUsage => , strCPUUsage_details => , byMemUsage => , wMemSize => , iCPUTemperature => , strCPUFrequency => , strBootTime =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Active high	BOOL	TRUE, FALSE	FALSE	Active High.
Output	xValid	Output active	BOOL	TRUE, FALSE	FALSE	Output active.
	xBusy	Instruction executing	BOOL	TRUE, FALSE	FALSE	Executing
	xError	Fault	BOOL	TRUE, FALSE	FALSE	Error.
	eErrorID	Fault ID	SYS_HC_ERROR	-	NO_ERROR	See Fault Codes.
	byCPUUsage	CPU Load	BYTE	0 to 255	0	CPU load rate.
	strCPUUsage_details	Detailed CPU Load	STRING(128)	-	-	Separate CPU load rate.
	byMemUsage	Memory usage	BYTE	0 to 255	0	Memory usage.
	wMemSize	Memory size	WORD	0 to 65535	0	Memory size in MB.
	iCPUTemperature	CPU temperature	INT	0 to 255	0	CPU temperature in °C
	strCPUFrequency	CPU frequency	STRING[17]	-	-	CPU frequency in MHz.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	strBootTime	Startup time	STRING[17]	-	-	Startup time.

- **Function Description**

  This function block is mainly used to obtain CPU information such as memory, frequency, and temperature.

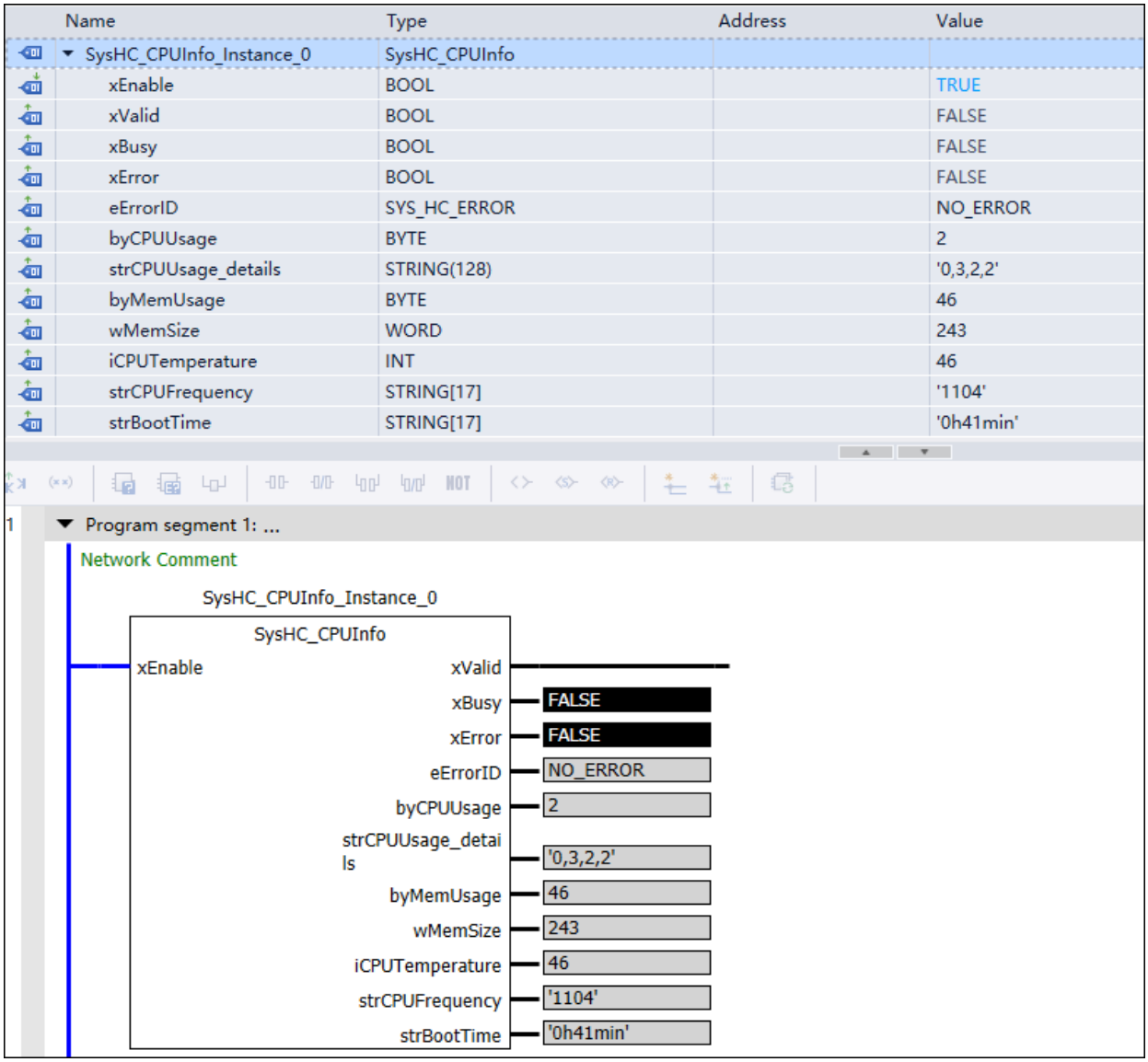
- **Program example**

  ST:

Name	Type	Address	Value
SysHC_CPUInfo_Instance_0	SysHC_CPUInfo		
xEnable	BOOL		TRUE
xValid	BOOL		FALSE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
eErrorID	SYS_HC_ERROR		NO_ERROR
byCPUUsage	BYTE		1
strCPUUsage_details	STRING(128)		'0,4,1,3'
byMemUsage	BYTE		46
wMemSize	WORD		243
iCPUTemperature	INT		46
strCPUFrequency	STRING[17]		'1104'
strBootTime	STRING[17]		'0h42min'

```
SysHC_CPUInfo_Instance_0(
xEnable TRUE := xEnable TRUE,
xValid FALSE => xValid FALSE,
xBusy FALSE => xBusy FALSE,
xError FALSE => xError FALSE,
eErrorID NO_ERROR => eErrorID NO_ERROR,
byCPUUsage 1 => byCPUUsage 1,
strCPUUsage_details '0,4,1,3' => strCPUUsage_details '0,4,1,3',
byMemUsage 46 => byMemUsage 46,
wMemSize 243 => wMemSize 243,
iCPUTemperature 46 => iCPUTemperature 46,
strCPUFrequency '1104' => strCPUFrequency '1104',
strBootTime '0h42min' => strBootTime '0h42min' );RETURN
```

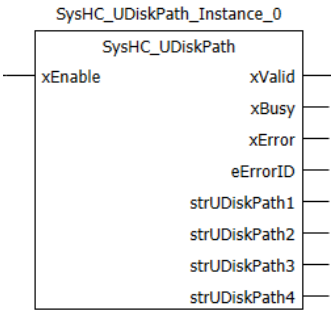
  LD:



18.4.1.19 SysHC_UDiskPath

This instruction gets the path information of USB flash disk.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SysHC_UDiskPath	Get the system path of the PLC USB interface	FB		SysHC_UDiskPath_Instance_0(xEnable := , xValid => , xBusy => , xError => , eErrorID => , strUDiskPath1 => , strUDiskPath2 => , strUDiskPath3 => , strUDiskPath4 =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Active high	BOOL	TRUE, FALSE	FALSE	Active High.
Output	xValid	Output active	BOOL	TRUE, FALSE	FALSE	Output active.
	xBusy	Instruction executing	BOOL	TRUE, FALSE	FALSE	Instruction is executing.
	xError	Fault	BOOL	TRUE, FALSE	FALSE	Error.
	eErrorID	Fault ID	SYS_HC_ERROR	-	NO_ERROR	See Fault Codes.
	strUDiskPath1	USB disk path 1	STRING[33]	-	-	USB disk path 1.
	strUDiskPath2	USB disk path 2	STRING[33]	-	-	USB disk path 2.
	strUDiskPath3	USB disk path 3	STRING[33]	-	-	USB disk path 3.
	strUDiskPath4	USB disk path 4	STRING[33]	-	-	USB disk path 4.

- **Function Description**

  This function block is used to obtain the system path of the PLC USB interface.

- **Program example**

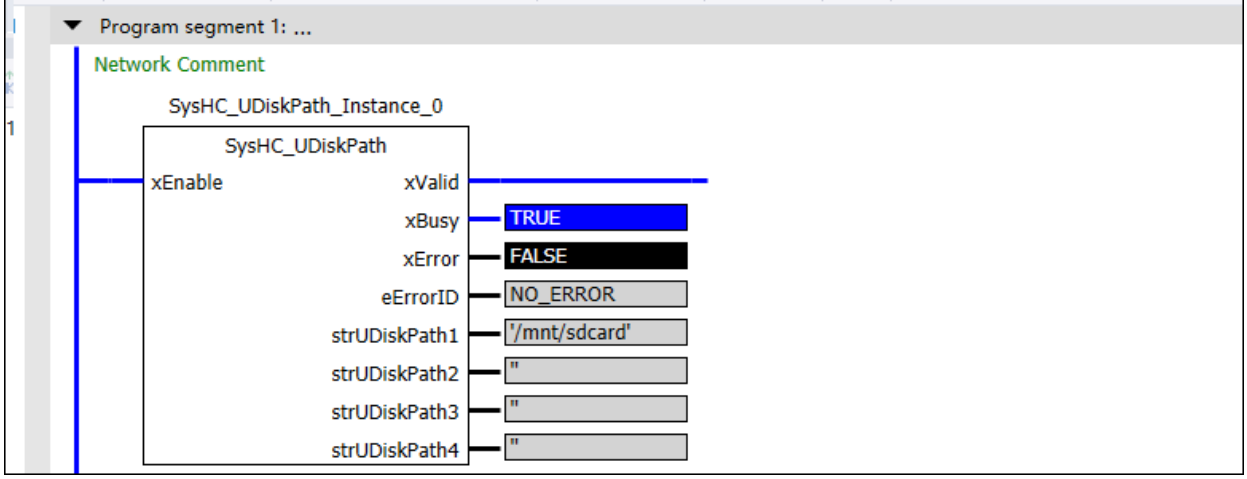
  ST:

Name	Type	Address	Value
SysHC_UDiskPath_Instance_0	SysHC_UDiskPath		
xEnable	BOOL		TRUE
xValid	BOOL		TRUE
xBusy	BOOL		TRUE
xError	BOOL		FALSE
eErrorID	SYS_HC_ERROR		NO_ERROR
strUDiskPath1	STRING[33]		'/mnt/sdcard'
strUDiskPath2	STRING[33]		''
strUDiskPath3	STRING[33]		''
strUDiskPath4	STRING[33]		''

```
SysHC_UDiskPath_Instance_0(
xEnable TRUE  := xEnable TRUE ,
xValid TRUE  => xValid TRUE ,
xBusy TRUE  => xBusy TRUE ,
xError FALSE  => xError FALSE ,
eErrorID NO_ERROR  => eErrorID NO_ERROR ,
strUDiskPath1 '/mnt/sdcard'  => strUDiskPath1 '/mnt/sdcard' ,
strUDiskPath2 ''  => strUDiskPath2 '' ,
strUDiskPath3 ''  => strUDiskPath3 '' ,
strUDiskPath4 ''  => strUDiskPath4 '' );RETURN
```

LD:

Name	Type	Address	Value
SysHC_UDiskPath_Instance_0	SysHC_UDiskPath		
xEnable	BOOL		TRUE
xValid	BOOL		TRUE
xBusy	BOOL		TRUE
xError	BOOL		FALSE
eErrorID	SYS_HC_ERROR		NO_ERROR
strUDiskPath1	STRING[33]		'/mnt/sdcard'
strUDiskPath2	STRING[33]		''
strUDiskPath3	STRING[33]		''
strUDiskPath4	STRING[33]		''



18.4.1.20 GetBootVersion

This instruction gets the Boot version number.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
GetBootVersion	Get BIOS version	FC		:= GetBootVersion();

- **Related variables**

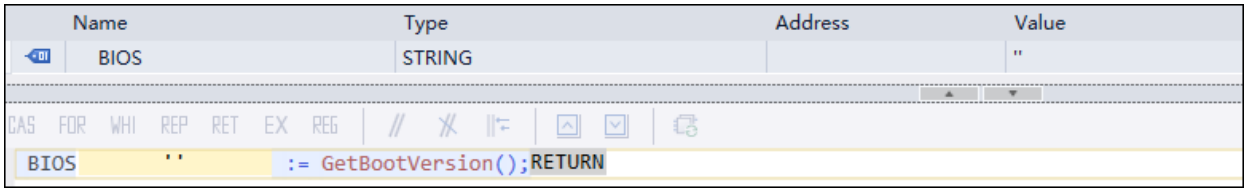
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	GetBootVersion	Boot version number	STRING	-	-	Boot version number

- **Function description**

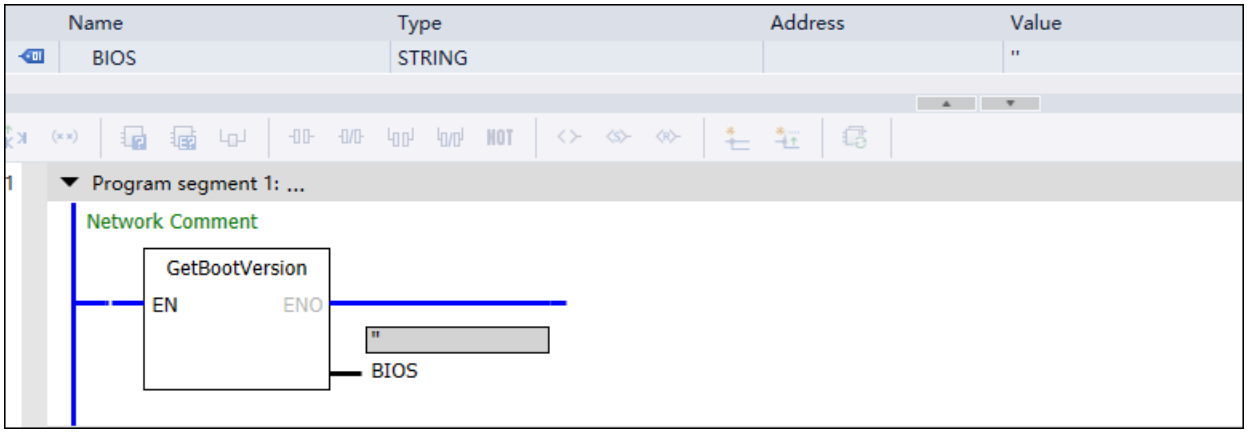
  This function block is mainly used to get the Boot version number.

- **Program example**

  ST:



  LD:



18.4.1.21 GetPLCVersion

This instruction gets the PLC version number.

- **Instruction format**

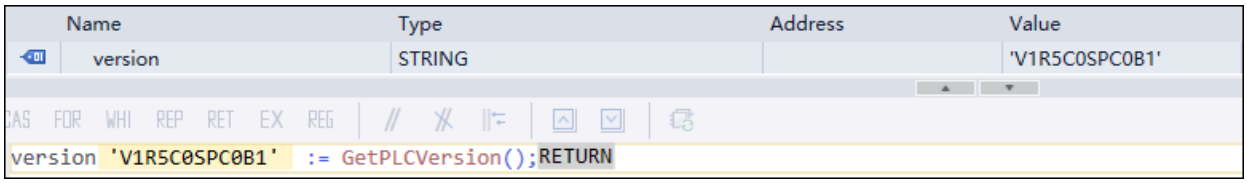
Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
GetPLCVersion	Get the PLC firmware version	FC		:= GetPLCVersion();

- **Related variables**

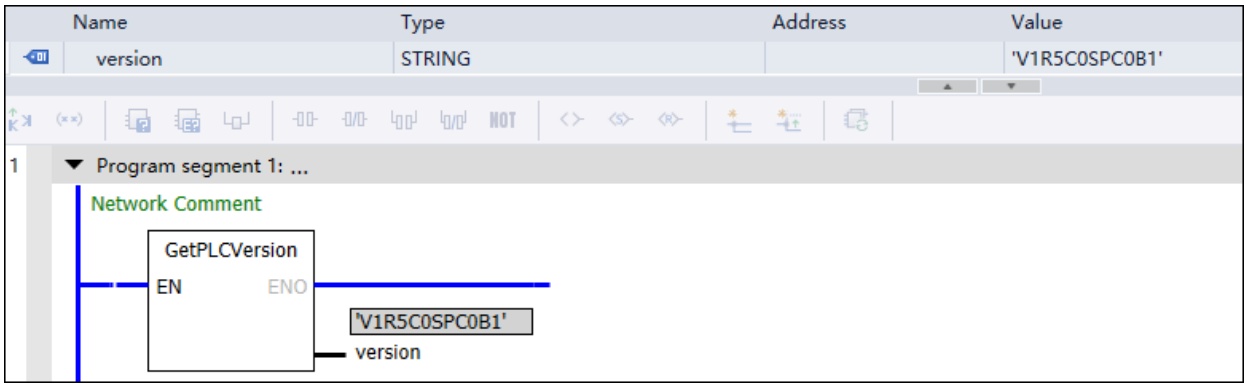
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	GetPLCVersion	Get PLC version number	STRING	-	-	This instruction gets the PLC version number.

- **Function description**
  This function block is mainly used to get the PLC version number.
- **Program example**

  ST:



  LD:



18.4.1.22 GetProductName

This instruction gets the PLC device name.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
GetProductName	Get PLC device name	FC	GetProductName EN ENO	:= GetProductName();

- **Related variables**

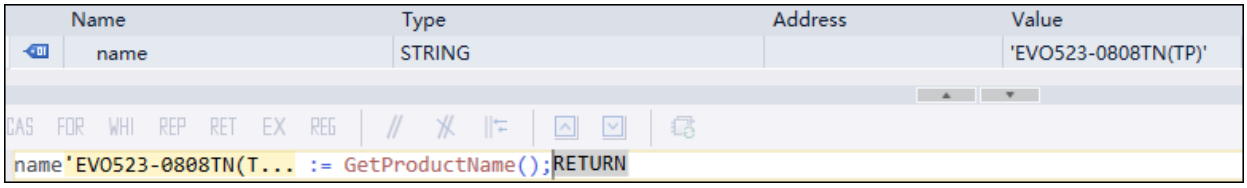
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	GetProductName	Get device name	STRING	-	-	This instruction gets the device name.

- **Function description**

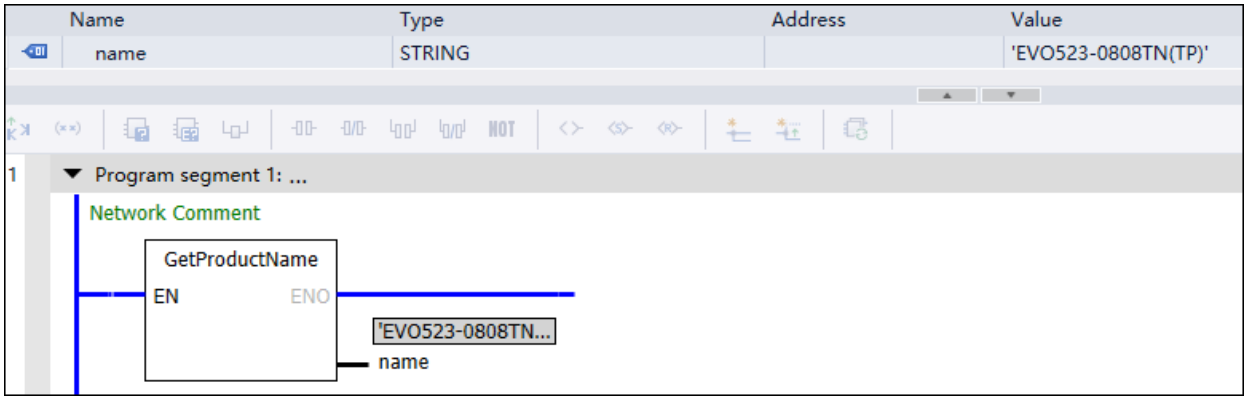
  This function block is mainly used to get the PLC device name.

- **Program example**

  ST:



  LD:



18.4.1.23 GetSerialNumber

This instruction gets the SN number.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
GetSerialNumber	Get the PLC serial number (unique)	FC	GetSerialNumber EN ENO	:= GetSerialNumber();

- **Related variables**

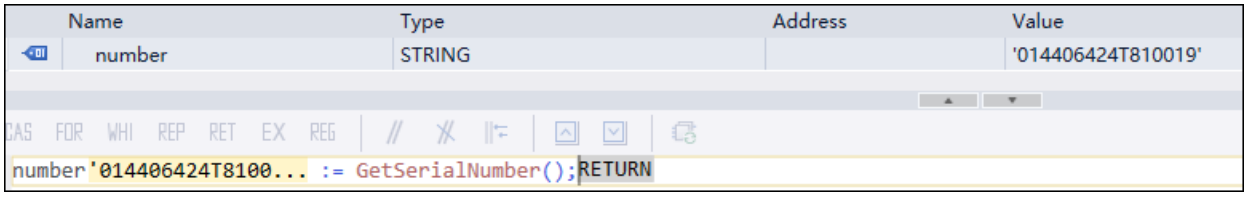
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	GetSerialNumber	Get SN number	STRING	-	-	This instruction gets the SN number.

- **Function description**

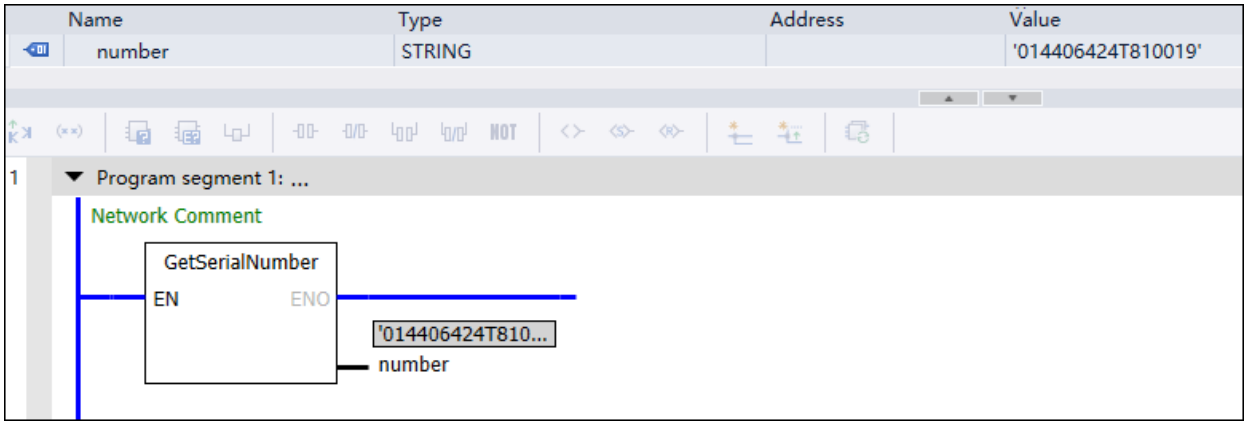
  This function block is mainly used to get the PLC serial number.

- **Program example**

  ST:



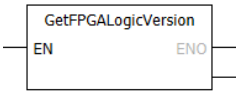
  LD:



18.4.1.24 GetFPGALogicVersion

This instruction gets the FPGA software version number.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
GetFPGALogicVersion	Get the FPGA logic version number	FC		:= GetFPGALogicVersion();

- **Related variables**

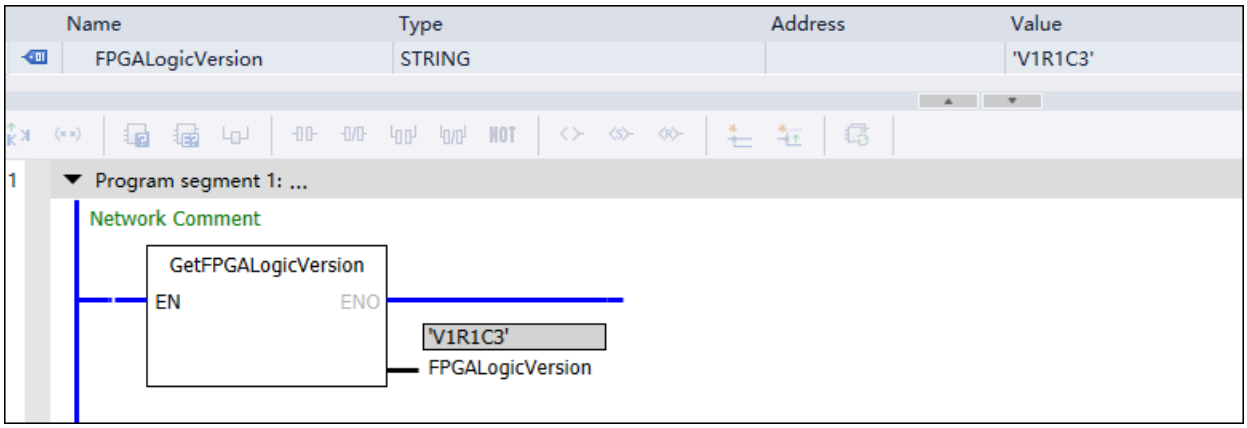
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	GetFPGALogicVersion	Get the FPGA software version number	STRING	-	-	This instruction gets the FPGA software version number.

- **Function description**
  This function block is primarily used to get the FPGA software version number.
- **Program example**

  ST:



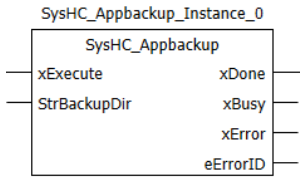
  LD:



18.4.1.25 SysHC_Appbackup

This instruction backups user program. The function block implements user program triggered backup through an external library method.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SysHC_Appbackup	Instruction-based user program data backup	FB		SysHC_Appbackup_Instance_0(xExecute := , StrBackupDir := , xDone => , xBusy => , xError => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Function block enable	BOOL	TRUE, FALSE	FALSE	Enable of triggering at the rising edge to obtain the backup result.
	StrBackupDir	Backup file directory name	STRING[64]	-	-	Specified directory name to start with the "/", without any illegal characters such as @ # , !, supporting up to 32 bytes.
Output	xDone	Output active	BOOL	TRUE, FALSE	FALSE	Output active.
	xBusy	Instruction executing	BOOL	TRUE, FALSE	FALSE	Executing
	xError	Fault	BOOL	TRUE, FALSE	FALSE	Error.
	eErrorID	Fault ID	BYTE	-	0	Error ID. For details, see Fault description.

- **Function**
  - This instruction applies to the EVO400, EVO500, EVO700, and EVO800 series products.
  - To write backup data to an external USB storage device (USB flash disk), call the SysHC_UDiskPath instruction to get the file path of the USB storage device connected to the PLC . The StrBackupDir parameter can be set to the path to call the SysHC_AppBackup backup instruction.
  - For EVO400/EVO500, if input parameter "StrBackupDir" is empty, the default storage path is: /home/ino/inosys/app/PLCprogram.
  - For EVO700, if input parameter "StrBackupDir" is empty, the default storage path is: /inocfg/app/PLCprogram.
  - For EVO800, if input parameter "StrBackupDir" is empty, the default storage path is: /home/ino/inocfg/app/PLCprogram.
- **Program example**

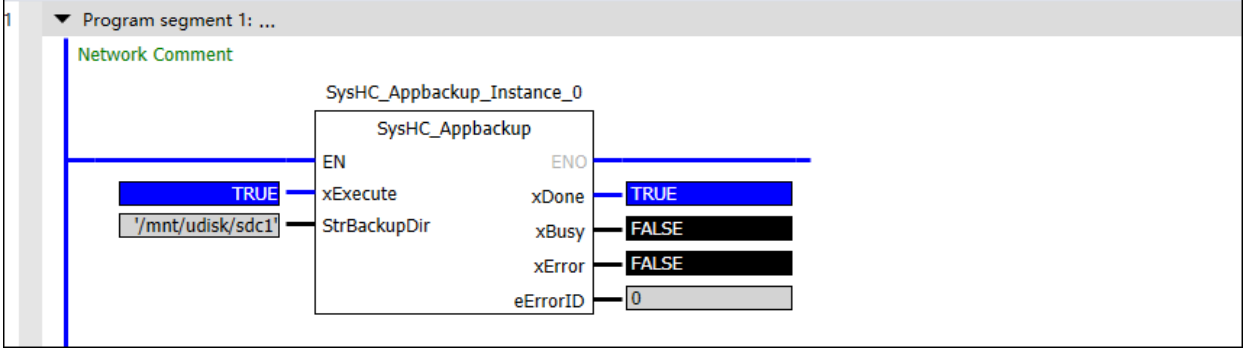
  ST:

Name	Type	Address	Value
SysHC_Appbackup_Instance_0	SysHC_Appbackup		
xExecute	BOOL		TRUE
StrBackupDir	STRING[64]		'/mnt/udisk/sdc1'
xDone	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
eErrorID	BYTE		0

```
SysHC_Appbackup_Instance_0(
xExecute TRUE := xExecute TRUE ,
StrBackupDir'/mnt/udisk/sd... := StrBackupDir'/mnt/udisk/sd...,
xDone TRUE => xDone TRUE ,
xBusy FALSE => xBusy FALSE ,
xError FALSE => xError FALSE ,
eErrorID 0 => eErrorID 0 );RETURN
```

LD:

Name	Type	Address	Value
SysHC_Appbackup_Instance_0	SysHC_Appbackup		
xExecute	BOOL		TRUE
StrBackupDir	STRING[64]		'/mnt/udisk/sdc1'
xDone	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
eErrorID	BYTE		0
xLastExecute	BOOL		TRUE



- **Fault description**

  When an error occurs, locate the corresponding error via eErrorID and find the cause of the error.

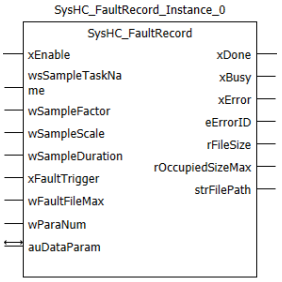
Fault Code	Description
0xEE	No SD card or USB flash disk is detected.
0xAA	The directory name "StrBackupDir" is invalid.
0x11	The source file for backup does not exist.
0x12	The system memory usage exceeds the limit, causing the failure to start backup.
0x13	Multiple instances are calling this instruction repeatedly. Repeated calls are prohibited.
0x31	Failed to create the backup directory before backup execution.
0x32	Failed to pack the backup files during the backup process.
0x33	No corresponding backup file is generated after backup ends.
0xEF	Multiple USB flash disks are inserted, but only one is allowed during backup.

- **Precautions**
  - EVO400/EVO500 PLCs have limited built-in flash storage space, so an external SD expansion card is required (which needs to be expanded by using the GE20-TF-RTC expansion card) to store the data backed up by the backup instruction. EVO400/AM500 PLCs do not support hot swapping recognition of SD cards, so the SD card must be connected before the device is powered on and then started.
  - When the system memory usage exceeds 80%, backup is not allowed, and calling the backup instruction returns the error code 0x12.
  - Only the latest backup file package is retained. Multiple backups will clear other file packages in the backup directory.
  - If an IEC task enables backup instruction pins by event triggering, events need to be continuously triggered to refresh the instruction execution result. This prevents pin occupancy from causing the failure to call instructions in other tasks.
  - The backup instruction function block is not suitable for calling under FUN and METH.
  - When the backup instruction is called for backup, it is necessary to ensure that all the .app and .crc files under the app folder exist. If any one is missing, the error code 0x11 will be returned.
  - The instruction cannot be triggered for backup when there are multiple apps in the PLC.
  - The user program restoration method is the same as the SD card user program upgrade method described above. For details, see 14.1.2.2 Upgrading User Program via SD Card/USB Disk.

18.4.1.26 SysHC_FaultRecord

When a specified fault is triggered, the associated variable data during the period before and after the fault is automatically collected and saved as a file.

- **Instruction format**

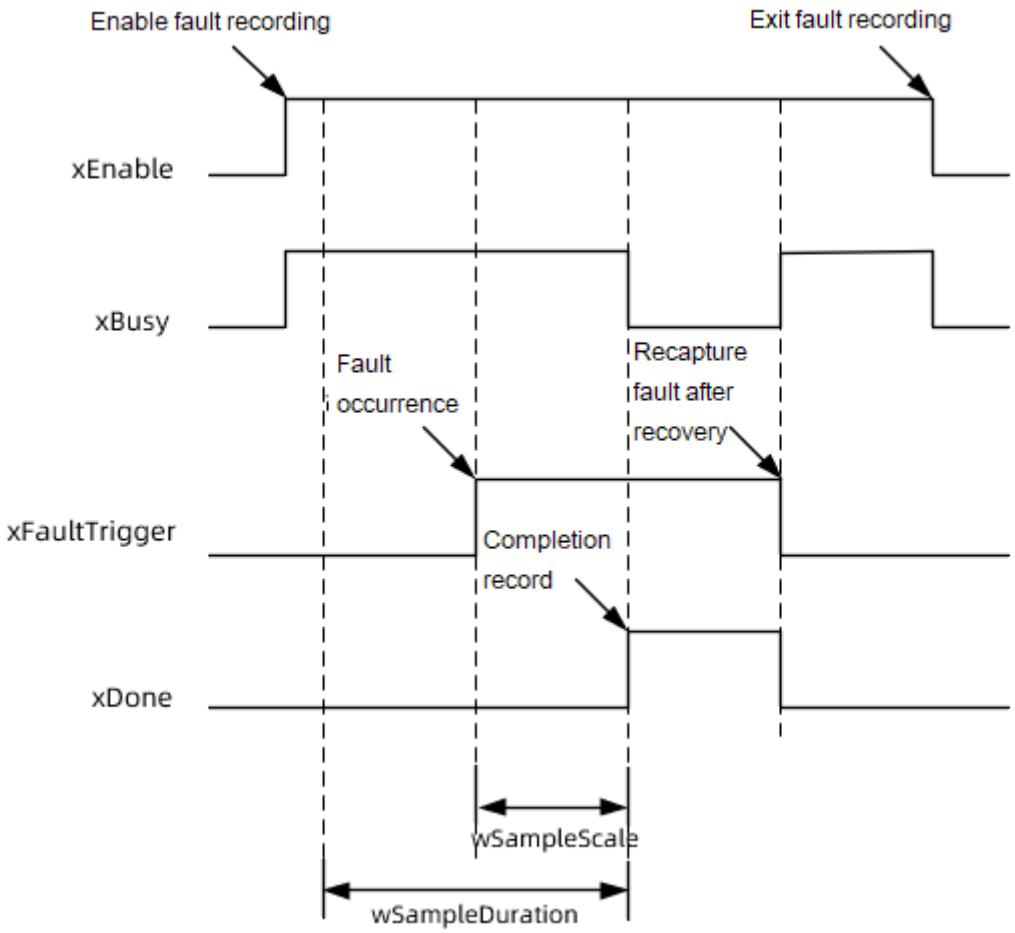
Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SysHC_FaultRecord	Get fault data recording by calling instruction	FB		SysHC_FaultRecord_Instance_0(xEnable := , wsSampleTaskName := , wSampleFactor := , wSampleScale := , wSampleDuration := , xFaultTrigger := , wFaultFileMax := , wParaNum := , xDone => , xBusy => , xError => , eErrorID => , rFileSize => , rOccupiedSizeMax => , strFilePath => , auDataParam :=);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input/ Output	auDataParam	Variable recording symbol	ARRAY[*] OF ANY	Follows data type	-	Array of variables to be recorded. The number of array elements must not exceed 64, and the element type is WSTRING.
Input	xEnable	Instruction enable	BOOL	TRUE, FALSE	FALSE	High level triggered
	wsSampleTaskName	Sampling task	WSTRING	Follows data type	-	The instruction configures the variable data to be collected in the task.
	wSampleFactor	Sampling cycle factor	WORD	>=1	-	One sampling cycle = Sampling cycle factor × Cycle of the sampling task The calculation result must be within 1ms to 1s.
	wSampleScale	Sampling ratio	WORD	10 to 90	-	The ratio of the sampling time after a fault trigger to the total sampling time (wSampleDuration).
	wSampleDuration	Total sampling time	WORD	EVO500 and APC520 series: [20, 120] Other series: [20, 600]	-	Total sampling duration, unit: seconds
	xFaultTrigger	Fault trigger signal	ANY	Follows data type	-	Indicates whether a fault is triggered.
	wFaultFileMax	Max. number of record files	WORD	(0, 1000]	-	If this value is exceeded, the oldest fault record file will be overwritten.

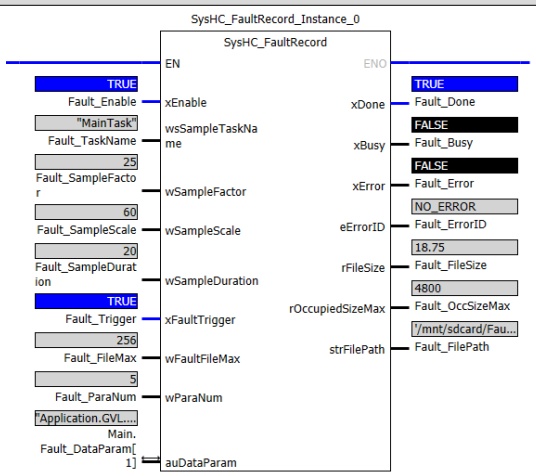
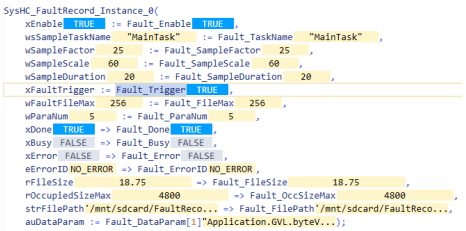
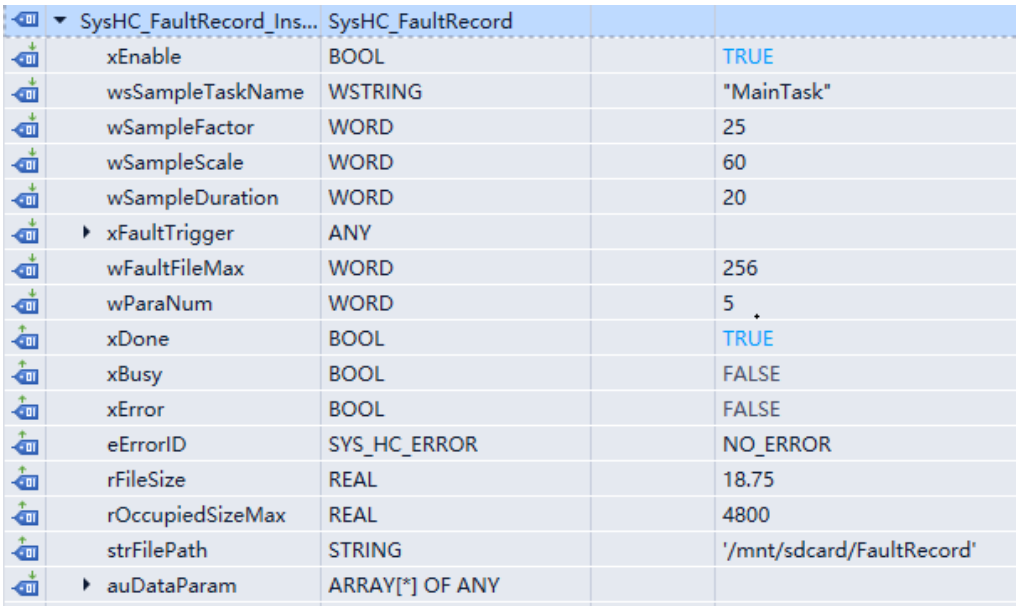
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	wParaNum	Number of variables to be sampled	WORD	EVO500 and APC520 series: <=32 Other series: <=64	-	The number of variables collected in the record file cannot exceed the number of elements in the auDataParam array. For example, 10 indicates that the values of the first 10 elements in the auDataParam array are recorded.
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	-	Indicates whether the instruction is completed.
	xBusy	Execution flag	BOOL	TRUE, FALSE	-	Indicates the instruction is being executed.
	xError	Error flag	BOOL	TRUE, FALSE	-	Indicates the error of the instruction.
	eErrorID	Fault code	SYS_HC_ERROR	Follows data type	-	See Fault Codes.
	rFileSize	Size of record file	REAL	Follows data type	-	Size of a record file, unit: kByte
	rOccupiedSizeMax	Maximum storage space occupied by record files	REAL	Follows data type	-	Size of all record files, rOccupiedSizeMax = rFileSize × wFaultFileMax, unit: kByte
	strFilePath	Storage path for record files	STRING	Follows data type	-	Absolute path for storing the record files.

- **Function description**
  When a fault is specified to trigger, the associated variable data within the time period before and after the fault (wSampleDuration) will be automatically collected and saved as a file to an external storage device (USB flash drive or SD card).
- **Timing diagram**



- **Program example**

Item	LD	ST
Define variables	VAR SysHC_FaultRecord_Instance_0: SysHC_FaultRecord; Fault_Enable: BOOL; Fault_TaskName: WSTRING; Fault_SampleFactor: WORD; Fault_SampleScale: WORD; Fault_SampleDuration: WORD; Fault_Trigger: BOOL; Fault_FileMax: WORD; Fault_ParaNum: WORD; Fault_Done: BOOL; Fault_Busy: BOOL; Fault_Error: BOOL; Fault_ErrorID: SYS_HC_ERROR; Fault_FileSize: REAL; Fault_OccSizeMax: REAL; Fault_FilePath: STRING; Fault_DataParam: ARRAY[1..32] OF WSTRING; END_VAR	

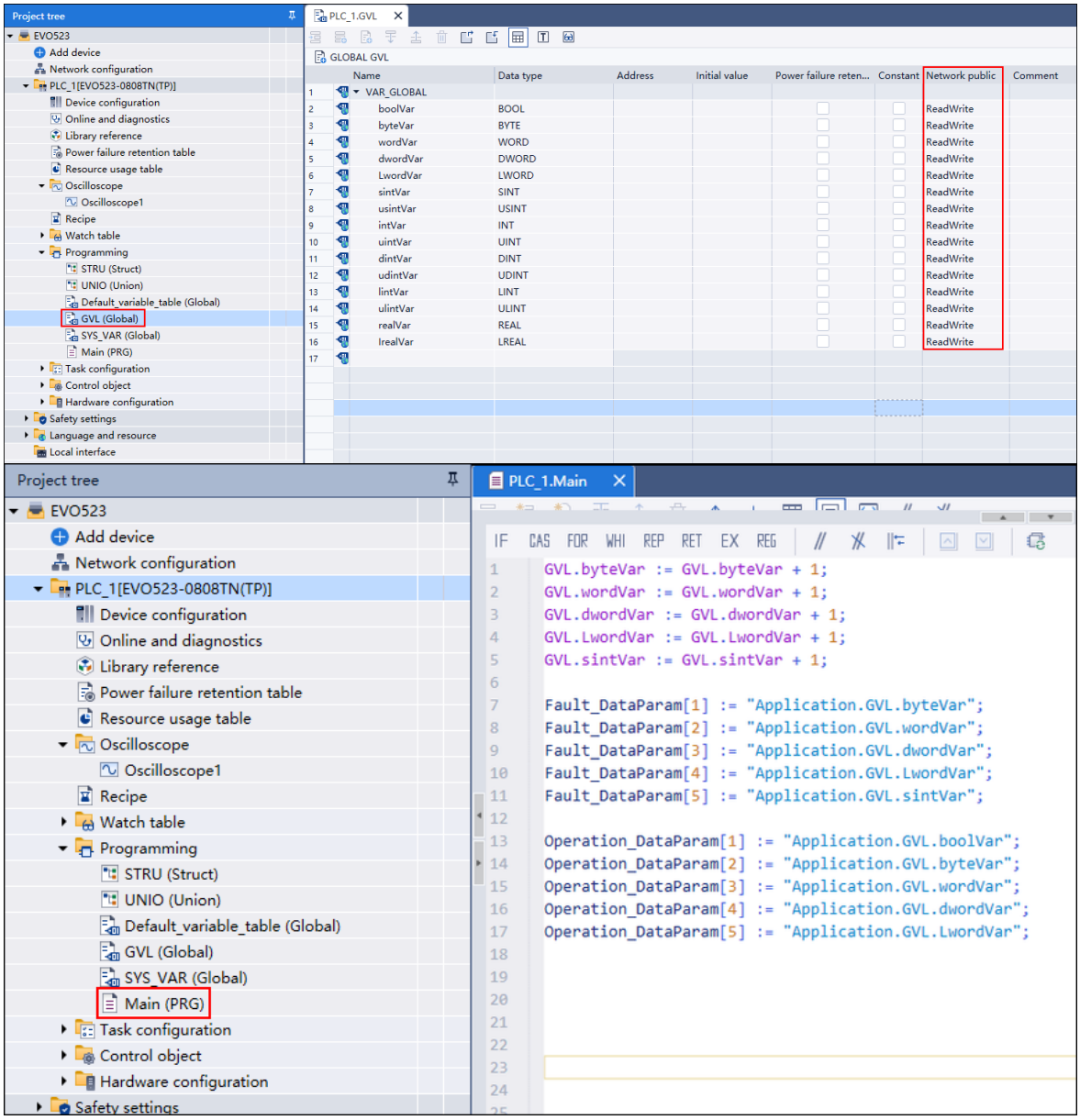
Item	LD	ST
Program		
Run result		

- **Usage Example**

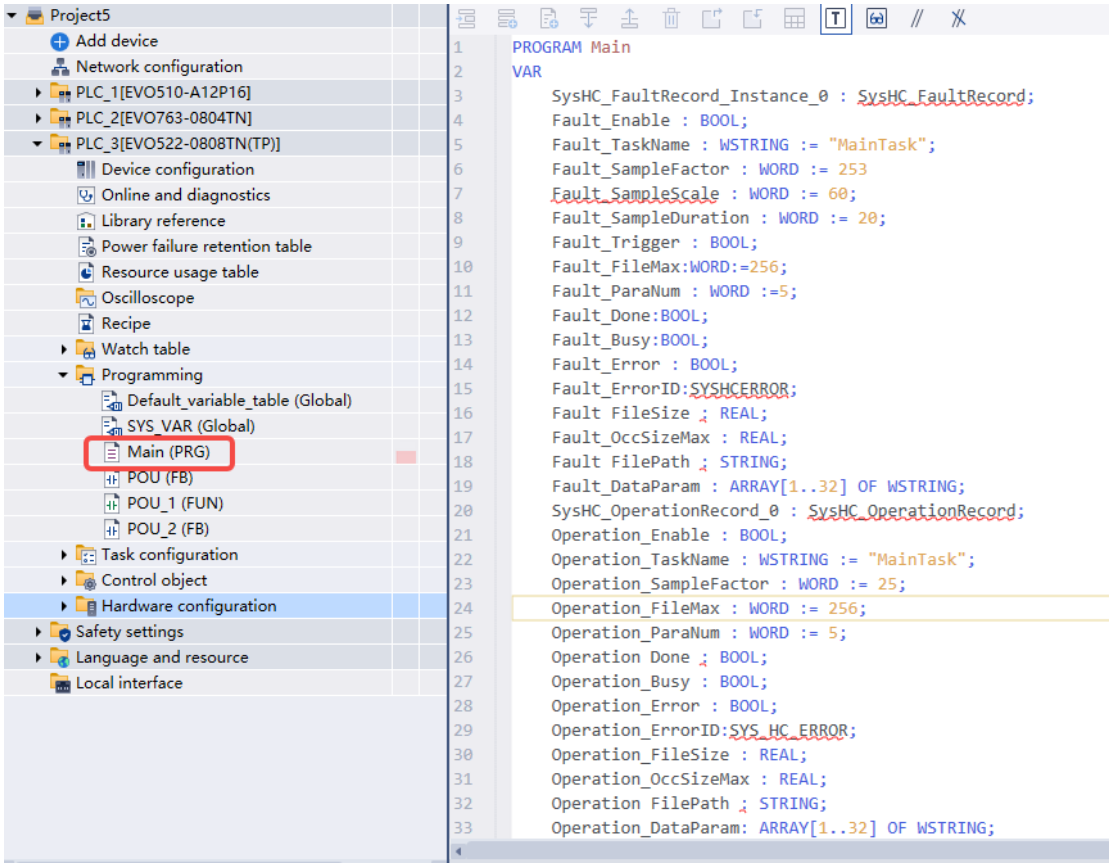
  1. Define the global or local variables to be recorded in the sampling task. The data types for fault and data recording include: BOOL, BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT, REAL and LREAL.

     > NOTE
     >
     > The data involved in recording must have read-write or read permissions.

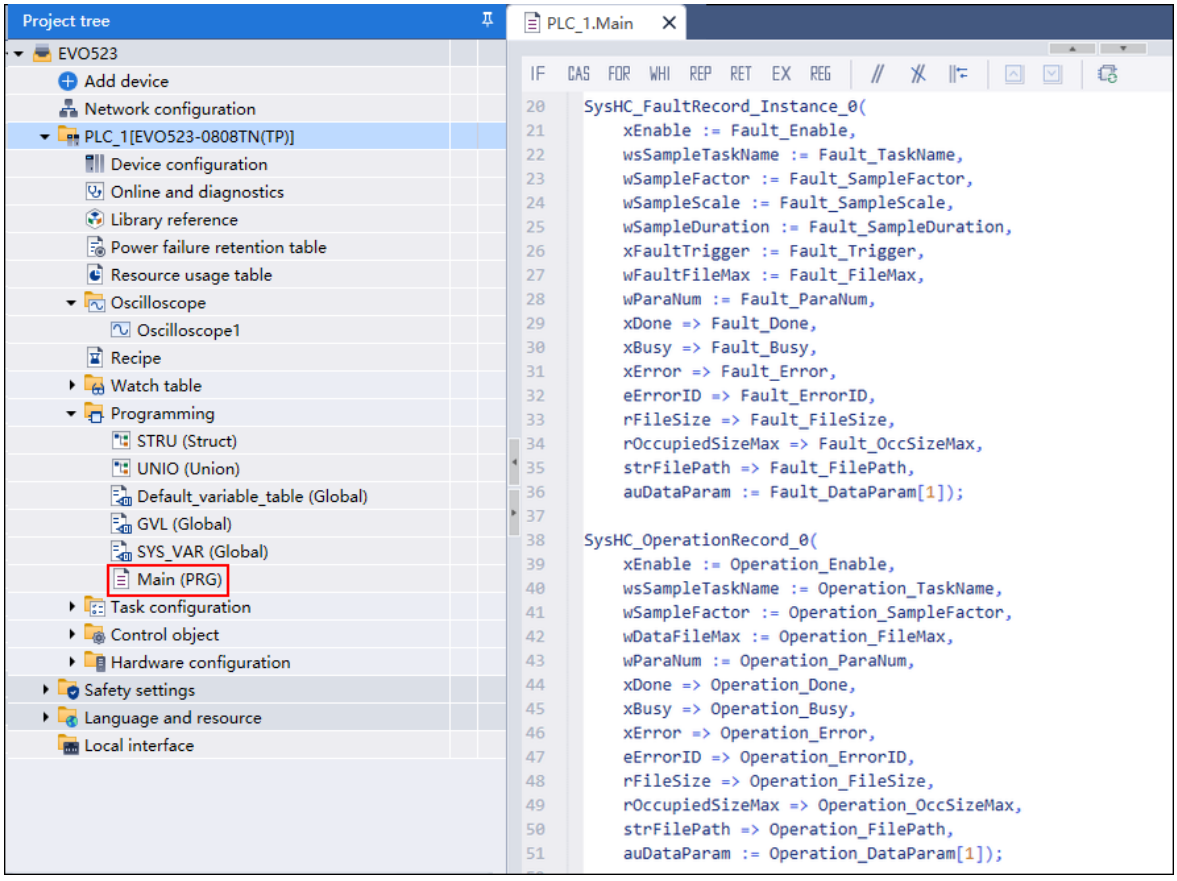
     The following figures take a global variable as an example.



2. Configure the fault and data recording instruction.



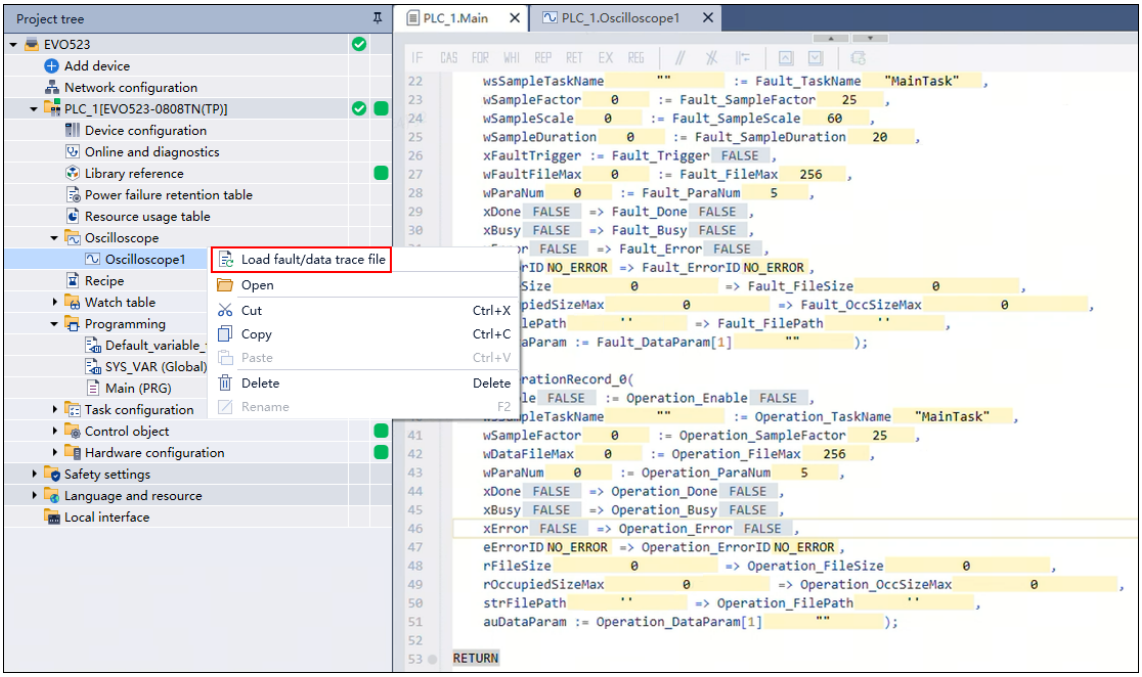
3. Call the fault and data recording function block.



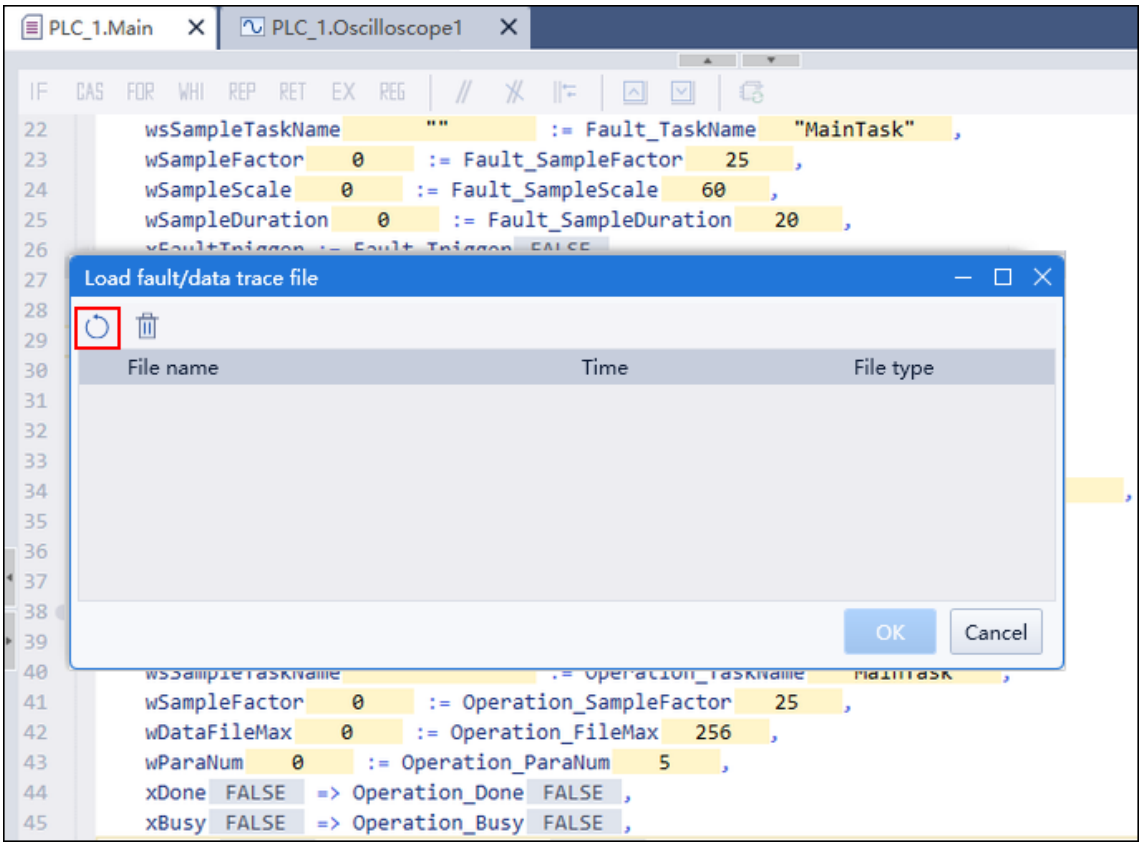
4. After executing the fault and data recording function block, you can view or export the related record data in the oscilloscope.

   a. Right-click "Oscilloscope" in the project tree, select "Add oscilloscope", and create a new oscilloscope.

   b. Right-click the newly created oscilloscope, and selected "Load fault/data trace file".

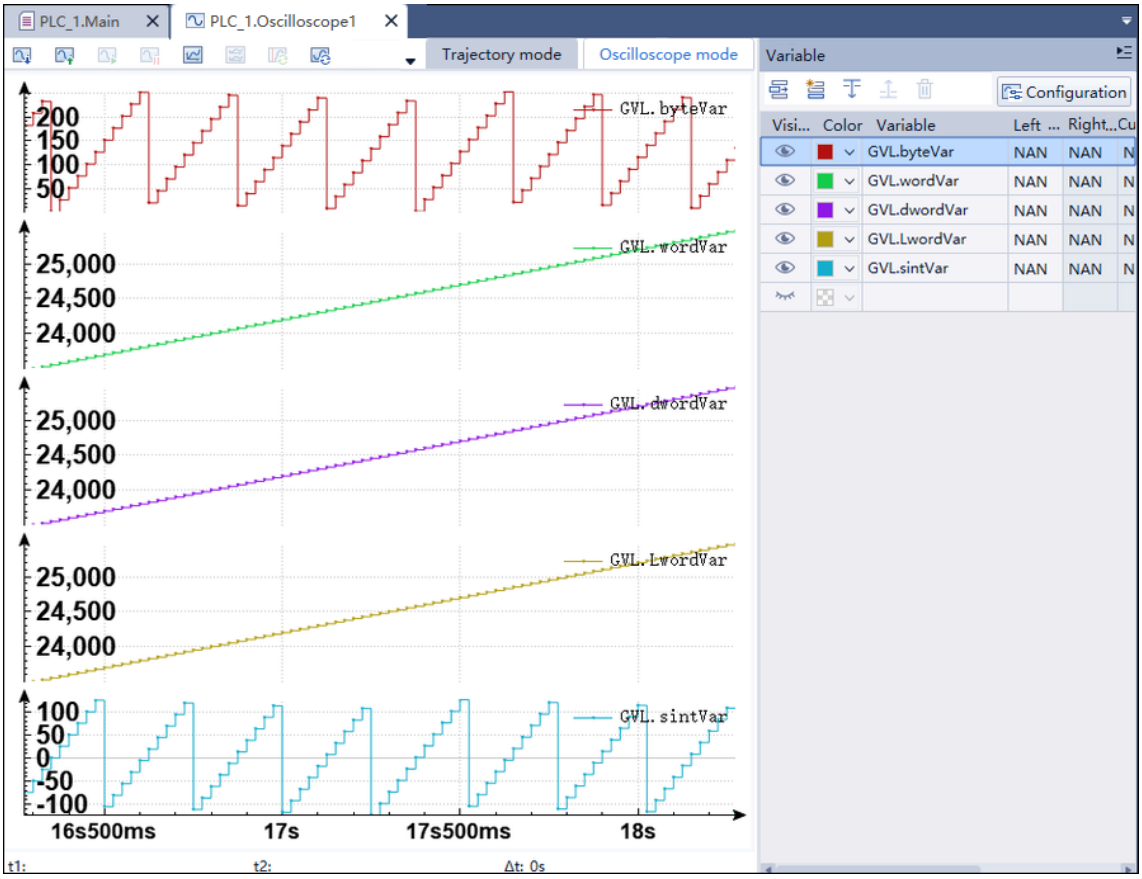


c. Click the refresh button in the pop-up window.



d.

Select the fault or data recording file, click "OK", and the data will be loaded.



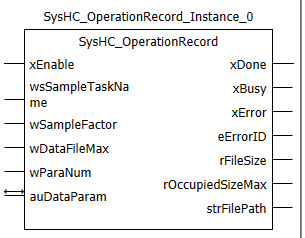
e. (Optional) Right-click the oscilloscope interface, select "Save oscilloscope data", and then choose the save format.

- **Precautions**
  - This instruction can be called only once at the same time.
  - After executing the instruction, a record file will be generated each time a fault is triggered.
  - The task mounting of the recording function is determined by the input parameter (wsSampleTaskName), independent of the task where the instruction resides.
  - Place the instruction in a lower-priority task to avoid affecting higher-priority IEC tasks.
    For example, if the EtherCAT task priority is 0 and MainTask priority is 2 in the project, place the instruction in MainTask and set the wsSampleTaskName parameter to the EtherCAT task. In this case, the recording function can synchronize sampling with the EtherCAT task without affecting the task stability.
  - Before using this instruction, a folder named FaultRecord must be created on the target external memory device (USB flash disk or SD card).
  - Ensure the total byte length of instruction data meets the requirements: 1 to 32 bytes for EVO500 and APC520 series, and 1 to 128 bytes for other series.

18.4.1.27 SysHC_OperationRecord

This instruction automatically collects associated variable data and store it as a file.

- **Instruction format**

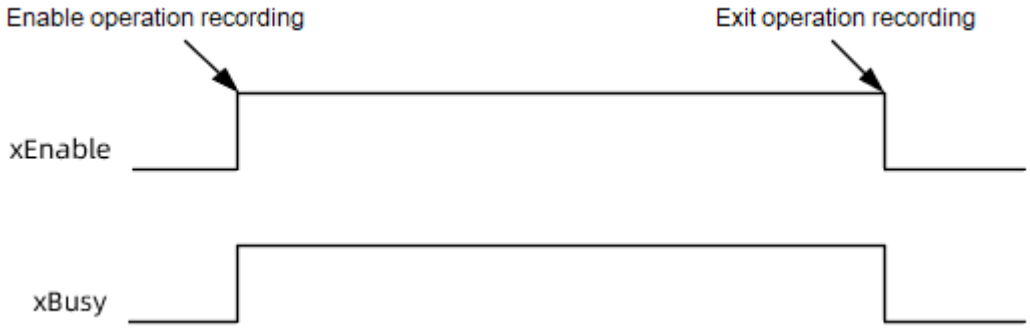
Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SysHC_OperationRecord	Get program data recording by calling instruction	FB		SysHC_OperationRecord_Instance_0(xEnable := , wsSampleTaskName := , wSampleFactor := , wDataFileMax := , wParaNum := , xDone => , xBusy => , xError => , eErrorID => , rFileSize => , rOccupiedSizeMax => , strFilePath => , auDataParam :=);

- **Related variables**

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input/Output	auDataParam	Variable recording symbol	ARRAY[*] OF ANY	Follows data type	-	Array of variables to be recorded. The number of array elements must not exceed 64, and the element type is WSTRING.
Input	xEnable	Instruction enable	BOOL	TRUE, FALSE	FALSE	High level triggered
	wsSampleTaskName	Sampling task	WSTRING	Follows data type	-	The instruction configures the variable data to be collected in the task.
	wSampleFactor	Sampling cycle factor	WORD	>=1	-	One sampling cycle = Sampling cycle factor × Cycle of the sampling task The calculation result must be within 1ms to 10s.
	wDataFileMax	Max. number of record files	WORD	(0, 720]	-	If this value is exceeded, the oldest record file will be overwritten.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	wParaNum	Number of variables to be sampled	WORD	EVO500 and APC520 series: <=32 Other series: <=64	-	The number of variables collected in the record file cannot exceed the number of elements in the auDataParam array. For example, 10 indicates that the values of the first 10 elements in the auDataParam array are recorded.
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	-	Indicates whether the instruction is completed.
	xBusy	Execution flag	BOOL	TRUE, FALSE	-	Indicates the instruction is being executed.
	xError	Error flag	BOOL	TRUE, FALSE	-	Indicates the error of the instruction.
	eErrorID	Fault code	SYS_HC_ERROR	Follows data type	-	See Fault Codes.
	rFileSize	Size of record file	REAL	Follows data type	-	Size of a record file, unit: kByte
	rOccupiedSizeMax	Maximum storage space occupied by record files	REAL	Follows data type	-	Size of all record files, rOccupiedSizeMax = rFileSize × wDataFileMax, unit kByte.
	strFilePath	Storage path for record files	STRING	Follows data type	-	Absolute path for storing the record files.

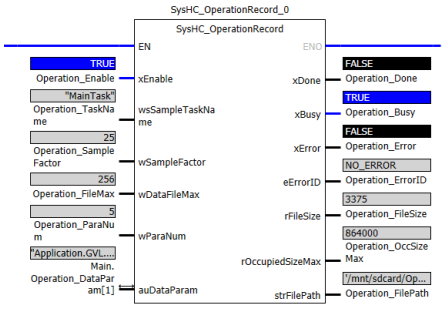
- **Function description**
  This instruction collects associated variable data automatically and saves it as a file to an external storage device (USB flash disk or SD card).
- **Timing diagram**



- **Program example**

Item	LD	ST
Define variables		
Program		

```
VAR
    SysHC_OperationRecord_Instance_0: SysHC_OperationRecord;
    Operation_Enable: BOOL;
    Operation_TaskName: WSTRING;
    Operation_SampleFactor: WORD;
    Operation_FileMax: WORD;
    Operation_ParaNum: WORD;
    Operation_Done: BOOL;
    Operation_Busy: BOOL;
    Operation_Error: BOOL;
    Operation_ErrorID: SYS_HC_ERROR;
    Operation_FileSize: REAL;
    Operation_OccSizeMax: REAL;
    Operation_FilePath: STRING;
    Operation_DataParam: ARRAY[1..32] OF WSTRING;
END_VAR
```




```
SysHC_OperationRecord_0(
    xEnable  TRUE  := Operation_Enable  TRUE ,
    wsSampleTaskName  "MainTask"  := Operation_TaskName  "MainTask" ,
    wSampleFactor  25  := Operation_SampleFactor  25 ,
    wDataFileMax  256  := Operation_FileMax  256 ,
    wParaNum  5  := Operation_ParaNum  5 ,
    xDone  FALSE  => Operation_Done  FALSE ,
    xBusy  TRUE  => Operation_Busy  TRUE ,
    xError  FALSE  => Operation_Error  FALSE ,
    eErrorID NO_ERROR  => Operation_ErrorID NO_ERROR ,
    rFileSize  3375  => Operation_FileSize  3375 ,
    rOccupiedSizeMax  864000  => Operation_OccSizeMax  864000 ,
    strFilePath'/mnt/sdcard/OperationReco...  => Operation_FilePath'/mnt/sdcard/OperationReco...,
    auDataParam := Operation_DataParam[1]"Application.GVL.boolV...);
```

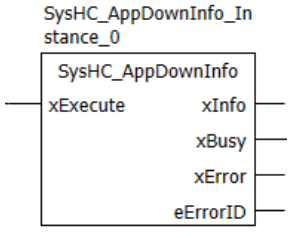
Item	LD	ST
Operation result	SysHC_OperationReco... SysHC_OperationRecord xEnable BOOL TRUE wsSampleTaskName WSTRING "MainTask" wSampleFactor WORD 25 wDataFileMax WORD 256 wParaNum WORD 5 xDone BOOL FALSE xBusy BOOL TRUE xError BOOL FALSE eErrorID SYS_HC_ERROR NO_ERROR rFileSize REAL 3375 rOccupiedSizeMax REAL 864000 strFilePath STRING '/mnt/sdcard/OperationRecord' auDataParam ARRAY[*] OF ANY	

- **Usage example**
  For details, see the Usage example in the SysHC_FaultRecord instruction.
- **Precautions**
  - This instruction can be called only once at the same time.
  - After executing the instruction, a record file will be generated every hour.
  - The task mounting of the recording function is determined by the input parameter (wsSampleTaskName), independent of the task where the instruction resides.
  - Place the instruction in a lower-priority task to avoid affecting higher-priority IEC tasks.
    For example, if the EtherCAT task priority is 0 and MainTask priority is 2 in the project, place the instruction in MainTask and set the wsSampleTaskName parameter to the EtherCAT task. In this case, the recording function can synchronize sampling with the EtherCAT task without affecting the task stability.
  - Before using this instruction, a folder named OperationRecord must be created on the target external memory device (USB flash disk or SD card).
  - Ensure the total byte length of instruction data meets the requirements: 1 to 32 bytes for EVO500 and APC520 series, and 1 to 128 bytes for other series.

18.4.1.28 SysHC_AppDownInfo

This instruction gets the latest user program information for the PLC.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SysHC_AppDownInfo	Get the latest user program download information of the PLC	FB		SysHC_AppDownInfo_Instance_0(xExecute := , xInfo => , xBusy => , xError => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Instruction enable	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
Output	xInfo	Download project info	ARRAY [0..4] OF APPDOWN_INFO	-	-	Download the project information.
	xBusy	Execution flag	BOOL	TRUE, FALSE	-	Indicates the instruction is being executed.
	xError	Error flag	BOOL	TRUE, FALSE	-	Indicates the error of the execution.
	eErrorID	Fault code	SYS_HC_ERROR	Follows data type	-	Instruction fault codes. For details, see Fault Codes.

- **Function**
  - This instruction gets the latest user program information of the PLC, including the project name, download timestamp, project checksum, host computer name, and download mode.
  - xInfo is of structure type APPDOWN_INFO. The structure members are described in the table below.

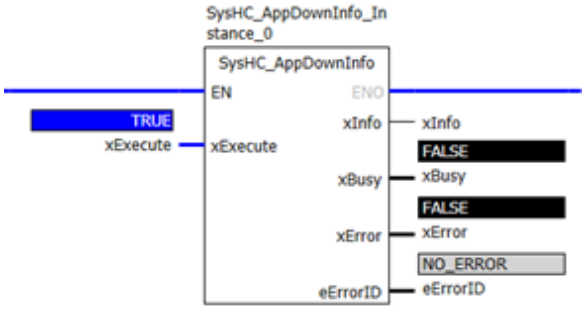
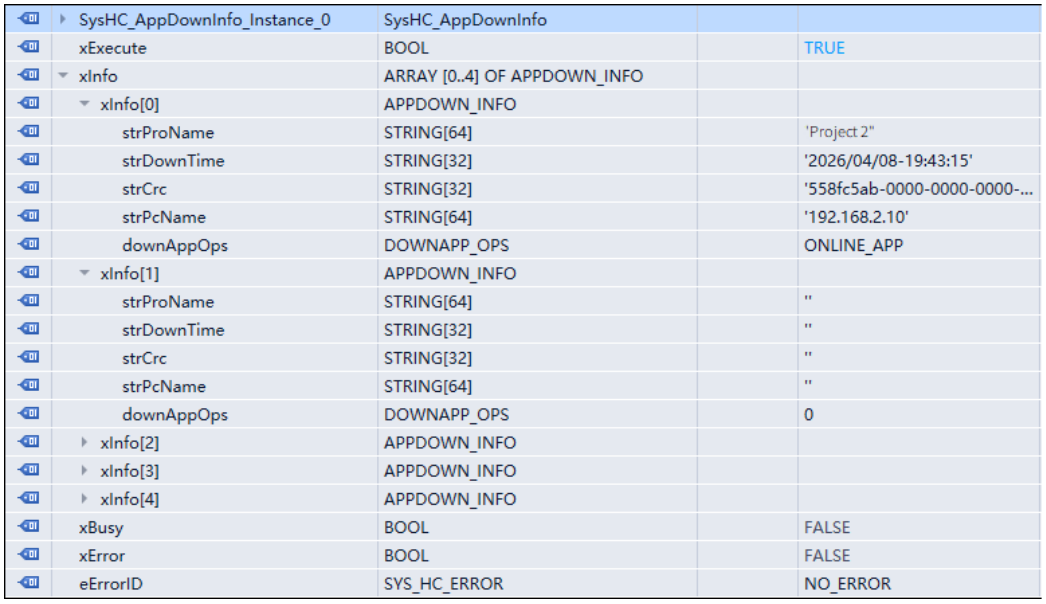
Variable	Data Type	Valid Range	Default Value	Description
strProName	STRING[64]	-	-	Project name
strDownTime	STRING[32]	-	-	Read timestamp
strCrc	STRING[32]	-	-	Project checksum
strPcName	STRING[64]	-	-	Host computer name
downAppOps	DOWNAPP_OPS	-	-	Download mode

downAppOps is of enumeration type DOWNAPP_OPS. The descriptions of enumeration values are in the table below.

Variable	Value	Description
ONLINE_APP	1	Online download
UDISK_APP	2	USB flash disk download
DOWNLOAD_APP	3	Full download

- **Program example**

Item	LD	ST
Define variables	VAR SysHC_AppDownInfo_Instance_0: SysHC_AppDownInfo; xExecute: BOOL; xInfo: ARRAY [0..4] OF APPDOWN_INFO; xBusy: BOOL; xError: BOOL; eErrorID: SYS_HC_ERROR; END_VAR	

Item	LD	ST
Program		SysHC_AppDownInfo_Instance_0(xExecute TRUE := xExecute TRUE, xInfo => xInfo, xBusy FALSE => xBusy FALSE, xError FALSE => xError FALSE, eErrorID NO_ERROR => eErrorID NO_ERROR);
Operation result		

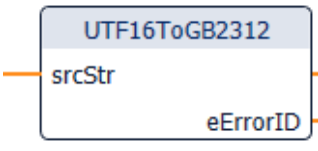
SysHC_AppDownInfo_Instance_0	SysHC_AppDownInfo		
xExecute	BOOL		TRUE
xInfo	ARRAY [0..4] OF APPDOWN_INFO		
xInfo[0]	APPDOWN_INFO		
strProName	STRING[64]		'Project 2"
strDownTime	STRING[32]		'2026/04/08-19:43:15'
strCrc	STRING[32]		'558fc5ab-0000-0000-0000-...
strPcName	STRING[64]		'192.168.2.10'
downAppOps	DOWNAPP_OPS		ONLINE_APP
xInfo[1]	APPDOWN_INFO		
strProName	STRING[64]		''
strDownTime	STRING[32]		''
strCrc	STRING[32]		''
strPcName	STRING[64]		''
downAppOps	DOWNAPP_OPS		0
xInfo[2]	APPDOWN_INFO		
xInfo[3]	APPDOWN_INFO		
xInfo[4]	APPDOWN_INFO		
xBusy	BOOL		FALSE
xError	BOOL		FALSE
eErrorID	SYS_HC_ERROR		NO_ERROR

- **Precautions**
  - This instruction only gets the latest five download records of the first user program.
  - If the output parameters strProName and strPcName display garbled text, you need to use the ConvertUTF8toUTF16 instruction to convert them to Chinese.

18.4.1.29 UTF16ToGB2312

This instruction converts the encoding format from UTF-16 to GB2312.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
UTF16ToGB2312	UTF16 to GB2312 encoding conversion	FC		:= UTF16ToGB2312(srcStr := , eErrorID =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	srcStr	Source string	Follows data type	-	Input string

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Output	UTF16ToGB2312	Conversion result	Follows data type	-	The conversion result obtained after the instruction execution is completed.
	eErrorID	Fault code	Follows data type	-	Fault code after conversion. 0: NO_ERROR (Conversion completed) 1: INVAILD_INPUT (Input error) 2: CONVERT_FAIL (Conversion failed)

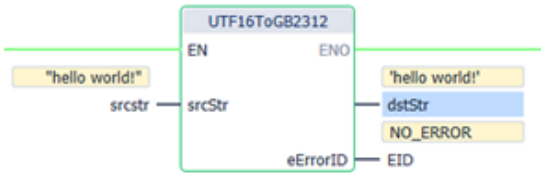
Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
srcStr	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
UTF16ToGB2312	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
eErrorID	SYS_HC_CONVERT_ERROR																				

- **Function**
  This instruction converts the encoding format from UTF-16 to GB2312.
- **Program example**

<table>
<tr><th>Item</th><th>LD</th><th>ST</th></tr>
<tr><td>Define variables</td><td colspan="2">

```
VAR
    srcstr: WSTRING;
    EID: SYS_HC_CONVERT_ERROR;
    dstStr: STRING;
END_VAR
```

</td></tr>
<tr><td>Program</td><td>
</td><td>

```
dstStr 'hello world!'  := UTF16ToGB2312(
    srcStr := srcstr "hello world!" ,
    eErrorID => EID NO_ERROR );RETURN
```

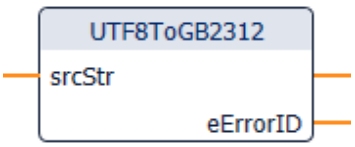
</td></tr>
<tr><td>Execution result</td><td colspan="2">
<table>
<tr><th>Nmae</th><th>Type</th><th>Address</th><th>Value</th></tr>
<tr><td>▼ VAR</td><td></td><td></td><td></td></tr>
<tr><td>srcstr</td><td>WSTRING</td><td></td><td>"hello world!"</td></tr>
<tr><td>EID</td><td>SYS_HC_CONVERT_ERROR</td><td></td><td>NO_ERROR</td></tr>
<tr><td>dstStr</td><td>STRING(1985)</td><td></td><td>'hello world!'</td></tr>
</table>
</td></tr>
</table>

- **Precautions**
  - This instruction supports a maximum conversion length of 1985 bytes. Overlimit will cause automatic truncation.
  - If the input contains characters unsupported by GB2312, UTF-8, or UTF-16, the conversion will fail.
  - The input variable of the UTF16ToGB2312 conversion instruction is limited to WSTRING. If a constant value is input manually, it must be strictly assigned in the WSTRING wide string format. The iFA software interface uniformly adopts UTF-8 character encoding. When calling the encoding conversion interface, strictly match the data type and character encoding format of the input variable. Mismatched parameters will directly cause conversion failure.

18.4.1.30 UTF8ToGB2312

This instruction converts the encoding format from UTF-8 to GB2312.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
UTF8ToGB2312	UTF8 To GB2312 encoding conversion	FC		:= UTF8ToGB2312(srcStr := , eErrorID =>);

- **Related variables**

Variable

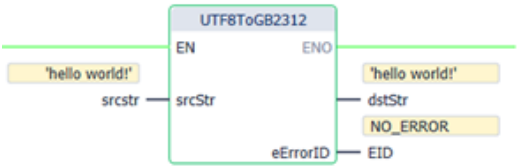
Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	srcStr	Source string	Follows data type	-	Input string

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	UTF8ToGB2312	Conversion result	Follows data type	-	The conversion result obtained after the instruction execution is completed.
	eErrorID	Fault code	Follows data type	-	Fault code after conversion. 0: NO_ERROR (Conversion completed) 1: INVAILD_INPUT (Input error) 2: CONVERT_FAIL (Conversion failed)

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
srcStr	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
UTF8ToGB2312	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
eErrorID	SYS_HC_CONVERT_ERROR																				

- **Function**
  This instruction converts the encoding format from UTF-8 to GB2312.
- **Program example**

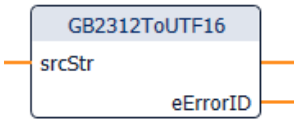
Item	LD	ST
Define variables	VAR srcstr: STRING; EID: SYS_HC_CONVERT_ERROR; dstStr: STRING; END_VAR	
Program		dstStr 'hello world!' := UTF8ToGB2312(srcStr := srcstr 'hello world!', eErrorID => EID NO_ERROR);
Execution result	VAR srcstr: STRING 'hello world!' EID: SYS_HC_CONVERT_ERROR NO_ERROR dstStr: STRING 'hello world!'	

- **Precautions**
  - This instruction supports a maximum conversion length of 1985 bytes. Overlimit will cause automatic truncation.
  - If the input contains characters unsupported by GB2312, UTF-8, or UTF-16, the conversion will fail.
  - The input variable of the UTF8ToGB2312 conversion instruction is limited to STRING. If a constant value is input manually, it must be strictly assigned in the STRING string format. The iFA software interface uniformly adopts UTF-8 character encoding. When calling the encoding conversion interface, strictly match the data type and character encoding format of the input variable. Mismatched parameters will directly cause conversion failure.

18.4.1.31 GB2312ToUTF16

This instruction converts the encoding format from GB2312 to UTF-16.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
GB2312ToUTF16	GB2312 To UTF16 encoding conversion	FC		:= GB2312ToUTF16(srcStr := , eErrorID =>);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	srcStr	Source string	Follows data type	-	Input string
Output	GB2312ToUTF16	Conversion result	Follows data type	-	The conversion result obtained after the instruction execution is completed.

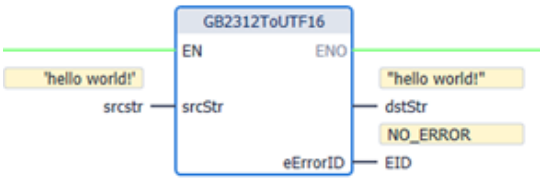
Input/ Output	Variable	Name	Valid Range	Default Value	Description
	eErrorID	Fault code	Follows data type	-	Fault code after conversion. 0: NO_ERROR (Conversion completed) 1: INVAILD_INPUT (Input error) 2: CONVERT_FAIL (Conversion failed)

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
srcStr	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
GB2312ToUTF16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
eErrorID	SYS_HC_CONVERT_ERROR																				

- **Function**
  This instruction converts the encoding format from GB2312 to UTF16.
- **Program example**

Item	LD	ST
Define variables	VAR srcstr: STRING; EID: SYS_HC_CONVERT_ERROR; dstStr: WSTRING; END_VAR	

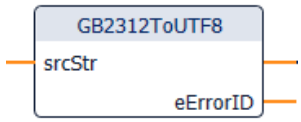
Item	LD	ST
Program		dstStr "hello world!" := GB2312ToUTF16(srcStr := srcstr 'hello world!', eErrorID => EID NO_ERROR);
Execution result	VAR srcstr STRING 'hello world!' EID SYS_HC_CONVERT_ERROR NO_ERROR dstStr WSTRING(1985) "hello world!"	

- **Precautions**
  - This instruction supports a maximum conversion length of 1985 bytes. Overlimit will cause automatic truncation.
  - If the input contains characters unsupported by GB2312, UTF-8, or UTF-16, the conversion will fail.
  - The input variable of the GB2312ToUTF16 conversion instruction is limited to STRING. If a constant value is input manually, it must be strictly assigned in the STRING string format. The iFA software interface uniformly adopts UTF-8 character encoding. When calling the encoding conversion interface, strictly match the data type and character encoding format of the input variable. Mismatched parameters will directly cause conversion failure.

18.4.1.32 GB2312ToUTF8

This instruction converts the encoding format from GB2312 to UTF-8.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
GB2312ToUTF8	GB2312 To UTF8 encoding conversion	FC		:= GB2312ToUTF8(srcStr := , eErrorID =>);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	srcStr	Source string	Follows data type	-	Input string
Output	GB2312ToUTF8	Conversion result	Follows data type	-	The conversion result obtained after the instruction execution is completed.

Input/ Output	Variable	Name	Valid Range	Default Value	Description
	eErrorID	Fault code	Follows data type	-	Fault code after conversion. 0: NO_ERROR (Conversion completed) 1: INVAILD_INPUT (Input error) 2: CONVERT_FAIL (Conversion failed)

Variable types

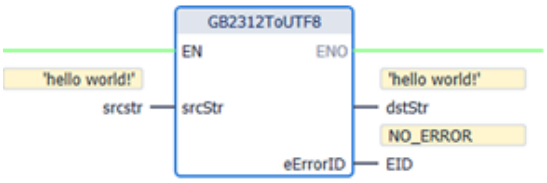
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String					
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING	WSTRING
srcStr	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
GB2312ToUTF8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	-
eErrorID	SYS_HC_CONVERT_ERROR																				

- **Function**

  This instruction converts the encoding format from GB2312 to UTF-8.

- **Program example**

Item	LD	ST
Define variables	VAR srcstr: STRING; EID: SYS_HC_CONVERT_ERROR; dstStr: STRING; END_VAR	

Item	LD	ST
Program		`dstStr 'hello world!' := GB2312ToUTF8( srcStr := srcstr 'hello world!' , eErrorID => EID NO_ERROR );`
Execution result	VAR srcstr: STRING: 'hello world!' EID: SYS_HC_CONVERT_ERROR: NO_ERROR dstStr: STRING(1985): 'hello world!'	

- **Precautions**
  - This instruction supports a maximum conversion length of 1985 bytes. Overlimit will cause automatic truncation.
  - If the input contains characters unsupported by GB2312, UTF-8, or UTF-16, the conversion will fail.
  - The input variable of the GB2312ToUTF8 conversion instruction is limited to STRING. If a constant value is input manually, it must be strictly assigned in the STRING string format. The iFA software interface uniformly adopts UTF-8 character encoding. When calling the encoding conversion interface, strictly match the data type and character encoding format of the input variable. Mismatched parameters will directly cause conversion failure.

18.4.1.33 SDCard1StatusGet

This instruction gets the SD card status and updates the corresponding variable in the system.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SDCard1StatusGet	Get all the status flags of SD memory card	FC		:= SDCard1StatusGet( );

- **Example**

  The SDCard1StatusGet instruction gets the SD card status and update the corresponding variable in the system. The following provides instructions for use.

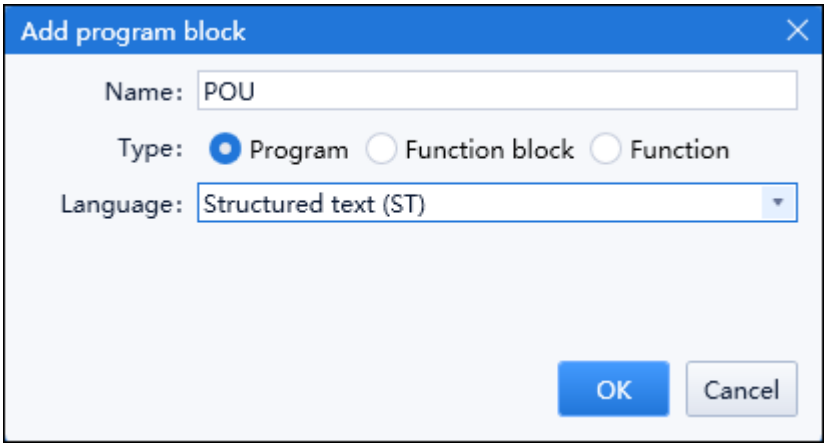
  Prerequisites

  • A project has been created and devices have been added. For specific operations, see 7 Project Deployment.

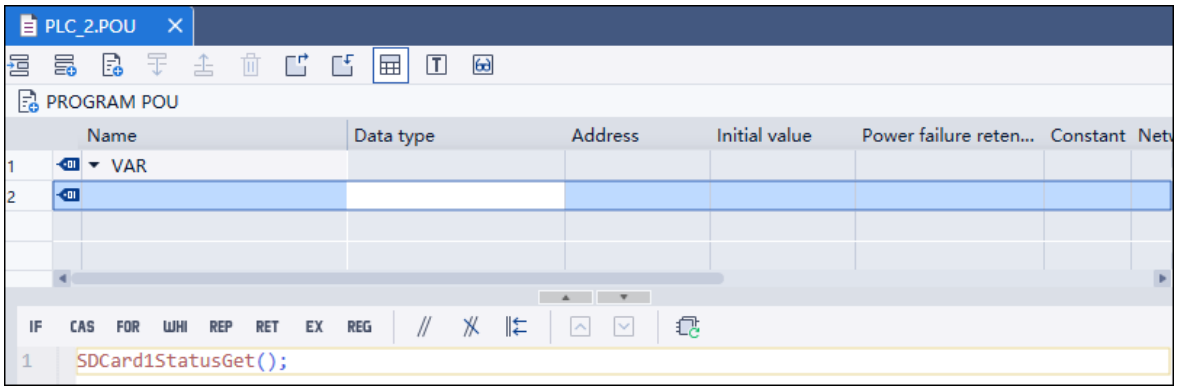
• The SysHCPlcInfo library version 1.5.0.0 or later has been referenced. For reference operations, see 12.3 Library Reference.

Procedure

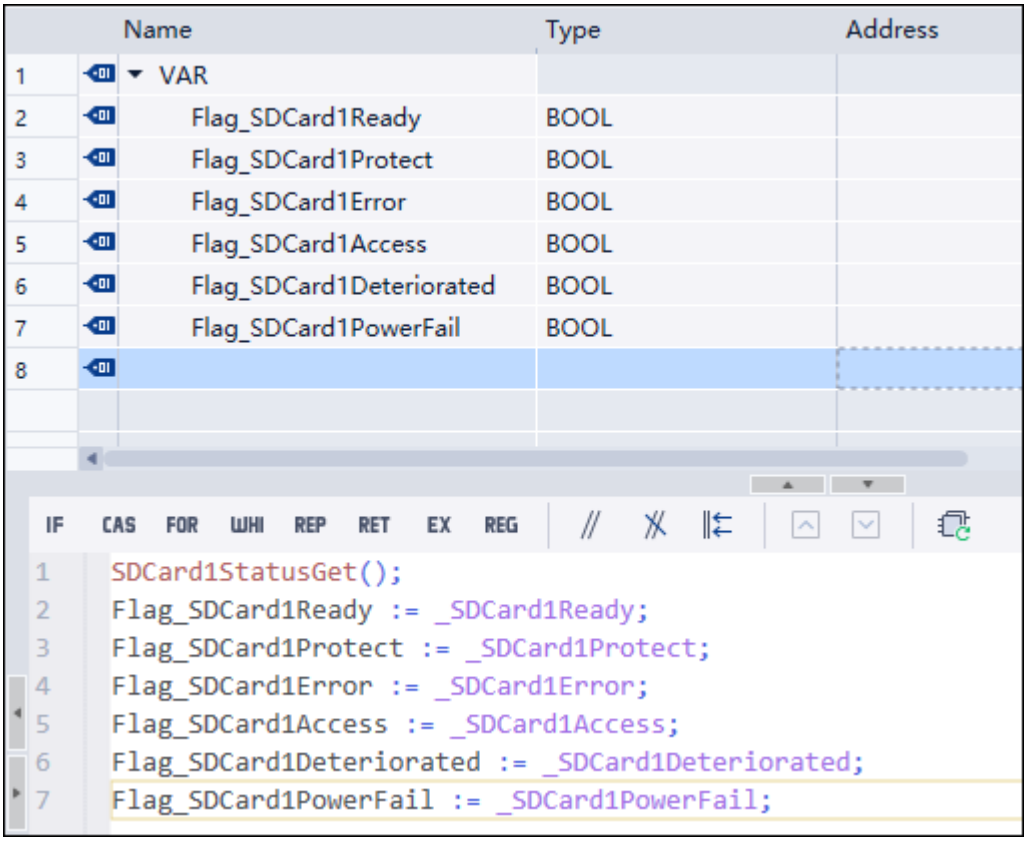
1. Click "Programming" in the project tree to create a POU of PRG type, with ST and LD languages both supported. This section takes ST language as an example.



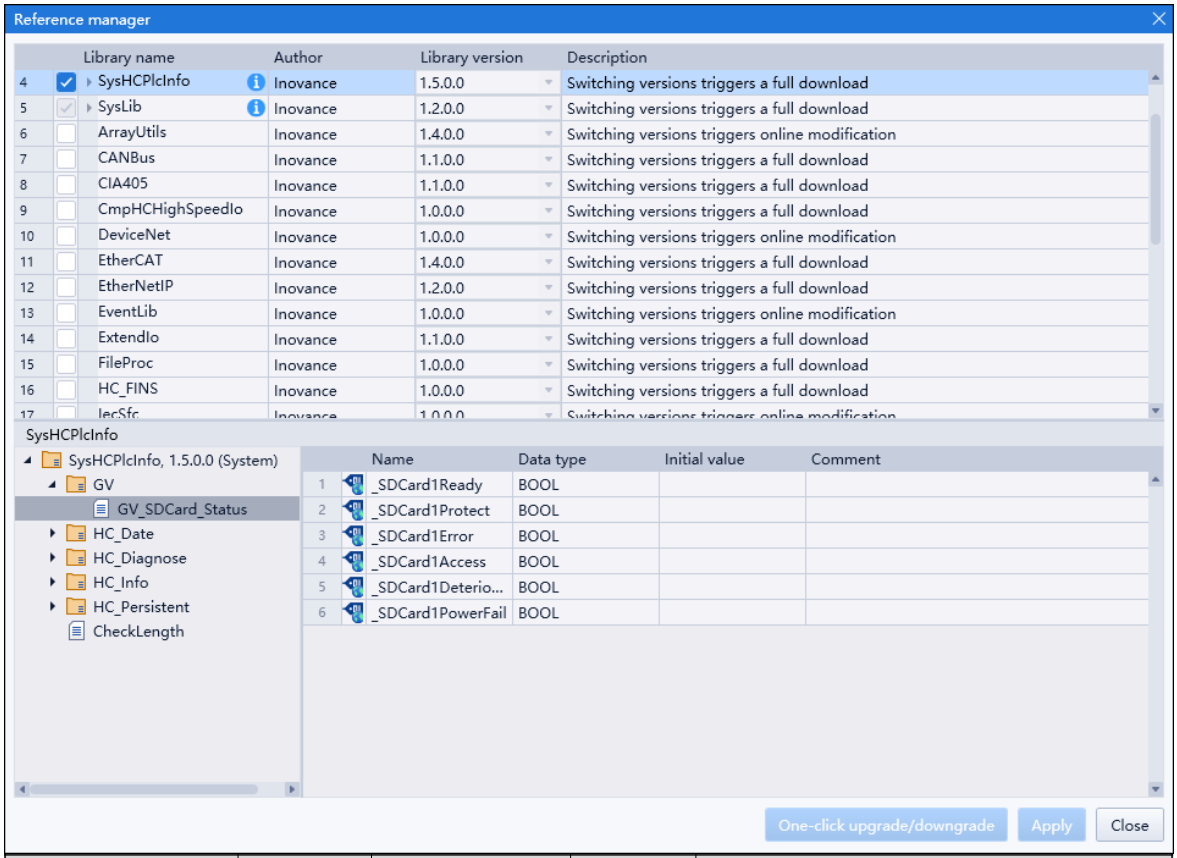
2. Double-click to open the POU interface, select the "SDCard1StatusGet" instruction in the toolbox on the right, double-click it or drag it directly to the program editing area, delete ":=", and create the instruction instance object.



3. Use the GV_SDCard_Status global variable from the SysHCPlcInfo library to get the SD card status.



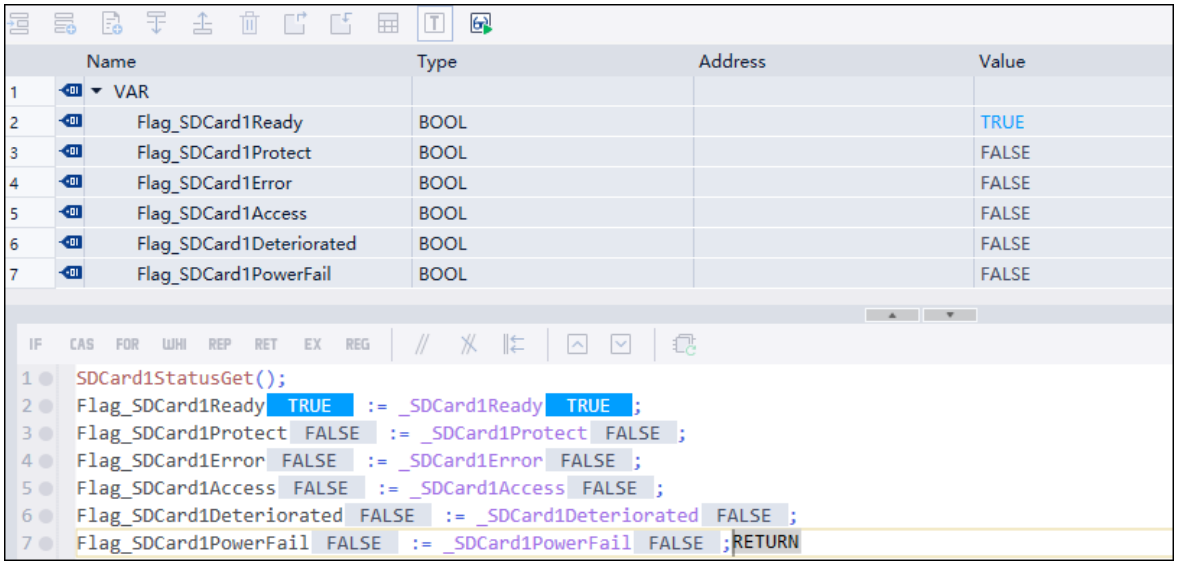
The GV_SDCard_Status global variable is shown in the figure below.



Variable	Data Type	Valid Range	Default Value	Description
_SDCard1Ready	BOOL	TRUE, FALSE	FALSE	TRUE: The SD card is correctly mounted and can be used. FALSE: The SD card is not mounted and cannot be used.

Variable	Data Type	Valid Range	Default Value	Description
_SDCard1Protect	BOOL	TRUE, FALSE	FALSE	TRUE: The SD card is write-protected and files cannot be written. FALSE: The SD card is not write-protected and files can be written.
_SDCard1Error	BOOL	TRUE, FALSE	FALSE	TRUE: The SD card is incorrectly formatted and cannot be used. FALSE: The SD card is correctly formatted and ready for use.
_SDCard1Access	BOOL	TRUE, FALSE	FALSE	TRUE: The SD card is reading/writing files. FALSE: The SD card is idle.
_SDCard1Deteriorated	BOOL	TRUE, FALSE	FALSE	TRUE: The SD card has reached its service life and needs to be replaced. FALSE: The SD card has not reached its service life.
_SDCard1PowerFail	BOOL	TRUE, FALSE	FALSE	Its is a global power-loss-retentive variable. TRUE: The SD card has been inserted, removed, unmounted, or experienced a power loss during use. FALSE: The SD card has not been inserted, removed, unmounted, or experienced a power loss during use.

4. Add the POU to a cyclic task. After the task starts running, check the execution result.



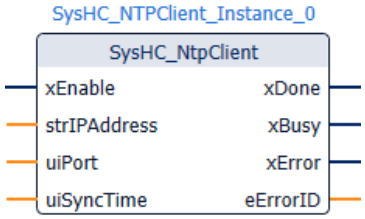
- **Precautions**
  - If you need to obtain the SD card status, call the SDCard1StatusGet() function in a cyclic task, and then the corresponding variable will be updated in the system.

- The _SDCard1Access system variable notifies external systems that the SD card is being accessed. The SD card access status is not reflected in real time, causing incorrect operations of the user program. Therefore, do not use this variable as a flag in the user program.
- The _SDCard1PowerFail system variable notifies external systems that the SD card has been inserted, removed, unmounted, or experienced a power loss during use. The initial value of this flag is FALSE. After the corresponding event is triggered, it will be set to TRUE and will not automatically reset to FALSE. You need to manually set or reset it.
- For devices with multiple USB ports, this instruction only reads the variable corresponding to the USB flash disk status of the first USB port.

18.4.1.34 SysHC_NtpClient

This instruction synchronizes NTP time.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SysHC_NtpClient	NTP time sync	FB		SysHC_NTPClient_Instance_0(xEnable := , strIPAddress := , uiPort := , uiSyncTime := , xDone => , xBusy => , xError => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Function block enable	BOOL	TRUE, FALSE	FALSE	Active High.
	strIPAddress	IP address	STRING	Follows data type	-	IP address
	uiPort	Network port	UINT	Follow data type	123	Network port address
	uiSyncTime	PLC synchronization time	UDINT	10 to 86400	-	PLC synchronization time, unit: s
Output	xDone	Output active	BOOL	TRUE, FALSE	FALSE	Output active.
	xBusy	Instruction executing	BOOL	TRUE, FALSE	FALSE	Executing

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	xError	Fault	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs.
	eErrorID	Fault code	SYS_HC_ERROR	Follows data type	NO_ERROR	See Fault Codes.

- **Function**
  This instruction requests and receives time from the NTP server to calibrate the device time.
- **Program example**
  - Activate the NTP server:
    1. Set and deploy the NTP server parameters including xEnable (value set to TRUE), strIPAddress (actual address of the NTP server), uiPort, and uiSyncTime.
    2. After successful parameter deployment, set R_TRIG to TRUE to activate the NTP server, and the system time will be updated according to the set synchronization time.
  - Modify the NTP parameters:
    1. Set and deploy the NTP server parameters including xEnable (value set to TRUE), strIPAddress (actual address of the NTP server to be modified), uiPort, and uiSyncTime (synchronization time value to be modified).
    2. After successful parameter deployment, set R_TRIG to TRUE to update the NTP parameters, and the system time will be updated according to the new NTP server address and synchronization time.
  - Deactivate the NTP server:
    1. Set xEnable to FALSE.
    2. After successful parameter deployment, set R_TRIG to TRUE to deactivate the NTP server.

    ST:

1		NTP_Enable	BOOL	TRUE	FALSE
2		NTP_IpAddress	STRING	'192.168.1.111'	
3		NTP_Port	UINT	123	
4		Synchronize_Time	UDINT	10	
5		Completion_Flag	BOOL	FALSE	
6		Busy_Flag	BOOL	TRUE	
7		Err_Flag	BOOL	FALSE	
8		Err_Code	SYS_HC_ERROR	NO_ERROR	
9		Trigger	BOOL	FALSE	
10	▸	SysHC_NTPClient_Instan...	SysHC_NtpClient		
11	▸	R_TRIG_instance_0	R_TRIG		
12	▸	F_TRIG_Instance_0	F_TRIG		

IF CAS FOR WHI REP RET EX REG

```
1  R_TRIG_instance_0(CLK FALSE := Trigger FALSE , Q =>);
2  IF R_TRIG_instance_0.Q FALSE THEN
3  SysHC_NTPClient_Instance_0(xEnable TRUE := R_TRIG_instance_0.Q FALSE ,
4                             strIPAddress '192.168.1.111' := NTP_IpAddress '192.168.1.111' ,
5                             uiPort 123 := NTP_Port 123 ,
6                             uiSyncTime 10 := Synchronize_time 10 ,
7                             xDone FALSE => Completion_Flag FALSE ,
8                             xBusy TRUE => Busy_Flag TRUE ,
9                             xError FALSE => Err_Flag FALSE ,
10                            eErrorID NO_ERROR => Err_Code NO_ERROR );
11 Trigger FALSE := FALSE;
12 END_IF
13 RETURN
```

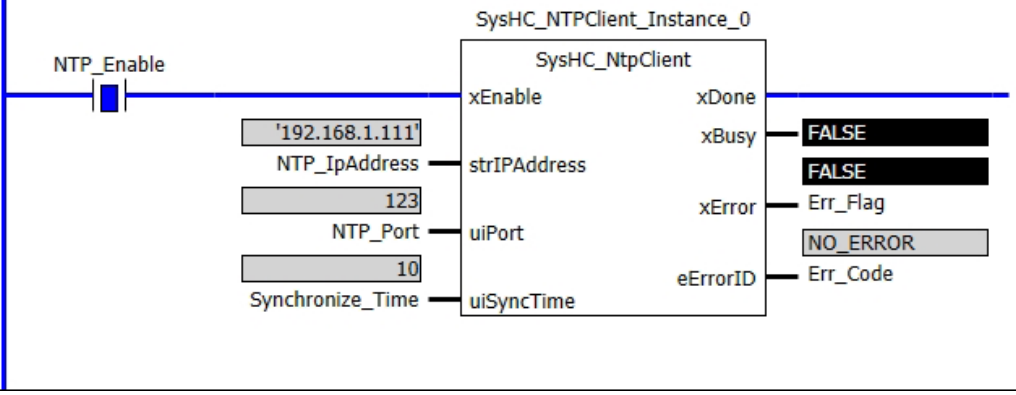
LD:

		Name	Type	Address	Value	Prepared value
1	▸	SysHC_NTPClient_Instan...	SysHC_NtpClient			
2		NTP_Enable	BOOL		TRUE	
3		NTP_IpAddress	STRING		'192.168.1.111'	
4		NTP_Port	UINT		123	
5		Synchronize_Time	UDINT		10	
6		Completion_Flag	BOOL		FALSE	
7		Busy_Flag	BOOL		FALSE	
8		Err_Flag	BOOL		FALSE	
9		Err_Code	SYS_HC_ERROR		NO_ERROR	
10	▸	R_TRIG_Instance_0	R_TRIG			

NOT

1 Program segment 1: ...

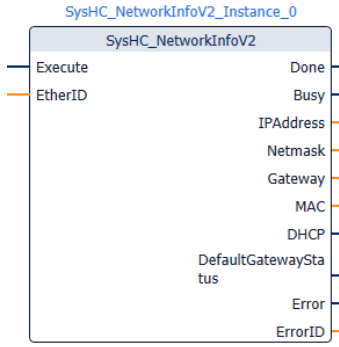
Network Comment



18.4.1.35 SysHC_NetworkInfoV2

This instruction gets the network information of the device, including IP address, subnet mask, gateway, DHCP status, and the enable status of

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SysHC_NetworkInfoV2	Get network information	FB		SysHC_NetworkInfoV2_Instance_0(Execute := , EtherID := , Done => , Busy => , IPAddress => , Netmask => , Gateway => , MAC => , DHCP => , DefaultGatewayStatus => , Error => , ErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Rising edge-triggered	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	EtherID	Network port ID	BYTE	1 ~ 255	0	Network port ID number.
Output	Done	Done	BOOL	TRUE, FALSE	FALSE	Instruction is completed.
	Busy	Instruction executing	BOOL	TRUE, FALSE	FALSE	Instruction is executing.
	IPAddress	IP address	STRING	-	''	IP address.
	Netmask	Subnet mask	STRING	-	''	Subnet mask.
	Gateway	Gateway information	STRING	-	''	Gateway information.
	MAC	MAC address	STRING	-	''	MAC address.
	DHCP	DHCP status	BOOL	TRUE, FALSE	FALSE	DHCP status.
	DefaultGatewayStatus	Default gateway enable status	BOOL	TRUE, FALSE	FALSE	Query whether the default gateway is enabled.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs.
	ErrorID	Fault code	SYS_HC_ERROR	-	NO_ERROR	See Fault Codes.

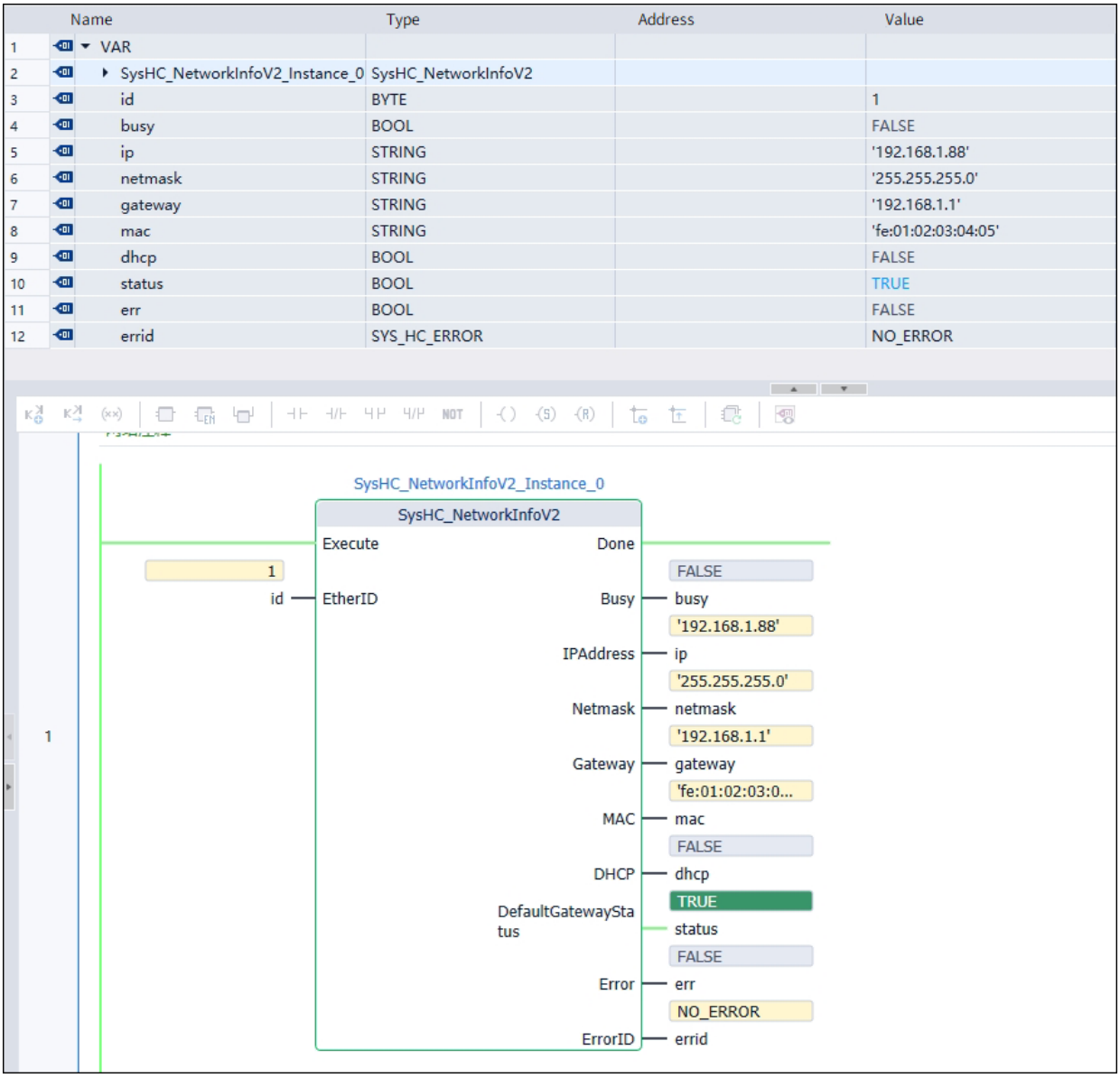
- **Function**
  - Devices that do not support DHCP default to FALSE.
  - The EtherID must correspond to the actual hardware Ethernet port, starting from 1.
- **Program example**

ST:

	Name	Type	Address	Vaule
1	VAR			
2	SysHC_NetworkInfoV2_Instance_0	SysHC_NetworkInfoV2		
3	exe	BOOL		TRUE
4	id	BYTE		1
5	done	BOOL		TRUE
6	busy	BOOL		FALSE
7	ip	STRING		'192.168.1.102'
8	netmask	STRING		'255.255.255.0'
9	gateway	STRING		'192.168.1.1'
10	mac	STRING		'fe:01:02:03:04:05'
11	dhcp	BOOL		FALSE
12	status	BOOL		TRUE
13	err	BOOL		FALSE
14	errid	SYS_HC_ERROR		NO_ERROR

```
IF  CAS  FOR  WHI  REP  RET  EX  REG
1  SysHC_NetworkInfoV2_Instance_0(Execute TRUE  := exe TRUE ,
2                                  EtherID  1   := id  1   ,
3                                  Done TRUE  => done TRUE ,
4                                  Busy FALSE => busy FALSE ,
5                                  IPAddress '192.168.1.102' => ip '192.168.1.102' ,
6                                  Netmask '255.255.255.0' => netmask '255.255.255.0' ,
7                                  Gateway  '192.168.1.1'  => gateway  '192.168.1.1'  ,
8                                  MAC'fe:01:02:03:04:... => mac'fe:01:02:03:04:...,
9                                  DHCP FALSE => dhcp FALSE ,
10                                 DefaultGatewayStatus TRUE  => status TRUE ,
11                                 Error FALSE => err FALSE ,
12                                 ErrorID NO_ERROR => errid NO_ERROR );RETURN
```

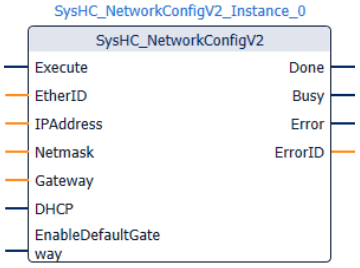
LD:



18.4.1.36 SysHC_NetworkConfigV2

This instruction configures the network information of the device, including IP address, subnet mask, gateway, and the enable/disable status of the default gateway.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SysHC_NetworkConfigV2	Configure network properties	FB		SysHC_NetworkConfigV2_Instance_0(Execute := , EtherID := , IPAddress := , Netmask := , Gateway := , DHCP := , EnableDefaultGateway := , Done => , Busy => , Error => , ErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Rising edge-triggered	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	EtherID	Network port ID	BYTE	1 ~ 255	0	Network port ID.
	IPAddress	IP address	STRING	-	''	IP address.
	Netmask	Subnet mask	STRING	-	''	Subnet mask.
	Gateway	Gateway information	STRING	-	''	Gateway information.
	DHCP	DHCP status	BOOL	TRUE, FALSE	FALSE	DHCP status.
	EnableDefaultGateway	Enable default gateway	BOOL	TRUE, FALSE	FALSE	Used to enable or disable the default gateway.
Output	Done	Done	BOOL	TRUE, FALSE	FALSE	Instruction is completed.
	Busy	Instruction executing	BOOL	TRUE, FALSE	FALSE	Instruction is executing.
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs.
	ErrorID	Fault code	SYS_HC_ERROR	-	NO_ERROR	See Fault Codes.

- **Function**
  - This instruction currently does not support setting DHCP.
  - The EtherID must correspond to the actual hardware Ethernet port, starting from 1 and so on.

- **Program example**

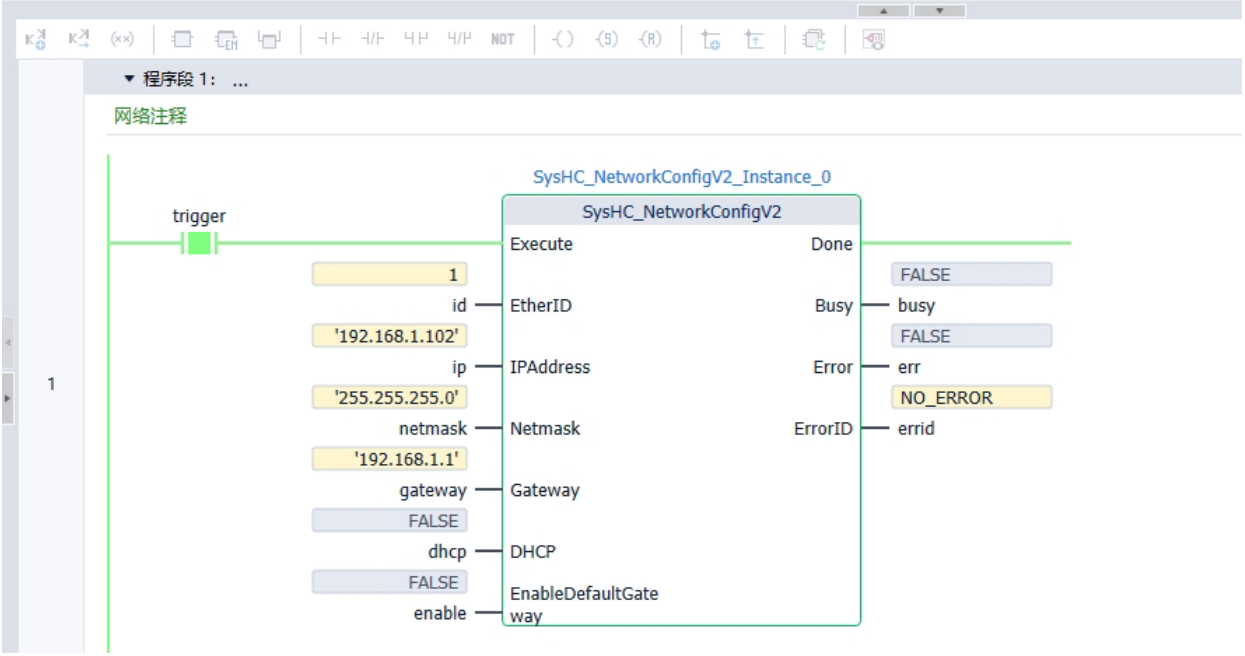
ST:

1	VAR			
2	SysHC_NetworkConfigV2_Instan...	SysHC_NetworkConfigV2		
3	exe	BOOL		TRUE
4	id	BYTE		1
5	ip	STRING		'192.168.1.101'
6	netmask	STRING		'255.255.255.0'
7	gateway	STRING		'192.168.1.1'
8	dhcp	BOOL		FALSE
9	enable	BOOL		TRUE
10	done	BOOL		TRUE
11	busy	BOOL		FALSE
12	err	BOOL		FALSE
13	errid	SYS_HC_ERROR		NO_ERROR

```
SysHC_NetworkConfigV2_Instance_0(Execute TRUE := exe TRUE,
                                 EtherID 1 := id 1,
                                 IPAddress '192.168.1.101' := ip '192.168.1.101',
                                 Netmask '255.255.255.0' := netmask '255.255.255.0',
                                 Gateway '192.168.1.1' := gateway '192.168.1.1',
                                 DHCP FALSE := dhcp FALSE,
                                 EnableDefaultGateway TRUE := enable TRUE,
                                 Done TRUE => done TRUE,
                                 Busy FALSE => busy FALSE,
                                 Error FALSE => err FALSE,
                                 ErrorID NO_ERROR => errid NO_ERROR );RETURN
```

LD:

1	VAR			
2	SysHC_NetworkConfigV2_Instan...	SysHC_NetworkConfigV2		
3	id	BYTE		1
4	ip	STRING		'192.168.1.102'
5	netmask	STRING		'255.255.255.0'
6	gateway	STRING		'192.168.1.1'
7	busy	BOOL		FALSE
8	err	BOOL		FALSE
9	errid	SYS_HC_ERROR		NO_ERROR
10	dhcp	BOOL		FALSE
11	enable	BOOL		FALSE
12	trigger	BOOL		TRUE



18.4.1.37 Fault Codes

- SYS_HC_ERROR

Fault code	Value	Description
NO_ERROR	0	No error occurred
WRONG_PARAMETER	1	Incorrect range of the input parameters; incorrect format of the input parameters
FAILURE_OBTAIN	2	Parameter obtainment failed; non-compliant parameter obtainment
ETHER_ID_ERROR	3	Network port ID out of range
Com_ID_ERROR	4	Serial port ID out of range
Slave_ID_ERROR	5	Slave ID input out of range
Slave_IP_ERROR	6	Slave IP out of range
FAILURE_SET	7	Parameter setting failed
UNKNOWN_ERROR	8	Unknown error
INVAILD_IP	9	Invalid configured IP
PARAMETER_ERROR	16	Invalid configured parameter
GET_TIMEZONE_ERROR	18	Time zone source file missing
SET_TIMEZONE_ERROR	19	Time zone source file missing
TIMEZONE_PARAM_ERROR	20	Invalid configured time zone
PARAM_ERROR_TASKNAME	21	Sampling task error
PARAM_ERROR_INTERVAL	22	Sampling interval exceeds the upper/lower limit
PARAM_ERROR_DURATION	23	Sampling duration exceeds the upper/lower limit
FUNCTION_NOT_SUPPORT	24	Some instructions are not supported under simulation mode
PARAM_ERROR_FILEMAX	25	Max. number of record files exceeds the upper/lower limit
PARAM ERROR_TRIGGER	26	Trigger signal error
PARAM_ERROR_DATAPARAM	27	Sampling data error
PARAM_ERROR_SCALE	28	Sampling ratio error
PARAM_ERROR_DATALEN	29	Length of record data exceeds the limit
NO SPACE	30	Insufficient remaining disk space
DOUBLE_TRIG	31	Overlapping enable signals of function block
WRITEFILE_ERROR	32	Failed to write record files
OCCUPIEDSPACE OVERFLOW	33	Recording file storage exceeds the limit
GETVARINFO ERROR	34	Failed to get parameter information
GETDATATYPE_ERROR	35	Failed to get parameter type

Fault code	Value	Description
NO_SUPPORTED_DATATYPE	36	Data type is not supported
GETVARADDR_ERROR	37	Failed to get parameter address
NO_TARGET_FILE	38	No target file
NO_READ_ACCESS	39	No read permission for variable

- PERSISTENT_ERROR

Fault code	Value	Description
NO_ERROR	0	No fault
PERSIS_NO_APP	1	The application startup order is not first.
PERSIS_NO_XML	2	No XML file is running.
PERSIS_NO_BAK	3	No backup XML file available.
PERSIS_APP_MATCH	4	The application and XML file do not match.
PERSIS_MRETAIN_VER	5	The data backup version is V1.
ERROR_FB_TIMEOUT	6	Failed to refresh fb variables.
PERSIS_SYS_ERROR	7	Internal system runtime error.
PERSIS_TIME_DELAY	8	Repeat operation prohibited within 10 seconds.
PERSIS_RUN_CONF	9	Data backup read/write operation conflict.
PERSIS_PARSE_ERROR	10	XML parsing error.
PERSIS_CRC_ERROR	11	XML CRC error.

18.4.2 Redundancy Information

18.4.2.1 Instruction List

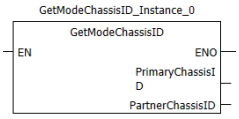
Instruction	FB/FC	Function	Library	Applicable Library Version
18.4.2.2 GetModeChassisID	FB	Obtain the rack number of redundant device	Redundancy	Library version 1.0.0.0 and later
18.4.2.3 GetRunStopKeyMatchState	FB	Obtain the status of whether the redundant rack run/stop key positions match		Library version 1.0.0.0 and later
18.4.2.4 GetSystemRedundancyMode	FB	This instruction obtains the primary and partner modes of the redundant rack.		Library version 1.0.0.0 and later
18.4.2.5 GetSystemRedundancyState	FB	Obtain the redundancy confirmation status of the redundant rack		Library version 1.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.4.2.6 GetControllerModuleDiagnosisInformation	FB	Obtain basic diagnostic information of the controller module	Diagnosis	Library version 1.0.0.0 and later
18.4.2.7 GetRedundancyModuleDiagnosisInformation	FB	This instruction obtains the basic diagnostic information of the redundancy module.		Library version 1.0.0.0 and later
18.4.2.8 GetTemperature	FB	Obtain the ambient temperature of the controller		Library version 1.0.0.0 and later

18.4.2.2 GetModeChassisID

This instruction obtains the rack number of redundant device.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetModeChassisID	Obtain the rack number of redundant device	FB		GetModeChassisID _Instance_0(PrimaryChassisID => , PartnerChassisID =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	PrimaryChassisID	Primary module rack number	SINT	-1, 1, 2	0	-1: Failed to get primary module rack number. 1: Rack A. 2: Rack B.

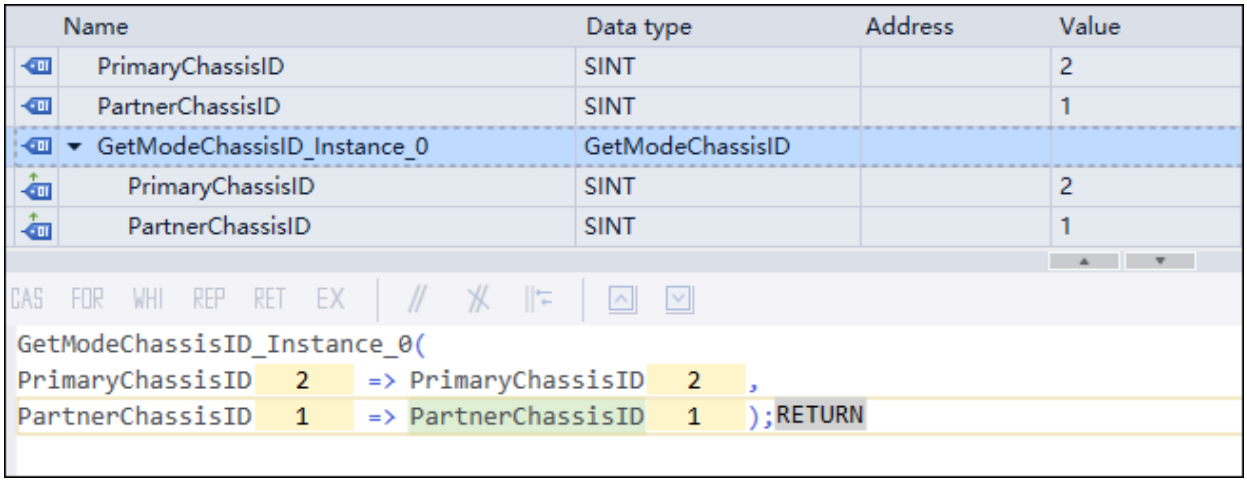
Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	PartnerChassisID	Redundant module rack number	SINT	-1, 1, 2, 3	0	-1: Failed to get redundant module rack number. 1: Rack A. 2: Rack B. 3: Non-redundant project or redundancy not established.

- **Function**

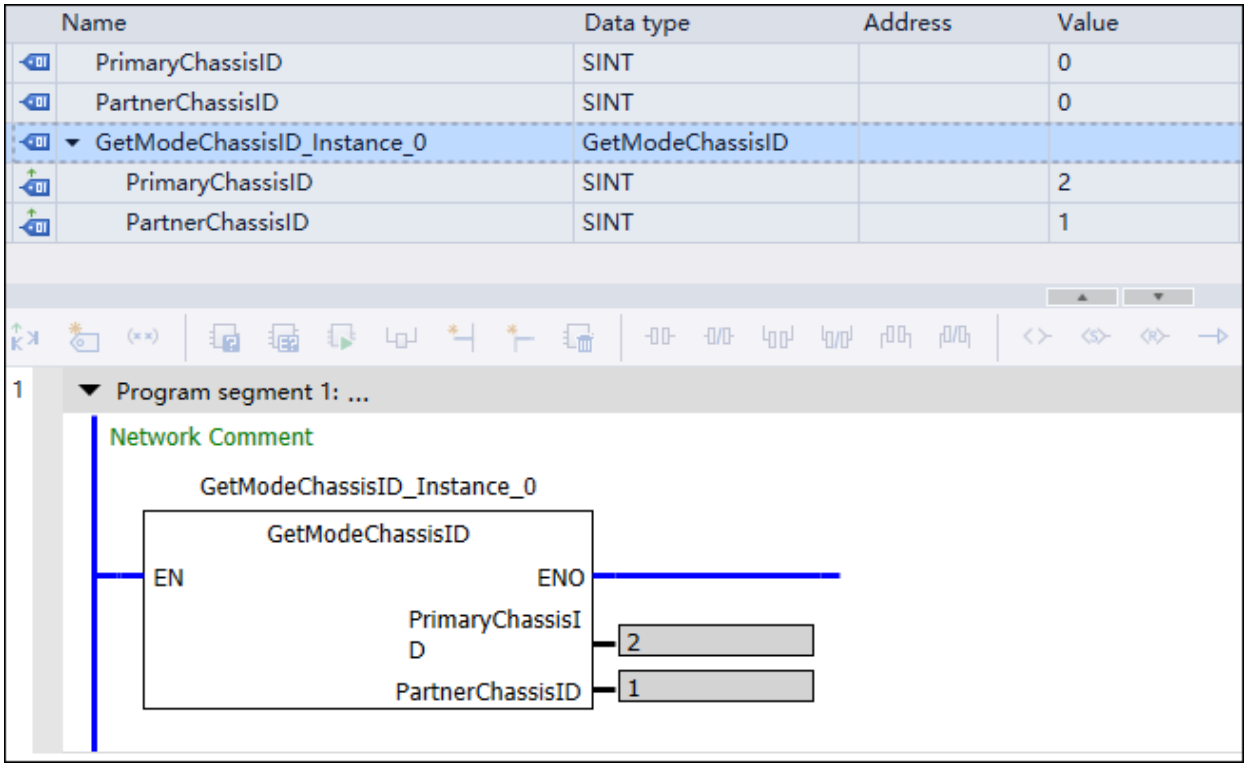
  The program starts to obatin the primary module rack number and the redundant module rack number when it is powered on and running.

- **Program example**

  ST:



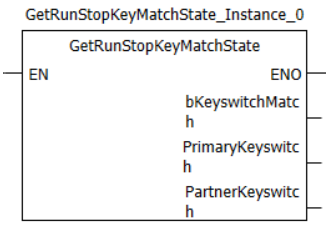
  LD:



18.4.2.3 GetRunStopKeyMatchState

This instruction obtains the status of whether the redundant rack run/stop key positions match.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetRunStopKeyMatchState	Obtain the status of whether the redundant rack run/stop key positions match	FB		GetRunStopKeyMatchState_Instance_0(bKeyswitchMatch => , PrimaryKeyswitch => , PartnerKeyswitch =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	bKeyswitchMatch	Whether the key positions of the dual racks match	BYTE	0, 1 , , 255	0	0: Key positions not matching. 1: Key positions matching. 255: Redundant mode not established or non-redundant project.
	PrimaryKeyswitch	Switch position of primary module	BYTE	1, 2	0	1: Run. 2: Stop.

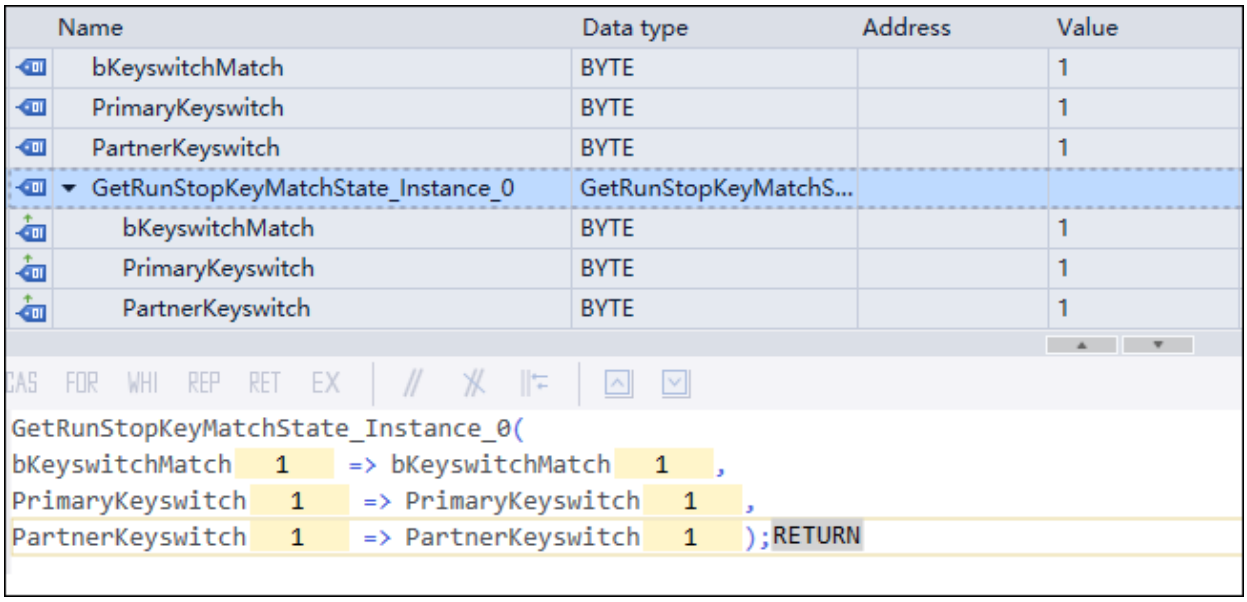
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	PartnerKeysw itch	Key position of backup module	BYTE	1, 2, 255	0	1: Run. 2: Stop. 255: Redundant mode not established or non-redundant project.

- **Function**

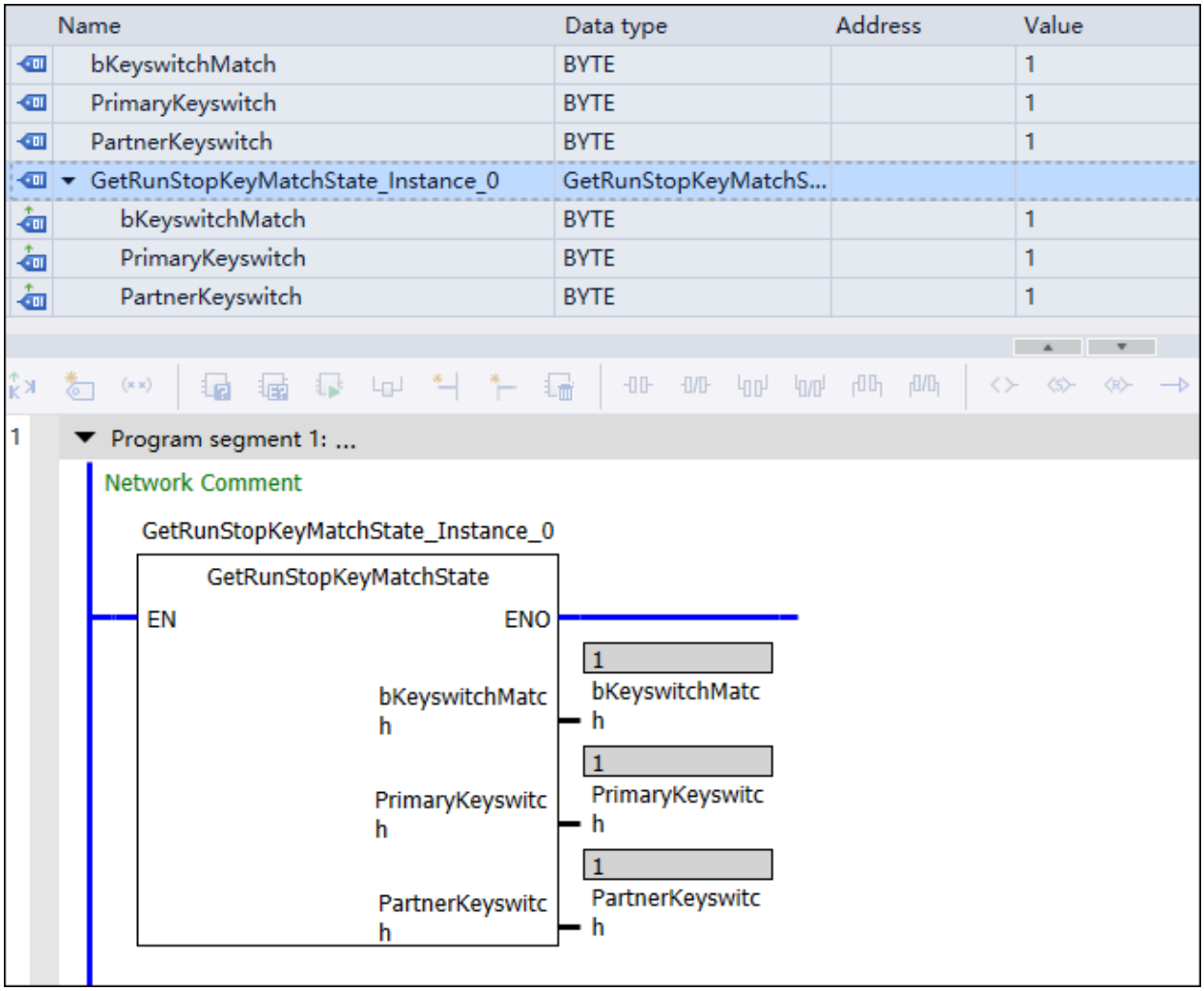
  The program starts module acquisition of the run/stop key positions and compares them for consistency when powered on.

- **Program example**

  ST:



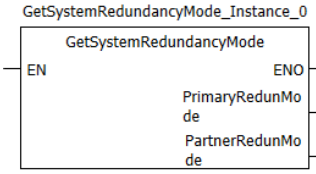
  LD:



18.4.2.4 GetSystemRedundancyMode

This instruction obtains the primary and partner modes of the redundant rack.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetSystemRedundancyMode	This instruction obtains the primary and partner modes of the redundant rack.	FB		GetSystemRedundancyMode_Instance_0(PrimaryRedunMode => , PartnerRedunMode =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	PrimaryRedunMode	Primary rack redundancy mode	BYTE	2, 3, 5	0	2: Primary. 3: Backup 5: Fault mode.

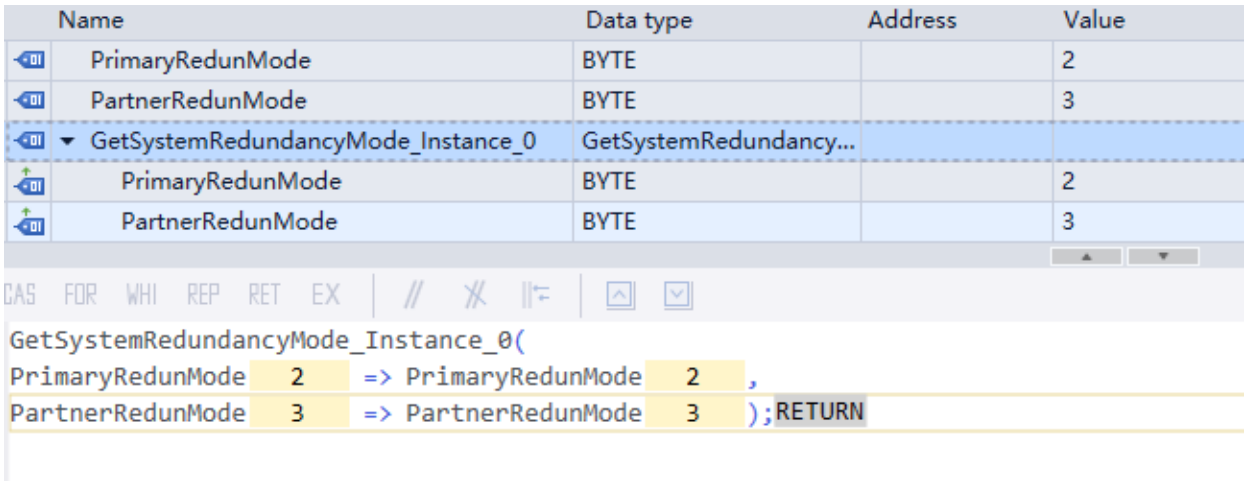
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	PartnerRedunMode	Backup rack redundancy mode	BYTE	2, 3, 5, 255	0	2: Primary. 3: Backup 5: Fault mode. 255: Obtainment timeout

- **Function**

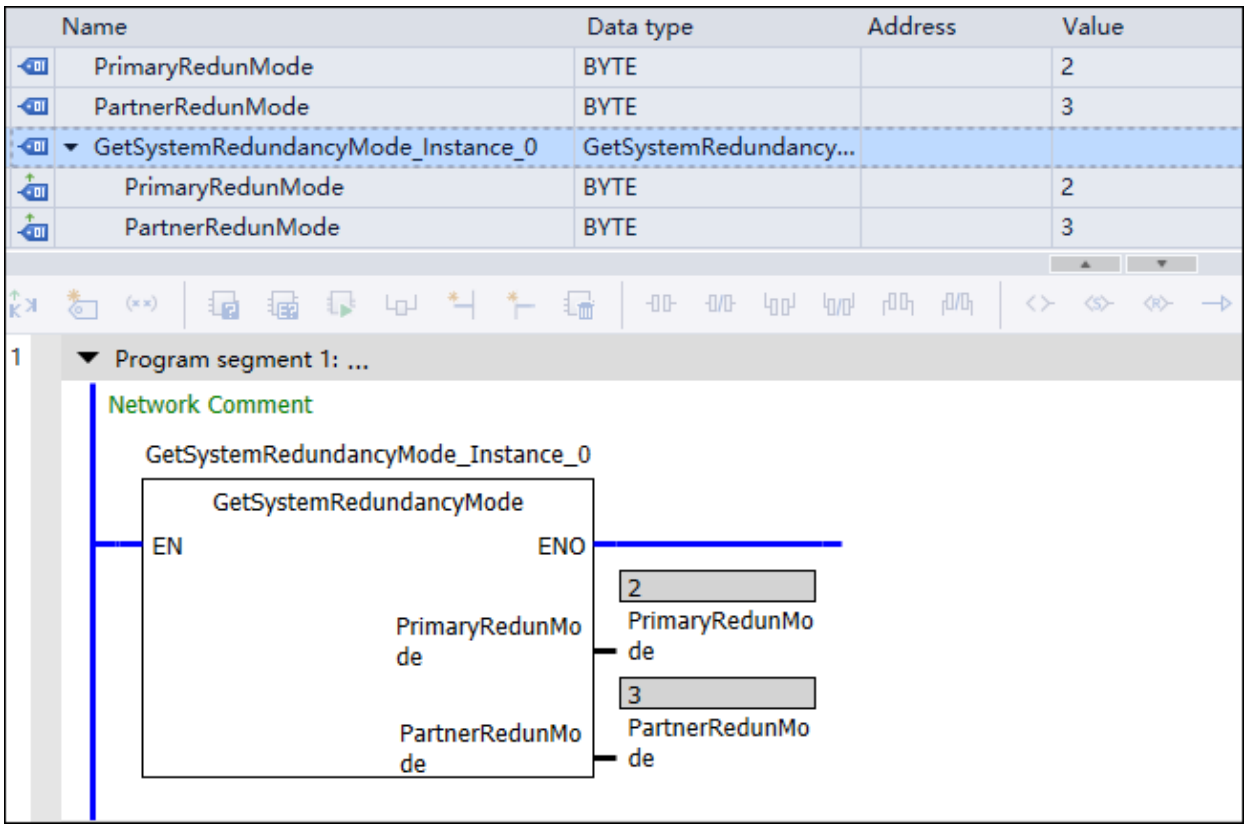
  The program starts to obtain the primary-partner status of the racks upon power-on and operation.

- **Program example**

  ST:



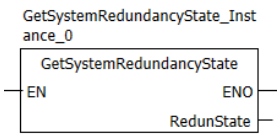
  LD:



18.4.2.5 GetSystemRedundancyState

This instruction obtains the redundancy confirmation state of the redundant rack.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetSystemRedundancyState	Obtain the redundancy confirmation status of the redundant rack	FB		GetSystemRedundancyState_Instance_0(RedunState =>);

- **Related variables**

  Variable

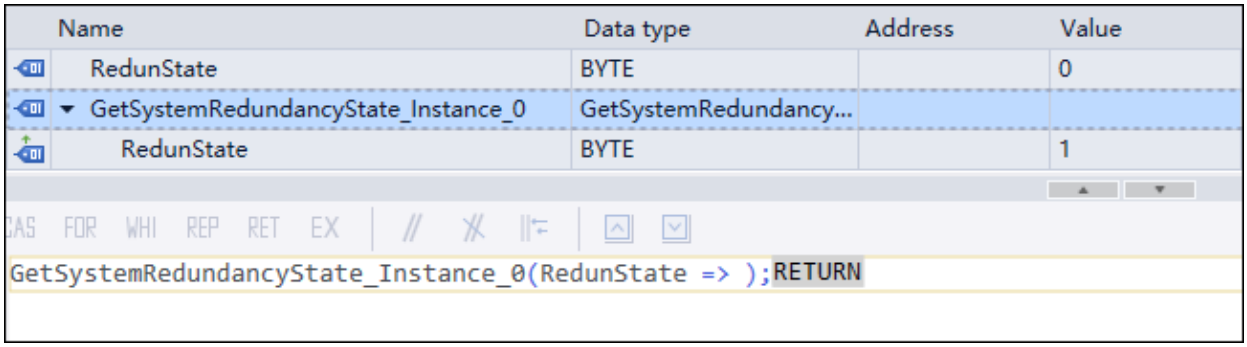
Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	RedunState	Redundancy state	BYTE	0, 1, 2	0	0: Redundancy not confirmed. 1: Redundancy confirmed. 2: Redundancy confirming.

- **Function**

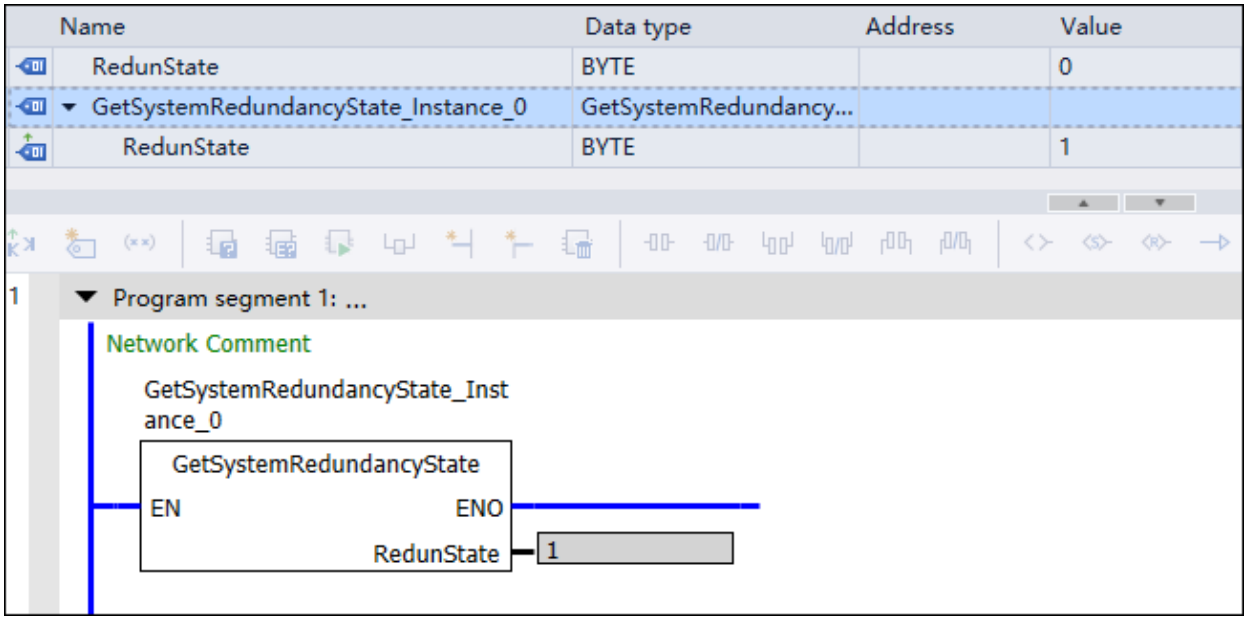
  The program starts to obtain the state of redundant rack as soon as it is powered on and running.

- **Program example**

  ST:



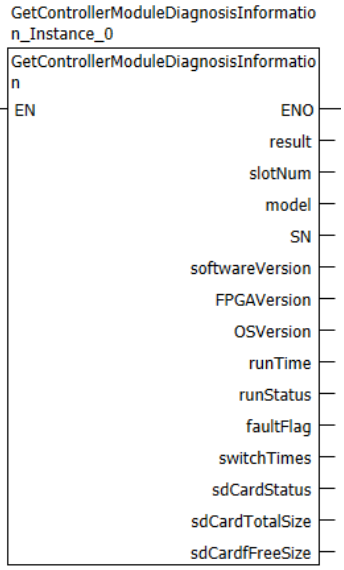
  LD:



18.4.2.6 GetControllerModuleDiagnosisInformation

This instructiuon obtains basic diagnostic information of the controller module.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetControllerModuleDiagnosisInformation	Obtain basic diagnostic information of the controller module	FB		GetControllerModuleDiagnosisInformation_Instance_0(result => , slotNum => , model => , SN => , softwareVersion => , FPGAVersion => , OSVersion => , runTime => , runStatus => , faultFlag => , switchTimes => , sdCardStatus => , sdCardTotalSize => , sdCardfFreeSize =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	result	Result obtained	BOOL	TRUE, FALSE	FALSE	FALSE: Obtainment successful. TRUE: Obtainment not supported.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	slotNum	Slot number	BYTE	0, 1, 2, 3, 4	0	The slot number of the controller module on the backplate.
	model	Model	STRING	Follows data type	''	Controller module model.
	SN	Serial No.	STRING	Follows data type	''	Controller serial number.
	softwareVersion	Software version	STRING	Follows data type	''	Software version number.
	FPGAVersion	FPGA version	STRING	Follows data type	''	FPGA version number.
	OSVersion	OS version	STRING	Follows data type	''	OS version number.
	runTime	Running time	LWORD	Follows data type	0	Controller running time in us.
	runStatus	Running status	BYTE	0, 1, 2, 3, 4, 5	0	Running status of controller: 0: None. 1: Run. 2: Stop. 3: Breakpoint commissioning. 4: Single-step debugging. 5: Single run.
	faultFlag	Fault flag of user program	BOOL	TRUE, FALSE	FALSE	Fault flag of user program: FALSE: Normal. TRUE: Error
	switchTimes	Primary-backup switch times	DWORD	Depends on data type	0	Primary-backup switch times.
	sdCardStatus	SD card status	BYTE	0, 1, 2, 5	0	SD card status: 0: Default value. 1: Inserted. 2: Not inserted. 5: In normal use.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	sdCardTotalSize	SD card size	LWORD	Follows data type	0	Total SD card size in bytes.
	sdCardfFreeSize	SD card available size	LWORD	Follows data type	0	Available SD card size in bytes.

- **Function**

  This instruction is used to obtain the runtime status information of some basic parameters of the controller.

- **Program example**

  ST:

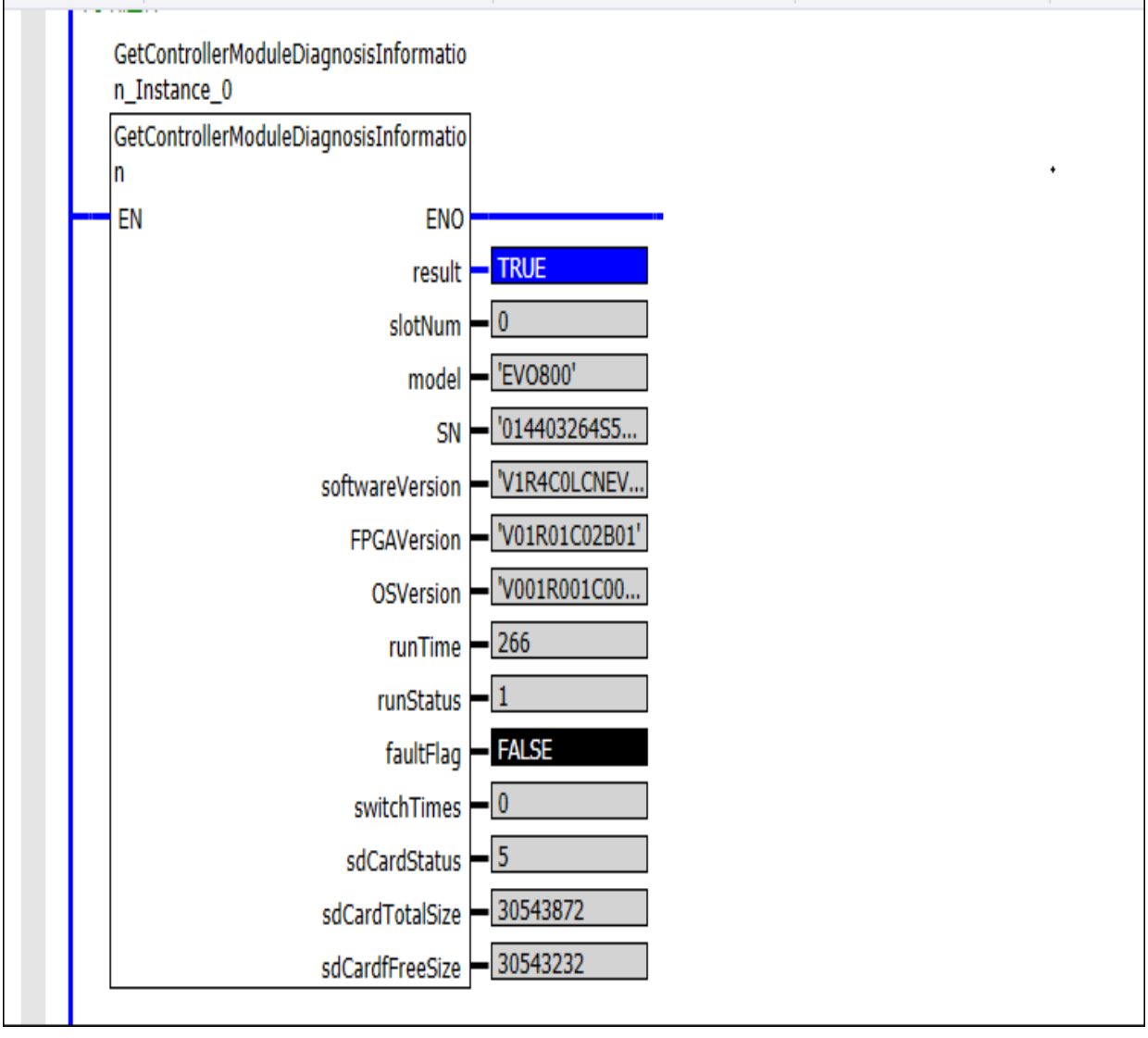
Name	Data type	Address	Value
GetControllerModuleDiagnosisInformatio...	GetControllerModuleDi...		
result	BOOL		TRUE
slotNum	BYTE		0
model	STRING		'EVO800'
SN	STRING		'014403264S510693'
softwareVersion	STRING		'V1R4C0LCNEVO80021B7'
FPGAVersion	STRING		'V01R01C02B01'
OSVersion	STRING		'V001R001C001SP3LCNAM1900001B006_20250403_3327'
runTime	LWORD		100
runStatus	BYTE		1
faultFlag	BOOL		FALSE
switchTimes	DWORD		0
sdCardStatus	BYTE		5
sdCardTotalSize	LWORD		30543872
sdCardfFreeSize	LWORD		30543232

```
CAS FOR WHI REP RET EX // //* |:=

GetControllerModuleDiagnosisInformation_Instance_0(
result => ,
slotNum => ,
model => ,
SN => ,
softwareVersion => ,
FPGAVersion => ,
OSVersion => ,
runTime => ,
runStatus => ,
faultFlag => ,
switchTimes => ,
sdCardStatus => ,
sdCardTotalSize => ,
sdCardfFreeSize => );
```

  LD:

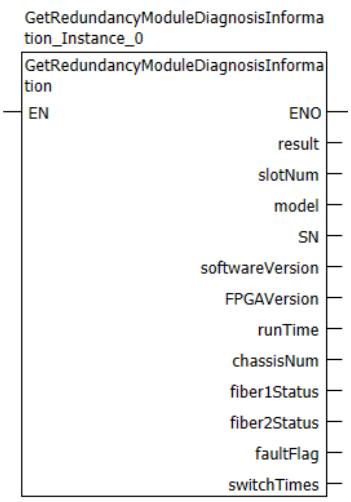
Name	Data type	Address	Value	Prepare value
GetControllerModuleDiagnosisInformatio...	GetControllerModuleDi...			
result	BOOL		TRUE	
slotNum	BYTE		0	
model	STRING		'EVO800'	
SN	STRING		'01440326455...	
softwareVersion	STRING		'V1R4C0LCNE...	
FPGAVersion	STRING		'V01R01C02B01'	
OSVersion	STRING		'V001R001C00...	
runTime	LWORD		266	
runStatus	BYTE		1	
faultFlag	BOOL		FALSE	
switchTimes	DWORD		0	
sdCardStatus	BYTE		5	
sdCardTotalSize	LWORD		30543872	
sdCardfFreeSize	LWORD		30543232	



18.4.2.7 GetRedundancyModuleDiagnosisInformation

This instruction obtains the basic diagnostic information of the redundancy module.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetRedundancyModuleDiagnosisInformation	This instruction obtains the basic diagnostic information of the redundancy module.	FB		GetRedundancyModuleDiagnosisInformation_Instance_0(result => , slotNum => , model => , SN => , softwareVersion => , FPGAVersion => , runTime => , chassisNum => , fiber1Status => , fiber2Status => , faultFlag => , switchTimes =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	result	Result obtained	BOOL	TRUE, FALSE	FALSE	FALSE: Obtainment successful. TRUE: Obtainment not supported.
	slotNum	Slot number	BYTE	0, 1, 2, 3, 4	0	The slot number of the controller module on the backplate.
	model	Model	STRING	Follows data type	''	Controller module model.
	SN	Serial No.	STRING	Follows data type	''	Controller serial number.
	softwareVersion	Software version	STRING	Follows data type	''	Software version number.
	FPGAVersion	FPGA version	STRING	Follows data type	''	FPGA version number.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	runTime	Running time	LWORD	Follows data type	0	Controller running time in us.
	chassisNum	Chassis number	BYTE	0, 1, 2	0	Chassis number: 0: Default value. 1: A chassis. 2: B chassis.
	fiber1Status	Optical fiber 1 status	BYTE	0, 1 to 6, 8, 9	0	Optical fiber 1 status: 0: Default value. 1 to 6: Fault. 8: In use. 9: Standby.
	fiber2Status	Optical fiber 2 status	BYTE	0, 1 to 6, 8, 9	0	Fiber 2 status: 0: Default value. 1 to 6: Fault. 8: In use. 9: Standby.
	faultFlag	Fault flag of user program	BYTE	0, 1	0	Program fault flag: 0: Normal 1: Error
	switchTimes	Primary-backup switch times	DWORD	Depends on data type	0	Primary-backup switch times.

- **Function**

  It is available only in redundant mode to obtain the basic diagnostic information of the redundant module.

- **Program example**

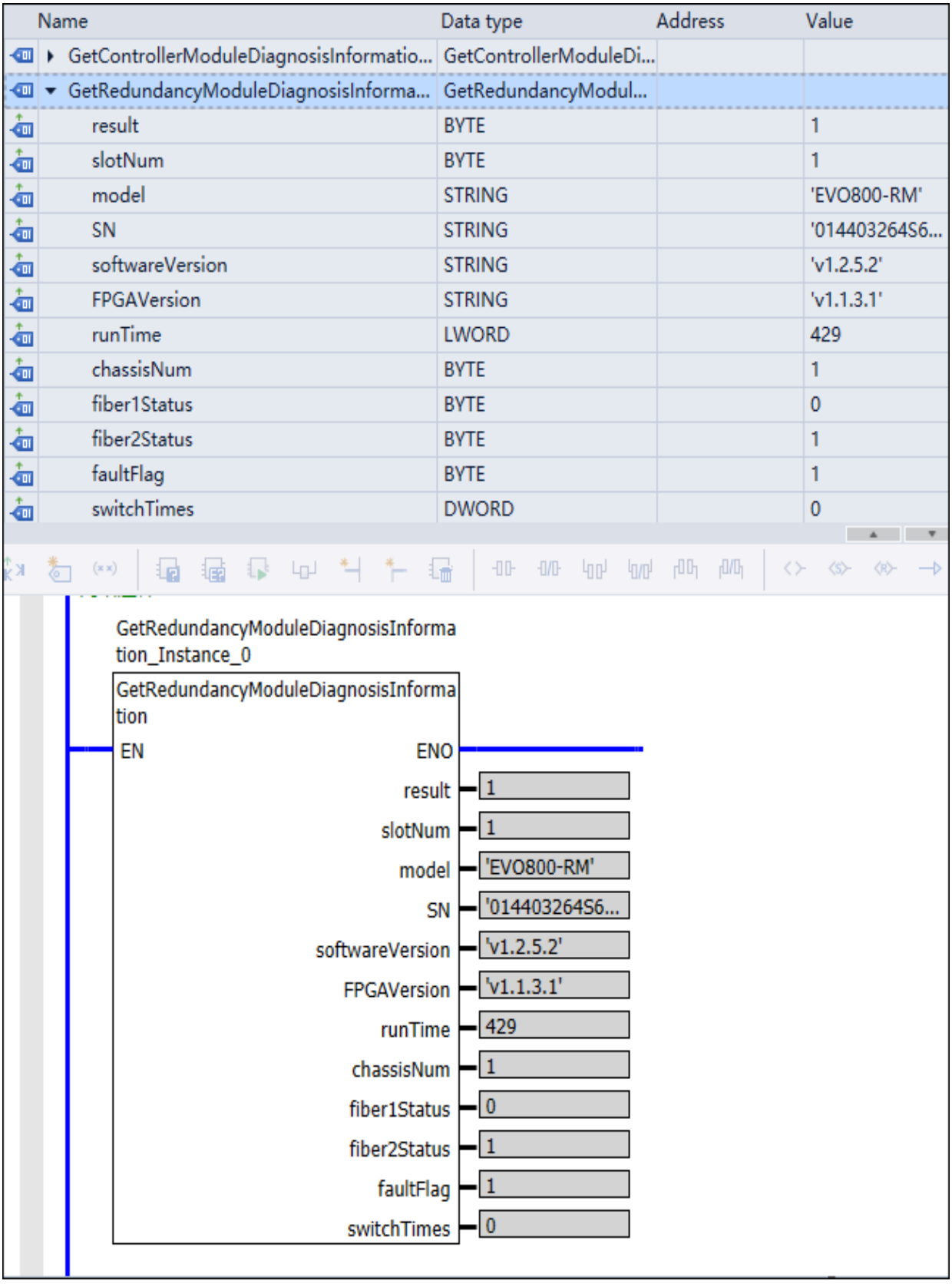
  ST:

Name	Data type	Address	Value
GetControllerModuleDiagnosisInformatio...	GetControllerModuleDi...		
GetRedundancyModuleDiagnosisInforma...	GetRedundancyModul...		
result	BYTE		1
slotNum	BYTE		1
model	STRING		'EVO800-RM'
SN	STRING		'014403264S612345'
softwareVersion	STRING		'v1.2.5.2'
FPGAVersion	STRING		'v1.1.3.1'
runTime	LWORD		408
chassisNum	BYTE		1
fiber1Status	BYTE		0
fiber2Status	BYTE		1
faultFlag	BYTE		1
switchTimes	DWORD		0

CAS FOR WHI REP RET EX // /* */

```
GetRedundancyModuleDiagnosisInformation_Instance_0(
result => ,
slotNum => ,
model => ,
SN => ,
softwareVersion => ,
FPGAVersion => ,
runTime => ,
chassisNum => ,
fiber1Status => ,
fiber2Status => ,
faultFlag => ,
switchTimes => );
```

LD:

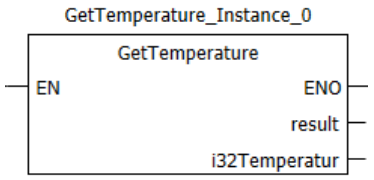


Name	Data type	Address	Value
GetControllerModuleDiagnosisInformatio...	GetControllerModuleDi...		
GetRedundancyModuleDiagnosisInforma...	GetRedundancyModul...		
result	BYTE		1
slotNum	BYTE		1
model	STRING		'EVO800-RM'
SN	STRING		'01440326456...
softwareVersion	STRING		'v1.2.5.2'
FPGAVersion	STRING		'v1.1.3.1'
runTime	LWORD		429
chassisNum	BYTE		1
fiber1Status	BYTE		0
fiber2Status	BYTE		1
faultFlag	BYTE		1
switchTimes	DWORD		0

18.4.2.8 GetTemperature

This instruction obtains the ambient temperature of the controller.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
GetTemperature	Obtain the ambient temperature of the controller	FB		GetTemperature_Instance_0(result => , i32Temperatur =>);

- **Related variables**

  Variable

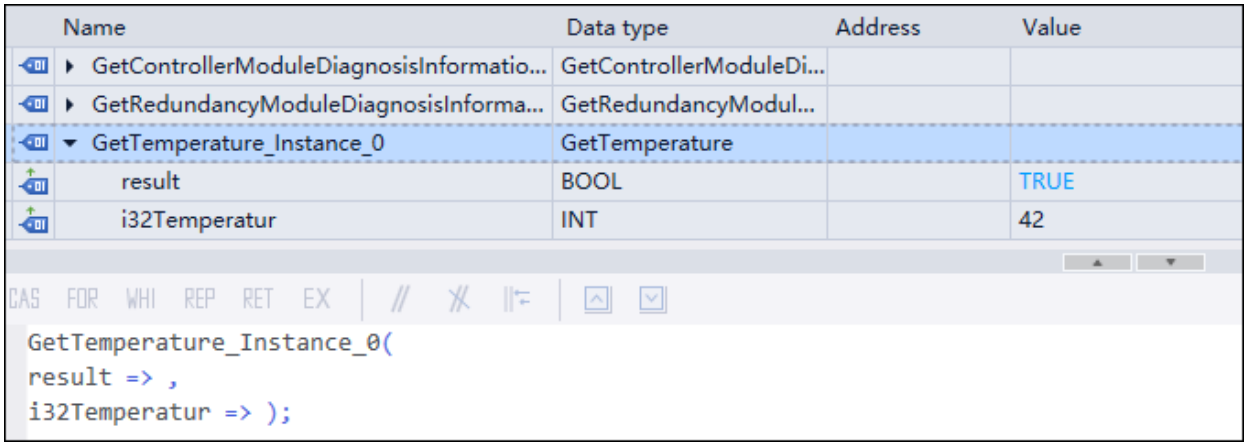
Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	result	Result obtained	BOOL	TRUE, FALSE	FALSE	FALSE: Obtainment successful. TRUE: Obtainment not supported.
	i32Temperatur	Temperature	INT	Follows data type	0	Ambient temperature of the controller.

- **Function**

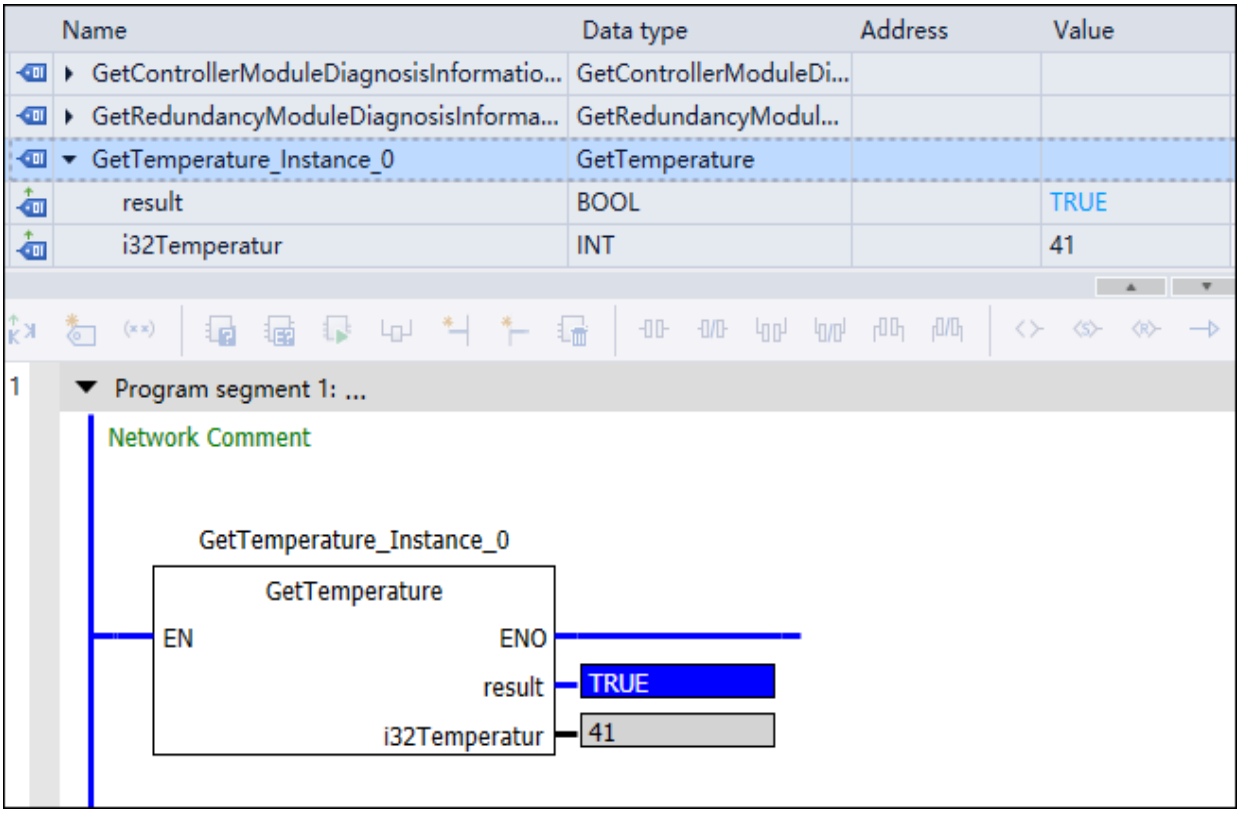
  This instruction obtains the ambient temperature of the controller.

- **Program example**

  ST:



  LD:



18.4.3 EtherNetIP

18.4.3.1 Instruction List

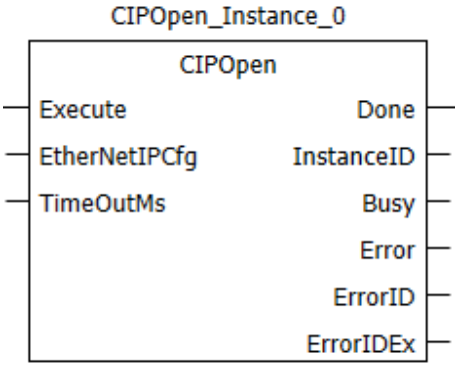
Instruction	FB/FC	Function	Library	Applicable Library Version
18.4.3.2 CIPOpen	FB	Establish EIP Class-3 connection	EtherNetIP	Library version 1.0.0.0 and later
18.4.3.3 CIPClose	FB	Close EIP Class 3 connection		Library version 1.0.0.0 and later
18.4.3.4 CIPOpenWithDataSize	FB	Create a Class 3 connection with a specified size		Library version 1.0.0.0 and later
18.4.3.5 CIPRead	FB	Read variable		Library version 1.0.0.0 and later
18.4.3.6 CIPWrite	FB	Write variable		Library version 1.0.0.0 and later
18.4.3.7 CIPSend	FB	Send Class3 display message		Library version 1.0.0.0 and later
18.4.3.8 generic_Service	FB	EIP generic service		Library version 1.2.0.0 and later
18.4.3.9 Get_Attributes_All	FB	Get all attributes of a specific instance		Library version 1.2.0.0 and later
18.4.3.10 Get_Attribute_Single	FB	Get specified attributes of a specific instance		Library version 1.2.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.4.3.11 Set_Attributes_All	FB	Set all attributes of a specific instance		Library version 1.2.0.0 and later
18.4.3.12 Set_Attribute_Single	FB	Set a specified attribute of a specific instance		Library version 1.2.0.0 and later
18.4.3.13 GetEIPSystemstats	FB	Get diagnostic information of EIP system		Library version 1.2.0.0 and later
18.4.3.14 GetEIPConnectstats	FB	Get diagnostic information of EIP connection		Library version 1.2.0.0 and later
18.4.3.15 EIP_SetRemAdapterConfIP	FB	Set the connection configuration IP address of the remote adapter		Library version 1.2.0.0 and later
18.4.3.16 SysHC_EthernetIPDeviceDiagnose	FB	Obtain the fault diagnostics of the EIP slave	SysHCPlcInfo	Library version 1.0.0.0 and later
18.4.3.17 SysHC_EthernetIPSlaveDiagnose	FB	Obtain the fault diagnostics of the EIP master		Library version 1.0.0.0 and later

18.4.3.2 CIPOpen

This instruction specifies the corresponding slave IP and the Ethernet port ID to be triggered, and establish an EIP connection.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CIPOpen	Establish EIP Class-3 connection	FB		CIPOpen_Instance_0(Execute := , EtherNetIPCfg := , TimeOutMs := , Done => , InstanceID => , Busy => , Error => , ErrorID => , ErrorIDEx =>);

- **Related variables**

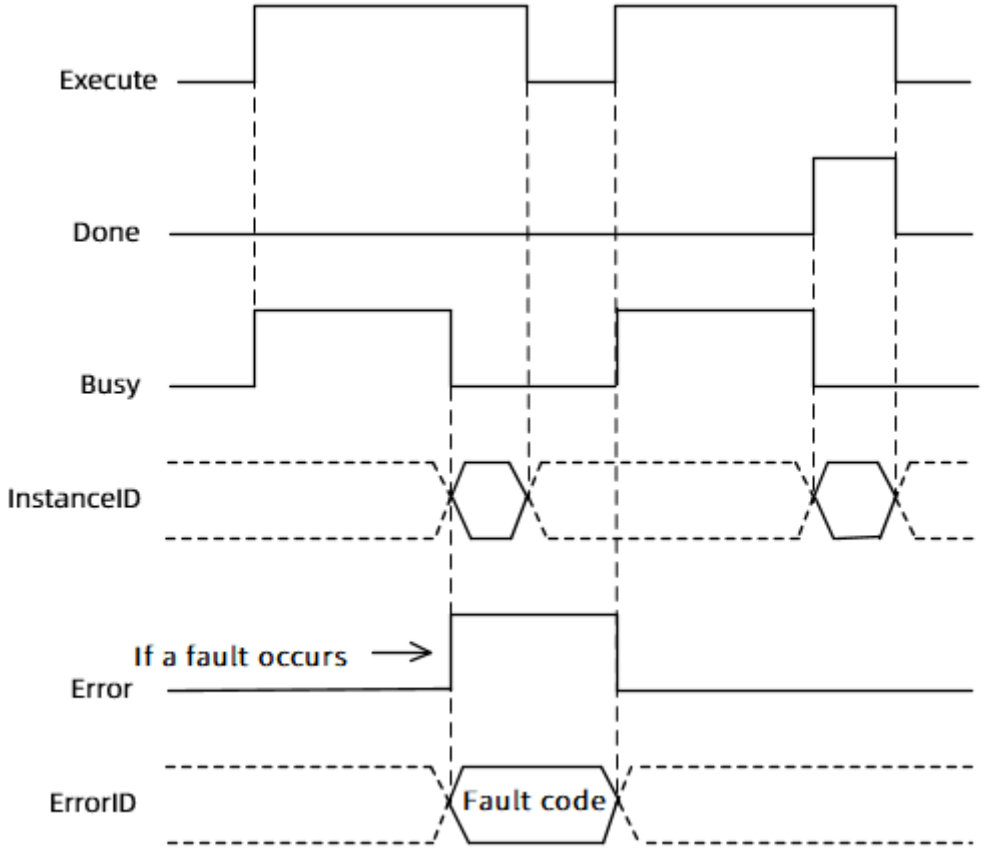
  Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Rising edge-triggered	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	EtherNetIPCfg	IP configuration and Ethernet port designation for the connected slave	EtherNet IP_Cfg	-	0	IP configuration and Ethernet port designation for the connected slave.
	TimeOutMs	Connection timeout period	UINT	20 ms to 5000ms	20	Connection timeout period.
Output	Done	Done	BOOL	TRUE, FALSE	FALSE	Instruction is completed.
	InstanceID	Created instance number	DINT	961 to 992	0	The instance number is an automatically generated variable by the program, and manually input values are invalid.
	Busy	Executing	BOOL	TRUE, FALSE	FALSE	Instruction is executing.
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	Error status.
	ErrorID	General fault code	WORD	Follow data type	0	General fault code. See 18.4.3.19 EIP Standard Fault Codes.
	ErrorIDEx	Extended fault code	DWORD	Depends on data type	0	Extended fault code. For details, see 18.4.3.19 EIP Standard Fault Codesand 18.4.3.20 Fault Codes.

- **Function description**
  - This instruction specifies the corresponding slave IP and the Ethernet port ID to be triggered, and establish an EIP connection.
  - The data type of EtherNetIPCfg is the structure EtherNetIP_Cfg, whose meaning is as follows.

Variable	Name	Data type	Initial Value	Description
LocalEtherNetID	Ethernet port ID	BYTE	0	Ethernet port ID: LAN1 = 1, LAN2 = 2, and so on.
RemoteAddress	IP configuration structure	ARRAY [0..3] OF BYTE	0	IP configuration follows standard IP specifications, and errors are reported after occurrence.

- **Timing diagram**



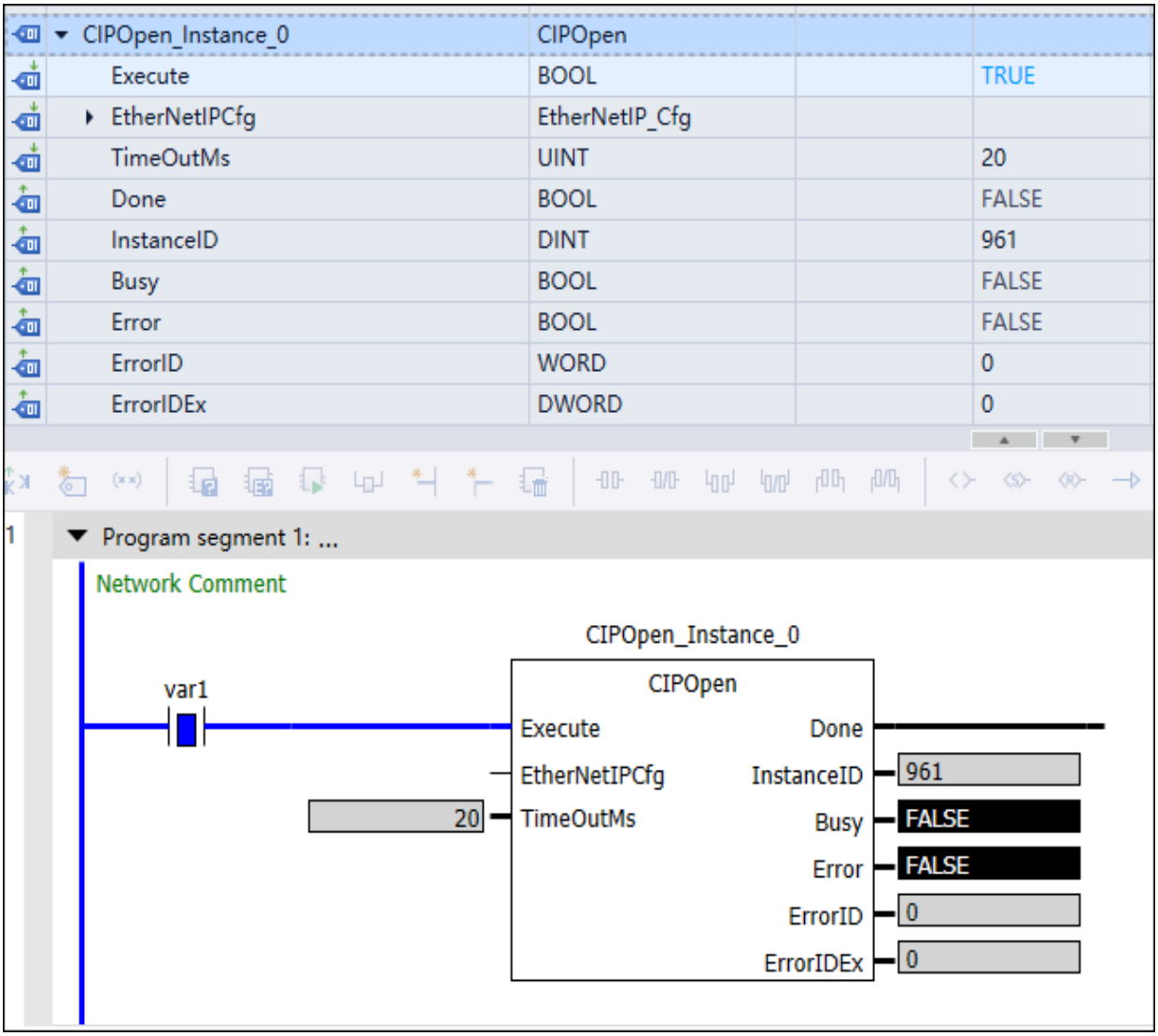
- **Program example**

  ST:

Name	Data type	Address	Value
CIPOpen_Instance_0	CIPOpen		
Execute	BOOL		TRUE
EtherNetIPCfg	EtherNetIP_Cfg		
LocalEtherNetID	BYTE		2
RemoteAddress	ARRAY [0..3] OF BYTE		
RemoteAddress[0]	BYTE		192
RemoteAddress[1]	BYTE		168
RemoteAddress[2]	BYTE		2
RemoteAddress[3]	BYTE		66
TimeOutMs	UINT		20
Done	BOOL		FALSE
InstanceID	DINT		963
Busy	BOOL		FALSE
Error	BOOL		FALSE
ErrorID	WORD		0
ErrorIDEx	DWORD		0
CIPOpenWithDataSize_Instance_0	CIPOpenWithDataSize		
CIPRead_Instance_0	CIPRead		

```
CIPOpen_Instance_0(
Execute TRUE := var1 TRUE ,
EtherNetIPCfg := ,
TimeOutMs := ,
Done => ,
InstanceID => ,
Busy => ,
Error => ,
ErrorID => ,
ErrorIDEx => );
```

LD:

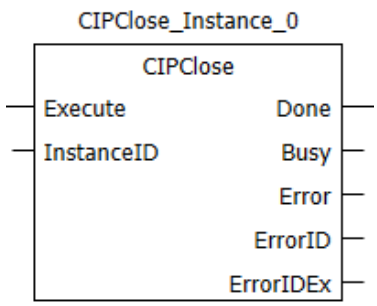


- **Precautions**
  - IP configuration follows standard IP specifications, and errors are reported after occurrence.
  - InstanceID represents the instance number of a successfully opened connection, supporting up to 32 connections.
  - Each time the CIPOpen instruction is successfully executed, an available InstanceID is generated, and the newly generated InstanceID increments sequentially. If the instruction execution fails, the available InstanceID generated by the last successfully executed instruction is returned.
  - ErrorID and ErrorIDEX are general fault codes and extended status codes, indicating faults at the time of connection.
  - When the EtherNetIPCfg and TimeOutMs settings are out of specification range, the Error bit is set.
  - The maximum amount of data accessible via CIP is 81920 bytes. If this limit is exceeded, fault code 61461 ("Requested data exceeding the specified size") will be reported.

18.4.3.3 CIPClose

This instruction specifies the corresponding instance ID to close this connection.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CIPClose	Close EIP Class 3 connection	FB		CIPClose_Instance_0(Execute := , InstanceID := , Done => , Busy => , Error => , ErrorID => , ErrorIDEx =>);

- **Related variables**

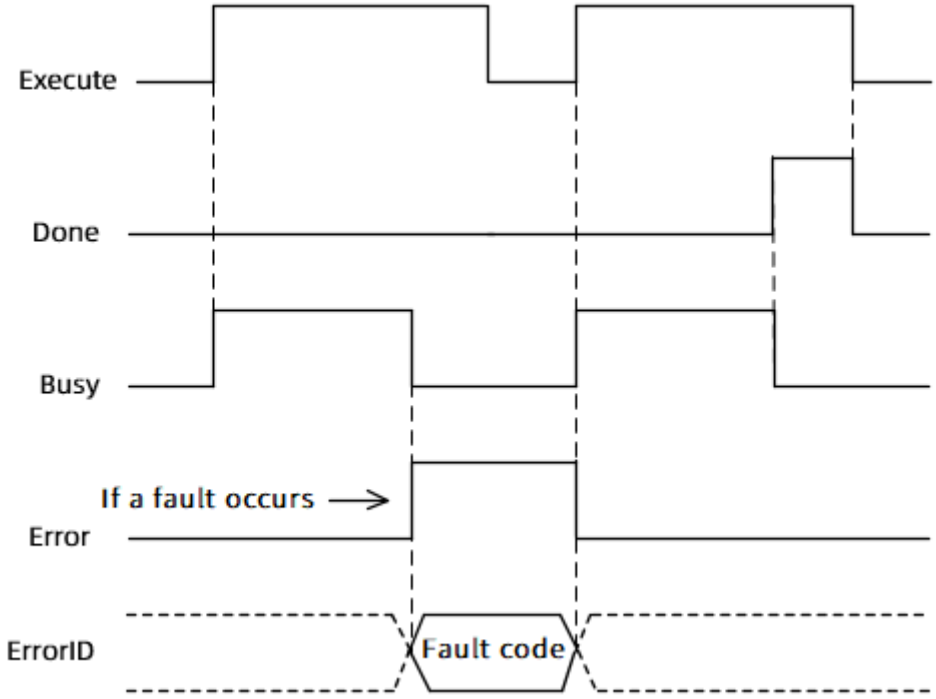
  Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Rising edge-triggered	BOOL	TRUE, FALSE	FALSE	SF_R_TRIG is a rising edge-triggered instruction.
	InstanceID	InstanceID of a connection window	DINT	InstanceID associated with CIPOpen output	0	InstanceID of a connection window.
Output	Done	Done	BOOL	TRUE, FALSE	FALSE	Instruction is completed.
	Busy	Executing	BOOL	TRUE, FALSE	FALSE	Instruction is executing.
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	Error status.
	ErrorID	General fault code	WORD	Follow data type	0	General fault code. See 18.4.3.19 EIP Standard Fault Codes.
	ErrorIDEx	Extended fault code	DWORD	Depends on data type	0	Extended fault code. For details, see 18.4.3.19 EIP Standard Fault Codesand 18.4.3.20 Fault Codes.

- **Function description**

  This instruction specifies the corresponding instance ID to close this connection.

- **Sequence diagram**



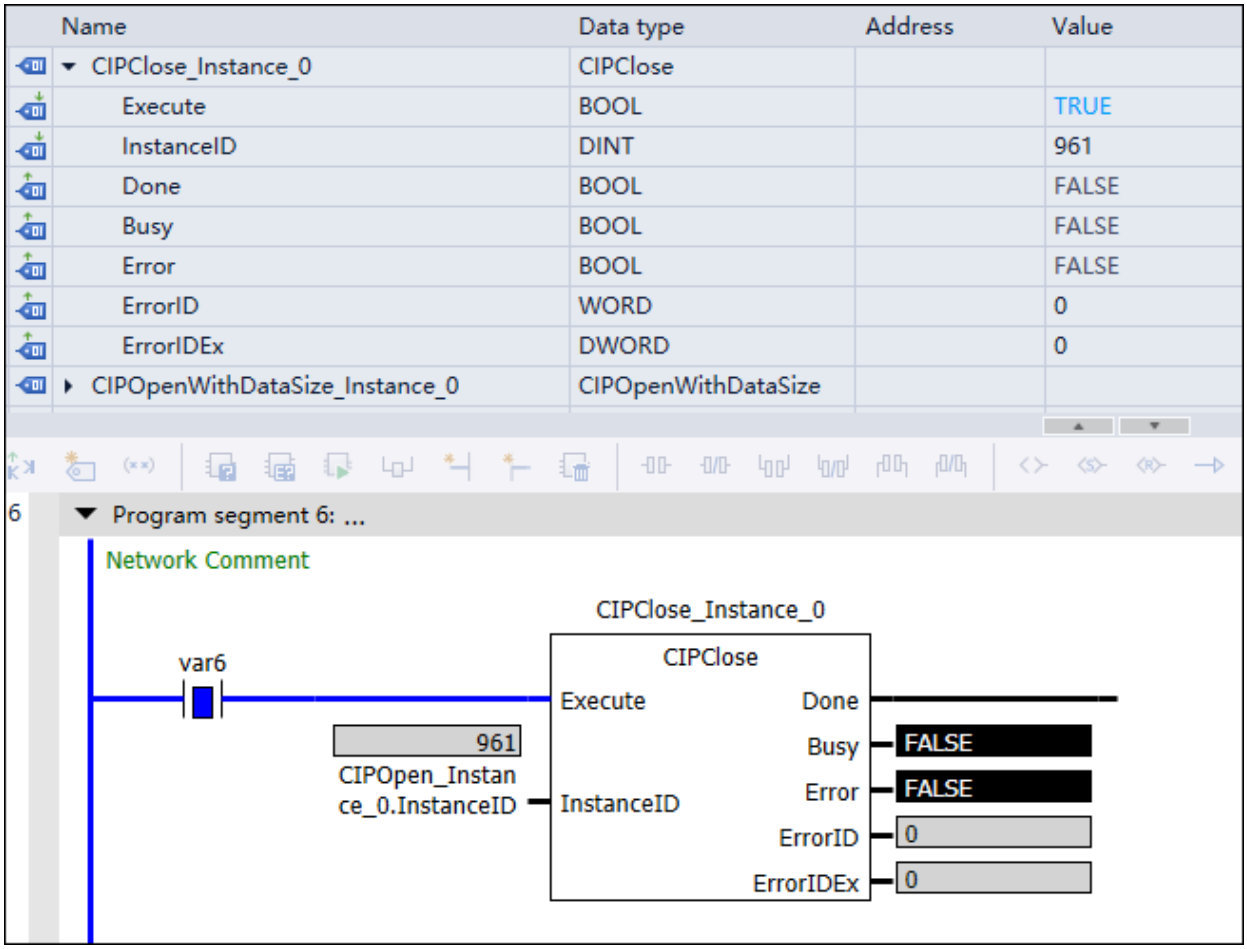
- **Program example**

  ST:

Name	Data type	Address	Value
CIPClose_Instance_0	CIPClose		
Execute	BOOL		TRUE
InstanceID	DINT		963
Done	BOOL		FALSE
Busy	BOOL		FALSE
Error	BOOL		FALSE
ErrorID	WORD		0
ErrorIDEx	DWORD		0
var1	BOOL		TRUE

```
CIPClose_Instance_0(
Execute  TRUE  := var6  TRUE  ,
InstanceID  963  := CIPOpen_Instance_0.InstanceID  963  ,
Done => ,
Busy => ,
Error => ,
ErrorID => ,
ErrorIDEx => );RETURN
```

  LD:

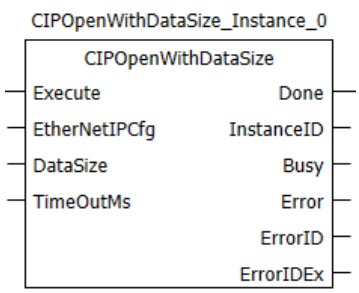


- **Precautions**
  - If the InstanceID parameter setting is out of specification or not created, the Error bit is set.
  - The instance ID created by CIPOpen cannot be reused after CIPClose is executed. Reuse will trigger fault code 61446 connection error.

18.4.3.4 CIPOpenWithDataSize

This instruction specifies the corresponding slave IP, the Ethernet port ID to be triggered, and the tag size to establish an EIP connection.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CIPOpenWithDataSize	Create a Class 3 connection with a specified size	FB		CIPOpenWithDataSize_Instance_0(Execute := , EtherNetIPCfg := , DataSize := , TimeOutMs := , Done => , InstanceID => , Busy => , Error => , ErrorID => , ErrorIDEx =>);

- **Related variables**

  Variable

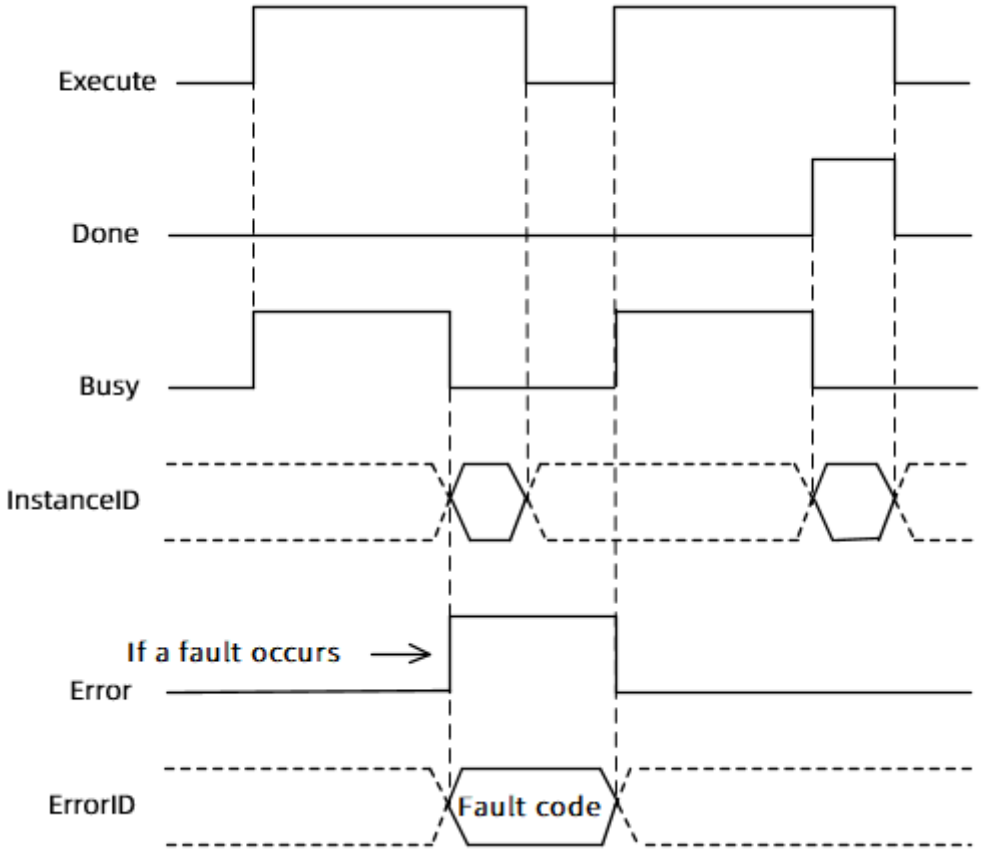
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Rising edge-triggered	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	EtherNetIPCfg	IP configuration and Ethernet port designation for the connected slave	EtherNetIP_Cfg	-	0	IP configuration and Ethernet port designation for the connected slave.
	DataSize	Connection window size	WORD	1 to 65481	1	Connection window size.
	TimeOutMs	Connection timeout period	UINT	20 to 5000	20	Connection timeout period.
Output	Done	Done	BOOL	TRUE, FALSE	FALSE	Instruction is completed.
	InstanceID	Created instance number	DINT	961 to 992	0	Created instance number.
	Busy	Executing	BOOL	TRUE, FALSE	FALSE	Instruction is executing.
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	Error status.
	ErrorID	General fault code	WORD	Follow data type	0	General fault code. See 18.4.3.19 EIP Standard Fault Codes.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	ErrorIDEx	Extended fault code	DWORD	Depends on data type	0	Extended fault code. For details, see 18.4.3.19 EIP Standard Fault Codesand 18.4.3.20 Fault Codes.

- **Function description**

  This instruction specifies the corresponding slave IP, the Ethernet port ID to be triggered, and the tag size to establish an EIP connection.

- **Sequence diagram**



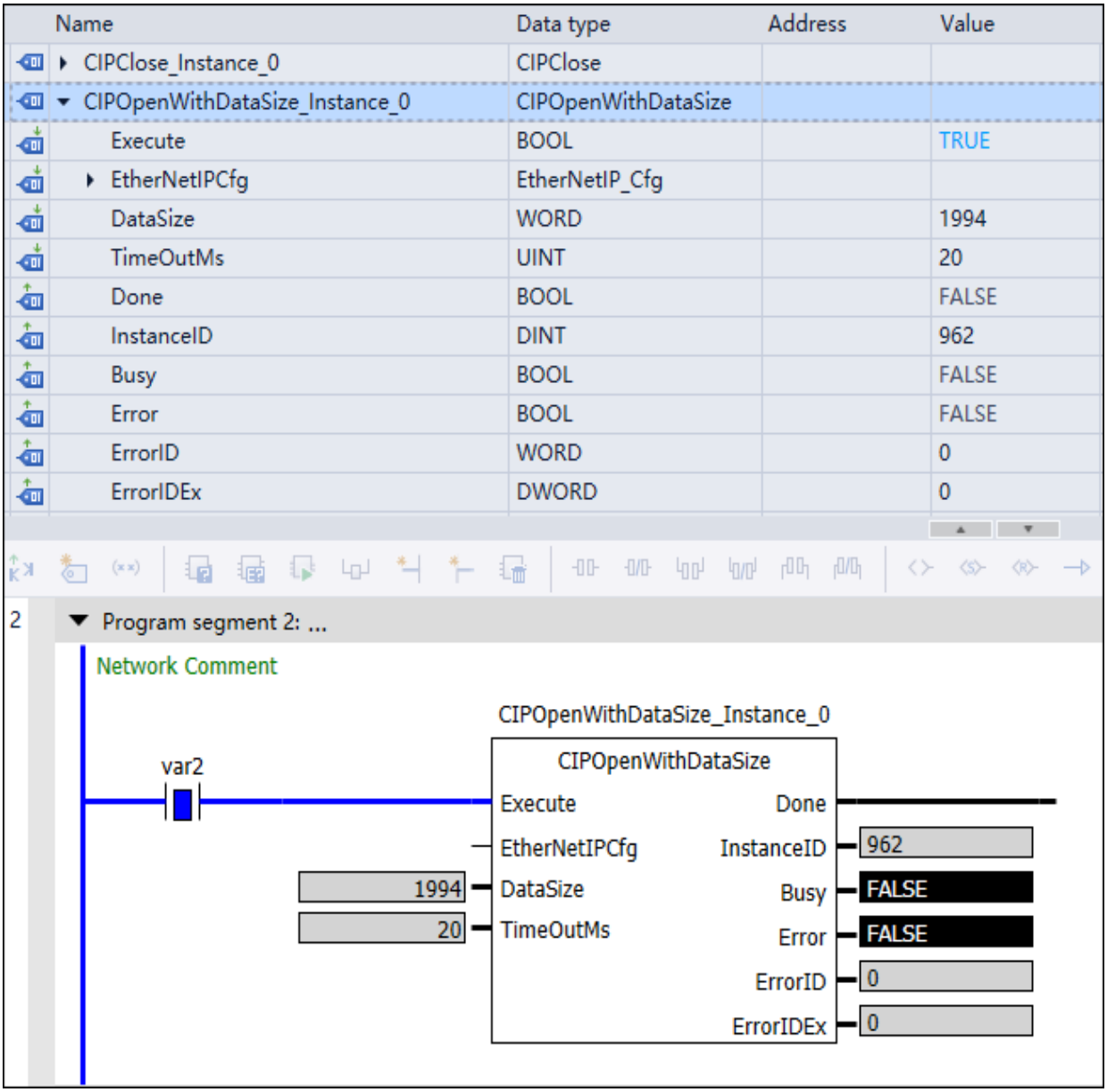
- **Program example**

  ST:

Name	Data type	Address	Value
CIPOpenWithDataSize_Instance_0	CIPOpenWithDataSize		
Execute	BOOL		TRUE
EtherNetIPCfg	EtherNetIP_Cfg		
LocalEtherNetID	BYTE		2
RemoteAddress	ARRAY [0..3] OF BYTE		
RemoteAddress[0]	BYTE		192
RemoteAddress[1]	BYTE		168
RemoteAddress[2]	BYTE		2
RemoteAddress[3]	BYTE		66
DataSize	WORD		1994
TimeOutMs	UINT		20
Done	BOOL		FALSE
InstanceID	DINT		964
Busy	BOOL		FALSE
Error	BOOL		FALSE
ErrorID	WORD		0
ErrorIDEx	DWORD		0

```
ErrorIDEx => );
CIPOpenWithDataSize_Instance_0(
Execute TRUE := var2 TRUE ,
EtherNetIPCfg := ,
DataSize := ,
TimeOutMs := ,
Done => ,
InstanceID => ,
Busy => ,
Error => ,
ErrorID => ,
ErrorIDEx => );
```

LD:

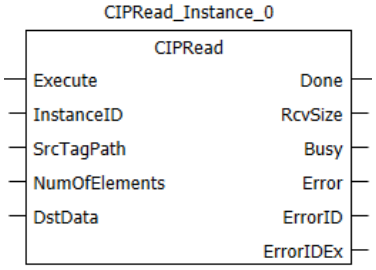


- **Precautions**
  - IP configuration follows standard IP specifications, and errors are reported after occurrence.
  - The maximum value of DataSize is 65481. If the DataSize value is 0, it is limited to the maximum of 65481.
  - InstanceID represents the instance number of a successfully opened connection, supporting up to 32 connections.
  - Each time the CIPOpenWithDataSize instruction is successfully executed, an available InstanceID is generated, and the newly generated InstanceID increments sequentially. If the instruction execution fails, the available InstanceID generated by the last successfully executed instruction is returned.

18.4.3.5 CIPRead

This instruction writes the data of remote controller public variables into local variables.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CIPRead	Variable reading	FB		CIPRead_Instance_0(Execute := , InstanceID := , SrcTagPath := , NumOfElements := , DstData := , Done => , RcvSize => , Busy => , Error => , ErrorID => , ErrorIDEx =>);

- **Related variables**

  Variable

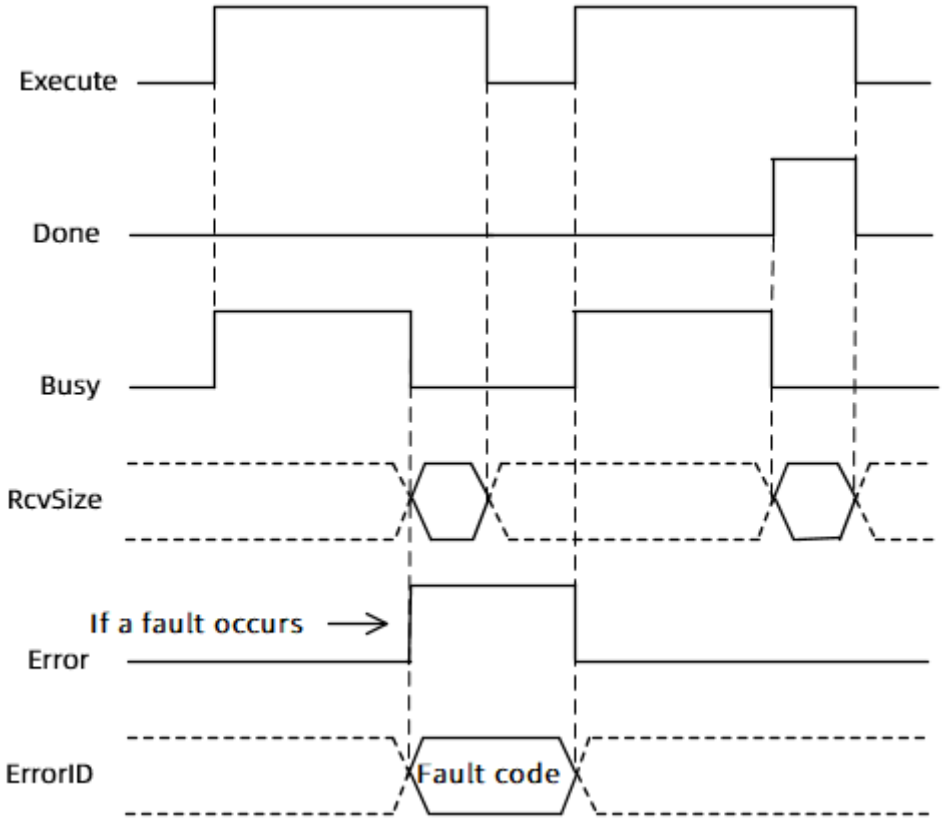
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Rising edge-triggered	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	InstanceID	Instance ID for the specified operation	DINT	InstanceID associated with CIPOpen output	0	Instance ID for the specified operation.
	SrcTagPath	External variable name	STRING	String composed of external variable names	''	External variable name or variable path.
	NumOfElements	Number of elements	UINT	Single element value is 1, and array is filled in based on the actual number of elements	0	Number of elements.
	DstData	Local variable for reading data	ANY	Local variable of the data	-	Local variable for reading data.
Output	Done	Done	BOOL	TRUE, FALSE	FALSE	Instruction is completed.
	RcvSize	Return byte size	DINT	Follows data type	0	Actual return byte size.
	Busy	Executing	BOOL	TRUE, FALSE	FALSE	Instruction is executing.
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	Error status.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	ErrorID	General fault code	WORD	Follow data type	0	General fault code. See 18.4.3.19 EIP Standard Fault Codes.
	ErrorIDEx	Extended fault code	DWORD	Depends on data type	0	Extended fault code. For details, see 18.4.3.19 EIP Standard Fault Codesand 18.4.3.20 Fault Codes.

- **Function description**

  This instruction writes the data of remote controller public variables into local variables.

- **Sequence diagram**



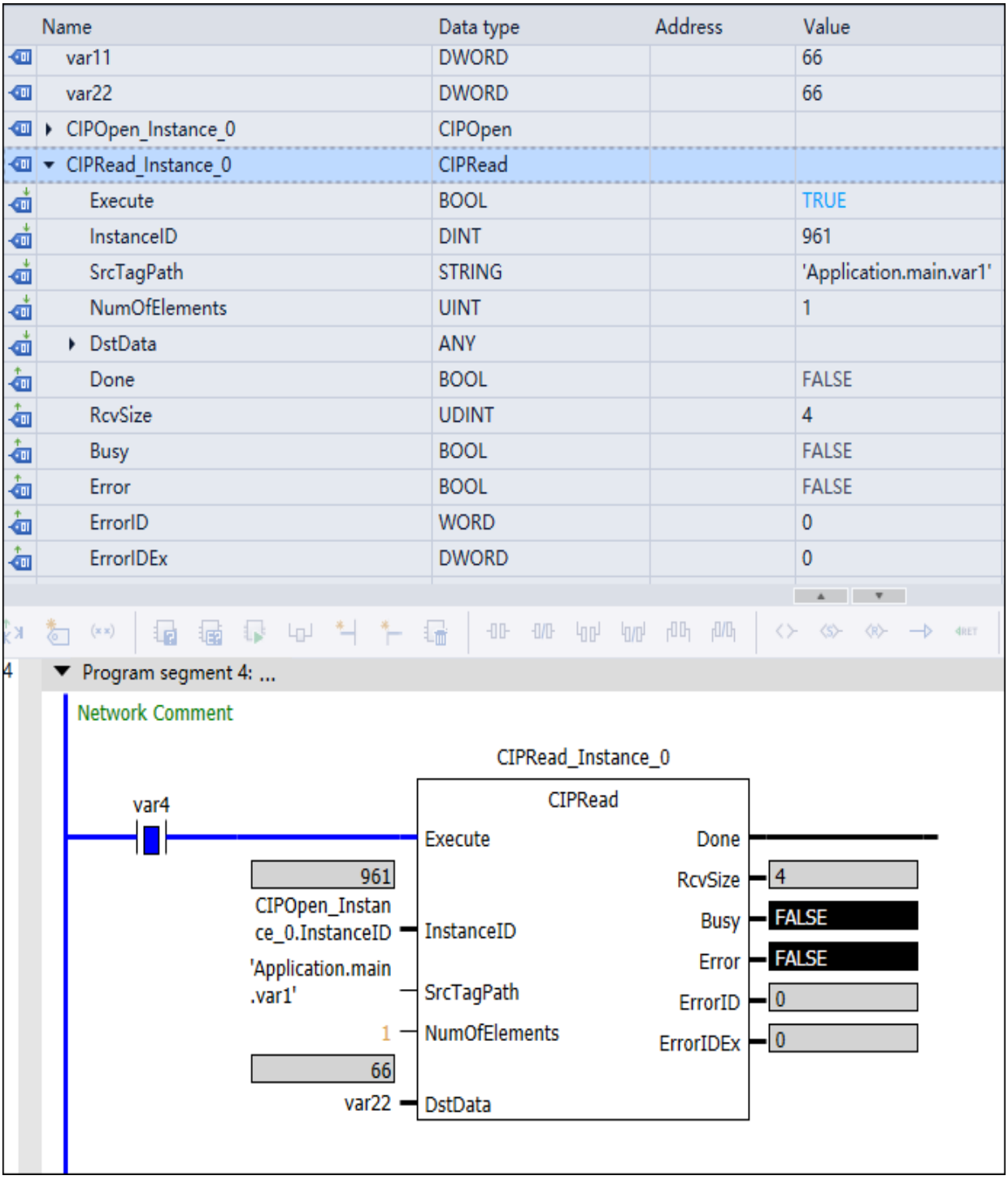
- **Program example**

  ST:

Name	Data type	Address	Value
CIPOpen_Instance_0	CIPOpen		
CIPOpenWithDataSize_Instance_0	CIPOpenWithDataSize		
CIPRead_Instance_0	CIPRead		
Execute	BOOL		TRUE
InstanceID	DINT		963
SrcTagPath	STRING		'Application.main.var1'
NumOfElements	UINT		1
DstData	ANY		
Done	BOOL		FALSE
RcvSize	UDINT		4
Busy	BOOL		FALSE
Error	BOOL		FALSE
ErrorID	WORD		0
ErrorIDEx	DWORD		0

```
ErrorIDEx => );
CIPRead_Instance_0(
Execute TRUE := var3 TRUE ,
InstanceID 963 := CIPOpen_Instance_0.InstanceID 963 ,
SrcTagPath'Application.main.va... := SrcTagPath2'Application.main.va...,
NumOfElements 1 := 1,
DstData := var22 22 ,
Done => ,
RcvSize => ,
Busy => ,
Error => ,
ErrorID => ,
ErrorIDEx => );
```

LD:

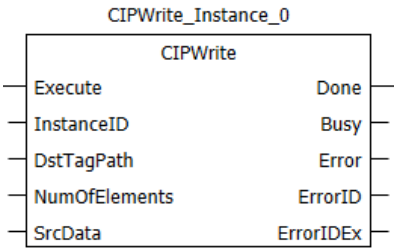


- **Precautions**
  - The CIPRead instruction does not support overall access to multi-dimensional arrays and structures.
  - If the selected InstanceID exceeds the length limit or has not been created, the Error bit is set.
  - An error is reported when the settings for InstanceID, SrcTagPath, NumberOfElements, and DstData output parameters do not match the specifications.
  - There are no restrictions on the length and validation rules for the external variable name SrcTagPath.
  - The NumberOfElements must not exceed the range of the DINT type. A single element value is 1, and an array is filled based on the actual number of elements.
  - The external variable accessed by SrcTagPath must be a network public variable.
  - When executing multiple CIPRead instructions, assign a different InstanceID (generated via CIPOpen or CIPOpenWithDataSize instruction) to each instruction instance. If multiple instructions use the same InstanceID, communication may be interrupted.

18.4.3.6 CIPWrite

This instruction writes the data of local public variables into the specified public variables of the remote controller.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CIPWrite	Variable writing	FB		CIPWrite_Instance_0(Execute := , InstanceID := , DstTagPath := , NumOfElements := , SrcData := , Done => , Busy => , Error => , ErrorID => , ErrorIDEx =>);

- **Related variables**

Variable

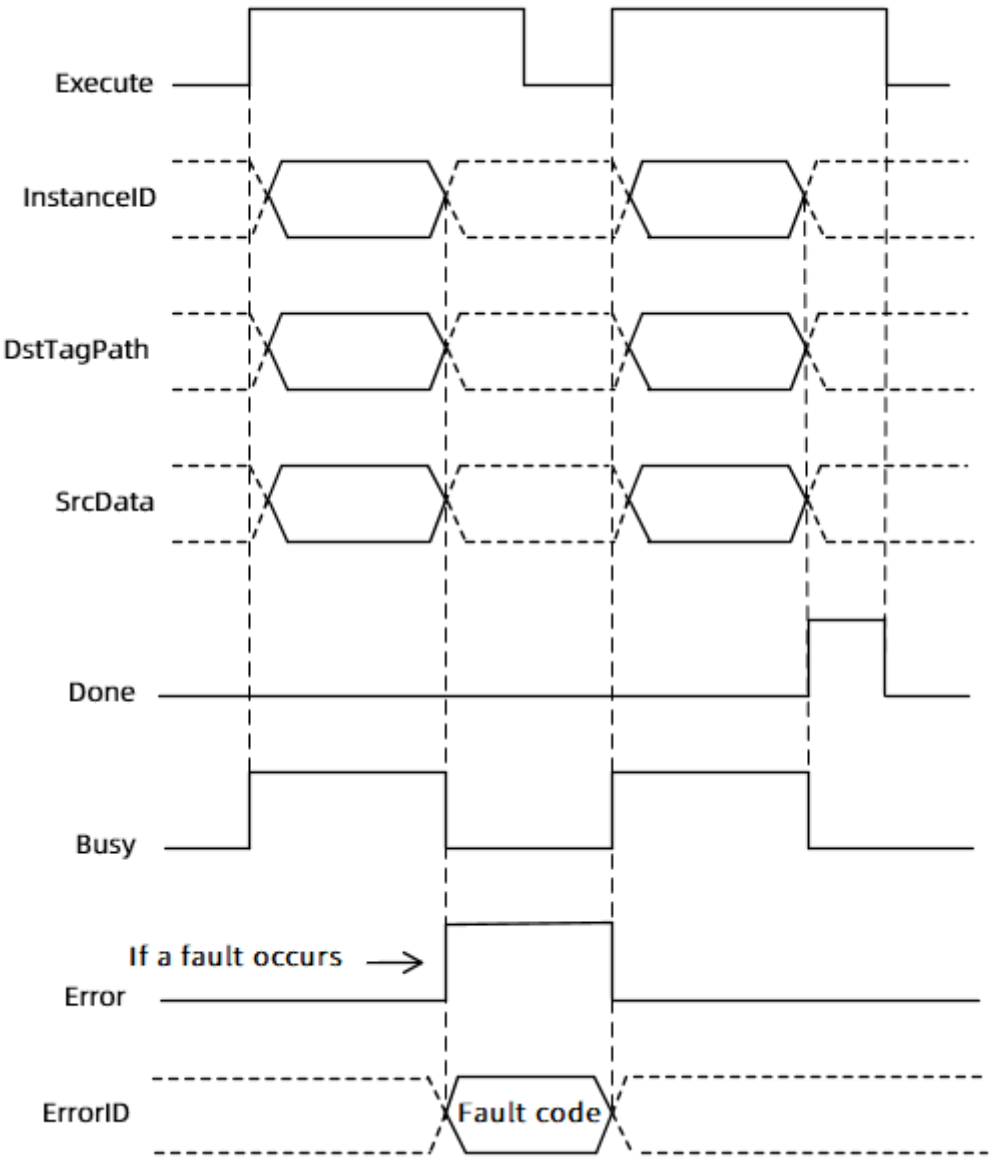
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Rising edge-triggered	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	InstanceID	Instance ID for the specified operation	DINT	InstanceID associated with CIPOpen output	0	Instance ID for the specified operation.
	DstTagPath	External variable name	STRING	String composed of external variable names	''	External variable name or variable path.
	NumOfElements	Number of elements	UINT	Single element value is 1, and array is filled in based on the actual number of elements	0	Number of elements.
	SrcData	Local variable for writing data	ANY	Local variable of the data	-	Local variable for writing data.
Output	Done	Done	BOOL	TRUE, FALSE	FALSE	Instruction is completed.
	Busy	Executing	BOOL	TRUE, FALSE	FALSE	Instruction is executing.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	Error status.
	ErrorID	General fault code	WORD	Follow data type	0	General fault code. See 18.4.3.19 EIP Standard Fault Codes.
	ErrorIDEx	Extended fault code	DWORD	Depends on data type	0	Extended fault code. For details, see 18.4.3.19 EIP Standard Fault Codesand 18.4.3.20 Fault Codes.

- **Function description**

  This instruction writes the data of local public variables into the specified public variables of the remote controller.

- **Sequence diagram**



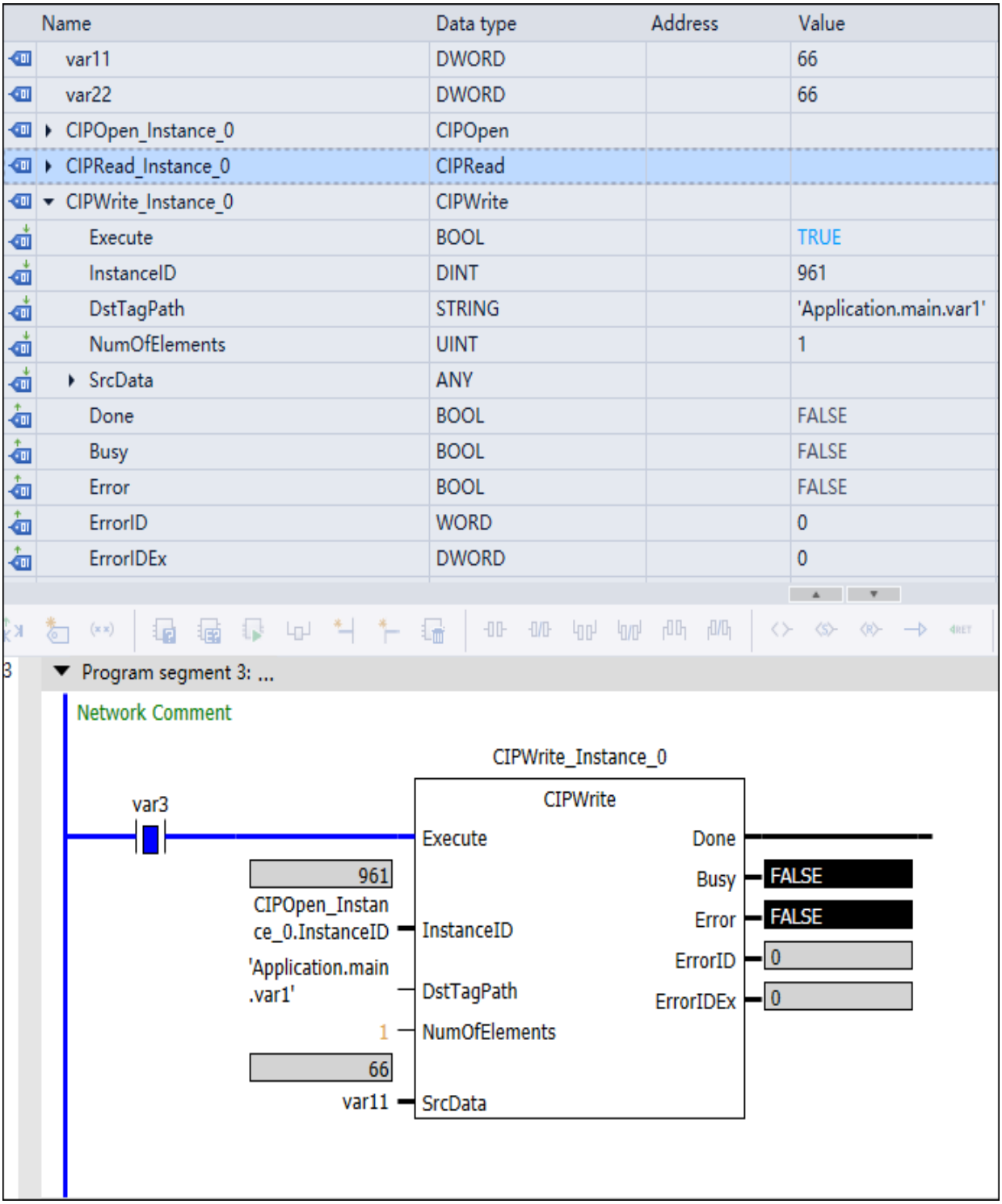
- **Program example**

ST:

Name	Data type	Address	Value
CIPOpen_Instance_0	CIPOpen		
CIPOpenWithDataSize_Instance_0	CIPOpenWithDataSize		
CIPRead_Instance_0	CIPRead		
CIPWrite_Instance_0	CIPWrite		
Execute	BOOL		TRUE
InstanceID	DINT		963
DstTagPath	STRING		'Application.main.var1'
NumOfElements	UINT		1
SrcData	ANY		
Done	BOOL		FALSE
Busy	BOOL		FALSE
Error	BOOL		FALSE
ErrorID	WORD		0
ErrorIDEx	DWORD		0
CIPSend_Instance_0	CIPSend		
CIPClose_Instance_0	CIPClose		
var1	BOOL		TRUE

```
CIPWrite_Instance_0(
Execute TRUE := var4 TRUE ,
InstanceID 963 := CIPOpen_Instance_0.InstanceID 963 ,
DstTagPath'Application.main.va... := DstTagPath1'Application.main.va...,
NumOfElements 1 := 1,
SrcData := var11 22 ,
Done => ,
Busy => ,
Error => ,
ErrorID => ,
ErrorIDEx => );
CIPRead_Instance_0(
Execute TRUE := var3 TRUE ,
InstanceID 963 := CIPOpen_Instance_0.InstanceID 963 ,
SrcTagPath'Application.main.va... := SrcTagPath2'Application.main.va...,
NumOfElements 1 := 1,
DstData := var22 22 ,
Done => ,
RcvSize => ,
Busy => ,
Error => ,
ErrorID => ,
```

LD:



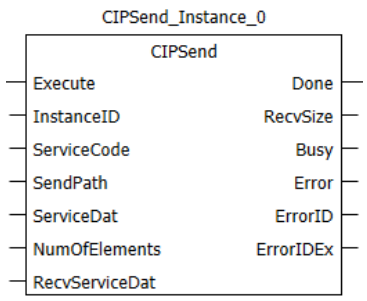
- **Precautions**
  - The CIPWrite instruction does not support overall access to multi-dimensional arrays and structures.
  - If the selected InstanceID exceeds the length limit or has not been created, the Error bit is set.
  - An error is reported when the InstanceID, DstTagPath, NumberOfElements, and SrcData output parameters do not match the specifications.
  - There are no restrictions on the length and validation rules for the external variable name DstTagPath.
  - The NumberOfElements must not exceed the range of the DINT type. A single element value is 1, and an array is filled based on the actual number of elements.
  - The external variable accessed by DstTagPath must be a network public variable.

◦ When executing multiple CIPWrite instructions, assign a different InstanceID (generated via CIPOpen or CIPOpenWithDataSize instruction) to each instruction instance. If multiple instructions use the same InstanceID, communication may be interrupted.

18.4.3.7 CIPSend

This instruction sends Class 3 data to the specified controller.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CIPSend	Send Class3 display message	FB		CIPSend_Instance_0(Execute := , InstanceID := , ServiceCode := , SendPath := , ServiceDat := , NumOfElements := , RecvServiceDat := , Done => , RecvSize => , Busy => , Error => , ErrorID => , ErrorIDEx =>);

- **Related variables**

Variable

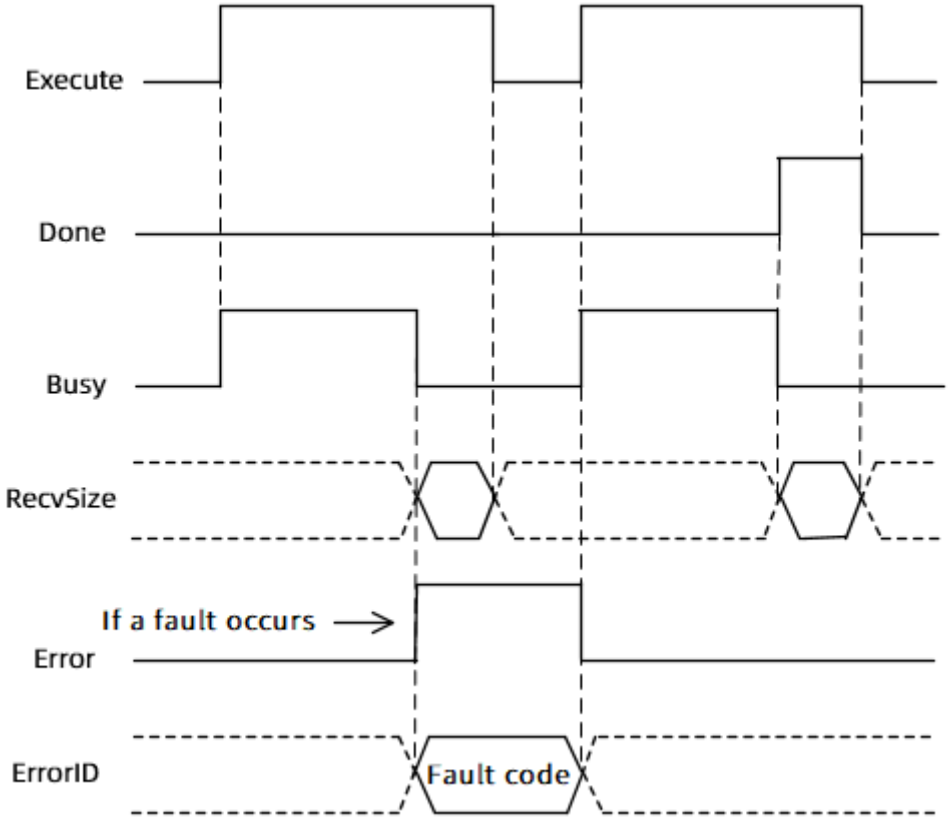
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Rising edge-triggered	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	InstanceID	Instance ID for the specified operation	DINT	InstanceID associated with CIPOpen output	0	Instance ID for the specified operation.
	ServiceCode	Service code	BYTE	EIP standard service code	0	Service code. See 18.4.3.18 EIP Standard Service Codes.
	SendPath	Send path	ExplicitMsg SendPath	Fill in according to the service code	0	The send path is composed of ClassID/InstanceID/isAttributeID /AttributeID.

<table>
<tr><th>Input/ Output</th><th>Variable</th><th>Name</th><th>Data Type</th><th>Valid Range</th><th>Default Value</th><th>Description</th></tr>
<tr><td rowspan="5"></td><td></td><td colspan="6"><table>
<tr><th>Variable</th><th>Name</th><th>Data type</th><th>Initial Value</th><th>Description</th></tr>
<tr><td>ClassID</td><td>Class ID</td><td>UINT</td><td>0</td><td>Class ID.</td></tr>
<tr><td>InstanceID</td><td>Instance ID</td><td>UINT</td><td>0</td><td>Instance ID.</td></tr>
<tr><td>isAttributeID</td><td>Whether the AttributeID is included</td><td>BOOL</td><td>FALSE</td><td>Whether the AttributeID is included.</td></tr>
<tr><td>AttributeID</td><td>Attribute ID</td><td>UINT</td><td>0</td><td>Attribute ID.</td></tr>
</table></td></tr>
<tr><td>ServiceDat</td><td>Service data</td><td>ANY</td><td>Local variable of the data</td><td>-</td><td>Service data.</td></tr>
<tr><td>NumOfElements</td><td>Number of elements</td><td>UINT</td><td>Follow data type</td><td>-</td><td>Number of elements.</td></tr>
<tr><td>RecvServiceDat</td><td>Return data</td><td>ANY</td><td>-</td><td>-</td><td>Return data.</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="6">Output</td><td>Done</td><td>Done</td><td>BOOL</td><td>TRUE, FALSE</td><td>FALSE</td><td>Instruction is completed.</td></tr>
<tr><td>RecvSize</td><td>Response data size</td><td>UDINT</td><td>Follows data type.</td><td>-</td><td>Response data size.</td></tr>
<tr><td>Busy</td><td>Executing</td><td>BOOL</td><td>TRUE, FALSE</td><td>FALSE</td><td>Instruction is executing.</td></tr>
<tr><td>Error</td><td>Faulty status</td><td>BOOL</td><td>TRUE, FALSE</td><td>FALSE</td><td>Error status.</td></tr>
<tr><td>ErrorID</td><td>General fault code</td><td>WORD</td><td>Follow data type</td><td>0</td><td>General fault code. See 18.4.3.19 EIP Standard Fault Codes.</td></tr>
<tr><td>ErrorIDEx</td><td>Extended fault code</td><td>DWORD</td><td>Depends on data type</td><td>0</td><td>Extended fault code. For details, see 18.4.3.19 EIP Standard Fault Codesand 18.4.3.20 Fault Codes.</td></tr>
</table>

- **Function description**

  This instruction sends Class 3 data to the specified controller.

- **Sequence diagram**



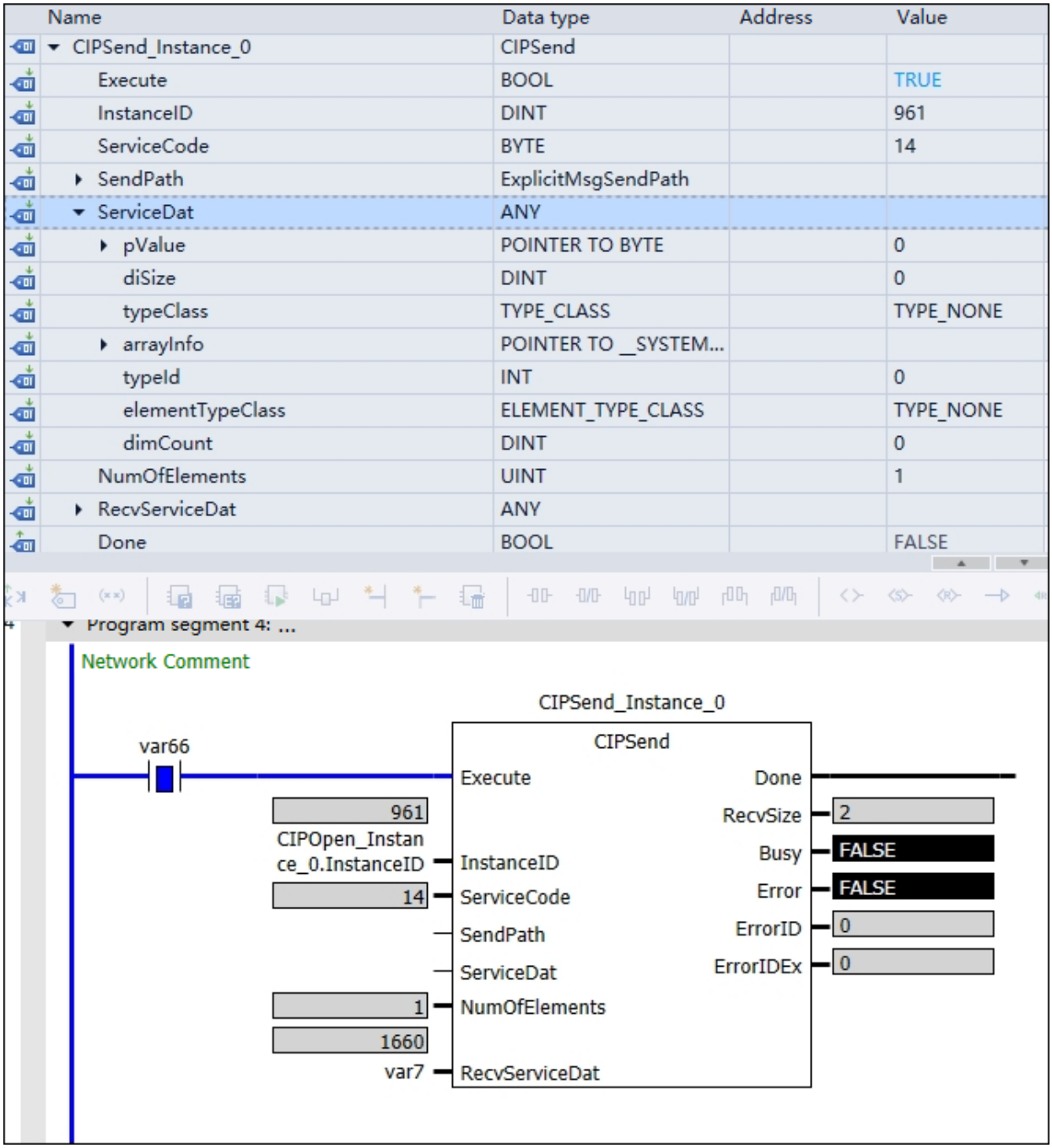
- **Program example**

  ST:

Name	Data type	Address	Value
ErrorIDEx	DWORD		0
CIPSend_Instance_0	CIPSend		
Execute	BOOL		TRUE
InstanceID	DINT		963
ServiceCode	BYTE		14
SendPath	ExplicitMsgSendPath		
ServiceDat	ANY		
NumOfElements	UINT		1
RecvServiceDat	ANY		
Done	BOOL		FALSE
RecvSize	UDINT		2
Busy	BOOL		FALSE
Error	BOOL		FALSE
ErrorID	WORD		0
ErrorIDEx	DWORD		0

```
CIPSend_Instance_0(
Execute  TRUE  := var5  TRUE  ,
InstanceID  963  := CIPOpen_Instance_0.InstanceID  963  ,
ServiceCode  14  := 14,
SendPath := ,
ServiceDat := ,
NumOfElements := ,
RecvServiceDat := var33  1660  ,
Done => ,
RecvSize => ,
Busy => ,
Error => ,
ErrorID => ,
ErrorIDEx => );
```

LD:

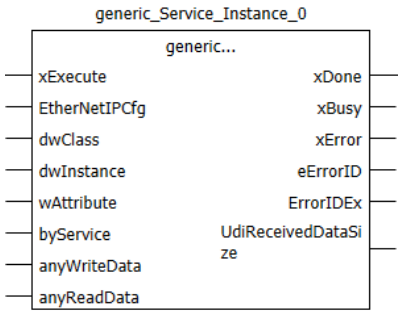


- **Precautions**
  - The CIPSend instruction does not support overall access to multi-dimensional arrays and structures.
  - Service code. For details, see 18.4.3.18 EIP Standard Service Codes.
  - ServiceData is a local variable that is only enabled when the service code includes write attributes; it does not need to be associated in other cases.
  - RecvServiceData is a local variable that receives the response of the service code, and RecvSize is the size of the response after the data is successfully sent.
  - When executing multiple CIPSend instructions, assign a different InstanceID (generated via CIPOpen or CIPOpenWithDataSize instruction) to each instruction instance. If multiple instructions use the same InstanceID, communication may be interrupted.

18.4.3.8 generic_Service

This is the instruction for EtherNet/IP generic service.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
generic_Service	EIP generic service	FB		generic_Service_Instance_0 (xExecute := , EtherNetIPCfg := , dwClass := , dwInstance := , wAttribute := , byService := , anyWriteData := , anyReadData := , xDone => , xBusy => , xError => , eErrorID => , ErrorIDEx => , UdiReceivedDataSize =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Enable	BOOL	TRUE, FALSE	-	Edge-triggered switch to execute the function block
	EtherNetIPCfg	Address of the target device	EtherNetIP_Cfg	Follows data type	-	IP address of the target device.
	dwClass	Class ID	DWORD	Follows data type	-	Requested class ID.
	dwInstance	Instance ID	DWORD	Follows data type	-	Requested instance ID.
	wAttribute	Attribute ID	WORD	Follows data type	-	Requested attribute ID. Set to 0xFFFF if not used.
	byService	Service code	BYTE	Follows data type	-	Requested service code. For details, see 18.4.3.18 EIP Standard Service Codes.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	anyWriteData	Requested data	ANY	Follows data type	-	Requested data will be filled into the data field of the request frame and sent.
	anyReadData	Response data	ANY	Follows data type	-	Content of the response message.
Output	xDone	Function block completed state	BOOL	TRUE, FALSE	-	Function block completed state
	xBusy	Execution status of function block	BOOL	TRUE, FALSE	-	Execution status of function block.
	xError	Error status of function block	BOOL	TRUE, FALSE	-	Error status of function block.
	eErrorID	Function block fault ID	ERROR	Follows data range	-	Function block fault ID. For details, see 18.4.3.19 EIP Standard Fault Codes.
	ErrorIDEx	Function block extended error ID	DWORD	Follows data range	-	Function block extended fault ID. For details, see 18.4.3.19 EIP Standard Fault Codes and 18.4.3.20 Fault Codes.
	udiReceivedDataSize	Actual length of received data	UDINT	Follows data range	-	Actual length of received data.

- **Function**
  - This instruction is for EtherNet/IP generic service, which is used to send an unconnected explicit message request to a device at a specific IP address.
  - The data type of eErrorID is the structure ERROR. The meaning is as follows.

Enumeration Value	Number	Description
NO_ERROR	0	The specified object has successfully executed the service.
CONNECTION_FAILURE	16#1	A connection-related service along the connection path has failed.
RESOURCE_UNAVAILABLE	16#2	The object resource required to execute the requested service is unavailable.

Enumeration Value	Number	Description
INVALID_PARAM_VALUE	16#3	See status code 0x20, which is the preferred value for this condition.
PATH_SEGMENT_ERROR	16#4	The processing node cannot recognize the path segment identifier or segment syntax. If a path segment error occurs, path processing must stop.
PATH_DESTINATION_UNKNOWN	16#5	The path references an object type, instance, or structural element that is unknown or does not exist in the processing node. If a destination path unknown error occurs, perform path processing.
PARTIAL_TRANSFER	16#6	Only partial expected data has been transferred.
CONNECTION_LOST	16#7	The message connection has been lost.
SERVICE_NOT_SUPPORTED	16#8	The requested service is not implemented or not defined for this object type/instance.
INVALID_ATTRIBUTE_VALUE	16#9	Invalid attribute data has been detected.
ATTRIBUTE_LIST_ERROR	16#A	In a Get_Attribute_List or Set_Attribute_List response, the status of an attribute is non-zero.
ALREADY_IN_REQUEST_STATE	16#B	The object is already in the service request mode/state.
OBJECT_STATE_ERROR	16#C	The object cannot perform the requested service in current mode/state.
OBJECT_ALREADY_EXISTS	16#D	The object instance requested to be created already exists.
ATTRIBUTE_NOT_SETTABLE	16#E	A modification request is received for an unmodifiable attribute.
PRIVILEGE_VIOLATION	16#F	Privilege/permission check failed.
DEVICE_STATE_ERROR	16#10	The current mode/state of the device prohibits execution of the requested service.
REPLY_DATA_TOO_LARGE	16#11	The data to be transmitted in the response buffer is larger than the allocated response buffer.
FRAGMENTATION_OF_VALUE	16#12	The service specifies an operation that splits a primary data value, i.e. half of a REAL data type.
NOT_ENOUGH_DATA	16#13	The service does not provide enough data to perform the specified operation.
ATTRIBUTE_NOT_SUPPORTED	16#14	The attribute specified in the request is not supported.
TOO_MUCH_DATA	16#15	The service provides more data than expected.
OBJECT_DOES_NOT_EXIST	16#16	The specified object does not exist in the device.
SERVICE_FRAGMENTATION_SEQUENCE_NOT_IN_PROGRESSS	16#17	The fragmentation sequence for this service is not active for this data.
NO_STORED_ATTRIBUTE_DATA	16#18	The attribute data for this object is not saved prior to the requested server.

Enumeration Value	Number	Description
STORE_OPERATION_FAILURE	16#19	The attribute data for this object is not saved due to a failed attempt.
ROUTING_FAILURE_REQUEST_PACKET_TOO_LARGE	16#1A	The service request packet is too large to be transmitted on the path from the target to the network. The routing device is forced to abort the service.
ROUTING_FAILURE_RESPONSE_PACKET_TOO_LARGE	16#1B	The service response packet is too large to be transmitted on the path from the target to the network. The routing device is forced to abort the service.
MISSING_ATTRIBUTE_LISTENTRY_DATA	16#1C	The service does not provide the attribute from the attribute list required to perform the requested operation.
INVALID_ATTRIBUTE_VALUE_LIST	16#1D	The service is returning a list of attributes, including status information for invalid attributes.
EMBEDDED_SERVICE_ERROR	16#1E	An embedded service causes an error.
VENDOR_SPECIFIC_ERROR	16#1F	A vendor-specific error occurs. The additional code field of the error response defines the specific error encountered. This applies only when the error codes displayed in this table or in the object type definition do not accurately reflect the condition.
INVALID_PARAMETER	16#20	The parameter associated with the request is invalid. This code is used when this specification and/or the specification of requireman application object are not met.
WRITE_ONEC_VALUE_OR_MEDIUM_ALREADY_WRITTEN	16#21	An attempt is made to write to a write-once medium that has already been written (e.g., WORM drive, PROM), or to modify a value that cannot be changed once established.
INVALID_REPLY_RECEIVED	16#22	An invalid reply is received (e.g., the reply service code does not match the expected service code, or the reply message is shorter than the expected minimum reply size). This status code can be used for other reasons of invalid replies.
BUFFER_OVERFLOW	16#23	The received message is larger than the receiving buffer can handle. The message is discarded.
MESSAGE_FORMAT_ERROR	16#24	The server does not support the format of the received message.
KEY_FAILURE_IN_PATH	16#25	The key segment as the first segment in the path does not match the target module. The object-specific status must indicate failure of each part of the key check.
PATH_SIZE_INVALID	16#26	The path size sent with the service request is either too small to route the request to the object, or contains too much routing data.

Enumeration Value	Number	Description
UNEXPECTED_ATTRIBUTE_IN_LIST	16#27	An attempt is made to set an attribute that cannot be set at this time.
INVALID_MEMBER_ID	16#28	The member ID specified in the request does not exist in the specified type/instance/attribute.
MEMBER_NOT_SETTABLE	16#29	A request is received to modify a non-modifiable member.
GROUP_2_ONLY_SERVER_GENERAL_FAILURE	16#2A	This error code can only be reported by DeviceNet Group 2 servers with a code space of 4K or lower, and can only be used in place of unsupported services, unsupported attributes, and non-settable attributes.
UNKNOWN_MODBUS_ERROR	16#2B	The CIP to Modbus converter receives an unknown Modbus error code.
ATTRIBUTE_NOT_GETTABLE	16#2C	A request is received to read a non-readable attribute.
INSTANCE_NOT_DELETABLE	16#2D	The requested object instance cannot be deleted.
SERVICE_NOT_SUPPORTED_FOR_SPECIFIED_PATH	16#2E	The object supports the service, but does not support the specified application path (e.g., attribute). Note: Do not use for any setting services (use general status code 0x0E or 0x29 instead).
TIME_OUT	16#100	Request timed out.
INTERFACE_MISSING	16#101	IEtherNetIPService is not implemented.
REMOTE_CALL_FAILED	16#102	No physical connection.
NULL_POINTER	16#103	Invalid input value: NULL.
INVALID_DATA_SIZE	16#104	Data size is invalid.
WRONG_INTERFACE_VERSION	16#105	Version does not match. The device implements a different version of the interface for the called method.
NO_MEMORY	16#106	Insufficient memory.
UNKNOWN_ERROR	16#107	An unknown error occurred.
ABORTED	16#108	The service is aborted.

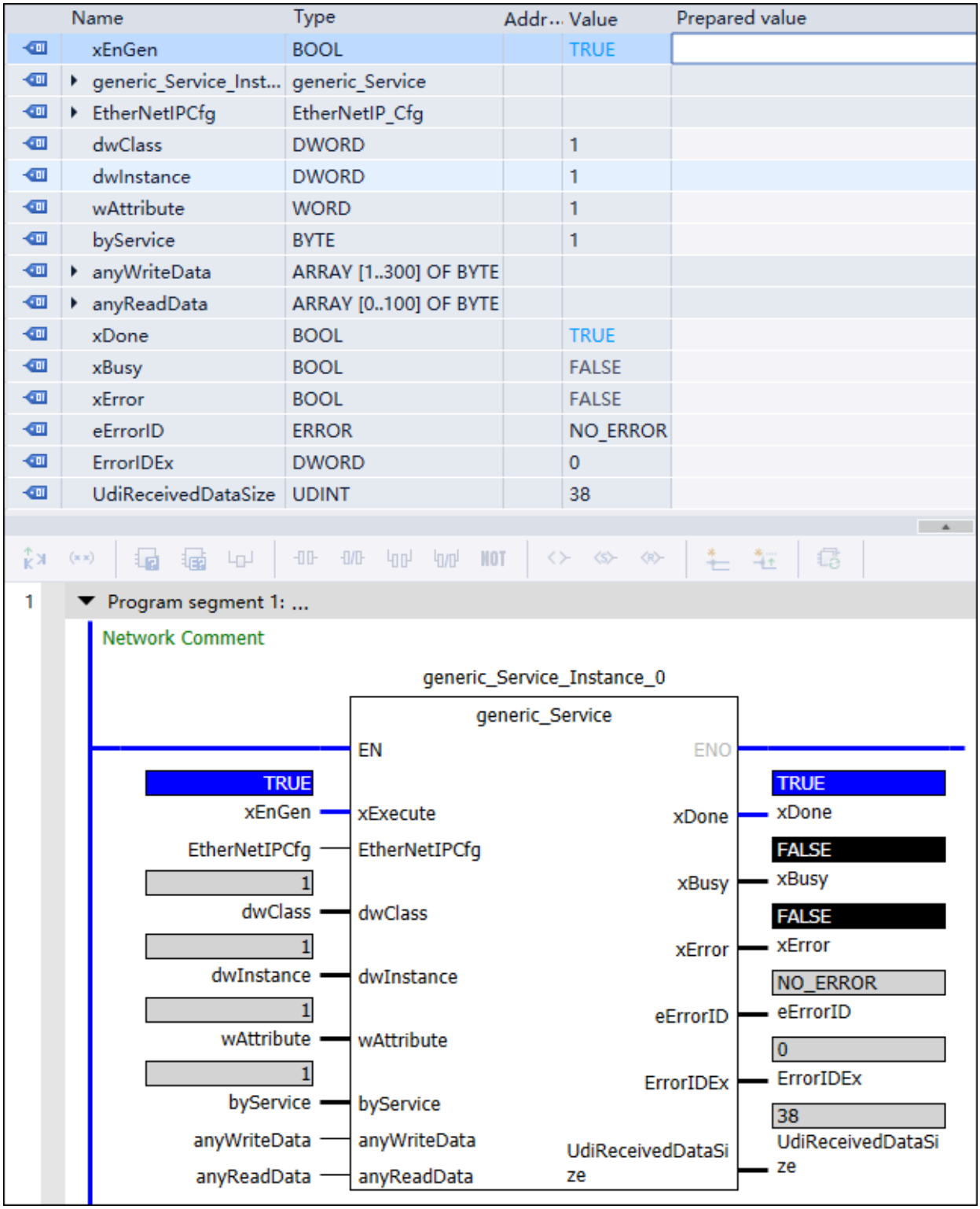
- **Program example**

  ST:

Name	Type	Addr...	Value
generic_Service_Instance_0	generic_Service		
getallen	BOOL		TRUE
EtherNetIPCfg	EtherNetIP_Cfg		
dwClass	DWORD		1
dwInstance	DWORD		1
wAttribute	WORD		1
byService	BYTE		1
anyWriteData	ARRAY [0..100] OF BYTE		
anyReadData	ARRAY [0..100] OF BYTE		
UdiReceivedDataSize	UDINT		38
xDone	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
eErrorID	ERROR		NO_ERROR
ErrorIDEx	DWORD		0

```
generic_Service_Instance_0
      (xExecute TRUE := getallen TRUE,
       EtherNetIPCfg := EtherNetIPCfg,
       dwClass 1 := dwClass 1,
       dwInstance 1 := dwInstance 1,
       wAttribute 1 := wAttribute 1,
       byService 1 := byService 1,
       anyWriteData := anyWriteData,
       anyReadData := anyReadData,
       xDone TRUE => xDone TRUE,
       xBusy FALSE => xBusy FALSE,
       xError FALSE => xError FALSE,
       eErrorID NO_ERROR => eErrorID NO_ERROR,
       ErrorIDEx 0 => ErrorIDEx 0,
       UdiReceivedDataSize 38 => UdiReceivedDataSize 38 );
   RETURN
```

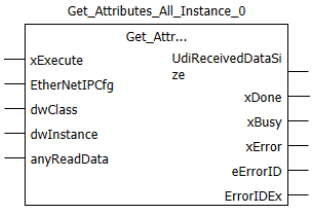
LD:



18.4.3.9 Get_Attributes_All

This instruction queries all attribute of a specific instance of a CIP object.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
Get_Attributes_All	Get all attributes of a specific instance	FB		Get_Attributes_All_Instance_0(xExecute := , EtherNetIPCfg := , dwClass := , dwInstance := , anyReadData := , UdiReceivedDataSize => , xDone => , xBusy => , xError => , eErrorID => , ErrorIDEx =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Enable	BOOL	TRUE, FALSE	-	Edge-triggered switch to execute the function block
	EtherNetIPCfg	Address of the target device	EtherNetIP_Cfg	Follows data type	-	IP address of the target device.
	dwClass	Class ID	DWORD	Follows data type	-	Requested class ID.
	dwInstance	Instance ID	DWORD	Follows data type	-	Requested instance ID.
	anyReadData	Response data	ANY	Follows data type	-	Content of the response message.
Output	UdiReceived DataSize	Actual length of received data	UDINT	Follows data type	-	Actual received data length.
	xDone	Function block completed state	BOOL	TRUE, FALSE	-	Function block completed state
	xBusy	Execution status of function block	BOOL	TRUE, FALSE	-	Execution status of function block.
	xError	Error status of function block	BOOL	TRUE, FALSE	-	Error status of function block.
	eErrorID	Function block fault ID	ERROR	Follows data type	-	Function block fault ID. For details, see 18.4.3.19 EIP Standard Fault Codes.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	ErrorIDEx	Function block extended error ID	DWORD	Follows data type	-	Function block extended fault ID. For details, see 18.4.3.19 EIP Standard Fault Codes and 18.4.3.20 Fault Codes.

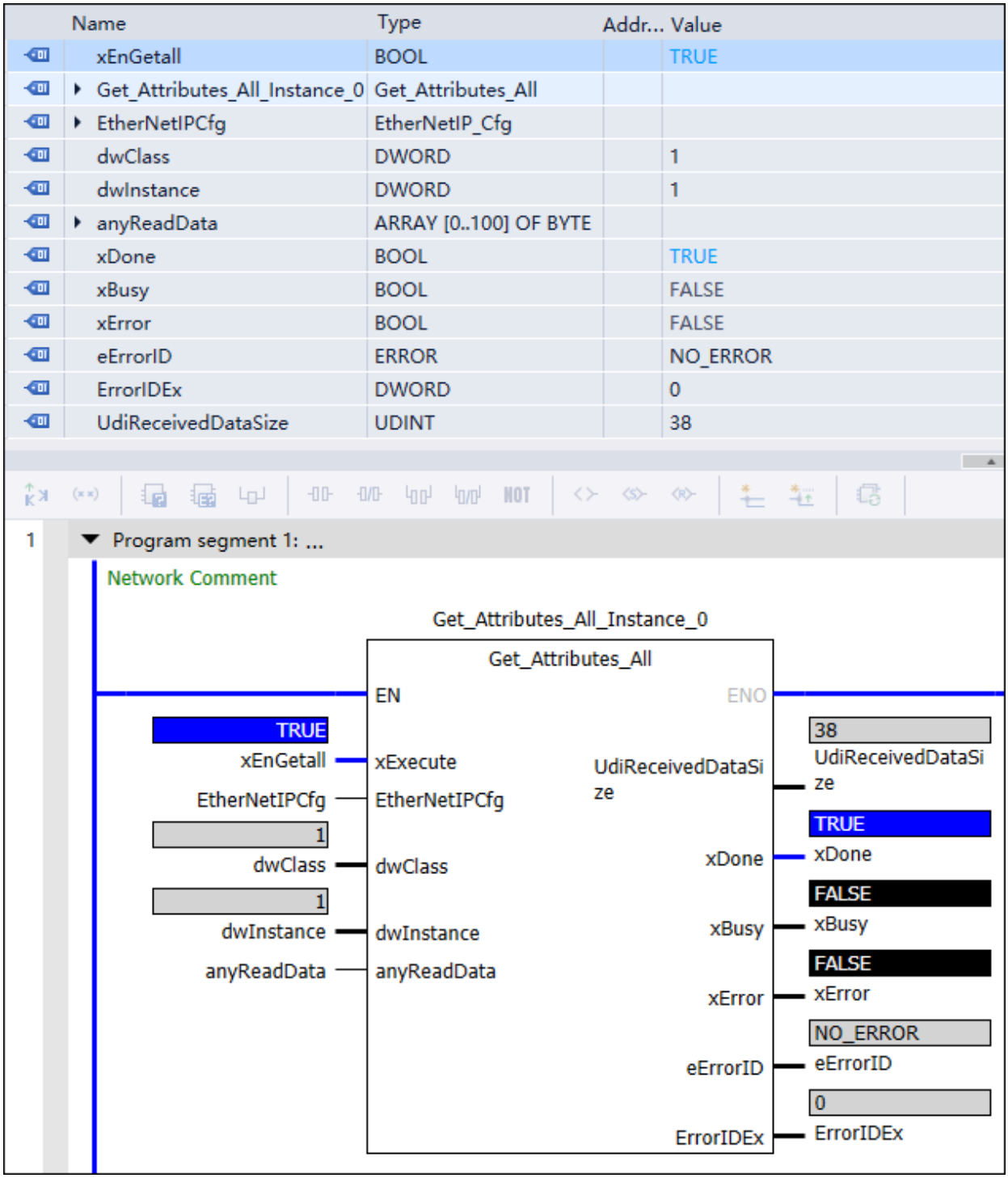
- **Function**
  This instruction is used to send a non-connected explicit message request to a device at a specific IP address to retrieve all attributes of a specified instance.
- **Program example**

  ST:

Name	Type	Ad...	Value
Get_Attributes_All_Instance_0	Get_Attributes_All		
getallen	BOOL		TRUE
EtherNetIPCfg	EtherNetIP_Cfg		
dwClass	DWORD		1
dwInstance	DWORD		1
anyReadData	ARRAY [0..100] OF BYTE		
UdiReceivedDataSize	UDINT		38
xDone	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
eErrorID	ERROR		NO_ERROR
ErrorIDEx	DWORD		0

```
CAS FOR WHI REP RET EX REG
Get_Attributes_All_Instance_0(xExecute TRUE := getallen TRUE ,
      EtherNetIPCfg := EtherNetIPCfg,
      dwClass 1 := dwClass 1 ,
      dwInstance 1 := dwInstance 1 ,
      anyReadData := anyReadData,
      UdiReceivedDataSize 38 => UdiReceivedDataSize 38 ,
      xDone TRUE => xDone TRUE ,
      xBusy FALSE => xBusy FALSE ,
      xError FALSE => xError FALSE ,
      eErrorID NO_ERROR => eErrorID NO_ERROR ,
      ErrorIDEx 0 => ErrorIDEx 0 );
```

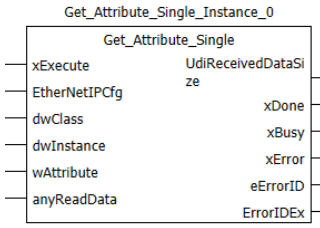
  LD:



18.4.3.10 Get_Attribute_Single

This instruction gets a specified attribute of a specific instance of a CIP object.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
Get_Attribute_Single	Get a specified attribute of a specific instance	FB		Get_Attribute_Single_Instance_0(xExecute := , EtherNetIPCfg := , dwClass := , dwInstance := , wAttribute := , anyReadData := , UdiReceivedDataSize => , xDone => , xBusy => , xError => , eErrorID => , ErrorIDEx =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Enable	BOOL	TRUE, FALSE	-	Edge-triggered switch to execute the function block
	EtherNetIPCfg	Address of the target device	EtherNetIP_Cfg	Follows data type	-	IP address of the target device.
	dwClass	Class ID	DWORD	Follows data type	-	Requested class ID.
	dwInstance	Instance ID	DWORD	Follows data type	-	Requested instance ID.
	wAttribute	Attribute ID	WORD	Follows data type	-	Requested attribute ID. Set to 0xFFFF if not used.
	anyReadData	Response data	ANY	Follows data type	-	Content of the response message.
Output	UdiReceivedDataSize	Actual length of received data	UDINT	Follows data type	-	Actual received data length.
	xDone	Function block completed state	BOOL	TRUE, FALSE	-	Function block completed state
	xBusy	Execution status of function block	BOOL	TRUE, FALSE	-	Execution status of function block.
	xError	Error status of function block	BOOL	TRUE, FALSE	-	Error status of function block.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	eErrorID	Function block fault ID	ERROR	Follows data type	-	Function block fault ID. For details, see 18.4.3.19 EIP Standard Fault Codes.
	ErrorIDEx	Function block extended error ID	DWORD	Follows data type	-	Function block extended fault ID. For details, see 18.4.3.19 EIP Standard Fault Codes and 18.4.3.20 Fault Codes.

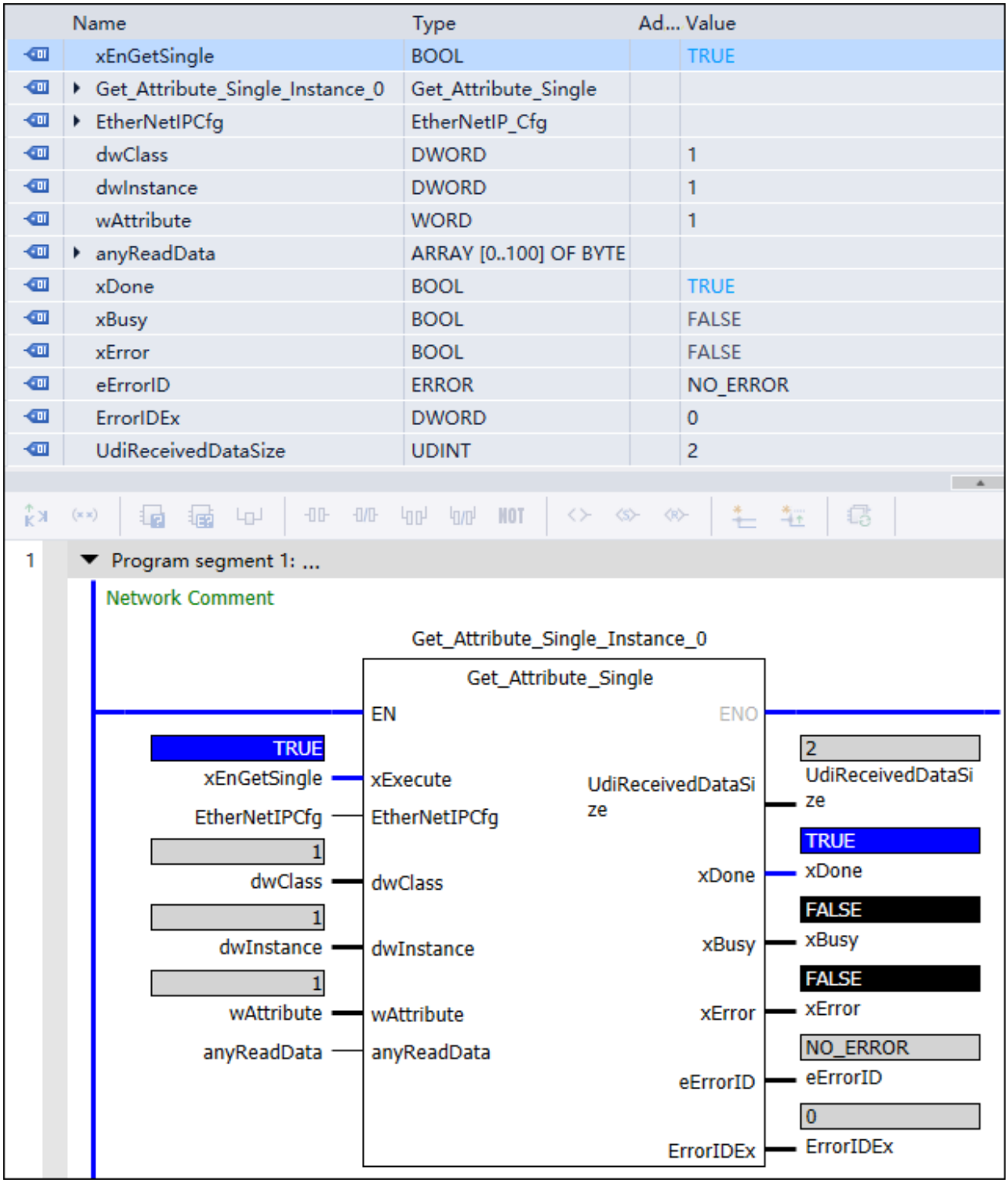
- **Function**
  This instruction is used to send a non-connected explicit message request to a device at a specific IP address to retrieve a specified attribute of a specified instance.
- **Program example**

  ST:

Name	Type	Ad…	Value
Get_Attribute_Single_Instance_0	Get_Attribute_Single		
getsingleen	BOOL		TRUE
EtherNetIPCfg	EtherNetIP_Cfg		
dwClass	DWORD		1
dwInstance	DWORD		1
wAttribute	WORD		1
UdiReceivedDataSize	UDINT		2
anyReadData	ARRAY [0..100] OF BYTE		
xDone	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE

```
Get_Attribute_Single_Instance_0(xExecute TRUE := getsingleen TRUE,
          EtherNetIPCfg := EtherNetIPCfg,
          dwClass 1 := dwClass 1,
          dwInstance 1 := dwInstance 1,
          wAttribute 1 := wAttribute 1,
          anyReadData := anyReadData,
          UdiReceivedDataSize 2 => UdiReceivedDataSize 2,
          xDone TRUE => xDone TRUE,
          xBusy FALSE => xBusy FALSE,
          xError FALSE => xError FALSE,
          eErrorID NO_ERROR => eErrorID NO_ERROR,
          ErrorIDEx 0 => ErrorIDEx 0 );RETURN
```

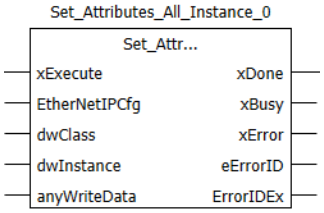
  LD:



18.4.3.11 Set_Attributes_All

Set all attributes of a CIP object-specific instance.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
Set_Attributes_All	Set all attributes of a specific instance	FB		Set_Attributes_All_Instance_0(x Execute := , EtherNetIPCfg := , dwClass := , dwInstance := , anyWriteData := , xDone => , xBusy => , xError => , eErrorID => , ErrorIDEx =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Enable	BOOL	TRUE, FALSE	-	Edge-triggered switch to execute the function block
	EtherNetIPCfg	Address of the target device	EtherNetIP_Cfg	Follows data type	-	IP address of the target device.
	dwClass	Class ID	DWORD	Follows data type	-	Requested class ID.
	dwInstance	Instance ID	DWORD	Follows data type	-	Requested instance ID.
	anyWriteData	Requested data	ANY	Follows data type	-	Requested data, which will be filled into the data field of the request frame and sent.
Output	xDone	Function block completed state	BOOL	TRUE, FALSE	-	Function block completed state
	xBusy	Execution status of function block	BOOL	TRUE, FALSE	-	Execution status of function block.
	xError	Error status of function block	BOOL	TRUE, FALSE	-	Error status of function block.
	eErrorID	Function block fault ID	ERROR	Follows data type	-	Function block fault ID. For details, see 18.4.3. 19 EIP Standard Fault Codes.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	ErrorIDEx	Function block extended error ID	DWORD	Follows data type	-	Function block extended fault ID. For details, see 18.4.3.19 EIP Standard Fault Codes and 18.4.3.20 Fault Codes.

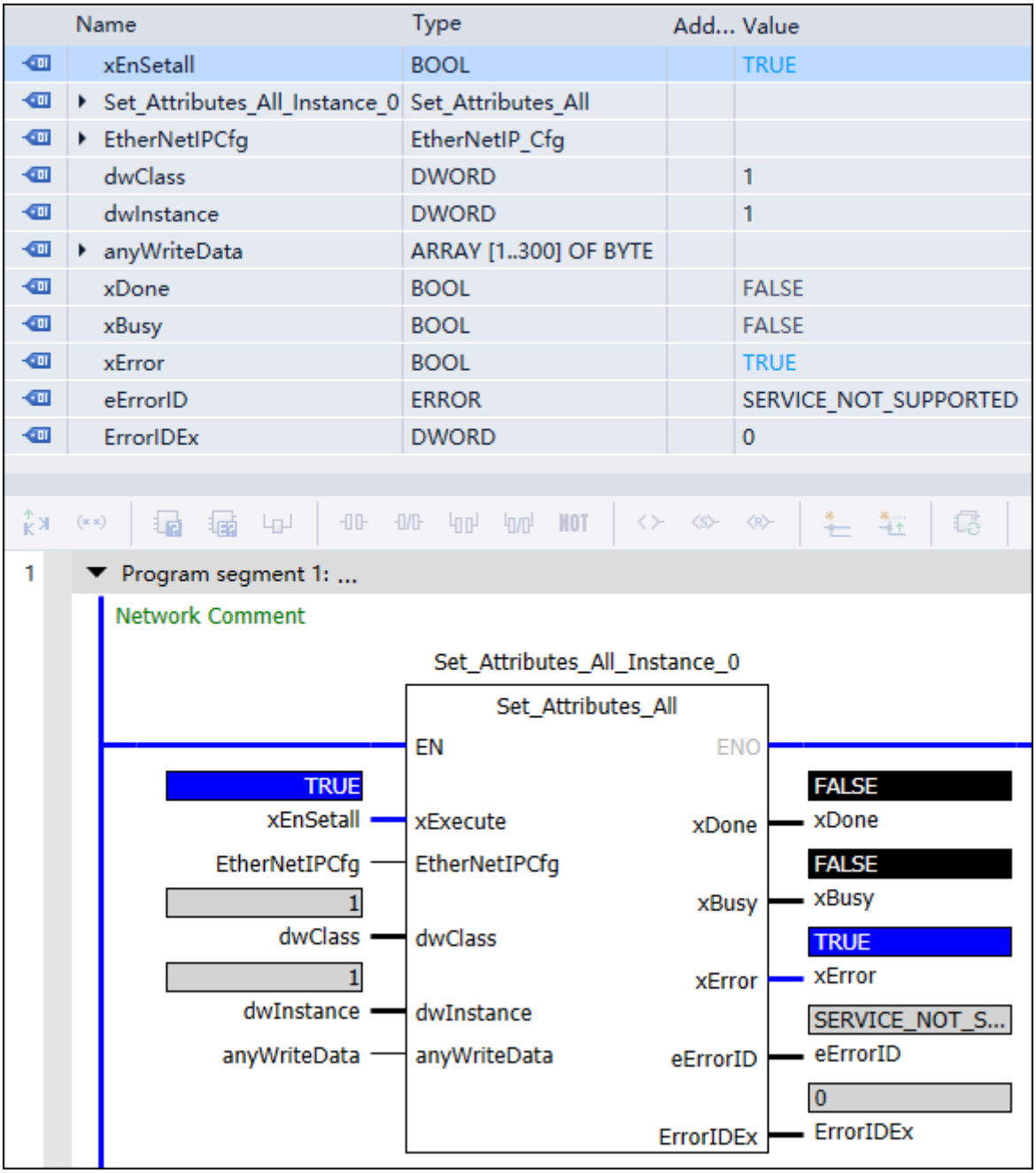
- **Function**
  This instruction is used to send a non-connected explicit message request to a specific IP address device to set all attributes of a specified instance.
- **Program example**

  ST:

Name	Type	Add...	Value
Set_Attributes_All_Instance_0	Set_Attributes_All		
setallen	BOOL		TRUE
EtherNetIPCfg	EtherNetIP_Cfg		
dwClass	DWORD		1
dwInstance	DWORD		1
anyWriteData	ARRAY [1..300] OF BYTE		
xDone	BOOL		FALSE
xBusy	BOOL		FALSE
xError	BOOL		TRUE
eErrorID	ERROR		SERVICE_NOT_SUPPORTED
ErrorIDEx	DWORD		0

```
Set_Attributes_All_Instance_0(xExecute TRUE := setallen TRUE,
        EtherNetIPCfg := EtherNetIPCfg,
        dwClass 1 := dwClass 1,
        dwInstance 1 := dwInstance 1,
        anyWriteData := anyWriteData,
        xDone FALSE => xDone FALSE,
        xBusy FALSE => xBusy FALSE,
        xError TRUE => xError TRUE,
        eErrorIDSERVICE_NOT_SUPPOR... => eErrorIDSERVICE_NOT_SUPPOR...,
        ErrorIDEx 0 => ErrorIDEx 0 );RETURN
```

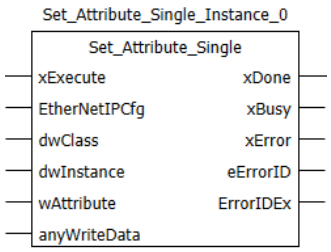
  LD:



18.4.3.12 Set_Attribute_Single

This instruction sets a specified attribute of a specific instance of a CIP object.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
Set_Attribute_Single	Set a specific attribute of a specific instance	FB		Set_Attribute_Single_Instance_0(xExecute := , EtherNetIPCfg := , dwClass := , dwInstance := , wAttribute := , anyWriteData := , xDone => , xBusy => , xError => , eErrorID => , ErrorIDEx =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Enable	BOOL	TRUE, FALSE		Edge-triggered switch to execute the function block
	EtherNetIPCfg	Address of the target device	EtherNetIP_Cfg	Follows data type	-	IP address of the target device.
	dwClass	Class ID	DWORD	Follows data type	-	Requested class ID.
	dwInstance	Instance ID	DWORD	Follows data type	-	Requested instance ID.
	wAttribute	Attribute ID	WORD	Follows data type	-	Requested attribute ID. Set to 0xFFFF if not used.
	anyWriteData	Requested data	ANY	Follows data type	-	Requested data, which will be filled into the data field of the request frame and sent.
Output	xDone	Function block completed state	BOOL	TRUE, FALSE	-	Function block completed state
	xBusy	Execution status of function block	BOOL	TRUE, FALSE	-	Execution status of function block.
	xError	Error status of function block	BOOL	TRUE, FALSE	-	Error status of function block.
	eErrorID	Function block fault ID	ERROR	Follows data type	-	Function block fault ID. For details, see 18.4.3.19 EIP Standard Fault Codes.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	ErrorIDEx	Function block extended error ID	DWORD	Follows data type	-	Function block extended fault ID. For details, see 18.4.3.19 EIP Standard Fault Codes and 18.4.3.20 Fault Codes.

- **Function**

  This instruction is used to send a non-connected explicit message request to a device at a specific IP address to set a specified attribute of a specified instance.

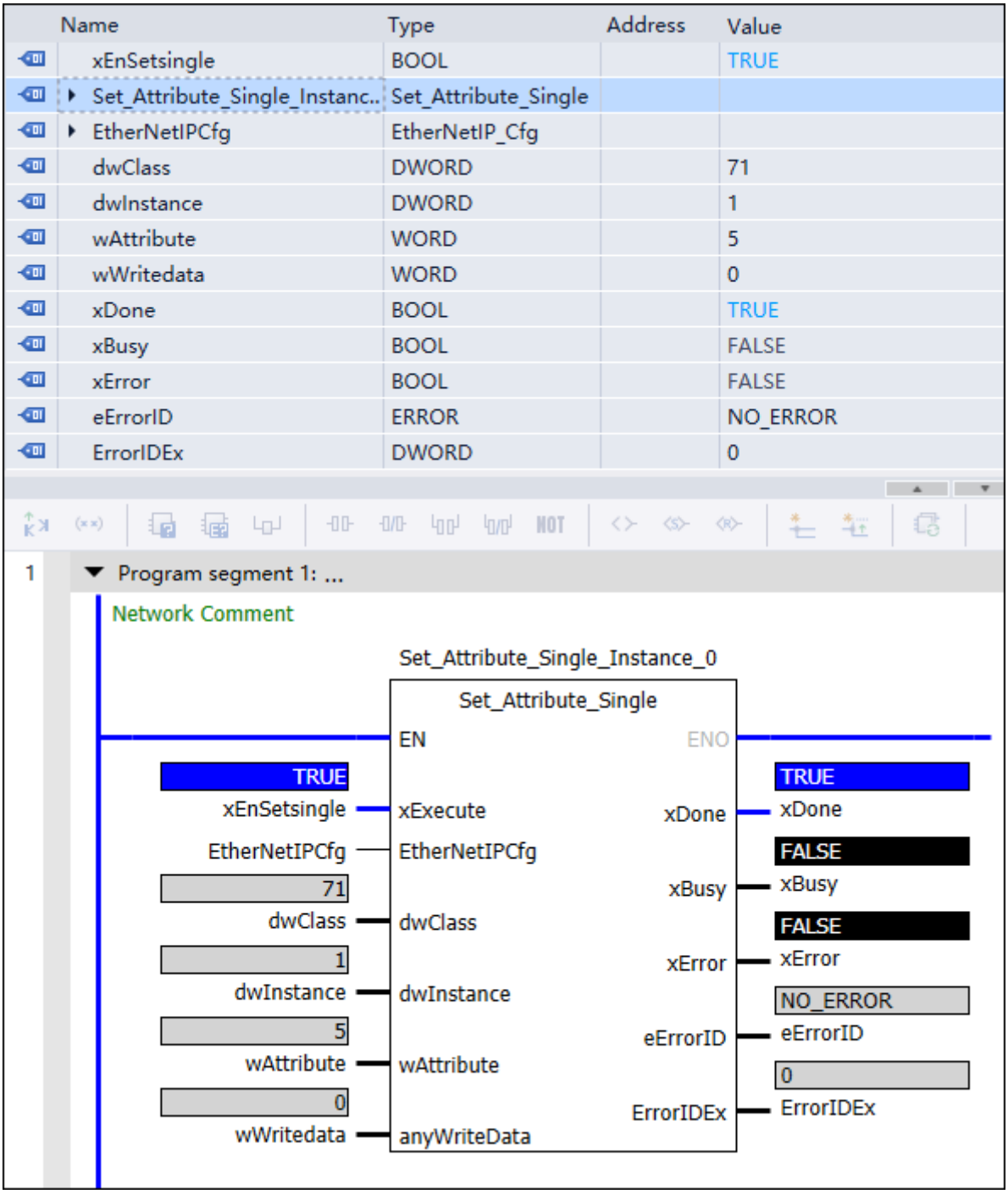
- **Program example**

  ST:

Name	Type	Addre...	Value
Set_Attribute_Single_Insta...	Set_Attribute_Single		
setsingleen	BOOL		TRUE
EtherNetIPCfg	EtherNetIP_Cfg		
dwClass	DWORD		71
dwInstance	DWORD		1
wAttribute	WORD		5
wWritedata	WORD		0
xDone	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
eErrorID	ERROR		NO_ERROR
ErrorIDEx	DWORD		0

```
Set_Attribute_Single_Instance_0(xExecute TRUE := setsingleen TRUE,
          EtherNetIPCfg := EtherNetIPCfg,
          dwClass 71 := dwClass 71,
          dwInstance 1 := dwInstance 1,
          wAttribute 5 := wAttribute 5,
          anyWriteData := wWritedata 0,
          xDone TRUE => xDone TRUE,
          xBusy FALSE => xBusy FALSE,
          xError FALSE => xError FALSE,
          eErrorID NO_ERROR => eErrorID NO_ERROR,
          ErrorIDEx 0 => ErrorIDEx 0 );
```

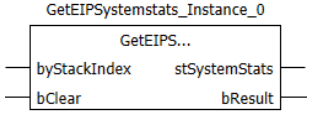
  LD:



18.4.3.13 GetEIPSystemstats

Retrieve EIP system-level diagnostic information.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
GetEIPSystemstats	Retrieve EIP system diagnostic information	FB		GetEIPSystemstats_Instance_0(byStackIndex := , bClear := , stSystemStats => , bResult =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	byStackIndex	Protocol stack number	BYTE	Follows data type	0	If EIP protocol is added to network ports, the value is 0 for the first port, 1 for the second port, and so on. If EIP is only configured on a single network port, the value is 0.
	bClear	Diagnostic data clear	BOOL	TRUE, FALSE	FALSE	Active high. Clears the current diagnostic data.
Output	stSystemStats	EIP system diagnostic information	EIPSystem Stats	Follows data type	-	EIP system diagnostic data structure, which stores the obtained diagnostic information.
	bResult	Execution result	BOOL	TRUE, FALSE	-	Execution result. TRUE: Error FALSE: No error

- **Function**
  - This function block retrieves real-time system-level diagnostics of the selected EIP stack, with data continuously counted internally. A high bClear signal clears historical statistics and restarts counting. The block returns instant sampled data on execution, so valid live data is still accessible when bClear is asserted.
  - The data type of stSystemStats is the EIPSystemStats structure, defined as follows.

Structure member	Data Type	Description
dwMaxActiveIOConnections	DWORD	Max number of I/O connections opened at the same time
dwNumCurrentIOConnections	DWORD	Current number of I/O connections
dwMaxActiveExplicitConnections	DWORD	Number of Class3 connections opened at the same time
dwNumCurrentExplicitConnections	DWORD	Number of currently active Class3 connections
wNumFailedConnections	WORD	Number of failed connection attempts
wNum TimedoutConnections	WORD	Total number of timed out connections
wNumTCPConnectionsUsed	WORD	Total number of established sessions
waxTCPConnectionsUsed	WORD	Maximum number of established sessions at the same time
wNumCurrent TCPConnections	WORD	Current number of established sessions

Structure member	Data Type	Description
dwNumSuccessfulSentlOMessages	DWORD	Total number of successfully sent I/O messages
dwNumSuccessfulRecvIOMessages	DWORD	Total number of successfully received I/O messages
wNum FailedSentlOMessages	WORD	Failed sent I/O packets
wNum FailedRecvlOMessages	WORD	Incorrect I/O data packets received
dwNumSentClass3Messages	DWORD	Total number of Class3 messages sent
dwNumRecvClass3Messages	DWORD	Total number of Class3 messages received
dwNumSentUCMMMessages	DWORD	Total number of UCMM messages sent
dwNumRecvUCMMMessages	DWORD	Total number of UCMM messages received
dwRcvdPPS	DWORD	Receive PPS
dwSendPPS	DWORD	Send PPS
rRecvSpeed	REAL	Receive speed
rSendSpeed	REAL	Send speed

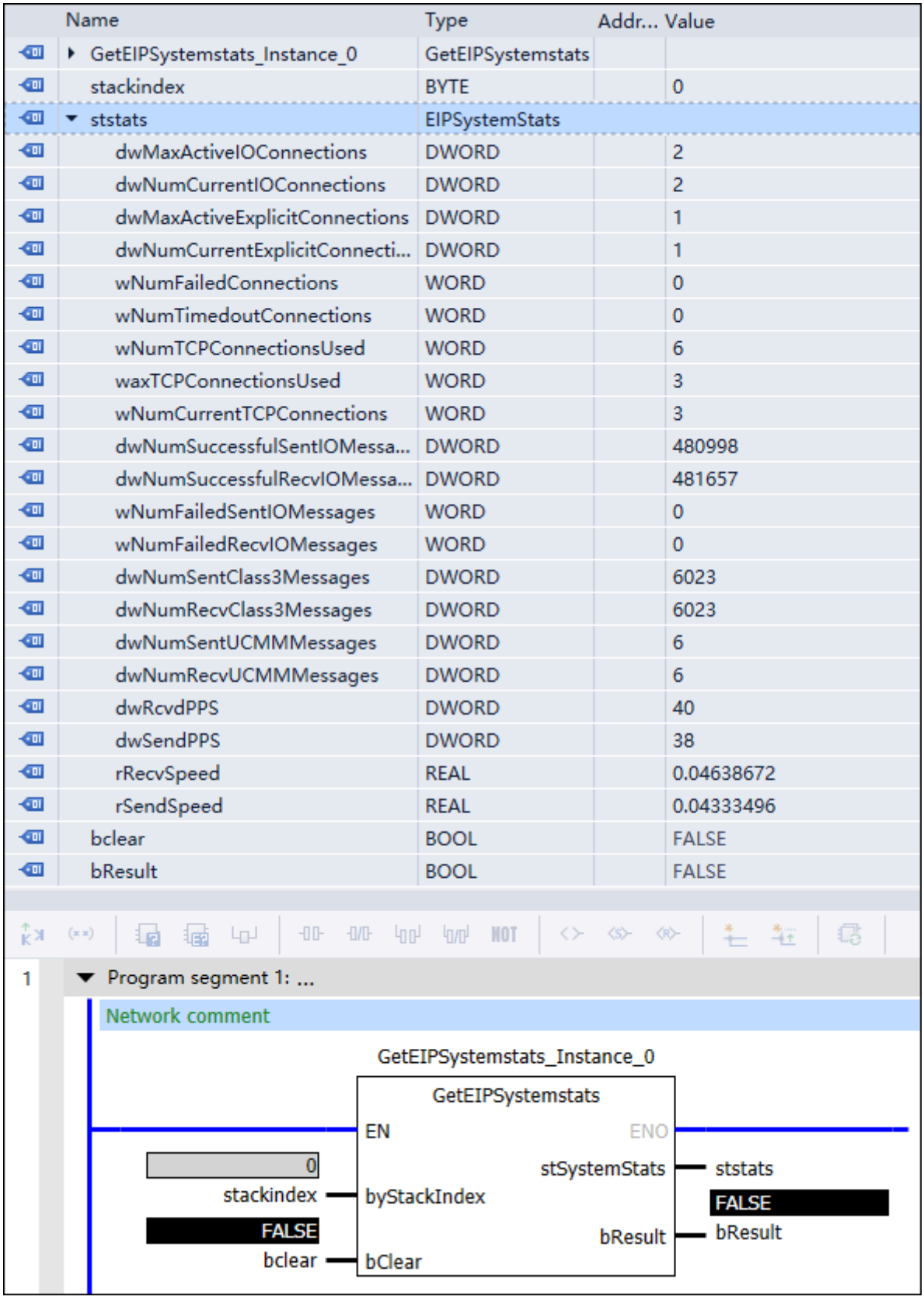
- **Program example**

  ST:

Name	Type	Addr...	Value
GetEIPSystemstats_Instance_0	GetEIPSystemstats		
stackindex	BYTE		0
bclear	BOOL		FALSE
ststats	EIPSystemStats		
dwMaxActiveIOConnections	DWORD		2
dwNumCurrentIOConnections	DWORD		2
dwMaxActiveExplicitConnections	DWORD		1
dwNumCurrentExplicitConnecti...	DWORD		1
wNumFailedConnections	WORD		0
wNumTimedoutConnections	WORD		0
wNumTCPConnectionsUsed	WORD		6
waxTCPConnectionsUsed	WORD		3
wNumCurrentTCPConnections	WORD		2
dwNumSuccessfulSentIOMessa...	DWORD		488622
dwNumSuccessfulRecvIOMessa...	DWORD		489291
wNumFailedSentIOMessages	WORD		0
wNumFailedRecvIOMessages	WORD		0
dwNumSentClass3Messages	DWORD		6118
dwNumRecvClass3Messages	DWORD		6118
dwNumSentUCMMMessages	DWORD		6
dwNumRecvUCMMMessages	DWORD		6
dwRcvdPPS	DWORD		40
dwSendPPS	DWORD		42
rRecvSpeed	REAL		0.049865723
rSendSpeed	REAL		0.044174194
bResult	BOOL		FALSE

```
GetEIPSystemstats_Instance_0( byStackIndex  0   := stackindex  0  ,
                bClear  FALSE  := bclear  FALSE  ,
                stSystemStats => ststats,
                bResult  FALSE  =>bResult  FALSE  );
                RETURN
```

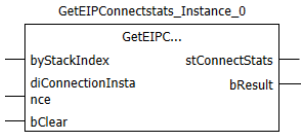
LD:



18.4.3.14 GetEIPConnectstats

This instruction gets the diagnostic information of EIP connection.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
GetEIPConnectstats	Get diagnostic information of EIP connection	FB		GetEIPConnectstats_Instance_0(byStackIndex := , diConnectionInstance := , bClear := , stConnectStats => , bResult =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	byStackIndex	Protocol stack number	BYTE	Follows data type	0	If EIP protocol is added to network ports, the value is 0 for the first port, 1 for the second port, and so on. If EIP is only configured on a single network port, the value is 0.
	diConnectionInstance	Connection ID	DINT	Follows data type	-	Connection instance.
	bClear	Diagnostic data clear	BOOL	TRUE, FALSE	FALSE	Active high. Clears the current diagnostic data.
Output	stConnectStats	Diagnostic information of EIP connection diagnostic	EIPConnectStats	Follows data type	-	EIP connection diagnostic data structure, which stores the obtained diagnostic information.
	bResult	Execution result	BOOL	TRUE, FALSE	-	Execution result. TRUE: Error FALSE: No error

- **Function**
  - This function block retrieves real-time system-level diagnostics of the specified EIP connection, with data continuously counted internally. A high bClear signal clears historical statistics and restarts counting. The block returns instant sampled data on execution, so valid live data is still accessible when bClear is asserted.
  - The data type of stConnectStats is the structure EIPConnectStats, whose meaning is as follows.

Structure member	Data Type	Description
dwSendPackets	DWORD	Number of data packets sent by the connection protocol stack
dwRcydPackets	DWORD	Number of data packets received by the connection protocol stack
rSendAPI	REAL	Actual interval between sent data packets for the connection
rRcvdAPI	REAL	Actual interval between received data packets for the connection
dwMaxSendDelay	DWORD	Maximum delay between any two sent data packets
dwMaxRcvdDelay	DWORD	Maximum delay between any two received data packets
dwLostPackets	DWORD	Number of lost data packets The PacketSequenceNumber of each new data packet increments by 1 to detect packet loss.
wSendPacketError	WORD	Number of data packets that failed to send
wRcydPacketError	WORD	Number of data packets that failed to receive
wNum Timeouts	WORD	Number of connection timeouts, also the number of retry attempts to establish the connection
wNumErrors	WORD	Number of connection errors, also the number of retry attempts to establish the connection
dwLastRcvdTimestamp	DWORD	Timestamp of the last received data packet
dwLastSendTimestamp	DWORD	Timestamp of the last sent data packet
dwSendBytes	DWORD	Number of bytes to transmit
dwRcvdBytes	DWORD	Number of bytes to receive

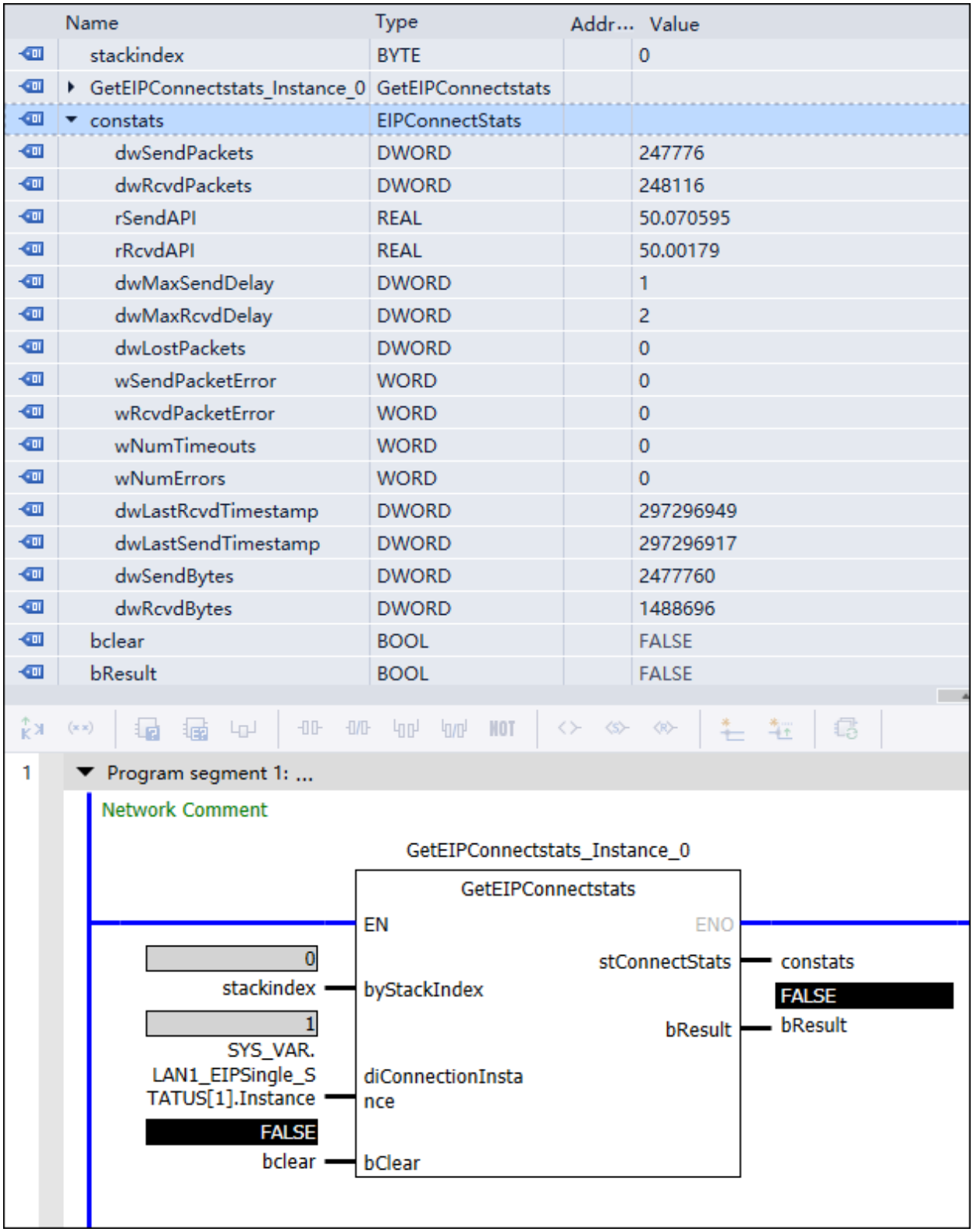
- **Program example**

  ST:

Name	Type	Addr...	Value	Prepared value
stackindex	BYTE		0	
GetEIPConnectstats_Instance_0	GetEIPConnectstats			
constats	EIPConnectStats			
dwSendPackets	DWORD		246516	
dwRcvdPackets	DWORD		246853	
rSendAPI	REAL		50.07047	
rRcvdAPI	REAL		50.00192	
dwMaxSendDelay	DWORD		1	
dwMaxRcvdDelay	DWORD		2	
dwLostPackets	DWORD		0	
wSendPacketError	WORD		0	
wRcvdPacketError	WORD		0	
wNumTimeouts	WORD		0	
wNumErrors	WORD		0	
dwLastRcvdTimestamp	DWORD		297233796	
dwLastSendTimestamp	DWORD		297233819	
dwSendBytes	DWORD		2465160	
dwRcvdBytes	DWORD		1481118	
diConnectionInstance	DINT		0	
bclear	BOOL		FALSE	
bResult	BOOL		FALSE	

```
GetEIPConnectstats_Instance_0(byStackIndex := ,
    byStackIndex   0    := stackindex   0    ,
    diConnectionInstance      1       := SYS_VAR.LAN1_EIPSingle_STATUS[1].Instance      1      ,
    bClear  FALSE   :=bclear  FALSE   ,
    stConnectStats =>constats,
    bResult  FALSE   =>bResult  FALSE   );
    RETURN
```

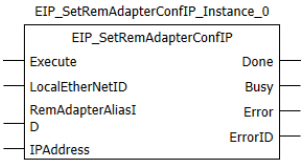
LD:



18.4.3.15 EIP_SetRemAdapterConfIP

This instruction sets the configuration IP address for the specified remote adapter.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
EIP_SetRemAdapterConfIP	Set the connection configuration IP address of the remote adapter	FB		EIP_SetRemAdapterConfIP_Instance_0(Execute := , LocalEtherNetID := , RemAdapterAliasID := , IPAddress := , Done => , Busy => , Error => , ErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Rising edge-triggered	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	LocalEtherNetID	Ethernet port ID	BYTE	1 to 255	0	Ethernet port ID: LAN1 = 1, LAN2 = 2, and so on.
	RemAdapterAliasID	Remote adapter alias ID	UDINT	1 to 2000	0	Adapter alias ID set in the iFA software.
	IPAddress	Configuring IP address	ARRAY[0..3] OF BYTE	[0 to 255, 0 to 255, 0 to 255, 0 to 255]	[0,0,0,0]	IP address configuration follows standard IP specifications, and faults are reported after occurrence.
	Done	Done	BOOL	TRUE, FALSE	FALSE	Instruction is completed.
	Busy	Executing	BOOL	TRUE, FALSE	FALSE	Instruction is executing.
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	Error status.

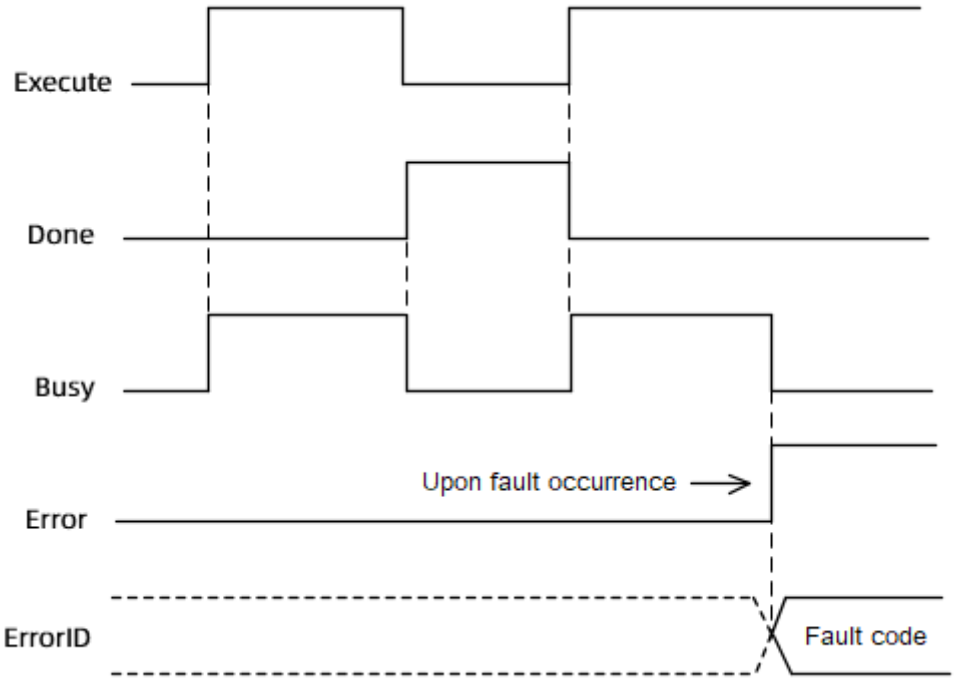
Output

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	ErrorID	Fault Codes	WORD	Follow data type	0	3: Network port ID out of range 5: Remote adapter alias ID out of range 6: Remote adapter IP duplicated

- **Function**

  This instruction specifies the corresponding remote adapter alias ID, network port ID, and IP address to be configured, and sets the configuration IP of the corresponding remote adapter.

- **Timing diagram**



- **Program example**

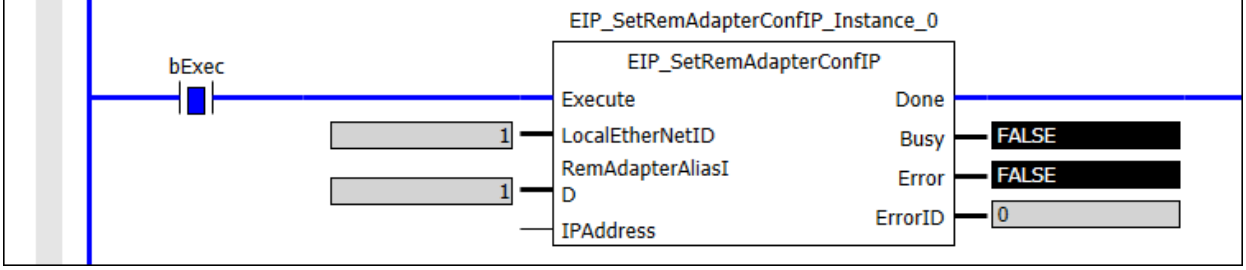
  ST:

Name	Type	Address	Value
EIP_SetRemAdapterConfIP_Instance_0	EIP_SetRemAdapterConfIP		
Execute	BOOL		TRUE
LocalEtherNetID	BYTE		1
RemAdapterAliasID	UDINT		2
IPAddress	ARRAY [0..3] OF BYTE		
IPAddress[0]	BYTE		192
IPAddress[1]	BYTE		168
IPAddress[2]	BYTE		1
IPAddress[3]	BYTE		52
Done	BOOL		TRUE
Busy	BOOL		FALSE
Error	BOOL		FALSE
ErrorID	WORD		0
bExec	BOOL		TRUE
bDone	BOOL		TRUE

```
EIP_SetRemAdapterConfIP_Instance_0(Execute TRUE := bExec TRUE,
                                   LocalEtherNetID := ,
                                   RemAdapterAliasID := ,
                                   IPAddress := ,
                                   Done TRUE => bDone TRUE,
                                   Busy => ,
                                   Error => ,
                                   ErrorID => );
```

LD:

Name	Type	Address	Value
EIP_SetRemAdapterConfIP_Instance_0	EIP_SetRemAdapterConfIP		
Execute	BOOL		TRUE
LocalEtherNetID	BYTE		1
RemAdapterAliasID	UDINT		1
IPAddress	ARRAY [0..3] OF BYTE		
IPAddress[0]	BYTE		192
IPAddress[1]	BYTE		168
IPAddress[2]	BYTE		1
IPAddress[3]	BYTE		50
Done	BOOL		TRUE
Busy	BOOL		FALSE
Error	BOOL		FALSE
ErrorID	WORD		0
bExec	BOOL		TRUE



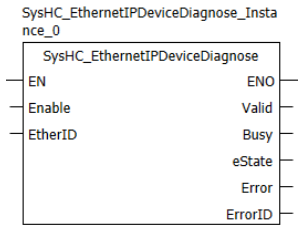
- **Precautions**
  - IP configuration follows standard IP specifications, and errors are reported after occurrence.
  - If LocalEtherNetID or RemAdapterAliasID is not within the PLC project configuration range, Error is set.

- If the IP address set by the instruction is the same as the currently active IP address of the remote adapter, Error is set.
- If the IP address set by the instruction is the same as the currently active IP address of another remote adapter under the same Ethernet port, Error is set.

18.4.3.16 SysHC_EthernetIPDeviceDiagnose

This instruction obtains the fault diagnostics of the PLC when it functions as an EthernetIP slave.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SysHC_EthernetIPDeviceDiagnose	Obtain the fault diagnostics of the EIP slave	FB		SysHC_EthernetIPDeviceDiagnose_Instance_0(Enable := , EtherID := , Valid => , Busy => , eState => , Error => , ErrorID =>);

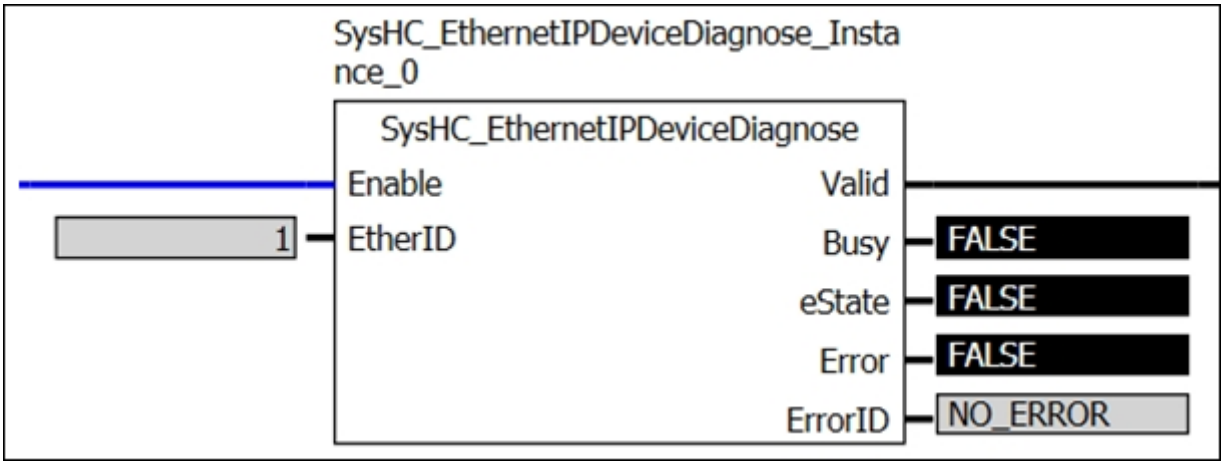
- **Related variables**

Variable

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Enable	High level triggered	BOOL	TRUE, FALSE	FALSE	High level triggered
	EtherID	Network port ID	BYTE	-	0	Network port ID number.
Output	Valid	Execution status	BOOL	TRUE, FALSE	FALSE	Instruction is actively executing.
	Busy	Executing	BOOL	TRUE, FALSE	FALSE	Instruction is executing.
	eState	Diagnostic status bit	BOOL	TRUE, FALSE	FALSE	Diagnostic status bit.
	Error	Error occurred	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	ErrorID	Fault code	SYS_HC_ERROR	-	NO_ERROR	See 18.4.1.37 Fault Codes.

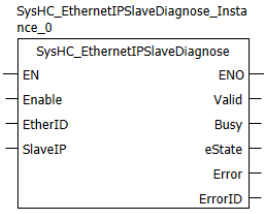
- **Function**
  - When Enable = TRUE, it enters a runnable state, at which point the fault diagnostics can be obtained; when Enable = FALSE, it enters an unrunnable state, and the fault diagnostics cannot be obtained.
  - If no slave is configured, all input EtherID values are invalid, and the error ETHER_ID_ERROR is returned.
  - The EtherID is assigned based on the order of EtherNet/IP-supported ports in the "Hardware Configuration" list of the project tree. For example, for the EVO800 PLC, the EtherIDs of LAN1, LAN2, and EIP3 ports are 1, 2, and 3, respectively; for the EVO523 PLC, if the LAN1 port is in switch mode, the EtherID is 1, and if it is in independent IP mode, the EtherID of the LAN1_A port is 1 and that of the LAN1_B port is 2.
  - If the slave corresponding to the input EtherID is disabled, acquisition of fault diagnosis information fails, and the error FAILURE_OBTAIN is returned.
- **Program example**



18.4.3.17 SysHC_EthernetIPSlaveDiagnose

This instruction obtains the fault diagnostics of the PLC when it functions as an EthernetIP slave that the master connected to.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SysHC_EthernetIPSlaveDiagnose	Obtain the fault diagnostics of the EIP master	FB		SysHC_EthernetIPSlaveDiagnose_Instance_0(Enable := , EtherID := , SlaveIP := , Valid => , Busy => , eState => , Error => , ErrorID =>);

- **Related variables**

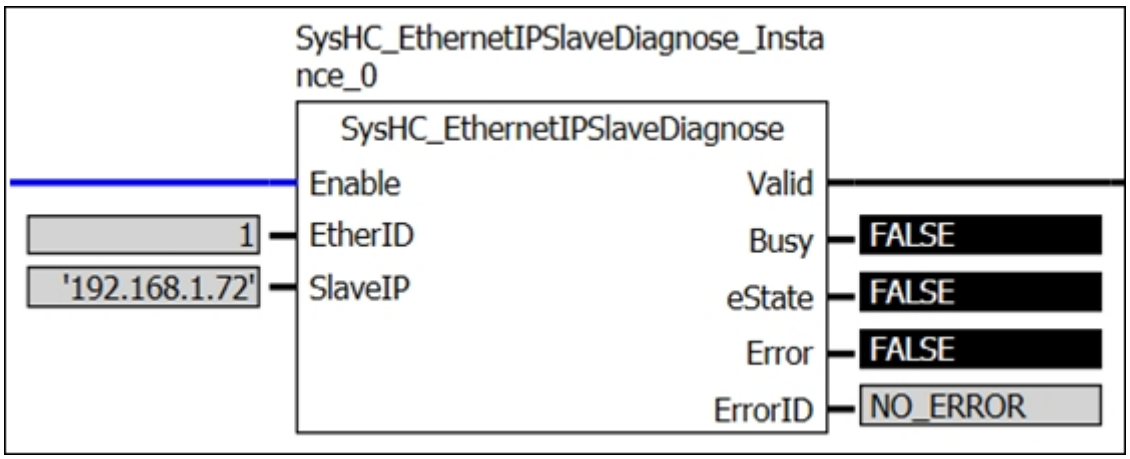
  Variable

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Enable	High level triggered	BOOL	TRUE, FALSE	FALSE	High level triggered
	EtherID	Network port ID	BYTE	-	0	Network port ID number.
	SlaveIP	Slave IP address	STRING	-	-	Slave IP address.
Output	Valid	Execution status	BOOL	TRUE, FALSE	FALSE	Instruction is actively executing.
	Busy	Executing	BOOL	TRUE, FALSE	FALSE	Instruction is executing.
	eState	Diagnostic status bit	BOOL	TRUE, FALSE	FALSE	Diagnostic status bit.
	Error	Error occurred	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs.
	ErrorID	Fault code	SYS_HC_ERROR	-	NO_ERROR	See 18.4.1.37 Fault Codes.

- **Function**
  - When Enable = TRUE, it enters a runnable state, at which point the fault diagnostics can be obtained; when Enable = FALSE, it enters an unrunnable state, and the fault diagnostics cannot be obtained.

- If no slave is configured, all input EtherID values are invalid, and the error ETHER_ID_ERROR is returned.
- The EtherID is assigned based on the order of EtherNet/IP-supported ports in the "Hardware Configuration" list of the project tree. For example, for the EVO800 PLC, the EtherIDs of LAN1, LAN2, and EIP3 ports are 1, 2, and 3, respectively; for the EVO523 PLC, if the LAN1 port is in switch mode, the EtherID is 1, and if it is in independent IP mode, the EtherID of the LAN1_A port is 1 and that of the LAN1_B port is 2.
- If the slave corresponding to the input EtherID is disabled, acquisition of fault diagnosis information fails, and the error FAILURE_OBTAIN is returned.
- If the slave IP address SlaveIP is invalid, or if there is no slave device corresponding to the current IP address, the error Slave_IP_ERROR is returned.

- **Program example**



18.4.3.18 EIP Standard Service Code

Service Code (Decimal)	Service Code (Hexadecimal)	Service Name
1	0x01	Get_Attributes_All
2	0x02	Set_Attributes_All
3	0x03	Get_Attribute_List
4	0x04	Set_Attribute_List
5	0x05	Reset
6	0x06	Start
7	0x07	Stop
8	0x08	Create
9	0x09	Delete
10	0x0A	Multiple_Service_Packet

Service Code (Decimal)	Service Code (Hexadecimal)	Service Name
13	0x0D	Apply Attributes
14	0x0E	Get_Attribute_Single
16	0x10	Set_Attribute_Single
17	0x11	Find_Next_Object_Instance
20	0x14	Error Response (used by DeviceNet only)
21	0x15	Restore
22	0x16	Save
23	0x17	No Operation (NOP)
24	0x18	Get_Member
25	0x19	Set_Member
26	0x1A	Insert_Member
27	0x1B	Remove Member
28	0x1C	Group Sync
29	0x1D	Get_Connection_Point_Member_List

18.4.3.19 EIP Standard Fault Codes

General Fault Code (Hexadecimal)	Extended Fault Code (Hexadecimal)	General Fault Code (decimal)	Extended Fault Code (decimal)	Description	Solution
0x01	0x100	1	256	Connection is in use or duplicate connection is attempted.	-
0x01	0x103	1	259	Transfer type or trigger method is not supported.	Check whether the transfer type and trigger method settings are correct.
0x01	0x105	1	261	Ownership conflict or OUNID mismatch: The configuration is already owned by another initiator.	Check whether the device has already been connected by another device.

General Fault Code (Hexadecimal)	Extended Fault Code (Hexadecimal)	General Fault Code (decimal)	Extended Fault Code (decimal)	Description	Solution
0x01	0x106	1	262	Ownership conflict: Other connections have exclusive access to part of the resources. For example, two PLCs control the output points of an IO module simultaneously.	Check whether multiple masters are connecting to the device simultaneously or whether there is a configuration error.
0x01	0x107	1	263	Target connection is not found. Forward_Close request to close a connection that does not exist on the target node.	-
0x01	0x108	1	264	Network connection parameters, connection type, connection priority are invalid. Redundancy owner, or fixed/variable value device is not supported.	Check whether the connection parameters are set correctly.
0x01	0x109	1	265	Connection size is invalid.	Check whether the connection parameters are set correctly.
0x01	0x110	1	272	Connection target is not configured, and the connection request does not include a data segment for configuration.	Check whether the connection includes configuration data and if the length and content of the configuration data are correct.
0x01	0x111	1	273	RPI is not supported, or the timeout value generated by the connection timeout multiplier is not supported by the device, or the production inhibit time is invalid.	Check whether the RPI, timeout multiplier, and production inhibit time meet the requirements of the target device.

General Fault Code (Hexadecimal)	Extended Fault Code (Hexadecimal)	General Fault Code (decimal)	Extended Fault Code (decimal)	Description	Solution
0x01	0x112	1	274	RPI value is unacceptable. RPI value in the Forward_Open request exceeds the range required by the target device or the target device is producing at a different cycle.	Check whether the RPI meets the target device requirements.
0x01	0x113	1	275	The connection limit has been exceeded, and the target device cannot support any more connections.	Check whether the number of connections to the target device has reached the limit.
0x01	0x114	1	276	Vendor ID or product code does not match.	Check whether the configured device matches the target device.
0x01	0x115	1	277	Product type does not match.	Check whether the configured device matches the target device.
0x01	0x116	1	278	Version does not match.	Check whether the EDS file device version matches the target device, or use the compatibility mode.
0x01	0x117	1	279	Invalid production or consumption path.	Check whether the EDS file is used correctly or whether the parameter configuration of the generic device is correct.
0x01	0x118	1	280	Invalid or inconsistent configuration path.	Check whether the EDS file is used correctly or whether the parameter configuration of the generic device is correct.
0x01	0x119	1	281	A non-ListenOnly connection is not enabled. No non-ListenOnly connection using the same path has been enabled for the ListenOnly connection, or the type of the currently established connection is not multicast.	Check whether other connections with the same path have been established before using ListenOnly, or whether the current connection type with the enabled connection is multicast.

General Fault Code (Hexadecimal)	Extended Fault Code (Hexadecimal)	General Fault Code (decimal)	Extended Fault Code (decimal)	Description	Solution
0x01	0x11A	1	282	Target object exceeds connection limit.	Check whether the number of connections to this object on the target device has reached the limit.
0x01	0x11B	1	283	Production inhibit time is greater than RPI.	Modify production inhibit time to be less than RPI.
0x01	0x11C	1	284	Transfer type is not supported.	Check whether the EDS file is used correctly or whether the parameter configuration of the generic device is correct.
0x01	0x11D	1	285	T → O trigger method is not supported.	Check whether the trigger type is correct.
0x01	0x11E	1	286	Direction specified in the transfer type/trigger method parameters is not supported.	Check whether the EDS file is used correctly or whether the parameter configuration of the generic device is correct.
0x01	0x11F	1	287	O → T fixed/variable flag is not supported.	Check whether the O → T fixed/variable flag setting is correct.
0x01	0x120	1	288	T → O fixed/variable flag is not supported.	Check whether the T → O fixed/variable flag setting is correct.
0x01	0x121	1	289	O → T priority is not supported.	Check whether the O → T priority setting is correct and whether the target device supports this priority.
0x01	0x122	1	290	T → O priority is not supported.	Check whether the T → O priority setting is correct and whether the target device supports this priority.
0x01	0x123	1	291	O → T connection type is not supported.	Check whether the O → T connection type setting is correct.
0x01	0x124	1	292	T → O connection type is not supported.	Check whether the T →O connection type setting is correct.

General Fault Code (Hexadecimal)	Extended Fault Code (Hexadecimal)	General Fault Code (decimal)	Extended Fault Code (decimal)	Description	Solution
0x01	0x125	1	293	O → T redundancy owner flag is not supported.	-
0x01	0x126	1	294	Configuration length is not supported.	Configuration data length setting is incorrect.
0x01	0x127	1	295	O → T connection length is not supported.	Check whether the O → T length is set correctly.
0x01	0x128	1	296	T → O connection length is not supported.	Check whether the T → O length is set correctly.
0x01	0x129	1	297	Configuration path is invalid.	Check whether the configuration instance ID in the connection path is correct.
0x01	0x12A	1	298	Consumption path is invalid (O → T).	Check whether the consumption connection point ID in the connection path is correct.
0x01	0x12B	1	299	Production path is invalid (T → O).	Check whether the production connection point ID in the connection path is correct.
0x01	0x12C	1	300	Configuration symbol does not exist.	Check whether the EDS file is used correctly or whether the parameter configuration of the generic device is correct.
0x01	0x12D	1	301	Consumption symbol does not exist (O->T).	Check whether the EDS file is used correctly or whether the parameter configuration of the generic device is correct.
0x01	0x12E	1	302	Production symbol does not exist (T->O).	Check whether the EDS file is used correctly or whether the parameter configuration of the generic device is correct.
0x01	0x12F	1	303	The combination of configuration consumption and production application paths is not supported.	Check whether the EDS file is used correctly or whether the parameter configuration of the generic device is correct.

General Fault Code (Hexadecimal)	Extended Fault Code (Hexadecimal)	General Fault Code (decimal)	Extended Fault Code (decimal)	Description	Solution
0x01	0x130	1	304	The information in the configuration data segment is inconsistent with the consumer data format.	Check whether the EDS file is used correctly or whether the parameter configuration of the generic device is correct.
0x01	0x131	1	305	The information in the configuration data segment is inconsistent with the producer data format.	Check whether the EDS file is used correctly or whether the parameter configuration of the generic device is correct.
0x01	0x132	1	306	The target does not support empty Forward Open requests.	Check whether the EDS file is used correctly or whether the parameter configuration of the generic device is correct.
0x01	0x133	1	307	Timeout multiplier is not unacceptable.	Check whether the timeout multiplier is set correctly.
0x01	0x134	1	308	T → O connection size does not match the existing multicast connection.	Check whether the current ListenOnly length matches the previous connection.
0x01	0x135	1	309	T → O fixed/variable connection type does not match the existing multicast connection.	Check whether the current ListenOnly connection type matches the previous connection.
0x01	0x136	1	310	T → O connection priority does not match the existing multicast connection.	Check whether the current ListenOnly priority matches the previous connection.
0x01	0x137	1	311	The transfer type does not match the existing multicast connection.	Check whether the current ListenOnly transfer type matches the previous connection.
0x01	0x138	1	312	T → O production trigger mode does not match the existing multicast connection.	Check whether the current ListenOnly trigger type matches the previous connection.
0x01	0x139	1	313	T → O production inhibition time does not match the existing multicast connection.	Check whether the current ListenOnly inhibition time matches the previous connection.

General Fault Code (Hexadecimal)	Extended Fault Code (Hexadecimal)	General Fault Code (decimal)	Extended Fault Code (decimal)	Description	Solution
0x01	0x203	1	515	Connection timeout.	Check the target device status or link status.
0x01	0x204	1	516	Response timeout for non-connected transmission.	-
0x01	0x205	1	517	Parameter error in non-connected transmission service.	-
0x01	0x206	1	518	Non-connected service message is too large.	-
0x01	0x207	1	519	Non-connected consumption service is acknowledged but no response received.	-
0x01	0x301	1	769	No available buffer memory.	-
0x01	0x302	1	770	Unable to allocate the specified bandwidth for the data.	-
0x01	0x303	1	771	No available consumed connection ID filter.	-
0x01	0x304	1	772	Not configured to send scheduled priority data.	-
0x01	0x311	1	785	Port is not available.	-
0x01	0x312	1	786	Link address is invalid.	-
0x01	0x315	1	789	There is an invalid segment in the connection path.	-
0x01	0x317	1	791	Schedule is not specified.	-
0x01	0x318	1	792	Local link address is invalid.	-
0x01	0x319	1	793	Backup resources are not available.	-
0x01	0x31A	1	794	Rack connection has been established.	-
0x01	0x31B	1	795	Module connection has been established.	-

General Fault Code (Hexadecimal)	Extended Fault Code (Hexadecimal)	General Fault Code (decimal)	Extended Fault Code (decimal)	Description	Solution
0x01	0x31C	1	796	Miscellaneous: No other extended status code can indicate the connection error.	-
0x01	0x31D	1	797	Redundant connection mismatch.	-
0x01	0x31E	1	798	There are no more available user-configurable link consumer resources in the production module.	-
0x01	0x31F	1	799	There are no available user-configurable link consumer resources in the production module.	-
0x01	0x800	1	2048	Target device is offline.	Check whether the EDS file is used correctly or whether the parameter configuration of the generic device is correct.
0x01	0x801	1	2049	Incompatible multicast time correction PRI. Connection has been established with a different time correction RPI.	Check whether all devices connected to this device are configured with the same time correction RPI.
0x01	0x802	1	2050	Incorrect security connection size.	Check whether the EDS is used correctly and whether the correct connection is selected.
0x01	0x803	1	2051	Incorrect security connection format.	Check whether the EDS is used correctly and whether the correct connection is selected.
0x01	0x804	1	2052	Invalid time correction connection parameters.	Check whether the EDS is used correctly and whether the correct connection is selected.
0x01	0x805	1	2053	Incorrect EPI multiplier for Ping interval.	Check whether the correct timeout multiplier is selected.

General Fault Code (Hexadecimal)	Extended Fault Code (Hexadecimal)	General Fault Code (decimal)	Extended Fault Code (decimal)	Description	Solution
0x01	0x806	1	2054	Incorrect minimum multiplier for time coordination messages.	Check whether the EDS is used correctly and whether the correct connection is selected.
0x01	0x807	1	2055	Incorrect multiplier for network expected response time.	Check whether the EDS is used correctly and whether the correct connection is selected.
0x01	0x808	1	2056	Incorrect timeout multiplier.	Check whether the EDS is used correctly and whether the correct connection is selected.
0x01	0x809	1	2057	Incorrect maximum number of consumers.	Check whether the EDS is used correctly and whether the correct connection is selected.
0x01	0x80A	1	2058	Incorrect CPCRC.	Check whether the EDS is used correctly and whether the correct connection is selected.
0x01	0x80B	1	2059	Invalid time correction connection ID.	Check whether the EDS is used correctly and whether the correct connection is selected.
0x01	0x80C	1	2060	SCID mismatch.	SCID is not 0 or does not match the target. No additional handling is required in Type1 mode. In Type2A mode, check whether the configured SCID does not match the target; if so, modify it to match.
0x01	0x80D	1	2061	Unset TUNID.	Configure TUNID through the back-end software.
0x01	0x80E	1	2062	TUNID mismatch.	Reset the TUNID of the target device through the backend software, and reconfigure the TUNID.

General Fault Code (Hexadecimal)	Extended Fault Code (Hexadecimal)	General Fault Code (decimal)	Extended Fault Code (decimal)	Description	Solution
0x01	0x80F	1	2063	Configuration operation is not allowed.	-
0x01	0x813	1	2067	Not configured to support cross-subnet multicast.	-
0x01	0x815	1	2069	Incompatible EPI multiplier for multicast ping interval. A connection has been established using incompatible EPI multipliers for Ping intervals.	Check whether all devices connected to this device are configured with the same connection parameters.
0x01	0x816	1	2070	Incompatible multicast maximum consumer count. A connection has been established with incompatible maximum consumer count.	Check whether all devices connected to this device are configured with the same connection parameters.
0x01	0x817	1	2071	Incompatible multicast security network segment types. A connection has been established with a different security network segment types.	Check whether all devices connected to this device are configured with the same connection parameters.
0x02	0xFFFF	2	65535	Resource unavailable - Internal processing buffer unavailable.	-
0x03	0xFFFF	3	65535	Invalid parameter value.	-
0x04	0xFFFF	4	65535	Incorrect path format.	-
0x05	0xFFFF	5	65535	Target path is unknown. Check whether the request path is correct.	-
0x06	0xFFFF	6	65535	Partial transfer: Only part of the expected data was transferred.	-
0x07	0xFFFF	7	65535	Network disconnection.	-
0x08	0xFFFF	8	65535	Service is not supported.	-
0x09	0xFFFF	9	65535	Invalid property data.	-

General Fault Code (Hexadecimal)	Extended Fault Code (Hexadecimal)	General Fault Code (decimal)	Extended Fault Code (decimal)	Description	Solution
0x0A	0xFFFF	10	65535	Attributes in the Get_Attribute_List or Set_Attribute_List response have a non-zero status..	-
0x0B	0xFFFF	11	65535	Object is already in the service request status.	-
0x0C	0xFFFF	12	65535	Object cannot perform the requested service in its current mode/status.	-
0x0D	0xFFFF	13	65535	Requested instance for object creation already exists.	-
0x0E	0xFFFF	14	65535	Attribute cannot be set.	-
0x0F	0xFFFF	15	65535	Permission/authorization check failed.	-
0x10	0xFFFF	16	65535	The current mode/status of the device prohibits the requested service from being executed.	-
0x11	0xFFFF	17	65535	Response data is too large.	-
0x12	0xFFFF	18	65535	The base type is fragmented. This service fragments the base data value, i.e., it decomposes half of the REAL data.	-
0x13	0xFFFF	19	65535	The service does not provide enough data to perform the specified operation.	-
0x14	0xFFFF	20	65535	Attribute is not supported.	-
0x15	0xFFFF	21	65535	The service provides more data than expected.	-
0x16	0xFFFF	22	65535	Object instance does not exist.	-
0x17	0xFFFF	23	65535	There is a sequence-related error in the fragmentation processing of the service.	-

General Fault Code (Hexadecimal)	Extended Fault Code (Hexadecimal)	General Fault Code (decimal)	Extended Fault Code (decimal)	Description	Solution
0x18	0xFFFF	24	65535	The attribute value was not saved before the request.	-
0x19	0xFFFF	25	65535	The attribute value failed to be saved due to a fault during the attempt.	-
0x1A	0xFFFF	26	65535	Routing error: The request packet is too large.	-
0x1B	0xFFFF	27	65535	Routing error: The response packet is too large.	-
0x1C	0xFFFF	28	65535	Attribute list data is missing or incomplete.	-
0x1D	0xFFFF	29	65535	Invalid attribute value list.	-
0x1E	0xFFFF	30	65535	Embedded system error: Multiple service responses contain errors.	-
0x1F	0xFFFF	31	65535	Vendor-specific error. See the target device guide.	-
0x20	0xFFFF	32	65535	Invalid parameter: Parameter does not meet specification requirements.	-
0x21	0xFFFF	33	65535	An attempt was made to perform a write operation on a write-once medium (e.g., WORM drive, PROM) to which data has already been written, or to modify a value that cannot be changed once set.	-
0x22	0xFFFF	34	65535	Invalid response received.	-
0x25	0xFFFF	37	65535	Electronic matching item error in the path.	-
0x25	0x114	37	276	Vendor ID or product code does not match.	-
0x25	0x115	37	277	Product type does not match.	-
0x25	0x116	37	278	Version does not match.	-
0x26	0xFFFF	38	65535	Invalid path size.	-

General Fault Code (Hexadecimal)	Extended Fault Code (Hexadecimal)	General Fault Code (decimal)	Extended Fault Code (decimal)	Description	Solution
0x27	0xFFFF	39	65535	Attempt to set an attribute that is currently immutable.	-
0x28	0xFFFF	40	65535	Invalid member ID.	-
0x29	0xFFFF	41	65535	Member cannot be set.	-
0x2B	0xFFFF	43	65535	Unknown Modbus error.	-
0x2C	0xFFFF	44	65535	Attribute cannot be obtained.	-
0x2D	0xFFFF	45	65535	Instance cannot be deleted.	-
0x6A	0xFFFF	106	65535	Encapsulation protocol header contains an error.	-
0xFF	0x2105	255	8453	Specified 0 elements or data out of array range for an array.	Check whether the tag read/write length is correct.
0xFF	0x2107	255	8455	Tag type used in the request does not match the data type of the target tag.	Check whether the tag read/write parameters are correct and whether the data types match.
0x1F	0x8007	31	32775	An inaccessible variable is specified.	Check whether access to the variable on the target device is allowed.
0x1F	0x0102	31	258 8451	Attempt to write to a read-only variable or constant.	Check whether access to the variable on the target device is allowed.
0x1F	0x0104	31	260 4355	Specified length exceeds actual length.	Check whether the tag read/write length is correct.
0x20	0x800F	32	32783	Data length information in the request data is inconsistent.	Check whether the data types of the two devices for tag read/write are consistent.
0x20	0x8017	32	32791	Multiple elements were requested for a variable that is not an array.	Check whether the tag read/write parameters are correct.

General Fault Code (Hexadecimal)	Extended Fault Code (Hexadecimal)	General Fault Code (decimal)	Extended Fault Code (decimal)	Description	Solution
0x20	0x8028	32	32808	An attempt was made to write a value outside the specified range to a variable with a defined subrange, or to write an undefined value to an enumerated variable.	Check whether the tag read/write parameters are correct.
0xFF	0x3108	255	12552	Failed to read master tag variable. Check the tag variables configured for the master and slaves.	Check whether the tag read/write parameters are correct.
0xFF	0x3109	255	12553	Failed to read the length of the master tag variable. Check the tag variables configured for the master and slaves.	Check whether the tag read/write parameters are correct.
0xFF	0x310A	255	12554	Tag variable type is not supported.	Check whether the tag read/write parameters are correct.
0xFF	0x310B	255	12555	Master tag variable type mismatch during read operation.	Check whether the tag read/write parameters are correct.
0xFF	0x1000	255	4096	Redundant mode does not support sending explicit messages from the partner rack.	Disconnect the connection to this device.
0xFF	0x800	255	2048	Request timeout: Target device is offline.	Check whether the network is properly connected and whether the target device is online.
0xFF	0xFFFF	255	65535	Vendor-specific error. See the target device guide.	-
0xFF	0x2000	255	8192	The I/O coupler rejects the connection. The specific diagnostic code must be read.	Troubleshoot based on the specific error code.
0xFF	0x2001	255	8193	The I/O coupler rejects the connection due to mismatched configuration.	Check whether the configuration matches the actual device.

General Fault Code (Hexadecimal)	Extended Fault Code (Hexadecimal)	General Fault Code (decimal)	Extended Fault Code (decimal)	Description	Solution
0xFF	0x2002	255	8194	The I/O coupler rejects the connection because the configuration does not match the actual device.	Check whether the configuration matches the actual device.
0xFF	0x2003	255	8195	The IO coupler rejects the connection because the module parameter validation failed.	Check whether the configuration data is correct.

18.4.3.20 Fault Codes

Fault Code (Decimal)	Fault Code (Hexadecimal)	Description
61440	F000	More than 32 instances
61441	F001	Network configuration error
61442	F002	Failure to create EIP protocol stack
61443	F003	EIP protocol stack not running
61444	F004	Trigger signal error
61445	F005	Request path error
61446	F006	Connection error
61447	F007	Path conversion error
61448	F008	Error in sending class3 request
61449	F009	Data transmission exceeds limit
61450	F00A	Protocol stack shutdown error or repeated shutdown
61451	F00B	Read request sending error or repeated request
61452	F00C	Write request sending error or repeated request
61453	F00D	Waiting time after timeout too short
61454	F00E	Exceeded open connection data limit
61455	F00F	BOOL type copy error
61456	F010	Insufficient memory
61457	F011	0 element
61458	F012	Protocol stack error
61459	F013	Unsupported data type
61460	F014	Timeout period limit exceeded

Fault Code (Decimal)	Fault Code (Hexadecimal)	Description
61461	F015	Size limit of the request data exceeded
61462	F016	Array access limit exceeded
61463	F017	Received data limit exceeded

18.4.4 Local Bus

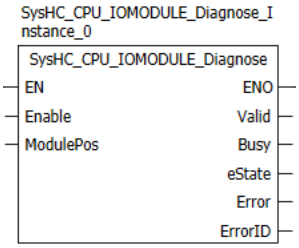
18.4.4.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.4.4.2 SysHC_CPU_IOMODULE_Diagnose	FB	Get the diagnostic state of local IO module	SysHCPlcInfo	Library version 1.0.0.0 and later
18.4.4.3 LBus_GetDiagMailBoxData	FB	Get diagnostic information for modules on the local bus	ExtendIo	Library version 1.1.0.0 and later
18.4.4.4 LBus_GetModuleInfo	FB	Get all information about the local bus and local I/O		Library version 1.1.0.0 and later
18.4.4.5 SysHC_GetAllGL20ModuleInfo	FB	Get basic information about all modules on the local bus		Library version 1.1.0.0 and later
18.4.4.6 SysHC_GetGL20ModuleInfo	FB	Get basic information about a specified module on the local bus		Library version 1.1.0.0 and later

18.4.4.2 SysHC_CPU_IOMODULE_Diagnose

This instruction gets the diagnostic state information of the local extended IO module.

- **Instruction format**

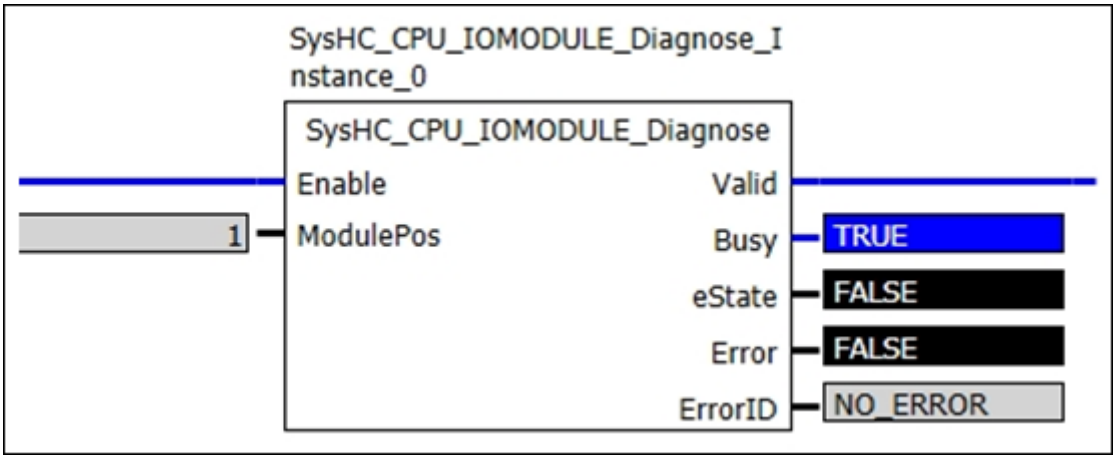
Instruction	Name	FB/FC	LD Representation	ST Representation
SysHC_CPU_IOMODULE_Diagnose	Get the diagnostic state of local IO module	FB		SysHC_CPU_IOMODULE_Diagnose_Instance_0(Enable := , ModulePos := , Valid => , Busy => , eState => , Error => , ErrorID =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Enable	Active high	BOOL	TRUE, FALSE	FALSE	Active high.
	ModulePos	Module slot number	BYTE	Follows data type	0	Module slot number.
Output	Valid	Output active	BOOL	TRUE, FALSE	FALSE	Instruction is actively executing.
	Busy	Instruction executing	BOOL	TRUE, FALSE	FALSE	Instruction is executing.
	eState	Diagnostic status bit	BOOL	TRUE, FALSE	FALSE	Diagnostic status bit.
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs.
	ErrorID	Fault code	SYS_HC_ERROR	-	NO_ERROR	See 18.4.1.37 Fault Codes.

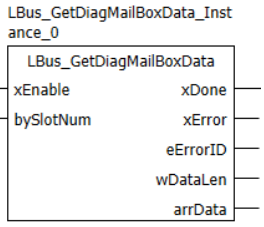
- **Function**
  - Enable set to TRUE indicates that the function block is enabled; otherwise, all outputs are invalid.
  - Checks if the ModulePos is within the valid range. If out of range, ErrorID outputs WRONG_PARAMETER.
  - If the ModulePos is successfully matched, check the disabled status of the slave node. If disabled, ErrorID outputs FAILURE_OBATION.
  - After the parameters are matched, the communication diagnostic data of the module specified by ModulePos is acquired, and the eState is updated in real time.
- **Program example**



18.4.4.3 LBus_GetDiagMailBoxData

This instruction reads diagnostic information from the local expansion module.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
LBus_GetDiagMailBoxData	Get diagnostic mailbox information of the specified module on the local bus	FB		LBus_GetDiagMailBoxData _Instance_0(xEnable := , bySlotNum := , xDone => , xError => , eErrorID => , wDataLen => , arrData =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Function block enable	BOOL	TRUE, FALSE	FALSE	Enable bit of the function block When the value of pin is TRUE, diagnostic information for the corresponding slot is read periodically.
	bySlotNum	Slot number	BYTE	[0, 31]	0	Set the slot number for which diagnostic information is to be read.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	xDone	Function block execution completion flag	BOOL	TRUE, FALSE	FALSE	Function block execution completion flag When the value of pin is TRUE, diagnostic information reading is completed.
	xError	Function block error flag	BOOL	TRUE, FALSE	FALSE	Function block fault flag.
	eErrorID	Fault code	LBUS_ERROR	-	NO_ERROR	Fault code of function block. For detail, see Fault Codes.
	wDataLen	Diagnostic data length	WORD	[0, 255]	0	Indicates the diagnostic data length of the module.
	arrData	Diagnostic mailbox data	ARRAY [0..255] OF BYTE	-	-	Indicates the diagnostic data of the module.

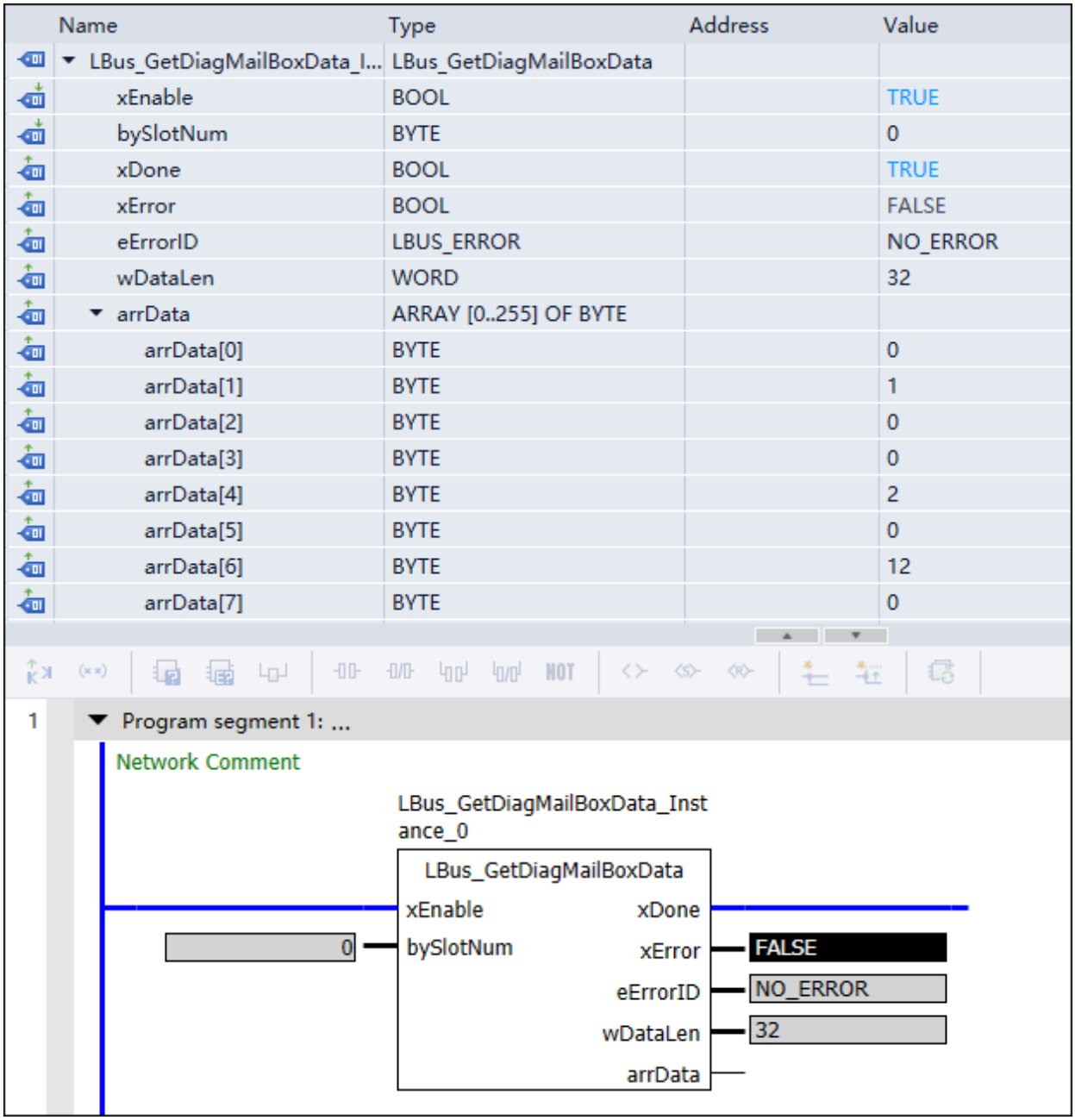
- **Function**
  This instruction reads the diagnostic mailbox data of the specified module on the local bus. After setting the slot number, enable the function block to read diagnostic data of the module.
- **Program example**

  ST:

Name	Type	Address	Value
LBus_GetDiagMailBoxData_I...	LBus_GetDiagMailBoxData		
xEnable	BOOL		TRUE
bySlotNum	BYTE		0
xDone	BOOL		TRUE
xError	BOOL		FALSE
eErrorID	LBUS_ERROR		NO_ERROR
wDataLen	WORD		32
arrData	ARRAY [0..255] OF BYTE		
arrData[0]	BYTE		0
arrData[1]	BYTE		1
arrData[2]	BYTE		0
arrData[3]	BYTE		0
arrData[4]	BYTE		2
arrData[5]	BYTE		0
arrData[6]	BYTE		12
arrData[7]	BYTE		0
arrData[8]	BYTE		0
arrData[9]	BYTE		0
arrData[10]	BYTE		0
arrData[11]	BYTE		0
arrData[12]	BYTE		0
arrData[13]	BYTE		0
arrData[14]	BYTE		0

```
LBus_GetDiagMailBoxData_Instance_0(xEnable := true,
                                   bySlotNum := 0,
                                   xDone => ,
                                   xError => ,
                                   eErrorID => ,
                                   wDataLen => ,
                                   arrData => );
```

LD:



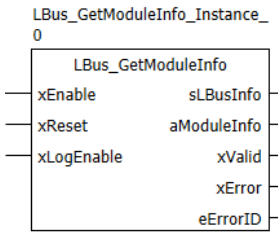
- **Note**

  The input parameter bySlotNum is used to set the slot number for which information is to be read. If the slot number is out of the valid range, or if the slot number is within range but no module is actually mounted on the slot, the function block will report a slot number error.

18.4.4.4 LBus_GetModuleInfo

This instruction gets the status information of the PLC local bus and the modules mounted on the local bus.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
LBus_GetModuleInfo	Get all information about the local bus and local I/O	FB		LBus_GetModuleInfo_Instance_0(xEnable := , xReset := , xLogEnable := , sLBusInfo => , aModuleInfo => , xValid => , xError => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Instruction enable	BOOL	TRUE, FALSE	FALSE	Enable bit of the function block. When the value of pin is TRUE, the local module information is read periodically.
	xReset	Reset output information	BOOL	TRUE, FALSE	FALSE	Pin to reset the module output information. When the value of pin is TRUE, the information in sLBusInfo and aModuleInfo is cleared, and the statistics of the local bus restart counting.
	xLogEnable	Enable debug log	BOOL	TRUE, FALSE	FALSE	Enable switch for the local bus debug log. When enabled, the low-level log information of the local modules is displayed.
Output	sLBusInfo	Bus status information	LBus_Stats	Follows data type	-	This data structure stores the operational status information of the local bus and the underlying thread.
	aModuleInfo	Module status information	ARRAY [0..31] OF LBus_ModuleStats	Follows data type	-	This data structure stores the status information of the local modules.
	xValid	Valid instruction	BOOL	TRUE, FALSE	FALSE	Data valid flag.
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Function block fault flag.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	eErrorID	Fault code	LBUS_ERROR	Follows data type	NO_ERROR	Fault code of function block. For details, see Fault Codes.

- **Function**
  - This function block obtains the status of the local bus and the status information of all modules mounted on the local bus.
  - The xReset signal only resets part of the statistics information in the protocol stack, not the protocol stack itself.
  - xReady can be used as a criterion to determine whether module data is valid. When its value is TRUE, the module I/O data is definitely valid.
  - The data type of sLBusInfo is the structure LBus_Stats, whose meaning is as follows.

Structure member	Name	Data Type	Description
xReady	Ready signal of all modules	BOOL	Status signal of all modules. When its value is TRUE, the input and output data of all modules are safe and available.
xLogStatus	Bus log status	BOOL	Status of the local bus debug log, determined by the input parameter xLogEnable.
eRunStatus	Bus running status	LBUS_RUN_STATUS	Running status of the local bus thread. 0: S_STOP (stopped). 1: S_RUN (running). 2: S_EXIT (exiting).
eBusStatus	Bus communication status	LBUS_COM_STATUS	Status of the local bus protocol stack. 0: RESET (reset state). 1: INIT (initialization state). 2: PREOP (pre-operational state). 4: SAFEOP (safe operational state). 8: OP (operational state). 128: ERROR (fault state).
bySlotNum	Number of the configured modules	BYTE	Number of modules configured on the local bus in the PLC programming software
byBusNum	Actual number of modules mounted on the bus	BYTE	Actual number of modules mounted on the local bus.

Structure member	Name	Data Type	Description
sTaskStats	Bus task information	LBus_TaskStats	Running status of the local bus thread.
sBusStats	Bus communication information	LBus_BusStats	Running status of the local bus protocol stack.

Table18-4-4-4-1 Description of the LBus_TaskStats struct type

Structure member	Name	Data Type	Description
dwInterval	Task interval	DWORD	Local bus task cycle configured in the PLC programming back end. Unit: μs.
dwCurCycleTime	Current task cycle	DWORD	Current actual running cycle in μs.
dwMinCycleTime	Minimum task cycle	DWORD	Minimum running cycle in μs.
dwMaxCycleTime	Maximum task cycle	DWORD	Maximum running cycle in μs.
dwCurRunTime	Current runtime	DWORD	Actual runtime of the local bus protocol stack in μs.
dwMinRunTime	Minimum runtime	DWORD	Minimum runtime of the local bus protocol stack in μs.
dwMaxRunTime	Maximum runtime	DWORD	Maximum runtime of the local bus protocol stack in μs.
dwLossCycleCount	Lost cycle count	DWORD	Statistical count of lost cycles for local bus tasks.

Table18-4-4-4-2 Description of the LBus_BusStats struct type

Structure member	Name	Data Type	Description
dwMailSendAddressLost	Addressed mail lost	DWORD	Count of addressed mail lost.
dwMailSendOpenDaisyChainLost	Daisy chain mail lost	DWORD	Count of lost daisy chain mail.
dwMailRecyModuleInfoLost	Mail for retrieving module information lost	DWORD	Count of lost mail for retrieving module information.
dwMailSendModuleConfigLost	Mail for setting module configuration data lost	DWORD	Count of lost mail for setting module configuration data.
dwMailRecvModuleConfigLost	Mail for retrieving module configuration data lost	DWORD	Count of lost mail for retrieving module configuration data.
dwMailRecvModuleStatusLost	Mail for retrieving module status lost	DWORD	Count of lost mail for retrieving module status.

Structure member	Name	Data Type	Description
dwMailSendModuleStatelost	Mail for setting module status lost	DWORD	Count of lost mail for setting module status.
dwMailRecvModuleDiagInfoLost	Mail for retrieving module diagnostic information lost	DWORD	Count of lost mail for retrieving module diagnostic information.
dwSyncSendFrameTotalCount	Total frames in module communication	DWORD	Count of total frames in module communication.
dwSyncFrameLostCount	Frame loss count per module	ARRAY[0..31] OF DWORD	Frame loss count per module.

◦ The data type of aModuleInfo is ARRAY [0..31] OF LBus_ModuleStats. The structure LBus_ModuleStats is defined as follows.

Structure member	Name	Data Type	Description
byIndex	Module position index	BYTE	Slot information of the module, which corresponds to the configured slot information.
xError	Module error status	BOOL	Fault flag of the module.
xDisable	Module disabled status	BOOL	Disable flag of the module.
dwConfigModule	Configuring module type	DWORD	Module type configured for the corresponding slot on the local bus in the PLC programming software.
dwMountedModule	Mounted module type	DWORD	Actual module type mounted on the corresponding slot on the local bus.
dwLogicVersion	Module logic version	DWORD	The logic software version of the module.
dwSoftwareVersion	Module software version	DWORD	The MCU software version of the module.
wModuleErrorID	Module fault code	WORD	Module fault code.
aChannelErrorID	Fault code of module channel	ARRAY [0..31] OF WORD	Fault code of channel error.

- **Program example**

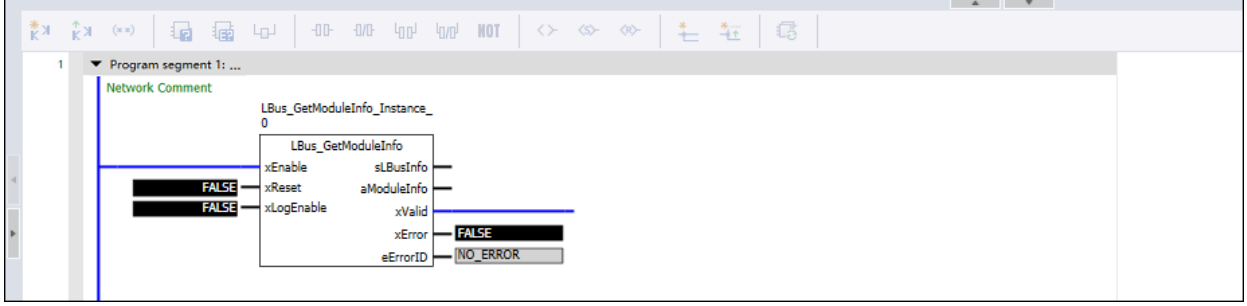
  ST:

	Name	Type	Address	Value
1	LBus_GetModuleInfo_Instance_0	LBus_GetModuleInfo		
2	xEnable	BOOL		TRUE
3	xReset	BOOL		FALSE
4	xLogEnable	BOOL		FALSE
5	sLBusInfo	LBus_Stats		
6	xReady	BOOL		TRUE
7	xLogStatus	BOOL		FALSE
8	eRunStatus	LBUS_RUN_STATUS		S_RUN
9	eBusStatus	LBUS_COM_STATUS		OP
10	bySlotNum	BYTE		4
11	byBusNum	BYTE		4
12	sTaskStats	LBus_TaskStats		
13	dwInterval	DWORD		500
14	dwCurCycleTime	DWORD		556
15	dwMinCycleTime	DWORD		506
16	dwMaxCycleTime	DWORD		614
17	dwCurRunTime	DWORD		52
18	dwMinRunTime	DWORD		2
19	dwMaxRunTime	DWORD		105
20	dwLossCycleCount	DWORD		0
21	sBusStats	LBus_BusStats		
22	aModuleInfo	ARRAY [0..31] OF LBus_ModuleStats		
23	xValid	BOOL		TRUE
24	xError	BOOL		FALSE
25	eErrorID	LBUS_ERROR		NO_ERROR

```
IF  CAS  FOR  WHI  REP  RET  EX  REG
1 LBus_GetModuleInfo_Instance_0(xEnable TRUE  := TRUE,
2                               xReset FALSE  := FALSE,
3                               xLogEnable FALSE  := FALSE,
4                               sLBusInfo => ,
5                               aModuleInfo => ,
6                               xValid => ,
7                               xError => ,
8                               eErrorID => );
```

LD:

	Name	Type	Address	Value
1	LBus_GetModuleInfo_Instance_0	LBus_GetModuleInfo		
2	xEnable	BOOL		TRUE
3	xReset	BOOL		FALSE
4	xLogEnable	BOOL		FALSE
5	sLBusInfo	LBus_Stats		
6	xReady	BOOL		TRUE
7	xLogStatus	BOOL		FALSE
8	eRunStatus	LBUS_RUN_STATUS		S_RUN
9	eBusStatus	LBUS_COM_STATUS		OP
10	bySlotNum	BYTE		4
11	byBusNum	BYTE		4
12	sTaskStats	LBus_TaskStats		
13	dwInterval	DWORD		500
14	dwCurCycleTime	DWORD		556
15	dwMinCycleTime	DWORD		506
16	dwMaxCycleTime	DWORD		614
17	dwCurRunTime	DWORD		50
18	dwMinRunTime	DWORD		2
19	dwMaxRunTime	DWORD		105
20	dwLossCycleCount	DWORD		0
21	sBusStats	LBus_BusStats		
22	aModuleInfo	ARRAY [0..31] OF LBus_ModuleStats		
23	xValid	BOOL		TRUE
24	xError	BOOL		FALSE
25	eErrorID	LBUS_ERROR		NO_ERROR

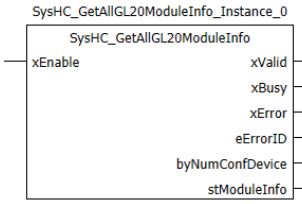


- **Note:**
  xLogEnable is used to control debug log printing on the local bus. Disable it during normal operation.

18.4.4.5 SysHC_GetAllGL20ModuleInfo

This instruction gets the status information of all modules on the local bus of the PLC.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SysHC_GetAllGL20ModuleInfo	This instruction gets basic information about all modules on the local bus.	FB		SysHC_GetAllGL20ModuleInfo_Instance_0(xEnable := , xValid => , xBusy => , xError => , eErrorID => , byNumConfDevice => , stModuleInfo =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Function block enable	BOOL	TRUE, FALSE	FALSE	Enable bit of the function block When the value of pin is TRUE, the local module information is read periodically.
Output	xValid	Valid function block output	BOOL	TRUE, FALSE	FALSE	Function block data valid flag. When the value of pin is TRUE, the read information is valid.
	xBusy	Function block executing	BOOL	TRUE, FALSE	FALSE	Function block execution flag. When the value of pin is TRUE, the function block is reading information.
	xError	Function block execution error	BOOL	TRUE, FALSE	FALSE	Function block fault flag.
	eErrorID	Fault code of function block	SYS_HC_ERROR	Follows data type	LB_NO_ERROR	Fault code of function block. For details, see Fault Codes.
	byNumConfDevice	Number of Modules	BYTE	[0, 31]	0	Actual number of modules configured in the project.
	stModuleInfo	Module status information	ARRAY [0..31] OF GL20_MODULE_INFO	Follows data type	-	This data structure stores the status information of the local modules.

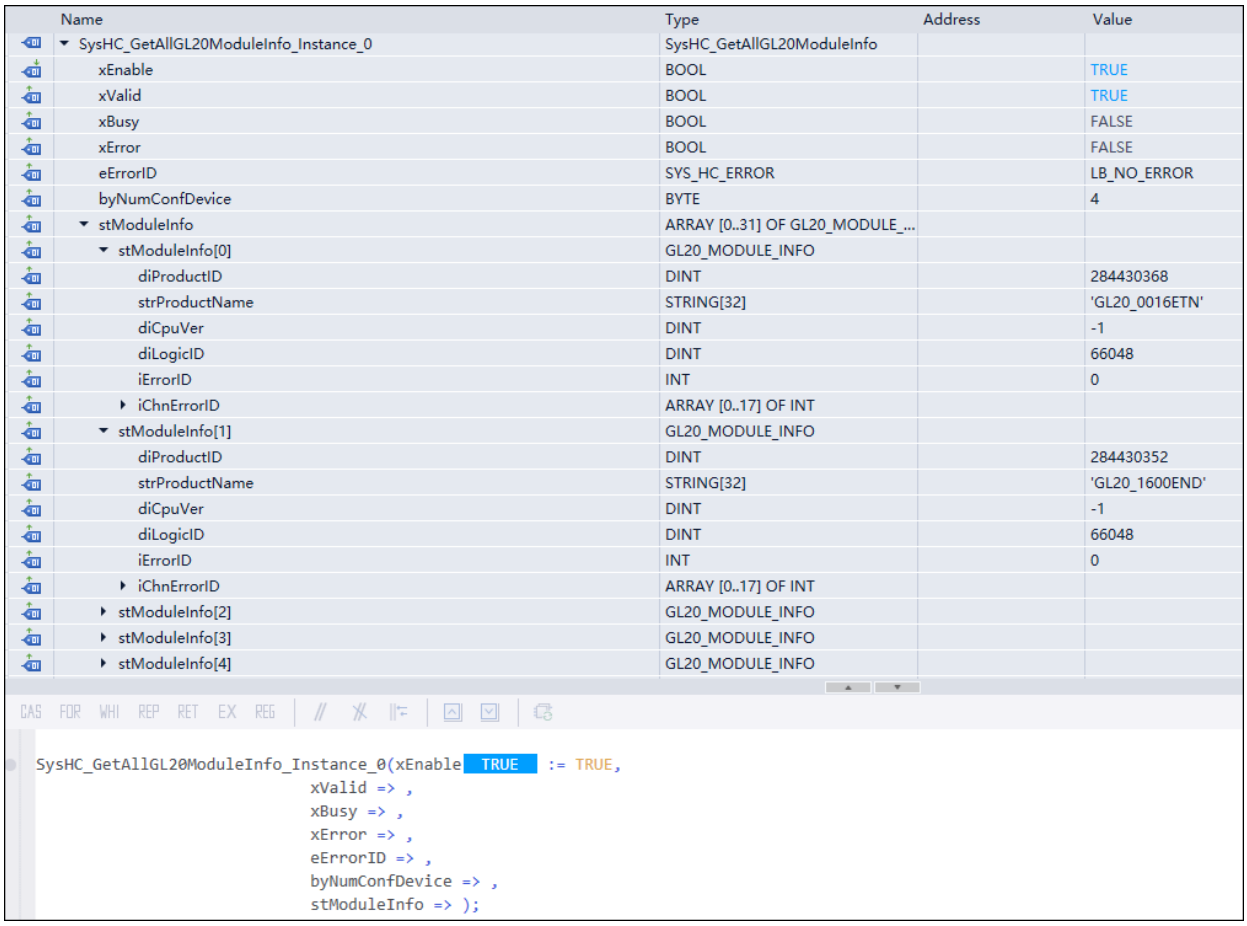
- **Function**
  - This function block is used to periodically read the status of all local modules.
  - In the output parameters, module status information is provided as a structure array, where the array index corresponds to the module slot number.
  - The data type of stModuleInfo is ARRAY [0..31] OF GL20_MODULE_INFO. The meaning of the structure GL20_MODULE_INFO is as follows.

Structure member	Name	Data Type	Description
diProductID	Module type ID	DINT	The module type on the local bus slot.
strProductName	Module name	STRING[32]	The module name on the local bus slot.
diCpuVer	CPU firmware version	DINT	The MCU software version of the module.

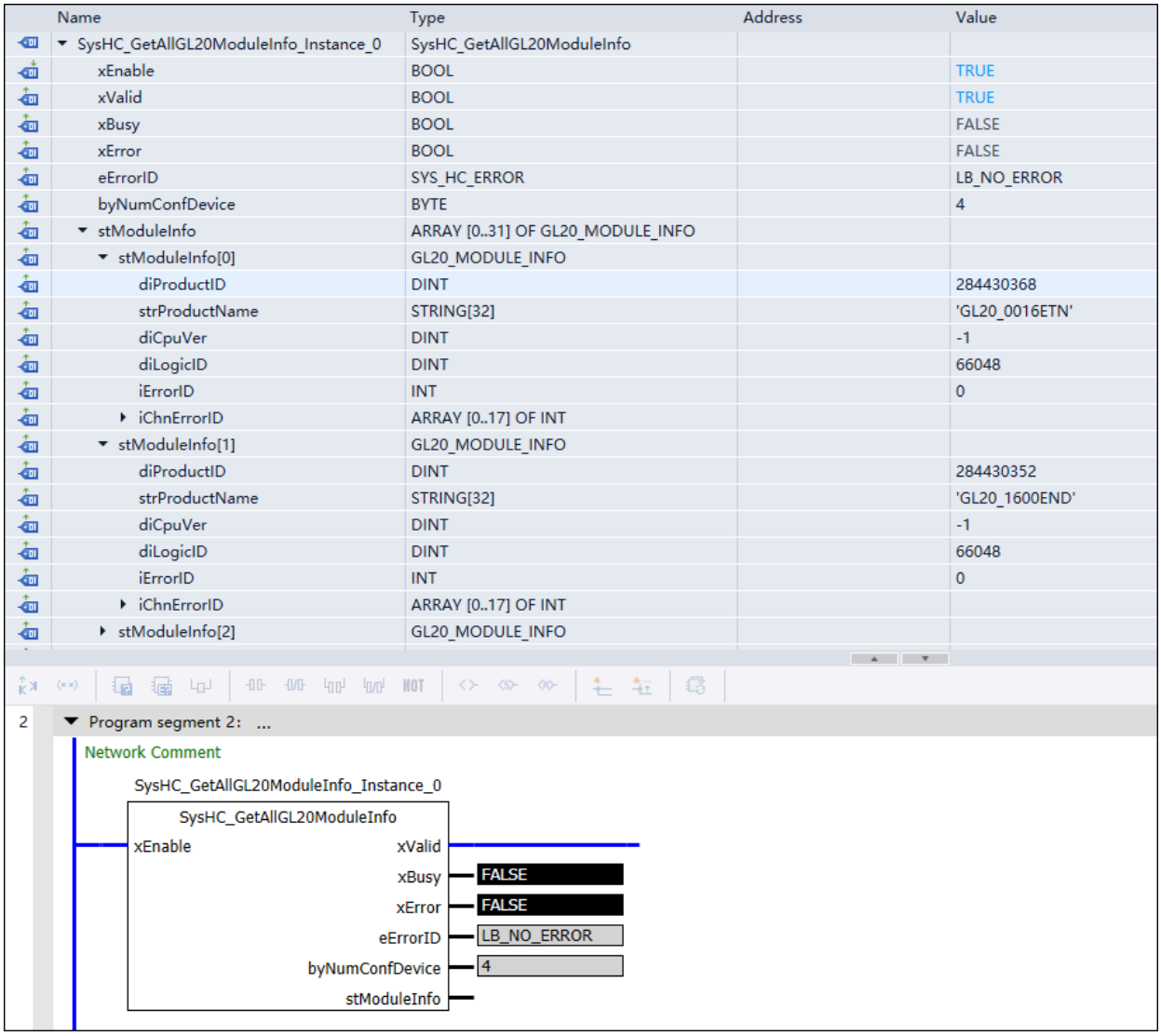
Structure member	Name	Data Type	Description
diLogicID	FPGA firmware version	DINT	The logic software version of the module.
iErrorID	Module fault code	INT	Module fault code. For details, see the corresponding module guide.
iChnErrorID	Channel fault code	ARRAY [0..17] OF INT	Fault code of module channel. For details, see the corresponding module guide.

- **Program example**

ST:



LD:

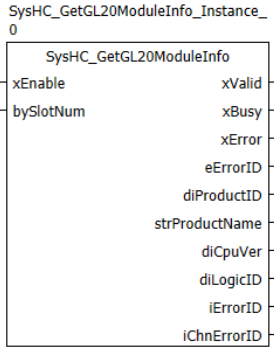


Name	Type	Address	Value
SysHC_GetAllGL20ModuleInfo_Instance_0	SysHC_GetAllGL20ModuleInfo		
xEnable	BOOL		TRUE
xValid	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
eErrorID	SYS_HC_ERROR		LB_NO_ERROR
byNumConfDevice	BYTE		4
stModuleInfo	ARRAY [0..31] OF GL20_MODULE_INFO		
stModuleInfo[0]	GL20_MODULE_INFO		
diProductID	DINT		284430368
strProductName	STRING[32]		'GL20_0016ETN'
diCpuVer	DINT		-1
diLogicID	DINT		66048
iErrorID	INT		0
iChnErrorID	ARRAY [0..17] OF INT		
stModuleInfo[1]	GL20_MODULE_INFO		
diProductID	DINT		284430352
strProductName	STRING[32]		'GL20_1600END'
diCpuVer	DINT		-1
diLogicID	DINT		66048
iErrorID	INT		0
iChnErrorID	ARRAY [0..17] OF INT		
stModuleInfo[2]	GL20_MODULE_INFO		

18.4.4.6 SysHC_GetGL20ModuleInfo

This instruction reads module information from a specified slot on the local bus.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SysHC_GetGL20ModuleInfo	Get basic information about a specified module on the local bus	FB		SysHC_GetGL20ModuleInfo_Instance_0(xEnable := , bySlotNum := , xValid => , xBusy => , xError => , eErrorID => , diProductID => , strProductName => , diCpuVer => , diLogicID => , iErrorID => , iChnErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Function block enable	BOOL	TRUE, FALSE	FALSE	Enable bit of the function block When the value of pin is TRUE, module information for the corresponding slot is read periodically.
	bySlotNum	Slot number	BYTE	[0, 31]	0	Set the slot number for which information is to be read.
Output	xValid	Valid output	BOOL	TRUE, FALSE	FALSE	Function block data valid flag. When the value of pin is TRUE, the read information is valid.
	xBusy	Instruction executing	BOOL	TRUE, FALSE	FALSE	Function block execution flag. When the value of pin is TRUE, the function block is reading information.
	xError	Instruction execution error	BOOL	TRUE, FALSE	FALSE	Function block fault flag.
	eErrorID	Instruction fault code	SYS_HC_ERROR	Follows data type	LB_NO_ERROR	Fault code of function block. For details, see Fault Codes.
	diProductID	Module type ID	DINT	Follows data type	-	The module type on the local bus slot.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	strProduct Name	Module name	STRING[32]	Follows data type	-	The module name on the local bus slot.
	diCpuVer	Module CPU firmware version	DINT	Follows data type	-	The MCU software version of the module.
	diLogicID	Module FPGA firmware version	DINT	Follows data type	-	The logic software version of the module.
	iErrorID	Module fault code	INT	Follows data type	-	Module fault code. For details, see the corresponding module guide.
	iChnErrorID	Channel fault code	ARRAY [0..17] OF INT	Follows data type	-	Fault code of module channel. For details, see the corresponding module guide.

- **Function description**

  This instruction reads the basic information of a specified module on the local bus. After setting the slot number, enable the function block to read module information.

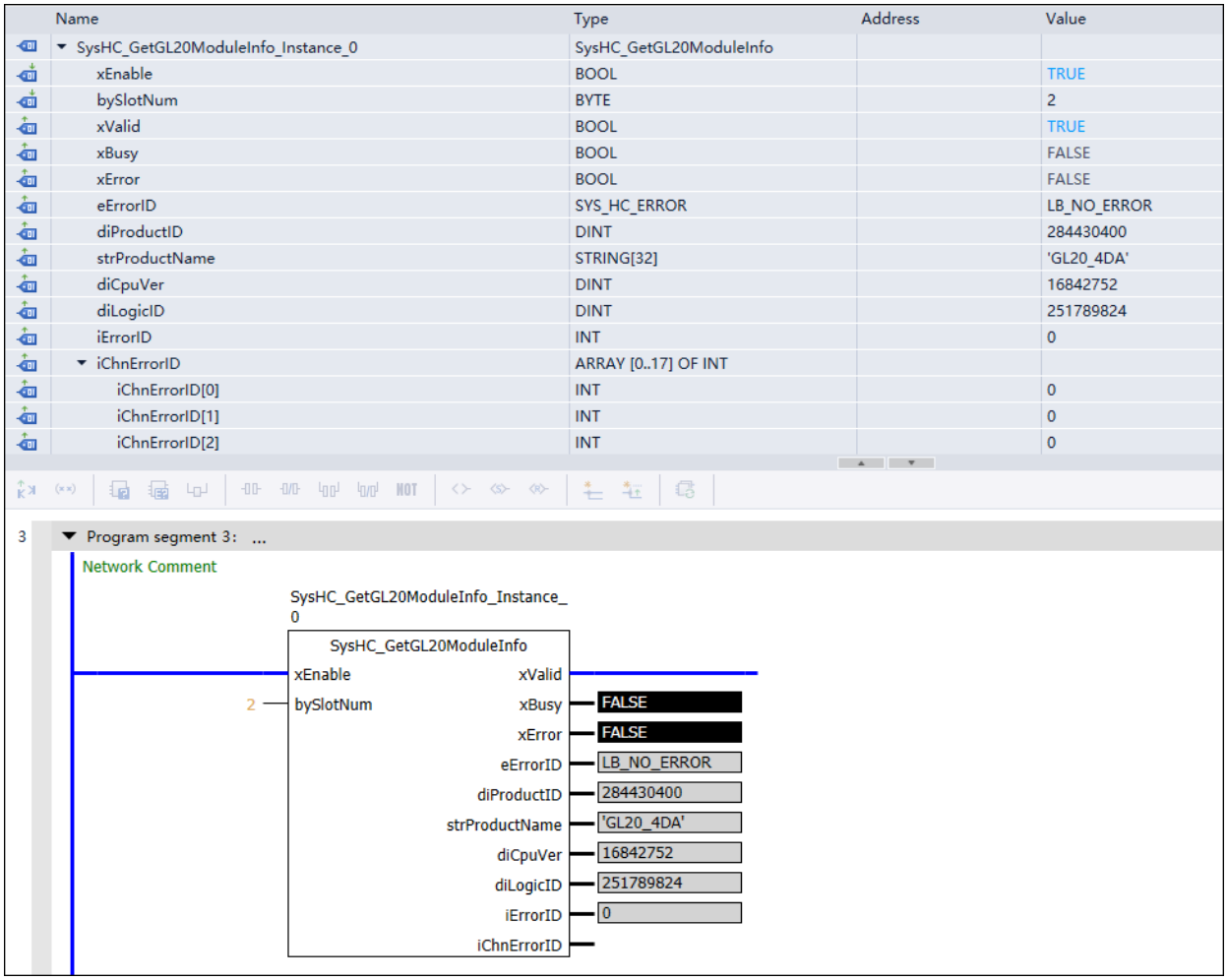
- **Program example**

  ST:

Name	Type	Address	Value
SysHC_GetGL20ModuleInfo_Instance_0	SysHC_GetGL20ModuleInfo		
xEnable	BOOL		TRUE
bySlotNum	BYTE		2
xValid	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
eErrorID	SYS_HC_ERROR		LB_NO_ERROR
diProductID	DINT		284430400
strProductName	STRING[32]		'GL20_4DA'
diCpuVer	DINT		16842752
diLogicID	DINT		251789824
iErrorID	INT		0
iChnErrorID	ARRAY [0..17] OF INT		
iChnErrorID[0]	INT		0
iChnErrorID[1]	INT		0
iChnErrorID[2]	INT		0
iChnErrorID[3]	INT		0
iChnErrorID[4]	INT		0
iChnErrorID[5]	INT		0
iChnErrorID[6]	INT		0

```
SysHC_GetGL20ModuleInfo_Instance_0(xEnable  TRUE  := TRUE,
                                   bySlotNum  2   := 2,
                                   xValid => ,
                                   xBusy => ,
                                   xError => ,
                                   eErrorID => ,
                                   diProductID => ,
                                   strProductName => ,
                                   diCpuVer => ,
                                   diLogicID => ,
                                   iErrorID => ,
                                   iChnErrorID => );
```

LD:



- **Note**

  The input parameter bySlotNum is used to set the slot number for which information is to be read. If the slot number is out of the valid range, or if the slot number is within range but no module is actually mounted on the slot, a slot number error will be reported.

18.4.4.7 Fault Codes

- SYS_HC_ERROR

Fault code	Value	Description
LB_NO_ERROR	0	No fault
LB_NOT_SUPPORTED	1	The bus encoder axis does not support this instruction.
LB_MODULE_DISABLE	2	Module is not enabled.
LB_PARAMETER_ERROR	3	Parameter fault.
LB_SLOT_NUM_ERROR	4	Slot number error.
LB_READ_TIME_OUT	5	Read operation timeout.
LB_WRITE_TIME_OUT	6	Write operation timeout.
LB_OBTAIN_FAILURE	7	Obtainment failed.

Fault code	Value	Description
LB_UNKNOWN_ERROR	255	Unknown error

- LBUS_ERROR

Fault code	Value	Description
NO_ERROR	0	No fault
NOT_READY	1	Not ready.
NOT_SUPPORTED	2	The instruction is not supported.
NOT_SUPPORTED_CURRENT_TASK	3	The current task is not supported.
INPUT_PARAMETER_INVALID	4	Invalid input parameter. The parameter must specify the corresponding slot.
MOD_TYPE_NOT_SUPPORTED	5	Module type not supported. Only array input and output are supported.
MOD_IO_TYPE_MISMATCH	6	Module input or output type does not match the instruction.
LB_STATE_MACHINE_NOT_SUPPORTED	7	The protocol state machine is not in the OP state and does not support this operation.
SLOT_NOT_EXIST	8	The module slot does not exist.
SLOT_DISABLE	9	The module slot is disabled.
SLOT_STATE_ERROR	10	Module fault state.
SLOT_DATA_READ_FAILURE	11	Module data read failed.
SLOT_DATA_WRITE_FAILURE	12	Module data write failed.
MOD_CONFIG_MORE	4097	The number of configured I/O modules is greater than the number of actual scanned I/O modules.
MOD_CONFIG_LESS	4098	The number of configured I/O modules is less than the number of actual scanned I/O modules.
MOD_CONFIG_MISMATCH	4099	The configured module type does not match the scanned module type.
MOD_CONFIG_FAILURE	4100	Failed to configure the I/O module on the protocol stack.
MOD_SET_STATE_FAILURE	4101	Failed to set the I/O module status on the protocol stack.
IOMODULE_LOSTLINK	4102	IO module is offline.
LBUS_ERR_SCAN_NO_SLOT	4103	The protocol stack failed to scan for modules.
MOD_CURRENT_OVER_LIMIT	4104	The power is overloaded because current exceeds the rated 2A of the bus.
STAT_TO_OP_TIMEOUT	4105	Timeout switching to OP state

Fault code	Value	Description
UNKNOWN	65535	Unknown error

18.4.5 ModbusRTU

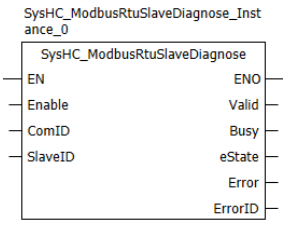
18.4.5.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.4.5.2 SysHC_ModbusRtuSlaveDiagnose	FB	Get the diagnostic status from the Modbus-RTU master	SysHCPlcInfo	Library version 1.0.0.0 and later
18.4.5.3 MB_Connect	FB	Establish a Modbus RTU connection	Modbus	Library version 1.0.0.0 and later
18.4.5.4 MB_Master	FB	Create Modbus RTU master communication		Library version 1.0.0.0 and later
18.4.5.5 MB_Slave	FB	Create Modbus RTU slave communication		Library version 1.0.0.0 and later

18.4.5.2 SysHC_ModbusRtuSlaveDiagnose

This instruction gets the diagnostic status information from the Modbus-RTU master.

- **Instruction format**

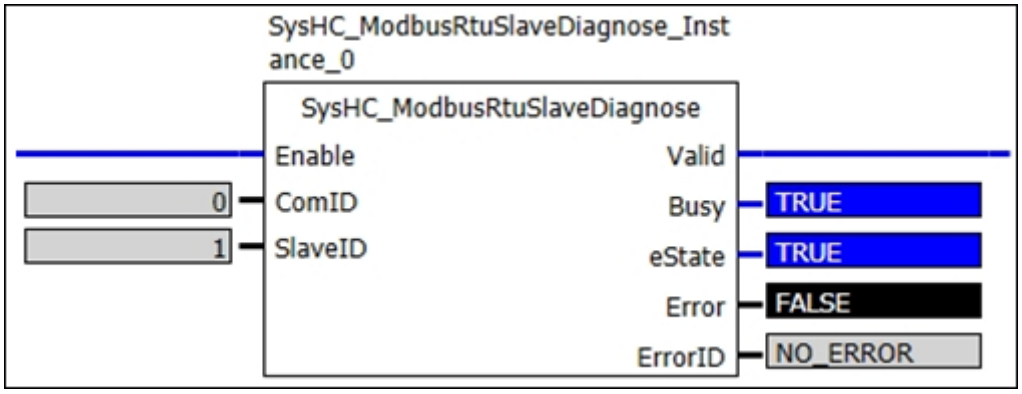
Instruction	Name	FB/FC	LD Representation	ST Representation
SysHC_ModbusRtuSlaveDiagnose	Get the diagnostic status from the Modbus-RTU master	FB		SysHC_ModbusRtuSlaveDiagnose_Instance_0(Enable :=, ComID :=, SlaveID :=, Valid =>, Busy =>, eState =>, Error =>, ErrorID =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Enable	Rising edge-triggered	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	ComID	Serial port number	BYTE	Follows data type	0	Serial port number. The value of ComID identifies serial ports, whose available number varies by product. For details, see relevant product guides. Take the EVO500 series as example: 0: Onboard serial port of the PLC 1: Expansion card A on the PLC 2: Expansion card B on the PLC
	SlaveID	Slave Address	BYTE	-	0	Slave address.
Output	Valid	Output active	BOOL	TRUE, FALSE	FALSE	Instruction is actively executing.
	Busy	Instruction executing	BOOL	TRUE, FALSE	FALSE	Instruction is executing.
	eState	Diagnostic status bit	BOOL	TRUE, FALSE	FALSE	Diagnostic status bit.
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs.
	ErrorID	Fault code	SYS_HC_ERROR	-	NO_ERROR	See 18.4.1.37 Fault Codes.

- **Function**
  - When Enable is TRUE, the function block is activated; otherwise, all outputs are invalid.
  - Check if the input ComID is within the valid range. If out of range, ErrorID outputs COM_ID_ERROR.
  - Check if the input SlaveID matches the configured slave ID in the project. If not matched, ErrorID outputs SLAVE_ID_ERROR.
  - If both ComID and SlaveID are successfully matched, verify whether the slave node is disabled. If disabled, ErrorID outputs FAILURE_OBATION.
  - After all parameters are matched, retrieve the communication diagnostic state for the specified RTU Slave and update eState in real time.
- **Program example**

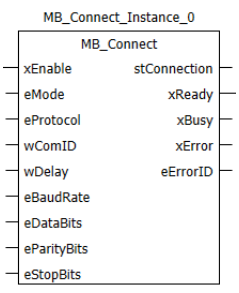


- **Precautions**
  - The value of ComID identifies serial ports, whose available number varies by product. For details, see relevant product guides. For example, on the EVO500 series, 0 represents the PLC onboard serial port, while 1 and 2 represent expansion cards A and B respectively.
  - Check the COM ports for locally or remotely mounted GL20_2S485/GL20_2SCOM modules as follows: In the project tree, double-click GL20_2S485 or GL20_2SCOM to open the configuration interface, and then check the COM number in the "General configuration" tab.

18.4.5.3 MB_Connect

This is the instruction for the configuration of Modbus protocol, which establishes a Modbus RTU connection.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
MB_Connect	Establish a Modbus RTU connection	FB		MB_Connect_Instance_0(xEnable := , eMode := , eProtocol := , wComID := , wDelay := , eBaudRate := , eDataBits := , eParityBits := , eStopBits := , stConnection => , xReady => , xBusy => , xError => , eErrorID =>);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Function block enable	BOOL	Follows data type	FALSE	Level triggered.
	eMode	Modbus mode	MB_MODE	Follows data type	MB_MODE.SLAVE	Modbus mode: Slave (default). MB_MODE.SLAVE: Modbus slave. MB_MODE.MASTER: Modbus master.
	eProtocol	Modbus protocol	MB_PROTOCOL	Follows data type	MB_PROTOCOL.RTU	Modbus protocol: RTU protocol (default). MB_PROTOCOL.RTU: ModbusRTU protocol. MB_PROTOCOL.ASCII: ModbusASCII protocol.
	wComID	Serial port number	WORD	Follow data type	0	COM port number. The value of wComID identifies serial ports, whose available number varies by product. For details, see relevant product guides. Take the EVO500 series as example: 0: Onboard serial port of the PLC 1: Expansion card A on the PLC 2: Expansion card B on the PLC
	wDelay	Frame interval	WORD	Follow data type	5	Modbus frame interval in ms.

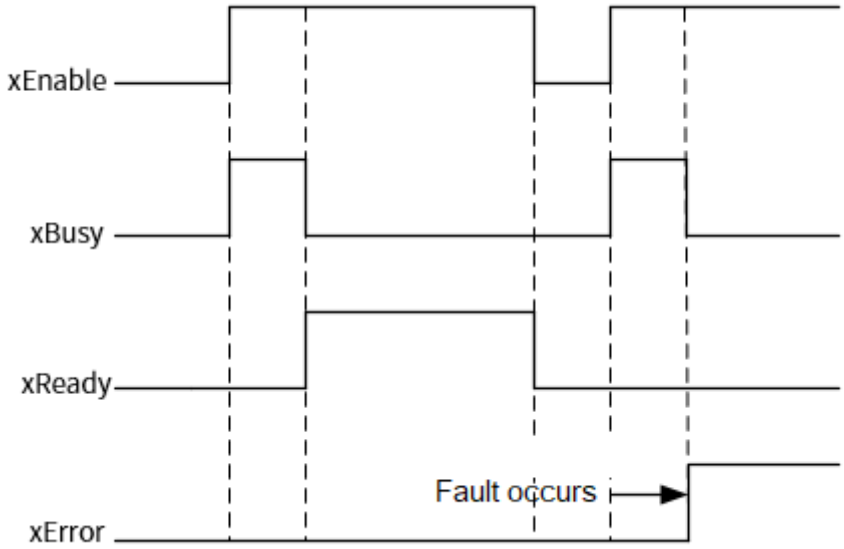
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	eBaudRate	Baud rate	SERIAL_BAUD	Follows data type	SERIAL_BAUD. B9600	Baud rate: 9600bps (default). SERIAL_BAUD.B4800: 4800bps. SERIAL_BAUD.B9600: 9600bps. SERIAL_BAUD.B19200: 19200bps. SERIAL_BAUD.B38400: 38400bps. SERIAL_BAUD.B57600: 57600bps. SERIAL_BAUD.B115200: 115200bps.
	eDataBits	Data bit	SERIAL_DATAB ITS	Follows data type	SERIAL_DATAB ITS.BIT8	Data bits: 8 bits (default). SERIAL_DATABITS.BIT7: 7 data bits SERIAL_DATABITS.BIT8: 8 data bits
	eParityBits	Parity bit	SERIAL_PARITY BITS	Follows data type	SERIAL_PARITY BITS.NONE	Parity Nno parity (default). SERIAL_PARITYBITS.NONE: No parity SERIAL_PARITYBITS.ODD: Odd parity SERIAL_PARITYBITS.EVEN: Even parity
	eStopBits	Stop bit	SERIAL_STOPB ITS	Follows data type	SERIAL_STOPB ITS.BIT1	Stop bit:1 (default). SERIAL_STOPBITS.BIT1: 1 data bit SERIAL_STOPBITS.BIT2: 2 data bits
Output	stConnection	Connection handle	CBus_Connect ion	Follows data type	-	Connection handle.
	xReady	Instruction status	BOOL	Follows data type	FALSE	Indicates the function block is active
	xBusy	Instruction execution flag	BOOL	Follows data type	FALSE	Function block execution flag

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	xError	Error flag	BOOL	Follows data type	FALSE	Indicates the function block is faulty
	eErrorID	Fault code	CBUS_ERROR	Follows data type	NO_ERROR	Fault code of function block. For details, see 18.4.5.6 Fault Codes.

- **Function**

  This instruction is used to configure serial port and master station parameters, such as the data transmission rate.

- **Sequence diagram**



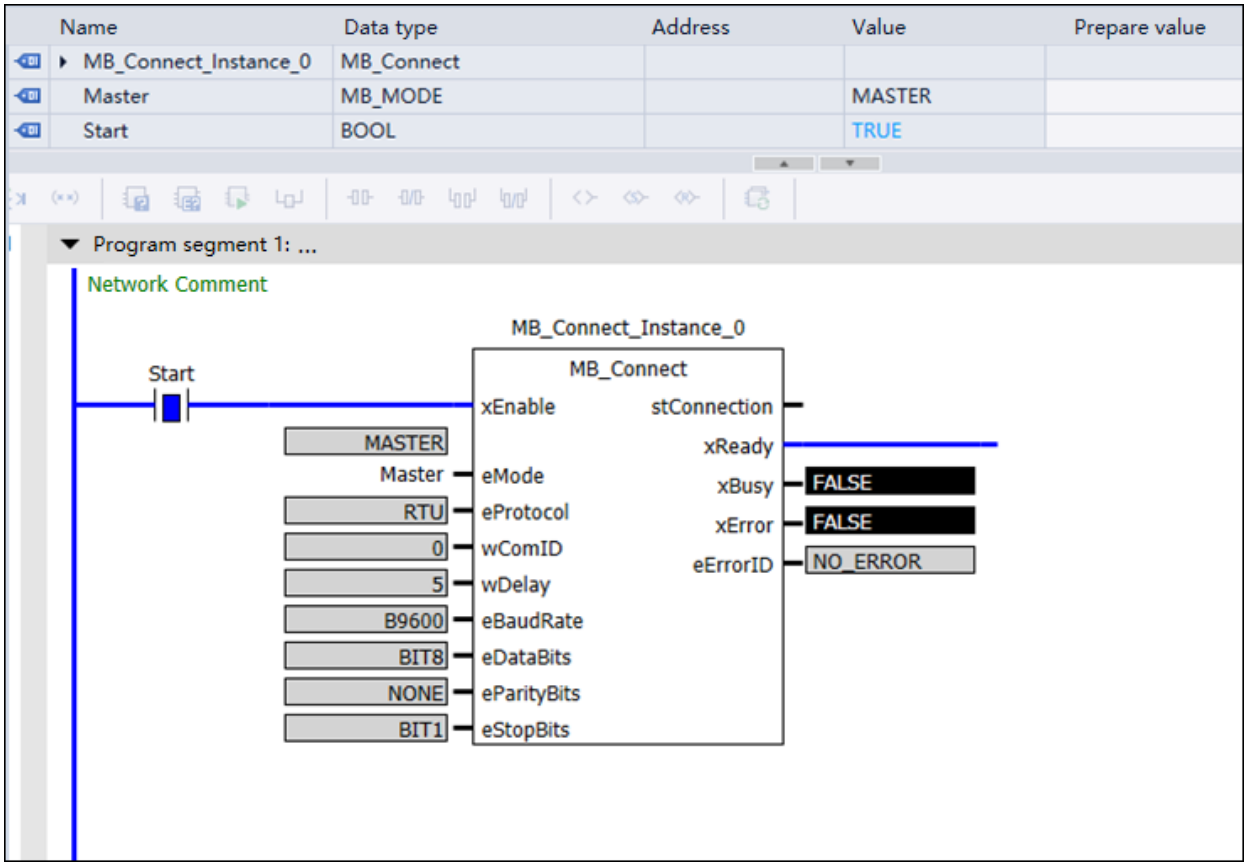
- **Program example**

  ST:

Name	Data type	Address	Value	Prepare value
MB_Connect_Instance_0	MB_Connect			
start	BOOL		TRUE	
mode	MB_MODE		SLAVE	
protocol	MB_PROTOCOL		RTU	
comid	WORD		0	
delay	WORD		0	
baudrate	SERIAL_BAUD		B4800	
databits	SERIAL_DATABITS		BIT7	
partitybits	SERIAL_PARITYBITS		NONE	
stopbits	SERIAL_STOPBITS		BIT1	
ready	BOOL		TRUE	
busy	BOOL		FALSE	
error	BOOL		FALSE	
errorid	CBUS_ERROR		NO_ERROR	
connection	CBus_Connection			

```
CAS FOR WHI REP RET EX REG
MB_Connect_Instance_0(
xEnable TRUE :=start TRUE,
eMode SLAVE :=mode SLAVE ,
eProtocol RTU := protocol RTU ,
wComID 0 := comid 0 ,
wDelay 0 := delay 0 ,
eBaudRate B4800 := baudrate B4800 ,
eDataBits BIT7 := databits BIT7 ,
eParityBits NONE := partitybits NONE ,
eStopBits BIT1 := stopbits BIT1 ,
stConnection => connection,
xReady TRUE => ready TRUE,
xBusy FALSE => busy FALSE ,
xError FALSE =>error FALSE ,
eErrorID NO_ERROR =>errorid NO_ERROR );RETURN
```

LD:

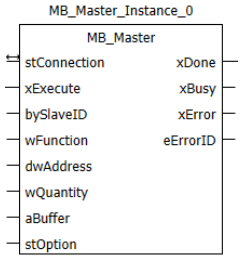


- **Precautions**
  - The connection handle of MB_Connect can only be used by either the MB_Master or MB_Slave instruction, not both.
  - Ensure that multiple instructions with the same connection handle are used in the same user task.
  - Online parameter modifications will not take effect immediately, requiring the instruction to be re-enabled.
  - The project tree and instruction configuration are mutually exclusive:
    - If a Modbus master, slave, or free serial port is configured for any serial port in the project tree, no instruction can establish a connection on the same port.
    - If a slave is configured for any serial port in the project tree, instructions cannot create a slave on any port, and only one slave is allowed to be configured via instructions.
  - The value of wComID identifies serial ports, whose available number varies by product. For details, see relevant product guides. For example, on the EVO500 series, 0 represents the PLC onboard serial port, while 1 and 2 represent expansion cards A and B respectively.
  - Check the COM ports for locally or remotely mounted GL20_2S485/GL20_2SCOM modules as follows: In the project tree, double-click GL20_2S485 or GL20_2SCOM to open the configuration interface, and then check the COM number in the "General configuration" tab.

18.4.5.4 MB_Master

This is the Modbus master communication instruction, which creates Modbus RTU master communication.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
MB_Master	Create Modbus RTU master communication	FB		MB_Master_Instance_0(stConnection := , xExecute := , bySlaveID := , wFunction := , dwAddress := , wQuantity := , aBuffer := , stOption := , xDone => , xBusy => , xError => , eErrorID =>);

- **Related variables**

  Variable

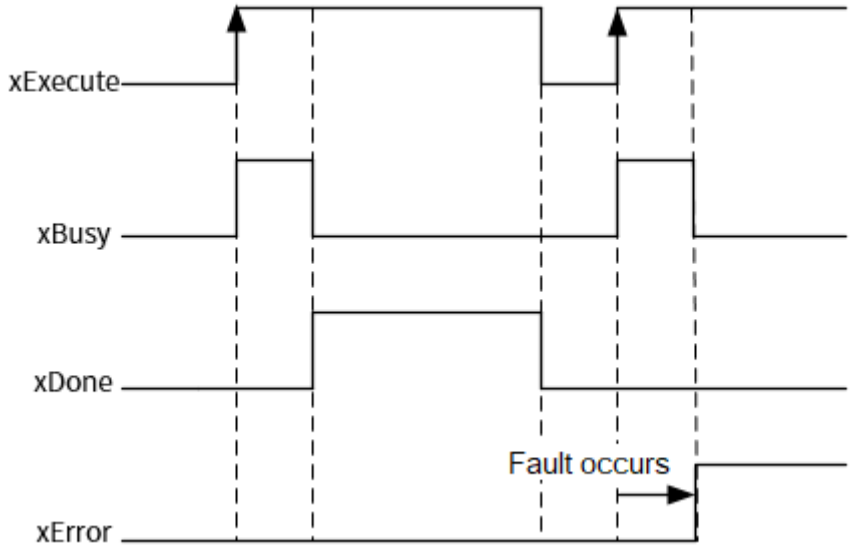
Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input/Output	stConnection	Connection handle	CBus_Connection	Follows data type	-	Connection handle.
Input	xExecute	Function block enable	BOOL	Follows data type	FALSE	Edge triggered.
	bySlaveID	Slave number	BYTE	Follows data type	1	Request access to the slave number.
	wFunction	Function code	MB_FUNCTION	Follows data type	0	Modbus function code.
	dwAddress	Slave data address	DWORD	Depends on data type	0	Starting address for access to slave data.
	wQuantity	Number of accessible address	WORD	Follow data type	0	Maximum number of function codes: - 01/02: 2000. - 03/04: 125. - 15: 1968. - 16: 123.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	aBuffer	Read/write buffer	ANY	Follows data type	-	Modbus data buffer.
	stOption	Configuration options	MB_OPTION	Follows data type	-	Optional configuration parameters for Modbus protocol.
Output	xDone	Instruction completion flag	BOOL	Follows data type	FALSE	Function block execution completion flag
	xBusy	Instruction execution flag	BOOL	Follows data type	FALSE	Function block execution flag.
	xError	Error flag	BOOL	Follows data type	FALSE	Indicates the function block is faulty
	eErrorID	Fault code	CBUS_ERROR	Follows data type	NO_ERROR	Fault code of function block. For details, see 1 8.4.5.6 Fault Codes.

- **Function**

  This instruction initiates data transmission, sending the user-configured Modbus data frame over the serial bus.

- **Sequence diagram**



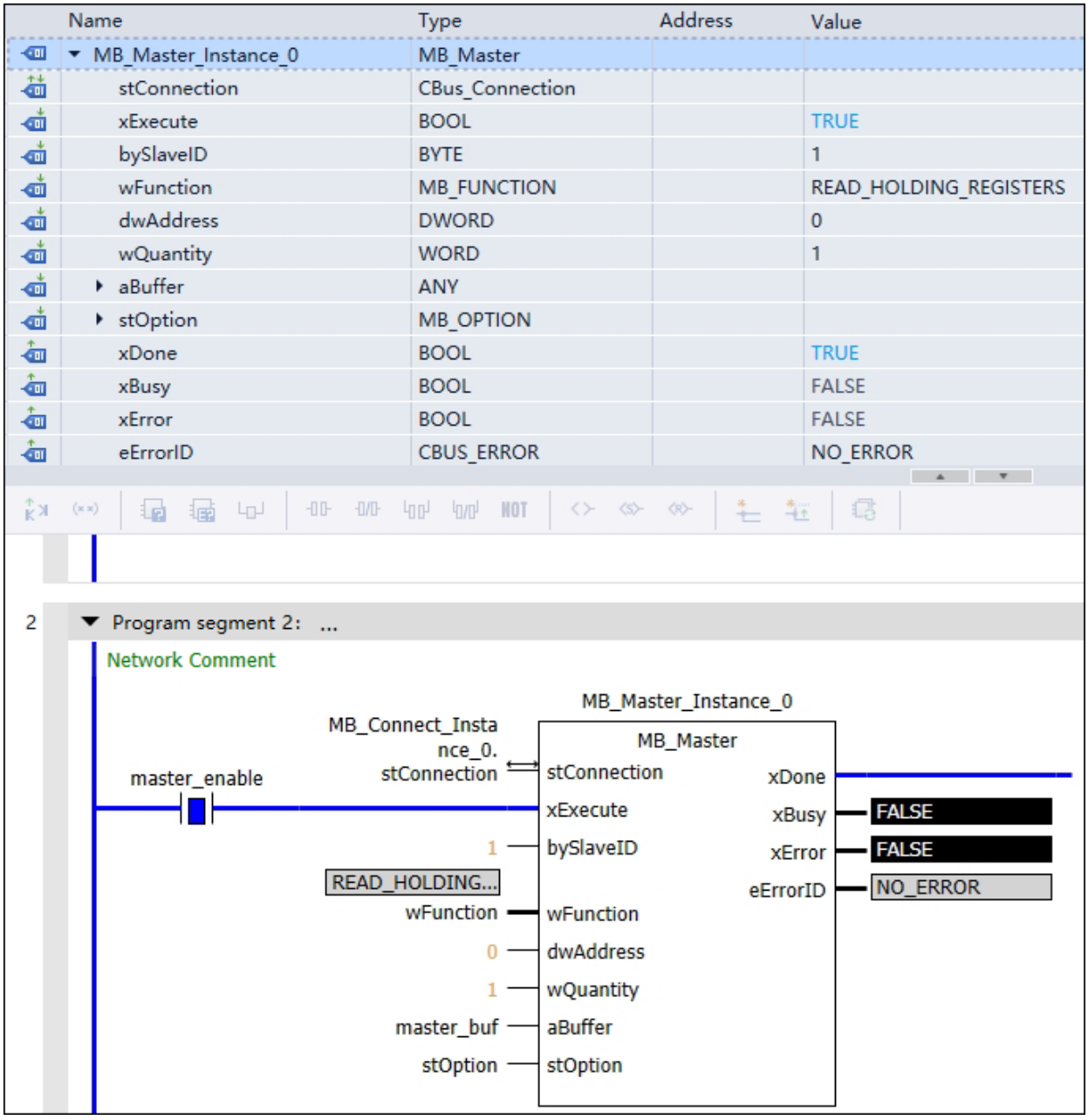
- **Program example**

  ST:

Name	Type	Address	Value	Prepared v
MB_Master_Instance_0	MB_Master			
stConnection	CBus_Connection			
xExecute	BOOL		TRUE	
bySlaveID	BYTE		1	
wFunction	MB_FUNCTION		READ_HOLDING_REGISTERS	
dwAddress	DWORD		0	
wQuantity	WORD		1	
aBuffer	ANY			
stOption	MB_OPTION			
xDone	BOOL		TRUE	
xBusy	BOOL		FALSE	
xError	BOOL		FALSE	
eErrorID	CBUS_ERROR		NO_ERROR	

```
MB_Master_Instance_0(stConnection := MB_Connect_Instance_0.stConnection,
                     xExecute TRUE := xExecute TRUE,
                     bySlaveID 1 := bySlaveID 1,
                     wFunctionREAD_HOLDING_REGIST... := wFunctionREAD_HOLDING_REGIST...,
                     dwAddress 0 := dwAddress 0,
                     wQuantity 1 := wQuantity 1,
                     aBuffer := master_buf,
                     stOption := stOption,
                     xDone TRUE => m_done TRUE,
                     xBusy FALSE => m_busy FALSE,
                     xError FALSE => m_error FALSE,
                     eErrorID NO_ERROR => m_errorid NO_ERROR );
```

  LD:



- **Precautions**
  - It must be used in conjunction with the MB_Connect instruction, and the eMode must be set to MASTER.
  - The connection handle configured by the MB_Connect instruction supports use by one or multiple MB_Master instances.
  - It is recommended to process the MB_Connect and MB_Master instructions within the same user task and program cycle. If processed in different priority tasks, operational errors may occur due to priority interruptions.
  - When the MB_Master instruction is in the xBusy set state, repeated instructions will be ignored until xDone or xError is set.
  - The MB_Master instruction cannot be executed during startup, diagnostic, safe mode, or in the event of a time error.
  - The bySlaveID slave address only supports standard station addresses: 1 to 247, with 0 reserved for broadcast.
  - Function code and number of accessible addresses
    - Function code: READ_COILS/ READ_DISCRETE_INPUTS, the maximum number of accessible addresses supported is 2000.

- Function code: WRITE_MULTIPLE_COILS, the maximum number of accessible addresses supported is 1968.
- Function code: READ_HOLDING_REGISTERS/ READ_INPUT_REGISTERS, the maximum number of accessible addresses supported is 125.
- Function code: WRITE_MULTIPLE_REGISTERS, the maximum number of accessible addresses supported is 123.

◦ Data types loaded by aBuffer:

- Supported types:

  One-dimensional array.

  Data types of the arrays/variables:

  Access coil data: BYTE.

  Access register data: WORD, DWORD, INT, UINT, DINT, UDINT, REAL.

- Unsupported types

  Multidimensional arrays.

  64-bit data types: LWORD, LINT, ULINT, and LREAL.

  Special types: STRING, WSTRING, TIME, TIME_OF_DAY(TOD), DATE, DATE_AND_TIME(DT), POINTER, and STRUCT.

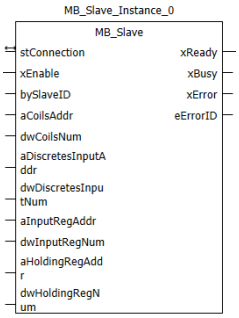
◦ aBuffer cannot be directly set to I/Q/M direct variables. It must be mapped to the above supported variables or array data via AT mapping.

◦ Online parameter modifications will not take effect immediately, requiring the instruction to be re-enabled.

18.4.5.5 MB_Slave

This is the Modbus slave communication instruction, which creates Modbus RTU slave communication.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
MB_Slave	Create Modbus RTU slave communication	FB		MB_Slave_Instance_0(stConnection := , xEnable := , bySlaveID := , aCoilsAddr := , dwCoilsNum := , aDiscretesInputAddr := , dwDiscretesInputNum := , aInputRegAddr := , dwInputRegNum := , aHoldingRegAddr := , dwHoldingRegNum := , xReady => , xBusy => , xError => , eErrorID =>);

- **Related variables**

Variable

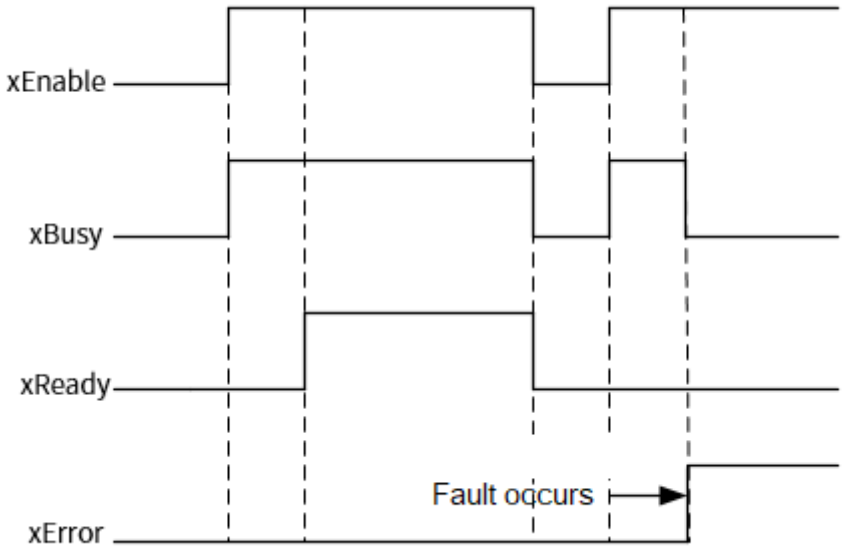
Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input/Output	stConnection	Connection handle	CBus_Connection	Follows data type	-	Connection handle.
Input	xEnable	Function block enable	BOOL	Follows data type	FALSE	Level triggered
	bySlaveID	Slave number	BYTE	Follows data type	1	Modbus slave number.
	aCoilsAddr	Coil data area	ANY	Follows data type	-	Supported function codes: 0x01/0x05/0x0F.
	dwCoilsNum	Number of coils	DWORD	Depends on data type	0	Number of accessible coils.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	aDiscretesInputAddr	Discrete input data area	ANY	Follows data type	-	Supported function code: 0x02.
	dwDiscretesInputNum	Number of discrete inputs	DWORD	Depends on data type	0	Number of accessible discrete inputs.
	aInputRegAddr	Input register data area	ANY	Follows data type	-	Supported function code: 0x04.
	dwInputRegNum	Number of input registers	DWORD	Depends on data type	0	Number of accessible input registers.
	aHoldingRegAddr	Holding register data area	ANY	Follows data type	-	Supported function codes: 0x03/0x06/0x10.
	dwHoldingRegNum	Number of holding registers	DWORD	Depends on data type	0	Number of accessible holding registers.
Output	xReady	Instruction status	BOOL	Follows data type	FALSE	Indicates the function block is active
	xBusy	Instruction execution flag	BOOL	Follows data type	FALSE	Function block execution flag
	xError	Error flag	BOOL	Follows data type	FALSE	Indicates the function block is faulty
	eErrorID	Fault code	CBUS_ERROR	Follows data type	NO_ERROR	Fault code of function block. For details, see 18.4.5.6 Fault Codes.

- **Function**

This instruction allows any master to perform Modbus requests and return response frames.

- **Sequence diagram**



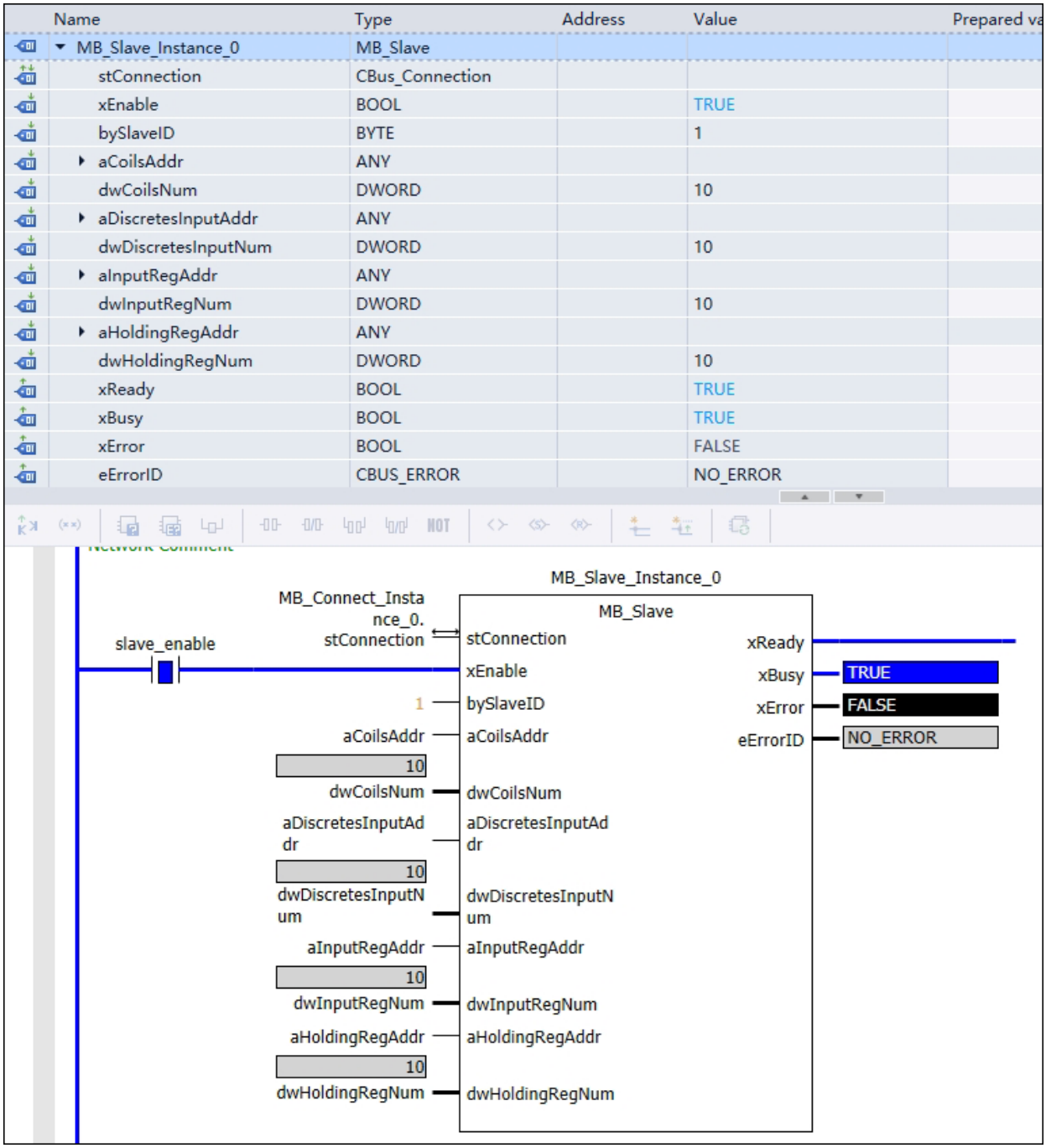
- **Program example**

ST:

Name	Type	Address	Value
MB_Slave_Instance_0	MB_Slave		
stConnection	CBus_Connection		
xEnable	BOOL		TRUE
bySlaveID	BYTE		1
aCoilsAddr	ANY		
dwCoilsNum	DWORD		10
aDiscretesInputAddr	ANY		
dwDiscretesInputNum	DWORD		10
aInputRegAddr	ANY		
dwInputRegNum	DWORD		10
aHoldingRegAddr	ANY		
dwHoldingRegNum	DWORD		10
xReady	BOOL		TRUE
xBusy	BOOL		TRUE
xError	BOOL		FALSE
eErrorID	CBUS_ERROR		NO_ERROR

```
MB_Slave_Instance_0(stConnection := MB_Connect_Instance_0.stConnection,
                    xEnable TRUE  := xEnable TRUE ,
                    bySlaveID  1  := bySlaveID  1 ,
                    aCoilsAddr := aCoilsAddr,
                    dwCoilsNum  10  := dwCoilsNum  10 ,
                    aDiscretesInputAddr := aDiscretesInputAddr,
                    dwDiscretesInputNum  10  := dwDiscretesInputNum  10 ,
                    aInputRegAddr := aInputRegAddr,
                    dwInputRegNum  10  := dwInputRegNum  10 ,
                    aHoldingRegAddr := aHoldingRegAddr,
                    dwHoldingRegNum  10  := dwHoldingRegNum  10 ,
                    xReady TRUE  => s_ready TRUE ,
                    xBusy TRUE  => s_busy TRUE ,
                    xError FALSE  => s_error FALSE ,
                    eErrorID NO_ERROR  => s_errorid NO_ERROR );
```

LD:



- **Precautions**
  - It must be used in conjunction with the MB_Connect instruction, and the eMode must be set to SLAVE.
  - The connection handle of the MB_Connect instruction can be used by only one instance of MB_Slave.
  - It is recommended that MB_Connect and MB_Slave instructions be processed within the same user task and the same program cycle. If they are executed in tasks with different priorities, operational errors may occur due to priority interrupts.
  - When the MB_Slave instruction is active and accessible, xReady/xBusy will be set. If an error occurs, xError is set and eErrorID displays the corresponding error code.
  - The MB_Slave can still be accessed even when the PLC mode switch is set to STOP.
  - Access area description:

Data Area Variable	Area Name	Supported Function Code	Accessible Area Size	Supported Data Type
aCoilsAddr	Coil data area	READ_COILS(01) WRITE_SINGLE_COIL(05) WRITE_MULTIPLE_COILS(15)	1 to 65536	BYTE, ARRAY
aDiscretesInputAddr	Discrete input data area	READ_DISCRETE_INPUTS(02)	1 to 65536	
aInputRegAddr	Input register data area	READ_INPUT_REGISTERS(04)	1 to 65536	WORD, DWORD, INT, UINT, DINT, UDINT, REAL, ARRAY
aHoldingRegAddr	Holding register data area	READ_HOLDING_REGISTERS(03) WRITE_SINGLE_REGISTER(06) WRITE_MULTIPLE_REGISTERS(16)	1 to 65536	

- Unsupported types for access area:

  Multidimensional arrays.
  64-bit data types: LWORD, LINT, ULINT, and LREAL.
  Special types: STRING, WSTRING, TIME, TIME_OF_DAY(TOD), DATE, DATE_AND_TIME(DT), POINTER, and STRUCT.

- Access areas cannot be directly set to I/Q/M direct variables. They must be mapped to the above supported variables or array data via AT mapping.
- Access areas can be freely combined. Unused access areas can be left empty, but all four access areas cannot be empty at the same time.
- Online parameter modifications will not take effect immediately, requiring the instruction to be re-enabled.

18.4.5.6 Fault Codes

Fault code	Value	Description
NO_ERROR	0	No error.
CONNECT_FAILED	200	Failed to create connection
CONNECT_MODE_INVALID	201	Connection mode invalid
CONNECTION_INVALID	202	Connection handle invalid
CONNECTION_OCCUPY	203	Connection handle occupied
CONNECTION_MODIFY	204	Connection handle modification error
MEMORY_ADDR_INVALID	205	Memory address invalid
MEMORY_OUT_OF_BOUNDS	206	Memory access out of bounds
DATA_DIM_INVALID	207	Data dimension invalid (only one-dimensional supported)

Fault code	Value	Description
DATA_ADDR_INVALID	208	Invalid data address.
DATA_TYPE_INVALID	209	Data type invalid
DATA_VALUE_INVALID	210	Data value invalid
SEND_FAILED	211	Data sending failed
RECV_FAILED	212	Data reception failed
RECV_TIMEOUT	213	Data reception timeout
SERIAL_COM_INVALID	300	COM port invalid
SERIAL_COM_UNPREADY	301	COM port not ready
SERIAL_COM_USED	302	COM occupied
SERIAL_BAUDRATE_ERR	303	Incorrect COM baud rate configuration
SERIAL_DATABITS_ERR	304	Incorrect configuration of COM data bit
SERIAL_PARITY_ERR	305	Incorrect configuration of COM parity bit
SERIAL_STOPBITS_ERR	306	Incorrect configuration of COM stop bit
MB_CONFIG_ERR	400	Configuration of Modbus failed
MB_FUNC_CODE_INVALID	401	Invalid Modbus function code
MB_QUANTITY_INVALID	402	Invalid Modbus address quantity
MB_RETRY_TIMES_INVALID	403	Invalid Modbus retransmission count (0 to15) configuration
MB_SLAVE_ID_INVALID	404	Invalid Modbus slave response station ID
MB_SLAVE_ADDR_ILLEGAL	405	Illegal Modbus slave response address
MB_SLAVE_DATA_ILLEGAL	406	Illegal Modbus slave response data
MB_MAPPING_MEM_ILLEGAL	407	Illegal Modbus slave mapping address
MB_MAPPING_NUM_INVALID	408	Invalid Modbus slave mapping size
MB_RTU_DATABITS7_ILLEGAL	409	Illegal ModbusRTU 7-bit data configuration
MB_ASC_DATABITS8_ILLEGAL	410	Illegal ModbusASCII 8-bit data configuration
MB_TCP_CONNECT_FAILED	500	ModbusTCP connection failed
MB_TCP_IP_ADDR_INVALID	501	Invalid ModbusTCP IP address
MB_TCP_ETHER_ID_INVALID	502	Invalid ModbusTCP Ethernet port configuration is invalid.
MB_TCP_DELAY_POLL_INVALID	503	Invalid ModbusTCP frame interval (1 to 1000).
MB_SLAVE_CONFIGURED	504	Modbus slave already configured
UNKNOWN	65535	Unknown error

18.4.5.7 Example of ModbusRTU Communication Instruction Usage

Two EVO523 controllers for Modbus RTU communication: PLC Master Com0 as Modbus RTU Master and PLC Slave Com0 as Modbus RTU Slave.

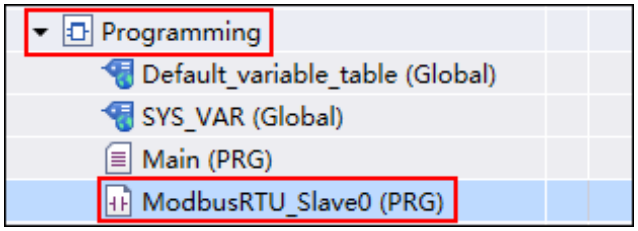
ModbusRTU slave configuration

Prerequisites

- A project has been created and the device added. For specific operations, see 7 Project Deployment.
- The Modbus library has been referenced. For the reference operations, see 12.3 Library Reference.

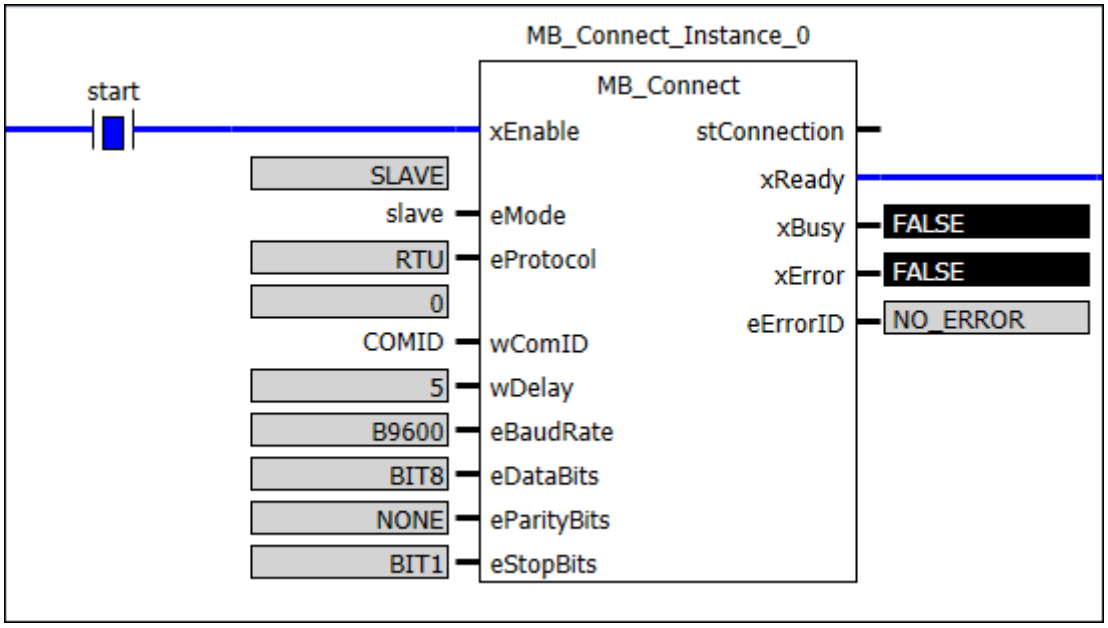
Procedure

1. Select the PLC_SLAVE Device and click on "Programming" in the left project tree to create a POU of type PRG (either ST or LD language is acceptable; ST language is used as an example).



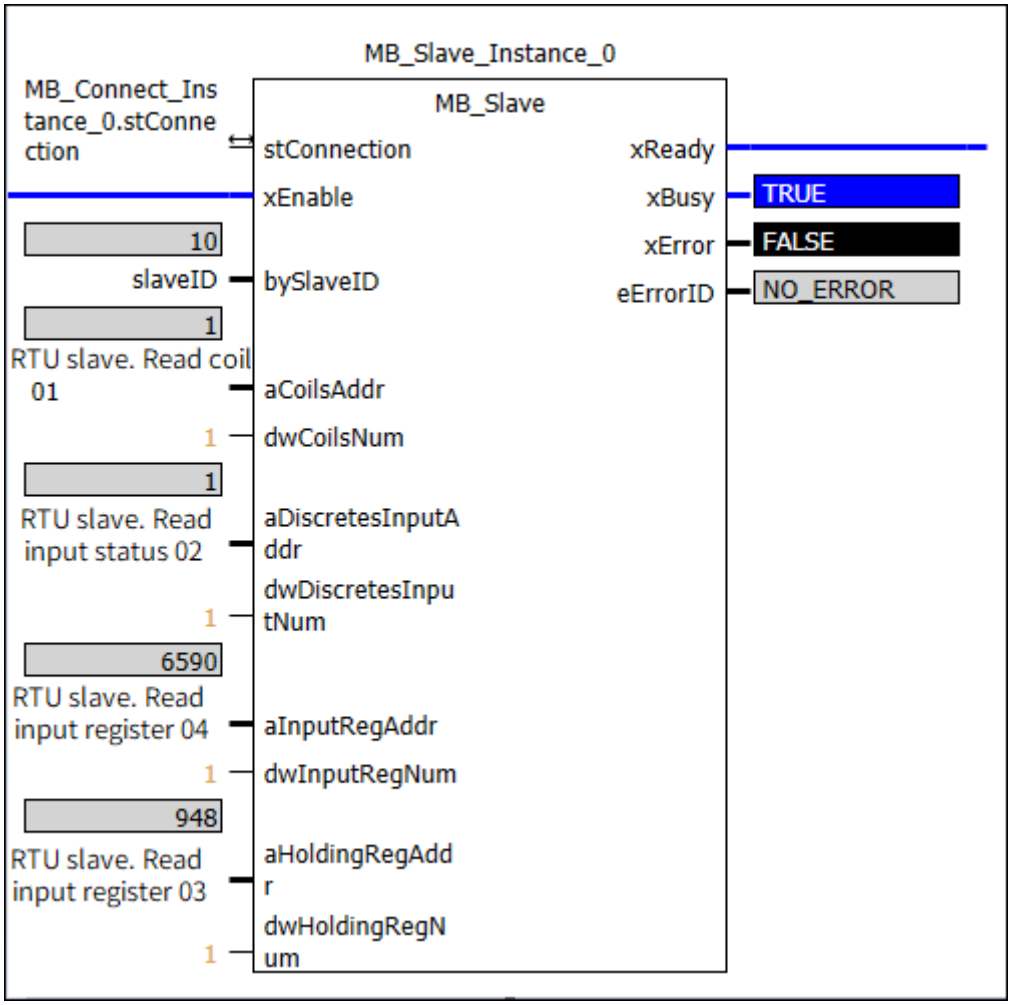
2. Add the "MB_TCP_Connect" instruction: Double-click to open the POU interface, select the "MB_Connect" instruction from the right Toolbox, and then double-click or drag it directly to the program editing area to create an instruction instance object.

3. Select SLAVE as the instruction mode, configure parameters (serial port number, frame interval, baud rate, etc.), and create a slave connection handle. Compile and download the program, enable the instruction after the device goes online, and observe the instruction pin status. The handle is successfully created when xReady is set to TRUE.



4. Switch the device to Offline mode, add the "MB_Slave" instruction by selecting it from the right-hand toolbox and double-clicking or dragging it to the program editing area. Configure parameters such as the slave ID, coil address, and number of coils. The stConnection pin needs to be passed the slave station connection handle created by the MB_Connect instruction.

5. Declare variables in the program to serve as data buffers for reading coils, discrete inputs, input registers, and holding registers (assigned to aCoilsAddr, aDiscretesInputAddr, aInputRegAddr, and aHoldingRegAddr, respectively). Then, set the number of coil register to 1. Compile and download the program, switch the device to Online mode, run the device, enable the instruction, and observe the instruction pin status. xReady and xBusy set to TRUE indicate successful slave creation.



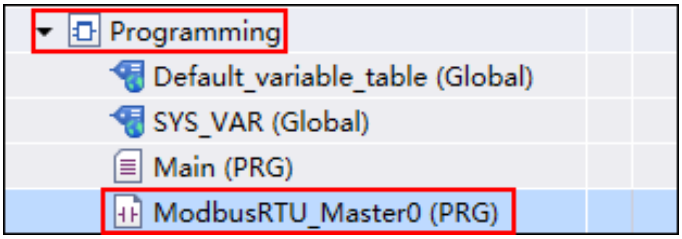
ModbusRTU master configuration

Prerequisites

- A project has been created and the device added. For specific operations, see 7 Project Deployment.
- The Modbus library has been referenced. For the reference operations, see 12.3 Library Reference.

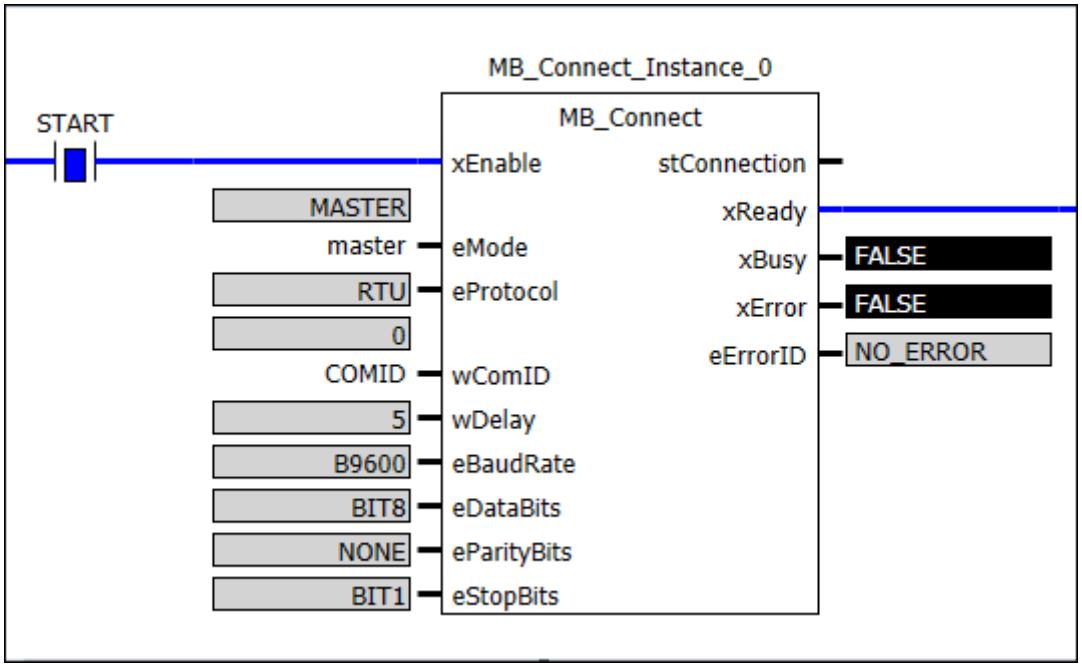
Procedure

1. Select the PLC_MASTER Device and click on "Programming" in the left project tree to create a POU of type PRG (either ST or LD language is acceptable; ST language is used as an example).



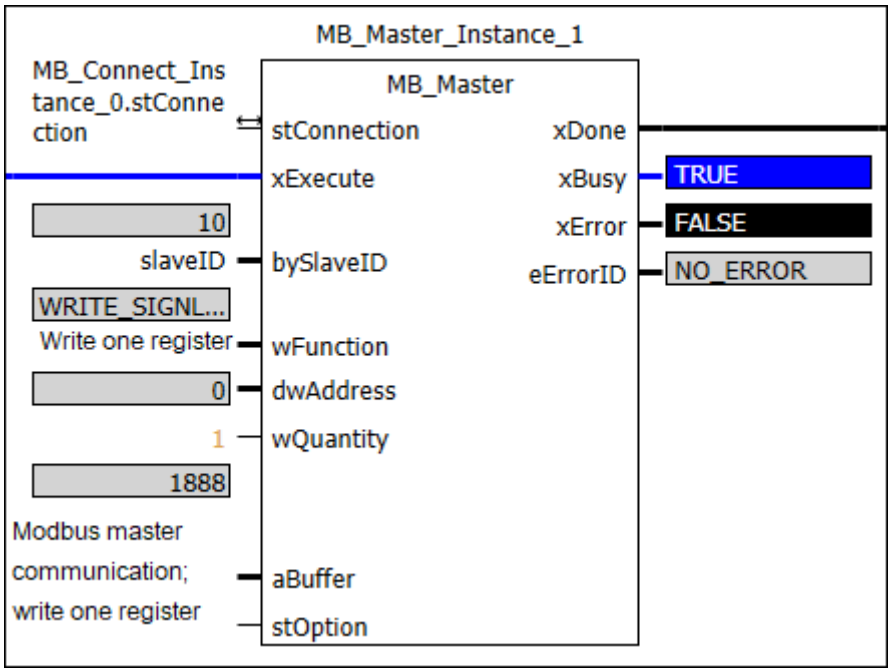
2. Add the "MB_TCP_Connect" instruction: Double-click to open the POU interface, select the "MB_Connect" instruction from the right Toolbox, and then double-click or drag it directly to the program editing area to create an instruction instance object.

3. Set the instruction mode to MASTER. Configure the master port parameters based on the slave port settings to create a master connection handle. Then, compile and download the program. After switching the device to online mode, enable the instruction and observe its pin status. The master handle is successfully created when xReady is set to TRUE.



4. Switch the device to Offline mode, add the "MB_Master" instruction by selecting it from the right-hand toolbox and double-clicking or dragging it into the program editor area. Pass the master connection handle to the stConnection pin, configure the slave ID, select the function code for writing a single holding register, add a variable for writing a single register with data type DWORD, connect this variable to the aBuffer pin, and set wQuantity to 1. Compile and download the program, then switch the device to Online mode and run the device.

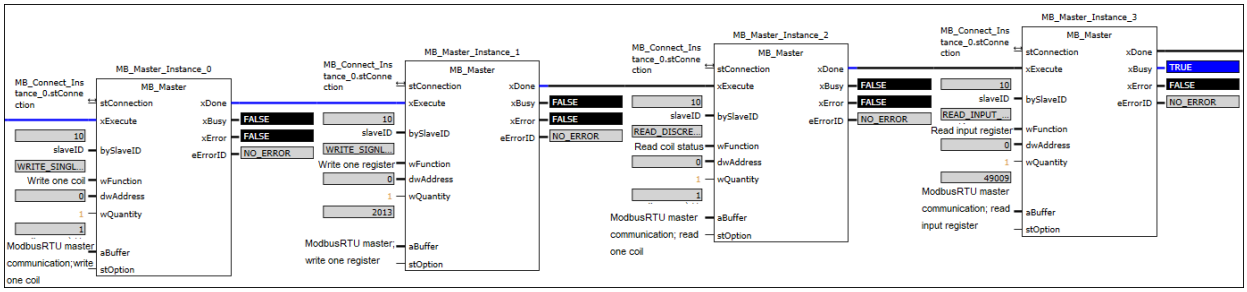
5. Modify the value of the variable connected to the aBuffer pin online. Apply a rising-edge signal to the instruction trigger. Observe the instruction pin status: xBusy changes from TRUE to FALSE and xDone is set to TRUE. Check the value of the read-register variable in the PLC_Slave device. If it matches the value written by the master-side register variable, the communication is successful.



Configuration of multiple ModbusRTU masters

The MB_Master instruction is rising edge-triggered. Each trigger executes only one communication transaction, and each instruction can be configured with only one function code. If you need continuous communication with multiple function codes, use the following configuration.

Drag multiple MB_Master instructions into the POU and connect them in series. The xReady pin of each instruction serves as the trigger signal for the next one, and the inverted xDone pin of the last instruction is fed back as the trigger signal for the first instruction. All instructions share the same master connection handle on their stConnection pins, and each is configured with the slave ID, function code, coil/register address, and quantity.



18.4.6 ModbusTCP

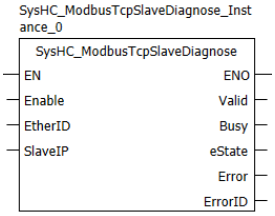
18.4.6.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.4.6.2 SysHC_ModbusTcpSlaveDiagnose	FB	Obatin Modbus-TCP master diagnostic status information	SysHCPlcInfo	Library version 1.0.0.0 and later
18.4.6.3 SysHC_ModbusTcpDeviceDiagnose	FB	Get Modbus-TCP as slave diagnostic status information		Library version 1.0.0.0 and later
18.4.6.4 MB_TCP_Connect	FB	Establish a Modbus TCP connection	Modbus	Library version 1.0.0.0 and later
18.4.6.5 MB_Server	FB	Create a Modbus TCP server		Library version 1.0.0.0 and later
18.4.6.6 MB_Client	FB	Create Modbus TCP client		Library version 1.0.0.0 and later

18.4.6.2 SysHC_ModbusTcpSlaveDiagnose

This instruction gets the diagnostic state information from the Modbus-TCP master.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SysHC_ModbusTcpSlaveDiagnose	Obatin Modbus-TCP master diagnostic status information	FB		SysHC_ModbusTcpSlaveDiagnose_Instance_0(Enable :=, EtherID :=, SlaveIP :=, Valid =>, Busy =>, eState =>, Error =>, ErrorID =>);

- **Related variables**

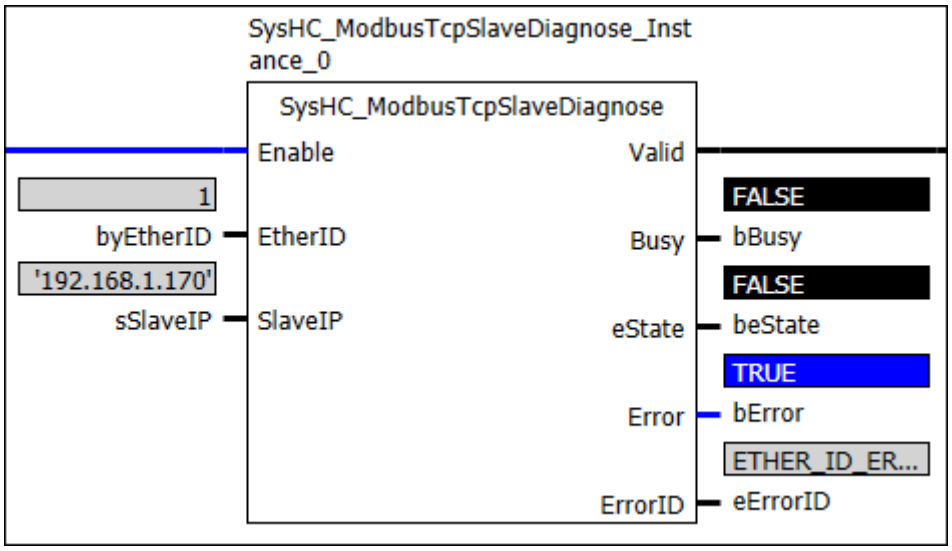
  Variable

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Enable	Rising edge-triggered	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	EtherID	Network port ID	BYTE	-	0	Network port ID number.
	SlaveIP	Slave IP address	STRING	-	-	Slave IP address.
Output	Valid	Output active	BOOL	TRUE, FALSE	FALSE	Instruction is actively executing.
	Busy	Instruction executing	BOOL	TRUE, FALSE	FALSE	Instruction is executing.
	eState	Diagnostic status bit	BOOL	TRUE, FALSE	FALSE	Diagnostic status bit.
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs.
	ErrorID	Fault code	SYS_HC_ERROR	-	NO_ERROR	See 18.4.1.37 Fault Codes.

- **Function**
  - Enable set to TRUE indicates that the function block is enabled; otherwise, all outputs are invalid.
  - Check if EtherID is supported on the device; if not supported, set ErrorID to ETHER_ID_ERROR.

- Input SlaveIP is a string type; if the SlaveIP parameter is valid and matches the configured Slave IP, it outputs normally; otherwise, ErrorID outputs SLAVE_IP_ERROR.
- If all the above conditions are met, it will check the disabled status of the slave node; if disabled, ErrorID outputs FAILURE_OBATION.
- After completing the parameter matching, obtain the current TCP Slave communication diagnosis to update the eState status in real time.

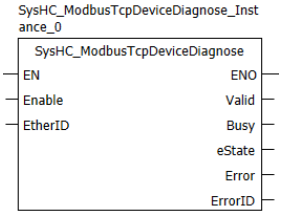
- **Program example**



18.4.6.3 SysHC_ModbusTcpDeviceDiagnose

This instruction gets the diagnostic state information from the Modbus-TCP slave.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
SysHC_ModbusTcpDeviceDiagnose	Get Modbus-TCP as slave diagnostic status information	FB		SysHC_ModbusTcpDeviceDiagnose_Instance_0(Enable :=, EtherID :=, Valid =>, Busy =>, eState =>, Error =>, ErrorID =>);

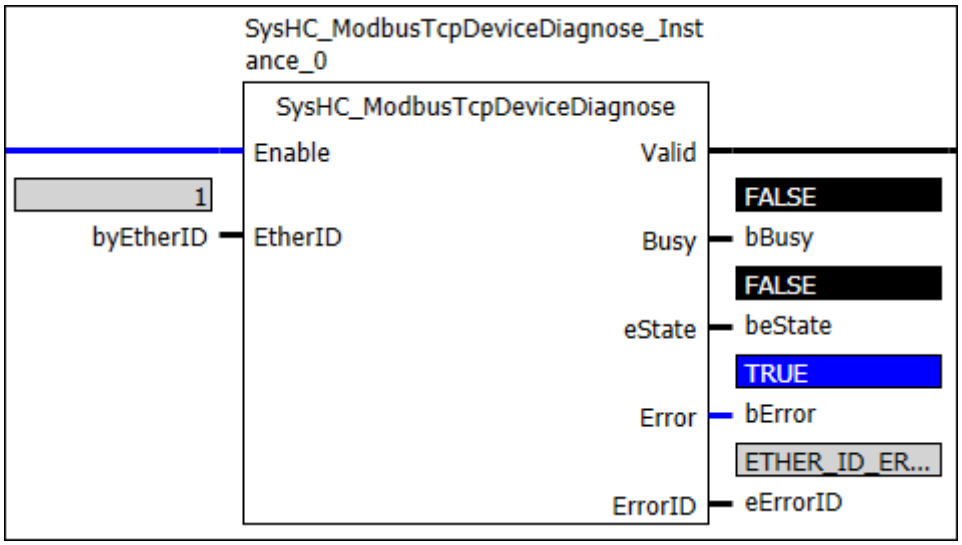
- **Related variables**

Variable

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Enable	Rising edge-triggered	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	EtherID	Network port ID	BYTE	-	0	Network port ID number.
Output	Valid	Output active	BOOL	TRUE, FALSE	FALSE	Instruction is actively executing.
	Busy	Instruction executing	BOOL	TRUE, FALSE	FALSE	Instruction is executing.
	eState	Diagnostic status bit	BOOL	TRUE, FALSE	FALSE	Diagnostic status bit.
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	Set to TRUE when an error occurs.
	ErrorID	Fault code	SYS_HC_ERROR	-	NO_ERROR	See 18.4.1.37 Fault Codes.

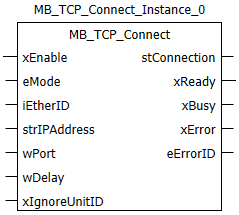
- **Function**
  - When Enable is TRUE, the function block is activated; otherwise, all outputs are invalid.
  - Checks whether the input EtherID is supported on the device. If not, ErrorID outputs ETHER_ID_ERROR.
  - When acting as a TCP slave, if the node is in a disabled state, ErrorID outputs FAILURE_OBATION.
  - After successful parameter matching, the eState status is updated in real time based on the current status of the specified Ethernet port.
- **Program example**



18.4.6.4 MB_TCP_Connect

This is the instruction for Modbus TCP protocol parameter configuration, which establishes a Modbus TCP connection.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
MB_TCP_Connect	Establish a Modbus TCP connection	FB		MB_TCP_Connect_Instance_0(xEnable := , eMode := , iEtherID := , strIPAddress := , wPort := , wDelay := , xIgnoreUnitID := , stConnection => , xReady => , xBusy => , xError => , eErrorID =>);

- **Related variables**

  Variable

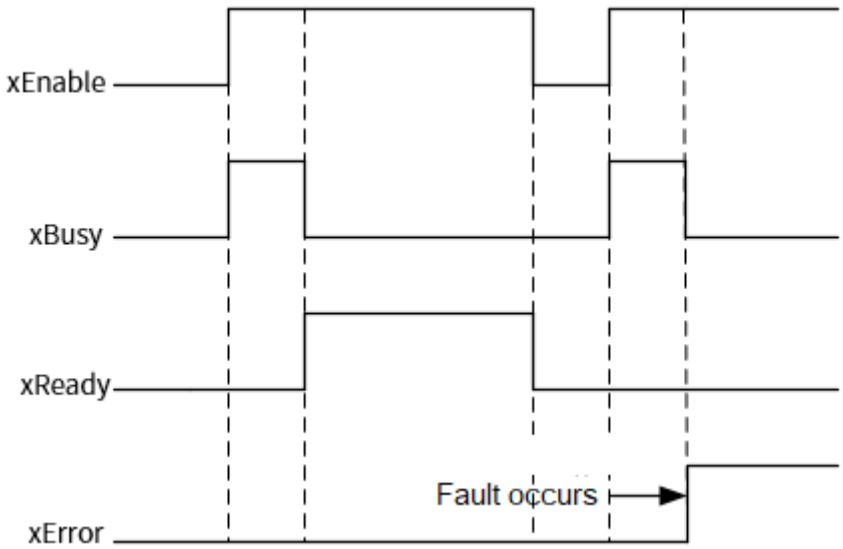
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Function block enable	BOOL	Follows data type	FALSE	FALSE: Disable network connection. TRUE: Enable network connection.
	eMode	Modbus TCP mode	MB_TCP_MODE	Follows data type	MB_TCP_MODE.SERVER	Modbus TCP Mode: Server (default). MB_TCP_MODE.CLIENT: Modbus TCP client. MB_TCP_MODE.SERVER: Modbus TCP server.
	iEtherID	Network port ID	INT	Follows data type	-1	Network port ID: -1: Automatic. 0: Ethernet port A. It changes in incremental mode.
	strIPAddress	IP address	STRING	Follows data type	''	IP Address.
	wPort	Terminal name	WORD	Follow data type	502	Port.
	wDelay	Frame interval	WORD	Follows data type	1	ModbusTCP frame interval.
	xIgnoreUnitID	Ignore unit_id	BOOL	Follows data type	FALSE	FALSE: Enable unit_id filtering. TRUE: Disable unit_id filtering.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	stConnection	Connection handle	CBus_Connection	Follows data type	-	Connection handle.
	xReady	Instruction status	BOOL	Follows data type	FALSE	Indicates the function block is active
	xBusy	Instruction execution flag	BOOL	Follows data type	FALSE	Function block execution flag
	xError	Error flag	BOOL	Follows data type	FALSE	Indicates the function block is faulty
	eErrorID	Fault code	CBUS_ERROR	Follows data type	NO_ERROR	Fault code of function block. For details, see 18.4.5.6 Fault Codes.

- **Function**

  This instruction allows modification of TCP parameters (such as frame intervals) during operation.

- **Sequence diagram**



- **Program example**

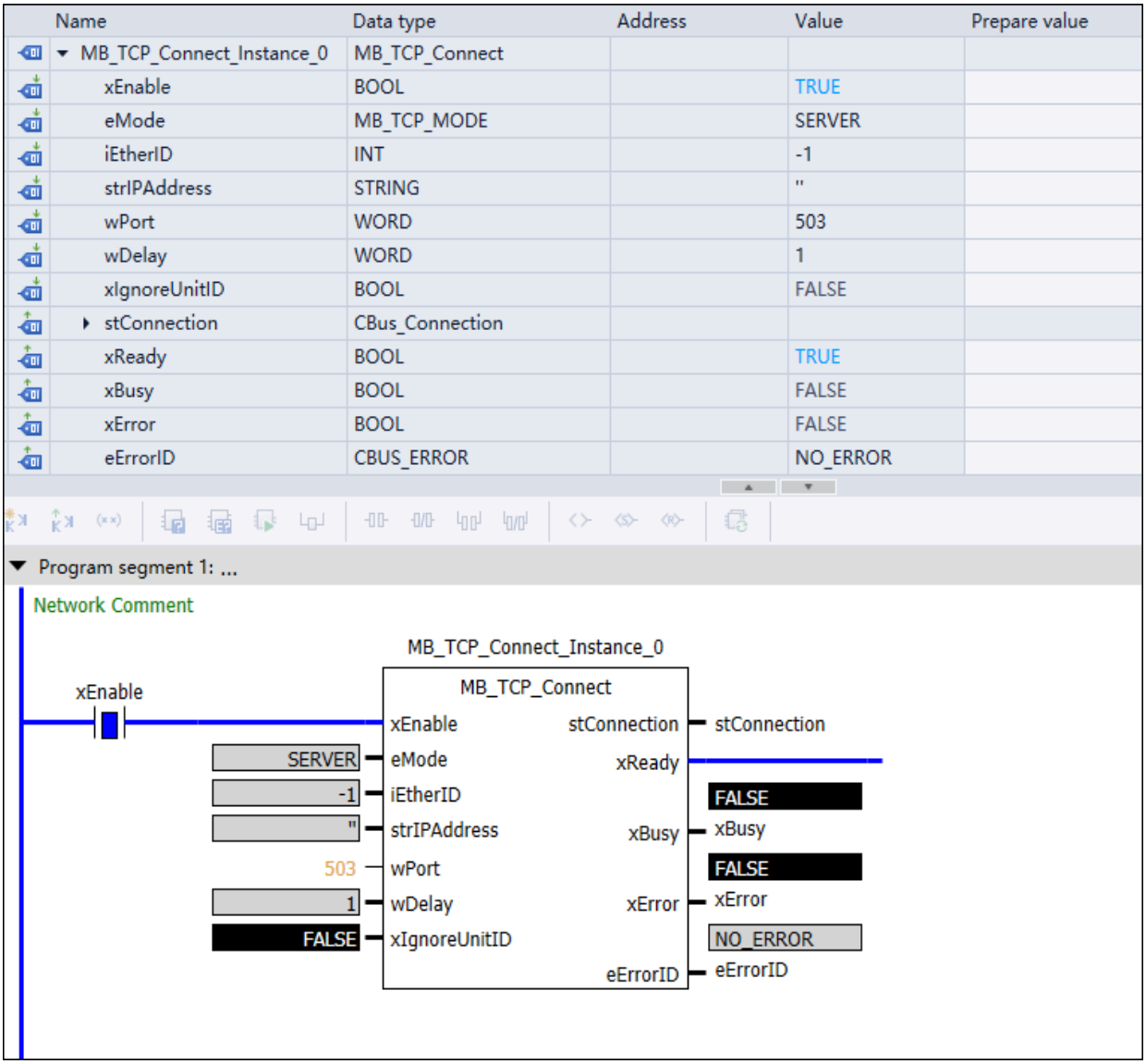
  ST:

Name	Data type	Address	Value	Prepare value
MB_TCP_Connect_0	MB_TCP_Connect			
xEnable	BOOL		TRUE	
eMode	MB_TCP_MODE		SERVER	
iEtherID	INT		-1	
strIPAddress	STRING		''	
wPort	WORD		503	
wDelay	WORD		5	
xIgnoreUnitID	BOOL		FALSE	
stConnection	CBus_Connection			
xReady	BOOL		TRUE	
xBusy	BOOL		FALSE	
xError	BOOL		FALSE	
eErrorID	CBUS_ERROR		NO_ERROR	

```
AS  FOR  WHI  REP  RET  EX  REG  //  (*  |=  |  |  .

MB_TCP_Connect_0(
    xEnable TRUE := xEnable TRUE ,
    eMode SERVER := MB_TCP_MODE.SERVER,
    iEtherID -1 := -1,
    strIPAddress '' := strIPAddress '' ,
    wPort 503 := 503,
    wDelay 5 := 5,
    xIgnoreUnitID FALSE := FALSE,
    stConnection=> stConnection,
    xReady TRUE => xReady TRUE ,
    xBusy FALSE => xBusy FALSE ,
    xError FALSE => xError FALSE ,
    eErrorID NO_ERROR => eErrorID NO_ERROR );
```

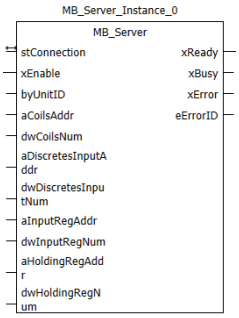
LD:



18.4.6.5 MB_Server

This is the Modbus TCP server communication instruction, which creates a Modbus TCP server.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
MB_Server	Create a Modbus TCP server	FB		MB_Server_Instance_0(stConnection := , xEnable := , byUnitID := , aCoilsAddr := , dwCoilsNum := , aDiscretesInputAddr := , dwDiscretesInputNum := , aInputRegAddr := , dwInputRegNum := , aHoldingRegAddr := , dwHoldingRegNum := , xReady => , xBusy => , xError => , eErrorID =>);

- **Related variables**

Variable

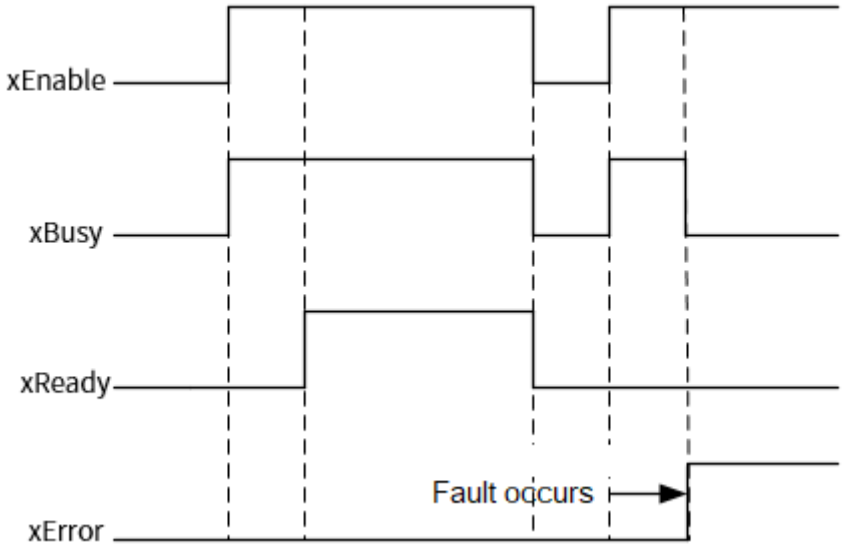
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input/ Output	stConnection	Connection handle	CBus_Connection	Follows data type	-	Connection handle.
Input	xEnable	Function block enable	BOOL	Follows data type	FALSE	Level triggered.
	byUnitID	Slave number	BYTE	Follows data type	1	Modbus slave number.
	aCoilsAddr	Coil data area	ANY	Follows data type	-	Supported function codes: 0x01/0x05/0x0F.
	dwCoilsNum	Number of coils	DWORD	Depends on data type	0	Number of accessible coils.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	aDiscretesInputAddr	Discrete input data area	ANY	Follows data type	-	Supported function code: 0x02.
	dwDiscretesInputNum	Number of discrete inputs	DWORD	Depends on data type	0	Number of accessible discrete inputs.
	aInputRegAddr	Input register data area	ANY	Follows data type	-	Supported function code: 0x04.
	dwInputRegNum	Number of input registers	DWORD	Depends on data type	0	Number of accessible input registers.
	aHoldingRegAddr	Holding register data area	ANY	Follows data type	-	Supported function codes: 0x03/0x06/0x10.
	dwHoldingRegNum	Number of holding registers	DWORD	Depends on data type	0	Number of accessible holding registers.
Output	xReady	Instruction status	BOOL	Follows data type	FALSE	Indicates the function block is active
	xBusy	Instruction execution flag	BOOL	Follows data type	FALSE	Function block execution flag
	xError	Error flag	BOOL	Follows data type	FALSE	Indicates the function block is faulty
	eErrorID	Fault code	CBUS_ERROR	Follows data type	NO_ERROR	Fault code of function block. For details, see 18.4.5.6 Fault Codes.

- **Function**

  This function initiates data transmission by transmitting the Modbus data frame over TCP.

- **Sequence diagram**



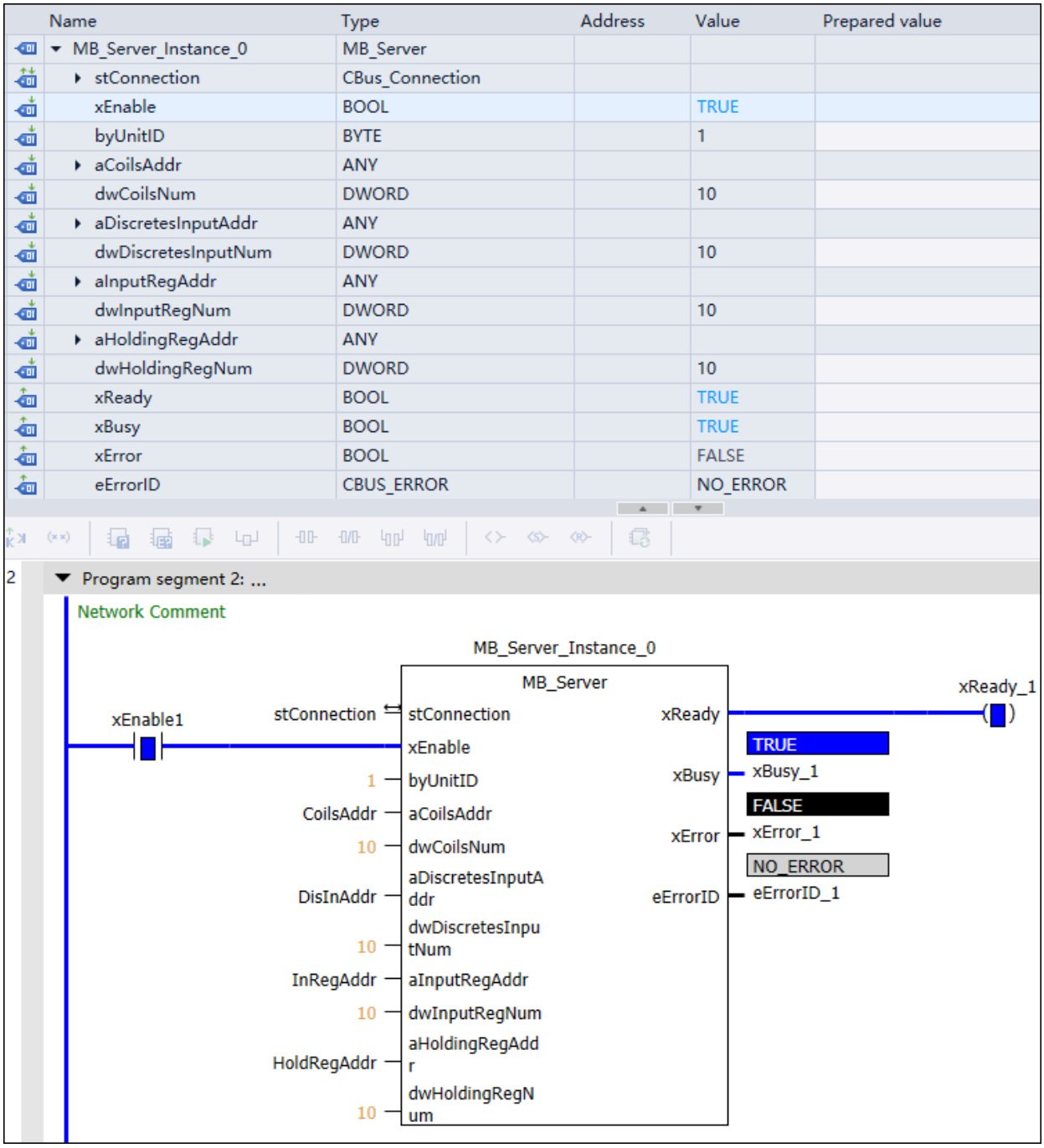
- **Program example**

  ST:

Name	Data type	Address	Value	Prepare value
MB_Server_0	MB_Server			
stConnection	CBus_Connection			
xEnable	BOOL		TRUE	
byUnitID	BYTE		1	
aCoilsAddr	ANY			
dwCoilsNum	DWORD		10	
aDiscretesInputAddr	ANY			
dwDiscretesInputNum	DWORD		10	
aInputRegAddr	ANY			
dwInputRegNum	DWORD		10	
aHoldingRegAddr	ANY			
dwHoldingRegNum	DWORD		10	
xReady	BOOL		TRUE	
xBusy	BOOL		TRUE	
xError	BOOL		FALSE	
eErrorID	CBUS_ERROR		NO_ERROR	

```
CAS FOR WHI REP RET EX REG
MB_Server_0(
    stConnection:= stConnection,
    xEnable TRUE := xEnable1 TRUE ,
    byUnitID 1 := 1,
    aCoilsAddr:= CoilsAddr,
    dwCoilsNum 10 := CoilsNum 10 ,
    aDiscretesInputAddr:= DisInAddr,
    dwDiscretesInputNum 10 := DisInNum 10 ,
    aInputRegAddr:= InRegAddr,
    dwInputRegNum 10 := InRegNum 10 ,
    aHoldingRegAddr:= HoldRegAddr,
    dwHoldingRegNum 10 := HoldRegNum 10 ,
    xReady TRUE => xReady_1 TRUE ,
    xBusy TRUE => xBusy_1 TRUE ,
    xError FALSE => xError_1 FALSE ,
    eErrorID NO_ERROR => eErrorID_1 NO_ERROR );RETURN
```

  LD:



- **Precautions**
  - It must be used in conjunction with the MB_TCP_Connect instruction, and the eMode must be set to SERVER.
  - The connection handle of the MB_TCP_Connect instruction supports only one MB_Server instance.
  - It is recommended that the MB_TCP_Connect and MB_Server instructions be processed within the same user task and program cycle. If they are in different priority tasks, operational errors may occur due to priority interruptions.
  - When the MB_Server instruction is in an active and accessible status, xReady/xBusy will be set. If there is an error, xError will be set, and eErrorID will display the error code.
  - MB_Server service requests still support access when the PLC DIP switch is set to the STOP position.
  - If the StrIPAddress of MB_TCP_Connect is empty, any remote Modbus TCP client is allowed to connect and access the configured data area. If a valid IP address is input in StrIPAddress, the IP filtering function is enabled, allowing only the remote ModbusTCP client with that IP address to access.
  - Access area description:

Data Area Variable	Area Name	Supported Function Code	Accessible Area Size	Supported Data Type
aCoilsAddr	Coil data area	READ_COILS(01) WRITE_SINGLE_COIL(05) WRITE_MULTIPLE_COILS(15)	1 to 65536	BOOL, BYTE, ARRAY
aDiscretesInputAddr	Discrete input data area	READ_DISCRETE_INPUTS(02)	1 to 65536	
aInputRegAddr	Input register data area	READ_INPUT_REGISTERS(04)	1 to 65536	WORD, DWORD, INT, UINT, DINT, UDINT, REAL, ARRAY
aHoldingRegAddr	Holding register data area	READ_HOLDING_REGISTERS(03) WRITE_SINGLE_REGISTER(06) WRITE_MULTIPLE_REGISTERS(16)	1 to 65536	

- Unsupported types for access area:

  Multidimensional arrays.

  64-bit data types: LWORD, LINT, ULINT, and LREAL.

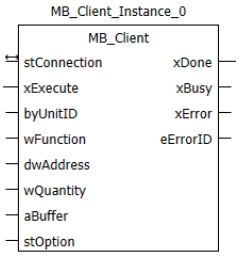
  Special types: STRING, WSTRING, TIME, TIME_OF_DAY(TOD), DATE, DATE_AND_TIME(DT), POINTER, and STRUCT.

- Access areas cannot be directly set to I/Q/M direct variables. They must be mapped to the above supported variables or array data via AT mapping.
- Access areas can be freely combined. Unused access areas can be left empty, but all four access areas cannot be empty at the same time.
- Online parameter modifications will not take effect immediately, requiring the instruction to be re-enabled.

18.4.6.6 MB_Client

This is the Modbus TCP client communication instruction, which creates a Modbus TCP client.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
MB_Client	Create Modbus TCP client	FB		MB_Client_Instance_0(stConnection := , xExecute := , byUnitID := , wFunction := , dwAddress := , wQuantity := , aBuffer := , stOption := , xDone => , xBusy => , xError => , eErrorID =>);

- **Related variables**

  Variable

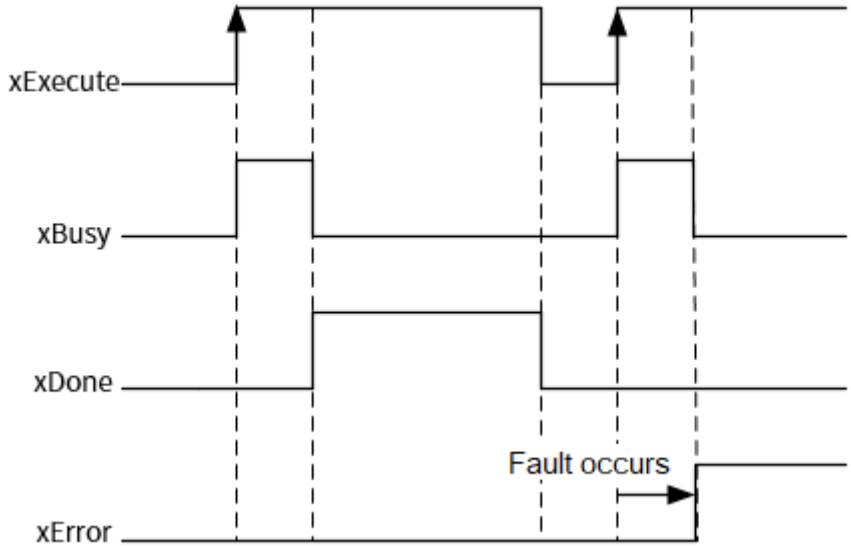
Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input/Output	stConnection	Connection handle	CBus_Connection	Follows data type	-	Connection handle.
Input	xExecute	Function block enable	BOOL	Follows data type	FALSE	Edge triggered.
	byUnitID	Slave number	BYTE	Follows data type	1	Request access to the slave number.
	wFunction	Function code	MB_FUNCTION	Follows data type	0	Modbus function code.
	dwAddress	Slave data address	DWORD	Depends on data type	0	Starting address for access to slave data.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	wQuantity	Number of accessible address	WORD	Follow data type	0	Maximum number of function codes: - 01/02: 2000. - 03/04: 125. - 15: 1968. - 16: 123.
	aBuffer	Read/write buffer	ANY	Follows data type	-	Modbus data buffer.
	stOption	Configuration options	MB_OPTION	Follows data type	-	Optional configuration parameters for Modbus protocol.
Output	xDone	Instruction completion flag	BOOL	Follows data type	FALSE	Function block execution completion flag
	xBusy	Instruction execution flag	BOOL	Follows data type	FALSE	Function block execution flag.
	xError	Error flag	BOOL	Follows data type	FALSE	Indicates the function block is faulty
	eErrorID	Fault code	CBUS_ERROR	Follows data type	NO_ERROR	Fault code of function block. For details, see 18.4.5.6 Fault Codes.

- **Function**

  This function initiates data transmission by transmitting the Modbus data frame over TCP.

- **Sequence diagram**



- **Program example**

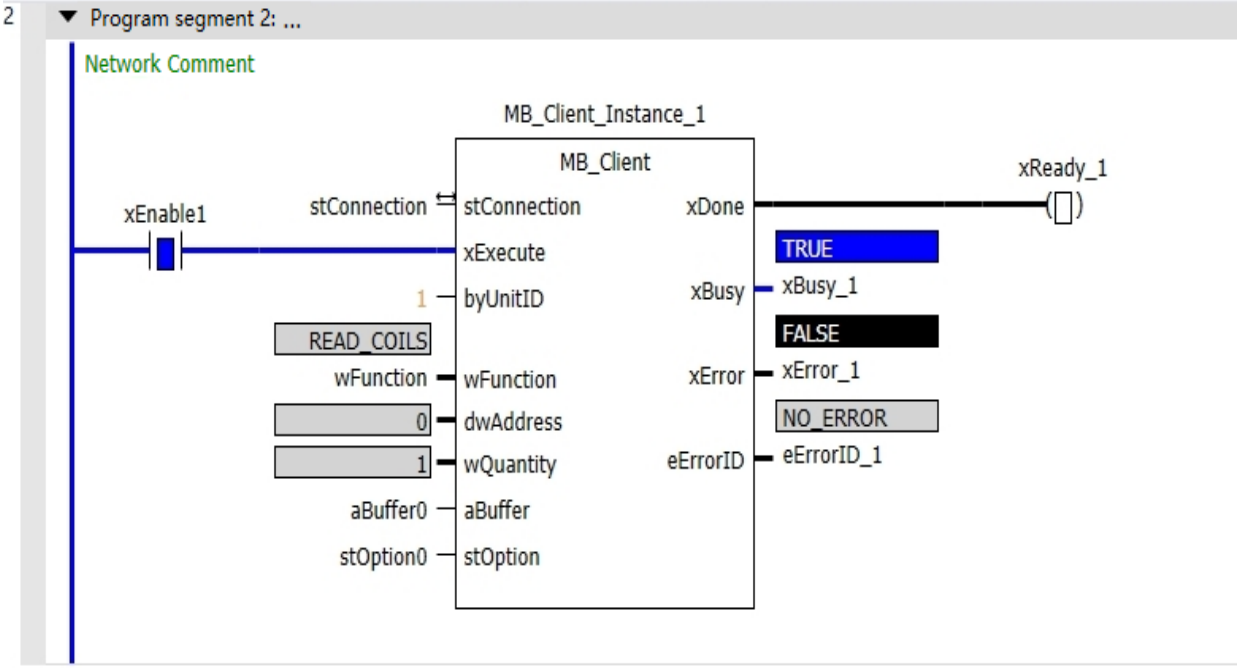
  ST:

Name	Data type	Address	Value	Prepare value
MB_Client_Instance_0	MB_Client			
stConnection	CBus_Connection			
xExecute	BOOL		TRUE	
byUnitID	BYTE		1	
wFunction	MB_FUNCTION		READ_COILS	
dwAddress	DWORD		0	
wQuantity	WORD		1	
aBuffer	ANY			
stOption	MB_OPTION			
xDone	BOOL		TRUE	
xBusy	BOOL		FALSE	
xError	BOOL		FALSE	
eErrorID	CBUS_ERROR		NO_ERROR	
eStatus	CBUS_STATE		FINISH	
xBool	INT		0	

```
MB_Client_Instance_0(
stConnection := stConnection,
xExecute TRUE := xEnable1 TRUE ,
byUnitID 1 := 1,
wFunctionREAD_CO... := 1,
dwAddress 0 := 0,
wQuantity 1 := 1,
aBuffer := aBuffer0,
stOption := stOption0,
xDone TRUE => xDone TRUE ,
xBusy FALSE => xError FALSE ,
xError FALSE =>xError FALSE ,
eErrorID NO_ERROR => eErrorID NO_ERROR );RETURN
```

  LD:

Name	Data type	Address	Value	Prepare value
MB_Client_Instance_1	MB_Client			
xReady_1	BOOL		FALSE	
xBusy_1	BOOL		TRUE	
xError_1	BOOL		FALSE	
eErrorID_1	CBUS_ERROR		NO_ERROR	
stOption0	MB_OPTION			
aBuffer0	ARRAY [0..100] OF BYTE			
xDone	BOOL		FALSE	
wFunction	MB_FUNCTION		READ_COILS	



- **Precautions**
  - It must be used in conjunction with the MB_TCP_Connect instruction, and the eMode must be set to CLIENT.
  - Connection handle configured by the MB_TCP_Connect instruction, supports one or more MB_Client instances.
  - It is recommended to process the MB_TCP_Connect and MB_Client instructions within the same user task and program cycle. If processed in different priority tasks, operational anomalies may occur due to priority interruptions.
  - When the MB_Client instruction is in the xBusy set status, repeated instructions will be ignored until xDone or xError is set.
  - The MB_Client instruction cannot be executed in startup, diagnostic, safety modes, or under time error conditions.
  - byUnitID supports only standard station addresses: 1~255, 0 is used for broadcasting.
  - Function code and number of accessible addresses
    - Function code: READ_COILS/ READ_DISCRETE_INPUTS, the maximum number of accessible addresses supported is 2000.
    - Function code: WRITE_MULTIPLE_COILS, the maximum number of accessible addresses supported is 1968.

- Function code: READ_HOLDING_REGISTERS/ READ_INPUT_REGISTERS, the maximum number of accessible addresses supported is 125.
- Function code: WRITE_MULTIPLE_REGISTERS, the maximum number of accessible addresses supported is 123.

◦ Data types loaded by aBuffer:

- Supported types:

  One-dimensional array.

  Data types of the arrays/variables:

  Access coil data: BYTE.

  Access register data: WORD, DWORD, INT, UINT, DINT, UDINT, REAL.

- Unsupported types

  Multidimensional arrays.

  64-bit data types: LWORD, LINT, ULINT, and LREAL.

  Special types: STRING, WSTRING, TIME, TIME_OF_DAY(TOD), DATE, DATE_AND_TIME(DT), POINTER, and STRUCT.

◦ aBuffer cannot be directly set to I/Q/M direct variables. It must be mapped to the above supported variables or array data via AT mapping.

◦ Online parameter modifications will not take effect immediately, requiring the instruction to be re-enabled.

18.4.6.7 Example of ModbusTCP Communication Instruction Usage

Example of ModbusTCPServer communication instruction usage

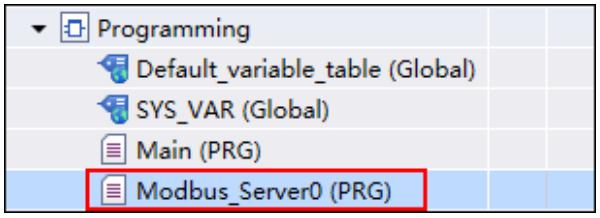
This section takes the ModbusTCPServer communication instruction as an example. An EVO820 controller is employed as the Modbus TCP server, and the NetAssist acts as the Modbus TCP client.

Prerequisites

- A project has been created and the device added. For specific operations, see 7 Project Deployment.
- The Modbus library has been referenced. For the reference operations, see 12.3 Library Reference.

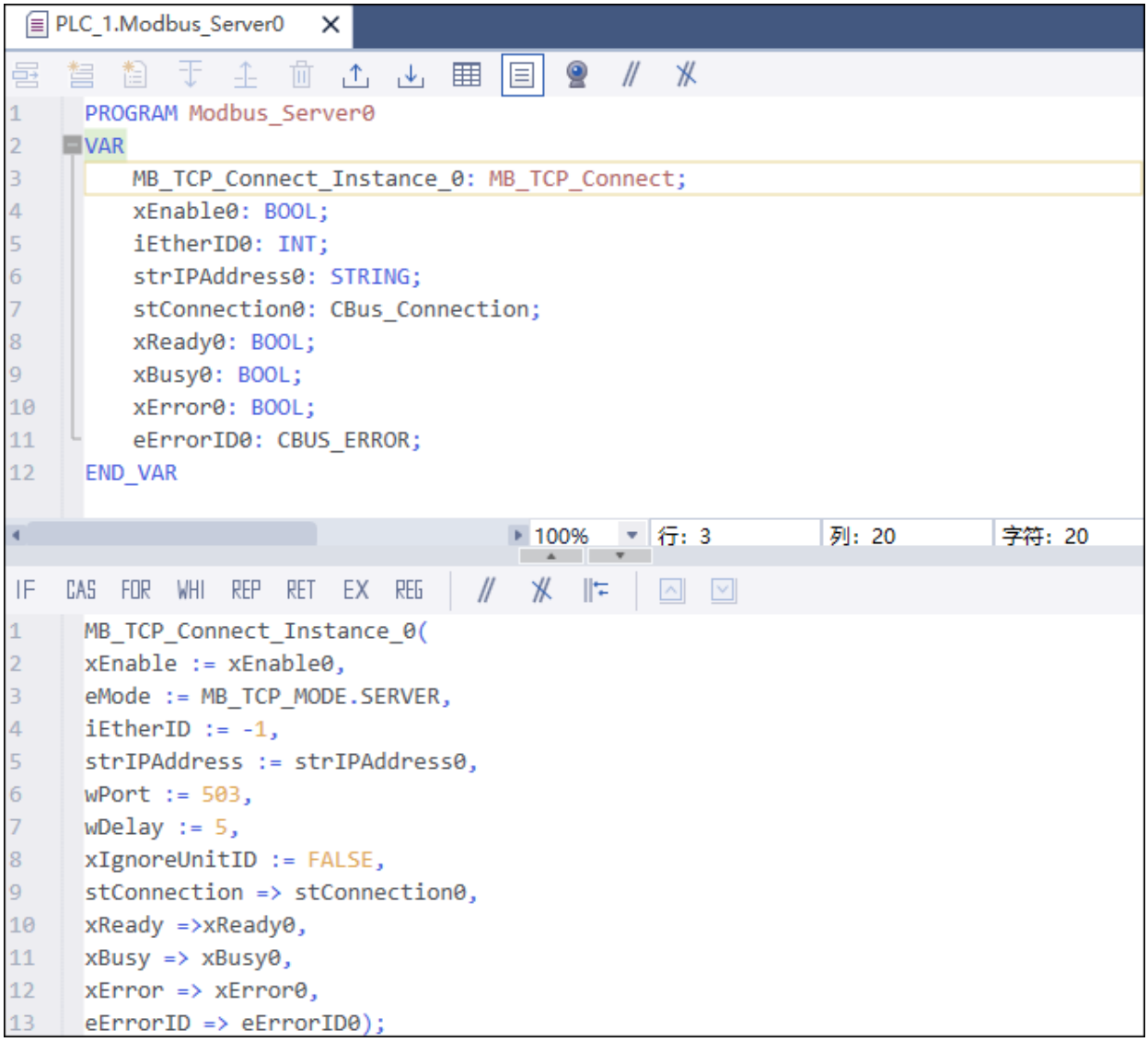
Procedure

1. Click "Programming" in the left project tree to create a POU of PRG type, with ST and LD languages both supported. This section takes the ST language as an example.



2. Add the "MB_TCP_Connect" instruction: Double-click to open the POU interface, select the "MB_TCP_Connect" instruction from the right Toolbox, and then double-click or drag it directly to the program editing area to create an instruction instance object.

3. In the table view/text view variable declaration editor, add variables, set properties such as variable name and data type, and configure the instruction-related pins. The text view variable declaration editor is used as an example below.



NOTE

eMode of current example MB_Server instruction is set to MB_TCP_MODE.SERVER.

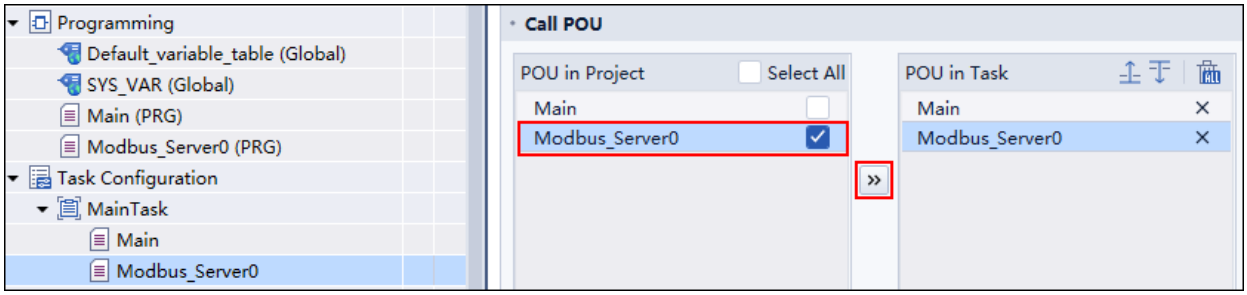
4. Add the "MB_Server" instruction: Double-click to open the POU interface, select the "MB_Server" instruction from the right Toolbox, and then double-click or drag it directly to the program editing area to create an instruction instance object.

5. In the table view/text view variable declaration editor, add variables, set properties such as variable name and data type, and configure the instruction-related pins. The text view variable declaration editor is used as an example.

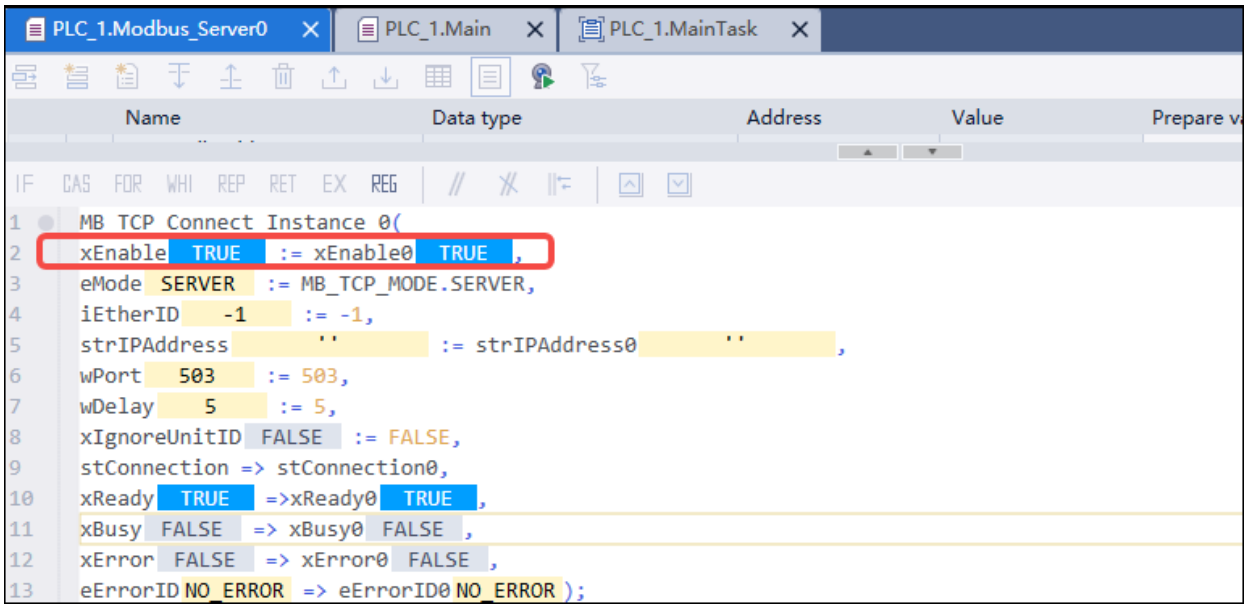
NOTE

stConnection must use the output parameter stConnection0 of the "MB_TCP_Connect" instruction.

6. Call Modbus_Server0 in MainTask, download it to the controller, and start its operation.



7. Open the iFA software, switch to the online mode, and set the xEnable pin of the "MB_TCP_Connect" instruction to TRUE to start the Modbus connection.



8. Open the iFA software, switch to the online mode, and set the XEnable pin of the "MB_Server" instruction to TRUE to start the Modbus service.

9. Open the NetAssist. In its network settings, select TCP_Client, configure the Remote Host Address to match the device's IP address (e.g., 192.168.1.88), and configure the Remote Host Port number to match the port number (e.g., 503). Then click "Connect".

10. Open the NetAssist, select "MODBUS Debugger" as the debugging tool, choose the "Modbus Instruction" panel in the right-side menu bar, configure the corresponding information, and then click "Send".

11. The transmitted and received data can be viewed in the NetAssist. For easier observation, convert the values into hexadecimal for viewing and sending.

Result

Normal communication and data exchange between the EVO820 controller and the NetAssist.

Example of ModbusTCPClient communication instruction usage

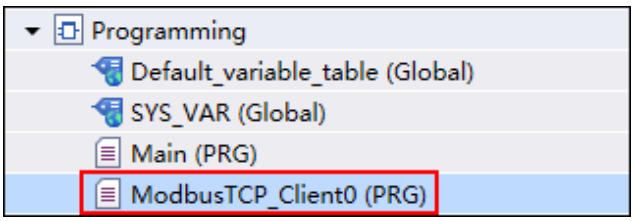
This section takes the ModbusTCPClient communication instruction as an example. An EVO820 controller is employed as the Modbus TCP client, and the NetAssist acts as the Modbus TCP client.

Prerequisites

- A project has been created and the device added. For specific operations, see 7 Project Deployment.
- The Modbus library has been referenced. For the reference operations, see 12.3 Library Reference.

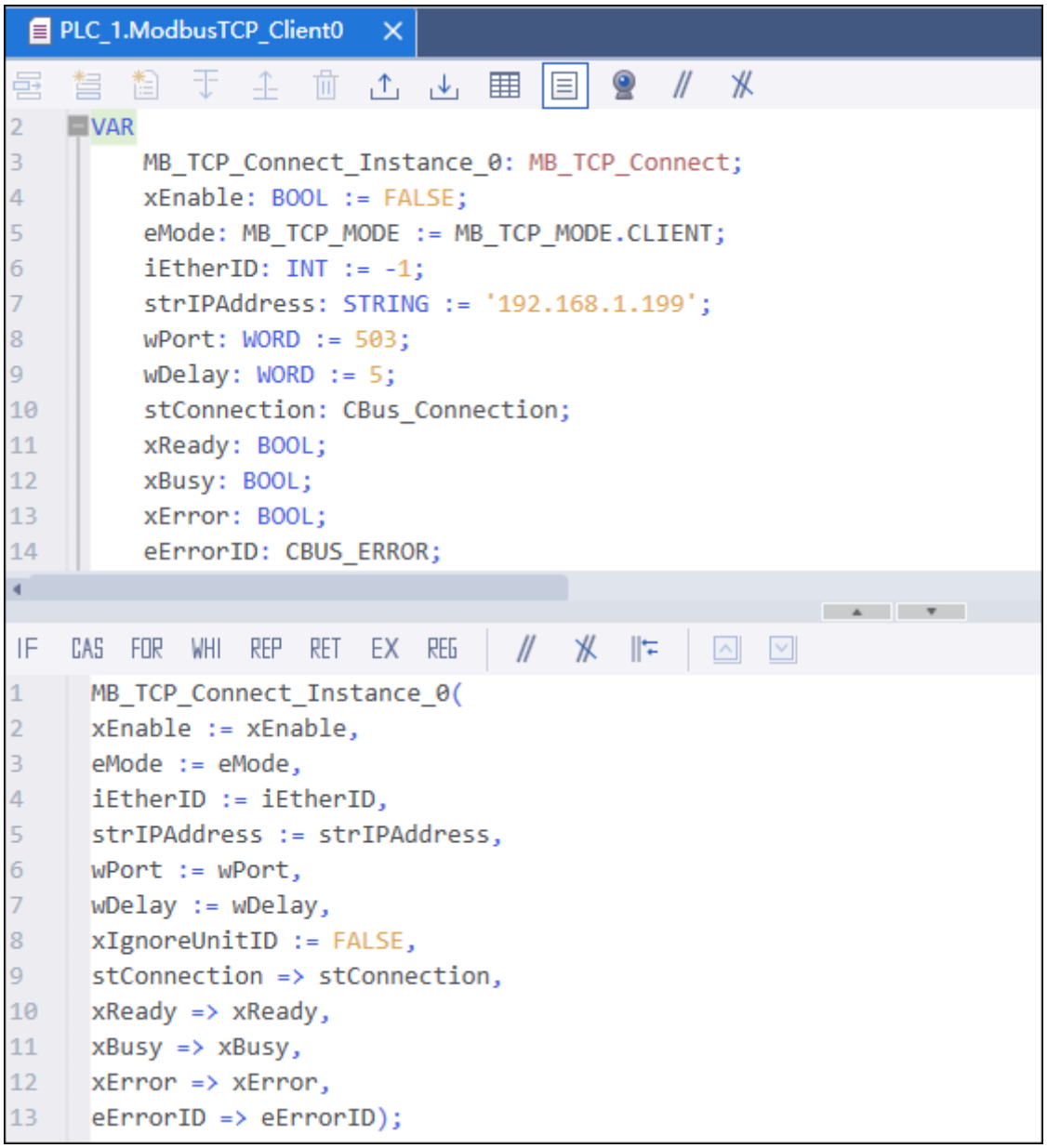
Procedure

1. Click "Programming" in the left project tree to create a POU of PRG type, with ST and LD languages both supported. This section takes the ST language as an example.



2. Add the "MB_TCP_Connect" instruction: Double-click to open the POU interface, select the "MB_TCP_Connect" instruction from the right Toolbox, and then double-click or drag it directly to the program editing area to create an instruction instance object.

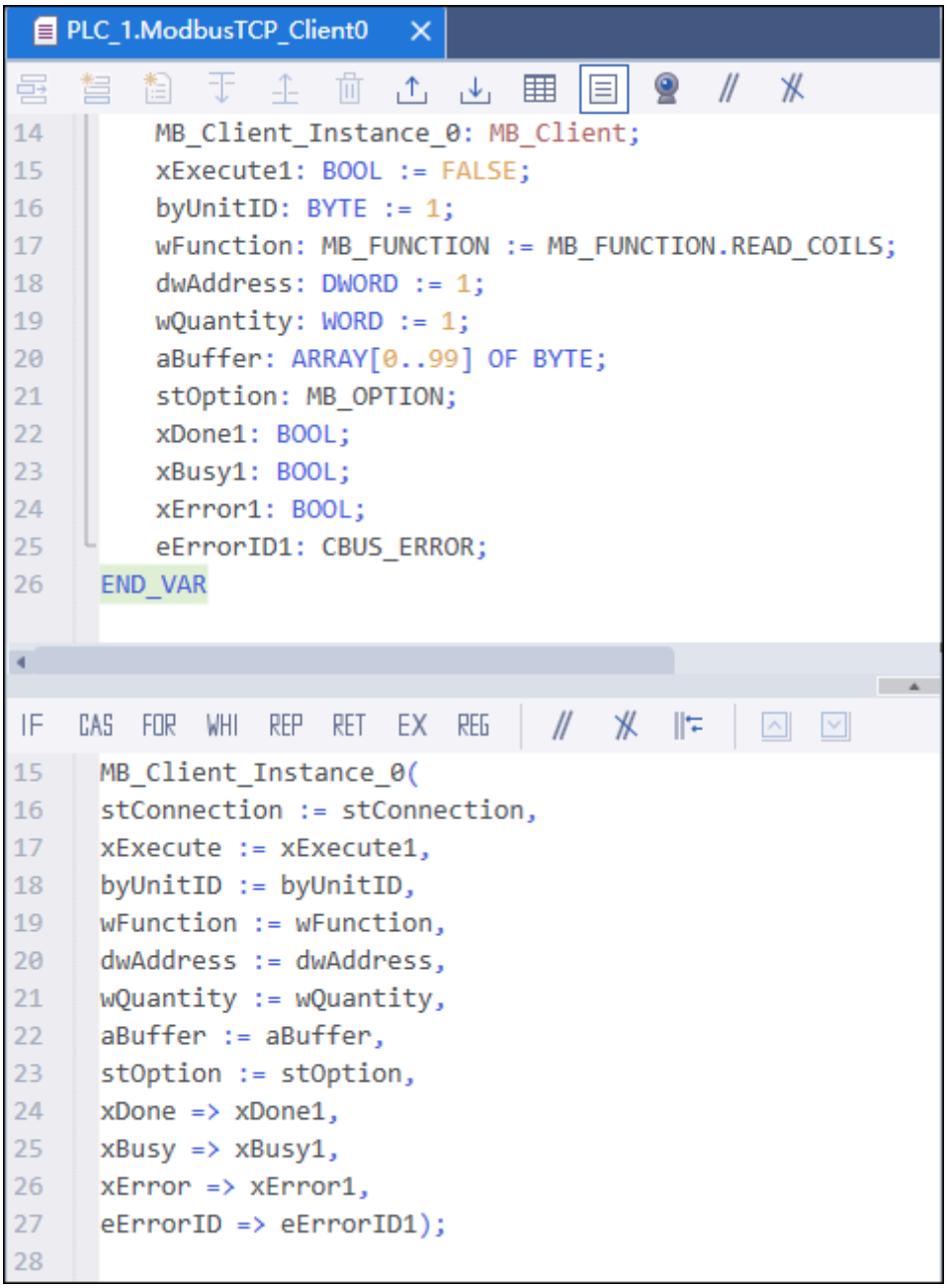
3. In the table view/text view variable declaration editor, add variables, set properties such as variable name and data type, and configure the instruction-related pins. The text view variable declaration editor is used as an example.



> NOTE
>
> eMode of current example MB_Client instruction is set to MB_TCP_MODE.CLIENT.

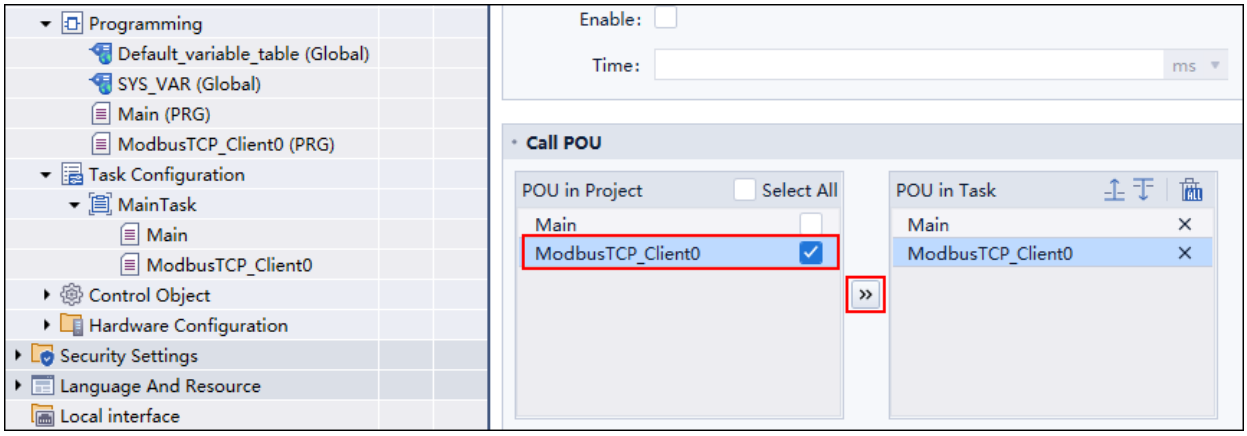
4. Add the "MB_CLIENT" instruction: Double-click to open the POU interface, select the "MB_CLIENT" instruction from the right Toolbox, and then double-click or drag it directly to the program editing area to create an instruction instance object.

5. In the table view/text view variable declaration editor, add variables, set properties such as variable name and data type, and configure the instruction-related pins. The text view variable declaration editor is used as an example.



> NOTE
>
> stConnection must use the output parameter stConnection of the "MB_TCP_Connect" instruction.

6. Call ModbusTCP_Client0 in MainTask, download it to the controller, and start its operation.



7. Open the NetAssist, select TCP_Server as protocol type in network settings, configure the correct Local Host Address (same network segment as device IP), set local port to 503, and then click "Open" to start the service.

7. Open the iFA software, switch to the online mode, and set the xEnable pin of the "MB_TCP_Connect" instruction to TRUE to eatablish the Modbus connection.

```
MB_TCP_Connect_Instance_0(
xEnable TRUE := xEnable TRUE,
eMode CLIENT := eMode CLIENT,
iEtherID -1 := iEtherID -1,
strIPAddress '192.168.1.199' := strIPAddress '192.168.1.199',
wPort 503 := wPort 503,
wDelay 503 := wPort 503,
xIgnoreUnitID FALSE := FALSE,
stConnection => stConnection,
xReady TRUE => xReady TRUE,
xBusy FALSE => xBusy FALSE,
xError FALSE => xError FALSE,
eErrorID NO_ERROR => eErrorID NO_ERROR );
```

9. Open the NetAssist and configure automatic data responses.

Configure the return information for the first packet of auto-response data. The received data is as follows: 00 01 00 00 00 06 01 01 00 01 00 01

Configure the corresponding response data as: 00 01 00 00 00 04 01 01 01 00

Then start the automatic response.

10. Open the iFA software, switch to the online mode, configure the corresponding parameters of the "MB_Client" instruction, and then set the xExecute pin from FALSE to TRUE to start the Modbus client for transmitting data.

```
MB_Client_Instance_0(
stConnection := stConnection,
xExecute TRUE := xExecute1 TRUE ,
byUnitID 1 := byUnitID 1 ,
wFunctionREAD_CO... := wFunctionREAD_CO...,
dwAddress 1 := dwAddress 1
wQuantity 1 := wQuantity 1 ,
aBuffer := aBuffer,
stOption := stOption,
xDone TRUE => xDone1 TRUE ,
xBusy FALSE => xBusy1 FALSE ,
xError FALSE => xError1 FALSE ,
eErrorID NO_ERROR => eErrorID1 NO_ERROR );
```

11. The transmitted and received data can be viewed in the NetAssist. For easier observation, convert the values into hexadecimal for viewing and sending.

Result

Normal communication and data exchange between the EVO820 controller client and the NetAssist.

18.4.7 Free Serial Port

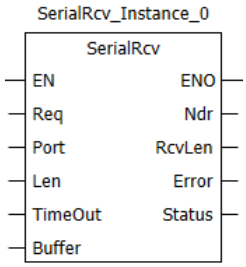
18.4.7.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.4.7.2 SerialRcv	FB	Serial port freeport reception	SerialCom	Library version 1.1.0.0 and later
18.4.7.3 SerialSend	FB	Serial port free port data sending instruction		Library version 1.1.0.0 and later
18.4.7.4 SerialRcvV2	FB	Serial port freeport reception		Library version 1.2.0.0 and later
18.4.7.5 SerialSendV2	FB	Serial port free port data sending instruction		Library version 1.2.0.0 and later
18.4.7.6 Serial_Connect	FB	Create a serial port connection	Modbus	Library version 1.2.0.0 and later
18.4.7.7 Serial_Recv	FB	Receive serial port data		Library version 1.2.0.0 and later
18.4.7.8 Serial_Recv_By_Idle_Time	FB	Receive serial port data during bus idle time		Library version 1.2.0.0 and later
18.4.7.9 Serial_Recv_Ch	FB	Receive serial port data with start/end character		Library version 1.2.0.0 and later
18.4.7.10 Serial_Send	FB	Send serial port data		Library version 1.2.0.0 and later

18.4.7.2 SerialRcv

This instruction receives data through serial ports by using freeport.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
SerialRcv	Serial port freeport reception	FB		SerialRcv_Instance_0(Req := , Port := , Len := , TimeOut := , Buffer := , Ndr => , RcvLen => , Error => , Status =>);

- **Related variables**

Variable

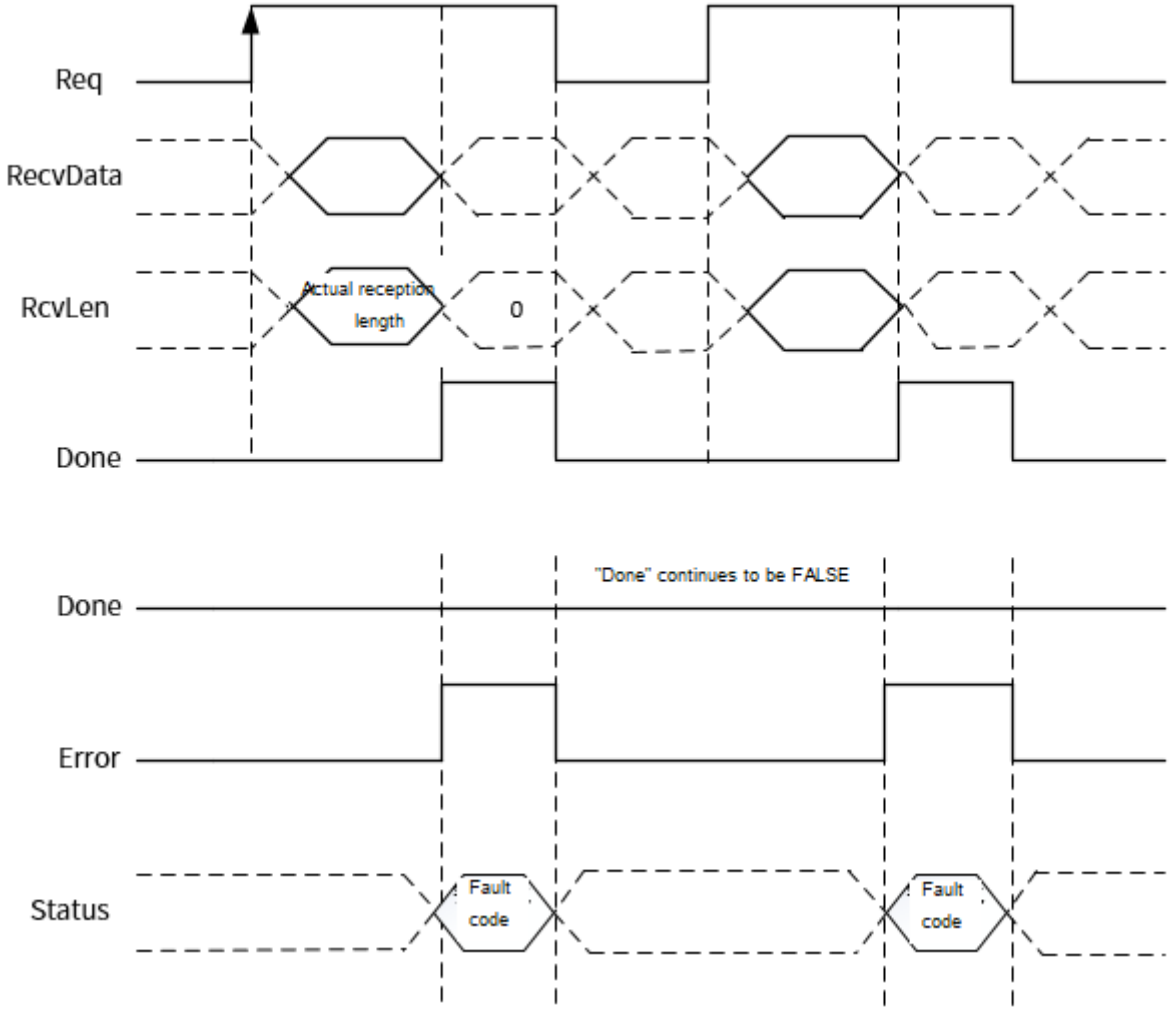
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Req	Data enable start bit	BOOL	Follows data type	FALSE	Enable start bit, active at high level.
	Port	Communication port	UINT	Follows data type	0	Serial port number configured through the software interface. 0: Built-in COM port of the PLC. 1: Expansion card A on the PLC 2: Expansion card B on the PLC
	Len	Received data length	UINT	1 to 1024	1024	Received data length.
	TimeOut	Timeout period	UINT	Follows data type	0	Timeout period in ms, with a default value of 0 indicating no timeout.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Buffer	Data reception buffer	POINTER TO BYTE	Follows data type	-	Pointer to the received data buffer.
Output	Ndr	Completion flag	BOOL	Follows data type	FALSE	Completion flag: FALSE: Reception not started or in progress. TRUE: Reception done.
	RcvLen	Received data length	UINT	1 to 1024	0	Actual length of received data.
	Error	Error flag	BOOL	Follows data type	FALSE	Error flag: 0: No error. 1: Reception error.
	Status	Status flag	WORD	Follow data type	1000	See Status code.

- **Function**

  This is the serial port free port data reception instruction.

- **Sequence diagram**



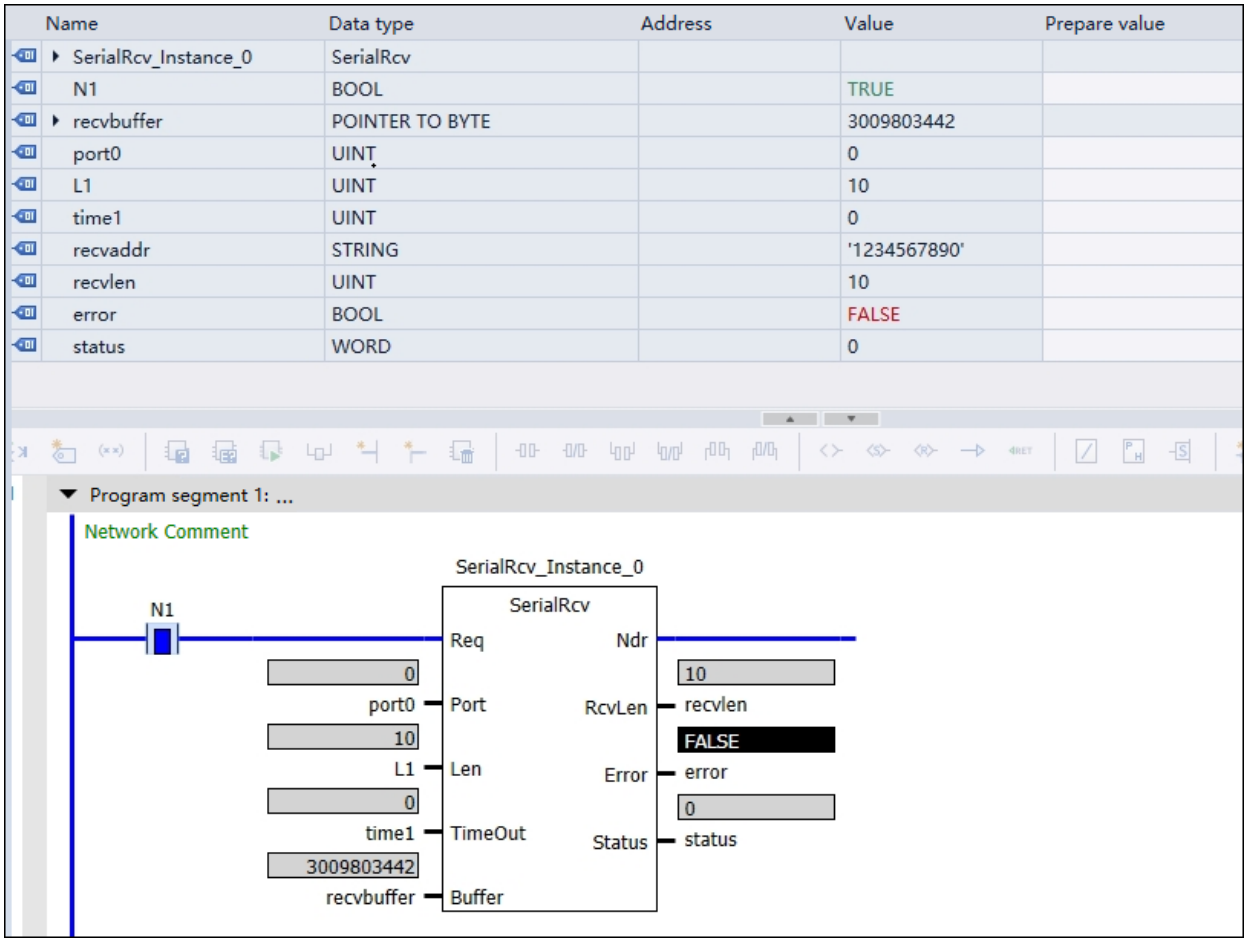
- **Program example**

  ST:

Name	Data type	Address	Value	Prepare value
SerialRcv_Instance_0	SerialRcv			
recvbuffer	POINTER TO BYTE		3009802426	
port0	UINT		0	
L1	UINT		10	
time1	UINT		0	
recvaddr	STRING		'1234567890'	
recvlen	UINT		10	
error	BOOL		FALSE	
status	WORD		0	
done	BOOL		TRUE	

```
SerialRcv_Instance_0(
Req TRUE := TRUE,
Port 0 := port0 0 ,
Len 10 := L1 10 ,
TimeOut 0 := time1 0 ,
Buffer 3009802426 := recvbuffer 3009802426 ,
Ndr TRUE => done TRUE ,
RcvLen 10 =>recvlen 10 ,
Error FALSE => error FALSE ,
Status 0 => status 0 );RETURN
```

  LD:



- **Precautions**
  - The actual length of received byte and baud rate are related to the task cycle. If the task cycle is shorter than the time required to receive the bytes, only part of the data may be received, because the baud rate determines the time needed to receive one byte of data.
  - Ensure that the size of data reception area is greater than or equal to the received byte length; otherwise, there is a risk of memory access out of bounds.
- **Status code**

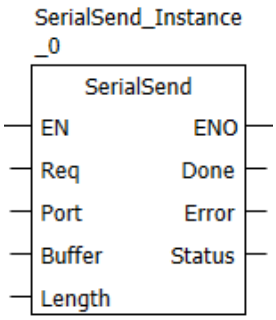
Status code	Note
0	Done
1000	Function not activated
1001	Serial port number error
1007	Reception address error
1008	No data
2000	Numbers of bytes received error
2001	Incorrect buffer parameter
3000	Data being received
3001	Data reception timeout

Status code	Note
3002	Data reception failed

18.4.7.3 SerialSend

This is the serial port free port data sending instruction.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
SerialSend	Serial port free port data sending instruction	FB		SerialSend_Instance_0(Req := , Port := , Buffer := , Length:=, Done => , Error => , Status =>);

- **Related variables**

Variable

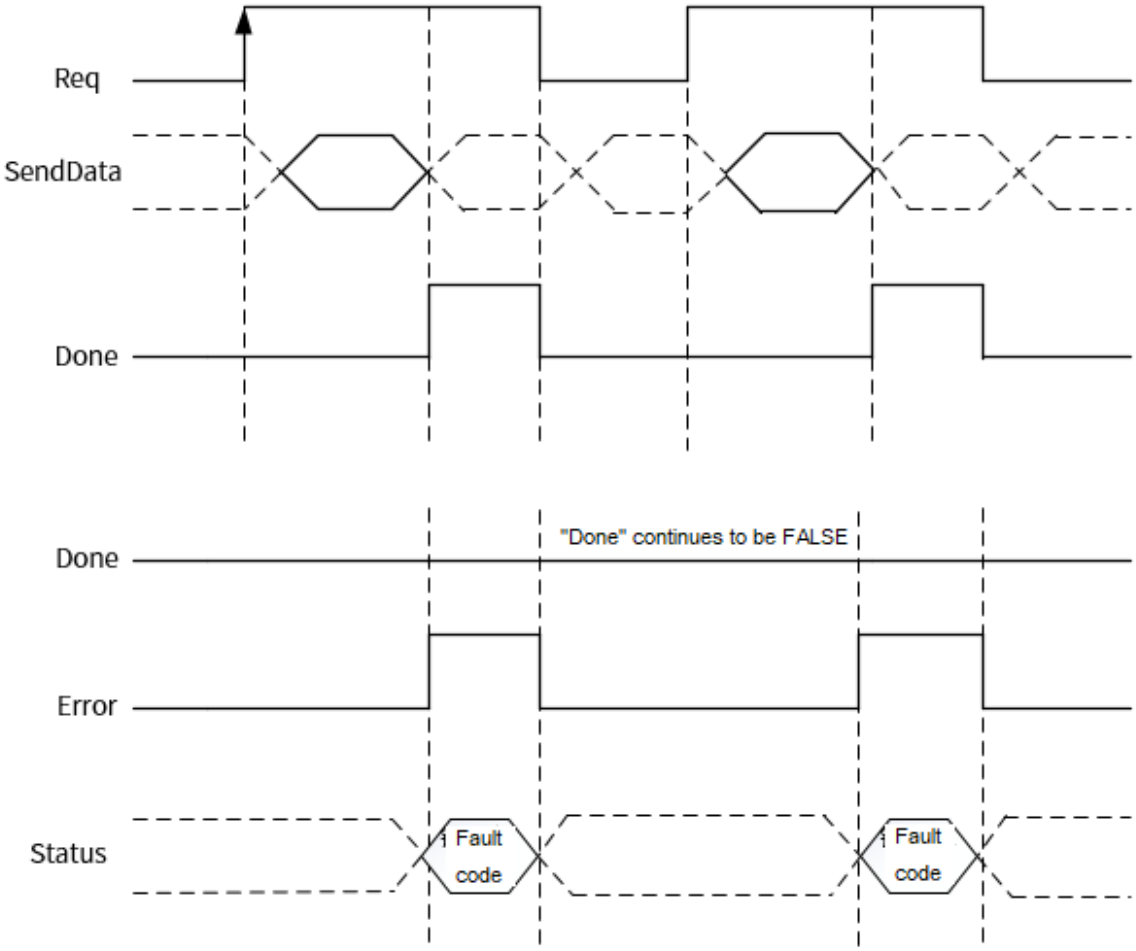
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Req	Enable start bit to send data	BOOL	Follows data type	FALSE	Enable start bit, active on rising edge.
	Port	Communication port	UINT	Follows data type	0	Serial port number configured through the software interface. 0: Built-in COM port of the PLC. 1: Expansion card A on the PLC 2: Expansion card B on the PLC
	Buffer	Data transmission buffer	POINTER TO BYTE	Follows data type	-	Pointer to the sending data buffer.
	Length	Transmit data length	UINT	1 to 1024	1024	Receive and transmit data length.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	Done	Completion flag	BOOL	Follows data type	FALSE	Completion flag: FALSE: Sending not started or in progress. TRUE: Sending done.
	Error	Error flag	BOOL	Follows data type	FALSE	Error flag: 0: No error. 1: Sending error.
	Status	Status flag	WORD	Follow data type	1000	See Status code.

- **Function**

  This is the serial port free port data sending instruction.

- **Sequence diagram**



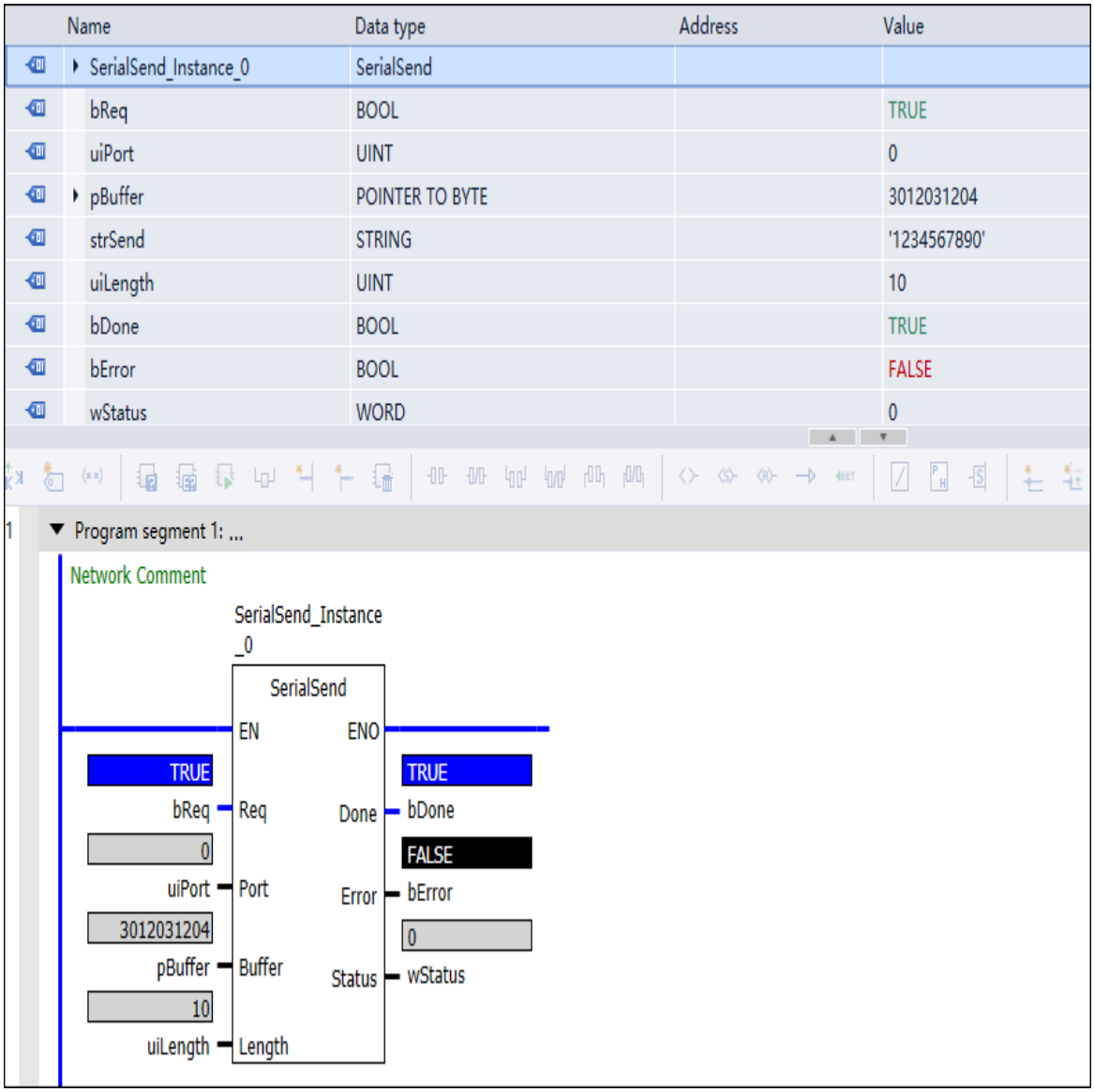
- **Program example**

  ST:

Name	Data type	Address	Value
SerialSend_Instance_0	SerialSend		
bReq	BOOL		TRUE
uiPort	UINT		0
pBuffer	POINTER TO BYTE		3012031068
strSend	STRING		'1234567890'
uiLength	UINT		10
bDone	BOOL		TRUE
bError	BOOL		FALSE
wStatus	WORD		0

```
SerialSend_Instance_0(
Req TRUE := bReq TRUE,
Port 0 := uiPort 0,
Buffer 3012031068 := pBuffer 3012031068,
Length 10 := uiLength 10,
Done TRUE => bDone TRUE,
Error FALSE => bError FALSE,
Status 0 => wStatus 0 );RETURN
```

LD:



- **Precautions**

  The actual length of sent byte and baud rate are related to the task cycle. If the task cycle is shorter than the time required to send the bytes, only part of the data may be sent, because the baud rate determines the time needed to send one byte of data.

- **Status code**

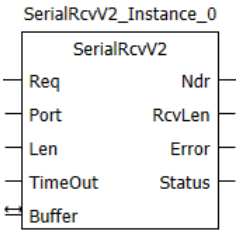
Status code	Note
0	Done
1000	Function not activated
1001	Serial port number error
1007	Sending address error
2000	Number of bytes sent error
2001	Incorrect buffer parameter
3000	Data being sent

Status code	Note
3002	Data sending failed

18.4.7.4 SerialRcvV2

This instruction receives data through serial ports by using freeport.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
SerialRcvV2	Serial port freeport reception	FB		SerialRcvV2_Instance_0(Req := , Port := , Len := , TimeOut := , Buffer := , Ndr => , RcvLen => , Error => , Status =>);

- **Related variables**

Variable

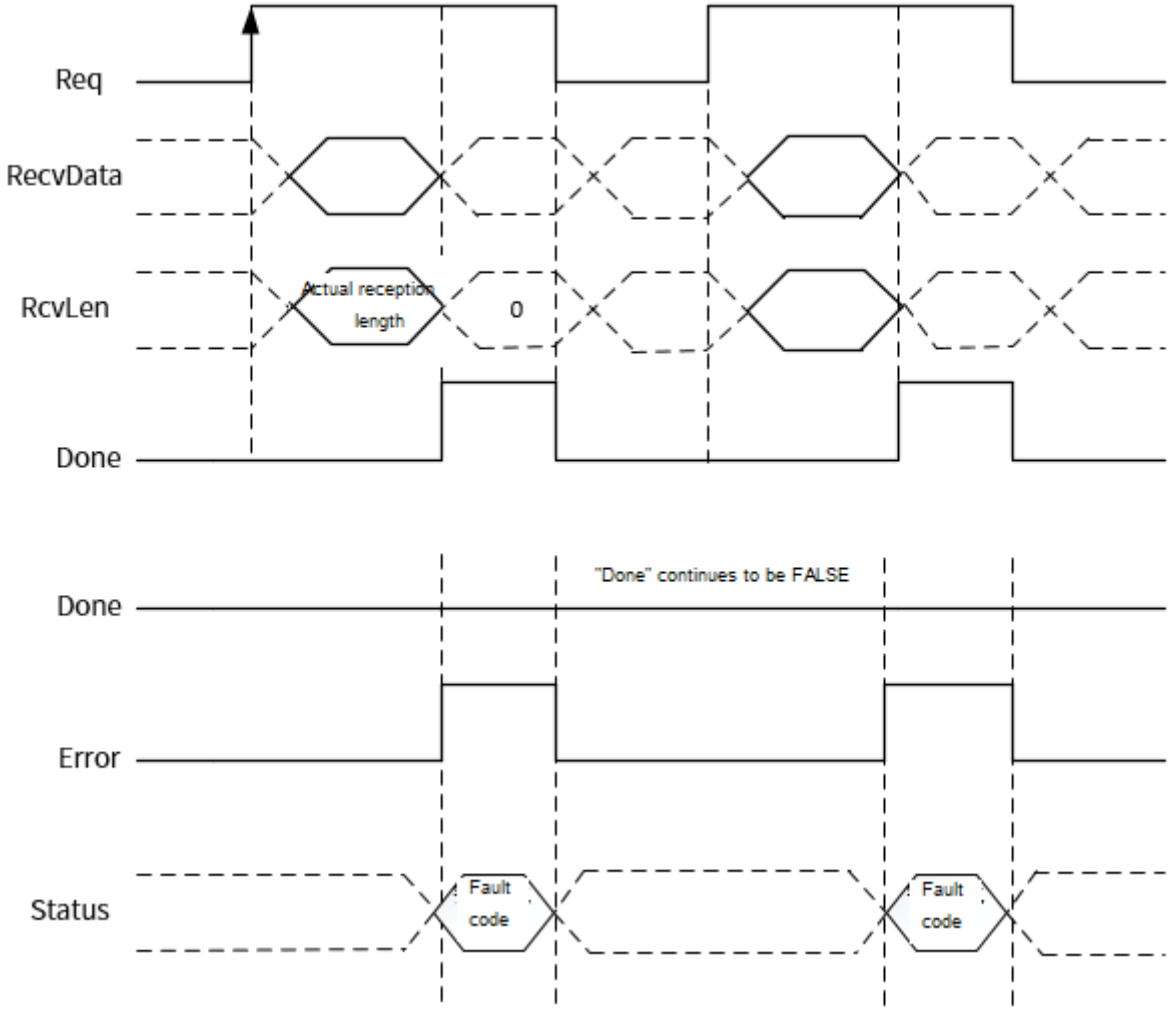
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Req	Data enable start bit	BOOL	Follows data type	FALSE	Enable start bit, active at high level.
	Port	Communication port	UINT	Follows data type	0	Serial port number configured through the software interface. 0: Built-in COM port of the PLC. 1: Expansion card A on the PLC 2: Expansion card B on the PLC
	Len	Received data length	UINT	1 to 1024	1024	When Len = 0, it indicates receiving data of indefinite length, and the maximum size of received data is determined by the Data. When Len > 0, it indicates receiving data of fixed length, with a range of 1 to 1024 bytes.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	TimeOut	Timeout period	UINT	Follows data type	0	Timeout period in ms, with a default value of 0 indicating no timeout.
Input/ Output	Buffer	Data reception buffer	ARRAY[*] OF BYTE	Follows data type	-	Data reception buffer.
Output	Ndr	Completion flag	BOOL	Follows data type	FALSE	Completion flag: FALSE: Reception not started or in progress. TRUE: Reception done.
	RcvLen	Received data length	UINT	1 to 1024	0	Actual length of received data.
	Error	Error flag	BOOL	Follows data type	FALSE	Error flag: 0: No error. 1: Reception error.
	Status	Status flag	WORD	Follow data type	1000	See Status code.

- **Function**

  This instruction receives data through serial ports by using freeports.

- **Sequence diagram**



- **Program example**

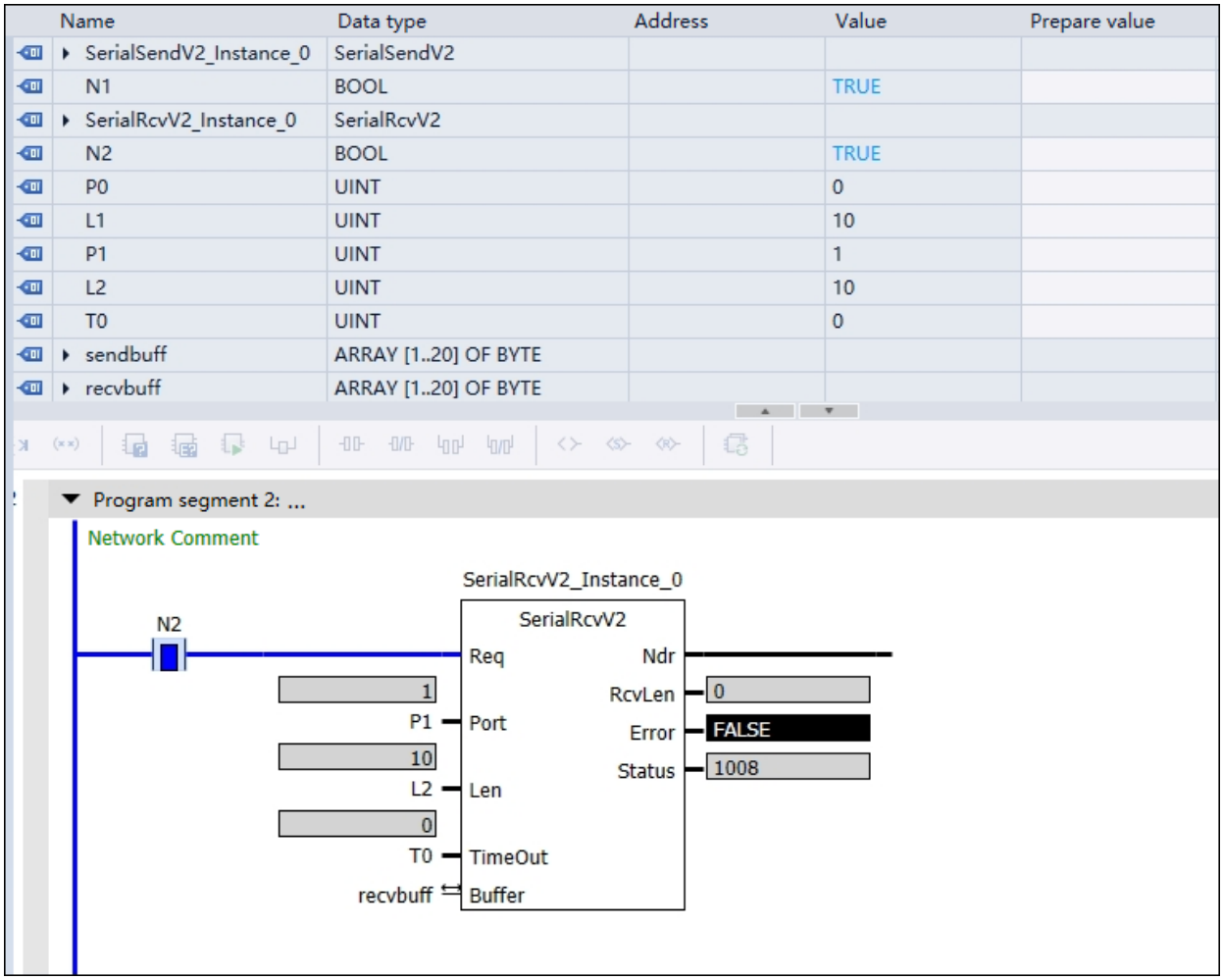
  ST:

Name	Data type	Address	Value	Prepare value
SerialRcvV2_Instance_1	SerialRcvV2			
N2	BOOL		TRUE	
P0	UINT		0	
L1	UINT		10	
P1	UINT		1	
L2	UINT		0	
T0	UINT		0	
sendbuff	ARRAY [1..20] OF BYTE			
recvbuff	ARRAY [1..20] OF BYTE			
ndr1	BOOL		FALSE	
recvlen2	UINT		0	
error2	BOOL		FALSE	
status2	WORD		1008	

CAS FOR WHI REP RET EX REG // //* ||=

```
SerialRcvV2_Instance_1(
Req  TRUE  :=N2  TRUE  ,
Port   1    := P1   1    ,
Len   20    := L2   0    ,
TimeOut   0     :=T0   0     ,
Buffer          3012295958          := recvbuff,
Ndr  FALSE  => ndr1  FALSE  ,
RcvLen   0     =>recvlen2   0     ,
Error  FALSE  =>error2  FALSE  ,
Status  1008   => status2  1008  );RETURN
```

LD:



- **Precautions**
  - The actual length of received byte and baud rate are related to the task cycle. If the task cycle is shorter than the time required to receive the bytes, only part of the data may be received, because the baud rate determines the time needed to receive one byte of data.
  - Ensure that the size of data reception area is greater than or equal to the received byte length; otherwise, there is a risk of memory access out of bounds.
  - SerialRcvV2 allows the received length to be configured, with a maximum of 1024 bytes. When set to 0, the received length is the actual length of data received. When set to a specific value, the actual received length is validated against the configured length.
- **Status code**

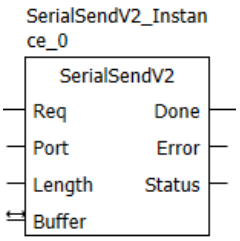
Status code	Note
0	Done
1000	Function not activated
1001	Serial port number error
1007	Reception address error
1008	No data
2000	Numbers of bytes received error
2001	Incorrect buffer parameter

Status code	Note
3000	Data being received
3001	Data reception timeout
3002	Data reception failed

18.4.7.5 SerialSendV2

This is the serial port free port data sending instruction.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
SerialSendV2	Serial port free-port data sending instruction	FB		SerialSendV2_Instance_0(Req := , Port := , Length := , Buffer := , Done => , Error => , Status =>);

- **Related variables**

Variable

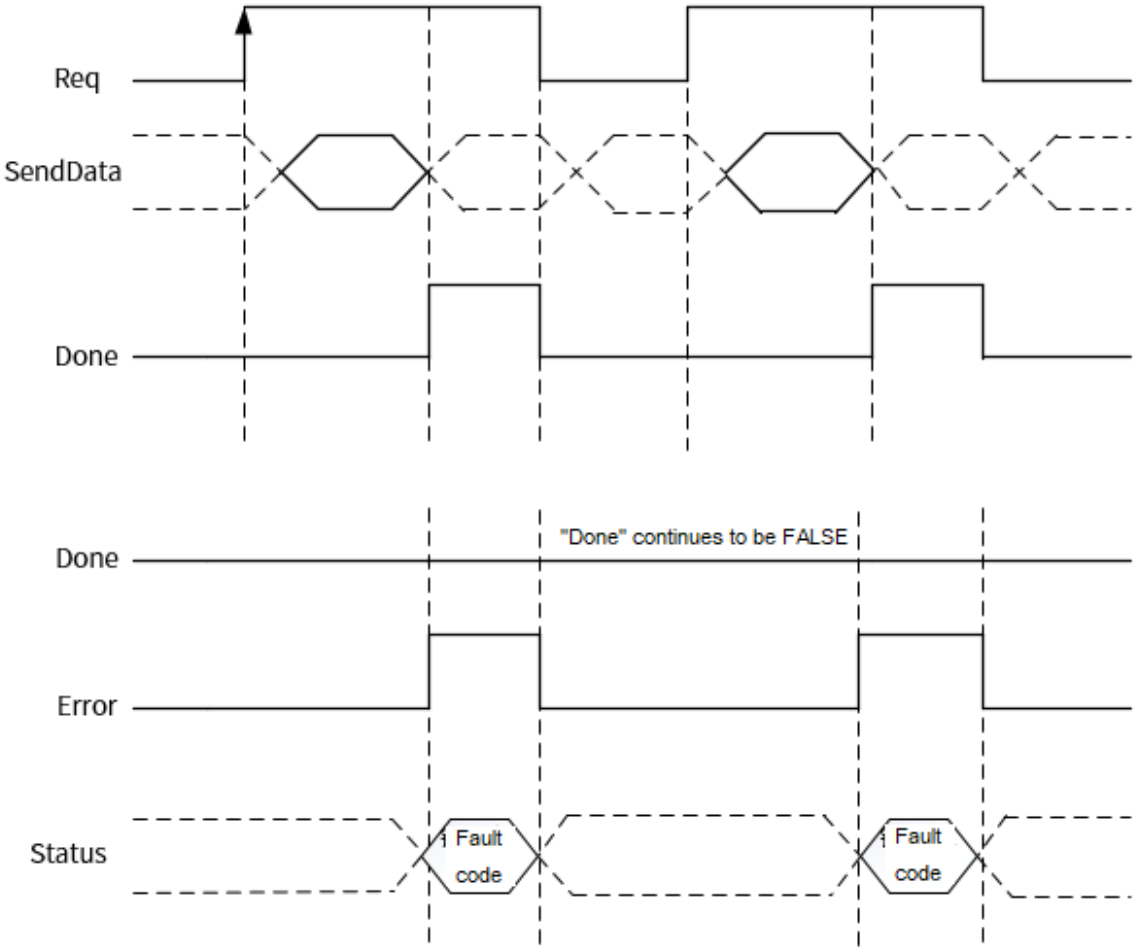
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Req	Enable start bit to send data	BOOL	Follows data type	FALSE	Enable start bit, active on rising edge.
	Port	Communication port	UINT	Follows data type	0	Serial port number configured through the software interface. 0: Built-in COM port of the PLC. 1: Expansion card A on the PLC 2: Expansion card B on the PLC
	Length	Transmit data length	UINT	1 to 1024	1024	Transmit data length
Input/ Output	Buffer	Sending data buffer	ARRAY[*] OF BYTE	Follows data type	-	Sending data buffer.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	Done	Completion flag	BOOL	Follows data type	FALSE	Completion flag: FALSE: Sending not started or in progress. TRUE: Sending done.
	Error	Error flag	BOOL	Follows data type	FALSE	Error flag: 0: No error. 1: Sending error.
	Status	Status flag	WORD	Follow data type	1000	See Status code.

- **Function**

  This is the serial port free port data sending instruction.

- **Sequence diagram**



- **Program example**

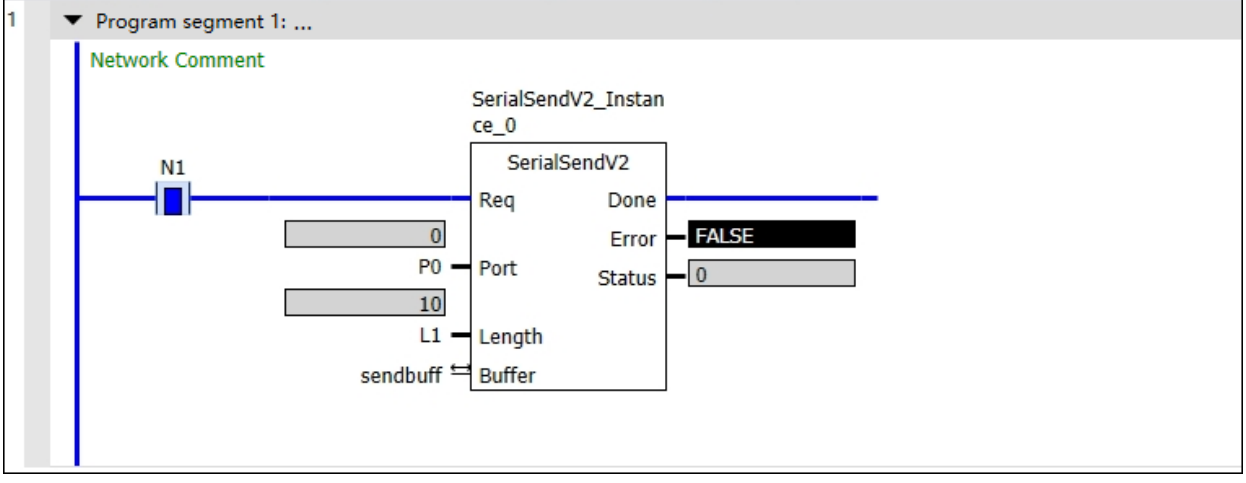
  ST:

Name	Data type	Address	Value	Prepare value
SerialSendV2_Instance_1	SerialSendV2			
N1	BOOL		TRUE	
SerialRcvV2_Instance_1	SerialRcvV2			
N2	BOOL		FALSE	
P0	UINT		0	
L1	UINT		10	
P1	UINT		1	
L2	UINT		0	
T0	UINT		0	
sendbuff	ARRAY [1..20] OF BYTE			
recvbuff	ARRAY [1..20] OF BYTE			
done1	BOOL		TRUE	
error1	BOOL		FALSE	
status1	WORD		0	

```
SerialSendV2_Instance_1(
Req    TRUE    := N1   TRUE   ,
Port    0      := P0    0     ,
Length   10     := L1    10    ,
Buffer          3012295938          := sendbuff,
Done   TRUE    => done1  TRUE   ,
Error  FALSE   => error1 FALSE  ,
Status   0     => status1   0    );
```

LD:

Name	Data type	Address	Value	Prepare value
SerialSendV2_Instance_0	SerialSendV2			
N1	BOOL		TRUE	
SerialRcvV2_Instance_0	SerialRcvV2			
N2	BOOL		FALSE	
P0	UINT		0	
L1	UINT		10	
P1	UINT		1	
L2	UINT		0	
T0	UINT		0	
sendbuff	ARRAY [1..20] OF BYTE			
recvbuff	ARRAY [1..20] OF BYTE			



- **Precautions**

The actual length of sent byte and baud rate are related to the task cycle. If the task cycle is shorter than the time required to send the bytes, only part of the data may be sent, because the baud rate determines the time needed to send one byte of data.

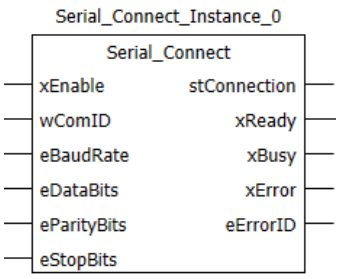
- **Status code**

Status code	Note
0	Done
1000	Function not activated
1001	Serial port number error
1007	Sending address error
2000	Number of bytes sent error
2001	Incorrect buffer parameter
3000	Data being sent
3002	Data sending failed

18.4.7.6 Serial_Connect

This instruction establishes a connection via the serial port using a free protocol.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
Serial_Connect	Create a serial port connection	FB		Serial_Connect_Instance_0(xEnable := , wComID := , eBaudRate := , eDataBits := , eParityBits := , eStopBits := , stConnection => , xReady => , xBusy => , xError => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Function block enable	BOOL	Follows data type	FALSE	Level triggered

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	wComID	Serial port number	WORD	Follow data type	0	COM port number. The value of wComID identifies serial ports, whose available number varies by product. For details, see relevant product guides. Take the EVO500 series as example: 0: Onboard serial port of the PLC 1: Expansion card A on the PLC 2: Expansion card B on the PLC
	eBaudRate	Baud rate	SERIAL_BAUD	Follows data type	SERIAL_BAUD.B9600	Baud rate, unit: bps SERIAL_BAUD.B4800: 4800bps. SERIAL_BAUD.B9600: 9600bps. SERIAL_BAUD.B19200: 19200bps. SERIAL_BAUD.B38400: 38400bps. SERIAL_BAUD.B57600: 57600bps. SERIAL_BAUD.B115200: 115200bps.
	eDataBits	Data bit	SERIAL_DATABITS	Follows data type	SERIAL_DATABITS.BIT8	Data bit SERIAL_DATABITS.BIT7: 7 data bits SERIAL_DATABITS.BIT8: 8 data bits

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	eParityBits	Parity bit	SERIAL_PARITYBITS	Follows data type	SERIAL_PARITYBITS.NONE	Parity bit SERIAL_PARITYBITS.NONE: No parity SERIAL_PARITYBITS.ODD: Odd parity SERIAL_PARITYBITS.EVEN: Even parity
	eStopBits	Stop bit	SERIAL_STOPBITS	Follows data type	SERIAL_STOPBITS.BIT1	Stop bit SERIAL_STOPBITS.BIT1: 1 data bit SERIAL_STOPBITS.BIT2: 2 data bits
Output	stConnection	Connection handle	CBus_Connection	Follows data type	-	Connection handle.
	xReady	Instruction status	BOOL	Follows data type	FALSE	Indicates the function block is active
	xBusy	Instruction execution flag	BOOL	Follows data type	FALSE	Function block execution flag
	xError	Error flag	BOOL	Follows data type	FALSE	Indicates the function block is faulty
	eErrorID	Fault code	CBUS_ERROR	Follows data type	0	Fault code of function block. For details, see 18.4.5.6 Fault Codes.

- **Function**

  You can open the serial port to modify the baud rate, parity and other parameters during operation.

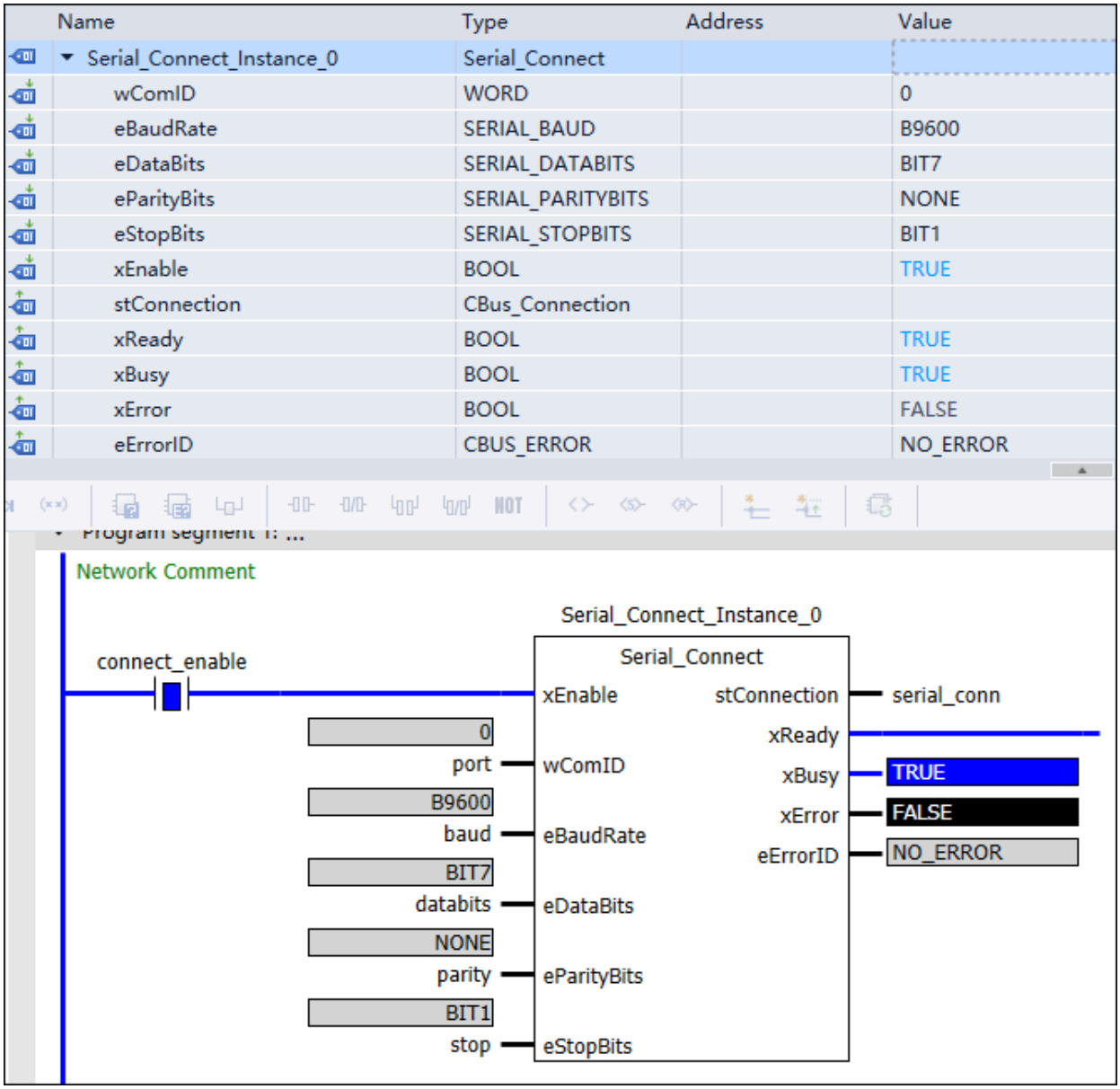
- **Program example**

  ST:

Name	Type	Address	Value
Serial_Connect_Instance_0	Serial_Connect		
wComID	WORD		0
eBaudRate	SERIAL_BAUD		B4800
eDataBits	SERIAL_DATABITS		BIT7
eParityBits	SERIAL_PARITYBITS		NONE
eStopBits	SERIAL_STOPBITS		BIT1
xEnable	BOOL		TRUE
stConnection	CBus_Connection		
xReady	BOOL		TRUE
xBusy	BOOL		TRUE
xError	BOOL		FALSE

```
                                    eErrorID NO_ERROR => area_errorid NO_ERROR );
Serial_Connect_Instance_0(xEnable TRUE := connect_enable TRUE ,
                          wComID 0 := port 0 ,
                          eBaudRate B4800 := baud B4800 ,
                          eDataBits BIT7 := databits BIT7 ,
                          eParityBits NONE := parity NONE ,
                          eStopBits BIT1 := stop BIT1 ,
                          stConnection => serial_conn,
                          xReady TRUE => connect_ready TRUE ,
                          xBusy TRUE => connect_busy TRUE ,
                          xError FALSE => connect_error FALSE ,
                          eErrorID NO_ERROR => connect_errorid NO_ERROR );
```

LD:

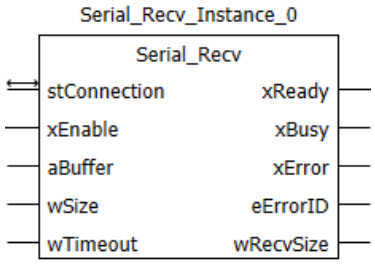


- **Precautions**
  - The connection handle of Serial_Connect handle can only be used by one instruction (Serial_Recv, Serial_Send, Serial_Recv_Ch, or Serial_Recv_By_Idle_Time) at the same time.
  - Ensure that multiple instructions with the same connection handle are used in the same user task.
  - Online parameter modifications will not take effect immediately, requiring the instruction to be re-enabled.
  - The value of wComID identifies serial ports, whose available number varies by product. For details, see relevant product guides. For example, on the EVO500 series, 0 represents the PLC onboard serial port, while 1 and 2 represent expansion cards A and B respectively.
  - Check the COM ports for locally or remotely mounted GL20_2S485/GL20_2SCOM modules as follows: In the project tree, double-click GL20_2S485 or GL20_2SCOM to open the configuration interface, and then check the COM number in the "General configuration" tab.

18.4.7.7 Serial_Recv

This instruction receives data through serial ports by using free protocols. It receives fixed-byte data, and terminate reception once the length of received bytes reach the preset value.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
Serial_Recv	Receive serial port data	FB		Serial_Recv_Instance_0(st Connection := , xEnable := , aBuffer := , wSize := , wTimeout := , xReady => , xBusy => , xError => , eErrorID => , wRecvSize =>);

- **Related variables**

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input/Output	stConnection	Connection handle	CBus_Connection	Follows data type	-	Connection handle.
Input	xEnable	Function block enable	BOOL	TRUE, FALSE	FALSE	Level triggered
	aBuffer	Receive buffer	ANY	Follows data type	-	Serial port reception buffer
	wSize	Number of bytes to receive	WORD	Follow data type	0	Number of data bytes to receive from the serial port
	wTimeout	Timeout interval (ms)	WORD	Follow data type	0	Timeout interval of serial port data
Output	xReady	Instruction status	BOOL	TRUE, FALSE	FALSE	Indicates the function block is active
	xBusy	Instruction execution flag	BOOL	TRUE, FALSE	FALSE	Function block execution flag
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Indicates the function block is faulty

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	eErrorID	Fault code	CBUS_ERROR	Follows data type	-	Fault code of function block. For details, see 1 8.4.5.6 Fault Codes.
	wRecvSize	Number of received bytes	WORD	Follow data type	0	Number of data bytes received from the serial port

- **Function**

  This instruction receives and stores serial port data in reception buffer. The serial port stays in receiving state continuously. When wRecvSize is set to wSize, the data reception completes. xReady will be set to one cycle and reset in the next cycle. wRecvSize will be cleared to zero, and then a new reception begins.

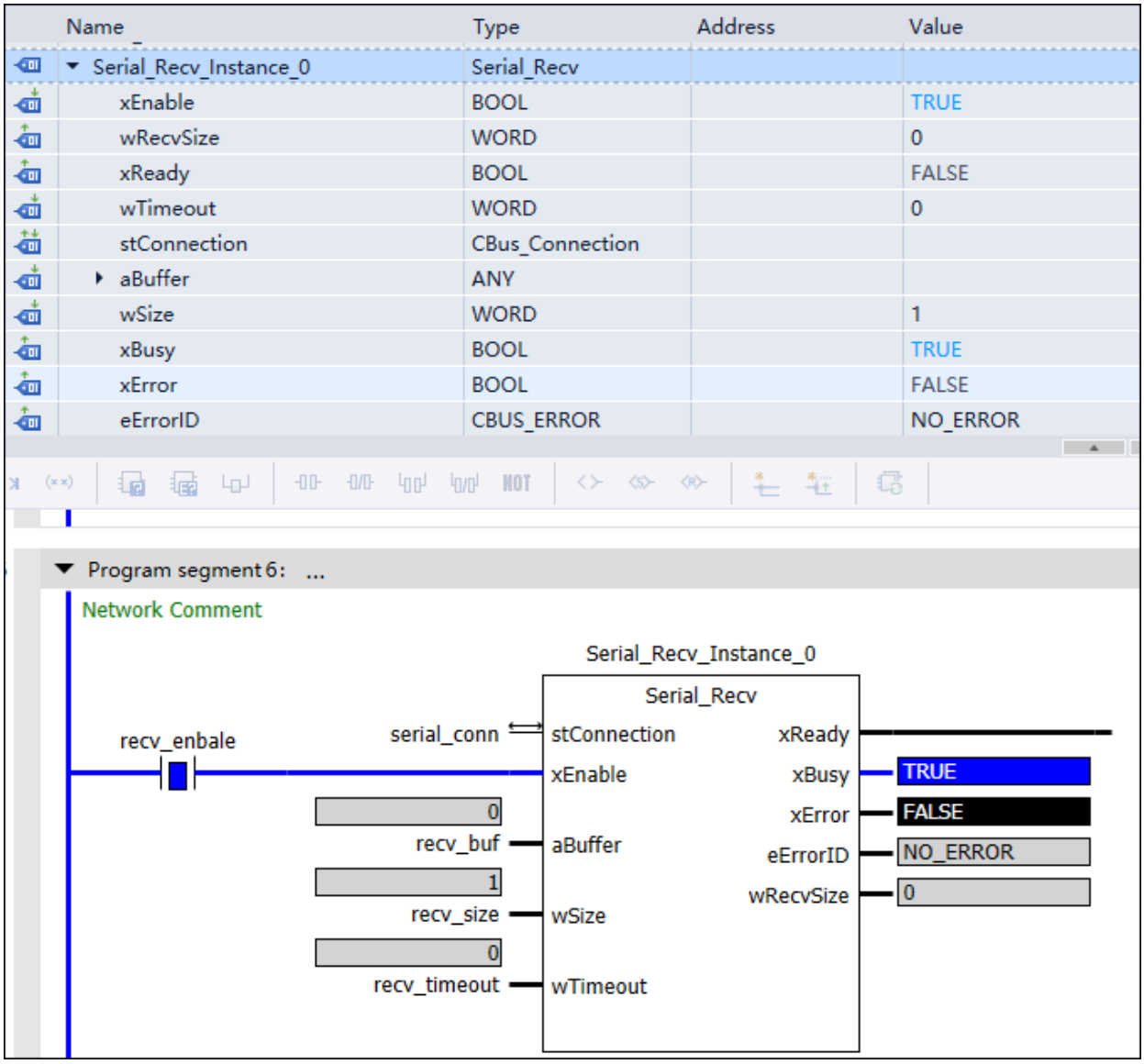
- **Program example**

  ST:

Name	Type	Address	Value
Serial_Recv_Instance_0	Serial_Recv		
xEnable	BOOL		TRUE
wRecvSize	WORD		0
xReady	BOOL		FALSE
wTimeout	WORD		0
stConnection	CBus_Connection		
aBuffer	ANY		
wSize	WORD		1
xBusy	BOOL		TRUE
xError	BOOL		FALSE
eErrorID	CBUS_ERROR		NO_ERROR

```
Serial_Recv_Instance_0(stConnection := serial_conn,
                       xEnable TRUE  := recv_enbale TRUE ,
                       aBuffer := recv_buf  0 ,
                       wSize  1  := recv_size  1 ,
                       wTimeout  0  := recv_timeout  0 ,
                       xReady FALSE  => recv_ready1 FALSE ,
                       xBusy TRUE  => recv_busy1 TRUE ,
                       xError FALSE  => recv_error1 FALSE ,
                       eErrorID NO_ERROR  => recv_errorid1 NO_ERROR ,
                       wRecvSize  0  => recv_rsize1  0 );
```

  LD:



- **Precautions**
  - It must be used with the Serial_Connect instruction.
  - The reception does not stop by timeout when wTimeout is set to 0. If a timeout interval is set, timeout timing starts on reception failure or idle serial bus.
    - If there is still no data when the timeout interval expires, an error will be reported for receiving timeout.
    - If timeout not reached and serial data is received, reset timeout.
  - When wRecvSize is set to wSize, the data reception is complete, xReady will be set to one cycle and reset in the next cycle, and wRecvSize will be cleared.
  - Data types loaded by aBuffer:
    - Supported types:
      - One-dimensional array.
      - Data types of arrays/variables: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, and REAL.
    - Unsupported types:
      - Multidimensional arrays.
      - 64-bit data types: LWORD, LINT, ULINT, and LREAL.

- Special types: STRING, WSTRING, TIME, TIME_OF_DAY(TOD), DATE, DATE_AND_TIME(DT), POINTER, and STRUCT.

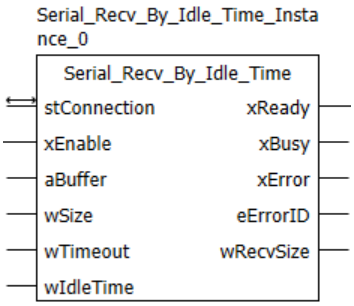
◦ aBuffer cannot be directly set to I/Q/M direct variables. It must be mapped to the above supported variables or array data via AT mapping.

◦ Online parameter modifications do not take effect immediately. It is necessary to re-enable the instruction.

18.4.7.8 Serial_Recv_By_Idle_Time

This instruction receives data through serial ports by using free protocols, and determines reception completion based on bus idle time.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
Serial_Recv_By_Idle_Time	Receive serial port data during bus idle time	FB		Serial_Recv_By_Idle_Time_Instance_0(stConnection := , xEnable := , aBuffer := , wSize := , wTimeout := , wIdleTime := , xReady => , xBusy => , xError => , eErrorID => , wRecvSize =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input/ Output	stConnection	Connection handle	CBus_Connection	Follows data type	-	Connection handle.
Input	xEnable	Function block enable	BOOL	TRUE, FALSE	FALSE	Level triggered
	aBuffer	Receive buffer	ANY	Follows data type	-	Serial port reception buffer
	wSize	Number of bytes to receive	WORD	Follow data type	0	Number of data bytes to receive from the serial port
	wTimeout	Timeout interval (ms)	WORD	Follow data type	0	Timeout interval of serial port data

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	wIdleTime	Bus idle time (ms)	WORD	Follow data type	0	Indicates the serial port bus idle time (ms)
Output	xReady	Instruction status	BOOL	TRUE, FALSE	FALSE	Indicates the function block is active
	xBusy	Instruction execution flag	BOOL	TRUE, FALSE	FALSE	Function block execution flag
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Indicates the function block is faulty
	eErrorID	Fault code	CBUS_ERROR	Follows data type	0	Fault code of function block. For details, see 18.4.5.6 Fault Codes.
	wRecvSize	Number of received bytes	WORD	Follow data type	0	Indicates the number of data bytes already received from the serial port

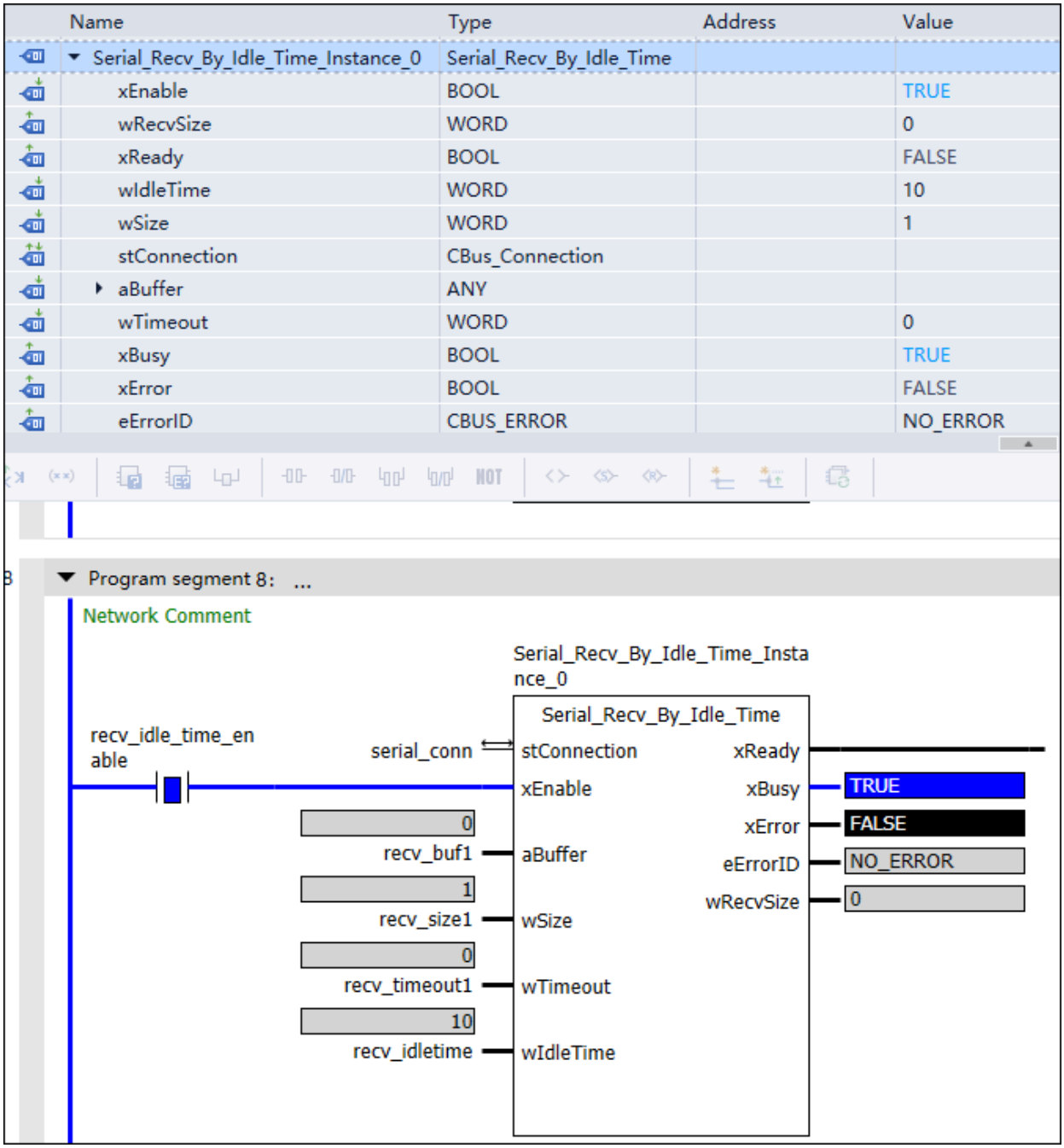
- **Function**
  After data reception, if no new data is received beyond wIdleTime (ms), xReady is set to indicate that the reception is complete and the next reception cycle begins.
- **Program example**

  ST:

Name	Type	Address	Value
Serial_Recv_By_Idle_Time_Instance_0	Serial_Recv_By_Idle_Time		
xEnable	BOOL		TRUE
wRecvSize	WORD		0
xReady	BOOL		FALSE
wIdleTime	WORD		2
wSize	WORD		1
stConnection	CBus_Connection		
aBuffer	ANY		
wTimeout	WORD		0
xBusy	BOOL		TRUE
xError	BOOL		FALSE
eErrorID	CBUS_ERROR		NO_ERROR

```
Serial_Recv_By_Idle_Time_Instance_0(stConnection := serial_conn,
                                    xEnable TRUE  := recv_idle_time_enable TRUE ,
                                    aBuffer := recv_buf1  0 ,
                                    wSize  1  := recv_size1  1 ,
                                    wTimeout  0  := recv_timeout1  0 ,
                                    wIdleTime  2  := recv_idletime  2 ,
                                    xReady FALSE  => recv_ready3 FALSE ,
                                    xBusy TRUE  => recv_busy3 TRUE ,
                                    xError FALSE  => recv_error3 FALSE ,
                                    eErrorID NO_ERROR  => recv_errorid3 NO_ERROR ,
                                    wRecvSize  0  => recv_rsize3  0 );
```

LD:



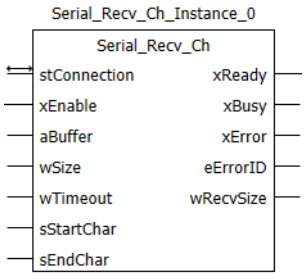
- **Precautions**
  - It must be used with the Serial_Connect instruction.
  - The reception does not stop by timeout when wTimeout is set to 0. If a timeout interval is set, timeout timing starts on reception failure or idle serial bus.
    - If there is still no data when the timeout interval expires, an error will be reported for receiving timeout.
    - If timeout not reached and serial data is received, reset timeout.
  - When wRecvSize is set to wSize, the data reception is complete, xReady will be set to one cycle and reset in the next cycle, and wRecvSize will be cleared.
  - Data types loaded by aBuffer:
    - Supported types:
      - One-dimensional array.
      - Data types of arrays/variables: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, and REAL.

- Unsupported types:
  - Multidimensional arrays.
  - 64-bit data types: LWORD, LINT, ULINT, and LREAL.
  - Special types: STRING, WSTRING, TIME, TIME_OF_DAY(TOD), DATE, DATE_AND_TIME(DT), POINTER, and STRUCT.
- aBuffer cannot be directly set to I/Q/M direct variables. It must be mapped to the above supported variables or array data via AT mapping.
- Online parameter modifications do not take effect immediately. It is necessary to re-enable the instruction.
- The task cycle time must be less than or equal to wIdleTime. Otherwise, the instruction will not work effectively.
- The wIdleTime value must be estimated based on the baud rate and the amount of data sent. For example, at a baud rate of 115200, 80 bytes take approximately 7 to 8 ms to complete reception. If the communication partner sends data every 10 ms, set wIdleTime to less than 2 ms. Otherwise, if the next data frame arrives before a sufficiently long idle time (i.e., within 10 ms), the received data may be incorrect.

18.4.7.9 Serial_Recv_Ch

This instruction receives data through serial ports by using free protocols. The reception starts only when a start character is detected in the received data, and stops when an end character is detected.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
Serial_Recv_Ch	Receive serial port data with start/end character	FB		Serial_Recv_Ch_Instance_0(stConnection := , xEnable := , aBuffer := , wSize := , wTimeout := , sStartChar := , sEndChar := , xReady => , xBusy => , xError => , eErrorID => , wRecvSize =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input/ Output	stConnection	Connection handle	CBus_Connection	Follows data type	-	Connection handle.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Function block enable	BOOL	TRUE, FALSE	FALSE	Level triggered
	aBuffer	Receive buffer	ANY	Follows data type	-	Serial port reception buffer
	wSize	Number of bytes to receive	WORD	Follow data type	0	Number of data bytes to receive from the serial port
	wTimeout	Timeout interval (ms)	WORD	Follow data type	0	Timeout interval of serial port data
	sStartChar	Start character	STRING	Follows data type	-	Start character
	sEndChar	End character	STRING	Follows data type	-	End character
Output	xReady	Instruction status	BOOL	TRUE, FALSE	FALSE	Indicates the function block is active
	xBusy	Instruction execution flag	BOOL	TRUE, FALSE	FALSE	Function block execution flag
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Indicates the function block is faulty
	eErrorID	Fault code	CBUS_ERROR	Follows data type	0	Fault code of function block. For details, see 18.4.5.6 Fault Codes.
	wRecvSize	Number of received bytes	WORD	Follow data type	0	Number of data bytes received from the serial port

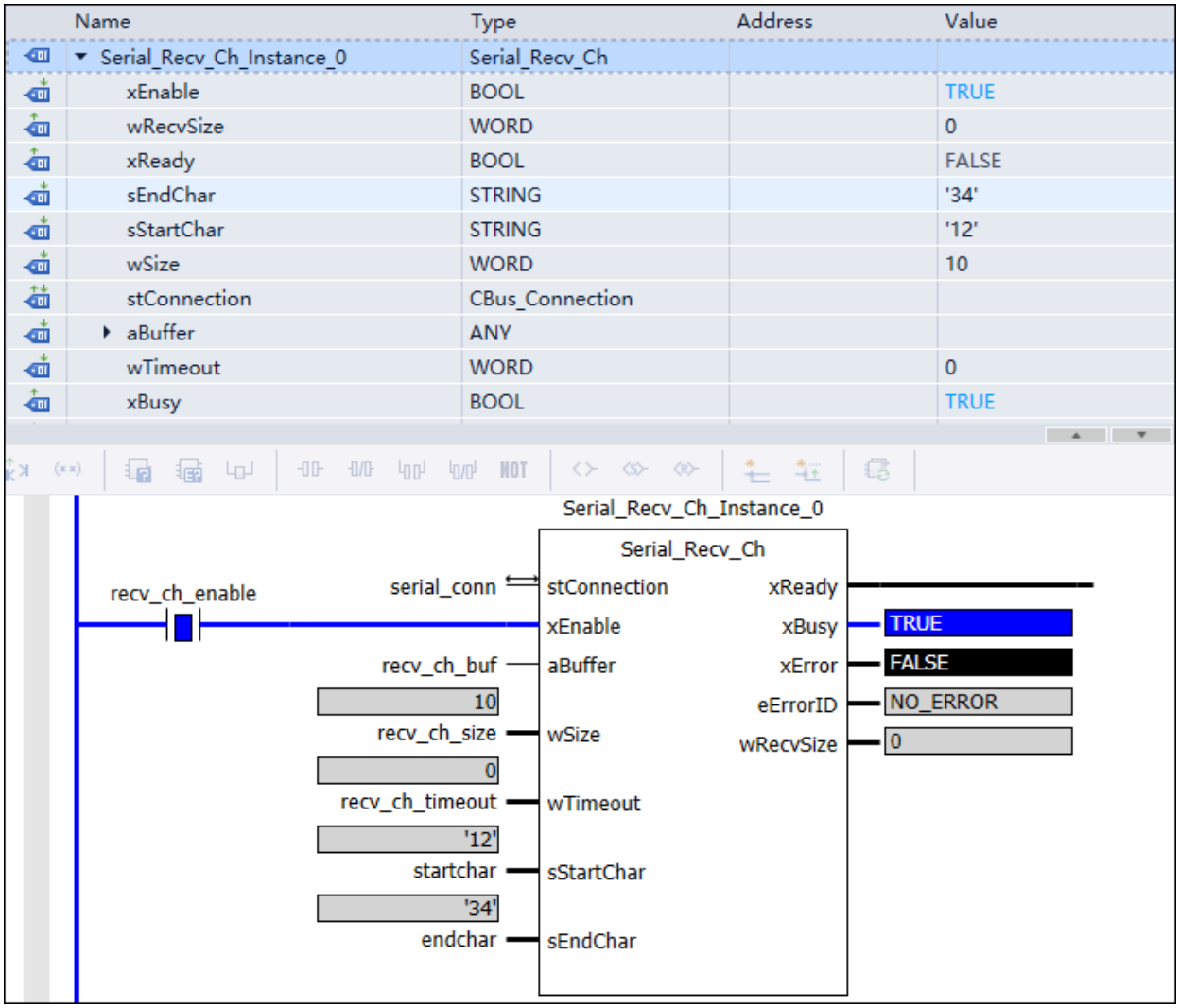
- **Function**
  This instruction receives and stores serial port data in reception buffer. The serial port stays in receiving state continuously. When data starts with the start character and ends with the end character, the data reception is complete. xReady will be set to one cycle and reset in the next cycle. wRecvSize will be cleared to zero, and then a new reception begins.
- **Program example**

  ST:

Name	Type	Address	Value
Serial_Recv_Ch_Instance_0	Serial_Recv_Ch		
xEnable	BOOL		TRUE
wRecvSize	WORD		0
xReady	BOOL		FALSE
sEndChar	STRING		'34'
sStartChar	STRING		'12'
wSize	WORD		10
stConnection	CBus_Connection		
aBuffer	ANY		
wTimeout	WORD		0
xBusy	BOOL		TRUE
xError	BOOL		FALSE
eErrorID	CBUS_ERROR		NO_ERROR

```
Serial_Recv_Ch_Instance_0(stConnection := serial_conn,
                          xEnable TRUE  := recv_ch_enable TRUE ,
                          aBuffer := recv_ch_buf,
                          wSize    10    := recv_ch_size    10    ,
                          wTimeout  0    := recv_ch_timeout  0    ,
                          sStartChar  '12'  := startchar  '12'  ,
                          sEndChar  '34'  := endchar  '34'  ,
                          xReady FALSE => recv_ready2 FALSE ,
                          xBusy TRUE => recv_busy2 TRUE ,
                          xError FALSE => recv_error2 FALSE ,
                          eErrorID NO_ERROR => recv_errorid2 NO_ERROR ,
                          wRecvSize  0    => recv_rsize2  0    );
```

LD:



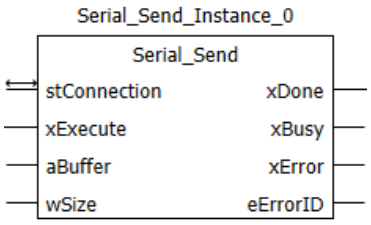
- **Precautions**
  - It must be used with the Serial_Connect instruction.
  - The reception does not stop by timeout when wTimeout is set to 0. If a timeout interval is set, timeout timing starts on reception failure or idle serial bus.
    - If there is still no data when the timeout interval expires, an error will be reported for receiving timeout.
    - If timeout not reached and serial data is received, reset timeout.
  - When wRecvSize is set to wSize, the data reception is complete, xReady will be set to one cycle and reset in the next cycle, and wRecvSize will be cleared.
  - Data types loaded by aBuffer:
    - Supported types:
      - One-dimensional array.
      - Data types of arrays/variables: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, and REAL.
    - Unsupported types:
      - Multidimensional arrays.
      - 64-bit data types: LWORD, LINT, ULINT, and LREAL.
      - Special types: STRING, WSTRING, TIME, TIME_OF_DAY(TOD), DATE, DATE_AND_TIME(DT), POINTER, and STRUCT.

- aBuffer cannot be directly set to I/Q/M direct variables. It must be mapped to the above supported variables or array data via AT mapping.
- Online parameter modifications do not take effect immediately. It is necessary to re-enable the instruction.

18.4.7.10 Serial_Send

This instruction send commands via serial port free protocol.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
Serial_Send	Send serial port data	FB		Serial_Send_Instance_0(stConnection := , xExecute := , aBuffer := , wSize := , xDone => , xBusy => , xError => , eErrorID =>);

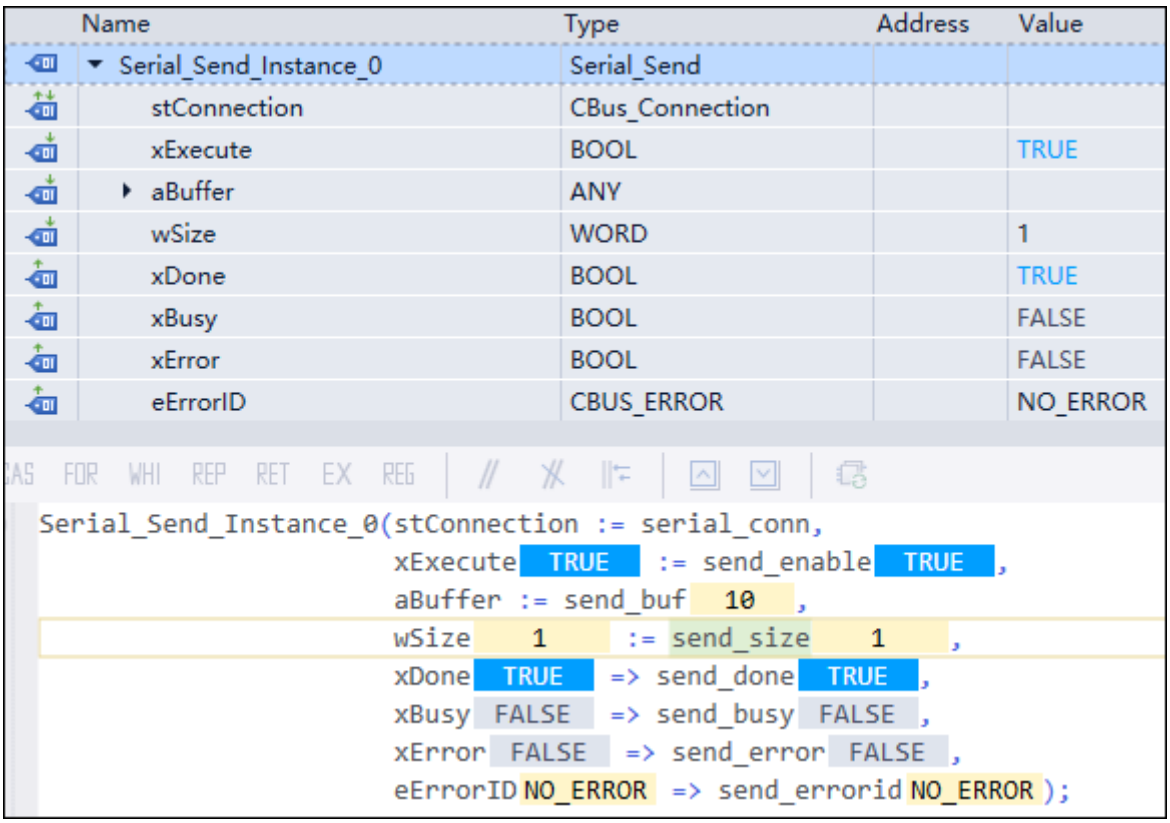
- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input/ Output	stConnection	Connection handle	CBus_Connection	Follows data type	-	Connection handle.
Input	xExecute	Function block enable	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	aBuffer	Send buffer	ANY	Follows data type	-	Serial port transmission buffer
	wSize	Number of bytes to transmit	WORD	Follow data type	0	Number of bytes to transmit by serial port
Output	xDone	Done flag	BOOL	TRUE, FALSE	FALSE	Function block execution completion flag
	xBusy	Instruction execution flag	BOOL	TRUE, FALSE	FALSE	Function block execution flag
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Indicates the function block is faulty

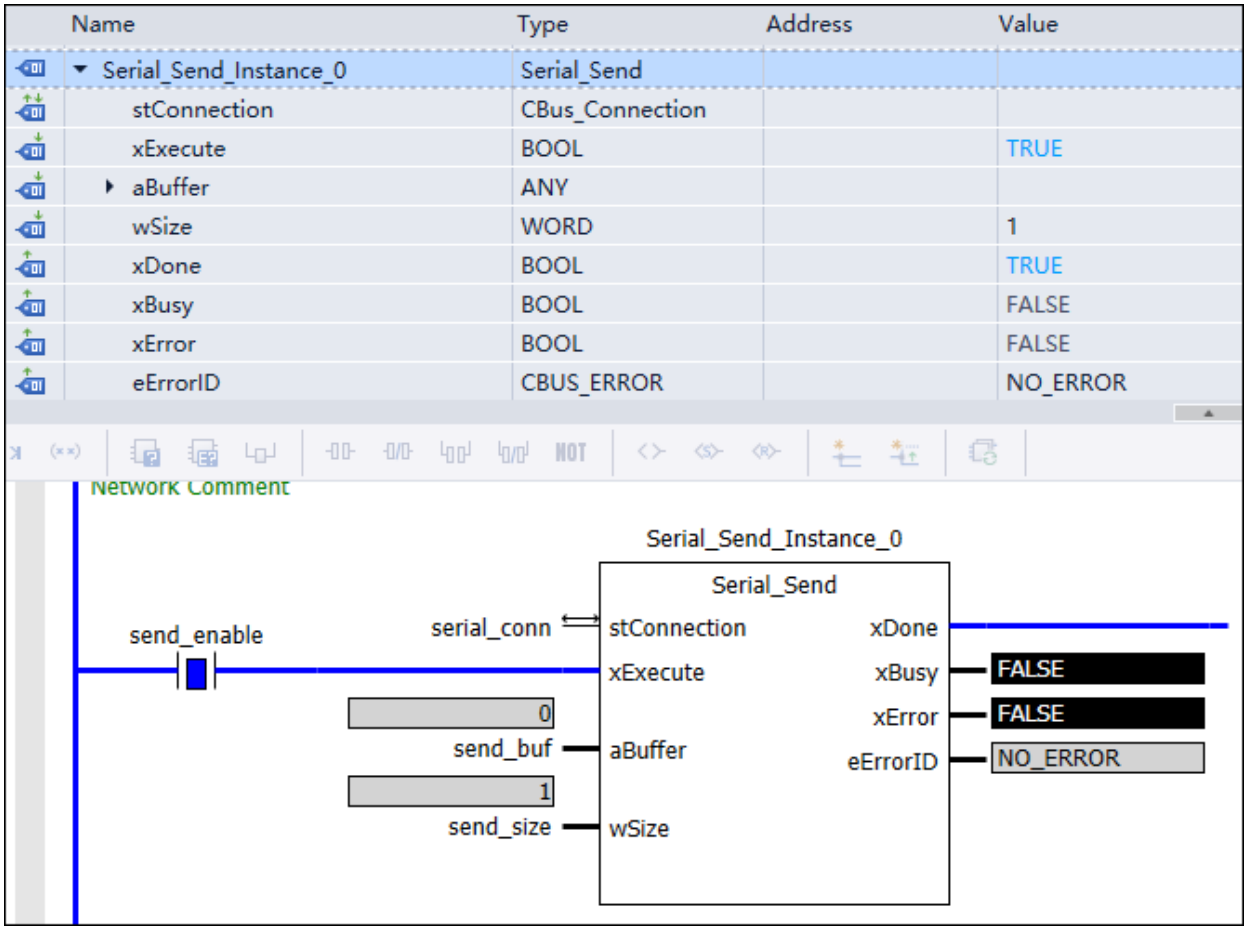
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	eErrorID	Fault code	CBUS_ERROR	Follows data type	0	Fault code of function block. For details, see 18.4.5.6 Fault Codes.

- **Function**
  This instruction starts data transmission to send data in the transmission buffer by using the serial port bus.
- **Program example**

  ST:



  LD:



- **Precautions**
  - It must be used with the Serial_Connect instruction.
  - Data types loaded by aBuffer:
    - Supported types:
      - One-dimensional array.
      - Data types of arrays/variables: BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, and REAL.
    - Unsupported types:
      - Multidimensional arrays.
      - 64-bit data types: LWORD, LINT, ULINT, and LREAL.
      - Special types: STRING, WSTRING, TIME, TIME_OF_DAY(TOD), DATE, DATE_AND_TIME(DT), POINTER, and STRUCT.
  - aBuffer cannot be directly set to I/Q/M direct variables. It must be mapped to the above supported variables or array data via AT mapping.
  - Online parameter modifications do not take effect immediately. It is necessary to re-enable the instruction.
  - The send and recv instructions (Serial_Recv, Serial_Recv_Ch, and Serial_Recv_By_Idle_Time) can be used at the same time. However, data corruption may occur on the half-duplex RS485 interface, so it is not recommended to use two types of instructions simultaneously in RS485 mode.
    Concurrent use is allowed for RS232 interface.

18.4.8 Free Network Port

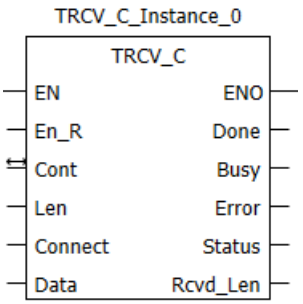
18.4.8.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.4.8.2 TRCV_C	FB	Network port freeport reception	NetFreePort	Applicable to library version 1.0.0.0
18.4.8.3 TSEND_C	FB	Ethernet free port data sending instruction		Applicable to library version 1.0.0.0
18.4.8.4 TRCV_CV2	FB	Network port freeport reception		Library version 1.1.0.0 and later
18.4.8.5 TSEND_CV2	FB	Ethernet free port data sending instruction		Library version 1.1.0.0 and later

18.4.8.2 TRCV_C

This instruction receives data through network ports by using freeports.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
TRCV_C	Network port freeport reception	FB		TRCV_C_Instance_0(En_R := , Cont := , Len := , Connect := , Data := , Done => , Busy =>, Error => , Status => , Rcvd_Len =>);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	En_R	Enable start bit to receive data	BOOL	Follows data type	FALSE	Enable start bit, active at high level.
Input/ Output	Cont	Control communication connection	BOOL	Follows data type	FALSE	Control communication connection: FALSE: Disconnect the communication connection. TRUE: Establish the communication connection and maintain it after data is received.
Input	Len	Received data length	UDINT	Follows data type	1024	The maximum length of data to be received (1 to 1024 bytes).
	Connect	Connection description structure	Tcont_Ip_v4	Follows data type	-	For connection description structure, see details in the Function description below.
	Data	Data reception buffer	POINTER TO BYTE	Follows data type	-	Pointer to the received data buffer.
Output	Done	Completion flag	BOOL	Follows data type	FALSE	Completion flag: FALSE: Sending not started or in progress. TRUE: Sending done.
	Busy	Reception status	BOOL	Follows data type	FALSE	Status parameter: FALSE: Data reception done or not started. TRUE: Data reception not done.
	Error	Error flag	BOOL	Follows data type	FALSE	Error flag: FALSE: No error TRUE: Reception error.
	Status	Status flag	WORD	Follow data type	1000	See Status code.
	Rcvd_Len	Received data length	UDINT	Follows data type	0	Actual amount of received data.

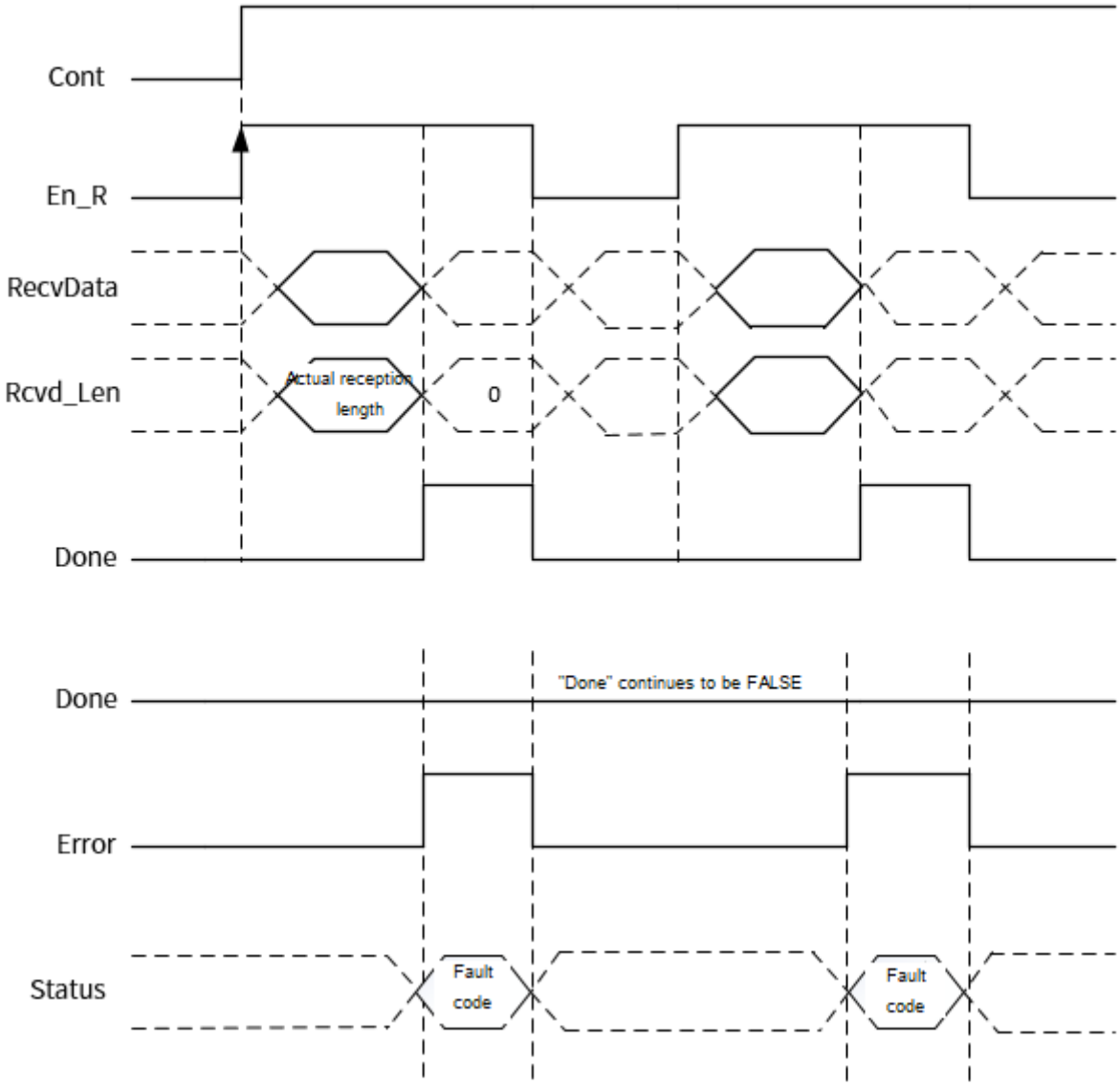
- **Function**

  When using the receive function block, set instruction reception frequency over 2x sending frequency to ensure data consistency per cycle; otherwise, reception delay occurs.

The Tcont_Ip_v4 structure is shown in the table below.

Parameter	Type	Note
ID	WORD	Module ID: One connection corresponds to one ID, and each Module ID value is unique (value range: 1 to 4095).
ConnectionType	BYTE	Connection type: 11: TCP 19: UDP
ActiveEstablished	BOOL	Identifier of connection establishment type: FALSE: Passive connection TRUE: Active connection
RemoteAddress	ARRAY[0..3] OF BYTE	It is the IP of remote connection partner. For example, 192.168.0.1, where addr[1] = 192, addr[2] = 168, addr[3] = 0, addr[4] = 1. Broadcasting is not supported. IP is invalid for TCP passive connection.
RemotePort	UINT	It is the port number of remote connection partner (range: 1 to 49151) with a default value of 2000, recommended to use port numbers after 2000 (except for dedicated port numbers).
LocalPort	UINT	It is the local port number (range: 1 to 49151) with a default value of 2000, recommended to use port numbers after 2000 (except for dedicated port numbers).

- **Sequence diagram**



- **Program example**

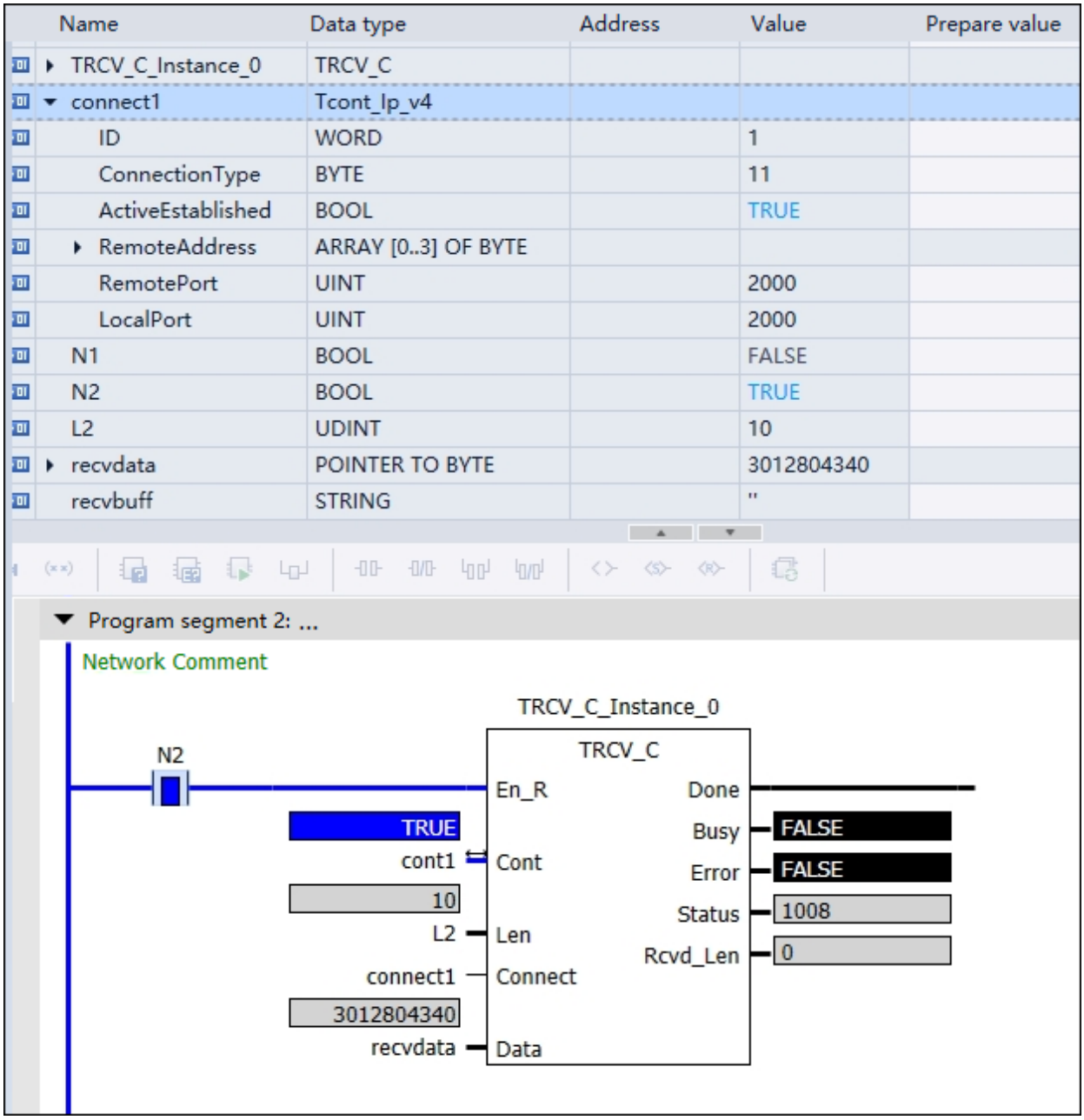
  This section takes TCP connection as example.

  ST:

Name	Data type	Address	Value	Prepare value
sedbuff	ARRAY [1..20] OF BYTE			
N1	BOOL		TRUE	
N2	BOOL		TRUE	
cont1	BOOL		TRUE	
L0	UDINT		10	
connect1	Tcont_Ip_v4			
cont2	BOOL		TRUE	
L1	UDINT		20	
recvbuff	ARRAY [1..1024] OF BYTE			
done0	BOOL		FALSE	
busy0	BOOL		FALSE	
error0	BOOL		FALSE	
status0	WORD		1008	
recvlen0	UDINT		0	
done1	BOOL		TRUE	
busy1	BOOL		FALSE	
error1	BOOL		FALSE	
status1	WORD		0	
TRCV_C_Instance_0	TRCV_C			
precvbuff	POINTER TO BYTE		3012802708	
TSEND_C_Instance_0	TSEND_C			
psedbuff	POINTER TO BYTE		3012802656	

```
TRCV_C_Instance_0(
En_R  TRUE  := N1  TRUE ,
Cont          TRUE          :=cont1  TRUE  ,
Len     10      := L0     10      ,
Connect := connect1,
Data        3012802708        := precvbuff       3012802708        ,
Done  FALSE  => done0  FALSE ,
Busy  FALSE  => busy0  FALSE ,
Error  FALSE  => error0  FALSE ,
Status  1008  =>status0  1008  ,
Rcvd_Len     0      => recvlen0     0      );
```

LD:



- **Precautions**
  - When connecting as a client, if the server has closed the connection, the client will also automatically close the socket connection.
  - When connecting as a server, after actively disconnecting the connection, the client needs to initiate a new connection request again.
- **Status code**

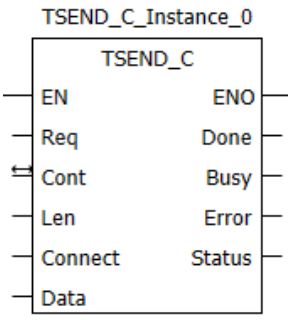
Status code	Note
0	Done
1000	Function not activated
1001	Port number error (local port/remote port)
1002	Module ID error (ID conflict or out of range)
1003	Connection type error (non-TCP/UDP)

Status code	Note
1004	Partner IP invalid or local IP conflict
1005	Maximum number of connections reached
1006	Data reception failed
1007	Reception address error
1008	No data
2000	Numbers of bytes received error (max bytes exceeded or negative value)
2001	Incorrect buffer parameter
3000	Data being received
3001	Connection established and monitored, no task in process
3002	Connection being established
3003	Duplicate IP connection attempt by another module
3004	Failed to create connection
3005	Disconnected
3006	The peer end disconnected

18.4.8.3 TSEND_C

This is the Ethernet free port data sending instruction.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
TSEND_C	Ethernet free port data sending instruction	FB		TSEND_C_Instance_0(Req := , Cont := , Len := , Connect := , Data := , Done => , Busy =>, Error => , Status =>);

- **Related variables**

  Variable

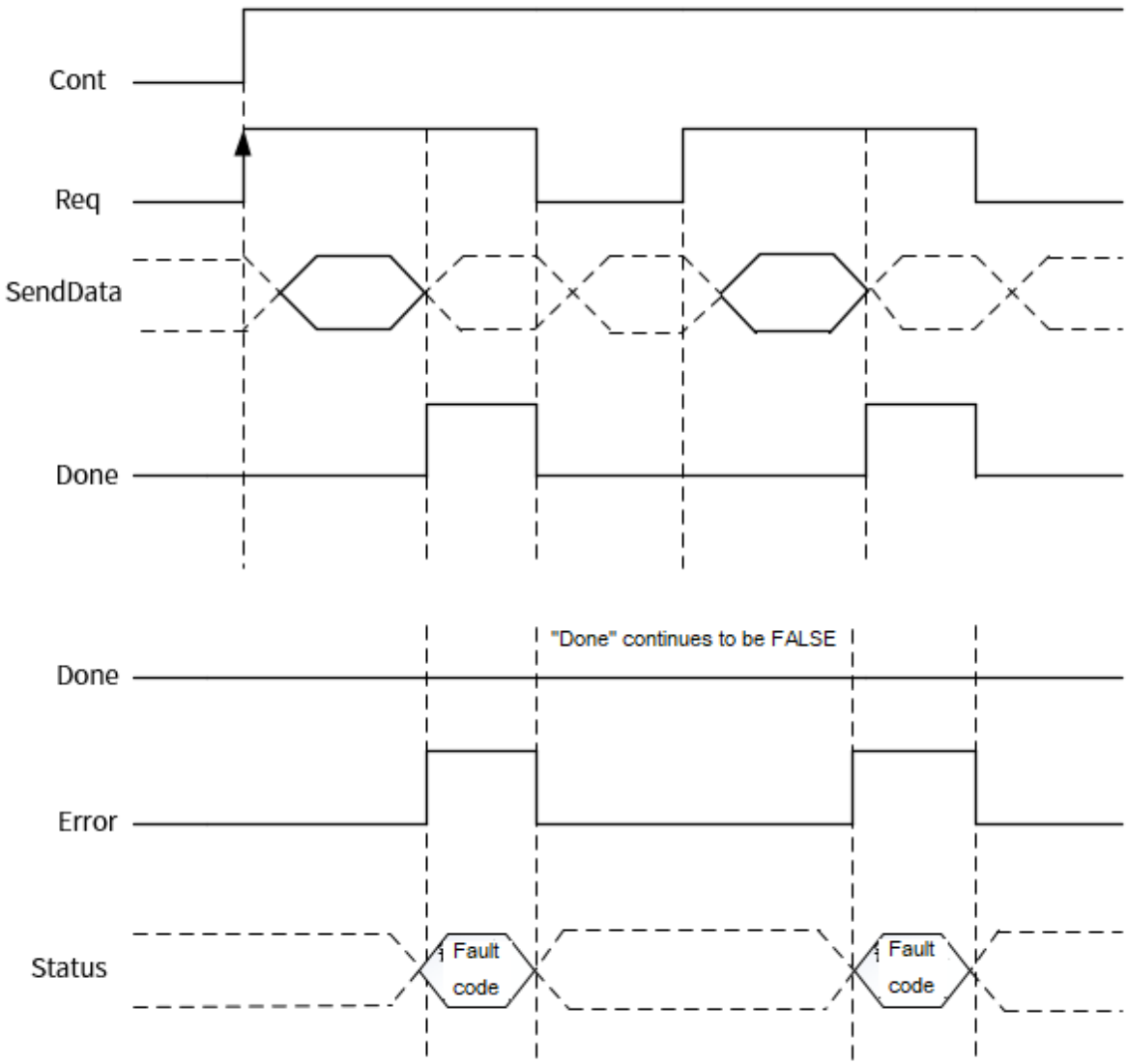
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Req	Enable start bit to send data	BOOL	Follows data type	FALSE	Enable start bit, active on rising edge.
Input/ Output	Cont	Control communication connection	BOOL	Follows data type	FALSE	Control communication connection: FALSE: Disconnect the communication connection. TRUE: Establish the communication connection and maintain it after data is received.
Input	Len	Sent data length	UDINT	Follows data type	1024	Maximum length of data to be sent (1 to 1024 bytes).
	Connect	Connection description structure	Tcont_Ip_v4	Follows data type	-	For connection description structure, see details in the Function description below.
	Data	Data transmission buffer	POINTER TO BYTE	Follows data type	-	Pointer to the sending data buffer.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	Done	Completion flag	BOOL	Follows data type	FALSE	Completion flag: FALSE: Sending not started or in progress. TRUE: Sending done.
	Busy	Reception status	BOOL	Follows data type	FALSE	Status parameter: FALSE: Data reception done or not started. TRUE: Data reception not done.
	Error	Error flag	BOOL	Follows data type	FALSE	Error flag: FALSE: No error TRUE: Reception error.
	Status	Status flag	WORD	Follow data type	1000	See Status code.

- **Function**

  The Tcont_Ip_v4 structure is identical to the 18.4.8.2 TRCV_CV receive instruction.

- **Sequence diagram**



- **Program example**

  This section takes TCP connection as example.

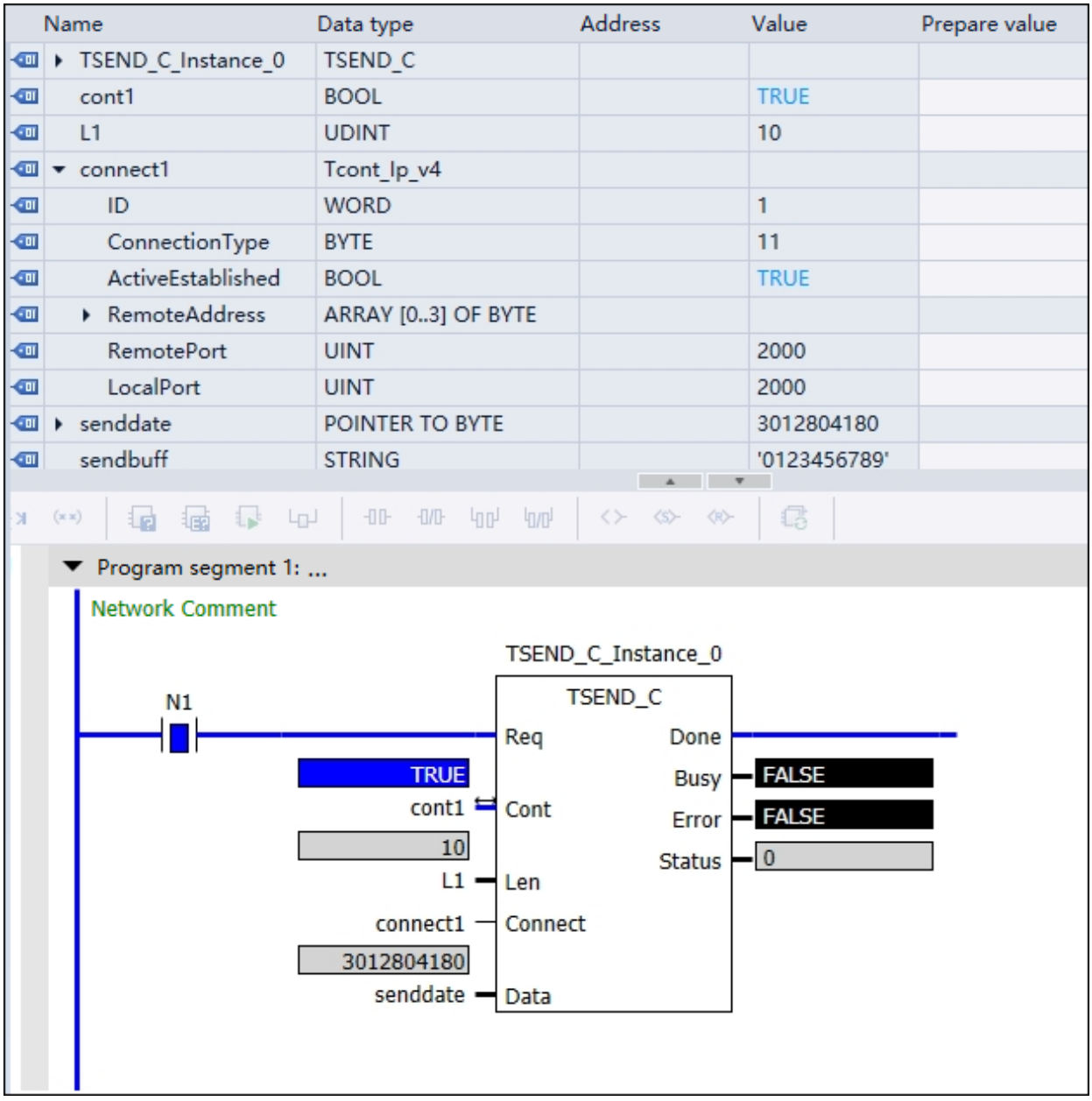
  ST:

Name	Data type	Address	Value	Prepare value
sedbuff	ARRAY [1..20] OF BYTE			
N1	BOOL		TRUE	
N2	BOOL		TRUE	
cont1	BOOL		TRUE	
L0	UDINT		10	
connect1	Tcont_Ip_v4			
cont2	BOOL		TRUE	
L1	UDINT		20	
recvbuff	ARRAY [1..1024] OF BYTE			
done0	BOOL		FALSE	
busy0	BOOL		FALSE	
error0	BOOL		FALSE	
status0	WORD		1008	
recvlen0	UDINT		0	
done1	BOOL		TRUE	
busy1	BOOL		FALSE	
error1	BOOL		FALSE	
status1	WORD		0	
TRCV_C_Instance_0	TRCV_C			
precvbuff	POINTER TO BYTE		3012802708	
TSEND_C_Instance_0	TSEND_C			
psedbuff	POINTER TO BYTE		3012802656	

```
TSEND_C_Instance_0(
Req    TRUE    := N2   TRUE   ,
Cont              TRUE          := cont1  TRUE   ,
Len      20       := L1     20       ,
Connect :=connect1 ,
Data          3012802656          := psedbuff          3012802656          ,
Done   TRUE    => done1  TRUE   ,
Busy   FALSE   => busy1  FALSE  ,
Error  FALSE   =>error1  FALSE  ,
Status    0      =>status1    0      );

RETURN
```

LD:



- **Precautions**
  - When connecting as a client, if the server has closed the connection, the client will also automatically close the socket connection.
  - When connecting as a server, after actively disconnecting the connection, the client needs to initiate a new connection request again.
- **Status code**

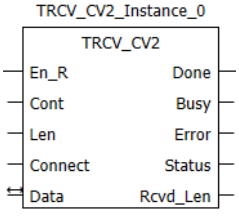
Status code	Note
0	Done
1000	Function not activated
1001	Port number error (local port/remote port)
1002	Module ID error (ID conflict or out of range)
1003	Connection type error (non-TCP/UDP)

Status code	Note
1004	Partner IP invalid or local IP conflict
1005	Maximum number of connections reached
1006	Data sending failed
1007	Sending address error
2000	Number of bytes sent error (max bytes exceeded or negative value)
2001	Incorrect buffer parameter
3000	Data being sent
3001	Connection established and monitored, no task in process
3002	Connection being established
3003	Duplicate IP connection attempt by another module
3004	Failed to create connection
3005	Disconnected
3006	The peer end disconnected

18.4.8.4 TRCV_CV2

This instruction receives data through network ports by using freeports.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
TRCV_CV2	Network port freeport reception	FB		TRCV_CV2_Instance_0(En_R := , Cont := , Len := , Connect := , Data := , Done => , Busy => , Error => , Status => , Rcvd_Len =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	En_R	Enable start bit to receive data	BOOL	Follows data type	FALSE	Enable start bit, active at high level.
	Cont	Control communication connection	BOOL	Follows data type	FALSE	Control communication connection: FALSE: Disconnect the communication connection. TRUE: Establish the communication connection and maintain it after data is received.
	Len	Received data length	UDINT	Follows data type	0	When Len = 0, it indicates receiving data of indefinite length, and the maximum size of received data is determined by the Data. When Len > 0, it indicates receiving data of fixed length, with a range of 1 to 8192 bytes.
	Connect	Connection description structure	Tcont_Ip_v4	Follows data type	-	For connection description structure, see details in the Function description below.
Input/ Output	Data	Data reception buffer	ARRAY[*] OF BYTE	Follows data type	-	Data reception buffer.
Output	Done	Completion flag	BOOL	Follows data type	FALSE	Completion flag: FALSE: Sending not started or in progress. TRUE: Sending done.
	Busy	Reception status	BOOL	Follows data type	FALSE	Status parameter: FALSE: Data reception done or not started. TRUE: Data reception not done.
	Error	Error flag	BOOL	Follows data type	FALSE	Error flag: FALSE: No error TRUE: Reception error.
	Status	Status flag	WORD	Follow data type	1000	See Status code.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Rcvd_Len	Received data length	UDINT	Follows data type	0	Actual amount of received data.

- **Function**

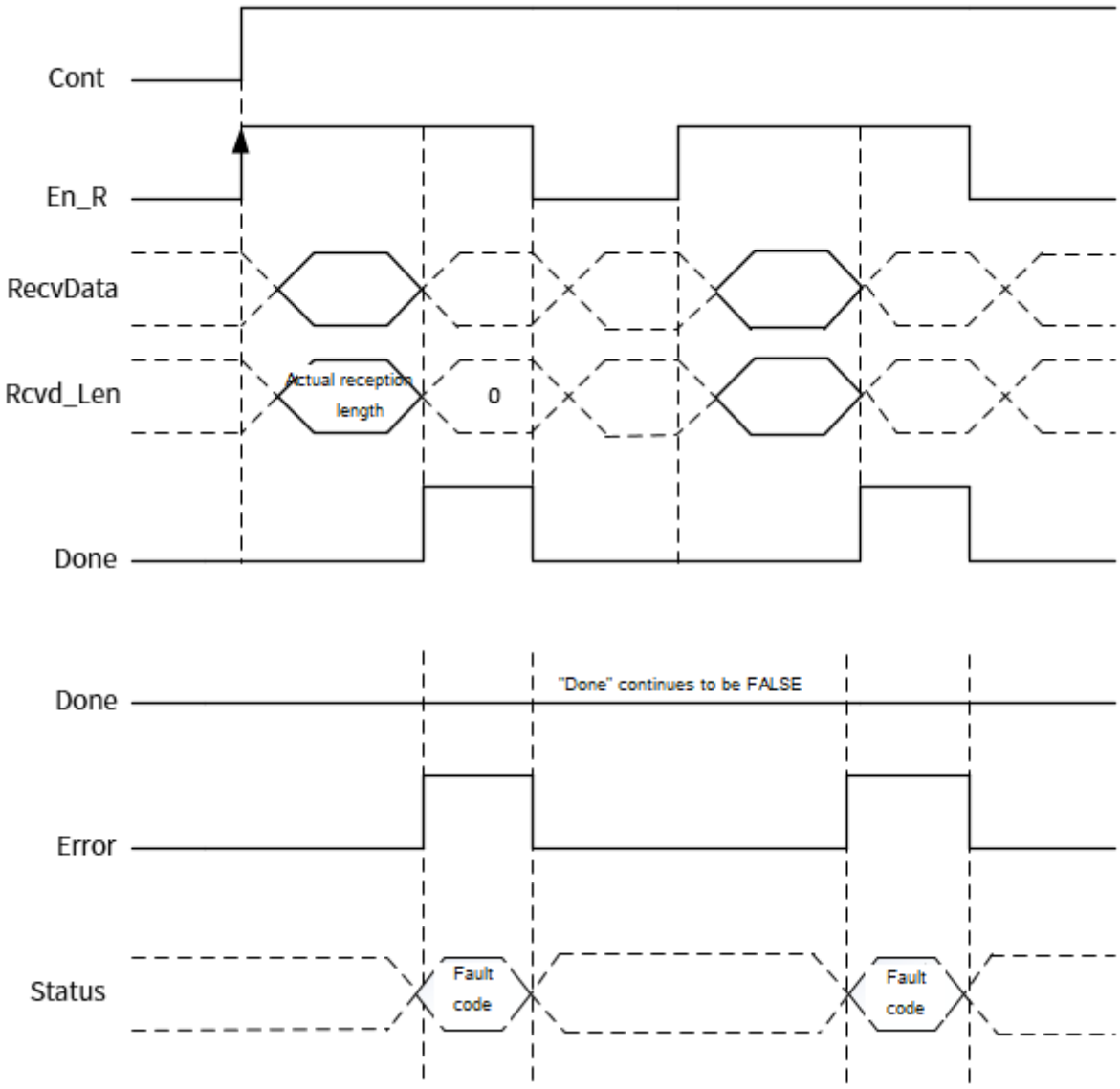
  When using the receive function block, set instruction reception frequency over 2x sending frequency to ensure data consistency per cycle; otherwise, reception delay occurs.

  The Tcont_Ip_v4 structure is shown in the table below.

Parameter	Type	Note
ID	WORD	Module ID: One connection corresponds to one ID, and each Module ID value is unique (value range: 1 to 4095).
ConnectionType	BYTE	Connection type: 11: TCP 19: UDP
ActiveEstablished	BOOL	Identifier of connection establishment type: FALSE: Passive connection TRUE: Active connection
RemoteAddress	ARRAY[0..3] OF BYTE	It is the IP of remote connection partner. For example, 192.168.0.1, where addr[1] = 192, addr[2] = 168, addr[3] = 0, addr[4] = 1. Broadcasting is not supported. IP is invalid for TCP passive connection.
RemotePort	UINT	It is the port number of remote connection partner (range: 1 to 49151) with a default value of 2000, recommended to use port numbers after 2000 (except for dedicated port numbers).

Parameter	Type	Note
LocalPort	UINT	It is the local port number (range: 1 to 49151) with a default value of 2000, recommended to use port numbers after 2000 (except for dedicated port numbers).

- **Sequence diagram**



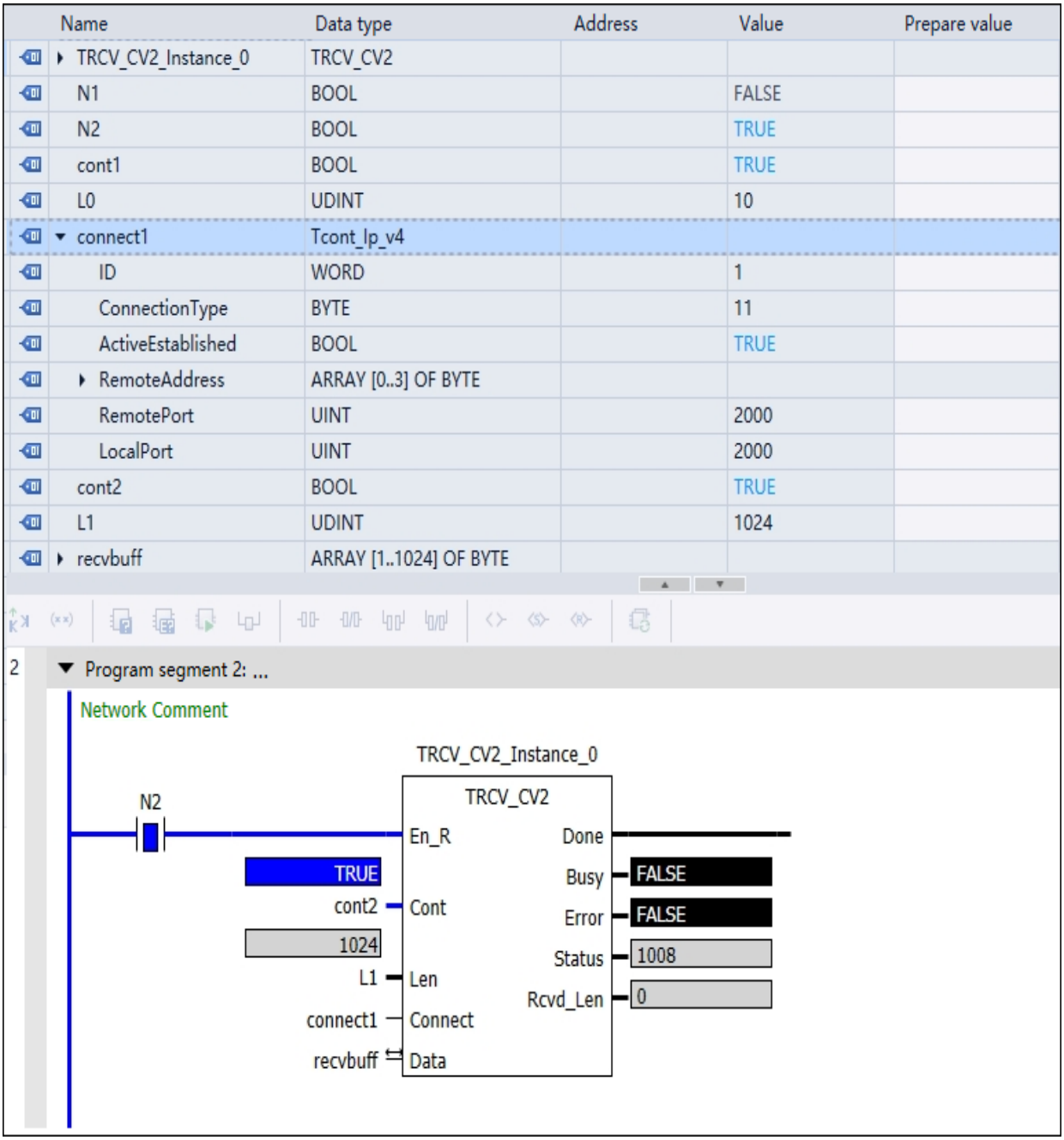
- **Program example**

  ST:

Name	Data type	Address	Value	Prepare value
TRCV_CV2_Instance_1	TRCV_CV2			
TSEND_CV2_Instance_1	TSEND_CV2			
sedbuff	ARRAY [1..20] OF BYTE			
N1	BOOL		TRUE	
N2	BOOL		FALSE	
cont1	BOOL		TRUE	
L0	UDINT		10	
connect1	Tcont_Ip_v4			
cont2	BOOL		TRUE	
L1	UDINT		1024	
recvbuff	ARRAY [1..1024] OF BYTE			

```
CAS FOR WHI REP RET EX REG
TRCV_CV2_Instance_1(
En_R  TRUE  := N1  TRUE ,
Cont  TRUE  :=cont1  TRUE  ,
Len     10     := L0     10     ,
Connect := connect1,
Data          3012294756          := recvbuff,
Done  FALSE  => done0  FALSE ,
Busy  FALSE  => busy0  FALSE ,
Error  FALSE  => error0  FALSE ,
Status   1008   =>status0   1008   ,
Rcvd_Len     0      => recvlen0     0      );
```

LD:



- **Precautions**
  - When connecting as a client, if the server has closed the connection, the client will also automatically close the socket connection.
  - When connecting as a server, after actively disconnecting the connection, the client needs to initiate a new connection request again.
  - The reception length of the TRCV_CV2 is configurable, with a maximum of 8192 bytes. When set to 0, the reception length is the actual length of the received data; when set to a specific value, it is necessary to verify whether the actual received length is consistent with the configured length.
- **Status code**

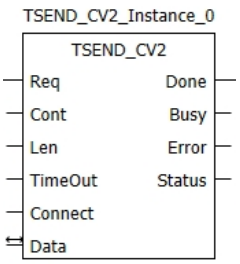
Status code	Note
0	Done
1000	Function not activated

Status code	Note
1001	Port number error (local port/remote port)
1002	Module ID error (ID conflict or out of range)
1003	Connection type error (non-TCP/UDP)
1004	Partner IP invalid or local IP conflict
1005	Maximum number of connections reached
1006	Data reception failed
1007	Reception address error
1008	No data
2000	Numbers of bytes received error (max bytes exceeded or negative value)
2001	Incorrect buffer parameter
3000	Data being received
3001	Connection established and monitored, no task in process
3002	Connection being established
3003	Duplicate IP connection attempt by another module
3004	Failed to create connection
3005	Disconnected
3006	The peer end disconnected

18.4.8.5 TSEND_CV2

This is the Ethernet free port data sending instruction.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
TSEND_CV2	Ethernet free port data sending instruction	FB		TSEND_CV2_Instance_0(Req := , Cont := , Len := , TimeOut := , Connect := , Data := , Done => , Busy => , Error => , Status =>);

- **Related variables**

Variable

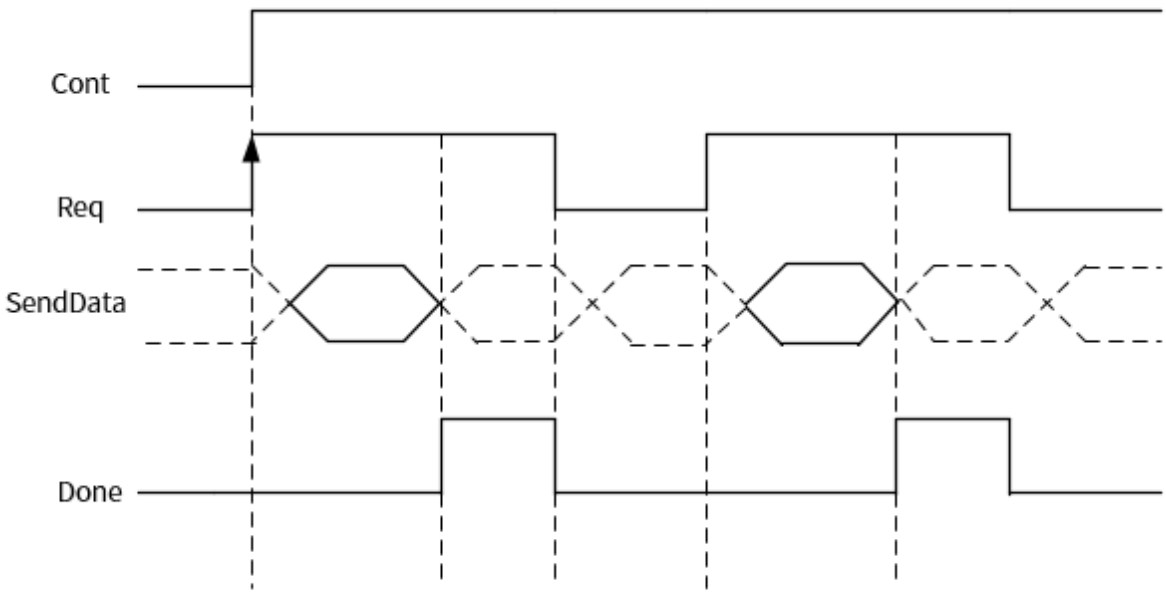
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Req	Enable start bit to send data	BOOL	Follows data type	FALSE	Enable start bit, active on rising edge.
	Cont	Control communication connection	BOOL	Follows data type	FALSE	Control communication connection: FALSE: Disconnect the communication connection. TRUE: Establish the communication connection and maintain it after data is received.
	Len	Sent data length	UDINT	Follows data type	0	Maximum length of data to be sent (1 to 8192 bytes).
	TimeOut	Sending timeout	UINT	Follow data type	50	Sending timeout in ms.
	Connect	Connection description structure	Tcont_Ip_v4	Follows data type	-	For connection description structure, see details in the Function description below.
Input/ Output	Data	Sending data buffer	ARRAY[*] OF BYTE	Follows data type	-	Sending data buffer.
Output	Done	Completion flag	BOOL	Follows data type	FALSE	Completion flag: FALSE: Sending not started or in progress. TRUE: Sending done.

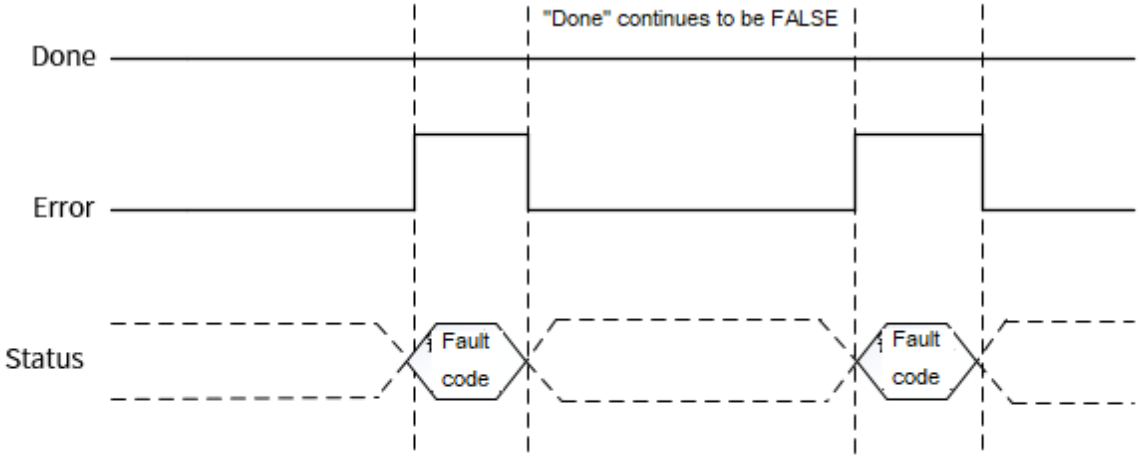
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Busy	Reception status	BOOL	Follows data type	FALSE	Status parameter: FALSE: Data reception done or not started. TRUE: Data reception not done.
	Error	Error flag	BOOL	Follows data type	FALSE	Error flag: FALSE: No error TRUE: Reception error.
	Status	Status flag	WORD	Follow data type	1000	See Status code.

- **Function**

  The Tcont_Ip_v4 structure is identical to the 18.4.8.4 TRCV_CV2 receive instruction.

- **Sequence diagram**





- **Program example**

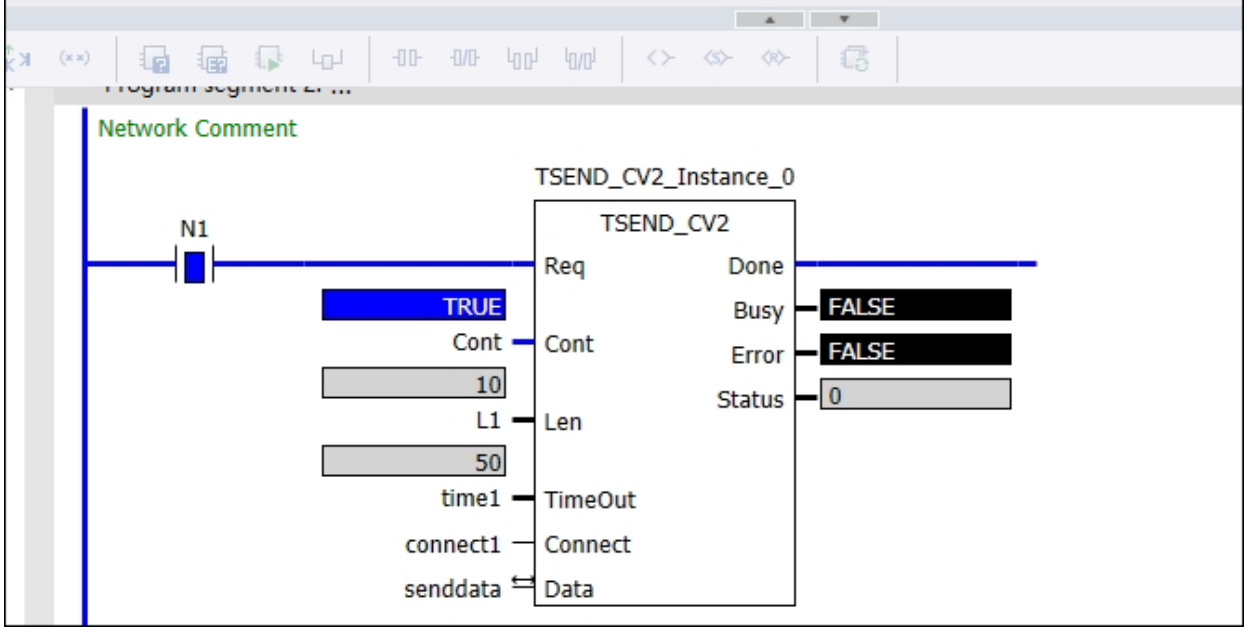
ST:

Name	Data type	Address	Value	Prepare value
TSEND_CV2_Instance_0	TSEND_CV2			
N1	BOOL		TRUE	
con1	BOOL		TRUE	
L1	UDINT		10	
time1	UINT		50	
connect1	Tcont_Ip_v4			
ID	WORD		1	
ConnectionType	BYTE		11	
ActiveEstablished	BOOL		TRUE	
RemoteAddress	ARRAY [0..3] OF BYTE			
RemotePort	UINT		2000	
LocalPort	UINT		2000	
senddata	ARRAY [1..10] OF BYTE			
done1	BOOL		TRUE	
busy1	BOOL		FALSE	
error1	BOOL		FALSE	
state1	WORD		0	

```
TSEND_CV2_Instance_0(
Req TRUE := N1 TRUE,
Cont TRUE :=con1 TRUE ,
Len 10 := L1 10,
TimeOut 50 :=time1 50 ,
Connect := connect1,
Data 140285107503588 := senddata,
Done TRUE => done1 TRUE,
Busy FALSE => busy1 FALSE ,
Error FALSE => error1 FALSE ,
Status 0 => state1 0 );
RETURN
```

LD:

Name	Data type	Address	Value	Prepare value
TSEND_CV2_Instance_0	TSEND_CV2			
N1	BOOL		TRUE	
Cont	BOOL		TRUE	
L1	UDINT		10	
time1	UINT		50	
connect1	Tcont_Ip_v4			
ID	WORD		1	
ConnectionType	BYTE		11	
ActiveEstablished	BOOL		TRUE	
RemoteAddress	ARRAY [0..3] OF BYTE			
RemoteAddress[0]	BYTE		10	
RemoteAddress[1]	BYTE		61	
RemoteAddress[2]	BYTE		83	
RemoteAddress[3]	BYTE		111	
RemotePort	UINT		2000	
LocalPort	UINT		2000	
senddata	ARRAY [1..10] OF BYTE			



- **Precautions**
  - When connecting as a client, if the server has closed the connection, the client will also automatically close the socket connection.
  - When connecting as a server, after actively disconnecting the connection, the client needs to initiate a new connection request again.
- **Status code**

Status code	Note
0	Done
1000	Function not activated
1001	Port number error (local port/remote port)
1002	Module ID error (ID conflict or out of range)

Status code	Note
1003	Connection type error (non-TCP/UDP)
1004	Partner IP invalid or local IP conflict
1005	Maximum number of connections reached
1006	Data sending failed
1007	Sending address error
2000	Incorrect number of transmitted bytes (max. bytes exceeded or negative value)
2001	Incorrect buffer parameter
2002	Send timeout
3000	Data being sent
3001	Connection established and monitored, no task in process
3002	Connection being established
3003	Duplicate IP connection attempt by another module
3004	Failed to create connection
3005	Disconnected
3006	The peer end disconnected

18.4.8.6 Example of Free Port Communication Instruction Usage

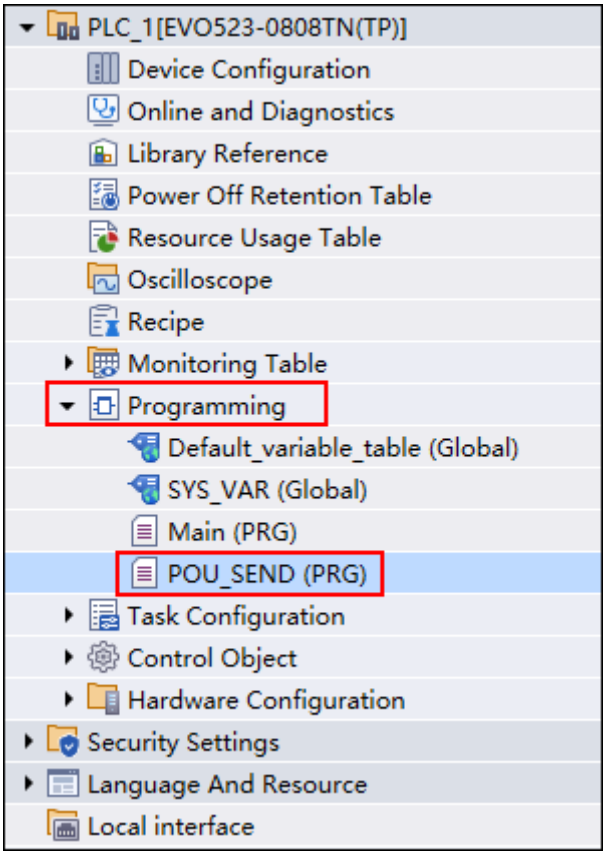
This section takes the TSEND_C Ethernet free port data sending instruction as an example. An EVO523 controller serves as the TCP client, and a commissioning assistant acts as the TCP server.

Prerequisites

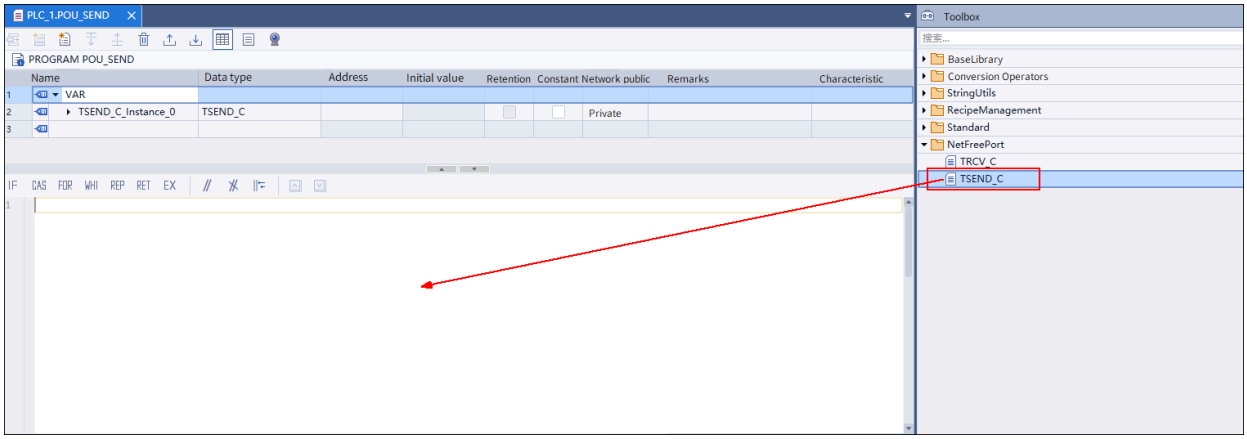
- A project has been created and the device added. For specific operations, see 7 Project Deployment.
- The NetFreePort library has been referenced. For reference operations, see 12.3 Library Reference.

Procedure

1. Click "Programming" in the left project tree to create a POU of PRG type, with ST and LD languages both supported. This section takes the ST language as an example.



2. Double-click to open the POU interface, select the "TSEND_C" instruction from the right Toolbox, and then double-click or drag it directly to the program editing area to create an instruction instance object.



3. In the table view/text view variable declaration editor, add variables, set properties such as variable name and data type, and configure the instruction-related pins. The text view variable declaration editor is used as an example below.

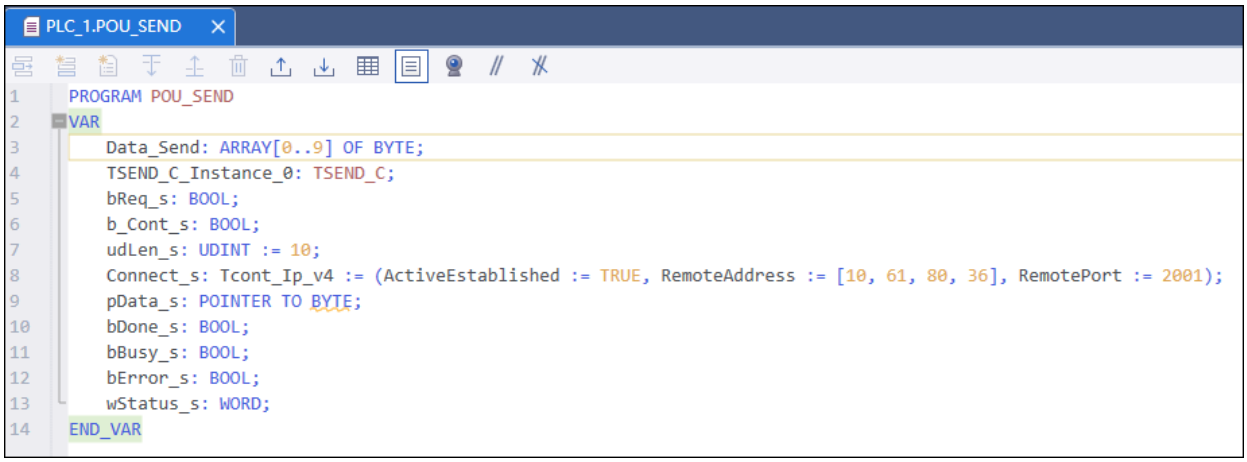
NOTE

To facilitate the direct input of data to send in variables, the ADR instruction can be used to bind the pointer of the data to the actual variable address. In the example above, a BYTE-type array of length 10 is defined. Set the array length according to actual needs.

Note:

- **The binding of the pointer to the variable address (i.e., the ADR instruction highlighted in the red in the program) must be placed before the sending instruction; otherwise, it may result in sending data error for the first time.**
- **The warning in the variable declaration area stating "It is not recommended to define or use pointer-type variables" can be ignored and does not affect functionality.**

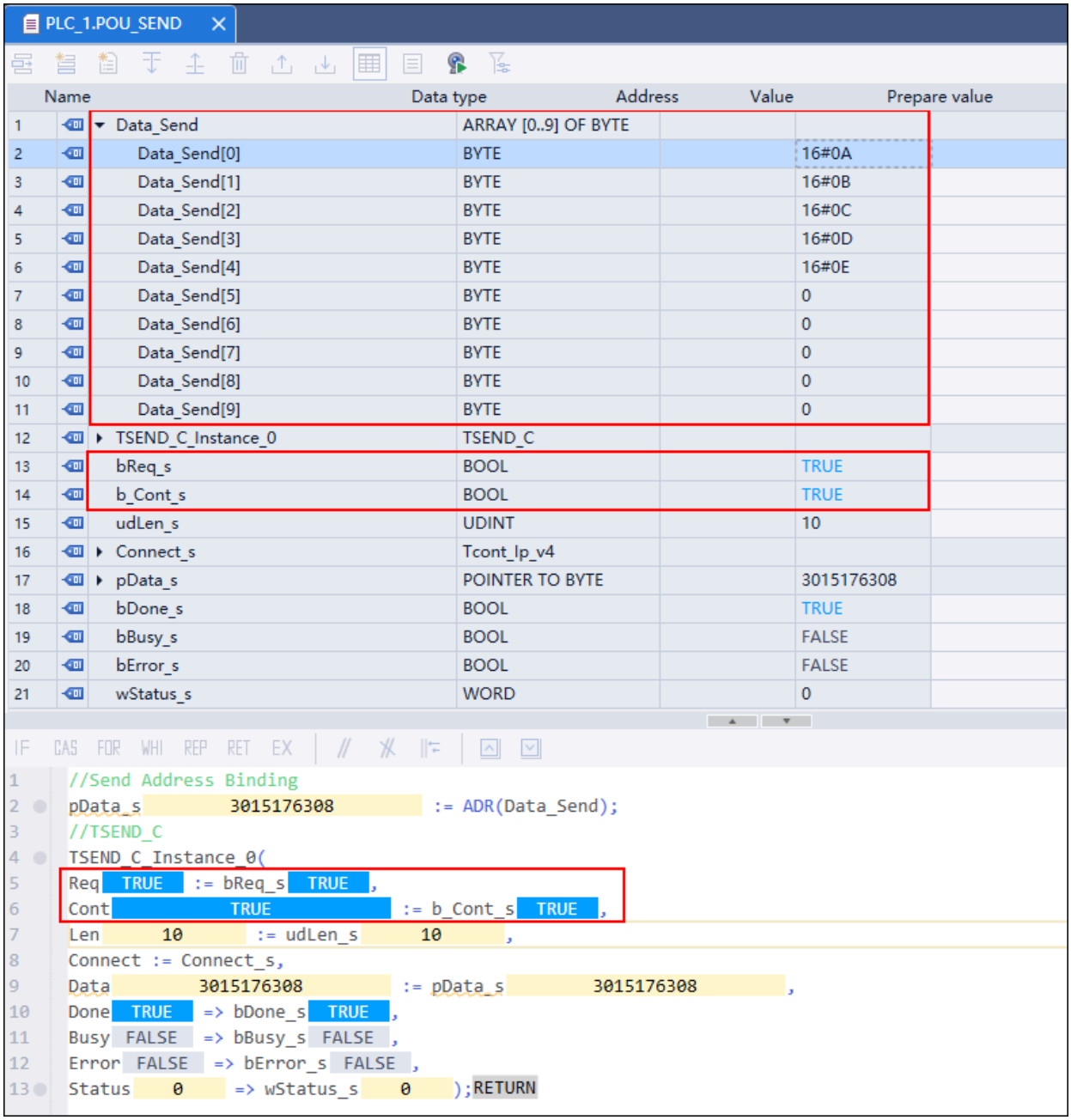
4. To configure the connection, specify the value of the instruction's Connect pin. For detailed parameter descriptions, see 18.4.8.3 TSEND_C.



5. Call POU_SEND in MainTask, download it to the controller, and start its operation.

6. Open the commissioning assistant, select the TCP Server mode. The local IP address must be consistent with the IP configured in Connect.RemoteAddress, and the port number must be consistent with Connect.RemotePort. Then, open the connection.

7. Open the iFA software and switch to the online mode. Write a value into Data_Send (i.e., the variable bound to the pointer), set the Cont pin to TRUE, and trigger the Req pin simultaneously to establish a connection. After the connection is successfully established, one data sending can be completed (one sending per rising edge).



8. In the commissioning assistant, the data sent can be viewed. For easier observation, configure the variable declaration editor to convert and send values in hexadecimal format.

NOTE

During commissioning, pay attention to the status code of Status, and see 18.4.8.3 TSEND_C for details.

Result

Normal communication and data sending between the EVO523 controller and the commissioning assistant.

18.4.9 EtherCAT

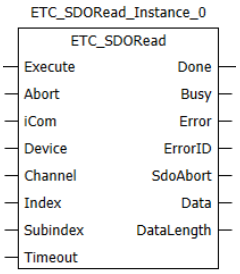
18.4.9.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.4.9.2 ETC_SDORead	FB	Read parameters from EtherCAT slave	EtherCAT	Library versions 1.3.0.0 and earlier
18.4.9.3 ETC_SDOWrite	FB	Write parameters to EtherCAT slave		Library versions 1.3.0.0 and earlier
18.4.9.4 ETC_COE_SdoRead	FB	Read parameters from EtherCAT slave		Library versions 1.4.0.0 and later
18.4.9.5 ETC_COE_SdoWrite	FB	Write parameters to EtherCAT slave		Library versions 1.4.0.0 and later
18.4.9.6 ETC_ClearSlavesErrorCounter	FB	Clear EtherCAT slave error frame count		Library version 1.2.0.0 and later
18.4.9.7 ETC_DisableDevice	FB	Disable EtherCAT device		Library version 1.2.0.0 and later
18.4.9.8 ETC_GetCouplerInfo	FB	Read EtherCAT coupler information		Library version 1.2.0.0 and later
18.4.9.9 ETC_GetMasterinfo	FB	Get EtherCAT master information		Library version 1.2.0.0 and later
18.4.9.10 ETC_GetSlaveInfo	FB	Get EtherCAT slave information		Library version 1.2.0.0 and later
18.4.9.11 ETC_SlaveReset	FB	Reset EtherCAT slave (resettable axis communication)		Library versions 1.4.0.0 and later
18.4.9.12 ETC_MasterRestart	FB	Restart EtherCAT master		Library versions 1.4.0.0 and later
18.4.9.13 ETC_MasterDiagnose	FB	Get EtherCAT master diagnosis information		Library versions 1.4.0.0 and later

18.4.9.2 ETC_SDORead

This instruction reads parameters from an EtherCAT slave. Based on the slave address, index, and subindex parameters configured on the pins, it obtains data from an entry in the corresponding object dictionary and outputs the data via the output pins.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ETC_SDORead	Read parameters from EtherCAT slave	FB		ETC_SDORead_Instance_0(Execute := , Abort := , iCom := , Device := , Channel := , Index := , Subindex := , Timeout := , Done => , Busy => , Error => , ErrorID => , SdoAbort => , Data => , DataLength =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Execute	BOOL	TRUE, FALSE	FALSE	Rising edge triggered.
	Abort	Abort	BOOL	TRUE, FALSE	FALSE	If it is set to "TRUE", the write process is aborted.
	iCom	Number of masters	BYTE	Follows data type	1	Number of EtherCAT masters: If only one EtherCAT master is used, iCom is '1'. When multiple masters are used, the value is 1 for the first master, 2 for the second master, and so on.
	Device	Slave physical address	UINT	Follow data type	-	If the automatic configuration mode of the master is disabled, the slave can be set to a specific address. The freely chosen address must be input here. If auto-configuration is enabled, the first slave will receive address 1001. The current slave address can always be verified in the EtherCAT address check.
	Channel	Reserved	BYTE	Follow data type	1	Reserved.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Index	Object dictionary index	WORD	Follow data type	-	Object dictionary index.
	Subindex	Object dictionary subindex	BYTE	Follow data type	-	Object dictionary subindex.
	Timeout	Timeout period	UDINT	Follow data type	-	Timeout period in ms. If the write parameter is not executed within this time, an error message will be generated.
Output	Done	Done	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is done.
	Busy	Instruction executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is executing.
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
	ErrorID	Fault code	ETC_SDO_ERROR	-	ETC_NO_ERROR	See 18.4.9.14 ETC_SDO_ERROR: ETC Fault Codes for details.
	SdoAbort	SDO fault code	UDINT	Follow data type	-	If a fault occurs in the device, more information about the fault can be obtained with this output variable. For details, see 18.4.9.17 SDO Fault Codes.
	Data	Read data	DWORD	Follow data type	-	The parameter data read will be copied into this 4-byte value.
	DataLength	Number of bytes read	BYTE	Follow data type	-	Number of bytes read.

- **Function**

  This instruction performs a CoE read operation. Based on the slave address, index, and subindex parameters configured on the pins, it retrieves data from an entry in the corresponding object dictionary and outputs the data via the output pins.

- **Program example**

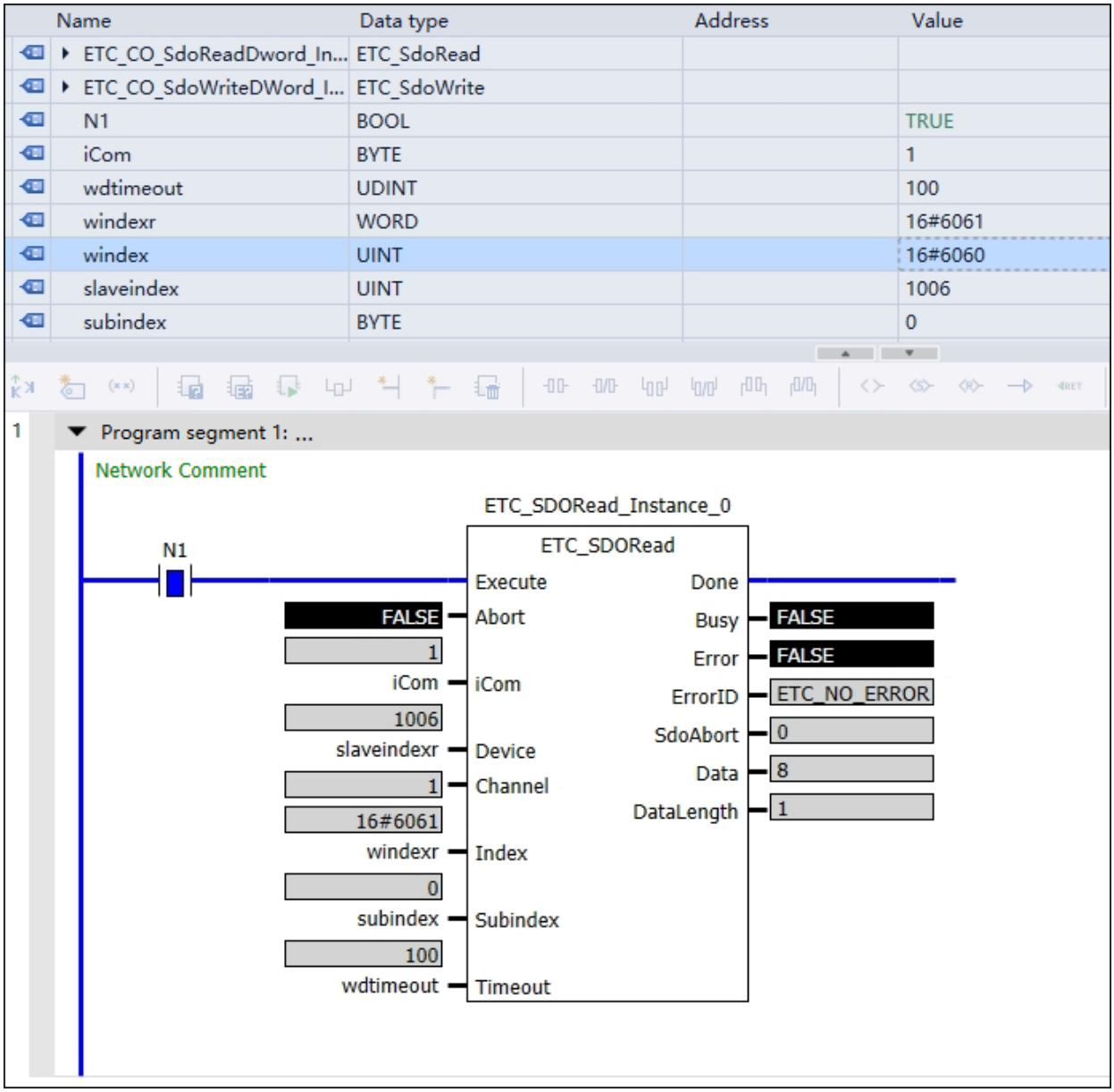
  Taking one of the slaves as an example, the slave address is 1006, with an index of 0x6060 and a subindex of 0.

  ST:

Name	Data type	Address	Value
ETC_SDORead_Instance_0	ETC_SDORead		
N1	BOOL		TRUE
iCom	BYTE		1
wdtimeout	UDINT		100
windex	UINT		16#6060
slaveindex	UINT		1006
subindex	BYTE		0
date1	DWORD		4
lent	USINT		1
N2	BOOL		FALSE
done1	BOOL		TRUE
busy1	BOOL		FALSE
error1	BOOL		FALSE
errorid1	ETC_SDO_ERROR		ETC_NO_ERROR
sdoabort1	UDINT		0
data1	DWORD		8
datalength1	BYTE		1

```
CAS  FOR  WHI  REP  RET  EX
ETC_SDORead_Instance_0(
Execute TRUE := N1 TRUE,
Abort FALSE :=FALSE ,
iCom 1 := iCom 1,
Device 1006 := slaveindex 1006,
Channel 1 := 1,
Index 16#6060 := windex 16#6060,
Subindex 0 := subindex 0,
Timeout 100 := wdtimeout 100,
Done TRUE => done1 TRUE,
Busy FALSE =>busy1 FALSE,
Error FALSE => error1 FALSE,
ErrorIDETC_NO_ER... =>errorid1ETC_NO_ER... ,
SdoAbort 0 => sdoabort1 0,
Data 8 =>data1 8 ,
DataLength 1 =>datalength1 1 );RETURN
```

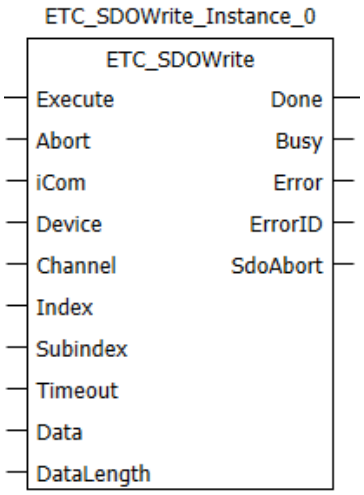
LD:



18.4.9.3 ETC_SDOWrite

This instruction writes parameters to an EtherCAT slave. Based on the slave address, index, and subindex parameters configured on the pins, it writes a specific value to an entry in the corresponding object dictionary.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ETC_SDOWrite	Write parameters to EtherCAT slave	FB		ETC_SDOWrite_Instance_0(Execute := , Abort := , iCom := , Device := , Channel := , Index := , Subindex := , Timeout := , Data := , DataLength := , Done => , Busy => , Error => , ErrorID => , SdoAbort =>);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Execute	BOOL	TRUE, FALSE	FALSE	Rising edge triggered.
	Abort	Abort	BOOL	TRUE, FALSE	FALSE	If it is set to "TRUE", the write process is aborted.
	iCom	Number of masters	BYTE	Follows data type	1	Number of EtherCAT masters: If only one EtherCAT master is used, iCom is '1'. When multiple masters are used, the value is 1 for the first master, 2 for the second master, and so on.
	Device	Slave physical address	UINT	Follow data type	-	If the automatic configuration mode of the master is disabled, the slave can be set to a specific address. The freely chosen address must be input here. If automatic configuration is enabled, the first slave will receive address 1001. The current slave address can always be verified in the EtherCAT address check.
	Channel	Reserved	BYTE	Follows data type	1	Reserved.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Index	Object dictionary index	WORD	Follows data type	-	Object dictionary index.
	Subindex	Object dictionary subindex	BYTE	Follows data type	-	Object dictionary subindex.
	Timeout	Timeout period	UDINT	Follows data type	-	Timeout period in ms. If the write parameter is not executed within this time, an error message will be generated.
	Data	Data to be written	DWORD	Follows data type	-	This parameter contains 4 bytes of data to be written.
	DataLength	Number of bytes to be written	BYTE	Follows data type	-	Number of bytes to be written.
Output	Done	Done	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is done.
	Busy	Instruction executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is executing.
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
	ErrorID	Fault code	ETC_SDO_ERROR	-	ETC_NO_ERROR	See 18.4.9.14 ETC_SDO_ERROR: ETC Fault Codes for details.
	SdoAbort	SDO fault code	UDINT	Follows data type	-	If a fault occurs in the device, more information about the fault can be obtained with this output variable. For details, see 18.4.9.17 SDO Fault Codes.

- **Function**

  This instruction performs a CoE write operation. Based on the slave address, index, and subindex parameters configured on the pins, it writes a specific value to the corresponding object dictionary.

- **Program example**

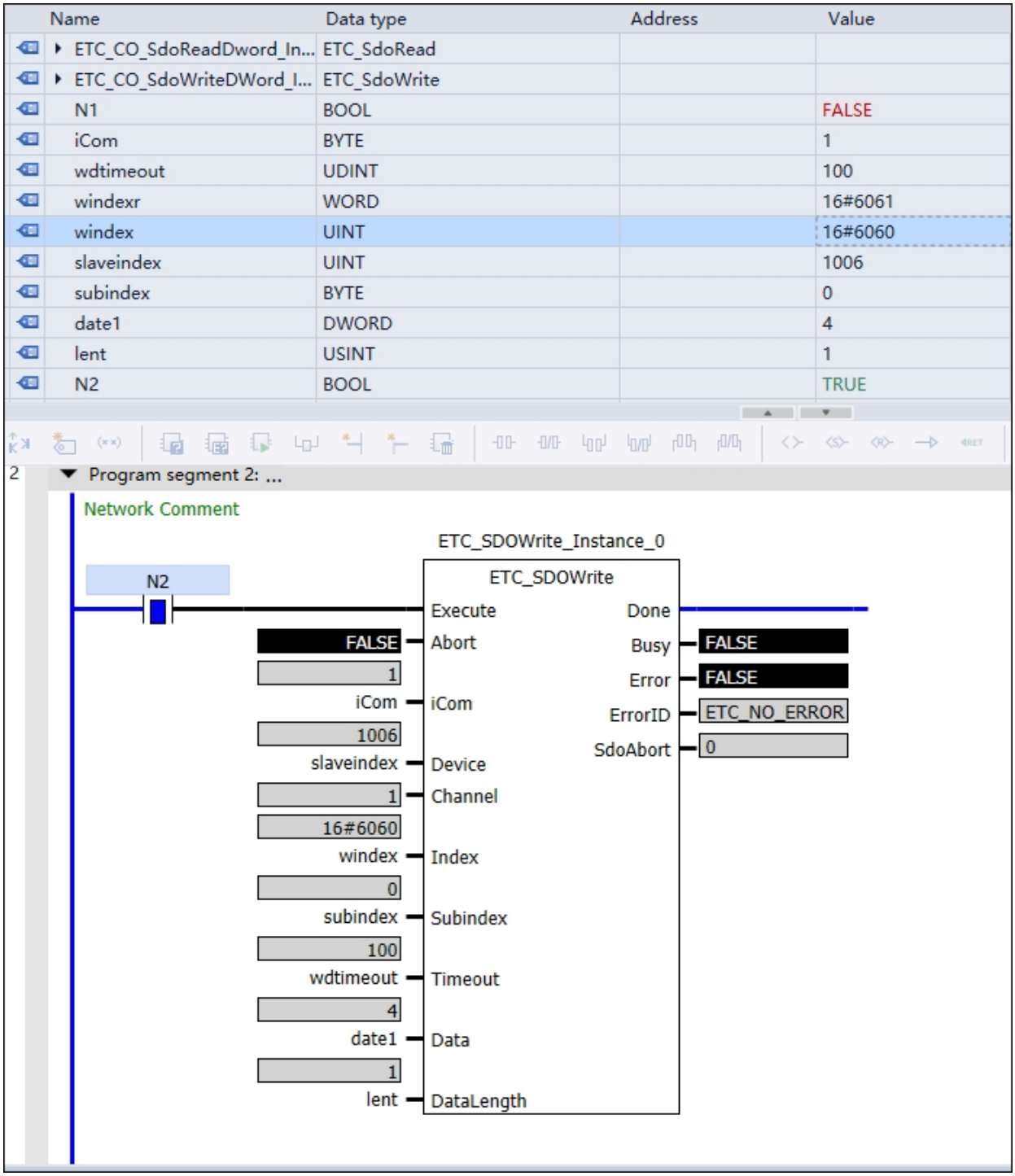
  Taking one of the slaves as an example, the slave address is 1006, with an index of 0x6060 and a subindex of 0.

  ST:

Name	Data type	Address	Value
ETC_SDOWrite_Instance_0	ETC_SDOWrite		
N1	BOOL		TRUE
iCom	BYTE		1
wdtimeout	UDINT		100
windex	UINT		16#6060
slaveindex	UINT		1006
subindex	BYTE		0
date1	DWORD		4
lent	USINT		1
N2	BOOL		FALSE
done1	BOOL		TRUE
busy1	BOOL		FALSE
error1	BOOL		FALSE
errorid1	ETC_SDO_ERROR		ETC_NO_ERROR
sdoabort1	UDINT		0
data1	DWORD		4
datalength1	BYTE		1

```
ETC_SDOWrite_Instance_0(
Execute  TRUE  := N1  TRUE ,
Abort  FALSE  :=FALSE ,
iCom  1  :=iCom  1  ,
Device  1006  := slaveindex  1006 ,
Channel  1  := 1,
Index  16#6060  := windex  16#6060 ,
Subindex  0  := subindex  0 ,
Timeout  100  := wdtimeout  100 ,
Data  4  := data1  4 ,
DataLength  1  := datalength1  1 ,
Done  TRUE  => done1  TRUE ,
Busy  FALSE  => busy1  FALSE ,
Error  FALSE  => error1  FALSE ,
ErrorIDETC_NO_ER... =>errorid1ETC_NO_ER... ,
SdoAbort  0  => sdoabort1  0 );RETURN
```

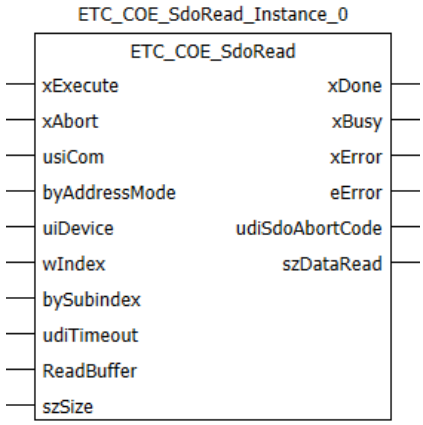
LD:



18.4.9.4 ETC_COE_SdoRead

This instruction reads parameters from entries in the object dictionary of EtherCAT slave and supports reading object data of any length.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
ETC_COE_SdoRead	Read parameters from EtherCAT slave	FB		ETC_COE_SdoRead_Instance_0(xExecute := , xAbort := , usiCom := , byAddressMode := , uiDevice := , wIndex := , bySubindex := , udiTimeout := , ReadBuffer := , szSize := , xDone => , xBusy => , xError => , eError => , udiSdoAbortCode => , szDataRead =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Function block execution	BOOL	TRUE, FALSE	FALSE	In the input rising edge, read slave parameters.
	xAbort	Function block execution aborted	BOOL	TRUE, FALSE	FALSE	If it is set to "TRUE", the read process is aborted.
	usiCom	Number of EtherCAT masters	BYTE	1 to 2	1	• When only one EtherCAT master is used, the value is 1. • When multiple masters are used, the value is 1 for the first master, 2 for the second master, and so on.
	byAddressMode	Slave addressing mode	BYTE	0 or 1	0	• 0: Sequential mode • 1: Alias mode

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	uiDevice	Physical address of the slave	UINT	1 to 65535	0	• In sequential mode, values are auto-assigned such as 1001, 1002. • In alias mode, values are user-defined such as 1, 2.
	wIndex	Object dictionary index	WORD	0 to 65535	0	Example: Object dictionary index 16#6061 of the servo running mode
	bySubindex	Object dictionary subindex	BYTE	0 to 255	0	Example: Object dictionary sub-index 16#00 of the servo homing mode
	udiTimeout	Timeout period	UDINT	0 to 65535	0	During the period (unit: ms), if the request to read parameters of the slave is not replied, an error message is displayed.
	ReadBuffer	Data buffer	ANY	-	-	Area for storing data after successful parameter transmission
	szSize	Size of the data buffer	_UXINT	0 to 255	0	Available size of the data buffer (ReadBuffer)
Output	xDone	Reading parameters completed	BOOL	TRUE, FALSE	FALSE	The value is "TRUE" when parameter read is completed.
	xBusy	Reading parameters not completed	BOOL	TRUE, FALSE	FALSE	The value is "TRUE" if the parameter read is not completed.
	xError	Reading parameters error	BOOL	TRUE, FALSE	FALSE	The value is "TRUE" if an error occurs, and eError will indicate the cause.
	eError	Fault ID	ETC_CO_ERROR	-	ETC_CO_NO_ERROR	For details, see 18.4.9.16 ETC_CO_ERROR: ETC Fault Codes.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	udiSdoAbortCode	SDO fault code	UDINT	-	0	If a fault occurs in the device, more information about the fault can be obtained with this output variable. For details, see 18.4.9.17 SDO Fault Codes.
	szDataRead	Length of data read	_UXINT	0 to 255	0	Length of read data, unit: bytes

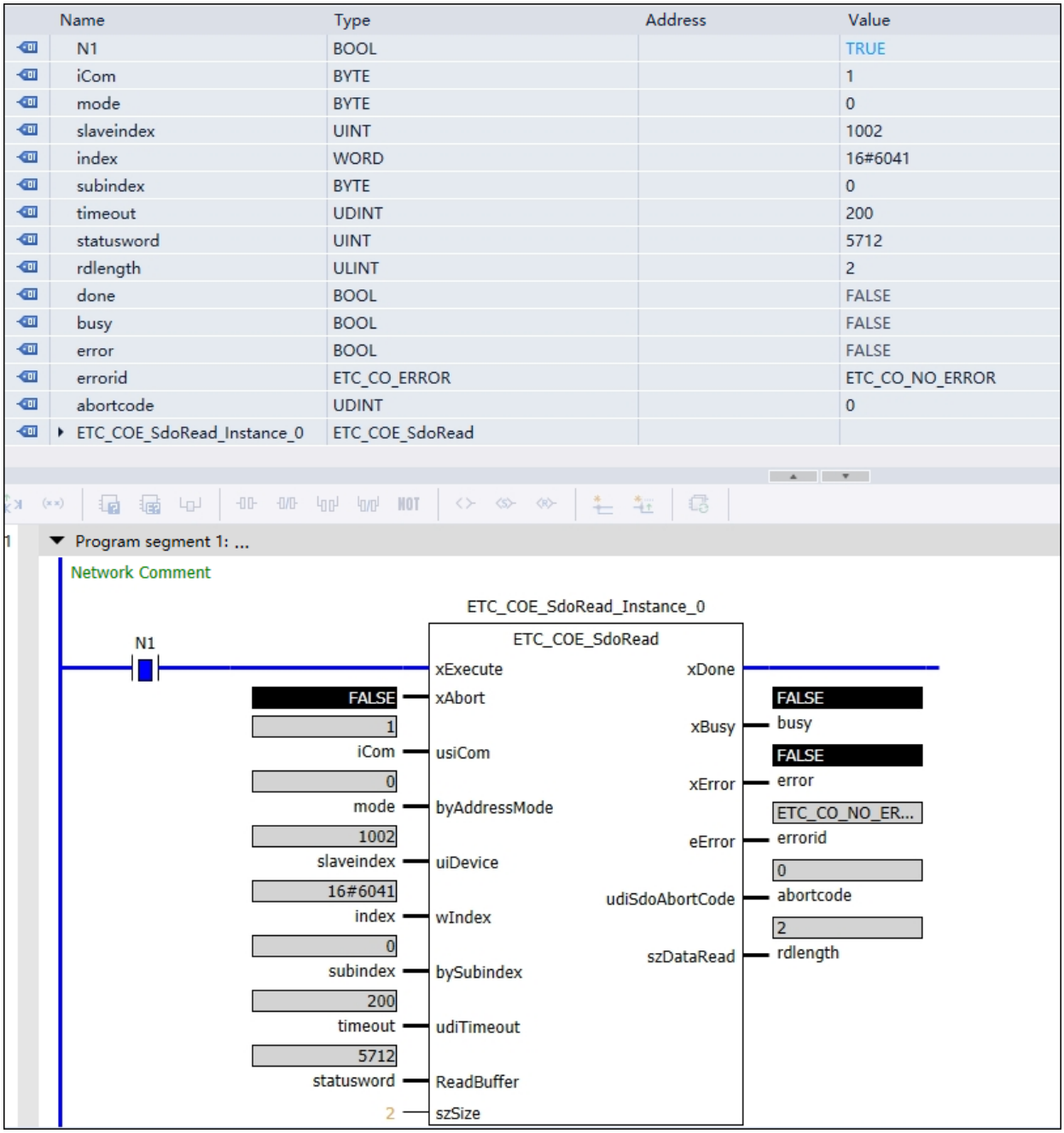
- **Function**
  This instruction reads parameters from entries in the object dictionary of EtherCAT slave and supports reading object data of any length.
- **Program example**
  - In sequential mode, read the status word of the SV660N (sequential address 1002) with index 16#6041, sub-index 16#00, and data length of 2 bytes.

    ST:

Name	Type	Address	Value
ETC_COE_SdoRead_Instance_0	ETC_COE_SdoRead		
N1	BOOL		TRUE
iCom	BYTE		1
mode	BYTE		0
slaveindex	UINT		1002
index	WORD		16#6041
subindex	BYTE		0
timeout	UDINT		200
statusword	UINT		16#1650
rdlength	ULINT		2
done	BOOL		TRUE
busy	BOOL		FALSE
error	BOOL		FALSE
errorid	ETC_CO_ERROR		ETC_CO_NO_ERROR
abortcode	UDINT		0

```
AS  FOR  WHI  REP  RET  EX  REG  //  /*
ETC_COE_SdoRead_Instance_0(xExecute TRUE := N1 TRUE,
                           xAbort := ,
                           usiCom 1 := iCom 1 ,
                           byAddressMode 0 := mode 0 ,
                           uiDevice 1002 := slaveindex 1002 ,
                           wIndex 16#6041 := index 16#6041 ,
                           bySubindex 0 := subindex 0 ,
                           udiTimeout 200 := timeout 200 ,
                           ReadBuffer := statusword 16#1650 ,
                           szSize 2 := 2,
                           xDone TRUE => done TRUE ,
                           xBusy FALSE => busy FALSE ,
                           xError FALSE => error FALSE ,
                           eErrorETC_CO_NO_ER... => erroridETC_CO_NO_ER...,
                           udiSdoAbortCode 0 => abortcode 0 ,
                           szDataRead 2 => rdlength 2 );RETURN
```

LD:



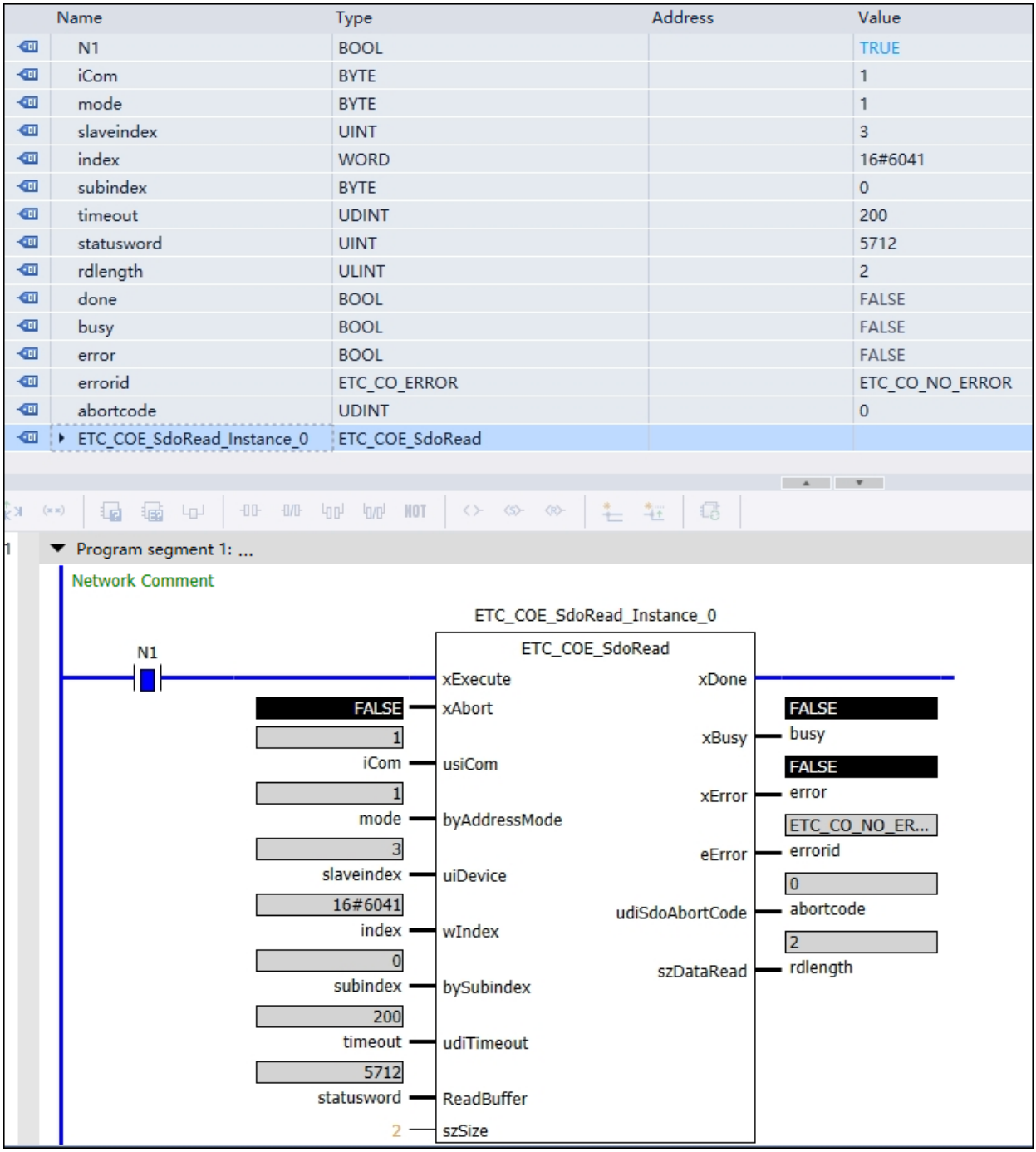
◦ In alias mode, read the status word of the SV660N (alias address 3) with index 16#6041, sub-index 16#00, and data length of 2 bytes.

ST:

Name	Type	Address	Value
ETC_COE_SdoRead_Instance_0	ETC_COE_SdoRead		
N1	BOOL		TRUE
iCom	BYTE		1
mode	BYTE		1
slaveindex	UINT		3
index	WORD		16#6041
subindex	BYTE		0
timeout	UDINT		200
statusword	UINT		5712
rdlength	ULINT		2
done	BOOL		TRUE
busy	BOOL		FALSE
error	BOOL		FALSE
errorid	ETC_CO_ERROR		ETC_CO_NO_ERROR
abortcode	UDINT		0

```
ETC_COE_SdoRead_Instance_0(xExecute TRUE := N1 TRUE,
                           xAbort := ,
                           usiCom 1 := iCom 1 ,
                           byAddressMode 1 := mode 1 ,
                           uiDevice 3 := slaveindex 3 ,
                           wIndex 24641 := index 16#6041 ,
                           bySubindex 0 := subindex 0 ,
                           udiTimeout 200 := timeout 200 ,
                           ReadBuffer := statusword 5712 ,
                           szSize 2 := 2,
                           xDone TRUE => done TRUE ,
                           xBusy FALSE => busy FALSE ,
                           xError FALSE => error FALSE ,
                           eErrorETC_CO_NO_ER... => erroridETC_CO_NO_ER...,
                           udiSdoAbortCode 0 => abortcode 0 ,
                           szDataRead 2 => rdlength 2 );RETURN
```

LD:



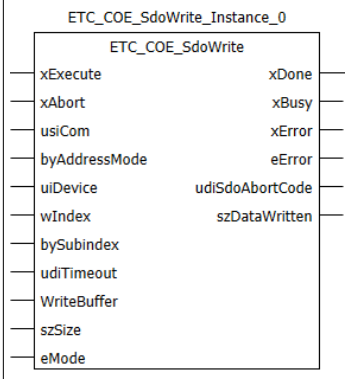
- **Precautions**
  - usiCom: EtherCAT master selection
    - When only one EtherCAT master is used for a device, usiCom is set to 1.
    - When two EtherCAT masters are used for a device:
      - When usiCom is set to 1, it indicates that the slave is assigned to the first EtherCAT master.
      - When usiCom is set to 2, it indicates that the slave is assigned to the second EtherCAT master.
  - When byAddressMode is set to 1, alias mode must be enabled to access the slave SDO. Otherwise, the error ETC_CO_
    FIRST_ERROR is reported.
  - ReadBuffer supports input of non-array elements, array names, and indexed array elements. The address operator ADR is not required.
  - szSize must be less than or equal to the available size of ReadBuffer to prevent memory overflow. Otherwise, the parameter validation fails and the error ETC_
    CO_DATA_OVERFLOW is reported.

- ReadBuffer supports data types of BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, and LINT, as well as arrays or array elements of these types.

18.4.9.5 ETC_COE_SdoWrite

This instruction writes parameters to entries in the object dictionary of EtherCAT slave and supports writing object data of any length.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
ETC_COE_SdoWrite	Write parameters to EtherCAT slave	FB		ETC_COE_SdoWrite_Instance_0(xExecute := , xAbort := , usiCom := , byAddressMode := , uiDevice := , wIndex := , bySubindex := , udiTimeout := , WriteBuffer := , szSize := , eMode := , xDone => , xBusy => , xError => , eError => , udiSdoAbortCode => , szDataWritten =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Function block execution	BOOL	TRUE, FALSE	FALSE	In the input rising edge, write slave parameters.
	xAbort	Function block execution aborted	BOOL	TRUE, FALSE	FALSE	If it is set to "TRUE", the write process is aborted.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	usiCom	Number of EtherCAT masters	BYTE	1 to 2	1	• When only one EtherCAT master is used, the value is 1. • When multiple masters are used, the value is 1 for the first master, 2 for the second master, and so on.
	byAddress Mode	Slave addressing mode	BYTE	0 or 1	0	• 0: Sequential mode • 1: Alias mode
	uiDevice	Physical address of the slave	UINT	0 to 65535	0	• In sequential mode, values are auto-assigned such as 1001, 1002. • In alias mode, values are user-defined such as 1, 2.
	wIndex	Object dictionary index	WORD	0 to 65535	0	Example: Object dictionary index 16#6061 of the servo running mode
	bySubindex	Object dictionary subindex	BYTE	0 to 255	0	Example: Object dictionary sub-index 16#00 of the servo homing mode
	udiTimeout	Timeout period	UDINT	0 to 65535	0	During the period (unit: ms), if the request to write parameters of the slave is not replied, an error message is displayed.
	WriteBuffer	Start address of data to write	ANY	-	-	Start address of data to write
	szSize	Write length	_UXINT	0 to 255	0	Written data length

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	eMode	Write mode selection	ETC_CO_MODE	-	0	• 0: ETC_CO_AUTO (automatic) • 1: ETC_CO_EXPEDITED (fast transmission) • 2: ETC_CO_SEGMENTED (segmented transmission)
Output	xDone	Parameter write completed	BOOL	TRUE, FALSE	FALSE	The value is "TRUE" when parameter write is completed.
	xBusy	Parameter write completion	BOOL	TRUE, FALSE	FALSE	The value is "TRUE" if the parameter write is not completed.
	xError	Parameter write error	BOOL	TRUE, FALSE	FALSE	The value is "TRUE" if an error occurs, and eError will indicate the cause.
	eError	Fault ID	ETC_CO_ERROR	-	ETC_CO_NO_ERROR	For details, see 18.4.9.16 ETC_CO_ERROR: ETC Fault Codes.
	udiSdoAbortCode	SDO fault code	UDINT	-	0	If a fault occurs in the device, more information about the fault can be obtained with this output variable. For details, see 18.4.9.17 SDO Fault Codes.
	szDataWritten	Length of write data	_UXINT	0 to 255	0	Length of written data

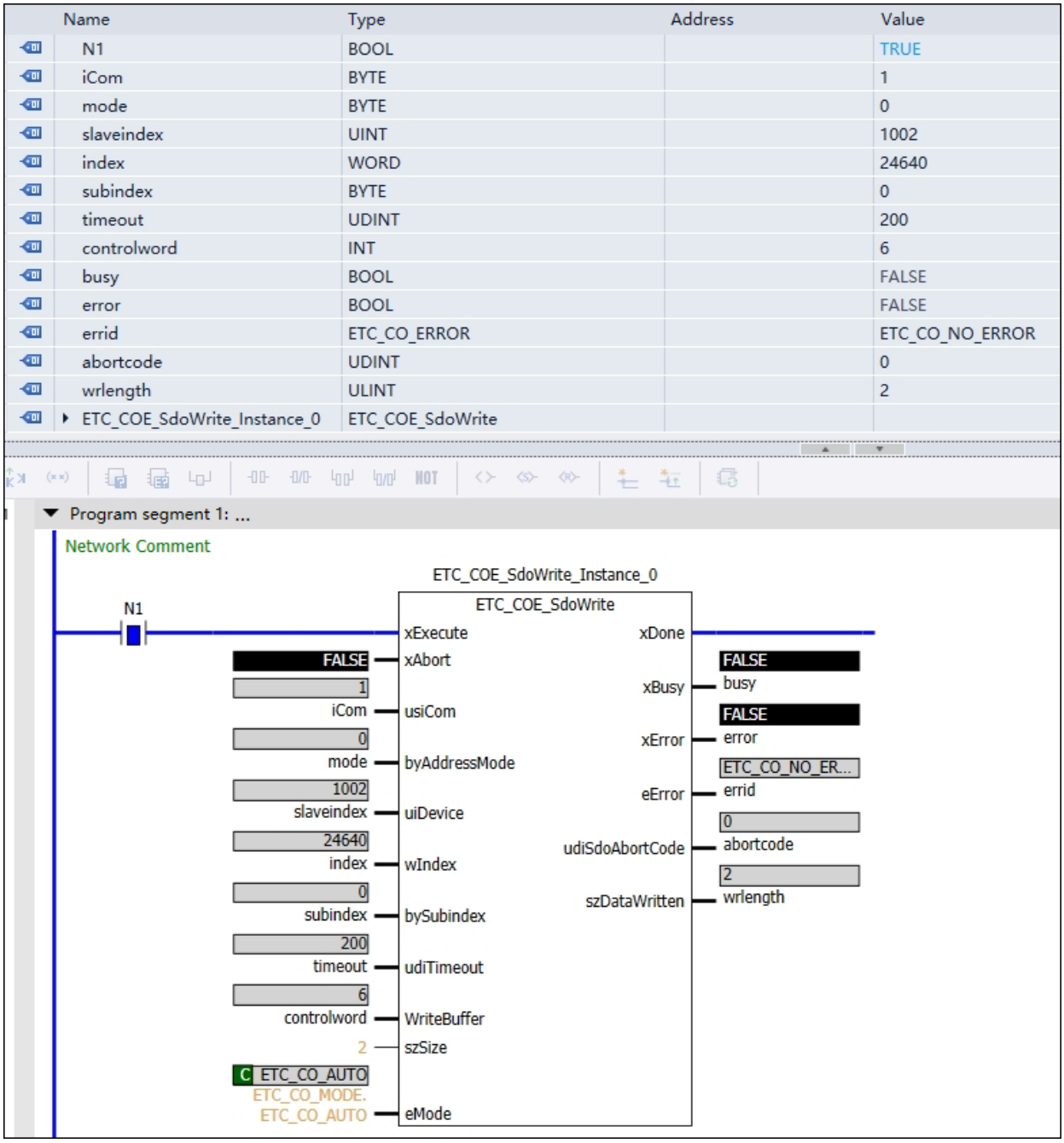
- **Function**
  This instruction writes parameters to entries in the object dictionary of EtherCAT slave and supports writing object data of any length.
- **Program example**
  - In sequential mode, write the control word of the SV660N (sequential address 1002) with index 16#6041, sub-index 16#00, and data length of 2 bytes.

    ST:

Name	Type	Address	Value
ETC_COE_SdoWrite_Instance_0	ETC_COE_SdoWrite		
N1	BOOL		TRUE
iCom	BYTE		1
mode	BYTE		0
slaveindex	UINT		1002
index	WORD		16#6040
subindex	BYTE		0
timeout	UDINT		200
controlword	INT		6
done	BOOL		TRUE
busy	BOOL		FALSE
error	BOOL		FALSE
errid	ETC_CO_ERROR		ETC_CO_NO_ERROR
abortcode	UDINT		0
wrlength	ULINT		2

```
CAS FOR WHI REP RET EX REG
ETC_COE_SdoWrite_Instance_0(xExecute TRUE := N1 TRUE,
                            xAbort := ,
                            usiCom 1 := iCom 1 ,
                            byAddressMode 0 := mode 0 ,
                            uiDevice 1002 := slaveindex 1002 ,
                            wIndex 16#6040 := index 16#6040 ,
                            bySubindex 0 := subindex 0 ,
                            udiTimeout 200 := timeout 200 ,
                            WriteBuffer := controlword 6 ,
                            szSize 2 := 2,
                            eModeETC_CO_A... := ETC_CO_MODE.ETC_CO_AUTO,
                            xDone TRUE => done TRUE ,
                            xBusy FALSE => busy FALSE ,
                            xError FALSE => error FALSE ,
                            eErrorETC_CO_NO_ER... => errIdETC_CO_NO_ER...,
                            udiSdoAbortCode 0 => abortcode 0 ,
                            szDataWritten 2 => wrlength 2 );RETURN
```

LD:



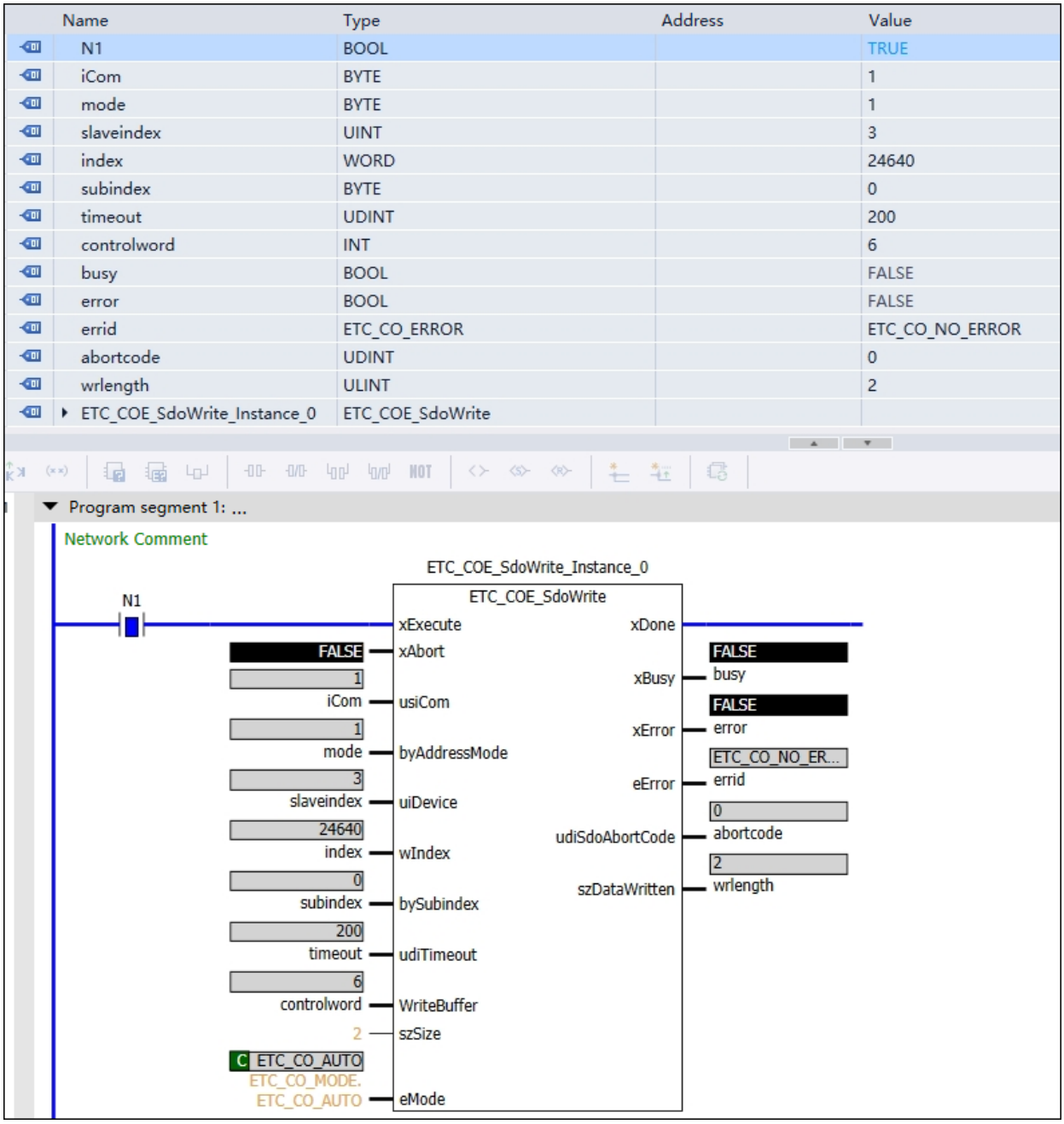
◦ In alias mode, write the control word of the SV660N (alias address 3) with index 16#6041, sub-index 16#00, and data length of 2 bytes.

ST:

Name	Type	Address	Value
ETC_COE_SdoWrite_Instance_0	ETC_COE_SdoWrite		
N1	BOOL		TRUE
iCom	BYTE		1
mode	BYTE		1
slaveindex	UINT		3
index	WORD		24640
subindex	BYTE		0
timeout	UDINT		200
controlword	INT		6
done	BOOL		TRUE
busy	BOOL		FALSE
error	BOOL		FALSE
errid	ETC_CO_ERROR		ETC_CO_NO_ERROR
abortcode	UDINT		0
wrlength	ULINT		2

```
AS FOR WHI REP RET EX REG
ETC_COE_SdoWrite_Instance_0(xExecute TRUE := N1 TRUE ,
                            xAbort := ,
                            usiCom 1 := iCom 1 ,
                            byAddressMode 1 := mode 1 ,
                            uiDevice 3 := slaveindex 3 ,
                            wIndex 24640 := index 24640 ,
                            bySubindex 0 := subindex 0 ,
                            udiTimeout 200 := timeout 200 ,
                            WriteBuffer := controlword 6 ,
                            szSize 2 := 2,
                            eModeETC_CO_A... := ETC_CO_MODE.ETC_CO_AUTO,
                            xDone TRUE => done TRUE ,
                            xBusy FALSE => busy FALSE ,
                            xError FALSE => error FALSE ,
                            eErrorETC_CO_NO_ER... => errIdETC_CO_NO_ER...,
                            udiSdoAbortCode 0 => abortcode 0 ,
                            szDataWritten 2 => wrlength 2 );RETURN
```

LD:



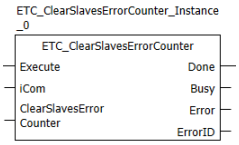
- **Precautions**
  - usiCom: EtherCAT master selection
    - When only one EtherCAT master is used for a device, usiCom is set to 1.
    - When two EtherCAT masters are used for a device:
      - When usiCom is set to 1, it indicates that the slave is assigned to the first EtherCAT master.
      - When usiCom is set to 2, it indicates that the slave is assigned to the second EtherCAT master.
  - When byAddressMode is set to 1, alias mode must be enabled to access the slave SDO. Otherwise, the error ETC_CO
    _FIRST_ERROR is reported.
  - WriteBuffer supports input of non-array elements, array names, and indexed array elements. The address operator ADR is not required.
  - szSize must be less than or equal to the available size of WriteBuffer to prevent memory overflow. Otherwise, the parameter validation fails and
    the error ETC_CO_DATA_OVERFLOW is reported.

◦ WriteBuffer supports data types of BYTE, WORD, DWORD, LWORD, USINT, UINT, UDINT, ULINT, SINT, INT, DINT, and LINT, as well as arrays or array elements of these types.

18.4.9.6 ETC_ClearSlavesErrorCounter

This instruction clears the error frame count of online EtherCAT slave.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ETC_ClearSlavesErrorCounter	Clear EtherCAT slave error frame count	FB		ETC_ClearSlavesErrorCounter_Instance_0(Execute := , iCom := , ClearSlavesErrorCounter := , Done => , Busy => , Error => , ErrorID =>);

- **Related variables**

Variable

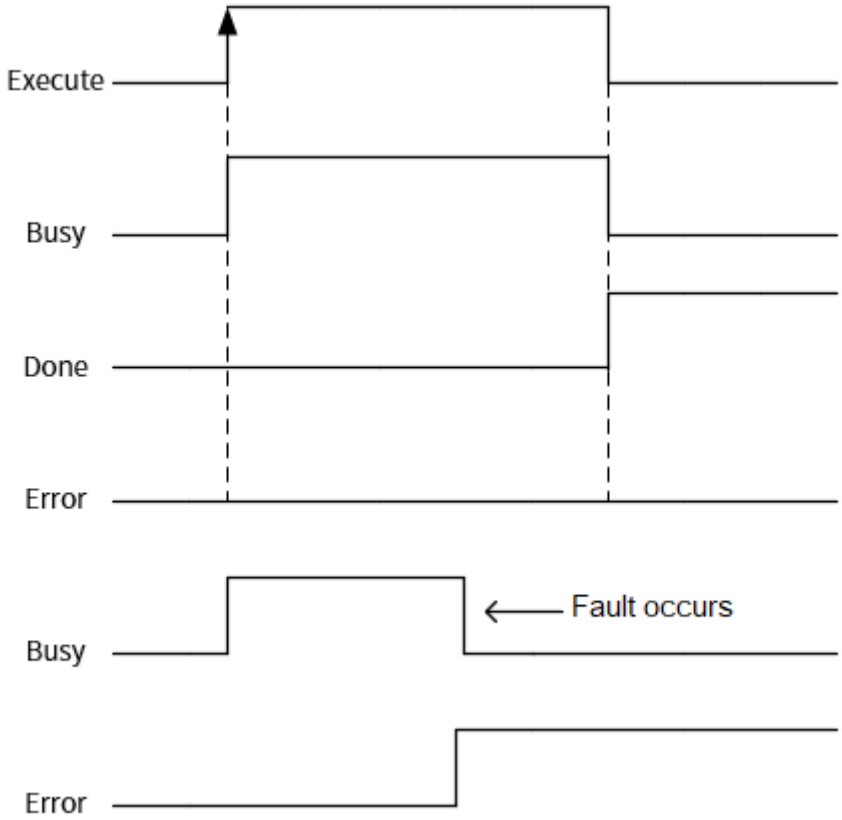
Input / Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Function block execution	BOOL	TRUE, FALSE	FALSE	Rising edge triggered.
	iCom	Number of EtherCAT masters	BYTE	Follows data type	1	Number of EtherCAT masters: If only one EtherCAT master is used, iCom is '1'. When multiple masters are used, the value is 1 for the first master, 2 for the second master, and so on.
	ClearSlavesErrorCounter	Enable clearing of error frame count	BOOL	TRUE, FALSE	FALSE	TRUE: Start clearing the error frame count of all online slave stations.
Output	Done	FB execution done	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the function block execution is done.
	Busy	Function block executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE while the function block is executing.

Input / Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
	ErrorID	Fault code	ETC_STD_ERROR	-	0	See 18.4.9.15 ETC_STD_ERROR: ETC Fault Codes.

- **Function**

  This instruction clears the error frame count of online EtherCAT slave.

- **Sequence diagram**



- **Program example**

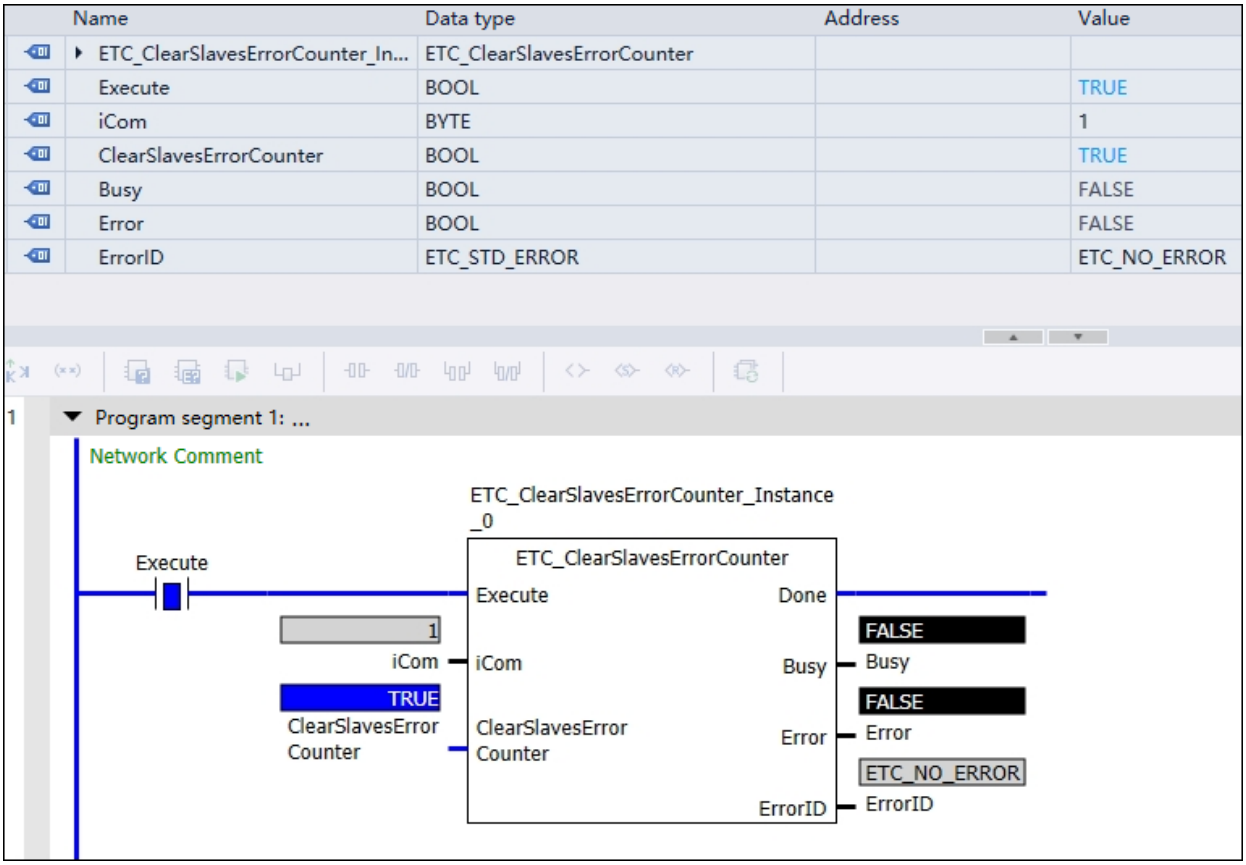
  ST:

	Name	Data type	Address	Value
	ETC_ClearSlavesErrorCount...	ETC_ClearSlavesErrorCounter		
	execute	BOOL		TRUE
	icom	BYTE		1
	clearflag	BOOL		TRUE
	done	BOOL		TRUE
	busy	BOOL		FALSE
	error	BOOL		FALSE
	errorid	ETC_STD_ERROR		ETC_NO_ERROR

```
ETC_ClearSlavesErrorCounter_Instance_0(
Execute TRUE := execute TRUE ,
iCom 1 := icom 1 ,
ClearSlavesErrorCounter TRUE := clearflag TRUE ,
Done TRUE => done TRUE ,
Busy FALSE => busy FALSE ,
Error FALSE => error FALSE ,
ErrorIDETC_NO_ER... => erroridETC_NO_ER...);RETURN
```

LD:

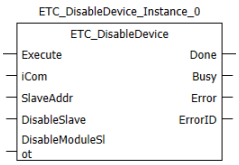
	Name	Data type	Address	Value
	ETC_ClearSlavesErrorCounter_In...	ETC_ClearSlavesErrorCounter		
	Execute	BOOL		TRUE
	iCom	BYTE		1
	ClearSlavesErrorCounter	BOOL		TRUE
	Busy	BOOL		FALSE
	Error	BOOL		FALSE
	ErrorID	ETC_STD_ERROR		ETC_NO_ERROR



18.4.9.7 ETC_DisableDevice

This instruction disables EtherCAT devices, and supports disabling slave devices as well as coupler modules separately.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ETC_DisableDevice	Disable EtherCAT device	FB		ETC_DisableDevice_Instance_0(Execute := , iCom := , SlaveAddr := , DisableSlave := , DisableModuleSlot := , Done => , Busy => , Error => , ErrorID =>);

- **Related variables**

  Variable

Input / Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Function block execution	BOOL	TRUE, FALSE	FALSE	Rising edge triggered.
	iCom	Number of EtherCAT masters	BYTE	Follows data type	1	Number of EtherCAT masters: If only one EtherCAT master is used, iCom is '1'. When multiple masters are used, the value is 1 for the first master, 2 for the second master, and so on.
	SlaveAddr	Slave Address	UINT	1 to 65535	0	Slave address.
	DisableSlave	Slave disable	BOOL	TRUE, FALSE	FALSE	FALSE: Disable the slave disable function and enable the module disable function. TRUE: Enable the slave disable function and disable the module disable function.
	DisableModuleSlot	Module slot disabling	LWORD	Follows data type	0	The slot length is 8 bytes (64 bits) and one bit represents one slot. Set to 1 if the slot is disabled. The module close to the coupler corresponds to the lower data bits.
Output	Done	FB execution done	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the function block execution is done.
	Busy	Function block executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE while the function block is executing.

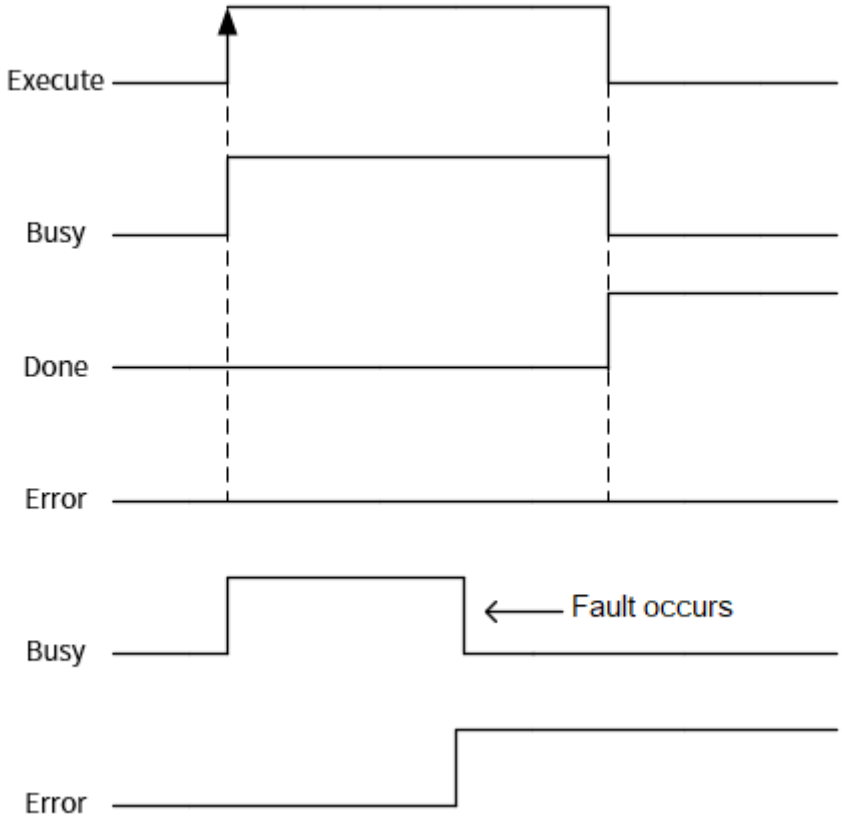
Input / Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
	ErrorID	Fault code	ETC_STD_ERROR	-	0	See 18.4.9.15 ETC_STD_ERROR: ETC Fault Codes.

- **Function**

  This instruction disables EtherCAT slave and coupler module.

> NOTE
>
> The EVO500/700 series PLC only supports the module disable function from version V1.4 and later.

- **Sequence diagram**



- **Program example**

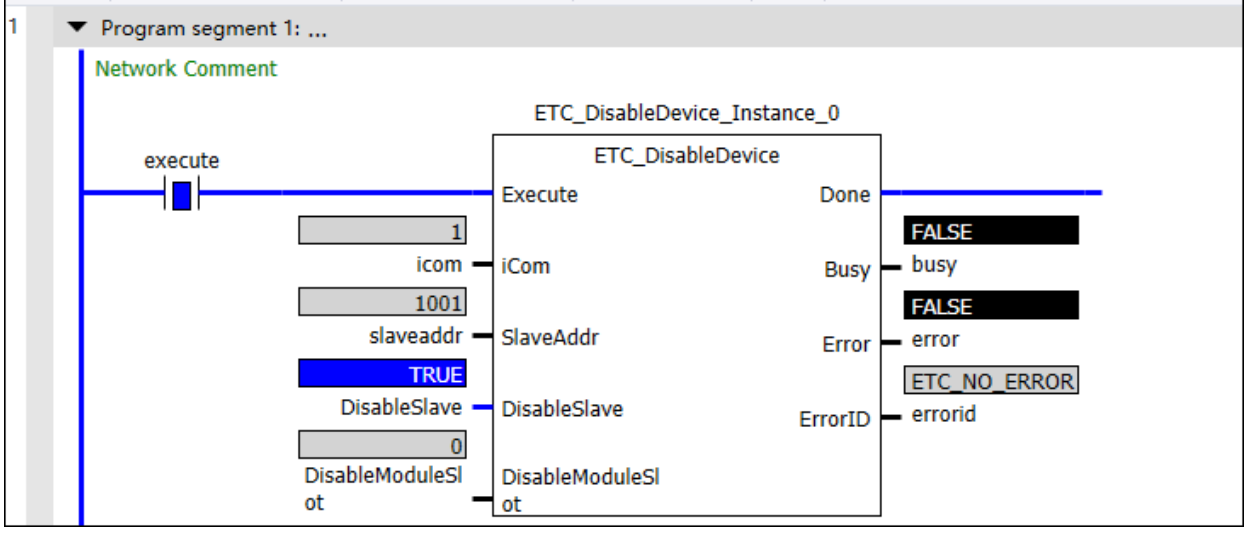
  ST:

Name	Data type	Address	Value
ETC_DisableDevice_Instance_0	ETC_DisableDevice		
execute	BOOL		TRUE
icom	BYTE		1
slaveaddr	UINT		1001
done	BOOL		TRUE
busy	BOOL		FALSE
error	BOOL		FALSE
errorid	ETC_STD_ERROR		ETC_NO_ERROR
DisableSlave	BOOL		TRUE
DisableModuleSlot	LWORD		0

```
ETC_DisableDevice_Instance_0(
Execute TRUE := execute TRUE ,
iCom 1 := icom 1 ,
SlaveAddr 1001 := slaveaddr 1001 ,
DisableSlave TRUE := DisableSlave TRUE ,
DisableModuleSlot 0 := DisableModuleSlot 0 ,
Done TRUE => done TRUE ,
Busy FALSE => busy FALSE ,
Error FALSE => error FALSE ,
ErrorIDETC_NO_ER... => erroridETC_NO_ER...);RETURN
```

LD:

Name	Data type	Address	Value
ETC_DisableDevice_Instance_0	ETC_DisableDevice		
execute	BOOL		TRUE
icom	BYTE		1
slaveaddr	UINT		1001
done	BOOL		FALSE
busy	BOOL		FALSE
error	BOOL		FALSE
errorid	ETC_STD_ERROR		ETC_NO_ERROR
DisableSlave	BOOL		TRUE
DisableModuleSlot	LWORD		0



- **Precautions**
  - Enable the slave disable function

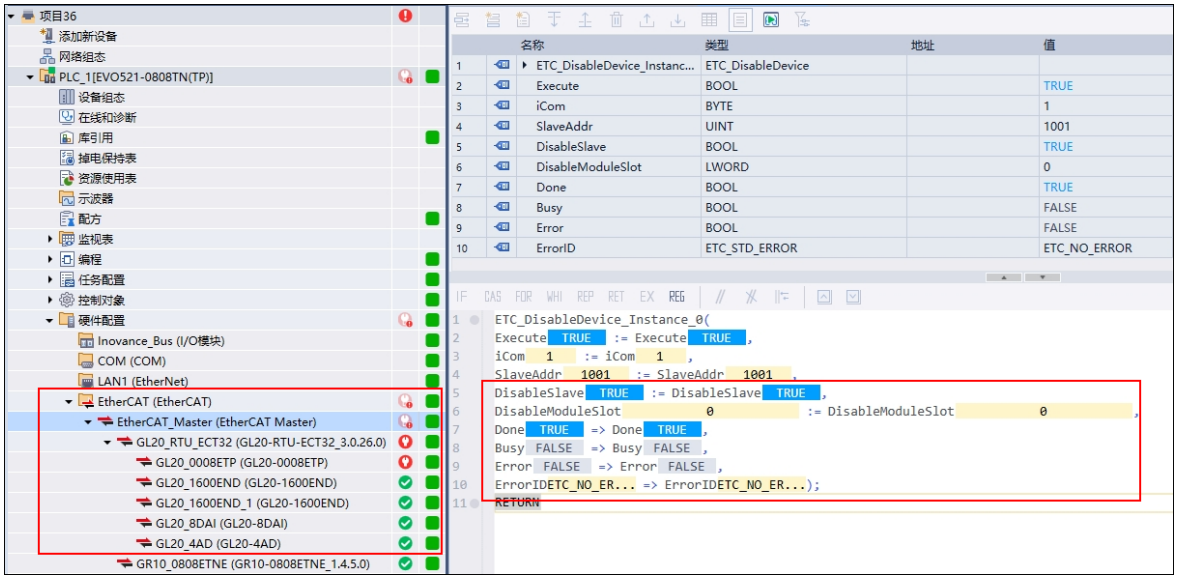
    After successful execution of this function block, DisableSlave is set to TRUE to enable the EtherCAT

slave disable function. The activation behavior of this instruction varies depending on the firmware version.

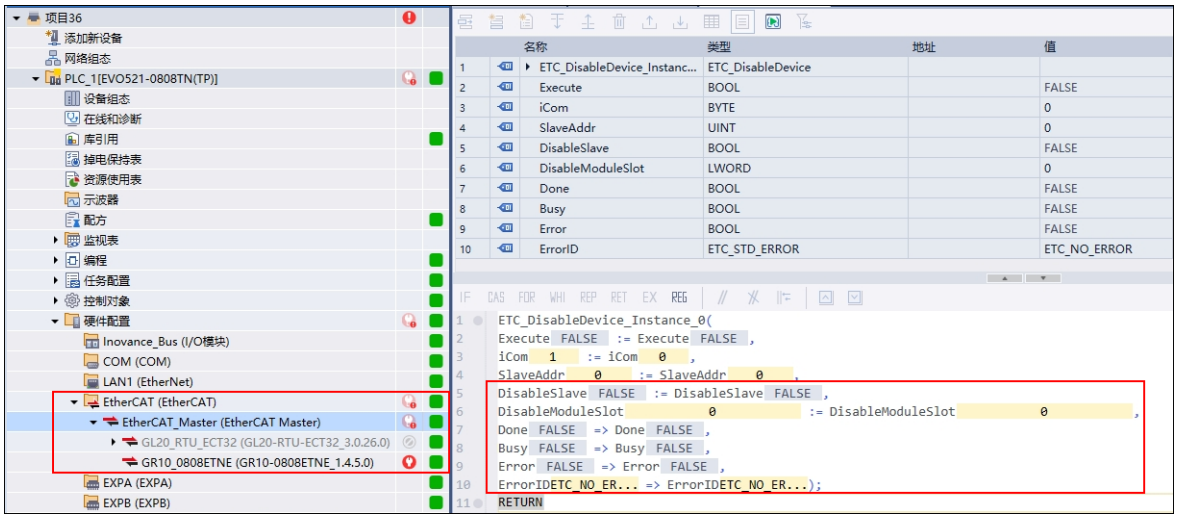
For devices with firmware higher than V1.4 (V1.4 excluded):

If the instruction is executed in the first scan cycle of the EtherCAT_Master_Task, the slave disable function takes effect immediately during the EtherCAT startup phase without EtherCAT master restart.

If the instruction is not executed in the first scan cycle of EtherCAT_Master_Task or is executed in other non-EtherCAT_Master_Task tasks, restart the master via ETC_MasterRestart for the disable function to take effect.

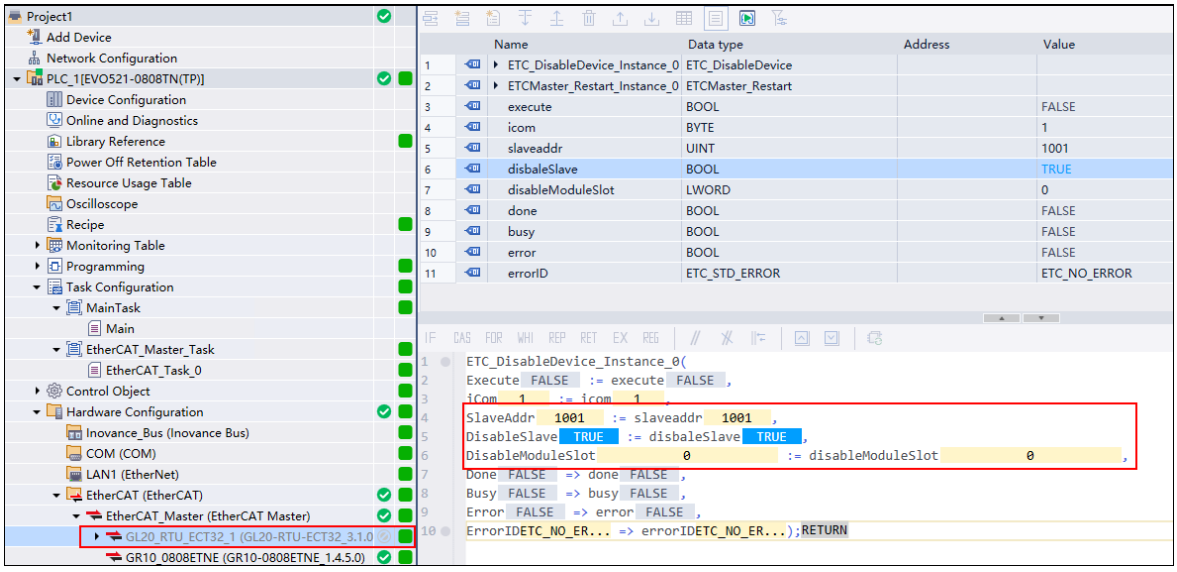


For devices with firmware lower than V1.4, after enabling the slave disable function, perform a PLC hot reset for the disable function to take effect. Click "Online > Hot reset" in the menu bar to perform reset, and then the disable function takes effect.



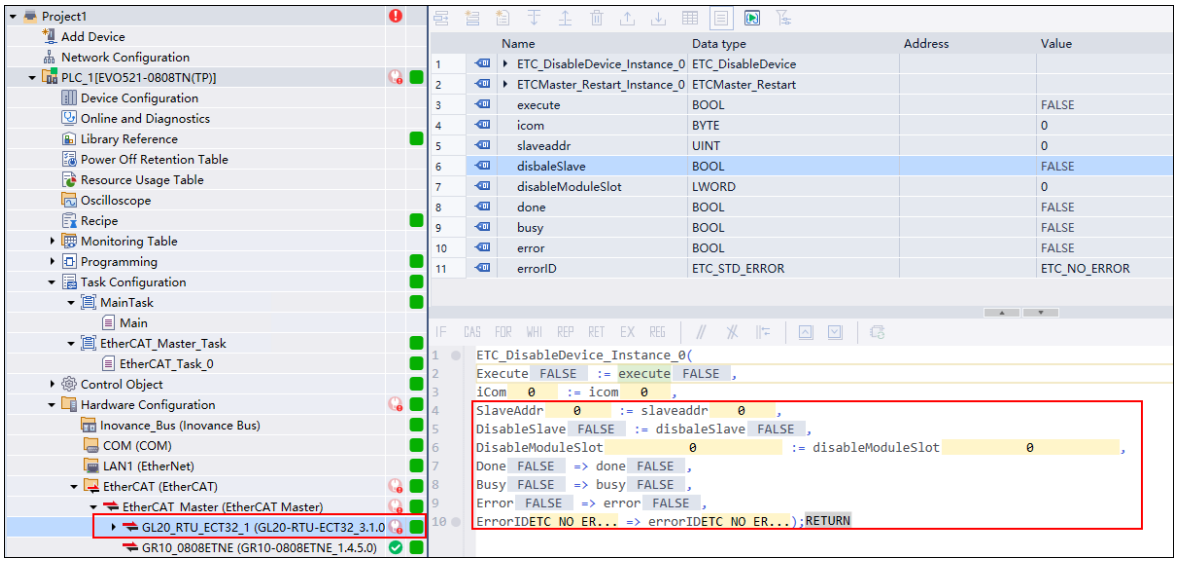
◦ Disable the salve disable function

To disable a slave, a falling edge of the disable instruction must be triggered. Set SlaveAddr to 1001 and DisableSlave to TRUE to enable the first slave. Trigger a falling edge of Execute with its value changing from FALSE to TRUE then back to FALSE.



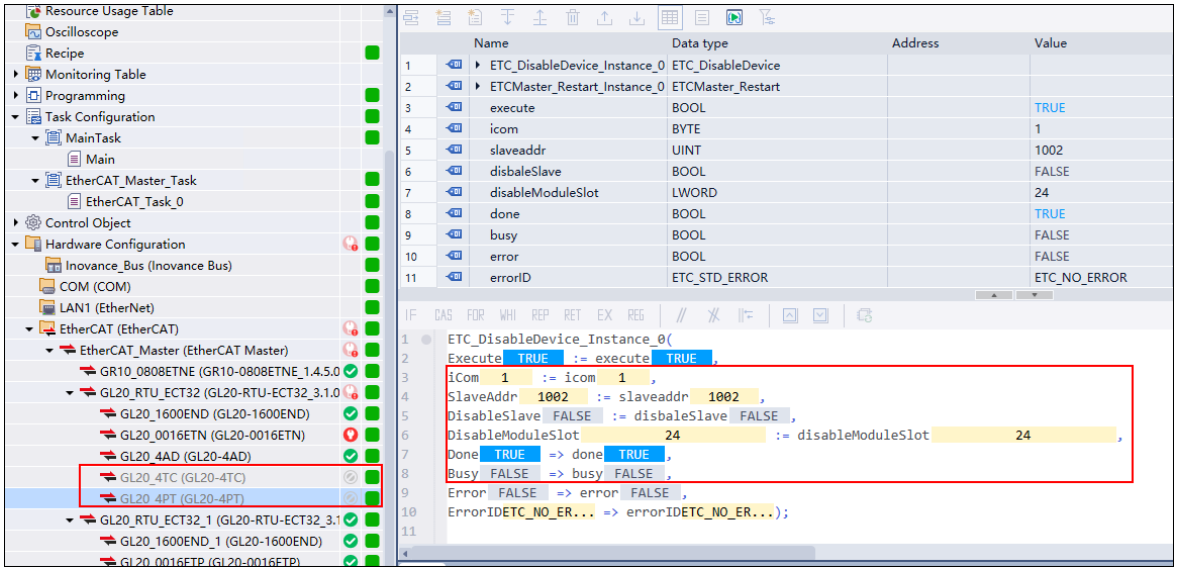
For devices with firmware higher than V1.4 (V1.4 excluded), after the slave disable instruction is disabled, restart the master via ETC_MasterRestart for the disable function to take effect.

For devices with firmware lower than V1.4, click "Online > Hot reset" in the menu bar to perform reset, and then the disable function is canceled.

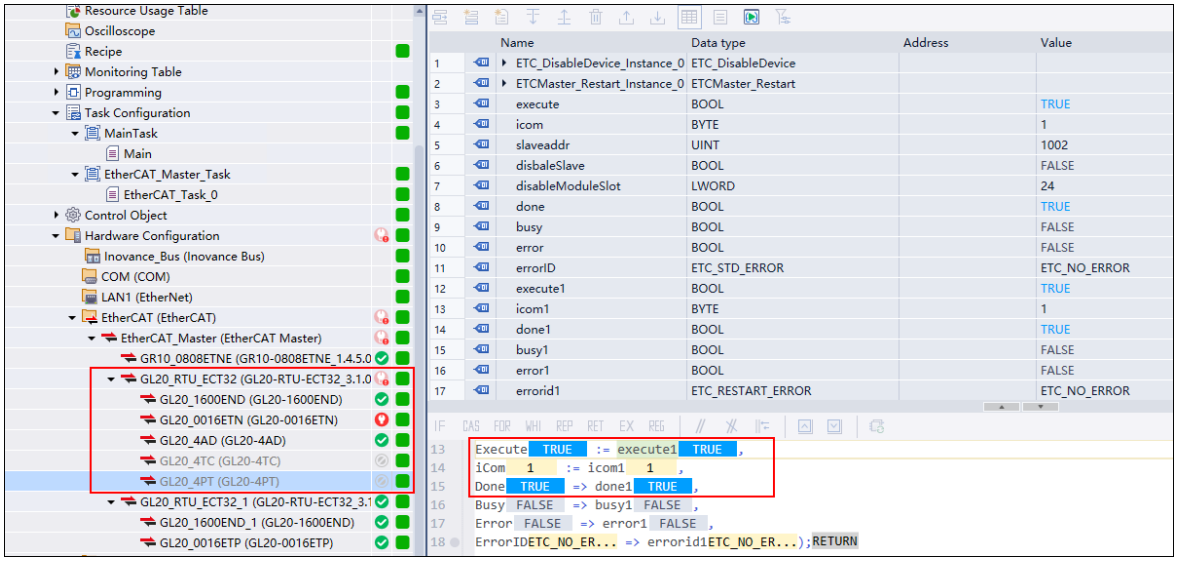


- Enable the module disable function

  Set SlaveAddr to the sequential address of the coupler slave and DisableSlave to FALSE. Assign target slot numbers to DisableModuleSlot where each bit corresponds to one slot. A value of 24 for DisableModuleSlot disables the fourth and fifth slots. Ensure that the instruction is triggered on a rising edge after the input parameters are configured.

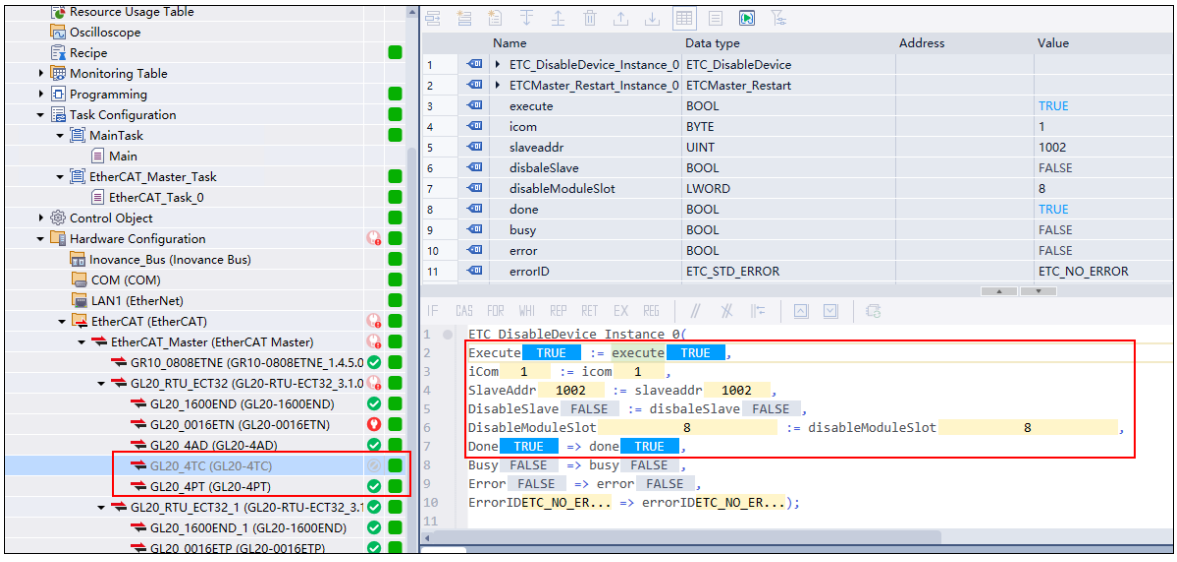


After the instruction execution, trigger the ECTMaster_Restart instruction and wait for the slave re-initialization.

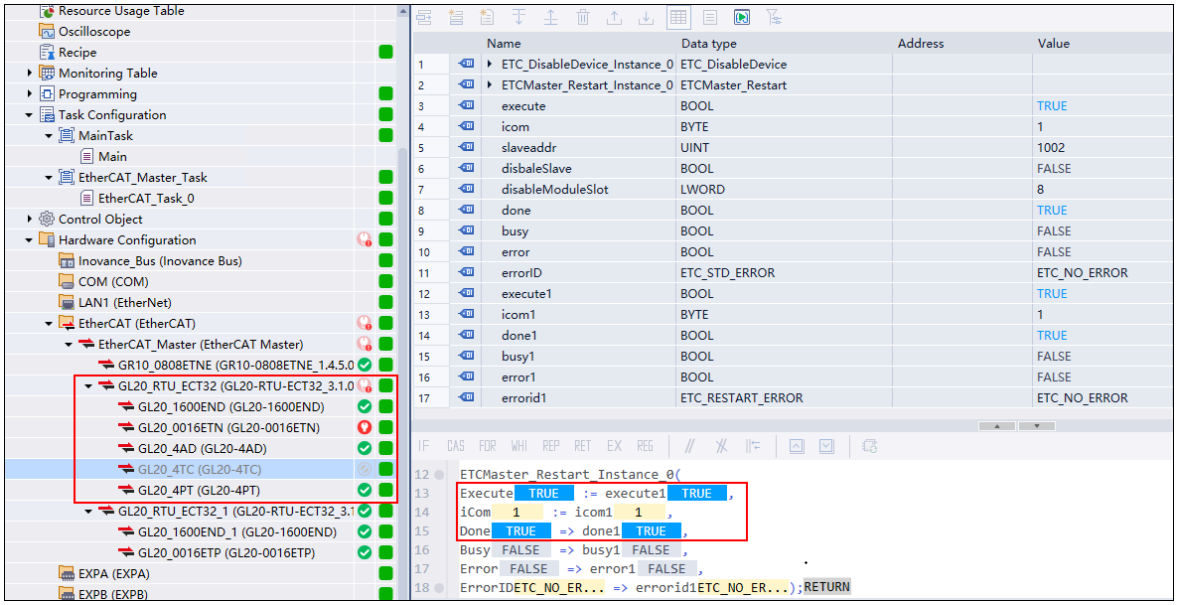


- Module slot disabling deactivation

  Input the sequential address of the coupler slave in Slaveaddr, set DisableSlave to FALSE, and input the slots to be deactivated in DisableModuleSlot (each bit represents one slot). Ensure that the instruction is triggered on a rising edge after the input parameters are configured.



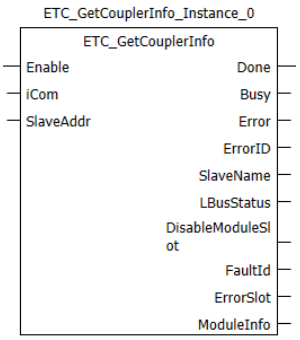
After the instruction execution, trigger the ECTMaster_Restart instruction and wait for the slave re-initialization.



18.4.9.8 ETC_GetCouplerInfo

This instruction reads information of the GL20-RTU-ECT slave and local modules in EtherCAT.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ETC_GetCouplerInfo	Read EtherCAT coupler information	FB		ETC_GetCouplerInfo_Instance_0(Enable := , iCom := , SlaveAddr := , Done => , Busy => , Error => , ErrorID => , SlaveName => , LBusStatus => , DisableModuleSlot => , FaultId => , ErrorSlot => , ModuleInfo =>);

- **Related variables**

Variable

Input / Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Enable	Function block execution	BOOL	TRUE, FALSE	FALSE	Active High.

Input / Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	iCom	Number of EtherCAT masters	BYTE	Follows data type	1	Number of EtherCAT masters: If only one EtherCAT master is used, iCom is '1'. When multiple masters are used, the value is 1 for the first master, 2 for the second master, and so on.
	SlaveAddr	Slave Address	UINT	1 to 65535	0	Slave address.
Output	Done	FB execution done	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the function block execution is done.
	Busy	Function block executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE while the function block is executing.
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
	ErrorID	Fault code	ETC_STD_ERROR	-	0	See 18.4.9.15 ETC_STD_ERROR: ETC Fault Codes.
	SlaveName	Slave name	STRING	Follows data type	-	Slave device name.
	LBusStatus	Local bus status of coupler	UINT	0 to 255	0	See the Local bus status of coupler table.
	DisableModuleSlot	Module disable slot	LWORD	Follows data type	0	The slot length is 8 bytes (64 bits) and one bit represents one slot. Set to 1 if the slot is disabled. The module close to the coupler corresponds to the lower data bits.
	FaultId	Configuration fault code	UINT	Follows data type	0	See the Configuration fault code of coupler table.
	ErrorSlot	Module fault slot	LWORD	Follows data type	0	The slot length is 8 bytes (64 bits) and one bit represents one slot. Set to 1 if there is a slot fault. The module close to the coupler corresponds to the lower data bits. (Output module fault location is subject to the network hardware topology.)

Input / Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	ModuleInfo	Module information	ARRAY [0..63] OF ETC_StructModuleInfo	-	-	See the Coupler module information.

- **Function**
  - Local bus status of coupler table

Status value	Description
0x01	Local bus initialization.
0x02	Local bus pre-operational status.
0x04	Local bus safe-operational status
0x08	Local bus operational status.
0x10	Switchover from safe-operational to operational status.
0x20	Switchover from operational status to safe-operational status.
0x41	DNM/2SCOM-MDB waiting for configuration parameter distribution.
0x80	Local bus error.

  - Configuration fault code of coupler table

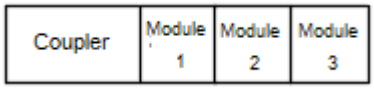
Fault ID	Description
0x0000	Configuration is consistent.
0x1000	No expansion module scanned.
0x1001	The number of configured I/O modules is greater than the number of actual scanned I/O modules.
0x1002	The number of configured I/O modules is less than the number of actual scanned I/O modules.
0x1003	The type of configured I/O modules is inconsistent with that of actual scanned I/O modules.

  - Coupler module information (ETC_StructModuleInfo structure)

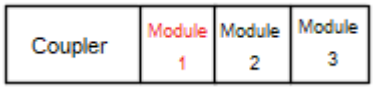
Variable	Data Type	Description
ModuleName	STRING	Module type.
SlotError	BOOL	Whether the module has failed, TRUE indicates a failure.
ModuleErrorCode	UINT	Module fault ID. For details, see 18.4.9.18 IO Fault Codes.

Variable	Data Type	Description
ChannelErrorCode	ARRAY [0..128] OF UINT	IO channel fault ID. The array sub-index corresponds to the channel number, see 18.4.9.18 IO Fault Codes.

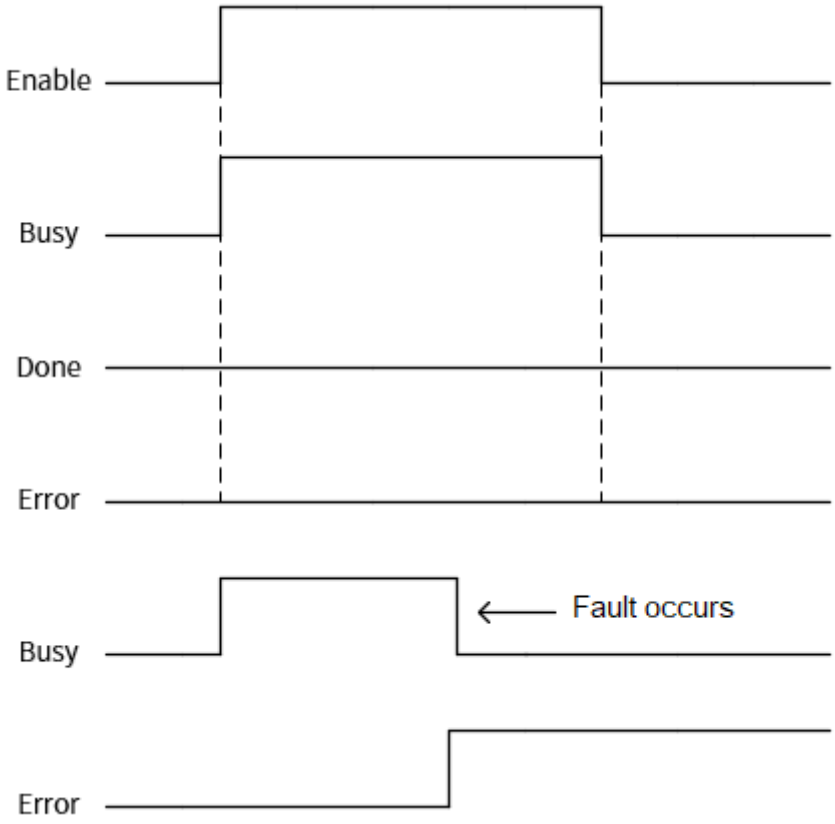
◦ Array sub-index corresponds to the actual connected module slot number. As shown in the figure below, three modules are actually connected: astModuleInfo[0] represents module 1, astModuleInfo[1] represents module 2, astModuleInfo[2] represents module 3.



If two modules are actually connected, module 1 is disabled. Then astModuleInfo[0] represents module 2 and astModuleInfo[1] represents module 3.



- **Sequence diagram**



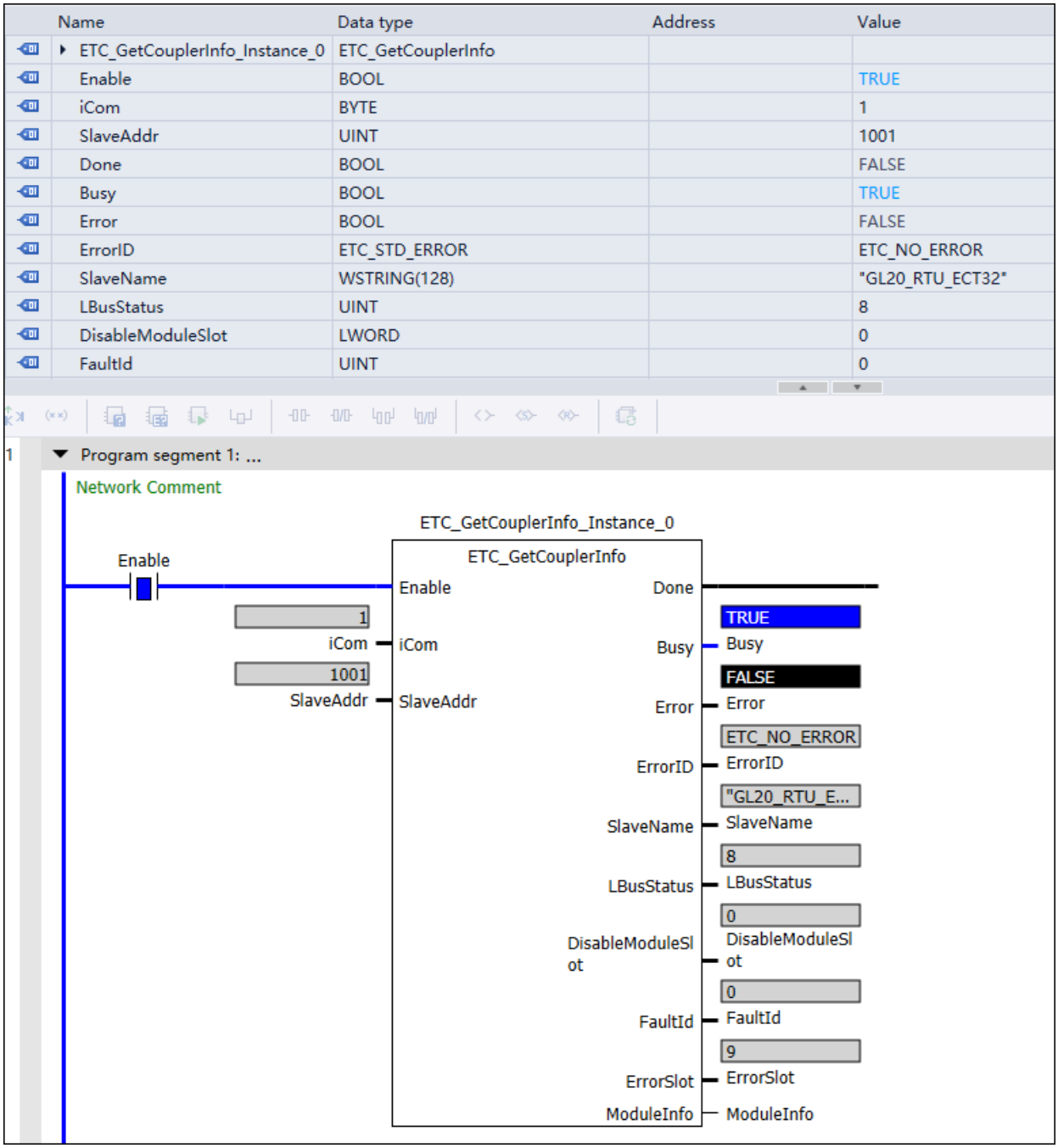
- **Program example**

ST:

	Name	Data type	Address	Value
	ETC_GetCouplerInfo_Instance_0	ETC_GetCouplerInfo		
	Enable	BOOL		TRUE
	iCom	BYTE		1
	SlaveAddr	UINT		1001
	Done	BOOL		FALSE
	Busy	BOOL		TRUE
	Error	BOOL		FALSE
	ErrorID	ETC_STD_ERROR		ETC_NO_ERROR
	SlaveName	WSTRING(128)		"GL20_RTU_ECT32"
	LBusStatus	UINT		8
	DisableModuleSlot	LWORD		0
	FaultId	UINT		0
	ErrorSlot	LWORD		9
	ModuleInfo	ARRAY [0..63] OF ETC_StructModuleIn...		

```
CAS FOR WHI REP RET EX REG
ETC_GetCouplerInfo_Instance_0(
Enable TRUE := Enable TRUE,
iCom 1 := iCom 1,
SlaveAddr 1001 := SlaveAddr 1001,
Done FALSE => Done FALSE,
Busy TRUE => Busy TRUE,
Error FALSE => Error FALSE,
ErrorIDETC_NO_ER... => ErrorIDETC_NO_ER...,
SlaveName"GL20_RTU_ECT... => SlaveName"GL20_RTU_ECT...,
LBusStatus 8 => LBusStatus 8,
DisableModuleSlot 0 => DisableModuleSlot 0,
FaultId 0 => FaultId 0,
ErrorSlot 9 => ErrorSlot 9,
ModuleInfo => ModuleInfo);RETURN
```

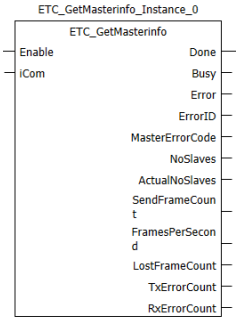
LD:



18.4.9.9 ETC_GetMasterinfo

This instruction gets EtherCAT master information.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ETC_GetMasterinfo	Get EtherCAT master information	FB		ETC_GetMasterinfo_Instance_0(Enable := , iCom := , Done => , Busy => , Error => , ErrorID => , MasterErrorCode => , NoSlaves => , ActualNoSlaves => , SendFrameCount => , FramesPerSecond => , LostFrameCount => , TxErrorCount => , RxErrorCount =>);

- **Related variables**

  Variable

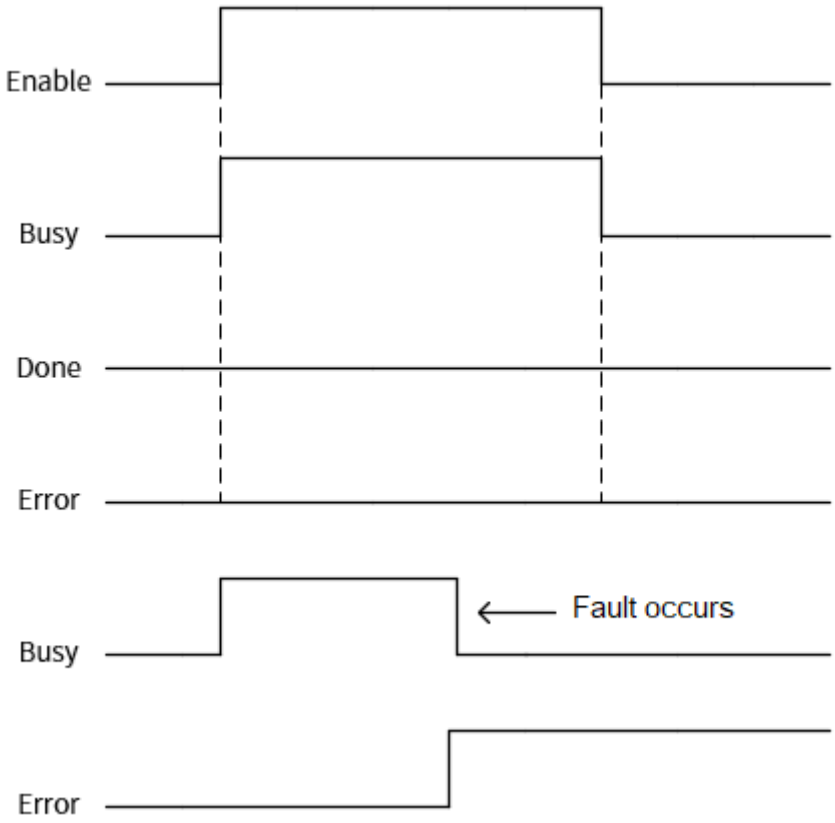
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Enable	Function block execution	BOOL	TRUE, FALSE	FALSE	Active High.
	iCom	Number of EtherCAT masters	BYTE	Follows data type	1	Number of EtherCAT masters: If only one EtherCAT master is used, iCom is '1'. When multiple masters are used, the value is 1 for the first master, 2 for the second master, and so on.
Output	Done	FB execution done	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the function block execution is done.
	Busy	Function block executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE while the function block is executing.
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
	ErrorID	Fault code	ETC_STD_ERROR	-	0	See 18.4.9.15 ETC_STD_ERROR: ETC Fault Codes.
	MasterErrorCode	Master fault code	WORD	Follow data type	0	Master fault code.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	NoSlaves	Number of configured slaves	UINT	Follows data type	0	Number of configured slaves.
	ActualNoSlaves	Number of online slaves	UINT	Follows data type	0	Number of online slaves.
	SendFrameCount	Number of frames sent by master	UDINT	Follows data type	0	Number of frames sent by master.
	FramesPerSecond	Master frame sending frequency	UDINT	Follows data type	0	Master frame sending frequency.
	LostFrameCount	Master station lost frame count	UDINT	Follows data type	0	Master station lost frame count.
	TxErrorCount	Master transmitted error frame count	UDINT	Follows data type	0	Master transmitted error frame count.
	RxErrorCount	Master received error frame count	UDINT	Follows data type	0	Master received error frame count.

- **Function**

  This instruction gets EtherCAT master information.

- **Sequence diagram**



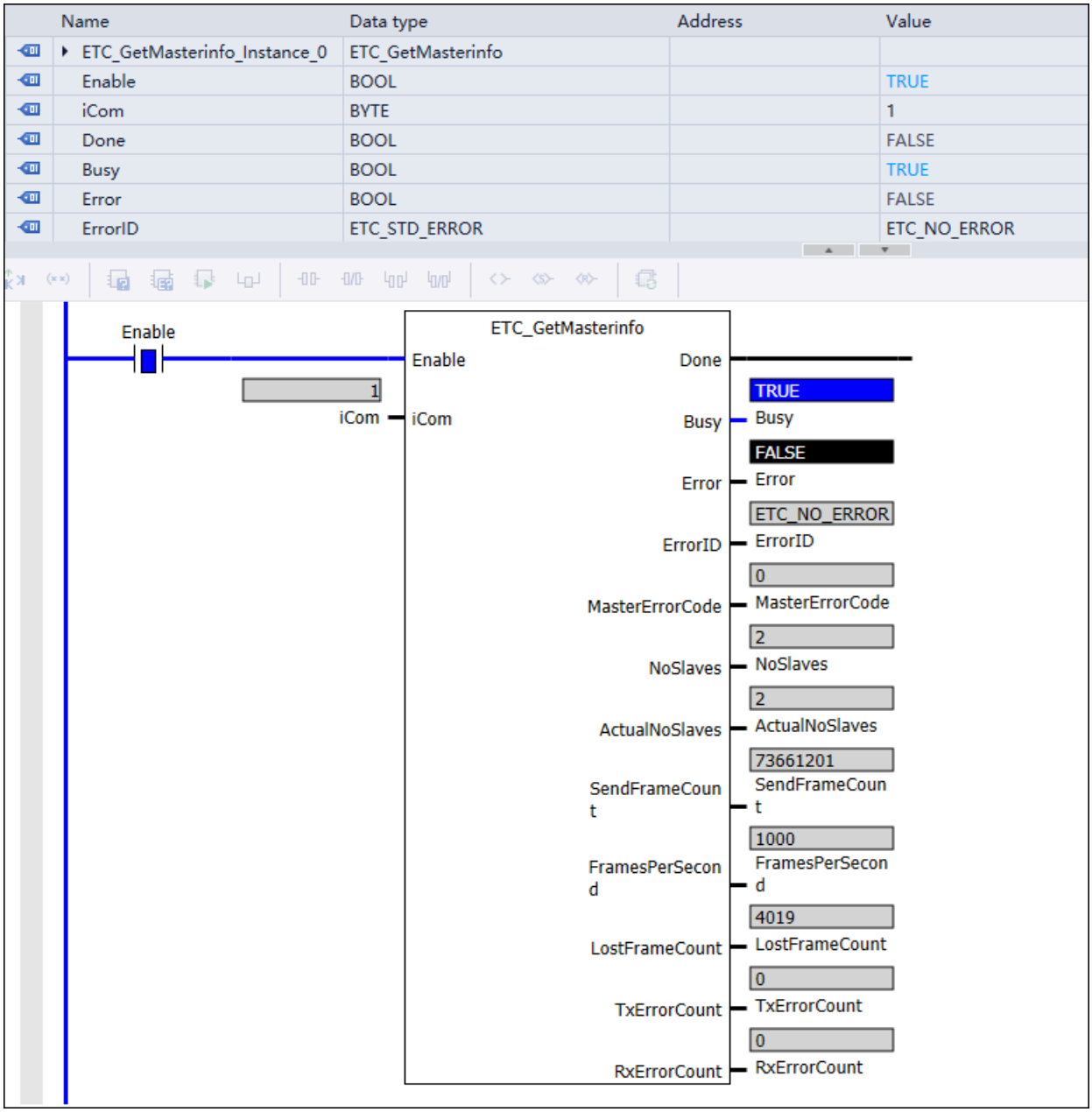
- **Program example**

  ST:

Name	Data type	Address	Value
ETC_GetMasterinfo_Instance_0	ETC_GetMasterinfo		
Enable	BOOL		TRUE
iCom	BYTE		1
Done	BOOL		FALSE
Busy	BOOL		TRUE
Error	BOOL		FALSE
ErrorID	ETC_STD_ERROR		ETC_NO_ERROR
MasterErrorCode	WORD		0
NoSlaves	UINT		2
ActualNoSlaves	UINT		2
SendFrameCount	UDINT		73493736
FramesPerSecond	UDINT		1000
LostFrameCount	UDINT		4019
TxErrorCount	UDINT		0
RxErrorCount	UDINT		0

```
CAS FOR WHI REP RET EX REG
ETC_GetMasterinfo_Instance_0(
Enable TRUE := Enable TRUE ,
iCom 1 := iCom 1 ,
Done FALSE => Done FALSE ,
Busy TRUE => Busy TRUE ,
Error FALSE => Error FALSE ,
ErrorIDETC_NO_ER... => ErrorIDETC_NO_ER...,
MasterErrorCode 0 => MasterErrorCode 0 ,
NoSlaves 2 => NoSlaves 2 ,
ActualNoSlaves 2 => ActualNoSlaves 2 ,
SendFrameCount 73506548 => SendFrameCount 73506548 ,
FramesPerSecond 1000 => FramesPerSecond 1000 ,
LostFrameCount 4019 => LostFrameCount 4019 ,
TxErrorCount 0 => TxErrorCount 0 ,
RxErrorCount 0 => RxErrorCount 0 );RETURN
```

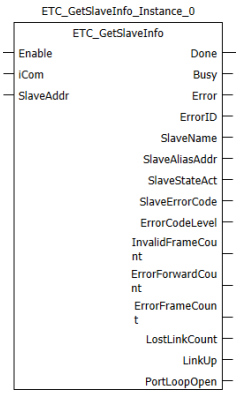
LD:



18.4.9.10 ETC_GetSlaveInfo

This instruction gets EtherCAT slave information.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ETC_GetSlaveInfo	Get EtherCAT slave information	FB		ETC_GetSlaveInfo_Instance_0(Enable := , iCom := , SlaveAddr := , Done => , Busy => , Error => , ErrorID => , SlaveName => , SlaveAliasAddr => , SlaveStateAct => , SlaveErrorCode => , ErrorCodeLevel => , InvalidFrameCount => , ErrorForwardCount => , ErrorFrameCount => , LostLinkCount => , LinkUp => , PortLoopOpen =>);

- **Related variables**

  Variable

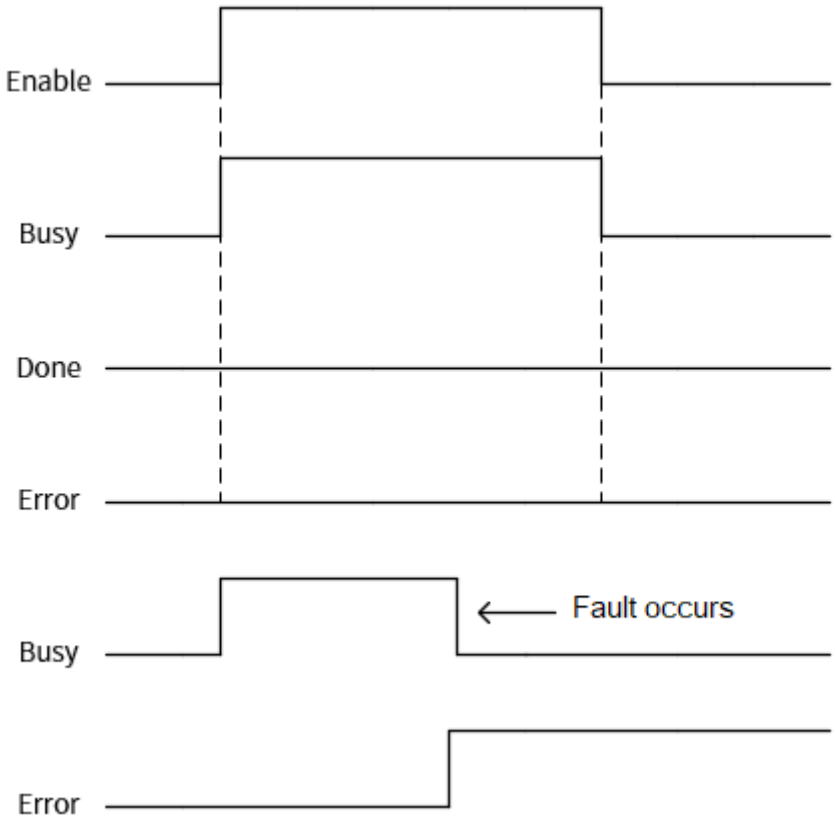
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Enable	Function block execution	BOOL	TRUE, FALSE	FALSE	Active High.
	iCom	Number of EtherCAT masters	BYTE	Follows data type	1	Number of EtherCAT masters: If only one EtherCAT master is used, iCom is '1'. When multiple masters are used, the value is 1 for the first master, 2 for the second master, and so on.
	SlaveAddr	Slave Address	UINT	1 to 65535	0	Slave address.
Output	Done	FB execution done	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the function block execution is done.
	Busy	Function block executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE while the function block is executing.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
	ErrorID	Fault code	ETC_STD_ERROR	-	0	See 18.4.9.15 ETC_STD_ERROR: ETC Fault Codes.
	SlaveName	Slave name	WSTRING	Follows data type		Slave device name.
	SlaveAlias Addr	Slave alias address	UINT	Follows data type	0	Slave Alias Address, displayed only when the alias is used.
	SlaveState Act	Slave state	WORD	Follow data type	0	Slave state.
	SlaveError Code	Slave fault code	WORD	Follow data type	0	Slave fault code, corresponding to slave fault information.
	ErrorCode Level	Slave fault code level	BYTE	Follows data type	0	Slave fault code level.
	InvalidFra meCount	Invalid frame count	ARRAY [0..3] OF BYTE	Follows data type	0	Check error from CRC. There are CRC check errors in the network, such as non-Ethernet frames and other issues.
	ErrorForw ardCount	Forwarded frame count	ARRAY [0..3] OF BYTE	Follows data type	0	Data forwarding error: If the received data is wrong and ended with an extra error flag, it indicates the data has been processed by other stations.
	ErrorFram eCount	Error frame count	ARRAY [0..3] OF UINT	Follows data type	0	Slave error frame count.
	LostLinkC ount	Lost link count	ARRAY [0..3] OF BYTE	Follows data type	0	Lost slave link count.
	LinkUp	Port connection status	ARRAY [0..3] OF BOOL	Follows data type	FALSE	Slave port connection status.
	PortLoop Open	Port open status	ARRAY [0..3] OF BOOL	Follows data type	FALSE	Slave port open status.

- **Function**

  This instruction gets EtherCAT slave information.

- **Sequence diagram**



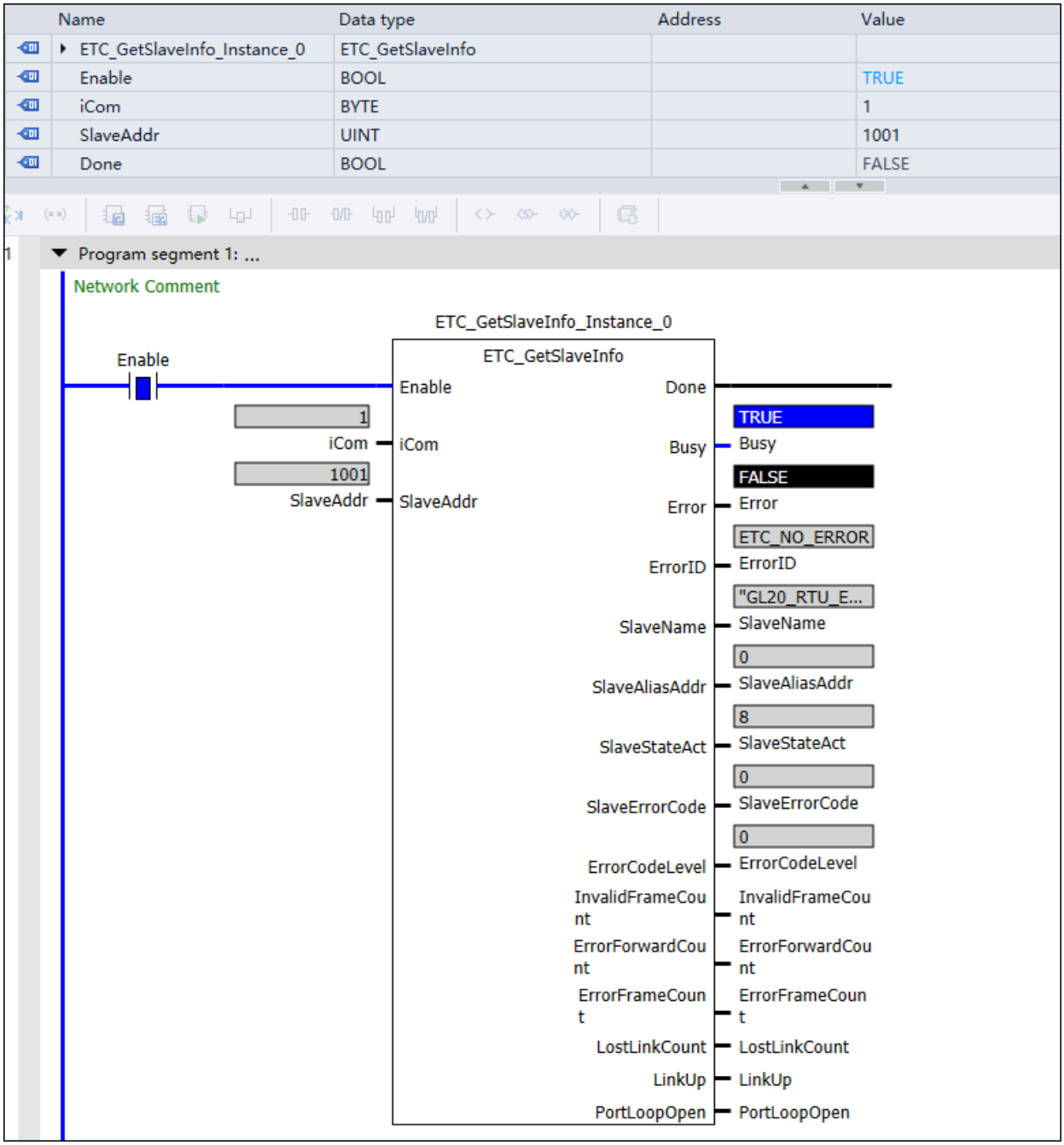
- **Program example**

  ST:

Name	Data type	Address	Value
ETC_GetSlaveInfo_Instance_0	ETC_GetSlaveInfo		
Enable	BOOL		TRUE
iCom	BYTE		1
SlaveAddr	UINT		1001
Done	BOOL		FALSE
Busy	BOOL		TRUE
Error	BOOL		FALSE
ErrorID	ETC_STD_ERROR		ETC_NO_ERROR
SlaveName	WSTRING(128)		"GL20_RTU_ECT32"
SlaveAliasAddr	UINT		0
SlaveStateAct	WORD		8
SlaveErrorCode	WORD		0
ErrorCodeLevel	BYTE		0
InvalidFrameCount	ARRAY [0..3] OF BYTE		
ErrorForwardCount	ARRAY [0..3] OF BYTE		
ErrorFrameCount	ARRAY [0..3] OF UINT		
LostLinkCount	ARRAY [0..3] OF BYTE		
LinkUp	ARRAY [0..3] OF BOOL		
PortLoopOpen	ARRAY [0..3] OF BOOL		

```
ETC_GetSlaveInfo_Instance_0(
Enable TRUE := Enable TRUE,
iCom 1 := iCom 1,
SlaveAddr 1001 := SlaveAddr 1001,
Done FALSE => Done FALSE,
Busy TRUE => Busy TRUE,
Error FALSE => Error FALSE,
ErrorIDETC_NO_ER... => ErrorIDETC_NO_ER...,
SlaveName"GL20_RTU_ECT... => SlaveName"GL20_RTU_ECT...,
SlaveAliasAddr 0 => SlaveAliasAddr 0,
SlaveStateAct 8 => SlaveStateAct 8,
SlaveErrorCode 0 => SlaveErrorCode 0,
ErrorCodeLevel 0 => ErrorCodeLevel 0,
InvalidFrameCount => InvalidFrameCount,
ErrorForwardCount => ErrorForwardCount,
ErrorFrameCount => ErrorFrameCount,
LostLinkCount => LostLinkCount,
LinkUp => LinkUp,
PortLoopOpen => PortLoopOpen);RETURN
```

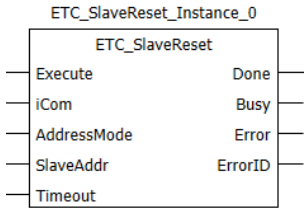
LD:



18.4.9.11 ETC_SlaveReset

This instruction resets slave communication faults and axis faults.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ETC_SlaveReset	Reset EtherCAT slave (resettable axis communication)	FB		ETC_SlaveReset_Instance_0(Execute :=, iCom := , AddressMode := , SlaveAddr := , Timeout := , Done =>, Busy =>, Error =>, ErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Execute	BOOL	TRUE, FALSE	FALSE	Rising edge triggered.
	iCom	Master identifier	BYTE	Follows data type	1	Number of EtherCAT masters: If only one EtherCAT master is used, iCom is '1'. When multiple masters are used, the value is 1 for the first master, 2 for the second master, and so on.
	AddressMode	Addressing mode	BYTE	Follows data type	0	Slave addressing mode 0: Sequential mode 1: Alias mode
	SlaveAddr	Slave Address	UINT	Follows data type	0	Physical address of the slave. If slave addressing mode is set to 0, physical address is used; if set to 1, alias address is used.
	Timeout	Timeout period	UINT	Follow data type	10000	Timeout interval
Output	Done	Done	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is done.
	Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is executing.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
	ErrorID	Fault code	ETC_CO_ERROR	-	ETC_CO_NO_ERROR	For details, see 18.4.9.16 ETC_CO_ERROR: ETC Fault Codes.

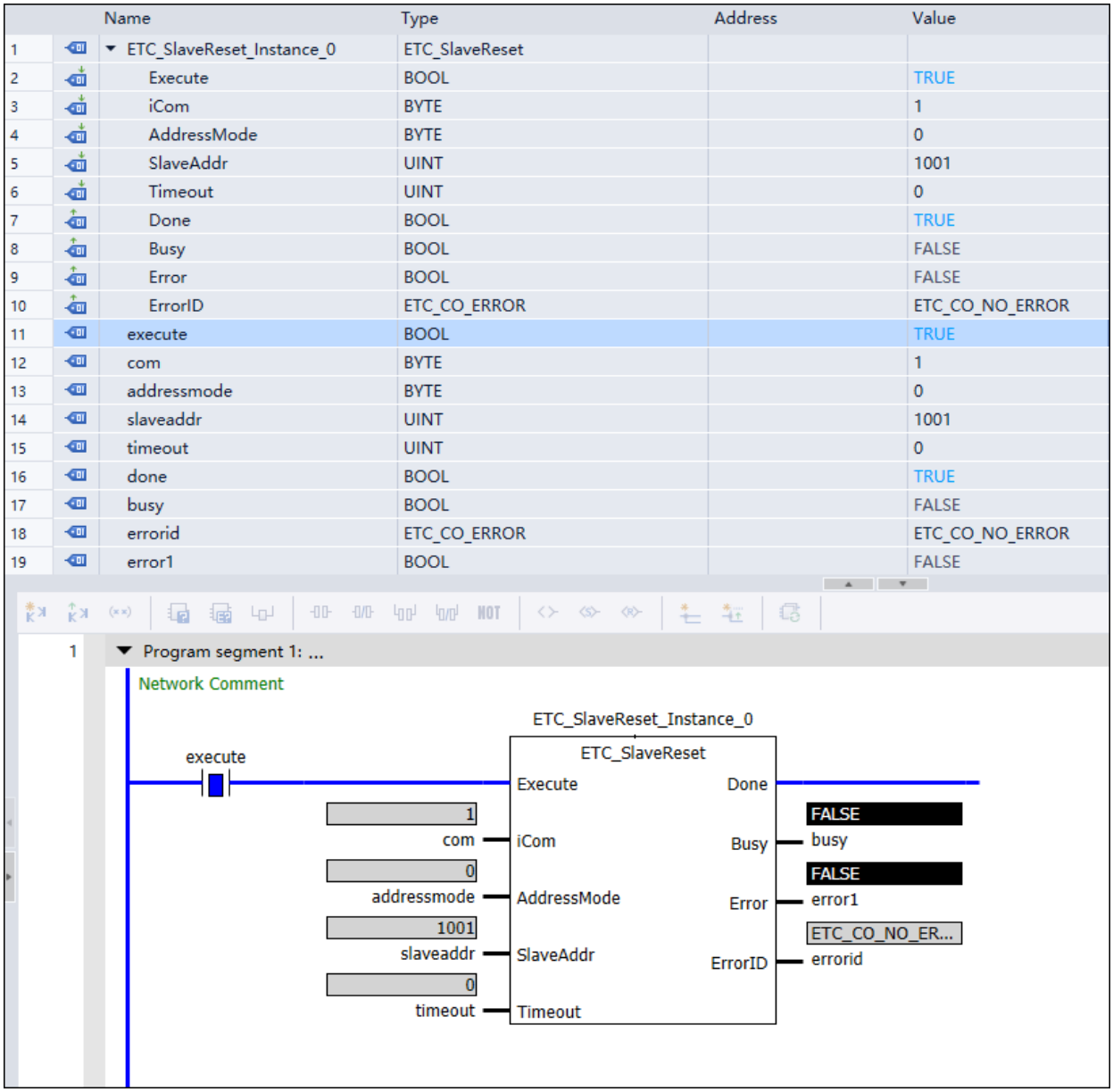
- **Function**
  - This instruction resets EtherCAT slaves and CIA402 axes, such as servo drive, AC drive, and EtherCAT remote I/O modules.
  - It automatically detects the number and status of CIA402 axes on EtherCAT slave devices, and resets the axis in Errorstop state for the state machine to facilitate axis reset for multidrive devices.
- **Program example**

  ST:

	Name	Type	Address	Value
1	ETC_SlaveReset_Instance_0	ETC_SlaveReset		
2	Execute	BOOL		TRUE
3	iCom	BYTE		1
4	AddressMode	BYTE		0
5	SlaveAddr	UINT		1001
6	Timeout	UINT		0
7	Done	BOOL		TRUE
8	Busy	BOOL		FALSE
9	Error	BOOL		FALSE
10	ErrorID	ETC_CO_ERROR		ETC_CO_NO_ERROR
11	execute	BOOL		TRUE
12	com	BYTE		1
13	addressmode	BYTE		0
14	slaveaddr	UINT		1001
15	timeout	UINT		0
16	done	BOOL		TRUE
17	busy	BOOL		FALSE
18	errorid	ETC_CO_ERROR		ETC_CO_NO_ERROR
19	error1	BOOL		FALSE

```
ETC_SlaveReset_Instance_0(Execute TRUE := execute TRUE,
                          iCom 1 := com 1,
                          AddressMode 0 := addressmode 0,
                          SlaveAddr 1001 := slaveaddr 1001,
                          Timeout 0 := timeout 0,
                          Done TRUE => done TRUE,
                          Busy FALSE => busy FALSE,
                          Error FALSE => error1 FALSE,
                          ErrorIDETC_CO_NO_ER... => erroridETC_CO_NO_ER...);RETURN
```

  LD:



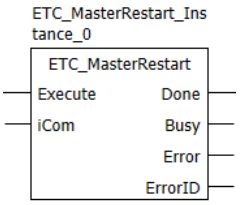
	Name	Type	Address	Value
1	ETC_SlaveReset_Instance_0	ETC_SlaveReset		
2	Execute	BOOL		TRUE
3	iCom	BYTE		1
4	AddressMode	BYTE		0
5	SlaveAddr	UINT		1001
6	Timeout	UINT		0
7	Done	BOOL		TRUE
8	Busy	BOOL		FALSE
9	Error	BOOL		FALSE
10	ErrorID	ETC_CO_ERROR		ETC_CO_NO_ERROR
11	execute	BOOL		TRUE
12	com	BYTE		1
13	addressmode	BYTE		0
14	slaveaddr	UINT		1001
15	timeout	UINT		0
16	done	BOOL		TRUE
17	busy	BOOL		FALSE
18	errorid	ETC_CO_ERROR		ETC_CO_NO_ERROR
19	error1	BOOL		FALSE

18.4.9.12 ETC_MasterRestart

This instruction restarts EtherCAT master.

> **NOTE**
>
> When the EtherCAT library version is 1.4.0.0 or later, ETCMaster_Restart is renamed to ETC_MasterRestart with unchanged functionality. It is recommended to use of the latest version.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ETC_MasterRestart	Restart EtherCAT master	FB		ETC_MasterRestart_Instance_0(Execute := , iCom := , Done => , Busy => , Error => , ErrorID =>);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Execute	BOOL	TRUE, FALSE	FALSE	Rising edge triggered.
	iCom	Master identifier	BYTE	Follows data type	1	Number of EtherCAT masters: If only one EtherCAT master is used, iCom is '1'. When multiple masters are used, the value is 1 for the first master, 2 for the second master, and so on.
Output	Done	Done	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is done.
	Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is executing.
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	TRUE when a fault occurs.
	ErrorID	Fault code	ETC_RESTART_ERROR	-	ETC_NO_ERROR	0: ETC_NO_ERROR: No error. 1: ETC_OTHER_ERROR: Master not found.

- **Function**

  This instruction restarts EtherCAT master. The ECT protocol stack restarts when the instruction is enabled.

- **Program example**

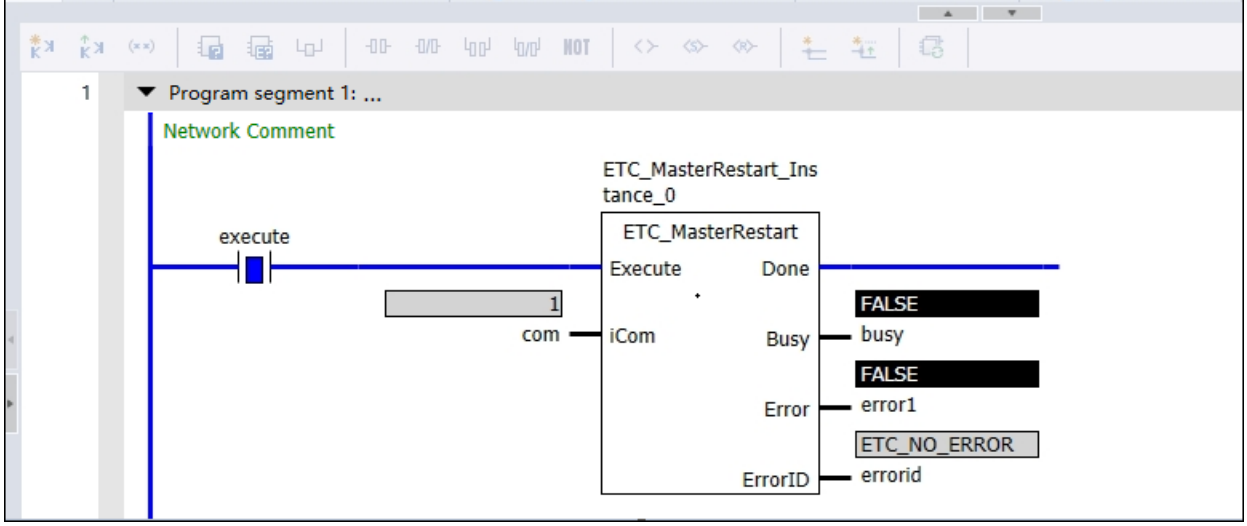
  ST:

	Name	Type	Address	Value
1	ETC_MasterRestart_Instance_0	ETC_MasterRestart		
2	Execute	BOOL		TRUE
3	iCom	BYTE		1
4	Done	BOOL		TRUE
5	Busy	BOOL		FALSE
6	Error	BOOL		FALSE
7	ErrorID	ETC_RESTART_ERROR		ETC_NO_ERROR
8	execute	BOOL		TRUE
9	com	BYTE		1
10	done	BOOL		TRUE
11	busy	BOOL		FALSE
12	error1	BOOL		FALSE
13	errorid	ETC_RESTART_ERROR		ETC_NO_ERROR

```
1  ETC_MasterRestart_Instance_0(Execute TRUE := execute TRUE ,
2                               iCom 1 := com 1 ,
3                               Done TRUE => done TRUE ,
4                               Busy FALSE => busy FALSE ,
5                               Error FALSE => error1 FALSE ,
6                               ErrorIDETC_NO_ER... => erroridETC_NO_ER...);
7                               RETURN
```

LD:

	Name	Type	Address	Value
1	ETC_MasterRestart_Instance_0	ETC_MasterRestart		
2	Execute	BOOL		TRUE
3	iCom	BYTE		1
4	Done	BOOL		TRUE
5	Busy	BOOL		FALSE
6	Error	BOOL		FALSE
7	ErrorID	ETC_RESTART_ERROR		ETC_NO_ERROR
8	execute	BOOL		TRUE
9	com	BYTE		1
10	done	BOOL		TRUE
11	busy	BOOL		FALSE
12	error1	BOOL		FALSE
13	errorid	ETC_RESTART_ERROR		ETC_NO_ERROR

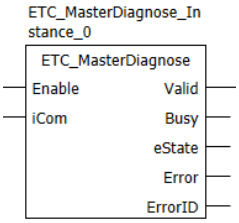


18.4.9.13 ETC_MasterDiagnose

This instruction gets the EtherCAT master diagnostic information.

> **NOTE**
>
> When the EtherCAT library version is 1.4.0.0 or later, SysHC_ETCMasterDiagnose is renamed to ETC_MasterDiagnose with unchanged functionality. It is recommended to use of the latest version.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ETC_MasterDiagnose	Get EtherCAT master diagnosis information	FB		ETC_MasterDiagnose_Instance_0(Enable := , iCom := , Valid => , Busy => , eState => , Error => , ErrorID =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Enable	Execute	BOOL	TRUE, FALSE	FALSE	Active High.
	iCom	Master identifier	BYTE	Follows data type	1	Number of EtherCAT masters: If only one EtherCAT master is used, iCom is '1'. When multiple masters are used, the value is 1 for the first master, 2 for the second master, and so on.
Output	Valid	Completed	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is done.
	Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is executing.
	eState	Diagnostic status bit	BOOL	TRUE, FALSE	FALSE	This is diagnostic status bit. eState is set when the EtherCAT master has a communication error, synchronization deviation exceeds the set threshold, or a slave is in a non-OP state.
	Error	Faulty status	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction execution failed.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	ErrorID	Fault code	SYS_HC_ETCERROR	-	HC_NO_ETCERROR	0: HC_NO_ETCERROR: No error. 1: HC_ETCMASTER_ERROR: Master not found.

- **Function**

  This instruction gets the EtherCAT master diagnostic information.

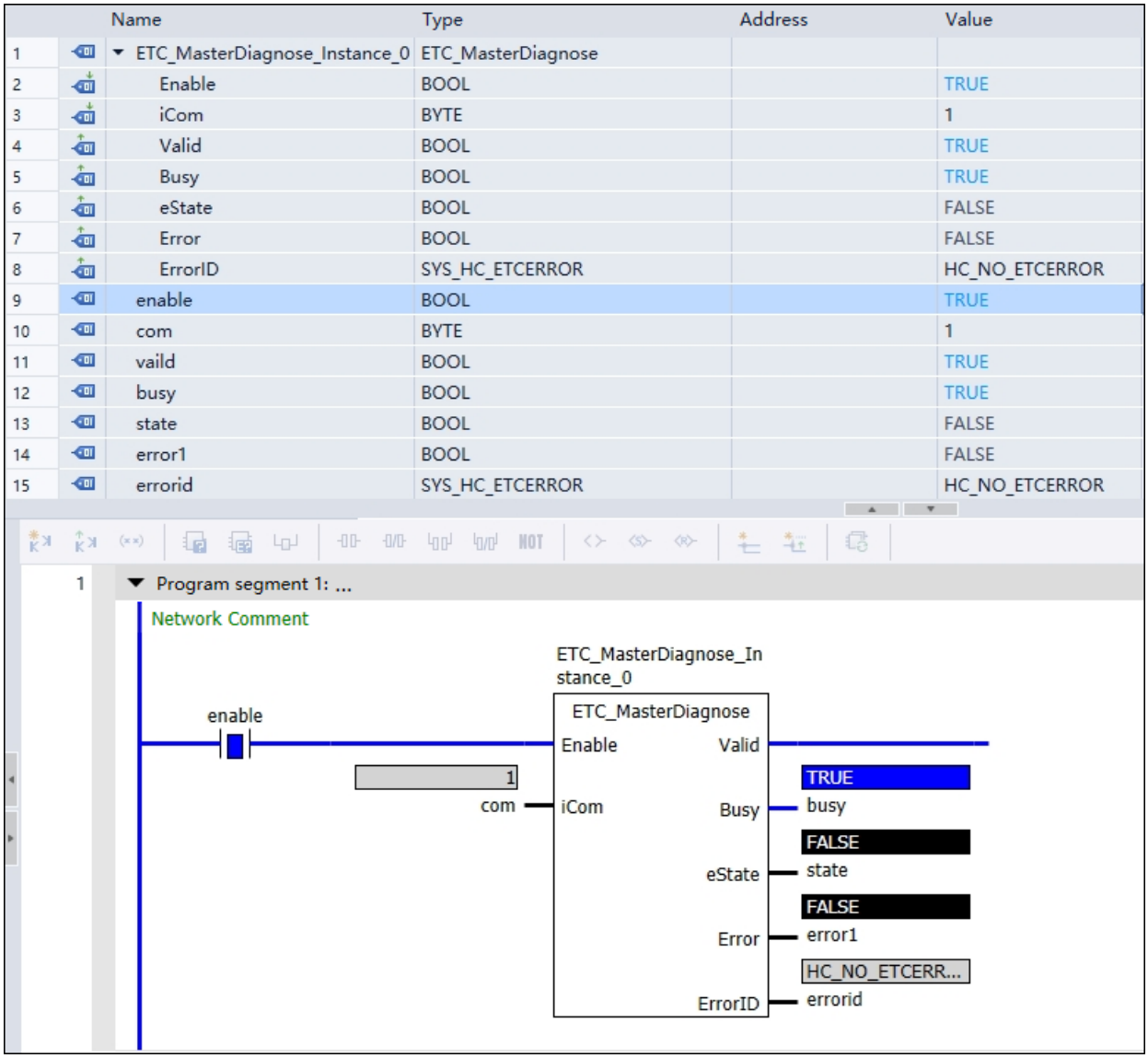
- **Program example**

  ST:

	Name	Type	Address	Value
1	ETC_MasterDiagnose_Instance_0	ETC_MasterDiagnose		
2	Enable	BOOL		TRUE
3	iCom	BYTE		1
4	Valid	BOOL		TRUE
5	Busy	BOOL		TRUE
6	eState	BOOL		FALSE
7	Error	BOOL		FALSE
8	ErrorID	SYS_HC_ETCERROR		HC_NO_ETCERROR
9	enable	BOOL		TRUE
10	com	BYTE		1
11	vaild	BOOL		TRUE
12	busy	BOOL		TRUE
13	state	BOOL		FALSE
14	error1	BOOL		FALSE
15	errorid	SYS_HC_ETCERROR		HC_NO_ETCERROR

```
1  ETC_MasterDiagnose_Instance_0(Enable TRUE := enable TRUE,
2                                iCom 1 := com 1,
3                                Valid TRUE => vaild TRUE,
4                                Busy TRUE => busy TRUE,
5                                eState FALSE => state FALSE,
6                                Error FALSE => error1 FALSE,
7                                ErrorIDHC_NO_ETCER... => erroridHC_NO_ETCER...);
8                               RETURN
```

  LD:



	Name	Type	Address	Value
1	ETC_MasterDiagnose_Instance_0	ETC_MasterDiagnose		
2	Enable	BOOL		TRUE
3	iCom	BYTE		1
4	Valid	BOOL		TRUE
5	Busy	BOOL		TRUE
6	eState	BOOL		FALSE
7	Error	BOOL		FALSE
8	ErrorID	SYS_HC_ETCERROR		HC_NO_ETCERROR
9	enable	BOOL		TRUE
10	com	BYTE		1
11	vaild	BOOL		TRUE
12	busy	BOOL		TRUE
13	state	BOOL		FALSE
14	error1	BOOL		FALSE
15	errorid	SYS_HC_ETCERROR		HC_NO_ETCERROR

18.4.9.14 ETC_SDO_ERROR: ETC Fault Codes

See the table below for the fault code ID of instruction.

Fault code	Value	Description
ETC_NO_ERROR	0	No fault
ETC_FIRST_ERROR	1000	The cause of the error is stored in the output of SdoAbort.
ETC_OTHER_ERROR	1001	Check if the master and slave exist.
ETC_DATA_OVERFLOW	1002	Byte size > 4.
ETC_TIME_OUT	1003	Timeout.
ETC_FIRST_MF	2000	Reserved.
ETC_LAST_ERROR	2100	Reserved.

18.4.9.15 ETC_STD_ERROR: ETC Fault Codes

Fault code	Value	Description
ETC_NO_ERROR	0	No error.

Fault code	Value	Description
ETC_CFG_NOSLAVE_ERROR	1	Slave configuration not generated.
ETC_CFG_NOMASTER_ERROR	2	Master index does not exist.
ETC_CFG_LOOPNET_ERROR	3	Network topology is a loop.
ETC_CO_GETSLAVE_ERROR	4	Current slave not found.
ETC_OTHER_ERROR	5	Other errors.
ETC_MD_PDO_NOT_MAPPED_ERROR	6	Diagnostic PDO not checked.
ETC_MD_UNSUPPORTED_DEVICE_ERROR	7	This slave device is not supported.
ETC_MD_SDO_READ_ERROR	8	Failed to read slave diagnostic information, requiring function block reset.
ETC_MD_ISNOT_OPERATIAL ERROR	9	Slave status has not reached operational status.
ETC_DD_DISABLED_ERROR	10	Module has been disabled via the backend, requiring backend disable to be canceled.
ETC_DD_WRONG_CALL_TIME_ERROR	11	The function block is not placed in the first line of the EtherCAT task, requiring the protocol stack to be restarted.
ETC_OLD_DATED_VERSION_ERROR	12	The EtherCAT version is too low to support this function block.
ETC NOT_IN_ETC_TASK_ERROR	13	This function block call is not under the EtherCAT task.
ETC UNCHECKED AUTOCONFIG_ERROR	14	The automatic master/slave configuration is not checked, requiring this optionto be selected.
ETC_DD_OBJECT_ERROR	15	Multiple instance calls of the function block are disabled, but the disablement is invalid.

18.4.9.16 ETC_CO_ERROR: ETC Fault Codes

Fault code	Value	Description
ETC_CO_NO_ERROR	0	No fault
ETC_CO_FIRST_ERROR	1	The cause of the error is stored in SdoAbort. For details, see 18.4.9.17 SDO Fault Codes.
ETC_CO_OTHER_ERROR	2	Check if the master and slaves exist.
ETC_CO_DATA_OVERFLOW	3	szSize exceeds the available size of ReadBuffer.
ETC_CO_TIME_OUT	4	Timeout.
ETC_CO_INPUT_INVALID	5	Invalid input parameters.
ETC_CO_FIRST_MF	32	Reserved.
ETC_CO_LAST_ERROR	256	Reserved.

18.4.9.17 SDO Fault Codes

SDO Access Fault Code	Description
0x00000002	Slave state inaccessible.
0xFFFFFFFF	Master or slave information not found.
0x05030000	Switch bit not changed.
0x05040000	SDO protocol timeout
0x05040001	Client/server command specifier not valid or unknown.
0x05040005	Insufficient memory.
0x06010000	Unsupported access to an object.
0x06010001	Attempt to read a write-only object.
0x06010002	Attempt to write to a read-only object.
0x06020000	Object does not exist in the object directory.
0x06040041	object cannot be mapped to PDO.
0x06040042	The number and length of the objects to be mapped would exceed PDO range.
0x06040043	General parameter incompatibility reason.
0x06040047	General incompatibility issue within the device.
0x06060000	Access failed due to a hardware error.
0x06070010	Data type and service parameter length mismatch.
0x06070012	Data type mismatch, length of service parameter too high.
0x06070013	Data type mismatch, length of service parameter too low.
0x06090011	Sub-index does not exist.
0x06090030	Parameter value out of range.
0x06090031	Written parameter value too high.
0x06090032	Written parameter value too low.
0x06090036	Maximum value is less than minimum value.
0x08000000	General error
0x08000020	Data cannot be transmitted/stored to application.
0x08000021	Data cannot be transmitted/stored to application due to local control.
0x08000022	Data cannot be transmitted or stored to the application due to the current device status.
0x08000023	Object dictionary dynamic generation fails or no object dictionary is present.

18.4.9.18 IO Fault Codes

Fault Code	Description
Communication Protocol Conversion Module Fault	

Fault Code	Description
0x1000	The number of scanned expansion modules is 0.
0x1001	The number of I/O modules configured is more than the number of I/O modules actually scanned.
0x1002	The number of I/O modules configured is less than the number of I/O modules actually scanned.
0x1003	The type of I/O modules configured is inconsistent with that of the I/O modules actually scanned.
0x1004	IO module configuration failed.
0x1005	I/O module status switch failed.
0x1006	IO module is offline.
Module configuration data fault.	
0x2004	The number of sub-modules on serial port 0/1 of the 2SCOM-MDB module exceeds 15.
0x2005	The process data of the 2SCOM-MDB module exceeds 480 bytes.
0x2010	The baud rate of the RS485 module channel is configured incorrectly.
0x2011	The communication mode of the RS485 module channel is configured incorrectly.
0x2012	The data bit-related parameters of the RS485 module channel is configured incorrectly.
0x2013	The data packet size of the RS485 module channel is configured incorrectly.
0x2014	The message interval of the RS485 module channel is configured incorrectly.
0x2015	The sending buffer of the RS485 module channel is configured incorrectly.
0x2016	The receiving buffer of the RS485 module channel is configured incorrectly.
0x2020	The baud rate of the CAN module channel is configured incorrectly.
0x2021	The communication mode of the CAN module channel is configured incorrectly.
0x2022	The data bit-related parameters of the CAN module channel is configured incorrectly.
0x2023	The buffer size of the CAN module channel is configured incorrectly.
0x2024	The heartbeat interval of the CAN module channel is configured incorrectly.
0x2025	The standard frame upper limit of the CAN module channel is configured incorrectly.
0x2026	The standard frame lower limit of the CAN module channel is configured incorrectly.
0x2030	The serial port parameters of the 2SCOM/2SCOM-MDB module channel are configured incorrectly.
0x2031	The data packet size is incorrectly configured for the channels of the 2SCOM module.
0x2032	The timeout parameter is incorrectly configured for the channels of the 2SCOM module.

Fault Code	Description
0x2033	The port protocol of the 2SCOM/2SCOM-MDB module channel is configured incorrectly.
0x2040	Incorrect Modbus slave address
0x2100	The CH0 preset value is out of range.
0x2101	The CH0 counter is configured incorrectly.
0x2102	The CH0 signal type is not supported.
0x2103	The CH0 counter limit is configured incorrectly.
0x2104	The CH0 unit pulse increment is configured incorrectly.
0x2105	The CH0 time base of the speed measurement is configured incorrectly.
0x2106	The CH0 input filter parameter is configured incorrectly.
0x2107	The CH0 frequency sampling time is configured incorrectly.
0x2108	The CH0 reset function is configured incorrectly.
0x2109	The CH0 preset function is configured incorrectly.
0x210A	The CH0 comparison 0 function is configured incorrectly.
0x210B	The CH0 comparison 1 function is configured incorrectly.
0x210C	The CH0 comparison 2 function is configured incorrectly.
0x210D	The CH0 comparison 3 function is configured incorrectly.
0x210E	The CH0 probe 0 function is configured incorrectly.
0x210F	The CH0 probe 1 function is configured incorrectly.
0x2110	The CH0 probe 2 function is configured incorrectly.
0x2111	The CH0 probe 3 function is configured incorrectly.
0x2112	The CH0 DO0 parameter is configured incorrectly.
0x2113	The CH0 DO1 parameter is configured incorrectly.
0x2114	The CH0 DO2 parameter is configured incorrectly.
0x2115	The CH0 DO3 parameter is configured incorrectly.
0x2116	The CH0 D10 parameter is configured incorrectly.
0x2117	The CH0 DI1 parameter is configured incorrectly.
0x2118	The CH0 DI2 parameter is configured incorrectly.
0x2119	The CH0 DI3 parameter error is configured incorrectly.
0x2200 to 0x2219	The CH1 is the same as CH0.
0x2300-0x2319	The CH2 is the same as CH0.
0x2400-0x2419	The CH3 is the same as CH0.
IO module fault	
0x5001	Power supply overvoltage.
0x5002	Power supply undervoltage.

Fault Code	Description
0x5003	External 24 V power failure.
0x5004	ADC device configuration parameter fault.
0x5011	Chip overheating.
0x5021	ADC device failure.
0x5022	ADC device failure.
0x5031	Reference channel failure.
IO module fault	
0x6001	Channel open-circuit.
0x6002	Channel short-circuit.
0x6003	Channel data exceeds upper limit.
0x6004	Channel data exceeds lower limit.
0x6005	Channel data overflow.
0x6006	Channel data underflow.
0x6007	Channel overload.
0x6008	Channel overcurrent.
0x6009	Channel configuration data fault.
0x6010	The sending buffer of the RS485 module channel is full.
0x6011	The receiving buffer of the RS485 module channel is full.
0x6012	The transmitting frame counter value of the RS485 module channel is faulty.
0x6013	The receiving frame segnum value of the RS485 module channel is faulty.
0x6014	Noise detected on the RS485 module channel.
0x6015	Parity bit fault detected on the RS485 module channel.
0x6020	The sending buffer of the CAN module channel is full.
0x6021	The receiving buffer of the CAN module channel is full.
0x6022	The receiving frame counter value of the CAN module channel is faulty.
0x6023	The bus load ratio of the CAN module exceeds 30%.
0x6024	The value of TEC of the CAN module exceeds 127.
0x6025	The value of REC of the CAN module exceeds 127.
0x6030	24 V output fault of SSI module channel.
0x6031	Communication fault of SSI module channel.
0x6032	Data error of SSI module channel.
0x6040	The sending buffer of the 2SCOM module channel is full.
0x6041	The receiving buffer of the 2SCOM module channel is full.
0x6042	The receiving frame counter value of the 2SCOM module channel is faulty.
0x6043	The receiving frame segnum value of the 2SCOM module channel is faulty.

Fault Code	Description
0x6044	Noise detected on the 2SCOM/2SCOM-MDB module channel.
0x6045	Parity bit fault detected on the 2SCOM/2SCOM-MDB module channel.
0x6046	Submodules of the 2SCOM-MDB module channel are configured incorrectly.
0x6050	Channel overcurrent.
0x6051	Switch OFF open-circuit.
0x6052	Switch ON open-circuit.
0x6053	Short-circuit to VDD.
0x6054	Overload.
0x6060	M: Modbus response parameter mismatch.
0x6061	M: Modbus slave ID mismatch.
0x6062	M: Modbus response address mismatch.
0x6063	M: Modbus response data length mismatch.
0x6064	M: Modbus communication timeout.
0x6065	M: Modbus CRC check error.
0x6066	Illegal function code.
0x6067	Illegal data address.
0x6068	Illegal data value.
0x6069	Slave device fault.
0x6800	CH0 phase A open-circuit.
0x6801	CH0 phase B open-circuit.
0x6802	CH0 frequency out of limit.
0x6803	CH0 count value overflow.
0x6804	CH0 count value underflow.
0x6805	Number/stepsize limit of CH0 comparison 0 function exceeded
0x6806	Number/stepsize limit of CH0 comparison 1 function exceeded
0x6807	Number/stepsize limit of CH0 comparison 2 function exceeded
0x6808	Number/stepsize limit of CH0 comparison 3 function exceeded
0x6809	Hold time/pulse number of CH0 comparison 0 output function is configured incorrectly.
0x680A	Hold time/pulse number of CH0 comparison 1 output function is configured incorrectly.
0x680B	Hold time/pulse number of CH0 comparison 2 output function is configured incorrectly.
0x680C	Hold time/pulse number of CH0 comparison 3 output function is configured incorrectly.

Fault Code	Description
0x680D	Start/stop position and stepsize direction of CH0 comparison 0 output function is configured incorrectly.
0x680E	Start/stop position and stepsize direction of CH0 comparison 1 output function is configured incorrectly.
0x680F	Start/stop position and stepsize direction of CH0 comparison 2 output function is configured incorrectly.
0x6810	Start/stop position and stepsize direction of CH0 comparison 3 output function is configured incorrectly.
0x6811	CH0 comparison 0 limit exceeded
0x6812	CH0 comparison 1 limit exceeded
0x6813	CH0 comparison 2 limit exceeded
0x6814	CH0 comparison 3 limit exceeded
0x6815	CH0 comparison 0 type fault
0x6816	CH0 comparison 1 type fault
0x6817	CH0 comparison 2 type fault
0x6818	CH0 comparison 3 type fault
0x6819	Control mode fault of CH0 comparison 0 output function
0x681A	Control mode fault of CH0 comparison 1 output function
0x681B	Control mode fault of CH0 comparison 2 output function
0x681C	Control mode fault of CH0 comparison 3 output function
0x681D	CH0 DO0 short-circuit fault.
0x681E	CH0 DO1 short-circuit fault.
0x681F	CH0 DO2 short-circuit fault.
0x6820	CH0 DO3 short-circuit fault.
0x6900-0x6920	The CH1 is the same as CH0.
0x6A00-0x6A20	The CH2 is the same as CH0.
0x6B00-0x6B20	The CH3 is the same as CH0.

18.4.10 CANBus

18.4.10.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.4.10.12 CANBus_CtrlDevState	FB	Switch control device status	CANBus	Library version 1.0.0.0 and later
18.4.10.13 CANBus_DevRxMsg	FB	Device receives messages		Library version 1.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.4.10.14 CANBus_DevRxMultiMsg	FB	Device receives multiple frame messages		Library version 1.0.0.0 and later
18.4.10.15 CANBus_DevTxMsg	FB	Device transmitting messages		Library version 1.0.0.0 and later
18.4.10.16 CANBus_GetDevDiagInfo	FB	Obtain device diagnostic information		Library version 1.0.0.0 and later
18.4.10.17 CANBus_GetIoDrvVer	FB	Get the version information of the CmpDrvCANBus component		Library version 1.0.0.0 and later
18.4.10.18 CANBus_SetDevBaud	FB	Set baud rate of device		Library version 1.0.0.0 and later

18.4.10.2 CANBus_CmpDrvVer_t: Version of the Drive Component Used by CANBus

Variable	Data Type	Description
dwCmpDrvID	DWORD	Component ID.
dwCmpDrvVer	DWORD	Component version number.
dwCmpDrvDate	DWORD	Component creation date

18.4.10.3 CANBus_DiagInfo_t: CANBus Diagnostic Information Structure

Variable	Data Type	Description
xEnableFlag	BOOL	Enable flag. FALSE: Disable. TRUE: Enable.
xErrorFlag	BOOL	Error flag. FALSE: Normal. TRUE: Error.
uiNetId	UINT	Network ID of the CANBus device.
eStateType	18.4.10.11 CANBus_StateType_Enum: Enumeration of CANBus State	Operating state of the CANBus device.
eErrorType	18.4.10.8 CANBus_ErrorType_Enum: Enumeration of CANBus Error	Error type of the CANBus device.
uiBaudrate	UINT	Bus baud rate of CAN in kbps.
uiBusload	UINT	Bus load rate of CAN in ‰.
uiTxErrCnt	UINT	Transmit error counter TEC.

Variable	Data Type	Description
uiRxErrCnt	UINT	Receive error counter REC.
udiTxMsgNumInQueue	UDINT	Number of frames to be transmitted in the queue.
udiTxMsgNumFinish	UDINT	Number of transmitted frames.
udiTxBitCntPerSecond	UDINT	Transmit bit counter per second.
udiRxMsgNuminQueue	UDINT	Number of receive frames in the queue.
udiRxMsgNumFinish	UDINT	Number of received frames.
udiRxBitCntPerSecond	UDINT	Receive bit counter per second.

18.4.10.4 CANBus_FrameMsg_t: CANBus Frame Message Structure

Variable	Data Type	Description
xExtFlag	BOOL	Extended frame flag. FALSE: Standard frame. TRUE: Extended frame.
xRtrFlag	BOOL	Remote frame flag. FALSE: Data frame. TRUE: Remote frame.
dwCanId	DWORD	CANID. Standard frame: Lower 11 bits valid; Extended frame: All 29 bits valid.
usiDataLen	USINT	Data Length: 0 to 8 for data frames, and 0 for remote frames.
abyDataStr	ARRAY[0..7] OF BYTE	Data Area: The maximum number of bytes in the CAN frame data area is 8.

18.4.10.5 CANBus_RxFilter_t: CANBus Receive Frame Filter Structure

Variable	Data Type	Description
xExtFlag	BOOL	Extended frame flag. FALSE: Standard frame. TRUE: Extended frame.
xRtrFlag	BOOL	Remote frame flag. FALSE: Data frame. TRUE: Remote frame.

Variable	Data Type	Description
dwCanId	DWORD	CAN identifier. Standard frame: Lower 11 bits are valid (Bit 10~0); Extended frame: Lower 29 bits are valid (Bit 28~0).
xFilterFlag	BOOL	Enable flag for filter mechanism. FALSE: Disable filter mechanism and receive frames by single ID. TRUE: Enable filter mechanism.
xFilterExt	BOOL	Extended frame filtering. FALSE: Disable xExtFlag filter. TRUE: Enable xExtFlag filter.
xFilterRtr	BOOL	Remote frame filtering. FALSE: Disable xRtrFlag filter. TRUE: Enable xRtrFlag filter.
dwFilterCanId	DWORD	CANID bitwise filtering. bitn=FALSE: Disable filtering on dwCanId.bit n. bitn=TRUE: Enable filtering on dwCanId.bit n.

18.4.10.6 CANBus_BaudrateType_Enum: CANBus Baud Rate Enumeration

Variable	Value	Description
CANBUS_BAUDRATE_10KBPS	10	Baud rate is 10 kbps.
CANBUS_BAUDRATE_20KBPS	20	Baud rate is 20 kbps.
CANBUS_BAUDRATE_50KBPS	50	Baud rate is 50 kbps.
CANBUS_BAUDRATE_100KBPS	100	Baud rate is 100 kbps.
CANBUS_BAUDRATE_125KBPS	125	Baud rate is 125 kbps.
CANBUS_BAUDRATE_250KBPS	250	Baud rate is 250 kbps.
CANBUS_BAUDRATE_500KBPS	500	Baud rate is 500 kbps.
CANBUS_BAUDRATE_800KBPS	800	Baud rate is 800 kbps.
CANBUS_BAUDRATE_1000KBPS	1000	Baud rate is 1000 kbps.

18.4.10.7 CANBus_CtrlCmdType_Enum: CANBus Control Command Enumeration

Variable	Value	Description
CANBUS_CTRLCMD_INVALID	0	Invalid command.
CANBUS_CTRLCMD_RESTART	1	Restart CANBus.
CANBUS_CTRLCMD_START	2	Start CANBus.
CANBUS_CTRLCMD_STOP	3	Stop CANBus.
CANBUS_CTRLCMD_CLEAR	4	Clear CANBus parameters.

18.4.10.8 CANBus_ErrorType_Enum: CANBus Error Enumeration

Variable	Value	Description
CANBUS_ERR_OK_NOERR	0	Normal or no error.
CANBUS_ERR_CFG_ERR	1	Configuration error.
CANBUS_ERR_DRV_ILLEGAL_ERR	2	Driver class illegal error.
CANBUS_ERR_DRV_NO_ACK_ERR	3	Driver class no response error.
CANBUS_ERR_DRV_BUSOFF_ERR	4	Driver class bus shutdown error.
CANBUS_ERR_BUS_LOAD_OV_ERR	5	High bus load error.
CANBUS_ERR_TX_BUSY_ERR	16	Transmission busy error.
CANBUS_ERR_TX_TIMEOUT_ERR	17	Transmisson timeout error.
CANBUS_ERR_TX_BUFF_FULL_ERR	18	Transmit buffer full error.
CANBUS_ERR_RX_BUFF_EMPTY_ER R	32	Reception buffer empty error.
CANBUS_ERR_RX_BUFF_FULL_ERR	33	Reception buffer full error.
CANBUS_ERR_OTHER_UNDEF_ERR	65535	Undefined errors.

18.4.10.9 CANBus_FBErrorType_Enum: CANBus Function Block Error Enumeration

Variable	Value	Description
CANBUS_FBERR_NO_ERROR	0	Function block operates normally or there is no error.
CANBUS_FBERR_EXE_FAILED	1	Function block execution failed.
CANBUS_FBERR_INVAILD_DEVICEID	2	Invalid device ID, which is not enabled.
CANBUS_FBERR_DEVICE_FAULT	3	There is a fault with the device.
CANBUS_FBERR_WRONG_PARAME TER	4	The input parameters of the function block are incorrect.
CANBUS_FBERR_BUSY_ERROR	5	Function block is busy.

Variable	Value	Description
CANBUS_FBERR_ABORT_ERROR	6	Function block abort error.
CANBUS_FBERR_TIME_OUT_ERROR	7	Function block timeout error.
CANBUS_FBERR_POINTER_NULL	8	Function block null pointer error.
CANBUS_FBERR_BUFF_FULL	9	Function block buffer full error.
CANBUS_FBERR_UNKNOWN_ERRO R	255	Function block unknown error.

18.4.10.10 CANBus_FBStateType_Enum: CANBus Function Block State Enumeration

Variable	Value	Description
CANBUS_FBSTATE_DORMANT	0	Function block is waiting for startup.
CANBUS_FBSTATE_EXECUTING	1	Function block operates periodically.
CANBUS_FBSTATE_ABORTING	2	Function block aborts operation.
CANBUS_FBSTATE_DONE	3	Function block has completed operation.
CANBUS_FBSTATE_ERROR	4	Function block has an operation error.
CANBUS_FBSTATE_RESETTING	5	Function block is resetting.

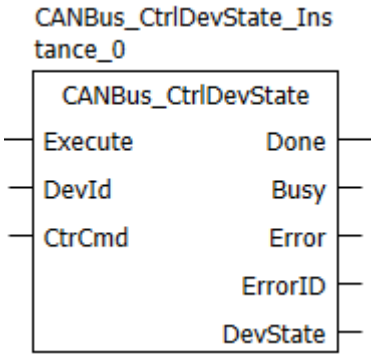
18.4.10.11 CANBus_StateType_Enum : CANBus State Enumeration

Variable	Value	Description
CANBUS_STATE_UNKNOWN	0	Unknown
CANBUS_STATE_INITIALIZE	1	Initializing
CANBUS_STATE_RUNNING	2	Operating with the position system restored.
CANBUS STATE STOPPING	3	Stopped
CANBUS STATE FAULTRUN	4	Fault running
CANBUS STATE_FAULTSTOP	5	Fault stopped

18.4.10.12 CANBus_CtrlDevState

This instruction switches control device status.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
CANBus_CtrlDevState	Switch control device status	FB		CANBus_CtrlDevState_Instance_0(Execute := , DevId := , CtrCmd := , Done => , Busy => , Error => , ErrorID => , DevState =>);

- **Related variables**

  Variable

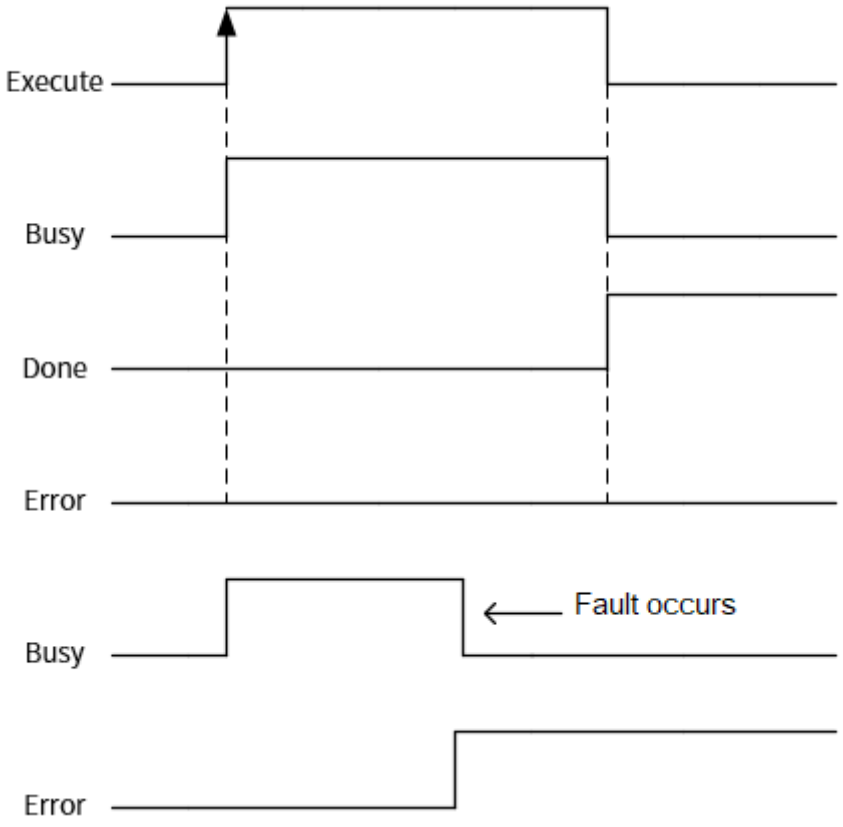
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Function block enable	BOOL	Follows data type	FALSE	Function block enable.
	DevId	Network ID of the CANBus device.	UINT	Follow data type	-	The device ID must correspond to the network ID of the CANBus device.
	CtrCmd	Instruction for control device status switching	18.4.10.7 CANBus_CtrlCmdType_Enum: Enumeration of CANBus Control Command	Follows data type	-	Instruction for control device status switching.
Output	Done	Instruction completion flag	BOOL	Follows data type	FALSE	Function block completion flag.
	Busy	Instruction execution flag	BOOL	Follows data type	FALSE	Function block execution flag.
	Error	Error flag	BOOL	Follows data type	FALSE	Function block error flag.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	ErrorID	Fault code	18.4.10.9 CANBus_FB ErrorType_Enum: Enumeration of CANBus Function Block Error	Follows data type	-	Fault code of function block. See Fault description.
	DevState	Device state	18.4.10.11 CANBus_StateType_Enum: Enumeration of CANBus State	Follows data type	-	Device status.

- **Function**

  This instruction controls the state of the device, which is determined by18.4.10.7 CANBus_CtrlCmdType_Enum: Enumeration of CANBus Control Command.

- **Sequence diagram**



- **Program example**

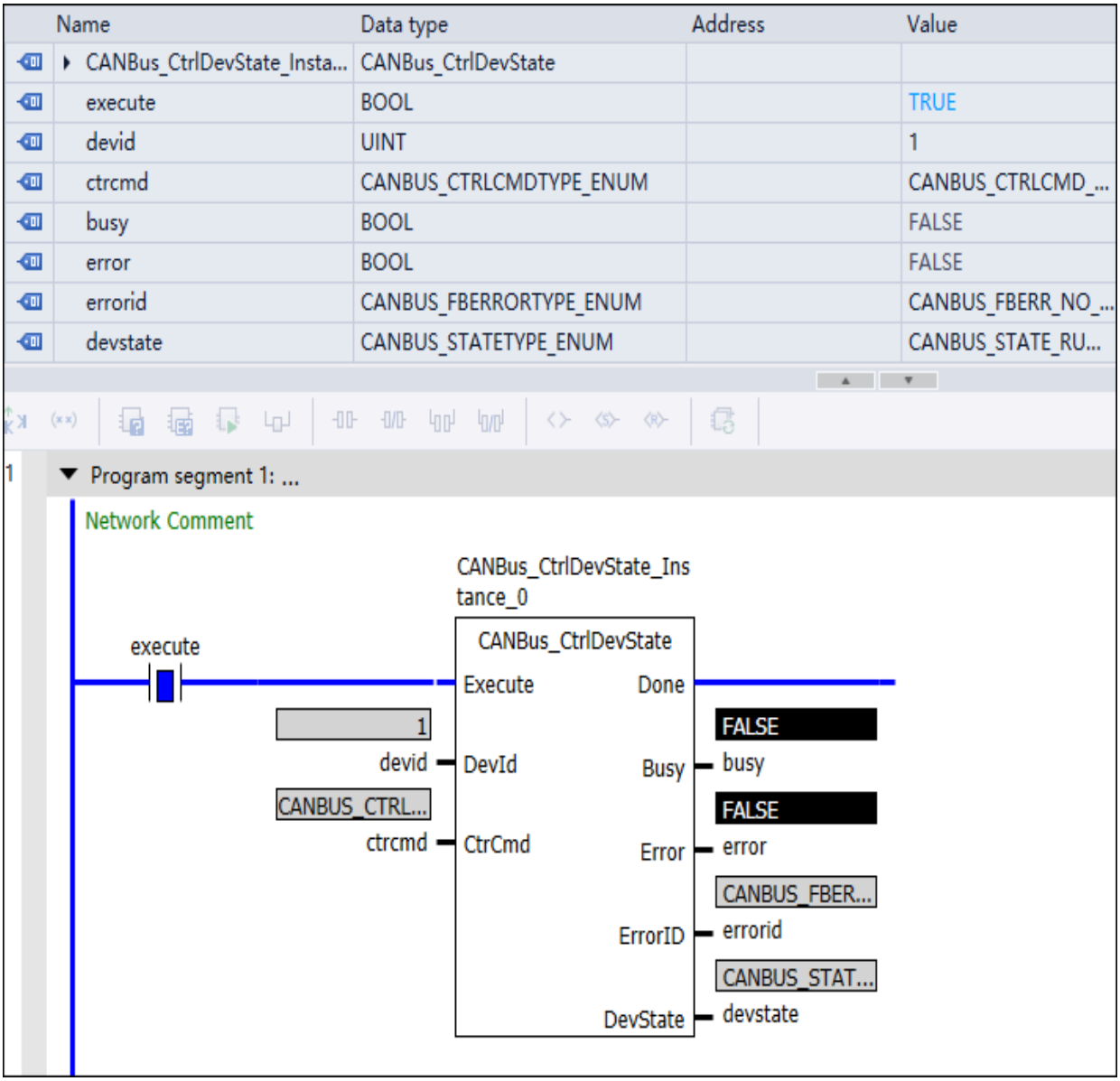
  ST:

Name	Data type	Address	Value
CANBus_CtrlDevState_Instan...	CANBus_CtrlDevState		
execute	BOOL		TRUE
devid	UINT		1
ctrcmd	CANBUS_CTRLCMDTYPE_ENUM		CANBUS_CTRLCMD_R...
done	BOOL		TRUE
busy	BOOL		FALSE
error	BOOL		FALSE
errorid	CANBUS_FBERRORTYPE_ENUM		CANBUS_FBERR_NO_E...
devstate	CANBUS_STATETYPE_ENUM		CANBUS_STATE_RUN...

CAS FOR WHI REP RET EX REG // /* */

```
CANBus_CtrlDevState_Instance_0(
Execute TRUE := execute TRUE,
DevId 1 := devid 1,
CtrCmdCANBUS_CTRLCMD_REST... := ctrcmdCANBUS_CTRLCMD_REST...,
Done TRUE => done TRUE,
Busy FALSE => busy FALSE,
Error FALSE => error FALSE,
ErrorIDCANBUS_FBERR_NO_ER... => erroridCANBUS_FBERR_NO_ER...,
DevStateCANBUS_STATE_RUNN... => devstateCANBUS_STATE_RUNN...);
```

LD:



- **Fault description**

  When an error occurs, find the corresponding error via ErrorID, and locate the root cause of the error.

Fault code	Value	Description	Troubleshooting Method
CANBUS_FBERR_NO_ERROR	0x0000	Function block operates normally or there is no error.	Function block executes normally.
CANBUS_FBERR_EXE_FAILED	0x0001	Function block execution failed.	-
CANBUS_FBERR_INVAILD_DEVICEID	0x0002	Invalid device ID, which is not enabled.	Invalid input DevID.
CANBUS_FBERR_DEVICE_FAULT	0x0003	There is a fault with the device.	The device is starting up.
CANBUS_FBERR_WRONG_PARAMETER	0x0004	The input parameters of the function block are incorrect.	Invalid input CtrlCmd.
CANBUS_FBERR_BUSY_ERROR	0x0005	Function block is busy.	-

Fault code	Value	Description	Troubleshooting Method
CANBUS_FBERR_ABORT_ERROR	0x0006	Function block abort error.	-
CANBUS_FBERR_TIME_OUT_ERROR	0x0007	Function block timeout error.	-
CANBUS_FBERR_POINTER_NULL	0x0008	Function block null pointer error.	-
CANBUS_FBERR_BUFF_FULL	0x0009	Function block buffer full error.	-
CANBUS_FBERR_UNKNOWN_ERROR	0x00FF	Function block unknown error.	-

- **Precautions**

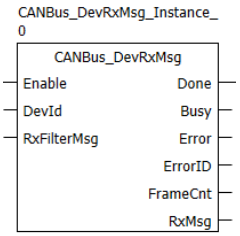
  If the function block instruction input is CANBUS_CTRLCMD_CLEAR (clear CANBus parameters), the following information will be cleared after the function block is triggered: (1) Current load rate (2) Internal receiving and sending buffers of the protocol stack (3) Receive function blocks.

  The data in items (1) and (2) will be automatically overwritten by the operation results, while item (3) requires the receive function blocks to be restarted manually.

18.4.10.13 CANBus_DevRxMsg

This instruction allows the device to receive messages.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
CANBus_DevRxMsg	Device receives messages	FB		CANBus_DevRxMsg_Instance_0(Enable := , DevId := , RxFilterMsg := , Done => , Busy => , Error => , ErrorID => , FrameCnt => , RxMsg =>);

- **Related variables**

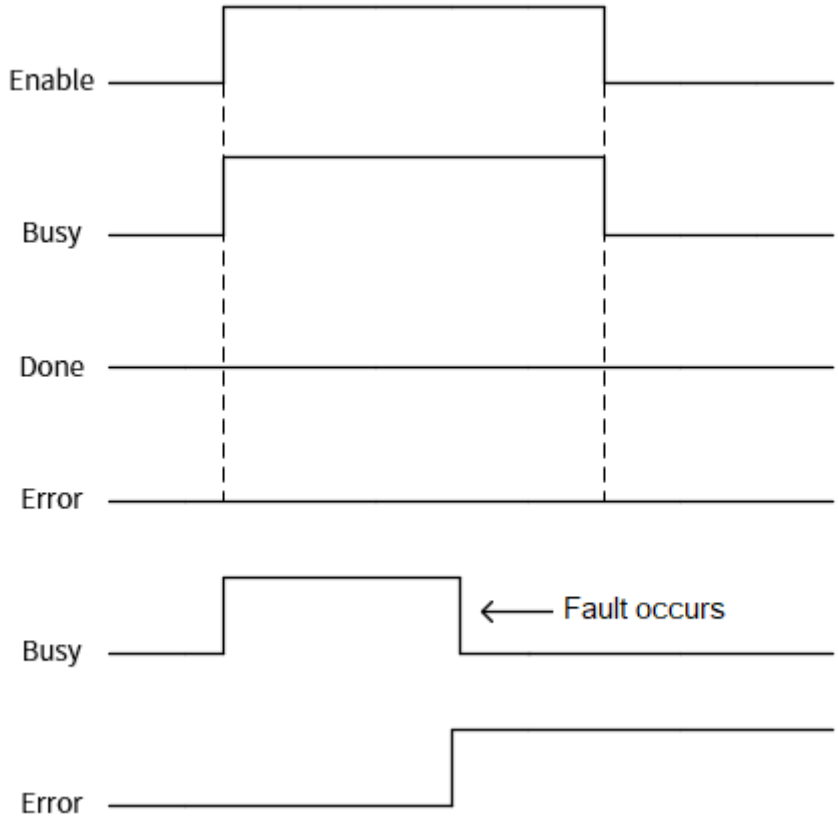
  Variable

Input / Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Enable	Function block enable flag	BOOL	Follows data type	FALSE	Function block enable flag TRUE: Active at high level FALSE: Stop/Reset at low level
	DevId	Network ID of the CANBus device.	UINT	Follow data type	-	The device ID must correspond to the network ID of the CANBus device.
	RxFilterMsg	Setting of receiving filtered messages	18.4.10.5 CANBus_RxFilter_t: Structure of CANBus Receive Frame Filter	Follows data type	-	Set reception parameters and filtering parameters.
Output	Done	Instruction completion flag	BOOL	Follows data type	FALSE	Function block completion flag.
	Busy	Instruction execution flag	BOOL	Follows data type	FALSE	Function block execution flag.
	Error	Error flag	BOOL	Follows data type	FALSE	Function block error flag.
	ErrorID	Fault code	18.4.10.9 CANBus_FBErrorType_Enum: Enumeration of CANBus Function Block Error	Follows data type	-	Fault code of function block. See Fault description.
	FrameCnt	Frame counter	UDINT	Follows data type	-	Frame counter that indicates the number of received qualified frames.
	RxMsg	Received messages	18.4.10.4 CANBus_FrameMsg_t: Structure of the CANBus Frame Message	Follows data type	-	Stores the received qualified CAN frame messages. (Only the latest received CAN frame is buffered)

- **Function**

  When the function block is enabled, it receives CAN data from the network number corresponding to DevId. The filtering is determined by RxFilterMsg. Received qualified CAN frames are counted in FrameCnt, and the messages of the latest qualified CAN frame is stored in RxMsg.

- **Sequence diagram**



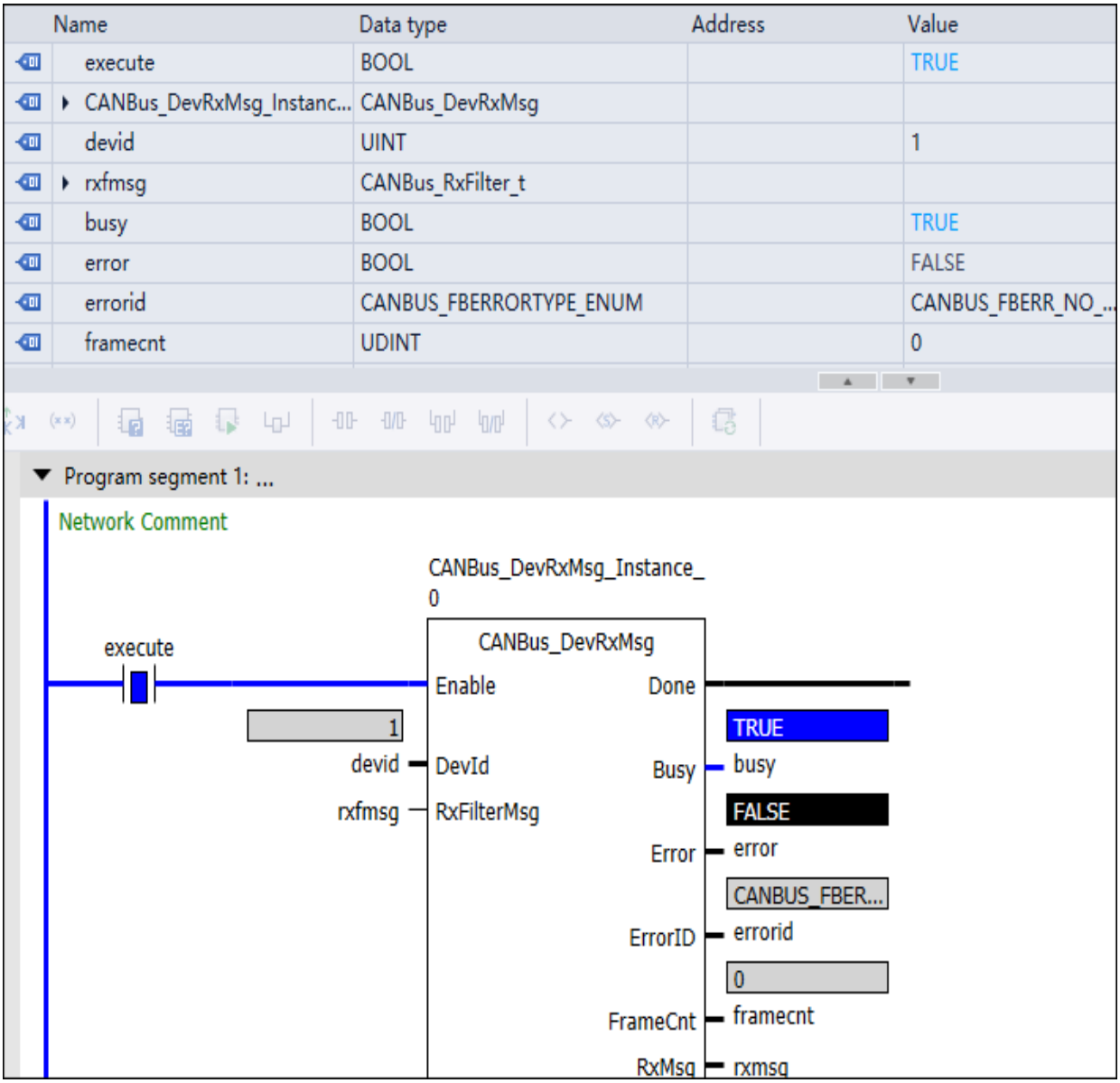
- **Program example**

  ST:

	Name	Data type	Address	Value
	CANBus_DevRxMsg_Instance...	CANBus_DevRxMsg		
	execute	BOOL		TRUE
	devid	UINT		1
	rxfiltermsg	CANBus_RxFilter_t		
	done	BOOL		FALSE
	busy	BOOL		TRUE
	error	BOOL		FALSE
	errorid	CANBUS_FBERRORTYPE_ENUM		CANBUS_FBERR_NO_E...
	framecnt	UDINT		0
	rxmsg	CANBus_FrameMsg_t		

```
CANBus_DevRxMsg_Instance_0(
Enable TRUE  := execute TRUE ,
DevId   1    := devid    1   ,
RxFilterMsg := rxfiltermsg,
Done  FALSE  => done  FALSE ,
Busy  TRUE   => busy  TRUE  ,
Error FALSE  => error FALSE ,
ErrorIDCANBUS_FBERR_NO_ER... => erroridCANBUS_FBERR_NO_ER...,
FrameCnt      0      => framecnt      0      ,
```

  LD:



- **Fault description**

  When an error occurs, find the corresponding error via ErrorID, and locate the root cause of the error.

Fault code	Value	Description	Troubleshooting Method
CANBUS_FBERR_NO_ERROR	0x0000	Function block operates normally or there is no error.	Function block executes normally.
CANBUS_FBERR_EXE_FAILED	0x0001	Function block execution failed.	-
CANBUS_FBERR_INVAILD_DEVICEID	0x0002	Invalid device ID, which is not enabled.	Invalid input DevID.
CANBUS_FBERR_DEVICE_FAULT	0x0003	There is a fault with the device.	The device is starting up.
CANBUS_FBERR_WRONG_PARAMETER	0x0004	The input parameters of the function block are incorrect.	Invalid input CtrlCmd.
CANBUS_FBERR_BUSY_ERROR	0x0005	Function block is busy.	-

Fault code	Value	Description	Troubleshooting Method
CANBUS_FBERR_ABORT_ERROR	0x0006	Function block abort error.	-
CANBUS_FBERR_TIME_OUT_ERROR	0x0007	Function block timeout error.	-
CANBUS_FBERR_POINTER_NULL	0x0008	Function block null pointer error.	-
CANBUS_FBERR_BUFF_FULL	0x0009	Function block buffer full error.	-
CANBUS_FBERR_UNKNOWN_ERROR	0x00FF	Function block unknown error.	-

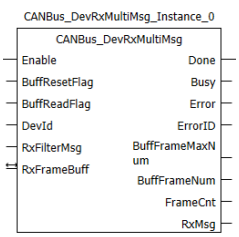
- **Precautions**

  At the same time, a maximum of 128 instantiated receive function blocks are supported, including CANBus_DevRxMsg and CANBus_DevRxMultiMsg.

18.4.10.14 CANBus_DevRxMultiMsg

This instruction allows the device to receive multiple frame messages.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
CANBus_DevRxMultiMsg	Device receives multiple frame messages	FB		CANBus_DevRxMultiMsg_Instance_0(Enable := , BuffResetFlag := , BuffReadFlag := , DevId := , RxFilterMsg := , RxFrameBuff := , Done => , Busy => , Error => , ErrorID => , BuffFrameMaxNum => , BuffFrameNum => , FrameCnt => , RxMsg =>);

- **Related variables**

  Variable

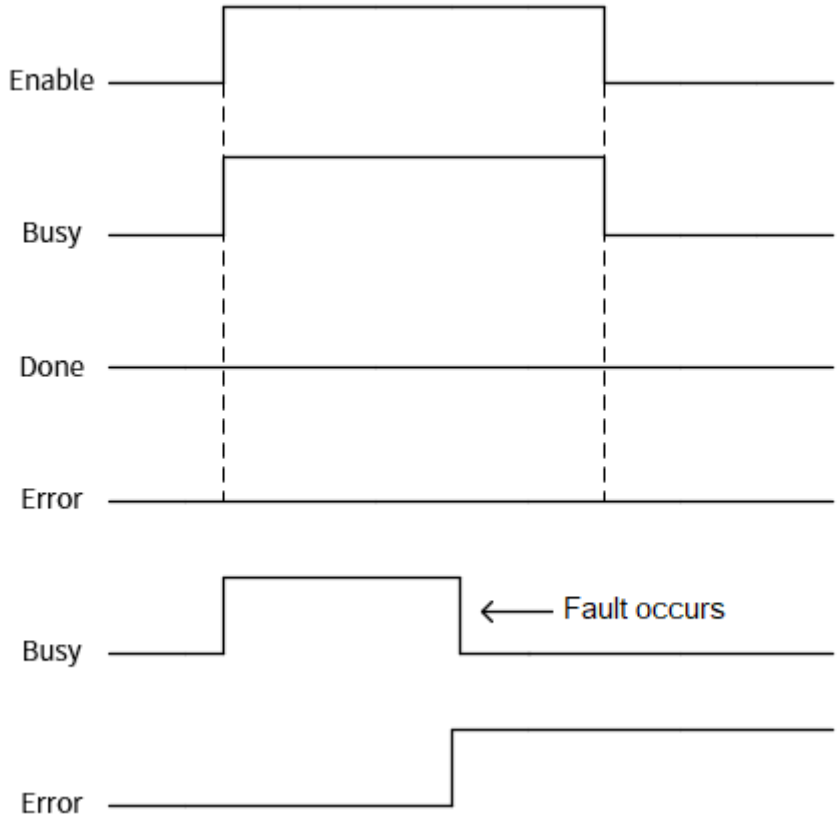
Input / Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Enable	Function block enable flag	BOOL	Follows data type	FALSE	Function block enable flag TRUE: Active at high level FALSE: Stop/Reset at low level
	BuffResetFlag	Buffer reset flag	BOOL	Follows data type	-	Buffer reset flag. TRUE: Clears the buffer. The buffer starts filling from the beginning, and the function block is automatically reset.
	BuffReadFlag	Buffer read flag	BOOL	Follows data type	-	Buffer read flag. TRUE:The application layer reads data from the buffer, preventing the function block from writing to the buffer.
	DevId	Network ID of the CANBus device.	UINT	Follow data type	-	The device ID must correspond to the network ID of the CANBus device.
	RxFilterMsg	Setting of receiving filtered messages	18.4.10 CANBus_RxFilter_t: Structure of CANBus Receive Frame Filter	Follows data type	-	Set receive parameters and filter parameters.
Input / Output	RxFrameBuff	Received frame buffer	ARRAY[*] OF 18.4.10.4 CANBus_FrameMsg_t: Structure of the CANBus frame message	Follows data type	-	A one-dimensional variable-length array of structures, with the number of array elements not exceeding 65,535.
Output	Done	Instruction completion flag	BOOL	Follows data type	FALSE	Function block completion flag.
	Busy	Instruction execution flag	BOOL	Follows data type	FALSE	Function block execution flag.
	Error	Error flag	BOOL	Follows data type	FALSE	Function block error flag.

Input / Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	ErrorID	Fault code	18.4.10.9 CANBus_FBErrorType_Enum: Enumeration of CANBus Function Block Error	Follows data type	-	Fault code of function block. See Fault description.
	BuffFrameMaxNum	Maximum bumber of received buffer frames	UINT	Follow data type	-	Maximum number of received buffer frames. 0: No buffer. >0 - Maximum number of buffered frames.
	BuffFrameNum	Number of received buffer frames	UINT	Follow data type	-	Number of received buffer frames, which is less than BuffFrameMaxNum. Clear and re-count automatically after the buffer is full.
	FrameCnt	Frame counter	UDINT	Follows data type	-	Frame counter that indicates the number of received qualified frames.
	RxMsg	Received messages	18.4.10.4 CANBus_FrameMsg_t: Structure of the CANBus Frame Message	Follows data type	-	Stores the received qualified CAN frame messages. (Only the latest received CAN frame is buffered)

- **Function**

  When the function block is enabled, it receives CAN data from the network number corresponding to DevId. The filtering is determined by RxFilterMsg. Received qualified CAN frames are counted in FrameCnt, and the messages of the latest qualified CAN frame is stored in RxMsg.

- **Sequence diagram**



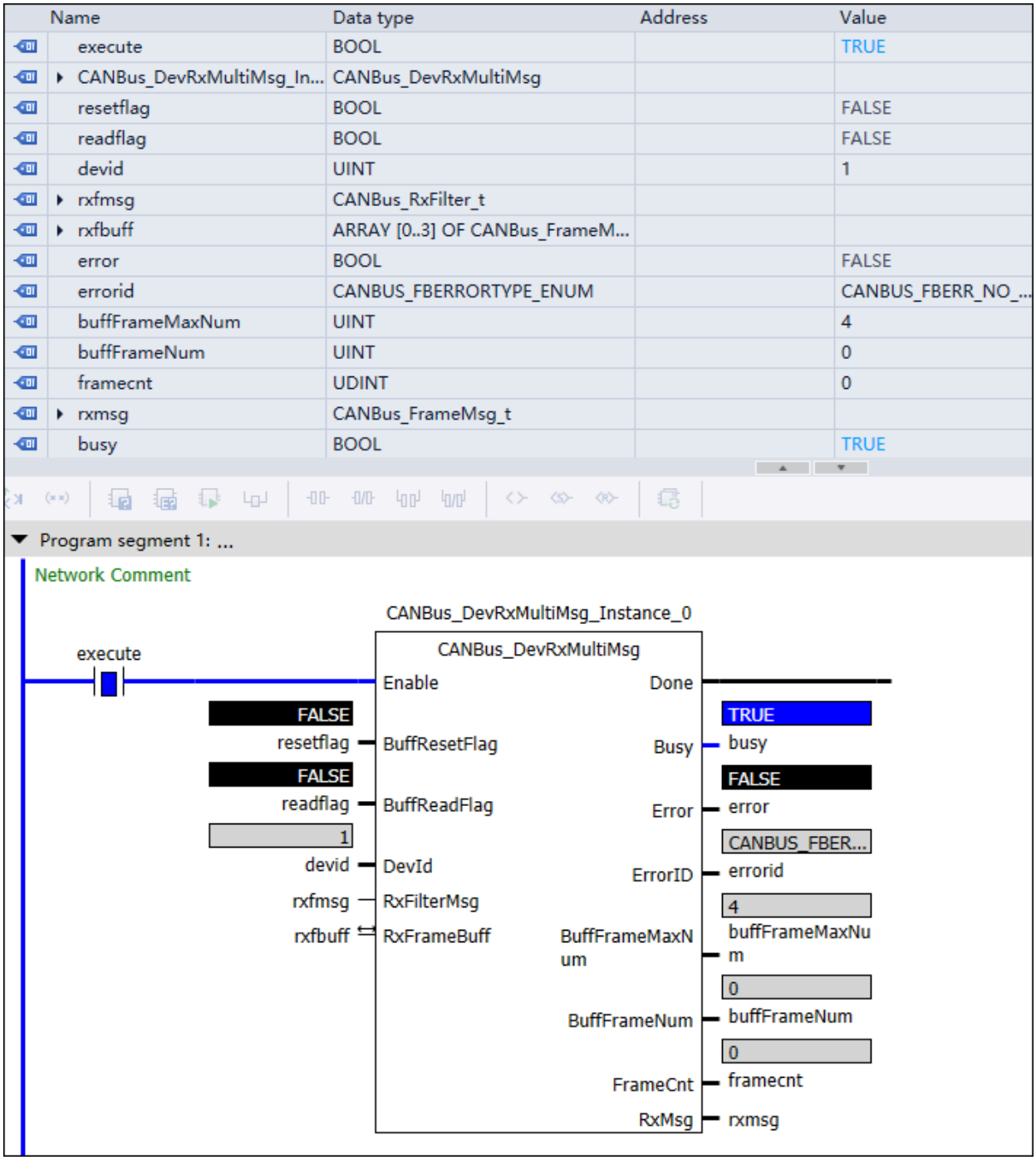
- **Program example**

  ST:

Name	Data type	Address	Value
CANBus_DevRxMultiMsg_In...	CANBus_DevRxMultiMsg		
execute	BOOL		TRUE
resetflag	BOOL		FALSE
readflag	BOOL		FALSE
devid	UINT		1
rxfmsg	CANBus_RxFilter_t		
rxfbuff	ARRAY [0..3] OF CANBus_FrameM...		
done	BOOL		FALSE
busy	BOOL		TRUE
error	BOOL		FALSE
errorid	CANBUS_FBERRORTYPE_ENUM		CANBUS_FBERR_NO_...
buffFrameMaxnum	UINT		4
buffFramenum	UINT		0
framecnt	UDINT		0
rxmsg	CANBus_FrameMsg_t		

```
CANBus_DevRxMultiMsg_Instance_0(
Enable TRUE := execute TRUE,
BuffResetFlag FALSE := resetflag FALSE,
BuffReadFlag FALSE := readflag FALSE,
DevId 1 := devid 1,
RxFilterMsg := rxfmsg,
RxFrameBuff 3009413584 := rxfbuff,
Done FALSE => done FALSE,
Busy TRUE => busy TRUE,
Error FALSE => error FALSE,
ErrorIDCANBUS_FBERR_NO_ER... => erroridCANBUS_FBERR_NO_ER...,
BuffFrameMaxNum 4 => buffFrameMaxnum 4,
BuffFrameNum 0 => buffFramenum 0,
FrameCnt 0 => framecnt 0,
RxMsg => rxmsg);
RETURN
```

LD:



- **Fault description**

When an error occurs, find the corresponding error via ErrorID, and locate the root cause of the error.

Fault code	Value	Description	Troubleshooting Method
CANBUS_FBERR_NO_ERROR	0x0000	Function block operates normally or there is no error.	Function block executes normally.
CANBUS_FBERR_EXE_FAILED	0x0001	Function block execution failed.	-
CANBUS_FBERR_INVAILD_DEVICEID	0x0002	Invalid device ID, which is not enabled.	Invalid input DevID.
CANBUS_FBERR_DEVICE_FAULT	0x0003	There is a fault with the device.	-
CANBUS_FBERR_WRONG_PARAMETER	0x0004	The input parameters of the function block are incorrect.	Maximum buffer size is 0.

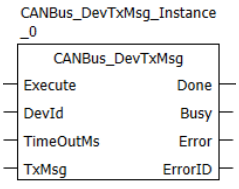
Fault code	Value	Description	Troubleshooting Method
CANBUS_FBERR_BUSY_ERROR	0x0005	Function block is busy.	Function block instance limit exceeded.
CANBUS_FBERR_ABORT_ERROR	0x0006	Function block abort error.	-
CANBUS_FBERR_TIME_OUT_ERROR	0x0007	Function block timeout error.	-
CANBUS_FBERR_POINTER_NULL	0x0008	Function block null pointer error.	Input/output parameters or buffer address not specified.
CANBUS_FBERR_BUFF_FULL	0x0009	Function block buffer full error.	Reception buffer full.
CANBUS_FBERR_UNKNOWN_ERROR	0x00FF	Function block unknown error.	-

- **Precautions**
  - At the same time, a maximum of 128 instantiated receive function blocks are supported, including CANBus_DevRxMsg and CANBus_DevRxMultiMsg.
  - When reading the buffer, the BuffReadFlag flag can be set to prevent new data from overwriting the data in the buffer. It is important to note that during the buffer reading process, the internal register only stores the most recently received message. Therefore, this flag should be set for the shortest possible time and reset in a timely manner to avoid frame loss during reception.

18.4.10.15 CANBus_DevTxMsg

This instruction allows the device to transmit messages.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
CANBus_DevTxMsg	Device transmitting messages	FB		CANBus_DevTxMsg_Instance_0(Execute := , DevId := , TimeOutMs := , TxMsg := , Done => , Busy => , Error => , ErrorID =>);

- **Related variables**

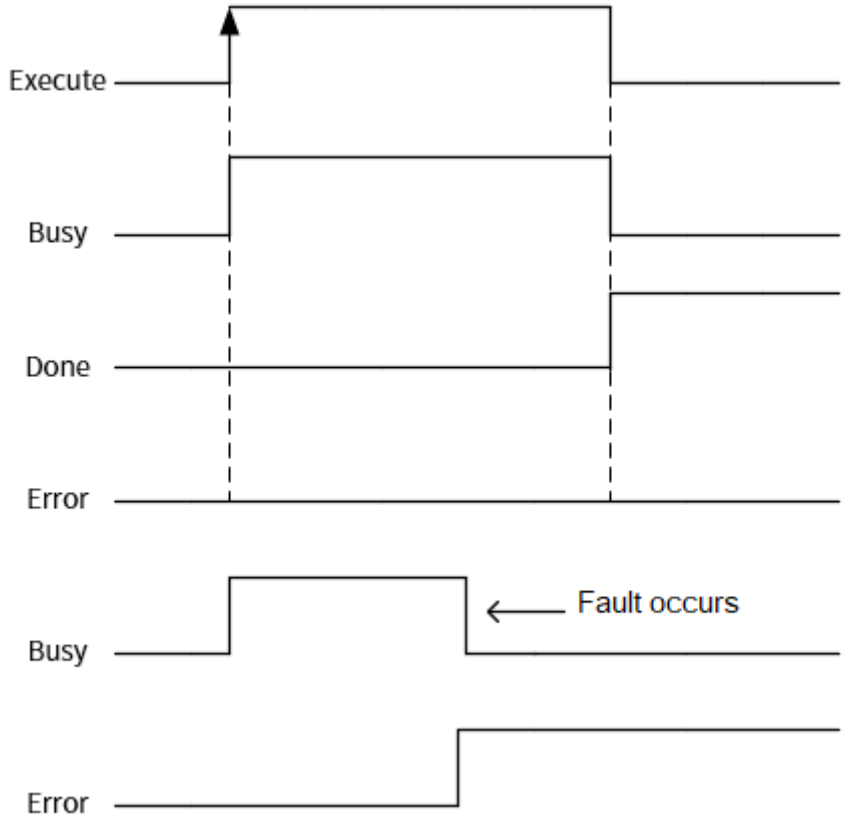
  Variable

Input / Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Function block trigger flag	BOOL	Follows data type	FALSE	Function block trigger flag. TRUE: Triggered on rising edge. FALSE: Stop/reset on falling edge.
	DevId	Network ID of the CANBus device.	UINT	Follow data type	-	The device ID must correspond to the network ID of the CANBus device.
	TimeOutMs	Timeout period	UINT	Follow data type	-	Transmitting timeout in ms.
	TxMsg	Transmit messages	18.4.10.4 CANBus_FrameMsg_t: Structure of the CANBus Frame Message	Follows data type	-	Transmitted message, including the extended frame flag, remote frame flag, CAN ID, data length and data.
Output	Done	Instruction completion flag	BOOL	Follows data type	FALSE	Function block completion flag.
	Busy	Instruction execution flag	BOOL	Follows data type	FALSE	Function block execution flag.
	Error	Error flag	BOOL	Follows data type	FALSE	Function block error flag.
	ErrorID	Fault code	18.4.10.9 CANBus_FBErrorType_Enum: Enumeration of CANBus Function Block Error	Follows data type	-	Fault code of function block. See Fault description.

- **Function**

  When the function block is triggered, it transmits one frame of CAN data to the network number corresponding to the DevId. The content of the transmission is determined by TxMsg.

- **Sequence diagram**



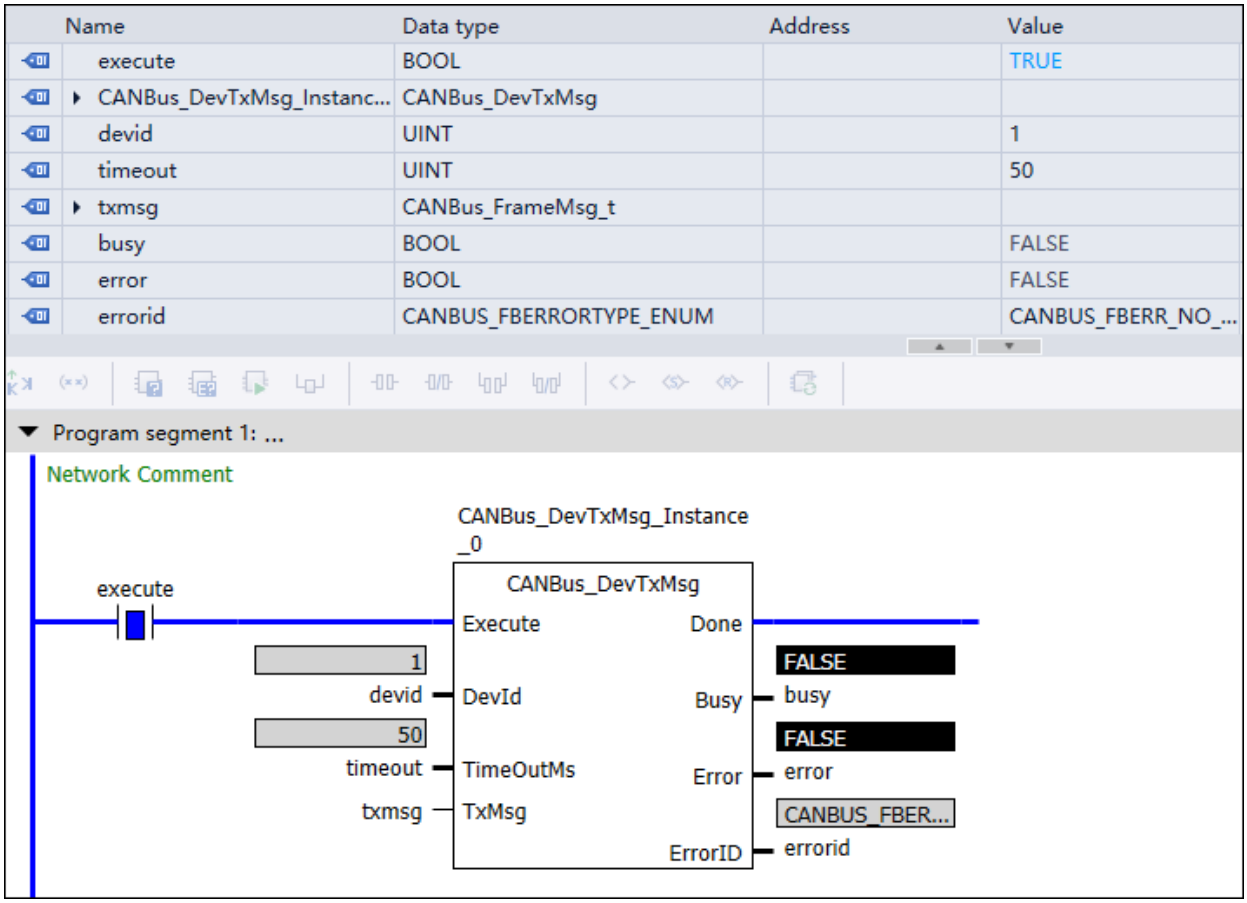
- **Program example**

  ST:

	Name	Data type	Address	Value
	CANBus_DevTxMsg_Instanc...	CANBus_DevTxMsg		
	execute	BOOL		TRUE
	devid	UINT		1
	timeout	UINT		50
	txmsg	CANBus_FrameMsg_t		
	done	BOOL		TRUE
	busy	BOOL		FALSE
	error	BOOL		FALSE
	errorid	CANBUS_FBERRORTYPE_ENUM		CANBUS_FBERR_NO_...

```
CANBus_DevTxMsg_Instance_0(
Execute TRUE := execute TRUE,
DevId 1 := devid 1,
TimeOutMs 50 := timeout 50,
TxMsg := txmsg,
Done TRUE => done TRUE,
Busy FALSE => busy FALSE,
Error FALSE => error FALSE,
ErrorIDCANBUS_FBERR_NO_ER... => erroridCANBUS_FBERR_NO_ER...);RETURN
```

  LD:



- **Fault description**

  When an error occurs, find the corresponding error via ErrorID, and locate the root cause of the error.

Fault code	Value	Description	Troubleshooting Method
CANBUS_FBERR_NO_ERROR	0x0000	Function block operates normally or there is no error.	Function block executes normally.
CANBUS_FBERR_EXE_FAILED	0x0001	Function block execution failed.	-
CANBUS_FBERR_INVAILD_DEVICEID	0x0002	Invalid device ID, which is not enabled.	Invalid input DevID.
CANBUS_FBERR_DEVICE_FAULT	0x0003	There is a fault with the device.	Device is not running or has a fault.
CANBUS_FBERR_WRONG_PARAMETER	0x0004	The input parameters of the function block are incorrect.	CANID is invalid, whosedata length exceeds limit (data frame greater than 8 and remote frame greater than 0).
CANBUS_FBERR_BUSY_ERROR	0x0005	Function block is busy.	Tranmission request conflict.
CANBUS_FBERR_ABORT_ERROR	0x0006	Function block abort error.	-
CANBUS_FBERR_TIME_OUT_ERROR	0x0007	Function block timeout error.	Transmission failed due to timeout.
CANBUS_FBERR_POINTER_NULL	0x0008	Function block null pointer error.	-

Fault code	Value	Description	Troubleshooting Method
CANBUS_FBERR_BUFF_FULL	0x0009	Function block buffer full error.	-
CANBUS_FBERR_UNKNOWN_ERROR	0x00FF	Function block unknown error.	-

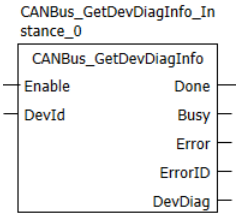
- **Precautions**

  If the timeout period is set to 0, the function block defaults to using a 50ms detection period for transmission timeout during operation.

18.4.10.16 CANBus_GetDevDiagInfo

This instruction gets the diagnostic information of the device.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
CANBus_GetDevDiagInfo	Get device diagnostic information	FB		CANBus_GetDevDiagInfo_Instance_0(Enable := , DevId := , Done => , Busy => , Error => , ErrorID => , DevDiag =>);

- **Related variables**

  Variable

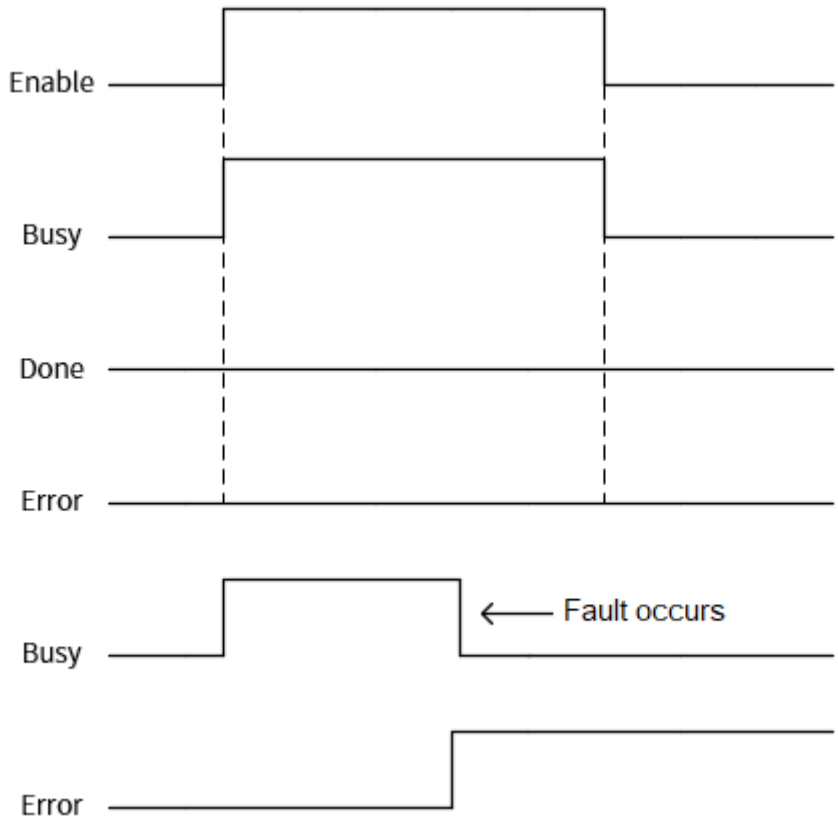
Input / Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Enable	Function block enable flag	BOOL	Follows data type	FALSE	Function block enable flag TRUE: Active at high level FALSE: Stop/Reset at low level
	DevId	Network ID of the CANBus device.	UINT	Follow data type	-	The device ID must correspond to the network ID of the CANBus device.
Output	Done	Instruction completion flag	BOOL	Follows data type	FALSE	Function block completion flag.

Input / Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	Busy	Instruction execution flag	BOOL	Follows data type	FALSE	Function block execution flag.
	Error	Error flag	BOOL	Follows data type	FALSE	Function block error flag.
	Error ID	Fault code	18.4.10.9 CANBus_FBErrorType_Enum: Enumeration of CANBus Function Block Error	Follows data type	-	Fault code of function block. See Fault description.
	DevDiag	Device diagnostic information	18.4.10.3 CANBus_DiagInfo_t: Structure of the CANBus Diagnosis Information	Follows data type	-	Device diagnostic information.

- **Function**

  During the activation of the function block, diagnostic information for the network number corresponding to the DevId is retrieved and saved in DevDiag.

- **Sequence diagram**



- **Program example**

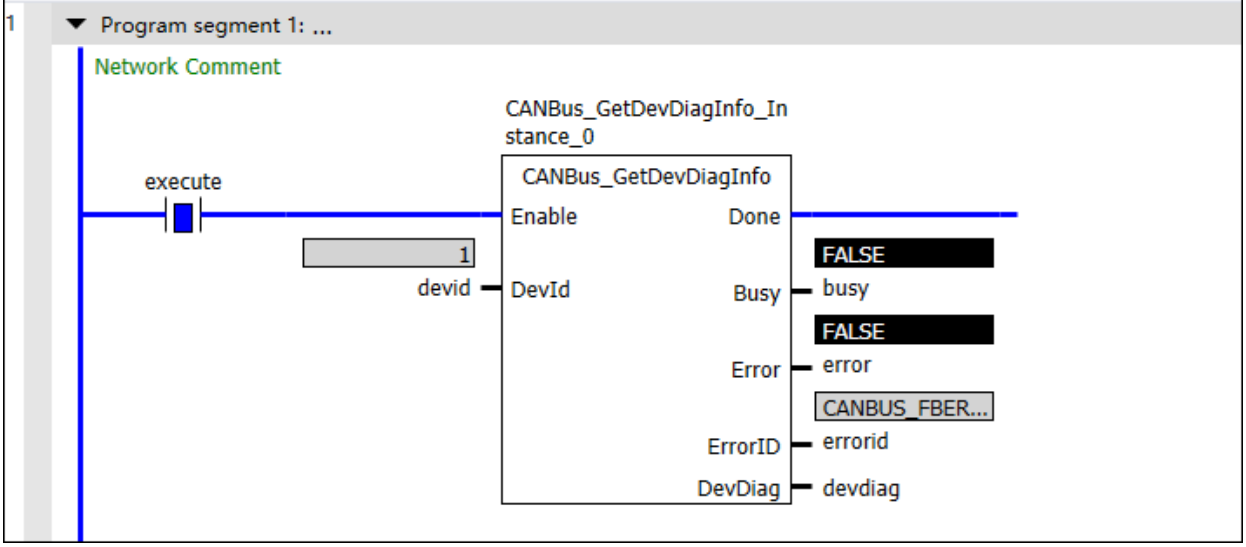
ST:

	Name	Data type	Address	Value
	CANBus_GetDevDiagInfo_In...	CANBus_GetDevDiagInfo		
	enable	BOOL		TRUE
	devid	UINT		1
	done	BOOL		TRUE
	busy	BOOL		FALSE
	error	BOOL		FALSE
	errorid	CANBUS_FBERRORTYPE_ENUM		CANBUS_FBERR_NO_...
	devdiag	CANBus_DiagInfo_t		

```
CANBus_GetDevDiagInfo_Instance_0(
Enable TRUE  := enable TRUE ,
DevId   1    := devid   1   ,
Done  TRUE   => done  TRUE  ,
Busy  FALSE  => busy  FALSE ,
Error FALSE  => error FALSE [TRUE] ,
ErrorIDCANBUS_FBERR_NO_ER... => erroridCANBUS_FBERR_NO_ER...,
DevDiag => devdiag);
RETURN
```

LD:

	Name	Data type	Address	Value
	execute	BOOL		TRUE
	CANBus_GetDevDiagInfo_In...	CANBus_GetDevDiagInfo		
	devid	UINT		1
	busy	BOOL		FALSE
	error	BOOL		FALSE
	errorid	CANBUS_FBERRORTYPE_ENUM		CANBUS_FBERR_NO_...
	devdiag	CANBus_DiagInfo_t		



- **Fault description**

  When an error occurs, find the corresponding error via ErrorID, and locate the root cause of the error.

Fault code	Value	Description	Troubleshooting Method
CANBUS_FBERR_NO_ERROR	0x0000	Function block operates normally or there is no error.	Function block executes normally.
CANBUS_FBERR_EXE_FAILED	0x0001	Function block execution failed.	-
CANBUS_FBERR_INVAILD_DEVICEID	0x0002	Invalid device ID, which is not enabled.	Invalid input DevID.
CANBUS_FBERR_DEVICE_FAULT	0x0003	There is a fault with the device.	-
CANBUS_FBERR_WRONG_PARAMETER	0x0004	The input parameters of the function block are incorrect.	-
CANBUS_FBERR_BUSY_ERROR	0x0005	Function block is busy.	-
CANBUS_FBERR_ABORT_ERROR	0x0006	Function block abort error.	-
CANBUS_FBERR_TIME_OUT_ERROR	0x0007	Function block timeout error.	-
CANBUS_FBERR_POINTER_NULL	0x0008	Function block null pointer error.	-
CANBUS_FBERR_BUFF_FULL	0x0009	Function block buffer full error.	-
CANBUS_FBERR_UNKNOWN_ERROR	0x00FF	Function block unknown error.	-

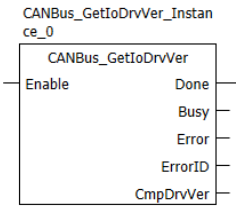
- **Precautions**

  The diagnostic information is updated in real-time and includes, but is not limited to: operational state, error status, DevID, baud rate, bus load rate, and transmit/receive counters.

18.4.10.17 CANBus_GetIoDrvVer

This instruction gets the version information of the CmpDrvCANBus component.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
CANBus_GetIoDrvVer	Get the version information of the CmpDrvCANBus component	FB		CANBus_GetIoDrvVer_Instance_0(Enable := , Done => , Busy => , Error => , ErrorID => , CmpDrvVer =>);

- **Related variables**

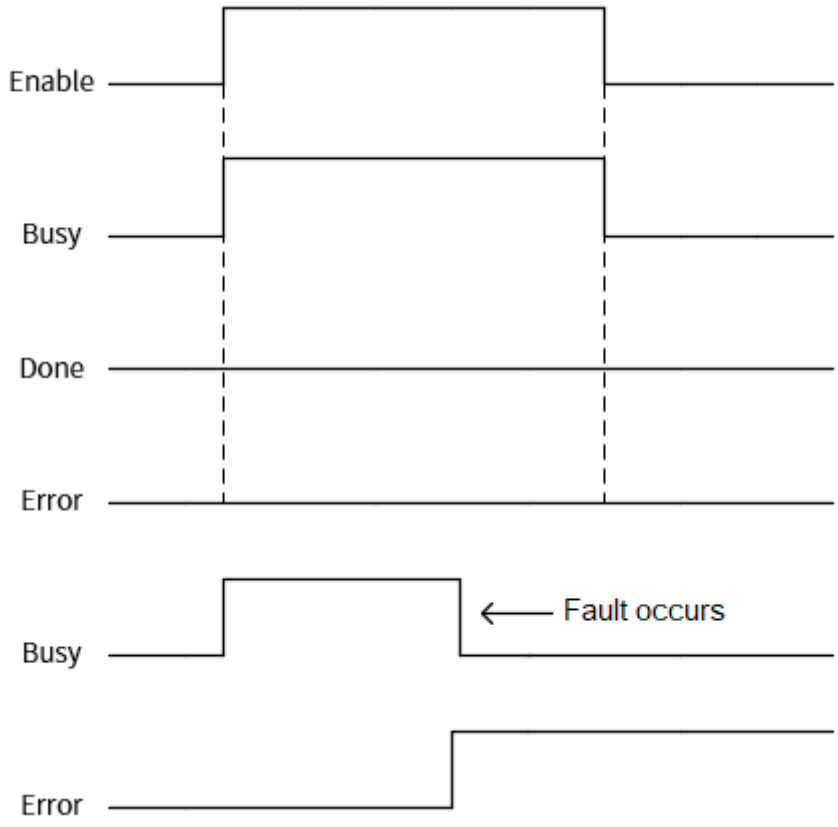
Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Enable	Function block enable flag	BOOL	Follows data type	FALSE	Function block enable flag TRUE: Active at high level FALSE: Stop/Reset at low level
Output	Done	Instruction completion flag	BOOL	Follows data type	FALSE	Function block completion flag.
	Busy	Instruction execution flag	BOOL	Follows data type	FALSE	Function block execution flag.
	Error	Error flag	BOOL	Follows data type	FALSE	Function block error flag.
	ErrorID	Fault code	18.4.10.9 CANBus_FBErrorType_Enum: Enumeration of CANBus Function Block Error	Follows data type	-	Fault code of function block. See Fault description.
	CmpDrvVer	Version information of CANBus component	18.4.10.2 CANBus_CmpDrvVer_t: Version of the Drive Component Used by CANBus	Follows data type	-	Version information of CANBus component.

- **Function**

When the function block is triggered, it retrieves the version number of the CANBus component.

- **Sequence diagram**



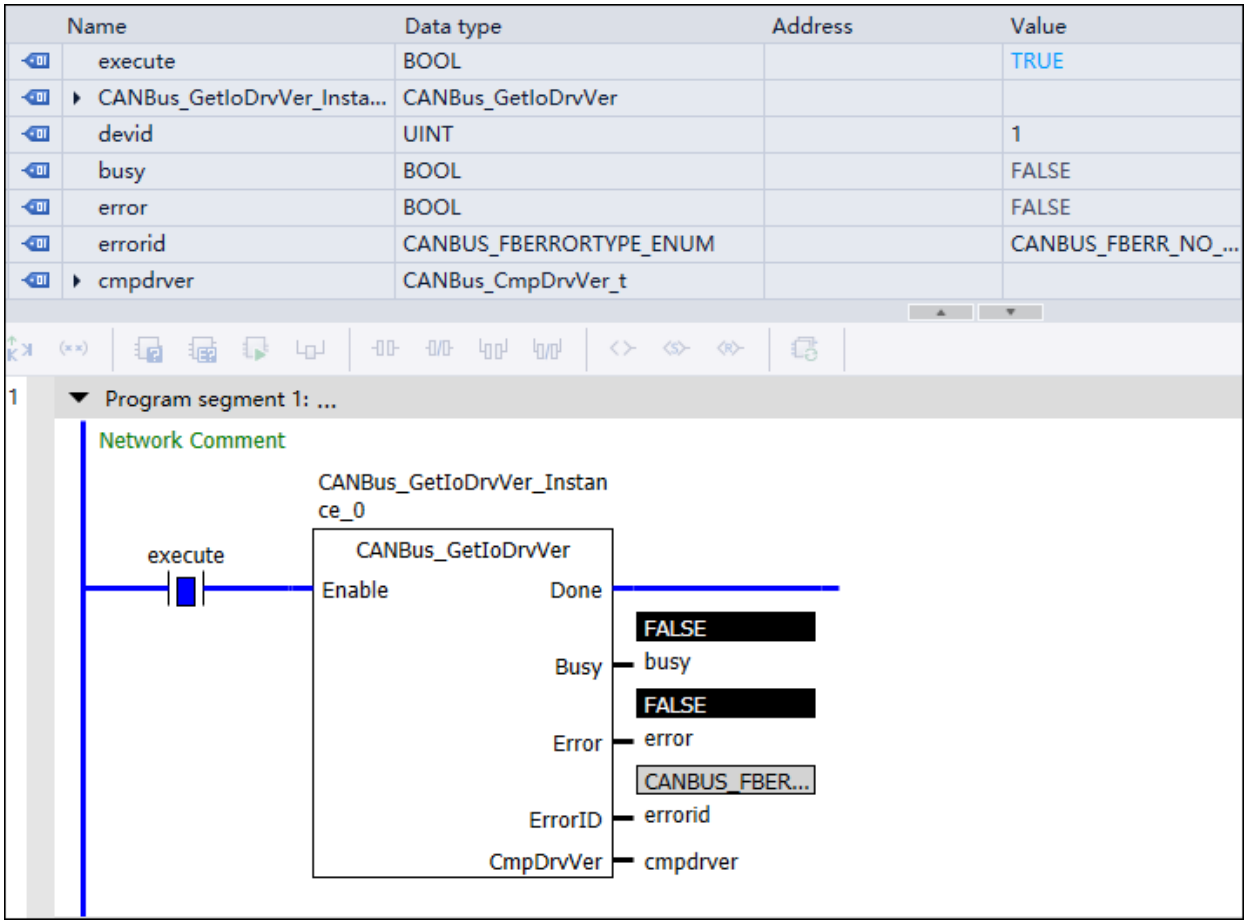
- **Program example**

ST:

	Name	Data type	Address	Value
	CANBus_GetIoDrvVer_Insta...	CANBus_GetIoDrvVer		
	enable	BOOL		TRUE
	done	BOOL		TRUE
	busy	BOOL		FALSE
	error	BOOL		FALSE
	errorid	CANBUS_FBERRORTYPE_ENUM		CANBUS_FBERR_NO_...
	cmpdrver	CANBus_CmpDrvVer_t		

```
CAS FOR WHI REP RET EX REG
CANBus_GetIoDrvVer_Instance_0(
Enable TRUE := enable TRUE,
Done TRUE => done TRUE,
Busy FALSE => busy FALSE,
Error FALSE => error FALSE,
ErrorIDCANBUS_FBERR_NO_ER... => erroridCANBUS_FBERR_NO_ER...,
CmpDrvVer => cmpdrver);
RETURN
```

LD:



- **Fault description**

  When an error occurs, find the corresponding error via ErrorID, and locate the root cause of the error.

Fault code	Value	Description	Troubleshooting Method
CANBUS_FBERR_NO_ERROR	0x0000	Function block operates normally or there is no error.	Function block executes normally.
CANBUS_FBERR_EXE_FAILED	0x0001	Function block execution failed.	-
CANBUS_FBERR_INVAILD_DEVICEID	0x0002	Invalid device ID, which is not enabled.	-
CANBUS_FBERR_DEVICE_FAULT	0x0003	There is a fault with the device.	-
CANBUS_FBERR_WRONG_PARAMETER	0x0004	The input parameters of the function block are incorrect.	-
CANBUS_FBERR_BUSY_ERROR	0x0005	Function block is busy.	-
CANBUS_FBERR_ABORT_ERROR	0x0006	Function block abort error.	-
CANBUS_FBERR_TIME_OUT_ERROR	0x0007	Function block timeout error.	-
CANBUS_FBERR_POINTER_NULL	0x0008	Function block null pointer error.	-

Fault code	Value	Description	Troubleshooting Method
CANBUS_FBERR_BUFF_FULL	0x0009	Function block buffer full error.	-
CANBUS_FBERR_UNKNOWN_ERROR	0x00FF	Function block unknown error.	-

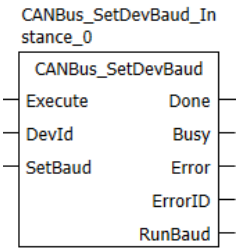
- **Precautions**

  The parameters in the CmpDrvVer structure are uniformly in hexadecimal format.

18.4.10.18 CANBus_SetDevBaud

This instruction sets the baud rate of the device.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
CANBus_SetDevBaud	Set baud rate of device	FB		CANBus_SetDevBaud_Instance_0(Execute := , DevId := , SetBaud := , Done => , Busy => , Error => , ErrorID => , RunBaud =>);

- **Related variables**

Variable

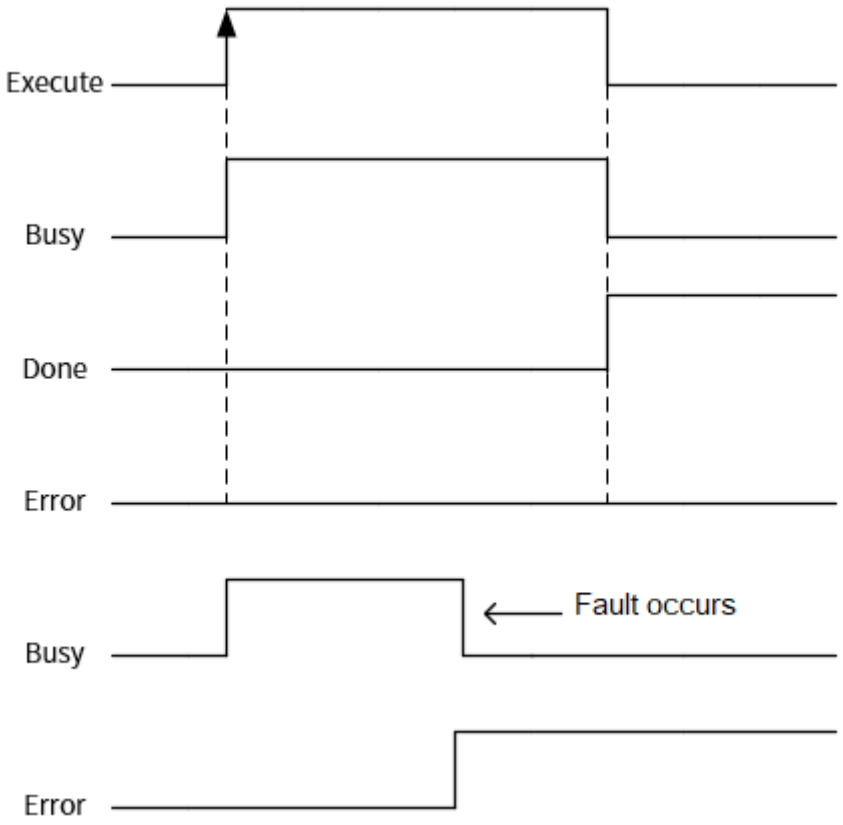
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Function block enable flag	BOOL	Follows data type	FALSE	Function block enable flag TRUE: Active at high level FALSE: Stop/Reset at low level
	DevId	Network ID of the CANBus device.	UINT	Follow data type	-	The device ID must correspond to the network ID of the CANBus device.
	SetBaud	CAN bus setting baud rate	18.4.10.6 CANBus_BaudrateType_Enum: Enumeration of CANBus Baud Rate	Follows data type	-	CAN bus setting baud rate in kbps.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	Done	Instruction completion flag	BOOL	Follows data type	FALSE	Function block completion flag.
	Busy	Instruction execution flag	BOOL	Follows data type	FALSE	Function block execution flag.
	Error	Error flag	BOOL	Follows data type	FALSE	Function block error flag.
	ErrorID	Fault code	18.4.10.9 CANBus_FBErrorType_Enum: Enumeration of CANBus Function Block Error	Follows data type	-	Fault code of function block. See Fault description.
	RunBaud	CAN bus running baud rate	UINT	Follow data type	-	CAN bus running baud rate.

- **Function**

  When the function block is triggered, change the CAN port baud rate of the network number corresponding to DevId.

- **Sequence diagram**



- **Program example**

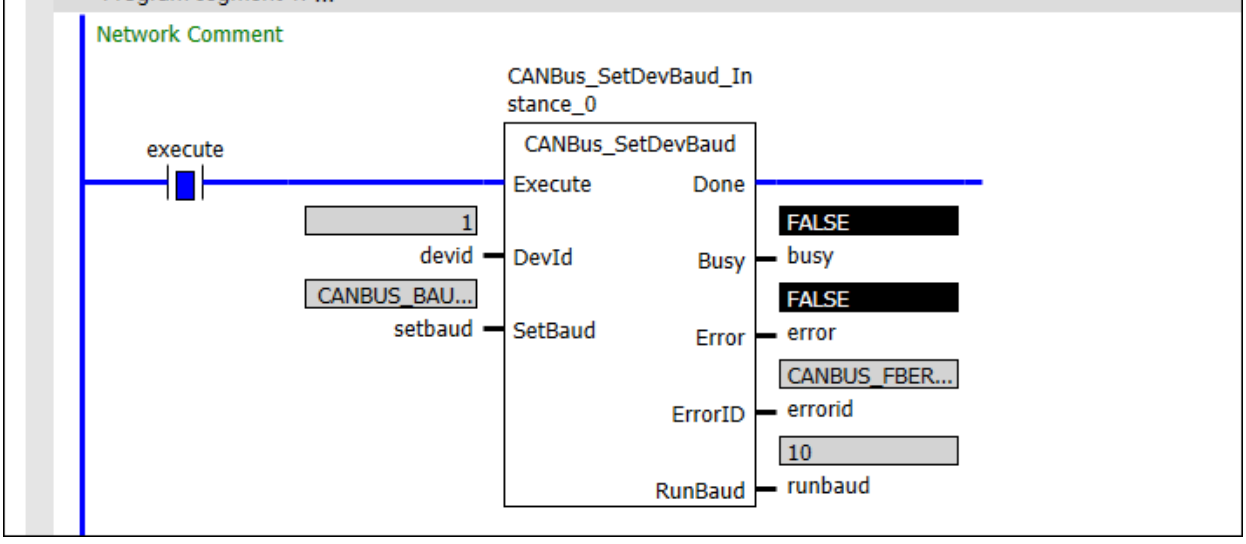
  ST:

	Name	Data type	Address	Value
	CANBus_SetDevBaud_Insta...	CANBus_SetDevBaud		
	execute	BOOL		TRUE
	devid	UINT		1
	setbaud	CANBUS_BAUDRATETYPE_ENUM		CANBUS_BAUDRATE...
	done	BOOL		TRUE
	busy	BOOL		FALSE
	error	BOOL		FALSE
	errorid	CANBUS_FBERRORTYPE_ENUM		CANBUS_FBERR_NO_...
	runbaud	UINT		10

```
CANBus_SetDevBaud_Instance_0(
Execute TRUE := execute TRUE,
DevId 1 := devid 1,
SetBaudCANBUS_BAUDRATE_10K... := setbaudCANBUS_BAUDRATE_10K...,
Done TRUE => done TRUE,
Busy FALSE => busy FALSE,
Error FALSE => error FALSE,
ErrorIDCANBUS_FBERR_NO_ER... => erroridCANBUS_FBERR_NO_ER...,
RunBaud 10 => runbaud 10 );
RETURN
```

LD:

	Name	Data type	Address	Value
	execute	BOOL		TRUE
	CANBus_SetDevBaud_Insta...	CANBus_SetDevBaud		
	devid	UINT		1
	setbaud	CANBUS_BAUDRATETYPE_ENUM		CANBUS_BAUDRATE...
	busy	BOOL		FALSE
	error	BOOL		FALSE
	errorid	CANBUS_FBERRORTYPE_ENUM		CANBUS_FBERR_NO_...
	runbaud	UINT		10



- **Fault description**

  When an error occurs, find the corresponding error via ErrorID, and locate the root cause of the error.

Fault code	Value	Description	Troubleshooting Method
CANBUS_FBERR_NO_ERROR	0x0000	Function block operates normally or there is no error.	Function block executes normally.
CANBUS_FBERR_EXE_FAILED	0x0001	Function block execution failed.	-
CANBUS_FBERR_INVAILD_DEVICEID	0x0002	Invalid device ID, which is not enabled.	Invalid input DevID.
CANBUS_FBERR_DEVICE_FAULT	0x0003	There is a fault with the device.	-
CANBUS_FBERR_WRONG_PARAMETER	0x0004	The input parameters of the function block are incorrect.	Baud rate input error.
CANBUS_FBERR_BUSY_ERROR	0x0005	Function block is busy.	-
CANBUS_FBERR_ABORT_ERROR	0x0006	Function block abort error.	-
CANBUS_FBERR_TIME_OUT_ERROR	0x0007	Function block timeout error.	-
CANBUS_FBERR_POINTER_NULL	0x0008	Function block null pointer error.	-
CANBUS_FBERR_BUFF_FULL	0x0009	Function block buffer full error.	-
CANBUS_FBERR_UNKNOWN_ERROR	0x00FF	Function block unknown error.	-

- **Precautions**
  - The baud rate modification function block currently only supports changing the baud rate of the CAN ports on the PLC body or expansion card. It does not support changing the baud rate of ports expanded by the CAN module.
  - SetBaud sets the CAN bus baud rate. An enumeration type is used to prevent the input of baud rates not supported by the device.

18.4.11 CIA405

18.4.11.1 Instruction List

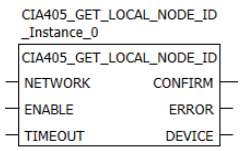
Instruction	FB/FC	Function	Library	Applicable Library Version
18.4.11.2 CIA405_GET_LOCAL_NODE_ID	FB	Get the CANopen node ID of the controller	CIA405	Library version 1.0.0.0 and later
18.4.11.3 CIA405_GET_STATE	FB	Obtain the status of the CANopen device		Library version 1.0.0.0 and later
18.4.11.4 CIA405_NMT	FB	Device NMT state management		Library version 1.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.4.11.5 CIA405_RECV_EMCY_DEV	FB	Get device EMCY message		Library version 1.0.0.0 and later
18.4.11.6 CIA405_SDO_READ4	FB	Read up to 4 bytes of a CANopen object		Library version 1.0.0.0 and later
18.4.11.7 CIA405_SDO_WRITE4	FB	Write up to 4 bytes of a CANopen object		Library version 1.0.0.0 and later
18.4.11.8 COP_GET_MST_STATISTICS	FB	Obtain statistics		Library version 1.0.0.0 and later

18.4.11.2 CIA405_GET_LOCAL_NODE_ID

This instruction gets the CANopen node ID of the controller.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
CIA405_GET_LOCAL_NODE_ID	Get the CANopen node ID of the controller	FB		CIA405_GET_LOCAL_NODE_ID_Instance_0(NETWORK := , ENABLE := , TIMEOUT := , CONFIRM => , ERROR => , DEVICE =>);

- **Related variables**

Variable

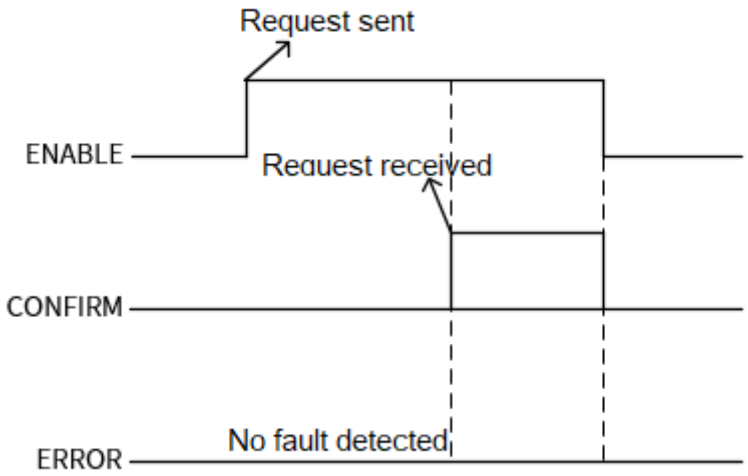
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	NETWORK	CAN network number	USINT	Follows data type	-	The CAN port executed by the function block corresponds to the CAN port number of the host.
	ENABLE	Enable	BOOL	Follows data type	FALSE	Level-triggered, controls the ON/OFF execution of the function block.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	TIMEOUT	Maximum execution time	UDINT	Follows data type	-	Maximum execution time in ms. If the timeout is reached before a response is received, execution will be aborted with a timeout error. 0 (factory setting) = Disable timeout. 1...65535 = Timeout value.
Output	CONFIRM	FB completion flag	BOOL	Follows data type	FALSE	Function block completion flag.
	ERROR	Function block error message	CIA405_CANOPEN_KERNEL_ERROR	Follows data type	-	Function block error message. For details, see 18.4.11.9 Fault Codes.
	DEVICE	CANopen node ID of the controller	CIA405_DEVICE	Follows data type	-	CANopen node ID of the controller. 0 = Invalid. 1...127 = Controller node ID.

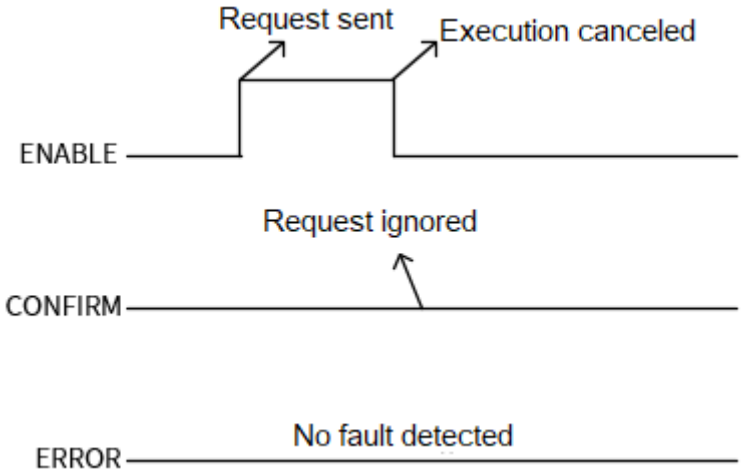
- **Function**

  This function block returns the CANopen node ID of the controller on the specified CAN bus interface.

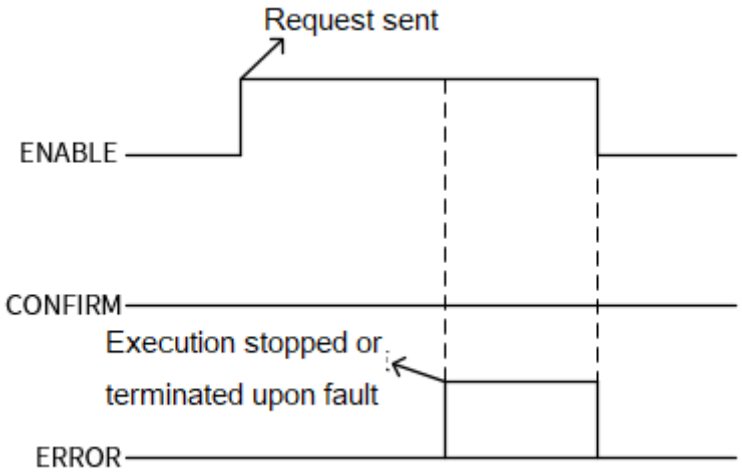
- **Sequence diagram**
  - If a response is generated for the current request and no errors are detected, the CONFIRM is immediately set to TRUE and remains TRUE (provided that the function block is called with ENABLE = TRUE). If the function block is called with ENABLE is reset to FALSE, the CONFIRM will be reset to FALSE, and the function block can restart execution.



  - If the function block is called with ENABLE reset to FALSE before the current execution completes, the execution will be canceled. Responses to cancellation requests may have already been sent or will arrive later, but these responses will be ignored at this time.



- If the current execution is aborted due to an SDO abort message or completes due to an error detected, the ERROR will be set to a non-zero value (for more information on the detected fault codes, see 18.4.11.9 Fault Codes). If the function block is called with ENABLE reset to FALSE, the ERROR will be reset to 0, and the function block can restart execution.



- **Program example**

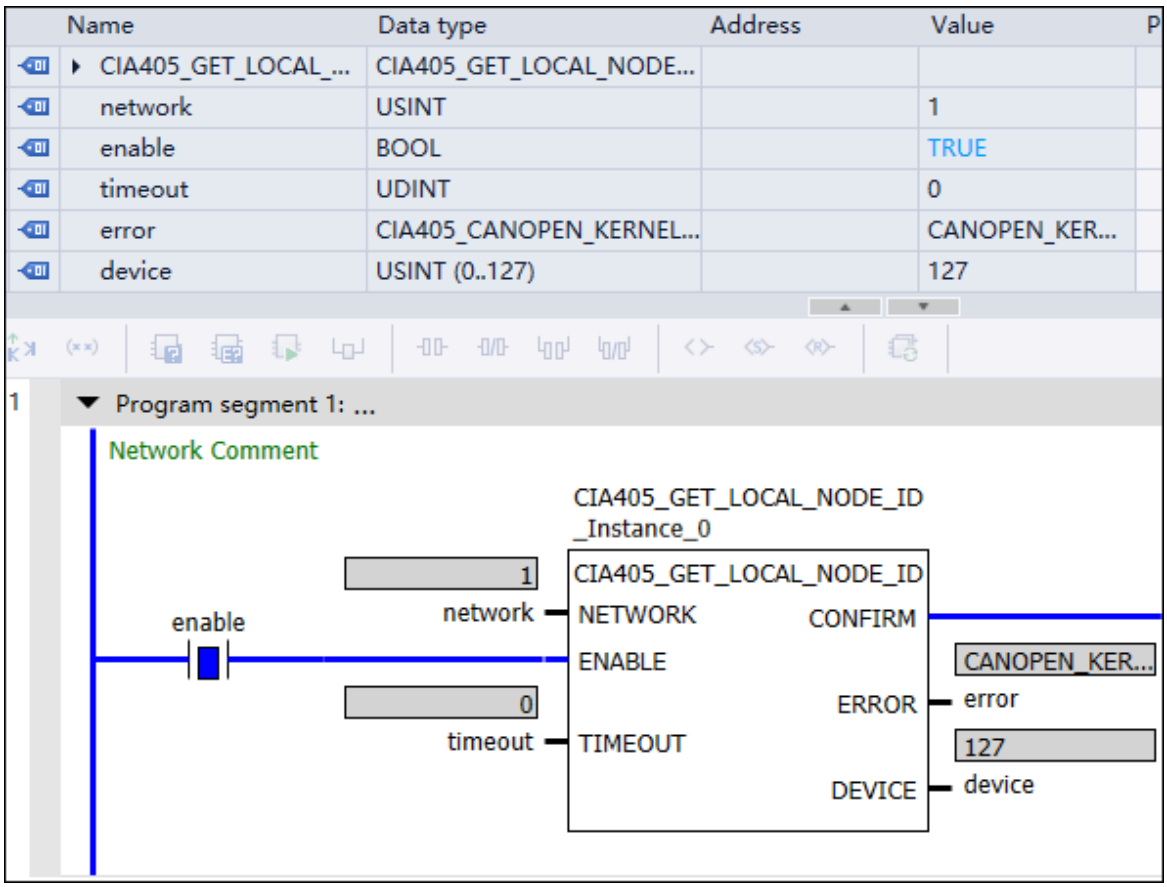
  ST:

Name	Data type	Address	Value
CIA405_GET_LOCAL_...	CIA405_GET_LOCAL_NODE...		
network	USINT		1
enable	BOOL		TRUE
timeout	UDINT		50
confirm	BOOL		TRUE
error	CIA405_CANOPEN_KERNEL...		CANOPEN_KER...
device	USINT (0..127)		127

```
CIA405_GET_LOCAL_NODE_ID_Instance_0(
NETWORK  1     := network  1     ,
ENABLE   TRUE  := enable   TRUE  ,
TIMEOUT  50    := timeout  50    ,
CONFIRM  TRUE  => confirm  TRUE  ,
ERRORCANOPEN_KERNEL_NO_ER... => errorCANOPEN_KERNEL_NO_ER...,
DEVICE   127   => device   127   );
RETURN
```

LD:

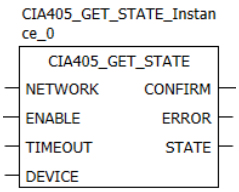
Name	Data type	Address	Value
CIA405_GET_LOCAL_...	CIA405_GET_LOCAL_NODE...		
network	USINT		1
enable	BOOL		TRUE
timeout	UDINT		0
error	CIA405_CANOPEN_KERNEL...		CANOPEN_KER...
device	USINT (0..127)		127



18.4.11.3 CIA405_GET_STATE

This instruction obtains the status of the CANopen device.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
CIA405_GET_STATE	Obtain the status of the CANopen device	FB		CIA405_GET_STATE_Instance_0(NETWORK := , ENABLE := , TIMEOUT := , DEVICE := , CONFIRM => , ERROR => , STATE =>);

- **Related variables**

  Variable

Input / Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	NETWORK	CAN network number	USINT	Follows data type	-	The CAN port executed by the function block corresponds to the CAN port number of the host.
	ENABLE	Enable	BOOL	Follows data type	FALSE	Level-triggered, controls the ON/OFF execution of the function block.
	TIMEOUT	Maximum execution time	UDINT	Follows data type	-	Maximum execution time in ms. If the timeout is reached before a response is received, execution will be aborted with a timeout error. 0 (factory setting) = Disable timeout. 1...65535 = Timeout value.
	DEVICE	CANopen node ID of the controller	CIA405_DEVICE	Follows data type	-	CANopen node ID of the controller. 0 = Invalid. 1...127 = Controller node ID.
Output	CONFIRM	FB completion flag	BOOL	Follows data type	FALSE	Function block completion flag.
	ERROR	Function block error message	CIA405_CANOPEN_KERNEL_ERROR	Follows data type	-	Function block error message. For details, see 18.4.11.9 Fault Codes.

Input / Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	STATE	CANopen kernel state	CIA405_DEVICE_STATE	Follows data type	-	Current state of the CANopen kernel. For details, see Function description.

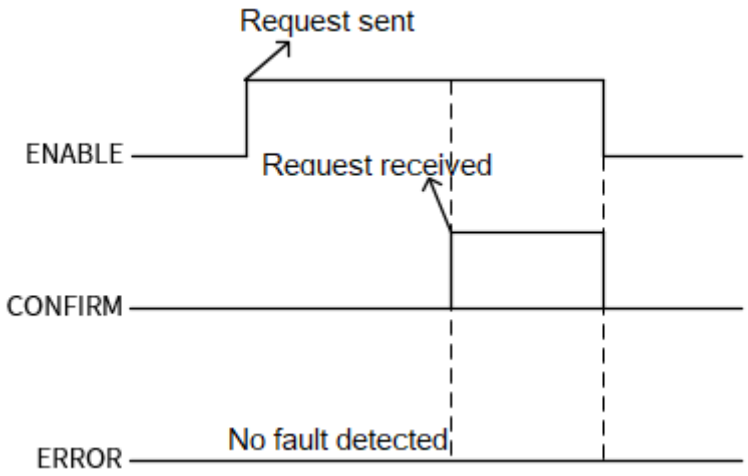
- **Function**

  This instruction returns the current NMT state of the specified CANopen device when either heartbeat or node protection is active.

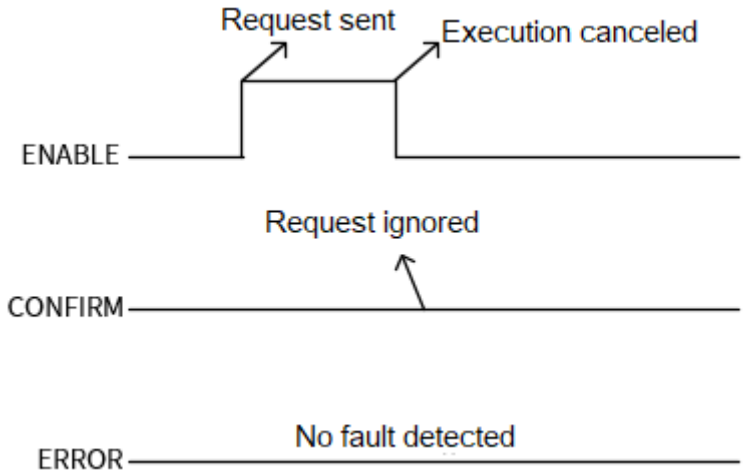
  The data type of "STATE" is the CIA405_Device_STATE enumeration, with enumeration values defined as follows.

Variable	Value	Description
INIT	0	Initializing
RESET_COMM	1	Reset communication
RESET_APP	2	Reset application
PRE_OPERATION	3	Pre-operational
STOPPED	4	Stopped
OPERATIONAL	5	Normal operation
UNKNOWN	6	Unknown NMT Node protection or heartbeat of the selected device is not active, or the controller is not a heartbeat consumer.
NOT_AVAIL	7	NMT not available Node protection or heartbeat of the selected device is active, but the device did not correctly report its NMT state before the timeout.

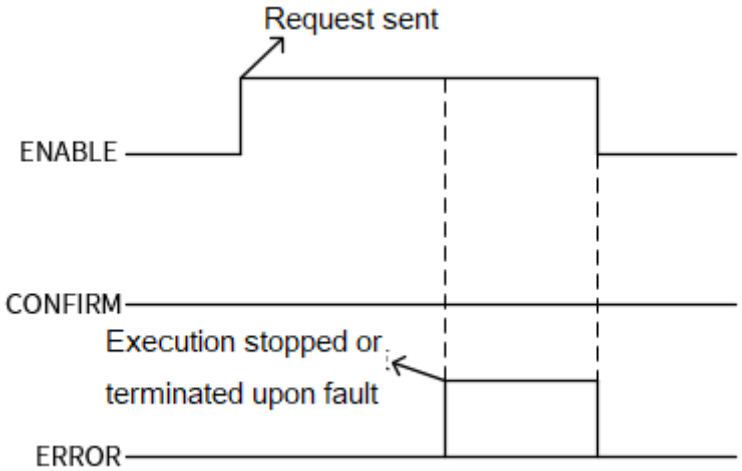
- **Sequence diagram**
  - If a response is generated for the current request and no errors are detected, the CONFIRM is immediately set to TRUE and remains TRUE (provided that the function block is called with ENABLE = TRUE). If the function block is called with ENABLE is reset to FALSE, the CONFIRM will be reset to FALSE, and the function block can restart execution.



◦ If the function block is called with ENABLE reset to FALSE before the current execution completes, the execution will be canceled. Responses to cancellation requests may have already been sent or will arrive later, but these responses will be ignored at this time.



◦ If the current execution is aborted due to an SDO abort message or completes due to an error detected, the ERROR will be set to a non-zero value (for more information on the detected fault codes, see 18.4.11.9 Fault Codes). If the function block is called with ENABLE reset to FALSE, the ERROR will be reset to 0, and the function block can restart execution.



- **Program example**

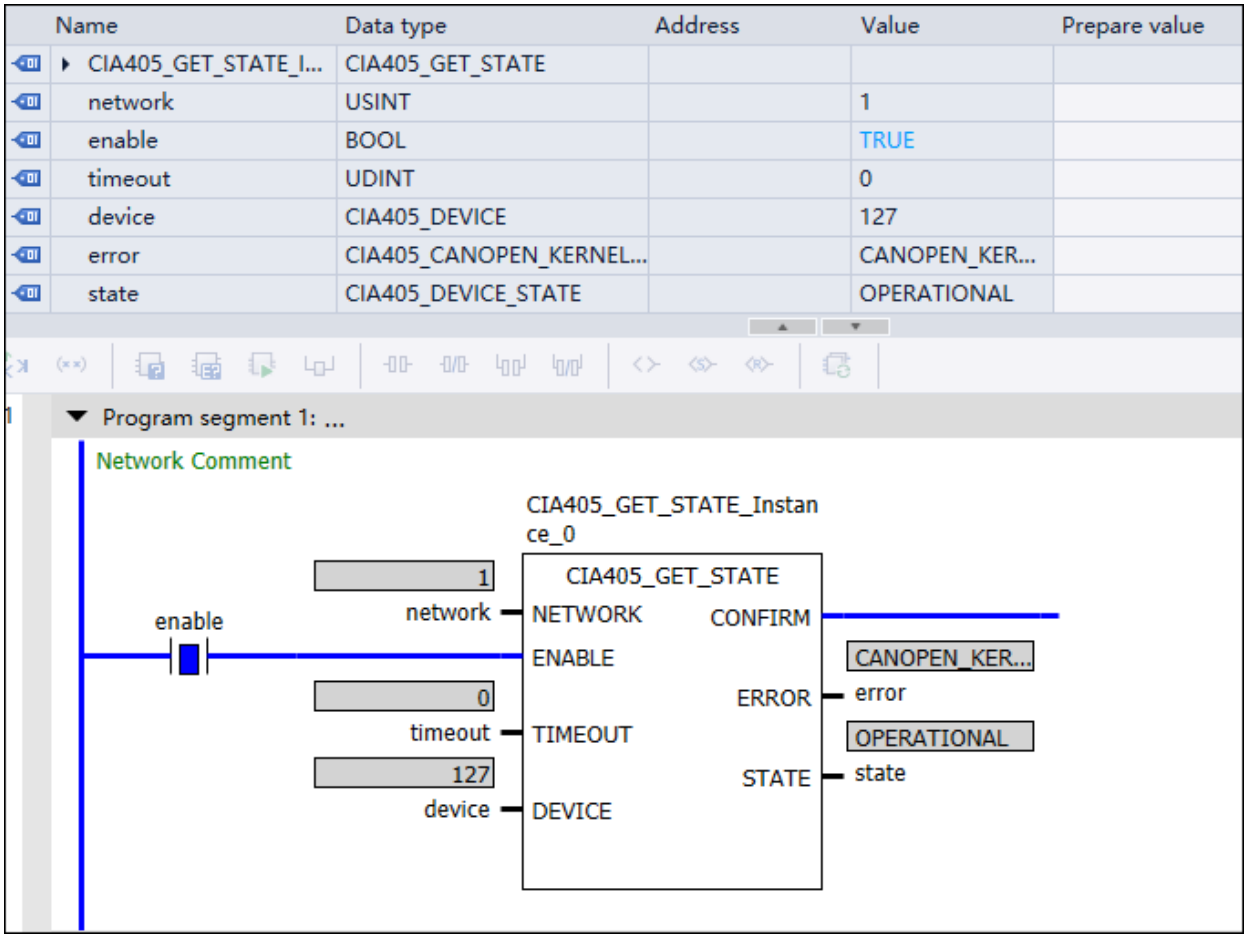
  ST:

Name	Data type	Address	Value	Prepare value
CIA405_GET_STATE_I...	CIA405_GET_STATE			
network	USINT		1	
enable	BOOL		TRUE	
timeout	UDINT		50	
device	CIA405_DEVICE		127	
confirm	BOOL		TRUE	
error	CIA405_CANOPEN_KERNEL...		CANOPEN_KER...	
state	CIA405_DEVICE_STATE		OPERATIONAL	

```
CIA405_GET_STATE_Instance_0(
NETWORK   1     := network   1     ,
ENABLE  TRUE    := enable  TRUE   ,
TIMEOUT       50          := timeout       50         ,
DEVICE  127     := device  127    ,
CONFIRM  TRUE   => confirm  TRUE  ,
ERRORCANOPEN_KERNEL_NO_ER... => errorCANOPEN_KERNEL_NO_ER...,
STATEOPERATIO... => stateOPERATIO...);
RETURN
```

LD:

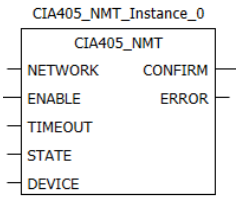
Name	Data type	Address	Value	Prepare value
CIA405_GET_STATE_I...	CIA405_GET_STATE			
network	USINT		1	
enable	BOOL		TRUE	
timeout	UDINT		0	
device	CIA405_DEVICE		127	
error	CIA405_CANOPEN_KERNEL...		CANOPEN_KER...	
state	CIA405_DEVICE_STATE		OPERATIONAL	



18.4.11.4 CIA405_NMT

This instruction involves device NMT state management.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
CIA405_NMT	Device NMT state management	FB		CIA405_NMT_Instance_0(NETWORK := , ENABLE := , TIMEOUT := , STATE := , DEVICE := , CONFIRM => , ERROR =>);

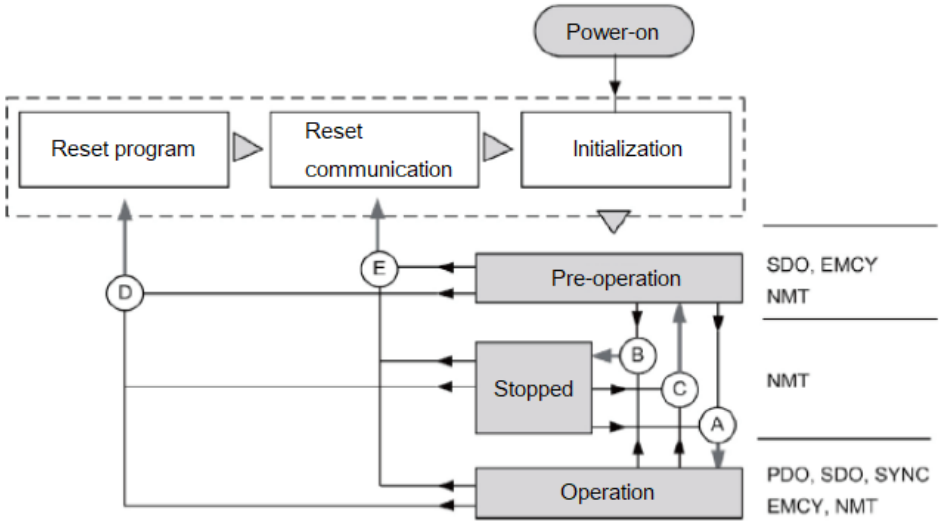
- **Related variables**

Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	NETWORK	CAN network number	USINT	Follows data type	-	The CAN port executed by the function block corresponds to the CAN port number of the host.
	ENABLE	Enable	BOOL	Follows data type	FALSE	Level-triggered, controls the ON/OFF execution of the function block.
	TIMEOUT	Maximum execution time	UDINT	Follows data type	-	Maximum execution time in ms. If the timeout is reached before a response is received, execution will be aborted with a timeout error. 0 (factory setting) = Disable timeout. 1...65535 = Timeout value.
	STATE	Requested NMT state	CIA405_TRANSITION_STATE	Follows data type	-	Requested NMT status transition. For details, see Function description.
	DEVICE	Target device node ID	CIA405_DEVICE	Follows data type	-	CANopen target device node ID. 0 = All NMT slave devices. 1...127 = Target device node ID.
Output	CONFIRM	FB completion flag	BOOL	Follows data type	FALSE	Function block completion flag.
	ERROR	Function block error message	CIA405_CANOPEN_KERNEL_ERROR	Follows data type	-	Function block error message. For details, see 18.4.11.9 Fault Codes.

- **Function**

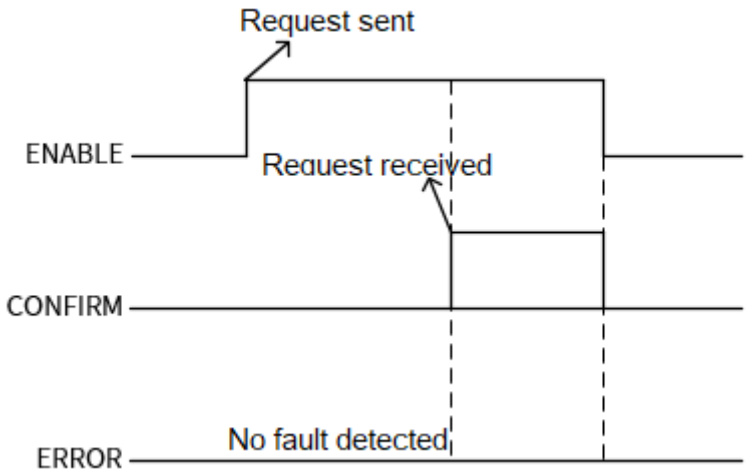
The NMT state machine describes the initialization and state of NMT slaves during main operations. The figure below shows the NM states, the associated available communication objects (PDO, SDO, SYNC, EMCY, and NMT), and the five state transitions (A to E).



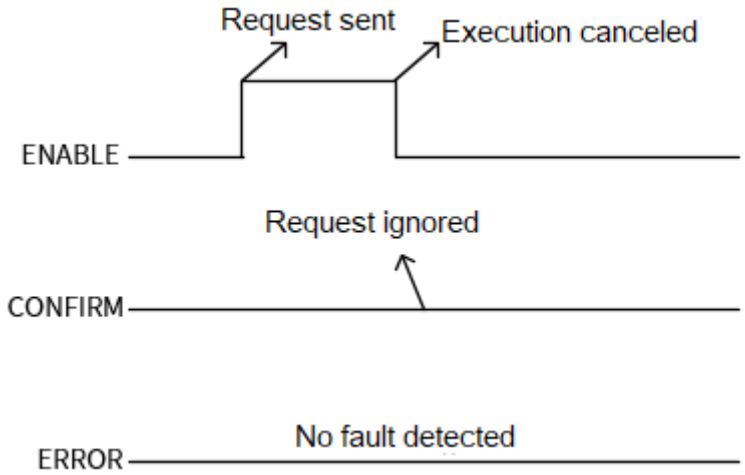
The data type of "STATE" is the CIA405_TRANSITION_STATE enumeration, with enumeration values defined as follows.

Variable Name	Value	Description
STOP_REMOTE_NODE	4	Switch to stop state. (Transition B)
START_REMOTE_NODE	5	Switch to operational state. (Transition A)
RESET_NODE	6	Switch to the reset application state. Load the saved data from the device configuration file and automatically switch from reset communication to pre-operational state. (Transition D)
RESET_COMMUNICATION	7	Switch to the reset communication state. Load the persisted data from the communication configuration file and automatically switch to the pre-operational state. (Transition E)
ENTER_PRE_OPERATIONAL	127	Switch to the pre-operational state. (Transition C)
ALL_EXCEPT_NMT_AND_SENDER	2048	Not implemented. (Invalid parameter)

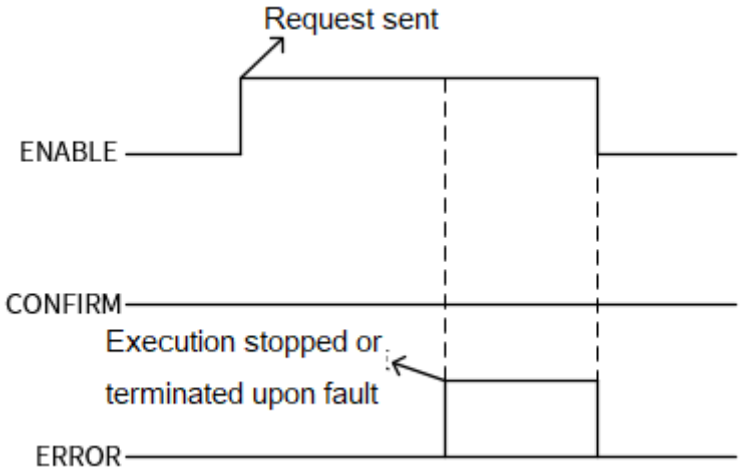
- **Sequence diagram**
  - If a response is generated for the current request and no errors are detected, the CONFIRM is immediately set to TRUE and remains TRUE (provided that the function block is called with ENABLE = TRUE). If the function block is called with ENABLE is reset to FALSE, the CONFIRM will be reset to FALSE, and the function block can restart execution.



- If the function block is called with ENABLE reset to FALSE before the current execution completes, the execution will be canceled. Responses to cancellation requests may have already been sent or will arrive later, but these responses will be ignored at this time.



- If the current execution is aborted due to an SDO abort message or completes due to an error detected, the ERROR will be set to a non-zero value (for more information on the detected fault codes, see 18.4.11.9 Fault Codes). If the function block is called with ENABLE reset to FALSE, the ERROR will be reset to 0, and the function block can restart execution.



- **Program example**

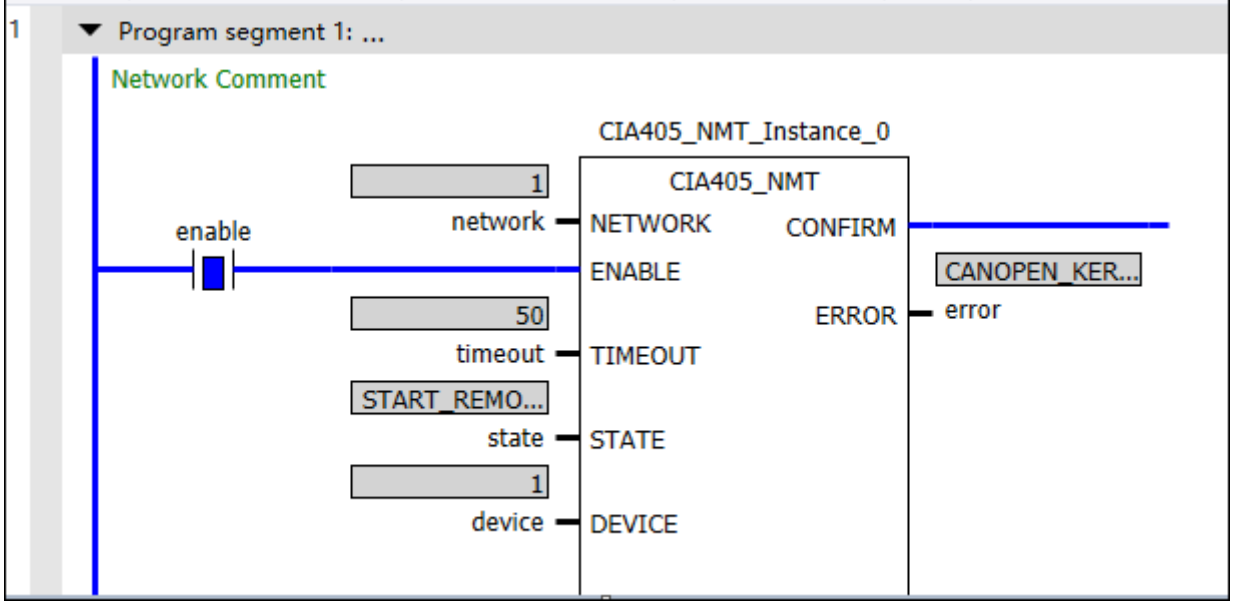
  ST:

Name	Data type	Address	Value
CIA405_NMT_Instance_0	CIA405_NMT		
network	USINT		1
enable	BOOL		TRUE
timeout	UDINT		50
state	CIA405_TRANSITION_STATE		START_REMOTE_...
device	CIA405_DEVICE		1
confirm	BOOL		TRUE
error	CIA405_CANOPEN_KERNEL_...		CANOPEN_KERN...

```
CIA405_NMT_Instance_0(
NETWORK  1     := network  1     ,
ENABLE   TRUE  := enable   TRUE  ,
TIMEOUT  50    := timeout  50    ,
STATESTART_REMOTE_N... := stateSTART_REMOTE_N...,
DEVICE   1     := device   1     ,
CONFIRM  TRUE  => confirm  TRUE  ,
ERRORCANOPEN_KERNEL_NO_ER... => errorCANOPEN_KERNEL_NO_ER...);RETURN
```

LD:

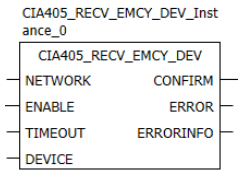
Name	Data type	Address	Value
CIA405_NMT_Instance_0	CIA405_NMT		
network	USINT		1
enable	BOOL		TRUE
timeout	UDINT		50
state	CIA405_TRANSITION_STATE		START_REMOTE_...
device	CIA405_DEVICE		1
error	CIA405_CANOPEN_KERNEL_...		CANOPEN_KERN...



18.4.11.5 CIA405_RECV_EMCY_DEV

This instruction get device EMCY message.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
CIA405_RECV_EMCY_DEV	Get device EMCY message	FB		CIA405_RECV_EMCY_DEV_Instance_0(NETWORK := , ENABLE := , TIMEOUT := , DEVICE := , CONFIRM => , ERROR => , ERRORINFO =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	NETWORK	CAN network number	USINT	Follows data type	-	The CAN port executed by the function block corresponds to the CAN port number of the host.
	ENABLE	Enable	BOOL	Follows data type	FALSE	Level-triggered, controls the ON/OFF execution of the function block.
	TIMEOUT	Maximum execution time	UDINT	Follows data type	-	Maximum execution time in ms. If the timeout is reached before a response is received, execution will be aborted with a timeout error. 0 (factory setting) = Disable timeout. 1...65535 = Timeout value. **Currently not supported for use.**
	DEVICE	Target device node ID	CIA405_DEVICE	Follows data type	-	CANopen target device node ID. 0 = All NMT slave devices. 1...127 = Target device node ID.
Output	CONFIRM	FB completion flag	BOOL	Follows data type	FALSE	Function block completion flag.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	ERROR	Function block error message	CIA405_CANOPEN_KERNEL_ERROR	Follows data type	-	Function block error message. For details, see 18.4.11.9 Fault Codes.
	ERROR INFO	EMCY information	CIA405_EMCY_ERROR	Follows data type	-	EMCY information

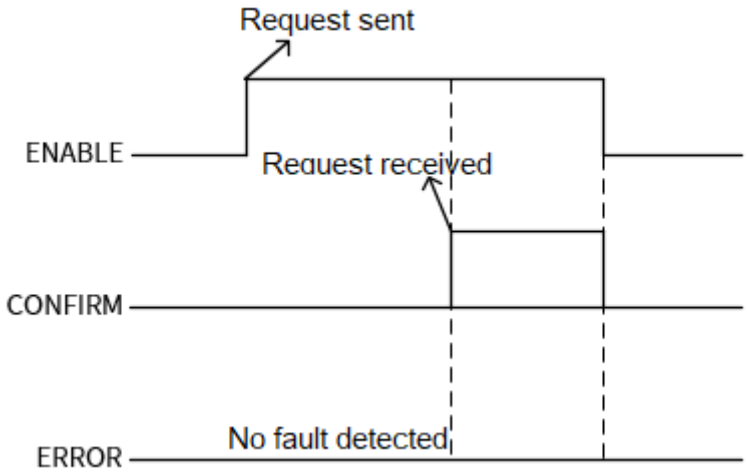
- **Function**

  This function block returns the last stored emergency (EMCY) message received from the specified CANopen device.

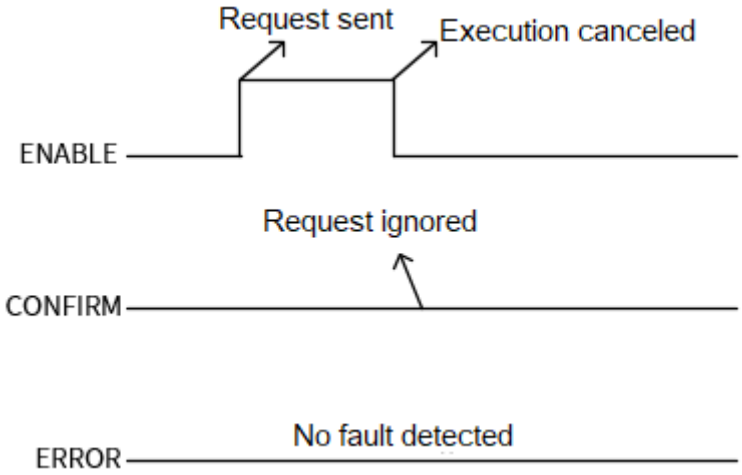
  "ERRORINFO" is of the structured data type CIA405_EMCY_ERROR, with the meanings of the structure members defined as follows.

Variable Name	Data Type	Description
EMCY_ERROR_CODE	WORD	Error code of the EMCY message.
ERROR_REGISTER	BYTE	Error register (bit field) of the EMCY message.
ERROR_FIELD	ARRAY [0..4] OF BYTE	Manufacturer-specific error field of the EMCY message.

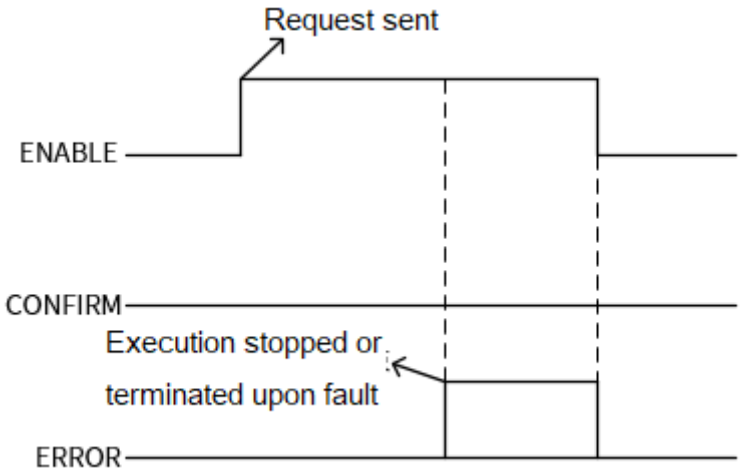
- **Sequence diagram**
  - If a response is generated for the current request and no errors are detected, the CONFIRM is immediately set to TRUE and remains TRUE (provided that the function block is called with ENABLE = TRUE). If the function block is called with ENABLE is reset to FALSE, the CONFIRM will be reset to FALSE, and the function block can restart execution.



  - If the function block is called with ENABLE reset to FALSE before the current execution completes, the execution will be canceled. Responses to cancellation requests may have already been sent or will arrive later, but these responses will be ignored at this time.



- If the current execution is aborted due to an SDO abort message or completes due to an error detected, the ERROR will be set to a non-zero value (for more information on the detected fault codes, see 18.4.11.9 Fault Codes). If the function block is called with ENABLE reset to FALSE, the ERROR will be reset to 0, and the function block can restart execution.



- **Program example**

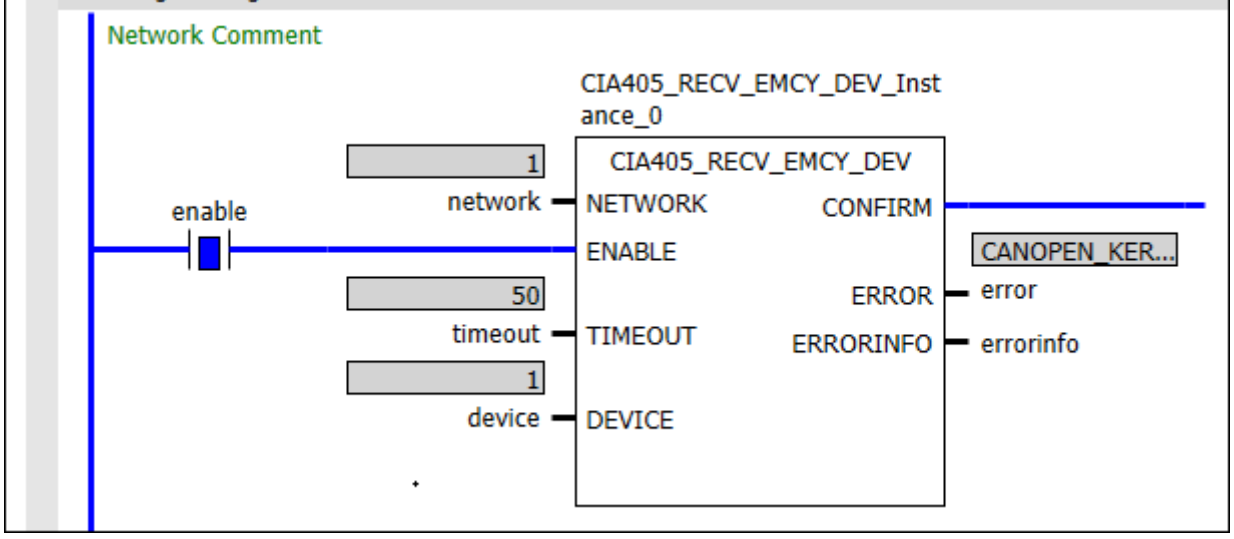
  ST:

	Name	Data type	Address	Value
	CIA405_RECV_EMCY_D...	CIA405_RECV_EMCY_DEV		
	network	USINT		1
	enable	BOOL		TRUE
	timeout	UDINT		50
	device	CIA405_DEVICE		1
	confirm	BOOL		TRUE
	error	CIA405_CANOPEN_KERNEL_...		CANOPEN_KERN...
	errorinfo	CIA405_EMCY_ERROR		

```
CIA405_RECV_EMCY_DEV_Instance_0(
NETWORK  1     := network  1     ,
ENABLE   TRUE  := enable   TRUE  ,
TIMEOUT  50    := timeout  50    ,
DEVICE   1     := device   1     ,
CONFIRM  TRUE  => confirm  TRUE  ,
ERRORCANOPEN_KERNEL_NO_ER... => errorCANOPEN_KERNEL_NO_ER...,
ERRORINFO => errorinfo);RETURN
```

LD:

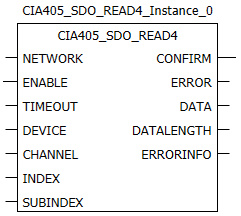
	Name	Data type	Address	Value
	CIA405_RECV_EMCY_D...	CIA405_RECV_EMCY_DEV		
	network	USINT		1
	enable	BOOL		TRUE
	timeout	UDINT		50
	device	CIA405_DEVICE		1
	error	CIA405_CANOPEN_KERNEL_...		CANOPEN_KERN...
	errorinfo	CIA405_EMCY_ERROR		



18.4.11.6 CIA405_SDO_READ4

This instruction reads up to 4 bytes of a CANopen object.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
CIA405_SDO_READ4	Read up to 4 bytes of a CANopen object	FB		CIA405_SDO_READ4_Instance_0(NETWORK := , ENABLE := , TIMEOUT := , DEVICE := , CHANNEL := , INDEX := , SUBINDEX := , CONFIRM => , ERROR => , DATA => , DATALENGTH => , ERRORINFO =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	NETWORK	CAN network number	USINT	Follows data type	-	The CAN port executed by the function block corresponds to the CAN port number of the host.
	ENABLE	Enable	BOOL	Follows data type	FALSE	Level-triggered, controls the ON/OFF execution of the function block.
	TIMEOUT	Maximum execution time	UDINT	Follows data type	-	Maximum execution time in ms. If the timeout is reached before a response is received, execution will be aborted with a timeout error. 0 (factory setting) = Disable timeout. 1...65535 = Timeout value.
	DEVICE	CANopen device node ID	CIA405_DEVICE	Follows data type	-	CANopen device node ID. 0 (default value) = local device (controller). 1...127 = device node ID.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	CHANNEL	SDO channel number	USINT	Follows data type	-	SDO channel number. Default value = 1.
	INDEX	Object Index	WORD	Follows data type	-	Object index. Range: 0000 (hexadecimal)...FFFF (hexadecimal)
	SUBINDEX	Object subindex	BYTE	Follows data type	-	Object subindex. Range: 00 (hexadecimal)… FF (hexadecimal).
Output	CONFIRM	FB completion flag	BOOL	Follows data type	FALSE	Function block completion flag.
	ERROR	Function block error message	CIA405_CANOPEN_KERNEL_ERROR	Follows data type	-	Function block error message. For details, see 18.4.11.9 Fault Codes.
	DATA	Data reception	ARRAY [1..4] OF BYTE	Follows data type	-	Receives the data array read from the slave device object.
	DATALENGTH	Size of the read object	USINT	Follows data type	-	Size of the read object in bytes.
	ERRORINFO	SDO abort message	CIA405_SDO_ERROR	Follows data type	-	When ERROR = 1, return SDO abort message content (size is 4 bytes).

- **Function**

  This instruction reads up to 4 bytes of a CANopen object from the specified device via an SDO message.

  The table below shows the object size and the corresponding contents of the DATA array.

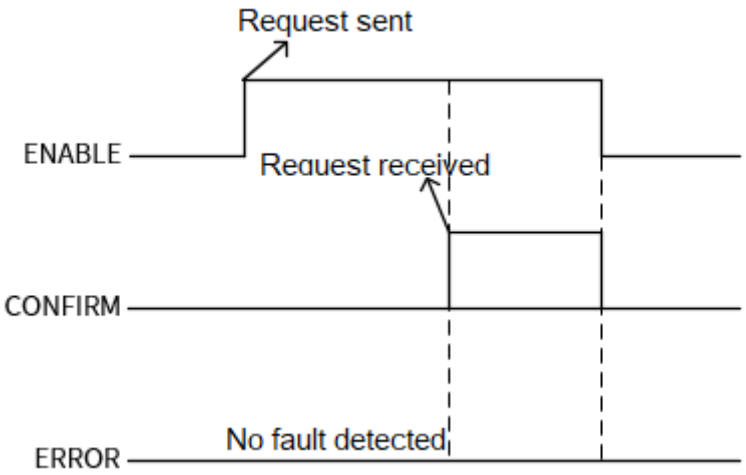
Example of Object Size	DATALENGTH	DATA(1)	DATA(2)	DATA(3)	DATA(4)
1 byte 01 (hexadecimal)	1	01 (hexadecimal)	Invalid	Invalid	Invalid
2 bytes 01 23 (hexadecimal)	2	LSB 23 (hexadecimal)	MSB 01 (hexadecimal)	Invalid	Invalid
3 bytes 01 23 45 (hexadecimal)	3	LSB 45 (hexadecimal)	23 (hexadecimal)	MSB 01 (hexadecimal)	Invalid

Example of Object Size	DATALENGTH	DATA(1)	DATA(2)	DATA(3)	DATA(4)
4 bytes 01 23 45 67 (hexadecimal)	4	LSB 67 (hexadecimal)	45 (hexadecimal)	23 (hexadecimal)	MSB 01 (hexadecimal)

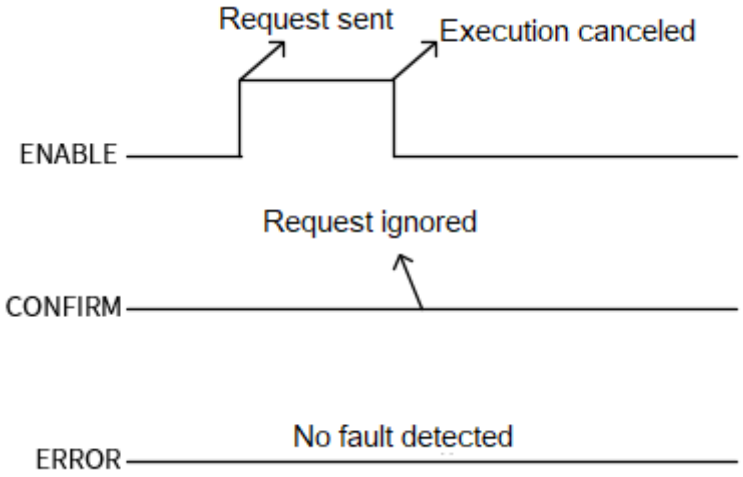
NOTE

LSB = Least Significant Byte; MSB = Most Significant Byte.

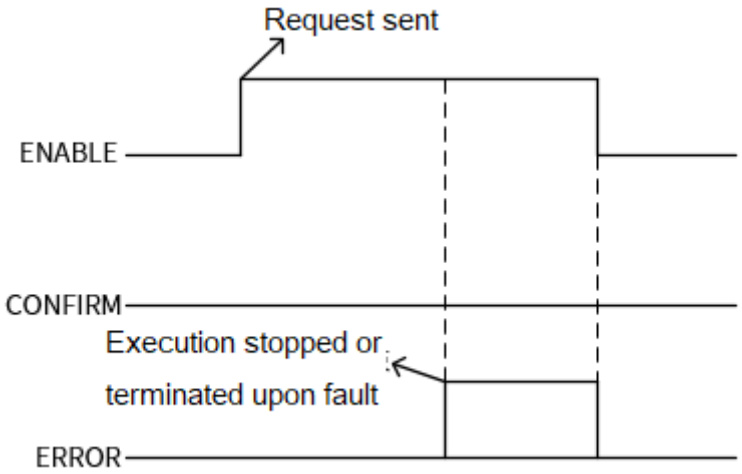
- **Sequence diagram**
  - If a response is generated for the current request and no errors are detected, the CONFIRM is immediately set to TRUE and remains TRUE (provided that the function block is called with ENABLE = TRUE). If the function block is called with ENABLE is reset to FALSE, the CONFIRM will be reset to FALSE, and the function block can restart execution.



  - If the function block is called with ENABLE reset to FALSE before the current execution completes, the execution will be canceled. Responses to cancellation requests may have already been sent or will arrive later, but these responses will be ignored at this time.



  - If the current execution is aborted due to an SDO abort message or completes due to an error detected, the ERROR will be set to a non-zero value (for more information on the detected fault codes, see 18.4.11.9 Fault Codes). If the function block is called with ENABLE reset to FALSE, the ERROR will be reset to 0, and the function block can restart execution.



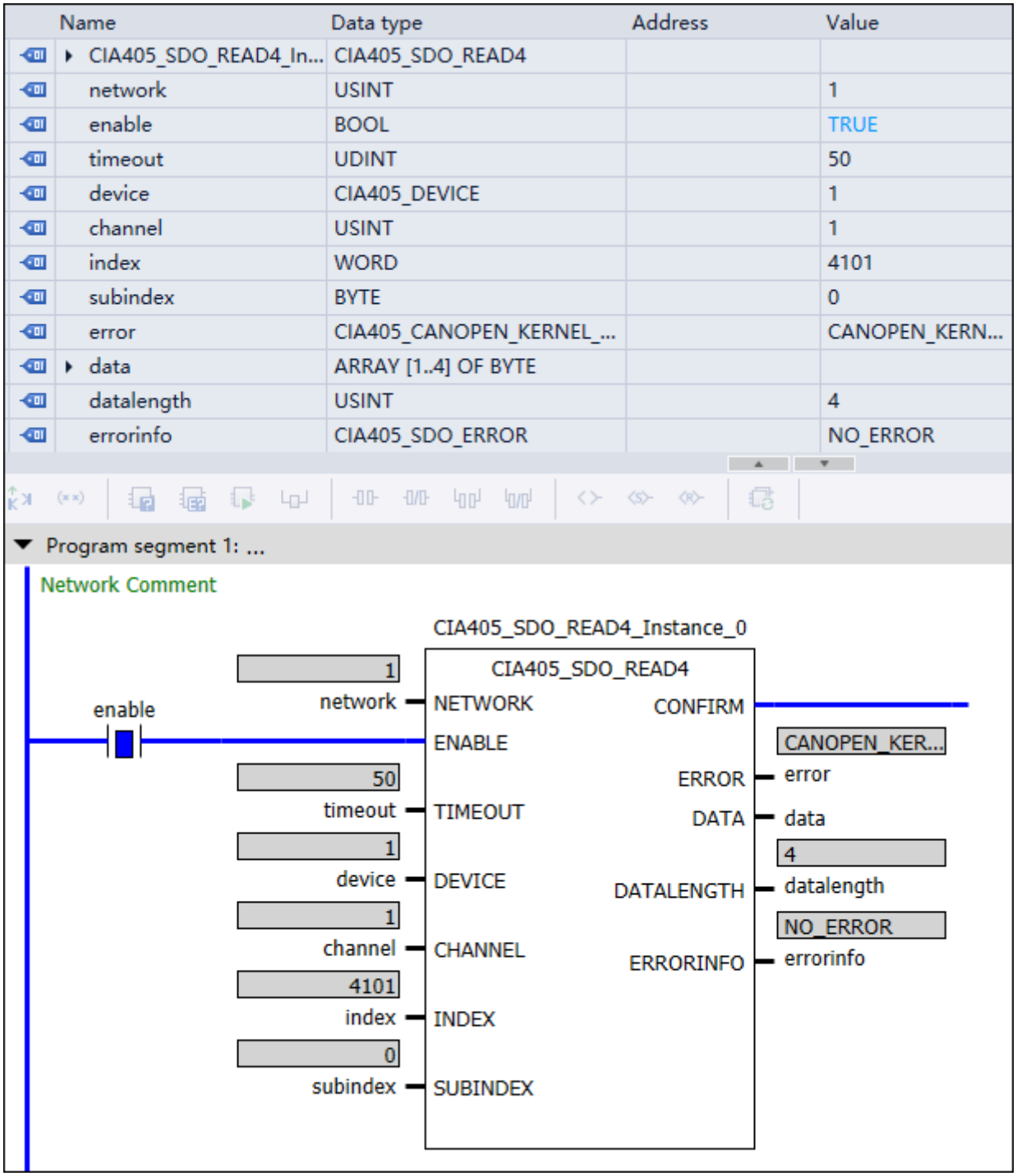
- **Program example**

  ST:

	Name	Data type	Address	Value
	CIA405_SDO_READ4_In...	CIA405_SDO_READ4		
	network	USINT		1
	enable	BOOL		TRUE
	timeout	UDINT		50
	device	CIA405_DEVICE		1
	channel	USINT		1
	index	WORD		16#1005
	subindex	BYTE		0
	confirm	BOOL		TRUE
	error	CIA405_CANOPEN_KERNEL_...		CANOPEN_KERN...
	data	ARRAY [1..4] OF BYTE		
	datalength	USINT		4
	errorinfo	CIA405_SDO_ERROR		NO_ERROR

```
CIA405_SDO_READ4_Instance_0(
NETWORK  1     := network  1     ,
ENABLE  TRUE   := enable  TRUE  ,
TIMEOUT  50    := timeout  50    ,
DEVICE  1      := device  1      ,
CHANNEL  1     := channel  1     ,
INDEX  4101    := index  16#1005 ,
SUBINDEX  0    := subindex  0    ,
CONFIRM  TRUE  => confirm  TRUE  ,
ERRORCANOPEN_KERNEL_NO_ER... => errorCANOPEN_KERNEL_NO_ER...,
DATA => data,
DATALENGTH  4  => datalength  4  ,
ERRORINFO NO_ERROR  => errorinfo NO_ERROR );
RETURN
```

  LD:



- **Precautions**

  The following specific parameters must be passed to the function block:

  1) Device node ID.

  2) SDO client/server channel (only one channel is defined by default).

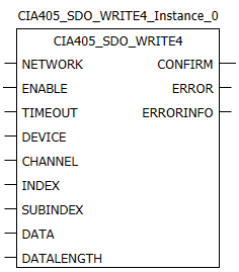
  3) CANopen object index/subindex.

  If the read operation completes successfully, the function block returns the size of the read object, and the data is provided in a 4-byte array.

18.4.11.7 CIA405_SDO_WRITE4

This instruction writes up to 4 bytes of a CANopen object.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
CIA405_SDO_WRITE4	Write up to 4 bytes of a CANopen object	FB		CIA405_SDO_WRITE4_Instance_0(NETWORK := , ENABLE := , TIMEOUT := , DEVICE := , CHANNEL := , INDEX := , SUBINDEX := , DATA := , DATALENGTH := , CONFIRM => , ERROR => , ERRORINFO =>);

- **Related variables**

Variable

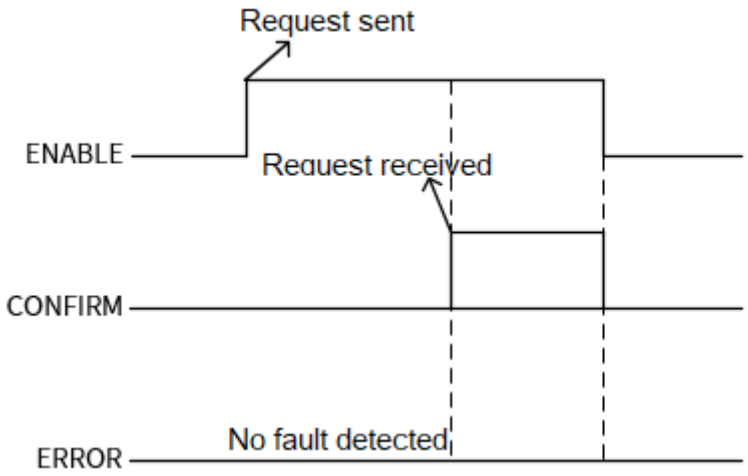
Input / Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	NETWORK	CAN network number	USINT	Follows data type	-	The CAN port executed by the function block corresponds to the CAN port number of the host.
	ENABLE	Enable	BOOL	Follows data type	FALSE	Level-triggered, controls the ON/OFF execution of the function block.
	TIMEOUT	Maximum execution time	UDINT	Follows data type	-	Maximum execution time in ms. If the timeout is reached before a response is received, execution will be aborted with a timeout error. 0 (factory setting) = Disable timeout. 1...65535 = Timeout value.
	DEVICE	CANopen device node ID	CIA405_DEVICE	Follows data type	-	CANopen device node ID. 0 (default value) = local device (controller). 1...127 = device node ID.
	CHANNEL	SDO channel number	USINT	Follows data type	-	SDO channel number. Default value = 1.

Input / Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	INDEX	Object Index	WORD	Follows data type	-	Object index. Range: 0000 (hexadecimal)...FFFF (hexadecimal)
	SUBINDEX	Object subindex	BYTE	Follows data type	-	Object subindex. Range: 00 (hexadecimal)… FF (hexadecimal).
	DATA	Received data	ARRAY [1..4] OF BYTE	Follows data type	-	Array used to store the object value data to be written.
	DATALENGTH	Size of the written object	USINT	Follows data type	-	Size of the written object to in bytes.
Output	CONFIRM	FB completion flag	BOOL	Follows data type	FALSE	Function block completion flag.
	ERROR	Function block error message	CIA405_CANOPEN_KERNEL_ERROR	Follows data type	-	Function block error message. For details, see 18.4.11.9 Fault Codes.
	ERRORINFO	SDO abort message	CIA405_SDO_ERROR	Follows data type	-	When ERROR = 1, return SDO abort message content (size is 4 bytes).

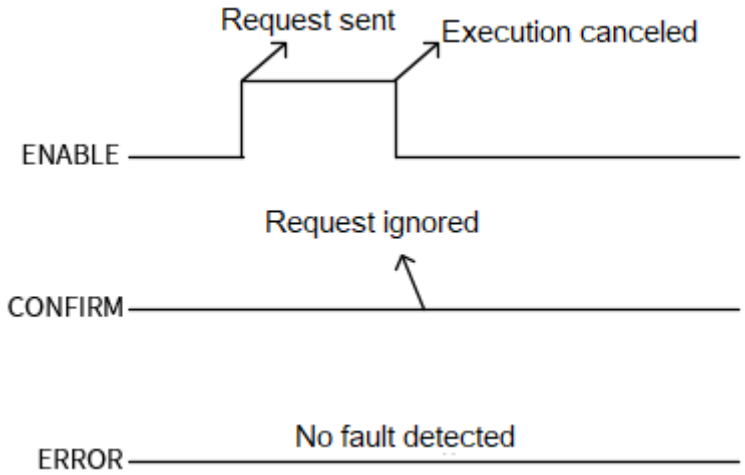
- **Function**

  This instruction writes up to 4 bytes of a CANopen object from the specified device via an SDO message.

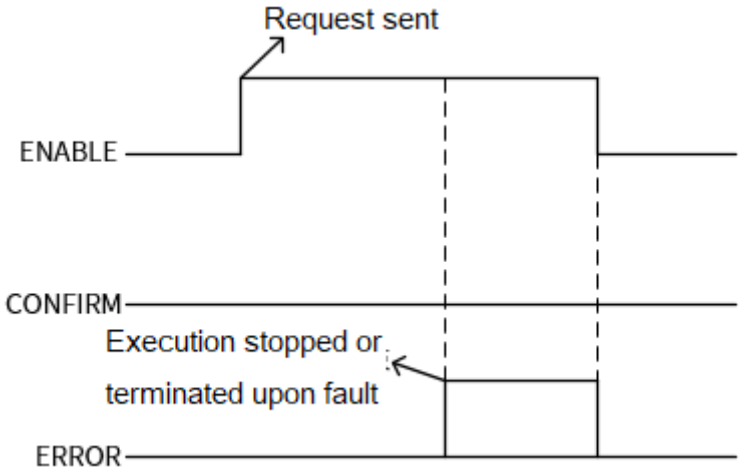
- **Sequence diagram**
  - If a response is generated for the current request and no errors are detected, the CONFIRM is immediately set to TRUE and remains TRUE (provided that the function block is called with ENABLE = TRUE). If the function block is called with ENABLE is reset to FALSE, the CONFIRM will be reset to FALSE, and the function block can restart execution.



- If the function block is called with ENABLE reset to FALSE before the current execution completes, the execution will be canceled. Responses to cancellation requests may have already been sent or will arrive later, but these responses will be ignored at this time.



- If the current execution is aborted due to an SDO abort message or completes due to an error detected, the ERROR will be set to a non-zero value (for more information on the detected fault codes, see 18.4.11.9 Fault Codes). If the function block is called with ENABLE reset to FALSE, the ERROR will be reset to 0, and the function block can restart execution.



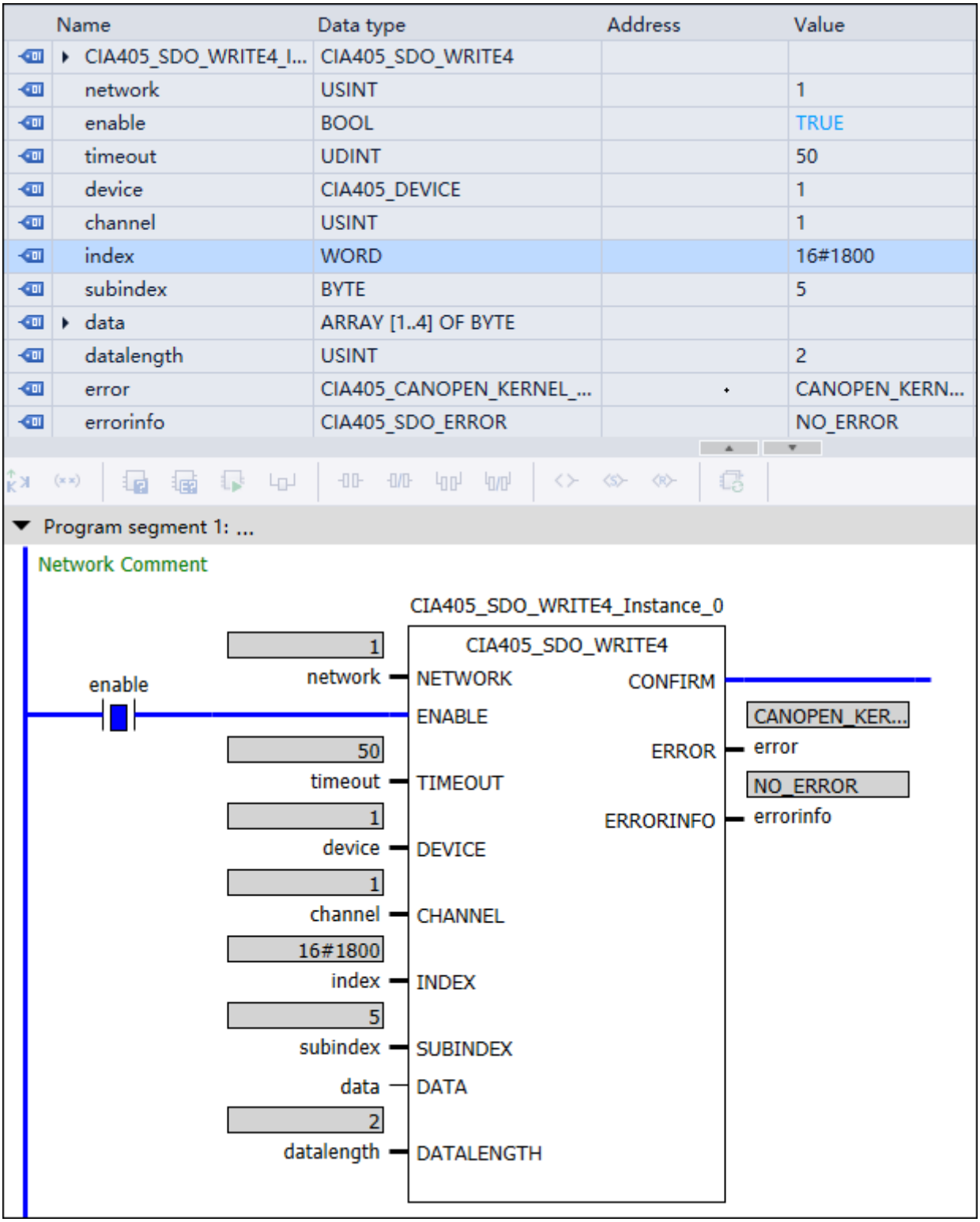
- **Program example**

  ST:

Name	Data type	Address	Value
CIA405_SDO_WRITE4_I...	CIA405_SDO_WRITE4		
network	USINT		1
enbale	BOOL		TRUE
timeout	UDINT		50
device	CIA405_DEVICE		1
channel	USINT		1
index	WORD		6144
subindex	BYTE		5
data	ARRAY [1..4] OF BYTE		
datalength	USINT		2
confirm	BOOL		TRUE
error	CIA405_CANOPEN_KERNEL_...		CANOPEN_KERN...
errorinfo	CIA405_SDO_ERROR		NO_ERROR

```
CIA405_SDO_WRITE4_Instance_0(
NETWORK  1    := network  1    ,
ENABLE  TRUE  := enbale  TRUE  ,
TIMEOUT  50   := timeout  50   ,
DEVICE  1     := device  1     ,
CHANNEL  1    := channel  1    ,
INDEX  6144   := index  6144   ,
SUBINDEX  5   := subindex  5   ,
DATA := data,
DATALENGTH  2 := datalength  2 ,
CONFIRM  TRUE => confirm  TRUE ,
ERRORCANOPEN_KERNEL_NO_ER... => errorCANOPEN_KERNEL_NO_ER...,
ERRORINFO NO_ERROR => errorinfo NO_ERROR );RETURN
```

LD:



- **Precautions**

  The following specific parameters must be passed to the function block:

  1) Device node ID.

  2) SDO client/server channel (only one channel is defined by default).

  3) CANopen object index/subindex.

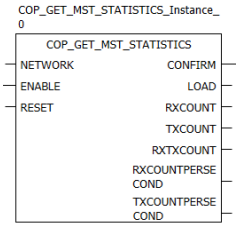
  4) The value to be written.

  5) The number of bytes to be written.

18.4.11.8 COP_GET_MST_STATISTICS

This instruction obtains statistics.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
COP_GET_MST_STATISTICS	Obtain statistics	FB		COP_GET_MST_STATISTICS_Instance_0 (NETWORK := , ENABLE := , RESET := , CONFIRM => , LOAD => , RXCOUNT => , TXCOUNT => , RXTXCOUNT => , RXCOUNTPERSECOND => , TXCOUNTPERSECOND =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	NETWORK	CAN network number	USINT	Follows data type	-	The CAN port executed by the function block corresponds to the CAN port number of the host.
	ENABLE	Enable	BOOL	Follows data type	FALSE	Level-triggered, controls the ON/OFF execution of the function block.
	RESET	Clear the values of current output parameters	BOOL	Follows data type	FALSE	Clear the values of current output parameters.
Output	CONFIRM	FB completion flag	BOOL	Follows data type	FALSE	Function block done flag.
	LOAD	Network load rate	BYTE	Follows data type	-	Network load rate
	RXCOUNT	Count of received frames	DWORD	Follows data type	-	Count of received frames
	TXCOUNT	Count of sent frames	DWORD	Follows data type	-	Count of sent frames

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	RXTXCOUNT	Count of sent and received frames	DWORD	Follows data type	-	Count of sent and received frames
	RXCOUNTPERSECOND	Count of received frame per second	DWORD	Follows data type	-	Count of received frame per second, **currently not supported**.
	TXCOUNTPERSECOND	Transmitted frame count per second	DWORD	Follows data type	-	Transmitted frame count per second, **currently not supported**.

- **Function**

  This instruction obtains statistics.

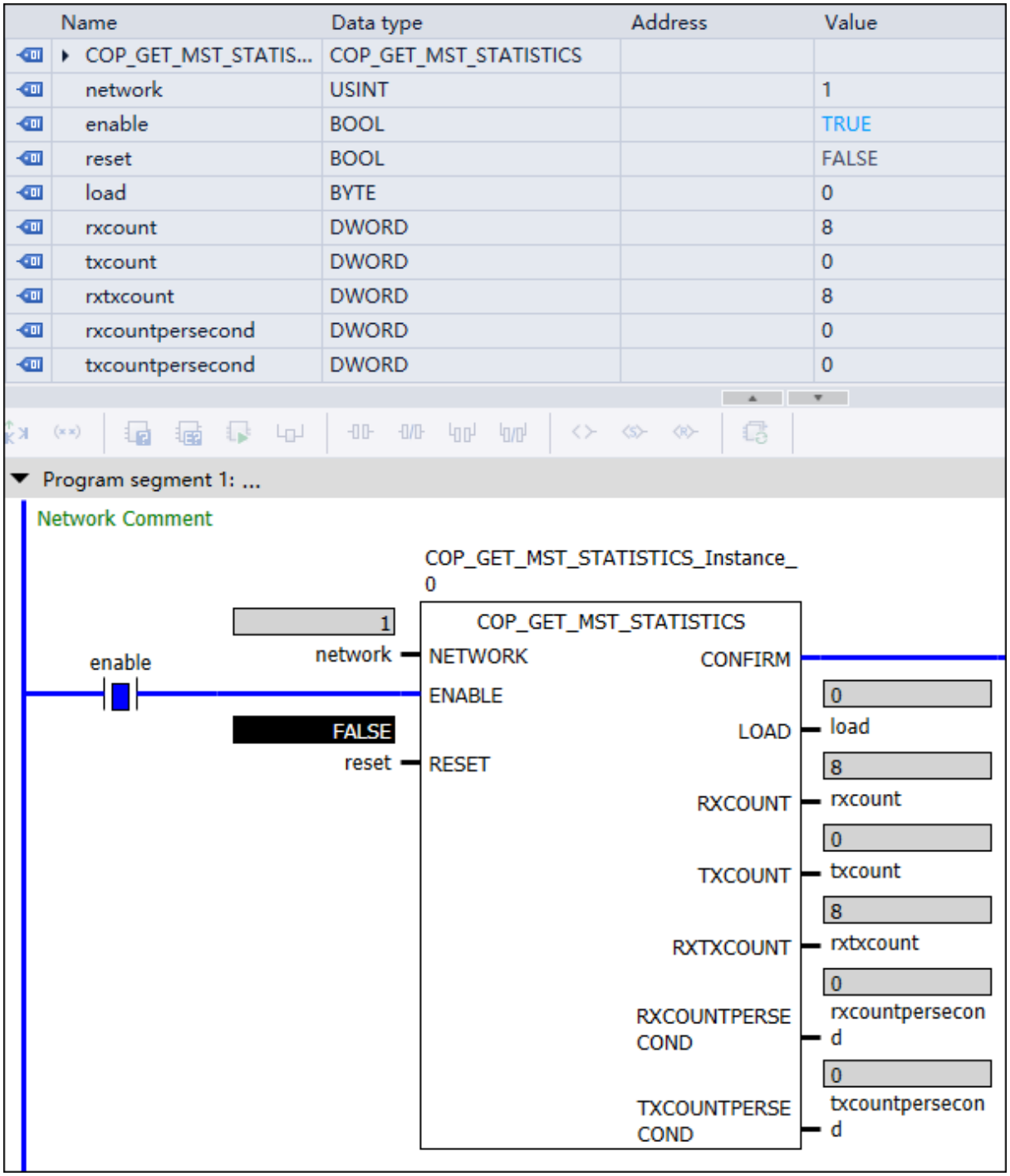
- **Program example**

  ST:

Name	Data type	Address	Value
COP_GET_MST_STATIS...	COP_GET_MST_STATISTICS		
network	USINT		1
enable	BOOL		TRUE
reset	BOOL		FALSE
confirm	BOOL		TRUE
load	BYTE		0
rxcount	DWORD		8
txcount	DWORD		0
rxtxcount	DWORD		8
rxcountpersecond	DWORD		0
txcountpersecond	DWORD		0

```
COP_GET_MST_STATISTICS_Instance_0(
NETWORK    1      := network    1      ,
ENABLE     TRUE   := enable     TRUE   ,
RESET      FALSE  := reset      FALSE  ,
CONFIRM    TRUE   => confirm    TRUE   ,
LOAD       0      => load       0      ,
RXCOUNT    8      => rxcount    8      ,
TXCOUNT    0      => txcount    0      ,
RXTXCOUNT  8      => rxtxcount  8      ,
RXCOUNTPERSECOND   0   => rxcountpersecond   0   ,
TXCOUNTPERSECOND   0   => txcountpersecond   0   );RETURN
```

  LD:



18.4.11.9 Fault Codes

Fault code	Value	Description
CANOPEN_KERNEL_NO_ERROR	0	No error detected by the CANopen kernel.

Fault code	Value	Description
CANOPEN_KERNEL_OTHER_ERROR	1	If the ERROR = 01 (hexadecimal), an error has been detected. If other error codes are present on the function block, the ERROR contains more specific information. Example: CIA405 .SDO_READ and CIA405.SDO_WRITE function blocks: CIA405.SDO_READ and CIA405.SDO_WRITE function blocks: The ERRORINFO output contains the content of the SDO abort message. CIA405 .RECV_EMCY and CIA405.RECV_EMCY_DEV function blocks: The ERRORINFO output contains the content of the received EMCY message. If an emergency message is provided in the ERRORINFO output, then the ERROR = 1 and the CONFIRM = 0. CIA405 .GET_CANOPEN_KERNEL_STATE function block: The value 01 (hexadecimal) will never be provided, as no other error codes are output.
CANOPEN_KERNEL_DATA_OVERFLOW	2	Overflow in the transmit or receive buffer of the CANopen object.
CANOPEN_KERNEL_TIMEOUT	3	Function block execution timeout.
CANOPEN_KERNEL_CANBUS_OFF	16	CANopen node disconnected from the CAN bus.
CANOPEN_KERNEL_CAN_ERROR_PASSIVE	17	The CANopen node is in error passive state: the node can communicate but is not allowed to transmiterror flags upon error detection.
CANOPEN_INTERNAL_FB_ERROR	33	Manufacturer-specific error code. Note: Values from 21 to FF (hexadecimal) are reserved for device manufacturer use.

18.4.12 Free TCP

18.4.12.1 Instruction List

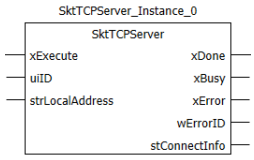
Instruction	FB/FC	Function	Library	Applicable Library Version
18.4.12.2 SktTCPServer	FB	Create TCP server communication service	TcpUdpFreeProtocol	Library version 1.0.0.0 and later
18.4.12.3 SktTCPClient	FB	Create TCP client communication service		Library version 1.0.0.0 and later
18.4.12.4 SktTCPSend	FB	Send TCP data		Library version 1.0.0.0 and later
18.4.12.5 SktTCPRecv	FB	Receive TCP data		Library version 1.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.4.12.6 SktTCPGetStatus	FB	This instruction gets TCP connection status.		Library version 1.0.0.0 and later
18.4.12.7 SktTCPClearBuffer	FB	Clear TCP protocol stack buffer		Library version 1.0.0.0 and later
18.4.12.8 SktTCPRecv_C	FB	Receive TCP data (integrating the functions of creation, closure, and status acquisition)		Library version 1.0.0.0 and later
18.4.12.9 SktTCPSend_C	FB	Send TCP data (integrating the functions of creation, closure, and status acquisition)		Library version 1.0.0.0 and later
18.4.12.10 SktClose	FB	Close TCP communication service		Library version 1.0.0.0 and later

18.4.12.2 SktTCPServer

This instruction creates TCP server communication.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
SktTCPServer	Create TCP server communication service	FB		SktTCPServer_Instance_0(x Execute := , uiID := , strLocalAddress := , xDone => , xBusy => , xError => , wErrorID => , stConnectInfo =>);

- **Related variables**

Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Trigger execution	BOOL	TRUE, FALSE	FALSE	Execution starts on the rising edge and must remain at a high level during the entire process.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	uiID	ID number	UINT	1 to 256	0	Each connection corresponds to an ID number.
	stLocalAddress	Local IP address	ADDRESS	-	-	Local IP address and port number, used for listening to connection requests. Port number range: 2000 to 65535.
Output	xDone	Done	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is done and the connection has been successfully established.
	xBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is executing, waiting for connection requests.
	xError	Execution error	BOOL	TRUE, FALSE	FALSE	Error flag that is set to TRUE when an instruction execution error occurs.
	wErrorID	Fault ID	WORD	-	0	Outputs an error ID greater than 0 when the instruction execution error occurs. For details, see 18.4.12.11 Fault Codes.
	stConnectInfo	Connection information	TCP_CONNECT_INFO	-	-	Outputs the local and remote IP addresses and ports upon successful connection establishment.

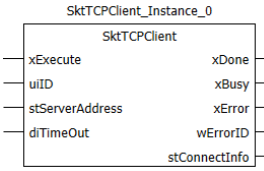
- **Function**
  - Before enabling the instruction (xExecute=TRUE), input the stLocalAddress to set the server's IP address and port number, and the uiID to specify the ID range from 1 to 256, for subsequent calls to SktTCPSend, SktTCPRecv, SktTCPGetStatus, SktTCPClearBuffer, and SktClos instructions.
  - During the instruction execution, xExecute=TRUE is maintained, waiting for connection requests from client.
  - After a successful connection is established, the instruction execution ends (xDone=TRUE) and stConnectInfo outputs the IP addresses and port numbers of both connected parties.
  - If an error occurs during instruction execution, xError is set to TRUE, and wErrorID outputs an error code to explain the cause of the error.

- **Precautions**
  - After a successful connection, uiID will no longer be bound to the instruction. To close the connection and release the ID, the SktClose instruction must be called.
  - In on-site applications, multiple tasks should not call the same instance of SktTCPServer.

18.4.12.3 SktTCPClient

This instruction creates TCP client communication.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
SktTCPClient	Create TCP client communication service	FB		SktTCPClient_Instance_0(xExecute := , uiID := , stServerAddress := , diTimeOut := , xDone => , xBusy => , xError => , wErrorID => , stConnectInfo =>);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Trigger execution	BOOL	TRUE, FALSE	FALSE	Execution starts on the rising edge and must remain at a high level during the entire process.
	uiID	ID number	UINT	1 to 256	0	Each connection corresponds to an ID number.
	stServerAddress	Remote server address	ADDRESS	-	-	The IP address and port number of the remote server are used to specify the target server to be accessed. Port number range: 2000 to 65535.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	diTimeOut	Connection timeout period	DINT	50 to 3000	50	Time to wait after initiating a connection request in ms.
Output	xDone	Done	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is done and the connection has been successfully established.
	xBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is executing, waiting for connection requests.
	xError	Execution error	BOOL	TRUE, FALSE	FALSE	Error flag that is set to TRUE when an instruction execution error occurs.
	wErrorID	Fault ID	WORD	-	0	Outputs an error ID greater than 0 when the instruction execution error occurs. For details, see 18.4.12.11 Fault Codes.
	stConnectInfo	Connection information	TCP_CONNECT_INFO	-	-	Outputs the local and remote IP addresses and ports upon successful connection establishment.

- **Function**
  - Before enabling the instruction (xExecute=TRUE), input the stServerAddress to set the IP address and port number of the target server to be accessed; input uiID to specify the ID range 1 to 256 for subsequent calls to SktTCPSend, SktTCPRecv, SktTCPGetStatus, SktTCPClearBuffer, and SktClose instructions; input the timeout period, which is generally the default value, but can be adjusted appropriately based on network environment.
  - During the instruction execution (xExecute=TRUE maintained), a connection request is initiated to the target server.

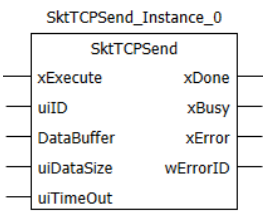
- After a successful connection is established, the instruction execution ends (xDone=TRUE) and stConnectInfo outputs the IP addresses and port numbers of both connected parties.
- If an error occurs during instruction execution, xError is set to TRUE, and wErrorID outputs an error code to explain the cause of the error.

- **Precautions**
  - After a successful connection, uiID will no longer be bound to the instruction. To close the connection and release the ID, the SktClose instruction must be called.
  - In on-site applications, multiple tasks should not call the same instance of SktTCPClient.

18.4.12.4 SktTCPSend

This instruction sends TCP data.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
SktTCPSend	Send TCP data	FB		SktTCPSend_Instance_0(xExecute := , uiID := , DataBuffer := , uiDataSize := , uiTimeOut := , xDone => , xBusy => , xError => , wErrorID =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Trigger execution	BOOL	TRUE, FALSE	FALSE	Data sending is triggered on a rising edge.
	uiID	ID number	UINT	1 to 256	0	Each connection corresponds to an ID number.
	DataBuffer	Sending data buffer	ANY	-	-	It supports any data type, with array types limited to a maximum of three dimensions.
	uiDataSize	Size of sent data	UINT	1 to 8192	0	Size of sent data in byte.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	uiTimeOut	Sending timeout	UINT	1 to 1000	50	Sending timeout in ms.
Output	xDone	Done	BOOL	TRUE, FALSE	FALSE	Set to TRUE when data sending is done.
	xBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is executing, waiting for connection requests.
	xError	Execution error	BOOL	TRUE, FALSE	FALSE	Error flag that is set to TRUE when an instruction execution error occurs.
	wErrorID	Fault ID	WORD	-	0	Outputs an error ID greater than 0 when the instruction execution error occurs. For details, see 18.4.1 2.11 Fault Codes.

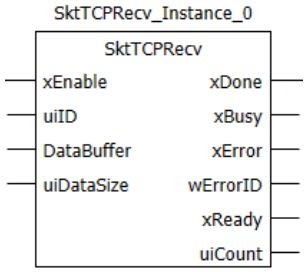
- **Function**
  - Before instruction execution (xEnable=TRUE), input uiID to specify the ID number (created by SktTCPClient and SktTCPServer instruction). Input uiDataSize (in Byte) to specify the size of sent data, which must not exceed the size of DataBuffer or a maximum of 8192 bytes. Input the sending timeout uiTimeOut in ms (default: 50). Generally, the default value can be used, but it can be appropriately increased when the network is congested.
  - During instruction execution, xExecute=TRUE is maintained.
  - After instruction execution, set xDone to TRUE.
  - If an error occurs during instruction execution, xError is set to TRUE, and wErrorID outputs an error code to explain the cause of the error.
- **Precautions**

  In on-site applications, multiple tasks should not call the same instance of SktTCPSend.

18.4.12.5 SktTCPRecv

This instruction receives TCP data.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
SktTCPRecv	Receive TCP data	FB		SktTCPRecv_Instance_0(x Enable := , uiID := , DataBuffer := , uiDataSize := , xDone => , xBusy => , xError => , wErrorID => , xReady => , uiCount =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Enable	BOOL	TRUE, FALSE	FALSE	Input TRUE to execute reception and FALSE to exit reception.
	uiID	ID number	UINT	1 to 256	0	Each connection corresponds to an ID number.
	DataBuffer	Data reception buffer	ANY	-	-	It supports any data type, with array types limited to a maximum of three dimensions.
	uiDataSize	Size of received data	UINT	0 to 8192	0	Size of received data in byte. • Input > 0: Fixed-length reception mode. The maximum size must not exceed 8192 Bytes, nor exceed the actual size of received buffer. • Input = 0: Standard mode.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	xDone	Reserved	BOOL	TRUE, FALSE	FALSE	Always FALSE.
	xBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is executing, waiting for connection requests.
	xError	Execution error	BOOL	TRUE, FALSE	FALSE	Error flag that is set to TRUE when an instruction execution error occurs.
	wErrorID	Fault ID	WORD	-	0	Outputs an error ID greater than 0 when the instruction execution error occurs. For details, see 18.4.12.11 Fault Codes.
	xReady	Reception completion flag	BOOL	TRUE, FALSE	FALSE	Set to TRUE when data reception is done and maintain this state for one task cycle.
	uiCount	Actual size of received data	UINT	0 to 8192	0	Outputs the actual size of received data in Byte when xReady = TRUE and maintains this state for one task cycle.

- **Function**
  - SktTCPRecv is the data reception instruction based on the TCP protocol. It operates in two modes, determined by the value of uiDataSize: standard mode (uiDataSize = 0) and fixed-length reception mode (uiDataSize > 0).

Mode	uiDataSize	Description
Standard mode	0	In standard mode, after data is received in the current execution cycle, the reception completion flag xReady is set to TRUE, and the actual size of received data uiCount is output. The maximum reception size is determined by the input buffer DataBuffer.
Fixed-length reception mode	1 to 8192	In fixed-length reception mode, the reception completion flag xReady is set to TRUE, and the actual size of received data uiCount is output only when the data size matches the specified reception size (uiDataSize = uiCount).

  - Before the instruction execution (xEnable = TRUE), input uiID to specify the ID number (created by the SktTCPClient and SktTCPServer instructions); input DataBuffer to specify the application layer of data buffer, which supports one-dimensional arrays of BYTE type; input uiDataSize (in byte) to specifies the size of the data to be received. The value must not exceed the size of DataBuffer or 8192 bytes.

- xBusy is set to TRUE, and data reception continues during instruction execution (xEnable = TRUE maintained). When data is received, the reception completion flag xReady is set to TRUE, and the actual size of received data uiCount is output. This maintains for only one cycle.
- If an error occurs during instruction execution, xError is set to TRUE, and wErrorID outputs an error code to explain the cause of the error.

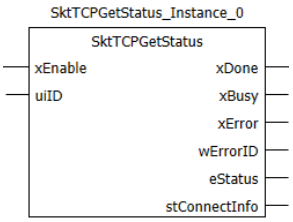
- **Precautions**

  In on-site applications, multiple tasks should not call the same instance of SktTCPRecv.

18.4.12.6 SktTCPGetStatus

Get TCP connection status.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
SktTCPGetStatus	This instruction gets TCP connection status.	FB		SktTCPGetStatus_Instance_0(xEnable := , uiID := , xDone => , xBusy => , xError => , wErrorID => , eStatus => , stConnectInfo =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Enable	BOOL	TRUE, FALSE	FALSE	Input TRUE to execute TCP connection acquisition.
	uiID	ID number	UINT	1 to 256	0	Each connection corresponds to an ID number.
Output	xDone	Reserved	BOOL	TRUE, FALSE	FALSE	Always FALSE.
	xBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is executing, and FALSE when the execution ends.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	xError	Execution error	BOOL	TRUE, FALSE	FALSE	Error flag that is set to TRUE when an instruction execution error occurs.
	wErrorID	Fault ID	WORD	-	0	Outputs an error ID greater than 0 when the instruction execution error occurs. For details, see 18.4.12.11 Fault Codes.
	eStatus	Connection status	TCP_STATUS	-	-	TCP connection status.
	stConnectInfo	Connection information	TCP_CONNECT_INFO	-	-	Connection information includes the addresses of both communication parties.

- **Function**
  - The data type of "eStatus" is the TCP_STATUS enumeration, with enumeration values defined as follows.

Enumeration Value	Definition
0: TCP_NOTCONNECT	No connection
1: TCP_ESTABLISHED	Connection established
2: TCP_SYN_SENT	Connection request sent
3: TCP_SYN_RECV	Connection request responded
4: TCP_FIN_WAIT1	Connection release request initiated
5: TCP_FIN_WAIT2	Connection release request received
6: TCP_FIN_WAIT	Waiting for connection release response
7: TCP_CLOSE	Protocol stack connection released
8: TCP_CLOSE_WAIT	Waiting for application layer to call close to release resources
9: TCP_LAST_ACK	Waiting for final release confirmation
10: TCP_LISTEN	Waiting for connection request
11: TCP_CLOSING	Closing

  - Before the instruction is executed (xEnable = TRUE), Input uiID to specify the ID number (created by the SktTsERVER instruction).
  - xBusy is set to TRUE during instruction execution (xEnable=TRUE maintained).
  - After the instruction execution, set xEnable to FALSE and xBusy to FALSE.

◦ If an error occurs during instruction execution, xError is set to TRUE, and wErrorID outputs an error code to explain the cause of the error.

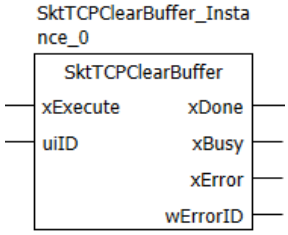
- **Precautions**

  In on-site applications, multiple tasks should not call the same instance of SktTCPGetStatus.

18.4.12.7 SktTCPClearBuffer

This instruction clears TCP receive buffer.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
SktTCPClearBuffer	Clear TCP protocol stack buffer	FB		SktTCPClearBuffer_Instance_0(xExecute := , uiID := , xDone => , xBusy => , xError => , wErrorID =>);

- **Related variables**

  Variable

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Trigger execution	BOOL	TRUE, FALSE	FALSE	Buffer clearance is triggered on a rising edge.
	uiID	ID number	UINT	1 to 256	0	Each connection corresponds to an ID number.
Output	xDone	Done	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is done.
	xBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is executing.
	xError	Execution error	BOOL	TRUE, FALSE	FALSE	Error flag that is set to TRUE when an instruction execution error occurs.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	wErrorID	Fault ID	WORD	-	0	Outputs an error ID greater than 0 when the instruction execution error occurs. For details, see 18.4.12.11 Fault Codes.

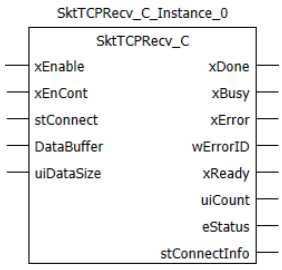
- **Function**
  - Before instruction execution (xExecute=TRUE), input uiID to specify the ID number (created by SktTCPClient and SktTCPServer instructions).
  - During instruction execution, xExecute=TRUE is maintained.
  - After the instruction execution, set xDone to TRUE upon completion, at which point the TCP protocol stack buffer is cleared.
  - If an error occurs during instruction execution, xError is set to TRUE, and wErrorID outputs an error code to explain the cause of the error.
- **Precautions**

  In on-site applications, multiple tasks should not call the same instance of SktTCPClearBuffer.

18.4.12.8 SktTCPRecv_C

This instruction receives TCP data, and integrates the functions of connection creation, closure and status acquisition.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
SktTCPRecv_C	Receive TCP data (integrating the functions of creation, closure, and status acquisition)	FB		SktTCPRecv_C_Instance_0 (xEnable := , xEnCont := , stConnect := , DataBuffer := , uiDataSize := , xDone => , xBusy => , xError => , wErrorID => , xReady => , uiCount => , eStatus => , stConnectInfo =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Enable	BOOL	TRUE, FALSE	FALSE	Set to TRUE to execute reception and FALSE to exit reception.
	xEnCont	Enable connection	BOOL	TRUE, FALSE	FALSE	Input TRUE to establish and maintain the connection, input FALSE to close the connection.
	stConnect	Connection parameters	TCP_CONNECT_PARAM	-	-	Connection parameters include ID number, mode, and address.
	DataBuffer	Data reception buffer	ANY	-	-	It supports any data type, with array types limited to a maximum of three dimensions.
	uiDataSize	Size of received data	UINT	0 to 8192	0	Size of received data in byte. • Input > 0: Fixed-length reception mode. The maximum size must not exceed 8192 Bytes, nor exceed the actual size of received buffer. • Input = 0: Standard mode.
Output	xDone	Reserved	BOOL	TRUE, FALSE	FALSE	Always FALSE.
	xBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is executing, waiting for connection requests.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	xError	Execution error	BOOL	TRUE, FALSE	FALSE	Error flag that is set to TRUE when an instruction execution error occurs.
	wErrorID	Fault ID	WORD	-	0	Outputs an error ID greater than 0 when the instruction execution error occurs. For details, see 18.4.12.11 Fault Codes.
	xReady	Reception completion flag	BOOL	TRUE, FALSE	FALSE	Set to TRUE when data reception is done and maintain this state for one task cycle.
	uiCount	Actual size of received data	UINT	0 to 8192	0	Outputs the actual size of received data in byte when xReady=TRUE and maintains this state for one task cycle.
	eStatus	Connection status	TCP_STATUS	-	-	TCP connection status.
	stConnectInfo	Connection information	TCP_CONNECT_INFO	-	-	Connection information includes addresses of both parties in the connection.

- **Function**
  - The data type of "eStatus" is the TCP_STATUS enumeration, with enumeration values defined as follows.

Enumeration Value	Definition
0: TCP_NOTCONNECT	No connection
1: TCP_ESTABLISHED	Connection established
2: TCP_SYN_SENT	Connection request sent
3: TCP_SYN_RECV	Connection request responded
4: TCP_FIN_WAIT1	Connection release request initiated
5: TCP_FIN_WAIT2	Connection release request received

Enumeration Value	Definition
6: TCP_FIN_WAIT	Waiting for connection release response
7: TCP_CLOSE	Protocol stack connection released
8: TCP_CLOSE_WAIT	Waiting for application layer to call close to release resources
9: TCP_LAST_ACK	Waiting for final release confirmation
10: TCP_LISTEN	Waiting for connection request
11: TCP_CLOSING	Closing

  - Connection creation and connection status monitoring
    - SktTCPRecv_C calls SktTCPServer/SktTCPClient to create a connection.
    - When xEnCont=TRUE, the connection is created using the connection parameters stConnect, and the connection status is detected in real time and output via eStatus. If the connection is disconnected by the peer end while xEnCont=TRUE is maintained, a new connection will be re-established.
    - When xEnCont=FALSE, the connection is closed.
  - Receive data
    - SktTCPRecv_C calls SktTCPRecv to receive data.
    - When xEnable=TRUE, data reception is performed only if the connection has been successfully created (xEnCont=TRUE and eStatus=TCP_ESTABLISHED); otherwise, xEnable=TRUE is invalid.
    - When xEnable=FALSE, data reception stops.

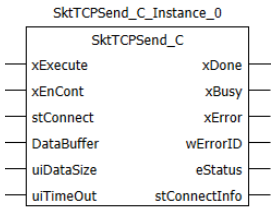
- **Precautions**

  In on-site applications, multiple tasks should not call the same instance of SktTCPRecv_C.

18.4.12.9 SktTCPSend_C

This instruction sends TCP data, and integrates the functions of connection creation, closure and status acquisition.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
SktTCPSend_C	Send TCP data (integrating the functions of creation, closure, and status acquisition)	FB		SktTCPSend_C_Instance_0(xExecute := , xEnCont := , stConnect := , DataBuffer := , uiDataSize := , uiTimeOut := , xDone => , xBusy => , xError => , wErrorID => , eStatus => , stConnectInfo =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Trigger execution	BOOL	TRUE, FALSE	FALSE	Data sending is triggered on a rising edge.
	xEnCont	Enable connection	BOOL	TRUE, FALSE	FALSE	Input TRUE to establish and maintain the connection, input FALSE to close the connection.
	stConnect	Connection parameters	TCP_CONNECT_PARAM	-	-	Connection parameters include ID number, mode, and address.
	DataBuffer	Sending data buffer	ANY	-	-	It supports any data type, with array types limited to a maximum of three dimensions.
	uiDataSize	Size of sent data	UINT	1 to 8192	0	Size of sent data in byte.
	uiTimeOut	Sending timeout	UINT	1 to 1000	50	Sending timeout in ms.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	xDone	Done	BOOL	TRUE, FALSE	FALSE	Set to TRUE when data sending is done.
	xBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is executing, waiting for connection requests.
	xError	Execution error	BOOL	TRUE, FALSE	FALSE	Error flag that is set to TRUE when an instruction execution error occurs.
	wErrorID	Fault ID	WORD	-	0	Outputs an error ID greater than 0 when the instruction execution error occurs. For details, see 18.4.12.11 Fault Codes.
	eStatus	Connection status	TCP_STATUS	-	-	TCP connection status.
	stConnectInfo	Connection information	TCP_CONNECT_INFO	-	-	Connection information includes addresses of both parties in the connection.

- **Function**
  - The data type of "eStatus" is the TCP_STATUS enumeration, with enumeration values defined as follows.

Enumeration Value	Definition
0: TCP_NOTCONNECT	No connection
1: TCP_ESTABLISHED	Connection established
2: TCP_SYN_SENT	Connection request sent
3: TCP_SYN_RECV	Connection request responded
4: TCP_FIN_WAIT1	Connection release request initiated
5: TCP_FIN_WAIT2	Connection release request received
6: TCP_FIN_WAIT	Waiting for connection release response
7: TCP_CLOSE	Protocol stack connection released

Enumeration Value	Definition
8: TCP_CLOSE_WAIT	Waiting for application layer to call close to release resources
9: TCP_LAST_ACK	Waiting for final release confirmation
10: TCP_LISTEN	Waiting for connection request
11: TCP_CLOSING	Closing

- Connection creation and connection status monitoring
  - SktTCPSend_C calls SktTCPServer/SktTCPClient to create a connection.
  - When xEnCont=TRUE, the connection is created using the connection parameters stConnect, and the connection status is detected in real time and output via eStatus. If the connection is disconnected by the peer end while xEnCont=TRUE is maintained, a new connection will be re-established.
  - When xEnCont=FALSE, the connection is closed.
- Send data
  - SktTCPSend_C calls SktTCPSend to send data.
  - When xExecute=TRUE, data will be sent, provided that the connection has been successfully established (xEnCont=TRUE, eStatus=TCP_ESTABLISHED). Otherwise, xExecute = TRUE will not take effect and the execution will not be carried out.

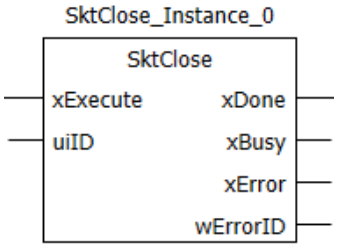
- **Precautions**

  In on-site applications, multiple tasks should not call the same instance of SktTCPSend_C.

18.4.12.10 SktClose

This instruction closes TCP communication.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
SktClose	Close TCP communication service	FB		SktClose_Instance_0(xExecute := , uiID := , xDone => , xBusy => , xError => , wErrorID =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Trigger execution	BOOL	TRUE, FALSE	FALSE	Communication shutdown is triggered on a rising edge.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	uiID	ID number	UINT	1 to 256	0	Each connection corresponds to an ID number.
Output	xDone	Done	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is done.
	xBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is executing.
	xError	Execution error	BOOL	TRUE, FALSE	FALSE	Error flag that is set to TRUE when an instruction execution error occurs.
	wErrorID	Fault ID	WORD	-	0	Outputs an error ID greater than 0 when the instruction execution error occurs. For details, see 18.4.12.11 Fault Codes.

- **Function**
  - Before instruction execution (xExecute=TRUE), input uiID to specify the ID number (created by SktTCPClient and SktTCPServer instructions).
  - During instruction execution, xExecute=TRUE is maintained.
  - After the instruction execution, set xDone to TRUE, at which point the TCP communication closing is terminated and the ID number is released.
  - If an error occurs during instruction execution, xError is set to TRUE, and wErrorID outputs an error code to explain the cause of the error.
- **Precautions**

  In on-site applications, multiple tasks should not call the same instance of SktClose.

18.4.12.11 Fault Codes

Fault code	Value	Description
IEC_ERROR_OK	0	Normal
IEC_ERROR_FAILED	1	Internal error
IEC_ERROR SOCKET	2	Socket creation error
IEC_ERROR_BIND	3	Address binding error
IEC_ERROR_LISTEN	4	Listening error
IEC_ERROR_ACCEPT	5	Accept connection error
IEC_ERROR_CONNECT	6	Send connection error
IEC_ERROR_SEND	7	Send data error
IEC_ERROR_RECV	8	Receive data error
IEC_ERROR_TIMEOUT	9	Timeout error

Fault code	Value	Description
IEC_ERROR_ID_USED	10	ID already in use
IEC_ERROR_ID_DISENABLE	11	ID not enabled
IFC_ERROR_PROTO	12	Protocol error
IEC_ERROR_JOIN_GROUT	13	Failed to join multicast
IEC_ERROR_PARAM_ID	100	ID error
IEC_ERROR_PARAM_IP	101	IP address error
IEC_ERROR_PARAM_PORT	102	Port number error
IEC_ERROR_PARAM_BUFFER	103	Cache error
IEC_ERROR_PARAM_TIMEOUT	104	Timeout error
IEC_ERROR_PARAM_MODE	105	Mode error
ERROR_LOCAL_PORT	1001	Local port mismatch
ERROR_LOCAL_IP	1002	Local IP mismatch
ERROR_REMOTE_PORT	1003	Remote port mismatch
ERROR_REMOTE_IP	1004	Remote IP mismatch
ERROR_PATAM_MODE	1005	TCP communication mode mismatch
ERROR_PATAM_TYPE	1006	Communication type mismatch

18.4.12.12 Example of Free TCP Communication Instruction Usage

Example of TCP client usage

Create a TCP client to access the server address (IP Address: 192.168.1.10, Port: 8080). After a successful connection is established, enable data reception, then wait for and receive the data from the server. Finally, send the received data back to the server.

- **Implementation with basic instructions**

Variable declarations are as follows:

```
VAR
en_start: BOOL;
start_connect: BOOL;
exit_connect: BOOL;
SktTCPClient_Instance_0: SktTCPClient;
stServerAddress: ADDRESS;
uiID: UINT;
diTimeOut: DINT;
SktTCPRecv_Instance_0: SktTCPRecv;
DataBuffer: ARRAY[1..1024] OF byte;
SktTCPSend_Instance_0: SktTCPSend;
R_TRIG_Instance_0: R_TRIG;
F_TRIG_Instance_0: F_TRIG;
SktClose_Instance_0: SktClose;
SktTCPGetStatus_Instance_0: SktTCPGetStatus;
END_VAR
```

The logic implemented in the program editing area is as follows:

```
// Execute connection establishment on the rising edge of en_start
R_TRIG_Instance_0(
CLK := en_start,
Q => );
IF R_TRIG_Instance_0.Q THEN
start_connect := TRUE;
exit_connect := FALSE;
END_IF

// Execute connection closure on the falling edge of en_start
F_TRIG_Instance_0(
CLK := en_start,
Q => );
IF F_TRIG_Instance_0.Q THEN
start_connect := FALSE;
exit_connect := TRUE;
END_IF

// Create TCP client
stServerAddress.strIPAddr := '192.168.1.10'; // Server IP address to access
stServerAddress.uiPort := 8080; // Server port to access
diTimeOut := 1000; // Set a 1s timeout for waiting for the server to respond to the connection request.
uiID := 1; // Use ID number 1
SktTCPClient_Instance_0(
xExecute := start_connect,
uiID := uiID,
stServerAddress := stServerAddress,
diTimeOut := diTimeOut,
xDone => ,
xBusy => ,
xError => ,
wErrorID => ,
stConnectInfo => );

// Close TCP client
SktClose_Instance_0(
```

```
xExecute := exit_connect,
uiID := uiID,
xDone => ,
xBusy => ,
xError => ,
wErrorID => );

// Receive TCP data
SktTCPRecv_Instance_0(
xEnable := SktTCPClient_Instance_0.xDone,// Enable data reception after the TCP client is successfully created.
uiID := uiID,
DataBuffer := DataBuffer,
uiDataSize := 0, // Use standard receive mode
xDone => ,
xBusy => ,
xError => ,
wErrorID => ,
xReady => ,
uiCount => );

// Send TCP data
SktTCPSend_Instance_0(
xExecute := SktTCPRecv_Instance_0.xReady, // After the data is received, enable data sending and forward the received data.
uiID := uiID,
DataBuffer := DataBuffer,
uiDataSize := SktTCPRecv_Instance_0.uiCount,
uiTimeOut := ,
xDone => ,
xBusy => ,
xError => ,
wErrorID => );

// Obtain connection status
SktTCPGetStatus_Instance_0(
xEnable := SktTCPClient_Instance_0.xDone,
```

```
uiID := uiID,
xDone => ,
xBusy => ,
xError => ,
wErrorID => ,
eStatus => ,
stConnectInfo => );
IF SktTCPGetStatus_Instance_0.eStatus = TCP_STATUS.TCP_ESTABLISHED THEN
// The connection is normal, and corresponding actions can be performed.
   ;
END_IF
```

- **Implementation with integrated instructions**

Variable declarations are as follows:

```
VAR
SktTCPRecv_C_Instance_0: SktTCPRecv_C;
SktTCPSend_C_Instance_0: SktTCPSend_C;
stConnect: TCP_CONNECT_PARAM;
DataBuffer: ARRAY[1..1024] OF byte;
END_VAR
```

The logic implemented in the program editing area is as follows:

```
// Set connection parameters
stConnect.byMode := 1; // Client mode
stConnect.stRemote.strIPAddr := '192.168.1.10';// IP address of remote server
stConnect.stRemote.uiPort := 8080;// Port of remote server
stConnect.uiID := 1;// ID number

// Receive TCP data
SktTCPRecv_C_Instance_0(
xEnable := TRUE,
xEnCont := TRUE,
stConnect := stConnect,// Connection parameters
DataBuffer := DataBuffer,
uiDataSize := ,
xDone => ,
xBusy => ,
xError => ,
wErrorID => ,
xReady => ,
uiCount => ,
eStatus => ,
stConnectInfo => );

// Send TCP data
SktTCPSend_C_Instance_0(
xExecute := SktTCPRecv_C_Instance_0.xReady,
xEnCont := TRUE,
stConnect := stConnect,// Connection parameters
DataBuffer := DataBuffer,
uiDataSize := SktTCPRecv_C_Instance_0.uiCount,
uiTimeOut := ,
xDone => ,
xBusy => ,
xError => ,
wErrorID => ,
eStatus => ,
stConnectInfo => );
```

Example of TCP server usage

Create a TCP server to listen on port 8080 of all network cards. After a successful connection is established, enable data reception, then wait for and receive the data from the client. Finally, send the received data back to the client.

- **Implementation with basic instructions**

Variable declarations are as follows:

```
VAR
en_start: BOOL;
start_listen: BOOL;
exit_connect: BOOL;
SktTCPServer_Instance_0: SktTCPServer;
SktClose_Instance_0: SktClose;
SktTCPRecv_Instance_0: SktTCPRecv;
SktTCPSend_Instance_0: SktTCPSend;
SktTCPGetStatus_Instance_0: SktTCPGetStatus;
stLocalAddress: ADDRESS;
uiID: UINT;
R_TRIG_Instance_0: R_TRIG;
F_TRIG_Instance_0: F_TRIG;
DataBuffer: ARRAY[1..1024] OF byte;
END_VAR
```

The logic implemented in the program editing area is as follows:

```
// Execute connection closure on the falling edge of en_start
F_TRIG_Instance_0(
CLK := en_start,
Q => );
IF F_TRIG_Instance_0.Q THEN
start_listen := FALSE;
exit_connect := TRUE;
END_IF

// create a TCP server and wait for client connection request
stLocalAddress.strIPAddr := '0.0.0.0';// Multi-network adapter device: 0.0.0.0 = listen on all network adapters
stLocalAddress.uiPort := 8080;// Port number
uiID := 1; // Use ID number 1
SktTCPServer_Instance_0(
xExecute := start_listen,
uiID := uiID,
stLocalAddress := stLocalAddress,
xDone => ,
xBusy => ,
xError => ,
wErrorID => ,
stConnectInfo => );

//Close connection
SktClose_Instance_0(
xExecute := exit_connect,
uiID := uiID,
xDone => ,
xBusy => ,
xError => ,
wErrorID => );

// Receive TCP data
SktTCPRecv_Instance_0(
xEnable := SktTCPServer_Instance_0.xDone,
```

```
uiID := uiID,
DataBuffer := DataBuffer,
uiDataSize := 0,
xDone => ,
xBusy => ,
xError => ,
wErrorID => ,
xReady => ,
uiCount => );

// Send TCP data
SktTCPSend_Instance_0(
xExecute := SktTCPRecv_Instance_0.xReady,
uiID := uiID,
DataBuffer := DataBuffer,
uiDataSize := SktTCPRecv_Instance_0.uiCount,
uiTimeOut := ,
xDone => ,
xBusy => ,
xError => ,
wErrorID => );

// Obtain connection status
SktTCPGetStatus_Instance_0(
xEnable := SktTCPServer_Instance_0.xDone,
uiID := uiID,
xDone => ,
xBusy => ,
xError => ,
wErrorID => ,
eStatus => ,
stConnectInfo => );
IF SktTCPGetStatus_Instance_0.eStatus = TCP_STATUS.TCP_ESTABLISHED THEN
// The connection is normal, and corresponding actions can be performed.
   ;
END_IF
```

- **Implementation with integrated instructions**

Variable declarations are as follows:

```
VAR
SktTCPRecv_C_Instance_0: SktTCPRecv_C;
SktTCPSend_C_Instance_0: SktTCPSend_C;
stConnect: TCP_CONNECT_PARAM;
DataBuffer: ARRAY[1..1024] OF byte;
END_VAR
```

The logic implemented in the program editing area is as follows:

```
// Set connection parameters
stConnect.byMode := 0; // Server mode
stConnect.stLocal.strIPAddr := '0.0.0.0'; // Local server IP address: '0.0.0.0' allows access from all network interfaces on multi-NIC devices
stConnect.stLocal.uiPort := 8080; // Local server port
stConnect.uiID := 1;// ID number

// Receive TCP data
SktTCPRecv_C_Instance_0(
xEnable := TRUE,
xEnCont := TRUE,
stConnect := stConnect,// Connection parameters
DataBuffer := DataBuffer,
uiDataSize := ,
xDone => ,
xBusy => ,
xError => ,
wErrorID => ,
xReady => ,
uiCount => ,
eStatus => ,
stConnectInfo => );

// Send TCP data
SktTCPSend_C_Instance_0(
xExecute := SktTCPRecv_C_Instance_0.xReady,
xEnCont := TRUE,
stConnect := stConnect,// Connection parameters
DataBuffer := DataBuffer,
uiDataSize := SktTCPRecv_C_Instance_0.uiCount,
uiTimeOut := ,
xDone => ,
xBusy => ,
xError => ,
wErrorID => ,
eStatus => ,
```

```
stConnectInfo => );
```

18.4.13 Free UDP

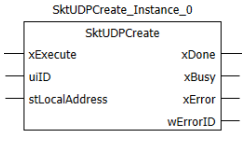
18.4.13.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.4.13.2 SktUDPCreate	FB	Create UDP communication service	TcpUdpFreeProtocol	Library version 1.0.0.0 and later
18.4.13.3 SktUDPRecv	FB	Receive UDP data		Library version 1.0.0.0 and later
18.4.13.4 SktUDPSend	FB	Send UDP data		Library version 1.0.0.0 and later
18.4.13.5 SktUDPJoinGroup	FB	Join UDP multicast		Library version 1.0.0.0 and later
18.4.13.6 SktUDPClearBuffer	FB	Clear UDP protocol stack buffer		Library version 1.1.0.0 and later
18.4.13.7 SktClose	FB	Close UDP communication service		Library version 1.0.0.0 and later

18.4.13.2 SktUDPCreate

This instruction creates UDP communication.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
SktUDPCreate	Create UDP communication service	FB		SktUDPCreate_Instance_0 (xExecute := , uiID := , stLocalAddress := , xDone => , xBusy => , xError => , wErrorID =>);

- **Related variables**

  Variable

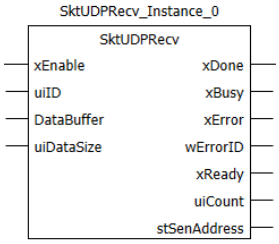
Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Trigger execution	BOOL	TRUE, FALSE	FALSE	Execution starts on rising edge.
	uiID	ID number	UINT	1 to 256	0	Each connection corresponds to an ID number.
	stLocalAddress	Local IP address	ADDRESS	-	-	Local IP address and port number (port range: 2000 to 65535)
Output	xDone	Done	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is done.
	xBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is executing.
	xError	Execution error	BOOL	TRUE, FALSE	FALSE	Error flag that is set to TRUE when an instruction execution error occurs.
	wErrorID	Fault ID	WORD	-	0	Outputs an error ID greater than 0 when the instruction execution error occurs. For details, see 18.4.1 2.11 Fault Codes.

- **Function**
  - Before the instruction execution, input uiID to specify the ID number to be created for subsequent calls to SktUDPSend, SktUDPRecv, and SktClose. Input StLocalAddress to set the local IP and port number for UDP communication. When the IP is set to 0.0.0.0, data can be received on all network adapters. The port number should be greater than 2000.
  - During instruction execution, xExecute=TRUE is maintained.
  - Set xDone to TRUE when the instruction is done and the communication is successfully established.
  - If an error occurs during instruction execution, xError is set to TRUE, and wErrorID outputs an error code to explain the cause of the error.
- **Precautions**
  - After a successful connection, uiID will no longer be bound to the instruction. To close the UDP communication and release the ID, the SktClose instruction must be called.
  - In on-site applications, multiple tasks should not call the same instance of SktUDPCreate.

18.4.13.3 SktUDPRecv

This instruction receives UDP data.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
SktUDPRecv	Receive UDP data	FB		SktUDPRecv_Instance_0(xEnable := , uiID := , DataBuffer := , uiDataSize := , xDone => , xBusy => , xError => , wErrorID => , xReady => , uiCount => , stSenAddress =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Enable	BOOL	TRUE, FALSE	FALSE	Input TRUE to execute reception and FALSE to exit reception.
	uiID	ID number	UINT	1 to 256	0	Each connection corresponds to an ID number.
	DataBuffer	Data reception buffer	ANY	-	-	It supports any data type, with array types limited to a maximum of three dimensions.
	uiDataSize	Size of received data	UINT	0 to 2048	0	Size of received data in byte. uiDataSize=0 indicates the maximum value of the custom cache array.
Output	xDone	Reserved	BOOL	TRUE, FALSE	FALSE	Always FALSE.
	xBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is executing, waiting for connection requests.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	xError	Execution error	BOOL	TRUE, FALSE	FALSE	Error flag that is set to TRUE when an instruction execution error occurs.
	wErrorID	Fault ID	WORD	-	0	Outputs an error ID greater than 0 when the instruction execution error occurs. For details, see 18.4.12.11 Fault Codes.
	xReady	Reception completion flag	BOOL	TRUE, FALSE	FALSE	Set to TRUE when data reception is done and maintain this state for one task cycle.
	uiCount	Actual size of received data	UINT	0 to 2048	0	Outputs the actual size of received data in byte when xReady=TRUE and maintains this state for one task cycle.
	stSenAddress	Sender Address	ADDRESS	-	-	Sender IP address and port number (port range: 2000 to 65535)

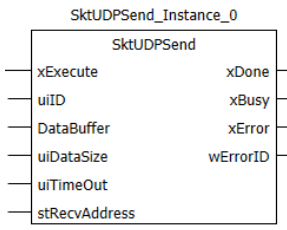
- **Function**
  - Before instruction execution (xEnable=TRUE), input uiID to specify the ID number (created by the SktUDPCreate); input DataBuffer to specify the application layer of data buffer, which supports one-dimensional arrays of BYTE type; input uiDataSize (in byte) to specifies the size of the data to be received, which must not exceed the size of DataBuffer or a maximum of 2048 bytes. If uiDataSize is set to 0, the system will default to using the buffer size of DataBuffer as the length of the received data.
  - xBusy is set to TRUE, and data reception continues during instruction execution (xEnable=TRUE maintained). When data is received, output the reception completion flag xReady=TRUE, the actual size of received data size uiCount, and the sender IP address and port stSenAddress**.** This state is maintained for only one cycle.
  - If an error occurs during instruction execution, xError is set to TRUE, and wErrorID outputs a fault code to explain the cause of the error.
- **Precautions**

  In on-site applications, multiple tasks should not call the same instance of SktTCPRecv.

18.4.13.4 SktUDPSend

This instruction sends UDP data.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
SktUDPSend	Send UDP data	FB		SktUDPSend_Instance_0(xExecute := , uiID := , DataBuffer := , uiDataSize := , uiTimeOut := , stRecvAddress := , xDone => , xBusy => , xError => , wErrorID =>);

- **Related variables**

Variable

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Trigger execution	BOOL	TRUE, FALSE	FALSE	Data sending is triggered on a rising edge.
	uiID	ID number	UINT	1 to 256	0	Each connection corresponds to an ID number.
	DataBuffer	Sending data buffer	ANY	-	-	It supports any data type, with array types limited to a maximum of three dimensions.
	uiDataSize	Size of sent data	UINT	1 to 2048	0	Size of sent data in byte.
	uiTimeOut	Sending timeout	UINT	1 to 1000	50	Sending timeout in ms.
	stRecvAddress	Receiver address	ADDRESS	-	-	Receiver IP address and port number (port range: 2000 to 65535).
Output	xDone	Done	BOOL	TRUE, FALSE	FALSE	Set to TRUE when data sending is done.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	xBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is executing, waiting for connection requests.
	xError	Execution error	BOOL	TRUE, FALSE	FALSE	Error flag that is set to TRUE when an instruction execution error occurs.
	wErrorID	Fault ID	WORD	-	0	Outputs an error ID greater than 0 when the instruction execution error occurs. For details, see 18.4.1 2.11 Fault Codes.

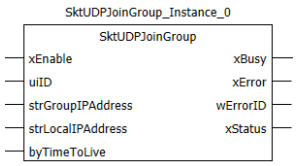
- **Function**
  - Before the instruction execution (xExecute=TRUE), input uiID to specify the ID number (created by SktUDPCreate instruction); input uiDataSize in byte as the data size to be sent, which must not exceed the size of DataBuffer or a maximum of 2048 bytes; input sending timeout uiTimeOut (in ms, default: 50); the default value is sufficient for most cases, and it can be appropriately increased when network congestion occurs.
  - During instruction execution, xExecute=TRUE is maintained.
  - After instruction execution, set xDone to TRUE.
  - If an error occurs during instruction execution, xError is set to TRUE, and wErrorID outputs an error code to explain the cause of the error.
- **Precautions**

  In on-site applications, multiple tasks should not call the same instance of SktUDPSend.

18.4.13.5 SktUDPJoinGroup

The instruction joins UDP multicast group.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
SktUDPJoinGroup	Join UDP multicast	FB		SktUDPJoinGroup_Instance_0(xEnable := , uiID := , strGroupIPAddress := , strLocalIPAddress := , byTimeToLive := , xBusy => , xError => , wErrorID => , xStatus =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Enable	BOOL	TRUE, FALSE	FALSE	Enable joining multicast.
	uiID	ID number	UINT	1 to 256	0	Each UDP port corresponds to an ID number.
	strGroupIPAddress	Multicast address	STRING	-	-	Multicast IP address.
	strLocalIPAddress	Local IP address	STRING	-	-	Local IP address.
	byTimeToLive	Time to live	BYTE	0 to 255	0	Time to live (number of routings.)
Output	xBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is executing, and FALSE when the execution ends.
	xError	Execution error	BOOL	TRUE, FALSE	FALSE	Error flag that is set to TRUE when an instruction execution error occurs.
	wErrorID	Fault ID	WORD	-	0	Outputs an error ID greater than 0 when the instruction execution error occurs. For details, see 18.4.12.11 Fault Codes.

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	xStatus	Status	BOOL	TRUE, FALSE	FALSE	Set to TRUE when successfully joined the multicast group.

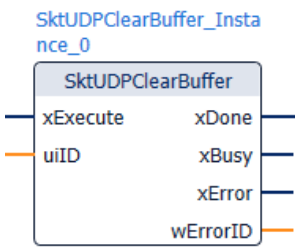
- **Function**
  - Before the instruction is executed (xEnable set to TRUE), Input uiID to specify the ID number (created by the SktUDPCreate instruction). strGroupIPAddress: Multicast IP address (D-class IPv4 range: 224.0.0.0 to 239.255.255.255, allocated by standard organizations for multicast use). strLocalIPAddress: IP address of the local network adapter. For devices with multiple network adapters, input the IP address of the corresponding adapter to join the multicast group. byTimeToLive (TTL): Time-to-live, determining the number of routings the multicast packet can traverse.
  - xBusy is set to TRUE during instruction execution (xEnable=TRUE maintained). xStatus is set to TRUE when the multicast group is joined successfully.
  - xBusy is set to FALSE, and the multicast group is exited upon completion of execution (xEnable=FALSE).
  - If an error occurs during instruction execution, xError is set to TRUE, and wErrorID outputs a fault code to explain the cause of the error.
- **Precautions**

  In on-site applications, multiple tasks should not call the same instance of SktUDPJoinGroup.

18.4.13.6 SktUDPClearBuffer

This instruction clears the UDP protocol stack buffer.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SktUDPClearBuffer	Clear UDP protocol stack buffer	FB		SktUDPClearBuffer_Instance_0(xExecute := , uiID := , xDone => , xBusy => , xError => , wErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Trigger execution	BOOL	TRUE, FALSE	FALSE	Buffer clearance is triggered on a rising edge.
	uiID	ID number	UINT	1 to 256	0	Each UDP port corresponds to an ID number.

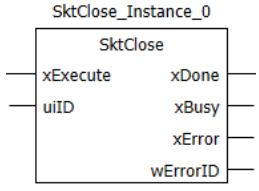
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	xDone	Done	BOOL	Follows data type	-	Set to TRUE when the instruction is done.
	xBusy	Executing	BOOL	Follows data type	-	Set to TRUE when the instruction is executing.
	xError	Execution error	BOOL	Follows data type	-	Error flag that is set to TRUE when an instruction execution error occurs.
	wErrorID	Fault code	WORD	Follow data type	0	Outputs an error ID greater than 0 when the instruction execution error occurs. For details, see 18.4.12.11 Fault Codes.

- **Function**
  SktUDPClearBuffer is the instruction used to clear the UDP protocol stack data buffer.
  - Before the instruction execution (xEnable set to TRUE), Input uiID to specify the ID number (created by the SktUDPCreate instruction).
  - After the execution is completed, xDone is set to TRUE, indicating that the UDP protocol stack buffer has been cleared. If the UDP protocol stack receives new data immediately after the execution is completed, its buffer cannot be guaranteed to be cleared.
  - When an execution error occurs, xError is set to TRUE. You can analyze the cause of the error based on the output wErrorID.
- **Precaution**
  In on-site applications, multiple tasks should not call the same instance of SktUDPClearBuffer.

18.4.13.7 SktClose

Close UDP communication.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	ST Representation
SktClose	Close UDP communication service	FB		SktClose_Instance_0(xExecute := , uiID := , xDone => , xBusy => , xError => , wErrorID =>);

- **Related variables**

  Variable

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Trigger execution	BOOL	TRUE, FALSE	FALSE	Communication shutdown is triggered on a rising edge.
	uiID	ID number	UINT	1 to 256	0	Each connection corresponds to an ID number.
Output	xDone	Done	BOOL	TRUE, FALSE	FALSE	Instruction completion flag set to TRUE.
	xBusy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the instruction is executing.
	xError	Execution error	BOOL	TRUE, FALSE	FALSE	Error flag that is set to TRUE when an instruction execution error occurs.
	wErrorID	Fault ID	WORD	-	0	Outputs an error ID greater than 0 when the instruction execution error occurs. For details, see 18.4.12.11 Fault Codes.

- **Function**
  - Before the instruction execution(xExecute=TRUE), Input uiID to specify the ID number (created by the SktUDPCreate instruction).
  - During instruction execution, xExecute=TRUE is maintained.
  - After the instruction execution, set xDone to TRUE, at which point the UDP communication closing is terminated and the ID number is released.
  - If an error occurs during instruction execution, xError is set to TRUE, and wErrorID outputs a fault code to explain the cause of the error.
- **Precautions**

  In on-site applications, multiple tasks should not call the same instance of SktClose.

18.4.13.8 Example of Free UDP Communication Instruction Usage

The section involves creating UDP communication, receiving data from port 8080 of all local network adapters, and returning the received data to the sender via the original path.

Variable declarations are as follows:

```
VAR
en_start: BOOL;
start_udp: BOOL;
exit_udp: BOOL;
SktUDPCreate_Instance_0: SktUDPCreate;
SktUDPRecv_Instance_0: SktUDPRecv;
SktUDPSend_Instance_0: SktUDPSend;
SktClose_Instance_0: SktClose;
R_TRIG_Instance_0: R_TRIG;
F_TRIG_Instance_0: F_TRIG;
stLocalAddress: ADDRESS;
uiID: UINT;
DataBuffer: ARRAY[1..1024] OF byte;
END_VAR
```

The logic implemented in the program editing area is as follows:

```
// Create UDP on the rising edge of en_start
R_TRIG_Instance_0(
CLK := en_start,
Q => );
IF R_TRIG_Instance_0.Q THEN
start_udp := TRUE;
exit_udp := FALSE;
END_IF

//Close UDP on the falling edge of en_start
F_TRIG_Instance_0(
CLK := en_start,
Q => );
IF F_TRIG_Instance_0.Q THEN
start_udp := FALSE;
exit_udp := TRUE;
END_IF

//Create UDP
stLocalAddress.strIPAddr := '0.0.0.0';
stLocalAddress.uiPort := 8080;
uiID := 1;
SktUDPCreate_Instance_0(
xExecute := start_udp,
uiID := uiID,
stLocalAddress := stLocalAddress,
xDone => ,
xBusy => ,
xError => ,
wErrorID => );

//close UDP
SktClose_Instance_0(
xExecute := exit_udp,
uiID := uiID,
xDone => ,
```

```
xBusy => ,
xError => ,
wErrorID => );

// Receive UDP data
SktUDPRecv_Instance_0(
xEnable := SktUDPCreate_Instance_0.xDone,
uiID := uiID,
DataBuffer := DataBuffer,
uiDataSize := ,
xDone => ,
xBusy => ,
xError => ,
wErrorID => ,
xReady => ,
uiCount => ,
stSenAddress => );

//Send UDP data
SktUDPSend_Instance_0(
xExecute := SktUDPRecv_Instance_0.xReady,
uiID := uiID,
DataBuffer := DataBuffer,
uiDataSize := SktUDPRecv_Instance_0.uiCount,
uiTimeOut := ,
stRecvAddress := SktUDPRecv_Instance_0.stSenAddress,
xDone => ,
xBusy => ,
xError => ,
wErrorID => );
```

18.4.14 S7 Communication

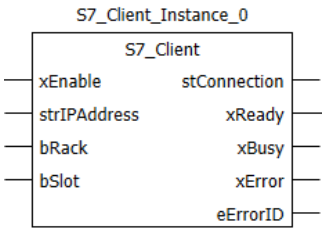
18.4.14.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.4.14.2 S7_Client	FB	Create a S7 client	S7Comm	Library version 1.0.0.0 and later
18.4.14.3 S7_Server	FB	Create a S7 server		Library version 1.0.0.0 and later
18.4.14.4 S7_Get	FB	Read data as an S7 client		Library version 1.0.0.0 and later
18.4.14.5 S7_Put	FB	Write data as an S7 client		Library version 1.0.0.0 and later
18.4.14.6 S7_Data_Area_Create	FB	Create memory areas on S7 server		Library version 1.0.0.0 and later

18.4.14.2 S7_Client

This instruction create a S7 client.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
S7_Client	Create a S7 client	FB		S7_Client_Instance_0(xEnable := , strIPAddress := , bRack := , bSlot := , stConnection => , xReady => , xBusy => , xError => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Function block enable	BOOL	TRUE, FALSE	FALSE	Level triggered
	strIPAddress	Remote IP address	STRING	Follows data type	''	Remote IP Address

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	bRack	Rack ID	BYTE	Follows data type	0	Unused and reserved
	bSlot	Slot ID	BYTE	Follows data type	0	Unused and reserved
Output	stConnection	Connection handle	S7_Connection	Follows data type	-	Connection handle.
	xReady	Ready function block	BOOL	TRUE, FALSE	-	Indicates the function block is ready for execution
	xBusy	Function block execution flag	BOOL	TRUE, FALSE	-	Function block execution flag
	xError	Function block fault flag	BOOL	TRUE, FALSE	-	Function block fault flag
	eErrorID	Function block fault ID	S7_ERROR	Follows data type	-	Function block fault ID. For details, see 18.4.14.7 Fault Codes.

- **Function**
  This instruction connects to the server at specified IP when xEnable is set to TRUE and disconnects when xEnable is set to FALSE.
- **Program example**

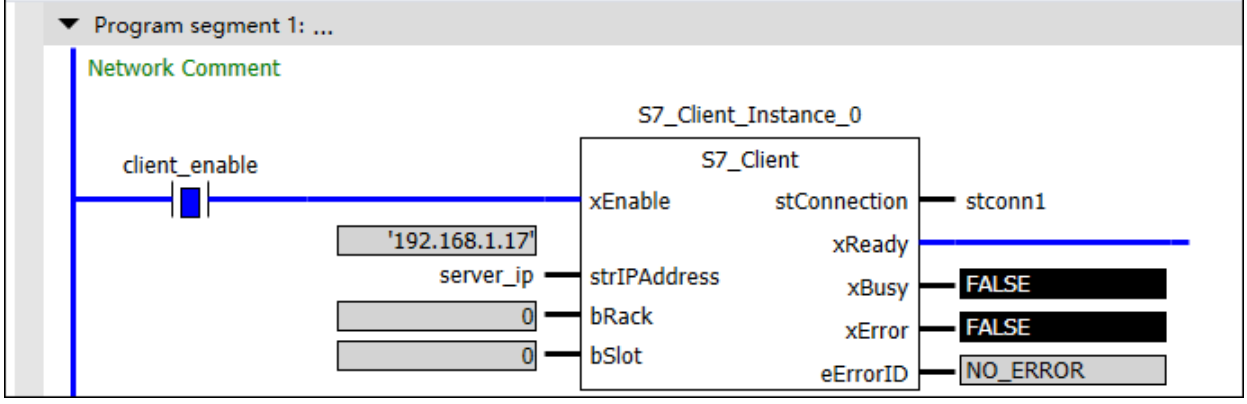
  ST:

Name	Type	Address	Value
S7_Client_Instance_0	S7_Client		
client_enable	BOOL		TRUE
ip	STRING		'192.168.1.17'
client_stconn	S7_Connection		
client_ready	BOOL		TRUE
client_busy	BOOL		FALSE
client_error	BOOL		FALSE
client_errorid	S7_ERROR		NO_ERROR
S7_Get_Instance_0	S7_Get		
get_execute	BOOL		FALSE

```
S7_Client_Instance_0(xEnable TRUE := client_enable TRUE ,
                     strIPAddress '192.168.1.17' := ip '192.168.1.17' ,
                     bRack := ,
                     bSlot := ,
                     stConnection => client_stconn,
                     xReady TRUE => client_ready TRUE ,
                     xBusy FALSE => client_busy FALSE ,
                     xError FALSE => client_error FALSE ,
                     eErrorID NO_ERROR => client_errorid NO_ERROR );
```

LD:

Name	Type	Address	Value
S7_Client_Instance_0	S7_Client		
xEnable	BOOL		TRUE
strIPAddress	STRING		'192.168.1.17'
bRack	BYTE		0
bSlot	BYTE		0
stConnection	S7_Connection		
xReady	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE



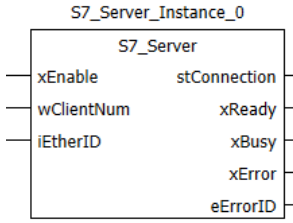
- **Precautions**
  - bRack and bSlot are reserved variables, which do not need to be configured and can be left at default values.
  - xBusy is set during reconnection upon server disconnection, After successful reconnection, it will be set again. Reconnect time is 10s.

◦ A maximum of eight client instructions can be executed at the same time, an error will be reported if this number is exceeded.

18.4.14.3 S7_Server

This instruction create a S7 server.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
S7_Server	Create a S7 server	FB		S7_Server_Instance_0(xEnable := , wClientNum := , iEtherID := , stConnection => , xReady => , xBusy => , xError => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Function block enable	BOOL	TRUE, FALSE	FALSE	Level triggered
	wClientNum	Number of connected clients	WORD	1 to 8	0	Maximum number of connected clients
	iEtherID	Network adapter ID	INT	Follows data type	0	Network port ID bound to the server 0: Port A 1: Port B It changes in incremental mode.
Output	stConnection	Connection handle	S7_Connection	Follows data type	-	Connection handle.
	xReady	Ready function block	BOOL	TRUE, FALSE	-	Indicates the function block is ready for execution
	xBusy	Function block execution flag	BOOL	TRUE, FALSE	-	Function block execution flag
	xError	Function block fault flag	BOOL	TRUE, FALSE	-	Function block fault flag

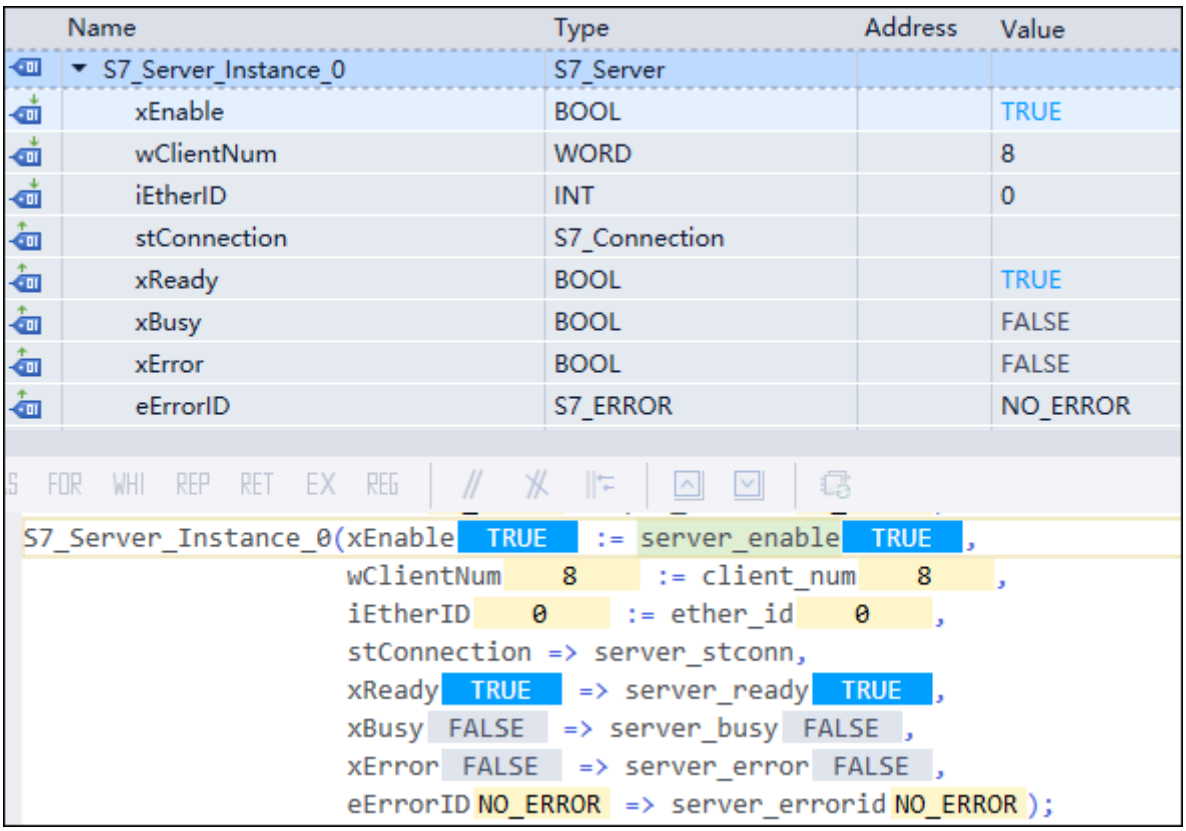
Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	eErrorID	Function block fault ID	S7_ERROR	Follows data type	-	Function block fault ID. For details, see 18.4.14.7 Fault Codes.

- **Function**

  This instruction enables server startup and port binding via iEtherID with maximum numbers of clients set by wClientNum when xEnable is set to TRUE. It shuts down the server when xEnable is set to false.

- **Program example**

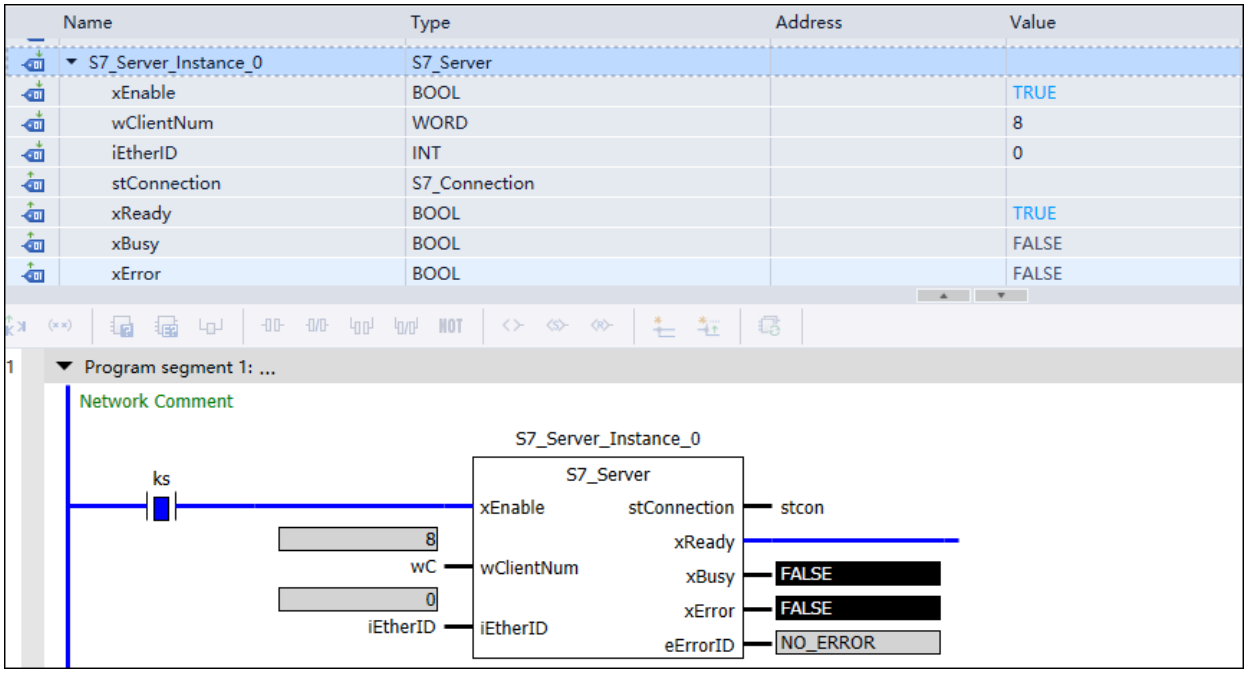
  ST:



Name	Type	Address	Value
S7_Server_Instance_0	S7_Server		
xEnable	BOOL		TRUE
wClientNum	WORD		8
iEtherID	INT		0
stConnection	S7_Connection		
xReady	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
eErrorID	S7_ERROR		NO_ERROR

```
S7_Server_Instance_0(xEnable TRUE := server_enable TRUE,
                     wClientNum 8 := client_num 8,
                     iEtherID 0 := ether_id 0,
                     stConnection => server_stconn,
                     xReady TRUE => server_ready TRUE,
                     xBusy FALSE => server_busy FALSE,
                     xError FALSE => server_error FALSE,
                     eErrorID NO_ERROR => server_errorid NO_ERROR);
```

  LD:

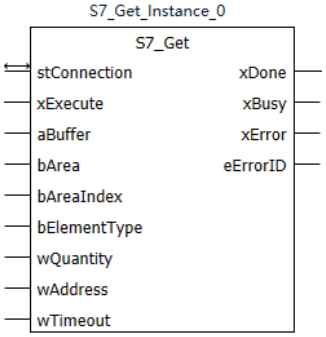


- **Precautions**
  - One port matches one S7_Server. Each S7_Server supports up to eight client connections. 0 maps to Port A and 1 maps to Port B in order.
  - The network adapter can only be called when it is in the available state. If it is in the EtherCAT state, an error will be reported.

18.4.14.4 S7_Get

This instruction reads data via the S7 protocol as an S7 client.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
S7_Get	Read data as an S7 client	FB		S7_Get_Instance_0(stConnection := , xExecute := , aBuffer := , bArea := , bAreaIndex := , bElementType := , wQuantity := , wAddress := , wTimeout := , xDone => , xBusy => , xError => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input/ Output	stConnection	Handle of S7_Client	S7_Connection	Follows data type	-	Handle of S7_Client
Input	xExecute	Function block enable	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	aBuffer	Data buffer	ANY	Follows data type	-	Data Buffer
	bArea	Data area type	S7_AREA_TYPE	Follows data type	S7_AREA_DB	Data area type 0: S7_AREA_INPUTS (I area) 1: S7_AREA_OUTPUTS (Q area) 2: S7_AREA_MERKERS (M area) 5: S7_AREA_DB (DB area)
	bAreaIndex	Data area index	BYTE	Follows data type	0	It is valid only for DB data area.
	bElementType	Data element type to access	S7_ELEMENT_TYPE	Follows data type	S7_BYTE	Element type 1: S7_BIT (Bit access) 2: S7_BYTE (Byte access) 3: S7_CHAR (Character access) 4: S7_WORD (Word access) 5: S7_INT (Integer access) 6: S7_DWORD (Double word access) 7: S7_DINT (Double integer access) 8: S7_REAL (Real number access)
	wQuantity	Number of elements to access	WORD	Follow data type	0	Number of elements to access
	wAddress	Remote data base address	WORD	Follow data type	0	Start address for accessing the remote data area

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	wTimeout	Timeout interval (ms)	WORD	Follow data type	1000	Maximum time for remote response
Output	xDone	Function block execution completion	BOOL	TRUE, FALSE	-	Function block execution completion flag
	xBusy	Function block execution flag	BOOL	TRUE, FALSE	-	Function block execution flag
	xError	Function block fault flag	BOOL	TRUE, FALSE	-	Function block fault flag
	eErrorID	Function block fault ID	S7_ERROR	Follows data type	-	Function block fault ID. For details, see 18.4.14.7 Fault Codes.

- **Function**
  This instruction reads the server data into buffer when xExecute is set to TRUE.
- **Program example**

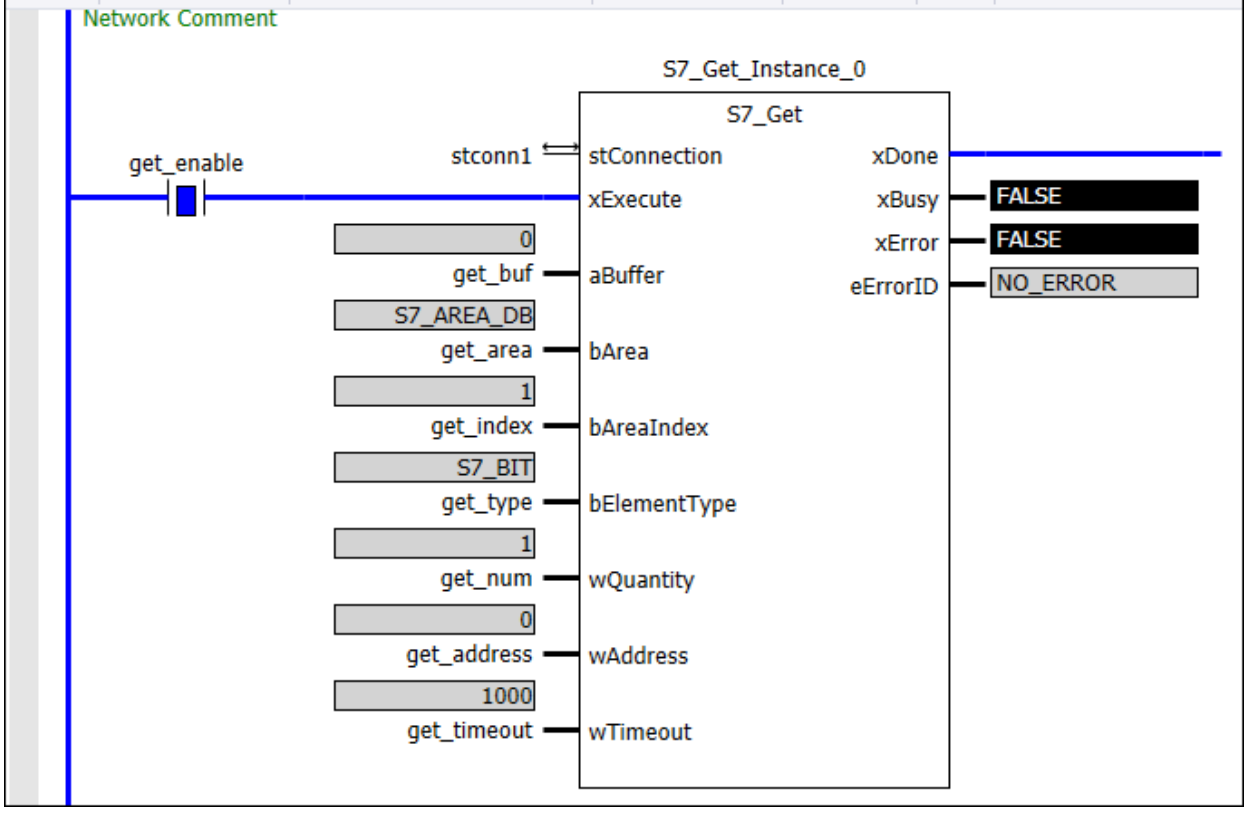
  ST:

Name	Type	Address	Value
S7_Get_Instance_0	S7_Get		
stConnection	S7_Connection		
xExecute	BOOL		TRUE
aBuffer	ANY		
bArea	S7_AREA_TYPE		S7_AREA_DB
bAreaIndex	BYTE		1
bElementType	S7_ELEMENT_TYPE		S7_BIT
wQuantity	WORD		1
wAddress	WORD		0
wTimeout	WORD		1000
xDone	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
eErrorID	S7_ERROR		NO_ERROR

```
S7_Get_Instance_0(stConnection := client_stconn,
                  xExecute TRUE  := get_execute TRUE ,
                  aBuffer := get_buf   0    ,
                  bAreaS7_AREA... := get_areaS7_AREA...,
                  bAreaIndex   1    := get_index   1    ,
                  bElementType S7_BIT  := get_type S7_BIT ,
                  wQuantity    1      := get_num    1      ,
                  wAddress     0      := get_address     0      ,
                  wTimeout   1000   := get_timeout   1000   ,
                  xDone TRUE  => get_done TRUE ,
                  xBusy FALSE  => get_busy FALSE ,
                  xError FALSE  => get_error FALSE ,
                  eErrorID NO_ERROR  => get_errorid NO_ERROR );
```

LD:

Name	Type	Address	Value
S7_Get_Instance_0	S7_Get		
stConnection	S7_Connection		
xExecute	BOOL		TRUE
aBuffer	ANY		
bArea	S7_AREA_TYPE		S7_AREA_DB
bAreaIndex	BYTE		1
bElementType	S7_ELEMENT_TYPE		S7_BIT
wQuantity	WORD		1
wAddress	WORD		0
wTimeout	WORD		1000
xDone	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
eErrorID	S7_ERROR		NO_ERROR

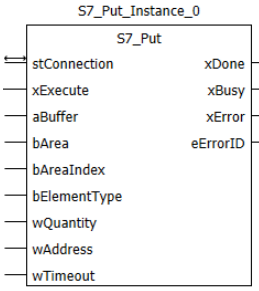


- **Precautions**
  - Multiple S7_Get instructions cannot be called at the same time and must be executed in sequence. Start the next call only after the Done flag of the current instruction is set. Otherwise, an error will be reported. In practical applications, the Done flag of previous instruction can trigger the next instruction.
  - When bElementType is set to S7_BIT, the value of wQuantity is 1, indicating single-bit access.
  - When bElementType is set to S7_BIT, wAddress adopts bitwise addressing. When bElementType is set to other data types,
    wAddress adopts bytewise addressing.

18.4.14.5 S7_Put

This instruction writes data via the S7 protocol as an S7 client.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
S7_Put	Write data as an S7 client	FB		S7_Put_Instance_0(stConnection := , xExecute := , aBuffer := , bArea := , bAreaIndex := , bElementType := , wQuantity := , wAddress := , wTimeout := , xDone => , xBusy => , xError => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input/ Output	stConnection	Handle of S7_Client	S7_Connection	Follow data type	-	Handle of S7_Client
Input	xExecute	Function block enable	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	aBuffer	Data buffer	ANY	Follows data type	-	Data Buffer
	bArea	Data area type	S7_AREA_TYPE	Follows data type	S7_AREA_DB	Data area type 0: S7_AREA_INPUTS (I area) 1: S7_AREA_OUTPUTS (Q area) 2: S7_AREA_MERKERS (M area) 5: S7_AREA_DB (DB area)
	bAreaIndex	Data area index	BYTE	Follows data type	0	It is valid only for DB data area.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	bElementType	Data element type to access	S7_ELEMENT_TYPE	Follows data type	S7_BYTE	Element type 1: S7_BIT 2: S7_BYTE 3: S7_CHAR 4: S7_WORD 5: S7_INT 6: S7_DWORD 7: S7_DINT 8: S7_REAL
	wQuantity	Number of elements to access	WORD	Follow data type	0	Number of elements to access
	wAddress	Remote data base address	WORD	Follow data type	0	Start address for accessing the remote data area
	wTimeout	Timeout interval (ms)	WORD	Follow data type	1000	Maximum time for remote response
Output	xDone	Ready function block	BOOL	TRUE, FALSE	-	Function block execution completion flag
	xBusy	Function block execution flag	BOOL	TRUE, FALSE	-	Function block execution flag
	xError	Function block fault flag	BOOL	TRUE, FALSE	-	Function block fault flag
	eErrorID	Function block fault ID	S7_ERROR	Follows data type	-	Function block fault ID. For details, see 18.4.14.7 Fault Codes.

- **Function**
  This instruction writes the buffer data into server when xExecute is set to TRUE.
- **Program example**

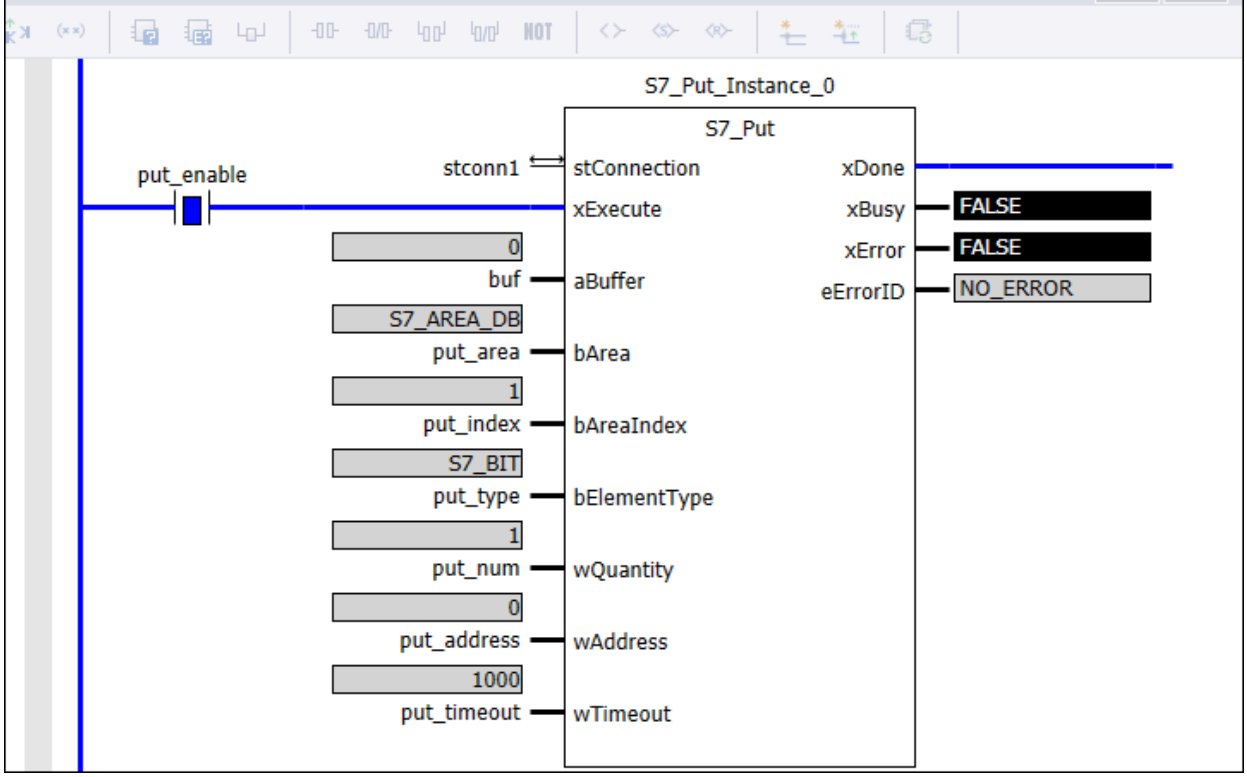
  ST:

Name	Type	Address	Value
S7_Put_Instance_0	S7_Put		
stConnection	S7_Connection		
xExecute	BOOL		TRUE
aBuffer	ANY		
bArea	S7_AREA_TYPE		S7_AREA_DB
bAreaIndex	BYTE		1
bElementType	S7_ELEMENT_TYPE		S7_BIT
wQuantity	WORD		1
wAddress	WORD		0
wTimeout	WORD		1000
xDone	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
eErrorID	S7_ERROR		NO_ERROR
s7_err	DINT		0

```
S7_Put_Instance_0(stConnection := client_stconn,
                  xExecute TRUE  := put_execute TRUE ,
                  aBuffer := put_buf    0     ,
                  bAreaS7_AREA... := put_areaS7_AREA...,
                  bAreaIndex   1    := put_index   1    ,
                  bElementType S7_BIT  := put_type S7_BIT ,
                  wQuantity    1    := put_num    1    ,
                  wAddress     0    := put_address    0    ,
                  wTimeout   1000   := put_timeout  1000   ,
                  xDone  TRUE  => put_done  TRUE ,
                  xBusy  FALSE => put_busy  FALSE ,
                  xError FALSE => put_error FALSE ,
                  eErrorID NO_ERROR => put_errorid NO_ERROR );
```

LD:

Name	Type	Address	Value
S7_Put_Instance_0	S7_Put		
stConnection	S7_Connection		
xExecute	BOOL		TRUE
aBuffer	ANY		
bArea	S7_AREA_TYPE		S7_AREA_DB
bAreaIndex	BYTE		1
bElementType	S7_ELEMENT_TYPE		S7_BIT
wQuantity	WORD		1
wAddress	WORD		0
wTimeout	WORD		1000
xDone	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
eErrorID	S7_ERROR		NO_ERROR

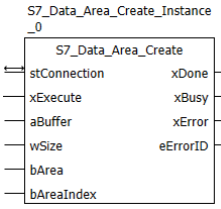


- **Precautions**
  - Multiple S7_Put instructions cannot be called at the same time and must be executed in sequence. Start the next call only after the Done flag of the current instruction is set. Otherwise, an error will be reported. In practical applications, the Done flag of previous instruction can trigger the next instruction.
  - When bElementType is set to S7_BIT, the value of wQuantity is 1, indicating single-bit access.
  - When bElementType is set to S7_BIT, wAddress adopts bitwise addressing. When bElementType is set to other data types,
    wAddress adopts bytewise addressing.

18.4.14.6 S7_Data_Area_Create

This instruction creates data memory areas on the S7 protocol communication server.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
S7_Data_Area_Create	Create memory areas on S7 server	FB		S7_Data_Area_Create_Instance_0(stConnection := , xExecute := , aBuffer := , wSize := , bArea := , bAreaIndex := , xDone => , xBusy => , xError => , eErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input/ Output	stConnection	Handle of S7_Client	S7_Connection	Follow data type	-	Handle of S7_Client
Input	xExecute	Function block enable	BOOL	TRUE, FALSE	FALSE	Rising edge-triggered
	aBuffer	Data buffer	ANY	Follows data type	-	Data Buffer
	wSize	Size of data area	WORD	Follow data type	0	Size of data area (bytes)
	bArea	Data area type	S7_AREA_TYPE	Follows data type	S7_AREA_DB	Data area type 0: S7_AREA_INPUTS (I area) 1: S7_AREA_OUTPUTS (Q area) 2: S7_AREA_MERKERS (M area) 5: S7_AREA_DB (DB area)
	bAreaIndex	Data area index	BYTE	Follows data type	0	It is valid only for DB data area.
Output	xDone	Function block execution completion	BOOL	TRUE, FALSE	-	Function block execution completion flag

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	xBusy	Function block execution flag	BOOL	TRUE, FALSE	-	Function block execution flag
	xError	Function block fault flag	BOOL	TRUE, FALSE	-	Function block fault flag
	eErrorID	Function block fault ID	S7_ERROR	Follows data type	-	Function block fault ID. For details, see 18.4.14.7 Fault Codes.

- **Function**

  When xExecute is set to TRUE, this instruction creates data memory areas (I/Q/M/DB) based on the value of bArea for client access. When xExecute is set to FALSE, it destroys the data areas.

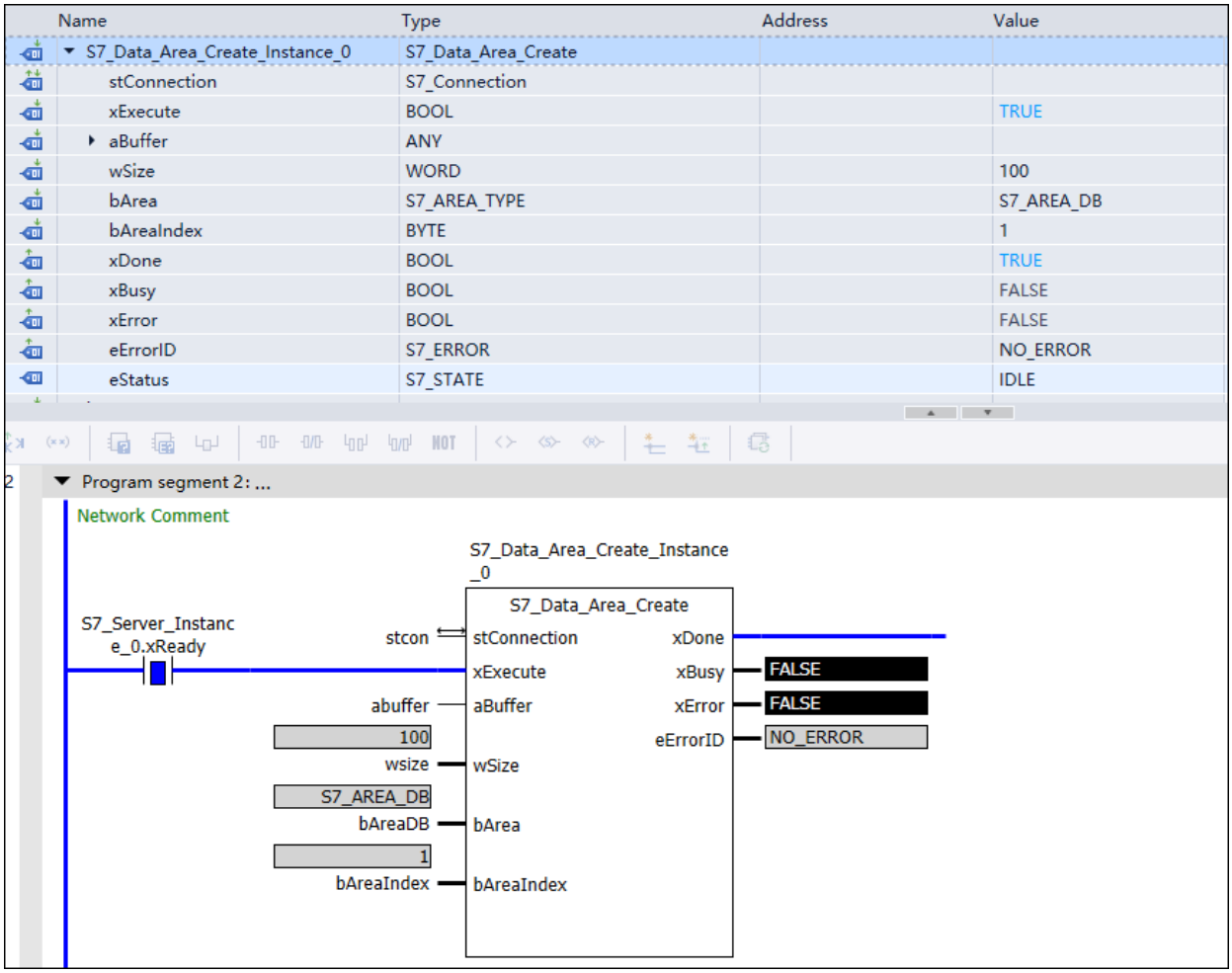
- **Program example**

  ST:

Name	Type	Address	Value
S7_Data_Area_Create_Instance_0	S7_Data_Area_Create		
stConnection	S7_Connection		
xExecute	BOOL		TRUE
aBuffer	ANY		
wSize	WORD		1
bArea	S7_AREA_TYPE		S7_AREA_DB
bAreaIndex	BYTE		0
xDone	BOOL		TRUE
xBusy	BOOL		FALSE
xError	BOOL		FALSE
eErrorID	S7_ERROR		NO_ERROR

```
S7_Data_Area_Create_Instance_0(stConnection := server_stconn,
                               xExecute TRUE  := server_execute TRUE ,
                               aBuffer := server_buf  0 ,
                               wSize  1  := server_size  1 ,
                               bAreaS7_AREA... := server_areaS7_AREA...,
                               bAreaIndex  0  := server_index  0 ,
                               xDone TRUE  => server_done TRUE ,
                               xBusy FALSE  => area_busy FALSE ,
                               xError FALSE  => area_error FALSE ,
                               eErrorID NO_ERROR => area_errorid NO_ERROR );
```

  LD:



- **Precautions**
  - The data areas (I/Q/M/DB) created by this instruction serve only as area identifiers and are used only for remote read/write access. For the interaction with medium-sized PLCs, map the array to the I/Q/M areas.
  - The data areas (I/Q/M) created by this instruction are unique. Multiple DB data areas can be created, but their data area index (bAreaIndex) must be unique.

18.4.14.7 Fault Codes

Name	Fault Code	Description
NO_ERROR	0	No error.
CONNECT_FAILED	1	Failed to create connection
CONNECT_MODE_INVALID	2	Connection mode invalid
CONNECTION_INVALID	3	Connection handle invalid
CONNECTION_OCCUPY	4	Connection handle occupied
CONNECTION_MODIFY	5	Connection handle modification error
MEMORY_ADDR_INVALID	6	Memory address invalid
MEMORY_OUT_OF_BOUNDS	7	Memory access out of bounds
DATA_DIM_INVALID	8	Data dimension invalid (only one-dimensional supported)
DATA_ADDR_INVALID	9	Invalid data address.

Name	Fault Code	Description
DATA_TYPE_INVALID	10	Data type invalid
DATA_VALUE_INVALID	11	Data value invalid
GET_DATA_FAILED	12	Failed to read data
PUT_DATA_FAILED	13	Failed to write data
GET_DATA_TIMEOUT	14	Data read timeout
PUT_DATA_TIMEOUT	15	Data write timeout
CLIENT_COUNT_MAX	16	Maximum number of clients reached
AREA_NO_EXIST	17	No data area
AREA_ALREADY_EXIST	18	Already existing data area
SERVER_START_FAILED	19	Failed to start the server
UNKNOWN	65535	Unknown error

18.4.15 FINS Communication

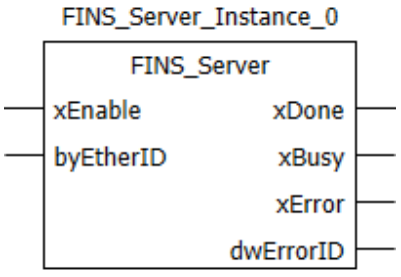
18.4.15.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable
18.4.15.2 FINS_Server	FB	Start the FINS server	HC_FINS	Library version 1.0.0.0 and later
18.4.15.3 FINS_ServerDiag	FB	Get the diagnostic information of FINS server		

18.4.15.2 FINS_Server

This instruction starts the FINS server.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
FINS_Server	Start the FINS server	FB		FINS_Server_Instance_0(xEnable := , byEtherID := , xDone => , xBusy => , xError => , dwErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Function block enable	BOOL	TRUE, FALSE	FALSE	Function block enable flag • When xEnable is set to TRUE, it indicates that the FINS server is enabled, and clients can read and write PLC data over TCP/UDP connections. • When xEnable is set to FALSE, it indicates that the FINS server is disabled, and data exchanges with clients over TCP/UDP connections are terminated.
	byEtherID	Ethernet network ID	BYTE	Follows data type	0	The valid values of Ethernet network ID start from 0, and the valid range is determined by the number of Ethernet ports on the device. For example, EVO760 supports two Ethernet ports with valid values ranging from 0 to 1: 0 for LANA and 1 for LANB.
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Function block completion flag.
	xBusy	Execution flag	BOOL	TRUE, FALSE	FALSE	Function block execution flag
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Function block fault flag
	dwErrorID	Fault code	DWORD	Depends on data type	0	0: No fault 1: Input parameter fault (network ID configuration error) 2: Network conflict fault 3: Execution fault (execution error)

- **Function**

  This instruction starts the FINS server. The detailed execution is as follows.

  ◦ Before execution, set byEtherID to start the FINS service on the corresponding Ethernet port (0-LAN1/1-LAN2), and determine the range by the maximum number of Ethernet ports supported by the device.

  ◦ Start/stop the FINS service. Set xEnable to TRUE to start, and set xEnable to FALSE to stop.

  ◦ When xBusy is set to TRUE, the FINS server is starting.

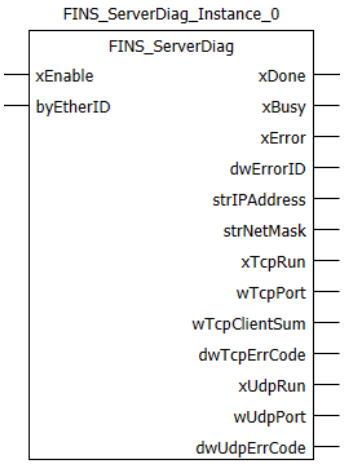
- When xDone is set to TRUE, the FINS server starts successfully.
- When xError is set toTRUE, the FINS server fails to start, and the cause can be identified by dwErrorID.

- **Precautions**
  - This instruction does not support simulation.
  - The FINS server stops upon cold or warm reset, and needs to be enabled again for restart.

18.4.15.3 FINS_ServerDiag

This instruction get the diagnostic information of FINS server.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
FINS_ServerDiag	Get the diagnostic information of FINS server	FB		FINS_ServerDiag_Instance_0(xEnable := , byEtherID := , xDone => , xBusy => , xError => , dwErrorID => , strIPAddress => , strNetMask => , xTcpRun => , wTcpPort => , wTcpClientSum => , dwTcpErrCode => , xUdpRun => , wUdpPort => , dwUdpErrCode =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Function block enable	BOOL	TRUE, FALSE	FALSE	Function block enable flag • When xEnable is set to TRUE, refresh the diagnostic information of FINS server. • When xEnable is set to FALSE, stop reading the diagnostic information of FINS server.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	byEtherID	Ethernet network ID	BYTE	Follows data type	0	The valid values of Ethernet network ID start from 0, and the valid range is determined by the number of Ethernet ports on the device. For example, EVO760 supports two Ethernet ports with valid values ranging from 0 to 1: 0 for LANA and 1 for LANB.
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Function block completion flag.
	xBusy	Execution flag	BOOL	TRUE, FALSE	FALSE	Function block execution flag
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Function block fault flag
	dwErrorID	Fault code	DWORD	Depends on data type	0	0: No fault 1: Input parameter fault (network ID configuration error) 2: Network conflict fault 3: Execution fault (execution error)
	strIPAddress	IP address	STRING(17)	Follow data type	'192.168.1.88'	IP address (FinsServer diagnostic information)
	strNetMask	Subnet mask	STRING(17)	Follow data type	'255.255.255.0'	Subnet mask (FinsServer diagnostic information)
	xTcpRun	TCP operation state	BOOL	TRUE, FALSE	FALSE	TCP operation state (FinsServer diagnostic information)
	wTcpPort	TCP port number	WORD	Follow data type	9600	TCP port number (FinsServer diagnostic information)
	wTcpClientSum	Number of TCP clients	WORD	Follow data type	0	Number of TCP clients (FinsServer diagnostic information)

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	dwTcpErrCode	TCP fault codes	DWORD	Depends on data type	0	TCP fault codes (FinsServer diagnostic information). For details, see TCP/UDP Fault Codes.
	xUdpRun	UDP operation state	BOOL	TRUE, FALSE	FALSE	UDP operation state (FinsServer diagnostic information)
	wUdpPort	UDP port number	WORD	Follow data type	9600	UDP port number (FinsServer diagnostic information)
	dwUdpErrCode	UDP fault codes	DWORD	Depends on data type	0	UDP fault codes (FinsServer diagnostic information). For details, see TCP/UDP Fault Codes.

- **Function**
  - This instruction enables the FINS server diagnostic function. The detailed execution is as follows.
    - Before execution, set byEtherID to start the FINS service diagnosis for the corresponding port.
    - Start/stop the FINS service diagnosis. Set xEnable to TRUE to start, and set xEnable to FALSE to stop.
    - When xBusy is set TRUE, the FINS server diagnosis have been started and continue to run. Thediagnostic code dwTcpErrCode indicates whether the protocol stack execution is abnormal, and the number of connections wTcpClientSum indicates how many clients are accessing.
    - When xError is set toTRUE, the FINS server diagnosis fails to start, and the cause can be identified by dwErrorID.
  - The UDP/TCP fault codes of the FINS_ServerDiag diagnostic function block are shown in the table below.

Fault Code	Description	Remarks
0	No fault	-
1	Incorrect FINS frame header	The FINS TCP data frame header should be 0x46494E53.
2	Incorrect FINS fault code	The fault code of FINS TCP handshake command frame should be 0x00000000.
3	FINS client nodes overlimt	The FINS client nodes cannot exceed 254.
4	FINS handshake failure	The FINS TCP client handshake failed.
5	Incorrect FINS fixed value	The FINS fixed value is filled incorrectly and does not match the FINS command frame.
6	Incorrect type of FINS memory area	The memory area type for FINS read/write command frame is not supported.
7	Incorrect FINS command frame format	The length of FINS command frame is incorrect.

Fault Code	Description	Remarks
8	Number of FINS read/write operations overlimit	The read/write length of FINS command frame exceeds 2000 or is 0.
9	Incorrect FINS access address range	The FINS access address range is incorrect.
10	Incorrect FINS function request code	The FINS function code is not supported.
11	Incorrect FINS frame length	The frame length is incorrect and does not conform to a FINS frame.

- **Precautions**
  - This instruction does not support simulation.
  - The FINS server diagnosis stops upon cold or warm reset, and needs to be enabled again for restart.

18.4.15.4 Example of FINS Communication Instruction Usage

The following is an example of starting the FINS server function.

Variable declarations are as follows:

```
VAR
FINS_Server_Instance_0: FINS_Server;
FINS_ServerDiag_Instance_0: FINS_ServerDiag;
en_fins: BOOL;
en_fins_diag: BOOL;
ether_id: BYTE;
END_VAR
```

The logic implemented in the program editing area is as follows:

```
// Set network port ID
ether_id := 0; // 0 - Start LAN1 port
// Enable the FINS server
en_fins := TRUE;
// Enable the FINS diagnosis
en_fins_diag := TRUE;

FINS_Server_Instance_0(
xEnable := en_fins,
byEtherID := ether_id,
xDone => ,
xBusy => ,
xError => ,
dwErrorID => );

FINS_ServerDiag_Instance_0(
xEnable := en_fins_diag,
byEtherID := ether_id,
xDone => ,
xBusy => ,
xError => ,
dwErrorID => ,
strIPAddress => ,
strNetMask => ,
xTcpRun => ,
wTcpPort => ,
wTcpClientSum => ,
dwTcpErrCode => ,
xUdpRun => ,
wUdpPort => ,
dwUdpErrCode => );
```

18.4.16 Melsec Communication

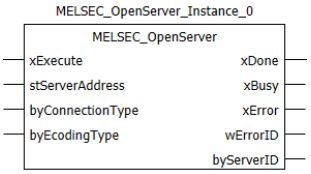
18.4.16.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable
18.4.16.2 MELSEC_OpenServer	FB	Create a Melsec server	MelsecServer	Library version 1.0.0.0 and later
18.4.16.3 MELSEC_CloseServer	FB	Close the Melsec server		
18.4.16.4 MELSEC_ServerDiag	FB	Get the diagnostic information of Melsec server		
18.4.16.5 MELSEC_SetServerSoftAddr	FB	Set the device address		

18.4.16.2 MELSEC_OpenServer

This instruction enables the MC (MELSEC) slave.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
MELSEC_OpenServer	Create a Melsec server	FB		MELSEC_OpenServer_Instance_0(xExecute := , stServerAddress := , byConnectionType := , byEcodingType := , xDone => , xBusy => , xError => , wErrorID => , byServerID =>);

- **Related variables**

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Instruction Enable	BOOL	TRUE, FALSE	FALSE	Triggered execution.
	stServerAddress	Server address	ADDRESS	Follows data type	-	Server address, including IP and port number

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	byConnectionType	Connection type	BYTE	0 and 1	0	Connection type 0: TCP 1: UDP
	byEcodingType	Encoding type	BYTE	0 and 1	0	Encoding type 0: binary 1: ASCII
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Execution completed
	xBusy	Execution flag	BOOL	TRUE, FALSE	FALSE	Executing
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Execution error
	wErrorID	Fault code	WORD	Follow data type	0	See 18.4.16.6 Fault Codes.
	byServerID	Server ID	BYTE	Follows data type	0	Server ID

- **Function**
  - This instruction starts the MC slave when xExecute is set to TRUE. The detailed execution is as follows.
    - Before execution, set the local server address via stServerAddress, and specify the connection type and encoder type via byConnectionType and byEcodingType. When xExecute is set toTRUE, the instruction starts execution.
    - When xDone is set to TRUE, the instruction is successfully executed, and the generated byServerID can be used with MELSEC_CloseServer and MELSEC_ServerDiag.
    - When xBusy is set to TRUE, it indicates that the instruction is executing during current scan cycle.
    - If xError is set to TRUE, it indicates that an error occurred during instruction execution. The cause can be identified by wErrorID.
  - The data type of stServerAddress is ADDRESS. The descriptions of structure members are in the table below.

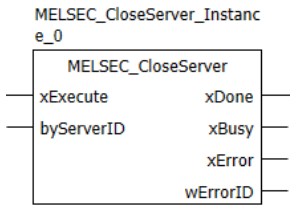
Name	Data Type	Description
strIPAddr	STRING	IP address
uiPort	UINT	Port number with rangeof 1025 to 65535

- **Precautions**
  - If the input port number is within the specified range of 1025 to 65535 yet an error still occurs, the port may already be used by another function.
  - The MC slave automatically shuts down upon cold/hot reset or RUN/STOP mode switching and needs to be re-enabled.
  - This instruction does not support simulation.

18.4.16.3 MELSEC_CloseServer

This instruction closes the Melsec server.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
MELSEC_CloseServer	Close the Melsec server	FB		MELSEC_CloseServer_Instance_0(xExecute := , byServerID := , xDone => , xBusy => , xError => , wErrorID =>);

- **Related variables**

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Instruction enable	BOOL	TRUE, FALSE	FALSE	Triggered execution.
	byServerID	Server ID	BYTE	-	-	Server ID
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Execution completed
	xBusy	Execution flag	BOOL	TRUE, FALSE	FALSE	Executing
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Execution error
	wErrorID	Fault code	WORD	Follow data type	0	See 18.4.16.6 Fault Codes

- **Function**

  This instruction closes the MC slave when xExecute is set to TRUE. The detailed execution is as follows.

  - Before execution, input the byServerID (generated by MELSEC_OpenServer). When xExecute is set to TRUE, the instruction starts execution.
  - When xDone is set to TRUE, the instruction is successfully executed.
  - When xBusy is set to TRUE, the instruction is being executed in the current scan cycle.
  - If xError is set to TRUE, it indicates that an error occurred during instruction execution. The cause can be identified by wErrorID.

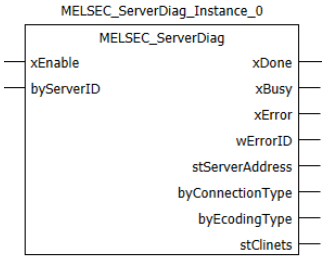
- **Precautions**

  This instruction does not support simulation.

18.4.16.4 MELSEC_ServerDiag

This instruction gets the diagnostic information of Melsec server.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
MELSEC_ServerDiag	Get the diagnostic information of Melsec server	FB		MELSEC_ServerDiag_Instance_0(xEnable := , byServerID := , xDone => , xBusy => , xError => , wErrorID => , stServerAddress => , byConnectionType => , byEcodingType => , stClinets =>);

- **Related variables**

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xEnable	Instruction enable	BOOL	TRUE, FALSE	FALSE	Triggered execution.
	byServerID	Server ID	BYTE	-	-	Server ID
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Execution completed
	xBusy	Execution flag	BOOL	TRUE, FALSE	FALSE	Executing
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Execution error
	wErrorID	Fault code	WORD	Follow data type	0	See 18.4.16.6 Fault Codes.
	stServerAddress	Server IP address	ADDRESS	Follows data type	-	Server address
	byConnectionType	Connection type	BYTE	0 and 1	0	Connection type 0: TCP 1: UDP
	byEcodingType	Encoding type	BYTE	0 and 1	0	Encoding type 0: binary 1: ASCII

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	stClinets	Client information	CLIENT_INFO	Follows data type	-	Client information It takes effect only under TCP connection type (byConnectionType set to 0).

- **Function**
  - This instruction gets the diagnostic information of MC slave when xEnable is set to TRUE. The detailed execution is as follows.
    - Before execution, input the byServerID (generated by MELSEC_OpenServer). When xEnable is set to TRUE, the instruction starts execution.
    - When xDone is set to TRUE, the instruction is successfully executed.
    - When xBusy is set to TRUE, the instruction is being executed in the current scan cycle, and the real-time information of clients connected to the server is output.
    - If xError is set to TRUE, it indicates that an error occurred during instruction execution. The cause can be identified by wErrorID.
  - The data type of stServerAddress is ADDRESS. For details, see ADDRESS.
  - The data type of stClinets is CLIENT_INFO. The descriptions of structure members are in the table below.

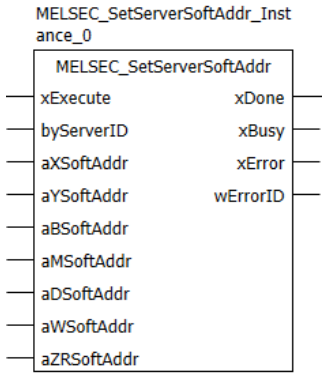
Name	Data Type	Description
byConnectedCount	BYTE	Number of connections
ConnectAddress	ARRAY [1..32] OF ADDRESS	Connection address

- **Precautions**
  This instruction does not support simulation.

18.4.16.5 MELSEC_SetServerSoftAddr

This instruction sets the device address of MC slave.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
MELSEC_SetServerSoftAddr	Set the device address	FB		MELSEC_SetServerSoftAddr_Instance_0(xExecute := , byServerID := , aXSoftAddr := , aYSoftAddr := , aBSoftAddr := , aMSoftAddr := , aDSoftAddr := , aWSoftAddr := , aZRSoftAddr := , xDone => , xBusy => , xError => , wErrorID =>);

- **Related variables**

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Instruction enable	BOOL	TRUE, FALSE	FALSE	Triggered execution.
	byServerID	Server ID	BYTE	Follows data type	0	Server ID
	aXSoftAddr	Address of X device	ANY	Follows data type	-	Address of X device
	aYSoftAddr	Address of Y device	ANY	Follows data type	-	Address of Y device
	aBSoftAddr	Address of B device	ANY	Follows data type	-	Address of B device
	aMSoftAddr	Address of M device	ANY	Follows data type	-	Address of M device
	aDSoftAddr	Address of D device	ANY	Follows data type	-	Address of D device
	aWSoftAddr	Address of W device	ANY	Follows data type	-	Address of W device
	aZRSoftAddr	Address of Z device	ANY	Follows data type	-	Address of Z device
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Execution completed
	xBusy	Execution flag	BOOL	TRUE, FALSE	FALSE	Executing
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Execution error

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	wErrorID	Fault code	WORD	Follow data type	0	See 18.4.16.6 Fault Codes.

- **Function**
  - This instruction sets the device addresses of the MC slave, including X/Y/B/M/D/W/ZR devices. The device types are listed in the table below.

Device	Type
X/Y/B/M	Bit device
D/W/ZR	Word device

  - Default access range of devices
    For medium-sized PLCs, the M, Q, and I areas support bit, half-word, and word addressing, with Q and I areas allocated uniformly by iFA. Since the MC slave is configured via this instruction, a mapping relationship must be established between the M area and Mitsubishi devices, as shown in the table below (direct access to Q/I areas is unavailable).

Inovance Device	Type	Range	Number of Points	Mitsubishi Deivice
% MX	Bit	%MX0.0 to %MX127.7	1024	X
% MX	Bit	%MX200.0 to %MX327.7	1024	Y
% MX	Bit	%MX400.0 to %MX1399.7	8000	M
% MX	Bit	%MX1500.0 to %MX5595.7	32768	B
% MW	Word	%MW3000 to %MW10999	8000	D
% MW	Word	%MW12000 to %MW44767	32768	W
% MW	Word	%MW45000 to %MW94999	50000 points	ZR

  - Custom device address
    The mapping rules between custom arrays and devices are shown in the table below.

Device	Memory Size in Byte	Custom BOOL Array	Custom BYTE Array	Custom WORD Array	Device Address
X/Y/B/M	1	0	0	0	0
	1	1	1		1
	1	2	2	1	2
	1	3	3		3
D/W/ZR	1	0	0	0	0
	1	1	1		
	1	2	2	1	1
	1	3	3		
	1	4	4	2	2
	1	5	5		
	1	6	6	3	3

Device	Memory Size in Byte	Custom BOOL Array	Custom BYTE Array	Custom WORD Array	Device Address
	1	7	7		

◦ Before execution, ensure the input parameters are correct, as shown in the table below.

Input Variable	Description
byServerID	The server ID is generated when the MC protocol is started by MELSEC_OpenServer. Input the server ID corresponding to the server you wish to use.
aXSoftAddr	X device address supports only one-dimensional arrays. You can input either the array name or the array name with an index (subscript). Since the X device is a bit device, it is recommended to input a one-dimensional BOOL array.
Other device addresses	For details, see aXSoftAddr. It is recommended to input a one-dimensional array matching the device type. For example, input a one-dimensional BOOL array for bit devices and a one-dimensional WORD array for word devices.

◦ During execution, xBusy is to TRUE and remains at high level.

◦ When the execution is completed and the configuration is successful, xDone is set to TRUE.

◦ If the execution and configuration fail, xError is set to TRUE. The cause can be identified by wErrorID.

- **Precautions**
  ◦ A cold/warm reset or RUN/STOP mode switch will restore the default device address.
  ◦ When the device address changes, the instruction must be executed again to update the address.
  ◦ This instruction does not support simulation.

18.4.16.6 Fault Codes

Fault code	Value	Description
MELSEC_ERR_OK	0	No fault
MELSEC_ERR_FAILED	1	Internal error
MELSEC_ERR_SERVER_ENALBED	2	MC service is enabled.
MELSEC_ERR_SERVER DISALBED	3	MC service is disabled.
MELSEC_ERR_SOFT_MAP_ADDR	4	Incorrect device mapping address
MELSEC_ERR_IPADDR	5	Incorrect IP address
MELSEC_ERR_CONNECT_TYPE	6	Incorrect connection type
MELSEC_ERR_PORT	7	Incorrect port number
MELSEC_ERR_ECODING_TYPE	8	Incorrect encoding type
MELSEC_ERR_SERVER_ID	9	Incorrect server ID
MELSEC_ERR_SERVER_START	10	Failed to start the service
MELSEC_ERR_SEVER_SOCKET	11	Service socket error
MELSEC_ERR_SEVER_BIND	12	Failed to bind the server

Fault code	Value	Description
MELSEC_ERR_SEVER_LISTEN	13	Failed to listen the server
MELSEC_ERR_SEVER_STATE	14	Abnormal service status
MELSEC_ERR_SEVER_EXECUTING	15	Operation in progress
MELSEC_ERR_MAP_SOFT_X	32	Failed to map X device
MELSEC_ERR_MAP_SOFT_Y	33	Failed to map Y device
MELSEC_ERR_MAP_SOFT_B	34	Failed to map B device
MELSEC_ERR_MAP_SOFT_M	35	Failed to map M device
MELSEC_ERR_MAP_SOFT_D	36	Failed to map D device
MELSEC_ERR_MAP_SOFT_W	37	Failed to map W device
MELSEC_ERR_MAP_SOFT_ZR	38	Failed to map ZR device

18.4.16.7 Example of Melsec Communication Instruction Usage

This is an example of starting the MC slave, getting diagnostic information, and then shutting down the MC slave.

Variable declarations are as follows:

```
VAR
MELSEC_OpenServer_0: MELSEC_OpenServer;
MELSEC_ServerDiag_0: MELSEC_ServerDiag;
MELSEC_CloseServer_0: MELSEC_CloseServer;
stServerAddress: MelsecServer.ADDRESS;
byConnectionType: BYTE;
byEcodingType: BYTE;
byServerID: BYTE;
xEnOpenServer: BOOL;
xEnServerDiag: BOOL;
xEnCloseServer: BOOL;
END_VAR
```

The logic implemented in the program editing area is as follows:

```
// Set the listening address of the local server
stServerAddress.strIPAddr:='192.168.1.88';
stServerAddress.uiPort:=8080;
// Set the communication type: 0 TCP, 1 UDP
byConnectionType:=0;
// MELSEC communication encoding type: 0 binary, 1 ASCII
byEcodingType:=0;

MELSEC_OpenServer_0(
xExecute:= xEnOpenServer,
stServerAddress:= stServerAddress,
byConnectionType:= byConnectionType,
byEcodingType:= byEcodingType,
xDone=> ,
xBusy=> ,
xError=> ,
wErrorID=> ,
byServerID=> byServerID);
IF MELSEC_OpenServer_0.xError THEN;// Add error handling
END_IF

MELSEC_ServerDiag_0(
xEnable:= xEnServerDiag,
byServerID:= byServerID,
xDone=> ,
xBusy=> ,
xError=> ,
wErrorID=> ,
stServerAddress=> ,
byConnectionType=> ,
byEcodingType=> ,
stClinets=> );
IF MELSEC_ServerDiag_0.xError THEN;// Error handling
END_IF

MELSEC_CloseServer_0(
```

```
xExecute:= xEnCloseServer,
byServerID:= byServerID,
xDone=> ,
xBusy=> ,
xError=> ,
wErrorID=> );
IF MELSEC_CloseServer_0.xError THEN;// Error handling
END_IF
```

18.4.17 SFTP Communication

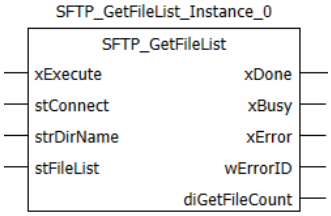
18.4.17.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable
18.4.17.2 SFTP_GetFileList	FB	Get file list from SFTP server	SftpClient	Library version 1.0.0.0 and later
18.4.17.3 SFTP_GetFile	FB	Get file from SFTP server		
18.4.17.4 SFTP_PutFile	FB	Upload local file to SFTP server		
18.4.17.5 SFTP_DeleteFile	FB	Delete file on SFTP server		
18.4.17.6 SFTP_MoveFile	FB	Move file on SFTP server		
18.4.17.7 SFTP_RenameFile	FB	Rename file on SFTP server		
18.4.17.8 SFTP_CreateDir	FB	Create directory on SFTP server		

18.4.17.2 SFTP_GetFileList

This instruction gets all files or directories from the specified server path.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SFTP_GetFileList	Get file list from SFTP server	FB		SFTP_GetFileList_Instance_0(xExecute := , stConnect := , strDirName := , stFileList := , xDone => , xBusy => , xError => , wErrorID => , diGetFileCount =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Instruction enable	BOOL	TRUE, FALSE	FALSE	Triggered execution.
	stConnect	Connection information	SFTP_CONNECT_INFO	Follows data type	-	Connection information, including the server address and port.
	strDirName	Server directory name (including the absolute path)	STRING	Follows data type	-	Server directory name (including the absolute path)
	stFileList	File list	ANY	Follows data type	-	File list: one-dimensional array of SFTP_FILE_INFO
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Execution completed
	xBusy	Execution flag	BOOL	TRUE, FALSE	FALSE	Executing
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Execution error
	wErrorID	Fault code	WORD	Follow data type	0	See 18.4.17.9 Fault Codes.
	diGetFileCount	Number of files obtained	DINT	Follows data type	0	Number of files obtained

- **Function**
  - The SFTP_GetFileList instruction is used to download all the files or directories from the specified SFTP server path. It is triggered for one-time execution by xExecute. It establishes a connection with the serverbased on the information specified by stConnect, obtains all files from the server directory specified by strDirName, and stores the data into stFileList. If the execution is successful, xDone is set

to TRUE and the actual number of files obtained specified by diGetFileCount is output. If the execution fails, xError is set to TRUE and wErrorID is output.

◦ The data type of stConnect is SFTP_CONNECT_INFO. The descriptions of structure members are in the table below.

Name	Data Type	Description
strServerIPAddr	STRING	Server address
uiServerPort	UINT	Server port
strUsername	STRING	Login account
strPassword	STRING	Login password

◦ The data type of stFileList SFTP_FILE_INFO. The descriptions of structure members are in the table below.

Name	Data Type	Description
strName	STRING	File name
diaDate	ARRAY [0..5] OF DINT	File update time: year\month\day\hour\minute\second
uliSize	ULINT	File size
xReadOnly	BOOL	File Read-only flag
xFileType	BOOL	File type flag

- **Program example**

The variable declarations are as follows.

```
VAR
SFTP_GetFileList_0: SFTP_GetFileList;
stConnect: SftpClient.SFTP_CONNECT_INFO;
strDirName: STRING;
stFileList:ARRAY[0..10] OF SFTP_FILE_INFO;
en_get_file_list:BOOL;
END_VAR
```

The program editing area implements the logic as follows:

```
// SFTP server connection settings
stConnect.strServerIPAddr := '192.168.1.88';// SFTP server IP address
stConnect.uiServerPort := 22;// SFTP server port number
stConnect.strUsername := 'username';// Username
stConnect.strPassword := 'password';// Password

strDirName:='/DirName';// Server directory, including the path.

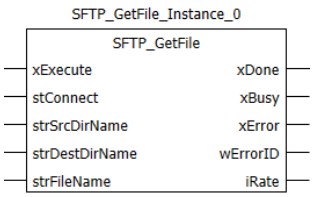
SFTP_GetFileList_0(
xExecute:= en_get_file_list,
stConnect:= stConnect,
strDirName:= strDirName,
stFileList:= stFileList,
xDone=> ,
xBusy=> ,
xError=> ,
wErrorID=> ,
diGetFileCount=> );
IF SFTP_GetFileList_0.xDone THEN
;// Execution completed
ELSIF SFTP_GetFileList_0.xError THEN
;// Error handling
ELSIF SFTP_GetFileList_0.xBusy THEN
;// Executing
ELSE
;// Not executed
END_IF
```

- **Precautions**
  This instruction does not support simulation.

18.4.17.3 SFTP_GetFile

This instruction get server files to the local device.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SFTP_GetFile	Get file from SFTP server	FB		SFTP_GetFile_Instance_0(xExecute := , stConnect := , strSrcDirName := , strDestDirName := , strFileName := , xDone => , xBusy => , xError => , wErrorID => , iRate =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Instruction enable	BOOL	TRUE, FALSE	FALSE	Triggered execution.
	stConnect	Connection information	SFTP_CONNECT_INFO	Follows data type	-	Connection information, including the server address and port.
	strSrcDirName	Server directory name (including the absolute path)	STRING	Follows data type	-	Server directory name (including the absolute path)
	strDestDirName	Local directory name	STRING	Follows data type	-	Local directory name. For details, see the description of the local directory in the 18.2.11.2 File Operations > Overview section.
	strFileName	Name of the file to be obtained	STRING	Follows data type	-	Name of the file to be obatained
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Execution completed
	xBusy	Execution flag	BOOL	TRUE, FALSE	FALSE	Executing

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Execution error
	wErrorID	Fault code	WORD	Follow data type	0	See 18.4.17.9 Fault Codes.
	iRate	Download progress %	INT	Follows data type	-	Download progress %

- **Function**
  - The SFTP_GetFile instruction is used to obtain files from the SFTP server. This instruction is executed once when the xExecute signal is triggered. It establishes a connection with the server based on the information specified by strSrcDirName. After that, it downloads a file strFileName from the server directory strServerDirName to the local directory strDestDirName. If the execution is successful, xDone is set to TRUE. If the execution fails, xError is set to TRUE, and wErrorID is output.
  - The data type of stConnect is SFTP_CONNECT_INFO. For details, see SFTP_CONNECT_INFO.
- **Program example**

  The variable declarations are as follows.

```
VAR
SFTP_GetFile_0: SFTP_GetFile;
stConnect: SftpClient.SFTP_CONNECT_INFO;
strSrcDirName: STRING;
strDestDirName: STRING;
strFileName: STRING;
en_get_file: BOOL;
END_VAR
```

  The program editing area implements the logic as follows:

```
// SFTP server connection settings
stConnect.strServerIPAddr := '192.168.1.88';// SFTP server IP address
stConnect.uiServerPort := 22;// SFTP server port number
stConnect.strUsername := 'username';// Username
stConnect.strPassword := 'password';// Password

strSrcDirName:='/SrcDirName';// Source directory name for download (server directory), including the path. The leading '/' indicates the top-level directory of the server, also known as the root directory. To access a subdirectory, use '/subdirectory_name/'.
strDestDirName:='/DestDirName';// Target directory name for download (local directory), including the path.
strFileName:='filename'; // Name of the file to be downloaded

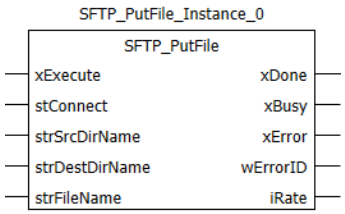
SFTP_GetFile_0(
xExecute:= en_get_file,
stConnect:= stConnect,
strSrcDirName:= strSrcDirName,
strDestDirName:= strDestDirName,
strFileName:= strFileName,
xDone=> ,
xBusy=> ,
xError=> ,
wErrorID=> ,
iRate=> );
IF SFTP_GetFile_0.xDone THEN
;// Execution completed
ELSIF SFTP_GetFile_0.xError THEN
;// Error handling
ELSIF SFTP_GetFile_0.xBusy THEN
;// Executing
ELSE
;// Not executed
END_IF
```

- **Precautions**
  This instruction does not support simulation.

18.4.17.4 SFTP_PutFile

This instruction upload local files to the server.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SFTP_PutFile	Upload local file to SFTP server	FB		SFTP_PutFile_Instance_0(xExecute := , stConnect := , strSrcDirName := , strDestDirName := , strFileName := , xDone => , xBusy => , xError => , wErrorID => , iRate =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Instruction enable	BOOL	TRUE, FALSE	FALSE	Triggered execution.
	stConnect	Connection information	SFTP_CONNECT_INFO	Follows data type	-	Connection information, including the server address and port.
	strSrcDirName	Local directory name	STRING	Follows data type	-	Local directory name. For details, see the description of the local directory in the 18.2.11.2 File Operations > Overview section.
	strDestDirName	Server directory name (including the absolute path)	STRING	Follows data type	-	Server directory name (including the absolute path)
	strFileName	Name of the file to be uploaded	STRING	Follows data type	-	Name of the file to be uploaded

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Execution completed
	xBusy	Execution flag	BOOL	TRUE, FALSE	FALSE	Executing
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Execution error
	wErrorID	Fault code	WORD	Follow data type	0	See 18.4.17.9 Fault Codes.
	iRate	Upload progress %	INT	Follows data type	-	Download progress %

- **Function**
  - The SFTP_PutFile instruction is used to upload a local file to the SFTP server. This instruction is executed once when the xExecute signal is triggered. It establishes a connection with the server based on the information specified by strSrcDirName. After that, it uploads a file strFileName from the local directory strSrcDirName to the server directory strDestDirName. If the execution is successful, xDone is set to TRUE. If the execution fails, xError is set to TRUE, and wErrorID is output.
  - The data type of stConnect is SFTP_CONNECT_INFO. For details, see SFTP_CONNECT_INFO.
- **Program example**

  The variable declarations are as follows.

```
VAR
SFTP_PutFile_0: SFTP_PutFile;
stConnect: SftpClient.SFTP_CONNECT_INFO;
strSrcDirName: STRING;
strDestDirName: STRING;
strFileName: STRING;
en_put_file: BOOL;
END_VAR
```

  The program editing area implements the logic as follows:

```
// SFTP server connection settings
stConnect.strServerIPAddr := '192.168.1.88';// SFTP server IP address
stConnect.uiServerPort := 22;// SFTP server port number
stConnect.strUsername := 'username';// Username
stConnect.strPassword := 'password';// Password

strSrcDirName:='/SrcDirName';// Source directory name for upload (local directory), including the path.
strDestDirName:='/DestDirName';// Target directory name for download (server directory), including the path.
strFileName:='file';// Name of the file to be uploaded

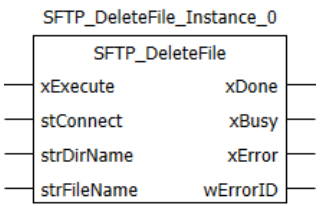
SFTP_PutFile_0(
xExecute:= en_put_file,
stConnect:= stConnect,
strSrcDirName:= strSrcDirName,
strDestDirName:= strDestDirName,
strFileName:= strFileName,
xDone=> ,
xBusy=> ,
xError=> ,
wErrorID=> ,
iRate=> );
IF SFTP_PutFile_0.xDone THEN
;// Execution completed
ELSIF SFTP_PutFile_0.xError THEN
;// Error handling
ELSIF SFTP_PutFile_0.xBusy THEN
;// Executing
ELSE
;// Not executed
END_IF
```

- **Precautions**
  This instruction does not support simulation.

18.4.17.5 SFTP_DeleteFile

This instruction deletes the file or directory at the specified path on the server.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SFTP_DeleteFile	Delete file on SFTP server	FB		SFTP_DeleteFile_Instance_0(xExecute := , stConnect := , strDirName := , strFileName := , xDone => , xBusy => , xError => , wErrorID =>);

- **Related variables**

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Instruction enable	BOOL	TRUE, FALSE	FALSE	Triggered execution.
	stConnect	Connection information	SFTP_CONNECT_INFO	Follows data type	-	Connection information, including the server address and port.
	strDirName	Server directory name (including the absolute path)	STRING	Follows data type	-	Server directory name (including the absolute path)
	strFileName	Name of the file or directory to be deleted	STRING	Follows data type	-	Name of the file or directory to be deleted
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Execution completed
	xBusy	Execution flag	BOOL	TRUE, FALSE	FALSE	Executing
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Execution error

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	wErrorID	Fault code	WORD	Follow data type	0	See 18.4.17.9 Fault Codes.

- **Function**
  - The SFTP_DeleteFile instruction is used to delete files or directories at the specified path on the server. This instruction is executed once when the xExecute signal is triggered. It establishes a connection with the server based on the information specified by stConnect. After that, it deletes a file or directory strFileName on the server directory strDirName. If the execution is successful, xDone is set to TRUE. If the execution fails, xError is set to TRUE, and wErrorID is output.
  - The data type of stConnect is SFTP_CONNECT_INFO. For details, see SFTP_CONNECT_INFO.
- **Program example**

  The variable declarations are as follows.

```
VAR
SFTP_DeleteFile_0: SFTP_DeleteFile;
stConnect: SftpClient.SFTP_CONNECT_INFO;
strDirName: STRING;
strFileName: STRING;
en_delete_file: BOOL;
END_VAR
```

  The program editing area implements the logic as follows:

```
// SFTP server connection settings
stConnect.strServerIPAddr := '192.168.1.88';// SFTP server IP address
stConnect.uiServerPort := 22;// SFTP server port number
stConnect.strUsername := 'username';// Username
stConnect.strPassword := 'password';// Password

strDirName:='/DirName';// Directory name (including path)
strFileName:='filename';// Name of the file to be deleted

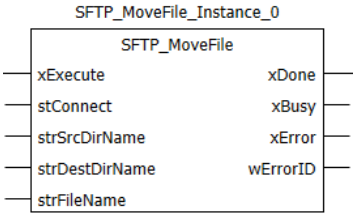
SFTP_DeleteFile_0(
xExecute:= en_delete_file,
stConnect:= stConnect,
strDirName:= strDirName,
strFileName:= strFileName,
xDone=> ,
xBusy=> ,
xError=> ,
wErrorID=> );
IF SFTP_DeleteFile_0.xDone THEN
;// Execution completed
ELSIF SFTP_DeleteFile_0.xError THEN
;// Error handling
ELSIF SFTP_DeleteFile_0.xBusy THEN
;// Executing
1. ELSE
;// Not executed
END_IF
```

- **Precautions**
  This instruction does not support simulation.

18.4.17.6 SFTP_MoveFile

This instruction moves files or directories on the server to the specified directory.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SFTP_MoveFile	Move file on SFTP server	FB		SFTP_MoveFile_Instance_0(xExecute := , stConnect := , strSrcDirName := , strDestDirName := , strFileName := , xDone => , xBusy => , xError => , wErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Instruction enable	BOOL	TRUE, FALSE	FALSE	Triggered execution.
	stConnect	Connection information	SFTP_CONNECT_INFO	Follows data type	-	Connection information, including the server address and port .
	strSrcDirName	Source directory name (including absolute path)	STRING	Follows data type	-	Source directory name (including absolute path)
	strDestDirName	Destination directory name (including absolute path)	STRING	Follows data type	-	Destination directory name (including absolute path)
	strFileName	Name of the file or directory to be moved	STRING	Follows data type	-	Name of the file or directory to be moved
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Execution completed
	xBusy	Execution flag	BOOL	TRUE, FALSE	FALSE	Executing
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Execution error

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	wErrorID	Fault code	WORD	Follow data type	0	See 18.4.17.9 Fault Codes.

- **Function**
  - The SFTP_MoveFile instruction is used to move a file or directory on the SFTP server to the designated directory. This instruction is executed once when the xExecute signal is triggered. It establishes a connection with the server based on the information specified by stConnect. After that, it moves a file or directory strFileName from the source server directory strSrcDirName to the specified directory strDestDirName. If the execution is successful, xDone is set to TRUE. If the execution fails, xError is set to TRUE, and wErrorID is output.
  - The data type of stConnect is SFTP_CONNECT_INFO. For details, see SFTP_CONNECT_INFO.
- **Program example**

  The variable declarations are as follows.

```
VAR
SFTP_MoveFile_0: SFTP_MoveFile;
stConnect: SftpClient.SFTP_CONNECT_INFO;
strSrcDirName: STRING;
strDestDirName: STRING;
strFileName: STRING;
en_move_file: BOOL;
END_VAR
```

  The program editing area implements the logic as follows:

```
// SFTP server connection settings
stConnect.strServerIPAddr := '192.168.1.88';// SFTP server IP address
stConnect.uiServerPort := 22;// SFTP server port number
stConnect.strUsername := 'username';// Username
stConnect.strPassword := 'password';// Password

strSrcDirName:='/SrcDirName';// Source directory (server directory), including path
strDestDirName:='/DestDirName';// Target directory (server directory), including path
strFileName:='filename';// Name of the file to be moved

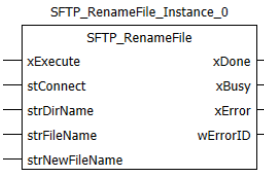
SFTP_MoveFile_0(
xExecute:= en_move_file,
stConnect:= stConnect,
strSrcDirName:= strSrcDirName,
strDestDirName:= strDestDirName,
strFileName:= strFileName,
xDone=> ,
xBusy=> ,
xError=> ,
wErrorID=> );
IF SFTP_MoveFile_0.xDone THEN
;// Execution completed
ELSIF SFTP_MoveFile_0.xError THEN
;// Error handling
ELSIF SFTP_MoveFile_0.xBusy THEN
;// Executing
ELSE
;// Not executed
END_IF
```

- **Precautions**
  This instruction does not support simulation.

18.4.17.7 SFTP_RenameFile

This instruction renames the file or directory at the specified path on the server.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SFTP_RenameFile	Rename file on SFTP server	FB		SFTP_RenameFile_Instance_0(xExecute := , stConnect := , strDirName := , strFileName := , strNewFileName := , xDone => , xBusy => , xError => , wErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Instruction enable	BOOL	TRUE, FALSE	FALSE	Triggered execution.
	stConnect	Connection information	SFTP_CONNECT_INFO	Follows data type	-	Connection information, including the server address and port.
	strDirName	Server directory name (including the absolute path)	STRING	Follows data type	-	Server directory name (including the absolute path)
	strFileName	Original name of the file or directory	STRING	Follows data type	-	Original name of the file or directory
	strNewFileName	New name of the file or directory	STRING	Follows data type	-	New name of the file or directory
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Execution completed
	xBusy	Execution flag	BOOL	TRUE, FALSE	FALSE	Executing
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Execution error
	wErrorID	Fault code	WORD	Follow data type	0	See 18.4.17.9 Fault Codes.

- **Function**
  - The SFTP_RenameFile instruction is used to rename files or directories at the specified path on the server. This instruction is executed onece when the xExecute signal is triggered. It establishes a connection with the server based on the information specified by stConnect. After that, it changes the name of a file or directory on the server directory strDirName from strFileName to strNewFileName. If

the execution is successful, xDone is set to TRUE. If the execution fails, xError is set to TRUE, and wErrorID is output.

- The data type of stConnect is SFTP_CONNECT_INFO. For details, see SFTP_CONNECT_INFO.

- **Program example**

  The variable declarations are as follows.

```
VAR
SFTP_RenameFile_0: SFTP_RenameFile;
stConnect: SftpClient.SFTP_CONNECT_INFO;
strDirName: STRING;
strFileName: STRING;
strNewFileName: STRING;
en_rename_file: BOOL;
END_VAR
```

  The program editing area implements the logic as follows:

```
// SFTP server connection settings
stConnect.strServerIPAddr := '192.168.1.88';// SFTP server IP address
stConnect.uiServerPort := 22;// SFTP server port number
stConnect.strUsername := 'username';// Username
stConnect.strPassword := 'password';// Password

strDirName:='/DirName';// Directory name (server directory), including the path.
strFileName:='filename';// Original name of the file
strNewFileName:='newfilename';// New name of the file

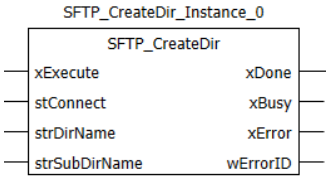
SFTP_RenameFile_0(
xExecute:= en_rename_file,
stConnect:= stConnect,
strDirName:= strDirName,
strFileName:= strFileName,
strNewFileName:= strNewFileName,
xDone=> ,
xBusy=> ,
xError=> ,
wErrorID=> );
IF SFTP_RenameFile_0.xDone THEN
;// Execution completed
ELSIF SFTP_RenameFile_0.xError THEN
;// Error handling
ELSIF SFTP_RenameFile_0.xBusy THEN
;// Executing
ELSE
;// Not executed
END_IF
```

- **Precautions**
  This instruction does not support simulation.

18.4.17.8 SFTP_CreateDir

This instruction creates a new directory at the specified server path.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SFTP_CreateDir	Create directory on SFTP server	FB		SFTP_CreateDir_Instance_0(xExecute := , stConnect := , strDirName := , strSubDirName := , xDone => , xBusy => , xError => , wErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	xExecute	Instruction enable	BOOL	TRUE, FALSE	FALSE	Triggered execution.
	stConnect	Connection information	SFTP_CONNECT_INFO	Follows data type	-	Connection information, including the server address and port.
	strDirName	Server directory name (including the absolute path)	STRING	Follows data type	-	Server directory name (including the absolute path)
	strSubDirName	Name of the subdirectory to create	STRING	Follows data type	-	Name of the subdirectory to create
Output	xDone	Execution completion flag	BOOL	TRUE, FALSE	FALSE	Execution completed
	xBusy	Execution flag	BOOL	TRUE, FALSE	FALSE	Executing
	xError	Error flag	BOOL	TRUE, FALSE	FALSE	Execution error
	wErrorID	Fault code	WORD	Follow data type	0	See 18.4.17.9 Fault Codes.

- **Function**
  - The SFTP_CreateDir instruction is used to create a directory at the specified server path. This instruction is executed once when the xExecute signal is triggered. It establishes a connection with the server based on the information specified by stConnect. After that, it creates a new directory strSubDirName in the server directory strDirName. If the execution is successful, xDone is set to TRUE. If the execution fails, xError is set to TRUE, and wErrorID is output.

- The data type of stConnect is SFTP_CONNECT_INFO. For details, see SFTP_CONNECT_INFO.

- **Program example**

  The variable declarations are as follows.

```
VAR
SFTP_CreateDir_0: SFTP_CreateDir;
stConnect: SftpClient.SFTP_CONNECT_INFO;
strDirName: STRING;
strSubDirName: STRING;
en_create_dir: BOOL;
END_VAR
```

  The program editing area implements the logic as follows:

```
// SFTP server connection settings
stConnect.strServerIPAddr := '192.168.1.88';// SFTP server IP address
stConnect.uiServerPort := 22;// SFTP server port number
stConnect.strUsername := 'username';// Username
stConnect.strPassword := 'password';// Password


strDirName:='/DirName';// Directory name (server directory), including the path.
strSubDirName:='SubDirName';//Name of the subdirectory to create
SFTP_CreateDir_0(
xExecute:= en_create_dir,
stConnect:= stConnect,
strDirName:= strDirName,
strSubDirName:= strSubDirName,
xDone=> ,
xBusy=> ,
xError=> ,
wErrorID=> );
IF SFTP_CreateDir_0.xDone THEN
;// Execution completed
ELSIF SFTP_CreateDir_0.xError THEN
;// Error handling
ELSIF SFTP_CreateDir_0.xBusy THEN
;// Executing
ELSE
;// Not executed
END_IF
```

- **Precautions**
  This instruction does not support simulation.

18.4.17.9 Fault Codes

Fault code	Value	Description
SFTP_ERROR_OK	0	Normal
SFTP_ERROR_FAILED	1	Internal error
SFTP_ERROR_CLIENT_DISABLE	2	Client is not started
SFTP_ERROR_NO_SDCARD	3	SD card is not inserted (reserved, SD card detection not supported temporarily)

Fault code	Value	Description
SFTP_ERROR_SDCARD_SIZE	4	Insufficient available space on SD card (reserved, SD card detection not supported temporarily)
SFTP_ERROR_UNSUPPORTED_DIR	5	Unsupported directory
SFTP_ERROR_PARA_IP	20	Incorrect IP address
SFTP_ERROR_PARA_PORT	21	Incorrect port number
SFTP_ERROR_PARA_USERNAME	22	Invalid account
SFTP_ERROR_PARA_PASSWORD	23	Incorrect password
SFTP_ERROR_PARA_PARA_PATH	40	Invalid directory name
SFTP_ERROR_PARA_FILENAME	41	Invalid file name
SFTP_ERROR_PARA_MAX_LEN	42	Path length exceeds the upper limit
SFTP_ERROR_FILE_LIST_TYPE	60	Incorrect type of file information list
SFTP_ERROR_FILE_LIST_DIME	61	Incorrect dimension of file information list
SFTP_ERROR_SESSION	1002	Failed to create a session
SFTP_ERROR_TIMEOUT	1003	Connection timeout
SFTP_ERROR_SOCK	1004	Connection error
SFTP_ERROR_PROTOCOL_UNMATCH	1005	Protocol mismatch
SFTP_ERROR_PASSWORD_NONSUPPORT	1006	Password authentication is not supported.
SFTP_ERROR_AUTH	1007	Authentication failed
SFTP_ERROR_PASSWORD_MEM	1008	Failed to allocate memory
SFTP_ERROR_SOCKET	1009	Failed to initialize socket
SFTP_ERROR_LOCAL_PATH	1030	Incorrect local path
SFTP_ERROR_REMOTE_PATH	1031	Incorrect remote path
SFTP_ERROR_TRANSFER	1032	Failed to transfer the file
SFTP_ERROR_FILE_TYPE	1033	Incorrect file type
SFTP_ERROR_FILE_MOVE	1034	Failed to move the file
SFTP_ERROR_FILE_DELETE	1035	Failed to delete the file
SFTP_ERROR_MKDIR	1036	Failed to create a directory
SFTP_ERROR_PERMISSION	1037	No permission
SFTP_ERROR_NO_SPACE	1038	No space

18.4.18 SysPub

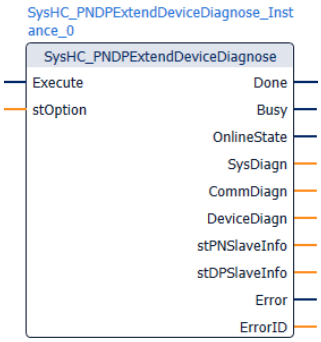
18.4.18.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable
18.4.18.2 SysHC_PNDPExtend DeviceDiagnose	FB	Get the fault diagnosis status of the master and slave of the PN and DP modules	SysPub	Library version 1.0.0.0 and later

18.4.18.2 SysHC_PNDPExtendDeviceDiagnose

This instruction gets the fault diagnosis status of the master and slave of the PN and DP modules.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
SysHC_PNDPExtend DeviceDiagnose	Get the fault diagnosis status of the master and slave of the PN and DP modules	FB		SysHC_PNDPExtendDeviceDiagnose_Instance_0 (Execute := , stOption := , Done => , Busy => , OnlineState => , SysDiagn => , CommDiagn => , DeviceDiagn => , stPNSlaveInfo => , stDPSlaveInfo => , Error => , ErrorID =>);

- **Related variables**

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Input	Execute	Enable	BOOL	TRUE, FALSE	FALSE	Enable, triggered by high-level
	stOption	Configure parameters	NETX_CFG_ OPTION	Follows data type	-	Obtain the diagnostic status of the corresponding PN/DP master and slave based on the input stOption.
Output	Done	Done	BOOL	Follows data type	FALSE	Set to TRUE when the function block execution is completed.
	Busy	Executing	BOOL	TRUE, FALSE	FALSE	Set to TRUE when the function block is executing.

Input/ Output	Variable	Name	Data Type	Valid Range	Default Value	Description
	OnlineState	Online status	BOOL	TRUE, FALSE	FALSE	TRUE: The slave is online. FALSE: The slave is offline.
	SysDiagn	Master system diagnostic information	DWORD	Depends on data type	0	PN master system diagnostic code. For details, see PN system master diagnostic codes.
	CommDiagn	Master communication diagnostic information	DWORD	Depends on data type	0	PN master communication diagnostic code. For details, see PN master communication diagnostic codes.
	DeviceDiagn	Communication module diagnostic information	DWORD	Depends on data type	0	PN communication module diagnostic code. For details, see PN master diagnostic codes.
	stPNSlaveInfo	PN slave information	NETX_PN_SLAVEINFO	Follows data type	-	PN slave information
	stDPSlaveInfo	DP slave information	NETX_DP_SLAVEINFO	Follows data type	-	DP slave information
	Error	Faulty status	BOOL	Follows data type	FALSE	Indicates whether a fault occurred during instruction execution.
	ErrorID	Instruction fault ID	NETX_ERROR	Follows data type	0	1: MASTER_TYPE_ERROR (Master type error) 2: MASTER_SLOT_ERROR (Master slot number error) 3: SLAVE_INDEX_ERROR (PN slave index error) 4: SLAVE_ADDR_ERROR (DP slave address error)

- **Function**
  - When the input parameter "Execute" changes from FALSE to TRUE, the instruction is enabled and can be used to obtain diagnostic information after certain conditions are met.
  - Based on the input stOption information, the system outputs the corresponding diagnostic status of the PN/DP master and slaves. The data type of stOption is NETX_CFG_OPTION. The meanings of the structure members are shown in the following table.

Name	Data Type	Description
SlotNum	BYTE	PN/DP physical slot number Valid values are 1, 2, 3, 4; default: 0
SlaveIndex	BYTE	PN slave index number Valid range is 1 to 128; default: 0

Name	Data Type	Description
SlaveAddr	BYTE	DP slave address Valid range is 0 to 126; default:0

◦ The data type of stPNSlaveInfo is NETX_PN_SLAVEINFO. The meanings of the structure members are shown in the following table.

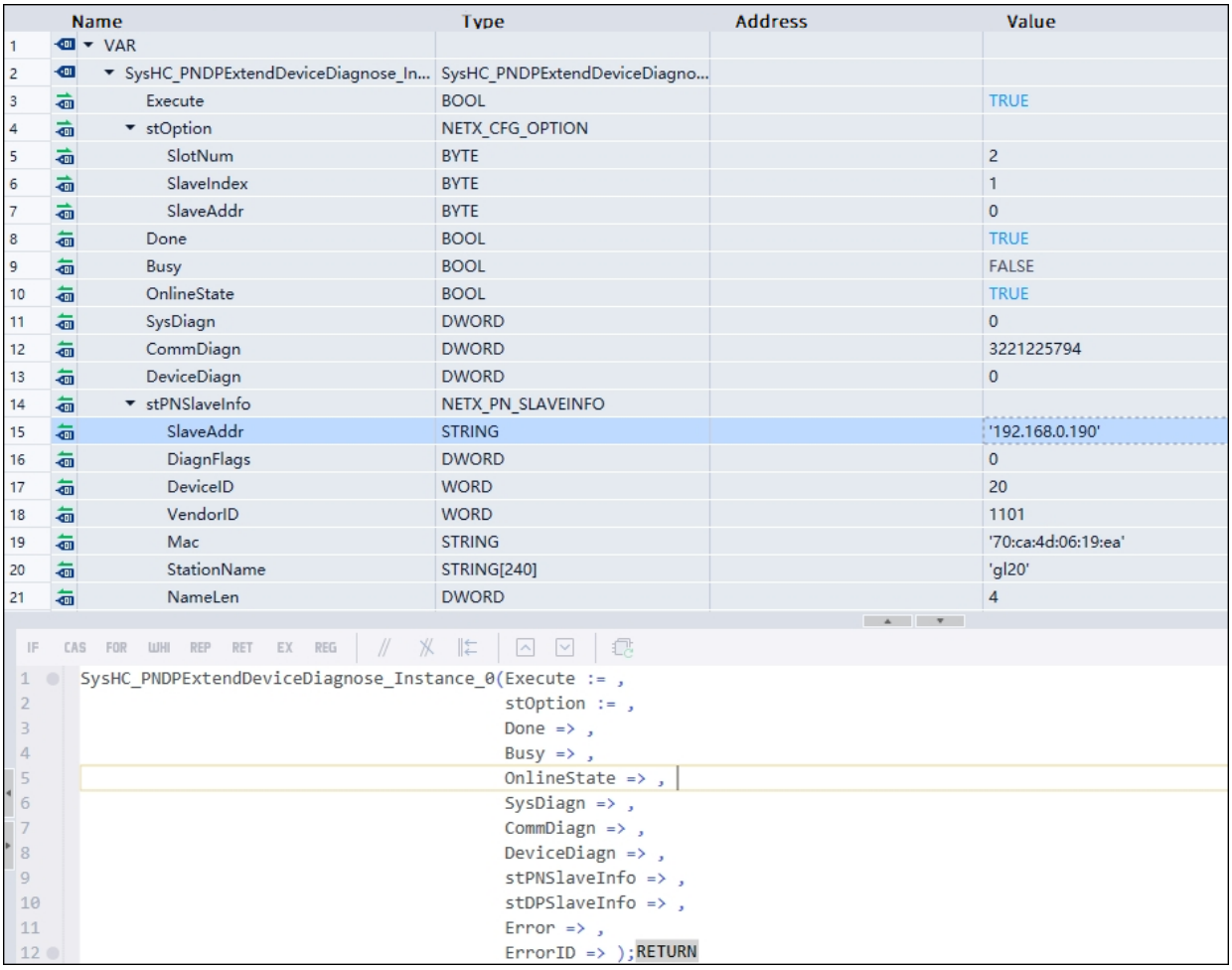
Name	Data Type	Description
SlaveAddr	STRING	PN slave IP address
DiagnFlags	DWORD	PN slave diagnostic code information. For details, see PN slave diagnostic codes.
DeviceID	WORD	Device ID
VendorID	WORD	Vendor ID
Mac	STRING	PN slave Mac
StationName	STRING	PN slave name
NameLen	DWORD	Name length of PN slave

◦ The data type of stDPSlaveInfo is NETX_DP_SLAVEINFO. The meanings of the structure members are shown in the table below.

Name	Data Type	Description
SlaveState	BOOL	DP slave status information. For details, see DP slave information.
SlaveDiag	BOOL	DP slave diagnostic information. For details, see DP slave information.

- **Program example**

  ST:

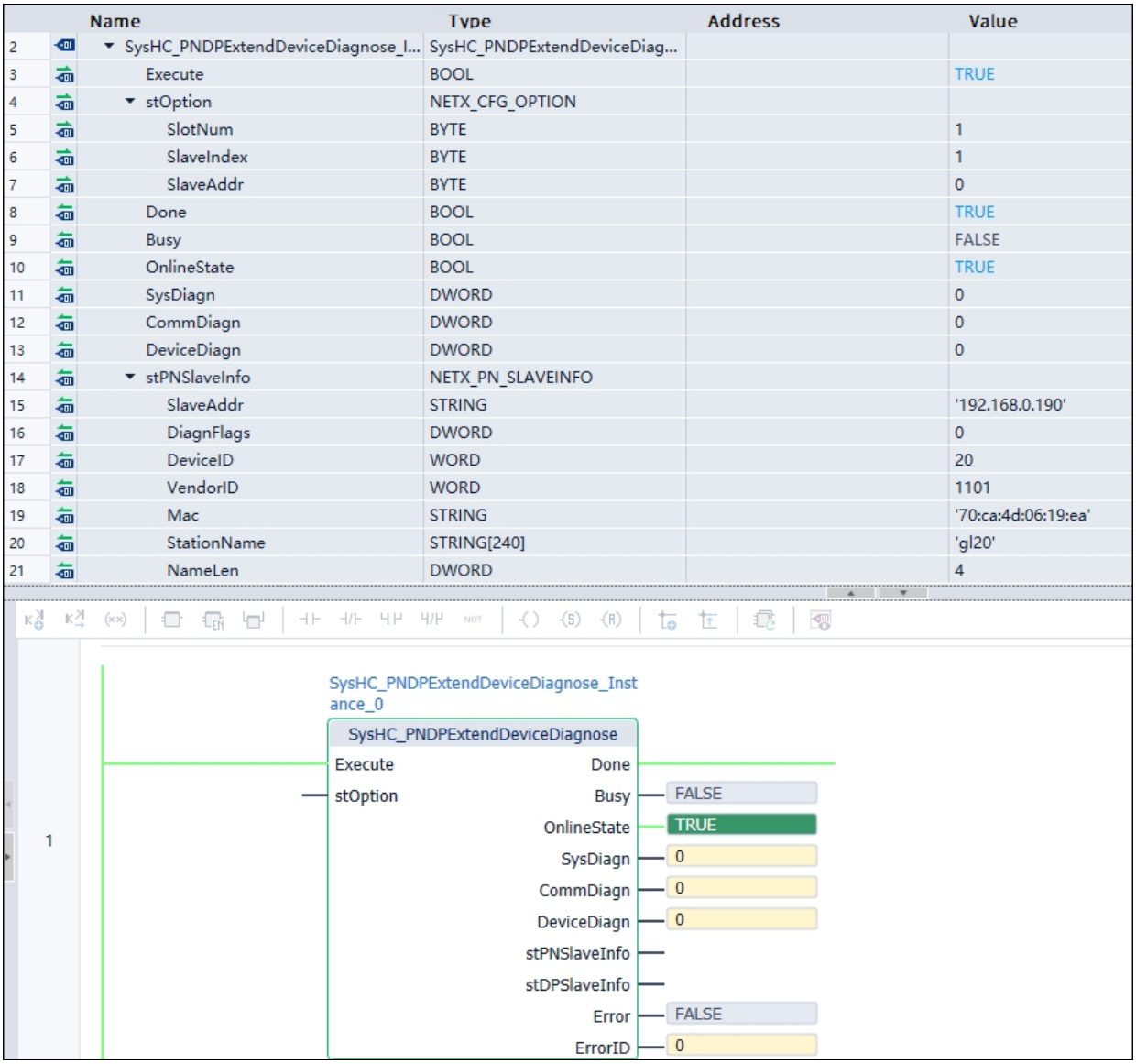


	Name	Type	Address	Value
1	VAR			
2	SysHC_PNDPExtendDeviceDiagnose_In...	SysHC_PNDPExtendDeviceDiagno...		
3	Execute	BOOL		TRUE
4	stOption	NETX_CFG_OPTION		
5	SlotNum	BYTE		2
6	SlaveIndex	BYTE		1
7	SlaveAddr	BYTE		0
8	Done	BOOL		TRUE
9	Busy	BOOL		FALSE
10	OnlineState	BOOL		TRUE
11	SysDiagn	DWORD		0
12	CommDiagn	DWORD		3221225794
13	DeviceDiagn	DWORD		0
14	stPNSlaveInfo	NETX_PN_SLAVEINFO		
15	SlaveAddr	STRING		'192.168.0.190'
16	DiagnFlags	DWORD		0
17	DeviceID	WORD		20
18	VendorID	WORD		1101
19	Mac	STRING		'70:ca:4d:06:19:ea'
20	StationName	STRING[240]		'gl20'
21	NameLen	DWORD		4

IF CAS FOR WHI REP RET EX REG

```
1  SysHC_PNDPExtendDeviceDiagnose_Instance_0(Execute := ,
2                                            stOption := ,
3                                            Done => ,
4                                            Busy => ,
5                                            OnlineState => ,
6                                            SysDiagn => ,
7                                            CommDiagn => ,
8                                            DeviceDiagn => ,
9                                            stPNSlaveInfo => ,
10                                           stDPSlaveInfo => ,
11                                           Error => ,
12                                           ErrorID => );RETURN
```

LD:



	Name	Type	Address	Value
2	SysHC_PNDPExtendDeviceDiagnose_I...	SysHC_PNDPExtendDeviceDiag...		
3	Execute	BOOL		TRUE
4	stOption	NETX_CFG_OPTION		
5	SlotNum	BYTE		1
6	SlaveIndex	BYTE		1
7	SlaveAddr	BYTE		0
8	Done	BOOL		TRUE
9	Busy	BOOL		FALSE
10	OnlineState	BOOL		TRUE
11	SysDiagn	DWORD		0
12	CommDiagn	DWORD		0
13	DeviceDiagn	DWORD		0
14	stPNSlaveInfo	NETX_PN_SLAVEINFO		
15	SlaveAddr	STRING		'192.168.0.190'
16	DiagnFlags	DWORD		0
17	DeviceID	WORD		20
18	VendorID	WORD		1101
19	Mac	STRING		'70:ca:4d:06:19:ea'
20	StationName	STRING[240]		'gl20'
21	NameLen	DWORD		4

18.4.18.3 Diagnostic Codes

PN master diagnostic codes

Each bit mask corresponds to a specific diagnostic status or error type. When you obtain a diagnostic code, you can perform AND operation between the code and each bit mask.

- If the result equals the bit mask, the corresponding diagnostic status exists and the bit is set to 1.
- If the result is 0, the corresponding diagnostic status does not exist.

  For example, if the obtained diagnostic code is 0x00011, it indicates errors of 0x00001 and 0x00010 occur simultaneously.

Bitmask	Description
0x00001	NetX module initialization failed
0x00002	NetX module status query error
0x00004	NetX module I/O data transmission error
0x00008	NetX module mailbox data transmission error
0x00010	Parallel port communication timeout
0x00020	Downlink mailbox data timeout

Bitmask	Description
0x00040	Downlink I/O data timeout
0x00080	Module not ready

18.4.19 VarForce

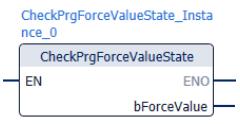
18.4.19.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable
18.4.19.2 CheckPrgForceValueState	FB	Check whether there are forced variables during the execution of the user program	VarForce	Library version 1.0.0.0 and later

18.4.19.2 CheckPrgForceValueState

This instruction checks whether there are forced variables during the execution of the user program.

- **Instruction format**

Instruction	Name	FB/FC	Ladder Diagram Representation	Structured Text Representation
CheckPrgForceValueState	Check whether there are forced variables during the execution of the user program	FB		CheckPrgForceValueState_Instance_0(bForceValue =>);

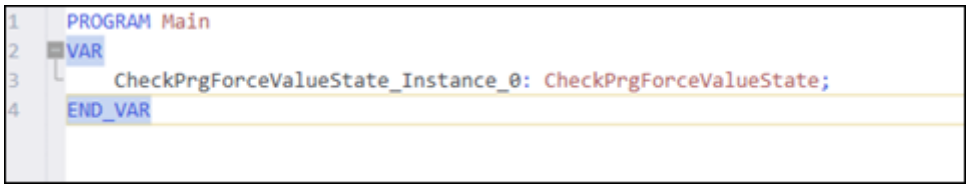
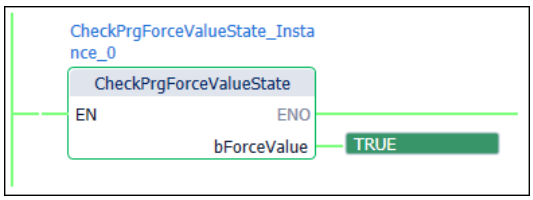
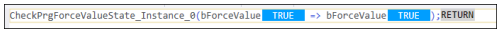
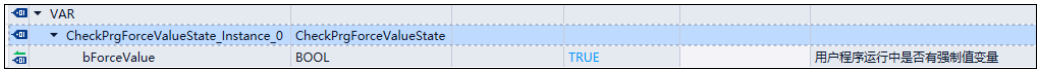
- **Related variables**

Input/Output	Variable	Name	Data Type	Valid Range	Default Value	Description
Output	bForceValue	Checks whether there are forced value variables during the execution of the user program	BOOL	TRUE, FALSE	FALSE	Return value of the function.

- **Function**

  This function block is used to obtain status. After it is called, it checks whether there are forced variables in the user program during the execution of the dependent task.

- **Program example**

	LD	ST
Define variables		
Program		
Operation result		

18.5 Special Instructions

18.5.1 Instruction Description

All instructions in this chapter involve pointers, you must use carefully. For string operations, you can use alternative instructions in 18.1.10 String.

18.5.2 Util

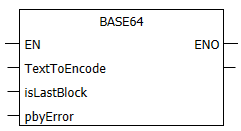
18.5.2.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.5.2.2 BASE64	FC	BASE64 encoding	Util	Library version 1.0.0.0 and later

18.5.2.2 BASE64

BASE64 is one of the encoding methods for transmitting 8-bit byte code, which represents binary data based on 64 printable characters.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
BASE64	BASE64 encoding	FC		:= BASE64(TextToEncode := , isLastBlock := , pbyError :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	TextToEncode	Text to be encoded	Follows data type	-	Data type is PT_SIZE: pString: POINTER TO BYTE (text address). uiSize: UINT (text length). The maximum supported encoding length is 22800.
	isLastBlock	Whether the current text is complete	Follows data type	-	TRUE: The current text is complete. FALSE: The current text is only a part.
	pbyError	Error pointer	Follows data type	-	0: No error. 1: The text is too long. 2: The text character length is not an integer multiple of 3. 4: The text pointer is null or the length uiSize = 0.
Output	OUT	Output data	Follows data type	-	Output data.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
TextToEncode	PT_SIZE: pString: POINTER TO BYTE (text address) uiSize: UINT (text length)																			
isLastBlock	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
pbyError	POINTER TO BYTE																			
OUT	PT_SIZE																			

- **Function**

  Specific encoding method:

  Base64 requires converting every three 8-bit bytes into four 6-bit bytes (since 3 × 8 = 4 × 6 = 24). Then, two high-order zeros are padded to each 6-bit bytes to form four 8-bit bytes. Each byte corresponds to a number less than 64, which is the character index. The converted string will theoretically be one-third longer than the original.

  Three English characters can be exactly converted into four Base64 characters. If the condition is not to meet, the remaining bytes of the source text are individually converted according to the encoding rules (1 byte → 2 Base64 chars, 2 bytes → 3 Base64 chars, missing bits are padded with zeros). The result is then filled to 4 bytes using equal signs (=), with at most two equal signs. If the remainder is 0, the original byte count is exactly a multiple of 3 (ideal case). If the remainder is 1, it is converted into two Base64 characters. To make the Base64 output length a multiple of 4, two equal signs are appended; similarly, if the remainder is 2, one equal sign is appended. If the length of uiSize is greater than the length of the original string, append the difference with character "A" to the converted result.

  ◦ When isLastBlock is TRUE and TextToEncode.uiSize is 2, the encoding principle and results are shown in the table below:

Character	y								o								-							
ASCII code	121								111								-							
Binary bits	0	1	1	1	1	0	0	1	0	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0
Encoding byte	00 011110								00 010110								00 111100							
Index number	30								22								60							
																	0							

Character	y	o	-
Base64 encoding	e	W	8	=

◦ When isLastBlock is TRUE and TextToEncode.uiSize is 1, the encoding principle and results are shown in the table below:

Character	y	-	-
ASCII code	121	-	-
Binary bits	0 1 1 1 1 0 0 1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
Encoding byte	00 011001	00 010000	00 000000	00 000000
Index number	25	16	0	0
Base64 encoding	Z	Q	=	=

◦ When isLastBlock is TRUE and TextToEncode.uiSize is 3, the encoding principle and results are shown in the table below:

Character	y	o	u
ASCII code	121	111	117
Binary bits	0 1 1 1 1 0 0 1	0 1 1 0 1 1 1 1	0 1 1 1 0 1 0 1
Encoding byte	00 011110	00 010110	00 111101	00 110101
Index number	30	22	61	53
Base64 encoding	e	W	9	1

- **Program example**

  Set TextToEncode to 'son', and the encoded output data is 'c29u'.

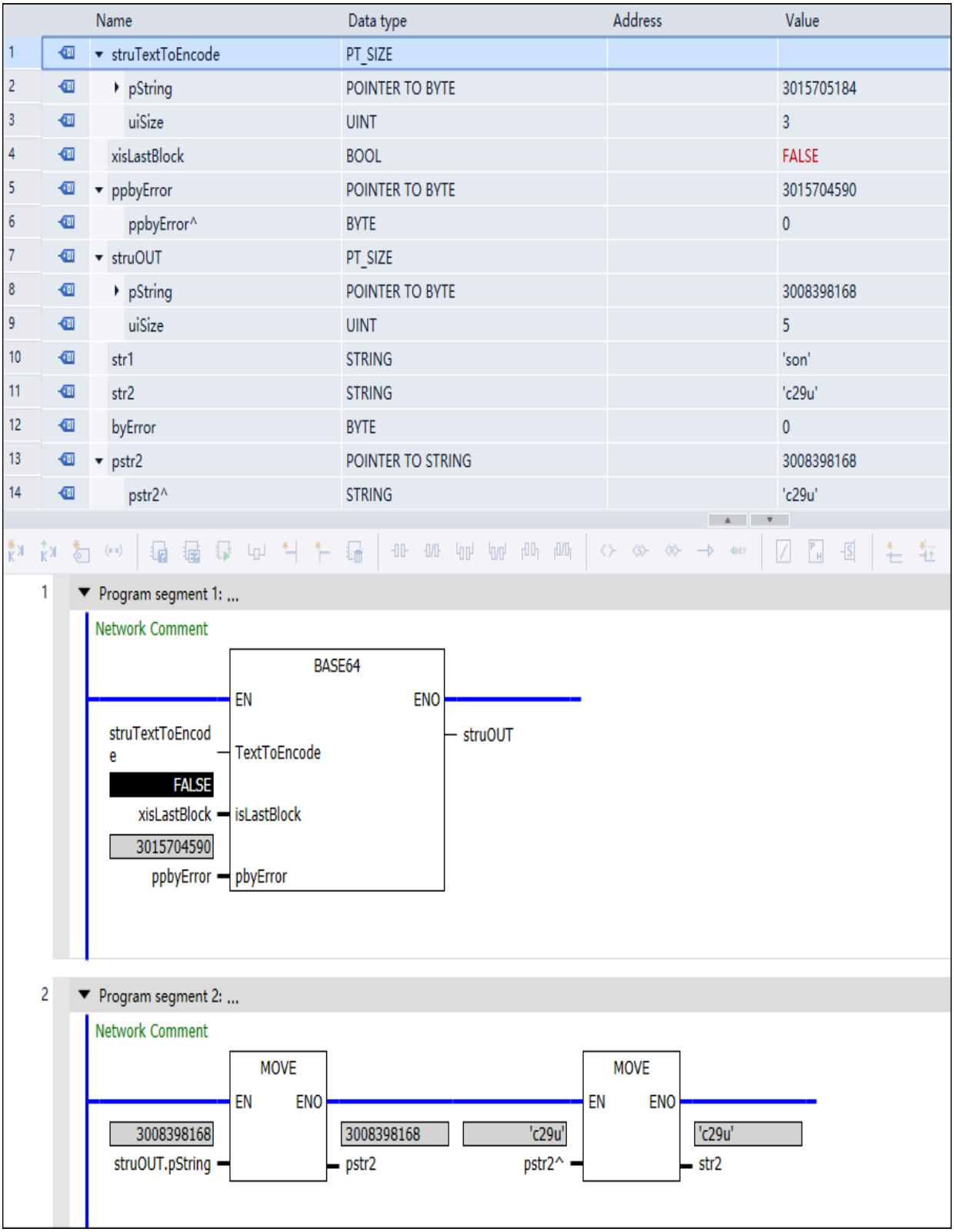
  ST:

	Name	Data type	Address	Value
1	struTextToEncode	PT_SIZE		
2	pString	POINTER TO BYTE		3015704984
3	uiSize	UINT		3
4	xisLastBlock	BOOL		FALSE
5	ppbyError	POINTER TO BYTE		3015704588
6	ppbyError^	BYTE		0
7	struOUT	PT_SIZE		
8	pString	POINTER TO BYTE		3008398168
9	uiSize	UINT		5
10	str1	STRING		'son'
11	str2	STRING		'c29u'
12	byError	BYTE		0
13	pstr2	POINTER TO STRING		3008398168
14	pstr2^	STRING		'c29u'

IF CAS FOR WHI REP RET EX

```
1  struTextToEncode.uiSize 3 := 3;
2  struTextToEncode.pString 3015704984 := ADR(str1 'son' );
3  ppbyError 3015704588 := ADR(byError 0 );
4
5  struOUT := BASE64(TextToEncode := struTextToEncode,
6  isLastBlock := xisLastBlock FALSE ,
7  pbyError := ppbyError 3015704588 );
8
9  pstr2 3008398168 := struOUT.pString 3008398168 ;
10 str2 'c29u' := pstr2^ 'c29u' ;RETURN
```

LD:



◦ Encoding example 1: Input 'Son', uiSize=3, isLastBlock=1 → Base64 encoded result: 'U29u'

ASCII character	S	o	n
Decimal	83	111	110
Binary	01010011	01101111	01101110

Each group consists of 6 bits	010100	110110	111101	101110
High-order padding with zeros	00010100	00110110	00111101	00101110

ASCII character	S	o	n
Corresponding Base64 index	20	54	61	46
Corresponding Base64 character	U	2	9	u

◦ Encoding example 2: Input 'Son', uiSize = 2, isLastBlock = 1 → Base64 encoded result: 'U28u'

ASCII character	S	o	-
Decimal	83	111	-
Binary	01010011	01101111	-

Each group consists of 6 bits	010100	110110	111100	-
High-order padding with zeros	00010100	00110110	00111100	-
Corresponding Base64 index	20	54	60	-
Corresponding Base64 character	U	2	8	=

◦ Encoding example 3: Input 'Son', uiSize = 1, isLastBlock = 1 → Base64 encoded result: 'Uw=='

ASCII character	S	-	-
Decimal	83	-	-
Binary	01010011	-	-

Each group consists of 6 bits	010100	110000	-	-
High-order padding with zeros	00010100	00110000	-	-
Corresponding Base64 index	20	48	-	-
Corresponding Base64 character	U	w	=	=

◦ Encoding example 4 (Chinese): For Base64 encoding of Chinese characters, use the String type under the UTF-8 format.
For example, input "你", uiSize = 3 → Base64 encoded result: '5L2g'

ASCII character	你		
Decimal	228	189	160
Binary	11100100	10111101	10100000

Each group consists of 6 bits	111001	001011	110110	000000
High-order padding with zeros	00111001	00001011	00110110	00100000
Corresponding Base64 index	57	11	54	32

ASCII character	你			
Corresponding Base64 character	5	L	2	g

◦ Fault description:
If any of the following errors occur, Base64.uiSize outputs 0 and the text result is incorrect. Normal usage should proceed when no ERROR is detected.

Condition	ERROR
TextToEncode.pString = 0 The input string pointer is null	4
TextToEncode.uiSize=0	4
TextToEncode.uiSize > 32767 The length of input string is greater than 32767	1
isLastBlock = FALSE (only part of the text) and the length is not a multiple of 3	2

- **Precautions**

  BASE64 supports a maximum encoding length of 22800

18.5.3 StringUtils

18.5.3.1 Ansi

18.5.3.1.1 Instruction List

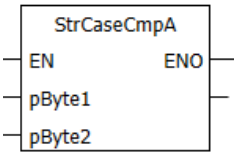
Instruction	FB/FC	Function	Library	Applicable Library Version
18.5.3.1.2 StrCaseCmpA	FC	String comparison (case-insensitive)	StringUtils	Library version 1.0.0.0 and later
18.5.3.1.3 StrCaseCmpEndA	FC	String end comparison (case-insensitive)		Library version 1.0.0.0 and later
18.5.3.1.4 StrCaseCmpStartA	FC	String start comparison (case-insensitive)		Library version 1.0.0.0 and later
18.5.3.1.5 StrCaseFindA	FC	String finding (case-insensitive)		Library version 1.0.0.0 and later
18.5.3.1.6 StrCmpA	FC	String comparison (case-sensitive)		Library version 1.0.0.0 and later
18.5.3.1.7 StrCmpEndA	FC	String end comparison (case-sensitive)		Library version 1.0.0.0 and later
18.5.3.1.8 StrCmpStartA	FC	String start comparison (case-sensitive)		Library version 1.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.5.3.1.9 StrConcatA	FC	String concatenation		Library version 1.0.0.0 and later
18.5.3.1.10 StrCpyA	FC	String copy		Library version 1.0.0.0 and later
18.5.3.1.11 StrDeleteA	FC	String deletion		Library version 1.0.0.0 and later
18.5.3.1.12 StrFindA	FC	String find (case-sensitive)		Library version 1.0.0.0 and later
18.5.3.1.13 StrIsNullOrEmptyA	FC	Empty string judgement		Library version 1.0.0.0 and later
18.5.3.1.14 StrLenA	FC	Calculation of string length		Library version 1.0.0.0 and later
18.5.3.1.15 StrMidA	FC	Substring acquisition		Library version 1.0.0.0 and later
18.5.3.1.16 StrReplaceA	FC	This instruction replaces the string.		Library version 1.0.0.0 and later
18.5.3.1.17 StrToLowerA	FC	String lowercase conversion		Library version 1.0.0.0 and later
18.5.3.1.18 StrToUpperA	FC	String uppercase conversion		Library version 1.0.0.0 and later
18.5.3.1.19 StrTrimA	FC	Trim the leading and trailing spaces from the string		Library version 1.0.0.0 and later
18.5.3.1.20 StrTrimEndA	FC	This instruction trims the trailing spaces from the string.		Library version 1.0.0.0 and later
18.5.3.1.21 StrTrimStartA	FC	Trim leading spaces from the string		Library version 1.0.0.0 and later
18.5.3.1.22 CharToUpper	FC	Character uppercase conversion		Library version 1.0.0.0 and later

18.5.3.1.2 StrCaseCmpA

This instruction compares two strings without case sensitivity.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrCaseCmpA	String comparison (case-insensitive)	FC		:= StrCaseCmpA(pByte1 := , pByte2 :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pByte1	Comparison STR 1	Follows data type	-	Pointer to the comparison STR 1.
	pByte2	Comparison STR 2	Follows data type	-	Pointer to the comparison STR 2.
Output	StrCaseCmpA	Return value	Follows data type	-	Comparison result.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pByte1	POINTER TO BYTE																			
pByte2	POINTER TO BYTE																			
StrCaseCmpA	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

- **Function**

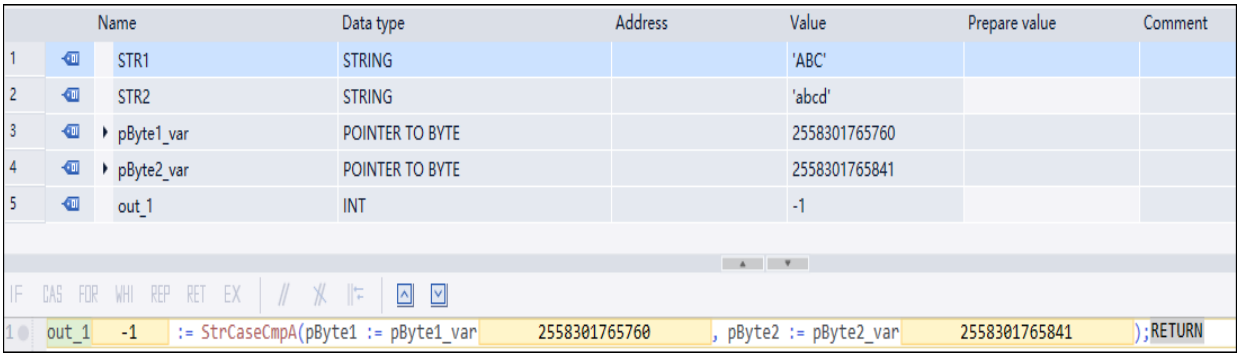
  This instruction compares two strings in a case-sensitive manner. The comparison results are as follows.

Return Value	Definition
0	The two strings are equal.
1	pByte1 is greater than pByte2.
-1	pByte1 is less than pByte2.
-2	pByte1 or pByte2 is a null pointer.

- **Program example**

  STR1 is the string pointed to by the pByte1_var pointer, STR2 is the string pointed to by the pByte2_var pointer.

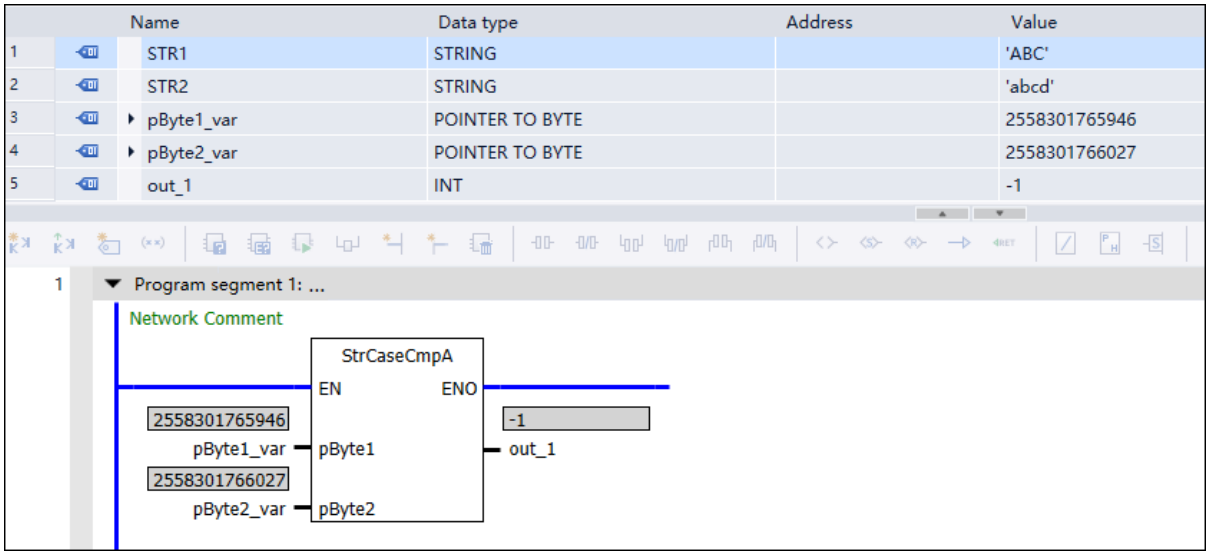
  ST:



	Name	Data type	Address	Value	Prepare value	Comment
1	STR1	STRING		'ABC'		
2	STR2	STRING		'abcd'		
3	pByte1_var	POINTER TO BYTE		2558301765760		
4	pByte2_var	POINTER TO BYTE		2558301765841		
5	out_1	INT		-1		

```
1  out_1  -1  := StrCaseCmpA(pByte1 := pByte1_var  2558301765760 , pByte2 := pByte2_var  2558301765841 );RETURN
```

LD:

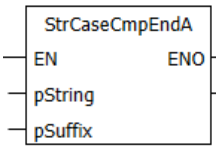


	Name	Data type	Address	Value
1	STR1	STRING		'ABC'
2	STR2	STRING		'abcd'
3	pByte1_var	POINTER TO BYTE		2558301765946
4	pByte2_var	POINTER TO BYTE		2558301766027
5	out_1	INT		-1

18.5.3.1.3 StrCaseCmpEndA

This instruction compares the endings of two strings in a case-insensitive manner.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrCaseCmpEndA	String end comparison (case-insensitive)	FC		:= StrCaseCmpEndA(pString := , pSuffix :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pString	Pointer to the string to be checked	Follows data type	-	Pointer to the string to be checked.

Input/Output	Variable	Name	Valid Range	Default Value	Description
	pSuffix	Pointer to the suffix to be found	Follows data type	-	Pointer to the suffix to be found.
Output	StrCaseCmpEndA	Return value	Follows data type	-	Comparison result.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pString	CharBufferPtr: POINTER TO BYTE																			
pSuffix	CharBufferPtr: POINTER TO BYTE																			
StrCaseCmpEndA	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction compares two strings in a case-sensitive manner. The comparison results are as follows.

Return Value	Definition
0	The ends and prefixes of both strings match.
-1	The ends and prefixes of both strings do not match.
-2	pString or pSuffix is a null pointer.

- **Precautions**

  When the suffix check result is empty, the return value will be 0.

- **Program example**

  STR1 is the string pointed to by the pString_var pointer, and STR2 is the string pointed to by the pSuffix_var pointer.

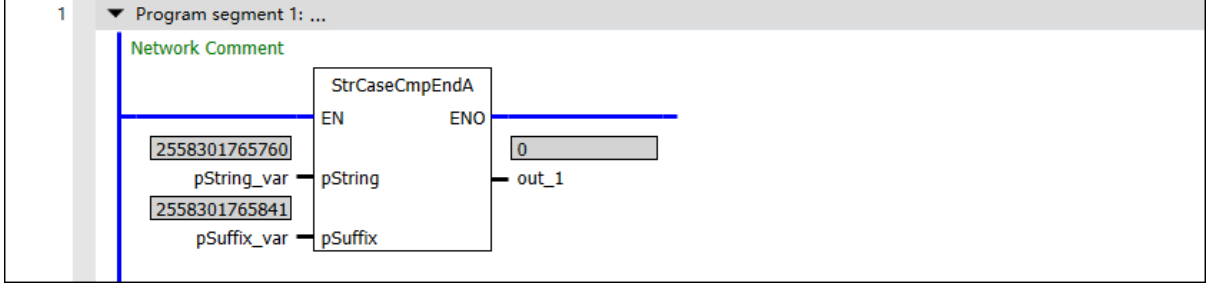
  ST:

	Name	Data type	Address	Value	Prepare value	Comment
1	STR1	STRING		'ABCdefgh'		
2	STR2	STRING		'FgH'		
3	pString_var	POINTER TO BYTE		2558301765946		
4	pSuffix_var	POINTER TO BYTE		2558301766027		
5	out_1	INT		0		

```
1  out_1 := StrCaseCmpEndA(pString := pString_var , pSuffix := pSuffix_var );RETURN
```

LD:

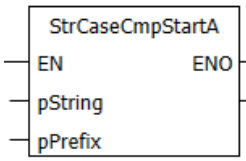
	Name	Data type	Address	Value
1	STR1	STRING		'ABCdefgh'
2	STR2	STRING		'FgH'
3	pString_var	POINTER TO BYTE		2558301765760
4	pSuffix_var	POINTER TO BYTE		2558301765841
5	out_1	INT		0



18.5.3.1.4 StrCaseCmpStartA

This instruction compares the starts of two strings in a case-insensitive manner.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrCaseCmpStartA	String start comparison (case-insensitive)	FC		:= StrCaseCmpStartA(pString := , pPrefix :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pString	Pointer to the string to be checked	Follows data type	-	Pointer to the string to be checked.
	pPrefix	Pointer to the prefix to be searched for	Follows data type	-	Pointer to the prefix to be searched for.

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	StrCaseCmpStartA	Return value	Follows data type	-	Comparison result.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pString	CharBufferPtr: POINTER TO BYTE																			
pPrefix	CharBufferPtr: POINTER TO BYTE																			
StrCaseCmpStartA	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction compares two strings in a case-sensitive manner. The comparison results are as follows.

Return Value	Definition
0	The starts and prefixes of both strings match.
-1	The starts and prefixes of both strings do not match.
-2	Errors arise from invalid strings, i.e., there is a null pointer.

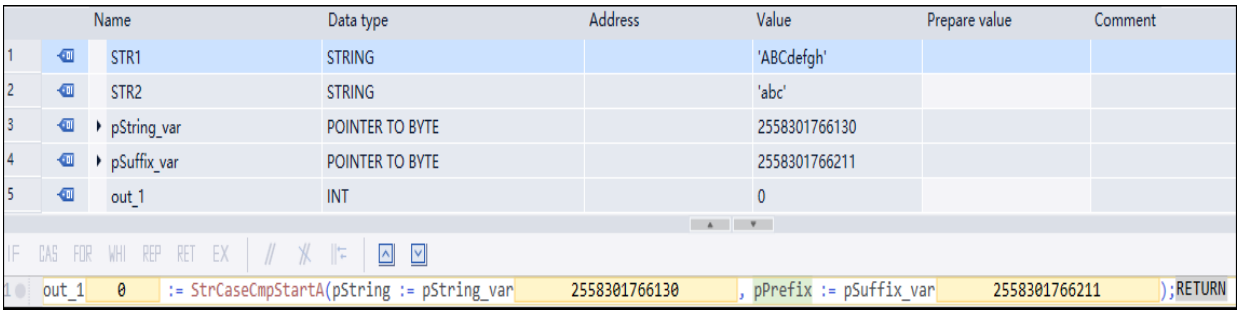
- **Precautions**

  When the prefix to be checked is empty, the return value is 0.

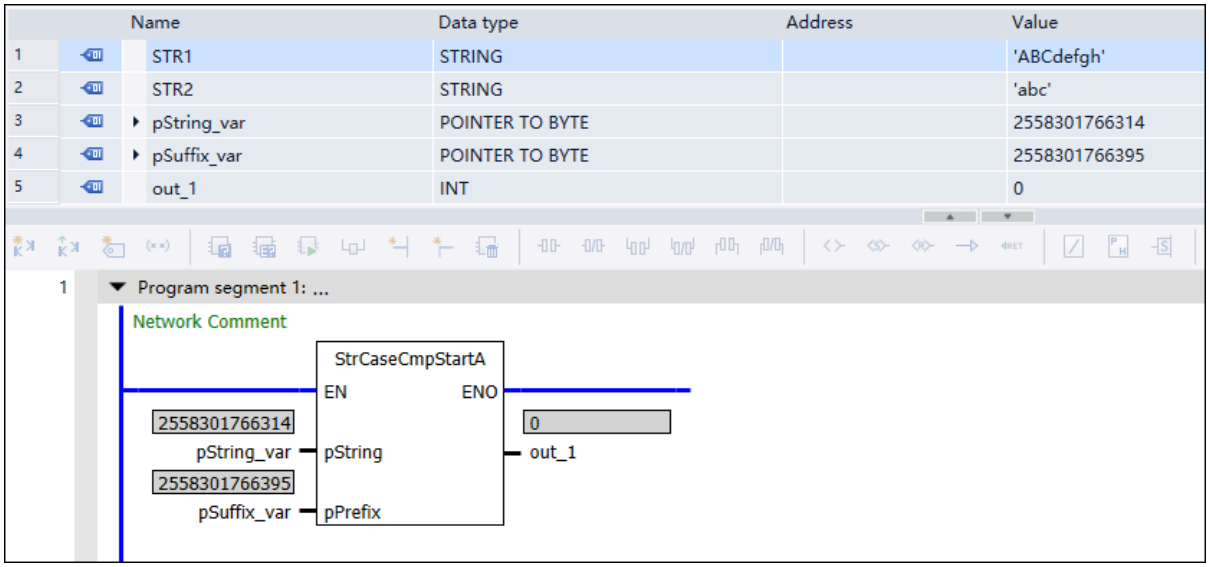
- **Program example**

  STR1 is the string pointed to by the pString_var pointer, and STR2 is the string pointed to by the pSuffix_var pointer.

  ST:



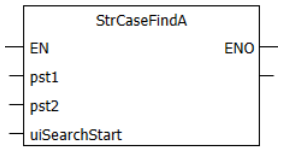
LD:



18.5.3.1.5 StrCaseFindA

This instruction performs a case-insensitive find operation on two strings.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrCaseFindA	String finding (case-insensitive)	FC		:= StrCaseFindA(pst1 := , pst2 := , uiSearchStart :=);

- **Related variables**

Variable

Input/ Output	Variable	Name	Valid Range	Default Value	Description
Input	pst1	Pointer to the string to be checked	Follows data type	-	Pointer to the string to be checked.
	pst2	Pointer to the string to be found	Follows data type	-	Pointer to the string to be found.
	uiSearchStart	Search start position	Follows data type	-	Search start position.
Output	StrCaseFindA	Return value	Follows data type	-	Result.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pst1	POINTER TO STRING[255]																			
pst2	POINTER TO STRING[255]																			
uiSearchStart	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
StrCaseFindA	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction performs a case-insensitive find operation on two strings, starting from the uiSearchStart position in STR1. If STR 2 appears in STR 1, return the position of the first character of STR 2 within STR 1. The result are as follows.

Return Value	Definition
0	The string to be found does not exist.
-1	Errors arise from invalid strings, i.e., there is a null pointer.
>=1	The position of the first character of STR 2 within STR 1, where the first character is counted as position 1.

- **Precautions**
  - When the search position uiSearchStart is 0 or 1, the search starts from the first character of STR1.
  - If the input is Chinese, use StrCaseFindW.
- **Program example**

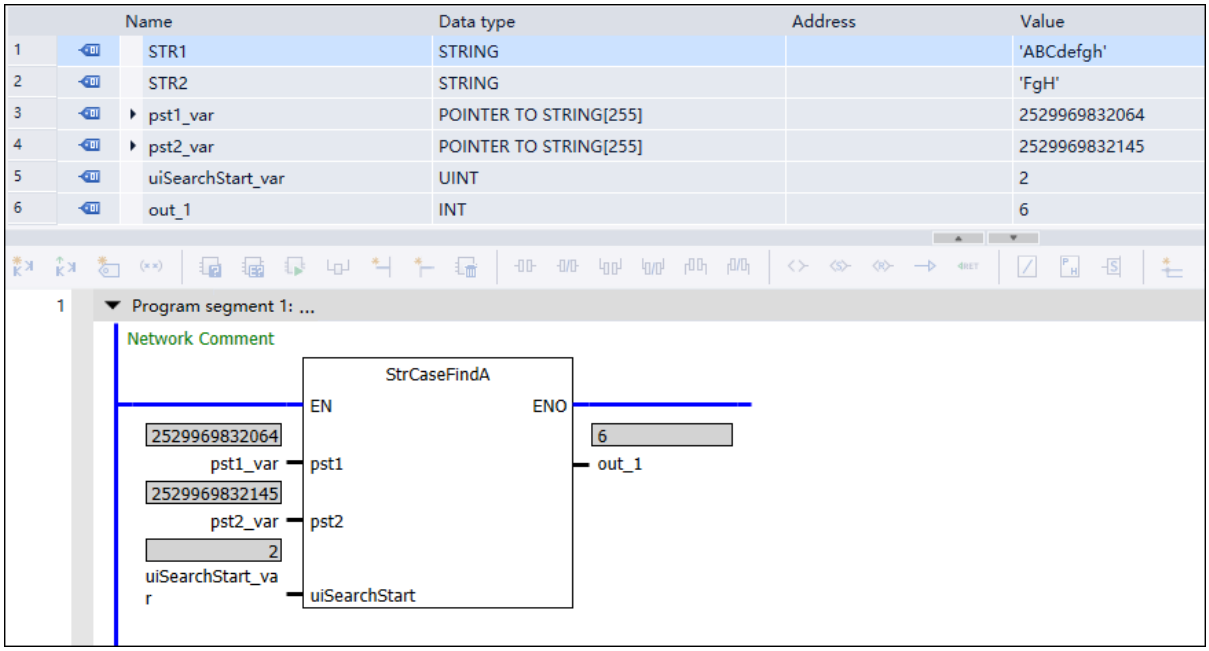
  STR1 is the string pointed to by the pst1_var pointer, and STR2 is the string pointed to by the pst2_var pointer.

  ST:

	Name	Data type	Address	Value	Prepare value	Comment
1	STR1	STRING		'ABCdefgh'		
2	STR2	STRING		'FgH'		
3	pst1_var	POINTER TO STRING[255]		2529969832258		
4	pst2_var	POINTER TO STRING[255]		2529969832339		
5	uiSearchStart_var	UINT		2		
6	out_1	INT		6		

```
1  out_1 := StrCaseFindA(pst1 := pst1_var, pst2 := pst2_var, uiSearchStart := uiSearchStart_var );RETURN
```

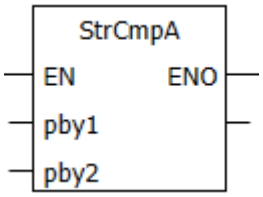
  LD:



18.5.3.1.6 StrCmpA

This instruction compares two strings in a case-sensitive manner.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrCmpA	String comparison (case-sensitive)	FC		:= StrCmpA(pby1 := , pby2 :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pBy1	Comparison STR 1	Follows data type	-	Pointer to the comparison STR 1.
	pBy2	Comparison STR 2	Follows data type	-	Pointer to the comparison STR 2.
Output	StrCmpA	Return value	Follows data type	-	Comparison result.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pBy1	POINTER TO BYTE																			
pBy2	POINTER TO BYTE																			
StrCmpA	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction compares two strings in a case-sensitive manner. The comparison results are as follows.

Return Value	Definition
0	The two strings are equal.
1	pByte1 is greater than pByte2.
-1	pByte1 is less than pByte2.

- **Precautions**

  If the input is Chinese, it is recommended to use StrCmpW.

- **Program example**

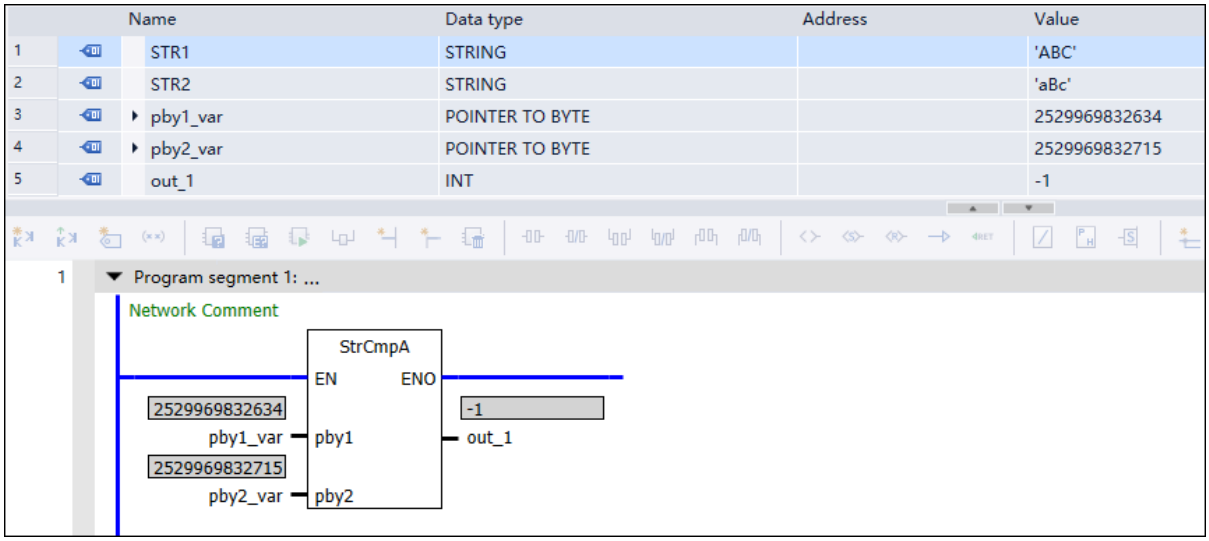
  STR1 is the string pointed to by the pby1_var pointer, and STR2 is the string pointed to by the pby2_var pointer.

  ST:

	Name	Data type	Address	Value	Prepare value
1	STR1	STRING		'ABC'	
2	STR2	STRING		'aBc'	
3	pby1_var	POINTER TO BYTE		2529969832450	
4	pby2_var	POINTER TO BYTE		2529969832531	
5	out_1	INT		-1	

```
IF  CAS  FOR  WHI  REP  RET  EX
1   out_1  -1  := StrCmpA(pby1 := pby1_var  2529969832450 , pby2 := pby2_var  2529969832531 );RETURN
```

  LD:



18.5.3.1.7 StrCmpEndA

This instruction compares the ends of two strings in a case-sensitive manner.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrCmpEndA	String end comparison (case-sensitive)	FC	StrCmpEndA EN ENO pString pSuffix	:= StrCmpEndA(pString := , pSuffix :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pString	Pointer to the string to be checked	Follows data type	-	Pointer to the string to be checked.
	pSuffix	Pointer to the suffix to be found	Follows data type	-	Pointer to the suffix to be found.
Output	StrCmpEndA	Return value	Follows data type	-	Comparison result.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pString	CharBufferPtr: POINTER TO BYTE																			
pSuffix	CharBufferPtr: POINTER TO BYTE																			
StrCmpEndA	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction compares two strings in a case-sensitive manner. The comparison results are as follows.

Return Value	Definition
0	The ends and prefixes of both strings match.
-1	The ends and prefixes of both strings do not match.
-2	Errors arise from invalid strings, i.e., there is a null pointer.

- **Program example**

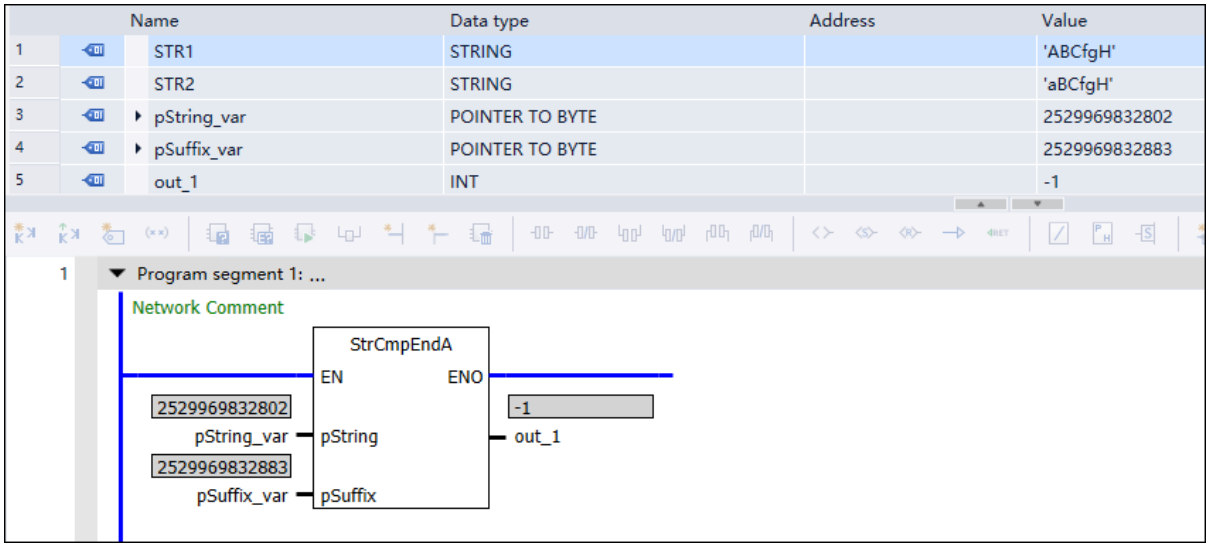
  STR1 is the string pointed to by the pString_var pointer, and STR2 is the string pointed to by the pSuffix_var pointer.

  ST:

	Name	Data type	Address	Value	Prepare value	Comment
1	STR1	STRING		'ABCfgH'		
2	STR2	STRING		'aBCfgH'		
3	pString_var	POINTER TO BYTE		2529969832986		
4	pSuffix_var	POINTER TO BYTE		2529969833067		
5	out_1	INT		-1		

```
1  out_1 := StrCmpEndA(pString := pString_var, pSuffix := pSuffix_var);RETURN
```

  LD:



18.5.3.1.8 StrCmpStartA

This instruction compares the starts of two strings in a case-sensitive manner.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrCmpStartA	String start comparison (case-sensitive)	FC	StrCmpStartA EN ENO pString pPrefix	:= StrCmpStartA(pString := , pPrefix :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pString	Pointer to the string to be checked	Follows data type	-	Pointer to the string to be checked.
	pPrefix	Pointer to the prefix to be searched for	Follows data type	-	Pointer to the prefix to be searched for.
Output	StrCmpStartA	Return value	Follows data type	-	Comparison result.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pString	CharBufferPtr: POINTER TO BYTE																			
pPrefix	CharBufferPtr: POINTER TO BYTE																			
StrCmpStartA	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction compares two strings in a case-sensitive manner. The comparison results are as follows.

Return Value	Definition
0	The starts and prefixes of both strings match.
-1	The starts and prefixes of both strings do not match.
-2	Errors arise from invalid strings, i.e., there is a null pointer.

- **Program example**

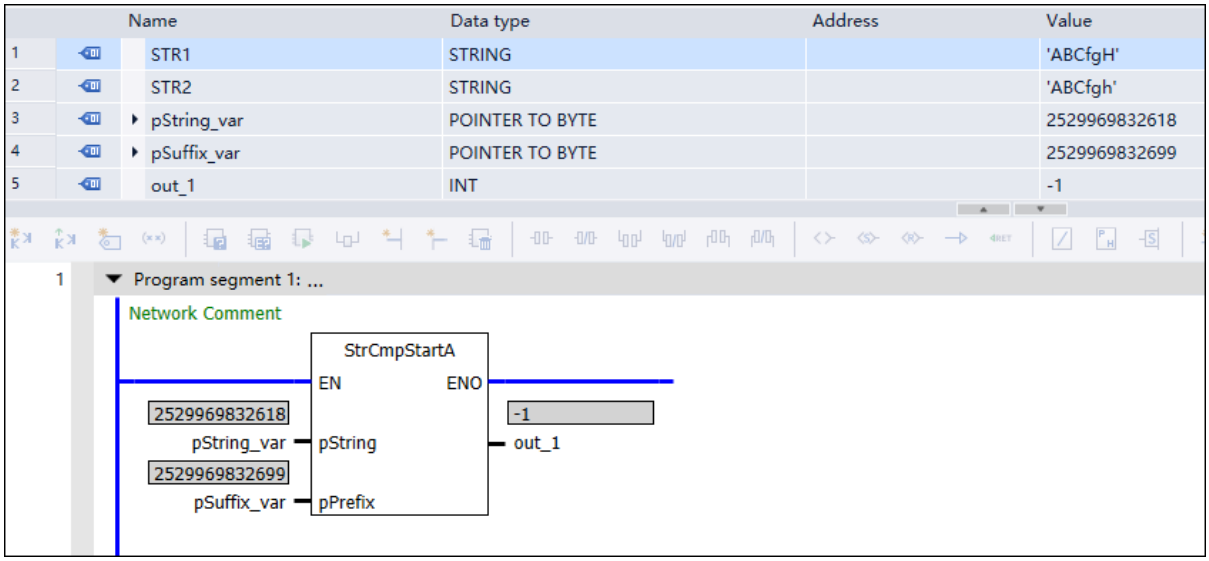
  STR1 is the string pointed to by the pString_var pointer, and STR2 is the string pointed to by the pSuffix_var pointer.

  ST:

	Name	Data type	Address	Value	Prepare value	Comment
1	STR1	STRING		'ABCfgH'		
2	STR2	STRING		'ABCfgh'		
3	pString_var	POINTER TO BYTE		2529969832434		
4	pSuffix_var	POINTER TO BYTE		2529969832515		
5	out_1	INT		-1		

```
1  out_1 := StrCmpStartA(pString := pString_var, pPrefix := pSuffix_var);RETURN
```

  LD:



18.5.3.1.9 StrConcatA

This instruction concatenates two strings.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrConcatA	String concatenation	FC	StrConcatA EN ENO pstFrom pstTo iBufferSize	:= StrConcatA(pstFrom := , pstTo := , iBufferSize :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pstFrom	Pointer to byte	Follows data type	-	Pointer to the bytes to be concatenated.
	pstTo	Pointer to byte	Follows data type	-	Pointer to the bytes to be appended.
	iBufferSize	Buffer size	Follows data type	-	Buffer size of "pstTo".
Output	StrConcatA	Return value	Follows data type	-	Operation result.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pstFrom	CharBufferPtr: POINTER TO BYTE																			
pstTo	CharBufferPtr: POINTER TO BYTE																			
iBufferSize	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
StrConcatA	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction concatenates the string pointed to by pstFrom to the end of pstTo. The possible scenarios are as follows.

Return Value	Definition
FALSE	The length of the concatenated string exceeds iBufferSize, the string pointed to by pstTo remains unchanged.
	Invalid string (i.e., there is a null pointer); the string pointed to by pstTo remains unchanged.
TRUE	If the result is normal, assign the concatenated string to the string pointed to by pstTo.

- **Precautions**

  If the input is in Chinese, it is recommended to use StrConcatW.

- **Program example**

  STR1 is the string pointed to by the pstFrom_var pointer, STR2 is the string pointed to by the pstTo_var pointer.

  ST:

  (1) Original state

	Name	Data type	Address	Value
1	STR1	STRING		'EFGH'
2	STR2	STRING		'ABCD'
3	pstFrom_var	POINTER TO BYTE		3016476706
4	pstTo_var	POINTER TO BYTE		3016476787
5	iBufferSize_var	INT		9
6	out_1	BOOL		FALSE

```
1  out_1 FALSE := StrConcatA(
2          pstFrom := pstFrom_var 3016476706 ,
3          pstTo := pstTo_var 3016476787 ,
4          iBufferSize := iBufferSize_var 9 );RETURN
```

(2) Enable

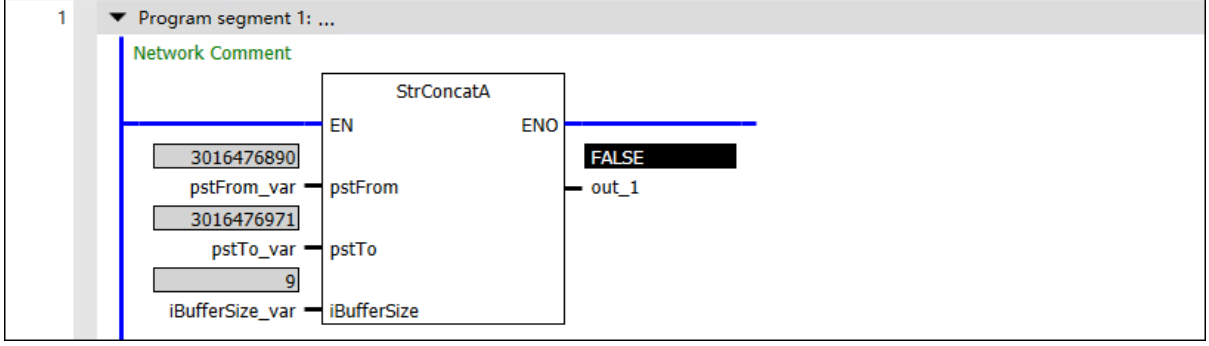
	Name	Data type	Address	Value
1	STR1	STRING		'EFGH'
2	STR2	STRING		'ABCDEFGH'
3	pstFrom_var	POINTER TO BYTE		3016476706
4	pstTo_var	POINTER TO BYTE		3016476787
5	iBufferSize_var	INT		9
6	out_1	BOOL		TRUE

```
1  out_1 TRUE := StrConcatA(
2          pstFrom := pstFrom_var 3016476706 ,
3          pstTo := pstTo_var 3016476787 ,
4          iBufferSize := iBufferSize_var 9 );RETURN
```

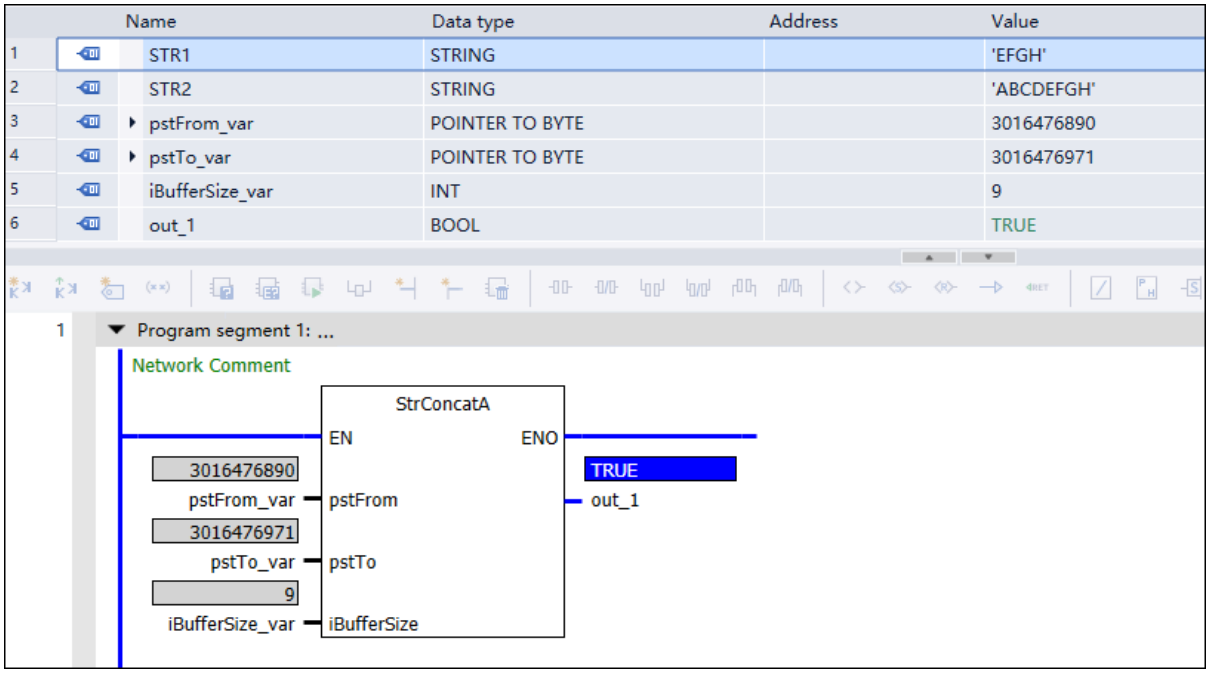
LD:

(1) Original state

	Name	Data type	Address	Value
1	STR1	STRING		'EFGH'
2	STR2	STRING		'ABCD'
3	pstFrom_var	POINTER TO BYTE		3016476890
4	pstTo_var	POINTER TO BYTE		3016476971
5	iBufferSize_var	INT		9
6	out_1	BOOL		FALSE



(2) Enable



18.5.3.1.10 StrCpyA

This instruction copies string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrCpyA	String copy	FC	StrCpyA EN ENO pBuffer iBufferSize pStr	:= StrCpyA(pBuffer := , iBufferSize := , pStr :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pBuffer	Pointer to the buffer	Follows data type	-	Pointer to the buffer.
	iBufferSize	Buffer size	Follows data type	-	Buffer size.
	pStr	Pointer to the source string	Follows data type	-	Pointer to the source string.
Output	StrCpyA	Return value	Follows data type	-	Return value.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pBuffer	CharBufferPtr: POINTER TO BYTE																			
iBufferSize	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
pStr	CharBufferPtr: POINTER TO BYTE																			
StrCpyA	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **Function**

  This instruction copies the string pointed to by pStr to the string pointed to by pBuffer. Note: The number of copied characters is the smaller value of the pointed source string length and buffer size.

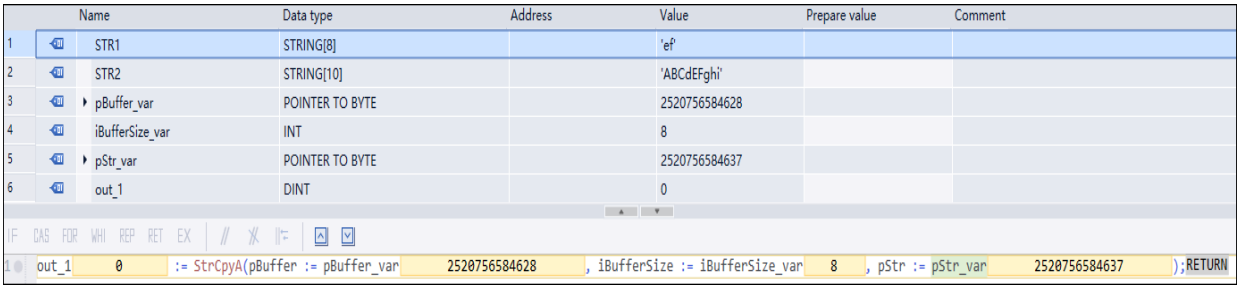
- **Precautions**
  - pBuffer must not be NULL.
  - Strings are typically null-terminated with “\0”, which should be considered when calculating the length.
  - If the copied length (the smaller value of pStr length and iBufferSize) exceeds the length of the string pointed to by pBuffer, unexpected results will occur.
  - If the input is Chinese, it is recommended to use StrCpyW.
  - The return value of the function is 0.

- **Program example**

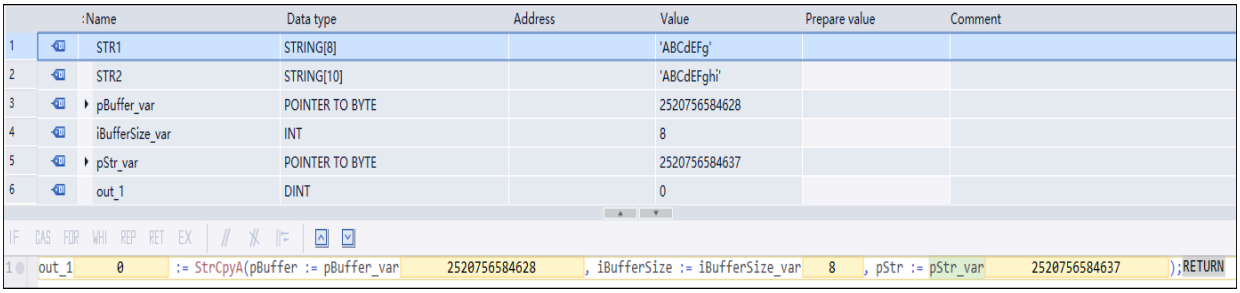
  STR1 is the string pointed to by the pBuffer_var pointer, and STR2 is the string pointed to by the pStr_var pointer.

  ST:

  (1) Original state



  (2) Enable

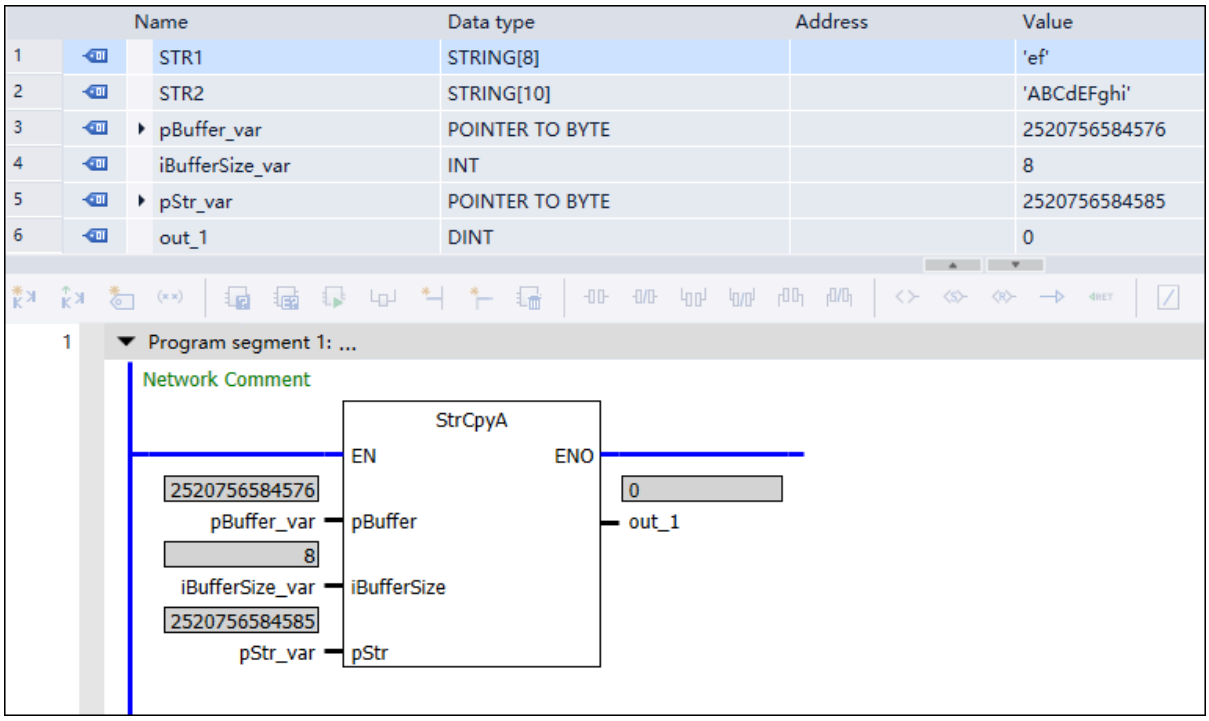


	Name	Data type	Address	Value	Prepare value	Comment
1	STR1	STRING[8]		'ABCdEFg'		
2	STR2	STRING[10]		'ABCdEFghi'		
3	pBuffer_var	POINTER TO BYTE		2520756584628		
4	iBufferSize_var	INT		8		
5	pStr_var	POINTER TO BYTE		2520756584637		
6	out_1	DINT		0		

```
1  out_1  0  := StrCpyA(pBuffer := pBuffer_var  2520756584628 , iBufferSize := iBufferSize_var  8 , pStr := pStr_var  2520756584637 );RETURN
```

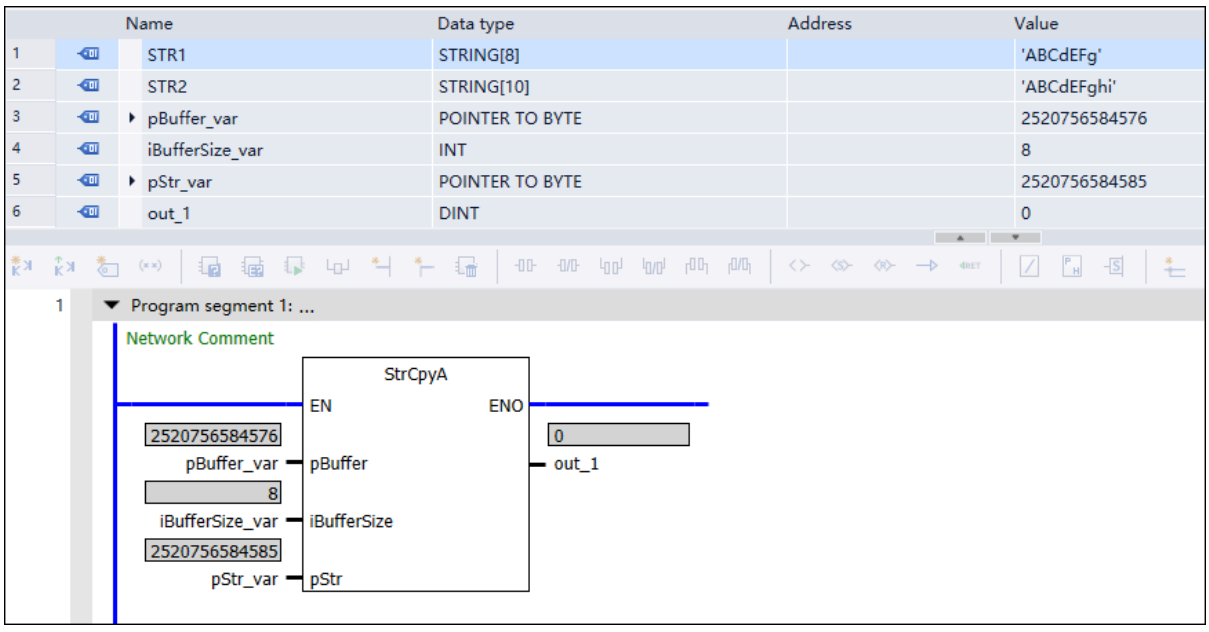
LD:

(1) Original state



	Name	Data type	Address	Value
1	STR1	STRING[8]		'ef'
2	STR2	STRING[10]		'ABCdEFghi'
3	pBuffer_var	POINTER TO BYTE		2520756584576
4	iBufferSize_var	INT		8
5	pStr_var	POINTER TO BYTE		2520756584585
6	out_1	DINT		0

(2) Enable

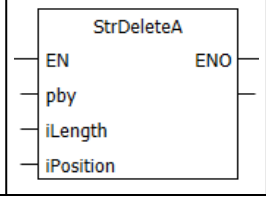


	Name	Data type	Address	Value
1	STR1	STRING[8]		'ABCdEFg'
2	STR2	STRING[10]		'ABCdEFghi'
3	pBuffer_var	POINTER TO BYTE		2520756584576
4	iBufferSize_var	INT		8
5	pStr_var	POINTER TO BYTE		2520756584585
6	out_1	DINT		0

18.5.3.1.11 StrDeleteA

This instruction deletes a specified-length string at a specified position from the input string, and output the processed string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrDeleteA	String deletion	FC		:= StrDeleteA(pby := , iLength := , iPosition :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pby	Pointer to the string to be deleted	Follows data type	-	Pointer to the string to be checked.
	iLength	Deletion length	Follows data type	-	Length of the string to be deleted.
	iPosition	Deletion position	Follows data type	-	Position in the string pointed to by the pby pointer.
Output	StrDeleteA	Return value	Follows data type	-	Return value.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pby	POINTER TO BYTE																			
iLength	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
iPosition	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
StrDeleteA	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction deletes a specified-length string at a specified position from the input string, and output the processed string.

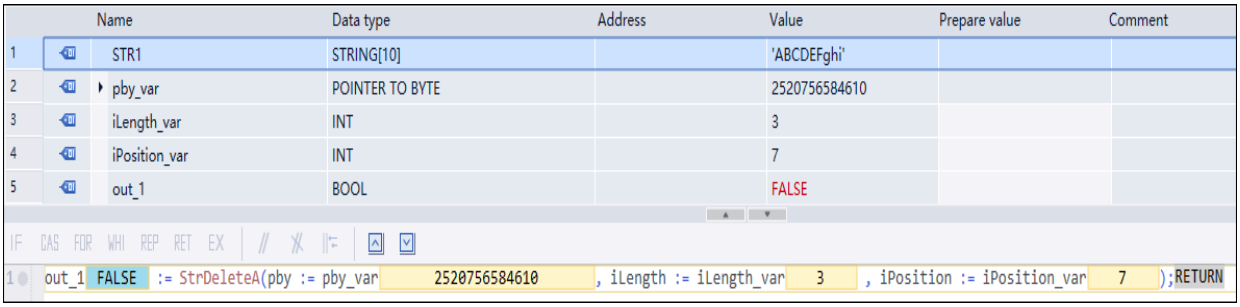
- **Precautions**
  - The input string must not be NULL.
  - If the deletion length is greater than the remaining length of the string from the deletion position, delete the remaining length.
  - If the input is Chinese, it is recommended to use StrDeleteW.
  - The return value of the function is 0.

- **Program example**

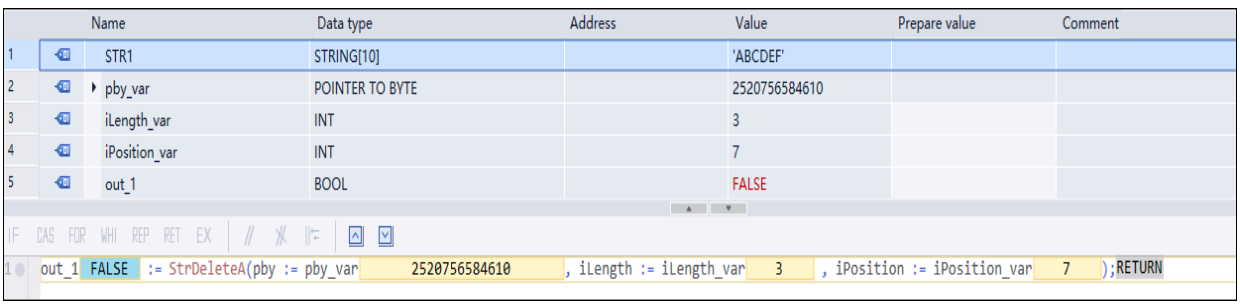
  STR1 is the string pointed to by the pby_var pointer.

  ST:

  (1) Original state

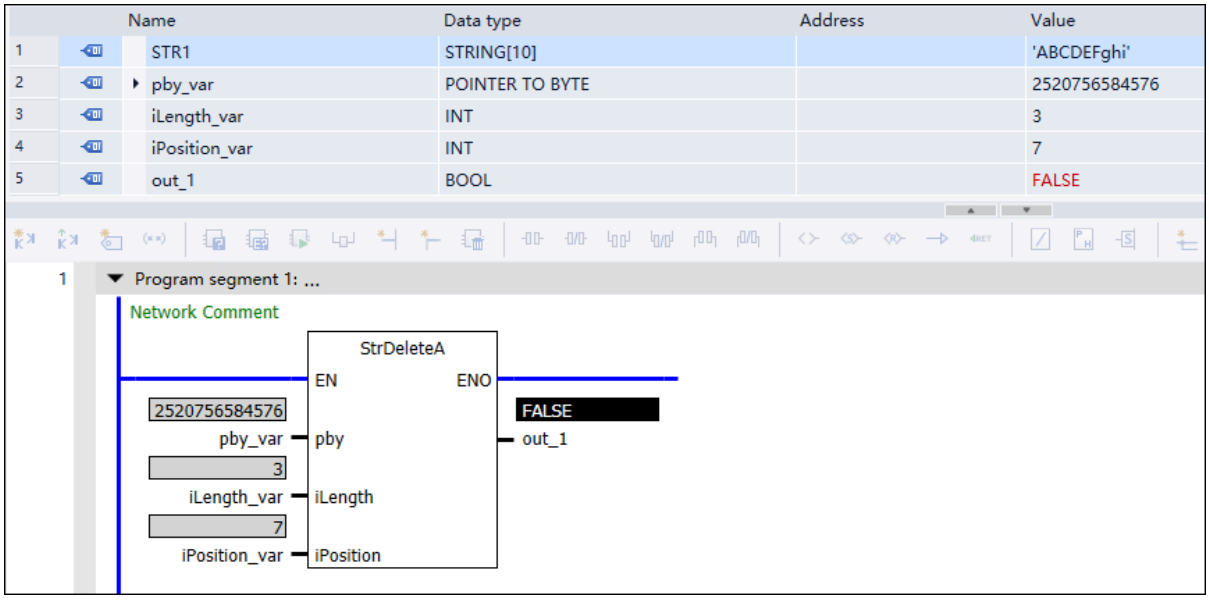


  (2) Enable

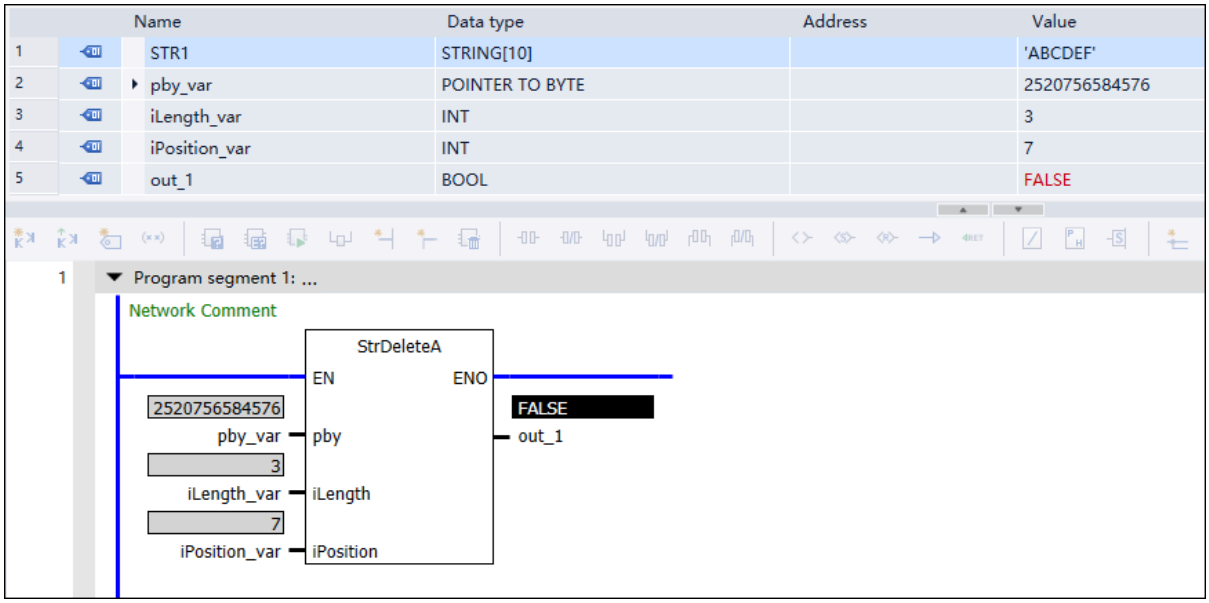


  LD:

(1) Original state



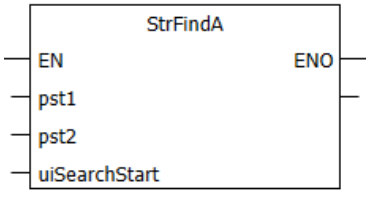
(2) Enable



18.5.3.1.12 StrFindA

This instruction performs a case-sensitive find operation on two strings.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrFindA	String find (case-sensitive)	FC		:= StrFindA(pst1 := , pst2 := , uiSearchStart :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pst1	Pointer to the string to be checked	Max length: 255	-	Pointer to the string to be checked.
	pst2	Pointer to the string to be found	Max length: 255	-	Pointer to the string to be found.
	uiSearchStart	Search start position	Follows data type	-	Search start position.
Output	StrFindA	Return value	Follows data type	-	Result.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pst1	POINTER TO STRING[255]																			
pst2	POINTER TO STRING[255]																			
uiSearchStart	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
StrFindA	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction performs a case-sensitive find operation on two strings starting from the uiSearchStart-th character of the STR 1. If the STR 2 appears in the STR 1, return the position of the first character of STR2 within STR 1. The results are as follows.

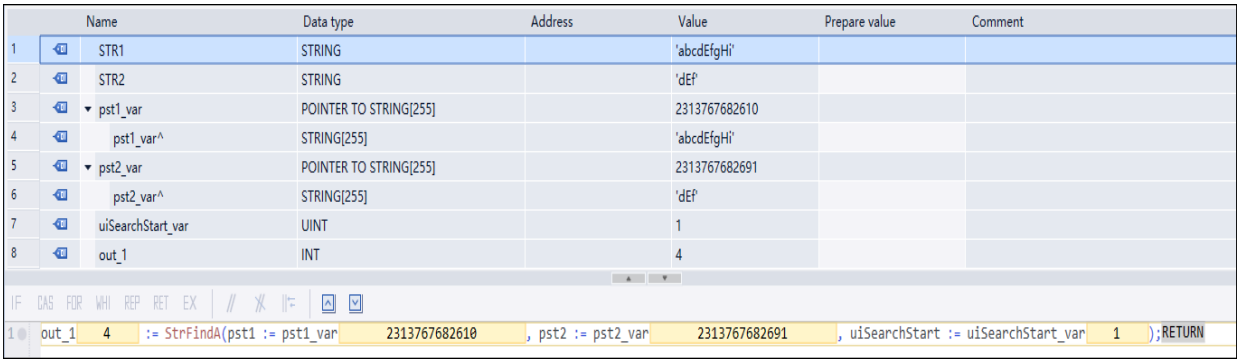
Return Value	Description
0	The string to be found does not exist.
-1	Errors arise from invalid strings, i.e., there is a null pointer.

Return Value	Description
>= 1	The position of the first character of STR 2 within STR 1, where the first character is counted as position 1.

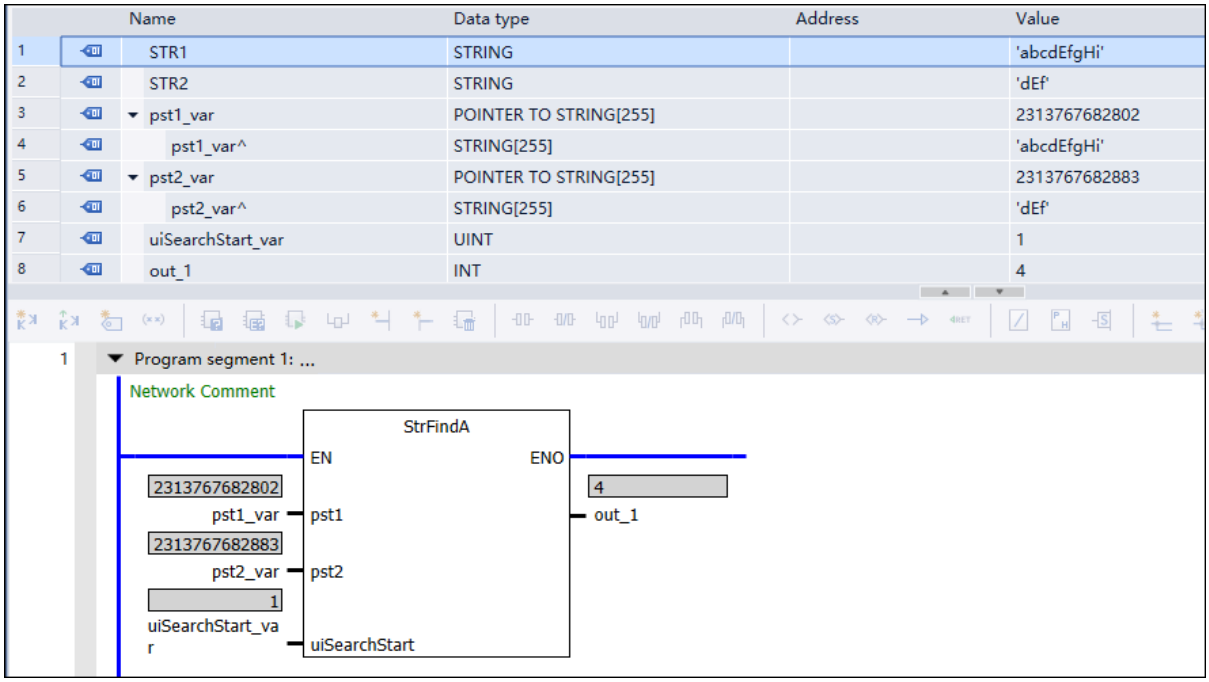
- **Precautions**
  - When the search position uiSearchStart is 0 or 1, the search starts from the first character of STR1.
  - If the input is Chinese, it is recommended to use StrFindW.
- **Program example**

  STR1 is the string pointed to by the pst1_var pointer, and STR2 is the string pointed to by the pst2_var pointer.

  ST:



  LD:



18.5.3.1.13 StrIsNullOrEmptyA

This instruction judges if a string is empty.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrIsNullOrEmptyA	Empty string judgement	FC	StrIsNullOrEmptyA EN ENO pstData	:= StrIsNullOrEmptyA(pstData :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pstData	Pointer to byte	Follows data type	-	Pointer to byte.
Output	StrIsNullOrEmptyA	Return value	Follows data type	-	Return value.

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pstData	CharBufferPtr: POINTER TO BYTE																			
StrIsNullOrEmptyA	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

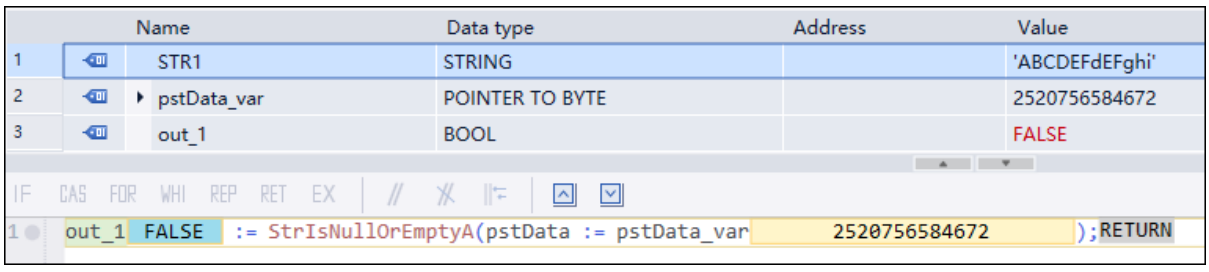
  This instruction judges if the string is empty. The result are as follows.

Return Value	Definition
0	The string is not empty.
1	The string is empty.

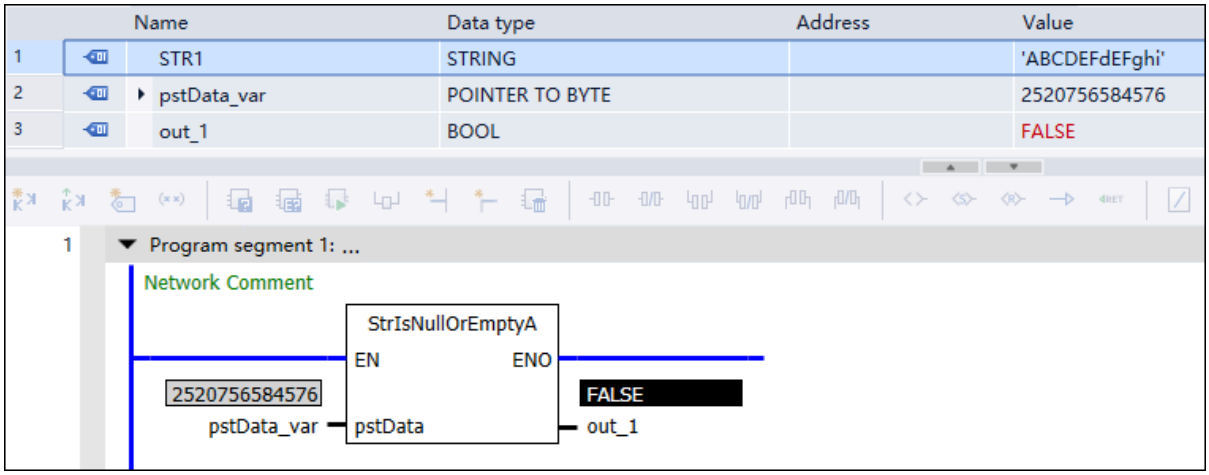
- **Program example**

  STR1 is the string pointed to by the pstData_var pointer.

  ST:



LD:



18.5.3.1.14 StrLenA

This instruction calculates the length of a string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrLenA	Calculation of string length	FC	StrLenA EN ENO pstData	:= StrLenA(pstData :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pstData	Pointer to byte	Follows data type	-	Pointer to byte.
Output	StrLenA	Return value	Follows data type	-	Return value.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pstData	CharBufferPtr: POINTER TO BYTE																			

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
StrLenA	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

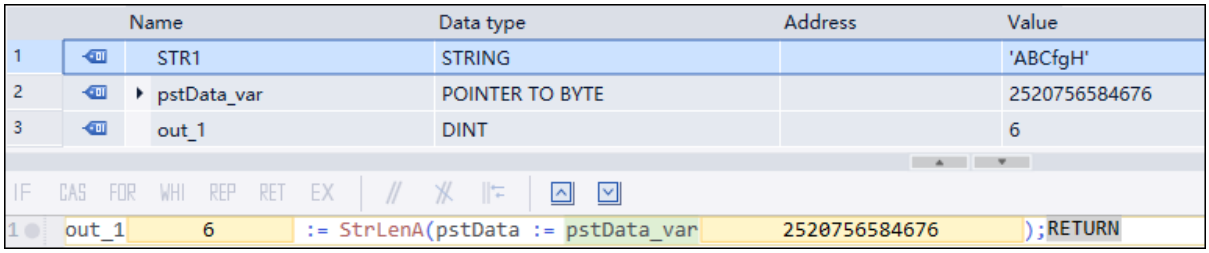
- **Function**
  - This instruction calculates the length of the string pointed to by pstData with null characters excluded, and output the result as a return value.
  - If the input is Chinese, it is recommended to use StrLenW.
  - Return value results are as follows.

Return Value	Definition
>=1	The length of a string.
-1	Pointer being null.

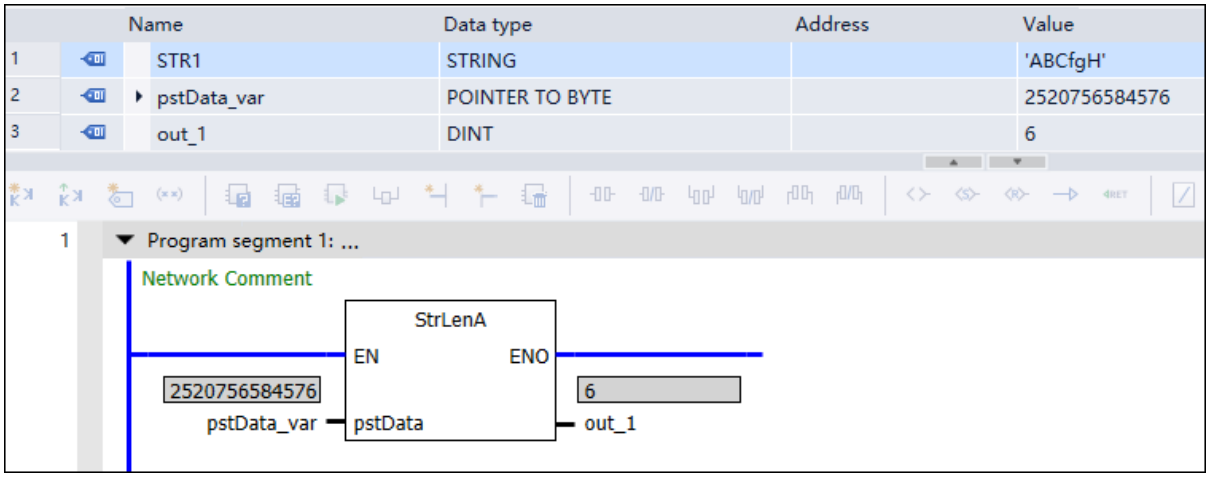
- **Program example**

  STR1 is the string pointed to by the pstData_var pointer.

  ST:



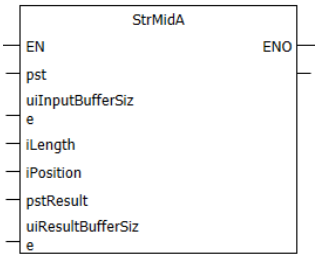
  LD:



18.5.3.1.15 StrMidA

This instruction extracts a string of specified length starting from a specified position in the source string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrMidA	Substring acquisition	FC		:= StrMidA(pst := , uiInputBufferSize := , iLength := , iPosition := , pstResult := , uiResultBufferSize :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pst	Pointer to the string to be extracted	Follows data type	-	Pointer to the string to be extracted.
	uiInputBufferSize	Buffer size pointed to by pst	Follows data type	-	Buffer size pointed to by pst.
	iLength	Caracter length	Follows data type	-	Character length.
	iPosition	Target position	Follows data type	-	Target position.
	pstResult	Return result	Follows data type	-	Return result.
	uiResultBufferSize	Buffer size for the return result	Follows data type	-	Buffer size for the return result.
Output	StrMidA	Return value	Follows data type	-	Return value.

Variable types

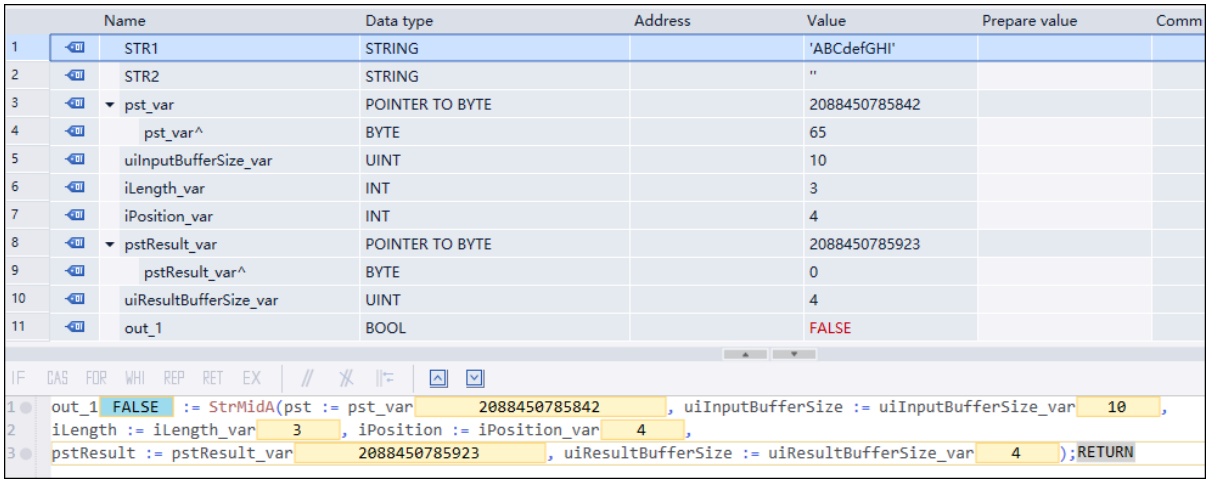
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pst	POINTER TO BYTE																			
uiInputBufferSize	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
iLength	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
iPosition	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
pstResult	POINTER TO BYTE																			
uiResultBufferSize	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
StrMidA	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction extracts a string of specified length from a specified position in the source string, and return the string of the specified length.
  - If iPosition is less than or equal to 0 or iLength is less than 0, the output will remain unchanged.
  - If the set position iPosition exceeds the length of the string, there will be no output.
  - If the substring of the set length iLength starting from the set position iPosition exceeds the length of the input string, then all characters from the source data starting at iPosition will be extracted.
  - If the length of the source string to be extracted exceeds the size of the set buffer uiResultBufferSize, only the data of the buffer size length will be truncated.
  - If the input is Chinese, it is recommended to use StrMidW.
  - The return value is currently unused and always remains 0.
- **Program example**

  STR1 is the string pointed to by the pst_var pointer, and STR2 is the string pointed to by the pstResult_var pointer.

  ST:

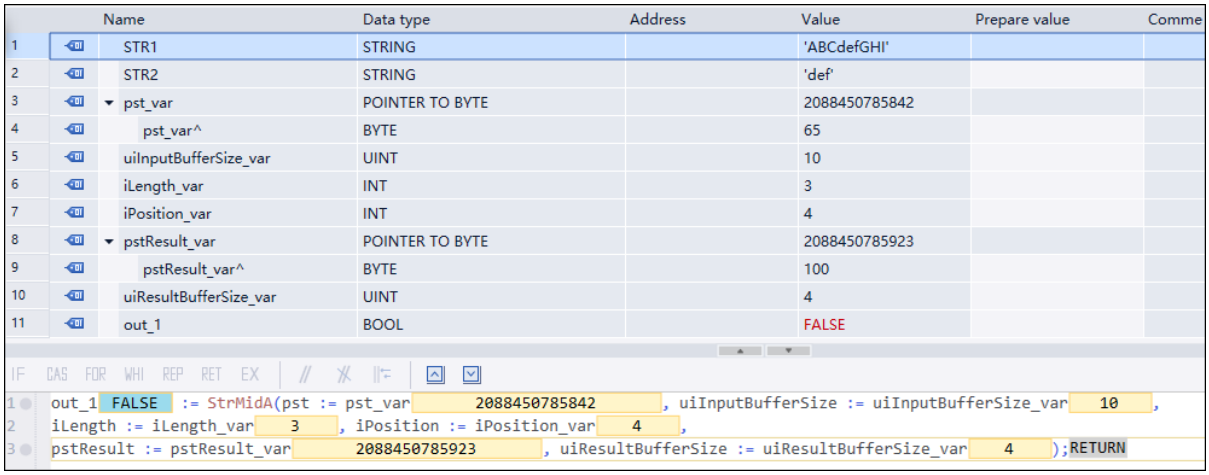
  (1) Original state



	Name	Data type	Address	Value	Prepare value	Comm
1	STR1	STRING		'ABCdefGHI'		
2	STR2	STRING		''		
3	pst_var	POINTER TO BYTE		2088450785842		
4	pst_var^	BYTE		65		
5	uiInputBufferSize_var	UINT		10		
6	iLength_var	INT		3		
7	iPosition_var	INT		4		
8	pstResult_var	POINTER TO BYTE		2088450785923		
9	pstResult_var^	BYTE		0		
10	uiResultBufferSize_var	UINT		4		
11	out_1	BOOL		FALSE		

```
1  out_1 FALSE := StrMidA(pst := pst_var 2088450785842, uiInputBufferSize := uiInputBufferSize_var 10,
2  iLength := iLength_var 3, iPosition := iPosition_var 4,
3  pstResult := pstResult_var 2088450785923, uiResultBufferSize := uiResultBufferSize_var 4 );RETURN
```

(2) Enable

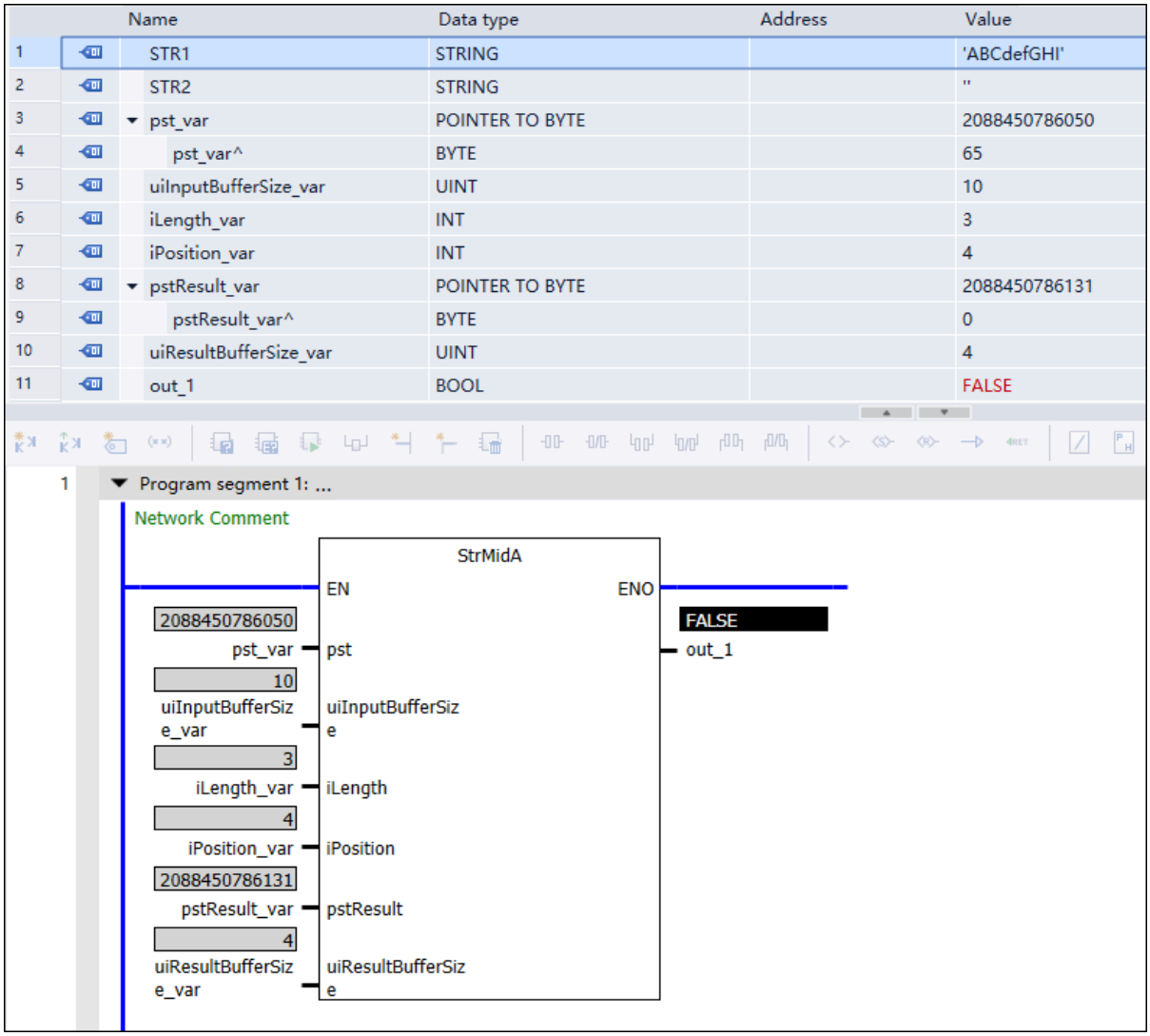


	Name	Data type	Address	Value	Prepare value	Comme
1	STR1	STRING		'ABCdefGHI'		
2	STR2	STRING		'def'		
3	pst_var	POINTER TO BYTE		2088450785842		
4	pst_var^	BYTE		65		
5	uiInputBufferSize_var	UINT		10		
6	iLength_var	INT		3		
7	iPosition_var	INT		4		
8	pstResult_var	POINTER TO BYTE		2088450785923		
9	pstResult_var^	BYTE		100		
10	uiResultBufferSize_var	UINT		4		
11	out_1	BOOL		FALSE		

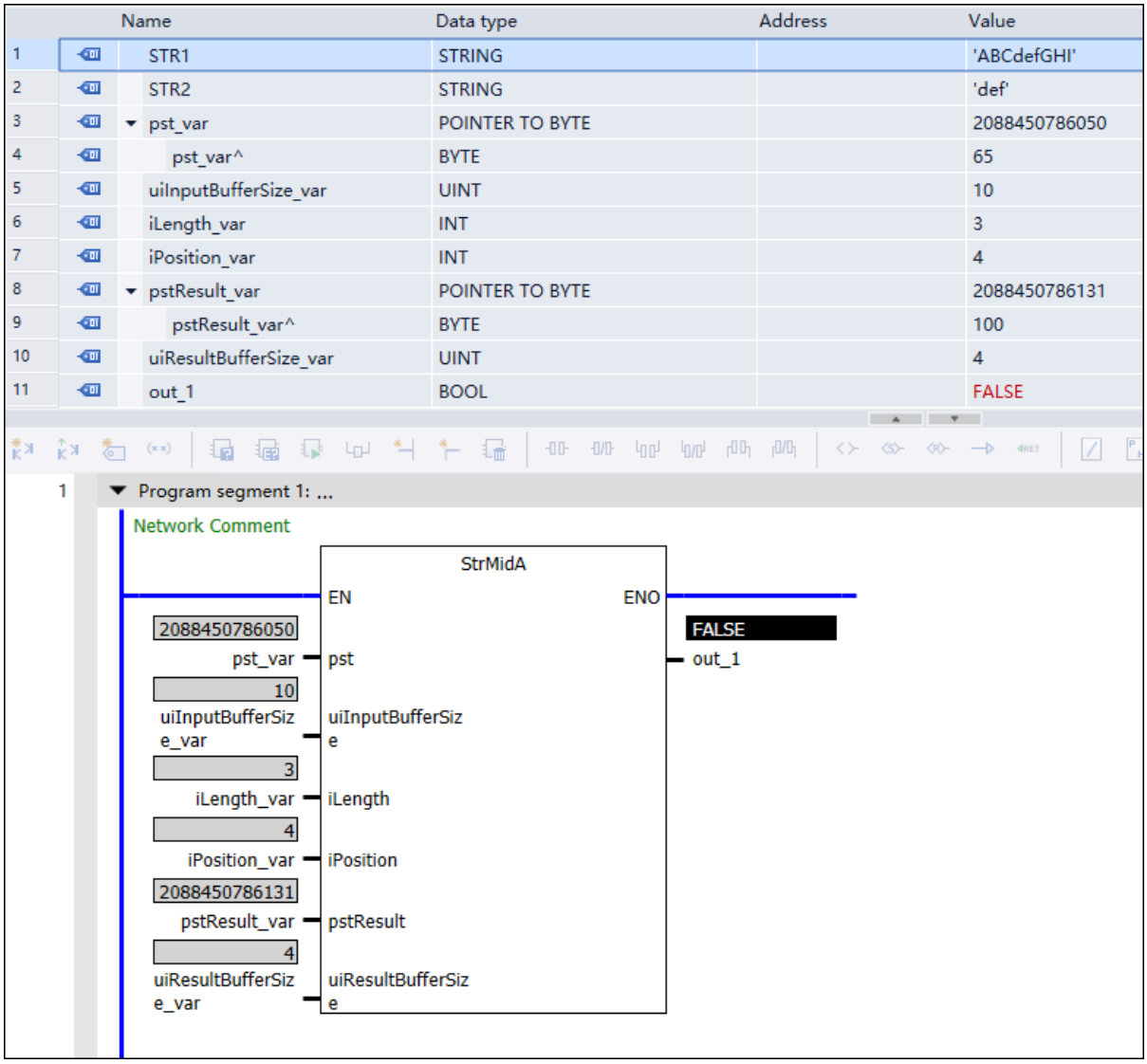
```
1  out_1 FALSE := StrMidA(pst := pst_var 2088450785842, uiInputBufferSize := uiInputBufferSize_var 10,
2  iLength := iLength_var 3, iPosition := iPosition_var 4,
3  pstResult := pstResult_var 2088450785923, uiResultBufferSize := uiResultBufferSize_var 4 );RETURN
```

LD:

(1) Original state


Name
Data type
Address
Value
1 STR1 STRING 'ABCdefGHI'
2 STR2 STRING ''
3 pst_var POINTER TO BYTE 2088450786050
4 pst_var^ BYTE 65
5 uiInputBufferSize_var UINT 10
6 iLength_var INT 3
7 iPosition_var INT 4
8 pstResult_var POINTER TO BYTE 2088450786131
9 pstResult_var^ BYTE 0
10 uiResultBufferSize_var UINT 4
11 out_1 BOOL FALSE
1
Program segment 1: ...
Network Comment
StrMidA
EN
ENO
2088450786050
pst_var
pst
FALSE
out_1
10
uiInputBufferSize_var
uiInputBufferSize
3
iLength_var
iLength
4
iPosition_var
iPosition
2088450786131
pstResult_var
pstResult
4
uiResultBufferSize_var
uiResultBufferSize

(2) Enable



18.5.3.1.16 StrReplaceA

This instruction replaces the string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrReplaceA	String replacement	FC	StrReplaceA EN ENO pstInput uiInputBufferSize pstReplaceWith iLengthInput iLengthToReplace iLengthToReplaceWith iPosition	:= StrReplaceA(pstInput := , uiInputBufferSize := , pstReplaceWith := , iLengthInput := , iLengthToReplace := , iLengthToReplaceWith := , iPosition :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pstInput	Pointer to the string to be replaced	Follows data type	-	Pointer to the string to be replaced.
	uiInputBufferSize	Buffer size pointed to by pstInput	Follows data type	-	Buffer size pointed to by pstInput.
	pstReplaceWith	Pointer to the replacement string	Follows data type	-	Pointer to the replacement string.
	iLengthInput	Length of input character	Follows data type	-	Character length
	iLengthToReplace	Target position	Follows data type	-	Target position.
	iLengthToReplaceWith	The length of the string that the pstReplaceWith pointer points to	Follows data type	-	The length of the string that the pstReplaceWith pointer points to.
	iPosition	The target position in the string that the pstInput pointer points to	Follows data type	-	The target position in the string that the pstInput pointer points to.
Output	StrReplaceA	Return value	Follows data type	-	Return value.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pstInput	POINTER TO STRING[255]																			
uiInputBufferSize	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pstReplaceWith	POINTER TO STRING[255]																			
iLengthInput	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
iLengthToReplace	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
iLengthToReplaceWith	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
iPosition	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
StrReplaceA	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction replaces a part of the original string with the target string.
  - If iPosition is less than or equal to 0, or iLengthInput is less than 0, the output will remain unchanged.
  - If the set position iPosition exceeds the length of the string, there will be no output.
  - If the substring selected from the set position iPosition with the set length iLengthInput exceeds the remaining length of the input string, the function is equivalent to extracting characters to the right of the specified position in the source string.
  - If the input is Chinese, it is recommended to use StrReplaceW.
  - The return value of the function is 0.
- **Program example**

  pstStr is the string pointed to by the pointer pstInput_var, and pstDst is the string pointed to by the pointer pstReplaceWith_var.

  ST:

  (1) Original state

	Name	Data type	Address	Value
1	pstStr	STRING		'ABCHIJK'
2	pstDst	STRING		'DEFG'
3	pstInput_var	POINTER TO STRING[255]		3016477290
4	pstInput_var^	STRING[255]		'ABCHIJK'
5	uiInputBufferSize_var	UINT		8
6	pstReplaceWith_var	POINTER TO STRING[255]		3016477371
7	pstReplaceWith_var^	STRING[255]		'DEFG'
8	iLengthInput_var	INT		8
9	iLengthToReplace_var	INT		4
10	iLengthToReplaceWith_var	INT		4
11	iPosition_var	INT		4
12	out_1	BOOL		FALSE

```
1  out_1 FALSE := StrReplaceA(
2          pstInput := pstInput_var 3016477290 ,
3          uiInputBufferSize := uiInputBufferSize_var 8 ,
4          pstReplaceWith := pstReplaceWith_var 3016477371 ,
5          iLengthInput := iLengthInput_var 8 ,
6          iLengthToReplace := iLengthToReplace_var 4 ,
7          iLengthToReplaceWith := iLengthToReplaceWith_var 4 ,
8          iPosition := iPosition_var 4 );RETURN
```

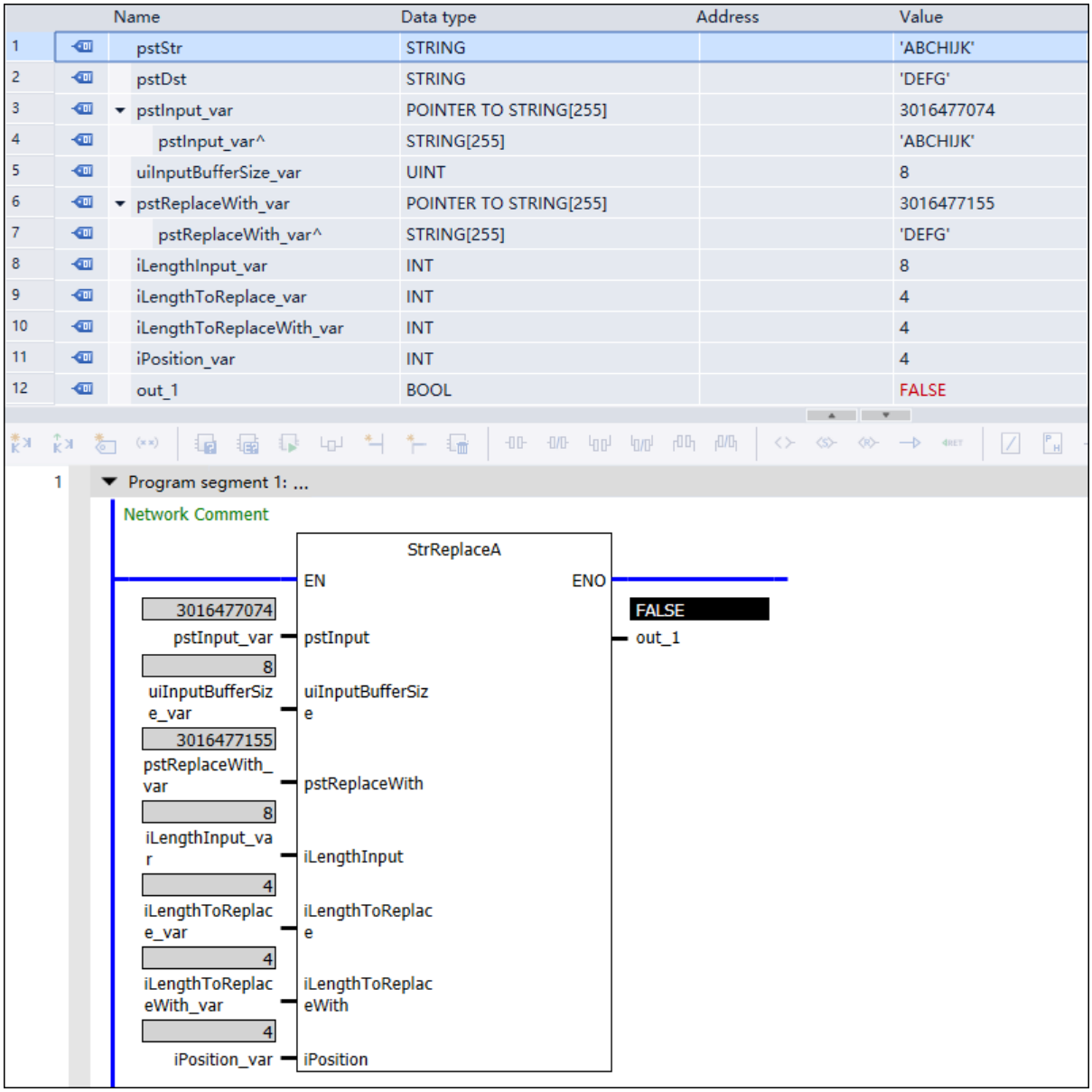
(2) Enable

	Name	Data type	Address	Value
1	pstStr	STRING		'ABCDEFG'
2	pstDst	STRING		'DEFG'
3	pstInput_var	POINTER TO STRING[255]		3016477290
4	pstInput_var^	STRING[255]		'ABCDEFG'
5	uiInputBufferSize_var	UINT		8
6	pstReplaceWith_var	POINTER TO STRING[255]		3016477371
7	pstReplaceWith_var^	STRING[255]		'DEFG'
8	iLengthInput_var	INT		8
9	iLengthToReplace_var	INT		4
10	iLengthToReplaceWith_var	INT		4
11	iPosition_var	INT		4
12	out_1	BOOL		FALSE

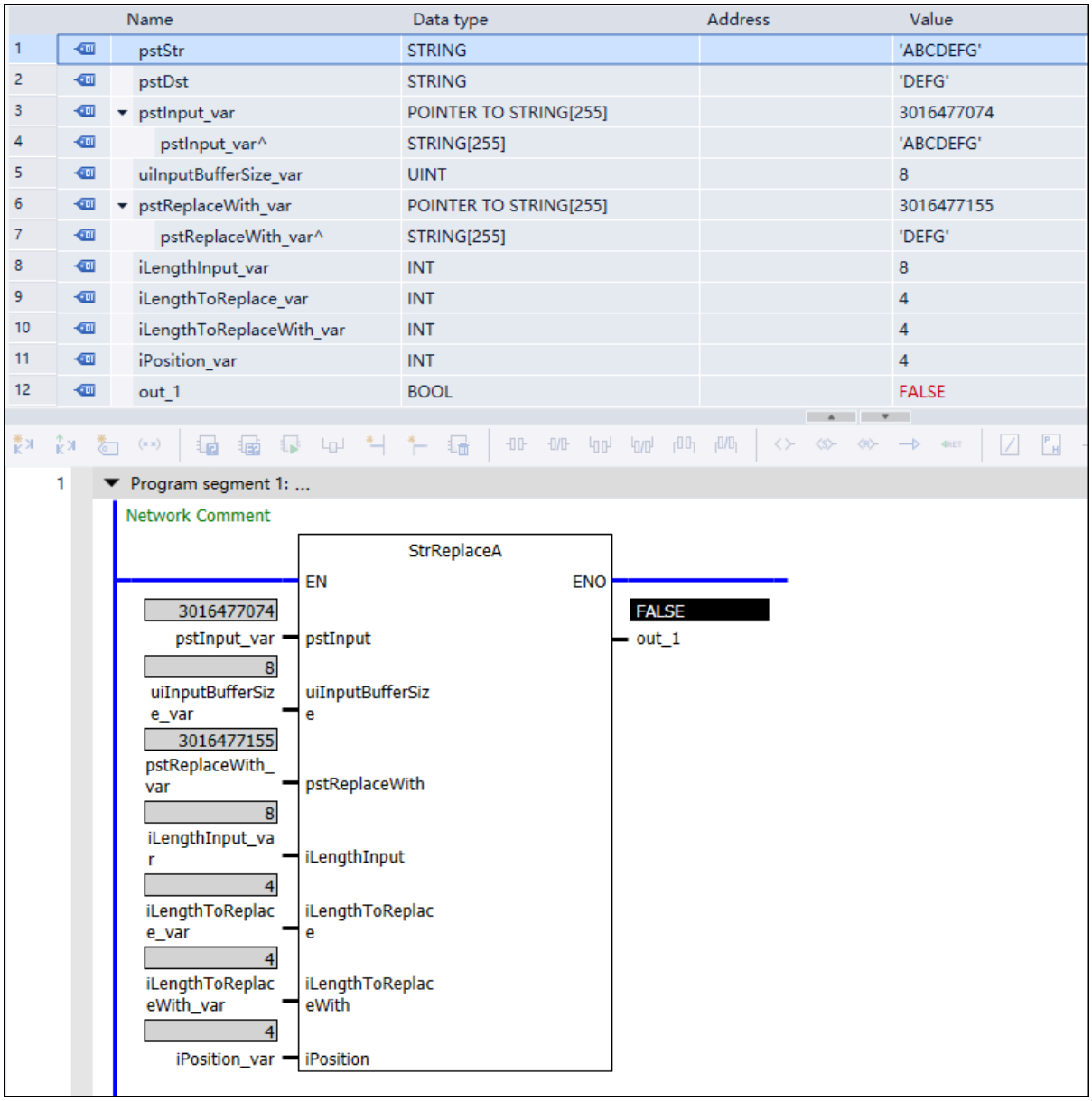
```
1  out_1 FALSE := StrReplaceA(
2          pstInput := pstInput_var 3016477290 ,
3          uiInputBufferSize := uiInputBufferSize_var 8 ,
4          pstReplaceWith := pstReplaceWith_var 3016477371 ,
5          iLengthInput := iLengthInput_var 8 ,
6          iLengthToReplace := iLengthToReplace_var 4 ,
7          iLengthToReplaceWith := iLengthToReplaceWith_var 4 ,
8          iPosition := iPosition_var 4 );RETURN
```

LD:

(1) Original state


Name
Data type
Address
Value
pstStr
STRING
'ABCHIJK'
pstDst
STRING
'DEFG'
pstInput_var
POINTER TO STRING[255]
3016477074
pstInput_var^
STRING[255]
'ABCHIJK'
uiInputBufferSize_var
UINT
8
pstReplaceWith_var
POINTER TO STRING[255]
3016477155
pstReplaceWith_var^
STRING[255]
'DEFG'
iLengthInput_var
INT
8
iLengthToReplace_var
INT
4
iLengthToReplaceWith_var
INT
4
iPosition_var
INT
4
out_1
BOOL
FALSE
Program segment 1: ...
Network Comment
StrReplaceA
EN
ENO
3016477074
pstInput_var
pstInput
FALSE
out_1
8
uiInputBufferSize_var
uiInputBufferSize
3016477155
pstReplaceWith_var
pstReplaceWith
8
iLengthInput_var
iLengthInput
4
iLengthToReplace_var
iLengthToReplace
4
iLengthToReplaceWith_var
iLengthToReplaceWith
4
iPosition_var
iPosition

(2) Enable



18.5.3.1.17 StrToLowerA

This instruction converts the string to lowercase.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrToLowerA	String lowercase conversion	FC	StrToLowerA EN ENO pString	:= StrToLowerA(pString :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pString	String to be converted	Follows data type	-	Character to be converted.

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	StrToLowerA	Return value	Follows data type	-	Converted character.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pString	CharBufferPtr: POINTER TO BYTE																			
CharToUpper	√	-	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction converts all characters in the pString to lowercase.

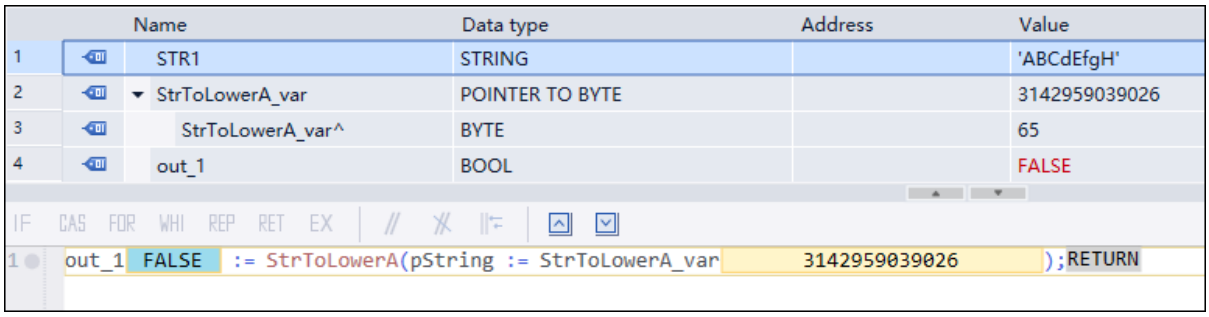
- **Precautions**
  - The string to be converted must not be NULL.
  - The return value of the function is 0.

- **Program example**

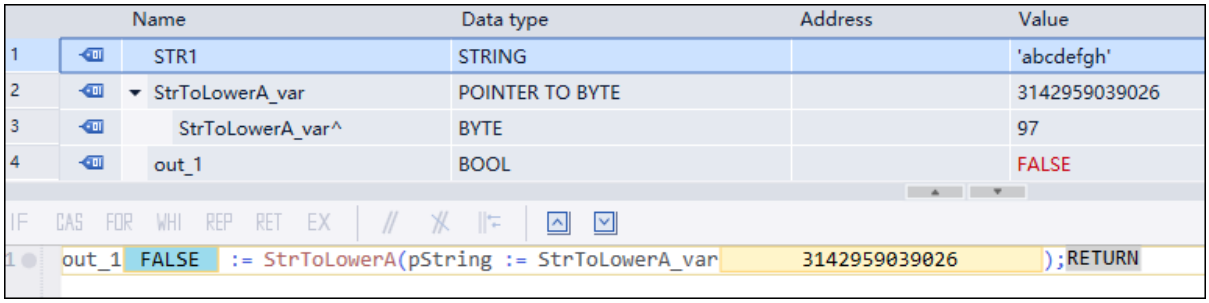
  STR1 is the string pointed to by the StrToLowerA_var pointer.

  ST:

  (1) Original state

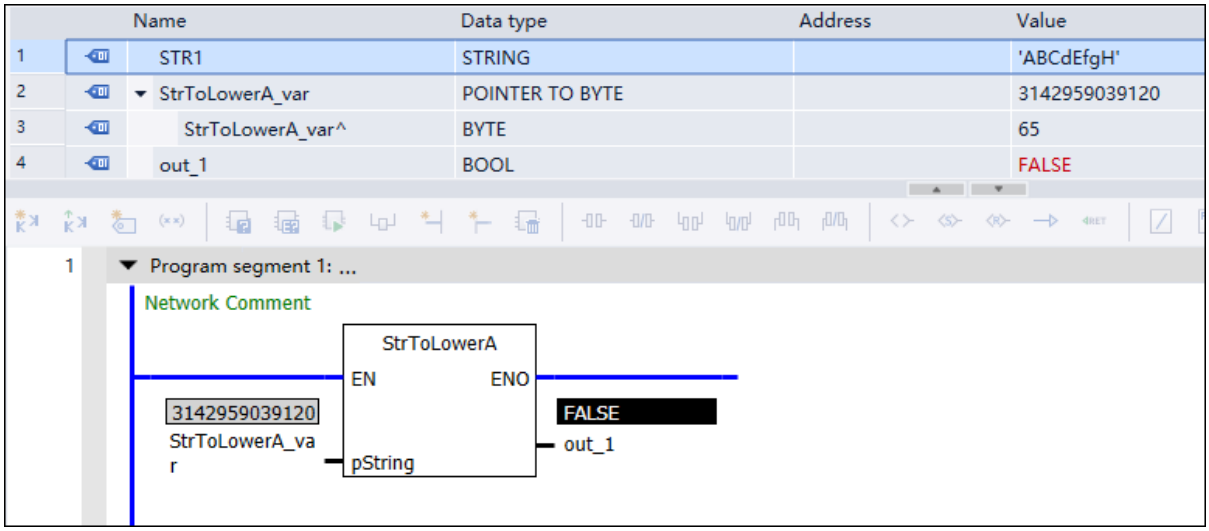


  (2) Enable

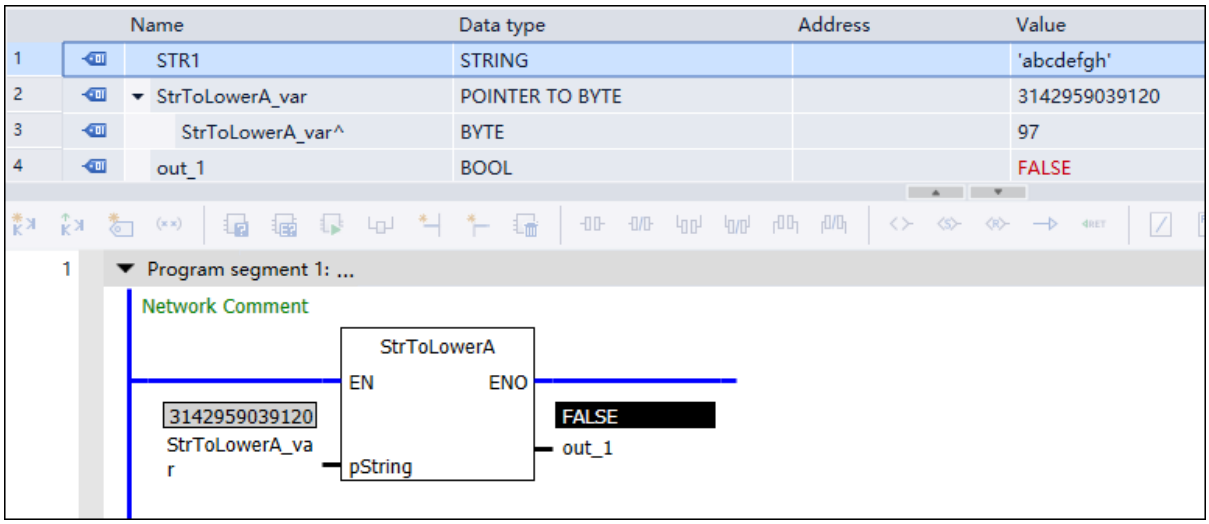


  LD:

(1) Original state



(2) Enable



18.5.3.1.18 StrToUpperA

This instruction converts the string to uppercase.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrToUpperA	String uppercase conversion	FC	StrToUpperA EN ENO pString	:= StrToUpperA(pString :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pString	String to be converted	Follows data type	-	Character to be converted.

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	StrToUpperA	Return value	Follows data type	-	Converted character.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pString	CharBufferPtr: POINTER TO BYTE																			
StrToUpperA	√	-	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction converts all characters in the pString to uppercase.

- **Precautions**
  - The string to be converted must not be NULL.
  - The return value of the function is 0.

- **Program example**

  STR1 is the string pointed to by the StrToUpperA_var pointer.

  ST:

  (1) Original state

	Name	Data type	Address	Value
1	STR1	STRING		'abcdEfgH'
2	StrToUpperA_var	POINTER TO BYTE		3142959039026
3	StrToUpperA_var^	BYTE		97
4	out_1	BOOL		FALSE

```
1  out_1 FALSE := StrToUpperA(pString := StrToUpperA_var 3142959039026 );RETURN
```

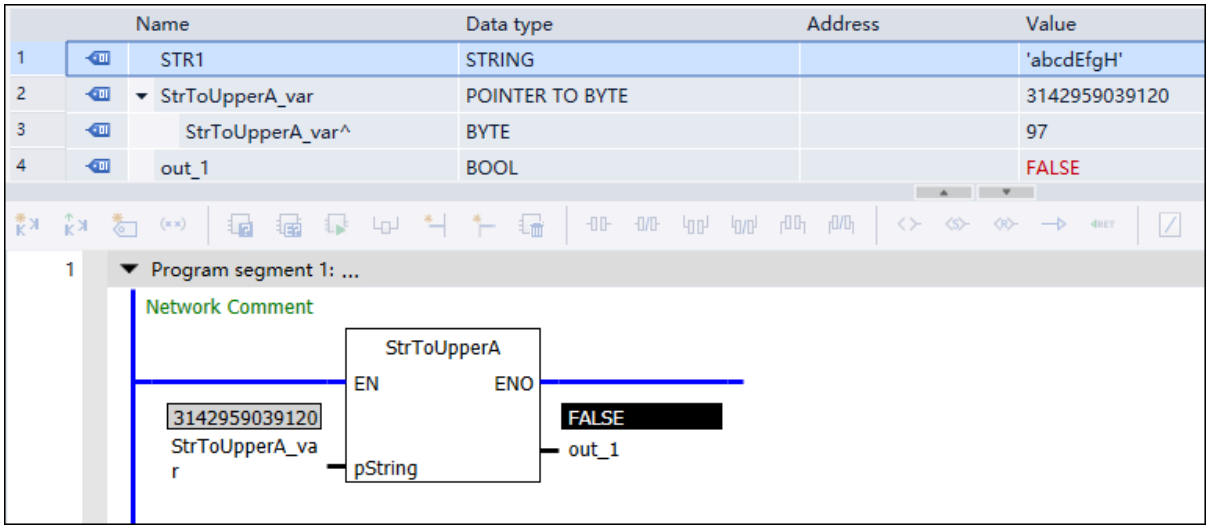
  (2) Enable

	Name	Data type	Address	Value
1	STR1	STRING		'ABCDEFGH'
2	StrToUpperA_var	POINTER TO BYTE		3142959039026
3	StrToUpperA_var^	BYTE		65
4	out_1	BOOL		FALSE

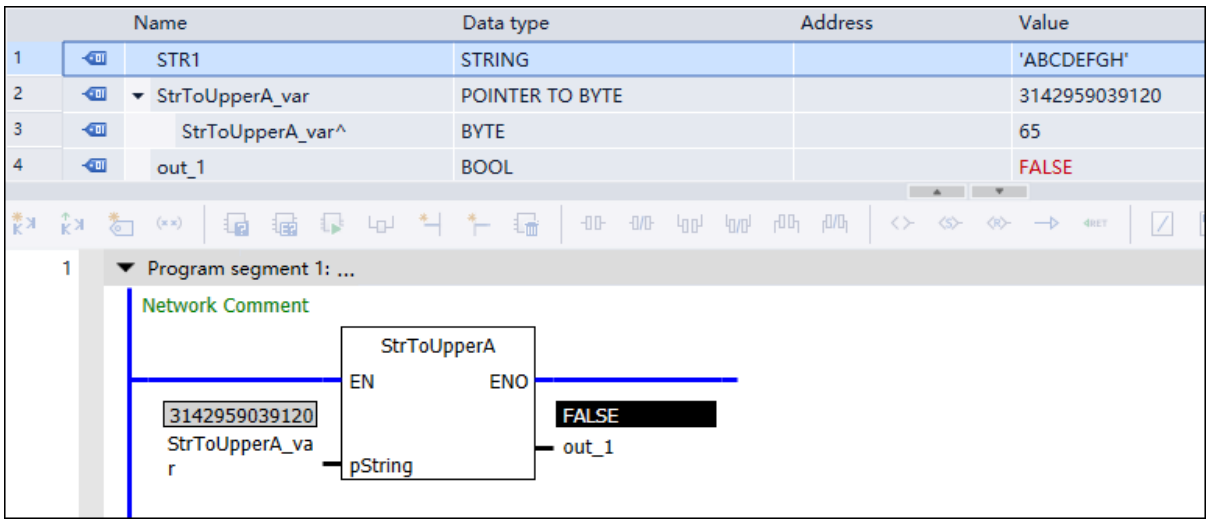
```
1  out_1 FALSE := StrToUpperA(pString := StrToUpperA_var 3142959039026 );RETURN
```

  LD:

(1) Original state



(2) Enable



18.5.3.1.19 StrTrimA

This instruction trims the leading and trailing spaces from the string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrTrimA	Trim the leading and trailing spaces from the string	FC	StrTrimA EN ENO pString	:= StrTrimA(pString :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pString	String to be converted	Follows data type	-	Character to be converted.

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	StrTrimA	Return value	Follows data type	-	Converted character.

Variable types

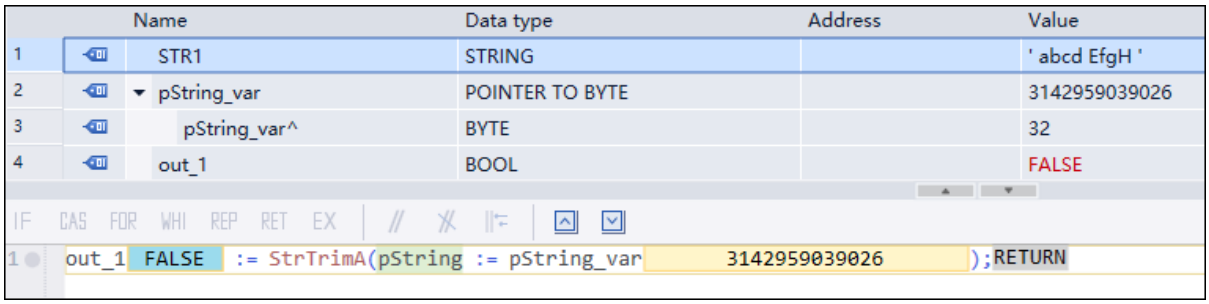
	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pString	POINTER TO BYTE																			
StrTrimA	√	-	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction trims the leading and trailing spaces from the string.
  - The return value of the function is always 0.
- **Program example**

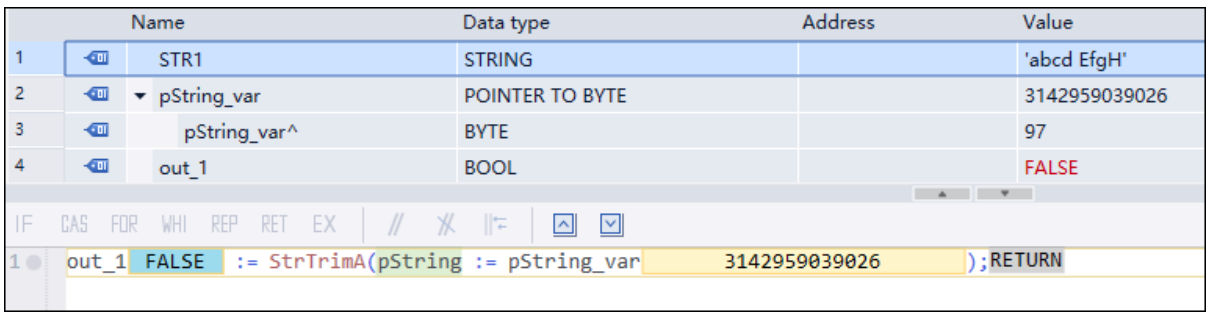
  STR1 is the string pointed to by the pString_var pointer.

  ST:

  (1) Original state

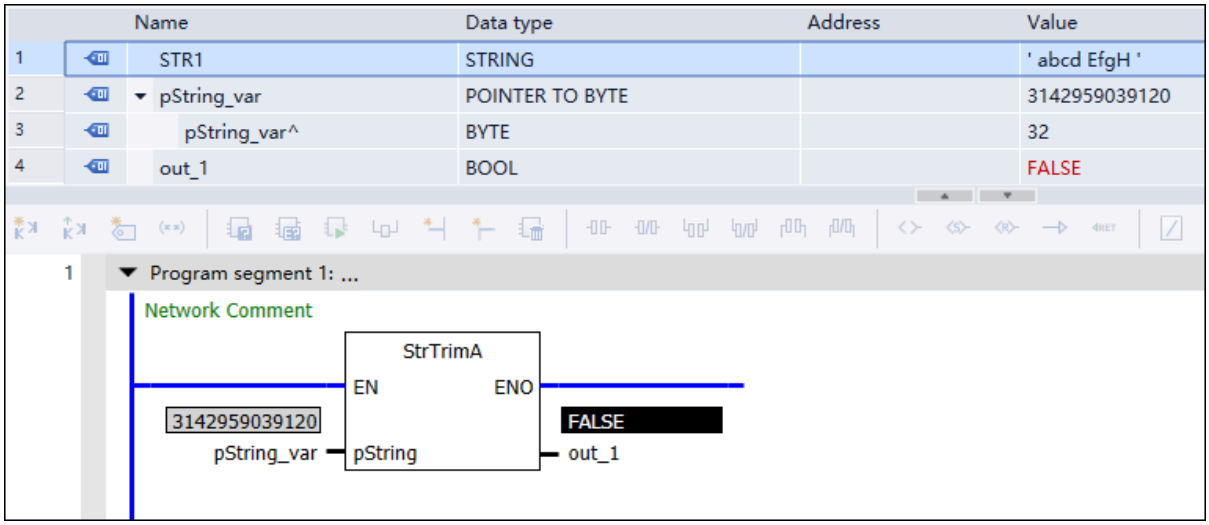


  (2) Enable

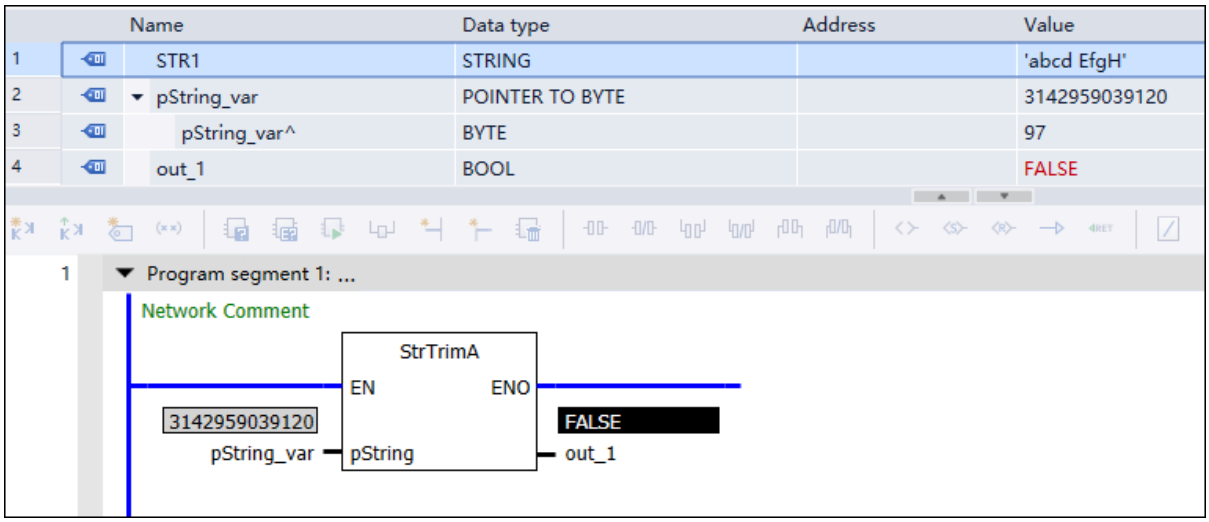


  LD:

  (1) Original state



(2) Enable



18.5.3.1.20 StrTrimEndA

This instruction trims the trailing spaces from the string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrTrimEndA	Trim the trailing spaces from the string	FC	StrTrimEndA EN ENO pString	:= StrTrimEndA(pString :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pString	String to be converted	Follows data type	-	Character to be converted.
Output	StrTrimEndA	Return value	Follows data type	-	Return value.

Variable types

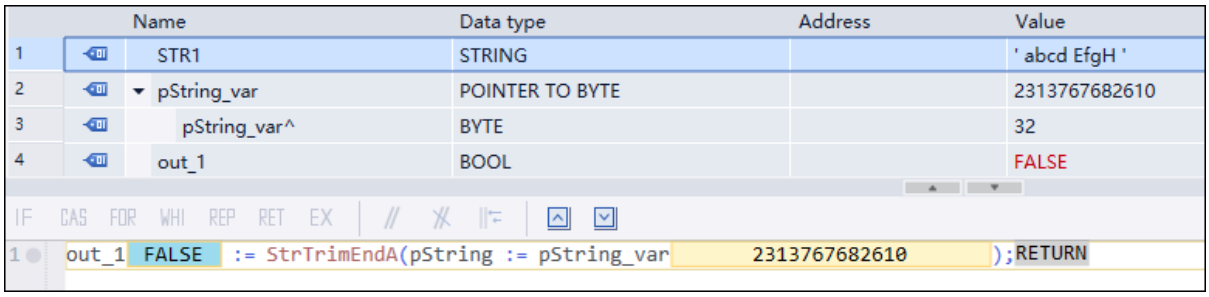
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pString	POINTER TO BYTE																			
StrTrimEndA	√	-	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction trims trailing spaces from the string.
  - The return value of the function is always 0.
- **Program example**

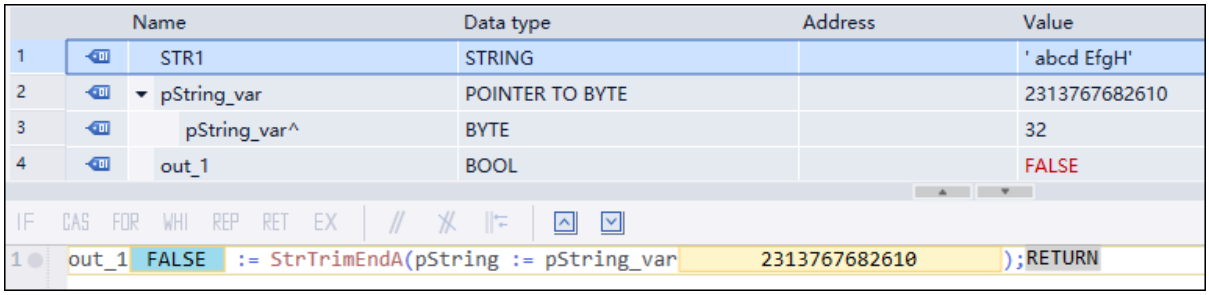
  STR1 is the string pointed to by the pString_var pointer.

  ST:

  (1) Original state

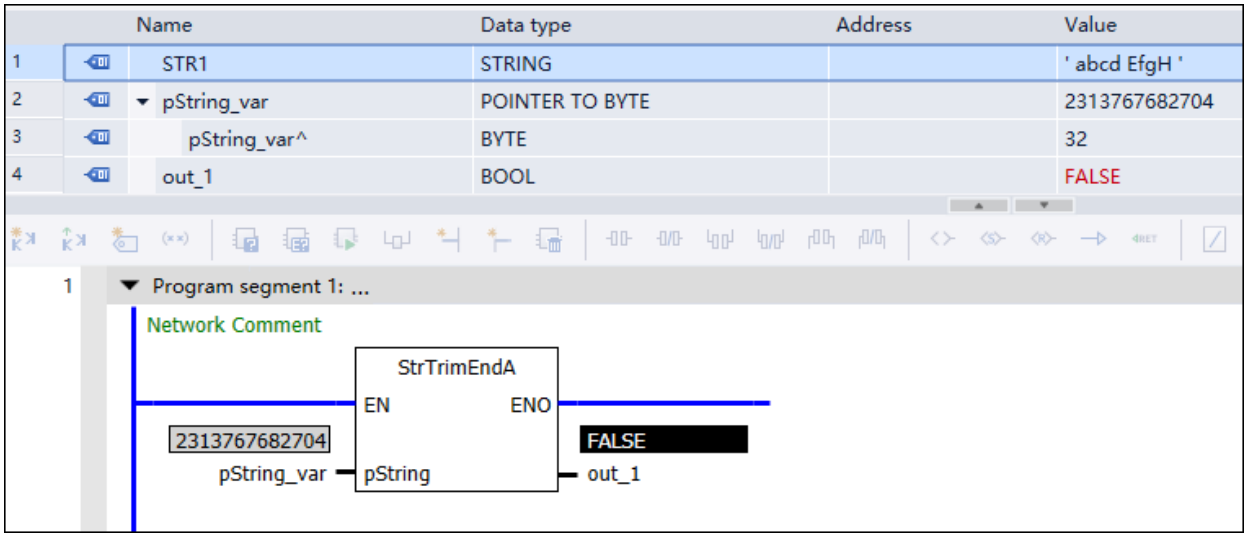


  (2) Enable

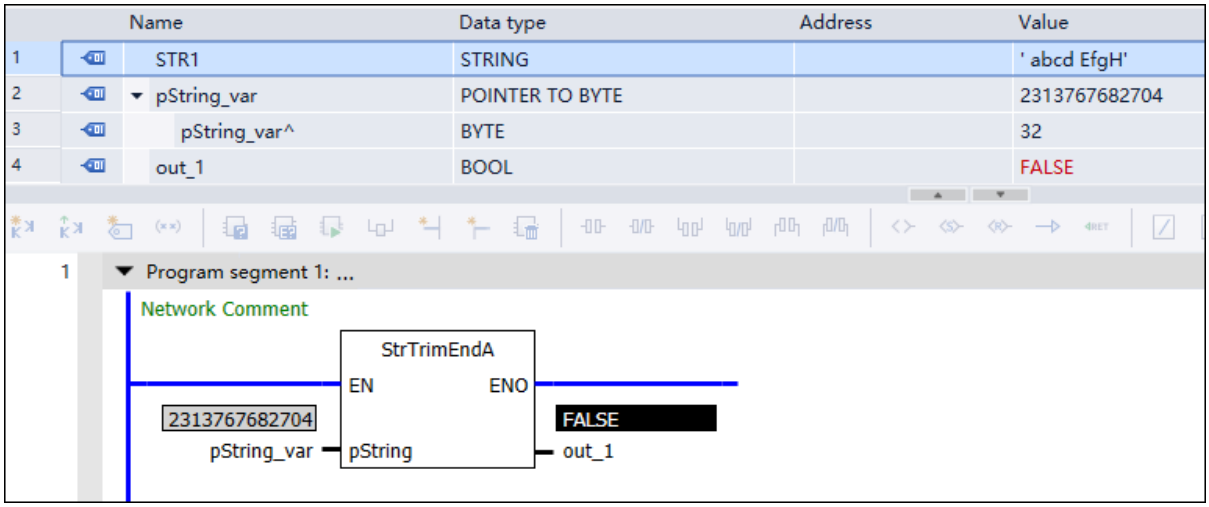


  LD:

  (1) Original state



(2) Enable



18.5.3.1.21 StrTrimStartA

This instruction trims leading spaces from the string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrTrimStartA	Trim leading spaces from the string	FC	StrTrimStartA EN ENO pString	:= StrTrimStartA(pString :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pString	String to be converted	Follows data type	-	Character to be converted.
Output	StrTrimStartA	Return value	Follows data type	-	Return value.

Variable types

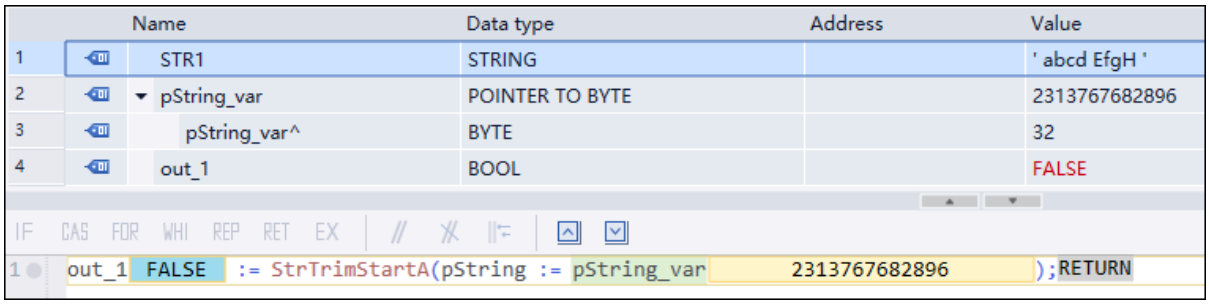
	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pString	POINTER TO BYTE																			
StrTrimStartA	√	-	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**
  - This instruction trims leading spaces from the string.
  - The return value of the function is always 0.
- **Program example**

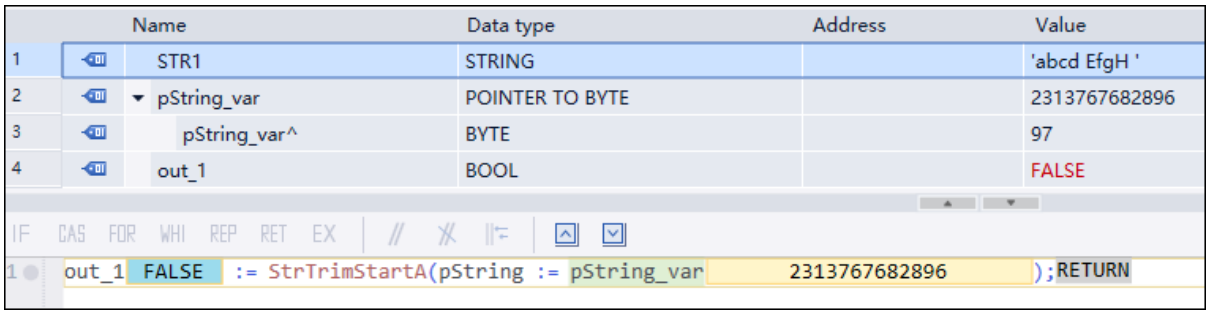
  STR1 is the string pointed to by the pString_var pointer.

  ST:

  (1) Original state

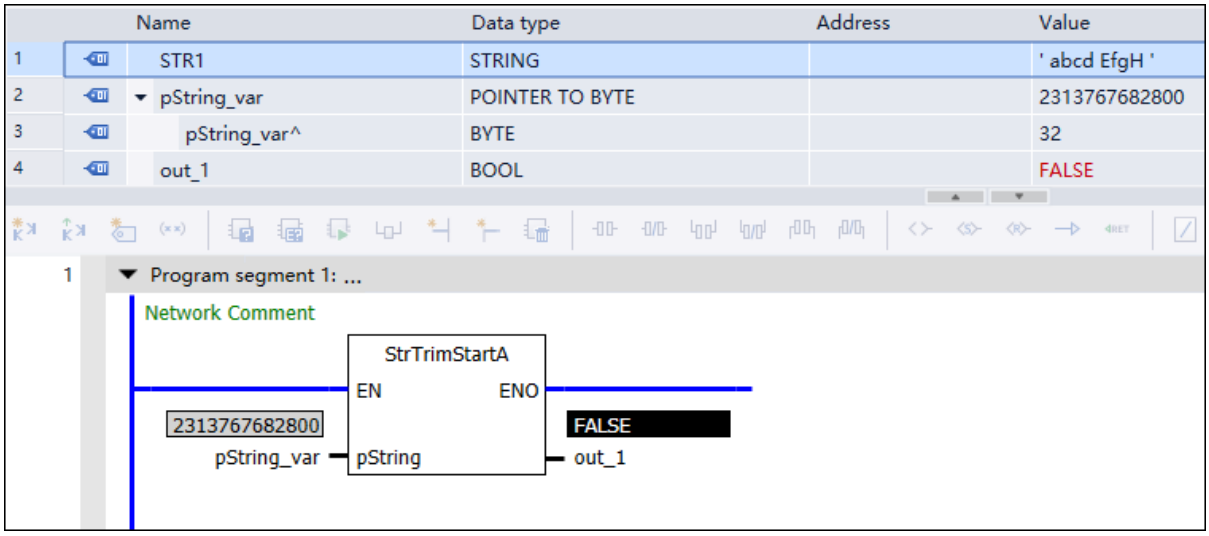


  (2) Enable

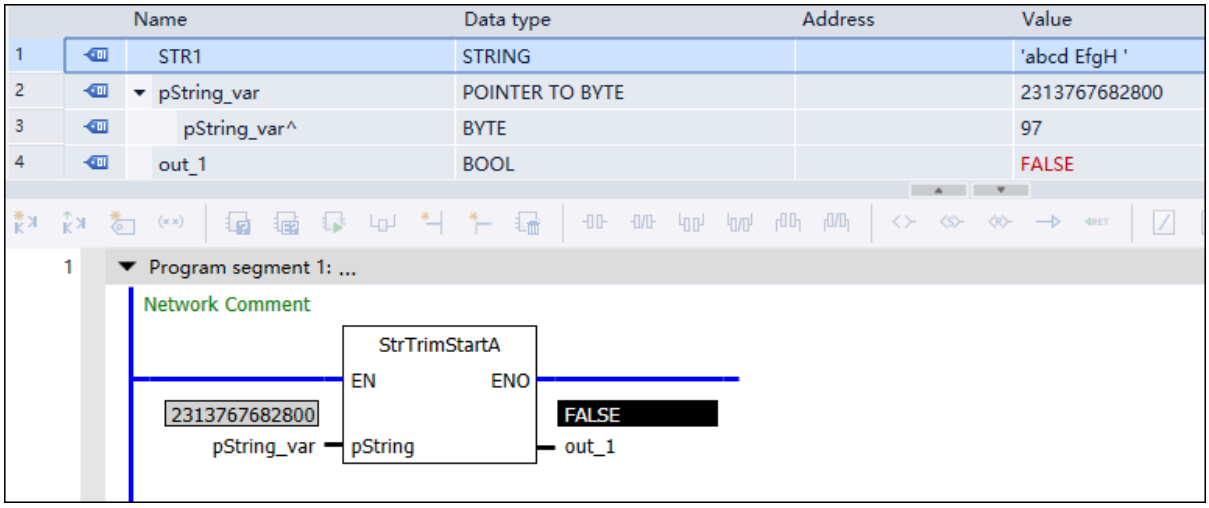


  LD:

  (1) Original state



(2) Enable



18.5.3.1.22 CharToUpper

This instruction converts character to uppercase.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
CharToUpper	Character uppercase conversion	FC	CharToUpper EN ENO byChar	:= CharToUpper(byChar :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	byChar	Character to be converted	Follows data type	-	Character to be converted.
Output	CharToUpper	Return value	Follows data type	-	Converted character.

- **Variable types**

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
byChar	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
CharToUpper	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

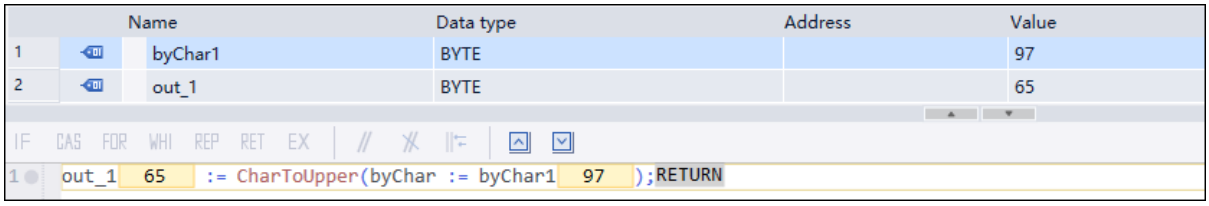
  This instruction converts the character byChar to uppercase and output it as the BYTE type.

- **Precautions**

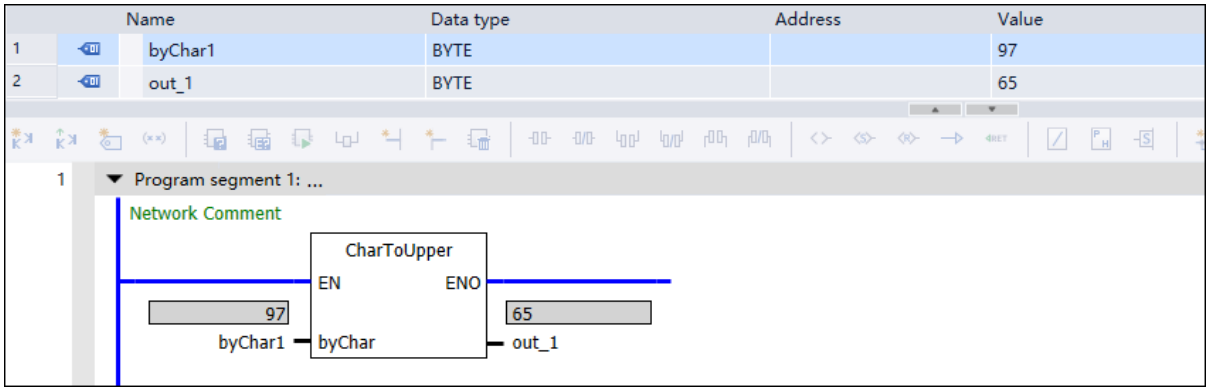
  When the ASCII value of the character to be converted lies between 'a' and 'z', the algorithm performs the conversion; otherwise, the return value remains the ASCII code value of the original character.

- **Program example**

  ST:



  LD:



18.5.3.2 Unicode

18.5.3.2.1 Instruction List

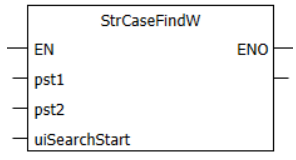
Instruction	FB/FC	Function	Library	Applicable Library Version
18.5.3.2.2 StrCaseFindW	FC	Wide string find (case-insensitive)	StringUtils	Library version 1.0.0.0 and later

Instruction	FB/FC	Function	Library	Applicable Library Version
18.5.3.2.3 StrCmpW	FC	Wide string comparison (case-sensitive)		Library version 1.0.0.0 and later
18.5.3.2.4 StrConcatW	FC	Wide string concatenation		Library version 1.0.0.0 and later
18.5.3.2.5 StrCpyW	FC	Wide string copy		Library version 1.0.0.0 and later
18.5.3.2.6 StrDeleteW	FC	Wide string deletion		Library version 1.0.0.0 and later
18.5.3.2.7 StrFindW	FC	Wide string find (case-sensitive)		Library version 1.0.0.0 and later
18.5.3.2.8 StrIsNullOrEmptyW	FC	Empty string judgement		Library version 1.0.0.0 and later
18.5.3.2.9 StrLenW	FC	Calculation of wide string length		Library version 1.0.0.0 and later
18.5.3.2.10 StrMidW	FC	Substring acquisition		Library version 1.0.0.0 and later
18.5.3.2.11 StrReplaceW	FC	Wide string replacement		Library version 1.0.0.0 and later
18.5.3.2.12 WCharToUpper	FC	Character uppercase conversion		Library version 1.0.0.0 and later

18.5.3.2.2 StrCaseFindW

This instruction performs a case-insensitive find operation on two wide strings.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrCaseFindW	Wide string find (case-insensitive)	FC		:= StrCaseFindW(pst1 := , pst2 := , uiSearchStart :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pst1	Pointer to the string to be checked	Follows data type	-	Pointer to the string to be checked.
	pst2	Pointer to the string to be found	Follows data type	-	Pointer to the string to be found.

Input/Output	Variable	Name	Valid Range	Default Value	Description
	uiSearchStart	Search start position	Follows data type	-	Search start position.
Output	StrCaseFindW	Return value	Follows data type	-	Result.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	Largeword	USINT	UINT	UDINT	Unsigned long integer	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pst1	POINTER TO WSTRING[255]																			
pst2	POINTER TO WSTRING[255]																			
uiSearchStart	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
StrCaseFindW	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction performs a case-insensitive find operation on two strings starting from the uiSearchStart-th position of the STR 1. If the STR 2 appears in the STR 1, return the position of the first character of STR2 within STR 1. The results are as follows.

Return Value	Comparison Result
0	The string to be found does not exist.
-1	Errors arise from invalid strings, i.e., there is a null pointer.
>=1	The position of the first character of STR 2 within STR 1, where the first character is counted as position 1.

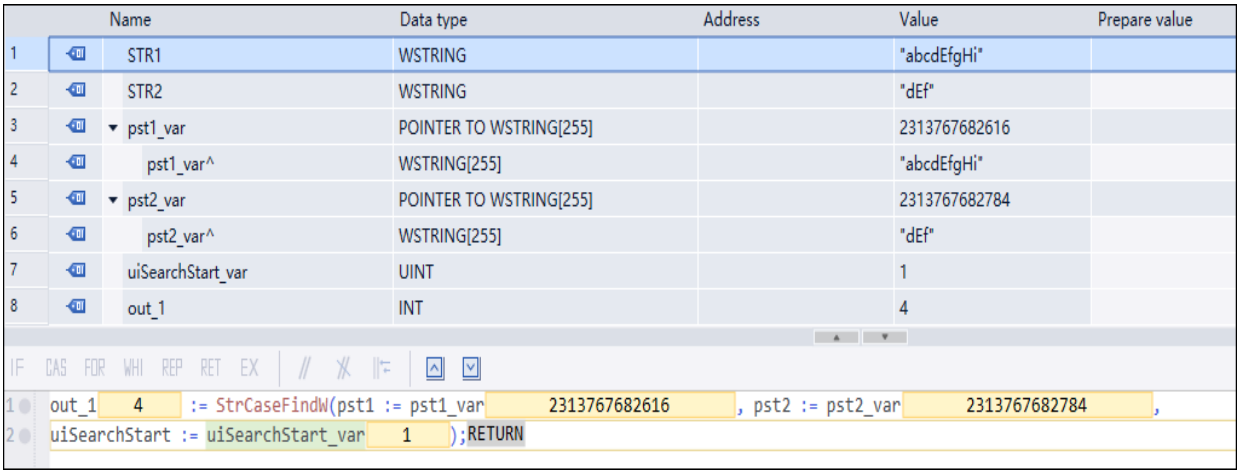
- **Precautions**

When the search position uiSearchStart is 0 or 1, the search starts from the first character of STR1.

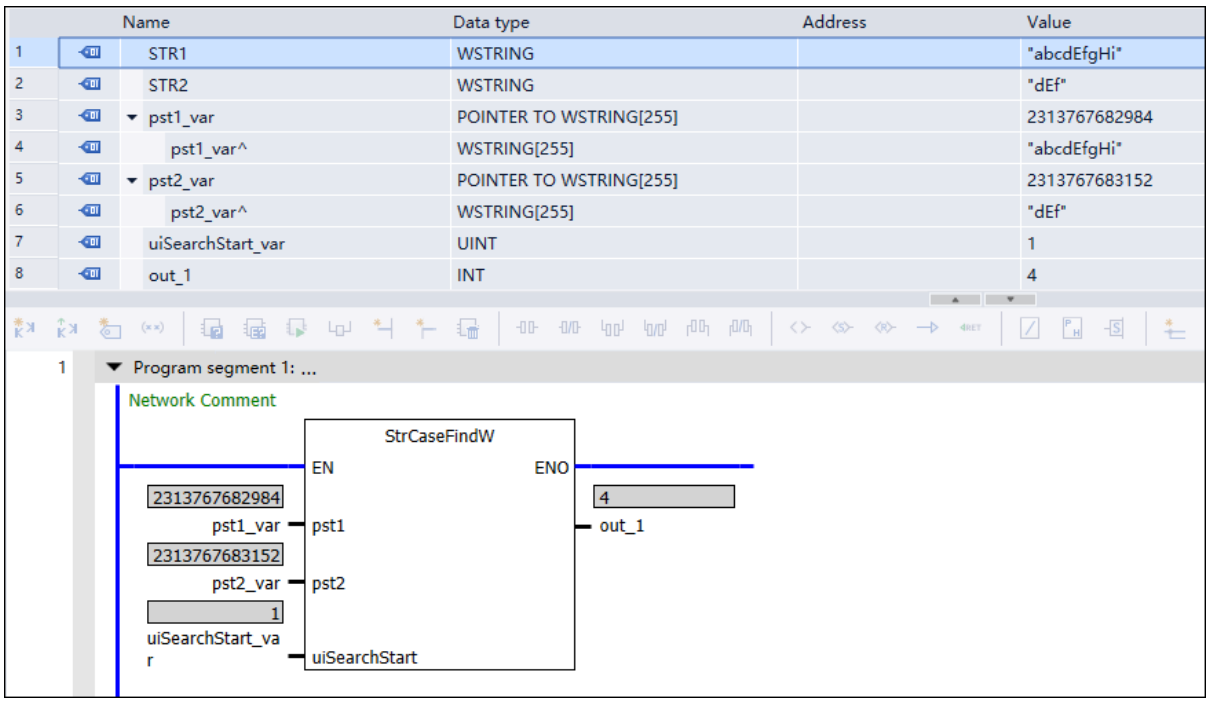
- **Program example**

  STR1 is the string pointed to by the pst1_var pointer, and STR2 is the string pointed to by the pst2_var pointer.

  ST:



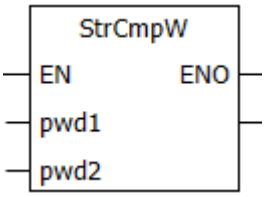
  LD:



18.5.3.2.3 StrCmpW

This instruction compares two wide strings in a case-sensitive manner.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrCmpW	Wide string comparison (case-sensitive)	FC		:= StrCmpW(pwd1 := , pwd2 :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pwd1	Comparison STR 1	Follows data type	-	Pointer to the comparison STR 1.
	pwd2	Comparison STR 2	Follows data type	-	Pointer to the comparison STR 2.
Output	StrCmpW	Return value	Follows data type	-	Comparison result.

  Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pwd1	POINTER TO WORD																			
pwd2	POINTER TO WORD																			
StrCmpW	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

- **Function**

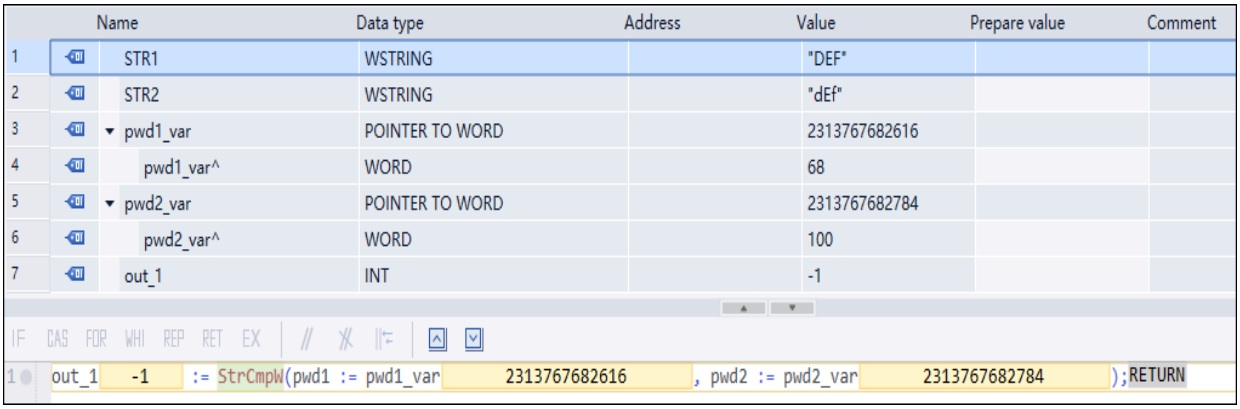
  This instruction compares two strings in a case-sensitive manner. The comparison results are as follows.

Return Value	Comparison Result
0	The two strings are equal.
1	pwd1 is greater than pwd2.
-1	pwd1 is less than pwd2.

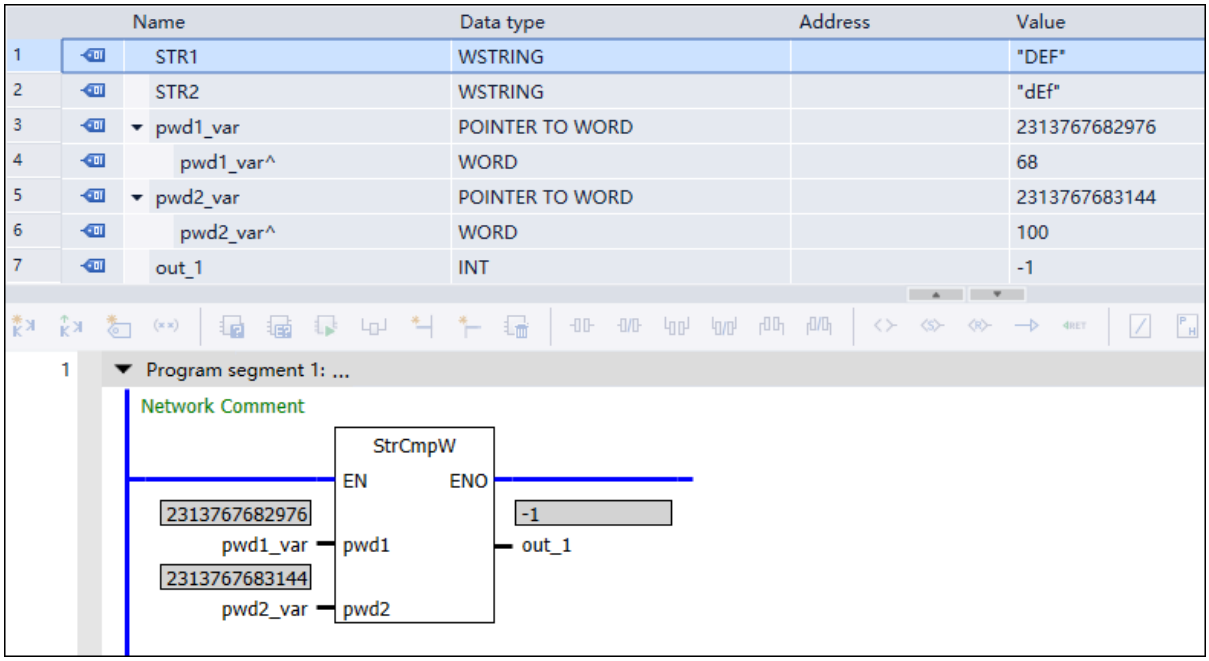
- **Program example**

STR1 is the string pointed to by the pwd1_var pointer, and STR2 is the string pointed to by the pwd2_var pointer.

ST:



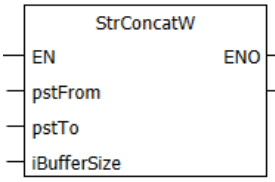
LD:



18.5.3.2.4 StrConcatW

This instruction concatenates two wide strings.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrConcatW	Wide string concatenation	FC		:= StrConcatW(pstFrom := , pstTo := , iBufferSize :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pstFrom	Pointer to byte	Follows data type	-	Pointer to the bytes to be concatenated.
	pstTo	Pointer to byte	Follows data type	-	Pointer to the bytes to be appended.
	iBufferSize	Buffer size	Follows data type	-	Buffer size of "pstTo".
Output	StrConcatW	Return value	Follows data type	-	Operation result.

Variable types

	Boolean	Bit String				Integer							Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pstFrom	CharBufferPtr: POINTER TO BYTE																			
pstTo	CharBufferPtr: POINTER TO BYTE																			
iBufferSize	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
StrConcatW	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction concatenates the string pointed to by pstFrom to the end of pstTo. The possible scenarios are as follows.

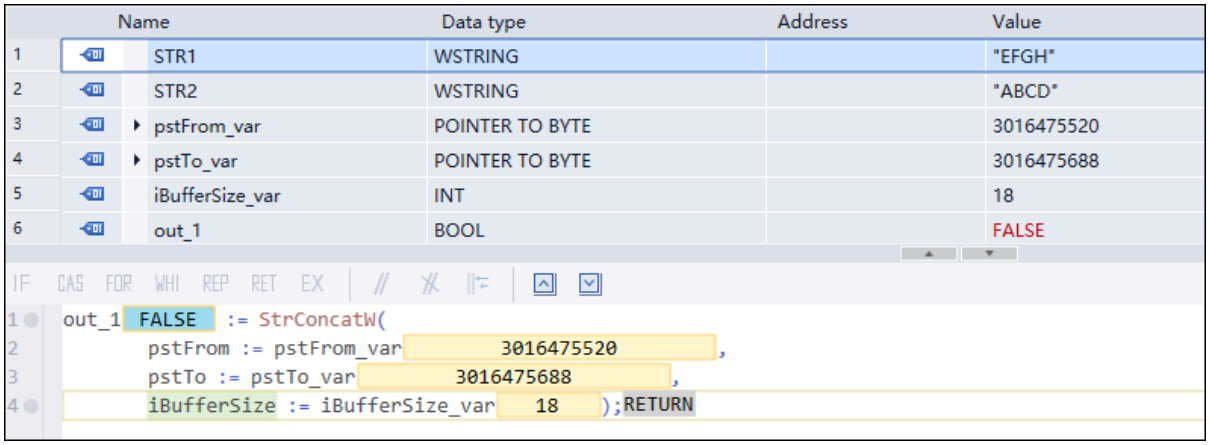
Return Value	Comparison Result
FALSE	The length of the concatenated string exceeds iBufferSize, the string pointed to by pstTo remains unchanged.
	Invalid string (i.e., there is a null pointer); the string pointed to by pstTo remains unchanged.
TRUE	If the result is normal, assign the concatenated string to the string pointed to by pstTo.

- **Program example**

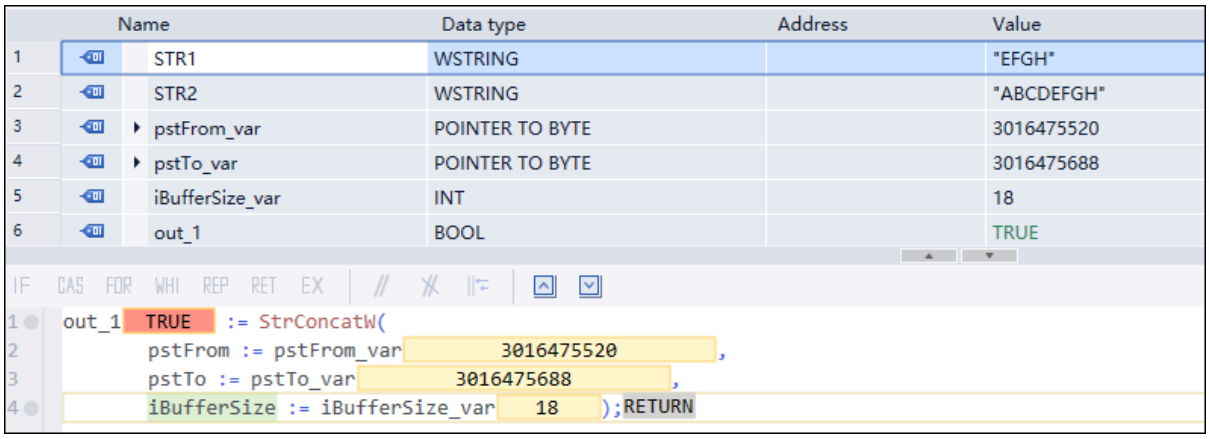
STR1 is the string pointed to by the pstFrom_var pointer, STR2 is the string pointed to by the pstTo_var pointer.

ST:

(1) Original state

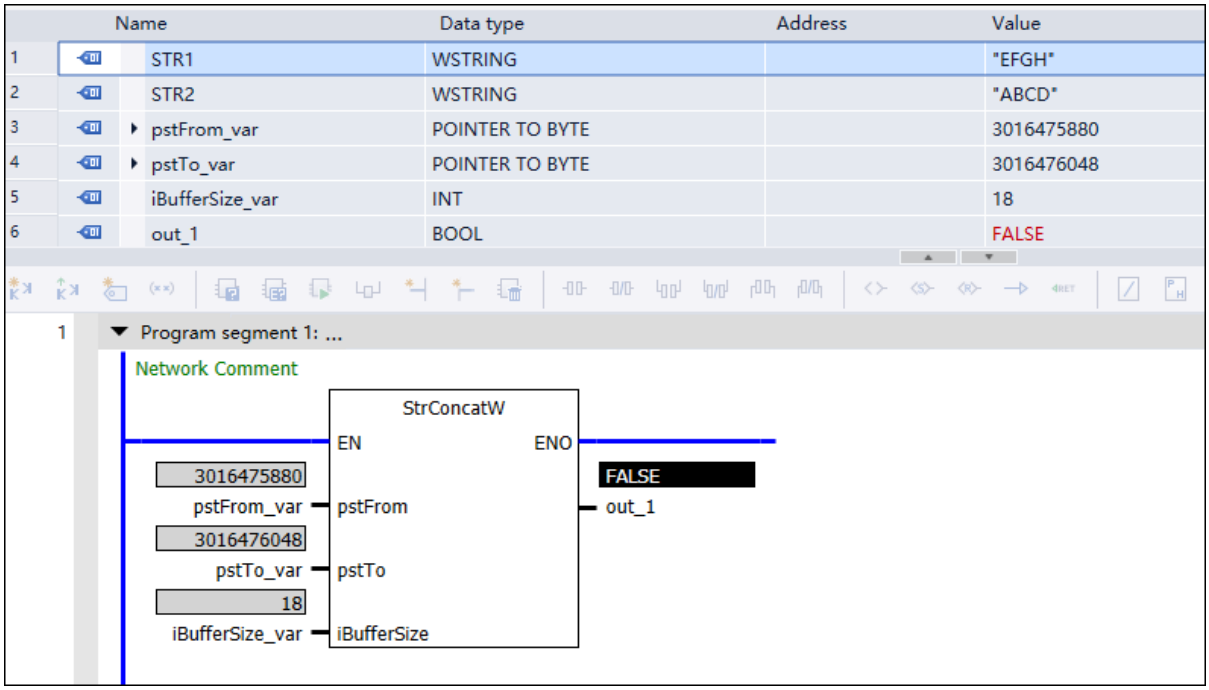


(2) Enable

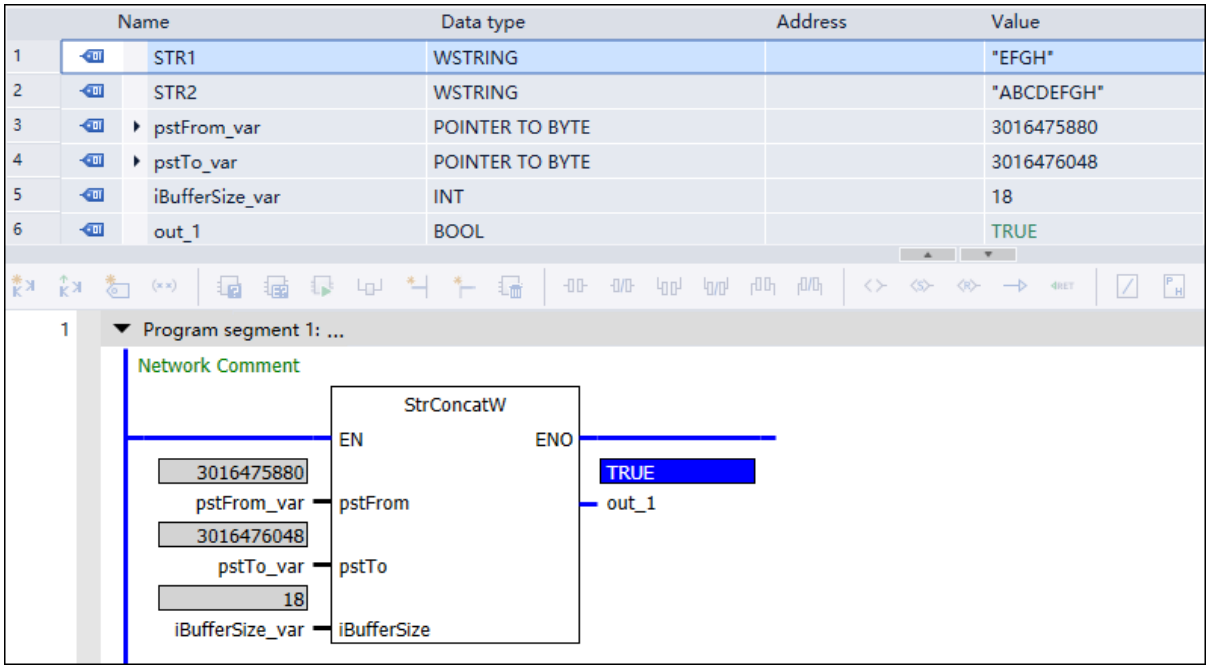


LD:

(1) Original state



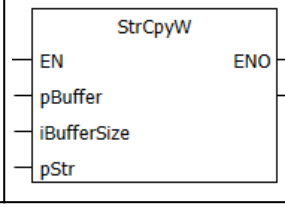
(2) Enable



18.5.3.2.5 StrCpyW

This instruction copies wide string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrCpyW	Wide string copy	FC		:= StrCpyW(pBuffer := , iBufferSize := , pStr :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pBuffer	Pointer to the buffer	Follows data type	-	Pointer to the buffer.
	iBufferSize	Buffer size	Follows data type	-	Buffer size.
	pStr	Pointer to the source string	Follows data type	-	Pointer to the source string.
Output	StrCpyW	Return value	Follows data type	-	Return value.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pBuffer	CharBufferPtr: POINTER TO BYTE																			
iBufferSize	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–
pStr	CharBufferPtr: POINTER TO BYTE																			
StrCpyW	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

- **Function**

  This instruction copies the string pointed to by pStr to the string pointed to by pBuffer. It should be noted that the number of characters copied is the smaller value between the number of characters in the source string being pointed to and the buffer size.

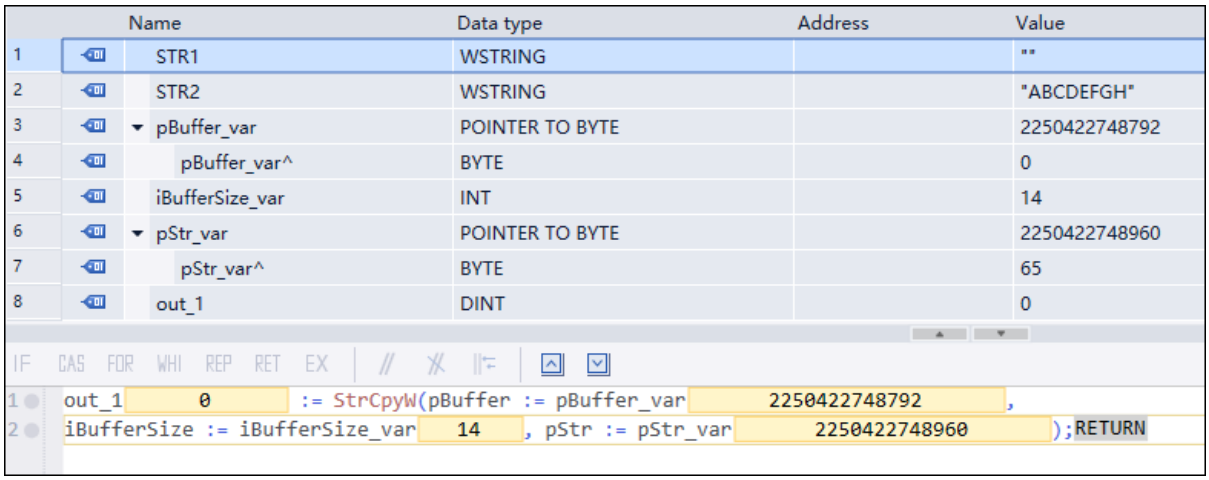
- **Precautions**
  - pBuffer must not be NULL.
  - Strings are typically null-terminated with "\0", which should be considered when calculating the length.
  - The return value of the function is always 0.

- **Program example**

  STR1 is the string pointed to by the pBuffer_var pointer, and STR2 is the string pointed to by the pStr_var pointer.

  ST:

  (1) Original state



  (2) Enable

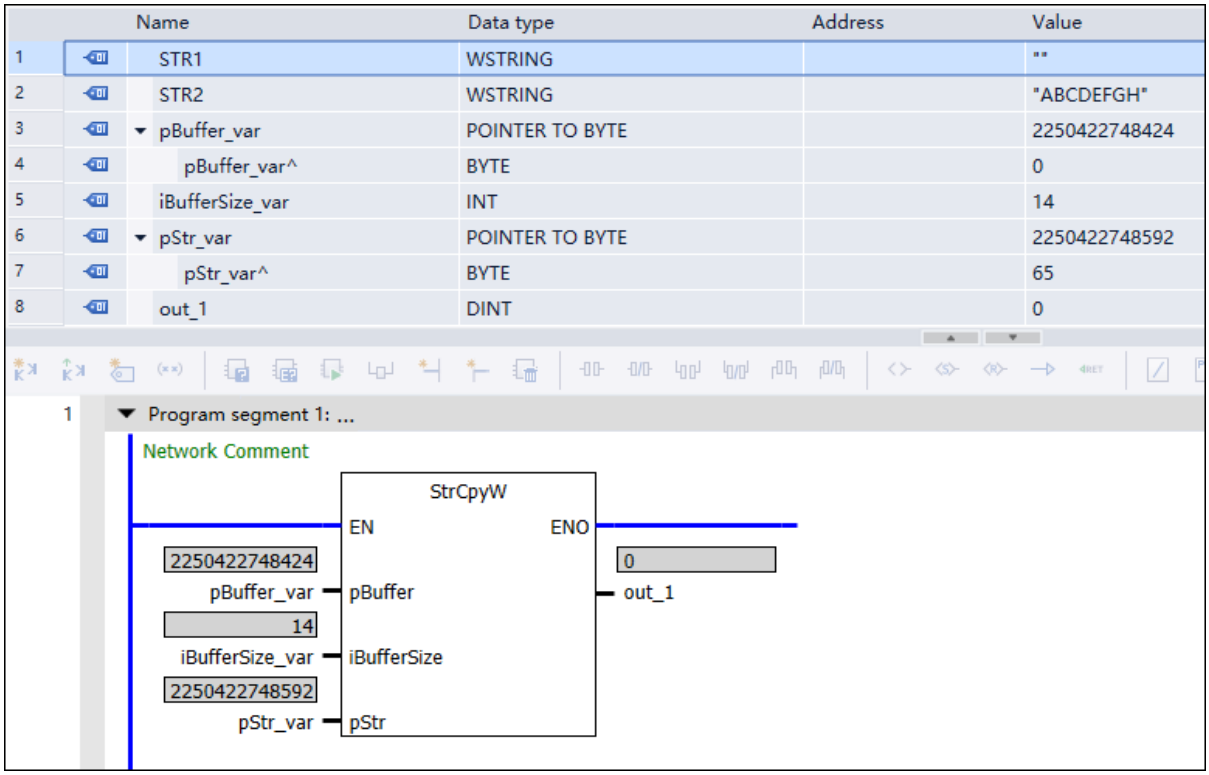
	Name	Data type	Address	Value
1	STR1	WSTRING		"ABCDEF"
2	STR2	WSTRING		"ABCDEFGH"
3	pBuffer_var	POINTER TO BYTE		2250422748792
4	pBuffer_var^	BYTE		65
5	iBufferSize_var	INT		14
6	pStr_var	POINTER TO BYTE		2250422748960
7	pStr_var^	BYTE		65
8	out_1	DINT		0

```
1  out_1 := StrCpyW(pBuffer := pBuffer_var,
2  iBufferSize := iBufferSize_var, pStr := pStr_var);RETURN
```

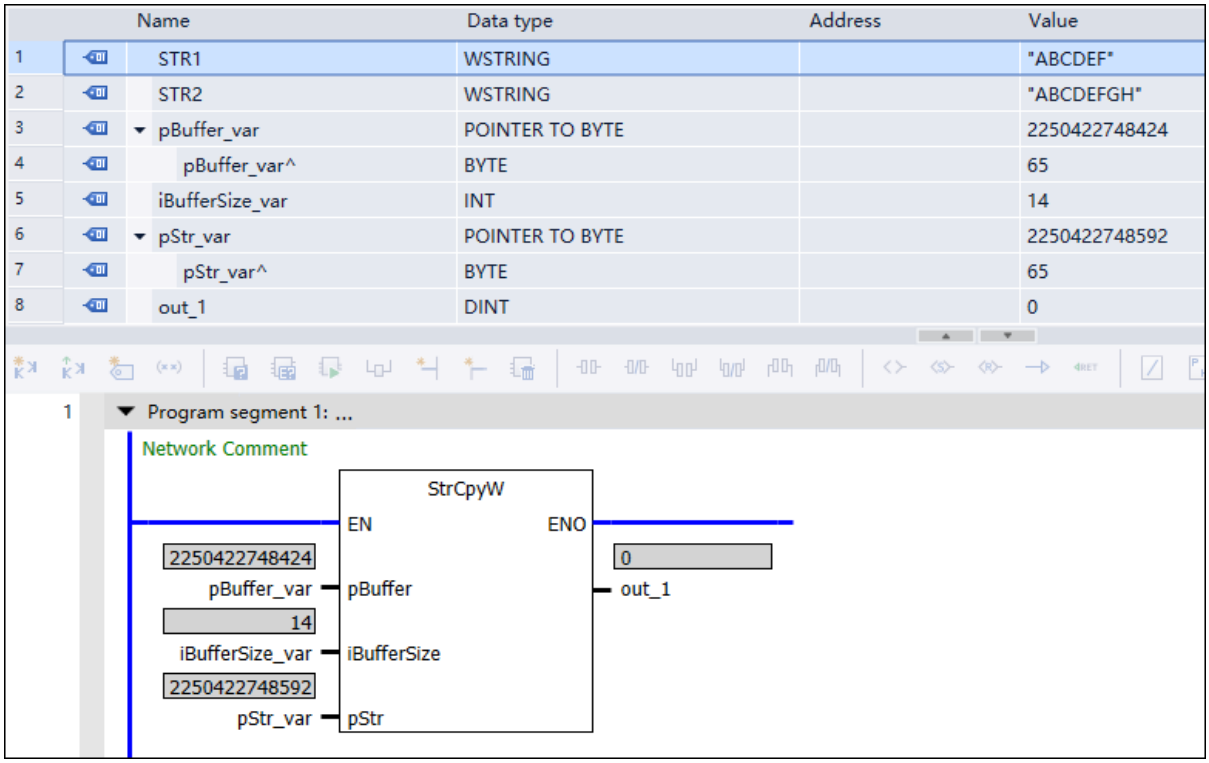
LD:

(1) Original state

	Name	Data type	Address	Value
1	STR1	WSTRING		""
2	STR2	WSTRING		"ABCDEFGH"
3	pBuffer_var	POINTER TO BYTE		2250422748424
4	pBuffer_var^	BYTE		0
5	iBufferSize_var	INT		14
6	pStr_var	POINTER TO BYTE		2250422748592
7	pStr_var^	BYTE		65
8	out_1	DINT		0



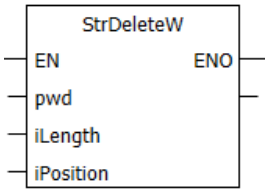
(2) Enable



18.5.3.2.6 StrDeleteW

This instruction deletes a specified-length string at a specified position from the input string, and output the processed string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrDeleteW	Wide string deletion	FC		:= StrDeleteW(pwd := , iLength := , iPosition :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pwd	Pointer to the string to be deleted	Follows data type	-	Pointer to the string to be checked.
	iLength	Deletion length	Follows data type	-	Length of the string to be deleted.
	iPosition	Deletion position	Follows data type	-	Position in the string pointed to by pwd pointer.

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	StrDeleteW	Return value	Follows data type	-	Return value.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pwd	POINTER TO WORD																			
iLength	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
iPosition	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
StrDeleteW	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction deletes a specified-length string at a specified position from the input string, and output the processed string.

- **Precautions**

  - The input string must not be NULL.
  - If the deletion length is greater than the remaining length of the string starting from the deletion position, the remaining length will be deleted.
  - The return value of the function is always 0.

- **Program example**

  STR1 is the string pointed to by the pwd1_var pointer.

  ST:

  (1) Original state

	Name	Data type	Address	Value
1	STR1	WSTRING		"ABCdeDEF"
2	pwd1_var	POINTER TO WORD		1380089987640
3	pwd1_var^	WORD		65
4	iLength_var	INT		2
5	iPosition_var	INT		4
6	out_1	BOOL		FALSE

```
1  out_1 FALSE := StrDeleteW(pwd := pwd1_var 1380089987640 ,
2  iLength := iLength_var 2 , iPosition := iPosition_var 4 );RETURN
```

(2) Enable

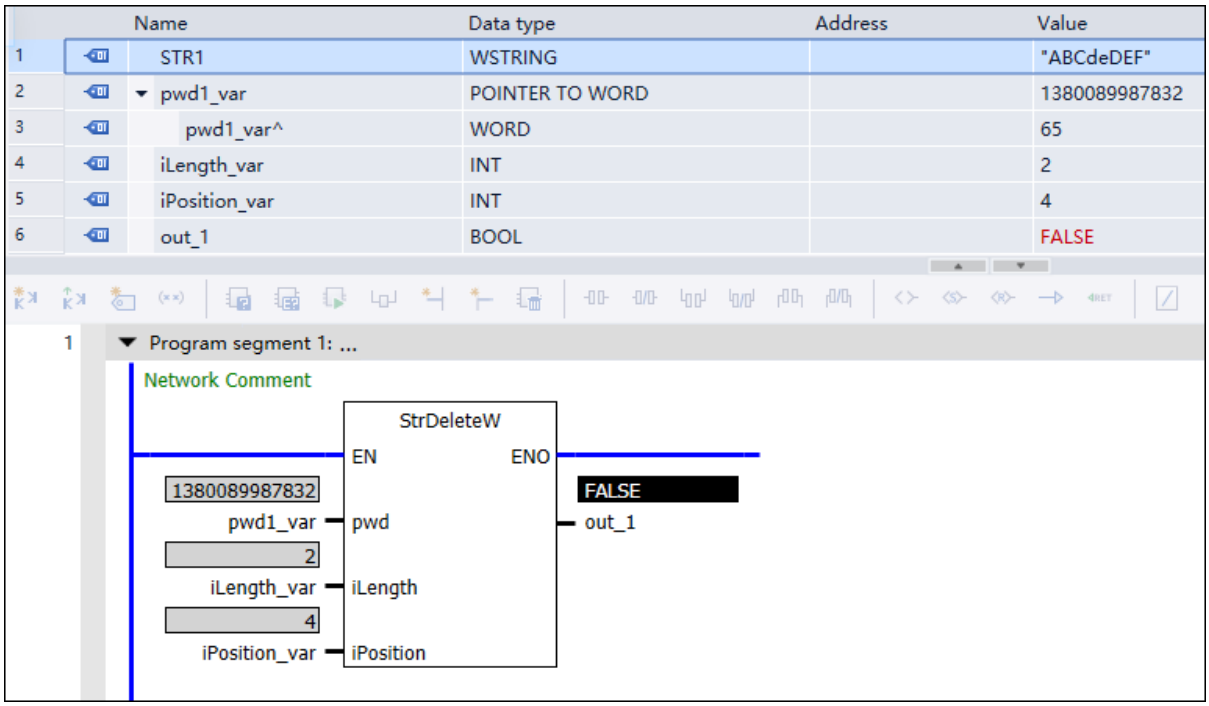
	Name	Data type	Address	Value
1	STR1	WSTRING		"ABCDEF"
2	pwd1_var	POINTER TO WORD		1380089987640
3	pwd1_var^	WORD		65
4	iLength_var	INT		2
5	iPosition_var	INT		4
6	out_1	BOOL		FALSE

```
1  out_1 FALSE := StrDeleteW(pwd := pwd1_var 1380089987640 ,
2  iLength := iLength_var 2 , iPosition := iPosition_var 4 );RETURN
```

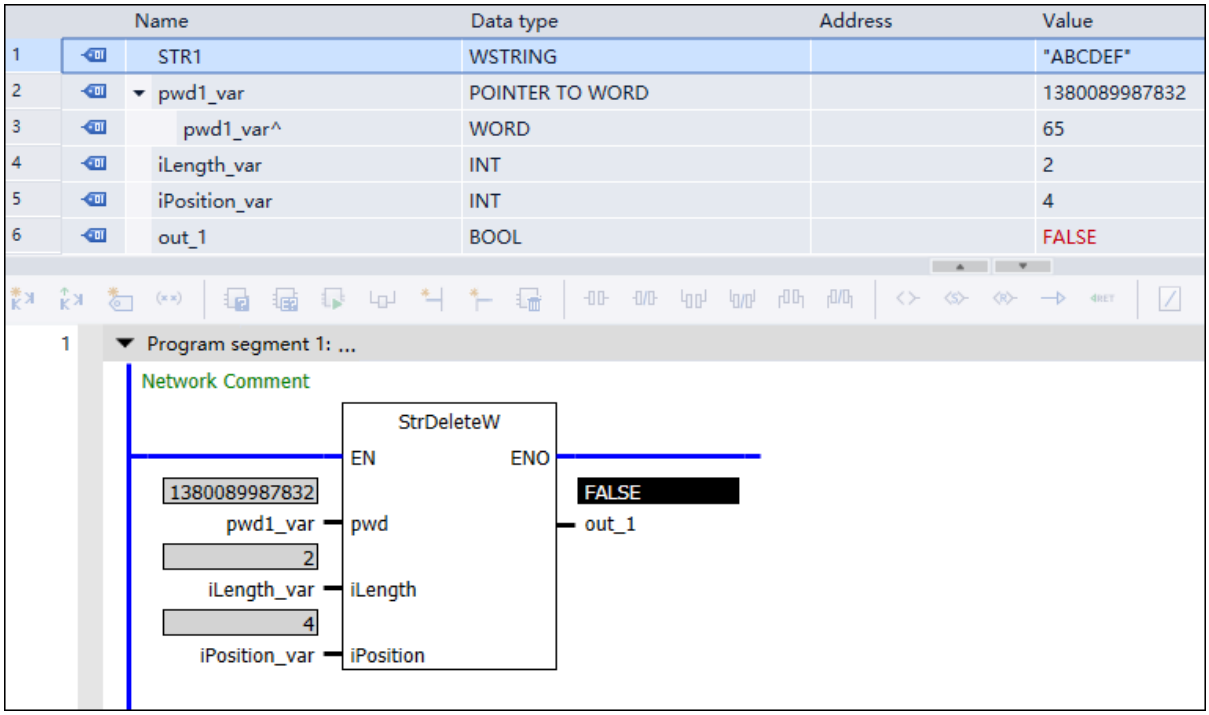
LD:

(1) Original state

	Name	Data type	Address	Value
1	STR1	WSTRING		"ABCdeDEF"
2	pwd1_var	POINTER TO WORD		1380089987832
3	pwd1_var^	WORD		65
4	iLength_var	INT		2
5	iPosition_var	INT		4
6	out_1	BOOL		FALSE



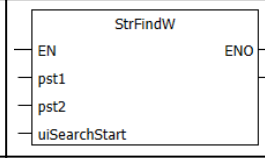
(2) Enable



18.5.3.2.7 StrFindW

This instruction performs a case-sensitive find operation on two wide strings.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrFindW	Wide string find (case-sensitive)	FC		:= StrFindW(pst1 := , pst2 := , uiSearchStart :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pst1	Pointer to the string to be checked	Follows data type	-	Pointer to the string to be checked.
	pst2	Pointer to the string to be found	Follows data type	-	Pointer to the string to be found.
	uiSearchStart	Search start position	Follows data type	-	Search start position.
Output	StrCaseFindW	Return value	Follows data type	-	Result.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pst1	POINTER TO WSTRING[255]																			
pst2	POINTER TO WSTRING[255]																			
uiSearchStart	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
StrCaseFindW	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction performs a case-sensitive find operation between two strings starting from the uiSearchStart-th character of the string pst1. If the string pst2 appears in the string pst1, return the position of the first character of string pst2 in string pst1. The results are as follows.

Return Value	Comparison Result
0	The string to be found does not exist.
-1	Errors arise from invalid strings, i.e., there is a null pointer.
>= 1	The position of the first character of STR 2 within STR 1, where the first character is counted as position 1.

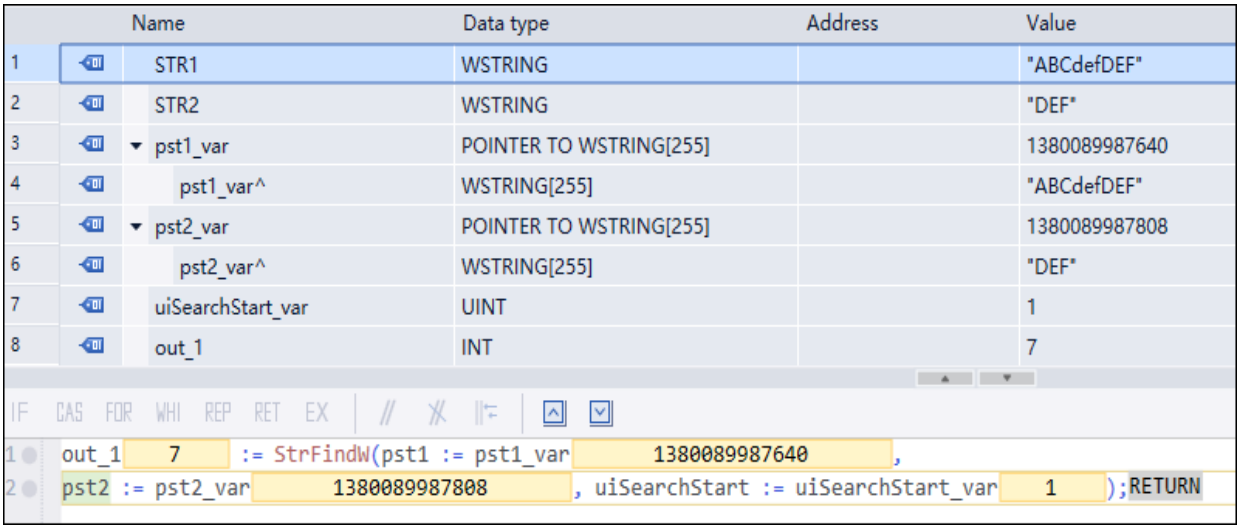
- **Precautions**

  When the search position uiSearchStart is 0 or 1, the search starts from the first character of STR1.

- **Program example**

  STR1 is the string pointed to by the pst1_var pointer, and STR2 is the string pointed to by the pst2_var pointer.

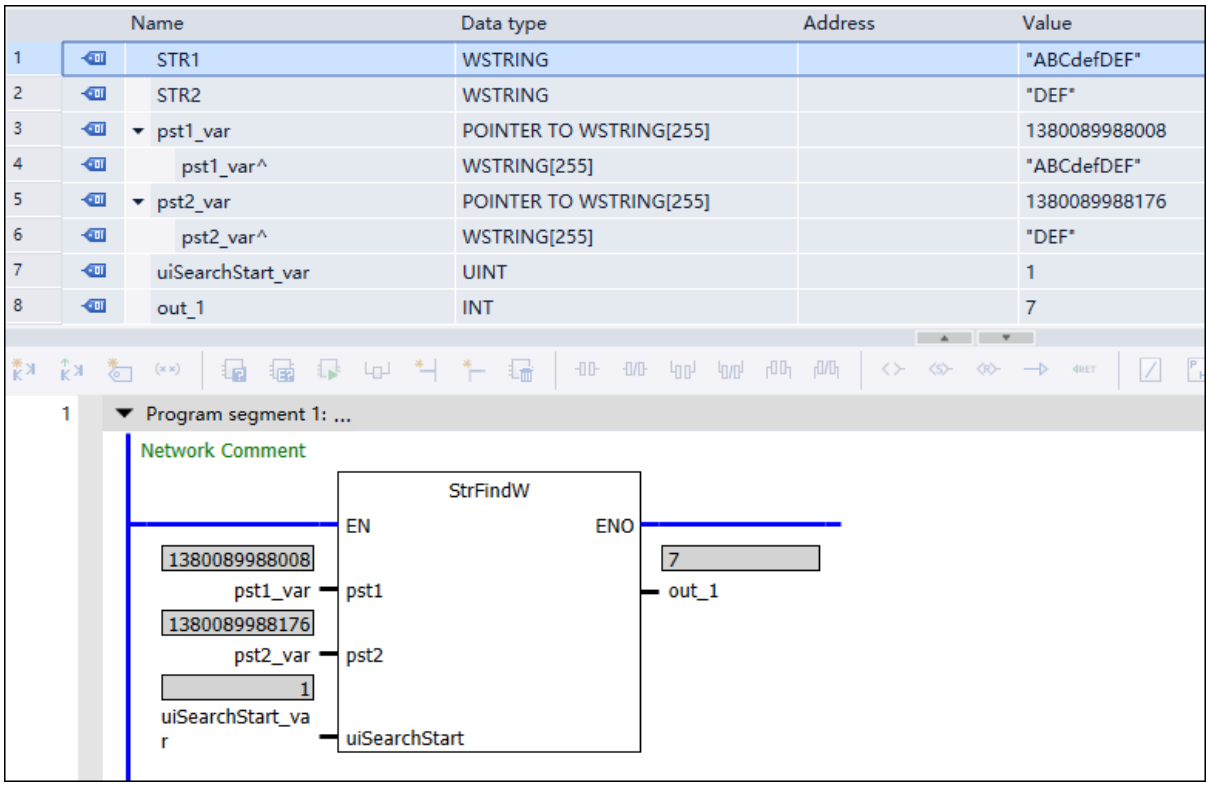
  ST:



	Name	Data type	Address	Value
1	STR1	WSTRING		"ABCdefDEF"
2	STR2	WSTRING		"DEF"
3	pst1_var	POINTER TO WSTRING[255]		1380089987640
4	pst1_var^	WSTRING[255]		"ABCdefDEF"
5	pst2_var	POINTER TO WSTRING[255]		1380089987808
6	pst2_var^	WSTRING[255]		"DEF"
7	uiSearchStart_var	UINT		1
8	out_1	INT		7

```
1  out_1 := StrFindW(pst1 := pst1_var , 1380089987640 ,
2  pst2 := pst2_var , 1380089987808 , uiSearchStart := uiSearchStart_var , 1 );RETURN
```

LD:



	Name	Data type	Address	Value
1	STR1	WSTRING		"ABCdefDEF"
2	STR2	WSTRING		"DEF"
3	pst1_var	POINTER TO WSTRING[255]		1380089988008
4	pst1_var^	WSTRING[255]		"ABCdefDEF"
5	pst2_var	POINTER TO WSTRING[255]		1380089988176
6	pst2_var^	WSTRING[255]		"DEF"
7	uiSearchStart_var	UINT		1
8	out_1	INT		7

18.5.3.2.8 StrIsNullOrEmptyW

This instruction judges if a wide string is empty.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrIsNullOrEmptyW	Empty string judgement	FC	StrIsNullOrEmptyW EN ENO pstData	:= StrIsNullOrEmptyW(pstData :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pstData	Pointer to byte	Follows data type	-	Pointer to byte.
Output	StrIsNullOrEmptyW	Return value	Follows data type	-	Return value.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pstData	CharBufferPtr: POINTER TO BYTE																			
StrIsNullOrEmptyW	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

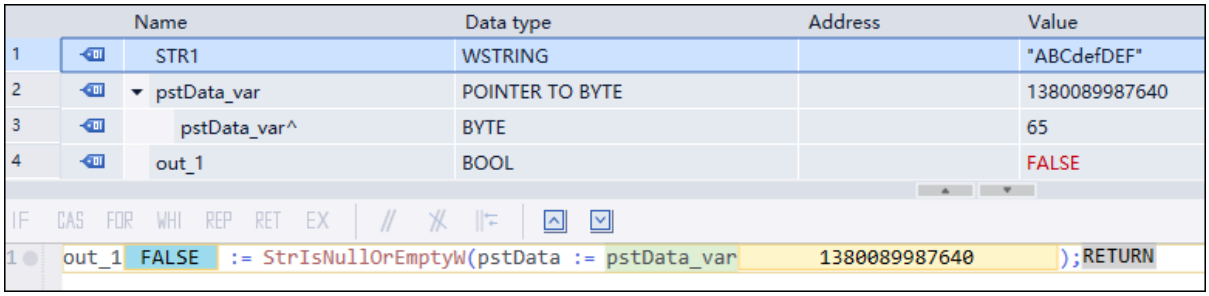
  This instruction judges if the string is empty. The result are as follows.

Return Value	Comparison Result
FALSE	The string is not empty.
TRUE	The string is empty.

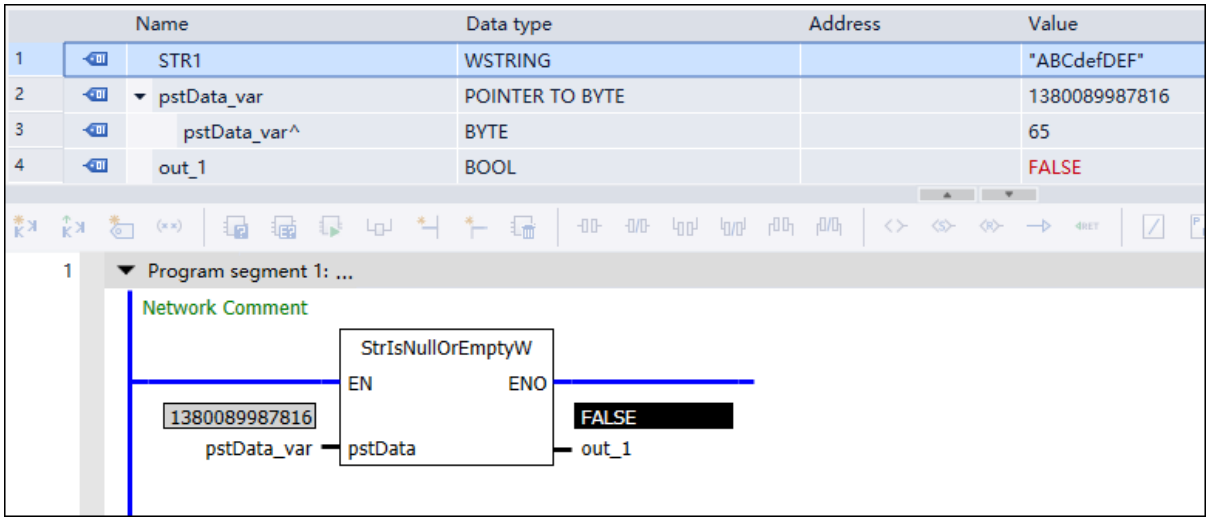
- **Program example**

  STR1 is the string pointed to by the pstData_var pointer.

  ST:



  LD:



18.5.3.2.9 StrLenW

This instruction calculates the length of a wide string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrLenW	Calculation of wide string length	FC	StrLenW EN ENO pstData	:= StrLenW(pstData :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pstData	Pointer to byte	Follows data type	-	Pointer to byte.
Output	StrLenW	Return value	Follows data type	-	Return value.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pstData	CharBufferPtr: POINTER TO BYTE																			

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
StrLenW	–	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–

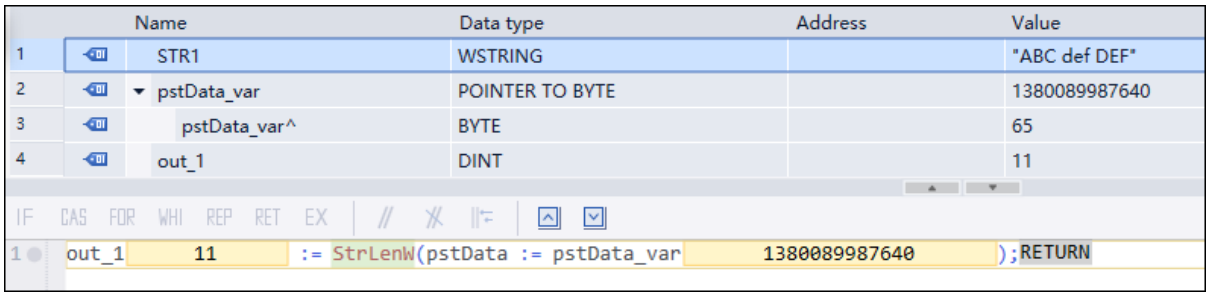
- **Function**

  This instruction calculates the length of the string pointed to by the pstData pointer with null characters excluded, and output the result as a return value.

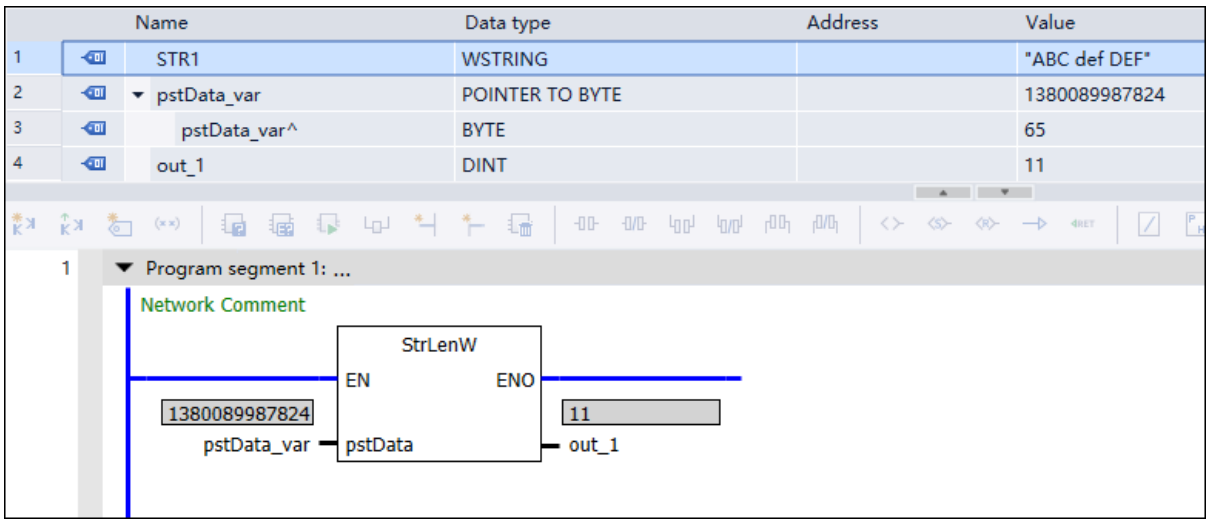
- **Program example**

  STR1 is the string pointed to by the pstData_var pointer.

  ST:



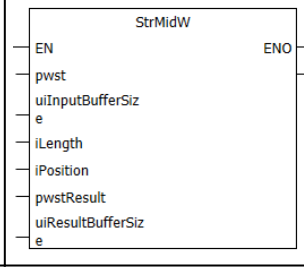
  LD:



18.5.3.2.10 StrMidW

This instruction extracts a wide string of specified length from a specified position in the source wide string, and return the string of the specified length.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrMidW	Substring acquisition	FC		:= StrMidW(pwst := , uiInputBufferSize := , iLength := , iPosition := , pwstResult := , uiResultBufferSize :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pwst	Pointer to the string to be extracted	Follows data type	-	Pointer to the string to be extracted.
	uiInputBufferSize	Buffer sizer pointed to by Pst	Follows data type	-	Buffer size pointed to by pwst.
	iLength	Caracter length	Follows data type	-	Character length.
	iPosition	Target position	Follows data type	-	Target position.
	pwstResult	Return result	Follows data type	-	Return result.
	uiResultBufferSize	Buffer size for the return result	Follows data type	-	Buffer size for the return result.
Output	StrMidW	Return value	Follows data type	-	Return value.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pwst	POINTER TO WORD																			

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
uiInputBufferSize	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
iLength	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
iPosition	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
pwstResult	POINTER TO WORD																			
uiResultBufferSize	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–
StrMidW	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction extracts a string of specified length from a specified position in the source string, and return the string of the specified length.

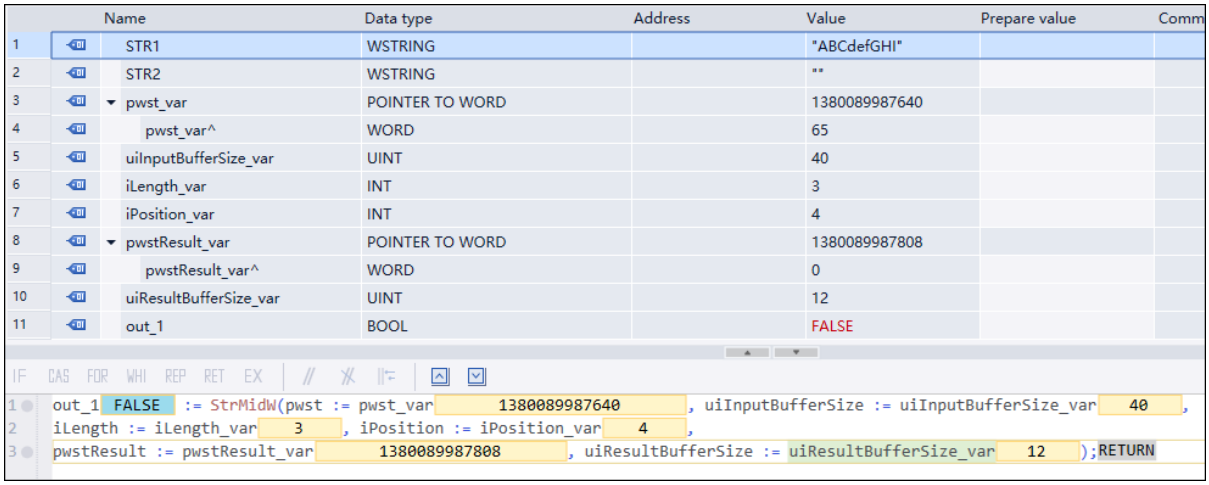
- **Precautions**
  - If iPosition is less than or equal to 0 or iLength is less than 0, the output will remain unchanged.
  - If the set position iPosition exceeds the length of the string, there will be no output.
  - If the substring of the set length iLength starting from the set position iPosition exceeds the length of the input string, then all characters from the source data starting at iPosition will be extracted.
  - If the length of the source string to be extracted exceeds the size of the set buffer uiResultBufferSize, only the data of the buffer size length will be truncated.
  - The return value of the function is always 0.

- **Program example**

  STR1 is the string pointed to by the pwst_var pointer, and STR2 is the string pointed to by the pwstResult_var pointer.

  ST:

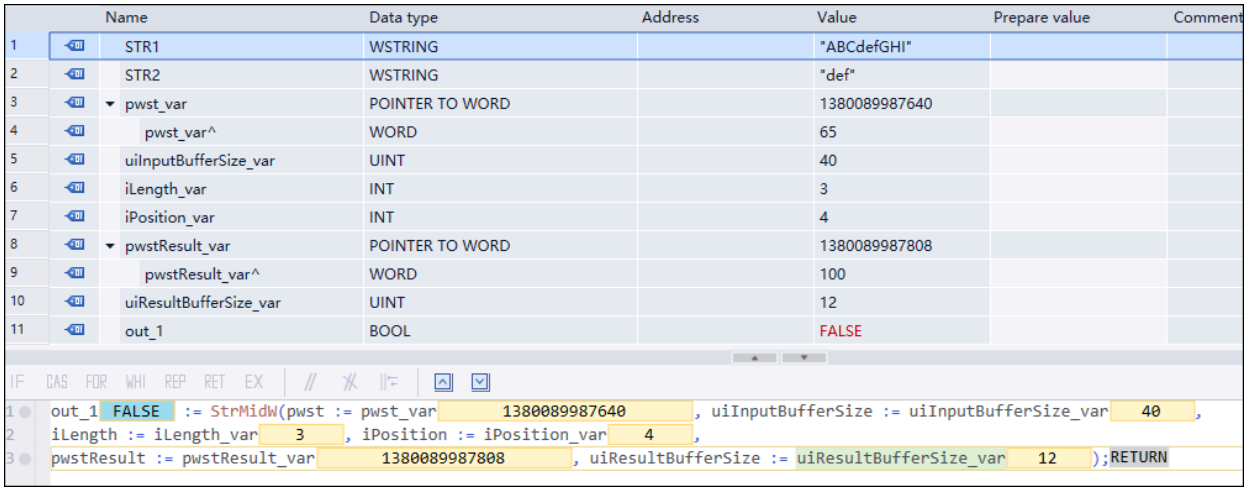
  (1) Original state



	Name	Data type	Address	Value	Prepare value	Comm
1	STR1	WSTRING		"ABCdefGHI"		
2	STR2	WSTRING		""		
3	pwst_var	POINTER TO WORD		1380089987640		
4	pwst_var^	WORD		65		
5	uiInputBufferSize_var	UINT		40		
6	iLength_var	INT		3		
7	iPosition_var	INT		4		
8	pwstResult_var	POINTER TO WORD		1380089987808		
9	pwstResult_var^	WORD		0		
10	uiResultBufferSize_var	UINT		12		
11	out_1	BOOL		FALSE		

```
1 out_1 FALSE := StrMidW(pwst := pwst_var 1380089987640 , uiInputBufferSize := uiInputBufferSize_var 40 ,
2 iLength := iLength_var 3 , iPosition := iPosition_var 4 ,
3 pwstResult := pwstResult_var 1380089987808 , uiResultBufferSize := uiResultBufferSize_var 12 );RETURN
```

(2) Enable

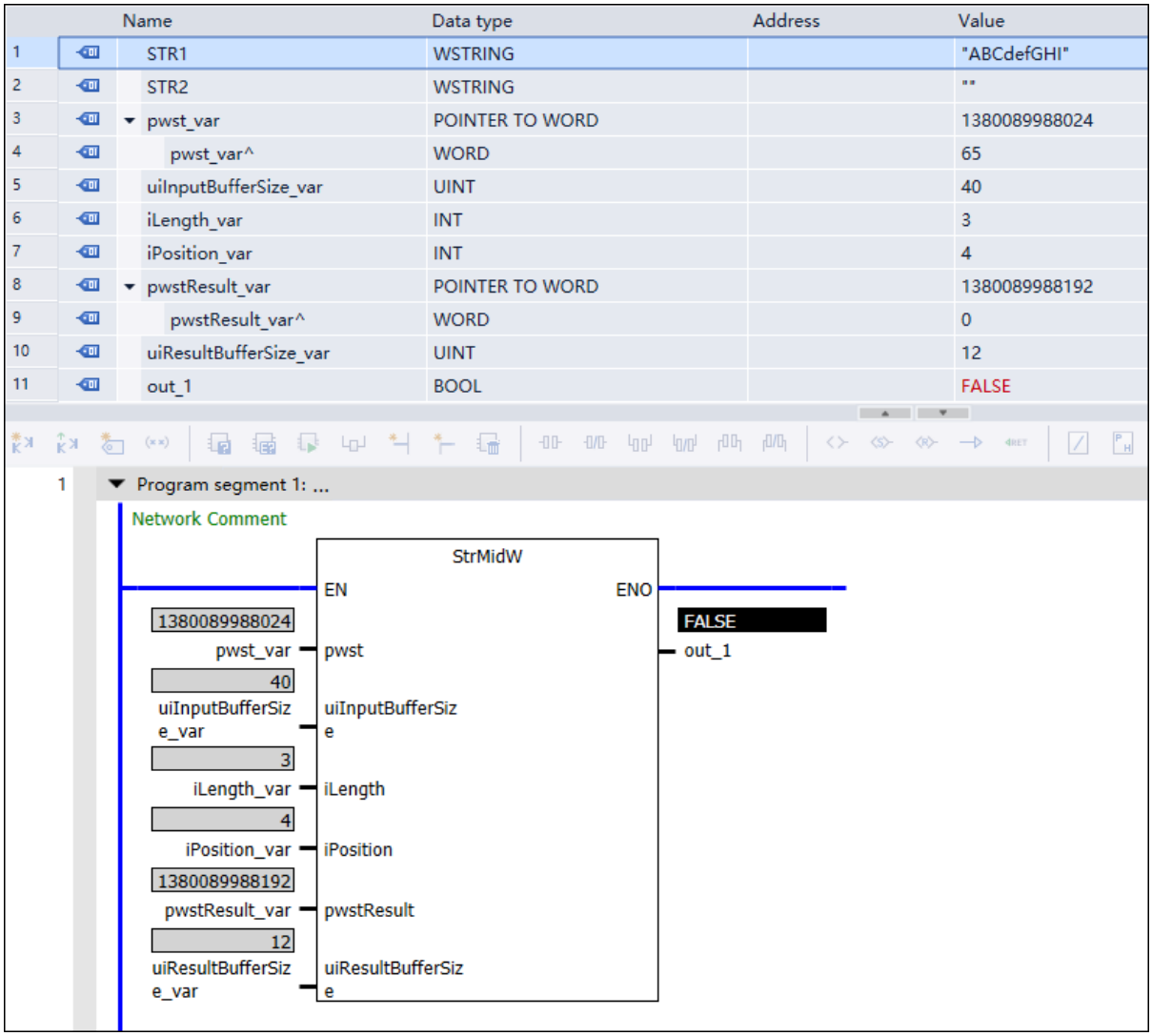


	Name	Data type	Address	Value	Prepare value	Comment
1	STR1	WSTRING		"ABCdefGHI"		
2	STR2	WSTRING		"def"		
3	pwst_var	POINTER TO WORD		1380089987640		
4	pwst_var^	WORD		65		
5	uiInputBufferSize_var	UINT		40		
6	iLength_var	INT		3		
7	iPosition_var	INT		4		
8	pwstResult_var	POINTER TO WORD		1380089987808		
9	pwstResult_var^	WORD		100		
10	uiResultBufferSize_var	UINT		12		
11	out_1	BOOL		FALSE		

```
1 out_1 FALSE := StrMidW(pwst := pwst_var 1380089987640 , uiInputBufferSize := uiInputBufferSize_var 40 ,
2 iLength := iLength_var 3 , iPosition := iPosition_var 4 ,
3 pwstResult := pwstResult_var 1380089987808 , uiResultBufferSize := uiResultBufferSize_var 12 );RETURN
```

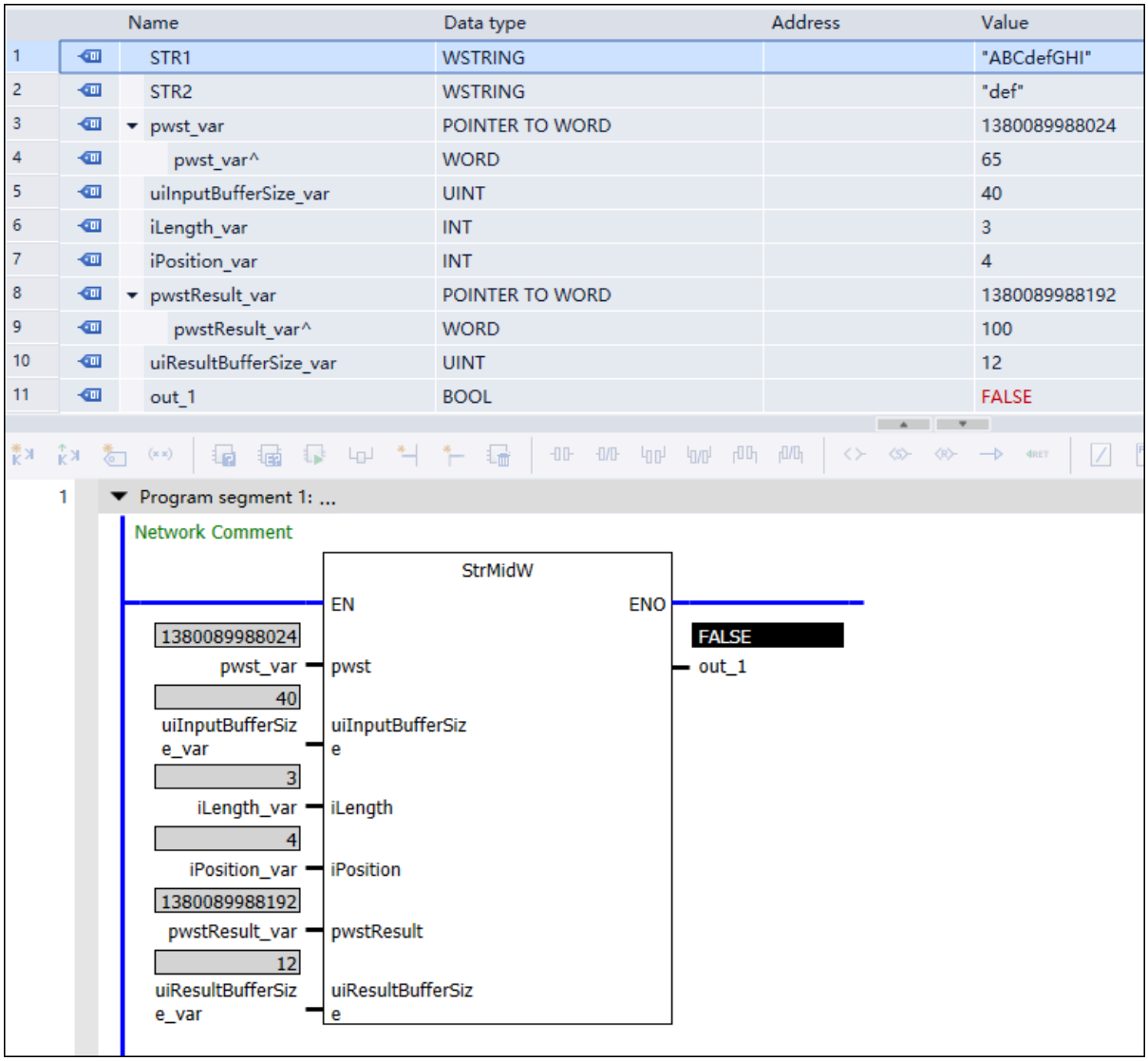
LD:

(1) Original state



	Name	Data type	Address	Value
1	STR1	WSTRING		"ABCdefGHI"
2	STR2	WSTRING		""
3	pwst_var	POINTER TO WORD		1380089988024
4	pwst_var^	WORD		65
5	uiInputBufferSize_var	UINT		40
6	iLength_var	INT		3
7	iPosition_var	INT		4
8	pwstResult_var	POINTER TO WORD		1380089988192
9	pwstResult_var^	WORD		0
10	uiResultBufferSize_var	UINT		12
11	out_1	BOOL		FALSE

(2) Enable



18.5.3.2.11 StrReplaceW

This instruction replaces a portion of the original string with the target string, and return the new string generated after the replacement as the return value.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
StrReplaceW	Wide string replacement	FC	StrReplaceW EN ENO pstInput uiInputBufferSize pstReplaceWith iLengthInput iLengthToReplace iLengthToReplaceWith iPosition	:= StrReplaceW(pstInput := , uiInputBufferSize := , pstReplaceWith := , iLengthInput := , iLengthToReplace := , iLengthToReplaceWith := , iPosition :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	pstInput	Pointer to the string to be replaced	Follows data type	-	Pointer to the string to be replaced.
	uiInputBufferSize	Buffer size pointed to by pstInput	Follows data type	-	Buffer size pointed to by pstInput.
	pstReplaceWith	Pointer to the replacement string	Follows data type	-	Pointer to the replacement string.
	iLengthInput	Length of input character	Follows data type	-	Character length
	iLengthToReplace	Target position	Follows data type	-	Target position.
	iLengthToReplaceWith	The length of the string that the pstReplaceWith pointer points to	Follows data type	-	The length of the string that the pstReplaceWith pointer points to.
	iPosition	Target position	Follows data type	-	Buffer size for the return result.
Output	StrReplaceW	Return value	Follows data type	-	Return value.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pstInput	POINTER TO WSTRING[255]																			
uiInputBufferSize	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
pstReplaceWith	POINTER TO WSTRING[255]																			
iLengthInput	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
iLengthToReplace	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
iLengthToReplaceWith	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
iPosition	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
StrReplaceW	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction replaces a part of the original string with the target string.

- **Precautions**
  - If iPosition is less than or equal to 0 or iLength is less than 0, the output will remain unchanged.
  - If the set position iPosition exceeds the length of the string, there will be no output.
  - If the substring selected from the set position iPosition with the set length iLengthInput exceeds the remaining length of the input string, the function is equivalent to extracting characters to the right of the specified position in the source string.
  - The return value of the function is always 0.

- **Program example**

  pstStr is the string pointed to by the pointer pstInput_var, and pstDst is the string pointed to by the pointer pstReplaceWith_var.

  ST:

  (1) Original state

	Name	Data type	Address	Value
1	pstStr	WSTRING		"ABCHIJK"
2	pstDst	WSTRING		"DEFG"
3	pstInput_var	POINTER TO WSTRING[255]		3016475128
4	pstInput_var^	WSTRING[255]		"ABCHIJK"
5	uiInputBufferSize_var	UINT		18
6	pstReplaceWith_var	POINTER TO WSTRING[255]		3016475296
7	pstReplaceWith_var^	WSTRING[255]		"DEFG"
8	iLengthInput_var	INT		18
9	iLengthToReplace_var	INT		4
10	iLengthToReplaceWith_var	INT		4
11	iPosition_var	INT		4
12	out_1	BOOL		FALSE

```
1  out_1 FALSE := StrReplaceW(
2          pstInput := pstInput_var 3016475128 ,
3          uiInputBufferSize := uiInputBufferSize_var 18 ,
4          pstReplaceWith := pstReplaceWith_var 3016475296 ,
5          iLengthInput := iLengthInput_var 18 ,
6          iLengthToReplace := iLengthToReplace_var 4 ,
7          iLengthToReplaceWith := iLengthToReplaceWith_var 4 ,
8          iPosition := iPosition_var 4 );RETURN
```

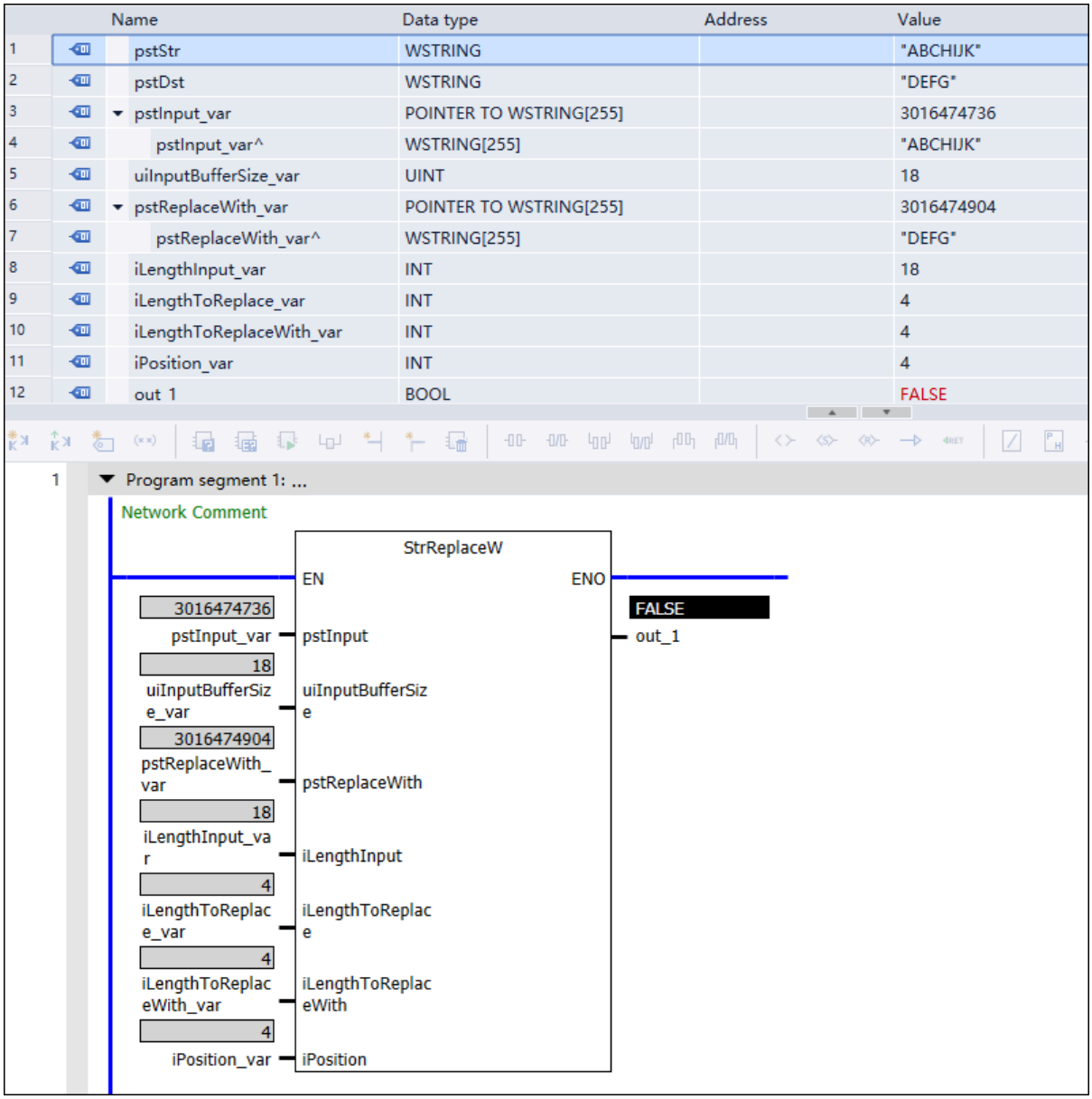
(2) Enable

	Name	Data type	Address	Value
1	pstStr	WSTRING		"ABCDEFG"
2	pstDst	WSTRING		"DEFG"
3	pstInput_var	POINTER TO WSTRING[255]		3016475128
4	pstInput_var^	WSTRING[255]		"ABCDEFG"
5	uiInputBufferSize_var	UINT		18
6	pstReplaceWith_var	POINTER TO WSTRING[255]		3016475296
7	pstReplaceWith_var^	WSTRING[255]		"DEFG"
8	iLengthInput_var	INT		18
9	iLengthToReplace_var	INT		4
10	iLengthToReplaceWith_var	INT		4
11	iPosition_var	INT		4
12	out_1	BOOL		FALSE

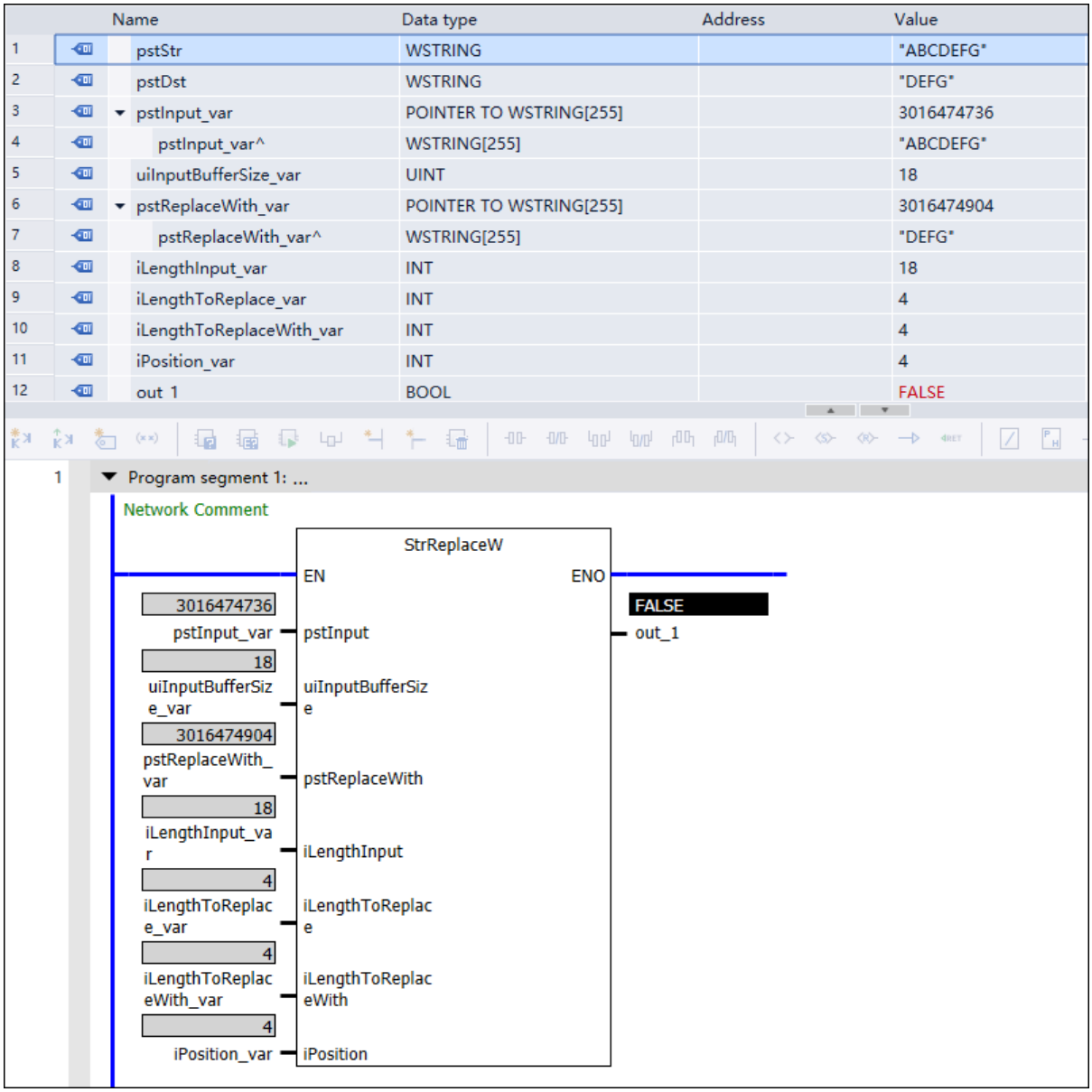
```
1  out_1 FALSE := StrReplaceW(
2          pstInput := pstInput_var 3016475128 ,
3          uiInputBufferSize := uiInputBufferSize_var 18 ,
4          pstReplaceWith := pstReplaceWith_var 3016475296 ,
5          iLengthInput := iLengthInput_var 18 ,
6          iLengthToReplace := iLengthToReplace_var 4 ,
7          iLengthToReplaceWith := iLengthToReplaceWith_var 4 ,
8          iPosition := iPosition_var 4 );RETURN
```

LD:

(1) Original state


Name
Data type
Address
Value
pstStr
WSTRING
"ABCHIJK"
pstDst
WSTRING
"DEFG"
pstInput_var
POINTER TO WSTRING[255]
3016474736
pstInput_var^
WSTRING[255]
"ABCHIJK"
uiInputBufferSize_var
UINT
18
pstReplaceWith_var
POINTER TO WSTRING[255]
3016474904
pstReplaceWith_var^
WSTRING[255]
"DEFG"
iLengthInput_var
INT
18
iLengthToReplace_var
INT
4
iLengthToReplaceWith_var
INT
4
iPosition_var
INT
4
out 1
BOOL
FALSE
Program segment 1: ...
Network Comment
StrReplaceW
EN
ENO
3016474736
pstInput_var
pstInput
FALSE
out_1
18
uiInputBufferSize_var
uiInputBufferSize
3016474904
pstReplaceWith_var
pstReplaceWith
18
iLengthInput_var
iLengthInput
4
iLengthToReplace_var
iLengthToReplace
4
iLengthToReplaceWith_var
iLengthToReplaceWith
4
iPosition_var
iPosition

(2) Enable



18.5.3.2.12 WCharToUpper

This instruction converts character to uppercase.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
WCharToUpper	Character uppercase conversion	FC	WCharToUpper EN ENO wdChar	:= WCharToUpper(wdChar :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	wdChar	Character to be converted	Follows data type	-	Character to be converted.

Input/Output	Variable	Name	Valid Range	Default Value	Description
Output	WCharToUpper	Return value	Follows data type	-	Converted character.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
wdChar	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
WCharToUpper	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

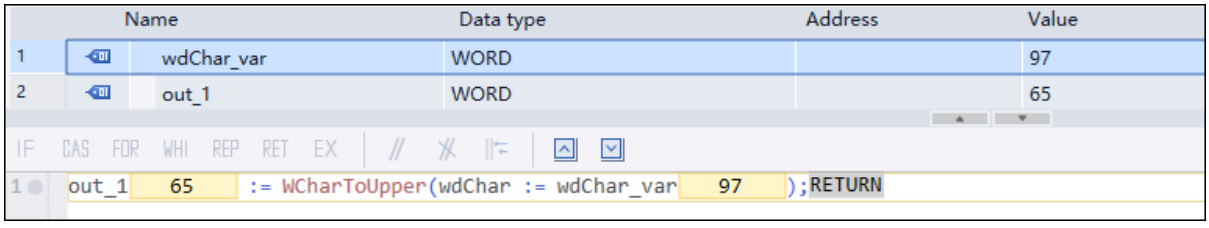
- **Function**

  This instruction converts the character wdChar to uppercase and output it as a WORD type.

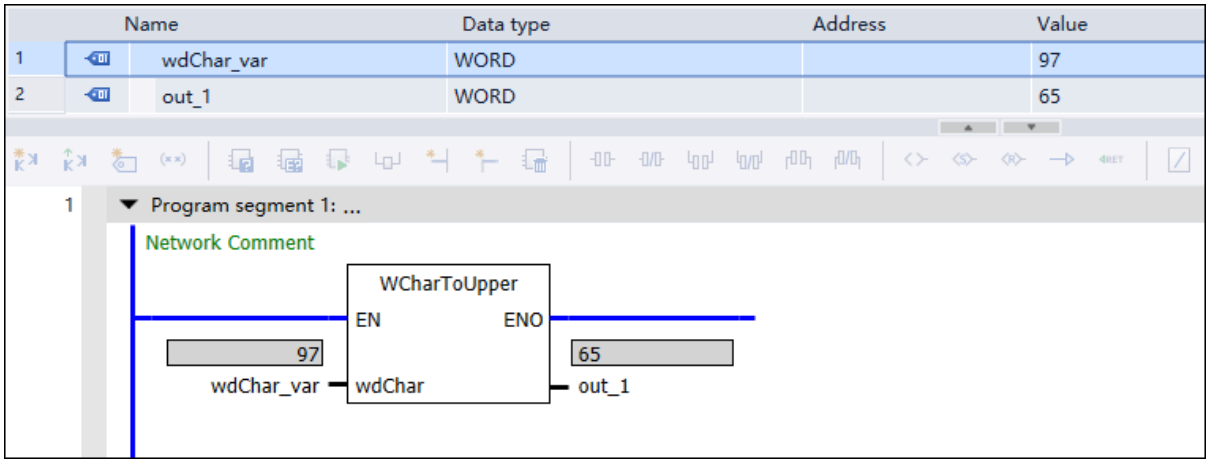
- **Precautions**
  - When the ASCII value of the character to be converted lies between 'a' and 'z', the algorithm performs the conversion; otherwise, the return value remains the ASCII code value of the original character.
  - The return value of the function is always 0.

- **Program example**

  ST:



  LD:



18.5.3.3 UTF8

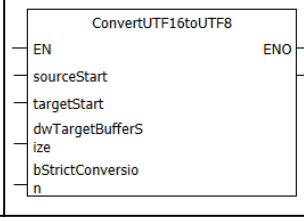
18.5.3.3.1 Instruction List

Instruction	FB/FC	Function	Library	Applicable Library Version
18.5.3.3.2 ConvertUTF16toUTF8	FC	String format conversion	StringUtils	Library version 1.0.0.0 and later
18.5.3.3.3 ConvertUTF8toUTF16	FC	String format conversion		Library version 1.0.0.0 and later
18.5.3.3.4 IsLegalUTF8	FC	UTF-8 format check		Library version 1.0.0.0 and later

18.5.3.3.2 ConvertUTF16toUTF8

This instruction converts a UTF-16 string to a UTF-8 string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ConvertUTF16toUTF8	String format conversion	FC		:= ConvertUTF16toUTF8(sourceStart := , targetStart := , dwTargetBufferSize := , bStrictConversion :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	sourceStart	Pointer to the UTF-16 string to be converted	Follows data type	-	Pointer to the UTF-16 string to be converted.
	targetStart	Pointer to the UTF-8 string to be converted to	Follows data type	-	Pointer to the UTF-8 string to be converted to.
	dwTargetBufferSize	Buffer size	Follows data type	-	Buffer size.
	bStrictConversion	Conversion restriction	Follows data type	-	Conversion restriction.
Output	ConvertUTF16toUTF8	Return value	Follows data type	-	Return value.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
sourceStart	POINTER TO WORD																			
targetStart	POINTER TO BYTE																			
dwTargetBufferSize	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bStrictConversion	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ConvertUTF16toUTF8	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction converts a UTF-16 string to a UTF-8 string.

  The bstrictConversion parameter controls conversion strictness, ensuring proper handling of invalid or non-convertible characters during UTF-16-to-UTF-8 conversion
  and preventing data corruption or erroneous output.

- **Program example**

  STR1 is the UTF-16 string pointed to by the sourceStart_var pointer, and STR2 is the UTF-8 string pointed to by the targetStart_var pointer. Convert the UTF-16 input variable of WORD type to a UTF-8 string.

  Before conversion: the UTF-16 string STR1 := 16#6C49 (UTF16 represents character "汉")

  After conversion: the UTF-8 string is [16#E6, 16#B1, 16#89] (UTF8 represents character "汉")

  ST:

  LD:

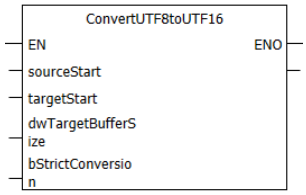
- **Precautions**

  - If the data buffer size stored in "dwTargetBufferSize" is incorrect, the function return value is 16#40A1.
  - If the input is non-convertible data, the function return value is 16#40A2.

18.5.3.3.3 ConvertUTF8toUTF16

This instruction converts a UTF-8 string to a UTF-16 string.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
ConvertUTF8toUTF16	String format conversion	FC		:= ConvertUTF8toUTF16(sourceStart := , targetStart := , dwTargetBufferSize := , bStrictConversion :=);

- **Related variables**

Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	sourceStart	Pointer to the UTF-8 string to be converted	Follows data type	-	Pointer to the UTF-8 string to be converted.
	targetStart	Pointer to the UTF-16 string to be converted to	Follows data type	-	Pointer to the UTF-16 type string to be converted to.
	dwTargetBufferSize	Buffer size	Follows data type	-	Buffer size.
	bStrictConversion	Conversion restriction	Follows data type	-	Conversion restriction.
Output	ConvertUTF8toUTF16	Return value	Follows data type	-	Return value.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
sourceStart	POINTER TO BYTE																			
targetStart	POINTER TO WORD																			
dwTargetBufferSize	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
bStrictConversion	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
ConvertUTF8toUTF16	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–	–	–	–

- **Function**

  This instruction converts a UTF-8 string to a UTF-16 string.

  The bstrictConversion parameter controls conversion strictness, ensuring proper handling of invalid or non-convertible characters during UTF-8-to-UTF-16 conversion
  and preventing data corruption or erroneous output.

- **Program example**

  STR1 is the UTF-8 string pointed to by the sourceStart_var pointer, and STR2 is the UTF-16 string pointed to by the targetStart_var pointer. Convert the BYTE-type UTF-8 input variable to a UTF-16 string.

  Before conversion: The UTF-8 string STR[0] := 16#E6, STR[1] := 16#B1, STR[2] := 16#89 (UTF8 represents character "汉")

  After conversion: The UTF-16 string is 16#6C49 (UTF16 represents character "汉")

  ST:

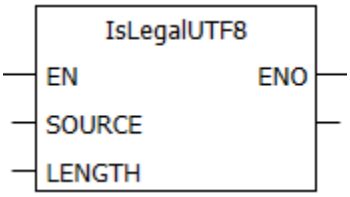
  LD:

- **Precautions**
  - If the data buffer size stored in "dwTargetBufferSize" is incorrect, the function return value is 16#40A1.
  - If the input is non-convertible data, the function return value is 16#40A2.

18.5.3.3.4 IsLegalUTF8

This instruction checks if the given string contains valid UTF-8 characters.

- **Instruction format**

Instruction	Name	FB/FC	LD Representation	ST Representation
IsLegalUTF8	UTF-8 format check	FC		:= IsLegalUTF8(SOURCE := , LENGTH :=);

- **Related variables**

  Variable

Input/Output	Variable	Name	Valid Range	Default Value	Description
Input	SOURCE	Pointer to the source string	Follows data type	-	Pointer to the source string.
	LENGTH	String length	Follows data type	-	String length
Output	IsLegalUTF8	Return value	Follows data type	-	Return value.

Variable types

	Boolean	Bit String				Integer								Real Number		Time/Duration/ Date/String				
	BOOL	BYTE	WORD	DWORD	LWORD	USINT	UINT	UDINT	ULINT	SINT	INT	DINT	LINT	REAL	LREAL	TIME	DATE	TIME_OF_DAY	DATE_AND_TIME	STRING
source Start	POINTER TO BYTE																			
LENGTH	–	–	–	–	–	–	–	–	–	–	√	–	–	–	–	–	–	–	–	–
IsLegal UTF8	√	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

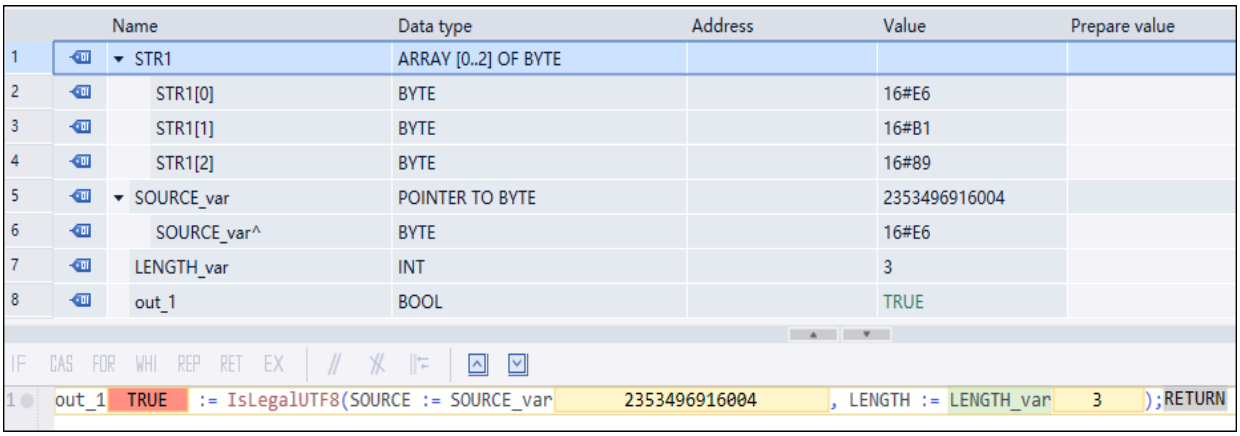
- **Function**

  This instruction checks if the given string consists of valid UTF-8 characters; return TRUE if it does, otherwise return FALSE.

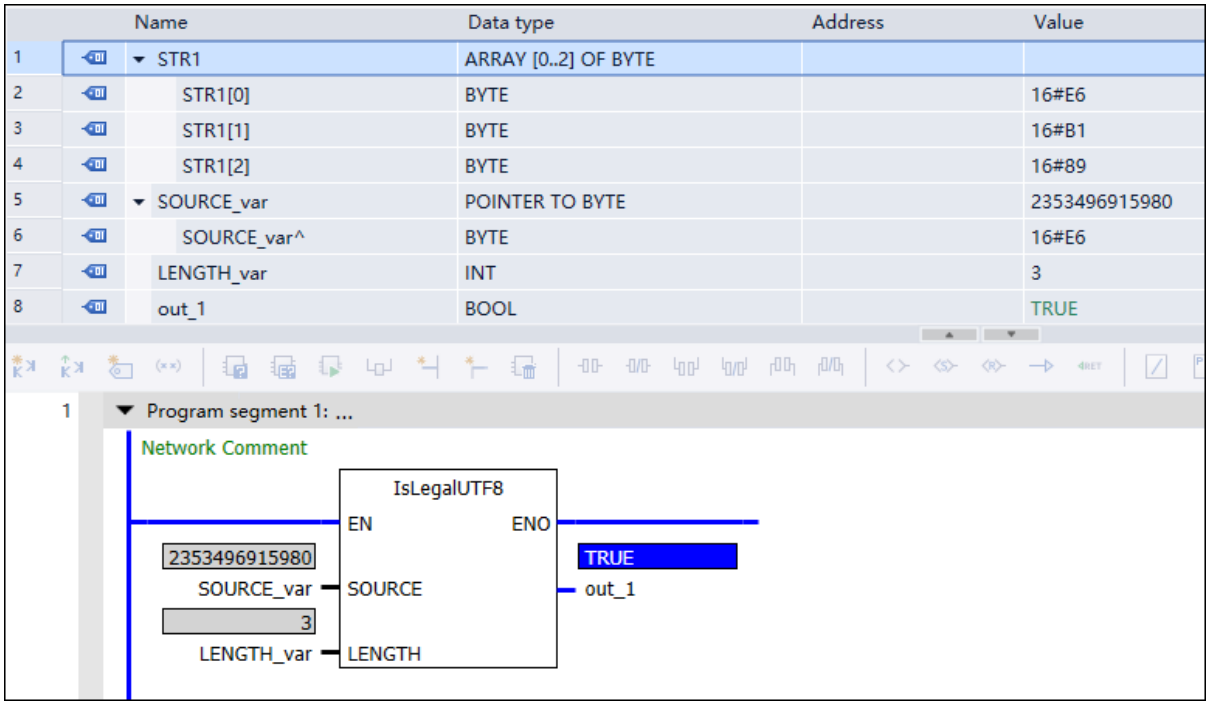
- **Program example**

  STR1 is the string pointed to by the sourceStart_var pointer.

  ST:



LD:


Name
Data type
Address
Value
1 STR1 ARRAY [0..2] OF BYTE
2 STR1[0] BYTE 16#E6
3 STR1[1] BYTE 16#B1
4 STR1[2] BYTE 16#89
5 SOURCE_var POINTER TO BYTE 2353496915980
6 SOURCE_var^ BYTE 16#E6
7 LENGTH_var INT 3
8 out_1 BOOL TRUE
1 Program segment 1: ...
Network Comment
IsLegalUTF8
EN
ENO
2353496915980
SOURCE_var
SOURCE
TRUE
out_1
3
LENGTH_var
LENGTH

19 HMI System Functions

19.1 Variables

- This function is used for getting a variable's class object by its name.
  - **Function format**
    smartTags('Tag_1')
  - **Related parameters**

Parameter	Description	Value/Interface
Variable name	Configured as a String variable, such as "Tag_1"	The default value is null.
Return value	Class variable, used to describe a class of variables in monitoring	getName()----Get the variable name valueOf()----Get the variable value

  - **Examples**

Application Type	Example

Get the variable value	function Script_1() {
	//The value of INT variable Tag_1 is 100
	//smartTags('Tag_1') -- object of variable Tag_1
	let a = getValue(smartTags('Tag_1')); //After the getValue is executed, a = 100
	
	let b = smartTags('Tag_1').valueOf(); // After the valueOf is executed, b = 100
	}

Set the variable value	function Script_1() {
	//The default value of INT variable Tag_1 is 0
	//smartTags('Tag_1') -- object of variable Tag_1
	setValue(smartTags('Tag_1'), 100); //After the setValue is executed, the value of variable Tag_1 is 100
	}

Use the variable object as input parameter	function Script_1() {
	//Set the screensaver time based on the value of INT variable Tag_1
	//smartTags('Tag_1') -- object of variable Tag_1
	setScreenSaverTime(smartTags('Tag_1'))
	
	}

Application Type	Example

Use the variable object as output parameter	function Script_1(){
	//Export the specified data records to the specified path, smartTags('Tag_1') -- object of variable Tag_1
	exportTagRecord("data record_1", 0, "/logRecord", smartTags('Tag_1')) //After the exportTagRecord is executed, the result is written to variable Tag_1
	}

Use the variable object as input and output parameters	function Script_1(){
	//smartTags('Tag_1') -- The default value of INT variable Tag_1 is 10
	//During the execution of increaseValue, the value of variable Tag_1 is obtained and then 1 is added and assigned back to the variable Tag_1
	increaseValue(smartTags('Tag_1'), 1, true); //After the increaseValue is executed, the value of variable Tag_1 is 11
	}

19.2 Bit Operations

Name	Function	Example

Invert a bit	invertBit	function Script_2(){ //The default value of BOOL variable Tag_1 is true
		invertBit(smartTags('Tag_1')); //After the invertBit is executed, the default value of Tag_1 is false
		}

Invert a bit in the given tag	invertBitInTag	function Script_2() { //The default value of INT variable Tag_1 is 0b0000000001. Invert the second bit
		invertBitInTag(smartTags('Tag_1'),2,false); //After the invertBitInTag is executed, the default value of Tag_1 is 000000000101
		}

Set a bit	setBit	function Script_2() { //The default value of BOOL variable Tag_1 is false
		setBit(smartTags('Tag_1')); //After the setBit is executed, the value of Tag_1 is true
		}

Set a bit in the given tag	setBitInTag	function Script_2(){ //The default value of INT variable Tag_1 is 0b0000000001. Set the second bit to 1
		resetBitInTag(smartTags('Tag_1'),2); //After the resetBitInTag is executed, the value of Tag_1 is 0000000000101
		}

Name	Function	Example

Reset a bit	resetBit	function Script_2(){ //The default value of BOOL variable Tag_1 is true
		resetBit(smartTags('Tag_1')); //After the resetBit is executed, the default value of Tag_1 is false
		}

Reset a bit in the given tag	resetBitInTag	function Script_2(){ //The default value of INT variable Tag_1 is 0b0000000101. Reset the second bit to 0
		resetBitInTag(smartTags('Tag_1'),2); //After the resetBitInTag is executed, the default value of Tag_1 is 0b00000000001
		}

19.3 Calculation

19.3.1 Increasing the Variable Value

This function is used for increasing a value to the variable.

- **Function format**

  increaseValue(/*variable(InOut)*/, /*variable/constant(In)*/, /*reset*/, /*whether to record the operation*/)

- **Related parameters**

Parameter	Description	Value
Variable	Integers and decimals are supported.	Configured as an integer or a decimal. The default value is null.
Value	Constants and variables are supported.	Configured as a constant or a variable. The default value is 1.
Reset	Reset the value.	Options: Yes/No. The default option is No. When Yes is selected, the variable value is reset to the lower limit when reaching the upper limit.
Whether to record the operation	Whether to record the operation event when this function is executed. The records can be found in the 13.3.5.4.8 Operation Logs view.	The default option is No.

- **Examples**

  Use in scripts:

```
function Script_1() {
//The default value of INT variable Tag_1 is 10, and the value range is [0,100]
//smartTags('Tag_1') -- object of variable Tag_1; true means reset after exceeding the limit
increaseValue(smartTags('Tag_1'), 1, true);
//After the increaseValue is executed, the value of variable Tag_1 is 11

//The default value of INT variable Tag_1 is 100, and the value range is [0,100]
```

```
//smartTags('Tag_1') -- object of variable Tag_1; true means reset after exceeding the limit
increaseValue(smartTags('Tag_1'), 1, true);
//After the increaseValue is executed, the value of variable Tag_1 is 0. The variable value increases by 1 each cycle until it exceeds the upper limit, at which point it is reset to the lower limit 0.
}
```

19.3.2 Decreasing the Variable Value

This function is used for decreasing a value of the variable.

- **Function format**

  ```
  decreaseValue(/*variable(InOut)*/, /*variable/constant(In)*/, /*reset*/, /*whether to record the operation*/)
  ```

- **Related parameters**

Parameter	Description	Value
Variable	Integers and decimals are supported.	Configured as an integer or a decimal. The default value is null.
Value	Constants and variables are supported.	Configured as a constant or a variable. The default value is constant 1.
Reset	Reset the value.	Options: Yes/No. The default option is No. When Yes is selected, the variable value is reset to the upper limit when reaching the lower limit.
Whether to record the operation	Whether to record the operation event when this function is executed. The records can be found in the 13.3.5.4.8 Operation Logs view.	The default option is No.

- **Examples**

  Use in scripts:

```
function Script_1() {

//The default value of INT variable Tag_1 is 10, and the value range is [0,100]
decreaseValue(smartTags(('Tag_1'),1,true); //smartTags(*Tag_1') - object of variable Tag_1, true means reset after exceeding the limit
//after the decreaseValue is executed, the value of variable Tag_1 is 9

//The default value of INT variable Tag_1 is 0, and the value range is [0,100]
decreaseValue(smartTags('Tag_1'),1,true); //smartTags('Tag_1)--object of variable Tag_1 , true means reset after exceeding the limit
//after the decreaseValue is executed, the value of variable Tag_1 is 100. The variable value decreases by 1 on each call, until it is below the lower limit, at which point it is reset to the upper limit 100.

}
```

19.3.3 Setting the Variable Value

This function is used for setting a value for the variable.

It is applicable to set values for variables of integer, word, decimal, boolean, and text types.

- **Function format**

  setValue(/*variable(Inout)*/, /*variable/constant(In)*/, /*whether to record the operation*/)

- **Related parameters**

Parameter	Description	Value
Variable	In-out	The default value is null.
Variable/ Constant	Constants and variables are supported.	The default value is constant 0.
Whether to record the operation	Whether to record the operation event when this function is executed. The records can be found in the 13.3.5.4.8 Operation Logs view.	The default option is No.

- **Examples**

  Use in scripts:

```
//Example 1. Set a fixed constant value of 100 for variable Tag1.
setValue(smartTags('Tag1'), 100);

//Example 2. Set a dynamically calculated value for variable Tag1.
//This value is obtained by adding 100 to the current value of Tag2.
setValue(smartTags('Tag1'), getValue(smartTags('Tag2')) + 100);
```

19.3.4 Getting the Variable Value

This function allows users to read the values of configured variables in the monitoring system and performs proper logical processing.

1. This function can be called in user-defined scripts.
2. This function cannot be called through configuration in variable events and screen events.

- **Function format**

  let tagValue=getValue(/*variable(Inout)*/)

- **Examples**

  Use in scripts:

```
function Script_1() {
//The value of INT variable Tag_1 is 100
//smartTags('Tag_1') -- object of variable Tag_1
let a = getValue(smartTags('Tag_1'));
//After the getValue is executed, a = 100
}
```

19.3.5 Linear Scaling

This function is used for linear scaling. Assign variable Y a value, which is calculated from the given value of variable X using the linear function Y = (a * X) + b.

Linear scaling can be performed on variables of integer, word, and decimal types.

- **Function format**

  linearScaling(/*variable(InOut)*/,/*variable(In)*/,/*constant(In)*/,/*constant(In)*/)

- **Related parameters**

Parameter	Description	Value
Y	Output	Configured as a variable only.
a	Parameter	Configured as a constant or a variable. The default value is constant 1.
X	Input	Configured as a variable only.
b	Parameter	Configured as a constant or a variable. The default value is constant 0.

- **Examples**

  Use in scripts:

```
function Script_1() {
//The default value of INT variable Tag_1 is 0, and the default value of INT variable Tag_2 is 100
//smartTags('Tag_1')--object of variable Tag_1; smartTags('Tag_2')--object of variable Tag_2
//Execute linearscaling to calculate the value of (2 * value of variable Tag_2) + 100, and write the value to variable Tag_1
linearScaling(smartTags('Tag_1'),2,smartTags('Tag2'),100);
//After the linearScaling is executed, the value of variable Tag_1 is 300
}
```

19.3.6 Inverse Linear Scaling

This function is used for inverse linear scaling. Assign variable X a value, which is calculated from the given value of variable Y using the linear function X = (Y - b) / a.

Inverse linear scaling can be performed on variables of integer, word, and decimal types.

- **Function format**

  inverseLinearScaling(/*variable(In)*/,/*variable(InOut)*/,/*constant(In)*/,/*constant(In)*/)

- **Related parameters**

Parameter	Description	Value
Y	Input	Configured as a variable only.
X	Output	Configured as a variable only.

Parameter	Description	Value
a	Parameter	Configured as a constant or a variable. The default value is constant 1.
b	Parameter	Configured as a constant or a variable. The default value is constant 0.

- **Examples**
  Use in scripts:

```
function Script_1() {
//The default value of INT variable Tag_1 is 0, and the default value of INT variable Tag_2 is 200
//smartTags('Tag_1'): object of variable Tag_1; smartTags('Tag_2'): object of variable Tag_2
//Execute inverseLinearScaling to calculate the value of (value of Tag_2 - 100) / 2, and write the result to variable Tag_1
inverseLinearScaling(smartTags('Tag_1'), smartTags('Tag_2'), 100, 2);
//After the inverseLinearScaling is executed, the value of variable Tag_1 is 50
}
```

19.3.7 Generating a Random Value

This function is used for generating a random value based on the preset conditions for the variable.

- **Random value range reference**
  - If the variable has upper and lower limits: A value will be randomly selected from this range.
  - If the variable has no upper or lower limits: The default range matching the variable's data type will be automatically used. For example: An integer will be randomly selected from [-32768, 32767] for INT variables.

    The configured range has higher priority than the default range.
- **Function format**

```
random(/*variable (In)*/)
```

- **Example**
  Use in scripts:

```
function Script_1() {
//Generate a random value for variableTag1, of which the configured range is [10, 100]
let a = random(smartTags('Tag1'));
//After successful execution, the value of variable Tag1 is randomly set to 55, a = 55
}
```

19.4 Screen

19.4.1 Activating a Screen

This function is used for activating a screen.

- **Function format**

```
activateScreen(/*screen name*/)
```

- **Related parameters**

Parameter	Description	Value
Screen name	Configured as "screen" or "variable": ◦ Screen: associated with overlay screens. ◦ Variable: associated with strings.	The default option is screen. The default value is null.

- **Example**

Use in scripts:

```
function Script_1() {
activateScreen("screen_1");
//Activate the specified screen based on the screen name "screen_1"
}
```

19.4.2 Activating a Screen by Number

This function is used for activating a screen by number.

- **Function format**

```
activateScreenByNumber(/*screen ID*/)
```

- **Related parameters**

Parameter	Description	Value
Screen ID	Configured as a variable of integer type.	The default value is null.

- **Example**

Use in scripts:

```
function Script_1() {
activateScreenByNumber(smartTags('ScreenID'));
//Open the specified screen based on the value of the variable ScreenID.
}
```

19.4.3 Activating the Previous Screen

This function is used for activating the previous screen.

- **Function format**

```
activatePreviousScreen()
```

- **Example**

Use in scripts:

```
function Script_1() {
activatePreviousScreen();
//Activate the previous screen
}
```

19.4.4 Showing the Popup Screen

This function is used to determine the display position of a popup screen. It determines the start position of the top-left corner of the popup screen based on the specified coordinate reference and automatically adjusts the position if the screen space is insufficient, ensuring the popup is fully displayed within the visible area.

- **Function format**

  showPopup(/*screen name*/,/*horizontal position*/,/*vertical position*/,/*bound variable mapping*/)

- **Related parameters**

Parameter	Description	Value
Popup screen name	Associated popup screen name.	The default value is null.
Position X	Center: horizontal center of the screen Left side of current graphic object: Leftmost border of the current graphic object Right side of current graphic object: Rightmost border of the current graphic object Custom: Specify an X-axis coordinate value	Center: -1 Left side of current graphic object: -2 Right side of current graphic object: -3 Custom: Within the screen boundaries and greater than zero.
Position Y	Center: Vertical center of the screen Current graphic object top: Topmost border of the current graphic object Current graphic object bottom: Bottommost border of the current graphic object Custom: Specify a Y-axis coordinate value	Center: -1 Current graphic object top: -2 Current graphic object bottom: -3 Custom: Within the screen boundaries and greater than zero.
Page parameter	Optional configuration, used to pass dynamic data or control commands to the popup screen.	When configuring the page parameters, associate a variable with each parameter.

- **Examples**

  Use in scripts:

  ◦ Define X-axis and Y-axis positions

```
function Script_1() {
//Show the popup screen
//"Popup screen 1", popup screen name
//100: X-axis position of the popup window
//200: Y-axis position of the popup window
//{"P_1":"Tag_1","P_2":"Tag_2","P_3":"Tag_3"} association of page parameters (P_1/2/3) and variables (Tag_1/2/3)
showPopup("popup screen 1",100,200,{"P_1":"Tag_1","P_2":"Tag_2","P_3":"Tag_3"});
}
```

◦ Determine X-axis and Y-axis positions based on current graphic object position

```
function Script_1() {
//Show the popup screen
//"Popup screen 1", popup screen name
//-2: X-axis position of the popup window is based on the leftmost border of the current graphic object
//-3: Y-axis position of the popup window is based on the bottommost border of the current graphic object
showPopup("Popup screen 1",-2,-3);
}
```

19.4.5 Hiding a Popup Screen

This function is used for hiding the specified popup screen.

- **Function format**

  hidePopup(/*screen name*/)

- **Related parameters**

Parameter	Description	Value
Screen name	Configured as "screen", which is associated with popup screens.	The default value is null.

- **Example**

  Use in scripts:

```
function Script_1() {
hidePopup("popup screen_1");
//Hide the popup screen based on the screen name "popup screen_1"
}
```

19.4.6 Hiding All Popup Screens

This function is used for hiding all popup screens.

- **Function format**

  hideAllPopups()

- **Related parameters**

  No input parameters.

- **Example**

  Use in scripts:

```
function Script_1() {
//Hide all popup screens
hideAllPopups();
}
```

19.4.7 Showing the Message Box

This function is used for showing the message box.

- **Function format**

```
showMessageBox(/*message content*/)
```

- **Related parameters**

Parameter	Description	Value
Message content	Configured as "constant" or "variable": Variable: associated with strings.	The default option is constant. The default value is null.

- **Example**

Use in scripts:

```
function Script_1() {
showMessageBox("Hello iFA");
//Show the message. The content in parentheses is "Hello iFA"
}
```

19.4.8 Getting Parameters from a Popup Screen

This function is used for getting page parameters from a popup screen.

- **Function format**

```
getParametersFromScreen(/*screen name*/)
```

- **Related parameters**

Parameter	Description	Value
Screen name	Configured as "screen", which is associated with popup screens.	The default value is null.

- **Example**

Use in scripts:

```
function Script_1() {
//Get page parameters from a popup screen, and return the result in the following data format:
{"P_1":"Tag_1","P_2":"Tag_2","P_3":"Tag_3"}
var lst = getParametersFromScreen("screen_2");
//Show each item in the returned list using message box
for (var item of lst)
{
showMessageBox(item);
}
}
```

19.5 Alarm

19.5.1 Acknowledging an Alarm in the Alarm View

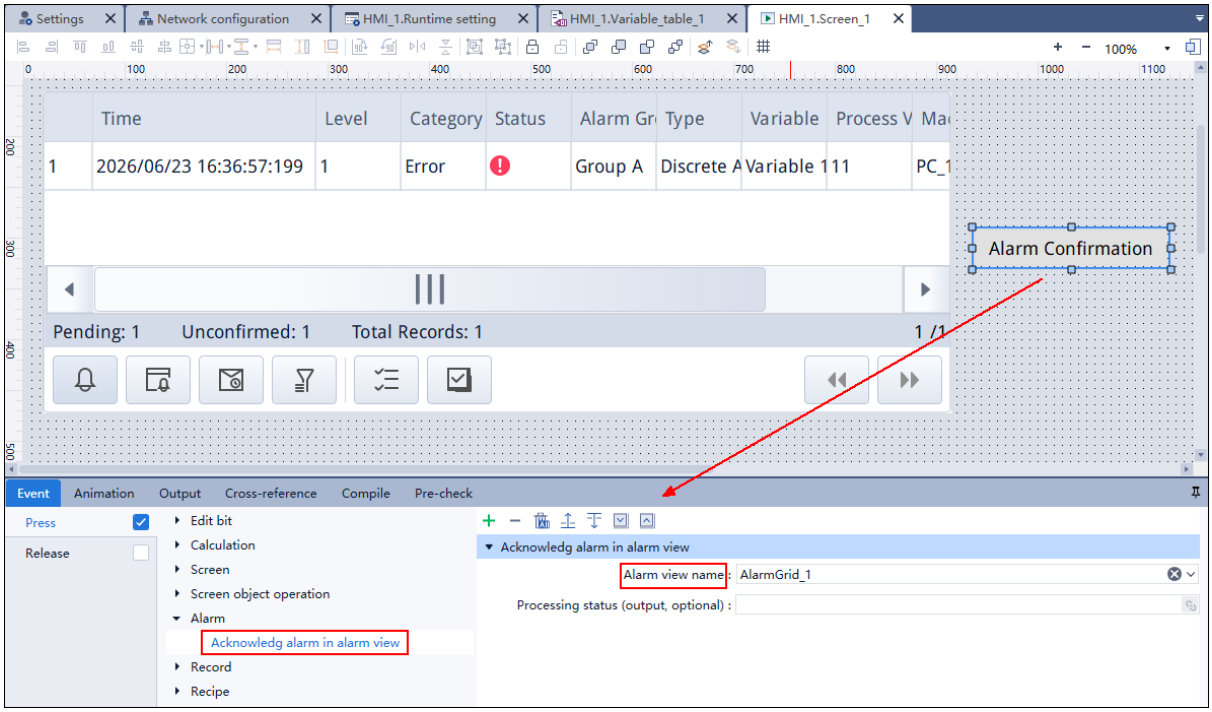
This system function is used to perform a "confirmation" operation on a selected alarm in the specified alarm view.

- **Restrictions**
  - This function can only be configured on the configuration screen. For example, associate the "press" event of a button with this function.
  - The graphic object (such as a button) for performing the confirmation operation must be located on the same screen as the target alarm view.
- **Related parameters**

Parameter	Description	Value
Alarm view name	Control name.	Configured as alarm view. The default value is null.
Status	Return a value.	2: Executing 4: Execution success 12: Execution failure This parameter is optional.

- **Example**

  Add an alarm view control and a button to the screen, and configure the trigger event as the "Alarm" function (as shown in the following figure). When the operator presses the button, the system acknowledges the currently selected alarm in the "Alarm view_1".



19.6 Record

19.6.1 Clearing a Record

This function is used for deleting all data in the specified record.

- **Function format**

  clearRecord(/*record type*/,/*record*/)

- **Related parameters**

Parameter	Description	Value
Record type	Data type of the record.	Configured as data record, alarm record, or operation record. The default option is data record.
Record	The created data record or alarm record.	The default value is null.

- **Examples**

  Use in scripts:

```
function Script_1() {
//Clear the data file generated by the record named "Data record_1" of data record type
clearRecord(1,"Data record_1")
//Clear the data file generated by the record named "Alarm record_1" of data record type
clearRecord(2,"Alarm record_1")
}
```

19.6.2 Starting Recording

This function is used for starting recording process values, operation logs, or alarms to the specified log file.

- **Function format**

  startRecord(/*record type*/,/*record*/)

- **Related parameters**

Parameter	Description	Value
Record type	Data type of the record.	Configured as data record, alarm record, or operation record. The default option is data record.
Record	Recorded data.	The default value is null.

- **Examples**

  Use in scripts:

```
function Script_1() {
//Start the record named "Data record_1" of data record type
startRecord(1, "Data record_1")
```

```
//Start the record named "Alarm record_1" of alarm record type
startRecord(2,"Alarm record_1")
}
```

19.6.3 Stopping Recording

This function is used for stopping recording process values or alarms to the specified log file.

- **Function format**

  stopRecord(/*record type*/,/*record*/)

- **Related parameters**

Parameter	Description	Value
Record type	Data type of the record.	Configured as data record, alarm record, or operation record. The default option is data record.
Record	Recorded data.	The default value is null.

- **Examples**

  Use in scripts:

```
function Script_1() {
//Stop the record named "Data record_1" of data record type
stopRecord(1, "Data record_1")
//Stop the record named "Alarm record_1" of alarm record type
stopRecord(2,"Alarm record_1")
}
```

19.6.4 Starting All Records

This function is used for re-establishing the connection between the monitoring system and the records to resume recording.

- **Function format**

  startAllRecords()

- **Example**

  Use in scripts:

```
function Script_1() {
//Start all configured records
startAllRecords();
}
```

19.6.5 Stopping All Records

This function is used for disconnecting the monitoring system from the records to stop recording.

- **Function format**

  stopAllRecords()

- **Example**

  Use in scripts:

```
function Script_1() {
//Stop all configured records
stopAllRecords();
}
```

19.6.6 Exporting a Data Record

This function is used for exporting the data of the specified record to the target path.

- **Function format**
  exportTagRecord(/*variable record name*/, /*storage location (SD card/USB1/USB2/local disk)*/, /*storage path*/, /*result variable object*/)
- **Related parameters**

Parameter	Description	Value
Data record	Enter the data record name	Match the name on the "record" node.
Storage location	Specify the location where the file is stored.	◦ 1: SD card of the HMI device ◦ 2: Device connected to USB1 port of the HMI device ◦ 3: Device connected to USB2 port of the HMI device ◦ 4: Local disk of the PC
Storage path	Specify the file storage path.	◦ The default storage path on the HMI device is "/TagRecord". ◦ The default storage path on the PC is "D:/TagRecord".
Result variable object	Return a value	◦ 2: Executing ◦ 4: Execution success ◦ 12: Execution failure This parameter is optional.

- **Example**
  Use in scripts:

```
function Script_1() {
//Export the specified data record to the specified path. "Data record 1": specified data record name; 1: storage in SD card; "/TagRecord": relative storage path. Write the execution result to variable Tag_1.
exportTagRecord(Data record_1",1,"/TagRecord",smartTags('Tag_1'))
}
```

19.6.7 Exporting an Alarm Record

This function is used for exporting the data of an alarm record to the target path.

- **Function format**
  exportAlarmRecord(/*alarm record name*/, /*storage location (SD card/USB1/USB2/local disk)*/, /*storage path*/, /*result variable object*/)
- **Related parameters**

Parameter	Description	Value
Data record	Enter the alarm record name	Match the name on the "record" node.
Storage location	Specify the location where the file is stored.	◦ 1: SD card of the HMI device ◦ 2: Device connected to USB1 port of the HMI device ◦ 3: Device connected to USB2 port of the HMI device ◦ 4: Local disk of the PC
Storage path	Specify the file storage path.	◦ The default storage path on the HMI device is "/AlarmRecord". ◦ The default path on the PC is "D:/AlarmRecord".
Result variable object	Return a value	◦ 2: Executing ◦ 4: Execution success ◦ 12: Execution failure This parameter is optional.

- **Example**
  Use in scripts:

```
function Script_1(){
//Export the specified alarm record to the specified path. "Alarm record_1": specified alarm record name; 1: storage in SD card; "/AlarmRecord": relative storage path. Write the execution result to variable Tag_1.
exportAlarmRecord("Alarm record_1",1,"/AlarmRecord",smartTags('Tag_1'))
}
```

19.6.8 Exporting an Operation Record

This function is used for exporting an operation record to the target path.

- **Function format**
  exportOperationRecord(/*operation record name*/, /*storage location (SD card/USB1/USB2/local disk)*/, /*storage path*/, /*result variable object*/)
- **Related parameters**

Table19-6-8-1

Parameter	Description	Value
Operation record name	Enter the operation record name	Match the name on the "record" node.

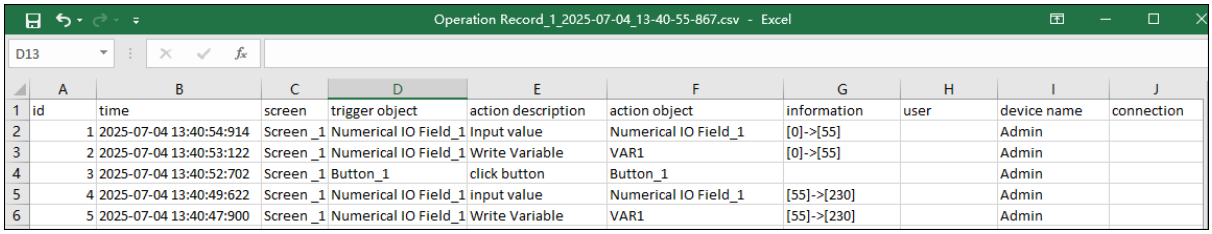
Parameter	Description	Value
Storage location	Specify the location where the file is stored.	◦ 1: SD card of the HMI device ◦ 2: Device connected to USB1 port of the HMI device ◦ 3: Device connected to USB2 port of the HMI device ◦ 4: Local disk of the PC
Storage path	Specify the file storage path.	◦ The default storage path on the HMI device is "/OperationRecord". ◦ The default storage path on the PC is "D:/OperationRecord".
Result variable object	Return a value	◦ 2: Executing ◦ 4: Execution success ◦ 12: Execution failure This parameter is optional.

- **Example**

  Use in scripts:

```
function Script_1() {
//Export an operation record to the specified path. "Operation record_1": specified operation record name; 1: storage in SD card; "/operationRecord": relative storage path. Write the execution result to variable Tag_1.
exportOperationRecord("operation record_1",1,"/OperationRecord",smartTags('Tag_1'))
}
```

  The exported data records are stored in table form at the specified path. The table is provided as follows.



Operation Record_1_2025-07-04_13-40-55-867.csv - Excel

D13

	A	B	C	D	E	F	G	H	I	J
1	id	time	screen	trigger object	action description	action object	information	user	device name	connection
2	1	2025-07-04 13:40:54:914	Screen _1	Numerical IO Field_1	Input value	Numerical IO Field_1	[0]->[55]		Admin	
3	2	2025-07-04 13:40:53:122	Screen _1	Numerical IO Field_1	Write Variable	VAR1	[0]->[55]		Admin	
4	3	2025-07-04 13:40:52:702	Screen _1	Button_1	click button	Button_1			Admin	
5	4	2025-07-04 13:40:49:622	Screen _1	Numerical IO Field_1	input value	Numerical IO Field_1	[55]->[230]		Admin	
6	5	2025-07-04 13:40:47:900	Screen _1	Numerical IO Field_1	Write Variable	VAR1	[55]->[230]		Admin	

19.6.9 Exporting Data Records by Time

This function is used for exporting and storing data records within a specific time period to the target path.

- **Function format**

  exportTagRecordByTime(/*variable record name*/,/*storage location(SD card/USB1/USB2/local disk)*/,/*storage path*/,/*start time*/,/*end time*/,/*result variable object*/)

- **Related parameters**

Parameter	Description	Value
Variable record name	Specify the data record	Match the name on the "record" node.

Parameter	Description	Value
Storage location	Specify the location where the file is stored.	◦ 1: SD card of the HMI device ◦ 2: Device connected to USB1 port of the HMI device ◦ 3: Device connected to USB2 port of the HMI device ◦ 4: Local disk of the PC
Storage path	Specify the file storage path.	◦ The default storage path on the HMI device is "/TagRecord". ◦ The default storage path on the PC is "D:/TagRecord".
Start time	Specify the time range for exporting data. Configured as DATETIME variable or fixed time.	The end time must be equal to or later than the start time. When the start time and end time are exactly the same [i.e., date and time (hour, minute, second) are identical], the exported file is empty.
End time		
Result variable object	Return a value	◦ 2: Executing ◦ 4: Execution success ◦ 12: Execution failure This parameter is optional.

- **Exported file**
  The exported file is in CSV format. The file naming format is: Data record name_export time (YYYY-MM-DD_HH-MM-SS-MS).
- **Example**
  Use in scripts:

```
function Script_1() {
//Export the data between TIME1 and TIME2 from Data record_1 to the path /TagRecord of the SD card of the IT9000 touchscreen, and write the result status to variable T1.
exportTagRecordByTime("Data record_1", 1, "/TagRecord", smartTags('TIME1'), smartTags('TIME2'), smartTags('T1'))
}
```

  The exported file  Record _1_2025-05-07_11-20-13-123.csv is stored in the path /TagRecord of the SD card.

19.7 Recipe

19.7.1 Creating a Recipe Data Record

This function is used for creating a recipe data record.

- **Function format**

  newDataRecord(/*recipe number*/, /*whether to pass the data record name through variable*/, /*data record name input variable number*/, /*data record name output variable number*/,/*data record number output variable number*/, /*whether to use the default value for the ingredient variable (true-initial value of the recipe ingredient/false-collected value of the ingredient variable)*/,/*result variable object*/)

- **Related parameters**

Parameter	Description	Value
Recipe number	Number of the recipe for which a data record is to be created.	Recipe or variable can be selected. The default option is recipe. INT variables are supported: SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT.
Variable input data record name	Whether to write new data record name through variable.	true: through variable false: through popup window
Data record name (input)		Supported variable types: String and Wstring. Data record name rules: any combination of Chinese characters, English letters, numbers, underlines (_), and middle hyphens (-), with a length of up to 64 characters.
Data record name (output)	Read data record name through variable.	Supported variable types: String and Wstring.
Data record number (output)	Read data record number through variable.	INT variables are supported: SINT, USINT, INT, UINT, DINT, UDINT, LINT, ULINT.
Value source	Specify the value source for each ingredient in the new data record.	true: Initial value of the recipe ingredient. false: Collected value of the ingredient variable.
Status	Return a value (optional).	2: Executing 4: Execution success 12: Execution failure

- **Example**

  Use in scripts:

```
function Script_1() {
//Create a data record for recipe 1.
```

```
//The data record name is input through popup window, and current values are used for each ingredient in the data record.
//The variable (Recipe1DataRecord3rdName_in) means inputting the name of the new data record.
//The variable (Recipe1DataRecord3rdName_out) means reading the name of the new data record.
//The variable (Recipe1DataRecord3rdID_out) means reading the number of the new data record.
newDataRecord(1,false,smartTags('Recipe1DataRecord3rdName_in'),smartTags('Recipe1DataRecord3rdName_out'),smartTags('Recipe1DataRecord3rdID_out'),false);
}
```

19.7.2 Deleting a Recipe Data Record

This function is used for deleting a recipe data record.

- **Function format**

  ```
  deleteDataRecord(/*recipe number*/,/*data record number*/,/*confirm*/,/*status*/)
  ```

- **Related parameters**

Parameter	Description	Value
Recipe number/name	The recipe name or corresponding number to be deleted.	Configured as recipe name or variable. The default option is recipe name. The default value is null.
Data record number	Data record number of the recipe to be deleted.	Configured as a constant or a variable. The default value is constant 1.
Confirm	Select On or Off.	On: When the operating system executes the deletion, a confirmation dialog appears. Off: When the operating system executes the deletion, the data is deleted directly. The default option is On.
Status	Return a value	2: Executing 4: Execution success 12: Execution failure This parameter is optional.

- **Example**

  Use in scripts:

  ```
  function Script_1() {
  //Delete the data record (with the recipe number 1 and data record number 2)
  //Confirm: "On". In this case, selecting "Confirm" means deleting the record when the popup window "Confirm to delete the data record" appears; selecting "Cancel" means not deleting the record
  //Confirm: "Off". In this case, the data record will be deleted directly without any popup window
  // Status: smartTags('Tag_1'), associated with variable Tag_1. When the data record is deleted successfully, Tag_1 returns 4; otherwise, Tag_1 returns 12
  ```

```
deleteDataRecord(1, 2, "On", smartTags('Tag_1'));
}
```

19.7.3 Setting a Recipe Data Record to PLC

This function is used for downloading the component values of a saved data record in the recipe view to the PLC.

- **Function format**

  ```
  setDataRecordToPLC(/*recipe number*/, /*data record number*/, /*variable indicating the status*/)
  ```

- **Related parameters**

Parameter	Description	Value
Recipe number/name	The recipe name or corresponding number from which the data record is to be set.	Configured as recipe name or variable. The default option is recipe name. The default value is null.
Data record number	The number of the data record to be set.	Configured as a constant or a variable. The default value is constant 1.
Status	Return a value.	2: Executing 4: Execution success 12: Execution failure This parameter is optional.

- **Example**

  Use in scripts:

  ```
  function Script_1() {
  //Set the specified data record of the specified recipe to the PLC
  //Status: smartTags('Tag_1'): associated with variable Tag_1. When the setting is successful, Tag_1 returns 4; otherwise, Tag_1 returns 12
  //Set the data record with recipe number 1 and data record number 2 in the recipe view to the PLC
  setDataRecordToPLC(1, 2, smartTags('Tag_1'))
  }
  ```

19.7.4 Getting a Recipe Data Record from PLC

This function is used for getting a recipe data record from the PLC and uploading it to the recipe view on the screen.

- **Function format**

  ```
  getDataRecordFromPLC(/*recipe number*/, /*data record number*/, /*override*/, /*status*/)
  ```

- **Related parameters**

Parameter	Description	Value
Recipe number/ name	The recipe name or corresponding number from which the data record is to be obtained.	Configured as recipe name or variable. The default option is recipe name. The default value is null.
Data record number	The number of the data record to be obtained.	Configured as a constant or a variable. The default value is constant 1.
Override	Whether to overwrite the current data record with the PLC value	◦ true: Overwrite directly ◦ false: Pop up a dialog box for re-confirmation. The default option is false.
Status	Return a value.	2: Executing 4: Execution success 12: Execution failure This parameter is optional.

- **Example**

  Use in scripts:

```
function Script_1() {
//Get the data record from the PLC to recipe view
// Overwrite: Perform re-confirmation before overwriting the data record.
//Status: smartTags('Tag_1'), associated with variable Tag_1. When the operation is successful, Tag_1 returns 4; otherwise, Tag_1 returns 12
//Get the data record from the PLC to recipe view and override the data record with recipe number 1 and data record number 2
getDataRecordFromPLC(1,2,"false",smartTags('Tag_1'))
}
```

19.7.5 Getting the Recipe Data Record Name

This function is used for getting the data record name (writing the given recipe and recipe data record name to the given variable).

- **Function format**

  getDataRecordName(/*recipe number*/,/*data record number*/,/*recipe name variable ID*/,/*data record name variable ID*/,/*status*/)

- **Related parameters**

Parameter	Description	Value
Recipe number/name	The recipe name or corresponding number from which the data record is to be obtained.	Configured as recipe name or variable. The default option is recipe name. The default value is null.

Parameter	Description	Value
Data record number	The number of the data record to be obtained.	Configured as a constant or a variable. The default value is constant 1.
Recipe name variable	The returned recipe name.	Displayed through variable value. The default value is null.
Data record name variable	The returned data record name.	Displayed through variable value. The default value is null.
Status	Return a value.	2: Executing 4: Execution success 12: Execution failure This parameter is optional.

- **Example**
  Use in scripts:
  ```
  function Script_1() {
  //Get the data record name of the specified data record in the specified recipe
  //Status: smartTags('Tag_3'), associated with variable Tag_3. When the operation is successful, Tag_2 returns 4; otherwise, Tag_2 returns 12
  //Get the data record name of the data record with recipe number 1 and data record number 2; Tag_1 returns the recipe name, and Tag_2 returns the recipe data record name
  getDataRecordName(1, 2, smartTags('Tag_1'), smartTags('Tag_2'), smartTags('Tag_3'))
  }
  ```

19.7.6 Exporting Recipe Data Records

This function exports the data records of a specified recipe to a designated location.

- **Function format**
  exportDataRecords(/*recipe number*/,/*storage location (SD card/USB1/USB2/local disk)*/,/*storage path*/,/*file name*/,/*encoding style*/, /*result variable object*/)
- **Related parameters**

Parameter	Description	Value
Recipe number	Specify the number of the recipe to be exported.	Configured as a constant or a variable.
Storage location	Specify the location where the file is stored.	◦ 1: SD card of the HMI device ◦ 2: Device connected to USB1 port of the HMI device ◦ 3: Device connected to USB2 port of the HMI device ◦ 4: Local disk of the PC

Parameter	Description	Value
Storage path	Specify the file storage path.	◦ The default storage path on the HMI device is "/DataRecord". ◦ The default path on the PC is "D:/DataRecord".
Exported file name	Specify the name of the exported file.	Configured as a constant or a variable. The default option is constant "DataRecord".
Encoding style	Specify the encoding style for the exported file.	Configured as GBK or UTF-8. The default option is GBK.
Result variable object	Return a value	◦ 2: Executing ◦ 4: Execution success ◦ 12: Execution failure This parameter is optional.

In addition to this function, the recipe data records can be exported through interface operations. For specific operations, see 13.6.6 Importing and Exporting a Recipe.

- **Example**

  Use in scripts:

```
function Script_1() {
//Export the data record with recipe number 1 to the path /DataRecord of the SD card and save it as the file DataRecord.csv in GBK encoding style. Tag_2 indicates the export operation status.
exportDataRecords(1, 1,"/DataRecord","DataRecord","GBK", smartTags('Tag_2') ) ;
}
```

19.7.7 Importing Recipe Data Records

This function is used for importing the data record file at the specified location into the device and displaying it in the recipe view.

- **Function format**

```
importDataRecords(/*storage location (SD card/USB1/USB2/local disk)*/,/*storage path*/,/*file name*/,/*result variable object*/)
```

- **Related parameters**

Parameter	Description	Value
Storage location	Specify the location where the file is stored.	◦ 1: SD card of the HMI device ◦ 2: Device connected to USB1 port of the HMI device ◦ 3: Device connected to USB2 port of the HMI device ◦ 4: Local disk of the PC

Parameter	Description	Value
Storage path	Specify the location where the file is stored.	◦ The default storage path on the HMI device is "/DataRecord". ◦ The default path on the PC is "D:/DataRecord".
File name	Specify the name of the file to be imported.	Configured as a constant or a variable. The default option is constant "DataRecord".
Result variable object	Return a value	◦ 2: Executing ◦ 4: Execution success ◦ 12: Execution failure This parameter is optional.

In addition to this function, the recipe data records can be imported through interface operations. For specific operations, see 13.6.6 Importing and Exporting a Recipe.

- **Example**

  Use in scripts:

```
function Script_1() {
//Import the file Recipe1.csv from the USB flash disk connected to USB1 port into the device. The operation status is reflected by the Tag_2 variable.
importDataRecords(2,"/DataRecord","Recipe1", "Datarecord", smartTags('Tag_2'));
}
```

19.8 User Management

19.8.1 Displaying the Login Dialog

This function is used for popping up the login dialog.

- **Function format**

```
loginDialog(/*result variable object (InOut) */)
```

- **Related parameters**

Parameter	Description	Value
Status	Return a value.	2: Executing 4: Execution success 12: Execution failure This parameter is optional.

- **Example**

  Use in scripts:

```
function Script_1() {
//Pop up the login dialog
```

```
//Status: smartTags('Tag_1'): associated with variable Tag_1. When the login dialog pops up successfully, Tag_1 returns 4; otherwise, Tag_1 returns 12
//Pop up the login dialog
loginDialog(smartTags('Tag_1'))
}
```

19.8.2 Login

This function is used for user login.

- **Function format**

  ```
  login(/*username(In)*/,/*password(In)*/,/*result variable object(InOut)*/)
  ```

- **Related parameters**

Parameter	Description	Value
Username	The username to log in.	Configured as a constant or a variable. The default option is constant. The default value is null.
Password	Password for the user.	Configured as a constant or a variable. The default option is constant. The default value is null.
Status	Return a value.	2: Executing 4: Execution success 12: Execution failure This parameter is optional.

- **Example**

  Use in scripts:

  ```
  function Script_1() {
  //User logs in
  //Status: smartTags('Tag_1'): associated with variable Tag_1. When the user logs in successfully, Tag_1 returns 4; otherwise, Tag_1 returns 12
  //Username and password can be set using variables or constants
  //User name: zhangsan, password: Aa123456
  login("zhangsan","Aa123456",smartTags('Tag_1'))
  //username = Tag_2, password = Tag_3
  login(smartTags('Tag_2'),smartTags('Tag_3'),smartTags('Tag_1'))
  }
  ```

19.8.3 Logout

This function is used for user logout.

- **Function format**

```
logout(/*result variable object(Inout)*/)
```

- **Related parameters**

Parameter	Description	Value
Status	Return a value.	2: Executing 4: Execution success 12: Execution failure This parameter is optional.

- **Example**

Use in scripts:

```
function Script_1() {
//The user logs out
//Status: smartTags('Tag_1'): associated with variable Tag_1. When the user logs out successfully, Tag_1 returns 4; otherwise, Tag_1 returns 12
//Current user logs out
logout(smartTags('Tag_1'))
}
```

19.8.4 Getting the Username

This function is used for getting information about the currently logged-in user.

- **Function format**

```
getUserName(/*username*/,/*status*/,/*result variable object*/)
```

- **Related parameters**

Parameter	Description	Value
Username	Return the currently logged-in username.	The default value is null.
Status	Return a value.	2: Executing 4: Execution success 12: Execution failure This parameter is optional.
Additional information	Return a value.	2: Only when the function is executed successfully with valid user information being returned. 4: Only when the function is executed successfully with no username being returned. This parameter is optional.

- **Example**

Use in scripts:

```
function Script_1() {
```

```
//Get the username of the logged-in user
//Status: smartTags('Tag_1')
//Additional information: smartTags('Tag_3')
//If a user logs in and the desired information is obtained successfully, Tag_1 returns 4 and Tag_3 returns 2
//If no user logs in, Tag_1 returns 4 and Tag_3 returns 4
//If the desired information is not obtained, Tag_1 returns 12 and Tag_3 returns 0
//Get the username of the currently logged-in user. If successful, return the username to variable Tag_2
getUserName(smartTags('Tag_2'), smartTags('Tag_1'), smartTags('Tag_3'))
}
```

19.8.5 Switching the User

This function is used for switching the logged-in user.

- **Function format**

  ```
  switchUser(/*username(In)*/, /*password(In)*/, /*result variable object(Inout)*/)
  ```

- **Related parameters**

Parameter	Description	Value
Username	The username to switch.	Configured as a constant or a variable. The default option is constant. The default value is null.
Password	Password for the user.	Configured as a constant or a variable. The default option is constant. The default value is null.
Status	Return a value.	2: Executing 4: Execution success 12: Execution failure This parameter is optional.

- **Example**

  Use in scripts:

```
function Script_1() {
//Switch the logged-in user
//Status: smartTags('Tag_1'): associated with variable Tag_1. When the user is switched successfully, Tag_1 returns 4; otherwise, Tag_1 returns 12
//Username and password can be set using variables or constants
//Switch the logged-in user to lisi with password Bb123456
switchUser("lisi", "Bb123456", smartTags('Tag_1'))
//Switch the logged-in user to Tag_2 with password Tag_3
switchUser(smartTags('Tag_2'), smartTags('Tag_3'), smartTags('Tag_1'))
}
```

19.8.6 Changing the Password

This function is used for changing the password.

- **Function format**

  changePassword(/*username(In)*/, /*old password(In)*/, /*new password(In)*/, /*result variable object(Inout)*/)

- **Related parameters**

Parameter	Description	Value
Username	Username of the user whose password is to be changed.	Configured as a constant or a variable. The default option is constant. The default value is null.
Old password	Old password for the user.	Configured as a constant or a variable. The default option is constant. The default value is null.
New password	New password for the user.	Configured as a constant or a variable. The default option is constant. The default value is null. The password must meet the complexity rules defined in "Safety settings -> Settings". For details, see 9.5 Setting Password Policies.
Status	Return a value.	2: Executing 4: Execution success 12: Execution failure This parameter is optional.

- **Example**

  Use in scripts:

```
function Script_1() {
//Change the user password
//Status: smartTags('Tag_1'): associated with variable Tag_1. When the change is successful, Tag_1 returns 4; otherwise, Tag_1 returns 12
//Username, old password, and new password can be configured using constants or variables
//Change the password of the username "zhangsan" from old password Aa123456 to new password Bb123456
changePassword("zhangsan", "Aa123456", "Ba123456", smartTags('Tag_1'))
//Change the password of the username Tag_2 from old password Tag_3 to new password Tag_4
//smartTags('Tag_2'), smartTags('Tag_3'), smartTags('Tag_4') are variables corresponding to username, old password, and new password respectively. Use these variables to change the user password
```

```
changePassword(smartTags('Tag_2'), smartTags('Tag_3'), smartTags('Tag_4'), smartTags('Tag_1'))
}
```

19.8.7 Exporting User Data

This function is used for exporting user data to the specified path.

The exported data includes information that can be configured in the user view, such as user IDs, passwords, and deactivation time.

- **Function format**

  exportUserData(/*storage location (SD card /USB1/USB2/local disk)*/,/*storage path*/,/*file name*/,/*result variable object*/)

- **Related parameters**

Parameter	Description	Value
Storage location	Specify the location where the file is stored.	◦ 1: SD card of the HMI device ◦ 2: Device connected to USB1 port of the HMI device ◦ 3: Device connected to USB2 port of the HMI device ◦ 4: Local disk of the PC
Storage path	Specify the file storage path.	◦ The default storage path on the HMI device is "/UserData". ◦ The default storage path on the PC is "D:/UserData".
File name	Specify the name of the exported file.	Configured as a constant or a variable. The default option is constant "UserData".
Result variable object	Return a value	◦ 2: Executing ◦ 4: Execution success ◦ 12: Execution failure This parameter is optional.

- **Example**

  Use in scripts:

```
function Script_1() {
//Export user data to UserData.csv, save it to the /UserData directory on the USB flash disk connected to USB2 port. The operation status is reflected by the Tag_1 variable.
exportUserData(3,"/UserData ","UserData",SmartTags('Tag_1'));
}
```

19.8.8 Importing User Data

This function is used for importing the specified user data file into the device.

- **Function format**

  importUserData(/*storage location (SD card /USB1/USB2/local disk)*/,/*storage path*/,/*file name*/,/*result variable object*/)

- **Related parameters**

Parameter	Description	Value
Storage location	Specify the location where the file is stored.	◦ 1: SD card of the HMI device ◦ 2: Device connected to USB1 port of the HMI device ◦ 3: Device connected to USB2 port of the HMI device ◦ 4: Local disk of the PC
Storage path	Specify the file storage path.	◦ The default storage path on the HMI device is "/UserData". ◦ The default storage path on the PC is "D:/UserData".
File name	Specify the name of the exported file.	Configured as a constant or a variable. The default option is constant "UserData".
Result variable object	Return a value	◦ 2: Executing ◦ 4: Execution success ◦ 12: Execution failure This parameter is optional.

- **Precautions**

  After 19.8.7 exporting user data from a device, ensure all of the following conditions are met to import the data successfully:

  ◦ The number and names of all roles on the source and target devices, including system roles and user-defined roles, are exactly the same.

  ◦ Each user role also exists on the target device with the same name.

  ◦ Permission differences between roles with the same name do not prevent import; the imported role will inherit the original permissions on the target device.

- **Examples**

  Use in scripts:

```
function Script_1() {
//Import the UserData.csv from the /UserData directory of the USB flash disk connected to USB2 port to the device. The operation status is reflected by the Tag_1 variable.
importUserData(3,"/UserData","UserData",SmartTags('Tag_1'));
}
```

19.9 Configuration

19.9.1 Opening the Control Panel

This function is used for opening the touchscreen control panel.

- **Conditions**

  This function is available only on touchscreens, not on PCs. The control panel cannot be opened on a simulated touchscreen of a PC; it can only be opened on an actual touchscreen.

- **Function format**

  openControlPanel()

- **Example**

  Use in scripts:

```
function Script_1() {
//Open the control panel
openControlPanel()
}
```

19.9.2 Setting the IP Address

This function is used for setting the IP address for the HMI.

- **Conditions**

  This function is available only on touchscreens, not on PCs. It can be executed only on an actual touchscreen, not on a simulated touchscreen of a PC.

- **Function format**

  setIPAddress(/*whether to automatically get the IP (true/false)*/, /*network port selection (0--Port 1/1--Port 2)*/, /*IP address (string): "0.0.0.0"*/, /*subnet mask (string): "0.0.0.0"*/, /*gateway (string): "0.0.0.0"*/, /*result variable object (InOut)*/)

- **Related parameters**

Parameter	Description	Value
Whether to automatically get the IP	Whether to automatically get the IP address.	Configured as a constant. Select {Yes} or {No} from the drop-down list. The default option is {Yes}. ◦ {Yes}: The IP address, subnet mask, and gateway parameters are set to unavailable; ◦ {No}: All parameters are set to available.
Network port selection	Match the physical network port supported by the device.	For IT9000, both Port 1 and Port 2 are supported. The default option is Port 1.

Parameter	Description	Value
IP address	Target IP address to be set by the user.	Configured as a variable or constant. The data type is String. The default option is constant. The default value is null.
Subnet mask	Target subnet mask to be set by the user.	Configured as a variable or constant. The data type is String. The default option is constant. The default value is null.
Gateway	Target gateway to be set by the user.	Configured as a variable or constant. The data type is String. The default option is constant. The default value is null.
Status	Return a value.	Configured as a variable. The data types are Int and UInt. Description of return results: 4: Execution success 12: Execution failure This parameter is optional.

- **Example**

  Use in scripts:

```
function Script_1() {
//Set a static IP. false: not automatically get the IP; 0: Port 1; "0.0.0.127": IP address; "255.255.255.0": Subnet mask; "0.0.0.1": Gateway; smartTags('Tag_1"): object of variable Tag_1
setIPAddress(false,0, "0.0.0.127", "255.255.255.0", "0.0.0.1", smartTags('Tag_1'))
//After the setIPAddress is executed, the result is written to variable Tag_1
//Set a dynamic IP. true: Automatically get the IP; 0: Port 1; smartTags('Tag_1'): object of variable Tag_1
setIPAddress(false, 0, "", "", "", smartTags('Tag_1'))
//After the setIPAddress is executed, the result is written to variable Tag_1
}
```

19.9.3 Setting the Screensaver Time

This function is used for setting the screensaver time.

- **Conditions**

  This function is available only on touchscreens, not on PCs. It can be executed only on an actual touchscreen, not on a simulated touchscreen of a PC.

- **Function format**

  setScreenSaverTime(/*screensaver time (min)*/)

- **Related parameters**

Parameter	Description	Value
Screensaver time	The time interval before the touchscreen switches to the screensaver mode.	Configured as a constant or variable. Only integers can be entered. The constant range is 0-255. The default value is 3, and the unit is minutes.

- **Example**

  Use in scripts:

```
function Script_1() {
//Set the screensaver time to 5 min
setScreenSaverTime(5)
//Set the screensaver time to change based on the value of INT variable Tag_1, smartTags('Tag_1'): object of variable Tag_1
setScreenSaverTime(smartTags('Tag_1'))
}
```

19.9.4 Setting the Backlight Brightness

This function is used for setting the backlight brightness.

- **Conditions**

  This function is available only on touchscreens, not on PCs. It can be executed only on an actual touchscreen, not on a simulated touchscreen of a PC.

- **Function format**

```
setBrightness(/*backlight brightness value (0-100)*/)
```

- **Related parameters**

Parameter	Description	Value
Backlight brightness value	Backlight brightness value of the touchscreen.	Configured as a constant or variable. Only integers can be entered. The constant range is 0-100. The default value is 90.

- **Example**

  Use in scripts:

```
function Script_1() {
//Set the backlight brightness to 80
setBrightness(80)
//Set the backlight brightness to change based on the value of INT variable Tag_1, smartTags('Tag_1') -- object of variable Tag_1
setBrightness(smartTags('Tag_1'))
}
```

19.9.5 Setting the Backlight Mode

This function is used for setting the backlight mode.

- **Conditions**

  This function is available only on touchscreens, not on PCs. It can be executed only on an actual touchscreen, not on a simulated touchscreen of a PC.

- **Function format**

  setBacklightMode(/*Set backlight mode*/)

- **Related parameters**

Parameter	Description	Value
Setting the Backlight Mode	Set whether to enable the backlight mode.	Configured as Enable or Disable. The default option is Enable.

- **Example**

Use in scripts:

```
function Script_1() {
//Enable the backlight mode
setBacklightMode(true)
//Disable the backlight mode
setBacklightMode(false)
}
```

19.9.6 Setting the Backlight Time

This function is used for setting the backlight time.

- **Conditions**

  This function is available only on touchscreens, not on PCs. It can be executed only on an actual touchscreen, not on a simulated touchscreen of a PC.

- **Function format**

  setBacklightTime(/*Backlight off time (min)*/)

- **Related parameters**

Parameter	Description	Value
Backlight off time	Backlight off time of the touchscreen.	Configured as a constant or variable. Only integers can be entered. The constant range is 0–255. The default value is 5, and the unit is minutes.

- **Example**

Use in scripts:

```
function Script_1() {
//Set the backlight time to 5 min
```

```
setBacklightTime(5)
//Set the backlight time to change based on the value of INT variable Tag_1, smartTags('Tag_1'): object of variable Tag_1
setBacklightTime(smartTags('Tag_1'))
}
```

19.9.7 Setting the LED Mode

This function is used for setting the LED mode.

- **Conditions**

  This function is available only on touchscreens, not on PCs. It can be executed only on an actual touchscreen, not on a simulated touchscreen of a PC.

- **Function format**

  setLEDMode(/*LED switch (0--Off/1--On)*/)

- **Related parameters**

Parameter	Description	Value
LED	Set the LED color of the touchscreen.	1--On (yellow), 0--Off (green). The default value is 1.

- **Examples**

  Use in scripts:

```
function Script_1() {
//Set the LED to yellow
setLEDMode(1)
//Set the LED to green
setLEDMode(0)
}
```

19.9.8 Playing a Beep

This function is used for playing a single beep.

- **Conditions**

  This function is available only on touchscreens, not on PCs. It can be executed only on an actual touchscreen, not on a simulated touchscreen of a PC.

- **Function format**

  beep()

- **Example**

  Use in scripts:

```
function Script_1() {
//Calling the beep function triggers the beeper on the device
beep()
}
```

19.9.9 Setting the Sound Mode

This function is used for setting the sound mode. After the setting, both click sound and continuous unacknowledged alarm sounds will follow the defined mode.

- **Conditions**

  This function is available only on touchscreens, not on PCs. It can be executed only on an actual touchscreen, not on a simulated touchscreen of a PC.

- **Function format**

  setSoundMode(/*sound mode (0--Toggle/1--On/2--Off)*/)

- **Related parameters**

Parameter	Description	Value
Sound mode	The On/Off state of the sound.	Configured as Toggle, On, or Off. The default option is Toggle. ◦ Off: Neither the click sound nor the unacknowledged alarm sound can be played. ◦ On: Both click sound and unacknowledged alarm sound can be played. ◦ Toggle: Each execution of this function toggles the sound On/Off state once.

- **Examples**

  Use in scripts:

```
function Script_1() {
//Set the sound mode to be Toggle
setSoundMode(0)
//Set the sound mode to be On
setSoundMode(1)
//Set the sound mode to be Off
setSoundMode(2)
}
```

19.9.10 Restart

This function is used for restarting the operating system of the touchscreen.

- **Conditions**

  This function is available only on touchscreens, not on PCs. It can be executed only on an actual touchscreen, not on a simulated touchscreen of a PC.

- **Function format**

restart()

- **Example**
  Use in scripts:
  ```
  function Script_1() {
  //Restart the device
  restart()
  }
  ```

19.9.11 Safely Removing the Hardware

This function is used for safely removing the hardware.

- **Conditions**

  This function is available only on touchscreens, not on PCs. It can be executed only on an actual touchscreen, not on a simulated touchscreen of a PC.

- **Function format**

  safelyRemoveHardware(/*hardware selection (0--SD card/1--USB flash disk)*/, /*result variable object (InOut)*/)

- **Related parameters**

Parameter	Description	Value
Hardware selection	Select the hardware storage device.	Configured as SD card or USB flash disk. The default option is SD card. For IT9000, SD card, USB1, and USB2 are supported.
Result variable object	Return a value.	4: Execution success 12: Execution failure This parameter is optional.

- **Example**
  Use in scripts:
  ```
  function Script_1() {
  //Safely remove the hardware device
  //smartTags('Tag_1'): object of variable Tag_1;
  safelyRemoveHardware(0, smartTags('Tag_1'))
  //After the safelyRemoveHardware is executed, the result is written to variable Tag_1
  }
  ```

19.9.12 Setting the Language

This function is used for dynamically switching the text language displayed on the screen while the touchscreen device is running. This feature applies only to the text on user screens and does not affect the text displayed on the touchscreen control panel.

- **Function format**
  setLanguage(1)

- **Related parameters**

Parameter	Description	Value
Language	Specify the target language	Check 13.17.2 Runtime Configuration to confirm available languages and their numbers. ◦ Language number: Enter the target language number. Each language corresponds to a unique number. ◦ "-1": Switch languages in a loop according to the order specified in the Runtime Configuration. If the entered number is outside the range of available languages in the 13.17.2 Runtime Configuration, it is an invalid value and the system will not respond. You can also dynamically switch the language number using the SINT, INT, DINT, or LINT variable.
Status	Return a value.	4: Execution success 12: Execution failure This parameter is optional.

- **Examples**

  Use in scripts:

```
setLanguage(2057) //Set "English (UK)" as the fixed target language.
setLanguage(smartTags('Language'), smartTags('Status')) // Switch the language according to the value of variable Language and return the execution result to variable Status.
setLanguage(-1) //The target language is not fixed; the language scope and order are specified in the Runtime Configuration.
```

19.9.13 Unlocking Installments

This function is used to manage the unlock installment range and validate the installment password under the device's installment payment strategy during operation.

- **Function format**

```
unlockTerm(/*installment selection (1--Current installment / 2--All installments)*/,/*installment password variable object (InOut) **/)
```

- **Related parameters**

Parameter	Description	Value
Mode	Installment unlocking option	Current installment: the installment about to be unlocked. All installments: all installments to be unlocked under the installment payment strategy.
Password	Installment unlocking password	Supported variable types: String and Wstring.

- **Examples**

  Use in scripts:

```
function Script_1() {
```

```
//Dynamically unlock the next installment and verify the accuracy of the password entered for the variable "password".
unlockTerm(1,smartTags('password'));
}
```

19.9.14 Triggering Installments in Advance

This function allows early unlocking of the installment that is about to be unlocked or all subsequent installments not yet unlocked in the active installment payment strategy of a device.

- **Conditions**
  The installment payment strategy contains installments to be unlocked.
- **Function format**
  advanceTriggerTerm()
- **Examples**
  Use in scripts:

```
function Script_1() {
//Trigger the installment unlocking
advanceTriggerTerm();
}
```

19.9.15 Minimizing the Software

This function is used to minimize the software.

- **Conditions**

This function is available only on PCs.

- **Function format**

minimizeWindow()

- **Example**
  Use in scripts:

```
function Script_1 () {
//Minimize the software
minimizeWindow()
}
```

19.9.16 Exiting the Software

This function is used to exit the software.

- **Conditions**

This function is available only on PCs.

- **Function format**

exitServiceManager()

- **Example**
  Use in scripts:

```
function Script_1() {
//Exit the software
exitServiceManager()
}
```

19.10 Date and Time

19.10.1 Setting the Date and Time

This function is used for setting the HMI date and time, including year, month, day, hour, minute, and second.

- **Conditions**

  This function is available only on touchscreens, not on PCs. It can be executed only on an actual touchscreen, not on a simulated touchscreen of a PC.

- **Function format**

  setDateTime(/*year*/,/*month*/,/*day*/,/*hour*/,/*minute*/,/*second*/)

- **Related parameters**

Parameter	Description	Value
Year	Configured as Int or UInt variable.	The default value is null.
Month	Configured as Int or UInt variable.	The default value is null.
Day	Configured as Int or UInt variable.	The default value is null.
Hour	Configured as Int or UInt variable.	The default value is null.
Minute	Configured as Int or UInt variable.	The default value is null.
Second	Configured as Int or UInt variable.	The default value is null.

- **Precautions**
  When the currently logged-in account is locked, the system does not allow online changes to the date and time.

- **Example**
  Use in scripts:

```
function Script_1() {
//Set the HMI date and time, including year, month, day, hour, minute, and second. This function is supported by touchscreens only
//smartTags('Tag_1'), variable bound to the year. After the setting, the year of the current system is the current value of Tag_1
//smartTags('Tag_2'), variable bound to the month. After the setting, the month of the current system is the current value of Tag_2
//smartTags('Tag_3'), variable bound to the day. After the setting, the day of the current system is the current value of Tag_3
//smartTags('Tag_4'), variable bound to the hour. After the setting, the hour of the current system is the current value of Tag_4
//smartTags('Tag_5'), variable bound to the minute. After the setting, the minute of the current system is
```

```
the current value of Tag_5
//smartTags('Tag_6'), variable bound to the second. After the setting, the second of the current system is the current value of Tag_6
setDateTime(smartTags('Tag_1'),smartTags('Tag_2'),smartTags('Tag_3'),smartTags('Tag_4'),smartTags('Tag_5'),smartTags('Tag_6'));
}
```

19.10.2 Setting the Time Zone

This function is used for setting the HMI time standard (Universal Time Coordinated, UTC).

- **Conditions**

  This function is available only on touchscreens, not on PCs. It can be executed only on an actual touchscreen, not on a simulated touchscreen of a PC.

- **Function format**

  setTimeZone(/*time zone (UTC)*/)

- **Related parameters**

Parameter	Description	Value
Time zone (UTC)	Configured as a constant. Select an option from the drop-down list.	The default option is UTC.

- **Precautions**

  When the currently logged-in account is locked, the system does not allow online changes to the time zone.

- **Example**

  Use in scripts:

```
function Script_1() {
//Set the HMI time standard as UTC
//After the setting, the system's current time zone changes to the selected time zone. See the parameter settings in the scripts
setTimeZone("international date line (UTC-12:00)");
}
```

19.10.3 Setting the Date

This function is used for setting the HMI date, including the year, month, and day.

- **Conditions**

  This function is available only on touchscreens, not on PCs. It can be executed only on an actual touchscreen, not on a simulated touchscreen of a PC.

- **Function format**

  setDate(/*year*/, /*month*/, /*day*/)

- **Related parameters**

Parameter	Description	Value
Year	Configured as Int or UInt variable.	The default value is null.
Month	Configured as Int or UInt variable.	The default value is null.
Day	Configured as Int or UInt variable.	The default value is null.

- **Precautions**
  When the currently logged-in account is locked, the system does not allow online changes to the date.
- **Example**
  Use in scripts:

```
function Script_1() {
//Set the HMI date, including year, month, and day. This function is supported by touchscreens only
//smartTags('Tag_1'), variable bound to the year. After the setting, the year of the current system time is the current value of Tag_1
//smartTags('Tag_2'), variable bound to the month. After the setting, the month of the current system time is the current value of Tag_2
//smartTags('Tag_3'), variable bound to the day. After the setting, the day of the current system time is the current value of Tag_3
setDate(smartTags('Tag_1'), smartTags('Tag_2'), smairtTags('Tag_3'));
}
```

19.10.4 Setting the Time

This function is used for setting the HMI time, including hour, minute, and second.

- **Conditions**

  This function is available only on touchscreens, not on PCs. It can be executed only on an actual touchscreen, not on a simulated touchscreen of a PC.

- **Function format**

  ```
  setTime(/*hour*/,/*minute*/,/*second*/)
  ```

- **Related parameters**

Parameter	Description	Value
Hour	Configured as Int or UInt variable.	The default value is null.
Minute	Configured as Int or UInt variable.	The default value is null.
Second	Configured as Int or UInt variable.	The default value is null.

- **Precautions**
  When the currently logged-in account is locked, the system does not allow online changes to the time.
- **Example**
  Use in scripts:

```
function Script_1() {
//Set the HMI time, including hour, minute, and second
//smartTags('Tag_4'), variable bound to the hour. After the setting, the hour of the current system time is the current value of Tag_4
```

```
//smartTags('Tag_5'), variable bound to the minute. After the setting, the minute of the current system time is the current value of Tag_5
//smartTags('Tag_6'), variable bound to the second. After the setting, the second of the current system time is the current value of Tag_6
setTime(smartTags('Tag_4'), smartTags('Tag_5'), smartTags('Tag_6'));
}
```

19.10.5 Getting the Date and Time

This function is used for writing the latest date and time to a variable.

- **Function format**

```
getDateTime(/*variable (Inout)*/)
```

- **Related parameters**

Parameter	Description	Value
Variable	Configured as a variable. The data type is DateTime.	The default value is null.

- **Example**

Use in scripts:

```
function Script_1() {
//Assign the current system time to the datetime variable
//After this function is executed, the value of variable Tag_8 is the current system time, which can be viewed in the date time domain
getDateTime(smartTags('Tag_8'));
}
```

19.10.6 Getting the Timestamp

This function is used for getting the current timestamp of the CPU and writing it to a variable.

- **Function format**

```
getTimeStamp(/*variable (InOut)*/)
```

- **Related parameters**

Parameter	Description	Value
Variable	Output variable for the timestamp.	The default value is null.

- **Example**

Use in scripts:

```
function Script_1() {
//Get the current timestamp of the system and write it to a Lint or ULINT variable
//smartTags('Tag_19'): After this function is executed, the value of variable Tag_19 is the current timestamp of the system, accurate to milliseconds
getTimeStamp(smartTags('Tag_19'));
}
```

19.11 Print

19.11.1 Screenshot

The screenshot function is used for capturing the current screen image. The screenshot area on a monitor is the display area of the monitoring software (excluding device borders); the screenshot area on a touchscreen is the entire screen display area.

Screenshot files are named according to the rule "screenshot_YYYY-MM-DD_HH-MM-SS-MS" and stored in PNG format.

- **Function format**

  screenshot(/*storage location*/, /*storage path*/, /*status*/)

- **Related parameters**

Parameter	Description	Value
Storage location	The supported storage locations include SD card and USB.	SD card, USB: The number of stored screenshots depends on the file size and available storage space. If the storage is full, the screenshot operation will fail. ◦ 1: SD card of IT9000 device ◦ 2: Device connected to USB1 port of IT9000 device ◦ 3: Device connected to USB2 port of IT9000 device
Storage path	Specify the storage path for the screenshot files. Both constants and variables are supported, but variables cannot be used within JavaScript scripts.	◦ Constant: fixed value. The relative path must be entered. Enter the folder name only, such as "/screenShot", rather than the full path. If the folder titled Screenshot does not exist, the system will create it automatically. ◦ Variable: String variables are supported.
Status	Return the result of the screenshot operation, represented by INT or UINT variable value.	2: Executing 4: Execution success 12: Execution failure This parameter is optional.

- **Example**

  Use in scripts:

```
//2: Take a screenshot and save it to USB1 of IT9000
// "/screenShot": storage path
// smartTags('status_1'): associated with variable status_1. When the operation is successful, status_1 returns 4; otherwise, status_1 returns 12

screenshot(2,"/screenShot",smartTags('status_1'));
```

19.12 VNC Operations

19.12.1 Setting the VNC Enable

19.12.2 Setting the VNC Username

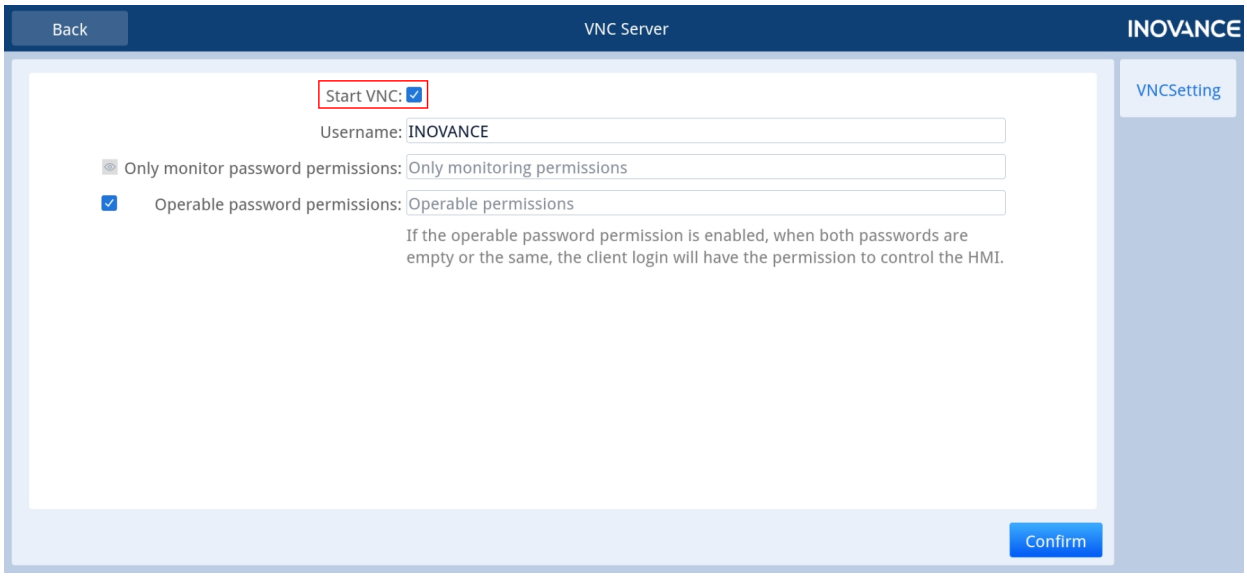
19.12.3 Setting the VNC View-Only Password

19.12.4 Setting the VNC Control Password

19.12.5 Setting the VNC Control Permission

19.12.1 Setting the VNC Enable

The setVNCEnable function is used for controlling whether to enable the VNC server. It has the same function as the Enable VNC option on the hardware touchscreen.



- **Function format**

  setVNCEnable(/*enable flag*/, /*status*/)

- **Related parameters**

Parameter	Description	Value
Enable flag	Enable command	Constants: Enable (True), Disable (False) Variables: Bool type
Status	Return a value	2: Executing 4: Execution success 12: Execution failure This parameter is optional.

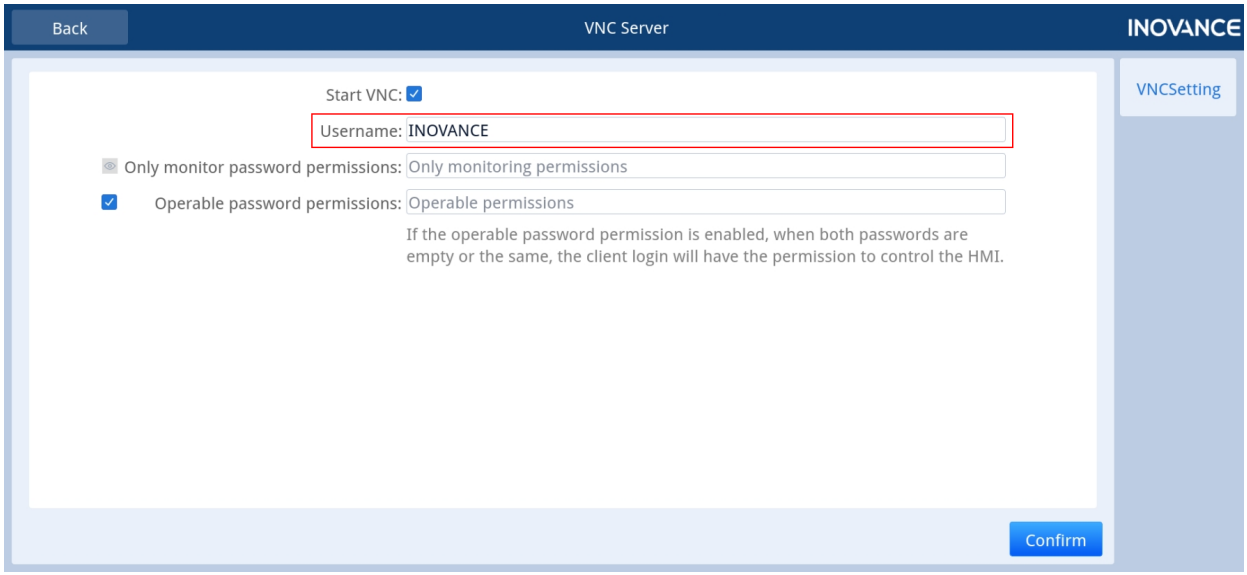
- **Example**

  Use in scripts:

```
//Enable the VNC server and return the execution result to the variable Status.
setVNCEnable(true, smartTags('status'));
```

19.12.2 Setting the VNC Username

The setVNCName function is used for specifying the username for accessing the VNC server. It has the same function as the "Username" option on the hardware touchscreen.



- **Function format**

  setVNCName(/*username*/,/*status*/)

- **Related parameters**

Parameter	Description	Value
Username	Required field. Up to 32 English characters.	Constant or String variable
Status	Return a value	2: Executing 4: Execution success 12: Execution failure This parameter is optional.

- **Example**

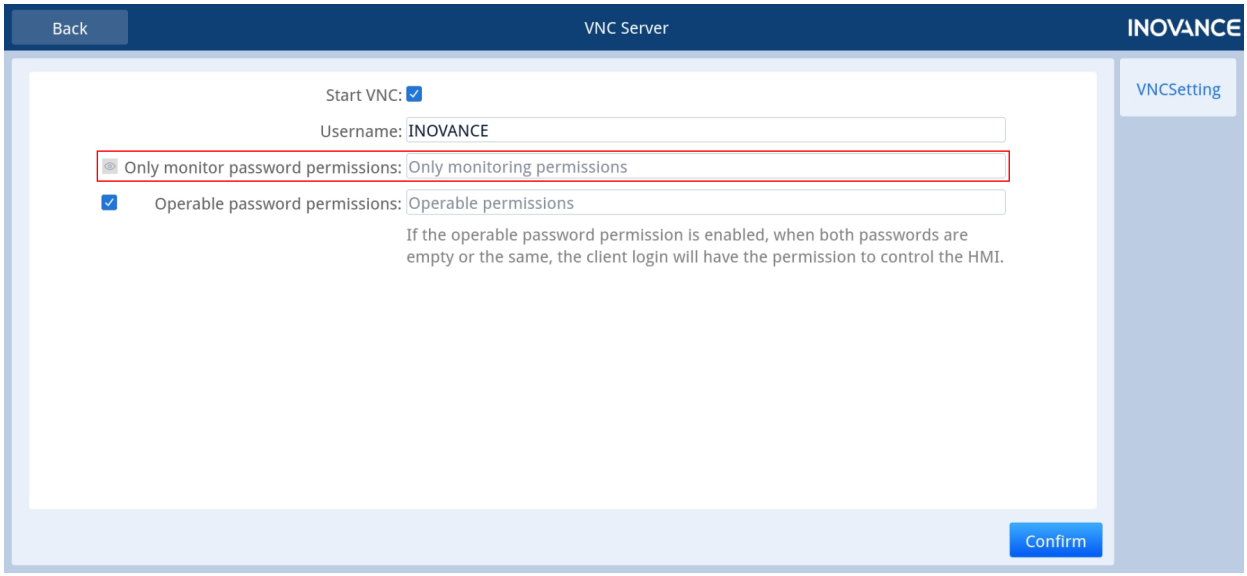
  Use in scripts:

```
function Script_2() {
//Specify the username for accessing the VNC server as User_01, and return the execution result to the variable Status.
setVNCName("User_01", smartTags('Status'));
}
```

19.12.3 Setting the VNC View-Only Password

The setVNCOnlyViewPassword function is used for setting the VNC view-only password. It has the same function as the View-Only Password option on the hardware touchscreen.

This password allows users to access the server in view-only mode, providing the permission to view screen without any interactive operations.



- **Function format**

  setVNCOnlyViewPassword(/*password*/, /*status*/)

- **Related parameters**

Parameter	Description	Value
Password	Optional field. Up to 32 characters, including letters, numbers, special characters, and punctuations.	
Status	Return a value	2: Executing 4: Execution success 12: Execution failure This parameter is optional.

- **Example**

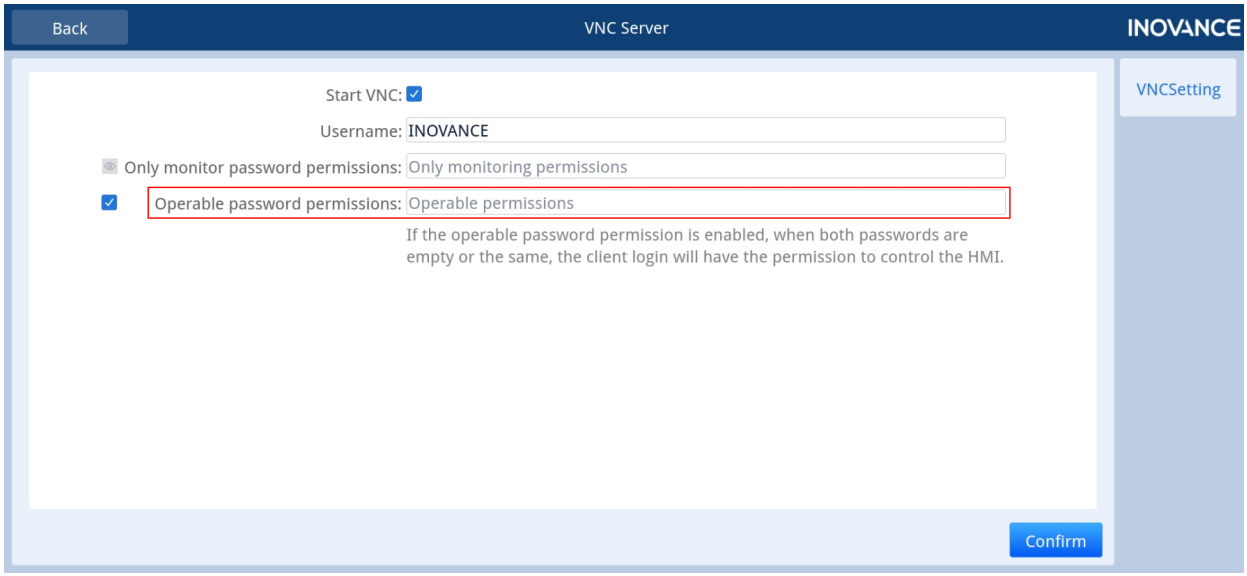
  Use in scripts:

```
function Script_3() {
//Set the username and password with view-only permission, and return the execution result of setVNCOnlyViewPassword to the variable Status_view.
setVNCName("User_01",smartTags('Status'));
setVNCOnlyViewPassword("Aa123456", smartTags('Status_view'))
}
```

19.12.4 Setting the VNC Control Password

The setVNCControlPassword function is used for setting the VNC control password. It has the same function as the Control Password option on the hardware touchscreen.

This function can be used to set a password. However, you can use the password to access the server and control/modify system configurations only when the control permission is enabled (through the setVNCControlPermission function).



- **Function format**

  setVNCControlPassword(/*password*/,/*status*/)

- **Related parameters**

Parameter	Description	Value
Password	Optional field. Up to 32 characters, including letters, numbers, special characters, and punctuations.	-
Status	Return a value	2: Executing 4: Execution success 12: Execution failure This parameter is optional.

- **Example**

  Use in scripts:

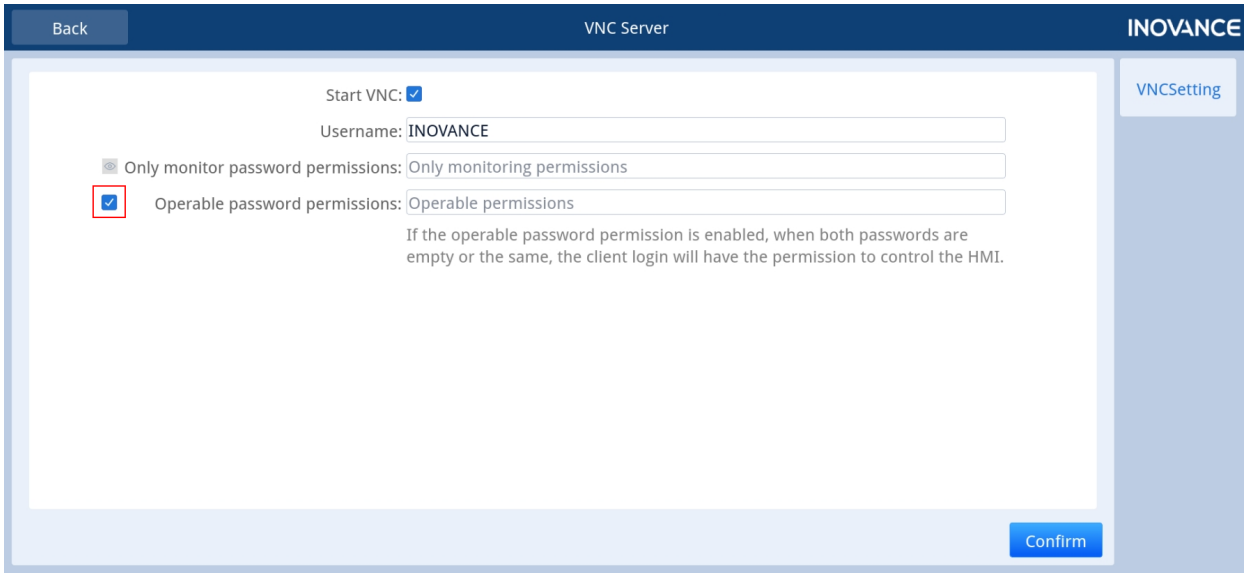
```
function Script_4() {
//Set the username and password with control permission, and return the execution result of setVNCControlPassword to the variable Status_control.
setVNCName("User_01",smartTags('Status'));
setVNCControlPassword("Aa123456", smartTags('Status_control'));
}
```

19.12.5 Setting the VNC Control Permission

The setVNCControlPermission function is used for enabling or disabling the VNC control permission. It has the same function as the Control Password option on the hardware touchscreen.

This function enables interactive operations, including controlling and modifying system configurations, after control password verification.

Only when this permission is enabled can users use the control password to gain full access rights.



- **Function format**

  setVNCControlPermission(/*enable flag*/, /*status*/)

- **Related parameters**

Parameter	Description	Value
Enable flag	Enable command	Constants: Enable (True), Disable (False) Variables: Bool type
Status	Return a value	2: Executing 4: Execution success 12: Execution failure This parameter is optional.

- **Example**

  Use in scripts:

```
//Enable the VNC remote control permission, and return the execution result to the variable Status.
setVNCControlPermission(true, smartTags('status')
```

19.13 Timer

Name	Function	Format	Example

Set periodic task	setIntervalTask	setintervalTask(/*callback function*/,/*execution interval*/)	function Script_1() {
			//Execute the increaseValue function periodically, where increaseValue is the function object executed by the periodic task, and the scheduled interval is 1000 (ms)
			//smartTags('Tag_1')--object of variable Tag_1; 1 means increment step; true means resetting after exceeding the limit; false means not recording the operation
			let id=setIntervalTask(increaseValue, 1000,smartTags('Tag_1'),1,true,false);
			//The return value id is the id of the periodic task and is used to stop the periodic task. Write the id into variable Tag_1
			setValue(smartTags('Tag_1'),id);
			//After the setValue is executed, the value of variable Tag_1 is the id
			}

Clear periodic task	clearIntervalTask	clearIntervalTask(/*timer flag*/)	function Script_1() {
			//Stop a periodic task
			//Get the id of the periodic task, smartTags('Tag_1')--object of variable Tag_1
			let id=getValue(smartTags('Tag_1');
			//Stop the periodic task based on the id clearIntervalTask(id);
			}

Name	Function	Format	Example

Set timeout task	setTimeoutTask	setTimeoutTask(/*callback func*/,/*scheduled time*/)	function Script_1() {
			//Delay the execution of the increaseValue function, where increaseValue is the function object executed by the delay task, and the delay interval is 5000 (ms)
			//smartTags('Tag_1')--object of variable Tag_1; 1 means increment step; true means resetting after exceeding the limit; false means not recording the operation
			let id=setTimeoutTask(increaseValue, 1000,smartTags('Tag_1'),1,true,false);
			//The return value id is the id of the delay task and is used to stop the delay task. Write the id into variable
			Tag_1setValue(smartTags('Tag_1'),id);
			//After the setValue is executed, the value of variable Tag_1 is the id
			}

Clear timeout task	clearTimeoutTask	clearTimeoutTask(/*timer flag*/)	function Script_1() {
			//Stop a delay task
			//Get the id of the delay task, smartTags('Tag_1')--object of variable Tag_1
			let id=getValue(smartTags('Tag_1');
			//Stop the delay task based on
			the id clearTimeoutTask(id);
			}

20 Hardware Device

20.1 Introduction to Hardware Guides

Diverse hardware devices can be configured in the iFA software. You can obtain the guides for a device in the following two ways to review parameter details and other information.

- iFA help system
  The guides for hardware devices supported by iFA have been integrated into the help system. After selecting a device in the device list of the iFA software, press “F1” to access the guide information page for the selected device in the help system.
- Inovance website
  Visit https://www.inovance.com/global, choose Service&Support > Support > Documentation Download.

> NOTE
>
> The guide links can be accessed with network connectivity.
>
> The guides for some devices are not yet available on the Inovance website. Pressing “F1” after selecting such a device redirects to the help system homepage. We will continuously update or complete the device guides.

20.2 PLC

20.2.1 EVO400

Product Series	Description	Link to the Guide
EVO410-A12P16	Introduces product information, installation, wiring, configuration and commissioning, fault diagnosis, and maintenance of the EVO400 series PLCs.	EVO400 Series Programmable Logic Controller User Guide
EVO410-A14P16		
EVO410-A15P16		
EVO410-A17P16		
EVO410-A12N16		
EVO410-A14N16		
EVO410-A15N16		
EVO410-A17N16		

20.2.2 EVO500

Product Series	Description	Link to the Guide
EVO510-A12P16	Introduces product information, installation, wiring, configuration and commissioning, fault diagnosis, and maintenance of the EVO510 series PLCs.	EVO510 Programmable Logic Controller User Guide
EVO510-A14P16		
EVO510-A15P16		
EVO521-0808TN(TP)	Introduces product information, mechanical installation, electrical installation, and communication connections of the EVO500 series PLCs.	EVO500 Series Programmable Logic Controller User Guide
EVO522-0808TN(TP)		
EVO523-0808TN(TP)		

20.2.3 EVO700

Product Series	Description	Link to the Guide
EVO763-0804TN	Introduces product information, installation, wiring, power connection and disconnection, configuration and commissioning, functions, fault diagnosis, maintenance, and servicing of the EVO700 series PLCs.	EVO700 Series Programmable Logic Controller User Guide
EVO764-0804TN		
EVO764-0804TN-Q		
EVO765-0804TN		
EVO766-0804TN		

20.2.4 EVO700C

Product Series	Description	Link to the Guide
EVO760C-C30	Introduces product information, installation, wiring, configuration and commissioning, fault diagnosis, and maintenance of the EVO760C series PLCs.	EVO760C Ultra-Compact Intelligent Controller User Guide
EVO760C-C40		
EVO780C-A60	Introduces product information, installation, wiring, configuration and commissioning, fault diagnosis, and maintenance of the EVO780C series PLCs.	EVO780C Series Intelligent Programmable Logic Controller User Guide
EVO780C-A70		
EVO780C-A80		

20.2.5 EVO800

Product Series	Description	Link to the Guide
EVO810	A collection of guides for the EVO800 series large-scale programmable logic controller redundant system, including system guide and equipment guide. • Equipment guide: Provides a brief overview of product properties, including model and nameplate, components, technical specifications, and terminal assignment. • System guide: Covers all typical use scenarios of the system, with detailed description of the system, installation, wiring, configuration and commissioning, and fault diagnosis.	EVO800 Series Large-Scale Programmable Logic Controller Redundant System Guide Collection
EVO820		

20.2.6 EVO1000

Product Series	Description	Link to the Guide
EVO1000EF	A collection of guides for the EVO1000EF series failsafe PLC, including system guide and equipment guide. • Equipment guide: Provides a brief overview of product properties, including model and nameplate, components, technical specifications, and terminal assignment. • System guide: Covers all typical use scenarios of the system, with detailed description of the system, installation, wiring, configuration and commissioning, and fault diagnosis.	EVO1000EF Series Failsafe PLC User Guide

20.2.7 APC

Product Series	Description	Link to the Guide
APC520-A10P16	Introduces product information, installation, wiring, configuration and commissioning, functions, and fault diagnosis of the APC520C series PLCs.	APC520 Series Advanced Process Controller User Guide
APC520-A10N16		
APC760C-A10	Introduces product information, installation, wiring, configuration and commissioning, functions, and fault diagnosis of the APC760C series PLCs.	APC760C Series Advanced Process Controller User Guide
APC760C-A20		
APC760C-A30		

20.3 HMI

20.3.1 IT9000

Product Series	Description	Link to the Guide
IT9156E	Introduces specifications, mounting dimensions, and terminal assignment of the IT9156E series product.	IT9000 Series Human Machine Interface User Guide

20.4 I/O Modules

20.4.1 GL10

20.4.1.1 I/O Coupler

Product Series	Description	Link to the Guide
GL10-RTU-ECTA	Introduces product information, mechanical design, electrical design, and software diagnostics of the GL10-RTU-ECTA module.	GL10-RTU-ECTA EtherCAT Communication Module User Guide

20.4.1.2 DI

Product Series	Description	Link to the Guide
GL10-1600END	Introduces specifications, features, and use methods of the GL10-1600END module.	GL10-1600END Digital Input Module User Guide
GL10-3200END	Introduces specifications, features, and use methods of the GL10-3200END module.	GL10-3200END Digital Input Module User Guide

20.4.1.3 DO

Product Series	Description	Link to the Guide
GL10-0016ETN	Introduces specifications, features, and use methods of the GL10-0016ETN module.	GL10-0016ETN Digital Output Module User Guide
GL10-0016ETP	Introduces specifications, features, and use methods of the GL10-0016ETP module.	GL10-0016ETP Digital Output Module User Guide

Product Series	Description	Link to the Guide
GL10-0016ER	Introduces specifications, features, and use methods of the GL10-0016ER module.	GL10-0016ER Digital Output Module User Guide
GL10-0032ETN	Introduces specifications, features, and use methods of the GL10-0032ETN module.	GL10-0032ETN Digital Output Module User Guide

20.4.1.4 AI

Product Series	Description	Link to the Guide
GL10-4AD	Introduces specifications, features, and use methods of the GL10-4AD module.	GL10-4AD Analog Input Module User Guide

20.4.1.5 AO

Product Series	Description	Link to the Guide
GL10-4DA	Introduces specifications, features, and use methods of the GL10-4DA module.	GL10-4DA Analog Output Module User Guide

20.4.1.6 Temperature

Product Series	Description	Link to the Guide
GL10-4PT	Introduces specifications, features, and use methods of the GL10-4PT module.	GL10-4PT Temperature Module User Guide
GL10-4TC	Introduces specifications, features, and use methods of the GL10-4TC /8TC module.	GL10-4TC/8TC Temperature Module User Guide
GL10-8TC		

20.4.2 GL20

20.4.2.1 I/O Coupler

Product Series	Description	Link to the Guide
GL20-RTU-EIP	Introduces product information, specifications, mechanical installation, electrical installation, program commissioning, and fault diagnosis of the GL20-RTU-EIP module.	GL20-RTU-EIP Series Communication Interface Module User Guide

Product Series	Description	Link to the Guide
GL20-RTU-ECT	Introduces product information, mechanical installation, electrical installation, program commissioning, and fault diagnosis of the GL20-RTU-ECT module.	GL20-RTU-ECT Communication Interface Module User Guide
GL20-RTU-ECT32	Introduces product information, mechanical installation, electrical installation, program commissioning, and fault diagnosis of the GL20-RTU-ECT32 module.	GL20-RTU-ECT32 Communication Interface Module User Guide

20.4.2.2 DI

Product Series	Description	Link to the Guide
GL20-0800END	Introduces mechanical installation, electrical installation, and programming examples of the GL20-0800END module.	GL20-0800END Digital Input Module User Guide
GL20-1600END	Introduces product information, mechanical installation, electrical installation, and program commissioning of the GL20-1600END module.	GL20-1600END Digital Input Module User Guide
GL20-1600END-5V	Introduces product information, terminal wiring, and technical specifications of the GL20-1600END-5V module.	GL20-1600END-5V Digital Input Module Equipment Guide
GL20-0800ENA	Introduces product information, mechanical installation, electrical installation, and program commissioning of the GL20-0800ENA module.	GL20-0800ENA Digital Input Module User Guide
GL20-3200END	Introduces product information, mechanical installation, electrical installation, and program commissioning of the GL20-3200END module.	GL20-3200END Digital Input Module User Guide

20.4.2.3 DO

Product Series	Description	Link to the Guide
GL20-0004ER	Introduces product information, mechanical installation, electrical installation, fault diagnosis, programming examples, and version compatibility information of the GL20-0004ER module.	GL20-0004ER Channel-Separated Digital Output Module User Guide
GL20-0004ETP-2A	Introduces product information, mechanical installation, electrical installation, program commissioning, troubleshooting, and version compatibility information of the GL20-0004ETP-2A module.	GL20-0004ETP-2A High Current Digital Output Module User Guide
GL20-0008ER	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and version compatibility information of the GL20-0008ER module.	GL20-0008ER Digital Output Module User Guide
GL20-0008ETN	Introduces product information, mechanical installation, electrical installation, and programming examples of the GL20-0008ETN module.	GL20-0008ETN Digital Output Module User Guide
GL20-0008ETP	Introduces product information, mechanical installation, electrical installation, and programming examples of the GL20-0008ETP module.	GL20-0008ETP Digital Output Module User Guide
GL20-0016ETN	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and version compatibility information of the GL20-0016ETN module.	GL20-0016ETN Digital Output Module User Guide

Product Series	Description	Link to the Guide
GL20-0016ETP	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and version compatibility information of the GL20-0016ETP module.	GL20-0016ETP Digital Output Module User Guide
GL20-0032ETN	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and version compatibility information of the GL20-0032ETN module.	GL20-0032ETN Digital Output Module User Guide
GL20-0032ETP	Introduces product information, mechanical installation, electrical installation, program commissioning, and fault diagnosis of the GL20-0032ETP module.	GL20-0032ETP Digital Output Module User Guide
GL20-0032ETP-M	Introduces product information, mechanical installation, electrical installation, program commissioning, and fault diagnosis of the GL20-0032ETP-M module.	GL20-0032ETP-M Digital Output Module User Guide
GL20-0016ETP-5V	Introduces product information, mechanical installation, electrical installation, program commissioning, and fault diagnosis of the GL20-0016ETP-5V module.	GL20-0016ETP-5V Digital Output Module User Guide

20.4.2.4 DI/DO

Product Series	Description	Link to the Guide
GL20-0404ETP-5V	Introduces product information, mechanical installation, electrical installation, fault diagnosis, programming examples, and version compatibility information of the GL20-0404ETP-5V module.	GL20-0404ETP-5V 4-Channel DI/DO Module User Guide

Product Series	Description	Link to the Guide
GL20-0808ETN	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and version compatibility information of the GL20-0808ETN module.	GL20-0808ETN Series Digital Input and Output Module User Guide
GL20-3232ETN	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and version compatibility information of the GL20-3232ETN module.	GL20-3232ETN-M Series Digital Input and Output Module User Guide

20.4.2.5 AI

Product Series	Description	Link to the Guide
GL20-4AD	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and version compatibility information of the GL20-4AD module.	GL20-4AD Analog Input Module User Guide
GL20-8ADI	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and version compatibility information of the GL20-8ADI and GL20-8ADV modules.	GL20-8ADI/8ADV Analog Input Module User Guide
GL20-8ADV		
GL20-4AD-H	Includes the product overview, model and nameplate, components, terminal wiring, as well as technical and environmental specifications of the GL20-4AD-H module.	GL20-4AD-H High-Precision Analog Input Module Equipment Guide

20.4.2.6 AO

Product Series	Description	Link to the Guide
GL20-4DA	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and version compatibility information of the GL20-4DA module.	GL20-4DA Analog Output Module User Guide
GL20-8DAI	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and version compatibility information of the GL20-8DAI and GL20-8DAV modules.	GL20-8DAI/8DAV Analog Output Module User Guide
GL20-8DAV		

20.4.2.7 Temperature

Product Series	Description	Link to the Guide
GL20-4PT	Introduces product information, mechanical installation, electrical installation, program commissioning, and fault diagnosis of the GL20-4PT module.	GL20-4PT Temperature Detection Module User Guide
GL20-4PT-ISO	Introduces product information, mechanical installation, electrical installation, program commissioning, and fault diagnosis of the GL20-4PT-ISO module.	GL20-4PT-ISO Series Isolated Thermal Resistor Temperature Detection Module User Guide
GL20-4TC	Introduces product information, mechanical installation, electrical installation, program commissioning, and fault diagnosis of the GL20-4TC module.	GL20-4TC Series Temperature Detection Module User Guide
GL20-4TC-ISO	Introduces product information, mechanical installation, electrical installation, program commissioning, and fault diagnosis of the GL20-4TC-ISO module.	GL20-4TC-ISO Series Isolated Thermocouple Temperature Detection Module User Guide

Product Series	Description	Link to the Guide
GL20-4LC-PID	Introduces product information, mechanical installation, electrical installation, PID temperature control, program commissioning, and fault diagnosis of the GL20-4LC-PID module.	GL20-4LC-PID Series Thermal Resistor/Thermocouple Temperature Control Module User Guide
GL20-8TC-PID	Introduces product information, mechanical installation, electrical installation, PID temperature control, program commissioning, and fault diagnosis of the GL20-8TC-PID module.	GL20-8TC-PID Series Thermocouple Input Temperature Detection Module User Guide
GL20-8TC	Introduces mechanical installation, electrical installation, program commissioning, fault diagnosis, and version compatibility information of the GL20-8TC module.	GL20-8TC Series Temperature Detection Module User Guide

20.4.2.8 Communication

Product Series	Description	Link to the Guide
GL20-2S485-MDB	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and version compatibility information of the GL20-2S485-MDB module.	GL20-2S485-MDB Series Communication Module User Guide
GL20-2SCOM-MDB	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and version compatibility information of the GL20-2SCOM-MDB module.	GL20-2SCOM-MDB Serial Communication Module User Guide
GL20-1DNM	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and version compatibility information of the GL20-1DNM module.	GL20-1DNM DeviceNet Master Module User Guide

Product Series	Description	Link to the Guide
GL20-2CAN	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and version compatibility information of the GL20-2CAN module.	GL20-2CAN Series Communication Module User Guide
GL20-2S485	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and version compatibility information of the GL20-2S485 module.	GL20-2S485 Series Communication Module User Guide
GL20-2SCOM	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and version compatibility information of the GL20-2SCOM module.	GL20-2SCOM Serial Communication Module User Guide

20.4.2.9 Power Supply

Product Series	Description	Link to the Guide
GL20-PS2	Introduces product information, mechanical installation, electrical installation, and programming examples of the GL20-PS2 module.	GL20-PS2 Series Relay Power Module User Guide

20.4.2.10 Encoder

Product Series	Description	Link to the Guide
GL20-2SSI	Introduces product information, mechanical installation, electrical installation, program commissioning, and fault diagnosis of the GL20-2SSI module.	GL20-2SSI 2-Channel Encoder Input Module User Guide

20.4.2.11 High-Speed Counting

Product Series	Description	Link to the Guide
GL20-2HC	Introduces product information, specifications, mechanical installation, electrical installation, functions, program commissioning, fault diagnosis, object dictionary, and address space of the GL20-2HC module.	GL20-2HC Two-Channel High-Speed Counter Module User Guide

20.4.3 GR10

20.4.3.1 DI/DO

Product Series	Description	Link to the Guide
GR10-0808ETNE	Introduces product information, mechanical design, electrical design, and software diagnostics of the GR10-0808ETNE module.	GR10-0808ETNE EtherCAT Slave I/O Module User Guide
GR10-1616ETNE	Introduces product information, mechanical design, electrical design, and software diagnostics of the GR10-1616ETNE module.	GR10-1616ETNE EtherCAT Slave I/O Module User Guide

20.4.3.2 AI

Product Series	Description	Link to the Guide
GR10-4ADE	Introduces product information, mechanical and electrical design, programming examples, and software diagnostics of the GR10-4ADE module.	GR10-4ADE Analog Input Module User Guide

20.4.3.3 AO

Product Series	Description	Link to the Guide
GR10-4DAE	Introduces product information, mechanical and electrical design, programming examples, and software diagnostics of the GR10-4DAE module.	GR10-4DAE Analog Output Module User Guide

20.4.3.4 Temperature

Product Series	Description	Link to the Guide
GR10-8TCE	Introduces product information, mechanical design, electrical design, programming examples, and software diagnostics of the GR10-8TCE module.	GR10-8TCE Temperature Detection Module User Guide

20.4.3.5 Pulse Input

Product Series	Description	Link to the Guide
GR10-2HCE	Introduces specifications, features, and use methods of the GR10-2HCE module.	GR10-2HCE EtherCAT Slave Counter Module User Guide
GR10-5HCE	Introduces specifications, features, and use methods of the GR10-5HCE module.	GR10-5HCE EtherCAT Slave Counter Module User Guide

20.4.3.6 Pulse Output

Product Series	Description	Link to the Guide
GR10-2PHE	Introduces product information, mechanical design, electrical design, and software diagnostics of the GR10-2PHE module.	GR10-2PHE High-Speed Differential Pulse Positioning Remote Module User Guide
GR10-4PME	Introduces product information, mechanical design, electrical design, and software diagnostics of the GR10-4PME module.	GR10-4PME EtherCAT Slave Positioning Module User Guide

20.4.3.7 EtherCAT Splitter

Product Series	Description	Link to the Guide
GR10-EC-3SW	Introduces product information, specifications, mechanical installation, and electrical installation of the GR10-EC-3SW module.	GR10-EC-3SW 3-Port EtherCAT Splitter Module User Guide
GR10-EC-6SW	Introduces product information, specifications, mechanical design, and electrical design of the GR10-EC-6SW module.	GR10-EC-6SW 6-Port EtherCAT Splitter Module User Guide

20.4.4 GR20

20.4.4.1 Digital Slave

Product Series	Description	Link to the Guide
GR20-16EMNL	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GR20-16EMNL module.	GR20-16EMNL Digital Input and Output Module User Guide
GR20-16EMPL	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GR20-16EMPL module.	GR20-16EMPL Digital Input and Output Module User Guide
GR20-16EMNL-BIT	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GR20-16EMNL module.	GR20-16EMNL Digital Input and Output Module User Guide
GR20-16EMNL-BYTE	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GR20-16EMNL module.	GR20-16EMNL Digital Input and Output Module User Guide
GR20-16EMPL-BIT	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GR20-16EMPL module.	GR20-16EMPL Digital Input and Output Module User Guide
GR20-16EMPL-BYTE	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GR20-16EMPL module.	GR20-16EMPL Digital Input and Output Module User Guide

20.4.4.2 Optical Fiber Converter

Product Series	Description	Link to the Guide
GR20-1OP	Introduces product information, mechanical installation, and electrical installation of the GR20-1OP-EN module.	GR20-1OP-EN Fiber Optic Converter Module User Guide

20.4.5 GR20T

20.4.5.1 DI

Product Series	Description	Link to the Guide
GR20T-ECT-1600END	Introduces product information, mechanical installation, electrical installation, program commissioning, and fault diagnosis of the GR20T-ECT-1600END/3200END module.	GR20T-ECT-1600END/3200END Digital Input Module User Guide
GR20T-ECT-3200END		

20.4.5.2 DO

Product Series	Description	Link to the Guide
GR20T-ECT-0016ETN	Introduces product information, mechanical installation, electrical installation, program commissioning, and fault diagnosis of the GR20T-ECT-0016ETN /0032ETN module.	GR20T-ECT-0016ETN/0032ETN Digital Output Module User Guide
GR20T-ECT-0032ETN		
GR20T-ECT-0016ETP	Introduces product information, mechanical installation, electrical installation, program commissioning, and fault diagnosis of the GR20T-ECT-0016ETP /0032ETP module.	GR20T-ECT-0016ETP/0032ETP Digital Output Module User Guide
GR20T-ECT-0032ETP		

20.4.5.3 DI/DO

Product Series	Description	Link to the Guide
GR20T-ECT-0808EMN	Introduces product information, mechanical installation, electrical installation, program commissioning, and fault diagnosis of the GR20T-ECT-0808EMN/1616EMN module.	GR20T-ECT-0808EMN/1616EMN Digital Input and Output Module User Guide
GR20T-ECT-1616EMN		
GR20T-ECT-1616EMN-E	Introduces product information, mechanical installation, electrical installation, program commissioning, and fault diagnosis of the GR20T-ECT-1616EMN-E module.	GR20T-ECT-1616EMN-E Digital Input and Output Module User Guide

20.4.6 GS20

20.4.6.1 IO-Link Master

Product Series	Description	Link to the Guide
GS20-ECT-8L	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GS20-ECT-8L high-protection IO-Link master module.	GS20-ECT-8L High Protection IO-Link Master Module User Guide
GS20-EIP-8L	Introduces product information, mechanical installation, electrical installation, program commissioning, WEB functionality, fault diagnosis, and version compatibility information of the GS20-EIP-8L module.	GS20-EIP-8L Series High Protection IO-Link Master Module User Guide

20.4.6.2 Digital Slave

Product Series	Description	Link to the Guide
GS20-16EMNL	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GS20-16EMNL module.	GS20-16EMNL Digital Input and Output Module User Guide
GS20-16EMPL	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GS20-16EMPL module.	GS20-16EMPL Digital Input and Output Module User Guide
GS20-16EMNL-S	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GS20-16EMNL-S module.	GS20-16EMNL-S Series Digital Input and Output Module User Guide

Product Series	Description	Link to the Guide
GS20-16EMPL-S	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GS20-16EMPL-S module.	GS20-16EMPL-S Series Digital Input and Output Module User Guide
GS20-IOL-16EMPHC	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GS20-IOL-16EMPHC/16EMNHC module.	GS20-IOL-16EMPHC/16EMNHC Series Digital Input and Output Module User Guide
GS20-IOL-16EMNHC		
GS20-IOL-16EMPTEC	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GS20-IOL-16EMPTEC/16EMNTEC module.	GS20-IOL-16EMPTEC_16EMNTEC Series Digital Input and Output Module User Guide
GS20-IOL-16EMNTEC		
GS20-IOL-16EMPT	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GS20-IOL-16EMPT/16EMNT module.	GS20-IOL-16EMPT/16EMNT Digital Input and Output Module User Guide
GS20-IOL-16EMNT		
GS20-16EMNL-BIT	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GS20-16EMNL module.	GS20-16EMNL Digital Input and Output Module User Guide
GS20-16EMNL-BYTE	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GS20-16EMNL module.	GS20-16EMNL Digital Input and Output Module User Guide
GS20-16EMNL-S-BYTE	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GS20-16EMNL-S module.	GS20-16EMNL-S Series Digital Input and Output Module User Guide

Product Series	Description	Link to the Guide
GS20-16EMNL-SC	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GS20-16EMNL-SC module.	GS20-16EMXL-SC Digital Input and Output Module User Guide
GS20-16EMPL-BIT	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GS20-16EMPL module.	GS20-16EMPL Digital Input and Output Module User Guide
GS20-16EMPL-BYTE	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GS20-16EMPL module.	GS20-16EMPL Digital Input and Output Module User Guide
GS20-16EMPL-S-BYTE	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GS20-16EMPL-S module.	GS20-16EMPL-S Series Digital Input and Output Module User Guide
GS20-IOL-16EMPHC-J	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GS20-IOL-16EMPHC/16EMNHC module.	GS20-IOL-16EMPHC/16EMNHC Series Digital Input and Output Module User Guide
GS20-IOL-16EMNHC-J		
GS20-IOL-16EMPTEC-J	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GS20-IOL-16EMPTEC/16EMNTEC module.	GS20-IOL-16EMPTEC_16EMNTEC Series Digital Input and Output Module User Guide
GS20-IOL-16EMNTEC-J		

20.4.6.3 Analog Slave

Product Series	Description	Link to the Guide
GS20-IOL-8ADH	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GS20-IOL-8ADH module.	GS20-IOL-8ADH Series Analog Input Module User Guide
GS20-IOL-12ENP2DAH	Introduces product information, mechanical installation, electrical installation, program commissioning, fault diagnosis, and object list of the GS20-IOL-12ENP2DAH module.	GS20-IOL-12ENP2DAH Series Digital and Analog Hybrid Module User Guide

20.4.7 GE20

20.4.7.1 Local Expansion Module

Product Series	Description	Link to the Guide
GE20-232/485	Introduces product information, mechanical installation, electrical installation, and programming examples of the GE20-232/485 communication expansion card.	GE20-232/485 Communication Expansion Card User Guide
GE20-232/485-RTC	Introduces product information, mechanical installation, electrical installation, and programming examples of the GE20-232/485-RTC communication expansion card.	GE20-232/485-RTC Communication Expansion Card User Guide
GE20-2AD1DA-I	Introduces product information, mechanical installation, electrical installation, and programming examples for the GE20-2AD1DA-I analog current input and output expansion card.	GE20-2AD1DA-I Analog Input and Output Current-Type Expansion Card User Guide
GE20-2AD1DA-V	Introduces product information, mechanical installation, electrical installation, and programming examples of the GE20-2AD1DA-V analog voltage input and output expansion card.	GE20-2AD1DA-V Analog Input and Output Voltage-Type Expansion Card User Guide

Product Series	Description	Link to the Guide
GE20-4DI	Introduces product information, mechanical installation, electrical installation, and programming examples of the GE20-4DI digital input expansion card.	GE20-4DI Digital Input Expansion Card User Guide
GE20-4DO-TN	Introduces product information, mechanical installation, electrical installation, and programming examples of the GE20-4DO-TN sink transistor output expansion card.	GE20-4DO-TN Sink-Type Transistor Output Expansion Card User Guide
GE20-RTC	Introduces product information, mechanical installation, electrical installation, and programming examples of the GE20-RTC clock expansion card.	GE20-RTC Clock Expansion Card User Guide
GE20-TF	Introduces product information, mechanical installation, and programming examples of the GE20-TF memory expansion card.	GE20-TF Memory Expansion Card User Guide
GE20-TF-RTC	Introduces product information, mechanical installation, and programming examples of the GE20-TF-RTC memory expansion card.	GE20-TF-RTC Memory Expansion Card (Integrated RTC) User Guide
GE20-CAN-485	Introduces product information, mechanical installation, electrical installation, and programming examples of the GE20-CAN-485 communication expansion card.	GE20-CAN-485 Communication Expansion Card User Guide

20.4.8 Hi50

20.4.8.1 I/O Coupler

Product Series	Description	Link to the Guide
Hi50-RTU-EIP-R00	A collection of equipment guides and system guides for Hi50 series I/O modules, covering all application scenarios of the system and including product information, system overview, EtherNet/IP, installation, wiring, configuration and commissioning, and fault diagnosis.	Hi50 Series Failsafe I/O System User Guide

20.4.8.2 DI

Product Series	Description	Link to the Guide
Hi50-0800END-F00	A collection of equipment guides and system guides for Hi50 series I/O modules, covering all application scenarios of the system and including product information, system overview, EtherNet/IP, installation, wiring, configuration and commissioning, and fault diagnosis.	Hi50 Series Failsafe I/O System User Guide

20.4.8.3 DO

Product Series	Description	Link to the Guide
Hi50-0004ETP-F2A	A collection of equipment guides and system guides for Hi50 series I/O modules, covering all application scenarios of the system and including product information, system overview, EtherNet/IP, installation, wiring, configuration and commissioning, and fault diagnosis.	Hi50 Series Failsafe I/O System User Guide
Hi50-0008ETP-F00		

20.4.8.4 AI

Product Series	Description	Link to the Guide
Hi50-4ADI-F00	A collection of equipment guides and system guides for Hi50 series I/O modules, covering all application scenarios of the system and including product information, system overview, EtherNet/IP, installation, wiring, configuration and commissioning, and fault diagnosis.	Hi50 Series Failsafe I/O System User Guide
Hi50-4ADV-F00		

20.4.8.5 Temperature

Product Series	Description	Link to the Guide
Hi50-4PT-F00	A collection of equipment guides and system guides for Hi50 series I/O modules, covering all application scenarios of the system and including product information, system overview, EtherNet/IP, installation, wiring, configuration and commissioning, and fault diagnosis.	Hi50 Series Failsafe I/O System User Guide
Hi50-4TC-F00		

20.4.8.6 Stopper Plate

Product Series	Description	Link to the Guide
Hi50-3ATT-000	A collection of equipment guides and system guides for Hi50 series I/O modules, covering all application scenarios of the system and including product information, system overview, EtherNet/IP, installation, wiring, configuration and commissioning, and fault diagnosis.	Hi50 Series Failsafe I/O System User Guide

20.4.9 Hi80

20.4.9.1 I/O Coupler

Product Series	Description	Link to the Guide
Hi80-RTU-EIP	A collection of equipment guides and system guides for Hi80 series I/O modules, covering all application scenarios of the system and including product information, system overview, EtherNet/IP, installation, wiring, configuration and commissioning, and fault diagnosis.	Hi80 Series High-Performance Distributed I/O System User Guide

20.4.9.2 DI

Product Series	Description	Link to the Guide
Hi80-1600ENP	A collection of equipment guides and system guides for Hi80 series I/O modules, covering all application scenarios of the system and including product information, system overview, EtherNet/IP, installation, wiring, configuration and commissioning, and fault diagnosis.	Hi80 Series High-Performance Distributed I/O System User Guide

20.4.9.3 DO

Product Series	Description	Link to the Guide
Hi80-0016ETP	A collection of equipment guides and system guides for Hi80 series I/O modules, covering all application scenarios of the system and including product information, system overview, EtherNet/IP, installation, wiring, configuration and commissioning, and fault diagnosis.	Hi80 Series High-Performance Distributed I/O System User Guide

20.4.9.4 AI

Product Series	Description	Link to the Guide
Hi80-8ADI	A collection of equipment guides and system guides for Hi80 series I/O modules, covering all application scenarios of the system and including product information, system overview, EtherNet/IP, installation, wiring, configuration and commissioning, and fault diagnosis.	Hi80 Series High-Performance Distributed I/O System User Guide
Hi80-8ADV		

20.4.9.5 AO

Product Series	Description	Link to the Guide
Hi80-8DAI	A collection of equipment guides and system guides for Hi80 series I/O modules, covering all application scenarios of the system and including product information, system overview, EtherNet/IP, installation, wiring, configuration and commissioning, and fault diagnosis.	Hi80 Series High-Performance Distributed IO System User Guide
Hi80-8DAV		

20.4.9.6 Temperature

Product Series	Description	Link to the Guide
Hi80-8LC	A collection of equipment guides and system guides for Hi80 series I/O modules, covering all application scenarios of the system and including product information, system overview, EtherNet/IP, installation, wiring, configuration and commissioning, and fault diagnosis.	Hi80 Series High-Performance Distributed I/O System User Guide

20.5 Wireless Module

Product Series	Description	Link to the Guide
AIR813-ECT-20S4	Introduces the installation, wiring, local management system operations, quick configuration examples, fault diagnosis, and maintenance of INO AIR wireless products.	Miniaturized Autonomous Wireless Product and Solution User Guide
AIR812-ECT-65S4		

20.6 Valve Terminal

Product Series	Description	Link to the Guide
IPT210-A-ECT	Introduces the installation of the EtherCAT fieldbus valve terminal series product, including installation procedure, mechanical installation, electrical installation, and commissioning instructions.	EtherCAT Fieldbus Valve Terminal User Guide
		IPT210 Series Fieldbus Valve Terminal Selection Guide
IPT300-A-ECT	Introduces the installation of the IPT300 distributed valve terminal series product, including installation procedure, mechanical installation, electrical installation, and commissioning instructions.	IPT300 Series Distributed Valve Terminal User Guide
IPT300s-A-ECT		IPT300 Series Distributed Valve Terminal Selection Guide

Product Series	Description	Link to the Guide
IPT210-A-EIP	Introduces the installation of the EtherNet/IP fieldbus valve terminal series product, including installation procedure, mechanical installation, electrical installation, and commissioning instructions.	EtherNet/IP Fieldbus Valve Terminal User Guide
		IPT210 Series Fieldbus Valve Terminal Selection Guide

20.7 Servo Drive

20.7.1 Single-Axis Servo Drive

Product Series	Description	Link to the Guide
IS650N/IS650	Introduces product information, mechanical installation, electrical installation of the IS650N series servo drive.	IS650N Series Servo Drive User Guide
SV600EN	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the SV600EN series servo drive.	SV600EN Series Servo Drive Guide Package
SV605N/SV605	Introduces selection, installation, commissioning, function, troubleshooting, and parameters of the SV605N series servo drive.	SV605N Series Servo Drive User Guide
SV630N/SV630/SV630NS	Introduces selection, installation, commissioning, function, troubleshooting, and parameters of the SV630N series servo drive.	SV630N Series Servo Drive User Guide
SV630C	Introduces selection, installation, commissioning, function, troubleshooting, and parameters of the SV630C series servo drive.	SV630C Series Servo Drive User Guide
SV660C	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the SV660C series servo drive.	SV660C Series Servo Drive User Guide
SV660N/SV660	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the SV660N series servo drive.	SV660N Series Servo Drive User Guide
SV670N/SV670	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the SV670N series servo drive.	SV670N Series Servo Drive User Guide

Product Series	Description	Link to the Guide
SV670-INT	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the SV670-INT series servo drive.	SV670-INT Series Servo Drive User Guide
SV670N-LK	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the SV670N-LK series servo drive.	SV670N-LK Series Servo Drive User Guide
SV680N/SV680	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the SV680N series servo drive.	SV680N Series Servo Drive User Guide
SV680-INT	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the SV680N-INT series servo drive.	SV680-INT Series Servo Drive User Guide
SV635N	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the SV635N series servo drives.	SV635N Series Servo Drive User Guide
SV640	-	-
IS810	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the IS810N series servo drives.	IS810N Series Standard Servo Drive (Multidrive System) User Guide
SV605C	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the SV605 series servo drives.	SV605P_C Series Servo Drive User Guide
SV630C	Introduces product information, installation, wiring, commissioning, and troubleshooting for common faults of the SV630C series servo drives.	SV630C Series Servo Drive User Guide
SV660C	Presents the functions and parameters of the SV660C series products, including function overview, basic servo functions, adjustment instructions, and parameter description.	SV660C Series Servo Drive User Guide
SV670C	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the SV670C series servo drives.	SV670C Series Servo Drive User Guide

Product Series	Description	Link to the Guide
SV680P	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the SV680P series servo drives.	SV680P Series Servo Drive User Guide

20.7.2 Multi-Axis Servo Drive

Product Series	Description	Link to the Guide
IS810N/IS810	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the IS810N series servo drives.	IS810N Series Standard Servo Drive (Multidrive System) User Guide
SV630ND	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the SV630ND series servo drive.	SV630ND Series Servo Drive Guide Package
SV660ND	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the SV660ND series servo drive.	SV660ND Series Servo Drive Guide Package
ES680	Introduces selection, installation, wiring, commissioning, functions, communication, troubleshooting, and routine maintenance of the ES680 series servo drives.	ES680 Series Servo Drive Advanced User Guide

20.7.3 Direct Drive Servo Drive

Product Series	Description	Link to the Guide
SV680LN	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the SV680LN series servo drive.	SV680LN Direct Drive Servo Drive Guide Package
SV670LN	Introduces model selection, installation, wiring, commissioning, functions, troubleshooting, and parameter descriptions.	SV670LN Series Direct Drive Servo Drive User Guide
TV670C	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the TV670C series servo drive.	-

20.7.4 Low-Voltage Servo Drive

Product Series	Description	Link to the Guide
SVD60N	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the SVD60N series servo drive.	SVD60N Series Servo Drive Guide Package

Product Series	Description	Link to the Guide
SVD60C	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the SVD60C series servo drive.	SVD60C Series Servo Drive User Guide

20.7.5 Stepper Servo Drive

Product Series	Description	Link to the Guide
STD60N	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the STD60N series servo drive.	STD60N Series Stepper System Manual Package

20.8 Low-Voltage AC Drive

20.8.1 General-Purpose Low-Voltage AC Drive

Product Series	Description	Link to the Guide
MD630N	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the MD630N series AC drive.	MD630 Series General-Purpose AC Drive User Guide
MD810	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the MD810 series AC drive.	MD810 Series AC Drive (Multidrive System) Function Guide
MD800	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the MD800 series AC drive.	MD800 Series AC Drive (Multidrive System) User Guide
MD520	Introduces product selection, mechanical design, electrical design, installation, communication, commissioning, function application, troubleshooting, and product certifications and standards.	MD520 Series General-Purpose AC Drive User Guide

20.8.2 Industrial AC Drive

Product Series	Description	Link to the Guide
MD580	Introduces selection, installation, wiring, commissioning, functions, troubleshooting, and parameters of the MD580 series hoist-specific AC drive.	MD580 Series Low-Voltage High-Performance Industrial AC Drive Guide Package

21 Appendix

21.1 Appendix 1: List of Fault Codes

Fault Code	Level	Description	Cause	Remedy
0x10060010	Fatal	IEC task runtime exceeding watchdog threshold or task exit timeout	The watchdog malfunctions.	Check the program execution time.
0x10060011	Fatal	Hardware watchdog error (hardware support required: not applicable to current product)	The hardware watchdog malfunctions.	Program timeout. Check the program execution time.
0x10060012	Fatal	I/O configuration error	An error occurs in I/O parameter configuration.	iFA software I/O parameter configuration error. Contact the manufacturer for support.
0x10060013	Fatal	Program verification error	An error occurs during program verification.	Attempt to re-download the program. If the fault persists, contact the manufacturer.
0x10060014	Fatal	Local bus access error	An error occurs while accessing the local bus.	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10060015	Fatal	I/O data update error	An error occurs during I/O data update.	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10060016	Fatal	Execution time exceeding cycle time	Task timeout.	Program timeout. Check the program execution time.
0x10060017	Fatal	Online program modification timeout	An error occurs in online modification.	Re-download fully.
0x10060018	Fatal	External library interface not implemented in RTS	The external library interface is improper.	Check the user-defined library type.
0x1006001B	Fatal	Loading startup project failed	An error occurs while loading the startup project.	Check whether the startup project is configured incorrectly.
0x1006001C	Fatal	Heap memory overflow	A heap memory overflow has occurred.	/

Fault Code	Level	Description	Cause	Remedy
0x1006001E	Fatal	Loading the startup project causes a crash.	An error occurs while loading the startup project.	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10060021	Fatal	The startup project does not match the current device.	The project startup is abnormal.	The project does not match the device model.
0x10060022	Fatal	Task scheduling error	An error occurred during task scheduling.	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10060023	Fatal	Project file verification error	An error occurs during project file verification.	/
0x10060024	Fatal	The current power-loss retention does not match the project.	The current power-loss retention does not match the project.	/
0x10060025	Fatal	IEC task not configured properly	The IEC task is not configured properly.	Contact the manufacturer for correct IDE version.
0x10060027	Critical	Power-off file verification error	An error occurs during power-off file verification.	Contact the manufacturer to resolve the power-off retention error.
0x10060028	Critical	Unable to load the program for online modification	A timeout occurred while acquiring the system time slice.	Attempt to re-download the program. If the fault persists, contact the manufacturer.
0x10060029	Critical	Downloading user project failed	The user project size exceeds the available system disk space.	Clear the PLC space and try re-downloading, or reduce the user project size.
0x1006003A	Critical	Disk offline	Disk offline	Generally caused by interference or hardware damage. Check the environmental conditions, or restart.
0x1006003B	Critical	Bad block on hard drive	Bad block on hard drive	It is recommended to replace the hard drive.
0x1006003C	Warning	Hard drive overheating	Hard drive overheating	It is recommended to check the ambient temperature.
0x1006003D	Warning	Hard drive life critical	Hard drive life critical	It is recommended to replace the hard drive.

Fault Code	Level	Description	Cause	Remedy
0x1006003E	Warning	Hard drive undercooling	Hard drive undercooling	It is recommended to check the ambient temperature.
0x10060050	Fatal	Illegal instruction	The instruction is illegal.	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10060051	Fatal	Access violation	An access violation occurs.	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10060052	Fatal	Privileged access	Privileged access	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10060053	Fatal	Page error	An error occurs on the page.	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10060054	Fatal	Stack overflow error	A stack overflow has occurred.	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10060055	Fatal	Invalid location	Invalid location	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10060056	Fatal	Invalid handle	Invalid handle	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10060057	Fatal	Page protection	Page protection	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10060059	Fatal	Invalid OP instruction operation	The OP instruction operation is invalid.	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x1006005A	Fatal	Unaligned memory access	The memory access is unaligned.	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.

Fault Code	Level	Description	Cause	Remedy
0x100600CC	Warning	CPU overheat	CPU overheat	It is recommended to check the ambient temperature, CPU usage or other factors.
0x100600CE	Warning	CPU undercooling	CPU undercooling	It is recommended to check the ambient temperature.
0x100600FF	Fatal	TCP thread fault	The TCP thread enters a deadlock state.	A deadlock has occurred in the TCP thread. Check and resolve the deadlock issue in the program, then restart the system and verify whether normal operation is restored. If the issue persists, contact the manufacturer.
0x10060100	Fatal	Data type misaligned	The data type is misaligned.	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10060101	Warning	An array index out-of-bounds error occurs in the project.	An array element in the project exceeds the defined upper or lower bound for the array.	Check arrays in the project where the elements are under a user-defined variable, and verify whether the computed values of the user-defined variable are out of the array range.
0x10060102	Warning	Division by zero in the project	The project contains a case where the divisor is zero.	Check division operations in the project for cases where the divisor is zero.
0x10060103	Fatal	Floating-point overflow	A floating-point overflow has occurred.	/
0x10060104	Fatal	Non-continuable error caused by missing operating system library interface.	A "Non-continuable" error occurred because the operating system library interface could not be found.	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10060105	Fatal	CPU monitoring thread watchdog error	A CPU monitoring thread watchdog error occurs.	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.

Fault Code	Level	Description	Cause	Remedy
0x10060106	Fatal	An abnormal pointer exists in the project.	A null pointer or invalid pointer exists in the project.	Check whether a null pointer or invalid pointer exists in the project.
0x10060150	Fatal	FPU: Error	FPU: Error	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10060151	Fatal	FPU: Floating-point operand exception	FPU: Floating-point operand exception	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10060152	Fatal	FPU: Floating-point division by zero	FPU: Floating-point division by zero	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10060153	Fatal	FPU: Inaccurate floating-point results	FPU: The floating-point operation result is inaccurate.	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10060154	Fatal	FPU: Invalid operation	FPU: Invalid operation	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10060155	Fatal	FPU: Floating-point overflow	FPU: Floating-point overflow	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10060157	Fatal	FPU: Floating-point underflow	FPU: Floating-point underflow	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10060158	Warning	Incorrect TYPE_CLASS parameter passed in external library function.	Incorrect TYPE_CLASS parameter passed in external library function.	Modify the TYPE_CLASS parameter passed to the external library function so that its data type aligns with that of the ANY type parameter.

Fault Code	Level	Description	Cause	Remedy
0x10060182	Warning	Server timeout	The connection channel between iFA and the controller is closed or fails, causing a server timeout.	Check the network cable connection status, or check whether the channel is unexpectedly closed.
0x10060184	Warning	Transmission verification error on the server channel between iFA and the controller.	A transmission verification error occurred on the connection channel between iFA and the controller.	Log out of all iFA sessions, and then log back in to the device.
0x10060185	Warning	A transmission error occurred on the server channel between iFA and the controller (receive mode).	The client between iFA and the controller attempts to send information, but the channel was in receive mode.	Log out of all iFA sessions, and then log back in to the device.
0x10060187	Warning	The number of available server channels for the connection between iFA and the controller has been exhausted.	There are no available channels for the connection from iFA to the controller.	Check whether the number of connection channels exceeds the limit. Close any unnecessary connection channels, and then reconnect.
0x10060189	Warning	The message length on the server connected from iFA to the controller exceeds the buffer capacity.	The message length between iFA and controller the buffer capacity.	Log out of all iFA sessions, and then log back in to the device.
0x1006018A	Information	The channel request on the server connection between iFA and the controller is invalid.	The channel request on the server connection between iFA and the controller is invalid.	No impact on normal use
0x1006018B	Information	The server channels for the connection between iFA and the controller is closed.	The server channels for the connection between iFA and the controller is closed.	The service is closed with prompt message.

Fault Code	Level	Description	Cause	Remedy
0x10060200	Fatal	The program hit a hardware breakpoint.	The program hit a hardware breakpoint.	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10061000	Fatal	Infinite loop type error	Infinite loop type error	Check the user program.
0x10061010	Fatal	IEC task runtime exceeding two task cycles	The IEC task runtime exceeds two task cycles.	Check the user program execution time.
0x10061020	Fatal	Local rack power failure	The power supply of the local rack is abnormal.	Check the cause of power failure.
0x10061021	Fatal	Backup rack power failure	The power supply of the backup rack is abnormal.	Check the cause of power failure.
0x10061022	Fatal	Identical redundant rack numbers	Two redundant racks have the same number.	Check the redundant rack numbers, set them to Rack A/B, and then restart the racks.
0x10061023	Fatal	Inconsistent redundant rack firmware versions	Two redundant racks have different firmware versions.	Check that the firmware versions of the redundant racks are consistent.
0x10061024	Fatal	Module removed	The module is faulty.	Check whether the module has been removed.
0x10061025	Fatal	PCIE communication error (redundant module)	The module is faulty.	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10061026	Fatal	Backplane communication SLOT0 channel abnormal	The module is faulty.	Check whether the module is seated and operating properly, and whether the CPU core 3 load is high; adjust the task cycle and task load properly to reduce the CPU core 3 load.
0x10061027	Critical	Both fiber optic links are disconnected.	The redundant communication links are faulty.	Check the fiber optic links of the synchronization module, and check the indicator lights. Reinsert the fiber optic cable to check if the issue is resolved. Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.

Fault Code	Level	Description	Cause	Remedy
0x10061028	Fatal	EtherNet/IP link fault	The communication link is faulty.	Check the status of the EtherNet/IP port cable. Try re-inserting the network cable to see if the issue is resolved. Restart the controller and check whether the fault is cleared. If not, contact the manufacturer.
0x10061029	Critical	Controller module CPU junction temperature exceeding the threshold	The device junction temperature is abnormal.	/
0x1006102A	Critical	Controller module FPGA junction temperature exceeding the threshold	The device junction temperature is abnormal.	/
0x1006102B	Critical	Synchronization module CPU junction temperature exceeding the threshold	The device junction temperature is abnormal.	/
0x1006102C	Critical	Synchronization module FPGA junction temperature exceeding the threshold	The device junction temperature is abnormal.	/
0x1006102D	Critical	Primary and backup data synchronization lasting for more than one cycle	The task cycle is set too short, and thus redundant data cannot be synchronized in time.	Increase the user task cycle and re-download the project.
0x1006102E	Critical	Start/Stop switch mismatch	The primary and backup start/stop switches are set at different positions.	Set the same position for primary and backup start/stop switches.
0x1006102F	Critical	Power-off retentive memory self-test failed	The power-off retentive memory self-test fails.	Contact the manufacturer for hardware replacement.

Fault Code	Level	Description	Cause	Remedy
0x10061030	Critical	Backplane communication SLOT1 channel abnormal	The communication between the module and backplane is abnormal.	Check whether the module is seated and operating properly, and whether the CPU core 3 load is high; adjust the task cycle and task load properly to reduce the CPU core 3 load.
0x10061031	Critical	Backplane communication SLOT2 channel abnormal	The communication between the module and backplane is abnormal.	Check whether the module is seated and operating properly, and whether the CPU core 3 load is high; adjust the task cycle and task load properly to reduce the CPU core 3 load.
0x10061032	Critical	Backplane communication SLOT3 channel abnormal	The communication between the module and backplane is abnormal.	Check whether the module is seated and operating properly, and whether the CPU core 3 load is high; adjust the task cycle and task load properly to reduce the CPU core 3 load.
0x10061033	Critical	Backplane communication SLOT4 channel abnormal	The communication between the module and backplane is abnormal.	Check whether the module is seated and operating properly, and whether the CPU core 3 load is high; adjust the task cycle and task load properly to reduce the CPU core 3 load.
0x10061034	Critical	Module removed from SLOT1	The module is removed.	Check whether the module is operating properly.
0x10061035	Critical	Module removed from SLOT2	The module is removed.	Check whether the module is operating properly.
0x10061036	Critical	Module removed from SLOT3	The module is removed.	Check whether the module is operating properly.
0x10061037	Critical	Module removed from SLOT4	The module is removed.	Check whether the module is operating properly.
0x10061038	Critical	IP address conflict detected on LAN1	Multiple devices in the network share the same IP address.	Check the IP addresses of devices in the network, and reconfigure a new IP address for the current device or other devices that share the same IP address as the current device.

Fault Code	Level	Description	Cause	Remedy
0x10061039	Critical	IP address conflict detected on LAN2	Multiple devices in the network share the same IP address.	Check the IP addresses of devices in the network, and reconfigure a new IP address for the current device or other devices that share the same IP address as the current device.
0x1006103A	Critical	IP address conflict detected on EIP3	Multiple devices in the network share the same IP address.	Check the IP addresses of devices in the network, and reconfigure a new IP address for the current device or other devices that share the same IP address as the current device.
0x1006103B	Fatal	Instantaneous voltage drop followed by recovery	The supply voltage is unstable with instantaneous drop.	Turn the device off and then back on.
0x1006103C	Fatal	Synchronizing board offline	The synchronizing board is faulty or not inserted.	Insert the synchronizing board as needed.
0x1006103D	Fatal	EIP PCIE link fault (main control module)	EMC interference	Turn the device off and then back on.
0x1006103E	Fatal	Controller module not seated properly	The controller module is not seated properly.	Power off the device, re-insert the module, and then power on the device.
0x1006103F	Critical	System temperature abnormal	The controller module operating temperature exceeds the threshold.	Check if the operating temperature exceeds the limit.
0x10061040	Critical	Slot number mismatch	The board slot numbers are mismatched.	Insert the proper board according to the structural diagram.
0x10061041	Critical	Backup PLC synchronizing board not inserted properly	The synchronizing board of the redundant backup PLC is not inserted properly.	Check the synchronizing board status. If abnormal, power off and re-insert the board.

Fault Code	Level	Description	Cause	Remedy
0x10061042	Critical	Backup PLC control board offline	The control board firmware of the backup PLC is running abnormally, or the control board is not inserted.	Check the fault code on the control board segment display, and take appropriate actions based on the fault code displayed.
0x10061043	Critical	Primary PLC synchronizing board not inserted properly	The redundant primary PLC synchronizing board is not inserted properly.	Check the synchronizing board status. If abnormal, power off and re-insert the board.
0x10061044	Critical	Dual optical fiber receive packets verification error	The optical fiber links are faulty.	Re-insert the optical fiber.
0x10061045	Critical	Abnormal	The temperature sensor failed to read the temperature.	Check if the module is normal.
0x10061046	Critical	The temperature readings from two CPUs differ excessively.	The temperature sensor or the ambient temperature is abnormal.	Check whether the ambient temperature is normal.
0x10061047	Fatal	Input parameter value out of range	1. The value of the input parameter for the type conversion instruction exceeds the instruction conversion range. 2. The K parameter of the MUX instruction is out of range.	1. Check whether the input parameter of the type conversion instruction is within the valid range. 2. Check whether the K parameter of the MUX instruction exceeds the number of comparison values.
0x10061048	Fatal	Power-off memory capacitor data abnormal	The power-off memory capacitor performance has degraded.	Contact the service provider for component replacement or maintenance.
0x10061049	Critical	IP address conflict detected on LAN1_A	Multiple devices in the network share the same IP address.	Check the IP addresses of devices in the network, and reconfigure a new IP address for the current device or other devices that share the same IP address as the current device.

Fault Code	Level	Description	Cause	Remedy
0x1006104A	Critical	IP address conflict detected on LAN1_B	Multiple devices in the network share the same IP address.	Check the IP addresses of devices in the network, and reconfigure a new IP address for the current device or other devices that share the same IP address as the current device.
0x1006104B	Warning	The system resource usage is nearing the upper limit.	The current process has reached the upper limit of file descriptor usage.	Contact the manufacturer.
0x1006104C	Information	The system resource usage has returned to normal.	The current process for the file descriptor usage has returned to normal.	/
0x1006104E	Critical	IP address conflict detected on LAN4	Multiple devices in the network share the same IP address.	Check the IP addresses of devices in the network, and reconfigure a new IP address for the current device or other devices that share the same IP address as the current device.
0x1006104F	Critical	IP address conflict detected on LAN3	Multiple devices in the network share the same IP address.	Check the IP addresses of devices in the network, and reconfigure a new IP address for the current device or other devices that share the same IP address as the current device.
0x1006104D	Critical	DHCP IP change	The DHCP IP address has been updated due to network changes, and the communication service is currently not functioning properly. Re-download the program to the PLC, or power cycle the PLC to restore normal operation.	Re-download the PLC, or perform a power cycle.

Fault Code	Level	Description	Cause	Remedy
0x10061050	Warning	Multiple default gateways configured on current PLC	1. Multiple default gateways have been configured on the PLC, which may affect cross-subnet communication. It is recommended to set a single default gateway. 2. For details on the configuration of multiple default gateways, you can view them in the "Network setting" interface under the "Online" menu.	1. In the iFA device tree interface, right-click the PLC device name. In the pop-up "Change device" interface, upgrade to the latest device version. 2. When performing a full download via iFA, select the "Set IP address in the project" mode, and disable all redundant default gateways, retaining a single default gateway only on the network ports that require cross-subnet communication.
0x10061051	Information	Current PLC gateway configuration restored to normal	/	/
0x10061052	Warning	Current project network configuration not effective	There may be an IP conflict.	1. Check the local network for conflicting IP addresses. 2. Perform the download again.
0x10062000	Warning	CPU average load too high	The average load of the CPU is too high.	Check the project configuration to inspect excessive load in the user logic or device configuration.
0x10062001	Warning	CPU2 high load alarm	The load of the CPU2 is too high.	Check the project configuration to inspect excessive load in the user logic or device configuration.
0x10062002	Warning	CPU3 load too high	The load of the CPU3 is too high.	Check the project configuration to inspect excessive load in the user logic or device configuration.
0x10062003	Warning	CPU0 load too high	The load of the CPU0 is too high.	Check the project configuration to inspect excessive load in the user logic or device configuration.

Fault Code	Level	Description	Cause	Remedy
0x10062004	Warning	CPU1 load too high	The load of the CPU1 is too high.	Check the project configuration to inspect excessive load in the user logic or device configuration.
0x10062005	Information	CPU average load too high fault cleared	The fault "CPU average load too high" has been cleared.	The fault "CPU average load too high" has been cleared.
0x10062006	Information	CPU2 load too high fault cleared	The fault "CPU2 load too high" has been cleared.	The fault "CPU2 load too high" has been cleared.
0x10062007	Information	CPU3 load too high fault cleared	The fault "CPU3 load too high" has been cleared.	-
0x10062008	Information	CPU0 load too high fault cleared	The fault "CPU0 load too high" has been cleared.	-
0x10062009	Information	CPU1 load too high fault cleared	The fault "CPU1 load too high" has been cleared.	-
0x1006200A	Information	CPU average load high	The average load of the CPU is high.	Check the project configuration to inspect excessive load in the user logic or device configuration.
0x1006200B	Information	CPU2 load high	The load of the CPU2 is high.	Check the project configuration to inspect excessive load in the user logic or device configuration.
0x1006200C	Information	CPU3 load high	The load of the CPU3 is high.	Check the project configuration to inspect excessive load in the user logic or device configuration.
0x1006200D	Information	CPU0 load high	The load of the CPU0 is high.	Check the project configuration to inspect excessive load in the user logic or device configuration.
0x1006200E	Information	CPU1 load high	The load of the CPU1 is high.	Check the project configuration to inspect excessive load in the user logic or device configuration.
0x1006200F	Information	CPU average load high cleared	The CPU average load high is cleared.	-

Fault Code	Level	Description	Cause	Remedy
0x10062010	Information	CPU2 load high cleared	The CPU2 load high is cleared.	-
0x10062011	Information	CPU3 load high cleared	The CPU3 load high is cleared.	-
0x10062012	Information	CPU0 load high cleared	The CPU0 load high is cleared.	-
0x10062013	Information	CPU1 load high cleared	The CPU1 load high is cleared.	-
0x10062014	Warning	Memory load too high	The memory load is too high.	Check the project configuration to inspect excessive load in the user logic or device configuration.
0x10062015	Information	Memory load too high fault cleared	The fault "memory load too high" has been cleared.	The fault "memory load too high" has been cleared.
0x10062016	Information	Memory load high	The memory load is high.	Check the project configuration to inspect excessive load in the user logic or device configuration.
0x10062017	Information	Memory load high cleared	The memory load high is cleared.	-
0x10062018	Warning	CPU temperature exceeds 115℃	The PLC operating ambient temperature exceeds the product specification requirements.	1. Check whether the PLC operating ambient temperature exceeds the specified product temperature limit. 2. Check whether the PLC cooling condition is normal and whether the cabinet cooling system is operating normally.
0x10062019	Critical	Failed to switch the network port to independent IP mode.	The current device does not support independent IP mode on the network ports.	Contact the manufacturer.
0x10062020	Warning	Manual primary-backup switchover	Manual primary-backup switchover is detected.	Check whether the primary-backup switchover is manually triggered.
0x10062021	Warning	Primary unit switched to backup unit	The primary unit is switched to the backup unit.	Identify the reason for switchover

Fault Code	Level	Description	Cause	Remedy
0x10062022	Warning	Optical fiber 1 link disconnected	The redundant optical fiber link 1 is faulty.	Check the connection of optical fiber link 1 of the synchronization module, and re-insert the optical fiber to see if the alarm is cleared.
0x10062023	Warning	Optical fiber 2 link disconnected	The redundant optical fiber link 2 is faulty.	Check the connection of optical fiber link 2 of the synchronization module, and re-insert the optical fiber to see if the alarm is cleared.
0x10062024	Warning	Optical fiber link switched	The redundant optical fiber link is faulty.	Check the connection of the optical fiber link of the synchronization module.
0x10062025	Warning	SD card inserted	The SD card is recovered.	Check the SD card.
0x10062026	Warning	SD card removed	The SD card is not detected.	Check the SD card.
0x10062027	Warning	SD card full	SD card warning	Manage the SD card contents to free up space.
0x10062028	Warning	Local rack project running abnormally (project not loaded successfully)	The local rack project is faulty.	Check the local rack project.
0x10062029	Warning	Local rack running status abnormal (project running abnormally)	The local rack is running abnormally.	Identify the cause and take appropriate actions.
0x1006202A	Warning	RTC battery voltage low	The RTC battery has a low voltage or is not inserted.	Replace the RTC battery.
0x1006202B	Warning	Primary-backup switchover timeout	The optical fiber link is faulty, or the data exchanged over the link is too large.	Check the switchover log, data traffic size, and optical fiber link status to comprehensively analyze the cause of timeout.
0x1006202C	Warning	Controller module ambient temperature exceeding the threshold	The ambient temperature is high.	Cool down the device operating environment.
0x1006202D	Warning	The start/stop DIP switch states on the two racks are inconsistent.	The start/stop DIP switch states on the two racks are inconsistent.	Adjust the start/stop DIP switch states on both racks so that they match.

Fault Code	Level	Description	Cause	Remedy
0x1006202E	Warning	A-side power supply module fault	The A-side power supply is not connected, overvoltage, or undervoltage.	Check the A-side power supply.
0x1006202F	Warning	B-side power supply module fault	The B-side power supply is not connected, overvoltage, or undervoltage.	Check the B-side power supply.
0x10062030	Warning	Modbus TCP configuration error	The Modbus parameters are not configured properly.	Recheck the Modbus configuration parameters, modify the improper ones, and then re-download the project.
0x10062031	Warning	SD card read-only	The inserted SD card is set to read-only.	Modify the SD card attribute to read-write and then re-inserting the card.
0x10062032	Warning	SLOT0 backplane communication data verification error	The configured backplane communication parameters are inconsistent.	After modifying the backplane communication settings to ensure consistency, re-download the project.
0x10062033	Warning	SLOT1 backplane communication data verification error	The configured backplane communication parameters are inconsistent.	After modifying the backplane communication settings to ensure consistency, re-download the project.
0x10062034	Warning	SLOT2 backplane communication data verification error	The configured backplane communication parameters are inconsistent.	After modifying the backplane communication settings to ensure consistency, re-download the project.
0x10062035	Warning	SLOT3 backplane communication data verification error	The configured backplane communication parameters are inconsistent.	After modifying the backplane communication settings to ensure consistency, re-download the project.
0x10062036	Warning	SLOT4 backplane communication data verification error	The configured backplane communication parameters are inconsistent.	After modifying the backplane communication settings to ensure consistency, re-download the project.

Fault Code	Level	Description	Cause	Remedy
0x10062037	Warning	Wi-Fi off	The Wi-Fi function is turned off.	/
0x10062038	Warning	No controller module program operation permissions	The permissions are insufficient.	Check the user permissions.
0x10062039	Warning	Project file consistency check failed during download	The project file consistency check fails.	Check the project or re-download the full project.
0x1006203A	Warning	Device configuration file MD5 mismatch	The MD5 value for the device configuration file is mismatched.	Re-compile the project and download the full project.
0x1006203B	Warning	Device configuration file MD5 verification error	The MD5 verification for the device configuration file fails.	Re-compile the project and download the full project.
0x1006203C	Warning	Loading device configuration file failed	The device configuration file is not loaded successfully.	Check the log for details and take appropriate actions.
0x1006203D	Warning	Parsing device configuration file failed	The device configuration file is not parsed successfully.	Check the log for details and take appropriate actions, and then re-compile and download the project. If the problem persists, contact technical support engineers.
0x1006203E	Warning	The master redundancy project has become non-redundant - redundancy exited.	/	/
0x1006203F	Warning	Primary and backup unit IP configuration mismatch	The IP addresses of the two units for a certain port is mismatched.	Check the network settings to identify the port, and then modify the IP addresses to keep them consistent.
0x10062040	Warning	Primary-backup synchronization data verification error	Data synchronization is delayed due to optical fiber module fault or too short IEC task cycle.	Check whether the optical fiber link is faulty or whether the IEC task cycle is set too short.

Fault Code	Level	Description	Cause	Remedy
0x10062041	Warning	SD card working directory not found	The SD card working directory has hung up unexpectedly.	Restart the controller system to automatically create the SD card mount directory.
0x10062042	Warning	Optical fiber 2 receive packet verification error	The optical fiber 1 link is faulty.	Re-insert the optical fiber.
0x10062043	Warning	Optical fiber 1 receive packet verification error	The optical fiber 2 link is faulty.	Re-insert the optical fiber.
0x10062044	Warning	Abnormal reading of backup capacitor parameters	The capacitor capacity decreases, and the internal resistance increases.	1. In single mode: See the maintenance section of the product manual for handling procedures. 2. In redundant mode: Manually switch between primary and backup racks.
0x10062045	Warning	Abnormal reading of backup capacitor parameters	Line anomaly; internal communication link failure	Restart the controller and check whether the fault is cleared. If not, contact the manufacturer for hardware inspection.
0x10062046	Warning	SD card access failure	SD card type mismatch	Check the SD card file system type.
0x10062047	Warning	SD card power supply fault	1. The SD card itself has an issue. 2. An SD card slot issue exists.	1. Verify with a new SD card. If the SD card works properly on a PC but not on the controller, contact after-sales support. 2. Alternatively, verify this SD card on a PC or another controller. If it works normally, contact after-sales support. If it does not, replace the SD card.
0x10062048	Warning	No redundancy allowed due to breakpoint set for primary rack	The breakpoint is set in the user program of the primary rack.	After the execution of "Delete breakpoint" under "Commissioning > Delete Breakpoint" menu in the iFA software, the primary and backup racks can normally achieve the redundancy.
0x10062049	Warning	CPU4 high load alarm	The load of the CPU4 is too high.	Check the project configuration to inspect excessive load in the user logic or device configuration.

Fault Code	Level	Description	Cause	Remedy
0x1006204A	Warning	CPU5 high load alarm	The load of the CPU5 is too high.	Check the project configuration to inspect excessive load in the user logic or device configuration.
0x1006204B	Information	CPU4 load too high fault cleared	The fault "CPU4 load too high" has been cleared.	The fault "CPU4 load too high" has been cleared.
0x1006204C	Information	CPU5 load too high fault cleared	The fault "CPU5 load too high" has been cleared.	The fault "CPU5 load too high" has been cleared.
0x1006204D	Warning	The dual rack modules do not match, so redundancy is lost.	The positions, version information, or IP addresses of the dual rack modules do not match.	Check if the positions, version information, or IP addresses of the dual rack modules match.
0x1006204E	Fatal	EN module not seated properly	EN module is not seated correctly.	Power off the device, re-insert the module, and then power on the device.
0x1006204F	Fatal	EN module SDIO RT interface clock frequency deviation	The clock frequency deviation has occurred on the SDIO RT interface of the EN module.	Contact the manufacturer.
0x10062050	Fatal	EN module SDIO NRT interface clock frequency deviation	The clock frequency deviation has occurred on the SDIO NRT interface of the EN module.	Contact the manufacturer.
0x10062051	Warning	CPU6 high load alarm	The load of the CPU6 is too high.	Check the project configuration to inspect excessive load in the user logic or device configuration.
0x10062052	Warning	CPU7 high load alarm	The load of the CPU7 is too high.	Check the project configuration to inspect excessive load in the user logic or device configuration.
0x10062053	Warning	CPU8 high load alarm	The load of the CPU8 is too high.	Check the project configuration to inspect excessive load in the user logic or device configuration.

Fault Code	Level	Description	Cause	Remedy
0x10062054	Warning	CPU9 high load alarm	The load of the CPU9 is too high.	Check the project configuration to inspect excessive load in the user logic or device configuration.
0x10062055	Warning	CPU10 high load alarm	The load of the CPU10 is too high.	Check the project configuration to inspect excessive load in the user logic or device configuration.
0x10062056	Warning	CPU11 high load alarm	The load of the CPU11 is too high.	Check the project configuration to inspect excessive load in the user logic or device configuration.
0x10062057	Warning	CPU12 high load alarm	The load of the CPU12 is too high.	Check the project configuration to inspect excessive load in the user logic or device configuration.
0x10062058	Warning	CPU13 high load alarm	The load of the CPU13 is too high.	Check the project configuration to inspect excessive load in the user logic or device configuration.
0x10062059	Warning	CPU14 high load alarm	The load of the CPU14 is too high.	Check the project configuration to inspect excessive load in the user logic or device configuration.
0x1006205A	Warning	CPU15 high load alarm	The load of the CPU15 is too high.	Check the project configuration to inspect excessive load in the user logic or device configuration.
0x1006205B	Warning	CPU16 high load alarm	The load of the CPU16 is too high.	Check the project configuration to inspect excessive load in the user logic or device configuration.
0x1006205C	Information	CPU6 load too high fault cleared	The fault "CPU6 load too high" has been cleared.	The fault "CPU6 load too high" has been cleared.
0x1006205D	Information	CPU7 load too high fault cleared	The fault "CPU7 load too high" has been cleared.	The fault "CPU7 load too high" has been cleared.
0x1006205E	Information	CPU8 load too high fault cleared	The fault "CPU8 load too high" has been cleared.	The fault "CPU8 load too high" has been cleared.

Fault Code	Level	Description	Cause	Remedy
0x1006205F	Information	CPU9 load too high fault cleared	The fault "CPU9 load too high" has been cleared.	The fault "CPU9 load too high" has been cleared.
0x10062060	Information	CPU10 load too high fault cleared	The fault "CPU10 load too high" has been cleared.	The fault "CPU10 load too high" has been cleared.
0x10062061	Information	CPU11 load too high fault cleared	The fault "CPU11 load too high" has been cleared.	The fault "CPU11 load too high" has been cleared.
0x10062062	Information	CPU12 load too high fault cleared	The fault "CPU12 load too high" has been cleared.	The fault "CPU12 load too high" has been cleared.
0x10062063	Information	CPU13 load too high fault cleared	The fault "CPU13 load too high" has been cleared.	The fault "CPU13 load too high" has been cleared.
0x10062064	Information	CPU14 load too high fault cleared	The fault "CPU14 load too high" has been cleared.	The fault "CPU14 load too high" has been cleared.
0x10062065	Information	CPU15 load too high fault cleared	The fault "CPU15 load too high" has been cleared.	The fault "CPU15 load too high" has been cleared.
0x10062066	Information	CPU16 load too high fault cleared	The fault "CPU16 load too high" has been cleared.	The fault "CPU16 load too high" has been cleared.
0x10063020	Information	Backup unit switched to primary unit	The backup unit is switched to the primary unit.	/
0x10063021	Information	Setting local rack as primary	The local rack is set as the primary rack.	/
0x10063022	Information	Setting local rack as backup	The local rack is set as the backup rack.	/
0x10063023	Information	Project status self-test passed	The project status self-test passed.	/
0x10063024	Information	Operation status self-test passed	The operation status self-test passed.	/
0x10063025	Information	Normal redundant rack status frame exchange	The redundant communication messages are normal.	/

Fault Code	Level	Description	Cause	Remedy
0x10063026	Information	Backup rack operational detected	The local rack detected that the backup rack is operational.	/
0x10063027	Information	Synchronizing project files between redundant racks	The project is synchronized between two redundant racks.	/
0x10063028	Information	Rack module inserted	The module is inserted.	/
0x10063029	Information	SLOT0 backplane communication restored	A message appears indicating that backplane communication has been restored.	/
0x1006302A	Information	EtherNet/IP link returned to normal.	The communication link has returned to normal.	/
0x1006302B	Information	Optical fiber link 1 returned to normal.	The redundant optical fiber link 1 has returned to normal.	/
0x1006302C	Information	Optical fiber link 2 returned to normal.	The redundant optical fiber link 2 has returned to normal.	/
0x1006302D	Information	PCIE link returned to normal	The PCIE link has returned to normal.	/
0x1006302E	Information	The start/stop switch mismatch is cleared.	/	/
0x1006302F	Information	Controller module operating temperature returned to normal.	The operating temperature has returned to normal.	/
0x10063030	Information	Controller module CPU junction temperature returned to normal.	The device junction temperature has returned to normal.	/

Fault Code	Level	Description	Cause	Remedy
0x10063031	Information	Controller module FPGA junction temperature returned to normal.	The device junction temperature has returned to normal.	/
0x10063032	Information	Synchronization module CPU junction temperature returned to normal	The device junction temperature has returned to normal.	/
0x10063033	Information	Synchronization module FPGA junction temperature returned to normal	The device junction temperature has returned to normal.	/
0x10063034	Information	SLOT0 backplane communication restored	The backplane communication is restored.	/
0x10063035	Information	SLOT1 backplane communication restored	The backplane communication is restored.	/
0x10063036	Information	SLOT2 backplane communication restored	The backplane communication is restored.	/
0x10063037	Information	SLOT3 backplane communication restored	The backplane communication is restored.	/
0x10063038	Information	SLOT4 backplane communication restored	The backplane communication is restored.	/
0x10063039	Information	Local rack power supply (A/B) restored	The power supply of the local rack is restored.	/
0x1006303A	Information	MRAM self-test passed	The MRAM self-test passed.	/
0x1006303B	Information	Module inserted into SLOT1	The module is inserted.	/
0x1006303C	Information	Module inserted into SLOT2	The module is inserted.	/
0x1006303D	Information	Module inserted into SLOT3	The module is inserted.	/

Fault Code	Level	Description	Cause	Remedy
0x1006303E	Information	Module inserted into SLOT4	The module is inserted.	/
0x1006303F	Information	LAN1 IP address conflict resolved	The IP address conflict is resolved.	/
0x10063040	Information	LAN2 IP address conflict resolved	The IP address conflict is resolved.	/
0x10063041	Information	EIP3 IP address conflict resolved	The IP address conflict is resolved.	/
0x10063042	Information	RTC battery voltage returned to normal	The RTC battery voltage has returned to normal.	/
0x10063043	Information	Synchronizing board restored to online	The synchronizing board is restored to online.	/
0x10063044	Information	Primary and backup unit IP configurations matched	The IP configurations for the primary and backup units are matched.	/
0x10063045	Information	Controller module seated properly	The controller module is seated properly.	/
0x10063046	Information	Redundant backup PLC synchronizing board inserted properly	The synchronizing board for the redundant backup PLC is inserted properly.	/
0x10063047	Information	Redundant backup PLC control board detected online	The control board for the redundant backup PLC is detected online.	/
0x10063048	Information	Primary PLC synchronizing board inserted properly	The synchronizing board for the primary PLC is inserted properly.	/
0x10063049	Information	Synchronization data verification error counter reset	The counter for synchronization data verification errors is reset.	/

Fault Code	Level	Description	Cause	Remedy
0x1006304A	Information	Optical fiber 2 receive packet verification fault cleared	The optical fiber 1 receive packet is restored to normal.	/
0x1006304B	Information	Optical fiber 1 receive packet verification fault cleared	The optical fiber 1 receive packet is restored to normal.	/
0x1006304C	Information	LAN1_A IP address conflict resolved	The IP address conflict is resolved.	/
0x1006304D	Information	LAN1_B IP address conflict resolved	The IP address conflict is resolved.	/
0x1006304E	Information	Primary rack breakpoint cleared	The primary rack breakpoint information is cleared.	/
0x1006304F	Information	The primary and backup rack numbers are different, which meets the redundancy condition.	The issue where the primary and backup rack numbers were identical has been resolved.	/
0x10063050	Information	The software versions on the primary and backup racks have been verified to be consistent.	The software version inconsistency between the primary and backup racks has been resolved.	/
0x10063051	Information	Fault cleared after primary/backup data synchronization exceeded one cycle.	Issue with primary and backup data synchronization exceeding the cycle limit has been resolved.	/
0x10063052	Information	EN module seated properly	The EN module is seated properly.	/

Fault Code	Level	Description	Cause	Remedy
0x10063053	Information	EN module SDIO RT interface clock frequency deviation returned to normal	The clock frequency deviation on the EN module SDIO RT interface has returned to normal.	/
0x10063054	Information	EN module SDIO NRT interface clock frequency deviation returned to normal	The clock frequency deviation on the EN module SDIO NRT interface has returned to normal.	/
0x10063055	Information	LAN3 IP address conflict resolved	The IP address conflict is resolved.	/
0x10063056	Information	LAN4 IP address conflict resolved	The IP address conflict is resolved.	/
0x10064010	Information	Time synchronization disabled	The time synchronization is disabled.	-
0x10064011	Information	Time synchronization is functioning normally, and the time source is the server.	Time synchronization is functioning normally, and the time source is the server.	-
0x10064012	Warning	Time synchronization is functioning normally, and the time source is the partner device.	Time synchronization is functioning normally, and the time source is the partner device.	-
0x10064013	Warning	Time synchronization is abnormal; the time source is the local free-running clock.	Time synchronization is abnormal; the time source is the local free-running clock.	The time synchronization server is unavailable, or the communication link is faulty. The current time source is the local free-running clock.

Fault Code	Level	Description	Cause	Remedy
0x10064020	Critical	The OPC UA service is not activated, because there are insufficient CPU resources on the device.	The OPC UA service is not activated, because there are insufficient CPU resources on the device.	Check that the available device CPU resources are not below the threshold for activating the OPC UA service.
0x10064021	Critical	The OPC UA service is not activated because there is insufficient memory on the device.	The OPC UA service is not activated because there is insufficient memory on the device.	Check that available device memory resources are not below the threshold for activating the OPC UA service.
0x10064030	Fatal	Network loop detected on port LAN1.	The device connections in the network form a closed loop.	Check the device connection in the network to avoid forming a closed loop.
0x10064031	Fatal	Network loop detected on port LAN2.	The device connections in the network form a closed loop.	Check the device connection in the network to avoid forming a closed loop.
0x10064032	Fatal	Network loop detected on port LAN1_A.	The device connections in the network form a closed loop.	Check the device connection in the network to avoid forming a closed loop.
0x10064033	Fatal	Network loop detected on port LAN1_B.	The device connections in the network form a closed loop.	Check the device connection in the network to avoid forming a closed loop.
0x10064034	Fatal	Network loop detected on port EIP3.	The device connections in the network form a closed loop.	Check the device connection in the network to avoid forming a closed loop.
0x10064035	Fatal	Network loop detected on port LAN3.	The device connections in the network form a closed loop.	Check the device connection in the network to avoid forming a closed loop.
0x10064036	Fatal	Network loop detected on port LAN4.	The device connections in the network form a closed loop.	Check the device connection in the network to avoid forming a closed loop.
0x10064040	Information	Network loop on port LAN1 cleared	The network loop has been cleared.	/

Fault Code	Level	Description	Cause	Remedy
0x10064041	Information	Network loop on port LAN2 cleared	The network loop has been cleared.	/
0x10064042	Information	Network loop on port LAN1_A cleared	The network loop has been cleared.	/
0x10064043	Information	Network loop on port LAN1_B cleared	The network loop has been cleared.	/
0x10064044	Information	Network loop on port EIP3 cleared	The network loop has been cleared.	/
0x10064045	Information	Network loop on port LAN3 cleared	The network loop has been cleared.	/
0x10064046	Information	Network loop on port LAN4 cleared	The network loop has been cleared.	/
0x10065000	Information	Normal connection with no fault	/	/
0x10065001	Fatal	Out of memory	Insufficient memory	Check whether any memory is available.
0x10065002	Critical	Socket failed to open	The permissions or system resources are insufficient, or the socket number is occupied.	1. Check the network connection. 2. Check the log for specific error information.
0x10065003	Critical	Unspecified IP	No IP address is specified when the socket is bound.	Check whether the IP is configured properly. (Correct: it must be local device IP)
0x10065004	Critical	Socket failed to be bound	Insufficient permission, insufficient system resources, and IP address or port occupied.	1. Check the log for specific error information. 2. Check whether the IP address is correct. 3. Check whether the port number is occupied.
0x10065006	Critical	hostname too long	The length of the hostname exceeds 512 bytes.	Check whether the length of the server URL identifier exceeds 512 bytes.

Fault Code	Level	Description	Cause	Remedy
0x10065008	Critical	Invalid EndpointUrl	The server endpoint URL is invalid.	Check whether the server URL identifier is proper. If the device name is used, check that the identifier is consistent with the device name. If the IP address is used, check that the IP address is configured properly.
0x10065009	Warning	Empty security policy	The security policy is not configured properly.	Check whether no option is selected on the security configuration page.
0x1006500A	Warning	Failed to set security policy	The security policy is not configured properly.	Check whether the options are configured properly on the security configuration page.
0x1006500B	Warning	Mismatch between security mode and security protocol	When the security mode is not None, the security protocol is set to None, which is not allowed.	Check whether the options are configured properly on the security configuration page and take effect.
0x1006500D	Warning	OPC UA network anomaly	The OPCUA network connection is disconnected, or the remote network is unreachable.	1. Re-insert the network cable and check whether the network port link indicator is flashing properly. 2. Check the switch or router configuration to confirm the network reachability. 3. Contact technical support engineers to log in to the device backend and check whether the specified network port is functioning properly by using"ipconfig -a, ethtool" commands.
0x1006500E	Warning	Port settings out of limit	Among the issued parameters, the port settings exceed the limits (which have been reset to the default values).	Check whether the port settings in IDE are improper or exceed the limits by choosing "Property > OPCUA > Server > Options" to view the port settings.

Fault Code	Level	Description	Cause	Remedy
0x1006500F	Warning	Maximum session timeout out of limit	Among the issued parameters, the maximum session timeout interval exceeds the limit (which has been reset to the default value).	Check whether the port settings in IDE are improper or exceed the limits by choosing "Property > OPCUA > Server > Options" to view the maximum session timeout interval.
0x10065010	Warning	Maximum number of sessions out of limit	Among the issued parameters, the maximum number of sessions exceeds the limit (which has been reset to the default value).	Check whether the port settings in IDE are improper or exceed the limits by choosing "Property > OPCUA > Server > Options" to view the maximum number of sessions.
0x10065011	Critical	Sampling interval out of limit	Among the issued parameters, the sampling interval exceeds the limit.	Check whether the port settings in IDE are improper or exceed the limits by choosing "Property > OPCUA > Server > Options" to view the minimum sampling interval.
0x10065012	Warning	Publishing interval out of limit	Among the issued parameters, the publishing interval exceeds the limit (which has been reset to the default value).	Check whether the port settings in IDE are improper or exceed the limits by choosing "Property > OPCUA > Server > Options" to view the minimum publishing interval.
0x10065013	Warning	Maximum number of subscriptions per session out of limit	Among the issued parameters, the maximum number of subscriptions per session exceeds the limit (which has been reset to the default value).	Check whether the port settings in IDE are improper or exceed the limits by choosing "Property > OPCUA > Server > Options" to view the maximum number of subscriptions per session.

Fault Code	Level	Description	Cause	Remedy
0x10065014	Warning	Maximum number of monitored items per subscription out of limit	Among the issued parameters, the maximum monitoring count per subscription exceeds the limit (which has been reset to the default value).	Check whether the port settings in IDE are improper or exceed the limits by choosing "Property > OPCUA > Server > Options" to view the maximum monitoring count per subscription.
0x10065015	Fatal	Configuration file not exist	The INI configuration file is missing.	Reload the device firmware.
0x10065016	Warning	Server certificate about to expire	Server certificate about to expire	1. Check whether the system time is set incorrectly. 2. After modifying the device IP, refresh the certificate.
0x10065017	Fatal	Certificate content corrupted	The server certificate exists, but the content is corrupted.	Re-download the full project before starting the device.
0x10065201	Fatal	Out of memory	Insufficient memory	Check whether any memory is available.
0x10065401	Fatal	Out of memory	Insufficient memory	Check whether any memory is available.
0x10065402	Warning	Excessive sessions	The number of session connections exceeds the limit.	Check whether the number of sessions connected to the server exceeds the limit.
0x10065404	Information	Session status changed to: Created	The session is created with session ID displayed in the attachment information bar.	-
0x10065405	Information	Session status changed to: Activated	The session is activated with session ID displayed in the attachment information bar.	-

Fault Code	Level	Description	Cause	Remedy
0x10065406	Information	Session status changed to: Timeout	The session is timed out with session ID displayed in the attachment information bar.	-
0x10065407	Information	Session status changed to: Closed	The session is closed with session ID displayed in the attachment information bar.	-
0x10065601	Fatal	Out of memory	Insufficient memory	Check whether any memory is available.
0x10065602	Warning	Excessive read/write operations	During properties and historical data reading or writing, the number of read/write requests in a single frame exceeds the upper limit.	1. Check whether the number of nodes per read/write operation at the client exceeds the maximum number of nodes per read/write operation at the server. Does it comply with product specifications?
0x10065603	Warning	Node value unreadable	The node's permissions do not allow read access when the variable value property is read.	Check whether "Public variable" (OPC UA variable) is selected for the variable node on the configuration page.
0x10065605	Warning	No data in indexRange	No data is found for the specified indexRange during node property reading.	1. Check whether the numerical variable in the server is an array, whether the array size is matched, and whether any value is null. 2. Check whether the read request at the client is within the array range.
0x10065606	Warning	Node value unwritable	The node's permissions do not allow write access when the variable value property is written.	Check whether "Public variable" (OPC UA variable) is selected on the configuration page. Check whether the variable is writable.

Fault Code	Level	Description	Cause	Remedy
0x10065801	Fatal	Out of memory	Insufficient memory	Check whether any memory is available.
0x10065802	Warning	Excessive subscriptions	The number of subscriptions in a single session exceeds the upper limit during subscription creation or transfer.	Check whether the number of session subscriptions of the server exceeds the maximum number. Check whether it matches the product specifications. Check whether the total number of subscriptions meets the product specifications.
0x10065803	Information	Subscription status changed to: Creating	The subscription is created with subscription ID displayed in the attachment information bar.	-
0x10065804	Information	Subscription status changed to: Normal	The subscription status is changed to Normal with subscription ID displayed in the attachment information bar.	-
0x10065805	Information	Subscription status changed to: Late	The subscription status is changed to Late with subscription ID displayed in the attachment information bar.	-
0x10065806	Information	Subscription status changed to: Closed	The subscription is closed with subscription ID displayed in the attachment information bar.	-
0x10065807	Warning	Excessive monitored nodes	The number of monitored nodes exceeds the limit.	Check whether the number of monitor request nodes requested by the client has reached the product specification limit.
0x10065A01	Fatal	Out of memory	Insufficient memory	Check whether any memory is available.

Fault Code	Level	Description	Cause	Remedy
0x10065C01	Fatal	Out of memory	Insufficient memory	Check whether any memory is available.
0x10066101	Fatal	Clock safety module CLK1 diagnostic procedure abnormal	Primary clock oscillator failure	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066102	Fatal	Clock safety module CLK2 diagnostic procedure abnormal	PLL output clock drift	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066103	Fatal	Clock safety module CLK3 diagnostic procedure abnormal	Dual clock comparison result abnormal	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066105	Fatal	Clock safety module CLK7 diagnostic procedure abnormal	Clock module static configuration register readback comparison error	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066106	Fatal	Clock safety module CLK8 diagnostic procedure abnormal	Clock module write configuration readback comparison error	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066107	Fatal	Clock safety module CLK-T1 diagnostic procedure abnormal	Dual clock comparison (DCC) function abnormal	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066201	Warning	Single-bit ECC error on the DDR module	A single-bit ECC error has occurred on the DDR module.	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066202	Fatal	Double-bit ECC error on the DDR module	A double-bit ECC error has occurred on the DDR module.	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066203	Fatal	DDR safety module DDR11 diagnostic procedure abnormal	DDR module write configuration readback comparison error	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066204	Fatal	DDR safety module DDR12 diagnostic procedure abnormal	DDR module static configuration register readback comparison error	After confirming it is safe, power cycle the system and check whether normal operation is restored.

Fault Code	Level	Description	Cause	Remedy
0x10066205	Fatal	DDR safety module DDR9 diagnostic procedure abnormal	DDR module basic function abnormal	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066206	Fatal	DDR safety module DDR-T7 diagnostic procedure abnormal	DDR module ECC function abnormal	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066301	Fatal	CBASS safety module CBA4 diagnostic procedure abnormal	A parity error occurs in the chip internal bus.	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066302	Fatal	CBASS safety module CBA6 diagnostic procedure abnormal	CBASS module static configuration register readback comparison error	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066303	Fatal	CBASS safety module CBA7 diagnostic procedure abnormal	CBASS module write configuration readback comparison error	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066304	Fatal	CBASS safety module CBA9 diagnostic procedure abnormal	Invalid bus transaction generated on the internal chip bus	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066306	Fatal	CBASS safety module CBA-T1 diagnostic procedure abnormal	CBASS parity check function abnormal	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066401	Fatal	MMR safety module MMR1 diagnostic procedure abnormal	MMR module static configuration register readback comparison error	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066402	Fatal	MMR safety module MMR2 diagnostic procedure abnormal	MMR module write configuration readback comparison error	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066501	Warning	Single-bit ECC error on the MSRAM safety module	A single-bit ECC error has occurred on the MSRAM safety module.	After confirming it is safe, power cycle the system and check whether normal operation is restored.

Fault Code	Level	Description	Cause	Remedy
0x10066502	Fatal	Double-bit ECC error on the MSRAM safety module	A double-bit ECC error has occurred on the MSRAM safety module.	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066503	Fatal	MSRAM safety module MSRAM-T1 diagnostic procedure abnormal	MSRAM safety module ECC function abnormal	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066601	Fatal	R5F safety module R5F.CPU3 diagnostic procedure abnormal	Illegal memory access by the R5F core. MPU triggered protection.	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066602	Fatal	R5F safety module R5F.CPU5 diagnostic procedure abnormal	Illegal instruction access by the R5F core	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066603	Fatal	R5F safety module R5F.CPU6 diagnostic procedure abnormal	R5F module static configuration register readback comparison error	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066604	Fatal	R5F safety module R5F.CPU7 diagnostic procedure abnormal	R5F module write configuration readback comparison error	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066606	Warning	Single-bit ECC error on the R5F safety module	A single-bit ECC error has occurred on the R5F safety module.	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066607	Fatal	Double-bit ECC error on the R5F safety module	A double-bit ECC error has occurred on the R5F safety module.	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066608	Fatal	R5F safety module R5F.RAM-T1 diagnostic procedure abnormal	R5F safety module ECC function abnormal	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066701	Fatal	VIM safety module R5F.VIM4 diagnostic procedure abnormal	VIM module static configuration register readback comparison error	After confirming it is safe, power cycle the system and check whether normal operation is restored.

Fault Code	Level	Description	Cause	Remedy
0x10066702	Fatal	VIM safety module R5F.VIM5 diagnostic procedure abnormal	VIM module write configuration readback comparison error	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066703	Warning	Single-bit ECC error on the VIM safety module	A single-bit ECC error has occurred on the VIM safety module.	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066704	Fatal	Double-bit ECC error on the VIM safety module	A double-bit ECC error has occurred on the VIM safety module.	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066705	Fatal	VIM safety module R5F.VIM2 diagnostic procedure abnormal	PBIST memory check error	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066707	Fatal	VIM safety module R5F.VIM-T1 diagnostic procedure abnormal	VIM safety module ECC function abnormal	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066801	Fatal	WWDT safety module RTI1 diagnostic procedure abnormal	WWDT module static configuration register readback comparison error	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066802	Fatal	WWDT safety module RTI2 diagnostic procedure abnormal	WWDT module write configuration readback comparison error	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066901	Fatal	Timers safety module TIM1 diagnostic procedure abnormal	Dual timer comparison error, Timer 1oo2 error	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066902	Fatal	Timers safety module TIM2 diagnostic procedure abnormal	Timer basic function test result abnormal	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066903	Fatal	Timers safety module TIM3 diagnostic procedure abnormal	Timers module write configuration readback comparison error	After confirming it is safe, power cycle the system and check whether normal operation is restored.

Fault Code	Level	Description	Cause	Remedy
0x10066904	Fatal	Timers safety module TIM4 diagnostic procedure abnormal	Timers module write configuration readback comparison error	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066A01	Fatal	ESM safety module ESM1 diagnostic procedure abnormal	ESM module static configuration register readback comparison error	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066A02	Fatal	ESM safety module ESM2 diagnostic procedure abnormal	ESM module write configuration readback comparison error	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066A03	Fatal	ESM safety module ESM3 diagnostic procedure abnormal	ESM module basic function test result abnormal	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066B01	Fatal	Program execution sequence monitoring fault	The program did not follow the expected execution path.	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066C01	Fatal	External window watchdog triggered.	Timeout due to failure to feed the watchdog.	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066C02	Fatal	External window watchdog triggered.	Feed the watchdog ahead of schedule	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066C03	Fatal	Internal watchdog action	Timeout due to failure to feed the watchdog.	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066C04	Fatal	External window watchdog not enabled	The external window watchdog is not enabled.	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0x10066D01	Fatal	Abnormal execution of CPU safety diagnostic measures	Abnormal execution of CPU safety diagnostic measures	After confirming it is safe, power cycle the system and check whether normal operation is restored.
0xA1C20000	Information	No fault	/	/

Fault Code	Level	Description	Cause	Remedy
0xA1C21001	Critical	The number of I/O modules configured is more than the number of I/O modules actually scanned.	The number of the configured modules exceeds the actual number of I/O modules.	1. Check the number and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA1C21002	Critical	The number of I/O modules configured is less than the number of I/O modules actually scanned.	The number of the configured modules is less than the actual number of I/O modules.	1. Check the number and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA1C21003	Critical	I/O module type mismatch	The number of the configured modules does not match the actual number of I/O modules.	1. Check the number and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA1C21004	Critical	An error occurred during I/O module parameter configuration.	It failed to configure the I/O module on the protocol stack.	1. Check the number, type and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA1C21005	Critical	An error occurred while switching the I/O module status.	It failed to set the I/O module on the protocol stack.	1. Check the number, type and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA1C21006	Critical	I/O module communication disconnection	Communication interruption between the I/O module and the PLC	1. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0xA1C21007	Critical	Protocol stack failed to scan for modules	The device did not detect the I/O modules or the I/O modules are not inserted.	1. Check whether the expansion module is properly installed and supplied with power. 2. Check the contact of the communication interface of the expansion module or restart the rack.
0xA1C21008	Critical	The power of the actual inserted modules exceed the limit.	The sum of rated currents for the modules inserted actually exceeds 2 A of the bus current.	1. Check the number and installation order of the expansion modules. 2. Check if the PS2 relay power is used.
0xA1C21009	Critical	I/O module state transition timeout	I/O module state transition timeout (30s): OP state not reached.	1. Check the number and installation order of the expansion modules. 2. Check for any unexpected module disconnections. 3. Check if the load of the current project configuration is too high. 4. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0320000	Information	No fault	/	/
0xA0321004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0321005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0321006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0xA0325001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0325002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0325003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0325011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0330000	Information	No fault	/	/
0xA0331004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0331005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0331006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0335001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0335002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.

Fault Code	Level	Description	Cause	Remedy
0xA0335003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0335011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0340000	Information	No fault	/	/
0xA0341004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0341005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0341006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0345001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0345002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0345003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0345011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0FB0000	Information	No fault	/	/

Fault Code	Level	Description	Cause	Remedy
0xA0FB1004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0FB1005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0FB1006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0FB5001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0FB5002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0FB5003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0FB5011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA03C0000	Information	No fault	/	/
0xA03C1004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA03C1005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0xA03C1006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA03C5001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA03C5002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA03C5003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA03C5011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA03D0000	Information	No fault	/	/
0xA03D1004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA03D1005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA03D1006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0xA03D5001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA03D5002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA03D5003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA03D5011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA03E0000	Information	No fault	/	/
0xA03E1004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA03E1005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA03E1006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA03E5001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA03E5002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.

Fault Code	Level	Description	Cause	Remedy
0xA03E5003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA03E5011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA03F0000	Information	No fault	/	/
0xA03F1004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA03F1005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA03F1006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA03F5001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA03F5002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA03F5003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA03F5011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0400000	Information	No fault	/	/

Fault Code	Level	Description	Cause	Remedy
0xA0401004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0401005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0401006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0405001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0405002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0405003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0405011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0410000	Information	No fault	/	/
0xA0411004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0411005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0xA0411006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0415001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0415002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0415003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0415011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA2330000	Information	No fault	/	/
0xA2331004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA2331005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA2331006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0xA2335001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA2335002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA2335003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA2335011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA2310000	Information	No fault	/	/
0xA2311004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA2311005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA2311006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA2315001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA2315002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.

Fault Code	Level	Description	Cause	Remedy
0xA2315003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA2315011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA1680000	Information	No fault	/	/
0xA1681004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA1681005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA1681006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA1685001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA1685002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA1685003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA1685011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0640000	Information	No fault	/	/

Fault Code	Level	Description	Cause	Remedy
0xA0641004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0641005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0641006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0645001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0645002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0645003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0645011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0660000	Information	No fault	/	/
0xA0661004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0661005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0xA0661006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0665001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0665002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0665003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0665011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0650000	Information	No fault	/	/
0xA0651004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0651005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0651006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0xA0655001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0655002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0655003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0655011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0460000	Information	No fault	/	/
0xA0461004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0461005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0461006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0465001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0465002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.

Fault Code	Level	Description	Cause	Remedy
0xA0465003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0465011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0465021	Critical	Module ADC malfunction	The ADC of the module malfunctions.	1. Check the module for hardware faults. 2. Replace the module.
0xA0465023	Critical	ADC reference voltage error	The ADC reference voltage is abnormal.	1. Check the module for hardware faults. 2. Replace the module.
0xA0466001	Critical	Channel open circuit	Data channel open circuit	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA0466002	Critical	Channel short circuit	The data channel is short-circuited.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA0466003	Critical	The channel data exceeds the upper limit.	The channel data exceeds the configured upper limit.	1. Check whether the external input signal of the input module is abnormal.
0xA0466004	Critical	The channel data is below the lower limit.	The channel data is below the configured lower limit.	1. Check whether the external input signal of the input module is abnormal.
0xA0466005	Critical	The channel data overflows.	The input data on the channel overflows.	1. Check whether the external input signal of the input module is abnormal.
0xA0466006	Critical	The channel data underflows.	The input data on the channel underflows.	1. Check whether the external input signal of the input module is abnormal.
0xA0490000	Information	No fault	/	/

Fault Code	Level	Description	Cause	Remedy
0xA0491004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0491005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0491006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0495001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0495002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0495003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0495004	Critical	ADC configuration error	ADC configuration error	1. Check the module for hardware faults. 2. Replace the module.
0xA0495021	Critical	Module ADC malfunction	The ADC of the module malfunctions.	1. Check the module for hardware faults. 2. Replace the module.
0xA0496001	Critical	Channel open circuit	Data channel open circuit	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.

Fault Code	Level	Description	Cause	Remedy
0xA0496002	Critical	Channel short circuit	The data channel is short-circuited.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA0496003	Critical	The channel data exceeds the upper limit.	The channel data exceeds the configured upper limit.	1. Check whether the external input signal of the input module is abnormal.
0xA0496004	Critical	The channel data is below the lower limit.	The channel data is below the configured lower limit.	1. Check whether the external input signal of the input module is abnormal.
0xA0496005	Critical	The channel data overflows.	The input data on the channel overflows.	1. Check whether the external input signal of the input module is abnormal.
0xA0496006	Critical	The channel data underflows.	The input data on the channel underflows.	1. Check whether the external input signal of the input module is abnormal.
0xA05A0000	Information	No fault	/	/
0xA05A1004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA05A1005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA05A1006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA05A5001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.

Fault Code	Level	Description	Cause	Remedy
0xA05A5002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA05A5003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA05A5011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA05A6001	Critical	Channel open circuit	Data channel open circuit	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA05A6002	Critical	Channel short circuit	The data channel is short-circuited.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA05A6003	Critical	The channel data exceeds the upper limit.	The channel data exceeds the configured upper limit.	1. Check whether the external input signal of the input module is abnormal.
0xA05A6004	Critical	The channel data is below the lower limit.	The channel data is below the configured lower limit.	1. Check whether the external input signal of the input module is abnormal.
0xA05A6005	Critical	The channel data overflows.	The input data on the channel overflows.	1. Check whether the external input signal of the input module is abnormal.
0xA05A6006	Critical	The channel data underflows.	The input data on the channel underflows.	1. Check whether the external input signal of the input module is abnormal.
0xA05F0000	Information	No fault	/	/
0xA05F1004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0xA05F1005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA05F1006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA05F5021	Critical	Module ADC malfunction	The ADC of the module malfunctions.	1. Check the module for hardware faults. 2. Replace the module.
0xA05F6001	Critical	Channel open circuit	Data channel open circuit	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA05F6002	Critical	Channel short circuit	The data channel is short-circuited.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA05F6003	Critical	The channel data exceeds the upper limit.	The channel data exceeds the configured upper limit.	1. Check whether the external input signal of the input module is abnormal.
0xA05F6004	Critical	The channel data is below the lower limit.	The channel data is below the configured lower limit.	1. Check whether the external input signal of the input module is abnormal.
0xA05F6005	Critical	The channel data overflows.	The input data on the channel overflows.	1. Check whether the external input signal of the input module is abnormal.
0xA05F6006	Critical	The channel data underflows.	The input data on the channel underflows.	1. Check whether the external input signal of the input module is abnormal.
0xA05F6008	Critical	Channel ADC malfunction	The channel ADC malfunctions.	1. Check the module for hardware faults. 2. Replace the module.
0xA05B0000	Information	No fault	/	/

Fault Code	Level	Description	Cause	Remedy
0xA05B1004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA05B1005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA05B1006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA05B5001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA05B5002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA05B5003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA05B5011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA05B6001	Critical	Channel open circuit	Data channel open circuit	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA05B6002	Critical	Channel short circuit	The data channel is short-circuited.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.

Fault Code	Level	Description	Cause	Remedy
0xA05B6003	Critical	The channel data exceeds the upper limit.	The channel data exceeds the configured upper limit.	1. Check whether the external input signal of the input module is abnormal.
0xA05B6004	Critical	The channel data is below the lower limit.	The channel data is below the configured lower limit.	1. Check whether the external input signal of the input module is abnormal.
0xA05B6005	Critical	The channel data overflows.	The input data on the channel overflows.	1. Check whether the external input signal of the input module is abnormal.
0xA05B6006	Critical	The channel data underflows.	The input data on the channel underflows.	1. Check whether the external input signal of the input module is abnormal.
0xA0600000	Information	No fault	/	/
0xA0601004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0601005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0601006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0605021	Critical	Module ADC malfunction	The ADC of the module malfunctions.	1. Check the module for hardware faults. 2. Replace the module.
0xA0605031	Critical	Module external cold junction fault	Module external cold junction fault	1. Check the external cold junction sensor and its wiring. 2. If the external wiring is correct, check the internal hardware of the module or replace the module.

Fault Code	Level	Description	Cause	Remedy
0xA0606001	Critical	Channel open circuit	Data channel open circuit	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA0606002	Critical	Channel short circuit	The data channel is short-circuited.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA0606003	Critical	The channel data exceeds the upper limit.	The channel data exceeds the configured upper limit.	1. Check whether the external input signal of the input module is abnormal.
0xA0606004	Critical	The channel data is below the lower limit.	The channel data is below the configured lower limit.	1. Check whether the external input signal of the input module is abnormal.
0xA0606005	Critical	The channel data overflows.	The input data on the channel overflows.	1. Check whether the external input signal of the input module is abnormal.
0xA0606006	Critical	The channel data underflows.	The input data on the channel underflows.	1. Check whether the external input signal of the input module is abnormal.
0xA0606008	Critical	Channel ADC malfunction	The channel ADC malfunctions.	1. Check the module for hardware faults. 2. Replace the module.
0xA060600A	Critical	Internal cold junction fault (open circuit, overflow/underflow)	Internal cold junction fault (open circuit, overflow/ underflow)	1. Check the module for hardware faults. 2. Replace the module.
0xA0780000	Information	No fault	/	/
0xA0781005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0xA0781006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0785001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0785002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0785003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0785011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0350000	Information	No fault	/	/
0xA0351004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0351005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0351006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0xA0355001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0355002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0355003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0355011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0680000	Information	No fault	/	/
0xA0681005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0681006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0685001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0685002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0685003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.

Fault Code	Level	Description	Cause	Remedy
0xA0685005	Critical	External 5 V power supply failure	The external 5 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0685011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0686002	Critical	Channel short circuit	Incorrect channel wiring on the I/O module	1. Check the external wiring of the module channel. 2. Replace the module.
0xA0440000	Information	No fault	/	/
0xA0441005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0441006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0445001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0445002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0445003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0445011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.

Fault Code	Level	Description	Cause	Remedy
0xA0446050	Critical	Short circuit	For any channel that is not outputting, the chip detects a short circuit between the DO and VDD.	/
0xA0446051	Critical	Switch ON disconnection	For any channel that is outputting, the chip detects that the DO is not connected to a load.	/
0xA0446052	Critical	Switch OFF disconnection	For any channel that is not outputting, the chip detects that the DO is not connected to a load.	/
0xA0446053	Critical	Overcurrent	For any channel that is outputting, the chip detects that the output current exceeds the current limit of the chip (2.4 A to 3.2 A, typical value 2.8 A).	/

Fault Code	Level	Description	Cause	Remedy
0xA0446054	Critical	Overload	When the output current is too large, but the overcurrent protection value is not reached, the power consumption of the chip output channel increases and the temperature increases. When the junction temperature reaches 165℃, an overload error occurs.	/
0xA0430000	Information	No fault	/	/
0xA0431005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0431006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0435001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0435002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0435003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.

Fault Code	Level	Description	Cause	Remedy
0xA0435011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0520000	Information	No fault	/	/
0xA0521004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0521005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0521006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0525001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0525002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0525003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0525011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0525021	Critical	Module ADC malfunction	The ADC of the module malfunctions.	1. Check the module for hardware faults. 2. Replace the module.
0xA0525023	Critical	ADC reference voltage error	The ADC reference voltage is abnormal.	1. Check the module for hardware faults. 2. Replace the module.

Fault Code	Level	Description	Cause	Remedy
0xA0526001	Critical	Channel open circuit	Data channel open circuit	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA0526002	Critical	Channel short circuit	The data channel is short-circuited.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA0526003	Critical	The channel data exceeds the upper limit.	The channel data exceeds the configured upper limit.	1. Check whether the external input signal of the input module is abnormal.
0xA0526004	Critical	The channel data is below the lower limit.	The channel data is below the configured lower limit.	1. Check whether the external input signal of the input module is abnormal.
0xA0480000	Information	No fault	/	/
0xA0481004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0481005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0481006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0485001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.

Fault Code	Level	Description	Cause	Remedy
0xA0485002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0485003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0485011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0485021	Critical	Module ADC malfunction	The ADC of the module malfunctions.	1. Check the module for hardware faults. 2. Replace the module.
0xA0485023	Critical	ADC reference voltage error	The ADC reference voltage is abnormal.	1. Check the module for hardware faults. 2. Replace the module.
0xA0486001	Critical	Channel open circuit	Data channel open circuit	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA0486002	Critical	Channel short circuit	The data channel is short-circuited.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA0486003	Critical	The channel data exceeds the upper limit.	The channel data exceeds the configured upper limit.	1. Check whether the external input signal of the input module is abnormal.
0xA0486004	Critical	The channel data is below the lower limit.	The channel data is below the configured lower limit.	1. Check whether the external input signal of the input module is abnormal.
0xA0486005	Critical	The channel data overflows.	The input data on the channel overflows.	1. Check whether the external input signal of the input module is abnormal.
0xA0486006	Critical	The channel data underflows.	The input data on the channel underflows.	1. Check whether the external input signal of the input module is abnormal.

Fault Code	Level	Description	Cause	Remedy
0xA0510000	Information	No fault	/	/
0xA0511004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0511005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0511006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0515001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0515002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0515003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0515011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0515021	Critical	Module ADC malfunction	The ADC of the module malfunctions.	1. Check the module for hardware faults. 2. Replace the module.
0xA0515023	Critical	ADC reference voltage error	The ADC reference voltage is abnormal.	1. Check the module for hardware faults. 2. Replace the module.

Fault Code	Level	Description	Cause	Remedy
0xA0516001	Critical	Channel open circuit	Data channel open circuit	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA0516002	Critical	Channel short circuit	The data channel is short-circuited.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA0516003	Critical	The channel data exceeds the upper limit.	The channel data exceeds the configured upper limit.	1. Check whether the external input signal of the input module is abnormal.
0xA0516004	Critical	The channel data is below the lower limit.	The channel data is below the configured lower limit.	1. Check whether the external input signal of the input module is abnormal.
0xA0470000	Information	No fault	/	/
0xA0471004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0471005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0471006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0475001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.

Fault Code	Level	Description	Cause	Remedy
0xA0475002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0475003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0475011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0475021	Critical	Module ADC malfunction	The ADC of the module malfunctions.	1. Check the module for hardware faults. 2. Replace the module.
0xA0475023	Critical	ADC reference voltage error	The ADC reference voltage is abnormal.	1. Check the module for hardware faults. 2. Replace the module.
0xA0476001	Critical	Channel open circuit	Data channel open circuit	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA0476002	Critical	Channel short circuit	The data channel is short-circuited.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA0476003	Critical	The channel data exceeds the upper limit.	The channel data exceeds the configured upper limit.	1. Check whether the external input signal of the input module is abnormal.
0xA0476004	Critical	The channel data is below the lower limit.	The channel data is below the configured lower limit.	1. Check whether the external input signal of the input module is abnormal.
0xA0476005	Critical	The channel data overflows.	The input data on the channel overflows.	1. Check whether the external input signal of the input module is abnormal.
0xA0476006	Critical	The channel data underflows.	The input data on the channel underflows.	1. Check whether the external input signal of the input module is abnormal.

Fault Code	Level	Description	Cause	Remedy
0xA0850000	Information	No fault	/	/
0xA0851005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0851006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0855001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0855002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0855003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0855011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0856030	Critical	SSI module channel 24 V output error	24 V output open circuit, overvoltage, or undervoltage	Check the output voltage.
0xA0856031	Critical	SSI module channel communication error	SSI module D+ and D- open circuit or crossed	Check the connection between D+ and D- terminals.
0xA0856032	Critical	SSI module channel data error	Data loss during transmission	Check whether the baud rate is configured properly.
0xA07A0000	Information	No fault	/	/
0xA07A1005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0xA07A1006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA07A5001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA07A5002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA07A5003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA07A5011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA07A6800	Critical	Phase A open circuit	Phase A open circuit	/
0xA07A6801	Critical	Phase B open circuit	Phase B open circuit	/
0xA07A6802	Critical	Frequency out of limit	Differential interface: frequency > 4 M; Single-ended interface: frequency > 500K	/
0xA07A6803	Critical	Count value overflow	Count value >= Maximum value	/
0xA07A6804	Critical	Count value underflow	Count value <= Minimum value	/
0xA07A6805	Critical	Comparison function 0 quantity/step out of range	The quantity is greater than 1024 or less than 1, or the step is less than 1.	/

Fault Code	Level	Description	Cause	Remedy
0xA07A6806	Critical	Comparison function 1 quantity/step out of range	The quantity is greater than 1024 or less than 1, or the step is less than 1.	/
0xA07A6807	Critical	Comparison function 2 quantity/step out of range	The quantity is greater than 1024 or less than 1, or the step is less than 1.	/
0xA07A6808	Critical	Comparison function 3 quantity/step out of range	The quantity is greater than 1024 or less than 1, or the step is less than 1.	/
0xA07A6809	Critical	The output hold time/ pulse number is not configured properly for the comparison function 0.	The hold time is smaller than 10, or the pulse number is smaller than 1.	/
0xA07A680A	Critical	Comparison function 1 output hold time/pulse number not configured properly	The hold time is smaller than 10, or the pulse number is smaller than 1.	/
0xA07A680B	Critical	Comparison function 2 output hold time/pulse number not configured properly	The hold time is smaller than 10, or the pulse number is smaller than 1.	/
0xA07A680C	Critical	Comparison function 3 output hold time/pulse number not configured properly	The hold time is smaller than 10, or the pulse number is smaller than 1.	/
0xA07A680D	Critical	The output start position and step direction is not configured properly for the comparison function 0.	Start and end positions are equal; start position < end position, step size < 0; start position > end position, step size > 0.	/

Fault Code	Level	Description	Cause	Remedy
0xA07A680E	Critical	The output start position and step direction is not configured properly for the comparison function 1.	Start and end positions are equal; start position < end position, step size < 0; start position > end position, step size > 0.	/
0xA07A680F	Critical	The output start position and step direction is not configured properly for the comparison function 2.	Start and end positions are equal; start position < end position, step size < 0; start position > end position, step size > 0.	/
0xA07A6810	Critical	The output start position and step direction is not configured properly for the comparison function 3.	Start and end positions are equal; start position < end position, step size < 0; start position > end position, step size > 0.	/
0xA07A6811	Critical	The comparison value is out of range for the comparison function 0.	The comparison value is greater than the maximum value or less than the minimum value.	/
0xA07A6812	Critical	Comparison function 1 comparison value out of range	The comparison value is greater than the maximum value or less than the minimum value.	/
0xA07A6813	Critical	Comparison function 2 comparison value out of range	The comparison value is greater than the maximum value or less than the minimum value.	/

Fault Code	Level	Description	Cause	Remedy
0xA07A6814	Critical	Comparison function 3 comparison value out of range	The comparison value is greater than the maximum value or less than the minimum value.	/
0xA07A6815	Critical	The type is not configured properly for the comparison function 0.	The comparison type is not set to the allowed value, which includes multi-point comparison and uniformly-spaced comparison.	/
0xA07A6816	Critical	Comparison function 1 type not configured properly	The comparison type is not set to the allowed value, which includes multi-point comparison and uniformly-spaced comparison.	/
0xA07A6817	Critical	Comparison function 2 type not configured properly	The comparison type is not set to the allowed value, which includes multi-point comparison and uniformly-spaced comparison.	/
0xA07A6818	Critical	Comparison function 3 type not configured properly	The comparison type is not set to the allowed value, which includes multi-point comparison and uniformly-spaced comparison.	/
0xA07A6819	Critical	The output control mode is not configured properly for the comparison function 0.	The control mode is not set to the allowed value, which includes time control, pulse control, and level control.	/

Fault Code	Level	Description	Cause	Remedy
0xA07A681A	Critical	Comparison function 1 output control mode not configured properly	The control mode is not set to the allowed value, which includes time control, pulse control, and level control.	/
0xA07A681B	Critical	Comparison function 2 output control mode not configured properly	The control mode is not set to the allowed value, which includes time control, pulse control, and level control.	/
0xA07A681C	Critical	Comparison function 3 output control mode not configured properly	The control mode is not set to the allowed value, which includes time control, pulse control, and level control.	/
0xA07A681D	Critical	DO0 is shorted.	DO0 is shorted.	/
0xA07A681E	Critical	DO1 short circuit	DO1 short circuit	/
0xA07A681F	Critical	DO2 short circuit	DO2 short circuit	/
0xA07A6820	Critical	DO3 short circuit	DO3 short circuit	/
0xA07A2100	Critical	Preset value out of range	The preset value is larger than the maximum value or smaller than the minimum value.	/

Fault Code	Level	Description	Cause	Remedy
0xA07A2101	Critical	Counter configuration error	1. The instruction enabled the single-ended counting function, but the DI is not configured for counting. 2. The single-ended interface enabled by the instruction does not match the counting interface configured on the DI. 3. The counter is configured as a differential interface, but DI0 or DI1 is set to counting mode.	/
0xA07A2102	Critical	Signal type not supported	The signal type is not set to an allowed value, which includes phase A/B 1x frequency, phase A/B 2x frequency, phase A/B 4x frequency, pulse + direction, CW/CCW, single-phase counting (A), and single-phase counting (B).	/
0xA07A2103	Critical	Counter limit error	Maximum value <= Minimum value	/
0xA07A2104	Critical	Incorrect setting of unit pulse increment	The unit pulse increment is less than 1.	/

Fault Code	Level	Description	Cause	Remedy
0xA07A2105	Critical	Incorrect setting of time base for speed measurement	The time base is not set to an allowed value. See object 800nh .3 in the object dictionary for allowed values.	/
0xA07A2106	Critical	Incorrect setting of input filter parameter	The input filter parameter is not set to the allowed value. See object 800nh.2 in the object dictionary for allowed values.	/
0xA07A2107	Critical	Incorrect setting of frequency sampling time	The frequency sampling time is not set to the allowed value. See object 800nh .5 in the object dictionary for allowed values.	/
0xA07A2108	Critical	Incorrect configuration of the clearing function	The clearing function is enabled through commands, but the DI is not configured with the clearing function.	/
0xA07A2109	Critical	Incorrect configuration of the preset function	The preset function is enabled through commands, but the DI is not configured with the preset function.	/

Fault Code	Level	Description	Cause	Remedy
0xA07A210A	Critical	Incorrect configuration of comparison 0 function	The comparison function is enabled through commands, but the DO is not configured with the comparison function.	/
0xA07A210B	Critical	Incorrect configuration of comparison 1 function	The comparison function is enabled through commands, but the DO is not configured with the comparison function.	/
0xA07A210C	Critical	Incorrect configuration of comparison 2 function	The comparison function is enabled through commands, but the DO is not configured with the comparison function.	/
0xA07A210D	Critical	Incorrect configuration of comparison 3 function	The comparison function is enabled through commands, but the DO is not configured with the comparison function.	/
0xA07A210E	Critical	Incorrect configuration of probe 0 function	The probe function is enabled through commands, but the DI is not configured with the probe function.	/

Fault Code	Level	Description	Cause	Remedy
0xA07A210F	Critical	Incorrect configuration of probe 1 function	The probe function is enabled through commands, but the DI is not configured with the probe function.	/
0xA07A2110	Critical	Incorrect configuration of probe 2 function	The probe function is enabled through commands, but the DI is not configured with the probe function.	/
0xA07A2111	Critical	Incorrect configuration of probe 3 function	The probe function is enabled through commands, but the DI is not configured with the probe function.	/
0xA07A2112	Critical	DO0 parameter error	The parameter is not set to the allowed value, which includes normal output and comparison output.	/
0xA07A2113	Critical	DO1 parameter error	The parameter is not set to the allowed value, which includes normal output and comparison output.	/
0xA07A2114	Critical	DO2 parameter error	The parameter is not set to the allowed value, which includes normal output and comparison output.	/

Fault Code	Level	Description	Cause	Remedy
0xA07A2115	Critical	DO3 parameter error	The parameter is not set to the allowed value, which includes normal output and comparison output.	/
0xA07A2116	Critical	DI0 parameter error	The parameter is not set to the allowed value, which includes normal input, probe function, counting, clear, preset, and gating.	/
0xA07A2117	Critical	DI1 parameter error	The parameter is not set to the allowed value, which includes normal input, probe function, counting, clear, preset, and gating.	/
0xA07A2118	Critical	DI2 parameter error	The parameter is not set to the allowed value, which includes normal input, probe function, counting, clear, preset, and gating.	/
0xA07A2119	Critical	DI3 parameter error	The parameter is not set to the allowed value, which includes normal input, probe function, counting, clear, preset, and gating.	/
0xA05D0000	Information	No fault	/	/

Fault Code	Level	Description	Cause	Remedy
0xA05D1004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA05D1005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA05D1006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA05D5001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA05D5002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA05D5003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA05D5011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA05D5021	Critical	Module ADC malfunction	The ADC of the module malfunctions.	1. Check the module for hardware faults. 2. Replace the module.
0xA05D5031	Critical	Module external cold junction fault	Module external cold junction fault	1. Check the external cold junction sensor and its wiring. 2. If the external wiring is correct, check the internal hardware of the module or replace the module.

Fault Code	Level	Description	Cause	Remedy
0xA05D6001	Critical	Channel open circuit	Data channel open circuit	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA05D6002	Critical	Channel short circuit	The data channel is short-circuited.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA05D6003	Critical	The channel data exceeds the upper limit.	The channel data exceeds the configured upper limit.	1. Check whether the external input signal of the input module is abnormal.
0xA05D6004	Critical	The channel data is below the lower limit.	The channel data is below the configured lower limit.	1. Check whether the external input signal of the input module is abnormal.
0xA05D6005	Critical	The channel data overflows.	The input data on the channel overflows.	1. Check whether the external input signal of the input module is abnormal.
0xA05D6006	Critical	The channel data underflows.	The input data on the channel underflows.	1. Check whether the external input signal of the input module is abnormal.
0xA05D6035	Critical	DO0 output short circuit or overload	Short circuit or overload	1. Check the external wiring for short circuits. 2. Check the load for overcurrent. 3. Check the module for damage.
0xA05D6036	Critical	DO1 output short circuit or overload	Short circuit or overload	1. Check the external wiring for short circuits. 2. Check the load for overcurrent. 3. Check the module for damage.
0xA05E0000	Information	No fault	/	/
0xA05E1004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0xA05E1005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA05E1006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA05E5011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA05E5021	Critical	Module ADC malfunction	The ADC of the module malfunctions.	1. Check the module for hardware faults. 2. Replace the module.
0xA05E6001	Critical	Channel open circuit	Data channel open circuit	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA05E6003	Critical	The channel data exceeds the upper limit.	The channel data exceeds the configured upper limit.	1. Check whether the external input signal of the input module is abnormal.
0xA05E6004	Critical	The channel data is below the lower limit.	The channel data is below the configured lower limit.	1. Check whether the external input signal of the input module is abnormal.
0xA05E6005	Critical	The channel data overflows.	The input data on the channel overflows.	1. Check whether the external input signal of the input module is abnormal.
0xA05E6006	Critical	The channel data underflows.	The input data on the channel underflows.	1. Check whether the external input signal of the input module is abnormal.
0xA05C0000	Information	No fault	/	/
0xA05C1004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0xA05C1005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA05C1006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA05C5011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA05C5021	Critical	Module ADC malfunction	The ADC of the module malfunctions.	1. Check the module for hardware faults. 2. Replace the module.
0xA05C6001	Critical	Channel open circuit	Data channel open circuit	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA05C6003	Critical	The channel data exceeds the upper limit.	The channel data exceeds the configured upper limit.	1. Check whether the external input signal of the input module is abnormal.
0xA05C6004	Critical	The channel data is below the lower limit.	The channel data is below the configured lower limit.	1. Check whether the external input signal of the input module is abnormal.
0xA05C6005	Critical	The channel data overflows.	The input data on the channel overflows.	1. Check whether the external input signal of the input module is abnormal.
0xA05C6006	Critical	The channel data underflows.	The input data on the channel underflows.	1. Check whether the external input signal of the input module is abnormal.
0xA08A0000	Information	No fault	/	/
0xA08A1004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0xA08A1005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA08A1006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA08A2003	Critical	2SCOM-MDB module placed in the wrong slot	The 2SCOM-MDB module is placed in an incorrect slot.	The module must be placed in one of the first four slots.
0xA08A2004	Critical	The number of instructions (sub-modules) on serial port 0/1 of the 2SCOM-MDB module exceeds 31.	The number of instructions (submodules) for serial port 0/1 exceeds 31.	Reduce the number of instructions (submodules) for serial port 0/1.
0xA08A2005	Critical	The process data of the 2SCOM-MDB module exceeds 480 bytes.	The module process data (whether reading or writing) exceeds 480 bytes.	Reduce the module process data.
0xA08A2030	Critical	The serial port parameters of the 2SCOM-MDB module channel are configured incorrectly.	The configuration parameters sent to the module are outside the set range or do not meet the format requirement.	Check the configuration parameters according to the instructions.
0xA08A2033	Critical	The port parameters of the 2SCOM-MDB module channel are configured incorrectly.	The configuration parameters sent to the module are outside the set range or do not meet the format requirement.	Check the configuration parameters according to the instructions.

Fault Code	Level	Description	Cause	Remedy
0xA08A2040	Critical	The Modbus slave address of the 2SCOM-MDB module channel is configured incorrectly.	The configuration parameters sent to the module are outside the set range or do not meet the format requirement.	Check the configuration parameters according to the instructions.
0xA08A6044	Critical	Noise detected on 2SCOM-MDB module channel	Noise has been detected on the channel signal line.	Check whether disturbance exists.
0xA08A6045	Critical	2SCOM-MDB module channel parity bit error	Parity bit error	Check whether the parity bit configuration parameter is correct and whether the parity bit calculation is correct.
0xA08A6046	Critical	Channel instructions (submodules) of the 2SCOM-MDB module are configured incorrectly.	The added instruction (submodule) type does not match the operation mode of serial port 0/1.	Check whether the added instruction (submodule) type matches the operation mode of serial port 0/1.
0xA08A6060	Critical	M: Modbus response parameter mismatch	Transmit/receive instructions mismatch (The parameter in the response message does not match that in the query), or the instruction is not supported by the master station (the instruction code is not supported in the master query message).	Check for communication errors. Verify if the function code is supported by the master station.
0xA08A6061	Critical	M: Modbus slave ID mismatch	Transmit/receive address mismatch (The slave address in response message and query message do not match.)	Check for communication errors.

Fault Code	Level	Description	Cause	Remedy
0xA08A6062	Critical	M: Modbus response address mismatch	The start address in response message does not match that in the query.	Check for communication faults. Use a serial port assistant to monitor communication data frames. If this error occurs frequently, set the free coefficient in the startup parameters to 0.
0xA08A6063	Critical	M: Modbus response data length mismatch	Data frame length error: The received frame length does not conform to the standard, or the received frame length is less than 5.	Check for communication errors.
0xA08A6064	Critical	M: Modbus communication timeout	Receiving timeout	1. Check if the wiring is correct. 2. Check if the module configuration parameters match the parameter settings of the slave. 3. Use a serial port assistant to monitor communication data, and check if the slave responds.
0xA08A6065	Critical	M: Modbus CRC check fault	CRC check error	Check for communication errors.
0xA08A6066	Critical	Illegal function code	Illegal function code	The slave device does not support this parameter.
0xA08A6067	Critical	Illegal data address	Illegal data address	The specified data address does not exist in the slave device.
0xA08A6068	Critical	Illegal data value	The data value is illegal.	The specified data exceeds the range or is not allowed.
0xA08A6069	Critical	Slave device fault	Slave device fault	An unknown error occurred during response processing on the slave device.
0xA08A6500	Critical	modbus slave offline	In the modbus communication network, a slave has lost communication.	1. Check the modbus configuration, including station number/baud rate and so on. 2. Check the hardware wiring and termination resistors.
0xA08B0000	Information	No fault	/	/

Fault Code	Level	Description	Cause	Remedy
0xA08B1004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA08B1005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA08B1006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA08B2003	Critical	2S485-MDB module placed in the wrong slot	The 2S485-MDB module is placed in an incorrect slot.	The module must be placed in one of the first four slots.
0xA08B2004	Critical	The number of instructions (submodules) on serial port 0/1 of the 2S485-MDB module exceeds 31.	The number of instructions (submodules) for serial port 0/1 exceeds 31.	Reduce the number of instructions (submodules) for serial port 0/1.
0xA08B2005	Critical	The process data of the 2S485-MDB module exceeds 480 bytes.	The module process data (whether reading or writing) exceeds 480 bytes.	Reduce the module process data.
0xA08B2040	Critical	The Modbus slave address of the 2S485-MDB module channel is configured incorrectly.	The configuration parameters sent to the module are outside the set range or do not meet the format requirement.	Check the configuration parameters according to the instructions.
0xA08B2041	Critical	The serial port parameters of the 2S485-MDB module channel are configured incorrectly.	The configuration parameters sent to the module are outside the set range or do not meet the format requirement.	Check the configuration parameters according to the instructions.

Fault Code	Level	Description	Cause	Remedy
0xA08B6070	Critical	Noise detected on 2S485-MDB module channel	Noise has been detected on the channel signal line.	Check whether disturbance exists.
0xA08B6071	Critical	2S485-MDB module channel parity bit error	Parity bit error	Check whether the parity bit configuration parameter is correct and whether the parity bit calculation is correct.
0xA08B6072	Critical	Channel instructions (submodules) of the 2S485-MDB module are configured incorrectly.	The added instruction (submodule) type does not match the operation mode of serial port 0/1.	Check whether the added instruction (submodule) type matches the operation mode of serial port 0/1.
0xA08B6060	Critical	M: Modbus response parameter mismatch	Transmit/receive instructions mismatch (The parameter in the response message does not match that in the query), or the instruction is not supported by the master station (the instruction code is not supported in the master query message).	Check for communication errors. Verify if the function code is supported by the master station.
0xA08B6061	Critical	M: Modbus slave ID mismatch	Transmit/receive address mismatch (The slave address in response message and query message do not match.)	Check for communication errors.
0xA08B6062	Critical	M: Modbus response address mismatch	The start address in response message does not match that in the query.	Check for communication faults. Use a serial port assistant to monitor communication data frames. If this error occurs frequently, set the free coefficient in the startup parameters to 0.

Fault Code	Level	Description	Cause	Remedy
0xA08B6063	Critical	M: Modbus response data length mismatch	Data frame length error: The received frame length does not conform to the standard, or the received frame length is less than 5.	Check for communication errors.
0xA08B6064	Critical	M: Modbus communication timeout	Receiving timeout	1. Check if the wiring is correct. 2. Check if the module configuration parameters match the parameter settings of the slave. 3. Use a serial port assistant to monitor communication data, and check if the slave responds.
0xA08B6065	Critical	M: Modbus CRC check fault	CRC check error	Check for communication errors.
0xA08B6066	Critical	Illegal function code	Illegal function code	The slave device does not support this parameter.
0xA08B6067	Critical	Illegal data address	Illegal data address	The specified data address does not exist in the slave device.
0xA08B6068	Critical	Illegal data value	The data value is illegal.	The specified data exceeds the range or is not allowed.
0xA08B6069	Critical	Slave device fault	Slave device fault	An unknown error occurred during response processing on the slave device.
0xA08B6500	Critical	modbus slave offline	In the modbus communication network, a slave has lost communication.	1. Check the modbus configuration, including station number/baud rate and so on. 2. Check the hardware wiring and termination resistors.
0xA0500000	Information	No fault	/	/
0xA0501004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0501005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0xA0501006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0505001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0505002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0505003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0505011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0505022	Critical	Module DAC malfunction	The DAC of the module malfunctions.	1. Check the module for hardware faults. 2. Replace the module.
0xA0506001	Critical	Channel open circuit	Data channel open circuit	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA0506002	Critical	Channel short circuit	The data channel is short-circuited.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0xA0506003	Critical	The channel data exceeds the upper limit.	The channel data exceeds the configured upper limit.	1. Check whether the external input signal of the input module is abnormal.

Fault Code	Level	Description	Cause	Remedy
0xA0506004	Critical	The channel data is below the lower limit.	The channel data is below the configured lower limit.	1. Check whether the external input signal of the input module is abnormal.
0xA0506005	Critical	Channel data overflow	The channel data exceeds the configured high limit.	1. Check whether the external input signal of the input module is abnormal.
0xA0506006	Critical	Channel data underflow	The channel data is below the configured low limit.	1. Check whether the external input signal of the input module is abnormal.
0xA0870000	Information	No fault	/	/
0xA0871004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0871005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0871006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0875001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0875002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0875003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.

Fault Code	Level	Description	Cause	Remedy
0xA0875011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0876500	Critical	The DN slave connected to the master is offline.	In the DeviceNet network, a slave has lost communication.	1. Check the DN configuration, including station number/ baud rate/data length and so on. 2. Check the hardware wiring and termination resistors.
0xA0880000	Information	No fault	/	/
0xA0881004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0881005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0881006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0885001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0885002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0885003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0885011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0820000	Information	No fault	/	/

Fault Code	Level	Description	Cause	Remedy
0xA0821004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0821005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0821006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0822020	Critical	Baud rate configuration error	The baud rate of the CAN channel is configured incorrectly.	1. Check the corresponding configuration parameters.
0xA0822021	Critical	Communication mode configuration error	The communication mode of the CAN channel is incorrectly configured.	1. Check the corresponding configuration parameters.
0xA0822022	Critical	Buffer size configuration error	The buffer size of the CAN channel is configured incorrectly.	1. Check the corresponding configuration parameters.
0xA0822023	Critical	Heartbeat interval configuration error	The heartbeat interval configuration error has occurred in the CAN channel.	1. Check the corresponding configuration parameters.
0xA0822024	Critical	Standard frame upper/ lower limit configuration error	The upper and lower limits for the standard frame on the CAN channels are configured incorrectly.	1. Check the corresponding configuration parameters.

Fault Code	Level	Description	Cause	Remedy
0xA0822025	Critical	Extended frame upper limit configuration error	The upper limit for the extended frame on the CAN channel is incorrectly configured.	1. Check the corresponding configuration parameters.
0xA0822026	Critical	Extended frame lower limit configuration error	The lower limit for the extended frame on the CAN channel is incorrectly configured.	1. Check the corresponding configuration parameters.
0xA0825001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0825002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0825003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0825011	Critical	Chip overheating	The internal chip in the I/O module is overheated.	1. Check the module for hardware faults. 2. Replace the module.
0xA0826020	Critical	Transmit buffer full	The transmit buffer for the CAN channel is full.	1. Reduce the transmit data length or increase the transmit interval. 2. Increase the baud rate.
0xA0826021	Critical	Receive buffer full	The receive buffer for the CAN channel is full.	1. Reduce the transmit data length or increase the transmit interval.
0xA0826022	Critical	Receive frame count error	A receive frame count error has occurred on the CAN channel.	1. Check the frame counter value written to the module over the bus.
0xA0826023	Critical	Bus load rate too high	The bus load ratio of the CAN channel exceeds 30%.	1. Check the data on the bus and the load rate.

Fault Code	Level	Description	Cause	Remedy
0xA0826024	Critical	CAN data transmit error	The transmit error counter (TEC) on the CAN channel has reached 127.	1. Check the connections of the communication line. 2. Check the termination resistor.
0xA0826025	Critical	CAN data receive error	The receive error counter (REC) on the CAN channel has reached 127.	1. Check the connections of the communication line. 2. Check the termination resistor.
0xA0830000	Information	No fault	/	/
0xA0831004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0831005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0831006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0832010	Critical	Baud rate configuration error	The baud rate of the 485 channel is configured incorrectly.	1. Check the corresponding configuration parameters.
0xA0832011	Critical	Communication mode configuration error	The communication mode of the RS485 channel is incorrectly configured.	1. Check the corresponding configuration parameters.
0xA0832012	Critical	Bit-related parameter configuration error	Bit-related parameter configuration error for the RS485 channel	1. Check the corresponding configuration parameters.
0xA0832013	Critical	Data packet size configuration error	Data packet size configuration error for the RS485 channel	1. Check the corresponding configuration parameters.

Fault Code	Level	Description	Cause	Remedy
0xA0832014	Critical	Message interval configuration error	The message interval of the RS485 channel is configured incorrectly.	1. Check the corresponding configuration parameters.
0xA0832015	Critical	Transmit buffer configuration error	The transmit buffer for the RS485 channel is configured incorrectly.	1. Check the corresponding configuration parameters.
0xA0832016	Critical	Receive buffer configuration error	The receive buffer for the RS485 channel is configured incorrectly.	1. Check the corresponding configuration parameters.
0xA0835001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0835002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0835003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0836010	Critical	Transmit buffer full	The transmit buffer for the RS485 channel is full.	1. Reduce the transmit data length or increase the transmit interval. 2. Set the baud rate to a greater value.
0xA0836011	Critical	Receive buffer full	The receive buffer for the RS485 channel is full.	1. Reduce the transmit data length or increase the transmit interval.
0xA0836012	Critical	Receive frame count value error	Receive frame counter error on RS485 channel. Non-consecutive frame counter values.	1. Check the frame counter value written to the module over the local bus.

Fault Code	Level	Description	Cause	Remedy
0xA0836013	Critical	Receive frame segment value error	Receive frame segnum value error on RS485 channel. Non-consecutive frame segnum values.	1. Check the frame segnum value written to the module over the local bus.
0xA0836014	Critical	Communication noise interference	Noise detected on the RS485 channel.	1. Check for noise interference on the corresponding serial port.
0xA0836015	Critical	Data check error	Parity bit fault detected on the RS485 module channel.	1. Check whether the parity bit configuration parameter of the serial port matches the configured value.
0xA0860000	Information	No fault	/	/
0xA0861004	Critical	I/O module configuration failure	I/O module parameter configuration failed.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0861005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0861006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA0862030	Critical	Serial port parameter configuration error.	Serial port parameter configuration error on the COM channel.	1. Check the corresponding configuration parameters.
0xA0862031	Critical	Data packet size configuration error.	Data packet size configuration error on the COM channel.	1. Check the corresponding configuration parameters.
0xA0862032	Critical	Timeout configuration error	Timeout configuration error on the COM channel.	1. Check the corresponding configuration parameters.

Fault Code	Level	Description	Cause	Remedy
0xA0862033	Critical	Port protocol configuration error.	Port protocol configuration error on the COM channel.	1. Check the corresponding configuration parameters.
0xA0865001	Critical	Power supply overvoltage	The supply voltage exceeds the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0865002	Critical	Power supply undervoltage	The supply voltage is below the rated operating voltage of the I/O module.	1. Check the power supply of the module.
0xA0865003	Critical	External 24 V power supply failure	The external 24 V power supply to the I/O modules is lost.	1. Check the power supply of the module.
0xA0866040	Critical	Transmit buffer full	The transmit buffer for the COM channel is full.	1. Reduce the transmit data length or increase the transmit interval. 2. Set the baud rate to a greater value.
0xA0866041	Critical	Receive buffer full	The receive buffer for the COM channel is full.	1. Reduce the transmit data length or increase the transmit interval.
0xA0866042	Critical	Receive frame count value error	Receive frame counter error on RS485 channel. Non-consecutive frame counter values.	1. Check the frame counter value written to the module over the local bus.
0xA0866043	Critical	Receive frame segment value error	Receive frame segnum value error on RS485 channel. Non-consecutive frame segnum values.	1. Check the frame segnum value written to the module over the local bus.
0xA0866044	Critical	Communication noise interference	Noise detected on the COM channel.	1. Check for noise interference on the corresponding serial port.

Fault Code	Level	Description	Cause	Remedy
0xA0866045	Critical	Data check error	Parity bit error on the COM channel.	1. Check whether the parity bit configuration parameter of the serial port matches the configured value.
0xA85C1000	Information	The configured module type matches the type of the actually installed module.	/	/
0xA85C1001	Critical	The configured module type is inconsistent with the type of the module actually installed.	The configured module type is inconsistent with the type of the module actually installed.	1. Check the number and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA85C1002	Critical	Not inserted in the device	Not inserted in the device	1. Check whether the extension card is correctly inserted into the device.
0xA8661000	Information	The configured module type matches the type of the actually installed module.	/	/
0xA8661001	Critical	The configured module type is inconsistent with the type of the module actually installed.	The configured module type is inconsistent with the type of the module actually installed.	1. Check the number and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA8661002	Critical	Not inserted in the device	Not inserted in the device	1. Check whether the extension card is correctly inserted into the device.
0xA8701000	Information	The configured module type matches the type of the actually installed module.	/	/
0xA8701001	Critical	The configured module type is inconsistent with the type of the module actually installed.	The configured module type is inconsistent with the type of the module actually installed.	1. Check the number and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0xA8701002	Critical	Not inserted in the device	Not inserted in the device	1. Check whether the extension card is correctly inserted into the device.
0xA8711000	Information	The configured module type matches the type of the actually installed module.	/	/
0xA8711001	Critical	The configured module type is inconsistent with the type of the module actually installed.	The configured module type is inconsistent with the type of the module actually installed.	1. Check the number and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA8711002	Critical	Not inserted in the device	Not inserted in the device	1. Check whether the extension card is correctly inserted into the device.
0xA87A1000	Information	The configured module type matches the type of the actually installed module.	/	/
0xA87A1001	Critical	The configured module type is inconsistent with the type of the module actually installed.	The configured module type is inconsistent with the type of the module actually installed.	1. Check the number and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA87A1002	Critical	Not inserted in the device	Not inserted in the device	1. Check whether the extension card is correctly inserted into the device.
0xA87B1000	Information	The configured module type matches the type of the actually installed module.	/	/
0xA87B1001	Critical	The configured module type is inconsistent with the type of the module actually installed.	The configured module type is inconsistent with the type of the module actually installed.	1. Check the number and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0xA87B1002	Critical	Not inserted in the device	Not inserted in the device	1. Check whether the extension card is correctly inserted into the device.
0x32C81000	Information	The configured module type matches the type of the actually installed module.	/	/
0x32C81001	Critical	The configured module type is inconsistent with the type of the module actually installed.	The configured module type is inconsistent with the type of the module actually installed.	1. Check the number and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0x32C81002	Critical	Not inserted in the device	Not inserted in the device	1. Check whether the extension card is correctly inserted into the device.
0x32C91000	Information	The configured module type matches the type of the actually installed module.	/	/
0x32C91001	Critical	The configured module type is inconsistent with the type of the module actually installed.	The configured module type is inconsistent with the type of the module actually installed.	1. Check the number and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0x32C91002	Critical	Not inserted in the device	Not inserted in the device	1. Check whether the extension card is correctly inserted into the device.
0x36B01000	Information	The configured module type matches the type of the actually installed module.	/	/
0x36B01001	Critical	The configured module type is inconsistent with the type of the module actually installed.	The configured module type is inconsistent with the type of the module actually installed.	1. Check the number and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0x36B01002	Critical	Not inserted in the device	Not inserted in the device	1. Check whether the extension card is correctly inserted into the device.
0x2C241000	Information	The configured module type matches the type of the actually installed module.	/	/
0x2C241001	Critical	The configured module type is inconsistent with the type of the module actually installed.	The configured module type is inconsistent with the type of the module actually installed.	1. Check the number and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0x2C241002	Critical	Not inserted in the device	Not inserted in the device	1. Check whether the extension card is correctly inserted into the device.
0xA87C1000	Information	The configured module type matches the type of the actually installed module.	/	/
0xA87C1001	Critical	The configured module type is inconsistent with the type of the module actually installed.	The configured module type is inconsistent with the type of the module actually installed.	1. Check the number and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0xA87C1002	Critical	Not inserted in the device	Not inserted in the device	1. Check whether the extension card is correctly inserted into the device.
0x00440000	Information	No fault	No fault	/
0x00440001	Information	Module cold start	Module cold start	No fault, no corrective action required
0x00440002	Information	Module hot reset	Module hot reset	/
0x00440003	Information	Module state transition (Initialization -> Pre-operational)	Module state transition (Initialization -> Pre-operational)	/
0x00440004	Information	Module state transition (Pre-operational -> Safe-operational)	Module state transition (Pre-operational -> Safe-operational)	/

Fault Code	Level	Description	Cause	Remedy
0x00440005	Information	Module state transition (Safe-operational -> Operational)	Module state transition (Safe-operational -> Operational)	/
0x00440006	Information	Module state transition (Operational -> Pre-operational)	Module state transition (Operational -> Pre-operational)	/
0x00440007	Information	Module state transition (Operational -> Safe-operational)	Module state transition (Operational -> Safe-operational)	/
0x00440008	Information	Module state transition (Operational -> Fail-safe)	Module state transition (Operational -> Fail-safe)	/
0x00440009	Information	Module state transition (Fail-safe -> Operational)	Module state transition (Fail-safe -> Operational)	/
0x00440010	Information	Module state transition (Bootstrap)	Module state transition (Bootstrap)	/
0x00440011	Information	Module upgrade event	Module upgrade event	/
0x00440012	Information	Module encryption event	Module encryption event	/
0x00440013	Information	Module calibration event	Module calibration event	/
0x00440014	Information	Module configuration event	Module configuration event	/
0x00440015	Information	The module obtains absolute time for the first time.	The module obtains absolute time for the first time.	/

Fault Code	Level	Description	Cause	Remedy
0x00441000	Warning	Protocol stack failed to scan for modules	Possible causes: 1. The expansion module does not exist. 2. No expansion module was scanned due to local bus communication failure.	1. Check whether the expansion module is properly installed and supplied with power. 2. Check the contact of the communication interface of the expansion module or restart the rack.
0x00441001	Critical	The number of I/O modules configured is more than the number of I/O modules actually scanned.	Possible causes: 1. The actual slot position of the expansion module does not match the configuration settings. 2. A local bus communication fault causes the scanned modules to differ from the configured modules.	1. Check the number and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0x00441002	Critical	The number of I/O modules configured is less than the number of I/O modules actually scanned.	Possible causes: 1. The actual slot position of the expansion module does not match the configuration settings. 2. A local bus communication fault causes the scanned modules to differ from the configured modules.	1. Check the number and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0x00441003	Critical	The type of I/O modules configured is inconsistent with that of the I/O modules actually scanned.	Possible causes: 1. The actual slot position of the expansion module does not match the configuration settings. 2. A local bus communication fault causes the scanned modules to differ from the configured modules.	1. Check the number and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0x00441004	Critical	I/O module configuration failure	The I/O module is not configured successfully.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0x00441005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.
0x00441006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0x00441007	Critical	Module type mismatch	The hardware component has been inserted, but the type of this component does not match that of the configured module.	1. Check the inserted module, and correct the configuration if necessary.

Fault Code	Level	Description	Cause	Remedy
0x00441008	Critical	New module inserted	A new hardware component has been inserted, and the type of this component matches that of the configured module.	/
0x00441009	Warning	Local bus cyclic data communication error	Local bus cyclic data frame transmission error: Data inconsistency (CRC check error, etc.). Possible cause: 1. Increased electromagnetic interference. 2. F module fault	1. Check the contact of the communication interface of the expansion module or restart the rack. 2. Eliminate or reduce the electromagnetic interference. 3. Replace the F module.
0x0044100A	Warning	Local bus cyclic data communication timeout	Local bus cyclic data frame transmission timeout. Possible causes: 1. Incorrect communication period setting. 2. Bus faults exist.	1. Modify the communication cycle. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0x0044100B	Critical	Configuration conflict	The same I/O module is configured under multiple controllers.	1. Check the configuration, resolve the conflict, and then re-download.
0x0044100C	Critical	The stopper plate is faulty or missing.	The stopper plate is faulty or missing.	1. Reinstall or replace the stopper plate.
0x0044100D	Critical	Module watchdog timeout	Possible causes: 1. Module MCU power loss. 2. Module MCU crash.	1. Check the module power supply. In case of power loss, restore the power supply. 2. Power cycle the module.

Fault Code	Level	Description	Cause	Remedy
0x0044100E	Critical	IP address conflict	IP address conflict among network devices (PLCs, couplers, etc.) in the configuration	Check the IP addresses of the network devices (PLCs, couplers, etc.) in the configuration.
0x0044100F	Critical	The I/O module does not support the currently selected industry standard.	The I/O module does not support the currently selected industry standard.	Replace or update the I/O module.
0x00441010	Critical	Configuration error: Module 1 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00441011	Critical	Configuration error: Module 2 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00441012	Critical	Configuration error: Module 3 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00441013	Critical	Configuration error: Module 4 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00441014	Critical	Configuration error: Module 5 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00441015	Critical	Configuration error: Module 6 type error	The configured module type does not match the actual module type.	1. Check the module type.

Fault Code	Level	Description	Cause	Remedy
0x00441016	Critical	Configuration error: Module 7 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00441017	Critical	Configuration error: Module 8 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00441018	Critical	Configuration error: Module 9 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00441019	Critical	Configuration error: Module 10 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x0044101A	Critical	Configuration error: Module 11 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x0044101B	Critical	Configuration error: Module 12 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x0044101C	Critical	Configuration error: Module 13 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x0044101D	Critical	Configuration error: Module 14 type error	The configured module type does not match the actual module type.	1. Check the module type.

Fault Code	Level	Description	Cause	Remedy
0x0044101E	Critical	Configuration error: Module 15 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x0044101F	Critical	Configuration error: Module 16 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00441020	Critical	Configuration error: Module 17 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00441021	Critical	Configuration error: Module 18 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00441022	Critical	Configuration error: Module 19 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00441023	Critical	Configuration error: Module 20 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00441024	Critical	Configuration error: Module 21 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00441025	Critical	Configuration error: Module 22 type error	The configured module type does not match the actual module type.	1. Check the module type.

Fault Code	Level	Description	Cause	Remedy
0x00441026	Critical	Configuration error: Module 23 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00441027	Critical	Configuration error: Module 24 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00441028	Critical	Configuration error: Module 25 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00441029	Critical	Configuration error: Module 26 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x0044102A	Critical	Configuration error: Module 27 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x0044102B	Critical	Configuration error: Module 28 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x0044102C	Critical	Configuration error: Module 29 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x0044102D	Critical	Configuration error: Module 30 type error	The configured module type does not match the actual module type.	1. Check the module type.

Fault Code	Level	Description	Cause	Remedy
0x0044102E	Critical	Configuration error: Module 31 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x0044102F	Critical	Configuration error: Module 32 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00442003	Critical	The 2SCOM-MDB/2S485-MDB module is placed in a wrong slot.	Slot position error	Place the module in the first four slots.
0x00442004	Critical	The number of submodules for serial port 0/1 of the 2SCOM-MDB/2S485-MDB module exceeds 31.	The number of submodules for serial port 0/1 cannot exceed 31.	Reduce the number of submodules for serial port 0/1.
0x00442005	Critical	The process data of the 2SCOM-MDB/2S485-MDB module exceeds 480 bytes.	The module process data (whether reading or writing) must not exceed 480 bytes.	Reduce the module process data.
0x00442010	Critical	The baud rate of the RS485 module channel is configured incorrectly.	The baud rate of the RS485 module channel is configured incorrectly.	1. Check the configuration parameters according to the instructions.
0x00442011	Critical	The communication mode of the RS485 module channel is configured incorrectly.	The communication mode of the RS485 module channel is configured incorrectly.	1. Check the configuration parameters according to the instructions.
0x00442012	Critical	The data bit-related parameters of the RS485 module channel is configured incorrectly.	The data bit-related parameters of the RS485 module channel is configured incorrectly.	1. Check the configuration parameters according to the instructions.

Fault Code	Level	Description	Cause	Remedy
0x00442013	Critical	The data packet size of the RS485 module channel is configured incorrectly.	The data packet size of the RS485 module channel is configured incorrectly.	1. Check the configuration parameters according to the instructions.
0x00442014	Critical	The message interval of the RS485 module channel is configured incorrectly.	The message interval of the RS485 module channel is configured incorrectly.	1. Check the configuration parameters according to the instructions.
0x00442015	Critical	The transmit buffer of the RS485 module channel is configured incorrectly.	The transmit buffer of the RS485 module channel is configured incorrectly.	1. Check the configuration parameters according to the instructions.
0x00442016	Critical	The receive buffer of the RS485 module channel is configured incorrectly.	The receive buffer of the RS485 module channel is configured incorrectly.	1. Check the configuration parameters according to the instructions.
0x00442020	Critical	The baud rate of the CAN module channel is configured incorrectly.	The baud rate of the CAN module channel is configured incorrectly.	1. Check the configuration parameters according to the instructions.
0x00442021	Critical	The communication method of the CAN module channel is configured incorrectly.	The communication method of the CAN module channel is configured incorrectly.	1. Check the configuration parameters according to the instructions.
0x00442022	Critical	The data bit-related parameters of the CAN module channel is configured incorrectly.	The data bit-related parameters of the CAN module channel is configured incorrectly.	1. Check the configuration parameters according to the instructions.

Fault Code	Level	Description	Cause	Remedy
0x00442023	Critical	The buffer size of the CAN module channel is configured incorrectly.	The buffer size of the CAN module channel is configured incorrectly.	1. Check the configuration parameters according to the instructions.
0x00442024	Critical	The heartbeat interval of the CAN module channel is configured incorrectly.	The heartbeat interval of the CAN module channel is configured incorrectly.	1. Check the configuration parameters according to the instructions.
0x00442025	Critical	The standard frame upper limit of the CAN module channel is configured incorrectly.	The standard frame upper limit of the CAN module channel is configured incorrectly.	1. Check the configuration parameters according to the instructions.
0x00442026	Critical	The standard frame lower limit of the CAN module channel is configured incorrectly.	The standard frame lower limit of the CAN module channel is configured incorrectly.	1. Check the configuration parameters according to the instructions.
0x00442030	Critical	The serial port parameters of the 2SCOM module channel are configured incorrectly.	The serial port parameters of the 2SCOM module channel are configured incorrectly.	1. Check the configuration parameters according to the instructions.
0x00442031	Critical	The data packet size is incorrectly configured for the channels of the 2SCOM module.	The data packet size is incorrectly configured for the channels of the 2SCOM module.	1. Check the configuration parameters according to the instructions.
0x00442032	Critical	The timeout parameter is incorrectly configured for the channels of the 2SCOM module.	The timeout parameter is incorrectly configured for the channels of the 2SCOM module.	1. Check the configuration parameters according to the instructions.
0x00442033	Critical	The port parameters of the 2SCOM module channel are configured incorrectly.	The port parameters of the 2SCOM module channel are configured incorrectly.	1. Check the configuration parameters according to the instructions.

Fault Code	Level	Description	Cause	Remedy
0x00442040	Critical	The Modbus slave address of the 2SCOM-MDB/2S485-MDB module channel is configured incorrectly.	The Modbus slave address of the 2SCOM-MDB/ 2S485-MDB module channel is configured incorrectly.	1. Check the configuration parameters according to the instructions.
0x00442041	Critical	The serial port parameters of the 2S485-MDB module channel are configured incorrectly.	The serial port parameters of the 2S485-MDB module channel are configured incorrectly.	1. Check the configuration parameters according to the instructions.
0x00442100	Critical	Preset value out of range	The preset value is larger than the maximum value or smaller than the minimum value.	1. Check the configuration parameters according to the instructions.
0x00442101	Critical	Counter configuration error	1. The instruction enabled the single-ended counting function, but the DI is not configured for counting. 2. The single-ended interface enabled by the instruction does not match the counting interface configured on the DI. 3. The counter is configured as a differential interface, but DI0 or DI1 is set to counting mode.	1. Check the configuration parameters according to the instructions.

Fault Code	Level	Description	Cause	Remedy
0x00442102	Critical	Signal type not supported	The signal type is not set to an allowed value, which includes phase A/B 1x frequency, phase A/B 2x frequency, phase A/B 4x frequency, pulse + direction, CW/ CCW, single-phase counting (A), and single-phase counting (B).	1. Check the configuration parameters according to the instructions.
0x00442103	Critical	Counter limit error	Maximum value <= Minimum value	1. Check the configuration parameters according to the instructions.
0x00442104	Critical	Incorrect setting of unit pulse increment	The unit pulse increment is less than 1.	1. Check the configuration parameters according to the instructions.
0x00442105	Critical	Incorrect setting of time base for speed measurement	The time base is not set to an allowed value. See object 800nh .3 in the object dictionary for allowed values.	1. Check the configuration parameters according to the instructions.
0x00442106	Critical	Incorrect setting of input filter parameter	The input filter parameter is not set to the allowed value. See object 800nh.2 in the object dictionary for allowed values.	1. Check the configuration parameters according to the instructions.
0x00442107	Critical	Incorrect setting of frequency sampling time	The frequency sampling time is not set to the allowed value. See object 800nh .5 in the object dictionary for allowed values.	1. Check the configuration parameters according to the instructions.

Fault Code	Level	Description	Cause	Remedy
0x00442108	Critical	Incorrect configuration of the clearing function	The clearing function is enabled through commands, but the DI is not configured with the clearing function.	1. Check the configuration parameters according to the instructions.
0x00442109	Critical	Incorrect configuration of the preset function	The preset function is enabled through commands, but the DI is not configured with the preset function.	1. Check the configuration parameters according to the instructions.
0x0044210A	Critical	Incorrect configuration of comparison 0 function	The comparison function is enabled through commands, but the DO is not configured with the comparison function.	1. Check the configuration parameters according to the instructions.
0x0044210B	Critical	Incorrect configuration of comparison 1 function	The comparison function is enabled through commands, but the DO is not configured with the comparison function.	1. Check the configuration parameters according to the instructions.
0x0044210C	Critical	Incorrect configuration of comparison 2 function	The comparison function is enabled through commands, but the DO is not configured with the comparison function.	1. Check the configuration parameters according to the instructions.

Fault Code	Level	Description	Cause	Remedy
0x0044210D	Critical	Incorrect configuration of comparison 3 function	The comparison function is enabled through commands, but the DO is not configured with the comparison function.	1. Check the configuration parameters according to the instructions.
0x0044210E	Critical	Incorrect configuration of probe 0 function	The probe function is enabled through commands, but the DI is not configured with the probe function.	1. Check the configuration parameters according to the instructions.
0x0044210F	Critical	Incorrect configuration of probe 1 function	The probe function is enabled through commands, but the DI is not configured with the probe function.	1. Check the configuration parameters according to the instructions.
0x00442110	Critical	Incorrect configuration of probe 2 function	The probe function is enabled through commands, but the DI is not configured with the probe function.	1. Check the configuration parameters according to the instructions.
0x00442111	Critical	Incorrect configuration of probe 3 function	The probe function is enabled through commands, but the DI is not configured with the probe function.	1. Check the configuration parameters according to the instructions.

Fault Code	Level	Description	Cause	Remedy
0x00442112	Critical	DO0 parameter error	The parameter is not set to the allowed value, which includes normal output and comparison output.	1. Check the configuration parameters according to the instructions.
0x00442113	Critical	DO1 parameter error	The parameter is not set to the allowed value, which includes normal output and comparison output.	1. Check the configuration parameters according to the instructions.
0x00442114	Critical	DO2 parameter error	The parameter is not set to the allowed value, which includes normal output and comparison output.	1. Check the configuration parameters according to the instructions.
0x00442115	Critical	DO3 parameter error	The parameter is not set to the allowed value, which includes normal output and comparison output.	1. Check the configuration parameters according to the instructions.
0x00442116	Critical	DI0 parameter error	The parameter is not set to the allowed value, which includes normal input, probe function, counting, clear, preset, and gating.	1. Check the configuration parameters according to the instructions.

Fault Code	Level	Description	Cause	Remedy
0x00442117	Critical	DI1 parameter error	The parameter is not set to the allowed value, which includes normal input, probe function, counting, clear, preset, and gating.	1. Check the configuration parameters according to the instructions.
0x00442118	Critical	DI2 parameter error	The parameter is not set to the allowed value, which includes normal input, probe function, counting, clear, preset, and gating.	1. Check the configuration parameters according to the instructions.
0x00442119	Critical	DI3 parameter error	The parameter is not set to the allowed value, which includes normal input, probe function, counting, clear, preset, and gating.	1. Check the configuration parameters according to the instructions.
0x00445001	Critical	Power supply overvoltage	The module power supply is unstable or has failed.	1. Check the power supply of the module.
0x00445002	Critical	Power supply undervoltage	The module power supply is unstable or has failed.	1. Check the power supply of the module.
0x00445003	Critical	External 24 V power supply failure	The external 24 V power supply fails.	1. Check the power supply of the module.
0x00445004	Critical	Module ADC parameters not configured properly	1. The ADC parameters of the module are not configured properly.	1. Check the stability of the external 24 V power voltage for the module. 2. Restart the module.

Fault Code	Level	Description	Cause	Remedy
0x00445005	Critical	External 5 V power supply failure	Unstable power supply to the module or power failure	1. Check the power supply on the user side of the module. 2. If the power is normal after check but the fault persists, replace the module.
0x00445011	Critical	Chip overheating	The chip is overheating.	1. Check the module for hardware faults. 2. Replace the module.
0x00445012	Information	Chip overtemperature fault cleared	The chip overtemperature fault is cleared.	/
0x00445013	Critical	Chip connection error	The chip is connected improperly.	1. Check the module for hardware faults. 2. Replace the module.
0x00445014	Information	Chip connection fault cleared	The chip connection fault is cleared.	/
0x00445021	Critical	Module ADC malfunction	The ADC of the module malfunctions.	1. Check the module for hardware faults. 2. Replace the module.
0x00445022	Critical	Module DAC malfunction	The DAC of the module malfunctions.	1. Check the module for hardware faults. 2. Replace the module.
0x00445023	Critical	ADC reference voltage error	The ADC reference voltage is abnormal.	1. Check the module for hardware faults. 2. Replace the module.
0x00445024	Critical	Hardware fault	Hardware fault	1. Check whether the module hardware is faulty.
0x00445025	Information	Hardware fault cleared	The hardware fault has been cleared.	/
0x00445031	Critical	Reference channel fault	The reference channel is faulty.	1. Check the cold junction channel hardware of the thermocouple (TC) module. 2. Replace the module.
0x00446001	Critical	Channel open circuit	The channel is open-circuit.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module or replace the module.

Fault Code	Level	Description	Cause	Remedy
0x00446002	Critical	Channel short circuit	The channel is short-circuited.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module or replace the module.
0x00446003	Critical	The channel data exceeds the upper limit.	The channel data exceeds the upper limit.	1. Check whether the external input signal of the input module is abnormal.
0x00446004	Critical	The channel data is below the lower limit.	The channel data is below the lower limit.	1. Check whether the external input signal of the input module is abnormal.
0x00446005	Critical	Channel data overflow	The channel data overflows.	1. Check whether the external input signal of the input module is abnormal.
0x00446006	Critical	Channel data underflow	The channel data underflows.	1. Check whether the external input signal of the input module is abnormal.
0x00446007	Critical	Channel overcurrent	1. This module requires a rated load to operate properly. However, the detected output channel current is too high, triggering overcurrent protection.	1. Check the external wiring for short circuits. 2. Check the load for overcurrent. 3. Check the module for damage.
0x00446008	Critical	Channel ADC malfunction	1. The channel ADC has failed.	1. Check the module for hardware faults. 2. Replace the module.
0x00446009	Critical	Channel data not configured properly	1. The configuration parameters sent to the module are outside the set range or do not meet the requirements.	1. Check whether the downloaded configuration parameters are valid.

Fault Code	Level	Description	Cause	Remedy
0x0044600A	Critical	TC-ISO module channel cold junction fault	1. The sampling channel of the internal cold junction in the TC-ISO module is faulty.	1. Check the cold junction channel hardware of the thermocouple (TC) module. 2. Replace the module.
0x00446010	Critical	The transmit buffer of the RS485 module channel is full.	The transmit buffer of the RS485 module channel is full.	Reduce the transmit data length or increase the transmit interval. Set the baud rate to a greater value. All channels must be checked.
0x00446011	Critical	The receive buffer of the RS485 module channel is full.	The receive buffer of the RS485 module channel is full.	Reduce the transmit data length or increase the transmit interval. All channels must be checked.
0x00446012	Critical	The receiving frame counter value of the RS485 module channel is faulty.	The receiving frame counter value of the RS485 module channel is faulty.	Check all channels for the frame counter value.
0x00446013	Critical	The receiving frame segnum value of the RS485 module channel is faulty.	The receiving frame segnum value of the RS485 module channel is faulty.	Check all channels for the frame segnum value.
0x00446014	Critical	Noise detected on RS485 module channel	Noise detected on RS485 module channel	Check all channels for interference.
0x00446015	Critical	RS485 module channel parity bit error	RS485 module channel parity bit error	Check whether the parity bit configuration parameter is correct and whether the parity bit calculation is correct. All channels must be checked.
0x00446020	Critical	The transmit buffer of the CAN module channel is full.	The transmit buffer of the CAN module channel is full.	Reduce the transmit data length or increase the transmit interval. Set the baud rate to a greater value.
0x00446021	Critical	The receive buffer of the CAN module channel is full.	The receive buffer of the CAN module channel is full.	Reduce the transmit data length or increase the transmit interval. Set the baud rate to a greater value.

Fault Code	Level	Description	Cause	Remedy
0x00446022	Critical	The receive frame counter of the CAN module is faulty.	The receive frame counter of the CAN module is faulty.	Check the frame counter value.
0x00446023	Critical	The bus load ratio of the CAN module exceeds 30%.	The bus load ratio of the CAN module exceeds 30%.	Check the data on the bus and the load rate.
0x00446024	Critical	The transmit error counter (TEC) on the CAN module exceeds 127.	The transmit error counter (TEC) on the CAN module exceeds 127.	Check the connection between the transmission line and the bus, or the bus load rate.
0x00446025	Critical	The receive error counter (REC) on the CAN module exceeds 127.	The receive error counter (REC) on the CAN module exceeds 127.	Check the connection between the receiving line and the bus, or the bus load rate
0x00446030	Critical	SSI module channel 24 V output error	An error occurs in 24 V output of the SSI module channel.	Check the output voltage.
0x00446031	Critical	SSI module channel communication error	An error occurs during channel communication of the SSI module.	Check the connection between D+ and D- terminals.
0x00446032	Critical	SSI module channel data error	The SSI module channel data is abnormal.	Check whether the baud rate is configured properly.
0x00446035	Critical	4LC_PID module channel output overload DO0	DO output short circuit or overload	1. Check the external wiring for short circuits. 2. Check the load for overcurrent. 3. Check the module for damage.
0x00446036	Critical	4LC_PID module channel output overload DO1	DO output short circuit or overload	1. Check the external wiring for short circuits. 2. Check the load for overcurrent. 3. Check the module for damage.
0x00446040	Critical	The transmit buffer of the 2SCOM module channel is full.	The transmit buffer of the 2SCOM module channel is full.	Reduce the transmit data length or increase the transmit interval. Set the baud rate to a greater value. All channels must be checked.

Fault Code	Level	Description	Cause	Remedy
0x00446041	Critical	The receive buffer of the 2SCOM module channel is full.	The receive buffer of the 2SCOM module channel is full.	Reduce the transmit data length or increase the transmit interval. All channels must be checked.
0x00446042	Critical	The receiving frame counter value of the 2SCOM module channel is faulty.	The receiving frame counter value of the 2SCOM module channel is faulty.	Check all channels for the frame counter value.
0x00446043	Critical	The receiving frame segnum value of the 2SCOM module channel is faulty.	The receiving frame segnum value of the 2SCOM module channel is faulty.	Check all channels for the frame segnum value.
0x00446044	Critical	Noise detected on 2SCOM module channel	Noise detected on 2SCOM module channel	Check all channels for interference.
0x00446045	Critical	2SCOM module channel parity bit error	2SCOM module channel parity bit error	Check whether the parity bit configuration parameter is correct and whether the parity bit calculation is correct. All channels must be checked.
0x00446046	Critical	Channel submodules of the 2SCOM-MDB module are configured incorrectly.	The added sub-module type does not match the operation mode of serial port 0/1.	Check whether the submodule matches the operation mode of serial port 0/1. All channels must be checked.
0x00446050	Critical	Short circuit	For any channel that is not outputting, the chip detects a short circuit between the DO and VDD.	1. Check the configuration parameters according to the instructions.
0x00446051	Critical	Switch ON disconnection	For any channel that is outputting, the chip detects that the DO is not connected to a load.	1. Check the configuration parameters according to the instructions.

Fault Code	Level	Description	Cause	Remedy
0x00446052	Critical	Switch OFF disconnection	For any channel that is not outputting, the chip detects that the DO is not connected to a load.	1. Check the configuration parameters according to the instructions.
0x00446053	Critical	Overcurrent	For any channel that is outputting, the chip detects that the output current exceeds the current limit of the chip (2.4 A to 3.2 A, typical value 2.8 A).	1. Check the configuration parameters according to the instructions.
0x00446054	Critical	Overload	When the output current is too large, but the overcurrent protection value is not reached, the power consumption of the chip output channel increases and the temperature increases. When the junction temperature reaches 165℃, an overload error occurs.	1. Check the configuration parameters according to the instructions.

Fault Code	Level	Description	Cause	Remedy
0x00446060	Critical	M: Modbus response parameter mismatch	Transmit/receive instructions mismatch (The parameter in the response message does not match that in the query), or the instruction is not supported by the master station (the instruction code is not supported in the master query message).	Check for communication errors. Verify if the function code is supported by the master station. All channels must be checked.
0x00446061	Critical	M: Modbus slave ID mismatch	Transmit/receive address mismatch (The slave address in response message and query message do not match.)	Check all channels for communication errors.
0x00446062	Critical	M: Modbus response address mismatch	The start address in response message does not match that in the query.	Check all channels for communication errors.
0x00446063	Critical	M: Modbus response data length mismatch	Data frame length error: The received frame length does not conform to the standard, or the received frame length is less than 5.	Check all channels for communication errors.
0x00446064	Critical	M: Modbus communication timeout	Receiving timeout	Check all channels for communication errors.
0x00446065	Critical	M: Modbus CRC check fault	CRC check error	Check all channels for communication errors.
0x00446066	Critical	Illegal function code	Illegal function code	The slave device does not support this parameter.

Fault Code	Level	Description	Cause	Remedy
0x00446067	Critical	Illegal data address	Illegal data address	The specified data address does not exist in the slave device.
0x00446068	Critical	Illegal data value	The data value is illegal.	The specified data exceeds the range or is not allowed.
0x00446069	Critical	Slave device fault	Slave device fault	An unknown error occurred during response processing on the slave device.
0x00446070	Critical	Noise detected on 2S485-MDB module channel	Noise detected on the pin	Check all channels for interference.
0x00446071	Critical	2S485-MDB module channel parity bit error	Parity bit error	Check whether the parity bit configuration parameter is correct and whether the parity bit calculation is correct. All channels must be checked.
0x00446072	Critical	Channel submodules of the 2S485-MDB module are configured incorrectly.	The added sub-module type does not match the operation mode of serial port 0/1.	Check whether the submodule matches the operation mode of serial port 0/1. All channels must be checked.
0x00446800	Critical	Phase A open circuit	Phase A open circuit	1. Check the configuration parameters according to the instructions.
0x00446801	Critical	Phase B open circuit	Phase B open circuit	1. Check the configuration parameters according to the instructions.
0x00446802	Critical	Frequency out of limit	Differential interface: frequency > 4 M; Single-ended interface: frequency > 500K	1. Check the configuration parameters according to the instructions.
0x00446803	Critical	Count value overflow	Count value >= Maximum value	1. Check the configuration parameters according to the instructions.
0x00446804	Critical	Count value underflow	Count value <= Minimum value	1. Check the configuration parameters according to the instructions.

Fault Code	Level	Description	Cause	Remedy
0x00446805	Critical	Comparison function 0 quantity/step out of range	The quantity is greater than 1024 or less than 1, or the step is less than 1.	1. Check the configuration parameters according to the instructions.
0x00446806	Critical	Comparison function 1 quantity/step out of range	The quantity is greater than 1024 or less than 1, or the step is less than 1.	1. Check the configuration parameters according to the instructions.
0x00446807	Critical	Comparison function 2 quantity/step out of range	The quantity is greater than 1024 or less than 1, or the step is less than 1.	1. Check the configuration parameters according to the instructions.
0x00446808	Critical	Comparison function 3 quantity/step out of range	The quantity is greater than 1024 or less than 1, or the step is less than 1.	1. Check the configuration parameters according to the instructions.
0x00446809	Critical	The output hold time/ pulse number is not configured properly for the comparison function 0.	The hold time is smaller than 10, or the pulse number is smaller than 1.	1. Check the configuration parameters according to the instructions.
0x0044680A	Critical	Comparison function 1 output hold time/pulse number not configured properly	The hold time is smaller than 10, or the pulse number is smaller than 1.	1. Check the configuration parameters according to the instructions.
0x0044680B	Critical	Comparison function 2 output hold time/pulse number not configured properly	The hold time is smaller than 10, or the pulse number is smaller than 1.	1. Check the configuration parameters according to the instructions.
0x0044680C	Critical	Comparison function 3 output hold time/pulse number not configured properly	The hold time is smaller than 10, or the pulse number is smaller than 1.	1. Check the configuration parameters according to the instructions.

Fault Code	Level	Description	Cause	Remedy
0x0044680D	Critical	The output start position and step direction is not configured properly for the comparison function 0.	Start and end positions are equal; start position < end position, step size < 0; start position > end position, step size > 0.	1. Check the configuration parameters according to the instructions.
0x0044680E	Critical	The output start position and step direction is not configured properly for the comparison function 1.	Start and end positions are equal; start position < end position, step size < 0; start position > end position, step size > 0.	1. Check the configuration parameters according to the instructions.
0x0044680F	Critical	The output start position and step direction is not configured properly for the comparison function 2.	Start and end positions are equal; start position < end position, step size < 0; start position > end position, step size > 0.	1. Check the configuration parameters according to the instructions.
0x00446810	Critical	The output start position and step direction is not configured properly for the comparison function 3.	Start and end positions are equal; start position < end position, step size < 0; start position > end position, step size > 0.	1. Check the configuration parameters according to the instructions.
0x00446811	Critical	The comparison value is out of range for the comparison function 0.	The comparison value is greater than the maximum value or less than the minimum value.	1. Check the configuration parameters according to the instructions.
0x00446812	Critical	Comparison function 1 comparison value out of range	The comparison value is greater than the maximum value or less than the minimum value.	1. Check the configuration parameters according to the instructions.

Fault Code	Level	Description	Cause	Remedy
0x00446813	Critical	Comparison function 2 comparison value out of range	The comparison value is greater than the maximum value or less than the minimum value.	1. Check the configuration parameters according to the instructions.
0x00446814	Critical	Comparison function 3 comparison value out of range	The comparison value is greater than the maximum value or less than the minimum value.	1. Check the configuration parameters according to the instructions.
0x00446815	Critical	The type is not configured properly for the comparison function 0.	The comparison type is not set to the allowed value, which includes multi-point comparison and uniformly-spaced comparison.	1. Check the configuration parameters according to the instructions.
0x00446816	Critical	Comparison function 1 type not configured properly	The comparison type is not set to the allowed value, which includes multi-point comparison and uniformly-spaced comparison.	1. Check the configuration parameters according to the instructions.
0x00446817	Critical	Comparison function 2 type not configured properly	The comparison type is not set to the allowed value, which includes multi-point comparison and uniformly-spaced comparison.	1. Check the configuration parameters according to the instructions.
0x00446818	Critical	Comparison function 3 type not configured properly	The comparison type is not set to the allowed value, which includes multi-point comparison and uniformly-spaced comparison.	1. Check the configuration parameters according to the instructions.

Fault Code	Level	Description	Cause	Remedy
0x00446819	Critical	The output control mode is not configured properly for the comparison function 0.	The control mode is not set to the allowed value, which includes time control, pulse control, and level control.	1. Check the configuration parameters according to the instructions.
0x0044681A	Critical	The output control mode is not configured properly for the comparison function 1.	The control mode is not set to the allowed value, which includes time control, pulse control, and level control.	1. Check the configuration parameters according to the instructions.
0x0044681B	Critical	Comparison function 2 output control mode not configured properly	The control mode is not set to the allowed value, which includes time control, pulse control, and level control.	1. Check the configuration parameters according to the instructions.
0x0044681C	Critical	Comparison function 3 output control mode not configured properly	The control mode is not set to the allowed value, which includes time control, pulse control, and level control.	1. Check the configuration parameters according to the instructions.
0x0044681D	Critical	DO0 short circuit	DO0 short circuit	1. Check the configuration parameters according to the instructions.
0x0044681E	Critical	DO1 short circuit	DO1 short circuit	1. Check the configuration parameters according to the instructions.
0x0044681F	Critical	DO2 short circuit	DO2 short circuit	1. Check the configuration parameters according to the instructions.
0x00446820	Critical	DO3 short circuit	DO3 short circuit	1. Check the configuration parameters according to the instructions.

Fault Code	Level	Description	Cause	Remedy
0x00447001	Warning	The temperature is too low.	The internal temperature of the module box is lower than the lower limit.	Operate the F module within the specified temperature range.
0x00447002	Warning	High temperature	The internal temperature of the module box is higher than the upper limit.	Operate the F module within the specified temperature range.
0x00447004	Critical	Incorrect parameter	Parameter configuration error or out-of-range diagnostics	Correct the parameter assignment.
0x00447005	Critical	The parameter set version is incorrect.	The version of the downloaded parameter set does not match the version of the parameter set running in the module.	Check the parameter assignment.
0x00447007	Critical	Module error	Possible causes: 1. Excessive electromagnetic interference occurred. 2. Module initialization error. 3. Power-on self-test error (peripheral).	1. Correct the fault. Then, remove and reinsert the module, or power cycle it. 2. If the module cannot be put back into operation, consider replacing it with a new one.
0x00447008	Critical	Module error	Possible causes: 1. Excessive electromagnetic interference occurred. 2. MCU self-diagnosis error (CPU, ROM, RAM)	1. Correct the fault. Then, remove and reinsert the module, or power cycle it. 2. If the module cannot be put back into operation, consider replacing it with a new one.

Fault Code	Level	Description	Cause	Remedy
0x00447009	Critical	Module error	Possible causes: 1. Excessive electromagnetic interference. 2. Dual MCU interaction error.	1. Correct the fault. Then, remove and reinsert the module, or power cycle it. 2. If the module cannot be put back into operation, consider replacing it with a new one.
0x0044700A	Critical	Module error	Possible causes: 1. Excessive electromagnetic interference occurred. 2. Clock diagnosis error.	1. Correct the fault. Then, remove and reinsert the module, or power cycle it. 2. If the module cannot be put back into operation, consider replacing it with a new one.
0x0044700B	Critical	Module error	Possible causes: 1. Excessive electromagnetic interference occurred. 2. Program execution sequence error.	1. Correct the fault. Then, remove and reinsert the module, or power cycle it. 2. If the module cannot be put back into operation, consider replacing it with a new one.
0x0044700C	Critical	Module error	Possible causes: 1. Excessive electromagnetic interference occurred. 2. Watchdog trip.	1. Correct the fault. Then, remove and reinsert the module, or power cycle it. 2. If the module cannot be put back into operation, consider replacing it with a new one.
0x00447011	Fatal	Potential group type mismatch	Incorrect potential group type	Replace with the correct potential group module.
0x00447012	Warning	Excessive internal temperature deviation of the module	Possible causes: 1. Excessive electromagnetic interference occurred. 2. Excessive internal temperature deviation of the module.	If the fault persists and cannot be cleared automatically, replace the module with a new one.
0x00447013	Warning	The user-side power supply voltage is too high.	Power supply voltage too high	Check the user-side power supply voltage.
0x00447014	Critical	The user-side power supply voltage is too low.	Power supply undervoltage	Check the user-side power supply voltage.

Fault Code	Level	Description	Cause	Remedy
0x00447015	Critical	User-Side power supply detection circuit fault	Possible causes: 1. Increased electromagnetic interference. 2. F module fault	1. Eliminate/reduce electromagnetic interference. 2. Replace the F module.
0x00447016	Critical	CIP Safety communication fault	CIP Safety communication fault (CRC error, mixed frame error, data frame timeout, frame sequence error, etc.) Possible causes: 1. Increased electromagnetic interference. 2. F module fault	1. Eliminate/reduce electromagnetic interference. 2. Replace the F module.
0x00447017	Critical	The module safety address is out of range.	The DIP switch address on the module is set to all zeros or all ones.	Check the DIP switch address setting on the module and power cycle it.
0x00447030	Critical	Overflow	Measurement overflow	Check the communication connection between the module and the sensor.
0x00447031	Critical	Overlimit	Measurement overlimit	Check the communication connection between the module and the sensor.
0x00447032	Critical	Underlimit	Measurement underlimit	Check the communication connection between the module and the sensor.
0x00447033	Critical	Underflow	Measurement underflow	Check the communication connection between the module and the sensor.
0x00447034	Critical	Channel overload	Channel overload	Check the actuator.
0x00447035	Critical	Channel overload or disconnection	Possible causes: 1. Channel overload. 2. Channel not connected. 3. External cable disconnection of the channel.	1. Eliminate the overload. 2. Connect the external cable. 3. Disable overload and wire break detection in the parameters.

Fault Code	Level	Description	Cause	Remedy
0x00447036	Critical	Wiring break	Possible causes: 1. External cable disconnected. 2. External circuit fault. 3. Sensor fault. 4. Incorrect sensor type set in the parameters. 5. Input channel unused. 6. Output channel not connected. 7. F module fault	1. Identify and resolve possible causes of the fault. 2. Replace the F module.
0x00447037	Critical	Interference among channels	Interference among channels	1. Eliminate any crosstalk in the process wiring. 2. Check the configured test time and the process wiring.
0x00447038	Critical	Channel-to-power supply short circuit	Possible causes: 1. Channel-to-power supply short circuit. 2. Channel readback error. 3. Short circuit between two output channels. 4. Excessive capacitive load.	1. Eliminate any crosstalk in the process wiring. 2. Check the configured test time and the process wiring. 3. Increase the test time (shutdown test).
0x00447039	Critical	Channel-to-ground short circuit	A short circuit to ground means: 1. Output cable shorted to ground. 2. Output signal shorted to ground. 3. Capacitive load too high. 4. Channel overload. 5. Channel read-back fault.	1. Correct the process wiring. 2. Increase the test time (shutdown test). 3. Check the actuator. 4. Check load conditions. 5. Replace the F module.

Fault Code	Level	Description	Cause	Remedy
0x0044703A	Critical	Differential fault, channel status 0/0 (1oo2 non-equivalent evaluation)	The difference time set for the 1oo2 evaluation has been exceeded. Possible causes: 1. Process signal error. 2. Sensor fault. 3. Configured discrepancy time too low. 4. Channel-to-ground short circuit or open circuit. 5. Fault occurred during discrepancy check.	1. Check the process signal. 2. Replace the sensor. 3. Check the parameter assignment for the error time. 4. Check the process wiring.
0x0044703B	Critical	Differential fault, channel status 0/1 (1oo2 equivalent evaluation)	The difference time set for the 1oo2 evaluation has been exceeded. Possible causes: 1. Process signal error. 2. Sensor fault. 3. Configured discrepancy time too low. 4. Channel-to-ground short circuit or open circuit. 5. Fault occurred during discrepancy check.	1. Check the process signal. 2. Replace the sensor. 3. Check the parameter assignment for the error time. 4. Check the process wiring.

Fault Code	Level	Description	Cause	Remedy
0x0044703C	Critical	Differential fault, channel status 1/0 (1oo2 equivalent evaluation)	The difference time set for the 1oo2 evaluation has been exceeded. Possible causes: 1. Process signal error. 2. Sensor fault. 3. Configured discrepancy time too low. 4. Channel-to-ground short circuit or open circuit. 5. Fault occurred during discrepancy check.	1. Check the process signal. 2. Replace the sensor. 3. Check the parameter assignment for the error time. 4. Check the process wiring.
0x0044703D	Critical	Differential fault, channel status 1/1 (1oo2 non-equivalent evaluation)	The difference time set for the 1oo2 evaluation has been exceeded. Possible causes: 1. Process signal error. 2. Sensor fault. 3. Configured discrepancy time too low. 4. Channel-to-power supply short circuit or channel-to-channel short circuit. 5. Fault occurred during discrepancy check.	1. Check the process signal. 2. Replace the sensor. 3. Check the parameter assignment for the error time. 4. Check the process wiring.

Fault Code	Level	Description	Cause	Remedy
0x0044703E	Critical	Sensor signal jitter	The number of signal changes exceeds the limit within the configured "Monitoring Window" time. Possible causes: 1. "Jitter monitoring window time" parameter set too high. 2. "Number of signal changes" parameter set too low. 3. Momentary break or short circuit (poor contact) in the sensor cable. 4. Excessive electromagnetic interference. 5. Sensor or switch switchover. 6. Sensor failure.	1. Check the "Monitoring Window" parameter. 2. Check the "Number of Signal Changes" parameter. 3. Check the process wiring. 4. Eliminate/reduce electromagnetic interference. 5. Replace the sensor.
0x0044703F	Fatal	Channel switch fault	Possible causes: 1. Increased electromagnetic interference. 2. Channel-to-power supply short circuit. 3. Switch fault inside the module. 4. Excessive capacitive load.	1. Eliminate/reduce electromagnetic interference. 2. Monitor the process signal. 3. Increase the test time (shutdown test). 4. Replace the F module.
0x00447040	Warning	Too high frequency	Maximum switching frequency of the module exceeded.	Reduce the switching frequency.

Fault Code	Level	Description	Cause	Remedy
0x00447041	Fatal	Channel comparison fault	Code value comparison error between the two MCUs. Possible causes: 1. Increased electromagnetic interference 2. F module fault	1. Eliminate/reduce electromagnetic interference. 2. Replace the F module.
0x00447042	Fatal	Analog-to-digital conversion error (ADC fault)	Internal error during analog-to-digital conversion. Possible causes: 1. Increased electromagnetic interference. 2. F module fault	Replace the F module.
0x00447043	Critical	Error time exceeded	The error time set for the 1oo2 evaluation has been exceeded. Possible causes: 1. Improper parameter assignment. 2. Fault occurred in one of the two sensors.	Check the error time, sensor, and wiring.
0x00447044	Critical	TC cold junction measured overlimit	TC cold junction measured value exceeds upper limit.	Check the configuration parameters of TC cold junction upper and lower limits, as well as the TC cold junction sensor and wiring.
0x00447045	Critical	TC cold junction measured underlimit	TC cold junction measured value exceeds lower limit.	Check the configuration parameters of TC cold junction upper and lower limits, as well as the TC cold junction sensor and wiring.

Fault Code	Level	Description	Cause	Remedy
0x00447046	Critical	TC cold junction fault	The value is outside the measurement range of the TC cold junction. Possible causes: 1. External TC cold junction fault (external cold junction mode). 2. TC backplane fault (internal cold junction mode)	1. Check the TC cold junction sensor and wiring (external cold junction mode). 2. Replace the TC module backplane (internal cold junction mode).
0x00448000	Information	The channel fault has been cleared.	/	/
0x00448001	Critical	Channel 0 overflow	Channel overflow	1. Check whether the external input signal of the input module is abnormal.
0x00448002	Critical	Channel 0 underflow	Channel underflow	1. Check whether the external input signal of the input module is abnormal.
0x00448003	Critical	Channel 0 exceeds the upper limit.	Channel above the upper limit	1. Check whether the external input signal of the input module is abnormal.
0x00448004	Critical	Channel 0 is below the lower limit.	Channel below the lower limit	1. Check whether the external input signal of the input module is abnormal.
0x00448005	Critical	Channel 0 open circuit	Channel open circuit	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0x00448006	Critical	The channel 0 is short-circuited.	The channel is short-circuited.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module or replace the module.

Fault Code	Level	Description	Cause	Remedy
0x00448007	Critical	Hardware fault in the DAC of channel 0.	The hardware in the channel DAC is faulty.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0x00448101	Critical	Channel 1 overflow	Channel overflow	1. Check whether the external input signal of the input module is abnormal.
0x00448102	Critical	Channel 1 underflow	Channel underflow	1. Check whether the external input signal of the input module is abnormal.
0x00448103	Critical	Channel 1 exceeds the upper limit.	Channel above the upper limit	1. Check whether the external input signal of the input module is abnormal.
0x00448104	Critical	Channel 1 is below the lower limit.	Channel below the lower limit	1. Check whether the external input signal of the input module is abnormal.
0x00448105	Critical	Channel 1 open circuit	Channel open circuit	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0x00448106	Critical	The channel 1 is short-circuited.	The channel is short-circuited.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module or replace the module.
0x00448107	Critical	Hardware fault in the DAC of channel 1	The hardware in the channel DAC is faulty.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0x00448201	Critical	Channel 2 overflow	Channel overflow	1. Check whether the external input signal of the input module is abnormal.
0x00448202	Critical	Channel 2 underflow	Channel underflow	1. Check whether the external input signal of the input module is abnormal.

Fault Code	Level	Description	Cause	Remedy
0x00448203	Critical	Channel 2 exceeds the upper limit.	Channel above the upper limit	1. Check whether the external input signal of the input module is abnormal.
0x00448204	Critical	Channel 2 is below the lower limit.	Channel below the lower limit	1. Check whether the external input signal of the input module is abnormal.
0x00448205	Critical	Channel 2 open circuit	Channel open circuit	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0x00448206	Critical	The channel 2 is short-circuited.	The channel is short-circuited.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module or replace the module.
0x00448207	Critical	Hardware fault in the DAC of channel 2	The hardware in the channel DAC is faulty.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0x00448301	Critical	Channel 3 overflow	Channel overflow	1. Check whether the external input signal of the input module is abnormal.
0x00448302	Critical	Channel 3 underflow	Channel underflow	1. Check whether the external input signal of the input module is abnormal.
0x00448303	Critical	Channel 3 exceeds the upper limit.	Channel above the upper limit	1. Check whether the external input signal of the input module is abnormal.
0x00448304	Critical	Channel 3 is below the lower limit.	Channel below the lower limit	1. Check whether the external input signal of the input module is abnormal.
0x00448305	Critical	Channel 3 open circuit	Channel open circuit	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.

Fault Code	Level	Description	Cause	Remedy
0x00448306	Critical	The channel 3 is short-circuited.	The channel is short-circuited.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module or replace the module.
0x00448307	Critical	Hardware fault in the DAC of channel 3.	The hardware in the channel DAC is faulty.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0x00448401	Critical	Channel 4 overflow	Channel overflow	1. Check whether the external input signal of the input module is abnormal.
0x00448402	Critical	Channel 4 underflow	Channel underflow	1. Check whether the external input signal of the input module is abnormal.
0x00448403	Critical	Channel 4 exceeds the upper limit.	Channel above the upper limit	1. Check whether the external input signal of the input module is abnormal.
0x00448404	Critical	Channel 4 is below the lower limit.	Channel below the lower limit	1. Check whether the external input signal of the input module is abnormal.
0x00448405	Critical	Channel 4 open circuit	Channel open circuit	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0x00448406	Critical	The channel 4 is short-circuited.	The channel is short-circuited.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module or replace the module.
0x00448407	Critical	Hardware fault in the DAC of channel 4.	The hardware in the channel DAC is faulty.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.

Fault Code	Level	Description	Cause	Remedy
0x00448501	Critical	Channel 5 overflow	Channel overflow	1. Check whether the external input signal of the input module is abnormal.
0x00448502	Critical	Channel 5 underflow	Channel underflow	1. Check whether the external input signal of the input module is abnormal.
0x00448503	Critical	Channel 5 exceeds the upper limit.	Channel above the upper limit	1. Check whether the external input signal of the input module is abnormal.
0x00448504	Critical	Channel 5 is below the lower limit.	Channel below the lower limit	1. Check whether the external input signal of the input module is abnormal.
0x00448505	Critical	Channel 5 open circuit	Channel open circuit	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0x00448506	Critical	The channel 5 is short-circuited.	The channel is short-circuited.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module or replace the module.
0x00448507	Critical	Hardware fault in the DAC of channel 5.	The hardware in the channel DAC is faulty.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0x00448601	Critical	Channel 6 overflow	Channel overflow	1. Check whether the external input signal of the input module is abnormal.
0x00448602	Critical	Channel 6 underflow	Channel underflow	1. Check whether the external input signal of the input module is abnormal.
0x00448603	Critical	Channel 6 exceeds the upper limit.	Channel above the upper limit	1. Check whether the external input signal of the input module is abnormal.
0x00448604	Critical	Channel 6 is below the lower limit.	Channel below the lower limit	1. Check whether the external input signal of the input module is abnormal.

Fault Code	Level	Description	Cause	Remedy
0x00448605	Critical	Channel 6 open circuit	Channel open circuit	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0x00448606	Critical	The channel 6 is short-circuited.	The channel is short-circuited.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module or replace the module.
0x00448607	Critical	Hardware fault in the DAC of channel 6.	The hardware in the channel DAC is faulty.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0x00448701	Critical	Channel 7 overflow	Channel overflow	1. Check whether the external input signal of the input module is abnormal.
0x00448702	Critical	Channel 7 underflow	Channel underflow	1. Check whether the external input signal of the input module is abnormal.
0x00448703	Critical	Channel 7 exceeds the upper limit.	Channel above the upper limit	1. Check whether the external input signal of the input module is abnormal.
0x00448704	Critical	Channel 7 is below the lower limit.	Channel below the lower limit	1. Check whether the external input signal of the input module is abnormal.
0x00448705	Critical	Channel 7 open circuit	Channel open circuit	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0x00448706	Critical	The channel 7 is short-circuited.	The channel is short-circuited.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module or replace the module.

Fault Code	Level	Description	Cause	Remedy
0x00448707	Critical	Hardware fault in the DAC of channel 7.	The hardware in the channel DAC is faulty.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module, or replace the module.
0x00448A00	Information	Module power loss fault cleared	The module power loss fault has been cleared.	/
0x00448A01	Information	Chip overtemperature fault cleared	The chip overtemperature fault is cleared.	/
0x00448A02	Critical	Chip connection error	The chip is connected improperly.	1. Check the module for hardware faults. 2. Replace the module.
0x00448A03	Information	Chip connection fault cleared	The chip connection fault is cleared.	/
0x00448A04	Critical	Hardware fault	Hardware fault	1. Check whether the module hardware is faulty.
0x00448A05	Information	Hardware fault cleared	The hardware fault has been cleared.	/
0x00448B00	Critical	Configuration error: Module 1 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B01	Critical	Configuration error: Module 2 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B02	Critical	Configuration error: Module 3 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B03	Critical	Configuration error: Module 4 type error	The configured module type does not match the actual module type.	1. Check the module type.

Fault Code	Level	Description	Cause	Remedy
0x00448B04	Critical	Configuration error: Module 5 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B05	Critical	Configuration error: Module 6 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B06	Critical	Configuration error: Module 7 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B07	Critical	Configuration error: Module 8 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B08	Critical	Configuration error: Module 9 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B09	Critical	Configuration error: Module 10 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B0A	Critical	Configuration error: Module 11 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B0B	Critical	Configuration error: Module 12 type error	The configured module type does not match the actual module type.	1. Check the module type.

Fault Code	Level	Description	Cause	Remedy
0x00448B0C	Critical	Configuration error: Module 13 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B0D	Critical	Configuration error: Module 14 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B0E	Critical	Configuration error: Module 15 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B0F	Critical	Configuration error: Module 16 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B10	Critical	Configuration error: Module 17 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B11	Critical	Configuration error: Module 18 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B12	Critical	Configuration error: Module 19 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B13	Critical	Configuration error: Module 20 type error	The configured module type does not match the actual module type.	1. Check the module type.

Fault Code	Level	Description	Cause	Remedy
0x00448B14	Critical	Configuration error: Module 21 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B15	Critical	Configuration error: Module 22 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B16	Critical	Configuration error: Module 23 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B17	Critical	Configuration error: Module 24 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B18	Critical	Configuration error: Module 25 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B19	Critical	Configuration error: Module 26 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B1A	Critical	Configuration error: Module 27 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B1B	Critical	Configuration error: Module 28 type error	The configured module type does not match the actual module type.	1. Check the module type.

Fault Code	Level	Description	Cause	Remedy
0x00448B1C	Critical	Configuration error: Module 29 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B1D	Critical	Configuration error: Module 30 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B1E	Critical	Configuration error: Module 31 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00448B1F	Critical	Configuration error: Module 32 type error	The configured module type does not match the actual module type.	1. Check the module type.
0x00449680	Critical	The system power supply voltage exceeds the limit.	The system power supply voltage exceeds the limit.	1. Check the system power supply.
0x00449681	Critical	The system power supply voltage is below the limit.	The system power supply voltage is below the limit.	1. Check the system power supply.
0x00449682	Critical	The actuator power supply voltage exceeds the limit.	The actuator power supply voltage exceeds the limit.	1. Check the actuator power supply.
0x00449683	Critical	The actuator power supply voltage is below the limit.	The actuator power supply voltage is below the limit.	1. Check the actuator power supply.
0x00449684	Critical	MCU temperature too high	The MCU temperature is too high.	1. Check the equipment temperature.
0x00449685	Information	System power supply voltage over-limit fault cleared	The fault "The system power supply voltage exceeds the limit" has been cleared.	1. Fault cleared

Fault Code	Level	Description	Cause	Remedy
0x00449686	Information	System power supply voltage under-limit fault cleared	The fault "The system power supply voltage is below the limit" has been cleared.	1. Fault cleared
0x00449687	Information	Actuator power supply voltage over-limit fault cleared	The fault "The actuator power supply voltage exceeds the limit" has been cleared.	1. Fault cleared
0x00449688	Information	Actuator power supply voltage under-limit fault cleared	The fault "The actuator power supply voltage is below the limit" has been cleared.	1. Fault cleared
0x00449689	Information	MCU overtemperature fault cleared	The MCU overtemperature fault has been cleared.	1. Fault cleared
0x00449000	Critical	Port communication disconnection	Port communication disconnection	1. Check the device port.
0x00449001	Information	Configuration match successful	The fault "The number of I/O modules configured exceeds the number of I/O modules actually scanned" has been cleared.	/
0x00449002	Information	Configuration match successful	The fault "The number of I/O modules configured is less than the number of I/O modules actually scanned" has been cleared.	/

Fault Code	Level	Description	Cause	Remedy
0x00449003	Information	Configuration match successful	The fault "The type of I/O modules configured is inconsistent with that of the I/O modules actually scanned" has been cleared.	/
0x00449004	Information	Module configuration successful	/	/
0x00449005	Information	The module status is switched successfully.	/	/
0x00449009	Information	Cyclic data communication fault cleared	/	/
0x0044900A	Information	Cyclic data communication timeout fault cleared	/	/
0x0044900B	Information	Configuration conflict fault cleared	/	/
0x0044900C	Information	Stopper plate fault cleared	/	/
0x0044900D	Information	Module watchdog timeout cleared	/	/
0x0044900E	Information	IP address conflict cleared	/	/
0x0044900F	Information	The fault "The I/O module does not support the currently selected industry standard" has been cleared.	/	/
0x00449010	Information	Acyclic data communication fault cleared	/	/
0x00449011	Information	Acyclic data communication timeout fault cleared	/	/
0x00449020	Critical	Port Pin2 short circuit	Port Pin2 is short-circuited.	1. Check the wiring.
0x00449040	Critical	Port connection to no device	Port connection to no device	1. Check the device port.

Fault Code	Level	Description	Cause	Remedy
0x00449060	Critical	Port startup parameterization error	Port startup parameterization error	1. Check the parameters.
0x00449080	Critical	Port vendor ID incorrect	The port vendor ID is incorrect.	1. Check if the device is correct.
0x004490A0	Critical	Port device ID incorrect	The port device ID is incorrect.	1. Check if the device is correct.
0x004490C0	Critical	Port Pin4 short circuit	Port Pin4 is short-circuited.	1. Check the wiring.
0x004490E0	Critical	Overtemperature on the port PHY chip.	Overtemperature on the port PHY chip.	1. Check the host temperature and load.
0x00449100	Critical	Port Pin1 short circuit	Port Pin1 is short-circuited.	1. Check the wiring.
0x00449120	Critical	Port Pin1 overcurrent	Overcurrent occurs on port Pin 1.	1. Check the power supply.
0x00449140	Critical	Slave device event overflow on the port	A slave device event overflow occurs on the port.	1. Check the device.
0x00449160	Critical	Port Pin4 short circuit (DO mode)	Port Pin4 is short-circuited. (DO mode)	1. Check the wiring.
0x00449180	Critical	Port Pin4 overcurrent (DO mode)	Overcurrent occurs on port Pin 4. (DO mode)	1. Check the load.
0x004491A0	Warning	Port temperature out of range	The port temperature exceeds the limit.	1. Remove heating sources.
0x004491C0	Warning	Port temperature too low	The port temperature is too low.	1. Thermally insulate the device.
0x004491E0	Critical	Device function fault on the port	The device function on the port is faulty.	1. Replace the device.
0x00449200	Critical	Retentive data loss upon port power failure	The retentive data upon port power failure is lost.	1. Check the battery.

Fault Code	Level	Description	Cause	Remedy
0x00449220	Critical	Port device power supply fault	Power supply error for the device connected to the port.	1. Check the power availability.
0x00449240	Warning	Port system power supply voltage out of range	The system power supply voltage for the port is out of range.	1. Check the tolerance.
0x00449260	Warning	Port system power supply undervoltage	The system power supply voltage for the port is below the limit.	1. Check the tolerance.
0x00449280	Critical	Device function fault on the port	The device function on the port is faulty.	1. Check the device.
0x004492A0	Critical	Invalid cycle time for the port	The cycle time for the port is invalid.	1. Check the settings.
0x004492C0	Critical	Port version error	Port version error	1. Check the version.
0x004492E0	Critical	Port parameter error	The port parameter is faulty.	1. Check the data sheet.
0x00449300	Warning	Port ground fault	A ground fault has occurred on the port.	1. Check the wiring.
0x00449320	Warning	The port process variable range exceeds the limits.	The port process variable range exceeds the limits.	1. Check the application.
0x00449340	Information	Port communication disconnection cleared.	The fault "port communication disconnection" has been resolved.	1. Fault cleared
0x00449360	Information	Port Pin2 short circuit cleared	The short circuit on port Pin2 has been cleared.	1. Fault cleared
0x00449380	Information	Port device missing fault cleared.	The fault "no device detected on the port" has been cleared.	1. Fault cleared
0x004493A0	Information	Port startup parameterization error cleared	The port startup parameterization error has been cleared.	1. Fault cleared

Fault Code	Level	Description	Cause	Remedy
0x004493C0	Information	Invalid port supplier ID cleared	The invalid port vendor ID fault has been cleared.	1. Fault cleared
0x004493E0	Information	Invalid port device ID cleared	The invalid port device ID fault has been cleared.	1. Fault cleared
0x00449400	Information	Port Pin4 short circuit cleared	The short circuit on port Pin4 has been cleared.	1. Fault cleared
0x00449420	Information	Port PHY chip overtemperature cleared	The port PHY chip overtemperature has been cleared.	1. Fault cleared
0x00449440	Information	Port Pin1 short circuit cleared	The short circuit on port Pin1 has been cleared.	1. Fault cleared
0x00449460	Information	Port Pin1 overcurrent cleared	The overcurrent fault on port Pin1 has been cleared.	1. Fault cleared
0x00449480	Information	Port slave device event overflow cleared	The slave device event overflow on the port has been cleared.	1. Fault cleared
0x004494A0	Information	Port Pin4 short circuit cleared (DO mode)	The short circuit on port Pin4 has been cleared. (DO mode)	1. Fault cleared
0x004494C0	Information	Port Pin4 overcurrent cleared (DO mode)	The overcurrent fault on port Pin4 has been cleared (DO mode).	1. Fault cleared
0x004494E0	Information	Port temperature out-of-range fault cleared	The port temperature out-of-range fault has been cleared.	1. Fault cleared
0x00449500	Information	Port temperature too low cleared	The port temperature too low fault has been cleared.	1. Fault cleared
0x00449520	Information	Port device function fault cleared	The port device function fault has been cleared.	1. Fault cleared

Fault Code	Level	Description	Cause	Remedy
0x00449540	Information	The retentive data loss due to port power failure has been cleared.	The retentive data loss due to port power failure has been cleared.	1. Fault cleared
0x00449560	Information	Port device power supply error cleared	The port device power supply error has been cleared	1. Fault cleared
0x00449580	Information	Port system power supply voltage out-of-range cleared	The fault "The system power supply voltage for the port exceeds the limit" has been cleared.	1. Fault cleared
0x004495A0	Information	Port system power supply undervoltage cleared	The fault "The system power supply voltage for the port is below the limit" has been cleared.	1. Fault cleared
0x004495C0	Information	Port device function fault cleared	The port device function fault has been cleared.	1. Fault cleared
0x004495E0	Information	The invalid cycle time for the port has been cleared.	The invalid cycle time for the port has been cleared.	1. Fault cleared
0x00449600	Information	Port version error cleared	The port version error has been cleared.	1. Fault cleared
0x00449620	Information	Port parameter error cleared	The port parameter error has been cleared.	1. Fault cleared
0x00449640	Information	Port ground fault cleared	The port ground fault has been cleared.	1. Fault cleared
0x00449660	Information	Port process variable out-of-range cleared	The port process variable out-of-range fault has been cleared.	1. Fault cleared
0x004496A0	Critical	Port device short circuit	Port device short circuit	1. Check the wiring.

Fault Code	Level	Description	Cause	Remedy
0x004496C0	Information	Port device short circuit cleared	The short circuit on the port device has been cleared.	1. Fault cleared
0x004496E0	Critical	Port disconnection	Port disconnection	1. Check if the input connection is correct.
0x00449720	Warning	Port input out of range	The port input is out of range.	1. The input exceeds normal measuring range. Check the input.
0x00449740	Warning	Port input underflow	Port input underflow	1. The input is below the measuring range. Increase the input.
0x00449760	Critical	Missing parameter	Missing parameter	1. In expert mode, if an input signal is detected on a channel that is configured as an output, modify the I/O configuration parameter 0x41 to match the actual usage.
0x00449780	Critical	PIN1 short circuit	PIN1 short circuit	1. A short circuit occurs on the module PIN1. Check the external wiring.
0x004497A0	Critical	No cascaded devices connected	No cascaded device is connected.	1. Check if any cascaded devices are connected.
0x004497C0	Critical	Cascaded device verification failed	Cascaded device verification failed	1. Check if a compatible cascaded device is connected.
0x004497E0	Warning	Cascaded device temperature out of range	The cascaded device temperature is out of range.	1. The ambient temperature of the cascaded device is too high. Lower the temperature before using the device.
0x00449800	Warning	Cascaded device temperature too low	The cascaded device temperature is too low.	1. The ambient temperature of the cascaded device is too low. Increase the temperature before using the device.
0x00449820	Critical	Retentive data loss due to cascaded device power failure	The retentive data upon cascaded device power failure is lost.	1. Failed to save configuration parameters for the cascaded device. Re-configure the parameters.
0x00449840	Critical	Cascaded device power supply error	Power supply error on the cascaded device.	1. The UA power supply for the cascaded device module is not provided. Enable the master PIN2 power supply or connect an external UA power source.

Fault Code	Level	Description	Cause	Remedy
0x00449860	Warning	Cascaded device system power supply voltage out of range	The system power supply voltage for the cascaded device exceeds the limit.	1. The US power supply voltage of the cascaded device module is higher than the normal operating voltage. Reduce the US power supply voltage.
0x00449880	Warning	Cascaded device system power supply undervoltage	The system power supply voltage for the cascaded device is below the limit.	1. The US power supply voltage of the cascaded device module is lower than the normal operating voltage. Increase the US power supply voltage.
0x004498A0	Critical	Cascaded device parameter error	The parameter in the cascaded device is faulty.	1. The parameters for the cascaded device module have been configured incorrectly. Configure the parameters correctly according to the product data.
0x004498C0	Critical	Missing parameters in the cascaded device	The parameters in the cascaded device are missing.	1. In expert mode of the cascaded device, if an input signal is detected on a channel that is configured as an output, modify the I/O configuration parameter 0x41 to match the actual usage.
0x004498E0	Critical	Cascaded device DO short circuit	The DO on the cascaded device is short-circuited.	1. There is a short circuit on the output of the cascaded device module. Check the external wiring.
0x00449900	Critical	Cascaded device PIN1 short circuit	PIN1 on the cascaded device is short-circuited.	1. There is a short circuit on PIN1 of the cascaded device module. Check the external wiring.
0x00449920	Information	Port disconnection cleared	The port disconnection fault has been cleared.	1. Fault cleared
0x00449960	Information	Port input out-of-range fault cleared	The port input out-of-range fault has been cleared.	1. Fault cleared
0x00449980	Information	Port input underflow fault cleared	The port input underflow fault has been cleared.	1. Fault cleared

Fault Code	Level	Description	Cause	Remedy
0x004499A0	Information	Missing parameters cleared	The missing parameter fault has been cleared.	1. Fault cleared
0x004499C0	Information	PIN1 short circuit cleared	The PIN1 short circuit has been cleared.	1. Fault cleared
0x004499E0	Information	No cascaded devices connected fault cleared	The fault "No cascaded devices connected" has been cleared.	1. Fault cleared
0x00449A00	Information	Cascaded device verification failure cleared	The cascaded device verification failure has been cleared.	1. Fault cleared
0x00449A20	Information	Cascaded device temperature out-of-range fault cleared	The cascaded device temperature out-of-range fault has been cleared.	1. Fault cleared
0x00449A40	Information	Cascaded device temperature too low cleared	The fault "Cascaded device temperature too low" has been cleared.	1. Fault cleared
0x00449A60	Information	The retentive data loss due to cascaded device power failure has been cleared.	The retentive data loss due to cascaded device power failure has been cleared.	1. Fault cleared
0x00449A80	Information	Cascaded device power supply error cleared	The cascaded device power supply error has been cleared.	1. Fault cleared
0x00449AA0	Information	Cascaded device system power supply voltage out-of-range fault cleared	The fault "The system power supply voltage for the cascaded device exceeds the limit" has been cleared.	1. Fault cleared

Fault Code	Level	Description	Cause	Remedy
0x00449AC0	Information	Cascaded device system power supply undervoltage cleared	The fault "The system power supply voltage for the cascaded device is below the limit" has been cleared.	1. Fault cleared
0x00449AE0	Information	Cascaded device parameter error cleared	The cascaded device parameter error has been cleared.	1. Fault cleared
0x00449B00	Information	Missing parameters in the cascaded device cleared	The missing parameters in the cascaded device have been cleared.	1. Fault cleared
0x00449B20	Information	Cascaded device DO short circuit cleared	The DO short circuit on the cascaded device has been cleared.	1. Fault cleared
0x00449B40	Information	Cascaded device PIN1 short circuit cleared	The PIN1 short circuit on the cascaded device has been cleared.	1. Fault cleared
0x0044A000	Information	DNM normal operating status	DNM normal operating status	/
0x0044A002	Information	DNM slave initialization	DNM slave initialization	/
0x0044A004	Information	DNM master initialization	DNM master initialization	/
0x0044A005	Critical	DNM duplicate name detection	DNM duplicate name detection	Check the DNM configurations.
0x0044A006	Critical	DNM fault status	DNM fault status	Check the DNM configurations.
0x0044A800	Information	The DNM slave device is operating normally.	The DNM slave device is operating normally.	/
0x0044A801	Critical	The DNM slave device is offline.	The DNM slave device is offline.	/
0x0044A802	Critical	DNM slave device connection closed	The DNM slave device connection is closed.	/

Fault Code	Level	Description	Cause	Remedy
0x0044A803	Critical	Duplicate DNM slave device ID exists.	Duplicate DNM slave device IDs exist.	Check the DNM configurations.
0x0044A804	Critical	Receive data length configuration error	The receive data length is configured incorrectly.	Check the DNM configurations.
0x0044A808	Critical	Transmit data length configuration error	The transmit data length is configured incorrectly.	Check the DNM configurations.
0x0044A810	Critical	Vendor ID configuration fault	Vendor ID configuration fault	Check the DNM configurations.
0x0044A820	Critical	Device type configuration error	The device type is configured incorrectly.	Check the DNM configurations.
0x0044A840	Critical	Product code configuration error	The product code is configured incorrectly.	Check the DNM configurations.
0x0044A880	Critical	Connection rate configuration error	The connection rate is configured incorrectly.	Check the DNM configurations.
0x0044A100	Information	1. Normal 2. If a slave ID is configured as 0, the fault station number of the slave is 0.	1. Normal 2. If a slave ID is configured as 0, the fault station number of the slave is 0.	Check the DNM configurations.
0x0044A101	Critical	The fault station number of the DNM slave is 1.	The fault station number of the DNM slave is 1.	Check the DNM configurations.
0x0044A102	Critical	The fault station number of the DNM slave is 2.	The fault station number of the DNM slave is 2.	Check the DNM configurations.
0x0044A103	Critical	The fault station number of the DNM slave is 3.	The fault station number of the DNM slave is 3.	Check the DNM configurations.
0x0044A104	Critical	The fault station number of the DNM slave is 4.	The fault station number of the DNM slave is 4.	Check the DNM configurations.
0x0044A105	Critical	The fault station number of the DNM slave is 5.	The fault station number of the DNM slave is 5.	Check the DNM configurations.

Fault Code	Level	Description	Cause	Remedy
0x0044A106	Critical	The fault station number of the DNM slave is 6.	The fault station number of the DNM slave is 6.	Check the DNM configurations.
0x0044A107	Critical	The fault station number of the DNM slave is 7.	The fault station number of the DNM slave is 7.	Check the DNM configurations.
0x0044A108	Critical	The fault station number of the DNM slave is 8.	The fault station number of the DNM slave is 8.	Check the DNM configurations.
0x0044A109	Critical	The fault station number of the DNM slave is 9.	The fault station number of the DNM slave is 9.	Check the DNM configurations.
0x0044A10A	Critical	The fault station number of the DNM slave is 10.	The fault station number of the DNM slave is 10.	Check the DNM configurations.
0x0044A10B	Critical	The fault station number of the DNM slave is 11.	The fault station number of the DNM slave is 11.	Check the DNM configurations.
0x0044A10C	Critical	The fault station number of the DNM slave is 12.	The fault station number of the DNM slave is 12.	Check the DNM configurations.
0x0044A10D	Critical	The fault station number of the DNM slave is 13.	The fault station number of the DNM slave is 13.	Check the DNM configurations.
0x0044A10E	Critical	The fault station number of the DNM slave is 14.	The fault station number of the DNM slave is 14.	Check the DNM configurations.
0x0044A10F	Critical	The fault station number of the DNM slave is 15.	The fault station number of the DNM slave is 15.	Check the DNM configurations.
0x0044A110	Critical	The fault station number of the DNM slave is 16.	The fault station number of the DNM slave is 16.	Check the DNM configurations.
0x0044A111	Critical	The fault station number of the DNM slave is 17.	The fault station number of the DNM slave is 17.	Check the DNM configurations.
0x0044A112	Critical	The fault station number of the DNM slave is 18.	The fault station number of the DNM slave is 18.	Check the DNM configurations.
0x0044A113	Critical	The fault station number of the DNM slave is 19.	The fault station number of the DNM slave is 19.	Check the DNM configurations.

Fault Code	Level	Description	Cause	Remedy
0x0044A114	Critical	The fault station number of the DNM slave is 20.	The fault station number of the DNM slave is 20.	Check the DNM configurations.
0x0044A115	Critical	The fault station number of the DNM slave is 21.	The fault station number of the DNM slave is 21.	Check the DNM configurations.
0x0044A116	Critical	The fault station number of the DNM slave is 22.	The fault station number of the DNM slave is 22.	Check the DNM configurations.
0x0044A117	Critical	The fault station number of the DNM slave is 23.	The fault station number of the DNM slave is 23.	Check the DNM configurations.
0x0044A118	Critical	The fault station number of the DNM slave is 24.	The fault station number of the DNM slave is 24.	Check the DNM configurations.
0x0044A119	Critical	The fault station number of the DNM slave is 25.	The fault station number of the DNM slave is 25.	Check the DNM configurations.
0x0044A11A	Critical	The fault station number of the DNM slave is 26.	The fault station number of the DNM slave is 26.	Check the DNM configurations.
0x0044A11B	Critical	The fault station number of the DNM slave is 27.	The fault station number of the DNM slave is 27.	Check the DNM configurations.
0x0044A11C	Critical	The fault station number of the DNM slave is 28.	The fault station number of the DNM slave is 28.	Check the DNM configurations.
0x0044A11D	Critical	The fault station number of the DNM slave is 29.	The fault station number of the DNM slave is 29.	Check the DNM configurations.
0x0044A11E	Critical	The fault station number of the DNM slave is 30.	The fault station number of the DNM slave is 30.	Check the DNM configurations.
0x0044A11F	Critical	The fault station number of the DNM slave is 31.	The fault station number of the DNM slave is 31.	Check the DNM configurations.
0x0044A120	Critical	The fault station number of the DNM slave is 32.	The fault station number of the DNM slave is 32.	Check the DNM configurations.
0x0044A121	Critical	The fault station number of the DNM slave is 33.	The fault station number of the DNM slave is 33.	Check the DNM configurations.

Fault Code	Level	Description	Cause	Remedy
0x0044A122	Critical	The fault station number of the DNM slave is 34.	The fault station number of the DNM slave is 34.	Check the DNM configurations.
0x0044A123	Critical	The fault station number of the DNM slave is 35.	The fault station number of the DNM slave is 35.	Check the DNM configurations.
0x0044A124	Critical	The fault station number of the DNM slave is 36.	The fault station number of the DNM slave is 36.	Check the DNM configurations.
0x0044A125	Critical	The fault station number of the DNM slave is 37.	The fault station number of the DNM slave is 37.	Check the DNM configurations.
0x0044A126	Critical	The fault station number of the DNM slave is 38.	The fault station number of the DNM slave is 38.	Check the DNM configurations.
0x0044A127	Critical	The fault station number of the DNM slave is 39.	The fault station number of the DNM slave is 39.	Check the DNM configurations.
0x0044A128	Critical	The fault station number of the DNM slave is 40.	The fault station number of the DNM slave is 40.	Check the DNM configurations.
0x0044A129	Critical	The fault station number of the DNM slave is 41.	The fault station number of the DNM slave is 41.	Check the DNM configurations.
0x0044A12A	Critical	The fault station number of the DNM slave is 42.	The fault station number of the DNM slave is 42.	Check the DNM configurations.
0x0044A12B	Critical	The fault station number of the DNM slave is 43.	The fault station number of the DNM slave is 43.	Check the DNM configurations.
0x0044A12C	Critical	The fault station number of the DNM slave is 44.	The fault station number of the DNM slave is 44.	Check the DNM configurations.
0x0044A12D	Critical	The fault station number of the DNM slave is 45.	The fault station number of the DNM slave is 45.	Check the DNM configurations.
0x0044A12E	Critical	The fault station number of the DNM slave is 46.	The fault station number of the DNM slave is 46.	Check the DNM configurations.
0x0044A12F	Critical	The fault station number of the DNM slave is 47.	The fault station number of the DNM slave is 47.	Check the DNM configurations.

Fault Code	Level	Description	Cause	Remedy
0x0044A130	Critical	The fault station number of the DNM slave is 48.	The fault station number of the DNM slave is 48.	Check the DNM configurations.
0x0044A131	Critical	The fault station number of the DNM slave is 49.	The fault station number of the DNM slave is 49.	Check the DNM configurations.
0x0044A132	Critical	The fault station number of the DNM slave is 50.	The fault station number of the DNM slave is 50.	Check the DNM configurations.
0x0044A133	Critical	The fault station number of the DNM slave is 51.	The fault station number of the DNM slave is 51.	Check the DNM configurations.
0x0044A134	Critical	The fault station number of the DNM slave is 52.	The fault station number of the DNM slave is 52.	Check the DNM configurations.
0x0044A135	Critical	The fault station number of the DNM slave is 53.	The fault station number of the DNM slave is 53.	Check the DNM configurations.
0x0044A136	Critical	The fault station number of the DNM slave is 54.	The fault station number of the DNM slave is 54.	Check the DNM configurations.
0x0044A137	Critical	The fault station number of the DNM slave is 55.	The fault station number of the DNM slave is 55.	Check the DNM configurations.
0x0044A138	Critical	The fault station number of the DNM slave is 56.	The fault station number of the DNM slave is 56.	Check the DNM configurations.
0x0044A139	Critical	The fault station number of the DNM slave is 57.	The fault station number of the DNM slave is 57.	Check the DNM configurations.
0x0044A13A	Critical	The fault station number of the DNM slave is 58.	The fault station number of the DNM slave is 58.	Check the DNM configurations.
0x0044A13B	Critical	The fault station number of the DNM slave is 59.	The fault station number of the DNM slave is 59.	Check the DNM configurations.
0x0044A13C	Critical	The fault station number of the DNM slave is 60.	The fault station number of the DNM slave is 60.	Check the DNM configurations.
0x0044A13D	Critical	The fault station number of the DNM slave is 61.	The fault station number of the DNM slave is 61.	Check the DNM configurations.

Fault Code	Level	Description	Cause	Remedy
0x0044A13E	Critical	The fault station number of the DNM slave is 62.	The fault station number of the DNM slave is 62.	Check the DNM configurations.
0x0044A13F	Critical	The fault station number of the DNM slave is 63.	The fault station number of the DNM slave is 63.	Check the DNM configurations.
0x0044A150	Critical	The DNM slave is offline at position 0.	The DNM slave is offline at position 0.	Check the DNM configurations.
0x0044A151	Critical	The DNM slave is offline at position 1.	The DNM slave is offline at position 1.	Check the DNM configurations.
0x0044A152	Critical	The DNM slave is offline at position 2.	The DNM slave is offline at position 2.	Check the DNM configurations.
0x0044A153	Critical	The DNM slave is offline at position 3.	The DNM slave is offline at position 3.	Check the DNM configurations.
0x0044A154	Critical	The DNM slave is offline at position 4.	The DNM slave is offline at position 4.	Check the DNM configurations.
0x0044A155	Critical	The DNM slave is offline at position 5.	The DNM slave is offline at position 5.	Check the DNM configurations.
0x0044A156	Critical	The DNM slave is offline at position 6.	The DNM slave is offline at position 6.	Check the DNM configurations.
0x0044A157	Critical	The DNM slave is offline at position 7.	The DNM slave is offline at position 7.	Check the DNM configurations.
0x0044A158	Critical	The DNM slave is offline at position 8.	The DNM slave is offline at position 8.	Check the DNM configurations.
0x0044A159	Critical	The DNM slave is offline at position 9.	The DNM slave is offline at position 9.	Check the DNM configurations.
0x0044A15A	Critical	The DNM slave is offline at position 10.	The DNM slave is offline at position 10.	Check the DNM configurations.
0x0044A15B	Critical	The DNM slave is offline at position 11.	The DNM slave is offline at position 11.	Check the DNM configurations.

Fault Code	Level	Description	Cause	Remedy
0x0044A15C	Critical	The DNM slave is offline at position 12.	The DNM slave is offline at position 12.	Check the DNM configurations.
0x0044A15D	Critical	The DNM slave is offline at position 13.	The DNM slave is offline at position 13.	Check the DNM configurations.
0x0044A15E	Critical	The DNM slave is offline at position 14.	The DNM slave is offline at position 14.	Check the DNM configurations.
0x0044A15F	Critical	The DNM slave is offline at position 15.	The DNM slave is offline at position 15.	Check the DNM configurations.
0x0044A160	Critical	The DNM slave is offline at position 16.	The DNM slave is offline at position 16.	Check the DNM configurations.
0x0044A161	Critical	The DNM slave is offline at position 17.	The DNM slave is offline at position 17.	Check the DNM configurations.
0x0044A162	Critical	The DNM slave is offline at position 18.	The DNM slave is offline at position 18.	Check the DNM configurations.
0x0044A163	Critical	The DNM slave is offline at position 19.	The DNM slave is offline at position 19.	Check the DNM configurations.
0x0044A164	Critical	The DNM slave is offline at position 20.	The DNM slave is offline at position 20.	Check the DNM configurations.
0x0044A165	Critical	The DNM slave is offline at position 21.	The DNM slave is offline at position 21.	Check the DNM configurations.
0x0044A166	Critical	The DNM slave is offline at position 22.	The DNM slave is offline at position 22.	Check the DNM configurations.
0x0044A167	Critical	The DNM slave is offline at position 23.	The DNM slave is offline at position 23.	Check the DNM configurations.
0x0044A168	Critical	The DNM slave is offline at position 24.	The DNM slave is offline at position 24.	Check the DNM configurations.
0x0044A169	Critical	The DNM slave is offline at position 25.	The DNM slave is offline at position 25.	Check the DNM configurations.

Fault Code	Level	Description	Cause	Remedy
0x0044A16A	Critical	The DNM slave is offline at position 26.	The DNM slave is offline at position 26.	Check the DNM configurations.
0x0044A16B	Critical	The DNM slave is offline at position 27.	The DNM slave is offline at position 27.	Check the DNM configurations.
0x0044A16C	Critical	The DNM slave is offline at position 28.	The DNM slave is offline at position 28.	Check the DNM configurations.
0x0044A16D	Critical	The DNM slave is offline at position 29.	The DNM slave is offline at position 29.	Check the DNM configurations.
0x0044A16E	Critical	The DNM slave is offline at position 30.	The DNM slave is offline at position 30.	Check the DNM configurations.
0x0044A16F	Critical	The DNM slave is offline at position 31.	The DNM slave is offline at position 31.	Check the DNM configurations.
0x0044A170	Critical	The DNM slave is offline at position 32.	The DNM slave is offline at position 32.	Check the DNM configurations.
0x0044A171	Critical	The DNM slave is offline at position 33.	The DNM slave is offline at position 33.	Check the DNM configurations.
0x0044A172	Critical	The DNM slave is offline at position 34.	The DNM slave is offline at position 34.	Check the DNM configurations.
0x0044A173	Critical	The DNM slave is offline at position 35.	The DNM slave is offline at position 35.	Check the DNM configurations.
0x0044A174	Critical	The DNM slave is offline at position 36.	The DNM slave is offline at position 36.	Check the DNM configurations.
0x0044A175	Critical	The DNM slave is offline at position 37.	The DNM slave is offline at position 37.	Check the DNM configurations.
0x0044A176	Critical	The DNM slave is offline at position 38.	The DNM slave is offline at position 38.	Check the DNM configurations.
0x0044A177	Critical	The DNM slave is offline at position 39.	The DNM slave is offline at position 39.	Check the DNM configurations.

Fault Code	Level	Description	Cause	Remedy
0x0044A178	Critical	The DNM slave is offline at position 40.	The DNM slave is offline at position 40.	Check the DNM configurations.
0x0044A179	Critical	The DNM slave is offline at position 41.	The DNM slave is offline at position 41.	Check the DNM configurations.
0x0044A17A	Critical	The DNM slave is offline at position 42.	The DNM slave is offline at position 42.	Check the DNM configurations.
0x0044A17B	Critical	The DNM slave is offline at position 43.	The DNM slave is offline at position 43.	Check the DNM configurations.
0x0044A17C	Critical	The DNM slave is offline at position 44.	The DNM slave is offline at position 44.	Check the DNM configurations.
0x0044A17D	Critical	The DNM slave is offline at position 45.	The DNM slave is offline at position 45.	Check the DNM configurations.
0x0044A17E	Critical	The DNM slave is offline at position 46.	The DNM slave is offline at position 46.	Check the DNM configurations.
0x0044A17F	Critical	The DNM slave is offline at position 47.	The DNM slave is offline at position 47.	Check the DNM configurations.
0x0044A180	Critical	The DNM slave is offline at position 48.	The DNM slave is offline at position 48.	Check the DNM configurations.
0x0044A181	Critical	The DNM slave is offline at position 49.	The DNM slave is offline at position 49.	Check the DNM configurations.
0x0044A182	Critical	The DNM slave is offline at position 50.	The DNM slave is offline at position 50.	Check the DNM configurations.
0x0044A183	Critical	The DNM slave is offline at position 51.	The DNM slave is offline at position 51.	Check the DNM configurations.
0x0044A184	Critical	The DNM slave is offline at position 52.	The DNM slave is offline at position 52.	Check the DNM configurations.
0x0044A185	Critical	The DNM slave is offline at position 53.	The DNM slave is offline at position 53.	Check the DNM configurations.

Fault Code	Level	Description	Cause	Remedy
0x0044A186	Critical	The DNM slave is offline at position 54.	The DNM slave is offline at position 54.	Check the DNM configurations.
0x0044A187	Critical	The DNM slave is offline at position 55.	The DNM slave is offline at position 55.	Check the DNM configurations.
0x0044A188	Critical	The DNM slave is offline at position 56.	The DNM slave is offline at position 56.	Check the DNM configurations.
0x0044A189	Critical	The DNM slave is offline at position 57.	The DNM slave is offline at position 57.	Check the DNM configurations.
0x0044A18A	Critical	The DNM slave is offline at position 58.	The DNM slave is offline at position 58.	Check the DNM configurations.
0x0044A18B	Critical	The DNM slave is offline at position 59.	The DNM slave is offline at position 59.	Check the DNM configurations.
0x0044A18C	Critical	The DNM slave is offline at position 60.	The DNM slave is offline at position 60.	Check the DNM configurations.
0x0044A18D	Critical	The DNM slave is offline at position 61.	The DNM slave is offline at position 61.	Check the DNM configurations.
0x0044A18E	Critical	The DNM slave is offline at position 62.	The DNM slave is offline at position 62.	Check the DNM configurations.
0x0044A18F	Critical	The DNM slave is offline at position 63.	The DNM slave is offline at position 63.	Check the DNM configurations.
0x0044A190	Information	DNM slaves 0 to 15 restored from offline	DNM slaves 0 to 15 restored from offline	/
0x0044A191	Information	DNM slaves 16 to 31 restored from offline	DNM slaves 16 to 31 restored from offline	/
0x0044A192	Information	DNM slaves 32 to 47 restored from offline	DNM slaves 32 to 47 restored from offline	/
0x0044A193	Information	DNM slaves 48 to 63 restored from offline	DNM slaves 48 to 63 restored from offline	/

Fault Code	Level	Description	Cause	Remedy
0x0044F001	Information	Low temperature fault cleared	/	/
0x0044F002	Information	High temperature fault cleared	/	/
0x0044F004	Information	Parameter fault cleared	/	/
0x0044F005	Information	The fault "Parameter set version incorrect" has been cleared.	/	/
0x0044F007	Information	Module fault cleared	/	/
0x0044F008	Information	Module fault cleared	/	/
0x0044F009	Information	Module fault cleared	/	/
0x0044F00A	Information	Module fault cleared	/	/
0x0044F00B	Information	Module fault cleared	/	/
0x0044F00C	Information	Module fault cleared	/	/
0x0044F011	Information	Potential group type mismatch cleared	/	/
0x0044F012	Information	Excessive internal temperature deviation of the module cleared	/	/
0x0044F013	Information	User-Side supply overvoltage fault cleared	/	/
0x0044F014	Information	User-Side supply undervoltage fault cleared	/	/
0x0044F015	Information	User-Side power supply detection circuit fault cleared	/	/
0x0044F016	Information	CIP Safety communication fault cleared	/	/
0x0044F030	Information	Overflow error cleared	/	/
0x0044F031	Information	Overlimit fault cleared	/	/
0x0044F032	Information	Underlimit fault cleared	/	/
0x0044F033	Information	Underflow error cleared	/	/
0x0044F034	Information	Channel overload fault cleared	/	/
0x0044F035	Information	Channel overload or disconnection fault cleared.	/	/

Fault Code	Level	Description	Cause	Remedy
0x0044F036	Information	Disconnection fault cleared	/	/
0x0044F037	Information	Channel crosstalk fault cleared	/	/
0x0044F038	Information	Channel-to-power supply short circuit fault cleared	/	/
0x0044F039	Information	Channel-to-ground short-circuit fault cleared	/	/
0x0044F03A	Information	Channel status 0/0 (1oo2 non-equivalent evaluation) difference fault cleared	/	/
0x0044F03B	Information	Channel status 0/1 (1oo2 equivalent evaluation) difference fault cleared	/	/
0x0044F03C	Information	Channel status 1/0 (1oo2 equivalent evaluation) difference fault cleared	/	/
0x0044F03D	Information	Channel status 1/1 (1oo2 non-equivalent evaluation) difference fault cleared	/	/
0x0044F03E	Information	Sensor signal jitter fault cleared	/	/
0x0044F03F	Information	Channel switch fault cleared	/	/
0x0044F040	Information	High frequency fault cleared	/	/
0x0044F041	Information	Channel comparison fault cleared	/	/
0x0044F042	Information	Analog-to-digital conversion error (ADC fault) cleared	/	/
0x0044F043	Information	Error fault cleared	/	/
0x0044F044	Information	TC cold junction overlimit fault cleared	/	/
0x0044F045	Information	TC cold junction underlimit fault cleared	/	/
0x0044F046	Information	TC cold junction fault cleared	/	/

Fault Code	Level	Description	Cause	Remedy
0x00640000	Information	Connected successfully	/	/
0x00640001	Critical	Target device offline	The target device is offline.	1. Check if the target IP is configured properly. 2. Check if the network cable between the devices is disconnected. 3. In case of repeated disconnection and reconnection, try increasing the "Timeout multiplier" parameter on the connection configuration interface.
0x00640002	Critical	Target device online but not responding	The target device is online but not responding.	1. Check whether the target IP is configured properly. 2. Check whether the target device is operating properly.
0x00640003	Warning	Target device dropping frames	The target device drops frames.	Check the network connection stability.
0x00640004	Critical	Connection error	Connection error	1. On the "Connection status" interface, check the specific cause of each connection error, and resolve it according to the instructions. 2. In case of repeated disconnection and reconnection, try increasing the "Timeout multiplier" parameter on the connection configuration interface. 3. If a coupler device is connected, confirm that the configuration of the I/O module devices under the coupler matches the actual connection.
0x00640005	Critical	Declaring IP address failed	The IP address is not declared successfully.	Check if the IP address of the device in the iFA Evolution and the one configured in the network match that of the physical device, and check whether the EIP port number is occupied.

Fault Code	Level	Description	Cause	Remedy
0x00640006	Critical	Multicast not configured for EIP in redundant mode	Multicast is not configured for EIP in redundant mode.	In redundant mode, configure the EIP connection type as multicast in iFA software.
0x00640007	Warning	Ring opened	The ring is disconnected.	Check the EIP network for disconnected cables or powered-off I/O devices.
0x00640008	Information	Ring closed	/	/
0x00640009	Warning	Connection established but target device variables not yet ready	1. The primary PLC or target device is currently loading and starting up. 2. The connection with the primary PLC or target device is not configured properly. 3. Variables are temporarily not accessible due to online modification on the target device.	1. Check whether the target device is currently loading and starting up. 2. Check the connection issue details in the "Connection status" section.
0x0064000A	Warning	Closing connection	/	/
0x0064000B	Critical	EIP thread task exception	An exception occurs in the EIP thread task.	Re-compile and download the project. If the fault persists, contact the manufacturer.
0x0064000C	Critical	PLC and slave connection timeout less than 200 ms in redundant mode	The RPI * timeout multiplier in slave connection configuration of the configuration software is less than 200 ms.	Modify the PRI and timeout multiplier in the configuration software to keep their product greater than 200 ms.
0x0064000D	Critical	The PPS value of an EIP connection exceeds the protocol stack specification limit.	The PPS value of an EIP connection exceeds the protocol stack specification limit.	1. Reduce the number of EIP connections. 2. Increase the RPI parameter of EIP connections.

Fault Code	Level	Description	Cause	Remedy
0x00780005	Critical	Declaring IP address failed	The IP address is not declared successfully.	Check if the IP address of the device in the iFA Evolution and the one configured in the network match that of the physical device, and check whether the EIP port number is occupied.
0x0078000B	Critical	EIP thread task exception	An exception occurs in the EIP thread task.	Re-compile and download the project. If the fault persists, contact the manufacturer.
0x00830000	Information	Connected successfully	/	/
0x00830001	Critical	Target device offline	The target device is offline.	1. Check if the target IP is configured properly. 2. Check if the network cable between the devices is disconnected. 3. In case of repeated disconnection and reconnection, try increasing the "Timeoutmultiplier" parameter on the connection configuration interface.
0x00830002	Critical	Target device online but not responding	The target device is online but not responding.	1. Check whether the target IP is configured properly. 2. Check whether the target device is operating properly.
0x00830003	Warning	Target device dropping frames	The target device drops frames.	Check the network connection stability.
0x00830004	Critical	Connection error	Connection error	1. On the "Connection status" interface, check the specific cause of each connection error, and resolve it according to the instructions. 2. In case of repeated disconnection and reconnection, try increasing the "Timeout multiplier" parameter on the connection configuration interface.

Fault Code	Level	Description	Cause	Remedy
0x00830005	Critical	Declaring IP address failed	The IP address is not declared successfully.	Check whether the IP address of the device in iFA and the one configured in the network match that of the physical device, or use "netstat -ano | find 44818" to check the pid and verify whether the EIP port number is occupied.
0x0083000A	Warning	Closing connection	/	/
0x0083000B	Critical	EIP thread task exception	An exception occurs in the EIP thread task.	Re-compile and download the project. If the fault persists, contact the manufacturer.
0x00641000	Information	Connection closed	The connection is closed.	-
0x00641001	Critical	The device is attempting to reconnect to the target device which has already established a connection.	The device is attempting to reconnect to the target device which has already established a connection.	Check whether the configuration is correct.
0x00641002	Critical	Transmission type and trigger mode not supported	The transmission type and trigger mode are not supported.	Check if the transmission type and trigger mode are set properly.
0x00641011	Critical	Connection ownership conflict. Multiple connection attempts to the same target device.	Connection ownership conflict. Multiple connection attempts to the same target device.	Check if multiple masters are connecting to the device simultaneously or if there is a configuration error.
0x00641012	Critical	Invalid connection size	Invalid connection size	Check if the connection parameters are set correctly.
0x00641013	Critical	The device is not configured. The configuration data is not set for the device requiring configuration.	The device is not configured. The configuration data is not set for the device requiring configuration.	Check if the configuration data length is correctly set and if the configuration path is correct.

Fault Code	Level	Description	Cause	Remedy
0x00641014	Critical	RPI is not supported, or there is an issue with the timeout multiplier or inhibit time.	RPI is not supported, or there is an issue with the timeout multiplier or inhibit time.	Check if the RPI and the timeout multiplier meet the target device requirements.
0x00641015	Critical	RPI is not accepted.	RPI is not accepted.	Check if the RPI meets the target device requirements.
0x00641016	Critical	The local connection is at full capacity. The connection manager cannot support additional connections.	The local connection is at full capacity. The connection manager cannot support additional connections.	Check the project configuration, and if the number of connections to this device has reached the limit
0x00641017	Critical	Vendor Id or Product Code does not match the device.	Vendor Id or Product Code does not match the device.	Check if the configured device matches the target device.
0x00641018	Critical	The product type does not match the device.	The product type does not match the device.	Check if the configured device matches the target device.
0x00641019	Critical	The primary or secondary version number does not match the device.	The primary or secondary version number does not match the device.	Check whether the device version in the EDS file matches the target device.
0x0064101A	Critical	The connection request failed because there are currently no open connections.	The connection request failed because there are currently no open connections.	Check if other connections have been established before using ListenOnly.
0x0064101B	Critical	The target application cannot support more connections.	The target application cannot support more connections.	The target device connection is full, and check the connection status of the target device.
0x0064101C	Critical	RPI is less than the inhibit time.	RPI is less than the inhibit time.	Adjust RPI to a value greater than or equal to the inhibit time.
0x0064101D	Critical	Connection timeout	Connection timeout	In case of connection timeout, check the target device status or link status.

Fault Code	Level	Description	Cause	Remedy
0x0064101E	Critical	An invalid segment path or segment value exists in the path.	An invalid segment path or segment value exists in the path.	/
0x0064101F	Critical	Vendor-specific error	Vendor-specific error	Contact the manufacturer according to the specific error.
0x00641020	Critical	The specified transmission type in the Transmission type/ Trigger mode parameter is not supported.	The specified transmission type in the Transmission type/Trigger mode parameter is not supported.	/
0x00641021	Critical	The specified trigger mode in the Transmission type/ Trigger mode parameter is not supported.	The specified trigger mode in the Transmission type/Trigger mode parameter is not supported.	Check if the trigger type is correct.
0x00641022	Critical	The direction requested in the Transmission type /Trigger mode parameter is not supported.	The direction requested in the Transmission type/Trigger mode parameter is not supported.	/
0x00641023	Critical	O->T fixed/variable flag is not supported.	O->T fixed/ variable flag is not supported.	Check if the O->T fixed/ variable flag is set correctly.
0x00641024	Critical	T->O fixed/variable flag is not supported.	T->O fixed/ variable flag is not supported.	Check if the T-> O fixed/ variable flag is set correctly.
0x00641025	Critical	O->T priority is not supported.	O->T priority is not supported.	Check if the O-> T priority is set correctly and if the target device supports this priority.
0x00641026	Critical	T->O priority is not supported.	T->O priority is not supported.	Check if the T-> O priority is set correctly and if the target device supports this priority.
0x00641027	Critical	O->T connection type is not supported.	O->T connection type is not supported.	Check if the O-> T connection type is set correctly.
0x00641028	Critical	T->O connection type is not supported.	T->O connection type is not supported.	Check if the T->O connection type is set correctly.

Fault Code	Level	Description	Cause	Remedy
0x00641029	Critical	O->T redundancy owner flag is not supported.	O->T redundancy owner flag is not supported.	/
0x0064102A	Critical	The data segment provided in the connection path parameters does not contain the acceptable 16-bit word count for the requested configuration application path. The data area length is invalid.	The data segment provided in the connection path parameters does not contain the acceptable 16-bit word count for the requested configuration application path. The data area length is invalid.	The configuration data length is set incorrectly.
0x0064102B	Critical	The requested size in the O->T network connection parameters is not supported.	The requested size in the O->T network connection parameters is not supported.	Check if the O-> T length is set correctly.
0x0064102C	Critical	The requested size in the T->O network connection parameters is not supported.	The requested size in the T->O network connection parameters is not supported.	Check if the T-> O length is set correctly.
0x0064102D	Critical	The specified configuration application path does not match the valid configuration application path.	The specified configuration application path does not match the valid configuration application path.	Check if the configuration instance ID in the connection path is correct.
0x00641030	Critical	The specified consumption application path does not match the valid consumption application path.	The specified consumption application path does not match the valid consumption application path.	Check if the consumption connection point ID in the connection path is correct.

Fault Code	Level	Description	Cause	Remedy
0x00641031	Critical	The specified production application path does not match the valid production application path.	The specified production application path does not match the valid production application path.	Check if the production connection point ID in the connection path is correct.
0x00641032	Critical	Configuration symbol does not exist.	Configuration symbol does not exist.	/
0x00641033	Critical	Consumption symbol does not exist.	Consumption symbol does not exist.	/
0x00641034	Critical	Production symbol does not exist.	Production symbol does not exist.	/
0x00641035	Critical	The specified application paths for configuration, consumption, and/or production are inconsistent with each other.	The specified application paths for configuration, consumption, and/or production are inconsistent with each other.	Check whether the path is correctly set.
0x00641036	Critical	The information in the data segment is inconsistent with the consumer data format.	The information in the data segment is inconsistent with the consumer data format.	Check whether the configuration data format content is correct.
0x00641037	Critical	The information in the data segment is inconsistent with the production data format.	The information in the data segment is inconsistent with the production data format.	Check whether the configuration data format content is correct.
0x00641041	Critical	The target does not support the function that empty Forward Open requests.	The target does not support the function that empty Forward Open requests.	Check if the connection type is set correctly.

Fault Code	Level	Description	Cause	Remedy
0x00641042	Critical	The connection timeout multiplier is retained, or the resulting timeout value is too large for the device to support.	The connection timeout multiplier is retained, or the resulting timeout value is too large for the device to support.	Check if the timeout multiplier is set correctly.
0x00641043	Critical	The connection size does not match the existing multicast connection.	The connection size does not match the existing multicast connection.	Check if the current ListenOnly length matches the previous connection.
0x00641044	Critical	The fixed/variable connection type does not match the existing multicast connection.	The fixed/variable connection type does not match the existing multicast connection.	Check if the current ListenOnly connection type matches the previous connection.
0x00641045	Critical	The fixed/variable connection priority does not match the existing multicast connection.	The fixed/variable connection priority does not match the existing multicast connection.	Check if the current ListenOnly priority matches the previous connection.
0x0064104F	Critical	The fixed/variable transmission type does not match the existing multicast connection.	The fixed/variable transmission type does not match the existing multicast connection.	Check if the current ListenOnly transfer type matches the previous connection.
0x00641050	Critical	The fixed/variable production trigger does not match the existing multicast connection.	The fixed/variable production trigger does not match the existing multicast connection.	Check if the current ListenOnly trigger type matches the previous connection.
0x00641051	Critical	The fixed/variable production inhibit time does not match the existing multicast connection.	The fixed/variable production inhibit time does not match the existing multicast connection.	Check if the current ListenOnly inhibition time matches the previous connection.
0x00641052	Critical	Other errors	Other errors	Check the specific fault code in the iFA project for the device.

Fault Code	Level	Description	Cause	Remedy
0x0064E001	Critical	System supply voltage (between phases A and B) error	1. The combined 24 V power supply voltage on the system side exceeds the operating voltage range. 2. The hardware circuit is faulty.	Check the stability of the 24 V supply voltage to the system.
0x0064E002	Fatal	FLASH initialization failed	FLASH chip malfunction	Restart or replace the module.
0x0064E003	Fatal	Chip initialization failed	The key chips are not initialized successfully.	Restart or replace the module.
0x0064E004	Critical	Field supply voltage (between phases A and B) error	1. The combined 24 V power supply voltage on the user side exceeds the operating voltage range. 2. The hardware circuit is faulty.	1. Check whether the 24 V supply voltage of the user side is within the proper range. 2. Check the connection between the module and the backplane.
0x0064E005	Fatal	Component initialization failed	The software components are not initialized successfully.	Restart or replace the module.
0x0064E006	Critical	Abnormal semaphores for synchronization	The inter-thread communication fails.	Restart or replace the module.
0x0064E007	Critical	Module parameters not configured properly	Module parameters not configured properly	Check whether the issued parameters are proper.
0x0064E008	Fatal	ADC chip malfunction	ADC chip malfunction	Power on and restart the module.
0x0064E009	Fatal	DAC chip malfunction	DAC chip malfunction	Power on and restart the module.
0x0064E00A	Fatal	DO chip malfunction	DO chip malfunction	Power on and restart the module.
0x0064E00B	Critical	Wire break fault	Open circuit in the channel wiring.	Check the channel wiring for abnormalities.

Fault Code	Level	Description	Cause	Remedy
0x0064E00C	Critical	Short circuit	1. The load is excessive. 2. A short circuit to ground has occurred.	1. Check whether the load meets the specification requirements. 2. Check the wiring for a short circuit to ground.
0x0064E00D	Critical	Short circuit to supply voltage (VS)	1. The channel wiring is shorted to 24 V voltage.	Check whether the channel wiring is short to 24 V voltage.
0x0064E00E	Critical	Reference junction fault	The reference junction is faulty.	/
0x0064E00F	Critical	Overflow error	The acquired value/output value exceeds the maximum value of current range.	Check whether the range is proper or the wiring is improper.
0x0064E010	Critical	Underflow error	The acquired value/output value is below the maximum value of current range.	Check whether the range is proper or the wiring is improper.
0x0064E011	Warning	Over-limit error	The acquired value/output value exceeds the upper limit of current range.	Check whether the range is proper.
0x0064E012	Warning	Under-limit error	The acquired value/output value is below the lower limit of current range.	Check whether the range is proper.
0x0064E013	Critical	User-side module seated status abnormal	The user-side module is not seated properly.	Check the connection between the module and backplane.
0x0064E014	Critical	Key chip configuration lost	Chip configuration mismatch	Re-issue the configuration or restart the module.
0x0064E015	Fatal	FPGA initialization failed	FPGA initialization failed	Power on and restart the module.
0x0064E016	Fatal	System-side module seated status abnormal	The system-side module is not seated properly.	Check the connection between the module and backplane.
0x0064E017	Fatal	PHY chip malfunction	PHY chip malfunction	Power on and restart the module.

Fault Code	Level	Description	Cause	Remedy
0x0064E018	Warning	MCU chip temperature too high or too low	MCU chip temperature too high or too low	Check the operating temperature of the coupler.
0x0064E019	Critical	Dedicated connection disconnected or timed out	The dedicated connection is disconnected or timed out.	Check the connection with the dedicated master station.
0x0064E01A	Critical	Input connection disconnected or timeout	The input connection is disconnected or timed out.	Check the connection with the master station.
0x0064E01B	Critical	Module not detected by the coupler	The module is not configured.	Configure the module or check the module connection.
0x0064E01C	Critical	IP address conflict	An IP address conflict has occurred.	Check and update the IP addresses of the devices in the network.
0x0064E01F	Critical	FPGA communication error	The FPGA communication is faulty.	Power on and restart the module.
0x0064E020	Fatal	Acquiring basic information failed	Acquiring basic information failed	Replace the module or check for interference.
0x0064E021	Critical	Issuing configuration information failed	The module fails to issue the configuration information.	Replace the module or check for interference.
0x0064E022	Critical	Issuing status switch instruction failed	Issuing status switch instruction failed	Replace the module or check for interference.
0x0064E023	Critical	Reading module status failed	Reading the module status fails.	Replace the module or check for interference.
0x0064E024	Critical	User data error	User data error	Replace the module or check for interference.
0x0064E025	Fatal	Module not detected	The module is not detected.	Check the connection between the module and backplane.
0x0064E026	Critical	Type mismatch	The type of the configured module does not match that of the actually scanned one.	Replace with the correct module.

Fault Code	Level	Description	Cause	Remedy
0x0064E027	Critical	Version mismatch	The software version of the configured module does not match that of the actually scanned one.	Update the module firmware.
0x0064E028	Fatal	Module data length out of limit	Total module data length exceeds the acceptable range of the coupler.	Update the firmware or check the module configuration.
0x0064E029	Fatal	Configuration verification error	The controller configuration project differs from the actual devices.	Check the configuration and reconfigure the system.
0x0064E02A	Critical	Flash erase cycles approaching the erase endurance	The use time is long or fault logs are too much.	Replace the module.
0x0064E02B	Warning	Frame loss in periodic data exchange	The firmware is mismatched, or the interference is high.	Update the firmware or check surrounding environment.
0x0064E02C	Warning	Upgrading MCU firmware failed	The firmware transmission fails or encounters an error during upgrade.	Re-upgrade the firmware.
0x0064E02D	Warning	Upgrading FPGA firmware failed	The firmware transmission fails or encounters an error during upgrade.	Re-upgrade the firmware.
0x0064E02E	Warning	Configuration data verification error	The slave configuration data is modified unexpectedly.	Re-download the configuration parameters to the module.
0x0064E02F	Warning	Backplane identification error	The number of backplane slots is incorrect.	Replace or inspect the backplane.
0x0064E030	Warning	No PDO data received	The local bus is interfered.	Remove interference on the local bus.

Fault Code	Level	Description	Cause	Remedy
0x0064E031	Warning	High CPU or memory usage	The module is operating improperly or malfunctioning.	Restart or replace the module.
0x0064E032	Critical	Slave device fault	1. Slave device not responding or response timeout. 2. Slave device response data error. 3. The wiring is abnormal.	1. Check whether the slave device is operating normally. 2. Check whether the serial port parameters of the master/slave, "Response timeout" parameter, "CRC byte order", and "flow control" parameter settings are correct. 3. Check the wiring.
0x0064E033	Critical	Illegal function code	The slave device does not support this parameter.	Check whether the slave device supports this parameter.
0x0064E034	Critical	Illegal address.	The slave device does not support this address.	Check if the slave device supports this address.
0x0064E035	Critical	Illegal data value	An illegal value exists in the request frame.	Check whether the Modbus protocol supports this value.
0x0064E036	Warning	Noise interference.	The communication circuit is subject to interference.	Check the wiring or the noise interference.
0x0064E037	Critical	Parity error.	Parity error in the received serial data.	Check whether the "Parity check", "Data bits", and "Stop bits" settings match between the master and slave.
0x0064E038	Critical	Sensor 24 V power supply fault	The module's 24 V sensor power supply is abnormal.	Check whether the module's 24 V sensor power supply is stable.
0x0064E039	Critical	Sensor 5 V power supply fault	The module's 5 V sensor power supply is abnormal.	Check whether the module's 5 V sensor power supply is stable.
0x0064E03A	Critical	SSI communication error	Encoder wiring fault / noise interference / parity error.	1. Check whether the encoder is subject to interference. 2. Check whether the wiring of the encoder is correct.
0x0064E03B	Critical	Reserved	/	/

Fault Code	Level	Description	Cause	Remedy
0x0064E03C	Critical	Phase A wire break fault	Open circuit in the phase A wiring of the channel.	Check whether the phase A wiring of the channel is correct.
0x0064E03D	Critical	Phase B wire break fault	Open circuit in the phase B wiring of the channel.	Check whether the phase B wiring of the channel is correct.
0x0064E03E	Critical	Phase Z wire break fault	Open circuit in the phase Z wiring of the channel.	Check whether the phase Z wiring of the channel is correct.
0x0064E03F	Critical	DO0 wire break fault	Open circuit in DO0 wiring	Check whether the DO0 wiring is correct.
0x0064E040	Critical	DO1 wire break fault	Open circuit in DO1 wiring	Check whether the DO1 wiring is correct.
0x0064E041	Critical	DO0 shorted to power supply	The DO0 wiring is shorted to 24 V power supply.	Check whether the DO0 wiring is short to 24 V power supply.
0x0064E042	Critical	DO1 shorted to power supply	The DO1 wiring is shorted to 24 V power supply.	Check whether the DO1 wiring is short to 24 V power supply.
0x0064E043	Critical	DO0 shorted to ground	The DO0 wiring is shorted to the ground.	Check whether the DO0 wiring is shorted to the ground.
0x0064E044	Critical	DO1 shorted to ground	The DO1 wiring is shorted to the ground.	Check whether the DO1 wiring is short to the ground.
0x0064E045	Critical	Synchronization signal fault	Channel synchronization signal error	Power cycle or replace the module.
0x0064E046	Critical	Insufficient module memory	The module memory is insufficient.	Upgrade the new firmware or replace the module.
0x0064E047	Critical	Backplane type mismatch	Backplane type mismatch.	Check the configuration and the actual backplane.
0x0064E048	Warning	Process data specification error	The process data does not match the parameter configuration.	Check the process data.
0x0064E101	Information	System supply voltage (between phases A and B) error cleared	System supply voltage (between phases A and B) error cleared	/

Fault Code	Level	Description	Cause	Remedy
0x0064E102	Information	Flash initialization failure cleared	The flash initialization failure is cleared.	/
0x0064E103	Information	Chip initialization failure cleared	Chip initialization failure cleared	/
0x0064E104	Information	Field supply voltage (between phases A and B) error cleared	Field supply voltage (between phases A and B) error cleared	/
0x0064E105	Information	Component initialization failure cleared	Component initialization failure cleared	/
0x0064E106	Information	Abnormal semaphores for synchronization cleared	Abnormal semaphores for synchronization cleared	/
0x0064E107	Information	Module parameter configuration error cleared	Module parameter configuration error cleared	/
0x0064E108	Information	ADC chip fault cleared	ADC chip fault cleared	/
0x0064E109	Information	DAC chip fault cleared	DAC chip fault cleared	/
0x0064E10A	Information	DO chip fault cleared	The DO chip fault has been cleared.	/
0x0064E10B	Information	Wire break fault cleared	The wire break fault is cleared.	/
0x0064E10C	Information	Short circuit cleared	Short circuit cleared	/
0x0064E10D	Information	Short circuit to VS cleared	Short circuit to VS cleared	/
0x0064E10E	Information	Reference junction fault cleared	The reference junction fault is cleared.	/
0x0064E10F	Information	Overflow error cleared	Overflow error cleared	/
0x0064E110	Information	Underflow error cleared	Underflow error cleared	/
0x0064E111	Information	Over-limit error cleared	Over-limit error cleared	/

Fault Code	Level	Description	Cause	Remedy
0x0064E112	Information	Under-limit error cleared	Under-limit error cleared	/
0x0064E113	Information	User-Side module seated status abnormal cleared	User-Side module seated status abnormal cleared	/
0x0064E114	Information	Lost configuration recovered	Lost configuration recovered	/
0x0064E115	Information	FPGA initialization failure cleared	FPGA initialization failure cleared	/
0x0064E116	Information	Module not seated properly cleared	Module not seated properly cleared	/
0x0064E117	Information	PHY chip malfunction cleared	PHY chip malfunction cleared	/
0x0064E118	Information	MCU chip temperature too high or too low fault cleared	The fault "MCU chip temperature too high or too low" has been cleared.	/
0x0064E119	Information	The dedicated connection disconnection or timeout fault has been cleared.	The dedicated connection is disconnected or timed out.	/
0x0064E11A	Information	The dedicated connection disconnection or timeout fault has been cleared.	The dedicated connection disconnection or timeout fault has been cleared.	/
0x0064E11B	Information	Module not detected by the coupler cleared	The module not configured is cleared.	/
0x0064E11C	Information	The IP address conflict is cleared.	The IP address conflict is cleared.	/
0x0064E11F	Information	The FPGA communication error is cleared.	The FPGA communication error is cleared.	/
0x0064E120	Information	Basic information acquisition failure cleared	Basic information acquisition failure cleared	/

Fault Code	Level	Description	Cause	Remedy
0x0064E121	Information	Module configuration information download failure cleared.	The failure to download the configuration information to the module is cleared.	/
0x0064E122	Information	Failure to send state change command to the module cleared.	The failure to send the state switching command to the module is cleared.	/
0x0064E123	Information	Module status read failure cleared	The fault "Module status read failure" has been cleared.	/
0x0064E124	Information	User data error cleared	User data error cleared	/
0x0064E125	Information	Module not detected (offline) cleared	Module not detected (offline) cleared	/
0x0064E126	Information	Module type mismatch cleared	Module type mismatch cleared	/
0x0064E127	Information	Configuration version mismatch cleared	Configuration version mismatch cleared	/
0x0064E128	Information	Module data length out of limit cleared	Module data length out of limit cleared	/
0x0064E129	Information	Configuration verification failure cleared	Configuration verification failure cleared	/
0x0064E12A	Information	The flash erase cycles approaching the erase endurance is cleared.	The flash erase cycles approaching the erase endurance is cleared.	/
0x0064E12B	Information	Frame loss in periodic data exchange cleared	Frame loss in periodic data exchange cleared	/
0x0064E12C	Information	MCU firmware upgrade failure cleared	The MCU firmware upgrade failure is cleared.	/

Fault Code	Level	Description	Cause	Remedy
0x0064E12D	Information	FPGA firmware upgrade failure cleared	The FPGA firmware upgrade failure is cleared.	/
0x0064E12E	Information	Configuration data verification error cleared	Configuration data verification error cleared	/
0x0064E12F	Information	Backplane identification fault cleared	The backplane identification fault is cleared.	/
0x0064E130	Information	Received empty PDO data fault cleared	The fault "Received PDO data is empty" has been cleared.	/
0x0064E131	Information	High CPU or memory usage cleared	High CPU or memory usage cleared	/
0x0064E132	Information	Slave device fault cleared	The slave device fault has been cleared.	/
0x0064E133	Information	Illegal parameter cleared	The illegal parameter has been cleared.	/
0x0064E134	Information	Illegal address cleared	The illegal address has been cleared.	/
0x0064E135	Information	Illegal value cleared	The illegal value has been cleared.	/
0x0064E136	Information	Noise interference cleared	The noise interference has been cleared.	/
0x0064E137	Information	Parity check error cleared	The parity check error has been cleared.	/
0x0064E138	Information	Sensor 24 V power supply fault cleared	The module 24 V sensor power supply fault has been cleared.	/
0x0064E139	Information	Sensor 5 V power supply fault cleared	The module 5 V sensor power supply fault has been cleared.	/

Fault Code	Level	Description	Cause	Remedy
0x0064E13A	Information	SSI communication error cleared	The encoder interference has been cleared, or the wiring is correct.	/
0x0064E13B	Information	Reserved	/	/
0x0064E13C	Information	Phase A open circuit fault cleared	The fault "Open circuit in the phase A wiring of the channel" has been cleared.	/
0x0064E13D	Information	Phase B open circuit fault cleared	The fault "Open circuit in the phase B wiring of the channel" has been cleared.	/
0x0064E13E	Information	Phase C open circuit fault cleared	The fault "Open circuit in the phase Z wiring of the channel" has been cleared.	/
0x0064E13F	Information	DO0 open circuit fault cleared	The fault "Open circuit in DO0 wiring" has been cleared.	/
0x0064E140	Information	DO1 open circuit fault cleared	The fault "Open circuit in DO1 wiring" has been cleared.	/
0x0064E141	Information	The fault "DO0 shorted to power supply" has been cleared.	The fault "The DO0 wiring is shorted to 24 V." has been cleared.	/
0x0064E142	Information	The fault "DO1 shorted to power supply" has been cleared.	The fault "The DO1 wiring is shorted to 24 V." has been cleared.	/
0x0064E143	Information	The fault "DO0 shorted to ground" has been cleared.	The fault "The DO0 wiring is shorted to the ground." has been cleared.	/

Fault Code	Level	Description	Cause	Remedy
0x0064E144	Information	The fault "DO1 shorted to ground" has been cleared.	The fault "The DO1 wiring is shorted to the ground." has been cleared.	/
0x0064E145	Information	Synchronization signal fault cleared	The synchronization signal fault has been cleared.	/
0x0064E146	Information	Insufficient module memory fault cleared.	The insufficient module memory fault has been cleared.	/
0x0064E147	Information	Backplane type mismatch cleared	The backplane type mismatch has been cleared.	/
0x0064E148	Information	Process data specification error cleared	The mismatch between process data and parameter configuration has been cleared.	/
0x0064F001	Information	A too long array in EIP Class3/UCMM causes some elements to be transmitted at different times.	EIP Class3/UCMM contains the arrays with too large length.	It is recommended to set the UCMM array to 200 bytes or less, and the Class3 array to 1400 bytes or less.
0x0064FFFF	Critical	EIP stack startup failed	The EIP stack startup fails.	Try restarting the device, and then use the iFA software to start the EIP task.
0x00831000	Information	Connection closed	The connection is closed.	-
0x0083FFFF	Critical	EIP stack startup failed	The EIP stack startup fails.	Try restarting the device, and then use the iFA software to start the EIP task.
0x64200100	Information	Coupler fault cleared	The coupler fault is cleared.	/
0x64200101	Critical	Coupler software version outdated	The coupler software version is outdated.	Contact the manufacturer for updated coupler software.
0x64200102	Critical	Coupler diagnostic bytes mismatch	The number of coupler diagnostic bytes is mismatched.	Reconfigure the number of coupler diagnostic bytes.

Fault Code	Level	Description	Cause	Remedy
0x64200103	Information	The I/O module fault has been cleared.	The I/O module fault has been cleared.	/
0x64200104	Critical	Getting coupler version information failed	The coupler version information is not obtained.	1. Check the network connection stability. 2. Check whether the coupler is operating properly. 3. Power off and then restart the coupler. If the fault persists, contact the manufacturer.
0x64200105	Critical	Getting coupler diagnostic bytes failed	The number of coupler diagnostic bytes is not obtained.	1. Check the network connection stability. 2. Check whether the coupler is operating properly. 3. Power off and then restart the coupler. If the fault persists, contact the manufacturer.
0x64200106	Warning	Device configuration or module parameter configuration failure	The devices or module parameters are not configured successfully.	1. Check the network connection stability. 2. Check whether the coupler is operating properly. 3. Power off and then restart the coupler. If the fault persists, contact the manufacturer.
0x64201003	Critical	I/O module type mismatch	The type of the configured I/O module is mismatched with that of the actually scanned one.	1. Check the number and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0x64201004	Critical	I/O module configuration failure	The I/O module is not configured successfully.	1. Check the contact of the communication interface for the expansion module or restart the rack.
0x64201005	Critical	Switching I/O module status failed	Switching I/O module status failed	1. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0x64201006	Critical	I/O module offline	The I/O module is offline.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0x64201007	Critical	Module type mismatch	The hardware component has been inserted, but the type of this component does not match that of the configured module.	Check the inserted module, and correct the configuration if necessary.
0x64201008	Critical	New module inserted	A new hardware component has been inserted, and the type of this component matches that of the configured module.	/
0x64201009	Critical	Communication fault: CRC error	Data message frame has transmission error: data inconsistency (CRC (cyclic redundancy check) error).	Check the contact of the communication interface for the expansion module or restart the rack.
0x6420100A	Critical	Communication fault: timeout	Possible causes: 1. The monitoring time is set incorrectly. 2. There is a bus fault.	1. Correct the fault. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0x6420100B	Critical	Configuration conflict	The same I/O module is configured under multiple controllers.	Detect the configuration, resolve the conflict, and then re-download.
0x6420100C	Critical	The stopper plate is faulty or missing.	The stopper plate is faulty or missing.	Reinstall or replace the stopper plate.

Fault Code	Level	Description	Cause	Remedy
0x64202100	Critical	CH0 preset value out of range	The CH0 preset value is out of range.	Modify the preset value to keep it not greater than the maximum value and not smaller than the minimum value.
0x64202101	Critical	CH0 counter configuration error	The CH0 counter is not configured properly.	1. The instruction enabled the single-ended counting function, but the DI is not configured for counting. 2. Check whether the enabled single-ended interface matches the configured DI for counting. 3. Ensure that DI0 and DI1 are not set to counter mode simultaneously. 4. The counter is configured as a differential interface, but DI0 or DI1 is configured with the counting mode.
0x64202102	Critical	CH0 signal type not supported	The CH0 signal type is not supported.	The signal type is not set to an allowed value, which includes phase A/B 1x frequency, phase A/B 2x frequency, phase A/B 4x frequency, pulse + direction, CW/CCW, single-phase counting (A), and single-phase counting (B).
0x64202103	Critical	CH0 counter limit not configured properly	The CH0 counter limit is not configured properly.	Maximum value <= Minimum value
0x64202104	Critical	CH0 unit pulse increment not configured properly	The CH0 unit pulse increment is not configured properly.	The unit pulse increment is less than 1.
0x64202105	Critical	CH0 speed measurement time base not configured properly	The time base for CH0 speed measurement is not configured properly.	Check that the time base is an allowed value.
0x64202106	Critical	CH0 input filter parameter not configured properly	The CH0 input filter parameter is not configured properly.	Check that the input filter parameter is an allowed value.

Fault Code	Level	Description	Cause	Remedy
0x64202107	Critical	CH0 frequency sampling time not configured properly	The CH0 frequency sampling time is not configured properly.	Check that the frequency sampling time is an allowed value.
0x64202108	Critical	CH0 reset function not configured properly	CH0 reset function not configured properly	The clearing function is enabled through commands, but the DI is not configured with the clearing function.
0x64202109	Critical	CH0 preset function not configured properly	The CH0 preset function is not configured properly.	The preset function is enabled through commands, but the DI is not configured with the preset function.
0x6420210A	Critical	CH0 comparison 0 function not configured properly	The CH0 comparison 0 function is not configured properly.	The comparison function is enabled through commands, but the DO is not configured with the comparison function.
0x6420210B	Critical	CH0 comparison 1 function not configured properly	The CH0 comparison 1 function is not configured properly.	The comparison function is enabled through commands, but the DO is not configured with the comparison function.
0x6420210C	Critical	CH0 comparison 2 function not configured properly	The CH0 comparison 2 function is not configured properly.	The comparison function is enabled through commands, but the DO is not configured with the comparison function.
0x6420210D	Critical	CH0 comparison 3 function not configured properly	The CH0 comparison 3 function is not configured properly.	The comparison function is enabled through commands, but the DO is not configured with the comparison function.
0x6420210E	Critical	CH0 probe 0 function not configured properly	The CH0 probe 0 function is not configured properly.	The probe function is enabled through commands, but the DI is not configured with the probe function.
0x6420210F	Critical	CH0 probe 1 function not configured properly	The CH0 probe 1 function is not configured properly.	The probe function is enabled through commands, but the DI is not configured with the probe function.

Fault Code	Level	Description	Cause	Remedy
0x64202110	Critical	CH0 probe 2 function not configured properly	The CH0 probe 2 function is not configured properly.	The probe function is enabled through commands, but the DI is not configured with the probe function.
0x64202111	Critical	CH0 probe 3 function not configured properly	The CH0 probe 3 function is not configured properly.	The probe function is enabled through commands, but the DI is not configured with the probe function.
0x64202112	Critical	CH0 DO0 parameter not configured properly	The CH0 DO0 parameter is not configured properly.	The parameter is not set to the allowed value, which includes normal output and comparison output.
0x64202113	Critical	CH0 DO1 parameter not configured properly	The CH0 DO1 parameter is not configured properly.	The parameter is not set to the allowed value, which includes normal output and comparison output.
0x64202114	Critical	CH0 DO2 parameter not configured properly	The CH0 DO2 parameter is not configured properly.	The parameter is not set to the allowed value, which includes normal output and comparison output.
0x64202115	Critical	CH0 DO3 parameter not configured properly	The CH0 DO3 parameter is not configured properly.	The parameter is not set to the allowed value, which includes normal output and comparison output.
0x64202116	Critical	CH0 DI0 parameter not configured properly	The CH0 DI0 parameter is not configured properly.	The parameter is not set to the allowed value, which includes normal input, probe function, counting, clear, preset, and gating.
0x64202117	Critical	CH0 DI1 parameter not configured properly	The CH0 DI1 parameter is not configured properly.	The parameter is not set to the allowed value, which includes normal input, probe function, counting, clear, preset, and gating.
0x64202118	Critical	CH0 DI2 parameter not configured properly	The CH0 DI2 parameter is not configured properly.	The parameter is not set to the allowed value, which includes normal input, probe function, counting, clear, preset, and gating.
0x64202119	Critical	CH0 DI3 parameter not configured properly	The CH0 DI3 parameter is not configured properly.	The parameter is not set to the allowed value, which includes normal input, probe function, counting, clear, preset, and gating.

Fault Code	Level	Description	Cause	Remedy
0x64205001	Critical	Power supply overvoltage	The power supply voltage is too high.	1. Check the power supply of the module.
0x64205002	Critical	Power supply undervoltage	The power supply voltage is too low.	1. Check the power supply of the module.
0x64205003	Critical	External 24 V power supply failure	The external 24 V power supply fails.	1. Check the power supply of the module.
0x64205004	Critical	Module ADC parameters not configured properly	The ADC parameters of the module are not configured properly.	1. Check the stability of the external 24 V power voltage for the module. 2. Restart the module.
0x64205005	Critical	External 5 V power supply failure	The external 5 V power supply fails.	1. Check the power supply of the module.
0x64205011	Critical	Chip overheating	The chip is overheating.	1. Check the module for hardware faults. 2. Replace the module.
0x64205021	Critical	Module ADC malfunction	The ADC of the module malfunctions.	1. Check the module for hardware faults. 2. Replace the module.
0x64205022	Critical	Module DAC malfunction	The DAC of the module malfunctions.	1. Check the module for hardware faults. 2. Replace the module.
0x64205023	Critical	ADC reference voltage error	The ADC reference voltage is abnormal.	1. Check the module for hardware faults. 2. Replace the module.
0x64205031	Critical	Reference channel fault	The reference channel is faulty.	1. Check the cold junction channel hardware of the thermocouple (TC) module. 2. Replace the module.
0x64206001	Critical	Channel open circuit	The channel is open-circuit.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module or replace the module.

Fault Code	Level	Description	Cause	Remedy
0x64206002	Critical	Channel short circuit	The channel is short-circuited.	1. Check the external wiring of the module channel. 2. If the external wiring is correct, check the internal hardware of the module or replace the module.
0x64206003	Critical	The channel data exceeds the upper limit.	The channel data exceeds the upper limit.	1. Check whether the external input signal of the input module is abnormal.
0x64206004	Critical	The channel data is below the lower limit.	The channel data is below the lower limit.	1. Check whether the external input signal of the input module is abnormal.
0x64206005	Critical	Channel data overflow	The channel data overflows.	1. Check whether the external input signal of the input module is abnormal.
0x64206006	Critical	Channel data underflow	The channel data underflows.	1. Check whether the external input signal of the input module is abnormal.
0x64206007	Critical	Channel overcurrent	The channel current exceeds the limit.	1. Check the external wiring for short circuits. 2. Check the load for overcurrent. 3. Check the module for damage.
0x64206008	Critical	Channel ADC malfunction	The channel ADC malfunctions.	1. Check the module for hardware faults. 2. Replace the module.
0x64206009	Critical	Channel data not configured properly	The channel data is not configured properly.	1. Check whether the downloaded configuration parameters are valid.
0x64206030	Critical	SSI module channel 24 V output error	An error occurs in 24 V output of the SSI module channel.	Check the output voltage.
0x64206031	Critical	SSI module channel communication error	An error occurs during channel communication of the SSI module.	Check the connection between D+ and D- terminals.
0x64206032	Critical	SSI module channel data error	The SSI module channel data is abnormal.	Check whether the baud rate is configured properly.
0x64206050	Critical	Short circuit to VDD	A short circuit to VDD occurs.	Check the hardware wiring.

Fault Code	Level	Description	Cause	Remedy
0x64206051	Critical	Switch-on open circuit	An open circuit occurs when the switch is ON.	Check the hardware wiring.
0x64206052	Critical	Switch-off open circuit	An open circuit occurs when the switch is OFF.	Check the hardware wiring.
0x64206053	Critical	Overcurrent	The current exceeds the limit.	Check the hardware wiring.
0x64206054	Critical	Overload	Overload	Check the hardware wiring.
0x64206800	Critical	Phase A open circuit	An open circuit occurs in phase A.	Check the hardware wiring.
0x64206801	Critical	Phase B open circuit	An open circuit occurs in phase B.	Check the hardware wiring.
0x64206802	Critical	Frequency out of limit	The frequency is out of limit.	Check the hardware wiring.
0x64206803	Critical	Count value overflow	The count value overflows.	Check the hardware wiring.
0x64206804	Critical	Count value underflow	Count value underflow	Check the hardware wiring.
0x64206805	Critical	Comparison function 0 quantity/step out of range	The quantity/step is out of range for the comparison function 0.	Check the hardware wiring.
0x64206806	Critical	Comparison function 1 quantity/step out of range	The quantity/step is out of range for the comparison function 1.	Check the hardware wiring.
0x64206807	Critical	Comparison function 2 quantity/step out of range	The quantity/step is out of range for the comparison function 2.	Check the hardware wiring.
0x64206808	Critical	Comparison function 3 quantity/step out of range	The quantity/step is out of range for the comparison function 3.	Check the hardware wiring.
0x64206809	Critical	Comparison function 0 output hold time/pulse number not configured properly	The output hold time/pulse number is not configured properly for the comparison function 0.	Check the hardware wiring.

Fault Code	Level	Description	Cause	Remedy
0x6420680A	Critical	Comparison function 1 output hold time/pulse number not configured properly	The output hold time/pulse number is not configured properly for the comparison function 1.	Check the hardware wiring.
0x6420680B	Critical	Comparison function 2 output hold time/pulse number not configured properly	The output hold time/pulse number is not configured properly for the comparison function 2.	Check the hardware wiring.
0x6420680C	Critical	Comparison function 3 output hold time/pulse number not configured properly	The output hold time/pulse number is not configured properly for the comparison function 3.	Check the hardware wiring.
0x6420680D	Critical	Comparison function 0 output start/stop position and step direction not configured properly	The output start/stop position and step direction is not configured properly for the comparison function 0.	Check the hardware wiring.
0x6420680E	Critical	Comparison function 1 output start/stop position and step direction not configured properly	The output start/stop position and step direction is not configured properly for the comparison function 1.	Check the hardware wiring.
0x6420680F	Critical	The output start/stop position and step direction is not configured properly for the comparison function 2.	The output start/stop position and step direction is not configured properly for the comparison function 2.	Check the hardware wiring.

Fault Code	Level	Description	Cause	Remedy
0x64206810	Critical	The output start/stop position and step direction is not configured properly for the comparison function 3.	The output start/stop position and step direction is not configured properly for the comparison function 3.	Check the hardware wiring.
0x64206811	Critical	Comparison function 0 comparison value out of range	The comparison value is out of range for the comparison function 0.	Check the hardware wiring.
0x64206812	Critical	Comparison function 1 comparison value out of range	The comparison value is out of range for the comparison function 1.	Check the hardware wiring.
0x64206813	Critical	Comparison function 2 comparison value out of range	The comparison value is out of range for the comparison function 2.	Check the hardware wiring.
0x64206814	Critical	Comparison function 3 comparison value out of range	The comparison value is out of range for the comparison function 3.	Check the hardware wiring.
0x64206815	Critical	Comparison function 0 type not configured properly	The type is not configured properly for the comparison function 0.	Check the hardware wiring.
0x64206816	Critical	Comparison function 1 type not configured properly	The type is not configured properly for the comparison function 1.	Check the hardware wiring.
0x64206817	Critical	Comparison function 2 type not configured properly	The type is not configured properly for the comparison function 2.	Check the hardware wiring.

Fault Code	Level	Description	Cause	Remedy
0x64206818	Critical	Comparison function 3 type not configured properly	The type is not configured properly for the comparison function 3.	Check the hardware wiring.
0x64206819	Critical	Comparison function 0 output control mode not configured properly	The output control mode is not configured properly for the comparison function 0.	Check the hardware wiring.
0x6420681A	Critical	Comparison function 1 output control mode not configured properly	The output control mode is not configured properly for the comparison function 1.	Check the hardware wiring.
0x6420681B	Critical	Comparison function 2 output control mode not configured properly	The output control mode is not configured properly for the comparison function 2.	Check the hardware wiring.
0x6420681C	Critical	Comparison function 3 output control mode not configured properly	The output control mode is not configured properly for the comparison function 3.	Check the hardware wiring.
0x6420681D	Critical	DO0 short circuit	A short circuit occurs in DO0.	Check the hardware wiring.
0x6420681E	Critical	DO1 short circuit	A short circuit occurs in DO1.	Check the hardware wiring.
0x6420681F	Critical	DO2 short circuit	A short circuit occurs in DO2.	Check the hardware wiring.
0x64206820	Critical	DO3 short circuit	A short circuit occurs in DO3.	Check the hardware wiring.
0x64210103	Information	Slave fault cleared	The slave fault has been cleared.	/
0x64211800	Critical	Port connection to no device	Port connection to no device	/
0x64211802	Critical	Invalid vendor ID	Invalid vendor ID	/
0x64211803	Critical	Invalid device ID	Invalid device ID	/

Fault Code	Level	Description	Cause	Remedy
0x64211806	Critical	Pin1 short circuit	Pin1 short circuit	/
0x64211807	Critical	Pin1 overcurrent	Pin1 overcurrent	/
0x64211810	Critical	Pin2 short circuit (DO mode)	Pin2 short circuit (DO mode)	/
0x64216000	Critical	Invalid cycle time	Invalid cycle time	/
0x64216001	Critical	Incorrect version for slave station	Incorrect version for slave station	/
0x64214210	Warning	Overtemperature	Overtemperature	The ambient temperature is high. Lower down the temperature before using the module.
0x64214220	Warning	Low temperature	Low temperature	The ambient environment is low. Increase the temperature before using the module.
0x64215011	Critical	Loss of data for which retention upon power-off is configured	Loss of data for which retention upon power-off is configured	Configuration of retention upon power-off fails. Re-configure the parameter for retention upon power-off.
0x64215100	Critical	Device power supply error	Device power supply error	The UA power supply of the module fails. Enable the power supply to pin 2 on the master or use the external UA power supply.
0x64215110	Warning	System power supply overvoltage	System power supply overvoltage	The US power supply voltage of the module is higher than the normal operating voltage. Reduce the US power supply voltage.
0x64215111	Warning	System power supply undervoltage	System power supply undervoltage	The US power supply voltage of the module is lower than the normal operating voltage. Increase the US power supply voltage.
0x64216320	Critical	Incorrect parameter	Incorrect parameter	The module parameters have been configured incorrectly. Configure the parameters correctly according to the product data.

Fault Code	Level	Description	Cause	Remedy
0x64216321	Critical	Missing parameter	Missing parameter	In expert mode, an input signal is detected for a channel configured as output. Modify the input/output configuration parameter 0x41 according to actual usage.
0x64217710	Critical	DO short circuit	DO short circuit	A short circuit has occurred on the module output. Check the external wiring.
0x64218CA0	Critical	PIN1 short circuit	PIN1 short circuit	A short circuit occurs on the module PIN1. Check the external wiring.
0x64218CA1	Critical	No cascaded devices connected	No cascaded device is connected.	Check if any cascaded devices are connected.
0x64218CA2	Critical	Cascaded device verification failed	Cascaded device verification failed	Check if a compatible cascaded device is connected.
0x64218DA0	Warning	Cascaded device temperature out of range	The cascaded device temperature is out of range.	The ambient temperature of the cascaded device is too high. Decrease the temperature before using the device.
0x64218DA1	Warning	Cascaded device temperature too low	The cascaded device temperature is too low.	The ambient temperature of the cascaded device is too low. Increase the temperature before using the device.
0x64218DA4	Critical	Retentive data loss due to cascaded device power failure	The retentive data upon cascaded device power failure is lost.	Failed to save configuration parameters for the cascaded device. Re-configure the parameters.
0x64218DA6	Critical	Cascaded device power supply error	Power supply error on the cascaded device.	The UA power supply for the cascaded device module is not provided. Enable the master PIN2 power supply or connect an external UA power source.
0x64218DA8	Warning	Cascaded device system power supply voltage out of range	The system power supply voltage for the cascaded device exceeds the limit.	The US power supply voltage of the cascaded device module is higher than the normal operating voltage. Reduce the US power supply voltage.

Fault Code	Level	Description	Cause	Remedy
0x64218DA9	Warning	Cascaded device system power supply undervoltage	The system power supply voltage for the cascaded device is below the limit.	The US power supply voltage of the cascaded device module is lower than the normal operating voltage. Increase the US power supply voltage.
0x64218DAC	Critical	Cascaded device parameter error	The parameter in the cascaded device is faulty.	The parameters for the cascaded device module have been configured incorrectly. Configure the parameters correctly according to the product data.
0x64218DAD	Critical	Missing parameters in the cascaded device	The parameters in the cascaded device are missing.	In expert mode of the cascaded device, if an input signal is detected on a channel that is configured as an output, modify the I/O configuration parameter 0x41 to match the actual usage.
0x64218DAF	Critical	Cascaded device DO short circuit	The DO on the cascaded device is short-circuited.	There is a short circuit on the output of the cascaded device module. Check the external wiring.
0x64218DB0	Critical	Cascaded device PIN1 short circuit	PIN1 on the cascaded device is short-circuited.	There is a short circuit on PIN1 of the cascaded device module. Check the external wiring.
0x64217701	Critical	Port 1 disconnection	A disconnection has occurred at port 1.	Check if the input connection is correct.
0x64217702	Critical	Port 2 disconnection	A disconnection has occurred at port 2.	Check if the input connection is correct.
0x64217703	Critical	Port 3 disconnection	A disconnection has occurred at port 3.	Check if the input connection is correct.
0x64217704	Critical	Port 4 disconnection	A disconnection has occurred at port 4.	Check if the input connection is correct.
0x64217705	Critical	Port 5 disconnection	A disconnection has occurred at port 5.	Check if the input connection is correct.

Fault Code	Level	Description	Cause	Remedy
0x64217706	Critical	Port 6 disconnection	A disconnection has occurred at port 6.	Check if the input connection is correct.
0x64217707	Critical	Port 7 disconnection	A disconnection has occurred at port 7.	Check if the input connection is correct.
0x64217708	Critical	Port 8 disconnection	A disconnection has occurred at port 8.	Check if the input connection is correct.
0x64218C11	Warning	Port 1 input above the upper limit	The port 1 input exceeds the upper limit.	The input exceeds the upper measurement limit. Reduce the input.
0x64218C12	Warning	Port 2 input above the upper limit	The port 2 input exceeds the upper limit.	The input exceeds the upper measurement limit. Reduce the input.
0x64218C13	Warning	Port 3 input above the upper limit	The port 3 input exceeds the upper limit.	The input exceeds the upper measurement limit. Reduce the input.
0x64218C14	Warning	Port 4 input above the upper limit	The port 4 input exceeds the upper limit.	The input exceeds the upper measurement limit. Reduce the input.
0x64218C15	Warning	Port 5 input above the upper limit	The port 5 input exceeds the upper limit.	The input exceeds the upper measurement limit. Reduce the input.
0x64218C16	Warning	Port 6 input above the upper limit	The port 6 input exceeds the upper limit.	The input exceeds the upper measurement limit. Reduce the input.
0x64218C17	Warning	Port 7 input above the upper limit	The port 7 input exceeds the upper limit.	The input exceeds the upper measurement limit. Reduce the input.
0x64218C18	Warning	Port 8 input above the upper limit	The port 8 input exceeds the upper limit.	The input exceeds the upper measurement limit. Reduce the input.
0x64218C21	Critical	Port 1 input overflow	An overflow has occurred at port 1.	The input exceeds the normal measurement range. Check the input.
0x64218C22	Critical	Port 2 input overflow	An overflow has occurred at port 2.	The input exceeds the normal measurement range. Check the input.
0x64218C23	Critical	Port 3 input overflow	An overflow has occurred at port 3.	The input exceeds the normal measurement range. Check the input.

Fault Code	Level	Description	Cause	Remedy
0x64218C24	Critical	Port 4 input overflow	An overflow has occurred at port 4.	The input exceeds the normal measurement range. Check the input.
0x64218C25	Critical	Port 5 input overflow	An overflow has occurred at port 5.	The input exceeds the normal measurement range. Check the input.
0x64218C26	Critical	Port 6 input overflow	An overflow has occurred at port 6.	The input exceeds the normal measurement range. Check the input.
0x64218C27	Critical	Port 7 input overflow	An overflow has occurred at port 7.	The input exceeds the normal measurement range. Check the input.
0x64218C28	Critical	Port 8 input overflow	An overflow has occurred at port 8.	The input exceeds the normal measurement range. Check the input.
0x64218C31	Warning	Port 1 input below lower limit	The port 1 input is below the lower limit.	The input is below the lower measurement limit. Increase the input.
0x64218C32	Warning	Port 2 input below lower limit	The port 2 input is below the lower limit.	The input is below the lower measurement limit. Increase the input.
0x64218C33	Warning	Port 3 input below lower limit	The port 3 input is below the lower limit.	The input is below the lower measurement limit. Increase the input.
0x64218C34	Warning	Port 4 input below lower limit	The port 4 input is below the lower limit.	The input is below the lower measurement limit. Increase the input.
0x64218C35	Warning	Port 5 input below lower limit	The port 5 input is below the lower limit.	The input is below the lower measurement limit. Increase the input.
0x64218C36	Warning	Port 6 input below lower limit	The port 6 input is below the lower limit.	The input is below the lower measurement limit. Increase the input.
0x64218C37	Warning	Port 7 input below lower limit	The port 7 input is below the lower limit.	The input is below the lower measurement limit. Increase the input.
0x64218C38	Warning	Port 8 input below lower limit	The port 8 input is below the lower limit.	The input is below the lower measurement limit. Increase the input.
0x9DD20000	Information	Fault cleared	/	Channel fault cleared

Fault Code	Level	Description	Cause	Remedy
0x9DD20070	Critical	Slave address setting error	The set slave station address does not match the actual slave station address.	1. Check if the slave response data is correct. 2. Check if the master device address matches the slave device address.
0x9DD20072	Critical	Illegal data address	The accessed peer data address does not exist or is out of bounds.	Check if the data access address or data length exceeds the limit.
0x9DD20073	Critical	CRC check error	Data message CRC check error	1. Check if the terminal matching resistor is set properly. 2. Optimize the wiring by eliminating potential interference sources.
0x9DD20074	Critical	Unsupported command code	Attempted to access a command code not supported by the peer.	Check if the slave supports the parameter requested by the master.
0x9DD20075	Critical	Receiving timeout	1. The communication link is faulty. 2. The peer device is not operational.	1. Check if the master serial port configuration is correct. 2. Check if the serial port of the peer device is working properly. 3. Check if the peer device Modbus-RTU is started normally. 4. Optimize the wiring by eliminating potential interference sources.
0x9DD20076	Critical	Illegal data value	The data value is illegal.	Check if the response data from the peer slave is faulty.
0x9DD20077	Critical	Buffer overflow	Buffer overflow	Check if the number of requested coils or registers exceeds the maximum supported by Modbus.
0x9DD20078	Critical	Slave fault	Slave fault	Check if the peer slave device is faulty.
0x9E360000	Information	Fault cleared	/	The channel fault has been cleared.

Fault Code	Level	Description	Cause	Remedy
0x9E360072	Critical	Illegal data address	The accessed peer data address does not exist or is out of bounds.	Check if the data access address or data length exceeds the limit.
0x9E360074	Critical	Unsupported command code	Attempted to access a command code not supported by the peer.	Check if the slave supports the parameter requested by the master.
0x9E360075	Critical	Receiving timeout	1. The communication link is faulty. 2. The peer device is not operational.	1. Check if the master IP address or port is configured correctly. 2. Check if the TCP network of the peer device is working properly. 3. Check if the peer device Modbus-TCP is started normally. 4. Optimize the wiring by eliminating potential interference sources.
0x9E360076	Critical	Illegal data value	The data value is illegal.	Check if the response data from the peer slave is faulty.
0x9E360077	Critical	Buffer overflow	Buffer overflow	Check if the number of requested coils or registers exceeds the maximum supported by Modbus.
0x9E360078	Critical	Slave fault	Slave fault	Check if the peer slave is faulty.
0x9E36009A	Critical	TCP cannot be connected to the device.	TCP link error	1. Check if the master IP address or port is configured correctly. 2. Check if the network of the peer device is working properly. 3. Check if the wiring is disconnected. 4. Optimize the wiring by eliminating potential interference sources.
0x9E35009B	Critical	TCP master IP address declaration failed.	EtherNet port IP address does not match the device network port IP address.	Check if the EtherNet port IP address is set correctly.

Fault Code	Level	Description	Cause	Remedy
0x9E3E009B	Critical	TCP slave IP address declaration failed.	EtherNet port IP address does not match the device network port IP address.	Check if the EtherNet port IP address is set correctly.
0x9E36009C	Critical	TCP slave port is occupied.	The slave station was started on the PC, consuming the Modbus port configured in iFA.	Close the slave started on the PC to release the occupied port.
0x9E980001	Warning	Missing configuration parameter for the PLC's general I/O function: Bus cycle task.	The iFA software version is incompatible. The required parameters are missing in the downloaded device configuration file.	Update the iFA software version. (The current run uses default parameters.)
0x9E980002	Critical	The axis is in commissioning state.	The axis associated with this instruction is currently in commissioning state.	Exit the axis commissioning mode, and then execute this instruction.
0x9E980003	Critical	Invalid terminal setting for HC_PWM.	No appropriate terminal is checked on the iFA interface.	Check the PWM output terminal on the iFA interface.
0x9E980004	Critical	Terminal conflict for HC_PWM.	Multiple instances are output to the same terminal simultaneously.	Check whether multiple instances are output to the same terminal.
0x9E980005	Critical	HC_PWM pulse period and pulse width are both set to 0.	The period and pulse width are both set to 0.	Set a non-zero value.
0x9E980006	Critical	HC_PWM pulse period is set to 0.	The period is set to 0.	Set a non-zero value.
0x9E980007	Critical	HC_PWM pulse width is equal to 0.	The pulse width is set to 0.	Set a non-zero value.
0x9E980008	Critical	HC_PWM pulse width exceeds the pulse period.	The pulse width exceeds the pulse period.	Set the pulse width to a value less than the pulse period.

Fault Code	Level	Description	Cause	Remedy
0x9E980009	Critical	HC_PWM pulse period and pulse width are both set to values below the minimum.	The period and pulse width are both set to values below the minimum.	Set the pulse period to a value of at least 5, and the pulse width to a value of at least 2.
0x9E98000A	Critical	HC_PWM pulse period is set to a value below the minimum.	The period is below the minimum.	Set the period to a value of at least 5.
0x9E98000B	Critical	HC_PWM pulse width is set to a value below the minimum.	The pulse width is below the minimum.	Set the pulse width to a value of at lease 2.
0x9E98000C	Critical	HC_PWM pulse period and pulse width are both set to values above the maximum.	The period and pulse width are both above the maximum.	Set the pulse period and pulse width to values of at most 214748364.
0x9E98000D	Critical	HC_PWM pulse period is set to a value above the maximum.	The pulse period is greater than the maximum value.	Set the pulse period to a value of at most 214748364.
0x9E98000E	Critical	HC_PWM pulse period is set above the maximum.	The pulse width is greater than the maximum value.	Set the pulse width to a value of at most 214748364.
0x9E98000F	Critical	The External input of HC_EnableInterrupt is out of range.	1. The External input is out of range. 2. The pin for External is not configured in the software.	1. Set the External input to a value between 0 and 15. 2. Enable external interrupt for the External pin in the iFA.
0x9E980010	Critical	The Compare input of HC_EnableInterrupt is out of range.	1. The Compare input is out of range. 2. The corresponding channel is not configured with a high-speed counter.	1. Set the Compare input to a value between 0 and 15. 2. Assign the high-speed counter channel configured for Compare to a local encoder axis.
0x9E980011	Critical	HC_EnableInterrupt enabled repeatedly.	Multiple HC_EnableInterrupt function blocks are not allowed to be triggered at the same time.	Use the same HC_EnableInterrupt function block to enable all interrupts.

Fault Code	Level	Description	Cause	Remedy
0x9E980012	Critical	HC_EnableInterrupt is not supported on this device.	The interrupt task is supported on EVO500 and APC500 series devices only.	Check whether the PLC model is from the EVO500 or APC500 series.
0x00400000	Information	No fault	No fault	No fault
0x00400001	Critical	The network card has no connection.	No LINK signal from the network card	Check whether the network cable between the master and slave stations is disconnected.
0x00400002	Warning	Slave station number mismatch	The number of online slaves is greater than that of configured slaves.	Scan devices on the master station or compare device configurations, and check whether the configured number of slave stations matches the number of online slave stations.
0x00400003	Critical	Alias conflict	There is a slave alias conflict when the alias is enabled.	1. Scan devices on the master station or compare device configurations, and check if there is a slave alias conflict when the alias is enabled. 2. Check the PLC log and locate the specific error based on diagnostic information.
0x00400004	Critical	The master device continuously loses frames.	EtherCAT frames are not received.	Check the master station online interface and PLC logs, and locate the specific error based on diagnostic information.
0x00400005	Warning	The master device occasionally loses frames.	The link quality is affected by interference.	1. Replace the network cable. 2. Replace the current or the previous slave. 3. Add a ferrite core or replace the network cable with a higher-grade one. 4. Check the master online interface and PLC logs, and locate the specific error based on diagnostic information.
0x00400006	Warning	Main network card has no connection.	In redundant ring network, the main network card has no connection.	Check whether the network cable of the main network card is disconnected.

Fault Code	Level	Description	Cause	Remedy
0x00400007	Warning	Redundant network card has no connection.	In redundant ring network, the redundant network card has no connection.	Check whether the network cable of the redundant network card is disconnected.
0x00400008	Warning	An alias is 0.	A slave alias name is 0 when the alias is enabled.	1. Scan devices on the master station or compare device configurations, and check if a slave alias name is 0 when the alias is enabled. 2. Check the PLC log and locate the specific error based on diagnostic information.
0x00410000	Information	No fault	No fault	No fault
0x00410001	Critical	The slave is disconnected.	Slave network cable disconnected	1. Check whether the slave is disconnected from the network. 2. Check whether the slave is powered off.
0x00410002	Critical	Type mismatch	Vendor ID, product code, or version number mismatch	1. Use EtherCAT scanning function to compare the configuration with scanning results. 2. Power off and restart the slave which the master failed to access.
0x00410003	Critical	IN/OUT reversal	The IN and OUT ports on the slave are connected in reverse.	1. Check the Ethernet cable connection of the current device. 2. The device may be connected in reverse. Use the functions to scan the device or compare device configuration to identify the location of the reversed slave station, and check its Ethernet cable connection.
0x00410004	Critical	State error	Initialization, pre-operational, safe-operational, operational, or boot error	Check the master station online interface and PLC logs, and locate the specific error based on diagnostic information.

Fault Code	Level	Description	Cause	Remedy
0x00410005	Warning	SDO configuration error	SDO configuration error	Check the master station online interface and PLC logs, and locate the specific error based on diagnostic information.
0x00410006	Critical	Status switchover timeout	Timeout during transition between initialization, pre-operational, safe-operational, operational, or boot states.	Check the master station online interface and PLC logs, and locate the specific error based on diagnostic information.
0x00410007	Warning	SDO transmission timeout	SDO transmission timeout	Check the master station online interface and PLC logs, and locate the specific error based on diagnostic information.
0x00410008	Warning	IN port disconnected	In redundant ring network, the slave IN port is disconnected.	Check whether the Ethernet cable at the slave IN port is disconnected.
0x00410009	Warning	OUT port disconnected	In redundant ring network, the slave OUT port is disconnected.	Check if the Ethernet cable at the slave OUT port is disconnected.
0x00411001	Critical	Undefined error	Any state, 0x134 = 0x0001	/
0x00411002	Critical	No memory	Any state, 0x134 = 0x0002	Insufficient heap memory
0x00411011	Critical	Invalid requested state change	I->S, I->O, P->O, O->B, S->B ,P->B, 0x134=0x0011	Check whether the state requested by the master is valid.
0x00411012	Critical	Unknown requested state change	Any state, 0x134 = 0x0012	Check whether the state requested by the master is valid.
0x00411013	Critical	Bootstrap not supported	I->B, 0x134 = 0x0013. During transition from initialization to bootstrap state, the slave does not support bootstrap state.	Check if the slave XML and slave firmware versions match.

Fault Code	Level	Description	Cause	Remedy
0x00411014	Critical	No valid firmware	I->P, 0x134 = 0x0014. During transition from initialization to pre-operational state, the DPRAM address exceeds the limit.	1. Check if the slave ESC version matches. 2. Check if the MCU software version matches.
0x00411015	Critical	Invalid mailbox configuration	I->B, 0x134=0x0015. Mailbox configuration error during transition from initialization to bootstrap.	1. Check if the slave XML and slave firmware versions match. 2. Check if the slave mailbox SM configuration has been modified.
0x00411016	Critical	Invalid mailbox configuration	I->P, 0x134=0x0016. Mailbox configuration error during transition from initialization to pre-operational state.	1. Check if the slave XML and slave firmware versions match. 2. Check if the slave mailbox SM (0/1) configuration has been modified.
0x00411017	Critical	Invalid sync manager configuration	P->S, S->O, 0x134=0x0017. A fault occurs in the synchronization manager configuration during transition from pre-operational to safe-operational state or from safe-operational to operational state.	1. Check if the slave XML and slave firmware versions match. 2. Check if the slave SM (2/3) configuration has been modified.

Fault Code	Level	Description	Cause	Remedy
0x00411018	Critical	No valid inputs available	O, S->O, 0x134=0x0018. The input data is invalid in operational state or during transition from safe-operational to operational state.	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x00411019	Critical	No valid outputs available	O, S->O, 0x134=0x0019. The output data is invalid in operational state or during transition from safe-operational to operational state.	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x0041101A	Critical	Synchronization fault	O, S->O, 0x134=0x001A. The process data frame is lost or the jitter is excessive in operational state or during transition from safe-operational to operational state.	1. Check whether the network cable is connected reliably. 2. Check the master and slave cycles. 3. Check if there is a value in the slave 0x300 register (The data link error counter is enabled on the master station online interface). 4. Check the parameter value of the slave synchronization loss window (Inovance).
0x0041101B	Critical	Sync manager watchdog	O, S, 0x134=0x001B. Process data watchdog timeout in operational state or safe-operational state.	1. Check the watchdog configuration. 2. Check the master cycle. 3. Check if there is a value in the slave 0x300 register (The data link error counter is enabled on the master station online interface). 4. Check if the network port connection is normal.

Fault Code	Level	Description	Cause	Remedy
0x0041101C	Critical	Invalid sync manager type	O, S, P->S, 0x134=0x001C. A fault occurs in the synchronization manager type configuration in operational state, safe-operational state, or during transition from pre-operational to safe-operational state.	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x0041101D	Critical	Invalid output configuration	O, S, P->S, 0x134=0x001D. A fault occurs in the output synchronization manager configuration in operational state, safe-operational state, or during transition from pre-operational to safe-operational state.	1. Check if the slave XML and slave firmware versions match. 2. Check if the slave output SM (2) configuration has been modified. 3. Check if the PDO configuration is correct (Check if "PDO assignment" or "PDO configuration" is unchecked). 4. Check if the PDO mapping was configured successfully via SDO.
0x0041101E	Critical	Invalid input configuration	O, S, P->S, 0x134=0x001E. A fault occurs in the input synchronization manager configuration in operational state, safe-operational state, or during transition from pre-operational to safe-operational state	1. Check if the slave XML and slave firmware versions match. 2. Check if the slave input SM (3) configuration has been modified. 3. Check if the PDO configuration is correct (Check if "PDO assignment" or "PDO configuration" is unchecked). 4. Check if the PDO mapping was configured successfully via SDO.

Fault Code	Level	Description	Cause	Remedy
0x0041101F	Critical	Invalid watchdog configuration	O, S, P->S, 0x134=0x001F. A fault occurs in the watchdog configuration in operational state, safe-operational state, or during transition from pre-operational to safe-operational state.	1. Check the SM watchdog configuration. 2. Check the watchdog prescaler setting. (reg0x400)
0x00411020	Critical	Starting the slave in cold mode	Any state, 0x134=0x0020	/
0x00411021	Critical	Slave needs INIT	B, P, S, O, 0x134=0x0021	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x00411022	Critical	Slave needs PREOP	S, O, 0x134=0x0022	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x00411023	Critical	Slave needs SAFEOP	O, 0x134=0x0023	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x00411024	Critical	Invalid input mapping	P->S, 0x134=0x0024. A fault occurs in the input PDO mapping configuration during transition from pre-operational to safe-operational state.	1. Check the input PDO mapping configuration (0x1600~0x17FF). 2. Check if the PDO mapping length exceeds the limit.
0x00411025	Critical	Invalid output mapping	P->S, 0x134=0x0025. A fault occurs in the output PDO mapping configuration during transition from pre-operational to safe-operational state.	1. Check the output PDO mapping configuration (0x1A00~0x1BFF). 2. Check if the PDO mapping length exceeds the limit.

Fault Code	Level	Description	Cause	Remedy
0x00411026	Critical	Inconsistent settings	P->S, 0x134=0x0026	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x00411027	Critical	FreeRun not supported	P->S, 0x134=0x0027	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x00411028	Critical	Synchronization not supported	P->S, 0x134=0x0028	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x00411029	Critical	Mode requiring three buffer zones	P->S, 0x134=0x0029. Buffer configuration error during transition from pre-operational to safe-operational state.	1. Check the SM configuration. 2. Check the synchronization mode configuration.
0x0041102A	Critical	Background watchdog	S, O, 0x134=0x002A	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x0041102B	Critical	No valid inputs and outputs	O, S->O, 0x134=0x002B. The input and output data are invalid in operational state or during transition from safe-operational to operational state.	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x0041102C	Critical	Fatal sync error	O, 0x134=0x002C. The synchronization signal is lost in operational state.	1. Check the slave synchronization signal. 2. Check the synchronization cycle setting.

Fault Code	Level	Description	Cause	Remedy
0x0041102D	Critical	No sync error	S->O, 0x134=0x002D. There is no synchronization signal during transition from safe-operational to operational state.	1. Check the slave startup time configuration. 2. Check the time interval between the slave DC enable time and the configured startup time.
0x00411030	Critical	Invalid DC SYNC configuration	O, S->O, P->S, 0x134=0x0030. The synchronization (SYNC) signal is configured incorrectly in operational state, during transition from safe-operational state to operational state, or from pre-operational to safe-operational state.	1. Check the slave synchronization mode configuration. 2. Check the slave ESC status. 3. Check the DC activation status.
0x00411031	Critical	Invalid DC latch configuration	O, S->O, P->S, 0x134=0x0031. The latch (LATCH) signal is configured incorrectly in operational state, during transition from safe-operational state to operational state, or from pre-operational to safe-operational state.	1. Power cycle the device. 2. Contact the manufacturer for assistance.

Fault Code	Level	Description	Cause	Remedy
0x00411032	Critical	PLL error	O, S->O, 0x134=0x0032. A PLL fault occurs in operational state or during transition from safe-operational to operational state.	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x00411033	Critical	DC sync I/O error	O, S->O, 0x134=0x0033	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x00411034	Critical	DC sync timeout fault	O, S->O, 0x134=0x0034	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x00411035	Critical	DC invalid sync period	P->S, 0x134=0x0035	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x00411036	Critical	DC sync0 period	P->S, 0x134=0x0036. The synchronization (SYNC0) signal period is configured incorrectly during transition from pre-operational to safe-operational state.	1. Check the slave synchronization signal (SYNC0) period configuration.
0x00411037	Critical	DC Sync1 period	P->S, 0x134=0x0037. The synchronization (SYNC1) signal period is configured incorrectly during transition from pre-operational to safe-operational state.	1. Check the slave synchronization signal (SYNC1) period configuration.
0x00411041	Critical	AOE mailbox communication error	B, P, S, O, 0x134=0x0041	1. Power cycle the device. 2. Contact the manufacturer for assistance.

Fault Code	Level	Description	Cause	Remedy
0x00411042	Critical	EOE mailbox communication error	B, P, S, O, 0x134=0x0042	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x00411043	Critical	COE mailbox communication error	B, P, S, O, 0x134=0x0043	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x00411044	Critical	FOE mailbox communication error	B, P, S, O, 0x134=0x0044	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x00411045	Critical	SOE mailbox communication error	B, P, S, O, 0x134=0x0045	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x0041104F	Critical	VOE email communication error	B, P, S, O, 0x134=0x004F	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x00411050	Critical	EEPROM no access	Any, 0x134=0x0050	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x00411051	Critical	EEPROM error	Any, 0x134=0x0051	Check if the XML file is programmed.
0x00411052	Critical	External hardware is not ready.	The EtherCAT slave cannot switch states due to loss of connection with an external device or signal.	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x00411060	Critical	Local restart of the slave	Any, 0x134=0x0060	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x00411061	Critical	The device's identification value has been changed.	P, P+E	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x00411070	Critical	Configuration mismatch	The actual device does not match the configuration.	Check if the configuration matches the actual device.
0x00411080	Critical	Power supply undervoltage	/	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x00411081	Critical	Power supply voltage too high	/	1. Power cycle the device. 2. Contact the manufacturer for assistance.

Fault Code	Level	Description	Cause	Remedy
0x00411082	Critical	The temperature is too low.	/	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x00411083	Critical	High temperature	/	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x004110F0	Critical	The application controller is available.	/	1. Power cycle the device. 2. Contact the manufacturer for assistance.
0x00412101	Critical	Wireless connection establishment timeout	The TS terminal device cannot connect to the wireless network.	1. Troubleshoot based on the MS device indicators, and eliminate issues such as power supply faults, overheating, and version upgrade problems. 2. Check the wireless network configuration on the MS side, including frequency channels, number of TS connections, and whitelist settings. 3. Verify that the antennas of the MS and TS devices are correctly installed and tightened. 4. Check the surroundings for radio interference from other devices. Use a near-end tool with a frequency scanning function to scan for a clear frequency band. 5. Check if there is significant obstruction or excessive distance between the MS and the TS devices.
0x00412102	Critical	MS is not the first slave.	The MS is not directly connected to the PLC; other devices exist between them.	1. Check if there are any other devices between the PLC and the MS. If there are, remove them and reconnect them to the MS's three-port wired side.

Fault Code	Level	Description	Cause	Remedy
0x00412103	Critical	Wireless side real-time channel resource allocation failed.	Under the current channel conditions, the wireless side is unable to allocate sufficient real-time channel resources for transmitting PDO data.	1. Check if the antennas of the MS and TS devices are properly tightened. 2. Check the surroundings for radio interference from other devices. Use a near-end tool with a frequency scanning function to scan for a clear frequency band. 3. Check if there is significant obstruction or excessive distance between the MS and the TS devices. 4. It is recommended to enable LRW mode. 5. Reduce the number of wireless slave devices or reduce the PDO mapping data for the slaves.
0x00412104	Critical	PLC transmission phase is not aligned with the wireless side.	In MS mode, there are scenarios where no ECAT PDO frames or multiple ECAT PDO frames are received within a single communication cycle.	1. Check and increase the synchronization offset parameter in the background configuration, then restart the PLC. 2. Alternatively, increase the PLC control cycle and configure the corresponding synchronization offset parameter. 3. Replace the PLC with a high-performance model that has lower packet jitter.

Fault Code	Level	Description	Cause	Remedy
0x00412105	Warning	Non-process data continuous loss	Under current channel conditions, the non-process data packet error rate is high or the transmission delay is large on the wireless side.	1. Reduce the number of wireless-side slaves or reduce the number of TS devices. 2. It is recommended to enable slave auto-restart. 3. Check if the antennas of the MS and TS devices are properly tightened. 4. Check the surroundings for radio interference from other devices. Use a near-end tool with a frequency scanning function to scan for a clear frequency band. 5. Check if there is significant obstruction or excessive distance between the MS and the TS devices.
0x00412106	Warning	Process data continuous loss	Under current channel conditions, the process data packet error rate is high on the wireless side.	1. Check if the antennas of the MS and TS devices are properly tightened. 2. Check the surroundings for radio interference from other devices. Use a near-end tool with a frequency scanning function to scan for a clear frequency band. 3. Check if there is significant obstruction or excessive distance between the MS and the TS devices.
0x00412107	Warning	MS processing PLC data continuous abnormality	Some PLC data frames are not sent to slaves.	1. It is recommended to enable slave auto-restart. 2. Reduce the number of wireless-side slaves or reduce the number of TS devices, then restart the system.
0x00412108	Critical	Invalid iFA configuration parameters	iFA configuration parameter validation fails on the MS side.	1. Check the Mac address, frequency points, wireless frequency band identifier, and other parameters configured on the MS.

Fault Code	Level	Description	Cause	Remedy
0x00412109	Critical	Failed to read system time	MS fails to read the system time as the first slave.	1. Restart the wireless MS device and check if the fault is cleared. 2. If the fault cannot be cleared, it is recommended to replace the wireless MS device.
0x00412301	Critical	IN/OUT reverse connection detected on TS side	The IN/OUT connection for the slave on the TS side is reversed.	1. Check each connected slave by its TS-id to determine if there is any IN/OUT reverse connection.
0x00412302	Critical	Abnormal communication between TS and slave	The communication between the TS and its connected slaves is abnormal.	1. Check whether the TS is connected to an EtherCAT slave device. 2. Check if any slave device connected after the TS reports an alarm. If an alarm is present, clear it and restart the PLC. 3. Check if any non-ECAT slave switch exists among the devices connected after the TS.
0x00412303	Critical	Synchronization anomaly detected on TS side	The synchronization function on the TS side is abnormal.	1. Try to restart the PLC service. 2. Power off and then power on the slave device, restart the wireless system, and re-run the PLC. 3. Replace with a DC synchronous high-performance slave. 4. Use non-DC mode on the PLC to control slave operation.
0x00412304	Warning	Changing number of devices connected to TS	Devices connected to the TS side are removed or added.	1. Check if the network cables between the TS and the slaves are disconnected. 2. When the number of devices connected to the TS side changes, re-run the PLC.

Fault Code	Level	Description	Cause	Remedy
0x00412305	Critical	TS offline	Some TS devices are offline in the wireless system, resulting in communication interruption.	1. Troubleshoot based on the MS device indicators, and eliminate issues such as power supply faults, overheating, and version upgrade problems. 2. Verify that the antennas of the MS and TS devices are correctly installed and tightened. 3. Check the surroundings for radio interference from other devices. Use a near-end tool with a frequency scanning function to scan for a clear frequency band. 4. Check if there is significant obstruction or excessive distance between the MS and the TS devices.
0x00412306	Warning	Topological anomaly	TS detects a branch-type slave.	1. Check if a branch-type device is connected to the TS side and if there are three or more ports connected.
0x04030000	Information	Fault cleared	Fault cleared	The fault has been cleared.
0x04030004	Critical	The following error is out of range.	Tracking error out of range	1. Check if "Position hysteresis management" is enabled in the software settings. 2. Increase the set value of "Hysteresis limit". 3. If possible, increase the stiffness parameters of the servo drive.
0x04030005	Critical	EtherCAT communication error	Slave communication error	1. Check if the physical network of the slave station is disconnected. 2. Check whether the current state and requested state of the slave are Operational. 3. Manually switch to operational state. 4. Check if there is a drive fault.

Fault Code	Level	Description	Cause	Remedy
0x04030006	Critical	The axis cycle time is less than 0.	The axis cycle time is incorrectly set.	1. Check if the task cycle time in the axis parameters is less than 0.
0x04030007	Critical	Control axis emergency stop active	The axis is in emergency stop control.	1. Check if the immediate stop instruction was called. 2. Call the MC_Reset instruction to reset the fault. 3. Clear the emergency stop status via the software cold reset or hot reset.
0x04030009	Critical	Servo drive alarm or fault	Servo drive alarm or fault	1. Drive error. Check the value 0x603F in the object dictionary. 2. Determine the drive fault type according to the corresponding drive guide, and then clear the fault.
0x0403000A	Critical	Synchronization error	Function block and input/output interface refresh cycle do not match.	1. The input/output refresh cycle is out of sync with the function block call cycle. 2. If the program is written in ST, check whether the motion function block is being inhibited by an IF condition during execution. 3. If the program is written in LD language EN/ENO, check whether the motion function block is being inhibited by an EN during execution.
0x0403000B	Critical	Faults occurred during homing.	Faults occurred during homing.	1. Check whether the limit switch signal is installed in an incorrect position, causing simultaneous response of the limit switch signal. 2. Check whether the homing process exceeds the homing timeout.
0x0403000C	Critical	SDO data read timeout	SDO data read timeout	1. Check for communication faults on the slave.

Fault Code	Level	Description	Cause	Remedy
0x0403000D	Critical	The servo absolute value system does not match the position restoration mode.	The servo absolute value system does not match the position restoration mode.	1. Select a recovery mode compatible with the position system.
0x0403000E	Critical	The area mapped to the axis and storing its key data is not found.	The area mapped to the axis and storing its key data is not found.	1. The position system cannot be restored to normal.
0x0403000F	Critical	The power-retain variables needed for the servo position recovery scheme are missing.	The power-retain variables needed for the servo position recovery scheme are missing.	1. The position system cannot be restored to normal.
0x04030010	Critical	The power-retain variables needed for the PLC position recovery scheme are missing.	The power-retain variables needed for the PLC position recovery scheme are missing.	1. The position system cannot be restored to normal.
0x04030011	Critical	The encoder revolution count after restoration exceeds the limited range of the data type.	The encoder revolution count after restoration exceeds the limited range of the data type.	1. The position system cannot be restored to normal.
0x04030012	Critical	Abnormal termination during input parameter processing.	Abnormal termination during input parameter processing.	1. Check whether the communication status is abnormal. 2. Check whether the axis controlled by the instruction is in a faulty state.
0x04030001	Critical	The axis is in commissioning state.	The axis is in commissioning state.	1. Exit the axis commissioning mode, then execute the corresponding instruction.
0x01957001	Warning	The temperature is too low.	The internal temperature of the module box is lower than the lower limit.	Operate the F module within the specified temperature range.

Fault Code	Level	Description	Cause	Remedy
0x01957002	Warning	High temperature	The internal temperature of the module box is higher than the upper limit.	Operate the F module within the specified temperature range.
0x01957004	Critical	Incorrect parameter	Parameter configuration error or out-of-range diagnostics	Correct the parameter assignment.
0x01957005	Critical	The parameter set version is incorrect.	The version of the downloaded parameter set does not match the version of the parameter set running in the module.	Check the parameter assignment.
0x01957007	Critical	Module error	Possible causes: 1. Excessive electromagnetic interference occurred. 2. Module initialization error. 3. Power-on self-test error (peripheral).	1. Correct the fault. Then, remove and reinsert the module, or power cycle it. 2. If the module cannot be put back into operation, consider replacing it with a new one.
0x01957008	Critical	Module error	Possible causes: 1. Excessive electromagnetic interference occurred. 2. MCU self-diagnosis error (CPU, ROM, RAM)	1. Correct the fault. Then, remove and reinsert the module, or power cycle it. 2. If the module cannot be put back into operation, consider replacing it with a new one.
0x01957009	Critical	Module error	Possible causes: 1. Excessive electromagnetic interference. 2. Dual MCU interaction error.	1. Correct the fault. Then, remove and reinsert the module, or power cycle it. 2. If the module cannot be put back into operation, consider replacing it with a new one.

Fault Code	Level	Description	Cause	Remedy
0x0195700A	Critical	Module error	Possible causes: 1. Excessive electromagnetic interference occurred. 2. Clock diagnosis error.	1. Correct the fault. Then, remove and reinsert the module, or power cycle it. 2. If the module cannot be put back into operation, consider replacing it with a new one.
0x0195700B	Critical	Module error	Possible causes: 1. Excessive electromagnetic interference occurred. 2. Program execution sequence error.	1. Correct the fault. Then, remove and reinsert the module, or power cycle it. 2. If the module cannot be put back into operation, consider replacing it with a new one.
0x0195700C	Critical	Module error	Possible causes: 1. Excessive electromagnetic interference occurred. 2. Watchdog trip.	1. Correct the fault. Then, remove and reinsert the module, or power cycle it. 2. If the module cannot be put back into operation, consider replacing it with a new one.
0x01957011	Fatal	Potential group type mismatch	Incorrect potential group type	Replace with the correct potential group module.
0x01957012	Warning	Excessive internal temperature deviation of the module	Possible causes: 1. Excessive electromagnetic interference occurred. 2. Excessive internal temperature deviation of the module.	If the fault persists and cannot be cleared automatically, replace the module with a new one.
0x01957013	Warning	The user-side power supply voltage is too high.	Power supply voltage too high	Check the user-side power supply voltage.
0x01957014	Critical	The user-side power supply voltage is too low.	Power supply undervoltage	Check the user-side power supply voltage.
0x01957015	Critical	User-Side power supply detection circuit fault	Possible causes: 1. Increased electromagnetic interference. 2. F module fault	1. Eliminate/reduce electromagnetic interference. 2. Replace the F module.

Fault Code	Level	Description	Cause	Remedy
0x01957016	Critical	Communication fault: CIP Safety communication fault	CIP Safety communication fault (CRC error, mixed frame error, data frame timeout, frame sequence error, etc.) Possible causes: 1. Increased electromagnetic interference. 2. F module fault	1. Eliminate/reduce electromagnetic interference. 2. Replace the F module.
0x01957017	Critical	The module safety address is out of range.	The DIP switch address on the module is set to all zeros or all ones.	Check the DIP switch address setting on the module and power cycle it.
0x01957030	Critical	Overflow	Measurement overflow	Check the communication connection between the module and the sensor.
0x01957031	Critical	Overlimit	Measurement overlimit	Check the communication connection between the module and the sensor.
0x01957032	Critical	Underlimit	Measurement underlimit	Check the communication connection between the module and the sensor.
0x01957033	Critical	Underflow	Measurement underflow	Check the communication connection between the module and the sensor.
0x01957034	Critical	Channel overload	Channel overload	Check the actuator.
0x01957035	Critical	Channel overload or disconnection	Possible causes: 1. Channel overload. 2. Channel not connected. 3. External cable disconnection of the channel.	1. Eliminate the overload. 2. Connect the external cable. 3. Disable overload and wire break detection in the parameters.

Fault Code	Level	Description	Cause	Remedy
0x01957036	Critical	Wiring break	Possible causes: 1. External cable disconnected. 2. External circuit fault. 3. Sensor fault. 4. Incorrect sensor type set in the parameters. 5. Input channel unused. 6. Output channel not connected. 7. F module fault	1. Identify and resolve possible causes of the fault. 2. Replace the F module.
0x01957037	Critical	Interference among channels	Interference among channels	1. Eliminate any crosstalk in the process wiring. 2. Check the configured test time and the process wiring.
0x01957038	Critical	Channel-to-power supply short circuit	Possible causes: 1. Channel-to-power supply short circuit. 2. Channel readback error. 3. Short circuit between two output channels. 4. Excessive capacitive load.	1. Eliminate any crosstalk in the process wiring. 2. Check the configured test time and the process wiring. 3. Increase the test time (shutdown test).
0x01957039	Critical	Channel-to-ground short circuit	A short circuit to ground means: 1. Output cable shorted to ground. 2. Output signal shorted to ground. 3. Capacitive load too high. 4. Channel overload. 5. Channel read-back fault.	1. Correct the process wiring. 2. Increase the test time (shutdown test). 3. Check the actuator. 4. Check load conditions. 5. Replace the F module.

Fault Code	Level	Description	Cause	Remedy
0x0195703A	Critical	Differential fault, channel status 0/0 (1oo2 non-equivalent evaluation)	The difference time set for the 1oo2 evaluation has been exceeded. Possible causes: 1. Process signal error. 2. Sensor fault. 3. Configured discrepancy time too low. 4. Channel-to-ground short circuit or open circuit. 5. Fault occurred during discrepancy check.	1. Check the process signal. 2. Replace the sensor. 3. Check the parameter assignment for the error time. 4. Check the process wiring.
0x0195703B	Critical	Differential fault, channel status 0/1 (1oo2 equivalent evaluation)	The difference time set for the 1oo2 evaluation has been exceeded. Possible causes: 1. Process signal error. 2. Sensor fault. 3. Configured discrepancy time too low. 4. Channel-to-ground short circuit or open circuit. 5. Fault occurred during discrepancy check.	1. Check the process signal. 2. Replace the sensor. 3. Check the parameter assignment for the error time. 4. Check the process wiring.

Fault Code	Level	Description	Cause	Remedy
0x0195703C	Critical	Differential fault, channel status 1/0 (1oo2 equivalent evaluation)	The difference time set for the 1oo2 evaluation has been exceeded. Possible causes: 1. Process signal error. 2. Sensor fault. 3. Configured discrepancy time too low. 4. Channel-to-ground short circuit or open circuit. 5. Fault occurred during discrepancy check.	1. Check the process signal. 2. Replace the sensor. 3. Check the parameter assignment for the error time. 4. Check the process wiring.
0x0195703D	Critical	Differential fault, channel status 1/1 (1oo2 non-equivalent evaluation)	The difference time set for the 1oo2 evaluation has been exceeded. Possible causes: 1. Process signal error. 2. Sensor fault. 3. Configured discrepancy time too low. 4. Channel-to-power supply short circuit or channel-to-channel short circuit. 5. Fault occurred during discrepancy check.	1. Check the process signal. 2. Replace the sensor. 3. Check the parameter assignment for the error time. 4. Check the process wiring.

Fault Code	Level	Description	Cause	Remedy
0x0195703E	Critical	Sensor signal jitter	The number of signal changes exceeds the limit within the configured "Monitoring Window" time. Possible causes: 1. "Jitter monitoring window time" parameter set too high. 2. "Number of signal changes" parameter set too low. 3. Momentary break or short circuit (poor contact) in the sensor cable. 4. Excessive electromagnetic interference. 5. Sensor or switch switchover. 6. Sensor failure.	1. Check the "Monitoring Window" parameter. 2. Check the "Number of Signal Changes" parameter. 3. Check the process wiring. 4. Eliminate/reduce electromagnetic interference. 5. Replace the sensor.
0x0195703F	Fatal	Channel switch fault	Possible causes: 1. Increased electromagnetic interference. 2. Channel-to-power supply short circuit. 3. Switch fault inside the module. 4. Excessive capacitive load.	1. Eliminate/reduce electromagnetic interference. 2. Monitor the process signal. 3. Increase the test time (shutdown test). 4. Replace the F module.
0x01957040	Warning	Too high frequency	Maximum switching frequency of the module exceeded.	Reduce the switching frequency.

Fault Code	Level	Description	Cause	Remedy
0x01957041	Fatal	Channel comparison fault	Code value comparison error between the two MCUs. Possible causes: 1. Increased electromagnetic interference 2. F module fault	1. Eliminate/reduce electromagnetic interference. 2. Replace the F module.
0x01957042	Fatal	Analog-to-digital conversion error (ADC fault)	Internal error during analog-to-digital conversion. Possible causes: 1. Increased electromagnetic interference. 2. F module fault	Replace the F module.
0x01957043	Critical	Error time exceeded	The error time set for the 1oo2 evaluation has been exceeded. Possible causes: 1. Improper parameter assignment. 2. Fault occurred in one of the two sensors.	Check the error time, sensor, and wiring.
0x01957044	Critical	TC cold junction measured overlimit	TC cold junction measured value exceeds upper limit.	Check the configuration parameters of TC cold junction upper and lower limits, as well as the TC cold junction sensor and wiring.
0x01957045	Critical	TC cold junction measured underlimit	TC cold junction measured value exceeds lower limit.	Check the configuration parameters of TC cold junction upper and lower limits, as well as the TC cold junction sensor and wiring.

Fault Code	Level	Description	Cause	Remedy
0x01957046	Critical	TC cold junction fault	The value is outside the measurement range of the TC cold junction. Possible causes: 1. External TC cold junction fault (external cold junction mode). 2. TC backplane fault (internal cold junction mode)	1. Check the TC cold junction sensor and wiring (external cold junction mode). 2. Replace the TC module backplane (internal cold junction mode).
0x0195F001	Information	Low temperature fault cleared	/	/
0x0195F002	Information	High temperature fault cleared	/	/
0x0195F004	Information	Parameter fault cleared	/	/
0x0195F005	Information	The fault "Parameter set version incorrect" has been cleared.	/	/
0x0195F007	Information	Module fault cleared	/	/
0x0195F008	Information	Module fault cleared	/	/
0x0195F009	Information	Module fault cleared	/	/
0x0195F00A	Information	Module fault cleared	/	/
0x0195F00B	Information	Module fault cleared	/	/
0x0195F00C	Information	Module fault cleared	/	/
0x0195F011	Information	Potential group type mismatch cleared	/	/
0x0195F012	Information	Excessive internal temperature deviation of the module cleared	/	/
0x0195F013	Information	User-Side supply overvoltage fault cleared	/	/
0x0195F014	Information	User-Side supply undervoltage fault cleared	/	/
0x0195F015	Information	User-Side power supply detection circuit fault cleared	/	/

Fault Code	Level	Description	Cause	Remedy
0x0195F016	Information	CIP Safety communication fault cleared	/	/
0x0195F030	Information	Overflow error cleared	/	/
0x0195F031	Information	Overlimit fault cleared	/	/
0x0195F032	Information	Underlimit fault cleared	/	/
0x0195F033	Information	Underflow error cleared	/	/
0x0195F034	Information	Channel overload fault cleared	/	/
0x0195F035	Information	Channel overload or disconnection fault cleared.	/	/
0x0195F036	Information	Disconnection fault cleared	/	/
0x0195F037	Information	Channel crosstalk fault cleared	/	/
0x0195F038	Information	Channel-to-power supply short circuit fault cleared	/	/
0x0195F039	Information	Channel-to-ground short-circuit fault cleared	/	/
0x0195F03A	Information	Channel status 0/0 (1oo2 non-equivalent evaluation) difference fault cleared	/	/
0x0195F03B	Information	Channel status 0/1 (1oo2 equivalent evaluation) difference fault cleared	/	/
0x0195F03C	Information	Channel status 1/0 (1oo2 equivalent evaluation) difference fault cleared	/	/
0x0195F03D	Information	Channel status 1/1 (1oo2 non-equivalent evaluation) difference fault cleared	/	/
0x0195F03E	Information	Sensor signal jitter fault cleared	/	/
0x0195F03F	Information	Channel switch fault cleared	/	/
0x0195F040	Information	High frequency fault cleared	/	/

Fault Code	Level	Description	Cause	Remedy
0x0195F041	Information	Channel comparison fault cleared	/	/
0x0195F042	Information	Analog-to-digital conversion error (ADC fault) cleared	/	/
0x0195F043	Information	Error fault cleared	/	/
0x0195F044	Information	TC cold junction overlimit fault cleared	/	/
0x0195F045	Information	TC cold junction underlimit fault cleared	/	/
0x0195F046	Information	TC cold junction fault cleared	/	/
0x01911000	Warning	Protocol stack failed to scan for modules	Possible causes: 1. The expansion module does not exist. 2. No expansion module was scanned due to local bus communication failure.	1. Check whether the expansion module is properly installed and supplied with power. 2. Check the contact of the communication interface of the expansion module or restart the rack.
0x01911001	Critical	The number of I/O modules configured is more than the number of I/O modules actually scanned.	Possible causes: 1. The actual slot position of the expansion module does not match the configuration settings. 2. A local bus communication fault causes the scanned modules to differ from the configured modules.	1. Check the number and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0x01911002	Critical	The number of I/O modules configured is less than the number of I/O modules actually scanned.	Possible causes: 1. The actual slot position of the expansion module does not match the configuration settings. 2. A local bus communication fault causes the scanned modules to differ from the configured modules.	1. Check the number and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0x01911003	Critical	The type of I/O modules configured is inconsistent with that of the I/O modules actually scanned.	Possible causes: 1. The actual slot position of the expansion module does not match the configuration settings. 2. A local bus communication fault causes the scanned modules to differ from the configured modules.	1. Check the number and installation order of the expansion modules. 2. Check the contact of the communication interface for the expansion module or restart the rack.
0x01911004	Warning	Module configuration failure	The local bus communication failure causes an error when the master module interacts with the expansion module.	Check the contact of the communication interface of the expansion module or restart the rack.
0x01911005	Warning	The module status is not switched successfully.	The local bus communication failure causes an error when the master module interacts with the expansion module.	Check the contact of the communication interface of the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0x01911006	Warning	The module is offline or has been removed.	Possible causes: 1. The I/O module is powered off or unplugged. 2. The error is caused by an excessive frame loss rate in local bus communication.	1. Check whether the module is powered off or removed from the slot. 2. Check the contact of the communication interface of the expansion module or restart the rack.
0x01911007	Warning	Module type mismatch	The hardware component has been inserted, but the type of this component does not match that of the configured module.	Check the inserted module, and correct the configuration if necessary.
0x01911008	Information	New module inserted	A new hardware component has been inserted, and the type of this component matches that of the configured module.	/
0x01911009	Warning	Local bus cyclic data communication error	Local bus cyclic data frame transmission error: Data inconsistency (CRC check error, etc.). Possible cause: 1. Increased electromagnetic interference. 2. F module fault	1. Check the contact of the communication interface of the expansion module or restart the rack. 2. Eliminate or reduce the electromagnetic interference. 3. Replace the F module.
0x0191100A	Warning	Local bus cyclic data communication timeout	Local bus cyclic data frame transmission timeout. Possible causes: 1. Incorrect communication period setting. 2. Bus faults exist.	1. Modify the communication cycle. 2. Check the contact of the communication interface for the expansion module or restart the rack.

Fault Code	Level	Description	Cause	Remedy
0x0191100B	Critical	Configuration conflict	The same I/O module is configured under multiple controllers.	Detect the configuration, resolve the conflict, and then re-download.
0x0191100C	Critical	The stopper plate is faulty or missing.	The stopper plate is faulty or missing.	Reinstall or replace the stopper plate.
0x0191100D	Critical	Module watchdog timeout	Possible causes: 1. Module MCU power loss. 2. Module MCU crash.	1. Check the module power supply. In case of power loss, restore the power supply. 2. Power cycle the module.
0x0191100E	Critical	IP address conflict	IP address conflict among network devices (PLCs, couplers, etc.) in the configuration	Check the IP addresses of the network devices (PLCs, couplers, etc.) in the configuration.
0x0191100F	Critical	The I/O module does not support the currently selected industry standard.	The I/O module does not support the currently selected industry standard.	Replace or update the I/O module.
0x01911010	Critical	Acyclic data communication fault: Unexpected	Local bus acyclic data frame transmission error: Data inconsistency (CRC error, request frame error, invalid response frame length, service code mismatch, sequence number mismatch, and so on.) Possible causes: 1. Increased electromagnetic interference 2. I/O module firmware version mismatch	1. Check the contact of the communication interface of the expansion module or restart the rack. 2. Eliminate or reduce the electromagnetic interference. 3. Replace or update the I/O module.

Fault Code	Level	Description	Cause	Remedy
0x01911011	Critical	Acyclic data communication fault: Timeout	Local bus cyclic data frame transmission timeout. Possible causes: 1. I/O module MCU power loss or hang 2. I/O module firmware version mismatch. 3. Bus fault present.	1. Check the module power supply or power cycle the module. 2. Replace or update the I/O module. 3. Check the contact of the communication interface of the expansion module or restart the rack.
0x01919001	Information	Configuration match successful	The fault "The number of I/O modules configured exceeds the number of I/O modules actually scanned" has been cleared.	/
0x01919002	Information	Configuration match successful	The fault "The number of I/O modules configured is less than the number of I/O modules actually scanned" has been cleared.	/
0x01919003	Information	Configuration match successful	The fault "The type of I/O modules configured is inconsistent with that of the I/O modules actually scanned" has been cleared.	/
0x01919004	Information	Module configuration successful	/	/
0x01919005	Information	The module status is switched successfully.	/	/

Fault Code	Level	Description	Cause	Remedy
0x01919009	Information	Cyclic data communication fault cleared	/	/
0x0191900A	Information	Cyclic data communication timeout fault cleared	/	/
0x0191900B	Information	Configuration conflict fault cleared	/	/
0x0191900C	Information	Stopper plate fault cleared	/	/
0x0191900D	Information	Module watchdog timeout cleared	/	/
0x0191900E	Information	IP address conflict cleared	/	/
0x0191900F	Information	The fault "The I/O module does not support the currently selected industry standard" has been cleared.	/	/
0x01919010	Information	Acyclic data communication fault cleared	/	/
0x01919011	Information	Acyclic data communication timeout fault cleared	/	/
0x01900000	Information	Connected successfully	/	/
0x01900001	Critical	CIP Safety connection failed. The target returned an error message.	1. The target node is different. 2. Connection parameters are abnormal. 3. The target node is abnormal.	1. Check if the IP address is correctly set. 2. Check if the configuration in iFA matches the actual device. 3. Check the specific fault code and modify the faulty communication parameters. 4. Check if the target device is operating normally. 5. Check if the project UNID matches the device UNID. If not, reset the UNID.

Fault Code	Level	Description	Cause	Remedy
0x01900002	Critical	CIP Safety connection timeout. The target node does not exist.	1. The network cable is disconnected. 2. The target node address is incorrectly set (IP, slot number). 3. The target node is faulty.	1. Check the link stability. 2. Check if the IP address is correctly set. 3. Check if the configuration in iFA matches the actual device. 4. Check if the target device is operating normally.
0x01900003	Critical	Communication timeout on the CIP Safety originator	1. The network cable is disconnected. 2. The communication timeout is set too short. 3. Excessive interference occurred. 4. The target device has entered an abnormal state.	1. Check the link stability. 2. Increase the timeout time. 3. Check if the target device is working normally.
0x01900004	Critical	Communication timeout on the CIP Safety destination	1. The network cable is disconnected. 2. The communication timeout is set too short. 3. Excessive interference occurred. 4. The target device has entered an abnormal state.	1. Check the link stability. 2. Increase the timeout time. 3. Check if the target device is working normally.
0x01900005	Critical	The IP address changes during device operation.	1. The IP address knob switch was adjusted during operation.	1. Set the IP address knob switch back to its original position. 2. Power cycle the device.
0xA0BF0070	Warning	The I/O module types between the master and slaves do not match.	The I/O module types between the master and slaves do not match.	Change the I/O configuration modules of the master and slave stations to make them consistent.

Fault Code	Level	Description	Cause	Remedy
0xA0BF0071	Warning	The input data slot length of the corresponding modules between the master and slaves does not match.	The input data slot length of the corresponding modules between the master and slaves does not match.	Change the I/O configuration modules of the master and slave stations to make them consistent. Overwritten by error 0X70 (higher priority).
0xA0BF0072	Warning	The output data slot length of the corresponding modules between the master and slaves does not match.	The output data slot length of the corresponding modules between the master and slaves does not match.	Change the I/O configuration modules of the master and slave stations to make them consistent. Overwritten by error 0X70 (higher priority).
0xA0BF0080	Warning	The PN master PLC is in stop status.	The PN master PLC is in stop status.	Switch the master PLC to run mode.
0xA0BF0081	Warning	The network between the PN master and slaves is not connected.	The network between the PN master and slaves is not connected.	Check whether the network ports between the PN master and slaves are properly connected.
0x9CA50000	Information	No fault	No fault	No fault
0x9CA50001	Critical	CANopen master configuration error	CANOpen configuration error	1. Check if CANopen is configured correctly. 2. Check the PDO COB-IDs for conflicts.
0x9CA50010	Warning	CANopen slave offline	The CANopen slave is offline.	1. Check whether the baud rate of the CAN bus device matches the configured value. 2. Check if the CAN bus wiring is abnormal, and if the termination resistors are consistent. 3. Check if the CANopen slave number matches the configured number.
0x00500001	Information	No fault	No fault	/
0x00500002	Critical	The master device continuously loses frames.	Unable to receive WTSN frames.	Check the IFA online interface and PLC logs. The network administrator can locate the specific error according to the diagnostic information.

Fault Code	Level	Description	Cause	Remedy
0x00500003	Critical	The network card has no connection.	No LINK signal from the network card	Check whether the network cable is properly inserted and securely connected, and whether all slave devices are powered normally.
0x00500004	Warning	The master device occasionally loses frames.	The link quality is affected by interference.	/
0x00500201	Critical	AcID configuration failure	AcID configuration error in multiple masters.	/
0x00500202	Warning	Slave station number mismatch	The number of online slaves is greater than/less than that of configured slaves.	Scan devices on the master station or compare device configurations, and check whether the configured number of slave stations matches the number of online slave stations.
0x00500203	Critical	Invalid configuration parameters of the master station	The configuration file (such as XML or NI) may contain syntax errors, parameter values out of range, or mismatches with the actual hardware.	Check the configuration parameters in the configuration file.
0x00501001	Critical	Failed to configure the master.	A problem occurred during master station initialization, possibly due to insufficient system resources, driver issues, or configuration file errors.	/
0x00510001	Information	No error	No fault	/
0x00510002	Warning	Unspecified error	An error is indicated on the device, but no specific details are available.	/

Fault Code	Level	Description	Cause	Remedy
0x00510003	Critical	The slave is disconnected.	Slave network cable disconnected	/
0x00510004	Warning	Type mismatch	Vendor ID, product code, or version number mismatch	/
0x00510005	Warning	The PS0 and PS1 connections are reversed.	The PS0 and PS1 ports on the slave station are reversed.	/
0x00510006	Warning	PS0 port disconnected.	The PS0 port on the slave station is disconnected.	/
0x00510007	Warning	PS1 port disconnected.	The PS1 port on the slave station is disconnected.	/
0x00510008	Warning	PT0 port disconnected.	The PT0 port on the slave station is disconnected.	/
0x00510009	Warning	PT1 port disconnected.	The PT1 port on the slave station is disconnected.	/
0x0051000a	Warning	PT2 port disconnected.	The PT2 port on the slave station is disconnected.	/
0x0051000b	Warning	PT3 port disconnected.	The PT3 port on the slave station is disconnected.	/
0x0051000c	Warning	Watchdog timeout	Watchdog timeout	/
0x0051000d	Warning	Module type mismatch	The hardware component has been inserted, but the type of this component does not match that of the configured module.	/

Fault Code	Level	Description	Cause	Remedy
0x0051000f	Warning	New module inserted	A new hardware component has been inserted, and the type of this component matches that of the configured module.	/
0x00510010	Critical	Data rate change	Data rate change	/
0x00510101	Critical	No memory	Less hardware memory, A-Device needs more memory.	Download a smaller file.
0x00510102	Critical	Incorrect FCS	FCS error detected	/
0x00510103	Critical	WTSn frame encoding error	A WTSn encoding error has been detected.	/
0x00510104	Critical	AnID not set	AnID has not been set before the device is operated.	/
0x00510105	Critical	AnID changed	AnID changed	/
0x00510201	Critical	Invalid requested state change	Invalid AR state transition: the state transition instruction violates the state mechanism, such as attempting to transition directly from the Init state to the OP state.	/
0x00510202	Critical	State error	The requested state is unknown or invalid.	/
0x00510203	Critical	Status switchover timeout	Status switchover timeout	/
0x00510204	Critical	SE configuration error	SE configuration error	/
0x00510205	Critical	SE transmission timeout	SE transmission timeout	/
0x00510206	Critical	Management controller change	Management controller change	/

Fault Code	Level	Description	Cause	Remedy
0x00510301	Critical	Configuration error	The same device is configured under multiple controllers.	/
0x00510302	Critical	Invalid DTNA address	Invalid DTNA address	/
0x00510303	Critical	TNA parameter change	TNA parameter change	/
0x00510304	Critical	Local subnet parameter change	Local subnet parameter change	/
0x00510401	Critical	Reference clock change	Reference clock change	/
0x00510402	Critical	Node desynchronization	Node desynchronization	/
0x00510403	Critical	Large time deviation	Large deviation from reference time	/
0x00510404	Critical	Unexpected time deviation	Unexpected deviation from reference time or neighbor time	/
0x00510405	Critical	Severe time deviation	Severe deviation from reference time or neighbor time	/
0x00510406	Critical	Synchronization error	Network jitter caused the slave to lose synchronization.	/
0x00510407	Critical	Sync manager watchdog	The time since the slave last received cyclic data has exceeded the watchdog timeout.	/
0x00510408	Critical	Fatal SYNC error	MCU did not receive the SYNC hardware interruption.	/
0x00511001	Critical	Invalid Output Configuration	The configured output process data is invalid.	/

Fault Code	Level	Description	Cause	Remedy
0x00511002	Critical	Invalid Input Configuration	The configured input process data is invalid.	/
0x00511003	Critical	Invalid Device Setup	Invalid state transition sequence (for example, attempting to transition directly from INIT to OP without passing through PRM).	/
0x00511004	Critical	Invalid Sync Cycle Time	The configured Cycle Time is not supported.	/
0x00511301	Critical	Segmentation fault	/	/
0x9CA60000	Information	No fault	No fault	No fault
0x9CA60001	Warning	CANopen slave offline	The CANopen slave is offline.	1. Check whether the baud rate of the CAN bus device matches the configured value. 2. Check if the CAN bus wiring is abnormal, and if the termination resistors are consistent. 3. Check if the CANopen slave number matches the configured number.
0x019F0000	Information	Connected successfully	/	/
0x019F0001	Critical	Unexpected FSoE instruction fault (master)	In the current state of FSoE communication, this command must not appear.	1. Check whether the instruction sent by the safety slave is incorrect. 2. Check whether the instruction field of the safety data frame is accidentally modified.
0x019F0002	Critical	FSoE unknown instruction (master)	An undefined instruction was detected in FSoE communication.	1. Check whether the instruction sent by the safety slave is incorrect. 2. Check whether the instruction field of the safety data frame is accidentally modified.

Fault Code	Level	Description	Cause	Remedy
0x019F0003	Critical	Invalid FSoE connection ID (master)	The ID of the connection established by the FSoE does not match the ID received.	1. Check whether the connection ID sent by the safety slave is incorrect. 2. Check whether the safety data field in the safety data frame has been accidentally modified in the connection state.
0x019F0004	Critical	FSoE CRC error (master)	The CRC value of the FSoE data frame does not match the calculated value.	1. Check whether the CRC sent by the safety slave is incorrect. 2. Check whether the CRC field of the safety data frame is accidentally modified.
0x019F0005	Critical	FSoE watchdog timeout (master)	No data frame was received within the FSoE communication cycle.	1. Check whether the safety slave is sending safety data frames. 2. Check whether the watchdog time setting is proper. 3. Check whether the cable is disconnected.
0x019F0006	Critical	Invalid FSoE slave address (master)	1. The FSoE communication address setting is out of range. 2. The address configured on the FSoE master does not match the address configured on the slave.	1. Check whether the slave address sent by the safety slave is correct. 2. Check whether the slave address of the safety slave is configured correctly. 3. Check whether the safety data field in the safety data frame has been accidentally modified in the connection state.
0x019F0007	Critical	Invalid FSoE safety data (master)	The FSoE communication parameters are incorrect or the connection fails. In this case, the safety data is invalid.	1. Check whether the data sent by the safety slave in the connection state or parameter state is incorrect. 2. Check whether the safety data field of the safety data frame has been accidentally modified in the connection state or parameter state.

Fault Code	Level	Description	Cause	Remedy
0x019F0008	Critical	Invalid FSoE communication parameter length (master)	The length and value of the FSoE communication parameters are out of the specified range.	1. Check whether the communication parameter data length sent by the safety slave is incorrect in the parameter state. 2. Check whether the safety data field of the safety data frame is accidentally modified in the parameter state.
0x019F0009	Critical	Invalid FSoE communication parameter data (master)	FSoE communication parameter data is out of range.	1. Check whether the communication parameter data sent by the safety slave is incorrect in the parameter state. 2. Check whether the safety data field of the safety data frame is accidentally modified in the parameter state.
0x019F000A	Critical	Invalid FSoE application parameter length (master)	The length of the FSoE communication application parameters does not match the preset value.	1. Check whether the application parameter data length sent by the safety slave is incorrect in the parameter state. 2. Check whether the safety data field of the safety data frame is accidentally modified in the parameter state.
0x019F000B	Critical	Invalid FSoE application parameter data (master)	The FSoE communication application program parameter data is out of the range.	1. Check whether the application parameter data sent by the safety slave is incorrect in the parameter state. 2. Check whether the safety data field of the safety data frame is accidentally modified in the parameter state.
0x019F00FF	Critical	Unknown fault (master)	1. Instance number out of range.	1. Check whether the connected device exceeds the specifications.

Fault Code	Level	Description	Cause	Remedy
0x019F2001	Critical	Unexpected FSoE instruction fault (slave)	In the current state of FSoE communication, this command must not appear.	1. Check whether the instruction sent by the safety master is incorrect. 2. Check whether the instruction field of the safety data frame is accidentally modified.
0x019F2002	Critical	FSoE unknown instruction (slave)	An undefined instruction was detected in FSoE communication.	1. Check whether the instruction sent by the safety master is incorrect. 2. Check whether the instruction field of the safety data frame is accidentally modified.
0x019F2003	Critical	Invalid FSoE connection ID (slave)	The ID of the connection established by the FSoE does not match the ID received.	1. Check whether the connection ID sent by the safety master is incorrect. 2. Check whether the safety data field in the safety data frame has been accidentally modified.
0x019F2004	Critical	FSoE CRC error (slave)	The CRC value of the FSoE data frame does not match the calculated value.	1. Check whether the CRC sent by the safety master is incorrect. 2. Check whether the CRC field of the safety data frame is accidentally modified.
0x019F2005	Critical	FSoE watchdog timeout (slave)	No data frame was received within the FSoE communication cycle.	1. Check whether the safety master is sending data. 2. Check whether the watchdog time setting is proper. 3. Check whether the cable is disconnected.
0x019F2006	Critical	Invalid FSoE slave address (slave)	1. The FSoE communication address setting is out of range. 2. The address configured on the FSoE master does not match the address configured on the slave.	1. Check whether the slave address sent by the safety master is correct. 2. Check whether the slave address of the safety slave is configured correctly. 3. Check whether the safety data field in the safety data frame has been accidentally modified in the connection state.

Fault Code	Level	Description	Cause	Remedy
0x019F2007	Critical	Invalid FSoE safety data (slave)	The FSoE communication parameters are incorrect or the connection fails. In this case, the safety data is invalid.	1. Check whether the data sent by the safety master in the connection state or parameter state is incorrect. 2. Check whether the safety data field of the safety data frame has been accidentally modified in the connection state or parameter state.
0x019F2008	Critical	Invalid FSoE communication parameter length (slave)	The length and value of the FSoE communication parameters are out of the specified range.	1. Check whether the communication parameter data length sent by the safety master is incorrect in the parameter state. 2. Check whether the safety data field of the safety data frame is accidentally modified in the parameter state.
0x019F2009	Critical	Invalid FSoE communication parameter data (slave)	FSoE communication parameter data is out of range.	1. Check whether the communication parameter data sent by the safety master is incorrect in the parameter state. 2. Check whether the safety data field of the safety data frame is accidentally modified in the parameter state.
0x019F200A	Critical	Invalid FSoE application parameter length (slave)	The length of the FSoE communication application parameters does not match the preset value.	1. Check whether the application parameter data length sent by the safety master is incorrect in the parameter state. 2. Check whether the safety data field of the safety data frame is accidentally modified in the parameter state.
0x019F200B	Critical	Invalid FSoE application parameter data (slave)	The FSoE communication application program parameter data is out of the range.	1. Check whether the application parameter data sent by the safety master is incorrect in the parameter state. 2. Check whether the safety data field of the safety data frame is accidentally modified in the parameter state.

Fault Code	Level	Description	Cause	Remedy
0x019F20FF	Critical	Unknown fault (slave)	1. The first byte of the safety data was accidentally modified in the reset state. 2. The instance number is out of range.	1. Check whether the safety data field in the safety data frame in the reset state has been accidentally modified. 2. Check whether the connected device exceeds the specifications.
0x019F0040	Information	Disconnected	/	/
0xA7F80001	Information	Communication module online	The communication module is operating normally.	/
0xA7F80002	Critical	Communication module offline	The communication module is not inserted, or the communication module is operating abnormally	Check the corresponding slot, and insert a matching communication module.
0xA7F80003	Critical	Communication module mismatch	The inserted communication module does not match the configured module.	Check the corresponding slot, and replace a matching communication module.
0xA7F80004	Critical	Communication module fault.	Communication module operation fault.	Check the fault code in the additional information, then refer to the relevant manual for a detailed description of the fault code.
0xA7F80005	Information	Communication module fault cleared	The communication module is operating normally.	/
0xA7F80006	Critical	Master station communication fault on the communication module.	Master station communication fault on the PN/DP module.	Check the fault code in the additional information, then refer to the relevant manual for a detailed description of the fault code.

Fault Code	Level	Description	Cause	Remedy
0xA7F80007	Information	The master station communication on the communication module has returned to normal.	The master station communication on the PN/DP module has returned to normal.	/
0xA7F80008	Critical	Slave network connection fault status.	Slave connection fault status on the communication module.	Check the fault code in the additional information, then refer to the relevant manual for a detailed description of the fault code.
0xA7F80009	Information	The slave network connection is normal.	The slave network connection on the communication module is normal.	/
0xA7F8000A	Warning	Slave network connection diagnostic status.	Slave connection diagnostic status on the communication module.	A diagnostic event has occurred on a slave. Clear the diagnostic information for the corresponding slave.
0xA7F8000B	Critical	Slave not recognized on network	A slave has not been recognized on the communication module.	A slave has not been recognized. Configure the correct slave and connect it to the communication network.
0xA7F8000C	Warning	The master station of the communication module is not configured with a slave module.	The master station of the communication module is not configured with a slave module.	Configure the correct slave for the corresponding master module, and connect it to the communication network.
0xA7F8000D	Warning	Communication module unable to receive diagnostic information	The communication module cannot receive diagnostic information.	Check the indicators of the expansion module, and power cycle the module. If the module still cannot operate normally after power cycling, send it back to the factory for repair.
0xA7F8000E	Critical	Expansion communication module not properly seated	The expansion communication module is not seated properly.	Power off the device, reseat the module in the corresponding slot securely, then power it on.

21.2 Appendix 2: Version Compatibility Table

The following table shows the compatibility of software version and device firmware version.

> NOTE
>
> The software version is backward compatible with device firmware versions. Select the corresponding firmware version as needed.

Compatibility table of controller firmware and module

Module Series	Module Type	Product Series and Firmware Versions											
		EVO4000-V1.5	EVO5000-V1.2	EVO5000-V1.3	EVO5000-V1.4	EVO5000-V1.5	EVO7000-V1.1	EVO7000-V1.4	EVO7000-V1.5	EVO8000-V1.0	EVO8000-V1.1	EVO1000EF-V1.0	APC5000-V1.4
GL10 series	GL10-1600END	√	√	√	√	√	√	√	√	×	×	×	√
	GL10-3200END	√	√	√	√	√	√	√	√	×	×	×	√
	GL10-0016ER	√	√	√	√	√	√	√	√	×	×	×	√
	GL10-0016ETP	√	√	√	√	√	√	√	√	×	×	×	√
	GL10-0016ETN	√	√	√	√	√	√	√	√	×	×	×	√
	GL10-0032ETN	√	√	√	√	√	√	√	√	×	×	×	√
	GL10-4AD	√	√	√	√	√	√	√	√	×	×	×	√
	GL10-4DA	√	√	√	√	√	√	√	√	×	×	×	√
	GL10-4PM	√	√	√	√	√	√	√	√	×	×	×	√
	GL10-2PH	√	√	√	√	√	√	√	√	×	×	×	√

Module Series	Module Type	Product Series and Firmware Versions											
		EVO4000-V1.5	EVO5000-V1.2	EVO5000-V1.3	EVO5000-V1.4	EVO5000-V1.5	EVO7000-V1.1	EVO7000-V1.4	EVO7000-V1.5	EVO8000-V1.0	EVO8000-V1.1	EVO1000EF-V1.0	APC5000-V1.4
	GL10-4PT	√	√	√	√	√	√	√	√	×	×	×	√
	GL10-4TC	√	√	√	√	√	√	√	√	×	×	×	√
	GL10-8TC	√	√	√	√	√	√	√	√	×	×	×	√
	GL10-PS2	√	√	√	√	√	√	√	√	×	×	×	√
	GL10-RTU-ECTA	√	√	√	√	√	√	√	√	×	×	×	√
GR10 series	GR10-0808ETNE	√	√	√	√	√	√	√	√	×	×	×	√
	GR10-1616ETNE	√	√	√	√	√	√	√	√	×	×	×	√
	GR10-4ADE	√	√	√	√	√	√	√	√	×	×	×	√
	GR10-4DAE	√	√	√	√	√	√	√	√	×	×	×	√
	GR10-4PME	√	√	√	√	√	√	√	√	×	×	×	√
	GR10-2HCE	√	√	√	√	√	√	√	√	×	×	×	√
	GR10-2PHE	√	√	√	√	√	√	√	√	×	×	×	√
	GR10-8TCE	√	√	√	√	√	√	√	√	×	×	×	√
	GR10-8PBE	√	√	√	√	√	√	√	√	×	×	×	√
	GR10-EC-3SW	√	√	√	√	√	√	√	√	×	×	×	√
	GR10-EC-6SW	√	√	√	√	√	√	√	√	×	×	×	√

Module Series	Module Type	Product Series and Firmware Versions											
		EVO400-V1.5	EVO500-V1.2	EVO500-V1.3	EVO500-V1.4	EVO500-V1.5	EVO700-V1.1	EVO700-V1.4	EVO700-V1.5	EVO800-V1.0	EVO800-V1.1	EVO1000EF-V1.0	APC500-V1.4
	GR10-4TCS-PID	√	√	√	√	√	√	√	√	×	×	×	√
	GR10-1616ERE-BD	√	√	√	√	√	√	√	√	×	×	×	√
	GR10-32ETP8AME	√	√	√	√	√	√	√	√	×	×	×	√
	GR10-28ETP12AME	√	√	√	√	√	√	√	√	×	×	×	√
	GR10-4ETP4ADE	√	√	√	√	√	√	√	√	×	×	×	√
GL20 series	GL20(GL20S)-1600END	√	√	√	√	√	√	√	√	√	√	×	√
	GL20(GL20S)-3200END	√	√	√	√	√	√	√	√	√	√	×	√
	GL20(GL20S)-0800END	√	√	√	√	√	√	√	√	√	√	×	√
	GL20(GL20S)-0016ETN	√	√	√	√	√	√	√	√	√	√	×	√

Module Series	Module Type	Product Series and Firmware Versions											
		EVO400-V1.5	EVO500-V1.2	EVO500-V1.3	EVO500-V1.4	EVO500-V1.5	EVO700-V1.1	EVO700-V1.4	EVO700-V1.5	EVO800-V1.0	EVO800-V1.1	EVO1000EF-V1.0	APC500-V1.4
	GL20(GL20S)-0032ETN	√	√	√	√	√	√	√	√	√	√	×	√
	GL20(GL20S)-0008ETN	√	√	√	√	√	√	√	√	√	√	×	√
	GL20(GL20S)-0016ETP	√	√	√	√	√	√	√	√	√	√	×	√
	GL20(GL20S)-0008ETP	√	√	√	√	√	√	√	√	√	√	×	√
	GL20(GL20S)-0808ETN	√	√	√	√	√	√	√	√	√	√	×	√
	GL20(GL20S)-0008ER	√	√	√	√	√	√	√	√	√	√	×	√
	GL20(GL20S)-4PT	√	√	√	√	√	√	√	√	√	√	×	√
	GL20(GL20S)-4TC	√	√	√	√	√	√	√	√	√	√	×	√

Module Series	Module Type	Product Series and Firmware Versions											
		EVO400-V1.5	EVO500-V1.2	EVO500-V1.3	EVO500-V1.4	EVO500-V1.5	EVO700-V1.1	EVO700-V1.4	EVO700-V1.5	EVO800-V1.0	EVO800-V1.1	EVO1000EF-V1.0	APC500-V1.4
	GL20 (GL20S)-3232ETN	√	√	√	√	√	√	√	√	√	√	×	√
	GL20 (GL20S)-4AD	√	√	√	√	√	√	√	√	√	√	×	√
	GL20 (GL20S)-4DA	√	√	√	√	√	√	√	√	√	√	×	√
	GL20 (GL20S)-PS2	√	√	√	√	√	√	√	√	√	√	×	√
	GL20 (GL20S)-1600END-5V (same configuration parameters as 1600END)	√	×	√	√	√	√	√	√	√	√	×	√
	GL20 (GL20S)-0004ER	√	×	√	√	√	√	√	√	√	√	×	√
	GL20 (GL20S)-0404ETP-5V	√	×	√	√	√	√	√	√	√	√	×	√

Module Series	Module Type	Product Series and Firmware Versions											
		EVO400-V1.5	EVO500-V1.2	EVO500-V1.3	EVO500-V1.4	EVO500-V1.5	EVO700-V1.1	EVO700-V1.4	EVO700-V1.5	EVO800-V1.0	EVO800-V1.1	EVO1000EF-V1.0	APC500-V1.4
	GL20(GL20S)-0004ETP-2A	√	×	√	√	√	√	√	√	√	√	×	√
	GL20(GL20S)-8ADV	√	×	√	√	√	√	√	√	√	√	×	√
	GL20(GL20S)-8ADI	√	×	√	√	√	√	√	√	√	√	×	√
	GL20(GL20S)-8DAI	√	×	√	√	√	√	√	√	√	√	×	√
	GL20(GL20S)-8DAV	√	×	√	√	√	√	√	√	√	√	×	√
	GL20-2SSI	√	×	√	√	√	√	√	√	√	√	×	√
	GL20-2HC	√	×	√	√	√	√	√	√	√	√	×	√
	GL20-2SCOM-MDB	√	×	×	√	√	×	√	√	×	√	×	√
	GL20-2S485-MDB	√	×	×	√	√	×	√	√	×	√	×	√
	GL20-0800ENA	√	×	×	√	√	×	√	√	×	√	×	√
	GL20-0016ETP-5V	√	×	×	√	√	×	√	√	×	√	×	√

Module Series	Module Type	Product Series and Firmware Versions											
		EVO400-V1.5	EVO500-V1.2	EVO500-V1.3	EVO500-V1.4	EVO500-V1.5	EVO700-V1.1	EVO700-V1.4	EVO700-V1.5	EVO800-V1.0	EVO800-V1.1	EVO1000EF-V1.0	APC500-V1.4
	GL20-0032ETP	√	×	×	√	√	×	√	√	×	√	×	√
	GL20-4TC-ISO	√	×	×	√	√	×	√	√	×	√	×	√
	GL20-4PT-ISO	√	×	×	√	√	×	√	√	×	√	×	√
	GL20-8TC-PID	√	×	×	√	√	×	√	√	×	√	×	√
	GL20-4LC-PID	√	×	×	√	√	×	√	√	×	√	×	√
GR20T	GR20T-ECT-1616EMN	√	×	×	√	√	×	√	√	×	×	×	√
	GR20T-ECT-0808EMN	√	×	×	√	√	×	√	√	×	×	×	√
	GR20T-ECT-1616EMN-E	√	×	×	√	√	×	√	√	×	×	×	√
	GR20T-ECT-1600END	√	×	×	√	√	×	√	√	×	×	×	√
	GR20T-ECT-3200END	√	×	×	√	√	×	√	√	×	×	×	√

Module Series	Module Type	Product Series and Firmware Versions											
		EVO400-V1.5	EVO500-V1.2	EVO500-V1.3	EVO500-V1.4	EVO500-V1.5	EVO700-V1.1	EVO700-V1.4	EVO700-V1.5	EVO800-V1.0	EVO800-V1.1	EVO1000EF-V1.0	APC500-V1.4
	GR20T-ECT-0016ETP	√	×	×	√	√	×	√	√	×	×	×	√
	GR20T-ECT-0032ETP	√	×	×	√	√	×	√	√	×	×	×	√
	GR20T-ECT-0016ETN	√	×	×	√	√	×	√	√	×	×	×	√
	GR20T-ECT-0032ETN	√	×	×	√	√	×	√	√	×	×	×	√
GS20	GS20-ECT-8L	√	√	√	√	√	√	√	√	×	×	×	√
GL20 coupler	GL20-EIP coupler	√	×	√	√	√	√	√	√	√	√	×	√
	GL20-ECT coupler	√	√	√	√	√	√	√	√	√	√	×	√
	GL20-Hi80-EIP coupler	×	×	×	×	×	×	×	×	√	√	×	×
HI80 series	Hi80-RTU-EIP	×	×	×	×	×	×	×	×	√	√	×	×

Module Series	Module Type	Product Series and Firmware Versions											
		EVO4000-V1.5	EVO5000-V1.2	EVO5000-V1.3	EVO5000-V1.4	EVO5000-V1.5	EVO7000-V1.1	EVO7000-V1.4	EVO7000-V1.5	EVO8000-V1.0	EVO8000-V1.1	EVO1000EF-V1.0	APC5000-V1.4
	Hi80-1600ENP	×	×	×	×	×	×	×	×	√	√	×	×
	Hi80-0016ETP	×	×	×	×	×	×	×	×	√	√	×	×
	Hi80-8ADV	×	×	×	×	×	×	×	×	√	√	×	×
	Hi80-8DAV	×	×	×	×	×	×	×	×	√	√	×	×
	Hi80-8ADI	×	×	×	×	×	×	×	×	√	√	×	×
	Hi80-8DAI	×	×	×	×	×	×	×	×	√	√	×	×
	Hi80-4LC	×	×	×	×	×	×	×	×	√	√	×	×
	Hi80-8PS-000	×	×	×	×	×	×	×	×	√	√	×	×
	Hi80-8EM	×	×	×	×	×	×	×	×	√	√	×	×
	Hi80-11EM	×	×	×	×	×	×	×	×	√	√	×	×
	Hi80-14EM	×	×	×	×	×	×	×	×	√	√	×	×
	Hi80-18EM	×	×	×	×	×	×	×	×	√	√	×	×
	Hi80-2ENC	×	×	×	×	×	×	×	×	×	√	×	×
	Hi80-4S485	×	×	×	×	×	×	×	×	×	√	×	×
HI50 series	Hi50-RTU-EIP	×	×	×	×	×	×	×	×	×	×	√	×
	Hi50-0800END-F00	×	×	×	×	×	×	×	×	×	×	√	×

Module Series	Module Type	Product Series and Firmware Versions											
		EVO4000-V1.5	EVO5000-V1.2	EVO5000-V1.3	EVO5000-V1.4	EVO5000-V1.5	EVO7000-V1.1	EVO7000-V1.4	EVO7000-V1.5	EVO8000-V1.0	EVO8000-V1.1	EVO1000EF-V1.0	APC5000-V1.4
	Hi50-0008ETP-F00	×	×	×	×	×	×	×	×	×	×	√	×
	Hi50-0004ETP-F2A	×	×	×	×	×	×	×	×	×	×	√	×
	Hi50-3ATT-000	×	×	×	×	×	×	×	×	×	×	√	×
	Hi50-4ADI-F00	×	×	×	×	×	×	×	×	×	×	√	×
	Hi50-4ADV-F00	×	×	×	×	×	×	×	×	×	×	√	×
	Hi50-4TC-F00	×	×	×	×	×	×	×	×	×	×	√	×
	Hi50-4PT-F00	×	×	×	×	×	×	×	×	×	×	√	×
GE20 series	GE20-4DO-TN	√	√	√	√	√	√	√	√	×	×	×	√
	GE20-4DI	√	√	√	√	√	√	√	√	×	×	×	√
	GE20-2AD1DA-V	√	√	√	√	√	√	√	√	×	×	×	√
	GE20-2AD1DA-I	√	√	√	√	√	√	√	√	×	×	×	√
	GE20-232/485-RTC	√	√	√	√	√	√	√	√	×	×	×	√

Module Series	Module Type	Product Series and Firmware Versions											
		EVO400-V1.5	EVO500-V1.2	EVO500-V1.3	EVO500-V1.4	EVO500-V1.5	EVO700-V1.1	EVO700-V1.4	EVO700-V1.5	EVO800-V1.0	EVO800-V1.1	EVO1000EF-V1.0	APC500-V1.4
	GE20-232/485	√	√	√	√	√	√	√	√	×	×	×	√
	GE20-RTC	√	√	√	√	√	√	√	√	×	×	×	√
	GE20-TF	√	√	√	√	√	√	√	√	×	×	×	√
EIP slave	AC700 series PLC EIP adapter	√	√	√	√	√	√	√	√	√	√	√	√
	AC800 series PLC EIP adapter	√	√	√	√	√	√	√	√	√	√	√	√
	AM700 series PLC EIP adapter	√	√	√	√	√	√	√	√	√	√	√	√
	EIP-DP	√	√	√	√	√	√	√	√	√	√	√	√
	EIP-PN	√	√	√	√	√	√	√	√	√	√	√	√

Compatibility table of iFA software version and device firmware

Product Type	Product Series	Firmware Versions Compatible with the iFA Software Version			
		V1.3.4.5	V1.4.2.5	V1.4.3.5	V1.4.4.5
Controller	EVO400 series	×	×	×	V1.5

Product Type	Product Series	Firmware Versions Compatible with the iFA Software Version			
		V1.3.4.5	V1.4.2.5	V1.4.3.5	V1.4.4.5
	EVO500 series	V1.2	V1.2 V1.3	V1.2 V1.3 V1.4	V1.2 V1.3 V1.4 V1.5
	EVO700 series	×	V1.1	V1.1 V1.4	V1.1 V1.4 V1.5
	EVO700C series	×	×	×	V1.5
	EVO800 series	×	V1.0	V1.0	V1.0 V1.1
	EVO1000EF series	×	V1.0	V1.0	V1.0
	APC500 series	×	×	V1.4	V1.4 V1.5
	APC760C series	×	×	×	V1.5
GL10 series	GL10-1600END	√	√	√	√
	GL10-3200END	√	√	√	√
	GL10-0016ER	√	√	√	√
	GL10-0016ETP	√	√	√	√
	GL10-0016ETN	√	√	√	√
	GL10-0032ETN	√	√	√	√
	GL10-4AD	√	√	√	√
	GL10-4DA	√	√	√	√
	GL10-4PT	√	√	√	√
	GL10-4TC	√	√	√	√
	GL10-8TC	√	√	√	√
	GL10-PS2	√	√	√	√
	GL10-RTU-ECTA	√	√	√	√
GR10 series	GR10-0808ETNE	√	√	√	√
	GR10-1616ETNE	√	√	√	√
	GR10-4ADE	√	√	√	√
	GR10-4DAE	√	√	√	√
	GR10-4PME	√	√	√	√

Product Type	Product Series	Firmware Versions Compatible with the iFA Software Version			
		V1.3.4.5	V1.4.2.5	V1.4.3.5	V1.4.4.5
	GR10-2HCE	√	√	√	√
	GR10-2PHE	√	√	√	√
	GR10-8TCE	√	√	√	√
	GR10-8PBE	√	√	√	√
	GR10-EC-3SW	√	√	√	√
	GR10-EC-6SW	√	√	√	√
	GR10-4TCS-PID	√	√	√	√
	GR10-1616ERE-BD	√	√	√	√
GL20 series	GL20(GL20S)-1600END	√	√	√	√
	GL20(GL20S)-3200END	√	√	√	√
	GL20(GL20S)-0800END	√	√	√	√
	GL20(GL20S)-0016ETN	√	√	√	√
	GL20(GL20S)-0032ETN	√	√	√	√
	GL20(GL20S)-0008ETN	√	√	√	√
	GL20(GL20S)-0016ETP	√	√	√	√
	GL20(GL20S)-0008ETP	√	√	√	√
	GL20(GL20S)-0808ETN	√	√	√	√
	GL20(GL20S)-0008ER	√	√	√	√
	GL20(GL20S)-4PT	√	√	√	√
	GL20(GL20S)-4TC	√	√	√	√
	GL20(GL20S)-3232ETN	√	√	√	√
	GL20(GL20S)-4AD	√	√	√	√

Product Type	Product Series	Firmware Versions Compatible with the iFA Software Version			
		V1.3.4.5	V1.4.2.5	V1.4.3.5	V1.4.4.5
	GL20(GL20S)-4DA	√	√	√	√
	GL20(GL20S)-PS2	√	√	√	√
	GL20 (GL20S)-1600END-5V (same configuration parameters as 1600END)	×	√	√	√
	GL20(GL20S)-0004ER	×	√	√	√
	GL20(GL20S)-0404ETP-5V	×	√	√	√
	GL20(GL20S)-0004ETP-2A	×	√	√	√
	GL20(GL20S)-8ADV	×	√	√	√
	GL20(GL20S)-8ADI	×	√	√	√
	GL20(GL20S)-8DAI	×	√	√	√
	GL20(GL20S)-8DAV	×	√	√	√
	GL20-2SSI	×	√	√	√
	GL20-2HC	×	√	√	√
	GL20-2SCOM-MDB	×	×	√	√
	GL20-2S485-MDB	×	×	√	√
	GL20-0800ENA	×	×	√	√
	GL20-0016ETP-5V	×	×	√	√
	GL20-0032ETP	×	×	√	√
	GL20-4TC-ISO	×	×	√	√
	GL20-4PT-ISO	×	×	√	√
	GL20-8TC-PID	×	×	√	√
	GL20-4LC-PID	×	×	√	√

Product Type	Product Series	Firmware Versions Compatible with the iFA Software Version			
		V1.3.4.5	V1.4.2.5	V1.4.3.5	V1.4.4.5
	GL20-1DM	×	×	×	√
	GL20-RTU-ETC32	×	×	√	√
GR20T series	GR20T-ECT-1616EMN	×	×	√	√
	GR20T-ECT-0808EMN	×	×	√	√
	GR20T-ECT-1616EMN-E	×	×	√	√
	GR20T-ECT-1600END	×	×	√	√
	GR20T-ECT-3200END	×	×	√	√
	GR20T-ECT-0016ETP	×	×	√	√
	GR20T-ECT-0032ETP	×	×	√	√
	GR20T-ECT-0016ETN	×	×	√	√
	GR20T-ECT-0032ETN	×	×	√	√
GS20 series	GS20-ECT-8L	√	√	√	√
GL20 coupler	GL20-EIP coupler	×	√	√	√
	GL20-ECT coupler	√	√	√	√
	GL20-Hi80-EIP coupler	×	√	√	√
HI80 series	Hi80-RTU-EIP	×	√	√	√
	Hi80-1600ENP	×	√	√	√
	Hi80-0016ETP	×	√	√	√
	Hi80-8ADV	×	√	√	√
	Hi80-8DAV	×	√	√	√
	Hi80-8ADI	×	√	√	√
	Hi80-8DAI	×	√	√	√
	Hi80-8LC	×	√	√	√
	Hi80-2PS	×	√	√	√
	Hi80-8EM	×	√	√	√

Product Type	Product Series	Firmware Versions Compatible with the iFA Software Version			
		V1.3.4.5	V1.4.2.5	V1.4.3.5	V1.4.4.5
	Hi80-11EM	×	√	√	√
	Hi80-14EM	×	√	√	√
	Hi80-18EM	×	√	√	√
HI50 series	Hi50-RTU-EIP	×	√	√	√
	Hi50-0800END-F00	×	√	√	√
	Hi50-0008ETP-F00	×	√	√	√
	Hi50-0004ETP-F2A	×	√	√	√
	Hi50-3ATT-000	×	√	√	√
	Hi50-4ADI-F00	×	√	√	√
	Hi50-4ADV-F00	×	√	√	√
	Hi50-4TC-F00	×	√	√	√
	Hi50-4PT-F00	×	√	√	√
GE20 series	GE20-4DO-TN	√	√	√	√
	GE20-4DI	√	√	√	√
	GE20-2AD1DA-V	√	√	√	√
	GE20-2AD1DA-I	√	√	√	√
	GE20-232/485-RTC	√	√	√	√
	GE20-232/485	√	√	√	√
	GE20-RTC	√	√	√	√
	GE20-TF	√	√	√	√
	GE20-TF-TRC	×	×	×	√
	GE20-CAN-485	×	√	√	√
EIP slave	AC700 series PLC EIP adapter	×	√	√	√
	AC800 series PLC EIP adapter	×	√	√	√
	AM300 series PLC EIP adapter	×	√	√	√
	AM500 series PLC EIP adapter	×	√	√	√
	AM700 series PLC EIP adapter	×	√	√	√

Product Type	Product Series	Firmware Versions Compatible with the iFA Software Version			
		V1.3.4.5	V1.4.2.5	V1.4.3.5	V1.4.4.5
	EVO500	×	√	√	√
	EVO700	×	√	√	√
	EVO760C	×	×	×	√
	EVO780C series	×	×	×	√
	EVO800	×	×	√	√
	IPT210-A-EIP-CM	×	×	×	√
	MD580	×	×	×	√
	MD880	×	×	×	√
	Generic EtherNetIP device	√	√	√	√
	EIP card	√	√	√	√
	Easy	√	√	√	√
	APC520	×	×	×	√
	APC760C	×	×	×	√

21.3 Appendix 3: PN/DP Master Diagnostic Codes

- PN master system diagnostic codes

Diagnostic Code	Description
0x00000000	HIL_SYS_SUCCESS Success
0x00000001	HIL_SYS_RAM_NOT_FOUND RAM Not Found
0x00000002	HIL_SYS_RAM_TYPE Invalid RAM Type
0x00000003	HIL_SYS_RAM_SIZE Invalid RAM Size
0x00000004	HIL_SYS_RAM_TEST Ram Test Failed
0x00000005	HIL_SYS_FLASH_NOT_FOUND Flash Not Found
0x00000006	HIL_SYS_FLASH_TYPE Invalid Flash Type
0x00000007	HIL_SYS_FLASH_SIZE Invalid Flash Size

Diagnostic Code	Description
0x00000008	HIL_SYS_FLASH_TEST Flash Test Failed
0x00000009	HIL_SYS_EEPROM_NOT_FOUND EEPROM Not Found
0x0000000A	HIL_SYS_EEPROM_TYPE Invalid EEPROM Type
0x0000000B	HIL_SYS_EEPROM_SIZE Invalid EEPROM Size
0x0000000C	HIL_SYS_EEPROM_TEST EEPROM Test Failed
0x0000000D	HIL_SYS_SECURE_EEPROM Security EEPROM Failure
0x0000000E	HIL_SYS_SECURE_EEPROM_NOT_INIT Security EEPROM Not Initialized
0x0000000F	HIL_SYS_FILE_SYSTEM_FAULT File System Fault
0x00000010	HIL_SYS_VERSION_CONFLICT Version Conflict
0x00000011	HIL_SYS_NOT_INITIALIZED System Task Not Initialized
0x00000012	HIL_SYS_MEM_ALLOC Memory Allocation Failed

- PN master communication diagnostic codes

Diagnostic Code	Description
0x00000000	SUCCESS_HIL_OK No error
0xC0000001	ERR_HIL_FAIL Common error, detailed error information optionally present in the data area of packet.
0xC0000002	ERR_HIL_UNEXPECTED Unexpected failure.
0xC0000003	ERR_HIL_OUTOFMEMORY Ran out of memory.
0xC0000004	ERR_HIL_UNKNOWN_COMMAND Unknown Command in Packet received.
0xC0000005	ERR_HIL_UNKNOWN_DESTINATION Unknown Destination in Packet received.
0xC0000006	ERR_HIL_UNKNOWN_DESTINATION_ID Unknown Destination Id in Packet received.

Diagnostic Code	Description
0xC0000007	ERR_HIL_INVALID_PACKET_LEN Packet length is invalid.
0xC0000008	ERR_HIL_INVALID_EXTENSION Invalid Extension in Packet received.
0xC0000009	ERR_HIL_INVALID_PARAMETER Invalid Parameter in Packet found.
0xC000000A	ERR_HIL_INVALID_ALIGNMENT Invalid alignment.
0xC000000C	ERR_HIL_WATCHDOG_TIMEOUT Watchdog error occurred.
0xC000000D	ERR_HIL_INVALID_LIST_TYPE List type is invalid.
0xC000000E	ERR_HIL_UNKNOWN_HANDLE Handle is unknown.
0xC000000F	ERR_HIL_PACKET_OUT_OF_SEQ A packet index has been not in the expected sequence.
0xC0000010	ERR_HIL_PACKET_OUT_OF_MEMORY The amount of fragmented data contained the packet sequence has been too large.
0xC0000011	ERR_HIL_QUE_PACKETDONE The packet done function has failed.
0xC0000012	ERR_HIL_QUE_SENDPACKET The sending of a packet has failed.
0xC0000013	ERR_HIL_POOL_PACKET_GET The request of a packet from packet pool has failed.
0xC0000014	ERR_HIL_POOL_PACKET_RELEASE The release of a packet-to-packet pool has failed.
0xC0000015	ERR_HIL_POOL_GET_LOAD The get packet pool load function has failed.
0xC0000016	ERR_HIL_QUE_GET_LOAD The get queue load function has failed.
0xC0000017	ERR_HIL_QUE_WAITFORPACKET The waiting for a packet from queue has failed.
0xC0000018	ERR_HIL_QUE_POSTPACKET The posting of a packet has failed.
0xC0000019	ERR_HIL_QUE_PEEKPACKET Peeking a packet from queue has failed.
0xC000001A	ERR_HIL_REQUEST_RUNNING Request is already running.

Diagnostic Code	Description
0xC000001B	ERR_HIL_CREATE_TIMER Creating a timer failed.
0xC000001C	ERR_HIL_BUFFER_TOO_SHORT Supplied buffer too short for the data.
0xC000001D	ERR_HIL_NAME_ALREADY_EXIST Supplied name already exists.
0xC000001E	ERR_HIL_PACKET_FRAGMENTATION_TIMEOUT The packet fragmentation has timed out.
0xC0000100	ERR_HIL_INIT_FAULT General initialization fault.
0xC0000101	ERR_HIL_DATABASE_ACCESS_FAILED Database access failure.
0xC0000102	ERR_HIL_CIR_MASTER_PARAMETER_FAILED Master parameter cannot activated at state operate.
0xC0000103	ERR_HIL_CIR_SLAVE_PARAMTER_FAILED Slave parameter cannot be activated at state operate.
0xC0000119	ERR_HIL_NOT_CONFIGURED Configuration not available.
0xC0000120	ERR_HIL_CONFIGURATION_FAULT General configuration fault.
0xC0000121	ERR_HIL_INCONSISTENT_DATA_SET Inconsistent configuration data.
0xC0000122	ERR_HIL_DATA_SET_MISMATCH Configuration data set mismatch.
0xC0000123	ERR_HIL_INSUFFICIENT_LICENSE Insufficient license.
0xC0000124	ERR_HIL_PARAMETER_ERROR Parameter error.
0xC0000125	ERR_HIL_INVALID_NETWORK_ADDRESS Network address invalid.
0xC0000126	ERR_HIL_NO_SECURITY_MEMORY Security memory chip missing or broken.
0xC0000127	ERR_HIL_NO_MAC_ADDRESS_AVAILABLE No MAC address available.
0xC0000140	ERR_HIL_NETWORK_FAULT General communication fault.
0xC0000141	ERR_HIL_CONNECTION_CLOSED Connection closed.
0xC0000142	ERR_HIL_CONNECTION_TIMEOUT Connection timeout.

Diagnostic Code	Description
0xC0000143	ERR_HIL_LONELY_NETWORK Lonely network.
0xC0000144	ERR_HIL_DUPLICATE_NODE Duplicate network address.
0xC0000145	ERR_HIL_CABLE_DISCONNECT Cable disconnected.
0xC0000180	ERR_HIL_BUS_OFF Bus Off flag is set.
0xC0000181	ERR_HIL_CONFIG_LOCK Changing configuration is not allowed.
0xC0000182	ERR_HIL_APPLICATION_NOT_READY Application is not at ready state.
0xC0000183	ERR_HIL_RESET_IN_PROCESS Application is performing a reset.
0xC0000200	ERR_HIL_WATCHDOG_TIME_INVALID Watchdog time is out of range.
0xC0000201	ERR_HIL_APPLICATION_ALREADY_REGISTERED Application is already registered.
0xC0000202	ERR_HIL_NO_APPLICATION_REGISTERED No application registered.
0xC0000203	ERR_HIL_INVALID_COMPONENT_ID Invalid component identifier.
0xC0000204	ERR_HIL_INVALID_DATA_LENGTH Invalid data length.
0xC0000205	ERR_HIL_DATA_ALREADY_SET The data was already set.
0xC0000206	ERR_HIL_NO_LOGBOOK_AVAILABLE Logbook not available.
0xC0001000	ERR_HIL_INVALID_HANDLE No description available - ERR_HIL_INVALID_HANDLE.
0xC0001001	ERR_HIL_UNKNOWN_DEVICE No description available - ERR_HIL_UNKNOWN_DEVICE.
0xC0001002	ERR_HIL_RESOURCE_IN_USE No description available - ERR_HIL_RESOURCE_IN_USE.
0xC0001003	ERR_HIL_NO_MORE_RESOURCES No description available - ERR_HIL_NO_MORE_RESOURCES.
0xC0001004	ERR_HIL_DRV_OPEN_FAILED No description available - ERR_HIL_DRV_OPEN_FAILED.
0xC0001005	ERR_HIL_DRV_INITIALIZATION_FAILED No description available - ERR_HIL_DRV_INITIALIZATION_FAILED.

Diagnostic Code	Description
0xC0001006	ERR_HIL_DRV_NOT_INITIALIZED No description available - ERR_HIL_DRV_NOT_INITIALIZED.
0xC0001007	ERR_HIL_DRV_ALREADY_INITIALIZED No description available - ERR_HIL_DRV_ALREADY_INITIALIZED.
0xC0001008	ERR_HIL_CRC No description available - ERR_HIL_CRC.
0xC0001010	ERR_HIL_DRV_INVALID_RESOURCE No description available - ERR_HIL_DRV_INVALID_RESOURCE.
0xC0001011	ERR_HIL_DRV_INVALID_MEM_RESOURCE No description available - ERR_HIL_DRV_INVALID_MEM_RESOURCE.
0xC0001012	ERR_HIL_DRV_INVALID_MEM_SIZE No description available - ERR_HIL_DRV_INVALID_MEM_SIZE.
0xC0001013	ERR_HIL_DRV_INVALID_PHYS_MEM_BASE No description available - ERR_HIL_DRV_INVALID_PHYS_MEM_BASE.
0xC0001014	ERR_HIL_DRV_INVALID_PHYS_MEM_SIZE No description available - ERR_HIL_DRV_INVALID_PHYS_MEM_SIZE.
0xC0001015	ERR_HIL_DRV_UNDEFINED_HANDLER No description available - ERR_HIL_DRV_UNDEFINED_HANDLER.
0xC0001020	ERR_HIL_DRV_ILLEGAL_VECTOR_ID No description available - ERR_HIL_DRV_ILLEGAL_VECTOR_ID.
0xC0001021	ERR_HIL_DRV_ILLEGAL_IRQ_MASK No description available - ERR_HIL_DRV_ILLEGAL_IRQ_MASK.
0xC0001022	ERR_HIL_DRV_ILLEGAL_SUBIRQ_MASK No description available - ERR_HIL_DRV_ILLEGAL_SUBIRQ_MASK.
0xC0001100	ERR_HIL_DPM_CHANNEL_UNKNOWN No description available - ERR_HIL_DPM_CHANNEL_UNKNOWN.
0xC0001101	ERR_HIL_DPM_CHANNEL_INVALID No description available - ERR_HIL_DPM_CHANNEL_INVALID.
0xC0001102	ERR_HIL_DPM_CHANNEL_NOT_INITIALIZED No description available - ERR_HIL_DPM_CHANNEL_NOT_INITIALIZED.
0xC0001103	ERR_HIL_DPM_CHANNEL_ALREADY_INITIALIZED No description available - ERR_HIL_DPM_CHANNEL_ALREADY_INITIALIZED.
0xC0001120	ERR_HIL_DPM_CHANNEL_LAYOUT_UNKNOWN No description available - ERR_HIL_DPM_CHANNEL_LAYOUT_UNKNOWN.
0xC0001121	ERR_HIL_DPM_CHANNEL_SIZE_INVALID No description available - ERR_HIL_DPM_CHANNEL_SIZE_INVALID.
0xC0001122	ERR_HIL_DPM_CHANNEL_SIZE_EXCEEDED No description available - ERR_HIL_DPM_CHANNEL_SIZE_EXCEEDED.
0xC0001123	ERR_HIL_DPM_CHANNEL_TOO_MANY_BLOCKS No description available - ERR_HIL_DPM_CHANNEL_TOO_MANY_BLOCKS.

Diagnostic Code	Description
0xC0001130	ERR_HIL_DPM_BLOCK_UNKNOWN No description available - ERR_HIL_DPM_BLOCK_UNKNOWN.
0xC0001131	ERR_HIL_DPM_BLOCK_SIZE_EXCEEDED No description available - ERR_HIL_DPM_BLOCK_SIZE_EXCEEDED.
0xC0001132	ERR_HIL_DPM_BLOCK_CREATION_FAILED No description available - ERR_HIL_DPM_BLOCK_CREATION_FAILED.
0xC0001133	ERR_HIL_DPM_BLOCK_OFFSET_INVALID No description available - ERR_HIL_DPM_BLOCK_OFFSET_INVALID.
0xC0001140	ERR_HIL_DPM_CHANNEL_HOST_MBX_FULL No description available - ERR_HIL_DPM_CHANNEL_HOST_MBX_FULL.
0xC0001141	ERR_HIL_DPM_CHANNEL_SEGMENT_LIMIT No description available - ERR_HIL_DPM_CHANNEL_SEGMENT_LIMIT.
0xC0001142	ERR_HIL_DPM_CHANNEL_SEGMENT_UNUSED No description available - ERR_HIL_DPM_CHANNEL_SEGMENT_UNUSED.
0xC0001143	ERR_HIL_NAME_INVALID No description available - ERR_HIL_NAME_INVALID.
0xC0001144	ERR_HIL_UNEXPECTED_BLOCK_SIZE No description available - ERR_HIL_UNEXPECTED_BLOCK_SIZE.
0xC0001145	ERR_HIL_COMPONENT_BUSY The component is busy and cannot handle the requested service.
0xC0001150	ERR_HIL_INVALID_HEADER Invalid (file) header. E.g. wrong CRC/MD5/Cookie.
0xC0001151	ERR_HIL_INCOMPATIBLE Firmware does not match device.
0xC0001152	ERR_HIL_NOT_AVAILABLE Update file or destination (XIP-Area) not found.
0xC0001153	ERR_HIL_READ Failed to read from file/area.
0xC0001154	ERR_HIL_WRITE Failed to write from file/area.
0xC0001155	ERR_HIL_IDENTICAL Update firmware and installed firmware are identical.
0xC0001156	ERR_HIL_INSTALLATION Error during installation of firmware.
0xC0001157	ERR_HIL_VERIFICATION Error during verification of firmware.
0xC0001158	ERR_HIL_INVALIDATION Error during invalidation of firmware files.
0xC0001160	ERR_HIL_FORMAT Volume is not formatted.

Diagnostic Code	Description
0xC0001161	ERR_HIL_VOLUME (De-)Initialization of volume failed.
0xC0001162	ERR_HIL_VOLUME_DRV (De-)Initialization of volume driver failed.
0xC0001163	ERR_HIL_VOLUME_INVALID The volume is invalid.
0xC0001164	ERR_HIL_VOLUME_EXCEEDED Number of supported volumes exceeded.
0xC0001165	ERR_HIL_VOLUME_MOUNT The volume is mounted (in use).
0xC0001166	ERR_HIL_ERASE Failed to erase file/directory/flash.
0xC0001167	ERR_HIL_OPEN Failed to open file/directory.
0xC0001168	ERR_HIL_CLOSE Failed to close file/directory.
0xC0001169	ERR_HIL_CREATE Failed to create file/directory.
0xC0001170	ERR_HIL_MODIFY Failed to modify file/directory.
0x0000F005	SUCCESS_HIL_FRAGMENTED Fragment accepted.
0x0000F006	ERR_HIL_RESET_REQUIRED Reset required.
0x0000F007	ERR_HIL_EVALUATION_TIME_EXPIRED Evaluation time expired. Reset required.

- PN slave diagnostic codes

  Each bit mask corresponds to a specific diagnostic status or fault type. When you obtain a diagnostic code, you can perform AND operation between the code and each bit mask.

  - If the result equals the bit mask, the corresponding diagnostic status exists and the bit is set to 1.
  - If the result is 0, the corresponding diagnostic status does not exist.

    For example, if the obtained diagnostic code is 0x00002001, it indicates errors of 0x00000001 and 0x00002000 occur simultaneously.

Bitmask	Description
0x00000001	PNM_AP_IOD_DIAGFLAG_NOT_EXISTING The device does not exist in the network
0x00000002	PNM_AP_IOD_DIAGFLAG_NOT_COMMUNICATING The device is not in cyclic dataexchange
0x00000004	PNM_AP_IOD_DIAGFLAG_ADDRESS_CONFLICT Multiple stations with the ip or the name exist

Bitmask	Description
0x00000008	PNM_AP_IOD_DIAGFLAG_FAULTY_DCP The device sent a faulty DCP response
0x00000010	PNM_AP_IOD_DIAGFLAG_NEGATIVE_RPC The device sent an RPC response with an error
0x00000020	PNM_AP_IOD_DIAGFLAG_DEACTIVATED The device sent an RPC response with an error
0x00000800	PNM_AP_IOD_DIAGFLAG_MODULEDIFF The device has a Module Diff Block with identification related entries
0x00001000	PNM_AP_IOD_DIAGFLAG_RESERVED_PNMV2 Used by PNMV2
0x00002000	PNM_AP_IOD_DIAGFLAG_MODULEDIFF_DIAGNOSIS The device has a Module Diff Block with diagnosis related entries

- DP slave information

SlaveState	SlaveState=0	SlaveState=1
SlaveDiag=0	No DataIOExchange between master and slave. This slave is not configured.	Slave is present on the bus. DataIOExchange between master and slave.
SlaveDiag=1	The master and the corresponding slave do not exchange their I/O data. The master holds newly received diagnostic data in the internal diagnostic buffer.	Slave is present on the bus. The master and the corresponding slave exchange their I/O data. The master holds newly received diagnostic data in the internal diagnostic buffer.

Service and Support

Should you encounter a safety accident during the use or operation of the product, or face challenges in operating and maintaining the equipment, which remain unresolved after the relevant documentation is consulted, we provide multiple channels to ensure prompt resolution:

- Channel #1: Contact service@inovance.com.
- Channel #2: Visit https://www.inovance.com/global to access document downloads, after-sales support, spare parts ordering, repair applications, and authenticity verification services.
- Channel #3: Download My Inovance app (https://zshc-eu.inovance.com/download-pc/) where you can access products info and documentation, and query product parameters.

We are committed to providing you with quick and professional technical support, and we look forward to your satisfaction and trust.