INOVANCE

Advancing industrial technology,
for a better world



Industrial Robot
Remote Ethernet Control Function

User Guide

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

Data code 19012860A00

Legal and Disclaimer

Copyright

Copyright © 2025 Shenzhen Inovance Technology Co., Ltd. All rights reserved

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

Legal action will be taken against infringement.

Trademarks

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

Disclaimer of Liability

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

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

Waste Disposal

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

Qualified Personnel

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

Proper Use of the Product

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

Preface

Introduction

This "Industrial Robot Remote Ethernet Control Function User Guide" provides reference instructions for using Inovance Technology industrial robots with remote Ethernet control permissions. It is primarily intended for engineers and system integrators, aiming to provide comprehensive content such as usage workflows for remote Ethernet control permissions and specifications for remote call.

More Data

Data Name	Data Code	Description
IR-TP200 Series Industrial Robot Teach Pendant User Guide	19012798	This guide introduces software-related operations of the robot teach pendant. Contact Inovance technical support to obtain this guide.
Industrial Robot InoRobotLab User Guide	19012861	This guide introduces software-related operations of the robot InoRobotLab software. Contact Inovance technical support to obtain this guide.
Industrial Robot Instruction Guide	19012825	This guide introduces the function descriptions, instruction formats and parameter explanations, usage examples, and application cases for project instructions of Inovance Technology industrial robots.
Industrial Robot Controller Software Update User Guide	19012875	This guide introduces the software update procedures and component update procedures for Inovance Technology industrial robots.
Industrial Robot Remote I/O Control Function User Guide	19012874	This guide introduces the interface configuration, bus read/write instruction formats, and bus invocation precautions for remote I/O control of Inovance Technology industrial robots.
Industrial Robot Communication Guide	19012876	This guide introduces Inovance industrial robot multi-protocol communication solutions, covering mainstream industrial communication standards such as Modbus, EtherNet/IP, PROFINET, EtherCAT, and MC protocol, providing comprehensive communication technology guidance for industrial automation system integration.

Data Name	Data Code	Description
Industrial Robot Remote Ethernet Control Function User Guide (This guide)	19012860	This guide introduces the interface configuration, API call format, API function call examples, common issues and troubleshooting for remote Ethernet control of Inovance Technology industrial robots.

Revision History

Revision Date	Data Version	Revision
2026-01	A00	First issue

Access to the Guide

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

- Do keyword search under Service and Support at http://www.inovance.com.
- Scan the QR code on the product with your smart phone.
- Scan the QR code below to install My Inovance app, where you can search for and download user guides.

Basic Safety Instructions

Safety Precautions

Safety Disclaimer

- This chapter presents essential safety instructions for a proper use of the equipment. Before operating the equipment, read through the guide and comprehend all the safety instructions. Failure to observe the safety instructions may result in death, severe personal injuries, or equipment damage.
- "DANGER", "WARNING", and "CAUTION" items in this guide do not indicate all safety precautions that need to be followed; instead, they just supplement the safety precautions.
- Use this product according to designated environment requirements. Damage caused by improper use is not covered by warranty.
- Inovance shall take no responsibility for any personal injuries or property damage caused by improper usage.

Safety Levels and Definitions



Indicates that failure to comply with the notice will result in severe personal injuries or even death.



Indicates that failure to comply with the notice may result in severe personal injuries or even death.



Indicates that failure to comply with the notice may result in minor or moderate personal injuries or equipment damage.

Safety Precautions

- Drawings in the guide are sometimes shown without covers or protective guards. Remember to install the covers or protective guards as specified first, and then perform operations in accordance with the instructions.
- The drawings in the guide are shown for illustration only and may be different from the product you purchased.
- Operators must take mechanical precautions to protect personal safety and wear protective equipment, such as safety shoes, safety clothing, safety glasses, protective gloves, and protective sleeves.

Unpacking
 • Do not install the equipment if you find damage, rust, or signs of use on the equipment or accessories upon unpacking. • Do not install the equipment if you find water seepage or missing or damaged components upon unpacking.

- Do not install the equipment if you find the packing list does not conform to the equipment you received.



- Before unpacking, check whether the outer package is intact, damaged, wet, damped, or deformed.
- Unpack the package in sequence. Do not hit the package violently.
- Check whether there is damage, rust, or injuries on the surface of the equipment and equipment accessories before unpacking.
- Check whether the package contents are consistent with the packing list before unpacking.

Storage and Transportation



- Large-scale or heavy equipment must be transported by qualified professionals using specialized hoisting equipment. Failure to comply may result in personal injuries or equipment damage.
- Before hoisting the equipment, ensure that components such as the front cover and terminal blocks are secured firmly with screws. Loosely-connected components may fall off and result in personal injuries or equipment damage.
- Never stand or stay below the equipment when the equipment is being hoisted by the hoisting equipment.
- When hoisting the equipment with a steel rope, ensure the equipment is hoisted at a constant speed without suffering from vibration or shock. Do not turn the equipment over or let the equipment stay hanging in the air. Failure to comply may result in personal injuries or equipment damage.



- Handle the product with care and mind your steps to prevent personal injuries or product damage.
- When carrying the equipment with bare hands, hold the equipment casing firmly with care to prevent parts from falling. Failure to comply may result in personal injuries.
- Store and transport the equipment based on the storage and transportation requirements. Failure to comply will result in equipment damage.
- Avoid storing or transporting the equipment in environments with water splash, rain, direct sunlight, strong electric field, strong magnetic field, and strong vibration.
- Avoid storing the product for more than 3 months. When the product needs to be stored for an extended period, take more strict protection and necessary inspection.
- Pack the equipment strictly before transportation. Use a sealed box for long-distance transportation.
- Never transport the equipment with other equipment or materials that may harm or have negative impacts on this equipment.

Installation



- The equipment can only be operated by professionals with electrical knowledge. Non-professionals are not allowed to operate on the equipment.



- Read through the guide and safety precautions before installation.
- Do not install the equipment in places with strong electric or magnetic fields.
- Before installation, check that the mechanical strength of the installation site can bear the weight of the equipment. Failure to comply will result in mechanical hazards.
- Do not wear loose clothes or accessories during installation. Failure to comply may result in an electric shock.
- When installing the equipment (such as controller) in a closed environment (such as a cabinet or casing), use a cooling device (such as a fan or air conditioner) to cool the environment down to the required temperature. Failure to comply may result in equipment over-temperature or a fire.
- Do not retrofit the equipment.
- Do not loosen the fixing bolts of the product parts and components.
- When the equipment (such as controller) is installed in a cabinet or final assembly, a fireproof enclosure providing both electrical and mechanical protections must be provided. The IP rating must meet IEC standards and local laws and regulations.
- Before installing devices with strong electromagnetic interference, such as a transformer, install a shielding device for the equipment to prevent malfunction.
- Install the product on an incombustible object such as metal and do not touch or attach the product to combustibles. Failure to comply can result in fire accident.



- Cover the top of the equipment (such as controller) with a piece of cloth or paper during installation. This is to prevent unwanted objects such as metal chippings, oil, and water from falling into the equipment and causing faults. After installation, remove the cloth or paper on top of the equipment to prevent over-temperature caused by poor ventilation due to blocked ventilation holes.

Wiring



- Equipment installation, wiring, maintenance, inspection, or component replacement must be performed only by professionals.
- Before wiring, disconnect all the power supplies of the equipment. Wait for at least the time designated on the equipment warning label before further operations because residual voltage still exists after

power-off. Measure the direct voltage of the main circuit and ensure that the voltage is within the safety range. Failure to comply can result in an electric shock.

- Do not perform wiring, remove the equipment cover, or touch the circuit board with power ON. Failure to comply can result in an electric shock.
- Check that the equipment is grounded properly. Failure to comply can result in an electric shock.



- Do not connect the input power supply to the output side of the equipment. Failure to comply can result in equipment damage or even a fire.
- When connecting a drive to the motor, check that the phase sequences of the drive and motor terminals are consistent to prevent reverse motor rotation.
- Cables used for wiring must meet cross sectional area and shielding requirements. The shield of the cable must be reliably grounded at one end.
- Tighten the terminal screws with the tightening torque specified in the user guide. Improper tightening torque may result in overheat, damage to the connecting parts, or a fire.
- After wiring is done, check that all cables are connected properly and no screws, washers or exposed cables are left inside the equipment. Failure to comply may result in electric shock or equipment damage.



- Follow the electrostatic discharge (ESD) procedure and wear an anti-static wrist strap to perform wiring. Failure to comply may result in damage to the equipment or to the internal circuit.
- Use shielded twisted pairs for the control circuit. Connect the shield to the grounding terminal of the equipment for grounding purpose. Failure to comply can result in equipment malfunction.

Power-On



- Before power-on, check that the equipment is installed properly with reliable wiring and the motor can be restarted.
- Check that the power supply meets equipment requirements before power-on to prevent equipment damage or a fire.
- Do not open the cabinet door or protective cover of the product, touch any wiring terminal of the product, or remove any part of the product with power on. Failure to comply can result in an electric shock.



- After the completion of wiring and parameter setting, perform a robot test run to confirm that the robot can operate safely. Failure to comply may result in personal injuries or equipment damage.

- Before power-on, check that the rated voltage of the equipment is consistent with that of the power supply. Failure to comply may result in fire accident.
- Before power-on, check that no one is near the equipment, motor, or machine. Failure to comply may result in death or personal injuries.

Operation



- The equipment must be operated only by professionals. Failure to comply can result in death or personal injuries.
- Do not touch any connecting terminals or disassemble any unit or component of the equipment during operation. Failure to comply can result in an electric shock.



- Do not touch the equipment enclosure, fan, or resistor with bare hands. Failure to comply may result in burns.
- Prevent metal or other objects from falling into the equipment during operation. Failure to comply may result in a fire or equipment damage.

Maintenance



- Equipment installation, wiring, maintenance, inspection, or component replacement must be performed only by professionals.
- Do not maintain the equipment with power ON. Failure to comply can result in an electric shock.
- Before maintenance, cut off all the power supplies of the equipment and wait for at least the time designated on the equipment warning label.
- When a permanent magnet motor is used, do not touch the motor terminals immediately after power-off because the motor terminals can generate induced voltage during rotation even after the equipment is powered off. Failure to comply can result in an electric shock.

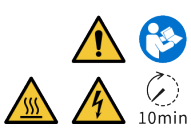


- Perform routine and regular inspection and maintenance on the equipment in accordance with maintenance requirements and keep a record.

Repair
 • Equipment installation, wiring, maintenance, inspection, or component replacement must be performed only by professionals. • Do not maintain the equipment with power ON. Failure to comply can result in an electric shock. • Before inspection and repair, cut off all the power supplies of the equipment and wait for at least the time designated on the equipment warning label.
 • Submit a request for repair services in accordance with the product warranty agreement. • When the fuse is blown or the circuit breaker or earth leakage circuit breaker (ELCB) trips, wait for at least the time designated on the equipment warning label before power-on or further operations. Failure to comply may result in death, personal injuries or equipment damage. • Troubleshooting and repair must be performed by professionals in accordance with the repair instructions, with repair records kept properly. • Replace quick-wear parts of the equipment in accordance with the replacement instructions. • Do not operate on damaged equipment. Failure to comply may result in death, personal injuries, or severe equipment damage. • After the equipment is replaced, check the wiring and set parameters again.
Disposal
 • Dispose of retired equipment in accordance with local regulations and standards. Failure to comply may result in death, personal injuries, or property damage. • Recycle retired equipment in accordance with industry waste disposal standards to avoid environmental pollution.

Safety Label

For safe operation and maintenance, comply with the safety labels on the equipment. Do not damage or remove the safety labels. See the following table for descriptions of the safety labels.

Safety Label	Description
	• Read through the safety instructions before operating the equipment. Failure to comply may result in death, personal injuries, or equipment damage. • Do not touch the terminals or remove the cover with power ON or within 10 min after power-off. Failure to comply can result in an electric shock.

Industrial Information Security

The product provides an interface to connect to the network and transmits data through the network interface. To protect factories, systems, machines, and networks from cyber attacks and ensure safe operation, a proper industrial information security protection mechanism must be implemented.
The customer is responsible for providing and continuously ensuring a secure connection between the product and the customer network or any other network to prevent unauthorized access to its factories, systems, machines, and networks. The system or machine can be connected to the corporate network or the Internet only when secure connections are ensured and appropriate security measures are taken (for example, anti-virus software or firewall installed).
Inovance continuously develops and improves products and solutions to improve safety. It is strongly recommended that you update the product promptly and always use the latest version.



Malware (such as viruses, Trojans, and worms) can put the product in an unsafe running state, which may result in death, serious injuries, and property damage. Observe the following precautions:

- Always use the latest software version. If the product version is no longer supported or the latest version is not used, customers will be exposed to increased risk of cyber attacks.
- Take appropriate protection measures (including but not limited to installing anti-virus software, firewall, WAF, IPS/IDS, and situation awareness system, and applying authentication, data encryption) to prevent files in mobile storage devices from being damaged by malware, and to prevent products, networks, systems, and interfaces from unauthorized access, interference, intrusion, data leakage, or information breach.
- Check all safety-related interfaces and settings after commissioning.

Contents

1 Release Notes 13

2 Overview 17

3 Configuration Through Software Interface 18

4 API Function Call Description 19

4.1 Development Language Selection 19

4.2 Basic Function Description 20

4.3 Function Call Description 20

5 Typical Application Cases of API Functions 24

5.1 Opening Robot Project and Starting Automatic Operation via Host Controller 24

5.2 Planning Point Positions and Controlling Robot Motion via Host Controller 27

5.3 Controlling Independent Axis Motion via Host Controller 32

5.4 Controlling Self-Learning Motion via Host Controller 40

6 API String Call Description 46

7 Common Issues and Troubleshooting 47

7.1 Issue 1: Robot Controller Error 0x0084 47

7.2 Issue 2: How to Query the API Version Number 47

8 Appendix 48

8.1 Appendix 1: API Function Table 48

8.1.1 Appendix 1.1 Structures 48

8.1.2 Appendix 1.2 Functions 50

8.2 Appendix 2: API String Table 207

8.3 Appendix 3: API Fault Description 508

1 Release Notes

API version	Controller version	Version description
4.24.4.3	V4R24C4SPC0 and later	1. Added compensation parameter acquisition interface: IMC100_Get_SupplementaryStrParamComp 2. Added supplementary parameter setting interface: IMC100_Set_SupplementaryStrParamComp
4.24.4.2	V4R24C4SPC0 and later	1. Added mode setting interface: IMC100_Set_OffSetMode 2. Added mode getting interface: IMC100_Get_OffSetMode 3. Added interface of translation in Cartesian reference frame with reference joint angles: IMC100_Get_OffsetRef_RobP 4. Added interface of translation in Cartesian reference frame without reference joint angles: IMC100_Get_OffsetRefnotJPos_RobP
4.24.4.1	V4R24C4SPC0 and later	1. Added interface of setting system diagnosis save parameters: SetDiagnosisItem 2. Added interface of reading system diagnosis save parameters: GetDiagnosisItem 3. Added interface of triggering system diagnosis save operation: SetDiagnosisSave 4. Added interface of reading system diagnosis save progress value: GetDiagnosisSave
3.15.0.13	V4R24C3SPC0L1B1.0T202403153 and later	Bug fixes
3.15.0.11	V4R24C3SPC0L1B1.0T20240223 and later	1. Added trigger pulse interface: IMC100_Pulse 2. Added global acceleration related interfaces: IMC100_Set_Acc/IMC100_Get_Acc 3. Added grip load related interfaces: IMC100_Set_GripLoad/IMC100_Get_GripLoad

API version	Controller version	Version description
3.15.0.10	V4R24C3SPC0L1B1.0T20240219 and later	Added related interfaces of move back distance for collision detection: IMC100_Set_CollDist/ IMC100_Get_CollDist IMC100_Set_TeachModeCollDist/ IMC100_Get_TeachModeCollDist IMC100_Set_PlayBackModeCollDist/ IMC100_Get_PlayBackModeCollDist/
3.15.0.8	V4R24C2SPC0B3 and later	1. Added interface of updating position parameters corresponding to global position variables stored in memory: IMC100_Set_MemRobP 2. Added interface of updating position parameters corresponding to global joint position variables stored in memory: IMC100_Set_MemRobJP
3.15.0.6	V4R24C2SPC0B2 and later	Added interference zone related interfaces: IMC100_Set_InterferZoneActStat IMC100_Get_InterferZoneActStat IMC100_Set_InterferZonePara IMC100_Get_InterferZonePara IMC100_Set_InterferToolActNum IMC100_Get_InterferToolActNum IMC100_Set_InterferToolPara IMC100_Get_InterferToolPara
3.15.0.5	V4R24C2SPC0B3 and later	Added linear speed related interfaces and adjustable trajectory recovery threshold interfaces: IMC100_Set_BindTcpSpeedValue IMC100_Get_BindTcpSpeedValue IMC100_Get_TcpSpeedDAOutAndVel IMC100_Close_BindTcpSpeed IMC100_Set_TraceRecoverMode IMC100_Get_TraceRecoverMode IMC100_Set_VarTraceRecoverParams IMC100_Get_VarTraceRecoverParams
3.14.1	V4R24C1SPC0B5 and later	Added interface of querying the number of robot manipulator axes:

API version	Controller version	Version description
		IMC100_Get_RobotAxisNum
3.14.0.2	S03.24R and later	1. Added independent axis related interfaces: IMC100_IndCMove/IMC100_IndSpeed/ IMC100_IndResetOld/IMC100_IndReset/ IMC100_Get_IndCMoveSts/ IMC100_Get_IndResetSts/ IMC100_Get_MotionStsExceptIndAxis 2. Added multi-point related interfaces: IMC100_Get_RobPFromFile/ IMC100_Set_RobPToFile/ IMC100_Get_CurRobPFileName/ IMC100_Set_RobPHereToFile 3. Added speed limit switch related interfaces: IMC100_Get_SpeedLimitSwitch/ IMC100_Set_SpeedLimitSwitch
3.13.0.1	S03.23R and later	1. Added interface of querying the number of motion data in the cache: IMC100_Get_CurCmdCacheNum 2. Added related interfaces of reading point position label: IMC100_Get_RobP_Label/ IMC100_Get_RobJP_Label/ IMC100_Get_RobLP_Label 3. Added collision detection parameter related interfaces: IMC100_Set_CollModeAndAction/ IMC100_Get_CollModeAndAction/ IMC100_Set_AxisCollMode/ IMC100_Get_AxisCollMode/ IMC100_Set_AxisCollLevel/ IMC100_Get_AxisCollLevel/ IMC100_Set_TeachModeAxisCollMode/ IMC100_Get_TeachModeAxisCollMode/ IMC100_Set_TeachModeCollAction/ IMC100_Get_TeachModeCollAction/ IMC100_Set_TeachModeAxisCollLevel/ IMC100_Get_TeachModeAxisCollLevel/ IMC100_Set_PlayBackModeAxisCollMode/ IMC100_Get_PlayBackModeAxisCollMode/ IMC100_Set_PlayBackModeCollAction/ IMC100_Get_PlayBackModeCollAction/ IMC100_Set_PlayBackModeAxisCollLevel/ IMC100_Get_PlayBackModeAxisCollLevel

API version	Controller version	Version description
		4. Added interface of moving to specified motion point using MovJAbs: IMC100_MovJAbs_JP
3.12.4	S03.22R and later	1. Added new interfaces for point position, tool parameters, and workobject parameters. 2. Deleted old interfaces for point position, tool parameters, and workobject parameters. 3. Added work origin (for external axis) interface: IMC100_Set_HomeJPos/IMC100_Get_HomeJPos 4. Added coordination switch interfaces: IMC100_Set_TeachCoordinate/ IMC100_Get_TeachCoordinate

2 Overview

The remote Ethernet control function allows you to develop dedicated application software for the robot system. In this architecture, the robot controller operates as a server on a fixed port (2222), while the user end acts as a client. Interaction between client and server can be accomplished through the following two methods:

- Method 1: Interact with the robot controller through API function interfaces, supporting programming languages such as VB, VC, C#, and LabView.
- Method 2: Interact with the robot controller through API strings, supporting more systems (e.g., Linux systems) and languages (in addition to the existing VB, VC, C#, and LabView languages supported by API functions).

3 Configuration Through Software Interface

After the communication is established between the user end and the robot controller:

- For monitoring or querying only, you can use the user end to directly query the information from the robot controller through API function interfaces or strings.
- For controlling the robot, you must switch to "Remote Ethernet client" control permission, and get the control permission through API function interfaces or strings.

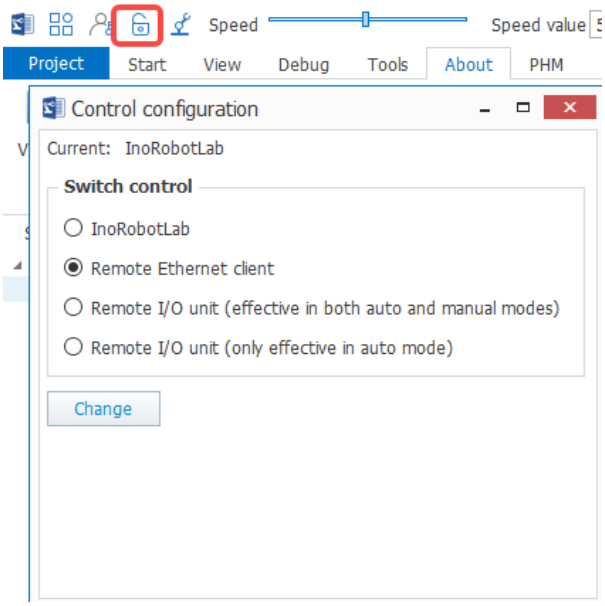
Methods for switching control permission:

- Through the InoRobotLab software

1. Switch to Admin or Manufacturer mode through the "Login" button.



2. Click the "Control permission" button and select "Remote Ethernet client", as shown in the figure below.



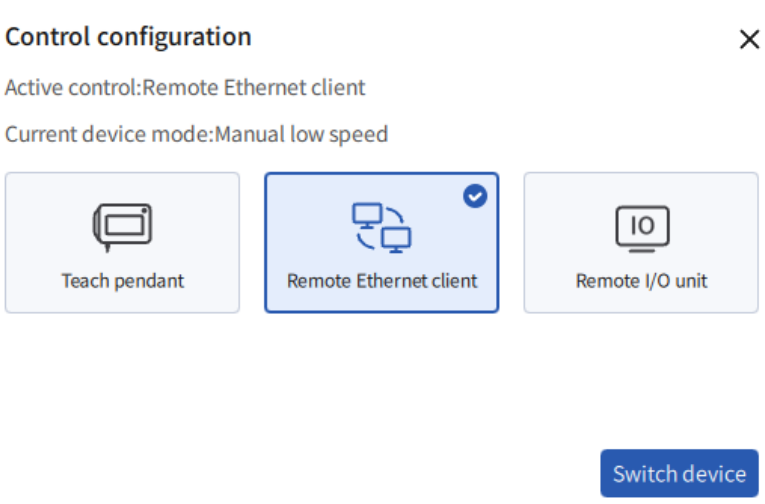
- Through the InoRobotTP software

1. Switch the user mode to Admin or Manufacturer mode.



2. Click the "Control" menu and select "Remote Ethernet client", as shown in the figure below.





4 API Function Call Description

For interaction with the controller through the API function interfaces, VB, VC, C#, and LabView programming languages are supported.

> NOTE
>
> The API function is currently not supported by the virtual controller.

4.1 Development Language Selection

- VB-based development
  1. Taking the Visual Basic 6.0 environment as an example, create a new project.
  2. Copy the IMC100API.bas and IMC100API.dll files provided with the product to the corresponding project directory.
  3. In the menu, select "Project/Add Module/Existence", locate the IMC100API.bas in the corresponding project directory, and add it to the project.
  4. Call the specific function interfaces to develop the application program.
- VC-based development
  1. Taking the Visual Studio 2010 environment as an example, create a new C++ project.
  2. Copy the IMC100API.lib, IMC100API.dll, and IMC100API.h files provided with the product to the corresponding project directory.
  3. Add the #include "IMC100API.h" statement in the program file.
  4. Add the #pragma comment(lib, "IMC100API.lib") statement in the program file, or add IMC100API.lib in the "Project Properties/Linker/Input/Additional Dependencies".
  5. Call the specific function interfaces to develop the application program.
- VB.Net-based development
  1. Taking the Visual Studio 2010 environment as an example, create a new Visual Basic .NET project.
  2. Copy the IMC100API.vb file provided with the product to the project directory, and copy the IMC100API.dll file to the project's target output path.
  3. In the menu, select "Project/Add existing item", locate the IMC100API.vb file in the corresponding project directory, and add it to the project.
  4. Call the specific function interfaces to develop the application program.
- C#-based development
  1. Taking the Visual Studio 2010 environment as an example, create a new Visual C# .NET project.
  2. Copy the IMC100API.cs file provided with the product to the project directory, and copy the IMC100API.dll file to the project's target output path.
  3. In the menu, select "Project/Add existing item", locate the IMC100API.cs file in the corresponding project directory, and add it to the project.
  4. Call the specific function interfaces to develop the application program.

4.2 Basic Function Description

API function interfaces are primarily divided into three categories: initialization application, monitoring, and control.

- Initialization application: Functions in this category serve as the foundation for calling other functions, including IMC100_Init_ETH() and IMC100_Exit_ETH().
- Monitoring: Functions in this category can be normally called after a successful call of IMC100_Init_ETH(), without restrictions of control permission and user level. These functions include IMC100_Get_RobPosHere(), IMC100_Get_DONum(), IMC100_Get_StrPara(), IMC100_Get_RobP(), etc.
- Control: Functions in this category not only require the successful call of IMC100_Init_ETH(), but also require the control device to be a "Remote Ethernet client". Additionally, control permission must be acquired by calling IMC100_AcqPermit(). Depending on the user level, some functions additionally require a login via IMC100_UserLogin(). These functions include IMC100_MovJ_P(), IMC100_Set_DO(), IMC100_Get_RobP(), IMC100_Set_HomeJPos(), etc.



- For the detailed function description, see 8.1 Appendix 1: API Function Table section.
- API functions and API strings are consistent in functionality, with a one-to-one correspondence between them.

4.3 Function Call Description

The following describes the implementation and call examples of basic functions using C/C++ application development on the VS platform.

- Call the IMC100_Init_ETH() function to connect the robot via the network, and call IMC100_Exit_ETH() to disconnect the robot.
- Before calling any other APIs, call IMC100_Init_ETH() once to ensure that the robot is connected every time the application is launched. If the function returns a non-zero value, check whether the robot control system has started normally. See 8.3 Appendix 3: API Fault Description for any troubleshooting.
- IMC100_Exit_ETH() should be called after other APIs. The robot will disconnect after this API returns 0.

A code example for calling IMC100_Init_ETH() is as follows:

```
int ret = 0;      DWORD dwIP1 = 0xc0a81719;   // Corresponding IP: 192.168.23.25
    int ipPort = 2222;   int timeOut = 5;    // Communication timeout 5s
    int robotNo = 0;     ret = IMC100_Init_ETH(dwIP1, ipPort, timeOut, robotNo); if(ret < 0)
    {
        // Add exception handlers here
        return;   }
```

A code example for calling IMC100_Exit_ETH() is as follows:

```
ret = IMC100_Exit_ETH(0)
    if(ret < 0)
    {
```

```
        // Add exception handlers here
        return;   }
```

- Monitor the robot status

The IMC100 robot is provided with hundreds of API function interfaces for monitoring the robot status. These functions are not restricted by control permissions or user levels. Before calling these functions, ensure that the robot system has started normally and that the robot is successfully connected. These functions include IMC100_Get_RobPosHere(), IMC100_Get_DINum(), IMC100_Get_StrPara(), IMC100_Get_RobP(), etc.

A code example for calling IMC100_Get_RobPosHere() is as follows:

```
    // Query the robot position value in the workobject frame
int ret = 0;    int robotNo = 0;    int dinum = 0;    int dists = 0; ROB_POS posTemp;    memset(&posTemp, 0, sizeof(posTemp));    ret = IMC100_Get_RobPosHere(&posTemp, robotNo);    if(ret < 0)
  {
        // Add exception handlers here
}
```

A code example for calling IMC100_Get_DI() is as follows:

```
    // Query the DI0 status. dists=1 indicates ON status
ret = IMC100_Get_DI(dinum, &dists, robotNo);
    if(ret < 0)
  {
        // Add exception handlers here
}
```

- Acquire the control permission

Call IMC100_AcqPermit() to acquire the robot control permission, and call IMC100_CurPermit() to query the current client that has the control permission.

Since a robot system can be connected to multiple Ethernet clients, any client that needs to control the robot must acquire the control permission. Before calling, ensure that the robot system has started normally and is successfully connected. Also, on the teach pendant interface, select "Remote Ethernet client" under "System settings" > "Other settings" > "Other" > "Control device".

A code example for calling IMC100_CurPermit() and IMC100_AcqPermit() is as follows:

```
int ret = 0;    int ower = 0;    DWORD IpAddr = 0;    int ipPort = 0;    int robotNo = 0;    ret = IMC100_CurPermit(&ower, &IpAddr, & ipPort, robotNo);    if(ret < 0)
    {
        // Add exception handlers here
        return;   }
    if(ower != 1)  // Current client device does not have control permission
```

```
{
    ret = IMC100_AcqPermit(1, robotNo);   // Forcibly acquire the control permission. Ordinary acquisition is possible when "ower" is 0.
    if(ret < 0)
    {
        // Add exception handlers here
    }
}
```

- User login

Call IMC100_CurUserType() to query the current user level of the device, call IMC100_UserLogin() for user login, and call IMC100_UserLogout() to log out the current user.

Different user levels allow different levels and scopes of control and operation access to the robot.

Before calling, ensure that the robot system has started normally and is successfully connected. On the teach pendant interface, ensure the "Remote Ethernet client" option under "System settings" > "Other settings" > "Other" > "Control device" is selected.

A code example for calling IMC100_CurUserType(), IMC100_UserLogin(), and IMC100_UserLogout() is as follows:

```
int ret = 0;    int type = 0;    char password[8]; int robotNo = 0;    ret = IMC100_CurUserType(&type, robotNo);
 if(ret < 0)
    {
        // Add exception handlers here
    }
    type = 2; memcpy(password, "000000", sizeof(password)); ret = IMC100_UserLogin(type, password, robotNo); // Log in to Admin mode, with the same password for the teach pendant
    if(ret < 0)
    {
        // Add exception handlers here
    }
    ret = IMC100_UserLogout(robotNo);    if(ret < 0)
    {
        // Add exception handlers here
    }
```

- Return the robot to the origin

Call IMC100_DsMode() to enable data stream mode, and call IMC100_Home() to control the robot to return to the origin.

Before calling, ensure that the robot system has started normally, the robot system is successfully connected, and the client has the control permission.

A code example for calling IMC100_DsMode() and IMC100_Home() is as follows:

```
int ret = 0;      int sts = 0;      int robotNo = 0;    ret = IMC100_Get_DsMode(&sts, robotNo);    if(ret < 0)
    {
        // Add exception handlers here
    }
    if(sts == 0)  // Data stream mode is disabled
    {
        int cmd = 1;          ret = IMC100_DsMode(cmd, robotNo);   // Enable data stream mode
       if(ret < 0)
{
// Add exception handlers here
        }
    }
    int num = 0;    ret = IMC100_Home(num, robotNo);    // Robot returns to origin point 0
    if(ret < 0)
    {
        // Add exception handlers here
    }
```

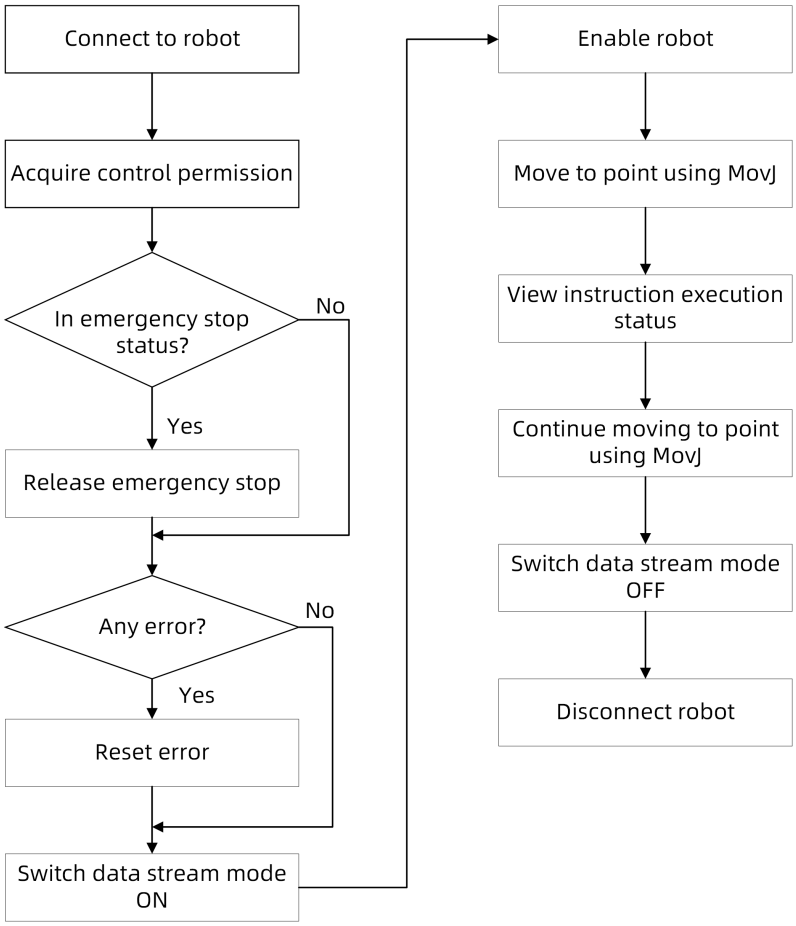
5 Typical Application Cases of API Functions

The following illustrates the calling process of API functions with complete typical application cases.

5.1 Opening Robot Project and Starting Automatic Operation via Host Controller

Before this operation, ensure that the target robot project can operate normally.

Overall process:



Code example:

```
int ret = 0;
DWORD dwIP1 = 0xc0a81719;   // Corresponding IP: 192.168.23.25
    int ipPort = 2222;   int timeOut = 5;    // Communication timeout 5s
    int robotNo = 0;     /* Connect to the robot */
    ret = IMC100_Init_ETH(dwIP1, ipPort, timeOut, robotNo); if(ret < 0)
    {
        // Add exception handlers here
        return;
}


    int ower = 0;
```

```
DWORD IpAddr = 0;
int ipPort = 0;
/* Acquire control permission */
ret = IMC100_CurPermit(&ower, &IpAddr, & ipPort, robotNo);
if(ret < 0)
    {
        // Add exception handlers here
    }
    if(ower != 1)  // Current client device does not have control permission
    {
ret = IMC100_AcqPermit(1, robotNo);   // Forcibly acquire the control permission. Ordinary acquisition is possible when "ower" is 0.
        if(ret < 0)
        {
            // Add exception handlers here
        }
    }

int type = 2;
char password[8];
memcpy(password, "000000", sizeof(password));        /* Log in to Admin mode */
ret = IMC100_UserLogin(type, password, robotNo);   // Log in to Admin mode, with the same password for teach pendant
    if(ret < 0)
    {
        // Add exception handlers here
    }

    int sts = 0;      /* Emergency stop status */
    ret = IMC100_Get_EStopSts(&sts, robotNo);
if(ret < 0)
    {
        // Add exception handlers here
    }
    if(sts == 1)
    {
        int cmd = 0;            /* Release emergency stop */
```

```
            ret = IMC100_EmergStop(cmd, robotNo);
if(ret < 0)
{
// Add exception handlers here
            }
      }

      int err = 0;      /* Error query */
ret = IMC100_Get_SysErr(&err, robotNo);
if(ret < 0)
      {
            // Add exception handlers here
      }
      if(err != 0)
      {
            /* Reset the error */
            ret = IMC100_ResetErr(robotNo);
if(ret < 0)
{
// Add exception handlers here
            }
      }

      int vel = 50;     /* Set the running speed */
      ret = IMC100_Set_Vel(vel, robotNo);
if(ret < 0)
{
  // Add exception handlers here
}

      int mode = 2;   // Auto mode
      /* Set the Auto mode */
      ret = IMC100_Set_Mode(mode, robotNo);
if(ret < 0)
      {
            // Add exception handlers here
```

```
    }

    cmd = 1; /* Start the program */
    ret = IMC100_PrgCtrl(cmd, robotNo);
if(ret < 0)
    {
        // Add exception handlers here
    }

    cmd = 0; /* Stop the program */
    ret = IMC100_PrgCtrl(cmd, robotNo);
if(ret < 0)
    {
        // Add exception handlers here
    }

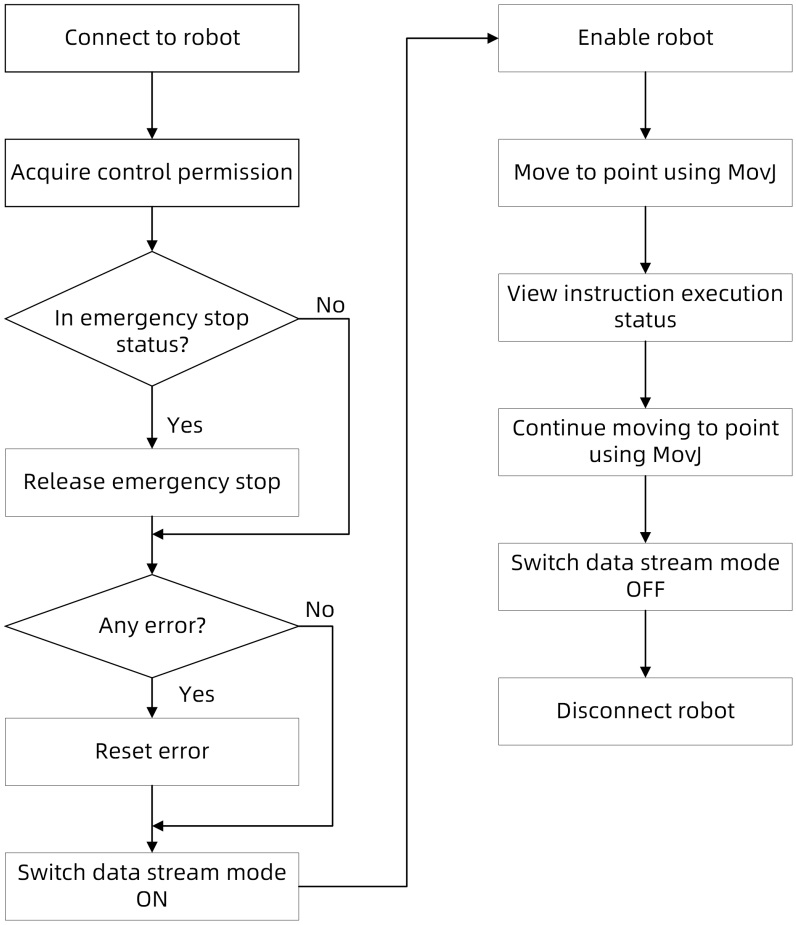
    int mode = 1; /* Set the Manual mode */
    ret = IMC100_Set_Mode(mode, robotNo);
if(ret < 0)
    {
        // Add exception handlers here
    }

    /* Disconnect the robot */
    ret = IMC100_Exit_ETH(0)
    if(ret < 0)
    {
        // Add exception handlers here
    }
```

5.2 Planning Point Positions and Controlling Robot Motion via Host Controller

Before this operation, ensure that the robot can safely reach the point positions planned by the host controller.

Overall process:



Code example:

```
int ret = 0;
DWORD dwIP1 = 0xc0a81719;   // Corresponding IP: 192.168.23.25
      int ipPort = 2222;
int timeOut = 5;    // Communication timeout time 5s
      int robotNo = 0;
/* Connect to the robot */
      ret = IMC100_Init_ETH(dwIP1, ipPort, timeOut, robotNo);
      if(ret < 0)
      {
            // Add exception handlers here
            return;
}


      int ower = 0;
DWORD IpAddr = 0;
int ipPort = 0; /* Acquire the control permission */
ret = IMC100_CurPermit(&ower, &IpAddr, & ipPort, robotNo);
if(ret < 0)
```

```
    {
        // Add exception handlers here
    }
    if(ower != 1)  // Current client device does not have control permission
    {
ret = IMC100_AcqPermit(1, robotNo);   // Forcibly acquire the control permission. Ordinary acquisition is possible when "ower" is 0.
        if(ret < 0)
        {
            // Add exception handlers here
        }
    }

    int sts = 0;      /* Emergency stop status */
    ret = IMC100_Get_EStopSts(&sts, robotNo);
if(ret < 0)
    {
        // Add exception handlers here
    }
    if(sts == 1)
    {
        int cmd = 0;          /* Release emergency stop */
        ret = IMC100_EmergStop(cmd, robotNo);
if(ret < 0)
    {
        // Add exception handlers here
    }
    }

    int err = 0;      /* Error query */
ret = IMC100_Get_SysErr(&err, robotNo);
if(ret < 0)
    {
        // Add exception handlers here
    }
    if(err != 0)
```

```
    {
        /* Reset the error */
        ret = IMC100_ResetErr(robotNo);
if(ret < 0)
    {
        // Add exception handlers here
    }
    }

    cmd = 1; /* Enable */
    ret = IMC100_MotorEnable(cmd, robotNo);
if(ret < 0)
    {
        // Add exception handlers here
    }

    /* Ensure the robot is enabled before enabling data stream mode */
    int motorEnableSts = 0;
    while (0 == motorEnableSts)
    {
        ret = IMC100_Get_MotorSts(&motorEnableSts, robotNo);        if(ret < 0)
    {
        // Add exception handlers here
    }
    Sleep(10);
}

    cmd = 1; /* Enable data stream mode */
    ret = IMC100_DsMode(cmd, robotNo);
    if(ret < 0)
    {
        // Add exception handlers here
    }

/* Note: The motion sending actions in data stream mode should not be too frequent (but just meet the working condition requirements), as it may cause the internal motion data queue to be full. */
```

```
    ROB_POS pos;
memset(&pos, 0, sizeof(pos));
pos. RPosData[0] = 110;
pos. RPosData[1] = 35;
pos. RPosData[2] = 956;
pos. RPosData[3] = -108;
pos. RPosData[4] = -5;
pos. RPosData[5] = -20;
VER_DATA movVel;
memset(&movVel, 0, sizeof(movVel));
movVel. velPercent = 100 ;
MOV_IO MovIOInfo ;
memset(&MovIOInfo, 0, sizeof(MovIOInfo)) ;
MovIOInfo. IONo = 1 ;
MovIOInfo. IOVa = 1 ;
MovIOInfo. Value =0.005 ;     /* MovJ motion */
    ret = IMC100_MovJ_RobPos(&pos, 100, 0, 1, &MovIOInfo, robotNo)
    if(ret < 0)
    {
        // Add exception handlers here
    }

    /* Motion completion status check */
/* Note: If the subsequent program execution depends on the completion of the preceding motion, it is necessary to obtain the motion completion status for blocking logic. Otherwise, this logic should not be applied, for example, when smooth transitions between consecutive motions are required. */
    ret = IMC100_Get_CurCmdSts(&sts, robotNo);
if(ret < 0)   // Add exception handlers here
    if(sts == 0)
    {
        // Motion not completed. Perform custom operations, e.g., loop query
    }

memset(&pos, 0, sizeof(pos));
pos. RPosData[0] = 324;
pos. RPosData[1] = -210;
```

```
pos. RPosData[2] = 297;
pos. RPosData[3] = 25;
pos. RPosData[4] = 10;
pos. RPosData[5] = 169;
memset(&movVel , 0, sizeof(movVel));
movVel. velPercent = 100 ;
memset(&MovIOInfo, 0, sizeof(MovIOInfo)) ;
MovIOInfo. IONo = 1 ;
MovIOInfo. IOVa = 0 ;
MovIOInfo. Value =0.005 ;
/* MovJ motion */
    ret1 = IMC100_MovJ_RobPos(&pos, 100, 0, 1, &MovIOInfo, robotNo)
    if(ret < 0)
    {
        // Add exception handlers here
    }

    cmd = 0; /* Disable data stream mode */
    ret = IMC100_DsMode(cmd, robotNo);
    if(ret < 0)
    {
        // Add exception handlers here
    }

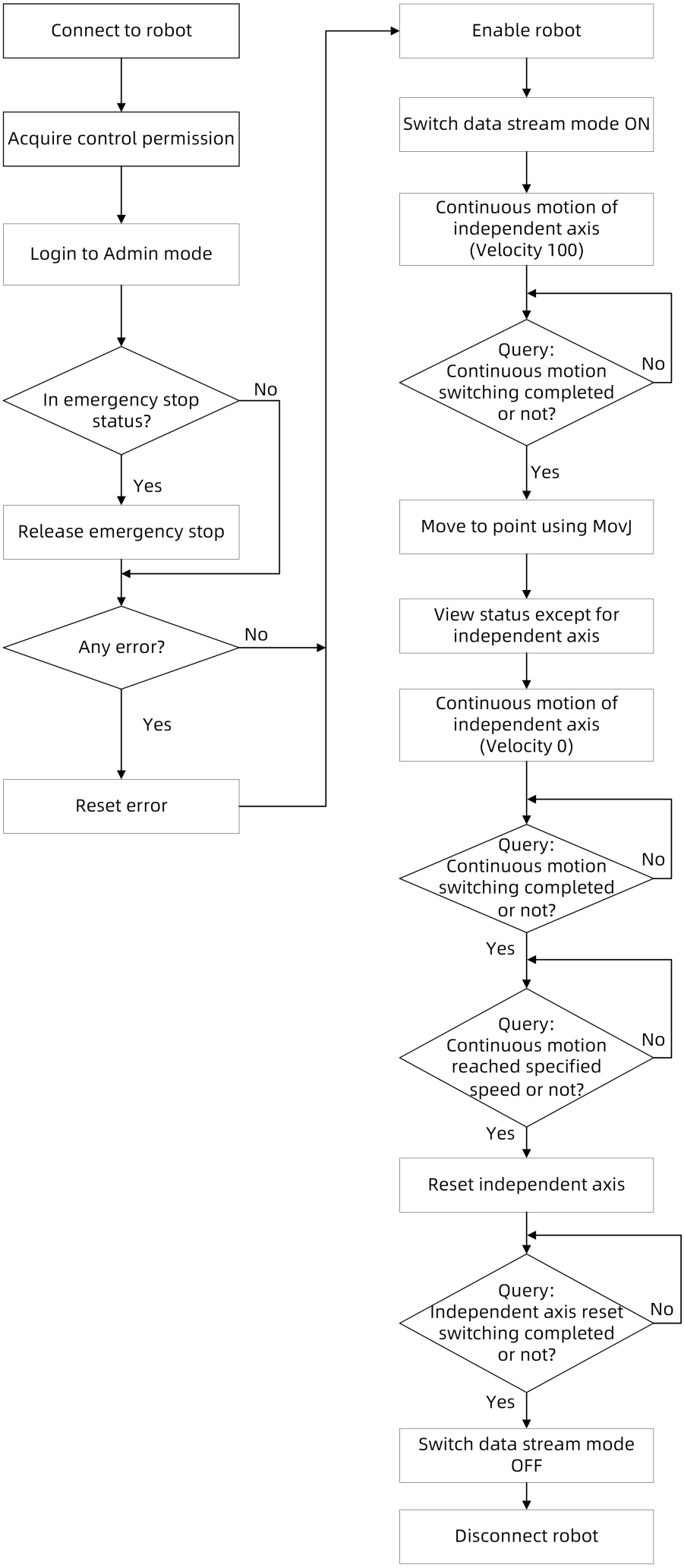
    /* Disconnect the robot */
    ret = IMC100_Exit_ETH(0)
    if(ret < 0)
    {
        // Add exception handlers here
    }
```

5.3 Controlling Independent Axis Motion via Host Controller

Before this operation, first configure the axis to support independent axis. Configuration path:

- InoRobotLab software: Controller parameter configuration > Peripheral configuration > Mechanical unit configuration
- InoTeachPad software: Settings > Peripheral configuration > Mechanical unit configuration

Overall process:


Connect to robot
Acquire control permission
Login to Admin mode
In emergency stop status?
No
Yes
Release emergency stop
Any error?
No
Yes
Reset error
Enable robot
Switch data stream mode ON
Continuous motion of independent axis (Velocity 100)
Query: Continuous motion switching completed or not?
No
Yes
Move to point using MovJ
View status except for independent axis
Continuous motion of independent axis (Velocity 0)
Query: Continuous motion switching completed or not?
No
Yes
Query: Continuous motion reached specified speed or not?
No
Yes
Reset independent axis
Query: Independent axis reset switching completed or not?
No
Yes
Switch data stream mode OFF
Disconnect robot

Code example:

```
int ret = 0;
DWORD dwIP1 = 0xc0a81719;   // Corresponding IP: 192.168.23.25
    int ipPort = 2222;
int timeOut = 5;    // Communication timeout time 5s
    int robotNo = 0;
/* Connect to the robot */
    ret = IMC100_Init_ETH(dwIP1, ipPort, timeOut, robotNo);
if(ret < 0)
    {
        // Add exception handlers here
        return;
}

    int ower = 0;
DWORD IpAddr = 0;
int ipPort = 0; /* Acquire the control permission */
ret = IMC100_CurPermit(&ower, &IpAddr, & ipPort, robotNo);
if(ret < 0)
    {
        // Add exception handlers here
    }
    if(ower != 1)  // Current client device does not have control permission
    {
ret = IMC100_AcqPermit(1, robotNo);   // Forcibly acquire the control permission. Ordinary acquisition is possible when "ower" is 0.
        if(ret < 0)
        {
            // Add exception handlers here
        }
    }

    int sts = 0;      /* Emergency stop status */
    ret = IMC100_Get_EStopSts(&sts, robotNo);
if(ret < 0)
    {
```

```
            // Add exception handlers here
        }
        if(sts == 1)
        {
            int cmd = 0;            /* Release emergency stop */
            ret = IMC100_EmergStop(cmd, robotNo);
if(ret < 0)
        {
            // Add exception handlers here
        }
        }

        int err = 0;       /* Error query */
ret = IMC100_Get_SysErr(&err, robotNo);
if(ret < 0)
        {
            // Add exception handlers here
        }
        if(err != 0)
        {
            /* Reset the error */
            ret = IMC100_ResetErr(robotNo);
if(ret < 0)
        {
            // Add exception handlers here
        }
        }

        cmd = 1; /* Enable */
        ret = IMC100_MotorEnable(cmd, robotNo);
if(ret < 0)
        {
            // Add exception handlers here
        }

        /* Ensure the robot is enabled before enabling data stream mode */
```

```
    int motorEnableSts = 0;
while (0 == motorEnableSts)
    {
        ret = IMC100_Get_MotorSts(&motorEnableSts, robotNo);
if(ret < 0)
    {
        // Add exception handlers here
    }
    Sleep(10); }

    cmd = 1; /* Enable data stream mode */
    ret = IMC100_DsMode(cmd, robotNo);
    if(ret < 0)
    {
        // Add exception handlers here
    }

    char mecUnit[18] = "Robot";
int indAxis = 6;
int indAxisVel = 100;
int indAxisAcc = 50;
int indAxisDec = 50;      /* Continuous motion of the independent axis (velocity 100) */
    ret = IMC100_IndCMove(mecUnit, indAxis, indAxisVel, indAxisAcc, indAxisDec, robotNo);
    if(ret < 0)
    {
        // Add exception handlers here
    }

    int indCMoveSts = 0;     /* Query whether the continuous motion switching of the independent axis is
completed */
    while (0 == indCMoveSts)
    {
        ret = IMC100_Get_IndCMoveSts (mecUnit, indAxis, &indCMoveSts, robotNo);                    if(ret <
0)
    {
        // Add exception handlers here
    }
```

```
    Sleep(10); }

    /* If it is necessary to wait for the independent axis speed to reach the set value, add IMC100_IndSpeed query here */

    ROB_POS pos;
memset(&pos, 0, sizeof(pos));
pos. RPosData[0] = 110;
pos. RPosData[1] = 35;
pos. RPosData[2] = 956;
pos. RPosData[3] = -108;
pos. RPosData[4] = -5;
pos. RPosData[5] = -20;
VER_DATA movVel;
memset(&movVel, 0, sizeof(movVel));
movVel. velPercent = 100 ;
MOV_IO MovIOInfo ;
memset(&MovIOInfo, 0, sizeof(MovIOInfo)) ;
MovIOInfo. IONo = 1 ;
MovIOInfo. IOVa = 1 ;
MovIOInfo. Value =0.005 ;     /* MovJ motion */
    ret = IMC100_MovJ_RobPos(&pos, 100, 0, 1, &MovIOInfo, robotNo)
    if(ret < 0)
    {
        // Add exception handlers here
    }

    /* Motion status check: Wait for motion to start */
/* Note: 1. Check the running status of all axes except for independent axes; 2. It takes a certain amount of time to start after motion data is issued (it is recommended to add timeout handling to avoid missing motion status for repeated points or trajectory segments that are very short). */
    sts = 0;
while (0 == sts)
    {
        ret = IMC100_Get_MotionStsExceptIndAxis (&sts, robotNo);
if(ret < 0)
    {
```

```
            // Add exception handlers here
        }
        Sleep(10); }

        /* Motion status check: Wait for motion to stop */
        sts = 1;
while (1 == sts)
        {
            ret = IMC100_Get_MotionStsExceptIndAxis (&sts, robotNo);
if(ret < 0)
        {
            // Add exception handlers here
        }
        Sleep(10);
}

indAxisVel = 0;         /* Continuous motion of the independent axis (velocity 0) */
        ret = IMC100_IndCMove(mecUnit, indAxis, indAxisVel, indAxisAcc, indAxisDec, robotNo);
        if(ret < 0)
        {
            // Add exception handlers here
        }

indCMoveSts = 0;    /* Query whether the continuous motion switching of the independent axis is completed
*/
        while (0 == indCMoveSts)
        {
            ret = IMC100_Get_IndCMoveSts (mecUnit, indAxis, &indCMoveSts, robotNo);                    if(ret <
0)
        {
            // Add exception handlers here
        }
        Sleep(10); }

indCMoveSts = 0;    /* Query whether the independent axis has reached the set speed */
        while (0 == indCMoveSts)
        {
```

```
        ret = IMC100_IndSpeed (mecUnit, indAxis, &indCMoveSts, robotNo);
if(ret < 0)
    {
        // Add exception handlers here
    }
    Sleep(10); }

/* Independent axis reset */
    ret = IMC100_IndResetOld(mecUnit, indAxis, robotNo);
    if(ret < 0)
    {
        // Add exception handlers here
    }

int indResetSts = 0;        /* Query whether the independent axis reset is complete */
    while (0 == indResetSts)
    {
        ret = IMC100_Get_IndResetSts (mecUnit, indAxis, &indResetSts, robotNo);                    if(ret <
0)
    {
        // Add exception handlers here
    }
    Sleep(100);     // Wait for 100 ms
}

    cmd = 0; /* Disable data stream mode */
    ret = IMC100_DsMode(cmd, robotNo);
    if(ret < 0)
    {
        // Add exception handlers here
    }

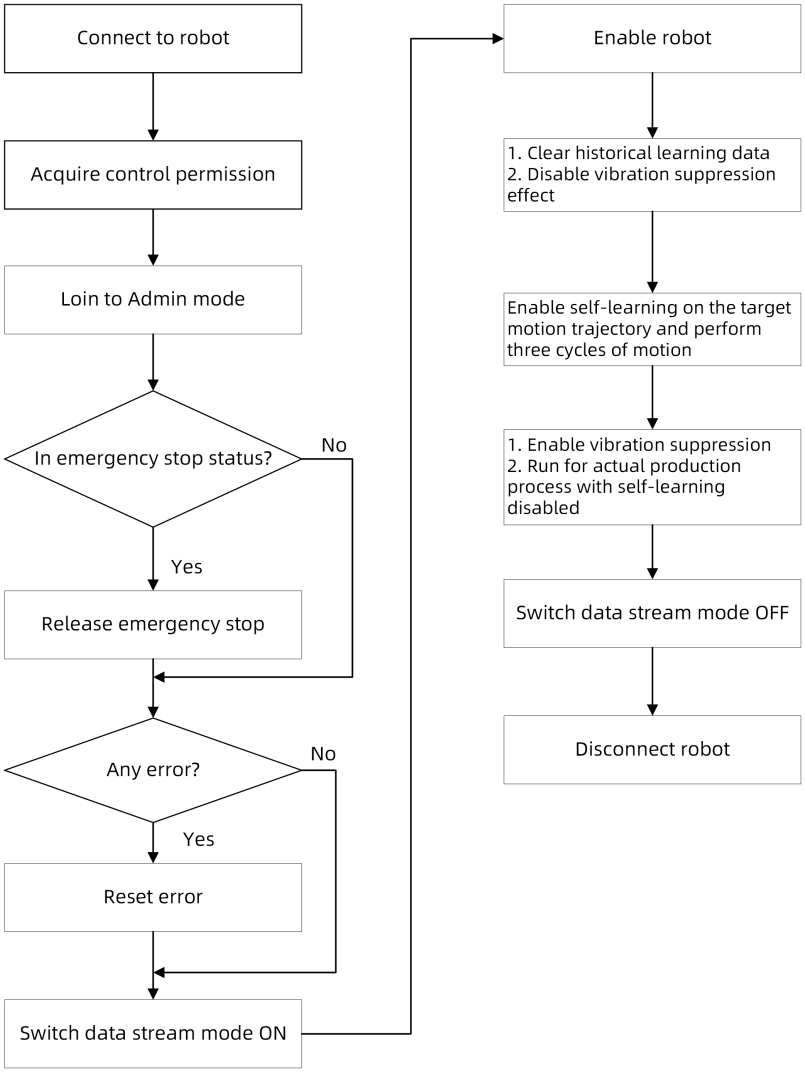
    /* Disconnect the robot */
    ret = IMC100_Exit_ETH(0)
    if(ret < 0)
    {
```

```
        // Add exception handlers here
    }
```

5.4 Controlling Self-Learning Motion via Host Controller

Assuming that global points 1, 2, and 3 have been obtained before this operation, perform cycle motion through these points and suppress vibrations through self-learning.

Overall process:



Code example:

```
int ret = 0;
DWORD dwIP1 = 0xc0a81719;   // Corresponding IP: 192.168.23.25
int ipPort = 2222;
int timeOut = 5;    // Communication timeout time 5s
int robotNo = 0;
/* Connect to the robot */
ret = IMC100_Init_ETH(dwIP1, ipPort, timeOut, robotNo);
if(ret < 0)
{
// Add exception handlers here
```

```
return;
}

int ower = 0;
DWORD IpAddr = 0;
int ipPort = 0; /* Acquire the control permission */
ret = IMC100_CurPermit(&ower, &IpAddr, & ipPort, robotNo);
if(ret < 0)
{
// Add exception handlers here
}
if(ower != 1)  // Current client device does not have control permission
{
ret = IMC100_AcqPermit(1, robotNo);   // Forcibly acquire the control permission. Ordinary acquisition is possible when "ower" is 0.
if(ret < 0)
{
// Add exception handlers here
}
}

int sts = 0;       /* Emergency stop status */
ret = IMC100_Get_EStopSts(&sts, robotNo);
if(ret < 0)
{
// Add exception handlers here
}
if(sts == 1)
{
int cmd = 0;   /* Release emergency stop */
ret = IMC100_EmergStop(cmd, robotNo);
if(ret < 0)
{
// Add exception handlers here
}
}
```

```
int err = 0; /* Error query */
ret = IMC100_Get_SysErr(&err, robotNo);
if(ret < 0)
{
// Add exception handlers here
}
if(err != 0)
{
/* Reset the error */
ret = IMC100_ResetErr(robotNo);
if(ret < 0)
{
// Add exception handlers here
}
}

cmd = 1; /* Enable */
ret = IMC100_MotorEnable(cmd, robotNo);
if(ret < 0)
{
// Add exception handlers here
}

/* Ensure the robot is enabled before enabling data stream mode */
int motorEnableSts = 0;
while (0 == motorEnableSts)
{
ret = IMC100_Get_MotorSts(&motorEnableSts, robotNo);
if(ret < 0)
{
// Add exception handlers here
}
Sleep(10);
}
```

```
cmd = 1; /* Enable data stream mode */
ret = IMC100_DsMode(cmd, robotNo);
if(ret < 0)
{
// Add exception handlers here
}

VER_DATA movVel;
memset(&movVel, 0, sizeof(movVel));
movVel. velPercent = 100 ;
MOV_IO MovIOInfo ;
memset(&MovIOInfo, 0, sizeof(MovIOInfo)) ;

// Step 1: Before self-learning, clear historical learning data and disable vibration suppression effect
ret = IMC100_Set_SLDataClear(2, robotNo);// Clear all learning data (depending on the user requirements)
if(ret < 0)
{
// Add exception handlers here
}
ret = IMC100_Set_SLVSMode(0, robotNo); // Disable vibration suppression effect (ensure the robot is in a vibrating state during learning)
if(ret < 0)
{
// Add exception handlers here
}

// Step 2: Learn the desired motion trajectory three times first, ensuring the learning is completed.
for(int i = 0; i < 3; i++)
{
ret = IMC100_MovJ_P_CS_SL(1, &movVel, 0, 2, 2, 1, 0, &MovIOInfo, robotNo);
if(ret < 0)
{
// Add exception handlers here
}
      ret = IMC100_MovJ_P_CS_SL(2, &movVel, 0, 2, 2, 1, 0, &MovIOInfo, robotNo);
if(ret < 0)
```

```
{
// Add exception handlers here
}
ret = IMC100_MovJ_P_CS_SL(3, &movVel, 0, 2, 2, 1, 0, &MovIOInfo, robotNo);
if(ret < 0)
{
// Add exception handlers here
}
}

// Step 3: Run for the actual production process. Enable vibration suppression, and disable self-learning.
ret = IMC100_Set_SLVSMode(2); // Enable vibration suppression effect, and set vibration suppression effect level as medium
for(int i = 0; i < 10; i++)
{
ret = IMC100_MovJ_P_CS_SL(1, &movVel, 0, 2, 2, 0, 0, &MovIOInfo, robotNo);
if(ret < 0)
{
// Add exception handlers here
}
    ret = IMC100_MovJ_P_CS_SL(2, &movVel, 0, 2, 2, 0, 0, &MovIOInfo, robotNo);
if(ret < 0)
{
// Add exception handlers here
}
ret = IMC100_MovJ_P_CS_SL(3, &movVel, 0, 2, 2, 0, 0, &MovIOInfo, robotNo);
if(ret < 0)
{
// Add exception handlers here
}

}

cmd = 0; /* Disable data stream mode */
ret = IMC100_DsMode(cmd, robotNo);
if(ret < 0)
```

```
{
// Add exception handlers here
}

/* Disconnect the robot */
ret = IMC100_Exit_ETH(0);
if(ret < 0)
{
// Add exception handlers here
}
```

6 API String Call Description

Interaction with the robot controller through API strings supports more systems (e.g., Linux systems) and languages (in addition to the existing VB, VC, C#, and LabView languages supported by API functions).

Steps:

1. Establish communication with the robot controller using the external platform interface (using the robot controller as the server, the controller IP as the IP address, and 2222 as the port number).
2. Use the corresponding function interfaces for string interactions with the robot controller (if you need to control the robot, the robot controller must be switched to "Remote Ethernet client" control permission).

- String request format: @@+request content+$$
- String response format: ##+response content+$$
- For example, to query the current system operation mode: Send the string "@@Get_Mode$$" to the robot controller, the controller responds with the string "##=1$$". After checking the guide, a return value of 1 indicates that the controller is currently in Manual mode.



- For the string request content, see 8.2 Appendix 2: API String Table. Pay attention to spaces and separators when using the strings.
- The controller will actively respond with data each time external data is sent. Do not send data until a response is received.
- If both API functions and strings are used simultaneously, only one client can acquire the control.

7 Common Issues and Troubleshooting

7.1 Issue 1: Robot Controller Error 0x0084

Symptom: When the remote Ethernet control authority is switched to InoTeachPad or InoRobotLab, the robot controller reports error 0x0084.

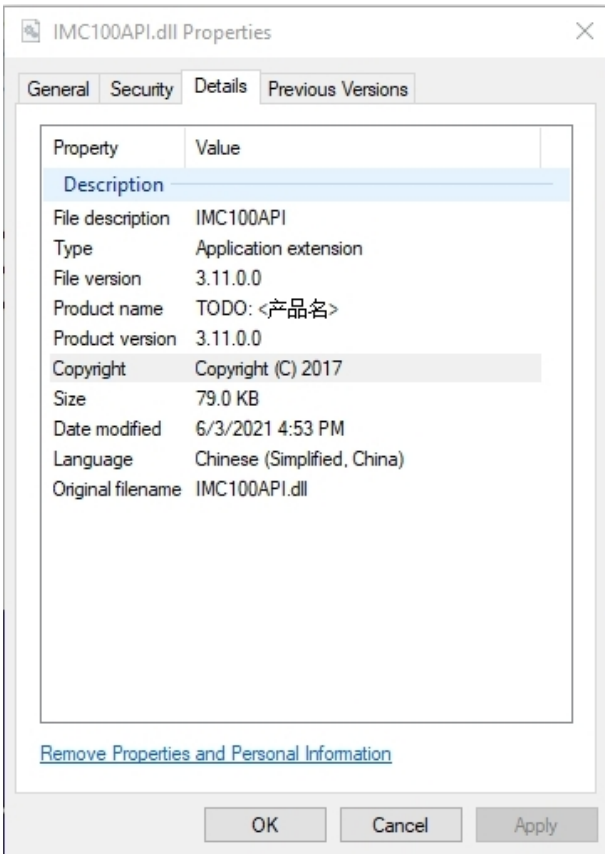
Cause of 0x0084 Error:

- The controller sends data to the API client incorrectly.
- When the API communication is disconnected, the corresponding API control permission has not been released, and the system remains in the enabled state.

To ensure safety, before switching from remote Ethernet control to InoTeachPad/InoRobotLab control, the API client first sends a disable command "IMC100_MotorEnable(0)" to the controller, then releases the API client control permission via "IMC100_RemovePermit", exits the API communication connection via "IMC100_Exit_ETH", and then switches to the teach pendant/InoRobotLab control permission.

7.2 Issue 2: How to Query the API Version Number

Related operation path: Right-click DLL > Properties > Details.



8 Appendix

8.1 Appendix 1: API Function Table

8.1.1 Appendix 1.1 Structures

```
typedef struct{
double RPosData[6];   // Robot position coordinate values
int ArmParm[4];       // Arm parameters
double EPosData[6];   // Robot external axis coordinate values
}ROB_POS;
typedef struct{
double JointData[8];  // Robot joint position coordinate values (the 7th and 8th joint values are reserved)
double EPosData[6];   // Robot external axis coordinate values
}ROB_JPOS;
typedef struct{
double Data[6];      // Position and orientation
}POSE;
typedef struct{
double Mass;         // Mass
double Cog[3];       // Center of gravity position
double Orient[3];    // Orientation
double Inertia[3];   // Load inertia
}LOAD_DATA;
typedef struct{
int RobHold;         // Whether held by robot (1 - grip tool; 0 - external tool)
POSE TFrame;         // Tool frame
// For grip tool, define the position and orientation of TCP in tooldata within the wrist frame (robot end flange)
// For external tool, define the position and orientation of TCP in tooldata within the world frame
LOAD_DATA TLoad;     // Tool load
double Inertia[3];   // Load inertia
}TOOL_DATA;
typedef struct{
int RobHold;         // Whether gripped by robot
// 1 - Workobject held by robot (using external fixed tool)
// 0 - Workobject not held by robot
int UFFix;           // Whether the user frame is fixed
```

// 1 - Fixed user frame (fixed in the world frame or at the robot end)

// 0 - Movable user frame, mounted on the external device, moving with the external device)

char UFMec[18]; // Associated mechanical unit

// Only valid when UFFix = 0, use this parameter to specify which mechanical unit the user frame is located on

double Inertia[3]; // Load inertia

POSE UFrame; // User frame

// In the scenario where the robot holds a tool, define the user frame in the world frame

// In the external tool scenario (Robhold=1 for the workobject), define the user frame in the flange frame (robot end flange)

// A movable user frame (UFFix=0 for the workobject), the user frame is continuously defined by the system and moves with the associated mechanical unit

POSE OFrame; // Workobject frame

// Workobject position, define the workobject frame in the user frame

}WOBJ_DATA;

typedef struct{

double vTcp; // TCP speed

double vOri; // TCP orientation speed

double vLeax; // External axis linear speed

double rReax; // External axis rotational speed

int bStatic; // 1-Speed is not affected by global percentage speed; 0-Speed is affected by global percentage speed

}SPEED;

typedef struct{

int velType; // Speed type (0-Percentage speed type; 1-Numeric speed type)

int velPercent; // Percentage speed, range 1 to 100

SPEED speed; // Numeric speed

}VER_DATA;

typedef struct{

int IONo; // I/O number (upper limit: the actual configured I/O resources of the controller)

int IOVa; // I/O output value (0-OFF, 1-ON)

int Kind; // Set I/O type during motion (0-time type, 1-path percentage, 2-distance)

double Value; // When the type is Time:

// ≥ indicates that signals are output Value seconds after motion starts. < 0 indicates that signals are output Value seconds before arrival at a motion point.

// When the type is Path Percentage:

// ≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals are output when the robot reaches Value% of the path before the motion point.

// When the type is Distance:

// ≥ indicates that signals are output after the robot moves by Value mm from the start point, <0 indicates that signals are output before the robot moves to Value mm from the end point.

}MOV_IO;

typedef struct

{

int ModuleType; // Component type

int PartNum; // Number of component sub-parts

int MaintainStatus; // Maintenance status of component sub-part:

double MaintainValue; // Remaining maintenance time for the component sub-part, or wear percentage for ball screw

double UsedTime; // Used time of component sub-part

double ModuleThreshold; // Maintenance threshold of component sub-part

char LastMaintainTime[9]; // Last maintenance time of component sub-part

}MT_INFO;

8.1.2 Appendix 1.2 Functions

No.	Description	
1	API function	int IMC100_Init_ETH(unsigned int ipAddr,unsigned short ipPort,int timeOut=5, int comId=0)
	Description	Establishes robot network connection.
	Parameter	• ipAddr: Network IP address of the robot controller, in host byte order • ipPort: Network port number of the robot controller, default 2222 • timeOut: Communication timeout setting, default 5s • comId: Connection ID, identifies different connections with the same target IP and port number, default value 0, maximum value 9 (same applies below)

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful connection, while any value less than 0 indicates failure.
	Remarks	• The host controller supports up to 10 different connections; the controller supports up to 4 different connections. • Connection number scope: Same process of the host controller **CAUTION** The connection is considered different if any one of the following differs between the host controller and the controller: IP address, or port number.
2	API function	int IMC100_Exit_ETH(int comId=0)
	Description	Closes the robot network connection.
	Parameter	comId: Connection ID, identifies the corresponding connection (this parameter will not be elaborated on further below)
	Response instruction explanation	A return value of 0 indicates a successful close, while any value less than 0 indicates failure.
	Remarks	-
3	API function	int IMC100_EmergStop(int cmd, int comId=0)
	Description	Emergency stop switch control

No.	Description	
	Parameter	cmd: Emergency stop command • 1: Press the emergency stop switch • 0: Release the emergency stop switch
	Response instruction explanation	A return value of 0 indicates the emergency stop command is completed, while any value less than 0 indicates failure.
	Remarks	-
4	API function	int IMC100_MotorEnable(int cmd, int comId=0)
	Description	Motor enable control
	Parameter	cmd: Motor enable command • 1: Enable • 0: Disable
	Response instruction explanation	A return value of 0 indicates the motor enable command is completed, while any value less than 0 indicates failure.
	Remarks	After the enable command is issued, the enable status can be read after 300 ms.
5	API function	int IMC100_ResetErr(int comId=0)
	Description	Fault reset
	Parameter	-
	Response instruction explanation	A return value of 0 indicates the fault reset command is completed, while any value less than 0 indicates failure.

No.	Description	
	Remarks	The processing time for this instruction is approximately 50 ms.
6	API function	int IMC100_Set_Mode(int mode, int comId=0)
	Description	Sets the system operation mode.
	Parameter	mode: Mode • 1: Manual • 2: Auto
	Response instruction explanation	A return value of 0 indicates a successful mode setting, while any value less than 0 indicates failure.
	Remarks	Controls the Manual/Auto mode switching for the robot project.
7	API function	int IMC100_PrgCtrl(int cmd, int comId=0)
	Description	Project operation control
	Parameter	cmd: Control command • 0: Stop • 1: Start/Resume
	Response instruction explanation	A return value of 0 indicates a successful project operation control, while any value less than 0 indicates failure.
	Remarks	Controls the start/stop operation of the robot project.
8	API function	int IMC100_BackStartLine(int comId=0)
	Description	Returns to the start line of the program.

No.	Description	
	Parameter	-
	Response instruction explanation	A return value of 0 indicates a successful returning to the start line, while any value less than 0 indicates failure.
	Remarks	Controls the main task and dynamic tasks of the robot project to return to the start line.
9	API function	int IMC100_Set_Vel(int val, int comId=0)
	Description	Sets the current operating speed level.
	Parameter	val: Current speed level, range 1 to 100
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
10	API function	int IMC100_Set_AccRamp(double startVal, double endVal , int comId=0)
	Description	Sets the acceleration jerk for the motion segment in data stream mode.
	Parameter	• startVal: Acceleration jerk percentage of the start segment, range 1.0 to 100.0. • endVal: Acceleration jerk percentage of the end segment, range 1.0 to 100.0
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.

No.	Description	
	Remarks	Valid only in data stream mode
11	API function	int IMC100_Set_RapidMove(int movType, int enableFlag, int comId=0);
	Description	Sets the optimal trajectory planning.
	Parameter	• movType: Motion type ◦ 0: cp motion ◦ 1: ptp motion • enableFlag: ◦ 0: Disable optimal planning ◦ 1: Enable optimal planning
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Valid only in data stream mode
12	API function	int IMC100_DsMode(int cmd, int comId=0)
	Description	Data stream mode control
	Parameter	cmd: Data stream command • 0: OFF • 1: ON • 2: Pause • 3: Continue
	Response instruction explanation	A return value of 0 indicates a successful mode control, while any value less than 0 indicates failure.

No.	Description	
	Remarks	When the data stream mode is ON, if the robot is disabled, it will automatically switch to data stream pause mode.
13	API function	int IMC100_Set_DO(int num, int status, int comId=0)
	Description	Sets the DO output status bitwise (The object is the DO under RC's control).
	Parameter	• num: DO bit number • status: DO status ◦ 0: OFF ◦ 1: ON
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
14	API function	int IMC100_Set_SlewMode(int cmd, int comId=0);
	Description	Sets the rotation optimization mode for the J4 axis of the SCARA robot or the J6 axis of the standard 6-axis robot.

No.	Description	
	Parameter	cmd: Rotation optimization mode • 0: No optimization, subject to arm parameters in the position variable. • 1: Optimization mode 1 to ensure that J4/J6 is within the range of -180° to +180° during motion. • 2: Ensure that J4/J6 always moves in the shortest way during motion. The robot calculates and determines whether reaching the target point requires J4/J6 to rotate by 180°. If the angle difference is ≤180°, the target point is fully reached. If >180°, then J4/J6 moves in the opposite direction and ultimately moves to a position where J4/J6 is 360° away from the target point. If the reverse motion of J4/J6 exceeds the limit range of the robot, it will stop the motion, and alarm. • 3: Ensure that J4/J6 always moves in the shortest way during motion, similar to mode 2. The difference is that if the reverse motion of J4/J6 exceeds the limit range of the robot, there will be no alarm. Instead, no reverse motion will be carried out and the original normal motion will be fully adopted.
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Valid only in data stream mode, reset after data stream mode is turned off
15	API function	int IMC100_Set_DOGroup(int num, int status, int comId=0)
	Description	Sets the DO output status by group.
	Parameter	• num: DO group number, range 0 to 15, subject to actual configuration • status: DO status of each group, with a range of 0 to 255, where bit0 to bit7 correspond to the DO

No.	Description	
		status of the group from the smallest to the largest index respectively.
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
16	API function	int IMC100_Set_DA(int num, float val, int comId=0)
	Description	Sets the DA output value by index.
	Parameter	• num: DA index, maximum range 0 to 15 • val: DA value, maximum range 0 mA to 20 mA for current type, -10 V to +10 V for voltage type, subject to DA channel type
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
17	API function	int IMC100_InchMode(int cmd, int comId=0)
	Description	Jog mode control
	Parameter	cmd: Jog mode command • 0: OFF • 1: ON

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful jog mode control, while any value less than 0 indicates failure.
	Remarks	-
18	API function	int IMC100_Set_InchStep(int val, int comId=0)
	Description	Sets the step size level for jog operation.
	Parameter	val: Step size level value, range 0 to 4 • 0: Exit jog mode • 1: Jog level G1, step size 0.05 • 2: Jog level G2, step size 0.3 • 3: Jog level G3, step size 2 • 4: Custom jog step size, the step size is the set value of the jog parameter The jog unit for joint coordinate is °, and the unit for other coordinate is mm
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
19	API function	int IMC100_AxisJog(int axis, int cmd, int comId);
	Description	Single-axis jog command

No.	Description	
	Parameter	• axis: Axis number When the currently activated mechanical unit is Robot, the range is 1 to 6. In joint coordinates, they correspond to the J1 to J6 axes respectively. In non-joint coordinates, they correspond to X/Y/Z/RZ/RY/RX respectively. When the current active mechanical unit is not Robot, the axis range is 1 to the total number of axes of the external mechanical unit. • cmd: Single-axis jog command ◦ 0: Stop ◦ 1: Forward motion ◦ -1: Reverse motion
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	Valid after data stream mode is OFF in manual mode
20	API function	int IMC100_AxisInch(int axis, int cmd, int comId);
	Description	Jog motion command
	Parameter	• axis: Axis number When the currently activated mechanical unit is Robot, the range is 1 to 6. In joint coordinates, they correspond to the J1 to J6 axes respectively. In non-joint coordinates, they correspond to X/Y/Z/RZ/RY/RX respectively. When mecUnit is an external mechanical unit, the axis range is 1 to the total number of axes in the external mechanical unit. • cmd: Single-axis jog command ◦ 0: Stop ◦ 1: Forward jog ◦ -1: Reverse jog

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	Valid after data stream mode is OFF in manual mode
21	API function	int IMC100_PoseAlign(int coord, int cmd, int comId);
	Description	Orientation calibration (for 6-axis robots)
	Parameter	• coord: Frame ◦ -1: Flange frame ◦ 1: Joint frame ◦ 2: Base frame ◦ 3: Tool frame ◦ 4: Workobject frame ◦ 5: World frame (currently supports flange frame (the currently activated tool is an external tool), world frame, base frame, workobject frame) • cmd: Orientation calibration command ◦ 0: Stop orientation calibration ◦ 1: Start orientation calibration
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	Do not perform any other motions during the orientation calibration process. The data stream mode will be automatically turned off after the orientation calibration is performed.
22	API function	int IMC100_Set_ActiveMechUnit(char *mecUnit, int comId=0)
	Description	Sets the active mechanical unit for manual mode motion.

No.	Description	
	Parameter	mecUnit: Mechanical unit name (case-insensitive)
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	-
23	API function	int IMC100_Get_ActiveMechUnit(char *mecUnit, int comId=0)
	Description	Gets the active mechanical unit for manual mode motion.
	Parameter	mecUnit: Mechanical unit name
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	-
24	API function	int IMC100_Set_TeachCoordinate(int flag, int comId=0)
	Description	Sets the coordination switch for manual mode motion.
	Parameter	flag: Coordination switch • 0: Coordination switch OFF • 1: Coordination switch ON
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.

No.	Description	
	Remarks	Conditions for enabling the coordination switch: • The current active mechanical unit is the guide rail, and the current frame is the world frame or workobject frame. • The current active mechanical unit is the positioner, and the current frame is the workobject frame, with the workobject's associated mechanical unit name being the currently active mechanical unit (i.e., the positioner).
25	API function	int IMC100_Get_TeachCoordinate(int *flag, int comId=0)
	Description	Gets the coordination switch for manual mode motion.
	Parameter	flag: Coordination switch • 0: Coordination switch OFF • 1: Coordination switch ON
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	-
26	API function	int IMC100_Home(int num, int comId=0)
	Description	Homing motion command
	Parameter	num: Home index, range 0 to 4

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	Valid only in data stream mode
27	API function	int IMC100_Set_DynamicBrake(int flag, int comId=0)
	Description	Sets the dynamic brake switch (for SCARA robots)
	Parameter	flag: Set dynamic brake switch • 0: Disable dynamic brake switch • 1: Enable dynamic brake switch
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	-
28	API function	int IMC100_Get_DynamicBrake(int flag, int comId=0)
	Description	Reads the dynamic brake switch status (for SCARA robots).
	Parameter	flag: Current obtained dynamic brake switch status
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	-

No.	Description	
29	API function	int IMC100_MovJ_P(int posNum, int vel=100, int zone=0, int comId=0)
	Description	Moves to the global position point with specified number using Movj.
	Parameter	• posNum: Target global position point number, range 0 to 9999 • vel: Motion speed, range 1 to 100, default 100 • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	Valid only in data stream mode
30	API function	int IMC100_MovL_P(int posNum, int vel=100, int zone=0, int comId=0)
	Description	Moves to the global position point with specified number using Movl.
	Parameter	• posNum: Target global position point number, range 0 to 9999 • vel: Motion speed, range 1 to 100, default 100 • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	Valid only in data stream mode

No.	Description	
31	API function	int IMC100_MovC_P(int posMidNum, int posDstNum, int vel=100, int zone=0, int comId=0)
	Description	Moves to the global position point with specified number using Movc.
	Parameter	• posMidNum: Global position point number of a certain point at the middle of the arc, range 0 to 9999 • posDstNum: Global position point number of the arc endpoint, range 0 to 9999 • vel: Motion speed, range 1 to 100, default 100 • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	Valid only in data stream mode
32	API function	int IMC100_MovJ_P_IO(int posNum, int vel, int zone, MOV_IO *movIo, int ioNum, int comId=0)
	Description	Moves to the global position point with specified number using Movj, and controls the corresponding I/O.
	Parameter	• posNum: Target global position point number, range 0 to 9999 • vel: Motion speed, range 1 to 100 • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 • movIo: I/O control structure, see definition for details • ioNum: Number of groups of the I/O control structure, range 1 to 3

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	Valid only in data stream mode
33	API function	int IMC100_MovL_P_IO(int posNum, int vel, int zone, MOV_IO *movIo, int ioNum,int comId=0)
	Description	Moves to the global position point with specified number using Movl, and controls the corresponding I/O.
	Parameter	• posNum: Target global position point number, range 0 to 9999 • vel: Motion speed, range 1 to 100 • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details • ioNum: Number of groups of the I/O control structure, range 1 to 3
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	Valid only in data stream mode
34	API function	int IMC100_MovC_P_IO(int posMidNum, int posDstNum, int vel, int zone, MOV_IO *movIo, int ioNum,int comId=0)
	Description	Moves to the global position point with specified number using Movc, and controls the corresponding I/O.

No.	Description	
	Parameter	• posMidNum: Global position point number of a certain point at the middle of the arc, range 0 to 9999 • posDstNum: Global position point number of the arc endpoint, range 0 to 1000 • vel: Motion speed, range 1 to 100 • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details • ioNum: Number of groups of the I/O control structure, range 1 to 3
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	Valid only in data stream mode
35	API function	int IMC100_MovJ_RobPos(ROB_POS *pos, VER_DATA *vel, int zone=0, int ioNum, MOV_IO *movIo, int comId=0)
	Description	Moves to the position point with specified value using Movj, and controls the corresponding I/O.
	Parameter	• pos: Position parameter structure, see 8.1.1 Appendix 1.1 Structures section for details • vel: Speed parameter structure, see 8.1.1 Appendix 1.1 Structures section for details. • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 • ioNum: Number of groups of the I/O control structure, range 0 to 3 • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	• Valid only in data stream mode • Does not support numeric speed type
36	API function	int IMC100_MovL_RobPos(ROB_POS *pos, VER_DATA *vel, int zone=0, int ioNum, MOV_IO *movIo, int comId=0)
	Description	Moves to the position point with specified value using Movl.
	Parameter	• pos: Position parameter structure, see 8.1.1 Appendix 1.1 Structures section for details • vel: Speed parameter structure, see 8.1.1 Appendix 1.1 Structures section for details. • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 • ioNum: Number of groups of the I/O control structure, range 0 to 3 • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	Valid only in data stream mode
37	API function	int IMC100_MovC_RobPos(ROB_POS *posMid, ROB_POS *posDst, VER_DATA *vel, int zone=0, int ioNum, MOV_IO *movIo, int comId=0)
	Description	Moves to the position point with specified value using Movc.

No.	Description	
	Parameter	• pos: Position parameter structure, see 8.1.1 Appendix 1.1 Structures section for details • vel: Speed parameter structure, see 8.1.1 Appendix 1.1 Structures section for details. • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 • ioNum: Number of groups of the I/O control structure, range 0 to 3 • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	Valid only in data stream mode
38	API function	int IMC100_Jump_RobPos(ROB_POS *pos, VER_DATA *vel, int zone=0, int ioNum, MOV_IO *movIo, int comId=0)
	Description	Moves to the position point with specified value using Jump.
	Parameter	• pos: Position parameter structure, see 8.1.1 Appendix 1.1 Structures section for details • vel: Speed parameter structure, see 8.1.1 Appendix 1.1 Structures section for details. • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 • ioNum: Number of groups of the I/O control structure, range 0 to 3 • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.

No.	Description	
	Remarks	Valid only in data stream mode
39	API function	int IMC100_JumpL_RobPos(ROB_POS *pos, VER_DATA *vel, int zone=0, int ioNum, MOV_IO *movIo, int comId=0)
	Description	Moves to the position point with specified value using JumpL.
	Parameter	• pos: Position parameter structure, see 8.1.1 Appendix 1.1 Structures section for details • vel: Speed parameter structure, see 8.1.1 Appendix 1.1 Structures section for details. • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 • ioNum: Number of groups of the I/O control structure, range 0 to 3 • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	Valid only in data stream mode
40	API function	int IMC100_Jump_P(int posNum, int vel=100, int zone=0, int comId=0)
	Description	Moves to the global position point with specified number using Jump.
	Parameter	• posNum: Target global position point number, range 0 to 9999 • vel: Motion speed, range 1 to 100, default 100, speed parameter structure, see 8.1.1 Appendix 1.1 Structures section for details. • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	Valid only in data stream mode
41	API function	int IMC100_JumpL_P(int posNum, int vel=100, int zone=0, int comId=0)
	Description	Moves to the global position point with specified number using JumpL.
	Parameter	• posNum: Target global position point number, range 0 to 9999 • vel: Motion speed, range 1 to 100, default 100, speed parameter structure, see 8.1.1 Appendix 1.1 Structures section for details. • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	Valid only in data stream mode
42	API function	int IMC100_Jump_P_IO(int posNum, int vel, int zone, MOV_IO *movIo, int ioNum, int comId=0)
	Description	Moves to the global position point with specified number using Jump, and controls the corresponding I/O.

No.	Description	
	Parameter	• posNum: Target global position point number, range 0 to 9999 • vel: Motion speed, range 1 to 100, speed parameter structure, see 8.1.1 Appendix 1.1 Structures section for details. • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details • ioNum: Number of groups of the I/O control structure, range 1 to 3
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	Valid only in data stream mode
43	API function	int IMC100_JumpL_P_IO(int posNum, int vel, int zone, MOV_IO *movIo, int ioNum,int comId=0)
	Description	Moves to the global position point with specified number using JumpL, and controls the corresponding I/O.
	Parameter	• posNum: Target global position point number, range 0 to 9999 • vel: Motion speed, range 1 to 100, speed parameter structure, see 8.1.1 Appendix 1.1 Structures section for details. • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details • ioNum: Number of groups of the I/O control structure, range 1 to 3

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	Valid only in data stream mode
44	API function	int IMC100_MovJAbs_RobJPos(ROB_JPOS *jpos, VER_DATA *vel, int zone=0, int ioNum, MOV_IO *movIo, int comId=0)
	Description	Moves to the joint point with specified value using Movj.
	Parameter	• jpos: Joint position parameter structure, see 8.1.1 Appendix 1.1 Structures section for details • vel: Speed parameter structure, see 8.1.1 Appendix 1.1 Structures section for details. • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 • ioNum: Number of groups of the I/O control structure, range 1 to 3 • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	• Valid only in data stream mode • Does not support numeric speed type
45	API function	int IMC100_MovJAbs_JP(int posNum, VER_DATA *vel, int zone=0, int ioNum, MOV_IO *movIo, int comId=0)
	Description	Moves to the global joint point with specified number using Movj.

No.	Description	
	Parameter	• posNum: Target global joint point number, range 0 to 9999 • vel: Speed parameter structure, see 8.1.1 Appendix 1.1 Structures section for details. • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 • ioNum: Number of groups of the I/O control structure, range 1 to 3 • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	• Valid only in data stream mode • Does not support numeric speed type
46	API function	int IMC100_IndCMove(char *mecUnit, int axis, double speed, double acc,double dec, int comId=0)
	Description	Enables continuous motion of the independent axis.

No.	Description	
	Parameter	• mecUnit: Mechanical unit name. If the mechanical unit is a robot, the parameter is "Robot". • axis: Axis number • For a 6-axis robot, only J6 is supported (for a four-axis robot, only the independent axis of the non-Robot mechanical unit is supported). • For the non-Robot mechanical unit, the range is 1 to the maximum number of axes configured for the mechanical unit. • speed: Numeric speed type, positive and negative signals the direction. The specified speed is affected by the global speed percentage. The formula is: Running speed = Speed set by the instruction x Global speed percentage, in °/s, or mm/s for linear axes. • Range: -10000°/s to 10000°/s (Speed = 0 indicates that continuous motion of the independent axis stops) • acc: acceleration, range 20 to 100 • dec: deceleration, range 20 to 100
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	• Valid only in data stream mode • Call this interface when all axes, except for independent axes, have stopped moving. • When Speed exceeds the upper limit of the controller speed, the upper limit of the controller speed is prevails.
47	API function	int IMC100_IndSpeed(char *mecUnit, int axis, int *sts, int comId=0)
	Description	Checks whether the independent axis reaches the set speed.

No.	Description	
	Parameter	• mecUnit: Mechanical unit name. If the mechanical unit is a robot, the parameter is "Robot". • axis: Axis number • For a 6-axis robot, only J6 is supported (for a four-axis robot, only the independent axis of the non-Robot mechanical unit is supported). • For the non-Robot mechanical unit, the range is 1 to the maximum number of axes configured for the mechanical unit. • sts: Whether the set speed has been reached ◦ 1: Specified speed reached ◦ 0: Specified speed not reached
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	-
48	API function	int IMC100_IndResetOld(char *mecUnit, int axis, int comId=0)
	Description	Resets the independent axis, switches it to the normal mode, and retains the turn count of the axis.
	Parameter	• mecUnit: Mechanical unit name. If the mechanical unit is a robot, the parameter is "Robot". • axis: Axis number • For a 6-axis robot, only J6 is supported (for a four-axis robot, only the independent axis of the non-Robot mechanical unit is supported). • For the non-Robot mechanical unit, the range is 1 to the maximum number of axes configured for the mechanical unit.

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	• Valid only in data stream mode • Call this interface when all axes, except for independent axes, have stopped moving.
49	API function	int IMC100_IndReset(char *mecUnit, int axis, double refNum, int direction, int comId=0)
	Description	Resets the independent axis, switches it to the normal mode, and adjusts the turn count of the axis according to RefPos and Direction.
	Parameter	• mecUnit: Mechanical unit name. If the mechanical unit is a robot, the parameter is "Robot". • axis: Axis number • For a 6-axis robot, only J6 is supported (for a four-axis robot, only the independent axis of the non-Robot mechanical unit is supported). • For the non-Robot mechanical unit, the range is 1 to the maximum number of axes configured for the mechanical unit. • refPos: Reference position, range -720° to 720° • direction: ◦ -1: Adjust the multi-turn count of the axis so that the reference position refNum is on the negative side of the modified current position. (Within 360°) ◦ 0: Adjust the multi-turn count of the axis so that the modified current position is as close as possible to the specified refNum position. (Within ±180°) ◦ 1: Adjust the multi-turn count of the axis so that the reference position refNum is on the positive side of the modified current position. (Within 360°)

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	• Valid only in data stream mode • Call this interface when all axes, except for independent axes, have stopped moving.
50	API function	int IMC100_Get_IndCMoveSts(char *mecUnit, int axis, int *sts, int comId=0)
	Description	Queries the continuous motion switching status of the independent axis.
	Parameter	• mecUnit: Mechanical unit name. If the mechanical unit is a robot, the parameter is "Robot". • axis: Axis number • For a 6-axis robot, only J6 is supported (for a four-axis robot, only the independent axis of the non-Robot mechanical unit is supported). • For the non-Robot mechanical unit, the range is 1 to the maximum number of axes configured for the mechanical unit. • sts: Whether the continuous motion switching of the independent axis is completed ◦ 0: Continuous motion switching of independent axis not completed ◦ 1: Continuous motion switching of independent axis completed
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	-
51	API function	int IMC100_Get_IndResetSts(char *mecUnit, int axis, int *sts, int comId=0)

No.	Description	
	Description	Queries the reset status of the independent axis.
	Parameter	• mecUnit: Mechanical unit name. If the mechanical unit is a robot, the parameter is "Robot". • axis: Axis number • For a 6-axis robot, only J6 is supported (for a four-axis robot, only the independent axis of the non-Robot mechanical unit is supported). • For the non-Robot mechanical unit, the range is 1 to the maximum number of axes configured for the mechanical unit. • sts: Whether the reset status of the independent axis is completed ◦ 0: Reset of independent axis not completed ◦ 1: Reset of independent axis completed
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	-
52	API function	int IMC100_Get_MotionStsExceptIndAxis(int *sts, int comId=0)
	Description	Queries the motion status of the robot, except the independent axis.
	Parameter	sts: System motion status • 0: Stopped/Motion completed • 1: In motion
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.

No.	Description	
	Remarks	-
53	API function	int IMC100_Get_RobPosHere(ROB_POS *pos, int comId=0)
	Description	Gets the current position, which is the value of the current TCP relative to the current workobject. If you need to get the values for different tools or different workobjects, set in advance. • IMC100_Set_ToolCNum • IMC100_Set_WobjNum
	Parameter	pos: Position parameter structure, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
54	API function	int IMC100_Get_RobJPosHere(ROB_JPOS *pos, int comId=0)
	Description	Gets the current joint position.
	Parameter	pos: Joint position parameter structure, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-

No.	Description	
55	API function	int IMC100_Set_ActiveMechUnitPosFormat(int posFormat, int comId=0)
	Description	Sets the position display format of the active mechanical unit.
	Parameter	posFormat: Position format • 1: Joint frame format • 2: Base frame format • 3: Flange frame format • 4: Workobject frame format • 5: World frame format
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
56	API function	int IMC100_Get_ActiveMechUnitPos(int *posFormat, double *mechUnitPos, int comId=0)
	Description	Gets the position of the active mechanical unit.
	Parameter	• posFormat: Position format of the current active mechanical unit ◦ 1: Joint frame format ◦ 2: Base frame format ◦ 3: Flange frame format ◦ 4: Workobject frame format ◦ 5: World frame format • mechUnitPos: mechanical unit position with an array length of 8
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.

No.	Description	
	Remarks	-
57	API function	int IMC100_Get_PosHerePulse(double pos[6], int comId=0)
	Description	Queries the pulse value of the current position point.
	Parameter	pos[]: Current pulse value, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
58	API function	int IMC100_Get_RobJToRobP(ROB_JPOS *pos, int toolnum, int wobjnum, int loadnum, ROB_POS *posDst, int comId=0)
	Description	Calculates the position point based on the joint point.
	Parameter	• pos: Robot joint point position to be converted • toolnum: Tool number • wobjnum: Workobject number • loadnum: Load number • posDst: Converted position point
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
59	API function	int IMC100_Get_RobPToRobJ(ROB_JPOS *pos, int toolnum, int wobjnum, int loadnum, ROB_POS *posDst, int comId=0)

No.	Description	
	Description	Calculates the joint point based on the position point.
	Parameter	• pos: Robot point position to be converted • toolnum: Tool number • wobjnum: Workobject number • loadnum: Load number • posDst: Converted joint point position
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
60	API function	int IMC100_Get_FixWobjRobP(ROB_POS *basePos, int wobjnum, ROB_POS *posDst, int comId=0)
	Description	Converts a position in the world frame into a position in the frame of a designated workobject that is fixed and not held by the robot (RobHold = 0, and UFFix = 1).
	Parameter	• basePos: Robot point position to be converted (point position in the world frame) • wobjnum: Workobject number (fixed workobject: RobHold = 0, UFFix = 1) • posDst: Converted point position in the workobject frame
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
61	API function	int IMC100_Get_RobHoldWobjRobP(ROB_POS *basePos, int wobjnum, ROB_POS *posRef, ROB_POS *posDst, int comId=0)

No.	Description	
	Description	Converts a position in the world frame into a position in the frame of a designated workobject that is held by the robot (RobHold = 1, and UFFix = 1).
	Parameter	• basePos: Robot point position to be converted (point position in the world frame) • wobjnum: Workobject number (gripped workobject: RobHold = 1, UFFix = 1) • posRef: Coordinate value of the flange in the world frame • posDst: Converted point position in the workobject frame CAUTION The arm parameter of P[*] is the same as that of BasePos.
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
62	API function	int IMC100_Get_OffsetJ_RobJP(ROB_JPOS *pos, double PR[6], ROB_JPOS *posDst, int comId=0)
	Description	Queries the point position after joint point offset.
	Parameter	• pos: Original joint point position • PR: Translation variable • posDst: Offset point position result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.

No.	Description	
	Remarks	-
63	API function	int IMC100_Get_Offset_RobP(ROB_POS *pos, double PR[6], ROB_POS *posDst, int comId=0)
	Description	Queries the offset point position in the workobject frame.
	Parameter	• pos: Original point • PR: Translation variable • posDst: Offset point position result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
64	API function	int IMC100_Get_OffsetT_RobP(ROB_POS *pos, double PR[6], ROB_POS *posDst, int comId=0)
	Description	Queries the offset point position in the tool frame.
	Parameter	• posSrc: Original point • PR: Translation variable • posDst: Offset point position result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
65	API function	int IMC100_Get_SysErrSts(int *sts, int comId=0)
	Description	Queries the current fault status of the system.

No.	Description	
	Parameter	sts: System fault status, representing the query result • 0: No fault • bit0: 1 indicates a system alarm, 0 indicates no system alarm • bit1: 1 indicates a system warning, 0 indicates no system warning
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
66	API function	int IMC100_Get_SysErr(int *error, int comId=0)
	Description	Queries the current system error code
	Parameter	error: System error code, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
67	API function	int IMC100_Get_TaskPrgPath(int taskId, char prgPath[128], int comId)
	Description	Queries the program path for the specified task channel in the robot project.
	Parameter	• taskId: Task channel, range: 0 to maximum number of tasks configured in the project, where 0 is the main task • prgPath: Current program path, representing the query result

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
68	API function	int IMC100_Get_TaskRunSts(int taskId, int *sts, int comId)
	Description	Queries the running status of the specified task channel in the robot project.
	Parameter	• taskId: Task channel, range: [0-15], where 0 is the main task • sts: Running status, representing the query result ◦ 0: Stop ◦ 1: Start/Resume ◦ 10: Ready ◦ 100: Task not activated ◦ -1: Task activated but program not configured
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
69	API function	int IMC100_Get_TaskProgramLine(int taskId, int *line, int comId=0)
	Description	Queries the number of the current processing line in the task channel program.

No.	Description	
	Parameter	line: Number of the current processing line in the current task, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
70	API function	int IMC100_Get_CurMotionLine(int *line, int comId=0)
	Description	Queries the line number of the current executing motion instruction.
	Parameter	line: Line number of the current executing motion instruction, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
71	API function	int IMC100_Get_InitSts(int *sts, int comId=0)
	Description	Queries the system initialization status.
	Parameter	sts: System initialization status, representing the query result, with a range of -1 to 11
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-

No.	Description	
72	API function	int IMC100_Get_AccRamp(double *startVal, double *endVal, int comId =0);
	Description	Queries the acceleration jerk for the motion segment in data stream mode.
	Parameter	• startVal: Acceleration jerk percentage of the start segment, range 1.0 to 100.0. • endVal: Acceleration jerk percentage of the end segment, range 1.0 to 100.0
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
73	API function	int IMC100_Get_RapidMove(int movType, int *enableFlag, int comId =0)
	Description	Queries the optimal trajectory switch for the current motion type.
	Parameter	• movType: Motion type ◦ 0: cp motion ◦ 1: ptp motion • enableFlag ◦ 0: Disable optimal planning for current motion type ◦ 1: Enable optimal trajectory for current motion
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
74	API function	int IMC100_Get_CoordType(int *type, int comId=0)

No.	Description	
	Description	Queries the current frame type.
	Parameter	type: Current frame type, range 1 to 5 • 1: Joint frame • 2: Base frame • 3: Tool frame • 4: Workobject frame • 5: World frame
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
75	API function	int IMC100_Get_Vel(int *val, int comId=0)
	Description	Queries the current global speed value.
	Parameter	val: Current global speed value, representing the query result, with a range of 1 to 100
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
76	API function	int IMC100_Get_Mode(int *mode, int comId=0)
	Description	Queries the current operating mode of the system.

No.	Description	
	Parameter	mode: System operating mode, representing the query result • 1: Manual • 2: Auto • 3: Single step • 5: Continuous
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
77	API function	int IMC100_Get_DsMode(int *val, int comId=0)
	Description	Queries whether the data stream mode is enabled.
	Parameter	val: Status of the data stream mode, representing the query result • 0: OFF • 1: ON/Continue • 2: Pause
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
78	API function	int IMC100_Get_InchMode(int *val, int comId=0)
	Description	Queries the jog mode.

No.	Description	
	Parameter	val: Jog mode, representing the query result • 0: Non-jog • 1: Jog
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
79	API function	int IMC100_Get_SlewMode(int *val, int comId=0)
	Description	Queries the rotation optimization mode for the J4 axis of the SCARA robot or the J6 axis of the standard 6-axis robot.
	Parameter	val: Rotation optimization mode for the J4 axis of the SCARA robot or the J6 axis of the standard 6-axis robot, representing the query result • 0: No optimization, subject to arm parameters in the position variable. • 1: Optimization mode 1 to ensure that J4/J6 is within the range of -180° to +180° during motion. • 2: Ensure that J4/J6 always moves in the shortest way during motion. The robot calculates and determines whether reaching the target point requires J4/J6 to rotate by 180°. If the angle difference is ≤180°, the target point is fully reached. If >180°, then J4/J6 moves in the opposite direction and ultimately moves to a position where J4/J6 is 360° away from the target point. If the reverse motion of J4/J6 exceeds the limit range of the robot, it will stop the motion, and alarm. • 3: Ensure that J4/J6 always moves in the shortest way during motion, similar to mode 2. The difference is that if the reverse motion of J4/J6 exceeds the limit range of the robot, there will be no alarm. Instead, no reverse motion will be

No.	Description	
		carried out and the original normal motion will be fully adopted.
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
80	API function	int IMC100_Get_EStopSts(int *sts, int comId=0)
	Description	Queries the current status of the emergency stop switch.
	Parameter	sts: Emergency stop switch status, representing the query result • 0: Emergency stop switch released • 1: Emergency stop switch pressed
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-

No.	Description	
81	API function	int IMC100_Get_MotorSts(int *sts, int comId=0)
	Description	Queries the current motor enable status.
	Parameter	sts: Motor enable status, representing the query result • 0: Disable • 1: Enable
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
82	API function	int IMC100_Get_MotionSts(int *sts, int comId=0)
	Description	Queries the current system motion status.
	Parameter	sts: System motion status, representing the query result • 0: Stopped/Motion completed • 1: In motion • 2: Motion interrupted
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
83	API function	int IMC100_Get_SysMode(int *mode, int comId=0)
	Description	Queries the current system mode.

No.	Description	
	Parameter	mode: System mode, representing the query result • 0: Normal mode • >0: Internal test mode
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
84	API function	int IMC100_Get_PrgRunTime(unsigned int *second, int comId=0)
	Description	Queries the project runtime.
	Parameter	second: Time count value (s), representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
85	API function	int IMC100_Get_CurCmdNum(unsigned int *num, int comId=0)
	Description	Queries the number of the currently successfully sent motion instruction.
	Parameter	num: Instruction number, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.

No.	Description	
	Remarks	Valid only in data stream mode
86	API function	int IMC100_Get_CurCmdSts(int *sts, int comId=0)
	Description	Queries the actual completion status (whether in position) of the currently successfully sent motion instruction.
	Parameter	sts: Completion status, representing the query result • 0: Motion not completed • 1: Motion completed
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	• Valid only in data stream mode • This interface determines the running status of the robot except for independent axes. If the last motion instruction is for an independent axis, use IMC100_IndSpeed to determine whether the position is reached. • Do not use IMC100_Get_CurCmdSts to determine the in-position status after sending duplicate points.
87	API function	int IMC100_Get_CurCmdCacheNum(int *num, int comId);
	Description	Queries the number of motion data items in the current cache.
	Parameter	num: Number of motion data items in the current cache
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.

No.	Description	
	Remarks	-
88	API function	int IMC100_Get_CmdSts(int num, int *sts, int comId=0)
	Description	Queries the actual completion status of a motion instruction with the specified number.
	Parameter	• num: Instruction number • sts: Completion status, representing the query result ◦ 0: Motion not completed ◦ 1: Motion completed
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	Valid only in data stream mode
89	API function	int IMC100_Get_DINum(int *num, int comId=0)
	Description	Queries the total number of system DIs.
	Parameter	num: Total number of DIs, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
90	API function	int IMC100_Get_DONum(int *num, int comId=0)
	Description	Queries the total number of system DOs.

No.	Description	
	Parameter	num: Total number of DOs, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
91	API function	int IMC100_Get_ADNum(int *num, int comId=0)
	Description	Queries the total number of system ADs (including the total number of ADs for IR-Link and EtherCAT).
	Parameter	num: Total number of ADs, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
92	API function	int IMC100_Get_DANum(int *num, int comId=0)
	Description	Queries the total number of system DAs (the result is the total number of activated DAs for IR-Link and EtherCAT).
	Parameter	num: Total number of DAs, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-

No.	Description	
93	API function	int IMC100_Get_DI(int num, int *sts, int comId=0)
	Description	Queries the DI input status bitwise.
	Parameter	• num: DI index (not exceeding the total number of DIs) • sts: DI status, representing the query result ◦ 0: OFF ◦ 1: ON
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
94	API function	int IMC100_Get_DIGroup(int num, int *sts, int comId=0)
	Description	Queries the DI input status by group.
	Parameter	• num: DI group number • sts: DI status of each group, representing the query result, with a range of 0 to 255, where bit0 to bit7 correspond to the DI status of the group from the smallest to the largest index respectively.
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
95	API function	int IMC100_Get_AD(int num, float *val, int comId=0)

No.	Description	
	Description	Queries the AD input value by index.
	Parameter	• num: AD index (range for IR-Link: 0 to total number of activated ADs (upper limit ≤15) for IR-Link, range for EtherCAT: 64 to total number of activated ADs (upper limit ≤79) for EtherCAT) • val: AD value, representing the query result, in mA for current type and V for voltage type
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
96	API function	int IMC100_Get_DOCfg(int num, int *val, int comId=0)
	Description	Queries the configuration authority of DO.
	Parameter	• num: DO index (not exceeding the total number of DOs) • val: Configuration authority, representing the query result ◦ 1: Controlled by RC ◦ 0: Controlled by PLC
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
97	API function	int IMC100_Get_DOGroupCfg(int num, int *val, int comId=0)
	Description	Queries the configuration authority of the DOs in each group.

No.	Description	
	Parameter	• num: DO group number • val: Configuration authority, representing the query result ◦ bit0 to bit7 represent the configuration authority of each DO in the group ◦ 1: Controlled by RC ◦ 0: Controlled by PLC
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
98	API function	int IMC100_Get_DO(int num, int *sts, int comId=0)
	Description	Queries the DO output status bitwise.
	Parameter	• num: DO index (not exceeding the total number of DOs) • sts: DO status, representing the query result ◦ 0: OFF ◦ 1: ON
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
99	API function	int IMC100_Get_DOGroup(int num, int *sts, int comId=0)
	Description	Queries the DO output status by group.

No.	Description	
	Parameter	• num: DO group number • sts: DO status of each group, representing the query result, with a range of 0 to 255, where bit0 to bit7 correspond to the DO status of the group from the smallest to the largest index respectively.
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
100	API function	int IMC100_Get_DACfg(int num, int *val, int comId=0)
	Description	Queries the configuration authority of DA.
	Parameter	• num: DA index (range for IR-Link: 0 to total number of activated DAs (upper limit ≤15) for IR-Link, range for EtherCAT: 64 to total number of activated DAs (upper limit ≤79) for EtherCAT) • val: Configuration authority, representing the query result, where 1 indicates controlled by RC, 0 indicates not controlled by RC (controlled by PLC or not connected)
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
101	API function	int IMC100_Get_DA(int num, float *val, int comId=0)
	Description	Queries the DA output value by index.

No.	Description	
	Parameter	• num: DA index (range for IR-Link: 0 to total number of activated DAs (upper limit ≤15) for IR-Link, range for EtherCAT: 64 to total number of activated DAs (upper limit ≤79) for EtherCAT) • val: DA value, representing the query result, in mA for current type and V for voltage type
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
102	API function	int IMC100_Get_DevSts(int sts[6], int comId=0)
	Description	Queries the connection status of system devices.
	Parameter	sts[]: Connection status of system device, representing the query result Description: • sts[0]: Connection status of network card 1, where 0 indicates not connected, 1 indicates connected, and 2 indicates disabled • sts[1]: Connection status of network card 2, same as above • sts[2]: USB device status, where 0 indicates not connected, 1 indicates connected and mounted successfully, and 2 indicates mounting failed • sts[3]: SD card status, where 0 indicates not connected, 1 indicates connected and mounted successfully, 2 indicates mounting failed, and 3 indicates file system format error • sts[4]: EtherCAT1 communication status, where 0 indicates normal communication, 1 indicates slave device offline, 2 indicates no network cable connected, 3 indicates connected to non-EtherCAT device, and 4 indicates disabled • sts[5]: IR-Link1 communication status, where 0 indicates normal communication, 1 indicates slave device offline, 2 indicates device not mounted, 3 indicates connected to a non-IR-

No.	Description	
		Link device, 4 indicates disabled, 5 indicates device not configured, 6 indicates communication abnormal
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
103	API function	int IMC100_Get_FwVersion(char ver[32], int comId=0)
	Description	Queries the system controller software version.
	Parameter	ver[]: Current system software version, representing the query result, for example S03.20R
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-

No.	Description	
104	API function	int IMC100_Get_SysTime(char time[16], int comId=0)
	Description	Queries the current system time.
	Parameter	time[]: Time string (including year, month, day, hour, minute, and second), representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
105	API function	int IMC100_Get_RobotType(char type[128], int comId=0)
	Description	Queries the current system model.
	Parameter	type[]: Model string, representing the query result, for example Scara_A_Ino1
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
106	API function	int IMC100_Get_ArmType(double pos[6], int armType[4], int comId=0)
	Description	Queries the arm parameters based on the joint coordinate values.
	Parameter	pos[]: Joint coordinate values armType[]: Arm parameters, representing the query result

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
107	API function	int IMC100_Get_ServoSts(int sts[8], int comId=0)
	Description	Queries the fault status of all servos in the system (including both robot axes and external axes).
	Parameter	sts[8]: Servo fault status, representing the query result. Currently, a maximum of 8 servo axes are supported. sts[0] corresponds to servo axis 0, and so on. • 0: No fault • bit0: 1 for servo alarm, 0 for no servo alarm • bit1: 1 for servo warning, 0 for no servo warning
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
108	API function	int IMC100_Get_ServoErr(int num, int *error, int comId=0)
	Description	Queries the error code of a single servo (robot axis) in the system.
	Parameter	• num: Servo axis number, starting from 0 • error: Servo error code, representing the query result

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
109	API function	int IMC100_Get_StrPara(float para[6], int comId=0)
	Description	Queries the robot structure parameters.
	Parameter	para[]: Structure parameters, representing the query result (for SCARA robots, para[0] to para[3] are valid; for 6-axis robots, para[0] to para[5] are valid; the same applies below)
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
110	API function	int IMC100_Set_StrPara(float para[6], int comId=0)
	Description	Sets the robot structure parameters.
	Parameter	para[]: Structure parameters
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes (switching modes by calling the function IMC100_UserLogin, the same applies below)

No.	Description	
111	API function	int IMC100_Get_StrParaComp(float para[6], int comId=0)
	Description	Queries the robot structure compensation parameters.
	Parameter	para[]: Structure compensation parameters, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
112	API function	int IMC100_Set_StrParaComp(float para[6], int comId=0)
	Description	Sets the robot structure compensation parameters.
	Parameter	para[]: Structure compensation parameters
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
113	API function	int IMC100_Get_RdctRatio(float para[6], int comId=0)
	Description	Queries the joint reduction ratio.
	Parameter	para[]: Joint reduction ratio, representing the query result

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
114	API function	int IMC100_Set_RdctRatio(float para[6], int comId=0)
	Description	Sets the joint reduction ratio.
	Parameter	para[]: Reduction ratio of each joint
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
115	API function	int IMC100_Get_CpParaM(float para[6], int comId=0)
	Description	Queries the joint master coupling parameters.
	Parameter	para[]: Master coupling parameters of each joint, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
116	API function	int IMC100_Set_CpParaM(float para[6], int comId=0)

No.	Description	
	Description	Sets the joint master coupling parameters.
	Parameter	para[]: Master coupling parameters of each joint
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
117	API function	int IMC100_Get_CpParaS(float para[6], int comId=0)
	Description	Queries the joint slave coupling parameters.
	Parameter	para[]: Slave coupling parameters of each joint, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
118	API function	int IMC100_Set_CpParaS(float para[6], int comId=0)
	Description	Sets the joint slave coupling parameters.
	Parameter	para[]: Slave coupling parameters of each joint
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.

No.	Description	
	Remarks	Available for Admin and Manufacturer modes.
119	API function	int IMC100_Get_HomeJPos(int num, ROB_JPOS *pos, int comId=0)
	Description	Queries the work origin.
	Parameter	• num: Work origin index, range 0 to 4 • pos[]: Joint coordinate values corresponding to the work origin, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
120	API function	int IMC100_Set_HomeJPos(int num, ROB_JPOS *pos, int comId=0)
	Description	Sets the work origin.
	Parameter	• num: Work origin index, range 0 to 4 • pos[]: Joint coordinate values corresponding to the work origin
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
121	API function	int IMC100_Get_ZeroPos(int pluse[6], int comId=0)
	Description	Queries the absolute zero point.

No.	Description	
	Parameter	pluse[]: Pulse value corresponding to the absolute zero point, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
122	API function	int IMC100_Set_ZeroPos(int pluse[6], int comId=0)
	Description	Sets the absolute zero point.
	Parameter	pluse[]: Pulse value corresponding to the absolute zero point
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
123	API function	int IMC100_Get_InchStep(int *val, int comId=0)
	Description	Queries the jog stepsize level.
	Parameter	val: Stepsize level for jog operation, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-

No.	Description	
124	API function	int IMC100_Get_StepMotionJ(float *para, int comId=0)
	Description	Queries the custom joint stepsize for jog operation.
	Parameter	para: Custom joint stepsize value, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
125	API function	int IMC100_Set_StepMotionJ(float para, int comId=0)
	Description	Sets the custom joint stepsize for jog operation.
	Parameter	para: Custom joint stepsize value, range 0.01 to 10
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
126	API function	int IMC100_Get_StepMotionL(float *para, int comId=0)
	Description	Queries the custom linear stepsize for jog operation.
	Parameter	para: Custom linear stepsize value, representing the query result

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
127	API function	int IMC100_Set_StepMotionL(float para, int comId=0)
	Description	Sets the custom linear stepsize for jog operation.
	Parameter	para: Custom linear stepsize value, range 0.01 to 10
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
128	API function	int IMC100_Get_StepMotionR(float *para, int comId=0)
	Description	Queries the custom orientation stepsize for jog operation.
	Parameter	para: Custom orientation stepsize value, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
129	API function	int IMC100_Set_StepMotionR(float para, int comId=0)

No.	Description	
	Description	Sets the custom orientation stepsize for jog operation.
	Parameter	para: Custom orientation stepsize value, range 0.01 to 10
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
130	API function	int IMC100_Get_TeachVelLimJ(float para[6], int comId=0)
	Description	Queries the maximum joint speed limit for manual mode motion.
	Parameter	para[]: Maximum allowable joint speed, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
131	API function	int IMC100_Set_TeachVelLimJ(float para[6], int comId=0)
	Description	Sets the maximum joint speed limit for manual mode motion.
	Parameter	para[]: Maximum allowable joint speed
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.

No.	Description	
	Remarks	Available for Admin and Manufacturer modes.
132	API function	int IMC100_Get_TeachVelLimL(float para[2], int comId=0)
	Description	Queries the maximum position and orientation speed limits for manual mode motion.
	Parameter	para[2]: Maximum allowable position and orientation speeds, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
133	API function	int IMC100_Set_TeachVelLimL(float para[2], int comId=0)
	Description	Sets the maximum position and orientation speed limits for manual mode motion.
	Parameter	para[2]: Maximum allowable position and orientation speeds
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
134	API function	int IMC100_Get_TeachAccLimJ(float para[6], int comId=0)
	Description	Queries the maximum joint acceleration limit for manual mode motion
	Parameter	para[]: Maximum allowable joint acceleration, representing the query result

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
135	API function	int IMC100_Set_TeachAccLimJ(float para[6], int comId=0)
	Description	Sets the maximum joint acceleration limit for manual mode motion.
	Parameter	para[]: Maximum allowable joint acceleration
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
136	API function	int IMC100_Get_TeachAccLimL(float para[2], int comId=0)
	Description	Queries the maximum position and orientation acceleration limits for manual mode motion.
	Parameter	para[]: Maximum allowable position and orientation accelerations, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
137	API function	int IMC100_Set_TeachAccLimL(float para[2], int comId=0)

No.	Description	
	Description	Sets the maximum position and orientation acceleration limits for manual mode motion.
	Parameter	para[]: Maximum allowable position and orientation accelerations
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
138	API function	int IMC100_Get_RunVelLimJ(float para[6], int comId=0)
	Description	Queries the maximum joint speed limit during runtime.
	Parameter	para[]: Maximum allowable joint speed, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
139	API function	int IMC100_Set_RunVelLimJ(float para[6], int comId=0)
	Description	Sets the maximum joint speed limit during runtime.
	Parameter	para[]: Maximum allowable joint speed
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.

No.	Description	
	Remarks	Available for Admin and Manufacturer modes.
140	API function	int IMC100_Get_RunVelLimL(float para[2], int comId=0)
	Description	Queries the maximum position and orientation speed limits during runtime.
	Parameter	para[]: Maximum allowable position and orientation speeds, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
141	API function	int IMC100_Set_RunVelLimL(float para[2], int comId=0)
	Description	Sets the maximum position and orientation speed limits during runtime.
	Parameter	para[]: Maximum allowable position and orientation speeds
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
142	API function	int IMC100_Get_RunAccLimJ(float para[6], int comId=0)
	Description	Queries the maximum joint acceleration limit during runtime.
	Parameter	para[]: Maximum allowable joint acceleration, representing the query result

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
143	API function	int IMC100_Set_RunAccLimJ(float para[6], int comId=0)
	Description	Sets the maximum joint acceleration limit during runtime.
	Parameter	para[]: Maximum allowable joint acceleration
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
144	API function	int IMC100_Get_RunAccLimL(float para[2], int comId=0)
	Description	Queries the maximum position and orientation acceleration limits during runtime.
	Parameter	para[]: Maximum allowable position and orientation accelerations, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
145	API function	int IMC100_Set_RunAccLimL(float para[2], int comId=0)

No.	Description	
	Description	Sets the maximum position and orientation acceleration limits during runtime.
	Parameter	para[2]: Maximum allowable position and orientation accelerations
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
146	API function	int IMC100_Get_StopDecLimJ(float para[6], int comId=0)
	Description	Queries the maximum joint deceleration limit during runtime.
	Parameter	para[]: Maximum allowable joint deceleration, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
147	API function	int IMC100_Set_StopDecLimJ(float para[6], int comId=0)
	Description	Sets the maximum joint deceleration limit during runtime.
	Parameter	para[]: Maximum allowable joint deceleration speed
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.

No.	Description	
	Remarks	Available for Admin and Manufacturer modes.
148	API function	int IMC100_Get_StopDecLimL(float para[2], int comId=0)
	Description	Queries the maximum position and orientation deceleration limits during runtime.
	Parameter	para[]: Maximum allowable position and orientation decelerations, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
149	API function	int IMC100_Set_StopDecLimL(float para[2], int comId=0)
	Description	Sets the maximum position and orientation deceleration limits during runtime.
	Parameter	para[]: Maximum allowable position and orientation decelerations
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
150	API function	int IMC100_Get_ZonePara(float para[2], int comId=0)
	Description	Queries the transition accuracy parameters.
	Parameter	para[]: Linear and joint transition accuracy, representing the query result

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
151	API function	int IMC100_Set_ZonePara(float para[2], int comId=0)
	Description	Sets the transition accuracy parameters: linear and joint.
	Parameter	para[]: Linear and joint transition accuracy
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
152	API function	int IMC100_Get_AxisNLim(int axis, float *para, int comId=0)
	Description	Queries the negative limit of the robot axis.
	Parameter	• axis: Axis number, related to the number of axes, range 1 to 6, corresponding to J1 to J6 axes respectively • para: Negative limit value, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-

No.	Description	
153	API function	int IMC100_Set_AxisNLim(int axis, float para, int comId=0)
	Description	Sets the negative limit of the robot axis.
	Parameter	• axis: Axis number, range 1 to 6, corresponding to J1 to J6 axes respectively • para: Negative limit value
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
154	API function	int IMC100_Get_AxisPLim(int axis, float *para, int comId=0)
	Description	Queries the positive limit of the robot axis.
	Parameter	• axis: Axis number, range 1 to 6, corresponding to J1 to J6 axes respectively • para: Positive limit value, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
155	API function	int IMC100_Set_AxisPLim(int axis, float para, int comId=0)
	Description	Sets the positive limit of the robot axis.

No.	Description	
	Parameter	• axis: Axis number, range 1 to 6, corresponding to J1 to J6 axes respectively • para: Positive limit value
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
156	API function	int IMC100_Get_ToolData(int num, TOOL_DATA *toolData, int comId=0)
	Description	Queries the tool frame parameters.
	Parameter	• num: Tool number, range 0 to 15 • toolData: Tool parameters, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
157	API function	int IMC100_Set_ToolData(int num, TOOL_DATA *toolData, int comId=0)
	Description	Sets the tool frame parameters.
	Parameter	• num: Tool number, range 1 to 15 • toolData: Tool parameters
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.

No.	Description	
	Remarks	Available for Admin and Manufacturer modes.
158	API function	int IMC100_Get_WobjData(int num, WOBJ_DATA *wobjData, int comId=0)
	Description	Queries the workobject frame parameters.
	Parameter	• num: Workobject number, range 0 to 15 • wobjData: Workobject parameters, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
159	API function	int IMC100_Set_WobjData(int num, WOBJ_DATA *wobjData, int comId=0)
	Description	Sets the workobject frame parameters.
	Parameter	• num: Workobject number, range 1 to 15 • wobjData: Workobject parameters (associated mechanical unit, case-insensitive)
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
160	API function	int IMC100_Get_ToolCNum(int *num, int comId=0)
	Description	Queries the current active tool number in the system.

No.	Description	
	Parameter	num: Tool number, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
161	API function	int IMC100_Set_ToolCNum(int num, int comId=0)
	Description	Sets the active tool number in the system.
	Parameter	num: Tool number, range 0 to 15
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
162	API function	int IMC100_Get_WobjNum(int *num, int comId=0)
	Description	Queries the current active workobject number in the system.
	Parameter	num: Workobject number, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-

No.	Description	
163	API function	int IMC100_Set_WobjNum(int num, int comId=0)
	Description	Sets the active workobject number in the system.
	Parameter	num: Workobject number, range 0 to 15
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
164	API function	int IMC100_Set_CoordType(int type, int comId=0)
	Description	Sets the current frame type.
	Parameter	type: Current frame type, range 1 to 5 • 1: Joint frame • 2: Base frame • 3: Tool frame • 4: Workobject frame • 5: World frame
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
165	API function	int IMC100_Get_Interf(int num, double pos[6], int comId=0)
	Description	Queries the coordinate parameters for the boundary points of the interference zone.

No.	Description	
	Parameter	• num: Interference zone number, range 0 to 7 • pos[]: Coordinates of the interference zone boundary points, representing the query result. pos[0] to pos[2] correspond to the XYZ coordinates of point 1, and pos[3] to pos[5] correspond to the XYZ coordinates of point 2
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
166	API function	int IMC100_Set_Interf(int num, double pos[6], int comId=0)
	Description	Sets the coordinate parameters for the boundary points of the interference zone.
	Parameter	• num: Interference zone number • pos[]: Coordinates of interference zone boundary points
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
167	API function	int IMC100_Get_CurInterf(int *num, int comId=0)
	Description	Queries the current active interference zone number.
	Parameter	num: Interference zone number, representing the query result, range 0 to 255. • bit0 to bit7 correspond to interference zones 0 to 7 respectively. • 0: Inactive

No.	Description	
		• 1: Active
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
168	API function	int IMC100_Set_CurInterf(int num, int comId=0)
	Description	Sets the interference zone number to be activated.
	Parameter	num: interference zone number, as above
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
169	API function	int IMC100_Get_JumpPara(float *lh, float *mh, float *rh, int comId=0)
	Description	Queries the height parameters for jump motion (Jump, JumpL).
	Parameter	• lh: Lift height at the start position, range 0 to 2000, representing the query result • mh: Highest motion height relative to the zero point of the base frame, range -2000 to 2000, representing the query result • rh: Descent height to the end position, range 0 to 2000, representing the query result

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
170	API function	int IMC100_Set_JumpPara(float lh, float mh, float rh, int comId=0)
	Description	Sets the height parameters for jump motion (Jump, JumpL).
	Parameter	• lh: Lift height at the start position, range 0 to 2000 • mh: Highest motion height relative to the zero point of the base frame, range -2000 to 2000 • rh: Descent height to the end position, range 0 to 2000
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Valid only in data stream mode
171	API function	int IMC100_Get_PalletPara(int *rowNum, int *colNum, int *layerNum, double *layerHeight, int comId=0)
	Description	Queries the pallet parameters.
	Parameter	• rowNum: Number of rows, range 0-1000 • colNum: Number of columns, range 0-1000 • layerNum: Number of layers, range 0-1000 • layerHeight: Layer height, in mm

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
172	API function	int IMC100_Set_PalletPara(int rowNum, int colNum, int layerNum, double layerHeight, int comId=0)
	Description	Sets the pallet parameters.
	Parameter	• rowNum: Number of rows, range 0-1000 • colNum: Number of columns, range 0-1000 • layerNum: Number of layers, range 0-1000 • layerHeight: Layer height, in mm
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
173	API function	int IMC100_Clear_PalletPara(int comId=0)
	Description	Clear the pallet parameters
	Parameter	Clear the pallet parameters
	Response instruction explanation	A return value of 0 indicates a successful clearing, while any value less than 0 indicates failure.
	Remarks	-

No.	Description	
174	API function	int IMC100_Get_Pallet_RobP(ROB_POS *pos1, ROB_POS *pos2, ROB_POS *pos3, int rowIndex, int colIndex, int layIndex, ROB_POS *posDst, int comId=0)
	Description	Queries the points of the corresponding pallet (Number of points defining the pallet boundary: 3 points)
	Parameter	• pos1-3: 3 points defining the pallet • rowIndex: Row index of the point to be queried • colIndex: Column index of the point to be queried • layIndex: Layer index of the point to be queried • posDst: Queried point position result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
175	API function	int IMC100_Get_Pallet4_RobP(ROB_POS *pos1, ROB_POS *pos2, ROB_POS *pos3, ROB_POS *pos4, int rowIndex, int colIndex, int layIndex, ROB_POS *posDst, int comId=0)
	Description	Queries the points of the corresponding pallet (Number of points defining the pallet boundary: 4 points)
	Parameter	• pos1-4: 4 points defining the pallet • rowIndex: Row index of the point to be queried • colIndex: Column index of the point to be queried • layIndex: Layer index of the point to be queried • posDst: Queried point position result

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
176	API function	int IMC100_SavePara(int comId=0)
	Description	Saves system parameters with power-off retention.
	Parameter	-
	Response instruction explanation	A return value of 0 indicates a successful operation, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
177	API function	int IMC100_RecoverPara(int comId=0)
	Description	Restores system parameters (restoring to the parameters after the last save operation).
	Parameter	-
	Response instruction explanation	A return value of 0 indicates a successful operation, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
178	API function	int IMC100_Get_RobJP(int pNum, ROB_JPOS *pos, int comId=0)

No.	Description	
	Description	Queries the position parameters corresponding to the global joint position variables.
	Parameter	• pNum: Index of the global position variable, range 0 to 9999 • pos: Joint position parameter structure, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
179	API function	int IMC100_Set_RobJP(int pNum, ROB_JPOS *pos, int comId=0)
	Description	Sets the position parameters corresponding to the global joint position variables.
	Parameter	• pNum: Index of the global position variable, range 0 to 9999 • pos: Joint position parameter structure
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Edit, Admin and Manufacturer modes (switching modes by calling the function IMC100_UserLogin, same applies below)
180	API function	int IMC100_Set_RobJPHere(int pNum, int comId=0)
	Description	Sets the global joint position parameters using the current point parameters.
	Parameter	pNum: Index of the global joint position variable

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Edit, Admin, and Manufacturer modes.
181	API function	int IMC100_Get_RobP(int pNum, ROB_POS *pos, int comId=0)
	Description	Queries the position parameters corresponding to the global position variables.
	Parameter	• pNum: Index of the global position variable, range 0 to 9999 • pos: Position parameter structure, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
182	API function	int IMC100_Set_RobP(int pNum, ROB_POS *pos, int comId=0)
	Description	Sets the position parameters corresponding to the global position variables.
	Parameter	• pNum: Index of the global position variable, range 0 to 9999 • pos: Position parameter structure
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Edit, Admin, and Manufacturer modes.

No.	Description	
183	API function	int IMC100_Get_RobPFromFile(char *pFileName, int pNum, ROB_POS *pos, int comId=0)
	Description	Queries the position parameters corresponding to the global position variable from the file.
	Parameter	• pFileName: Global position variable file name (case-insensitive for API) • pNum: Index of the global position variable, range 0 to 9999 • pos: Position parameter structure
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
184	API function	int IMC100_Set_RobPToFile(char *pFileName, int pNum, ROB_POS *pos, int comId=0)
	Description	Sets the position parameters corresponding to the global position variable into the specified file.
	Parameter	• pFileName: Global position variable file name (case-insensitive for API) • pNum: Index of the global position variable, range 0 to 9999 • pos: Position parameter structure
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Edit, Admin, and Manufacturer modes.
185	API function	int IMC100_Get_CurRobPFileName(char *pFileName, int comId=0)

No.	Description	
	Description	Queries the currently loaded global position variable file.
	Parameter	pFileName: Global position variable file name (case-insensitive for API)
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
186	API function	int IMC100_Set_RobPHere(int pNum, int comId=0)
	Description	Sets the global position parameters using the current point parameters.
	Parameter	pNum: Index of the global position variable
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
187	API function	int IMC100_Set_RobPHereToFile(char *pFileName,int pNum, int comId=0)
	Description	Sets the position data of the specified P point file using the current point parameters.
	Parameter	• pFileName: Global position variable file name (case-insensitive for API) • pNum: Index of the global position variable, range 0 to 9999

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Edit, Admin, and Manufacturer modes.
188	API function	int IMC100_Get_RobP_Label(int pNum, char *plabelName, int comId)
	Description	Queries the global position label information.
	Parameter	• pNum: Index of the global position point • plabelName: Label information
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
189	API function	int IMC100_Get_RobJP_Label(int pNum, char *plabelName, int comId)
	Description	Queries the global joint point label information.
	Parameter	• pNum: Index of the global Joint point • plabelName: Label information
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-

No.	Description	
190	API function	int IMC100_Get_RobLP_Label(int taskId, char *pProName, int pNum, char *plabelName, int comId)
	Description	Queries the local position point label information.
	Parameter	• taskId: Task channel • pProName: Name of the program where the local point is located (case-insensitive, .pro suffix required) • pNum: Index of the local position point • plabelName: Label information
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
191	API function	int IMC100_Get_SpeedLimitSwitch(int *sLimitSwitch, int comId)
	Description	Gets the speed limit switch status.
	Parameter	sLimitSwitch: Speed limit switch status • 0: Speed limit enabled • 1: Speed limit disabled
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-
192	API function	int IMC100_Set_SpeedLimitSwitch(int sLimitSwitch, int comId)

No.	Description	
	Description	Sets whether the robot is speed-limited.
	Parameter	sLimitSwitch: Speed limit switch status • 0: Speed limit enabled • 1: Speed limit disabled
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Edit, Admin, and Manufacturer modes.
193	API function	int IMC100_Get_PRVar(int prNum, POSE *pos, int comId=0)
	Description	Queries the parameters corresponding to the global translation variables.
	Parameter	• prNum: Index of the global translation variable, range 0 to 255 • pos: Position parameter structure, representing the query result, where the arm parameters are invalid
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
194	API function	int IMC100_Set_PRVar(int prNum, POSE pos, int comId=0)
	Description	Sets the parameters corresponding to the global translation variables.

No.	Description	
	Parameter	• prNum: Index of the global translation variable, range 0 to 255 • pos: Position parameter structure, where the arm parameters are invalid
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
195	API function	int IMC100_Get_B(int num, int *val, int comId=0)
	Description	Queries the value of global B variable.
	Parameter	• num: B variable number, range 0 to 255 • val: B variable value, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
196	API function	int IMC100_Set_B(int num, int val, int comId=0)
	Description	Sets the value of global B variable.
	Parameter	• num: B variable number, range 0 to 255 • val: B variable value, range 0 to 255
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.

No.	Description	
	Remarks	-
197	API function	int IMC100_Get_R(int num, int *val, int comId=0)
	Description	Queries the value of global R variable.
	Parameter	• num: R variable number, range 0 to 255 • val: R variable value, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
198	API function	int IMC100_Set_R(int num, int val, int comId=0)
	Description	Sets the value of global R variable.
	Parameter	• num: R variable number, range 0 to 255 • val: R variable value, range -2147483647 to 2147483647
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Edit, Admin, and Manufacturer modes.
199	API function	int IMC100_Get_D(int num, double *val, int comId=0)
	Description	Queries the value of global D variable.

No.	Description	
	Parameter	• num: D variable number, range 0 to 255 • val: D variable value, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
200	API function	int IMC100_Set_D(int num, double val, int comId=0)
	Description	Sets the value of global D variable.
	Parameter	• num: D variable number, range 0 to 255 • val: D variable value, range -9999999.999 to 9999999.999
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Edit, Admin, and Manufacturer modes.
201	API function	int IMC100_Get_ModbusCoil(int address, int sum, int *val, int comId=0)
	Description	Queries the coil values in the Modbus variable area.
	Parameter	• address: Modbus coil address, range 0 to 8191 • sum: Total number of coils read, range 1 to 8 • val: Coil value, representing the query result

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
202	API function	int IMC100_Set_ModbusCoil(int address, int sum, int val, int comId=0)
	Description	Sets the coil values in the Modbus variable area.
	Parameter	• address: Modbus coil address, range 2048 to 4095, 6144 to 8191 • sum: Total number of coils read, range 1 to 8 • val: Coil value
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Edit, Admin, and Manufacturer modes.
203	API function	int IMC100_Get_ModbusRegUshort(int address, int sum, unsigned short val[], int comId=0)
	Description	Queries the register values in the Modbus variable area, with a data type of unsigned short.
	Parameter	• address: Modbus register address, range 0 to 65535 • sum: Total number of registers read, range 1 to 8 • val: Value, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.

No.	Description	
	Remarks	-
204	API function	int IMC100_Set_ModbusRegUshort(int address, int sum, unsigned short val[], int comId=0)
	Description	Sets the register values in the Modbus variable area, with a data type of unsigned short.
	Parameter	• address: Modbus register address, range 16384 to 32767, 49152 to 65535 • sum: Total number of registers read, range 1 to 8 • val: Value, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Edit, Admin, and Manufacturer modes.
205	API function	int IMC100_Get_ModbusRegFloat(int address, int sum, float val[], int comId=0)
	Description	Queries the register values in the Modbus variable area, with a data type of float.
	Parameter	• address: Modbus register address, range 0 to 65535 • sum: Total number of registers read, range 1 to 8 • val: Value, representing the query result (one float data occupies 2 registers)
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-

No.	Description	
206	API function	int IMC100_Set_ModbusRegFloat(int address, int sum, float val[], int comId=0)
	Description	Sets the register values in the Modbus variable area, with a data type of float.
	Parameter	• address: Modbus register address, range 16384 to 32767, 49152 to 65535 • sum: Total number of registers read, range 1 to 8 • val: Variable value
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Edit, Admin, and Manufacturer modes.
207	API function	int IMC100_Get_PlcVarByte(int num, unsigned char *val, int comId=0)
	Description	Queries the PLC Byte-type variable value.
	Parameter	• num: Byte variable number, range 0 to 255 • val: Variable value, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
208	API function	int IMC100_Get_PlcVarInt(int num, short *val, int comId=0)
	Description	Queries the PLC Int-type variable value.

No.	Description	
	Parameter	• num: Int variable number, range 0 to 255 • val: Variable value, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
209	API function	int IMC100_Get_PlcVarDInt(int num, int *val, int comId=0)
	Description	Queries the PLC DInt-type variable value.
	Parameter	• num: DInt variable number, range 0 to 255 • val: Variable value, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
210	API function	int IMC100_Get_PlcVarLReal(int num, double *val, int comId=0)
	Description	Queries the PLC LReal-type variable value.
	Parameter	• num: LReal variable number, range 0 to 255 • val: Variable value, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.

No.	Description	
	Remarks	-
211	API function	int IMC100_Get_UserAlarm(int num, char alarm[40], int comId=0)
	Description	Queries the content of the user alarm.
	Parameter	• num: User alarm number, range 0 to 15 • alarm: Alarm content description, representing the query result, with a length limit of no exceeding 40 bytes
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
212	API function	int IMC100_Set_UserAlarm(int num, char alarm[40], int comId=0)
	Description	Sets the content of the user alarm.
	Parameter	• num: User alarm number, range 0 to 15 • alarm: Alarm content description, with a length limit of no exceeding 40 bytes
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
213	API function	int IMC100_Get_Print(char val[120], int comId=0)

No.	Description	
	Description	Queries the controller print information, including the content printed by the program print instruction and system error messages.
	Parameter	val: Print content, representing the query result, with a length limit of no exceeding 120 bytes
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
214	API function	int IMC100_CurCtrlDev(int *dev, int comId=0)
	Description	Queries the device that currently has the control authority.
	Parameter	dev: Number of the device that currently has the control authority • 0: InoTeachPad/InoRobotLab • 1: InoRobShop • 2: Remote Ethernet device • 3: Remote I/O • 4: Remote Modbus
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
215	API function	int IMC100_CurPermit(int *owner, unsigned int *ipAddr, unsigned short *ipPort, int comId=0)
	Description	Queries information of the Ethernet device that currently has the control permission.

No.	Description	
	Parameter	• owner: Identity of the Ethernet device that has the permission, representing the query result ◦ 0: No Ethernet device permitted ◦ 1: Current device permitted ◦ 2: Other Ethernet devices permitted • ipAddr: Device network IP address, representing the query result, meaningless when the first return value is 0 • ipPort: Device network port number, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
216	API function	int IMC100_AcqPermit(int cmd=0, int comId=0)
	Description	Requests control permission for the current API network client device.
	Parameter	cmd: Request command • 0: Normal request • 1: Force acquisition • Default 0
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-
217	API function	int IMC100_RemovePermit(int comId=0)
	Description	Releases control permission from the current API network client device.

No.	Description	
	Parameter	-
	Response instruction explanation	A return value of 0 indicates a successful release, while any value less than 0 indicates failure.
	Remarks	-
218	API function	int IMC100_CurUserType(int *type, int comId=0)
	Description	Queries the current user mode.
	Parameter	type: User mode, representing the query result • 0: User mode • 1: Edit mode • 2: Admin • 3: Manufacturer
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-
219	API function	int IMC100_UserLogin(int type, char password[8], int comId=0)
	Description	Logs the current API network client device into the corresponding user mode.

No.	Description	
	Parameter	• type: User mode ◦ 0: User mode ◦ 1: Edit mode ◦ 2: Admin ◦ 3: Manufacturer • password: Password for login (the default password for User mode and Edit Mode is "000000")
	Response instruction explanation	A return value of 0 indicates a successful login, while any value less than 0 indicates failure.
	Remarks	-
220	API function	int IMC100_UserLogout(int comId=0)
	Description	Logs the current API network client device out of the current login mode and reverts to the default user mode.
	Parameter	-
	Response instruction explanation	A return value of 0 indicates a successful logout, while any value less than 0 indicates failure.
	Remarks	-
221	API function	int IMC100_Set_SysTime(char time[16], int comId=0)
	Description	Sets the current system clock.
	Parameter	time: Time string (including year, month, day, hour, minute, and second), with a length of 14

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
222	API function	int IMC100_LatchEnable(int cmd, int comId=0)
	Description	Controls the enabling of latch function.
	Parameter	cmd: Control command • 1: ON • 0: OFF
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
223	API function	int IMC100_Get_LatchSts(int *sts, int comId=0)
	Description	Queries the latch function status
	Parameter	sts: Latch function status • 1: ON • 0: OFF
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-

No.	Description	
224	API function	int IMC100_Get_LatchSum(int *sum, int comId=0)
	Description	Queries the total number of latched position points.
	Parameter	sum: Query result
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	Available for Admin and Manufacturer modes.
225	API function	int IMC100_Get_LatchRobP(int index, int *sts, ROB_POS *pos, int comId=0)
	Description	Reads the position parameters of the corresponding latching point (read in sequence, each position can only be read once)
	Parameter	• index: Reserved parameter • sts: Return status, indicating whether there is latched data ◦ 0: No ◦ 1: Yes • pos: Return latched value, indicating the current tool value relative to the current workobject
	Response instruction explanation	A return value of 0 indicates a successful reading, while any value less than 0 indicates failure.
	Remarks	-
226	API function	int IMC100_Clr_LatchPos(int comId=0)
	Description	Clears the latch position.

No.	Description	
	Parameter	-
	Response instruction explanation	A return value of 0 indicates the command is completed, while any value less than 0 indicates failure.
	Remarks	-
227	API function	int IMC100_Set_CollModeAndAction(int checkflag, int action, int comId=0)
	Description	Sets the collision detection switch in data stream mode and the actions triggered after a collision alarm.
	Parameter	• checkflag: Collision detection switch in data stream mode: ◦ 0: Disable collision detection ◦ 1: Enable collision detection • action: Action triggered after a collision alarm: ◦ 0: Do not define the trigger action ◦ 3: Planned stop after alarm ◦ 4: Immediate stop after alarm ◦ 7: Alarm reset When "0 - Do not define triggered action" is selected, the actual triggered action is determined by the following setting: • API: IMC100_Set_TeachModeCollAction/ IMC100_Set_PlaybackModeCollAction • InoTeachPad: "Settings" > "Motion parameters" > "Collision detection" • InRobotLab: "Controller Parameter configuration" > "Motion parameters" > "Collision detection"

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Valid only in data stream mode
228	API function	int IMC100_Get_CollModeAndAction(int *checkflag, int *action, int comId=0)
	Description	Reads the collision detection switch and the actions triggered after a collision alarm, for data stream mode or project instruction.
	Parameter	• checkflag: Collision detection switch • action: Action triggered after a collision alarm
	Response instruction explanation	A return value of 0 indicates the command is completed, while any value less than 0 indicates failure.
	Remarks	-
229	API function	int IMC100_Set_AxisCollMode(int axisNo, int checkflag, int comId=0)
	Description	Sets the collision detection switch for a single axis in data stream mode.
	Parameter	• axisNo: Axis number, related to the number of axes, range 1 to 6 • checkflag: Collision detection switch: ◦ 0: Disable collision detection for a specific axis ◦ 1: Enable collision detection for a specific axis
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.

No.	Description	
	Remarks	Valid only in data stream mode
230	API function	int IMC100_Get_AxisCollMode(int axisNo, int *checkflag, int comId=0)
	Description	Reads the single-axis collision detection switch for data stream mode or project instruction.
	Parameter	• axisNo: Axis number, related to the number of axes, range 1 to 6 • checkflag: Collision detection switch
	Response instruction explanation	A return value of 0 indicates a successful reading, while any value less than 0 indicates failure.
	Remarks	-
231	API function	int IMC100_Set_AxisCollLevel(int axisNo, double level, int comId=0)
	Description	Sets the collision detection sensitivity for a single axis in data stream mode.
	Parameter	• axisNo: Axis number, related to the number of axes, range 1 to 6 • level: Axis collision detection sensitivity, range 25 to 200
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Valid only in data stream mode
232	API function	int IMC100_Get_AxisCollLevel(int axisNo, double *level, int comId=0)
	Description	Reads the single-axis collision detection sensitivity for the data stream mode or project instruction.

No.	Description	
	Parameter	• axisNo: Axis number, related to the number of axes, range 1 to 6 • level: Axis collision detection sensitivity
	Response instruction explanation	A return value of 0 indicates a successful reading, while any value less than 0 indicates failure.
	Remarks	-
233	API function	int IMC100_Set_TeachModeCollAction(int action, int comId=0)
	Description	Sets the action triggered after a collision alarm in manual mode.
	Parameter	action: Action triggered after a collision alarm • 3: Planned stop after alarm • 4: Immediate stop after alarm • 7: Alarm reset
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
234	API function	int IMC100_Get_TeachModeCollAction(int *action, int comId=0)
	Description	Reads the action triggered after a collision alarm in manual mode.
	Parameter	action: Action triggered after a collision alarm

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful reading, while any value less than 0 indicates failure.
	Remarks	-
235	API function	int IMC100_Set_TeachModeAxisCollLevel(int axisNo, double level, int comId=0)
	Description	Sets the collision detection sensitivity in manual mode.
	Parameter	• axisNo: Axis number, related to the number of axes, range 1 to 6 • level: Axis collision detection sensitivity, range 25 to 200
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
236	API function	int IMC100_Get_TeachModeAxisCollLevel(int axisNo, double *level, int comId=0)
	Description	Reads the collision detection sensitivity in manual mode.
	Parameter	• axisNo: Axis number, related to the number of axes, range 1 to 6 • level: Axis collision detection sensitivity
	Response instruction explanation	A return value of 0 indicates a successful reading, while any value less than 0 indicates failure.
	Remarks	-

No.	Description	
237	API function	int IMC100_Set_PlayBackModeCollAction(int action, int comId=0)
	Description	Sets the action triggered after a collision alarm in auto mode.
	Parameter	action: Action triggered after a collision alarm • 3: Planned stop after alarm • 4: Immediate stop after alarm • 7: Alarm reset
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
238	API function	int IMC100_Get_PlayBackModeCollAction(int *action, int comId=0)
	Description	Reads the action triggered after a collision alarm in auto mode.
	Parameter	action: Action triggered after a collision alarm
	Response instruction explanation	A return value of 0 indicates a successful reading, while any value less than 0 indicates failure.
	Remarks	-
239	API function	int IMC100_Set_PlayBackModeAxisCollLevel(int axisNo, double level, int comId=0)
	Description	Sets the collision detection sensitivity in auto mode.
	Parameter	• axisNo: Axis number, related to the number of axes, range 1 to 6

No.	Description	
		• level: Axis collision detection sensitivity, range 25 to 200
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
240	API function	int IMC100_Get_PlayBackModeAxisCollLevel(int axisNo, double *level, int comId=0)
	Description	Reads the collision detection sensitivity in auto mode.
	Parameter	• axisNo: Axis number, related to the number of axes, range 1 to 6 • level: Axis collision detection sensitivity
	Response instruction explanation	A return value of 0 indicates a successful reading, while any value less than 0 indicates failure.
	Remarks	-
241	API function	int IMC100_Get_RobotAxisNum(int *axisNum, int comId=0)
	Description	Queries the number of robot manipulator axes.
	Parameter	axisNum: Number of robot manipulator axes
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.

No.	Description	
	Remarks	-
242	API function	int IMC100_Set_BindTcpSpeedValue (int num, double minValue, double maxValue, double minSpeed, double maxSpeed, int comId = 0)
	Description	Binds the linear speed analog output to DA[num].
	Parameter	• num: Index of the bound DA, range (0 to the number of active DAs) • minValue: Minimum output analog value, range (output range of selected DA configuration) • maxValue: Maximum output analog value, range (output range of selected DA configuration) • minSpeed: Minimum speed, range (0 to maxSpeed) • maxSpeed: Maximum speed, range (0 to 100000)
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
243	API function	int IMC100_Get_BindTcpSpeedValue(int num, int *Status , double *minValue, double *maxValue, double *minSpeed, double *maxSpeed, int comId = 0)
	Description	Gets the setting data of the input DA index.

No.	Description	
	Parameter	• num: Index of the bound DA, range (0 to the number of active DAs) • Status: Binding status of DA, range 0 or 1 • minValue: Minimum output analog value, range (output range of selected DA configuration) • maxValue: Maximum output analog value, range (output range of selected DA configuration) • minSpeed: Minimum speed, range (0 to maxSpeed) • maxSpeed: Maximum speed, range (0 to 100000)
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-
244	API function	int IMC100_Close_BindTcpSpeed(int num, int comId = 0)
	Description	Unbinds the linear speed analog value from DA[num].
	Parameter	num: Range of num for unbound DA (0 to number of activated DAs)
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
245	API function	int IMC100_Get_TcpSpeedDAOutAndVel(int num, int *Status, double *DAout, double *Velocity, int comId)
	Description	Gets the data of the bound DA output and the linear speed value.

No.	Description	
	Parameter	• num: Index of the bound DA, range (0 to the number of active DAs) • Status: Binding status of DA, output range 0 or 1 • DAout: Real-time output value of DA • Velocity: Real-time output value of linear speed
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-
246	API function	int IMC100_Set_TraceRecoverMode(int TraceMode, int comId = 0)
	Description	Sets the trajectory recovery threshold setting mode.
	Parameter	TraceMode: Trajectory recovery threshold setting mode • 0: Joint mode • 1: Position control mode
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
247	API function	int IMC100_Get_TraceRecoverMode(int *TraceMode , int comId = 0)
	Description	Gets the trajectory recovery threshold setting mode.
	Parameter	TraceMode: Trajectory recovery threshold setting mode • 0: Joint mode • 1: Position control mode

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-
248	API function	int IMC100_Set_VarTraceRecoverParams(int Mode, double TCPDis, double TCPRot, double ExternalDis ,double ExternalRot, int comId = 0)
	Description	Sets the trajectory recovery threshold parameters in auto or manual mode.
	Parameter	• Mode: Manual or auto trajectory recovery ◦ 1: Manual ◦ 2: Auto • TCPDis: TCP distance, range 0 to 20000 • TCPRot: TCP rotation, range 0 to 360 • ExternalDis: External axis distance, range 0 to 20000 • ExternalRot: External axis rotation, range 0 to 360
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
249	API function	int IMC100_Get_VarTraceRecoverParams(int Mode, double *TCPDistance, double *TCPRotation, double *ExternalDistance, double *ExternalRotatioon ,int comId = 0)
	Description	Gets the trajectory recovery threshold parameters in auto or manual mode.

No.	Description	
	Parameter	• Mode: Manual or auto trajectory recovery ◦ 1: Manual ◦ 2: Auto • TCPDistance: TCP distance, range 0 to 20000 • TCPRotation: TCP rotation, range 0 to 360 • ExternalDistance: External axis distance, range 0 to 20000 • ExternalRotation: External axis rotation, range 0 to 360
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-
250	API function	int IMC100_Set_InterferZoneActStat(int iNum, int iStatus, int comId);
	Description	Sets the interference zone activation status.
	Parameter	• Index: Interference zone number, range 0 to 15 • iStatus: Interference zone status
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
251	API function	int IMC100_Get_InterferZoneActStat(int iStatus[16], int comId);
	Description	Gets the interference zone activation status.
	Parameter	iStatus: Interference zone status

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-
252	API function	int IMC100_Set_InterferZonePara(int iNum, S_INTERFER_ZONE* pstPara, int comId);
	Description	Sets the interference zone parameters based on the index.
	Parameter	• Index: Interference zone number, range 0 to 15 • pstPara: Interference zone parameter
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
253	API function	int IMC100_Get_InterferZonePara(int iNum, S_INTERFER_ZONE* pstPara, int comId);
	Description	Gets the interference zone parameters based on the index.
	Parameter	• Index: Interference zone number, range 0 to 15 • pstPara: Interference zone parameter
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-
254	API function	int IMC100_Set_InterferToolActNum(int iNum, int comId);

No.	Description	
	Description	Activates the monitoring object based on the index.
	Parameter	iNum: Monitoring object number, range 0 to 15
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
255	API function	int IMC100_Get_InterferToolActNum(int* iNum, int comId);
	Description	Gets the active monitoring object based on the index.
	Parameter	iNum: Monitoring object number, range 0 to 15
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-
256	API function	int IMC100_Set_InterferToolPara(int iNum, S_INTERFER_TOOL* pstPara, int comId);
	Description	Sets the monitoring object based on the index.
	Parameter	• iNum: Monitoring object number, range 0 to 15 • pstPara: Monitoring object parameter
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.

No.	Description	
	Remarks	-
257	API function	int IMC100_Get_InterferToolPara(int iNum, S_INTERFER_TOOL* pstPara, int comId);
	Description	Gets the monitoring object parameters based on the index.
	Parameter	• iNum: Monitoring object number, range 0 to 15 • pstPara: Monitoring object parameter
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-
258	API function	int IMC100_Set_MemRobP(int pNum, ROB_POS *pos, int comId=0)
	Description	Updates the position parameters corresponding to the global position variables stored in memory.
	Parameter	• pNum: Index of the global position variable, range 0 to 9999 • pos: Position parameter structure
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
259	API function	int IMC100_Set_MemRobJP(int pNum, ROB_JPOS *pos, int comId=0)
	Description	Updates the position parameters corresponding to the global joint variables stored in memory.

No.	Description	
	Parameter	• pNum: Index of the global position variable, range 0 to 9999 • pos: Joint position parameter structure
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
260	API function	int IMC100_Set_CollDist(int dist, int comId=0);
	Description	Sets the move back distance for collision detection in data stream mode.
	Parameter	dist: The move back distance triggered after a collision alarm, range 5 to 100
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
261	API function	int IMC100_Get_CollDist(int *dist, int comId=0)
	Description	Gets the move back distance in data stream mode.
	Parameter	dist: The move back distance triggered after a collision alarm, range 5 to 100
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-

No.	Description	
262	API function	int IMC100_Set_TeachModeCollDist (int dist, int comId=0);
	Description	Sets the move back distance for collision detection in teaching mode.
	Parameter	dist: The move back distance triggered after a collision alarm, range 5 to 100
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
263	API function	int IMC100_Get_TeachModeCollDist(int *dist, int comId=0);
	Description	Gets the move back distance in teaching mode.
	Parameter	dist: The move back distance triggered after a collision alarm, range 5 to 100
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-
264	API function	int IMC100_Set_PlayBackModeCollDist(int dist, int comId=0);
	Description	Sets the move back distance for collision detection in play mode.
	Parameter	dist: The move back distance triggered after a collision alarm, range 5 to 100

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
265	API function	int IMC100_Get_PlayBackModeCollDist(int *dist, int comId=0);
	Description	Gets the move back distance in play mode.
	Parameter	dist: The move back distance triggered after a collision alarm, range 5 to 100
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-
266	API function	int IMC100_Pulse(int donum, double dotime, int highflag, int comId);
	Description	Triggers the output I/O pulse.
	Parameter	• donum: Output I/O number, range 0 to 13823 • dotime: Pulse duration, range 0.1s to 2000s • highflag: High pulse output flag ◦ 0: Output inverted pulse ◦ 1: Output high pulse
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-

No.	Description	
267	API function	int IMC100_Set_Acc(double accpercent, double decpercent, int comId);
	Description	Sets the global acceleration.
	Parameter	• Accpercent: Acceleration value, range 1 to 120 • Decpercent: Deceleration value, range 1 to 120
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
268	API function	int IMC100_Set_Acc(double *accpercent, double *decpercent, int comId);
	Description	Gets the global acceleration.
	Parameter	• Accpercent: Acceleration value, range 1 to 120 • Decpercent: Deceleration value, range 1 to 120
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-
269	API function	int IMC100_Set_GripLoad(int griploadno, int comId);
	Description	Sets the grip load number.
	Parameter	griploadno: Grip load number, range 0 to 15

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
270	API function	int IMC100_Get_GripLoad(int *griploadno, int comId);
	Description	Gets the grip load number.
	Parameter	griploadno: Grip load number, range 0 to 15
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Parameter	-
271	API function	int IMC100_Set_DiagnosisItem (char ItemName, int Flag)
	Description	Sets the save parameters for system diagnosis.
	Parameter	• Name: ALL: All options ◦ 0: All unchecked ◦ 1: All checked • System: System ◦ 0: Unchecked ◦ 1: Checked • Logic: Logic ◦ 0: Unchecked ◦ 1: Checked • Trajectory: Trajectory ◦ 0: Unchecked ◦ 1: Checked

No.	Description	
	Response instruction explanation	• 0: Success • -2: Parameter count mismatch • -3: Invalid parameter • -29: Internal error
	Remarks	Startup parameters are checked by default, i.e., ALL = 1. -1 is returned when setting fails. Example: Ret = IMC100_Set_DiagnosisItem (ALL, 1); Ret = IMC100_Set_DiagnosisItem (System, 1);
272	API function	int IMC100_Get_DiagnosisItem (char ItemName, int *Flag)
	Description	Reads the save parameters for system diagnosis.
	Parameter	• Name: Name ◦ ALL: All options ◦ 0: All unchecked ◦ 1: All checked ◦ 2: Partially checked • System: System ◦ 0: Unchecked ◦ 1: Checked • Logic: Logic ◦ 0: Unchecked ◦ 1: Checked • Trajectory: Trajectory ◦ 0: Unchecked ◦ 1: Checked

No.	Description	
	Response instruction explanation	• Ret: ◦ 0: Success ◦ -2: Parameter count mismatch ◦ -3: Invalid parameter ◦ -29: Internal error • Flag: ◦ 0: Unchecked ◦ 1: Checked ◦ 2: Partially checked
	Remarks	Startup parameters are checked by default, i.e., ALL = 1. -1 is returned when the trigger fails during the system diagnosis. Example: Ret = IMC100_Get_DiagnosisItem (ALL);
273	API function	Int IMC100_Set_DiagnosisSave()
	Description	Triggers the save operation for system diagnosis.
	Parameter	-
	Response instruction explanation	• 0: Success • -2: Failure
	Remarks	During system diagnosis, if a success value of 0 is returned, a prompt message (same as the instruction) will be displayed. The trigger can be executed under read permission. Example: Ret = IMC100_Set_DiagnosisSave();
274	API function	int IMC100_Get_DiagnosisSave (int *flag)

No.	Description	
	Description	Reads the save progress value for system diagnosis.
	Parameter	-
	Response instruction explanation	• Ret: ◦ 0: Success ◦ -2: Parameter count mismatch • Flag: 0 to 100: Save process value
	Remarks	Example: Ret = IMC100_Get_DiagnosisSave ();
275	API function	int IMC100_Set_AbortAccRamp (int level, int comId)
	Description	Sets the acceleration jerk level for "Category 1 stop".
	Parameter	Level: Acceleration jerk level, range 100 to 500
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Example: Ret = IMC100_Set_AbortAccRamp (200, 0);
276	API function	int IMC100_Set_PauseAccRamp (int level, int comId)
	Description	Sets the acceleration jerk level for "Category 2 stop".
	Parameter	Level: Acceleration jerk level, range 100 to 500

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Example: Ret = IMC100_Set_ PauseAccRamp (200, 0);
277	API function	int IMC100_Set_PauseAbortMinAcc (int OnOff, int comId)
	Description	Sets whether to disable the minimum acceleration limit for "Category 1 stop" and "Category 2 stop".
	Parameter	ON/OFF: ON/OFF, range 0 and 1 • 0: OFF • 1: ON
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Example: Ret = IMC100_Set_ PauseAccRamp (0, 0);
278	API function	int IMC100_Get_ConnectState (int *pi32ConnectSts, int comId)
	Description	Gets the connection status of the host controller.
	Parameter	pi32ConnectSts: • 0: Host controller in disconnected status • 1: Host controller in connected status
	Response instruction explanation	Ret: • 0: Success • -2: Parameter count mismatch

No.	Description	
	Remarks	Example: Ret = IMC100_Get_ConnectState (&i32ConnectSts, 0);
279	API function	int IMC100_Get_TCPSpeed (double *pTcpSpeed, int comId=0)
	Description	Gets the current motion speed of the robot end point.
	Parameter	pTcpSpeed: Motion speed of the robot end point at the current moment, representing the query result, in mm/s
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-
280	API function	int IMC100_Get_JointSpeed(char *mecUnit, int axis, double *pJointSpeed, int comId=0)
	Description	Gets the joint speed of a specific axis of a specified mechanical unit.
	Parameter	• mecUnit: Mechanical unit name, 18-character string. The name for the robot manipulator is "Robot". • axis: Axis number in this mechanical unit (range: 1 to maximum axis number of mechanical unit, maximum value is 6) • pJointSpeed: Joint speed at the current moment, representing the query result, in °/s
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-

No.	Description	
281	API function	int IMC100_Get_SectionMovePercent(int *Percent, int comId);
	Description	Gets the segment movement percentage.
	Parameter	Percent: The obtained percentage data of the segment movement, range 0 to 100
	Response instruction explanation	A return value of 0 indicates a successful command issuance, while any value less than 0 indicates failure.
	Remarks	-
282	API function	IMC100_Set_KeyTriggerMode(int KeyTriggerMode, int comId = 0);
	Description	Sets the key trigger mode of the hand-held teach pendant.
	Parameter	KeyTriggerMode: key trigger mode • 0: key switch effective only for teach pendant permission • 1: key switch effective for all control permissions
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
283	API function	int IMC100_Get_KeyTriggerMode(int *pKeyTriggerMode, int comId = 0);
	Description	Gets the key trigger mode of the hand-held teach pendant.
	Parameter	pKeyTriggerMode: The obtained key trigger mode • 0: key switch effective only for teach pendant permission

No.	Description	
		• 1: key switch effective for all control permissions
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-
284	API function	int IMC100_Set_SLVSMode(int Mode, int comId=0)
	Description	Sets the self-learning vibration suppression mode.
	Parameter	Mode: Vibration suppression sensitivity • 3: High • 2: Medium • 1: Low • 0: Disable
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Effective only in data stream mode
285	API function	int IMC100_Get_SLVSMode(int *Mode, int comId=0)
	Description	Gets the self-learning vibration suppression mode.

No.	Description	
	Parameter	Mode: Vibration suppression sensitivity • 3: High • 2: Medium • 1: Low • 0: Disable
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-
286	API function	int IMC100_Set_SLDataClear(int ClearMode, int comId=0);
	Description	Clears the self-learning vibration suppression data.
	Parameter	ClearMode: Clear mode • 1: Clear current data set • 2: Clear all
	Response instruction explanation	A return value of 0 indicates a successful clearing, while any value less than 0 indicates failure.
	Remarks	Effective only in data stream mode
287	API function	int IMC100_MovJ_P_CS_SL(int posNum, VER_DATA *vel, int zone, int tool, int wobj, int SlMode, int ioNum, MOV_IO *movIo, int comId=0)
	Description	Moves to the global position point with specified number using Movj, supporting the tool/ workobject setting and self-learning.

No.	Description	
	Parameter	• posNum: Target global position point number, range 0 to 9999 • vel: Speed parameter structure • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 ◦ -2: NoWait ◦ -1: Fine ◦ 0 to 200: Z[0] to Z[200] ◦ 201: CP ◦ 1000 to 1200: ZR[0] to ZR[200] • tool: Tool number, range 0 to 15 • wobj: Workobject number, range 0 to 15 • SlMode: Self-learning vibration suppression parameter ◦ 0: Disable ◦ 1: Enable ◦ 2: Reset • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details • ioNum: Number of groups of the I/O control structure, range 0 to 3
	Response instruction explanation	A return value of 0 indicates a successful moving, while any value less than 0 indicates failure.
	Remarks	Effective only in data stream mode
288	API function	int IMC100_MovL_P_CS_SL(int posNum, VER_DATA *vel, int zone, int tool, int wobj, int SlMode, int ioNum, MOV_IO *movIo, int comId=0)
	Description	Moves to the global position point with specified number using Movl, supporting the tool/ workobject setting and self-learning.

No.	Description	
	Parameter	• posNum: Target global position point number, range 0 to 9999 • vel: Speed parameter structure • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 ◦ -2: NoWait ◦ -1: Fine ◦ 0-200: Z[0]-Z[200], 201: CP ◦ 1000 to 1200: ZR[0] to ZR[200] • tool: Tool number, range 0 to 15 • wobj: Workobject number, range 0 to 15 • SlMode: Self-learning vibration suppression parameter ◦ 0: Disable ◦ 1: Enable ◦ 2: Reset • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details • ioNum: Number of groups of the I/O control structure, range 0 to 3
	Response instruction explanation	A return value of 0 indicates a successful moving, while any value less than 0 indicates failure.
	Remarks	Effective only in data stream mode
289	API function	int IMC100_MovC_P_CS_SL(int posMidNum, int posDstNum, VER_DATA *vel, int zone, int tool, int wobj, int SlMode, int ioNum, MOV_IO *movIo, int comId=0)
	Description	Moves to the global position point with specified number using MovI, supporting the tool/ workobject setting and self-learning.

No.	Description	
	Parameter	• posMidNum: Global position point number of the arc intermediate point, range 0 to 9999 • posDstNum: Global position point number of the arc endpoint, range 0 to 9999 • vel: Speed parameter structure • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 ◦ -2: NoWait ◦ -1: Fine ◦ 0 to 200: Z[0] to Z[200] ◦ 201: CP ◦ 1000 to 1200: ZR[0] to ZR[200] • tool: Tool number, range 0 to 15 • wobj: Workobject number, range 0 to 15 • SlMode: Self-learning vibration suppression parameter ◦ 0: Disable ◦ 1: Enable ◦ 2: Reset • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details • ioNum: Number of groups of the I/O control structure, range 0 to 3
	Response instruction explanation	A return value of 0 indicates a successful moving, while any value less than 0 indicates failure.
	Remarks	Effective only in data stream mode
290	API function	int IMC100_MovJ_RobPos_CS_SL(ROB_POS *pos,VER_DATA *vel, int zone, int tool, int wobj, int SlMode, int ioNum, MOV_IO *movIo, int comId=0)
	Description	Moves to the specified position point using Movj, and controls the corresponding I/O, supporting the tool/workobject setting and self-learning.

No.	Description	
	Parameter	• pos: Position parameter structure • vel: Speed parameter structure • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 ◦ -2: NoWait ◦ -1: Fine ◦ 0 to 200: Z[0] to Z[200] ◦ 201: CP ◦ 1000 to 1200: ZR[0] to ZR[200] • tool: Tool number, range 0 to 15 • wobj: Workobject number, range 0 to 15 • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details • ioNum: Number of groups of the I/O control structure, range 0 to 3 • SlMode: Self-learning vibration suppression parameter ◦ 0: Disable ◦ 1: Enable ◦ 2: Reset
	Response instruction explanation	A return value of 0 indicates a successful moving, while any value less than 0 indicates failure.
	Remarks	Effective only in data stream mode
291	API function	int IMC100_MovL_RobPos_CS_SL(ROB_POS *pos,VER_DATA *vel, int zone, int tool, int wobj, int SlMode, int ioNum, MOV_IO *movIo, int comId=0)
	Description	Moves to the specified position point using Movl, and controls the corresponding I/O, supporting the tool/workobject setting and self-learning.

No.	Description	
	Parameter	• pos: Position parameter structure • vel: Speed parameter structure • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 ◦ -2: NoWait ◦ -1: Fine ◦ 0 to 200: Z[0] to Z[200] ◦ 201: CP ◦ 1000 to 1200: ZR[0] to ZR[200] • tool: Tool number, range 0 to 15 • wobj: Workobject number, range 0 to 15 • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details • ioNum: Number of groups of the I/O control structure, range 0 to 3 • SlMode: Self-learning vibration suppression parameter ◦ 0: Disable ◦ 1: Enable ◦ 2: Reset
	Response instruction explanation	A return value of 0 indicates a successful moving, while any value less than 0 indicates failure.
	Remarks	Effective only in data stream mode
292	API function	int IMC100_MovC_RobPos_CS_SL(ROB_POS *posMid,ROB_POS *posDst,VER_DATA *vel, int zone, int tool, int wobj, int SlMode, int ioNum, MOV_IO *movIo, int comId=0)
	Description	Moves to the specified position point using Movc, and controls the corresponding I/O, supporting the tool/workobject setting and self-learning.

No.	Description	
	Parameter	• posMid: Position parameter structure for the arc intermediate point • posDst: Position parameter structure for the arc endpoint • vel: Speed parameter structure • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 ◦ -2: NoWait ◦ -1: Fine ◦ 0 to 200: Z[0] to Z[200] ◦ 201: CP ◦ 1000 to 1200: ZR[0] to ZR[200] • tool: Tool number, range 0 to 15 • wobj: Workobject number, range 0 to 15 • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details • ioNum: Number of groups of the I/O control structure, range 0 to 3 • SlMode: Self-learning vibration suppression parameter ◦ 0: Disable ◦ 1: Enable ◦ 2: Reset
	Response instruction explanation	A return value of 0 indicates a successful moving, while any value less than 0 indicates failure.
	Remarks	Effective only in data stream mode
293	API function	int IMC100_JumpL_RobPos_CS_SL(ROB_POS *pos,VER_DATA *vel, int zone, int tool, int wobj, int SlMode, int ioNum, MOV_IO *movIo, int comId=0)
	Description	Moves to the specified position point using JumpL, and controls the corresponding I/O, supporting the tool/workobject setting and self-learning.

No.	Description	
	Parameter	• pos: Position parameter structure • vel: Speed parameter structure • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 ◦ -2: NoWait ◦ -1: Fine ◦ 0 to 200: Z[0] to Z[200] ◦ 201: CP ◦ 1000 to 1200: ZR[0] to ZR[200] • tool: Tool number, range 0 to 15 • wobj: Workobject number, range 0 to 15 • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details • ioNum: Number of groups of the I/O control structure, range 0 to 3 • SlMode: Self-learning vibration suppression parameter ◦ 0: Disable ◦ 1: Enable ◦ 2: Reset
	Response instruction explanation	A return value of 0 indicates a successful moving, while any value less than 0 indicates failure.
	Remarks	Effective only in data stream mode
294	API function	int IMC100_Jump_RobPos_CS_SL(ROB_POS *pos,VER_DATA *vel, int zone, int tool, int wobj, int SlMode, int ioNum, MOV_IO *movIo, int comId=0)
	Description	Moves to the specified position point using Jump, and controls the corresponding I/O, supporting the tool/workobject setting and self-learning.

No.	Description	
	Parameter	• pos: Position parameter structure • vel: Speed parameter structure • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 ◦ -2: NoWait ◦ -1: Fine ◦ 0 to 200: Z[0] to Z[200] ◦ 201: CP ◦ 1000 to 1200: ZR[0] to ZR[200] • tool: Tool number, range 0 to -15 • wobj: Workobject number, range 0 to 15 • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details • ioNum: Number of groups of the I/O control structure, range 0 to 3 • SlMode: Self-learning vibration suppression parameter ◦ 0: Disable ◦ 1: Enable ◦ 2: Reset
	Response instruction explanation	A return value of 0 indicates a successful moving, while any value less than 0 indicates failure.
	Remarks	Effective only in data stream mode
295	API function	int IMC100_JumpL_P_CS_SL(int posNum, VER_DATA *vel, int zone, int tool, int wobj, int SlMode, int ioNum, MOV_IO *movIo, int comId=0)
	Description	Moves to the global position point with specified number using JumpL, and controls the corresponding I/O, supporting the tool/workobject setting and self-learning.

No.	Description	
	Parameter	• posNum: Target global position point number, range 0 to 9999 • vel: Speed parameter structure • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 ◦ -2: NoWait ◦ -1: Fine ◦ 0 to 200: Z[0] to Z[200] ◦ 201: CP ◦ 1000 to 1200: ZR[0] to ZR[200] • tool: Tool number, range 0 to 15 • wobj: Workobject number, range 0 to 15 • SlMode: Self-learning vibration suppression parameter ◦ 0: Disable ◦ 1: Enable ◦ 2: Reset • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details • ioNum: Number of groups of the I/O control structure, range 0 to 3
	Response instruction explanation	A return value of 0 indicates a successful moving, while any value less than 0 indicates failure.
	Remarks	Effective only in data stream mode
296	API function	int IMC100_Jump_P_CS_SL(int posNum, VER_DATA *vel, int zone, int tool, int wobj, int SlMode, int ioNum, MOV_IO *movIo, int comId=0)
	Description	Moves to the global position point with specified number using Jump, and controls the corresponding I/O, supporting the tool/workobject setting and self-learning.

No.	Description	
	Parameter	• posNum: Target global position point number, range 0 to 9999 • vel: Speed parameter structure • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 ◦ -2: NoWait ◦ -1: Fine ◦ 0 to 200: Z[0] to Z[200] ◦ 201: CP ◦ 1000 to 1200: ZR[0] to ZR[200] • tool: Tool number, range 0 to 15 • wobj: Workobject number, range 0 to 15 • SlMode: Self-learning vibration suppression parameter ◦ 0: Disable ◦ 1: Enable ◦ 2: Reset • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details • ioNum: Number of groups of the I/O control structure, range 0 to 3
	Response instruction explanation	A return value of 0 indicates a successful moving, while any value less than 0 indicates failure.
	Remarks	Effective only in data stream mode
297	API function	int IMC100_MovJAbs_RobJPos_CS_SL(ROB_JPOS *pos,VER_DATA *vel, int zone, int tool, int wobj, int SlMode, int ioNum, MOV_IO *movIo, int comId=0)
	Description	Moves to the specified joint point using Movj, and controls the corresponding I/O, supporting the tool/workobject setting and self-learning.

No.	Description	
	Parameter	• pos: Joint position parameter structure • vel: Speed parameter structure • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 ◦ -2: NoWait ◦ -1: Fine ◦ 0 to 200: Z[0] to Z[200] ◦ 201: CP ◦ 1000 to 1200: ZR[0] to ZR[200] • tool: Tool number, range 0 to 15 • wobj: Workobject number, range 0 to 15 • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details • ioNum: Number of groups of the I/O control structure, range 0 to 3 • SlMode: Self-learning vibration suppression parameter ◦ 0: Disable ◦ 1: Enable ◦ 2: Reset
	Response instruction explanation	A return value of 0 indicates a successful moving, while any value less than 0 indicates failure.
	Remarks	Effective only in data stream mode
298	API function	int IMC100_MovJAbs_JP_CS_SL(int posNum,VER_DATA *vel, int zone, int tool, int wobj, int SlMode, int ioNum, MOV_IO *movIo, int comId=0)
	Description	Moves to the joint point with specified number using Movj, and controls the corresponding I/O, supporting the tool/workobject setting and self-learning.

No.	Description	
	Parameter	• posNum: Target global joint point number • vel: Speed parameter structure • zone: Interpolation accuracy, range -2 to +200 and 1000 to 1200, default 0 ◦ -2: NoWait ◦ -1: Fine ◦ 0 to 200: Z[0] to Z[200] ◦ 201: CP ◦ 1000 to 1200: ZR[0] to ZR[200] • tool: Tool number, range 0 to 15 • wobj: Workobject number, range 0 to 15 • movIo: I/O control structure, see 8.1.1 Appendix 1.1 Structures section for details • ioNum: Number of groups of the I/O control structure, range 0 to 3 • SlMode: Self-learning vibration suppression parameter ◦ 0: Disable ◦ 1: Enable ◦ 2: Reset
	Response instruction explanation	A return value of 0 indicates a successful moving, while any value less than 0 indicates failure.
	Remarks	Effective only in data stream mode
299	API function	int IMC100_Set_AccuracyMode (int Mode, int comId=0)
	Description	Sets the trajectory accuracy mode.
	Parameter	Mode: • 0: Default mode (Normal) • 1: Medium accuracy mode (MidAccuracy) • 2: High accuracy mode (HighAccuracy)

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Effective only in data stream mode
300	API function	IMC100_Get_AccuracyMode (int *Mode, int comId=0)
	Description	Gets the trajectory accuracy mode.
	Parameter	Mode: • 0: Default mode (Normal) • 1: Medium accuracy mode (MidAccuracy) • 2: High accuracy mode (HighAccuracy)
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-
301	API function	int IMC100_Set_ErrPosByAxis(int AxisNo, int ErrPos, int comId=0)
	Description	Sets the arrival position error.
	Parameter	• AxisNo: Axis number (range 1 to 6 for 6-axis robot, range 1 to 4 for four-axis robot) • ErrPos: Arrival error, range [500, 200000]
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.

No.	Description	
	Remarks	Effective only in data stream mode
302	API function	int IMC100_Get_ErrPosByAxis(int AxisNo, int *ErrPos, int comId=0)
	Description	Gets the arrival position error.
	Parameter	• AxisNo: Axis number (range 1 to 6 for 6-axis robot, range 1 to 4 for four-axis robot) • ErrPos: Arrival error, range [500, 200000]
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-
303	API function	int IMC100_Set_BandTimeByAxis(int AxisNo, int BandTime, int comId=0)
	Description	Sets the arrival time threshold.
	Parameter	• AxisNo: Axis number (range 1 to 6 for 6-axis robot, range 1 to 4 for four-axis robot) • BandTime: Arrival time threshold, range [0, 100000]
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Effective only in data stream mode
304	API function	int IMC100_Get_BandTimeByAxis(int AxisNo, int *BandTime, int comId=0)
	Description	Gets the arrival time threshold.

No.	Description	
	Parameter	• AxisNo: Axis number (range 1 to 6 for 6-axis robot, range 1 to 4 for four-axis robot) • BandTime: Arrival time threshold, range [0, 100000]
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-
305	API function	int IMC100_Set_SingAreaHandle(int State, int comId=0)
	Description	Sets the singularity crossing function switch.
	Parameter	State: Singularity crossing function enable state • 0: OFF • 1: ON
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Effective only in data stream mode
306	API function	int IMC100_Get_SingAreaHandle(int *State, int comId=0)
	Description	Gets the singularity traversal function switch state.
	Parameter	State: Singularity crossing function enable state • 0: Disable • 1: Enable

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-
307	API function	int IMC100_Get_ModuleSubPartNum(int ModuleType, int *PartNum, int comId = 0)
	Description	Gets the number of component sub-parts based on the component type.
	Parameter	• ModuleType: Component type ◦ 0: Encoder battery ◦ 1: Ball screw grease ◦ 2: Ball screw wear ◦ 3: Synchronous belt ◦ 4: Reducer • PartNum: Number of component sub-parts obtained, range 0 to 6
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	-
308	API function	int IMC100_Get_MaintainInfo(int ModuleType, int ModuleIndex, MT_INFO *ModuleMTInfo, int comId = 0)
	Description	Gets the maintenance information of component sub-parts based on the component type and component number.

No.	Description	
	Parameter	• ModuleType: Component type ◦ 0: Encoder battery ◦ 1: Ball screw grease ◦ 2: Ball screw wear ◦ 3: Synchronous belt ◦ 4: Reducer • ModuleIndex: Component sub-part number, range 1 to 6, indicating the index of a sub-part • ModuleMTInfo: Component maintenance information structure, representing the query result, see 8.1.1 Appendix 1.1 Structures section for details
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.
	Remarks	When the component type is ball screw, ModuleIndex can only be 1 (there can only be one ball screw).
309	API function	int IMC100_Clear_MaintainInfo(int comId = 0)
	Description	Clears all maintenance information.
	Parameter	comId: Connection ID, identifies the corresponding connection
	Response instruction explanation	A return value of 0 indicates a successful clearing, while any value less than 0 indicates failure.
	Remarks	Maintenance operations can only be performed in a non-motion state under Manufacturer mode, Emergency Stop mode, and Manual mode; otherwise, -29 is returned.

No.	Description	
310	API function	int IMC100_Set_MaintainThreshold(int ModuleType, int ModuleIndex, double ThresholdValue, int comId = 0)
	Description	Sets the maintenance threshold for the component sub-parts.
	Parameter	• ModuleType: Component type ◦ 0: Encoder battery ◦ 1: Ball screw grease ◦ 2: Ball screw wear ◦ 3: Synchronous belt ◦ 4: Reducer grease • ModuleIndex: Component sub-part number, range 1 to 6, indicating the index of a sub-part • ThresholdValue: Threshold. The range of the threshold parameter varies depending on the component type. • Synchronous belt: [0,10000] hours • Encoder battery: [0, 50000] hours • Reducer grease: [0, 50000] hours • Ball screw grease: [0, 50000] hours • Ball screw wear: [0, 100]%
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	• Maintenance operations can only be performed in a non-motion state under Manufacturer mode and Manual mode; otherwise, -29 is returned. • When the component type is ball screw, ModuleIndex can only be 1 (there can only be one ball screw). • Before calling this API, you can first use IMC100_Get_ModuleSubPartNum to get the number of component sub-parts, then determine the sub-part number to be set. This number is not the axis number.

No.	Description	
311	API function	int IMC100_Update_MaintainInfo(int ModuleType, int ModuleIndex, int comId = 0)
	Description	Updates the maintenance information of the component sub-part after maintenance is completed.
	Parameter	• ModuleType: Component type ◦ 0: Encoder battery ◦ 1: Ball screw grease ◦ 2: Ball screw wear ◦ 3: Synchronous belt ◦ 4: Reducer grease • ModuleIndex: Component sub-part number, range 1 to 6, indicating the index of a sub-part
	Response instruction explanation	A return value of 0 indicates a successful update, while any value less than 0 indicates failure.
	Remarks	• Maintenance operations can only be performed in a non-motion state under Manufacturer mode, Emergency Stop mode, and Manual mode; otherwise, -29 is returned. • When the component type is ball screw, ModuleIndex can only be 1 (there can only be one ball screw). • Before calling this API, you can first use IMC100_Get_ModuleSubPartNum to get the number of component sub-parts, then determine the sub-part number to be set. This number is not the axis number.
312	API function	int IMC100_Set_RtdeCfgLoad(int cmdId);
	Description	Loads the real-time data exchange configuration.
	Parameter	-

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful loading, while any value less than 0 indicates failure.
	Remarks	-
313	API function	int IMC100_Set_RtdeChannelStatus (int Index,int Status, int cmdId);
	Description	Sets the activation status of the real-time data exchange channel.
	Parameter	• Index: Channel number, range 0 to 15 • Status: Channel activation status; 0/1
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	Only one channel can be activated at a time; when activating a channel, the other channel will be deactivated.
314	API function	int IMC100_Get_OffsetRef_RobP(ROB_POS stBasePos, POSE stPR, int ToolNum, int WobjNum, ROB_JPOS stJPos, ROB_POS *pOutpos, int comId);
	Description	Translation in Cartesian reference frame with reference joint angles
	Parameter	• ROB_POS stBasePos: Base point • POSE stPR: Offset • int ToolNum: Tool number • int WobjNum: Workobject number • ROB_JPOS stJPos: Reference joint angle • ROB_POS *pOutpos: Offset point position

No.	Description	
	Response instruction explanation	• 0: Success • -2: Failure
	Remarks	-
315	API function	int IMC100_Get_OffsetRefnotJPos_RobP(ROB_POS stBasePos, POSE stPR, int ToolNum, int WobjNum, ROB_POS *pOutpos, int comId);
	Description	Translation in Cartesian reference frame without reference joint angles
	Parameter	• ROB_POS stBasePos: Base point • POSE stPR: Offset • int ToolNum: Tool number • int WobjNum: Workobject number • ROB_POS *pOutpos: Offset point position
	Response instruction explanation	• 0: Success • -2: Failure
	Remarks	-
316	API function	int IMC100_Get_OffSetMode(int iStatus, int comId);
	Description	Gets the offset solution selection mode.
	Parameter	• 0: Default offset solution selection mode • 1: Offset solution selection mode for S03.22 to V4R24C3 versions
	Response instruction explanation	A return value of 0 indicates a successful acquisition, while any value less than 0 indicates failure.

No.	Description	
	Remarks	-
317	API function	int IMC100_Set_OffSetMode(int iStatus, int comId);
	Description	Sets the offset solution selection mode.
	Parameter	• 0: Default offset solution selection mode • 1: Offset solution selection mode for S03.22 to V4R24C3 versions
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
318	API function	int IMC100_Set_SupplementaryStrParamComp(float para[14], int comId);
	Description	Sets the compensation parameters.
	Parameter	para[]: Structure compensation parameters, representing the query result
	Response instruction explanation	A return value of 0 indicates a successful setting, while any value less than 0 indicates failure.
	Remarks	-
319	API function	int IMC100_Get_SupplementaryStrParamComp(float para[14], int comId);
	Description	Queries the compensation parameters.
	Parameter	para[]: Structure compensation parameters, representing the query result

No.	Description	
	Response instruction explanation	A return value of 0 indicates a successful query, while any value less than 0 indicates failure.
	Remarks	-

8.2 Appendix 2: API String Table

API string application notes:

- For the error content corresponding to "e*" in the "Response instruction format", see the relevant fault description.
- The request instruction format is: @@instruction$$.
- In the request instruction explanation, "Para*:" refers to the set parameters, with spaces separating the index numbers.



The length of a single parameter string must not exceed 128.

No.	Description	
1	Request instruction format	Call the general communication interface to establish/close communication with the robot controller. The host controller acts as the client. The API port number is fixed at 2222.
	Request instruction explanation	-
	Response instruction format	-

No.	Description	
	Response instruction explanation	-
	Remarks	• The controller supports up to 4 different connections. • Connection number scope: Same process of the host controller  The connection is considered different if any one of the following differs between the host controller and the controller: IP address, or port number.
2	Request instruction format	• EStop ON • EStop OFF
	Request instruction explanation	• Press the emergency stop switch • Release the emergency stop switch
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error, where * is the specific error code, same convention applies below.

No.	Description	
	Remarks	-
3	Request instruction format	• Motor ON • Motor OFF
	Request instruction explanation	• Motor enable • Motor disable
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	After the enable command is issued, the enable status should be read after 300 ms or above.
4	Request instruction format	ResetErr
	Request instruction explanation	Resets the controller fault.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	The processing time for this instruction is approximately 50 ms.
5	Request instruction format	Set_Mode ModeType
	Request instruction explanation	Sets the system operation mode. ModeType: Mode • 1: Manual • 2: Auto
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Controls the Manual/Auto mode switching for the robot project.

No.	Description	
6	Request instruction format	• Prg Start • Prg Stop
	Request instruction explanation	• Starts the project operation • Stops the project operation
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Controls the start/stop operation of the robot project.
7	Request instruction format	BackStartLine
	Request instruction explanation	Motion line returns to the first line of the main program.
	Response instruction format	ok or e*:

No.	Description	
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Controls the main task and dynamic tasks of the robot project to return to the start line.
8	Request instruction format	Set_Vel Para1
	Request instruction example	Set_Vel 100
	Request instruction explanation	Sets the current global running speed. Para1: Global speed, range 1 to 100
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-

No.	Description	
9	Request instruction format	Set_AccRamp Para1 Para2
	Request instruction example	Set_AccRamp 50.0 50.0
	Request instruction explanation	Sets the acceleration jerk for the motion segment in data stream mode. • Para1: Acceleration jerk percentage of the start segment, range 10.0 to 100. • Para2: Acceleration jerk percentage of the end segment, range 10.0 to 100
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
10	Request instruction format	Set_RapidMove Para1 Para2
	Request instruction example	Set_RapidMove 0 0

No.	Description	
	Request instruction explanation	Sets the optimal trajectory planning. • Para1: Motion type ◦ 0: cp motion ◦ 1: ptp motion • Para2: Optimal planning switch ◦ 0: Disable optimal planning ◦ 1: Enable optimal planning
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
11	Request instruction format	• Dsmode ON • Dsmode OFF • Dsmode PAUSE • Dsmode CONTINUE
	Request instruction explanation	• The data stream mode is ON. • The data stream mode is OFF. • The data stream mode is PAUSE. • The data stream mode is CONTINUE.
	Response instruction format	ok or e*:

No.	Description	
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	When the data stream mode is ON, if the robot is disabled, it will automatically switch to data stream pause mode. Robot motion pauses when the data stream mode is OFF or Pause.
12	Request instruction format	Set_DO Para1 Para2
	Request instruction example	Set_DO 0 1
	Request instruction explanation	Sets the DO * status to ON. • Para1: RC controllable DO number • Para2: DO status ◦ 1: ON ◦ 0: OFF
	Response instruction format	ok or e*:
	Response instruction explanation	Setting completes, or returns an error.
	Remarks	-

No.	Description	
13	Request instruction format	Set_Slewmode Para1
	Request instruction example	Set_Slewmode 1
	Request instruction explanation	Sets the rotation optimization mode for the J4 axis of the SCARA robot or the J6 axis of the standard 6-axis robot. Para1: Rotation optimization mode • 0: No optimization, subject to arm parameters in the position variable. • 1: Optimization mode 1 to ensure that the rotation of SCARA robot's J4/6-axis robot's J6 is within the range of -180° to +180° during motion. • 2: Ensure that the SCARA robot's J4/6-axis robot's J6 always moves in the shortest way during motion. The robot calculates and determines whether reaching the target point requires J4/J6 to rotate by 180°. If the angle difference is ≤180°, the target point is fully reached. If >180°, then J4/J6 moves in the opposite direction and ultimately moves to a position where J4/J6 is 360° away from the target point. If the reverse motion of J4/J6 exceeds the limit range of the robot, it will stop the motion, and alarm. • 3: Ensure that the SCARA robot's J4/6-axis robot's J6 always moves in the shortest way during motion, similar to mode 2. The difference is that if the reverse motion of J4/J6 exceeds the limit range of the robot, there will be no alarm. Instead, no reverse motion will be carried out and the original normal motion will be fully adopted.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode, reset after data stream mode is turned off
14	Request instruction format	Set_DOGroup Para1 Para2
	Request instruction example	Set_DOGroup 1 255
	Request instruction explanation	Sets a group of DO statuses, with group number 1 and a quantity of 8. • Para1: RC controllable DO group number, range 0 to 7, subject to actual configuration • Para2: DO group output status, value range 0 to 255, with each bit corresponding to the status of each DO ◦ 1: ON ◦ 0: OFF
	Response instruction format	ok or e*:

No.	Description	
	Response instruction explanation	Setting completes, or returns an error.
	Remarks	-
15	Request instruction format	Set_DA Para1 Para2
	Request instruction example	Set_DA 0 0.000
	Request instruction explanation	Sets the DA output value by index. • Para1: RC controllable DA number • Para2: DA value, maximum range 0 mA to 20 mA for current type, -10 V to +10 V for voltage type, subject to DA channel type
	Response instruction format	ok or e*:
	Response instruction explanation	Setting completes, or returns an error.
	Remarks	-

No.	Description	
16	Request instruction format	• InchMode ON • InchMode OFF
	Request instruction explanation	• Enables jog mode. • Disables jog mode.
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
17	Request instruction format	Set_InchStep Para1
	Request instruction example	Set_InchStep 1

No.	Description	
	Request instruction explanation	Sets the step size level for jog operation. Para1: Step size level value, range 0 to 4 • 0: Exit jog mode • 1: Jog level G1, step size 0.05 • 2: Jog level G2, step size 0.3 • 3: Jog level G3, step size 2 • 4: Custom jog step size, the step size is the set value of the jog parameter The jog unit for joint frame is degree, and the unit for other frame is mm (for custom step size, refer to the use of Set_StepMotionJ instruction).
	Response instruction format	ok or e*:
	Response instruction explanation	Setting completes, or returns an error.
	Remarks	-
18	Request instruction format	AxisJog Para1 Para2
	Request instruction example	AxisJog 1 1

No.	Description	
	Request instruction explanation	Single-axis jog motion command • Para1: Axis number: When the currently activated mechanical unit is Robot, the range is 1 to 6. In joint coordinates, they correspond to the J1 to J6 axes respectively. In non-joint coordinates, they correspond to X/Y/Z/RZ/RY/RX respectively. When the currently active mechanical unit is external mechanical unit, the axis range is 1 to the total number of axes of the external mechanical unit. • Para2: Single-axis jog motion command ◦ 0: Stop ◦ 1: Forward motion ◦ -1: Reverse motion
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid after data stream mode is OFF in manual mode
19	Request instruction format	AxisInch Para1 Para2
	Request instruction example	AxisInch 1 1

No.	Description	
	Request instruction explanation	Jog motion command • Para1: Axis number When the currently activated mechanical unit is Robot, the range is 1 to 6. In joint coordinates, they correspond to the J1 to J6 axes respectively. In non-joint coordinates, they correspond to X/Y/Z/RZ/RY/RX respectively. When mecUnit is an external mechanical unit, the axis range is 1 to the total number of axes in the external mechanical unit. • Para2: Jog command ◦ 0: Stop ◦ 1: Forward jog ◦ -1: Reverse jog
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid after data stream mode is OFF in manual mode
20	Request instruction format	PoseAlign Para1 Para2
	Request instruction example	PoseAlign 4 1

No.	Description	
	Request instruction explanation	Orientation calibration (for 6-axis robots) • Para1: Frame ◦ -1: Flange frame ◦ 1: Joint frame ◦ 2: Base frame ◦ 3: Tool frame ◦ 4: Workobject frame ◦ 5: World frame (currently supports flange frame (the currently activated tool is an external tool), world frame, base frame, workobject frame) • Para2: Orientation calibration command ◦ 0: Stop orientation calibration ◦ 1: Start orientation calibration
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Do not perform any other motions during the orientation calibration process. The data stream mode will be automatically turned off after the orientation calibration is performed.
21	Request instruction format	Set_ActiveMechUnit Para1
	Request instruction example	Set_ActiveMechUnit Robot

No.	Description	
	Request instruction explanation	Sets the active mechanical unit for manual mode motion. Para1: Mechanical unit name (case-insensitive)
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
22	Request instruction format	Get_ActiveMechUnit
	Request instruction explanation	Gets the active mechanical unit for manual mode motion.
	Response instruction format	=Robot or e*:

No.	Description	
	Response instruction explanation	Returns the name of active mechanical unit for manual mode motion. Instruction returns an error.
	Remarks	-
23	Request instruction format	• Set_TeachCoordinate ON • Set_TeachCoordinate OFF
	Request instruction explanation	• Turns on the coordination switch. • Turns off the coordination switch.
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Conditions for enabling the coordination switch: • The current active mechanical unit is the guide rail, and the current frame is the world frame or workobject frame. • The current active mechanical unit is the positioner, and the current frame is the workobject frame, with the workobject's associated mechanical unit name being the currently active mechanical unit (i.e., the positioner).

No.	Description	
24	Request instruction format	Get_TeachCoordinate
	Request instruction explanation	Gets the coordination switch for manual mode motion.
	Response instruction format	=0 or e*:
	Response instruction explanation	• Returns the coordination switch for manual mode motion. ◦ 0: Coordination switch OFF ◦ 1: Coordination switch ON • Instruction returns an error.
	Remarks	-
25	Request instruction format	Home Para1
	Request instruction example	Home 0
	Request instruction explanation	Returns to the working origin 0. Para1: Home index, range 0 to 4

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
26	Request instruction format	Set_DynamicBrake Para1
	Request instruction example	Set_DynamicBrake 1
	Request instruction explanation	Sets the dynamic brake switch. Para1: • 0: Disable dynamic brake switch • 1: Enable dynamic brake switch
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.

No.	Description	
	Remarks	-
27	Request instruction format	Get_DynamicBrake
	Request instruction explanation	Current obtained dynamic brake switch status
	Response instruction format	=0 or e*:
	Response instruction explanation	• Returns the current dynamic brake switch status: ◦ 0: Dynamic brake switch OFF ◦ 1: Dynamic braking switch ON • Instruction returns an error.
	Remarks	-
28	Request instruction format	MovJGP Para1 Para2 Para3
	Request instruction example	MovJGP GP0 100 0 Para1: GP0 Para2: 100 Para3: 0

No.	Description	
	Request instruction explanation	Moves to GP0 point using Movj, with a speed of 100 and an interpolation accuracy of 0. • Para1: Target global position point number, range 0 to 9999 • Para2: Motion speed, range 1 to 100 • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length.
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
29	Request instruction format	MovLGP Para1 Para2 Para3

No.	Description	
	Request instruction example	MovLGP GP0 100 0 Para1: GP0 Para2: 100 Para3: 0
	Request instruction explanation	Moves to GP0 point using Movl, with a speed of 100 and an interpolation accuracy of 0. • Para1: Target global position point number, range 0 to 9999 • Para2: Motion speed, range 1 to 100 • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length.
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode

No.	Description	
30	Request instruction format	MovCGP Para1 Para2 Para3 Para4
	Request instruction example	MovCGP GP0 GP1 100 0 Para1: GP0 Para2: GP1 Para3: 100 Para4: 0
	Request instruction explanation	Moves using Movc, with GP0 as the arc intermediate point and GP1 as the arc endpoint, and with a speed of 100 and an interpolation accuracy of 0. • Para1: Global position point number of the arc intermediate point, range 0 to 9999 • Para2: Global position point number of the arc endpoint, range 0 to 9999 • Para3: Motion speed, range 1 to 100 • Para4: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length.
	Response instruction format	ok or e*:

No.	Description	
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
31	Request instruction format	MovJGP Para1 Para2 Para3 Para4
	Request instruction example	MovJGP GP0 100 0 [7,1,0,0.05] Para1: GP0 Para2: 100 Para3: 0 Para4: [7,1,0,0.05]

No.	Description	
	Request instruction explanation	Moves to the set point using Movj, with a speed of 100 and an interpolation accuracy of 0, and outputs DO7 as ON after 0.05s from the start of motion. • Para1: Target global position point number, range 0 to 9999 • Para2: Motion speed, range 1 to 100 • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para4: Motion I/O control information, supporting 1 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter 1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals

No.	Description	
		are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.

No.	Description	
	Remarks	Valid only in data stream mode
32	Request instruction format	MovLGP Para1 Para2 Para3 Para4
	Request instruction example	MovLGP GP0 100 0 [7,1,2,5] Para1: GP0 Para2: 100 Para3: 0 Para4: [7,1,2,5]

No.	Description	
	Request instruction explanation	Moves to the GP0 point using Movl, with a speed of 100 and an interpolation accuracy of 0, and outputs DO7 as ON after moving 5 mm from the start of motion. • Para1: Target global position point number, range 0 to 9999 • Para2: Motion speed, range 1 to 100 • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para4: Motion I/O control information, supporting 1 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter 1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals

No.	Description	
		are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.

No.	Description	
	Remarks	Valid only in data stream mode
33	Request instruction format	MovCGP Para1 Para2 Para3 Para4 Para5
	Request instruction example	MovCGP GP0 GP1 100 0 [7,1,2,5] Para1: GP0 Para2: GP1 Para3: 100 Para4: 0 Para5: [7,1,2,5]

No.	Description	
	Request instruction explanation	Moves to the GP0 point using Movc, with a speed of 100 and an interpolation accuracy of 0, and outputs DO7 as ON after moving 5 mm from the start of motion. • Para1: Global position point number of the arc intermediate point, range 0 to 9999 • Para2: Global position point number of the arc endpoint, range 0 to 9999 • Para3: Motion speed, range 1 to 100 • Para4: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para5: Motion I/O control information, supporting 1 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter 1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the

No.	Description	
		robot reaches Value% of the path after motion starts, Value<0 indicates that signals are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.

No.	Description	
	Remarks	Valid only in data stream mode
34	Request instruction format	MovJRobP Para1 Para2 Para3 Para4 Para5
	Request instruction example	MovJRobP [370.000,50.000,765.000,-115.000,-75.000,-55.000 ; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] 0,50,0,0.000,0.000,0.000,0.000 1 1 1,0,1,1.000; Para1: [370.000,50.000,765.000,-115.000,-75.000,-55.000 ; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] Para2: 0,50,0,0.000,0.000,0.000,0.000 Para3: 1 Para4: 1 Para5: 1,0,1,1.000

No.	Description	
	Request instruction explanation	Moves to the position point with specified value using Movj, and controls the corresponding I/O. • Para1: Target position point information • Para2: Speed parameter, format: speed type, percentage speed property of whether the numeric speed is affected by the global percentage speed, TCP speed, TCP orientation speed, external axis linear speed, external axis rotational speed ◦ Speed type: ▪ 0: Percentage speed type ▪ 1: Numeric speed type ◦ Percentage speed: range 1 to 100 ◦ Property of whether the numeric speed is affected by the global percentage speed: ▪ 1: Not affected by the global percentage speed ▪ 0: Affected by the global percentage speed • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para4: Number of motion I/O groups, range 0 to 3 • Para5: Motion I/O control information, supporting 1 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter

No.	Description	
		1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	• Valid only in data stream mode • Does not support numeric speed type
35	Request instruction format	MovLRobP Para1 Para2 Para3 Para4 Para5
	Request instruction example	MovLRobP [370.000,50.000,765.000,-115.000,-75.000,-55.000 ; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] 0,50,0,0.000,0.000,0.000,0.000 1 1 1,0,1,1.000; Para1: [370.000,50.000,765.000,-115.000,-75.000,-55.000 ; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] Para2: 0,50,0,0.000,0.000,0.000,0.000 Para3: 1 Para4: 1 Para5: 1,0,1,1.000

No.	Description	
	Request instruction explanation	Moves to the position point with specified value using Movl, and controls the corresponding I/O. • Para1: Target position point information • Para2: Speed parameter, format: speed type, percentage speed property of whether the numeric speed is affected by the global percentage speed, TCP speed, TCP orientation speed, external axis linear speed, external axis rotational speed ◦ Speed type: ▪ 0: Percentage speed type ▪ 1: Numeric speed type ◦ Percentage speed: range 1 to 100 ◦ Property of whether the numeric speed is affected by the global percentage speed: ▪ 1: Not affected by the global percentage speed ▪ 0: Affected by the global percentage speed • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para4: Number of motion I/O groups, range 0 to 3 • Para5: Motion I/O control information, supporting 1 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter

No.	Description	
		1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
36	Request instruction format	MovCRobP Para1 Para2 Para3 Para4 Para5 Para6
	Request instruction example	MovCRobP [370.000,50.000,765.000,-115.000,-75.000,-55.000; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] [375.263,44.809,769.512,-118.306,-68.009,-47.077; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] 0,50,0,0.000,0.000,0.000,0.000 1 1 1,0,1,1.000; Para1: [370.000,50.000,765.000,-115.000,-75.000,-55.000; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] Para2: [375.263,44.809,769.512,-118.306,-68.009,-47.077; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] Para3: 0,50,0,0.000,0.000,0.000,0.000 Para4: 1 Para5: 1 Para6: 1,0,1,1.000

No.	Description	
	Request instruction explanation	Moves to the position point with specified value using Movc, and controls the corresponding I/O. • Para1: Position point information of the arc intermediate point • Para2: Position point information of the arc endpoint • Para3: Speed parameter, format: speed type, percentage speed property of whether the numeric speed is affected by the global percentage speed, TCP speed, TCP orientation speed, external axis linear speed, external axis rotational speed ◦ Speed type: ▪ 0: Percentage speed type ▪ 1: Numeric speed type ◦ Percentage speed: range 1 to 100 ◦ Property of whether the numeric speed is affected by the global percentage speed: ▪ 1: Not affected by the global percentage speed ▪ 0: Affected by the global percentage speed • Para4: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para5: Number of motion I/O groups, range 0 to 3 • Para6: Motion I/O control information, supporting 1 to 3 groups, format: [I/O number

No.	Description	
		1, I/O output value 1, I/O type 1, I/O parameter 1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
37	Request instruction format	JumpRobP Para1 Para2 Para3 Para4 Para5
	Request instruction example	JumpRobP [370.000,50.000,765.000,-115.000,-75.000,-55.000 ; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] 0,50,0,0.000,0.000,0.000,0.000 1 1 1,0,1,1.000; Para1: [370.000,50.000,765.000,-115.000,-75.000,-55.000 ; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] Para2: 0,50,0,0.000,0.000,0.000,0.000 Para3: 1 Para4: 1 Para5: 1,0,1,1.000

No.	Description	
	Request instruction explanation	Moves to the position point with specified value using Jump, and controls the corresponding I/O. • Para1: Target position point information • Para2: Speed parameter, format: speed type, percentage speed property of whether the numeric speed is affected by the global percentage speed, TCP speed, TCP orientation speed, external axis linear speed, external axis rotational speed ◦ Speed type: ▪ 0: Percentage speed type ▪ 1: Numeric speed type ◦ Percentage speed: range 1 to 100 ◦ Property of whether the numeric speed is affected by the global percentage speed: ▪ 1: Not affected by the global percentage speed ▪ 0: Affected by the global percentage speed • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para4: Number of motion I/O groups, range 0 to 3 • Para5: Motion I/O control information, supporting 1 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter

No.	Description	
		1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
38	Request instruction format	JumpLRobP Para1 Para2 Para3 Para4 Para5
	Request instruction example	JumpLRobP [370.000,50.000,765.000,-115.000,-75.000,-55.000 ; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] 0,50,0,0.000,0.000,0.000,0.000 1 1 1,0,1,1.000; Para1: [370.000,50.000,765.000,-115.000,-75.000,-55.000 ; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] Para2: 0,50,0,0.000,0.000,0.000,0.000 Para3: 1 Para4: 1 Para5: 1,0,1,1.000

No.	Description	
	Request instruction explanation	Moves to the position point with specified value using JumpL, and controls the corresponding I/O. • Para1: Target position point information • Para2: Speed parameter, format: speed type, percentage speed property of whether the numeric speed is affected by the global percentage speed, TCP speed, TCP orientation speed, external axis linear speed, external axis rotational speed ◦ Speed type: ▪ 0: Percentage speed type ▪ 1: Numeric speed type ◦ Percentage speed: range 1 to 100 ◦ Property of whether the numeric speed is affected by the global percentage speed: ▪ 1: Not affected by the global percentage speed ▪ 0: Affected by the global percentage speed • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para4: Number of motion I/O groups, range 0 to 3 • Para5: Motion I/O control information, supporting 1 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter

No.	Description	
		1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
39	Request instruction format	JumpGP Para1 Para2 Para3
	Request instruction example	JumpGP GP0 100 0 Para1: GP0 Para2: 100 Para3: 0
	Request instruction explanation	Moves to GP0 point using Jump, with a speed of 100 and an interpolation accuracy of 0. • Para1: Target global position point number, range 0 to 9999 • Para2: Motion speed, range 1 to 100 • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length

No.	Description	
		◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length.
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
40	Request instruction format	JumpLGP Para1 Para2 Para3
	Request instruction example	JumpLGP GP0 100 0 Para1: GP0 Para2: 100 Para3: 0

No.	Description	
	Request instruction explanation	Moves to GP0 point using JumpL, with a speed of 100 and an interpolation accuracy of 0. • Para1: Target global position point number, range 0 to 9999 • Para2: Motion speed, range 1 to 100 • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length.
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
41	Request instruction format	JumpGP Para1 Para2 Para3 Para4

No.	Description	
	Request instruction example	JumpGP GP0 100 0 [7,1,0,0.05] Para1: GP1 Para2: 100 Para3: 0 Para4: [7,1,0,0.05]
	Request instruction explanation	Moves to the GP0 point using Jump, with a speed of 100 and an interpolation accuracy of 0, and outputs DO7 as ON after 0.05s from the start of motion. • Para1: Target global position point number, range 0 to 9999 • Para2: Motion speed, range 1 to 100 • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para4: Motion I/O control information, supporting 1 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter 1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output

No.	Description	
		Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.

No.	Description	
	Remarks	Valid only in data stream mode
42	Request instruction format	JumpLGP Para1 Para2 Para3 Para4
	Request instruction example	JumpLGP GP0 100 0 [7,1,0,0.05] Para1: GP0 Para2: 100 Para3: 0 Para4: [7,1,0,0.05]

No.	Description	
	Request instruction explanation	Moves to the GP0 point using JumpL, with a speed of 100 and an interpolation accuracy of 0, and outputs DO7 as ON after moving 5 mm from the start of motion. • Para1: Target global position point number, range 0 to 9999 • Para2: Motion speed, range 1 to 100 • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para4: Motion I/O control information, supporting 1 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter 1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals

No.	Description	
		are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.

No.	Description	
	Remarks	Valid only in data stream mode
43	Request instruction format	MovJAbsRobJPPara1 Para2 Para3 Para4 Para5
	Request instruction example	MovJAbsRobJP [0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000; 0.000,0.000,0.000,0.000,0.000,0.000] 0,50,0,0.000,0.000,0.000,0.000 0 1 1,0,1,1.000 Para1: [0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000; 0.000,0.000,0.000,0.000,0.000,0.000] Para2: 0,50,0,0.000,0.000,0.000,0.000 Para3: 0 Para4: 1 Para5: 1,0,1,1.000

No.	Description	
	Request instruction explanation	Moves to the position point with specified value using Movj, and controls the corresponding I/O. • Para1: Target joint point information • Para2: Speed parameter, format: speed type, percentage speed property of whether the numeric speed is affected by the global percentage speed, TCP speed, TCP orientation speed, external axis linear speed, external axis rotational speed ◦ Speed type: ▪ 0: Percentage speed type ▪ 1: Numeric speed type ◦ Percentage speed: range 1 to 100 ◦ Property of whether the numeric speed is affected by the global percentage speed: ▪ 1: Not affected by the global percentage speed ▪ 0: Affected by the global percentage speed • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para4: Number of motion I/O groups, range 0 to 3 • Para5: Motion I/O control information, supporting 1 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter

No.	Description	
		1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	• Valid only in data stream mode • Does not support numeric speed type
44	Request instruction format	MovJAbsRobIndexJP Para1 Para2 Para3 Para4 Para5
	Request instruction example	MovJAbsRobIndexJP GJP1 0,50,0,0.000,0.000,0.000,0.000 0 1 1,0,1,1.000 Para1: GJP1 Para2: 0,50,0,0.000,0.000,0.000,0.000 Para3: 0 Para4: 1 Para5: 1,0,1,1.000

No.	Description	
	Request instruction explanation	• Moves to the global joint point with specified number using Movj, and controls the corresponding I/O. • Para1: Target global joint point number, range 0 to 9999 • Para2: Speed parameter, format: speed type, percentage speed property of whether the numeric speed is affected by the global percentage speed, TCP speed, TCP orientation speed, external axis linear speed, external axis rotational speed ◦ Speed type: ▪ 0: Percentage speed type ▪ 1: Numeric speed type ◦ Percentage speed: range 1 to 100 ◦ Property of whether the numeric speed is affected by the global percentage speed: ▪ 1: Not affected by the global percentage speed ▪ 0: Affected by the global percentage speed • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para4: Number of motion I/O groups, range 0 to 3 • Para5: Motion I/O control information, supporting 1 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter

No.	Description	
		1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	• Valid only in data stream mode • Does not support numeric speed type
45	Request instruction format	IndCMove Para1 Para2 Para3 Para4 Para5
	Request instruction example	IndCMove 6 20.0 50.0 50.0 Robot Para1: 6 Para2: 20.0 Para3: 50.0 Para4: 50.0 Para5: Robot

No.	Description	
	Request instruction explanation	Enables continuous motion of the independent axis. • Para1: Axis number ◦ For a 6-axis robot, only J6 is supported (for a four-axis robot, only the independent axis of the non-Robot mechanical unit is supported). ◦ For the non-Robot mechanical unit, the range is 1 to the maximum number of axes configured for the mechanical unit. • Para2: Numeric speed type, positive and negative signals the direction. The specified speed is affected by the global speed percentage. The formula is: Running speed = Speed set by the instruction x Global speed percentage, in °/s, or mm/s for linear axes. Range: -10000°/s to 10000°/s (Speed = 0 indicates that continuous motion of the independent axis stops) • Para3: Acceleration range: 20 to 100 • Para4: Deceleration range: 20 to 100 • Para5: Mechanical unit name ◦ For J6 of a 6-axis robot, the parameter is "Robot". ◦ For a non-Robot mechanical unit, the parameter is the mechanical unit name.
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.

No.	Description	
	Remarks	• Valid only in data stream mode • Call this interface when all axes, except for independent axes, have stopped moving. • When Speed exceeds the upper limit of the controller speed, the upper limit of the controller speed is prevails.
46	Request instruction format	IndSpeed Para1 Para2
	Request instruction example	IndSpeed 6 Robot
	Request instruction explanation	Checks whether the independent axis reaches the set speed. • Para1: Axis number ◦ For a 6-axis robot, only J6 is supported (for a four-axis robot, only the independent axis of the non-Robot mechanical unit is supported). ◦ For the non-Robot mechanical unit, the range is 1 to the maximum number of axes configured for the mechanical unit. • Para2: Mechanical unit name If the mechanical unit is a robot, the parameter is "Robot".
	Response instruction format	=1 or e*:
	Response instruction explanation	• Returns whether the set speed has been reached. ◦ 1: Specified speed reached ◦ 0: Specified speed not reached • Instruction returns an error.

No.	Description	
	Remarks	-
47	Request instruction format	IndResetOld Para1 Para2
	Request instruction example	IndResetOld 6 Robot
	Request instruction explanation	Resets the independent axis, switches it to the normal mode, and retains the turn count of the axis. • Para1: Axis number ◦ For a 6-axis robot, only J6 is supported (for a four-axis robot, only the independent axis of the non-Robot mechanical unit is supported). ◦ For the non-Robot mechanical unit, the range is 1 to the maximum number of axes configured for the mechanical unit. • Para2: Mechanical unit name If the mechanical unit is a robot, the parameter is "Robot".
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	• Valid only in data stream mode • Call this interface when all axes, except for independent axes, have stopped moving.

No.	Description	
48	Request instruction format	IndReset Para1 Para2 Para3 Para4
	Request instruction example	IndReset 6 100.0 0 Robot Para1: 6 Para2: 100.0 Para3: 0 Para4: Robot
	Request instruction explanation	Resets the independent axis, switches it to the normal mode, and adjusts the turn count of the axis according to RefPos and Direction. • Para1: Axis number ◦ For a 6-axis robot, only J6 is supported (for a four-axis robot, only the independent axis of the non-Robot mechanical unit is supported). ◦ For the non-Robot mechanical unit, the range is 1 to the maximum number of axes configured for the mechanical unit. • Para2: Reference position, range -720° to +720° • Para3: ◦ -1: Adjust the multi-turn count of the axis so that the reference position refNum is on the negative side of the modified current position. (Within 360°) ◦ 0: Adjust the multi-turn count of the axis so that the modified current position is as close as possible to the specified refNum position. (Within ±180°) ◦ 1: Adjust the multi-turn count of the axis so that the reference position refNum is on the positive side of the modified current position. (Within 360°) • Para4: Mechanical unit name If the mechanical unit is a robot, the parameter is "Robot".

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	• Valid only in data stream mode • Call this interface when all axes, except for independent axes, have stopped moving.
49	Request instruction format	IMC100_Get_IndCMoveSts Para1 Para2
	Request instruction example	IMC100_Get_IndCMoveSts 6 Robot
	Request instruction explanation	Queries the continuous motion switching status of the independent axis. • Para1: Axis number ◦ For a 6-axis robot, only J6 is supported (for a four-axis robot, only the independent axis of the non-Robot mechanical unit is supported). ◦ For the non-Robot mechanical unit, the range is 1 to the maximum number of axes configured for the mechanical unit. • Para2: Mechanical unit name If the mechanical unit is a robot, the parameter is "Robot".

No.	Description	
	Response instruction format	=1 or e*:
	Response instruction explanation	• Returns whether the continuous motion switching of the independent axis is completed. ◦ 0: Continuous motion switching of independent axis not completed ◦ 1: Continuous motion switching of independent axis completed • Instruction returns an error.
	Remarks	-
50	Request instruction format	Get_IndResetSts Para1 Para2
	Request instruction example	Get_IndResetSts 6 Robot
	Request instruction explanation	Queries the reset status of the independent axis. • Para1: Axis number ◦ For a 6-axis robot, only J6 is supported (for a four-axis robot, only the independent axis of the non-Robot mechanical unit is supported). ◦ For the non-Robot mechanical unit, the range is 1 to the maximum number of axes configured for the mechanical unit. • Para2: Mechanical unit name If the mechanical unit is a robot, the parameter is "Robot".

No.	Description	
	Response instruction format	=1 or e*:
	Response instruction explanation	• Returns whether the reset status of the independent axis is completed. ◦ 0: Reset of independent axis not completed ◦ 1: Reset of independent axis completed • Instruction returns an error.
	Remarks	-
51	Request instruction format	Get_MotionStsExceptIndAxis
	Request instruction explanation	Queries the motion status of the robot, except the independent axis.
	Response instruction format	=1 or e*:
	Response instruction explanation	• Returns the system motion status: ◦ 0: Stopped/Motion completed ◦ 1: In motion • Instruction returns an error.
	Remarks	-

No.	Description	
52	Request instruction format	Get_RobPHere
	Request instruction explanation	Gets the current position. Ret: TCP of the current tool relative to the current workobject If you need to get the values for different tools or different workobjects, set in advance. • Set_ToolCNum • Set_WobjNum
	Response instruction format	=425.011,2.741,718.553,-73.956,-88.026,-105.676; 0,0,0,1; 0.000,0.000,0.000,0.000,0.000,0.000 or e*:
	Response instruction explanation	Returns the current position value, or instruction returns an error.
	Remarks	-
53	Request instruction format	Get_RobJPHere
	Request instruction explanation	Gets the current joint position.

No.	Description	
	Response instruction format	=0.368,0.075,4.656,0.101,-5.205,1.800,0.000,0.000; 0.000,0.000,0.000,0.000,0.000,0.000 or e*:
	Response instruction explanation	Returns the current joint position value, or instruction returns an error.
	Remarks	-
54	Request instruction format	Set_ActiveMechUnitPosFormat Para1
	Request instruction example	Set_ActiveMechUnitPosFormat 1
	Request instruction explanation	Sets the position display format of the active mechanical unit. Para1: Position format • 1: Joint frame format • 2: Base frame format • 3: Flange frame format • 4: Workobject frame format • 5: World frame format
	Response instruction format	ok or e*:

No.	Description	
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
55	Request instruction format	Get_ActiveMechUnitPos
	Request instruction explanation	Gets the position of the active mechanical unit. • Ret: mechanical unit position, format: position format of the current active mechanical unit • Position format of the current active mechanical unit: ◦ 1: Joint frame format ◦ 2: Base frame format ◦ 3: Flange frame format ◦ 4: Workobject frame format ◦ 5: World frame format • Mechanical unit position: Array length 8 (Double type)
	Response instruction format	=1,0.368,0.075,4.656,0.101,-5.205,1.800,0.000,0.000 or e*:

No.	Description	
	Response instruction explanation	Returns the current position format and position values. • Format: Position format of the current active mechanical unit ◦ 1: Joint frame format ◦ 2: Base frame format ◦ 3: Flange frame format ◦ 4: Workobject frame format ◦ 5: World frame format • Mechanical unit position (array length 8) • Instruction returns an error.
	Remarks	-
56	Request instruction format	Get_PosHerePulse
	Request instruction explanation	Queries the pulse value of the current position point.
	Response instruction format	=87421.0,17473.0,961184.0,17651.0,-758105.0,207982.0 or e*:
	Response instruction explanation	Returns the pulse value of the current position point, or instruction returns an error.
	Remarks	-

No.	Description	
57	Request instruction format	Get_RobJToRobP Para1 Para2
	Request instruction example	Get_RobJToRobP 0.368,0.075,4.656,0.101,-5.205,1.800,0.000,0.000; 0.000,0.000,0.000,0.000,0.000,0.000 0,0,0
	Request instruction explanation	Calculates the position point based on the joint point. • Para1: Joint point to be converted • Para2: Tool number, workobject number, load number
	Response instruction format	=541.960956,3.493287,898.202116,-73.329554,-88.019769,-106.302493; 0,0,0,1; 0.000000,0.000000,0.000000,0.000000,0.000000,0.000000 or e*:
	Response instruction explanation	Returns the converted position point, or instruction returns an error.
	Remarks	-
58	Request instruction format	Get_RobPToRobJ Para1 Para2
	Request instruction example	Get_RobPToRobJ 541.960,3.493,898.202,-73.329,-88.019,-106.302; 0,0,0,1; 0.000,0.000,0.000,0.000,0.000,0.000 0,0,0

No.	Description	
	Request instruction explanation	Calculates the joint point based on the position point. • Para1: Position point to be converted • Para2: Tool number, workobject number, load number
	Response instruction format	=0.368,0.075,4.656,0.115,-5.205,1.787,0.000,0.000; 0.000,0.000,0.000,0.000,0.000,0.000 or e*:
	Response instruction explanation	Returns the converted joint point position, or instruction returns an error.
	Remarks	-
59	Request instruction format	Get_FixWobjRobP Para1 Para2
	Request instruction example	Get_FixWobjRobP 329.271,146.581,715.472,-16.876,-69.828,-152.263; 0,-1,0,1; 0.000,0.000,0.000,0.000,0.000,0.000 2
	Request instruction explanation	Converts a position in the world frame into a position in the frame of a designated workobject that is fixed and not held by the robot (RobHold = 0, and UFFix = 1). • Para1: Robot point position to be converted (point position in the world frame) • Para2: Workobject number

No.	Description	
	Response instruction format	=329.271,146.581,715.472,-16.876,-69.828,-152.263; 0,-1,0,1; 0.000,0.000,0.000,0.000,0.000,0.000 or e*:
	Response instruction explanation	Returns the converted point in the workobject frame, or instruction returns an error.
	Remarks	-
60	Request instruction format	Get_RobHoldWobjRobP Para1 Para2 Para3
	Request instruction example	Get_RobHoldWobjRobP 370.763,49.600,765.861,-114.829,-76.872,-56.185; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000 1 -765.569,204.516,-117.829,26.024,-68.456,146.217; 0,-1,0,1; 0.000,0.000,0.000,0.000,0.000,0.000 Para1: 370.763,49.600,765.861,-114.829,-76.872,-56.185; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000 Para2: 1 Para3: -765.569,204.516,-117.829,26.024,-68.456,146.217; 0,-1,0,1; 0.000,0.000,0.000,0.000,0.000,0.000
	Request instruction explanation	Converts a position in the world frame into a position in the frame of a designated workobject that is held by the robot (RobHold = 1, and UFFix = 1). • Para1: Robot point position to be converted (point position in the world frame) • Para2: Workobject number, range: 0 to 15 • Para3: Coordinate value of the flange in the world frame

No.	Description	
	Response instruction format	=1171.964,217.554,821.784,25.938,20.712,23.176; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000 or e*:
	Response instruction explanation	Returns the converted point in the workobject frame, or instruction returns an error.
	Remarks	-
61	Request instruction format	Get_OffsetJ_RobJP Para1 Para2
	Request instruction example	Get_OffsetJ_RobJP 10.000,10.000,10.000,10.000,10.000,10.000,0.000,0.000; 0.000,0.000,0.000,0.000,0.000,0.000 5.000,5.000,5.000,5.000,5.000,5.000
	Request instruction explanation	Queries the point position after joint point offset. • Para1: Original joint point • Para2: Translation variable
	Response instruction format	=15.000,15.000,15.000,15.000,15.000,15.000,0.000, 0.000; 0.000,0.000,0.000,0.000,0.000,0.000 or e*:
	Response instruction explanation	Returns the offset point position result, or instruction returns an error.

No.	Description	
	Remarks	-
62	Request instruction format	Get_Offset_RobP Para1 Para2
	Request instruction example	Get_Offset_RobP 370.000,50.000,765.000,-115.000,-75.000,-55.000; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000 5.000,5.000,5.000,5.000,5.000,5.000
	Request instruction explanation	Queries the offset point position in the workobject frame. • Para1: Original point • Para2: Translation variable
	Response instruction format	=375.000,55.000,770.000,-89.834,-71.398,-76.263; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000 or e*:
	Response instruction explanation	Returns the offset point position result, or instruction returns an error.
	Remarks	-
63	Request instruction format	Get_OffsetT_RobP Para1 Para2

No.	Description	
	Request instruction example	Get_OffsetT_RobP 370.000,50.000,765.000,-115.000,-75.000,-55.000; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000 5.000,5.000,5.000,5.000,5.000,5.000
	Request instruction explanation	Queries the offset point position in the tool frame. • Para1: Original point • Para2: Translation variable
	Response instruction format	=375.263,44.809,769.512,-118.306,-68.009,-47.077; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000 or e*:
	Response instruction explanation	Returns the offset point position result, or instruction returns an error.
	Remarks	-
64	Request instruction format	Get_SysErrSts
	Request instruction explanation	Queries the current fault status of the system.
	Response instruction format	=1 or e*:

No.	Description	
	Response instruction explanation	• Returns fault status: ◦ 0: No fault ◦ bit0: 1 indicates a system alarm, 0 indicates no system alarm ◦ bit1: 1 indicates a system warning, 0 indicates no system warning • Instruction returns an error.
	Remarks	-
65	Request instruction format	Get_SysErr
	Request instruction explanation	Queries the system error code.
	Response instruction format	=100dh or e*:
	Response instruction explanation	Returns fault value (hexadecimal), or instruction returns an error.
	Remarks	-
66	Request instruction format	Get_TaskPrgPath Para1

No.	Description	
	Request instruction example	Get_TaskPrgPath 0
	Request instruction explanation	Queries the program path for the specified task channel in the robot project. Para1: Task channel, range: 0 to maximum number of tasks configured in the project, where 0 is the main task
	Response instruction format	=TeachProgram/newprj/main.pro or e*:
	Response instruction explanation	Returns the program path of the specified task channel, or instruction returns an error.
	Remarks	-
67	Request instruction format	Get_TaskRunSts Para1
	Request instruction example	Get_TaskRunSts 0
	Request instruction explanation	Queries the running status of the specified task channel in the robot project. Para1: Task channel, range: [0-15], where 0 is the main task

No.	Description	
	Response instruction format	=0 or e*:
	Response instruction explanation	• Returns the running status of the task channel: ◦ 0: Stop ◦ 1: Start/Resume ◦ 10: Ready ◦ 100: Task not activated ◦ -1: Task activated but program not configured • Instruction returns an error.
	Remarks	-
68	Request instruction format	Get_TaskProgramLine Para1
	Request instruction example	Get_TaskProgramLine 0
	Request instruction explanation	Queries the processing line number of the specified task channel. Para1: Task channel, range: [0-15], where 0 is the main task
	Response instruction format	=1 or e*:

No.	Description	
	Response instruction explanation	Returns the value (processing line number of the current task channel), or instruction returns an error.
	Remarks	-
69	Request instruction format	Get_CurMotionLine
	Request instruction explanation	Queries the line number of the current executing motion instruction.
	Response instruction format	=2 or e*:
	Response instruction explanation	Returns the value (line number of the current executing motion instruction), or instruction returns an error.
	Remarks	-
70	Request instruction format	Get_InitSts

No.	Description	
	Request instruction explanation	Queries the system initialization status.
	Response instruction format	=11 or e*:
	Response instruction explanation	Returns the system initialization status, range -1 to 11, where 11 indicates that the system startup completes or the instruction returns an error.
	Remarks	-
71	Request instruction format	Get_AccRamp
	Request instruction explanation	Queries the acceleration jerk for the motion segment in data stream mode.
	Response instruction format	=50.000000, 50.000000 or e*:

No.	Description	
	Response instruction explanation	Returns the queried acceleration jerk of the motion segment in data stream mode, format: acceleration jerk percentage of the start segment (range 10.0 to 100.0), acceleration jerk percentage of the end segment (range 10.0 to 100.0). Instruction returns an error.
	Remarks	-
72	Request instruction format	Get_RapidMove Para1
	Request instruction example	Get_RapidMove 0
	Request instruction explanation	Queries the optimal trajectory switch for the current motion type. Para1: Motion type • 0: cp motion • 1: ptp motion
	Response instruction format	=0 or e*:
	Response instruction explanation	• Returns the optimal trajectory switch for the specified motion type: ◦ 0: Disable optimal planning for current motion type ◦ 1: Enable optimal trajectory for current motion • Instruction returns an error.
	Remarks	-

No.	Description	
73	Request instruction format	Get_CoordType
	Request instruction explanation	Queries the current frame type.
	Response instruction format	=1 or e*:
	Response instruction explanation	• Returns the current frame type: ◦ 1: Joint frame ◦ 2: Base frame ◦ 3: Tool frame ◦ 4: Workobject frame ◦ 5: World frame • Instruction returns an error.
	Remarks	-
74	Request instruction format	Get_Vel
	Request instruction explanation	Queries the current speed level.

No.	Description	
	Response instruction format	=100 or e*:
	Response instruction explanation	Returns the current speed level, range 1 to 100, or instruction returns an error.
	Remarks	-
75	Request instruction format	Get_Mode
	Request instruction explanation	Queries the current operating mode of the system.
	Response instruction format	=1 or e*:
	Response instruction explanation	• Returns the current operating mode of the system: ◦ 1: Manual ◦ 2: Auto ◦ 3: Single step ◦ 5: Continuous • Instruction returns an error.
	Remarks	-

No.	Description	
76	Request instruction format	Get_DsMode
	Request instruction explanation	Queries whether the data stream mode is enabled.
	Response instruction format	=0 or e*:
	Response instruction explanation	• Returns the data stream mode status: ◦ 0: OFF ◦ 1: ON/CONTINUE ◦ 2: PAUSE • Instruction returns an error.
	Remarks	-
77	Request instruction format	Get_InchMode
	Request instruction explanation	Queries the jog mode.
	Response instruction format	=0 or e*:

No.	Description	
	Response instruction explanation	• Returns the jog mode: ◦ 0: Non-jog ◦ 1: Jog • Instruction returns an error.
	Remarks	-
78	Request instruction format	Get_SlewMode
	Request instruction explanation	Queries the rotation optimization mode for the J4 axis of the SCARA robot or the J6 axis of the standard 6-axis robot.
	Response instruction format	=0 or e*:

No.	Description	
	Response instruction explanation	• Returns the rotation optimization mode for the J4 axis of the SCARA robot or the J6 axis of the standard 6-axis robot. ◦ 0: No optimization, subject to arm parameters in the position variable. ◦ 1: Optimization mode 1 to ensure that the rotation of SCARA robot's J4/6-axis robot's J6 is within the range of -180° to 180° during motion. ◦ 2: Ensure that the SCARA robot's SCARA-J4/6-axis robot's J6 always moves in the shortest way during motion. The robot calculates and determines whether reaching the target point requires J4/J6 to rotate by 180°. If the angle difference is ≤180°, the target point is fully reached. If >180°, then J4/J6 moves in the opposite direction and ultimately moves to a position where J4/J6 is 360° away from the target point. If the reverse motion of J4/J6 exceeds the limit range of the robot, it will stop the motion, and alarm. ◦ 3: Ensure that the SCARA robot's J4/6-axis robot's J6 always moves in the shortest way during motion, similar to mode 2. The difference is that if the reverse motion of J4/J6 exceeds the limit range of the robot, there will be no alarm. Instead, no reverse motion will be carried out and the original normal motion will be fully adopted. • Instruction returns an error.
	Remarks	-
79	Request instruction format	Get_EStopSts

No.	Description	
	Request instruction explanation	Queries the current status of the emergency stop.
	Response instruction format	=0 or e*:
	Response instruction explanation	• Returns the emergency stop switch status: ◦ 0: Emergency stop switch released ◦ 1: Emergency stop switch pressed • Instruction returns an error.
	Remarks	-
80	Request instruction format	Get_MotorSts
	Request instruction explanation	Queries the current motor enable status.
	Response instruction format	=0 or e*:

No.	Description	
	Response instruction explanation	• Returns the motor enable status: ◦ 0: Disable ◦ 1: Enable • Instruction returns an error.
	Remarks	-
81	Request instruction format	Get_MotionSts
	Request instruction explanation	Queries the current robot motion status.
	Response instruction format	=0 or e*:
	Response instruction explanation	• Returns the current robot motion status: ◦ 0: Stopped/Motion completed ◦ 1: In motion ◦ 2: Motion interrupted • Instruction returns an error.
	Remarks	-
82	Request instruction format	Get_SysMode

No.	Description	
	Request instruction explanation	Queries the current system mode.
	Response instruction format	=0 or e*:
	Response instruction explanation	• Returns the current system mode: ◦ 0: Normal mode ◦ >0: Internal test mode • Instruction returns an error.
	Remarks	-
83	Request instruction format	Get_PrgRunTime
	Request instruction explanation	Queries the project runtime.
	Response instruction format	=0 or e*:

No.	Description	
	Response instruction explanation	Returns the project runtime, or instruction returns an error.
	Remarks	-
84	Request instruction format	Get_CurCmdNum
	Request instruction explanation	Queries the number of the currently successfully sent motion instruction.
	Response instruction format	=1 or e*:
	Response instruction explanation	Returns the number of the currently successfully sent motion instruction, instruction returns an error.
	Remarks	Valid only in data stream mode Note: After the data stream is turned off and on, the count will restart from 0.
85	Request instruction format	Get_CurCmdSts

No.	Description	
	Request instruction explanation	Queries the actual completion status (whether in position) of the current motion instruction.
	Response instruction format	=0 or e*:
	Response instruction explanation	• Returns the number of the currently successfully sent motion instruction: ◦ 0: Motion not completed ◦ 1: Motion completed • Instruction returns an error.
	Remarks	Valid only in data stream mode
86	Request instruction format	Get_CurCmdCacheNum
	Request instruction explanation	Queries the number of motion data items in the current cache.
	Response instruction format	=0 or e*:

No.	Description	
	Response instruction explanation	Returns the number of motion data in the current cache, or instruction returns an error.
	Remarks	-
87	Request instruction format	Get_CmdSts Para1
	Request instruction example	Get_CmdSts 1
	Request instruction explanation	Queries the actual completion status of a motion instruction with the specified number. Para1: The motion instruction with the instruction number to be queried
	Response instruction format	=0 or e*:
	Response instruction explanation	• Returns the actual completion status of the motion instruction with the specified number: ◦ 0: Motion not completed ◦ 1: Motion completed • Instruction returns an error.
	Remarks	Valid only in data stream mode

No.	Description	
88	Request instruction format	Get_DINum
	Request instruction explanation	Queries the total number of current system DIs.
	Response instruction format	=16 or e*:
	Response instruction explanation	Returns the total number of current DIs (subject to actual configuration), or instruction returns an error.
	Remarks	-
89	Request instruction format	Get_DONum
	Request instruction explanation	Queries the total number of current system DOs.
	Response instruction format	=16 or e*:

No.	Description	
	Response instruction explanation	Returns the total number of current system DOs (subject to the actual configuration), or instruction returns an error.
	Remarks	-
90	Request instruction format	Get_ADNum
	Request instruction explanation	Queries the total number of current AD analog input ports in the system.
	Response instruction format	=0 or e*:
	Response instruction explanation	Returns the total number of current AD analog input ports in the system (the result is the total number of activated ADs for IR-Link and EtherCAT). Instruction returns an error.
	Remarks	-
91	Request instruction format	Get_DANum

No.	Description	
	Request instruction explanation	Queries the total number of DA analog output ports in the system.
	Response instruction format	=0 or e*:
	Response instruction explanation	Returns the total number of current DA analog output ports in the system (the result is the total number of activated DAs for IR-Link and EtherCAT). Instruction returns an error.
	Remarks	-
92	Request instruction format	Get_DI Para1
	Request instruction example	Get_DI 0
	Request instruction explanation	Queries the status of DI0. Para1: DI number to be queried, range 0 to the maximum actual configuration value
	Response instruction format	=0 or e*:

No.	Description	
	Response instruction explanation	• Returns the status of DI. ◦ 0: OFF ◦ 1: ON • Instruction returns an error.
	Remarks	-
93	Request instruction format	Get_DIGroup Para1
	Request instruction example	Get_DIGroup 0
	Request instruction explanation	Queries the status of DI0 group, Para1: DI group to be queried, with 8 channels per group (parameter range 0 to 7, subject to actual configuration)
	Response instruction format	=0 or e*:
	Response instruction explanation	• Returns the status of DI group: 0 to 255, for each bit. ◦ 0: OFF ◦ 1: ON • Instruction returns an error.
	Remarks	-

No.	Description	
94	Request instruction format	Get_AD Para1
	Request instruction example	Get_AD 0
	Request instruction explanation	Queries the input value of AD0. Para1: AD index to be queried, range 0 to the maximum actual configuration value (range for IR-Link: 0 to total number of activated ADs (upper limit ≤15) for IR-Link, range for EtherCAT: 64 to total number of activated ADs (upper limit ≤79) for EtherCAT)
	Response instruction format	=0 or e*:
	Response instruction explanation	Returns the AD input value, in mA for current type and V for voltage type, or instruction returns an error.
	Remarks	-
95	Request instruction format	Get_DOCfg Para1
	Request instruction example	Get_DOCfg 0

No.	Description	
	Request instruction explanation	Queries the configuration authority of DO0. Para1: DO number to be queried, range 0 to the maximum actual configuration value
	Response instruction format	=1 or e*:
	Response instruction explanation	• Returns the configuration authority of DO: ◦ 1: Controlled by RC ◦ 0: Controlled by PLC • Instruction returns an error.
	Remarks	-
96	Request instruction format	Get_DOGroupCfg Para1
	Request instruction example	Get_DOGroupCfg 0
	Request instruction explanation	Queries the configuration authority of DO0 group. Para1: DO group to be queried, range 0 to the maximum actual configuration value
	Response instruction format	=255 or e*:

No.	Description	
	Response instruction explanation	• Returns the status of DO: bit0 to bit7 represent the configuration authority of each DO in the group. ◦ 1: Controlled by RC ◦ 0: Controlled by PLC • Instruction returns an error.
	Remarks	-
97	Request instruction format	Get_DO Para1
	Request instruction example	Get_DO 0
	Request instruction explanation	Queries the output status of DO0. Para1: DO to be queried, range 0 to the maximum actual configuration value
	Response instruction format	=0 or e*:
	Response instruction explanation	• Returns the output status of DO: ◦ 0: OFF ◦ 1: ON • Instruction returns an error.
	Remarks	-

No.	Description	
98	Request instruction format	Get_DOGroup Para1
	Request instruction example	Get_DOGroup 0
	Request instruction explanation	Queries the status of DO0 group, with a group number of 0. Para1: DO to be queried, range 0 to the maximum actual configuration value
	Response instruction format	=0 or e*:
	Response instruction explanation	• Returns the output status of DO group: 0 to 255, for each bit. ◦ 0: OFF ◦ 1: ON • Instruction returns an error.
	Remarks	-
99	Request instruction format	Get_DACfg Para1
	Request instruction example	Get_DACfg 0

No.	Description	
	Request instruction explanation	Queries the configuration authority of DA. Para1: DA index to be queried, range 0 to the maximum actual configuration value (range for IR-Link: 0 to total number of activated DAs (upper limit ≤15) for IR-Link, range for EtherCAT: 64 to total number of activated DAs (upper limit ≤79) for EtherCAT)
	Response instruction format	=1 or e*:
	Response instruction explanation	• Returns the configuration authority of DA: ◦ 1: Controlled by RC ◦ 0: Controlled by PLC • Instruction returns an error.
	Remarks	-
100	Request instruction format	Get_DA Para1
	Request instruction example	Get_DA 0
	Request instruction explanation	Queries the output value of DA0. Para1: DA index to be queried, range 0 to the maximum actual configuration value (range for IR-Link: 0 to total number of activated DAs (upper limit ≤15) for IR-Link, range for EtherCAT: 64 to total number of activated DAs (upper limit ≤79) for EtherCAT)

No.	Description	
	Response instruction format	=0 or e*:
	Response instruction explanation	Returns the DA output value, using voltage value for voltage type and current value for current type, or instruction returns an error.
	Remarks	-
101	Request instruction format	Get_DevSts
	Request instruction explanation	Queries the connection status of system devices.
	Response instruction format	=100105 or e*:

No.	Description	
	Response instruction explanation	Connection status of system device, representing the query result Description: • First digit: Network card 1 status (value 1 in the example) ◦ 0. Not connected ◦ 1: Connected ◦ 2: Disabled • Second digit: Network card 2 status (value 0 in the example) • Third digit: USB device status (value 0 in the example) ◦ 0. Not connected ◦ 1: Connected and successfully mounted ◦ 2: Mount failed • Fourth digit: SD card status (value 1 in the example) ◦ 0. Not connected ◦ 1: Connected and successfully mounted ◦ 2: Mount failed ◦ 3: File system format error • Fifth digit: EtherCAT 1 communication status (value 0 in the example) ◦ 0: Communication normal ◦ 1: Slave offline ◦ 2: Network cable not connected ◦ 3: Non-EtherCAT device connected ◦ 4: Disabled • Sixth digit: IR-Link 1 communication status (value 5 in the example) ◦ 0: Communication normal ◦ 1: Slave offline ◦ 2: Device not mounted ◦ 3: Non-IR-Link device connected ◦ 4: Disabled ◦ 5: Device not configured ◦ 6: Communication abnormal • Instruction returns an error.

No.	Description	
	Remarks	-
102	Request instruction format	Get_FwVersion
	Request instruction explanation	Queries the system controller software version.
	Response instruction format	=S03.22R or e*:
	Response instruction explanation	Returns the system software version number, or instruction returns an error.
	Remarks	-
103	Request instruction format	Get_SysTime
	Request instruction explanation	Queries the current system time.

No.	Description	
	Response instruction format	=20170101083000 or e*:
	Response instruction explanation	Returns the system time (including year, month, day, hour, minute, and second), or instruction returns an error.
	Remarks	-
104	Request instruction format	Get_RobotType
	Request instruction explanation	Queries the current system model.
	Response instruction format	=IRS311-3-60TP5-S0_01740352 or e*:
	Response instruction explanation	Returns the system model, or instruction returns an error.
	Remarks	-

No.	Description	
105	Request instruction format	Get_ArmType Para1 Para2 Para3 Para4 Para5 Para6
	Request instruction example	Get_ArmType -30.0,-30.0,-30.0,-30.0,-30.0,-30.0
	Request instruction explanation	Queries the arm parameters corresponding to the joint coordinate values.
	Response instruction format	=-1,-1,-1,1 or e*:
	Response instruction explanation	Returns the arm parameters corresponding to the joint coordinate values, or instruction returns an error.
	Remarks	-
106	Request instruction format	Get_ServoSts
	Request instruction explanation	Queries the servo fault status.

No.	Description	
	Response instruction format	=1000 or e*:
	Response instruction explanation	• Returns the fault status of each servo, from left to right, starting with Servo 0 and ending with the last servo in the system. The total number of servos includes both the robot axes and external axes, up to 8 servos. The fault status of each servo: ◦ 0: No fault ◦ bit0: 1 for servo alarm, 0 for no servo alarm ◦ bit1: 1 for servo warning, 0 for no servo warning • Instruction returns an error.
	Remarks	-
107	Request instruction format	Get_ServoErr Para1
	Request instruction example	Get_ServoErr 0
	Request instruction explanation	Queries the error code of a single servo.
	Response instruction format	=0740h or e*:

No.	Description	
	Response instruction explanation	Returns the error code (with the servo number as the first parameter) of a single servo, or instruction returns an error.
	Remarks	-
108	Request instruction format	Get_StrPara
	Request instruction explanation	Queries the structure parameters.
	Response instruction format	=225.000,175.000,0.000,16.000,0.000,0.000 or e*:
	Response instruction explanation	Returns the structural parameter values (The number of valid parameters is related to the specific robot model. For SCARA robots, the last two parameters are meaningless.) Instruction returns an error.
	Remarks	-
109	Request instruction format	Set_StrPara [Para1,Para2,Para3,Para4,Para5,Para6]

No.	Description	
	Request instruction example	Set_StrPara [225.000,175.000,0.000,16.000,0.000,0.000]
	Request instruction explanation	Sets the structure parameters (The number of parameters is 6. The valid parameters are related to the robot model. For SCARA robots, the last two parameters are meaningless and can be filled with 0.)
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes (switching mode by calling UserLogin, the same applies below).
110	Request instruction format	Get_StrParaComp
	Request instruction explanation	Queries the structure compensation parameters.

No.	Description	
	Response instruction format	=0.000,0.000,0.000,0.000,0.000,0.000 or e*:
	Response instruction explanation	Returns the structural parameter values (The number of valid parameters is related to the specific robot model. For SCARA robots, the last two parameters are meaningless.) Instruction returns an error.
	Remarks	-
111	Request instruction format	Set_StrParaComp Para1
	Request instruction example	Set_StrParaComp [0.000,0.000,0.000,0.000,0.000,0.000]
	Request instruction explanation	Sets the structure compensation parameters. Para1: Structure compensation parameter
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.

No.	Description	
	Remarks	Available for Admin and Manufacturer modes.
112	Request instruction format	Get_RdctRatio
	Request instruction explanation	Queries the reduction ratio.
	Response instruction format	=50.000,50.000,2.000,10.000 or e*:
	Response instruction explanation	Returns the reduction ratio value (The number of parameters is related to the number of axes; since the reduction ratio value will be modified during robot factory calibration, there are differences in the reduction ratio parameter values for each robot.) Instruction returns an error.
	Remarks	-
113	Request instruction format	Set_RdctRatio Para1
	Request instruction example	Set_RdctRatio [50.000,50.000,2.000,10.000]

No.	Description	
	Request instruction explanation	Sets the reduction ratio. Para1: Reduction ratio, the number of valid parameters is related to the number of axes. For SCARA robot, the number of parameters is 4.
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.
114	Request instruction format	Get_CpParaM
	Request instruction explanation	Queries the master coupling parameters.
	Response instruction format	=0.000,0.000,1.000,0.000 or e*:

No.	Description	
	Response instruction explanation	Returns the master coupling parameters (The number of valid parameters equals the specific number of axes. For SCARA robots, the number of parameters is 4.) Instruction returns an error.
	Remarks	-
115	Request instruction format	Set_CpParaM Para1
	Request instruction example	Set_CpParaM [0.000,0.000,1.000,0.000]
	Request instruction explanation	Sets the master coupling parameters. Para1: Master coupling parameters
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.

No.	Description	
116	Request instruction format	Get_CpParaS
	Request instruction explanation	Queries the slave coupling parameters.
	Response instruction format	=0.000,0.000,0.000,1.000 or e*:
	Response instruction explanation	Returns the slave coupling parameters (The number of valid parameters equals the specific number of axes. For SCARA robots, the number of parameters is 4.) Instruction returns an error.
	Remarks	-
117	Request instruction format	Set_CpParaS Para1
	Request instruction example	Set_CpParaS [0.000,0.000,0.000,1.000]
	Request instruction explanation	Sets the slave coupling parameters. Para1: Slave coupling parameters

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.
118	Request instruction format	Get_HomeJPos Para1
	Request instruction example	Get_HomeJPos 1
	Request instruction explanation	Queries the work origin 0. Para1: Work origin number, range 0 to 4
	Response instruction format	=0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000; 0.000,0.000,0.000,0.000,0.000,0.000 or e*:
	Response instruction explanation	Returns the origin coordinate value, or instruction returns an error.

No.	Description	
	Remarks	-
119	Request instruction format	Set_HomeJPos Para1 Para2
	Request instruction example	Set_HomeJPos 0 0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000; 0.000,0.000,0.000,0.000,0.000,0.000
	Request instruction explanation	Sets the work origin 0. Para1: Work origin number, range: 0 to 4 Para2: Joint coordinate values (Robot joints (array length 8) + External axis joints (array length 6))
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.
120	Request instruction format	Get_ZeroPos

No.	Description	
	Request instruction explanation	Queries the absolute zero point value.
	Response instruction format	=0.0,0.0 or e*:
	Response instruction explanation	Returns the pulse values corresponding to the zero point (The number of pulse values equals the number of axes. For SCARA robots, the number of parameters is 4.) Instruction returns an error.
	Remarks	-
121	Request instruction format	Set_ZeroPos Para1
	Request instruction example	Set_ZeroPos [0,0,0,0]
	Request instruction explanation	Sets the absolute zero point value. Para1: Absolute zero point value, the number of valid parameters equals the number of axes. For SCARA robot, the number of parameters is 4.
	Response instruction format	ok or e*:

No.	Description	
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.
122	Request instruction format	Get_InchStep
	Request instruction explanation	Queries the jog stepsize level.
	Response instruction format	=1 or e*:
	Response instruction explanation	• Returns the jog stepsize level: ◦ 1: Stepsize is 0.05 ◦ 2: Stepsize is 0.5 ◦ 3: Stepsize is 2 ◦ 4: Stepsize is set to jog parameter value • Instruction returns an error.
	Remarks	-
123	Request instruction format	Get_StepMotionJ

No.	Description	
	Request instruction explanation	Queries the custom joint stepsize for jog operation.
	Response instruction format	=0 or e*:
	Response instruction explanation	Returns the custom joint stepsize for jog operation, unit °, or instruction returns an error.
	Remarks	-
124	Request instruction format	Set_StepMotionJ Para1
	Request instruction example	Set_StepMotionJ 0.100
	Request instruction explanation	Sets the custom joint stepsize for jog operation. Para1: Joint stepsize, range: 0.01 to 10
	Response instruction format	ok or e*:

No.	Description	
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.
125	Request instruction format	Get_StepMotionL
	Request instruction explanation	Queries the custom linear stepsize for jog operation.
	Response instruction format	=0 or e*:
	Response instruction explanation	Returns the custom linear stepsize for jog operation, unit mm, or instruction returns an error.
	Remarks	-
126	Request instruction format	Set_StepMotionL Para1

No.	Description	
	Request instruction example	Set_StepMotionL 0.1000
	Request instruction explanation	Sets the custom linear stepsize for jog operation. Para1: Linear stepsize, range 0.01 to 10
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.
127	Request instruction format	Get_StepMotionR
	Request instruction explanation	Queries the custom orientation stepsize for jog operation.
	Response instruction format	=0 or e*:

No.	Description	
	Response instruction explanation	Returns the custom orientation stepsize for jog operation, unit °, or instruction returns an error.
	Remarks	-
128	Request instruction format	Set_StepMotionR Para1
	Request instruction example	Set_StepMotionR 0.1000
	Request instruction explanation	Sets the custom orientation stepsize for jog operation. Para1: Orientation stepsize, range 0.01 to 10
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.

No.	Description	
129	Request instruction format	Get_TeachVelLimJ
	Request instruction explanation	Queries the joint speed limits for manual mode motion.
	Response instruction format	=36.000,36.000,900.000,90.000 or e*:
	Response instruction explanation	Returns the joint speed limits, which are related to the number of axes, unit °/s. Instruction returns an error.
	Remarks	-
130	Request instruction format	Set_TeachVelLimJ Para1
	Request instruction example	Set_TeachVelLimJ [36.000,36.000,900.000,90.000]
	Request instruction explanation	Sets the joint speed limits for manual mode motion. Para1: Joint speed limits

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.
131	Request instruction format	Get_TeachVelLimL
	Request instruction explanation	Queries the speed limits for manual mode motion in Cartesian frame.
	Response instruction format	=150.000,72.000 or e*:
	Response instruction explanation	Returns the position and orientation speed, or instruction returns an error.
	Remarks	-

No.	Description	
132	Request instruction format	Set_TeachVelLimL Para1
	Request instruction example	Set_TeachVelLimL [150.000,72.000]
	Request instruction explanation	Sets the speed limits for manual mode motion in Cartesian frame. Para1: Position speed, orientation speed
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.
133	Request instruction format	Get_TeachAccLimJ
	Request instruction explanation	Queries the joint acceleration limits for manual mode motion.

No.	Description	
	Response instruction format	=960.000,960.000,24000.000,2400.000 or e*:
	Response instruction explanation	Returns the joint acceleration limits, which are related to the number of axes, or instruction returns an error.
	Remarks	-
134	Request instruction format	Set_TeachAccLimJ Para1
	Request instruction example	Set_TeachAccLimJ [960.000,960.000,24000.000,2400.000]
	Request instruction explanation	Sets the joint acceleration limits for manual mode motion. Para1: Joint acceleration limits. The number of parameters set is related to the number of axes.
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.

No.	Description	
	Remarks	Available for Admin and Manufacturer modes.
135	Request instruction format	Get_TeachAccLimL
	Request instruction explanation	Queries the acceleration limits for manual mode motion in Cartesian frame.
	Response instruction format	=800.000,384.000 or e*:
	Response instruction explanation	Returns the acceleration limits in Cartesian frame, which are position and orientation acceleration respectively, or instruction returns an error.
	Remarks	-
136	Request instruction format	Set_TeachAccLimL Para1
	Request instruction example	Set_TeachAccLimL [800.000,384.000]

No.	Description	
	Request instruction explanation	Sets the acceleration limits for manual mode motion in Cartesian frame. Para1: Acceleration limits. The number of parameters set is related to the number of axes.
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.
137	Request instruction format	Get_RunVelLimJ
	Request instruction explanation	Queries the joint speed limits during runtime.
	Response instruction format	=360.000,360.000,9000.000,900.000 or e*:

No.	Description	
	Response instruction explanation	Returns the joint speed limits during runtime, which are related to the number of axes, unit °/s. Instruction returns an error.
	Remarks	-
138	Request instruction format	Set_RunVelLimJ Para1
	Request instruction example	Set_RunVelLimJ [360.000,360.000,9000.000,900.000]
	Request instruction explanation	Sets the joint speed limits during runtime. Para1: Joint speed limits during runtime. The number of parameters set is related to the number of axes.
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.

No.	Description	
139	Request instruction format	Get_RunVelLimL
	Request instruction explanation	Queries the speed limits during runtime in Cartesian frame.
	Response instruction format	=1500.000,720.000 or e*:
	Response instruction explanation	Returns the speed limits in Cartesian frame, which are position and orientation speeds respectively. Instruction returns an error.
	Remarks	-
140	Request instruction format	Set_RunVelLimL Para1
	Request instruction example	Set_RunVelLimL [1500.000,720.000]
	Request instruction explanation	Sets the speed limits during runtime in Cartesian frame. Para1: Speed limits in Cartesian frame, which are position and orientation speeds respectively

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.
141	Request instruction format	Get_RunAccLimJ
	Request instruction explanation	Queries the joint acceleration limits during runtime.
	Response instruction format	=1920.000,1920.000,48000.000,4800.000 or e*:
	Response instruction explanation	Returns the joint acceleration limits during runtime, which are related to the number of axes. Instruction returns an error.
	Remarks	-

No.	Description	
142	Request instruction format	Set_RunAccLimJ Para1
	Request instruction example	Set_RunAccLimJ [1920.000,1920.000,48000.000,4800.000]
	Request instruction explanation	Sets the joint acceleration limits during runtime. Para1: Joint acceleration limits during runtime. The number of parameters set is related to the number of axes.
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.
143	Request instruction format	Get_RunAccLimL
	Request instruction explanation	Queries the acceleration limits during runtime in Cartesian frame.

No.	Description	
	Response instruction format	=8000.000,3840.000 or e*:
	Response instruction explanation	Returns the acceleration limits in Cartesian frame, which are position and orientation acceleration respectively. Instruction returns an error.
	Remarks	-
144	Request instruction format	Set_RunAccLimL Para1
	Request instruction example	Set_RunAccLimL [8000.000,3840.000]
	Request instruction explanation	Sets the acceleration limits during runtime in Cartesian frame. Para1: Position acceleration, orientation acceleration
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.

No.	Description	
	Remarks	Available for Admin and Manufacturer modes.
145	Request instruction format	Get_StopDecLimJ
	Request instruction explanation	Queries the joint deceleration limits upon stop.
	Response instruction format	=2400.000,2400.000,60000.000,6000.000 or e*:
	Response instruction explanation	Returns the joint deceleration limits upon stop, the number of returned parameters equals the number of robot axes. For SCARA robots, the number of parameters is 4. Instruction returns an error.
	Remarks	-
146	Request instruction format	Set_StopDecLimJ Para1
	Request instruction example	Set_StopDecLimJ [2400.000,2400.000,60000.000,6000.000]

No.	Description	
	Request instruction explanation	Sets the joint deceleration limits upon stop. Para1: Joint deceleration limits upon stop. The number of parameters set equals the number of axes.
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.
147	Request instruction format	Get_StopDecLimL
	Request instruction explanation	Queries the deceleration limits upon stop in Cartesian frame.
	Response instruction format	=10000.000,4800.000 or e*:

No.	Description	
	Response instruction explanation	Returns the deceleration limits upon stop in Cartesian frame, which are position and orientation deceleration respectively. Instruction returns an error.
	Remarks	-
148	Request instruction format	Set_StopDecLimL [10000.000,4800.000]
	Request instruction example	Set_StopDecLimL Para1
	Request instruction explanation	Sets the deceleration limits upon stop in Cartesian frame. Para1: Position and orientation deceleration
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.

No.	Description	
149	Request instruction format	Get_ZonePara
	Request instruction explanation	Queries the transition accuracy parameters.
	Response instruction format	=10.000,3.000 or e*:
	Response instruction explanation	Returns the linear and joint transition accuracy, or instruction returns an error.
	Remarks	-
150	Request instruction format	Set_ZonePara Para1
	Request instruction example	Set_ZonePara [10.000,3.000]
	Request instruction explanation	Sets the transition accuracy parameters. Para1: Linear and joint transition accuracy

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.
151	Request instruction format	Get_AxisNLim Para1
	Request instruction example	Get_AxisNLim J1
	Request instruction explanation	Queries the J1 axis negative limit. Para1: Axis number, related to the number of axes, range J1 to J6
	Response instruction format	=-130.000 or e*:
	Response instruction explanation	Returns the negative axis limit value of the corresponding axis, or instruction returns an error.

No.	Description	
	Remarks	-
152	Request instruction format	Set_AxisNLim Para1 Para2
	Request instruction example	Set_AxisNLim J1 -130.000
	Request instruction explanation	Sets the J1 axis negative limit. • Para1: Axis number, related to the number of axes, range J1 to J6 • Para2: Negative limit parameter value
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.
153	Request instruction format	Get_AxisPLim Para1

No.	Description	
	Request instruction example	Get_AxisPLim J1
	Request instruction explanation	Queries the J1 axis positive limit. Para1: Axis number, related to the number of axes, range J1 to J6
	Response instruction format	=130.000 or e*:
	Response instruction explanation	Returns the negative axis limit value of the corresponding axis, or instruction returns an error.
	Remarks	-
154	Request instruction format	Set_AxisPLim Para1 Para2
	Request instruction example	Set_AxisPLim J1 130.000
	Request instruction explanation	Sets the J1 axis positive limit. • Para1: Axis number, related to the number of axes, range J1 to J6 • Para2: Positive limit parameter value

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.
155	Request instruction format	Get_ToolData Para1
	Request instruction example	Get_ToolData 1
	Request instruction explanation	Queries the tool frame 1 parameters. Para1: Tool number, range 0 to 15
	Response instruction format	=1; 0.000,0.000,0.000,0.000,0.000,0.000; 0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000, 0.000,0.000 or e*:
	Response instruction explanation	Returns the tool parameters, or instruction returns an error.

No.	Description	
	Remarks	-
156	Request instruction format	Set_ToolData Para1 Para2 Para3
	Request instruction example	Set_ToolData 1 1; 0.000,0.000,0.000,0.000,0.000,0.000 0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000 Para1: 1 Para2: 1; 0.000,0.000,0.000,0.000,0.000,0.000 Para3: 0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000,0.000
	Request instruction explanation	Sets the tool frame 1 parameters. • Para1: Tool number, range: 1 to 15 • Para2: Tool parameters, format: whether held by robot; tool frame (array length 6) • Para3: Tool load parameters (mass + center of mass position (array length 3) + center of mass orientation (array length 3) + load inertia (array length 3))
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.

No.	Description	
157	Request instruction format	Get_WobjData Para1
	Request instruction example	Get_WobjData 1
	Request instruction explanation	Queries the workobject frame 1 parameters. Para1: Workobject number, range: 0 to 15
	Response instruction format	0,1; 0.000,0.000,0.000,0.000,0.000,0.000; 0.000,0.000,0.000,0.000,0.000,0.000 or e*:
	Response instruction explanation	Returns the workobject parameters ("; " followed by the associated mechanical unit, no association if no parameter followed). Instruction returns an error.
	Remarks	-
158	Request instruction format	Set_WobjData Para1 Para2
	Request instruction example	Set_WobjData 1 0,1; 0.000,0.000,0.000,0.000,0.000,0.000; 0.000,0.000,0.000,0.000,0.000,0.000;

No.	Description	
	Request instruction explanation	Sets the workobject frame 1 parameters. • Para1: Workobject number, range: 1 to 15 • Para2: Tool parameters, format: whether held by robot, whether fixed user frame; user frame (array length 6); workobject frame (array length 6); associated mechanical unit name (optional)
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.
159	Request instruction format	Get_ToolCNum
	Request instruction explanation	Queries the current tool frame number.
	Response instruction format	=0 or e*:

No.	Description	
	Response instruction explanation	Returns the current active tool number, with a range of 0 to 15. Instruction returns an error.
	Remarks	-
160	Request instruction format	Set_ToolCNum Para1
	Request instruction example	Set_ToolCNum 0
	Request instruction explanation	Sets the current tool frame number. Para1: Tool number, range: 0 to 15
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.

No.	Description	
161	Request instruction format	Get_WobjNum
	Request instruction explanation	Queries the current active workobject number in the system.
	Response instruction format	=0 or e*:
	Response instruction explanation	Returns the current active workobject number, with a range of 0 to 15. Instruction returns an error.
	Remarks	
162	Request instruction format	Set_WobjNum Para1
	Request instruction example	Set_WobjNum 0
	Request instruction explanation	Sets the active workobject number in the system. Para1: Workobject number, range: 0 to 15

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.
163	Request instruction format	Set_CoordType Para1
	Request instruction example	Set_CoordType 1
	Request instruction explanation	Sets the current frame type. Para1: Current frame type, range 1 to 5 • 1: Joint frame • 2: Base frame • 3: Tool frame • 4: Workobject frame • 5: World frame
	Response instruction format	ok or e*:

No.	Description	
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
164	Request instruction format	Get_Interf Para1
	Request instruction example	Get_Interf 0
	Request instruction explanation	Queries the boundary coordinates of interference zone 0. Para1: Interference zone number, range 0 to 7
	Response instruction format	=400.000,0.000,0.000; 372.000,130.000,-5.300 or e*:
	Response instruction explanation	Returns the XYZ coordinates of the two boundary points of interference zone 0, or instruction returns an error.
	Remarks	-

No.	Description	
165	Request instruction format	Set_Interf Para1 Para2
	Request instruction example	Set_Interf 0 [400.000000,0.000000,0.000000; 372.000000,130.000000,-5.300000]
	Request instruction explanation	Sets the boundary coordinates of interference zone 0. • Para1: Interference zone number, range 0 to 7 • Para2: XYZ coordinate values of the two boundary points of interference zone
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.
166	Request instruction format	Get_CurInterf
	Request instruction explanation	Queries the current active interference zone number.

No.	Description	
	Response instruction format	=1 or e*:
	Response instruction explanation	• Returns the current active interference zone number: ◦ 0: Inactive ◦ 1: Active ◦ From left to right, they represent interference zones 0 to 7, respectively. • Instruction returns an error.
	Remarks	This is the activation method of the previous interference zone. For the new one, Get_InterferToolActNum instruction is used.
167	Request instruction format	Set_CurInterf Para1
	Request instruction example	Set_CurInterf 1
	Request instruction explanation	Sets the current active interference zone number. Para1: Interference zone number • 0: Inactive • 1: Active • From left to right, they represent interference zones 0 to 7, respectively.
	Response instruction format	ok or e*:

No.	Description	
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	This is the activation method of the previous interference zone. For the new one, Set_InterferToolActNum instruction is used.
168	Request instruction format	Get_JumpPara
	Request instruction explanation	Queries the height parameters for the current jump motion.
	Response instruction format	=0.0,0 or e*:
	Response instruction explanation	Returns the height parameters for the jump motion, corresponding to LH, MH, RH in sequence. Returns an error.
	Remarks	-
169	Request instruction format	Set_JumpPara Para1

No.	Description	
	Request instruction example	Set_JumpPara [0,0,0]
	Request instruction explanation	Sets the height parameters for the current jump motion. Para1: Height parameters for jump motion, LH, MH, and RH in sequence
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode Note: If this instruction has not been used to set parameters, LH, MH, RH are all default to 0.
170	Request instruction format	Get_PalletPara
	Request instruction explanation	Queries the pallet parameters.

No.	Description	
	Response instruction format	= 2, 3, 1, 15.000000 or e*:
	Response instruction explanation	Returns the pallet parameters: number of rows, number of columns, number of layers, and layer height in sequence. Instruction returns an error.
	Remarks	-
171	Request instruction format	Set_PalletPara Para1
	Request instruction example	Set_PalletPara 2,3,1,15
	Request instruction explanation	Sets the pallet parameters. Para1: Pallet parameters: Number of rows (range 0 to 1000), number of columns (range 0 to 1000), number of layers (range 0 to 1000), and layer height (in mm)
	Response instruction format	ok or e*:
	Response instruction explanation	Setting completes, or returns an error.

No.	Description	
	Remarks	-
172	Request instruction format	Clear_PalletPara
	Request instruction explanation	Clear the pallet parameters
	Response instruction format	ok or e*:
	Response instruction explanation	Clear successful, or returns an error
	Remarks	-
173	Request instruction format	Get_PalletRobP Para1 Para2 Para3 Para4

No.	Description	
	Request instruction example	Get_PalletRobP 378.910,-40.179,435.563,20.388,0.205,-179.000; 0,0,0,0; 0.000,0.000,0.000,0.000,0.000,0.000 502.715,-40.179,435.563,20.388,0.205,-179.000; 0,0,0,0; 0.000,0.000,0.000,0.000,0.000,0.000 502.715,172.988,435.563,20.388,0.205,-179.000; 0,0,0,0; 0.000,0.000,0.000,0.000,0.000,0.000 1,0,0 Para1: 378.910,-40.179,435.563,20.388,0.205,-179.000; 0,0,0,0; 0.000,0.000,0.000,0.000,0.000,0.000 Para2: 502.715,-40.179,435.563,20.388,0.205,-179.000; 0,0,0,0; 0.000,0.000,0.000,0.000,0.000,0.000 Para3: 502.715,172.988,435.563,20.388,0.205,-179.000; 0,0,0,0; 0.000,0.000,0.000,0.000,0.000,0.000 Para4: 1,0,0
	Request instruction explanation	Queries the corresponding pallet points. • Para1: First point for defining the pallet • Para2: Second point for defining the pallet • Para3: Third point for defining the pallet • Para4: Row number of the point to be queried, column number of the point to be queried, layer number of the point to be queried
	Response instruction format	=502.715,-40.179,435.563,20.388,0.205,-179.000; 0,0,0,0; 0.000,0.000,0.000,0.000,0.000,0.000 or e*:
	Response instruction explanation	Returns the values of the pallet points to be queried, or returns an error.
	Remarks	-

No.	Description	
174	Request instruction format	Get_Pallet4RobP Para1 Para2 Para3 Para4 Para5
	Request instruction example	Get_Pallet4RobP 378.910,-40.179,435.563,20.388,0.205,-179.000; 0,0,0,0; 0.000,0.000,0.000,0.000,0.000,0.000 502.715,-40.179,435.563,20.388,0.205,-179.000; 0,0,0,0; 0.000,0.000,0.000,0.000,0.000,0.000 502.715,172.988,435.563,20.388,0.205,-179.000; 0,0,0,0; 0.000,0.000,0.000,0.000,0.000,0.000 368.910,162.988,435.563,20.388,0.205,-179.000; 0,0,0,0; 0.000,0.000,0.000,0.000,0.000,0.000 1,0,0 Para1: 378.910,-40.179,435.563,20.388,0.205,-179.000; 0,0,0,0; 0.000,0.000,0.000,0.000,0.000,0.000 Para2: 502.715,-40.179,435.563,20.388,0.205,-179.000; 0,0,0,0; 0.000,0.000,0.000,0.000,0.000,0.000 Para3: 502.715,172.988,435.563,20.388,0.205,-179.000; 0,0,0,0; 0.000,0.000,0.000,0.000,0.000,0.000 Para4: 368.910,162.988,435.563,20.388,0.205,-179.000; 0,0,0,0; 0.000,0.000,0.000,0.000,0.000,0.000 Para5: 1,0,0
	Request instruction explanation	Queries the corresponding pallet points. • Para1: First point for defining the pallet • Para2: Second point for defining the pallet • Para3: Third point for defining the pallet • Para4: Fourth point for defining the pallet • Para5: Row number of the point to be queried, column number of the point to be queried, layer number of the point to be queried

No.	Description	
	Response instruction format	=502.715,-40.179,435.563,20.388,0.205,-179.000; 0,0,0,0; 0.000,0.000,0.000,0.000,0.000,0.000 or e*:
	Response instruction explanation	Returns the values of the pallet points to be queried, or returns an error.
	Remarks	-
175	Request instruction format	SavePara
	Request instruction explanation	Saves system parameters with power-off retention.
	Response instruction format	ok or e*:
	Response instruction explanation	Save completes, or instruction returns an error.
	Remarks	Available for Admin and Manufacturer modes (switching mode by calling UserLogin)

No.	Description	
176	Request instruction format	RecoverPara
	Request instruction explanation	Recovers the system parameters (to the parameters after the last save operation).
	Response instruction format	ok or e*:
	Response instruction explanation	Recovery completes, or returns an error for the save operation.
	Remarks	Available for Admin and Manufacturer modes.
177	Request instruction format	Get_RobJP Para1
	Request instruction example	Get_RobJP 0
	Request instruction explanation	Queries the joint position parameters of JP0 joint position point. Para1: Index of the position variable JP, existing point within the range JP0 to JP9999

No.	Description	
	Response instruction format	=30.000,30.000,30.000,30.000,30.000,30.000,0.000,0.000; 0.000,0.000,0.000,0.000,0.000,0.000 or e*:
	Response instruction explanation	Returns the joint position information of the point with specified number, or instruction returns an error.
	Remarks	-
178	Request instruction format	Set_RobJP Para1 Para2
	Request instruction example	Set_RobJP 0 15.000,15.000,15.000,15.000,-15.000,-15.000,0.000,0.000; 0.000,0.000,0.000,0.000,0.000,0.000
	Request instruction explanation	Sets the position parameters of JP0 position point. • Para1: Index of the position variable JP, existing point within the range JP0 to JP9999 • Para2: Joint position value
	Response instruction format	ok or e*:
	Response instruction explanation	Setting completes, or returns an error.

No.	Description	
	Remarks	Available for Edit, Admin and Manufacturer modes (switching mode by calling UserLogin, the same applies below).
179	Request instruction format	Set_RobJPHere Para1
	Request instruction example	Set_RobJPHere 0
	Request instruction explanation	Sets the global joint position parameters using the current point parameters. Para1: Index of the position variable JP, existing point within the range JP0 to JP9999
	Response instruction format	ok or e*:
	Response instruction explanation	Setting completes, or returns an error.
	Remarks	Available for Edit, Admin, and Manufacturer modes.
180	Request instruction format	Get_RobP Para1

No.	Description	
	Request instruction example	Get_RobP 0
	Request instruction explanation	Queries the position parameters of P0 position point. Para1: Index of the position variable P, existing point within the range P0 to P9999
	Response instruction format	=370.763,49.600,765.861,-114.829,-76.872,-56.185; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000 or e*:
	Response instruction explanation	Returns the position information of the point with specified number, or instruction returns an error.
	Remarks	-
181	Request instruction format	Set_RobP Para1 Para2
	Request instruction example	Set_RobP 0 370.000,50.000,765.000,-115.000,-77.000,-56.185; 0,0,0,1; 0.000,0.000,0.000,0.000,0.000,0.000
	Request instruction explanation	Sets the position parameters of P0 position point. • Para1: Index of the position variable P, existing point within the range P0 to P9999 • Para2: Position value

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Setting completes, or returns an error.
	Remarks	Available for Edit, Admin, and Manufacturer modes.
182	Request instruction format	Get_RobPFromFile Para1 Para2
	Request instruction example	Get_RobPFromFile 0 P.pts
	Request instruction explanation	Queries the position parameters of P0 position point in the P.pts file. • Para1: Index of the position variable P, existing point within the range P0 to P9999 • Para2: Global position point file name CAUTION File name matching is case-insensitive.
	Response instruction format	=370.763,49.600,765.861,-114.829,-76.872,-56.185; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000 or e*:

No.	Description	
	Response instruction explanation	Returns the position information of the point with specified number in the global position point file. Instruction returns an error.
	Remarks	-
183	Request instruction format	Set_RobPToFile Para1 Para2 Para3
	Request instruction example	Set_RobPToFile 0 370.000,50.000,765.000,-115.000,-77.000,-56.185; 0,0,0,1; 0.000,0.000,0.000,0.000,0.000,0.000 P.pts Para1: 0 Para2: 370.000,50.000,765.000,-115.000,-77.000,-56.185; 0,0,0,1; 0.000,0.000,0.000,0.000,0.000,0.000 Para3: P.pts
	Request instruction explanation	Sets the position parameters of P0 position point in the P.pts file. • Para1: Index of the position variable P, existing point within the range P0 to P9999 • Para2: Position value • Para3: Global position point file name CAUTION File name matching is case-insensitive.
	Response instruction format	ok or e*:

No.	Description	
	Response instruction explanation	Setting completes, or returns an error.
	Remarks	Available for Edit, Admin, and Manufacturer modes.
184	Request instruction format	Get_CurRobPFileName
	Request instruction explanation	Queries the name of the global point file currently loaded by the controller.
	Response instruction format	=P.pts or e*:
	Response instruction explanation	Returns the name of the position file currently loaded by the controller, or instruction returns an error.
	Remarks	-
185	Request instruction format	Set_RobPHere Para1

No.	Description	
	Request instruction example	Set_RobPHere 0
	Request instruction explanation	Sets the global position parameters using the current point parameters. Para1: Index of the position variable P, existing point within the range P0 to P9999
	Response instruction format	ok or e*:
	Response instruction explanation	Setting completes, or returns an error.
	Remarks	-
186	Request instruction format	Set_RobPHereToFile Para1 Para2
	Request instruction example	Set_RobPHereToFile 0 P.pts

No.	Description	
	Request instruction explanation	Sets the global position parameters for the specified file using the current point parameters. • Para1: Index of the position variable P, existing point within the range P0 to P9999 • para2: Global position point file name **CAUTION** File name matching is case-insensitive.
	Response instruction format	ok or e*:
	Response instruction explanation	Setting completes, or returns an error.
	Remarks	-
187	Request instruction format	Get_RobP_Label Para1
	Request instruction example	Get_RobP_Label 0
	Request instruction explanation	Queries the global position label information. Para1: Index of the position variable P, range 0 to 9999

No.	Description	
	Response instruction format	=RobP0 or e*:
	Response instruction explanation	Returns the label information of the point with specified number, or instruction returns an error.
	Remarks	-
188	Request instruction format	Get_RobJP_Label Para1
	Request instruction example	Get_RobJP_Label 0
	Request instruction explanation	Queries the global joint point label information. Para1: Index of the position variable JP, range 0 to 9999
	Response instruction format	=RobJP0 or e*:
	Response instruction explanation	Returns the label information of the point with specified number, or instruction returns an error.

No.	Description	
	Remarks	-
189	Request instruction format	Get_RobLP_Label Para1 Para2 Para3
	Request instruction example	Get_RobLP_Label 1 0 test.pro Para1: 1 Para2: 0 Para3: test.pro
	Request instruction explanation	Queries the local position point label information. • Para1: Task channel, range 0 to 3 • Para2: Index of the position variable LP, range 0 to 9999 • Para3: Name of the program where the local point is located (case-insensitive, .pro suffix required)
	Response instruction format	=Task1LP0 or e*:
	Response instruction explanation	Returns the label information of the point with specified number, or instruction returns an error.
	Remarks	-
190	Request instruction format	Get_SpeedLimitSwitch

No.	Description	
	Request instruction explanation	Queries the status of the speed limit switch in the controller.
	Response instruction format	=0 or e*:
	Response instruction explanation	Returns the status of the speed limit switch, or instruction returns an error.
	Remarks	-
191	Request instruction format	Set_SpeedLimitSwitch Para1
	Request instruction example	Set_SpeedLimitSwitch 0
	Request instruction explanation	Sets the status of the speed limit switch in the controller. Para1: • 0: Speed limit enabled • 1: Speed limit disabled
	Response instruction format	ok or e*:

No.	Description	
	Response instruction explanation	Setting completes, or returns an error.
	Remarks	-
192	Request instruction format	Get_PR Para1
	Request instruction example	Get_PR 0
	Request instruction explanation	Queries the global offset variable PR0. Para1: PR variable number, range 0 to 255
	Response instruction format	=0.000000,0.000000,0.000000,0.000000,0.000000, 0.000000,0.000000 or e*:
	Response instruction explanation	Returns the value of the specified PR variable, or instruction returns an error.
	Remarks	-

No.	Description	
193	Request instruction format	Set_PR Para1 Para2
	Request instruction example	Set_PR 0 [0.0,0.0,0.0,0.0,0.0,0.0]
	Request instruction explanation	Sets the global offset variable PR0. • Para1: PR variable number, range 0 to 255 • Para2: PR value
	Response instruction format	ok or e*:
	Response instruction explanation	Setting completes, or returns an error.
	Remarks	-
194	Request instruction format	Get_B Para1
	Request instruction example	Get_B 0

No.	Description	
	Request instruction explanation	Queries the value of global numeric variable B0. Para1: B variable number, range 0 to 255
	Response instruction format	ok or e*:
	Response instruction explanation	Returns the value of the specified B variable, or instruction returns an error.
	Remarks	-
195	Request instruction format	Set_B Para1 Para2
	Request instruction example	Set_B 0 1
	Request instruction explanation	Sets the value of global numeric variable B0. • Para1: B variable number, range 0 to 255 • Para2: Variable value, range 0 to 255
	Response instruction format	ok or e*:

No.	Description	
	Response instruction explanation	Setting completes, or returns an error.
	Remarks	-
196	Request instruction format	Get_R Para1
	Request instruction example	Get_R 0
	Request instruction explanation	Queries the value of global numeric variable R0. Para1: R variable number, range 0 to 255
	Response instruction format	=1 or e*:
	Response instruction explanation	Returns the value of the specified R variable, or instruction returns an error.
	Remarks	-

No.	Description	
197	Request instruction format	Set_R Para1 Para2
	Request instruction example	Set_R 0 1
	Request instruction explanation	Sets the value of global numeric variable R0. • Para1: R variable number, range 0 to 255 • Para2: Variable value, range -2147483647 to +2147483647
	Response instruction format	ok or e*:
	Response instruction explanation	Setting completes, or returns an error.
	Remarks	Available for Edit, Admin, and Manufacturer modes.
198	Request instruction format	Get_D Para1
	Request instruction example	Get_D 0

No.	Description	
	Request instruction explanation	Queries the value of global numeric variable D0. Para1: D variable number, range 0 to 255
	Response instruction format	=1.000 or e*:
	Response instruction explanation	Returns the value of the specified D variable, or instruction returns an error.
	Remarks	-
199	Request instruction format	Set_D Para1 Para2
	Request instruction example	Set_D 0 1
	Request instruction explanation	Sets the value of global numeric variable D0 to 1. • Para1: D variable number, range 0 to 255 • Para2: Variable value, range -9999999.999 to +9999999.999
	Response instruction format	ok or e*:

No.	Description	
	Response instruction explanation	Setting completes, or returns an error.
	Remarks	Available for Edit, Admin, and Manufacturer modes.
200	Request instruction format	Get_ModbusCoil Para1 Para2
	Request instruction example	Get_ModbusCoil 2048 1
	Request instruction explanation	Queries the value of one coil at address 0 in the Modbus variable area. • Para1: Coil address, range 0 to 8191 • Para2: Number of coils, range 1 to 8
	Response instruction format	=1 or e*:
	Response instruction explanation	Returns the corresponding value, or instruction returns an error.
	Remarks	-

No.	Description	
201	Request instruction format	Set_ModbusCoil Para1 Para2 Para3
	Request instruction example	Set_ModbusCoil 2048 1 1 Para1: 2048 Para2: 1 Para3: 1
	Request instruction explanation	Sets the value of one coil at address 2048 in the Modbus variable area to 0. • Para1: Coil address, range 2048 to 4095, 6144 to 8191 • Para2: Number of coils, range 1 to 8 • Para3: Coil value to be set, range 0 to 255
	Response instruction format	ok or e*:
	Response instruction explanation	Setting completes, or returns an error.
	Remarks	Available for Edit, Admin, and Manufacturer modes.
202	Request instruction format	Get_ModbusReg Para1 Para2 Pra3
	Request instruction example	Get_ModbusReg 16384 2 u16 Para1: 16384 Para2: 2 Para3: u16

No.	Description	
	Request instruction explanation	Queries the value of one register at address 0 in the Modbus variable area. • Para1: Register address, range 0 to 65535 • Para2: Number of registers read, range 1 to 8 • Para3: Register type read, u16 for unsigned 16-bit integer, f32 for single-precision floating point
	Response instruction format	=1.1 or e*:
	Response instruction explanation	Returns the corresponding value, or instruction returns an error.
	Remarks	-
203	Request instruction format	Set_ModbusReg Para1 Para2 Para3 Para4
	Request instruction example	Set_ModbusReg 16384 2 u16 [1,1] Para1: 16384 Para2: 2 Para3: u16 Para4: [1,1]
	Request instruction explanation	Sets the value of one register at address 16384 in the Modbus variable area to 0. • Para1: Register address, range 16384 to 32767, 49152 to 65535 • Para2: Number of registers written • Para3: Register type written, u16 for unsigned 16-bit integer, f32 for single-precision floating point

No.	Description	
		• Para4: Register value to be set
	Response instruction format	ok or e*:
	Response instruction explanation	Setting completes, or returns an error.
	Remarks	Available for Edit, Admin, and Manufacturer modes.
204	Request instruction format	Get_ModbusReg Para1 Para2 Para3
	Request instruction example	Get_ModbusReg 60000 8 f32 Para1: 60000 Para2: 8 Para3: f32
	Request instruction explanation	Queries the register values in the Modbus variable area, with a data type of float. • Para1: Modbus register address, range 0 to 65535 • Para2: Total number of registers read, range 1 to 8 • Para3: float data type

No.	Description	
	Response instruction format	=0.1,0.2,0.3,0.4 or e*:
	Response instruction explanation	Returns the queried register values in the Modbus variable area, or instruction returns an error.
	Remarks	-
205	Request instruction format	Set_ModbusReg Para1 Para2 Para3 Para4
	Request instruction example	Set_ModbusReg 60000 2 f32 1.5 Para1: 60000 Para2: 2 Para3: f32 Para4: 1.5
	Request instruction explanation	Sets the register values in the Modbus variable area, with a data type of float. • Para1: Modbus register address, range 16384 to 32767, 49152 to 65535 • Para2: Total number of registers read, range 1 to 8 • Para3: float data type • Para4: Parameters to be set, use commas to separate multiple parameters
	Response instruction format	ok or e*:

No.	Description	
	Response instruction explanation	Setting completes, or returns an error.
	Remarks	Available for Edit, Admin, and Manufacturer modes.
206	Request instruction format	Get_PlcVar Para1 Para2
	Request instruction example	Get_PlcVar Byte 0
	Request instruction explanation	Queries the PLC Byte-type variable value. • Para1: Byte type • Para2: Byte variable number, range 0 to 255
	Response instruction format	=5 or e*:
	Response instruction explanation	Returns the PLC Byte-type variable value, or query returns an error.
	Remarks	-

No.	Description	
207	Request instruction format	Get_PlcVar Para1 Para2
	Request instruction example	Get_PlcVar Int 0
	Request instruction explanation	Queries the PLC Int-type variable value. • Para1: Int type • Para2: Int variable number, range 0 to 255
	Response instruction format	=15 or e*:
	Response instruction explanation	Returns the PLC Int-type variable value, or query returns an error.
	Remarks	-
208	Request instruction format	Get_PlcVar Para1 Para2
	Request instruction example	Get_PlcVar DInt 0

No.	Description	
	Request instruction explanation	Queries the PLC DInt-type variable value. • Para1: DInt type • Para2: DInt variable number, range 0 to 255
	Response instruction format	=0 or e*:
	Response instruction explanation	Returns the PLC DInt-type variable value, or query returns an error.
	Remarks	-
209	Request instruction format	Get_PlcVar Para1 Para2
	Request instruction example	Get_PlcVar LReal 0
	Request instruction explanation	Queries the PLC LReal-type variable value. • Para1: LReal type • Para2: LReal variable number, range 0 to 255
	Response instruction format	=0.000000 or e*:

No.	Description	
	Response instruction explanation	Returns the PLC LReal-type variable value, or query returns an error.
	Remarks	-
210	Request instruction format	Get_UserAlarm 0
	Request instruction explanation	Queries the content of user alarm 0. Para1: User alarm number, range 0 to 15
	Response instruction format	=UserError1 = e*:
	Response instruction explanation	Returns the corresponding alarm content description. Returns empty value. Instruction returns an error.
	Remarks	-
211	Request instruction format	Set_UserAlarm Para1 Para2

No.	Description	
	Request instruction example	Set_UserAlarm 0 [UserError1]
	Request instruction explanation	Sets the user alarm 0. • Para1: User alarm number, range 0 to 15 • Para2: Alarm content to be set, with a byte length not exceeding 40 bytes **CAUTION** When this instruction has only the first parameter, it means to clear the alarm content.
	Response instruction format	ok or e*:
	Response instruction explanation	Setting completes, or returns an error.
	Remarks	Available for Admin and Manufacturer modes.
212	Request instruction format	Get_Print
	Request instruction explanation	Queries the controller print information, including the content printed by the program print instruction and system error messages.

No.	Description	
	Response instruction format	=PRINT:test = e*:
	Response instruction explanation	Returns the corresponding value. Returns empty value. Instruction returns an error.
	Remarks	-
213	Request instruction format	CurCtrlDev
	Request instruction explanation	Queries the device that currently has the control authority.
	Response instruction format	=0 or e*:
	Response instruction explanation	• Returns the device number. ◦ 0: InoTeachPad/InoRobotLab ◦ 1: InoRobShop ◦ 2: API network device ◦ 3: Remote I/O ◦ 4: Remote Modbus • Instruction returns an error.
	Remarks	-

No.	Description	
214	Request instruction format	CurPermit
	Request instruction explanation	Queries the API network device that currently has the control permission.
	Response instruction format	=0 null =1 Ip:10.44.52.35 Port:5000 e*:
	Response instruction explanation	• Returns the corresponding value. The first return value 0 indicates no API device has the control permission. ◦ 1: Current API device has the control permission, and subsequent return values are the IP address and port of the current device ◦ 2: Other API device has the control permission, and subsequent return values are their IP addresses and ports • Instruction returns an error.
	Remarks	-
215	Request instruction format	AcqPermit AcqPermit forcibly
	Request instruction explanation	• Requests control permission for the current API network client device. • Forcibly acquires the control permission for the current API network client device.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
216	Request instruction format	RemovePermit
	Request instruction explanation	Releases control permission from the current API network client device.
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-

No.	Description	
217	Request instruction format	CurUserType
	Request instruction explanation	Queries the current user mode.
	Response instruction format	=1 or e*:
	Response instruction explanation	• Returns the current user mode. ◦ 0: User mode ◦ 1: Edit mode ◦ 2: Admin ◦ 3: Manufacturer • Instruction returns an error.
	Remarks	-
218	Request instruction format	UserLogin Para1 Para2
	Request instruction example	UserLogin 1 000000

No.	Description	
	Request instruction explanation	Logs in to the corresponding user mode. • Para1: User mode ◦ 0: User mode ◦ 1: Edit mode ◦ 2: Admin ◦ 3: Manufacturer • Para2: Password (the default password for User and Edit modes is "000000")
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
219	Request instruction format	UserLogout
	Request instruction explanation	Logs out of the current login mode. The default mode is User mode.
	Response instruction format	ok or e*:

No.	Description	
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
220	Request instruction format	Set_SysTime Para1
	Request instruction example	Set_SysTime 20170101083000
	Request instruction explanation	Sets the system time. Para1: Date and time in the format of year-month-day-hour-minute-second
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-

No.	Description	
221	Request instruction format	Latch ON Latch OFF
	Request instruction explanation	• Enables the latch function. • Disables the latch function.
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
222	Request instruction format	Get_LatchSts
	Request instruction explanation	Queries the latch function status
	Response instruction format	=1 or e*:

No.	Description	
	Response instruction explanation	• Returns the latch function enable status: ◦ 0: OFF ◦ 1: ON • Instruction returns an error.
	Remarks	-
223	Request instruction format	Get_LatchSum
	Request instruction explanation	Queries the total number of latched position points.
	Response instruction format	=1 or e*:
	Response instruction explanation	Returns the total number of latched position points, or instruction returns an error.
	Remarks	Available for Admin and Manufacturer modes.
224	Request instruction format	Get_LatchRobP Para1

No.	Description	
	Request instruction example	Get_LatchRobP 0
	Request instruction explanation	Reads the position parameters of the corresponding latched point. Para1: Reserved parameter CAUTION Read in sequence. Each position can only be read once.
	Response instruction format	=0; 0.000,0.000,0.000,0.000,0.000,0.000; 0,0,0,0; 0.000,0.000,0.000,0.000,0.000,0.000 or e*:
	Response instruction explanation	Returns the value. The first returned number indicates whether there is latched data (0: no, 1: yes), followed by the latched value, indicating the current tool value relative to the current workobject. Instruction returns an error.
	Remarks	-
225	Request instruction format	Clr_LatchPos
	Request instruction explanation	Clears the latch position.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
226	Request instruction format	Set_CollModeAndAction Para1 Para2
	Request instruction example	Set_CollModeAndAction 0 0

No.	Description	
	Request instruction explanation	Sets the collision detection switch in data stream mode and the actions triggered after a collision alarm. • Para1: Collision detection switch in data stream mode: ◦ 0: Disable collision detection ◦ 1: Enable collision detection • Para2: Action triggered after a collision alarm ◦ 0: Do not define the trigger action ◦ 3: Planned stop after alarm ◦ 4: Immediate stop after alarm ◦ 7: Alarm reset When "0 - Do not define triggered action" is selected, the actual triggered action is determined by the following setting: • API: IMC100_Set_TeachModeCollAction/ IMC100_Set_PlaybackModeCollAction • InoTeachPad: "Settings" > "Motion parameters" > "Collision detection" • InRobotLab: "Controller Parameter configuration" > "Motion parameters" > "Collision detection"
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode

No.	Description	
227	Request instruction format	Get_CollModeAndAction
	Request instruction explanation	Reads the collision detection switch and the actions triggered after a collision alarm, for data stream mode or project instruction.
	Response instruction format	=0.0 or e*:
	Response instruction explanation	• Returns the value. The first return value is the status of collision detection function switch. ◦ 0: Disable collision detection ◦ 1: Enable collision detection • The second return value is the action triggered after a collision alarm. ◦ 0: Do not define the trigger action ◦ 3: Stop upon collision ◦ 7: Move back upon collision • Instruction returns an error.
	Remarks	-
228	Request instruction format	Set_AxisCollMode Para1 Para2
	Request instruction example	Set_AxisCollMode 1 0

No.	Description	
	Request instruction explanation	Sets the collision detection switch for a single axis in data stream mode. • Para1: Axis number, related to the number of axes, range 1 to 6 • Para2: Collision detection switch: ◦ 0: Disable collision detection for a specific axis ◦ 1: Enable collision detection for a specific axis
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
229	Request instruction format	Get_AxisCollMode 1
	Request instruction explanation	Reads the single-axis collision detection switch for data stream mode or project instruction.
	Response instruction format	=0 or e*:

No.	Description	
	Response instruction explanation	Returns the single-axis collision detection switch, or instruction returns an error.
	Remarks	-
230	Request instruction format	Set_AxisCollLevel Para1 Para2
	Request instruction example	Set_AxisCollLevel 1 100
	Request instruction explanation	Sets the collision detection sensitivity for a single axis in data stream mode. • Para1: Axis number, related to the number of axes, range 1 to 6 • Para2: Axis collision detection sensitivity, range 25 to 200 (smaller values indicate higher sensitivity)
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode

No.	Description	
231	Request instruction format	Get_AxisCollLevel 1
	Request instruction explanation	Reads the single-axis collision detection sensitivity for the data stream mode or project instruction.
	Response instruction format	=100 or e*:
	Response instruction explanation	Returns the single-axis collision detection sensitivity, or instruction returns an error.
	Remarks	-
232	Request instruction format	Set_TeachModeCollAction Para1
	Request instruction example	Set_TeachModeCollAction 3
	Request instruction explanation	Sets the action triggered after a collision alarm in manual mode. Para1: Action triggered after a collision alarm • 3: Planned stop after alarm • 4: Immediate stop after alarm

No.	Description	
		• 7: Alarm reset
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
233	Request instruction format	Get_TeachModeCollAction
	Request instruction explanation	Reads the action triggered after a collision alarm in manual mode.
	Response instruction format	=3 or e*:

No.	Description	
	Response instruction explanation	• Returns the action triggered after a collision alarm: ◦ 3: Planned stop after alarm ◦ 4: Immediate stop after alarm ◦ 7: Alarm reset • Instruction returns an error.
	Remarks	-
234	Request instruction format	Set_TeachModeAxisCollLevel Para1 Para2
	Request instruction example	Set_TeachModeAxisCollLevel 1 100
	Request instruction explanation	Sets the collision detection sensitivity in manual mode. • Para1: Axis number, related to the number of axes, range 1 to 6 • Para2: Axis collision detection sensitivity, range 25 to 200
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-

No.	Description	
235	Request instruction format	Get_TeachModeAxisCollLevel Para1
	Request instruction example	Get_TeachModeAxisCollLevel 1
	Request instruction explanation	Reads the collision detection sensitivity in manual mode. Para1: Axis number, related to the number of axes, range 1 to 6
	Response instruction format	=100 or e*:
	Response instruction explanation	Returns the collision detection sensitivity, or instruction returns an error.
	Remarks	-
236	Request instruction format	Set_PlayBackModeCollAction Para1
	Request instruction example	Set_PlayBackModeCollAction 3

No.	Description	
	Request instruction explanation	Sets the action triggered after a collision alarm in auto mode. Para1: Action triggered after a collision alarm • 3: Planned stop after alarm • 4: Immediate stop after alarm • 7: Alarm reset
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
237	Request instruction format	Get_PlayBackModeCollAction
	Request instruction explanation	Reads the action triggered after a collision alarm in auto mode.
	Response instruction format	=0 or e*:

No.	Description	
	Response instruction explanation	Returns the action triggered after a collision alarm, or instruction returns an error.
	Remarks	-
238	Request instruction format	Set_PlayBackModeAxisCollLevel Para1 Para2
	Request instruction example	Set_PlayBackModeAxisCollLevel 1 100
	Request instruction explanation	Sets the collision detection sensitivity in auto mode. • Para1: Axis number, related to the number of axes, range 1 to 6 • Para2: Axis collision detection sensitivity, range 25 to 200
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-

No.	Description	
239	Request instruction format	Get_PlayBackModeAxisCollLevel Para1
	Request instruction example	Get_PlayBackModeAxisCollLevel 1
	Request instruction explanation	Reads the collision detection sensitivity in auto mode. Para1: Axis number, related to the number of axes, range 1 to 6
	Response instruction format	=100 or e*:
	Response instruction explanation	Returns the collision detection sensitivity, or instruction returns an error.
	Remarks	-
240	Request instruction format	Get_RobotAxisNum
	Request instruction explanation	Reads the current number of robot manipulator axes.

No.	Description	
	Response instruction format	=4 or e*:
	Response instruction explanation	Returns the current number of robot manipulator axes, or instruction returns an error.
	Remarks	-
241	Request instruction format	Set_BindTcpSpeedValue Para1 Para2 Para3 Para4 Para5
	Request instruction example	Set_BindTcpSpeedValue 0 0 5 0 500 Para1: 0 Para2: 0 Para3: 5 Para4: 0 Para5: 500
	Request instruction explanation	Binds the linear speed analog output to DA[num]. • Para1: Index of the bound DA, range (0 to the number of active DAs) • Para2: Minimum output analog value, range (output range of selected DA configuration) • Para3: Maximum output analog value, range (output range of selected DA configuration) • Para4: Minimum speed, range (0 to maxSpeed) • Para5: Maximum speed, range (0 to 100000)

No.	Description	
	Response instruction format	=0 or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
242	Request instruction format	Get_BindTcpSpeedValue Para1
	Request instruction example	Get_BindTcpSpeedValue 0
	Request instruction explanation	Gets the setting data of the input DA index. Para1: Index of the bound DA, range (0 to the number of active DAs)
	Response instruction format	=0 or e*:
	Response instruction explanation	Instruction completes, or returns an error.

No.	Description	
	Remarks	-
243	Request instruction format	Close_BindTcpSpeed Para1
	Request instruction example	Close_BindTcpSpeed 0
	Request instruction explanation	Unbinds the linear speed analog value from DA[num]. Para1: Index of the bound DA, range (0 to the number of active DAs)
	Response instruction format	=0 or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
244	Request instruction format	Get_TcpSpeedDAOutAndVel Para1

No.	Description	
	Request instruction example	Get_TcpSpeedDAOutAndVel 0
	Request instruction explanation	Gets the data of the DA output and the linear speed value. Para1: Index of the bound DA, range (0 to the number of active DAs)
	Response instruction format	=0 or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
245	Request instruction format	Set_TraceRecoverMode Para1
	Request instruction example	Set_TraceRecoverMode 1
	Request instruction explanation	Sets the trajectory recovery threshold setting mode. Para1: Trajectory recovery threshold setting mode • 0: Joint mode • 1: Position control mode

No.	Description	
	Response instruction format	=0 or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
246	Request instruction format	Get_TraceRecoverMode
	Request instruction explanation	Gets the trajectory recovery threshold setting mode.
	Response instruction format	=0 or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-

No.	Description	
247	Request instruction format	Set_VarTraceRecoverParams Para1 Para2 Para3 Para4 Para5
	Request instruction example	Set_VarTraceRecoverParams 1 50 90 50 90 Para1: 1 Para2: 50 Para3: 90 Para4: 50 Para5: 90
	Request instruction explanation	Sets the trajectory recovery threshold parameters in auto or manual mode. • Para1: Manual or auto trajectory recovery ◦ 1: Manual ◦ 2: Auto • Para2: TCP distance, range 0 to 20000 (recommended value: 50 for manual mode, 100 for auto mode) • Para3: TCP rotation, range 0 to 360 (recommended value: 15 for manual mode, 30 for auto mode) • Para4: External axis distance, range 0 to 20000 (recommended value: 50 for manual mode, 100 for auto mode) • Para5: External axis rotation, range 0 to 360 (recommended value: 15 for manual mode, 30 for auto mode)
	Response instruction format	=0 or e*:

No.	Description	
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
248	Request instruction format	Get_VarTraceRecoverParams Para1
	Request instruction example	Get_VarTraceRecoverParams 1
	Request instruction explanation	Gets the trajectory recovery threshold parameters in auto or manual mode. Para1: Auto or Manual mode selection for the trajectory recovery threshold position mode • 1: Manual mode • 2: Auto mode
	Response instruction format	=0 or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-

No.	Description	
249	Request instruction format	Set_InterferZoneActStat Para1 Para2
	Request instruction example	Set_InterferZoneActStat 5 1
	Request instruction explanation	Sets the active interference zone. • Para1: Interference zone number, range 0 to 15 • Para2: Activation status: ◦ 0: Deactivate a certain interference zone ◦ 1: Activate a certain interference zone
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
250	Request instruction format	Get_InterferZoneActStat
	Request instruction explanation	Gets the activation status of all interference zones.

No.	Description	
	Response instruction format	=0,1,1,0,1,1,1,1,1,1,1,1,1,1,0 or e*:
	Response instruction explanation	• Outputs the status of all interference zones (16 in total) in sequence: ◦ 0: Deactivate a certain interference zone ◦ 1: Activate a certain interference zone • Instruction returns an error.
	Remarks	Valid only in data stream mode
251	Request instruction format	Set_InterferZonePara Para1 Para2
	Request instruction example	Set_InterferZonePara 5 22,25,0,1,10,13,1,0.0,0.0,0.0,10.1,11.1,12.1 Para1: 5 Para2: 22,25,0,1,10,13,1,0.0,0.0,0.0,10.1,11.1,12.1
	Request instruction explanation	Sets the interference zone parameters. • Para1: Interference zone number, range 0 to 15 • Para2: Input signal, output signal, inside/outside (0: inside/1: outside), alarm status (0: no alarm/1: alarm), safety distance, current workobject number, setting method (0: diagonal/1: datum point + side length), diagonal point 1x/datum point X, diagonal point 1y/datum point Y, diagonal point 1z/datum point Z, diagonal point 2x/offset dx, diagonal point 2y/offset dy, diagonal point 2z/offset dz
	Response instruction format	ok or e*:

No.	Description	
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
252	Request instruction format	Get_InterferZonePara 5
	Request instruction explanation	Gets the parameters of the specified interference zone.
	Response instruction format	= 22,25,0,1,10,13,1,0.0,0.0,0.0,10.1,11.1,12.1
	Response instruction explanation	Gets the configuration parameters of the specified interference zone: Input signal, output signal, inside/outside (0: inside/1: outside), alarm status (0: no alarm/1: alarm), safety distance, current workobject number, setting method (0: diagonal/ 1: datum point + side length), diagonal point 1x/ datum point X, diagonal point 1y/datum point Y, diagonal point 1z/datum point Z, diagonal point 2x/offset dx, diagonal point 2y/offset dy, diagonal point 2z/offset dz
	Remarks	-

No.	Description	
253	Request instruction format	Set_InterferToolActNum Para1
	Request instruction example	Set_InterferToolActNum 14
	Request instruction explanation	Activates the monitoring object. Para1: Monitoring object number, range 0 to 15
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
254	Request instruction format	Get_InterferToolActNum
	Request instruction explanation	Gets the activation status of all interference zones.

No.	Description	
	Response instruction format	=14 or e*:
	Response instruction explanation	Returns the index of the active monitoring object, or instruction returns an error.
	Remarks	Valid only in data stream mode
255	Request instruction format	Set_InterferToolPara Para1 Para2

No.	Description	
	Request instruction example	• Set_InterferToolPara 10 0 ◦ Para1: 10, monitoring object number ◦ Para2: 0, monitoring object type: TCP • Set_InterferToolPara 10 1,1,1,1,1,5,6,7,5 ◦ Para1: 10, monitoring object number ◦ Para2: 1,1,1,1,1,5,6,7,5, monitoring object type: Multi-TCP (1), TCP1 activation status (1), TCP2 activation status (1), TCP3 activation status (1), TCP4 activation status (1), TCP1 number (5), TCP2 number (6), TCP3 number (7), and TCP4 number (5) • Set_InterferToolPara 10 2,10,20 ◦ Para1: 10, monitoring object number ◦ Para2: 2,10,20, monitoring object type: sphere (2), sphere center Z (10), and sphere radius R (20) • Set_InterferToolPara 10 3,0,10.0,11.0,12.0,13.0,14.0,15.0 ◦ Para1: 10, monitoring object number ◦ Para2: 3,0,10.0,11.0,12.0,13.0,14.0,15.0, monitoring object type: cuboid bounding box (3), diagonal point type (0), point 1X (10.0), point 1Y (11.0), point 1Z (12.0), point 2X (13.0), point 2Y (14.0), and point 3Z (15.0) • Set_InterferToolPara 10 3,1,10.0,11.0,12.0,13.0,14.0,15.0 ◦ Para1: 10, monitoring object number ◦ Para2: 3,1,10.0,11.0,12.0,13.0,14.0,15.0, monitoring object type: cuboid bounding box (3), datum point + offset type (1), datum point X (10.0), datum point Y (11.0), datum point Z (12.0), offset dx (13.0), offset dy (14.0), and offset dz (15.0) • Set_InterferToolPara 10 3,2,10.0,11.0,12.0,13.0,14.0,15.0,10.0,11.0,12.0,13.0,14.0,15.0,18.1 ◦ Para1: 10, monitoring object number ◦ Para2: 3,2,10.0,11.0,12.0,13.0,14.0,15.0,10.0,11.0,12.0,13.0,14.0,15.0,18.1, monitoring object type: cuboid bounding box (3), 4 points + height type (2), point 1X (10.0), point 1Y (11.0),

No.	Description	
		point 1Z (12.0), point 2X (13.0), point 2Y (14.0), point 2Z (15.0), point 3X (10.0), point 3Y (11.0), point 3Z (12.0), point 4X (13.0), point 4Y (14.0), point 4Z (15.0), and height H (18.1)

No.	Description	
	Request instruction explanation	Sets the monitoring object parameters. • Para1: Monitoring object number, range 0 to 15 • Para2: ◦ Monitoring object type: TCP ◦ Monitoring object type: Multi-TCP, TCP1 activation status, TCP2 activation status, TCP3 activation status, TCP4 activation status, TCP1 number, TCP2 number, TCP3 number, and TCP4 number ◦ Monitoring object type: Sphere, center Z, radius R ◦ Monitoring object type: cuboid bounding box, setting method - 0 diagonal point, point 1X, point 1Y, point 1Z, point 2X, point 2Y, and point 3Z ◦ Monitoring Object Type: cuboid bounding box, setting method-1 datum point + offset, datum point X, datum point Y, datum point Z, offset dx, offset dy, and offset dz ◦ Monitoring object type: cuboid bounding box, setting method-2 taking 4 points + height, point 1X, point 1Y, point 1Z, point 2X, point 2Y, point 2Z, point 3X, point 3Y, point 3Z, point 4X, point 4Y, point 4Z, and height H
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode

No.	Description	
256	Request instruction format	Get_InterferToolPara 5
	Request instruction explanation	Gets the parameters of the specified monitoring object.
	Response instruction format	• =10 0 • =10 1,1,1,1,1,5,6,7,5 • =10 2,10,20 • =10 3,0,10.0,11.0,12.0,13.0,14.0,15.0 • =10 3,1,10.0,11.0,12.0,13.0,14.0,15.0 • =10 3,2,10.0,11.0,12.0,13.0,14.0,15.0,10.0,11.0,12.0,13.0,14.0,15.0,18.1 For specific meanings, see the setting example.
	Response instruction explanation	Gets the configuration parameters of the specified monitoring object. • Monitoring object type: TCP • Monitoring object type: Multi-TCP, TCP1 activation status, TCP2 activation status, TCP3 activation status, TCP4 activation status, TCP1 number, TCP2 number, TCP3 number, and TCP4 number • Monitoring object type: Sphere, center Z, radius R • Monitoring object type: cuboid bounding box, setting method - 0 diagonal point, point 1X, point 1Y, point 1Z, point 2X, point 2Y, and point 3Z • Monitoring Object Type: cuboid bounding box, setting method - 1 datum point + offset, datum point X, datum point Y, datum point Z, offset dx, offset dy, and offset dz

No.	Description	
		• Monitoring object type: cuboid bounding box, setting method-2 taking 4 points + height, point 1X, point 1Y, point 1Z, point 2X, point 2Y, point 2Z, point 3X, point 3Y, point 3Z, point 4X, point 4Y, point 4Z, and height H
	Remarks	-
257	Request instruction format	SetMemRobP Para1 Para2
	Request instruction example	SetMemRobP 0 370.000,50.000,765.000,-115.000,-77.000,-56.185; 0,0,0,1; 0.000,0.000,0.000,0.000,0.000,0.000
	Request instruction explanation	Sets the position parameters of P0 position point. • Para1: Index of the position variable P, existing point within the range P0 to P9999 • Para2: Position value
	Response instruction format	ok or e*:

No.	Description	
	Response instruction explanation	Setting completes, or returns an error.
	Remarks	-
258	Request instruction format	SetMemRobJP Para1 Para2
	Request instruction example	SetMemRobJP 0 15.000,15.000,15.000,15.000,-15.000,-15.000,0.000,0.000; 0.000,0.000,0.000,0.000,0.000,0.000
	Request instruction explanation	Sets the position parameters of JP0 position point. • Para1: Index of the position variable JP, existing point within the range JP0 to JP9999 • Para2: Joint position value
	Response instruction format	ok or e*:
	Response instruction explanation	Setting completes, or returns an error.
	Remarks	-

No.	Description	
259	Request instruction format	Set_CollDist Para1
	Request instruction example	Set_CollDist 20
	Request instruction explanation	Sets the move back distance after a collision alarm in data stream mode. Para1: The move back distance triggered after a collision alarm, range 5 to 100
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
260	Request instruction format	Get_CollDist
	Request instruction explanation	Reads the move back distance after a collision alarm in data stream mode.

No.	Description	
	Response instruction format	=20 or e*:
	Response instruction explanation	Returns the move back distance after a collision alarm, or instruction returns an error.
	Remarks	-
261	Request instruction format	Set_TeachModeCollDist Para1
	Request instruction example	Set_TeachModeCollDist 20
	Request instruction explanation	Sets the move back distance after a collision alarm in manual mode. Para1: The move back distance triggered after a collision alarm, range 5 to 100
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.

No.	Description	
	Remarks	-
262	Request instruction format	Get_TeachModeCollDist
	Request instruction explanation	Reads the move back distance after a collision alarm in manual mode.
	Response instruction format	=20 or e*:
	Response instruction explanation	Returns the move back distance after a collision alarm, or instruction returns an error.
	Remarks	-
263	Request instruction format	Set_PlayBackModeCollDist Para1
	Request instruction example	Set_PlayBackModeCollDist 20

No.	Description	
	Request instruction explanation	Sets the move back distance after a collision alarm in auto mode. Para1: The move back distance triggered after a collision alarm, range 5 to 100
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
264	Request instruction format	Get_PlayBackModeCollDist
	Request instruction explanation	Reads the move back distance after a collision alarm in auto mode.
	Response instruction format	=20 or e*:

No.	Description	
	Response instruction explanation	Returns the move back distance after a collision alarm, or instruction returns an error.
	Remarks	-
265	Request instruction format	pulse Para1 Para2 Para3
	Request instruction example	pulse 10 10.0 1 Para1: 10 Para2: 10 Para3: 1
	Request instruction explanation	Controls the robot Out port to trigger the pulse. Para1: Out port number for triggering the pulse signal, range 0 to 13823 Para2: Pulse time width, range 0.1 to 2000 Para3: Pulse mode configuration, with a value of 0 or 1 • The value 0 indicates the default mode, outputting pulses opposite to the current ones. • The value 1 indicates the high-level mode, outputting only high pulses.
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.

No.	Description	
	Remarks	-
266	Request instruction format	set_acc Para1 Para2
	Request instruction example	set_acc 10.0 10.0 Para1: 10.0 Para2: 10.0
	Request instruction explanation	Sets the global acceleration percentage. Para1: Acceleration percentage value, range 0 to 120 Para2: Deceleration percentage value, range 0 to 120
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
267	Request instruction format	get_acc

No.	Description	
	Request instruction explanation	Gets the global acceleration.
	Response instruction format	=10.10 or e*:
	Response instruction explanation	Returns the acceleration value or deceleration value, or instruction returns an error.
	Remarks	-
268	Request instruction format	set_gripload Para1
	Request instruction example	set_gripload 8
	Request instruction explanation	Sets the grip load number. Para1: Grip load number, range 0 to 15
	Response instruction format	ok or e*:

No.	Description	
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
269	Request instruction format	get_gripload
	Request instruction explanation	Gets the grip load number.
	Response instruction format	=10 or e*:
	Response instruction explanation	Return the grip load number, or instruction returns an error.
	Remarks	-
270	Request instruction format	Set_DiagnsisItem Para1 Para2

No.	Description	
	Request instruction explanation	Sets the system diagnosis startup parameters. • Para1: Keyword ◦ string: Type ◦ ALL: All ◦ SYSTEM: System ◦ LOGIC: Logic ◦ TRAJECTORY: Trajectory • Para2: Check status ◦ 1: Checked ◦ 0: Unchecked
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
271	Request instruction format	Get_DiagnosisItem Para1
	Request instruction explanation	Reads the system diagnosis startup parameters. • Para1: Keyword ◦ string: Type ◦ ALL: All ◦ SYSTEM: System ◦ LOGIC: Logic ◦ TRAJECTORY: Trajectory • Para2: Check status

No.	Description	
	Response instruction format	=1 or e*:
	Response instruction explanation	• 1: Checked • 0: Unchecked
	Remarks	-
272	Request instruction format	Set_DiagnosisSave
	Request instruction explanation	Triggers the save operation for system diagnosis.
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-

No.	Description	
273	Request instruction format	Get_DiagnosisSave Para1
	Request instruction explanation	Reads the save progress status for system diagnosis. Para1: System diagnosis trigger save progress, range 0 to 100
	Response instruction format	=100 or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
274	Request instruction format	set_abortaccramp para1
	Request instruction explanation	Sets the acceleration jerk level for "Category 1 stop". Para1: Acceleration jerk level, range 100 to 500
	Response instruction format	ok or e*:

No.	Description	
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
275	Request instruction format	set_pauseaccramp para1
	Request instruction explanation	Sets the acceleration jerk level for "Category 2 stop". Para1: Acceleration jerk level, range 100 to 500
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
276	Request instruction format	set_pauseAbortAccMin para1

No.	Description	
	Request instruction explanation	Sets whether to disable the minimum acceleration limit for "Category 1 stop" and "Category 2 stop". Para1: 0 to 1 • 0: OFF • 1: ON
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
277	Request instruction format	Get_ConnectState
	Request instruction explanation	Gets the connection status of the host controller.
	Response instruction format	=1 or e*:

No.	Description	
	Response instruction explanation	• Returns the connection status of the host controller. ◦ 0: Disconnected ◦ 1: Connected • Instruction returns an error.
	Remarks	-
278	Request instruction format	Get_TCPSpeed
	Request instruction explanation	Gets the current motion speed of the robot end point.
	Response instruction format	=198.12 or e*:
	Response instruction explanation	Returns the current motion speed of the robot end point, or instruction returns an error.
	Remarks	-
279	Request instruction format	Get_JointSpeed Para1 Para2

No.	Description	
	Request instruction example	Get_JointSpeed 4 Robot
	Request instruction explanation	Gets the current joint speed.
	Response instruction format	=93.1 or e*:
	Response instruction explanation	Returns the current joint speed, or instruction returns an error.
	Remarks	-
280	Request instruction format	Get_SectionMovePercent
	Request instruction explanation	Gets the segment movement percentage.
	Response instruction format	=53 or e*:

No.	Description	
	Response instruction explanation	Returns the obtained percentage data of the segment movement, or instruction returns an error.
	Remarks	-
281	Request instruction format	Set_KeyTriggerMode [KeyTriggerMode]
	Request instruction explanation	Sets the key trigger mode of the hand-held teach pendant. KeyTriggerMode: key trigger mode • 0: key switch effective only for teach pendant permission • 1: key switch effective for all control permissions
	Response instruction format	ok or e*:
	Response instruction explanation	Setting completes, or instructions returns an error.
	Remarks	-
282	Request instruction format	Get_KeyTriggerMode

No.	Description	
	Request instruction explanation	Gets the key trigger mode of the hand-held teach pendant.
	Response instruction format	0/1 or e*:
	Response instruction explanation	• Returns the key trigger mode: ◦ 0: key switch effective only for teach pendant permission ◦ 1: key switch effective for all control permissions • Instruction returns an error.
	Remarks	-
283	Request instruction format	Set_SLVSMode Para1
	Request instruction example	Set_SLVSMode 2
	Request instruction explanation	Sets the self-learning vibration suppression mode. Para1: Vibration suppression sensitivity mode • 3: High • 2: Medium • 1: Low • 0: OFF

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
284	Request instruction format	Get_SLVSMode
	Request instruction explanation	Gets the self-learning vibration suppression mode.
	Response instruction format	=1 or e*:
	Response instruction explanation	• Returns the self-learning vibration suppression mode. ◦ 3: High ◦ 2: Medium ◦ 1: Low ◦ 0: OFF • Instruction returns an error.
	Remarks	-

No.	Description	
285	Request instruction format	SLData_Clear Para1
	Request instruction example	SLData_Clear 2
	Request instruction explanation	Clears the self-learning vibration suppression data. Para1: Clear mode • 1: Clear current data set • 2: Clear all
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
286	Request instruction format	MovJGPCSSL Para1 Para2 Para3 Para4 Para5 Para6 Para7 Para8

No.	Description	
	Request instruction example	MovJGPCSSL GP0 [0,50,0,0.000,0.000,0.000,0.000] 0 1 1 2 1 [1,0,1,1.000] Para1: GP0 Para2: [0,50,0,0.000,0.000,0.000,0.000] Para3: 0 Para4: 1 Para5: 1 Para6: 2 Para7: 1 Para8: [1,0,1,1.000]

	Request instruction explanation	Moves to the position point with specified number using Movj, and controls the corresponding I/O, supporting the tool/workobject setting and self-learning. • Para1: Target global position point number, range 0 to 9999 • Para2: Speed parameter, format: speed type, percentage speed property of whether the numeric speed is affected by the global percentage speed, TCP speed, TCP orientation speed, external axis linear speed, external axis rotational speed ◦ Speed type: ▪ 0: Percentage speed type ▪ 1: Numeric speed type ◦ Percentage speed: range 1 to 100 ◦ Property of whether the numeric speed is affected by the global percentage speed: ▪ 1: Not affected by the global percentage speed ▪ 0: Affected by the global percentage speed • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para4: Tool number, range 0 to 15 • Para5: Workobject number, range: 0 to 15 • Para6: Self-learning vibration suppression parameter ◦ 0: Disable ◦ 1: Enable

		◦ 2: Reset • Para7: Number of motion I/O groups, range 0 to 3 • Para8: Motion I/O control information, supporting 0 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter 1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
287	Request instruction format	MovLGPCSSL Para1 Para2 Para3 Para4 Para5 Para6 Para7 Para8
	Request instruction example	MovLGPCSSL GP0 [0,50,0,0.000,0.000,0.000,0.000] 0 1 1 2 1 [1,0,1,1.000] Para1: GP0 Para2: [0,50,0,0.000,0.000,0.000,0.000] Para3: 0 Para4: 1 Para5: 1 Para6: 2 Para7: 1 Para8: [1,0,1,1.000]

	Request instruction explanation	Moves to the position point with specified number using Movl, and controls the corresponding I/O, supporting the tool/workobject setting and self-learning. • Para1: Target global position point number, range 0 to 9999 • Para2: Speed parameter, format: speed type, percentage speed property of whether the numeric speed is affected by the global percentage speed, TCP speed, TCP orientation speed, external axis linear speed, external axis rotational speed ◦ Speed type: ▪ 0: Percentage speed type ▪ 1: Numeric speed type ◦ Percentage speed: range 1 to 100 ◦ Property of whether the numeric speed is affected by the global percentage speed: ▪ 1: Not affected by the global percentage speed ▪ 0: Affected by the global percentage speed • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para4: Tool number, range 0 to 15 • Para5: Workobject number, range: 0 to 15 • Para6: Self-learning vibration suppression parameter ◦ 0: Disable ◦ 1: Enable

		◦ 2: Reset • Para7: Number of motion I/O groups, range 0 to 3 • Para8: Motion I/O control information, supporting 0 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter 1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
288	Request instruction format	MovCGPCSSL Para1 Para2 Para3 Para4 Para5 Para6 Para7 Para8 Para9
	Request instruction example	MovCGPCSSL GP0 GP1 [0,50,0,0.000,0.000,0.000,0.000] 0 1 1 2 1 [1,0,1,1.000] Para1: GP0 Para2: GP1 Para3: [0,50,0,0.000,0.000,0.000,0.000] Para4: 0 Para5: 1 Para6: 1 Para7: 2 Para8: 1 Para9: [1,0,1,1.000]

	Request instruction explanation	Moves to the position point with specified number using Movc, and controls the corresponding I/O, supporting the tool/workobject setting and self-learning. • Para1: Global position point number of the arc intermediate point, range 0 to 9999 • Para2: Global position point number of the arc intermediate point, range 0 to 9999 • Para3: Speed parameter, format: speed type, percentage speed property of whether the numeric speed is affected by the global percentage speed, TCP speed, TCP orientation speed, external axis linear speed, external axis rotational speed ◦ Speed type: ▪ 0: Percentage speed type ▪ 1: Numeric speed type ◦ Percentage speed: range 1 to 100 ◦ Property of whether the numeric speed is affected by the global percentage speed: ▪ 1: Not affected by the global percentage speed ▪ 0: Affected by the global percentage speed • Para4: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para5: Tool number, range 0 to 15 • Para6: Workobject number, range: 0 to 15

		• Para7: Self-learning vibration suppression parameter ◦ 0: Disable ◦ 1: Enable ◦ 2: Reset • Para8: Number of motion I/O groups, range 0 to 3 • Para9: Motion I/O control information, supporting 0 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter 1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
289	Request instruction format	MovJRobPCSSL Para1 Para2 Para3 Para4 Para5 Para6 Para7 Para8
	Request instruction example	MovJRobPCSSL [370.000,50.000,765.000,-115.000,-75.000,-55.000 ; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] [0,50,0,0.000,0.000,0.000,0.000] 1 1 1 2 1 [1,0,1,1.000] Para1: [370.000,50.000,765.000,-115.000,-75.000,-55.000 ; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] Para2: [0,50,0,0.000,0.000,0.000,0.000] Para3: 1 Para4: 1 Para5: 1 Para6: 2 Para7: 1 Para8: [1,0,1,1.000]

	Request instruction explanation	Moves to the specified position point using Movj, and controls the corresponding I/O, supporting the tool/workobject setting and self-learning. • Para1: Target position point information • Para2: Speed parameter, format: speed type, percentage speed property of whether the numeric speed is affected by the global percentage speed, TCP speed, TCP orientation speed, external axis linear speed, external axis rotational speed ◦ Speed type: ▪ 0: Percentage speed type ▪ 1: Numeric speed type ◦ Percentage speed: range 1 to 100 ◦ Property of whether the numeric speed is affected by the global percentage speed: ▪ 1: Not affected by the global percentage speed ▪ 0: Affected by the global percentage speed • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para4: Tool number, range 1 to 15 • Para5: Workobject number, range: 1 to 15 • Para6: Self-learning vibration suppression parameter: ◦ 0: Disable ◦ 1: Enable ◦ 2: Reset

		• Para7: Number of motion I/O groups, range 0 to 3 • Para8: Motion I/O control information, supporting 0 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter 1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	• Valid only in data stream mode • Does not support numeric speed type
290	Request instruction format	MovLRobPCSSL Para1 Para2 Para3 Para4 Para5 Para6 Para7 Para8
	Request instruction example	MovLRobPCSSL [370.000,50.000,765.000,-115.000,-75.000,-55.000 ; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] [0,50,0,0.000,0.000,0.000,0.000] 1 1 1 2 1 [1,0,1,1.000] Para1: [370.000,50.000,765.000,-115.000,-75.000,-55.000 ; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] Para2: [0,50,0,0.000,0.000,0.000,0.000] Para3: 1 Para4: 1 Para5: 1 Para6: 2 Para7: 1 Para8: [1,0,1,1.000]

	Request instruction explanation	Moves to the specified position point using Movl, and controls the corresponding I/O, supporting the tool/workobject setting and self-learning. • Para1: Target position point information • Para2: Speed parameter, format: speed type, percentage speed property of whether the numeric speed is affected by the global percentage speed, TCP speed, TCP orientation speed, external axis linear speed, external axis rotational speed ◦ Speed type: ▪ 0: Percentage speed type ▪ 1: Numeric speed type ◦ Percentage speed: range 1 to 100 ◦ Property of whether the numeric speed is affected by the global percentage speed: ▪ 1: Not affected by the global percentage speed ▪ 0: Affected by the global percentage speed • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para4: Tool number, range 1 to 15 • Para5: Workobject number, range: 1 to 15 • Para6: Self-learning vibration suppression parameter: ◦ 0: Disable ◦ 1: Enable ◦ 2: Reset

		• Para7: Number of motion I/O groups, range 0 to 3 • Para8: Motion I/O control information, supporting 1 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter 1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
291	Request instruction format	MovCRobPCSSL Para1 Para2 Para3 Para4 Para5 Para6 Para7 Para8 Para9
	Request instruction example	MovCRobPCSSL [370.000,50.000,765.000,-115.000,-75.000,-55.000 ; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] [375.263,44.809,769.512,-118.306,-68.009,-47.077; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] [0,50,0,0.000,0.000,0.000,0.000] 1 1 1 2 1 [1,0,1,1.000] Para1: [370.000,50.000,765.000,-115.000,-75.000,-55.000 ; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] Para2: [375.263,44.809,769.512,-118.306,-68.009,-47.077; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] Para3: [0,50,0,0.000,0.000,0.000,0.000] Para4: 1 Para5: 1 Para6: 1 Para7: 2 Para8: 1 Para9: [1,0,1,1.000]

	Request instruction explanation	Moves to the position point with specified value using Movc, and controls the corresponding I/O. • Para1: Information of the arc intermediate point • Para2: Information of the target point position • Para3: Speed parameter, format: speed type, percentage speed property of whether the numeric speed is affected by the global percentage speed, TCP speed, TCP orientation speed, external axis linear speed, external axis rotational speed ◦ Speed type: ▪ 0: Percentage speed type ▪ 1: Numeric speed type ◦ Percentage speed: range 1 to 100 ◦ Property of whether the numeric speed is affected by the global percentage speed: ▪ 1: Not affected by the global percentage speed ▪ 0: Affected by the global percentage speed • Para4: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para5: Tool number, range 1 to 15 • Para6: Workobject number, range: 1 to 15 • Para7: Self-learning vibration suppression parameter: ◦ 0: Disable ◦ 1: Enable ◦ 2: Reset

		• Para8: Number of motion I/O groups, range 0 to 3 • Para9: Motion I/O control information, supporting 0 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter 1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
292	Request instruction format	JumpRobPCSSL Para1 Para2 Para3 Para4 Para5 Para6 Para7 Para8
	Request instruction example	JumpRobPCSSL [370.000,50.000,765.000,-115.000,-75.000,-55.000 ; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] [0,50,0,0.000,0.000,0.000,0.000] 1 1 1 2 1 [1,0,1,1.000] Para1: [370.000,50.000,765.000,-115.000,-75.000,-55.000 ; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] Para2: [0,50,0,0.000,0.000,0.000,0.000] Para3: 1 Para4: 1 Para5: 1 Para6: 2 Para7: 1 Para8: [1,0,1,1.000]

	Request instruction explanation	Moves to the position point with specified value using Jump, and controls the corresponding I/O. • Para1: Target position point information • Para2: Speed parameter, format: speed type, percentage speed property of whether the numeric speed is affected by the global percentage speed, TCP speed, TCP orientation speed, external axis linear speed, external axis rotational speed ◦ Speed type: ▪ 0: Percentage speed type ▪ 1: Numeric speed type ◦ Percentage speed: range 1 to 100 ◦ Property of whether the numeric speed is affected by the global percentage speed: ▪ 1: Not affected by the global percentage speed ▪ 0: Affected by the global percentage speed • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para4: Tool number, range 1 to 15 • Para5: Workobject number, range: 1 to 15 • Para6: Self-learning vibration suppression parameter: ◦ 0: Disable ◦ 1: Enable ◦ 2: Reset • Para7: Number of motion I/O groups, range 0 to 3

		• Para8: Motion I/O control information, supporting 0 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter 1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
293	Request instruction format	JumpLRobPCSSL Para1 Para2 Para3 Para4 Para5 Para6 Para7 Para8
	Request instruction example	JumpLRobPCSSL [370.000,50.000,765.000,-115.000,-75.000,-55.000 ; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] [0,50,0,0.000,0.000,0.000,0.000] 1 1 1 2 1 [1,0,1,1.000] Para1: [370.000,50.000,765.000,-115.000,-75.000,-55.000 ; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] Para2: [0,50,0,0.000,0.000,0.000,0.000] Para3: 1 Para4: 1 Para5: 1 Para6: 2 Para7: 1 Para8: [1,0,1,1.000]

	Request instruction explanation	Moves to the position point with specified value using JumpL, and controls the corresponding I/O. • Para1: Target position point information • Para2: Speed parameter, format: speed type, percentage speed property of whether the numeric speed is affected by the global percentage speed, TCP speed, TCP orientation speed, external axis linear speed, external axis rotational speed ◦ Speed type: ▪ 0: Percentage speed type ▪ 1: Numeric speed type ◦ Percentage speed: range 1 to 100 ◦ Property of whether the numeric speed is affected by the global percentage speed: ▪ 1: Not affected by the global percentage speed ▪ 0: Affected by the global percentage speed • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para4: Tool number, range 1 to 15 • Para5: Workobject number, range: 1 to 15 • Para6: Self-learning vibration suppression parameter: ◦ 0: Disable ◦ 1: Enable ◦ 2: Reset • Para7: Number of motion I/O groups, range 0 to 3

		• Para8: Motion I/O control information, supporting 0 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter 1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
294	Request instruction format	JumpGPCSSL Para1 Para2 Para3 Para4 Para5 Para6 Para7 Para8
	Request instruction example	JumpGPCSSL GP0 [0,50,0,0.000,0.000,0.000,0.000] 0 1 1 2 1 [1,0,1,1.000] Para1: GP0 Para2: [0,50,0,0.000,0.000,0.000,0.000] Para3: 0 Para4: 1 Para5: 1 Para6: 2 Para7: 1 Para8: [1,0,1,1.000]

	Request instruction explanation	Moves to the global position point with specified number using Jump, and controls the corresponding I/O, supporting the tool/workobject setting and self-learning. • Para1: Target global position point number, range 0 to 9999 • Para2: Speed parameter, format: speed type, percentage speed property of whether the numeric speed is affected by the global percentage speed, TCP speed, TCP orientation speed, external axis linear speed, external axis rotational speed ◦ Speed type: ▪ 0: Percentage speed type ▪ 1: Numeric speed type ◦ Percentage speed: range 1 to 100 ◦ Property of whether the numeric speed is affected by the global percentage speed: ▪ 1: Not affected by the global percentage speed ▪ 0: Affected by the global percentage speed • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para4: Tool number, range 0 to 15 • Para5: Workobject number, range: 0 to 15 • Para6: Self-learning vibration suppression parameter: ◦ 0: Disable ◦ 1: Enable

		◦ 2: Reset • Para7: Number of motion I/O groups, range 0 to 3 • Para8: Motion I/O control information, supporting 0 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter 1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
295	Request instruction format	JumpLGPCSSL Para1 Para2 Para3 Para4 Para5 Para6 Para7 Para8
	Request instruction example	JumpLGPCSSL GP0 [0,50,0,0.000,0.000,0.000,0.000] 0 1 1 2 1 [1,0,1,1.000] Para1: GP0 Para2: [0,50,0,0.000,0.000,0.000,0.000] Para3: 0 Para4: 1 Para5: 1 Para6: 2 Para7: 1 Para8: [1,0,1,1.000]

	Request instruction explanation	Moves to the global position point with specified number using JumpL, and controls the corresponding I/O, supporting the tool/workobject setting and self-learning. • Para1: Target global position point number, range 0 to 9999 • Para2: Speed parameter, format: speed type, percentage speed property of whether the numeric speed is affected by the global percentage speed, TCP speed, TCP orientation speed, external axis linear speed, external axis rotational speed ◦ Speed type: ▪ 0: Percentage speed type ▪ 1: Numeric speed type ◦ Percentage speed: range 1 to 100 ◦ Property of whether the numeric speed is affected by the global percentage speed: ▪ 1: Not affected by the global percentage speed ▪ 0: Affected by the global percentage speed • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para4: Tool number, range 0 to 15 • Para5: Workobject number, range: 0 to 15 • Para6: Self-learning vibration suppression parameter: ◦ 0: Disable ◦ 1: Enable

		◦ 2: Reset • Para7: Number of motion I/O groups, range 0 to 3 • Para8: Motion I/O control information, supporting 0 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter 1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Valid only in data stream mode
296	Request instruction format	MovJAbsRobJPCSSL Para1 Para2 Para3 Para4 Para5 Para6 Para7 Para8
	Request instruction example	MovJAbsRobJPCSSL [370.000,50.000,765.000,-115.000,-75.000,-55.000 ; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] [0,50,0,0.000,0.000,0.000,0.000] 0 1 1 2 1 [1,0,1,1.000] Para1: [370.000,50.000,765.000,-115.000,-75.000,-55.000 ; 0,0,0,1; 1.000,2.000,3.000,4.000,5.000,6.000] Para2: [0,50,0,0.000,0.000,0.000,0.000] Para3: 0 Para4: 1 Para5: 1 Para6: 2 Para7: 1 Para8: [1,0,1,1.000]

	Request instruction explanation	Moves to the specified joint point using Movj, and controls the corresponding I/O, supporting the tool/workobject setting and self-learning. • Para1: Target position point information • Para2: Speed parameter, format: speed type, percentage speed property of whether the numeric speed is affected by the global percentage speed, TCP speed, TCP orientation speed, external axis linear speed, external axis rotational speed ◦ Speed type: ▪ 0: Percentage speed type ▪ 1: Numeric speed type ◦ Percentage speed: range 1 to 100 ◦ Property of whether the numeric speed is affected by the global percentage speed: ▪ 1: Not affected by the global percentage speed ▪ 0: Affected by the global percentage speed • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para4: Tool number, range 1 to 15 • Para5: Workobject number, range: 1 to 15 • Para6: Self-learning vibration suppression parameter: ◦ 0: Disable ◦ 1: Enable ◦ 2: Reset

		• Para7: Number of motion I/O groups, range 0 to 3 • Para8: Motion I/O control information, supporting 0 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter 1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	• Valid only in data stream mode • Does not support numeric speed type
297	Request instruction format	MovJAbsRobIndexJPCSSL Para1 Para2 Para3 Para4 Para5 Para6 Para7 Para8
	Request instruction example	MovJAbsRobIndexJPCSSL GJP1 [0,50,0,0.000,0.000,0.000,0000] 0 1 1 2 1 [1,0,1,1.000] Para1: GJP1 Para2: [0,50,0,0.000,0.000,0.000,0000] Para3: 0 Para4: 1 Para5: 1 Para6: 2 Para7: 1 Para8: [1,0,1,1.000]

	Request instruction explanation	Moves to the joint point with specified number using Movj, and controls the corresponding I/O, supporting the tool/workobject setting and self-learning. • Para1: Target position point information • Para2: Speed parameter, format: speed type, percentage speed property of whether the numeric speed is affected by the global percentage speed, TCP speed, TCP orientation speed, external axis linear speed, external axis rotational speed ◦ Speed type: ▪ 0: Percentage speed type ▪ 1: Numeric speed type ◦ Percentage speed: range 1 to 100 ◦ Property of whether the numeric speed is affected by the global percentage speed: ▪ 1: Not affected by the global percentage speed ▪ 0: Affected by the global percentage speed • Para3: Interpolation accuracy, range -2 to 200 & 1000 to 1200 ◦ -2: No transition ◦ -1: Fine instruction, indicating precise arrival. The robot is considered to achieve precise arrival when it enters and stays within the set arrival error range. ◦ 0 to 200: Z[0] to Z[200]: The robot does not stop at the target point, but smooths through the set transition area. Transition length = Transition level x Transition unit length ◦ 1000 to 1200: ZR[0] to ZR[200], relative to the transition level, indicating the percentage of the maximum allowable transition length. • Para4: Tool number, range 1 to 15 • Para5: Workobject number, range: 1 to 15 • Para6: Self-learning vibration suppression parameter: ◦ 0: Disable ◦ 1: Enable

		◦ 2: Reset • Para7: Number of motion I/O groups, range 0 to 3 • Para8: Motion I/O control information, supporting 0 to 3 groups, format: [I/O number 1, I/O output value 1, I/O type 1, I/O parameter 1; I/O number 2, I/O output value 2, I/O type 2, I/O parameter 2...] ◦ I/O number: The I/O number to be controlled ◦ I/O output value: The output value of the I/O (0: OFF, 1: ON) ◦ I/O type: Set I/O type during motion (0: Time, 1: Path Percentage, 2: Distance) ◦ I/O parameters: When the type is Time, Value≥0 indicates that signals are output Value seconds after motion starts. Value<0 indicates that signals are output Value seconds before arrival at a motion point; when the type is Path Percentage, Value≥0 indicates that signals are output when the robot reaches Value% of the path after motion starts, Value<0 indicates that signals are output when the robot reaches Value% of the path before the motion point; when the type is Distance, Value≥0 indicates that signals are output after the robot moves by Value mm from the start point, Value<0 indicates that signals are output before the robot moves to Value mm from the end point.

No.	Description	
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	• Valid only in data stream mode • Does not support numeric speed type
298	Request instruction format	SetAccuracyMode Para1
	Request instruction example	SetAccuracyMode 0
	Request instruction explanation	Sets the trajectory accuracy mode. Para1: Trajectory accuracy mode, range 0 to 2 • 0: Default mode • 1: Medium accuracy mode • 2: High accuracy mode
	Response instruction format	ok or e*:

No.	Description	
	Response instruction explanation	Instruction completes; Instruction returns an error.
	Remarks	Effective only in data stream mode
299	Request instruction format	SetErrPosByAxis Para1 Para2
	Request instruction example	SetErrPosByAxis 1 1000
	Request instruction explanation	Sets the arrival position error. • Para1: Axis number, range 1 to 4 for four-axis robots; range 1 to 6 for 6-axis robots • Para2: Arrival position error, range 500 to 200000
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Effective only in data stream mode

No.	Description	
300	Request instruction format	GetErrPosByAxis Para1
	Request instruction example	GetErrPosByAxis 1
	Request instruction explanation	Gets the arrival position error. Para1: Axis number, range 1 to 4 for four-axis robots; range 1 to 6 for 6-axis robots
	Response instruction format	=50 or e*:
	Response instruction explanation	Returns the arrival position error for the axis number corresponding to Para1, or instruction returns an error.
	Remarks	-
301	Request instruction format	SetBandTimeByAxis Para1 Para2
	Request instruction example	SetBandTimeByAxis 1 1000

No.	Description	
	Request instruction explanation	Sets the arrival time threshold. • Para1: Axis number, range 1 to 4 for four-axis robots; range 1 to 6 for 6-axis robots • Para2: Arrival time threshold, range 500 to 200000
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Effective only in data stream mode
302	Request instruction format	GetBandTimeByAxis Para1
	Request instruction example	GetBandTimeByAxis 1
	Request instruction explanation	Gets the arrival time threshold. Para1: Axis number, range 1 to 4 for four-axis robots; range 1 to 6 for 6-axis robots
	Response instruction format	=50 or e*:

No.	Description	
	Response instruction explanation	Returns the arrival time threshold for the axis number corresponding to Para1. Instruction returns an error.
	Remarks	-
303	Request instruction format	SetSingAreaHandle Para1
	Request instruction example	SetSingAreaHandle 1
	Request instruction explanation	Sets the singularity crossing switch. Para1: Singularity crossing function status: • 0: OFF • 1: ON
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	Effective only in data stream mode

No.	Description	
304	Request instruction format	GetSingAreaHandle
	Request instruction explanation	Gets the singularity crossing switch status.
	Response instruction format	=1 or e*:
	Response instruction explanation	• Returns the singularity crossing switch status. ◦ 0: OFF ◦ 1: ON • Instruction returns an error.
	Remarks	-
305	Request instruction format	Get_ModuleSubPartNum [ModuleType]
	Request instruction explanation	Gets the number of component sub-parts based on the component type. ModuleType: Component type: • 0: Encoder battery • 1: Ball screw grease • 2: Ball screw wear • 3: Synchronous belt • 4: Reducer

No.	Description	
	Response instruction format	=6 or =e*:
	Response instruction explanation	• Returns the number of component sub-parts, ranging from 0 to 6. • Instruction returns an error.
	Remarks	-
306	Request instruction format	Get_MaintainInfo [ModuleType] [ModuleIndex]
	Request instruction explanation	Gets the maintenance information of component sub-parts based on the component type and component number. • ModuleType: Component type: ◦ 0: Encoder battery ◦ 1: Ball screw grease ◦ 2: Ball screw wear ◦ 3: Synchronous belt ◦ 4: Reducer • ModuleIndex: Component sub-part number, range 1 to 6, indicating the index of a sub-part
	Response instruction format	=4,6,1,4320,13680,18000,20250210=e*

No.	Description	
	Response instruction explanation	• Returns the component maintenance information: ◦ Component type: ▪ 0: Encoder battery ▪ 1: Ball screw grease ▪ 2: Ball screw wear ▪ 3: Synchronous belt ▪ 4: Reducer ◦ Number of component sub-parts: 1 to 6 ◦ Maintenance status of component sub-part: ▪ 0: Pending maintenance ▪ 1: Normal ◦ Remaining usage time of component sub-part: [0, maintenance threshold] ◦ Used time of component sub-part: [0, maintenance threshold] ◦ Maintenance threshold of component sub-part: [0, 50000] or [0, 100] ◦ Last maintenance time of component sub-part: February 10, 2025 • Instruction returns an error.
	Remarks	When the component type is ball screw, ModuleIndex can only be 1 (there can only be one ball screw).
307	Request instruction format	Clear_MaintainInfo
	Request instruction explanation	Clears all maintenance information.

No.	Description	
	Response instruction format	=ok or =e*
	Response instruction explanation	Clearing succeeded, or instruction returns an error.
	Remarks	Maintenance operations can only be performed in a non-motion state under Manufacturer mode, Emergency Stop mode, and Manual mode; otherwise, e29 is returned.
308	Request instruction format	Set_MaintainThreshold [ModuleType] [ModuleIndex] [ThresholdValue]
	Request instruction explanation	Sets the maintenance threshold for the component sub-parts. • ModuleType: Component type: ◦ 0: Encoder battery ◦ 1: Ball screw grease ◦ 2: Ball screw wear ◦ 3: Synchronous belt ◦ 4: Reducer grease • ModuleIndex: Component sub-part number, range 1 to 6, indicating the index of a sub-part • ThresholdValue: Threshold. The range of the threshold parameter varies depending on the component type. • Synchronous belt: [0,10000] hours • Encoder battery: [0, 50000] hours • Reducer grease: [0, 50000] hours • Ball screw grease: [0, 50000] hours • Ball screw wear: [0, 100]%

No.	Description	
	Response instruction format	=ok or =e*
	Response instruction explanation	Setting completes, or instructions returns an error.
	Remarks	• Maintenance operations can only be performed in a non-motion state under Manufacturer mode, Emergency Stop mode, and Manual mode; otherwise, e29 is returned. • When the component type is ball screw, ModuleIndex can only be 1 (there can only be one ball screw). • Before calling this API, you can first use IMC100_Get_ModuleSubPartNum to get the number of component sub-parts, then determine the sub-part number to be set. This number is not the axis number.
309	Request instruction format	Update_MaintainInfo [ModuleType] [ModuleIndex]
	Request instruction explanation	Updates the maintenance information of the component sub-part after maintenance is completed. • ModuleType: Component type: ◦ 0: Encoder battery ◦ 1: Ball screw grease ◦ 2: Ball screw wear ◦ 3: Synchronous belt ◦ 4: Reducer grease • ModuleIndex: Component sub-part number, range 1 to 6, indicating the index of a sub-part

No.	Description	
	Response instruction format	=ok or =e*
	Response instruction explanation	Update succeeded, or instruction returns an error.
	Remarks	• Maintenance operations can only be performed in a non-motion state under Manufacturer mode, Emergency Stop mode, and Manual mode; otherwise, e29 is returned. • When the component type is ball screw, ModuleIndex can only be 1 (there can only be one ball screw). • Before calling this API, you can first use IMC100_Get_ModuleSubPartNum to get the number of component sub-parts, then determine the sub-part number to be set. This number is not the axis number.
310	Request instruction format	set_rtdecfgload
	Request instruction explanation	Loads the real-time data exchange configuration.
	Response instruction format	=1 or e*:

No.	Description	
	Response instruction explanation	0: Success Non-zero: Parameter loading exception
	Remarks	-
311	Request instruction format	set_rtdeChannelStatus 5 0
	Request instruction explanation	• Index: Channel number, range 0 to 15 • Status: Channel activation status; 0/1
	Response instruction format	=0 or =e*
	Response instruction explanation	0: Success Non-0: Parameter setting exception
	Remarks	Only one channel can be activated at a time; when activating a channel, the other channel will be deactivated.
312	Request instruction format	get_offsetref_robp 300.0,0.0,540.0,-50.0,0.0,-100.0;0,0,0,1;0.0,0.0,0.0,0.0,0.0,0.0,0.0 10.0,1.0,1.0,1.0,1.0,1.0;0,0,1 0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0

No.	Description	
	Request instruction explanation	Gets the translation of a target point relative to a specified joint angle position in a Cartesian reference frame. 300.0,0.0,540.0,-50.0,0.0,-100.0;0,0,0,1;0.0,0.0,0.0,0.0,0.0,0.0: Reference point 10.0,1.0,1.0,1.0,1.0,1.0: Offset point 0,0,1: Tool number, workobject number, Whether joint reference point included (0 for no, 1 for yes) 0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0: Reference joint angle
	Response instruction format	Returns the converted point or e*.
	Response instruction explanation	Instruction completes; returns the converted coordinates. Instruction returns an error.
	Remarks	-
313	Request instruction format	get_offsetref_robp 300.0,0.0,540.0,-50.0,0.0,-100.0;0,0,0,1;0.0,0.0,0.0,0.0,0.0,0.0 10.0,1.0,1.0,1.0,1.0,1.0;0,0,0
	Request instruction explanation	Gets the translation of a target point relative to a specified joint angle position in a Cartesian reference frame. 300.0,0.0,540.0,-50.0,0.0,-100.0;0,0,0,1;0.0,0.0,0.0,0.0,0.0,0.0: Reference point 10.0,1.0,1.0,1.0,1.0,1.0: Offset point 0,0,0: Tool number, workobject number, Whether joint reference point included (0 for no, 1 for yes)

No.	Description	
	Response instruction format	Returns the point position or e*.
	Response instruction explanation	Instruction completes; returns the converted coordinates. Instruction returns an error.
	Remarks	-
314	Request instruction format	get_offsetmode
	Request instruction explanation	• 0: Default offset solution selection mode • 1: Offset solution selection mode for S03.22 to V4R24C3 versions
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes; returns the offset solution selection mode. Instruction returns an error.
	Remarks	-

No.	Description	
315	Request instruction format	set_offsetmode 0
	Request instruction explanation	• 0: Default offset solution selection mode • 1: Offset solution selection mode for S03.22 to V4R24C3 versions
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes; returns the converted coordinates. Instruction returns an error.
	Remarks	-
316	Request instruction format	Set_SupplementaryStrParamComp Para1 Para2 Para3 Para4
	Request instruction example	Set_SupplementaryStrParamComp 0.1 0.1 1.1 2.1 Para1: 0.1 Para2: 0.1 Para3: 1.1 Para4: 2.1

No.	Description	
	Request instruction explanation	Sets the compensation parameters. Para1: Internal compensation parameter 1 Para2: Internal compensation parameter 2 Para3: Internal compensation parameter 3 Para4: Internal compensation parameter 4
	Response instruction format	ok or e*:
	Response instruction explanation	Instruction completes, or returns an error.
	Remarks	-
317	Request instruction format	Get_SupplementaryStrParamComp
	Request instruction explanation	Queries the compensation parameters.
	Response instruction format	ok or e*:

No.	Description	
	Response instruction explanation	Instruction completes; returns 4 compensation parameters. Instruction returns an error.
	Remarks	-

8.3 Appendix 3: API Fault Description

No.	Description	
1	Function return value	0
	Fault print information	-
	Description	(Instruction normal)
	Solution	-
2	Function return value	-1
	Fault print information	e1: syntax error
	Description	Instruction syntax error
	Solution	Caused by a dynamic link library version issue, contact technical support (or the parameters of the point position instruction are not initialized).
3	Function return value	-2
	Fault print information	e2: number of parameter unmatched
	Description	Instruction parameter count mismatch
	Solution	Caused by a dynamic link library version issue, contact technical support.
4	Function return value	-3
	Fault print information	e3: parameter value illegal
	Description	Instruction parameter value illegal
	Solution	

No.	Description	
		Reassign function parameters according to the function description.
5	Function return value	-4
	Fault print information	e4: not allowed in current mode
	Description	The instruction is not allowed in the current mode.
	Solution	Switch to Manual, Auto, or data stream mode as needed, then recall the function.
6	Function return value	-5
	Fault print information	e5: not allowed when robot is running
	Description	The instruction is not allowed while the robot is running.
	Solution	Recall after the robot has stopped.
7	Function return value	-6
	Fault print information	e6: system in emergency
	Description	The system is in an emergency stop state.
	Solution	Recall after the emergency stop is released.
8	Function return value	-7
	Fault print information	e7: system fault
	Description	The system is faulty.
	Solution	Recall after the system faults are cleared.
9	Function return value	-8
	Fault print information	e8: motion mode closed
	Description	The data stream mode is OFF.
	Solution	Recall after the data stream mode is ON.
10	Function return value	-9
	Fault print information	e9: motor off
	Description	The motor is not enabled.

No.	Description	
	Solution	Recall after the motor is enabled.
11	Function return value	-10
	Fault print information	e10: rsv
	Description	Reserved
	Solution	Reserved
12	Function return value	-11
	Fault print information	e11: instruction unfinished
	Description	The motion instruction buffer still has uncompleted instruction.
	Solution	Recall when the motion instruction buffer is idle.
13	Function return value	-12
	Fault print information	e12: output unavailable
	Description	User does not have permission to control the output port.
	Solution	Recall after obtaining permission for this port, or control other available ports.
14	Function return value	-13
	Fault print information	e13: read AD failed
	Description	Reading AD channel fails.
	Solution	Recall after checking the module is correctly configured.
15	Function return value	-14
	Fault print information	e14: write DA failed
	Description	Failed to write to the AD channel.
	Solution	Recall after checking the module is correctly configured.
16	Function return value	-15
	Fault print information	e15: write DO failed

No.	Description	
	Description	Failed to write to the DO channel.
	Solution	Recall after checking the module is correctly configured.
17	Function return value	-16
	Fault print information	e16: command invalid
	Description	Pause instruction is invalid.
	Solution	Do not call the pause function when the program is in the stopped state.
18	Function return value	-17
	Fault print information	e17: not allowed in current coordinate
	Description	The instruction is not allowed in the current coordinate frame.
	Solution	Recall after switching the frame.
19	Function return value	-18
	Fault print information	e18: mode conflict
	Description	Motion mode conflict
	Solution	Recall after switching the data stream mode OFF.
20	Function return value	-19
	Fault print information	e19: indAxis mode err
	Description	Independent axis mode error
	Solution	Recall after switching to independent axis mode.
21	Function return value	-20
	Fault print information	e20: program non-existent
	Description	The specified program path does not exist.
	Solution	Re-enter the path.
22	Function return value	-21
	Fault print information	e21: point non-existent

No.	Description	
	Description	Position point P or offset LPR does not exist.
	Solution	Recall after checking that the point is correct and reopening the project.
23	Function return value	-22
	Fault print information	e22: calc error
	Description	Internal calculation error
	Solution	Recall after checking that the parameters passed in are correct.
24	Function return value	-23
	Fault print information	e23: rsv
	Description	Reserved
	Solution	Reserved
25	Function return value	-24
	Fault print information	e24: without permit
	Description	The current Ethernet device does not have the control permission.
	Solution	Recall after switching the control authority and obtaining the control permission.
26	Function return value	-25
	Fault print information	e25: ETH without authorization
	Description	The current Ethernet device does not have the control authority.
	Solution	Recall after switching control authority.
27	Function return value	-26
	Fault print information	e26: rsv
	Description	Reserved
	Solution	Reserved
28	Function return value	-27

No.	Description	
	Fault print information	e27: low user grade
	Description	The current user permission level is insufficient.
	Solution	Log in with a higher user level and recall the function.
29	Function return value	-28
	Fault print information	e28: permit occupied
	Description	Control permission has been requested by another Ethernet device.
	Solution	Recall after forcibly obtaining the control permission.
30	Function return value	-29
	Fault print information	e29: internal fault
	Description	Internal call error
	Solution	Call API instruction according to the specifications.
31	Function return value	-30
	Fault print information	e30: Modbus unavailable
	Description	Modbus slave is not configured.
	Solution	Recall after configuring Modbus slave.
32	Function return value	-31
	Fault print information	e31: condition unfulfilled to write
	Description	System parameters cannot be set at present.
	Solution	Recall after the robot stops or the configuration file is saved
33	Function return value	-32
	Fault print information	e32: can't write para when no project
	Description	Parameter writing is not allowed when no project is selected.

No.	Description	
	Solution	Select the project before writing project parameters.
34	Function return value	-33
	Fault print information	e33: pallet para err
	Description	Pallet parameter error
	Solution	Check whether the pallet has been created and verify if its parameters are correct. Recreate the pallet or reconfigure the pallet parameters as needed.
35	Function return value	-34
	Fault print information	e34: can't read or write points when Loading Points
	Description	Points cannot be read from or written to while loading points.
	Solution	Wait for the point loading to complete.
36	Function return value	-35
	Fault print information	e35: the interface is not supported with virtual controller
	Description	This API does not support virtual controller.
	Solution	Not supported for virtual controller
37	Function return value	-36
	Fault print information	e36: Ownership of key control, does not support remote switching of manual mode
	Description	The key trigger mode of the teach pendant is set to "Effective for all control permissions". Remote mode switching is not supported.
	Solution	Switch the key trigger mode to "Effective only for teach pendant permission"
38	Function return value	-253
	Fault print information	-
	Description	Internal calculation error

No.	Description	
	Solution	Caused by a dynamic link library version issue, contact technical support.
39	Function return value	-254
	Fault print information	-
	Description	Function input parameter incorrect
	Solution	Reassign function parameters according to the function description.
40	Function return value	-255
	Fault print information	-
	Description	Network communication error
	Solution	Recall after checking that the network connection is correct.

Service and Support

Should you encounter a safety accident during the use or operation of the product, or face challenges in operating and maintaining the equipment, which remain unresolved after the relevant documentation is consulted, we provide multiple channels to ensure prompt resolution:

- Channel #1: Contact service@inovance.com.
- Channel #2: Visit https://www.inovance.com/global to access document downloads, after-sales support, spare parts ordering, repair applications, and authenticity verification services.
- Channel #3: Download My Inovance app (https://zshc-eu.inovance.com/download-pc/) where you can access products info and documentation, and query product parameters.

We are committed to providing you with quick and professional technical support, and we look forward to your satisfaction and trust.

19012860A00

Copyright © Shenzhen Inovance Technology Co., Ltd.

Shenzhen Inovance Technology Co., Ltd.

www.inovance.com

Add.: Inovance Headquarters Tower, High-tech Industrial Park,
Guanlan Street, Longhua New District,
Shenzhen 518000, P.R. China

Tel: (0755) 2979 9595 Fax: (0755) 2961 9897

Suzhou Inovance Technology Co., Ltd.

www.inovance.com

Add.: No. 52, Tian E Dang Road, Wuzhong District, 215104,
Suzhou City, Jiangsu Province, P.R. China

Tel: (0512) 6637 6666 Fax: (0512) 6285 6720