1 Contents

Preface....4
1 Basic Concepts....4
1.1 Robot System Overview....4
1.2 Operating Environment....5
1.3 Classification of Robots....5
1.4 Coordinate System....6
1.4.1 Joint Coordinate System....7
1.4.2 Base Coordinate System....7
1.4.3 Tool Coordinate System....8
1.4.4 User Coordinate System....8
1.4.5 Other Coordinate Systems....9
1.5 Position Variables....9
1.5.1 Overview....9
1.5.2 Storage Format of Position Variable....10
1.5.3 Attributes of Position Variables....10
1.5.4 Position Variables for Special Processes....15
1.6 Offset....15
1.6.1 Joint Offset....15
1.6.2 Offset Along the Current Tool Pose....16
1.6.3 Offset in a Cartesian Frame....16
1.7 Interpolation and Transition....16
1.8 Singular Position....17
1.9 Motion Range and Interference Area....18
2 Getting Started....19
2.1 Operating Process....19
2.2 Power-On and Connection....19
2.3 User Login....21
2.4 Status Check....24
2.5 Robot Manipulation....26
3 Programming and Running....28
3.1 Basic Features on Main Interface....28
3.2 Project Manager....34
3.3 Editing the Project....41
3.4 Debugging....58
3.5 Play Mode....63
3.6 Viewing Project Under Other Controls....65
3.7 Example: Programming and Running a Project....65
3.8 Shortcut Keys....71
4 Settings....71
4.1 Robot Settings....72
4.2 Zero Point Settings....72
4.2.1 Absolute Zero Point....72

4.2.2 Work Origin 73
4.2.3 Zeroing..74
4.3 Installation Parameter Settings..80
4.4 Motion Settings..81
4.5 Peripheral Settings..94
4.5.1 Bus Switch..94
4.5.2 I/O Mapping..97
4.5.3 Project ID Settings..100
4.5.4 IRLink Settings..102
4.6 System Settings..106
4.7 Extended Functions..126
5 Monitoring..126
5.1 Basic Operations..126
5.2 Global Variable Monitoring..128
5.3 I/O Monitoring..131
5.3.1 Introduction of Robot Bus Address..131
5.3.2 How to Use I/O Monitoring..132
5.4 Communication state..139
5.4.1 Device Connection..139
5.4.2 Bus Monitoring..140
5.5 Servo State..142
5.6 Log..143
5.7 Version..143
5.8 Current Protection..144
6 Process Application..144
6.1 Tracking Process..144
6.1.1 Overview..144
6.1.2 Hardware Configuration..146
6.1.3 Coordinate System Setting..147
6.1.4 Parameter Setting..149
6.1.5 Tracking Instructions..158
6.1.6 Application Cases..162
6.2 Vision Calibration..164
6.2.1 Overview..164
6.2.2 Vision Calibration of SCARA Robots..165
6.2.3 Vision Calibration of 6-Axis Robots..166
6.2.4 Eye-to-Hand Overlook Calibration..168
6.2.5 Eye-to-Hand Look-up Calibration..174
6.2.6 Eye-on-Hand J2 Calibration..176
6.2.7 Eye-on-Hand J4 Calibration..178
6.2.8 Eye-on-Hand J5 Calibration..179
6.2.9 Eye-on-Hand J6 Calibration..181
6.2.10 Calibration Result Verification..182
7 Others..184

7.1 TCP multi-port connectivity 184
7.2 Permission Management 184
7.2.1 Robot Control Permissions 184
7.2.2 IRLink Configuration Permissions 186
7.2.3 I/O Control Permissions 186
7.2.4 Modbus Configuration Permissions 187
7.3 Multitasking 187
7.3.1 Task Description 187
7.3.2 Use of Multitasking 189
7.3.3 Multitasking Alarms 191
7.4 Flying Trigger 191
7.5 Teach Pendant Synchronization 193
7.6 Retentive Memory 193
7.7 Safety door 194
7.8 Current Protection 196
7.9 API 200
7.9.1 Description of API Call 200
7.9.2 Typical Application Cases 204
7.10 Pose Calibration 210
7.11 Optimal Trajectory 211
7.11.1 Description 211
7.11.2 Commissioning Procedure 211
7.11.3 Example 212
7.12 Self-Learning Vibration Suppression 212
7.12.1 Description 212
7.12.2 Related Instructions 213
7.12.3 Backup, loading, recovery, and clearing of self-learning data 216
7.12.4 Example 221
7.13 Releasing Dynamic Brake 222
Appendix 1: Robot Alarms and Handling Method 223
Appendix 2: API Instructions and Connection Fault Table API Instructions 293
(1) API Instructions 293
(2) API Connection Failure Table 327
Appendix 3: Modbus Slave Address Table 331
Appendix 4: Servo Commissioning 356
Appendix 5: Simple Calculation of Load Parameters 370

Teach Pendant User Guide

Preface

This guide is intended to help readers learn how to program the Inovance robots on the teach pendant.

Document version	Release date	Teach pendant version	Controller version	InoRobotLab version
S03.21R	November 07, 2021	S03.21R	S03.21R	S03.21R

1 Basic Concepts

1.1 Robot System Overview

The robot system consists of human-machine interface (HMI) software, robot controller (including teach pendant), robot manipulator and secondary development software.

Components	Description
HMI software	■ Teaching software: InoTeachPad (can be installed on 7-inch handheld teach pendant IRTP80, as well PC) ■ PC platform software: InoRobotLab
Robot controller (including teach pendant)	Configures, programs, and monitors operation of the robot. You can control the motion of the robot through the HMI software or other control devices.
Manipulator	The actuator of the robot system, the motion of which is controlled through the controller.

This document describes operations performed on the teach pendant.

1.2 Operating Environment

Handheld teach pendent:

The teaching software is embedded in teach pendant IRTP80 at factory. No additional operating environment configuration is needed.

PC-based teach pendant:

The teaching software is installation-free on the PC. The PC must meet the following requirements:

Operating system: Win XP and above

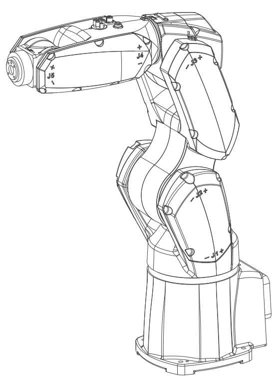
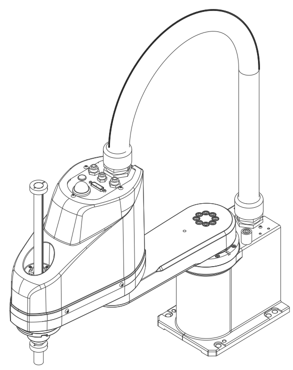
Memory: At least 128 MB

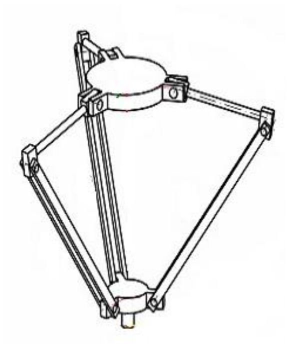
Resolution: 1024*768 and above

Runtime environment: Visual C++ Redistributable Package (Microsoft Visual C++ Redistributable Package.exe should be installed if the runtime environment is missing.)

1.3 Classification of Robots

Robots can be classified based on the number of axes, series/parallel characteristics, etc. The following table lists several common robots:

Name	Type	Number of axes	Serial/Parallel	Features	Application
6-axis robot		6	Serial	Extremely flexible and suitable for working with virtually any trajectory or angle.	Loading, painting, measuring, arc welding, spot welding, packaging, assembly, forging, casting, etc.
SCARA robot		4	Serial	Lightweight structure and fast response	Mechanical assembly, material distribution and dispensing, robot assembly, labeling, placement, dispensing, etc. in 3C industry.

Delta robot		4	Parallel	High precision, fast speed	Handling, sorting, etc. of pharmaceutical and food products.

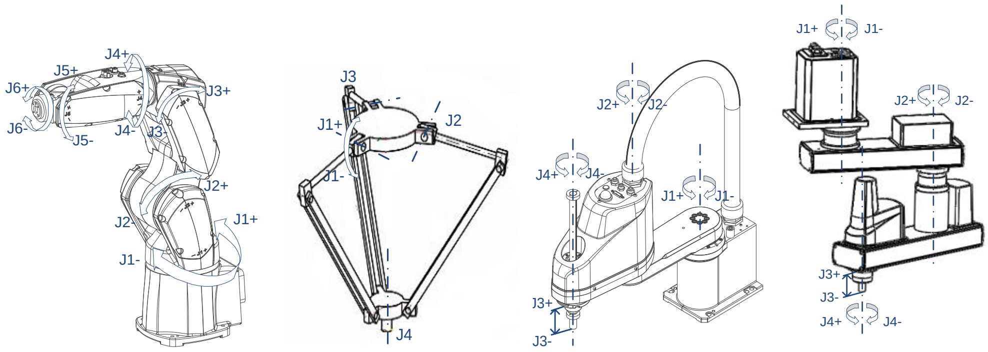
1.4 Coordinate System

Seven coordinate systems are available in Inovance robot system.

Coordinate system number	Coordinate system name	Definition
1	Joint coordinate system	A coordinate system defined at each joint of the robot, which is a direct description of the motion of the robot joints.
2	Base coordinate system	A coordinate system defined on the base of the robot, often used as a reference for motion.
3	Tool coordinate system	A coordinate system defined on the tool, which can be customized.
4	User coordinate system	A coordinate system defined on the workpiece, which can be customized.
5	Fixed camera FOV coordinate system	A coordinate system dedicated to the vision process. It is established at reference point of the fixed mobile camera's field of view, and is used to describe the position points in the camera's field of view (X, Y, θ).
6	Mobile camera FOV coordinate system	A coordinate system dedicated to the vision process. It is established at reference point of the mobile camera's field of view, and is used to describe the position points in the camera's field of view (X, Y, θ).
7	Object coordinate system on the conveyor belt	A coordinate system dedicated to the tracking process. Points (X, Y, Z, A, B, C) relative to the object coordinate system on the conveyor belt are defined in this coordinate system.

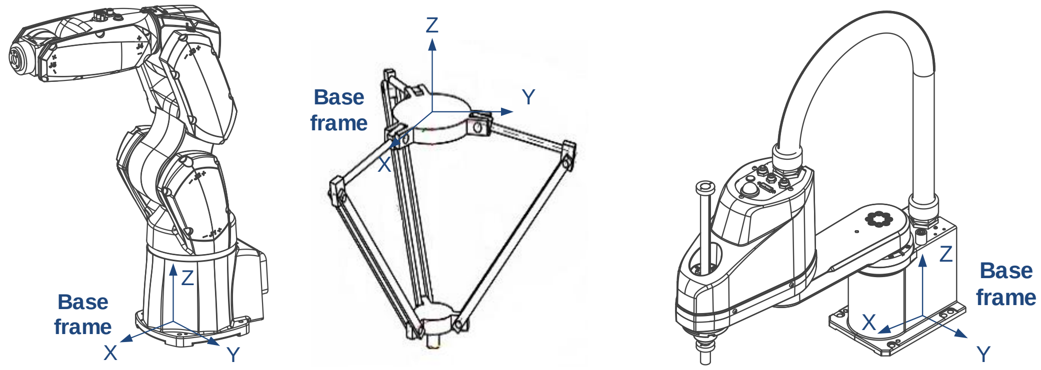
1.4.1 Joint Coordinate System

The joint coordinate system is located at the joints of the robot.

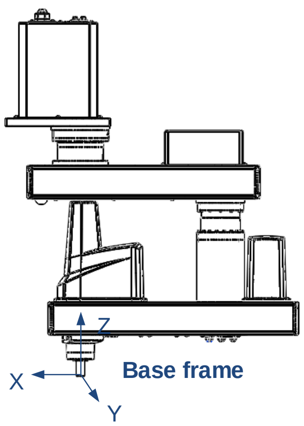


1.4.2 Base Coordinate System

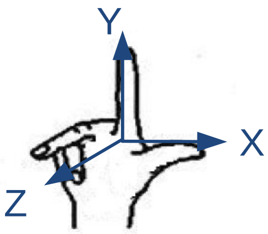
The base coordinate system is also called the robot coordinate system and is generally located at the base of the robot.



The base coordinate system of inverted SCARA robot is located on the flange at the zero point and does not move with the flange.



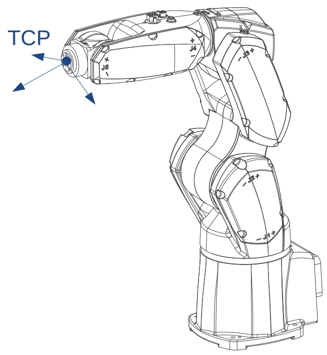
The base coordinate system is a Cartesian coordinate system. (The tool and user coordinate systems are also Cartesian coordinate systems.) The X and Z directions can be determined first, and then the Y direction can be determined by the right-hand rule.



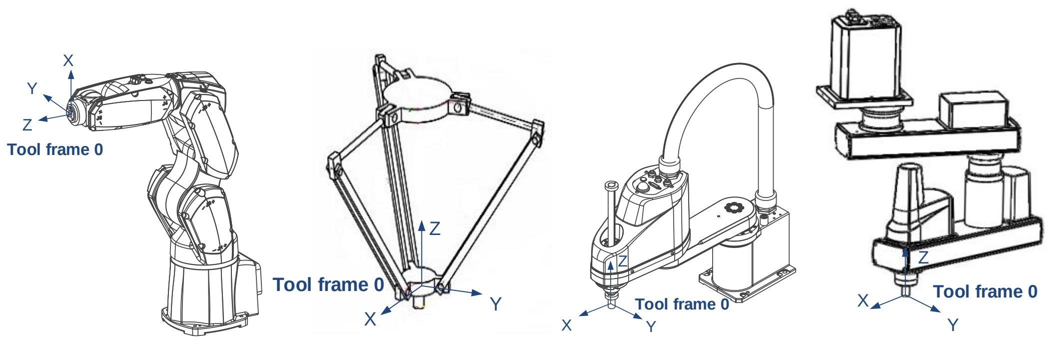
Right-hand rule

1.4.3 Tool Coordinate System

The tool coordinate system is attached to a tool. The tool center point (TCP) is a reference point for a position where the robot reaches. The tool endpoint is generally taken as the TCP and the direction can be freely defined.



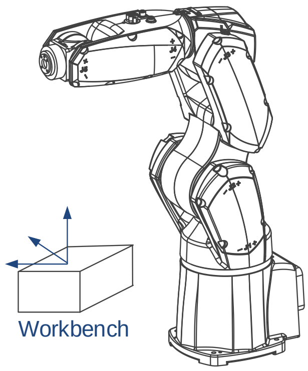
Up to 16 tool coordinate systems can be defined in Inovance robot control system. Tool 0 indicates that no tool is used, in which case, the tool coordinate system is located at the end of the manipulator.



Tools 1-15 can be user defined.

1.4.4 User Coordinate System

It is a user-defined coordinate system. The user coordinate system can be set arbitrarily and is generally set on specific objects, such as a workbench or a conveyor.



In Inovance robot control system, up to 16 user coordinate systems can be defined in one project. User 0 indicates that the base coordinate system is also used as the user coordinate system. Users 1-15 can be user defined.

1.4.5 Other Coordinate Systems

Some specialized coordinate systems are used for some special processes.

Coordinate System No.	Coordinate System	Process Application	Description	Remarks
5	Fixed camera FOV coordinate system	Specially used for the field of view function	Points (X, Y, θ) within fixed camera field of view are defined in this coordinate system.	See also 1.5.4 Position Variables for Special Processes
6	Mobile camera FOV coordinate system	Specially used for the field of view function	Points (X, Y, θ) within mobile camera field of view are defined in this coordinate system.	
7	Object coordinate system on the conveyor belt	Specially used for the tracking process	Points (X, Y, Z, A, B, C) relative to the object coordinate system on the conveyor belt are defined in this coordinate system.	

1.5 Position Variables

1.5.1 Overview

In Inovance robot control system, a point in the space is expressed using a "position variable" that stores information about coordinate values, arm parameters, coordinate system, tool No. and user No. Position variables are divided into global position variables and local position variables.

- The scope of the global location variable is a single project and the global location variable can be used in all programs within a single project. It is denoted by P[***] and stored in the project.
- The scope of the local position variable is a single program file and the local position variable

is used only in the current program file. It is denoted by LP[***] and stored in the program file.

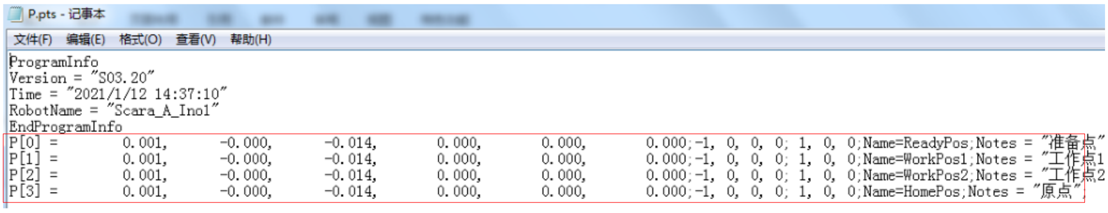
1.5.2 Storage Format of Position Variable

The position variables are stored in the following format:

```
LP[1] = 0.001, -0.000, -0.014, 0.000, 0.000, 0.000;-1, 0, 0, 0; 1, 0, 0;Name=GetLPos;Notes ="pickup point";
        ①                                           ②           ③        ④            ⑤
```

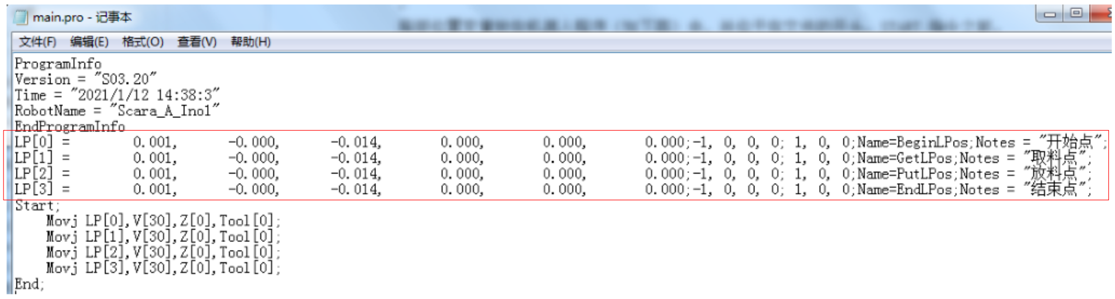
①It records the coordinate data of the position variable, and the number of data is always 6 (the value of extra axes is 0). The data are separated by "," and end with ";".
②It records the arm parameters of the position variable. The arm parameters are separated by "," and ended with ";".
③It records the coordinate system number + tool number + user number of the position variable, which are separated by "," and end with ";".
④It records the label of the position variable and ends with ";". The label should be less than 20 characters, start with letters and contain only letters, numbers and underscores. No entry when no label is present.
⑤It records the remarks of the position variable and ends with ";". The remarks are required to be less than 100 characters, with no restrictions on Chinese and English characters, and no equal sign can be entered. (Chinese input is not supported on the handheld device, but display is supported). No entry when no remarks are available.

The global position variables are stored in the project file P.pts and are common to all programs within the project.



```
P.pts - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
ProgramInfo
Version = "S03.20"
Time = "2021/1/12 14:37:10"
RobotName = "Scara_A_Ino1"
EndProgramInfo
P[0] =    0.001,   -0.000,   -0.014,   0.000,   0.000,   0.000;-1, 0, 0, 0; 1, 0, 0;Name=ReadyPos;Notes = "准备点"
P[1] =    0.001,   -0.000,   -0.014,   0.000,   0.000,   0.000;-1, 0, 0, 0; 1, 0, 0;Name=WorkPos1;Notes = "工作点1
P[2] =    0.001,   -0.000,   -0.014,   0.000,   0.000,   0.000;-1, 0, 0, 0; 1, 0, 0;Name=WorkPos2;Notes = "工作点2
P[3] =    0.001,   -0.000,   -0.014,   0.000,   0.000,   0.000;-1, 0, 0, 0; 1, 0, 0;Name=HomePos;Notes = "原点";
```

The local position variables are placed in the robot program and are at the beginning of the file before the START instruction. See the illustration below.



```
main.pro - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
ProgramInfo
Version = "S03.20"
Time = "2021/1/12 14:38:3"
RobotName = "Scara_A_Ino1"
EndProgramInfo
LP[0] =    0.001,   -0.000,   -0.014,   0.000,   0.000,   0.000;-1, 0, 0, 0; 1, 0, 0;Name=BeginLPos;Notes = "开始点";
LP[1] =    0.001,   -0.000,   -0.014,   0.000,   0.000,   0.000;-1, 0, 0, 0; 1, 0, 0;Name=GetLPos;Notes = "取料点";
LP[2] =    0.001,   -0.000,   -0.014,   0.000,   0.000,   0.000;-1, 0, 0, 0; 1, 0, 0;Name=PutLPos;Notes = "放料点";
LP[3] =    0.001,   -0.000,   -0.014,   0.000,   0.000,   0.000;-1, 0, 0, 0; 1, 0, 0;Name=EndLPos;Notes = "结束点";
Start;
    Movj LP[0],V[30],Z[0],Tool[0];
    Movj LP[1],V[30],Z[0],Tool[0];
    Movj LP[2],V[30],Z[0],Tool[0];
    Movj LP[3],V[30],Z[0],Tool[0];
End;
```

1.5.3 Attributes of Position Variables

Coordinate system:

Coordinate system used to get points. The meanings of coordinate values are indicated by the coordinate system.

Coordinate value:

The meanings of coordinate values are indicated by a coordinate system.

When the coordinate system is 1, coordinate values are joint values of the robot (J1, J2, J3, J4, J5, J6). Joint values are rotation angles of an arm relative to the zero position.

When the coordinate system is 2, coordinate values are poses (X, Y, Z, A, B, C) of the robot flange center point* relative to its base coordinate system.

When the coordinate system is 3, coordinate values are poses (X, Y, Z, A, B, C) of the TCP relative to its base coordinate system.

When the coordinate system is 4, coordinate values are poses (X, Y, Z, A, B, C) of the TCP relative to the user coordinate system.

*Flange center point: Center point on the flange end face of the last axis of the robot that represents the end reference position of the manipulator.

Tool number:

Tool currently used.

Note: Always pick up the correct tool!

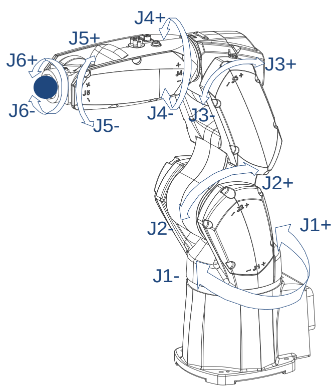
User number:

User coordinate system currently used.

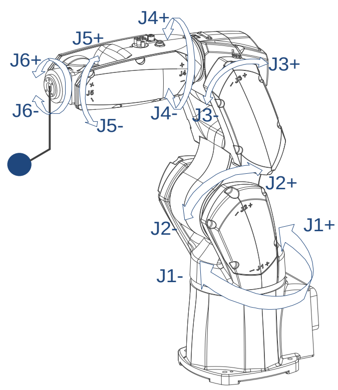
Note: Used when the user coordinate system is used.

Example:

Variable name	Coordinate value						Coordinate system	Tool	User number
P[1]	0	0	0	0	-90	0	1	0	0
P[2]	0	0	0	0	-90	0	1	1	0
P[3]	100	0	100	0	0	0	2	0	0
P[4]	100	0	100	0	0	0	2	1	0
P[5]	110	0	60	0	0	0	3	1	0
P[6]	50	0	60	0	0	0	4	1	1

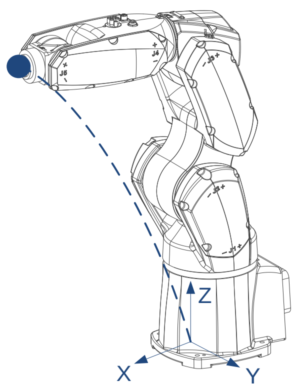


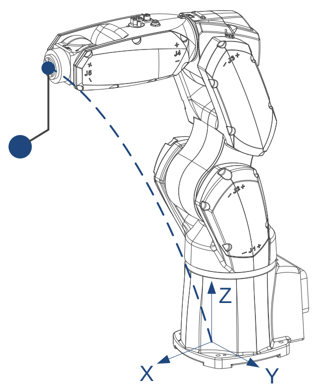
P[1]: Without tools, points taken in joint coordinate system



P[2]: With tools, points taken in joint coordinate system

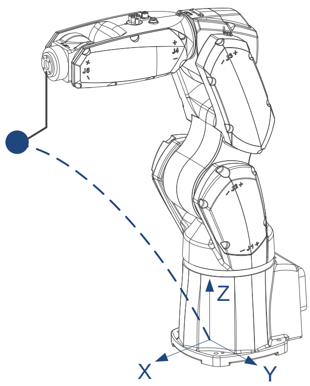
Note that the TCP position is not directly represented by a coordinate system, but is associated with a tool

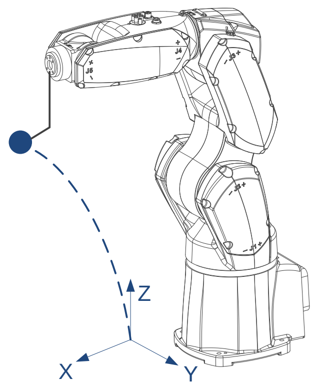




P[3]: Without tools, points taken in base coordinate system

P[4]: With tools, points taken in base coordinate system Note that the TCP position is not directly represented by the coordinate values, but is associated with a tool.





P[5]: With tools, points taken in tool coordinate system

P[6]: With tools, points taken in user coordinate system

Arm parameters:

The robot can reach the same pose in many ways. Arm parameters are used to distinguish these ways.

For a 6-axis serial robot:

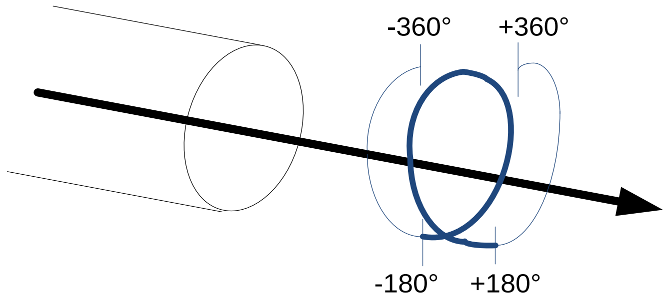
Arm parameter 1 represents multiple turns of J1;

Arm parameter 2 represents multiple turns of J4;

Arm parameter 3 represents multiple turns of J6;

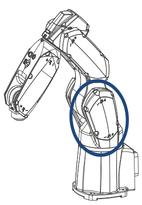
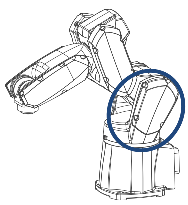
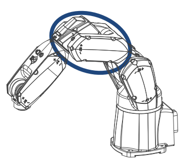
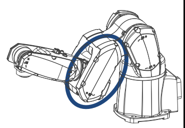
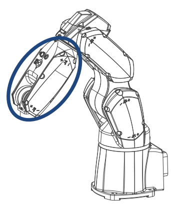
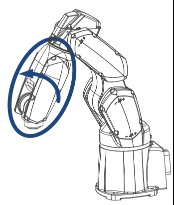
Arm parameter 4 represents a combination of different configurations of the waist joint, elbow joint, and wrist joint

Arm parameters 1 and 2		
-1	0	1
Jx (-360° to -180°)	Jx (-180° to180°)	Jx (180° to 360°)



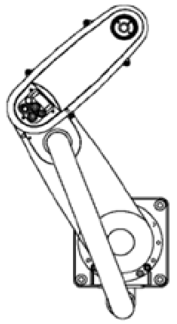
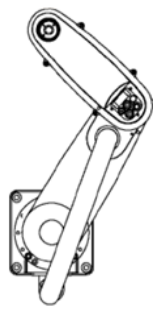
Arm parameter 3 (value of joint coordinate J6)				
(-900 to -540)	(-540 to -180)	(-180 to 180)	(180,540)	(540,900)
-2	-1	0	1	2

Arm parameter 4								
Value of arm parameter 4	0	1	2	3	4	5	6	7
Waist arm type	Forward	Forward	Forward	Forward	Backward	Backward	Backward	Backward
Elbow arm type	Upward	Upward	Downward	Downward	Upward	Upward	Downward	Downward
Wrist arm type	Upward	Downward	Upward	Downward	Upward	Downward	Upward	Downward

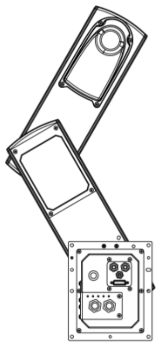
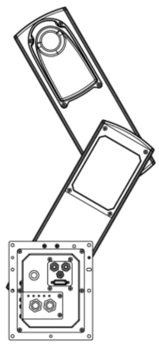
Waist forward	Waist backward	Elbow upward	Elbow downward	Wrist not flipped	Wrist flipped
					

For SCARA robots, the values of the arm parameters 1, 4 are valid.

Arm parameter 1		Arm parameter 4 (value of joint coordinate J4)				
-1	1	(-900,-540)	(-540 to -180)	(-180 to 180)	(180,540)	(540,900)
Left arm	Right arm	-2	-1	0	1	2

		When multiple turns are involved, the arm parameter is increased by 1 for every 360° and decreased by 1 for every 360°.

For inverted SCARA robots, arm parameters 1, 2, 3, 4 are valid.

Arm parameter 1	
-1	1
Left arm orientation	Right arm orientation
	

Arm parameter 2 (value of joint coordinate J1)		
(-540 to -180)	(-180 to 180)	(180,540)
-1	0	1
When multiple turns are involved, the arm parameter is increased by 1 for every 360° and decreased by 1 for every 360°.		

Arm parameter 3 (value of joint coordinate J2)		
(-540 to -180)	(-180 to 180)	(180,540)
-1	0	1
When multiple turns are involved, the arm parameter is increased by 1 for every 360° and decreased by 1 for every 360°.		

Arm parameter 4 (value of joint coordinate J4)				
(-900 to -540)	(-540 to -180)	(-180 to 180)	(180,540)	(540,900)
-2	-1	0	1	2
When multiple turns are involved, the arm parameter is increased by 1 for every 360° and decreased by 1 for every 360°.				

For the Delta robot, arm parameter 4 is meaningful.

Delta arm parameter 4 (Joint coordinate J4 value)		
(-540 to -180)	(-180 to 180)	(180,540)
-1	0	1
When multiple turns are involved, the arm parameter is increased by 1 for every 360° and decreased by 1 for every 360°.		

Note: If arm parameters are modified after teaching, the robot will reach the same pose in an another arm type with a great change to motion status. Modify the arm parameters with caution!

1.5.4 Position Variables for Special Processes

For some special processes, more position variable types are defined as follows:

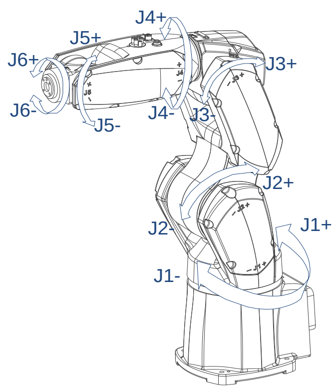
Position points within fixed camera field of view	The coordinate system number is 5, the coordinate value of the point in the camera field of view is (X, Y, θ), and the corresponding storage form is (X, Y, 0, A, 0, 0). The tool number is the number of the tool used, and the user number is the visual coordinate system number used.	Specially used for the field of view function
Position points within mobile camera field of view	The coordinate system number is 6, the coordinate value of the point in the camera field of view is (X, Y, θ), and the corresponding storage form is (X, Y, 0, A, 0, 0). The tool number is the number of the tool used, and the user number is the visual coordinate system number used.	Specially used for the field of view function
Used for the tracking process	The coordinate system No. is 7. Coordinate values (X, Y, Z, A, B, C) are used to define the synchronous motion of an object on the conveyor belt and represent the coordinate position of the object relative to the conveyor belt.	Specially used for the tracking process

1.6 Offset

Offset is used to describe the motion in space, which can be the motion of joints J1-J6 described in joint coordinate system, or the spatial change of a certain position and pose XYZABC relative to a certain coordinate system in three-dimensional space.
In the Inovance robot system, the offset is divided into the following three types.

1.6.1 Joint Offset

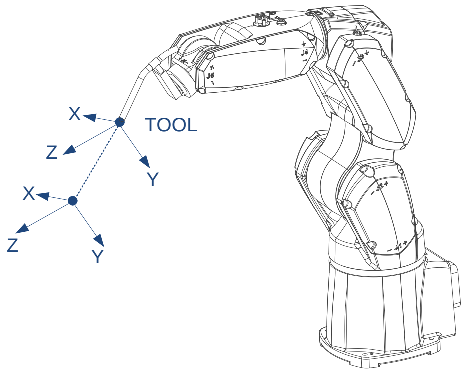
Function: Moves the joints of robot.
Instruction: OffsetJ



1.6.2 Offset Along the Current Tool Pose

Function: Moves the robot along the current tool pose.

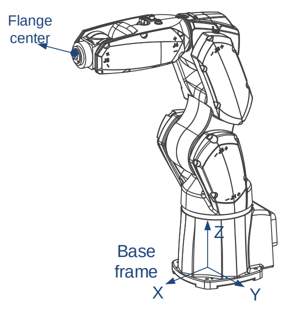
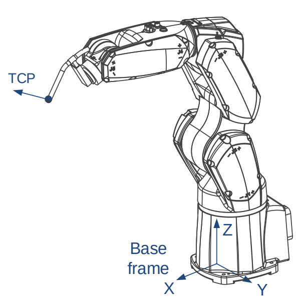
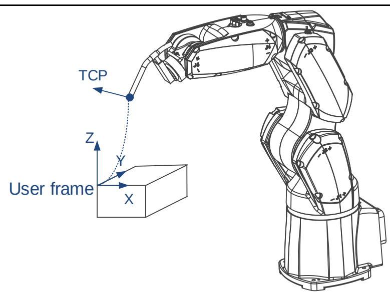
Instruction: OffsetT



1.6.3 Offset in a Cartesian Frame

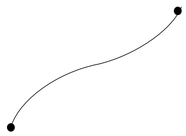
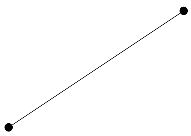
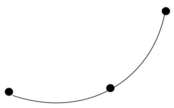
Function: Offsets a target point in a Cartesian reference system. A target point may be the TCP or flange center point. A reference system may be a user or base coordinate system.

Instruction: Offset

Several offset modes		
		

1.7 Interpolation and Transition

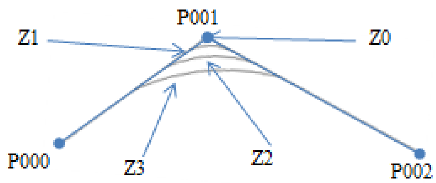
Interpolation is the basic form of motion of a robot, and complex motion is made up of a series of interpolation motions. There are three types of interpolation, depending on the interpolation trajectory.

Interpolation Type	Track	Characteristics
Joint interpolation (Movj)		Point-to-point interpolation, the fastest interpolation in which each joint moves at the fastest speed. The motion trajectory is unpredictable. It is often used for applications such as spot welding and transportation.
Linear interpolation (Movl)		The motion trajectory is straight. It is often used for applications such as track welding and surface mounting.
Arc interpolation (Movc)		The trajectory is arc-shaped.

Note: When executing Movl and Movc, the arm parameters of the robot are not allowed to change. If you need to change the arm parameters, insert the Movj instruction to complete the arm posture transition.

In the actual continuous motion process, in order to accelerate rhythm, accurate arrival is not required. At this time, the middle point of the motion will show the form of trajectory approximation, which is called transition.

In Inovance robot system, the transition can be divided into the following levels: Fine, Z[0], Z[1], Z[2], Z[3], Z[4], Z[5] and Z[CP]. See the transition feature for details.



Interpolation precision

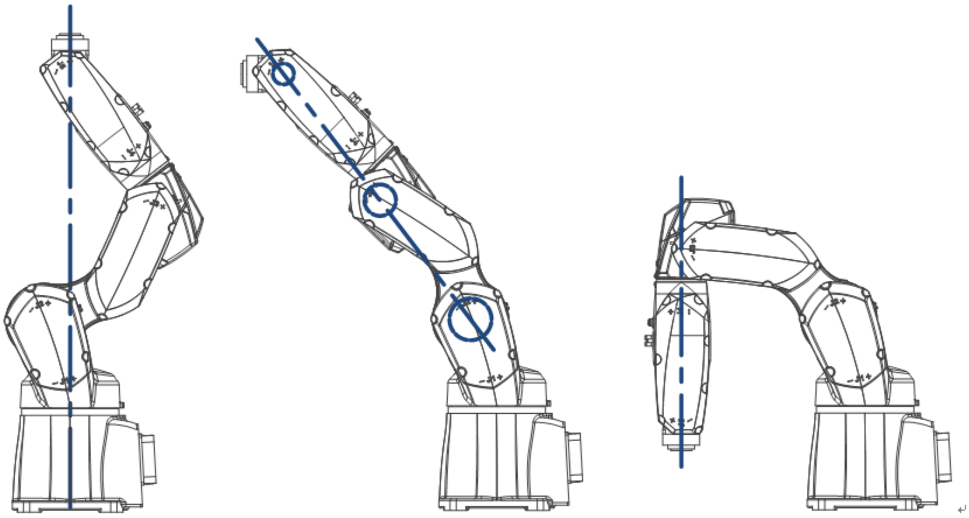
1.8 Singular Position

When moving in a non-joint coordinate system, the robot may move to certain special positions where the robot loses some freedom of movement called singular positions.

In joint interpolation Movj, the singular position does not affect normal motion. In the process of linear interpolation Movl and circular interpolation Movc, the singular position prevents the robot from moving properly.

Note: When a singular position alarm is encountered, the singular position can be exited using the joint motion mode.

There are three singular positions for the 6-axis robots, as shown in the figure below.

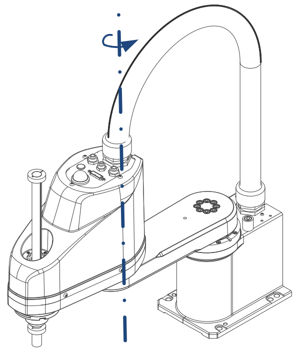


Top singularity
The intersection of J4, J5 and J6 axes is directly above J1

Extended singularity
The centers of J2, J3 and J5 axes are collinear

Wrist singularity
J4 and J6 axes are collinear

The SCARA robot has only one singular position, at J2=0°, when the 1st and 2nd arms are in a straight line.

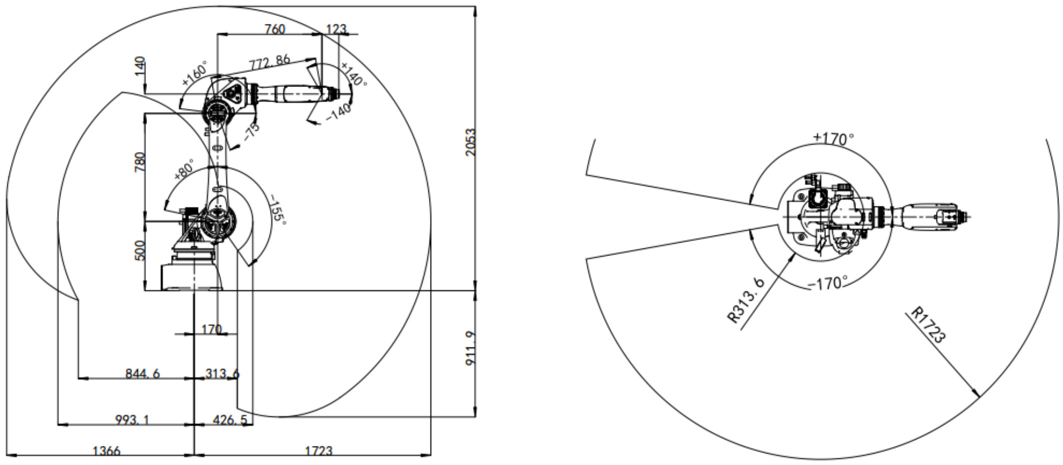


(SCARA robot's singular position at J2=0°)

The singular position of Delta robot is not in the working range, and there is no singular position.

1.9 Motion Range and Interference Area

The range of motion of a robot is a collection of all points that can be reached by the end of the robot's arm. The range of motion of a robot is related to the length of its arm and the range of motion of its joints.



Interference area: In the range of motion, there are often areas where the end actuator is prevented from reaching. When in these areas, the robot will collide with its own components or external devices. These areas are called interference areas.

The user can customize the interference area.

2 Getting Started

2.1 Operating Process

As an example, the following describes the process of using the robot, including power-on and connection, user login, status check, and manipulation of the robot.

2.2 Power-On and Connection

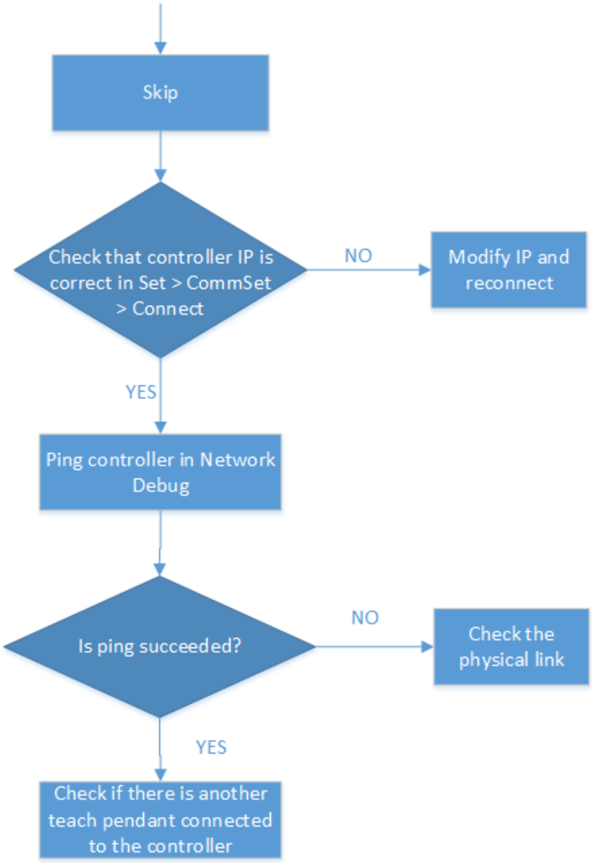
Power up the controller and the teach pendant displays the connecting status as shown below.

When the connection is successful, the following screen is displayed.

- For the handheld teach pendant, by default it automatically connects to the controller successfully after power-up of the controller.
- For the PC-based teach pendant, when connecting it to the controller for the first time, you need to click the Skip button, go to Set > System > CommSet, enter the IP address of the controller, and then click Connect.

  When the teach software is launched, the connection is automatically performed based on the last communication address. If the connection fails, take measures given in the following table according to the screen display. For details, refer to the following table.

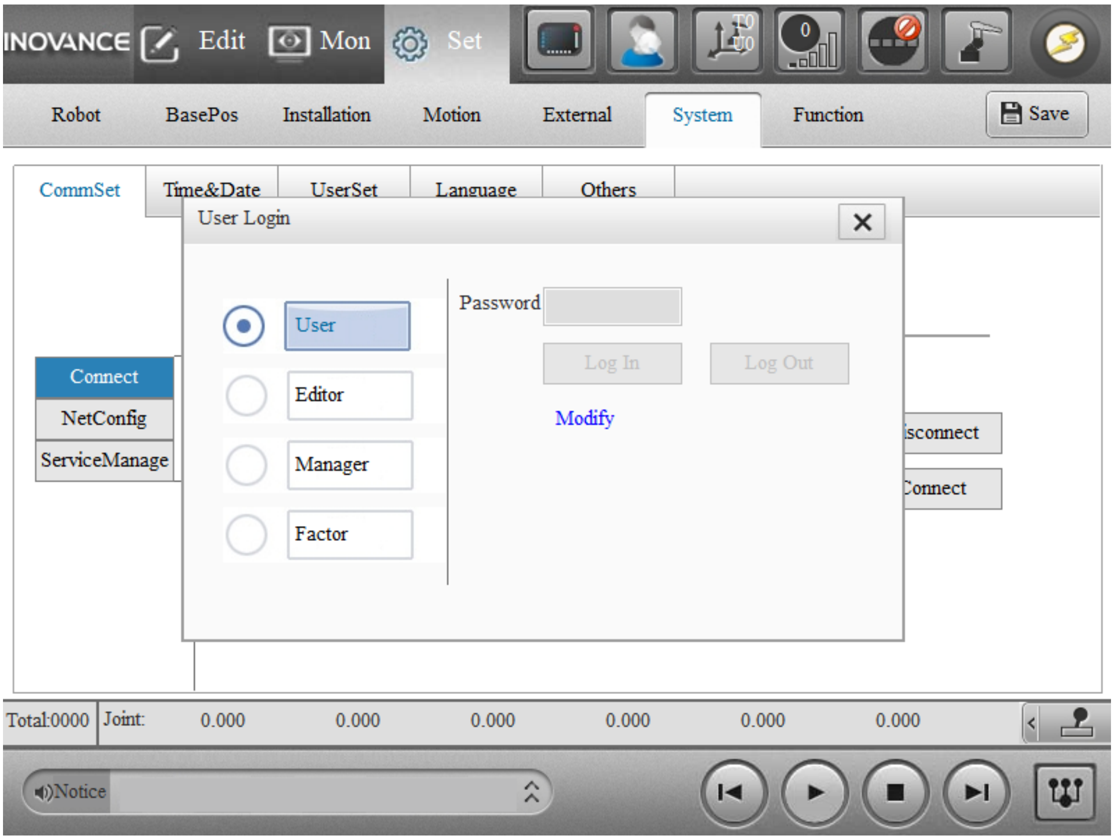
Screen Display	Problem	Solution

Communicating with the controller...	Network connection failed	
Others	Software or hardware problems	Contact the manufacturer for help

Note: If the current project is empty, InoTeachPad will jump to the file list page after connecting to the controller, but subsequent operations can still be carried out.

2.3 User Login

You can click  in the upper right corner to perform login.



Select a mode according to the current user's role and log in.

Mode	Audience	Initial Default Password
User	Operators on the production line. The operators can directly control robot motion and run programs, control the status of I/O and force the I/O.	No password is required.
Editor	Teaching programmer. Compared with the user mode, the program editing function is added so that users can perform teaching programming. Permissions such as equipment control and mechanical locking are also added.	The initial password is 000000.
Manager	Advanced users. Most of the system operation permissions are included.	The initial password is 000000.
Factory	Manufacturer maintenance personnel. The manufacturer maintenance personnel have the highest permissions and can perform operations such as upgrading and joystick calibration.	Reserved by the manufacturer

Different user modes correspond to different operation permissions:

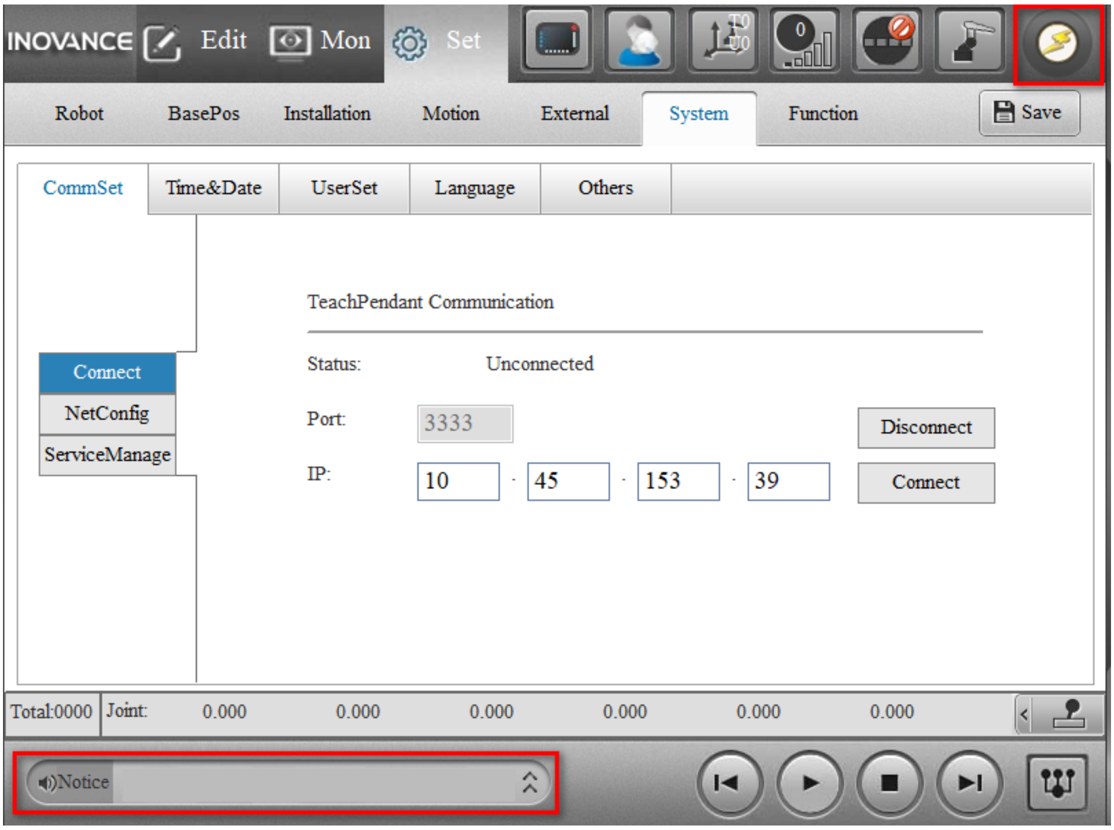
Operation	Permission

	User	Editor	Manager	Factory
Motion control	√	√	√	√
Switch coordinate system, tool number, user number and grip load number	√	√	√	√
Switch speed	√	√	√	√
Switch jog mode	√	√	√	√
Run the project	√	√	√	√
Modify, edit the project	×	√	√	√
I/O control	√	√	√	√
Robot settings	×	×	×	√
Work origin	×	√	√	√
Absolute zero	×	×	√	√
Homing calibration (SCARA robot only)	×	×	√	√
Load on arm	×	√	√	√
Setup form (6-axis robot only)	×	×	×	√
Teach/play parameters	×	×	√	√
Axis limit	×	×	√	√
Tracking error/arrival error/current limit/average load rate limit	×	×	×	√
Interference area settings	×	√	√	√
Collision detection settings	×	×	√	√
Advanced features	×	×	√	√
Bus switch	×	×	√	√
I/O mapping	×	√	√	√
IRLink settings	×	×	√	√
Communication settings	Only for communication of the teach pendant	×	√	√
Time and date	×	×	√	√
Mechanical lock	×	√	√	√
COM port switch	×	√	√	√
Emergency stop trigger	×	×	√	√
Emergency stop mode	×	×	√	√
Safety door	×	×	√	√

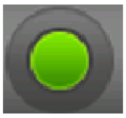
Flying trigger I/O	×	×	√	√
SN match	×	×	×	√
Screen calibration	×	×	√	√
Screen rotation	×	×	√	√
Brightness and screensaver	×	×	√	√
Joystick calibration	×	×	√	√
Configuration backup	×	×	√	√
Configuration load	×	×	√	√
Memory card backup	×	√	√	√
Memory card load	×	√	√	√
Point file load	×	√	√	√
System update	×	×	×	√
Factory reset	×	×	×	√
Memory card formatting	×	×	×	√
Clear historical alarm	×	×	√	√
Clear PLC configuration	×	×	×	√
Network debugging	√	√	√	√
Controller debugging	×	×	√	√
Teach pendant commissioning	×	×	×	√
System diagnostics	×	×	√	√
Servo check	×	×	×	√
Control device	×	√	√	√
Feature expansion	×	×	√	√

2.4 Status Check

After connection is successful, check the status indicator in the upper right corner and the message bar at the bottom. The robot is normal only when the status indicator indicates a standby or enabled status. In case of anomalies, take necessary measures.



Status indicator:

Status indicator	Description	Measures
	Servo enabled: The emergency stop is released and the servo is enabled.	--
	Emergency stop: The emergency stop button is pressed and the robot cannot move.	--
	Standby: The emergency stop button is released and servo is not enabled.	--
	Alarm: An error occurs and needs to be handled immediately.	Read the prompt in the message bar, take measures according to the "Robot Alarms and Handling Method" in the Appendix, and then click the alarm button to clear the alarm.
	Warning: An abnormality occurs and the system prompts users.	Read the prompt in the message bar, take measures according to the "Robot Alarms and Handling Method" in the Appendix, and then click the alarm button to clear the warning.
	Offline: The network is disconnected and communication with the	Go to **System** > **CommSet**, enter the IP address of controller and reconnect the teach pendant to the controller.

	controller fails.	

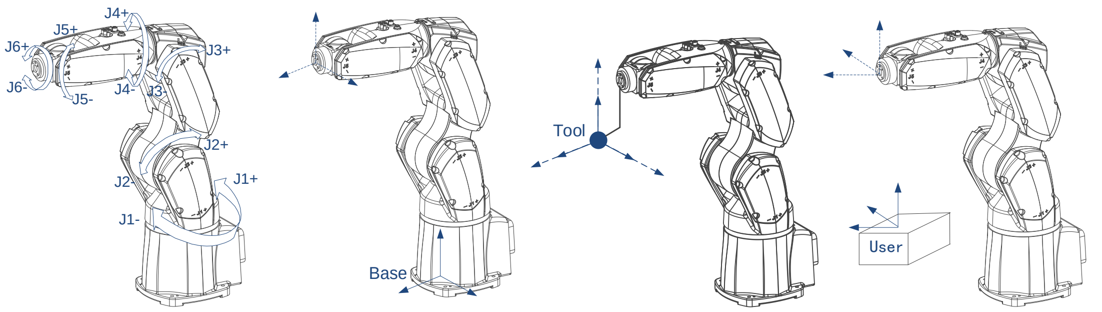
2.5 Robot Manipulation

Select the coordinate system and speed (or jog mod), click and hold the ENABLE button, and click the corresponding function buttons on the teach pendant to move the robot.

Step 1: Select the coordinate system

Click  in the upper right corner and select the coordinate system in the pop-up page.

The coordinate system determines the direction of motion. The following coordinate systems are available:



Motion along the joint coordinate system Motion along the base coordinate system Motion along the tool coordinate system Motion along the user coordinate system

Step 2: Select the speed and mode of motion

You can set the speed and jog mode by clicking the speed button and jog mode button in the upper right code.

The first button allows you to set the motion speed. There are four levels of speed, that is, 5%, 25%, 50% and 100%. You can also fine tune the speed through the speed adjustment button on the teach pendant.

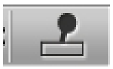
The second button allows you to set the mode of motion. You can select whether the robot jogs and set the jog parameters.

In the jog mode, when you press the axis motion button, the robot will only move a specific step at most, rather than continuously moving when you press the button.

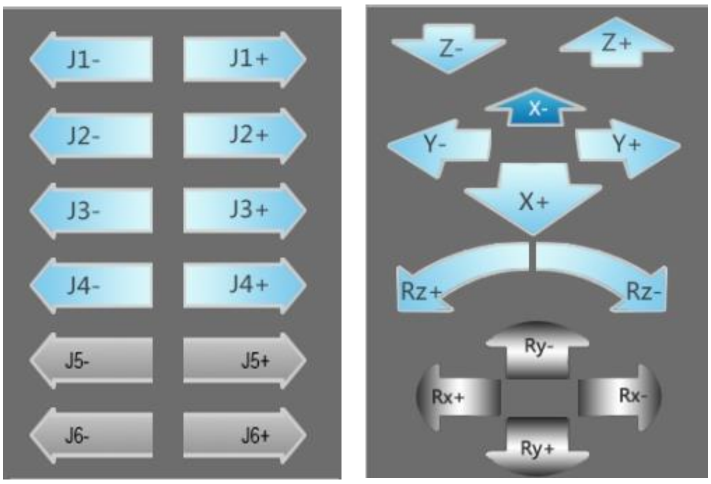
	Non-jog mode, that is, the robot moves at normal speed.
	G1 jog. Joint step 0.05°, position step 0.05mm, rotation step 0.05°
	G2 jog. Joint step 0.5°, position step 0.5mm, rotation step 0.5°
	G3 jog. Joint step 2°, position step 2mm, rotation step 2°
	User-defined jog. Go to **Set** > **Motion** > **TeachPara** > **Jog** and set the jog parameters.

Step 3: Enable and manipulate the robot

- For the handheld teach pendant, press and hold the ENABLE button.
- For the PC-based teach pendant, just click the ENABLE button and the enabled state will be kept.

Click  in the lower right corner and the teaching panel pops up.

The left teaching panel is for the joint coordinate system, and the right teaching panel is for the base/tool/user coordinate system.



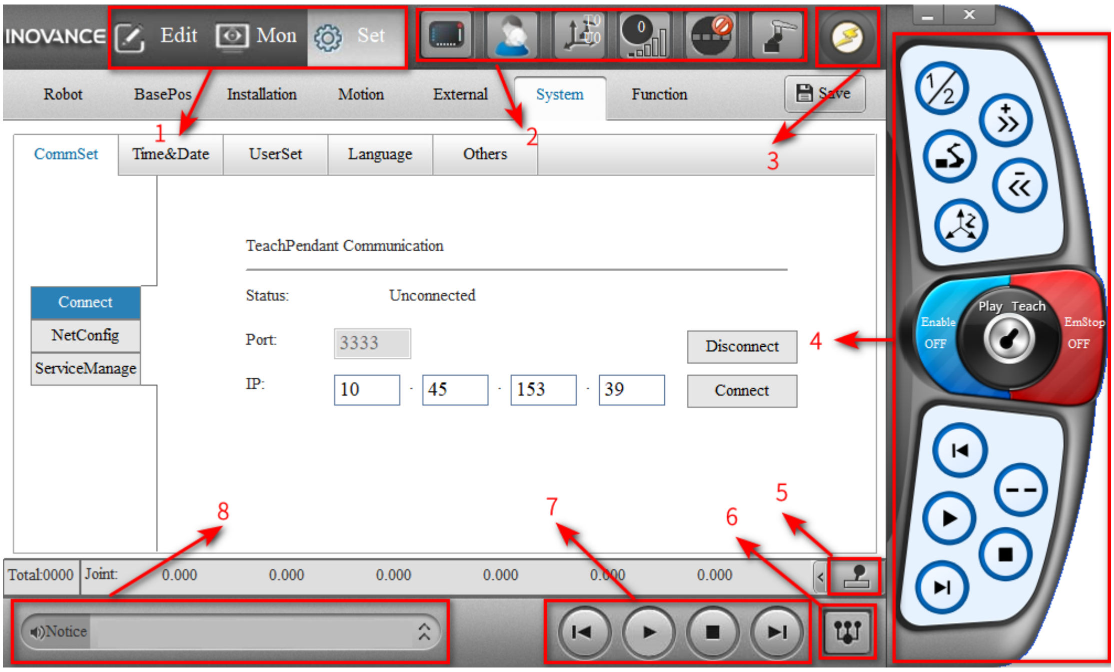
You can click the arrow buttons to move the robot accordingly.

Also, you can use physical buttons on the IRTP80 teach pendant.

3 Programming and Running

3.1 Basic Features on Main Interface

The main interface provides the following basic features.



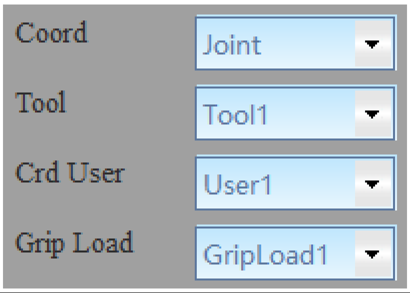
① **Panel switching bar**

You can switch among edit panel, monitoring panel, and settings panel.

② **Control toolbar**

The control toolbar includes the following buttons.

Robot control switch button		Control by InoTeachPad
		Control by other devices (detailed in 7.2.1 Robot Control)
User login button		User
		Editor

		Manager
		Factory
Coordinate system/tool/user/gripload switch button		Sets the coordinate system, tool number, and user number. Click this button to pop up the setting page. Icons: , , ,  indicate the joint coordinate system, base coordinate system, tool coordinate system and user coordinate system, respectively. T: Represents the tool selected U: Represents the user coordinate system selected The following shows the popup: 
Speed multiplying ratio		Running at 5% of a set speed
		Running at 25% of a set speed
		Running at 50% of a set speed
		Running at 75% of a set speed
		Running at 100% of a set speed
Motion mode selection button		Non-jog motion, that is, the robot moves at normal speed. When the motion button is pressed and hold, the robot moves continuously.

		Jog motion. Joint step: 0.05° Position step: 0.05 mm Rotation step: 0.05°
		Jog motion. Joint step: 0.5° Position step: 0.5 mm Rotation step: 0.5°
		Jog motion. Joint step: 2° Position step: 2 mm Rotation step: 2°
		User defined jog motion. Go to **Set** > **Motion** > **TeachPara** > **Jog** and set the jog parameters.
Mechanical lock button		Normal movement, non-mechanical locking
		Mechanically locked. Robots do not actually move.

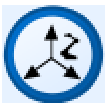
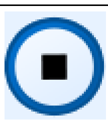
③ **Status Indicator**

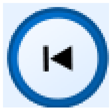
The status indicator indicates the current status of the robot, including servo enabled, standby, emergency stop, error, warning and offline.

	Servo enabled: The emergency stop button is released and servo is enabled. Motion can be performed only in enabled state.
	Emergency stop: The emergency stop button is pressed and the robot cannot move.
	Standby: The emergency stop button is released and servo is not enabled.
	Error: An abnormality occurs and needs to be handled immediately.
	Warning state: An abnormality occurs and the system prompts users.
	Offline: The teach pendant is disconnected from the controller.

④ **Control buttons on the right**

The following figures show the control buttons on the right side of the PC-based teach pendant. (Similar buttons are also provided on the handheld teach pendant. For details, see IRTP80 manual.)

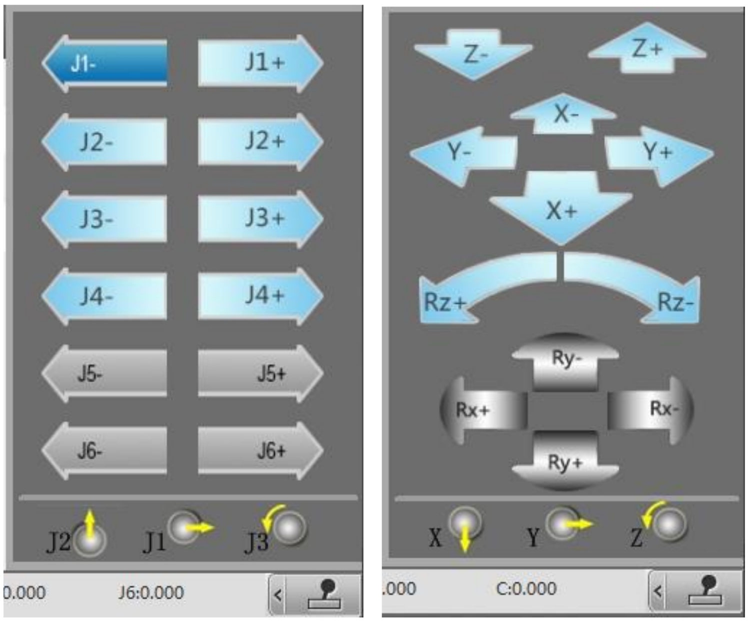
Button	Name	Description
	Left blue button: Enable Middle keyhole: Mode switch Right red button: Emergency stop	Enable: Enables the motor. Mode switch: Switches between teach and play modes. Emergency stop: Stops the robot in emergency.
	Speed +	Increases the speed increase. When clicked, the speed value increase by 1%. When clicked and hold, the speed continues to increase.
	Speed -	Decreases the speed. When clicked, the speed value decreases by 1%. When clicked and hold, the speed continues to decrease.
	Axis switch	Only for teach pendant with joystick. You can use the joystick to switch between the axis group 1/2/3 (X/Y/Z) and the axis group 4/5/6 (Rx/Ry/ Rz) axis group.
	External axis switch	Reserved
	Coordinate system selection	Switches between coordinate systems including joint coordinate system, base coordinate system, tool coordinate system, and user coordinate system.
	Handheld teach pendant: Teach/Play switch PC-based teach pendant: Jog	Handheld teach pendant: Switches between teach and play modes. PC-based teach pendant: Sets the jog motion parameters.
	Start	In play mode, click this button to start running the program. In teach mode, when you click and hold this button, the robot runs continuously; when you release the button, the robot pauses.
	Stop	When the robot is running, click this button to stop the robot.
	Forward	In teach mode, click this button to execute one line of the program.

	Back	In teach mode, click this button to go back to the previous line. This button is reserved.

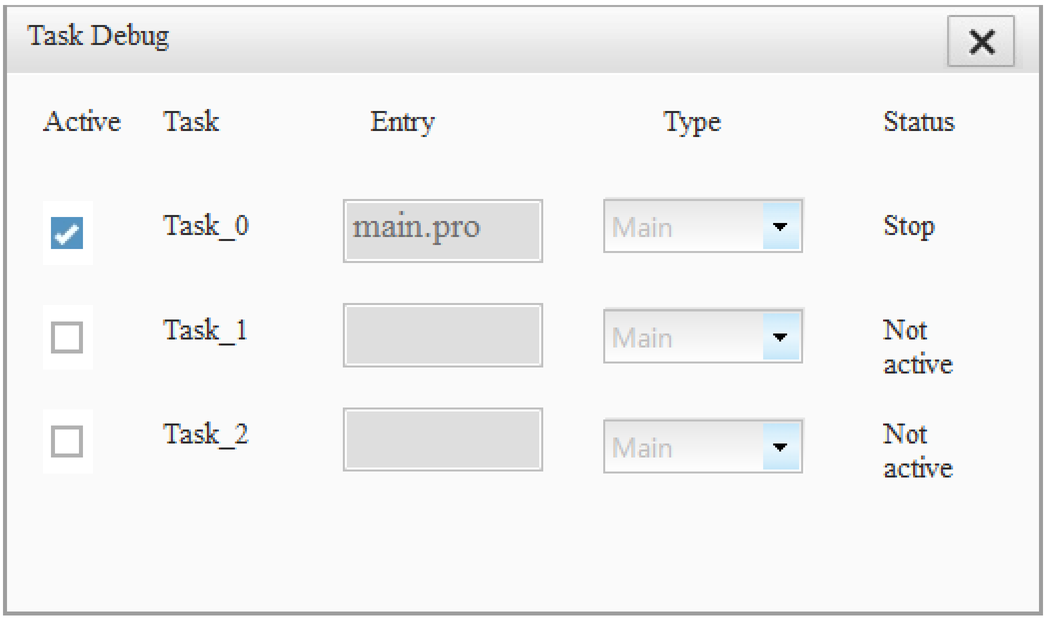
⑤ **Teaching panel**

You can click the buttons on the teaching panel to control motion of the robot.

The teaching panel in the left figure is displayed when you select the joint coordinate system, and the teaching panel in the right figure is displayed when you select the base coordinate system, the tool coordinate system, or the user coordinate system.



⑥ **Task manager**



You can activate/deactivate the main task and multitask, but cannot activate/deactivate the xqt task. Check the box to take effect immediately.

The entry program, task type, and task status are displayed.

Task status: Inactive, Running, Stopped, Finished.
Inactive: An entry program is available, but the task is set to inactive.
Running: Single step or continuous operation in teach mode, or continuous operation in play mode
Stopped: The task is stopped.
Finished: The task is finished.

To activate or deactivate a task, the robot must be under the control or InoTeachPad and in a non-teach mode.
Note: If you modify a program for a static task, the new program will not take effect when you directly reactivate it; you must click Save again on the project configuration page.

⑦ Motion control bar

The motion control bar is used to control program execution and includes start, stop, forward and back buttons. These buttons work differently in programming mode, teach mode, and play mode.

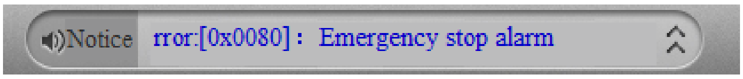
Button	Function	Programming	Teach	Play
	Back	Disabled	Disabled	Disabled
	Start	Disabled	Click and hold this button to move the robot, and release it to stop the robot.	Click this button to start.
	Stop	Disabled	Stop	Stop
	Forward	Disabled	Click and hold this button to execute one step of the program, and release it to stop the execution.	Disabled

⑧ Message window

The message window displays prompt and error messages.

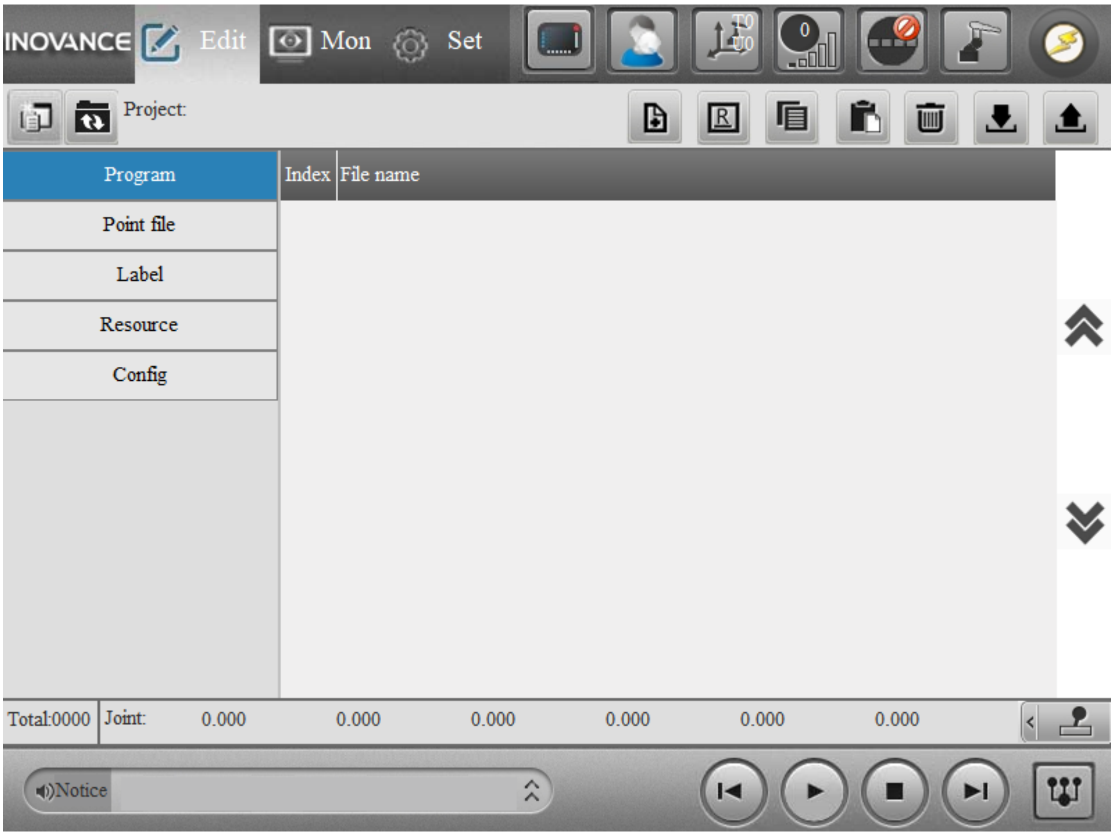
Prompt message: Prompts the user for information about certain actions. For example, when you perform SD card formatting operation in user mode (SD card formatting requires manager permissions higher than user permissions), a prompt message displays in the message bars.

Alarm message: Displays the alarm number and alarm description. For handling of the alarms, see Appendix: Robot Alarms and Solution.



3.2 Project Manager

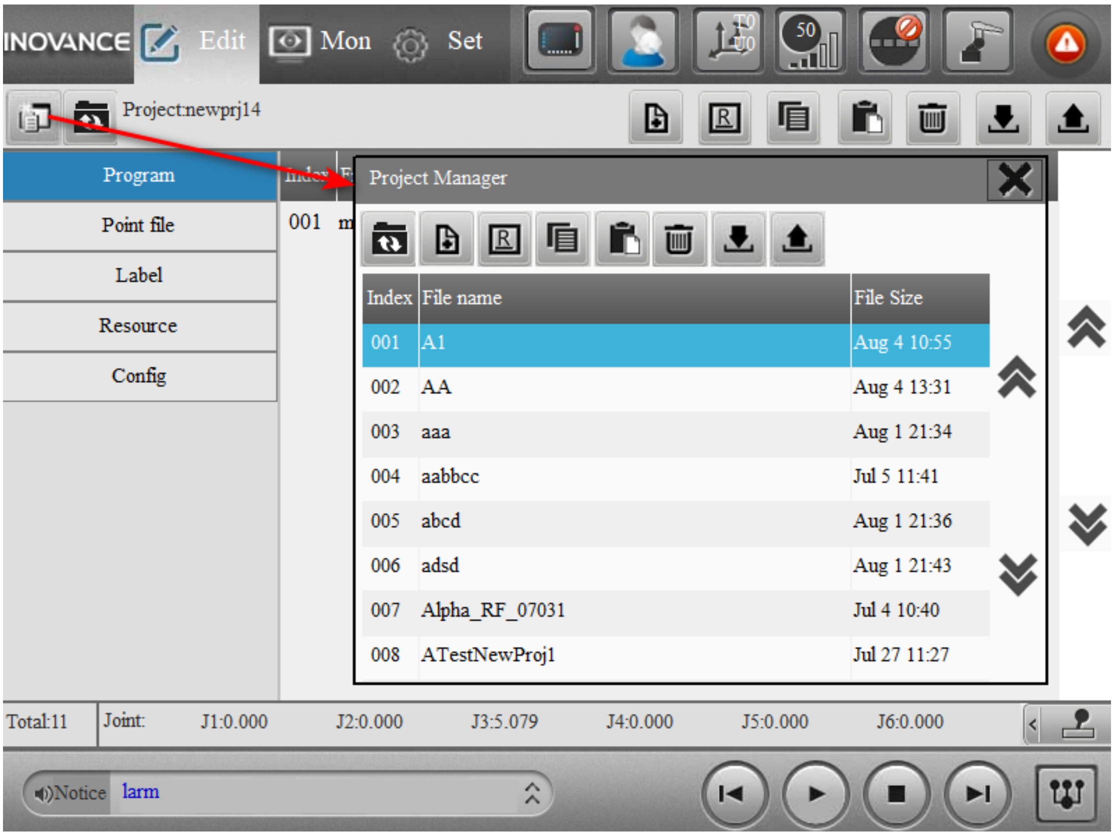
The figure shows the project manager.



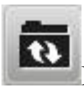
The following features are supported.

(1) Managing the project

 Opens the project list window.



In pop-up window, you can perform the following operations.

Refreshes the project directory and obtains the latest project directory from the controller.

 Creates a new project.

Renames a project.

Copies a project.

Pastes a project.

Deletes a project.

 Imports a project from local to the controller. You can choose a file with the extension prj for import, and the entire folder where the current prj file is located as a project to the controller.

 Exports a project from the controller to local

Note:

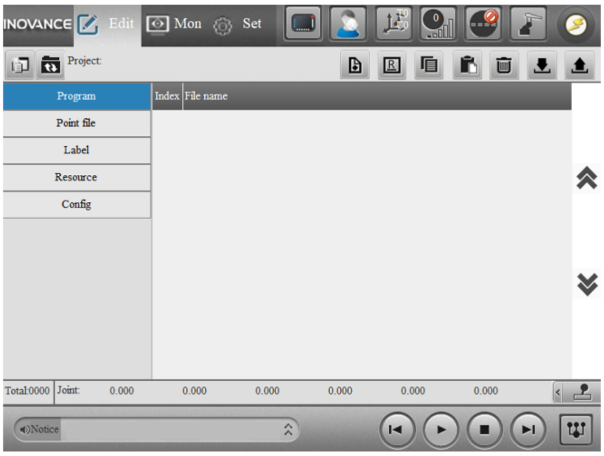
The maximum project name length is 16 characters, beginning with a letter and consisting of letters, numbers, and underscores.

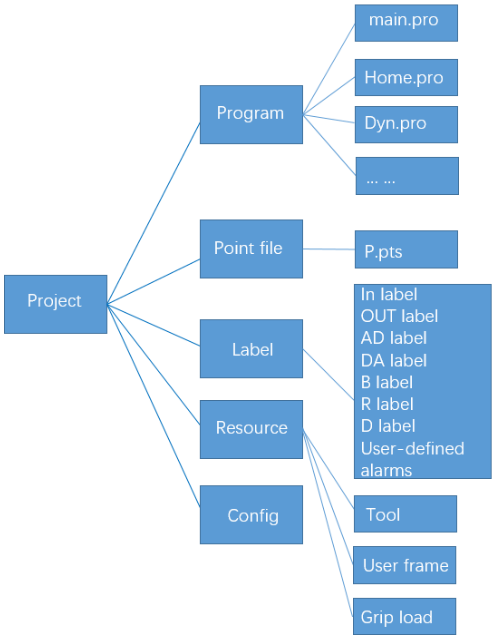
The current active project cannot be renamed or deleted.

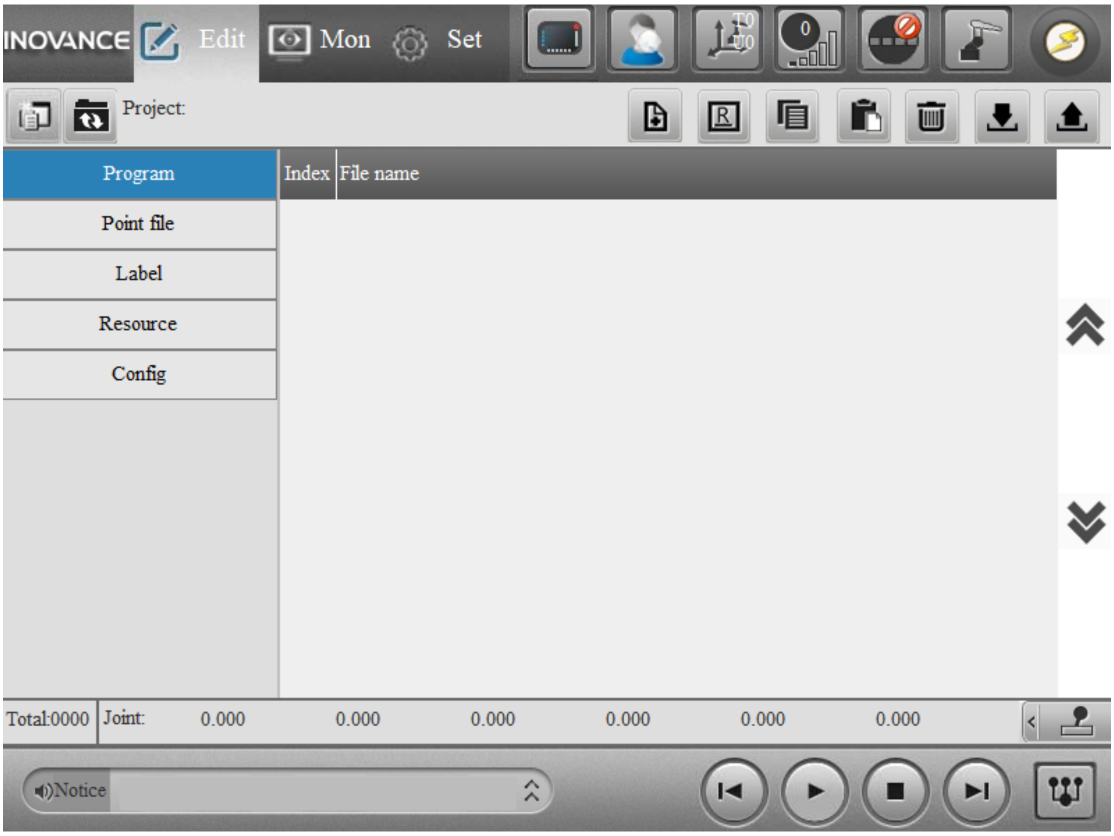
Note: Users are not allowed to directly operate program files or other configuration files of the project using FTP.

(2) Viewing/operating the project files

Click the project item on the left side and the corresponding files are displayed on the right side.







Use the buttons in the upper right corner to perform operations.

Add Rename Copy Paste Delete Import Export

The program files support operations including creation, rename, copy, delete, import and export. The point files and resource files support operations including import and export.

Precautions:

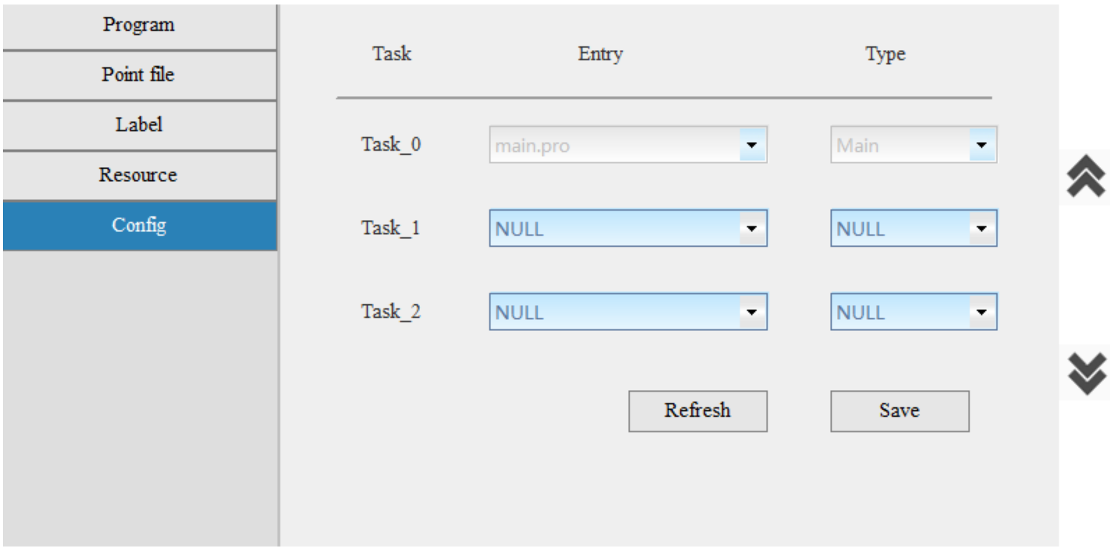
The program name has a length of up to 26 characters, beginning with a letter and consisting of letters, numbers, and underscores.

One project contains up to 16 program files.

The names of the label files and resource files in each project are fixed, and will be checked during file import.

For program files, those with too old version and those with illegal file names cannot be imported.

(3) Project configuration

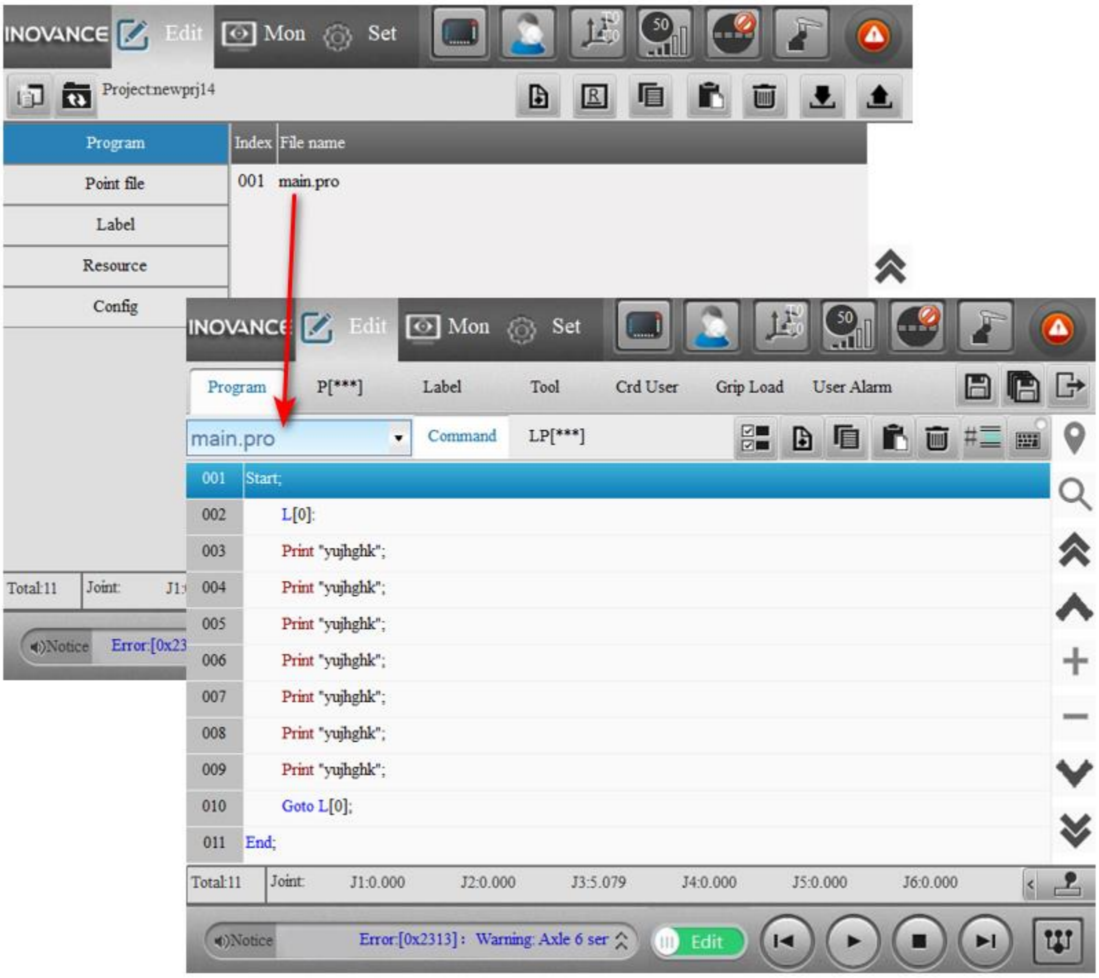


The entry program for MainTask is main.pro.
You can configure the entry program for Task_1 and Task_2 and set the task properties to static or dynamic.

The configuration takes effect after you click **Save**. The **Entry** drop-down list only shows the program name.
When you click **Save**, the static task will be reset and start running.

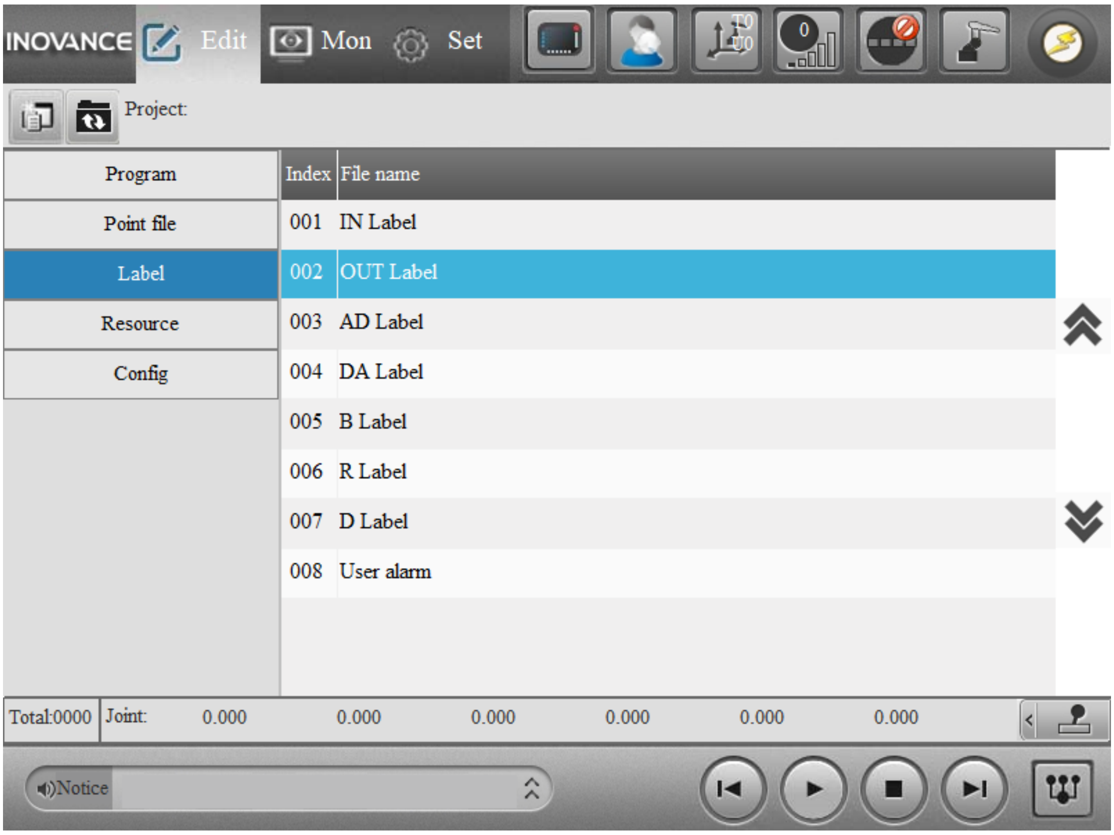
If you want to view the previously configured entry program after selecting a new entry program, click **Refresh**.

(3) Open the file
Double-click a file in the file list to open it.



(4) Project configuration

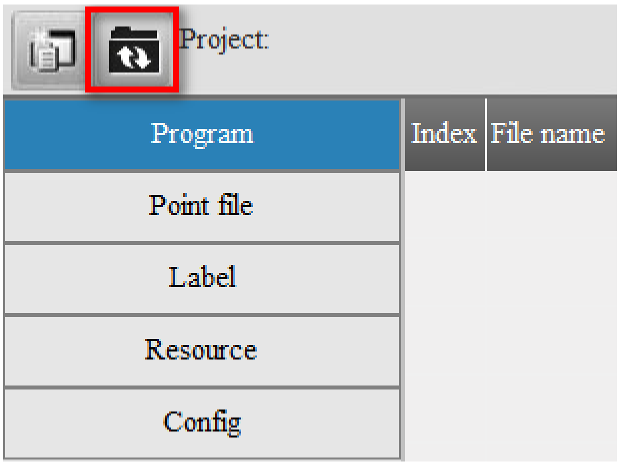
Click **Config** to configure the entry program of the task and the task type.



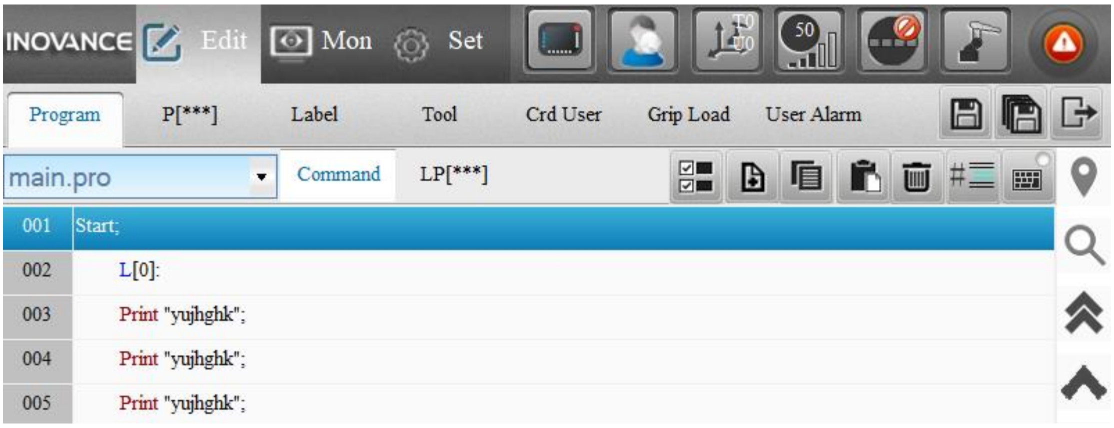
If you want to view the previous configuration after modifying the configuration, click **Refresh** before clicking **Save**.

(5) Refreshing the project

You can refresh the current project using the **Refresh** button in the upper left corner.



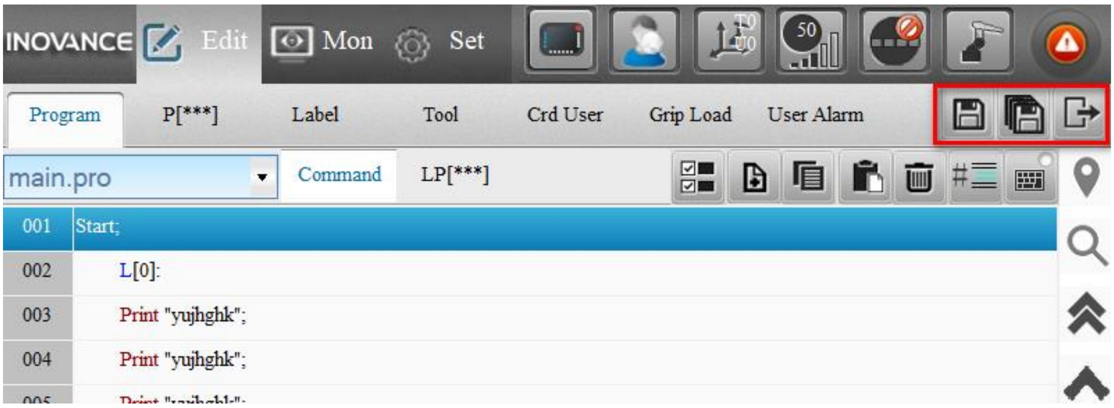
When the robot is controlled by a device other than InoTeachPad, a **Refresh** button is also available in the upper right corner for you to refresh the project.



3.3 Editing the Project

(a) Saving/refreshing/returning to the project

On the editing page, **Save**, **Save All**, and **Back** buttons are provided in the upper right corner.



Save: Saves the current page

Save All: Saves all open pages

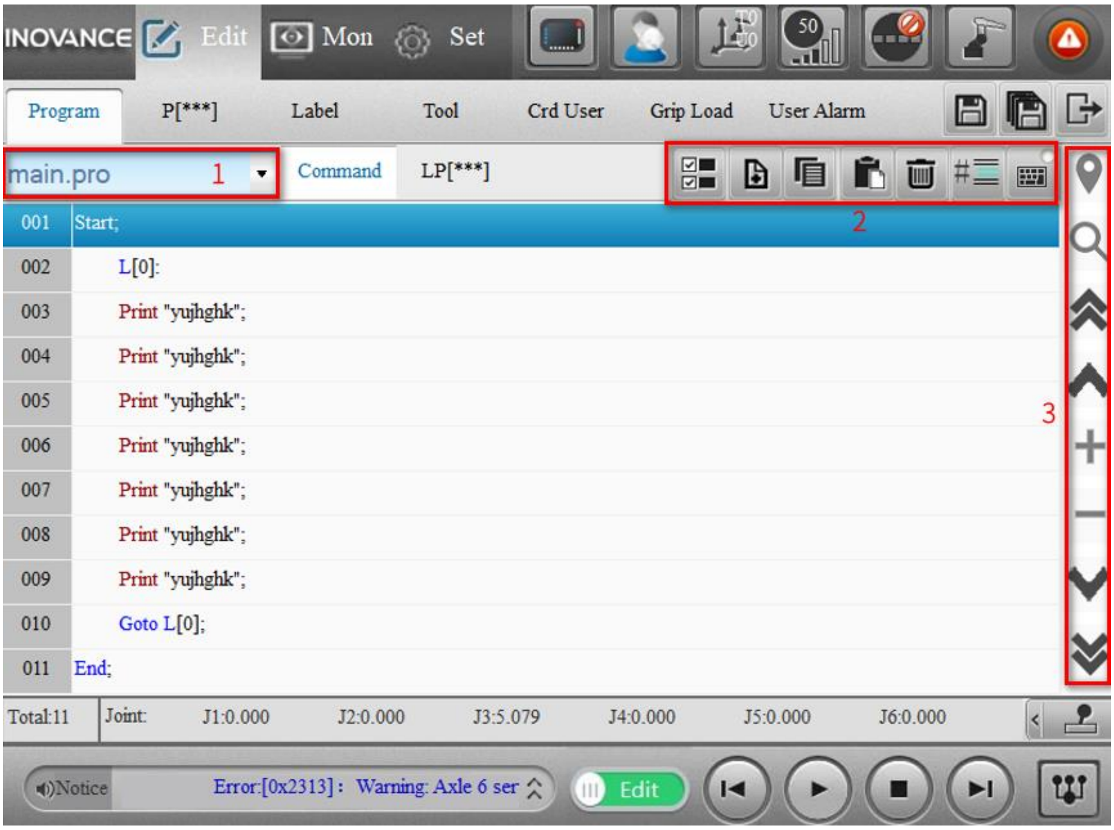
Back: Returns to the project file list page

Note: When you click **Back**, if the modified project is not saved, you will be prompted to save the modification.

When the robot is controlled by a device other than InoTeachPad, only **Refresh** and **Back** buttons are available in the upper right corner.

Refresh: Reloads the current project from the controller

(b) Editing the program



No.	Area	Function
1	Program selection area	The program drop-down list contains all the files in the project. Click **Command** to view the current program instructions. Click **LP[***]** to view the points corresponding to the current program.
2	Tool area	Edits the programs, points, etc.
3	Auxiliary operating area	Provides auxiliary functions such as page turning, quick positioning, etc.

(1) Selecting the program

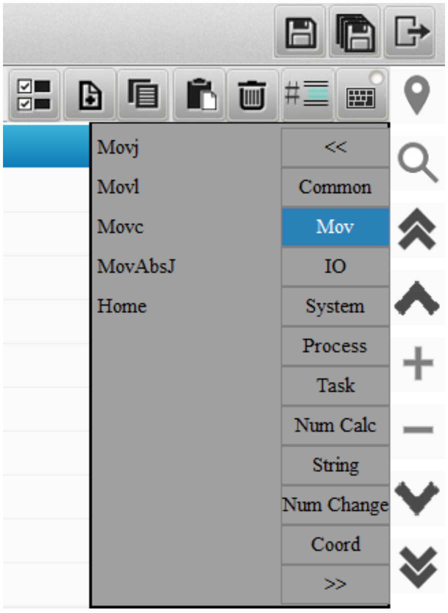
The drop-down list contains all the program files in the current project. Select a program file from the drop-down list.

(2) Editing the program

The functions shown above, from left to right, are:

Multi-select: Turns on or off the multi-select function, allowing the operation to take effect on multiple lines of instructions. You can select multiple non-consecutive lines.

New: When clicked, the following page pops up. Select the instruction.



Copy: Copies the instruction

Paste: Pastes the instruction

Delete: Deletes the instruction

Comment: Comments or uncomments the selected instruction

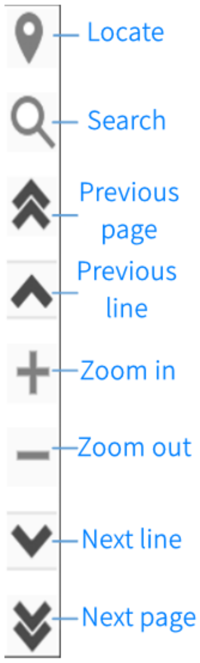
Keyboard: Turns on or off the keyboard editing mode. When turned on, when you double-click on an instruction line or add a new instruction, a full keyboard pops up.

When the robot is controlled by a device other than InoTeachPad, there is only one button here.

: Refreshes the project.

(3) Auxiliary operation

The auxiliary operation buttons include:



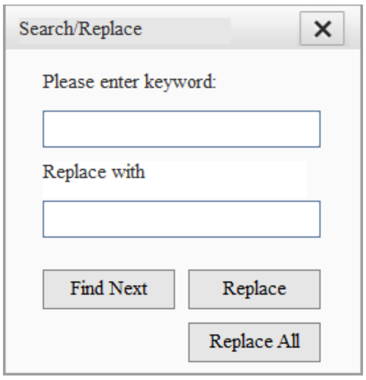
Zoom In/Out: Zooms in and out on the current program.

Locate: Jumps to the specified line of program after you enter the program line number in the pop-up numeric keypad.

Page Up/Page Down: Goes to the previous or next page of the program

Previous line/Next line: Goes to the previous or next line of the instruction

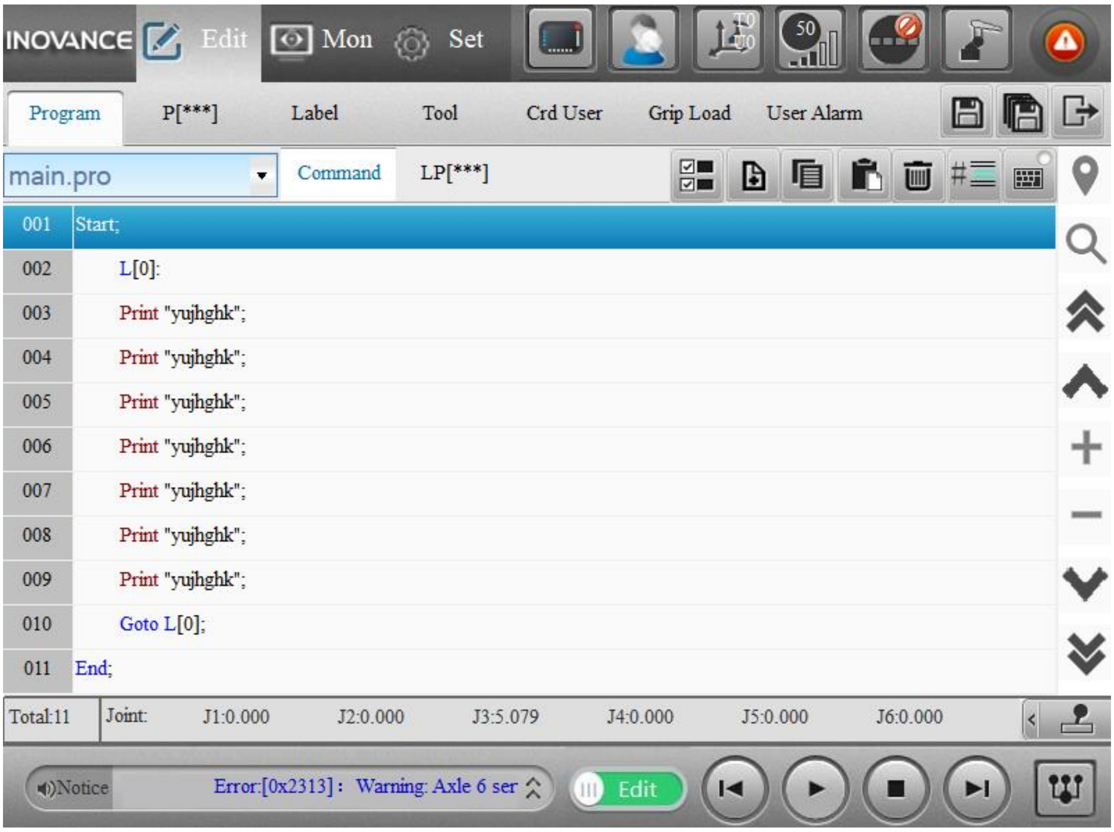
Find: Searches or replaces by keyword.



- Find next: Finds the next object from the current line.
- Replace: Replaces an object from the current line.
- Replace all: Replaces all objects from the current line.

(4) Editing the instruction

Double-click the instruction line and you can edit the instruction in the pop-up keyboard.

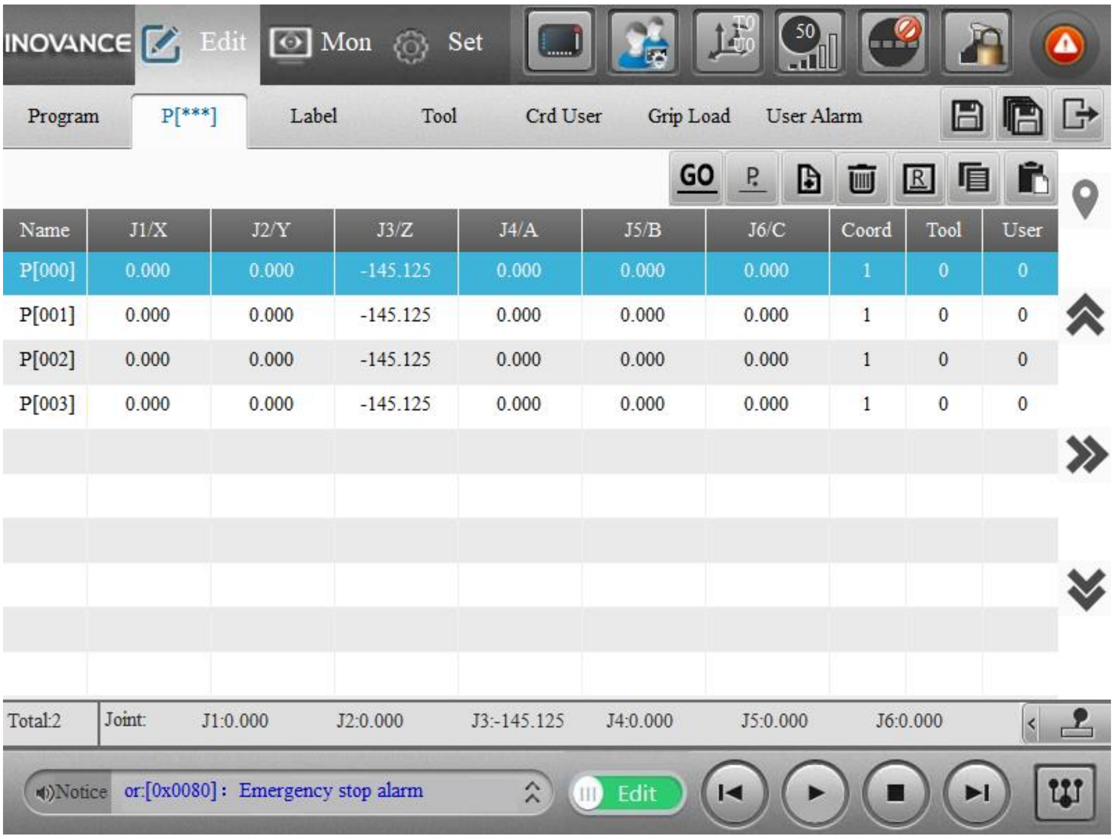


(c) Editing the points

The **LP[***]** page in the **Program** displays the local points, which are defined for the program file.

The **P[***]** page displays the global points, which are common to all programs in the project.

The local points LP can be edited in the same way as the global points P.



(1) Editing the points

As shown above, the functions from left to right are:

Replace: Replaces the selected point with the current point.

New: Adds a new LP, and the new point gets the current value.

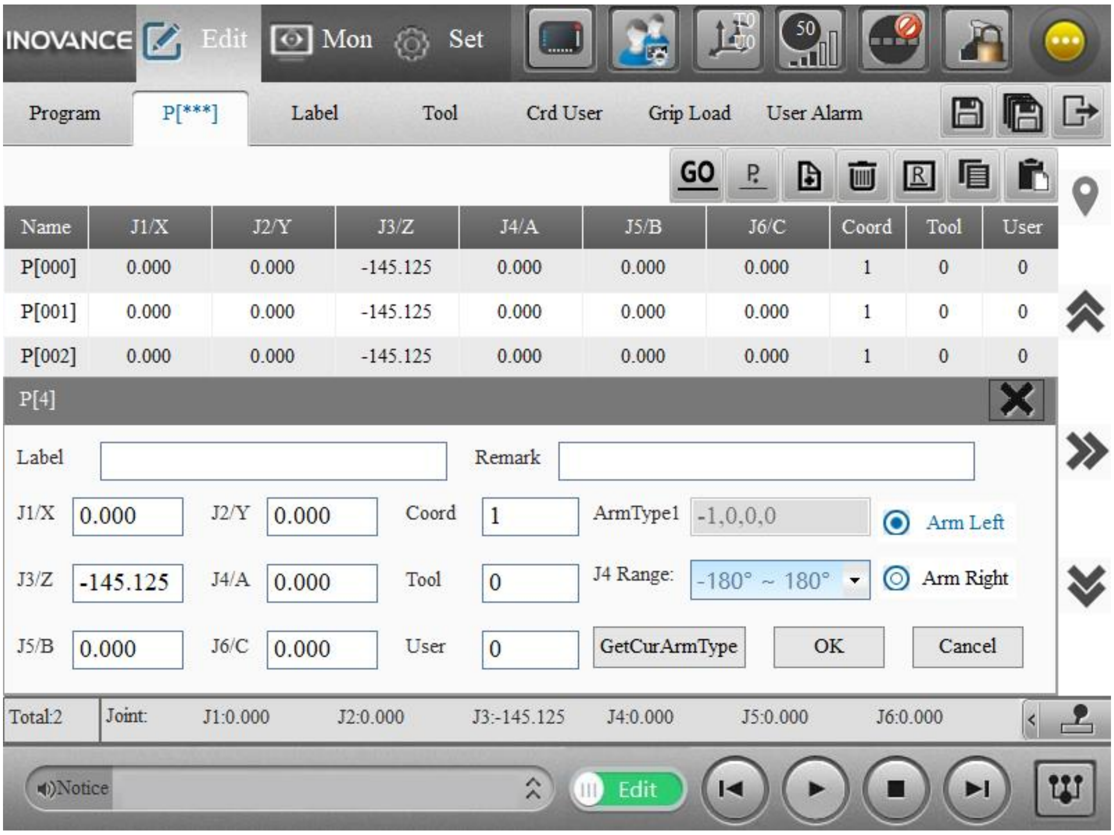
Delete: Deletes the selected point.

Rename: Renames the selected point (in the pop-up keyboard, enter the point number).

Copy: Copies the selected point.

Paste: Pastes the copied point.

To edit a point: Double-click an item in the point list and then edit the point in the pop-up edit page.



(2) Display of the points (page turning, locate, toggle display)

The vertical toolbar on the right shows the page turning, locate, and toggle display buttons.

 : Locates a point by clicking this button and entering the point number in the pop-up keyboard.

 : Toggles the display between the coordinate value and the label of the point.

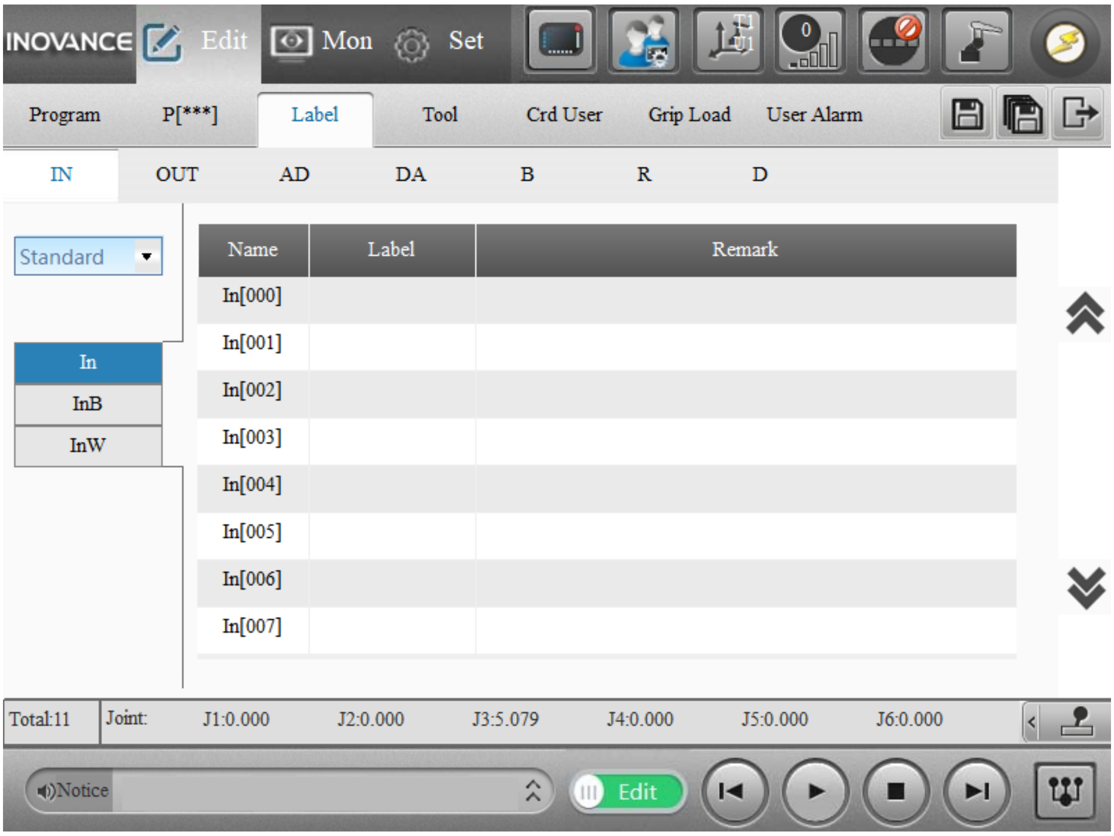
(3) Moving to a point

You can move to a point in two ways:

- Select a point in the list, turn on enable mode, click and hold . This is a Movj movement.
- Select a point in the list, turn on enable mode, click  and in the pop-up page click and hold **Execute**. This approach supports various ways to reach the point.

Note: Regardless of the way of reaching the point, the motion stops immediately while you release the button.

(d) Editing the label



The **Label** tab includes [IN], [OUT], [AD], [DA], [B], [R] and [D] pages. On the corresponding page, double-click the **Label** column or **Remark** column to edit the label or remarks.

Label: Up to 20 characters, starting with a letter and containing only letters, numbers, and underscores.

Note:

Labels cannot be duplicated, with the exception that duplicate LP labels are allowed in multiple program files.

Label cannot be the same as keyword, program name.

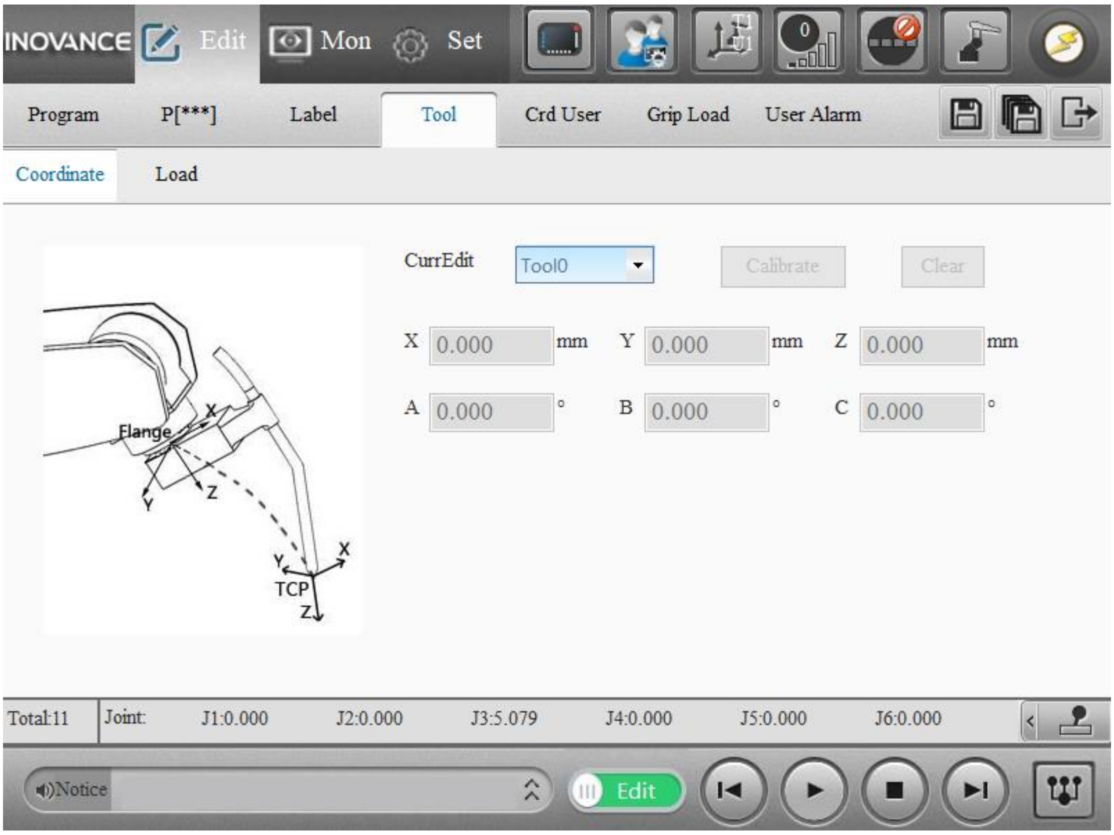
Remark: Up to 100 characters, Chinese or English, equal sign not allowed. If the remark contains Chinese characters, you can enter a maximum of 50 Chinese characters.

Other information:

For IN and OUT variables, they include standard I/O, fieldbus I/O, and memory I/O, see Section 5.3.1. By data type, they include Bit-, Byte- and Word-type I/Os, see section 5.3.2.

(E) Editing the tool

The **Tool** tab includes **Coordinate** and **Load** pages.



On the **Coordinate** page, select the tool you want to edit from the drop-down list. You can edit the coordinates directly.

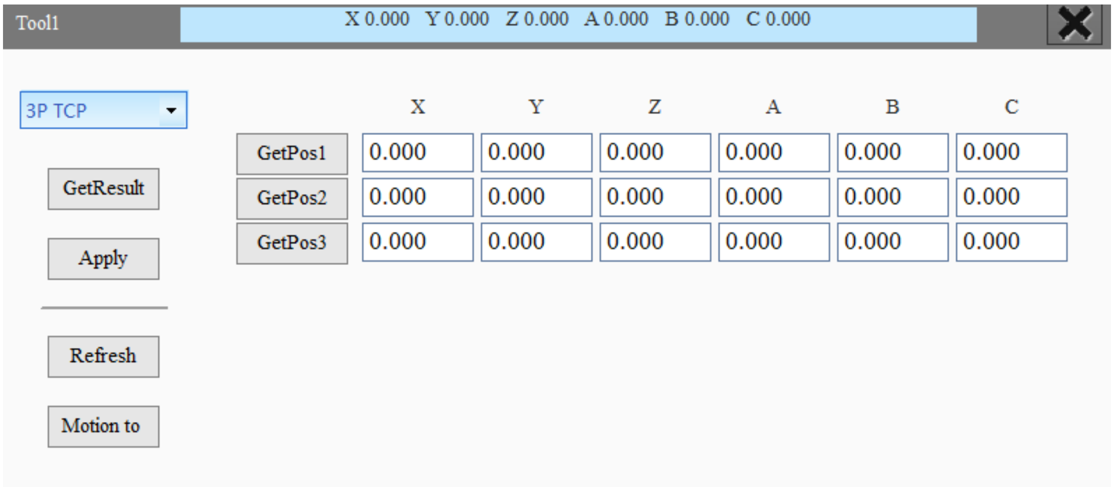
Note: Tool0 cannot be edited.

Calibrate: Calibrates the corresponding tool.

Clear: Clears the coordinate system values, while clearing the intermediate data.

Click **Calibrate** to go to the calibration page.

The calibration page displays the last calibration data, including calibration results, calibration method, calibration center point.



Description of the calibration interface:

Blue area: Displays the current coordinate system values.

Drop-down list: Selects calibration method. The possible calibration methods include direct method, 3-point TCP, 5-point TCP, 3-point TCP+ZX, and 5-point TCP+ZX. Different calibration methods present different pages (Note: Unsaved calibration intermediate data will be cleared each time a calibration method is selected).

GetPos: Gets the current TCP position in the base coordinate system. Each time a point is gotten successfully, a value is displayed and a check mark in front of the point button indicates that the point is complete.

GetResult: Generates a calculation based on the value of the points, which is displayed in the blue area above the page.

Apply: Applies the calibrated results and intermediate data. Note: The application results are not saved to the project and need to be saved separately.

Refresh: If you are not satisfied with the results generated, you can click the **Refresh** button to refresh the last saved calibration data as long as it is not saved.

Motion to: In the pop-up dialog, select the mode of motion, and move to the destination point by enabling the servo, clicking and hold the start button. To stop the motion, release the start button.

Description of calibration method:

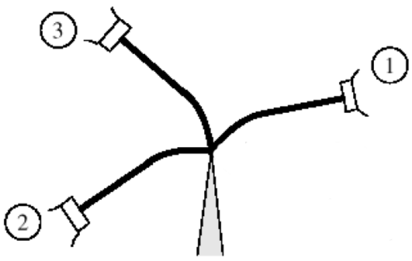
Method	Characteristics	Applicable Scenario
Direct	Directly enter coordinate system parameters.	Coordinate system parameters are known.
3-point TCP	Obtains position of the tool through three points.	Position values of the tool coordinate system need to be manually calibrated.
5-point TCP	Obtains position of the tool through five points.	Position values of the tool coordinate system need to be manually calibrated. It is more accurate than 3-point TCP because more points are used.
3-point TCP+ZX	Obtains position of the tool through three points and pose of the tool through three additional points.	Position and orientation values of the tool coordinate system need to be manually calibrated.
5-point TCP+ZX	Obtains position of the tool through five points and orientation of the tool through three additional points.	Position and orientation values of the tool coordinate system need to be manually calibrated. It is more accurate than 3-point TCP+ZX because more points are used.

Direct entry:

Enter the values of the tool coordinate system directly, click **Apply**, and then click the **Save** or **Save All** button.

3-point TCP

After installing a tool at the end of the robot, adjust the pose of the tool to align the TCP with one point in the space in three different directions. Then click the **GetPos** button to record the point values. After the alignment of three points have been finished, click **GetResult** to obtain tool coordinate system parameters.



3-point TCP

Operation steps:

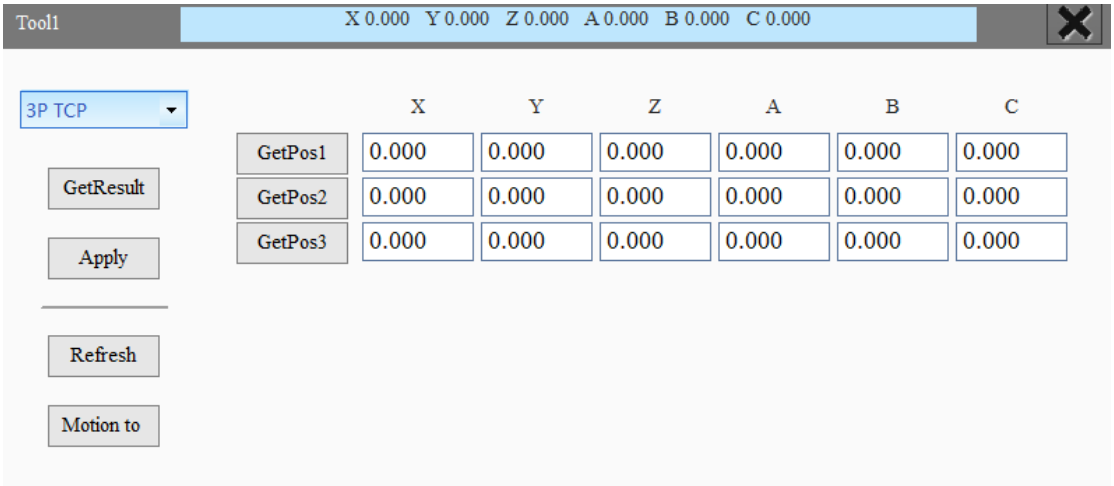
Step 1: Control the motion of robot so that the TCP is aligned with the reference point in Direction 1, and then click **GetPos1**.

Step 2: Control the motion of robot so that the TCP is aligned with the reference point in Direction 2, and then click **GetPos2**.

Step 3: Control the motion of robot so that the TCP is aligned with the reference point in Direction 3, and then click **GetPos3**.

Step 4: Click **GetResult**, and tool coordinate system parameters are automatically generated.

Step 5: Click **Apply**, and then click the **Save** or **Save All** button.



Note:

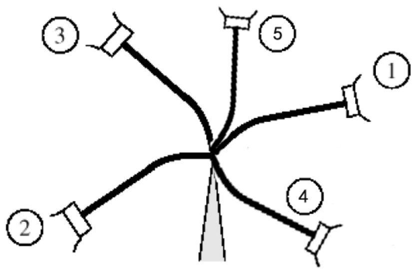
The interval between the three different poses selected for the three-point TCP method and the five-point TCP method should be as large as possible. An interval of over 20 degrees is recommended.

After you click **GetResult**, error parameters are displayed. It is generally deemed to be accurate that the maximum error is smaller than the absolute precision of the robot body+0.1mm. When the maximum error is too large, it is recommended to get points to generate tool coordinate system parameters again.

When the tool end of the SCARA and Delta robots is coaxial with the end axis (i.e. there is only a Z value), this Z position parameter cannot be obtained via the 3-point or 5-point TCP method.

5-point TCP

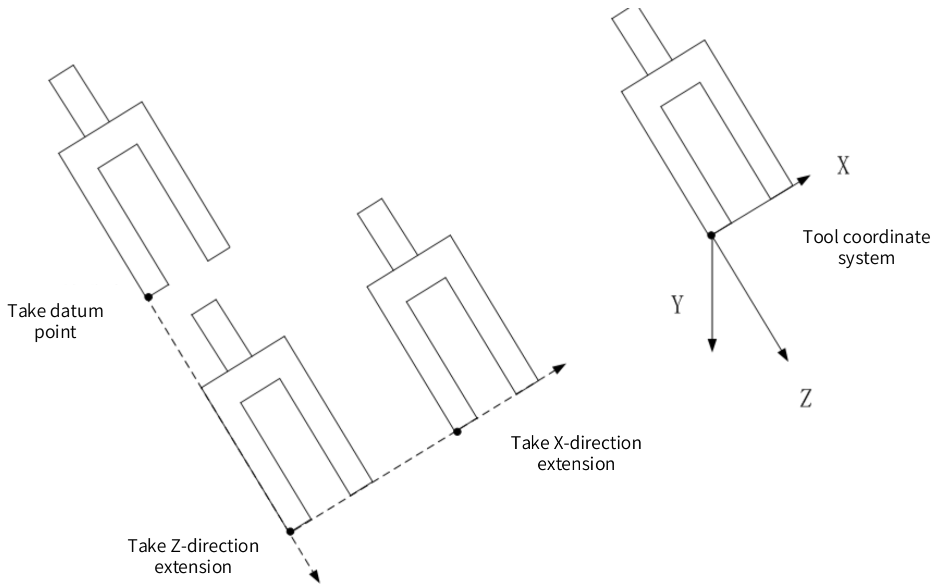
Align the TCP with a reference point in the space in five different directions, similar to the three-point TCP method.



5-point TCP

3-point TCP+ZX

Get three more points to calibrate poses of the tool coordinate system on the basis of 3-point TCP method. The Z extended point and the reference point compose the Z direction of the tool coordinate system. The X and Z extended points compose the X direction of the tool coordinate system. The Y direction can be obtained from the Z and X directions by the right-hand rule. See the following figure.



The operations are as follows:

Step 1: Control the motion of robot so that the TCP is aligned with the reference point in Direction 1, and then click **GetPos1**.

Step 2: Control the motion of robot so that the TCP is aligned with the reference point in Direction 2, and then click **GetPos2**.

Step 3: Control the motion of robot so that the TCP is aligned with the reference point in Direction 3, and then click **GetPos3**.

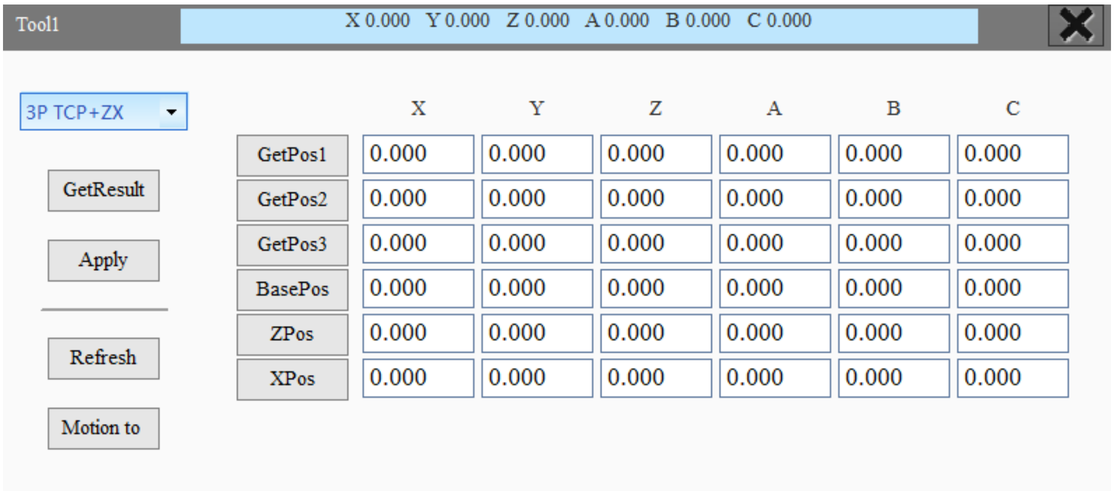
Step 4: Control robot motion. Select the TCP at any moment as the reference point. Click **BasePos**.

Step 5: Control the motion of robot so that the tool extends a distance in Z direction, and then click **ZPos**.

Step 6: Control the motion of robot so that the tool extends a distance in X direction, and then click **XPos**.

Step 7: Click **GetResult**, and tool coordinate system parameters are automatically generated.

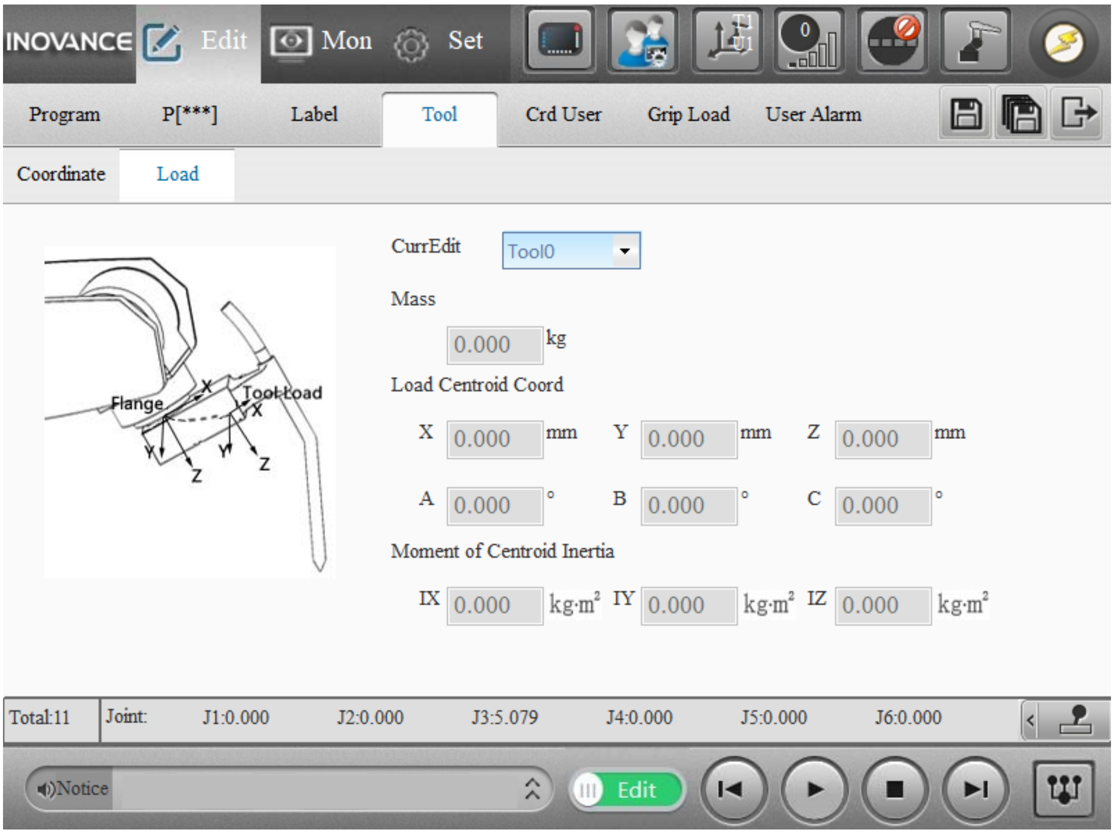
Step 8: Click **Apply**, and then click the **Save** or **Save All** button.



5-point TCP+ZX

Align the TCP with a reference point in the space in five different directions, similar to the three-point TCP+ZX method.

On the **Load** page, modify the parameters.



Note:

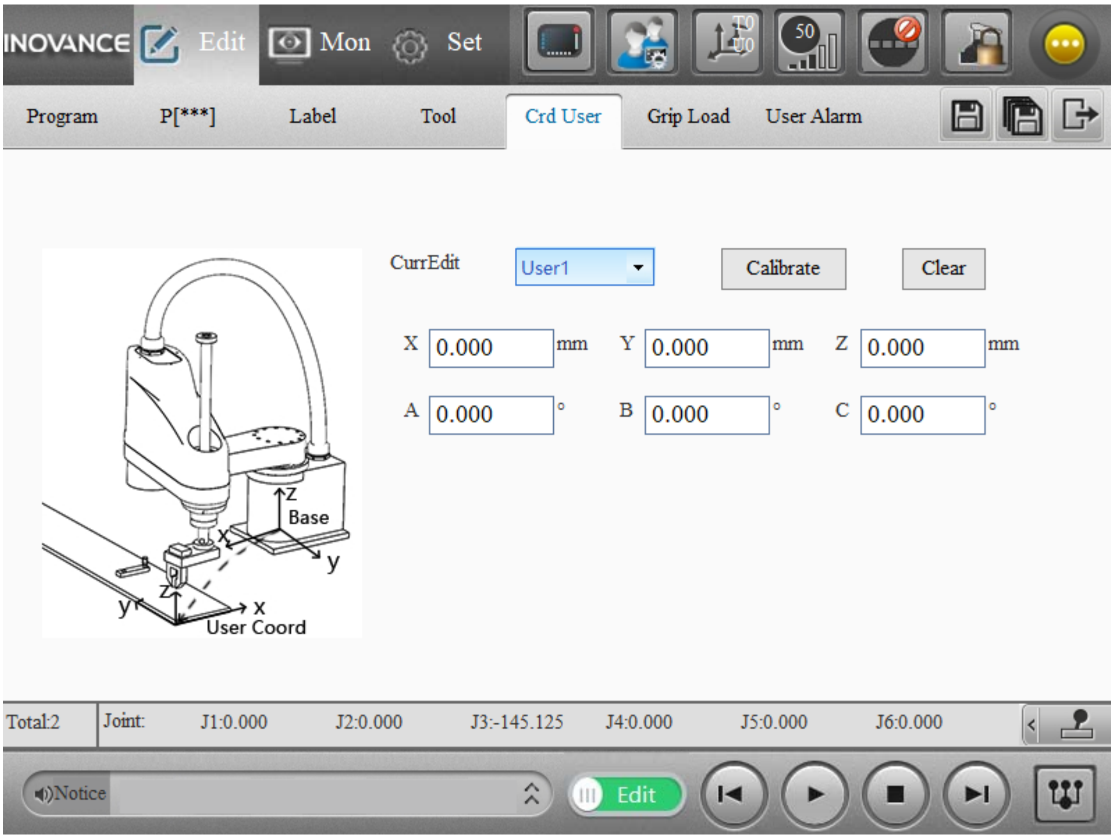
Tool0 cannot be edited.

For SCARA robots, parameters A, B, C, IX and IY cannot be edited. For 6-axis robots, all parameters can be edited.

Make sure all the load parameters are correctly set; otherwise, collision detection false alarms, too long or short cycle time, or abnormal current may occur.

(f) Introduction to the user coordinate system page

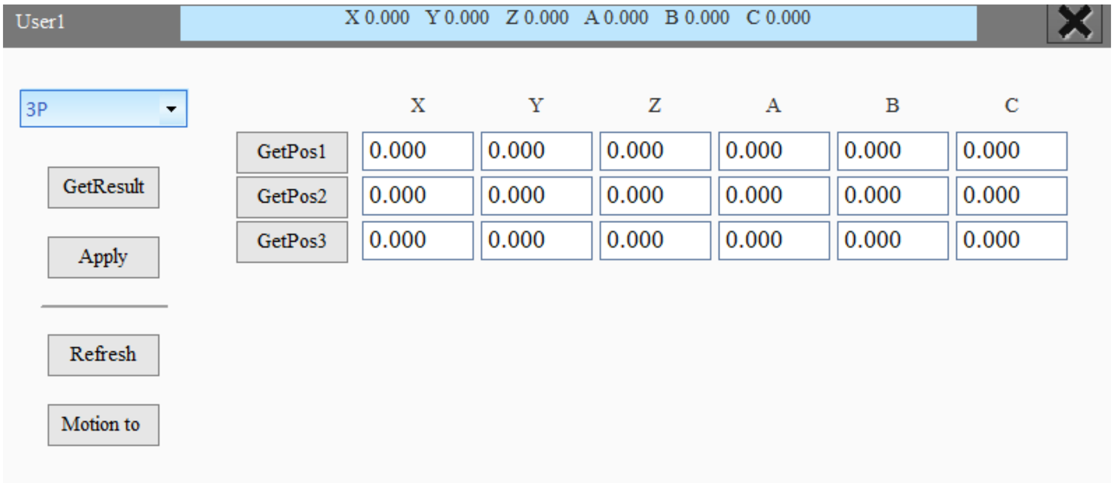
On the **Crd User** page, select the user coordinate system from the drop-down list so that you can view the coordinate parameters. You can edit the coordinates directly.



Note: User0 cannot be edited.

Click **Calibrate** to go to the calibration page.

The calibration page displays the last calibration data, including calibration results, calibration method, calibration center point.



Description of the calibration interface:

Blue area: Displays the current coordinate system values.

Drop-down list: Selects calibration method. The calibration methods include direct method, 3-point TCP, and rotation method. Different calibration methods present different pages (Note:

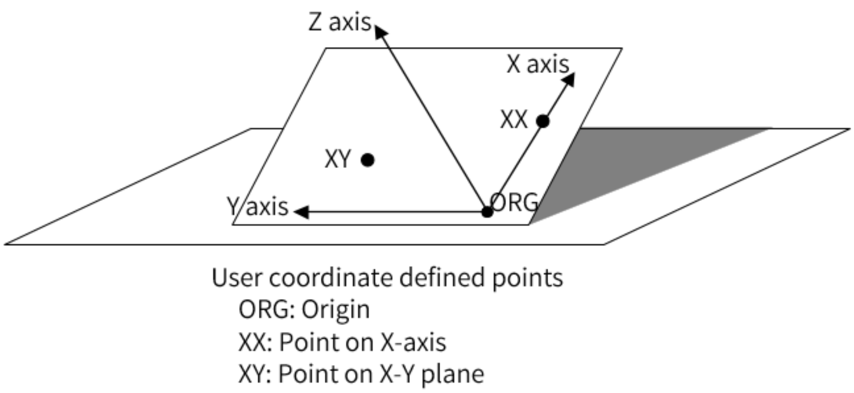
Unsaved calibration intermediate data will be cleared each time a calibration method is selected).
GetPos: Get the current TCP position in the base coordinate system. Each time a point is gotten successfully, a value is displayed and a check mark in front of the point button indicates that the point is complete.
GetResult: Generates a calculation based on the value of the points, which is displayed in the blue area above the page.
Apply: Applies the calibrated results and intermediate data. Note: The application results are not saved to the project and need to be saved separately.
Refresh: If you are not satisfied with the results generated, you can click the **Refresh** button to refresh the last saved calibration data as long as it is not saved.
Motion to: In the pop-up dialog, select the mode of motion, and move to the destination point by enabling the servo, clicking and hold the start button. To stop the motion, release the start button.

Description of calibration method:

Method	Characteristics	Applicable Scenario
Direct	Directly enter coordinate system parameters.	Coordinate system parameters are known.
3-point TCP	Calibrates the user coordinate system using three points.	Values of the user coordinate system need to be manually calibrated.
Rotate	Make marks on the turnplate. Rotate the turnplate and get points for teaching.	A turnplate is provided and the user coordinate system is located at the center of the turnplate.

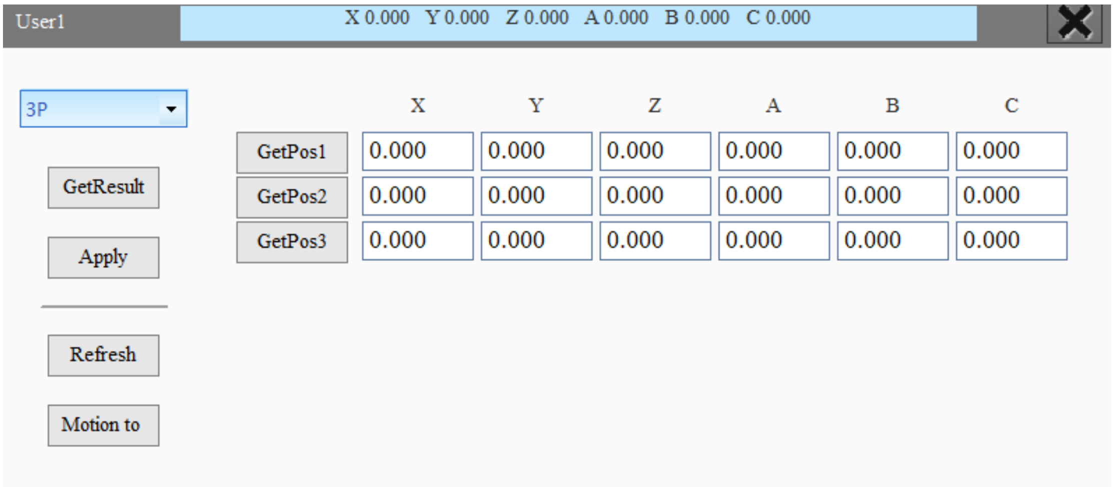
Direct method:
Enter the values of the tool coordinate system directly, click **Apply**, and then click the **Save** or **Save All** button.

3-point TCP:



Step 1: Get the origin of the user coordinate system and click **GetPos1**.
Step 2: Take a point in the positive direction of the X-axis of the user coordinate system and click **GetPos2**.
Step 3: Take a point in the Y+ direction on the XY plane of the user coordinate system and click **GetPos3**.
Step 4: Click **GetResult**, and tool coordinate system parameters are automatically generated.

Step 5: Click **Apply**, and then click the **Save** or **Save All** button.

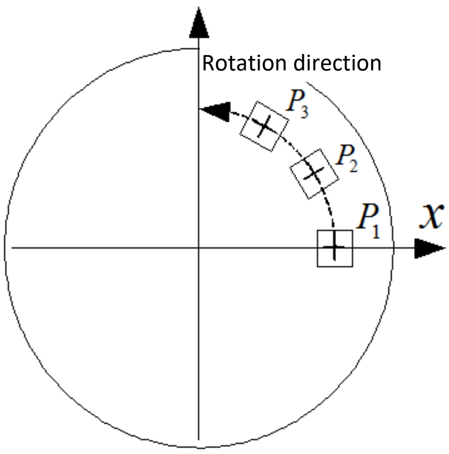


Note:

1.The XY point is not directly on the Y-axis, but in the Y+ direction of the XY plane. Thus, the XY point indirectly defines the Y-axis, and the final Z-axis is obtained by the right-hand rule.

2. For SCARA and Delta robots, if Z direction of the user coordinate system has a negative component in the Z direction of the base coordinate system, the system will automatically reverse Z and Y directions of the user coordinate system, with the X direction remaining unchanged.

Rotate:



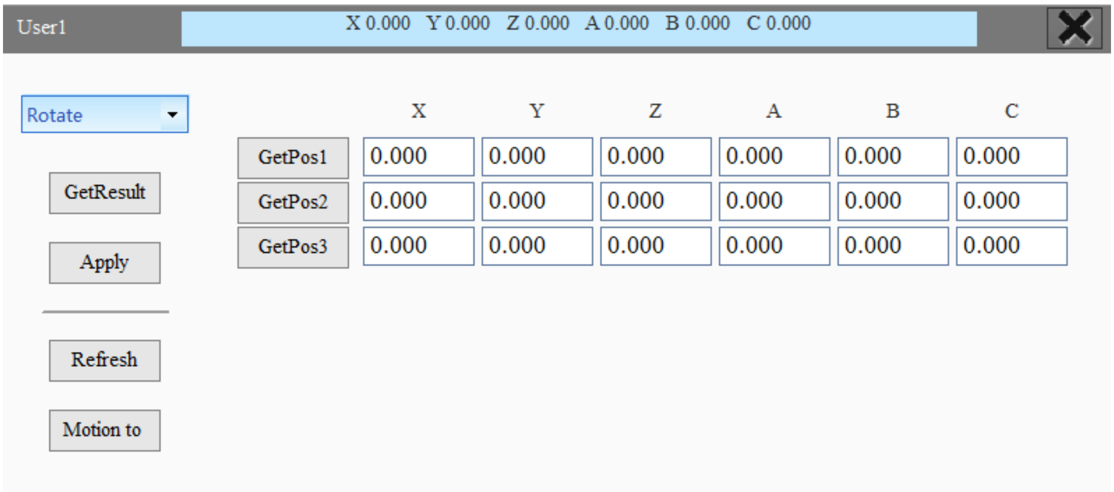
Step 1: Mark a fixed point on the turnplate, align the TCP with the marked point and then click **GetPos1**.

Step 2: Rotate the turnplate by an angle, align the TCP with the marked point again and then click **GetPos2**.

Step 3: Rotate the turnplate by an angle, align the TCP with the marked point again and then click **GetPos3**.

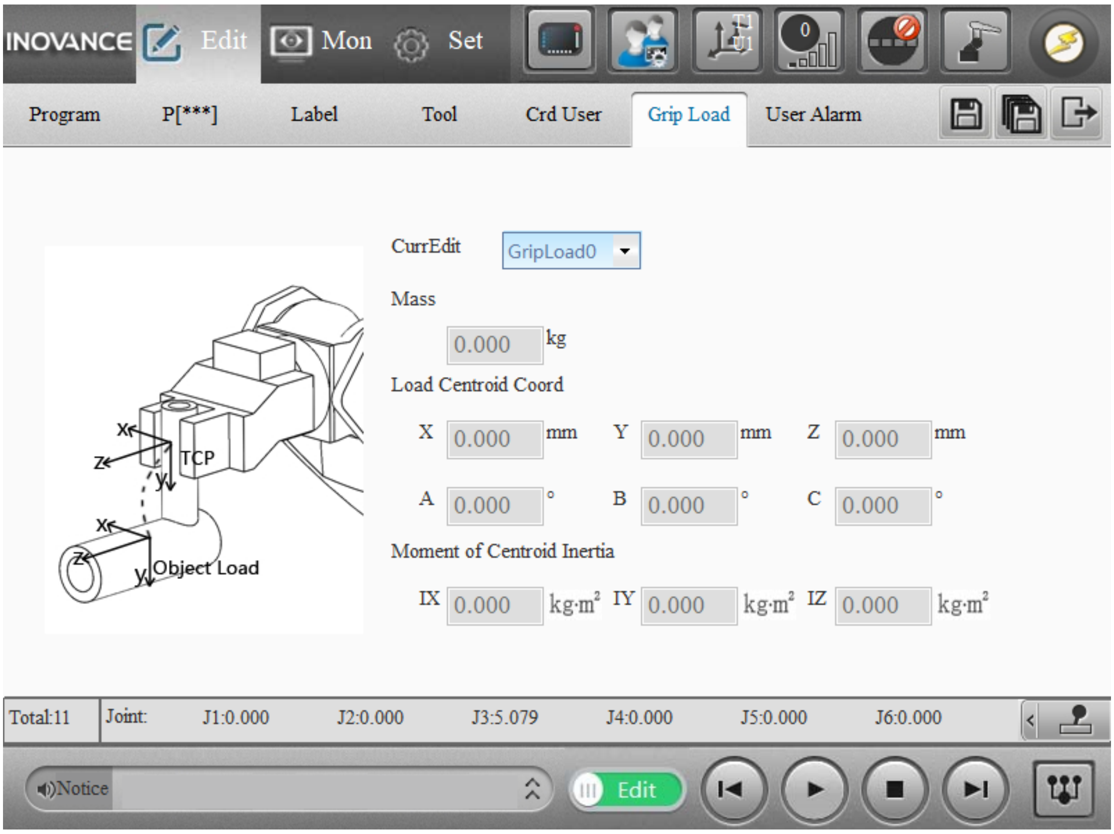
Step 4: Click **GetResult**, and user coordinate system parameters are automatically generated.

Step 5: Click **Apply**, and then click the **Save** or **Save All** button.



(g) Editing the grip load

On the **Grip Load** page, select the grip load from the drop-down list. You can edit the parameters directly.



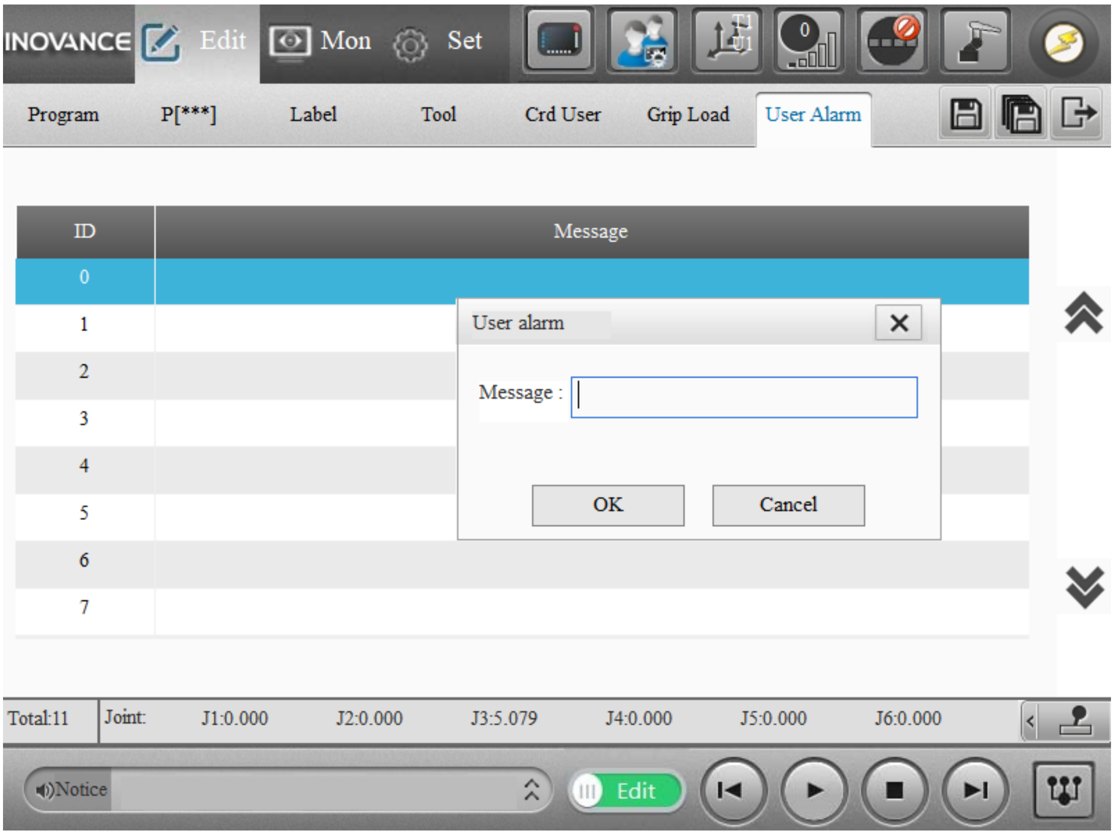
Note: GripLoad0 cannot be edited.

For SCARA robots, parameters A, B, C, IX and IY cannot be edited. For 6-axis robots, all parameters can be edited.

Make sure all the load parameters are correctly set; otherwise, collision detection false alarms, too long or short cycle time, or abnormal current may occur.

(H) Editing the user alarms

On the **User Alarm** page, double-click an item to modify the alarm message.



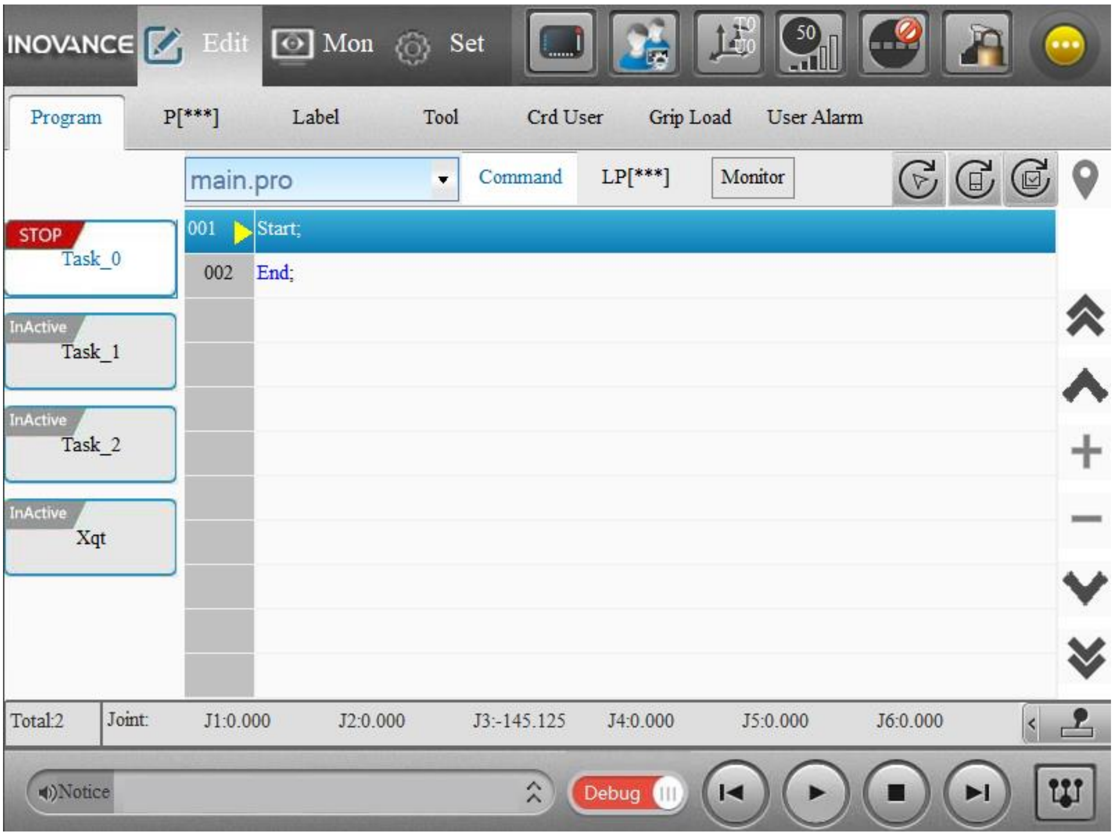
Note that the message bar displays the detailed alarm messages in the current project, while in the logs the contents of the messages will not be displayed.

3.4 Debugging

You can slide the **Edit** button to **Debug** to switch from the programming interface to the debugging interface.

indicates debug mode.

Note: When switching to the debug mode, if you are prompted to save the project, you need to click the **Save All** button before switching to the play mode.



In debug mode, you will be able to run the project. The status of the tasks on the page is refreshed in real time as the tasks run. The execution of the program under the currently viewed task is refreshed in real time. The program file name in the drop-down list, the program displayed on the page, and the program line number are refreshed automatically.

Task tab: By toggling the task tab, the programs under the different tasks are displayed on the right.

Task_0: Main task

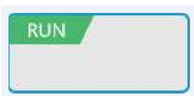
Task_1: Multi-tasking, either dynamic or static

Task_2: Multi-tasking, either dynamic or static

Task_3: Multi-tasking, xqt task (The status and status line of the xqt task cannot be set.)

The motion status is indicated on the task tab:

: Stopped

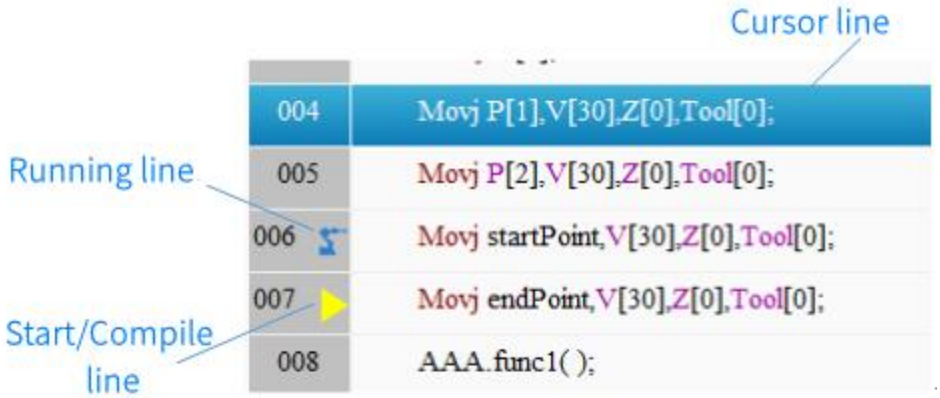


: Running

: Inactive (including idle)

: Finished

Program lines:



Running line: The line of instructions being executed, representing the current position of the robot.
Start/Compiling line: A pre-compiling line of instructions, which is always ahead of the running line. You can set the start line of the program.
Cursor line: The line in which the cursor is located.

Setting the start line:

: Sets the current cursor line as the start line.

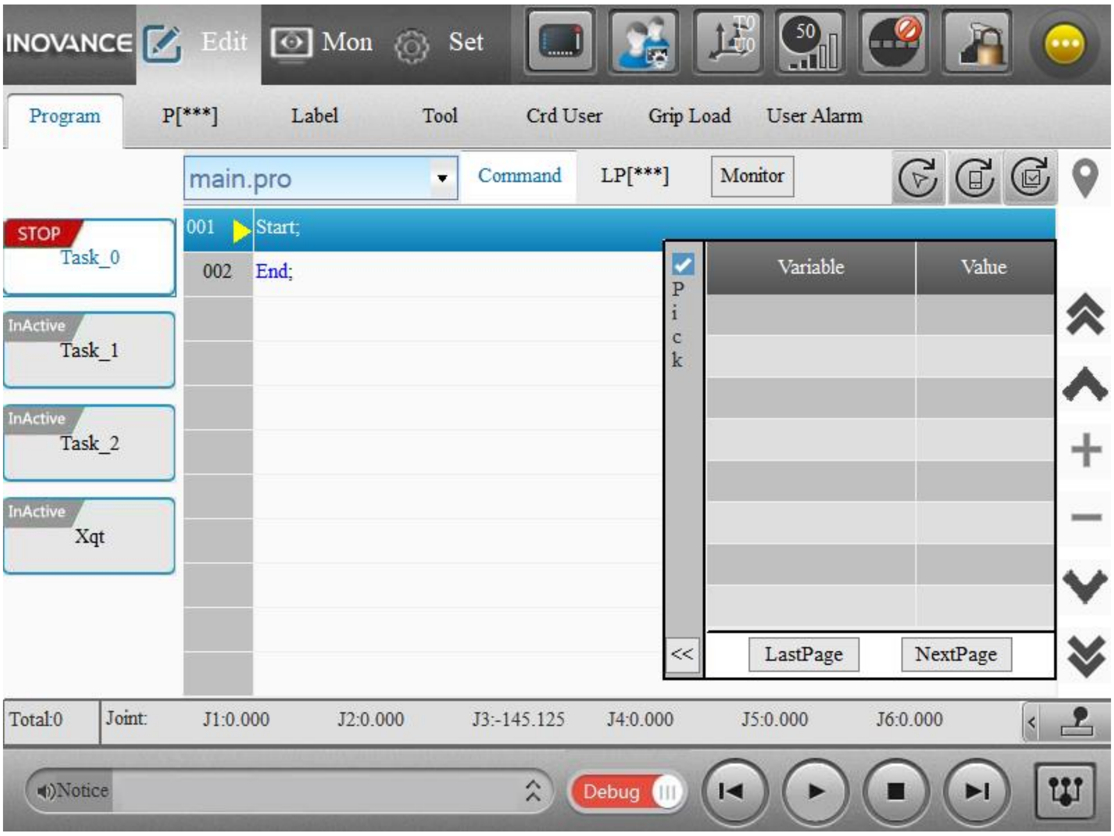
: The current task returns to the start line, that is, the start line of the current task is reset to the start line of the entry program.

: All tasks return to the start line, that is, the start line of all tasks is reset to the start line of the entry program.
Note: In debug mode, you can navigate through the program using the Page Up/Down, Zoom In/out, and Locate buttons to help set the start line of the task.

Viewing instructions and points: Select the **Command** tab to view the program; select **[LP***]** to view the position points (You can view the values defined in the currently selected program file, not the real-time memory value!)

Start in debug mode: Enable the servo, click and hold the start button. Release the start button to stop immediately.

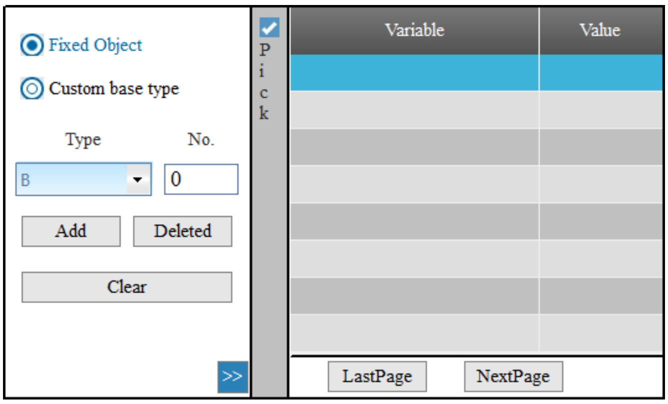
Variable monitoring in debug mode:
In the debug mode, you can access the quick monitor panel by clicking the **Monitor** button.



In the monitor panel, you can monitor the corresponding values of variables in the current start line of the current task.

Pick: When checked, click on a line of the program to monitor variables in that line.
Note:
You can only monitor the line you currently select, and cannot monitor the line previously selected.
For the custom struct variables, you cannot pick them for monitoring.

You can expand or collapse the monitor panel. On the expanded panel, you can add or remove the monitor object.

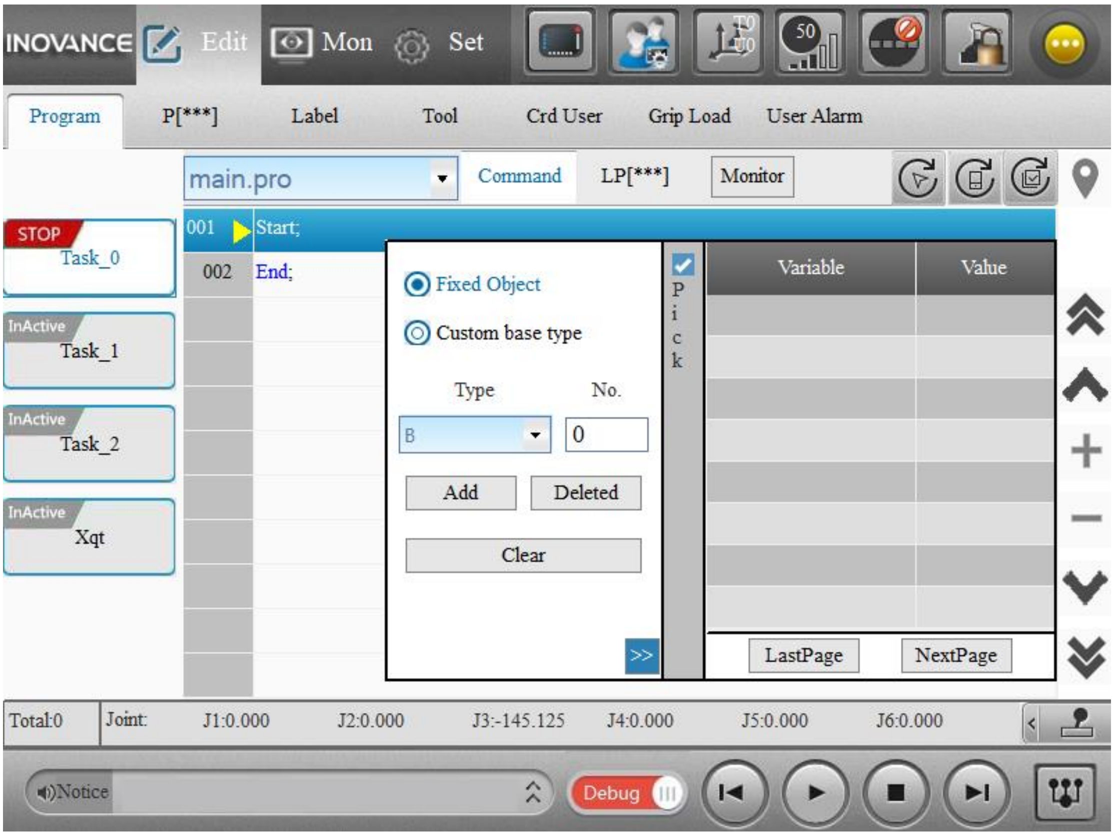


- Monitoring of fixed objects: If you want to monitor fixed objects including B/R/D/LB/LR/LD/P/LP/PR/LPR/IN/OUT/Tool/User variables, select **Fixed Object**. Then select the variable type and enter the variable number. For example, if you set **Type** to "B" and enter "1” to **No.**, then the monitored object is "B[1]".

- Monitoring of custom objects: If you want to monitor custom objects, such as Bool/Int/Byte/float/double data type, select **Custom base type**. Then enter the object name directly in the **Name** field.

Click **Add** to add the object to monitoring task. At most 10 variables can be added (Note that a composite variable such as a P variable consisting of multiple sub-variables such as P[1].Data[0] is considered as one variable.) When you have multiple pages of data, you can navigate through the pages via **LastPage** and **NextPage** buttons.

Note:
Scope of monitored variables:
B, R, d, P, PR, Str, custom global variables
LB, LR, LD, LP, LPR, custom local string variable
System variables such as Tool, User, IN, OUT, IG, InB, InW, OutB, OutW
As an example, the following describes how to monitor the I/O variables.
In the debug mode, click **Monitor**, select **Custom base type**, enter the variable name, click **Add** and then the variable is automatically displayed in the list of monitored objects. The supported variable formats include
In[X], InB[X], InB[X].Int, InB[X].Float, InB[X].Double, InW[X], InW[X].Int, InW[X].Float, InW[X].Double, as shown in the following figure.



Custom structs cannot be picked for monitoring. You can only monitor member variables in the custom struct through the **Custom base type** option.

2. Only when the program is not running can you select a program line to set the monitor objects.

3. Automatic clearing of monitoring data:
When you switch to the play mode or switch the task or program, the monitor panel closes and the monitoring data are cleared.

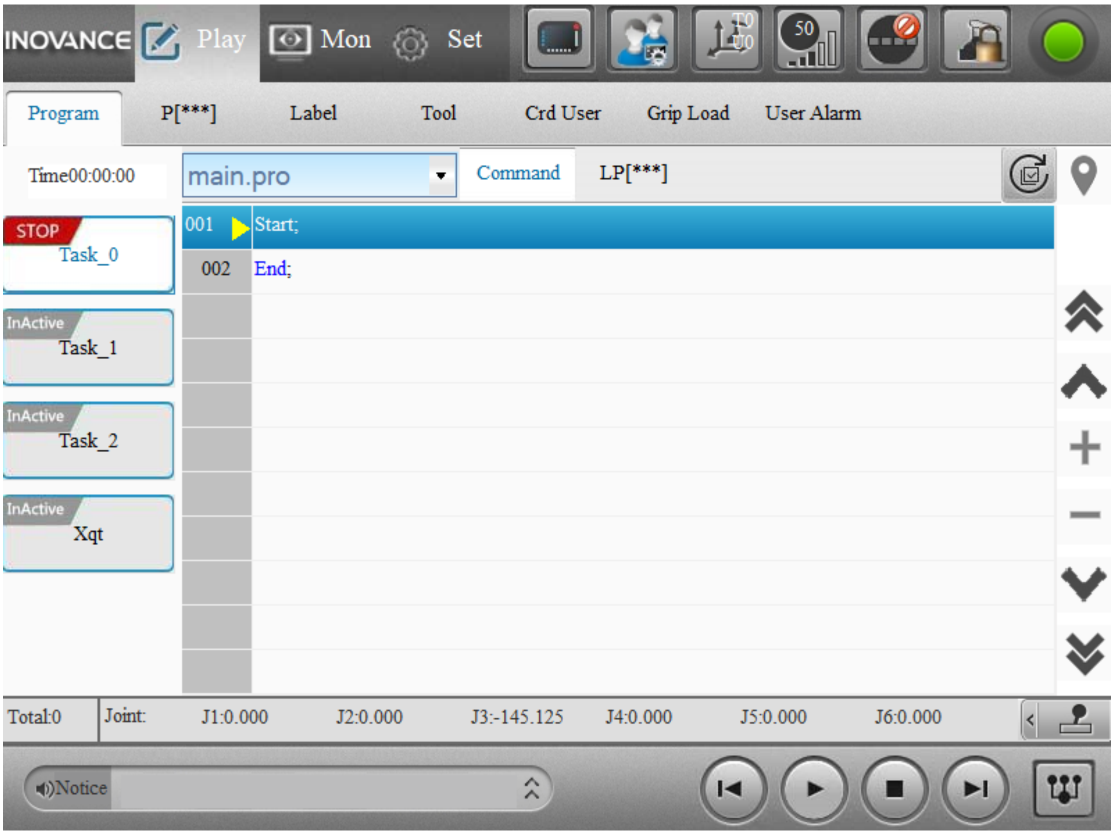
Modifying the value of the monitored variables:
To modify the value of a global variable or a local variable in the monitor list, double click the variable. Note that both global and local variables can be modified, but system variables cannot be modified.

3.5 Play Mode

When you switch to the play mode, the system is automatically enabled.
Note:
When switching to the play mode, if you are prompted to save the project, you need to click the **Save All** button before switching to the play mode.
For different types of teach pendant, the mode switch button is located in different positions.

For IRTP80 teach pendant, you can switch the mode using .

For the PC-based teach pendant, you can switch the mode using .
The following figure shows the Play interface.



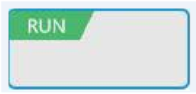
The Play interface is basically the same as the Debug interface.

Task tab: By toggling the task tab, the programs under the different tasks are displayed on the

right.
Task_0: Main task
Task_1: Multi-tasking, either dynamic or static
Task_2: Multi-tasking, either dynamic or static
Task_3: Multi-tasking, xqt task (The status and status line of the xqt task cannot be set)

The task status is indicated on the task tab:

: Stopped

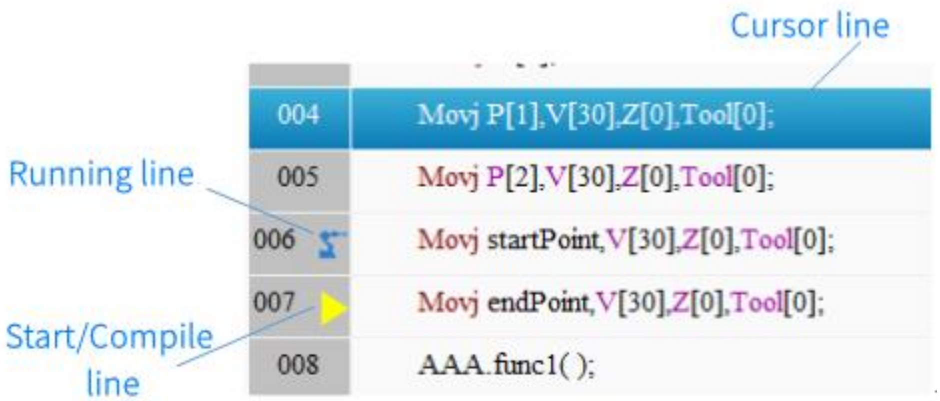


: Running

: Inactive (not activated, or idle)

: Finished

Program lines:



Running line: The line of instructions being executed, representing the current position of the robot.
Start/Compiling line: A pre-compiling line of instructions, which is always ahead of the running line. You can set the start line of the program.
Cursor line: The line in which the cursor is located.

Setting the start line:


: All tasks return to the start line, that is, the start line of all tasks is reset to the start line of the entry program.
Note: In the play mode, you are only allowed to return all tasks to the start line.

Viewing instructions and points: Select the **Command** tab to view the program; select **[LP***]** to view the points (You can view the values defined in the currently selected program file, not the real-time memory value!)

Start in play mode: Click the start button and the program keeps running until the stop button is clicked.

In play mode, the run time and safety door status are also displayed.
Run time: Time from start to stop of the main task.
Safety door: When the safety door is activated, if the safety door is open, a red light indicates that the safety door has been opened.

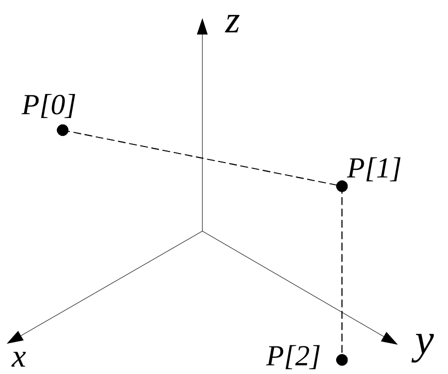
3.6 Viewing Project Under Other Controls

When the robot is controlled by a device other than InoTeachPad, you can only view the project, but cannot edit the project.
You can observe the execution of the program on the InoTeachPad. However, if the project has been modified by other control devices, the modification will not be automatically refreshed and you must manually refresh it using the refresh button.

Note:
The values on the programming interface are values defined in the project file, not the current memory values. The current memory values need to be viewed on the monitoring interface. When you want to see the results of the instructions "LP=XX" "P=XX", you need to go to the monitoring interface. However, if the P and LP values are modified via Modbus or API, the values defined in the project file and the current memory values will be modified synchronously. Therefore, you can view the values in either the programming or the monitoring interface.

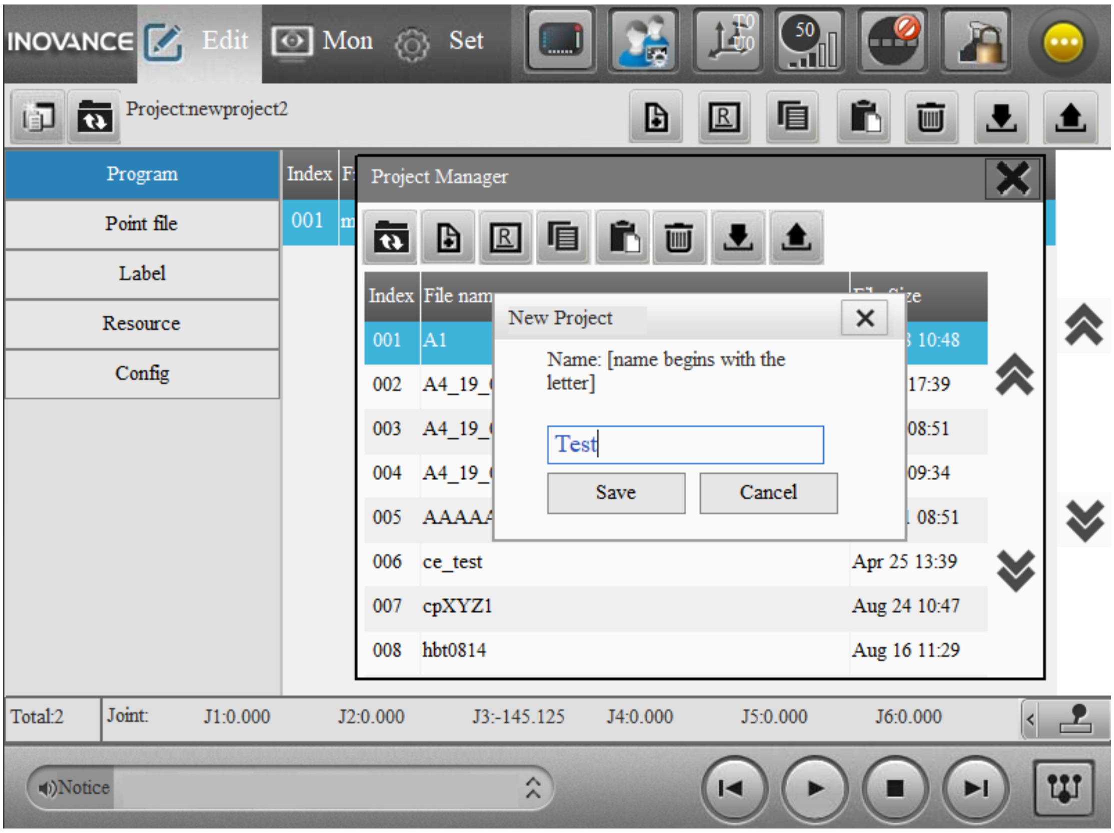
3.7 Example: Programming and Running a Project

Task: Create a new project "Test", edit the motion P[0]-P[1]-P[2] in the default program "main.pro". After editing the program, test it and switch to the play mode to run the project.

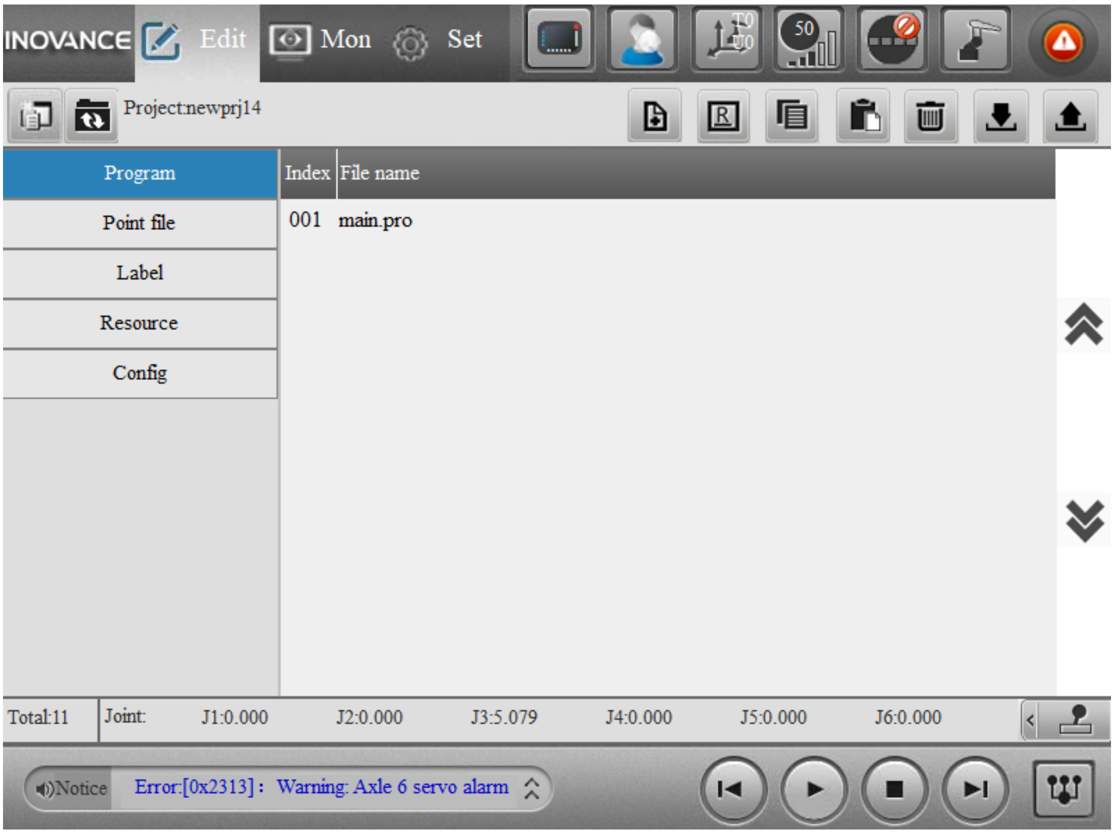


The detailed operations are as follows:

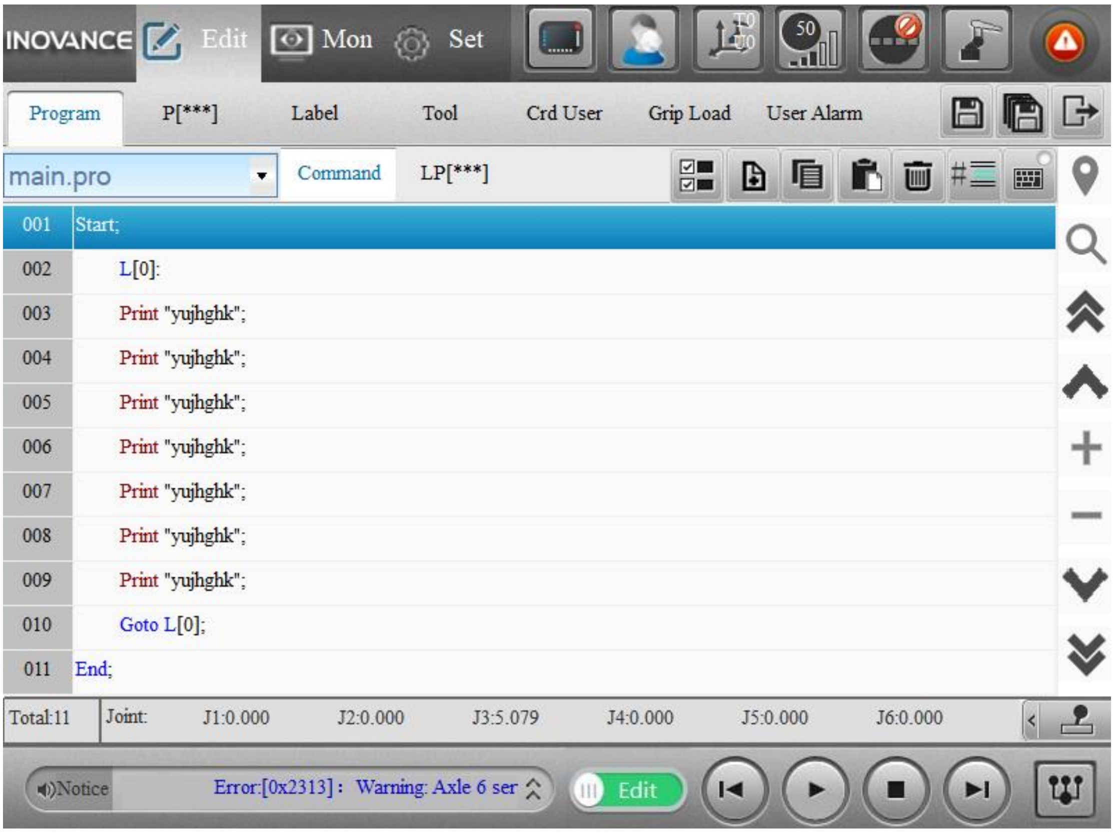
1. Create a new project "Test" in the project manager.



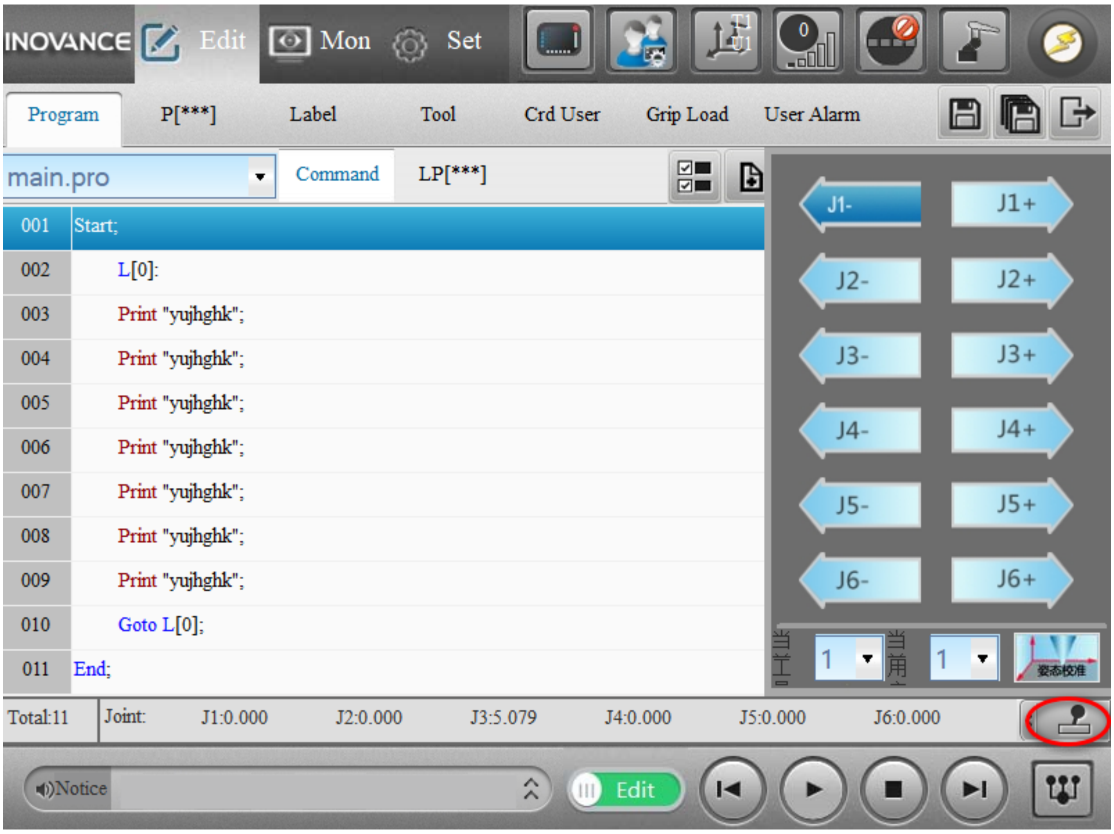
2. Double-click the project "Test" to activate it. The project includes program "main.pro" by default.



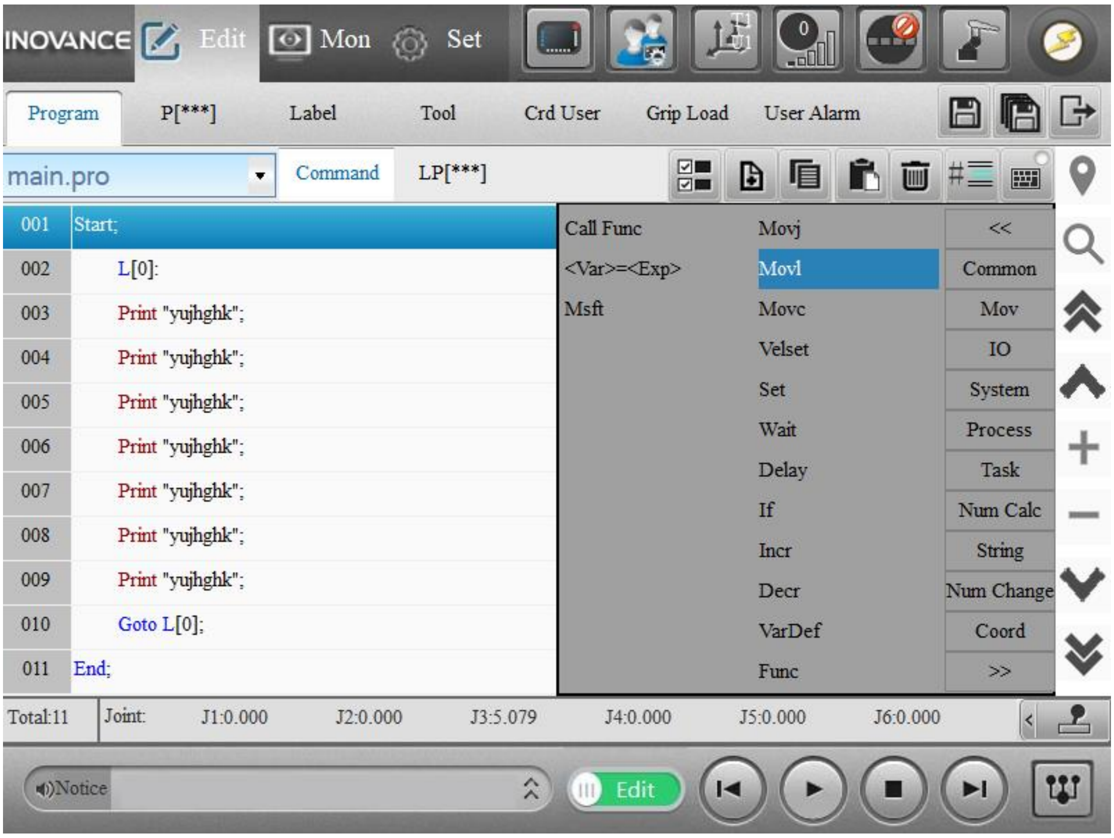
3. Double-click the program "main.pro" to access the editing interface.

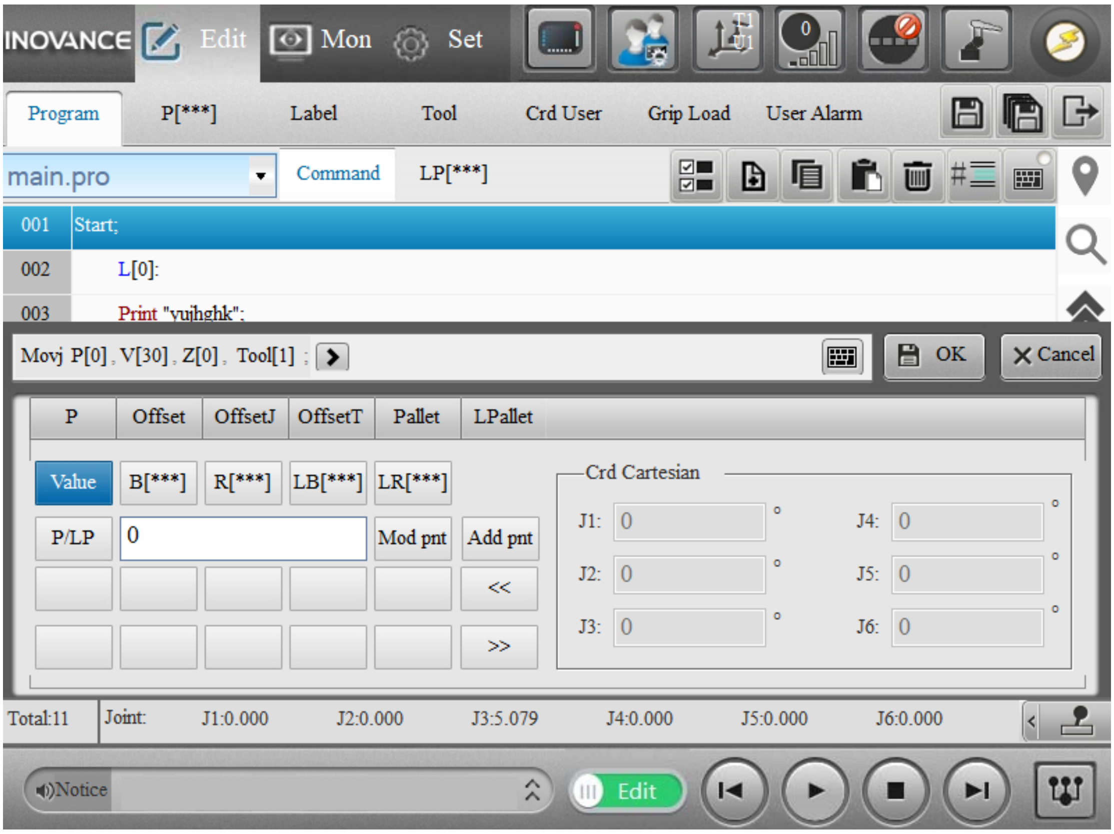


4. Open the teach panel, select an appropriate coordinate system, and move to point P[0].

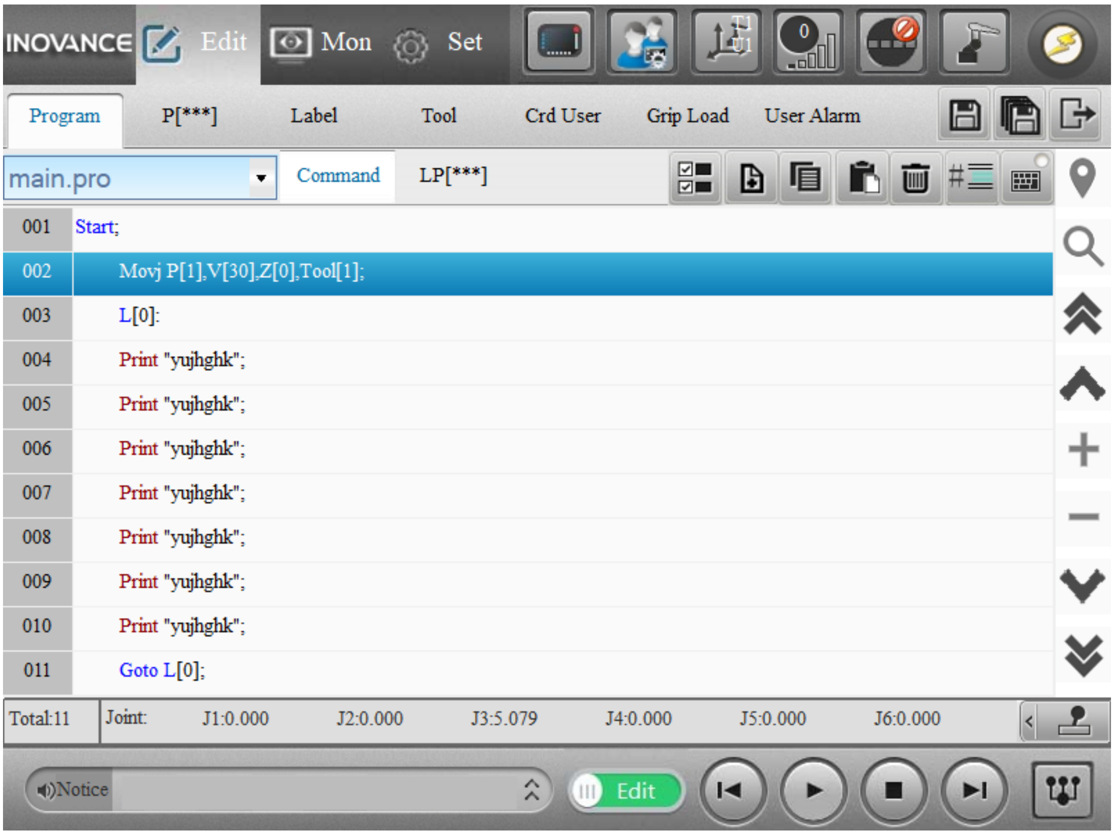


5. Create a new instruction, select Movj from the motion instructions, and click **Add pnt** on the pop-up page.

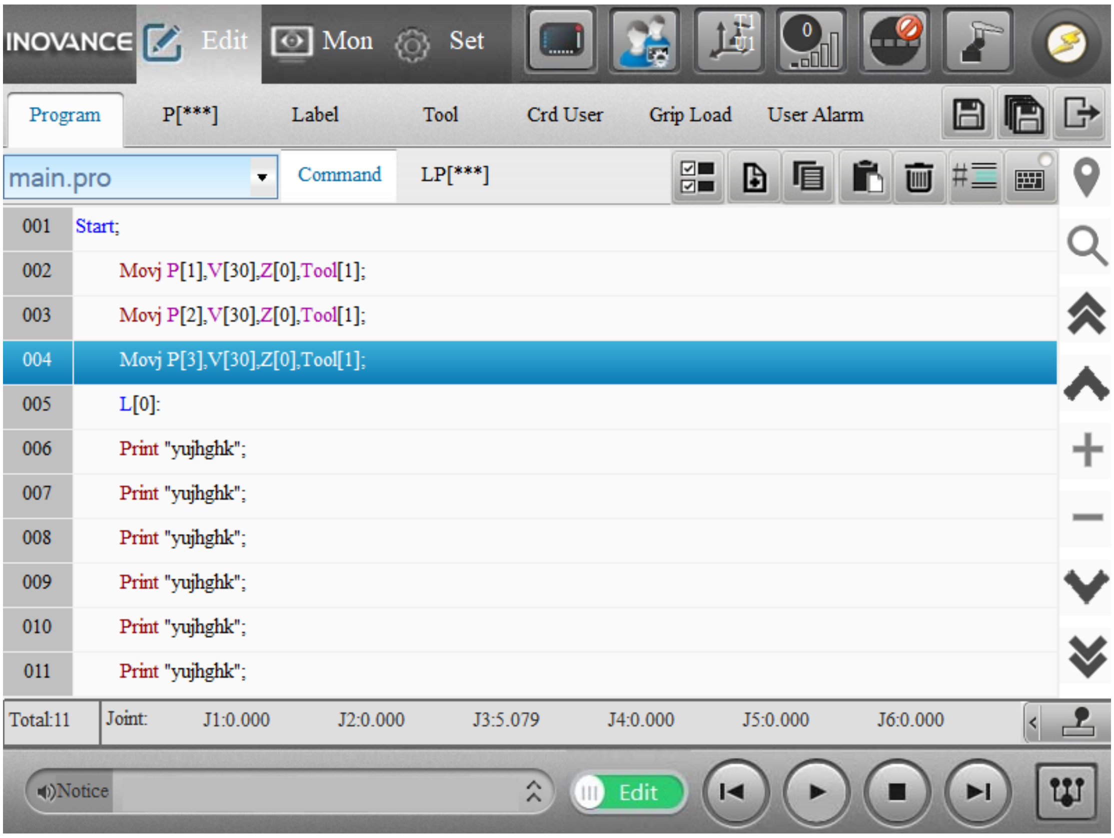




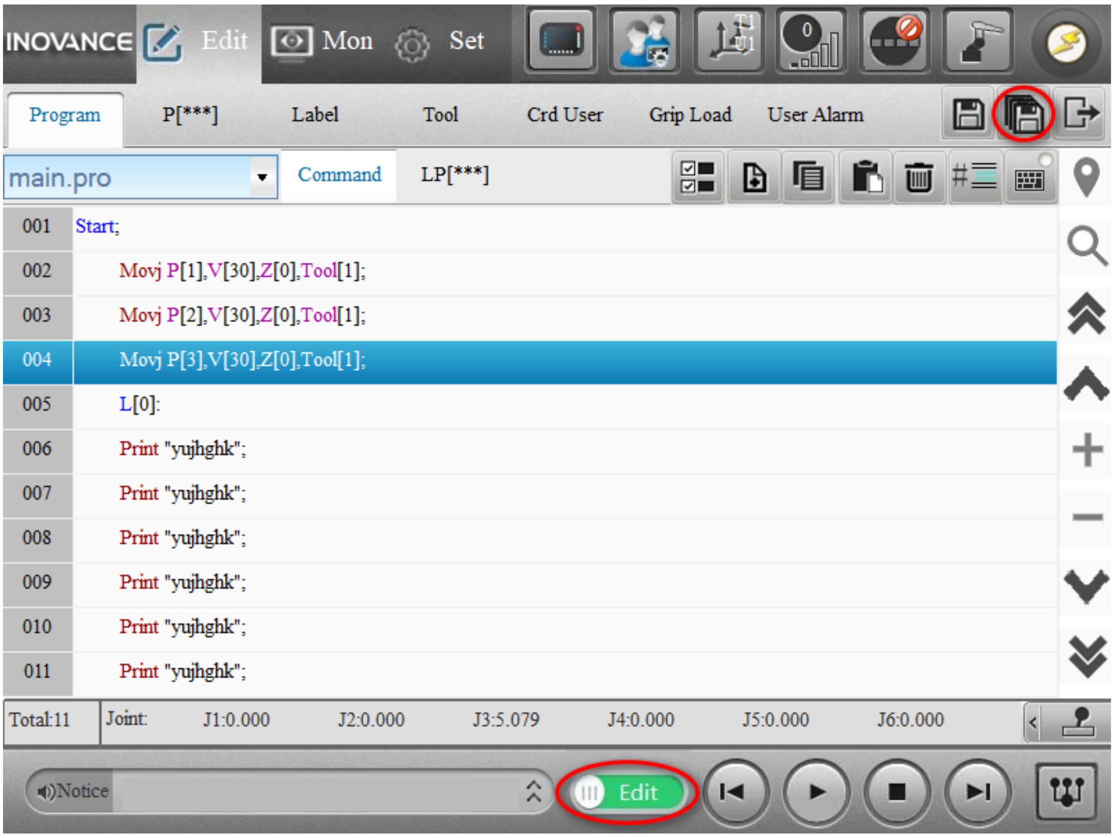
6. Click **OK**. The added instruction is displayed in the program.

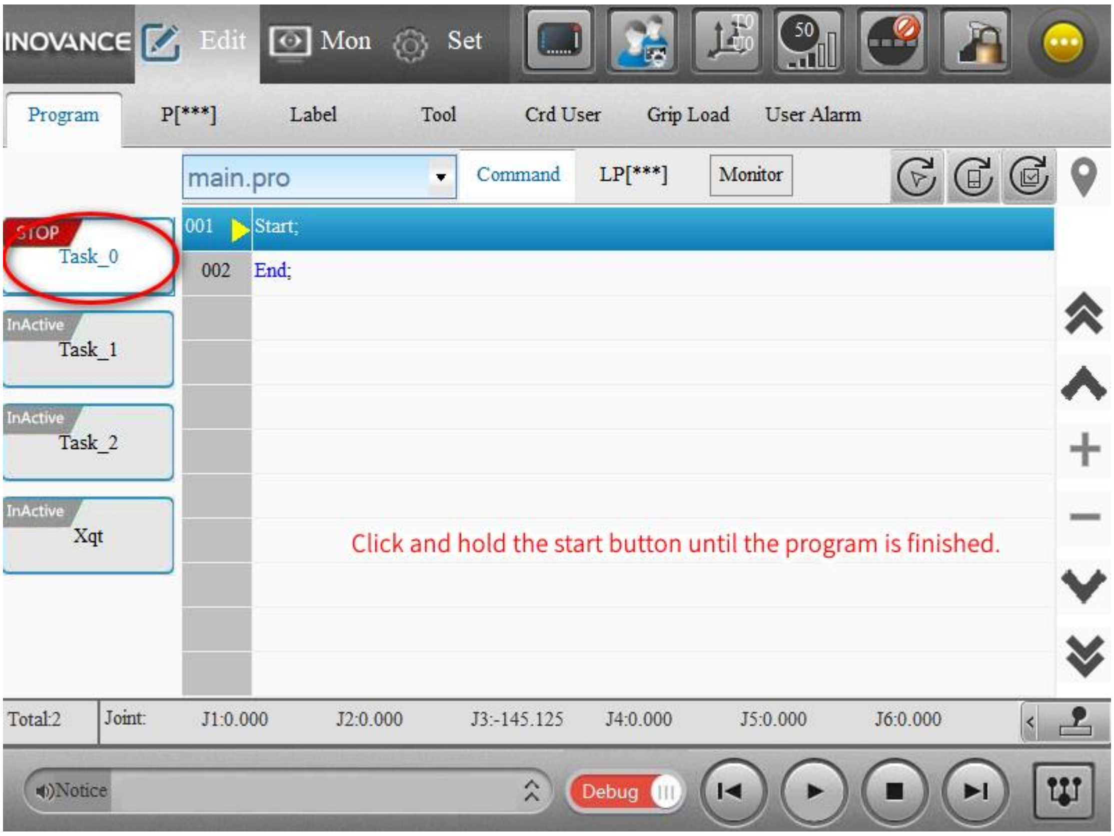


7. Repeat the above steps to teach points P[1], P[2] and add two Movl instructions.

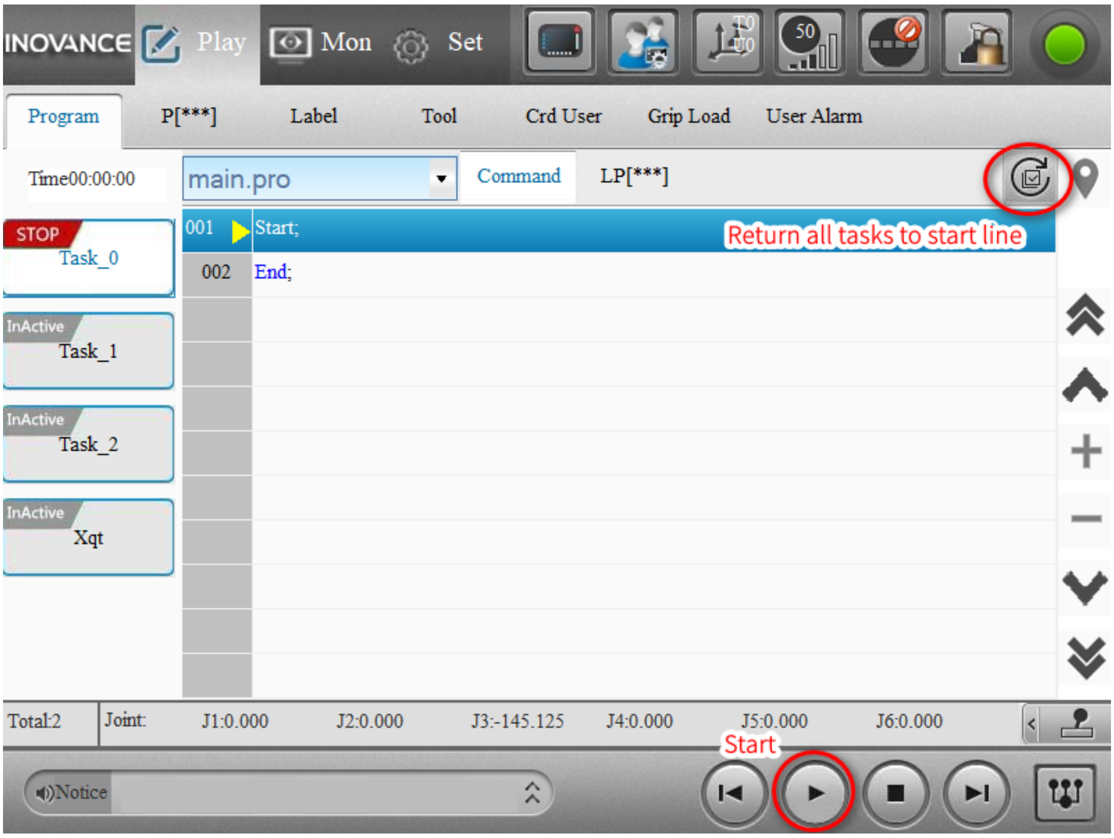


8. When you have finished editing, click the **Save All** button. Switch from edit mode to debug mode, enable the servo, click and hold the start button until the program is finished.





9. Switch to the play mode, return all tasks to the start line, and then click the start button.



3.8 Shortcut Keys

For the PC-based teach pendant, shortcut keys are supported in the programming interface.

Shortcut Combination	Description
Ctrl + A	Select all items, and bring up the multi-select box at the same time. Or unselect all items if all items have already been selected.
Ctrl + Q	Bring up the multi-select box.
Ctrl + C	Copy
Ctrl + V	Paste
Ctrl + D	Comment line
Ctrl + F	Find
Ctrl + G	Go to a specific line
Ctrl + S	Save
Command search	Open the instruction list. Enter the string for search. Press Backspace or Del to delete the string, or ESC to cancel search and return to the list of commonly used instructions.

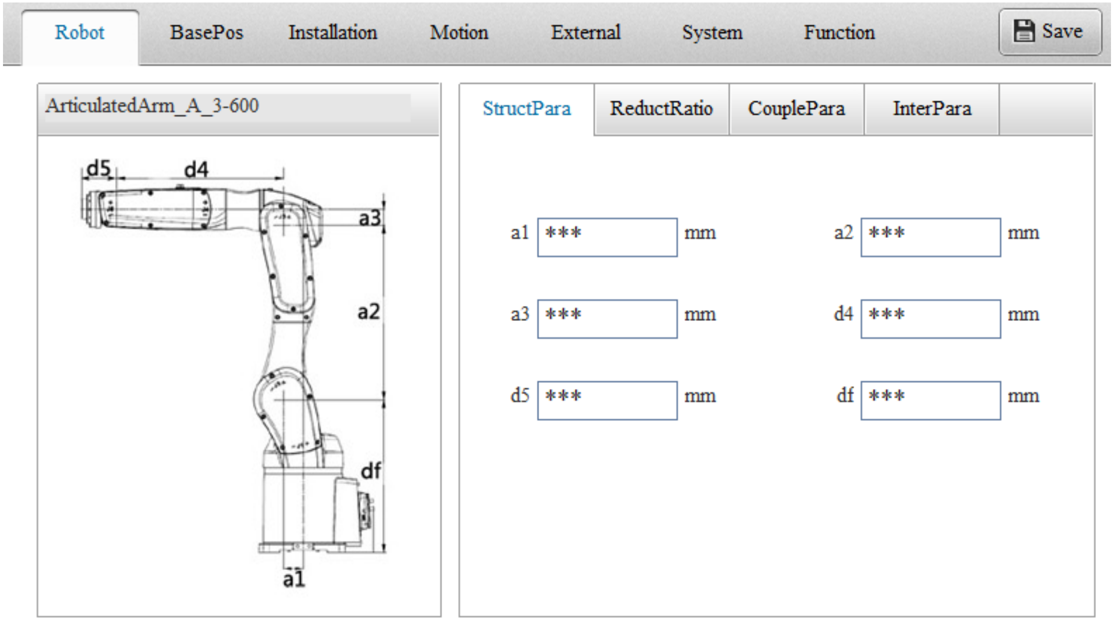
4 Settings

Before teaching the robot, you need to make a series of settings, including robot settings, zero

point settings, coordinate system settings, motion parameter settings, peripheral settings, system settings, extension settings, etc.
After setting the parameters on a certain page, remember to click the **Save** button in the upper right corner to save the settings.

4.1 Robot Settings

The robot settings include structural parameters, reduction ratio, coupling parameters, etc., and need to be configured in factory mode.
Note: Robot settings are of great significance. Please contact the manufacturer if you need to change them.

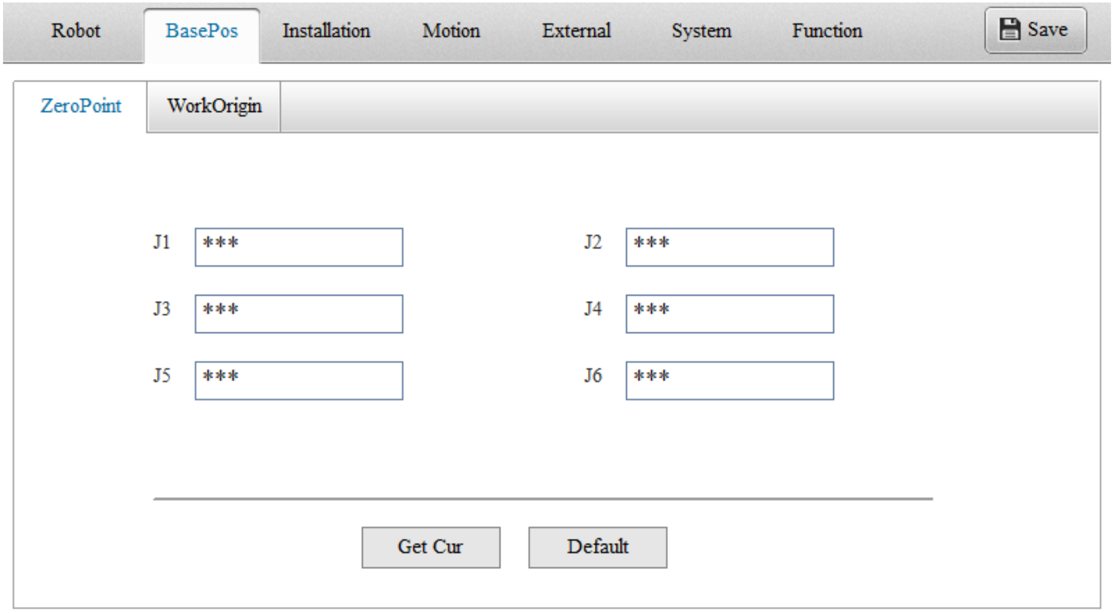


4.2 Zero Point Settings

The zero point settings include absolute zero point, working origin and zero point calibration (zero point calibration is only available to the SCARA robots).

4.2.1 Absolute Zero Point

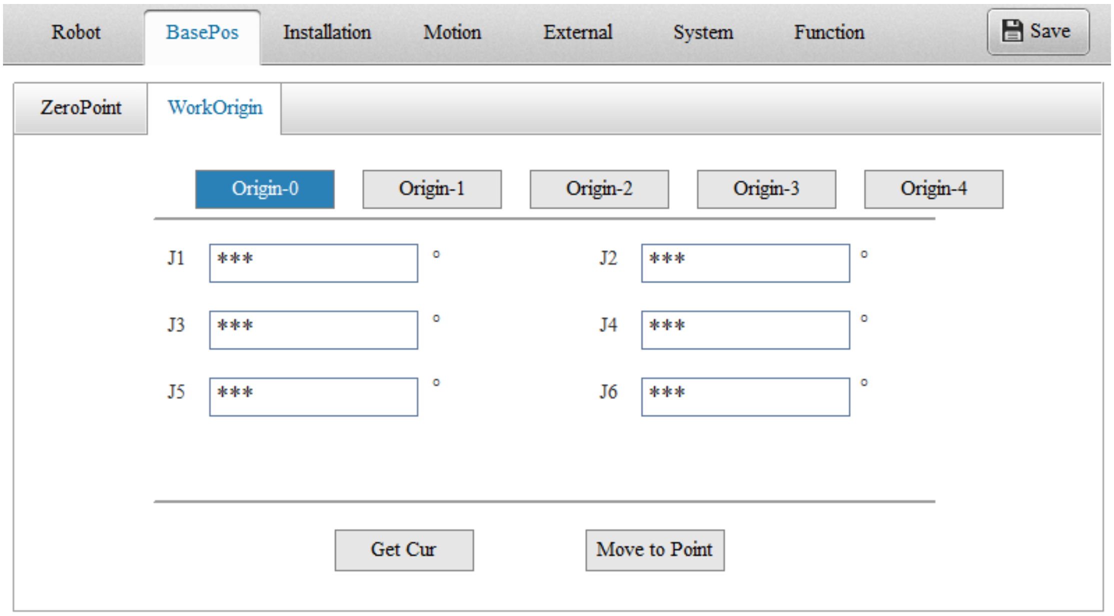
The following figure shows the interface for setting the absolute zero point. To set the absolute zero point, do as follows:
Step 1: Adjust the robot's joints to zero position using the teach pendant. Click the **Get Cur** button, and the encoder values are automatically displayed in the text fields.
Step 2: Click the **EmStop** button, then click the **Save** button, and restart the controller.



Note: The encoder values supported by the teach pendant for J1 to J6 range from -2^{31} to 2^{31}. If the encoder values are too large, it is recommended to clear the number of encoder turns and repeat the above steps. For the operation of clearing the number of encoder turns, see the related servo and encoder manuals.

4.2.2 Work Origin

Different from the zero point, the work origin is a user-defined position variable that can be used in the program. You can set up to five work origins and save them into variables Home[0] to Home[4]. You can manually enter the coordinates of the work origin. Alternatively, you can click and hold the **Move to Point** button to move the robot to the origin, then click the **Get Cur** button to get the coordinates, and then click the **Save** button, as shown in the following figure.



4.2.3 Zeroing

This feature is only available to SCARA robots.

During use, SCARA robots occasionally lose their zero point due to impacts on hard objects, unreasonable parameter settings, and other reasons, which affects their absolute accuracy. In addition, this may also result in the already taught points being unusable. In order to effectively solve the problem of zero point loss and improve the usability of SCARA robots, the zeroing feature has been introduced.

Note: All axes are reset to zero using automated collision stops. The stops for J1 and J2 axes are hard stops, and additional tooling is required for J3 and J4 axes. Additionally, due to the robot structure, the zeroing of J3 and J4 axes must be carried out simultaneously.

4.2.3.1 Preparation before Zeroing

(1) Reference position: All axes are zeroed depending on the reference positions. The reference positions for J1 and J2 axes are the positive and negative hard stops, and the reference position for J3 and J4 axes is the limit ring. Before zeroing, ensure that the positive and negative hard stops of J1 and J2 axes, as well as the limit ring of J3 axis are unchanged as they left the factory.

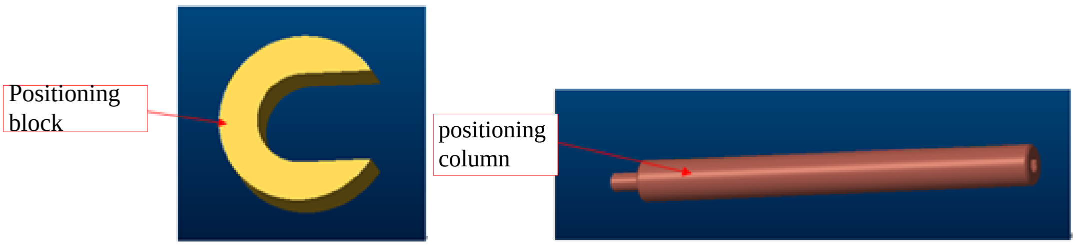
(2) If you want to zero the J3 and J4 axes, please install the zeroing tooling first*. (See below for details)

(3) Remove the fixture to avoid entanglement of the air tubes, preventing mistaken belief that the robot has reached the hard limit position.

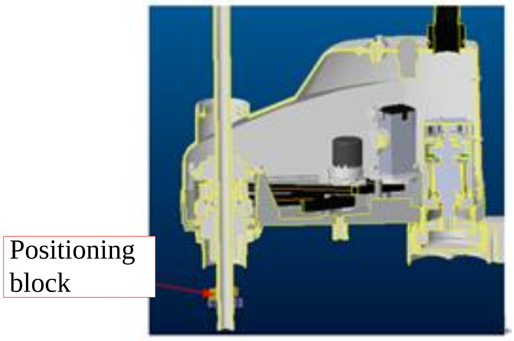
(4) Clear items between the current position and the positive limit stop to eliminate interference.

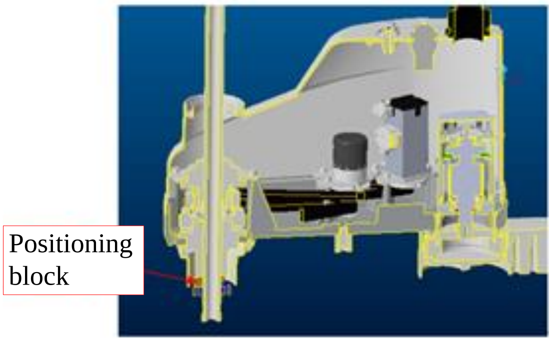
Installation of the zeroing tooling for J3 and J4 axes:

Two types of tooling are required: positioning block and positioning column, as shown in the following figure:

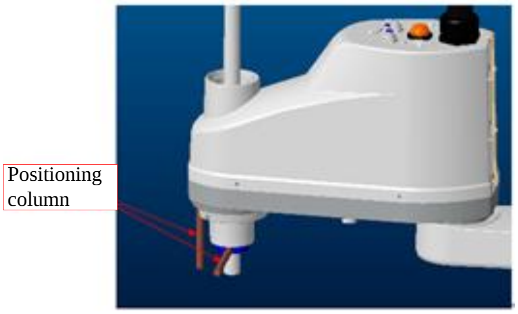


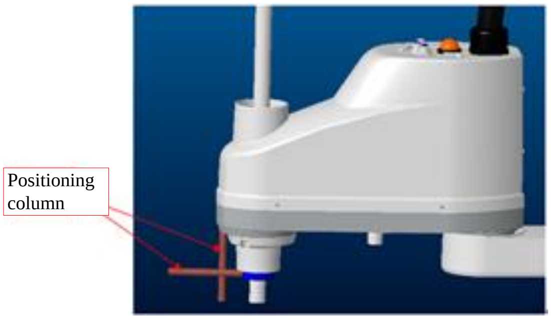
Release the brake of the J3 axis, move the lead screw to an appropriate position (where the positioning block can be easily snapped onto the lead screw), and snap the positioning block onto the lead screw, as shown in the left figure below. Then slowly move the J3 axis in the positive direction by hand until the positioning block hits the spline nut or spline nut housing, reaching the zero point of the J3 axis, as shown in the right figure below.





Secure the positioning column to the forearm and the limit ring, as shown in the left figure below. (Ensure that the two rods do not collide. The zeroing of J3 and J4 axes is the process of collision of the two rods, as shown in the right figure below.)



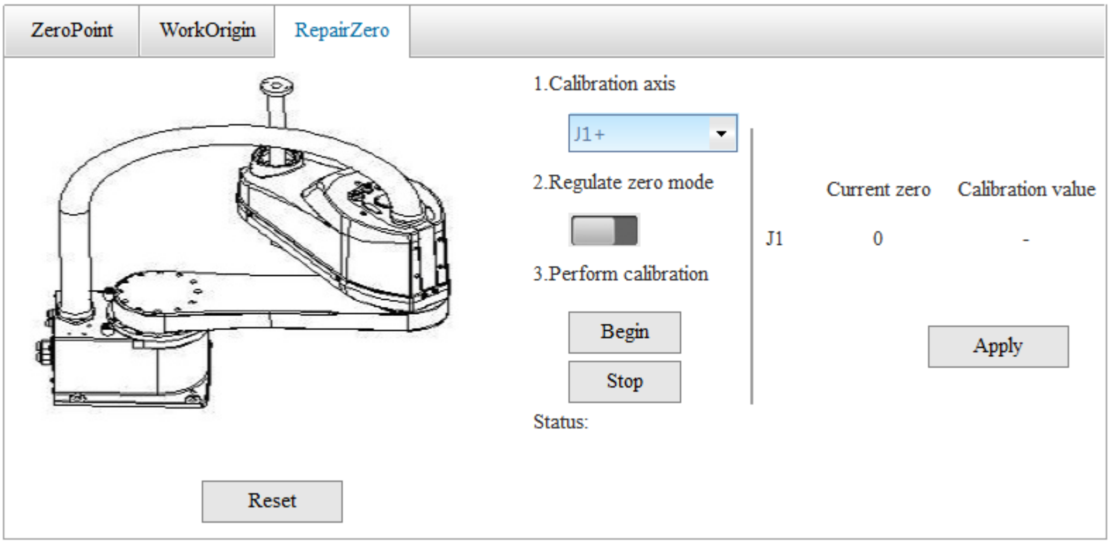


4.2.3.2 Steps of Zeroing

(1) Log in to the teach pendant as Manager or Factory user.

(2) Go to the zeroing page.

Set > **BasePos** > **RepairZero**



(3) Select the axis number.

Select the axis number in the **Calibration axis** drop-down list.

Note: The movement direction of all axes during zeroing is positive.

(4) Enter the zeroing mode.

Turn on the **Regulate zero mode** switch.

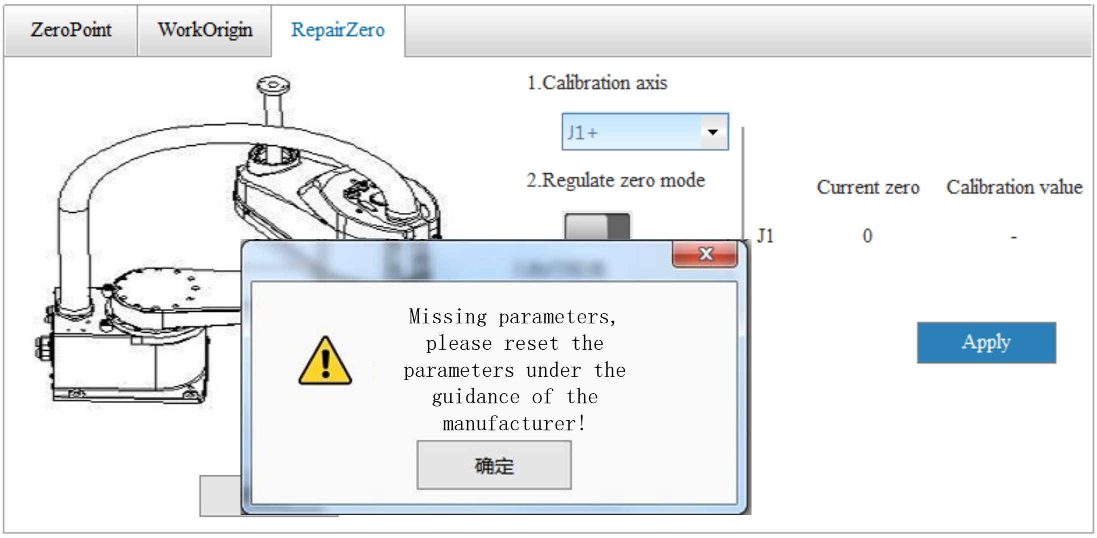
Note:

(a) If you are prompted for missing parameters, reset the parameters under the guidance of the manufacturer.

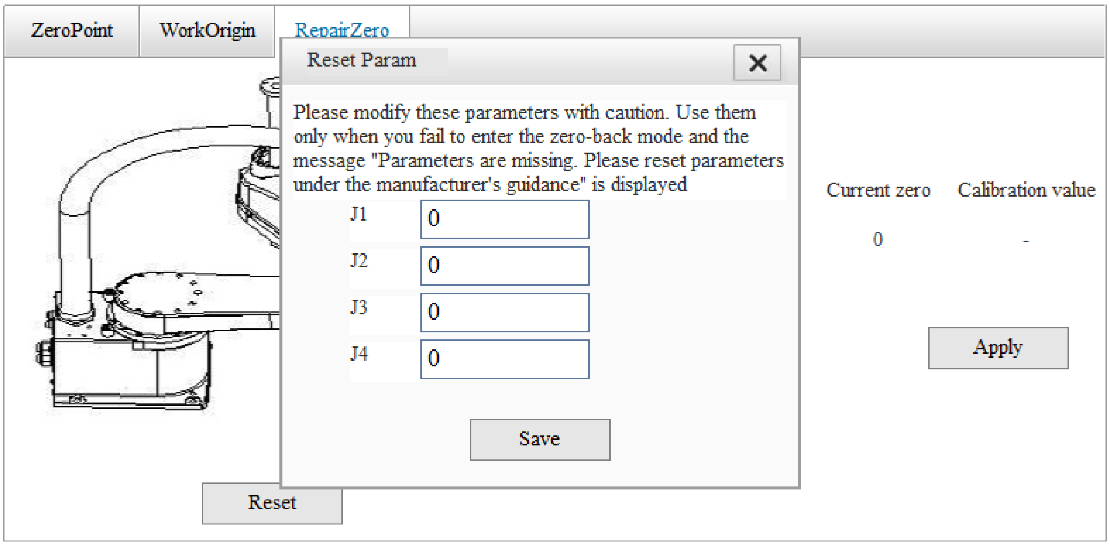
Under normal circumstances, the newly produced robots will not lose parameters and can normally enter the zeroing mode. However, in the following two situations, it may fail to enter the zeroing mode:

- The robot system is produced before the introduction of S03.21R;
- The robot system is produced after the introduction of S03.21R, but is recovered and loses the zeroing configuration.

In these two situations, you will be prompted for missing parameters.

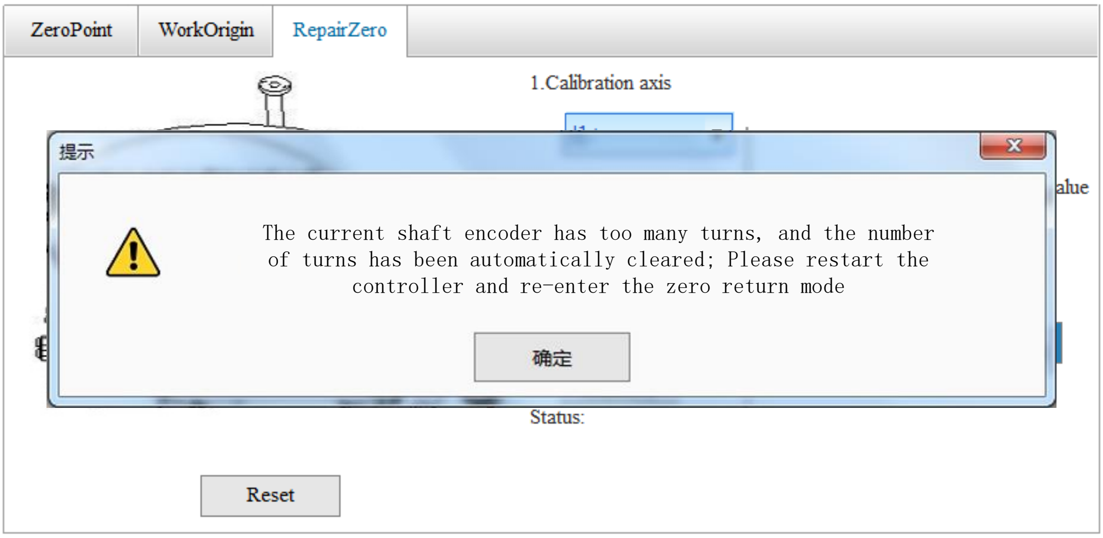


- ➢ The switch cannot be turned on for robot system produced before the introduction of S03.21R. (The manufacturer only retains the parameters for the new robot system.)
- ➢ If the product system is produced after the introduction of S03.21R, please contact the manufacturer and reset the parameters under the instructions. (Ask for the parameters from the manufacturer, enter the parameters into the corresponding axis in the **Reset Param** dialog and click **Save**. Then you can continue with the zeroing process.)



➢ If it is not possible to identify whether the robot system was manufactured before or after S03.21R, please contact the manufacturer to check whether the parameters are retained by the manufacturer. If yes, reset the parameters as described above.

(b) When switched to zeroing mode, if the encoder's multi turn value exceeds 2000, a pop-up prompt will appear, triggering an emergency stop and generating a permanent alarm. In this case, you need to restart the controller and perform zeroing operation again.



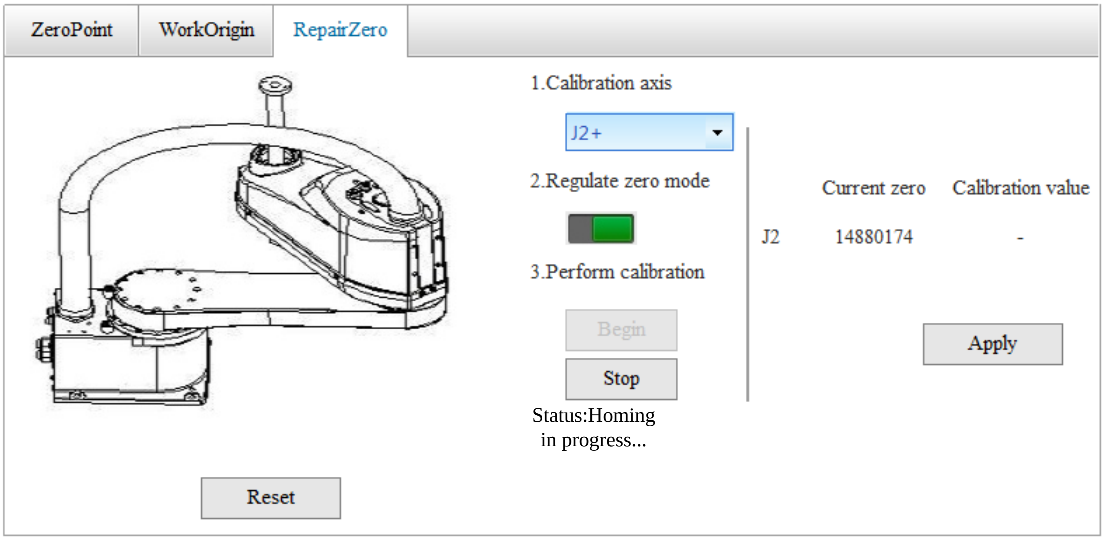
(c) When switched to zeroing mode, the robot is automatically enabled, please pay attention to safety.

(5) Start zeroing

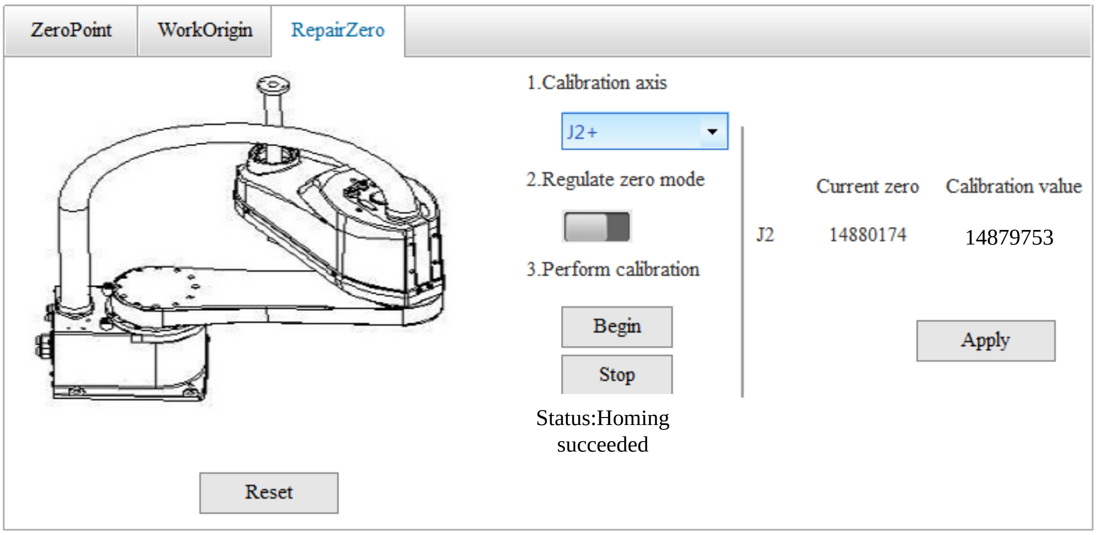
Take the J2 axis as an example, select J2+ and click **Begin**.

Note:

- To ensure accuracy, the zeroing is slow and may take a few minutes. (To accelerate the zeroing process, you can move the axis to about 10° from the hard stop before starting the zeroing operation.)



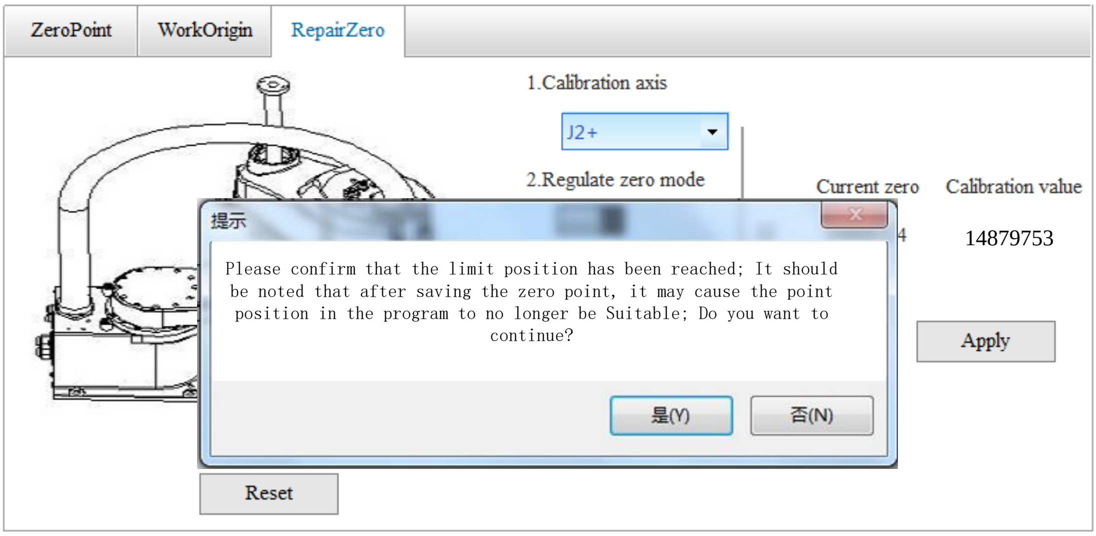
When the zeroing is complete, the status displays "Homing succeeded" and the **Calibration value** is updated.

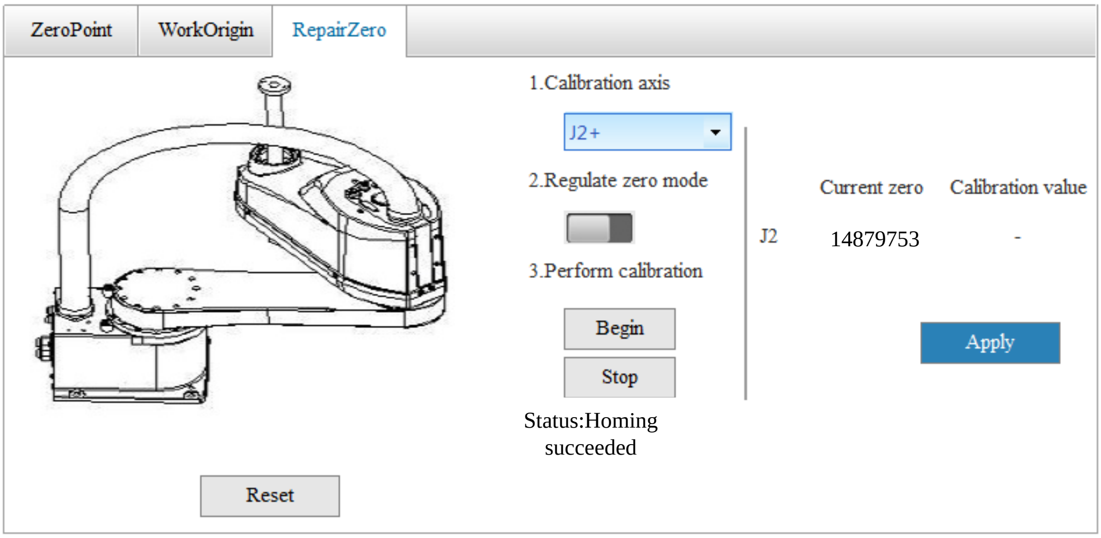


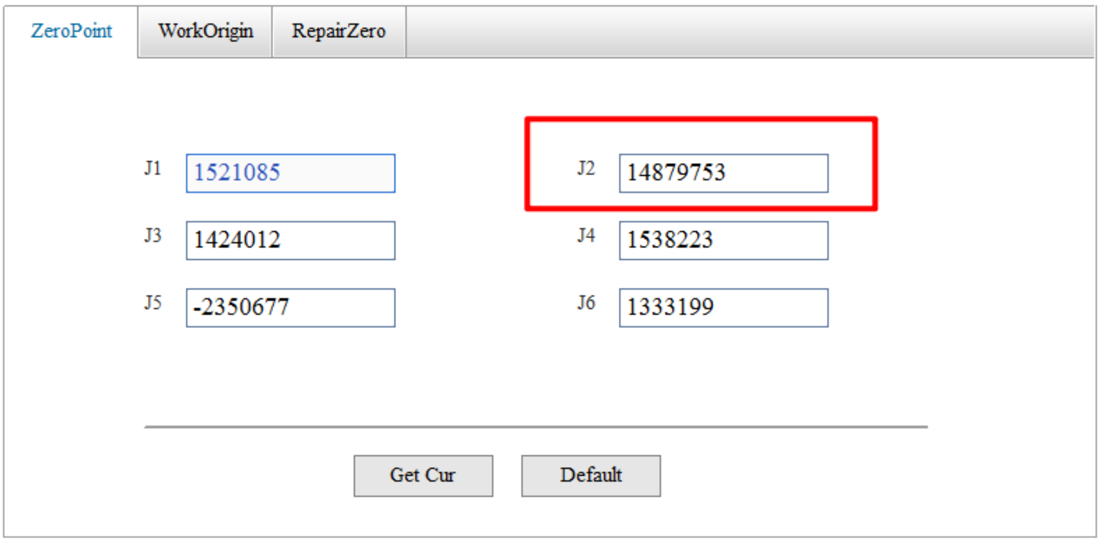
(6) Update the zero point.

Perform emergency stop manually and then click **Apply**.

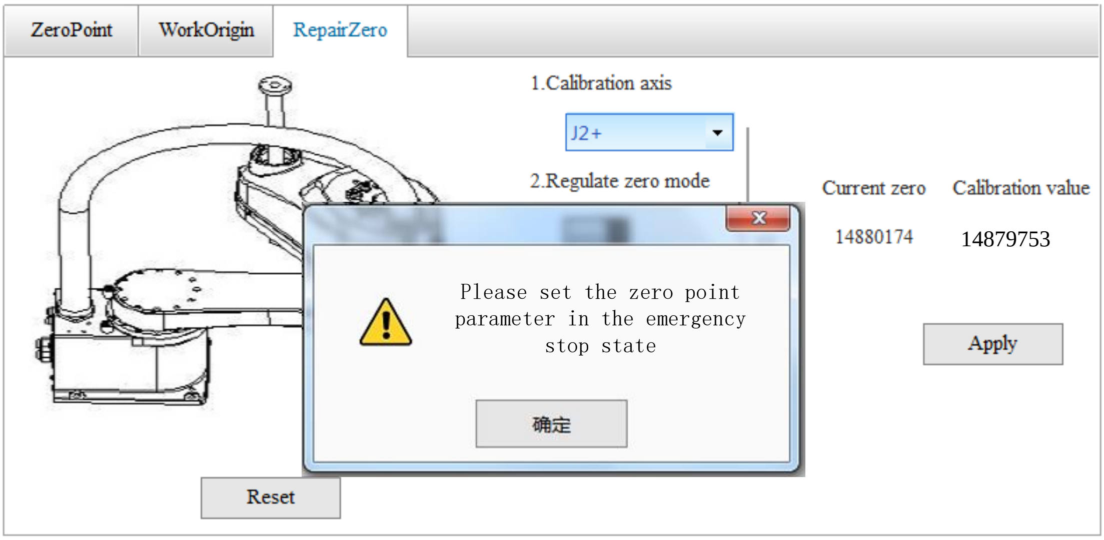
In the pop-up prompt, click **Yes**. Then the calibrated zero point values will be applied and replace the current zero point values.







Note: If you click **Apply** without making emergency stop first, a pop-up prompt will appear.



(7) Post-zeroing check

After updating the system zero point, check the zeroing effect.

4.2.3.3 Cautions

Please note that:

1) If you want to zero the J3 and J4 axes, please install the zeroing tooling first*.
2) Before zeroing, ensure that the positive and negative hard stops of J1 and J2 axes, as well as the limit ring of J3 axis are unchanged as they left the factory.
3) Remove the fixture to avoid entanglement of the air tubes, preventing mistaken belief that the robot has reached the hard limit position.
4) The movement direction of all axes during zeroing is positive. Clear items between the current position and the positive limit stop to eliminate interference.
5) Only one axis can be zeroed at a time. If you want to zero multiple axes, zero them one by one.
6) If you are prompted for missing parameters, please reset the parameters under the guidance of the manufacturer.
7) When switched to zeroing mode, if the encoder's multi turn value exceeds 2000, a pop-up prompt will appear, triggering an emergency stop and generating a permanent alarm. In this case, you need to restart the controller and perform zeroing operation again.
8) After updating the system zero point, check the zeroing effect.

4.3 Installation Parameter Settings

You can configure the arm load parameters and the installation form of the robot.

a) Arm Load Settings

The arm load parameters all default to 0. After configuring the parameters, click the **Save** button in the upper right corner to make them take effect.

Parameters:

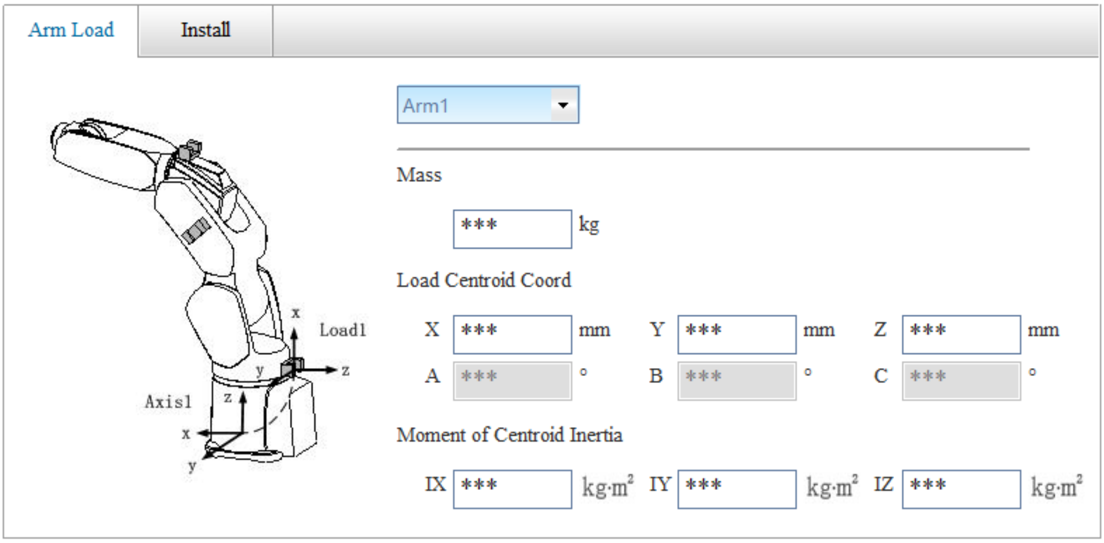
Mass: The load mass of the tool, in kg;

X, Y, Z: The X, Y, and Z coordinates of the centroid relative to the joint coordinate system, in mm;

A, B, C: Orientation of the load, currently not supported;

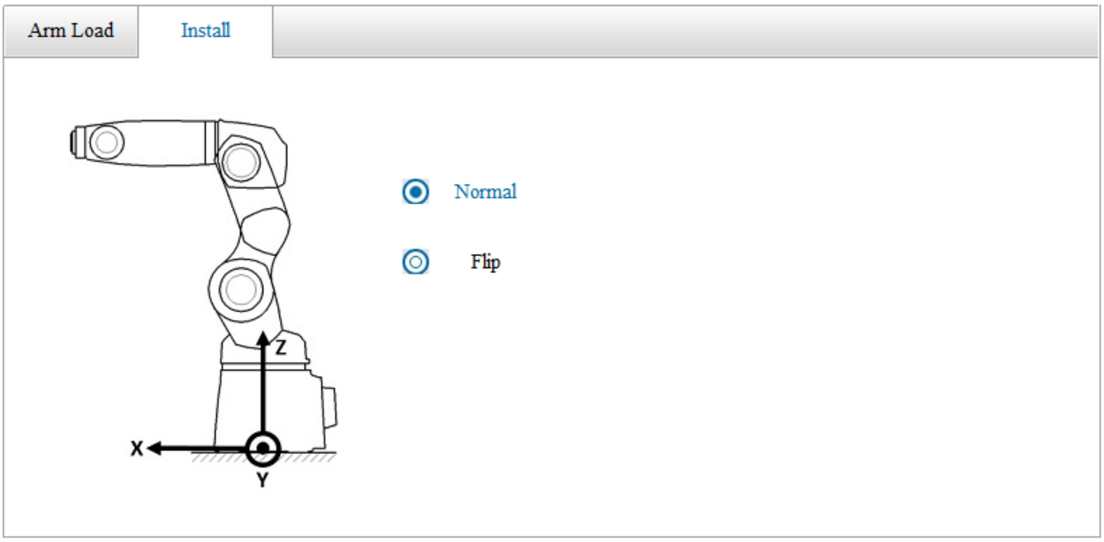
IX, IY, IZ: The moment of inertia of the load around the X, Y, Z axes of its centroid coordinate system, in kg*m2.

Make sure all the load parameters are correctly set; otherwise, collision detection false alarms, too long or short cycle time, or abnormal current may occur.



b) Installation Form Settings

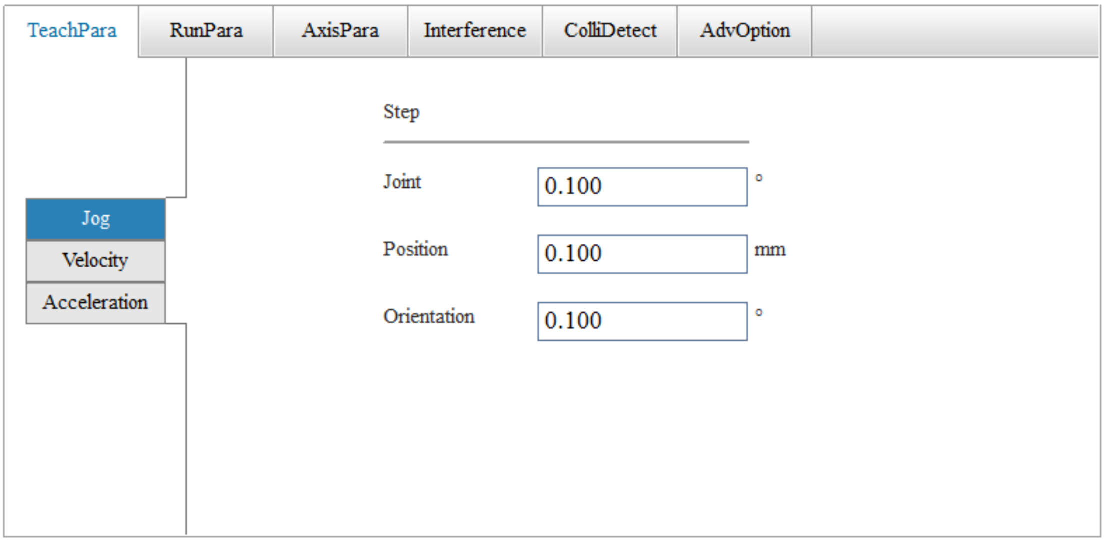
The installation form settings are only available to 6-axis robots. You can choose to install the robot on the floor (upright) or the ceiling (inverted). After modification, you need to click the **Save** button in the upper right corner and restart the controller to take effect. This setting affects the robot motion. Ensure that the setting matches the actual situation.



4.4 Motion Settings

a) Teach Parameter Setting

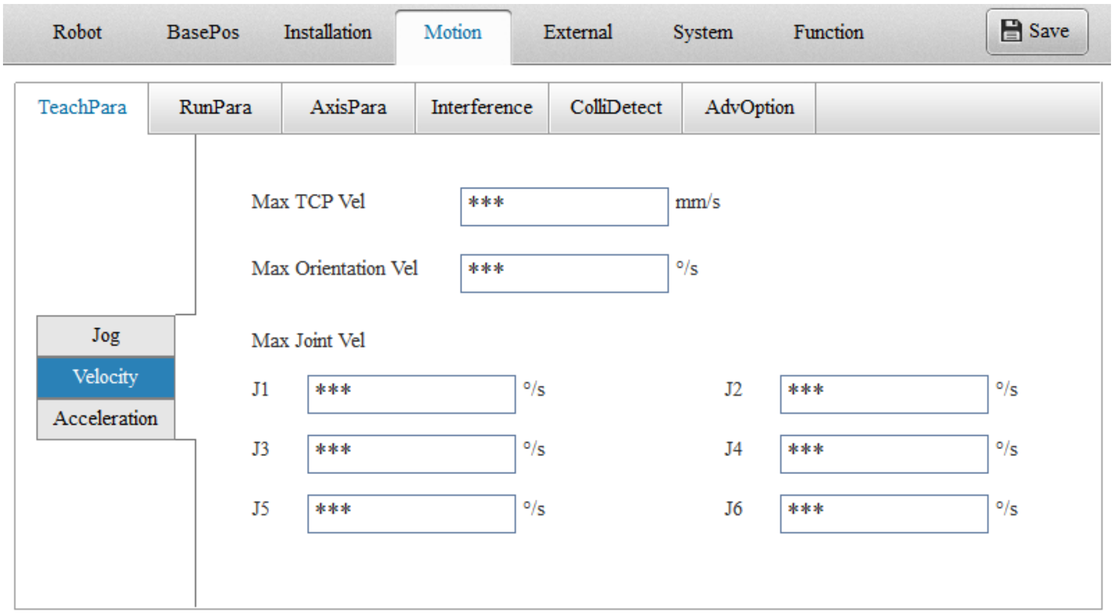
1. Jog



Jog: You can customize the step value of jog, which can be used when you select .

Level	Joint (deg)	Linear (mm)	Orientation (deg)
G1 (short)	0.05	0.05	0.05
G2 (medium)	0.5	0.5	0.5
G3 (long)	2	2	2
U (user-defined)	Max: 10 Min: 0.01	Max: 10 Min: 0.01	Max: 10 Min: 0.01

2. Velocity



Definition: The maximum velocity of robot during teaching.

Velocity = Velocity setting * Velocity percentage selected in tool bar.

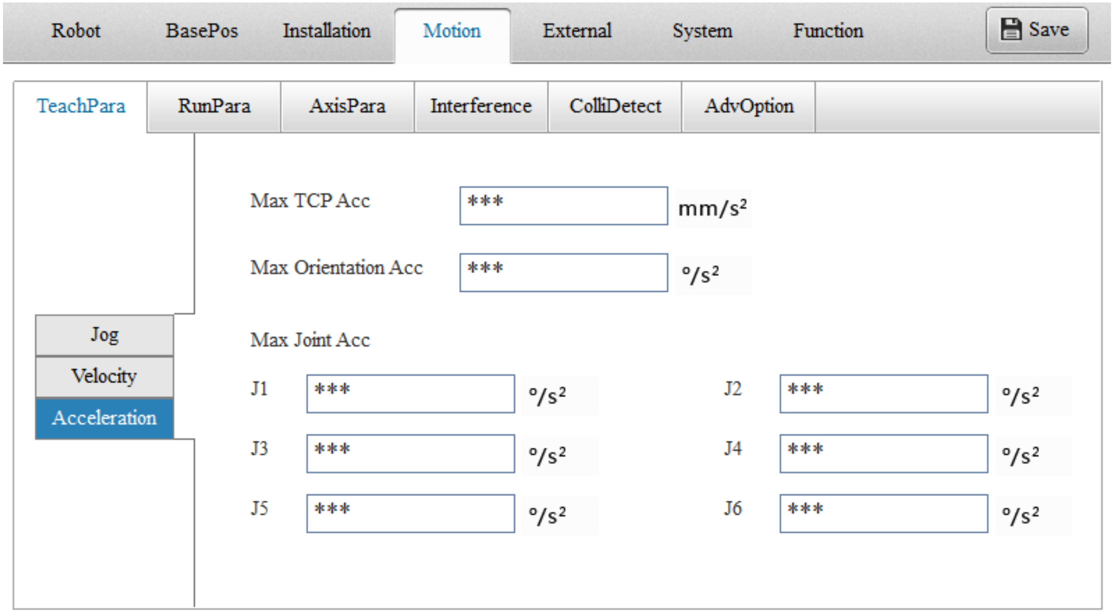
Setting range:

1. Max TCP Vel: 0.01 to 9999999.999;
2. Max Orientation Vel: 0.01 to 9999999.999;
3. Max Joint Vel: 0.01 to 9999999.999;

Setting permissions:
Manager, InoTeachPad control, edit mode

Note:
1. The actual effective value is subject to the controller, and the controller will read back the set value each time the set value is saved. Therefore, if the set value is out of range, it will not take effect.

3. Acceleration



Definition: The maximum acceleration of robot during teaching.

Setting range:
1. Max TCP Acc: 0.01 to 9999999.999;
2. Max Orientation Acc: 0.01 to 9999999.999;
3. Max Joint Acc: 0.01 to 9999999.999;

Setting permissions:
Manager, InoTeachPad control, edit mode

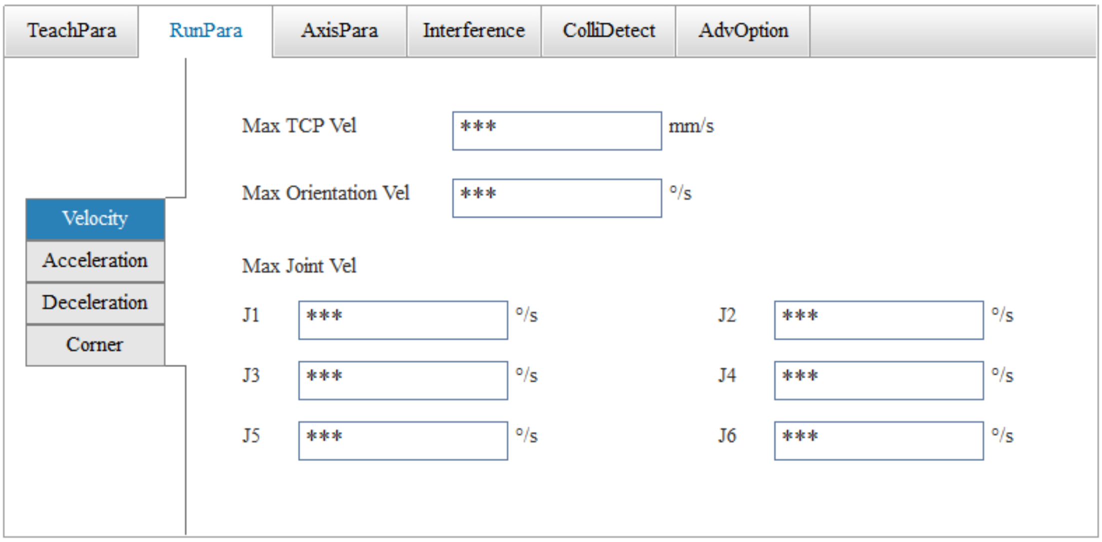
Note:
1. The actual effective value is subject to the controller, and the controller will read back the set value each time the set value is saved. Therefore, if the set value is out of range, it will not take effect.
2. What is set here is the velocity and acceleration of the robot's motion in various coordinate systems. In teach mode, the parameters set here are not used for single-step or continuous

operation of the robot, instead, 50% of the operating speed and acceleration is used.

b) Run Parameter Setting

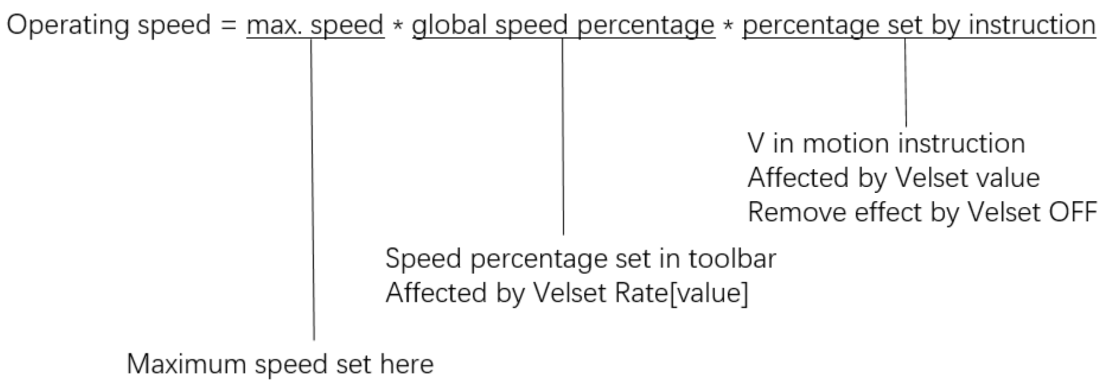
In the play mode, the parameters in the following figure are used.

In the teach mode, single-step operation or continuous operation is performed at 50% of the velocity and acceleration set here. However, the deceleration set here is used in the same way in both teach mode and play mode.

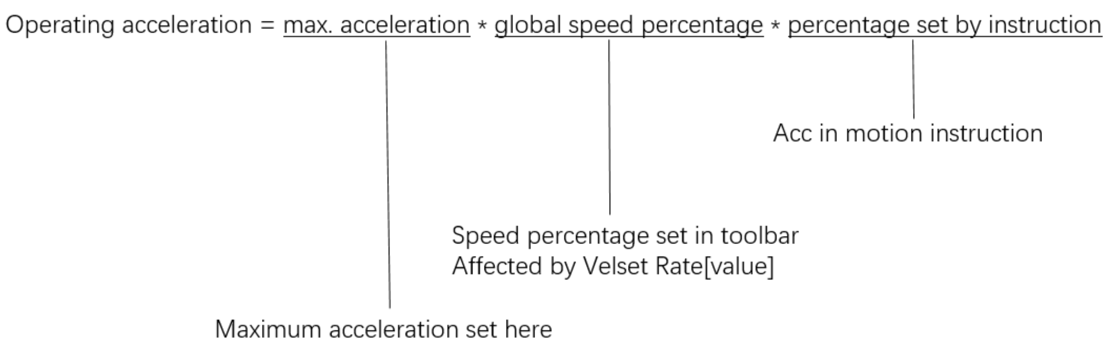


Velocity: The maximum velocity during operation.

The following figure shows the definition of the actual operating velocity:



Acceleration: The maximum acceleration during operation.



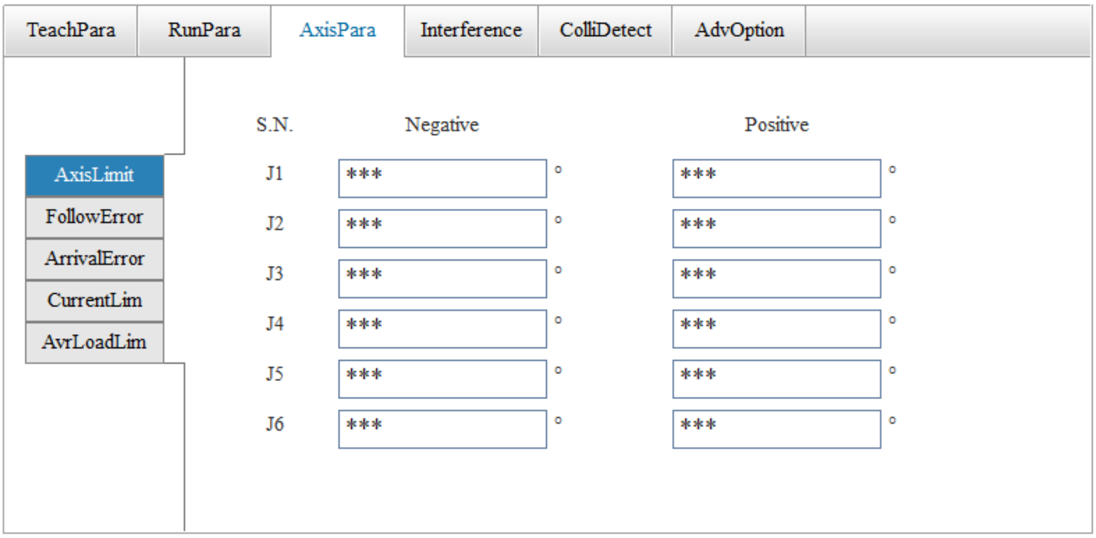
Deceleration: Deceleration for stop.

Note: The above parameters are subject to performance of the servo system.

Transition Settings:

You can set the unit transition length here.

c) Axis Parameter Settings



AxisLimit: The limit position of each axis. For safety reasons, always set the axis limits within the range of mechanical stops.

The J4 axis limit of the floor-mounted SCARA robots and the ceiling-mounted SCARA robots ranges from -36000° to 36000°. The J6 axis limit of the 6-axis robots ranges from -720° to 720°.

FollowError: This parameter is determined by the robot's commanded acceleration and the servo's stiffness. When an alarm is generated as the FollowError is too large, reduce the acceleration or increase the FollowError setting.

ArrivalError: This parameter defines the allowable error between position planned for the robot and the actual position the robot reaches.

CurrentLim, AvrLoadLim: See 7.10 Current Protection.

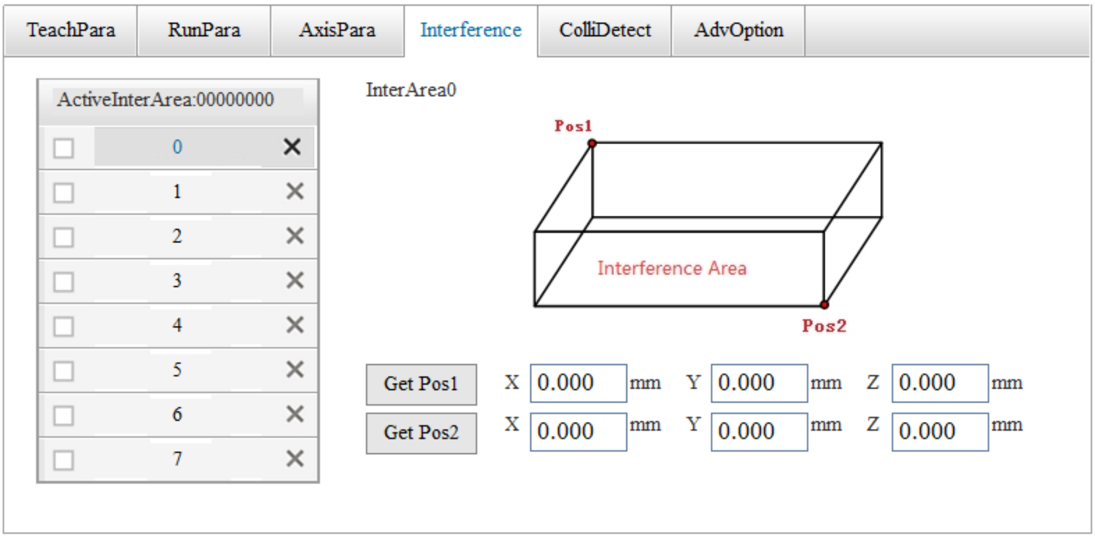
Note: ArrivalError, FollowError, CurrentLim and AvrLoadLim can only be modified in the Factory mode.

d) Interference Area Settings

An interference area is a cuboid determined by X, Y and Z coordinates of two points of a diagonal. You can set eight interference areas. Interference areas take only positions into account. When interference areas are activated, an error is generated upon entry of the robot into the interference areas. Multiple interference areas can be activated at the same time.

To edit an interference area, click on its serial number and then edit the parameters on the right side.

To activate an interference area, check the corresponding checkbox.



Note:

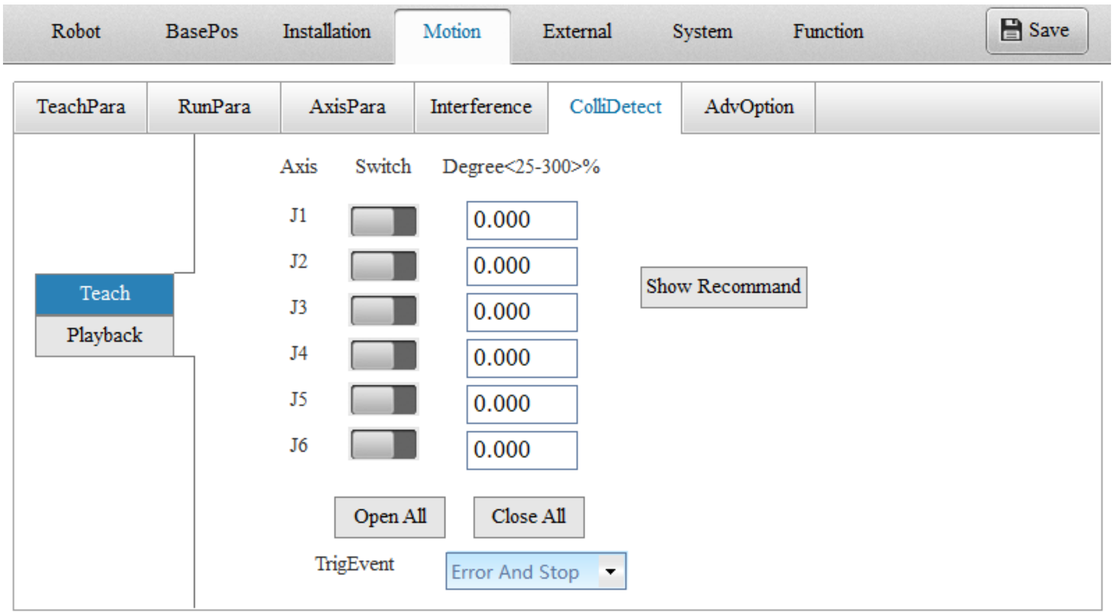
The detection of entry into the interference areas is only applicable to the end of the robot, not to the tool.

After entering the interference zone, the robot cannot immediately stop as deceleration takes time. Therefore, it is recommended to set the interference area to be larger than the actual one.

e) Collision Detection Settings

Collision detection parameters are divided into collision detection parameters for teach mode and collision detection parameters for play mode. The collision detection parameters for teach mode only take effect in teach mode, while the collision detection parameters for play mode take effect in the teach mode.

The collision detection parameters include three types of parameters: collision detection switch (ON/OFF) for each axis; collision detection sensitivity for each axis, ranging from 25 to 300, with a default of 100; and event triggered upon collision, the supported events currently only include Error and Stop.



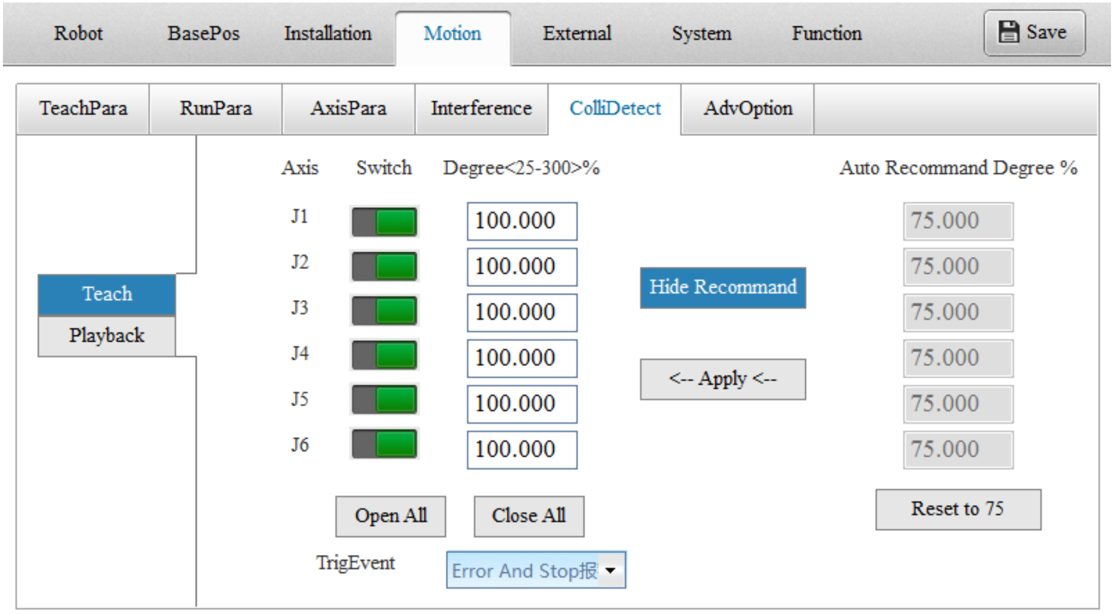
In addition to the above parameters, three buttons are provided.

The **Open All** and **Close All** buttons open or close the collision detection switch of all axes at the same time.

The **Show Recommend** button gets the system recommended sensitivity. When you click the **Show Recommend** button, the button text changes to **Hide Recommend** and an **Apply** button appears below and the recommended sensitivity values are displayed on the right side. Four or six recommended sensitivity values are displayed and these values are updated approximately every 3s. The recommended sensitivity is automatically reset to 75 each time the controller is powered. Each time the robot moves, the system automatically calculates the recommended sensitivity value, which is updated when the system calculates the recommended sensitivity value greater than the value displayed. The **Reset to 75** button restores the displayed recommended sensitivity value to 75. If the robot has experienced slight contact or other accidents that have caused the recommended sensitivity value to increase abnormally, you can click this button to calculate the recommended sensitivity again.

There are several recommended sensitivity changes that require special attention:

1. When there is a significant change in the operating conditions (such as a significant increase or decrease in load or velocity, or a slight collision or contact that has caused an abnormal increase in recommended sensitivity), in order to avoid false alarms and ensure sufficient sensitivity, it is recommended to reset the recommended values and apply the latest recommended values.
2. After the robot runs for a long time, due to changes in joint friction and other factors, even if it still operates under the same working conditions, the recommended sensitivity values may also change. When the sensitivity change is within 10%, the collision detection sensitivity can be adjusted less frequently (unless a false alarm has already occurred). When it exceeds 10%, it is recommended to apply the latest recommended values.
3. When the recommended value is less than 100%, it is not recommended to change the sensitivity to a value less than 100%; otherwise false alarms may occur when the working conditions change. Set the sensitivity between 75% and 100% only when there is a high requirement for collision detection sensitivity.
4. When you manually change the velocity in the play mode, a false alarm can occur when the changes are large (e.g., directly from 25% to 100%), even if the recommended sensitivity at 100% velocity is used. This is normal. It is recommended to switch the velocity step by step and slowly, e.g. from 25% to 50%, then to 80%, and finally to 100%.

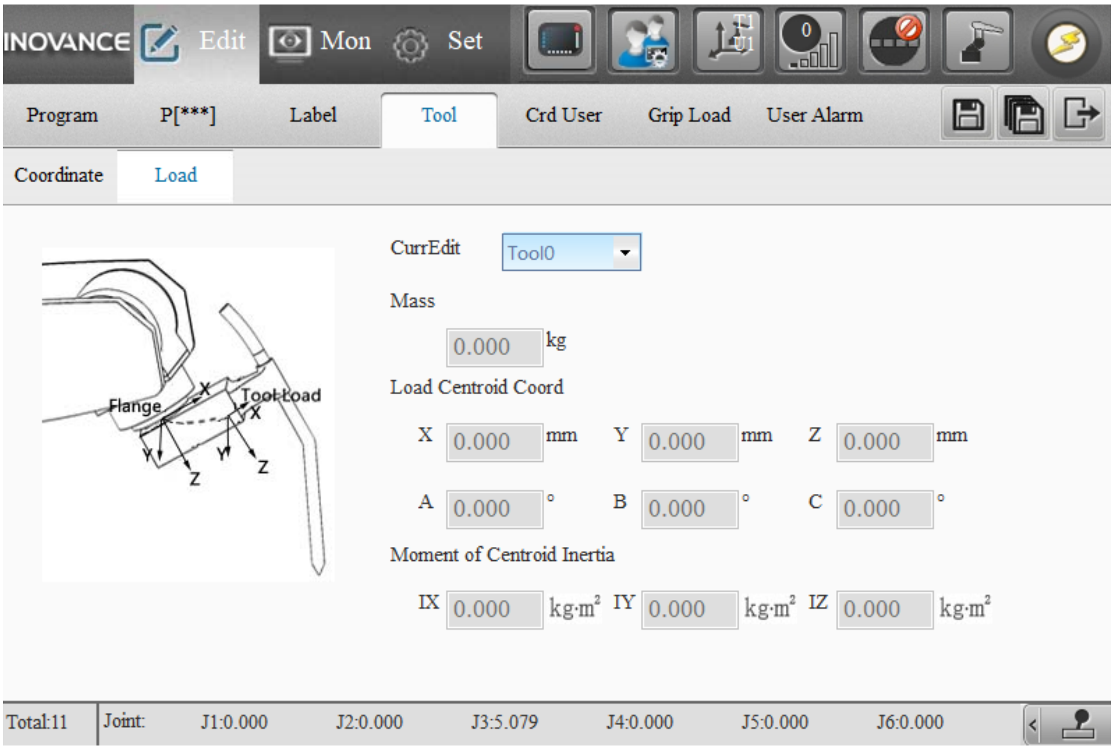


The commissioning procedure for collision detection is as follows:

1） Set the load parameters

The load parameters include mass, eccentricity, inertia, etc., which can be calculated from the 3D model of the load. For loads with a simpler shape, the load parameters can be calculated simply by referring to the calculation method in the appendix.

Enter the calculated load parameters into the selected tool load or grip load as shown in the following figure.



2） Activate the corresponding tool load

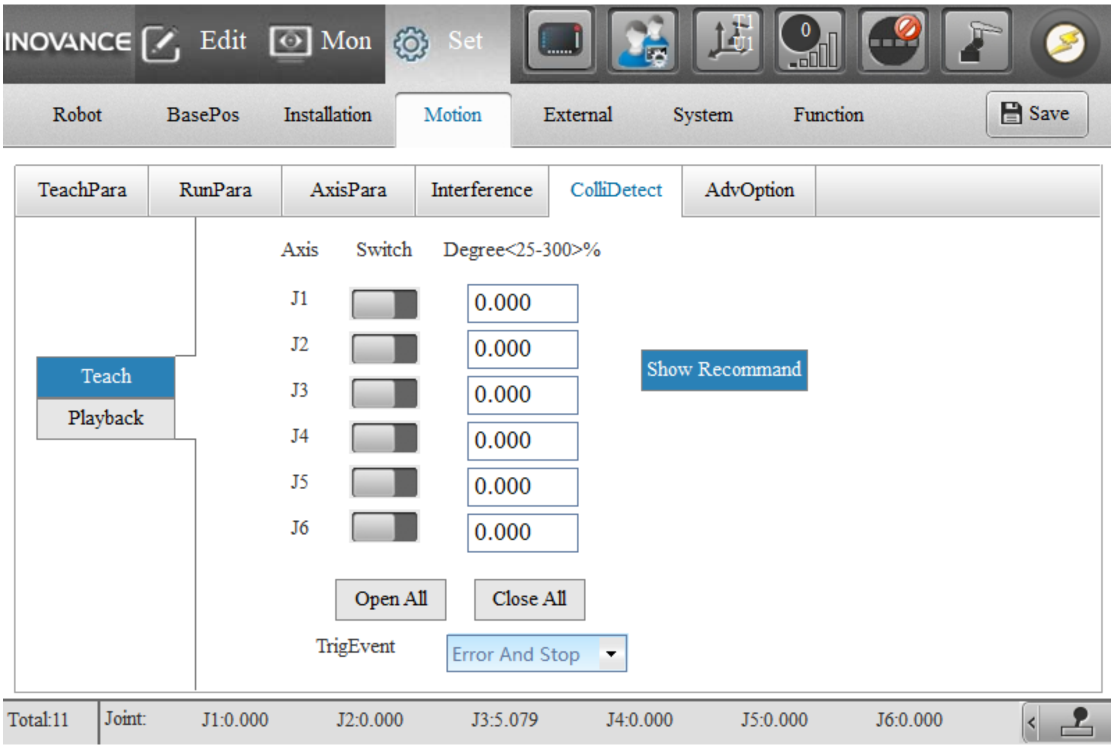
There are two ways to activate a tool load: a) Add the corresponding tool number to the motion instruction, which takes effect in the playback and in the single-step or continuous operation, and b) Switch to the corresponding tool number in the **Coordinate** interface, which takes effect in the jog state.

3） Activate the corresponding grip load

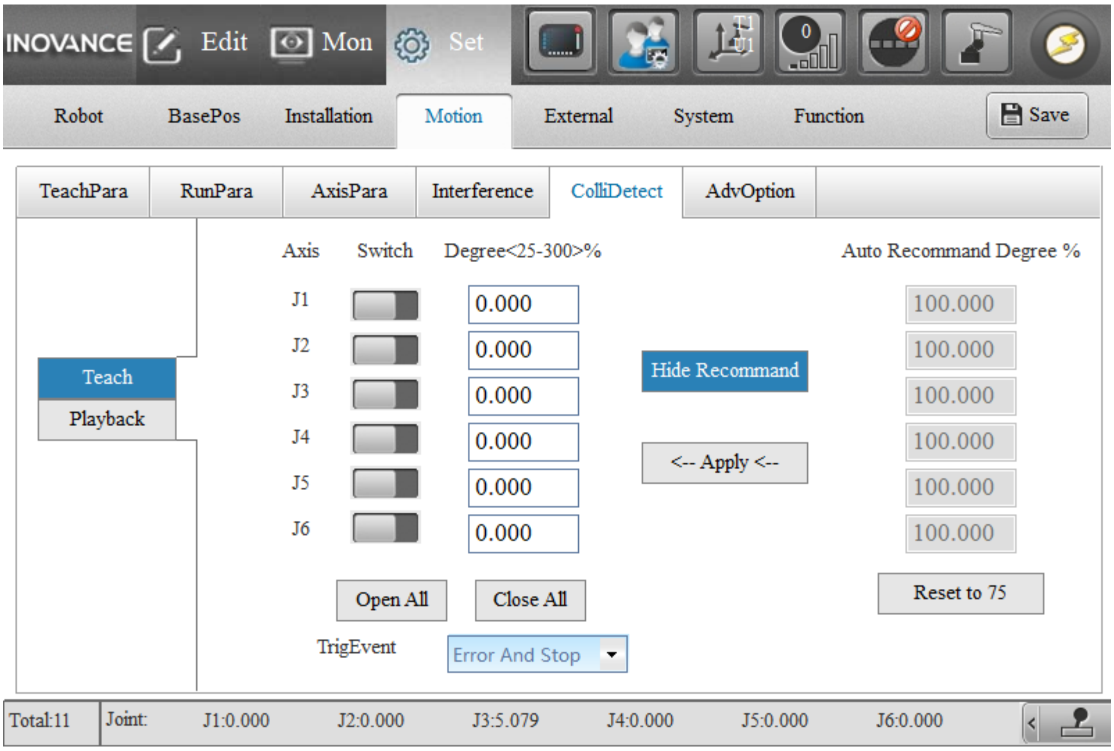
There are two ways to activate the grip load: a) Activate the specified grip load in the program via the instruction GripLoad, which takes effect in the playback and in the single-step or continuous operation, and b) Switch to the corresponding grip load number in the **Monitor** interface, which takes effect in the jog state (if the grip load number is not changed in the program using the instruction GripLoad, the grip load number set in the **Monitor** interface is always in effect).

4） Set collision detection parameters for the teach mode

Turn off collision detection switch for the teach mode, click the **Save** button. Then click **Show Recommend** to show the recommended sensitivity values.



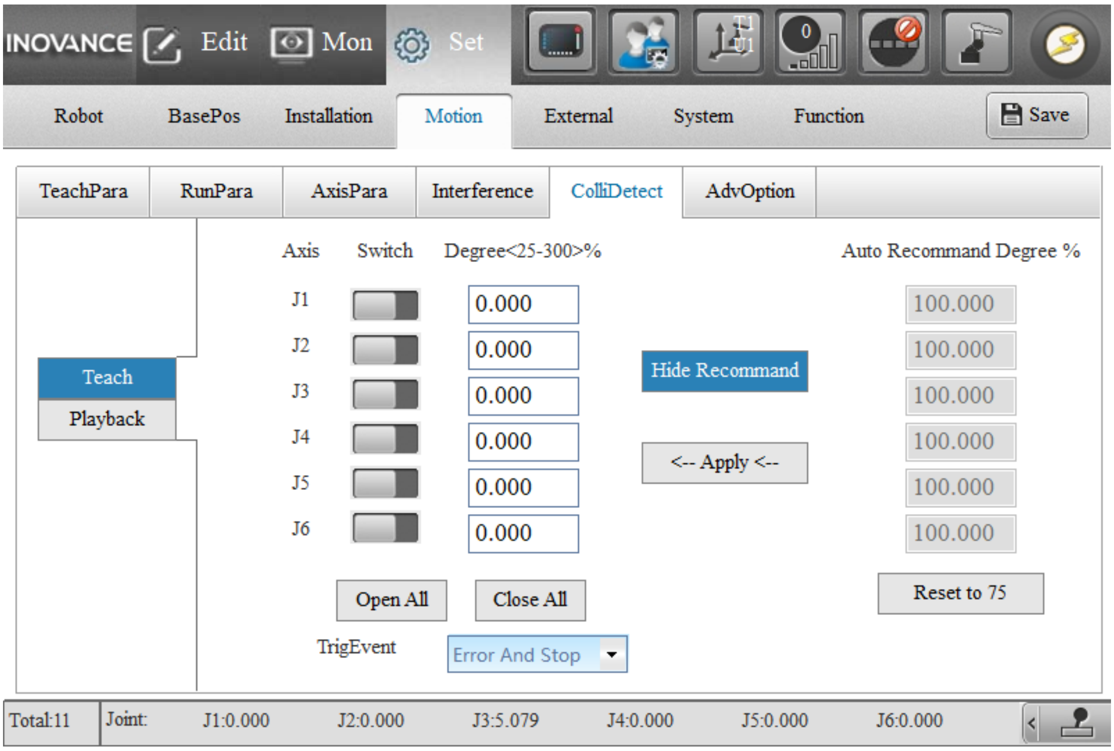
After a period of normal motion in the teach mode, return to the collision detection parameter setting interface. Click **Apply** to write the recommended values, then turn on the collision detection switch, and click the **Save** button.



Note: The recommended sensitivity is only updated upon motion in the teach mode. If the recommended value is less than 100, the interface will keep refreshing until the recommended value is greater than 100. If the recommended value is found to be less than 100, check if motion has been performed in the teach mode or if the motion has been performed for sufficient time (approximately 30s; recommended more than 2 minutes).

5） Set the collision detection parameters for the play mode

Similar to the commissioning process of collision detection parameters for the teach mode, first turn off the collision detection switch for the play mode and then click the **Save** button. Then click **Show Recommend** to show the recommended sensitivity values.



After a period of normal motion in the play mode (recommended for more than 5 minutes), return to the collision detection parameter setting interface. Click **Apply** to write the recommended values, then turn on the collision detection switch, and click the **Save** button.

If a collision alarm occurs, proceed as follows:

1) Check that the parameters are correct. i) Check if the collision detection sensitivity is too low. False alarms are more likely to occur when the sensitivity is less than 100. Make sure all sensitivity values are greater than 100. ii) Check that the load matches the set tool load. Check that the correct tool number is activated. When false alarms occur in the playback and in the single-step or continuous operation, check that the instruction includes the tool number and whether the tool number is correct. Check that the corresponding tool load parameters are correct. Check that the eccentricity is set correctly, especially in case of high eccentricity. iii) Check that the load matches the set grip load. Check that the correct grip load number is activated, that the correct grip load number is activated correctly using the instruction GripLoad, or that the grip load number is correct on the **Monitor** interface; check that the corresponding grip load parameters are correct. Check that the eccentricity is set correctly, especially in case of high eccentricity.
2) If none of the previous parameter checks indicate a problem, check the robot operating conditions. i) Check whether there is a significant difference between the operating conditions when the sensitivity is set and the actual conditions when the alarm occurs. The required collision detection sensitivity may vary under different operating conditions. In addition, if the ambient temperature at the time of the alarm differs significantly from that when the sensitivity is set, it may also affect the collision detection. In this case, apply the latest recommended sensitivity. Ii) Check whether the robot comes into contact with the outside world during its movement. In the event of contact with the outside world, because the contact force is unpredictable and therefore the moment fed back to the robot's motor is unpredictable,

the collision detection sensitivity set can no longer be used. In this case, it is recommended to use instruction SetAxisCollLevel to temporarily increase the detection sensitivity near instructions that may cause the robot to come into contact with the outside world in the program, or use the instruction SetAxisCollMode to temporarily turn off the collision detection function. This method works only in the play mode. iii) Check whether there is severe shaking when the robot is moving. Severe shaking greatly affects the collision detection and is prone to false alarms. In this case, it is recommended to increase the set sensitivity by 10 to 30 based on the recommended value.

3) Inspect the manipulator. i) Check that the robot base or end gripper is properly secured. ii) Check that the brake on the axis which produces the alarm is released. Turn off collision detection, run the alarm axis alone at low speed in teach mode, observe the current protection interface to see if the average load rate and maximum current of the axis are large. If so, check the servo parameters or cables of the brake. iii) Rotate the alarm axis vigorously and observe if there is a large gap in the robot. If the gap is large, it indicates that the reducer or timing belt is abnormal and needs to be disassembled for inspection. iii) If the previous inspection indicates no problem, it is possible that the reducer is damaged. Please export the fault information and contact the manufacturer for analysis.
4) If a collision alarm occurs when the collision detection switch is turned off, and all above investigations and attempts cannot solve the alarm, you can try to turn off the second level of collision detection. To turn off the second level of collision detection, set the collision detection sensitivity to 300 through user interface and turn off the collision detection switch through instructions or user interface. It is recommended that the settings be made only near instructions that are prone to false alarms and that the normal collision detection settings be restored as soon as possible in the subsequent programs.

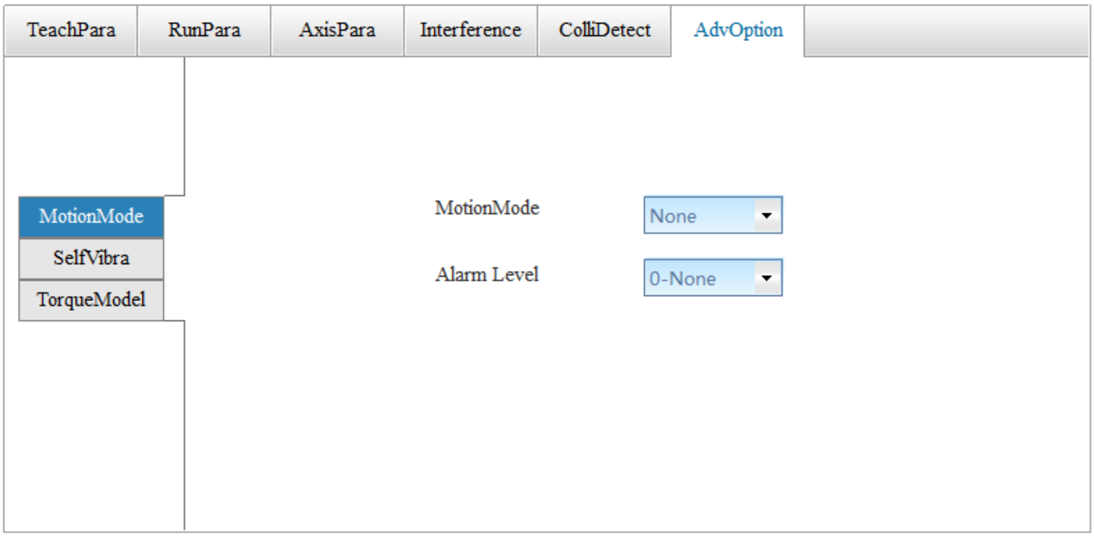
f) Advanced Functions

1 Closed-loop Vibration Suppression

The motion mode option is only available to the IRCB500 series controllers which are equipped with data acquisition board. The configurable parameters are as follows:

Vibration suppression mode: ON, OFF

Data acquisition board alarm level: 0-Alarm OFF, 1-Low sensitivity, 2-Medium sensitivity, 3-High sensitivity



Note:

When the data acquisition board is missing or damaged, the data acquisition board alarm level needs to be set to 0-Alarm OFF to avoid continuous alarms.

2 Self-learning Vibration Suppression

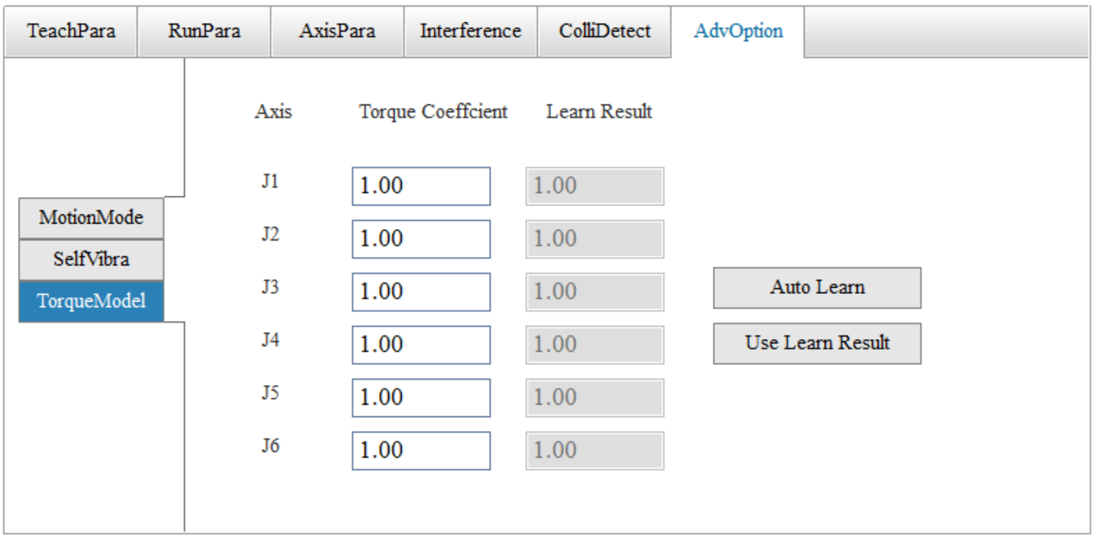
See 7.13 Self-Learning Vibration Suppression.

3 Torque Model Correction

When RapidMove is enabled, the system automatically generates motion trajectories based on the model. If the model is inaccurate, it will cause excessive current and cause an alarm. In this case, you can set the model error parameters to avoid the alarm.

Manual setting: If the overcurrent alarm still occurs after the load is set correctly, the torque correction factor for the corresponding axis can be set to a value less than 1. The smaller the value the lower the current and the slower the movement.

Note: This factor alters the joint output. When this factor is reduced, all RapidMove-enabled motion instructions will slow down. Therefore, if there is only one point with high current, try to reduce the local acceleration factor for the corresponding instruction first.



Automatic setting:
Automatic setting is to automatically identify the deviation between theoretical current and actual current based on the data recorded before the alarm, and automatically calculate the model error coefficient, as follows:
Step 1: Stop the robot after the overcurrent alarm occurs.
Step 2: Click the **Auto Learn** button and wait for the pop-up prompt that learning is complete.
Step 3: Click the **Use Learn Result** button and the learning results will be copied.
Step 4: Click the **Save** button to save the parameters to the project file and then run the program for verification.
Note that automatic learning is based on data from the period of motion prior to the alarm. The learning results only ensure that there will be no further alarms during this period of motion. There may be alarms during the execution of new motion and learning needs to be performed again.

4.5 Peripheral Settings

4.5.1 Bus Switch

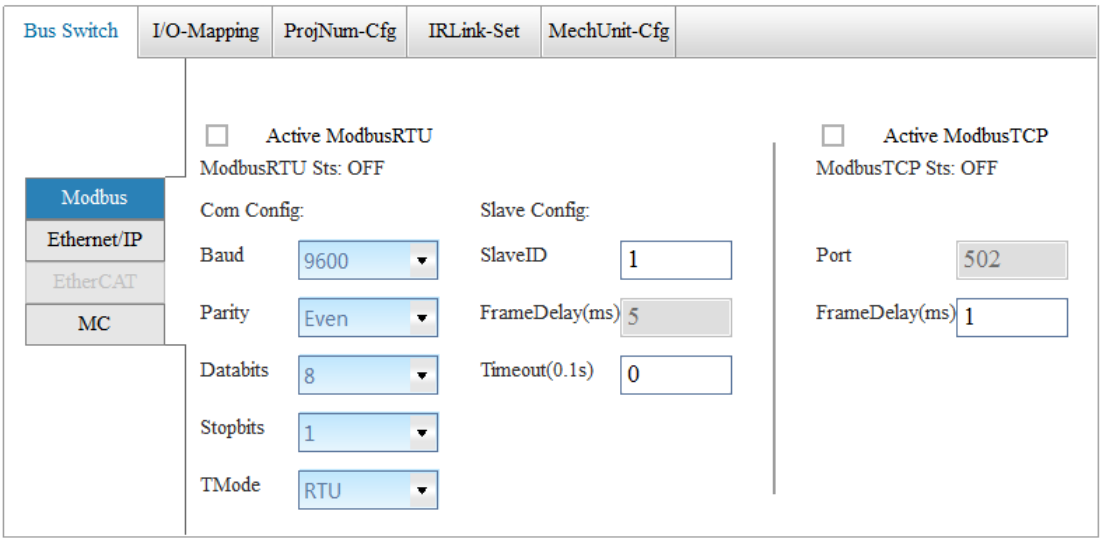
You can configure three fieldbuses including Modbus, Ethernet/IP and EtherCAT. **Note that only one fieldbus can be activated at a time**.

a) Modbus Settings

ModbusRTU and ModbusTCP can be activated at the same time for the robot controller.
Regardless of ModbusRTU or ModbusTCP, the robotic controller can only be configured in InoTeachPad and InoRobotLab as a Modbus slave, not as a Modbus master.

Operation Procedure

Go to **Set** > **External** > **Bus Switch** > **Modbus**, as shown below:



The status of ModbusRTU and ModbusTCP function is indicated. You can activate or deactivate the function and save the change to make it take effect.
In the ModbusRTU function, the frame delay is dependent on the baud rate. The larger the baud rate the smaller the frame delay.

In the ModbusTCP function, the port number is fixed to 502 and the frame delay can be set as needed.

There is no sequential requirement for parameter settings and the operation of function activation/deactivation. All configurations are saved to the controller only when you click the **Save** button.

Operation Result

After the operation is complete, the Modbus client can be used for communication to verify the configuration.

Note

1. If Modbus for the robot controller has already been configured through InoRobShop, it is not allowed to configure the Modbus through the teach pendant again. If you want to control via the teach pendant , there are two options:

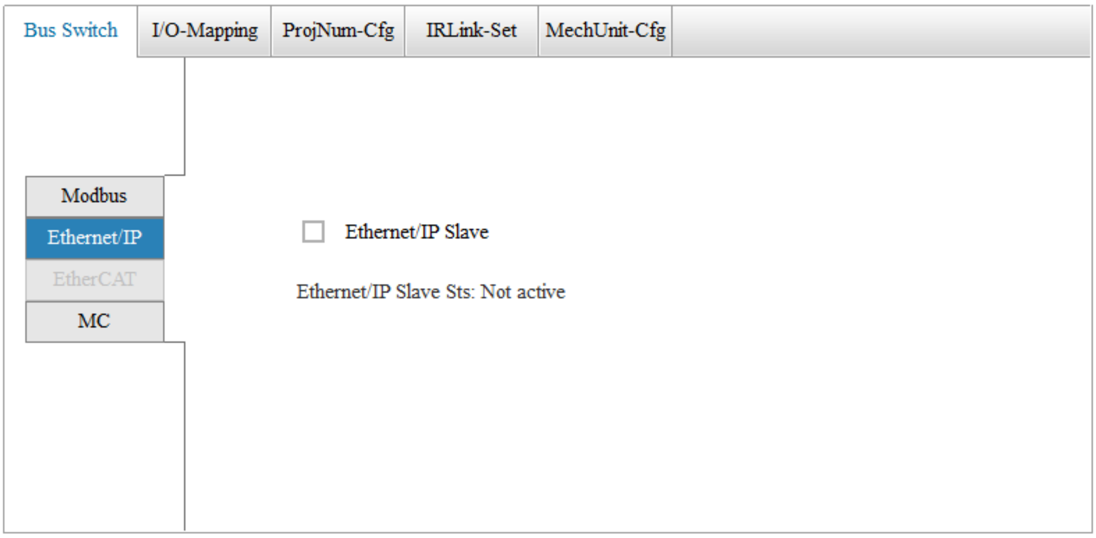
（1） Clear the configuration on the teach pendant using the "Clear PLC Settings" function;

（2） Download a secondary development project with no Modbus configuration via the teach pendant.

For more information on how to use the Modbus function, see the "Inovance Robot Modbus User Guide".

b) Ethernet/IP Settings

You can activate or deactivate the Ethernet/IP slave function of the robot. After configuration, click the **Save** button in the upper right corner to make the change take effect. The interface also shows the port number and connection status of the slave. When there is a master connection, a green light is visible and the IP address and port number of the master can be displayed.



For more information on how to use the Ethernet/IP function, see the "Inovance Robot Ethernet/IP User Guide".

c) EtherCAT Settings

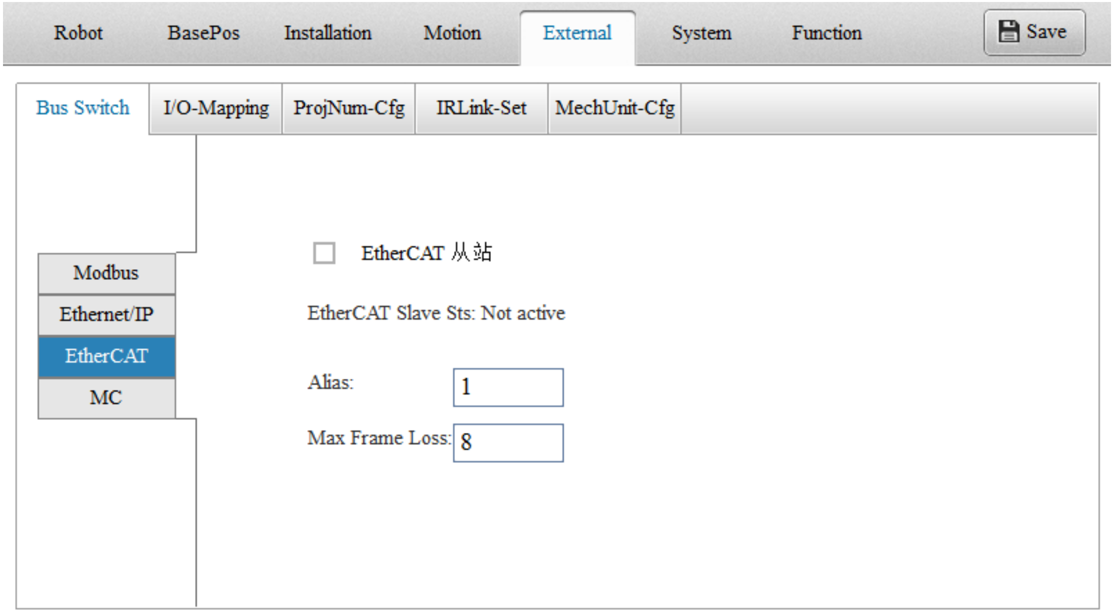
d) ***This feature is optional and can only be used on controllers that support the EtherCAT slave function.***

e) You can activate or deactivate the EtherCAT slave function of the robot and configure the relevant parameters.

f) After configuration, click the **Save** button in the upper right corner to make the change take effect. You can configure the alias and maximum number of frame drops for the slave.

g) It also displays the EtherCAT connection status. When not connected, red indicator is displayed. When there is a master connection, green indicator is displayed.

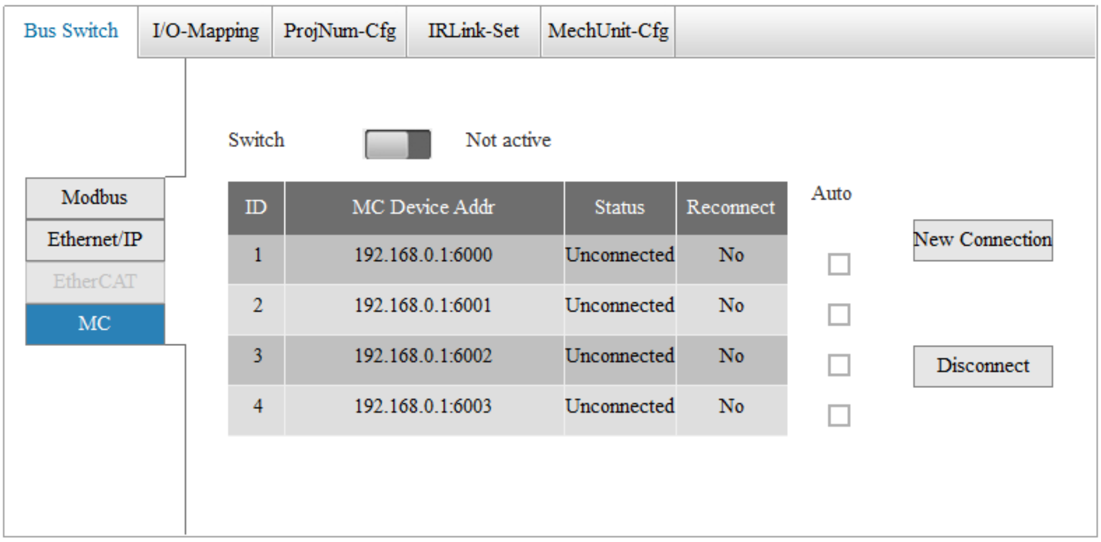
h) Note: After the slave alias is modified, the robot needs to be powered up again to make the change take effect.



i) For more information on how to use the EtherCAT function, see the "OMRON PLC and Inovance Robot EtherCAT User Guide".

j) MC Settings

You can activate the MC function of the robot controller to make the robot act as a slave. Currently, it supports connection to up to four masters.



Operation Procedure

You can select an address from the list and create a new connection or break the existing connection.

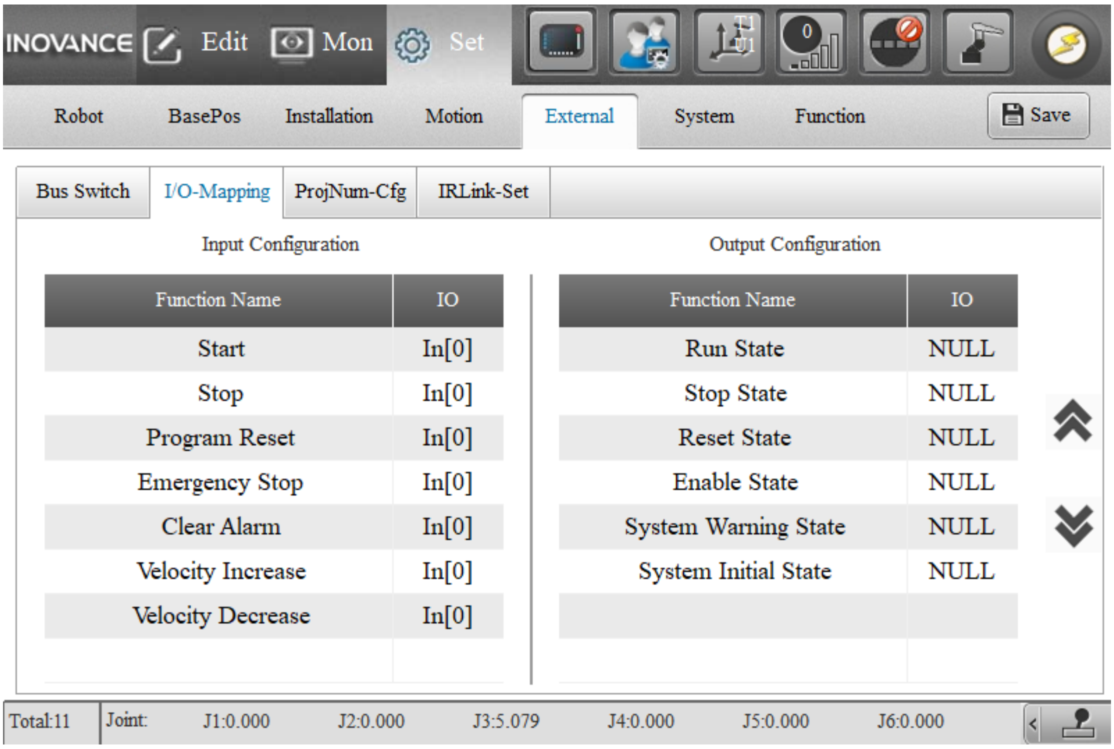
The MC function can be activated or deactivated directly by toggling the switch. To create a connection to an external MC device, click the **New Connection** button, enter the IP address and

port number of the MC device.
To disconnect the controller from the MC device, select the MC device and click the **Disconnect** button.

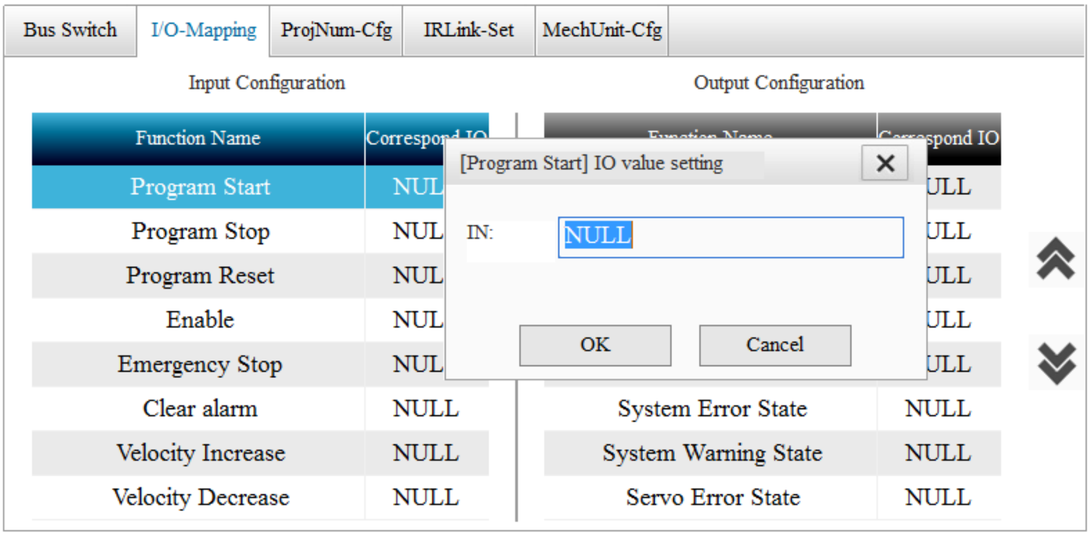
Note: The IP address and port number of the MC device cannot be modified after the controller successfully connects to the MC device. You can check the auto-reconnect option only after the controller successfully connects to the MC device. When the auto-reconnect option is selected, it is not allowed to modify the IP address and port number.

4.5.2 I/O Mapping

The remote I/O control function enables the control of program start/stop, reset, emergency stop of the robot with I/O (including digital I/O, fieldbus I/O and memory I/O), as well as monitoring of the execution and fault of the robot program.
The I/O mapping setting is to map some of the instructions in the remote I/O control function to the robot's In signal, the robot's state to the robot's Out signal, to achieve the purpose of controlling the robot with the In signal and monitor the robot through the Out signal.
Operation Procedure
Go to **Set** > **External** > **I/O-Mapping**, as shown in the following figure.



To modify the I/O mapping, click an I/O in the I/O column, enter the I/O number in the pop-up dialog. If you do not want to modify any I/O, enter "NULL".



For the teach pendant, only the I/O mapping for the following functions is supported:

I/O Function		Description
Input (Used for external control)	Start program	In Remote I/O control mode, start the current program, rising edge active.
	Stop program	In Remote I/O control mode, stop the program, rising edge active
	Reset program.	In Remote I/O control mode, reset robot program, rising edge active (can only be reset in stop state).
	Enable	In Remote I/O control mode, enable the robot, both rising edge (enable) and the falling edge (disable) active.
	Emergency stop	In Remote I/O control mode, place the robot into an emergency stop, non-emergency stop command at ON, and emergency stop command at OFF.
	Clear alarm	In Remote I/O control mode, clear the alarm, rising edge active.
	Increase velocity	In Remote I/O control mode, increase the global operating velocity of the system by 5% at a time, rising edge active.
	Decrease velocity	In Remote I/O control mode, decrease the global operating velocity of the system by 5% at a time, rising edge active.
	Homing	In Remote I/O control mode, return the robot to the work origin 0, rising edge active.
	Switch to teach mode	In Remote I/O control mode, switch the robot to the teach mode, rising edge active.
	Switch to play mode	In Remote I/O control mode, switch the robot to the play mode, rising edge active.
Output (Used for state display)	Program run state	In any control mode, the specified I/O outputs an ON signal when Task 0 is in running state.
	Program	In any control mode, the specified I/O outputs an ON signal

	stop state	when Task 0 is in stop state.
	Program reset state	In any control mode, the specified I/O outputs an ON signal when the program is reset successfully.
	Robot enabled state	In any control mode, the specified I/O outputs an ON signal when the system is enabled.
	Robot emergency stop state	In any control mode, the specified I/O outputs an ON signal when the system is in emergency stop state.
	System fault state	In any control mode, the specified I/O outputs an ON signal when the system has a fault.
	System warning state	In any control mode, the specified I/O outputs an ON signal when the system encounters an alarm.
	Servo fault state	In any control mode, the specified I/O outputs an ON signal when the servo has a fault.
	Servo warning state	In any control mode, the specified I/O outputs an ON signal when the servo encounters an alarm.
	Safety door warning state	In any control mode, the specified I/O outputs an ON signal when the safety door encounters an alarm.
	System startup completion state	In any control mode, the specified I/O outputs an ON signal when the system finishes startup.
	Robot motion state	In any control mode, the specified I/O outputs an ON signal when the robot is in moving state.
	Robot arrival state	In **Remote I/O** control mode, the specified I/O outputs an ON signal when the robot reaches the destination by executing the direct motion instruction.
	Bus communication heartbeat	In any control mode, as long as the I/O is configured, an ON-OFF state switch occurs continuously, with a default switch interval of 1s.

Note:
1. The teach pendant only supports the I/O mapping for some basic functions such as program execution and monitoring. For the I/O mapping for advanced functions such as position point modification, perform the configuration in InoRobotLab.
2. If you are prompted that a selected I/O is already in use but cannot find the function to which

the I/O is mapped, it indicates that the I/O is mapped to another function through InoRotLab. You can open InoRobotLab to view the configuration.

3. The input configurations in the I/O mapping take effect only in the **Remote I/O** control mode. The output configurations take effect in any control modes.

3.The output port selected in the I/O mapping will be occupied by the system and can no longer be controlled through the monitor panel or through the instructions.

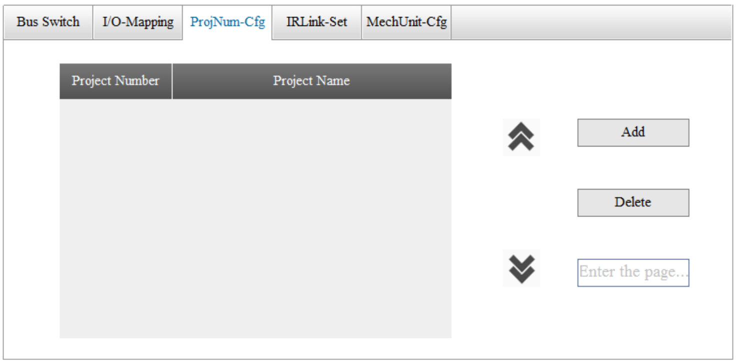
4. The run state and stop state in the output configuration refer to run state and stop state of the program, rather than start state and stop state of the robot motion. Once started, the robot will remain in motion until it is commanded to stop or stopped due to a fault.

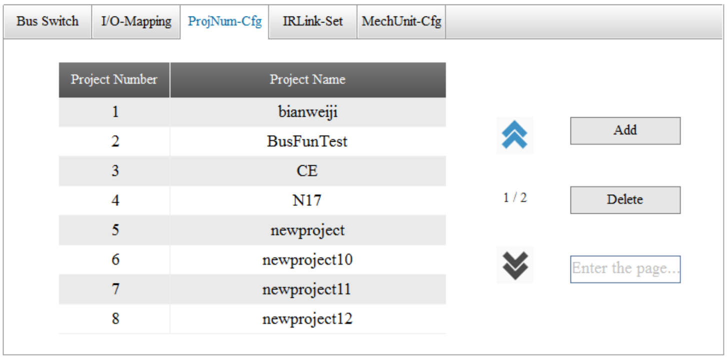
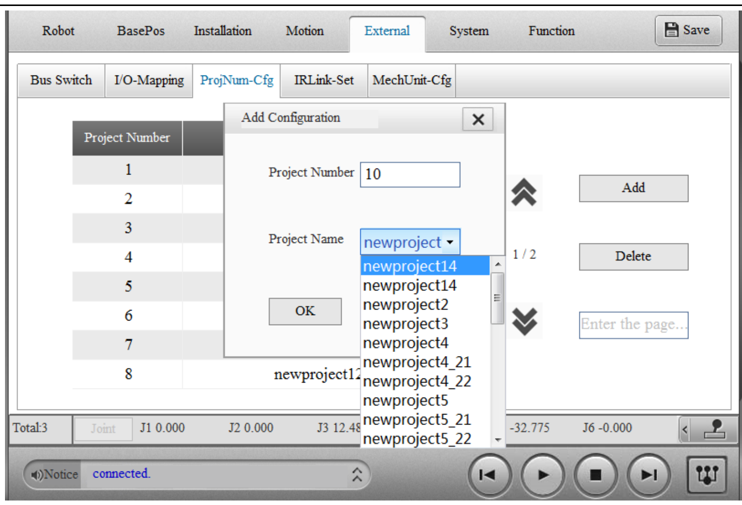
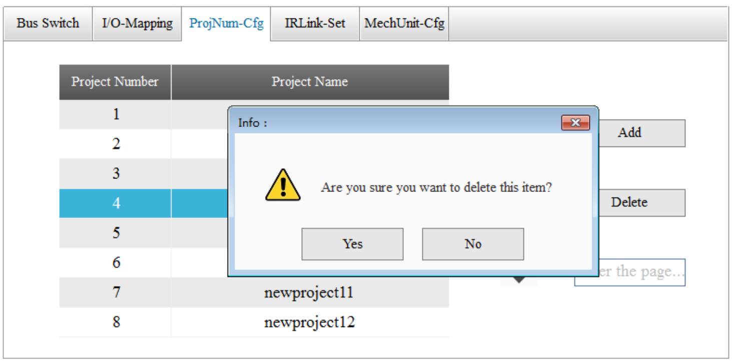
5. After the robot is placed into emergency stop through an In signal, if you switch the robot control to InoTeachPad, the emergency stop will be automatically released.

4.5.3 Project ID Settings

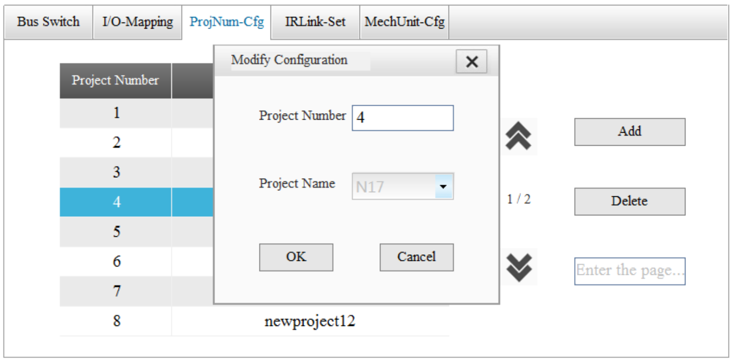
Different projects contain different recipes. If the robot is controlled by a device other than InoTeachPad, you need to switch the project via the field bus. The system includes 256 project IDs. You can set the project path corresponding to each ID in the I/O mapping.

Operation Procedure

Operation	Instance
Open project ID settings: Go to **Set** > **External** > **ProjNum-Cfg**. The teach pendant obtains the project configuration information from the controller.	

<table>
<tr>
<td>Display project information: The obtained project ID configuration information are displayed. The information may be presented by multiple pages. You can switch the pages and go to the specific page by entering the page number. If the entered page number exceeds the total number of pages, a prompt will be given.</td>
<td>
</td>
</tr>
<tr>
<td>Add a project ID: You can add a project ID and assign it to a project. By default, the project ID is automatically generated. Only Editor, Manager and Factory users can add the project ID. If you are not satisfied with the generated ID, enter another value that has not been used. You can select a project from the Project Name drop-down list. The list contains projects that have not been assigned with a project ID. If all projects have been assigned with a project ID, no new project ID can be added.</td>
<td>
</td>
</tr>
<tr>
<td>Delete a project ID: Select a project ID and click the Delete button.</td>
<td>
</td>
</tr>
</table>

Edit the project ID: Double-click a row to edit the selected project ID. One project ID uniquely corresponds to one project.



Note: All of the above settings need to be saved before they can take effect. The settings will be lost if you switch to another interface without saving them.

4.5.4 IRLink Settings

IRLink is a custom communication protocol used by the IMC100 series expansion modules to manage the networking and parameter settings for expansion modules such as robot I/O, AD, DA, and encoders. Configuring IRLink on the teach pendant requires IRlink configuration permissions. IRLink control is defaulted to the teach pendant or INOBOTLAB. Once IRLink has been configured via InoRobShop, IRLink control is transferred to InoRobShop. In this case, when you change the IRLink configuration via teach pendant or InoRobotLab, an error message will be given. To obtain the IRLink control on the teach pendant, cancel the IRLink configuration on the InoRobShop or use the "Clear PLC Configuration" function.

IRLink Configuration Specifications:

■ For the IRCB10 series controllers, you can add I/Os only through the expansion modules.

Each RTU (IRlink communications expansion module) has power consumption limits. The overall IRLink configuration is subject to resource limits.

■ For the IRCB300, IRCB100 series controllers, you can add I/Os through expansion cards and expansion modules.

Up to four expansion cards are supported, with no power consumption limits. The overall IRLink configuration is subject to resource limits.

■ For IRCB500 series controllers, you can add I/Os sequentially only via expansion cards (up to 4 expansion cards are supported).

Note: The expansion cards supported by the IRCB500 series controllers are different from those supported by the IRCB300 and IRCB100 series controllers!

(1) Power consumption specifications for the RTU

Each RTU can deliver 15W of power. The following table describes the power consumption of the respective modules.

Type	Power Consumption
IMC100-0808-ETND	1.44W
IMC100-1600-END	1.25W
IMC100-0016-ETPD	1.25W
IMC100-0016-ETND	1.25w
IMC100-4DA	1.44W

IMC100-8AD	2.88W
IMC100-2ENID	2.88W

Make sure the sum of the power consumed by the modules after each RTU does not exceed the power capacity of the RTU. If more modules are required, add a RTU and then cascade the modules after the RTU.

(2) Resource specifications

The total resources supported by the IMC100 are limited. Refer to the table below to configure the modules so that the total number of channels of the modules does not exceed the maximum capacity of the IMC100.

Object	Number of DI channels	Number of DO channels	Number of AI channels	Number of AO channels	Number of encoder channels
IMC100 capacity	**64**	**64**	**16**	**16**	**8**
Occupancy by one 1616 module	16	16			
Occupancy by one 0808 module	8	8			
Occupancy by one 1600 module	16				
Occupancy by one 0016 module		16			
Occupancy by one 8AD module			8		
Occupancy by one 4AD module			4		
Occupancy by one 4DA module				4	
Occupancy by one 2ENC module					2

Code	Expansion Module	Expansion Card	Drive-control integrated expansion card
0808	IMC100-0808-ETND		
1600:	IMC100-1600-END	IRCB-1600END-BD	IRCB500-1600END-BD
0016	IMC100-0016-ETPD IMC100-0016-ETND	IRCB-0016ETND-BD IRCB-0016ETPD-BD	IRCB500-0016ETND-BD
8AD	IMC100-8AD		
4AD		IRCB-4AD-BD	
4DA	IMC100-4DA	IRCB-4DA-BD	
2ENC	IMC100-2ENID	IRCB-2EN1D-BD	IRCB500-2ENID-BD

1616: IRCB-1616ETND-BD, standard inside the IRCB300 series controllers.

Note:

For IRCB300 series controllers, the system is equipped with one 1616 module as standard.

For IRCB100-6AT series controllers, the system is equipped with the following three modules as standard: 1616, 0016, 1600.

For IRCB500 drive-control integrated controllers, the system does not have a standard IRLink module.

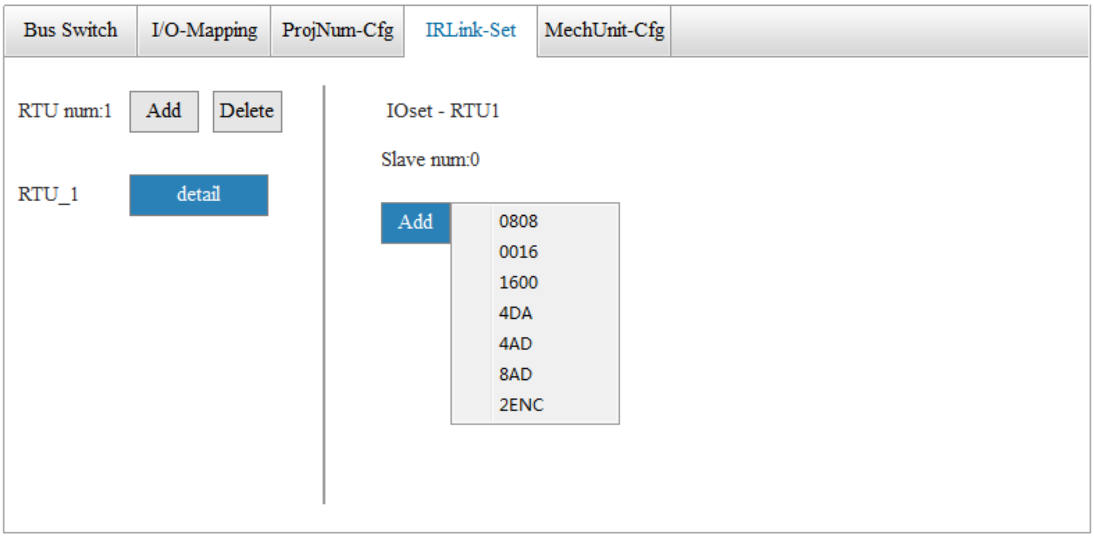
IRLink Configuration Method

After the hardware device is connected, you can make IRLink configuration in InoTeachPad.

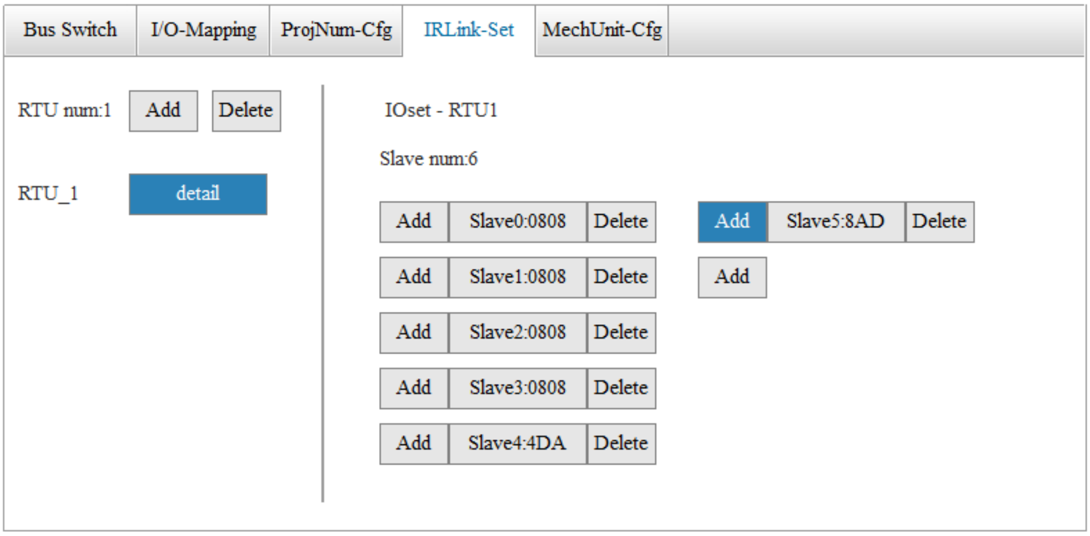
Click the **Add** button on the left. An RTU is generated automatically and the details of the RTU are displayed on the right. You can add up to five RTU expansion modules.

Click the **Add** button on the right, it will pop up a box with 0808, 0016, 1600, 4DA, 8AD, 4AD, 2ENC.

Add an expansion module according to the actual connection.

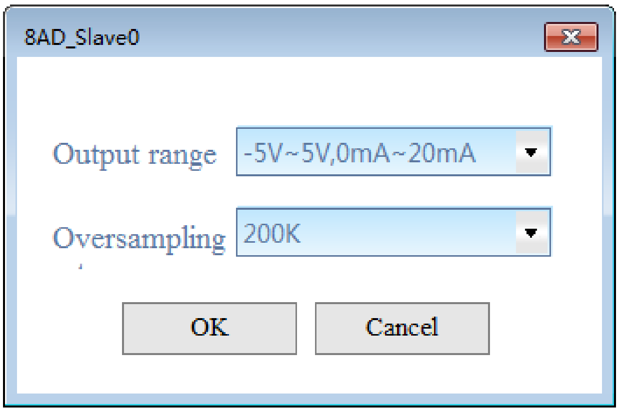
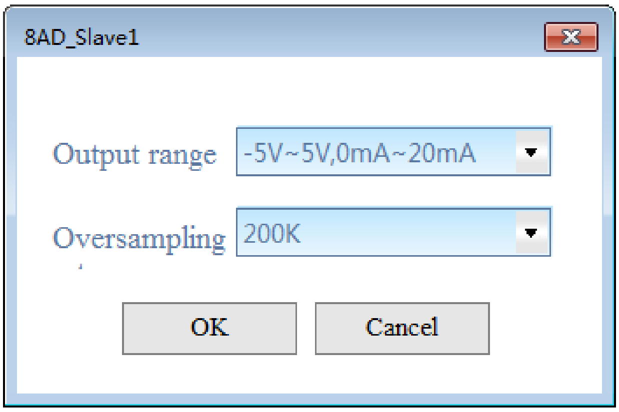
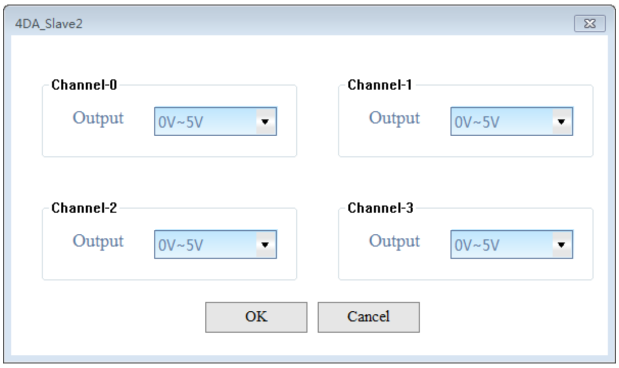


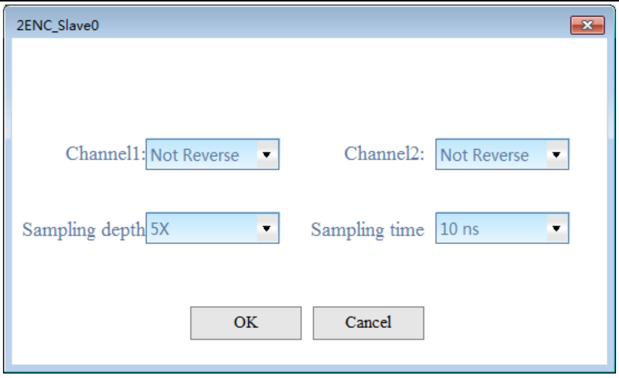
For example, add four 0808 modules, one 4DA module, and one 8AD module, as shown in the following figure.



For the 8AD, 4DA, 2ENC modules, click the module to access the configuration page.

Module	Setting Dialog	Parameters

8AD (4-channel voltage, 4-channel current analog conversion input expansion module)		You can configure the analog input range and the oversampling rate. The same configuration applies to all channels. The range is -5V-5V, 0-20mA, and -10V-10V, 0-40mA. The over-sampling rate affects the sampling accuracy of the input value. The smaller the oversampling rate, the higher the sampling accuracy.
4AD (2-channel voltage, 2-channel current analog conversion input expansion module)		Input range: -5V-5V, 0-20mA; -10V-10V, 0-20mA. Oversampling rate: 3.125K, 6.25K, 12.5K, 25K, 50K, 100K, 200K. Recommended: 3.125K
4DA (4-channel voltage/current analog conversion output expansion module)		You can set the range of analog output for the four channels.

2ENC (2-channel differential input incremental encoder expansion module)		You can set whether to reverse the channels and the signal filter time. The filter time is the product of the sampling depth and the sampling time. The longer the filter time, the lower the rated signal input frequency.

References:
For IRLink configuration for IRCB10 series controllers, see IMC100 Series Controller Local Expansion Module User Guide.
For IRLink configuration for IRCB300 series controllers, see IRCB300 Series 4-Axis Robot Controller User Guide.
For IRLink configuration for IRCB300 series controllers, see IRCB300 Series 6-Axis Robot Controller User Guide.
IRLink configuration for IRCB500 series controllers, see IRCB500 Series Robot Controller User Guide.

4.6 System Settings

System settings include communication settings, time and date, user settings, language selection, system features, and other settings.

a) Communication Settings

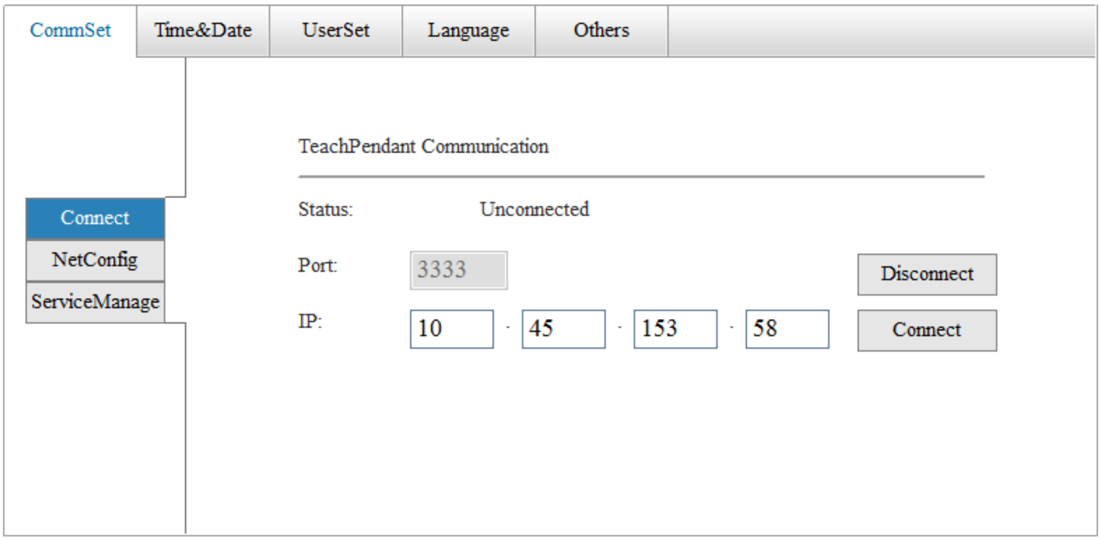
The communication settings include controller connection, network configuration, and communication service management.

Controller Connection

The connection between the teach pendant and the controller can be made in two ways.

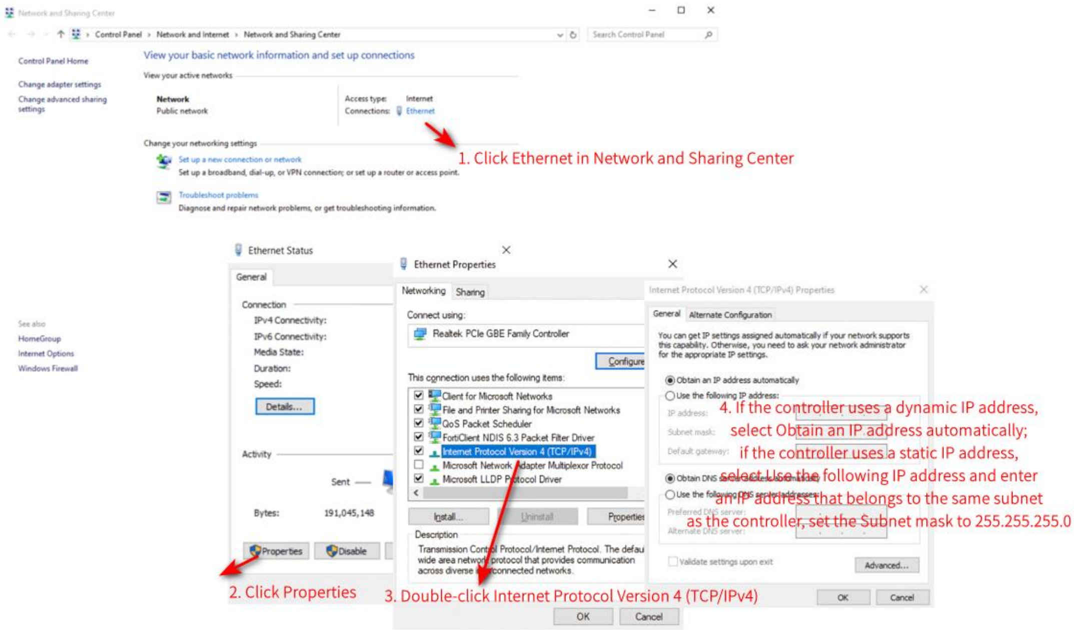
Connection Type	Connect Method	Characteristics
Connection to Ethernet port 2 of the controller (the IP address of Ethernet port 2 is fixed, 192.168.23.25)	For the hand-held teach pendant, plug its connector into the TP port of the controller	The IP address of Ethernet port 2 is static, 192.168.23.25
	For the PC-based teach pendant, connect the PC where it is installed to the PC port of the	The IP address of Ethernet port 2 is static, 192.168.23.25 The IP address needs to be set on the PC so that it is in the same subnet as the controller. For example, since the controller IP is fixed at

	controller via a network cable.	192.168.23.25, you can set the IP address of the PC to 192.168.23.26.
Connection to Ethernet port 1 of the controller	For the PC-based teach pendant, connect the PC where it is installed to the Ethernet port of the controller (or LAN port on some controllers) via a network cable.	The controller IP can be set to either dynamic IP or static IP, see Network Configuration. The IP address needs to be set on the PC so that it is in the same subnet as the controller.



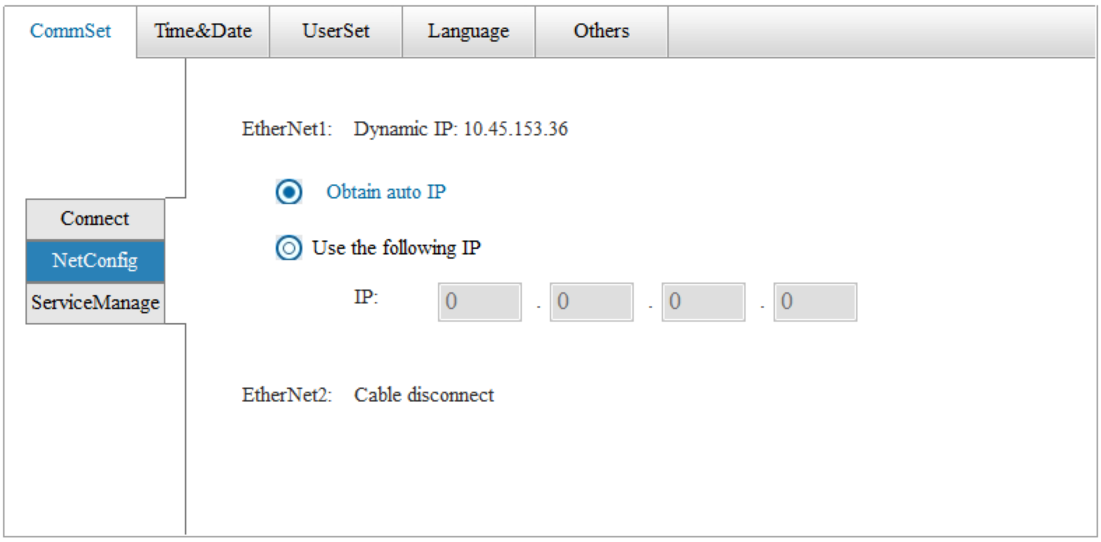
Setting IP address for connecting to the controller

For the PC-based teach pendant, configure the IP address of the PC where it is installed so that it is in the same subnet as the controller. The following figure shows the modification of PC's IP address.



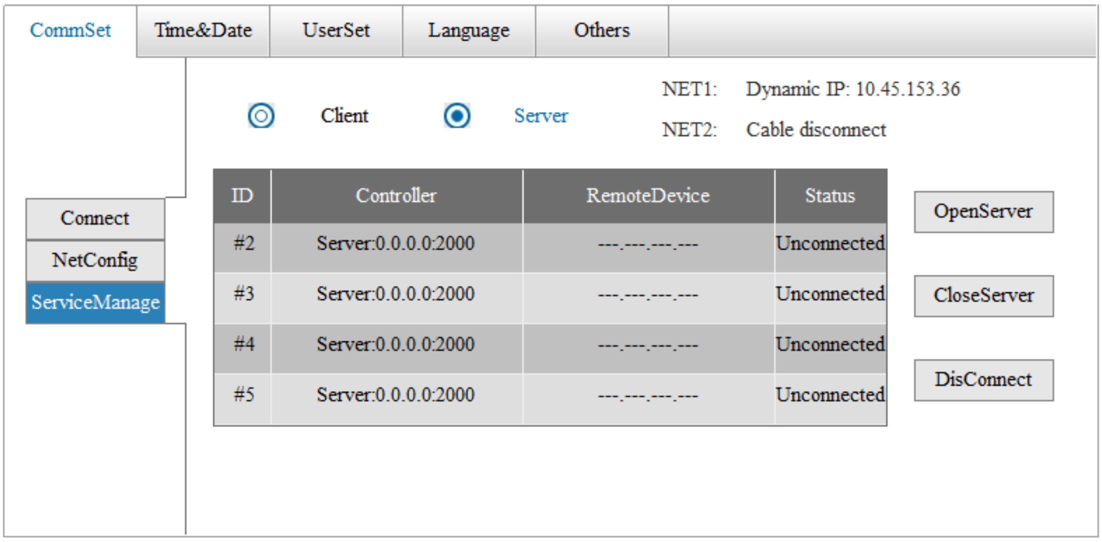
Network Configuration

When you need to modify the IP address of Ethernet port 1, do as follows.



Communications Service Management

It allows you to manage the socket configuration when the controller serves as a server or a client.



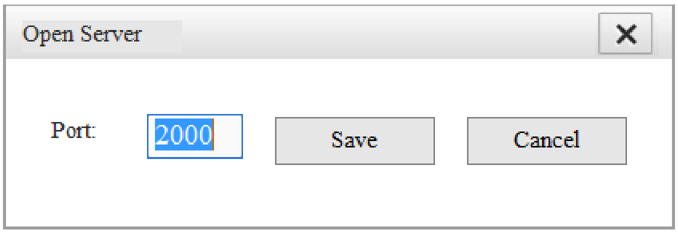
Description of client/server:

The controller can act as a client or server during communication with an external device. This function is mostly used in vision process.

- Server:

The controller acts as a server and supports connection of up to four clients.

Open Server: Opens the server function by specifying the port number of the controller.



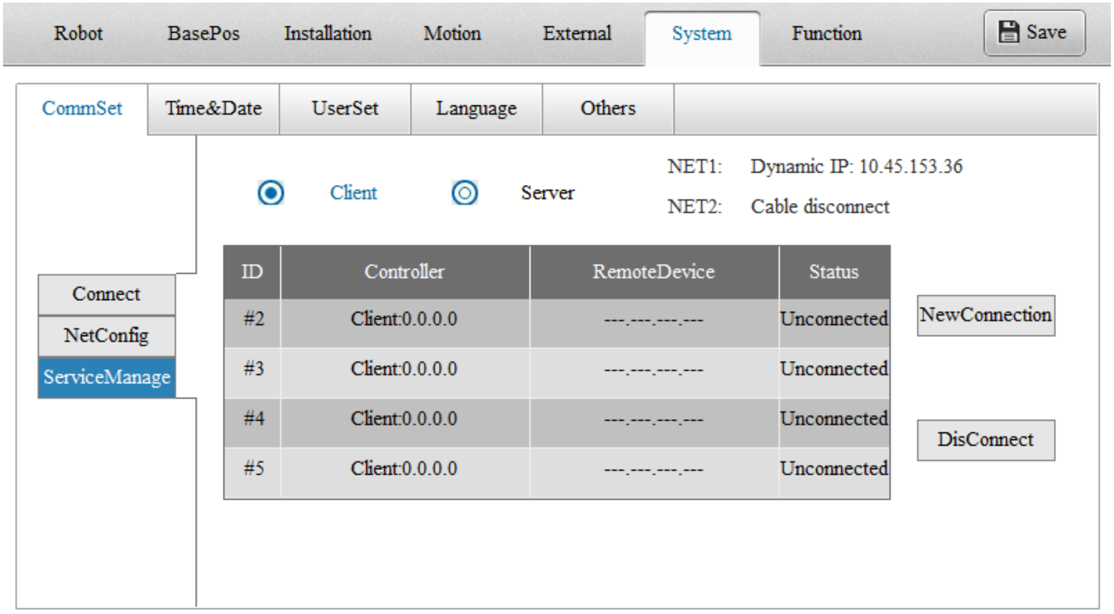
Note: The controller as a server supports connection of up to four clients. The server function is open on the controller by default and the default port is 2000.

Close Server: Turns off the server function of the controller.

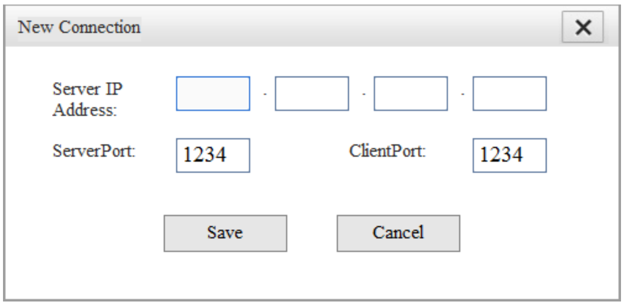
Disconnect: Disconnects the specified client from the server.

- Client:

The controller acts as a client and an external device acts as a server.



New Connection: Connects the controller as a client to the server by specifying the server IP, server port, and client port.

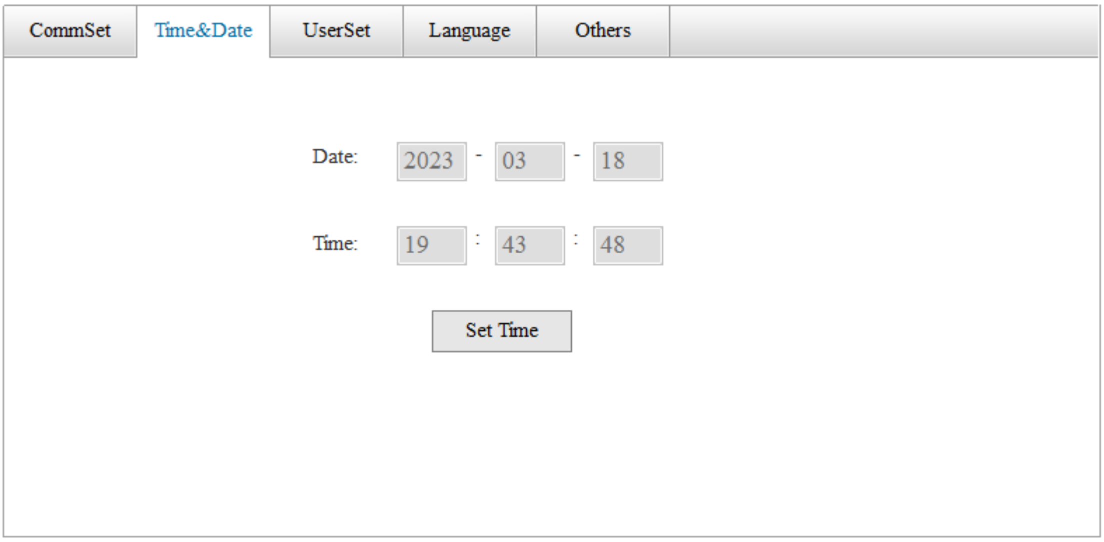


Disconnect: Breaks the selected connection.

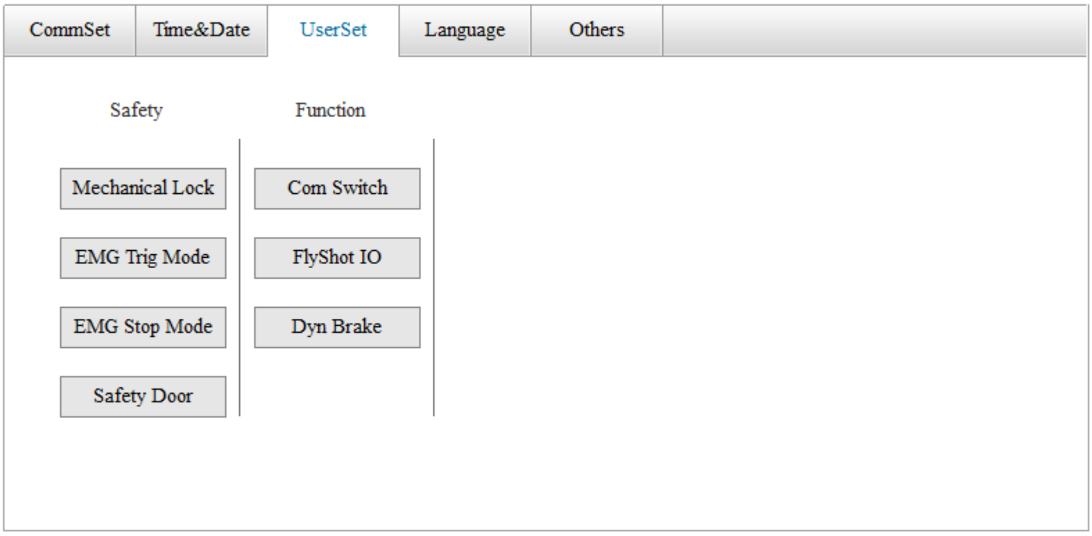
Note: The camera communication configured in **Function** > **VisionCalib** is process-specific communication and is not displayed in the **ServiceManage** interface.

b) Time and Date

This page displays the time and date in the controller.
Adjust the controller time via the **Set Time** button.

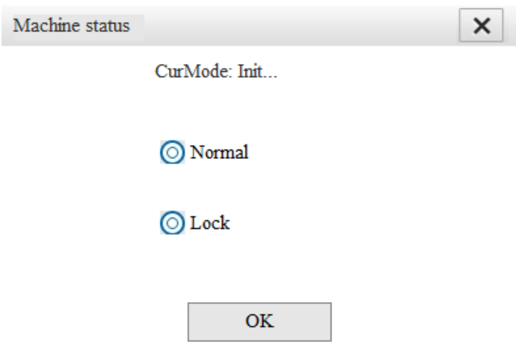


c) **System Functions**



Mechanical Lock

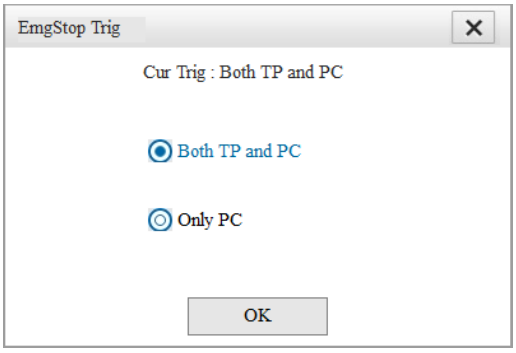
The robot cannot move in the mechanical lock mode.



Emergency Stop Trigger Mode

For IRCB500 series controllers, you can configure the emergency stop to be triggered via both

PC-based teach pendant and handheld teach pendant, or only via PC-based teach pendant.
When using a handheld teach pendant, select **Both TP and PC**.
When using PC-based teach pendant, select **Only PC**.

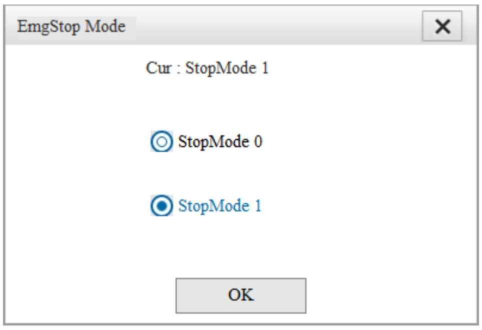


Note:

When you want to switch from PC-based teach pendant to handheld teach pendant, select **Both TP and PC** and you will be prompted whether to disconnect InoTeachPad. Select **Yes**. The PC-based teach pendant is automatically disconnected.
When you want to switch from handheld teach pendant to PC-based teach pendant, select **Only PC** and you will be prompted whether to disconnect the handheld teach pendant. Select **Yes**. The handheld teach pendant is automatically disconnected.

Emergency Stop Mode

For IRCB500 series controllers, the emergency stop mode can be configured to a Category 0 stop or a Category 1 stop.
Category 0 stop: Stop in an uncontrollable manner through hardware circuit by immediately removing the power to the motor, .
Category 1 stop: Stop in a controllable manner according to software planning.

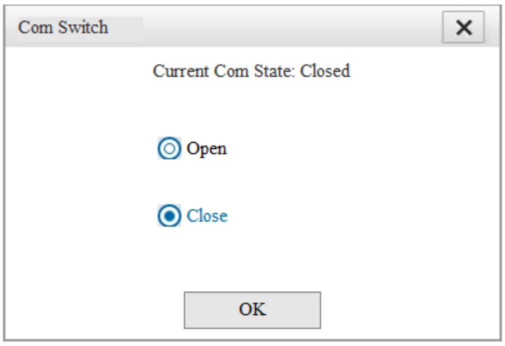


Safety Door

See 7.9 Safety Door.

COM Switch

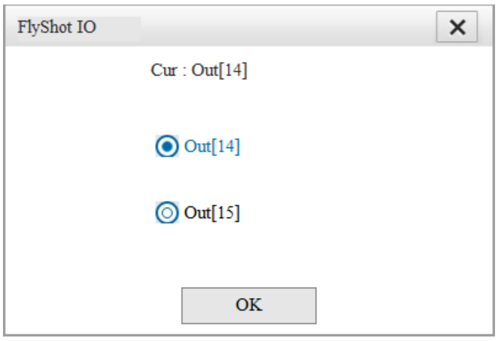
Open the COM switch before you can use the instruction Open Com. Restart the controller to make the setting take effect.



Flying Trigger I/O

For the IRCB500 series controllers, the flying trigger-related I/Os can be configured.

User I/O Out[14] or Out[15] can be selected as the output port that triggers the servo latch function.



See 7.6 Flying Trigger.

SN Match (Reserved)

For IRCB500 series controllers, the SN match function is available. This function is used to protect the encoder from reversed polarity. When the system is powered on and started, the servo drive SN and motor SN combination recorded in the controller is matched with the read SN combination. An alarm is generated if the match fails. This alarm cannot be cleared and the system needs to be repowered after recovery.

The SN match function is off by default. A switch is provided to turn SN match on or off.

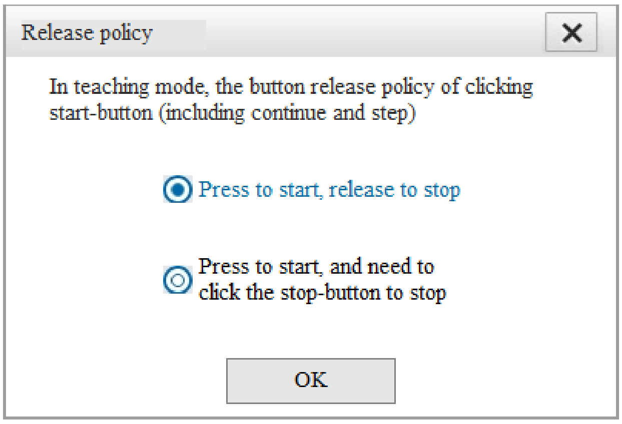
Note: There are two prerequisites for using this function:

1. The motor used supports SN reading.
2. The SN code has already been reset by clicking the **Reset SN** button.

Work Mode of Start and Forward Buttons

This function configures how the Start button  and Forward button  work when they are pressed and released in debug mode.

Click **Release policy** and the following pop-up dialog appears.



Two modes are available:

(1) Mode 1: Press to start and release to stop

(2) Mode 2: Press to start, no stop at release and the stop button must be pressed to stop

Note: Mode 1 is selected by default.

- To change the mode:

（1） Select the mode.

（2） Click the **OK** button.

（3） When the notification bar indicates success, the save is successful.

- Description of the modes

(1) Mode 1

: When you have completed program editing and switched to the debug mode, press this button to start the program and release the button to stop the program. If you want the program to run until it ends, keep the **Start** button pressed.

: When you have completed program editing and switched to the debug mode, press this button to execute the line of instructions where the cursor is located and release the button to stop the execution. To run the current line of instructions completely, keep the **Forward** button pressed.

(2) Mode 2

: When you have completed program editing and switched to the debug mode, press this button to start the program and the program will run until it ends if the **Stop** button is not pressed.

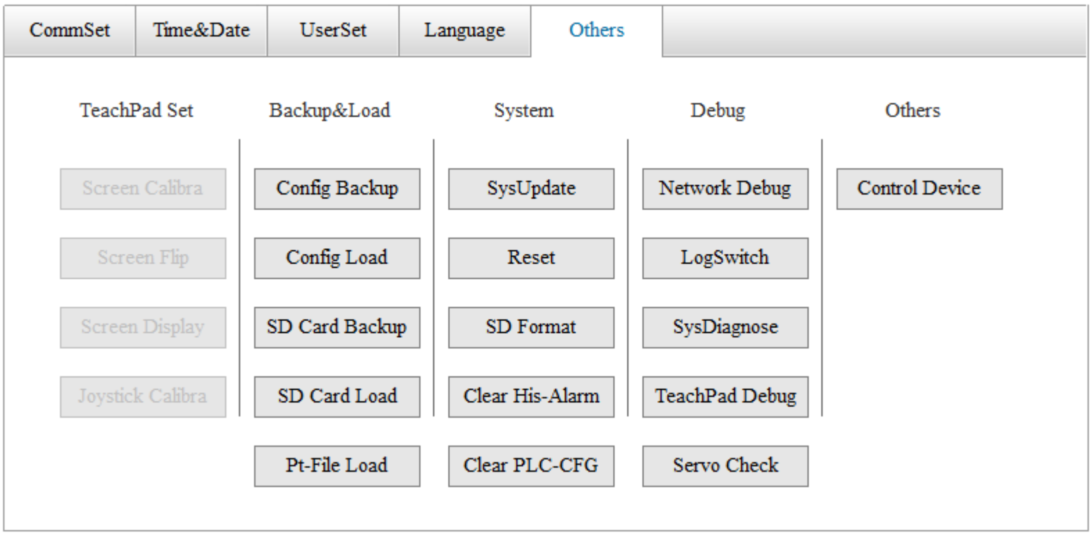
: When you have completed program editing and switched to the debug mode, press this button to execute the line of instructions where the cursor is located and the line of instructions will be fully executed if the **Stop** button is not pressed.

- Note:

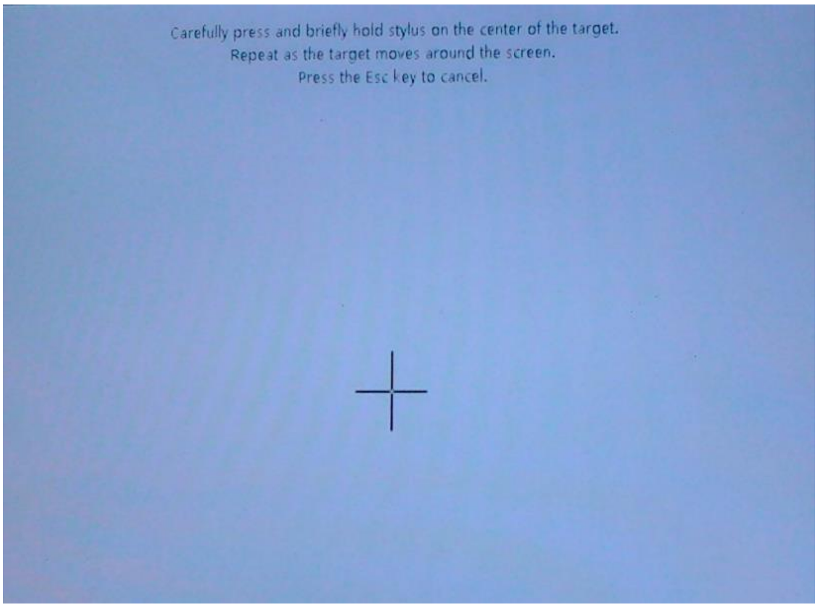
When a non-static task is running, the work mode cannot be modified.

d) Other Settings

Other settings include TeachPad Set, Back&Load, System, Debug and Others. See the following figure. TeachPad Set is valid for handheld teach pendant only.



Screen Calibration: Solves the problem of inaccurate touch on the touch screen. Use the stylus to tap the "+" symbol that appears in sequence on the screen, then tap any blank space on the screen to complete the screen calibration.



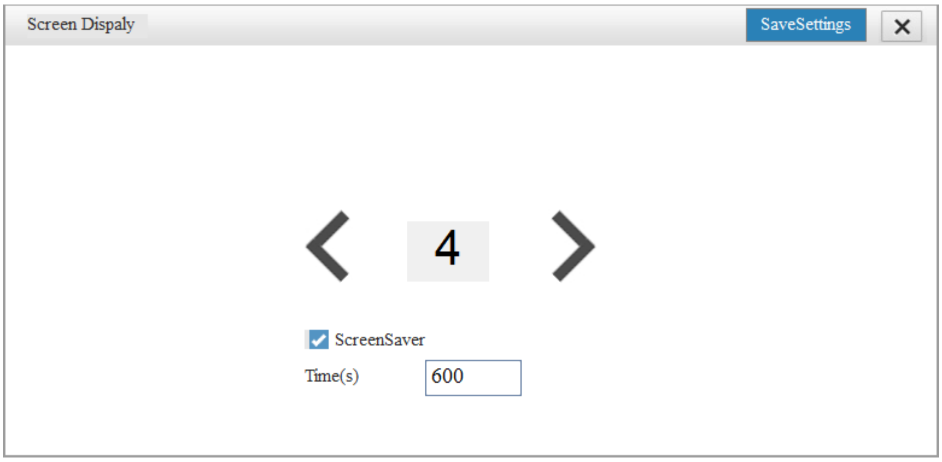
Note: You can also access the screen calibration by pressing the **External Axis** button for 10 seconds. Screen calibration is often required after the system is flashed.

Screen Flip: By default, the display of the teach pendant is suitable for holding the teach pendant with the left hand. For the ITP100 teach pendant, the screen can be flipped for the holding the teach pendant with the right hand.

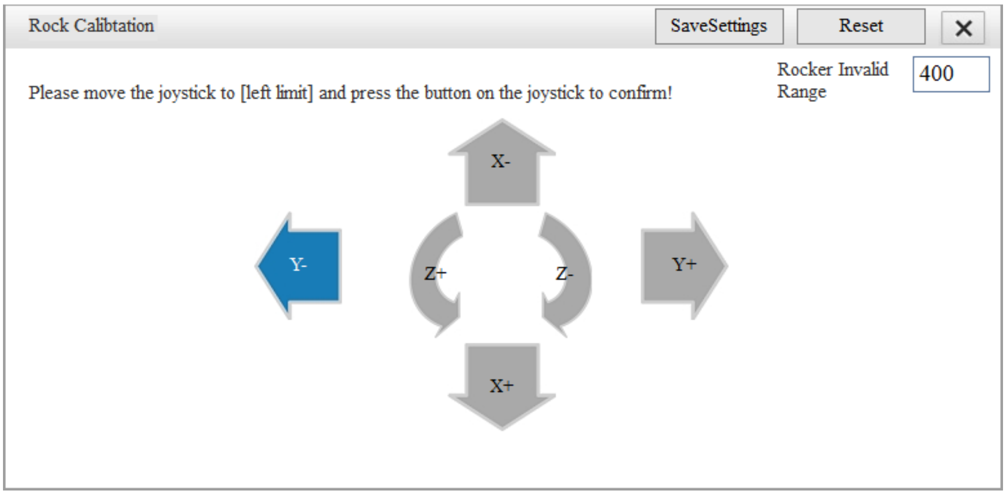
Note: After confirmation, the system automatically enters the screen calibration interface and you must re-calibrate the screen.

Brightness and Screensaver:

Brightness: A total of six brightness levels 1-6 is available.
Screensaver: You can activate the screensaver and set screensaver time.



Joystick Calibration: For ITP100 teach pendant only. Only the Factory user can calibrate the joystick. The joystick movement directions include left and right (Y-/Y+), up and down (X-/X+), clockwise/counterclockwise (Z-/Z+). You need to operate the joystick according to onscreen instructions.
Rocker Invalid Range: Controls the sensitivity of the joystick at the software level, 300 by default. The larger the value, the less impact your movements have; the smaller the value, the more impact your movements have. It is recommended not to change this value!



Config Backup
*All robot-related setting parameters are saved in the file RobotInfo.cfg, including the following parameters.

Config backup, add-ins	1. PLC program 2. Robot settings; zero point settings; installation parameters; motion parameters; peripheral configuration; and communication settings in system settings. 3. Robot user account 4. Robot model, controller information (only backup without loading, for robot and controller match check)

Insert a USB drive into the controller and click this button. The robot configuration file is backed up to the root directory of the USB drive.

Note:

Before operation, insert a USB drive into the controller and check the connection status of the USB drive. If the monitored communication status in the teaching software displays "The USB controller has been inserted into the device and successfully mounted", it indicates that connection is successful. Otherwise, check the connection. Keep the USB drive connected during operation.

Config Load

Prepare the configuration file RobotInfo.cfg under the root directory of the USB drive in advance. Insert the USB drive into the controller and click this button to load the file into the controller. Then power on the controller again to make the changes take effect.

Note:

(1) Before operation, insert a USB drive into the controller cabinet and check the connection status of the USB drive. If the monitored communication status in the teaching software displays "The USB controller has been inserted into the device and successfully mounted", it indicates that connection is successful. Otherwise, check the connection. Keep the USB drive connected during operation.

(2) Loading configuration files of different robot models is not allowed.

SD Card Backup

The following describes the backup contents:

	Version 14 and earlier	Version 15 and later	Version 17 and later
SD card backup	1.TeachProgram 2.PalletInfo	1.TeachProgram 2.PalletInfo 3.TecParameter	1.TeachProgram 2.PalletInfo 3.TecParameter 4.robot_other_pfile
SD card load	1.TeachProgram 2.PalletInfo	1.TeachProgram 2.PalletInfo 3.TecParameter	1.TeachProgram 2.PalletInfo 3.TecParameter 4.robot_other_pfile

TeachProgram: A program folder that contains all the project files.

PalletInfo: A pallet folder that contains the pallet information, and imported external point files ".pt ". It is required when pallet variables are used.

TecParameter: A process folder that contains information about the screw locking and dispensing process.

robot_other_pfile: BRD global variables, global translation variables, string variables, global position variables P.

Insert a USB drive into the controller and click this button. The program in the SD card is backed up to the root directory of the USB drive.

Note:

(1) Before operation, insert a USB drive into the controller cabinet and check the connection between USB dive and SD card. If the monitored communication status in the teaching software

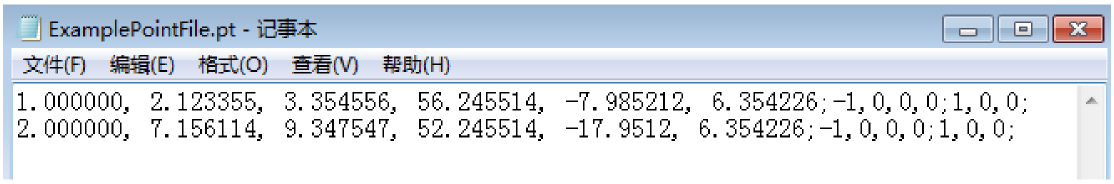
displays "The USB controller has been inserted into the device and successfully mounted" and "The SD card has been inserted into the device and successfully mounted", it indicates that the connection is successful. Otherwise, check the connection. Keep the USB drive and SD card connected during operation.
(2) Precautions for loading backup files of different versions:
S01.15 and later are referred to as the new version.
For files backed up prior to S01.15 and loaded to system S01.15 and later, they can be used normally, but the TecParameter will not be loaded (because it is not included in the backup file). The system prompt that the version is too low and requires S01.15 or above.
For files backed up in S01.15 and later and loaded to system prior to S01.15, they can be used normally, but the TecParameter will not be loaded.
The robot_other_pfile folder is backed up and loaded only in version 17 and later. If it is detected that the robot_other_pfile folder does not exist during loading, it will be ignored.
(3) Due to differences in file storage types under Windows and Linux, do not edit the backed up files, otherwise it may cause global string variables, I/O comments, etc. to show garbled characters after the program is loaded.

SD card load
Prepare the corresponding file under the root directory of the USB drive in advance. Insert the USB drive into the controller and click this button to load the program into the controller. When the load is complete, reboot the robot as prompted.

Note:
Before operation, insert a USB drive into the controller cabinet and check the connection between USB dive and SD card. If the monitored communication status in the teaching software displays "The USB controller has been inserted into the device and successfully mounted" and "The SD card has been inserted into the device and successfully mounted", it indicates that the connection is successful. Otherwise, check the connection. Keep the USB drive and SD card connected during operation.

Point File Load
Prepare a point file in a USB drive in advance. Insert the USB drive into the controller to load it to the robot control system.

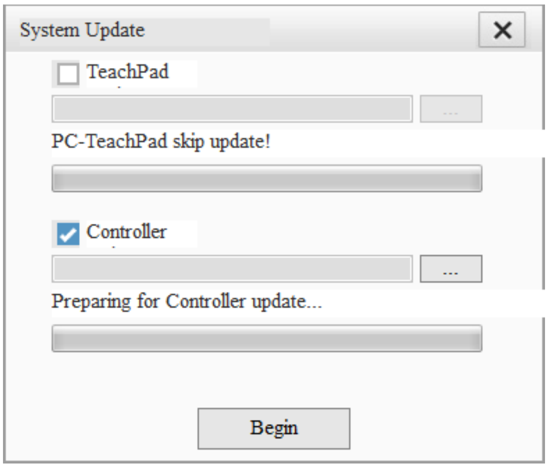
* The point file is suffixed with ".pt". The file contents are data information of position variables. One line indicates one piece of position variable information. For the format of every line, see the definition of position variables. Each line is divided into three paragraphs. The first six coordinate parameters compose the first paragraph. The middle four arm parameters compose the second paragraph. The last three parameters (coordinate system number, tool number, and user number) compose the third paragraph. The paragraphs are separated by ";", and the parameters in each paragraph are separated by ",".
An example is shown below:



System Update

Operation method:

- Select an item to update. The teach pendant and the controller can be updated simultaneously.
- Select the update package.
- Click the **Begin** button to start updating.



Note:

1. Update is applicable to the handheld teach pendant only, not to the PC-based teach pendant.

2. Do not cut off the power supply during controller update process, otherwise it may cause abnormalities and can only be restored by flashing the controller!

Reset

You can initialize all robot setup parameters to default.

Note: Reset with caution because the zero point of the robot will be lost.

SD Formatting

Format the SD card on the controller into a program storage card suitable for the robot. This operation will clear the program in the SD card. Therefore, it is recommended to back up the program before formatting.

Note:

Before formatting the SD card, go to **Monitor** > **Connection** to check the status of the SD card. If the monitored communication status in the teaching software displays "The SC card is connected and successfully mounted", it indicates that connection is normal. Otherwise, check the connection.

Clear Historical Alarm

Clear the operation records and alarm records in the monitoring interface.

Note: This operation clears both the operation records and the alarm records.

Clear PLC configuration

Click the **Clear PLC-CFG** button to clear settings made in InoRobShop.

Function:

Clears secondary developed PLC programs and various bus configurations, including external axis configurations.

- For PLC programs, they will be cleared.
- For EtherCAT configurations (including external axes), the configurations will be reset to default.
- For Modbus configurations, the configurations will be unlocked and retained.
- For IRLink configurations, in Version 18, the configurations will be unlocked and retained.

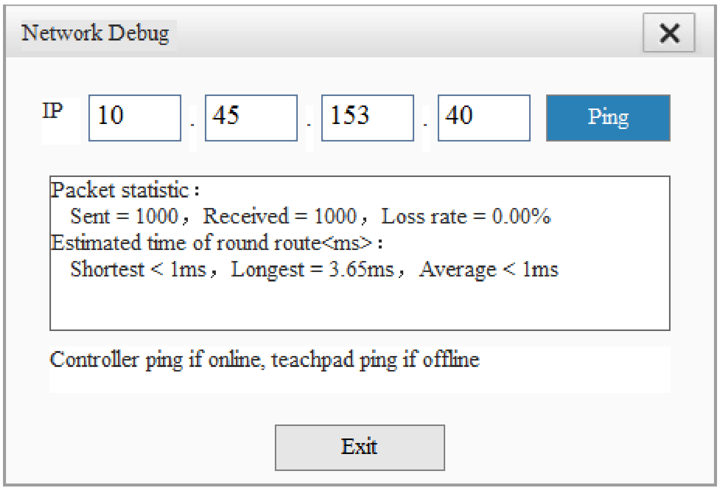
Typical use: IRLink and Modbus are configured for secondary development and cannot be edited. When cleared with this function, the configurations will be unlocked and you can edit the IRLink and Modbus.

Note: A restart is required to make the change take effect.

Network Debug

Debug communication of teach pendant and controller with other devices. It is equal to the Ping function.

Enter an IP address and click this button.



Note:

The controller pings other devices when the teach pendant is connected to the controller.

The teach pendant pings other devices when the teach pendant is not connected to the controller.

Controller Debug

During operation, the controller can output and record process information, monitor the status and flow of the controller for online viewing and post-analysis by the relevant personnel. Please operate under the guidance of the manufacturer!

Debug object:

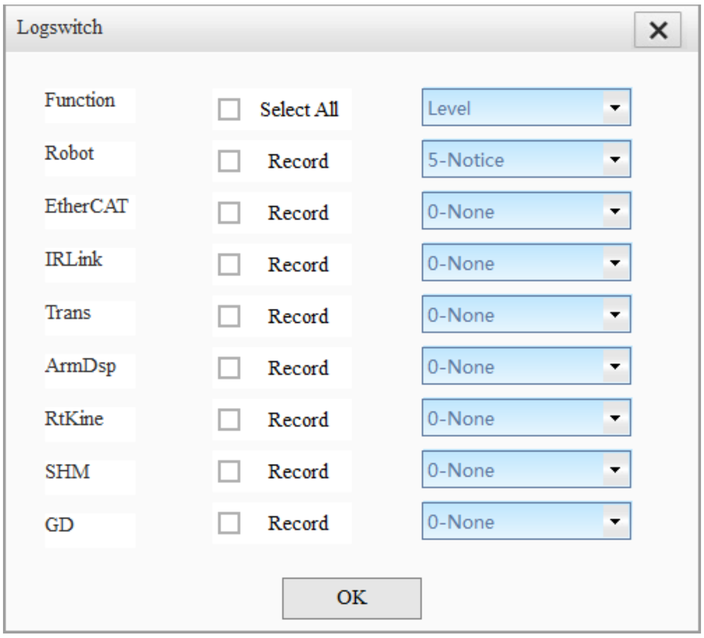
Name	Description
Robot	Core dispatch module
EtherCAT	EtherCAT communication module
IRLink	IRLink communication module
Trans	Language interpreter module
ArmDsp	ARM and DSP interaction module
RtKine	Kinematics module
ShM	Shared memory modules
GD	Configuration information module

Debug level:

For each debug object, you can select the level of logging, the more detailed the logging is at the higher level.

Record level	Description
0-None	Do not log
1-Alert	Only log the most serious errors
2-Critical	Log critical errors
3-Error	Log error-level events
4-Warning	Log warning-level events
5-Notice	Log notice-level events
6-Information	Log general events
7-Debug	Log all details

Use of debugging feature:



Click the **LogSwitch** button to open the log level selection interface. Check the object as needed and select the corresponding level. The settings take effect immediately upon confirmation.

(1) Online debugging: Prints information through the serial or TCP/IP port of the controller.

Serial port: Connect the controller through serial port. Note that the serial port on the controller is RS485. A RS232 to RS485 adapter is generally required.

TCP/IP: Connect the controller via another device (such as a PC) over TCP/IP protocol, port number 5555.

(2) Saving and exporting debug information:

The system automatically records the recent debugging information. In the **SysDiagnose** dialog, check **Path**, click **Begin**, and then click **Export** to save the debug information to the USB drive on the controller.

Teach Pendant Debug

For use by the manufacturer only.

System Diagnostics

Steps:

1) Check the diagnostic object.

The object includes System, Logic, and Path.

System: Contains internal configuration parameters, etc. of the system.

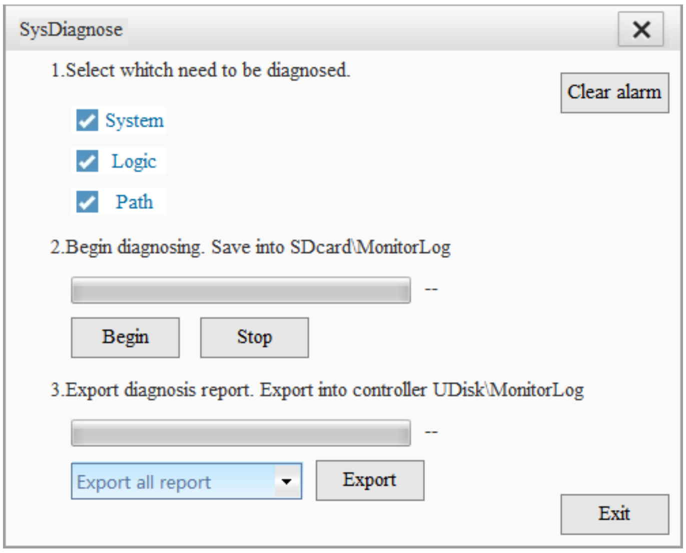
Logic: Contains the variable state during system operation.

Path: The trajectory of the robot motion.

2) Start diagnostics. The system saves files for the checked items. If you click **Stop** during diagnosis, the current save operation will stop and the internal file will still be the last saved file.

3) Export the diagnostic report. The process of exporting the report is indicated by the progress bar.

You can choose to export either the full diagnostic report or the report of the most recent diagnosis.



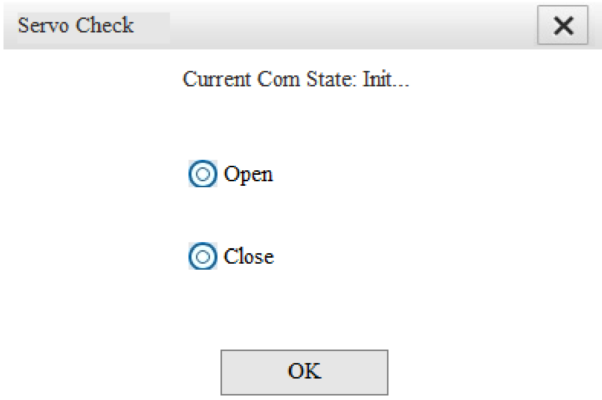
Note: When an exception occurs in the diagnosis that causes an alarm, click **Clear Alarm** to clear the exception before proceeding.

Servo Check:

During commissioning or use, it is necessary to change the servo parameters when replacing the controller, servo drive, servo software, or due to excessive load. At this point, the servo parameters may differ from the default parameters written on the production line. To ensure safety, it is necessary to confirm whether the current servo parameters are within a reasonable range.

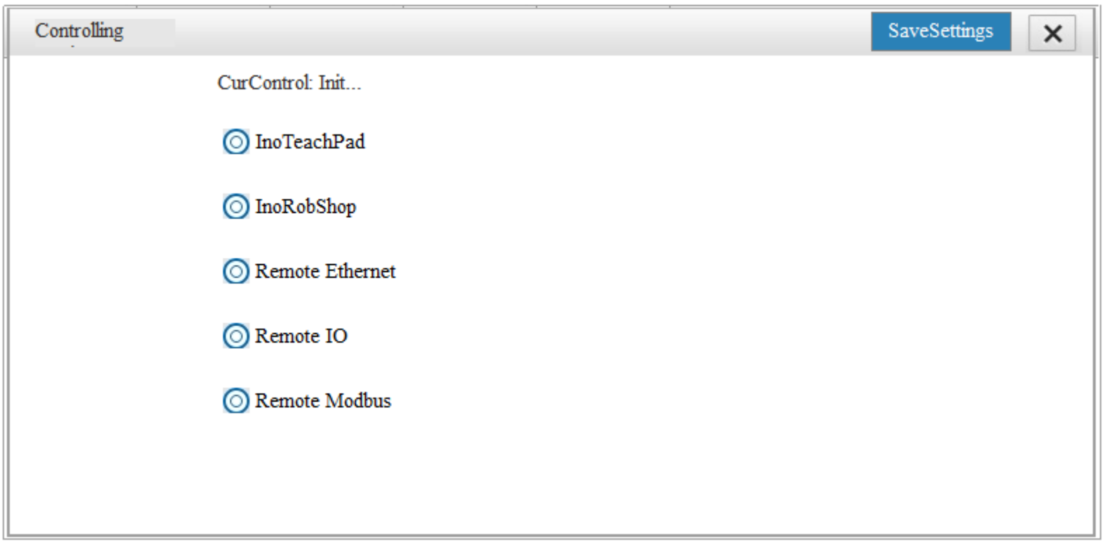
Servo parameter check includes power-on parameter check and operation parameter check. When parameter check fails, a permanent alarm will be generated. In this case, do not operate the robot and contact the manufacturer for help.

Do not turn off the servo parameter check function. If you indeed need to turn it off, turn off the function in factory mode and restart the robot.



Control Device:

Click the **Control Device** button, and an interface appears. You can select which device has the control of the robot.



When the control is not assigned to the teach pendant, a lock icon appears in the tool bar and you cannot control the robot (including modifying parameters, running programs, etc.) through the teach pendant. You can only use the teach pendant for monitoring purposes.

When the control is assigned to Remote I/O, you need to set the startup speed percentage (Start Vel) for the first running of the program. Typically the speed is less than 100% to ensure safety. Note: You can change the startup speed even when the control is already assigned to Remote I/O or Remote Modbus.

Brake Release:

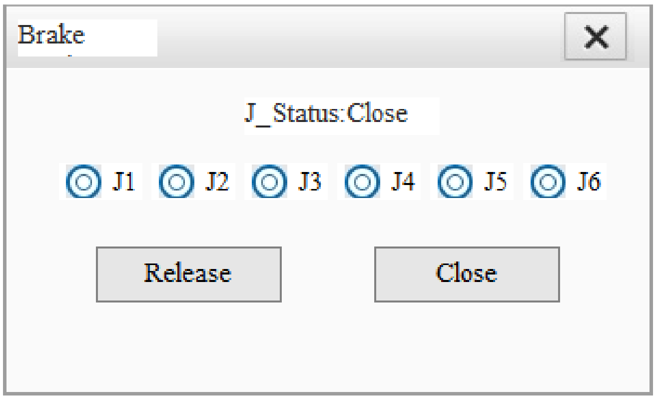
This function allows the unlocking and locking of one of the J1-J6 axes.

This feature is available only when the following conditions are met:

1) Control is assigned to the teach pendant
2) Manager user or Factory user
3) The robot is currently in an emergency stop

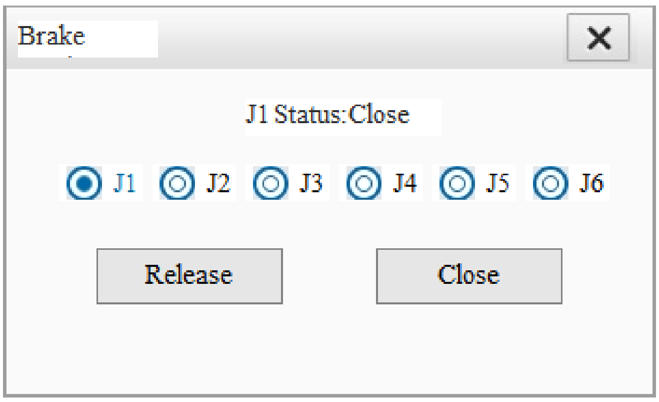
Operation steps:

1） Click the **Brake** button under **Other** tab.



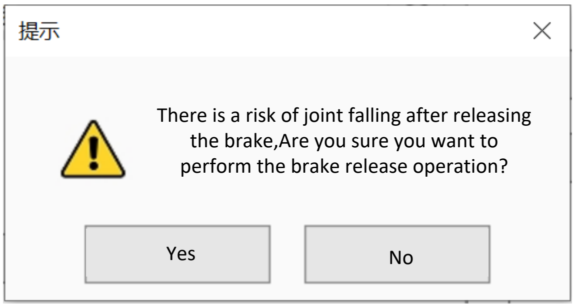
2） Select the axis number.

You can select one of the J1-J6 axes. For example, when you select J1, you can see the status of J1.

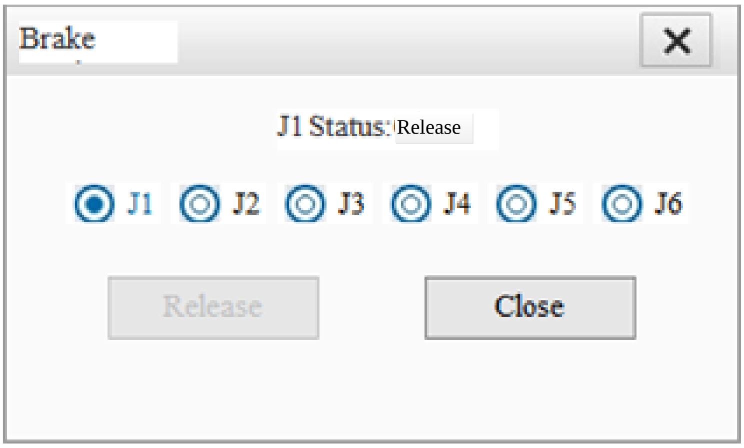


3） Release the brake.

a） Click the **Release** button.



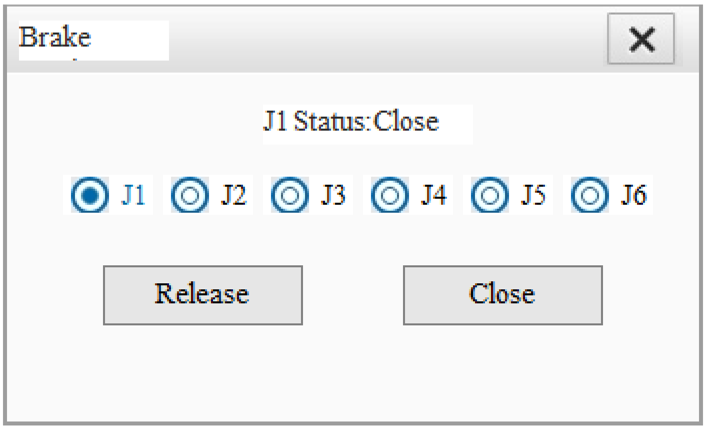
b） Click **Yes**.



c） The notification bar prompts the result.

4） Close the brake.

a) Click the **Close** button.



b) The notification bar prompts the result.

5） (Optional) Close the brake by exiting the setting.

a) When you close the setting interface by clicking the X sign, the brake of the currently selected axis will also be closed. The notification bar prompts the result.

Note:

1） On the handheld teach pendant, when the brake of a certain axis is released, the emergency stop button will be bounced. In this case, you can close the brake of the axis. Only when the emergency stop button has been pressed again can you click the button to release the brake.

2） When the axis number has been selected and the brake release is not applied, it is allowed to close the brake as this is a safe operation.

4.7 Extended Functions

a) Vision Calibration

This function determines the relationship between the vision coordinate system and the robot's coordinate system. See 6.4 Vision Calibration.

b) Tracking Process Setting

This function configures the tracking process for the linear conveyor and circular turntables. In conjunction with interaction with external cameras, dynamic grasping can be achieved. See 6.3 Tracking Process.

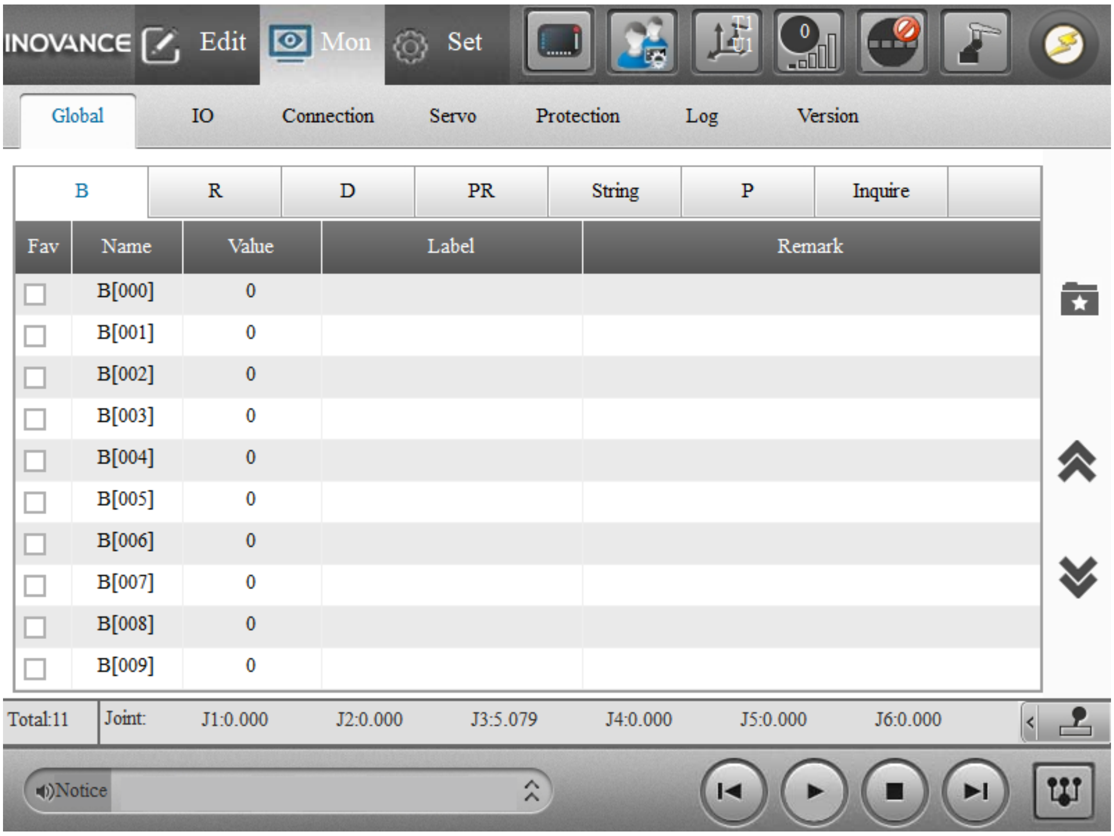
5 Monitoring

You can monitor information such as variables, I/Os, communication status, logs, etc.

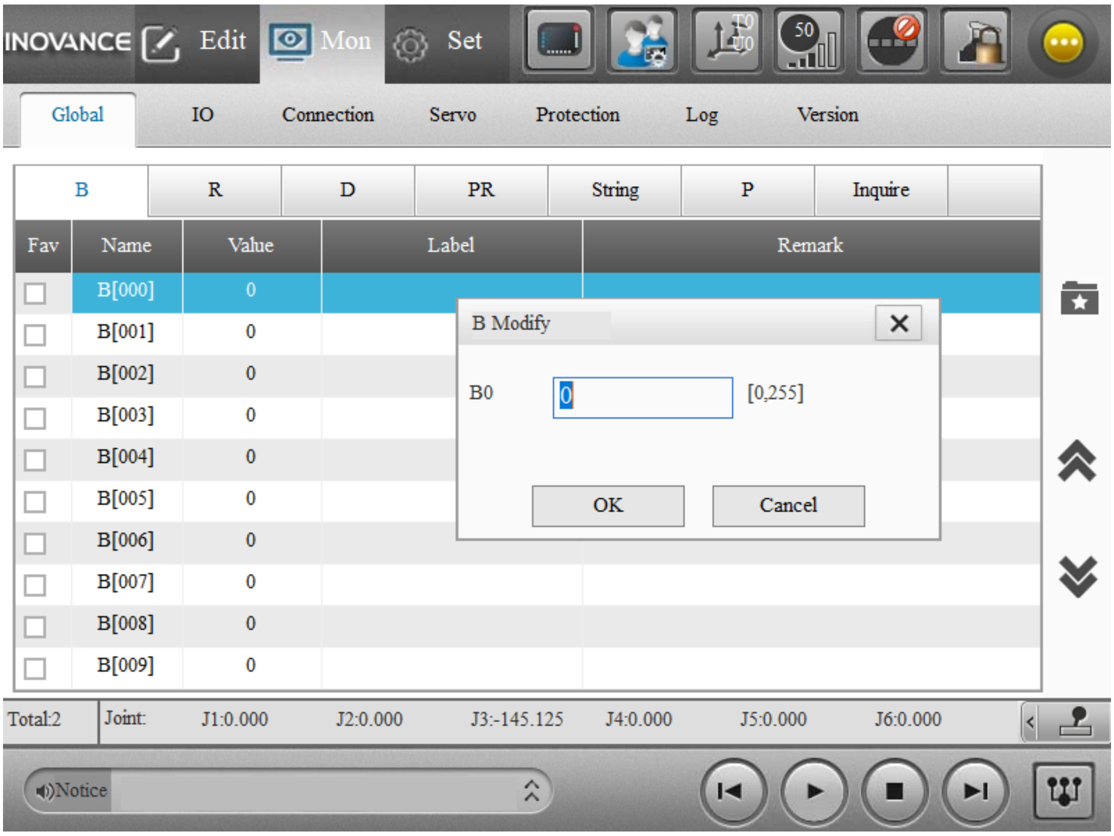
5.1 Basic Operations

The following describes some basic operations on the monitoring interface.

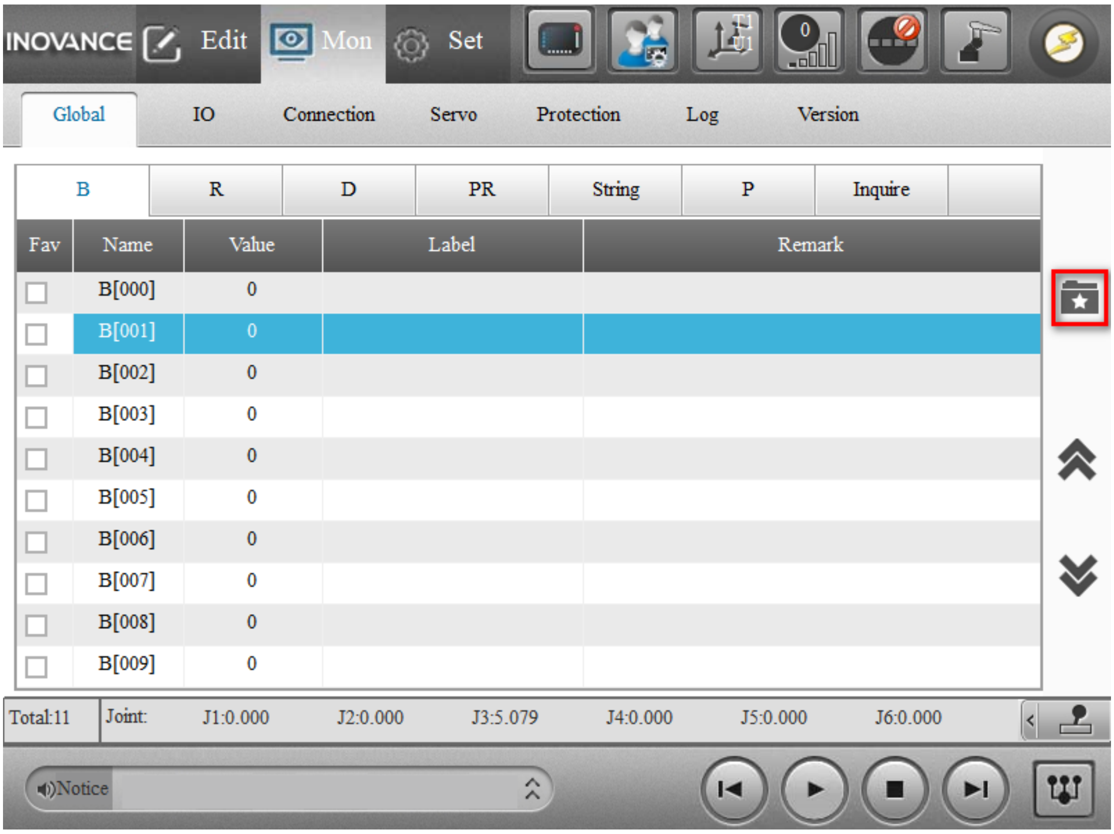
Paging: You can browse through the list using the  and  buttons.



Modify: Double-click an object to directly modify it.



Favorite: For global value variables and I/Os, you can check/uncheck the items to add them to or remove them from the favorites. You can click  on the right to switch between the display modes: display favorites, and display all.

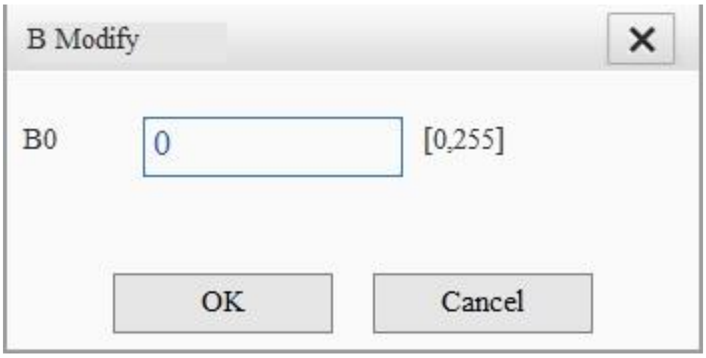


5.2 Global Variable Monitoring

The global variables monitoring interface includes seven tabs: B, R, D, PR, String, P, and Inquire.

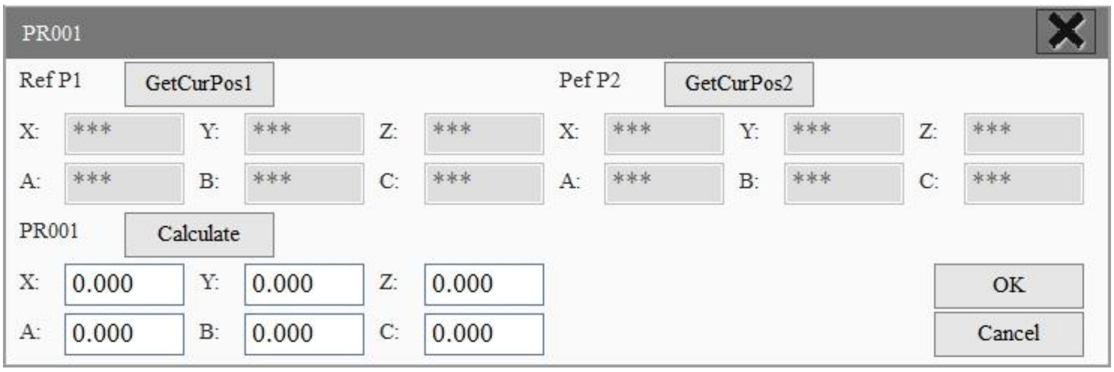
a) B/R/D Variables

Double-click a variable, and the following window where you can modify this variable value appears.



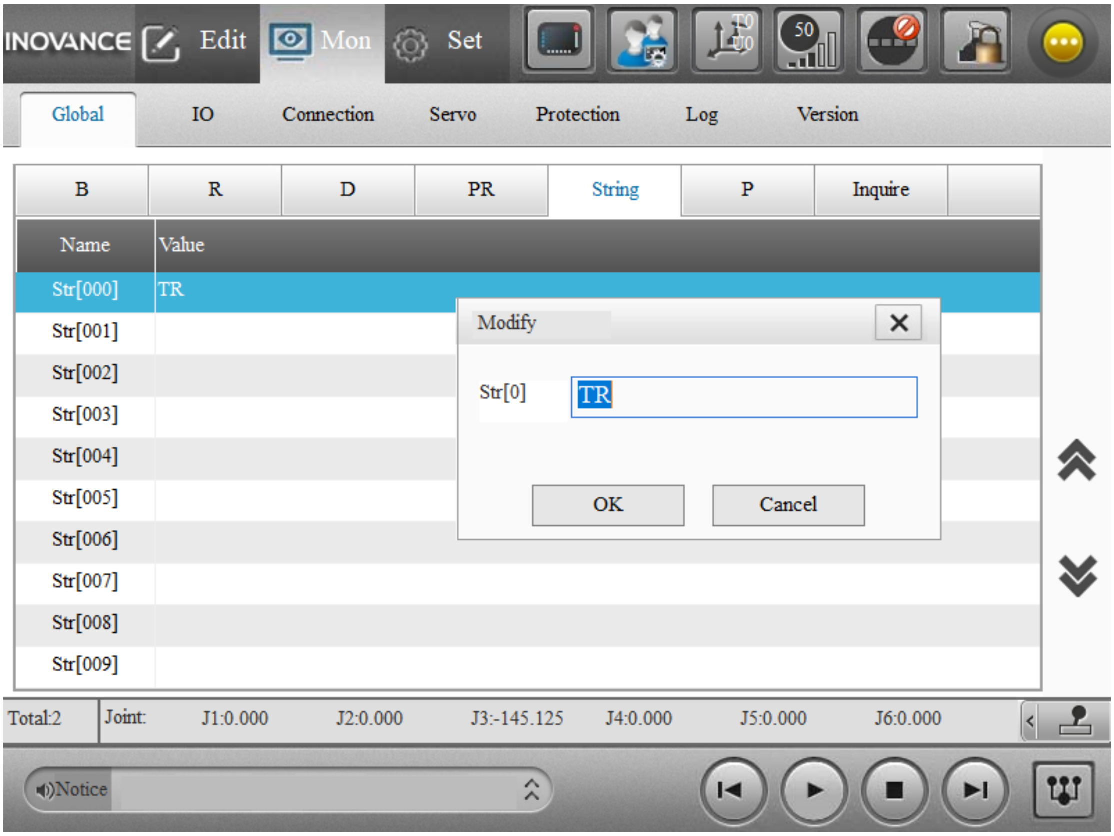
b) PR Variables

Double-click a PR variable, you can directly enter a value or calculate it by taking two points.



c) Global String Variables

Double-click a string variable and modify it.



d) Global Position Variables

The values of the monitored global position variables are the current values.

Note:

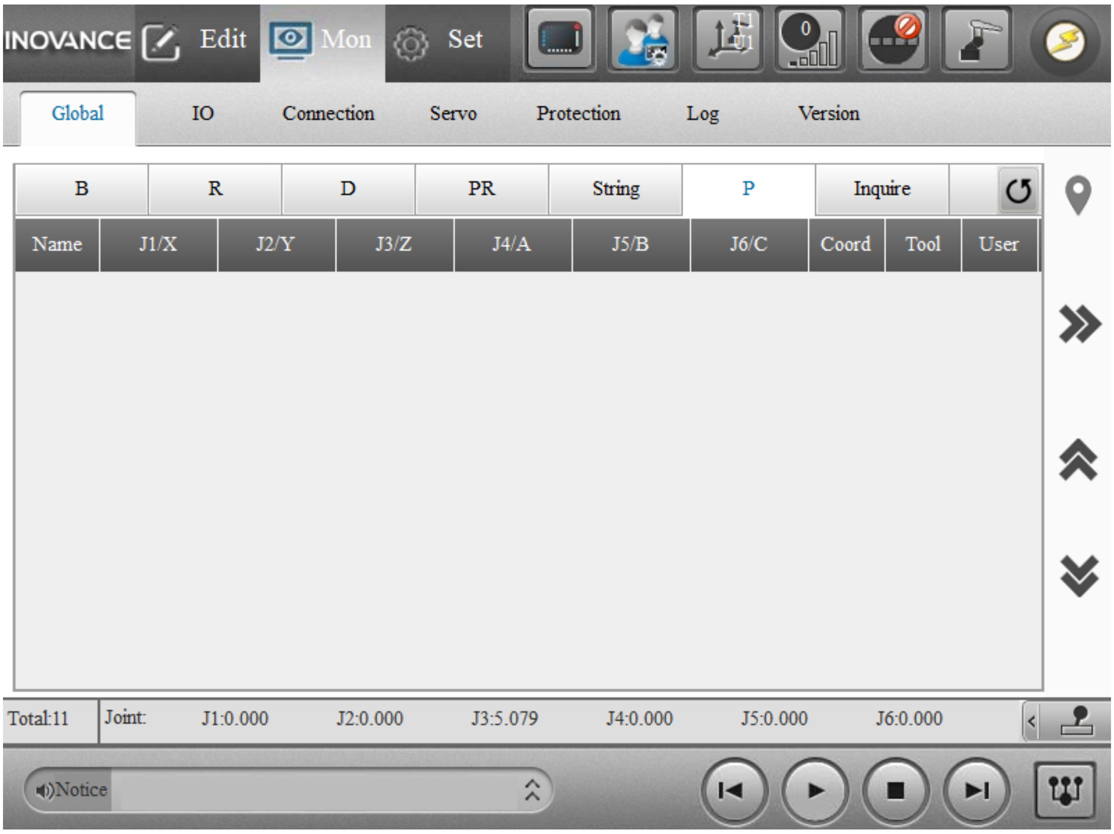
1. The P variables in the programming interface are initial points defined in the global point file, while the P variables in the monitoring interface indicate the current values. If an initial point is modified through the instruction p=, the value of the point will be updated in the monitoring interface.

2. The monitoring interface always displays all P variables from 0 to 9999. For unused P variables, they are also displayed, with the value being null.

3. The refresh mechanism of P variables:

The P variable on a certain page will be refreshed when you click the refresh button, or when you

switch to the P page, or when you turn the pages of P variables, or when you locate a P variable. The P variables are not refreshed in real time and can only be refreshed through these operations.
4. The P variables cannot be modified in the monitoring interface. If you want to modify the initial values defined in the point file, make the modification on the programming interface; if you want to modify the memory value without modifying the initial values defined in the point file, do the modification on the quick monitoring panel of the debug interface.

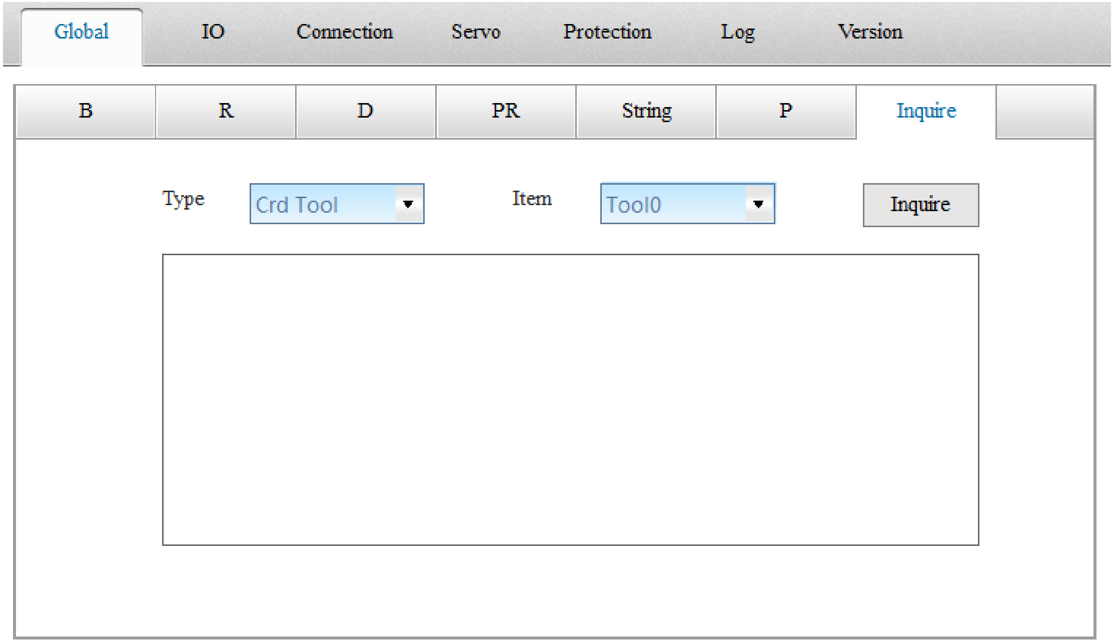


Click  on the right to switch between displaying the label and remarks of point and displaying the coordinates of the point.

Click  on the right to locate a specific point quickly.

e) Global Variable Query

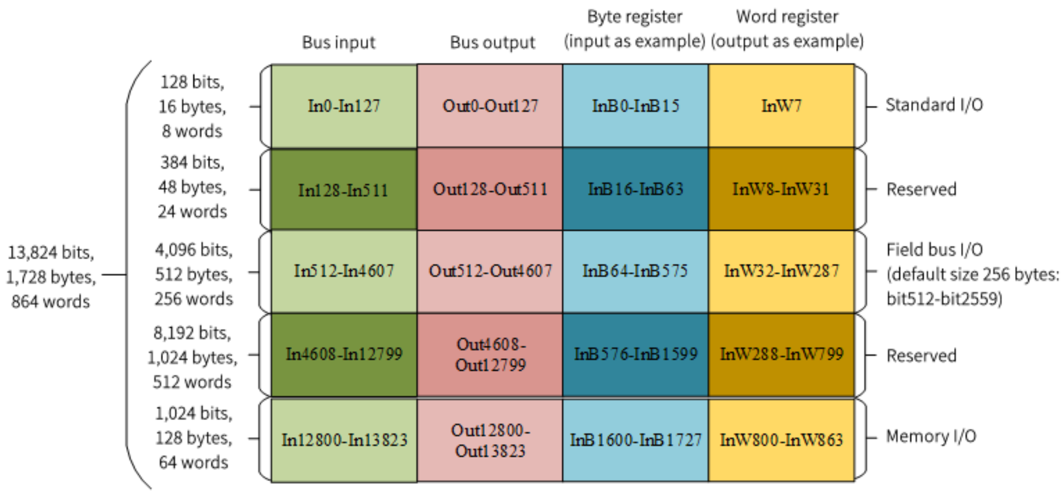
For tool coordinate system, tool load, user coordinate system, and grip load, memory values can be queried. After modifying the value using instructions, you can query the results here.
Select **Type** and **Item** and click **Inquire**.
Note: Switching interfaces will not cause a refresh.



5.3 I/O Monitoring

5.3.1 Introduction of Robot Bus Address

The following introduces the concept of robot bus address. The bus address of the robot is divided into standard I/O, fieldbus I/O, and memory I/O, as shown in the following figure.



Standard I/O: Equivalent to digital I/O, that is, the I/O whose switching state is directly associated with a high or low physical level.
Fieldbus I/O: The current fieldbus I/O of robots refers to the I/O used by the robot as a fieldbus slave for data exchange with the master.
Memory I/O: Equivalent to Bool-type variables inside the robot, used for internal operations.

5.3.2 How to Use I/O Monitoring

Precondition: Ensure that IRLink configurations are correct before using I/O monitoring!
The I/O monitoring interface includes five sub-interfaces: IN, OUT, AD, DA, SysIO.

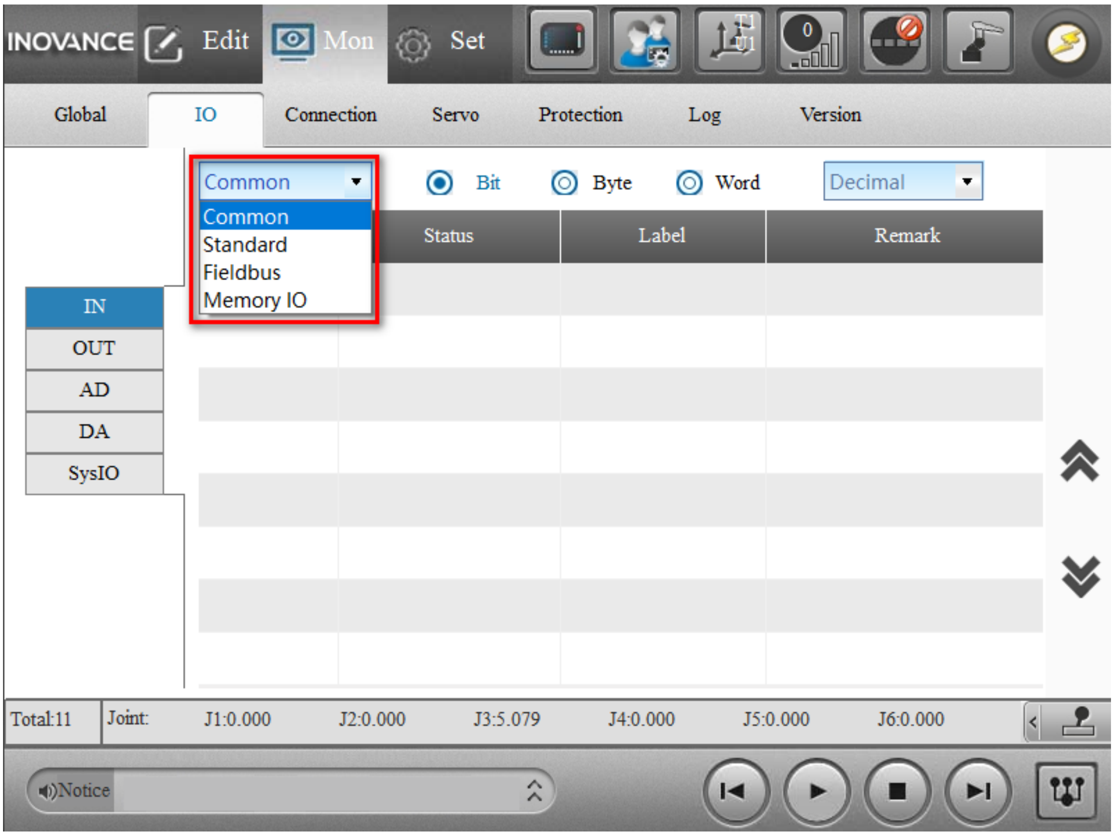
a) **IN**

1. Input (IN) I/Os includes common I/O, standard I/O, fieldbus I/O, and memory I/O, which can be selected from the drop-down list, as shown in the following figure.

Description:

Common I/O: Refers to I/Os whose Label or Remark is defined.

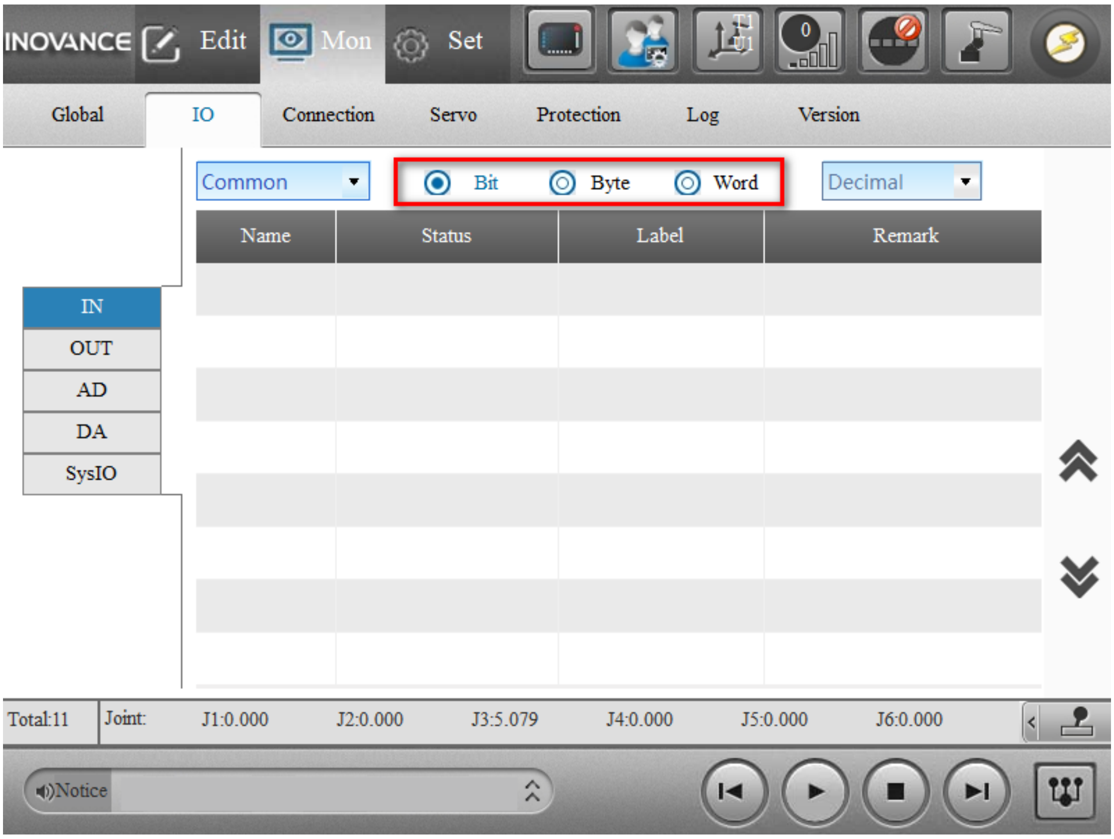
For the definition of standard I/O, fieldbus I/O, and memory I/O, see Section 5.3.1.



2. Each type of I/O can be displayed by bit, by byte or by word.

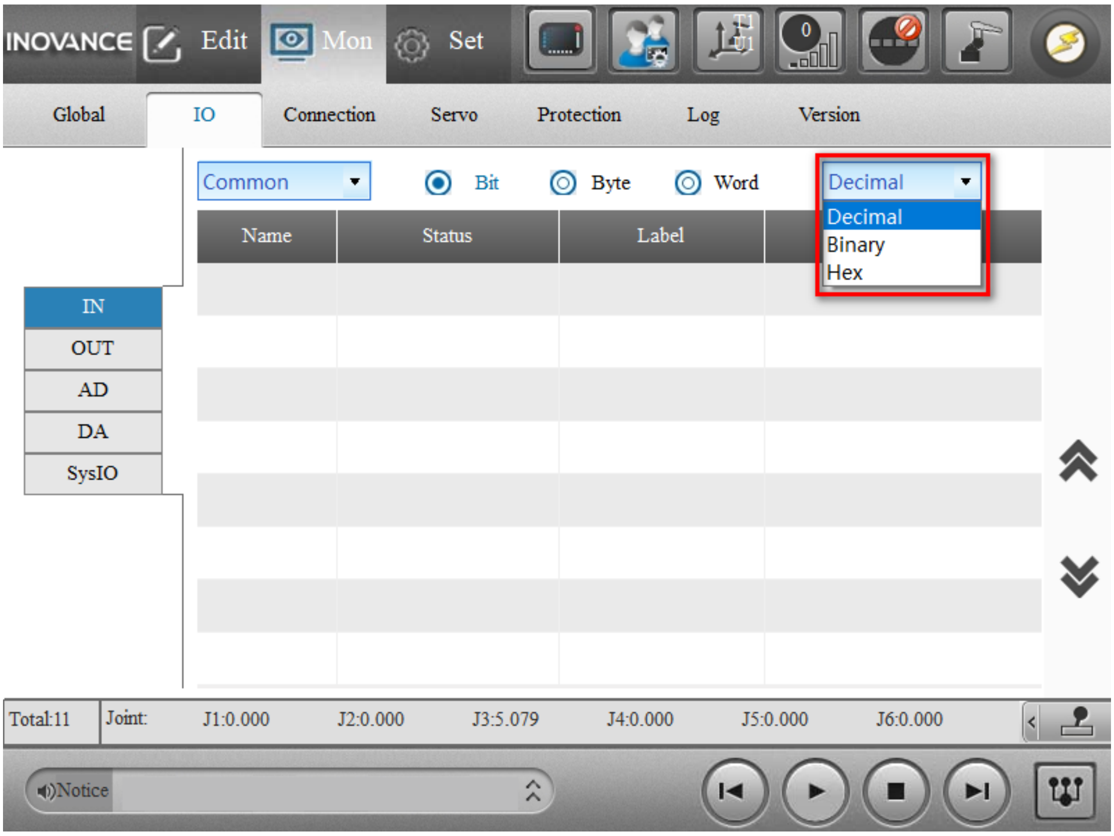
Description:

8 bits equals 1 byte, and 16 bits equal 1 word (Example: The value of I/O consisting of bits indexed from 0 to 7 corresponds to the value of InB[0]; the value of I/O consisting of bits indexed from 0 to 15 corresponds to the value of InW.)

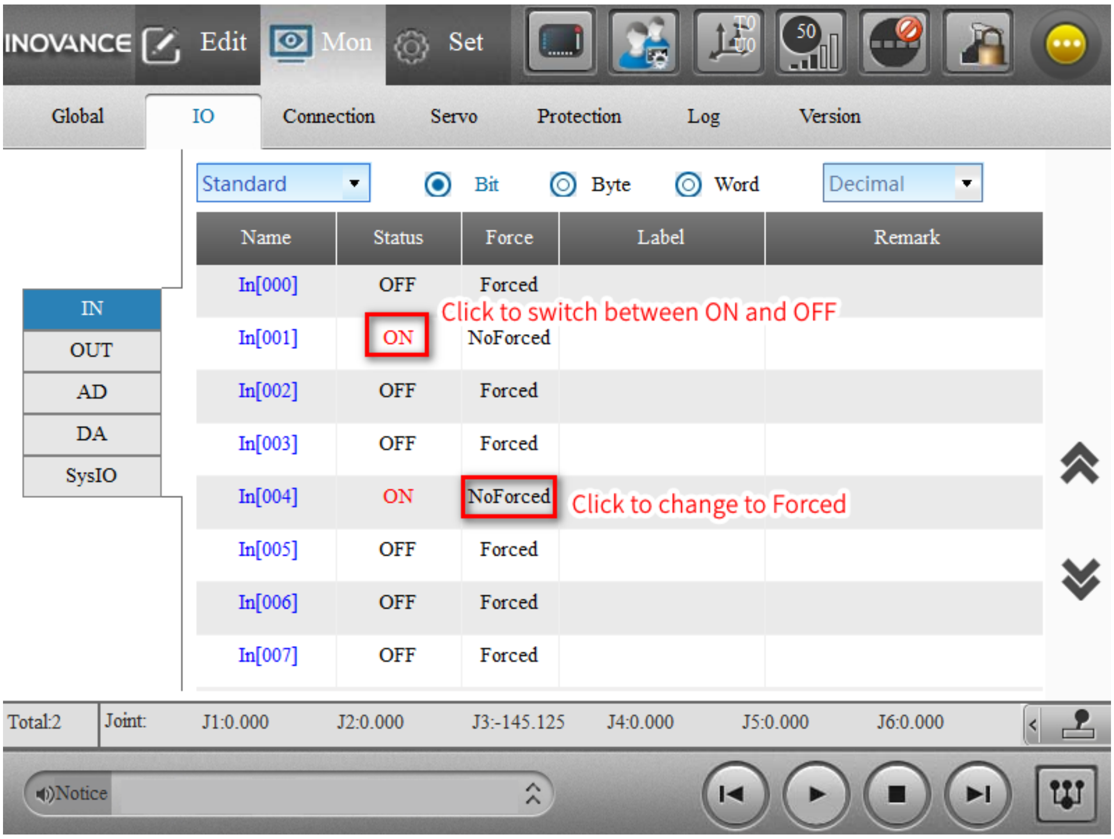


3. When you choose to display I/O by bit, the status may be ON or OFF depending on the return value, 1 for ON, 0 for OFF, ON in red, OFF in black.

4. I/O displayed by byte and word can further be displayed in decimal, binary, or hexadecimal. Decimal by default.



5. When you choose to display I/O by bit, the "Force" column is displayed. When the value of "Force" column of an I/O is "Forced", you can click the column to change its value.



6. Description of columns

Force: Input signal state is determined by an external source by default. However, IN can be changed to "Forced" state using the forced switch. In this case, signals can be manually forced to be ON or OFF.

Status: Displays the current status. In particular, for the standard I/O, you can click the signal value to reverse the state.

Label: The label string of the I/O corresponding to the I/O variable name.

Remark: The remark string of the I/O corresponding to the I/O variable name.

Note:

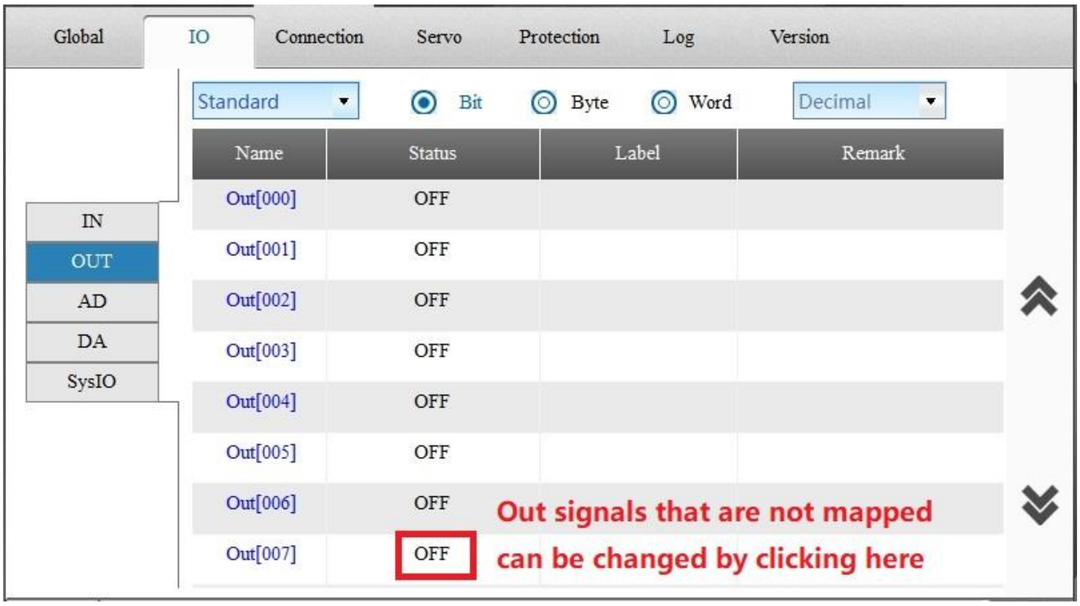
You can monitor the IN variables in Monitor in the debug mode (for details, see the detailed description of the Monitor function in Section 3.4.)

Enter debug mode, click **Monitor**, check the **Pick** option; or select **Custom base type**, enter the variable name, click **Add** and then the variable is automatically displayed in the list of monitored objects.

b) OUT

The operation on the OUT variables is generally similar to that on the IN variables.

Note: When the I/O is orange, you do not have control to change the ON/OFF status by clicking it.



Like input I/O, 8 bits equal to 1 byte, and the 16 bits equal to 1 word.

Note:

You can monitor the OUT variables in Monitor in the debug mode (for details, see the detailed description of the Monitor function in Section 3.4.)

Enter debug mode, click **Monitor**, check the **Pick** option; or select **Custom base type**, enter the variable name, click **Add** and then the variable is automatically displayed in the list of monitored objects.

c) AD/DA

Type: Indicates that an analog signal is current or voltage.

Range: Analog signal range. Every analog port has several ranges for selection depending on the IRLink product model.

Status: An analog parameter value. When it is a voltage signal, the unit is V; When it is a current signal, it is measured in mA. Click the status to directly modify the status value.

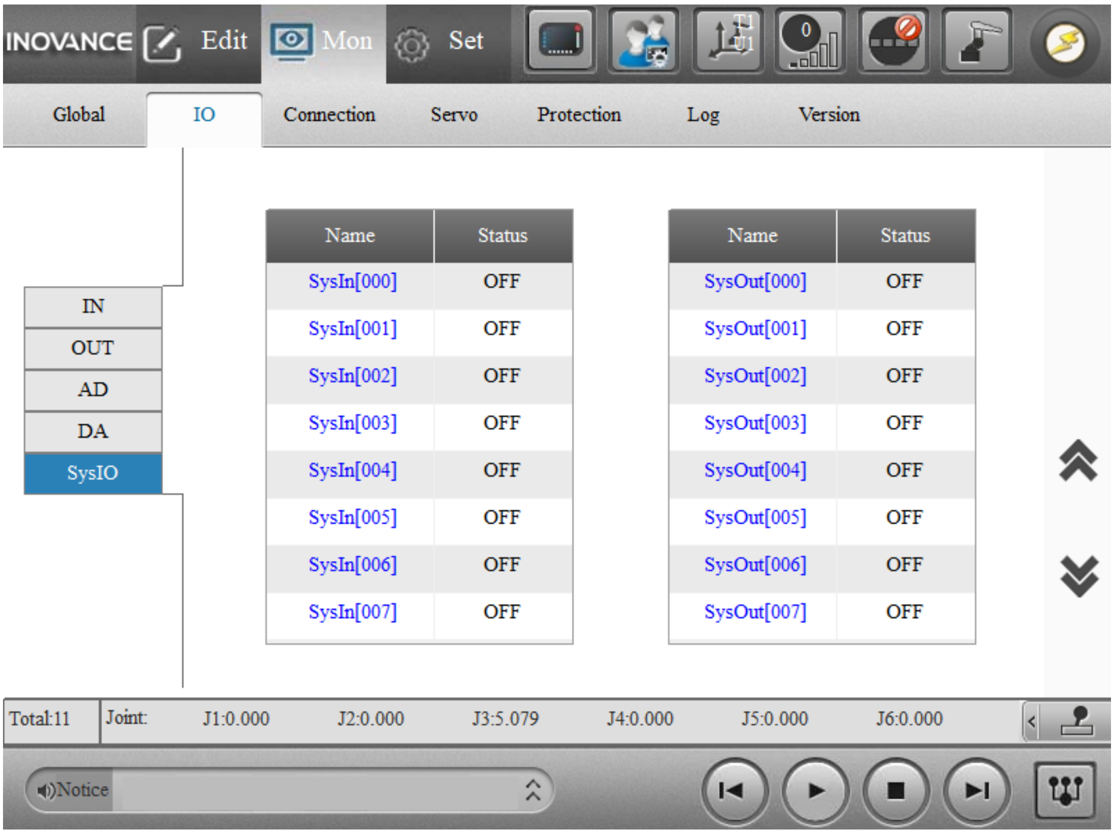
Switch: Determines whether a DA status value is valid. You can change the value by clicking the switch.

Label: Displays the label set in the project

Remark: Displays the remarks set in the project

Note: Labels and remarks cannot be modified here, please modify them in the project.

d) System I/Os



For the IRCB10 controller, System I/Os are In[0] to In[2] that are linked to emergency stop, enable function and mode switching respectively and displayed on the IN/OUT interface.

For the IRCB300, IRCB100 and IRCB500 controllers, System I/Os have a separate SysIO interface with 16 inputs and 16 outputs as follows:

IRCB300, IRCB100 and IRCB500 controllers			
Input	Function	Output	Function
SysIn[0]	Emergency stop	SysOut[0]	System running indication
SysIn[1]	Enable	SysOut [1]	System error indicator
SysIn[2]	Mode switching (Teach/Play)	SysOut [2]	System enable indication
SysIn[3]	-	SysOut [3]	EtherNet1 connection indication On: Connected; Off: Disconnected
SysIn[4]	Safety door	SysOut [4]	EtherNet1 frame transmission indication Flash: Data is being transferred; Normally on: Connected, but no data is being transferred
SysIn[5]	Safety door	SysOut [5]	EtherNet2 connection indication On: Connected; Off: Disconnected
SysIn[6]	ITP100 enable	SysOut [6]	EtherNet2 frame transmission indication Flash: Data is being transferred;

			Normally on: Connected, but no data is being transferred
SysIn[7]	-	SysOut [7]	-
SysIn[8]	-	SysOut [8]	Robot enable indicator
SysIn[9]	-	SysOut [9]	-
SysIn[10]	-	SysOut [10]	-
SysIn[11]	-	SysOut [11]	-
SysIn[12]	-	SysOut [12]	-
SysIn[13]	-	SysOut [13]	-
SysIn[14]	-	SysOut [14]	-
SysIn[15]	-	SysOut [15]	-

System I/Os of the IRCB500 controller			
Input	Function	Output	Function
SysIn[0]	Emergency stop	SysOut[0]	Soft STO
SysIn[1]	Enable	SysOut [1]	Robot enable indicator
SysIn[2]	Mode switching (Teach/Play)	SysOut [2]	
SysIn[3]	Start confirmation signal	SysOut [3]	EtherNet1 connection indication On: Connected; Off: Disconnected
SysIn[4]	Safety door	SysOut [4]	EtherNet1 frame transmission indication Flash: Data is being transferred; Normally on: Connected, but no data is being transferred
SysIn[5]	Safety door	SysOut [5]	EtherNet2 connection indication On: Connected; Off: Disconnected
SysIn[6]	ITP100 enable	SysOut [6]	EtherNet2 frame transmission indication Flash: Data is being transferred; Normally on: Connected, but no data is being transferred
SysIn[7]	-	SysOut [7]	-
SysIn[8]	-	SysOut [8]	
SysIn[9]	-	SysOut [9]	-
SysIn[10]	-	SysOut [10]	-
SysIn[11]	-	SysOut [11]	-
SysIn[12]	-	SysOut [12]	-
SysIn[13]	-	SysOut [13]	-
SysIn[14]	-	SysOut [14]	-
SysIn[15]	-	SysOut [15]	-

e) **Description of Colors of I/O Monitor**

In the I/O monitor, the variable names are differentiated by colors: blue, orange and black, respectively.

Blue: Indicates standard I/Os connected to the actual device or bus I/Os assigned with an address.

Orange: Indicates the occupied output I/Os with control permissions other than RC_ ACTIVE and whose status cannot be changed through monitoring interfaces or instructions.

Black: Indicates I/Os that meet the following conditions:

1、 Standard I/Os not connected to the actual device or bus I/Os not assigned with an address.

2. Occupied output I/Os with control permissions being RC_ ACTIVE.

Description:

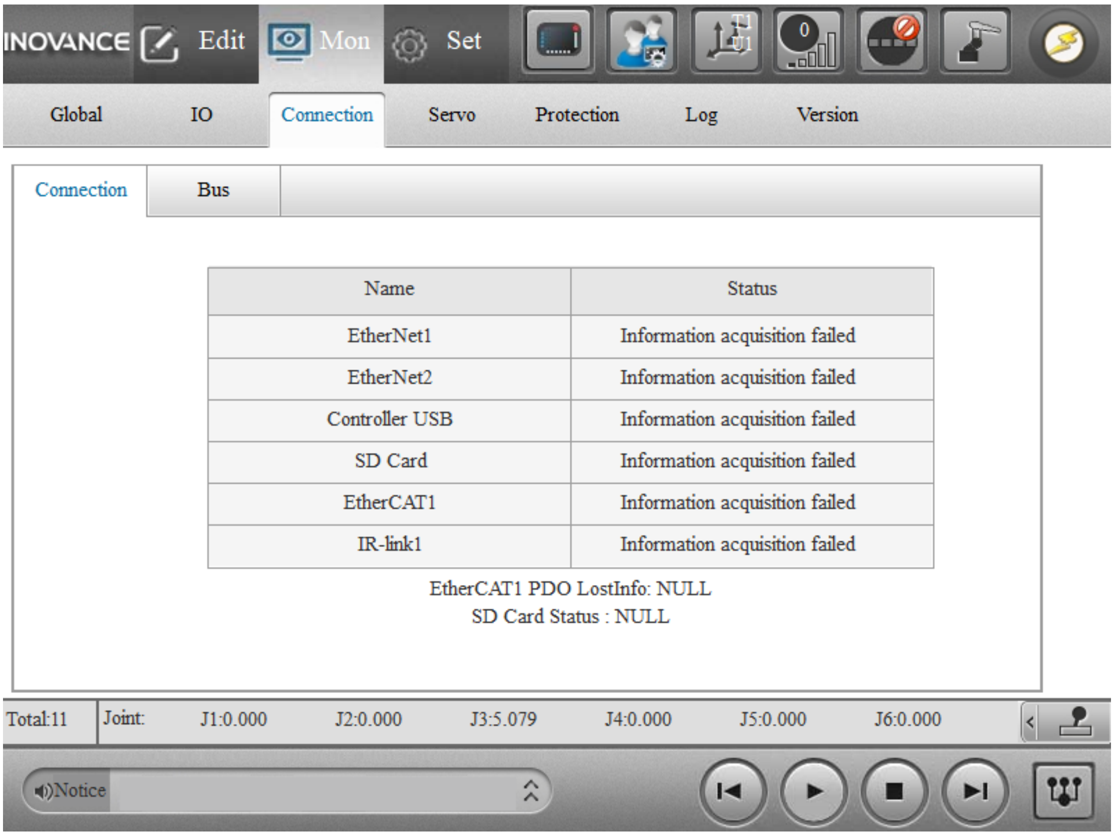
1. The priority of orange display is higher than that of blue and black. When an I/O is displayed in orange, if you want to know whether the I/O has blue or black attributes, you can infer based on the color of the I/Os before and after it.

2. Control authority of output signals:

Control permissions	Description
RC_STATIC	It indicates occupation by the RC system. That is, Out is bound to system functions under **External** > **I/O-Config**. In this case, output port signals are related to functions only and no signal state can be manually changed.
RC_ACTIVE	It indicates normal RC control state. In this case, signals are normally controlled. For example, signals can be changed in I/O monitor or using the Set command.
PLC_ACTIVE	It indicates PLC control state. In this case, signals are controlled by PLC software like InoRobShop.

5.4 Communication state

5.4.1 Device Connection



This interface allows you to view the status of the system hardware, as described in the following table.

Name	Status	Name	Status
EtherNet1	Network cable not connected	EtherNet2	Network cable not connected
	Dynamic IP: XX.XX.XX.XX or Static IP: XX.XX.XX.XX.XX		Static IP: 192.168.23.25
	Disabled		Disabled
	Undefined		Undefined
	Failed to get information		Failed to get information
Controller USB*	Device not connected	Memory card	Device not connected
	Connected and mounted successfully		Connected and mounted successfully
	Connected but failed to mount		Connected but failed to mount
	Undefined		Undefined
	Failed to get information		Failed to get information
EtherCAT1	Communication normal	IR-link1	Communication normal
	Slave disconnected		Slave disconnected
	Network cable not connected		Network cable not connected
	A non-EtherCAT device is connected		A non-IR-link device is connected
	Disabled		Disabled

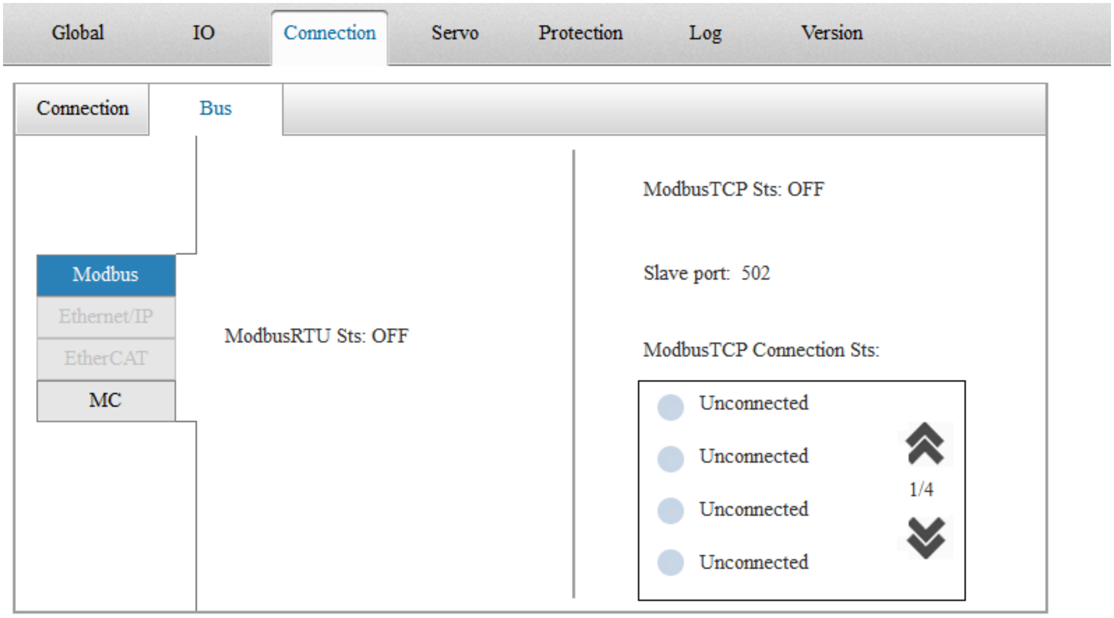
	Undefined		Undefined
	Failed to get information		Failed to get information

*Controller USB: The USB drive mounted to the controller. The controller USB must be in FAT32, EXT2 or EXT 3 format to be loaded successfully.

5.4.2 Bus Monitoring

Go to **Monitor** > **Connection** > **Bus**, three types of buses are displayed: ModbusTCP, Ethernet/IP and EtherCAT.

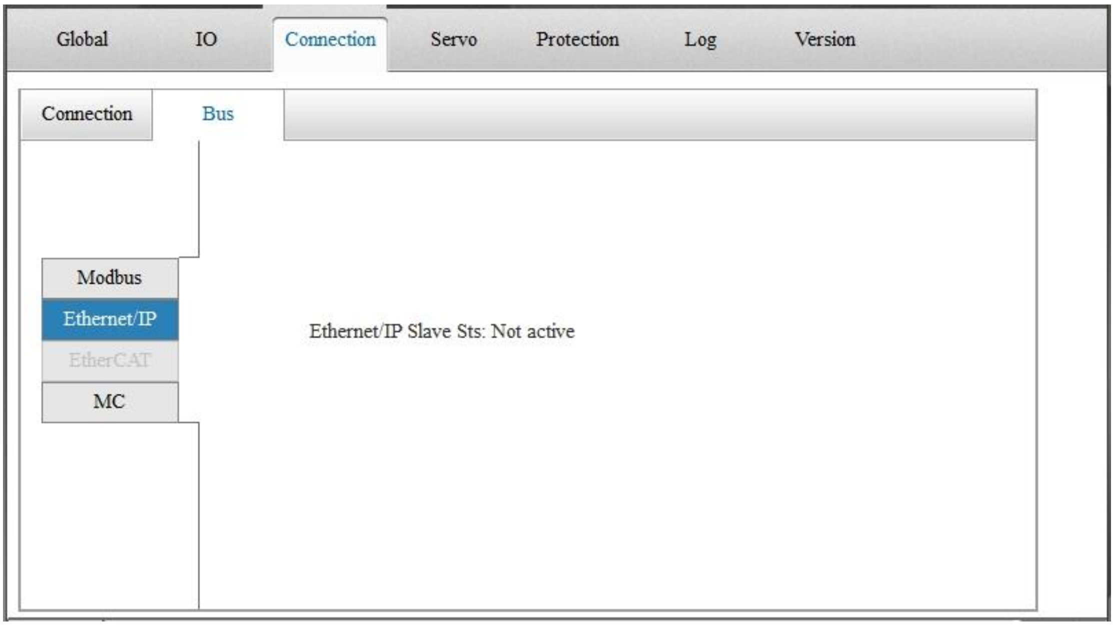
Modbus



This page shows the connection status of Modbus. When this page is open, the activation status and connection status are refreshed in real time.

1. The left shows whether ModbusRTU is activated, the right shows whether ModbusTCP is activated, and the slave port (default 502).
2. ModbusTCP connects up to 16 masters and the IP address and port of the master are displayed.
3. ModbusTCP alarm mechanism is as follows: No alarm if the disconnection is made by the master actively. If the network cable is removed, only "Eth1 physical network link down" or "Eth2 physical network link down” alarm is reported.

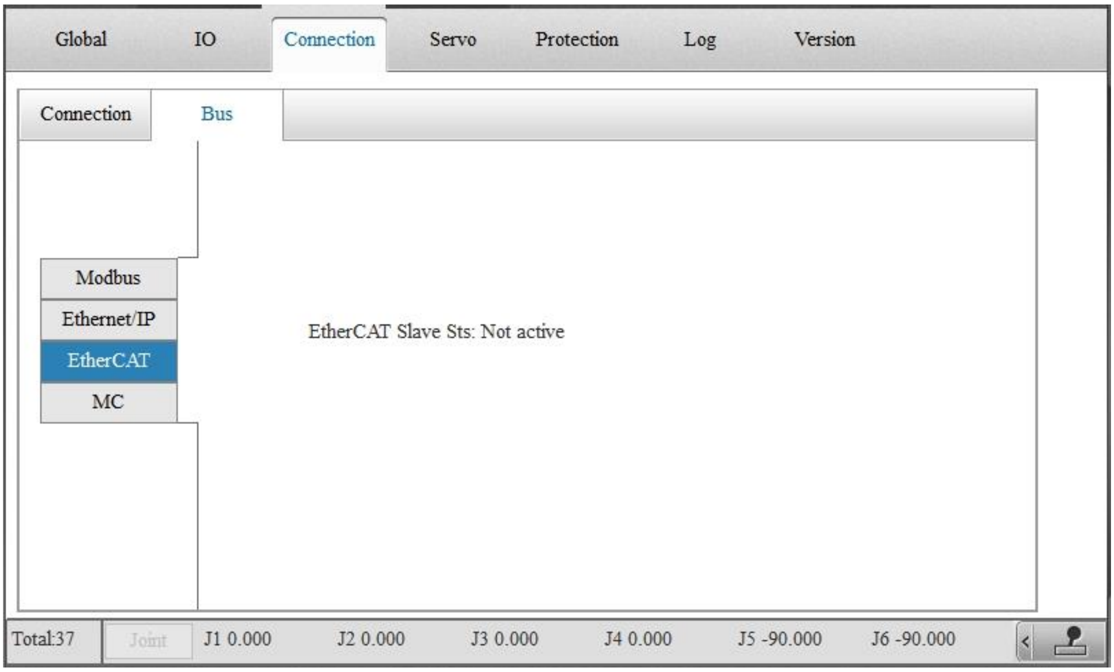
Ethernet/IP



This page displays the EIP connection status. When this page is open, the activation status and connection status are refreshed in real time.

1. It displays the activation status, the slave port number, and the connection status of the EIP slave. Up to one EIP master can be connected, and the IP address and port number of the master can be displayed when connected.

EtherCAT

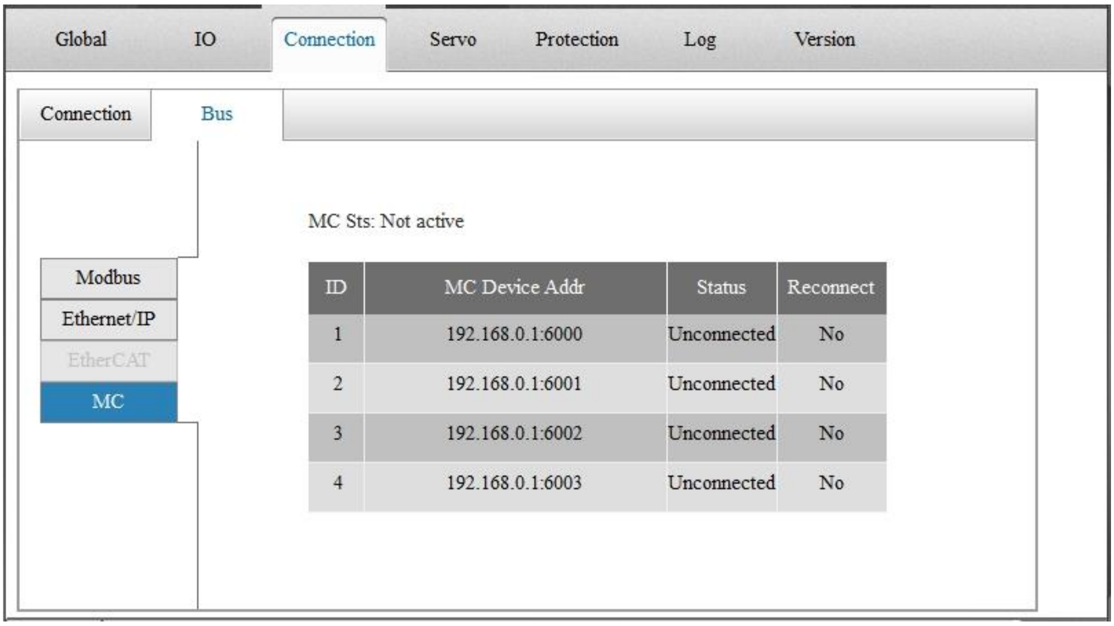


This page shows the connection status of EtherCAT. When this page is open, the activation status and connection status are refreshed in real time.

1. Displays whether EtherCAT is active.
2. Displays whether EtherCAT master is connected.

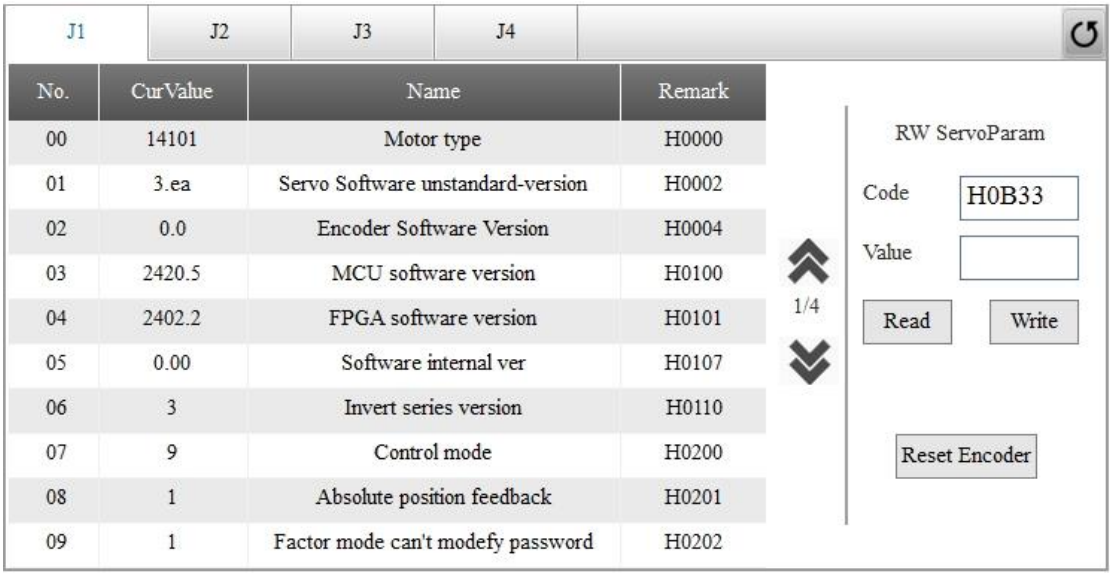
3. MC

4. You can monitor the connection status of the master when the robot acts as a MC slave. The monitoring list displays the configurations of 4 default masters when the masters are not connected. When the masters are connected, the connection status is green. The MC feature supports the connection of 4 masters.



5.5 Servo State

In the Servo status panel, the servo parameters on the left are refreshed as you switch between axes. On the right, you can read and write some servo parameters.



About read and write of servo parameters:

Servo parameters are currently available as follows:

Parameter	Type	Function
H0b33	Read and write	Enter the index of servo fault record to be queried
H0b34	Read only	Fault code corresponding to the index written by H0b33
H0b51	Read only	Sub-fault code corresponding to the index written by H0b33
H0b45	Read only	Current sub-fault code of servo

The detailed operations are as follows:

1、 When a servo alarm occurs, write H0B33 with the index of servo fault record to be queried.

2、 Use the following parameters to read the fault code for the corresponding index of servo fault

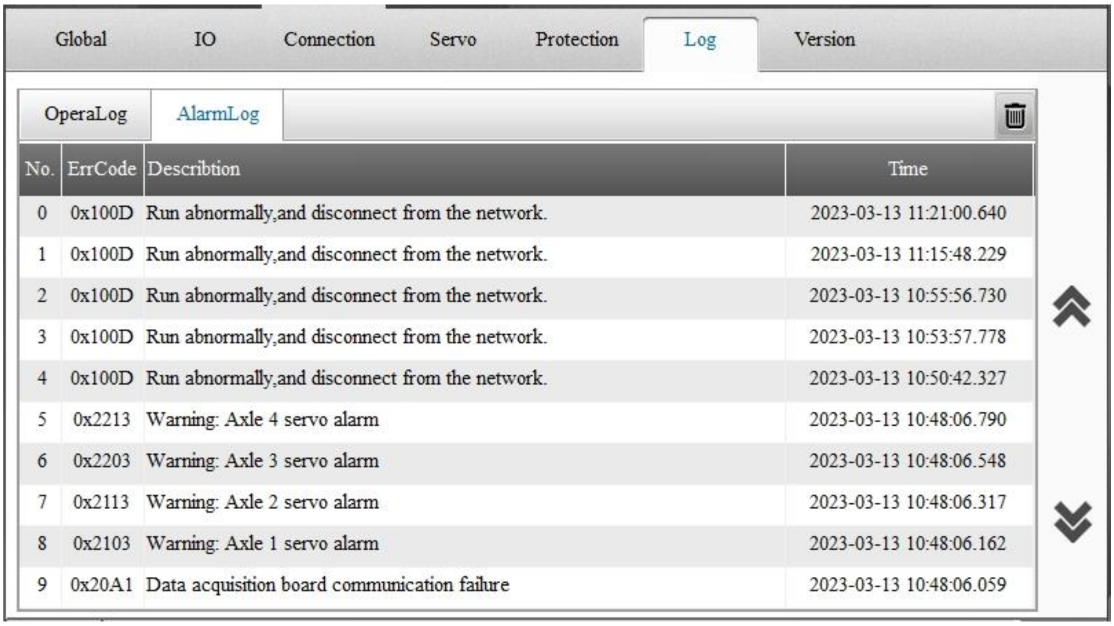
H0b34 Fault code corresponding to the index written by H0b33

H0b51 Sub-fault code corresponding to the index written by H0b33

H0b45 Current sub-fault code of servo

5.6 Log

The log panel in the monitoring interface allows you to view the operation logs and alarm logs, as shown in the following figure. The operation logs record critical actions the user made on the teach pendant, and the alarm logs record the errors occurred in the controller. The operation logs and alarm logs can provide information to support troubleshooting, see Appendix 1 Troubleshooting of Robot Alarms.



No.	ErrCode	Describtion	Time
0	0x100D	Run abnormally,and disconnect from the network.	2023-03-13 11:21:00.640
1	0x100D	Run abnormally,and disconnect from the network.	2023-03-13 11:15:48.229
2	0x100D	Run abnormally,and disconnect from the network.	2023-03-13 10:55:56.730
3	0x100D	Run abnormally,and disconnect from the network.	2023-03-13 10:53:57.778
4	0x100D	Run abnormally,and disconnect from the network.	2023-03-13 10:50:42.327
5	0x2213	Warning: Axle 4 servo alarm	2023-03-13 10:48:06.790
6	0x2203	Warning: Axle 3 servo alarm	2023-03-13 10:48:06.548
7	0x2113	Warning: Axle 2 servo alarm	2023-03-13 10:48:06.317
8	0x2103	Warning: Axle 1 servo alarm	2023-03-13 10:48:06.162
9	0x20A1	Data acquisition board communication failure	2023-03-13 10:48:06.059

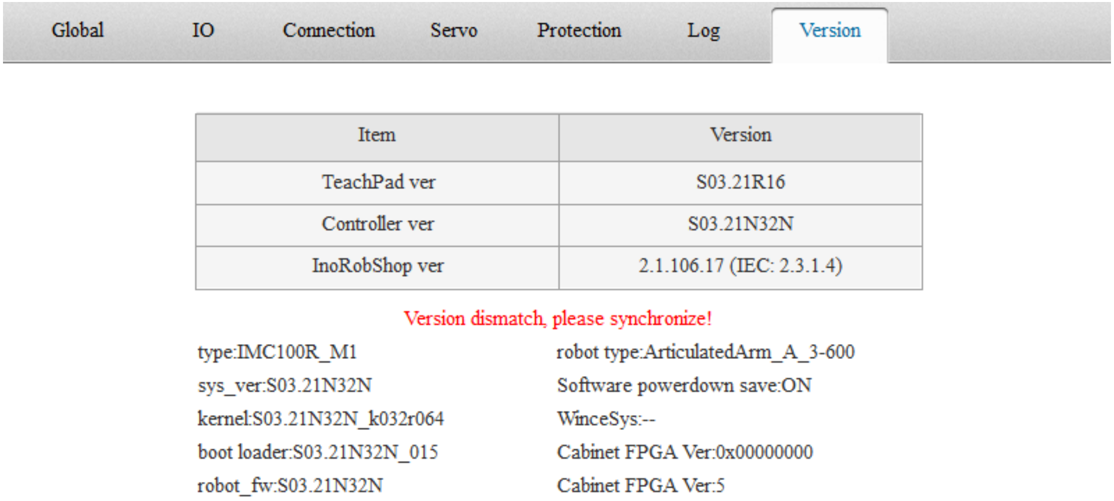
Note:

1. You can click  to clear the operation logs or alarm logs.

2. The operation log and the alarm log can each contain up to 1000 entries. Only the latest 1000 entries are recorded.

5.7 Version

You can view version information as shown in the following figure.



You can see more information in factory mode.

Note:

1. The teach pendant automatically jumps to this interface upon power on when the teach pendant and controller are inconsistent in version.

2. For the handheld teach pendant, when the version mismatch occurs, the interface will display a Sync button so that you can upgrade the teach pendant to the same version as the controller. See 7.7 Synchronizing Teach Pendant.

5.8 Current Protection

See 7.10 Current Protection.

6 Process Application

6.1 Tracking Process

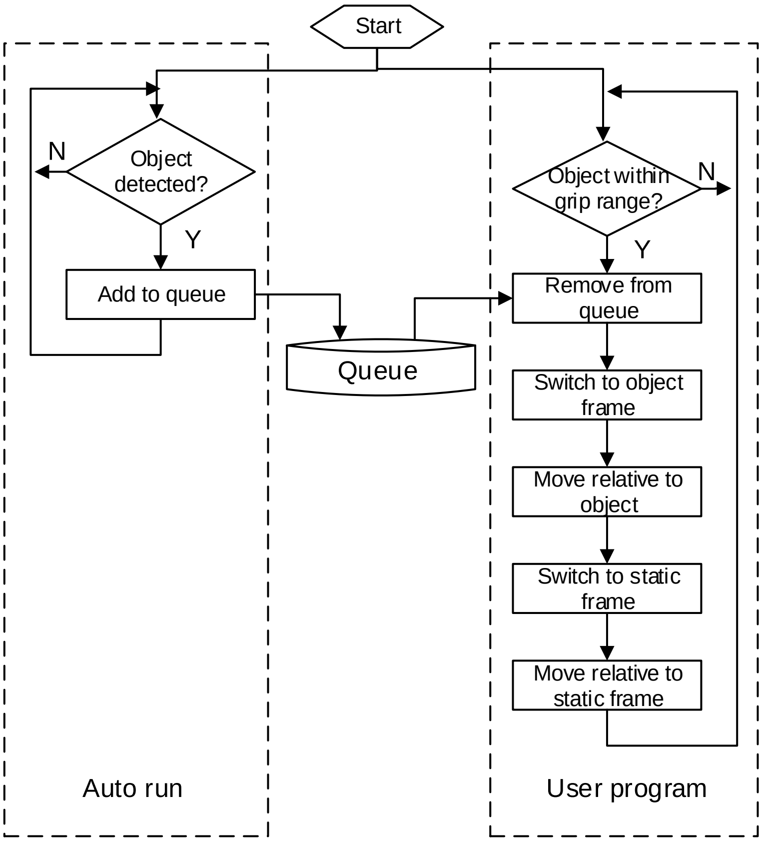
6.1.1 Overview

Tracking process is an application for tracking moving objects and planning robot movements with reference to the moving objects. Conveyor tracking is a typical application in the tracking process. Conveyor tracking is the process by which objects on the conveyor are detected by visual or photoelectric sensors so that the robot can accurately grasp the moving objects.

1. Workflow

1) Detection is the process of obtaining the position of an object on a conveyor through visual or photoelectric sensors, placing the obtained object position information into the conveyor object queue, and tracking the changes in the object position through encoder feedback.

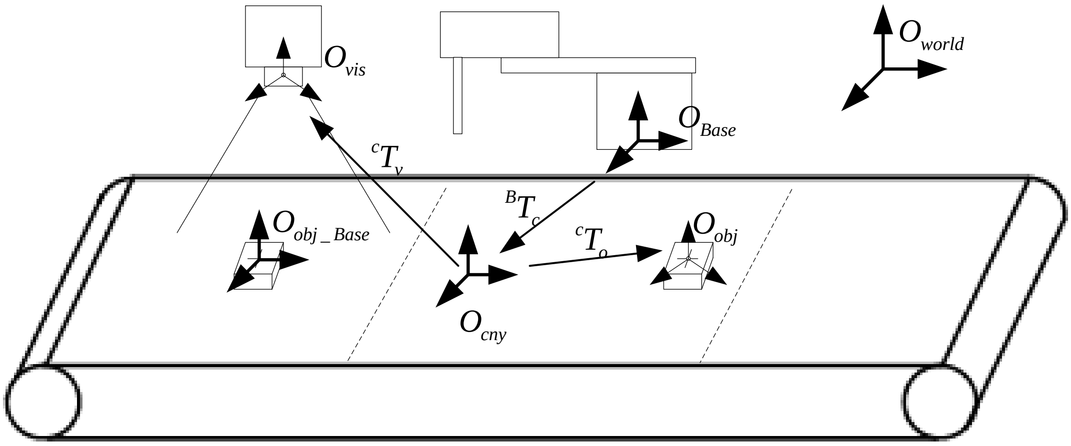
2) Tracking motion is a robot motion that takes the moving object in the queue as the reference coordinate system. The detection is processed in parallel with the tracking motion, as shown in the following figure.



Conveyor tracking process flow

2. Position description system

The tracking motion takes the object coordinate system as the frame of reference. The object coordinate system (O_{obj}) is a coordinate system that is fixedly connected to the object and moves with the object. It is a dynamic coordinate system. The object coordinate system describes the position and orientation of the object on the conveyor, and is the coordinate of the object reference point in the conveyor coordinate system. The conveyor coordinate system (O_{cny}) indicates the position and orientation of the conveyor in the robot base coordinate system, and is used to describe the relationship between the conveyor and the robot. It is a static coordinate system. The vision coordinate system (O_{vis}) describes the position conversion relationship between the camera and the conveyor. The instantaneous position of an object on the conveyor is obtained through the camera, and the movement of the object on the conveyor is obtained in real time by the encoder, enabling the robot to track the position of the object in real time. The relationship between the coordinate systems in the tracking process is as follows.



Relationship between the coordinate systems in the tracking process

3. Functional specifications

1. Conveyor type: Linear, circular.

2. Number of conveyors: 4, at most 2 conveyors can be used at the same time.

3. Detection method: Visual or photoelectric sensor, at most one vision device can be used.

4. Tracking motion instructions: MoveL, MoveC, JumpL.

5. Number of work object types: One conveyor supports at most 16 types of work objects.

6. Visual data type: Robot coordinates or pixel coordinates.

7. Visual communication format: Fixed format (see instruction CnvVison in Section 6.1.5)

8. Number of objects to shoot at one time: 0-10.

9. Object queue length: 500.

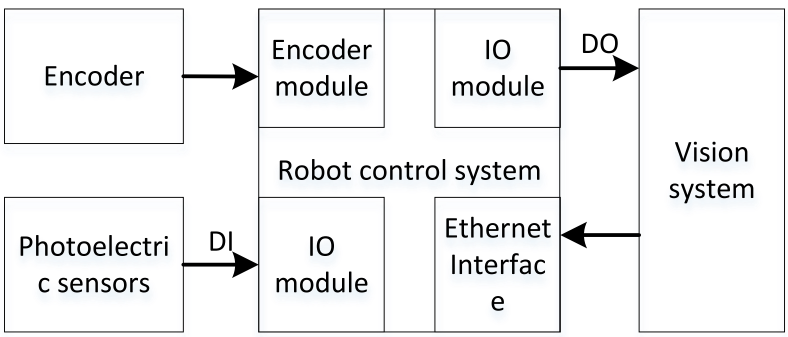
4. Configuration process

The following parts are required to use the tracking process, as detailed in Section 6.1.2-6.1.5:

1. Hardware configuration
2. Coordinate system setting
3. Parameter setting
4. Tracking instructions

6.1.2 Hardware Configuration

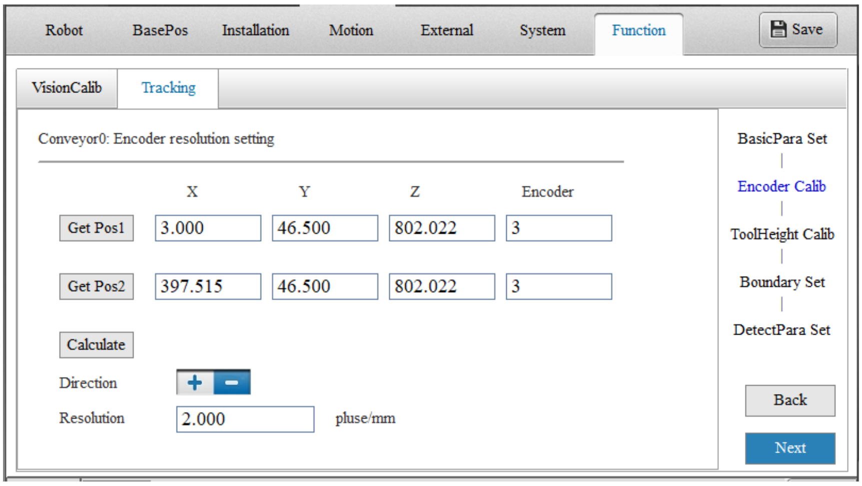
The hardware configuration for the tracking process is shown in the following figure. Photoelectric sensors and vision systems can be selected according to the specific detection method used. If visual detection is used, the camera trigger must be hardware triggered.



Hardware configuration for the tracking process

1. Connect the encoder

Connect the signal and power lines of the pulse encoder to the encoder interface of the robot controller. See the controller user guide for specific wiring logic. After the physical connection is completed, rotate the encoder and take the value of the encoder through the following interface. Observe the changes in the encoder value to determine whether the physical connection is correct.



Read encoder position

2. Connect the photoelectric sensor

The photoelectric sensors output different level signals when triggered by an object, and the robot controller detects the object through the signal edge. Connect the signal wire of the sensor to the normal DI interface of the I/O module of the controller. For the wiring requirements, see the controller user guide. Once the wiring is complete, observe the change of corresponding DI value on the monitoring interface to determine whether the sensor works normally.

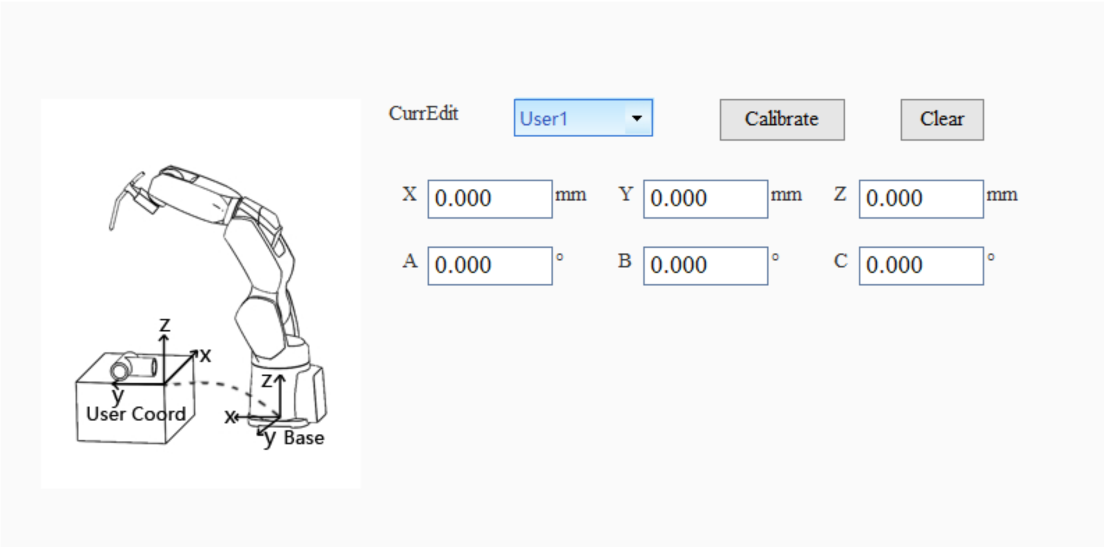
3. Vision

The interaction between vision and the robot consists of DO triggering and Ethernet communication. The DO triggering requires a signal line from the DO interface of the robot controller to be connected to the I/O trigger interface of the camera. Once the physical connection is complete, manually output the DO to see if the camera can be triggered properly.

6.1.3 Coordinate System Setting

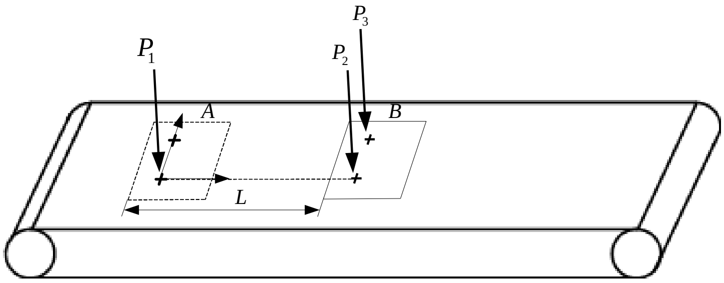
The conveyor coordinate system is a user coordinate system for special functions. To set or calibrate the coordinate system parameters, go to **Edit** > **Crd User**. The conveyor coordinate system describes the position and orientation of the conveyor. The positive x-axis direction coincides with the direction of conveyor movement. The setting is described below.

1) For linear conveyor, use the 3-point method to set the coordinate system, as show below.



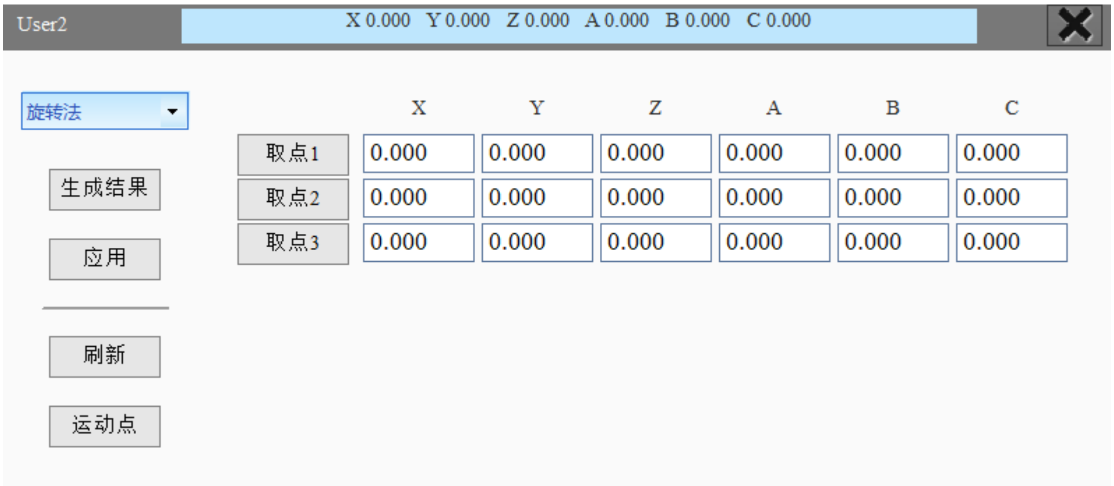
Calibrating the conveyor coordinate system using 3-point method

The three points are selected as shown in the following figure. Place a calibration plate or other calibration reference on the conveyor, and align robot end to the reference point to get point 1, namely, the origin of the conveyor coordinate system. During the movement, keep the relative position of the calibration board and the conveyor unchanged and move from P1 to P2, with P1-P2 being the positive X direction. The Z axis is perpendicular to the direction of the conveyor, and a point P3 in the first quadrant of XY is taken according to the right hand rule.



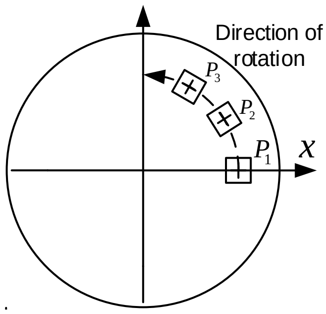
Calibrating the linear conveyor using the 3-point method

If the conveyor is circular, use the rotation method and the calibration interface is shown in the following figure.



Rotation method

2) The rotation method also takes three points, as shown in the figure below. Fix the calibration reference point on the disk, rotate the reference point to the appropriate position and then take the first point P1. The direction from the center of rotation to P1 is the positive X direction of the coordinate system. Rotate the disk counterclockwise, take points P2 and P3 in sequence, and determine an arc from the three points. The center of the arc is the origin of the coordinate system.

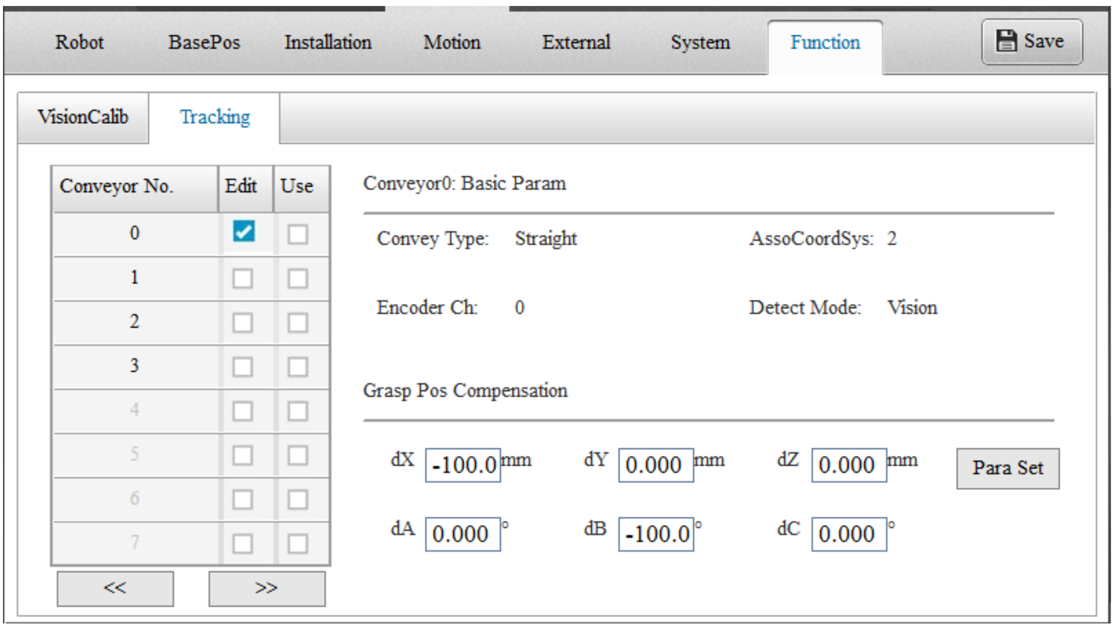


Establishing a circular coordinate system using the rotation method

Note: For accuracy, it is recommended that the three points be positioned as far away from each other as possible. When there is a tool at the end of robot, it is necessary to first calibrate the tool coordinate system and activating the corresponding tool coordinate system.

6.1.4 Parameter Setting

The conveyor parameters include basic parameters, encoder calibration, work object height calibration, boundary parameter settings, and detection parameter settings. According to different detection methods, the detection parameters are divided into vision parameters or sensor parameters. To set the conveyor parameters, go to **Set** > **Function** > **Tracking**, as shown below.



Configuring the tracking process

The main interface contains several parts and their functions are described as follows.

Edit: When ticked, you can view or edit the parameters for that conveyor.

Use: When checked, the conveyor can be used in the user program.

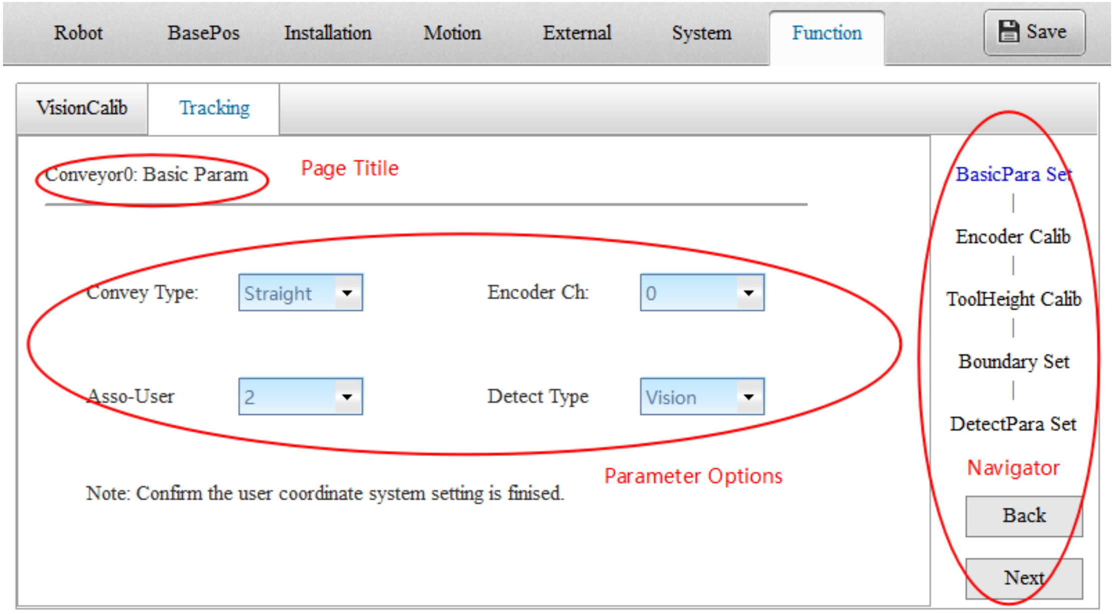
Basic Param: Displays basic information of the selected conveyor. These parameters are for display only and cannot be modified.

Grasp Pos Compensation: Finely adjusts the grasping position.

Parameter Set: When clicked, takes you to the parameter editing page.

1. Basic parameter settings

Click the **Para Set** button to access the parameter setting interface, as shown in the following figure. The setting is wizard-based, click **Back/Next** in the lower right corner to switch the interfaces. The parameters are temporarily in effect when the interface is switched, but are not permanently saved to the controller and will be lost after a power failure. If you want to save the parameters permanently, click the **Save** or **OK** button.



Basic setting interface

You can set the following parameters:

1. Conveyor Type: Straight or circular.
2. Encoder Ch: Selects the signal input channel of the conveyor encoder on the robot controller.
3. Ass-User: Selects the user coordinate system number obtained in 6.3.3 to associate the user coordinate system with the conveyor.
4. Detect Type: Sensor or vision.

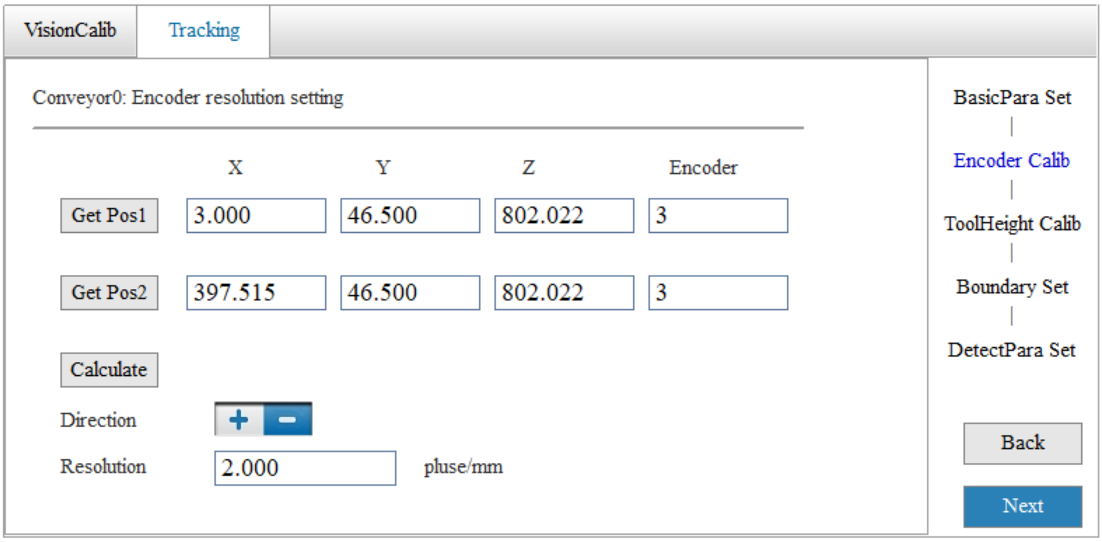
Note: Since the user coordinate system 0 is system-fixed, coincident with the base coordinate system and cannot be modified, select 1-15 when calibrating the conveyor coordinate system.

2. Encoder calibration

The second interface is the encoder calibration interface, which completes the setting of the encoder resolution and direction.

Encoder resolution is the pulse increment of the encoder as the conveyor moves by a unit length (mm or rad).

Encoder direction is the increase or decrease in encoder pulse value as the conveyor moves forward.

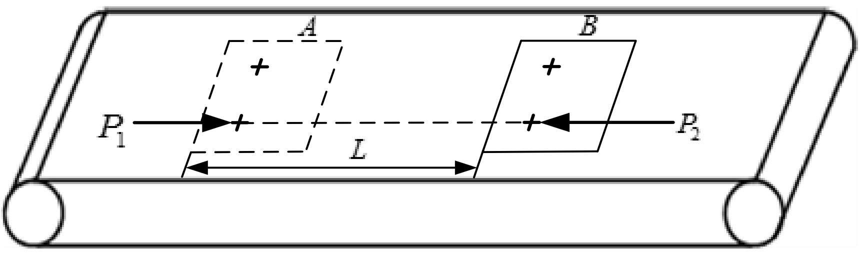


The calculation process is as follows:

1) Place a mark point P1 on the conveyor, align the robot with P1, click **Get Pos1**.

2) Keep the position of the mark point relative to the conveyor unchanged, move the conveyor to reach P2 position, and align the robot with P2, and click **Get Pos2**.

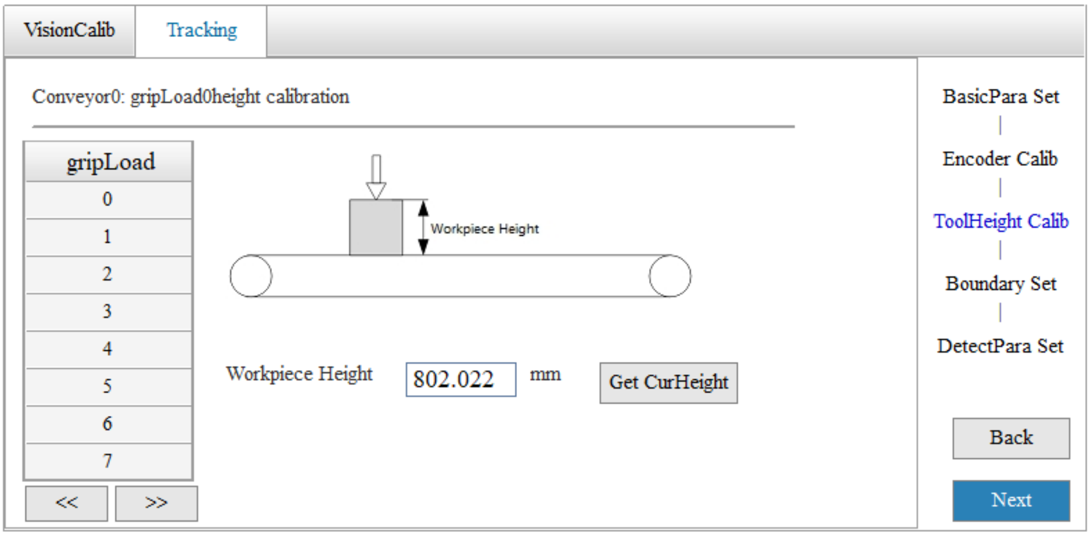
3) Click **Calculate** to complete the automatic calculation of resolution and direction.



Encoder calibration method

3. Work object height calibration

The third interface is the work object height calibration interface, as shown in the figure below. The height of the work load cannot be detected by either the vision system or the sensor, and the height of each work object type needs to be specified by the parameter setting. The work object height value is the Z coordinate value of the work object reference point (the origin of the object coordinate system) in the conveyor coordinate system.



The calibration method is as follows:

1) Select the work object number in the list.

2) Move the end of the robot to the work object reference point and click **Get CurHeight** to complete the automatic height reading. If a tool is mounted at the end of the robot, you need to enable the corresponding tool coordinate system.

4. Boundary parameter setting

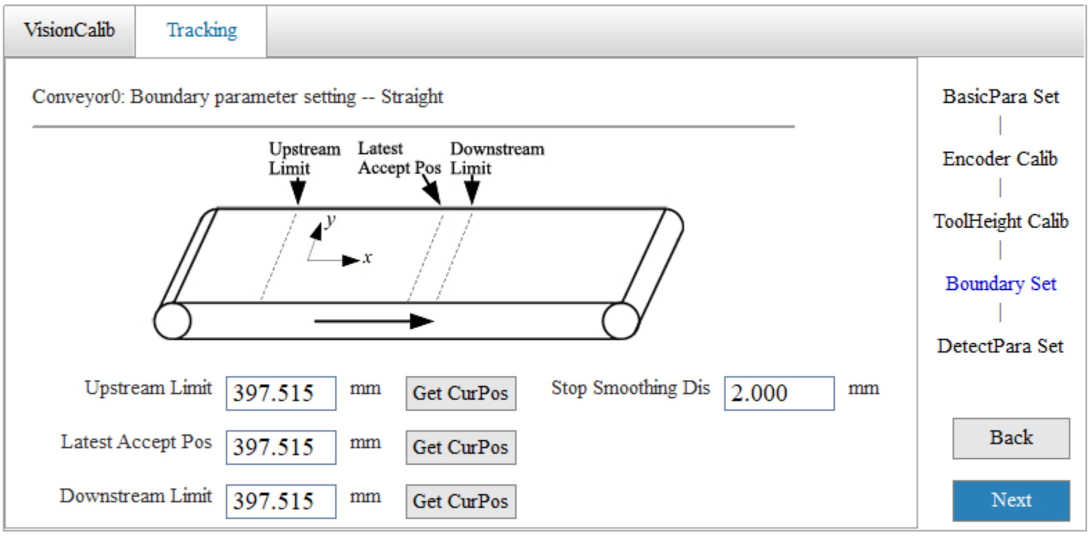
The fourth interface is the boundary parameter setup interface, which is shown in the following figures A (linear conveyor) and B (circular conveyor).

Upstream Limit: Refers to the highest boundary that the robot can reach, and also represents the upper boundary of the object search area. When the object exceeds this boundary, it can be queried by the conveyor inquiry instruction. It is expressed as the x-coordinate value in the conveyor coordinate system. Ensure that the upper boundary is within the right-angle motion range of the robot and away from singular positions.

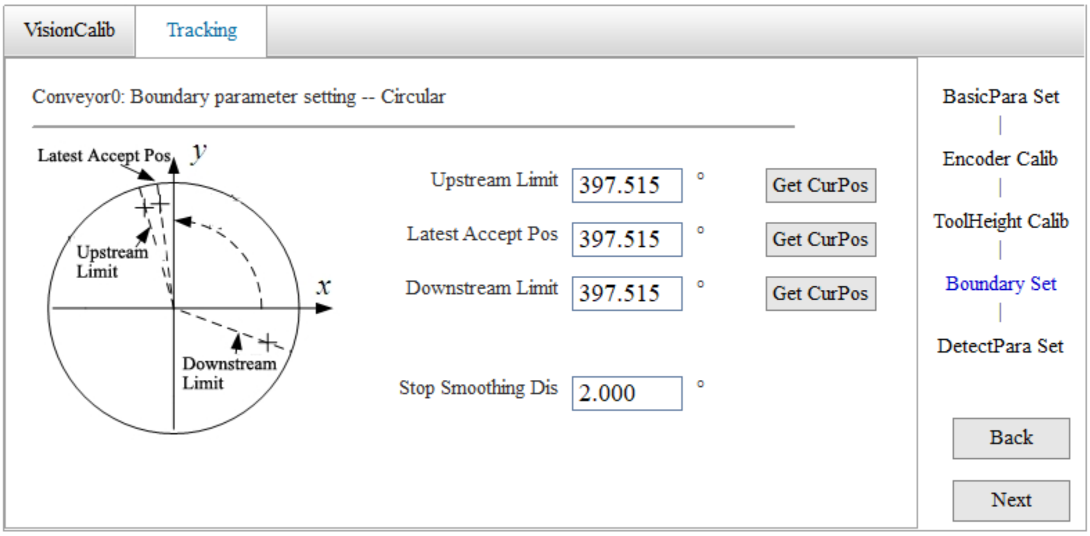
Downstream Limit: Refers to the lowest boundary that the robot can reach. When the distance between the robot and the boundary is less than **Stop Smoothing Dis**, an alarm will be triggered.

Latest Accept Pos: Refers to the lower boundary of the object search area. When an object exceeds this boundary and has not yet been picked up, it will no longer be picked up and will be discarded from the queue. Dynamic gripping takes a certain amount of time. Objects that are close to the downstream limit may exceed the downstream limit before the gripping process is complete, causing an alarm. Set the **Latest Accept Pos** appropriately to prevent this situation. When setting the **Upstream Limit**, make sure the time for the object to move from the **Upstream Limit** to the **Downstream Limit** should be greater than the time taken to grip the object.

Stop Smoothing Dis: Refers to the distance the object smooths to stop when it approaches the **Downstream Limit**. When the distance between the object and the **Downstream Limit** is less than **Stop Smoothing Dis**, the object smooths to stop.



A: Linear conveyor boundary parameters



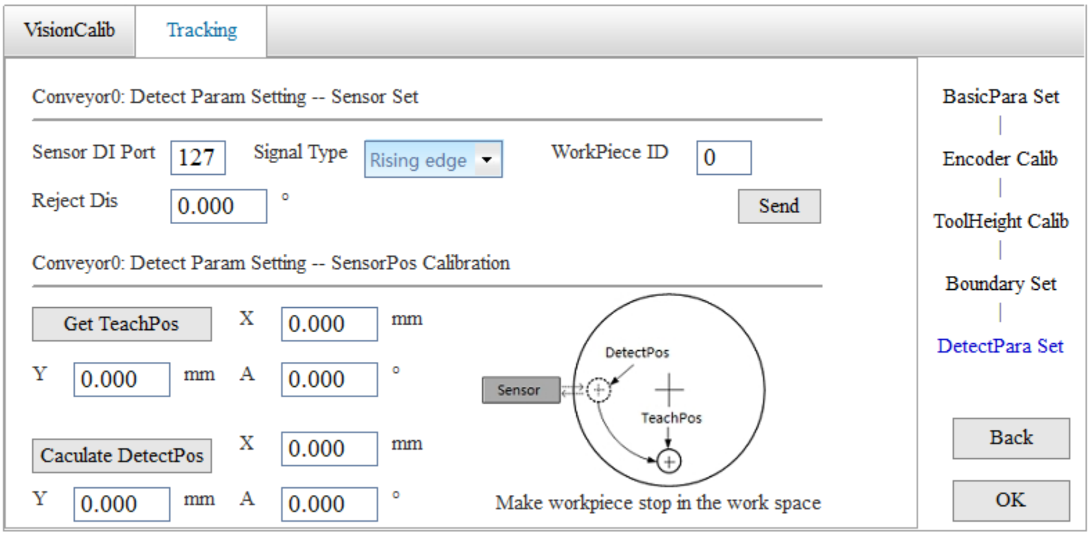
B: Circular conveyor boundary parameters

The **Upstream Limit**, **Downstream Limit**, and **Latest Accept Pos** are three lines perpendicular to the x-axis direction of the conveyor coordinate system, expressed in x-coordinates. These three parameters can be entered directly, or can be obtained by moving the robot to the appropriate positions and clicking **Get CurPos**. The **Stop Smoothing Dis** is normally set to 5° or 5 mm.

5. Detection parameter setting - sensor parameters

The fifth interface is the detection parameter setting interface. There is two types of interfaces depending on the detection method used: sensor, or vision.

1. When sensor detection is used, the parameters and settings are as follows.



C: Sensor parameter setting

Sensor detection parameters include sensor settings and sensor position calibration.

3) The sensor settings are to set the basic parameters of the sensor, including the sensor DI port, signal type, workpiece ID, and rejection distance.

Sensor DI Port: The input port of the photoelectric sensor on the IRLINK module.

Signal Type: The edge of the signal output when the sensor is triggered by the work object.

Workpiece ID: Up to 16 types of work objects are supported on a conveyor. The type of work object cannot be recognized when using the sensor and therefore needs to be specified.

Reject Dis: After a valid signal is detected, if a signal change is detected within a subsequent distance, it is considered that the signal is triggered by the same work object and is automatically rejected as an invalid signal. This distance is called the rejection distance.

After setting the above parameters, click **Send** before performing the sensor position calibration.

4) The purpose of the sensor position calibration is to obtain the pose of the object reference point in the conveyor coordinate system at the moment when the sensor is triggered, which is the detection position on the left side of Figure C. The detailed operations are as follows:

1） Adjust the conveyor speed to the normal working speed, place the object upstream of the sensor, and the object moves with the conveyor. When passing through the sensor, the sensor is triggered. Stop the conveyor after the object moves to the robot's range of motion, ensuring that the sensor is only triggered once during this process.

2） Align the end tool of the robot with the object reference point, click **Get TeachPos**, and the pose of the robot in the conveyor coordinate system is the pose in the object coordinate system.

3） Click **Calculate DetectPos** and the controller automatically calculates the pose of the object at the time the sensor is triggered.

Note:

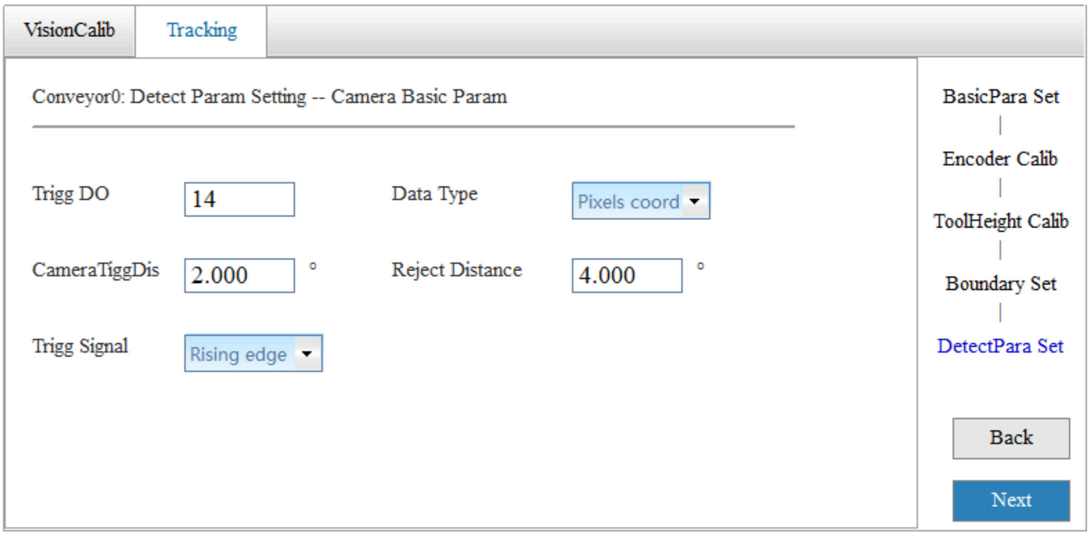
1) Make sure that the workpiece passes through the sensor at the same speed as normal operation during calibration.

2) Do not use hands or other objects to trigger the sensor during calibration.

3) If the robot has a tool at the end, select the corresponding tool number when getting the teach position.

When the settings are complete, click **OK** to finish.

2. If you use vision detection, you need to set vision parameters, including basic parameters and vision calibration, as shown in the following figure.



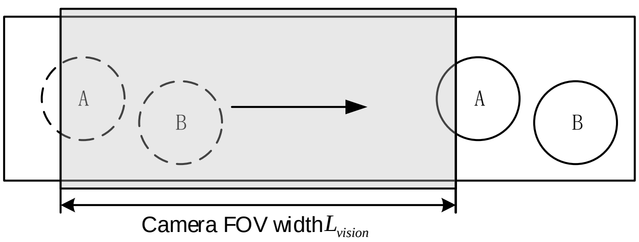
Camera basic parameters

Trigg DO: Input port of the I/O signal that triggers the camera to take a picture on the IRLINK module.

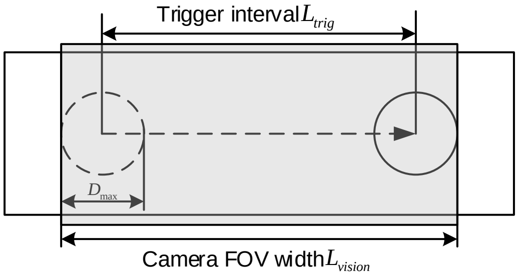
Data Type: The type of change in the output signal of the I/O module when the camera is triggered: rising or falling edge, which should be consistent with the actual situation of the camera.

CameraTiggDis: The visual photography of the conveyor is controlled by the movement distance of the conveyor. Every time the conveyor moves by a CameraTiggDis, the controller sends a DO signal to the camera.

The principle of setting this distance is to ensure that every object identification on the conveyor can be photographed completely, ensuring no missed shots. If the set distance is equal to the width of camera's field of view , and the camera's previous and subsequent shots are seamless, so that each object on the conveyor can be captured. If an object happens to be located at the junction of the two shots, then half of the object will be captured in each shot, and the object cannot be identified. As shown in the figure below, the work object was located at the dashed line position during the first photo taking, and the work object moved to the solid line position on the right during the second photo taking. Both previous and subsequent photos did not recognize the work object A.



To avoid this situation, the previous and subsequent shots should overlap slightly, with an overlap width greater than the maximum width of the object identification, achieving at least one complete photo, as shown in the following figure.



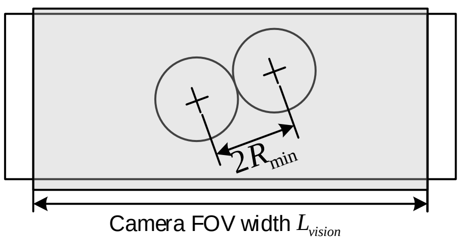
The visual system must complete image recognition within a photography cycle, which is equal to CameraTiggDis/conveyor speed. Setting CameraTiggDis too small can lead to a shorter photography cycle, requiring faster visual processing speed. Therefore, CameraTiggDis should be as large as possible without missing shots. In comprehensive consideration, CameraTiggDis can be set according to the following formula, where L_{trig} is CameraTiggDis, L_{vision} is the width of field of view, $D_{\max}$ is the maximum width of the object detection, and δ_{vision} is the setting margin.

$$\begin{cases} L_{trig} = L_{vision} - D_{\max} - \delta_{vision} \\ \delta_{vision} < 10mm \end{cases}$$

Reject Distance: According to the above CameraTiggDis setting principle, there is overlap between the previous and subsequent photos, which may cause the same object to be repeatedly recognized in the two photos. The distance between the work objects at the time of the two photographs is called Reject Distance.

The controller will compare the positions of the two photographed objects internally. If the distance between the two objects is less than the distance determined by Reject Distance, it is considered a duplicate and the object obtained from the subsequent shot is removed. The principle for setting the Reject Distance is to correctly identify the same object that is overlapped and the two objects that are normally close. As shown in the following figure, two objects can be considered as two different objects when the distance is greater than $2R_{\min}$, otherwise as the same object. Therefore, set the Reject Distance as follows:

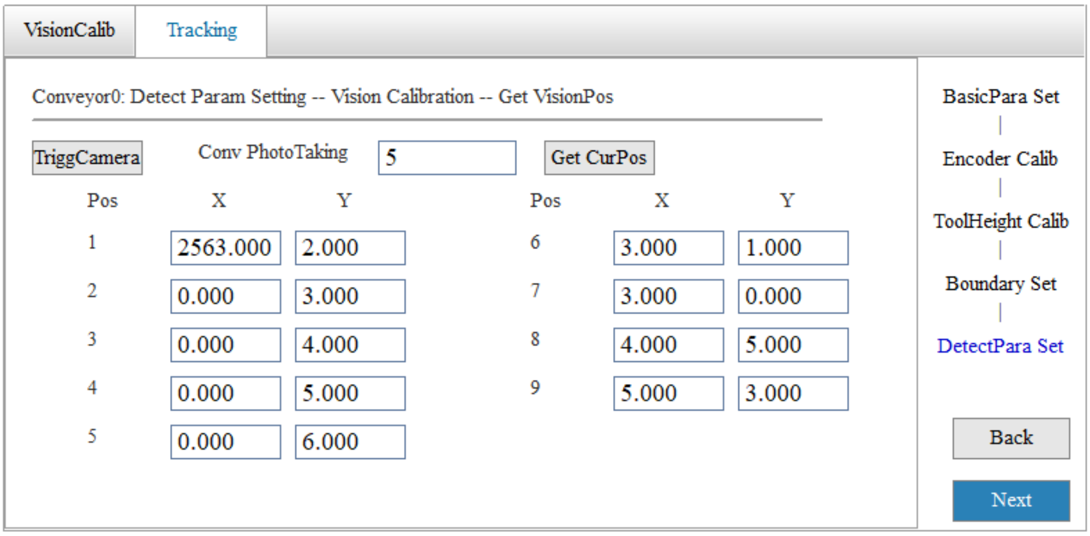
$$L_{remove} < 2R_{\min} = D_{\min}$$



Data Type: Pixel coordinates or robot coordinates. Select "Pixels coord" if the data the vision system sends to the robot is a pixel, otherwise select "Robot coord". (When robot coordinates are selected, the coordinate conversion takes place in the vision system, and the robot controller does not require vision calibration. When performing vision calibration in the vision software, you can still get the coordinates of 9 points of the robot.)

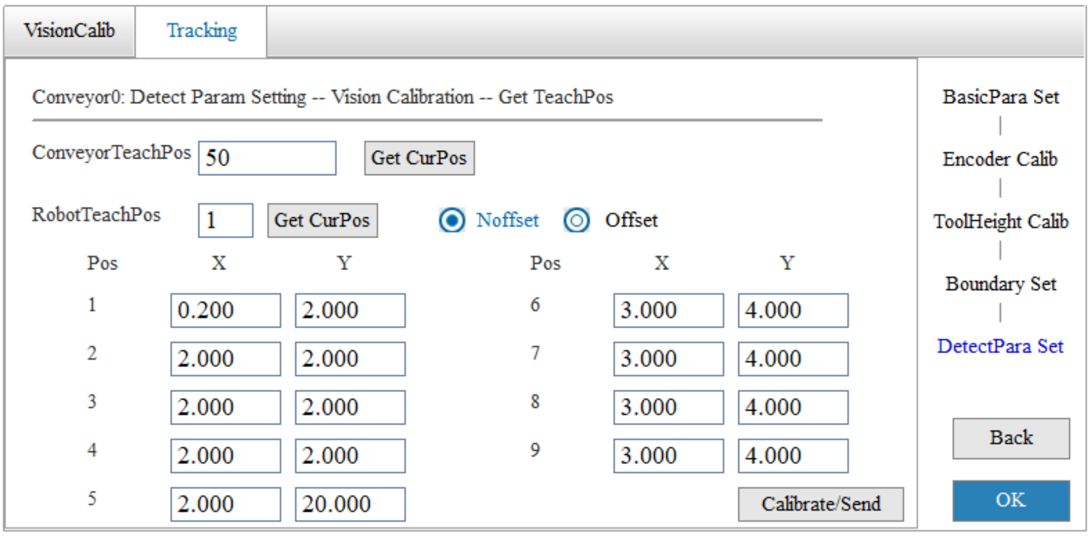
VisionCoordSys: If the data sent by the vision system to the controller is pixel coordinates, vision calibration and coordinate transformation need to be carried out in the controller, and the vision calibration results are stored in the vision coordinates table. This parameter specifies the vision coordinate system number.

If the position information sent by the vision system to the robot is pixel coordinates, vision calibration needs to be performed in the robot controller. After completing the basic parameter settings mentioned above, click **Next** to enter the following interface:



Place the calibration board on the conveyor below the camera. Take photos at the vision end to obtain the pixel coordinates of the 9 points on the calibration board, and fill the coordinates in the corresponding fields. Click **GetCurPos** to read the conveyor position information at the time of the photo.

After you have finished entering the pixel coordinates and reading the photo position, click Next to enter the following interface:



Move the conveyor with the relative position of the calibration board and the conveyor unchanged, and stop the conveyor after the calibration board enters the robot's motion range. Move the end of the robot in order of the pixel points and align the robot with these 9 points. Click **GetCurPo** to obtain the coordinates of the 9 points. Click **GetCurPo** to read the position of the conveyor at this time. After reading the current position of the conveyor and the coordinates of the 9 points, click **Offset** to calculate the coordinates of the 9 points at the photo position. Click **Calibrate/Send** to calculate the calibration results and save them to the vision coordinate system.

Click **OK** to finish all settings and save them.

Note: 1. In the process of vision calibration, it is necessary to activate the corresponding tools. Ensure that you activate the correct tools.

2. If **Data Type** is set to "Robot coord", the robot coordinates entered when performing vision calibration are based on the conveyor coordinate system and have been offset.

6.1.5 Tracking Instructions

1. CnvVison

Function: Opens/closes the conveyor vision port. After the conveyor vision port is opened, the controller automatically stores visually detected objects in a queue, without the need for users to program and process the vision coordinates.

Format: CnvVision(Conveyor[***], ON/OFF, client port number);

Parameter: Conveyor[***] conveyor number, *** range 0-3

Description: Once the conveyor vision is turned on, the vision data needs to be sent to the robot controller in a fixed format:

With objects: TA, X1, Y1, A1 T1, TA, X2, Y2, A2, T2…; (max. 10)

Without objects: NG;

Note that the angle unit is degrees and do not miss the semicolon. One photo corresponds to one packet, and it is not allowed to be sent in multiple times.

2. GetCnvObject

Function: Queries whether there is an object of specified type in the pickup area from the conveyor object queue. If there are any, remove them from the queue. If there are no objects, jump to the next line of instruction.

Format: GetCnvObject(CnvID, ObjID), Goto L[***];

Parameter	Meaning
CnvID	Conveyor number. The range is 0 to 3.
ObjID	Object type number, 0-15
L[***]	Marks the program line to which the program shall return when the target information is not received within time limit. If the target information is successfully received, the program proceeds to the next program line.

Description:

The objects detected by visual or photoelectric sensors are automatically placed in the storage queue of the robot controller, and the controller tracks the position of these objects in real-time. Once the objects enter the pickup area, they can be queried by through GetCnvObject instruction. If there are more than one object in the pickup area, take the most downstream one.

If an object is queried, the object is removed from the storage queue as the target object, and the next line of instruction is executed (without executing the Goto L[***] instruction in the statement). If the object is not queried, the Goto L[***] instruction is executed, i.e., it jumps to L[***].

Example:

```
START;
L[0]:
## Opens the port. External device as server, Address 10.44.53.13, port number 1025;
##Local controller as client, port number 1026.
Open Socket ("10.44.53.13", 1025, 1026, B0);
If B0==0
If B0==0
EndIf;
CnvVision (Conveyor[1], ON, 1026);
P[30]=(0,0,10,0,0,0),(0,0,0,0),(7,0,0);      ##Defines P[30] as a point in the object coordinate
system with coordinates of (0,0,0,10,0,0,0)
L[1]:
Movj P[0], V[30], Z[0];                 ##Moves to wait position
L[2]:
GetCnvObject(1,0's), Goto L[2];      ##Queries queue for type 0 object on conveyor #1
RefSys Conveyor(1,Tool[2]);          ##Switches the motion reference coordinate system to the
object just queried
Movl P[30], V[100], Z[1], Tool[2];        ## Uses tool #2 to move to the P30 point in the object
coordinate system
Set Out[1],ON;                              ##Switches on to absorb object
RefSys Base;                              ##Switches motion reference system to robot
Jump P[1], V[100], Z[0], LH[10], MH[-750], RH[10];          ##Moves the object to P[1]
Set Out[1], OFF, T[0];                             ##Places the object
Goto L[1];
CnvVision (Conveyor[1], OFF, 1026);
Close Socket, 1026;
END;
```

3. CopyCnvObject

Function: Queries whether there is an object of specified type in the pickup area from the conveyor object queue. If there are any, copy them from the queue. If there are no objects, jump to the next line of instruction.

Format: Refer to **GetCnvObject**

Description:

This instruction is basically the same as GetCnvObject in format, syntax, and function, except that CopyCnvObject copies the object from the queue and the object can still be found the next time, while GetCnvObject removes the object from the queue and the removed object cannot be found the next time.

4. RefSys

Function: Switches the motion reference coordinate system.

Format 1: RefSysBase;

Format 2: RefSysConveyor(***, Tool[***]);

Format 3: RefSysWorkBench(***, Tool[***]);

Format	Description
RefSys Base	Switches the motion reference system to base reference system
RefSys Conveyor(***,Tool[***]);	Switches the motion reference system to the coordinate system of the object on the conveyor. Since the object coordinate system is a dynamic coordinate system, the tool end will be synchronized with the object on the conveyor when this instruction is executed. *** Conveyor number, range 0 to 3; Tool[***], the tool number, which indicates that the specified tool end is synchronized with the object on the conveyor.
RefSys WorkBench(***, Tool[***]);	Switches the motion reference system to the table coordinate system. Since the table is a dynamic coordinate system, the tool end is synchronized with the table when this instruction is executed. *** Table number, range 0 to 3; Tool[***], the tool number, which indicates that the tool end is synchronized with the table.

Description: When the motion reference system is switched to a dynamic coordinate system such as a conveyor, table, etc., the description of trajectory and points is relative to the dynamic coordinate system. In the case of a conveyor, the movement of the robot is relative to the object coordinate system after the RefSys Conveyor instruction is performed. When there is no motion instruction, the robot and the object remain relatively stationary (moving synchronously with the object). When there is a motion instruction, the trajectory is in the object coordinate system, and the target point is in the object coordinate system (type 7).

Note: For the conveyor tracking process, the PE parameter must not be included in the motion instruction, and the VelSet instruction is not effective. During the tracking process, instructions such as WaitInPos, Print, etc. that require the robot to be stationary to execute cannot be used.

START;
P[30]=(0,0,10,0,0,0),(0,0,0,0),(7,0,0); ##Defines P[30] as a point in the object coordinate system with coordinates of (0,0,0,10,0,0,0)
Movj P[0], V[30], Z[0]; ##Moves to wait position
L[2]:
GetCnvObject(1,0's), Goto L[2]; ##Queries queue for type 0 object on conveyor #1
RefSys Conveyor(1,Tool[2]); ##Switches the motion reference coordinate system to the object just queried
Movl P[30], V[100], Z[1], Tool[2]; ##Moves to P30 in the object coordinate system, with the movement trajectory being straight

RefSys Base;   ##Switches the motion reference frame to robot
END;

5. Notes

1. After executing RefSys Conveyor, the robot will continue to synchronize with the conveyor until RefSys Base is executed. If the RefSys Base is not executed in time, the robot will follow the conveyor until it goes out of bounds. Therefore, logically, it is necessary to ensure that RefSys Conveyor and RefSys Base exist in pairs. If using Goto, subroutines, etc., it is necessary to return in time to ensure that RefSys Base can be executed. The duration of the program segment between two instructions cannot be greater than the time when the conveyor moves to the boundary.
2. The coordinate system switching instructions shall be used in pairs. At the end of a dynamic tracking, switch to the base coordinate system first. It is not allowed to switch directly between different dynamic reference systems.
3. The motion in the tracking mode is Cartesian trajectory, and the use of joint motion is prohibited. After entering the tracking mode, the robot remains in motion and the instruction to stop the robot will not work.
Prohibited instructions (If used, an error occurs):
    (1) Only Movl/Movc/JumpL instructions are allowed and other motion instructions are prohibited;
    (2) The use of the Until and PE parameters is prohibited in motion instructions;
    (3) Non-motion instructions other than Delay, Set Out, Print are prohibited.
4. In the tracking process, if the robot speed and conveyor speed are large, it is normal to alarm that the joint speed planning is abnormal. In this case, the robot speed or conveyor speed should be reduced.
5. When the coordinate system number in the position variable is 7, only Offset instruction can be used and only offset in X, Y and Z directions can be performed.
6. After version 18, the Z axis in the initial coordinate values of the dynamic point is consistent with the Z direction of the object coordinate system. It is necessary to assign a value to the Z axis according to the actual situation, otherwise a limit alarm will be triggered. (Change the orientation values of the dynamic point).
7. RefSys Base cannot be directly followed by motion instructions with PE. If necessary, use motion instructions with fixed points first, and then use motion instructions with PE. For example,
RefSys Base;
Movj P[0],V[100],Z[0];
Movl Offset (PE, PR0), V[100], Z[0];
8. The use of the GetCnvObject instruction in multitasking can affect the gripping timing in the main task, and may result in alarms, such as joint overspeed, exceeding the working limit, etc.
9. During the process of the robot tracking the conveyor (RefSys Conveyor), if the user manually stops the robot or the robot stops due to alarms, directly restarting the robot cannot complete the tracking movement, and you need to return to the program start line and run the program again.
10. The maximum supported speed of linear conveyor is 500 mm/s, and the maximum

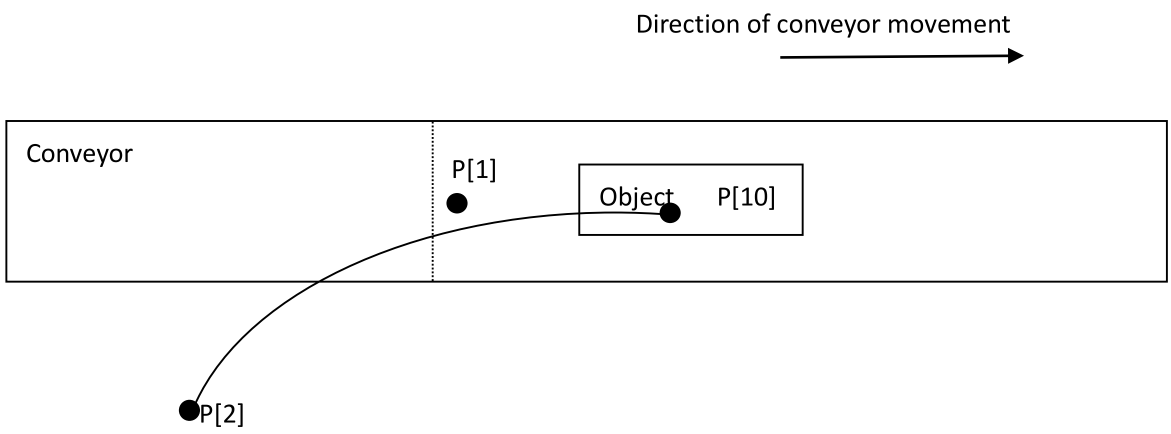
supported speed of circular conveyor is 30 °/s.

6.1.6 Application Cases

6.1.6.1 Dynamic Pick and Place

1) Schematic diagram of points

In the figure below, P[10] is a point on an object, P[1] is a waiting point, P[2] is the placing point. A dynamic object coordinate system (type 7) is used. The robot waits from the P[1] point. An object flows through the upstream limit and is queried by GetCnvObjet. The motion reference system switches to the object coordinate system, and the robot moves to the P[10] point in the object coordinate system to pick up the object. The object is grasped and lift 10 mm off the conveyor surface. Then the object coordinate system is switched to the static coordinate system and moved to the P[2] point and placed.



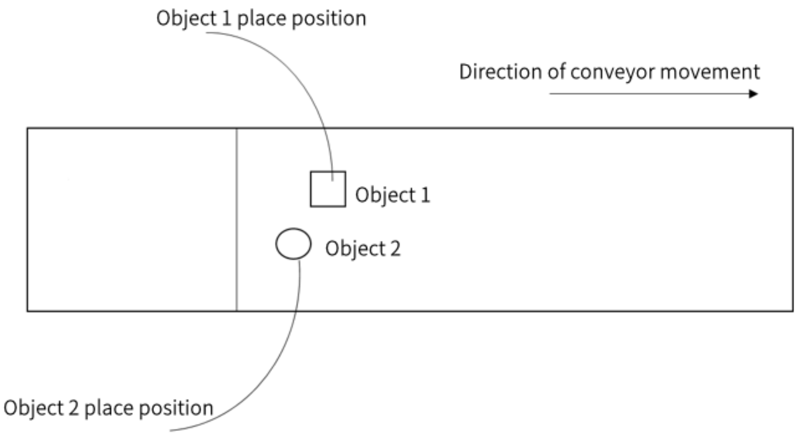
2) Program

START ##Program start flag
B0=0; ##B0 variable initialization
P[10]=(0,0,0,0,0,0),(0,0,0),(7,0,2); ##Defines the point in the object coordinate system: object origin
P[11]=(0,0,0,10,0,0,0),(0,0,0,0,0),(7,0,2); ##Defines the point in the object coordinate system: 10mm above the object
While B0<> 1
    Open Socket (" 192.168.24.55 ", 1025, 1026, B0); ##Establishes a connection to the vision coordinate system
EndWhile;
ConVision(Converyor[0],ON,1026); ##Turns on conveyor vision
L[0]; ##Identifier L[0]
Jump P[1], V[100], Z[0], LH[10], MH[-50], RH[10]; ##Robot wait point, which is normally set near the upstream limit
L[1]; ##Program jump identifier L[1]
GetCnvObjet (0,1), Goto L[1]; ##Queries object. If there is an object, continue to execute the following instructions; otherwise, jump to L[1].
ResSys Converyor[0]; ##Switches motion reference system to the coordinate system of the

object on the conveyor
JumpL P[10], V[100], Z[0], LH[0], MH[10], RH[0]; .##Moves to P[10] in the object coordinate system
Set Out[8], ON; ##Switches on to absorb object
Movl P[11], V[30], Z[0]; ##Lifts the object 10mm off the conveyor surface
RefSys Base ##Unsynchronized from the conveyor and the motion reference system switches to the robot
Jump P[2], V[100], Z[0], LH[0], MH[10], RH[0]; ##Runs to the unloading point P[2]
Set Out[8], OFF; ##Turns off switch and places the object
Goto L[0]; ##Program loop
END ##Program end flag

6.1.6.2 Dynamic Sorting

1） Schematic diagram of points



When there are two or more objects on the conveyor, the vision system recognizes the attributes of the objects and sends them to the robot, which can sort the different objects according to the object attributes in the sent data.

2) Program

START;
P[11] =(0,0,0,0,0,0),(0,0,0,0),(7,0,0);
L[0]:
Jump P[1],V[100],Z[0],Tool[0],LH[0],MH[-800],RH[0]; ##Waiting position
L[1]:
GetCnvObject(1,1),Goto L[2]; ##Queries for object #1
RefSys Conveyor(1,Tool[0]);
Jump P[11],V[100],Z[0],Tool[0],LH[0],MH[-800],RH[0]; ##Picks up object #1
RefSys Base;
Jump P[2], V[100], Z[0], Tool[0], LH[0], MH[-800], RH[0]; ##Places the object in unload zone #1
L[2]:
GetCnvObject(1,2),Goto L[1]; ##Queries for object #2
RefSys Conveyor(1,Tool[0]);

```
Jump P[11],V[100],Z[0],Tool[0],LH[0],MH[-800],RH[0];   ##Picks up object #2
RefSys Base;
Jump P[3], V[100], Z[0], Tool[0], LH[0], MH[-800], RH[0];        ##Places the object in unload zone #2
Goto L[0];
END;
```

Note:
1. Dynamic sorting supports 16 different types of objects, and the data of up to 10 objects can be sent in visual photography.
2. The target objects for sorting need to be of similar size.

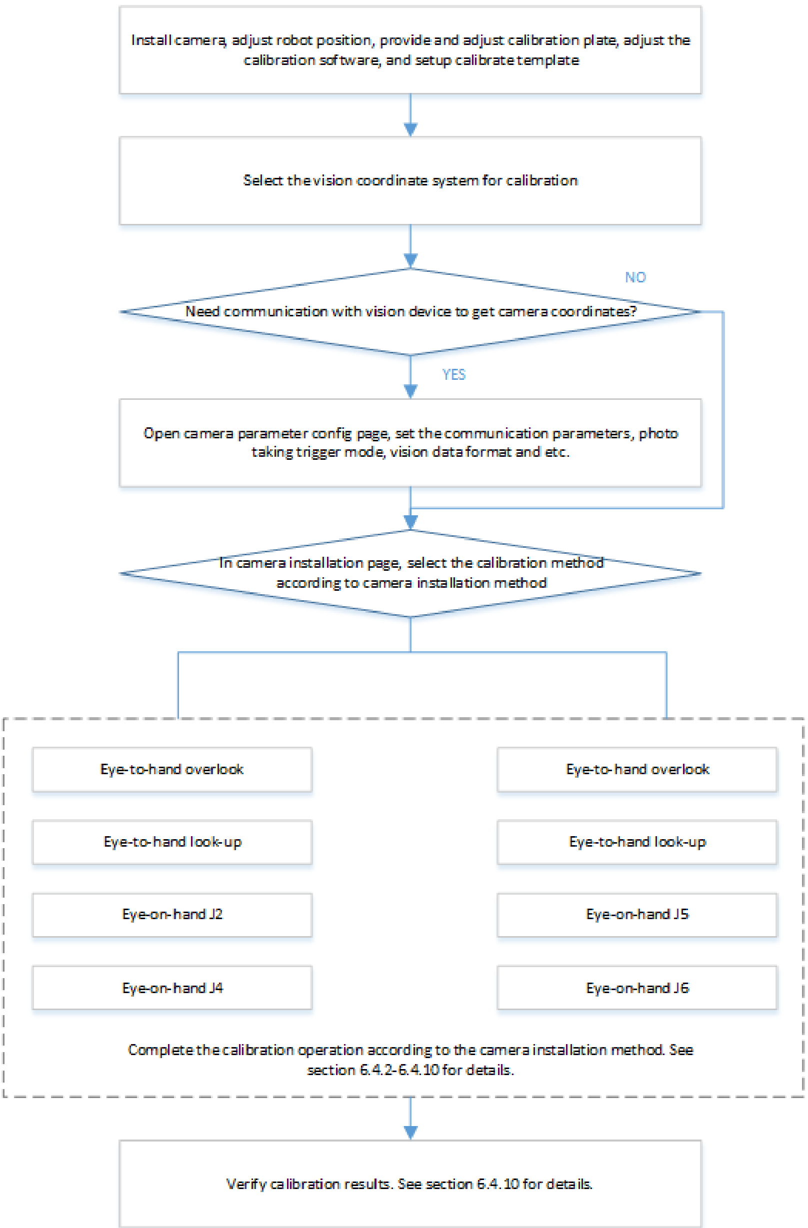
6.2 Vision Calibration

6.2.1 Overview

Vision calibration is a prerequisite for the use of vision functions in robot systems. Calibration is to obtain the relative position relationship between the camera and the robot, so that the pixel coordinates can be transferred to the robot coordinates in other subsequent operations. Inovance Technology offers SCARA robots and 6-axis robots with the following differences in the operational steps of the vision calibration function.

Calibration Process	SCARA Robots	6-Axis Robots
Calibrate the end fixture	-	Calibrate the end fixture
Calibrate the user coordinate system	-	Calibrating the user coordinate system (Using the calibrated end fixture)
Calibration parameters setting	Go to the vision calibration interface and select parameters such as calibration method, camera installation mode, etc.	Go to the vision calibration interface and select parameters such as calibration method, camera installation mode, etc.
Select the reference point	Select two reference points	Select three reference points
9-point teach calibration	Perform 9-point teach calibration	Perform 9-point teach calibration
Completed	Completed	Completed

The visual calibration main process is shown in the following figure:



6.2.2 Vision Calibration of SCARA Robots

The vision calibration of SCARA robots includes eye-to-hand overlook calibration, eye-to-hand look-up calibration, eye-on-hand J2 calibration, and eye-on-hand J4 calibration. All controller versions support vision calibration of SCARA robots.

Due to structural limitations, SCARA robots generally operate in the user plane parallel to the base plane, and the calibration process does not need to be associated with the user coordinate system, and only two reference points need to be selected in the reference point teaching. The vision calibration process of SCARA robots is shown in Figure 2-1:

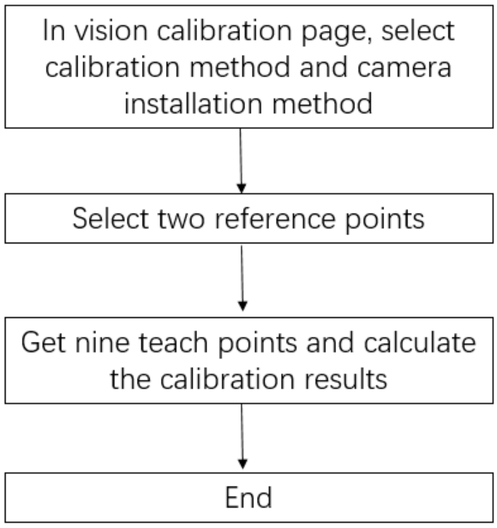


Figure 2-1 Vision calibration process of SCARA robots

6.2.3 Vision Calibration of 6-Axis Robots

The vision calibration of 6-axis robots includes eye-to-hand overlook calibration, eye-to-hand look-up calibration, eye-on-hand J5 calibration, and eye-on-hand J6 calibration. The controller after version 17 can support visual calibration of the 6-axis robots .

A 6-axis robot can not only operate on a user plane parallel to the base plane, like a SCARA robot, but also on a user plane that is not parallel to the base plane (but the Z direction must point upwards). During its calibration process, it is necessary to select the associated user coordinate system. When selecting the associated user coordinate system, it is necessary to confirm that both the tool and the user coordinate systems have been calibrated. In the reference point teaching, it is necessary to select three reference points to determine the calibration tool (Note: when calibrating the reference points, try to make the camera and calibration board in a relatively parallel state, and the orientation between every two reference points needs to change by more than 20°). The vision calibration process of 6-axis robots is shown in Figure 2-2:

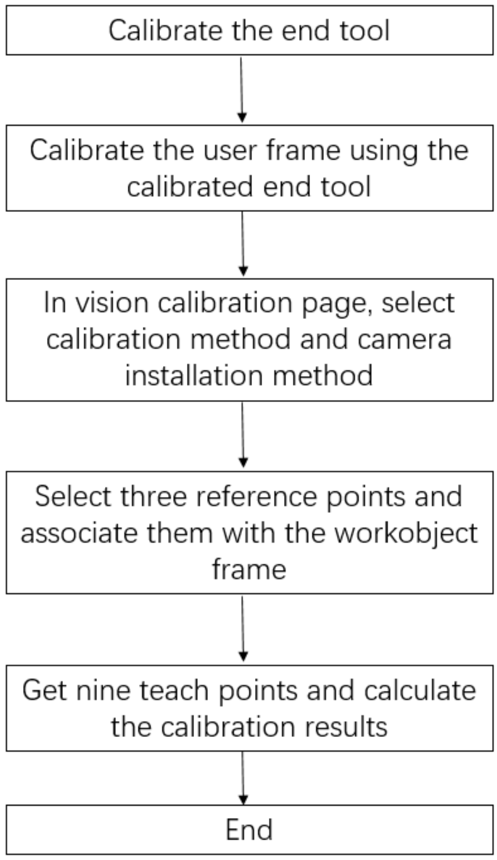


Figure 2-2 Vision calibration process of 6-axis robots

NOTE: During the calibration process, attention needs to be paid to the introduction of vision calibration errors. Factors affecting camera calibration accuracy mainly include human calibration accuracy, fixture accuracy, visual inspection accuracy, absolute accuracy of the manipulator, and calibration algorithm accuracy.

6.2.3.1 Operation Procedure

Depending on how the camera is mounted, different calibration methods are used, see sections 6.4.4-6.4.9.

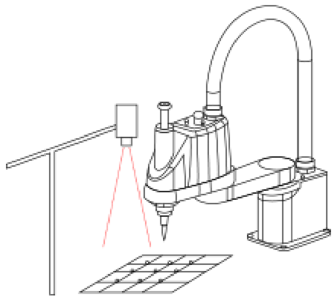
Eye-to-hand overlook calibration	6.4.4 Eye-to-Hand Overlook Calibration
Eye-to-hand look-up calibration	6.4.5 Eye-to-Hand Look-up Calibration
Eye-on-hand J2 calibration	6.4.6 Eye-on-hand J2 calibration
Eye-on-hand J4 calibration	6.4.7 Eye-on-Hand J4 Calibration
Eye-on-hand J5 calibration	6.4.8 Eye-on-hand J5 calibration
Eye-on-hand J6 calibration	6.4.9 Eye-on-hand J6 calibration

6.2.3.2 Results Validation

Calibration results obtained by various means of calibration can be verified by programming to see if the calibration results meet the requirements. See 6.4.10 Calibration Results Verification.

6.2.4 Eye-to-Hand Overlook Calibration

The camera is mounted above the calibration board, looking down, as shown in the following figure:

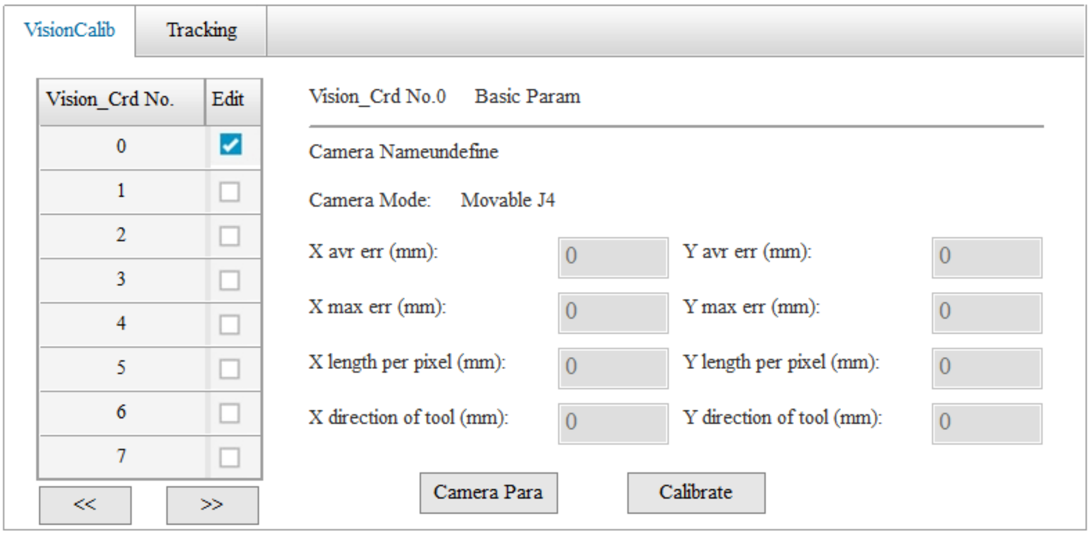


6.2.4.1 Mounting Condition

As shown in the figure, there is a nine grid calibration board directly below the camera, which contains 9 marker points that need to be calibrated by the robot. Keep the camera, robot, and calibration board on the same horizontal plane. Make sure that the 9 marker points on the calibration board are clearly imaged in the lens. Install fixtures at the end of the robot, which can be either tip fixtures or fixtures that can be recognized by the vision system.

6.2.4.2 Calibration Procedure

Step 1: Install the camera, position the robot, place and adjust the calibration board, adjust the vision software, and create calibration templates, etc.
Step 2: Select the vision coordinate system to calibrate, as shown in the following figure.



Step 3: Click the **Calibrate** button to enter the camera parameters setting interface.
Communication parameter configuration: Sets the vision system information, including the camera IP, port number, and camera name (optional). When you have finished setting up, click **Connect** to establish communication.

Camera trigger mode: I/O trigger, Ethernet trigger.
When you select I/O trigger is selected, you need to set the following parameters:

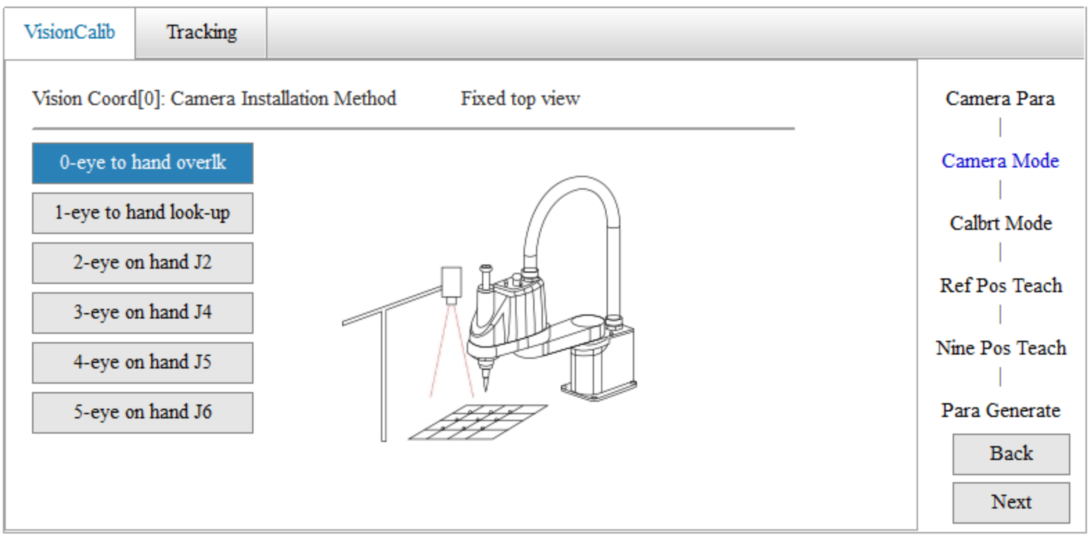
When you select Ethernet trigger, you need to set the string that triggers the camera to take a picture:

Reception data format: Includes head, separator, and tail.
If you do not need to communicate with the vision system to get the camera coordinates, select **Next**.

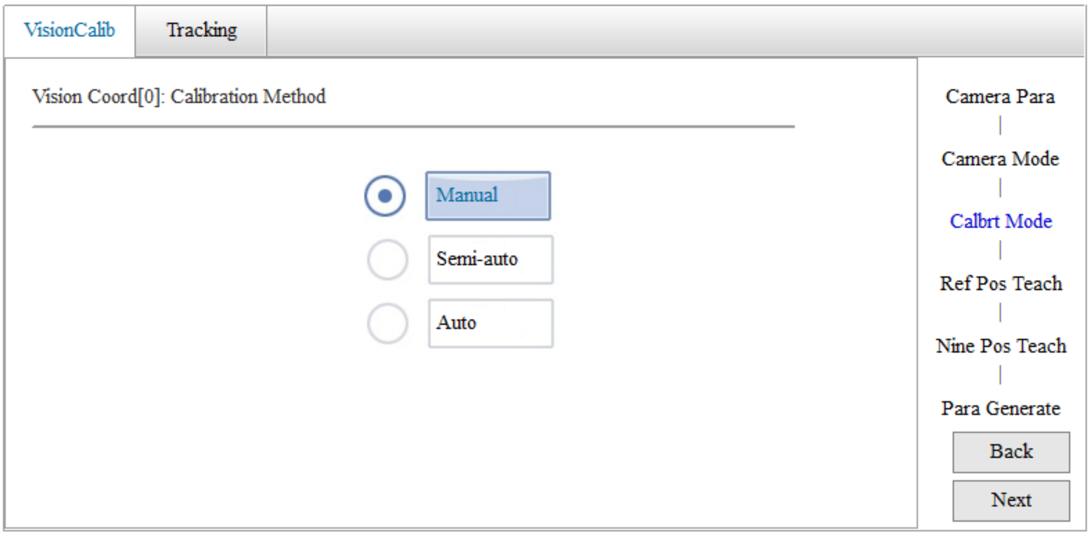
Note:
1. When the camera trigger mode is set to Ethernet trigger, click **Comm-Test**. If the camera has sent data, but the data is not received, check the network cables and check the connection between the camera and the robot controller.

6.2.4.3 Camera Communication Test

It is to test communication with the vision software. The robot controller triggers a signal (I/O signal or a string sent) to the vision side, the string sent by the vision side is displayed in the data reception area. Check if the sent data format is consistent with the specified format.
Step 4: Go to the camera installation selection interface, select "0-eye to hand overlk" and click **Next**.



Step 5: Select the calibration method, and click **Next**.



There are three methods of calibration.

Manual calibration: You need to manually teach the robot to calibrate the 9 marker points in the calibration board and obtain the corresponding camera coordinates.

Semi-automatic calibration: Set 3 marker points on the pallet and generate 9 marker points by calibrating the pallet. The robot automatically runs to 9 marker points and obtains the corresponding camera coordinates.

Automatic calibration: 9 marker points are automatically generated by teaching the center of the field of view and setting the distance between 9 points, the robot automatically runs to 9 marker points and obtains the corresponding camera coordinates.

Step 6: Go to the reference point acquisition interface. Figure 1 shows the acquisition of reference points for the SCARA robot, Figure 2 shows the acquisition of reference points for the 6-axis robot reference point. Click **Next** when finished.

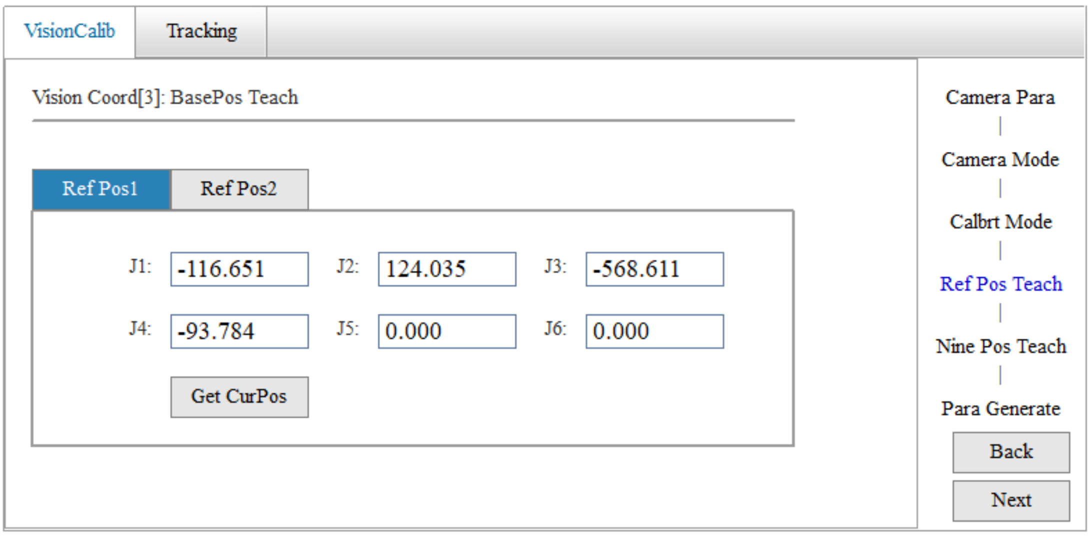


Figure 1

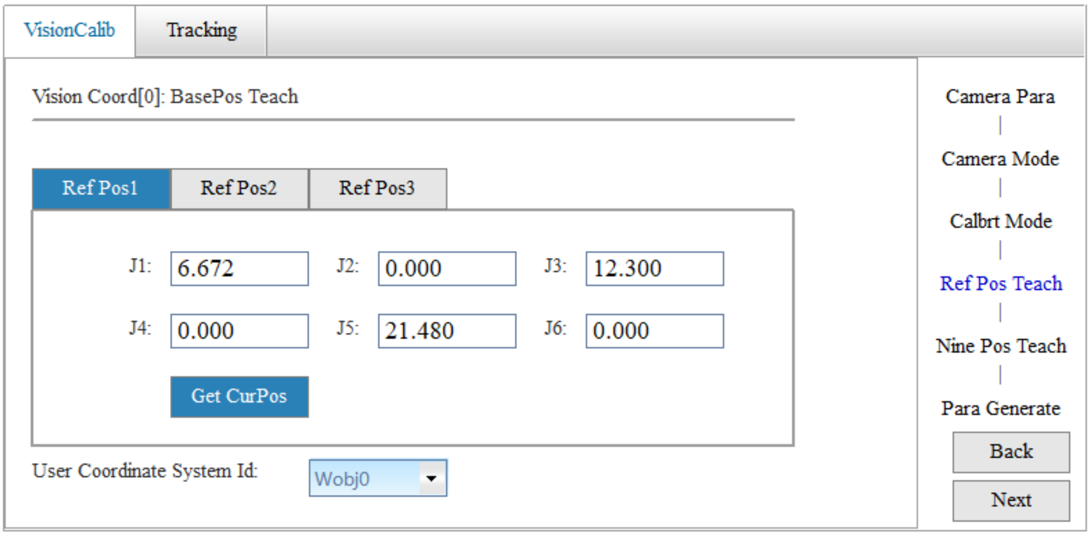


Figure 2

Note: Two reference points are required for the SCARA robot and three for the 6-axis robot. Set the reference point according to the calibration method selected.

If the end calibration fixture of the robot is a tip fixture, teach the robot to align the tip of the fixture with the marker point in the center of the calibration board. For the SCARA robot, teach it with two reference points. First, obtain the position of the robot as the reference point 1, and then adjust the robot's orientation by rotating it at a certain angle and re-aligning it with the marker point, so that the position of the robot is obtained as the reference point 2.

Note: Do not teach the robot with two points by changing the arm type.

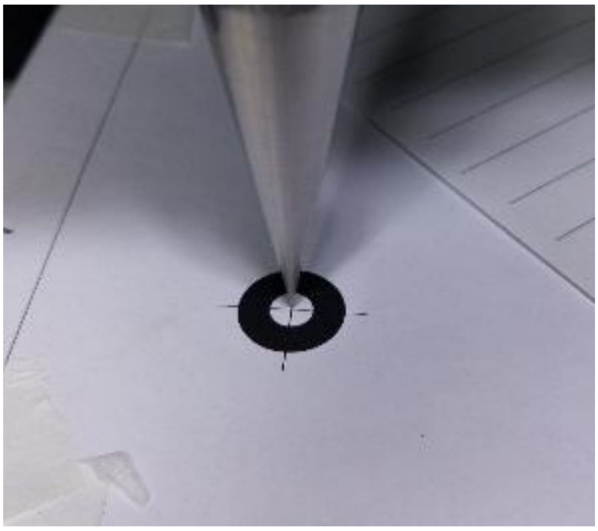
Unlike the SCARA robots, you need to teach the 6-axis robots with three reference points (Unlike the SCARA robots, these three points are obtained according to the method of calibrating the tip tool of the 6-axis robot. For the calibration method of the tip tool, see 4.2 Coordinate System Settings, and the steps are the same as those for the SCARA robots.)

Note:

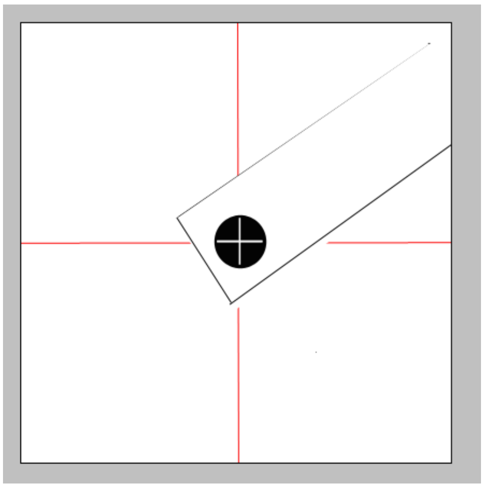
1. When calibrating the reference points, try to keep the camera and calibration board in a relatively parallel state, and the pose angle interval between every two reference points should be as large as possible.

2. In vision calibration of the 6-axis robots, you need to select the associated user coordinate system, which is obtained from the calibration of the robot with end calibration fixtures.

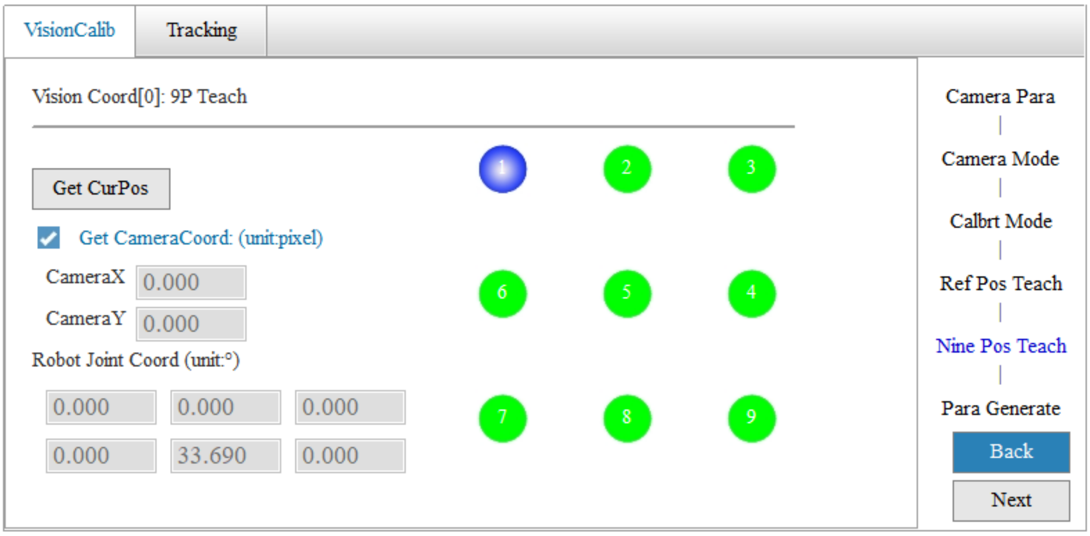
3. Before vision calibration, calibrate the tool first, and then calibrate the user coordinate system.



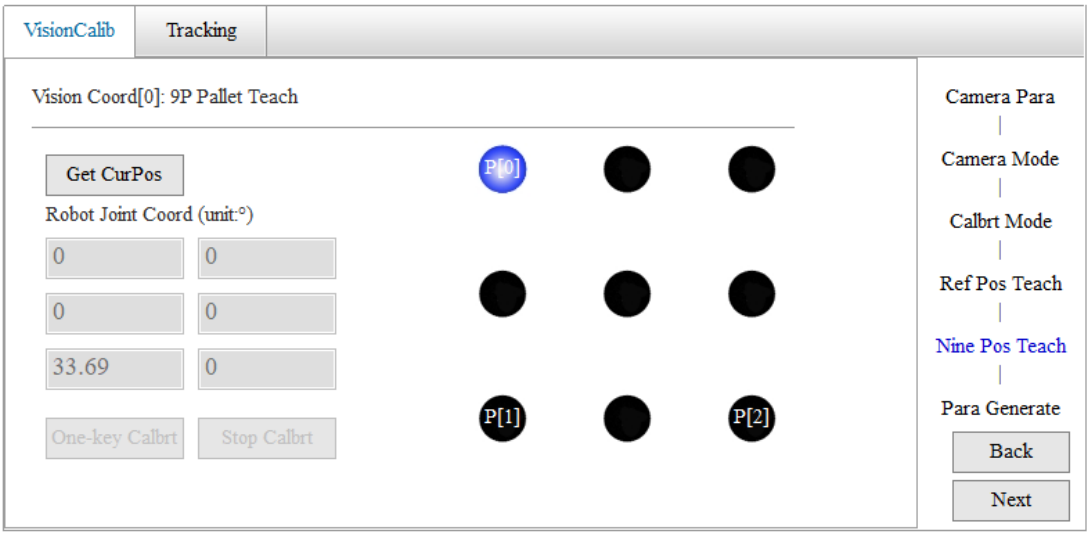
If the end calibration fixture of the robot is a visually recognizable template tool, you can adjust the orientation of the robot and rotate the calibration fixture by a certain angle to obtain other reference points.



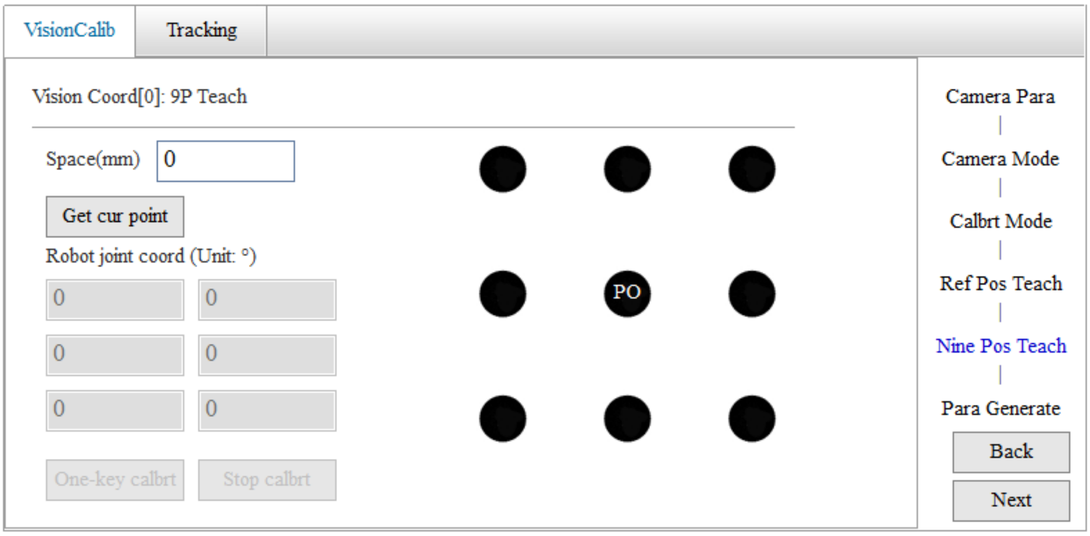
Step 7: Go to the 9-point teaching interface, click **Next** when you are finished.
Manual calibration: Since all the pixel coordinates are photographed at one time, the individual pixel coordinates cannot be acquired by communication, but only by manual input. You can uncheck **Get CameraCoord** and then manually enter the pixel coordinates. Adjust the robot position, get the 9 positions of the robot on the calibration board through manual calibration and enter the corresponding 9 pixel coordinates.



Semi-automatic calibration: Get three positions P[1], P[2], P[3] in the field of view of camera, click **One-key Calbrt**. The robot automatically moves to the 9 corresponding positions in the field of view map, saves the robot coordinates and the corresponding pixel coordinates until the motion is completed.

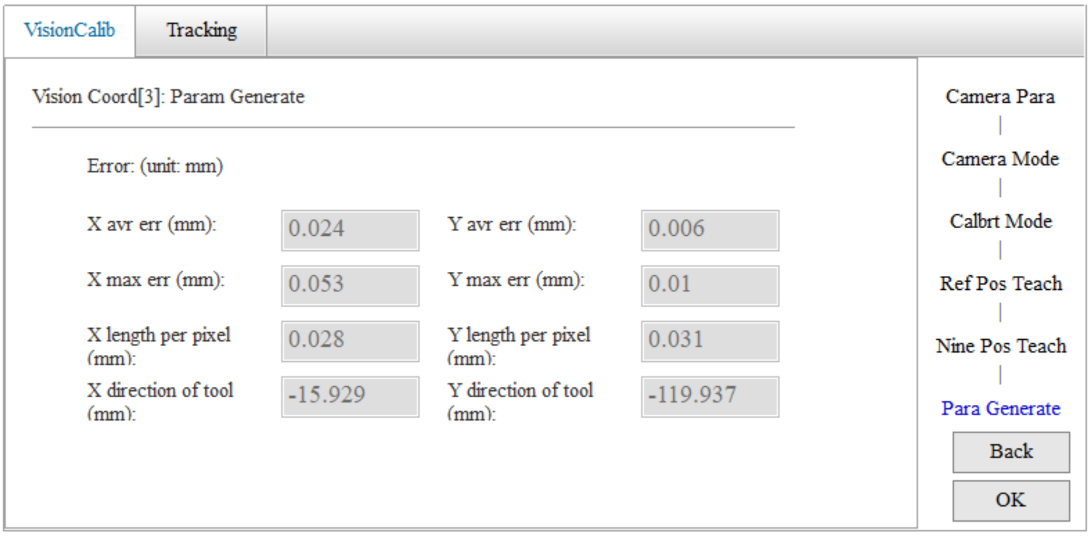


Automatic calibration: Get the position P[0] of the robot in the field of view of camera, set the **Space** appropriately so that the final nine points are all within the field of view of the camera, click **One-key Calbrt**, and the robot will automatically run to the nine points, save the robot coordinates and corresponding pixel coordinates until the motion is completed.



Step 8: Go to the teaching point list and check if each of the position coordinates and pixels of the robot is normal. You can also double-click the points in the list to modify the point data and click Next.

Step 9: The system calculates and generates a vision coordinate system calibration matrix and calibration result parameters as shown in the following figure.



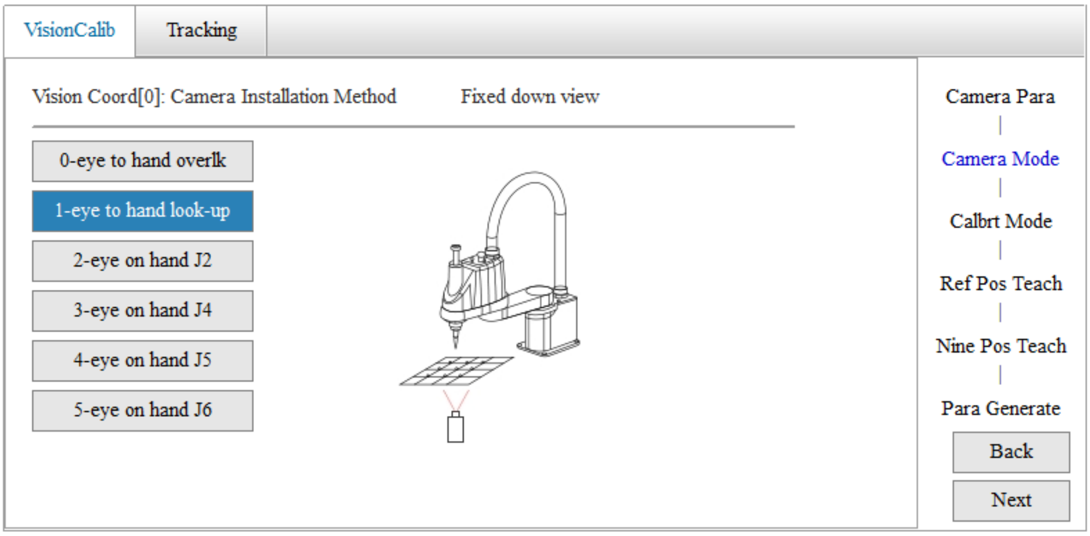
The parameters include average error in the X direction, average error in the Y direction, maximum error in the X direction, maximum error in the Y direction, unit pixel size in the X direction, unit pixel size in the Y direction, offset of calibration tool in X direction, offset of calibration tool in Y direction, and other parameters.

The offset of calibration tool in X direction and offset of calibration tool in Y direction can be used as tool parameters without rotation direction for the calibration tool.

Step 10: Click **OK** to complete the vision calibration.

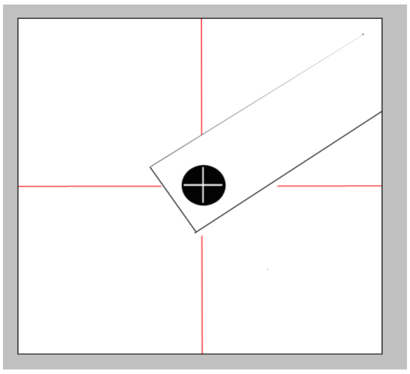
6.2.5 Eye-to-Hand Look-up Calibration

The camera is mounted below the robot, looking up, as shown in the following figure.



6.2.5.1 Mounting Condition

As shown in the figure, the camera is installed directly below the robot, looking upwards. Keep the camera and robot on the same level. Install fixtures at the end of the robot as visually recognizable fixtures. Adjust the camera focus and the height of the robot to ensure that the marked points on the fixture are clearly identified in the field of view.

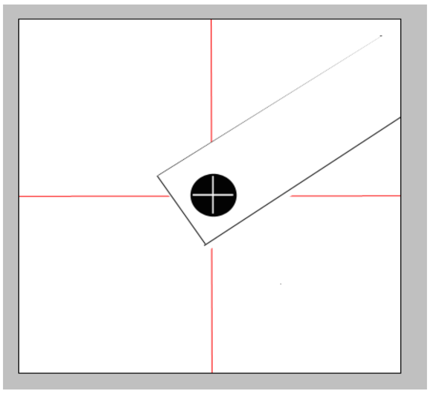


6.2.5.2 Calibration Procedure

Step 1-5: Same as steps 1-5 in 6.4.4 Eye-to-Hand Overlook Calibration. Complete setting of basic camera parameters, camera mounting mode, and calibration mode.

Step 6: For SCARA robots, move the marker point on the fixture to the center of the field of view, as shown in the above figure, to obtain reference point 1. Adjust the orientation of the robot, rotate the fixture and then move the marker point to the center of the field of view to get reference point 2. For a 6-axis robot, it is necessary to obtain three reference points (Note: When calibrating the reference points, try to make the camera and calibration board relatively parallel, and the orientation between every two reference points needs to change by more than 20°.), and select the user coordinate system associated with the calibration table.

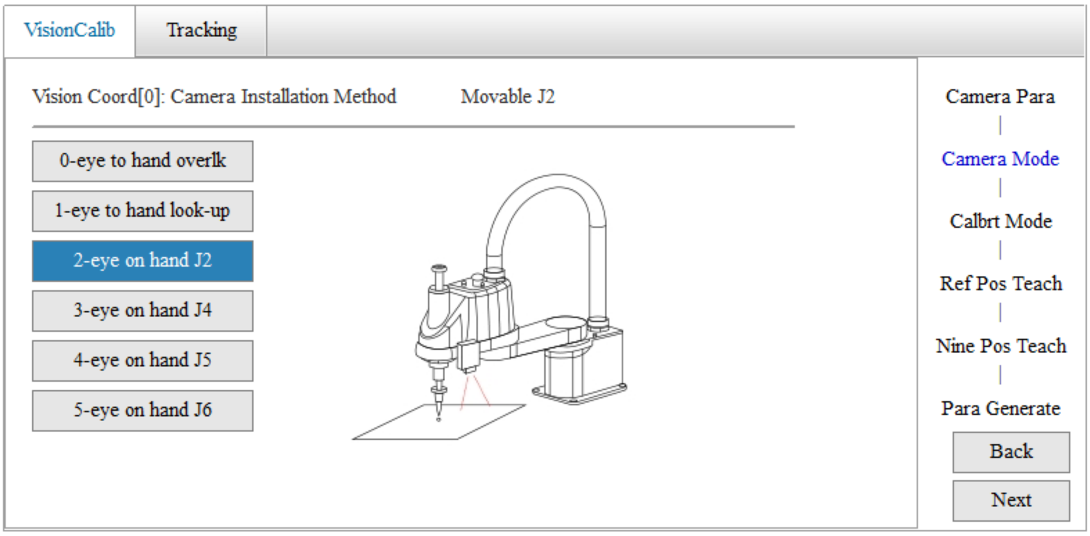
Note: Make sure that the plane where the reference points are selected is consistent with the user coordinate system associated with the calibration when calibrating the 6-axis robots.



Step 7: Complete the 9-point calibration as in Step 7 of 6.4.4 Eye-to-Hand Overlook Calibration. The point selection method in manual calibration is consistent with step 6. Manipulate the robot, move the marker points on fixture to obtain 9 points evenly distributed in the field of view. Also, obtain the corresponding vision coordinates, which can be obtained through manual input or automatic acquisition.
Step 8-10: Same as Step 8-10 in 6.4.4 Eye-to-Hand Overlook Calibration.

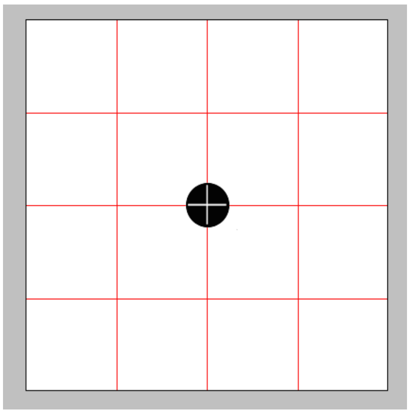
6.2.6 Eye-on-Hand J2 Calibration

The camera is mounted on the J2 axis, looking downwards, as shown in the following figure.



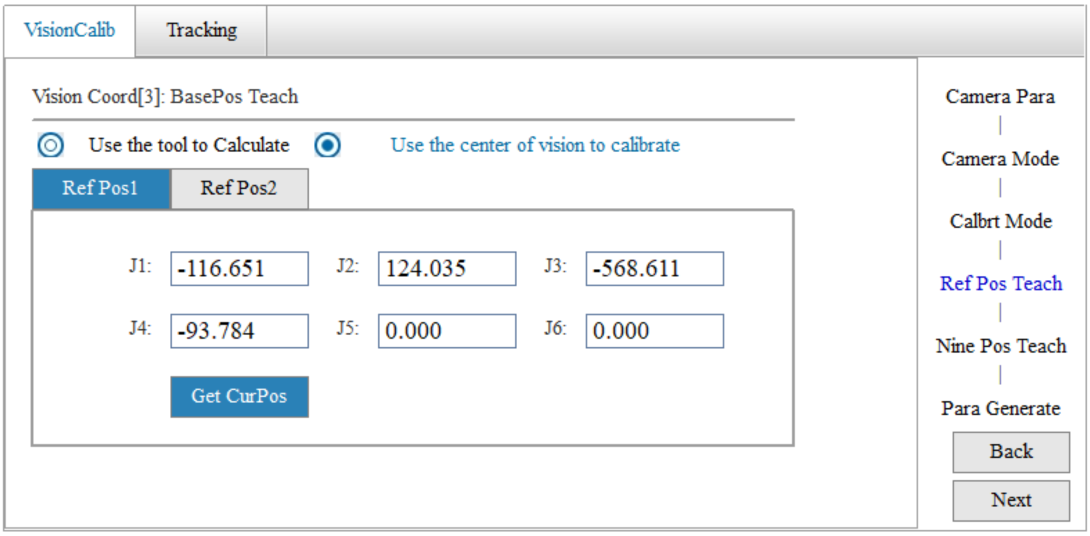
6.2.6.1 Mounting Condition

The camera is mounted on the J2 axis. Keep the camera plane parallel to the horizontal plane without calibration fixtures, as shown in the following figure. The black marker point in the figure represents the marker point on the calibration board, the red lines represent the grid lines in the camera's field of view, and the intersection of the grid lines is the required marker point.



6.2.6.2 Calibration Procedure

Step 1-5: Same as steps 1-5 in 6.4.4 Eye-to-Hand Overlook Calibration. Complete setting of basic camera parameters, camera mounting mode, and calibration mode.



6. Move the robot and adjust the camera pose to align with the center marker point to obtain two different points as reference points.

Note:

1) If you use the camera field of view center to calibrate the reference point, select **Use the center of vision to calibrate**. The tool parameters displayed after calibration represent the deviation of the camera field of view center from the end of the J2 axis.

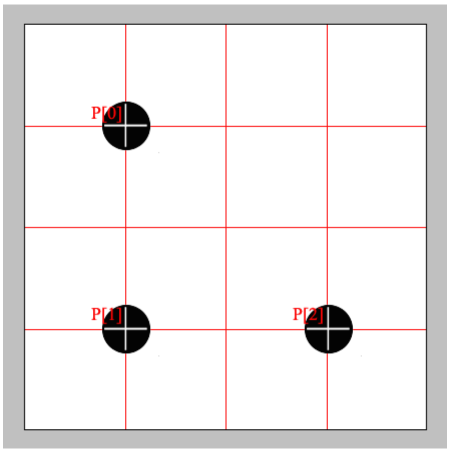
2) If you use the robot's end tool to calibrate the reference point, select **Use the tool to calibrate**. The tool parameters displayed after calibration are those of the tool installed at the end of the robot.

3) If you choose to use the camera as a tool for manual calibration, then do not use the end tool.

7. Click **Next** to go to the 9-point calibration interface.

7-1. Manual calibration: Move the robot, align the nine marker points in the field of view with the marker points on the calibration board to obtain the robot coordinates and pixel coordinates respectively. The camera coordinates can be automatically obtained or entered manually.

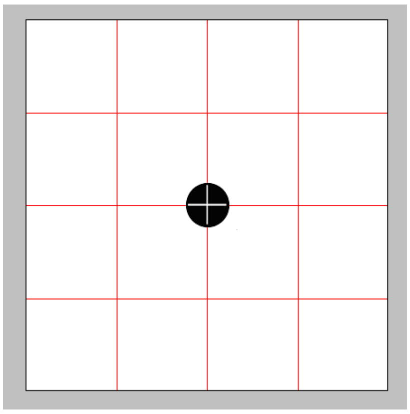
7-2. semi-automatic calibration: Move robot, align the P[0], P[1], P[2] points in the field of view with the marked points on the calibration board, as shown in the following figure.



Click **One-key Calbrt** to complete the 9-point calibration.

Note: The robot calculates the 9 points on the pallet based on the end of the robot, while the camera is installed on the J2 axis of the robot. Therefore, if three reference points are calibrated according to the grid in the field of view, the actual operation of the robot may not be based on the 9 grid points in the camera's field of view, but may be a parallelogram, which may exceed the field of view. Therefore, when calibrating the three reference points, please try to calibrate them in the middle of the field of view.

7-3. Automatic calibration: Move the robot, align the center of the field of view to the marker point on the calibration board, enter the length of the field of view grid corresponding to the length of the robot coordinate system (roughly enough to ensure that the 9 automatically generated points do not exceed the field of view).

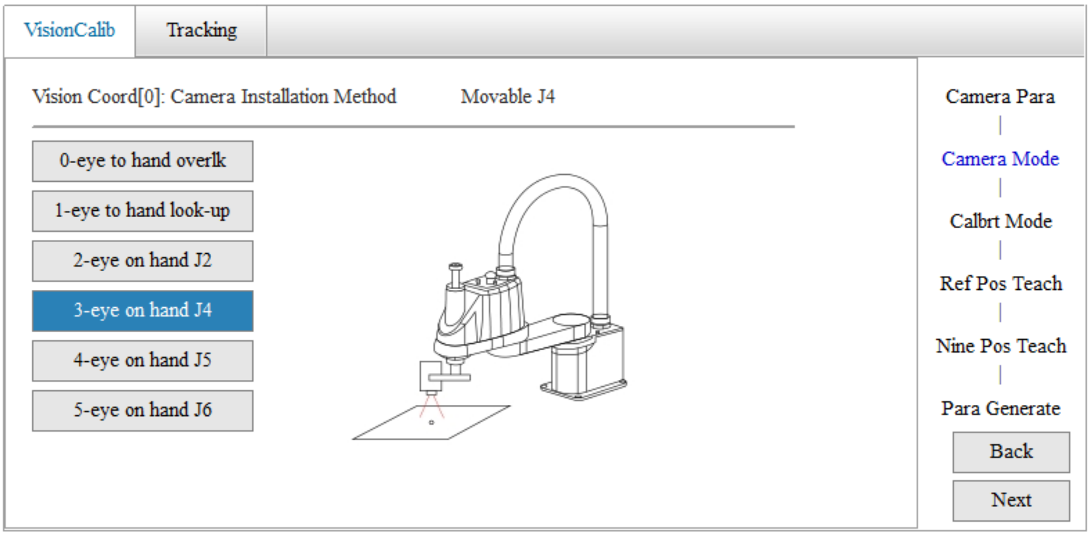


Click **One-key Calbrt** to complete the 9-point calibration.

Step 8-10: Same as Step 8-10 in 6.4.4 Eye-to-Hand Overlook Calibration.

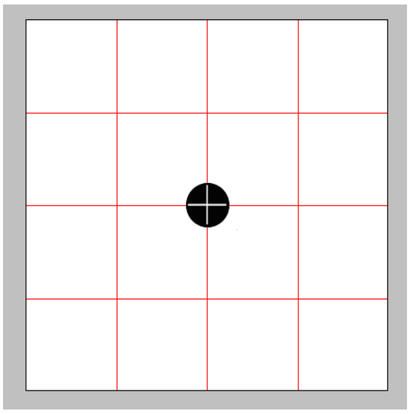
6.2.7 Eye-on-Hand J4 Calibration

The camera is mounted on the J4 axis, looking downwards, as shown in the following figure.



6.2.7.1 Mounting Condition

The camera is mounted on the J4 axis. Keep the camera plane parallel to the horizontal plane without calibration fixtures, as shown in the following figure. The black marker point in the figure represents the marker point on the calibration board, the red lines represent the grid lines in the camera's field of view, and the intersection of the grid lines is the required marker point.

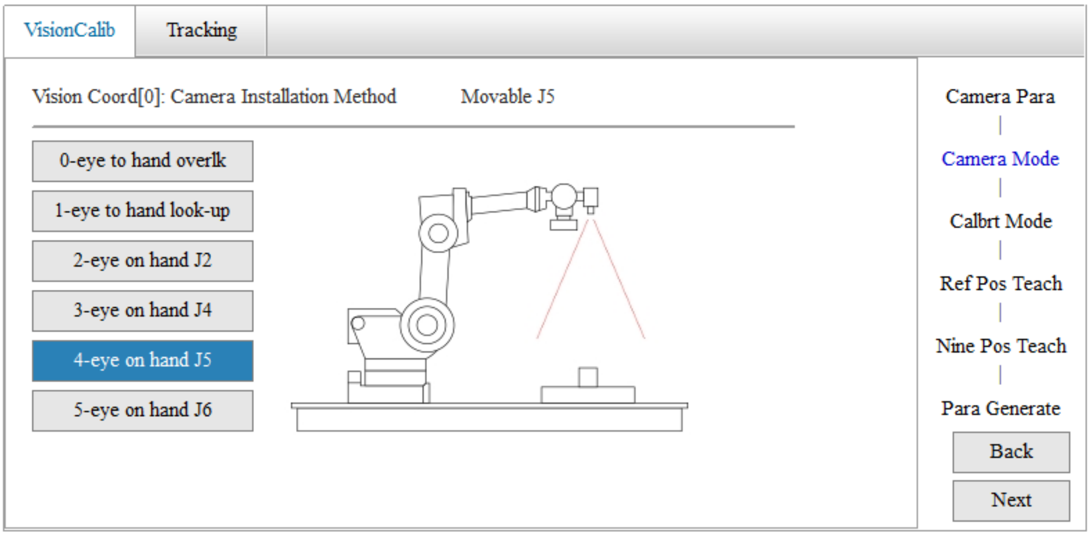


6.2.7.2 Calibration Procedure

Same as the calibration steps in 6.4.6 Eye-on-Hand J2 Calibration.

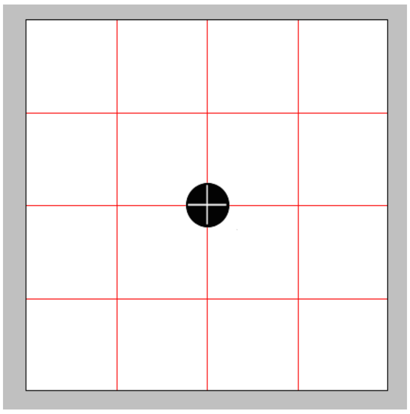
6.2.8 Eye-on-Hand J5 Calibration

The camera is mounted on the J5 axis, looking downwards, as shown in the following figure.



6.2.8.1 Mounting Condition

The camera is mounted on the J5 axis. Keep the camera plane parallel to the horizontal plane without calibration fixtures, as shown in the following figure. The black marker point in the figure represents the marker point on the calibration board, the red lines represent the grid lines in the camera's field of view, and the intersection of the grid lines is the required marker point.



6.2.8.2 Calibration Procedure

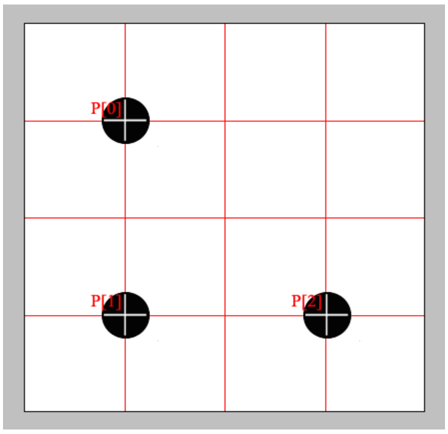
Step 1-5: Same as steps 1-5 in 6.4.4 Eye-to-Hand Overlook Calibration. Complete setting of basic camera parameters, camera mounting mode, and calibration mode.

6. Move the robot and adjust the camera pose to align with the center marker point to obtain three different points as reference points, and select the associated user coordinate system. For the selection of reference points and user coordinate system, see 6.4.4 Eye-to-Hand Overlook Calibration.

7. Click **Next** to go to the 9-point calibration interface.

7-1. Manual calibration: Move the robot, align the nine marker points in the field of view with the marker points on the calibration board to obtain the robot coordinates and pixel coordinates respectively. The camera coordinates can be automatically obtained or entered manually.

7-2. semi-automatic calibration: Move robot, align the P[0], P[1], P[2] points in the field of view with the marked points on the calibration board, as shown in the following figure.

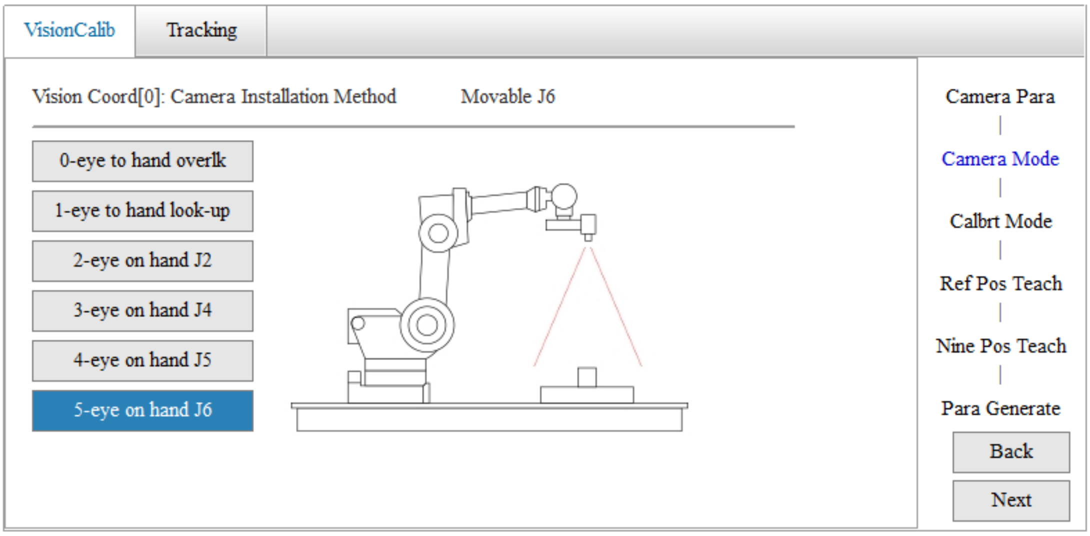


Click **One-key Calbrt** to complete the 9-point calibration.

7-3. Automatic calibration: Move the robot, align the center of the field of view to the marker point on the calibration board, enter the length of the field of view grid corresponding to the length of the robot coordinate system (roughly enough to ensure that the 9 automatically generated points do not exceed the field of view).

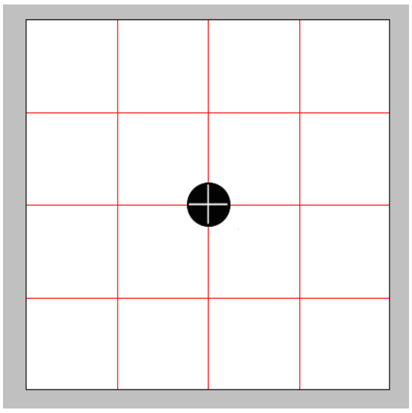
6.2.9 Eye-on-Hand J6 Calibration

Moving the J6 axis calibration: Refers to the camera mounted overhead on the robot J6 axis arm, as shown in the following figure:



6.2.9.1 Mounting Condition

The camera is mounted on the J6 axis. Keep the camera plane parallel to the horizontal plane without calibration fixtures, as shown in the following figure. The black marker point in the figure represents the marker point on the calibration board, the red lines represent the grid lines in the camera's field of view, and the intersection of the grid lines is the required marker point.

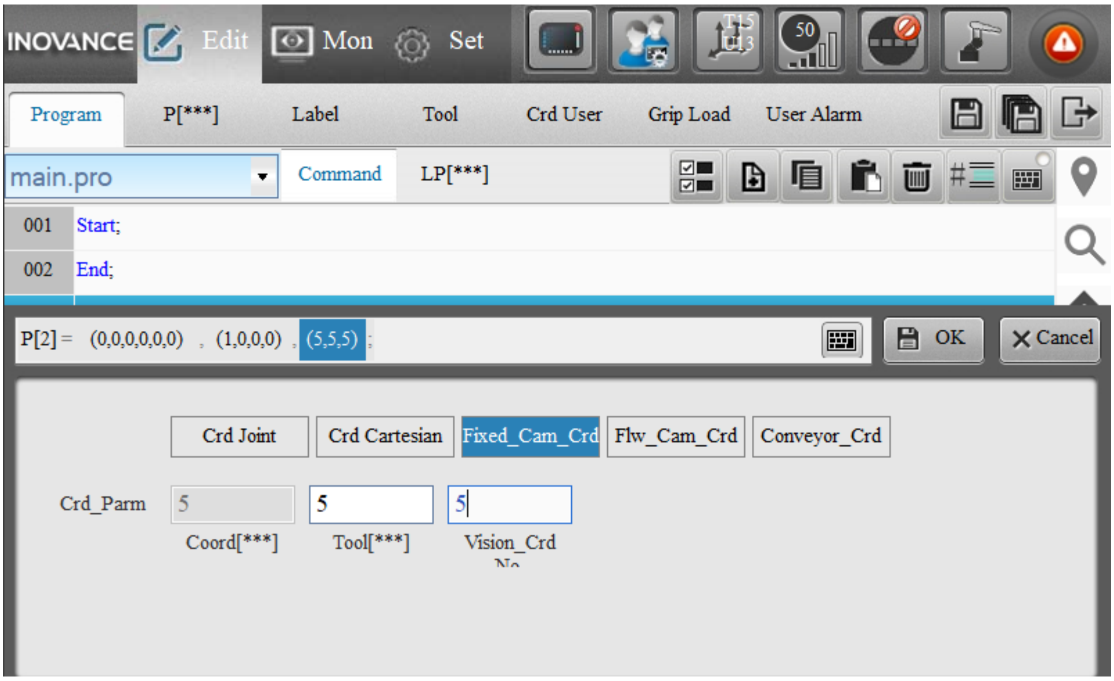


6.2.9.2 Calibration Procedure

Same as the calibration steps in 6.4.8 Eye-on-Hand J5 Calibration.

6.2.10 Calibration Result Verification

When vision calibration is carried out with a calibration fixture, if you want to use the calibration fixture as verification tools, go to **Edit** > **Tool** > **Coordinate** > **Direct Method**.
Create the tool coordinate system by entering the last recorded two deviations in the X and Y directions of the calibration fixture into X and Y of the tool coordinate system.
An example of a complete vision calibration programming is given below.



The coordinates of P[2] are pixel coordinates in the vision coordinate system. Regarding the coordinate parameter (5,5,5), the first "5" indicates the fixed camera coordinate system number, the second "5" indicates the tool coordinate system number, respectively.

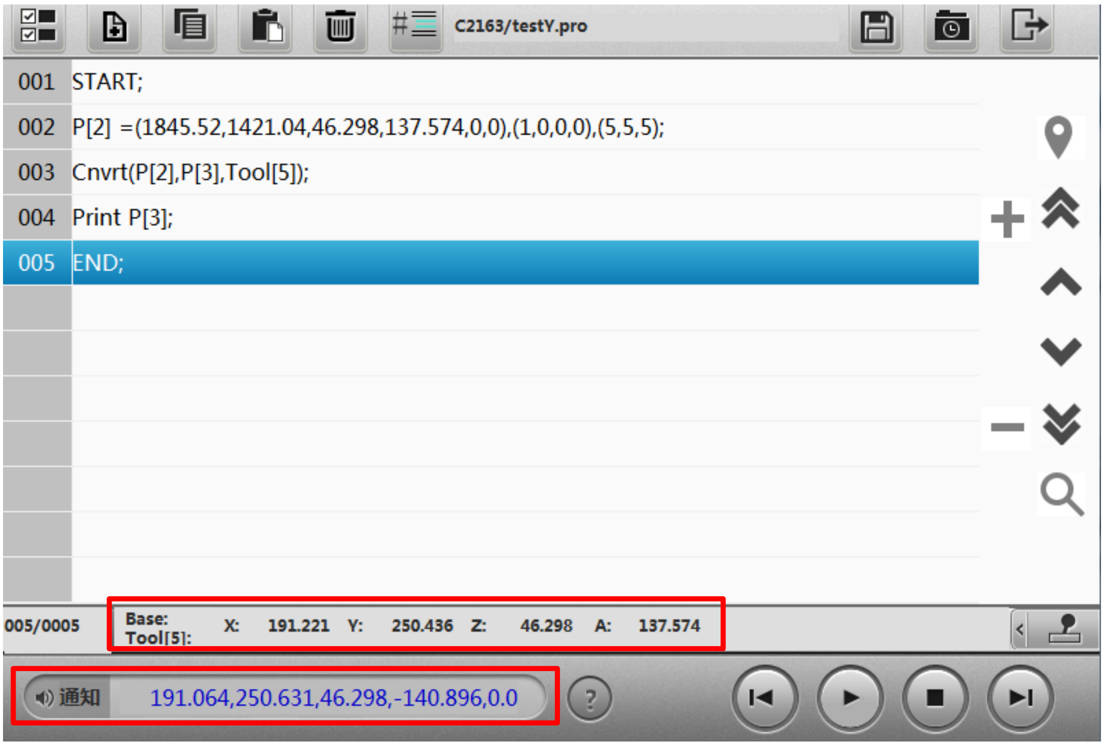
Note: For the 6-axis robots, in case of verification of eye-to-hand calibration, the Z value of the

P[2] point Z value is set to 0, and in case of verification of eye-on-hand calibration, the Z value is the height relative to the user coordinate system.

The third “5” indicates the vision coordinate system number.(1,0,0,0) is the arm parameters of the robot at the position where photo is taken.

The Cnvrt instruction is used to convert the P[2] point to a point in the tool coordinate system and store it in P[3].

If you choose "Eye-to-Hand" calibration, do not check **Visual reference point**.



The final results are displayed in the notification bar and are compared to the calibration points to determine the accuracy of the calibration. The A value -140.896 of the converted tool coordinates is inconsistent with value A 137.574 of the tool coordinates of the robot where the photo is taken, due to the inconsistency in direction of the vision coordinate system, direction of the robot coordinate system, and direction of the tool.

A value represents the rotation angle of the object in the vision coordinate system, and is converted to the angle in the tool coordinate system as A1, then

A1 = A +A’, A’ indicates the angle conversion difference

A2 = Angular difference between the direction of the robot coordinate system and the direction of the camera coordinate system

A3 = Angular difference between the direction of the robot coordinate system and the gripper direction or load gripping direction

A4 = Angle at the position where photo is taken in the tool coordinate system

A4 = <u>A + A’</u>+ A2 + A3 or A4 = A1 + A2 + A3

A4 = 137.574, A = 137.574, A1 = -140.896, then it is calculated:

A2 + A3 = A4-A1 = 137.574-(-140.896) = 278.47 or

A’ + A2 + A3 = A4-a = 137.574-137.574 = 0

Therefore, when a pixel coordinate A is given, the coordinates of the robot at the position where the photo is taken can be calculated as: A4=A+A '+A2+A3=A+0.

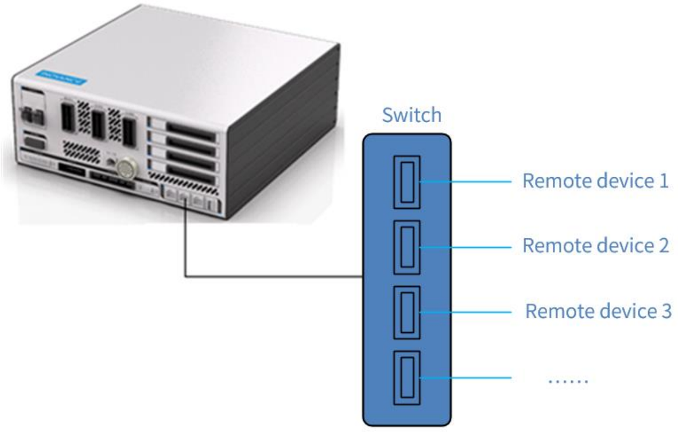
Alternatively, you can use the conversion instruction CNVRT to obtain the corresponding value A1 in the tool coordinate system from the pixel coordinate A:

A4= A1 + A2 +A3 = 278.47 + A1

When you choose “Eye-on-hand” calibration, please check **Visual reference point** and enter the photo taking point P[3].\

7 Others

7.1 TCP multi-port connectivity



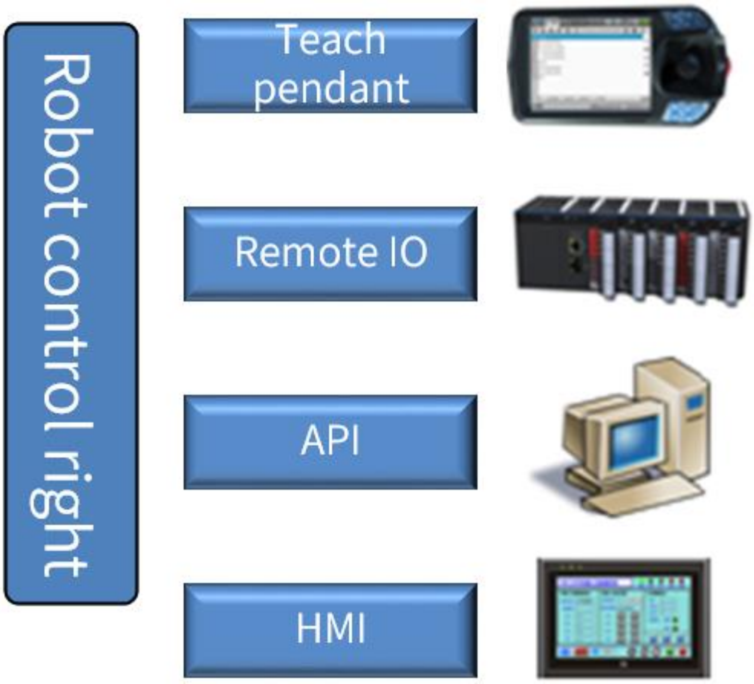
Like ordinary Ethernet devices, the Ethernet port of the controller can communicate with multiple external devices that support TCP protocol simultaneously through a switch.

For example, one PC can monitor data through API while another visual device interacts with the controller.

7.2 Permission Management

7.2.1 Robot Control Permissions

There are multiple devices that can control robots, but the robot controller can only be controlled by one device at a time.



Control permissions are a prerequisite for the device to take control of the robot. Only after obtaining control permissions can the device control the robot, otherwise it can only read and monitor the robot status.

In the teach pendant, you can manage the robot control permissions. Only editor, manager and factory users can edit the robot control permission and the robot must be in a non-motion, non-debug, non-play mode.

The default robot control permission is the teach pendant (i.e., InoTeachPad). To control the robot by other devices (InoRobShop, Remote Ethernet devices, Remote I/O devices, Modbus devices), you need switch the control in the teach pendant.

Note:

The control switch does not require a reboot of the robot controller.

The emergency stop switch of the teach pendant is always in effect, regardless of the type of device to which the control permission is assigned.

a) Teach pendant with control

The teach pendant gets control of the robot when the control permission is granted to the teach pendant. Other devices can only read or observe parameters, and cannot modify parameters or operate the robot.

b) InoRobShop with control

When the control permission is granted to InoRobShop, you can operate the robot via InoRobShop. Other devices can only read or observe parameters, and cannot modify parameters or operate the robot.

c) Remote Ethernet device with control

The remote Ethernet device gets control of the robot when the control permission is granted to the remote Ethernet device. Other devices can only read or observe parameters, and cannot modify parameters or operate the robot.

Up to four remote Ethernet devices can be connected to the robot at the same time. When control is switched from another device to an Ethernet device for the first time, the Ethernet device also needs to apply for the control permission through instructions.

Upon successful application of the control permission, the Ethernet device gets control of the robot. The other three devices do not have control permission and can read parameter and monitor robot status. However, these three devices can apply for mandatory control through instructions. Upon successful application, the device that has the control permission will be deprived of the control. When the control permission is granted to the Ethernet device, the default control is given to the first Ethernet device each time the controller is turned on.

d) Remote I/O device with control

The remote I/O device gets control of the robot when the control permission is granted to the remote Ethernet device. Other devices can only read or observe parameters, and cannot modify parameters or operate the robot.

e) Remote Modbus device with control

The remote Modbus device gets control of the robot when the control permission is granted to the remote Modbus device. Other devices can only read or observe parameters, and cannot modify parameters or operate the robot.

7.2.2 IRLink Configuration Permissions

The IRLink module can be configured either through InoRobShop or teach pendant.

Only when the IRLink module is not configured via InoRobShop can you configure it via the teach pendant.

When the IRLink module is configured via InoRobShop, the IRLink module can no longer be configured via the teach pendant. To add another new IRLink module, you can create a new IRLink module via InoRobShop, or clear the previous configuration using the Clear PLC-CFG function. In either case, a system restart is required after the configuration change.

7.2.3 I/O Control Permissions

The I/O control here refers to the control of the output ports (Out, DA, etc.), not the input ports.

Control Permission	Description
RC_STATIC	It indicates occupation by the RC system. Go to **External** > **I/O Mapping** and bind the outputs to system functions. At this point, the output signal is only function dependent. Signal state cannot be changed manually.
RC_ACTIVE	It indicates normal RC control state. Out or DA in this state can be controlled normally.
PLC_ACTIVE	Indicates control by PLC. Out or DA can be controlled only by PLC software such as InoRobShop.

When IRLink configuration is made via InoRobShop, the control of the first 16 DOs (two 0808 or one 0016) of the module is granted to RC by default. The control of the subsequently configured DOs is granted to PLC by default. By default, the control of the AOs is granted to PLC by default.

With the InoRobshop, you can switch the control between RC and PLC. The change takes effect without the need to restart the controller.

When an IRLink module is added via the teach pendant, the I/O control defaults to RC_ACTIVE and cannot be modified to PLC via InoTeachPad. However, you can modify the control to RC_STATIC by associating a certain function to a DO in **External** > **I/O-Mapping**.

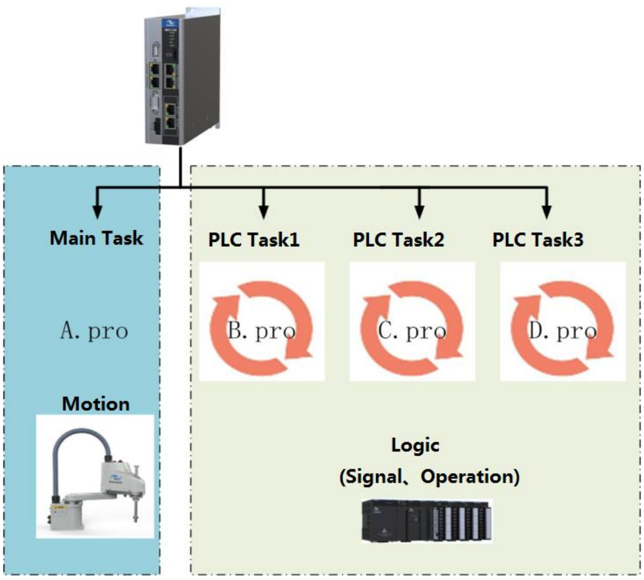
7.2.4 Modbus Configuration Permissions

The initial Modbus configuration permissions are pending, that is, the Modbus configuration can be made either via InoRobShop or via the teach pendant. However, when the Modbus configuration permissions are granted to InoRobShop, you cannot configure Modbus via the teach pendant. That is,

- ModbusRTU cannot be configured via the teach pendant if the ModbusRTU Master or Slave is configured via InoRobShop. When the configuration in InoRobShop is canceled, ModbusRTU can be configured via the teach pendant.
- ModbusTCP cannot be configured via the teach pendant if the ModbusRTU Slave is configured via InoRobShop, and can be configured via the teach pendant if the configuration in InoRobShop is canceled.

7.3 Multitasking

It allows multiple tasks to run simultaneously. The main task is responsible for motion, and other PLC tasks perform logical operations.



7.3.1 Task Description

	Main Task	Static Task	Dynamic Task	xqt Task

<table>
<tr><td>Type</td><td>Main.pro, which is always active and is mostly responsible for robot motion. Controlled by Start and Stop instructions.</td><td>The static tasks are started immediately upon power-on of the robot and run until the stop conditions are met.</td><td>The static tasks are not started immediately upon power-on of the robot and are controlled by Start and Stop instructions along with the main task.</td><td>The xqt tasks are controlled separately by the instructions Xqt, Halt, and Resume.</td></tr>
<tr><td>Recompile the task</td><td>1) Restart the controller
2) Re-save the project configuration information
3) Switch edit mode to debug mode
4) Save all settings</td><td>1) Restart the controller
2) Re-save the project configuration information</td><td>1) Restart the controller
2) Re-save the project configuration information
3) Switch edit mode to debug mode
4) Save all settings</td><td>1) Restart the controller
2) Re-save the project configuration information
3) Switch edit mode to debug mode
4) Save all settings</td></tr>
<tr><td rowspan="2">Activate the task</td><td>1) The task is automatically activated after the project configuration information is re-saved</td><td>1) The task is automatically activated after the project configuration information is re-saved
2) Activate or deactivate the static task via the task commissioning panel</td><td>1) The task is automatically activated after the project configuration information is re-saved
2) Activate or deactivate the task via the task commissioning panel</td><td>1) Activate the task by xqt instruction and deactivate the task by quit instruction
2) The xqt task is deactivated in the following situations:
①The projects is recompiled
②Any task returns to the start line
③Control switch occurs</td></tr>
<tr><td colspan="4">Conclusion:
1. There is no coupling relationship between activation and deactivation of each task.</td></tr>
<tr><td rowspan="2">Start/Stop the task</td><td>1) Start the task via the start button;</td><td>1) The task starts automatically after being activated;
2) If a static task has been configured before restart, the task runs automatically after restart;
3) InoTeachPad: When the task is active, stop the task by deactivating the task in the task management interface;</td><td>1) The dynamic task starts with the start of the main task.</td><td>1) The xqt task is activated and started by xqt instruction;
2) After stopping the task by halt instruction, you can start it again by resume instruction;</td></tr>
<tr><td>1) Stop the task via the stop button;
2) The task stops when an alarm occurs;</td><td>1. Stop the task via the start button in the task management interface;
2. The task stops when the corresponding static task channel encounters an error;
3. PC platform: When the task is started, stop the task via the stop button in the task management interface;
4. InoTeachPad: When the task is inactive, stop the task via the stop button in the task management interface;
Note: When the static task is reactivated, the static task starts again.</td><td>1. The dynamic task stops with the stop of the main task;
2. PC platform: When the main task is inactive and the dynamic task is started, stop the dynamic task via the stop button in the task management interface;</td><td>1. The xqt task stops with the stop of the main task;
2. When the xqt task is activated, stop it by halt instruction;</td></tr>
</table>

	Conclusion: 1. Tasks can only be started and stopped in an active state; 2. Static tasks are not affected by the state of the main task;			
Resume the task after alarm	Clear the alarm and restart the task	Clear the alarm, reactivate the static task, or synchronize the project	Clear the alarm and restart the task	Clear the alarm and restart the task
Support for single-step debugging	Available	No	Available	No
	Conclusion: 1. When you click a line of instruction, the main task and dynamic task advance one line each; 2. If the current line of the main task is not completed, when you click another line of instruction, the main task will continue to execute the current line, and the dynamic task will advance one more line;			
Support for return to the start line	Yes, return to the start line for a single task or for all tasks	Yes, return to the start line for a single task	Yes, return to the start line for a single task or for all tasks	No
	Conclusion: 1. The task can return to the start line only when the task is in stop or ready state.			

7.3.2 Use of Multitasking

Common usage:

The main task is responsible for motion, and the PLC tasks perform logical operations. The main task changes motion in real time according to data from the processing of the PLC tasks.

Note:

1. There should be no motion or motion parameter type instructions in tasks 1, 2, and 3:

   Movj, Movl, Movc, Jump, JumpL, Home, Velset value/OFF, RefSys, LockScrew, UnLockScrew, ArmChange, SlewMode, MovToPut, MovToGet, MovFromPut, MovFromGet, EoffsOn, EoffsOff, LoadScrewParm, CheckLock, UnLockScrew, CheckUnLock, ScrewStop, SetAccRamp, SetFlyMode, SetFlyPress, AvgCurLmt, MaxTrqLmt, LatchEnable, ClearLatchPos, GetLatchPos, SetCollMode, SetAxiscollMode, SetAxisCollLevel, GripLoad, SetGripLoadMass, SetGripLoadCog, SetGripLoadOrient, SetLoadInertia, SetToolMass, SetToolCog, SetToolOrient, SetToolInertia, SetSysToolNo

   If the above motion instructions are introduced during programming, they will be skipped.

2. From the perspective of execution efficiency, task 0 (main task) has the highest execution efficiency, while tasks 2, 3, and 4 have relatively low execution efficiency. This is reflected in the fact that when multiple tasks perform calculations or process I/O separately, task 0 performs faster.

3. The PLC cyclically executes tasks with a cycle time of about 10ms.

Case 1:

Selective motion based on communication data

Scenarios:

The controller performs two tasks simultaneously:

1) Constantly communicates with the external device and sets the flag to the corresponding value depending on whether the value of the data received from the external device is 201 or not;

2) Selectively performs Movj P[1] and Movj P[0] motion based on the flag value.

Design:

Main task: Continuously determines the value of a global numerical variable (such as B4 below) within the loop, and executes the corresponding motion based on the value.

PLC task: Constantly communicates with the outside world and sets B4 as the corresponding value based on the read results.

Main task: Main.pro

```
START;
    Movj P[0],V[30],Z[0],Tool[0];
    L[0]:
    If   B[4] == 2
        Movj P[1],V[30],Z[0],Tool[0];
    EndIf;
    If   B[4] == 4
        Movj P[2],V[30],Z[0],Tool[0];
    EndIf;
    Goto L[0];
END;
```

PLC task: Communication.pro

```
START;
    String recv= "000000";
    While LB[0] <> 1
        Open Socket("10.44.97.53",2000,3000, LB[0]);
    EndWhile;
    L[1]:                    ##PLC task characteristics, self-cycling, this line can also be removed
    Send Port[3000],"hello",String;
    L[0]:
    Get Port[3000],T[0],Goto L[0];
    recv = GetPortbuf(0,100,3000);
    If   StrToR(recv) == 201
        B[4] =4;
    Else
        B[4] =2;
    EndIf;
    Goto L[1];          ##PLC task characteristics, self-cycling, this line can also be removed
END;
```

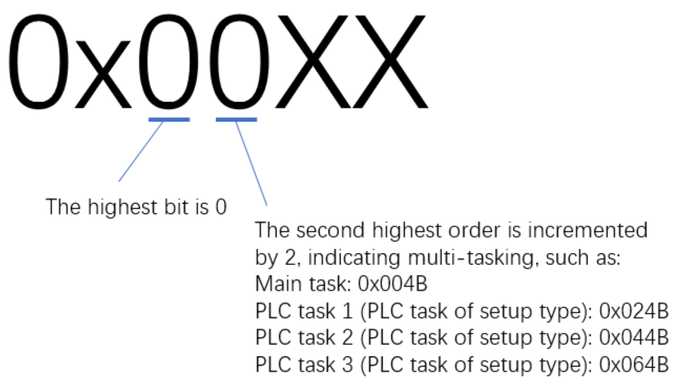
For the above PLC task,

1. If set to a static task, it starts immediately upon power-on of the robot or re-save of the project configuration, regardless of the start and stop instructions.
2. If set to dynamic task, it start and stops as the main task starts and stops.

3. If the task communicating with the outside world is Task 3, the following should be added at the beginning of the program of Task 0: Xqt 3, "Communicatin.pro";

7.3.3 Multitasking Alarms

In Appendix 1, only the main task alarms are listed. For multitasking alarms, the following rules apply:



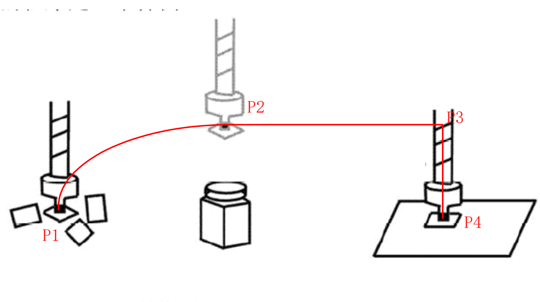
For example:

Robot system alarm 0x0230 occurs. The corresponding alarm code cannot be found in the alarm list in Appendix 1, so it is considered a multitasking alarm. In the alarm list, it can be found that the alarm code 0x0030 is "Decoding error during operation". Due to the high order being 2, it is inferred that 0x0230 is the alarm for task 1, and the alarm message is "Decoding error during operation".

7.4 Flying Trigger

The flying trigger function allows the robot to take photos and adjust the pose of the part accordingly without stopping the robot or part during movement of the robot from the part pickup point to the part discharge point. This function reduces the cycle time.

As shown in the image below, P1 is the part pickup point, P2 is the photo taking point, P3 is the transition point and P4 is the part discharge point. Normally, the robot needs to pause at P1, P2, P3 and P4. The robot does not pause at P2. Instead, when the robot is predicted to reach P2 during the movement from P1 to P3, the controller controls DO for output, triggering the camera to take photos and the servo to latch the position. Then the controller determines the position and orientation at which the part will be placed at P4 based on the vision processing results and the latched position.



The flying trigger function is achieved by a combination of three functions: position latching, motion I/O, and motion without waiting.

For programming of the flying trigger function, see the Programming Guide. The trigger signals available include Out[14] and Out[15].

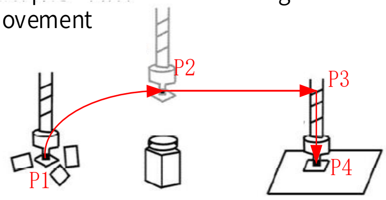
Note:

1) For IRCB500, IRCB300 series controllers, the position latch function is supported. Users only need to connect the corresponding I/O to the camera as a trigger for taking photos.
2) The position latch function is not supported for IRCB10 series controllers.
3) The IRCB300 series controllers support position latching triggered by 1 signal (fixed to Out15). To avoid the problem of unavailability of position latching function due to damage, IRCB500 series controllers support position latching triggered by 2 signals: Out14 or Out15, and the default is Out[14].

The following is a comparison between normal photo-taking and flying trigger photo-taking.

Common photo taking

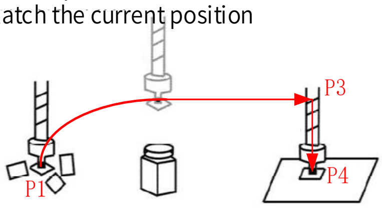
Stop at P2 and signal the camera to take photos before continuing the movement




```
START;
#Vision initialization
...
#Move to pickup point
Jump P[1],V[100],Z[0],LH[0],MH[0],RH[10];
Set Out[15],OFF;
#Motion transition point, output rising
edge when passing P2, preprocess
subsequent instructions
Jump P[2],V[100],Fine,LH[10],MH[0],RH[0];
Set Out[15],ON;
Movj P[3],V[100],Z[5],LH[0],MH[0],RH[0];
#Acquire vision data and store it to P[10]
...
#Correct the coordinates of the discharge
point according to P[20] and P[10]
P[4]=...
#Move to discharge point
Movj P[4],V[100],Z[0];
END;
```

Flying trigger

During movement, signal the camera to take a picture while and the servo to latch the current position




```
START;
#Enable latch function
LatchEnable ON;
ClearLatchPos;
#Vision initialization
...
#Move to pickup point
Jump P[1],V[100],Z[0],LH[0],MH[0],RH[10];
#Motion transition point, output rising edge when passing P2,
preprocess subsequent instructions
Jump
P[3],V[100],Z[5],NWait,Out(15,OFF,D[0]),Out(15,ON,T[50]),LH[10],MH[0]
,RH[0];
B1 =0;
#Wait for latch to be triggered by rising edge
While B1 == 0
B1 =GetLatchPos(P[20],2,0,0);#Save latched position to P[20]
EndWhile;
Print P[10];
#Acquire vision data and save it to P[10]
...
#Correct the coordinates of the discharge point according to P[20] and
P[10]
P[4]=...
#Move to discharge point
Movj P[4],V[100],Z[0];
END;
```

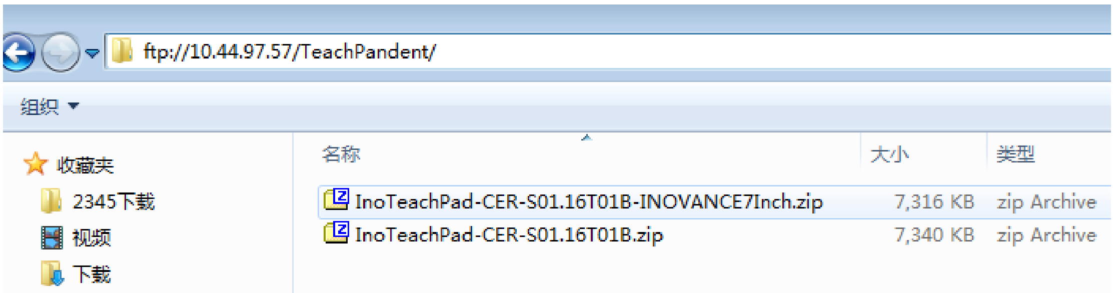
7.5 Teach Pendant Synchronization

This feature is for handheld teach pendant only. With this feature, you can upgrade the handheld teach pendant to the same version as the controller.

If the version of the teach pendant is different from the version of the controller (e.g. S03.21 compared to S03.20), the teach pendant will automatically pop up the "Synchronization or not" prompt after being powering on.
In addition, the Sync button will also appear on the **Monitor** > **Version** interface. You can click this button to synchronize the version.

Limitations:
After the controller has been flashed through the SD card, the synchronization function is not supported. In this case, additional manual operations are required to support the synchronization function. If the controller is flashed via network or updated through the update button on the teach pendant, the synchronization function is not affected.

Manual operations:
Place two teach pendant packages (standard and simple versions) that are consistent with the controller version into the "TeachPandent" directory of SD card storing the controller program. (Use FTP to place the packages.)
As an example, assuming that the controller now has an IP of 10.44.97.57, do as follows on the PC:



Note:
1. Place the packages strictly as required. Do not place unqualified or excessive packages.
2. It is not feasible to directly remove the SD card from the controller and place it in the card reader on the PC, as the SD card format is not supported by the PC.

7.6 Retentive Memory

This feature allows the system to save the relevant variable values immediately when the system is shut down normally or when there is a sudden power failure during operation, and the supported variable types are as follows:
1) Global translation variables (PR);
2) Global numerical variables (B/R/D);
3) Global string variables (Str).

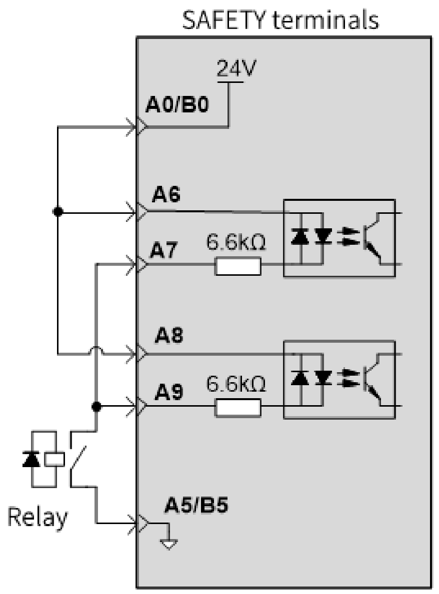
Note:
1) The retentive memory requires hardware support.
2) The global translation variables (PR), global numerical variables (B/R/D), and global string variables (Str) are not saved immediately at backup, and are only saved when power is lost.
3) The global string variables (Str) can be saved to a file when they are modified in the monitoring interface.
4) The global position variables (GP) can be saved to a file by clicking the Save button in the monitoring list.

7.7 Safety door

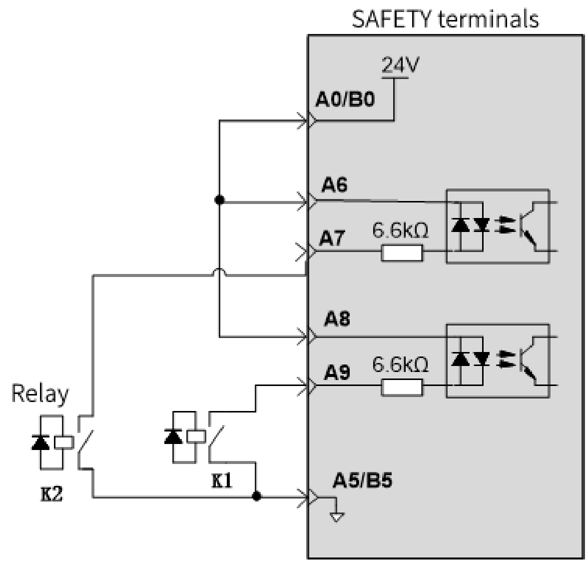
The safety door function is supported by the IRCB300 series controllers, the IRCB100-6AT series controllers, and the IRCB500 series controllers.
It is not supported by the IRCB10 series controllers.

Activation of the safety door:

1. Hardware wiring: The safety door supports dual-channel control. Connect the safety door lines to the safety door interfaces designated by the system I/Os (A6 and A7 for one channel; A8 and A9 for another). When only one channel is used, set the wiring of the other channel to the normally closed state. The following figure shows the wiring of safety door when in-cabinet power supply is used.



Controlling two channels with a shared relay



Controlling two channels with a relay for each channel

For more information, see:

IRCB300 Series 4-Axis Robot Controller User Guide
IRCB300 Series 6-Axis Robot Controller User Guide
IRCB500 Series 4-Axis Robot Controller User Guide
IRCB500 Series 6-Axis Robot Controller User Guide

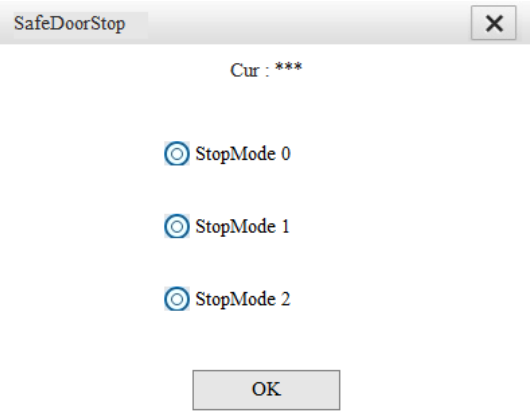
2. In the software, enable the safety door and select the safe stop mode.

Safety door switch:

For the IRCB500 series controllers, there is no safety door switch and the safety door feature is enabled by default in order to meet the safety requirements.

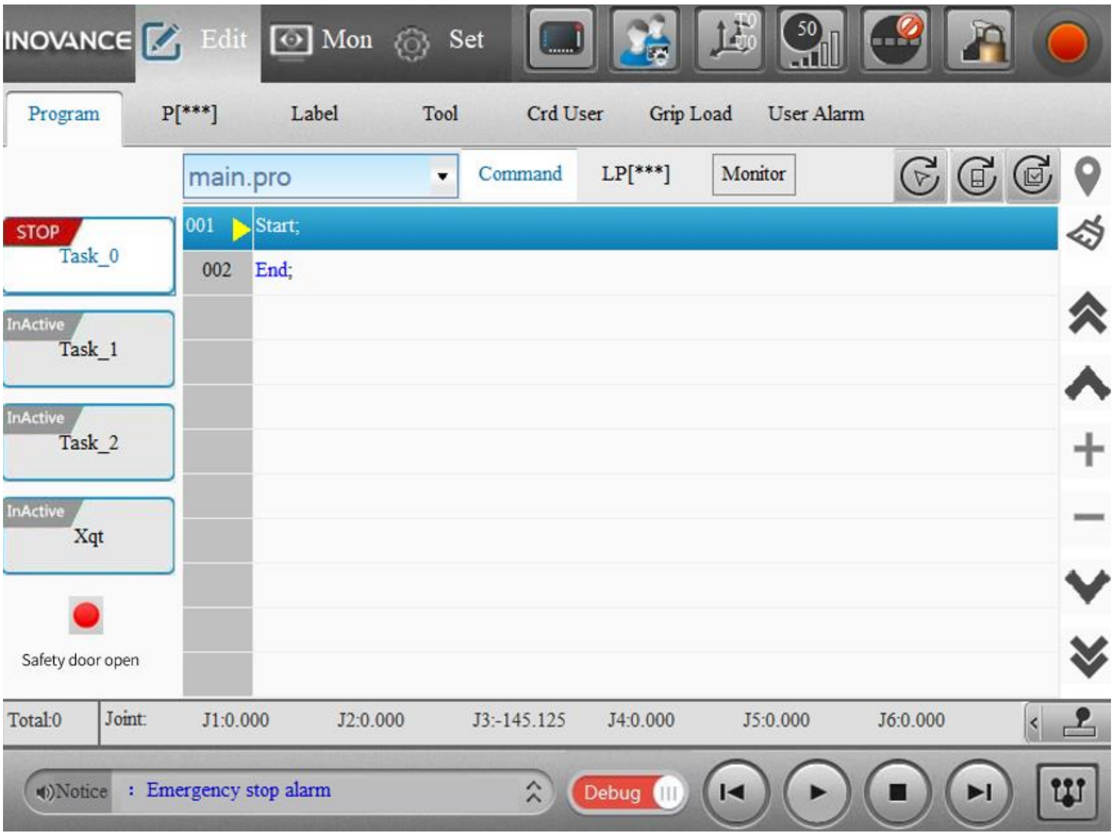
For the SCARA robots, the safety door feature is disabled by default; and for the 6-axis robots, the safety door feature is enabled by default.

Stop mode:



Use of safety door:

When the safety door feature is activated, the safety door signal will be detected in the play mode. In the play mode, when the safety door is opened, the program is suspended or stopped (depending on the user's configuration) and a prompt is displayed on the interface.



Note: For the IRCB500 series controllers, the system also generates an alarm "Safety door triggered" when the stop mode is set to StopMode 0 or StopMode 1.

7.8 Current Protection

(1) Application scenario:
By default, the motion parameters for each model are factory default. When the motion efficiency on site does not meet the requirements, the robot motion parameters can be adjusted. For example, you can increase the motion speed and acceleration appropriately in the **Set** interface.
When commissioning the motion parameters, pay attention to the average load rate and the current value.

- The robot generates warnings and alarms to protect motors and reducers when the rate is too high.
- When motion parameters are adjusted and no warnings or alarms are desired, you can appropriately increase the "global current limit coefficient" and "local current limit coefficient".

Note: The increase of the limit coefficient should be moderate, excessive increase will reduce the performance and life of the motor and reducer.

Description of the current protection parameters:

$$\frac{\text{Real time value of average load rate}}{\text{Average load rate protection threshold}*\text{Global average load rate limit coefficient}*\text{Local average load rate limit coefficient}} = \text{Average load rate}$$

Global average load rate limit coefficient: Go to **Set** > **Motion** > **AxisPara** > **AvrLoadLim**.
Local average load rate limit coefficient: Set the coefficient in the instruction AvgCurLmt.
Average load rate: Go to **Monitor** > **Protection**.

$$\frac{\text{Real-time current value}}{\text{Current protection threshold}*\text{Global current limit coefficient}*\text{Local current limit coefficient}} = \text{Current rate}$$

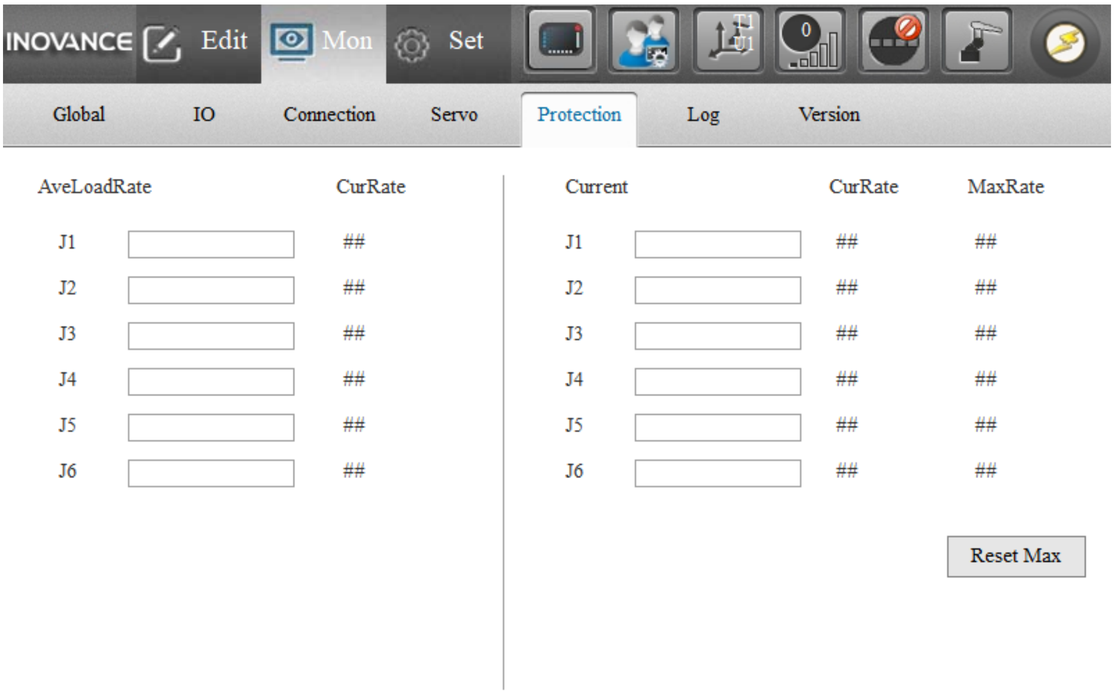
$$\frac{\text{Max current value}}{\text{Current protection threshold}*\text{Global current limit coefficient}*\text{Local current limit coefficient}} = \text{Max current rate}$$

Global current limit coefficient: Go to **Set** > **Motion** > **AxisPara** > **CurrentLim**.
Local current limit coefficient: Set the factor size in the instruction MaxTrqLmt.
Current rate: Go to **Monitor** > **Protection**.
Max current rate: Go to **Monitor** > **Protection**. Due to the rapid changes in the real-time value of the current during movement, it is inconvenient to view it. Therefore, a maximum current rate is

added, which is equivalent to the maximum value recorded in history, enabling easy observation.

(2) Monitoring and debugging method:

Go to **Monitor** > **Protection** to monitor the average load rate and current rate.



Note: For the current, in addition to the current rate, the maximum rate is also displayed to record the historical maximum current. It can be reset by the **Reset Max** button.

The rate is divided into the following three levels.

Level	Safe	Warning	Danger
Average load rate/set threshold	0 to 100%	100% to 110%	110% or higher

Level	Safe	Warning	Danger
Current/set threshold reference	0 to 110%	110% to 130%	130% or higher

Note:

- The above thresholds apply after the robot is warmed up.
- The current detection triggers a warning or alarm only when the threshold is exceeded for 30ms. When the instantaneous current is too high, an alarm will also be triggered, and the maximum current value may not exceed 130% at this time.
- The average load rate detection triggers a warning or alarm only when the threshold is exceeded for 30s.

Debugging principles:

- Efforts should be made to ensure that the average load rate and current rate are in a "safe" state during operation.
- When in the "warning" state, a system warning will be triggered, but the robot will not stop automatically. You can continue to use the robot without adjusting the motion parameters.
- When in a "danger" state, a system alarm will be triggered and the robot stops automatically.

In this case, you need to adjust the motion parameters to return them to the safe range.

- In particular, in applications where efficiency is required, if you want to adjust motion parameters while avoiding warnings or alarms of high average load rate or high current, you can appropriately increase the "global limit coefficient" and "local limit coefficient". It should be noted that in this case, the performance and lifespan of the motor and reducer are sacrificed, and it is only recommended to debug under the guidance of the manufacturer.
- Low ambient temperature may cause an average load rate alarm. Before using the robot, warm it up first.

(3) Set the limit coefficient:

You can set the global limit coefficient for all axes through the user interface, and set the local limit coefficient for an individual axis through the program.

Characteristics:

- Switch: Only when both the main switch and the sub-switch of a certain axis are turned on can the detection of that axis be enabled, thus continuous detection is made possible. If the main switch is turned off and the sub switch is turned on, the detection of that axis is still disabled
- Values: The global limit coefficient and the local limit coefficient act together, multiplying the two together acts on the threshold.

To set the global limit coefficient through the user interface, go to **Set** > **Motion** > **AxisPara** > **AvrLoadLim/CurrentLim**.

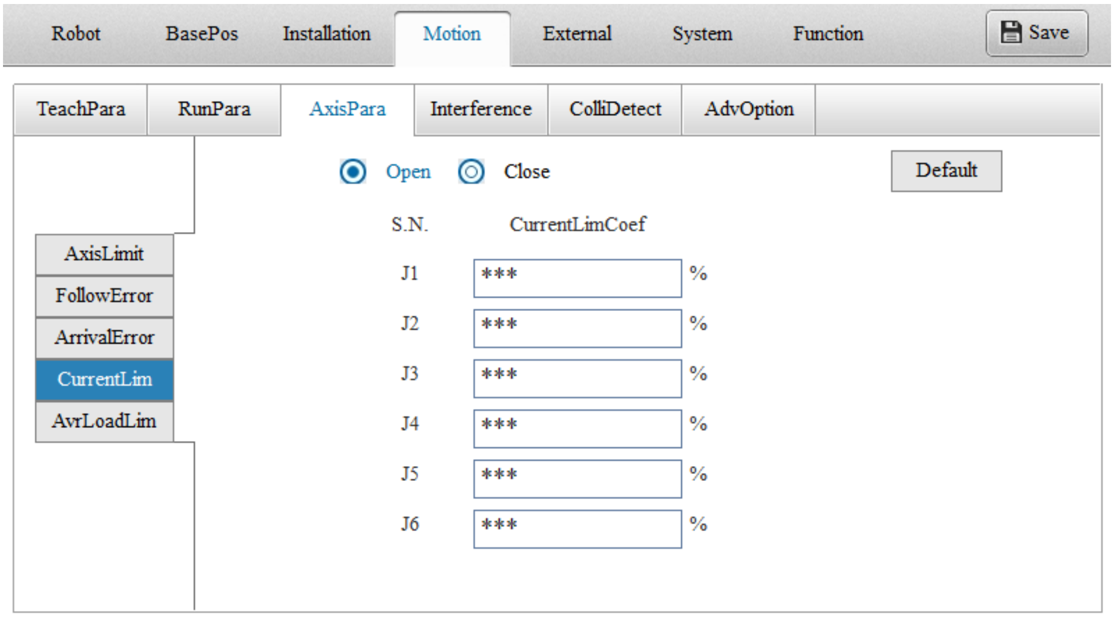
You can use the AvgCurLmt/MaxTrqLmt instruction to set the local limit coefficient.

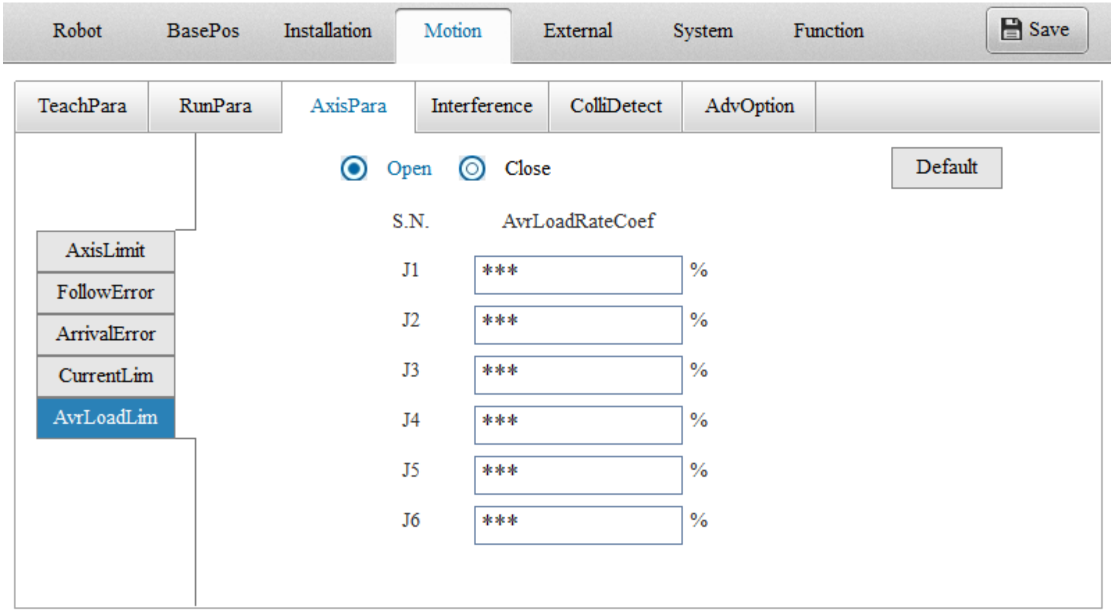
① Configuration through User Interface

The parameters related to the current protection can be configured in factory mode.

Detection switch: Once opened, motion detection of all axes is active; once closed, no more detection is performed. It is opened by default.

Setting value: Sets the global limit coefficient. In form of percentage, default 100%.





② **Configuration through Instructions**

The single axis can be dynamically configured by instruction.

You choose to turn off/on detection for a single axis and adjust the detection threshold in the robot program, which is flexible and convenient.

a) AvgCurLmt

Function: Configures the average load rate limit.

Description: Sets whether the average load rate detection is in effect and the average load rate limit parameters in the program. Applicable to a single axis or all axes.

Format: AvgCurLmt (Enable, AxisNo, ratio);

Parameters:

Enable: Specifies valid/invalid through integer data. 1 for valid, 0 for invalid.

AxisNo: Specifies the axis number using integer data. Special case: Specifies all axes by '0'.

ratio: Specifies a local limit coefficient for the single-axis average load rate using integer data, in percentage, range (1, 150). This parameter is invalid if Enable is set to 0.

Detection switch properties:

1. The main switch on the teach pendant defaults to Open. When the robot restarts, it restores to the default value.
2. The detection switch of each single axis in the instruction defaults to "Open” (True), and the local limit coefficient of single axis is set to 100 by default. Restore the default parameters in the following situations:

   (1) The robot restarts.

   (2) The program is executed from the beginning (stop and start again from the beginning).

   (3) Switch programs or exit programs (excluding Call).

   (4) Save the program.

   (5) Switch between Teach/Play mode.

b) MaxTrqLmt

Function: Configures the maximum torque limit (current limit).

Description: Sets whether the current detection is in effect and the current limit parameters in the program. Applicable to a single axis or all axes.

Format: MaxTrqLmt (Enable, AxisNo, ratio);

Parameters:

Enable: Specifies valid/invalid through integer data. 1 for valid, 0 for invalid.

AxisNo: Specifies the axis number using integer data. Special case: Specifies all axes by '0'.

ratio: Specifies a local limit coefficient for the single-axis current using integer data, in percentage, range (1, 150). This parameter is invalid if Enable is set to 0.

Detection switch properties:

1. The main switch on the teach pendant defaults to Open. When the robot restarts, it restores to the default value.
2. The detection switch of each single axis in the instruction defaults to "Open” (True), and the local limit coefficient of single axis is set to 100 by default. Restore the default parameters in the following situations:

   (1) The robot restarts.

   (2) The program is executed from the beginning (stop and start again from the beginning).

   (3) Switch programs or exit programs (excluding Call).

   (4) Save the program.

   (5) Switch between Teach/Play mode.

7.9 API

7.9.1 Description of API Call

Using the APIs, users can develop proprietary robotic system application software through programming languages such as VB, VC, C#. The following is an example of C/C++ application development in VS platform, introducing the implementation and invocation examples of basic functions.

a) Connecting/Disconnecting the robot

Call the IMC100_Init_ETH() function to connect the robot over the network, and call IMC100_Exit_ETH() to disconnect the robot.

Before calling any other API, you should first call IMC100_Init_ETH() once to ensure that the robot is connected each time you open the application. If the function returns a value other than zero, check that the robot control system starts up correctly and troubleshoot according to Appendix III.

IMC100_Exit_ETH () should be called after other APIs are called, and the robot will be disconnected after 0 is returned.

A code example for calling IMC100_Init_ETH() is shown below:

```
int ret = 0;
DWORD dwIP1 = 0xc0a81719; //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 shown below:

```
ret = IMC100_Exit_ETH(0)
if(ret < 0)
{
    //Add exception handlers here
    return;
}
```

b) Monitoring the robot status

The IMC100 robot provides hundreds of API function interfaces for monitoring robot status. This class of functions is not restricted by control permission or user level. Before calling this class of functions, make sure that the robot system has been started normally and the connection to the robot system is successful. This class of functions includes IMC100_Get_PosHere(), IMC100_Get_DINum(), IMC100_Get_StrPara(), IMC100_Get_P(), and so on.

A code example for calling IMC100_Get_PosHere() is shown below:

```
//Query the position value of the robot in the current coordinate system
int ret = 0;
int robotNo = 0;
int dinum = 0;
int dists = 0;
ROBOT_POS posTemp;
memset(&posTemp, 0, sizeof(posTemp));
ret = IMC100_Get_PosHere(&posTemp, robotNo);
if(ret < 0)
{
    //Add exception handlers here
 }
```

A code example for calling IMC100_Get_DI() is shown below:

```
//Query the DI0 status, dists being 1 means ON
ret = IMC100_Get_DI(dinum, &dists, robotNo);
if(ret < 0)
```

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

c) Obtaining the control permission

Call IMC100_AcqPermit() to obtain permission for robot control, and call IMC100_CurPermit() to query client that currently holds the permission.

Since one robot system can be connected to multiple Ethernet clients, a control permission must be obtained when one of the clients needs to control the robot. Before calling the function, make sure that the robot system has been started normally and the connection to the robot system is successful. Also, go to **Set** > **System** > **Others** > **Others** > **Control Device**, and select "Remote Ethernet".

A code example for calling IMC100_CurPermit() and IMC100_AcqPermit() is shown below:

```
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) //The current client device is not granted the permission
{
      Ret = IMC100_AcqPermit(1, robotNo); //Forced to get the permission, can get it normally when ower is 0
      if(ret < 0)
      {
            //Add exception handlers here
      }
}
```

d) User login

Call IMC100_ CurUserType() to query the current user level and call IMC100_ UserLogin() to log in to the system and calls IMC100_ UserLogout() to log out of the system.

Users of different levels can control and operate the robot to varying degrees and ranges.

Before calling the function, make sure that the robot system has been started normally and the connection to the robot system is successful. Go to **Set** > **System** > **Others** > **Others** > **Control Device**, and select "Remote Ethernet".

A code example for calling IMC100_CurUserType()， IMC100_UserLogin() and IMC100_UserLogout() is shown below:

```
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 as admin, password is the same
as the teach pendant password
if(ret < 0)
{
	//Add exception handlers here
}
ret = IMC100_UserLogout(robotNo);
if(ret < 0)
{
	//Add exception handlers here
}
```

e) Returning the robot to the origin

Call IMC100_DsMode() to turn on the data streaming mode and IMC100_Home() to control the robot to return to the origin.

Before calling the function, make sure that the robot system has been started normally and the connection to the robot system is successful, and the control permission is obtained.

A code example for calling IMC100_DsMode() and IMC100_Home() is shown below:

```
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 streaming mode is off
{
	int cmd = 1;
	ret = IMC100_DsMode(cmd, robotNo); //Turn on data streaming mode
	if(ret < 0)
	{
```

```
            //Add exception handlers here
        }
}
int num = 0;
ret = IMC100_Home(num, robotNo); //The robot returns to origin 0
if(ret < 0)
{
        //Add exception handlers here
}
```

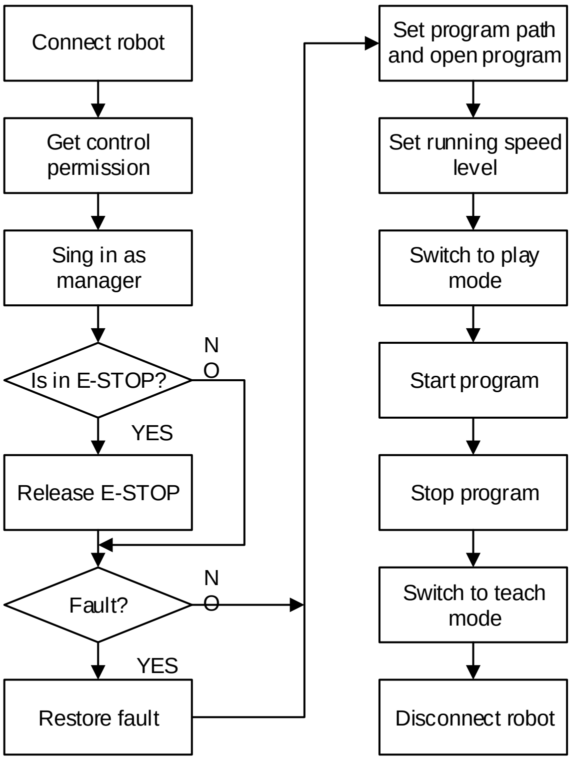
7.9.2 Typical Application Cases

The following is a complete typical application case to further illustrate the process of calling functions.

a) Running robot program through the remote Ethernet client

Make sure the target robot program can run normally before this operation.

Overall process:



The code example is as follows:

```
int ret = 0;
DWORD dwIP1 = 0xc0a81719; //IP: 192.168.23.25
int ipPort = 2222;
int timeOut = 5; //Communication timeout 5s
int robotNo = 0;
/*Connect to 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;
/*Obtain control permission*/
ret = IMC100_CurPermit(&ower, &IpAddr, & ipPort, robotNo);
if(ret < 0)
{
    //Add exception handlers here
}
if (ower! = 1) //The current client device is not granted the permission
{
    ret = IMC100_AcqPermit(1, robotNo); //Forced to get the permission, can get it normally when ower is 0
    if(ret < 0)
    {
        //Add exception handlers here
    }
}

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

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

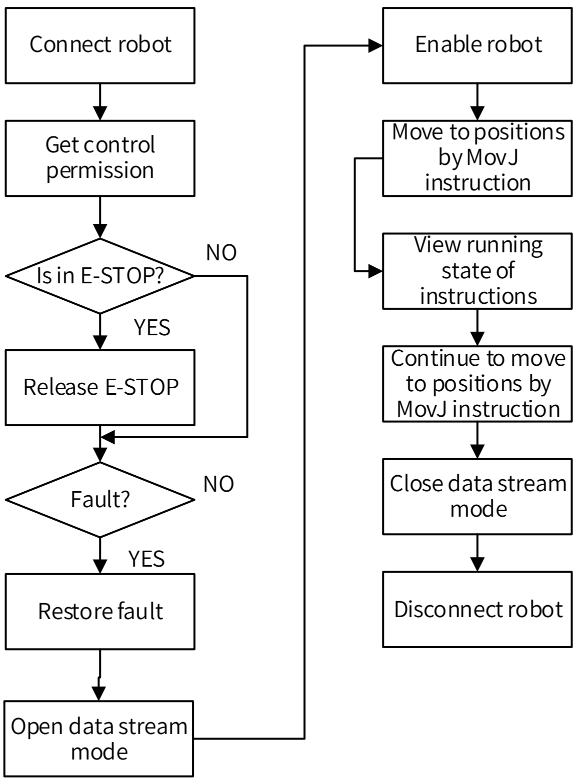
```
{
        int cmd = 0;
        /*Emergency stop released* /
        ret = IMC100_EmergStop(cmd, robotNo);
        if(ret < 0)      //Add exception handlers here
}

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

char path[128];
memcpy(path, "TeachProgram/Test.pro", sizeof(path));
/*Set program path*/
ret = IMC100_Set_CurPrgPath(path, robotNo);
if(ret < 0)      //Add exception handlers here

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

int mode = 2;    //Play mode
/*Set play mode*/
ret = IMC100_Set_Mode(mode, robotNo);
if(ret < 0)      //Add exception handlers here

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

cmd = 0;
/*Stop program*/
```

```
ret = IMC100_PrgCtrl(cmd, robotNo);
if(ret < 0)     //Add exception handlers here

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

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

b) Planning points and controlling robot motion through the remote Ethernet client

Make sure that the robot can safely reach the points planned by the remote Ethernet client.

Overall process:



The code example is as follows:

```
int ret = 0;
DWORD dwIP1 = 0xc0a81719; //IP: 192.168.23.25
int ipPort = 2222;
int timeOut = 5; //Communication timeout 5s
```

```
int robotNo = 0;
/*Connect to 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;
/*Obtain control permission*/
ret = IMC100_CurPermit(&ower, &IpAddr, & ipPort, robotNo);
if(ret < 0)
{
	//Add exception handlers here
}
if (ower! = 1) //The current client device is not granted the permission
{
	ret = IMC100_AcqPermit(1, robotNo); //Forced to get the permission, can get it
	normally when ower is 0
	if(ret < 0)
	{
		//Add exception handlers here
	}
}

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

int err = 0;
```

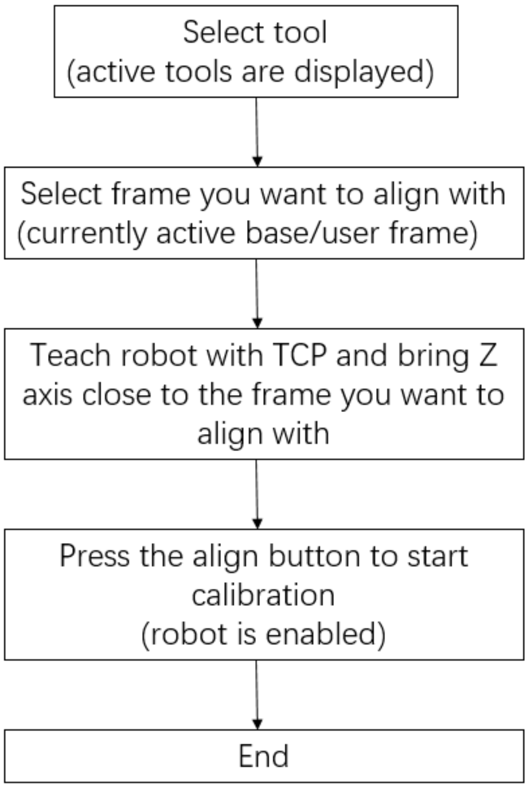
```
/*Fault query*/
ret = IMC100_Get_SysErr(&err, robotNo);
if(ret < 0)
{
	//Add exception handlers here
}
if(err != 0)
{
	/*Reset fault*/
	ret = IMC100_ResetErr(robotNo);
	if(ret < 0)	//Add exception handlers here
}

cmd = 1;
/*Turn on data streaming mode*/
ret = IMC100_DsMode(cmd, robotNo);
if(ret < 0)	//Add exception handlers here
cmd = 1;
/*Enable the motor*/
ret = IMC100_MotorEnable(cmd, robotNo);
if(ret < 0)	//Add exception handlers here

ROBOT_POS pos;
memset(&pos, 0, sizeof(pos));
pos. pos[0] = 10;
pos.coord = 1;
/*MovJ motion*/
ret1 = IMC100_MovJ2(pos1, 100, 0, robotNo)
if(ret < 0)	//Add exception handlers here

/*Motion completion status check*/
ret = IMC100_Get_CurCmdSts(&sts, robotNo);
if(ret < 0)	//Add exception handlers here
if(sts == 0)
{
	//Motion incomplete, custom action, can be queried by cycle
}

memset(&pos, 0, sizeof(pos));
pos. pos[0] = 100;
pos.coord = 1;
/*MovJ motion to next point*/
ret1 = IMC100_MovJ2(pos1, 100, 0, robotNo)
if(ret < 0)	//Add exception handlers here
```

```
cmd = 0;
/*Turn off data streaming mode*/
ret = IMC100_DsMode(cmd, robotNo);
if(ret < 0)    //Add exception handlers here

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

7.10 Pose Calibration

Due to the large number of degrees of freedom and flexible movements of the 6-axis robot, the TCP can be in any pose toward the workbench at different angles. Therefore, it is difficult to ensure that TCP is working directly towards the work object through joint teaching. With this function, the Z-axis direction can be calibrated to align with the axis direction of a user coordinate system (X+, X -, Y+, Y -, Z+, Z -).

The calibration flowchart is as follows:



1. When you select the tool to be calibrated and the user coordinate system, the current tool number and user coordinate system number are activated, and the coordinate setting interface will also be changed accordingly.
2. Press and hold the **Pose-Calib** button. The Z-axis of TCP automatically aligns to the closest axis of the user coordinate system.

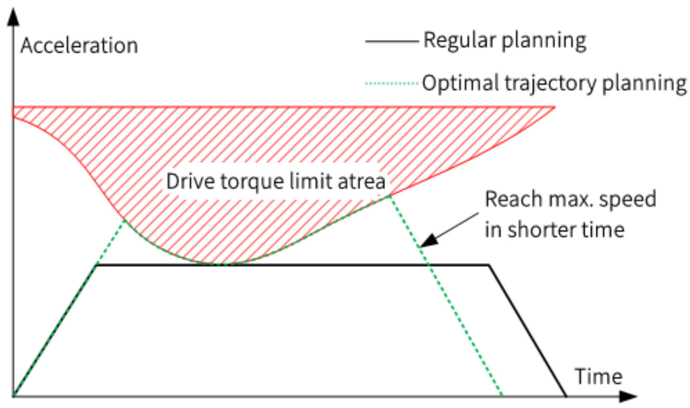
Limit: When around J1 = 0 °, when you align with the X-axis of the base coordinate system through the **Pose-Calib** button, the motion will approach the singularity, resulting in abnormal acceleration or excessive motion.

7.11 Optimal Trajectory

7.11.1 Description

The optimal trajectory function automatically adjusts motion parameters based on load, eliminating the need for users to manually adjust motion parameters, which can bring two improvements:

1. Improved ease of use: No need to manually adjust the parameters, automatically use the appropriate acceleration to achieve the optimal cycle time, while not exceeding the current specification.
2. Improved efficiency: As shown in the figure below, the allowable acceleration at different positions is different, and the optimal trajectory can continuously change with the allowable value, keeping the joint output at maximum and achieving optimal acceleration and deceleration efficiency.



7.11.2 Commissioning Procedure

When the optimal trajectory function is turned on, the robot motion is not affected by the motion acceleration parameters set through the user interface, but are automatically determined from the model.

Step 1: Set the load parameters correctly, including the tool load and the work object load. Note that even if the load is very small, it should be set. If the load mass is zero, the system will assume that the user has forgotten to set the load and will move at the maximum load for safety reasons, resulting in very slow motion.

Step 2: Use the instruction "RapidMove (PTP, ON)" to start RapidMove mode. This instruction can be used at the beginning of the program so that RapidMove can be turned on for the entire project, or the RapidMove mode can be turned on or off at any time during the process. For details, see the instruction reference guide.

Step 3: Use instructions to commission the current and cycle time.

The magnitude of the current is positively correlated with the acc coefficient in the instruction. The acc coefficient in a single instruction can be modified, or it can be batch modified through the

SetAcc instruction. The details of the SetAcc instruction, see the instruction reference guide.
After operation, if the maximum current is less than 100%, it indicates that the cycle time can be further shortened. In this case, gradually increase Acc to 120 or until an alarm occurs. If there are no vibration and current alarms, you can gradually increase the value of SetAccRamp from 50.
If there is a current over limit alarm or drive alarm, first check if the load setting is correct. If the vibration is significant, first reduce the SetAccRamp parameter. If the problem persists, appropriately reduce the value of SetAcc.
Step 4: For situations where the requirements on cycle time are very high and the problem cannot be solved after the above steps, you can observe where the current alarm occurs and only reduce the Acc coefficient at that location, while using 100% or 120% for Acc coefficient at other locations.
Note: This function is based on a kinetic model, if the model is not accurate it can lead to current alarms or low efficiency. If there is a high current alarm only for individual axes and the current of other axes is low, the model may be inaccurate and can be corrected by the model correction function. See 4.4 - **Motion** > **AdvOption** > **TorqueModel**.

7.11.3 Example

```
Start;
    GripLoad 1; //Activates the load and sets the corresponding load parameters in the interface
    RapidMove (ALL, ON); //Enables optimal trajectory function
    SetAcc(100, 100); //Adjusts the current level via the SetAcc instruction
    L[0]:
    Movj P[0],V[100],fine,Tool[0];
    Movj P[1],V[100],fine,Tool[0];
    Goto L[0];
    End;
```

7.12Self-Learning Vibration Suppression

7.12.1 Description

Scenario: The self-learning vibration suppression feature can be used to reduce vibrations when a large vibration condition is found when the program is running.
Step 1: Observe the execution of the program to identify the vibrating instructions with high vibrations, and attach SLON to these motion instructions.
Step 2: Add the SLMode instruction at the beginning of the program to set the vibration suppression level.
Step 3: Re-execute the program and the system automatically learns during the movement process. (The self-learning process will cause the motion instructions to pause for an additional 1.5 seconds after the motion is completed.)
Step 4: After self-learning is completed, the system can be used normally and vibration is suppressed (As self-learning has already been finished in step 3, running the program again will not repeat the learning. The SLMode instruction at the beginning of the program suppresses the

vibration of all motion instructions with SLOn.)
Note:
The learned data does not move with the project and therefore the suppression effect does not directly follow the project.
If self-learned vibration suppression is used on one device and the project is subsequently migrated to another device, the vibration suppression will need to be re-learned. Alternatively, you can export the learned data from the previous device to another device after migrating the project. This saves the self-learning time for unlearned robots in the case of batch replications of production lines.

7.12.2 Related Instructions

The instructions relating to self-learning vibration suppression are as follows.

7.12.2.1 SLVSMode

Name	SLVSMode
Function	Sets the self-learning vibration suppression mode
Description	This instruction is used to enable or disable the self-learning vibration suppression.
Format	SLVSMode modePara;
Parameter	The modePara parameter includes four modes: HighLevel, MidLevel, LowLevel, and Off.
Control permission	Supported under every control permission, including APIs
Scope of instruction	1） Only active in main task, not supported in multitasking 2） Project level: Upon compilation of the program or return to the start of the main program, the parameter is initialized to Off.
Detailed description	1） HighLevel, MidLevel, and LowLevel indicate that self-learning vibration suppression is turned on, with different levels of vibration suppression. Off means vibration suppression is turned off. 2） The higher the level of vibration suppression, the better the effect of vibration suppression. In terms of the amplitude of vibration, HighLevel ≤ MidLevel ≤ LowLevel, and in terms of execution time, HighLevel ≥ MidLevel ≥ LowLevel. 3） When you need to use the self-learning vibration suppression feature, it is recommended to test with MidLevel first. If the vibration does not meet the requirements, switch to HighLevel; if the execution time does not meet the requirements, switch to LowLevel.
Example	1) In the following program, the self-learning vibration suppression mode will be set to the medium level mode after the SLVSNode MidLevel instruction has been executed. For B[0]=0,B[0]<2,Step[1] Movj LP[0],V[30],Z[0],SLOn; Delay T[1]; Movj LP[1],V[30],Z[0],SLOn; Delay T[1];

	SLVSMode MidLevel; EndFor;
Note	

7.12.2.2 SLOn/SLOff/SLReset

Name	SLOn/SLOff/SLReset
Function	It marks movements that require self-learning to suppress vibration.
Description	This parameter is the default parameter for the motion instructions and is used to mark movements that require self-learning to suppress vibrations.
Format	Movj/Movl/Movc/Jump/JumpL, …, [SLOn/SLOff/SLReset];
Parameter	1) This parameter includes SLOn, SLOff, and SLReset. SLOn indicates that the motion requires self-learning, SLOff indicates that the motion does not require self-learning, and SLReset indicates that the motion requires re-learning. 2) SLOff is the default, indicating that the motion does not require self-learning.
Control permission	Not supported for APIs
Scope of instruction	Same with motion instructions, not supported in multitasking
Detailed description	1) Self-learning is possible only when the global motion speed is greater than 49%, and not when it is less than 49%. 2) The time for each self-learning is approximately 1.5 seconds. 3) If SLOn, when the instruction is executed for the first time, the robot will stop for about 1.5 seconds after reaching the destination and automatically learn relevant data and save it in the learning data file. Normally, the robot can automatically learn the required relevant information by running the instruction once, so the robot does not need to stop to learn again when the instruction is repeatedly called later. The total number of times the robot stops to learn during the same motion does not exceed 3. 4) If SLReset, the robot will be forced to stop for about 1.5 seconds after reaching the destination and automatically learn relevant information and save it in the learning data file. 5) If the vibration suppression effect cannot meet the requirements after self-learning using the SLOn parameter, the SLReset parameter can be used, which is equivalent to forcing self-learning every time the motion is executed. The SLReset parameter is generally only used during debugging. After debugging and confirming that the vibration meets the requirements, it is necessary to change SLReset to SLOn or SLOff. 6) For motions with good vibration conditions, it is best not to mark them (default or SLOff), otherwise the robot may stop to perform self-learning when it first reaches the marked position. 7) In general, the more significant the vibration, the better the likelihood of self-learning effect. However, if the vibration is too significant, it may lead to incorrect data being learned. 8) For a motion instruction with SLOn and SLReset parameters, it is necessary to wait

	for the instruction to be executed before the subsequent instructions can be executed. Therefore, for the motion corresponding to such motion instruction, the transition parameters and NWait parameter will be forcibly invalidated.

Example	1) In the following program, after the SLDataClear All instruction is executed, all historical self-learning data will be cleared. Therefore, regardless of whether self-learning has been performed before, self-learning will be performed when the robot moves to points LP[1] and LP[2]. For the motion to point LP [2], the robot continues self-learning before the program jumps out of the for loop. Because the self-learning parameter is SLOff or default, the robot will not perform self-learning when moving to points LP[0] and LP[3]. Because the SLOn and SLReset parameters are present, neither the motion to LP[1] nor the motion to LP[2] will have a transitional effect.
	SLDataClear All;
	For B[0]=0,B[0]<10,Step[1]
	Movj LP[0],V[30],Z[0],SLOff;
	Movj LP[1],V[30],Z[CP],SLOn;
	Movj LP[2],V[30],Z[CP],NWait,SLReset;
	Movj LP[3],V[30],Z[0];
	Set Out[1];
	SLVSMode MidLevel;
	EndFor;

Note	

7.12.2.3 SLDataClear

Name	SLDataClear
Function	Clears the data that has been learned
Description	This instruction is used to clear learned data. When it is called, the robot needs to perform self-learning again.
Format	SLDataClear ClearPara;
Parameter	ClearPara includes All, Current, and Designated Position, which respectively represent the erasure of all learning data and the erasure of learning data at the current position of the robot, and erasure of learning data at the designated position of the robot. Currently, only **Current** is supported.
Control permission	Not supported for APIs
Scope of instruction	Only active in main task, not supported in multitasking
Detailed description	1) When SLDataClear All is called, all learning data from previous learning will be restored to factory defaults. 2) If the robot's self-learning effect is still poor after increasing the global motion speed (the robot's vibration is visible to the naked eyes), SLDataClear All can be called to clear all historical learning data, and then self-learning can be re-started, or the robot can be moved to a position with poor self-learning effect and SLDataClear Current

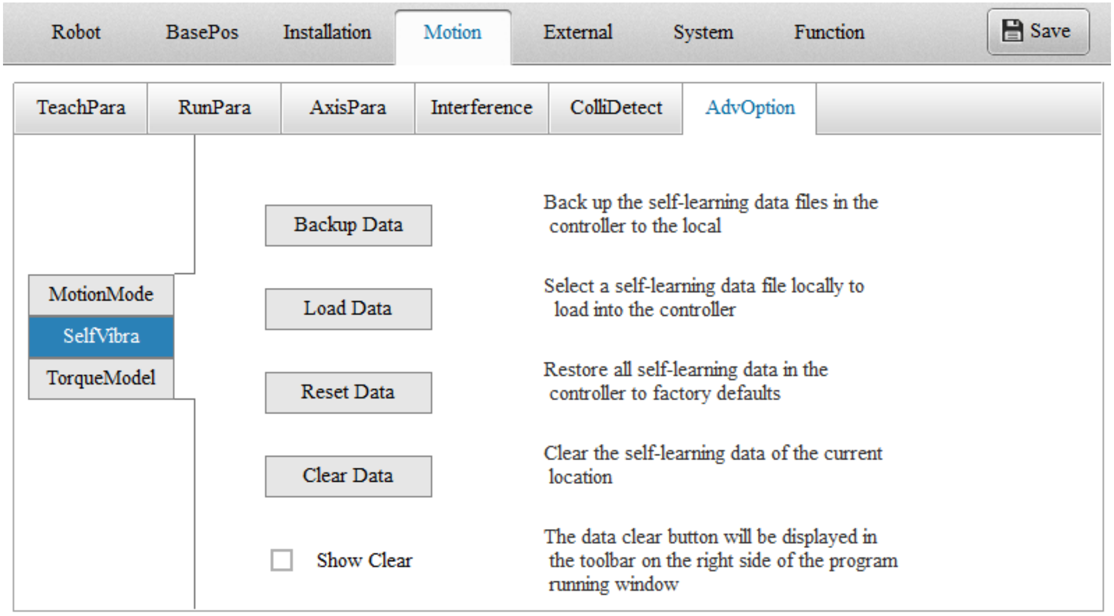
	can be called to clear the learning data of the current position, and then re-start self-learning at that position. 3) If the weight or inertia of the end load of the robot changes significantly (more than 30%), SLDataClear All needs to be called to clear all historical learning data before learning again. 4) If you need to clear the historical learning data and re-learn, you generally only need to call the instruction SLDataClear All at the beginning of the program, and only need to call it once. 5) The SLDataClear All instruction is generally only used during debugging. After debugging is completed, it needs to be deleted or commented out. Otherwise, the learning data that has been learned from each SLDataClear All execution will be completely cleared, resulting in vibration suppression failure and the robot needs to learn again.

Example	1) In the following program, after the SLDataClear All instruction is executed, all historical self-learning data will be cleared. When the motion instruction with SLOn or SLReset parameter is executed later, self-learning will be performed again, that is, self-learning will be performed upon the first execution of Movj LP[1], V[30], Z[0], SLON.
	SLDataClear All;
	For B[0]=0,B[0]<2,Step[1]
	    Movj LP[0],V[30],Z[0];
	    Delay T[1];
	    Movj LP[1],V[30],Z[0],SLOn;
	    Delay T[1];
	    SLVSMode MidLevel;
	EndFor;

Note	

7.12.3 Backup, loading, recovery, and clearing of self-learning data

As shown in the figure below, you can configure the self-learning vibration suppression function in **Set** > **Motion** > **AdvOption** > **SelfVibra** interface. The interface contains 4 options, including Data Backup, Load Data, Reset Data, and Clear Data.



A. Backing up data

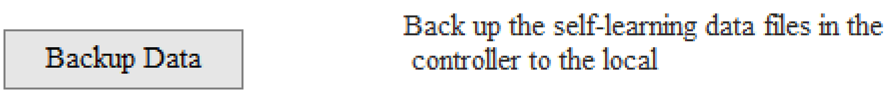
1. Function:

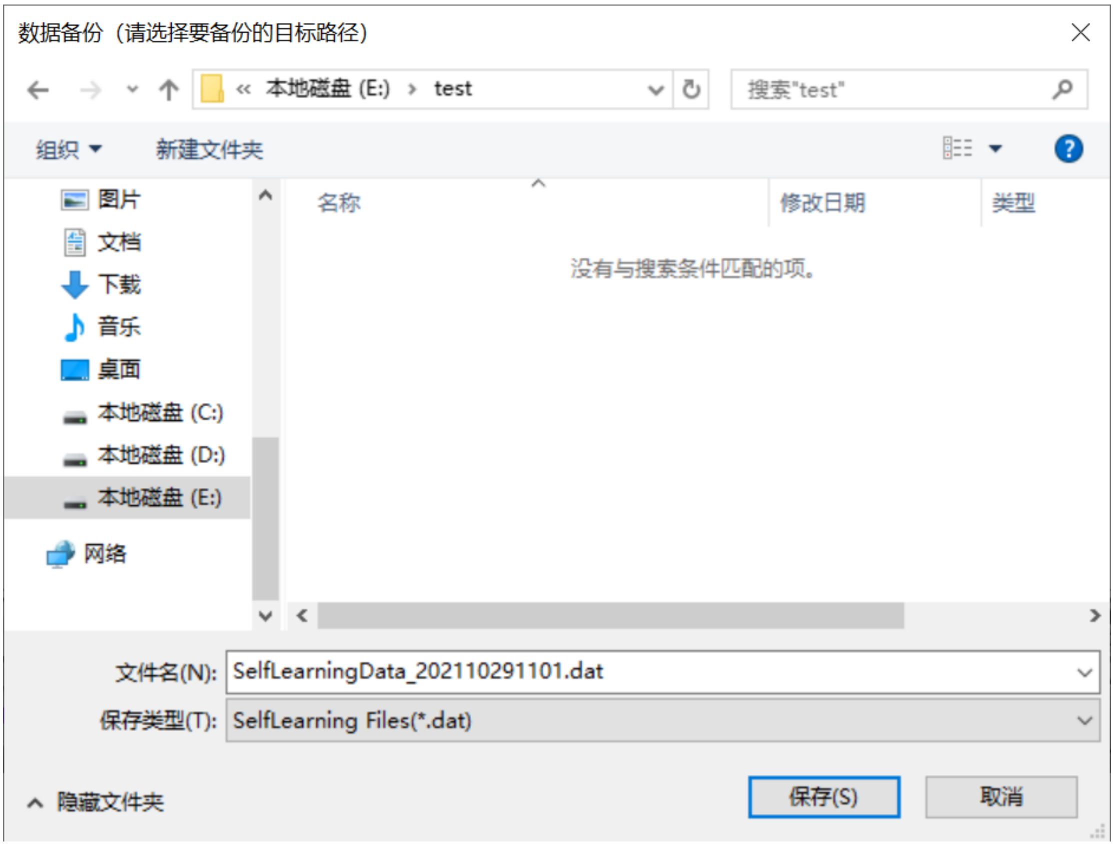
Save the self-learning data in the robot controller to a learning data file and store it to the computer's local disk.

2. Operation method:

1) Click **Data Backup**.

2) Select the save path and file name.





3) Click **Save**.

3. Note:

1) For the backup operation, the default save path is the last saved path. The default file name contains the project name and time information, which makes it convenient for users to manage the learning data file.

B. Loading data

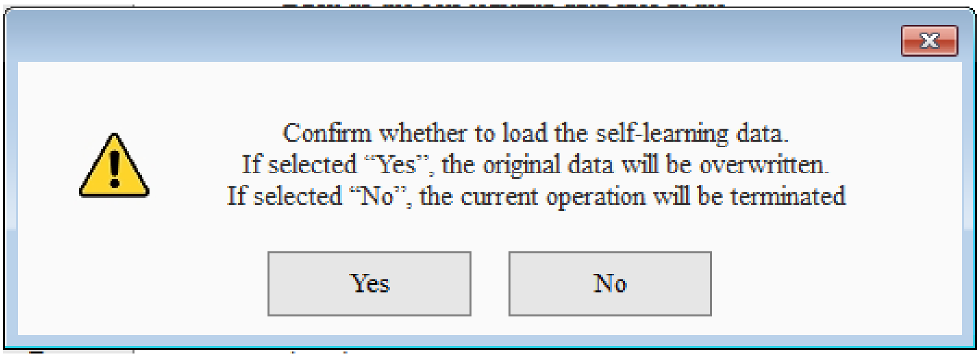
1. Function:

Loads the locally stored learning data file into the robot controller.

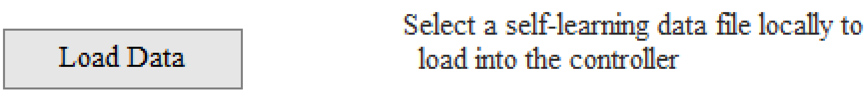
2. Operation method:

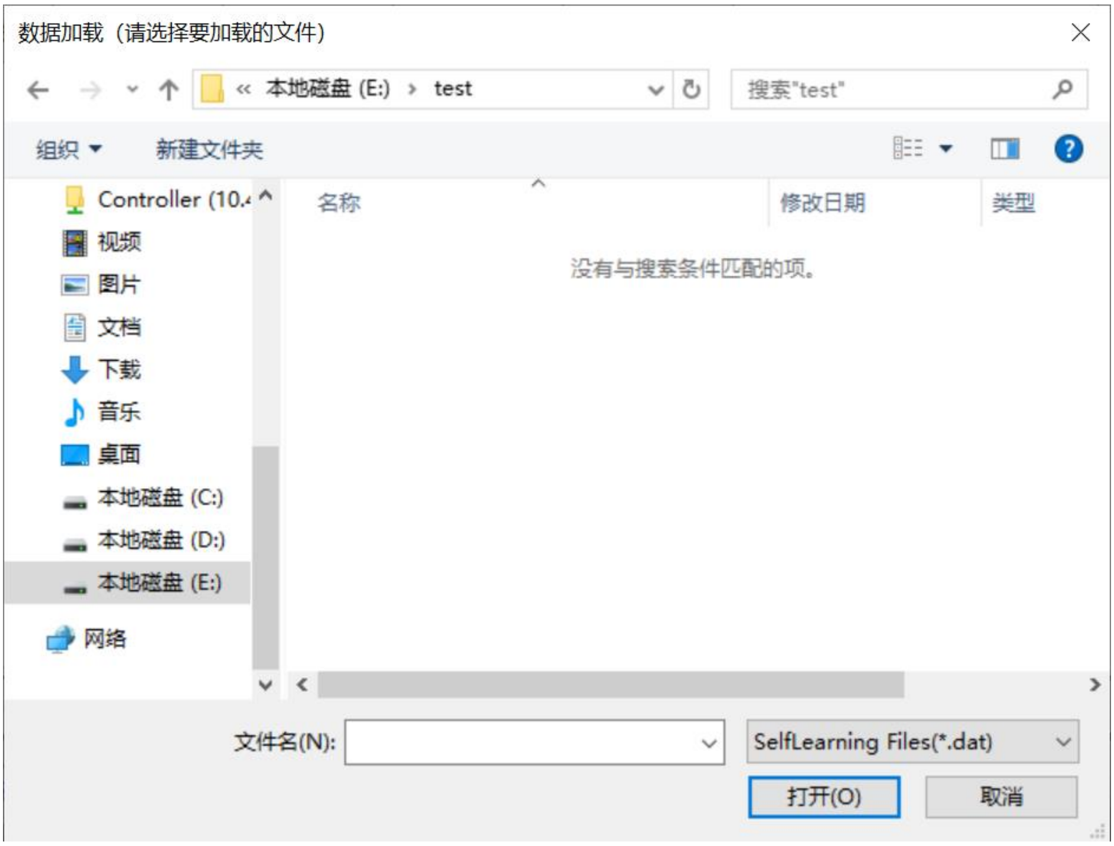
1) Click **Load Data**.

2) The user will be prompted whether to load the learning data. Click **Yes** to load the learning data, or **No** to terminate the current operation.



3) Then, select the load file in the pop-up dialog. The original learning data will be overwritten once the data is loaded successfully.





3. Note:

1) When the learning data is loaded, the file type, model, software version, and checksum information is verified. If the model and software version do not match the controller, the user will be prompted that the loading has failed.
2) If the user changes the content of the learning data file, the user will also be prompted that the loading has failed.

C. Restoring data

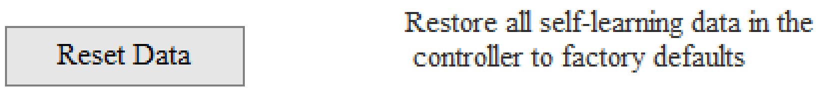
1. Function:

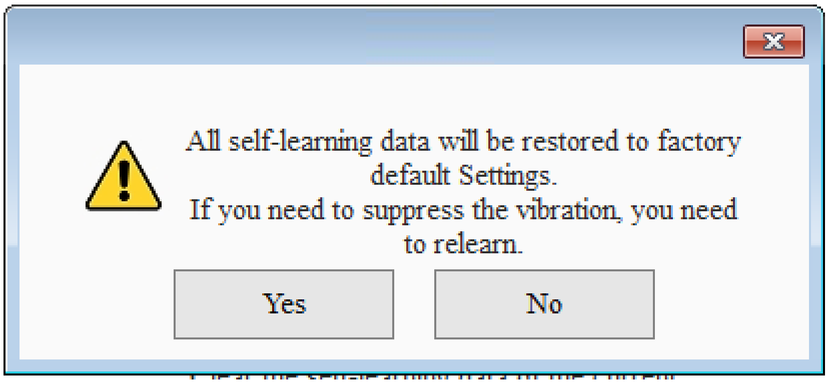
Restores all self-learning data in the controller to factory defaults.

2. Operation method:

1) Click **Reset Data**.

2) The user will be prompted whether to restore the learning data. Click **Yes** to restore the learning data, or **No** to terminate the current operation.





3. Note:

1) After the learning data is restored to factory defaults, since there is no valid learning data, the self-learning vibration suppression does not actually take effect. The robot needs to perform self-learning again before vibration suppression takes effect.

2) When the system is flashed, the learning data is restored to factory defaults. When the system is upgraded, the learning data file will not be automatically upgraded. If the previous and subsequent versions of the learning data files are incompatible, the user will be prompted to upgrade the learning data file.

D. Clearing data

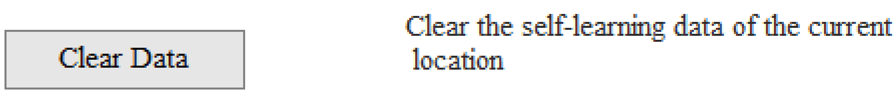
1. Function:

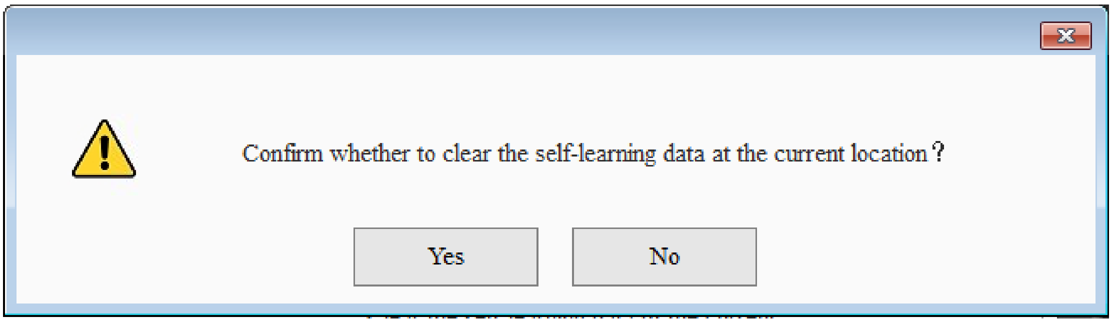
Clears the self-learning data for the current position of the robot in the controller.

2. Operation method:

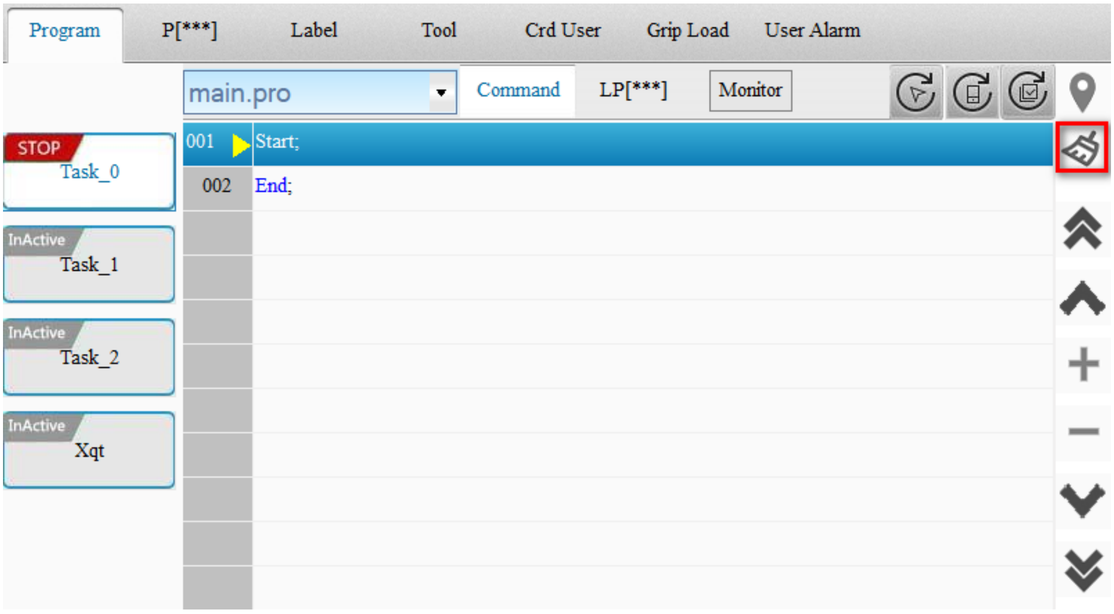
1) Click **Clear Data**.

2) The user will be prompted whether to clear the learning data. Click **Yes** to clear the learning data, or **No** to terminate the current operation.





3) If you check **Show Clear**, the **Clear Data** button will be displayed in the toolbar on the right side of the program running window to facilitate the user to clear the learning data of the current position.



7.12.4 Example

RapidMove(All, OFF); //Avoid significant vibration of the robot after the optimal trajectory is turned on, which may result in poor learning performance

VelSet 100;

SetAccRamp(100,100);

SLDataClear All; //Place this instruction before motion instructions to clear learning data before motion is executed

For B[0]=0,B[0]<2,Step[1]

Movj LP[1],V[30],Z[0];

Movj LP[2], V[30], Z[CP], SLOn; //The robot stops when it reaches the current motion instruction and automatically performs self-learning

Delay T[1];

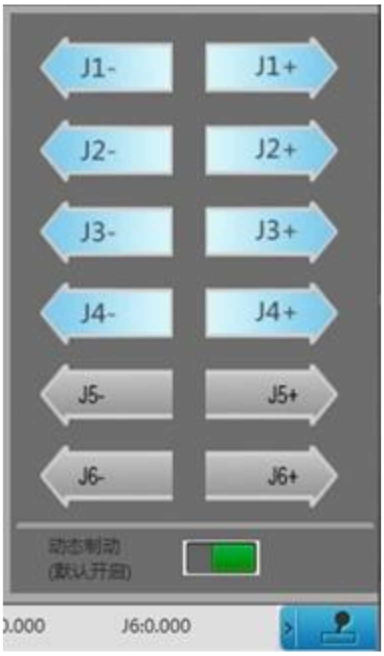
Movl LP[3], V[30], Z[0], SLOn; //The robot stops when it reaches the current motion instruction and automatically performs self-learning

```
        Delay T[1];
        SLVSMode MidLevel;
    EndFor;
```

1) After the SLVSNode MidLevel instruction is executed for the first for loop, the self-learning vibration suppression mode is set to the medium level mode. If effective data has been learned, the second for loop will not perform self-learning again. The vibration of the two motions Movj LP[2] and Movl LP[3] is effectively suppressed.
2) For motion instructions with SLOn or SLReset parameter, the transition parameter is automatically shielded. Therefore, it is best to remove the SLOn parameter after it is confirmed that the vibration suppression effects meet the on-site requirements.
3) To determine whether vibration has been effectively suppressed, you can add a delay instruction to the instruction that require vibration suppression, which can make it easier to determine whether the suppressed vibration can meet on-site requirements.

7.13Releasing Dynamic Brake

For a SCARA robot, you can move its axes easily by turning off the Dynamic Brake switch after the robot is disabled.



Note:

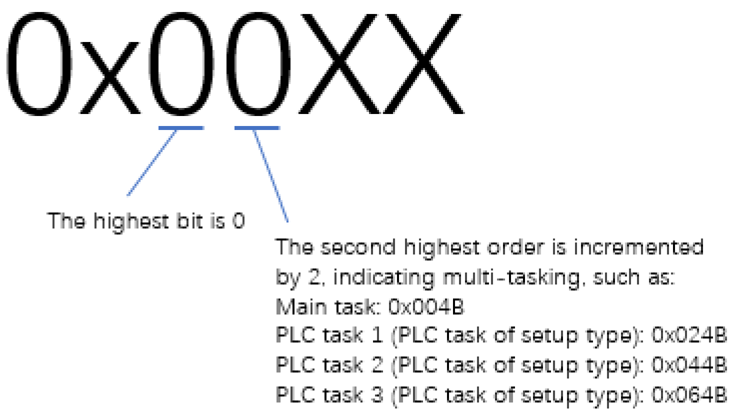
1. This feature is only applicable to the J1, J2, J4 axes.
2. Only when the robot is disabled can the dynamic brake switch be turned on or off.
3. The dynamic brake switch is always on when the robot is enabled.
4. When the robot controller is re-powered, the dynamic brake switch will be reset to ON.

Appendix 1: Robot Alarms and Handling Method

Description:

1. Description of multitasking alarms

Multitasking alarms comply with the following rules.



Only the main task alarms are listed in the following alarm list.

Alarm Code	Description	Cause	Solution
0x0001	Initialization failed (reboot required after fixing error)	1.PF file creation or opening failed. 2.PF file parsing failed.	Restart and check if it returns to normal. If not, please contact the manufacturer.
0x0002	Teach pendant communication module scheduling failure (reboot required after fixing error)	1. The teach pendant thread was not started properly.	Restart and check if it returns to normal. If not, please contact the manufacturer.
0x0003	Vision communication module scheduling failure (reboot required after fixing error)	1. The vision communication thread was not started properly.	Restart and check if it returns to normal. If not, please contact the manufacturer.
0x0004	Internal communication module scheduling failure (reboot required after fixing error)	1. The DSP communication thread was not started properly.	Restart and check if it returns to normal. If not, please contact the manufacturer.
0x0005	Play/Teach function module scheduling failure (reboot required after fixing error)	1. The ARM scheduling thread was not started properly.	Restart and check if it returns to normal. If not, please contact the manufacturer.
0x0006	Data interpolation module	1. The interpolation thread was	Restart and check if it returns to

	scheduling failure (reboot required after fixing error)	not started properly.	normal. If not, please contact the manufacturer.
0x0007	Failed to open EtherCAT communication (reboot needed after fixing error)	1. Configuration file error. 2. EtherCAT slave does not match the system configurations.	Step 1: Check if the EtherCAT connection status on the monitoring interface is abnormal. Step 2: Contact the manufacturer.
0x0008	Failed to open parameter configuration file	1. Failed to open parameter configuration file. 2. Parameter configuration file is corrupted.	System configuration file error, please contact the manufacturer.
0x0009	Decoding error	Program syntax error	Check the program for syntax errors. Refer to the message bar for detailed error messages.
0x000A	Unreasonable program line number	The instruction line number sent by the teach pendant is out of range.	Check whether the line where the blue cursor is located is out of the program range, or reselect the start line number.
0x000B	Wait instruction timeout	Wait instruction has waited longer than set time	1. Check the Wait condition. 2. Reset the wait time for the Wait instruction.
0x000C	Error reading instructions	1. The program file is corrupted. 2. The program file does not conform to specifications.	1. Re-write the program file. 2. Check that the program file conforms to the specifications.
0x000D	0	The nesting call of the subprogram exists.	Check if the Call instruction is nested to subroutines.
0x000E	Motion command decoding error	Motion command decoding error	Check the teaching program.
0x000F	Configuration file operation failed	1. Failed to save the configuration file. 2. Failed to recover the configuration file.	1. Restore factory defaults and power on the system again. 2. Power off and restart.
0x0011	Failed to create axis interpolation thread	1. Failed to create the interpolation thread. 2. The internal testing function is not open.	1. Replace the hardware. 2. Replace the software.
0x0012	Jump instruction failed	1. Point data calculation error in the jump instruction.	Reselect teaching points.
0x0013	IRLink initialization failed (reboot required after fixing error)	1. The number of IRLink slaves is incorrectly configured. 2. The order of IRLink slaves is incorrectly configured.	Reconfirm the IRLink configurations.
0x0014	Failed to save the teaching program	The memory card is loose or cannot be identified.	Check the memory card.

0x0015	Internal communication error between the system and the motion module	DSP software running error	Power on the controller again.
0x0016	Internal enable error	DSP software running error	Power on the controller again.
0x0017	System motion module program running error	DSP software running error	Power on the controller again.
0x0018	Homing failed	Homing failed	Perform homing again.
0x0019	Enable missing	The enablement of the running state is lost.	Check whether the system is in enabled state.
0x001A	Servo parameter error detected	The set parameters do not match the actual servo parameters.	Change the servo parameters using the servo panel or servo background according to the factory parameters. If the actual parameters are inconsistent with the factory parameters, replace the motor or servo.
0x001B	Motion status acquisition abnormality	1.The internal processing of the system is busy. 2. The system motion firmware is damaged.	1. Power off and restart the system again. 2. Contact the manufacturer.
0x001C	Configuration unsuccessful, PLC configuration conflicts with actual controller model	The PLC configuration conflicts with the actual controller model.	Check that the PLC configuration parameters match the actual controller model.
0x001D	Old version PLC configuration, some functions are affected	Old version PLC configuration	The secondary development version number and the robot secondary development version number do not match, please contact the manufacturer.
0x001E	Parameter verification Error (reboot required after fixing error)	When setting servo position on arm-dsp channel, the setup (planned position) and the retrieved value (encoder feedback position) are not consistent with each other.	System failure, please contact the manufacturer.
0x001F	Position synchronization error (reboot required after fixing error)	Error synchronizing controller to servo position	1. Power off and restart the system again. 2. Contact the manufacturer.
0x0020	Robot not allowed to move during startup	The robot is in motion when it is started.	Wait for the robot to stop before starting it.
0x0021	Bad parameter passed in (reboot required after fixing error)	Bad parameter passed in	Check the basic parameter settings of the system.
0x0022	Instruction line not found	The input line number is out of range.	Select the instruction line to run.

0x0023	No point data found	No point is defined.	Check whether a point is defined.
0x0024	Inverse kinematic error	This point is a singular point of the robot.	Modify the coordinate of this point.
0x0025	Point coordinate system parameter error	The coordinate system values exceed the range.	Get a point again.
0x0026	Line or arc instructions do not allow sudden changes in arm parameters	A sudden change in the arm parameter of the MOVL and MOVC instruction is not allowed.	Get a new point or add a joint transition point.
0x0027	V parameter out of range	The V parameter exceeds the range (1-100).	Modify the V parameter.
0x0028	Z parameter out of range	The Z parameter exceeds the range (0-5).	Modify the Z parameter.
0x0029	TOOL parameter out of range	The TOOL parameter exceeds the range (0-15).	Modify the Tool parameter
0x002A	USER parameter out of range	The User parameter exceeds the range (0-15).	Modify the User parameter.
0x002B	ACC parameter out of range	The ACC parameter exceeds the range (1-100).	Modify the Acc parameter.
0x002C	Until parameter out of range	The I/O number exceeds the range (0-255).	Modify the Until In parameter.
0x002D	Pallet parameter error	Pallet (PNo, i, j, k), PNo, i, j, k greater than or equal to 0.	Modify the Pallet parameter.
0x002E	Pallet not defined	No pallet number is defined.	Define the pallet before using it.
0x002F	Repeat parameter error	The Repeat parameter exceeds the range.	Modify the Repeat parameter.
0x0030	Decoding error	Error in instruction parsing during operation	Check the running instructions and modify them according to the prompts.
0x0031	Error in inverse kinematic calculation for jog motion in base coordinate system	The destination for jog motion in the base coordinate system exceeds the running space or is located at a singularity.	1. Check the step of jog motion. 2. Check the direction of jog motion.
0x0032	Error in inverse kinematic calculation for jog motion in tool coordinate system	The destination for jog motion in the tool coordinate system exceeds the running space or is located at a singularity.	1. Check the step of jog motion. 2. Check the direction of jog motion.
0x0033	Error in inverse kinematic calculation for jog motion in user coordinate system	The destination for jog motion in the user coordinate system exceeds the running space or is located at a singularity.	1. Check the step of jog motion. 2. Check the direction of jog motion.
0x0034	Tool load parameter setting exceeds limit	The tool load parameter setting value is out of range.	Modify the tool load parameters.
0x0035	Work object load parameter	The work object load parameter	Modify the work object load

	setting exceeds limit	setting is out of range.	parameters.
0x0036	Arm load parameter setting exceeds limit	The arm load parameter setting value is out of range.	Modify the arm load parameters.
0x0038	Parameter setting error	Parameter setting is out of range.	Check that the parameter setting is within the range.
0x0039	API initialization failed (reboot required after fixing error)	The API communication thread was not started properly.	1. Power off and restart the system again. 2. Contact the manufacturer.
0x004B	Failed to mount memory card	Failed to mount memory card	1. Power off and restart the system again. 2. Remove the memory card and install it again. 3. Replace the memory card.
0x004C	Duplicate port number or IP address	The port number or IP address is duplicated.	Replace the port number or IP address.
0x004D	Ethernet communication error	Ethernet communication error	Check the communication line and retransmit the data.
0x004E	Network interference	The controller is connected to multiple Ethernet terminals.	Check whether multiple terminals are connected to the same controller.
0x004F	Failed to open vision port	Vision instruction error	Check the vision instruction.
0x0050	Memory card not recognized	1. Memory card not inserted into the controller. 2. Poor memory card contact. 3. Memory card damaged. 4. There is a problem with initialization of the memory card upon power on.	Check the system hardware; power off and restart the system.
0x0051	EtherCAT disconnected	EtherCAT is disconnected.	Check EtherCAT communication.
0x0052	IRLink disconnected	IRLink is disconnected.	Check IRLink communication.
0x0053	Too short interval between start and pause	The interval between start and pause is too short.	Restart
0x0054	Failed to acquire vision characteristic values	Failed to acquire vision characteristic values while executing the instruction.	Obtain vision characteristic values again after checking vision processing.
0x0055	Decoding not completed	The program may have instruction symbols or syntax error.	Check the syntax of instructions used in the editing program.
0x0056	Robot type error	The robot has no related type firmware.	Restart the system or check the DSP firmware.
0x0057	FPGA firmware loading failed (reboot required after fixing error)	FPGA firmware error or startup exception	Restart the system or check the FPGA firmware.
0x0058	DSP firmware loading failed	DSP firmware error or startup	Restart the system or check the

	(reboot required after fixing error)	exception	DSP firmware.
0x0059	System status error upon mode switching	When the robot is switched to teach or play mode, it is detected that the project has not been compiled properly or the configuration file has not been completed properly.	Internal system error, please contact the manufacturer.
0x005A	I/O parameter setting error	I/O parameter setting error	Check that the range of I/O parameters set in the instruction is reasonable.
0x005B	IP address error	IP address acquisition or setting error	Check network line connection.
0x005C	The teach pendant does not have access to IRLink configuration.	IRLink has already been configured on the secondary development platform.	Use the current configuration or cancel the configuration on the secondary development platform.
0x005D	Shared memory mapping error	Error in mapping shared memory between RC and PLC.	Contact the technical support.
0x005E	(User-defined alarm 0)	The system triggered user-defined alarm 0.	Check the user alarm 0.
0x005F	(User-defined alarm 1)	The system triggered user-defined alarm 1.	Check the user alarm 1.
0x0060	(User-defined alarm 2)	The system triggered user-defined alarm 2.	Check the user alarm 2.
0x0061	(User-defined alarm 3)	The system triggered user-defined alarm 3.	Check the user alarm 3.
0x0062	(User-defined alarm 4)	The system triggered user-defined alarm 4.	Check the user alarm 4.
0x0063	(User-defined alarm 5)	The system triggered user-defined alarm 5.	Check the user alarm 5.
0x0064	(User-defined alarm 6)	The system triggered user-defined alarm 6.	Check the user alarm 6.
0x0065	(User-defined alarm 7)	The system triggered user-defined alarm 7.	Check the user alarm 7.
0x0066	(User-defined alarm 8)	The system triggered user-defined alarm 8.	Check the user alarm 8.
0x0067	(User-defined alarm 9)	The system triggered user-defined alarm 9.	Check the user alarm 9.
0x0068	(User-defined alarm 10)	The system triggered user-defined alarm 10.	Check the user alarm 10.
0x0069	(User-defined alarm 11)	The system triggered user-defined alarm 11.	Check the user alarm 11.
0x006A	(User-defined alarm 12)	The system triggered user-defined alarm 12.	Check the user alarm 12.

0x006B	(User-defined alarm 13)	The system triggered user-defined alarm 13.	Check the user alarm 13.
0x006C	(User-defined alarm 14)	The system triggered user-defined alarm 14.	Check the user alarm 14.
0x006D	(User-defined alarm 15)	The system triggered user-defined alarm 15.	Check the user alarm 15.
0x006E	Interference area 0 alarm	The robot is located in the interference area 0.	Check the robot position and setting value of the interference area 0.
0x006F	Interference area 1 alarm	The robot is located in the interference area 1.	Check the robot position and setting value of the interference area 1.
0x0070	Interference area 2 alarm	The robot is located in the interference area 2.	Check the robot position and setting value of the interference area 2.
0x0071	Interference area 3 alarm	The robot is located in the interference area 3.	Check the robot position and setting value of the interference area 3.
0x0072	Interference area 4 alarm	The robot is located in the interference area 4.	Check the robot position and setting value of the interference area 4.
0x0073	Interference area 5 alarm	The robot is located in the interference area 5.	Check the robot position and setting value of the interference area 5.
0x0074	Interference area 6 alarm	The robot is located in the interference area 6.	Check the robot position and setting value of the interference area 6.
0x0075	Interference area 7 alarm	The robot is located in the interference area 7.	Check the robot position and setting value of the interference area 7.
0x0076	Interference area 8 alarm	The robot is located in the interference area 8.	Check the robot position and setting value of the interference area 8.
0x0077	Interference area 9 alarm	The robot is located in the interference area 9.	Check the robot position and setting value of the interference area 9.
0x0078	Interference area 10 alarm	The robot is located in the interference area 10.	Check the robot position and setting value of the interference area 10.
0x0079	Interference area 11 alarm	The robot is located in the interference area 11.	Check the robot position and setting value of the interference area 11.
0x007A	Interference area 12 alarm	The robot is located in the interference area 12.	Check the robot position and setting value of the interference

			area 12.
0x007B	Interference area 13 alarm	The robot is located in the interference area 13.	Check the robot position and setting value of the interference area 13.
0x007C	Interference area 14 alarm	The robot is located in the interference area 14.	Check the robot position and setting value of the interference area 14.
0x007D	Interference area 15 alarm	The robot is located in the interference area 15.	Check the robot position and setting value of the interference area 15.
0x007F	Data streaming mode not turned off	The system is in data streaming mode.	Turn off the data streaming mode.
0x0080	Emergency stop alarm	The emergency stop button is pressed.	Release the emergency stop button to clear the alarm.
0x0081	No reverse movement data found	The reverse movement data has been executed.	Terminate the reverse movement.
0x0082	Error closing port	The port number is out of range or the port is not opened.	Check the port number.
0x0083	TCP port overflow	Peripheral TCP application connections are excessive.	Close useless TCP connections.
0x0084	API communication processing error	The API channel is occupied by other applications for a long time or has been blocked due to faults in previous API application processing.	Close or reduce the previous API application process.
0x0085	Arc trajectory uncontrollable	The arc start point is uncertain.	Add other motion instructions before and after the arc instruction.
0x0086	Version mismatch	The teach pendant version does not match the controller version.	Match the teach pendant and controller version.
0x0089	IP conflict	IP settings conflict.	Reset the IP address.
0x008A	File system not identified in the memory card.	The file system on the memory card is incorrect.	Reformat the memory card on the teach pendant.
0x008B	Servo parameter reading failed	The controller fails to read servo parameters.	Optimize the connection between the servo and the controller.
0x008C	Parameter out of range	Parameter out of range	Set the parameters within the range.
0x008D	Illegal I/O configuration	Configured I/O is controlled by a PLC or does not exist.	Reset I/O.
0x008E	Illegal I/O setup operation.	The set I/O lacks the I/O control.	Check the I/O control.
0x008F	I/O does not exist	Sending screw locking startup parameters to the servo fails.	1. Check that the electric screwdriver servo firmware

			matches with the controller. 2. Clear parameters and restart the system.
0x0090	Failed to start tightening	Sending screw locking startup parameters to the servo fails.	1. Check that the electric screwdriver servo firmware matches with the controller. 2. Clear parameters and restart the system.
0x0091	Failed to stop electric screwdriver	Sending screw locking stop parameters to the servo fails.	1. Check that the electric screwdriver servo firmware matches with the controller. 2. Clear parameters and restart the system.
0x0092	Screw status detection failed	Reading screw locking status from the servo fails.	1. Check that the electric screwdriver servo firmware matches with the controller. 2. Clear parameters and restart the system.
0x0093	Failed to read screw tightening parameters	Reading screw locking setting parameters from the servo fails.	1. Check that the electric screwdriver servo firmware matches with the controller. 2. Clear parameters and restart the system.
0x0094	Failed to write screw tightening parameters	Writing screw locking setting parameters to the servo fails.	1. Check that the electric screwdriver servo firmware matches with the controller. 2. Clear parameters and restart the system.
0x0095	Screw data display failed	Obtaining screw locking display data from the servo fails.	1. Check that the electric screwdriver servo firmware matches with the controller. 2. Clear parameters and restart the system.
0x0096	Failed to reset lock count	Writing a screw locking counter clearing flag to the servo fails.	1. Check that the electric screwdriver servo firmware matches with the controller. 2. Clear parameters and restart the system.
0x0097	Failed to get thread for servo error	Failed to get thread for servo error	Restart the robot.
0x0098	Failed to start loosening	Sending screw removal startup parameters to the servo fails	1. Check that the electric screwdriver servo firmware matches with the controller. 2. Clear parameters and restart the

			system.
0x0099	Failed to write screw loosening parameters	Writing screw removal setting parameters to the servo fails.	Check the electric screwdriver servo firmware matches with the controller.
0x009A	Failed to read screw loosening parameters	Obtaining screw removal setting data from the servo fails.	Check the electric screwdriver servo firmware matches with the controller.
0x009B	Fallback point setting out of range	Fallback point setting out of range	Set the fallback point for screw loosening within the range.
0x00A0	TCP port for reading and writing is not open	The TCP port is not open.	Make sure that the TCP port is properly connected.
0x00A1	Conveyor vision port not closed	Normal vision is used without closing dynamic vision.	Close the dynamic vision by instruction CNVIOSION OFF.
0x00A2	Conveyor error or camera pixel error	The conveyor number is incorrect or the transfer data type of the camera is incorrect.	Reset the conveyor number or camera data transfer type.
0x00A3	Dynamic vision coordinate conversion error	Error converting dynamic vision pixels to camera coordinate system.	Check that the pixels sent from vision equipment are correct, otherwise the camera needs to be recalibrated.
0x00A4	No reverse movement data found	There is no reverse movement data.	Clear the error.
0x00A5	Lack of motion instructions before arc instruction	The instruction before arc motion is not Movj or Movl (This appears in only the single-step teaching.)	Ensure that the instruction before the arc instruction is Movj or Movl.
0x00A6	Specified file not found	The file does not exist or the file path is incorrect.	Check that the file exists or the file path is correct.
0x00A7	Error saving diagnostic information	The relevant files do not exist during the process of saving diagnostic information.	Internal system error, please contact the manufacturer.
0x00A8	Error exporting diagnostic information	The relevant files do not exist during the process of exporting diagnostic information.	Check that the USB device is normal; otherwise it is an internal system error, please contact the manufacturer.
0x00A9	Excessive number of files	There are too many files (including folders) in the current folder.	Delete excessive files. Ensure the number of files+folders in a single folder is not greater than 100.
0x00AA	Pause buffer data sending failure	Failed to send buffer data when going from pause to start.	Restart from stop.
0x00AB	Multitasking number does not exist	Multitasking number is written incorrectly or in the wrong range.	Check the task number.

0x00AC	Multitasking number incorrect or occupied	The multitasking number is incorrect or has been occupied.	Check the task number.
0x00AD	Multitasking creation failed	The system is busy; the task is already running or has been restarted; task 0 was started under non-remote I/O permissions.	(1) Re-run the XQT command to check that it runs normally. (2) Stop the current multitasking and start it again. (3) Switch to remote I/O permissions and restart task 0.
0x00AE	Failed to open or close the position latching function.	The hardware is not connected or the servo is not ready.	Check the hardware connections and servo configuration.
0x00AF	Failed to call the controller debug function	The controller debug function module failed.	Check that the configuration of the controller debug switch is correct.
0x00B0	Failed to set load parameters	The system is busy or the parameter is out of range.	Reset or check the parameter.
0x00B1	Serial port number or baud rate error	1. The input serial port number is out of range. 2. With the serial port not closed, another serial port with the same number but different baud rate is opened.	1. Check that the input serial port number is within the range. 2. Close the serial port via the close instruction and open a serial port with new baud rate via the close instruction.
0x00B2	Modbus parameter read-write error	Modbus read-write execution failed	Update the software version.
0x00B3	Failed to modify P variable	Failed to open or write the program file where the P variable is located.	Re-modify it or update the software version.
0x00B4	Vision communication exception: Client and server connection error	The communication between the robot and the target device has been interrupted.	Reestablish the connection.
0x00B5	Vision Communication Exception: Robot sending data error	The network was disconnected while the robot was sending data or there was an error while transmitting data.	Check that the network connection is normal.
0x00B6	Pose correction calculation error	Wrong tool was selected or the alignment position is a singularity.	1. Check that the tool selection is correct. 2. Check that the position that the robot needs to align with is not a singularity.
0x00B7	Error in switching station program in Modbus mode	In Modbus mode, the station program is switched when the current station program is not stopped.	Set the current station program to the stop state, and then switch the station program.
0x00B8	System power-down error	The system suddenly loses	Clear alarm

		power during operation.	
0x00B9	Illegal InoRobShop axis configuration	The number or type of axes configured by InoRobShop does not match the actual condition.	1. Reconfigure the correct axis in InRobShop. 2. In InoTeachPad, go to **Set** > **System** > **Others** > **Clear PLC-CFG** and clear the PLC configuration (Note: The built-in PLC program will also be cleared.)
0x00BA	Error selecting multiple programs in Modbus mode	In Modbus mode, multiple station programs are selected simultaneously.	Select only one program.
0x00BB	Network disconnected during motion in teach mode under teach pendant control	When the robot is under the control of teach pendant and operating in teach mode, it is detected that the network connection between the teach pendant and the controller is disconnected.	Clear the alarm after reconnecting the teach pendant and the controller.
0x00BC	The read-write serial port is not opened.	The read-write serial port is not opened.	Open the serial port by configuring the controller parameter or by executing the open instruction.
0x00BD	P variable cannot be changed before a program is selected or when the program is running.	P variable cannot be changed before a program is selected or when the program is running.	Select a program or make sure that the robot is stopped.
0x00BE	Client-server communication timeout, network connection lost	Client-server communication timeout, network connection lost	Check the network connection.
0x00BF	Peer server shutdown, network connection lost	Peer server shutdown, network connection lost	Open the server.
0x00C0	J1 servo parameter error detected (reboot required after fixing error)	One or more servo parameters of J1 are wrong.	Reboot the robot after modifying the wrong servo parameter values.
0x00C1	J2 servo parameter error detected (reboot required after fixing error)	One or more servo parameters of J2 are wrong.	Reboot the robot after modifying the wrong servo parameter values.
0x00C2	J3 servo parameter error detected (reboot required after fixing error)	One or more servo parameters of J3 are wrong.	Reboot the robot after modifying the wrong servo parameter values.
0x00C3	J4 servo parameter error detected (reboot required after fixing error)	One or more servo parameters of J4 are wrong.	Reboot the robot after modifying the wrong servo parameter values.

0x00C4	J5 servo parameter error detected (reboot required after fixing error)	One or more servo parameters of J5 are wrong.	Reboot the robot after modifying the wrong servo parameter values.
0x00C5	J6 servo parameter error detected (reboot required after fixing error)	One or more servo parameters of J6 are wrong.	Reboot the robot after modifying the wrong servo parameter values.
0x00CB	Robot software version and servo version do not match	Robot software version and servo version do not match	Upgrade the servo version to match the robot software.
0x00D0	Failed to set servo acceleration parameter	Failed to set servo acceleration parameter	Check the servo module status.
0x00D1	Failed to restore system configuration file	Failed to restore configuration file.	Re-save the configuration file or restart the robot.
0x00D2	P-variable not allowed to be modified during program execution in play mode or during program execution	P-variable was modified in play mode or during program execution under the remote Modbus control.	Stop the program and switch to teach mode before modifying the P variable.
0x00D3	Failed to modify P variable	Failed to modify P variable in memory or failed to modify P variable in file.	1. Try to modify again. 2. Check that the GP variable to be modified exists in the file. 3. Update the software version. 4. Contact the manufacturer.
0x00D4	Configured I/O program does not exist	Configured I/O program does not exist	Check that the configured I/O program file exists.
0x00D5	The length of data sent by the peer device exceeds the limit.	The length of the data sent by the peer device exceeds the limit when communicating via socket or serial port.	Reduce the length of data sent by the peer.
0x00D6	The BRD variables entered under Modbus control are out of range.	The BRD variables entered under Modbus control are out of range.	Check the range of BRD values set via Modbus.
0x00D7	The calculated P value is illegal.	Internal system error	1. Try to modify the P variable again. 2. Contact the manufacturer.
0x00D8	Input value of P variable is illegal.	The arm parameter, coordinate system, user number, or tool number of the entered P variable is illegal.	Check the arm parameter, coordinate system, user number, or tool number of the entered P variable.
0x00D9	GetCurPoint instruction failed to get current location	GetCurPoint instruction failed to get current location	Reduce the frequency of GetCurPoint instruction calls as appropriate.
0x00DA	Start position deviates too much from the motion interrupt position	Under the remote control, the motion was interrupted in play mode. When the motion was restarted, it was detected that	1. Manually move the robot to the interrupt position. 2. Return to the start line of the program and run the program

		the difference between the current position and the interrupt position exceeds the following thresholds: SCARA robot: 5° 5° 500° 3° (any joint angle); 6-axis robot: 5° 5° 5° 5° 5° 5° 5° 5°.	again.
0x00DB	Failed to clear trajectory data	Failed to clear trajectory data	Contact the manufacturer.
0x00DC	Failed to write servo parameters	Failed to write the servo parameters into the controller.	Optimize the connection between the servo and the controller.
0x00DD	Direct motion point not reachable	1. The target point of direct motion exceeds the limit. 2, The target point of direct motion is a singular point.	1. Check the range of the target point and whether the target point is a singular point.
0x00DE	Encoder multi-turn value has been cleared, please restart.	The encoder multi-turn value is too large and is cleared when the encoder is zeroed. A reboot is required.	Restart the controller.
0x00E0	SN mismatch between drive and encoder (motor)	1. The encoder voltage is too low. 2. The battery is not connected during power-off.	1. Replace the encoder cable; replace a new battery with matching voltage. 2. Set the parameter "Absolute encoder reset enable" to 1 to clear the fault.
0x00E1	Bus undervoltage	1. The power supply of the main circuit is unstable or power failure occurs. 2. Instantaneous power failure occurs. 3. The power supply voltage drops during operation. 4. Controller fault.	1. Increase the capacity of the power supply. 2. Increase the capacity of the power supply. 3. Increase the capacity of the power supply. 4. Replace the controller.
0x00E2	Bus overvoltage	1. The voltage input to the main circuit is too high. 2. The power supply is unstable or affected by lightning. 3. The motor is in abrupt deceleration status and the maximum braking energy exceeds the energy absorption value. 4. The bus voltage sampling value deviates greatly from the measured value.	1. Replace or adjust the power supply according to the following specifications: 220V-240V±10% (198V to 264V). 2. Connect a surge protection device and then switch on the main circuit and control circuit power supplies again. If the fault persists, replace the servo drive. 3. After confirming the input voltage of the main circuit is within the specified range,

		5. Controller fault.	increase the acceleration/deceleration time if the operating conditions allow. 4. Contact the technical support. 5. Replace the controller.
0x00E3	Main circuit open	Mains power supply is unstable with voltage fluctuations.	Check if the power supply system is stable and if there are fluctuations in the voltage range.
0x00E4	Controller and drive power undervoltage	Controller fault	Replace the controller.
0x00E5	Controller and drive power overvoltage	Controller fault	Replace the controller.
0x00E6	Abnormal brake and I/O power supply	Controller fault	Replace the controller.
0x00E7	System power detection error	A power loss was detected by the servo module, but not by the controller module, and the power detection module is faulty.	Ensure the stability of the power supply and restart the system.
0x00F0	J1 axis encoder battery alarm	1. The encoder voltage is too low. 2. The battery is not connected during power-off.	1. Replace the encoder cable; replace a new battery with matching voltage. 2. Set the parameter "Absolute encoder reset enable" to 1 to clear the fault.
0x00F1	J2 axis encoder battery alarm	1. The encoder voltage is too low. 2. The battery is not connected during power-off.	1. Replace the encoder cable; replace a new battery with matching voltage. 2. Set the parameter "Absolute encoder reset enable" to 1 to clear the fault.
0x00F2	J3 axis encoder battery alarm	1. The encoder voltage is too low. 2. The battery is not connected during power-off.	1. Replace the encoder cable; replace a new battery with matching voltage. 2. Set the parameter "Absolute encoder reset enable" to 1 to clear the fault.
0x00F3	J4 axis encoder battery alarm	1. The encoder voltage is too low. 2. The battery is not connected during power-off.	1. Replace the encoder cable; replace a new battery with matching voltage. 2. Set the parameter "Absolute encoder reset enable" to 1 to clear the fault.
0x00F4	J5 axis encoder battery alarm	1. The encoder voltage is too	1. Replace the encoder cable;

		low. 2. The battery is not connected during power-off.	replace a new battery with matching voltage. 2. Set the parameter "Absolute encoder reset enable" to 1 to clear the fault.
0x00F5	J6 axis encoder battery alarm	1. The encoder voltage is too low. 2. The battery is not connected during power-off.	1. Replace the encoder cable; replace a new battery with matching voltage. 2. Set the parameter "Absolute encoder reset enable" to 1 to clear the fault.
0x00F6	J1 axis encoder overtemperature	The encoder temperature is too high.	Cooling
0x00F7	J2 axis encoder overtemperature	The encoder temperature is too high.	Cooling
0x00F8	J3 axis encoder overtemperature	The encoder temperature is too high.	Cooling
0x00F9	J4 axis encoder overtemperature	The encoder temperature is too high.	Cooling
0x00FA	J5 axis encoder overtemperature	The encoder temperature is too high.	Cooling
0x00FB	J6 axis encoder overtemperature	The encoder temperature is too high.	Cooling
0x00FC	Failed to set servo acceleration parameter	1. The servo SDO channel is not smooth. 2. The servo has a hardware or software fault.	1. Try to save the acceleration parameters again. 2. Contact the manufacturer to update the servo software.
0x00FD	Failed to restore system configuration file	The file system is corrupted, or the internal channel of the system is damaged.	Try to save the acceleration parameters again or restart the robot.
0x1001	Duplicate directory	The directory to be created already exists.	Rename the directory to be created.
0x1002	Memory operation error, parent directory does not exist.	The currently created directory has no parent directory.	Recreate the directory in another path.
0x1003	Renamed directory does not exist.	The directory to be renamed does not exist.	Refresh the directory and check that the directory exists.
0x1004	Deleted directory does not exist.	The directory to be deleted does not exist.	Refresh the directory and check that the directory exists.
0x1005	Error sending directory (not a directory or the directory does not exist)	The directory to be sent to the host is illegal.	Refresh the directory and check that the directory exists.
0x1006	Memory error and path error in creating file	Error creating path	Refresh the file and check the path.

0x1007	Renamed file does not exist.	The file to be renamed does not exist.	Refresh the file and check that the file exists.
0x1008	Deleted file does not exist or the path does not exist.	The file to be deleted does not exist.	Refresh the file and check that the file exists.
0x1009	The given path to the file does not exist.	The given file path is illegal.	Check the path.
0x100A	Non-file sent	What is to be sent is not a file.	Check on the handheld teach pendant whether the object to send is a file.
0x100B	Non-directory sent	What is to be sent is not a directory.	Check on the handheld teach pendant whether the object to send is a directory.
0x100C	Frame sequence error	Frame sequence error in the process of sending a large file	Resave or open the file.
0x100D	The device is not functioning properly and is actively disconnected from the network.	1. The handheld teach pendant is not closed according to normal operations; 2. The network is disconnected abnormally.	1. For any error due to abnormal operations, please actively disconnect the network cable; 2. For abnormality, check the error causes in conjunction with the existing error codes.
0x100E	Time format error	Time format error	Set the time according to correct format.
0x100F	System time correction error	System time calculation circuit error	Check the network connection.
0x1010	RTC time correction error	RTC external batteries do not exist or are low.	Replace batteries or check the current hardware.
0x1011	Copy error	Misoperation during file copying	Perform copy operations again by referring to the user manual.
0x1012	Cut error	Misoperation during file cutting	Perform cut operations again by referring to the user manual.
0x1014	File encryption failed	Error encrypting file	Try to encrypt the file again.
0x1015	Unknown communication code	Version mismatch	Ensure the teach pendant and the controller match in version.
0x1016	Eth1 physical link down	Eth1 line not working	1.Check if the network cable is plugged in or not in good contact. 2.Hardware failure, contact the manufacturer.
0x1017	Eth2 physical link down	Eth2 line not working	1.Check if the network cable is plugged in or not in good contact. 2.Hardware failure, contact the manufacturer.
0x1018	Controller fan 1 failure	FAN1 blocked	1. Check if there are foreign objects blocking FAN1 in the front chamber of the cabinet,

			which prevents the fan from running. 2. The fan itself is faulty, replace it with a new fan.
0x1019	Controller fan 2 failure	FAN2 blocked	1. Check if there are foreign objects blocking FAN1 in the front chamber of the cabinet, which prevents the fan from running. 2. The fan itself is faulty, replace it with a new fan.
0x101A	The temperature on the top of the controller is too high.	ST1 ambient temperature exceeds 60°C.	1. Check whether FAN1 in the front chamber operates normally and whether the dust-proof cotton of the fan cover is seriously blocked. 2. Check whether there are any components with abnormal temperature in the front chamber. The temperature control switch automatically resets when the ambient temperature drops to 45±5°C. 3. The temperature control switch is faulty, replace it with a new one.
0x101B	Transformer overtemperature	The transformer ST2 temperature exceeds 140℃.	1. Check whether FAN2 in the rear chamber operates normally and whether the dust-proof cotton of the fan cover is seriously blocked. 2. Check whether the QF2 circuit breaker is faulty, and whether there is an abnormally large load on the secondary side of the transformer. The temperature control switch automatically resets when the B-phase coil of the transformer cools down to 105±15°C. 3. The transformer is damaged, replace it with a new one.
0x101C	Output I/O overcurrent	Output I/O overcurrent	Check the hardware line.
0x101D	Fan not installed	The fan is not installed or has	Check the fan installation.

		poor contact.	
0x101E	Controller overtemperature alarm	The temperature inside the controller is too high.	Cool down and restart
0x101F	Controller overtemperature warning	The temperature inside the controller is too high.	Cool down and restart
0x1020	Oscilloscope function thread startup failed	System fault	Re-power
0x1021	Mains power fluctuations detected	Mains power supply is unstable with voltage fluctuations.	Check if the power supply system is stable and if there are fluctuations in the voltage range.
0x1022	J1 brake disconnected	1. The Power line is not connected. 2. Parameter H02-16 is incorrectly configured. 3. The brake is disconnected. 4. The power supply of the brake is abnormal.	1. Turn off the power supply, connect the power line and power up again. 2. For motors without brake function, set H02-16 to 0. 3. Replace the cable. 4. Replace the controller.
0x1023	J2 brake disconnected	1. The Power line is not connected. 2. Parameter H02-16 is incorrectly configured. 3. The brake is disconnected. 4. The power supply of the brake is abnormal.	1. Turn off the power supply, connect the power line and power up again. 2. For motors without brake function, set H02-16 to 0. 3. Replace the cable. 4. Replace the controller.
0x1024	J3 brake disconnected	1. The Power line is not connected. 2. Parameter H02-16 is incorrectly configured. 3. The brake is disconnected. 4. The power supply of the brake is abnormal.	1. Turn off the power supply, connect the power line and power up again. 2. For motors without brake function, set H02-16 to 0. 3. Replace the cable. 4. Replace the controller.
0x1025	J4 brake disconnected	1. The Power line is not connected. 2. Parameter H02-16 is incorrectly configured. 3. The brake is disconnected. 4. The power supply of the brake is abnormal.	1. Turn off the power supply, connect the power line and power up again. 2. For motors without brake function, set H02-16 to 0. 3. Replace the cable. 4. Replace the controller.
0x1026	J5 brake disconnected	1. The Power line is not connected. 2. Parameter H02-16 is incorrectly configured. 3. The brake is disconnected. 4. The power supply of the	1. Turn off the power supply, connect the power line and power up again. 2. For motors without brake function, set H02-16 to 0. 3. Replace the cable.

		brake is abnormal.	4. Replace the controller.
0x1027	J6 brake disconnected	1. The Power line is not connected. 2. Parameter H02-16 is incorrectly configured. 3. The brake is disconnected. 4. The power supply of the brake is abnormal.	1. Turn off the power supply, connect the power line and power up again. 2. For motors without brake function, set H02-16 to 0. 3. Replace the cable. 4. Replace the controller.
0x1028	Safety door alarm	The safety circuit is open and the safety door is open.	Close the safety circuit or the safety door.
0x1029	System is not fully powered off and may cause system problems, power the system on again	The servo's 24V drive power is cut off, causing abnormal power supply to the inverter module, resulting in an N-phase overcurrent alarm. The fault cannot be recovered and the system must be powered on again.	Re-power the system.
0x102A	Program directory file path does not exist.	The program directory is missing.	Contact the manufacturer.
0x102B	Program file path does not exist.	The specified program file is missing.	1. Check that the program exists. 2. Contact the manufacturer.
0x102C	Invalid variable setting	Incorrect variable name or value	1. Check if the variable name or label is P, B, R, D, Str, and whether the subscript is reasonable. 2. Check that the type of the value matches and is within the range.
0x1030	Self-learning vibration suppression data file missing	The self-learning vibration suppression data file is deleted after the robot is turned on.	1. Restart the robot to automatically restore the self-learning vibration suppression data file. 2. Import the self-learning vibration suppression data file from outside.
0x1031	Print messages are too frequent and the buffer is full, print information may be lost	The Print instruction is used too often in the program to print messages, resulting in too many messages being printed and the output buffer is full.	Reduce the frequency or amount of Print instructions in the program to avoid loss of print messages.
0x1032	Device in use	The device is being used.	Close any programs or windows that may be using the device and try again.
0x1033	Controller fan rotor locked.	The fan is stuck by a foreign	Turn off the power and remove

		object.	the foreign object from the fan, or contact the technical support.
0x1040	Self-learning vibration suppression profile does not exist.	The self-learning vibration suppression profile is deleted by mistake after the robot is turned on.	1. Restart the robot to automatically restore the data file for self-learning vibration suppression function. 2. Import the data file for self-learning vibration suppression function from outside.
0x1041	Failed to clear all self-learning vibration suppression data	1. The robot system is busy. 2. The file system is corrupted.	1. Try clearing again. 2. Contact the manufacturer.
0x1042	Failed to clear self-learning vibration suppression data file for current position	1. The robot system is busy. 2. The file system is corrupted.	1. Try clearing again. 2. Contact the manufacturer.
0x1043	Self-learning vibration suppression data file corrupted	The file exists but cannot be opened, or the contents of the file are illegal.	Restore default parameters on the host controller and restart the robot.
0x1044	Failed to create self-learning vibration suppression data file	File system exception or other system exception.	Contact the manufacturer.
0x1045	Failed to import self-learning vibration suppression data file	1. The contents of the imported file are illegal. 2. Channels are busy, causing failure in setting DSP parameters.	1. Re-import a legal file. 2. Try to restart the robot.
0x1101	Invalid length of data exchanged internally	When requested to provide data, the DSP does not return valid data as required.	Check the DSP software version or replace the hardware.
0x1102	Verification error for data exchanged internally	When requested to provide data, the DSP did not return valid data as requested or returned data with error codes.	Check the DSP software version or replace the hardware.
0x1103	Error writing block data	The GPMC channel is abnormal.	Check the DSP software version or replace the hardware.
0x1104	Error reading block data	The GPMC channel is abnormal.	Check the DSP software version or replace the hardware.
0x1105	System internal block data buffer full	The FPGA buffer is full and cannot accept new data.	Retry to write data after delay for a period of time.
0x1106	Internal channel open error: Abnormal or occupied	The GPMC channel is abnormal or has been occupied.	Restart the controller.
0x1107	Internal channel open error: Abnormal or closed	The GPMC channel is abnormal or has been closed.	Restart the controller.
0x1108	System busy internally	The DSP does not respond to	1. Retry after a while. 2. Restart

		robot instructions.	the robot. 3. Check if the DSP has been suspended or terminated.
0x1109	Error getting channel resources	The CPMC channel is frequently occupied so that currently the system cannot apply for use of CPMC resources.	Retry to use them after delay for a period of time.
0x110A	Timeout waiting for motion module response	The response time to robot instructions from the DSP exceeds the set maximum waiting time.	1. Retry after a while. 2. Restart the robot. 3. Check if the DSP has been suspended or terminated.
0x110B	Motion module failed to execute the command sent by ARM	The DSP failed to execute the robot instructions.	Check if the current operation is legal or check the DSP software version.
0x110C	Illegal parameters set for motion module	Parameters set by the robot for the DSP are incorrect, e.g. out of the parameter range.	Check that function call interface parameters are correct.
0x110D	Illegal commands set for motion module	Commands requested by the robot to the DSP are invalid.	Check the command word and ensure that there is processing on this command in the DSP.
0x110E	The number of axes configured for the system is inconsistent with the number of online scanned axes.	The model is not matched or servo is disconnected.	Check the model and servo connection to ensure that the current robot is consistent with the configured robot.
0x110F	The axis data sent is inconsistent with the data read.	Data check error	Check the software version or contact the manufacturer.
0x1110	The I/O data sent is inconsistent with the data read.	Data check error	Check the software version or contact the manufacturer.
0x1111	Error in servo entering homing mode via EtherCAT instructions	The servo cannot enter the homing mode.	Check the software version or contact the manufacturer.
0x1112	Error in servo exiting homing mode via EtherCAT instructions	The servo cannot exit the homing mode.	Check the software version or contact the manufacturer.
0x1113	Error in setting homing parameters for the servo via EtherCAT instructions	The set homing parameters are not accepted by servo.	Check the software version or contact the manufacturer.
0x1114	System parameter check error	Data check error	Check the software version or contact the manufacturer.
0x1115	Error opening data channel	The GPMC channel number is incorrect.	The default channel is 0. Ensure that the channel number is correct.

0x1116	Data channel mapping error	The GPMC data channel cannot map data to the memory.	Check the software version or contact the manufacturer.
0x1117	Data channel mapping error	The GPMC channel is abnormal.	Check the software version or contact the manufacturer.
0x1118	Data channel mapping error	The GPMC channel is abnormal.	Check the software version or contact the manufacturer.
0x1119	Device error	The GPMC channel is abnormal.	Check the software version or contact the manufacturer.
0x111A	Error opening data channel	The GPMC channel is abnormal.	Check the software version or contact the manufacturer.
0x111B	Data communication discrepancy	The GPMC channel is abnormal.	Check the software version or contact the manufacturer.
0x111C	Data communication discrepancy	The GPMC channel is abnormal.	Check the software version or contact the manufacturer.
0x111D	Error system requesting memory	The system cannot assign requested memory.	Check whether the system memory is close to the limit or whether the requested memory is too large.
0x111E	Error configuring I/O data bias information on the IR-LINK bus	Data check error	Check the software version or contact the manufacturer.
0x111F	Error configuring AD data bias information on the IR-LINK bus	Data check error	Check the software version or contact the manufacturer.
0x1120	Error configuring DA data bias information on the IR-LINK bus	Data check error	Check the software version or contact the manufacturer.
0x1121	Error configuring encoder data bias information on the IR-LINK bus	Data check error	Check the software version or contact the manufacturer.
0x1122	Error configuring AD parameter (range) on IR-LINK bus	Data check error	Check the software version or contact the manufacturer.
0x1123	Error configuring DA parameter (range) on IR-LINK bus	Data check error	Check the software version or contact the manufacturer.
0x1124	Error configuring module number on IR-LINK bus	Data check error	Check the software version or contact the manufacturer.
0x1125	Error synchronizing controller planning position with encoder feedback position	The current axis does not exist or the GPMC channel or DSP firmware is abnormal.	Check the software version or contact the manufacturer.
0x1200	Robot in emergency stop state when enabled	The robot is in an emergency stop when the enable command is issued.	Check the emergency stop button, clear the emergency stop status and re-enable the robot.

0x1201	Excessive fluctuation in robot joint position when the robot is enabled	The robot is currently in a vibrating or dragged state.	Reduce the joint vibration of the robot and enable it after the robot stabilizes.
0x1202	Servo alarm present when the robot is enabled	The robot has a servo alarm when enabled.	Clear the servo alarm and re-enable the robot.
0x1203	Robot enabled too quickly after being disabled	The robot is enabled too quickly after it is disabled.	Clear the alarm and re-enable the robot.
0x1204	Alarms other than joint overrun present when robot is enabled	There are other alarms besides joint overrun.	Clear the alarm and re-enable the robot.
0x1205	Excessive fluctuation in robot joint position when getting zero point	The robot is currently in a vibrating or dragged state.	Reduce the joint vibration of the robot and get the zero point after the robot stabilizes.
0x2001	Segment data overlap	The previously input target point is the same with the currently input target point.	Reteach the robot with different points.
0x2002	Error calculating input arc parameters	Circular arc interpolation information cannot be calculated because: (1) At least two points are too close;(2) Three points are approximately in the same line;(3) Pose change is too large;(4) Transition is performed near a singularity.	Re-teach the robot with other points to calculate the circular arc.
0x2003	Error calculating input linear parameters	Linear interpolation information cannot be calculated because: (1) Pose change is too large;(2) Transition is performed near a singularity.	Re-teach other points to calculate the line.
0x2004	Inverse kinematics error	The robot is at a singularity or out of reach.	Disable the robot, switch to the joint mode and then move the robot out of the singularity, or change the target points to those that can be arrived.
0x2005	Singularity error	The robot moves to a singular position.	Switch to the joint mode and move the robot out of the singularity.
0x2006	Enable off during running	(1) Power failure of a drive occurs;(2) A drive is wired incorrectly;(3) A drive fails.	Check that the drive is normal.
0x2007	Reserved	Reserved	Reserved
0x2008	I/O index out of range	The physical I/O module does not exist.	Check that a corresponding physical I/O module is available.
0x2009	Jump parameter setting error	MH parameter is beyond the	Modify the limit height or reselect

		limit of J3 axis.	a start or end point.
0x200A	Incorrect arm type parameter	Three arm type parameters before the end point are not consistent with those of the start point in linear or circular motion.	Modify the motion to joint motion or reselect points to ensure consistent arm type.
0x200B	Inappropriate motion characteristic parameters	The motion parameter input range is unreasonable.	Modify motion parameters such as speed and acceleration.
0x200C	DA operation error	The channel is configured as current output, but the voltage command is used. Or the channel is configured as voltage output, but the current command is used.	Operate the DA port using a command consistent with the configuration.
0x200D	The command to enable servo is sent, but the servo is not enabled actually.	(1) The main power supply of the servo is not switched on. (2) The joint may be in deceleration process. (3) The joint is in motion state and does not arrive at the position.	(1) Check whether the strong current button of the controller is pressed down. (2) The interval between servo stop and start or enable is too short. (3) Check whether the joint arrives at the position. Amplify the arrival error or adjust the servo parameters. (4) Status word feedback is too slow.
0x200E	Joint motion parameter input error	(1) The points are beyond the space range for the Delta robots. (2) MoveJ and MOVL or MOVC performs transition near a singularity.	(1) Check whether the points are beyond the space range for the Delta robots. (2) Check whether MoveJ and MOVL or MOVC performs transition near a singularity.
0x200F	Robot not returned to zero	The robot does not perform the homing operation when an incremental encoder is used.	For incremental encoder, perform the homing operation first.
0x2010	Robot radius direction out of bounds	The X and Y combined radius at the end of the robot is greater than the set radius.	Under the rectangular coordinate system, step so that the robot end moves in the direction of reduction of the X and Y combined radius.
0x2011	Robot positive Z direction out of bounds	The Z at the end of the robot is greater than the setpoint.	Under the rectangular coordinate system, step so that the robot end moves in the negative direction of Z.

0x2012	Robot negative Z direction out of bounds	The Z at the end of the robot is smaller than the setpoint.	Under the rectangular coordinate system, step so that the robot end moves in the positive direction of Z.
0x2013	Robot out of bounds	The teaching point exceeds the boundary.	Change the teaching points so that they are within the work space of the robot.
0x2016	The included angle between J2 and J3 axes of the palletizing robot is too small.	The included angle between J2 and J3 axes of the palletizing robot is too small.	In the teach mode, rotate the J3 axis positively or the J2 axis negatively.
0x2017	The included angle between J2 and J3 axes of the palletizing robot is too large.	The included angle between J2 and J3 axes of the palletizing robot is too large.	In the teach mode, rotate the J3 axis positively or the J2 axis negatively.
0x2018	Abnormal robot speed	The robot joint speed exceeds twice the allowable maximum speed.	1) Ensure that the allowable maximum speed and acceleration of joints are reasonably set; 2) Ensure that the robot is not near a singularity; 3) Reduce the linear motion speed.
0x2019	Motion parameter error	The motion planning parameters are abnormal.	Check that the motion parameters (speed, acceleration) are reasonable.
0x201A	The robot's position speed or orientation speed exceeds the setting value.	The robot end motion exceeds the set position or orientation speed.	1. If the MoveJ instruction is used, set the MoveJ speed coefficient to a smaller value. 2. Check whether the set orientation speed is consistent with the J4 joint speed.
0x201D	Arm type change error	The arm type change instruction (specially used for SCARA robots) is not supported near a singularity.	Replace the point with a point far from the singularity and then call the arm type change instruction.
0x201E	The robot acceleration is abnormal.	The robot joint speed exceeds 50 times the allowable maximum acceleration.	1) Ensure that the allowable maximum speed and acceleration of joints are reasonably set; 2) Ensure that the robot is not near a singularity; 3) Reduce the speed and acceleration of the linear motion.
0x201F	Abnormal speed setting	Insufficient internal storage in DSP.	Decrease the Cartesian speed or increase the joint speed
0x2020	Abnormal speed setting	Insufficient internal spline storage in DSP	Reduce the position and orientation speed, or increase the

			joint speed.
0x2021	The lower bound of worksapce is too large or distance of stop is too small.	The lower bound of worksapce is too large or distance of stop is too small for the tracking process.	Decrease the lower bound first, and then increase the robot stop distance.
0x2022	Robot out of lower bound of workspace	The robot exceeds the set operating range in the tracking process.	Adjust the lower working boundary and the upper pickup boundary; increase the robot speed; reduce the conveyor speed.
0x2023	Conveyor speed too large	The conveyor speed exceeds a reasonable range.	The conveyor speed exceeds the maximum speed limit (1 m/s for the linear conveyor and 180°/s for the turntable conveyor).
0x2024	Conveyor speed fluctuation too large	The conveyor speed fluctuates excessively.	Check that the conveyor motor speed does not fluctuate excessively or that the conveyor is abnormal.
0x2025	Vision data waiting timeout	No returned data is received for a long period of time after vision triggering signals are sent, and the vision processing cycle is greater than the photographing interval.	Check whether the vision processing period is greater than the photographing interval.
0x2026	Robot coordinate type error	A static coordinate is used in the tracking instruction or a dynamic coordinate is used in the ordinary motion instruction.	Check whether the point type is 7 (dynamic object coordinate system) in the tracking instruction or whether a type 7 point is used in the non-tracking instruction.
0x2027	Dynamic point coordinate error	The given dynamic target position is incorrect, singular or out of bounds.	Check that the coordinate given by vision is within a reasonable range.
0x2028	Syntax error in conveyor tracking instruction	Refsys Convyor or Refsys Base is used continuously.	Check that RefConvyor and RefBase are jointly used.
0x2029	Failed to establish coordinate system for grasping the work object	The Refsys Convyor instruction is executed before the GetCnvObject instruction is executed.	Call the GetCnvObject instruction first.
0x202A	Conveyor vision port error	Multiple conveyors adopting vision detection are used.	Check whether more than two vision inputs are used at the same time.
0x202B	Single-step teaching is not allowed.	The conveyor tracking-related instructions do not allow single-step teaching.	Single-step operation on instructions between Refsys Convyor and Refsys Base is not allowed.

0x202C	A disabled conveyor is used in the instruction.	A disabled conveyor is used in the instruction.	Check whether the conveyor used in the program is disabled.
0x202D	PTP motion not allowed in the tracking process	Joint motions such as MovJ and JumP are used in the tracking process.	Check whether the MovJ or jump instruction is used between Refsys Convyor and Refsys Base.
0x202E	Conveyor speed direction error	The conveyor speed is detected to be a negative value.	(1) Check that the encoder direction parameter on the conveyor setting interface is correct. (2) Check that the conveyor is not slipping.
0x202F	The start position for teaching in Cartesian system is at a singular position where inverse kinematic solution cannot be executed.	The start position for teaching in Cartesian system is at a singular position where inverse kinematic solution is impossible.	Switch to joint mode and move out of the singular position.
0x2030	Inconsistent latch counters	The robot position latching counters of the respective axes are inconsistent, resulting in incorrect latch state feedback from the servo.	Check that the latching signals of the respective axes are active.
0x2031	Latching buffer full	When the robot position is latched, the latching buffer is full.	The latching speed is too fast, or there are too many latched positions that are not used.
0x2032	The motion range of J1 or J2 exceeds 180°.	When the inverted SCARA robot performs interpolation motion, the motion range of J1 or J2 axis exceeds 180°, making it impossible to ensure that the termination angle of interpolation motion is consistent with the predetermined angle.	Avoid the above situations.
0x2034	The motion range of J2 is through the singular point.	When the inverted SCARA robot performs interpolation motion, the J1 axis moves through the singularity, making it impossible to ensure that the termination angle of interpolation motion is consistent with the predetermined angle.	Avoid the above situations.
0x2035	EtherCAT communication feedback error	EtherCAT bus feedback data is missing.	Check that the EtherCAT cable is properly connected and there is

			no external interference.
0x2036	Position latch timeout	1. The servo is not properly configured with probe function. 2. The correct I/O edge signal is not triggered. 3. Hardware interference.	1. Check if a valid edge signal is output. 2. Check if the servo probe is configured. 3. Check for hardware interference.
0x2041	Limit-triggering point present in planned trajectory	The planned trajectory includes points that may trigger the limit.	Check the corresponding motion segment for points that may trigger limit.
0x2042	The constraint of velocity in trajectory is very low.	(1) In the planning, the calculated spatial velocity constraint is too small, and the velocity parameters are set incorrectly. (2) The trajectory is close to a singularity.	(1) Adjust the set speed parameters. (2) Adjust the trajectory away from the singularity.
0x2043	A position where inverse kinematic solution cannot be executed exists in the planned trajectory.	Inverse kinematic solution cannot be executed at an intermediate point of the planned trajectory.	Modify points in the erroneous section of trajectory.
0x2044	The position goes beyond the lower operating boundary during the dynamic tracking preprocessing.	The target point is detected to go beyond the lower operating boundary during the dynamic tracking preprocessing.	1. Check that the received vision data point is within a reasonable range. 2. Check that the dynamic coordinates in the given motion instruction are not out of the lower operating boundary.
0x2045	Limit triggered at J1 in the planned trajectory	A limit is triggered at J1 in the planned trajectory.	Check the corresponding motion segment for points that may trigger limit.
0x2046	Limit triggered at J2 in the planned trajectory	A limit is triggered at J2 in the planned trajectory.	Check the corresponding motion segment for points that may trigger limit.
0x2047	Limit triggered at J3 in the planned trajectory	A limit is triggered at J3 in the planned trajectory.	Check the corresponding motion segment for points that may trigger limit.
0x2048	Limit triggered at J4 in the planned trajectory	A limit is triggered at J4 in the planned trajectory.	Check the corresponding motion segment for points that may trigger limit.
0x2049	Limit triggered at J5 in the planned trajectory	A limit is triggered at J5 in the planned trajectory.	Check the corresponding motion segment for points that may trigger limit.
0x204A	Limit triggered at J6 in the planned trajectory	A limit is triggered at J6 in the planned trajectory.	Check the corresponding motion segment for points that may trigger limit.

0x204B	Excessive orientation change found in planned trajectory	The motion angle of J4, J5, and J6 in the trajectory planned through linear and circular motion instructions exceeds 179.9°.	1. Reduce the orientation change during a single motion. 2. Check the difference in ARM_TYPE of points between the alarming program line and the previous program line. 3. Change the orientation for linear or circular motion.
0x2060	Insufficient instruction space	The joint speed is too small while the spatial speed is too large.	(1) Increase the joint speed. (2) Reduce the spatial speed.
0x2061	Insufficient space for planned positions	The joint speed is too small while the spatial speed is too large.	(1) Increase the joint speed. (2) Reduce the spatial speed.
0x2062	Interpolation data error	(1) The joint speed and acceleration are not reasonably set. (2) The motion trajectory is too long.	(1) Set the joint speed and acceleration appropriately. (2) Split the trajectory into multiple segments.
0x2063	Insufficient spline space	(1) The spline distance is too large. (2) The speed is too low during transition.	(1) Reduce the transition length. (2) Increase the set speed. (3) Save the error message and contact the manufacturer.
0x2064	Spline interpolation planning error	Spline interpolation planning error	Save the error message and contact the manufacturer.
0x2065	Speed fitting error	Speed fitting error	Save the error message and contact the manufacturer.
0x2066	Speed planning error	Speed planning error	Save the error message and contact the manufacturer.
0x2067	Joint transition planning error	Incorrect calculation results during joint transition planning due, for example, to unreasonable input parameters.	If possible, fine-tune the points or motion parameters
0x2068	Speed look-ahead check not reasonable	It is checked that the look-ahead results are not reasonable.	Save the error message and contact the manufacturer.
0x2069	Adaptive planning results check not reasonable	Adaptive planning results check not reasonable	Save the error message and contact the manufacturer.
0x206A	General planning results check not reasonable	General planning results check not reasonable	Save the error message and contact the manufacturer.
0x206B	Joint speed or acceleration exceeds the set value during normal transition	The set joint speed may be exceeded during a normal transition in CP mode.	(1) Decrease the position speed and orientation speed or increase the joint speed appropriately. (2) Adjust points to move away from the singularities. Reduce the

			transition level and transition length.
0x206C	The joint speed in tracking motion exceeds the set value.	The joint speed in tracking motion exceeds the set value.	Reduce the conveyor speed or the robot speed.
0x206D	Smooth stop exception	The deceleration distance is insufficient when a smooth stop is made.	Save the error message and contact the manufacturer.
0x206E	Joint transition parameter error	Joint transition parameter error	Modify the points.
0x206F	Singular position alarm	Proximity to the singular position causes joint speed and acceleration to be too large.	1. Adjust the motion away from the singularities. 2. Set the motion speed near the singularities small.
0x2071	In teaching mode, a limit is triggered at J1 in the planned trajectory.	A limit is triggered at J1 in the planned trajectory.	Check the corresponding motion segment for points that may trigger limit.
0x2072	In teaching mode, a limit is triggered at J2 in the planned trajectory.	A limit is triggered at J2 in the planned trajectory.	Check the corresponding motion segment for points that may trigger limit.
0x2073	In teaching mode, a limit is triggered at J3 in the planned trajectory.	A limit is triggered at J3 in the planned trajectory.	Check the corresponding motion segment for points that may trigger limit.
0x2074	In teaching mode, a limit is triggered at J4 in the planned trajectory.	A limit is triggered at J4 in the planned trajectory.	Check the corresponding motion segment for points that may trigger limit.
0x2075	In teaching mode, a limit is triggered at J5 in the planned trajectory.	A limit is triggered at J5 in the planned trajectory.	Check the corresponding motion segment for points that may trigger limit.
0x2076	In teaching mode, a limit is triggered at J6 in the planned trajectory.	A limit is triggered at J6 in the planned trajectory.	Check the corresponding motion segment for points that may trigger limit.
0x2077	In teaching mode, a position where inverse kinematic solution cannot be executed exists in the planned trajectory.	Inverse kinematic solution cannot be executed at an intermediate point of the planned trajectory.	Modify points in the erroneous segment.
0x2078	Error in calculating linear input parameters in teaching mode	Unable to calculate linear interpolation information.	Re-teach the robot with other points to calculate the linear interpolation information.
0x2079	The arm type is inconsistent with the current point during preprocessing in teaching state and space interpolation cannot be performed.	The ARM_TYPE parameter is inconsistent with the current point during preprocessing and spatial interpolation cannot be performed.	Modify points in the erroneous segment.

0x207A	Orientation motion is found to be too large in teaching state.	The orientation motion angle is found to be greater than 180 degrees in the planned trajectory.	(1) Reduce the pose change during a single motion. (2) Check the difference in ARM_TYPE of points between the alarming program line and the previous program line.
0x207B	The constraint of velocity in trajectory is very low in manual mode.	The constraint of velocity in trajectory is too small.	In teaching mode, the spatial velocity constraint calculated in the planning is too small due to incorrect speed parameter setting or singularities.
0x2081	Valid hardware signal for emergency stop	Hardware emergency stop is active.	A hardware emergency stop is active. Confirm safety and then release the alarm.
0x2082	Mode switch timeout	1. The robot's acceleration setting is too small, resulting in a long downtime. 2. The robot is moving while switching mode.	1. Check that the motion parameters are set correctly; stop the current motion and make sure the robot is stationary before switching mode. 2. Stop the current motion and make sure the robot is stationary before switching mode.
0x2083	Power error	1. The input power supply is abnormal. 2. System hardware failure.	View the servo parameter H0B-45 and handle the problem according to the user guide.
0x2084	Fan error	1. The fan is not connected. 2. The fan cable has poor contact or broken wire.	View the servo parameter H0B-45 and handle the problem according to the user guide.
0x2085	Discharge tube shorted or regenerative resistor not connected	1. The regenerative resistor is not connected. 2. The discharge tube is shorted.	View the servo parameter H0B-45 and handle the problem according to the user guide.
0x2086	Discharge tube open circuit	Discharge tube open circuit	View the servo parameter H0B-45 and handle the problem according to the user guide.
0x2091	Trajectory recovery exception	1. Exception in line number processing for trajectory recovery. 2. Exception in data communication for trajectory recovery.	1. Reset the line number. 2. Save the error message and contact the manufacturer.
0x2092	Trajectory recovery target point error	1. Exception in line number processing for trajectory recovery. 2. Exception in data	1. Reset the line number. 2. Save the error message and contact the manufacturer.

		communication for trajectory recovery.	
0x2093	Trajectory recovery motion state error	1. Exception in line number processing for trajectory recovery. 2. Exception in data communication for trajectory recovery.	1. Reset the line number. 2. Save the error message and contact the manufacturer.
0x20A1	Data acquisition board connection failure	The data acquisition board is damaged; the data acquisition board cable is disconnected.	Check that the data acquisition board is installed; check that the data acquisition board is properly connected (flashing green); check that the cable and plug connections are intact; replace the data acquisition board.
0x20A2	Data acquisition board data error	The data collection board is damaged.	Check that the data acquisition board is properly connected (flashing green); replace the data acquisition board.
0x20A3	Data acquisition board communication error	The data acquisition board encounters data frame loss.	Check that the data acquisition board cable and plug are well connected; check whether the robot motion may cause the data acquisition board plug to have poor contact under stress.
0x20A4	Data acquisition board data changed significantly	Electromagnetic interference; robot collides with the environment; robot moves with abnormal sound.	Check the environment for severe electromagnetic interference; reduce the acceleration of motion; replace the data acquisition board; check if the robot collides with the environment.
0x20A5	Data acquisition board overtemperature	Data acquisition board overtemperature	Check that the temperature of the robot's working environment meets the requirements; reduce the speed and acceleration of the robot.
0x20A6	Excessive robot vibration	Excessive robot vibration	Check the robot for visible vibrations; check the robot joints for obvious transmission clearance; reduce the acceleration or speed of motion.
0x20A7	Robot positioning time too long	Residual robot vibration is significant, resulting in long positioning time.	Check the robot for visible vibrations; check the robot joints for obvious transmission

			clearance; increase the arrival error threshold; reduce the acceleration or speed of motion.
0x20A8	Collision detected	Robot collides with environment; robot moves with abnormal sound..	Check whether the robot collides with the environment; check whether there are abnormal noises during the robot motion.
0x20A9	Vibration suppression failure	Vibration suppression failure	Check that the data acquisition board cable is connected properly. If the connection is normal and the alarm persists after re-power, turn off the vibration suppression function.
0x20AA	Robot motion state error	The robot model in software is not consistent with actual one; mechanical anomalies such as collision, loose or stuck drive mechanism; excessive zero deviation of the robot; joint reduction ratio set incorrectly; reverse direction of joint movement; wrong installation direction of data acquisition board.	Check in the following order: Make sure the robot model in software is the same as with the actual one; check for mechanical anomalies such as collision, loose or stuck drive mechanism; check that the zero point of the robot is accurate; check that the reduction ratio parameter is correctly set; check that the direction of joint movement is correct; check that the data acquisition board is mounted in the correct direction. If the alarm persists, turn off the data acquisition board alarm function.
0x20AB	Self-learning vibration suppression calculation error	Self-learning vibration suppression calculation error	Reduce the motion speed; turn off the self-learning vibration suppression function.
0x20AC	Self-learning vibration suppression learning timeout	Self-learning vibration suppression learning timeout;	1. Change the self-learning flag in the motor instruction to SLOff. 2. Contact the manufacturer.
0x20B1	Tool load mass exceeds limit	The mass setting of the tool load exceeds the limit.	1. Reduce the mass setting of the tool load to the limit range.
0x20B2	Tool load centroid position exceeds limit	The centroid position setting of the tool load exceeds the limit.	1. Reduce the centroid position setting of the tool load to the limit range.
0x20B3	Tool load centroid pose exceeds limit	The centroid pose setting of the tool load exceeds the limit.	1. Reduce the centroid pose setting of the tool load to the limit range.
0x20B4	Tool load inertia exceeds limit	The inertia setting of the tool	1. Reduce the inertia setting of the

		load exceeds the limit.	tool load to the limit range.
0x20B5	Workobject load mass exceeds the limit	The mass setting of the workobject load exceeds the limit.	1. Reduce the mass setting of the workobject load to the limit range.
0x20B6	Workobject load centroid position exceeds limit	The centroid position setting of the workobject load exceeds the limit.	1. Reduce the centroid position setting of the workobject load to the limit range.
0x20B7	Workobject load centroid pose exceeds limit	The centroid pose setting of the workobject load exceeds the limit.	1. Reduce the centroid pose setting of the workobject load to the limit range.
0x20B8	Workobject load inertia exceeds limit	The inertia setting of the workobject load exceeds the limit.	1. Reduce the inertia setting of the workobject load to the limit range.
0x20B9	Arm load mass exceeds the limit	The mass setting of the arm load exceeds the limit.	1. Reduce the mass setting of the arm load to the limit range.
0x20BA	Arm load centroid position exceeds limit	The centroid position setting of the arm load exceeds the limit.	1. Reduce the centroid position setting of the arm load to the limit range.
0x20BB	Arm load centroid pose exceeds limit	The centroid pose setting of the arm load exceeds the limit.	1. Reduce the centroid pose setting of the arm load to the limit range.
0x20BC	Arm load inertia exceeds limit	The inertia setting of the arm load exceeds the limit.	1. Reduce the inertia setting of the arm load to the limit range.
0x20C1	Collision detected on J1 axis	Collision occurs or the motor of the corresponding axis is stuck, or the brake is not opened.	Check whether a collision has occurred, if not: 1) Check that the load parameters are set correctly; 2) Check that the robot model in the controller matches the actual robot; 3) Check the robot for motor jam, brake not opened, etc.; 4) If the robot operates at high speed and heavy load, it will cause current saturation phenomenon, which is prone to false alarms, so prevent the robot from operating under such conditions; 5) If it is determined to be a false alarm, the collision detection sensitivity of the axis can be appropriately increased; 6) If the alarm persists, turn off the collision detection switch for that axis.
0x20C2	Collision detected on J2 axis	Collision occurs or the motor of	Check whether a collision has

		the corresponding axis is stuck, or the brake is not opened.	occurred, if not: 1) Check that the load parameters are set correctly; 2) Check that the robot model in the controller matches the actual robot; 3) Check the robot for motor jam, brake not opened, etc.; 4) If the robot operates at high speed and heavy load, it will cause current saturation phenomenon, which is prone to false alarms, so prevent the robot from operating under such conditions; 5) If it is determined to be a false alarm, the collision detection sensitivity of the axis can be appropriately increased; 6) If the alarm persists, turn off the collision detection switch for that axis.
0x20C3	Collision detected on J3 axis	Collision occurs or the motor of the corresponding axis is stuck, or the brake is not opened.	Check whether a collision has occurred, if not: 1) Check that the load parameters are set correctly; 2) Check that the robot model in the controller matches the actual robot; 3) Check the robot for motor jam, brake not opened, etc.; 4) If the robot operates at high speed and heavy load, it will cause current saturation phenomenon, which is prone to false alarms, so prevent the robot from operating under such conditions; 5) If it is determined to be a false alarm, the collision detection sensitivity of the axis can be appropriately increased; 6) If the alarm persists, turn off the collision detection switch for that axis.
0x20C4	Collision detected on J4 axis	Collision occurs or the motor of the corresponding axis is stuck, or the brake is not opened.	Check whether a collision has occurred, if not: 1) Check that the load parameters are set correctly; 2) Check that the robot model in the controller matches the actual

			robot; 3) Check the robot for motor jam, brake not opened, etc.; 4) If the robot operates at high speed and heavy load, it will cause current saturation phenomenon, which is prone to false alarms, so prevent the robot from operating under such conditions; 5) If it is determined to be a false alarm, the collision detection sensitivity of the axis can be appropriately increased; 6) If the alarm persists, turn off the collision detection switch for that axis.
0x20C5	Collision detected on J5 axis	Collision occurs or the motor of the corresponding axis is stuck, or the brake is not opened.	Check whether a collision has occurred, if not: 1) Check that the load parameters are set correctly; 2) Check that the robot model in the controller matches the actual robot; 3) Check the robot for motor jam, brake not opened, etc.; 4) If the robot operates at high speed and heavy load, it will cause current saturation phenomenon, which is prone to false alarms, so prevent the robot from operating under such conditions; 5) If it is determined to be a false alarm, the collision detection sensitivity of the axis can be appropriately increased; 6) If the alarm persists, turn off the collision detection switch for that axis.
0x20C6	Collision detected on J6 axis	Collision occurs or the motor of the corresponding axis is stuck, or the brake is not opened.	Check whether a collision has occurred, if not: 1) Check that the load parameters are set correctly; 2) Check that the robot model in the controller matches the actual robot; 3) Check the robot for motor jam, brake not opened, etc.; 4) If the robot operates at high speed and heavy load, it will

			cause current saturation phenomenon, which is prone to false alarms, so prevent the robot from operating under such conditions; 5) If it is determined to be a false alarm, the collision detection sensitivity of the axis can be appropriately increased; 6) If the alarm persists, turn off the collision detection switch for that axis.
0x20D1	Motion state error detected on J1 axis	1. Robot collision; or abnormal current due to serious errors in load parameters, model parameters, and serious deviation of zero point. 2. Servo faults, including: power line UVW phase sequence error, motor angle error, encoder model mismatch, encoder wiring failure, servo gain mismatch, etc.	When troubleshooting, always make sure that the person is outside the robot's operating range before activating the motor on any axis! 1. First check if a violent collision has occurred, if no collision occurs, check that the load parameters are set correctly, check that the robot matches the model displayed in the controller, and check if there is a serious deviation from the zero point of the robot. 2. If the collision and controller side factors are excluded, then investigate the servo side factors in order. 1) The UVW phase sequence is incorrect. Connect the U/V/W cables in the correct phase sequence. 2) An error occurs on the initial phase detection of the motor rotor due to disturbing signals upon power-on. Power on the system again. 3. The encoder model is wrong or the encoder is wired improperly. Confirm the motor model, the parameter H00-00, the encoder type, and the encoder wiring are correct.

			4. The encoder is wired improperly, aged, or connected loosely. Re-solder, tighten or replace the encoder cable. 5. Improper parameter setting leads to excessive vibration. Set the parameters appropriately to avoid excessive vibration. For more information, see Appendix "Robot Alarms and Handling Method".
0x20D2	Motion state error detected on J2 axis	1. Robot collision; or abnormal current due to serious errors in load parameters, model parameters, and serious deviation of zero point. 2. Servo faults, including: power line UVW phase sequence error, motor angle error, encoder model mismatch, encoder wiring failure, servo gain mismatch, etc.	When troubleshooting, always make sure that the person is outside the robot's operating range before activating the motor on any axis! 1. First check if a violent collision has occurred, if no collision occurs, check that the load parameters are set correctly, check that the robot matches the model displayed in the controller, and check if there is a serious deviation from the zero point of the robot. 2. If the collision and controller side factors are excluded, then investigate the servo side factors in order. 1) The UVW phase sequence is incorrect. Connect the U/V/W cables in the correct phase sequence. 2) An error occurs on the initial phase detection of the motor rotor due to disturbing signals upon power-on. Power on the system again. 3. The encoder model is wrong or the encoder is wired improperly. Confirm the motor model, the parameter H00-00, the encoder type, and the encoder wiring are correct.

			4. The encoder is wired improperly, aged, or connected loosely. Re-solder, tighten or replace the encoder cable. 5. Improper parameter setting leads to excessive vibration. Set the parameters appropriately to avoid excessive vibration. For more information, see Appendix "Robot Alarms and Handling Method".
0x20D3	Motion state error detected on J3 axis	1. Robot collision; or abnormal current due to serious errors in load parameters, model parameters, and serious deviation of zero point. 2. Servo faults, including: power line UVW phase sequence error, motor angle error, encoder model mismatch, encoder wiring failure, servo gain mismatch, etc.	When troubleshooting, always make sure that the person is outside the robot's operating range before activating the motor on any axis! 1. First check if a violent collision has occurred, if no collision occurs, check that the load parameters are set correctly, check that the robot matches the model displayed in the controller, and check if there is a serious deviation from the zero point of the robot. 2. If the collision and controller side factors are excluded, then investigate the servo side factors in order. 1) The UVW phase sequence is incorrect. Connect the U/V/W cables in the correct phase sequence. 2) An error occurs on the initial phase detection of the motor rotor due to disturbing signals upon power-on. Power on the system again. 3. The encoder model is wrong or the encoder is wired improperly. Confirm the motor model, the parameter H00-00, the encoder type, and the encoder wiring are correct.

			4. The encoder is wired improperly, aged, or connected loosely. Re-solder, tighten or replace the encoder cable. 5. Improper parameter setting leads to excessive vibration. Set the parameters appropriately to avoid excessive vibration. For more information, see Appendix "Robot Alarms and Handling Method".
0x20D4	Motion state error detected on J4 axis	1. Robot collision; or abnormal current due to serious errors in load parameters, model parameters, and serious deviation of zero point. 2. Servo faults, including: power line UVW phase sequence error, motor angle error, encoder model mismatch, encoder wiring failure, servo gain mismatch, etc.	When troubleshooting, always make sure that the person is outside the robot's operating range before activating the motor on any axis! 1. First check if a violent collision has occurred, if no collision occurs, check that the load parameters are set correctly, check that the robot matches the model displayed in the controller, and check if there is a serious deviation from the zero point of the robot. 2. If the collision and controller side factors are excluded, then investigate the servo side factors in order. 1) The UVW phase sequence is incorrect. Connect the U/V/W cables in the correct phase sequence. 2) An error occurs on the initial phase detection of the motor rotor due to disturbing signals upon power-on. Power on the system again. 3. The encoder model is wrong or the encoder is wired improperly. Confirm the motor model, the parameter H00-00, the encoder type, and the encoder wiring are correct.

			4. The encoder is wired improperly, aged, or connected loosely. Re-solder, tighten or replace the encoder cable. 5. Improper parameter setting leads to excessive vibration. Set the parameters appropriately to avoid excessive vibration. For more information, see Appendix "Robot Alarms and Handling Method".
0x20D5	Motion state error detected on J5 axis	1. Robot collision; or abnormal current due to serious errors in load parameters, model parameters, and serious deviation of zero point. 2. Servo faults, including: power line UVW phase sequence error, motor angle error, encoder model mismatch, encoder wiring failure, servo gain mismatch, etc.	When troubleshooting, always make sure that the person is outside the robot's operating range before activating the motor on any axis! 1. First check if a violent collision has occurred, if no collision occurs, check that the load parameters are set correctly, check that the robot matches the model displayed in the controller, and check if there is a serious deviation from the zero point of the robot. 2. If the collision and controller side factors are excluded, then investigate the servo side factors in order. 1) The UVW phase sequence is incorrect. Connect the U/V/W cables in the correct phase sequence. 2) An error occurs on the initial phase detection of the motor rotor due to disturbing signals upon power-on. Power on the system again. 3. The encoder model is wrong or the encoder is wired improperly. Confirm the motor model, the parameter H00-00, the encoder type, and the encoder wiring are correct.

			4. The encoder is wired improperly, aged, or connected loosely. Re-solder, tighten or replace the encoder cable. 5. Improper parameter setting leads to excessive vibration. Set the parameters appropriately to avoid excessive vibration. For more information, see Appendix "Robot Alarms and Handling Method".
0x20D6	Motion state error detected on J6 axis	1. Robot collision; or abnormal current due to serious errors in load parameters, model parameters, and serious deviation of zero point. 2. Servo faults, including: power line UVW phase sequence error, motor angle error, encoder model mismatch, encoder wiring failure, servo gain mismatch, etc.	When troubleshooting, always make sure that the person is outside the robot's operating range before activating the motor on any axis! 1. First check if a violent collision has occurred, if no collision occurs, check that the load parameters are set correctly, check that the robot matches the model displayed in the controller, and check if there is a serious deviation from the zero point of the robot. 2. If the collision and controller side factors are excluded, then investigate the servo side factors in order. 1) The UVW phase sequence is incorrect. Connect the U/V/W cables in the correct phase sequence. 2) An error occurs on the initial phase detection of the motor rotor due to disturbing signals upon power-on. Power on the system again. 3. The encoder model is wrong or the encoder is wired improperly. Confirm the motor model, the parameter H00-00, the encoder type, and the encoder wiring are correct.

			4. The encoder is wired improperly, aged, or connected loosely. Re-solder, tighten or replace the encoder cable. 5. Improper parameter setting leads to excessive vibration. Set the parameters appropriately to avoid excessive vibration. For more information, see Appendix "Robot Alarms and Handling Method".
0x2101	J1 axis positive limit alarm	The joint limit position is reached.	Perform negative motion to clear the alarm.
0x2102	J1 axis negative limit alarm	The joint limit position is reached.	Perform positive motion to clear the alarm.
0x2103	J1 axis drive alarm	The drive has an alarm.	Perform troubleshooting by referring to the servo manual according to fault codes.
0x2104	J1 axis path planning out of range	The planned value exceeds the maximum calculation range (-1073741823 to 1073741824).	Check whether the absolute origin position is selected to be near the counting limit. If the zero point is near the limit, clear the drive turns.
0x2105	Excessive tracking error of J1 axis	The difference between the planned and actual positions is too large.	(1) View the tracking error threshold. (2) Check for mechanical stuck. (3) Check that the power lines are properly connected. (4) Check that the load matches the servo gain. (5) Consult the troubleshooting guide.
0x2106	J1 axis overspeed	The running speed is greater than the set maximum speed.	Reduce the speed of the J1 axis.
0x2107	Strong current of J1 axis is not switched on.	Strong current is not switched on.	Check the drive circuit for whether strong current is not switched on.
0x2108	J1 axis torque overlimit	The actual maximum current exceeds the limit	Check whether speed and acceleration of the J1 axis are too large.
0x2109	J1 axis average load rate overlimit	The average load rate exceeds the limit.	Check whether speed and acceleration of the J1 axis are too large. Troubleshoot the brake, mechanical parts and drive.

0x210A	J1 axis speed overlimit	When verifying the planned output, it was found that the set joint speed was exceeded.	(1) In case of CP motion, reduce position speed or increase joint speed appropriately, or adjust the points to move away from singularities. (2) Check if there are any other accompanying alarms in the alarm record.
0x210B	J1 axis acceleration overlimit	When verifying the planned output, it was found that the set joint acceleration was exceeded.	(1) In case of CP motion, reduce position speed or increase joint speed appropriately, or adjust the points to move away from singularities. (2) Check if there are any other accompanying alarms in the alarm record.
0x210C	J1 axis enable synchronization error	The robot vibrates before it is enabled, or the robot moves due to external reasons during the enabling process.	Check whether there is joint movement during power on, and whether there is external interference during the enabling process.
0x210D	J1 axis arrival timeout	The servo takes too long to arrive the target position.	Check whether the arrival error threshold is too small or whether the servo gain is not proper.
0x210E	J1 axis position feedback error	(1) Multi-turn value changes. (2) Encoder position jumps.	Check if the multi turn value has been manually cleared. If not, check for the interference source.
0x210F	J1 axis disable status error	The robot failed to be disabled at the specified time and distance.	1. Check that servo parameter 6084 and brake parameter 0209 are correctly set. 2. Check if the load exceeds the limit.
0x2111	J2 axis positive limit alarm	The joint limit position is reached.	Perform negative motion to clear the alarm.
0x2112	J2 axis negative limit alarm	The joint limit position is reached.	Perform positive motion to clear the alarm.
0x2113	J2 axis drive alarm	The drive has an alarm.	Perform troubleshooting by referring to the servo manual according to fault codes.
0x2114	J2 axis path planning out of range	The planned value exceeds the maximum calculation range.	Check whether the absolute origin position is selected to be near the counting limit. If the zero point is near the limit, clear the drive turns.

0x2115	Excessive tracking error of J2 axis	The difference between the planned and actual positions is too large.	(1) View the tracking error threshold. (2) Check for mechanical stuck. (3) Check that the power lines are properly connected. (4) Check that the load matches the servo gain. (5) Consult the troubleshooting guide.
0x2116	J2 axis overspeed	The running speed is greater than the set maximum speed.	Reduce the speed of J2 axis.
0x2117	Strong current of J2 axis is not switched on.	Strong current is not switched on.	Check the drive circuit for whether strong current is not switched on.
0x2118	J2 axis torque overlimit	The actual maximum current exceeds the limit	Check whether the speed or acceleration is too large.
0x2119	J2 axis average load rate overlimit	The average load rate exceeds the limit.	Check whether the speed or acceleration is too large.
0x211A	J2 axis speed overlimit	When verifying the planned output, it was found that the set joint acceleration was exceeded.	(1) In case of CP motion, reduce position speed or increase joint speed appropriately, or adjust the points to move away from singularities. (2) Check if there are any other accompanying alarms in the alarm record.
0x211B	J2 axis acceleration overlimit	When verifying the planned output, it was found that the set joint acceleration was exceeded.	(1) In case of CP motion, reduce position speed or increase joint speed appropriately, or adjust the points to move away from singularities. (2) Check if there are any other accompanying alarms in the alarm record.
0x211C	J2 axis enable synchronization error	The robot vibrates before it is enabled, or the robot moves due to external reasons during the enabling process.	Check whether there is joint movement during power on, and whether there is external interference during the enabling process.
0x211D	J2 axis arrival timeout	The servo takes too long to arrive the target position.	Check whether the arrival error threshold is too small or whether the servo gain is not proper.
0x211E	J2 axis position feedback error	(1) Multi-turn value changes. (2) Encoder position jumps.	Check if the multi turn value has been manually cleared. If not,

			check for the interference source.
0x211F	J2 axis disable status error	The robot failed to be disabled at the specified time and distance.	1. Check that servo parameter 6084 and brake parameter 0209 are correctly set. 2. Check if the load exceeds the limit.
0x2201	J3 axis positive limit error	The joint limit position is reached.	Perform negative motion to clear the alarm.
0x2202	J3 axis negative limit alarm	The joint limit position is reached.	Perform positive motion to clear the alarm.
0x2203	J3 axis drive alarm	The drive has an alarm.	Perform troubleshooting by referring to the servo manual according to fault codes.
0x2204	J3 axis path planning out of range	The planned value exceeds the maximum calculation range (-1073741823 to 1073741824).	Check whether the absolute origin position is selected to be near the counting limit. If the zero point is near the limit, clear the drive turns.
0x2205	Excessive tracking error of J3 axis	The difference between the planned and actual positions is too large.	(1) View the tracking error threshold. (2) Check for mechanical stuck. (3) Check that the power lines are properly connected. (4) Check that the load matches the servo gain. (5) Consult the troubleshooting guide.
0x2206	J3 axis overspeed	The running speed is greater than the set maximum speed.	Reduce the speed of J3 axis.
0x2207	Strong current of J3 axis is not switched on.	Strong current is not switched on.	Check the drive circuit for whether strong current is not switched on.
0x2208	J3 axis torque overlimit	The actual maximum current exceeds the limit	Check whether the speed or acceleration is too large.
0x2209	J3 axis average load rate overlimit	The average load rate exceeds the limit.	Check whether speed and acceleration are too large. Troubleshoot the brake, mechanical parts and drive.
0x220A	J3 axis speed overlimit	When verifying the planned output, it was found that the set joint acceleration was exceeded.	(1) In case of CP motion, reduce position speed or increase joint speed appropriately, or adjust the points to move away from singularities. (2) Check if there are any other

			accompanying alarms in the alarm record.
0x220B	J3 axis acceleration overlimit	When verifying the planned output, it was found that the set joint acceleration was exceeded.	(1) In case of CP motion, reduce position speed or increase joint speed appropriately, or adjust the points to move away from singularities. (2) Check if there are any other accompanying alarms in the alarm record.
0x220C	J3 axis enable synchronization error	The robot vibrates before it is enabled, or the robot moves due to external reasons during the enabling process.	Check whether there is joint movement during power on, and whether there is external interference during the enabling process.
0x220D	J3 axis arrival timeout	The servo takes too long to arrive the target position.	Check whether the arrival error threshold is too small or whether the servo gain is not proper.
0x220E	J3 axis position feedback error	(1) Multi-turn value changes. (2) Encoder position jumps.	Check if the multi turn value has been manually cleared. If not, check for the interference source.
0x220F	J3 axis disable status error	The robot failed to be disabled at the specified time and distance.	1. Check that servo parameter 6084 and brake parameter 0209 are correctly set. 2. Check if the load exceeds the limit.
0x2211	J4 axis positive limit alarm	The joint limit position is reached.	Perform negative motion to clear the alarm.
0x2212	J4 axis negative limit alarm	The joint limit position is reached.	Perform negative motion to clear the alarm.
0x2213	J4 axis drive alarm	The drive has an alarm.	Perform troubleshooting by referring to the servo manual according to fault codes.
0x2214	J4 axis path planning out of range	The planned value exceeds the maximum calculation range (-1073741823 to 1073741824).	Check whether the absolute origin position is selected to be near the counting limit. If the zero point is near the limit, clear the drive turns.
0x2215	Excessive tracking error of J4 axis	The difference between the planned and actual positions is too large.	(1) View the tracking error threshold. (2) Check for mechanical stuck. (3) Check that the power lines are properly connected. (4) Check that the load matches

			the servo gain. (5) Consult the troubleshooting guide.
0x2216	J4 axis overspeed	The running speed is greater than the set maximum speed.	Reduce the speed of J4 axis.
0x2217	Strong current of J4 axis is not switched on.	Strong current is not switched on.	Check the drive circuit for whether strong current is not switched on.
0x2218	J4 axis torque overlimit	The actual maximum current exceeds the limit	Check whether the speed or acceleration is too large.
0x2219	J4 axis average load rate overlimit	The average load rate exceeds the limit.	Check whether speed and acceleration are too large. Troubleshoot the brake, mechanical parts and drive.
0x221A	J4 axis speed overlimit	When verifying the planned output, it was found that the set joint acceleration was exceeded.	(1) In case of CP motion, reduce position speed or increase joint speed appropriately, or adjust the points to move away from singularities. (2) Check if there are any other accompanying alarms in the alarm record.
0x221B	J4 axis acceleration overlimit	When verifying the planned output, it was found that the set joint acceleration was exceeded.	(1) In case of CP motion, reduce position speed or increase joint speed appropriately, or adjust the points to move away from singularities. (2) Check if there are any other accompanying alarms in the alarm record.
0x221C	J4 axis enable synchronization error	The robot vibrates before it is enabled, or the robot moves due to external reasons during the enabling process.	Check whether there is joint movement during power on, and whether there is external interference during the enabling process.
0x221D	J4 axis arrival timeout	The servo takes too long to arrive the target position.	Check whether the arrival error threshold is too small or whether the servo gain is not proper.
0x221E	J4 axis position feedback error	(1) Multi-turn value changes. (2) Encoder position jumps.	Check if the multi turn value has been manually cleared. If not, check for the interference source.
0x221F	J4 axis disable status error	The robot failed to be disabled at the specified time and distance.	1. Check that servo parameter 6084 and brake parameter 0209 are correctly set.

			2. Check if the load exceeds the limit.
0x2301	J5 axis positive limit alarm	The joint limit position is reached.	Perform negative motion to clear the alarm.
0x2302	J5 axis negative limit alarm	The joint limit position is reached.	Perform positive motion to clear the alarm.
0x2303	J5 axis drive alarm	The drive has an alarm.	Perform troubleshooting by referring to the servo manual according to fault codes.
0x2304	J5 axis path planning out of range	The planned value exceeds the maximum calculation range (-1073741823 to 1073741824).	Check whether the absolute origin position is selected to be near the counting limit. If the zero point is near the limit, clear the drive turns.
0x2305	Excessive tracking error of J5 axis	The difference between the planned and actual positions is too large.	(1) View the tracking error threshold. (2) Check for mechanical stuck. (3) Check that the power lines are properly connected. (4) Check that the load matches the servo gain. (5) Consult the troubleshooting guide.
0x2306	J5 axis overspeed	The running speed is greater than the set maximum speed.	Reduce the speed of J5 axis.
0x2307	Strong current of J5 axis is not switched on.	Strong current is not switched on.	Check the drive circuit for whether strong current is not switched on.
0x2308	J5 axis torque overlimit	The actual maximum current exceeds the limit	Check whether the speed or acceleration is too large.
0x2309	J5 axis average load rate overlimit	The average load rate exceeds the limit.	Check whether speed and acceleration are too large. Troubleshoot the brake, mechanical parts and drive.
0x230A	J5 axis speed overlimit	When verifying the planned output, it was found that the set joint acceleration was exceeded.	(1) In case of CP motion, reduce position speed or increase joint speed appropriately, or adjust the points to move away from singularities. (2) Check if there are any other accompanying alarms in the alarm record.
0x230B	J5 axis acceleration overlimit	When verifying the planned output, it was found that the set	(1) In case of CP motion, reduce position speed or increase joint

		joint acceleration was exceeded.	speed appropriately, or adjust the points to move away from singularities. (2) Check if there are any other accompanying alarms in the alarm record.
0x230C	J5 axis enable synchronization error	The robot vibrates before it is enabled, or the robot moves due to external reasons during the enabling process.	Check whether there is joint movement during power on, and whether there is external interference during the enabling process.
0x230D	J5 axis arrival timeout	The servo takes too long to arrive the target position.	Check whether the arrival error threshold is too small or whether the servo gain is not proper.
0x230E	J5 axis position feedback error	(1) Multi-turn value changes. (2) Encoder position jumps.	Check if the multi turn value has been manually cleared. If not, check for the interference source.
0x230F	J5 axis disable status error	The robot failed to be disabled at the specified time and distance.	1. Check that servo parameter 6084 and brake parameter 0209 are correctly set. 2. Check if the load exceeds the limit.
0x2311	J6 axis positive limit alarm	The joint limit position is reached.	Perform negative motion to clear the alarm.
0x2312	J6 axis negative limit alarm	The joint limit position is reached.	Perform positive motion to clear the alarm.
0x2313	J6 axis drive alarm	The drive has an alarm.	Perform troubleshooting by referring to the servo manual according to fault codes.
0x2314	J6 axis path planning out of range	The planned value exceeds the maximum calculation range (-1073741823 to 1073741824).	Check whether the absolute origin position is selected to be near the counting limit. If the zero point is near the limit, clear the drive turns.
0x2315	Excessive tracking error of J6 axis	The difference between the planned and actual positions is too large.	(1) View the tracking error threshold. (2) Check for mechanical stuck. (3) Check that the power lines are properly connected. (4) Check that the load matches the servo gain. (5) Consult the troubleshooting guide.
0x2316	J6 axis overspeed	The running speed is greater	Reduce the speed of J6 axis.

		than the set maximum speed.	
0x2317	Strong current of J6 axis is not switched on.	Strong current is not switched on.	Check the drive circuit for whether strong current is not switched on.
0x2318	J6 axis torque overlimit	The actual maximum current exceeds the limit	Check whether the speed or acceleration is too large.
0x2319	J6 axis average load rate overlimit	The average load rate exceeds the limit.	Check whether speed and acceleration are too large. Troubleshoot the brake, mechanical parts and drive.
0x231A	J6 axis speed overlimit	When verifying the planned output, it was found that the set joint acceleration was exceeded.	(1) In case of CP motion, reduce position speed or increase joint speed appropriately, or adjust the points to move away from singularities. (2) Check if there are any other accompanying alarms in the alarm record.
0x231B	J6 axis acceleration overlimit	When verifying the planned output, it was found that the set joint acceleration was exceeded.	(1) In case of CP motion, reduce position speed or increase joint speed appropriately, or adjust the points to move away from singularities. (2) Check if there are any other accompanying alarms in the alarm record.
0x231C	J6 axis enable synchronization error	Excessive tracking error	Check whether there is joint movement during power on.
0x231D	J6 axis arrival timeout	The servo takes too long to arrive the target position.	Check whether the arrival error threshold is too small or whether the servo gain is not proper.
0x231E	J6 axis position feedback error	(1) Multi-turn value changes. (2) Encoder position jumps.	Check if the multi turn value has been manually cleared. If not, check for the interference source.
0x231F	J6 axis disable status error	The robot failed to be disabled at the specified time and distance.	1. Check that servo parameter 6084 and brake parameter 0209 are correctly set. 2. Check if the load exceeds the limit.
0x2401	J7 axis positive limit alarm	The joint limit position is reached.	Perform negative motion to clear the alarm.
0x2402	J7 axis negative limit alarm	The joint limit position is reached.	Perform positive motion to clear the alarm.
0x2403	J7 axis drive alarm	The drive has an alarm.	Perform troubleshooting by

			referring to the servo manual according to fault codes.
0x2404	J7 axis path planning out of range	The planned value exceeds the maximum calculation range (-1073741823 to 1073741824).	Check whether the absolute origin position is selected to be near the counting limit. If the zero point is near the limit, clear the drive turns.
0x2405	Excessive tracking error of J7 axis	The difference between the planned and actual positions is too large.	(1) View the tracking error threshold. (2) Check for mechanical stuck. (3) Check that the power lines are properly connected. (4) Check that the load matches the servo gain. (5) Consult the troubleshooting guide.
0x2406	J7 axis overspeed	The running speed is greater than the set maximum speed.	Reduce the speed of J7 axis.
0x2407	Strong current of J7 axis is not switched on.	Strong current is not switched on.	Check the drive circuit for whether strong current is not switched on.
0x2408	J7 axis torque overlimit	The actual maximum current exceeds the limit	Check whether the speed or acceleration is too large.
0x2409	J7 axis average load rate overlimit	The average load rate exceeds the limit.	Check whether speed and acceleration are too large. Troubleshoot the brake, mechanical parts and drive.
0x240A	J7 axis planned speed error	The planned speed exceeds the limit.	Reduce the maximum speed in joint or Cartesian space.
0x240B	J7 axis planned acceleration error	The planned acceleration exceeds the limit.	Reduce the maximum acceleration or speed in joint or Cartesian space.
0x240C	J7 axis enable synchronization error	The robot vibrates before it is enabled, or the robot moves due to external reasons during the enabling process.	Check whether there is joint movement during power on, and whether there is external interference during the enabling process.
0x240D	J7 axis arrival timeout	The servo takes too long to arrive the target position.	Check whether the arrival error threshold is too small or whether the servo gain is not proper.
0x240E	J7 axis position feedback error	(1) Multi-turn value changes. (2) Encoder position jumps.	Check if the multi turn value has been manually cleared. If not, check for the interference source.
0x240F	J7 axis disable status error	The robot failed to be disabled	1. Check that servo parameter

		at the specified time and distance.	6084 and brake parameter 0209 are correctly set. 2. Check if the load exceeds the limit.
0x2411	J8 axis positive limit alarm	The joint limit position is reached.	Perform negative motion to clear the alarm.
0x2412	J8 axis negative limit alarm	The joint limit position is reached.	Perform positive motion to clear the alarm.
0x2413	J8 axis drive alarm	The drive has an alarm.	Perform troubleshooting by referring to the servo manual according to fault codes.
0x2414	J8 axis path planning out of range	The planned value exceeds the maximum calculation range (-1073741823 to 1073741824).	Check whether the absolute origin position is selected to be near the counting limit. If the zero point is near the limit, clear the drive turns.
0x2415	Excessive tracking error of J8 axis	The difference between the planned and actual positions is too large.	(1) View the tracking error threshold. (2) Check for mechanical stuck. (3) Check that the power lines are properly connected. (4) Check that the load matches the servo gain. (5) Consult the troubleshooting guide.
0x2416	J8 axis overspeed	The running speed is greater than the set maximum speed.	Reduce the joint speed.
0x2417	Strong current of J8 axis is not switched on.	Strong current is not switched on.	Check the drive circuit for whether strong current is not switched on.
0x2418	J8 axis torque overlimit	The actual maximum current exceeds the limit	Check whether the speed or acceleration is too large.
0x2419	J8 axis average load rate overlimit	The average load rate exceeds the limit.	Check whether speed and acceleration are too large. Troubleshoot the brake, mechanical parts and drive.
0x241A	J8 axis planned speed error	The planned speed exceeds the limit.	Reduce the maximum speed in joint or Cartesian space.
0x241B	J8 axis planned acceleration error	The planned acceleration exceeds the limit.	Reduce the maximum acceleration in joint or Cartesian space.
0x241C	J8 axis enable synchronization error	The robot vibrates before it is enabled, or the robot moves due to external reasons during the	Check whether there is joint movement during power on, and whether there is external

		enabling process.	interference during the enabling process.
0x241D	J8 axis arrival timeout	The servo takes too long to arrive the target position.	Check whether the arrival error threshold is too small or whether the servo gain is not proper.
0x241E	J8 axis position feedback error	(1) Multi-turn value changes. (2) Encoder position jumps.	Check if the multi turn value has been manually cleared. If not, check for the interference source.
0x241F	J8 axis disable status error	The robot failed to be disabled at the specified time and distance.	1. Check that servo parameter 6084 and brake parameter 0209 are correctly set. 2. Check if the load exceeds the limit.
0x3000	Multiple fieldbuses active at the same time	Multiple fieldbuses active at the same time	Keep only one fieldbus active.
0x3001	Ethernet/IP connection actively disconnected	The Ethernet/IP connection is disconnected by client or server.	Re-initiate the Ethernet/IP connection.
0x3002	Ethernet/IP connection down due to network timeout	The network communication of Ethernet/IP connection is not available.	Check whether the network cable is plugged in properly or in poor contact.
0x3003	EtherCAT (fieldbus) disconnected from master	The network cable of the EtherCAT connection is loose, the EtherCAT master is faulty, or the robot system is faulty.	Check if the network cable is plugged in or not in good contact, check the EtherCAT master, or contact the manufacturer.
0x3032	EtherCAT (fieldbus) communication synchronization failed	The synchronization signal is not generated due to hardware errors.	Contact the manufacturer.
0x3033	EtherCAT (fieldbus) communication IRQ loss overlimit	1. The data frame has been lost or discarded at an upstream station. 2. The performance of the host station is poor, the jitter of the IRQ exceeds the set value (H0E-22) * communication cycle.	1. Check the CPU usage of the master. 2. Reduce the communication time. 3. Check whether link loss occurs on the upstream slave.
0x3034	EtherCAT (fieldbus) EEPROM loading error	During program start-up, the first 8 bytes of data in the EEPROM were wrong, causing EtherCAT slave to fail to start.	Contact the manufacturer (Re-burn the XML file).
0x3035	EtherCAT (fieldbus) initialization error	EtherCAT (fieldbus) initialization error	Hardware failure, please contact the manufacturer.
0x3036	EtherCAT (fieldbus) state switching error	Bad state switching due to incorrect operation of the master or human error.	Contact the manufacturer.

0x3037	DC not enabled for EtherCAT (fieldbus) slave	In DC mode, there is no synchronization signal causing abnormal movement due to master fault or improper master operation	Contact the manufacturer.
0x3038	EtherCAT (fieldbus) PDO overlimit	EtherCAT (fieldbus) PDO overlimit	Check whether the number of PDOs configured for the master exceeds the limit.
0x3039	EtherCAT (fieldbus) link missing	The physical connection of the data link is unstable or the process data is lost due to plug-in/plug-out of the network cable.	Check whether the network cable is connected properly and whether the application site suffers from strong vibration.
0x303A	EtherCAT (fieldbus) link interfered	The data is lost due to EMC interference, poor quality of the network cable or improper connection.	Check whether proper grounding is performed and rectify EMC measures. Check whether the network cable used is provided by Inovance and whether the network cable is properly connected.
0x303B	EtherCAT (fieldbus) data forward error	The upstream station detects that the data frame has been corrupted, leading to a data transfer error.	Check the upstream station to locate the fault cause.
0x303C	EtherCAT (fieldbus) received no data	1. The data frame has been lost or discarded at an upstream station. 2. The performance of the host station is poor, the jitter of the IRQ exceeds the set value (H0E-22) * communication cycle.	1. Check the CPU usage of the master. 2. Reduce the communication time. 3. Check whether link loss occurs on the upstream slave.
0x3100	Failed to save configuration file	The system is busy or there is a problem with the file system.	1. Try to save again. 2. Contact the manufacturer.
0x3101	Failed to load DSP0 firmware	FPGA firmware error or startup abnormality	1. Restart the robot. 2. Contact the manufacturer.
0x3102	EtherCAT slave xml file loading failed	Xml file error or startup exception	1. Restart the robot. 2. Contact the manufacturer.
0x5001	J1 axis torque overlimit warning	The actual maximum current exceeds the limit for warning.	Check whether the speed or acceleration is too large.
0x5002	J1 axis average load rate overlimit warning	The average load rate exceeds the limit for warning.	Check whether speed and acceleration are too large. Troubleshoot the brake,

			mechanical parts and drive.
0x5003	J2 axis torque overlimit warning	The actual maximum current exceeds the limit for warning.	Check whether the speed or acceleration is too large.
0x5004	J2 axis average load rate overlimit warning	The average load rate exceeds the limit for warning.	Check whether speed and acceleration are too large. Troubleshoot the brake, mechanical parts and drive.
0x5005	J3 axis torque overlimit warning	The actual maximum current exceeds the limit for warning.	Check whether the speed or acceleration is too large.
0x5006	J3 axis average load rate overlimit warning	The average load rate exceeds the limit for warning.	Check whether speed and acceleration are too large. Troubleshoot the brake, mechanical parts and drive.
0x5007	J4 axis torque overlimit warning	The actual maximum current exceeds the limit for warning.	Check whether the speed or acceleration is too large.
0x5008	J4 axis average load rate overlimit warning	The average load rate exceeds the limit for warning.	Check whether speed and acceleration are too large. Troubleshoot the brake, mechanical parts and drive.
0x5009	J5 axis torque overlimit warning	The actual maximum current exceeds the limit for warning.	Check whether the speed or acceleration is too large.
0x500A	J5 axis average load rate overlimit warning	The average load rate exceeds the limit for warning.	Check whether speed and acceleration are too large. Troubleshoot the brake, mechanical parts and drive.
0x500B	J6 axis torque overlimit warning	The actual maximum current exceeds the limit for warning.	Check whether the speed or acceleration is too large.
0x500C	J6 axis average load rate overlimit warning	The average load rate exceeds the limit for warning.	Check whether speed and acceleration are too large. Troubleshoot the brake, mechanical parts and drive.
0x500D	J7 axis torque overlimit warning	The actual maximum current exceeds the limit for warning.	Check whether the speed or acceleration is too large.
0x500E	J7 axis average load rate overlimit warning	The average load rate exceeds the limit for warning.	Check whether speed and acceleration are too large. Troubleshoot the brake, mechanical parts and drive.
0x500F	J8 axis torque overlimit warning	The actual maximum current exceeds the limit for warning.	Check whether the speed or acceleration is too large.
0x5010	J8 axis average load rate overlimit warning	The average load rate exceeds the limit for warning.	Check whether speed and acceleration are too large. Troubleshoot the brake, mechanical parts and drive.
0x5011	J1 axis servo warning	Warning occurs on the servo of	View the parameter H0B-45 and

		J1 axis.	handle the problem according to the user guide.
0x5012	J2 axis servo warning	Warning occurs on the servo of J2 axis.	View the parameter H0B-45 and handle the problem according to the user guide.
0x5013	J3 axis servo warning	Warning occurs on the servo of J3 axis.	View the parameter H0B-45 and handle the problem according to the user guide.
0x5014	J4 axis servo warning	Warning occurs on the servo of J4 axis.	View the parameter H0B-45 and handle the problem according to the user guide.
0x5015	J5 axis servo warning	Warning occurs on the servo of J5 axis.	View the parameter H0B-45 and handle the problem according to the user guide.
0x5016	J6 axis servo warning	Warning occurs on the servo of J6 axis.	View the parameter H0B-45 and handle the problem according to the user guide.
0x5017	J7 axis servo warning	Warning occurs on the servo of J7 axis.	View the parameter H0B-45 and handle the problem according to the user guide.
0x5018	J8 axis servo warning	Warning occurs on the servo of J8 axis.	View the parameter H0B-45 and handle the problem according to the user guide.
0x50E1	Look-ahead error	Look-ahead error	Save the error message on the system diagnostics page and contact the manufacturer.
0x50E2	Adaptive speed planning verification error	Adaptive speed planning verification error	Save the error message on the system diagnostics page and contact the manufacturer.
0x50E3	General speed planning verification error	General speed planning verification error	Save the error message on the system diagnostics page and contact the manufacturer.
0x50E4	Trajectory time too long	The time of a single trajectory exceeds 100 seconds.	This does not affect the system operation, but may cause unexpected problems. The speed and acceleration should be increased to reduce the execution time of a single instruction.
0x50E5	Smooth transition between instructions degenerates to zero	Due to various limitations, the instruction smooth transition becomes 0.	This does not affect normal use, but be aware that transition failures can reduce execution efficiency.
0x50E6	Stop time too long	Stop time exceeded 500 milliseconds.	This does not affect normal use, but be aware that the robot cannot

			be operated again before it is completely stopped.
0x50EE	EtherCAT communication feedback error	The speed is abnormal when the robot coasts to stop.	Save the error message on the system diagnostics page and contact the manufacturer.
0x8001	No network device	The FPGA module is not functioning properly.	Hardware device error, contact the manufacturer.
0x8002	No master found	Failed to request EtherCAT master	Restart the controller. If the problem persists, it is a hardware device error, contact the manufacturer.
0x8003	Invalid domain	Failed to request EtherCAT master domain resource	Restart the controller. If the problem persists, it is a hardware device error, contact the manufacturer.
0x8004	Slave not found	The slave could not be found when configuring it.	Restart the controller or contact the manufacturer to check the EtherCAT slave device.
0x8005	Invalid process data	Error configuring slave PDO	Restart the controller or contact the manufacturer to check the EtherCAT slave device.
0x8006	Invalid service data	Error configuring slave SDO	Restart the controller or contact the manufacturer to check the EtherCAT slave device.
0x8007	Invalid entry object	Error configuring slave PDO entry	Restart the controller or contact the manufacturer to check the EtherCAT slave device.
0x8008	Domain memory address allocation error	Failed to request EtherCAT master domain memory	Restart the controller. If the problem persists, contact the manufacturer.
0x8009	Failed to activate master	Failed to apply for activation of EtherCAT master	Restart the controller. If the problem persists, contact the manufacturer.
0x800A	Service data public error	Length mismatch or slave not found while configuring slave SDO	Restart the controller. If the problem persists, contact the manufacturer to check the EtherCAT slave deice.
0x800B	Registration cycle callback error	Failed to create timed interrupt task	Restart the controller. If the problem persists, contact the manufacturer.
0x800C	Process communication configuration error	Error configuring PDO buffer	Restart the controller. If the problem persists, contact the manufacturer.
0x800D	Module initialization error	EtherCAT module resource	Restart the controller. If the

		initialization error	problem persists, contact the manufacturer.
0x800E	Error parsing configuration	Failed to parse the slave configuration	Check the secondary development configuration and reconfigure if necessary
0x800F	Channel parameter configuration error	Failure to configure the interval between channel synchronization signals and interrupts	Restart the controller. If the problem persists, contact the manufacturer.
0x8010	Domain registration error	Domain registration failure detected	Restart the controller. If the problem persists, contact the manufacturer.
0x8011	Timer creation error	Failed to create timer due to system reasons	Restart the controller. If the problem persists, contact the manufacturer.
0x8012	Timer startup error	Failed to start timer due to system reasons	Restart the controller. If the problem persists, contact the manufacturer.
0x8013	ECAT communication cycle configuration error	ECAT communication cycle is not an intergral number of 250us.	Check the secondary development configuration and reconfigure if necessary.
0x8014	ECAT version selection error	Wrong version of ECAT is used.	ECAT version error, contact the manufacturer.
0x8015	ECAT servo slave quantity error	The number of configured ECAT servo slaves is smaller than 1.	Check that the number of ECAT servo slaves is configured correctly.
0x8016	ECAT I/O slave quantity error	The number of configured ECAT I/O slaves is smaller than 0.	Check that the number of ECAT I/O slaves is configured correctly.
0x8017	ECAT I/O module quantity error	The number of configured ECAT I/O slaves is smaller than 1.	Check that the number of ECAT I/O slaves is configured correctly.
0x8018	ECAT I/O type error	The ECAT I/O type is unknown.	Check the secondary development configuration and reconfigure if necessary.
0x8019	ECAT I/O not supported	The type of configured ECAT I/O is currently not supported.	Check the secondary development configuration and reconfigure if necessary.
0x801A	ECAT memory request error	Current memory request failed due to excessive system resource usage.	Restart the controller. If the problem persists, contact the manufacturer.
0x801B	ECAT alarm shared memory request error	Failed to request ECAT alarm shared memory	Restart the controller. If the problem persists, contact the manufacturer.

0x801C	ECAT servo operation mode error	The servo control mode is not CSP-8, CSV-9, or CST-10.	Check the secondary development configuration and reconfigure if necessary.
0x801D	ECAT register error	Failed to access slave register	Check that the slave registers to be accessed are correct or allowed to be accessed.
0x801E	The number of configured ECAT I/Os does not match the number of online I/Os.	The number of configured ECAT I/Os does not match the number of online I/Os.	Check the number of configured ECAT I/Os and the number of online I/Os.
0x801F	The number of configured ECAT servos does not match the number of online servos.	The number of configured ECAT servos does not match the number of online servos.	Check the number of configured ECAT servos and the number of online servos.
0x8020	ECAT servo supplier code not supported	The slave device is not supported.	Check that all slave devices in all ECAT networks are supported.
0x8028	Error writing buffer	Failed to write buffer due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x8029	Error writing start command	Failed to write start command error due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x802A	Error reading status register	Failed to read status register due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x802B	Error reading data link status	Failed to read the data link status due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x802C	Error reading service data channel	Failed to read the service data channel due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x802D	Error reading service data length	Failed to read the service data length due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x802E	Service data length error	Service data length error due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x802F	Service data reception error	Service data reception error due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x8030	Service data channel busy	Service data channel busy error due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x8031	Service data message error	Service data message error due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.

0x8032	Communication error in reading process data	Error reading process data due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x8033	Error reading process data length	Error in the length of the process data buffer due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x8034	Process data length error	Error in the length of the process data buffer due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x8035	Process data reception error	Failed to receive process data due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x8036	Error opening network device	Failed to open network device due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x8037	Underlying communication error	The underlying communication control error is caused by a hardware error.	Restart the controller. If the problem persists, contact the manufacturer.
0x8038	Underlying communication error	Underlying communication read error due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x8039	Underlying communication error	Underlying communication write error due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x803A	Error reading send time stamp	Failed to read send time stamp due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x803B	Error reading receive time stamp	Failed to read receive time stamp due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x803C	Error reading remaining process data	Failed to read the remaining process data due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x803D	Error reading application time stamp	Failed to read application time stamp due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x803E	Error opening fieldbus LED	Failed to open fieldbus LED due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x803F	Field bus LED IOCTRL error	Fieldbus LED IOCTRL error due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x8040	Invalid underlying hardware	The underlying hardware module 1 is invalid due to a	Restart the controller. If the problem persists, contact the

		hardware error.	manufacturer.
0x8041	Bottom hardware invalid	The underlying hardware module 2 is invalid due to a hardware error.	Restart the controller. If the problem persists, contact the manufacturer.
0x8042	ECAT slave disconnected	ECAT slave is disconnected.	Check the ECAT device.
0x8043	Non-ECAT slave device connected	A non-ECAT slave device is connected.	Check that the connected device meets the requirements.
0x8044	ECAT network port 0 not connected	ECAT network port 0 is not connected.	Check the status of ECAT port 0.
0x8045	ECAT network port 1 disconnection error	ECAT network port 1 is not connected.	Check the status of ECAT port 1.
0x8046	ECAT startup time setting error	ECAT startup time is incorrectly set due to a hardware error.	Restart the controller. If the problem persists, contact the manufacturer.
0x8047	ECAT slave status error	The read ECAT slave status word indicates that the status is abnormal.	Check the error codes to troubleshoot and restart the system.
0x8048	Failed to open EOE virtual NIC device	System driver loading is not normal.	Confirm the system software version and restart the system.
0x805C	Toggle bit unchanged	The slave feeds back that the toggle bit is not changed.	Re-initiate a SDO access, or contact the manufacturer.
0x805D	SDO protocol timeout	The slave feeds back that the SDO protocol timed out.	Re-initiate a SDO access, or contact the manufacturer.
0x805E	Client/Server command specifier not valid or unknown	The slave feeds back that Client/Server command specifier is not valid or unknown.	Internal system error, contact the manufacturer.
0x805F	Object inaccessible	The slave feeds back that the object is not accessible.	This operation is prohibited.
0x8060	Error reading an write-only object	The slave feeds back that an error occurs while trying to read a write-only object.	This operation is prohibited.
0x8061	Error writing an read-only object	The slave feeds back that an error occurs while trying to write a read-only object.	This operation is prohibited.
0x8062	Object does not exist in the object directory	The slave feeds back that the object does not exist in the object directory.	Check that the object exists in the object dictionary.
0x8063	Object cannot be mapped into the PDO	The slave feeds back that the object cannot be mapped to the PDO.	Check that the object to be mapped is correct.
0x8064	The length of the object to be mapped exceeds the PDO length.	The slave feeds back that the length of the object to be mapped exceeds the PDO	Check that the length of the object to be mapped is correct.

		length.	
0x8065	Basic parameter incompatible	The slave feeds back that the base parameters are incompatible.	Check that the basic parameters of the slave are compatible.
0x8066	Device internal incompatibility	The slave feeds back that the device is internally incompatible.	Restart the controller and slave. If the problem persists, contact the manufacturer.
0x8067	Access failure due to hardware causes	The slave feeds back that the access failed due to hardware causes.	Restart the controller and slave. If the problem persists, contact the manufacturer.
0x8068	Service parameter length mismatch	The slave feeds back that the service parameter length mismatches.	Check that the length of the service parameter of the accessed slave is correct.
0x8069	Service parameter too long	The slave feeds back that the service parameter length is too long.	Check that the length of the service parameter of the accessed slave is correct.
0x806A	Service parameter too short	The slave feeds back that the service parameter length is too short.	Check that the length of the service parameter of the accessed slave is correct.
0x806B	Subindex does not exist	The slave feeds back that the subindex does not exist.	Check that the object subindex exists.
0x806C	Parameter value out of range	The slave feeds back that the parameter value is out of range.	Check that the parameters are set correctly.
0x806D	Written parameter value too large	The slave feeds back that the written parameter value is too large.	Check that the parameters are set correctly.
0x806E	Written parameter value too small	The slave feeds back that the written parameter value is too small.	Check that the parameters are set correctly.
0x806F	Maximum value smaller than minimum value	The slave feeds back that the maximum value is less than the minimum value.	Check that the values are set correctly.
0x8070	Error transferring or storing data	The slave feeds back that the data failed to be transferred or stored.	Restart the controller and slave. If the problem persists, contact the manufacturer.
0x8071	Error storing data due to local control	The slave feeds back that the data failed to be stored due to local control.	Restart the controller and slave. If the problem persists, contact the manufacturer.
0x8072	Error storing data due to device status	The slave feeds back that the data failed to be stored due to device status.	Restart the controller and slave. If the problem persists, contact the manufacturer.
0x8073	Object dictionary dynamic generation fails or no object dictionary is present.	The slave feeds back that the object dictionary dynamic generation fails or no object	Check that the object dictionary exists.

		dictionary is present.	
0x8079	IR-LINK initialization error	Failed to initialize IR-LINK due to IR-Link configuration or hardware connection error	Check the IR-Link configuration or hardware connection.
0x807A	Error configuring IR-Link communication cycle	The IR-Link communication cycle is not an integral number of 250us.	Check the IR-Link communication cycle configuration.
0x807B	IR-Link version error	Wrong version of IR-Link is used.	Contact the manufacturer.
0x807C	IR-Link servo slave quantity error	The number of configured IR-Link servos is less than 0.	Check the number of configured IR-Link servos.
0x807D	IR-Link slave quantity error	The number of configured IR-Link slaves is less than 1	Check the number of configured IR-Link slaves.
0x807E	IR-Link module quantity error	The number of configured IR-Link modules is less than 1.	Check the number of configured IR-Link modules.
0x807F	IR-Link type error	The type of configured IR-Link I/O is unrecognized.	Check the type of configured IR-Link I/O.
0x8080	IR-Link not supported	The configured IR-Link is not supported.	Check the IR-Link configuration.
0x8081	IR-Link memory request error	Current memory request failed due to excessive system resource usage.	Restart the controller. If the problem persists, contact the manufacturer.
0x8082	IR-LINK error shared memory request error	IR-Link error shared memory request failed.	Restart the controller. If the problem persists, contact the manufacturer.
0x8083	IR-LINK slave operation mode error	IR-Link slave control mode is not MODE-8.	Check the operation mode of IR-Link slave.
0x8084	IR-LINK register error	Failed to access slave register	Check that the slave registers to be accessed are correct or allowed to be accessed.
0x8085	The number of configured IR-Link I/Os does not match the number of online I/Os.	The number of configured IR-Link I/Os does not match the number of online I/Os.	Check the number of configured IR-Link I/Os and the number of online I/Os.
0x8086	The number of configured IR-Link slaves does not match the number of online slaves.	The number of configured IR-Link slaves does not match the number of online slaves.	Check the number of configured IR-Link slaves and the number of online slaves.
0x8087	IR-Link servo supplier codes not supported	The IR-Link servo supplier codes are not supported.	Check the IR-Link configuration.
0x8090	Error writing buffer	Failed to write buffer due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x8091	Error reading service data channel	Read service data channel error due to hardware error	Restart the controller. If the problem persists, contact the

			manufacturer.
0x8092	Error reading service data length	Failed to read the service data length due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x8093	Service data length error	Service data length error due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x8094	Service data reception error	Service data reception error due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x8095	Service data channel busy	Service data channel busy error due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x8096	Service data message error	Service data message error due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x8097	Communication error in reading process data	Error reading PDO due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x8098	Error reading process data length	Failed to read the process data length due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x8099	Process data length error	Process data length error due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x809A	Process data reception error	Process data reception error due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x809B	Error opening network device	Failed to open network device due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x809C	GPMC IR-Link read error	GPMC IR-Link read error due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x809D	GPMC IR-Link write error	GPMC IR-Link write error due to hardware error	Restart the controller. If the problem persists, contact the manufacturer.
0x809E	IR-Link slave disconnected	The IR-Link slave is disconnected.	Check the status of IR-Link device.
0x809F	Non-IR-Link slave connected	A non-IR-Link slave device is connected.	Check that the connected device meets the requirements.
0x80A0	IR-LINK network port 0 not connected	The IR-LINK network port 0 is not connected.	Check the connection of IR-Link port 0.
0x80A1	IR-LINK network port 1 not	The IR-LINK network port 1 is	Check the connection of IR-Link

	connected	not connected.	port 1.
0x80A2	IR-LINK startup time setting error	IR-LINK startup time is incorrectly set due to a hardware error.	Restart the controller. If the problem persists, contact the manufacturer.
0x8142	ECAT slave 1 disconnected	The ECAT slave 1 is disconnected.	Check the ECAT device.
0x819E	IR-LINK slave 1 disconnected	The IR-LINK slave 1 is disconnected.	Check the connection status of IR-LINK device.
0x8242	ECAT slave 2 disconnected	The ECAT slave 2 is disconnected.	Check the ECAT device.
0x829E	IR-LINK slave 2 disconnected	The IR-LINK slave 2 is disconnected.	Check the connection status of IR-LINK device.
0x8342	ECAT slave 3 disconnected	The ECAT slave 3 is disconnected.	Check the ECAT device.
0x839E	IR-LINK slave 3 disconnected	The IR-LINK slave 3 is disconnected.	Check the connection status of IR-LINK device.
0x8442	ECAT slave 4 disconnected	The ECAT slave 4 is disconnected.	Check the ECAT device.
0x849E	IR-LINK slave 4 disconnected	The IR-LINK slave 4 is disconnected.	Check the connection status of IR-LINK device.
0x8542	ECAT slave 5 disconnected	The ECAT slave 5 is disconnected.	Check the ECAT device.
0x859E	IR-LINK slave 5 disconnected	The IR-LINK slave 5 is disconnected.	Check the connection status of IR-LINK device.
0x8642	ECAT slave 6 disconnected	The ECAT slave 6 is disconnected.	Check the ECAT device.
0x869E	IR-LINK slave 6 disconnected	The IR-LINK slave 6 is disconnected.	Check the connection status of IR-LINK device.
0x8742	ECAT slave 7 disconnected	The ECAT slave 7 is disconnected.	Check the ECAT device.
0x879E	IR-LINK slave 7 disconnected	The IR-LINK slave 7 is disconnected.	Check the connection status of IR-LINK device.
0x8842	ECAT slave 8 disconnected	The ECAT slave 8 is disconnected.	Check the ECAT device.
0x889E	IR-LINK slave 8 disconnected	The IR-LINK slave 8 is disconnected.	Check the connection status of IR-LINK device.
0x8942	ECAT slave 9 disconnected	The ECAT slave 9 is disconnected.	Check the ECAT device.
0x899E	IR-LINK slave 9 disconnected	The IR-LINK slave 9 is disconnected.	Check the connection status of IR-LINK device.
0x8A42	ECAT slave 10 disconnected	The ECAT slave 10 is disconnected.	Check the ECAT device.
0x8A9E	IR-LINK slave 10 disconnected	The IR-LINK slave 10 is disconnected.	Check the connection status of IR-LINK device.

0x8B42	ECAT slave 11 disconnected	The ECAT slave 11 is disconnected.	Check the ECAT device.
0x8B9E	IR-LINK slave 11 disconnected	The IR-LINK slave 11 is disconnected.	Check the connection status of IR-LINK device.
0x8C42	ECAT slave 12 disconnected	The ECAT slave 12 is disconnected.	Check the ECAT device.
0x8C9E	IR-LINK slave 12 disconnected	The IR-LINK slave 12 is disconnected.	Check the connection status of IR-LINK device.
0x8D42	ECAT slave 13 disconnected	The ECAT slave 13 is disconnected.	Check the ECAT device.
0x8D9E	IR-LINK slave 13 disconnected	The IR-LINK slave 13 is disconnected.	Check the connection status of IR-LINK device.
0x8E42	ECAT slave 14 disconnected	The ECAT slave 14 is disconnected.	Check the ECAT device.
0x8E9E	IR-LINK slave 14 disconnected	The IR-LINK slave 14 is disconnected.	Check the connection status of IR-LINK device.
0x8F42	ECAT slave 15 disconnected	The ECAT slave 15 is disconnected.	Check the ECAT device.
0x8F9E	IR-LINK slave 15 disconnected	The IR-LINK slave 15 is disconnected.	Check the connection status of IR-LINK device.
0xE001	Normal operation	Normal operation	No action required.
0xE002	Firmware loading failed	Unable to get firmware from SD card	Check whether it is in test mode, check the SD content.
0xE003	Firmware loading failed	Unable to get the correct firmware from Flash	Re-upgrade the system.
0xE004	Firmware loading failed	FPGA failed to reset	Restart the controller. If the problem persists, contact the manufacturer.
0xE005	Firmware loading failed	FPGA transfer error	Restart the controller. If the problem persists, contact the manufacturer.
0xE006	Firmware loading failed	Handshake with FPGA failed	Restart the controller. If the problem persists, contact the manufacturer.
0xE007	Control channel 1 running normally	Control channel 1 running normally	No action required.
0xE008	Failed to allocate memory to control channel 1	Insufficient memory	Restart the controller. If the problem persists, contact the manufacturer.
0xE009	Firmware loading failed	Firmware not found or information not available	Restart the controller. If the problem persists, upgrade the controller.
0xE00A	Firmware loading failed	Firmware not found or information not available	Restart the controller. If the problem persists, upgrade the

			controller.
0xE00B	Control channel 1 firmware length out of range	Control channel 1 firmware length out of range	Restart the controller. If the problem persists, upgrade the controller.
0xE00C	Firmware loading failed	Unable to communicate with FPGA	Restart the controller. If the problem persists, contact the manufacturer.
0xE00D	Failed to reset control channel 1	Failed to reset control channel 1	Restart the controller. If the problem persists, contact the manufacturer.
0xE00E	SPI communication failure	Failed to send start word	Restart the controller. If the problem persists, contact the manufacturer.
0xE00F	SPI communication failed	POS failed	Restart the controller. If the problem persists, contact the manufacturer.
0xE010	Failed to regulate communication rate of channel 1	Failed to regulate communication rate of channel 1	Restart the controller. If the problem persists, contact the manufacturer.
0xE011	Data loading timeout in control channel 1	Data loading timeout in control channel 1	Restart the controller. If the problem persists, contact the manufacturer.
0xE012	Control channel 1 running response is abnormal.	Control channel 1 running response is abnormal.	Restart the controller. If the problem persists, contact the manufacturer.
0xE013	Control channel 0 running normally	Control channel 0 running normally	No action required.
0xE014	Failed to allocate memory to control channel 0	Insufficient memory	Restart the controller. If the problem persists, contact the manufacturer.
0xE015	Firmware loading failed	Firmware not found or information not available	Restart the controller. If the problem persists, upgrade the controller.
0xE016	Firmware loading failed	Firmware not found or information not available	Restart the controller. If the problem persists, upgrade the controller.
0xE017	Control channel 0 firmware length out of range	Control channel 0 firmware length out of range	Restart the controller. If the problem persists, upgrade the controller.
0xE018	Firmware loading failed	Unable to communicate with FPGA	Restart the controller. If the problem persists, contact the manufacturer.
0xE019	Failed to reset control channel 0	Failed to reset control channel 0	Restart the controller. If the problem persists, contact the

			manufacturer.
0xE01A	SPI communication failure	Failed to send start word	Restart the controller. If the problem persists, contact the manufacturer.
0xE01B	SPI communication failure	POS failed	Restart the controller. If the problem persists, contact the manufacturer.
0xE01C	Failed to regulate communication rate of channel 0	Failed to regulate communication rate of channel 0	Restart the controller. If the problem persists, contact the manufacturer.
0xE01D	Data loading timeout in control channel 0	Data loading timeout in control channel 0	Restart the controller. If the problem persists, contact the manufacturer.
0xE01E	Control channel 0 running response is abnormal.	Control channel 0 running response is abnormal.	Restart the controller. If the problem persists, contact the manufacturer.

Appendix 2: API Instructions and Connection Fault Table

API Instructions

(1) API Instructions

No.	Function Name	Description	Parameter	Return Value	Note
1	int **IMC100_Init_ETH**(unsigned int ipAddr,unsigned short ipPort,int timeOut=5, int comId=0)	Establishes a robotic network connection	ipAddr: Robot controller network IP address, host byte order ipPort: Robot controller network port number, default 2222 timeOut: Communication timeout setting, default 5s comId: Connection number, marking different connections under the same destination IP and port number, default 0, maximum 4 (same below);	0: Connection success; ＜0: Failure	1) Up to 5 different connections supported by the host controller; up to 4 different connections supported by the controller 2) Scope of the connection number: The same process on the host controller. *Any difference in the IP and port number between the host controller and the controller is considered a

					different connection.
2	int **IMC100_Exit_ETH**(int comId=0)	Closes the robot network connection	comId: Connection number, which marks the corresponding connection (this parameter is not repeated below)	0: Success; ＜0: Failure	
3	int **IMC100_EmergStop**(int cmd, int comId=0)	Controls emergency stop switch	cmd: Emergency stop command, 1-Presses emergency stop, 0-Releases emergency stop	0: Success; ＜0: Failure	
4	int **IMC100_MotorEnable**(int cmd, int comId=0)	Enables or disables the motor	cmd: Motor enable command, 1-Enable, 0-Disable	0: Success; ＜0: Failure	Read the enable status 300 ms after the enable command is issued.
5	int **IMC100_ResetErr**(int comId=0)	Fault reset		0: Success; ＜0: Failure	The command is delayed by approx. 50ms.
6	int **IMC100_Set_Mode**(int mode, int comId=0)	Sets the system operating mode	mode: 1-Teach, 2-Play	0: Success; ＜0: Failure	
7	int **IMC100_PrgCtrl**(int cmd, int comId=0)	Controls the teaching program	cmd: Control command, 0-Stop, 1-Start/Resume	0: Success; ＜0: Failure	
8	int **IMC100_BackStartLine**(int comId=0)	Returns the program to the start line		0: Success; ＜0: Failure	
9	int **IMC100_Set_Vel**(int val, int comId=0)	Sets the current operating speed level	val: Current speed level, range 1-100	0: Success; ＜0: Failure	

10	int **IMC100_Set_AccRamp**(double startVal, double endVal , int comId=0)	Sets the jerk of the motion segment in the data streaming mode	startVal: Speed percentage of the start segment, range 10.0-100.0 endVal: Speed percentage of the end segment, range 10.0-100.0	0: Success; <0: Failure	Only valid in data streaming mode.
11	int **IMC100_Set_RapidMove**(int movType, int enableFlag， int comId=0);	Sets the optimal trajectory planning	movType: Motion type, 0-CP motion, 1-PTP motion enableFlag: 0-Open optimal trajectory planning, 1-Close optimal trajectory planning	0: Success; <0: Failure	Only valid in data streaming mode.
12	int **IMC100_Set_FlyMode**(int cpMode, int flyMode, int comId);	Sets the transition mode of motion instruction	cpMode: Motion type, 0-CP motion flyMode: Transition mode, 0-Free transition, 1-Fixed path transition	0: Success; <0: Failure	Only valid in data streaming mode.
13	int **IMC100_Set_FlyPress**(int flyPressPos, int flyPressOrient, int comId);	Sets the transition stress for a fixed path	flyPressPos: Position transition stress, range 50-200 flyPressOrient: Orientation transition stress, range 50-200	0: Success; <0: Failure	Only valid in data streaming mode.
14	int **IMC100_DsMode**(int cmd, int comId=0)	Controls the data streaming mode	cmd: Data streaming command, 0-Off, 1-On, 2-Pause, 3-Resume	0: Success; <0: Failure	When the data streaming is on, if the robot is disabled the data streaming is paused.
15	int **IMC100_Set_DO**(int num, int status, int comId=0)	Sets the DO status by bit (DOs that can be controlled by RC)	num: DO bit sequence number status: DO status, 0-Off, 1-On	0: Success; <0: Failure	

16	int **IMC100_Set_SlewMode**(int cmd, int comId=0);	Set the rotation optimization for J4 axis of SCARA robots or J6 axis of standard 6-axis robot	cmd: Rotation optimization mode 0 - Optimization not applied, depending on the arm parameter of the position variable. 1 - Optimization mode 1 applied, to ensure that J4/J6 is within the range of -180° to 180° during movement. 2 - Ensure that J4/J6 moves in the closest possible manner, and the robot will calculate whether movement to the target point requires the J4/J6 to rotated by 180°. If the angle difference is ≤ 180°, the robot will fully move to the target point. If it is >180°, J4/J6 will move in the opposite direction and ultimately moves to a position that is 360° away from the position of J4/J6 at the target point. During the movement, if the reverse movement of J4/J6 exceeds the limit range of the robot, it will stop moving and issue an alarm. 3 - Ensure that J4/J6 moves in the closest possible manner. The difference from mode 2 is that if the reverse movement of J4/J6 exceeds the limit range, there will be no alarm, but instead no reverse movement will be carried out and the original normal movement will be fully adopted.	0: Success; ＜0: Failure	Only valid in stream mode, the data will be cleared after data streaming mode is turned off.
17	int **IMC100_Set_DOGroup**(int num, int status, int comId=0)	Sets the DO status by group	num: DO group number, range 0-7, depending on actual configuration status:	0: Success; ＜0:	

			DO status in each group, range 0-255, where bit0-bit7 corresponds to the DO status with the lowest to highest group number	Failure	
18	int **IMC100_Set_DA**(int num, float val, int comId=0)	Sets the output value of DA by number	num: DA number, range 0-15 val: DA value, 0mA to 20mA for current type, -10V to 10V for voltage type, depending on the DA channel type	0: Success; ＜0: Failure	
19	int **IMC100_InchMode**(int cmd, int comId=0)	Controls the jogging teach mode	cmd: Jogging teach mode command, 0-Off, 1-On	0: Success; ＜0: Failure	
20	int **IMC100_Set_InchStep**(int val, int comId=0)	Sets the step size of the jog motion	Val: Step size, range 1-4, where 1 indicates 0.05, 2 indicates 0.5 for step size, 3 indicates 2, 4 indicates that the step size is the setting value of jog parameter. The unit is degree in the joint coordinate system and mm in the base coordinate system.	0: Success; ＜0: Failure	
21	int **IMC100_Jog**(int mode, int axis, int cmd, int comId=0)	Teach motion command	Mode: Teaching mode, 0-Joint coordinate teaching, 1-Cartesian coordinate teaching axis: Axis number, range 1-6, corresponding to J1-J6 axis in joint coordinate teaching, and X/Y/Z/RZ/RY/RX axis in Cartesian coordinate teaching cmd: Teach command, 0-Stop, 1-Forward teach, -1-Reverse teach	0: Success; ＜0: Failure	Effective after data streaming mode is turned off in teach mode
22	int **IMC100_Inch**(int mode, int axis, int cmd, int comId=0)	Jog motion command	Mode: Teaching mode, 0-Joint coordinate teaching, 1-Cartesian coordinate teaching axis: Axis number, range	0: Success; ＜0: Failure	Effective after data streaming mode is turned off

			1-6, corresponding to J1-J6 axis in joint coordinate teaching, and X/Y/Z/RZ/RY/RX axis in Cartesian coordinate teaching cmd: Teach command, 1-Forward teach, -1-Reverse teach		in teach mode
23	int **IMC100_Home**(int num, int comId=0)	Homing motion command	num: Origin number, range 0-4	0: Success; ＜0: Failure	Only effective in data streaming mode.
24	int **IMC100_MovJ_P**(int posNum, int vel=100, int zone=0, int comId=0)	Moves to a global position with a specified number through joint interpolation	posNum: Target global position number, range 0-1000 vel: Motion speed, range 1-100, default 100 zone: Interpolation precision, range -1 to 5, default 0 (-1 for Fine, 0-5 for Z[0]-Z[5], same below)	0: Success; ＜0: Failure	Only effective in data streaming mode.
25	int **IMC100_MovL_P**(int posNum, int vel=100, int zone=0, int comId=0)	Moves to a global position with a specified number through linear interpolation	posNum: Target global position number, range 0-1000 vel: Motion speed, range 1-100, default 100 zone: Interpolation precision, range -1 to 5, default 0	0: Success; ＜0: Failure	Only effective in data streaming mode.
26	int **IMC100_MovC_P**(int posMidNum, int posDstNum, int vel=100, int zone=0, int comId=0)	Moves to a global position with a specified number through circular interpolation	PosMidNum: Global position number at an intermediate point of an arc, range 0-1000 posDstNum: Global position number at the end of an arc, range 0-1000 vel: Motion speed, range 1-100, default 100 zone: Interpolation precision, range -1 to 5, default 0	0: Success; ＜0: Failure	Only effective in data streaming mode.
27	int **IMC100_MovJ2**(ROBOT_POS pos, int vel=100, int zone=0, int comId=0)	Moves to a position with a specified value	pos: Position parameter structure, see definition vel: Motion speed, range	0: Success; ＜0:	Only effective in data

		through joint interpolation	1-100, default 100 zone: Interpolation accuracy, range -1 to 5, default 0	Failure	streaming mode.
28	int **IMC100_MovL2**(ROBOT_POS pos, int vel=100, int zone=0, int comId=0)	Moves to a position with a specified value through linear interpolation	pos: Position parameter structure, see definition vel: Motion speed, range 1-100, default 100 zone: Interpolation accuracy, range -1 to 5, default 0	0: Success; ＜0: Failure	Only effective in data streaming mode.
29	int **IMC100_MovC2**(ROBOT_POS posMid, ROBOT_POS posDst, int vel=100, int zone=0, int comId=0)	Moves to a position with a specified value through circular interpolation	posMid: Position parameter of an intermediate point of an arc posDst: Position of the end point of the arc vel: Motion speed, range 1-100, default 100 zone: Interpolation accuracy, range -1 to 5, default 0	0: Success; ＜0: Failure	Only effective in data streaming mode.
30	int **IMC100_MovJ_P_IO**(int posNum, int vel, int zone, MOV_IO *movIo, int ioNum, int comId=0)	Moves to a global position with a specified number through joint interpolation while controlling the corresponding I/O	posNum: Target global position number, range 0-1000 vel: Motion speed, range 1-100 zone: Interpolation accuracy, range -1 to 5 movIo: I/O control structure, see definition ioNum: The number of groups of the I/O control structure, range 1-3	0: Success; ＜0: Failure	Only effective in data streaming mode.
31	int **IMC100_MovL_P_IO**(int posNum, int vel, int zone, MOV_IO *movIo, int ioNum,int comId=0)	Moves to a global position with a specified number through linear interpolation while controlling the corresponding I/O	posNum: Target global position number, range 0-1000 vel: Motion speed, range 1-100 zone: Interpolation accuracy, range -1 to 5 movIo: I/O control structure, see definition ioNum: The number of groups of the I/O control structure, range 1-3	0: Success; ＜0: Failure	Only effective in data streaming mode.

32	int **IMC100_MovC_P_IO**(int posMidNum, int posDstNum, int vel, int zone, MOV_IO *movIo, int ioNum,int comId=0)	Moves to a global position with a specified number through circular interpolation while controlling the corresponding I/O	posMidNum: Global position number at an intermediate point of an arc, range 0-1000 posDstNum: Global position number at the end of an arc, range 0-1000 vel: Motion speed, range 1-100 zone: Interpolation accuracy, range -1 to 5 movIo: I/O control structure, see definition ioNum: Number of groups of I/O control structure, range 1-3	0: Success; ＜0: Failure	Only effective in data streaming mode.
33	int **IMC100_MovJ2_IO**(ROBOT_POS pos, int vel, int zone, MOV_IO *movIo, int ioNum, int comId=0)	Moves to a position with a specified value through joint interpolation while controlling the corresponding I/O	pos: Position parameter structure, see definition vel: Motion speed, range 1-100 zone: Interpolation accuracy, range -1 to 5 movIo: I/O control structure, see definition ioNum: The number of groups of the I/O control structure, range 1-3	0: Success; ＜0: Failure	Only effective in data streaming mode.
34	int **IMC100_MovL2_IO**(ROBOT_POS pos, int vel, int zone, MOV_IO *movIo, int ioNum, int comId=0)	Moves to a position with a specified value through linear interpolation while controlling the corresponding I/O	pos: Position parameter structure, see definition vel: Motion speed, range 1-100 zone: Interpolation accuracy, range -1 to 5 movIo: I/O control structure, see definition ioNum: The number of groups of the I/O control structure, range 1-3	0: Success; ＜0: Failure	Only effective in data streaming mode.
35	int **IMC100_MovC2_IO**(ROBOT_POS posMid, ROBOT_POS posDst, int vel, int zone, MOV_IO *movIo, int ioNum, int comId=0)	Moves to a position with a specified value through circular interpolation while controlling	posMid: Position parameter at an intermediate point of an arc posDst: Position parameter at the end of an arc vel: Motion speed, range	0: Success; ＜0: Failure	Only effective in data streaming mode.

		the corresponding I/O	1-100 zone: Interpolation accuracy, range -1 to 5 movIo: I/O control structure, see definition ioNum: Number of groups of I/O control structure, range 1-3		
36	int **IMC100_Jump_P**(int posNum, int vel=100, int zone=0, int comId=0)	Moves to a global position with a specified number through jump	posNum: Target global position number, range 0-1000 vel: Motion speed, range 1-100, default 100 zone: Interpolation precision, range -1 to 5, default 0	0: Success; ＜0: Failure	Only effective in data streaming mode.
37	int **IMC100_JumpL_P**(int posNum, int vel=100, int zone=0, int comId=0)	Moves to a global position with a specified number through linear jump	posNum: Target global position number, range 0-1000 vel: Motion speed, range 1-100, default 100 zone: Interpolation precision, range -1 to 5, default 0	0: Success; ＜0: Failure	Only effective in data streaming mode.
38	int **IMC100_Jump2**(ROBOT_POS pos, int vel=100, int zone=0, int comId=0)	Moves to a position with a specified value through jump	pos: Position parameter structure, see definition vel: Motion speed, range 1-100, default 100 zone: Interpolation accuracy, range -1 to 5, default 0	0: Success; ＜0: Failure	Only effective in data streaming mode.
39	int **IMC100_JumpL2**(ROBOT_POS pos, int vel=100, int zone=0, int comId=0)	Moves to a position with a specified value through linear jump	pos: Position parameter structure, see definition vel: Motion speed, range 1-100, default 100 zone: Interpolation accuracy, range -1 to 5, default 0	0: Success; ＜0: Failure	Only effective in data streaming mode.
40	int **IMC100_Jump_P_IO**(int posNum, int vel, int zone, MOV_IO *movIo, int ioNum, int comId=0)	Moves to a global position with a specified number through jump while controlling the corresponding I/O	posNum: Target global position number, range 0-1000 vel: Motion speed, range 1-100 zone: Interpolation accuracy, range -1 to 5 movIo: I/O control structure, see definition ioNum: The number of	0: Success; ＜0: Failure	Only effective in data streaming mode.

			groups of the I/O control structure, range 1-3		
41	int **IMC100_JumpL_P_IO**(int posNum, int vel, int zone, MOV_IO *movIo, int ioNum,int comId=0)	Moves to a global position with a specified number through linear jump while controlling the corresponding I/O	posNum: Target global position number, range 0-1000 vel: Motion speed, range 1-100 zone: Interpolation accuracy, range -1 to 5 movIo: I/O control structure, see definition ioNum: The number of groups of the I/O control structure, range 1-3	0: Success; ＜0: Failure	Only effective in data streaming mode.
42	int **IMC100_Jump2_IO**(ROBOT_POS pos, int vel, int zone, MOV_IO *movIo, int ioNum, int comId=0)	Moves to a global position with a specified value through jump while controlling the corresponding I/O	pos: Position parameter structure, see definition vel: Motion speed, range 1-100 zone: Interpolation accuracy, range -1 to 5 movIo: I/O control structure, see definition ioNum: The number of groups of the I/O control structure, range 1-3	0: Success; ＜0: Failure	Only effective in data streaming mode.
43	int **IMC100_JumpL2_IO**(ROBOT_POS pos, int vel, int zone, MOV_IO *movIo, int ioNum, int comId=0)	Moves to a global position with a specified value through linear jump while controlling the corresponding I/O	pos: Position parameter structure, see definition vel: Motion speed, range 1-100 zone: Interpolation accuracy, range -1 to 5 movIo: I/O control structure, see definition ioNum: The number of groups of the I/O control structure, range 1-3	0: Success; ＜0: Failure	Only effective in data streaming mode.
44	int **IMC100_Get_PosHere**(ROBOT_POS *pos, int comId=0)	Queries position parameters of the current point (in relation to the current coordinate	Pos: Position parameter structure, representing the result of the query	0: Success; ＜0: Failure	

		system)			
45	int **IMC100_Get_PosHereJ**(ROBOT_POS *pos, int comId=0)	Queries position parameters of the current point in the joint coordinate system	Pos: Position parameter structure, representing the result of the query (only coordinate values are valid, arm parameters and coordinate parameters are meaningless)	0: Success; ＜0: Failure	
46	int **IMC100_Get_PosHereC**(ROBOT_POS *pos, int comId=0)	Queries position parameters of the current point in the base coordinate system	Pos: Position parameter structure, representing the result of the query (only coordinate values are valid, arm parameters and coordinate parameters are meaningless)	0: Success; ＜0: Failure	
47	int **IMC100_Get_PosHerePulse**(double pos[6], int comId=0)	Queries pulse value of the current point in the base coordinate system	Pos[]: The current pulse value, representing the result of the query	0: Success; ＜0: Failure	
48	int **IMC100_Get_PosCnvt**(ROBOT_POS *posSrc, ROBOT_POS *posDst, int comId=0)	Queries the coordinate system conversion results of position parameters	posSrc: Original coordinate parameter structure, where the coord range is 1-4, which indicates that the robot points are converted in different coordinate systems. posDst: Target coordinate parameter structure, which represents the result of the conversion, where coord, toolNo, userNo represent the coordinate system parameters referenced for conversion, which need to be written in advance by the user. When coord is 1 and 2, toolNo and userNo are meaningless.	0: Success; ＜0: Failure	

49	int **IMC100_Get_VisionPosCnvt**(ROBOT_POS *posSrc, ROBOT_POS *basePos, ROBOT_POS *posDst, int comId=0)	Queries the results of converting camera pixels to robot coordinates	posSrc: Original pixel (in camera coordinate system) coordinate structure, with coord being 5 or 6. When coord is 5, it represents the conversion of points in the fixed camera coordinate system. When coord is 6, it represents the conversion of points in the dynamic camera coordinate system. toolNo represents the tool number, and userNo represents the vision coordinate system number. basePos: For fixed camera coordinate system, it is meaningless; for the mobile camera coordinate system, it represents the coordinates of the vision reference point (the camera's shooting point). posDst: Target coordinate parameter structure, which represents the result of the conversion, where coord, toolNo, userNo represent the coordinate system parameters referenced for the conversion, which need to be written in advance by the user. When coord is 1 and 2, toolNo and userNo are meaningless.	0: Success; ＜0: Failure	
50	int **IMC100_Get_OffsetJ**(ROBOT_POS *posSrc, double PR[6], ROBOT_POS *posDst, int comId=0)	Queries the offset point in the joint coordinate system	posSrc: Original point, in the joint coordinate system PR: Offset variable posDst: Resulting offset point	0: Success; ＜0: Failure	
51	int **IMC100_Get_Offset**(ROBOT_POS *posSrc, double PR[6], ROBOT_POS *posDst, int comId=0)	Queries the offset point in the Cartesian/user	posSrc: Original point, in the Cartesian/user coordinate system PR: Offset variable	0: Success; ＜0: Failure	

		coordinate system	posDst: Offset post point result		
52	int **IMC100_Get_OffsetT**(ROBOT_POS *posSrc, double PR[6], ROBOT_POS *posDst, int comId=0)	Queries the offset point in the tool coordinate system	posSrc: Original point, in the tool coordinate system PR: Offset variable posDst: Resulting offset point	0: Success; ＜0: Failure	
53	int **IMC100_Get_SysErrSts**(int *sts, int comId=0)	Queries the current error status of the system	sts: System error status, representing the result of the query, bit0 - System has an alarm, bit1 - System has a warning	0: Success; ＜0: Failure	
54	int **IMC100_Get_SysErr**(int *error, int comId=0)	Queries the error code of the system	error: System error code, representing the result of the query	0: Success; ＜0: Failure	
55	int **IMC100_Get_TaskPrgPath**(int taskId, char prgPath[128], int comId)	Queries the path to the program executed in the current task channel	taskId: Task channel, 0 is the main task prgPath: Current program path, representing the result of the query	0: Success; ＜0: Failure	
56	int **IMC100_Get_TaskRunSts**(int taskId, int *sts, int comId)	Queries the running status of the task channel	taskId: Task channel, 0 is the main task sts: Running status, representing the result of the query, 0-Stop, 1-Start/Resume, 10-Ready, 100-Task not activated, -1-Task active but no program configured	0: Success; ＜0: Failure	
57	int **IMC100_Get_TaskProgramLine**(int taskId, int *line, int comId=0)	Queries the number of line currently processed by the program executed in the task channel	line: The number of line currently processed by the current task, representing the result of the query	0: Success; ＜0: Failure	
58	int **IMC100_Get_CurMotionLine**(int *line, int comId=0)	Queries the number of line of the motion instruction that is currently being executed	line: The number of line of the motion instruction that is currently being executed, representing the result of the query	0: Success; ＜0: Failure	

59	int **IMC100_Get_InitSts**(int *sts, int comId=0)	Queries the system initialization status	sts: The initialization status of the system, representing the result of the query, range -1 to 11	0: Success; ＜0: Failure	
60	int **IMC100_Get_AccRamp**(double *startVal, double *endVal, int comId =0);	Queries the jerk of the motion segment in the data streaming mode	startVal: Speed percentage of the start segment, range 10.0-100.0 endVal: Speed percentage of the end segment, range 10.0-100.0	0: Success; ＜0: Failure	
61	int **IMC100_Get_RapidMove**(int movType, int *enableFlag, int comId =0)	Queries the optimal trajectory planning switch for the current motion type	movType: Motion type, 0-CP motion, 1-PTP motion enableFlag: 0-Optimal trajectory planning OFF for the current motion type, 1-Optimal trajectory planning ON for the current motion type	0: Success; ＜0: Failure	
62	int **IMC100_Get_FlyMode**(int cpMode, int *flyMode, int comId);	Queries the transition mode of motion instruction	cpMode: Motion type, 0-CP motion flyMode: Transition mode, 0-Free transition, 1-Fixed path transition	0: Success; ＜0: Failure	
63	int **IMC100_Get_FlyPress**(int *flyPressPos, int *flyPressOrient, int comId);	Queries the transition stress for a fixed path	flyPressPos: Position transition stress, range 50-200 flyPressOrient: Orientation transition stress, range 50-200	0: Success; ＜0: Failure	
64	int **IMC100_Get_Coord**(int *type, int comId=0)	Queries the current coordinate system type	type: Current coordinate system type, representing the results of the query, range 1 to 4, 1-Joint coordinate system, 2-Base coordinate system, 3-Tool coordinate system, 4-User coordinate system	0: Success; ＜0: Failure	
65	int **IMC100_Get_Vel**(int *val, int comId=0)	Queries the current speed level value	val: The current speed level value, representing the result of the query, range 1-100	0: Success; ＜0: Failure	
66	int **IMC100_Get_Mode**(int *mode, int comId=0)	Queries the current operation mode of the	mode: System operating mode, representing the result of the query, 1-Teach,	0: Success; ＜0:	

		system	2-Play, 3-Run in single step, 5-Run continuously	Failure	
67	int **IMC100_Get_DsMode**(int *val, int comId=0)	Queries if data streaming mode is on	val: Data streaming mode, representing the result of the query, 0 - off, 1-ON/Resume, 2-Pause	0: Success; <0: Failure	
68	int **IMC100_Get_InchMode**(int *val, int comId=0)	Queries the teaching method	val: Teach method, representing the result of the query, 0-Continuous teaching, 1-Jog teaching	0: Success; <0: Failure	
69	int **IMC100_Get_SlewMode**(int *val, int comId=0)	Queries the rotation optimization mode for J4 axis of SCARA robots or J6 axis of standard 6-axis robot	val: Rotation optimization mode for J4 axis of SCARA robots or J6 axis of standard 6-axis robot, for J4 axis of SCARA robots or J6 axis of standard 6-axis robot. 0 - Optimization not applied, depending on the arm parameter of the position variable. 1 - Optimization mode 1 applied, to ensure that J4/J6 is within the range of -180° to 180° during movement. 2 - Ensure that J4/J6 moves in the closest possible manner, and the robot will calculate whether movement to the target point requires the J4/J6 to rotated by 180°. If the angle difference is ≤ 180°, the robot will fully move to the target point. If it is >180°, J4/J6 will move in the opposite direction and ultimately moves to a position that is 360° away from the position of J4/J6 at the target point. During the movement, if the reverse movement of J4/J6 exceeds the limit range of the robot, it will stop moving and issue an alarm.	0: Success; <0: Failure	

			3 - Ensure that J4/J6 moves in the closest possible manner. The difference from mode 2 is that if the reverse movement of J4/J6 exceeds the limit range, there will be no alarm, but instead no reverse movement will be carried out and the original normal movement will be fully adopted.		
70	int **IMC100_Get_EStopSts**(int *sts, int comId=0)	Queries the current status of the emergency stop switch	sts: Emergency stop switch status, representing the result of the query, 0-Switch released, 1-Switch pressed	0: Success; ＜0: Failure	
71	int **IMC100_Get_MotorSts**(int *sts, int comId=0)	Queries the current motor enable status	sts: Motor enable status, representing the result of the query, 0-Disabled, 1-Enabled	0: Success; ＜0: Failure	
72	int **IMC100_Get_MotionSts**(int *sts, int comId=0)	Queries the current system motion status	sts: System motion status, representing the result of the query, 0-Stop/Motion complete, 1-In motion, 2-Motion interrupted	0: Success; ＜0: Failure	
73	int **IMC100_Get_SysMode**(int *mode, int comId=0)	Queries the current system mode	mode: System mode, representing the result of the query, 0-Normal mode, >0-Internal test mode	0: Success; ＜0: Failure	
74	int **IMC100_Get_PrgRunTime**(unsigned int *second, int comId=0)	Queries the running time of the teaching program	second: Time count value (in seconds), representing the result of the query	0: Success; ＜0: Failure	
75	int **IMC100_Get_CurCmdNum**(unsigned int *num, int comId=0)	Queries the number of the motion instructions (Home, MovJ, MovL) that were sent successfully	num: Instruction number, representing the result of the query	0: Success; ＜0: Failure	Only valid in data streaming mode.
76	int **IMC100_Get_CurCmdSts**(int *sts, int comId=0)	Queries the actual completion status of the motion instructions that	sts: Completion status, representing the result of the query, 0-Motion incomplete, 1-Motion complete	0: Success; ＜0: Failure	Only valid in data streaming mode.

		were sent successfully			
77	int **IMC100_Get_CmdSts**(int num, int *sts, int comId=0)	Queries the actual completion status of the motion instruction with a specified number	num: Instruction number sts: Completion status, representing the result of the query, 0-Motion incomplete, 1-Motion complete	0: Success; <0: Failure	Only valid in data streaming mode.
78	int **IMC100_Get_DINum**(int *num, int comId=0)	Queries the total number of system DIs	num: Total number of DIs, representing the result of the query	0: Success; <0: Failure	
79	int **IMC100_Get_DONum**(int *num, int comId=0)	Queries the total number of system DOs	num: Total number of DOs, representing the result of the query	0: Success; <0: Failure	
80	int **IMC100_Get_ADNum**(int *num, int comId=0)	Queries the total number of system ADs	Num: Total number of ADs, representing the result of the query	0: Success; <0: Failure	
81	int **IMC100_Get_DANum**(int *num, int comId=0)	Queries the total number of system DAs	Num: Total number of DAs, representing the result of the query	0: Success; <0: Failure	
82	int **IMC100_Get_DI**(int num, int *sts, int comId=0)	Queries the DI status by bit	num: DI number (not exceeding the total number of DIs) sts: DI status, representing the result of the query, 0-Off, 1-On	0: Success; <0: Failure	
83	int **IMC100_Get_DIGroup**(int num, int *sts, int comId=0)	Queries the DI status by group	num: DI group number sts: The DI status of each group, range 0-255, where bit0-bit7 corresponds to the DI status with the lowest to highest group number	0: Success; <0: Failure	
84	int **IMC100_Get_AD**(int num, float *val, int comId=0)	Queries the input values for AD by number	num: AD number (not exceeding total number of AD) val: AD value, representing the result of the query, current type in mA and voltage type in V	0: Success; <0: Failure	

85	int **IMC100_Get_DOCfg**(int num, int *val, int comId=0)	Queries the DO configuration permission	num: DO number (not exceeding total number of DO) val: Configuration permission, representing the result of the query, 1-Permission granted to RC, 0-Permission granted to PLC	0: Success; <0: Failure	
86	int **IMC100_Get_DOGroupCfg**(int num, int *val, int comId=0)	Queries the configuration permission for each group of DOs	num: DO group number val: Configuration permission, representing the result of the query, bit0-bit7 represents the configuration permission for each DO in the group, 1-Permission granted to RC, 0-Permission granted to PLC	0: Success; <0: Failure	
87	int **IMC100_Get_DO**(int num, int *sts, int comId=0)	Queries the DO status by group	num: DO number (not exceeding the total number of DO's) sts: DO status, representing the result of the query, 0-Off, 1-On	0: Success; <0: Failure	
88	int **IMC100_Get_DOGroup**(int num, int *sts, int comId=0)	Queries the DO status by group	num: DO group number sts: The DO status of each group, range 0-255, where bit0-bit7 corresponds to the DO status with the lowest to highest group number	0: Success; <0: Failure	
89	int **IMC100_Get_DACfg**(int num, int *val, int comId=0)	Queries the DA configuration permission	num: DA number val: Configuration permission, representing the result of the query, 1-Permission granted to RC, 0-Permission not granted to RC (namely, the permission is granted to PLC, or no connection is available)	0: Success; <0: Failure	
90	int **IMC100_Get_DA**(int num, float *val, int comId=0)	Queries the output value of DA by number	num: DA number (not exceeding total number of DA) val: DA value, representing the result of the query, current type in mA and	0: Success; <0: Failure	

			voltage type in V		
91	int **IMC100_Get_DevSts**(int sts[6], int comId=0)	Queries the connection status of the system devices	sts[]: System device connection status , representing the result of the query sts[0]: NIC 1 status, 0-Not connected, 1-Connected, 2-Disabled; sts[1]: NIC 2 status, as NIC1; sts[2]: USB device status, 0-Not connected, 1-Connected and mounted successfully, 2-Mounting failed; sts[3]: Memory card status, 0-Not connected, 1-Connected and mounted successfully, 2-Mounting failed, 3-File system format error; sts[4]: EtherCAT0 communication status, 0-Normal, 1-Slave disconnected, 2-Network cable not connected, 3-Connected to non-ECAT devices, 4-Disabled; sts[5]: IRLink0 communication status, as EtherCAT0	0: Success; ＜0: Failure	
92	int **IMC100_Get_FwVersion**(char ver[32], int comId=0)	Queries the system controller software version	ver[]: Current system software version, representing the result of a query, such as S03.20R	0: Success; ＜0: Failure	
93	int **IMC100_Get_SysTime**(char time[16], int comId=0)	Queries the current system time	time[]: A time string (YYYY-mm-dd-hours-seconds), representing the result of the query	0: Success; ＜0: Failure	
94	int **IMC100_Get_RobotType**(char type[128], int comId=0)	Queries the current system model	type[]: A model string, representing the result of the query, for example Scara_A_Ino1	0: Success; ＜0: Failure	
95	int **IMC100_Get_ArmType**(double pos[6], int armType[4], int comId=0)	Queries the arm parameters based on the joint coordinate values	pos[]: Joint coordinate value armType[]: Arm parameter, representing the result of the query	0: Success; ＜0: Failure	

96	int **IMC100_Get_TransArmType**(int armType[4], int transArmType[4], int comId=0)	Converts arm parameters for a point in system earlier than V18 to arm parameters in system above V18 (for 6-axis robots only)	armType[]: Arm parameters, in system earlier than V18 transArmType[]: Converted arm parameters, representing the result of the query	0: Success; ＜0: Failure	
97	int **IMC100_Get_ServoSts**(int sts[8], int comId=0)	Queries the error status of all servos in the system (both robot axes and external axes)	sts[8]: Fault status of the servo, representing the result of the query Currently up to 8 servo axes are supported, sts[0] corresponds to servo #0, and so on, 0-No fault, bit0-Servo alarm, Bit1-Servo warning	0: Success; ＜0: Failure	
98	int **IMC100_Get_ServoErr**(int num, int *error, int comId=0)	Queries the error code for a single servo (robot axis)	num: Servo axis number, starting from 0 error: Servo error code, representing the result of the query	0: Success; ＜0: Failure	
99	int **IMC100_Get_StrPara**(float para[6], int comId=0)	Queries the robot structure parameters	para[]: Structure parameters, representing the result of the query (for SCARA robots, para[0]-para[3] are valid, for 6-axis robots, Para[0]-para[5] are valid, same as below)	0: Success; ＜0: Failure	
100	int **IMC100_Set_StrPara**(float para[6], int comId=0)	Sets the robot structure parameters	para[]: Structure parameters	0: Success; ＜0: Failure	Can be used in Manager mode and above
101	int **IMC100_Get_StrParaComp**(float para[6], int comId=0)	Queries the robot structure compensation parameters	para[]: Structure compensation parameters, representing the result of the query	0: Success; ＜0: Failure	
102	int **IMC100_Set_StrParaComp**(float para[6], int comId=0)	Sets the robot structure compensation parameters	para[]: Structure compensation parameters	0: Success; ＜0: Failure	Can be used in Manager mode and above

103	int **IMC100_Get_RdctRatio**(float para[6], int comId=0)	Queries the reduction ratio of joints	para[]: Reduction ratio of joints, representing the result of the query	0: Success; <0: Failure	
104	int **IMC100_Set_RdctRatio**(float para[6], int comId=0)	Sets the reduction ratio of joints	para[]: Reduction ratio of joints	0: Success; <0: Failure	Can be used in Manager mode and above
105	int **IMC100_Get_CpParaM**(float para[6], int comId=0)	Queries the main coupling parameters of the joints	para[]: Main coupling parameter of each joint, representing the result of the query	0: Success; <0: Failure	
106	int **IMC100_Set_CpParaM**(float para[6], int comId=0)	Sets the main coupling parameters of the joints	para[]: Main coupling parameter of each joint	0: Success; <0: Failure	Can be used in Manager mode and above
107	int **IMC100_Get_CpParaS**(float para[6], int comId=0)	Queries the secondary coupling parameters of the joints	para[]: Secondary coupling parameter of each joint, representing the result of the query	0: Success; <0: Failure	
108	int **IMC100_Set_CpParaS**(float para[6], int comId=0)	Sets the secondary coupling parameters of the joints	para[]: Secondary coupling parameter of each joint	0: Success; <0: Failure	Can be used in Manager mode and above
109	int **IMC100_Get_HomePos**(int num, double pos[6], int comId=0)	Queries the work origin	num: Work origin number, range 0-4 pos[]: Joint coordinate value corresponding to the work origin, representing the result of the query	0: Success; <0: Failure	
110	int **IMC100_Set_HomePos**(int num, double pos[6], int comId=0)	Sets the work origin	num: Work origin number, range 0-4 pos[]: Joint coordinate value corresponding to the work origin	0: Success; <0: Failure	Can be used in Manager mode and above
111	int **IMC100_Get_ZeroPos**(int pluse[6], int comId=0)	Queries the absolute zero point	plus[]: The pulse value for the absolute zero point, representing the result of the query	0: Success; <0: Failure	

112	int **IMC100_Set_ZeroPos**(int pluse[6], int comId=0)	Sets the absolute zero point	plus[]: The pulse value for the absolute zero point	0: Success; ＜0: Failure	Can be used in Manager mode and above
113	int **IMC100_Get_InchStep**(int *val, int comId=0)	Queries the step size of jog motion	val: Step size of jog motion, representing the result of the query	0: Success; ＜0: Failure	
114	int **IMC100_Get_StepMotionJ**(float *para, int comId=0)	Queries the joint step size of jog motion in the teaching mode	para: Joint step size, representing the result of the query	0: Success; ＜0: Failure	
115	int **IMC100_Set_StepMotionJ**(float para, int comId=0)	Sets the joint step size of jog motion in the teaching mode	para: Joint step size	0: Success; ＜0: Failure	Can be used in Manager mode and above
116	int **IMC100_Get_StepMotionL**(float *para, int comId=0)	Queries the linear step size of jog motion in the teaching mode	para: Linear step size, representing the result of the query	0: Success; ＜0: Failure	
117	int **IMC100_Set_StepMotionL**(float para, int comId=0)	Sets the linear step size of jog motion in the teaching mode	para: Linear step size	0: Success; ＜0: Failure	Can be used in Manager mode and above
118	int **IMC100_Get_TeachVelLimJ**(float para[6], int comId=0)	Queries the upper limit of joint speed during teaching	para[]: Maximum allowable joint speed, representing the result of the query	0: Success; ＜0: Failure	
119	int **IMC100_Set_TeachVelLimJ**(float para[6], int comId=0)	Sets the upper limit of joint speed during teaching	para[]: Maximum allowable joint speed	0: Success; ＜0: Failure	Can be used in Manager mode and above
120	int **IMC100_Get_TeachVelLimL**(float para[2], int comId=0)	Queries the upper limit of position/orientation speed during teaching	para[2]: Maximum allowable position/orientation speed, representing the result of the query	0: Success; ＜0: Failure	
121	int **IMC100_Set_TeachVelLimL**(float para[2], int comId=0)	Sets the upper limit of position/orientati	para[2]: Maximum allowable position/orientation speed	0: Success; ＜0:	Can be used in Manager

		on speed during teaching		Failure	mode and above
122	int **IMC100_Get_TeachAccLimJ**(float para[6], int comId=0)	Queries the upper limit of joint acceleration during teaching	para[]: Maximum allowable joint acceleration, representing the result of the query	0: Success; <0: Failure	
123	int **IMC100_Set_TeachAccLimJ**(float para[6], int comId=0)	Sets the upper limit of joint acceleration during teaching	para[]: Maximum allowable joint acceleration	0: Success; <0: Failure	Can be used in Manager mode and above
124	int **IMC100_Get_TeachAccLimL**(float para[2], int comId=0)	Queries the upper limit of position/orientation acceleration during teaching	para[2]: Maximum allowable position/orientation acceleration, representing the result of the query	0: Success; <0: Failure	
125	int **IMC100_Set_TeachAccLimL**(float para[2], int comId=0)	Sets the upper limit of position/orientation acceleration during teaching	para[2]: Maximum allowable position/orientation acceleration	0: Success; <0: Failure	Can be used in Manager mode and above
126	int **IMC100_Get_RunVelLimJ**(float para[6], int comId=0)	Queries the upper limit of joint speed during operation	para[]: Maximum allowable joint speed, representing the result of the query	0: Success; <0: Failure	
127	int **IMC100_Set_RunVelLimJ**(float para[6], int comId=0)	Sets the upper limit of joint speed during operation	para[]: Maximum allowable joint speed	0: Success; <0: Failure	Can be used in Manager mode and above
128	int **IMC100_Get_RunVelLimL**(float para[2], int comId=0)	Queries the upper limit of position/orientation speed during operation	para[2]: Maximum allowable position/orientation speed, representing the result of the query	0: Success; <0: Failure	
129	int **IMC100_Set_RunVelLimL**(float para[2], int comId=0)	Sets the upper limit of position/orientation speed during operation	para[2]: Maximum allowable position/orientation speed	0: Success; <0: Failure	Can be used in Manager mode and above
130	int **IMC100_Get_RunAccLimJ**(float para[6], int comId=0)	Queries the upper limit of joint acceleration during operation	para[]: Maximum allowable joint acceleration, representing the result of the query	0: Success; <0: Failure	

131	int **IMC100_Set_RunAccLimJ**(float para[6], int comId=0)	Sets the upper limit of joint acceleration during operation	para[]: Maximum allowable joint acceleration	0: Success; ＜0: Failure	Can be used in Manager mode and above
132	int **IMC100_Get_RunAccLimL**(float para[2], int comId=0)	Queries the upper limit of position/pose acceleration during operation	para[2]: Maximum allowable position/pose acceleration, representing the result of the query	0: Success; ＜0: Failure	
133	int **IMC100_Set_RunAccLimL**(float para[2], int comId=0)	Sets the upper limit of position/pose acceleration during operation	Para[2]: Maximum allowable position/pose acceleration	0: Success; ＜0: Failure	Can be used in Manager mode and above
134	int **IMC100_Get_StopDecLimJ**(float para[6], int comId=0)	Queries the upper limit of joint deceleration during operation	para[]: Maximum allowable joint deceleration, representing the result of the query	0: Success; ＜0: Failure	
135	int **IMC100_Set_StopDecLimJ**(float para[6], int comId=0)	Sets the upper limit of joint acceleration during operation	Para[]: Maximum allowable joint deceleration	0: Success; ＜0: Failure	Can be used in Manager mode and above
136	int **IMC100_Get_StopDecLimL**(float para[2], int comId=0)	Queries the upper limit of position/pose acceleration during operation	para[2]: Maximum allowable position/pose acceleration, representing the result of the query	0: Success; ＜0: Failure	
137	int **IMC100_Set_StopDecLimL**(float para[2], int comId=0)	Sets the upper limit of position/pose acceleration during operation	Para[]: Maximum permissible position/pose deceleration	0: Success; ＜0: Failure	Can be used in Manager mode and above
138	int **IMC100_Get_ZonePara**(float para[2], int comId=0)	Queries the transition accuracy parameters	para[]: Linear and joint transition accuracy, representing the result of the query	0: Success; ＜0: Failure	
139	int **IMC100_Set_ZonePara**(float para[2], int comId=0)	Sets the transition accuracy parameters, including linear transition	para[]: Linear and joint transition accuracy	0: Success; ＜0: Failure	Can be used in Manager mode and above

		accuracy and joint transition accuracy			
140	int **IMC100_Get_AxisNLim**(int axis, float *para, int comId=0)	Queries the negative axis limit of the robot axis	axis: Axis number, depending on the number of axes, range 1-6, corresponding to J1-J6 axis para: Negative axis limit, representing the result of the query	0: Success; ＜0: Failure	
141	int **IMC100_Set_AxisNLim**(int axis, float para, int comId=0)	Sets the negative axis limit of the robot axis	axis: Axis number, range 1-6, corresponding to J1-J6 axis para: Negative axis limit	0: Success; ＜0: Failure	Can be used in Manager mode and above
142	int **IMC100_Get_AxisPLim**(int axis, float *para, int comId=0)	Queries the positive axis limit of the robot axis	axis: Axis number, range 1-6, corresponding to J1-J6 axis para: Positive axis limit, representing the result of the query	0: Success; ＜0: Failure	
143	int **IMC100_Set_AxisPLim**(int axis, float para, int comId=0)	Sets the positive axis limit of the robot axis	axis: Axis number, range 1-6, corresponding to J1-J6 axis para: Positive axis limit	0: Success; ＜0: Failure	Can be used in Manager mode and above
144	int **IMC100_Get_ToolC**(int num, double pos[6], int comId=0)	Queries the tool coordinate system parameters	num: Tool number, range 1-15 pos[]: Tool coordinate system parameters, representing the result of the query	0: Success; ＜0: Failure	
145	int **IMC100_Set_ToolC**(int num, double pos[6], int comId=0)	Sets the tool coordinate system parameters	num: Tool number, range 1-15 pos[]: Tool coordinate system parameters	0: Success; ＜0: Failure	Can be used in Manager mode and above
146	int **IMC100_Get_UserC**(int num, double pos[6], int comId=0)	Queries the user coordinate system parameters	num: User number, range 1-15 pos[]: User coordinate system parameters, representing the result of the query	0: Success; ＜0: Failure	

147	int **IMC100_Set_UserC**(int num, double pos[6], int comId=0)	Sets the user coordinate system parameters	num: User number, range 1-15 pos[]: User coordinate system parameters	0: Success; <0: Failure	Can be used in Manager mode and above
148	int **IMC100_Get_ToolCNum**(int *num, int comId=0)	Queries the current tool coordinate system number	num: Currently selected tool number, representing the result of the query	0: Success; <0: Failure	
149	int **IMC100_Set_ToolCNum**(int num, int comId=0)	Sets the current tool coordinate system number	num: Tool number	0: Success; <0: Failure	Can be used in Manager mode and above
150	int **IMC100_Get_UserCNum**(int *num, int comId=0)	Queries the current user coordinate system number	num: Currently selected user number, representing the result of the query	0: Success; <0: Failure	
151	int **IMC100_Set_UserCNum**(int num, int comId=0)	Sets the current user coordinate system number	num: User number	0: Success; <0: Failure	Can be used in Manager mode and above
152	int **IMC100_Set_Coord**(int type, int comId=0)	Sets the current coordinate system type	type: Current coordinate system type, range 1 to 4, 1-Joint coordinate system, 2-Base coordinate system, 3-Tool coordinate system, 4-User coordinate system	0: Success; <0: Failure	
153	int **IMC100_Get_Interf**(int num, double pos[6], int comId=0)	Queries the coordinate parameters of the boundary points in the interference area	num: Interference area number, range 0 to 7 pos[]: Coordinates of boundary points in the interference area, representing the result of the query, pos[0] to pos[2] correspond to the XYZ coordinates of point 1, Pos[3] to pos[5] correspond to the XYZ coordinates of point 2	0: Success; <0: Failure	
154	int **IMC100_Set_Interf**(int num, double pos[6], int comId=0)	Sets the coordinate parameters of the	num: Interference area number pos[]: Coordinates of the	0: Success; <0:	Can be used in Manager

		boundary points in the interference area	boundary points in the interference area	Failure	mode and above
155	int **IMC100_Get_CurInterf**(int *num, int comId=0)	Queries the number of active area that is currently active	num: Interference area number, representing the result of the query, range 0 to 255, where bit0 to bit7 correspond to the interference area 0 to the interference area 7, 0-Inactive, 1-Active	0: Success; ＜0: Failure	
156	int **IMC100_Set_CurInterf**(int num, int comId=0)	Sets the number of the interference area that needs to be activated	num: Interference area number, as above	0: Success; ＜0: Failure	Can be used in Manager mode and above
157	int **IMC100_Get_JumpPara**(float *lh, float *mh, float *rh, int comId=0)	Queries the height parameters of the jump motion (Jump, JumpL)	lh: Height raised relative to the starting position, range 0 to 2000, representing the query of the result mh: Height of the highest point of motion relative to zero point of the base coordinate system, range -2000 to 2000, representing the query of the result rh: Height dropped when reaching the destination position, representing the query of the result	0: Success; ＜0: Failure	
158	int **IMC100_Set_JumpPara**(float lh, float mh, float rh, int comId=0)	Sets the height parameters of the jump motion (Jump, JumpL)	lh: Height raised relative to the starting position, range 0 to 2000, mh: Height of the highest point of motion relative to zero point of the base coordinate system, range -2000 to 2000 rh: Height dropped when reaching the destination position	0: Success; ＜0: Failure	Only effective in data streaming mode.
159	int **IMC100_Get_PalletPara**(int *rowNum, int *colNum, int *layerNum, double *layerHeight, int	Queries the pallet parameters	rowNum: Row number, range 0 to 1000 colNum: Columns number,	0: Success; ＜0:	

	comId=0)		range 0 to 1000 layerNum: Layer number, range 0 to 1000 layerHeight: Layer height, unit: mm	Failure	
160	int **IMC100_Set_PalletPara**(int *rowNum, int *colNum, int *layerNum, double *layerHeight, int comId=0)	Sets the pallet parameters	rowNum: Row number, range 0 to 1000 colNum: Columns number, range 0 to 1000 layerNum: Layer number, range 0 to 1000 layerHeight: Layer height, unit: mm	0: Success; ＜0: Failure	
161	int **IMC100_Clear_PalletPara**(int comId=0)	Clears the tray parameters	Clears the tray parameters	0: Success; ＜0: Failure	
162	int **IMC100_Get_PalletPoint**(ROBOT_POS pos1, ROBOT_POS pos2, ROBOT_POS pos3, int rowIndex, int colIndex, int layIndex, ROBOT_POS *posDst, int comId=0)	Queries corresponding pallet points (3 points defining the boundary of pallet)	pos1-3: Three points that define the pallet. rowIndex: The row number of point to be queried. colIndex: The column number of point to be queried. layIndex: The layer number of point to be queried. PosDst: The result of point query.	0: Success; ＜0: Failure	The calculation results are determined by the number of the tool coordinate system of the first point.
163	int **IMC100_Get_Pallet4Point**(ROBOT_POS pos1, ROBOT_POS pos2, ROBOT_POS pos3, ROBOT_POS pos4, int rowIndex, int colIndex, int layIndex, ROBOT_POS *posDst, int comId=0)	Queries corresponding pallet points (4 points defining the boundary of pallet)	pos1-3: Four points that define the pallet. rowIndex: The row number of point to be queried. colIndex: The column number of point to be queried. layIndex: The layer number of point to be queried. PosDst: The result of point query.	0: Success; ＜0: Failure	The calculation results are determined by the number of the tool coordinate system of the first point.
164	int **IMC100_SavePara**(int comId=0)	Saves the system parameters, retentive upon		0: Success; ＜0:	Can be used in Manager

		power failure		Failure	mode and above
165	int **IMC100_RecoverPara**(int comId=0)	Restores the system parameters to the last saved ones		0: Success; <0: Failure	Can be used in Manager mode and above
166	int **IMC100_Get_P**(int pNum, ROBOT_POS *pos, int comId=0)	Queries the position parameters corresponding to the global position variable	pNum: Global position variable number, range 0-1000 pos: Position parameter structure, representing the result of the query	0: Success; <0: Failure	
167	int **IMC100_Set_P**(int pNum, ROBOT_POS *pos, int comId=0)	Sets the position parameters corresponding to the global position variable	pNum: Global position variable number, range 0-1000 pos: Position parameter structure	0: Success; <0: Failure	Can be used in Editor mode and above
168	int **IMC100_Set_Phere**(int pNum, int comId=0)	Sets the global position parameters with the parameters of the current point	pNum: Global position variable number	0: Success; <0: Failure	Can be used in editor mode and above
169	int **IMC100_Get_PR**(int prNum, ROBOT_POS *pos, int comId=0)	Queries the parameters for the global translation variable	prNum: Global translation variable number, range 0 to 255 pos: Position parameter structure, representing the result of the query, where the arm parameters are invalid	0: Success; <0: Failure	
170	int **IMC100_Set_PR**(int prNum, ROBOT_POS pos, int comId=0)	Sets the parameters for the global translation variable	prNum: Global translation variable number, range 0 to 255 pos: Position parameter structure, where the arm parameters are invalid	0: Success; <0: Failure	Can be used in Editor mode and above
171	int **IMC100_WriteFile_PR**(int comId=0)	Saves all PR variables, retentive upon power failure		0: Success; <0: Failure	Can be used in Editor mode and above
172	int **IMC100_Get_B**(int num, int *val, int comId=0)	Queries the value of the global B variable	num: B variable number val: B variable value, representing the result of the	0: Success; <0:	

			query	Failure	
173	int **IMC100_Set_B**(int num, int val, int comId=0)	Sets the value of the global B variable	num: B variable number val: B variable value, range 0-255	0: Success; ＜0: Failure	Can be used in Editor mode and above
174	int **IMC100_Get_R**(int num, int *val, int comId=0)	Queries the value of the global R variable	num: R variable number val: R variable value, representing the result of the query	0: Success; ＜0: Failure	
175	int **IMC100_Set_R**(int num, int val, int comId=0)	Sets the value of the global R variable	num: B variable number val: B variable value, range -65536 to 65535	0: Success; ＜0: Failure	Can be used in Editor mode and above
176	int **IMC100_Get_D**(int num, double *val, int comId=0)	Queries the value of the global D variable	num: D variable number val: D variable value, representing the result of the query	0: Success; ＜0: Failure	
177	int **IMC100_Set_D**(int num, double val, int comId=0)	Sets the value of the global D variable	num: D variable number val: D variable value, range -9999999.999 to 999999999.999	0: Success; ＜0: Failure	Can be used in Editor mode and above
178	int **IMC100_Get_ModbusCoil**(int address, int sum, int *val, int comId=0)	Queries the coil value of the Modbus variable area	address: Modbus area coil address, range 0-8191 sum: Total number of coils read, range 1-8 val: Coil value, representing the result of the query	0: Success; ＜0: Failure	
179	int **IMC100_Set_ModbusCoil**(int address, int sum, int val, int comId=0)	Sets the coil value of the Modbus variable area	address: Modbus area coil address, range 2048-4095, 6144-8191 sum: Total number of coils read, range 1-8 val: Coil value	0: Success; ＜0: Failure	Can be used in Editor mode and above
180	int **IMC100_Get_ModbusRegUshort**(int address, int sum, unsigned short val[], int comId=0)	Queries the register value of the Modbus variable area, with the data type being unsigned short	address: Modbus area register address, range 0-65535 sum: Total number of registers read, range 1-8 val: Represents the result of the query	0: Success; ＜0: Failure	

181	int **IMC100_Set_ModbusRegUshort**(int address, int sum, unsigned short val[], int comId=0)	Sets the register value of the Modbus variable area, with the data type being unsigned short	address: Modbus area register address, range 16384-32767, 49152-65535 sum: Total number of registers read, range 1-8 val: Represents the result of the query	0: Success; ＜0: Failure	Can be used in Editor mode and above
182	int **IMC100_Get_ModbusRegFloat**(int address, int sum, float val[], int comId=0)	Queries the register value of the Modbus variable area, with the data type being float	address: Modbus area register address, range 0-65535 sum: Total number of registers read, range 1-8 val: Represents the result of the query (one float data occupies 2 registers)	0: Success; ＜0: Failure	
183	int **IMC100_Set_ModbusRegFloat**(int address, int sum, float val[], int comId=0)	Sets the register value of the Modbus variable area, with the data type being float	address: Modbus area register address, range 16384-32767, 49152-65535 sum: Total number of registers read, range 1-8 val: Represents the result of the query	0: Success; ＜0: Failure	Can be used in Editor mode and above
184	int **IMC100_Get_PlcVarByte**(int num, unsigned char *val, int comId=0)	Queries the value of a Byte-type PLC variable	num: Byte variable number, range 0-255 val: Variable value, representing the result of the query	0: Success; ＜0: Failure	
185	int **IMC100_Get_PlcVarInt**(int num, short *val, int comId=0)	Sets the value of an Int-type PLC variable	num: Int variable number, range 0-255 val: Variable value, representing the result of the query	0: Success; ＜0: Failure	
186	int **IMC100_Get_PlcVarDInt**(int num, int *val, int comId=0)	Sets the value of an DInt-type PLC variable	num: DInt variable number, range 0-255 val: Variable value, representing the result of the query	0: Success; ＜0: Failure	
187	int **IMC100_Get_PlcVarLReal**(int num, double *val, int comId=0)	Queries the value of a LReal-type PLC variable	num: LReal variable number, range 0-255 val: Variable value, representing the result of the query	0: Success; ＜0: Failure	
188	int **IMC100_Get_UserAlarm**(int num, char alarm[40], int comId=0)	Queries the contents of a	num: Custom alarm number, range 0-15	0: Success;	

		custom alarm	alarm: Description of alarm, representing the result of the query, with a length of 40 bytes or less	＜0: Failure	
189	int **IMC100_Set_UserAlarm**(int num, char alarm[40], int comId=0)	Sets the contents of a custom alarm	num: Custom alarm number, range 0-15 alarm: Description of alarm, with a length of 40 bytes or less	0: Success; ＜0: Failure	Can be used in Manager mode and above
190	int **IMC100_Get_Print**(char val[120], int comId=0)	Queries the controller print information, including printed contents of the print instructions and the system error message	val: Printed contents, representing the result of the query, with a length of 120 bytes or less	0: Success; ＜0: Failure	
191	int **IMC100_Get_InCfg**(int func, int *diNum, int comId=0)	Queries the DI number corresponding to the query input function	func: Enter function number, 0-Start, 1-Stop, 2-Program reset, 3-Emergency stop, 4-Clear alarm, 5-Increase speed, 6-Decrease speed diNum: DI number, representing the result of the query, -1 means that the corresponding DI is not set, and the other range is 0-15	0: Success; ＜0: Failure	
192	int **IMC100_Set_InCfg**(int func, int diNum, int comId=0)	Sets the DI number corresponding to the query input function	func: Input function number, 0-Start, 1-Stop, 2-Program reset, 3-Emergency stop, 4-Clear alarm, 5-Increase speed, 6-Decrease speed diNum: DI number, range 0-15, -1 means that the corresponding DI is not set	0: Success; ＜0: Failure	Can be used in Manager mode and above
193	int **IMC100_Get_OutCfg**(int func, int *doNum, int comId=0)	Queries the DO number corresponding to the query input function	func: Output function number, 0-Alarm, 1-Run, 2-Stop, 3-Start completed, 4-Enable, 5-Reset successfully diNum: DO number, representing the result of the query, -1 means that the corresponding DO is not set,	0: Success; ＜0: Failure	

			and the other range is 0-15		
19 4	int **IMC100_Set_OutCfg**(int func, int doNum, int comId=0)	Sets the DO number corresponding to the query input function	func: Output function number, 0-Alarm, 1-Run, 2-Stop, 3-Start completed, 4-Enable, 5-Reset successfully diNum: DO number, range 0-15, -1 means that the corresponding DO is not set	0: Success; ＜0: Failure	Can be used in Manager mode and above
19 5	int **IMC100_CurCtrlDev**(int *dev, int comId=0)	Queries the device to which the current control permission belongs	dev: Number of current control device, 0-InoTeachPad, 1-InoRobShop, 2-Remote Ethernet, 3-Remote I/O, 4-Remote Modbus	0: Success; ＜0: Failure	
19 6	int **IMC100_CurPermit**(int *owner, unsigned int *ipAddr, unsigned short *ipPort, int comId=0)	Queries for information about the Ethernet device that currently has control permission	owner: Identity of Ethernet device that has the control permission, representing the result of the query, 0-No Ethernet device granted permission, 1-Current device granted permission, 2-Other Ethernet device granted permission. IpAddr: Device IP address, representing the result of the query. When the first return value is 0, this value is meaningless. ipPort: The device port number, representing the result of the query.	0: Success; ＜0: Failure	
19 7	int **IMC100_AcqPermit**(int cmd=0, int comId=0)	Current API network client device requests to obtain control permission	cmd: Request command, 0 for general request for permission, 1 for preemption of permission, default is 0	0: Success; ＜0: Failure	
19 8	int **IMC100_RemovePermit**(int comId=0)	Current API network client device requests to release control permission		0: Success; ＜0: Failure	

199	int **IMC100_CurUserType**(int *type, int comId=0)	Queries the current user mode	type: User mode, representing the result of the query, 0-User, 1-Editor, 2-Manager, 3-Factory	0: Success; ＜0: Failure	
200	int **IMC100_UserLogin**(int type, char password[8], int comId=0)	The user mode to which the current API network client device logs	type: User mode, 0-User, 1-Editor, 2-Manager, 3-Factory	0: Success; ＜0: Failure	
201	int **IMC100_UserLogout**(int comId=0)	The current API network client device exits the current login mode and returns to the default User mode		0: Success; ＜0: Failure	
202	int **IMC100_Set_SysTime**(char time[16], int comId=0)	Sets the current system clock	Time: A time string (yyyy-mm-dd, hh:mm:ss), with a length of 14 characters	0: Success; ＜0: Failure	Can be used in Manager mode and above
203	int **IMC100_LatchEnable**(int cmd, int comId=0)	Enables or disables the position latch function	cmd: Control command, 1-Enable, 0-Disable	0: Success; ＜0: Failure	
204	int **IMC100_Get_LatchSts**(int *sts, int comId=0)	Queries the status of the position latch function	sts: Latch function status, 1-Enabled, 0-Disabled	0: Success; ＜0: Failure	
205	int **IMC100_Get_LatchSum**(int *sum, int comId=0)	Queries the total number of latched position points	sum: Query result	0: Success; ＜0: Failure	
206	int **IMC100_Get_LatchPos**(int index, int *sts, double pos[6], int comId=0)	Reads the latched positions (read in sequence, each position can only be read once)	index: Reserved sts: Returned status, indicating whether there is latched data (0-No, 1-Yes) pos: Returned latched value, the joint coordinates	0: Success; ＜0: Failure	
207	int **IMC100_Clr_LatchPos**(int comId=0)	Clears the latch positions		0: Success; ＜0: Failure	

Structure	Description
typedef struct{ double pos[6]; int armType[4]; int coord; int toolNo; Int userNo; }ROBOT_POS;	 Coordinate values of robot position, 64 bits Arm parameter Coordinate system type (1-Joint, 2-Cartesian) Tool coordinate number User coordinate number
typedef struct{ int IONo; int IOVa; int Kind; double Value; }MOV_IO;	 I/O number (0–63) Output value of I/O (0-OFF, 1-ON) Type of set I/O in motion (0-Time, 1-Path percentage, 2-Distance) When the type is Time, a value greater than 0 means the signal is outputted after robot moves for "Value" seconds, a value less than or equal to 0 means the output signal is outputted "Value" seconds before the robot reaches the target point; When the type is Path Percentage, the signal is output when the robot moves by "Value"% of the entire path from the start point to the end point; When type is Distance, a value greater than 0 means the signal is outputted after the robot moves by "value" mm from the start point, and a value less than or equal to 0 indicates that the signal is outputted "value" mm before the robot reaches the end point.

(2) API Connection Failure Table

No.	Function Return Value	Fault Print Information	Description	Solution
0	0	/	Instruction is normal	/
1	-1	e1:syntax error	The instruction has a syntax error.	The version of the dynamic link library is incorrect. Ask for technical support.
2	-2	e2:number of parameter	The number of	The version of the

		unmatched	instruction parameters does not match.	dynamic link library is incorrect. Ask for technical support.
3	-3	e3:parameter value illegal	The instruction parameter value is not reasonable.	Reset the function parameters according to the function description.
4	-4	e4:not allowed in current mode	This instruction is not allowed in current mode.	Switch to teach or play mode and then recall the instruction according to the actual situation.
5	-5	e5:not allowed when robot is running	This instruction is not allowed while the robot is running.	Recall the instruction after the robot stops.
6	-6	e6:system in emergency	The system is in an emergency stop.	Release the emergency stop and recall the instruction.
7	-7	e7:system fault	The system is faulty.	Recall the instruction after clearing system fault.
8	-8	e8:motion mode closed	The data streaming mode is closed.	Recall the instruction after opening the data streaming mode.
9	-9	E9: Motor off	The motor is not enabled.	Recall the instruction after enabling the motor.
10	-10	e10:rsv	Reserved	Reserved
11	-11	e11:instruction unfinished	The motion instruction buffer still contains unfinished instructions.	Recall the instruction when the motion instruction buffer is free.
12	-12	e12:output unavailable	User does not have permission to control the output port.	Recall the instruction after obtaining permission for the port, or use other available ports.
13	-13	e13:read AD failed	Failed to read AD channel.	Confirm that the module is configured correctly and then recall the instruction.
14	-14	e14:write DA failed	Failed to write the AD channel.	Confirm that the module is configured correctly and then recall the instruction.
15	-15	e15:write DO failed	Failed to write the DO channel.	Confirm that the module is configured correctly and then recall the instruction.

16	-16	e16:command invalid	Pause instruction is invalid.	The pause function cannot be not called when the program is stopped.
17	-17	e17:not allowed in current coordinate	The instruction is not allowed under the current coordinate system.	Recall the instruction after switching the coordinate system.
18	-18	e18:mode conflict	Motion mode conflict.	Recall the function after closing the data streaming mode.
19	-19	e19:rsv	Reserved	Reserved
20	-20	e20:program non-existent	The program path does not exist.	Re-enter the path or re-edit the teaching program.
21	-21	e21:point non-existent	Position point P or offset LPR does not exist.	Call the instruction after confirming that the points are correct.
22	-22	e22:calc error	Internal calculation error.	Make sure that the robot is not at a singular position and recall the instruction.
23	-23	e23:rsv	Reserved	Reserved
24	-24	e24:without permit	The current Ethernet device is not granted the control permission.	Recall the instruction after obtaining the control permission.
25	-25	e25:ETH without authorization	The current control device is not an Ethernet device.	Recall the instruction after switching the control permission.
26	-26	e26:rsv	Reserved	Reserved
27	-27	e27:low user grade	The current user privilege level is low.	Recall the instruction after login as a user with higher privilege.
28	-28	e28:permit occupied	The control permission has been granted to another Ethernet device.	Recall the instruction after obtaining the control permission.
29	-29	e29:internal fault	Internal call error	The version of the dynamic link library is incorrect. Ask for technical support.
30	-30	e30:modbus unavailable	Modbus slave is not configured.	Configure Modbus slave and recall the instruction.
31	-31	e31:condition unfulfilled to write	System parameters cannot be set at this time.	Recall the instruction after the robot stops.
32	-32	e32:rsv	Reserved	Reserved
33	-33	e33:pallet para err	Pallet parameter error	33

33	-253	/	Internal calculation exception	The version of the dynamic link library is incorrect. Ask for technical support.
34	-254	/	The input function parameters are incorrect.	Reset the function parameters according to the function description.
35	-255	/	The network communication is abnormal.	Check that the network connection is correct and recall the instruction.

Appendix 3: Modbus Slave Address Table

Note: The following Modbus slave address table is only applicable for S03.20R version (or version for which the legacy address table feature is enabled)

- **Read-only (4096) physical discrete input, parameter: 0x02**

Address DEC	**Address HEX**	**Variable Name**	**Data Type**	**Description**	**Remarks**
0	0x0000	QW65024,bit0	Bit	Enable status (0-OFF, 1-ON)	
1	0x0001	QW65024,bit1	Bit	Program status (0-Non-running, 1-Running)	This coil value is also 0 when the system is paused.
2	0x0002	QW65024,bit2	Bit	Emergency stop status (0-OFF, 1-ON)	
3	0x0003	QW65024,bit3	Bit	System fault status (0-No fault, 1-Fault)	
4	0x0004	QW65024,bit4	Bit	Servo fault (0-No fault, 1-Fault)	
5	0x0005	QW65024,bit5	Bit	System startup completion status (0-Incomplete, 1-Complete)	
6	0x0006	QW65024,bit6	Bit	Program reset completion status (main task and dynamic task return to start line)	1 indicates a successful reset; 0 when program is running.
7	0x0007	QW65024,bit7	Bit	System warning status (0-No warning, 1-Warning)	
8	0x0008	QW65024,bit8	Bit	Servo warning (0-No warning, 1-Warning)	
9	0x0009	QW65024,bit9	Bit	Communication heartbeat	A 0-1-0 value indicates normal communication and a constant value indicates abnormal

Address	Address	Variable Name	Data	Description	Remarks
					communication.
10	0x000A	QW65024,bit10	Bit	Safety door warning status	0-No warning, 1-Warning (An alarm is generated only when the safety door is opened in the play mode.)
…	…	….	Bit	Reserved	
18	0x0012	QW65025,bit2	Bit	Direct motion status (0-Invalid or Not arrived, 1-Arrived)	
19	0x0013	QW65025,bit3	Bit	P variable read (0-Failure, 1-Success)	
20	0x0014	QW65025,bit4	Bit	P variable writ (0-Failure, 1-Success)	
21	0x0015	QW65025,bit5	Bit	Reserved	
22	0x0016	QW65025,bit6	Bit	P variable batch read (0-Failure, 1-Success)	1 only if all variables are read successfully.
…	…	….	Bit	Reserved	
64	0x0040	QW65028,bit0	Bit	IN[000] status (0-OFF, 1-ON, same below)	When the corresponding IN signal is not supported by the I/O module, the initialized value OFF is displayed by default, same below.
65	0x0041	QW65028,bit1	Bit	IN[001]	
66	0x0042	QW65028,bit2	Bit	IN[002]	
67	0x0043	QW65028,bit3	Bit	IN[003]	

Address	Address	Variable Name	Data	Description	Remarks
68	0x0044	QW65028,bit4	Bit	IN[004]	
69	0x0045	QW65028,bit5	Bit	IN[005]	
70	0x0046	QW65028,bit6	Bit	IN[006]	
71	0x0047	QW65028,bit7	Bit	IN[007]	
72	0x0048	QW65028,bit8	Bit	IN[008]	
73	0x0049	QW65028,bit9	Bit	IN[009]	
74	0x004A	QW65028,bit10	Bit	IN[010]	
75	0x004B	QW65028,bit11	Bit	IN[011]	
76	0x004C	QW65028,bit12	Bit	IN[012]	
77	0x004D	QW65028,bit13	Bit	IN[013]	
78	0x004E	QW65028,bit14	Bit	IN[014]	
79	0x004F	QW65028,bit15	Bit	IN[015]	
80	0x0050	QW65029,bit0	Bit	IN[016]	
81	0x0051	QW65029,bit1	Bit	IN[017]	
82	0x0052	QW65029,bit2	Bit	IN[018]	
83	0x0053	QW65029,bit3	Bit	IN[019]	
84	0x0054	QW65029,bit4	Bit	IN[020]	
85	0x0055	QW65029,bit5	Bit	IN[021]	
86	0x0056	QW65029,bit6	Bit	IN[022]	
87	0x0057	QW65029,bit7	Bit	IN[023]	
88	0x0058	QW65029,bit8	Bit	IN[024]	
89	0x0059	QW65029,bit9	Bit	IN[025]	
90	0x005A	QW65029,bit10	Bit	IN[026]	
91	0x005B	QW65029,bit11	Bit	IN[027]	
92	0x005C	QW65029,bit12	Bit	IN[028]	
93	0x005D	QW65029,bit13	Bit	IN[029]	
94	0x005E	QW65029,bit14	Bit	IN[030]	
95	0x005F	QW65029,bit15	Bit	IN[031]	
96	0x0060	QW65030,bit0	Bit	IN[032]	
97	0x0061	QW65030,bit1	Bit	IN[033]	

Address	Address	Variable Name	Data	Description	Remarks
98	0x0062	QW65030,bit2	Bit	IN[034]	
99	0x0063	QW65030,bit3	Bit	IN[035]	
100	0x0064	QW65030,bit4	Bit	IN[036]	
101	0x0065	QW65030,bit5	Bit	IN[037]	
102	0x0066	QW65030,bit6	Bit	IN[038]	
103	0x0067	QW65030,bit7	Bit	IN[039]	
104	0x0068	QW65030,bit8	Bit	IN[040]	
105	0x0069	QW65030,bit9	Bit	IN[041]	
106	0x006A	QW65030,bit10	Bit	IN[042]	
107	0x006B	QW65030,bit11	Bit	IN[043]	
108	0x006C	QW65030,bit12	Bit	IN[044]	
109	0x006D	QW65030,bit13	Bit	IN[045]	
110	0x006E	QW65030,bit14	Bit	IN[046]	
111	0x006F	QW65030,bit15	Bit	IN[047]	
112	0x0070	QW65031,bit0	Bit	IN[048]	
113	0x0071	QW65031,bit1	Bit	IN[049]	
114	0x0072	QW65031,bit2	Bit	IN[050]	
115	0x0073	QW65031,bit3	Bit	IN[051]	
116	0x0074	QW65031,bit4	Bit	IN[052]	
117	0x0075	QW65031,bit5	Bit	IN[053]	
118	0x0076	QW65031,bit6	Bit	IN[054]	
119	0x0077	QW65031,bit7	Bit	IN[055]	
120	0x0078	QW65031,bit8	Bit	IN[056]	
121	0x0079	QW65031,bit9	Bit	IN[057]	
122	0x007A	QW65031,bit10	Bit	IN[058]	
123	0x007B	QW65031,bit11	Bit	IN[059]	
124	0x007C	QW65031,bit12	Bit	IN[060]	
125	0x007D	QW65031,bit13	Bit	IN[061]	
126	0x007E	QW65031,bit14	Bit	IN[062]	
127	0x007F	QW65031,bit15	Bit	IN[063]	

Address	Address	Variable Name	Data	Description	Remarks
128	0x0080	QW65032,bit0	Bit	OUT[000] status (0-OFF, 1-ON, same below)	
129	0x0081	QW65032,bit1	Bit	OUT[001]	
130	0x0082	QW65032,bit2	Bit	OUT[002]	
131	0x0083	QW65032,bit3	Bit	OUT[003]	
132	0x0084	QW65032,bit4	Bit	OUT[004]	
133	0x0085	QW65032,bit5	Bit	OUT[005]	
134	0x0086	QW65032,bit6	Bit	OUT[006]	
135	0x0087	QW65032,bit7	Bit	OUT[007]	
136	0x0088	QW65032,bit8	Bit	OUT[008]	
137	0x0089	QW65032,bit9	Bit	OUT[009]	
138	0x008A	QW65032,bit10	Bit	OUT[010]	
139	0x008B	QW65032,bit11	Bit	OUT[011]	
140	0x008C	QW65032,bit12	Bit	OUT[012]	
141	0x008D	QW65032,bit13	Bit	OUT[013]	
142	0x008E	QW65032,bit14	Bit	OUT[014]	
143	0x008F	QW65032,bit15	Bit	OUT[015]	
144	0x0090	QW65033,bit0	Bit	OUT[016]	
145	0x0091	QW65033,bit1	Bit	OUT[017]	
146	0x0092	QW65033,bit2	Bit	OUT[018]	
147	0x0093	QW65033,bit3	Bit	OUT[019]	
148	0x0094	QW65033,bit4	Bit	OUT[020]	
149	0x0095	QW65033,bit5	Bit	OUT[021]	
150	0x0096	QW65033,bit6	Bit	OUT[022]	
151	0x0097	QW65033,bit7	Bit	OUT[023]	
152	0x0098	QW65033,bit8	Bit	OUT[024]	
153	0x0099	QW65033,bit9	Bit	OUT[025]	
154	0x009A	QW65033,bit10	Bit	OUT[026]	
155	0x009B	QW65033,bit11	Bit	OUT[027]	
156	0x009C	QW65033,bit12	Bit	OUT[028]	
157	0x009D	QW65033,bit13	Bit	OUT[029]	
158	0x009E	QW65033,bit1	Bit	OUT[030]	

Address	Address	Variable Name	Data	Description	Remarks
		4			
159	0x009F	QW65033,bit1 5	Bit	OUT[031]	
160	0x00A0	QW65034,bit0	Bit	OUT[032]	
161	0x00A1	QW65034,bit1	Bit	OUT[033]	
162	0x00A2	QW65034,bit2	Bit	OUT[034]	
163	0x00A3	QW65034,bit3	Bit	OUT[035]	
164	0x00A4	QW65034,bit4	Bit	OUT[036]	
165	0x00A5	QW65034,bit5	Bit	OUT[037]	
166	0x00A6	QW65034,bit6	Bit	OUT[038]	
167	0x00A7	QW65034,bit7	Bit	OUT[039]	
168	0x00A8	QW65034,bit8	Bit	OUT[040]	
169	0x00A9	QW65034,bit9	Bit	OUT[041]	
170	0x00AA	QW65034,bit1 0	Bit	OUT[042]	
171	0x00AB	QW65034,bit1 1	Bit	OUT[043]	
172	0x00AC	QW65034,bit1 2	Bit	OUT[044]	
173	0x00AD	QW65034,bit1 3	Bit	OUT[045]	
174	0x00AE	QW65034,bit1 4	Bit	OUT[046]	
175	0x00AF	QW65034,bit1 5	Bit	OUT[047]	
176	0x00B0	QW65035,bit0	Bit	OUT[048]	
177	0x00B1	QW65035,bit1	Bit	OUT[049]	
178	0x00B2	QW65035,bit2	Bit	OUT[050]	
179	0x00B3	QW65035,bit3	Bit	OUT[051]	
180	0x00B4	QW65035,bit4	Bit	OUT[052]	
181	0x00B5	QW65035,bit5	Bit	OUT[053]	
182	0x00B6	QW65035,bit6	Bit	OUT[054]	
183	0x00B7	QW65035,bit7	Bit	OUT[055]	
184	0x00B8	QW65035,bit8	Bit	OUT[056]	
185	0x00B9	QW65035,bit9	Bit	OUT[057]	
186	0x00BA	QW65035,bit1 0	Bit	OUT[058]	
187	0x00BB	QW65035,bit1 1	Bit	OUT[059]	
188	0x00BC	QW65035,bit1 2	Bit	OUT[060]	
189	0x00BD	QW65035,bit1	Bit	OUT[061]	

Address	Address	Variable Name	Data	Description	Remarks
		3			
190	0x00BE	QW65035,bit1 4	Bit	OUT[062]	
191	0x00BF	QW65035,bit1 5	Bit	OUT[063]	
192	0x00C0	QW65036,bit0	Bit	J1 servo fault (0-No Fault, 1-Fault, same below)	
193	0x00C1	QW65036,bit1	Bit	J2 servo fault	
194	0x00C2	QW65036,bit2	Bit	J3 servo fault	
195	0x00C3	QW65036,bit3	Bit	J4 servo fault	
196	0x00C4	QW65036,bit4	Bit	J5 servo fault	
197	0x00C5	QW65036,bit5	Bit	J6 servo fault	
198	0x00C6	QW65036,bit6	Bit	J7 servo fault	
199	0x00C7	QW65036,bit7	Bit	J8 servo fault	
200	0x00C8	QW65036,bit8	Bit	J1 servo warning (0-No warning, 1-Warning, same below)	
201	0x00C9	QW65036,bit9	Bit	J2 servo warning	
202	0x00CA	QW65036,bit1 0	Bit	J3 servo warning	
203	0x00CB	QW65036,bit1 1	Bit	J4 servo warning	
204	0x00CC	QW65036,bit1 2	Bit	J5 servo warning	
205	0x00CD	QW65036,bit1 3	Bit	J6 servo warning	
206	0x00CE	QW65036,bit1 4	Bit	J7 servo warning	
207	0x00CF	QW65036,bit1 5	Bit	J8 servo warning	
…	…	…	Bit	Reserved	
2048	0x0800	QW65152,bit0	Bit	User-defined	
…	…	…	Bit		
4095	0x0FFF	QW65279,bit1 5	Bit		

- **Read-write (4096) coil, parameter: 0x01, 0x05, 0x0f**

Address DEC	Address HEX	Variable Name	Data Type	Description	Remarks
4096	0x1000	QW65280,bit0	Bit	Program start	(Recurrent)
4097	0x1001	QW65280,bit1	Bit	Program stop	(Recurrent)

Address	Address	Variable Name	Data	Description	Remarks
4098	0x1002	QW65280,bit2	Bit	Program reset (main task and dynamic task return to start line)	(Recurrent)
4099	0x1003	QW65280,bit3	Bit	Enable switch (0-OFF, 1-ON)	(Special, 0->1 for ON, 1->0 for OFF)
4100	0x1004	QW65280,bit4	Bit	Emergency stop switch (0-OFF, 1-ON)	(Hold)
4101	0x1005	QW65280,bit5	Bit	Clears alarm	(Recurrent)
4102	0x1006	QW65280,bit6	Bit	Switches to teach mode	(Recurrent) When switched from the play mode to the teach mode, the program will automatically return to the start line.
4103	0x1007	QW65280,bit7	Bit	Switches to play mode	(Recurrent) When switched from the teach mode to the play mode, the program will automatically return to the start line.
…	…	……	Bit	Reserved	
4112	0x1010	QW65281,bit0	Bit	Teach J1/X+	(Special type, 0-> 1 for Start, 1-> 0 for Stop, same below)

Address	Address	Variable Name	Data	Description	Remarks
4113	0x1011	QW65281,bit1	Bit	Teach J2/Y+	(Special)
4114	0x1012	QW65281,bit2	Bit	Teach J3/Z+	(Special)
4115	0x1013	QW65281,bit3	Bit	Teach J4/ (Rz+ of SCARA robot, Rx+ of 6-axis robot)	(Special)
4116	0x1014	QW65281,bit4	Bit	Teach J5+/Six Joint Ry+	(Special)
4117	0x1015	QW65281,bit5	Bit	Teach J6+/Rz+ of 6-axis robot	(Special)
4118	0x1016	QW65281,bit6	Bit	Reserved	
4119	0x1017	QW65281,bit7	Bit	Reserved	
4120	0x1018	QW65281,bit8	Bit	Teach J1/X-	(Special)
4121	0x1019	QW65281,bit9	Bit	Teach J2/Y-	(Special)
4122	0x101A	QW65281,bit10	Bit	Teach J3/Z-	(Special)
4123	0x101B	QW65281,bit11	Bit	Teach J4/ (Rz- of SCARA robot, Rx- of 6-axis robot)	(Special)
4124	0x101C	QW65281,bit12	Bit	Teach J5-/Ry- of 6-axis robot	(Special)
4125	0x101D	QW65281,bit13	Bit	Teach J6-/Rz- of 6-axis robot	(Special)
4126	0x101E	QW65281,bit14	Bit	Reserved	
4127	0x101F	QW65281,bit15	Bit	Reserved	
4128	0x1020	QW65282,bit0	Bit	Writes the robot's current position to the current P variable	(Recurrent)
4129	0x1021	QW65282,bit1	Bit	Writes the modified position (MW34855, etc.) to the current P variable	(Recurrent)
4130	0x1022	QW65282,bit2	Bit	Moves directly to the current P variable position	(Special type, 0-> 1 for Start, 1-> 0 for Stop)
…	…	……	Bit	Reserved	
4144	0x1030	QW65283,bit0	Bit	OUT[000] control command (0-OFF, 1-ON, same below)	(Special, 0->1 for ON, 1->0 for OFF, same below)
4145	0x1031	QW65283,bit1	Bit	OUT[001]	
4146	0x1032	QW65283,bit2	Bit	OUT[002]	
4147	0x1033	QW65283,bit3	Bit	OUT[003]	
4148	0x1034	QW65283,bit4	Bit	OUT[004]	
4149	0x1035	QW65283,bit5	Bit	OUT[005]	
4150	0x1036	QW65283,bit6	Bit	OUT[006]	

Address	Address	Variable Name	Data	Description	Remarks
4151	0x1037	QW65283,bit7	Bit	OUT[007]	
4152	0x1038	QW65283,bit8	Bit	OUT[008]	
4153	0x1039	QW65283,bit9	Bit	OUT[009]	
4154	0x103A	QW65283,bit10	Bit	OUT[010]	
4155	0x103B	QW65283,bit11	Bit	OUT[011]	
4156	0x103C	QW65283,bit12	Bit	OUT[012]	
4157	0x103D	QW65283,bit13	Bit	OUT[013]	
4158	0x103E	QW65283,bit14	Bit	OUT[014]	
4159	0x103F	QW65283,bit15	Bit	OUT[015]	
4160	0x1040	QW65284,bit0	Bit	OUT[016]	
4161	0x1041	QW65284,bit1	Bit	OUT[017]	
4162	0x1042	QW65284,bit2	Bit	OUT[018]	
4163	0x1043	QW65284,bit3	Bit	OUT[019]	
4164	0x1044	QW65284,bit4	Bit	OUT[020]	
4165	0x1045	QW65284,bit5	Bit	OUT[021]	
4166	0x1046	QW65284,bit6	Bit	OUT[022]	
4167	0x1047	QW65284,bit7	Bit	OUT[023]	
4168	0x1048	QW65284,bit8	Bit	OUT[024]	
4169	0x1049	QW65284,bit9	Bit	OUT[025]	
4170	0x104A	QW65284,bit10	Bit	OUT[026]	
4171	0x104B	QW65284,bit11	Bit	OUT[027]	
4172	0x104C	QW65284,bit12	Bit	OUT[028]	
4173	0x104D	QW65284,bit13	Bit	OUT[029]	
4174	0x104E	QW65284,bit14	Bit	OUT[030]	
4175	0x104F	QW65284,bit15	Bit	OUT[031]	
4176	0x1050	QW65285,bit0	Bit	OUT[032]	
4177	0x1051	QW65285,bit1	Bit	OUT[033]	
4178	0x1052	QW65285,bit2	Bit	OUT[034]	
4179	0x1053	QW65285,bit3	Bit	OUT[035]	
4180	0x1054	QW65285,bit4	Bit	OUT[036]	
4181	0x1055	QW65285,bit5	Bit	OUT[037]	
4182	0x1056	QW65285,bit6	Bit	OUT[038]	
4183	0x1057	QW65285,bit7	Bit	OUT[039]	
4184	0x1058	QW65285,bit8	Bit	OUT[040]	
4185	0x1059	QW65285,bit9	Bit	OUT[041]	
4186	0x105A	QW65285,bit10	Bit	OUT[042]	
4187	0x105B	QW65285,bit11	Bit	OUT[043]	
4188	0x105C	QW65285,bit12	Bit	OUT[044]	
4189	0x105D	QW65285,bit13	Bit	OUT[045]	
4190	0x105E	QW65285,bit14	Bit	OUT[046]	
4191	0x105F	QW65285,bit15	Bit	OUT[047]	
4192	0x1060	QW65286,bit0	Bit	OUT[048]	

Address	Address	Variable Name	Data	Description	Remarks
4193	0x1061	QW65286,bit1	Bit	OUT[049]	
4194	0x1062	QW65286,bit2	Bit	OUT[050]	
4195	0x1063	QW65286,bit3	Bit	OUT[051]	
4196	0x1064	QW65286,bit4	Bit	OUT[052]	
4197	0x1065	QW65286,bit5	Bit	OUT[053]	
4198	0x1066	QW65286,bit6	Bit	OUT[054]	
4199	0x1067	QW65286,bit7	Bit	OUT[055]	
4200	0x1068	QW65286,bit8	Bit	OUT[056]	
4201	0x1069	QW65286,bit9	Bit	OUT[057]	
4202	0x106A	QW65286,bit10	Bit	OUT[058]	
4203	0x106B	QW65286,bit11	Bit	OUT[059]	
4204	0x106C	QW65286,bit12	Bit	OUT[060]	
4205	0x106D	QW65286,bit13	Bit	OUT[061]	
4206	0x106E	QW65286,bit14	Bit	OUT[062]	
4207	0x106F	QW65286,bit15	Bit	OUT[063]	
…	…	…	Bit	Reserved	
6144	0x1800	QW65408,bit0	Bit	User-defined	
…	…	……	Bit		
8191	0x1FFF	QW65535,bit15	Bit		

■ **Read-only (32768) input register, parameter:0x04**

Address	Address	Variable Name	Data Type	Description	Remarks
DEC	HEX				
0	0x0	MW0	Word	Reserved for other use of the system (2048 words)	
…		……	Word		
2047	0x07FF	MW2047	Word		
2048	0x0800	MW2048	Word	Current coordinate system	
2049	0x0801	MW2049	Word	Current speed	
2050	0x0802	MW2050	Word	Fault record (hex display)	
2051	0x0803	MW2051	Word	Current mode (1-Teach, 2-Play)	
2052	0x0804	MW2052	Single precision floating point	J1/X coordinate low byte	Refers to the robot's coordinate values in the current coordinate system (The tool
2053	0x0805	MW2053		J2/X coordinate high byte	
2054	0x0806	MW2054	Single precision floating point	J2/Y coordinate low byte	
2055	0x0807	MW2055		J2/Y coordinate high byte	
2056	0x0808	MW2056	Single precision	J3/Z coordinate low byte	
2057	0x0809	MW2057		J3/Z coordinate high byte	

			floating point		coordinate system/user coordinate system must be meaningful to correctly display the values in the tool coordinate system/user coordinate system.)
2058	0x080A	MW2058	Single precision floating point	J4/A coordinate low byte	
2059	0x080B	MW2059		J4/A coordinate high byte	
2060	0x080C	MW2060	Single precision floating point	J5/B coordinate low byte	
2061	0x080D	MW2061		J5/B coordinate high byte	
2062	0x080E	MW2062	Single precision floating point	J6/C coordinate low byte	
2063	0x080F	MW2063		J6/C coordinate high byte	
2064	0x0810	MW2064	Word (unsigned)	Current tool coordinate system number (Only meaningful under tool coordinate system)	
2065	0x0811	MW2065	Word (unsigned)	Current user coordinate system number (Only meaningful under user coordinate system)	
…	…	……	Word	Reserved	
2081	0x0821	MW2081	Word (unsigned)	Current direct motion mode (0-MovJ, 1-MovL, 2-Jump, 3-JumpL)	
2082	0x0822	MW2082	Single precision floating point	Low byte of LH parameter for current jump motion	
2083	0x0823	MW2083		High byte of LH parameter for current	

				jump motion	
2084	0x0824	MW2084	Single precision floating point	Low byte of MH parameter for current jump motion	
2085	0x0825	MW2085		High byte of MH parameter for current jump motion	
2086	0x0826	MW2086	Single precision floating point	Low byte of RH parameter for current jump motion	
2087	0x0827	MW2087		Current jump motion RH parameter high	
…	…	……	Word	Reserved	
2116	0x0844	MW2116	Word (unsigned)	J1 servo alarm code	
2117	0x0845	MW2117	Word (unsigned)	J2 servo alarm code	
2118	0x0846	MW2118	Word (unsigned)	J3 servo alarm code	
2119	0x0847	MW2119	Word (unsigned)	J4 servo alarm code	
2120	0x0848	MW2120	Word (unsigned)	J5 servo alarm code	
2121	0x0849	MW2121	Word (unsigned)	J6 servo alarm code	
2122	0x084A	MW2122	Word (unsigned)	J7 servo alarm code	
2123	0x084B	MW2123	Word (unsigned)	J8 servo alarm code	
…	…	……	Word	Reserved	
2146	0x0862	MW2146	Word (unsigned)	Queries the device to which the control permission belongs (0-InoTeachPad, 1-InoRobShop, 2-Remote Ethernet, 3-Remote I/O, 4-Remote Modbus)	
2147	0x0863	MW2147	Word (unsigned)	Reserved	
4168	0x1048	MW 4168	Single precision floating point	Low byte of J1/X coordinate read by P variable	
4169	0x1049	MW 4169		High byte of J1/X coordinate read by P	

				variable	
4170	0x104A	MW 4170	Single precision floating point	Low byte of J2/Y coordinate read by P variable	
4171	0x104B	MW 4171		High byte of J2/Y coordinate read by P variable	
4172	0x104C	MW 4172	Single precision floating point	Low byte of J3/Z coordinate read by P variable	
4173	0x104D	MW 4173		High byte of J3/Z coordinate read by P variable	
4174	0x104E	MW 4174	Single precision floating point	Low byte of J4/A coordinate read by P variable	
4175	0x104F	NW4175		High byte of J4/A coordinate read by P variable	
4176	0x1050	MW 4176	Single precision floating point	Low byte of J5/B coordinate read by P variable	
4177	0x1051	MW 4177		High byte of J5/B coordinate read by P variable	
4178	0x1052	MW 4178	Single precision floating point	Low byte of J6/C coordinate read by P variable	
4179	0x1053	MW 4179		High byte of J6/C coordinate read by P variable	
4180	0x1054	NW4180	Word (signed)	Arm parameter 1 read by P variable	
4181	0x1055	NW4181	Word (signed)	Arm parameter 2 read by P variable	
4182	0x1056	MW 4182	Word (signed)	Arm parameter 3 read by P variable	
4183	0x1057	MW 4183	Word (signed)	Arm parameter 4 read by P variable	
4184	0x1058	MW 4184	Word (unsigned)	Coordinate system read by P variable	
4185	0x1059	MW 4185	Word (unsigned)	Tool coordinate system number read by P variable	

4186	0x105A	MW 4186	Word (unsigned)	User coordinate system number read by P variable	
4187	0x105B	NW4187	Word (unsigned)	Reserved	
4188	0x105C	MW 4188	Single precision floating point	Low byte of J1/X coordinate read by P[i+0] variable	Data is invalid when MW34853 is 0, i is the value of MW34852, same below
4189	0x105D	MW 4189		High byte of J1/X coordinate read by P[i+0] variable	
4190	0x105E	MW 4190	Single precision floating point	Low byte of J2/Y coordinate read by P[i+0] variable	
4191	0x105F	MW 4191		High byte of J2/Y coordinate read by P[i+0] variable	
4192	0x1060	MW 4192	Single precision floating point	Low byte of J3/Z coordinate read by P[i+0] variable	
4193	0x1061	MW 4193		High byte of J3/Z coordinate read by P[i+0] variable	
4194	0x1062	MW 4194	Single precision floating point	Low byte of J4/A coordinate read by P[i+0] variable	
4195	0x1063	MW 4195		High byte of J4/A coordinate read by P[i+0] variable	
4196	0x1064	MW 4196	Single precision floating point	Low byte of J5/B coordinate read by P[i+0] variable	
4197	0x1065	MW 4197		High byte of J5/B coordinate read by P[i+0] variable	
4198	0x1066	MW 4198	Single precision floating point	Low byte of J6/C coordinate read by P[i+0] variable	
4199	0x1067	MW 4199		High byte of J6/C coordinate read by P[i+0]	

				variable	
4200	0x1068	MW 4200	Word (signed)	Arm parameter 1 read by P[i+0] variable	
4201	0x1069	MW 4201	Word (signed)	Arm parameter 2 read by P[i+0] variable	
4202	0x106A	MW 4202	Word (signed)	Arm parameter 3 read by P[i+0] variable	
4203	0x106B	MW 4203	Word (signed)	Arm parameter 4 read by P[i+0] variable	
4204	0x106C	MW 4204	Word (unsigned)	Coordinate system read by P[i+0] variable	
4205	0x106D	MW 4205	Word (unsigned)	Tool coordinate system number read by P[i+0] variable	
4206	0x106E	MW 4206	Word (unsigned)	User coordinate system number read by P[i+0] variable	
…	…	…		…	
[4168+20*n]	…	MW(2168+20*n)	Single precision floating point	Low byte of J1/X coordinate read by P[i+n-1] variable	N is the value of MW34853 with a valid range of 1 to 100, i is the value of MW34852, same below
[4169+20*n]	…	MW(2169+20*n)		High byte of J1/X coordinate read by P[i+n-1] variable	
[4170+20*n]	…	MW(2170+20*n)	Single precision floating point	Low byte of J2/Y coordinate read by P[i+n-1] variable	
[4171+20*n]	…	MW(2171+20*n)		High byte of J2/Y coordinate read by P[i+n-1] variable	
[4172+20*n]	…	MW(2172+20*n)	Single precision floating point	Low byte of J3/Z coordinate read by P[i+n-1] variable	
[4173+20*n]	…	MW(2173+20*n)		High byte of J3/Z coordinate read by P[i+n-1] variable	
[4174+20*n]	…	MW(2174+20*n)	Single precision floating point	Low byte of J4/A coordinate read by P[i+n-1] variable	
[4175+20*n]	…	MW(2175+20*n)		High byte of J4/A coordinate read by	

				P[i+n-1] variable	
[4176+20*n]	…	MW(2176+20*n)	Single precision floating point	Low byte of J5/B coordinate read by P[i+n-1] variable	
[4177+20*n]	…	MW(2177+20*n)		High byte of J5/B coordinate read by P[i+n-1] variable	
[4178+20*n]	…	MW(2178+20*n)	Single precision floating point	Low byte of J6/C coordinate read by P[i+n-1] variable	
[4179+20*n]	…	MW(2179+20*n)		High byte of J6/C coordinate read by P[i+n-1] variable	
[4180+20*n]	…	MW(2180+20*n)	Word (signed)	Arm parameter 1 read by P[i+n-1] variable	
[4181+20*n]	…	MW(2181+20*n)	Word (signed)	Arm parameter 2 read by P[i+n-1] variable	
[4182+20*n]	…	MW(2182+20*n)	Word (signed)	Arm parameter 3 read by P[i+n-1] variable	
[4183+20*n]	…	MW(2183+20*n)	Word (signed)	Arm parameter 4 read by P[i+n-1] variable	
[4184+20*n]	…	MW(2184+20*n)	Word (unsigned)	Coordinate system number read by P[i+n-1] variable	
[4185+20*n]	…	MW(2185+20*n)	Word (unsigned)	Tool coordinate system number read by P[i+n-1] variable	
[4186+20*n]	…	MW(2186+20*n)	Word (unsigned)	User coordinate system number read by P[i+n-1] variable	
…	…	…	Word	Reserved	
8192	0x2000	MW8192	Word (unsigned)	B0 variable read value	
8193	0x2001	MW8193	Word (unsigned)	B1 variable read value	
8194	0x2002	MW8194	Word (unsigned)	B2 variable read value	
8195	0x2003	MW8195	Word (unsigned)	B3 variable read value	
8196	0x2004	MW8196	Word (unsigned)	B4 variable read value	
…	…	…	Word (unsigned)	…	

[m]	…	…	Word (unsigned)	B[n] variable read value (m=8192+n)	n is the B variable index.
…	…	…	Word (unsigned)	…	
8442	0x20FA	MW8442	Word (unsigned)	B250 variable read value	
8443	0x20FB	MW8443	Word (unsigned)	B251 variable read value	
8444	0x20FC	MW8444	Word (unsigned)	B252 variable read value	
8445	0x20FD	MW8445	Word (unsigned)	B253 variable read value	
8446	0x20FE	MW8446	Word (unsigned)	B254 variable read value	
8447	0x20FF	MW8447	Word (unsigned)	B255 variable read value	
8448	0x2100	MW8448	Double word (signed)	Low byte of value read by R0 variable	
8449	0x2101	MW8449		High byte of value read by R0 variable	
8450	0x2102	MW8450	Double word (signed)	Low byte of value read by R1 variable	
8451	0x2103	MW8451		High byte of value read by R1 variable	
8452	0x2104	MW8452	Double word (signed)	Low byte of value read by R2 variable	
8453	0x2105	MW8453		High byte of value read by R2 variable	
…	…	…		…	
[m]			Double Word (signed)	Low byte of value read by R[n] variable (m=8448+n*2)	n is the R variable index.
[m+1]	…	…		High byte of value read by R[n] variable (m=8448+n*2)	n is the R variable index.
…	…	…		…	
8954	0x22FA	MW8954	Double word (signed)	Low byte of value read by R253 variable	
8955	0x22FB	MW8955		High byte of value read by R253 variable	
8956	0x22FC	MW8956	Double	Low byte of value read	

			word (signed)	by R254 variable	
8957	0x22FD	MW8957		High byte of value read by R254 variable	
8958	0x22FE	MW8958	Double word (signed)	Low byte of value read by R255 variable	
8959	0x22FF	MW8959		High byte of value read by R255 variable	
8960	0x2300	MW8960	Double-precision floating point	Lowest byte of value read by D0 variable	
8961	0x2301	MW8961		Low byte of value read by D0 variable	
8962	0x2302	MW8962		High byte of value read by D0 variable	
8963	0x2303	MW8963		Highest byte of value read by D0 variable	
8964	0x2304	MW8964	Double-precision floating point	Lowest byte of value read by D1 variable	
8965	0x2305	MW8965		Low byte of value read by D1 variable	
8966	0x2306	MW8966		High byte of value read by D1 variable	
8967	0x2307	MW8967		Highest byte of value read by D1 variable	
8968	0x2308	MW8968	Double-precision floating point	Lowest byte of value read by D2 variable	
8969	0x2309	MW8969		Low byte of value read by D2 variable	
8970	0x230A	MW8970		High byte of value read by D2 variable	
8971	0x230B	MW8971		Highest byte of value read by D2 variable	
…	…	…	…	…	
[m]	…	MW[m]	Double-precision floating point	Lowest byte of value read by D[n] variable (m=89604+n*4)	n is the D variable index.
[m+1]	…	MW[m+1]		Low byte of value read by D[n] variable	
[m+2]	…	MW[m+2]		High byte of value read by D[n] variable	
[m+3]	…	MW[m+3]		Highest byte of value read by D[n] variable	
…	…	…	…	…	
9972	0x26F4	MW9972	Double-prec	Lowest byte of value read	

			ision floating point	by D253 variable	
9973	0x26F5	MW9973		Low byte of value read by D253 variable	
9974	0x26F6	MW9974		High byte of value read by D253 variable	
9975	0x26F7	MW9975		Highest byte of value read by D253 variable	
9976	0x26F8	MW9976	Double-precision floating point	Lowest byte of value read by D254 variable	
9977	0x26F9	MW9977		Low byte of value read by D254 variable	
9978	0x26FA	MW9978		High byte of value read by D254 variable	
9979	0x26FB	MW9979		Highest byte of value read by D254 variable	
9980	0x26FC	MW9980	Double-precision floating point	Lowest byte of value read by D255 variable	
9981	0x26FD	MW9981		Low byte of value read by D255 variable	
9982	0x26FE	MW9982		High byte of value read by D255 variable	
9983	0x26FF	MW9983		Highest byte of value read by D255 variable	
…	…	…	…	Reserved	
16384	0x4000	……	Word	User-defined	
…		……	Word		
32767	0x7fff	MW32767	Word		

- **Read-write (32768) holding registers, parameter: 0x03, 0x06, 0x10**

Address DEC	Address HEX	Variable Name	Data Type	Description	Remarks
32768	0x8000	MW32768	Word	Reserved for other use of the system	
32769	0x8001	MW32769	Word		
…	…	……	Word		
34800	0x87F0	MW34800	Word	Selects the teaching mode (0-Continuous teaching, 1-Jog teaching)	(Level type)
34801	0x87F1	MW348	Single	Low byte of joint step size for	(Level type)

Address DEC	Address HEX	Variable Name	Data Type	Description	Remarks
		01	precision floating point	jog motion (in degree, valid in joint coordinate system)	
34802	0x87F2	MW34802		High byte of joint step size for jog motion	
34803	0x87F3	MW34803	Single precision floating point	Low byte of linear step size for jog motion (in mm, valid in Cartesian coordinate system)	(Level type)
34804	0x87F4	MW34804		High byte of linear step size for jog motion	
34805	0x87F5	MW34805	Word (unsigned)	Sets the direct motion mode (0-MovJ, 1-MovL, 2-Jump, 3-JumpL)	(Level type)
34806	0x87F6	MW34806	Single precision floating point	Low byte of LH parameter for jump motion	(Level type)
34807	0x87F7	MW34807		High byte of LH parameter for jump motion	
34808	0x87F8	MW34808	Single precision floating point	Low byte of MH parameter for jump motion	(Level type)
34809	0x87F9	MW34809		High byte of MH parameter for jump motion	
34810	0x87FA	MW34810	Single precision floating point	Low byte of RH parameter for jump motion	(Level type)
34811	0x87FB	MW34811		High byte of RH parameter for jump motion	
…	…	…	Word	Reserved	
34815	0x87FF	MW34815	Word	Heartbeat interval (unit: ms)	The default value is 500ms and the range is [200-65000]ms.
34816	0x8800	MW34816	Word	Selects the coordinate system (1-Joint, 2-Cartesian, 3-Tool, 4-User)	(Level type)
34817	0x8801	MW34817	Word	Speed setting (1-100)	(Level type)
34818	0x8802	MW34818	Word	Selects the tool number (0-15)	(Level type)
34819	0x8803	MW34819	Word	Selects the user coordinate system number (0-15)	(Level type)
…	…	…	Word	Reserved	
34852	0x8824	MW34852	Word (unsigned)	Starting subscript i (0-9999) of batch read P variables	Range 0-9999 (level type)

Address DEC	Address HEX	Variable Name	Data Type	Description	Remarks
34853	0x8825	MW34853	Word (unsigned)	Number of batch read P variables, n (P[0] - P[n-1], n has a valid range of 1 to 100, not read in batch for other values)	N+i should not exceed 10000 (level type)[1]
34854	0x8826	MW34854	Word (unsigned)	Serial number of the current P variable (waiting for read, write, or motion operation)	(Level type)
34855	0x8827	MW34855	Single precision floating point	Low byte of J1/X coordinate written by P variable	(Level type, same below)
34856	0x8828	MW34856		High byte of J1/X coordinate written by P variable	
34857	0x8829	MW34857	Single precision floating point	Low byte of J2/Y coordinate written by P variable	
34858	0x882A	MW34858		High byte of J2/Y coordinate written by P variable	
34859	0x882B	MW34859	Single precision floating point	Low byte of J3/Z coordinate written by P variable	
34860	0x882C	MW34860	Bit	High byte of J3/Z coordinate written by P variable	
34861	0x882D	MW34861	Single precision floating point	Low byte of J4/A coordinate written by P variable	
34862	0x882E	MW34862		High byte of J4/A coordinate written by P variable	
34863	0x882F	MW34863	Single precision floating point	Low byte of J5/B coordinate written by P variable	
34864	0x8830	MW34864		High byte of J5/B coordinate written by P variable	
34865	0x8831	MW34865	Single precision floating point	Low byte of J6/C coordinate written by P variable	
34866	0x8832	MW34866		High byte of J6/C coordinate written by P variable	
34867	0x8833	MW34867	Word (signed)	Arm parameter 1 written by P variable	Refer to Section 1.5.3 for the range of arm parameters.
34868	0x8834	MW34868	Word (signed)	Arm parameter 2 written by P variable	

[1] When n+i exceeds 10000 it cannot be read, and the value of addresses such as MW4188 will change to 0

Address DEC	Address HEX	Variable Name	Data Type	Description	Remarks
34869	0x8835	MW34869	Word (signed)	Arm parameter 3 written by P variable	
34870	0x8836	MW34870	Word (signed)	Arm parameter 4 written by P variable	
34871	0x8837	MW34871	Word (unsigned)	Coordinate system written by P variable	Range [1,7]
34872	0x8838	MW34872	Word (unsigned)	Too coordinate system number written by P variable	Range [0,15]
34873	0x8839	MW34873	Word (unsigned)	User coordinate system number written by P variable	Range [0,15]
34874	0x883A	MW34874	Word	Reserved	Occupied by the written value of the P variable
…	…	…	Word	-	
36864	0x9000	MW36864	Word (unsigned)	B0 variable write value	(Level type, same below)
36865	0x9001	MW36865	Word (unsigned)	B1 variable write value	
36866	0x9002	MW36866	Word (unsigned)	B2 variable write value	
36867	0x9003	MW36867	Word (unsigned)	B3 variable write value	
36868	0x9004	MW36868	Word (unsigned)	B4 variable write value	
…	…	…	Word (unsigned)	…	
[m]	…	…	Word (unsigned)	B[n] variable write value (m=36864+n)	n is the B variable index.
…	…	…	Word (unsigned)	…	
37114	0x90FA	MW371	Word	B250 variable write value	

Address DEC	Address HEX	Variable Name	Data Type	Description	Remarks
		14	(unsigned)		
37115	0x90FB	MW37115	Word (unsigned)	B251 variable write value	
37116	0x90FC	MW37116	Word (unsigned)	B252 variable write value	
37117	0x90FD	MW37117	Word (unsigned)	B253 variable write value	
37118	0x90FE	MW37118	Word (unsigned)	B254 variable write value	
37119	0x90FF	MW37119	Word (unsigned)	B255 variable write value	
37120	0x9100	MW37120	Double word (signed)	Low byte of value written by R0 variable	
37121	0x9101	MW37121		High byte of value written by R0 variable	
37122	0x9102	MW37122	Double word (signed)	Low byte of value written by R0 variable	
37123	0x9103	MW37123		High byte of value written by R0 variable	
…	…	…		…	
[m]			Double Word (signed)	Low byte of value written by R[n] variable (m=37120+2*n)	
[m+1]	…	…		High byte of value written by R[n] variable (m=37120+2*n)	n is the R variable index.
…	…	…		…	
37628	0x92FC	MW37628	Double word (signed)	Low byte of value written by R254 variable	
37629	0x92FD	MW37629		High byte of value written by R254 variable	
37630	0x92FE	MW37630	Double word (signed)	Low byte of value written by R255 variable	
37631	0x92FF	MW37631		High byte of value written by R255 variable	
37632	0x9300	MW37632	Double-precision floating	Lowest byte of value written by D0 variable	
37633	0x9301	MW376		Low byte of value written by D0	

Address DEC	Address HEX	Variable Name	Data Type	Description	Remarks
		33	point	variable	
37634	0x9302	MW37634		High byte of value written by D0 variable	
37635	0x9303	MW37635		Highest byte of value written by D0 variable	
37636	0x9304	MW37636	Double-precision floating point	Lowest byte of value written by D1 variable	
37637	0x9305	MW37637		Low byte of value written by D1 variable	
37638	0x9306	MW37638		High byte of value written by D1 variable	
37639	0x9307	MW37639		Highest byte of value written by D1 variable	
…	…	…	…	…	
[m]	…	MW[m]	Double-precision floating point	Lowest byte of value written by D[n] variable (m=37632+n*4)	n is the D variable index.
[m+1]	…	MW[m+1]		Low byte of value written by D[n] variable	
[m+2]		MW[m+2]		High byte of value written by D[n] variable	
[m+3]		MW[m+3]		Highest byte of value written by D[n] variable	
…	…	…	…	…	
38644	0x96F4	MW38644	Double-precision floating point	Lowest byte of value written by D253 variable	
38645	0x96F5	MW38645		High byte of value written by D253 variable	
38646	0x96F6	MW38646		Low byte of value written by D253 variable	
38647	0x96F7	MW38647		Highest byte of value written by D253 variable	
38648	0x96F8	MW38648	Double-precision floating point	Lowest byte of value written by D254 variable	
38649	0x96F9	MW38649		High byte of value written by D254 variable	
38650	0x96FA	MW38650		Low byte of value written by D254 variable	
38651	0x96FB	MW38651		Highest byte of value written by D254 variable	
38652	0x96FC	MW38652	Double-precision	Lowest byte of value written by D255 variable	

Address DEC	Address HEX	Variable Name	Data Type	Description	Remarks
38653	0x96FD	MW38653	floating point	High byte of value written by D255 variable	
38654	0x96FE	MW38654		Low byte of value written by D255 variable	
38655	0x96FF	MW38655		Highest byte of value written by D255 variable	
…		…	Word	Reserved	
49152		MW49152	Word	User-defined	
…		…	Word		
65535	0xFFFF	MW65535	Word		

Appendix 4: Servo Commissioning

1) Description

1.1) Communication links and ports:

Standard Ethernet link, Ethernet port on the controller.

1.2) Module functions:

Servo data supports all functions, DSP data supports only functions 1-4.

No.	Function	Description
1	Import and export of waveform files	Supports the export of waveform data and simultaneous export of multi-channel waveform data. Supports import of data file with suffix .inoparam and display of waveforms.
2	Continuous oscilloscope	Supports the continuous data acquisition function of the oscilloscope as well as the waveform display function.
3	Waveform analyzer	Supports functions such as waveform selection, dragging, scaling, adaptation, horizontal and vertical scale adjustment, FFT analysis, waveform comparison, coordinate value measurement, and cursor value measurement.
4	Trigger Oscilloscope	Supports the setting of channel data collection conditions, trigger conditions: rising edge/falling edge/edge change/above/below the level, trigger level setting, pre-trigger setting (%), number of conditions: 2, condition relationship: single

		condition, two conditions (and, or)
5	Parameters and operating status monitoring	Supports the monitoring of commonly used servo parameters and operating status of robots (error, ready, run, no ready, EtherCAT state machine)
6	Servo commissioning	Supports servo usability adjustment, tuning, fault management, I/O settings, servo parameter list, speed JOG, position JOG, homing, bus motor parameters, data monitoring and other functions.
7	Running status monitoring	Displays running status on the interface: servo alarm error, servo drive status (ready, run, no ready), servo EtherCAT state machine (1248)
8	Log	Log file in xxx format.
9	Historical fault records	Reads and displays the last 10 fault records for each axis

1.3) Description of the module function

1.4.1) Continuous oscilloscope

The oscilloscope includes servo oscilloscope and DSP oscilloscope.

1.4.1.1) Servo continuous oscilloscope

(Specifications in this Section refer to SV660N Servo Drive Software Specification V1.2.docx)

The range of data that a servo continuous oscilloscope can display is as follows.

No.	Data Name	Source
1	Bit monitoring channel (signals such as servo I/O)	Servo
2	Position reference	Servo
3	Position feedback	Servo
4	Position following error (position deviation)	Servo
5	Speed reference	Servo
6	Speed feedback	Servo
7	Torque reference	Servo
8	Current feedback	Servo
9	Bus voltage	Servo
10	U-phase feedback current	Servo
11	V-phase feedback current	Servo
12	W-phase feedback current	Servo
13	d-axis current feedback	Servo
14	Control word (received by servo)	Servo
15	Status word (sent from servo)	Servo
16	Average load rate	Servo
17	Input reference pulse counter	Servo
18	Current absolute position	Servo
19	Resonance auto-tuning results	Servo
20	Inertia auto-tuning results	Servo

21	Tracking deviation of position reference unit	Servo
22	Real-time target absolute position	Servo
23	Real-time target absolute speed	Servo
24	Real-time target absolute torque	Servo

The indicators of the servo continuous oscilloscope are as follows.

No.	Indicator	Description
1	Sampling interval	1ms to 100ms
2	Maximum number of supported channels	8
3	Transfer data type	16-Bit, 32-bit
4	Channel configuration	Each channel can be configured with any sampling data for any axis (repeatable sampling data configuration options for multiple channels) Supports channel settings: The visibility, scale, color, vertical scale, etc. for each channel can be set. The longitudinal movement of waveform is also supported.
5	Waveform acquisition start and stop	Start and stop buttons are provided. When the stop button is pressed, the waveform acquisition stops and is displayed automatically.
6	Waveform display time range	The horizontal timeline can be set, with a range of 300ms to 30000ms.

1.4.1.2) DSP Continuous oscilloscope

The range of data that a DSP continuous oscilloscope can display is as follows.

No.	Data Name	Source
1	Running line number (float)	DSP
2	Running status	DSP
3	Common DI	DSP
4	Common Do	DSP
5	Torque feedback (per axis)	DSP
6	Planed angle (per axis)	DSP
7	Feedback angle (per axis)	DSP
8	Control word (planned by DSP) (per axis)	DSP
9	Status word (received by DSP) (per axis)	DSP
10	Internal planning status	DSP
11	Stop mode (per axis)	DSP
12	System DI	DSP

13	System DO	DSP
14	Emergency stop mode	DSP
15	Position in Cartesian space (6)	DSP
16	Speed in Cartesian space (6)	DSP
17	DSP alarm code	DSP
18	Servo gain (*3 per axis)	DSP

The indicators of the DSP continuous oscilloscope are as follows.

No.	Indicator	Description
1	Sampling interval	1ms to 100ms
2	Maximum number of supported channels	8
3	Transfer data type	16-Bit, 32-bit

1.4.2) Import and export of waveform files

Supports the export of waveform data and simultaneous export of multi-channel waveform data.

Supports import of data file with suffix ".inoparam" and display of waveforms.

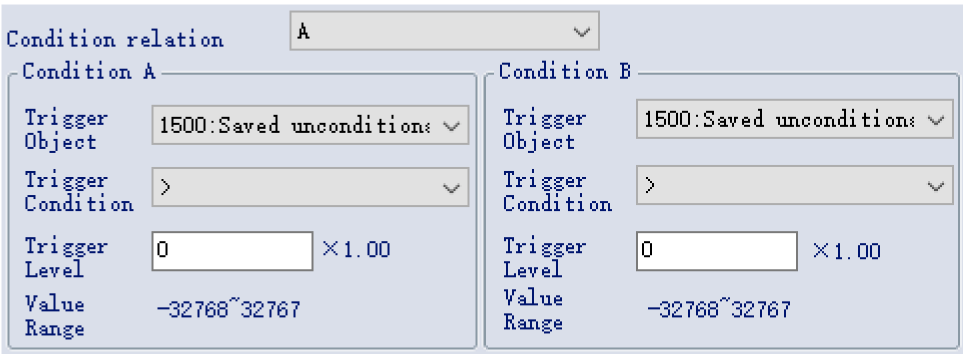
1.4.3) Waveform analyzer

Supports functions such as waveform selection, dragging, scaling, adaptation, horizontal and vertical scale adjustment, FFT analysis, waveform comparison, coordinate value measurement, and cursor value measurement.

1.4.4) Trigger oscilloscope

(Specifications in this Section refer to SV660N Servo Drive Software Specification V1.2.docx)

The trigger setting supports single or two trigger conditions, as shown below.



The trigger condition can be set to condition A, condition B, both conditions A and B, or condition A or B.

The trigger object can be set to channel variables, DI bit variables, DO bit variables.

The trigger conditions include: rising edge/falling edge/edge change/above level/below level.

Trigger level setting

Pre-trigger setting (%)

1.4.4) Parameter monitoring

The parameter monitoring function is divided into servo parameter and DSP parameter monitoring, and their specifications are as follows.

1.4.4.1) Servo parameter monitoring

It supports the monitoring of servo parameter data commonly used by robots, the data contents are as follows.

No.	Description	Parameter	Dictionary Object	Unit	Range	Length
1	Servo software version (J1 axis to J6 axis)	H0100	2001h-01h	1	0 to 65535	Uint16
2	Servo software non-standard version (J1 axis to J6 axis)	H0002	2000h-03h	1	0 to 65535	Uint32
3	Encoder software version (J1 axis to J6 axis)	H0004	2000h-05h	1	0 to 65535	Uint16
4	Motor model (J1 axis to J6 axis)	H0000	2000h-01h	1	0 to 65535	Uint16
5	Servo model (J1 axis to J6 axis)	H0102	2001h-03h	1	0 to 65535	Uint16
6	Absolute position mode (J1 axis to J6 axis)	H0201	2002-02h	1	0 to 2	Uint16
7	Actual motor speed (J1 axis to J6 axis)	H0B00	200Bh-01h	rpm	-9000 to +9000	int16
8	Average load rate (J1 axis to J6 axis)	H0B12	200Bh-0Dh	0.1%	-3000 to +3000	Uint16
9	Absolute position feedback (J1 axis to J6 axis)	H0B17	200Bh-12h	Encoder unit	-2^{31} to $+2^{31}$	int32
10	Encoder multi-turn data (J1 axis to J6 axis)	H0B70	200Bh-47h	Turn	-32767 to +32767	Uint16
11	Encoder single-turn data (J1 axis to J6 axis)	H0B71	200Bh-48h	Encoder unit	-2^{31} to $+2^{31}$	int32
12	Servo fault code (J1 axis to J6 axis)	H0B34	200Bh-23h	1	0 to 65535	Uint16
13	Servo Sub-fault code (J1 axis to J6 axis)	H0B45	200Bh-2Eh	1	0 to 65535	Uint16
14	Encoder sub-fault	H0B28	200Bh-1D	1	0 to 65535	Uint16

	code (J1 axis to J6 axis)		h			
15	Torque feedback (J1 axis to J6 axis)		6077h	0.1%	-5000 to +5000	int16
16	Target position (J1 axis to J6 axis)		607Ah	Reference unit	-2^{31} to $+2^{31}$	int32
17	Position feedback (J1 axis to J6 axis)		6064h	Encoder unit	-2^{31} to $+2^{31}$	int32
18	Gear ratio (J1 axis to J6 axis)		6091-01h 6091-02h	1	0 to 2^{32}-1	Uint32 Uint32
19	Position deviation (J1 axis to J6 axis)		60F4	Reference unit	-2^{31} to $+2^{31}$	int32
20	Excessive position deviation threshold (J1 axis to J6 axis)		6065	Reference unit	0 to 2^{32}-1	Uint32

1.4.4.2) DSP parameter Monitoring

It supports monitoring of commonly used DSP parameter data, the data contents are shown in the table below.

No.	Description	Parameter	Unit	Min. Value	Max. Value	Data Type	Change Method
1	Running line number	R0010	1	1		short	Read-only
2	Running status	R0020	1	0	1	short	Read-only
3	DSP alarm code	R0030	1	0x2000	0x6000	unsigned int	Read-only
4	Emergency stop mode	R0040	1	0	2	short	Read-only
5	DSP internal planning status	R0050	1	0	1	short	Read-only
6	J1 axis torque	R0101	0.10%	-5000	5000	float	Read-only

No.	Description	Parameter	Unit	Min. Value	Max. Value	Data Type	Change Method
	feedback						
12	J1 axis planned angle	R0102	°	-250	250	float	Read-only
18	J1 axis actual angle	R0103	°	-250	250	float	Read-only
24	J1 axis stop mode	R0104	1	0	9	short	Read-only
30	J1 axis servo control word	R0105	1	0	65535	unsigned short	Read-only
36	J1 axis servo status word	R0106	1	0	0xFFFF	unsigned short	Read-only
42	J1 axis servo position loop gain	R0107	0.1Hz	0	20000	unsigned short	Read-only
43	J1 axis servo speed loop gain	R0108	0.1Hz	1	20000	unsigned short	Read-only
44	J1 axis servo current loop gain	R0109	0.10%	0	2000	unsigned short	Read-only
7	J2 axis torque feedback	R0201	0.10%	-5000	5000	float	Read-only
13	J2 axis planned angle	R0202	°	-250	250	float	Read-only
19	J2 axis	R0203	°	-250	250	float	Read-only

No.	Description	Parameter	Unit	Min. Value	Max. Value	Data Type	Change Method
	actual angle						
25	J2 axis stop mode	R0204	1	0	9	short	Read-only
31	J2 axis servo control word	R0205	1	0	65535	unsigned short	Read-only
37	J2 axis servo status word	R0206	1	0	0xFFFF	unsigned short	Read-only
45	J2 axis servo position loop gain	R0207	0.1Hz	0	20000	unsigned short	Read-only
46	J2 axis servo speed loop gain	R0208	0.1Hz	1	20000	unsigned short	Read-only
47	J2 axis servo current loop gain	R0209	0.10%	0	2000	unsigned short	Read-only
8	J3 axis torque feedback	R0301	0.10%	-5000	5000	float	Read-only
14	J3 axis planned angle	R0302	°	-6000	360	float	Read-only
20	J3 axis actual angle	R0303	°	-6000	360	float	Read-only
26	J3 axis stop mode	R0304	1	0	9	short	Read-only
32	J3 axis	R0305	1	0	65535	unsigned	Read-only

No.	Description	Parameter	Unit	Min. Value	Max. Value	Data Type	Change Method
	servo control word					short	
38	J3 axis servo status word	R0306	1	0	0xFFFF	unsigned short	Read-only
48	J3 axis servo position loop gain	R0307	0.1Hz	0	20000	unsigned short	Read-only
49	J3 axis servo speed loop gain	R0308	0.1Hz	1	20000	unsigned short	Read-only
50	J3 axis servo current loop gain	R0309	0.10%	0	2000	unsigned short	Read-only
9	J4 axis torque feedback	R0401	0.10%	-5000	5000	float	Read-only
15	J4 axis planned angle	R0402	°	-540	540	float	Read-only
21	J4 axis actual angle	R0403	°	-540	540	float	Read-only
27	J4 axis stop mode	R0404	1	0	9	short	Read-only
33	J4 axis servo control word	R0405	1	0	65535	unsigned short	Read-only
39	J4 axis servo	R0406	1	0	0xFFFF	unsigned short	Read-only

No.	Description	Parameter	Unit	Min. Value	Max. Value	Data Type	Change Method
	status word						
51	J4 axis servo position loop gain	R0407	0.1Hz	0	20000	unsigned short	Read-only
52	J4 axis servo speed loop gain	R0408	0.1Hz	1	20000	unsigned short	Read-only
53	J4 axis servo current loop gain	R0409	0.10%	0	2000	unsigned short	Read-only
10	J5 axis torque feedback	R0501	0.10%	-5000	5000	float	Read-only
16	J5 axis planned angle	R0502	°	-180	180	float	Read-only
22	J5 axis actual angle	R0503	°	-180	180	float	Read-only
28	J5 axis stop mode	R0504	1	0	9	short	Read-only
34	J5 axis servo control word	R0505	1	0	65535	unsigned short	Read-only
40	J5 axis servo status word	R0506	1	0	0xFFFF	unsigned short	Read-only
54	J5 axis servo position	R0507	0.1Hz	0	20000	unsigned short	Read-only

No.	Description	Parameter	Unit	Min. Value	Max. Value	Data Type	Change Method
	loop gain						
55	J5 axis servo speed loop gain	R0508	0.1Hz	1	20000	unsigned short	Read-only
56	J5 axis servo current loop gain	R0509	0.10%	0	2000	unsigned short	Read-only
11	J6 axis torque feedback	R0601	0.10%	-5000	5000	float	Read-only
17	J6 axis planned angle	R0602	°	-540	540	float	Read-only
23	J6 axis actual angle	R0603	°	-540	540	float	Read-only
29	J6 axis stop mode	R0604	1	0	9	short	Read-only
35	J6 axis servo control word	R0605	1	0	65535	unsigned short	Read-only
41	J6 axis servo status word	R0606	1	0	0xFFFF	unsigned short	Read-only
57	J6 axis servo position loop gain	R0607	0.1Hz	0	20000	unsigned short	Read-only
58	J6 axis servo speed loop gain	R0608	0.1Hz	1	20000	unsigned short	Read-only

No.	Description	Parameter	Unit	Min. Value	Max. Value	Data Type	Change Method
59	J6 axis servo current loop gain	R0609	0.10%	0	2000	unsigned short	Read-only
60	Cartesian spatial position x	R1001	mm	-4000	4000	float	Read-only
61	Cartesian spatial position y	R1002	mm	-4000	4000	float	Read-only
62	Cartesian spatial position z	R1003	mm	-4000	4000	float	Read-only
63	Cartesian spatial orientation Rz	R1101	°	-180	180	float	Read-only
64	Cartesian spatial orientation Ry	R1102	°	-180	180	float	Read-only
65	Cartesian spatial orientation Rx	R1103	°	-180	180	float	Read-only
66	Cartesian spatial position speed Vx	R1201	mm/s	-4000	4000	float	Read-only
67	Cartesian spatial position speed Vy	R1202	mm/s	-4000	4000	float	Read-only
68	Cartesian spatial position	R1203	mm/s	-4000	4000	float	Read-only

No.	Description	Parameter	Unit	Min. Value	Max. Value	Data Type	Change Method
	speed Vz						
69	Cartesian spatial pose speed Wx	R1301	°/s	-100	100	float	Read-only
70	Cartesian spatial pose speed Wy	R1302	°/s	-100	100	float	Read-only
71	Cartesian spatial pose speed Wz	R1303	°/s	-100	100	float	Read-only
72	First 32 bits of common DI	R2001	1	0	0xffffffff	unsigned int	Read-only
73	Last 32 bits of common DI	R2002	1	0	0xffffffff	unsigned int	Read-only
74	First 32 bits of common DO	R3001	1	0	0xffffffff	unsigned int	Read-only
75	Last 32 bits of common DO	R3002	1	0	0xffffffff	unsigned int	Read-only
76	System DI	R2000	1	0	0xffff	unsigned short	Read-only
77	System DO	R3000	1	0	0xffff	unsigned short	Read-only

The DSP parameters are defined as follows:

There are 77 DSP parameters RXXXX, with R indicating robot, the first 8 digits of XXXX indicating item category, and the last 8 bits indicating function type. The item category is divided into four major categories: miscellaneous, axis (J1-J6), Cartesian, and DI/DO. The miscellaneous

(00) contains four function types such as DSP running line number (10), running state (20), etc. The axis (01-06) represents J1-J6 axes. The functions are subdivided into 9 function types such as torque feedback (01), planning angle (02), etc., Cartesian (10-13) and DI/DO (20,30) are shown in the table.

1.4.5) Servo commissioning function

1.4.5.1) Servo parameter list

Description:

Static attributes of the servo parameters and upload and download of current values.

After the servo parameters are uploaded, the background color is highlighted (gray) when the current value is different from the default value.

When downloading the servo parameters (not yet downloaded), the background color is highlighted (purple) if the current input value is different from the default value.

Parameter list:

Conforms to the parameter list in the general servo background software.

1.4.5.2) Speed JOG

(Specifications in this Section refer to SV660N Servo Drive Software Specification V1.2.docx)

Description: The robot jogs according to the preset speed. Verify that the motor rotates correctly.

Specifications:

1. The servo is automatically disabled if no operation is performed, protection time 2s.

2. Speed range: 0 to the maximum motor speed.

1.4.5.3) Bus motor parameters

(Specifications in this Section refer to SV660N Servo Drive Software Specification V1.2.docx)

Description:

1. Static attributes of the motor parameters and upload and download of current values.

2. After the servo parameters are uploaded, the background color is highlighted (gray) when the current value is different from the default value.

3. When downloading the servo parameters (not yet downloaded), the background color is highlighted (purple) if the current input value is different from the default value.

1.4.5.3) Mechanical characteristics analysis

(Specifications in this Section refer to SV660N Servo Drive Software Specification V1.2.docx)

Description:

Locate the mechanical resonance point by scanning the waveform.

Specifications:

1. The servo is automatically disabled if no operation is performed, protection time 2s.

2. Bode plot is output by background.

1.4.6) Running status monitoring

The following running status is displayed:

1. Servo alarm (error);

2. Servo drive status (ready, run, no ready);

3. Servo EtherCAT state machine (1, 2, 4, 8).

1.4.7) Log

The error log is saved to a local directory.

1.4.7) Fault Management

(Specifications in this Section refer to SV660N Servo Drive Software Specification V1.2.docx)

Description:

Reports current and historical equipment faults, gives all possible causes and solutions, and provides a reset function.

A total of 10 current and historical fault records of the servo, including the fault code, time stamp, fault name, the cause of the fault and the action to be taken.

Appendix 5: Simple Calculation of Load Parameters

When the load is a simple geometry, or is close to a simple geometry, its inertia parameters can be calculated using a simple scheme. The formula for calculating the inertia of common geometry is as follows.

Description	Moment of Inertia	Remarks
Thick cylinder open at both ends, with inner diameter r_1, outer diameter r_2, height h, mass m.	$I_z = \frac{1}{2}m(r_1^2 + r_2^2)$ $I_x = I_y = \frac{1}{12}m[3(r_1^2 + r_2^2) + h^2]$	
Solid cylinder with radius r, height h, and mass m.	$I_z = \frac{1}{2}mr^2$ $I_x = I_y = \frac{1}{12}m(3r^2 + h^2)$	The special case when the cylinder satisfies

		$r_1 = 0.$
Solid sphere with radius *r* and mass *m*.	$I = \frac{2}{5}mr^2$	
Solid cuboid with height *h*, width *w*, length *d*, and mass *m*.	$I_h = \frac{1}{12}m(w^2 + d^2)$ $I_w = \frac{1}{12}m(h^2 + d^2)$ $I_d = \frac{1}{12}m(w^2 + h^2)$	Moment of inertia of a cube with side length *s* $I_{CM} = \frac{ms^2}{6}$.
Thin rod with length *L* and mass *m*.	$I_{center} = \frac{mL^2}{12}$	The special case when the solid cuboid has w=L, h=d=0.